COURS IPV6

Point G6

But : Comprend le fonctionnement de IPv6

SALMON Nicolas

30/01/2010

Cours IPv6

Sommaire

I) Préambule

Dès le début des années 1990, l'évolution du réseau Internet semblait compromise à très court terme, car la conception du protocole IP (Internet Protocol) limitait le nombre d'équipements qui pouvaient s'y connecter. A l'origine, en 1973, ce réseau ne devait servir qu'à relier une centaine de machines. En fait, de nombreuses catégories d'utilisateurs sont très vite venues s'y joindre. Ce furent tout d'abord les scientifiques et les universitaires ; puis, en 1992, le réseau fut ouvert aux activités commerciales avec le succès que l'on sait. L'Internet n'avait pas été prévu pour supporter la croissance exponentielle du nombre d'équipements connectés. Le réseau a menacé d'atteindre la saturation et certains ont prédit son effondrement total en 1994. Comme toute prédiction de ce genre, elle s'est révélée fausse. En effet, dès 1993, des mesures d'urgence avaient été prises. Cela a permis de retarder l'échéance de quelques années. Les ingénieurs et chercheurs travaillant au sein de l'organisme de standardisation de l'Internet ont mis à profit ce délai pour concevoir une nouvelle version du protocole, s'affranchissant des limites imposées par l'actuelle version. Pour éviter toute confusion, la version initiale est désormais appelée IPv4. La version 5 ayant déjà été attribuée à un protocole expérimental, la version issue de ces travaux a été baptisée IPv6.

Ces travaux ont été l'occasion de spécifier les formats et mécanismes nécessaires pour prendre en compte les avancées issues des recherches sur les réseaux menées depuis 25 ans. Elles portent notamment sur l'auto-configuration, la mobilité, la diffusion multi-points, la sécurité (authentification de l'émetteur de l'information et chiffrement des données).

Les travaux principaux concernant IPv6 sont maintenant terminés. De nombreuses implantations sont disponibles aussi bien pour les équipements d'interconnexion que les ordinateurs. Les règles d'attribution des adresses IPv6 sont précisées et les opérateurs commencent à déployer des réseaux de production.

Cet ouvrage fait le point sur les travaux autour de la standardisation d'IPv6, sur ce qui peut être actuellement testé, les problèmes rencontrés au cours du développement, les pistes envisagées pour les résoudre et les sujets qui sont encore du domaine de la recherche. Il s'adresse aussi bien à des étudiants de troisième cycle qu'à des ingénieurs soucieux de préparer l'évolution de leurs réseaux. Cet ouvrage peut servir de référence sur cette nouvelle version du protocole IP en donnant de nombreux exemples tirés de cas réels.

Après une Introduction expliquant pourquoi le changement de protocole est devenu nécessaire et les principes fondamentaux qui ont été conservés dans IPv6, l'Adressage présente les différents types d'adresses (locale au lien, globale, multicast et anycast) et les plans d'adressage test et opérateur.

Le chapitre Protocoles réseau et transport décrit en détail la nouvelle pile de protocoles, le protocole ICMPv6, le protocole MLD utilisé pour la gestion locale des groupes multicast et les modifications à apporter aux protocoles de niveaux supérieurs.

Le chapitre Configuration automatique et contrôle traite des mécanismes de configuration automatique sans état et des mécanismes de découverte de voisins.

Le chapitre Nommage s'intéresse aux relations avec les mécanismes de haut niveau nécessaires pour faire la configuration automatique. En particulier les changements apportés au DNS pour prendre en compte les spécifications propres à IPv6.

Le chapitre Supports de transmission explique le transport d'IPv6 sur différents supports (Ethernet, LLP, PPP, tunnels et UMTS).

Le chapitre Installation d'un équipement détaille l'insertion des équipements dans un réseau IPv6. Il décrit comment activer et configurer la pile protocolaire des systèmes d'exploitation les plus répandus (Solaris, Windows, AIX, Linux, *BSD,...).

Le chapitre Routage est consacré aux routage. Il présente les différents protocoles utilisés pour IPv6 (RIPng, OSPF, IS-IS et BGP 4+). Le chapitre Configuration des routeurs donne des exemples de configuration des routeurs les plus couramment utilisés.

Le chapitre Multicast traite du multicast IPv6, définit plus en détail le format d'adresses et décrit les protocoles de routages utilisés.

Le chapitre Sécurité explique les mécanismes de sécurité définis pour IP. Il traite des différentes architectures et des échanges de clés.

Le chapitre Mobilité dans IPv6 traite ensuite des aspects liés à la mobilité.

Le chapitre Intégration d'IPv6 et des applications s'intéresse aux problèmes de transitions d'IPv4 vers IPv6. Il présente les différents mécanismes ainsi que des scénarios de déploiement.

Enfin, l'interface de programmation est présentée au chapitre Programmation d'applications (comment utiliser la résolution de nom dans un programme, comment programmer un serveur traitant à la fois des requêtes IPv4 et IPv6, comment programmer des applications réseau comme ping ou comment programmer des applications multicasts).

En annexe, le lecteur trouvera l'historique d'IPv6 depuis sa genèse, ainsi qu'un rappel du fonctionnement des instances de standardisation de l'Internet (appelé IETF), la bibliographie détaillée et la structure et l'utilisation des bases whois.

1) Le Groupe français d'expérimentateurs IPv6

L'idée du G6 est née d'une rencontre en novembre 1995 entre Alain Durand de l'IMAG (Institut d'Informatique et de Mathématiques Appliquées de Grenoble) et Bernard Tuy de l'UREC (Unité Réseau du CNRS) pour regrouper les actions de différents développeurs et expérimentateurs IPv6 en France. À cette époque, seuls quelques "illuminés" avaient entendu parler d'IPv6 mais, déjà, une implantation réalisée par Francis Dupont de l'INRIA (Institut National de Recherche en Informatique et en Automatique) était disponible.

Le groupe G6 s'est constitué avec des partenaires universitaires et industriels. Autour du noyau originel, se sont retrouvées des personnes venant des Universités de Bordeaux, Lille, Nantes, Paris, Strasbourg, des

Écoles Nationales Supérieures de Télécommunications de Bretagne et de Paris, de l'Institut Pasteur, de Bull, d'Alcatel et de 6Wind. Des réunions sont régulièrement organisées dans les différents lieux d'expérimentation.

Outre le partage d'expérience, la participation aux groupes de travail de l'IETF et la participation aux réunions RIPE (Réseaux IP Européens), le groupe s'est donné pour objectif de diffuser largement les connaissances acquises. Cet ouvrage en est la concrétisation majeure et de nombreux séminaires ont été organisés en France et en Europe. Un autre aspect très important des travaux du G6 est la mise en place du réseau G6bone pour relier en IPv6 les différents sites d'expérimentation. Ce réseau est bien sûr partie intégrante du réseau 6bone.

On trouvera plus d'informations sur http://www.g6.asso.fr/.

2) L'auteur

Au risque de décevoir ses admirateurs, Gisèle Cizault n'existe que dans l'esprit des membres de G6, qui regroupe les utilisateurs français d'IPv6. Les personnes qui ont contribué à ce livre sont, par ordre alphabétique :

Yann Adam (France Télécom R&D), Pascal Anelli (LIM/ université de la Réunion),
Alain Baudot (France Télécom R&D),
Philippe Bereski (Alcatel),
Jean-Marie Bonnin (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Julien Bournelle (GET/INT, département Logiciels-Réseaux)
Benoît Brodard (INRIA Sophia Antipolis à l'époque de la rédaction de ce livre),
Claude Castelluccia (INRIA Rhône-Alpes),
Isabelle Chrisment (LORIA / Université de Nancy II),
Luis H. M. K. Costa (Laboratoire d'Informatique de Paris 6),
Bernard Cousin (IRISA / université de Rennes 1),
Francis Dupont (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Yann Dupont (CRI Université de Nantes),
Alain Durand (Comcast),
Jérôme Durand (Renater),
Thierry Ernst (Wide project),
Olivier Festor (LORIA / INRIA Lorraine),
Jean-Olivier Gerphagnon (BULL),
Frédéric Gloppe (BULL à l'époque de la rédaction de ce livre),
Ibrahim Hajjeh (GET/ENST),
Martin Heusse (IMAG-LSR, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, Laboratoire Logiciels Systèmes Réseaux),
Mickael Hoerdt (Laboratoire des Sciences de l'Image de l'Informatique et de la Télédétection, Université de Strasbourg - Trondheim/NTNU),
Christophe Janneteau (Motorola),

Konstantin Kabassanov (Laboratoire d'Informatique de Paris 6),
Ghislaine Labouret (HSC, Hervé Schauer Consultants),
Arthur Lallet (Motorola),
Maryline Laurent-maknavicius (GET/INT, département Logiciels-Réseaux),
Yves Legrandgérard (Laboratoire Preuves, Programmes et Systemes - CNRS UMR 7126),
Aimé Le Rouzic (BULL),
Vincent Levigneron (AFNIC),
Emmanuel Lochin (Laboratoire d'Informatique de Paris 6),
Philippe Lubrano (AFNIC),
Jérôme Marchand (Artesys International),
Octavio Medina (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Ana Minaburo (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Simon Muyal (Renater),
Thomas Noël (Laboratoire des Sciences de l'Image de l'Informatique et de la Télédétection, Université de Strasbourg),
Alexandru Petrescu (Motorola),
Bernard Phan Dinh Tuy (CNRS/UREC),
Yanick Pouffary (HP),
David Ranch (Juniper Network),
Jean-Luc Richier (IMAG-LSR, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, Laboratoire Logiciels Systèmes Réseaux),
Emmanuel Riou (Motorola),
Ollivier Robert (Eurocontrol),
Vincent Roca (Laboratoire d'Informatique de Paris 6),
Jean-Pierre Roch (BULL),
Imed Romdhani (Napier University, Edinburgh, UK)
Olivier Salaün
Luc Saccavini (INRIA),
Mohsen Souissi (AFNIC),
Bruno Stévant (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Laurent Toutain (GET/ENST Bretagne, Département Réseaux, Sécurité et Multimédia),
Jean-Marc Uzé (Juniper Network),
Rolland Vida (Laboratoire d'Informatique de Paris 6).

Ont participé à cette quatrième édition : Yann Adam, Alain Baudot, Philippe Bereski, Jean-Marie Bonnin, Julien Bournelle, Bernard Cousin, Jérôme Durand, Thierry Ernst, Ibrahim Hajjeh, Martin Heusse, Mickael Hoerdt, Christophe Janneteau, Konstantin Kabassanov, Arthur Lallet, Maryline Laurent-maknavicius, Yves Legrandgérard, Octavio Medina, Ana Minaburo, Simon Muyal, Alexandru Petrescu, Bernard Phan Dinh Tuy, Jean-Luc Richier (éditeur), Emmanuel Riou, Imed Romdhani, Luc Saccavini, Bruno Stévant, Mohsen Souissi, Laurent Toutain (éditeur).

Nos remerciements vont à toutes les personnes qui nous ont aidés à réaliser cet ouvrage :

Jean-Luc Archimbaud,
Bob Fink

Philippe Girault,
Denis Joiret,
Mohamed Kassi-Lahlou,
Daniel Kofman,
Jean Yves Leboudec
Philippe Queinnec,
Rob Romano
Ahmed Serhouchni,
Philippe Sonntag,
Lionel Thual,
Hervé Troadec,
Yves et Micheline Troadec.

II) Introduction

1) *Principes fondamentaux d'IP*

L'Internet connaît un succès très important, bien au-delà des prévisions les plus optimistes faites à l'époque de sa conception. Les raisons de ce succès sont multiples ; on peut cependant essayer d'en dégager quelques-unes, tenant aux caractéristiques fondamentales de l'Internet et à l'architecture du protocole de communication IP (*Internet Protocol*) utilisé.

L'Internet est bâti sur un modèle de réseau de réseaux. Son nom vient d'ailleurs de l'anglais *Inter Networking*. On ne fait aucune hypothèse sur le type d'infrastructure ou d'équipement utilisés. Les extrémités, ou points de connexion aux réseaux, sont des objets capables de traitements évolués. Les données sont véhiculées dans des datagrammes séparés (que l'on appelle communément paquets). À partir de ces prémisses, les architectes de l'Internet ont retenu deux principes fondamentaux : la communication de "bout en bout" et le "meilleur effort" (*Best Effort*) pour l'acheminement.

Le principe du "bout en bout" implique que les partenaires d'une communication dialoguent depuis chaque extrémité du réseau pour établir et gérer leur communication. Les éléments intermédiaires sont transparents et n'interviennent pas dans le dialogue. Les objets d'extrémité étant "intelligents", ils sont à même de prendre les décisions nécessaires. Il n'y a pas de position intrinsèquement privilégiée sur le réseau : chaque ordinateur connecté à l'Internet a les mêmes potentialités. Ceci permet aussi une extension du modèle client/serveur : un serveur n'est plus forcément lié à un équipement particulier puisque tout ordinateur connecté à l'Internet peut devenir serveur et n'importe quel autre ordinateur peut en devenir client. C'est cette caractéristique qui a rendu possible la prolifération des serveurs Web dans le monde entier. N'importe qui possédant un ordinateur connecté à l'Internet peut en effet installer son propre serveur Web. Elle est également fondamentale pour les applications peer to peer.

Le principe du "meilleur effort" dit que les éléments d'interconnexion n'offrent aucune garantie sur l'acheminement des données. Ils se contentent de faire "de leur mieux" pour les acheminer. Par exemple,

les paquets de données peuvent ne pas emprunter deux fois de suite le même chemin, certains peuvent se perdre en route, d'autres arriver dans le désordre, même si cela est relativement rare dans l'Internet...

C'est ce principe qui fait dire qu'IP n'est pas un protocole "fiable" mais par contre "robuste". Il n'est pas "fiable" au sens où l'arrivée des données envoyées n'est pas garantie. Les réseaux, ainsi libérés d'une tâche très complexe, peuvent dynamiquement se reconfigurer en cas de panne d'une liaison ou d'un équipement. Les protocoles de niveau transport comme TCP se chargent de la gestion des réémissions des données perdues et du réassemblage de celles arrivées dans le désordre ; ils fournissent ainsi la "fiabilité" du service.

Ces deux principes permettent de s'affranchir des différences entre les supports et entre matériels d'interconnexion utilisés. Ce sont eux qui ont rendu possible la croissance de l'Internet que l'on connaît aujourd'hui. Cette croissance est maintenant freinée par deux problèmes majeurs : le manque d'adresses disponibles et la stabilité due à la taille des tables de routage des équipements d'interconnexion des opérateurs réseaux.

Les ordinateurs sont identifiés dans l'Internet par des adresses IP uniques. Le principe du datagramme impose que l'adresse de destination se retrouve dans l'ensemble des paquets émis sur le réseau. Pour permettre un traitement très rapide, les routeurs doivent trouver rapidement cette adresse. Dans IPv4, ces adresses sont codées dans un mot binaire de 32 bits et se retrouvent toujours à la même place dans l'en-tête. Ce principe d'ingénierie a montré son efficacité puisqu'il permet de construire des équipements d'interconnexion simple traitant un nombre important de paquets à la seconde.

Une adresse codée sur 32 bits permet théoriquement d'adresser 2^32 machines, soit à peu près 4 milliards. Ce nombre pourrait paraître au premier abord très élevé, mais les ordinateurs ne sont pas numérotés séquentiellement. Ils sont regroupés par réseaux. À chaque réseau est affecté un numéro qui est codé sur une partie des 32 bit de l'adresse des machines. On s'aperçoit alors que le nombre de réseaux disponibles n'est pas si important que cela conduit à une pénurie. La tendance actuelle consiste à freiner au maximum l'allocation des adresses réseaux. Ce n'est pas un problème pour les sites déjà équipés disposant déjà de larges plages d'adresses. Cette contrainte est déjà forte pour les nouveaux sites dans les pays dits "développés" pour lesquels un grand nombre d'adresses a été réservé mais se révèle être un problème majeur pour les pays émergeants où parfois moins de 10 réseaux de 250 machines ont été attribués pour l'ensemble du pays.

Les équipements d'interconnexion des réseaux, orientant les paquets vers leur destination finale, sont des routeurs. Pour prendre leurs décisions, ils consultent une table dite de routage. Le nombre de réseaux dans l'Internet croissant de manière vertigineuse, ces tables de routage deviennent de plus en plus volumineuses et difficiles à maintenir. Pour pallier ce problème, une solution d'adressage hiérarchique permettent de réunir un ensemble de numéros de réseaux contigus en un seul préfixe a été conçue (CIDR : *Classeless Inter Domain Routing*). En plus de la réduction des tables de routage, CIDR permet aussi de réduire la sur-allocation d'adresses aux sites terminaux, réduisant quelque peu la pénurie d'adresses. Avec CIDR le propriétaire de l'adresse est modifié. Dans les plans d'adressage initiaux, le site était propriétaire de son préfixe, avec CIDR le préfixe devient la propriété de son opérateur, rendant la renumérotation du réseau nécessaire, si le site change d'opérateur. Cet adressage hiérarchique a montré son efficacité opérationnelle et les règles d'adressage actuelles pour IPv6 généralisent ce principe.

Un autre palliatif à la pénurie est le recours à la traduction d'adresses (NAT : *Network Address Translator*) utilisant des adresses "privées" à l'intérieur d'un site. Ces adresses ne permettent pas de communiquer directement avec une machine connectée à l'Internet. Les communications avec l'extérieur étant quand même nécessaires, on a recours à un artifice pour les réaliser : le routeur de sortie de site "convertit" toutes les adresses privées en une ou plusieurs adresses officielles. Ce routeur établit donc les communications en lieu et place des machines internes au site.

Un tel mécanisme ne nécessite que quelques adresses IP officielles pour l'ensemble d'un site pouvant contenir plusieurs milliers de machines. Cette approche est une violation manifeste du principe de connexion de bout en bout. Elle est suffisante pour utiliser des applications simples comme l'accès au Web mais pénalise lourdement la mise en place de nombreuses autres applications. De plus, elle interdit la mise en œuvre de solutions à forte sécurité basées sur la cryptographie.

En résumé, ces mécanismes provisoires figent le réseau pour une utilisation dans un mode dit client/serveur. Les clients sont à l'intérieur de l'entreprise dans un Internet "privé" et les serveurs sont à l'extérieur sur l'Internet "public". Or ce paradigme est remis en cause par de nouvelles applications, comme le fax et la téléphonie sur Internet où chaque équipement doit être autorisé à recevoir des appels. De même pour les particuliers, les jeux répartis en réseau sont promis à un succès certain. Or, ils ne fonctionnent pas avec des mécanismes de traduction d'adresses, car les applications doivent s'échanger leur adresses.

Le succès d'un réseau n'est pas uniquement lié aux bons choix technologiques adoptés lors de la conception du protocole. Il est aussi lié aux services et aux applications disponibles sur ce réseau. IP joue ce rôle unificateur à la frontière entre des supports de transmission et des applications. Plus qu'une indépendance entre l'application et le médium, le réseau permet aux applications de communiquer entre les différents médias. Le risque à maintenir trop longtemps des adressages privés est de rompre cette communication entre différents mondes, créant la richesse du réseau. Elle pourrait même conduire chaque monde à développer un protocole réseau plus adapté à son besoin propre au détriment de l'interconnectivité.

Il était devenu impératif de s'attaquer simultanément à ces deux problèmes d'épuisement des adresses disponibles et d'explosion des tables de routage en s'appuyant sur les principes fondamentaux qui ont fait la réussite de l'Internet. C'est à cette tâche que depuis 1992 s'est attelé l'IETF (Internet Engineering Task Force), l'organisme de standardisation de l'Internet, pour définir le protocole IPv6.

Après plus de 10 ans d'efforts de standardisation, les spécifications de base du protocole et les règles d'attribution des adresses sont clairement définies. La plupart des routeurs et des systèmes d'exploitation incluent cette nouvelle version du protocole IP. Il reste maintenant à faire sortir IPv6 des laboratoires et des plates-formes d'expérimentation, à assurer l'interopérabilité avec IPv4 quand cela est nécessaire et à développer de nouvelles applications profitant de cette espace d'adressage quasi-illimité qu'offre IPv6. Un des défis dans les années à venir est d'utiliser IPv6 dans des domaines jusque là ignorés des réseaux (audio-visuel, domotique, automobile,...).

III) Adressage

1) Généralités

Le format et la représentation des adresses sont les modifications les plus visibles pour l'utilisateur expérimenté et l'ingénieur réseau dans cette nouvelle version du protocole. Même si les principes sont fortement similaires à ceux employés dans IPv4, cet adressage apparaît beaucoup plus complexe. Il est intéressant d'en comprendre le principe et les règles d'attribution avant d'aborder les aspects protocolaires.

Une adresse IPv6 est un mot de 128 bits. La taille d'une adresse IPv6 est le quadruple de celle d'une adresse IPv4. En prenant en compte les estimations les plus pessimistes et les plus optimistes [BM95], on obtient l'encadrement suivant où l'unité est le nombre d'adresses par mètre carré de surface terrestre (océans compris) :

1564 < *adresses disponibles* < 3911873538269506102

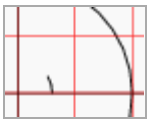


Figure :

D'autres calculs indiquent que l'on pourrait potentiellement attribuer 60 000 milliards de milliards d'adresses par habitant.

Ces calculs sont bien entendu complètement arbitraires. Il est difficile de prévoir l'utilisation des adresses dans les années futures. Ainsi, par exemple, le plan d'adressage actuellement mis en œuvre utilise un identifiant d'équipement sur 64 bits, c'est-à-dire la moitié de la taille de l'adresse. En fait ce genre de calcul sert de justification aux partisans des adresses de taille fixe (ce qui simplifie le traitement des paquets) en montrant que le nombre d'équipements adressables est colossal.

Il ne faut toutefois pas faire un contre-sens sur l'interprétation de ces calculs. Le but d'IPv6 n'est pas d'attribuer une fois pour toute une adresse IPv6 à un équipement (ou à un être humain). Une adresse IPv6 n'a de sens et d'utilité que lors que l'équipement est connecté sur le réseau. De plus si l'emplacement sur le réseau de cet équipement change, l'adresse devra également être modifiée pour refléter ce déplacement.

Ce chapitre, après avoir défini ce que l'on attend d'une adresse dans l'Internet, passe en revue les différents types d'adresses. Il explique en détail le plan d'adressage agrégé qui a été retenu pour construire les réseaux de tests et opérationnels. Il décrit également la manière de calculer les identifiants d'interface utilisé par plusieurs types d'adresses (voir également Supports de transmission).

A) Qu'est-ce qu'une adresse ?

La distinction entre les notions d'annuaires, de noms, d'adresses et de routes est comprise depuis longtemps. Cependant, depuis quelques années, au sein de l'Internet, la compréhension du rôle d'une adresse réseau a évolué. Dans l'Internet, une adresse sert en fait à deux fonctions distinctes : identification et localisation.

- L'identification est utilisée pendant une connexion par chacun des intervenants pour reconnaître son interlocuteur. Cela permet entre autres de s'assurer de l'origine des paquets reçus. Cette vérification se fait dans les pseudo-en-têtes TCP ou dans les associations de sécurité d'IPsec. La durée de vie minimale d'un identificateur est celle d'une connexion TCP.
- La localisation est utilisée pour trouver un intermédiaire qui saura délivrer les paquets. La durée de vie de la fonction de localisation est assez grande. En fait, elle ne varie qu'en cas de changement de prestataire IP ou de réorganisation du site.
  En général la localisation est découpée en deux parties : localisation globale (identifiant le réseau) et locale, distinguant les machines sur un même réseau. La localisation est intrinsèquement liée aux fonctions de routage d'IP.

En IPv4, on confond identification et localisation en une seule entité, l'adresse IP, globalement unique dans l'Internet. Cette construction a un prix : lors de la renumérotation d'un site, ou lorsqu'un mobile se déplace, la localisation change. Avec l'approche IPv4, l'adresse IP change, et donc l'identification... Cela implique une perte ou au mieux une renégociation des communications en cours.

Lors des études initiales pour IPv6, il a été proposé de séparer ces deux fonctions pour pouvoir résoudre simplement les problèmes de renumérotation, mobilité, multi-domiciliation... Ceci est encore un sujet de recherche. Cette proposition n'a donc pas été retenue ; en IPv6 comme en IPv4, l'adresse sert à la fois pour l'identification et la localisation. En effet, le plan d'adressage choisi dans un premier temps est une extension des règles d'adressage hiérarchiques (CIDR) utilisées dans IPv4.

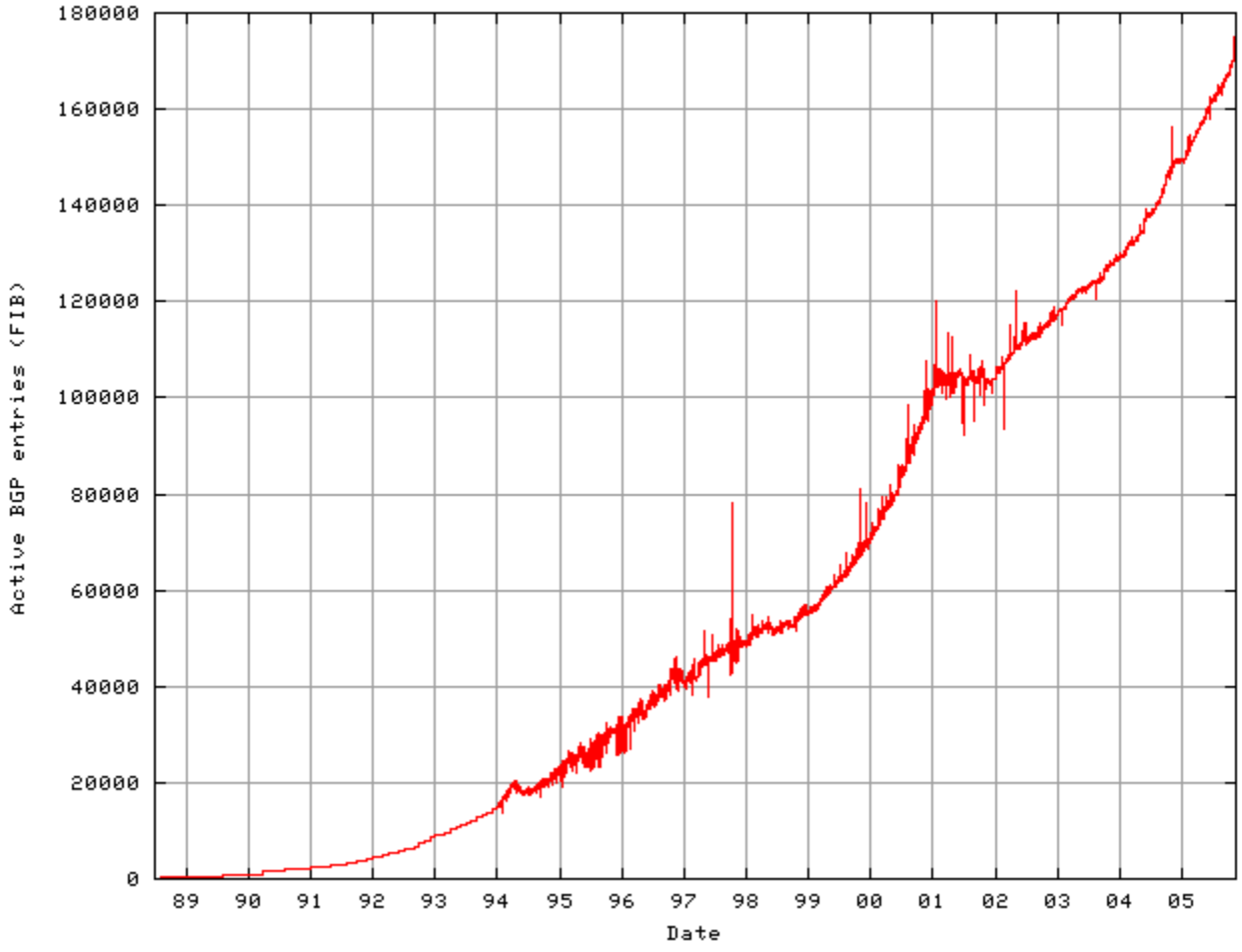
Un autre débat est de savoir si une adresse identifie une machine ou une interface. Cette distinction n'est pas très importante dans le cas d'une machine simple ne possédant qu'une seule interface ; elle le devient dans le cas où elle en possède plusieurs ou est multi-domiciliée. En effet pour essayer de simplifier les tables de routage dans le cœur de réseau, si un site est connecté à plusieurs fournisseurs d'accès, il possédera autant de préfixes IPv6 que de fournisseurs. Contrairement à IPv4, où l'on associe généralement qu'une seule adresse à une interface, une interface possède plusieurs adresses IPv6.

B) Structuration des adresses et agrégation

Un des problèmes majeurs d'IPv4 est la croissance incontrôlée des tables de routage. Ce phénomène est dû à une mauvaise agrégation des adresses dans les tables. Il faudrait être capable de router des ensembles de réseaux identifiés par un seul descripteur. CIDR apporte une amélioration, mais celle-ci est insuffisante en pratique : les adresses IPv4 sont trop courtes pour permettre une bonne structuration, et il faut surtout assumer le coût du passé avec les adresses déjà allouées.

Attribuer une adresse à un équipement est un processus complexe, basé sur un compromis entre la facilité d'attribution et la facilité de gestion. Idéalement, pour minimiser la taille de routage, le réseau devrait avoir une topologie en arbre, cela rendrait l'adressage hiérarchique très efficace. Dans la réalité pour des raisons économiques, techniques, géographiques ou de performances, le réseau est beaucoup plus complexe et peut être vu comme un graphe. Il faut introduire des exceptions dans les tables de routages pour refléter cette topologie. On voit que pour avoir l'adressage le plus efficace possible, il faut dans ce graphe trouver la représentation arborescente qui génère le moins d'exceptions possibles. Or s'il était possible aujourd'hui de trouver une représentation valide, elle ne le sera pas nécessairement demain. En conséquence, la définition du plan d'adressage doit être la plus souple possible pour permettre une évolution de nature imprévisible.

D'autant plus que l'agrégation pour IPv4 ne semble plus aussi efficace. La figure suivante donne l'évolution de table de routage dans le cœur de l'Internet, c'est-à-dire dans les réseaux des opérateurs où aucune route par défaut n'est définie.



Evolution de la taille des tables de routage
Source: http://www.cidr-report.org

En 2000, la progression linéaire de cette table a semblé compromise du fait :

- de la baisse du coût des liaisons longues distances, permettant une multi-domiciliation (*multi-homing*) des sites pour des raisons de fiabilité (en cas de panne d'un opérateur, le trafic pourra passer par un autre), de performances (aller directement sur le réseau avec lequel le site à un trafic important),
- le manque d'adresses IPv4 qui force les opérateurs à allouer des préfixes de plus en plus long.

Depuis, les opérateurs ont fortement agrégés pour revenir à une progression linéaire de la table. Des études [BTC02] montrent que :

- la multi-domiciliation, c'est-à-dire la connexion d'un site à plusieurs opérateurs pour fiabiliser l'accès, ajoute un surcoût de 20 à 30 pourcent,
- le partage de charge, c'est-à-dire réduire l'agrégation pour annoncer un sous-ensemble de préfixe à chaque opérateur, induit un surcoût de 20 à 25 pourcent,
- de mauvaises règles d'agrégation induisent une surcharge de 15 à 20 pourcent,
- la fragmentation de l'espace d'adressage liée à la gestion des préfixes avant CIDR, et à l'allocation séquentielle des blocks d'adresses contribue à plus de 75 pourcent de la taille de la table.

Actuellement, pour pouvoir assurer une bonne agrégation, les règles utilisées par CIDR pour IPv4 sont conservées. Mais la gestion des tables de routage dans le cœur du réseau s'en trouvera quand même améliorée car :

- dès le début le plan d'adressage est hiérarchique, éliminant les longs préfixes,
- les sites multi-domiciliés posséderont autant d'adresses que de fournisseurs, permettant ainsi de garantir une agrégation.
- des mécanismes de renumérotation automatiques permettent aux sites de changer facilement de préfixe quand cela est nécessaire.

Si un plan d'adressage hiérarchique semble actuellement le plus adapté, d'autres règles de numérotation pourraient être utilisées dans le futur, comme par exemple, les coordonnées géographiques de l'équipement. Ces techniques ne sont actuellement utilisées que dans quelques laboratoires de recherche pour des réseaux ad hoc, mais il reste assez de place dans l'espace d'adressage pour prendre en compte ces nouveaux types de réseaux si un jour ils se généralisent.

Le choix d'un plan d'adressage a fait l'objet de nombreux débats à l'IETF. Il a été beaucoup plus difficile à définir que le format du paquet IPv6 présenté au chapitre suivant. Plusieurs plans ont été proposés puis abandonnés. Ces divers plans d'adressages sont commentés dans le chapitre sur l'Historique de la standardisation d'IPv6. Le présent chapitre se contente de décrire les différents plans d'adressage actuellement utilisés.

2) Plans d'adressage

A) Durée de vie des adresses

IPv6 généralisant le plan d'adressage CIDR, les préfixes restent dans tous les cas la propriété des opérateurs. Ils ne peuvent plus être attribués "à vie" aux équipements. Pour faciliter la renumérotation d'une machine l'attribution d'une adresse à une interface est faites temporairement, les adresses IPv6 ne sont pas données mais prêtées. Une durée de vie est associée à l'adresse qui indique le temps pendant lequel l'adresse appartient à l'interface. Quand la durée de vie est épuisée, l'adresse devient invalide, elle est supprimée de l'interface et devient potentiellement assignable à une autre interface. Une adresse invalide ne doit jamais être utilisée comme adresse dans des communications. La valeur par défaut de la durée de vie d'une adresse est de 30 jours, mais cette durée peut être prolongée, ou portée à l'infini. L'adresse lien-local a une durée de vie illimitée.

La renumérotation d'une interface d'une machine consiste à passer d'une adresse à une autre. Lors d'une renumérotation, il n'est pas souhaitable de changer brusquement d'adresse, sinon toutes les communications TCP, qui l'utilisent comme identificateur de connexion, seraient immédiatement coupées. Ceci entraînerait des perturbations importantes au niveau des applications.

Pour faciliter cette transition, un mécanisme d'obsolescence est donc mis en place pour invalider progressivement une adresse. Ce mécanisme s'appuie sur la capacité d'affectation de plusieurs adresses valides à une même interface. Ensuite pour effectuer le choix de l'adresse à utiliser, un état est associé. Il indique dans quelle phase de sa durée de vie une adresse se situe vis à vis de l'interface. Le premier de ces états est qualifié de préféré : l'utilisation n'est aucunement restreinte. Peu avant son invalidation l'adresse passe dans un état de déprécié. Dans cet état, l'utilisation de l'adresse est déconseillée, mais pas interdite. L'adresse dépréciée ne doit plus être utilisée comme adresse de source pour les nouvelles communications (comme l'établissement de connexion TCP). Par contre l'adresse dépréciée peut encore servir d'adresse de source dans le cas des communications existantes. Les paquets reçus à une adresse dépréciée continuent à être remis normalement. À la durée de vie de validité d'une adresse, il est également associé une durée de vie pour son état préféré. La figure 3-2 représente les différents états que prend une adresse lorsqu'elle est allouée à une interface.

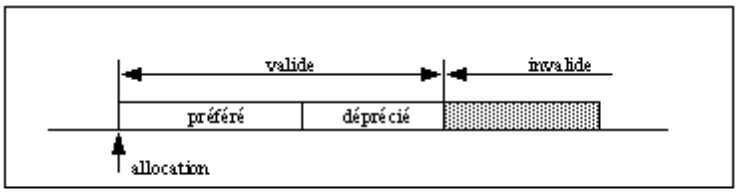


Figure 3-2 . Etats successifs d'une adresse sur une interface.

B) Notation

La représentation textuelle d'une adresse IPv6 se fait en découpant le mot de 128 bits de l'adresse en 8 mots de 16 bits séparés par le caractère «:», chacun d'eux étant représenté en hexadécimal. Par exemple :

`FEDC:BA98:7654:3210:EDBC:A987:6543:210F`

Dans un champ, il n'est pas nécessaire d'écrire les zéros placés en tête :

`FEDC:0:0:0:400:A987:6543:210F`

En outre plusieurs champs nuls consécutifs peuvent être abrégés par «::». Ainsi l'adresse précédente peut s'écrire comme suit :

`FEDC::400:A987:6543:210F`

Naturellement, pour éviter toute ambiguïté, l'abréviation «::» ne peut apparaître qu'une fois au plus dans une adresse.

La représentation des préfixes IPv6 est similaire à la notation CIDR RFC 1519 utilisée pour les préfixes IPv4. Un préfixe IPv6 est donc représenté par la notation :

adresse-ipv6/longueur-du-préfixe-en-bits

Les formes abrégées avec «::» sont autorisées.

`3EDC:BA98:7654:3210:0000:0000:0000:0000/64`

`3EDC:BA98:7654:3210:0:0:0:0/64`

`3EDC:BA98:7654:3210::/64`

Le seul piège de cette notation vient des longueurs de préfixes qui ne sont pas en frontière de «:». Ainsi le préfixe `3EDC:BA98:7654:3::/56` équivaut en réalité à `3EDC:BA98:7654:0000::/56` car il s'écrit `3EDC:BA98:7654:0003::/56`.

On peut combiner l'adresse d'une interface et la longueur du préfixe réseau associé en une seule notation.

`3EDC:BA98:7654:3210:945:1321:ABA8:F4E2/64`

Ces représentations peuvent apparaître beaucoup plus complexes qu'avec IPv4, mais leur attribution répond à des règles strictes, ce qui favorise leur mémorisation. De plus, les fonctions d'auto-configuration font qu'il est très rare, même pour un ingénieur réseau, de les manipuler.

Il est pourtant parfois nécessaire de manipuler littéralement des adresses IPv6. Le caractère ":" utilisé pour séparer les mots peut créer des ambiguïtés. C'est le cas avec les URL où il est aussi utilisé pour indiquer le numéro de port. Ainsi l'URL

`http://2001:1234:12::1:8000/`

peut aussi bien indiquer le port 8000 sur la machine ayant l'adresse IPv6 2001:1234:12::1, que la machine 2001:1234:12::1:8000 en utilisant le port par défaut. Pour lever cette ambiguïté, le RFC 2732 propose d'inclure l'adresse IPv6 entre "[ ]". L'adresse précédente s'écrirait :

```
http://[2001:1234:12::1]:8000/
```

ou

```
http://[2001:1234:12::1:8000]/
```

suivant les cas. Cette représentation peut être étendue à d'autres domaines comme X-window ou au protocole de signalisation téléphonique SIP.

C) Type des adresses

IPv6 reconnaît trois types d'adresses : unicast, multicast et anycast.

Le premier de ces types, le type unicast, est le plus simple. Une adresse de ce type désigne une interface unique. Un paquet envoyé à une telle adresse, sera donc remis à l'interface ainsi identifiée.

Parmi les adresses unicast, on peut distinguer celles qui auront une portée globale, c'est-à-dire désignant sans ambiguïté une machine sur le réseau Internet et celles qui auront une portée locale (lien ou site). Ces dernières ne pourront pas être routées sur l'Internet.

Une adresse de type multicast désigne un groupe d'interfaces qui en général appartiennent à des nœuds différents pouvant être situés n'importe où dans l'Internet. Lorsqu'un paquet a pour destination une adresse de type multicast, il est acheminé par le réseau à toutes les interfaces membres de ce groupe. Il faut noter qu'il n'y a plus d'adresses de type broadcast comme sous IPv4 ; elles sont remplacées par des adresses de type multicast qui saturent moins un réseau local constitué de commutateurs. L'absence de broadcast augmente la résistance au facteur d'échelle d'IPv6 dans les réseaux commutés.

Le dernier type, anycast, est une officialisation de propositions faites pour IPv4 RFC 1546. Comme dans le cas du multicast, une adresse de type anycast désigne un groupe d'interfaces, la différence étant que lorsqu'un paquet a pour destination une telle adresse, il est acheminé à un des éléments du groupe et non pas à tous. C'est, par exemple, le plus proche au sens de la métrique des protocoles de routage. Cet adressage est principalement expérimental, voir Adresses anycast.

Certains types d'adresses sont caractérisés par leur préfixe RFC 3513. Le tableau suivant (source : http://www.iana.org/assignments/ipv6-address-space) donne la liste de ces préfixes. La plage «réservée» du préfixe 0::/8 est utilisée pour les adresses spéciales (adresse indéterminée, de bouclage, mappée, compatible). On notera que plus de 70% de l'espace disponible n'a pas été alloué, ce qui permet de conserver toute latitude pour l'avenir.

Préfixe IPv6	Allouer	Référence
0000::/8	Réservé pour la transition et loopback	RFC 3513
0100::/8	Réservé	RFC 3513
0200::/7	Réservé (ex NSAP)	RFC 4048
0400::/6	Réservé (ex IPX)	RFC 3513
0800::/5	Réservé	RFC 3513
1000::/4	Réservé	RFC 3513
2000::/3	Unicast Global	RFC 3513
4000::/3	Réservé	RFC 3513
6000::/3	Réservé	RFC 3513
8000::/3	Réservé	RFC 3513
A000::/3	Réservé	RFC 3513
C000::/3	Réservé	RFC 3513
E000::/4	Réservé	RFC 3513
F000::/5	Réservé	RFC 3513
F800::/6	Réservé	RFC 3513
FC00::/7	Unique Local Unicast	RFC 4193
FE00::/9	Réservé	RFC 3513
FE80::/10	Lien-local	RFC 3513
FEC0::/10	Réservé	RFC 3879
FF00::/8	Multicast	RFC 3513

Dans un premier temps, des adresses du type site-local dans l'espace FEC0::/10 avaient été définies par l'IETF, mais elles ont été retirées dans les dernières versions des standards.

3) Adresses unicast Globales

A) Adressage global : plan d'adressage agrégé

Ce plan, proposée dans le RFC 3587, précise la structure d'adressage IPv6 définie dans le RFC 3513 en précisant les tailles de chacun des blocs. Une adresse intègre trois niveaux de hiérarchie :

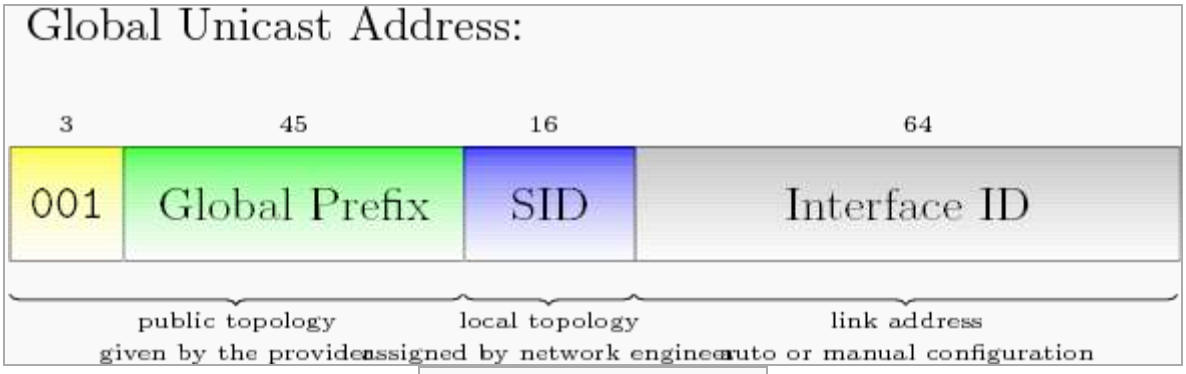


Figure : Adresses Globales

- ✓ Une topologie publique codée sur 48 bits, allouée par le fournisseur d'accès;
- ✓ Une topologie de site codé sur 16 bits. Ce champ permet de coder les numéros de sous réseau du site;
- ✓ Un identifiant d'interface (64 bits) distinguant les différentes machines sur le lien.

Il existe plusieurs instanciations de ce plan d'adressage. Historiquement la première (préfixe `3FFE::/16`) a servi aux réseaux expérimentaux, puis une seconde (préfixe `2001::/16`) est définie par les autorités régionales pour les réseaux dits de production, enfin une troisième est dédiée (préfixe `2002::/16`) au mécanisme de transition 6to4. Ces instanciations sont différenciées par la valeur du préfixe initial de 16 bits (cf. Tableau). Très récemment, d'autres préfixes ont été libérés. En effet, si l'on garde l'attribution de préfixe de longueur 48 pour les sites terminaux, et que l'on intègre les réseaux domotiques, les opérateurs peuvent justifier d'un besoin important d'adresses que les autorités régionales ne peuvent leur refuser. Il semble que cela pourrait remettre en cause l'attribution de préfixes de longueur 48 pour tous les utilisateurs au profit de préfixes plus long. Une version, à jour des allocations est disponible sur le site http://www.iana.org/assignments/ipv6-unicast-address-assignments.

B) Adresses de test

Les expérimentations d'IPv6 sur un réseau devaient commencer avant que les règles d'attribution des préfixes soient complètement finalisées. La valeur 0x1FFE a été attribuée par l'IANA au 6bone dans le plan d'adressage agrégé pour les expérimentations (RFC 3701). Il correspond au préfixe `3FFE::/16` pour l'ensemble du 6bone.

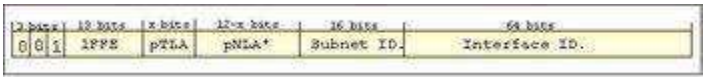


Figure 3-4 Adresse de test du plan agrégé

Les 48 bits restant avant le champ Subnet ID recréent les niveaux hiérarchiques d'un réseau IPv6 défini dans le RFC 3587, d'où le terme pseudo accolé au nom du champ. La taille réduite n'étant pas un facteur limitant dans la phase expérimentale. Des pseudo-TLA d'une taille initialement de 8, mais portées à 12 bits par la suite, sont attribués à des opérateurs voulant expérimenter le protocole. Les 24 ou 20 bits suivants sont utilisés pour numéroter les sites.

Les pseudo-TLA ont été alloués jusqu'en décembre 2003 aux opérateurs qui en faisaient la demande. La liste complète est disponible sur le serveur Web http://www.6bone.net/6bone_pTLA_list.html. Le tableau Pseudo TLA attribués par le groupe ngtrans reprend quelques unes de ces valeurs.

Pseudo TLA attribués par le groupe ngtrans

Organismes/Pays	Préfixe	Organismes/Pays	Préfixe
ROOT66/US-CA	3FFE:0000::/24	TRUMPET/AU	3FFE:8000::/28
TELEBIT/DK	3FFE:0100::/24	ICM-PL/PL	3FFE:8010::/28
SICS/SE	3FFE:0200::/24	IIJ/JP	3FFE:8020::/28
G6/FR	3FFE:0300::/24	QTPVSIX/EU	3FFE:8030::/28
JOIN/DE	3FFE:0400::/24	APAN-KR	3FFE:8040::/28
WIDE/JP	3FFE:0500::/24	MIBH	3FFE:8050::/28
SURFNET/NL	3FFE:0600::/24	ATNET-AT	3FFE:8060::/28

L'expérimentation liée au 6bone s'est terminée récemment; la date d'arrêt a été symboliquement choisie le mardi 6 juin 2006 RFC 3701. En effet, si ce réseau a servi palier à l'absence d'opérateurs officiels au début de l'introduction d'IPv6, il a vite montré ses limites. L'utilisation de tunnels pour créer la connectivité

a conduit à un trop fort maillage, à des routes relativement longues et par conséquence à une faible qualité de service.

C) Adresses gérées par les RIR

La valeur `0x0001` a été attribué par l'IANA pour un plan d'adressage où les autorités régionales attribuent les préfixes. Le préfixe initial est par conséquent `2001::/16`. Ce plan reproduit, en étendant la largeur des champs les principes de délégation et de gestion introduits en IPv4 par CIDR. Un site voulant se connecter reçoit de son fournisseur d'accès un préfixe global de longueur supérieure ou égale à 48 bits.

4) Adresses lien-local

Les adresses de type lien-local (*link local use address*) sont des adresses dont la validité est restreinte à un lien, c'est-à-dire l'ensemble de interfaces directement connectées sans routeur intermédiaire : par exemple machines branchées sur un même Ethernet, machines reliées par une connexion PPP, ou extrémités d'un tunnel. Les adresses lien-local sont configurées automatiquement à l'initialisation de l'interface et permettent la communication entre nœuds voisins. L'adresse est obtenue en concaténant le préfixe `FE80::/64` aux 64 bits de l'identifiant d'interface.

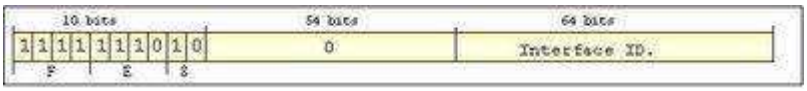


Figure 3-5 Adresse Lien_Local

Ces adresses sont utilisées par les protocoles de configuration d'adresse globale, de découverte de voisins (*neighbor discovery*) et de découverte de routeurs (*router discovery*). Ce sont de nouveaux dispositifs, le premier supplantant en particulier le protocole ARP (*Address Resolution Protocol*), qui permettent pas à un réseau local de se configurer automatiquement (voir Découverte de voisins).

Les adresses lien-local sont uniques à l'intérieur d'un lien. Le protocole de détection de duplication d'adresse (voir Détection d'adresse dupliquée) permet de s'en assurer. Par contre la duplication d'une adresse lien-local entre deux liens différents, ou entre deux interfaces d'un même nœud est autorisée.

Un routeur ne doit en aucun cas retransmettre un paquet ayant pour adresse source ou destination une adresse de type lien-local.

Le fait que ces adresses aient une portée très faible les limite dans la pratique au cas où un démarrage automatique (*bootstrap*) est nécessaire. Leur usage ne doit pas être généralisé dans les applications classiques en régime stabilisé.

A) Portée de l'adresse (*scoped address*)

Pour les adresses de type lien-local ou multicast qui ne permettent de désigner sans ambiguïté l'interface de sortie, il est nécessaire de la spécifier en ajoutant à la fin le nom de l'interface voulue, précédé du caractère "%".

Ainsi dans l'exemple suivant, issue d'une machine BSD :

```
>ping6 fe80::200:c0ff:fee4:caa0
PING6 fe80::1%lo0 --> fe80::200:c0ff:fee4:caa0
ping6: sendmsg: No route to host
ping6: wrote fe80::200:c0ff:fee4:caa0 16 chars, ret=-1
ping6: sendmsg: No route to host
ping6: wrote fe80::200:c0ff:fee4:caa0 16 chars, ret=-1
^C
--- fe80::200:c0ff:fee4:caa0 ping6 statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

La station est incapable de déterminer l'interface de sortie, par contre si l'on utilise la même adresse de destination en précisant l'interface de sortie :

```
>ping6 fe80::200:c0ff:fee4:caa0%xl0
PING6 fe80::2b0:d0ff:fe3b:e565%xl0 --> fe80::200:c0ff:fee4:caa0%xl0
16 bytes from fe80::200:c0ff:fee4:caa0%xl0, icmp_seq=0 hlim=255 time=1 ms
16 bytes from fe80::200:c0ff:fee4:caa0%xl0, icmp_seq=1 hlim=255 time=1.067 ms
^C
--- fe80::200:c0ff:fee4:caa0%xl0 ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1/1.033/1.067 ms
```

On obtient le résultat attendu.

5) Identifiant d'interface

Les types d'adresses global ou lien-local utilisent un identifiant sur 64 bits pour désigner une interface connectée sur un lien. Si cette longueur n'est pas directement imposée par la norme d'adressage d'IPv6 RFC 3513, elle bénéficie d'un fort consensus car elle permet de garantir facilement une unicité sur le lien et par conséquent de faciliter l'auto-configuration des équipements.

Plusieurs techniques ont été élaborées à l'IETF. La plus répandue est basée sur l'utilisation d'une valeur unique par construction comme l'adresse MAC de la machine. Mais l'on peut également choisir une valeur aléatoire pour garantir plus de confidentialité ou au contraire la dériver d'une clé publique pour mieux authentifier l'émetteur du message. La taille de 64 bits permet de réduire à une valeur proche de zéro la probabilité de conflits. Enfin dans certains cas l'affectation manuelle de cette valeur peut se justifier.

Les chapitres suivants décrivent ces différentes méthodes ainsi que leurs intérêts et leurs défauts.

Contents

A) Différents types d'identifiants d'interface

a) EUI-64

L'IEEE a défini un identificateur global à 64 bits (format EUI-64) pour les réseaux IEEE 1394 qui vise une utilisation dans le domaine de la domotique. L'IEEE décrit les règles qui permettent de passer d'un identifiant MAC codé sur 48 bits à un EUI-64.

Il existe plusieurs méthodes pour construire l'identifiant :

Figure 3-8 *identificateur global IEEE EUI-64*

- Si une machine ou une interface possède un identificateur global IEEE EUI-64, celui-ci a la structure décrite figure Identificateur global IEEE EUI-64. Les 24 premiers bits de l'EUI-64, comme pour les adresses MAC IEEE 802, identifient le constructeur et les 40 autres bits identifient le numéro de série (les adresses MAC IEEE 802 n'en utilisaient que 24). Les 2 bits `u` (septième bit du premier octet) et `g` (huitième bit du premier octet) ont une signification spéciale :
  - `u` (Universel) vaut `0` si l'identifiant EUI-64 est universel,
  - `g` (Groupe) indique si l'adresse est individuelle (`g = 0`), c'est-à-dire désigne un seul équipement sur le réseau, ou de groupe (`g = 1`), par exemple une adresse de multicast.

L'ordre des bits ne doit pas porter à confusion. Dans la représentation numérique des valeurs, le premier bit transmis est le bit de poids faible, c'est-à-dire le bit de droite. Ainsi sur le support physique le bit `g`, puis le bit `u` puis les bits suivants sont transmis.

Figure 3-9 Identificateur d'interface dérivé d'une EUI-64

- L'identifiant d'interface à 64 bits est dérivé de l'EUI-64 en inversant le bit `u` (cf. figure Identificateur d'interface dérivé d'une EUI-64). En effet, pour la construction des adresses IPv6, on a préféré utiliser `1` pour marquer l'unicité mondiale. Cette inversion de la sémantique du bit permet de garder la valeur `0` pour une numérotation manuelle, autorisant à numéroter simplement les interfaces locales à partir de `1`.

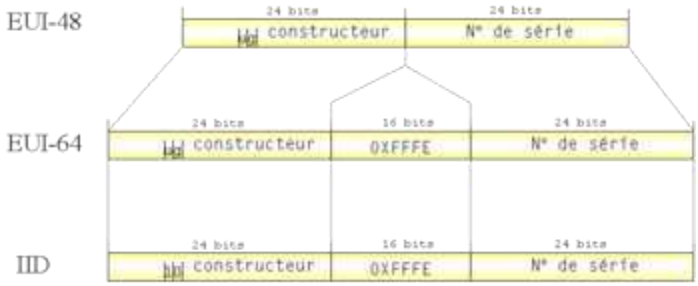


Figure 3-10 Transformation d'une adresse MAC en identifiant d'interface

- Si une interface possède une adresse MAC IEEE 802 à 48 bits universelle (cas des interfaces Ethernet ou FDDI), cette adresse est utilisée pour construire des identifiants d'interface sur 64 bits, comme indiqué sur la figure ci-contre.

A noter que l'IETF s'est trompée quand elle a défini l'algorithme de conversion. En effet, l'ajout de la valeur `0xFFFE` concerne les EUI-48, c'est-à-dire des identifiants, alors qu'Ethernet utilise des MAC-48, c'est-à-dire des adresses (ils servent à transporter des trames vers le bon destinataire). La bonne valeur aurait été `0xFFFF`. Mais cette erreur n'a aucune conséquence pour l'identification des équipements, elle n'a donc pas été corrigée par la suite.

- Si une interface possède une adresse locale unique sur le lien, mais non universelle (par exemple le format d'adresse IEEE 802 sur 2 octets ou une adresse sur un réseau Appletalk), l'identifiant d'interface est construit à partir de cette adresse en rajoutant des 0 en tête pour atteindre 64 bits.
- Si une interface ne possède aucune adresse (par exemple l'interface utilisée pour les liaisons PPP), et si la machine n'a pas d'identifiant EUI-64, il n'y a pas de méthode unique pour créer un identifiant d'interface. La méthode conseillée est d'utiliser l'identifiant d'une autre interface si c'est possible (cas d'une autre interface qui a une adresse MAC), ou une configuration manuelle ou bien une génération aléatoire, avec le bit u positionné à 0.
  S'il y a conflit (les deux extrémités ont choisi la même valeur), il sera détecté lors de l'initialisation de l'adresse lien-local de l'interface, et devra être résolu manuellement.

b) Manuel

L'utilisation de l'adresse MAC pour construire l'adresse IP de la machine peut conduire dans certains cas à des problèmes de configuration, en particulier si la machine est un serveur. En effet, s'il l'on change la carte Ethernet de l'équipement (voire l'équipement) l'adresse IPv6 qui en dépend change également.

Le résolveur DNS est le cas le plus flagrant ; chaque machine sur le réseau doit être configurée avec l'adresse IPv6 du serveur DNS. En cas de changement de carte réseau, l'ensemble des machines du domaine devront être reconfigurées. Si l'on ne souhaite pas utiliser des protocoles de configuration automatique de type DHCPv6, il est préférable d'attribuer au résolveur DNS une adresse manuelle.

c) Valeur aléatoire

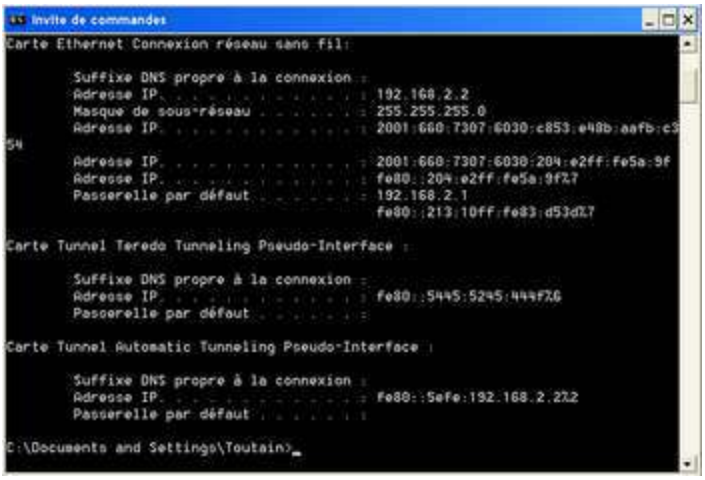
L'identifiant d'interface tel qu'il a été défini précédemment pourrait poser des problèmes pour la vie privée. Il identifie fortement la machine d'un utilisateur, qui même s'il se déplace de réseau en réseau garde ce même identifiant. Il serait alors possible de traquer un individu utilisant un portable, chez lui, au bureau, lors de ses déplacements. Ce problème est similaire à l'identificateur placé dans les processeurs Pentium III.

Pour couper court à toute menace de boycott d'un protocole qui «menacerait la vie privée», il a été proposé d'autres algorithmes de construction d'un identifiant d'interface basé sur des tirages aléatoires (voir RFC 3041). Un utilisateur particulièrement méfiant pourrait valider ces mécanismes. L'identifiant d'interface est soit choisi aléatoirement, soit construit par un algorithme comme MD5 à partir des valeurs précédentes, soit tiré au hasard si l'équipement ne peut pas mémoriser d'information entre deux démarrages. Périodiquement l'adresse est mise dans l'état «déprécié» et un nouvel identifiant d'interface est choisi. Les connexions déjà établies continuent d'utiliser l'ancienne valeur tandis que les nouvelles connexions utilisent la nouvelle adresse.

L'adresse publique globale est conservée, mais ne sera jamais choisie pour initier des communications. Elle permettra par contre d'en recevoir, même si l'anonymat est validé.

Bien entendu pour que ces mécanismes aient un sens, il faut que l'équipement ne s'enregistre pas sous un même nom dans un serveur DNS inverse ou que l'enregistrement de cookies dans un navigateur Web pour identifier l'utilisateur soit impossible.

Ces identifiants privés ne sont pas incompatibles avec les identifiants publics. Une machine peut attendre des ouvertures de connexions sur ses adresses publiques, par contre initier les connexions en utilisant ses identifiants privés. Il suffit de considérer que les adresses publiques sont dépréciées pour une durée indéterminée.



Configuration des interfaces sous Windows

La figure Configuration des interfaces sous Windows illustre cette propriété, en retournant le résultat de la commande `ipconfig`. On peut noter que l'interface du réseau sans-fils possède trois adresses IPv6 (une adresse lien locale et deux adresses globales). Les adresses globales possèdent le même préfixe de 64 octets (`2001:660:7307:6030::/64`). La première adresse globale a le bit `u=0` dans l'identifiant d'interface, il s'agit de celle générée aléatoirement. La deuxième à le bit `u` à `1` et l'on retrouve la séquence `0xFFFE` au milieu de l'identifiant d'interface; cette adresse est dérivée de l'adresse MAC. Sous Windows, par défaut, les adresses aléatoires ont une durée de vie d'une semaine. Par exemple, en utilisant la commande `netsh` :

```
netsh interface ipv6 show address
Recherche du statut actif...
Interface 7 : Connexion réseau sans fil
Type adr.  État DAD  Vie valide Vie préf.  Adresse
---------  --------- ---------- ---------- ----------------------------

Temporaire Préféré     6d21h18m38s    21h15m51s
2001:660:7307:6030:c853:e48b:aafb:c354

Public     Préféré    29d23h58m59s  6d23h58m59s
2001:660:7307:6030:204:e2ff:fe5a:9f

Liaison    Préféré        infinite     infinite fe80::204:e2ff:fe5a:9f
```

d) Cryptographique

Si un identifiant aléatoire permet de rendre beaucoup plus anonyme la source du paquet, des propositions sont faites à l'IETF pour lier l'identifiant d'interface à la clé publique de l'émetteur du paquet. Le RFC 3972 définit le principe de création de l'identifiant d'interface (CGA : *Cryptographic Generated Addresses*) à partir de la clé publique de la machine. Elles pourraient servir pour sécuriser les protocoles de découverte de voisins ou pour la gestion de la multi-domiciliation.

B) Sélection du type d'identifiant d'interface

Si l'on sélectionne correctement le type d'identifiant d'interface, la gestion de l'adressage IPv6 est aussi facile (voire plus simple) qu'en IPv4. Ainsi, il est préférable de donner aux serveurs des identifiants d'interface construit manuellement. Il sera ainsi beaucoup plus facile de se rappeler de leur adresse. De plus si l'équipement est remplacé et par conséquent que la carte réseau est différente, l'adresse IPv6 restera stable. Pour les clients, il est plus simple d'utiliser l'identifiant d'interface construit à partir de l'adresse MAC.

6) Site-local

Les adresses site-local sont des adresses dont la validité était restreinte à un site. Par exemple, un site qui n'est pas encore connecté à l'Internet pouvait utiliser ces adresses, ce qui le dispensait de demander ou d'emprunter un préfixe. Ce système généralisait le concept d'adresse privée d'IPv4 (comme le réseau `10.x.y.z`). Un autre intérêt apparent des adresses site-local est qu'elles ne sont pas modifiées lors d'un changement de fournisseur de connectivité, qui ne perturbe donc pas les communications locales.

Une adresse site-local était construite en concaténant le préfixe `FEC0::/48`, un champ de 16 bits qui permet de définir plusieurs sous-réseaux, et les 64 bits de l'identifiant d'interface.

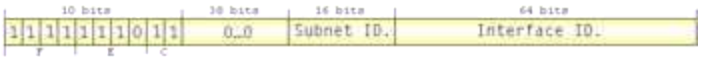


Figure 3-6 *Adresse Site_Local*

Malgré ces propriétés, les adresses site-local n'ont pas réussi à s'imposer durant la phase de standardisation d'IPv6. Suivant les règles de l'IETF, elles doivent donc être retirées des documents pour la version finale du RFC décrivant IPv6. Le RFC 3879 décrit les problèmes liés à l'utilisation des adresses site-local. Contrairement à un lien facilement délimité par le support physique, la frontière du site est beaucoup plus vague. Il s'en suit des ambiguïtés qui rendent difficile l'utilisation de ce concept. En particulier, si un utilisateur est connecté à deux sites de deux compagnies différentes, l'adressage à plat offert par les adresses site-local rend le routage difficile. Si le site dispose aussi d'adresses globales, l'ajout systématique d'adresses site-local rend également plus difficile le choix des adresses source et destination ainsi que la réponse aux requêtes DNS qui dépendent de la position de l'équipement demandeur.

De plus si le réseau de deux compagnies fusionne, comme l'adressage des sous-réseaux ne se fait que dans la partie Subnet ID, il y a de fortes chances de trouver des collisions dans les valeurs choisies.

A) Unique Local Address

Les adresses de type site-local étant supprimées du standard IPv6 [RFC 3879], le RFC 4193 définit un nouveau format d'adresse unicast : les adresses uniques locales (ULA : *Unique Local Address*). Ces adresses sont destinées à une utilisation locale. Elles ne sont pas définies pour être routées dans l'Internet, mais seulement au sein d'une zone limitée telle qu'un site ou entre un nombre limité de sites. Les adresses uniques locales ont les caractéristiques suivantes :

Préfixe *globalement* unique.

Préfixe clairement définit facilitant le filtrage sur les routeurs de bordure.

Permet l'interconnexion de sites sans générer de conflit d'adresse et sans nécessiter de renumérotation.

Indépendantes des fournisseurs d'accès à l'Internet et ne nécessitent donc pas de connectivité.

Pas de conflit en cas de routage par erreur en dehors d'un site.

Aucunes différences pour les applications, qui peuvent les considérer comme des adresses globales unicast standard.

Les adresses uniques locales sont créées en utilisant un identifiant global (*Global ID*) généré pseudo-aléatoirement. Ces adresses suivent le format suivant :

`Prefix` (7 bits) : `FC00::/7` préfixe identifiant les adresses IPv6 locales (*ULA*)
`L` (1 bit) : Positionné à 1, le préfixe est assigné localement. La valeur 0 est réservée pour une utilisation future.
`Global ID` (40 bits) : Identifiant global utilisé pour la création d'un préfixe *unique* (*Globally Unique Prefix*).
`Subnet ID` (16 bits) : Identifiant d'un sous réseau à l'intérieur du site.
`Interface ID` (64 bits) : L'identifiant d'interface tel que définit dans Identifiant d'interface.

7) Autres types d'adresses

Ce paragraphe passe en revue les différents types d'adresses qui n'utilisent pas l'identifiant d'interface.

Contents

A) Adresse indéterminée

L'adresse indéterminée (*unspecified address*) est utilisée comme adresse source par un nœud du réseau pendant son initialisation, avant d'acquérir une adresse. Sa valeur est `0:0:0:0:0:0:0:0` (en abrégé `::`).

Cette adresse est utilisée uniquement par des protocoles d'initialisation, elle ne doit jamais être attribuée à un nœud et ne doit jamais apparaître comme adresse destination d'un paquet IPv6.

B) Adresse de bouclage

L'adresse de bouclage (*loopback address*) vaut `0:0:0:0:0:0:0:1` (en abrégé `::1`). C'est l'équivalent de l'adresse 127.0.0.1 d'IPv4. Elle est utilisée par un nœud pour s'envoyer à lui-même des paquets IPv6.

Un paquet IPv6 transitant sur le réseau ne peut avoir l'adresse de bouclage comme adresse source ni comme adresse destination.

C) Adresses IPv4 mappées

Figure 3-11 *Adresse IPv4 mappée*

Elles sont représentées sous la forme `::FFFF:a.b.c.d` où `a.b.c.d` est une adresse IPv4. On peut bien entendu aussi les écrire sous la forme `::FFFF:XXXX:YYYY` où `XXXXYYYY` est la représentation hexadécimale de l'adresse IPv4 a.b.c.d (cf. Adresse IPv4 mappée).

Ces adresses permettent à une machine de communiquer en IPv4 avec une machine IPv4 tout en restant dans la famille d'adresse `AF_INET6`. Cela permet, entre autres, d'écrire des serveurs (au sens client/serveur) qui peuvent répondre à la fois à des requêtes IPv4 et IPv6 dans le même programme. Cela nécessite bien sûr d'avoir une machine à double pile de communication IPv4 et IPv6. En émission, la machine, voyant une adresse destination IPv4 mappée dans un datagramme IPv6, utilise la pile de communication IPv4 et envoie des paquets IPv4 ; en réception, une requête IPv4 est reçue par la pile IPv4 puis présentée aux applications sous la forme d'une requête IPv6 comportant des adresses de type IPv4 mappée. Une adresse mappée n'est pas censée apparaître dans l'en-tête IPv6. Le principe de fonctionnement est expliqué Programmation d'applications.

D) Les adresses IPv4 compatibles

Figure 3-12 *Adresse IPv4 compatible*

Elles sont représentées sous la forme `::a.b.c.d` où `a.b.c.d` est une adresse IPv4. Comme précédemment, on peut aussi les écrire sous la forme `::XXXX:YYYY` où `XXXXYYYY` est la représentation hexadécimale de l'adresse IPv4 a.b.c.d (cf. Adresse IPv4 compatible).

Ces adresses servent à deux machines IPv6 pour communiquer entre elles en IPv6 à travers un tunnel automatique IPv6 sur IPv4. Un paquet IPv6 transmis vers l'adresse `::a.b.c.d` est encapsulé dans un paquet IPv4 (cf. Tunnels) qui est acheminé à travers le réseau IPv4 vers l'adresse a.b.c.d. Arrivé à destination, le paquet IPv6 est extrait et traité normalement par la pile de communication IPv6.

On conseille d'éviter de généraliser trop l'usage des adresses compatibles. On juge en effet préférable d'utiliser nativement IPv6 à l'intérieur du site sur les liens physiques existants (par exemple Ethernet) et de n'utiliser qu'une machine, en sortie de site, faisant fonction de routeur/tunnelier pour encapsuler les paquets IPv6 dans des paquets IPv4 à destination d'un routeur/de-tunneleur situé en entrée d'un site distant. L'expérience montre que ce type d'adresse ne résout rien. Cela correspond à gérer un réseau avec des adresses IPv4 précédées de 96 bits à 0!

En généralisant ainsi l'usage d'adresses IPv6 globales, on peut espérer s'affranchir plus rapidement des plans d'adressage et de routage IPv4.

E) Les adresses NSAP

Quand les travaux sur IPv6 ont démarré, le protocole de l'ISO avec le format d'adresses NSAP (*Network Service Access Point address*) était encore utilisés, en particulier par DECnet. Il semblait intéressant de

prévoir une place dans l'espace d'adressage pour intégrer ces protocoles. Leurs préfixes était `200::/7` (cf. types des adresses). Depuis, l'intérêt pour ce protocole a beaucoup diminué. Le RFC 1888 décrivait la manière de représenter les adresses NSAP dans des adresses IPv6. Ces adresses sont obsolètes (RFC 4048).

F) Performance des plans d'adressage unicast

L'introduction de ce chapitre a justifié l'emploi d'une longueur fixe de 128 bits en indiquant le nombre d'adresses possibles par mètres carrés terrestre. Bien entendu, cette mesure est quelque peu fantaisiste et ne prend pas en compte le fait qu'un adressage hiérarchique induit un gaspillage d'adresses. Le RFC 3194 définit le ratio Haute Densité HD (du nom de ses auteurs ?):

$$HD = \frac{\log(alloue')}{\log(allouable)}$$

Puisque le numérateur et le dénominateur utilisent des logarithmes, la base de ceux-ci est quelconque. Le HD ratio est souvent exprimé en pourcentage. L'expérience montre, à partir de l'étude d'autres plans d'adressage (téléphonique, constructeurs,...), que lorsque le ratio dépasse 85%, l'exploitation du réseau devient difficile.

Inversement, ce ratio permet de déterminer le nombre d'adresses ou de préfixes qui peuvent être attribués sans que l'exploitation du réseau ne s'en ressente.

Par exemple, pour le plan d'adressage agrégé, les opérateurs allouent un préfixe d'une longueur de 48 bits (en réalité, ils ne disposent que de 45 bits). Si l'on suppose qu'un ratio de 80% est un maximum, si l'on prend les logarithmes en base 2 pour simplifier les calculs, on en déduit que la limite du nombre de sites est de :

$$\text{Nombre de sites} = 2^{0.8 \times 45} = 68719476736$$

Ce ratio est utilisé par les RIR pour juger des politiques d'allocation de préfixes.

8) Exemple d'utilisation des adresses unicast

Le listing suivant donne la configuration des interfaces d'une machine sous Unix.

```
>ifconfig -au
vr0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
   inet 192.108.119.159 netmask 0xffffff80 broadcast 192.108.119.255
   inet6 fe80::250:baff:febe:712%vr0 prefixlen 64 scopeid 0x1
   inet6 2001:660:7301:1:250:baff:febe:712 prefixlen 64 autoconf
   inet6 2001:688:1f99:1:250:baff:febe:712 prefixlen 64 autoconf
   ether 00:50:ba:be:07:12
   media: Ethernet autoselect (10baseT/UTP)
   status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
   inet 127.0.0.1 netmask 0xff000000
   inet6 ::1 prefixlen 128
   inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
```

L'interface Ethernet `vr0` possède une adresse IPv4 et trois adresses IPv6 :

- La première adresse correspond à l'adresse lien-local. On retrouve l'identifiant d'interface qui suit le préfixe `FE80::/64`. A noter que l'on retrouve les octets de l'adresse MAC, sauf pour le premier octet qui est à `02` au lieu de `00` suite à l'inversion du bit «universel/local». A noter que la portée de l'adresse est indiquée par la chaîne de caractère `%vr0`. La valeur scopeid indiquée à la fin de la ligne donne le numéro cette interface.
- Les deux autres adresses correspondent à des adresses globales dont les préfixes ont été attribués par deux opérateurs différents (la machine est sur un réseau multi-domicilié) :
  - `2001` : une adresse unicast globale attribuée par les autorités régionales (cf. Familles d'adressage),
  - `660` : est le préfixe attribué par RIPE-NCC au réseau Renater et `688` à France Télécom
  - `7301` est attribué par Renater à l'ENST-Bretagne et `1f99` par France Télécom,
  - `1` : est le numéro du réseau, pour ces deux préfixes, à l'intérieur de l'ENST Bretagne. Il n'est pas nécessaire d'attribuer le même identifiant de sous-réseaux pour les deux préfixes, mais cela est préférable pour des raisons de commodité d'administration.

L'interface de `lo0` possède les adresses de loopback pour IPv4 et IPv6 (`::1`).

9) Anycast

Le principe sous-jacent est simple : au lieu d'envoyer un paquet à une interface déterminée, on envoie ce paquet à une adresse (anycast) qui représente un ensemble de machines dans un domaine bien défini. Une adresse anycast permet de désigner un service par une adresse bien connue, de cette manière, il n'est pas nécessaire d'interroger un serveur pour connaître la localisation d'un équipement.

Figure 3-13 *Adresse anycast "sous-réseau"*

La figure Adresse anycast «sous-réseau» donne la structure de l'adresse anycast. On y retrouve une partie préfixe et une partie identifiant anycast. La partie préfixe est la même que celle utilisée pour les adresses unicast. Contrairement aux structures d'adresses vue précédemment, la longueur de ce préfixe n'est pas spécifiée, car une adresse anycast doit s'adapter aussi bien aux plans d'adressage actuels (où la longueur est généralement de 64 bits) qu'aux futurs plans qui pourraient avoir des tailles différentes.

Si le concept anycast est simple dans son principe, son implémentation est autrement délicate. En outre, ce concept n'est encore qu'un sujet de recherche. Enfin un autre argument, de taille, explique cette prudence : il n'y a eu aucune expérience grandeur nature analogue au projet Mbone pour le multicast.

Comme les adresses de type sont allouées dans le même espace d'adressage, elles sont créées en attribuant une même adresse unicast à des nœuds distincts, chacun des nœuds devant être configuré pour que l'adresse ainsi attribuée soit de type anycast (sinon on aurait une adresse unicast dupliquée). La seule manière de différencier une adresse anycast d'une adresse multicast est de regarder la partie identifiant anycast qui diffère d'un identifiant d'interface. Ainsi la séquence binaire composée uniquement de 0 a été la première valeur retenue. Elle permet d'identifier un des routeurs du lien. Le RFC 2526 définit des règles de construction d'autres identifiants anycast sur un lien en réservant les 128 identifiants d'interface locaux les plus élevés. Cela permet d'éviter les conflits avec une numérotation manuelle qui partent des identifiants les plus petits vers les plus élevés. Le tableau Allocation des identifiants Anycast donne l'allocation des identifiants utilisés.

Allocation des identifiants Anycast

Description	Valeur(hexadécimal)
Réservé	0x7F
Adresse Anycast de l'agent mère (cf.Mobilité dans IPv6)	0x7E
Réservé	0x00 à 0x7D

Deux modes de fonctionnement non exclusifs sont possibles. Le premier consiste à attribuer sur un préfixe déjà utilisé la même adresse anycast à plusieurs équipements. Le seconde consiste à définir des adresses sur un préfixe "virtuel" et à router au plus près. Les paragraphes suivants expliquent ces modes de fonctionnement.

A) Adresses anycast sur un même lien

Avec les adresses anycast, plusieurs équipements sur un lien physique possèdent une même adresse IPv6. Or il ne s'agit pas d'envoyer la même information à tous ces équipements comme en multicast, mais d'en choisir un seul. Une possibilité consiste à utiliser le mécanisme de découverte de voisins (cf. Découverte de voisins) pour trouver l'association entre l'adresse IPv6 et l'adresse MAC.

La figure Exemple d'utilisation de l'Anycast sur un lien illustre ce fonctionnement. La station A envoie un message de sollicitation de voisin pour déterminer l'adresse MAC de l'équipement. Trois serveurs reçoivent cette requête et répondent. La station A prendra une de ces réponses et dialoguera en point-à-point avec l'équipement choisi.

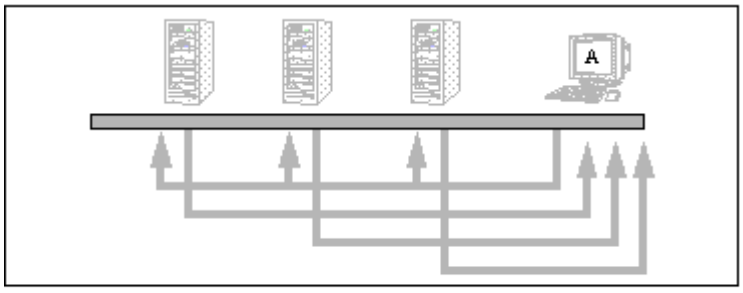


Figure 3-14 . Exemple d'utilisation de l'Anycast sur un lien

B) Préfixe virtuel

Une adresse anycast peut être aussi construite à partir d'un préfixe "virtuel", c'est-à-dire appartenant au site (ou même à un domaine plus grand), mais non alloué à un lien particulier. Le schéma Exemple d'utilisation de l'Anycast dans un site donne un exemple de cette configuration. Les routeurs contiennent des adresses dans les tables de routage (c'est-à-dire des préfixes de longueur 128) pour router vers l'équipement le plus proche. Le sous-réseau 7 a été réservé aux adresses anycast. Les réseaux de 1 à 4 correspondent à des liens. Quand la station C émet un paquet Anycast vers l'adresse `2001:...:7:FFFF:FFFF:FFFF:FF7D`, le routeur R2 route le paquet vers les sous-réseaux 1.

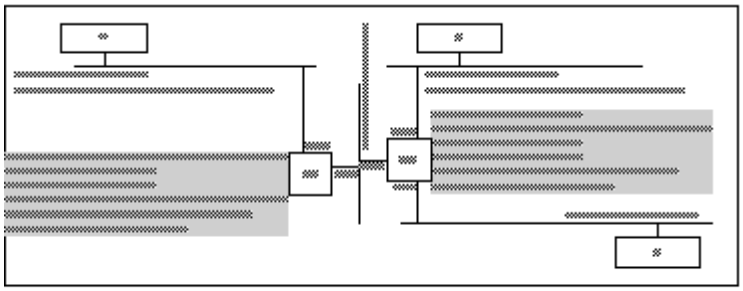


Figure 3-15 . Exemple d'utilisation de l'Anycast dans un site

IV) Protocoles réseau et transport

1) IPv6

Hormis la modification de la taille des adresses, ce qui conduit à une taille d'en-tête de 40 octets (le double de l'en-tête IPv4 sans les options), le protocole IP a subi un toilettage reprenant l'expérience acquise au fil des ans avec IPv4. Le format des en-têtes IPv6 est simplifié et permet aux routeurs de meilleures performances dans leurs traitements :

- L'en-tête ne contient plus le champ checksum, qui devait être ajusté par chaque routeur en raison de la décrémentation du champ durée de vie. Par contre, pour éviter qu'un paquet dont le contenu est erroné -- en particulier sur l'adresse de destination -- ne se glisse dans une autre communication, tous les protocoles de niveau supérieur doivent mettre en œuvre un mécanisme de checksum de bout en bout incluant un pseudo-en-tête qui prend en compte les adresses source et destination. Le checksum d'UDP, facultatif pour IPv4, devient ainsi obligatoire. Pour ICMPv6, le checksum intègre le pseudo-en-tête, alors que pour ICMPv4, il ne portait que sur le message ICMP.
- La taille des en-têtes est fixe. Le routeur peut facilement déterminer où commence la zone de données utiles.
- Les options ont été retirées de l'en-tête et remplacées par de nouveaux en-têtes appelés extensions qui peuvent être facilement ignorées par les routeurs intermédiaires.
- Les champs sont alignés sur des mots de 64 bits, ce qui optimise leur traitement, surtout avec les nouvelles architectures à 64 bits.
- La taille minimale des MTU : Maximum Transmission Unit est de 1 280 octets. Le choix de 1 280 comme MTU minimal en IPv6 permet le tunnelage de paquets IPv6. En effet, la taille de 1 500 octets est généralement admise car elle correspond à la valeur imposée par Ethernet. La majorité des autres réseaux offrent une taille supérieure. La valeur de 576 octets retenue pour IPv4 permettait de prendre en compte des réseaux comme Appletalk. Pour ces réseaux, une couche d'adaptation (comme avec les couches d'adaptation AAL d'ATM) devra être mise en œuvre pour pouvoir transporter les paquets IPv6.
- La fonction de fragmentation a été retirée des routeurs. Les champs qui s'y reportent (identification, drapeau, place du fragment) ont été supprimés. Normalement les algorithmes de

découverte du PMTU(Path MTU) évitent d'avoir recours à la fragmentation. Si celle-ci s'avère nécessaire, une extension est prévue (voir Fragmentation).

Le format d'en-tête d'un paquet IPv6 est donné par le RFC 2474 (cf. Format d'un datagramme IPv6).

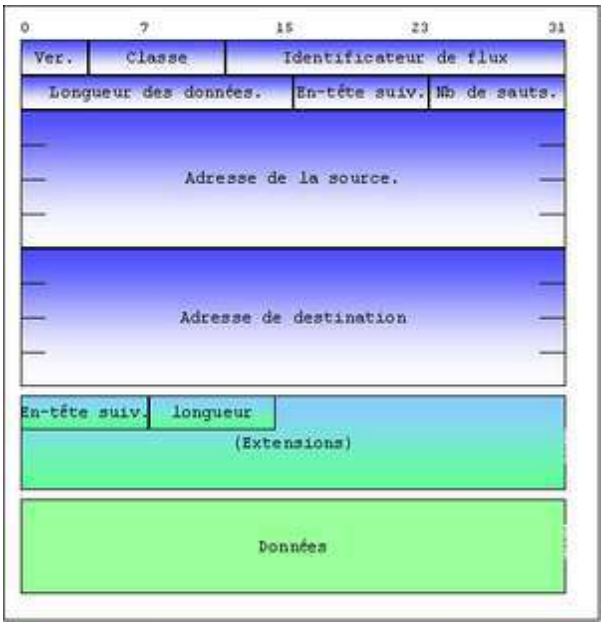


Figure 4-1 *Format d'un datagramme IPv6*

Version

Le champ version est le seul champ qui occupe la même place dans le paquet IPv6 et dans le paquet IPv4. Sa valeur est 6.

Classe de trafic

Identificateur de flux

Longueur des données utiles (payload)

Contrairement à IPv4, ce champ, sur deux octets, ne contient que la taille des données utiles, sans prendre en compte la longueur de l'en-tête. Pour des paquets dont la taille des données serait supérieure à 65 535 ce champ vaut 0 et l'option jumbogramme de l'extension de "proche-en-proche" est utilisée (cf. Jumbogramme) (type 194 ou 0xc2, RFC 2675). Cette option est utilisée quand le champ longueur des données du paquet IPv6 n'est pas suffisant pour coder la taille du paquet. Cette option est essentiellement prévue pour la transmission à grand débit entre deux équipements. Si l'option jumbogramme est utilisée, le champ longueur des données utiles dans l'en-tête IPv6 vaut 0. Noter que le type commence par la séquence binaire 11, ce qui permet au routeur ne traitant pas les jumbogrammes d'en informer la source. Celle-ci pourra réémettre l'information sans utiliser cette option.).

En-tête suivant

Ce champ a une fonction similaire au champ protocole du paquet IPv4. Il identifie le prochain en-tête. Il peut s'agir d'un protocole (de niveau supérieur ICMP, UDP, TCP...) ou de la désignation d'extensions (cf. tableau Valeurs du champ en-tête suivant).

Valeurs du champ en-tête suivant

valeur	Extension	valeur	Protocole
0	Proche-en-proche	4	IPv4
43	Routage	6	TCP
44	Fragmentation	17	UDP
50	Confidentialité	41	IPv6
51	Authentification	58	ICMPv6
59	Fin des en-têtes	132	SCTP
60	Destination	135	Mobilité
		136	UDP-lite

Les extensions contiennent aussi ce champ pour permettre un chaînage.

Nombre de sauts

Il est décrémenté à chaque nœud traversé. Un datagramme retransmis par un routeur est rejeté avec l'émission d'un message d'erreur ICMPv6 vers la source si la valeur après décrémentation atteint 0. Dans IPv4 ce champ est appelé durée de vie (ou TTL Time To Live). Sa vocation initiale est d'indiquer, en secondes, la durée maximale durant laquelle un paquet peut rester dans le réseau. En pratique, les paquets ne restent que quelques millisecondes dans les routeurs, et donc la décrémentation est arrondie à 1. Par contre, pour une liaison plus lente la décrémentation de ce champ peut être supérieure à 1. Dans IPv6, comme il s'agit d'un nombre de sauts, la décrémentation est toujours de 1.
La valeur initiale de ce champ devrait être donnée dans un document annexe de l'IANA (http://www.iana.org/) ce qui permettrait de la modifier en fonction de l'évolution de la topologie du réseau. La valeur n'est pas encore officiellement attribuée, mais certaines implantations prennent actuellement la valeur conseillée pour IPv4 : 64.
La valeur par défaut peut être dynamiquement attribuée aux équipements du réseau par les annonce des routeurs (cf. Configuration automatique), une modification de ce paramètre sera donc relativement simple quand la limite actuelle sera atteinte. On peut noter une limitation, puisque ce champ codé sur 8 bits n'autorise la traversée que de 255 routeurs. En réalité, dans l'Internet actuel, le nombre maximal de routeurs traversés est d'une quarantaine, ce qui laisse une bonne marge pour l'évolution du réseau.

Exemple

Le paquet IPv6 suivant a été capturé au cours d'une connexion FTP :

```
0000:  60  00  00  00  00  28  06  40  3f  fe  03  02  00  12  00  02
0010:  00  00  00  00  00  00  00  13  3f  fe  03  05  10  02  00  01
0020:  02  00  c0  ff  fe  11  cb  a0|ff  b3  00  15  55  4d  fd  d1
0030:  00  00  00  00  a0  02  40  00  7d  76  00  00  02  04  05  a0
0040: 01 03 03 00 01 01 08 0a 00 9a 17 44 00 00 00 00
```

Le paquet commence par 6, qui indique la version du protocole. Le second champ 00 donne la classe de trafic *DiffServ*. L'identificateur de flux n'a pas été défini par la source (0 00 00). La longueur du paquet est de 0x0028 octets. Le paquet ne contient pas d'extension puisque la valeur de l'en-tête suivant, 0x06, correspond au protocole de niveau 4 TCP. Le nombre maximal de routeurs que le paquet pourra traverser est de 64 (0x40). Les adresses source et destination sont des adresses de test appartenant au plan d'adressage agrégé.

A) Champs particuliers

a) Classe de trafic

Le premier mot de 32 bits étant exclu du calcul du checksum avec les pseudo-en-têtes, il est plus facile de le faire évoluer. Dans la version standardisée par le RFC 2460 un champ classe de trafic sur 8 bits permet la différenciation de services conformément aux spécifications du RFC 2474.

Le champ classe de trafic est aussi appelé dans les paquets IPv4 octet DiffServ (DS), il prend la place du champ ToS, initialement défini dans la spécification d'IPv4 (cf. figure Format de l'octet classe de trafic). Le champ DS est découpé en deux parties. Le sous-champ DSCP (*DiffServ Code Point*) contient les valeurs des différents comportements. Les deux derniers bits du champ sont actuellement non utilisés, mais devraient servir aux routeurs pour indiquer un risque de congestion en combinaison avec l'algorithme RED (*Random Early Detection*).

Figure 4-2 *Format de l'octet classe de trafic*

L'Internet différencié permet aux fournisseurs d'accès de gérer différemment les congestions qui surviennent dans le réseau. Sans différenciation, les paquets ont la même probabilité de rejet. Avec la différenciation, plusieurs classes de trafic seront définies. Les paquets appartenant aux classes les plus élevées ont une probabilité de rejet plus faible. Bien entendu pour que l'introduction de telles classes de service soit efficace, il faut offrir des tarifications différentes pour chacune des classes et des mécanismes de contrôle pour vérifier qu'un utilisateur n'utilise pas que les classes les plus élevées ou qu'il dépasse son contrat.

L'intérêt principal de la différenciation de services est qu'elle ne casse pas le modèle initial de l'Internet (version 4 ou version 6). Les flux sont toujours traités en "Best Effort" même si certains sont plus "Best" que d'autres. Il n'y a aucune garantie qu'un trafic d'une classe de service haute arrive à destination, mais la probabilité est plus importante. L'autre intérêt des classes de service vient de la possibilité d'agrégation des flux. La classe d'appartenance est indiquée dans l'en-tête du paquet. Les applications peuvent marquer les paquets en fonction de paramètres locaux (trafic du directeur de la société, flux multimédia interactif,...). Le fournisseur d'accès qui récupère le trafic n'a plus à se préoccuper des applicatifs, il vérifie que le trafic d'une classe ne dépasse pas le contrat préalablement établi.

Dans le cœur de son réseau, les routeurs prennent en compte les différentes classes. Le fournisseur d'accès devra également passer des accords avec les autres opérateurs pour pouvoir faire transiter les flux avec un traitement approprié. Cet aspect de dimensionnement de réseau et de négociation d'accords d'échange est au cœur du métier d'opérateur. Pour l'instant deux types de comportement sont standardisés :

Assured Forwarding : Ce comportement définit quatre classes de services et trois priorités suivant que l'utilisateur respecte son contrat, le dépasse légèrement ou est largement en dehors. Les classes sont donc choisies par l'utilisateur et restent les mêmes tout au long du trajet dans le réseau. La priorité, par contre, peut être modifiée dans le réseau par les opérateurs en fonction du respect ou non des contrats.

Explicit Forwarding : Ce comportement est comparable à un circuit à débit constant établi dans le réseau. Le trafic est mis en forme à l'entrée du réseau, en retardant l'émission des paquets qui sont hors contrat.

En plus de ces comportements, l'octet DS a gardé, pour des raisons de compatibilité avec les équipements existants, les valeurs du bit ToS qui étaient le plus fréquemment utilisées. En particulier, la valeur 0xE0 correspond à la classe de contrôle du réseau (*Network Control*). Elle est utilisée dans des mises en œuvre d'IPv6 pour l'émission de certains paquets ICMPv6.

b) Identificateur de flux

Ce champ contient un numéro unique choisi par la source, qui a pour but de faciliter le travail des routeurs et la mise en œuvre des fonctions de qualité de service comme RSVP. Cet indicateur peut être considéré comme une marque à un contexte dans le routeur. Le routeur peut alors faire un traitement particulier : choix d'une route, traitement en "temps-réel" de l'information.

Avec IPv4, certains routeurs, pour optimiser le traitement, se basent sur les valeurs des champs adresses de la source et de destination, numéros de port de la source et de destination et protocole pour construire un contexte. Ce contexte sert à router plus rapidement les paquets puisqu'il évite de consulter les tables de routage pour chaque paquet. Ce contexte est détruit après une période d'inactivité.

Avec IPv6, cette technique est officialisée. Le champ identificateur de flux peut être rempli avec une valeur aléatoire qui servira à référencer le contexte. La source gardera cette valeur pour tous les paquets qu'elle émettra pour cette application et cette destination. Le traitement est optimisé puisque le routeur n'a plus à consulter cinq champs pour déterminer l'appartenance d'un paquet. De plus si une extension de

confidentialité est utilisée, les informations concernant les numéros de port sont masquées aux routeurs intermédiaires.

L'utilisation de ce champ a été rendue confuse car Cisco dans le cadre du Tag Switching a proposé de l'utiliser pour augmenter la vitesse de commutation des paquets. Cette proposition consiste à ne garantir l'unicité de l'identificateur de flux que sur un lien. Le routeur possède dans sa mémoire une table de correspondance qui permet, en fonction du lien d'arrivée et du numéro d'identificateur de flux, de déterminer le lien de sortie et la nouvelle valeur de l'identificateur. Cette proposition se rapproche énormément des techniques utilisées dans les circuits virtuels (ATM, Frame Relay, X.25,...).

Le groupe de travail MPLS (*Multi Protocol Label Switching*) a intégré les travaux sur le Tag Switching et a précisé la manière dont la commutation des paquets pourra être faite. L'identificateur de flux d'IPv6 n'est plus utilisé, mais un en-tête spécifique est introduit entre l'encapsulation de niveau 2 et celle de niveau 3. L'identificateur de flux n'a plus à être modifié en cours de transmission. Cette évolution clarifie l'utilisation du protocole RSVP (*Reservation Protocol*) qui peut se baser sur cette valeur, identique tout au long du chemin, pour identifier un flux.

En fait, l'utilisation de l'étiquette de flux est très floue, les micro-flux, c'est-à-dire de flux applicatifs, ne sont pas vus dans le cœur du réseau pour des raisons de scalabilité, de plus MPLS a repris la notion de routage spécifique en fonction d'une étiquette. Pour l'instant ce champ peut être vu comme réservé et son utilisation pourra être mieux spécifiée dans le futur.

B) Justification des extensions

L'exemple suivant permet de souligner les problèmes d'utilisation des options dans IPv4, d'illustrer la notion de tunnel et le concept de transmission multicast.

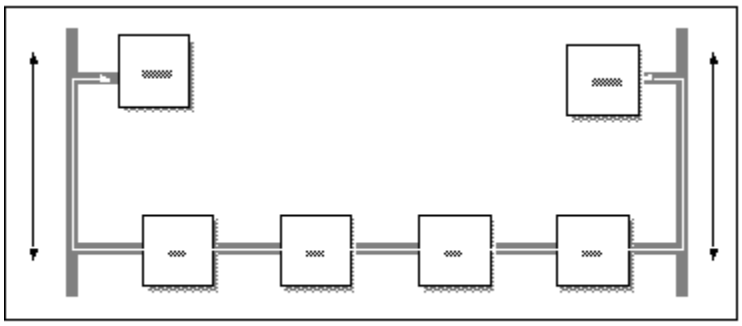


Figure 4-3 . Encapsulation des données multicast pour le Mbone d'IPv4

a) Multicast et options en IPv4

Le trafic multicast du réseau 1 (cf. figure Encapsulation des données multicast pour le Mbone d'IPv4) ne peut pas traverser les routeurs, car le multicast n'existe pas dans les spécifications d'origine d'IPv4 et certains équipements ne savent pas le traiter. Pour pouvoir atteindre le réseau 2, le trafic doit être encapsulé dans un tunnel point-à-point traversant les routeurs intermédiaires. L'équipement qui effectue cette encapsulation et exécute un algorithme de routage multicast s'appelle un mrouteur. Le mrouteur du réseau 1 envoie en point-à-point le trafic multicast vers le mrouteur 2 qui réémet le trafic multicast sur le réseau 2 et inversement.

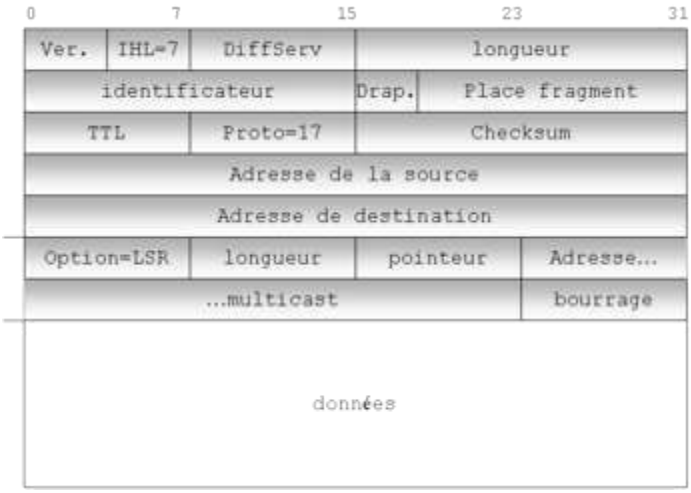


Figure 4-4 *Utilisation du champ option LSR d'IPv4*

La première solution (cf. figure Utilisation du champ option LSR d'IPv4) consiste à émettre le paquet multicast avec l'option de routage libéral par la source (*loose source routing*). Le paquet est destiné au mrouteur 2, qui permute l'adresse de destination avec celle contenue dans le champ option. Le paquet franchissant les routeurs R1 à R4 sera retardé à cause de la présence du champ option. Avec IPv4, les options sont obligatoirement prises en compte par tous les routeurs intermédiaires. Ceux-ci, pour des raisons de performance, privilégient les paquets sans option. De plus, par construction, la longueur du champ option est limitée à 40 octets, ce qui limite l'emploi simultané de plusieurs options.

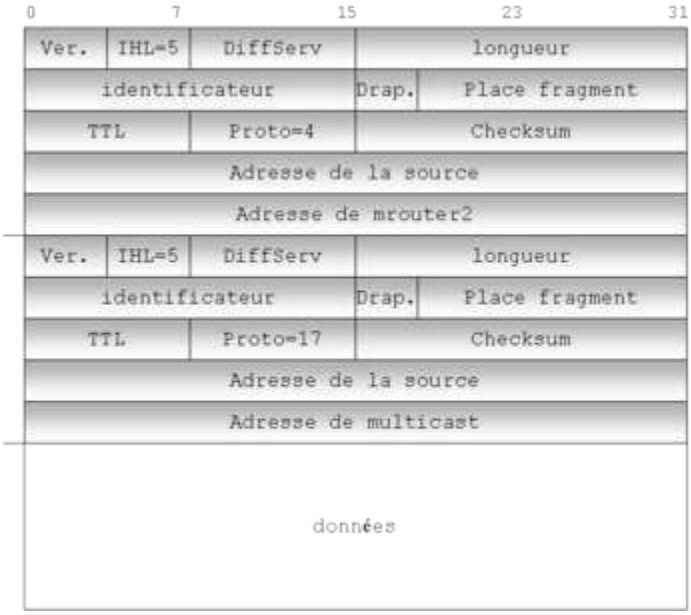


Figure 4-5 *Utilisation d'un tunnel IPv4*

La seconde solution (cf. figure Utilisation d'un tunnel IPv4) consiste à encapsuler le paquet multicast dans un paquet point-à-point destiné au mrouteur 2 ; cette liaison sera appelée un tunnel IPv4. Celui-ci, voyant que la valeur du champ protocole vaut 4 (IP dans IP), retire le premier en-tête et traite le second. Le paquet traversera les routeurs R1 à R4 sans subir de ralentissement puisqu'il ne porte aucun champ option apparent. Par contre, l'en-tête ajouté a une taille très importante ; par conséquent, cette technique ne peut pas être utilisée si plusieurs routeurs servent de relais.

C) Les extensions

Les extensions d'IPv6 peuvent être vues comme un prolongement de l'encapsulation d'IP dans IP. À part l'extension de proche-en-proche traitée par tous les routeurs intermédiaires, les autres extensions ne sont prises en compte que par les équipements destinataires du paquet.

Une extension a une longueur multiple de 8 octets. Elle commence par un champ en-tête suivant d'un octet qui définit le type de données qui suit l'extension : une autre extension ou un protocole de niveau 4 (voir tableau Valeurs du champ en-tête suivant). Pour les extensions à longueur variable, l'octet suivant contient la longueur de l'extension en mots de 8 octets, le premier n'étant pas compté (une extension de 16 octets a un champ longueur de 1).

Valeurs du champ en-tête suivant

valeur	Extension	valeur	Protocole
0	Proche-en-proche	4	IPv4
43	Routage	6	TCP
44	Fragmentation	17	UDP
50	Confidentialité	41	IPv6
51	Authentification	58	ICMPv6
59	Fin des en-têtes	132	SCTP
60	Destination	135	Mobilité
		136	UDP-lite

Le RFC 2460 recommande l'ordre suivant :

- Proche-en-proche (doit toujours être en première position)
- Destination (sera aussi traité par les routeurs listés dans l'extension de routage par la source)
- Routage par la source
- Fragmentation
- Authentification
- Destination (traité uniquement par l'équipement terminal)
-

a) Proche-en-proche

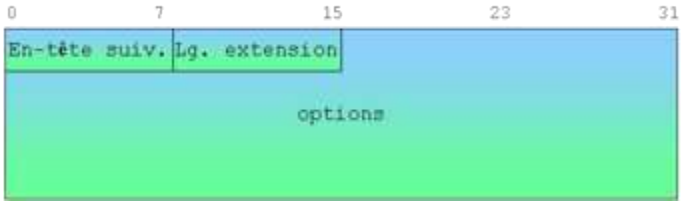


Figure 4-6 *Format générique des options "proche-en-proche" et "destination"*

Cette extension (en anglais : *hop-by-hop*) se situe toujours en première position et est traitée par tous les routeurs que le paquet traverse. Le type associé (contenu dans le champ d'en-tête en-tête suivant de l'en-tête précédent) est 0 et le champ longueur de l'extension contient le nombre de mots de 64 bits moins 1.

L'extension est composée d'options. Pour l'instant, seules quatre options, dont deux de bourrage, sont définies (cf. Format des options IPv6). Chaque option est une suite d'octets. Le premier octet est un type,

le deuxième (sauf pour l'option 0) contient la longueur de l'option moins 2. Les deux premiers bits de poids fort du type définissent le comportement du routeur quand il rencontre une option inconnue :

- 00 : le routeur ignore l'option ;
- 01 : le routeur rejette le paquet ;
- 10 : le routeur rejette le paquet et retourne un message ICMPv6 d'inaccessibilité ;
- 11 : le routeur rejette le paquet et retourne un message ICMPv6 d'inaccessibilité si l'adresse de destination n'est pas multicast.

Le bit suivant du type indique que le routeur peut modifier le contenu du champ option (valeur à 1) ou non (valeur à 0).

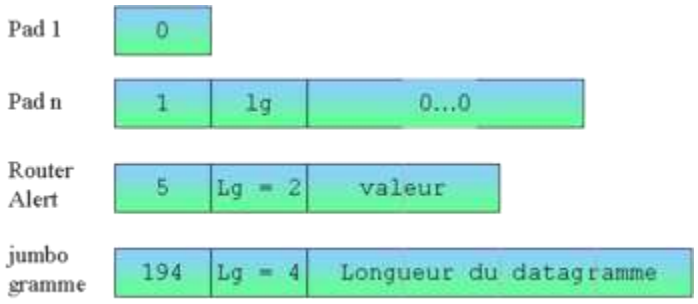


Figure 4-7 *Format des options IPv6*

Les quatre options de proche-en-proche sont :

- Pad1 (type 0). Cette option est utilisée pour introduire un octet de bourrage.
- Padn (type 1). Cette option est utilisée pour introduire plus de 2 octets de bourrage. Le champ longueur indique le nombre d'octets qui suivent.

Les options de bourrage peuvent sembler inutiles avec IPv6 puisqu'un champ longueur pourrait en donner la longueur exacte. En fait les options de bourrage servent à optimiser le traitement des paquets en alignant les champs sur des mots de 32, voire 64 bits ; le RFC 2460 discute en annexe de la manière d'optimiser le traitement tout en minimisant la place prise par les options.

- Jumbogramme (type 194 ou 0xc2, RFC 2675). Cette option est utilisée quand le champ longueur des données du paquet IPv6 n'est pas suffisant pour coder la taille du paquet. Cette option est essentiellement prévue pour la transmission à grand débit entre deux équipements. Si l'option jumbogramme est utilisée, le champ longueur des données utiles dans l'en-tête IPv6 vaut 0. Noter que le type commence par la séquence binaire 11, ce qui permet au routeur ne traitant pas les jumbogrammes d'en informer la source. Celle-ci pourra réémettre l'information sans utiliser cette option.
- L'option Router Alert (RFC 2711) demande à un routeur d'examiner le contenu des données qu'il relaie (Router Alert existe également en IPv4, RFC 2113). En principe, le processus de relayage (recopier le paquet sur une interface de sortie en fonction de l'adresse destination et des tables de routage) doit être le plus rapide possible. Mais pour des protocoles comme la gestion des groupes de multicast avec MLD (*Multicast Listener Discovery*) ou la signalisation des flux avec RSVP, tous les routeurs intermédiaires doivent tenir compte des données. L'émetteur envoie les données à la destination, mais s'il précise l'option Router Alert, les routeurs

intermédiaires vont analyser les données, voire modifier leur contenu avant de relayer le paquet. Ce mécanisme est efficace puisque les routeurs n'ont pas à analyser le contenu de tous les paquets d'un flux.

Le type de l'option vaut 5. Il commence par la séquence binaire 00, puisqu'un routeur qui ne connaît pas cette option doit relayer le paquet sans le modifier. Le champ valeur de l'option contient :

- 0 : pour les messages du protocole MLD de gestion des groupes multicast ;
- 1 : pour les messages RSVP ;
- 2 : pour les réseaux actifs ;

Les autres valeurs sont réservées.

b) Destination

Cette extension, dont le format est identique à l'extension de proche-en-proche (cf. figure Format des extensions "proche-en-proche" et "destination"), contient des options qui sont traitées par l'équipement destinataire. Pour l'instant, à part les options de bourrage (voir Pad1 et Padn) et de mobilité (cf. Mobilité dans IPv6), la seule autre option concerne le tunnelage dans des paquets IPv6 (RFC 2473). Cette option permet de limiter le niveau d'encapsulation dans des tunnels des paquets IPv6.

c) Routage

Cette extension permet d'imposer à un paquet une route différente de celle offerte par les politiques de routage présentes sur le réseau. Pour l'instant seul le routage par la source (type = 0), similaire à l'option *Loose Source Routing* d'IPv4, est défini pour IPv6. La mobilité IPv6 a également introduit une extension de routage (type = 2) (cf. Optimisation dans le cas du mobile dans un réseau étranger).

Dans IPv4, le routage peut être strict (le routeur suivant présent dans la liste doit être un voisin directement accessible) ou libéral (*loose*) (un routeur peut utiliser les tables de routage pour joindre le routeur suivant servant de relais). Dans IPv6, seul le routage libéral est autorisé. En effet, le routage strict était initialement mis en place surtout pour des raisons de sécurité. La source devait être absolument sûre du chemin pris par les paquets. Cette utilisation a maintenant disparu du réseau.

Le principe du routage par la source ou Source Routing dans IPv4 est rappelé en introduction de ce chapitre sur les extensions. Le principe est le même pour IPv6. L'émetteur met dans le champ destination du paquet IPv6, l'adresse du premier routeur servant de relais, l'extension contient la suite de la liste des autres routeurs relais et le destinataire. Quand un routeur reçoit un paquet qui lui est adressé comportant une extension de routage par la source, il permute son adresse avec l'adresse du prochain routeur et réémet le paquet vers cette adresse suivante.

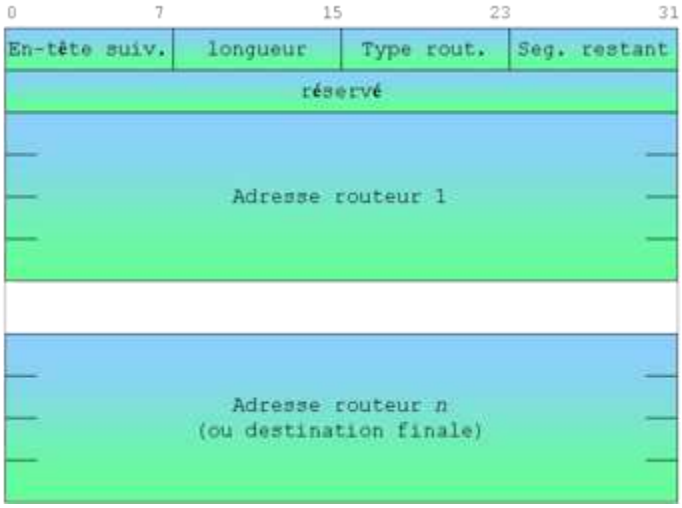


Figure 4-8 *Format de l'extension routage par la source*

La figure Format de l'extension routage par la source donne le format de l'extension de routage par la source :

- Le champ longueur de l'en-tête indique le nombre de mots de 64 bits qui composent l'extension. Pour l'extension de type 0, cela correspond au nombre d'adresses présentes dans la liste, multiplié par 2.
- Le champ type indique la nature du routage. Pour l'instant, seul le routage par la source, de type 0 est spécifié. La suite de l'en-tête correspond à ce type.
- Le nombre de segments restant est décrémenté après la traversée d'un routeur. Il indique le nombre d'équipements qui doivent encore être traversés. Il permet de trouver l'adresse qui devra être substituée.
- Les 32 bits suivants sont inutilisés pour préserver l'alignement.

La liste comprenant les routeurs à traverser et le destinataire est fournie. Ces adresses ne peuvent pas être multicast.

d) Fragmentation

La fragmentation telle qu'elle est pratiquée dans IPv4 n'est pas très performante. Initialement, elle servait à rendre transparente les limitations physiques des supports de transmission. Dans IPv4 quand un routeur ne peut pas transmettre un paquet à cause de sa trop grande taille et si le bit `DF` (*don't fragment*) est à 0, il découpe l'information à transmettre en fragments. Or le réseau IP étant un réseau à datagramme, il n'y a pas de possibilité de contrôler les fragments. Deux fragments successifs peuvent prendre deux chemins différents et par conséquent seul le destinataire peut effectuer le réassemblage. En conséquence, après la traversée d'un lien impliquant une fragmentation, le reste du réseau ne voit passer que des paquets de taille réduite.

Il est plus intéressant d'adapter la taille des paquets à l'émission. Ceci est fait en utilisant les techniques de découverte du MTU (voir Mécanisme de découverte du PMTU (RFC 1981)). En pratique une taille de paquets de 1 500 octets est presque universelle.

Il existe pourtant des cas où la fragmentation est nécessaire. Ainsi une application telle que NFS sur UDP suppose que la fragmentation existe et produit des messages de grande taille. Comme on ne veut pas modifier ces applications, la couche réseau d'IPv6 doit aussi être capable de gérer la fragmentation. Pour réduire le travail des routeurs intermédiaires, la fragmentation se fera chez l'émetteur et le réassemblage chez le récepteur.

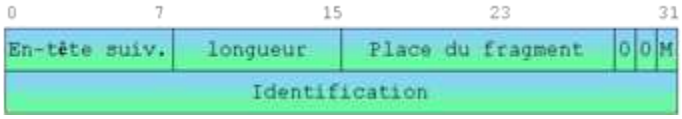


Figure 4-9 *Format de l'extension de fragmentation*

Le format de l'extension de fragmentation est donné figure Format de l'extension de fragmentation. La signification des champs est identique à celle d'IPv4 :

Le champ `place du fragment` indique lors du réassemblage où les données doivent être insérées. Ceci permet de parer les problèmes dus au déséquencement dans les réseaux orientés datagrammes. Comme ce champ est sur 13 bits, la taille de tous les segments, sauf du dernier, doit être multiple de 8 octets.

Le bit `M` s'il vaut 1 indique qu'il y aura d'autres fragments émis.

Le champ `identification` permet de repérer les fragments appartenant à un même paquet initial. Il est différent pour chaque paquet et recopié dans ses fragments.

Le bit `DF` (*don't fragment*) n'est plus nécessaire puisque, si un paquet est trop grand, il y aura rejet du paquet par le routeur.

Dans IPv4, la valeur d'une option était codée de manière à indiquer au routeur effectuant la fragmentation si elle devait être copiée dans les fragments. Dans IPv6, l'en-tête et les extensions qui concernent les routeurs intermédiaires (pour l'instant proche-en-proche, routage par la source) sont recopiées dans chaque fragment.

Sécurité

Deux extensions de sécurité -- les extensions d'authentification AH (*Authentication Header*) et de confidentialité ESP (*Encapsulating Security Payload*) -- sont définies par l'IETF. Elles permettent de protéger les communications passées sur les réseaux IPv6 mais aussi IPv4 en assurant les services de confidentialité, authentification, intégrité et détection de rejeux. Le chapitre Sécurité, RFC 2402 donne une description détaillée de ces extensions, et présente les modes de protection existants.

D) Extensions

a) Exemple de routage par la source

Le paquet suivant a été capturé lors de l'ouverture d'une connexion telnet. La commande telnet permet de spécifier des paramètres de routage par la source. Ainsi `telnet @routeur1@destination` permet un routage libéral vers la destination en passant par le routeur intermédiaire routeur1.

```
IPv6
Version : 6 Classe : 00 Label : 00000
Longueur : 64 octets (0x0040) Protocole : 43 (0x2b) En-tête de routage
Nombre de sauts : 64 (0x40)
Source : 3ffe:302:12:2::13
Desti. : 3ffe:302:12:5:2a0:c9ff:feaa:2201 (routeur1)
Routage
En-tête Suivant : 06 (0x06) TCP
Longueur Extension : 0x02 => 128 bits
Type de routage = 0x00 (Routage par la source)
Segments restant : 0x01. Réservé 0x00
Réservé : 0x00000000
Adresse suivante : 3ffe:305:1002:1:200:c0ff:fe11:cba0 (destination)
TCP
Port Source, 0xffb1 Port Destination :0x0017 (Telnet)
Sequence : 0x17107e57 Acquittement : 0x00000000
Offset : 0xa Drapeau : 0x02 (SYN) Fenêtre : 0x4000
Checksum : 0x356e Ptr Msg Urgent : 0x0000
Options TCP
```

```
0000: 60 00 00 00 00 40 2b 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 02 00 12 00 05
0020: 02 a0 c9 ff fe aa 22 01|06 02 00 01 00 00 00 00
0030: 3f fe 03 05 10 02 00 01 02 00 c0 ff fe 11 cb a0|
0040: ff b1 00 17 17 10 7e 57 00 00 00 00 a0 02 40 00
0050: 35 6e 00 00 02 04 05 a0 01 03 03 00 01 01 08 0a
0060: 00 9a 1d 04 00 00 00 0b
```

Dans l'en-tête IPv6, le numéro de protocole `0x2b` indique qu'une extension de routage est insérée. Noter que le champ longueur des données utiles prend en compte la longueur de l'extension. Le champ adresse de destination de l'en-tête IPv6 contient l'adresse du routeur intermédiaire.

La partie extension commence par l'encapsulation suivante, ici `0x06` pour TCP. Le champ suivant (`0x02`) donne la longueur de l'extension en mots de 64 bits. La partie donnée contient donc une seule adresse IPv6. Il s'agit de la destination. Le type de routage vaut `0` et le champ segment restant vaut `1` et pointe vers l'adresse de destination.

b) Exemple de fragmentation

Les paquets suivants correspondent à l'envoi d'un datagramme de longueur 3 500 octets en UDP alors que le MTU de l'interface est 1 500.

```
IPv6
```

```
Version : 6 Classe : 00 Label : 00000
Longueur : 1456 octets (0x05b0) Proto. : 44 (0x2c) En-tête de frag.
Nombre de sauts : 64 (0x40)
Source : 3f fe 03 02 00 12 00 02 00 00 00 00 00 00 00 13
Desti. : 3f fe 03 02 00 12 00 05 02 a0 c9 ff fe aa 22 01
Fragmentation
En-tête Suivant : 17 (0x11) UDP
Longueur Extension : 0x02 => 128 bits
Place du Fragment : 0x0000 bit M =1
Identificateur : 0x0000008e
UDP
Port Source : 0xf38e Port Destination : 0x000d
Longueur : 3508 (0x0db4) Checksum : 0xc227

0000: 60 00 00 00 05 b0 2c 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 02 00 12 00 05
0020: 02 a0 c9 ff fe aa 22 01|11 02 00 01 00 00 00 8e|
0030: f3 8e 00 0d 0d b4 c2 27 30 31 32 33 34 35 36 37
0040: 38 39 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e
0050: 4f 50 51 52 53 54 55 56 57 58 59 5a 61 62 63 64
0060: 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74
0070: ...
```

Le bit `M` de l'option de fragmentation est à `1`, un autre fragment va suivre.

```
IPv6
Version : 6 Classe : 00 Label : 00000
Longueur : 1456 octets (0x05b0) Proto. : 44 (0x2c) En-tête de frag.
Nombre de sauts : 64 (0x40)
Source : 3f fe 03 02 00 12 00 02 00 00 00 00 00 00 00 13
Desti. : 3f fe 03 02 00 12 00 05 02 a0 c9 ff fe aa 22 01
Fragmentation
En-tête Suivant : 17 (0x11) UDP
Longueur Extension : 0x02 => 128 bits
Place du Fragment : 0x05a8 bit M =1
Identificateur : 0x0000008e

0000: 60 00 00 00 05 b0 2c 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 02 00 12 00 05
0020: 02 a0 c9 ff fe aa 22 01|11 02 05 a9 00 00 00 8e|
0030: 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54
0040: 55 56 57 58 59 5a 61 62 63 64 65 66 67 68 69 6a
0050: 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a
0060: 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46
0070: 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56
0080: ...
```

Ce fragment est la suite du précédent puisque la valeur de l'identificateur est la même (`0x0000008e`), le bit `M` étant à 1, d'autres fragments vont suivre. Le champ Place du fragment contient 1 448. Si l'on prend en compte la taille des extensions (8 octets), on retrouve bien la taille des informations utiles (1 456) transportées dans le paquet précédent.

E) Checksum au niveau transport

Parmi les différences existant entre les datagrammes IPv4 et IPv6, il y a la disparition du checksum dans les en-têtes IP. Cette somme de contrôle était utilisée pour vérifier la validité de l'en-tête du paquet traité. En IPv4, il est nécessaire de la vérifier et de l'ajuster lors de chaque retransmission par un routeur, ce qui entraîne une augmentation du temps de traitement du paquet.

Cette somme ne vérifie que l'en-tête IPv4, pas le reste du paquet. Aujourd'hui les supports physiques sont de meilleure qualité et savent détecter les erreurs (par exemple, Ethernet a toujours calculé sa propre somme de contrôle ; PPP, qui a presque partout remplacé SLIP, possède un CRC). L'intérêt de la somme de contrôle a diminué et ce champ a été supprimé de l'en-tête IPv6.

Figure 4-10 *Champ du pseudo-en-tête*

Le checksum sur l'en-tête IPv6 n'existant plus, il faut quand même se prémunir des erreurs de transmission. En particulier, une erreur sur l'adresse de destination va faire router un paquet dans une mauvaise direction. Le destinataire doit donc vérifier que les informations d'en-tête IP sont incorrectes pour éliminer ces paquets. Dans les mises en œuvre des piles de protocoles Internet, les entités de niveau transport remplissent certains champs du niveau réseau. Il a donc été décidé que tous les protocoles au-dessus d'IPv6 devaient utiliser une somme de contrôle intégrant à la fois les données et les informations de l'en-tête IPv6. La notion de pseudo-en-tête dérive de cette conception. Pour un protocole comme TCP qui possède une somme de contrôle, cela signifie modifier le calcul de cette somme. Pour un protocole comme UDP qui possède une somme de contrôle facultative, cela signifie modifier le calcul de cette somme et le rendre obligatoire.

IPv6 a unifié la méthode de calcul des différentes sommes de contrôle. Celle-ci est calculée sur l'ensemble formé de la concaténation d'un pseudo-en-tête (cf. Champ du pseudo-en-tête) et du paquet du protocole concerné. L'algorithme de calcul du checksum est celui utilisé en IPv4. Il est très simple à mettre en œuvre et ne demande pas d'opérations compliquées. Il s'agit de faire la somme en complément à 1 des mots de 16 bits du pseudo-en-tête, de l'en-tête du protocole de transport, et des données, puis de prendre le complément à 1 du résultat.

Il faut noter que les informations contenues dans le pseudo-en-tête ne seront pas émises telles quelles sur le réseau. Le champ "en-tête suivant" du pseudo-en-tête ne reflète pas celui qui sera émis dans les paquets puisque les extensions ne sont pas prises en compte dans le calcul du checksum. Ainsi, si l'extension de routage est mise en œuvre, l'adresse de la destination est celle du dernier équipement. De même le champ longueur est sur 32 bits pour contenir la valeur de l'option jumbogramme, si celle-ci est présente.

2) ICMPv6

Le protocole de contrôle d'IP a été revu. Dans IPv4, ICMP (Internet Message Control Protocol) sert à la détection d'erreurs (par exemple : équipement inaccessible, durée de vie expirée,...), au test (par exemple ping), à la configuration automatique des équipements (redirection ICMP, découverte des routeurs). Ces trois fonctions ont été mieux définies dans IPv6. De plus ICMPv6 (RFC 2463) intègre les fonctions de gestion des groupes de multicast (MLD : Multicast Listener Discovery) qui sont effectuées par le protocole IGMP (Internet Group Message Protocol) dans IPv4. ICMPv6 reprend aussi les fonctions du protocole ARP utilisé par IPv4.

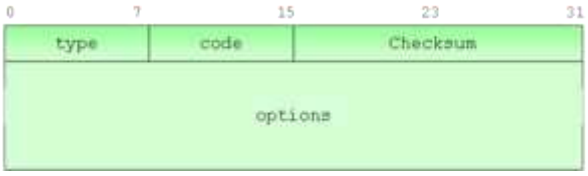


Figure 4-11 *Format générique d'un message ICMP*

Le protocole se voit attribuer le numéro 58. Le format générique des paquets ICMPv6 est donné figure Format générique d'un message ICMP :

- Le champ `type` (voir tableau Valeurs des champs type et code d'ICMPv6) code la nature du message ICMPv6. Contrairement à IPv4 où la numérotation ne suivait aucune logique, les valeurs inférieures à 127 sont réservées aux messages d'erreur. Les autres valeurs réservées aux messages d'information, parmi lesquels se trouvent ceux utilisés par le protocole découverte des voisins (*neighbor discovery*) pour la configuration automatique des équipements.
- Le champ `code` précise la cause du message ICMPv6.
- Le champ `checksum` permet de vérifier l'intégrité du paquet ICMP. Ce champ est calculé avec le pseudo-en-tête décrit au chapitre Checksum au niveau transport.

Les messages ICMPv6 de compte rendu d'erreur contiennent dans la partie ‘’données’’ le paquet IPv6 ayant provoqué l'erreur. Pour éviter des problèmes de fragmentation puisqu'il est difficilement envisageable de mettre en œuvre la découverte du MTU, la longueur du message ICMPv6 est limitée à 1 280 octets et par conséquent le contenu du paquet IPv6 peut être tronqué.

Valeurs des champs type et code d'ICMPv6

type code nature

Gestion des erreurs		
1		Destination inaccessible :
	0	* aucune route vers la destination
	1	* la communication avec la destination est administrativement interdite
	2	* hors portée de l'adresse source
	3	* l'adresse est inaccessible
	4	* le numéro de port est inaccessible
2		Paquet trop grand
3		Temps dépassé :
	0	* limite du nombre de sauts atteinte
	1	* temps de réassemblage dépassé
4		Erreur de paramètre :
	0	* champ d'en-tête erroné
	1	* champ d'en-tête suivant non reconnu
	2	* option non reconnue
Information		
128		Demande d'écho
129		Réponse d'écho
Gestion des groupes multicast (MLD, RFC 2710)		
130		Requête d'abonnement
131		Rapport d'abonnement
132		Fin d'abonnement

Découverte de voisins (RFC 2461)		
133		Sollicitation du routeur
134		Annonce du routeur
135		Sollicitation d'un voisin
136		Annonce d'un voisin
137		Redirection
Renumérotation des routeurs (expérimental, RFC 2894)		
138		Renumérotation des routeurs :
	0	* Commande
	1	* Résultat
	255	* Remise à zéro du numéro de séquence
Recherche d'information sur un nœud (expérimental)		
139		Demande d'information
140		Réponse
Découverte de voisins inverse (RFC 3122)		
141		Sollicitation
142		Annonce
Gestion des groupes multicast (MLDv2, RFC 3810)		
143		Rapport d'abonnement MLDv2
Mobilité (RFC 3775)		
144		Découverte d'agent mère (requête)
145		Découverte d'agent mère (réponse)
146		Sollicitation de préfixe mobile
147		Annonce de préfixe mobile

Découverte de voisins sécurisée (SEND, RFC 3971)		
148		Sollicitation de chemin de certification
149		Annonce de chemin de certification
Mobilité (expérimental)		
150		Protocoles de mobilité expérimentaux, tels que Seamoby

A) Destination inaccessible

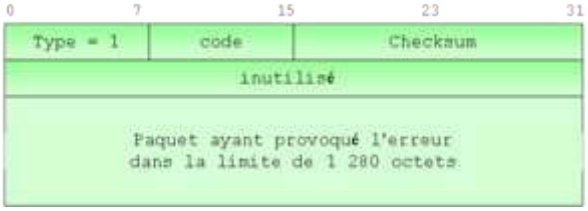


Figure 4-12 *Format d'un message ICMP Destination inaccessible*

Ce message est émis par un routeur intermédiaire quand le paquet ne peut pas être transmis parce que soit :

- le routeur ne trouve pas dans ses tables la route vers la destination (code = 0) ;
- le franchissement d'un équipement de type firewall est interdit ("raison administrative", code = 1) ;
- l'adresse destination ne peut être atteinte avec l'adresse source fournie, par exemple si le message est adressé à un destinataire hors du lien, l'adresse source ne doit pas être une adresse lien-local (code = 2) ;
- toute autre raison comme par exemple la tentative de routage d'une adresse locale au lien (code = 3) ;
- le destinataire peut aussi émettre un message ICMPv6 de ce type quand le port destination contenu dans le paquet n'est pas affecté à une application (code = 4).

B) Paquet trop grand

Figure 4-13 *Format d'un message ICMP Paquet trop grand*

Ce message ICMPv6 est utilisé par le protocole de découverte du MTU pour trouver la taille optimale des paquets IPv6 afin qu'ils puissent traverser les routeurs. Ce message contient la taille du MTU acceptée par le routeur pour que la source puisse efficacement adapter la taille des données. Ce champ manquait cruellement dans les spécifications initiales de IPv4, ce qui compliquait la découverte de la taille maximale des paquets utilisables sur l'ensemble du chemin (cf. découverte du PMTU (RFC 1981)). Pour IPv4, le RFC 1191 proposait déjà une modification du comportement des routeurs pour y inclure cette information.

C) Temps dépassé

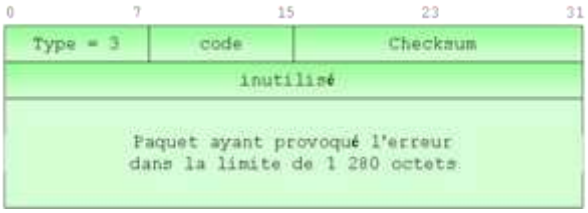


Figure 4-14 *Format d'un message ICMP Temps dépassé*

Ce message indique que le paquet a été rejeté par le routeur :

- soit parce que le champ nombre de sauts a atteint 0 (code = 0) ;
- soit qu'un fragment s'est perdu et le temps alloué au réassemblage a été dépassé (code = 1).

Ce message sert aussi à la commande traceroute pour déterminer le chemin pris par les paquets.

D) Erreur de paramètre

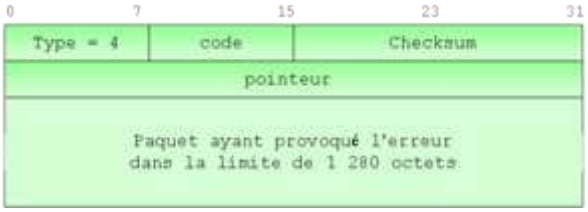


Figure 4-15 *Format d'un message ICMP Erreur de paramètre*

Ce message est émis par un nœud ayant détecté une erreur de syntaxe dans l'en-tête du paquet IP ou des extensions. Le champ code révèle la cause de l'erreur :

- la syntaxe de l'en-tête n'est pas correcte (code = 0) ;
- le numéro en-tête suivant n'est pas reconnu (code = 1) ;
- une option de l'extension (par exemple proche-en-proche ou destination) n'est pas reconnue et le codage des deux bits de poids fort oblige à rejeter le paquet (code = 2).

Le champ pointeur indique l'octet où l'erreur est survenue dans le paquet retourné.

E) Demande et réponse d'écho

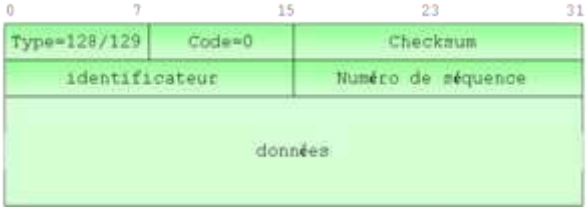


Figure 4-16 *Format d'un message ICMP Demande et réponse d'écho*

Ces deux messages servent en particulier à la commande ping permettant de tester l'accessibilité d'une machine. Le principe de fonctionnement est le même que pour IPv4, une requête (type 128) est envoyée vers l'équipement dont on veut tester le fonctionnement, celui-ci répond par le message réponse d'écho (type 129). Le champ identificateur permet de distinguer les réponses dans le cas où plusieurs commandes ping seraient lancées simultanément sur la machine. Le champ numéro de séquence permet d'associer la réponse à une requête pour mesurer le temps d'aller et retour dans le cas où les demandes sont émises en continu et que le délai de propagation est élevé. Le champ ''données'' permet d'augmenter la taille du message pour les mesures.

Le chapitre sur l'API, mini-ping, donne un exemple de programmation utilisant ces messages ICMPv6.

3) Protocoles de Niveau 4

A) UDP et TCP

Les modifications apportées aux protocoles de niveau 4 UDP et TCP sont minimes. L'un des pré-requis à la mise en œuvre d'IPv6 était de laisser en l'état aussi bien TCP (*Transmission Control Protocol*) qu'UDP (*User Datagram Protocol*). Ces protocoles de transport sont utilisés par la très grande majorité des applications réseau et l'absence de modification facilitera grandement le passage de IPv4 à IPv6.

La principale modification à ces protocoles concerne le checksum. Comme il a été précisé Checksum au niveau transport, il a été adapté au format de paquet IPv6 et englobe le pseudo-en-tête. De plus, pour UDP, le checksum qui était facultatif en IPv4, devient obligatoire.

Un autre changement au niveau des protocoles de niveau 4 concerne la prise en compte de l'option jumbogramme de l'extension proche-en-proche. Le RFC 2675 définit le comportement de UDP et de TCP quand les jumbogrammes sont utilisés. En effet, les en-têtes de ces messages contiennent eux aussi un champ longueur codé sur 16 bits et par conséquent insuffisant pour coder la longueur du jumbogramme :

Pour le protocole UDP, si la longueur des données excède 65 535 octets, le champ longueur est mis à 0. Le récepteur détermine la longueur des données par la connaissance de la taille dans l'option jumbogramme.

Le protocole TCP pose plus de problèmes. En effet, bien que les messages TCP ne contiennent pas de champ longueur, plusieurs compteurs sont codés sur 16 bits.

Le champ longueur de la fenêtre de réception ne pose pas de problème depuis que le RFC 1323 a défini l'option TCP window scale qui donne le facteur multiplicatif qui doit être appliqué à ce champ.

À l'ouverture de connexion, la taille maximale des segments (MSS) est négociée. Le RFC 2675 précise que si cette taille doit être supérieure à 65 535, la valeur 65 535 est envoyée et le récepteur prend en compte la longueur déterminée par l'algorithme de découverte du MTU.

Pour l'envoi de données urgentes avec TCP, on utilise un bit spécifique de l'en-tête (bit URG) ainsi que le champ "pointeur urgent". Ce dernier sert à référencer la fin des données à traiter de manière particulière. Trois cas peuvent se présenter :

Le premier, qui est identique à IPv4, est celui où le pointeur indique une position de moins de 65 535.

Le second se produit lorsque le déplacement est supérieur à 65 535 et supérieur ou égal à la taille des données TCP envoyées. Cette fois-ci, on place la valeur 65 535 dans le champ "pointeur urgent" et on continue le traitement normal des paquets TCP.

Le dernier cas intervient quand le pointeur indique un déplacement de plus de 65 535 qui est inférieur à la taille des données TCP. Un premier paquet est alors envoyé, dans lequel on met la valeur 65 535 dans le champ "pointeur urgent". L'important est de choisir une taille de paquet de manière à ce que le déplacement dans le second paquet, pour indiquer la fin des données urgentes, soit inférieur à 65 535.

Il existe d'autres propositions pour faire évoluer TCP. Il faut remarquer que le travail n'est pas de même ampleur que pour IP. En effet, TCP est un protocole de bout-en-bout, la transition vers une nouvelle génération du protocole peut se faire par négociation entre les deux extrémités. Pour IP, tous les routeurs intermédiaires doivent prendre en compte les modifications.

B) UDP-lite

UDP-lite permet de remonter aux couches supérieures des données erronées pendant leur transport. Si dans un environnement informatique, une erreur peut avoir des conséquences relativement grave quant à l'intégrité des données et il est normal de rejeter ces paquets, or, la plupart des décodeurs de flux multimédias sont capables de supporter un certains nombre d'erreurs binaires dans un flux de données. Pour améliorer la qualité perçue par l'utilisateur, il est donc préférable d'accepter des paquets erronés plutôt que de rejeter un bloc complet d'information.

En IPv4, l'utilisation du checksum UDP étant optionnelle (la valeur 0 indique que le checksum n'est pas calculé), UDP peut être utilisé pour transporter des flux multimédia. Avec IPv6, l'utilisation du checksum a été rendue obligatoire puisque le niveau 3 n'en possède pas. Pour éviter qu'un paquet comportant des erreurs ne puisse pas être remonté aux couches supérieures, le protocole UDP-lite a été défini RFC 3828. Les modifications sont minimes par rapport à UDP. Le format de la trame reste le même, seule la sémantique du champ longueur est changée. Avec UDP, ce champ est inutile puisqu'il est facilement déduit du champ longueur de l'en-tête IP. UDP-lite le transforme en champ couverture du checksum. Si la longueur est 0, UDP-lite considère que tout le checksum couvre tout le paquet. La valeur 8 indique que seul l'en-tête UDP est protégé par le checksum (ainsi qu'une partie de l'en-tête IP grâce au pseudo-header). Les valeurs comprises entre 1 et 7 sont interdites car le checksum UDP-lite doit toujours couvrir l'en-tête. Une valeur supérieure à 8 indique qu'une partie des données sont protégées. Si la couverture est égale à la longueur du message on se retrouve dans un cas compatible avec UDP.

C) SCTP

Le protocole SCTP (*Stream Control Transmission Protocol*) RFC 2960 est fortement lié au protocole IPv6. SCTP est un protocole de niveau 4 initialement conçu pour transporter des informations de signalisation. La fiabilité est donc un pré-requis important et la gestion de la multi-domiciliation est prise en compte. L'idée est de permettre aux deux équipements terminaux d'échanger à l'initialisation de la connexion (appelée dans le standard association), l'ensemble de leurs adresses IPv4 et IPv6. Chaque équipement choisi une adresse privilégiée pour émettre les données vers l'autre extrémité et surveille périodiquement l'accessibilité des autres adresses. Si l'équipement n'est plus accessible par l'adresse principale, une adresse secondaire sera choisie.

SCTP permet une transition douce d'IPv4 vers IPv6 puisque l'application n'a plus à se préoccuper de la gestion des adresses. Si les deux entités possèdent une adresse IPv6, celle-ci sera privilégiée. De plus, SCTP

peut servir de brique de base à la gestion de la multi-domiciliation IPv6. En effet, avec TCP une connexion est identifiée par ses adresses. Si une adresse n'est plus accessible, le fait d'en changer peut conduire à la coupure de la connexion. Il faut avoir recours à des super-fuges, comme la mobilité IP pour maintenir la connexion. SCTP brise ce lien entre la localisation de l'équipement et l'identification des associations.

V) Configuration automatique et contrôle

1) *Découverte de voisins*

Le protocole de découverte des voisins (*neighbor discovery*) permet à un équipement de s'intégrer dans l'environnement local, c'est-à-dire le lien sur lequel sont physiquement transmis les paquets IPv6. Il permet de dialoguer avec les équipements connectés au même support (stations et routeurs). Il ne s'agit pas pour un équipement de connaître exactement la liste de tous les autres équipements connectés sur le lien, mais uniquement de gérer ceux avec qui il dialogue.

Le protocole utilise cinq types de messages ICMPv6 (voir Valeurs des champs type et code d'ICMPv6). Le champ nombre de sauts de l'en-tête IPv6 contient la valeur 255 -- il peut sembler paradoxal d'utiliser une valeur aussi grande pour des datagrammes qui ne doivent pas être routés hors du lien physique ; en fait si un équipement reçoit un datagramme avec une valeur plus petite, cela signifie que l'information provient d'un autre réseau et que le datagramme doit être rejeté.

Le protocole réalise les différentes fonctions :

- Résolution d'adresses. Le principe est très proche du protocole ARP que l'on trouve avec IPv4. La principale différence vient de l'emploi de messages standards ICMPv6 à la place de la définition d'un autre protocole de niveau 3. Cela confère une plus grande souplesse d'utilisation en particulier sur les réseaux qui ne supportent pas la diffusion. Comme pour IPv4, ce protocole construit des tables de mise en correspondance entre les adresses IPv6 et physiques.
- Détection d'inaccessibilité des voisins ou NUD (*Neighbor Unreachability Detection*). Cette fonction n'existe pas en IPv4. Elle permet d'effacer des tables de configuration d'un équipement les voisins qui sont devenus inaccessibles (panne, changement d'adresse,...). Si un routeur devient inaccessible, la table de routage peut être modifiée pour prendre en compte une autre route.
- Configuration. La configuration automatique des équipements est l'un des attraits principaux d'IPv6. Plusieurs fonctionnalités du protocole de découverte des voisins sont mises en œuvre :
    - Découverte des routeurs. Ce protocole permet aux équipements de déterminer les routeurs qui sont sur leur lien physique. Dans IPv4, ces fonctionnalités sont assurées par le protocole ICMP Router Discovery.
    - Découverte des préfixes. L'équipement apprend le ou les préfixes du réseau en fonction des annonces faites par les routeurs. En y ajoutant l'identifiant d'interface de l'équipement, celui-ci construit son ou ses adresses IPv6. Il n'existe pas d'équivalent pour le protocole IPv4 puisque les adresses sont trop courtes pour faire de l'auto-configuration.

- Détection des adresses dupliquées. Comme les adresses sont construites automatiquement, il existe des risques d'erreurs en cas d'identité de deux identifiants. Ce protocole teste qu'aucun autre équipement sur le lien ne possède la même adresse IPv6. Cette fonctionnalité est une évolution de l'ARP gratuit d'IPv4 émis à l'initialisation de l'interface.
- Découverte des paramètres. Ce protocole permet aux équipements d'apprendre les différents paramètres du lien physique, par exemple, la taille du MTU, le nombre de sauts maximal autorisé (valeur initiale du champ nombre de sauts), si la configuration automatique avec état (comme DHCPv6) est active... Il n'existe pas d'équivalent en IPv4.

- Indication de redirection. Ce message est utilisé quand un routeur connaît une route meilleure (en nombre de sauts) pour aller à une destination.
  En IPv4 une indication de redirection ne peut servir qu'à corriger l'adresse du routeur utilisé pour accéder à une machine hors du réseau local. Les machines doivent connaître toutes les adresses correspondant aux réseaux locaux.

  Avec IPv6, la correspondance entre préfixe et réseau local est moins stricte. Il est prévu qu'un matériel ne connaisse pas tous les préfixes de son réseau local (si celui-ci est partagé par plusieurs préfixes), ou qu'un préfixe soit partagé entre plusieurs liens (une généralisation du modèle des réseaux logiques d'IP sur ATM). Dans certaines configurations, la machine émettra ses paquets au routeur alors que le destinataire se trouve sur le même segment que l'émetteur. Si c'est le cas, le routeur émettra un message de redirection pour que la suite du dialogue se fasse directement (cf. exemples Indication de redirection).

  Dans le cas le plus extrême, on peut imaginer en IPv6 qu'un équipement peut être configuré pour dialoguer uniquement avec son routeur par défaut. ICMPv6 «redirect» est alors utilisé pour informer l'équipement des destinataires sur le même lien.

A) Données véhiculées par les messages

L'intérêt du protocole de découverte des voisins est d'unifier différents protocoles qui existent dans IPv4. En particulier la plupart des données utilise un format d'options commun, ce qui simplifie la mise en œuvre du protocole. Le format contient les champs type, longueur en mots de 64 bits, données. La faible précision du champ longueur va introduire une perte de place. En contrepartie, elle va permettre aussi un alignement des options sur des mots de 64 bits, ce qui optimise leur traitement.

En plus des cinq options générales décrites dans le tableau Utilisation des options dans les messages de découverte des voisins, il existe d'autres options spécifiques pour la mobilité et les réseaux NBMA (*Non Broadcast Multiple Access*) comme ATM ou Frame Relay.

	sollicitation du routeur	annonce du routeur	sollicitation d'un voisin	annonce d'un voisin	indication de redirection
adresse physique de la source	présent	présent	présent		
adresse physique de la cible				présent	présent
information sur le préfixe		≥ 1			
en-tête redirigée					présent
MTU		possible			

a) Adresse physique de la source/cible

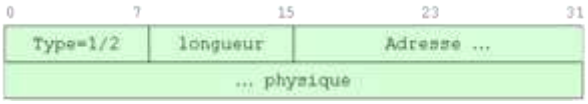


Figure 5-1 Format de l'option adresse physique source/cible

La figure Format de l'option adresse physique source/cible donne le format de ces options. Le type 1 est réservé à l'adresse physique de la source et le type 2 à l'adresse de la cible.

Le champ «longueur» est la taille en mots de 64 bits de l'option. Dans le cas d'une adresse MAC, d'une longueur de 6 octets, il contient donc la valeur 1.

b) Information sur le préfixe

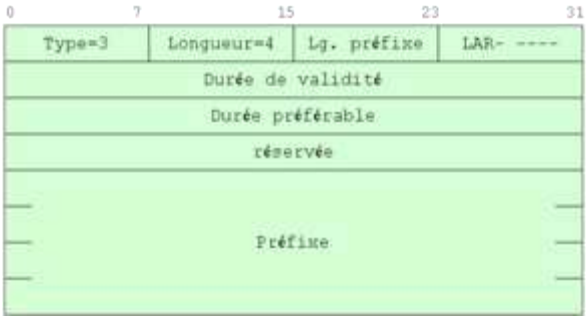


Figure 5-2 *Format de l'option information sur le préfixe*

Cette option contient les informations sur le préfixe pour permettre une configuration automatique des équipements. Le champ type vaut 3 et le champ longueur vaut 4. La figure Format de l'option information sur le préfixe donne le format de l'option :

- Le champ lg.préfixe indique combien de bits sont significatifs pour le préfixe annoncé dans un champ suivant.
- Le bit `L` indique, quand il est à 1, que le préfixe permet d'indiquer que tous les autres équipements partageant le même préfixe sont sur le même lien. L'émetteur peut donc directement les joindre. Dans le cas contraire, l'équipement émet le paquet vers le routeur. Si ce dernier sait que l'équipement émetteur peut joindre directement le destinataire, il émettra un message ICMPv6 d'indication de redirection.
- Le bit `A` indique, quand il est à 1, que le préfixe annoncé peut être utilisé pour construire l'adresse de l'équipement.
- Le bit `R`, indique, quand il est à 1, que le champ préfixe contient l'adresse globale d'un routeur «agent mère». Les bits de poids fort peuvent toujours être utilisés pour construire un préfixe.
- Le champ durée de validité indique en secondes la durée pendant laquelle le préfixe est valide.
- Le champ durée préférable indique la durée en secondes pendant laquelle une adresse construite avec le protocole de configuration sans état demeure «préférable» (cf. Durée de vie des adresses).

Pour ces deux champs, une valeur de `0xffffffff` représente une durée infinie. Ces champs peuvent servir dans la phase de passage d'un fournisseur d'accès à un autre ; c'est-à-dire d'un préfixe à un autre.

- Le champ réservé permet d'aligner le préfixe sur une frontière de mot de 64 bits.
- Le champ préfixe contient la valeur de préfixe annoncé sur le lien. Pour maintenir un alignement sur 64 bits pour le reste des données du paquet, ce champ a une longueur fixe de 128 bits.

c) *En-tête redirigée*

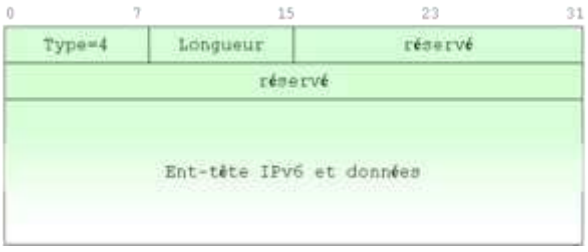


Figure 5-3 *Format de l'option en-tête redirigée*

Cette option est utilisée par le message d'indication de redirection. Elle permet d'encapsuler les premiers octets du paquet IPv6 qui a provoqué l'émission de ce message comme dans le cas des messages ICMPv6 d'erreur.

Le type vaut 4 et la taille de cette option ne doit pas conduire à un paquet IPv6 dépassant 1280 octets (cf. figure Format de l'option en-tête redirigée). Par contre le paquet doit contenir le maximum d'information possible.

d) MTU

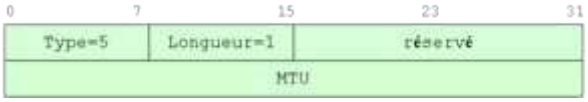


Figure 5-4 *Format de l'option MTU*

Cette option permet d'informer les équipements sur la taille maximale des données pouvant être émises sur le lien. La figure Format de l'option MTU donne le format de cette option. Il n'est pas nécessaire de diffuser cette information si l'équipement utilise toujours la taille maximale permise. Par exemple, sur les réseaux Ethernet, les équipements utiliseront la valeur 1 500. Par contre pour les réseaux anneau à jeton ou FDDI, il est souvent nécessaire de préciser si les équipements doivent utiliser la valeur maximale permise ou une valeur inférieure pour autoriser l'utilisation de ponts.

Le champ type vaut 5 et le champ longueur 1.

B) Messages de découverte de voisins

Les différentes fonctionnalités de découverte des voisins utilisent 5 messages : 2 pour le dialogue entre un équipement et un routeur, 2 pour le dialogue entre voisins et un dernier pour la redirection. Chacun de ces messages peut contenir des options.

a) Sollicitation du routeur

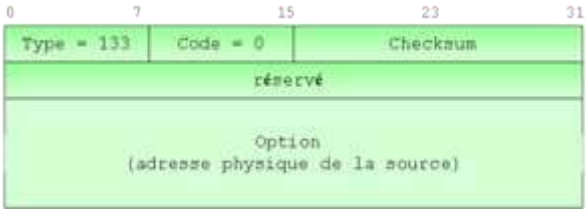


Figure 5-5 *Format des paquets de sollicitation du routeur*

Le message de sollicitation d'un routeur (cf. figure Format des paquets de sollicitation du routeur) est émis par un équipement au démarrage pour recevoir plus rapidement des informations du routeur. Ce message est émis à l'adresse IPv6 de multicast réservée aux routeurs sur le même lien `ff02::2`. Si l'équipement ne connaît pas encore son adresse source, l'adresse non spécifiée est utilisée.

Le champ option contient normalement l'adresse physique de l'équipement.

b) Annonce du routeur

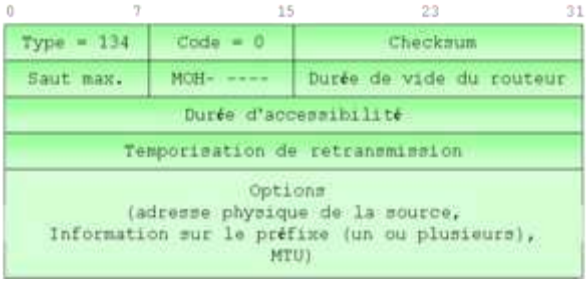


Figure 5-6 *Format des paquets d'annonce du routeur*

Ce message (cf. figure Format des paquets d'annonce du routeur) est émis périodiquement par les routeurs ou en réponse à un message de sollicitation d'un routeur par un équipement. Le champ adresse source contient l'adresse locale au lien du routeur, le champ destination contient soit l'adresse de l'équipement qui a émis la sollicitation, soit l'adresse de toutes les stations (`ff02::01`).

Un champ saut max. non nul donne la valeur qui pourrait être placée dans le champ nombre de sauts des paquets émis. Le bit `M` indique qu'une adresse de l'équipement doit être obtenue avec un protocole de configuration (cf. Configuration avec état :DHCPv6). Le bit `O` indique aussi la présence d'un service de configuration mais pour la récupération d'informations autres que l'adresse. Si l'adresse ne peut être obtenue d'un serveur, l'équipement procède à une configuration sans état en concaténant aux préfixes qu'il connaît son identifiant d'interface. Le bit `H` indique que le routeur peut être utilisé comme «agent mère» pour un nœud mobile (cf. Avertissement de l'agent mère).

Le champ ‘’durée’’ de vie du routeur donne, en secondes, la période pendant laquelle l'équipement annonçant effectuera les fonctions de routeur par défaut. La valeur maximale correspond à 18 heures 12 minutes, mais comme ce message est émis périodiquement il n'y a pas de limite théorique à la durée de vie d'un routeur. Une valeur de `0` indique que l'équipement ne remplit pas les fonctions de routeur par défaut. Cette durée de vie ne s'applique pas aux options que ce message véhicule.

Le champ durée d'accessibilité indique la durée en millisecondes pendant laquelle une information contenue dans le cache de la machine peut être considérée comme valide (par exemple, la table de correspondance entre adresse IPv6 et adresse physique). Au bout de cette période, un message de détection d'inaccessibilité est émis pour vérifier la pertinence de l'information.

Le champ temporisation de retransmission donne en millisecondes la période entre deux émissions non sollicitées de ce message. Il sert aux autres équipements pour détecter une inaccessibilité du routeur.

Ce message peut véhiculer les options :

Adresse physique de la source,

MTU,

Information sur le préfixe (une ou plus).

c) Sollicitation d'un voisin

Figure 5-7 *Format des paquets de sollicitation d'un voisin*

Ce message (cf. figure Format des paquets de sollicitation d'un voisin) permet d'obtenir des informations d'un équipement voisin, c'est-à-dire situé sur le même lien physique (ou connecté via des ponts). Le message peut lui être explicitement envoyé ou émis sur une adresse de diffusion. Dans le cas de la détermination de l'adresse physique, il correspond à la requête ARP du protocole IPv4.

Le champ adresse source du paquet IPv6 contient soit l'adresse locale au lien adresse lien-local, soit une adresse globale, soit l'adresse non spécifiée. Le champ destination contient soit l'adresse de multicast sollicité correspondant à l'adresse recherchée (cf. Identifiant de groupe), soit l'adresse de l'équipement (dans le cas d'une détection d'inaccessibilité des voisins, NUD)

Le champ adresse de la cible contient l'adresse IPv6 de l'équipement cherché. Le champ option contient en général l'adresse physique de la source.

d) Annonce d'un voisin

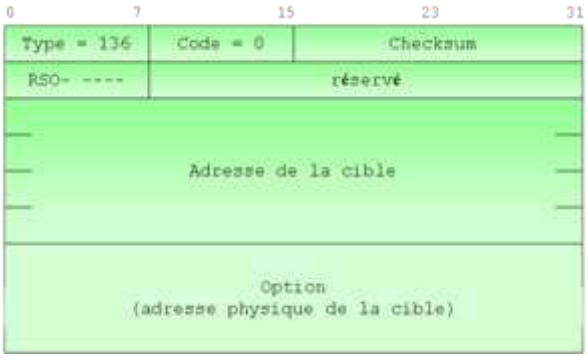


Figure 5-8 *Format des paquets d'annonce d'un voisin*

Ce message (cf. figure Format des paquets d'annonce d'un voisin) est émis en réponse à une sollicitation, mais il peut aussi être émis spontanément pour propager une information de changement d'adresse physique, ou de statut «routeur». Dans le cas de la détermination d'adresse physique, il correspond à la réponse ARP pour le protocole IPv4.

- Le bit R est mis à 1 si l'émetteur est un routeur. Ce bit est utilisé pour permettre la détection d'un routeur qui redevient un équipement ordinaire.
- Le bit S mis à 1 indique que cette annonce est émise en réponse à une sollicitation.
- Le bit O mis à 1 indique que cette annonce doit effacer les informations précédentes qui se trouvent dans les caches des autres équipements, en particulier la table contenant les adresses physiques.
- Le champ adresse de la cible contient, si le bit S est à 1, la valeur du champ adresse de la cible de la sollicitation auquel ce message répond. Si le bit S est à 0, ce champ contient l'adresse IPv6 lien-local de l'équipement émetteur.
- L'option adresse physique de la cible contient l'adresse physique de l'émetteur.

e) Indication de redirection

La technique de redirection est la même que dans IPv4. Un équipement ne connaît que les préfixes des réseaux auxquels il est directement attaché et l'adresse d'un routeur par défaut. Si la route peut être optimisée, le routeur par défaut envoie ce message pour indiquer qu'une route plus courte existe. En effet, avec IPv6, comme le routeur par défaut est appris automatiquement, la route n'est pas forcément la meilleure (cf. figure Routage par défaut non optimal).

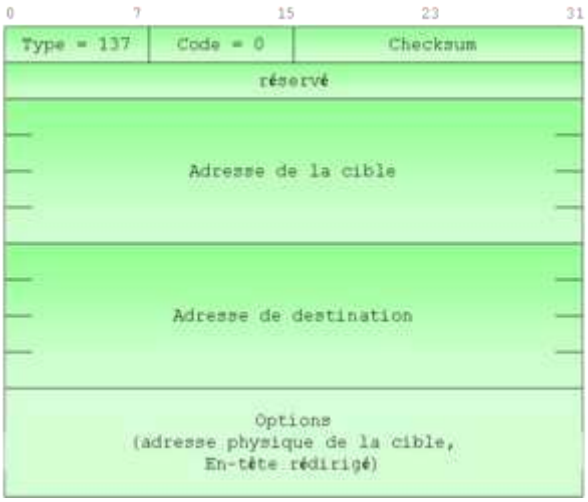


Figure 5-9 *Format des paquets d'indication de redirection*

Un autre cas d'utilisation particulier à IPv6 concerne des stations situées sur un même lien physique mais ayant des préfixes différents. Ces machines passent dans un premier temps par le routeur par défaut. Ce dernier les avertit qu'une route directe existe.

La figure Format des paquets d'indication de redirection donne le format du message :

- Le champ adresse cible contient l'adresse IPv6 de l'équipement vers lequel les paquets doivent être émis.
- Le champ adresse destination contient l'adresse IPv6 de l'équipement pour lequel la redirection s'applique.

Dans le cas de la redirection vers un équipement se situant sur le même lien, l'adresse cible et la destination sont identiques.

Les options contiennent l'adresse physique du nouveau routeur et l'en-tête du paquet redirigé.

C) Exemples de découverte de voisins

a) Ping et résolution d'adresses

Les paquets suivants ont été obtenus lors d'un ping entre deux stations IPv6 situées sur le même réseau physique de type Ethernet.

```
uma# ping6 ganesha
trying to get source for ganesha
source should be 3ffe:302:12:3:a00:20ff:fe0a:aa6d
PING ganesha (3ffe:302:12:3::3): 56 data bytes
64 bytes from 3ffe:302:12:3::3: icmp6_seq=0 ttl=255 time=5.121 ms
```

Avant de pouvoir émettre un paquet ICMPv6 de demande d'écho, l'émetteur a besoin de connaître l'adresse physique de l'équipement destinataire. Il utilise le protocole de découverte des voisins et émet une trame de sollicitation d'un voisin.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 33:33:ff:0:0:3 Type : 0x86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 32 octets (0x0020) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0xff)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : ff02::1:ff00:3 (multicast sollicité associé à 3ffe:302:12:3::3)
ICMPv6
 Type : 135 (0x87, Sollicitation de voisin) Code : 0 Checksum : 0x4d7f
 Cible : 3ffe:302:12:3::3 (ganesha)
 Option :
 Type : 1 (Adresse physique source) Lg : 8 octets (0x01) : 08-00-20-0a-aa-6d

0000: 6f 00 00 00 00 20 3a ff 3f fe 03 02 00 12 00 03
0010: 0a 00 20 ff fe 0a aa 6d ff 02 00 00 00 00 00 00
0020: 00 00 00 01 ff 00 00 03|87 00 4d 7f 00 00 00 00
0030: 3f fe 03 02 00 12 00 03 00 00 00 00 00 00 00 03|
0040: 01 01 08 00 20 0a aa 6d
```

Dans l'en-tête IPv6, l'adresse de la source est l'adresse globale de l'interface d'émission. On aurait pu penser que l'émetteur utilisait l'adresse locale au lien comme adresse de la source. L'utilisation de l'adresse source globale, comme on le verra par la suite, permet au destinataire de remplir directement sa table de correspondance entre adresse IPv6 et adresse physique, puisque ce dernier trouvera dans la suite du datagramme l'adresse physique de l'émetteur.

L'adresse de destination est l'adresse de multicast sollicité associée à l'adresse recherchée et l'adresse Ethernet de destination est l'adresse associée (cf. Supports de transmission RFC 2464).

L'en-tête ICMPv6 contient dans le champ cible l'adresse IPv6 de la machine dont l'adresse physique est recherchée. On peut remarquer que les trois derniers octets correspondent au groupe de multicast de l'en-tête IPv6.

Le champ option contient l'adresse physique de l'émetteur de la requête.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 8:0:20:a:aa:6d Type : 0x86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 32 octets (0x20) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0xff)
 Source : fe80::1800:20ff:fe0c:7a34 (ganesha, lien-local)
 Desti. : 3ffe:302:12:3:0a00:20ff:fe0a:aa6d (uma)
ICMPv6
 Type : 136 (0x88, Annonce de voisin) Code : 0 Checksum : 0xd7fb
 Bits (0x7) R = 1, S = 1, O = 1
 Cible : 3ffe:302:12:3::3 (ganesha)
 Option :
 Type : 2 (Adresse physique cible) Lg : 8 octets (0x01) : 1a-00-20-0c-7a-34
```

La machine `ganesha`, qui écoute sur tous les groupes multicast sollicité associés à ses adresses, reçoit le message de sollicitation de voisin, reconnaît dans la cible une de ses adresses IPv6, et répond. L'adresse source utilisée est locale au lien. Le bit `R` indique que l'équipement qui répond a une fonction de routeur. Le bit `S` indique que ce message est une réponse à une demande explicite (le message précédent). Le bit `O` indique que cette réponse doit remplacer toute valeur connue précédemment. Le champ cible rappelle l'adresse IPv6. Le champ option donne l'adresse physique recherchée.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 1a:0:20:c:7a:34 Type : 0x86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0xff)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:302:12:3::3 (ganesha)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0x0f20
 Identificateur : 0x00c0 Numéro de séquence : 0x0000
 Données : Date : 0x3468c4c7.000631c7 Remplissage ...
```

L'émetteur envoie un message ICMPv6 Demande d'écho.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 8:0:20:a:aa:6d Type : 0x86dd
IPv6
Version : 6 Classe : 0x00 Label : 000000
Longueur : 64 octets (0x0040) Protocole : 58 (0x3a, ICMPv6)
Nombre de sauts : 255 (0xff)
Source : 3ffe:302:12:3::3 (ganesha)
Desti. : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
ICMPv6
Type : 129 (0x81, Réponse d'écho) Code : 0 Checksum : 0x0e20
Identificateur : 0x00c0 Numéro de séquence : 0x0000
Données : Celles de la demande
```

Le destinataire acquitte en retournant un message ICMPv6 Réponse d'écho. Il n'est pas nécessaire de relancer une phase de résolution d'adresse puisque la précédente a permis de remplir le cache.

Les échanges ICMP Demande d'écho et Réponse d'écho continuent ensuite toutes les secondes. Si les échanges continuent assez longtemps, les deux machines vérifieront périodiquement que le correspondant

est toujours correct (il a pu tomber en panne ou être remplacé avec changement d'adresse Ethernet) en utilisant le protocole NUD. Aussi de temps en temps, chaque machine va émettre une trame sollicitation d'un voisin. Une réponse (annonce de voisin avec le bit `S`) montre que le correspondant est toujours valide.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 8:0:20:a:aa:6d Type : 0x86dd
IPv6
 Source : fe80::1800:20ff:fe0c:7a34 (ganesha, lien-local)
 Desti. : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
ICMPv6
 Type : 135 (0x87, Sollicitation de voisin) Code : 0
 Cible : 3f fe 03 02 00 12 00 03 0a 00 20 ff fe 0a aa 6d
Option : aucune
```

On remarque que le message de sollicitation est directement adressé au destinataire, avec l'adresse qui est enregistrée dans les tables de correspondance. Si une réponse n'arrive pas, la machine émettrice effacera l'entrée de son cache «Résolution de voisin». Tout trafic ultérieur reprendra l'enquête de résolution au début, avec diffusion vers l'adresse multicast sollicité-- au cas où l'adresse Ethernet aurait changée.

b) Configuration de la route par défaut

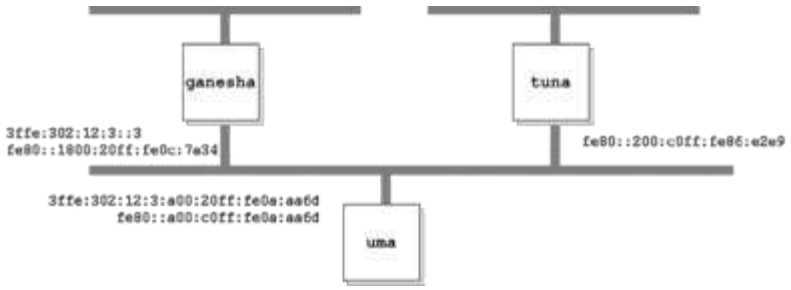


Figure 5-10 *Annonces du routeur*

En IPv6 seuls les routeurs utilisent des protocoles de routage pour définir leurs tables de routage. Le routage des autres machines repose sur la notion de route par défaut. Comme avec IPv4, l'envoi de messages de redirection est utilisé pour installer de meilleures routes. Périodiquement les routeurs envoient des Annonces du routeur qui permettent aux machines sur le câble de choisir un routeur par défaut, et aussi de calculer leur adresse (dans le mode d'auto-configuration sans état stateless).

Un même câble Ethernet relie 3 machines (cf. figure Annonces de routeurs) :

- deux routeurs `ganesha` et `tuna`,
- et une autre machine `uma`.

Les routeurs émettent périodiquement sur le réseau des messages d'annonce.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 33:33:0:0:0:1 Type : 0x86dd
IPv6
 Version : 6 Classe : 0xf Label : 000000
 Longueur : 56 octets (0x0038) Protocole : 58 (0x3a, ICMPv6)
```

```
 Nombre de sauts : 255 (0xff)
 Source : fe80::1800:200c:7a34 (ganesha, lien-local)
 Desti. : ff02::1 (multicast, tous les noeuds du lien)
ICMPv6
 Type : 134 (0x86, Annonce de routeur) Code : 0 Checksum : 0x773c
 Nombre de sauts : 0 (non précisé) Gestion d'adresse : 0 (Pas de DHCP)
 Validité : 6000 secondes (0x1770) Timers : 0, 0 (non précisés)
 Options :
  Type : 1 (Adresse physique source) Lg : 8 octets (0x01) : 1a-00-20-0c-7a-34
  Type : 3 (Information sur le préfixe) Lg : 32 octets (0x04)
   Drapeaux : L=1, A=1 Durée de validité : -1, -1 (infinie)
  Préfixe : 3ffe:302:12:3::/64
0000: 6f 00 00 00 00 38 3a ff fe 80 00 00 00 00 00 00
0010: 18 00 20 ff fe 0c 7a 34 ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 01|86 00 77 3c 00 00 17 70
0030: 00 00 00 00 00 00 00 00|01 01 1a 00 20 0c 7a 34|
0040: 03 04 40 c0 ff ff ff ff ff ff ff ff 00 00 00 00
0050: 3f fe 03 02 00 12 00 03 00 00 00 00 00 00 00 00
```

Ce message est envoyé par le routeur `ganesha` (l'adresse source est l'adresse locale, commençant par `fe80`), à destination de tous les nœuds sur le câble Ethernet (adresse de destination IPv6 «Tous les nœuds sur le lien» `ff02::1` et l'adresse de destination physique est l'adresse MAC de multicast associée). Les informations donnent la durée de vie de cette annonce, des paramètres de configuration pour les nœuds, dont le type de construction d'adresse : mode d'auto-configuration *stateless* en créant une adresse permanente à partir du préfixe `3ffe:302:12:3::/64`.

La table de routage d'`uma` après réception de ce message contient :

```
uma# netstat -nrf inet6
Routing tables IPv6:
Destination                       Gateway                     Flags Refs Use Mtu  Interf
default                           fe80::1800:20ff:fe0c:7a34 UGS    0   17  1500 le0
fe80::1800:20ff:fe0c:7a34         1a:0:20:c:7a:34 U         HDL    1   2   1500 le0
.....
```

La ligne avec le drapeau `L` correspond à une machine directement accessible, le champ *Gateway* contient l'adresse IEEE 802. Il s'agit des informations contenues dans la table de correspondance construite par le protocole de découverte des voisins.

Le routeur `tuna` avait lui aussi émis des annonces de routeur. On pourrait donc aussi enregistrer une route par défaut via `tuna` ; mais le système ne conserve qu'une route par défaut, et la route possible par `tuna` est ignorée. De même, il n'y a pas de ligne de correspondance adresse IPv6-adresse Ethernet pour `tuna` car cette adresse n'a pas été utilisée comme destination.

c) Indication de redirection

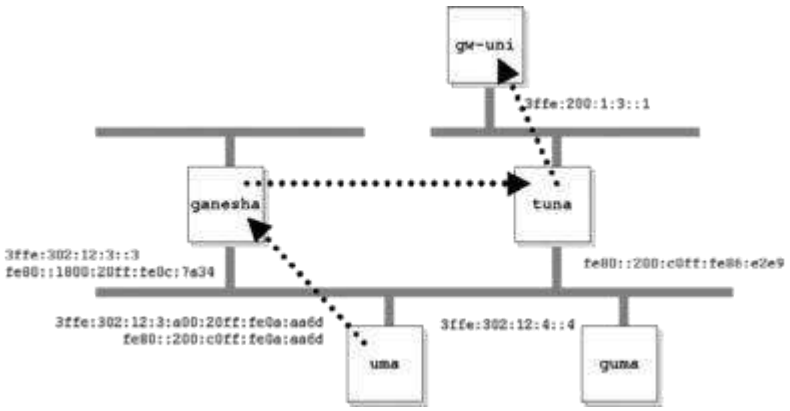


Figure 5-11 *Routage par défaut non optimal*

La configuration donnée figure Routage par défaut non optimal après la configuration des routes par défaut conduit aux tables de routage suivantes : la route par défaut d'`uma` pointe sur `ganesha`, et la route vers la machine externe `3ffe:200:1:3::1` sur `ganesha` pointe sur `tuna`.

La machine `uma` envoie un ping à la machine externe `3ffe:200:1:3::1` en utilisant la route par défaut, non optimale.

```
uma# ping6 3ffe:200:1:3::1
trying to get source for 3ffe:200:1:3::1
source should be 3ffe:302:12:3:a00:20ff:fe0a:aa6d
PING 3ffe:200:1:3::1: 56 data bytes
64 bytes from 3ffe:200:1:3::1: icmp6_seq=0 ttl=253 time=79.689 ms
.....
```

En observant le réseau, on trouve le trafic suivant.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 1a:0:20:c:7a:34 Type : 86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : 58 (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:200:1:3::1 (gw-uni)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0xd775
 Identificateur : 0x00d6 Numéro de séquence : 0x0000
 Données : Date : 0x3469a2a4.000d8c8b Remplissage ...
```

Le message ICMPv6 d'écho est transmis vers l'adresse Ethernet de `ganesha`, routeur par défaut d'`uma`.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 0:0:c0:86:e2:e9 Type : 86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
```

```
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:200:1:3::1 (gw-uni)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0xd775
 Identificateur : 0x00d6 Numéro de séquence : 0x0000
 Données : Date : 0x3469a2a4.000d8c8b Remplissage ...
```

`ganesha` retransmet le paquet IPv6 non modifié vers l'adresse Ethernet de `tuna`, qui est le premier relais sur la route vers la destination finale.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 8:0:20:a:aa:6d Type : 86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 160 octets (0x00a0) Protocole : (0x3a, ICMPv6)
 Source : fe80::1800:20ff:fe0c:7a34 (ganesha,lien-local)
 Desti. : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
ICMPv6
 Type : 137 (0x89, Redirection) Code : 0 Checksum : 0x869d
 Meilleur routeur : fe80::200:c0ff:fe86:e2e9 (tuna, lien-local)
 Destination : 3ffe:200:1:3::1 (gw-uni)
 Options :
  Type : 2 (Adresse physique cible) Lg : 8 octets (0x01) : 00-00-c0-86-e2-e9
  Type : 4 (Paquet ayant causé le message) Longueur : 112 octets (0x0e)
 Début du paquet IPv6 ayant causé le message
```

`ganesha` constate que le paquet d'écho a été réémis sur l'interface qui l'avait reçu, et génère donc un message de redirection vers `uma` pour lui indiquer qu'une meilleure route vers `3ffe:200:1:3::1` utilise le routeur `fe80::200:c0ff:fe86:e2e9`.

En IPv6 l'adresse Ethernet du routeur est fournie, ce qui évite une résolution supplémentaire.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 0:0:c0:86:e2:e9 Type : 86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:200:1:3::1 (gw-uni)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0xf51f
 Identificateur : 0x00d6 Numéro de séquence : 0x0001
Données : Date : 0x3469a2a6.000d6edd Remplissage ...
```

Le paquet de demande d'écho suivant est envoyé directement vers l'adresse Ethernet de `tuna`.

La table de routage d'`uma` est maintenant :

```
uma# netstat -nrf inet6
Routing tables IPv6:
Destination                Gateway                    Flags Refs Use Mtu  Interf
default                    fe80::1800:20ff:fe0c:7a34 UGS   0    17  1500 le0
3ffe:200:1:3::1            fe80::200:c0ff:fe86:e2e9  UGHD  0     2  1500 le0
fe80::200:c0ff:fe86:e2e9   0:0:c0:86:e2:e9           UHDL  1     0  1500 le0
fe80::1800:20ff:fe0c:7a34 1a:0:20:c:7a:34           UHDL  1     2  1500 le0
.....
```

d) Indication de redirection externe

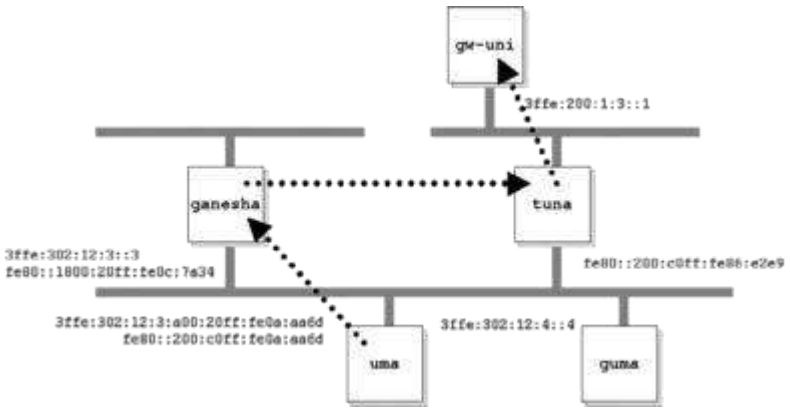


Figure 5-11 *Routage par défaut non optimal*

En IPv4, le mécanisme de redirection n'est prévu que pour les machines auxquelles on accède par l'intermédiaire d'un routeur ; la machine émettrice doit connaître les adresses IPv4 directement accessibles (le numéro de réseau correspondant au lien physique), un «ICMP Redirect» ne fonctionne pas pour celles-ci.

En IPv6, il n'est plus nécessaire de connaître tous les préfixes IPv6 correspondant au lien physique, une machine peut se contenter de connaître son adresse, un routeur par défaut, et envoyer tout le trafic inconnu sur ce routeur. Le mécanisme de redirection permettra de rediriger le trafic vers la destination la meilleure dans tous les cas.

Dans l'exemple correspondant à la figure Routage par défaut non optimal, supposons que `guma` a pour adresse globale sur l'interface partagée avec `uma` `3ffe:302:12:4::4`, et que `uma` n'a pas pris en compte dans ses tables de routage que le câble contient des machines appartenant à un préfixe `3ffe:302:12:4::4/64`. Ceci peut être dû au fait que `uma` n'analyse pas tous les préfixes, ou parce que le préfixe n'est pas annoncé sur le câble. Ce cas de figure est courant si le lien reliant `uma` et `guma` est un réseau ATM.

La machine `uma` veut accéder à `guma` :

```
uma# ping6 3ffe:302:12:4::4
trying to get source for 3ffe:302:12:4::4
source should be 3ffe:302:12:3:a00:20ff:fe0a:aa6d
PING 3ffe:302:12:4::4: 56 data bytes
64 bytes from 3ffe:302:12:4::4: icmp6_seq=0 ttl=255 time=7.267 ms
.....

Ethernet Src : 8:0:20:a:aa:6d Dst : 1a:0:20:c:7a:34 Type : 86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:302:12:4::4 (guma)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0x43cc
 Identificateur : 0x00fd Numéro de séquence : 0x0000
Données : Date : 0x337b4e95.0002725d Remplissage ...
```

Puisque uma ne sait pas que guma est directement accessible, le message ICMP d'écho est transmis vers l'adresse Ethernet de ganesha, routeur par défaut d'uma.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 0:0:c0:89:e2:e6 Type : 86dd
IPv6
 Version : 6 Classe : 0x00 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:302:12:4::4 (guma)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0x43cc
 Identificateur : 0x00fd Numéro de séquence : 0x0000
Données : Date : 0x337b4e95.0002725d Remplissage ...
```

ganesha retransmet le paquet non modifié vers guma.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 8:0:20:a:aa:6d Type : 86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 160 octets (0x00a0) Protocole : (0x3a, ICMPv6)
 Source : fe80::1800:20ff:fe0c:7a34
 Desti. : 3ffe:302:12:3:a00:20ff:fe0a:aa6d
ICMPv6
 Type : 137 (0x89, Redirection) Code : 0 Checksum : 0xfe45
 Meilleur routeur : 3ffe:302:12:4::4 (guma)
 Destination : 3ffe:302:12:4::4 (guma)
 Options :
  Type : 2 (Adresse physique cible) Lg : 8 octets (0x01) : 00-00-c0-89-e2-e6
  Type : 4 (Paquet ayant causé le message) Longueur : 112 octets (0x0e)
Début du paquet IPv6 ayant causé le message
```

De plus ganesha envoie à uma un message de redirection. C'est une redirection vers une machine sur le lien physique, ce qui est indiqué par le fait que les champs «Meilleur routeur» et «Destination» sont égaux (le premier relais est la destination finale).

```
Ethernet Src : 0:0:c0:89:e2:e6 Dst : 8:0:20:a:aa:6d Type : 86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:4::4
 Desti. : 3ffe:302:12:3:a00:20ff:fe0a:aa6d
ICMPv6
 Type : 129 (0x81, Réponse d'écho) Code : 0 Checksum : 0x42cc
 Identificateur : 0x00fd Numéro de séquence : 0x0001
Données : Celles de la demande
```

La réponse de guma parvient à uma (les messages de sollicitation de voisin échangés ne sont pas montrés dans cet exemple).

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 0:0:c0:89:e2:e6 Type : 86dd
IPv6
 Version : 6 Classe : 0xf0 Label : 000000
 Longueur : 64 octets (0x0040) Protocole : (0x3a, ICMPv6)
 Source : 3ffe:302:12:3:a00:20ff:fe0a:aa6d (uma)
 Desti. : 3ffe:302:12:4::4 (guma)
ICMPv6
 Type : 128 (0x80, Demande d'écho) Code : 0 Checksum : 0x43be
 Identificateur : 0x00fd Numéro de séquence : 0x0001
Données : Date : 0x337b4e96.00027269 Remplissage ...
```

Les demandes d'écho suivantes sont adressées directement à l'adresse Ethernet de `guma`. La table de routage d'`uma` est maintenant :

```
uma# netstat -nrf inet6
Routing tables IPv6:
Destination                Gateway                    Flags Refs Use Mtu  Interf
default                    fe80::1800:20ff:fe0c:7a34 UGS   0    17  1500 le0
3ffe:302:12:4::4           0:0:c0:89:e2:e6           UHDL  0     3  1500 le0
3ffe:200:1:3::1            fe80::200:c0ff:fe86:e2e9  UGHD  0     2  1500 le0
fe80::200:c0ff:fe86:e2e9   0:0:c0:86:e2:e9           UHDL  1     0  1500 le0
fe80::1800:20ff:fe0c:7a34 1a:0:20:c:7a:34           UHDL  1     2  1500 le0
.....
```

On voit qu'il y a maintenant une ligne avec le drapeau `L` pour la machine `guma`. Le drapeau `L` et l'absence de drapeau `G` indiquent une machine directement accessible, sans routeur intermédiaire.

2) Configuration automatique

Traditionnellement, la configuration d'une interface réseau d'une machine demande une configuration manuelle. C'est un travail souvent long et fastidieux. Avec IPv6, cette configuration est automatisée, introduisant par là-même des caractéristiques de fonctionnement immédiat (*plug and play*) à l'interface réseau. La configuration automatique signifie qu'une machine obtient toutes les informations nécessaires à sa connexion à un réseau local IP sans aucune intervention humaine. Dans le cas idéal, un utilisateur quelconque déballe son nouvel ordinateur, le connecte au réseau local et le voit fonctionner sans devoir y introduire des informations de "spécialiste". Nous avons vu comment les routes étaient installées dans la table de routage des machines. Nous allons maintenant étudier l'autre aspect de l'auto-configuration de IPv6 qui est l'auto-configuration d'adresses. Celle-ci a pour objectif :

- l'acquisition d'une adresse quand une machine est attachée à un réseau pour la première fois ;
- l'obtention de la nouvelle adresse suite à la renumérotation des machines du site après un changement d'opérateur. L'opération de renumérotation revient concrètement à changer la partie haute de l'adresse. L'auto-configuration d'adresses va servir de vecteur dans l'attribution du nouveau préfixe.

Le processus d'auto-configuration d'adresse d'IPv6 comprend la création d'une adresse lien-local, la vérification de son unicité et la détermination d'adresses unicast globales. IPv6 spécifie deux méthodes d'auto-configuration pour l'adresse unicast globale :

- l'auto-configuration sans état (*stateless auto-configuration*, RFC 2462) ; elle est utilisée quand la gestion administrative des adresses attribuées n'est pas nécessaire au sein d'un site. Ces mécanismes sont décrits dans la suite de ce chapitre.
- l'auto-configuration avec état (*stateful auto-configuration*) ; elle est retenue lorsqu'un site demande un contrôle strict de l'attribution des adresses (cf. DHCPv6).

Le rôle du routeur est important dans l'auto-configuration. Il dicte à la machine, par des bits (cf. Annonce du routeur) de l'en-tête du message d'annonce de routeurs, la méthode à retenir et fournit éventuellement les informations nécessaires à sa configuration. Le bit `M` (*Managed address configuration*)

mis à 1 indique que l'équipement ne doit pas construire lui-même l'adresse à partir de son identifiant d'interface et des préfixes reçus, mais doit explicitement demander son adresse auprès d'une application d'un serveur d'adresses. Le bit ○ (*Other stateful configuration*) indique que l'équipement doit interroger le serveur de configuration pour obtenir des paramètres autre que l'adresse. L'algorithme de la procédure d'auto-configuration d'adresse se décompose de la manière suivante :

- La toute première étape consiste à créer l'adresse lien-local.
- Une fois l'unicité de cette adresse vérifiée, la machine est en mesure de communiquer avec les autres machines du lien.
- La machine doit chercher à acquérir un message d'annonce du routeur pour déterminer la méthode d'obtention de l'adresse unicast globale.
- S'il y a un routeur sur le lien, la machine doit appliquer la méthode indiquée par le message d'annonce de routeurs, à savoir :
    - l'auto-configuration sans état,
    - l'auto-configuration avec état.
- En l'absence de routeur sur le lien, la machine doit essayer d'acquérir l'adresse unicast globale par la méthode d'auto-configuration avec état. Si la tentative échoue, c'est terminé. Les communications se feront uniquement sur le lien avec l'adresse lien-local. La machine n'a pas une adresse avec une portée qui l'autorise à communiquer avec des machines autres que celles du lien.

A) Création de l'adresse lien-local

À l'initialisation de son interface, la machine construit un identifiant pour l'interface qui doit être unique au lien. Cet identifiant utilise l'adresse EUI-64. Le principe de base de la création d'adresse IPv6 est de marier un préfixe avec l'identifiant. L'adresse lien-local est créée en prenant le préfixe lien-local (`FE80::/64`) qui est fixé. L'adresse ainsi constituée est encore interdite d'usage. Elle possède un état provisoire car la machine doit vérifier l'unicité de cette adresse sur le lien au moyen de la procédure de détection d'adresse dupliquée. Si la machine détermine l'adresse lien-local n'est pas unique, l'auto-configuration s'arrête et une intervention manuelle est nécessaire.

Une fois que l'assurance sur l'unicité de l'adresse lien-local est obtenue, l'adresse provisoire devient une adresse valide pour l'interface. La première phase de l'auto-configuration est achevée.

B) Détection d'adresse dupliquée

Pour vérifier l'unicité des adresses lien-local ou unicast, les machines doivent exécuter un algorithme de Détection d'Adresse Dupliquée (DAD) avant de les utiliser. L'algorithme utilise les messages ICMPv6 sollicitation d'un voisin et annonce d'un voisin. Si une adresse déjà en service est découverte, elle ne pourra être attribuée à l'interface. L'auto-configuration s'arrête et une intervention humaine devient obligatoire. Une adresse est qualifiée de "provisoire" pendant l'exécution de l'algorithme DAD et ce jusqu'à la confirmation de son unicité. Une adresse provisoire est assignée à une interface uniquement pour recevoir les messages de sollicitation et d'annonce d'un voisin. Les autres messages reçus sont ignorés. L'algorithme DAD consiste à envoyer un message sollicitation d'un voisin avec dans le champ adresse de la cible l'adresse provisoire. Afin de distinguer l'algorithme DAD de celui de découverte des voisins, le paquet IPv6 contenant un message de sollicitation d'un voisin a comme adresse de source l'adresse indéterminée. Trois cas se présentent :

- Un message annonce d'un voisin est reçu : l'adresse provisoire est utilisée comme adresse valide par une autre machine. L'adresse provisoire n'est pas unique et ne peut être retenue.
- Un message sollicitation d'un voisin est reçu dans le cadre d'une procédure DAD; l'adresse provisoire est également une adresse provisoire pour une autre machine. L'adresse provisoire ne peut être utilisée par aucune des machines.
- Rien n'est reçu au bout d'une seconde (valeur par défaut) : l'adresse provisoire est unique, elle passe de l'état de provisoire à celle de valide et elle est assignée à l'interface.

A noter que cet algorithme n'offre pas une fiabilité absolue, notamment lorsque le lien est coupé.

C) Auto-configuration sans état

L'auto-configuration sans état (RFC 2462) ne demande aucune configuration manuelle des machines, une configuration minimum pour les routeurs et aucun serveur supplémentaire. Elle se sert du protocole ICMPv6 et peut fonctionner sans la présence de routeurs. Elle nécessite cependant un sous-réseau à diffusion. Cette méthode ne s'applique que pour les machines et ne peut être retenue pour la configuration des routeurs. Le principe de base de l'auto-configuration sans état est qu'une machine génère son adresse IPv6 à partir d'informations locales et d'informations fournies par un routeur. Le routeur fournit à la machine les informations sur le sous-réseau associé au lien, il donne le préfixe.

Comme pour la création de l'adresse lien-local, l'adresse unicast globale est obtenue en concaténant le préfixe avec l'identifiant de l'interface. Le préfixe provient du message d'annonce de routeurs et plus précisément de l'option «information sur le préfixe». Bien qu'il faille vérifier l'unicité de toutes les adresses unicast, dans le cas d'une adresse unicast obtenue par auto-configuration sans état cela n'est pas obligatoire. En effet, l'unicité de l'identifiant de l'interface a déjà été contrôlé dans l'étape de création de l'adresse lien-local. L'identifiant étant le même, il n'y a plus aucune ambiguïté sur son unicité. L'adresse unicast globale constituée est aussi unique que celle lien-local.

La renumérotation des machines d'un lien s'effectue au moyen des routeurs qui passent les adresses utilisées dans un état déprécié et annoncent en même temps le nouveau préfixe. Les machines pourront recréer une adresse préférée.

D) Exemples de configuration sans état

La machine à l'activation de l'interface réseau crée l'adresse lien-local provisoire et débute l'algorithme DAD. Elle émet un message de sollicitation d'un voisin à l'adresse multicast sollicité associée à son adresse provisoire. Son adresse de source est indéterminé car son état est encore provisoire pour le moment et ne sert que pour la réception. L'adresse dont l'unicité est vérifiée est placée dans le champ cible.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 33:33:ff:a:aa:6d Type : 0x86dd
IPv6
 Version : 6 Priorité : 0xf0 Label: 000000
 Longueur : 24 octets (0x0018) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0x0ff)
 Source : ::
 Desti. : ff02::1:ff0a:aa6d (multicast sollicité associé à l'adresse cible)
ICMPv6
 Type : 135 (0x87, Sollicitation d'un voisin) Code : 0 Checksum : 0xfe37
 cible : fe80::0a00:20ff:fe0a:aa6d (uma, lien-local)

0000: 6f 00 00 00 00 18 3a ff 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 ff 02 00 00 00 00 00 00
0020: 00 00 00 01 ff 0a aa 6d|87 00 fe 37 00 00 00 00
0030: fe 80 00 00 00 00 00 00 0a 00 20 ff fe 0a aa 6d
```

Ce message ne peut être utilisé pour mettre à jour le cache de résolution d'adresses d'un voisin car l'adresse de la source a un statut de provisoire, en conséquence l'option adresse physique de la source ne figure pas dans le message. Le message de sollicitation d'un voisin sera émis au moins deux fois. Si rien n'est reçu, l'identifiant sera considéré unique et l'adresse passera dans l'état valide. Par contre, si elle reçoit le message annonce d'un voisin comme ci-dessous, l'adresse est entrée en collision avec une adresse valide utilisée par un voisin. L'adresse ne peut être affectée à l'interface. Une intervention humaine devient nécessaire.

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 33:33:0:0:0:1 Type : 0x86dd
IPv6
 Version : 6 Priorité : 0xf0 Label: 000000
 Longueur : 32 octets (0x0020) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0x0ff)
 Source : fe80::a00:20ff:fe0a:aa6d
 Desti. : ff02::1 (multicast, tous les noeuds du lien)
ICMPv6
 Type : 136 (0x88, Annonce d'un voisin) Code : 0 Checksum : 0xe036
 Bits (0x7) R = 0, S = 0, O = 1
 cible : fe80::a00:20ff:fe0a:aa6d
 Option :
  Type : 2 (Adresse physique cible) Lg : 8 octets (0x01) : 08-00-20-0a-aa-6d

0000: 6f 00 00 00 00 20 3a ff fe 80 00 00 00 00 00 00
0010: 0a 00 20 ff fe 0a aa 6d ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 01|88 00 e0 36 20 00 00 00
0030: fe 80 00 00 00 00 00 00 0a 00 20 ff fe 0a aa 6d|
0040: 02 01 08 00 20 0a aa 6d
```

Comme le message sollicitation d'un voisin ne comportait pas d'adresse de source, le message annonce d'un voisin est émis au groupe de tous les nœuds du lien (`ff02::1`).

Après avoir exécuté l'algorithme DAD avec l'adresse lien-local comme l'exemple précédent l'a montré (en supposant que l'adresse n'est pas entrée en collision), la machine émet un message sollicitation du routeur à tous les routeurs du lien (`ff02::2`).

```
Ethernet Src : 8:0:20:a:aa:6d Dst : 33:33:0:0:0:2 Type : 0x86dd
IPv6
 Version : 6 Priorité : 0xf0 Label: 000000
 Longueur : 16 octets (0x0010) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0x0ff)
 Source : fe80::a00:20ff:fe0a:aa6d (uma, lien-local)
 Desti. : ff02::2 (multicast, tous les routeurs du lien)
ICMPv6
 Type : 133 (0x85, Sollicitation du routeur) Code : 0 Checksum : 0xd63e
 Option :
  Type : 1 (Adresse physique source) Lg : 8 octets (0x01) : 08-00-20-0a-aa-6d

0000: 6f 00 00 00 00 10 3a ff fe 80 00 00 00 00 00 00
0010: 0a 00 20 ff fe 0a aa 6d ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 02|85 00 d6 3e 00 00 00 00|
0030: 01 01 08 00 20 0a aa 6d
```

Si un routeur est présent, un message annonce de routeur est reçu par la machine se configurant. Elle y trouve les bits `M`, `O` et les informations sur les préfixes du lien.

```
Ethernet Src : 1a:0:20:c:7a:34 Dst : 33:33:0:0:0:1 Type : 0x86dd
IPv6
 Version : 6 Priorité : 0xf0 Label: 000000
 Longueur : 56 octets (0x0038) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 255 (0x0ff)
 Source : fe80::1800:20ff:fe0c:7a34 (ganesha, lien-local)
 Desti. : ff02::1 (multicast, tous les noeuds du lien)
ICMPv6
 Type : 134 (0x86, Annonce de routeurs) Code : 0 Checksum : 0x773c
 Nombre de sauts : 0 (non précisé) Gestion d'adresse : 0 (Pas de DHCP)
 Validité : 6000 secondes (0x1770) Timers : 0, 0 (non précisés)
 Options :
  Type : 1 (Adresse physique source) Lg : 8 octets (0x01) : 1a-00-20-0c-7a-34
  Type : 3 (Information sur le préfixe) Lg : 32 octets (0x04)
  Drapeaux : L=1, A=1 Durée de validité : -1, -1 (infinie)
  Préfixe : 3ffe:302:12:3::/64

0000: 6f 00 00 00 00 38 3a ff fe 80 00 00 00 00 00 00
0010: 18 00 20 ff fe 0c 7a 34 ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 01|86 00 77 3c 00 00 17 70
0030: 00 00 00 00 00 00 00 00|01 01 1a 00 20 0c 7a 34|
0040: 03 04 40 c0 ff ff ff ff ff ff ff ff 00 00 00 00
0050: 3f fe 03 02 00 12 00 03 00 00 00 00 00 00 00 00
```

Le message annonce de routeurs est émis vers le groupe de tous les nœuds IPv6 du lien. Le drapeau `A` étant positionné, le préfixe annoncé peut alors servir à la construction de l'adresse unicast globale. La durée de validité de cette adresse n'est pas limitée. La machine se configurant aura donc l'adresse `3ffe:302:12:3:a00:20ff:fe0a:aa6d`.

3) Configuration avec état : DHCP v6

A) Principes

L'auto-configuration avec état vise à réduire les efforts d'installation des machines IPv6, tout comme l'auto-configuration sans état d'ailleurs. A la différence de cette dernière, elle offre une information de configuration plus riche et un contrôle sur l'affectation des paramètres de configuration.

Un paramètre de configuration est une information utile à une machine pour configurer son interface réseau de manière à ce qu'elle puisse communiquer sur son lien ou sur l'Internet. L'ensemble des paramètres de configuration comprend notamment les informations :

- d'adressage, de routage,
- du service de noms (DNS),
- du service d'information réseau (NIS)
- etc.

Attention, cependant, les informations de routage ne sont pas fournies en IPv6 par les mécanismes d'auto-configuration avec ou sans état mais par la procédure de découverte des routeurs d'ICMPv6. Les deux techniques d'auto-configuration ne sont pas exclusives et peuvent coexister dans un même environnement. Par exemple, une machine peut obtenir son adresse unicast globale par l'auto-configuration sans état et récupérer les informations sur le service de noms (DNS) par l'auto-configuration avec état.

Tout le mécanisme d'auto-configuration avec état est bâti sur le modèle du client-serveur et repose sur l'utilisation du protocole DHCPv6 (*Dynamic Host Configuration Protocol for IPv6*) comme DHCP pour IPv4. La principale différence vient d'une simplification du code : le DHCP d'IPv4 n'est qu'une extension du protocole bootp avec donc des fonctionnalités limitées. Le protocole DHCPv6 sert à remettre des paramètres de configuration d'un serveur DHCP à un client IPv6.

Le client IPv6 représente une machine candidate à une connectivité globale IPv6 et demandeur d'informations de configuration pour activer cette connectivité. Le serveur DHCP constitue un point central regroupant les paramètres de configuration. Il ne se trouve pas forcément sur le même lien que le client et dans ce cas, les échanges DHCP peuvent passer par un relais. Le serveur peut maintenir la configuration de machines situées sur plusieurs liens différents. Le client DHCP doit maintenir une connectivité directe soit avec un relais DHCP, soit avec le serveur DHCP lui-même.

Le relais fonctionne comme un "proxy" DHCP, c'est-à-dire son action se limite à la transmission des messages DHCP. Il est transparent aux informations échangées et ne modifie en rien les messages DHCP du serveur et du client. Un relais encapsule les messages DHCP provenant du client dans un message DHCP au format particulier (cf figure Structure des messages de relais) et effectue l'opération inverse pour les messages provenant du serveur (à savoir désencapsule les messages du serveur pour leur remise au client). Le serveur constitue le message que doit recevoir le client, et, faute de pouvoir le joindre, il constitue un message DHCP pour le relais qui contiendra le message DHCP du client.

Afin de connaître l'état des ressources gérées (représentées par les paramètres de configuration), le serveur DHCP maintient une liste d'associations entre le paramètre attribué et le client. Comme l'adresse unicast du client est une ressource dépendant du serveur, celle-ci n'est pas utilisable par le serveur DHCP pour identifier un client. Le serveur identifie donc le client par un identifiant unique à usage exclusif de DHCP : le DUID (*DHCP Unique IDentifier*). Cet identifiant est généré par le client et ne doit plus en changer dans le temps. Le DUID concerne le client (le nœud) et non une interface du client. Plusieurs schémas de génération sont proposés reposant sur l'adresse de lien-local complétée par un élément qui garantit l'unicité comme par exemple une estampille temporelle. Une fois qu'un client a un DUID, il doit le conserver même s'il change d'adresse lien local.

L'adresse lien-local du client peut servir d'identifiant mais il n'existe aucune garantie qu'elle soit unique au niveau d'un domaine. Son unicité est assurée uniquement au niveau du lien. Cependant, elle présente les intérêts suivants :

- elle est déterminée par le client lors de la première phase de l'auto-configuration (cf See Configuration automatique et contrôle),
- elle est permanente au client (il n'y a pas de renumérotation ou de changement d'adresse tant que la carte réseau n'est pas changée).

Les affectations d'adresses à un client sont gérées par le serveur avec une notion de container appelé IA (*Identity Association*). Une IA est définie comme une liste (pouvant être vide) d'adresses IPv6. L'idée est que chaque client a une IA par interface et que cette IA reste affectée en permanence à l'interface. Ainsi, par ce container, la gestion de la durée de vie des adresses ou la renumérotation du client s'effectue par le serveur. Cette notion simplifie le format des messages et le contrôle des adresses.

Conformément au modèle client-serveur, les échanges se composent de requêtes et de réponses. Une transaction est souvent entamée à l'initiative d'un client par une requête DHCP. La fiabilité de la transaction est assurée par le client, qui répète sa requête DHCP jusqu'à recevoir une réponse ou obtenir la certitude que le serveur DHCP est indisponible. DHCPv6 est un protocole d'application qui se sert du protocole de transport UDP. Un client envoie les requêtes sur le port 547 du serveur et recevra les réponses par le port 546. Les messages UDP seront encapsulés classiquement dans des paquets IPv6. En plus des communications point-à-point, DHCPv6 s'appuie également sur des communications multi-destination (multicast) pour la découverte des serveurs d'un site.

B) Format des messages DHCPv6

L'ensemble des éléments du protocole DHCPv6 est décrit dans le document (RFC 3315). A l'instar de nombreux protocoles de l'Internet, le protocole d'échange d'informations est découplé de l'information elle-même. La nature des informations échangées peut changer et évoluer rapidement sans impacter les mécanismes de cet échange. Cette séparation assure au protocole une certaine stabilité et une propriété d'extensibilité. On retrouve dans la structure des unités de protocole ce découpage. Une unité de protocole DHCP suit le schéma classique des unités de protocole : un en-tête pour les informations du protocole lui-même, une charge utile pour les informations applicatives.

Dans la terminologie DHCP, une unité de protocole DHCPv6 est désignée par le terme message. Chaque message DHCP a un format d'en-tête identique. De ce point de vue, DHCP reprend les principes qui ont guidé à la conception du format du segment TCP : un seul format pour l'ensemble des fonctions de TCP. La motivation tient à la simplification du processus de développement du protocole.

La figure Structure des messages DHCPv6 présente la structure d'un message DHCPv6. La partie en-tête se divise en trois parties :

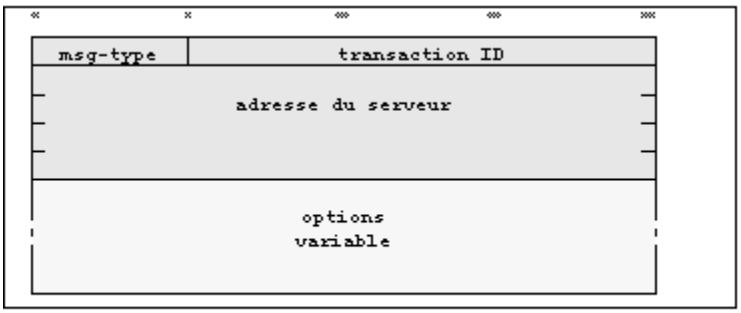


Figure 5-12. Structure des messages DHCPv6

- L'information de commande codée sur un mot de 32 bits désigne la fonction DHCP concernée par l'échange. Cette partie contient notamment le type de message, l'identification de l'échange. Le premier champ de cette partie est toujours le champ type de message codé sur 8 bits et qui définit la fonction du message dans le protocole. Le champ transaction-ID a comme rôle comme son nom l'indique, d'identifier la transaction vis à vis du client. Il sert au client à rapprocher les réponses reçues des demandes qu'il a faites.
- L'information d'adressage sert à indiquer l'adresse IPv6 du serveur d'un échange DHCP. L'adresse du serveur contient l'adresse de l'interface utilisée par le serveur dans la transaction.
- Le champ options sert à véhiculer les informations de configurations. Cette partie est de taille variable. Son codage suit le schéma classique "TLV" à savoir le type de l'option, la longueur en octet du champ valeur qui suit et le champ valeur du paramètre. Le champ type est toujours codé sur 2 octets. Le champ longueur sur 2 octets est toujours présent, même en l'absence de valeur ou pour une valeur de longueur fixe

Les messages utilisés pour la communication serveur - relais sont différents. Un message de relais contient l'information de remise du message DHCP du serveur au client. Les messages de relais suivent le format de la See Structure des messages de relais. Le préfixe du lien indique l'interface du relais par laquelle le message DHCP a été reçu ou par laquelle il doit être envoyé. L'adresse lien-local du client contient l'adresse de l'interface à configurer du client. Le message DHCP est encapsulé dans le message de relais sous forme d'une option.

Le protocole DHCPv6 met en jeu 12 messages DHCP différents :

- Sollicitation DHCP (DHCP Solicit) :
  Message d'interrogation de présence de serveurs DHCP. Il est émis vers un serveur ou un relais DHCP. Un client émet un tel message pour localiser les serveurs DHCP.

- Annonce DHCP (DHCP Advertise) :
  Message de présence de serveurs DHCP. Il est émis en réponse à un message sollicitation DHCP afin de communiquer l'adresse IP d'un serveur DHCP. Le destinataire est le client s'il est sur le même lien que le serveur sinon ce message est adressé au relais du client.

- Requête DHCP (DHCP Request) :
  Message de demande de paramètres de configuration de la part d'un client sans adresse.
- Confirmation DHCP (DHCP Confirm) :
  Message de demande de confirmation de validité des paramètres alloués au client.
- Renouvellement DHCP (DHCP Renew) :
  Message de demande de prolongation de l'adresse IP affectée
- Ré affectation DHCP (DHCP Rebind) :
  Identique au précédent message mais un autre serveur DHCP peut répondre, pas obligatoirement celui qui a alloué l'adresse IP.
- Réponse DHCP (DHCP Reply) : Message émis par le serveur suite à une demande du client. Il contient les valeurs des paramètres de configuration demandés.
- Libération DHCP (DHCP Release) :
  Message d'indication du client de libération des adresses IP préalablement allouées par le serveur.
- Refus DHCP (DHCP Decline) :
  Message d'indication du client qu'une ou plusieurs adresses affectées sont déjà utilisées sur son lien.
- Notification de reconfiguration DHCP (DHCP Reconfigure-init) :
  Message émis par le serveur pour informer le client qu'il a de nouvelles valeurs pour les paramètres de configuration. Le client doit alors commencer une nouvelle transaction pour acquérir ces informations.
- Encapsulation relais (Relay-Forward) :
  Message du relais pour véhiculer les messages du client vers le serveur. Le message du client est encapsulé dans ce message.
- Encapsulation serveur (Relay-Reply) :
  Message généré par le serveur contenant un message pour le client. Ce message est à destination du relais qui extraira un message pour le client afin de le transmettre sur le lien du client.

Les informations échangées entre le client et le serveur DHCP se font au moyen d'options. Les informations se divisent en trois catégories (cf. tableau Options de DHCPv6).

- les informations temporaires. Elles concernent les ressources réseaux demandées par le client et prêtées par le serveur pour une durée déterminée. Actuellement, le seul type de ressource temporaire est l'adresse IP dont la gestion est faite par la notion d'IA.
- Les informations générales. Elles traitent de l'ensemble des paramètres de configuration habituellement fournies pour la configuration d'une machine IPv6.
- Les informations spécifiques à DHCP.

Options de DHCPv6

Désignation	Définition
Association d'Identification (IA)	Liste des adresses IPv6 d'une interface du client.
Requêtes d'options (ORO)	Liste des informations de configuration demandées par le client.
Serveur de nom de domaine	Liste des serveurs DNS autorisés pour le client.
Recherche de domaine	Liste de noms de domaines qu'un client peu utiliser dans ses recherches de noms DNS.
Message client	Pour l'encapsulation du message DHCP du client par le relais
Message serveur	Pour l'encapsulation du message DHCP du serveur via un relais
DSTM adresse IPv4	Informe le client que l'adresse allouée est une adresse IPv4 mappée IPv6
Authentification	Pour authentification de la source du message DHCP et de la validation de son intégrité.
Unicast serveur	Pour indiquer au client l'adresse unicast du serveur afin d'établir des communications sans relais. Le client doit avoir une adresse unicast valide dans ce cas.
Identifiant DHCP (DUID)	Identifiant permanent du client.
Préférence	Moyen donné au client de choisir le serveur DHCP. Le serveur sélectionné aura en charge de fournir les paramètres de configuration à ce client.

C) Acquisition de l'adresse du serveur DHCP

En premier lieu la machine désirant se configurer doit localiser un serveur. Pour cela, elle envoie le message sollicitation DHCP. Pour construire ce message, elle remplit le champ d'identification de la transaction et ajoute l'option DUID. Ensuite elle envoie sa requête en mode multicast vers tous les agents DHCP du lien (adresse `FF02::1:2`). Le groupe des agents comprend à la fois les serveurs et les relais DHCP d'un domaine d'administration. Ce message ne peut être routé car le client utilise son adresse lien local. Si

le serveur n'est pas sur le même lien que le client, un relais doit y être. C'est pourquoi l'adresse multicast est de portée locale au lien.

Dans le cas où cette transaction passe par un relais, le message de sollicitation est encapsulé dans le champ option d'un message encapsulation relais avec comme préfixe celui de l'interface sur laquelle le relais a reçu ce message. Selon la configuration des relais, le message est émis à un serveur DHCP en mode unicast vers une liste de serveurs ou en mode multicast. Dans ce dernier cas, l'adresse multicast utilisée est celle du groupe des serveurs DHCP (`FF05::1:3`) du site ou une adresse de groupe propre au site.

Quand le message sollicitation DHCP arrive au serveur, il renvoie un message annonce DHCP en prenant bien soin d'indiquer son adresse dans le champ "adresse du serveur" et de recopier l'identification de transaction du message de sollicitation. Le serveur peut mettre une option préférence. La préférence codée sur 8 bits indique la volonté du serveur de servir le client. Le client doit retenir le serveur qui a la valeur de préférence la plus forte. Une certaine forme de répartition des clients entre les serveurs peut être ainsi effectuée. Le message d'annonce DHCP est ensuite émis en unicast au client directement ou via le relais.

Comme les communications s'appuient sur le protocole de transport non fiable UDP, DHCP inclus un mécanisme de fiabilisation très simple. Chaque message de requête DHCP émis doit être acquitté par un autre message DHCP. L'initiateur d'une transaction DHCP détecte la perte au moyen d'un temporisateur. A l'expiration du temporisateur, le message DHCP est réémis à l'identique.

D) Acquisition de l'adresse unicast globale

Dès que le client a trouvé un serveur, il peut à présent obtenir les informations de configuration. Pour ce faire, il envoie une requête DHCP au serveur «préféré», en remplissant le champ options avec les paramètres qu'il souhaite obtenir en retour. Lorsqu'une telle requête arrive au serveur, celui-ci construit un message de réponse DHCP en y incluant sous forme d'options, les informations de configuration demandées par le client. A chaque requête du client, l'option DUID d'identification du client est présente.

Dans le cas d'une acquisition d'une adresse unicast globale, l'option Association d'Identification (IA) est placée dans le message de requête DHCP. Cette option IA contient toutes les informations relatives au contrôle des allocations des adresses IP. Le serveur complète l'option et conserve une trace de ce prêt. Après vérification que le message est bien une réponse à sa demande, le client regarde les différents champs et peut ainsi commencer sa configuration. Ensuite le client doit entamer une procédure DAD (Détection d'Adresse Dupliquée). En cas d'échec de la procédure, l'adresse doit être restituée au serveur au moyen du message de refus DHCP. Dans le cas des messages renouvellement DHCP ou réaffectation DHCP, si l'adresse donnée dans l'option IA est celle d'une adresse déjà assignée à l'interface (cette situation correspond à un prolongement de la durée de vie), la procédure DAD n'a pas lieu d'être déroulée.

Un client peut restituer son adresse IP selon 2 méthodes :

- par une notification explicite au moyen du message de libération DHCP. Le message de libération est acquitté par le message de réponse DHCP. L'adresse IP restituée est mise dans l'option IA.

- en n'étendant pas la durée de vie de la validité de son adresse. Lorsque la durée de vie de l'état préféré de l'adresse est consommée, le client doit prolonger auprès du serveur les durées de vie des états de son adresse. Sinon une fois que la période de validité de l'adresse est épuisée, l'adresse ne doit plus être utilisée. Elle est potentiellement affectable par le serveur à un autre client.

E) Exemple

Pour illustrer le fonctionnement du protocole DHCPv6 nous allons prendre le cas simple (absence de relais) où une machine après avoir calculé son adresse lien-local souhaite récupérer son adresse unicast globale d'un serveur DHCP qui se situe sur le même lien. La figure Transactions DHCPv6 sans relais présente les échanges nécessaires.

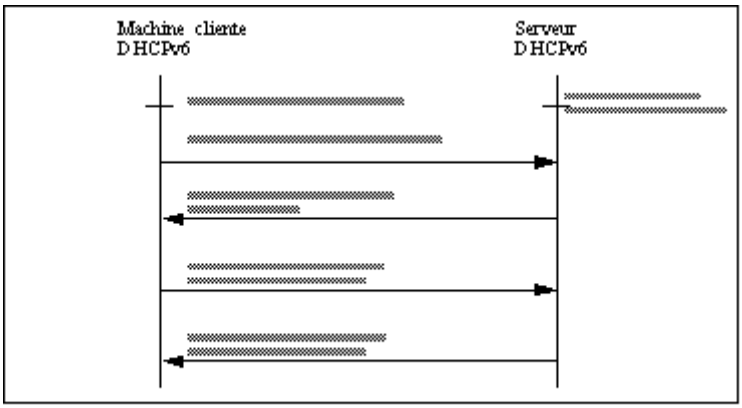


Figure 5-14 . Transactions DHCPv6 sans relais

F) Mise à jour de configuration

DHCP prévoit également que des transactions peuvent être déclenchées à l'initiative du serveur. Cette situation correspond aux cas de l'ajout d'un nouveau réseau, de la renumérotation de réseau ou du changement de situation des serveurs (d'impression, de noms, etc.) ou à l'ajout de nouveaux services. La méthode proposée par le protocole DHCP repose sur l'envoi du message de notification de reconfiguration DHCP. Quand un client reçoit un tel message, il doit commencer une transaction requête/réponse DHCP avec le serveur. Celui-ci indique par l'option ORO quelles sont informations de configurations qui ont changées ou quelles sont celles qui ont été ajoutées. Le client pourra alors inclure dans sa demande les options correspondantes afin de récupérer les nouvelles valeurs. Le message de notification de reconfiguration DHCP est émis par le serveur en unicast à chaque client impliqué par la reconfiguration.

G) Authentification des messages

Un administrateur peut vouloir assurer l'authentification de la source et l'intégrité du contenu des messages DHCP pour se protéger des attaques décrites dans See [Prigent-id]. Par exemple, cette fonctionnalité est indispensable dans le cas du message Démarrage d'une reconfiguration DHCP, ceci empêche que des clients entame des transactions de reconfiguration, la source de ce message n'est pas un serveur DHCP. Le mécanisme d'authentification de DHCPv6 repose sur le même principe d'authentification de DHCP de IPv4 (RFC 3118) à savoir sur une option d'authentification. DHCP et mobilité

L'atout majeur de l'auto-configuration est, comme son nom l'indique, la possibilité pour toute nouvelle machine d'obtenir, sans l'intervention humaine, une identité IP sur le réseau où elle se trouve. Un nœud mobile doit au moyen d'un protocole transmettre à son agent mère (cf. Mobilité dans IPv6) sa nouvelle adresse. Dans See [Dupont-id], il est proposé d'utiliser DHCPv6 pour assurer cet échange.

H) Renumérotation de réseau avec DHCP

DHCP peut servir d'outil pour la renumérotation d'un réseau. Cette tâche peut être réalisée selon deux méthodes:

- renumérotation passive. Pour que cette méthode soit utilisable, les durées de vie des adresses allouées au client doivent être courtes. Quand l'administrateur souhaite renuméroter un réseau, il met, sur ses serveurs, une valeur nulle à la durée de vie des adresses réseau en service. Les clients ne pourront plus prolonger la durée de vie de leur adresse. Ils demanderont donc une adresse dans le nouveau plan d'adressage. Quand la durée de vie de toutes les adresses de l'ancien plan d'adressage aura expiré, le réseau pourra être considéré renuméroté.
- renumérotation active. L'administrateur force la renumérotation du réseau au moyen de la reconfiguration de DHCP. Les serveurs génèrent un message démarrage d'une reconfiguration DHCP pour chacun de leur client devant subir la renumérotation. Ceux-ci entament une transaction Requête/Réponse DHCP portant sur l'adresse. La réponse du serveur contient l'adresse originale avec une durée de vie mise à nulle et la nouvelle adresse valide.

I) Avenir de DHCPv6

Au moment de la rédaction de ce livre, le protocole était toujours dans un état de document de travail. Ceci est principalement dû à l'inhérente complexité du protocole. Sa mise en œuvre est compliquée par le nombre important de messages avec des formats différents et des en-têtes de longueur variable. Seules deux distributions sont disponibles (dont une incomplète). Le manque d'expérience sur ce protocole est l'autre raison de son blocage à l'état de document de travail.

On peut se demander s'il existe un véritable intérêt pour DHCPv6, en effet ce protocole est fortement lié à IPv4 et a été largement utilisé pour parer au manque d'adresses. Initialement, les stations de travail Sun

utilisaient RARP comme protocole pour trouver leur adresse IP (en fonction de l'adresse MAC de la station) puis calculer quelques paramètres essentiels (comme le nom du fichier contenant le programme d'amorçage, composé à partir de l'adresse MAC, ce fichier étant téléchargé dans la phase suivante par TFTP). Ce système n'était pas très satisfaisant. L'IETF a donc standardisé un protocole fournissant un service identique, BOOTP (RFC 0951). Ce protocole a été ensuite étendu pour fournir d'autres paramètres que le nom du fichier du programme d'amorçage, regroupés sous le nom «BOOTP Vendor Information Extensions» (RFC 1048).

DHCP est un extension de BOOTP gérant l'allocation dynamique d'adresses IP (les leases/baux). Les implémentations de DHCP sont conçues afin de gérer ces adresses comme une ressource rare. Dans le cadre d'IPv6, les problèmes sont totalement différents et donc les protocoles devraient aussi l'être :

- tous les nœuds peuvent communiquer localement en formant des adresses de portée réduite générées automatiquement ou conservées dans de la mémoire stable (cas général des routeurs) ;
- une machine peut utiliser la configuration sans état pour acquérir des adresses globales ;
- un sous-réseau IPv6 dispose de 2^64 adresses, donc une adresse n'est pas une ressource rare ;
- enfin le (RFC 2462) définit la configuration avec état comme un service qui attribue les adresses permanentes aux machines, donc plus comme BOOTP que DHCP.

Cela conduit à deux problèmes de fond :

- la notion d'adresses temporaires «à la DHCP» a peu de sens en IPv6 ;
- même l'attribution centralisée d'adresses globales n'a pas beaucoup d'intérêt car la configuration sans état peut toujours être utilisée (dans les faits elle est utilisée depuis plus de six ans sans que le besoin d'un autre mécanisme se fasse réellement ressentir).

Par contre, un système optionnel de gestion des adresses, en particulier faisant l'interfaçage avec les fonctions de contrôle d'accès au réseau peut avoir un intérêt lorsque le protocole IPv6 sera utilisé dans la téléphonie de troisième génération.

Il est aussi nécessaire de disposer de moyens pour découvrir les paramètres d'un petit nombre de services de base comme le serveur de noms, le domaine DNS, ou les imprimantes disponibles. Plusieurs pistes alternatives à DHCPv6 sont envisagées comme l'utilisation d'adresses anycast.

4) Renumérotation des routeurs

Le mécanisme d'allocation d'adresses sans état permet aux équipements terminaux de s'auto-configurer automatiquement. Un nœud crée son (ou ses) adresse(s) en concaténant un identifiant d'interface, généralement construit à partir de l'adresse MAC, et des préfixes annoncés par les routeurs. Il n'existe pas de protocole automatique de configuration de routeur, les adresses des différents interfaces doivent être configurés statiquement, par exemple en donnant la liste des préfixes correspondant à un interface puis en créant des adresses en concaténant un identifiant à chacun des préfixes.

Le protocole de renumérotation automatique constitue une étape supplémentaire pour automatiser la gestion d'un réseau : il permet de manière globale et automatisée de modifier la configuration des routeurs et les annonces de préfixes. Un administrateur réseau peut ajouter, retirer ou modifier des préfixes dans les routeurs, ce qui aura pour effet de renuméroter toutes les machines d'un réseau (aussi bien un simple lien que tout un site)8.

Le protocole, défini dans le RFC 2894, concerne uniquement les routeurs. Il utilise des messages ICMPv6 de type 138 (cf. tableau Valeurs des champs type et code d'ICMPv6, See Valeurs des champs type et code d'ICMPv6) pour véhiculer les instructions de renumérotation.

Les messages sont diffusés en unicast ou en multicast (en utilisant l'adresse «tous les routeurs du lien» ou «tous les routeurs du site»). Ils contiennent un préfixe qui permet de sélectionner le ou les interfaces des routeurs sur lesquelles s'applique la rémunération. Trois commandes sont possibles sur ces interfaces :

- ajouter un nouveau préfixe,
- remplacer le préfixe «sélectionné» par un autre préfixe,
- remplacer tous les préfixes existant par un nouvel ensemble de préfixes.

Chaque fois qu'un préfixe est modifié sur une interface, l'adresse globale correspondante est elle aussi modifiée.

Vu les problèmes de sécurité lié à au protocole de rémunération automatique, l'utilisation de l'extension d'authentification est nécessaire, et un numéro de séquence toujours croissant permet d'interdire les rejeux.

Ce protocole est encore expérimental et n'est pas largement déployé dans les routeurs.

5) Mécanisme de découverte du PMTU

Pour des considérations d'efficacité (voir paragraphe Protocoles réseau et transport), il est généralement préférable que les informations échangées entre équipements soient contenues dans des datagrammes de taille maximale. Cette taille dépend du chemin suivi par les datagrammes et est égale à la plus grande taille autorisée par l'ensemble des liens traversés. Elle est de ce fait appelée PMTU, ou Path Maximum Transmission Unit (unité de transfert de taille maximale sur le chemin).

A) Principe

Initialement, l'équipement émetteur fait l'hypothèse que le PMTU d'un certain chemin est égal au MTU du lien auquel il est directement attaché (cf. le paquet 4 de l'exemple). S'il s'avère que les paquets transmis sur ce chemin excèdent la taille maximale autorisée par un lien intermédiaire, alors le routeur associé détruit ces paquets et retourne un message d'erreur ICMPv6 de type «paquet trop grand» (voir Protocoles réseau et transport), en y indiquant le MTU accepté (voir paquet 5 de l'exemple suivant). Fort de ces informations, l'équipement émetteur réduit le PMTU supposé pour ce chemin (paquet 6).

Plusieurs itérations peuvent être nécessaires avant d'obtenir un PMTU permettant à tout paquet d'arriver à l'équipement destinataire sans jamais excéder le MTU de chaque lien traversé. Le protocole IPv6 garantit que le MTU de tout lien ne peut descendre en dessous de 1 280 octets, valeur qui constitue ainsi une borne inférieure pour le PMTU. Ce protocole reposant sur la perte de paquets, il est laissé le soin aux couches supérieures de gérer la fiabilité de la communication en retransmettant si nécessaire (paquet 6 de l'exemple).

Si la détermination du PMTU se fait essentiellement lors des premiers échanges entre les équipements concernés, elle peut également être revue en cours de transfert si, suite à un changement de route, un lien plus contraignant est traversé.

L'émetteur vérifie aussi que le PMTU n'a pas augmenté en envoyant de temps en temps un paquet plus grand. Si celui-ci traverse le réseau sans problème, la valeur du PMTU est augmentée.

Signalons enfin que l'algorithme de découverte du PMTU fonctionne indifféremment avec des échanges point-à-point ou multipoints. Dans ce dernier cas, le PMTU sera le PMTU minimal permis par l'ensemble des chemins vers chaque site destinataire du groupe de diffusion.

B) Exemple

Cet exemple montre les premiers paquets échangés lors d'une ouverture de connexion TCP. Les machines sont situées sur deux réseaux Ethernet distincts (MTU de 1 500 octets) et interconnectés par un tunnel IPv4 (MTU de 1 480 octets du fait de la présence de l'en-tête IPv4 supplémentaire).

```
Paquet 1
IPv6
 Version : 6 Classe : 0x00 Label : 00000
 Longueur : 40 octets (0x0028) Protocole : 6 (0x06, TCP
 Nombre de sauts : 64 (0x40)
 Source : 3ffe:302:12:2::13 (oban)
 Desti. : 3ffe:304:115:8300:2c0:4fff:fe61:214c (duval)
TCP
 Port Source : 0xffad Port Destination : 0x1389
 Sequence : 0x5c3e066a Acquittement : 0x00000000
 Offset : 0xa Drapeaux : 0x2 (SYN) fenêtre :0x4000
 Checksum : 0x9c2e Ptr Msg urgent : 0x0000
 Options : - mss 1440, nop,wscale 0, timestamp 10386372 0

0000: 60 00 00 00 00 28 06 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 04 01 15 83 00
0020: 02 c0 4f ff fe 61 21 4c|ff ad 13 89 5c 3e 06 6a
0030: 00 00 00 00 a0 02 40 00 9c 2e 00 00 02 04 05 a0
0040: 01 03 03 00 01 01 08 0a 00 9e 7b c4 00 00 00 00
```

La phase de connexion commence avec l'émission d'un paquet SYN. Au niveau des options, la taille des segments proposée est de 1440 octets, soit une taille de paquets de 1500 octets si l'on ajoute l'en-tête IPv6 et l'en-tête TCP.

```
Paquet 2
IPv6
 Version : 6 Classe : 00 Label : 00000
 Longueur : 40 octets (0x0028) Protocole : 6 (0x06, TCP)
 Nombre de sauts : 60 (0x3c)
 Source : 3ffe:304:115:8300:2c0:4fff:fe61:214c(duval)
 Desti. : 3ffe:302:12:2::13 (oban)
TCP
 Port Source : 0x1389 Port Destination : 0xffad
 Sequence : 0xe3599c1a Acquittement : 0x5c3e066b
 Offset : 0xa Drapeaux : 0x12 (SYN ACK) fenêtre :0x42f0
 Checksum : 0x145a Ptr Msg urgent : 0x0000
 Options :
  - mss 4410, wscale 0, timestamp 4323715 10386372

0000: 60 00 00 00 00 28 06 3c 3f fe 03 04 01 15 83 00
0010: 02 c0 4f ff fe 61 21 4c 3f fe 03 02 00 12 00 02
0020: 00 00 00 00 00 00 00 13|13 89 ff ad e3 59 9c 1a
0030: 5c 3e 06 6b a0 12 42 f0 14 5a 00 00 02 04 11 3a
0040: 01 03 03 00 01 01 08 0a 00 41 f9 83 00 9e 7b c4
```

L'entité distante accepte la connexion ainsi que la taille de segment de 1440 octets.

```
Paquet 3
IPv6
 Version : 6 Classe : 00 Label : 00000
 Longueur : 32 octets (0x0020) Protocole : 6 (0x06, TCP)
 Nombre de sauts : 64 (0x40)
 Source : 3ffe:302:12:2::13 (oban)
 Desti. : 3ffe:304:115:8300:2c0:4fff:fe61:214c (duval)
```

```
TCP
 Port Source : 0xffad Port Destination : 0x1389
 Sequence : 0x5c3e066b Acquittement : 0xe3599c1b
 Offset : 0x8 Drapeaux : 0x10 (ACK) fenêtre :0x4380
 Checksum : 0x4b14 Ptr Msg urgent : 0x0000

0000: 60 00 00 00 00 20 06 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 04 01 15 83 00
0020: 02 c0 4f ff fe 61 21 4c|ff ad 13 89 5c 3e 06 6b
0030: e3 59 9c 1b 80 10 43 80 4b 14 00 00 01 01 08 0a
0040: 00 9e 7b c4 00 41 f9 83
```

Fin d'ouverture de connexion.

```
Paquet 4
IPv6
 Version : 6 Classe : 0 Label : 000000
 Longueur : 1460 octets (0x05b4) Protocole : 6 (0x06, TCP)
 Nombre de sauts : 64 (0x40)
 Source : 3ffe:302:12:2::13 (oban)
 Desti. : 3ffe:304:115:8300:2c0:4fff:fe61:214c (duval)
TCP
 Port Source : 0xffad Port Destination : 0x1389
 Sequence : 0x5c3e066b Acquittement : 0xe3599c1b
 Offset : 0x8 Drapeaux : 0x10 (ACK) fenêtre :0x4380
 Checksum : 0x40a9 Ptr Msg urgent : 0x0000
 Données : 1440 octets.

0000: 60 00 00 00 05 b4 06 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 04 01 15 83 00
0020: 02 c0 4f ff fe 61 21 4c|ff ad 13 89 5c 3e 06 6b
0030: e3 59 9c 1b 80 10 43 80 40 a9 00 00 01 01 08 0a
0040: 00 9e 7b c4 00 41 f9 83|20 21 22 23 24 25 26 27
...suite des 1440 octets de données...
```

Premier envoi de données, en supposant que le PMTU correspond au MSS négocié.

```
Paquet 5
IPv6
 Version : 6 Classe : 00 Label : 00000
 Longueur : 536 octets (0x0218) Protocole : 58 (0x3a, ICMPv6)
 Nombre de sauts : 254 (0x0fe)
 Source : 3ffe:302:11:1::8 (site-router)
 Desti. : 3ffe:302:12:2::13 (oban)
ICMPv6
 Type : 2 (0x02, Paquet trop grand) Code : 0 Checksum : 0e8f7
 MTU : 1480 (0x0000 05c8)
 Données : Paquet ayant provoque l'erreur

0000: 60 00 00 00 02 18 3a fe 3f fe 03 02 00 11 00 01
0010: 00 00 00 00 00 00 00 08 3f fe 03 02 00 12 00 02
0020: 00 00 00 00 00 00 00 13|02 00 e8 f7 00 00 05 c8|
0030: 60 00 00 00 05 b4 06 3e 3f fe 03 02 00 12 00 02
0040: 00 00 00 00 00 00 00 13 3f fe 03 04 01 15 83 00
0050: 02 c0 4f ff fe 61 21 4c|ff ad 13 89 5c 3e 06 6b
0060: e3 59 9c 1b 80 10 43 80 40 a9 00 00 01 01 08 0a
0070: 00 9e 7b c4 00 41 f9 83|20 21 22 23 24 25 26 27
suite du paquet IPv6 tronqué à 1280-48 octets ...
```

Le routeur gérant le tunnel annonce que ce paquet ne peut être relayé tel quel. Le paquet ICMP retourné inclut le MTU accepté par le tunnel et les premiers octets du paquet concerné. Noter que la taille de ce

message ICMPv6 est limitée à 1280 octets. En pratique seule la partie utile du paquet TCP (son en-tête) a été recopiée.

```
Paquet 6
IPv6
 Version : 6 Classe : 00 Label : 00000
 Longueur : 1440 octets (0x05a0) Protocole : (0x06, TCP)
 Nombre de sauts : 64 (0x40)
 Source : 3ffe:302:12:2::13 (oban)
 Desti. : 3ffe:304:115:8300:2c0:4fff:fe61:214c (duval)
TCP
 Port Source : 0xffad Port Destination : 0x1389
 Sequence : 0x5c3e066b Acquittement : 0xe3599c1b
 Offset : 0x8 Drapeaux : 0x10 (ACK) fenêtre :0x4380
 Checksum : 0x8bb3 Ptr Msg urgent : 0x0000
 Données : 1420 octets.

0000: 60 00 00 00 05 a0 06 40 3f fe 03 02 00 12 00 02
0010: 00 00 00 00 00 00 00 13 3f fe 03 04 01 15 83 00
0020: 02 c0 4f ff fe 61 21 4c|ff ad 13 89 5c 3e 06 6b
0030: e3 59 9c 1b 80 10 43 80 8b b3 00 00 01 01 08 0a
suite des 1420 octets de données...
```

L'émetteur retransmet les données perdues en se limitant cette fois aux 1 420 octets permis (soit 1 480 moins les en-têtes IPv6 et TCP).

C) Mise en œuvre

L'exploitation de l'information de PMTU se fait de plusieurs façons suivant l'endroit où les données à transmettre sont segmentées :

- si un protocole de type TCP est utilisé, celui-ci assurera la segmentation de façon transparente pour les applications, en fonction des informations de PMTU que pourra lui communiquer la couche IPv6.
- si un protocole de type UDP est utilisé, alors cette segmentation devra être assurée par une couche supérieure, éventuellement l'application. Il faut donc que celle-ci
    - (1) puisse être informée du PMTU autorisé, même dans le cas où celui-ci change par la suite, et
    - (2) puisse segmenter ses données en conséquence. Parce que ces deux conditions ne sont pas toujours réunies, IPv6 a conservé un mécanisme de fragmentation (voir fragmentation).

Un deuxième aspect concerne l'identification des chemins afin de pouvoir y associer les informations de PMTU. Plusieurs possibilités, laissées à l'implémenteur, sont possibles. Un chemin peut être identifié par l'adresse destination, ou par l'identificateur de flux si celui-ci est utilisé, ou par la route suivie dans le cas où elle est imposée (voir routage).

Enfin, s'il est fortement recommandé que chaque équipement supporte le mécanisme de recherche du PMTU, ce n'est pas obligatoire. Ainsi, un équipement qui n'en dispose pas (par exemple une ROM de boot) devra restreindre la taille de tout paquet transmis au MTU minimal que doit supporter tout lien, soit 1280 octets.

VI) Nommage

Lorsqu'une application souhaite communiquer avec une autre s'exécutant sur un équipement distant dont elle ne connaît que le nom, elle a besoin d'en trouver l'adresse, sans quoi la communication ne peut en général avoir lieu. Aux débuts de l'Internet, les adresses IP en usage n'étaient pas très nombreuses et il était donc relativement facile de les stocker dans une base de données centralisée, le fichier hosts.txt. Cette base de données pouvait alors être téléchargée (via ftp) par les utilisateurs souhaitant rafraîchir leurs informations stockées sous forme de fichier local (en l'occurrence `/etc/hosts` pour les systèmes Unix).

Dès le début des années 80, la croissance du nombre d'adresses IP utilisées et le besoin de plus en plus fréquent de renuméroter les équipements ont rendu de plus en plus difficiles la mise à jour et la mémorisation de ces adresses. Afin de remédier à ce problème, un nouveau système, le DNS (*Domain Name System*), a été conçu et mis en œuvre. Petit à petit, le DNS s'est imposé comme étant une infrastructure pour l'ensemble des applications TCP/IP classiques comme le mail, le web, le transfert de fichier et la connexion à distance. Ce système a l'avantage d'être :

- hiérarchique : sa structure d'arbre est analogue à celle d'un système de fichiers Unix. Cette structure d'arbre rend le système extensible (« scalable »),
- réparti : au niveau de chaque nœud, un ensemble de serveurs fait autorité sur les données contenues dans la zone décrivant ce nœud. Cet ensemble de serveurs représente la source officielle des données de la zone,
- et redondant : deux serveurs ou plus sont nécessaires pour chaque zone DNS afin d'assurer une meilleure disponibilité et un équilibrage de charge.

Le DNS avait initialement comme objectif premier d'offrir un service de résolution de noms de domaines Internet complètement qualifié (FQDN : *Fully Qualified Domain Name*) garantissant l'unicité du nom (exemple : `ns3.nic.fr`) en adresses IP et vice-versa.

En pratique, le service de résolution DNS consiste plus généralement à stocker et à retourner, sous forme d'enregistrements DNS (RR : *Resource Records*) et à la demande des applications, des informations associées à des noms de domaines, comme les adresses IP, les relais de messagerie (enregistrement de type `MX`) ou les serveurs de noms (enregistrement de type `NS`).

Rappelons au passage que pour ces applications, la communication est précédée par une phase lors de laquelle le client DNS local, appelé « stub resolver », interroge son serveur DNS récursif (ou cache) qui se charge d'effectuer les requêtes itératives nécessaires, en partant de la racine de l'arbre DNS s'il le faut, et de retourner les ressources recherchées. Pour les machines Unix, le fichier `/etc/resolv.conf` fournit l'adresse IP du (ou des) serveur(s) à interroger par défaut.

Le service de résolution DNS se trouvant au niveau de la couche application de la pile TCP/IP, il s'applique aux réseaux IPv6 de manière analogue aux réseaux IPv4. Le fait que les adresses IPv6 soient quatre fois plus longues que les adresses IPv4, qu'elles puissent être attribuées automatiquement et qu'elles soient représentées de surcroît en notation hexadécimale, a considérablement réduit les chances pour ces

adresses IPv6 d'être mémorisées par un être humain. Ainsi, avec l'arrivée d'IPv6, le DNS devient plus que jamais un service critique pour le fonctionnement des applications TCP/IP classiques.

Afin de supporter le nouveau schéma d'adressage d'IPv6, deux extensions DNS ont été définies (RFC 3596) :

- l'enregistrement `AAAA` (prononcé « quad A »), pour le nommage direct (correspondance : nom vers adresse). Ce nouveau type a pour code-valeur 28.
- le nouveau sous-arbre DNS inverse `ip6.arpa` pour le nommage inverse (correspondance : adresse vers nom)

1) Nommage direct : du nom vers les adresses

A) L'enregistrement AAAA

La correspondance entre un nom de domaine et son (ou ses) adresse(s) IPv4 est réalisée en associant au nom en question un ou plusieurs enregistrements DNS de type A. Chaque enregistrement contient une valeur qui est une adresse IPv4.

De manière analogue à l'enregistrement A, le nouveau type d'enregistrement `AAAA` défini pour IPv6, permet d'établir la correspondance entre un nom de domaine et son (ou une de ses) adresse(s) IPv6. Une machine ayant plusieurs adresses IPv6 globales a en principe autant d'enregistrements `AAAA` publiés dans le DNS. Une requête DNS de type `AAAA` concernant une machine particulière retourne dans ce cas tous les enregistrements `AAAA` publiés dans le DNS et correspondant à cette machine. Toutes les adresses n'ont cependant pas leur place dans le DNS. Ce sujet sera traité au paragraphe Publication des enregistrements AAAA dans le DNS.

B) Format

Le format textuel d'un enregistrement `AAAA` tel qu'il apparaît dans le fichier de zone DNS est le suivant :

```
<nom> [ttl] IN AAAA <adresse>
```

L'adresse est écrite suivant la représentation classique des adresses IPv6 (RFC 4291). Par exemple, l'adresse IPv6 de la machine `ns3.nic.fr` est publiée dans le fichier de zone `nic.fr` comme suit :

```
ns3.nic.fr. IN AAAA 2001:660:3006:1::1:1
```

Il est important de noter que toutes les adresses IPv4 et/ou IPv6 correspondant à un équipement donné, doivent cohabiter dans le même fichier de zone renseignant le nom de l'équipement en question. Ainsi, les adresses de `ns3.nic.fr` sont publiées dans le fichier de zone `nic.fr` comme suit :

```
$ORIGIN nic.fr.
```

```
ns3 IN A    192.134.0.49
    IN AAAA 2001:660:3006:1::1:1
```

2) Nommage inverse : de l'adresse vers les noms

L'enregistrement de type `PTR`, stocké sous l'arbre DNS inverse `in-addr.arpa`, permet d'établir la correspondance entre une adresse IPv4 et un (ou plusieurs) nom(s). C'est ce même type d'enregistrement `PTR`, qui, stocké sous l'arbre DNS inverse `ip6.arpa`, permet de mettre en correspondance une adresse IPv6 avec un ou plusieurs noms de domaines.

Notons au passage qu'auparavant, le RFC 1886, rendu obsolète par le RFC 3596, spécifiait une autre arborescence : `ip6.int`. Cette dernière a été arrêtée en 2006.

Une adresse IPv6 est transformée en un nom de domaine publié sous l'arborescence inverse `ip6.arpa` de la manière suivante : les 32 demi-octets formant l'adresse IPv6 sont séparés par le caractère `.' et concaténés dans l'ordre inverse au suffixe `ip6.arpa`.

Par exemple l'adresse `2001:660:3006:1::1:1` (adresse de `ns3.nic.fr`) est transformée en le nom de domaine inverse suivant :

1.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa.

On publie alors dans le DNS inverse l'enregistrement `PTR` correspondant au nom de domaine inverse ci-dessus. Dans cet exemple, l'enregistrement PTR vaut `` `ns3.nic.fr.' ``

En pratique, on procède par délégation de zones inverses afin de répartir les enregistrements `PTR` sur un système hiérarchique de serveurs DNS. Soulignons que la délégation DNS inverse suit le schéma classique d'attribution des adresses IP (le même pour IPv4 et IPv6) :

- 1) L'IANA délègue (en terme de provision) de larges blocs d'adresses IPv6 aux registres Internet régionaux (RIR : Regional Internet Registry), typiquement des préfixes de longueur 12 selon la politique actuelle
- 2) Les RIR allouent (en terme de provision) des blocs d'adresses IPv6 plus petits aux registres Internet locaux (LIR : Local Internet Registry), c'est-à-dire aux fournisseurs d'accès Internet de la région, typiquement des préfixes de longueur 32 (ou plus courts selon le besoin). À noter que dans les régions APNIC et LACNIC, il existe des Registres nationaux (NIR) comme registres intermédiaires entre le RIR et les LIR présents dans le pays en question.
- 3) Les LIR attribuent (pour un usage direct) des préfixes IPv6 aux clients finaux, typiquement des préfixes de longueur variable entre un /64 et un /48 (selon le besoin et selon la politique en vigueur).

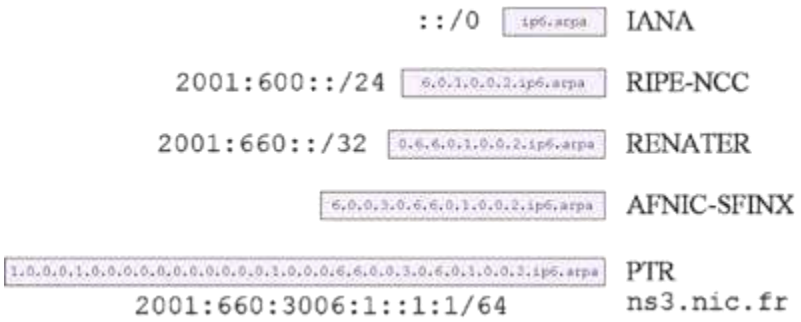


Figure 6-1 *Exemple de délégation de zones inverse*

À chaque nœud présent dans l'arborescence DNS inverse, illustrée par la figure Exemple de délégation de zones inverse, est associée une liste de serveurs DNS qui héberge la zone inverse décrivant ce nœud. Une telle liste comprend généralement un serveur primaire et un ou plusieurs serveurs secondaires, tous considérés comme faisant autorité sur la zone DNS inverse en question. C'est au client final, de publier dans ses propres zones DNS inverse les enregistrements `PTR` correspondant aux adresses IPv6 qu'il utilise.

Par exemple, Renater a reçu le préfixe `2001:660::/32` et la délégation de la zone DNS inverse `0.6.6.0.1.0.0.2.ip6.arpa` de la part du RIPE-NCC. Renater a affecté à l'AFNIC le préfixe `2001:660:3006::/48` et lui a délégué la zone DNS inverse correspondante :

```
6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa. IN NS ns1.nic.fr.
6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa. IN NS ns2.nic.fr.
6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa. IN NS ns3.nic.fr.
```

L'AFNIC publie alors dans sa zone DNS inverse les enregistrements `PTR` correspondant aux adresses IPv6 utilisées. Voici un extrait du fichier de zone DNS :

```
$ORIGIN 6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa.
1.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0 IN PTR ns3.nic.fr.
```

3) Logiciels DNS supportant IPv6 et configurations

Il existe aujourd'hui de nombreux logiciels DNS mais la présente section ne les liste pas de manière exhaustive. La plupart de ces logiciels DNS supportent IPv6 dans leurs versions récentes. Ce support peut être soit complet (enregistrements `AAAA`, enregistrements `PTR` sous l'arborescence `ip6.arpa` et transport IPv6 des messages DNS) soit partiel (uniquement les enregistrements `AAAA` et `PTR`) selon le logiciel.

En outre, certaines distributions logicielles comportent l'implémentation du client et du serveur, d'autres n'incluent que l'implémentation du client ou celle du serveur. Par exemple, la distribution BIND 9 développée par l'ISC (*Internet Systems Consortium*) représente la référence de fait dans le domaine. En effet, il s'agit d'un logiciel complet (client, serveur et outils) intégrant toutes les extensions DNS récentes (IPv6, DNSSEC, ...). Les distributions BIND 9 ont l'avantage d'être disponibles en code source et en format binaire pour la quasi-totalité des plates-formes (Unix, MS Windows *, ...). Ainsi, la distribution BIND 9 a été choisie comme base pour les exemples de fichiers de configuration.

Contents

A) Clients et outils de vérification de configurations DNS

Un client DNS se présente souvent sous forme d'une bibliothèque de résolution appelée « `libresolv` », le client est alors appelé « resolver » ou encore « stub resolver ». Rappelons que ce resolver est sollicité par les applications TCP/IP s'exécutant sur un équipement donné pour les renseigner sur les ressources DNS nécessaires à l'établissement de leur communication avec des applications distantes. Outre le resolver, il existe des outils et commandes selon le système d'exploitation, qui permettent d'interroger un serveur DNS dans un but de débogage et/ou de diagnostic. C'est le cas par exemple des outils `dig`, `host` et `nslookup` qui font partie des distributions BIND et pour lesquels des exemples sont donnés ci-après.

Notons que lorsque le serveur à interroger n'est pas explicitement renseigné, c'est le (ou les) serveurs par défaut qui est (sont) interrogé(s). Il s'agit de la liste des serveurs récursifs qui est configurée automatiquement (via DHCP par exemple) ou manuellement (dans le fichier `/etc/resolv.conf` pour les systèmes Unix par exemple ou au travers d'une interface graphique pour MS Windows et Mac OS) sur l'équipement. Les mécanismes de découverte de la liste des serveurs DNS récursifs seront décrits plus loin dans la section découverte de la liste de serveurs DNS récursifs, See Découverte de la liste de serveurs DNS récursifs. L'exemple suivant décrit un fichier `resolv.conf` sous Unix :

```
search nic.fr                   # domaine de recherche par défaut
nameserver     ::1              # prefer localhost-v6
nameserver     192.134.4.162    # backup v4
```

a) Exemples d'interrogation

Les six exemples suivants illustrent l'utilisation des outils `dig`, `host` et `nslookup` pour la même requête de résolution du nom `ns3.nic.fr' en adresse(s) IPv6 :

```
>dig ns3.nic.fr aaaa

; <<>> DiG 9.3.3 <<>> ns3.nic.fr aaaa
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3032
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 7, ADDITIONAL: 7
;; QUESTION SECTION:
;ns3.nic.fr.                    IN      AAAA
;; ANSWER SECTION:
ns3.nic.fr.             172800  IN      AAAA    2001:660:3006:1::1:1
```

```
;; AUTHORITY SECTION:
nic.fr.                 78032   IN      NS      ns1.nic.fr.
nic.fr.                 78032   IN      NS      ns2.nic.fr.
nic.fr.                 78032   IN      NS      ns3.nic.fr.
nic.fr.                 78032   IN      NS      ns-sec.ripe.net.
[...]
;; ADDITIONAL SECTION:
ns1.nic.fr.             78032   IN      A       192.93.0.1
ns1.nic.fr.             17168   IN      AAAA    2001:660:3005:1::1:1
ns2.nic.fr.             25737   IN      A       192.93.0.4
ns2.nic.fr.             25737   IN      AAAA    2001:660:3005:1::1:2
ns-sec.ripe.net.        96368   IN      A       193.0.0.196
ns-sec.ripe.net.        96368   IN      AAAA    2001:610:240:0:53::4
;; Query time: 2 msec
;; SERVER: ::1#53(::1)
;; WHEN: Thu Oct 25 19:13:54 2007
;; MSG SIZE  rcvd: 350
```

```
>dig ns3.nic.fr aaaa @ns-sec.ripe.net

; <<>> DiG 9.3.3 <<>> ns3.nic.fr aaaa @ns-sec.ripe.net
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16927
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 7, ADDITIONAL: 5

;; QUESTION SECTION:
;ns3.nic.fr. IN AAAA

;; ANSWER SECTION:
ns3.nic.fr. 345600 IN AAAA 2001:660:3006:1::1:1

;; AUTHORITY SECTION:

[...]

;; SERVER: 2001:610:240:0:53::4#53(ns-sec.ripe.net)
```

```
>host -t aaaa ns3.nic.fr
ns3.nic.fr has AAAA address 2001:660:3006:1::1:1
```

```
>host -t aaaa ns3.nic.fr ns-sec.ripe.net
Using domain server:
Name: ns-sec.ripe.net
Address: 2001:610:240:0:53::4#53
Aliases:

ns3.nic.fr has AAAA address 2001:660:3006:1::1:1
```

```
>nslookup -type=aaaa ns3.nic.fr
Server: 2001:660:3003:2::1:1
Address: 2001:660:3003:2::1:1#53

Non-authoritative answer:
```

```
ns3.nic.fr has AAAA address 2001:660:3006:1::1:1

[...]
```

```
>nslookup -type=aaaa ns3.nic.fr -secripe.net
Server: ns-sec.ripe.net
Address: 2001:610:240:0:53::4#53

ns3.nic.fr has AAAA address 2001:660:3006:1::1:1
```

Dans les exemples 1, 3 et 5, la requête est envoyée au serveur récursif par défaut (`2001:660:3003:2::1:1`). Dans les exemples 2, 4 et 6, la requête est envoyée au serveur `ns-sec.ripe.net` (qui est secondaire pour la zone `nic.fr`).

Notons que l'outil `nslookup` n'est plus maintenu par l'ISC et qu'il est amené à disparaître. L'usage de l'outil `dig` ou de `host` pour toutes sortes de requêtes est en revanche recommandé.

La deuxième version de `ZoneCheck`, l'outil utilisé par l'AFNIC pour vérifier la configuration et valider la délégation de zones DNS sous .fr et .re, supporte IPv6 complètement. En effet, pour une zone DNS quelconque, ZoneCheck permet d'interroger la liste des serveurs faisant autorité sur cette zone afin de vérifier leur bon fonctionnement (en termes de transport UDP et TCP au-dessus d'IPv4 et d'IPv6 si IPv6 est supporté) et la bonne configuration de la zone DNS en question (en termes de base de données, notamment concernant la cohérence des enregistrements DNS entre serveurs différents).

b) Fichier de configuration d'un serveur BIND9

Pour un serveur de nom BIND 9, le fichier de configuration named.conf contient une succession de parties déclaratives. La partie `options` par exemple, indique au serveur les différentes options de configuration telles que l'activation de l'écoute (socket) en IPv4 et/ou en IPv6, l'activation ou non du mode récursif ou le chemin d'accès aux données (option directory).

Les zones DNS sur lesquelles le serveur fait autorité (primaire ou secondaire) sont ensuite déclarées successivement grâce à des rubriques de type zone. Pour chaque zone, le nom du fichier contenant les enregistrements de cette zone est précisé. Lorsque le serveur est secondaire pour une zone donnée, on indique (à l'aide de la sous-rubrique `masters`) la liste des adresses IPv4 et/ou IPv6 des serveurs à partir desquels ce secondaire peut s'alimenter. Voici maintenant un extrait du fichier `named.conf` du serveur DNS `ns3.nic.fr` :

```
options {
   directory "/usr/local/bind";
   recursion no;
   listen-on { any;};
   listen-on-v6 {any; };
   [...]
};

[...]

zone "." {
```

```
   type hint;
   file "named.root";
};

zone "localhost" {
   type master;
   file "localhost";
};

// Zone contenant l'enregistrement inverse pour l'adresse loopback en IPv4

zone "0.0.127.in-addr.arpa" {
   type master;
   file "localhost.rev";
};

// Zone inverse (sous ipv6.arpa) contenant l'enregistrement inverse pour
// l'adresse loopback  en IPv6


zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" {
   type master;
   file "localhost.rev";
};

[...]

zone "nic.fr" {
   type slave;
   file "zone/nic.fr";
   masters {
    2001:660:3005:1::1:1; 192.93.0.1;
    2001:660:3005:1::1:2; 192.93.0.4;
   };
};

[...]

// Zone inverse IPv4 pour la réseau AFNIC-SFINX en 192.134.0/24

zone "0.134.192.in-addr.arpa" {
   type slave;
   file "rev/nic.fr.192.134.0";
   masters {
    2001:660:3005:1::1:1; 192.93.0.1;
    2001:660:3005:1::1:2; 192.93.0.4;
   };
};

[...]

// Blocs Ripe sous ip6.arpa.

zone "6.0.1.0.0.2.ip6.arpa" {
   type slave;
   file "rev/6.0.1.0.0.2.ip6.arpa";
   masters {
    2001:610:240:0:53::3;
    193.0.0.195;
   };
};

[...]

// Zone inverse IPv6 pour le reseau AFNIC-SFINX en 2001:660:3006::/48
```

```
zone "6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa" {
   type slave;
   file "rev/6.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa";
   masters {
    2001:660:3005:1::1:1; 192.93.0.1;
    2001:660:3005:1::1:2; 192.93.0.4;
   };
};

[...]
```

L'option « `listen-on` » peut avoir comme valeurs possibles :

- `any` : dans ce cas-là, le serveur écoutera sur toutes ces adresses IPv4 opérationnelles ;
- une liste explicite comprenant une ou plusieurs adresses IPv4 données : le serveur écoutera uniquement sur ses adresses pour ce qui est du transport IPv4 des requêtes et réponses ;
- `none` : pas de support d'IPv4 (cette valeur n'est pas utilisée aujourd'hui).

L'option « `listen-on-v6` » peut avoir comme valeurs possibles :

- `any` : dans ce cas-là, le serveur écoutera sur toutes ses adresses IPv6 opérationnelles ;
- une liste explicite comprenant une ou plusieurs adresses IPv6 données : le serveur écoutera uniquement sur ces adresses pour ce qui est du transport IPv6 des requêtes et réponses ;
- `none` : pas de support d'IPv6 (valeur par défaut).

Les cinq premières zones déclarées font partie de la configuration d'un serveur BIND de base. Les quatre zones restantes sont données à titre d'exemple parmi les nombreuses zones sur lesquelles le serveur `ns3.nic.fr` fait autorité.

c) Fichier de zone DNS direct

Voici à titre d'exemple, un extrait du fichier de zone DNS direct `nic.fr', faisant apparaître en même temps des adresses IPv4 et IPv6. On remarquera dans cet exemple que les adresses IPv6 ont été construites manuellement pour garantir leur pérennité dans le DNS. En effet, rappelons dans ce contexte que les adresses obtenues par auto-configuration dépendent généralement de l'adresse physique de la carte réseau utilisée (RFC 4291).

```
$TTL 172800
$ORIGIN nic.fr.
@ IN SOA maya.nic.fr. hostmaster.nic.fr. (
    2007102200 ;serial
    21600 ;refresh (6 h)
    3600 ;retry (1 h)
    3600000 ;expire
    86400 ) ;minimum (2 j)

        IN NS ns1.nic.fr.
        IN NS ns2.nic.fr.
        IN NS ns3.nic.fr.
[...]
            IN  MX 10   mx1.nic.fr.
```

```
            IN  MX 20   mx2.nic.fr.
[...]
 ns1        IN  A       192.93.0.1
            IN  AAAA    2001:660:3005:1::1:1
 ns2        IN  A       192.93.0.4
            IN  AAAA    2001:660:3005:1::1:2
 ns3        IN  A       192.134.0.49
            IN  AAAA    2001:660:3006:1::1:1

[...]

 www        IN  CNAME   rigolo
 rigolo     IN  A       192.134.4.20
            IN  AAAA    2001:660:3003:2::4:20
 kerkenna   IN  A 192.134.4.98
            IN  AAAA 2001:660:3003:8::4:98

[...]
```

d) Fichier de zone DNS inverse en IPv6

Voici un extrait de fichier de zone DNS inverse correspondant au préfixe IPv6 `2001:660:3003::/48` (réseau AFNIC-Saint-Quentin-en-Yvelines) et représentant quelques enregistrements de type PTR d'équipements supportant IPv6 :

```
$ORIGIN 3.0.0.3.0.6.6.0.1.0.0.2.ip6.arpa.
$TTL 172800
@       IN      SOA     maya.nic.fr.    hostmaster.nic.fr. (
                        2007031200      ;serial
                        43200   ;refresh (12 h)
                        14400   ;retry (4 h)
                        3600000 ;expire
                        3600  );
       IN      NS      ns1.nic.fr.
       IN      NS      ns2.nic.fr.
       IN      NS      ns3.nic.fr.
[...]
0.2.0.0.4.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.0         IN      PTR     rigolo.nic.fr.
7.2.0.0.4.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.0         IN      PTR     funk.nic.fr.
1.3.0.0.4.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.0         IN      PTR     wood.nic.fr.
8.9.0.0.4.0.0.0.0.0.0.0.0.0.0.0.0.0.8.0.0.0         IN      PTR     kerkenna.nic.fr.
[...]
```

4) Les solutions expérimentales A6 et bitstring labels

Vers la fin des années 90, on pensait qu'on allait bientôt avoir besoin de renuméroter les réseaux IPv6 de plus en plus souvent, notamment suite aux changements potentiellement fréquents des préfixes de sites IPv6 . Une telle renumérotation nécessiterait une mise à jour aussi fréquente du DNS pour assurer l'accessibilité des nouvelles adresses. On s'était vite rendu compte que l'enregistrement `AAAA` n'était pas adapté à ce besoin, notamment à cause du faible déploiement du mécanisme de mise à jour dynamique du DNS . Il a fallu alors spécifier une nouvelle extension et en même temps un nouveau type d'enregistrement, l'enregistrement `A6` (RFC 2874), dont la structure reflète deux parties distinctes d'une adresse IPv6 :

- une partie variable : le préfixe du lien auquel l'interface est attachée. Ce préfixe (les 64 bits de poids fort) est dérivé du préfixe de site et supporte plusieurs niveaux d'agrégation, chaque niveau d'agrégation étant renseigné dans un enregistrement `A6` faisant partie d'une chaîne ;
- une partie fixe (les 64 bits de poids faible) obtenue à partir de l'identificateur d'interface de l'équipement en question.

Malgré l'intérêt que présentait cette proposition, les groupes de travail de l'IETF dnsext et ngtrans ont décidé, après de longs débats, d'écarter l'extension `A6` de la voie de standardisation (*Standard Track*) en faisant passer le RFC 2874 à l'état « experimental » (RFC 3363) pour les raisons suivantes :

- on ne voyait toujours pas de besoin concret de renumérotation fréquente de réseaux IPv6 ;
- l'implémentation de l'extension `A6` et surtout son déploiement se sont avérés complexes. En effet, le fait qu'une requête de résolution DNS du type `A6` puisse faire appel récursivement à d'autres requêtes A6 afin de reconstituer l'adresse IPv6 complète recherchée, peut provoquer des temps d'attente trop longs faisant ainsi échouer la requête de résolution initiale ;
- il serait dangereux de mettre en œuvre la nouvelle extension `A6` sans s'assurer par avance que cette extension n'aura aucun impact négatif sur les performances du DNS en production.

Soulignons que le groupe de travail dnsext a d'un côté recommandé de continuer à expérimenter l'extension A6 et de l'autre, il a décidé de faire avancer l'extension `AAAA` initialement publié dans le RFC 1886 et par la même occasion l'extension `ip6.arpa`, initialement publiée dans le RFC 3152, sur la voie de standardisation. En effet, suite à des tests d'interopérabilité réussis, les RFC originels (1886 et 3152) qui spécifiaient ces extensions et qui étaient en *Proposed Standard* (PS), ont été recyclés en un document IETF (*Internet Draft*) qui a donné naissance par la suite au RFC 3596 (avec le statut de *Draft Standard*, DS).

Par ailleurs, une autre extension a été proposée pour améliorer la gestion des zones DNS inverse IPv6 : les bitstring labels (RFC 2673). Cette extension, qui était censée s'appliquer uniquement sur l'arbre inverse `ip6.arpa`, consistait à utiliser des étiquettes binaires pour nommer les enregistrements `PTR` plutôt que les étiquettes en notation hexadécimale. Le but était de permettre de déléguer les blocs d'adresses selon une longueur quelconque de préfixe et de lever ainsi la contrainte de délégation des zones DNS inverse sur la frontière du demi-octet. Cette extension a été écartée de la voie de standardisation en même temps que l'extension `A6` (RFC 3363) à cause de la complexité de sa mise en œuvre et du manque de retour d'expérience sur son utilisation.

5) Recommandations opérationnelles pour l'intégration d'IPv6

Comme cela a été décrit dans l'introduction de ce chapitre, le DNS a la particularité d'être à la fois une application TCP/IP et une infrastructure critique (en tant que base de données) pour le fonctionnement des autres applications TCP/IP classiques (web, mail, ftp, ...). Avec l'intégration progressive d'IPv6, de nouveaux problèmes opérationnels liés au DNS se posent ou risquent de se poser et il convient donc de les éviter ou de trouver tout au moins les solutions adéquates afin d'y remédier. À cet effet, RFC 3901 et RFC 4472 identifient les principaux problèmes et formulent une série de recommandations pratiques pour y faire face. Les sections qui suivent tentent de résumer ces recommandations.

Contents

A) Les deux aspects du DNS

En tant que base de données, le DNS supporte les enregistrements `A` et `AAAA` et ce, indépendamment de la version d'IP qui est utilisée pour transporter les requêtes et réponses DNS relatives à ces enregistrements. Par ailleurs, en tant qu'application TCP/IP, un serveur DNS supporte le transport IPv4 ou le transport IPv6 ou les deux à la fois. Dans tous les cas, il doit retourner ce qu'il a dans sa base de données pour répondre à une requête donnée, indépendamment de la version d'IP sur laquelle il a reçu cette requête. En effet, un serveur DNS ne peut a priori pas savoir si le client initiateur de la requête a transmis celle-ci en IPv4 ou en IPv6 à son serveur récursif (cache) : des serveurs intermédiaires appelés cache forwarders et n'utilisant pas la même version d'IP que le client, peuvent intervenir dans la chaîne des serveurs interrogés durant le processus de résolution DNS. En outre, en admettant même que le serveur DNS puisse connaître la version d'IP utilisée par le client qui a initié la requête, il faut souligner que le serveur n'a pas à faire d'hypothèse sur l'usage que fera le client de la réponse DNS retournée.

B) Continuité du service DNS

Avant l'arrivée d'IPv6, le processus de résolution DNS ne faisait intervenir que la version 4 d'IP et le service était donc garanti pour tous les clients DNS. Avec IPv6, on risque de se trouver dans des configurations où l'espace de nommage devient fragmenté en des partitions accessibles uniquement au-dessus d'IPv4 et d'autres accessibles uniquement au-dessus d'IPv6. Voici par exemple deux scénarios illustrant ce problème de fragmentation ainsi que la solution recommandée dans chaque scénario :

- premier scénario : un client ne supportant qu'IPv4 souhaite résoudre une requête relative à une zone DNS hébergée sur des serveurs ne supportant qu'IPv6. Dans ces cas-là, le processus de résolution se termine par un échec dû à l'impossibilité d'accéder aux serveurs qui font autorité sur cette zone. Pour y remédier, il faudra faire en sorte que toute zone DNS soit servie par au moins un serveur supportant IPv4.
- second scénario : un client DNS dans un réseau ne supportant qu'IPv6 souhaite résoudre une requête donnée. Si le serveur récursif interrogé ne supporte pas non plus IPv4, le processus de résolution risque d'échouer plus tard avant de joindre les serveurs faisant autorité sur l'information recherchée. En effet, lors de son parcours de la chaîne de délégations dans l'arborescence DNS, le serveur récursif risque de tomber sur un serveur ne supportant pas IPv6. Afin d'y remédier, il est recommandé dans ce cas, de configurer le serveur récursif en le faisant pointer vers un serveur `forwarder` en double pile IPv4/IPv6.

Par exemple, pour une distribution BIND, il suffit d'ajouter l'option :

```
forwarders {<liste des adresses de serveurs forwarders> ;}
```

sous la rubrique « `options` » dans le fichier `named.conf`.

C) Taille limitée des messages DNS en UDP

Les implémentations DNS s'appuient essentiellement sur deux standards de l'IETF (RFC 1034 et RFC 1035). De nombreux autres RFC complémentaires ont été publiés plus tard pour clarifier certains aspects pratiques ou pour apporter de nouvelles extensions répondant à de nouveaux besoins (enregistrements `AAAA`, `SRV`, extensions DNSSEC, ...). En tant qu'application TCP/IP, le DNS doit supporter les deux modes de transport UDP et TCP (RFC 1035), le port associé étant le même : 53.

Le mode UDP est généralement utilisé pour les requêtes/réponses DNS et le mode TCP est généralement utilisé pour les transferts de zones. Cependant, compte tenu de la taille limitée à 512 octets des messages DNS en mode UDP, certaines requêtes peuvent provoquer le passage en mode TCP si la taille de la réponse dépasse cette limite.

Dans ce cas, le client reçoit dans un premier temps un message dont la section réponse (*answer section*) est vide et dont le bit `TC` (*Truncated*) est positionné, ce qui signifie implicitement que le client est invité à réinterroger le serveur en mode TCP. Au passage, ce scénario constitue un argument justifiant le fait que le port 53 en TCP ne doit pas être ouvert exclusivement pour les transferts de zones.

Notons qu'un basculement trop fréquent en mode TCP risque de consommer davantage de ressources et dégrader par conséquent les performances du serveur DNS. Certains nouveaux types d'enregistrements (tels que le type `AAAA`) risquent d'augmenter la taille des réponses DNS de manière significative, ce qui risque de faire basculer plus souvent les requêtes/réponses DNS en mode TCP. Aujourd'hui, ce dépassement n'arrive que rarement car la plupart des réponses DNS ne dépasse guère les 400 octets. En effet, les sections `answer`, `authority` et `additional`, qui constituent la majeure partie de la réponse DNS, ne contiennent qu'un nombre limité d'enregistrements si cette réponse ne concerne pas directement une zone de haut niveau telle que `.com`, `.net`, `.fr`, `.de`...

Face à ce risque, l'extension EDNS.0 (RFC 2671) a été proposée et est déjà déployée dans les versions récentes des logiciels DNS. Cette extension, permet à un client DNS d'informer le serveur interrogé qu'il est capable de supporter des réponses de taille supérieure à la limite des 512 octets (4096 octets par exemple). Ainsi, en présence d'IPv6, le support d'EDNS.0 devient fortement recommandé. À noter que le support d'EDNS.0 devient également indispensable en présence des extensions DNSSEC.

Le faible taux de pénétration d'EDNS.0 dans les logiciels DNS (surtout les clients) est resté pendant plusieurs années l'un des principaux motifs du refus de l'IANA/ICANN de publier de nouvelles adresses (IPv4 ou IPv6) pour des serveurs de la racine. À partir du 4 février 2008, l'IANA publie dans la zone racine l'adresse IPv6 (enregistrement AAAA) des serveurs racine supportant le transport IPv6 et la nouvelle version du fichier de de démarrage (typiquement `named.root` pour BIND 9) contient également ces adresses.

Notons enfin que des informations sur les adresses IPv4 et IPv6 des serveurs de la racine ainsi que sur la répartition géographique de ces serveurs sont publiées sur le site web : http://www.root-servers.org/ .

D) Glue IPv6

La zone racine publie également les adresses des différents serveurs DNS pour chacun des domaines de haut niveau (TLD : *Top Level Domain*). Ces adresses, appelées « glues » sont nécessaires pour que le processus de résolution DNS puisse démarrer. En effet, rappelons que les serveurs DNS de la racine ne sont pas censés résoudre eux-mêmes les requêtes. Leur rôle est de faire le premier aiguillage (*referal*) vers des serveurs de haut niveau (TLD). L'information d'aiguillage comprend la liste des serveurs du TLD sous l'arborescence duquel se trouve la ressource ainsi que les adresses (glues) associées à ces serveurs. Sans ces glues, le processus de résolution risque de tourner en rond.

En attendant que les serveurs de la racine puissent recevoir des requêtes DNS et y répondre en IPv6, les TLD avaient œuvré pendant des années à l'introduction dans la zone racine des « glues » IPv6 qui lui sont associées. L'IANA/ICANN ont enfin été convaincus que la publication des glues IPv6 des serveurs DNS de TLD supportant IPv6 peut se faire sans risque pour la stabilité du DNS. L'ICANN/IANA a démarré en juillet 2004 la publication des glues IPv6 des TLD dans la zone racine. Les trois TLD `.fr`, `.jp` et `.kr` ont vu les premiers leur glue IPv6 publiée.

E) Publication des enregistrements AAAA dans le DNS

On choisit généralement de publier dans le DNS le (ou les) enregistrement(s) `AAAA` associé(s) à un équipement donné lorsque l'on souhaite que les applications initiant des communications à destination de cet équipement découvrent le support du transport IPv6. Par exemple, un navigateur supportant IPv6, découvre via le DNS qu'il est possible de se connecter en IPv6 au site `http://www.afnic.fr/`. Il peut alors choisir de privilégier la connexion http au serveur en IPv4 ou en IPv6. Or avec l'intégration progressive

d'IPv6, certaines applications s'exécutant sur l'équipement dont l'adresse IPv6 est publiée dans le DNS peuvent ne pas supporter IPv6. Ainsi, l'on risque de se trouver par exemple dans la situation suivante :

- l'équipement `foo.example.com` héberge plusieurs services : web, ftp, mail, dns ;
- les serveurs web et dns s'exécutant sur `foo.example.com` supportent IPv6 mais pas les serveurs ftp et mail ;
- une adresse IPv6 est publiée dans le DNS pour `foo.example.com` ;
- un client ftp supportant IPv6 tente de se connecter au serveur s'exécutant sur `foo.example.com`. Le client sélectionne alors l'adresse IPv6 de `foo.example.com` comme adresse destination ;
- la tentative de communication en IPv6 échoue. Selon les implémentations, certains clients réessaient (ou non) d'autres adresses IPv6 s'il y en a ou passent (ou non) en IPv4 en dernier recours.

Afin de pallier ce problème, il est recommandé d'associer des noms DNS aux services et non aux équipements. Ainsi, pour notre exemple précédent, il serait judicieux de publier dans le DNS d'une part, les noms `www.example.com` et `ns.example.com` avec adresses IPv6 (et IPv4 éventuellement) et d'autre part, les noms `ftp.example.com` et `mail.example.com` avec des adresses IPv4 uniquement. L'enregistrement `AAAA` pour `foo.example.com` n'est alors publié que lorsque l'on a la certitude que toutes les applications s'exécutant sur cet équipement supportent IPv6.

Par ailleurs, le DNS étant une ressource publique, il est fortement déconseillé (sauf si l'administrateur DNS sait très bien ce qu'il/elle fait !) d'y publier des adresses IPv6 non joignables de l'extérieur, soit à cause d'une portée faible (adresses lien-local par exemple), soit parce que toutes les communications provenant de l'extérieur du réseau et allant vers ces adresses sont filtrées. Notons que cette règle est déjà appliquée pour les adresses IPv4 privées (RFC 1918) et que certains logiciels DNS récents supportent aujourd'hui les vues DNS (on parle de *two-face DNS*, de *split-view DNS* ou encore de *split DNS*). Avec ce système de vues, la réponse à une requête DNS dépend de l'origine du client. Par exemple un client appartenant au réseau interne pourra voir les adresses privées des équipements. Les clients externes, quant à eux, ne verront que les adresses globales et joignables de l'extérieur.

6) Découverte de la liste de serveurs DNS récursifs

L'auto-configuration IPv6 sans état, telle que spécifiée dans le RFC 4862, n'a pas prévu de mécanisme de découverte automatique de la liste des serveurs DNS récursifs (cache). En revanche, il était prévu que ces informations complémentaires soient fournies par l'auto-configuration avec état. Les spécifications du protocole d'auto-configuration avec état, DHCPv6, ont mis longtemps (six ans environ) à passer en RFC (RFC 3315) et le besoin de découverte des serveurs DNS récursifs s'est accentué davantage. En effet, afin de renforcer le déploiement d'IPv6, la communauté IPv6 s'était vite trouvée dans l'urgence de mettre en oeuvre un mécanisme de découverte automatique du DNS avec ou sans DHCPv6 (qui était justement près de sortir).

Trois propositions ont ainsi vu le jour dans le cadre des travaux des groupes « ipv6 », « dhc » et « dnsop ». C'est le groupe dnsop qui a pris en charge les débats sur ces propositions. Les co-auteurs de ces trois propositions ont conjointement rédigé un document synthétique (RFC 4339) décrivant pour chaque technique le fonctionnement ainsi que les scénarios d'utilisation. Ce document donne également des

recommandations pratiques quant à la solution ou à la combinaison de solutions à adopter en fonction de l'environnement technique dans lequel se trouvent les équipements à configurer.

Voici maintenant un résumé des trois propositions :

- Proposition 1, mécanisme à base de DHCP : deux solutions légèrement différentes ont été proposées. Elles proposent toutes les deux d'utiliser la même option « DHCPv6 DNS Recursive Name Server » spécifiée dans le RFC 3646 :
  - découverte via un serveur DHCPv6 complet (RFC 3315 : c'est-à-dire qui alloue dynamiquement les adresses IPv6 ;
  - découverte du DNS via un serveur DHCPv6-lite (RFC 3736) : celui-ci n'alloue pas d'adresses IPv6 mais il est simplement chargé d'informer les clients des différents paramètres à utiliser (DNS récursif, serveur NTP, serveur d'impression, ...) ;
  - Dans les deux cas, si l'équipement est en double pile et s'il est configuré à la fois avec DHCPv4 (pour IPv4) et avec DHCPv6 (pour IPv6), il faut définir une politique d'arbitrage entre les deux listes de serveurs DNS récursifs obtenues si celles-ci sont incohérentes ;
- Proposition 2, RFC 5006, mécanisme à base d'annonce de routeur (RA): cette proposition apporte un complément à l'auto-configuration sans état (RFC 4862) et consiste à surcharger l'annonce du routeur, en tant que message de la découverte des voisins (RFC 4861) par l'information DNS nécessaire. Une telle extension n'a pas été standardisée à ce jour, le RFC étant dans un état EXPERIMENTAL ;
- Proposition 3, mécanisme à base d'adresses anycast bien connues (*Well-known anycast addresses*) : des adresses IPv4 et IPv6 anycast qui seraient connues par tous les clients et préconfigurées automatiquement par le logiciel d'installation du système d'exploitation de l'équipement. Cette proposition semble avoir été abandonnée.

7) Propagation et mise à jour dynamique du DNS

La résolution DNS classique est optimisée grâce à l'introduction, d'entités intermédiaires : les serveurs récursifs ou caches. En effet, lorsqu'un client DNS soumet une requête à un serveur récursif, ce dernier se charge de la relayer à une chaîne de serveurs mieux renseignés que lui et de rechercher, de proche en proche, la réponse à la requête en question. Cette chaîne commence généralement par un serveur racine et se termine par un serveur autoritaire (primaire ou secondaire) sur la zone contenant le nom de domaine objet de la requête en question.

Le serveur maintient alors cette réponse dans son cache, pendant une durée de vie donnée (TTL : *Time To Live*). Le TTL est configurable selon la fréquence de changement des informations dans la zone concernée et il est généralement compris entre quelques dizaines de minutes et quelques jours. Si un client envoie une requête pour laquelle la réponse est encore dans le cache avant l'expiration de sa durée de vie, le serveur cache restitue alors cette réponse au client sans avoir à refaire l'interrogation de toute la chaîne de serveurs qui l'y a conduit. Par conséquent, le client prend le risque que la ressource qu'il vient tout juste de récupérer soit, dans le même temps, déjà modifiée sur les serveurs autoritaires sur la zone DNS contenant la ressource en question. Autrement dit, le TTL est une source potentielle d'incohérence des ressources DNS entre les serveurs autoritaires et les serveurs cache.

Ce risque d'incohérence existe également entre les serveurs autoritaires eux-mêmes. En effet, pour une zone DNS donnée, chaque serveur secondaire se sert du paramètre refresh figurant dans l'enregistrement SOA de la zone en question. Il spécifie le délai entre deux synchronisations et vaut souvent plusieurs heures de cette zone.

En cas de mise à jour récente sur le serveur primaire, le rafraîchissement de la zone en question, sur les serveurs secondaires, s'impose. Par conséquent, un serveur récursif n'est en fait jamais sûr de la pertinence de l'information qu'il récupère car celle-ci risque de provenir d'un serveur secondaire mal synchronisé. Afin de pallier ce dernier problème d'incohérence entre serveurs autoritaires, deux extensions DNS ont été spécifiées :

- Notification (RFC 1996) (`NOTIFY`) et
- Transfert incrémental (RFC 1995) (`IXFR`).

Ces deux protocoles permettent de faciliter et d'accélérer la synchronisation des zones DNS hébergées par les serveurs secondaires. Le protocole de notification est utilisé par un serveur DNS primaire pour informer ses serveurs secondaires qu'une zone a été modifiée. Les serveurs secondaires peuvent alors récupérer la nouvelle zone immédiatement, sans attendre le délai normal de synchronisation (donné par le paramètre « `refresh` »). Le protocole de transfert incrémental est utilisé pour optimiser le volume de l'échange entre serveur primaire et secondaire : si les modifications d'une zone sont faibles, on transfère seulement ces modifications et non la zone complète.

En dépit des palliatifs `NOTIFY` et `IXFR`, le mécanisme classique de résolution DNS reste inadapté aux réseaux dans lesquels les équipements sont susceptibles de changer d'adresse(s) très fréquemment et s'attendent de surcroît à être immédiatement contactés sur leur(s) nouvelle(s) adresse(s). Ainsi, il est devenu nécessaire, il y a quelques années, d'étendre les fonctionnalités du DNS afin qu'il puisse suivre et restituer en temps réel l'évolution des adresses des équipements dans les environnements dynamiques. De telles fonctionnalités sont, bien entendu, indépendantes de la version d'IP même si l'auto-configuration dans IPv6 ne fait qu'accentuer ce besoin.

L'IETF a proposé une solution au problème de mise à jour dynamique du DNS dans le (RFC 2136) (*DNS Dynamic Update*). Ce mécanisme permet à un client de soumettre aux serveurs DNS autoritaires sur un nom de domaine donné, une requête de mise à jour DNS concernant ce nom de domaine. Les seules opérations possibles de mise à jour sont celles de l'ajout d'enregistrements DNS (`RRs`) ou de suppressions de RRs ou de RRsets (ensemble de RRs de même nom, classe et type). Dans ces cas-là, le TTL est relativement très court (de 0 secondes à quelques minutes). Les requêtes les plus fréquentes sont celles de l'ajout ou de la suppression de l'adresse IP d'un équipement dans le DNS direct (correspondance du nom vers l'adresse) ou de son nom dans le DNS inverse (correspondance de l'adresse vers le nom).

Dans le cas de l'auto-configuration IPv6 sans état (RFC 4862) (cf. Configuration automatique), le client de mise à jour dynamique du DNS (`nsupdate` par exemple) peut s'exécuter sur l'équipement-même concerné par la mise à jour. Dans le cas de l'auto-configuration avec état (DHCP) (cf. Configuration avec état :DHCPv6), le client de mise à jour peut soit s'exécuter sur l'équipement concerné soit être couplé au serveur DHCP.

Le protocole de mise à jour dynamique du DNS See (RFC 2136) n'a pas été sécurisé à la base, c'est-à-dire, il n'a pas prévu l'authentification des seuls clients autorisés à modifier des enregistrements DNS. Le RFC 3007 propose donc de sécuriser les transactions de mise à jour du DNS, notamment en utilisant les techniques `TSIG` (RFC 2845), `TKEY` (RFC 2930) ou `SIG(0)` (RFC 2931) :

- La première technique, `TSIG` (*Secret Key Transaction Signatures for DNS*) permet l'authentification des parties de la transaction et de signer les messages DNS à l'aide d'une clé secrète symétrique. Soulignons que `TSIG` ne décrit pas le mécanisme par lequel la clé secrète est mise en place (cette clé est donc configurée par défaut manuellement).
- La deuxième technique, `TKEY` (*Secret Key Establishment for DNS*), vient compléter la première : elle permet d'automatiser la construction et la mise en place de la clé secrète à utiliser par `TSIG`.
- Enfin, la troisième technique, `SIG(0)` (*DNS Request and Transaction Signatures*), permet d'authentifier les parties de la transaction et de signer les messages DNS à l'aide d'une paire de clés (Clé publique/Clé privée) dont la partie publique est stockée dans le DNS grâce à des enregistrements de type `KEY`.

Malheureusement, la mise à jour dynamique sécurisée du DNS n'est toujours pas déployée à grande échelle. En effet, d'un côté, même si la technique `TSIG` est largement implémentée, elle n'est pas extensible (« scalable ») et de l'autre, la technique `SIG(0)` n'est que partiellement implémentée aujourd'hui. Ainsi `SIG(0)` est partiellement implémentée dans la distribution BIND 9.3.0 et supérieures.

En outre, la mise à jour dynamique du DNS devient problématique si celle-ci résulte de la configuration automatique d'un équipement IPv6 dans un réseau étranger (par exemple dans le cadre de la mobilité). En effet, si cet équipement peut théoriquement soumettre une requête de mise à jour de son (ou de ses) enregistrement(s) `AAAA` dans sa zone DNS direct, il n'est généralement pas autorisé à soumettre des requêtes de mise à jour de son (ou de ses) enregistrement(s) `PTR` dans la zone DNS inverse car celle-ci est sous l'autorité de serveurs DNS dans le réseau visité.

VII) Supports de transmission

La méthode de transport d'un datagramme IPv6 entre deux machines directement reliées entre elles par un lien physique est le même que pour IPv4 : le datagramme est tout d'abord routé vers une interface d'émission qui l'encapsule dans une trame (PDU de niveau 2 dans le modèle de référence de l'OSI) ; cette trame est transmise sur le lien vers l'adresse physique de la machine destination (cette adresse sur un lien sera appelée Adresse MAC dans la suite) ; la machine destination reçoit la trame sur son interface, la décapsule et la traite.

Les différences avec IPv4 sont :

- Le code protocole encapsulé de la trame est différent. Par exemple, pour les réseaux à diffusion, le code est 0x86DD alors que pour IPv4 le code est `0x0800`. À l'origine, il était prévu de garder le même code et d'assurer l'aiguillage entre IPv4 et IPv6 en utilisant le champ version du paquet. Mais

certains équipements ne vérifient pas la valeur de ce champ et auraient eu un comportement incontrôlable en essayant de traiter un paquet IPv6 comme un paquet IPv4.

- Le calcul de l'adresse MAC destination change. Par exemple sur un réseau à diffusion le calcul est fait en IPv4 par le protocole ARP, alors qu'en IPv6 on utilise le protocole de découverte de voisins.
- La taille minimale d'une trame est passée à 1 280 octets; ceci peut forcer certains protocoles à utiliser plusieurs trames par datagramme IPv6.
- Enfin, certains protocoles ont des parties propres à IPv4. Ces parties doivent être modifiées. C'est le cas des protocoles de contrôle et de compression de PPP.

1) Réseaux à diffusion

Les réseaux à diffusion ont tous une approche de transport similaire, utilisant le protocole de découverte de voisins pour trouver l'adresse du destinataire. Ce chapitre décrit les réseaux les plus courants, sans chercher l'exhaustivité.

A) Ethernet (RFC 2464)

Les datagrammes IPv6 utilisent l'encapsulation standard Ethernet V2, chaque trame contenant un seul datagramme. Nous décrivons ici le cas de l'Ethernet natif, mais la méthode s'étend immédiatement aux VLAN IEEE 802.1q.

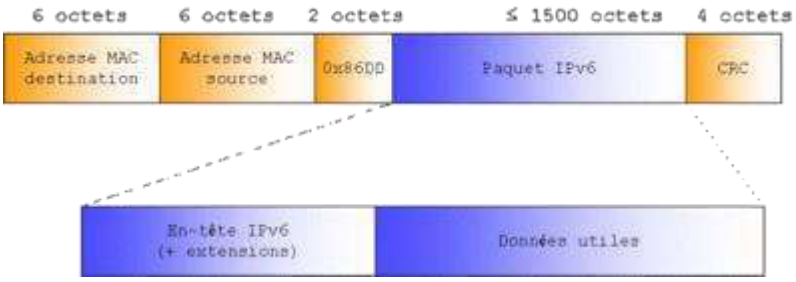


Figure 7-1 *Encapsulation Ethernet*

L'en-tête de la trame Ethernet contient les adresses Ethernet source et destination, et le champ type de protocole vaut `0x86DD`. La structure d'une trame est donnée à la figure Encapsulation Ethernet.

La taille maximale d'un datagramme pouvant être transmis directement par une interface Ethernet (MTU) est normalement de 1 500 octets. Une valeur différente peut être forcée par configuration manuelle ou en utilisant l'option MTU des annonces de routeurs.

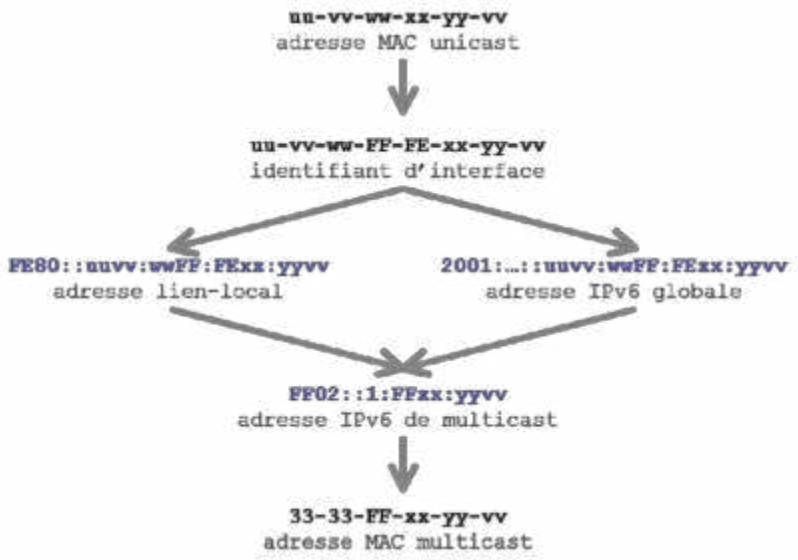


Figure 7-2 *Relation entre les adresses MAC et IPv6*

Pour la construction des adresses lien-local et des adresses auto-configurées, l'identifiant d'interface est celui dérivé de l'adresse MAC IEEE 802 constructeur de l'interface Ethernet, selon le procédé décrit au paragraphe Identifiant d'interface. Par exemple si une carte Ethernet a pour adresse constructeur `34:56:78:9A:BC:DE`, l'identifiant d'interface sera `36-56-FF-FE-9A-BC-DE` et l'adresse lien-local de l'interface aura pour valeur `FE80::3656:78FF:FE9A:BCDE`. La figure Relation entre les adresses MAC et IPv6 montre les relations entre les adresses MAC et IPv6. L'identifiant d'interface dérive de l'adresse MAC. À partir de cet identifiant est construit l'adresse lien-local et le plus souvent l'adresse dans le plan agrégé. L'adresse de multicast sollicité est construite à partir des trois derniers octets des adresses unicasts. De cette adresse de multicast sollicité est déduite l'adresse MAC de multicast.

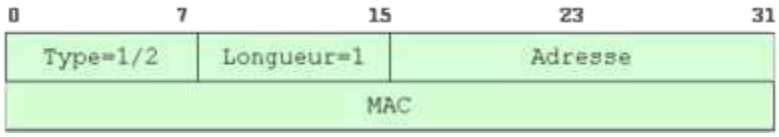


Figure 7-3 *Option adresse physique Source/Cible pour Ethernet/FDDI*

Pour une adresse IPv6 destination unicast ou anycast, le calcul de l'adresse MAC correspondant est fait par le protocole de découverte de voisin . Dans les messages du protocole, le format de l'option découverte de voisins est donné par la figure Option adresse physique Source/Cible pour Ethernet/FDDI avec :

- `type` : type de l'option (1 ou 2),
- `longueur` : 1 (8 octets).

Figure 7-4 *Calcul de l'adresse MAC destination pour un multicast IPv6*

Pour les adresses de multicast, le protocole de découverte des voisins n'est pas utilisable. L'adresse Ethernet est calculée en concaténant le préfixe `0x3333` et les 4 octets de poids faible de l'adresse IPv6 (cf. figure Calcul de l'adresse MAC destination pour un multicast IPv6).

B) Encapsulation LLC

L'autre encapsulation utilisée est l'encapsulation LLC/SNAP avec adresses MAC de 48 bits. Le champ type protocole vaut aussi `0x86DD`. La structure d'une trame est donnée par la figure Encapsulation des paquets IPv6 avec :

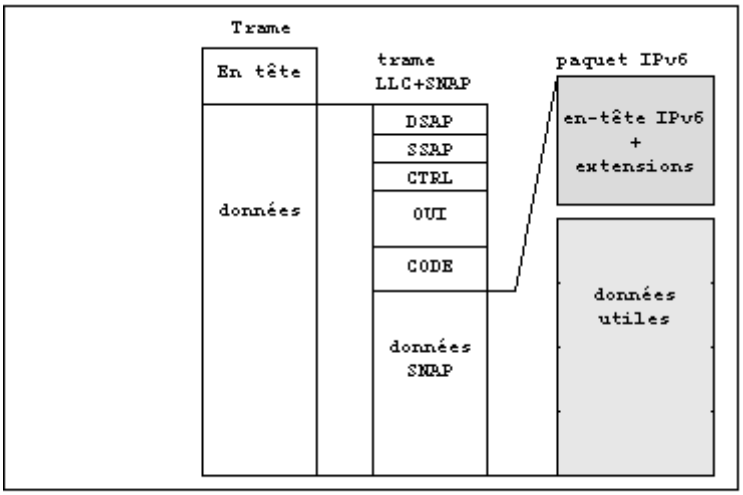


Figure 7-5 . Encapsulation des paquets IPv6

- FC : «Frame Code» (Doit être dans l'intervalle 0x51 - 0x57).
- DSAP, SSAP : `0xAA`, indiquant une encapsulation SNAP.
- CTRL : `0x03`, indiquant une information non numérotée.
- OUI : `0x000000` (*Organizationally Unique Identifier*).
- CODE : `0x86DD` (code protocole indiquant un contenu IPv6

Le principe régissant le calcul de l'identifiant d'interface et celui de l'adresse MAC à partir d'une adresse IPv6 de multicast est le même que pour Ethernet. L'option Découverte des voisins est aussi la même que pour Ethernet.

Cette encapsulation est utilisée pour FDDI (RFC 2467), IEEE 802.3 et Token-Ring (RFC 2470)

Pour FDDI, les datagrammes sont transmis dans des trames FDDI asynchrones avec jeton simple, chaque trame contenant un datagramme. Le MTU IPv6 par défaut est de 4352 octets. Toutefois, à cause de la présence possible de ponts IEEE 802.1d (Spanning Tree,...), le MTU effectif peut être plus faible (par exemple en présence de ponts Ethernet/FDDI, le MTU doit être de 1 500 octets). La valeur par défaut du MTU peut donc être modifiée, soit par configuration manuelle, soit en utilisant l'option MTU des paquets annonce du routeur.

Pour Token-Ring (RFC 2470), la taille maximale possible pour un paquet est très variable à cause de la possibilité du source routing qui ajoute des informations pour indiquer le chemin à travers les ponts. La

valeur du MTU est donc configurable, avec une valeur défaut de 1 500, mais les valeurs de longueur de trame indiquées dans les trames de source routing peuvent être prises en compte. La correspondance entre adresse multicast et adresse physique MAC n'est pas aussi simple qu'avec Ethernet ou FDDI, les composants pour l'anneau à jeton ne permettant de positionner qu'un seul bit dans l'adresse MAC multicast. On utilise donc seulement 10 classes d'adresses.

2) Réseaux NBMA

Les réseaux NBMA (Non Broadcast Multiple Access) posent un problème en IPv6 car on ne peut pas utiliser le protocole générique de découverte de voisins pour trouver le destinataire des trames. Il faut donc soit utiliser des tables statiques (par exemple en déclarant des circuits virtuels permanents), soit ajouter une couche de protocole permettant le multicast au transport afin de pouvoir utiliser le protocole de découverte de voisins, soit utiliser un protocole spécifique au réseau NBMA, soit considérer le réseau comme un arbre centré sur un routeur dédié qui possède une connexion point à point avec tous les autres matériels (cas de ISATAP, et GPRS). Parmi les réseaux NBMA, on peut citer ATM, GPRS, X.25 et Frame Relay.

Pour X.25 on peut utiliser une encapsulation semblable à celle utilisée par IP(v4) sur X.25. Un NLPID (Network Layer Protocol Identifier) spécifique a été réservé pour IPv6, `0x8E`. Comme les datagrammes IPv6 ont un MTU de 1280 et les paquets X.25 sont de 128 octets, un datagramme IPv6 peut être transmis dans plusieurs paquets X.25 successifs (utilisation du bit "suite" `M` des paquets données de X.25).

Pour ATM, on utilise une encapsulation LLC/SNAP dans une trame ALL5, le MTU étant négociable avec un défaut de 9 180 octets.

3) Liaisons point à point

Les liaisons point à point relient de manière fixe deux machines. En général il n'existe donc pas de notion d'adresse MAC, un datagramme est systématiquement envoyé à l'extrémité distante du lien. Pour le cas où l'adresse destination est multicast, le datagramme IPv6 est dupliqué et transmis vers la machine locale aussi bien qu'à la machine distante. Il reste un problème propre à IPv6 à prendre en compte : comme il n'existe pas en général pas d'adresse MAC IEEE 802 ou EUI-64 sur des liens point à point, il n'y a pas de méthode standard pour définir l'adresse IPv6 lien-local d'un interface point à point.

Il existe plusieurs types de liens point à point :

- Le plus simple est un lien physique, par exemple une ligne série. En IPv4 il existe deux protocoles de transport standard sur ligne série, SLIP et PPP. En IPv6 seul PPP a été défini, SLIP étant considéré comme obsolète car ne permettant pas une authentification suffisante. PPP est aussi utilisable pour d'autres liens physiques comme les fibres optiques en utilisant PPP sur SONET/SDH (Synchronous Optic NEtwork et Synchronous Digital Hierarchy, (RFC 2615)).

- Le deuxième type est l'utilisation d'une connexion fixe dans un réseau multipoint, par exemple l'utilisation d'un VC ATM ou X25 pour créer une liaison point à point entre deux machines. Dans ce cas l'encapsulation utilisée pour transporter les datagrammes IPv6 est celle définie pour le réseau multipoint. Le protocole est simplifié car il n'y a pas besoin de déterminer dynamiquement l'adresse destination de la trame.
- Un troisième type est formé des différents tunnels de transport de IP au dessus de IP. Ils seront détaillés plus bas au paragraphe Tunnels.
- Un dernier type apparaît avec l'utilisation de GPRS (2G ou 3G). Dans ce cas un ensemble de tunnels forme un lien logique entre l'équipement et le routeur de sortie du réseau GPRS.

A) PPP (RFC 2472)

L'encapsulation est faite dans une trame PPP Data Link Layer . Chaque datagramme IPv6 forme une trame séparée. Le champ protocole de la trame doit être `0x57`.

Le protocole de contrôle de PPP pour IPv6 s'appelle IPV6CP. Il permet la configuration, la validation et l'invalidation des modules IPv6 de PPP. Chaque datagramme IPV6CP est transmis dans une trame PPP Data Link Layer de champ protocole `0x8057`. Le protocole IPV6CP permet de négocier deux options, la valeur de l'identifiant d'interface et la compression.

Comme il n'existe pas d'adresse MAC IEEE 802 ou EUI-64 pour les interfaces PPP, il n'y a pas de valeur par défaut standard pour l'identifiant d'interface. Comme discuté dans le paragraphe Identifiant d'interface, si la machine (ou une autre interface) possède un EUI-64 ou une adresse MAC IEEE 802, on utilise l'identifiant associé. Sinon il faut fournir une valeur unique (non universelle) à partir d'un numéro de série, ou par exemple d'un générateur aléatoire. Dans tous les cas, la négociation d'identifiant d'interface de IPV6CP doit être utilisée pour forcer l'unicité de l'identifiant.

Le MTU est variable car déterminé par la connexion PPP sous-jacente. Il doit être au moins égal à 1 280.

Il faut remarquer que PPP n'a pas de notion d'adresse physique, donc l'option Découverte des voisins du protocole de découverte de voisin n'est pas utilisée. De même, il n'est pas nécessaire de définir un traitement particulier pour les paquets multicast : comme sur tout lien point à point, le multicast IPv6 est supporté sans action particulière.

Une méthode de compression des en-têtes IPv6 du même type que celle utilisée par IPv4 a été proposée. Elle est optionnelle et son utilisation est négociée par IPV6CP.

Suivant le contexte d'utilisation deux mécanismes de compression peuvent être utilisés :

- Pour les réseaux où le taux d'erreur est relativement faible et pour lesquels la bidirectionnalité est garantie, le RFC 2507 définit une méthode de compression qui est une formalisation et une amélioration de la méthode définie par Van Jacobson pour PPP/IPv4/TCP.
- Pour les réseaux de troisième génération, le problème est plus complexe. D'abord le taux d'erreur est plus important, de plus certaines applications pour le multimédia peuvent être monodirectionnelles et utiliser l'encapsulation UDP/RTP. Le groupe de travail ROHC (*RObust Header Compression*) travaille à la définition d'un protocole pour cet environnement.

B) Compression Robuste des en-têtes

Dans les réseaux téléphoniques de troisième génération IPv6 avait été retenu pour transporter la voix. Un mécanisme de compression robuste peut réduire le temps de transmission et augmenter l'utilisation d'une ressource très convoitée telle que le support de transmission Hertzien. Pour les services interactifs de voix sur IP et les liaisons cellulaires le protocole utilisé est le protocole RTP. La taille de l'en-tête d'un paquet IPv6/UDP/RTP varie de 60 octets à 120 octets, et de 40 octets à 100 octets pour un paquet IPv4/UDP/RTP. La charge utile, compte-tenu de l'algorithme de compression de la voix et des contraintes temps réel, varie entre 15 et 20 octets. La compression d'en-têtes est possible sur différentes couches du modèle ISO/OSI mais c'est au niveau de la couche réseau IP qu'elle est la plus efficace, puisque l'on a connaissance du format des paquets (et de celui des couches supérieures). Mais la compression signifie également réduction de la redondance dans l'information transmise, ce qui est antagoniste avec des transmissions bruitées. Les travaux présentés sont basés sur les résultats de standardisation de l'IETF du groupe ROHC (*Robust Header Compression*) et en particulier sur le RFC 3095.

Le principe à la base de la compression d'en-têtes est la réduction de la redondance de l'information contenue dans un en-tête, mais aussi de la redondance entre plusieurs en-têtes consécutifs. Ainsi, l'information qui ne change pas est envoyée au début de la session ou à un faible rythme et pour les autres champs, un mécanisme de prédiction ou de dépendance permet de réduire encore l'information transmise.

Le déroulement du protocole ROHC commence par une négociation, permettant au compresseur et au décompresseur de connaître les caractéristiques du lien et le profil à utiliser. Le mécanisme classifie les champs des en-têtes, l'analyse est basée sur le changement des valeurs de ces champs pendant la connexion, la classification se fait suivant cinq types de valeurs différents : INFERRED, STATIC-DEF, STATIC-KNOWN, CHANGING, et STATIC, qui forment les parties statique et dynamique de l'en-tête compressée ROHC.

ROHC maintient un contexte entre le compresseur et le décompresseur. Ce contexte contient une version non compressée du dernier en-tête envoyé et aussi l'information redondante dans le flot de données. Le contexte est gardé à la fois dans le décompresseur et dans le compresseur pour assurer la robustesse, et chaque fois que le compresseur doit envoyer des nouvelles valeurs, le contexte s'actualise. Si le contexte est perdu, il y a désynchronisation, et le décompresseur peut éventuellement à travers des acquittements reprendre le contexte. Sinon, le décompresseur doit attendre qu'une temporisation expire au niveau du compresseur pour retrouver le contexte.

Le principe de ROHC est d'envoyer l'information minimale pour que le décompresseur puisse reformer l'en-tête. L'élément clé est le CRC, calculé avant la compression, qui donne au décompresseur une information sur la validité de l'information qu'il possède et qui est susceptible de dériver suite à des erreurs de transmission.

Le mécanisme ROHC utilise des profils, des niveaux de compression, des modes d'opération et des modes de transition. Chaque mode d'opération a trois niveaux de compression, et chaque mode de transition travaille dans le mode d'opération précédent en utilisant les deux premiers niveaux de compression

jusqu'à la réception d'un acquittement pour changer de mode, comme le montre la figure Diagramme d'état de ROHC.

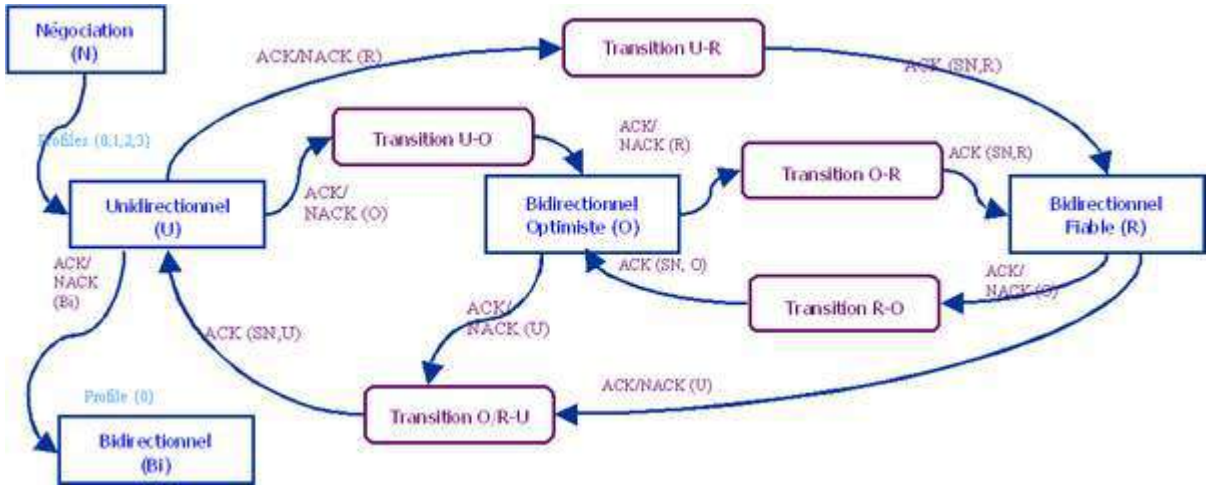


Figure 7-6 *Diagramme d'état de ROHC*

Les profils définissent les en-têtes protocolaires qui doivent être compressés dans l'en-tête. Ils permettent au décompresseur de connaître la version d'IP, si le flot utilise RTP ou ESP ou s'il s'agit d'un flot UDP seulement. Actuellement les profils définis sont les suivants, d'autres pourront être ajoutés dans le futur :

- Profil 0 sans compression. Si ce profil est choisi, seul l'identificateur de ROHC est ajouté pour que le décompresseur puisse reconnaître les paquets mais il n'y a pas de compression.
- Profil 1 compression des en-têtes IP/UDP/RTP. Ce profil est le plus générique, il compresse tout l'en-tête depuis IP jusqu'à l'en-tête RTP.
- Profil 2 compression des en-têtes IP/UDP. Ce profil est une variation du profil 1 sauf qu'ici la compression s'arrête au protocole UDP.
- Profil 3 compression des en-têtes IP/ESP. Ce profil compresse le protocole ESP, qui peut aussi être pris en compte comme une variation du profil 1.
- Profil 4 compression des en-têtes IP (RFC 3843). Ce profil ne compresse que les en-têtes du protocole IP (IPv4 et IPv6).
- Profil 7 pour la compression des en-têtes IP/UDP-lite/RTP (RFC 4019).
- Profil 8 pour la compression des en-têtes IP/UDP-lite (RFC 4019).

ROHC a trois niveaux de compression :

- Initialisation et Régénération (IR),
- Premier Niveau (FO),
- Deuxième Niveau (SO).

Chaque niveau de compression dans chaque mode d'opération utilise différents types de paquets (cf. See Différents en-têtes ROHC).

Différents en-têtes ROHC			
Niveau de compression	IR	FO	SO
Paquet Utilisé	IR (de 48 à 151 octets)	IR-DYN (de 21 à 84 octets) UOR-2 (3-18 octets)	UO-0 (1 octet), UO-1 (2 octets), R-0 (1 octet), R-0-CRC (2 octets), R-1 (2 octets)

Chaque paquet est envoyé avec une fréquence déterminée par différents facteurs : soit la réception d'un acquittement négatif ou positif, soit un délai dépassé, soit une mise à jour ou soit périodiquement en fonction de la confiance que le compresseur a de la qualité du lien. La taille de chaque paquet varie en fonction du type d'en-tête qui va être compressé, les limites des valeurs sont données dans le See Différents en-têtes ROHC.

Les niveaux de compression permettent de diminuer la taille de l'en-tête basée sur l'information que le compresseur a déjà envoyé. Dans le premier niveau de compression qui s'appelle initialisation et régénération, la taille des en-têtes envoyés varie entre 48 à 130 octets, et pour le deuxième niveau les en-têtes ont une taille comprise entre 3 à 84 octets.

Les modes d'opération permettent d'augmenter la performance du mécanisme car le mécanisme peut aller d'un mode d'opération à l'autre en fonction des caractéristiques de la liaison. Chaque mode d'opération a ses caractéristiques propres, le mode d'opération unidirectionnel et bidirectionnel optimiste sont plus complexes que le mode d'opération bidirectionnel fiable, parce qu'au début ni le compresseur ni le décompresseur n'ont d'information pertinente ni tous les paramètres pour effectuer la compression. Le compresseur change de niveau de compression en réduisant la taille de l'en-tête qu'il envoie, les changements sont faits chaque fois que le décompresseur assure au compresseur qu'il a l'information nécessaire pour reconstruire l'en-tête ou si le niveau de confiance en la liaison est suffisant dans le compresseur.

Le mode d'opération unidirectionnel possède différents algorithmes de contrôle de régénération. Il utilise deux temporisateurs pour effectuer la régénération de l'en-tête complète et un système de confiance (L) qui est basé sur le BER et le comportement des erreurs dans les liaisons. Dans ce mode d'opération, le compresseur contrôle la taille de l'en-tête utilisée, et s'il y a un changement dans l'en-tête il peut revenir à un niveau de compression inférieur pour donner toute l'information nécessaire au décompresseur. Le décompresseur ne peut pas communiquer avec le compresseur, les acquittements ne sont pas envoyés et tout est basé sur les mécanismes de contrôle. C'est le mode d'opération le moins performant mais le protocole assure que les en-têtes sont bien arrivés.

Le mode d'opération bidirectionnel optimiste est très similaire au mode d'opération unidirectionnel mais ici le décompresseur peut envoyer des acquittements pour confirmer les envois. Le mode optimiste n'utilise pas les deux temporisateurs mais il garde le système de confiance (L), les changements de niveaux de compression se font grâce à trois différents acquittements : les acquittements positifs/négatifs font une transition positive/négative d'un seul niveau de compression. Les acquittements statiques font une transition négative au niveau de compression plus bas. Si le compresseur reçoit un paquet qui va actualiser le contexte, le compresseur changera de niveau de compression en envoyant un en-tête plus grand.

Le mode d'opération bidirectionnel fiable travaille uniquement avec les acquittements reçus du décompresseur, chaque fois qu'il reçoit un acquittement positif/négatif le compresseur change de niveau de compression et revient à l'état d'initialisation avec un acquittement statique.

Les modes de transition sont déclenchés par le décompresseur, chaque fois qu'il est capable de travailler avec un nouveau mode d'opération il lance un acquittement avec le nouveau mode d'opération dans lequel il veut travailler. En général tous les modes de transition travaillent avec les deux premiers niveaux de compression du mode d'opération précédent qui peuvent actualiser le contexte. Tous les paquets envoyés pendant la transition contiennent le CRC pour vérifier l'information. Pendant les transitions le compresseur et le décompresseur gardent chacun deux variables de contrôle : Transition et Mode, si la variable transition a la valeur d'attente, la transition ne peut pas être déclenchée. Pour finir la transition, un acquittement avec le numéro de séquence et le mode d'opération doit être reçu par le compresseur, sinon la variable transition reste en attente et le mode d'opération conserve l'ancienne valeur. La seule transition différente est la transition d'Unidirectionnel à Optimiste qui est automatique. Pour faire les transitions au mode d'opération fiable (R), le contexte doit être mis en place. Pour les autres, la transition peut commencer à n'importe quel niveau de compression dans le mode d'opération actuel.

4) Tunnels

Un tunnel est un moyen de transporter des paquets IP directement entre deux points connectés par IP. L'extrémité émettrice du tunnel encapsule le datagramme IP dans un autre datagramme IP et l'envoie vers l'autre extrémité du tunnel. L'extrémité réceptrice reçoit le message, décapsule le datagramme IP et le traite. On peut voir ce mécanisme comme un lien point à point ou un lien NBMA, selon qu'on considère des relations 2 à 2 ou un émetteur en face du réseau Internet.

Il existe de nombreux formats de tunnels en IPv4 :

- IP dans IP (RFC 1853),
- l'encapsulation minimale pour la mobilité (RFC 2004),
- celui utilisé par le Mbone pour le multicast,
- le format GRE (*Generic Routing Encapsulation*, RFC 1701),
- L2TP (encapsulation de PPP sur IP, RFC 3437)
- ...

En IPv6, l'utilisation des en-têtes d'extension et le support du multicast en natif font que la beaucoup des utilisations spécifiques de tunnels par IPv4 n'ont a priori plus de raison d'être. Il reste cependant des protocoles importants qui utilisent explicitement des tunnels, si bien que les tunnels existent toujours en IPv6 : il existe un format générique, et des utilisations spécifiques pour la mobilité, pour la sécurité et, dans la phase de déploiement de IPv6, tant que la connectivité mondiale n'est pas assurée, pour relier des réseaux IPv6 isolés à travers l'Internet IPv4. Cette approche a été utilisée de manière intensive pour réaliser le réseau 6bone.

A noter que GRE (RFC 2473) a été également défini pour IPv6 pour transporter des protocoles non Internet dans une infrastructure IPv6.

A) Tunnel générique IP dans IPv6

Ce type de tunnel permet d'utiliser le réseau IPv6 pour transporter tout type de datagramme, en particulier des datagrammes IPv4 ou IPv6. Le RFC 2473 précise le principe général : envoi d'un paquet IPv6 contenant comme donnée le paquet encapsulé.

Dans le cas d'un paquet encapsulé IPv6, le type de donnée (champ En-tête suivant) utilisé est 41 (0x29) et dans le cas d'un paquet IPv4, le champ en-tête suivant vaut 4. Le RFC 2473 propose aussi un mécanisme de protection contre les bouclages dus à des tunnels imbriqués. Le MTU du tunnel est à configurer, une valeur possible est PMTU - LGH où PMTU est le Path MTU entre les deux extrémités du tunnel, et LGH est la taille de toutes les en-têtes ajoutées (en-tête IPv6, mais peut-être aussi routage, chiffrement, ...).

B) Transport de IPv6 sur IPv4

Le transport des paquets IPv6 à l'intérieur de paquets IPv4 est décrit dans le RFC 2893. Il utilise un tunnel point à point (ou configuré) qui établit une liaison fixe entre deux machines (en général des routeurs).

Le MTU est à configurer, et doit être au moins égal à 1280. Une valeur possible est 1280, ou PMTU - 20 où PMTU est le Path MTU IPv4 entre les deux extrémités du tunnel.

L'encapsulation est faite dans un paquet IPv4 de type protocole 41 (`0x29`). Chaque datagramme IPv6 forme un paquet séparé. Ce paquet peut être fragmenté par IPv4 (cf. figure Datagramme IPv6 encapsulé dans un datagramme IPv4).

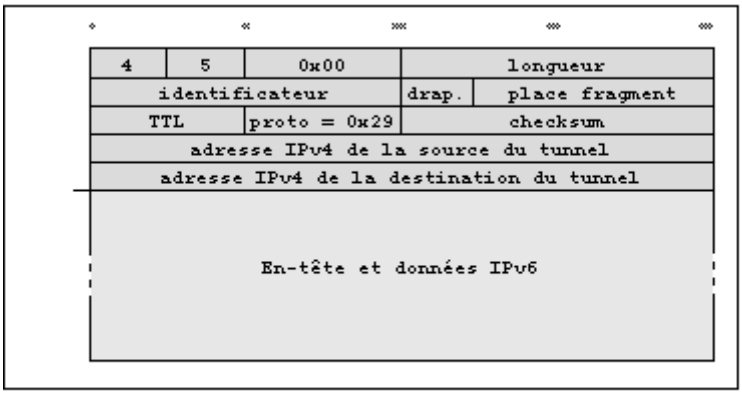


Figure 7-7. Datagramme IPv6 encapsulé dans un datagramme IPv4

Le tunnel utilise comme identifiant d'interface l'adresse IPv4 complétée en tête par des 0. Le tunnel doit implémenter les mécanismes de découverte des voisins, sans option d'adresse physique.

5) IPv6 dans la téléphonie mobile (UMTS)

A) Introduction

Le GSM a connu le succès que l'on sait pour la téléphonie mobile. Il a ensuite évolué pour permettre la transmission de données en offrant le service de transmission de paquet (GPRS : *Generalized Packet Radio Service*).

L'ITU (*International Telecommunication Union*) a la charge de la normalisation des systèmes dit de téléphonie de troisième génération appelée également IMT-2000 (*International Mobile Telecommunications 2000*). En Europe l'ETSI (*European Telecommunications Standards Institute*), responsable de la standardisation du GSM a créé le groupe 3GPP (*Third Generation Partnership Project*) en 1998 pour inclure des pays non européens dans ses travaux. Le travail de redéfinition est considérable et modifie en profondeur l'architecture des réseaux et des services existants. Toutefois elle se fera progressivement car plusieurs versions (ou *releases*) de la norme ont été échelonnées dans le temps. Au fil de ces versions, le protocole IPv6 est introduit dans les différents éléments de l'architecture et les mécanismes de cohabitation sont utilisés. L'architecture du GPRS a été reconduite dans les réseaux de troisième génération définis par le 3GPP et deviendra progressivement le mode de transfert principal des données lorsque les différents services utilisant le mode connecté seront transportés par le coeur de réseau paquet. Cette évolution est déjà sensible dans la version 5 qui sert de base à notre description.

Le service de transmission de paquet offert par l'UMTS (cf. figure Architecture très simplifiée de l'UMTS) offre une gestion de la mobilité transparente aux utilisateurs. Cette mobilité est gérée par un mécanisme de tunnel entre le routeur de sortie du réseau de téléphonie mobile et le terminal mobile. Les différentes fonctions et la signalisation nécessaire à la gestion de la session et de la mobilité sont assurées par les protocoles spécifiques du plan de contrôle. Le transfert des données elles-mêmes se fait dans le plan usager à l'aide de deux tunnels aboutés, l'un au-dessus du réseau d'accès radio, l'autre dans le coeur de réseau IP de l'opérateur.

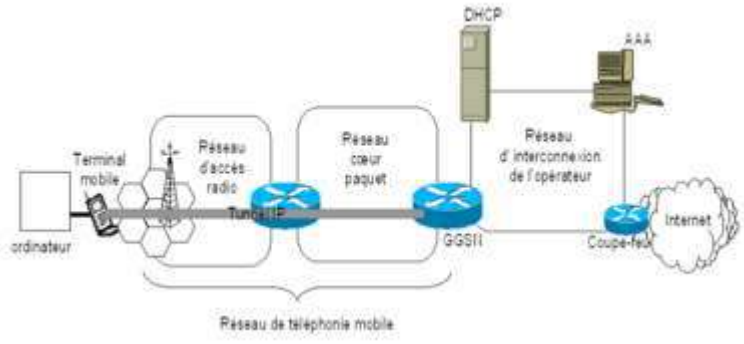


Figure 7-8 *Architecture très simplifiée de l'UMTS*

Quatre grandes classes de service peuvent être utilisées lors de l'établissement de la session. Elles sont classées par rapport à leur sensibilité au délai : conversationnelle, streaming, interactive et background. Ces classes définissent pour le moment le comportement de la liaison dans le réseau d'accès radio et leur utilisation dépend de l'abonnement souscrit par le client auprès de l'opérateur.

Lors de l'établissement de la session, le terminal mobile peut accéder à différents types de réseaux externes de données en fonction de ses besoins (par exemple l'Internet ou le réseau privé d'une entreprise). Il y a trois types de service :

- Le service PPP permet d'accéder directement au réseau du fournisseur d'accès de l'abonné à travers une liaison PPP prolongée par un tunnel IP (L2TP) au-dessus de l'Internet ; n'importe quel protocole de niveau réseau est ensuite utilisable.
- Les services IPv4 et IPv6 permettent d'offrir une connectivité IP au terminal mobile ou à un ordinateur.

Dans la suite, nous décrivons comment le service IPv6 est offert.

B) Architecture 3G

Le réseau d'un opérateur de téléphonie mobile est composé de plusieurs grandes parties (cf. figure Architecture très simplifiée de l'UMTS) : le réseau d'accès radio (WCDMA pour l'UMTS), les réseaux cœurs paquet et circuit qui comprennent aussi les serveurs gérant les données d'abonnement des utilisateurs et le plan de service ou IMS (*IP Multimedia Subsystem*).

Le réseau cœur du domaine paquet est un réseau IP qui interconnecte les réseaux d'accès radio et les serveurs de l'opérateur (facturation, localisation, ...). Ce réseau est aussi connecté à un réseau d'interconnexion lui-même connecté à l'Internet à travers un pare-feu (*firewall*). Le routeur de sortie du cœur de réseau paquet (GGSN) joue un rôle particulier dans la fourniture du service IPv6 car il termine les tunnels des terminaux mobiles et participe à l'établissement et au maintien des sessions établies. Il route aussi les paquets IPv6 émis par les équipements IPv6 vers le réseau d'interconnexion de l'opérateur.

Pour l'équipement IPv6 (ordinateur, PDA) qui utilise le terminal mobile (téléphone) comme un modem, l'interface réseau est une connexion PPP au-dessus de laquelle IPv6 est configuré grâce aux mécanismes standards adaptés au contexte de la téléphonie mobile.

C) Service IPv6

L'objectif du service IPv6 est d'offrir une connectivité IPv6 vers un réseau externe de données IPv6 (Internet dans la figure Architecture très simplifiée de l'UMTS). Ce service peut fonctionner en mode transparent ou en mode non transparent en fonction de l'implication du routeur frontière (GGSN) dans les procédures d'authentification et d'allocation d'adresse IPv6 du réseau externe de données. Dans le mode non transparent, le GGSN relaie les requêtes d'authentification et les sollicitations DHCP vers les serveurs du réseau externe, ces procédures font donc partie du processus d'établissement de la session. Dans le mode transparent ce sont les serveurs de l'opérateur, ou le GGSN directement, qui assurent l'authentification et l'allocation d'adresse.

Une adresse IPv6 statique peut être associée à chaque abonnement, dans le cas contraire l'adresse est dite dynamique et est allouée lors de l'établissement de la session (appelé aussi activation de contexte). Une adresse statique est choisie par l'opérateur au moment de la souscription de l'abonnement ; elle appartient à l'espace d'adressage de l'opérateur. Si l'adresse est dynamique une auto-configuration sans état ou avec état doit être réalisée par l'équipement. Le choix de la méthode d'auto-configuration est fait de manière standard par le GGSN lors de l'émission de l'annonce de routeur. L'équipement se conforme si possible à ce choix, sachant que l'auto-configuration avec état est facultative selon la norme.

D) Etablissement d'une session IPv6

Un terminal mobile peut établir une ou plusieurs sessions (contextes). Plusieurs sessions peuvent être associées à une même adresse IPv6 mais avec des qualités de service différentes, ce qui a pour objectif de différentier la qualité de service offerte aux différents flux IPv6. Les sessions peuvent aussi être associées à des adresses différentes (appartenant à des espaces d'adressage différents) et offrir la connexion vers plusieurs réseaux IPv6 externes. Dans ce cas, l'équipement utilise une interface PPP par session.

Lors de l'établissement de la session, un échange de message a lieu entre le terminal mobile et le GGSN. Pour la configuration IPv6 de l'interface PPP, les principes retenus dans la version 5 de l'UMTS sont d'attribuer un préfixe IPv6 de longueur 64 à chaque session, et de laisser le GGSN décider de la valeur de l'identifiant d'interface de l'adresse lien-local. Ainsi l'équipement n'a pas à effectuer la détection de duplication d'adresse (DAD) au moment du choix de l'adresse lien-local. Cet identifiant d'interface est transmis dans la confirmation d'établissement de session transmise par le GGSN au terminal mobile.

D'autres paramètres peuvent être demandés par le terminal mobile lors de la demande d'ouverture de session. Du point de vue de l'équipement informatique, la demande est alors transmise dans des options du protocole de contrôle IPv6CP lors de l'établissement de la connexion PPP entre l'équipement et le terminal mobile.

E) Configuration de l'interface IPv6

Le terminal mobile transmet l'identifiant d'interface fourni par le GGSN à l'équipement qui doit l'utiliser pour configurer l'adresse lien-local de l'interface PPP. Une fois son adresse lien-local configurée, l'équipement peut émettre une sollicitation de routeur pour déclencher l'émission immédiate d'une annonce de routeur. L'annonce de routeur qu'il reçoit du GGSN contient l'adresse du routeur par défaut et le préfixe IPv6 global. Elle indique aussi à l'équipement, comme prévu dans le standard sur la découverte de voisin, s'il doit effectuer une configuration avec ou sans état.

a) Configuration sans état (Stateless)

Le GGSN annonce un préfixe de son plan d'adressage (cas transparent) ou du plan d'adressage du réseau externe (cas non transparent). Il détermine le préfixe annoncé en utilisant un serveur Radius ou un serveur DHCPv6 pour une adresse dynamique et les données associées à l'abonnement pour une adresse statique. Il crée ensuite un état qui permettra de router les paquets à destination de ce préfixe vers l'équipement. Un seul préfixe doit être annoncé avec le bit `A` à 1 (construction de l'adresse de l'équipement à partir de ce préfixe) et sans le drapeau `L` à 0 (pas d'autres machines sur le lien), la durée de vie du préfixe est infinie.

Lorsqu'il reçoit le préfixe l'équipement construit son adresse IPv6 globale (voir Auto-configuration sans état) en utilisant un identifiant d'interface qu'il choisit mais n'effectue pas de détection de duplication d'adresse puisque le préfixe global annoncé est unique par construction.

Notons qu'il existe aussi la solution d'utiliser un préfixe partagé entre plusieurs sessions mais, dans ce cas, il faut soit obliger l'équipement à utiliser l'identifiant d'interface fourni par le GGSN lors de l'établissement de la session pour construire l'adresse globale, soit effectuer la détection de duplication d'adresse.

Lorsque le terminal a terminé de configurer son interface, il peut toujours utiliser DHCPv6 pour obtenir des paramètres comme l'adresse du serveur DNS (voir chapitre Nommage).

b) Configuration avec états (Statefull)

Dans le cas de la configuration avec état, l'équipement doit solliciter le serveur DHCPv6. Le GGSN agit dans ce cas comme un relais DHCPv6 transmettant les requêtes de l'équipement au serveur DHCPv6 configuré par l'opérateur (cf. figure Echanges de configuration). Lorsque le serveur DHCPv6 répond en attribuant une adresse IPv6 globale, le GGSN établit l'état qui lui sert à router les paquets à destination de l'équipement et transmet la réponse à ce dernier.

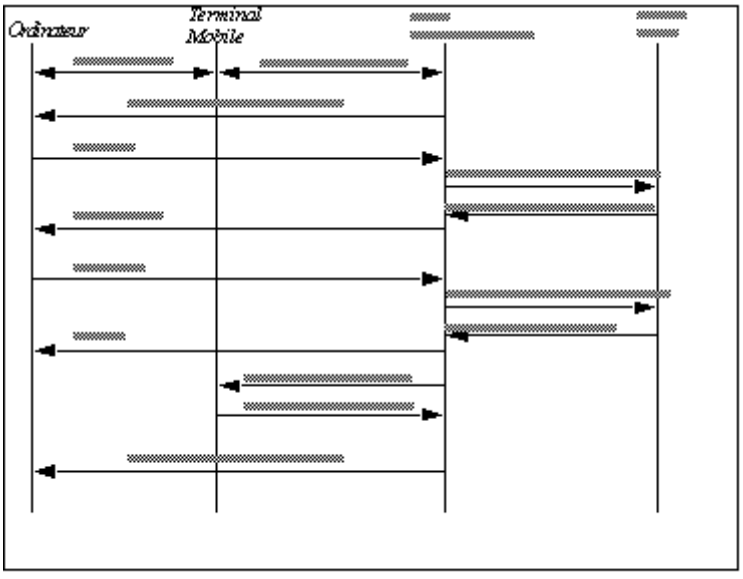


Figure 7-9 . Echanges de configuration

c) Configuration des constantes IPv6

Du point de vue de l'équipement, le GGSN se comporte comme un routeur IPv6 normal. Il est par contre nécessaire d'adapter la configuration des variables qui influent sur le comportement des mécanismes IPv6 pour adapter ceux-ci à l'environnement UMTS. Ainsi le délai maximum entre deux annonces de routeur (MaxRtrInterval) est de 6h et le minimum conseillé (MinRtrInterval) peut aller jusqu'à 4,5h, pour éviter de charger le lien UMTS qui est facturé à la quantité d'information transmise. La durée de vie des préfixes annoncés est infinie, et ceux-ci sont donc valides pour tout le temps de la session. Deux autres variables sont définies pour augmenter la fréquence des premières annonces de routeurs dans le but de réduire le temps de configuration automatique lorsque le terminal n'émet pas de sollicitation de routeur.

Les variables ci-dessus ainsi que le comportement du GGSN quant au mode d'auto-configuration sont décidés par l'opérateur pour chaque réseau externe de données (IPv6). Ainsi un même GGSN se comportera différemment en fonction du réseau que l'équipement cherche à joindre.

d) Support du multicast

Dans le cas ou l'opérateur mobile souhaite supporter le service multicast IPv6, le GGSN doit implémenter MLD (voir Gestion des abonnements sur le lien-local : MLD) et un ou plusieurs protocoles de routage multicast (DVMRP, MOSPF ou PIM-SM). Il doit aussi assumer le rôle de proxy multicast, c'est-à-dire :

- maintenir la liste des équipements ayant adhérés à tel ou tel groupe ;
- transmettre des rapports d'appartenance aux groupes aux routeurs multicast du domaine ;
- transmettre en point à point une copie des paquets multicast reçus à chaque équipement ayant adhéré au groupe.

F) Transition dans l'UMTS

Les réseaux de téléphonie mobile vont progressivement migrer leurs réseaux IPv4 vers IPv6. Trois méthodes vont être utilisées principalement :

- Une pile double (dual stack) IPv4/IPv6 dans le réseau cœur et les terminaux mobiles.
- L'utilisation de tunnels (tunneling) automatiques et configurés comme 6to4 or L2TP.
- Un protocole de traduction d'IPv4 à IPv6 dans le réseau comme NAT-PT.

Le See Versions d'IP pour les différentes versions de l'UMTS (3GPP), décrit l'évolution de l'utilisation des protocoles IPv4 et IPv6 dans les cœurs du réseau et pour les équipements terminaux comme le téléphone mobile. La standardisation du 3G est toujours en évolution.

Versions d'IP pour les différentes versions de l'UMTS (3GPP).

Phases	Réseau Coeur	Terminal mobile
Actuellement	IPv4 Utilisation de NAT	IPv4
Première Phase (Ilôts IPv6 sépares)	Utilisation de tunnels IPv6/IPv4	IPv4 Double Pile IPv4/IPv6
Deuxième Phase (déploiement élargie d'IPv6)	IPv6 Utilisation de tunnels vers IPv4	Double Pile IPv4/IPv6
Troisième Phase (Domination d'IPv6)	IPv6	IPv6

G) L2TP comme un outil de transition

Une solution possible pour la transition d'IPv6 dans les réseaux radio est l'utilisation du protocole L2TP (*Layer Two Tunneling Protocol*) entre un terminal IPv6 et son correspondant lies entre eux par le réseau UMTS. L2TP sera une prolongation du tunnel PPP en mode non transparente.

Le protocole L2TP définit deux entités fonctionnelles, le LAC et le LNS. Il s'agit de deux passerelles entre des réseaux d'accès, le réseau cœur et le réseau ou nœud correspondant qui sont de natures différents. Le LAC, ayant une interface vers le réseau d'accès, réalise la concentration et le traitement des appels PPP qui en émanent. Le LNS communique avec le LAC via ce même réseau. Un tunnel est créé entre le LAC et le LNS qui consiste d'une connexion de contrôle et zéro ou plusieurs sessions de données. Les sessions de données font la transmission de données du terminal i.e. des trames PPP, la connexion de contrôle gère les messages d'établissement, entretien et de libération des sessions de données.

H) IMS

L'IMS (*IP Multimédia Core Network Subsystem*) est une architecture permettant de supporter des services multimédia à travers une infrastructure SIP qui n'utilisent que IPv6. Il est constitué de proxy, de serveurs et de passerelles pour l'interaction avec des services non IP (lien avec la téléphonie classique). Le mobile doit utiliser une session de type IPv6 pour ses connexions à l'IMS.

Les problèmes liés à la transition peuvent se diviser en deux parties : les scénarios concernant l'utilisation du GPRS et les scénarios d'interaction avec l'IMS (IPv6). Le groupe v6ops de l'IETF a classé les différents scénarios de transition en fonction éléments disposant d'IPv6 (RFC 3574). Les scénarios concernant l'utilisation du GPRS sont les suivants :

- Le mobile dispose d'une double pile IPv4/IPv6. Malheureusement les adresses IPv4 sont une ressource rare pour l'ISP et le mobile ne peut avoir d'adresse IPv4 attribuée en permanence, il doit donc utiliser une adresse privée.
- Le mobile et son correspondant ont une adresse IPv6, mais les transmissions doivent utiliser un réseau de transport IPv4. Ce scénario sera utilisé dans les étapes initiales du déploiement d'IPv6 quand les réseaux d'interconnexion IPv4 ne disposant pas d'IPv6 seront encore présents.
- Le mobile et son correspondant ont une adresse IPv4, mais les transmissions utilisent un réseau IPv6. Ce scénario se produira lorsque les opérateurs migreront leurs réseaux d'interconnexion vers IPv6.
- Le mobile a seulement une adresse IPv6 et son correspondant seulement une adresse IPv4. Ce scénario se produira par exemple si un opérateur mobile supporte seulement le service IPv6 et que les terminaux mobiles cherchent à communiquer avec des équipements IPv4 de l'Internet.
- Le mobile a seulement une adresse IPv4 et son correspondant seulement une adresse IPv6. Ce scénario se produira au contraire si les opérateurs mobiles tardent trop à migrer leur service les contraignant ainsi à de coûteuses politiques d'adaptation.

Les scénarios liés à l'utilisation des services de l'IMS :

- Le mobile a un correspondant IPv4 et l'interconnexion se fait à travers l'IMS.
- Deux IMS IPv6 son interconnectés à travers un réseau de transport IPv4. Quand le déploiement de la version 5 sera complètement terminé, il restera encore quelques nœuds ou réseaux IPv4 qui devront utiliser des services de l'IMS.

VIII) Installation d'un équipement

Ce chapitre indique comment insérer un équipement terminal dans un réseau IPv6. Il prend comme hypothèse qu'il existe au moins un routeur émettant des messages de découverte des voisins. Pour une mise en œuvre plus complète du réseau, se reporter au chapitre Configuration des routeurs, qui décrit la configuration de différents types de routeurs.

Ce chapitre ne cherche pas à être exhaustif. En effet, aujourd'hui, la plupart des systèmes connaissent IPv6 ; il suffit en général de suivre les directives de configuration initiale pour activer IPv6 avec une adresse d'auto-configuration sans état. Nous donnons ici quelques exemples pour montrer comment faire des configurations moins automatiques.

- Solaris
- Windows
- Macintosh
- Linux
- BSD

1) Solaris

IPv6 fait partie intégrante du système depuis Solaris 8. La bibliothèque `libc` et la plupart des applications supportent IPv6, y compris les RPC et NFS.

Lors de la configuration initiale d'une machine, les menus d'installation système proposent de configurer IPv6, il suffit de suivre les questions. Pour les personnes n'ayant pas activé IPv6 à l'installation ou qui souhaitent modifier la configuration, ce chapitre décrit comment configurer «à la main».

Le premier point est de s'assurer que le service de noms connaît IPv6. Pour cela il faut valider la ligne «`ipnodes:`» dans le fichier `/etc/nsswitch.conf`. La syntaxe est la même que pour la ligne «`hosts:`» en IPv4. Ainsi pour rechercher l'adresse IPv6 associée à un nom d'abord dans le fichier de correspondance local (`/etc/inet/ipnodes`), puis par le DNS, la ligne est :

ipnodes: files dns

Ensuite il faut activer la configuration IPv6 des interfaces. Dans le cas de l'auto-configuration sans état, il suffit de créer un fichier de configuration vide de nom `/etc/hostname6.IFX` pour chaque interface de nom `IFX` (sauf pour l'interface `lo0`). Ainsi si la machine a une interface Ethernet `elxl0`, il faut exécuter :

> **touch /etc/hostname6.elxl0**

La configuration est suffisante, IPv6 sera disponible au prochain reboot.

Pour vérifier que IPv6 fonctionne correctement, on dispose des commandes `ping` et `traceroute` pour tester l'accessibilité d'une machine, et `netstat` et `rpcinfo` pour visualiser les tables de routage, et de connexions actives.

Par exemple pour tester la connectivité IPv6 :

```
> /usr/sbin/ping -s www6.ipv6.imag.fr
PING www6.ipv6.imag.fr: 56 data bytes
64 bytes from 2001:660:181:1::50: icmp_seq=0. time=46. ms
64 bytes from 2001:660:181:1::50: icmp_seq=1. time=45. ms
^C
----www6.ipv6.imag.fr PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip (ms) min/avg/max = 45/45/46
```

La commande suivante montre quelques services activés en IPv6: `rlogin`, `telnet`, `ftp`, impression, mail (`smtp`), partage de fichiers (`lockd`).

```
> netstat -af inet6
UDP: IPv6
Local Address               Remote Address          State    If
--------------------------- ----------------------- -------- ---
*.*                                                 Unbound
*.sunrpc                                            Idle
*.time                                              Idle
*.echo                                              Idle
*.daytime                                           Idle
.....
TCP: IPv6
```

```
Local Address      Remote Address    Swind Send-Q Rwind Recv-Q State If
------------------ ----------------- ----- ------ ----- ------ ----  --
*.*                *.*               0     0      24576 0      IDLE
*.sunrpc           *.*               0     0      65536 0      LISTEN
*.*                *.*               0     0      65536 0      IDLE
*.ftp              *.*               0     0      65536 0      LISTEN
*.echo             *.*               0     0      65536 0      LISTEN
*.printer          *.*               0     0      65536 0      LISTEN
*.smtp             *.*               0     0      65536 0      LISTEN
....
```

A) Configuration manuelle

Pour définir des adresses IPv6 de manière statique sur une interface `IFX` (ajout d'adresses ou configuration d'un routeur), il suffit de mettre les informations nécessaires dans le fichier de configuration `/etc/hostname6.IFX`. La syntaxe est celle des arguments de la commande ifconfig. L'adresse lien-local est toujours générée automatiquement et ne doit pas être positionnée de cette manière. Ainsi le fichier `/etc/hostname6.elxl0` suivant configure deux adresses IPv6 sur l'interface `elxl0` :

```
addif 3ffe:3ff:92:55::1000/64 up
addif 2001:6ff:43:55:A00:20ff:fe8e:F324/64 up
```

En Solaris, sur une machine qui n'est pas routeur, la génération d'adresses auto-configurées «sans état» est active par défaut. Pour les systèmes antérieurs à Solaris 10, il n'est pas possible de la désactiver simplement (il faut modifier des scripts système). En Solaris 10 c'est possible en ajoutant une directive dans le fichier `/etc/inet/ndpd.conf`. La ligne suivante désactive la génération automatique d'adresse sur toutes les interfaces :

```
ifdefault StatelessAddrConf off
```

B) Configuration d'un tunnel

Pour créer un tunnel configuré sur une machine Solaris, il faut créer un fichier de configuration `/etc/hostname6.ip.tun0` (0 pour le premier tunnel) indiquant les adresses source et destination IPv4 et IPv6. Le tunnel correspond à une interface de nom `ip.tun0`. Pour plus de détails voir le manuel (`man ifconfig`). L'exemple suivant crée un tunnel IPv6 dans IPv4, l'adresse de IPv4 de l'extrémité locale est `128.1.2.3`, celle de l'extrémité distante est `192.1.2.5`, le tunnel est entre les adresses IPv6 locale `2001:6ff::45` et distante `2001:6ff::46` :

```
tsrc 128.1.2.8 tdst 192.1.2.5 up
addif 2001:6ff::45 2001:6ff::46 up
```

Solaris permet aussi d'utiliser le mécanisme 6to4. Il est configuré en modifiant des variables dans le fichier `/etc/default/inetinit`.

C) Configuration de règles de sécurité

On peut contrôler les connexions en utilisant la librairie «tcpwrappers». De nombreuses commandes réseau (`sendmail`, `sshd`, les commandes lancées par `inetd`, ...) utilisent cette librairie pour vérifier si un accès distant est autorisé par un fichier de configuration (voir «`man -s 4 hosts_access`»). Voici un exemple de fichier `/etc/hosts.allow` qui limite les connections entrantes ssh à deux réseaux :

sshd: [2001:660:5301:2::/64] : allow

sshd: 192.0.2.32/255.255.255.224 : allow

sshd : ALL : deny

Le portage à IPv6 du filtrage des paquets («`firewall`» `ipf`) n'est pas disponible en Solaris 9 ni dans la version initiale de Solaris 10. Il devrait être fourni lors d'une mise à jour de Solaris 10.

2) Windows

Microsoft développe une pile IPv6 depuis plusieurs années. Pour Windows NT 4.0 et Windows 2000, une distribution expérimentale est fournie et documentée à l'adresse suivante :

http://www.microsoft.com/WINDOWS2000/technologies/communications/ipv6/default.asp

En Windows XP, le protocole IPv6 est supporté de base. Afin de l'installer, il suffit d'ouvrir une fenêtre DOS et d'exécuter la commande suivante :

```
C:\>ipv6 install
Installation en cours...
Opération réussie.
```

Contents

A) Configuration manuelle des interfaces physiques

Il est possible de configurer manuellement les interfaces sur Windows XP. Ceci peut être nécessaire dans le cas où l'on ne veut pas avoir une adresse d'auto-configuration (cas d'un serveur). La commande netsh est rès utile pour les configurations réseau. Voici un exemple de configuration d'interface :

```
C:\>netsh interface ipv6 set address "4" 2001:660:3001:4014::10
Ok.
```

L'adresse 2001:660:3001:4014::10 est configurée sur l'interface 4.

Pour supprimer cette adresse de l'interface 4, la commande est la suivante :

```
C:\>netsh interface ipv6 delete address "4" 2001:660:3001:4014::10
```

B) Configuration d'un tunnel

Voici un exemple de configuration manuelle de tunnel IPv6 dans IPv4 à l'aide de la commande netsh :

```
c:\netsh interface ipv6 add v6v4tunnel "Mon tunnel" 1.1.1.1 6.6.6.6
c:\netsh interface ipv6 add address "Mon tunnel" 2001:660:3001:4014::1
c:\netsh interface ipv6 add route ::/0 "Mon tunnel" fe80::208:74ff:fec8:16f9
```

Tout d'abord, le tunnel est créé avec l'adresse source `1.1.1.1` et l'adresse destination `6.6.6.6`. Une adresse IPv6 est configurée sur l'interface "`Mon tunnel`". Finalement, une route par défaut est ajoutée sur cette interface en spécifiant la route empruntée. Dans la plupart des cas, c'est l'adresse lien local du routeur auquel la station est raccordée. Configuration de mécanismes de tunnel automatique

Lorsqu'on se trouve sur un réseau IPv4, le mécanisme Tunnel Broker permet d'obtenir une adresse ou un préfixe IPv6 de manière automatique via une interface web. Une fois que l'utilisateur a fourni les renseignements nécessaires (système d'exploitation, adresse IP...), le tunnel est créé. Il suffit ensuite de configurer son extrémité de tunnel (ajout d'une route par défaut et de l'adresse d'extrémité). Voici un exemple :

```
c:\ipv6 rtu ::/0 2/::123.123.123.123
c:\ipv6 adu 2/2001:660:AAAA::1111:2222
```

Plusieurs logiciels Tunnel Broker sont disponibles pour Windows XP. Voici l'adresse de l'un d'eux : https://tb.ipv6.btexact.com/

Il existe aussi un client Windows XP pour TSP (**Tunnel Setup Protocol**) permettant de configurer de manière automatique l'extrémité du tunnel. Ce client peut être téléchargé sur http://www.freenet6.net.

Le mécanisme 6to4 est supporté par Windows XP. Il est installé par défaut lors de l'installation de la pile IPv6. Il est ainsi possible de disposer de connectivité IPv6 de manière automatique lorsqu'on se trouve sur un réseau IPv4.

```
Interface 3 : Pseudo-interface de tunneling 6to4
GUID {A995346E-9F3E-2EDB-47D1-9CC7BA01CD73}
n'utilise pas la découverte de voisin
n'utilise pas la découverte de routeur
préférence de routage 1
```

```
preferred global 2002:c131:9fa4::c131:9fa4, vie infinite
MTU de liaison 1280 (MTU de liaison réelle 65515)
limite de sauts actuelle 128
durée d'attente pour la communication 17500ms (base 30000ms)
intervalle de retransmission 1000ms
transmissions DAD 0
longueur par défaut du préfixe de site 48
```

Si l'interface 6to4 n'est pas activée (dans le cas où vous auriez déjà une adresse IPv6 globale par exemple), il suffit d'exécuter la commande suivante :

```
c:\netsh interface ipv6 6to4 set state enabled
```

C) Configuration de règles de sécurité (Firewall)

Lors de l'installation de la pile IPv6, un firewall spécifique pour IPv6 est installé par défaut ; il est lancé de manière automatique. Depuis le Service Pack 2 de Windows XP, le firewall IPv6 est intégré au firewall de Windows XP.

D) Configuration des applications

La plus part des serveurs IPv6 sont prévus pour des systèmes d'exploitation type Unix (Linux, BSD). Cependant de nombreux clients sont disponibles pour Windows XP. Ils ne nécessitent pas dans la plupart des cas une configuration particulière. On peut citer Internet Explorer, Mozilla pour le web, Mozilla/Thunderbird pour le mail, SmartFTP pour FTP.

3) Macintosh

Le support d'IPv6 dans MacOS est standard en MacOS X (10.3).

La commande `sysctl -a` permet de vérifier le support IPv6 (Net.inet6). La définition des paramètres noyau liés à inet6 se fera alors comme un système BSD classique (cf. `man sysctl`).

Afin de profiter du support IPv6, une collection d'outils de base peut être récupérée à l'adresse ftp://morth.nu/darwin/.

Une version modifiée du paquetage KAME (version stable 20000425) pourra également être téléchargée afin de recompiler ses propres outils (tcpdump,...). De plus, un certain nombre d'utilitaires ont été portés pour IPv6 au sein du projet GNU-Darwin

4) Linux

De nombreuses distributions de Linux existent. Debian, RedHat, Mandrake, Suse en sont quelques unes parmi les plus connues. Une distribution Linux est constituée du noyau Linux proprement dit, et d'un ensemble de programmes (essentiellement d'origine GNU ou BSD) utilisant des librairies. D'une manière générale, pour qu'une distribution Linux fonctionne en IPv6, il faut qu'elle intègre un noyau, des librairies, des scripts de configuration et des applications supportant IPv6. Bien que dérivant de noyaux et d'outils de même origine, chaque distribution a ses particularités. Le programme d'installation est différent, et chaque société ou organisme maintenant cette distribution fait le choix d'intégrer -ou non- des programmes différents en fonction du public visé. Il est donc impossible, dans ce chapitre, de particulariser pour chaque distribution. On donne ici des remarques générales, et on développera l'exemple de RedHat/FedoraCore. Pour les autres distributions, la documentation (ou une recherche sur le Web) permet de trouver les adaptations nécessaires.

La souche IPv6 est intégrée officiellement au noyau depuis les versions 2.2, mais ces noyaux étaient incomplets et non conformes aux RFC. Les noyaux 2.4 sont plus corrects, mais eux aussi présentent quelques lacunes. Les noyaux 2.6 sont donc préférables ; ils intègrent un partie des développements du projet japonais USAGI, en particulier la sécurité (IPsec). Il faut aussi un noyau compilé avec l'option IPv6 (dans le noyau ou en module). Ce type de noyau est en général disponible dans tous les distributions (au moins comme paquetage optionnel).

Les applications, quand à elles, doivent utiliser une librairie C supportant IPv6. La GNU Libc 2 intègre correctement le support IPv6 à partir de la version 2.2 de la Glibc. Aussi, il est important d'utiliser une distribution Linux qui réponde à ces critères.

C'est le cas des distributions récentes, par exemple Debian (version Woody ou supérieure), RedHat ≥ = 7.1, Mandrake ≥ = 8.0. Cette liste n'est bien entendu pas exhaustive.

Contents

A) Linux RedHat et FedoraCore

Linux RedHat et FedoraCore proposent IPv6 en standard depuis la version RedHat 7.1. La bibliothèque `libc` et la plupart des applications supportent IPv6 (sauf RPC et NFS). Pour le cas le plus simple (machine utilisant l'auto-configuration sans état), il suffit de valider à l'installation. On peut modifier l'activation de IPv6 en définissant à *yes* ou à *no* la variable `NETWORKING_IPV6` dans le fichier `/etc/sysconfig/network`. Notons que dans les dernières versions de FedoraCore, la génération d'adresses auto-configurées est

active par défaut ; toutefois les scripts de configuration IPv6 ne seront pas appelés si la variable `NETWORKING_IPV6` n'est pas validée, et la configuration risque donc d'être incomplète.

Pour vérifier que IPv6 fonctionne correctement, on dispose des commandes `ping6` ou `traceroute6` pour tester l'accessibilité d'une machine, et `netstat` ou `ip` pour visualiser les tables de routage, et de connexions actives.

B) Configuration manuelle

Pour configurer des adresses IPv6 de manière statique sur une interface de nom `ethX`, il suffit de mettre les informations nécessaires dans le fichier de configuration de l'interface `/etc/sysconfig/network-scripts/ifcfg-ethX`. Les variables à définir sont `IPV6INIT`, `IPV6_AUTOCONF`, `IPV6ADDR`. Toutes les variables utiles sont documentées dans le fichier `/etc/sysconfig/network-scripts/init.ipv6`. Par exemple voici comment définir une adresse statique sur une interface :

```
IPV6INIT=YES
IPV6_AUTOCONF=no
IPV6ADDR=2001:6ff:10:1::1000/64
```

Il est aussi possible d'ajouter une adresse IPv6 explicitement grace à la commande `ifconfig`.

```
ifconfig eth0 inet6 add 2001:6ff:10:1::1000/64
```

L'ajout de la route par défaut correspondante se fait comment en IPv4 à l'aide de la commande `route`.

```
route -A inet6 add default gw 2001:6ff:10:1::ffff
```

Une autre solution consiste à utiliser la commande `ip` du package iproute2. Iproute2 est une collection d'utilitaires permettant de contrôler divers aspects réseau sous Linux. Iproute2 et sa commande `ip` visent à remplacer les commandes `ifconfig` et `route` jugées obsolètes.

L'ajout d'une adresse IP en utilisant la commande `ip` se fait de la manière suivante :

```
ip -6 addr add 2001:6ff:10:1::1000/64 dev eth0
```

Cette commande est strictement équivalente à la commande `ifconfig` utilisée précédemment. Tout comme pour ifconfig, il est aussi possible d'utiliser la commande `ip` pour remplacer la commande `route`.

```
ip -6 route add default via 2001:6ff:10:1::ffff
```

C) Configuration d'un tunnel

Pour créer un tunnel configuré, il faut configurer une interface de nom `sitX` (X≥1). Le fichier suivant `/etc/sysconfig/network-scripts/ifcfg-sit1` déclare un tunnel IPv6 dans IPv4, l'adresse de IPv4 de l'extrémité distante étant `192.1.2.5`, sans adresses IPv6 :

```
DEVICE="sit1"
BOOTPROTO="none"
ONBOOT="yes"
IPV6INIT="yes"
IPV6TUNNELIPv4="192.1.2.5"
```

Comme le tunnel n'a pas d'adresse IPv6, il faut configurer le routage pour l'utiliser. Voici un exemple de fichier `/etc/sysconfig/static-routes-ipv6` qui définit une route statique envoyant tout le trafic IPv6 unicast dans le tunnel :

```
sit1 2001::/3
```

D) Configuration de règles de sécurité

Linux RedHat et FedoraCore proposent le filtrage de paquets «iptables» en IPv4 comme en IPv6 (voir http://www.netfilter.org). Le paquetage (rpm) à installer est iptables-ipv6, et la commande principale est ip6tables. La table de filtrage installée est rangée dans le fichier /etc/sysconfig/ip6tables. Par exemple, la table suivante bloque la commande «ping ::1» :

```
> cat /etc/sysconfig/ip6tables
*filter
:INPUT ACCEPT [13:1352]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [12:1248]
-A INPUT -s ::1/128 -d ::/0 -p ipv6-icmp -j DROP
COMMIT
# Completed on Sun Jan 30 17:03:16 2005
```

Ce fichier a été créé par les commandes :

```
> ip6tables -A INPUT -s ::1 -p ipv6-icmp -j DROP
> ip6tables-save > /etc/sysconfig/ip6tables
```

Pour plus de détails, se référer à la documentation de Iptables (en remplaçant les adresses IPv4 par des adresses IPv6 !).

Une autre méthode pour contrôler les connexions est d'utiliser la librairie «tcpwrappers». De nombreuses commandes réseau (sendmail, sshd, les commandes lancées par xinetd, ...) utilisent cette librairie pour vérifier si un accès distant est autorisé par un fichier de configuration (voir man 5 hosts_access). Voici un exemple de fichier de configuration /etc/hosts.allow qui limite les connections entrantes ssh à deux réseaux :

```
sshd: [2001:660:5301:2::]/64 : allow
sshd: 192.0.2.32/255.255.255.224 : allow
sshd : ALL : deny
```

5) BSD

Il existe de nombreux systèmes dérivés de BSD : BSD/OS, FreeBSD, NetBSD, OpenBSD, MAC OS X,... IPv6 est disponible sur ces systèmes depuis très longtemps, plusieurs implémentations ont existé. Aujourd'hui la plupart de ces systèmes proposent IPv6 en standard, en utilisant un code dérivé d'une même souche (KAME). On se concentrera ici sur FreeBSD et NetBSD, mais les mises en ?uvre sur les autres systèmes sont proches.

Contents

A) FreeBSD

FreeBSD propose IPv6 en standard depuis la version 4.0. La bibliothèque `libc` et la plupart des applications supportent IPv6 (RPC et NFS seulement à partir de FreeBSD 5). Dans le cas le plus simple (machine utilisant l'auto-configuration sans état), les menus d'installation système proposent de configurer IPv6, il suffit de répondre à la question de configuration d'une interface en IPv6. Si on n'a pas activé IPv6 à l'installation, on peut rappeler le programme de configuration `/stand/sysinstall` pour reconfigurer les interfaces. On peut aussi configurer «à la main» en éditant le fichier `/etc/rc.conf`.

Le fichier `/etc/rc.conf` sert à définir la plupart des variables de configuration d'une machine FreeBSD. Les valeurs par défaut (et les commentaires) sont dans le fichier de référence `/etc/default/rc.conf` (à ne pas modifier).

Pour activer manuellement IPv6 dans le cas le plus simple (auto-configuration sans état), il suffit d'ajouter dans le fichier `/etc/rc.conf` la ligne :

```
ipv6_enable=YES
```

IPv6 sera disponible au prochain reboot.

Pour vérifier que IPv6 fonctionne correctement, on dispose des commandes `ping6` et `traceroute6` pour tester l'accessibilité d'une machine, et netstat pour visualiser les tables de routage, et de connexions actives.

Par exemple pour tester la connectivité IPv6 :

```
> /usr/sbin/traceroute6 www6.ipv6.imag.fr
traceroute6 to www6.ipv6.imag.fr (2001:660:181:1::50) from
2001:660:282:5:2b0:d0ff:fe3b:e565, 30   hops max, 12 byte packets
1 2001:660:282:5:200:c0ff:fee4:caa0 22.234 ms 0.993 ms 0.919 ms
2 pallas.ipv6.rennes.enst-bretagne.fr 3.81 ms * 3.684 ms
3 horace.ipv6.rennes.enst-bretagne.fr 7.93 ms * 15.773 ms
4 2001:660:80:4002::1 14.876 ms * 14.941 ms
5 2001:660:80:4004::2 22.877 ms * 22.835 ms
6 luna-v6.ipv6.imag.fr 50.509 ms 46.267 ms 46.148 ms
```

La commande suivante montre les interfaces activées en IPv6. Il existe un tunnel IPv4 dans IPv6 `gif0`, et l'interface `xl0` a deux adresses IPv6 globales :

```
> netstat -inf inet6
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
xl0 1500 <Link#1> 00:b0:d0:3b:e5:65 82186 0 74502 0 0
xl0 1500 193.52.74 193.52.74.217 58006 - 72342 - -
xl0 1500 fe80:1::2b0 fe80:1::2b0:d0ff: 70 - 2131 - -
xl0 1500 2001:660:28 2001:660:282:5:2b 1388 - 0 - -
xl0 1500 3ffe:305:10 3ffe:305:1002:5:2 467 - 0 - -
lp0* 1500 <Link#2> 0 0 0 0 0
gif0 1280 <Link#3> 279 0 388 0 0
gif0 1280 fe80:3::2b0 fe80:3::2b0:d0ff: 0 - 0 - -
gif0 1280 192.108.119.1 192.108.119.195 279 - 386 - -
...
```

a) Configuration manuelle

Pour configurer des adresses IPv6 de manière statique sur une interface de nom `IFX`, il suffit de mettre dans le fichier de configuration `/etc/rc.conf` les informations nécessaires. Les variables à définir ont pour nom `ipv6_ifconfig_IFX` (une seule adresse) ou `ipv6_ifconfig_IFX_aliasY` (`Y` entier de 0 à N-1 si `IFX` a N adresses IPv6). La syntaxe est celle des arguments de la commande `ifconfig`. L'adresse lien-local est toujours générée automatiquement et ne doit pas être positionnée de cette manière. Ainsi les lignes suivantes ajoutées dans le fichier `/etc/rc.conf` configurent deux adresses IPv6 sur l'interface `fxp0` :

```
ipv6_ifconfig_fxp0_alias0="3ffe:3ff:92:55:a00:20ff:fe8e:f324 prefixlen 64"
ipv6_ifconfig_fxp0_alias1="2001:6ff:43:55:a00:20ff:fe8e:f324 prefixlen 64"
```

b) Configuration d'un tunnel

Pour créer un tunnel configuré, il faut configurer une interface de type gif. Les lignes suivantes ajoutées dans le fichier `/etc/rc.conf` déclarent un tunnel IPv6 dans IPv4, l'adresse de IPv4 de l'extrémité locale étant `128.1.2.3`, celle de l'extrémité distante `192.1.2.5`, le tunnel est entre les adresses IPv6 locale `2001:6ff::45` et distante `2001:6ff::46` :

```
gif_interfaces="gif0"
gif_config_gif0="128.1.2.3 192.1.2.5"
ipv6_ifconfig_gif0="2001:6ff:45 2001:6ff:46"
```

c) Configuration de règles de sécurité

FreeBSD propose le filtrage de paquets en IPv4 comme en IPv6. L'activation est contrôlée par les variables du fichier `/etc/rc.conf` `ipv6_firewall_enable` (`YES` ou `NO`) et `ipv6_firewall_type`. Cette seconde variable peut valoir `open` (pas de filtre installé), `client` (utiliser un jeu de filtre standard pour une machine simple), `simple` (utiliser un jeu de filtre standard pour un réseau), `closed` (tout interdire sauf le trafic via l'interface `lo0`), ou être le nom d'un fichier de règles (voir `man ip6fw`). Par exemple, on peut bloquer la commande «`ping6 ::1`» en positionnant «`ipv6_firewall_type=/etc/my_ip6fw_rules`» et en créant le fichier `/etc/my_ip6fw_rules` :

```
add 100 deny ipv6-icmp from ::1 to any
add 65000 pass all from any to any
```

Une autre méthode pour contrôler les connexions est d'utiliser la librairie «tcpwrappers». De nombreuses commandes réseau (`sendmail`, `sshd`, les commandes lancées par `inetd`, ...) utilisent cette librairie pour vérifier si un accès distant est autorisé par un fichier de configuration (voir `man 5 hosts_access`). Voici un exemple de fichier de configuration `/etc/hosts.allow` qui limite les connections entrantes ssh à deux réseaux :

```
sshd: [2001:660:5301:2::]/64 : allow
sshd: 192.0.2.32/255.255.255.224 : allow
sshd : ALL : deny
```

B) NetBSD

NetBSD propose IPv6 en standard depuis la version 1.5. La plupart des applications sont portées pour IPv6, y compris RPC et NFS.

Si on n'a pas activé IPv6 à l'installation, on peut le configurer «à la main» en éditant le fichier `/etc/rc.conf`.

Le fichier `/etc/rc.conf` définit la plupart des variables de configuration. Les valeurs par défaut (et les commentaires) sont dans le fichier `/etc/default/rc.conf` (à ne pas modifier).

Pour activer manuellement IPv6 sur une machine NetBSD dans le cas le plus simple (auto-configuration sans état), il suffit d'ajouter dans le fichier `/etc/rc.conf` la ligne :

```
ip6mode=autohost
```

IPv6 sera disponible au prochain reboot.

a) Configuration manuelle

Pour configurer des adresses IPv6 de manière statique sur une interface de nom IFX, il suffit de mettre dans le fichier de configuration `/etc/ifconfig.IFX` les informations nécessaires, selon la syntaxe des arguments de la commande ifconfig. L'adresse lien-local est toujours générée automatiquement et ne doit pas être positionnée de cette manière. Par exemple les deux dernières lignes du fichier suivant définissent 2 adresses sur l'interface `epic0` :

```
> cat /etc/ifconfig.epic0
up media autoselect
132.227.10.10 netmask 0xfffffe0
inet6 2001:660:282:1:260:97ff:fe41:6143 prefixlen 64
inet6 3ffe:304:282:1:260:97ff:fe41:6143 prefixlen 64 alias
```

Par défaut une machine NetBSD ne remplit pas la fonction de routeur. La valeur de la variable ip6mode dans le fichier `/etc/rc.conf` permet de choisir le mode de fonctionnement :

- routeur relayant les paquets (`ip6mode=router`),
- hôte s'autoconfigurant (`ip6mode=autohost`)
- hôte avec adresses IPv6 statiques (`ip6mode=host`).

b) Configuration d'un tunnel

La configuration d'un tunnel passe par la configuration du pseudo-device gif qui est utilisé pour la configuration de tunnel IPv6 dans IPv4 et IPv6 dans IPv6.

De la même manière que nous avons configuré notre interface Ethernet, nous configurons notre premier tunnel créant le fichier `/etc/ifconfig.gif0`.

Pour un tunnel IPv6 dans IPv6 on aurait :

```
> cat /etc/ifconfig.gif0
inet6 tunnel 2001:660:10c:3d:280:c8ff:fe46:308f 2001:660:80:4000::2
inet6 3ffe:304:124:ff01::1 3ffe:304:124:ff01::2
```

et pour IPv6 dans IPv4 :

```
> cat /etc/ifconfig.gif0
tunnel 132.227.72.30 129.88.26.8
inet6 3ffe:304:124:ff01::1 3ffe:304:124:ff01::2
```

La première ligne établit le tunnel ; elle est équivalente à la commande suivante :

```
> ifconfig gif0 inet6 tunnel 132.227.72.30 129.88.26.8
```

c) Configuration de règles de sécurité

NetBSD propose le filtrage de paquets en IPv4 comme en IPv6. L'activation est contrôlée par la variable du fichier `/etc/rc.conf ipv6_filter` (`YES` ou `NO`). Le fichier de règles est `/etc/ipf6.conf` (voir `man ipf6.conf`). Par exemple, on peut bloquer la commande «`ping6 ::1`» avec le fichier suivant :

```
> cat /etc/ipf6.conf
block in quick from ::1 to any
pass in all
```

Une autre méthode pour contrôler les connexions est d'utiliser la librairie «tcpwrappers». De nombreuses commandes réseau (`sendmail`, `sshd`, les commandes lancées par inetd, ...) utilisent cette librairie pour vérifier si un accès distant est autorisé par un fichier de configuration (voir `man 5 hosts_access`). Voici un exemple de fichier de configuration `/etc/hosts.allow` qui limite les connections entrantes ssh à deux réseaux :

```
sshd: [2001:660:5301:2::/64] : allow
sshd: 192.0.2.32/255.255.255.224 : allow
sshd : ALL : deny
```

IX) Routage

Ce chapitre a pour objectif de montrer l'impact d'IPv6 sur les protocoles de routage. Il ne sera pas détaillé ici le fonctionnement de tel ou tel protocole, mais plutôt les changements qui ont été nécessaires afin de prendre en compte la technologie IPv6 dans les protocoles de routage existants pour IPv4. Ces changements sont essentiellement liés à la prise en compte du nouveau format de l'adresse IPv6 (c.f. Adressage), ainsi qu'à l'ajout d'une nouvelle table de routage dédiée à IPv6.

Les différents types de routage sont passés en revue: routage statique, routage interne et routage externe. A l'issue du chapitre, on constatera que IPv6 est maintenant bien intégré dans ces protocoles et que cette évolution a eut un impact très faible pour l'utilisateur final.

1) Routage statique

Le routage statique est le même en IPv6 qu'en IPv4, avec bien sûr le préfixe et le next-hop qui sont IPv6. L'exemple suivant montre comment configurer une route statique par défaut sur un Cisco en IPv4 et en IPv6:

```
!
ip route 0.0.0.0 0.0.0.0 10.193.4.1
ipv6 route ::/0 2001:688:1F80:12::2
!
```

2) Protocoles de routage

Les algorithmes de routage n'ont pas changé avec IPv6. Les travaux en cours consistent principalement à les adapter au nouveau format de l'adresse IP. Ces protocoles de routage profitent des propriétés maintenant incluses dans la nouvelle version du protocole IPv6 comme l'authentification ou le multicast.

Une conséquence de l'apparition du routage IPv6 est que les équipements doivent alors prendre en compte les deux piles de protocoles, IPv4 et IPv6. Cela doit être pris en considération lors de l'activation des protocoles de routage. En particulier, il faut prêter attention à la congruence des topologies IPv4 et IPv6.

Comme dans IPv4, il faut faire la distinction entre deux grandes familles de protocoles de routage : les protocoles de routage internes (IGP : *Interior Gateway Protocols*) et externes (EGP : *Exterior Gateway Protocols*). C'est la notion de système autonome qui permet de faire la différence en définissant la portée des échanges d'informations de routage effectuée par ces protocoles de routage. Ainsi, la propagation des préfixes internes à un AS se fait par un IGP, alors que les annonces de préfixes entre AS se fait par un EGP.

Pour connecter un site à l'Internet, il faut donc mettre œuvre des protocoles de routage internes et des protocoles de routage externes. Ce chapitre traite les trois protocoles IGP suivants: RIPng (équivalent de RIPv2 pour IPv4), ISIS et OSPFv3 (équivalent de OSPFv2 pour IPv4), ainsi que du protocole de routage externe BGP.

3) Routage interne

Les protocoles de routage internes permettent une configuration automatique des tables de routage des routeurs à l'intérieur d'un même système autonome. Les routeurs déterminent le plus court chemin pour atteindre un réseau distant. Les protocoles de routage internes nécessitent une configuration minimale du routeur notamment en ce qui concerne les annonces de routes initiées par ce routeur (ex. réseaux directement accessibles par une interface du routeur, annonces statiques ...).

Deux types de protocole de routage interne existent: les protocoles à état de lien ("link state" en anglais) et les protocole à vecteur de distance ("distance vector" en anglais). Les premiers calculent le chemin le plus court en comptant le nombre de sauts pour atteindre le préfixe de destination, tandis que les seconds attribuent un coût à chaque lien en fonction de divers paramètres (type du lien...).

A) RIPng

Les algorithmes appelés "distance vector", sont utilisés par le protocole de routage RIPv2 (RFC 2453). Ils sont basés sur l'algorithme de Bellman-Ford et figurent parmi les premiers algorithmes de routage interne utilisés dans l'Internet.

Les routeurs diffusent leurs tables de routage sur les liens auxquels ils sont connectés. Les autres routeurs modifient une route dans leur table si la métrique reçue (dans ce cas le nombre de routeurs à traverser pour atteindre une destination) est plus petite que celle déjà stockée dans la table. Si une annonce de route n'est pas présente dans la table, le routeur la rajoute. Ces modifications sont à leur tour diffusées sur les autres réseaux auxquels sont connectés les routeurs. Elles se propagent donc sur l'ensemble du réseau à l'intérieur du système autonome. On montre que cet algorithme converge et qu'en condition stable, aucune boucle n'est créée sur le réseau (c'est-à-dire qu'un paquet ne sera pas transmis indéfiniment de routeur en routeur sans jamais pouvoir atteindre sa destination).

Les tables sont émises périodiquement. Si un routeur tombe en panne ou si le lien est coupé, les autres routeurs ne recevant plus l'information suppriment l'entrée correspondante de leur table de routage.

RIPng est le premier protocole de routage dynamique proposé pour IPv6 (RFC 2080) RIPng est une simple extension à IPv6 du protocole RIPv2 d'IPv4. Il en hérite les mêmes limitations d'utilisation (maximum de 15 sauts par exemple).

a) Fonctionnement

Le format générique des paquets RIPng, donné figure Format d'un paquet RIPng, est très proche de celui des paquets du protocole RIPv2. Seule la fonction d'authentification a disparu. Elle est en effet inutile car RIPng peut s'appuyer sur les mécanismes de sécurité IPSec disponibles en IPv6. La structure d'une information de routage est donnée par la figure Format d'un champ "Entrée" d'un paquet RIPng.. Le nombre d'informations de routage contenues dans un paquet RIPng dépend directement du MTU des liens utilisés. Si RIPng doit annoncer un grand nombre de réseaux, plusieurs paquets RIPng seront nécessaires.

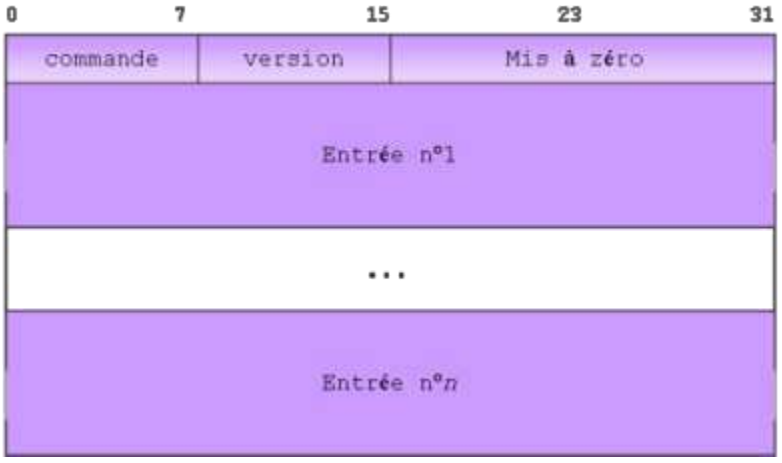


Figure 9-1 *Format d'un paquet RIPng*

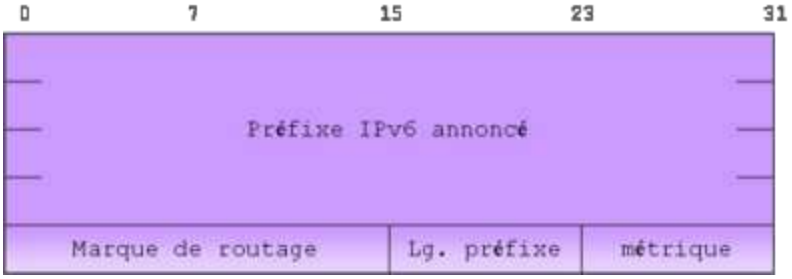


Figure 9-2 *Format d'un champ "Entrée" d'un paquet RIPng.*

Le format des messages RIPng est le suivant :

- Le champ version est aujourd'hui défini à la valeur 1.
- Le champ commande contient :
  - 1 : pour une requête de table de routage.
  - 2 : pour une réponse à une requête ou une émission périodique.
- Les champs préfixe et longueur de préfixe décrivent les réseaux annoncés.
- Le champ métrique comptabilise le nombre de routeurs traversés.
  Pour résoudre les problèmes de convergence ("comptage jusqu'à l'infini"), la valeur maximale de métrique ("infini") est 16.
  Cette valeur est largement suffisante pour le domaine d'application du protocole (le système autonome).
  Le champ métrique peut donc prendre toutes les valeurs comprises entre 0 et 16.
  Si le champ métrique d'une entrée vaut `0xFF`, (cf. figure Format d'un champ "Entrée" d'un paquet RIPng.) le champ préfixe de cette entrée donne l'adresse d'un routeur (prochain saut). Une telle entrée indique que les destinations des entrées suivantes sont accessibles par le routeur explicitement indiqué, et non par celui envoyant le paquet RIPng (utilisation en mode "proxy").
  Dans ce cas, les champs "marque de routage" et "longueur du préfixe" sont mis à 0.

Dans RIPv2, cette possibilité était indiquée dans chaque information de routage. Vue la longueur des adresses IPv6, dans RIPng l'indication du prochain saut est valable pour toutes les routes qui suivent

jusqu'à la fin du paquet ou jusqu'à la prochaine entrée de ce type. Ainsi, bien que les adresses soient quatre fois plus grandes qu'en IPv4, la taille des informations de routage est la même que dans RIPv2.

Les paquets RIPng sont émis vers l'adresse de multicast *all-rip-router* `FF02::9` et encapsulés dans un paquet UDP avec le numéro de port 521.

b) Exemples

Voici la trace des paquets RIPng échangés sur un Ethernet reliant 3 routeurs :

```
IPv6
Version : 6 Classe : 00 Label : 00000
Longueur : 252 octets (0x00fc) Protocole : 17 (0x11, UDP)
Nombre de sauts : 255 (0xff)
Source : fe80::a00:20ff:fe0c:7a34
Desti. : ff02::09 (ALL-RIP-ON-LINK)
UDP Src. port: 521 (0x0209, ripng) Dst. port: 521 (0x0209, ripng)
Lg : 252 octets (0x00fc) Checksum : 0x249d
RIP Commande : 2 (Réponse/émission) Version : 1
Entrées :
3ffe:302:12:3::/80(1) 3ffe:302:12:2::/64(1) 3ffe:302:11:1::/64(16)
3ffe:302:12:4::/64(16) 3ffe:301:2::/48(16) 3ffe:301:3::/48(16)
3ffe:301:5::/48(16) 3ffe:302:21::/48(16) 3ffe:306:1051::/48(16)
3ffe:303::/32(16) 3ffe:305::/32(16) ::/0(16)

0000: 60 00 00 00 00 fc 11 ff fe 80 00 00 00 00 00 00
0010: 0a 00 20 ff fe 0c 7a 34 ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 09|02 09 02 09 00 fc 24 9d|
0030: 02 01 00 00|3f fe 03 02 00 12 00 03 00 00 00 00
0040: 00 00 00 00 00 00 50 01|3f fe 03 02 00 12 00 02
0050: 00 00 00 00 00 00 00 00 00 00 40 01|3f fe 03 02
0060: 00 11 00 01 00 00 00 00 00 00 00 00 00 00 40 10|
0070: 3f fe 03 02 00 12 00 04 00 00 00 00 00 00 00 00
0080: 00 00 40 10|3f fe 03 01 00 02 00 00 00 00 00 00
0090: 00 00 00 00 00 00 30 10|3f fe 03 01 00 03 00 00
00a0: 00 00 00 00 00 00 00 00 00 00 30 10|3f fe 03 01
00b0: 00 05 00 00 00 00 00 00 00 00 00 00 00 00 30 10|
00c0: 3f fe 03 02 00 21 00 00 00 00 00 00 00 00 00 00
00d0: 00 00 30 10|3f fe 03 06 10 51 00 00 00 00 00 00
00e0: 00 00 00 00 00 00 30 10|3f fe 03 03 00 00 00 00
00f0: 00 00 00 00 00 00 00 00 00 00 20 10|3f fe 03 05
0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 10|
0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0120: 00 00 00 10



Source : fe80::a00:20ff:fe75:24ea Desti. : ff02::09 (ALL-RIP-ON-LINK)
UDP Src. port: 521 (0x0209, ripng) Dst. port: 521 (0x0209, ripng)
RIP Commande : 2 (Réponse/émission) Version : 1
Entrées :
3ffe:302:12:3::/80(16) 3ffe:302:12:2::/64(1) 3ffe:302:11:1::/64(1)
3ffe:302:12:4::/64(16) 3ffe:301:2::/48(2) 3ffe:301:3::/48(2)
3ffe:301:5::/48(2) 3ffe:302:21::/48(2) 3ffe:306:1051::/48(2)
3ffe:303::/32(2) 3ffe:305::/32(2) ::/0(2)
```

```
Source : fe80::200:c0ff:fe68:ece7 Desti. : ff02::09 (ALL-RIP-ON-LINK)
UDP Src. port: 521 (0x0209, ripng) Dst. port: 521 (0x0209, ripng)
RIP Commande : 2 (Réponse/émission) Version : 1
Entrées :
3ffe:302:12:3::/80(16) 3ffe:302:12:4::/64(1) 3ffe:302:12:2::/64(1)
3ffe:302:11:1::/64(16) 3ffe:301:2::/48(16) 3ffe:301:3::/48(16)
3ffe:301:5::/48(16) 3ffe:302:21::/48(16) 3ffe:306:1051::/48(16)
3ffe:303::/32(16) 3ffe:305::/32(16) ::/0(16)
```

Et ainsi de suite toutes les 30 secondes environ.

On remarquera les adresses source (lien-local) et destination (multicast portée lien), la métrique 16 associée à un préfixe si le routeur n'est pas le meilleur sur le câble (méthode dite *Poisonning Reverse*), et l'utilisation du préfixe `::/0` pour annoncer une route par défaut.

B) ISIS

IS-IS (*Intermediate System to Intermediate System*) est un protocole de routage interne à état de lien. Il a été standardisé par l'ISO (ISO 10589). C'est un protocole de niveau 3 (contrairement à OSPF et RIP qui sont de niveau 4) qui s'appuie sur une couche 2 de type Ethernet 802.2. Cet élément mérite d'être signalé car cela rend ce protocole indépendant d'IP, que ce soit IPv4 ou IPv6. Ce protocole travaille sur deux niveaux de hiérarchie : les aires (niveau 1) et le backbone (niveau 2).

Un routeur IS-IS peut être soit :

- level-1 (routage intra aire),
- level-2 (routage inter aire),
- level-1-2 (routage intra et inter aire).

Un routeur de niveau 1 n'a de voisins que dans son aire, alors qu'un routeur de niveau 2 peut avoir des voisins dans une autre aire. Il n'y a pas d'aire de backbone (contrairement à OSPF). Le backbone est constitué de la réunion de tous les routeurs de level-2. Sur un réseau de type LAN, il y a élection d'un routeur désigné (DIS) qui a la charge de produire les annonces.

Afin de construire sa topologie, IS-IS utilise 3 types de messages:

- les messages HELLO permettant de construire les adjacences;
- les messages LSP (*Link State Protocol*) permettant d'échanger les informations sur l'état des liens;
- les messages SNP (*Sequence Number Packet*) permettant de confirmer la topologie.

Pour élaborer ces messages, IS-IS se base sur l'utilisation d'éléments d'informations indépendants appelés TLV (Type, Longueur, Valeur). Le message est ainsi constitué d'un en-tête suivi d'une liste de TLV. Chaque TLV véhicule une information propre, et est donc standardisée. L'exemple ci-dessous montre une TLV Protocoles Supportés faisant partie d'un message HELLO, informant les voisins des protocoles supportés par l'émetteur du paquet :

```
0x81 0x02 0xCC 0x8E
```

Le premier octet donne le type de la TLV. Il s'agit ici du type `0x81`, c'est-à-dire Protocoles supportés. Le second octet donne la longueur en octets de la TLV : ici les deux octets qui suivent. Les autres octets composent la valeur de la TLV. Ici nous avons deux octets indiquant des numéros de protocoles supportés (NLPID : *Network Layer Protocol IDentifier*): `0xCC` pour IPv4 et `0x8E` pour IPv6.

a) Le support d'IPv6

Les TLV permettent une évolution souple du protocole IS-IS. Cela s'est vérifié lorsque le support d'IPv4 a été ajouté (IS-IS est alors devenu integrated IS-IS ou i/IS-IS): en 1990, le (RFC 1195) a défini 6 nouvelles TLV afin de prendre en compte le routage d'IPv4.

De même, deux nouvelles TLV ont été ajoutées pour le support d'IPv6 . Elles sont définies dans le draft IETF See [Hopps-id](début 2003), ainsi que le numéro de protocole NLPID propre à IPv6 : il vaut 142 (soit `0x8E` en héxadécimal). C'est cette valeur que l'on voit dans l'exemple ci-dessus (exemple de TLV).

La capture ci-dessous est celle d'un paquet IS-IS HELLO provenant d'un routeur qui annonce sa capacité à traiter les protocoles IP et IPv6 (dans la TLV vue précédemment) ainsi que l'adresse IPv6 d'une de ses interfaces (dans une TLV de type 232).

```
Frame 12 (97 bytes on wire, 97 bytes captured)
IEEE 802.3 Ethernet
Logical-Link Control
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
Intra Domain Routing Protocol Discriminator: ISIS (0x83)
PDU Header Length : 27
Version (==1) : 1
System ID Length : 0
PDU Type : L2 HELLO (R:000)
Version2 (==1) : 1
Reserved (==0) : 0
Max.AREAs: (0==3) : 0
ISIS HELLO
Circuit type : Level 2 only, reserved(0x00 == 0)
System-ID {Sender of PDU} : 1921.6815.0001
Holding timer : 9
PDU length : 80
Priority : 64, reserved(0x00 == 0)
System-ID {Designated IS} : 1921.6815.0001.04
IS Neighbor(s) (6)
Protocols Supported (2)
NLPID(s): IP (0xcc), IPv6 (0x8e)
IP Interface address(es) (4)
IPv6 Interface address(es) (16)
IPv6 interface address: fe80::290:69ff:febf:b03e
Area address(es) (4)
Restart Signaling (3)
Multi Topology (4)

0000 01 80 c2 00 00 15 00 90 69 bf b0 3e 00 53 fe fe ........i..>.S..
0010 03 83 1b 01 00 10 01 00 00 02 19 21 68 15 00 01 ...........!h...
0020 00 09 00 50 40 19 21 68 15 00 01 04 06 06 00 05 ...P@.!h........
0030 dd fb 20 06 81 02 cc 8e 84 04 c0 a8 0a 03 e8 10 .. ..............
0040 fe 80 00 00 00 00 00 00 02 90 69 ff fe bf b0 3e ..........i....>
0050 01 04 03 49 00 01 d3 03 00 00 00 e5 04 00 00 00 ...I............
0060 02 .
```

Le routeur est double pile car il annonce le support de IPv4 et IPv6. Son identificateur (System-ID: `1921.6815.0001.04`) est dérivé de son adresse de loopback IPv4 (`192.168.150.001`), c'est une pratique courante. Dans le cas où un routeur ne serait que IPv6, il faudrait alors convenir d'une autre méthode pour attribuer l'identificateur de système au routeur IS-IS (l'adresse MAC par exemple).

b) Le problème de la mono topologie

Soit le réseau représenté figure Exemple de multi-topologie. Le point important à remarquer ici est que le routeur R3 n'implémente que la pile IPv4, alors que les deux routeurs R2 et R4 sont double-pile.

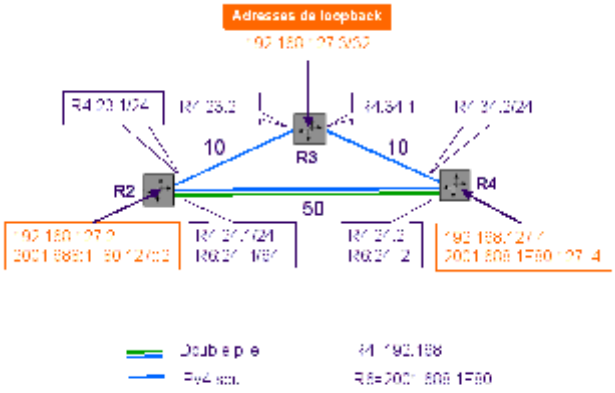


Figure 9-3. Exemple de multi-topologie

Ce paragraphe va démontrer que dans ce cas de configuration il est nécessaire d'implémenter la multi topologies IS-IS afin de prendre en compte la non congruence des deux topologies IPv4 et IPv6. Considérons dans un premier temps que IS-IS ne gère qu'une seule et donc unique topologie. Dans ce cas, les adjacences sont les suivantes:

```
R2#sh clns nei
System Id Interface SNPA State Holdtime Type Protocol
R4 Gi2/0 0005.ddc8.0438 Up 25 L2 IS-IS
R2#
```

On remarque qu'il n'y a pas d'adjacence entre R2 et R3. La même commande passée sur R4 montrerait la même chose. Le routeur R3 se trouve donc isolé (au sens IS-IS). Dans cette même configuration, les routes apprises par R2 sont :

```
R2#sh ip route isis
192.168.127.0/32 is subnetted, 2 subnets
i L2 192.168.127.4 [115/50] via 192.168.24.2, GigabitEthernet2/0
i L2 192.168.34.0/24 [115/60] via 192.168.24.2, GigabitEthernet2/0
R2#sh ipv6 route isis
Codes: I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
I2 2001:688:1F80:127::4/128 [115/50] via FE80::205:DDFF:FEC8:438, GigabitEthernet2/0
R2#
```

Pour atteindre la loopback IPv4 de R4, le chemin sélectionné est le chemin direct ayant un poids de 50 alors qu'un chemin plus léger existe en passant par R3. De plus les réseaux annoncés par R3 sont ignorés

(on ne voit pas son adresse de loopback). La solution pour contrer ce problème est l'utilisation des TLV multi-topologies.

c) Le support de la multi topologies (MT)

Comme cela a été fait pour IP puis IPv6, le support de la multi topologies s'est fait par l'ajout de quatre nouvelles TLV. Elles sont définies dans le draft IETF [Prz-id]. Les principes sont les suivants :

- adjacences : une nouvelle TLV annonce toutes les topologies auxquelles appartiennent chaque interface (TLV 229);
- chaque topologie gère son propre espace d'adressage grâce à des TLVs spécifiques (type 235 et 237);
- chaque topologie exécute un algorithme de plus court chemin et les résultats sont collectés dans des RIB séparées.

La capture ci-dessous est celle d'un paquet IS-IS LSP contenant des TLV multi-topologies :

```
Frame 11 (193 bytes on wire, 193 bytes captured)
IEEE 802.3 Ethernet
Logical-Link Control
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
Intra Domain Routing Protocol Discriminator: ISIS (0x83)
PDU Header Length : 27
Version (==1) : 1
System ID Length : 0
PDU Type : L2 LSP (R:000)
Version2 (==1) : 1
Reserved (==0) : 0
Max.AREAs: (0==3) : 0
ISO 10589 ISIS Link State Protocol Data Unit
PDU length: 176
Remaining Lifetime: 1197s
LSP-ID: 1921.6810.0001.00-00
Sequence number: 0x0000006d
Checksum: 0xf054 (correct)
Type block(0x03): Partition Repair:0, Attached bits:0, Overload bit:0, IS type:3
Area address(es) (4)
Multi Topology (4)
IPv4 unicast Topology (0x000), no sub-TLVs present
IPv6 unicast Topology (0x002), no sub-TLVs present
Protocols supported (2)
Hostname (2)
IP Interface address(es) (4)
IPv6 Interface address(es) (16)
Extended IS reachability (11)
Multi Topology IS Reachability (13)
4 most significant bits reserved, should be set to 0 (0)
IPv6 routing topology (2)
IS neighbor: 1921.6815.0001.03
Extended IP Reachability (29)
Multi Topology Reachable IPv6 Prefixes (44)
4 most significant bits reserved, should be set to 0 (0)
IPv6 routing topology (2)
IPv6 prefix: 2001:688:3::/64, Metric: 10, Distribution: up, internal, no sub-TLVs
present
```

```
IPv6 prefix: 2001:688:20::/64, Metric: 10, Distribution: up, internal, no sub-TLVs
present
IPv6 prefix: 2001:688:100::/64, Metric: 0, Distribution: up, internal, no sub-TLVs
present
```

En reprenant l'exemple précédent, auquel est ajoutée la prise en compte de la multi topologies, les résultats suivants sont obtenus :

```
R2#sh isis topo
IS-IS IP paths to level-2 routers
System Id Metric Next-Hop Interface SNPA
R2 --
R3 10 R3 Fa4/0 000c.ce66.6e01
R4 20 R3 Fa4/0 000c.ce66.6e01
R2#
R2#sh isis ipv6 topo
IS-IS IPv6 paths to level-2 routers
System Id Metric Next-Hop Interface SNPA
R2 --
R3 **
R4 50 R4 Gi2/0 0005.ddc8.0438
R2#
```

Les deux topologies IPv4 et IPv6 sont différentes. Depuis R2, R3 est le next-hop pour la topologie IPv4, alors que c'est R4 pour la topologie IPv6. Un extrait de la base de donnée topologique sur R2 montre que les TLV multi topologies sont mises en œuvre :

```
R1#sh isis data det
IS-IS Level-2 Link State Database:
LSPID LSP Seq Num LSP Checksum LSP Holdtime ATT/P/OL
R4.00-00 0x00000011 0x53A2 891 0/0/0
Area Address: 49.0001
Topology: IPv4 (0x0) IPv6 (0x2)
NLPID: 0xCC 0x8E
Hostname: R4
IP Address: 192.168.24.2
IPv6 Address: 2001:688:1F80:24::2
Metric: 10 IS-Extended R3.01
Metric: 50 IS-Extended R2.02
Metric: 50 IS (MT-IPv6) R2.02
Metric: 0 IP 192.168.127.4/32
Metric: 50 IP 192.168.24.0/24
Metric: 10 IP 192.168.34.0/24
Metric: 50 IPv6 (MT-IPv6) 2001:688:1F80:24::/64
Metric: 0 IPv6 (MT-IPv6) 2001:688:1F80:127::4/128
R2#
```

Les commandes traceroute suivantes sont intéressantes car elles montrent que les chemins IPv4 et IPv6 pour la même destination sont différents :

```
R2#traceroute 192.168.127.4
[...]
1 192.168.23.2 0 msec 0 msec 0 msec
2 192.168.34.2 4 msec * 0 msec
R2#
R2#traceroute 2001:688:1F80:127::4
[...]
1 2001:688:1F80:24::2 4 msec 0 msec 0 msec
R2#
```

De même, l'apprentissage des routes IPv4 et IPv6 montre que l'utilisation de la MT permet de différencier les routes IPv4 des routes IPv6 (on remarquera la métrique de 20 pour l'adresse de loopback IPv4 de R4, alors qu'elle vaut 50 pour l'adresse IPv6):

```
R2#sh ip route isis
192.168.127.0/32 is subnetted, 3 subnets
i L2 192.168.127.4 [115/20] via 192.168.23.2, FastEthernet4/0
i L2 192.168.127.3 [115/10] via 192.168.23.2, FastEthernet4/0
i L2 192.168.34.0/24 [115/20] via 192.168.23.2, FastEthernet4/0
R2#
R2#sh ipv6 route isis
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
U - Per-user Static route
I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
I2 2001:688:1F80:127::4/128 [115/50]
via FE80::205:DDFF:FEC8:438, GigabitEthernet2/0
R2#
```

C) OSPFv3

Le troisième protocole de routage interne, basé sur l'algorithme du plus court chemin, s'appelle OSPF (*Open Shortest Path First*). Relativement plus difficile à mettre en œuvre que RIPng, il est beaucoup plus efficace dans les détections et la suppression des boucles dans les phases transitoires. Ce protocole est basé sur plusieurs sous-protocoles, dont un qui permet une inondation fiable du réseau. Les routeurs possèdent alors chacun une copie des configurations de tous les routeurs présents sur le réseau, et peuvent calculer simultanément le plus court chemin pour aller vers l'ensemble des destinations.

Pour réduire la durée des calculs et surtout pour éviter un recalcule complet des routes à chaque changement de configuration, OSPF offre la possibilité de découper le réseau en aires. Une aire principale appelée backbone relie toutes les autres aires. Les réseaux trouvés dans une aire donnée sont envoyés aux autres aires par les routeurs qui sont en frontière d'aire.

OSPF a été adapté à IPv6 (RFC 2740) ; la version est passée de 2 à 3. La plupart des algorithmes implémentés dans OSPFv2 ont été réutilisés en OSPFv3 ; bien évidemment, certains changements ont été nécessaires en vue de l'adaptation aux fonctionnalités d'IPv6.

En IPv6, la notion IPv4 de sous-réseau est remplacée par celle de lien. Un lien correspond à un ensemble de machines directement connectées au niveau de la couche liaison. Une aire correspond à un ensemble de réseaux interconnectés. La notion de portée (lien, aire, ...) utilise le mécanisme de portée d'IPv6.

OSPF a opéré à une modification du format des champs adresses en vue d'accepter les nouvelles adresses IPv6. Afin de rendre le cœur d'OSPF indépendant du protocole réseau et de le restreindre au transport d'informations sur la topologie, les informations d'adressage ont été retirées des formats des paquets et des LSA (*Link State Advertisements*) sauf pour les LSUA (*Link State Update Packets*) qui utilisent un champ "LSA payload" contenant une adresse IPv6. Les LSA contiennent uniquement des informations sur la topologie du réseau, les routeurs voisins étant identifiés par un numéro de routeur (Router_ID). En liaison étroite avec ce champ d'identification, un champ diffusion a été ajouté afin de déterminer la portée des

paquets. Ce champ peut prendre trois valeurs correspondant à une diffusion sur le lien-local, dans l'aire ou dans le domaine de routage.

Un lien peut supporter plusieurs instances du protocole OSPFv3, ce qui permet de l'utiliser plus facilement sur des liens partagés entre plusieurs domaines de routage. OSPFv3 utilise des adresses lien-local pour l'échange sur les liens. Le mécanisme de sécurité de OSPFv2 a été remplacé par les mécanismes d'authentification et de confidentialité d'IPsec.

4) Routage externe

La seconde famille des protocoles de routage concerne le routage externe. Le terme externe vient du fait qu'il s'agit d'un échange d'informations de routage entre les deux domaines d'administration distincts que sont les systèmes autonomes. Ces systèmes autonomes sont de deux types : les systèmes autonomes terminaux (exemple celui d'un client) et les systèmes autonomes de transit (exemple celui d'un fournisseur d'accès IP). Pour les systèmes autonomes de transit, l'usage de BGP est impératif. Pour les systèmes autonomes terminaux l'usage de BGP est nécessaire dès que le système autonome est multi-connecté (par exemple pour gérer dynamiquement le back-up d'un fournisseur d'accès par un autre).

En IPv4 comme en IPv6, cette notion de domaine d'administration est représenté par un numéro de système autonome codé actuellement sur 2 octets (AS : Autonomous System).

Cette notion de système autonome permet de traiter le problème du routage global de l'Internet (150 000 routes annoncées début 2002) en limitant la complexité. En effet, chaque système autonome (qui peut contenir un grand nombre de routeurs) et ses N routeurs de bord est équivalent pour BGP à un unique routeur virtuel avec N interfaces. La complexité interne au système autonome est donc masquée à l'extérieur de celui-ci.

Avec un protocole de routage externe, il ne s'agit plus de trouver la topologie du réseau, mais d'échanger des informations de routage et de les traiter pour éliminer celles qui ne sont pas pertinentes ou contraires à la politique de routage définie pour le site. En effet, toute annonce de réseau par un domaine implique qu'il accepte de router les paquets vers cette destination.

5) MPLS

Ce chapitre a pour objectif de montrer l'impact d'IPv6 sur la technologie MPLS (Multi Protocol Label Switching). Trois usages qui lient IPv6 à MPLS peuvent être différenciés :

- transition IPv4 vers IPv6 : dans ce contexte, MPLS a été identifié comme une technologie permettant le transport de flux IPv6 à moindre coût. En effet, une fois que le paquet IPv6 est encapsulé dans une trame MPLS sur le routeur d'entrée (le PE-routeur en terminologie MPLS), celle-ci est commutée comme toute autre trame sur les routeurs MPLS de cœur (les P-routeurs). Cette méthode, appelée 6PE (IPv6 Provider Edge) permet de connecter des sites distants IPv6 au travers d'un réseau de cœur MPLS IPv4. La première partie de ce chapitre décrit en détail cette technique. On y trouvera également un exemple concret d'activation de tunnel 6PE sur routeur Cisco;
- mise en place des tunnels MPLS : des protocoles spécifiques (LDP : Label Distribution Protocol, TDP : Tag Distribution Protocol) ou adaptés (BGP, RSVP) construisent les chemins MPLS (les LSP : Label Switched Path) sur la base des informations contenues dans les tables de routage interne. Si les extensions sont décrites dans les RFC 3036 pour LDP et RFC 3209 pour RSVP-TE, aucun constructeur ne les implémente ;
- réseaux privés virtuels : les L3 VPN MPLS représentent le service le plus utilisé de la technologie MPLS. Ils permettent le déploiement de réseaux privés (virtuels car une seule infrastructure physique est utilisée) en assurant une étanchéité entre eux, tout comme si chaque réseau était physiquement différent. Ils se basent sur le RFC 2547 (BGP/MPLS VPN), à laquelle des extensions ont été ajoutées pour le support d'IPv6. La deuxième partie de ce chapitre explore cette technique.

A) MPLS comme outil de transition IPv4 vers IPv6

Utiliser un cœur de réseau MPLS pour transporter des flux IPv6 permet d'interconnecter des îlots IPv6 au travers d'un cœur de réseau IPv4 MPLS. Cette solution est intéressante dans le cadre d'un déploiement d'IPv6 car MPLS commute des labels et non pas des en-têtes IP. Elle offre donc l'avantage de ne pas avoir à mettre à jour les routeurs de cœur.

Historiquement, plusieurs solutions d'encapsulation ont été proposées. Nous allons les décrire rapidement afin d'arriver à la technique 6PE qui est la plus aboutie et finalement la dernière solution avant de passer à un cœur de réseau MPLS gérant directement IPv6.

La figure Architecture MPLS dans le contexte IPv6 schématise ces différentes architectures dans le cadre de l'interconnexion de sites IPv6.

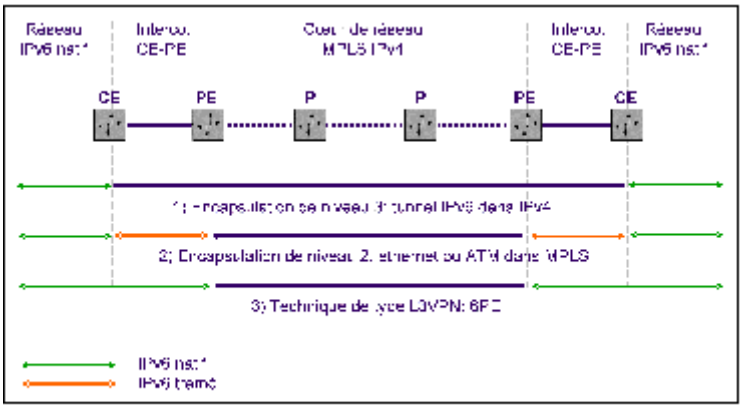


Figure 9-4 : Architecture MPLS dans le contexte IPv6

- Tunnels IP : dans ce cas, le routeur CE encapsule les paquets IPv6 dans des paquets IPv4, les envoie en direction du routeur PE qui les traite comme des paquets IPv4 "normaux" et les transmet dans le LSP MPLS. La liaison CE-PE dans ce cas n'est pas IPv6. Cette technique, repose sur un relayage traditionnel et par conséquent, il n'est pas nécessaire d'avoir des équipements MPLS. L'inconvénient est que d'une part les performances d'encapsulation IPv6 dans IPv4 sont médiocres (lorsqu'elle n'est pas traitée par une carte tunnel dédiée) et d'autre part que la configuration de ces tunnels doit être faite de façon unitaire (ce qui ne tient pas le facteur d'échelle);
- VPN de niveau 2 : dans ce cas, le routeur CE encapsule les paquets IPv6 dans des trames de niveau 2 (Ethernet ou ATM) et les envoie au routeur PE. Celui-ci les transmet directement dans les LSP MPLS. L'avantage de cette technique est la performance de commutation (qui se fait au niveau trame et non plus au niveau 3 comme la solution précédente). L'interconnexion CE-PE voit passer de l'IPv6 dans des trames de niveau 2 mais le routeur PE ne sait pas qu'il met en tunnel des paquets IPv6 (pour lui il s'agit uniquement du niveau 2 transmis dans MPLS). Il n'a donc pas besoin d'être double pile IPv4/IPv6;
- VPN de niveau 3 : dans ce cas, l'interconnexion CE-PE est IPv6. Le routeur PE créé une classe d'équivalence MPLS dédié à chaque préfixe IPv6 et lui attribue un label MPLS. Comme pour les autres types de L3VPN, les trames MPLS ont alors deux labels :
    - un label de transport pour déterminer le LSP (qui peut éventuellement changer à chaque commutation sur un routeur P) et
    - un label de "service 6PE" qui est inchangé pour un préfixe IPv6 donné.

L'avantage de cette méthode est la performance de commutation d'une part et le fait que l'interconnexion CE-PE est IPv6. Par ailleurs, MP-iBGP est utilisé pour attribuer dynamiquement les labels (cela évite d'avoir une configuration manuelle full-mesh des tunnels). Il faut que le routeur PE implémente la fonction 6PE (les routeurs P restent inchangés par contre).

Il est intéressant de constater l'évolution de ces techniques. En effet, au fur et à mesure, les paquets IPv6 se sont rapprochés du routeur PE et donc du cœur de réseau. Aujourd'hui la technique 6PE est la plus aboutie et la plus performante dans le cas d'un déploiement d'IPv6 sur un cœur MPLS IPv4.

B) La technique 6PE

La technique 6PE (cf. figure Architecture 6PE) permet de connecter des îlots IPv6 entre eux au travers d'un cœur de réseau IPv4 MPLS. L'architecture utilisée est la suivante :

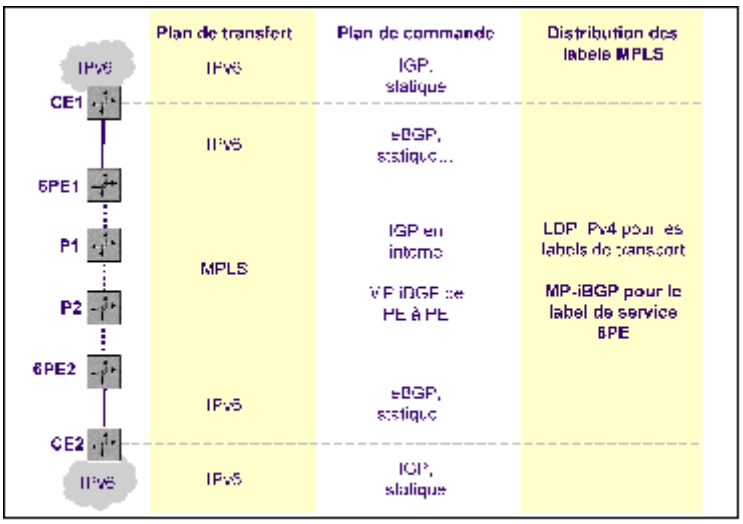


Figure 9-5 . Architecture 6PE

- tunnels MPLS dans le cœur;
- utilisation de MP-iBGP pour annoncer les préfixes IPv6.

Ce mode, décrit dans la RFC 4798, est nommé ainsi en référence à IPv6 et aux PE des VPN MPLS (RFC 2547), mais contrairement à ces derniers, la technique 6PE ne permet pas de faire des VPN. Ainsi, les préfixes IPv6 annoncés par les routeurs CE sont placés dans la table de routage globale du routeur 6PE. La technique 6PE décrit un mode de transition vers IPv6 pour un réseau IPv4 /MPLS dans lequel :

- Comme dans un mode tunnel, les routeurs de cœur ne sont pas doubles pile. Ils restent en IPv4 sans aucune modification;
- Les routeurs de périphérie raccordant des clients ou des sous réseaux IPv6 sont double pile. Ils utilisent MP-iBGP pour s'échanger les préfixes de leurs clients avec eux même comme next hop ;
- Les paquets IPv6 des clients sont reçus nativement par les 6PE, encapsulés par le routeur 6PE d'entrée (ingress), décapsulés par le routeur 6PE de sortie (egress) puis envoyés aux clients sur une interface IPv6 native (ou double pile). Les tunnels sont des LSP MPLS21 (cf. figure Plan de transfert 6PE entre deux clients IPv6).

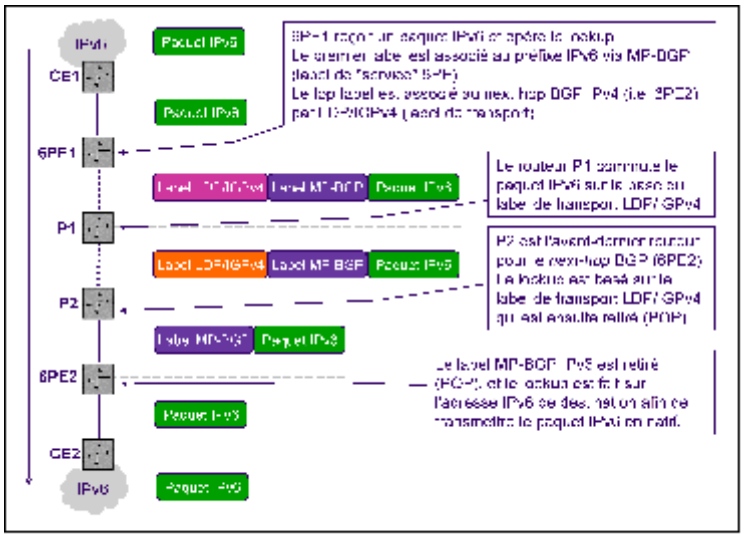


Figure 9-6 . Plan de transfert 6PE entre deux clients IPv6

- Aucun IGP IPv6 n'est utilisé sur les routeurs de cœur, ni sur les routeurs de périphérie. En effet, le next-hop des routes IPv6 est une route IPv6 (obligatoire en MP-BGP) mais cette adresse code en fait une adresse IPv4. Pour cela, une adresse IPv6 de type IPv4-mapped est utilisée permettant de représenter une adresse IPv4 avec une syntaxe IPv6 (le cas d'application ci-dessous en montre un exemple). Finalement, le next-hop des routes IPv6 est l'adresse IPv4 du routeur 6PE de sortie. Cette adresse IPv4 est routable grâce à l'IGP IPv4 existant.

Ce mode nécessite un maillage complet (*full mesh*) de tunnels entre les routeurs 6PE et donc un nombre de tunnels conséquent. Pour éviter la lourdeur et le coût de configuration de tout ces tunnels, le protocole MP-iBGP est utilisé : à chaque route IPv6 cliente annoncée dans MP-BGP, est associé un next-hop qui indique le point de sortie du tunnel et donc le tunnel à utiliser.

Le mode 6PE combine les avantages des modes tunnels :

- Aucun impact sur les routeurs de cœur : pas de nouveau code, pas de nouvelles fonctionnalités (commutation IPv6, ISIS MT ou IS-ISv6, MP-BGP), pas de nouvelles routes (IPv6 internes et externes), aucune dégradation des performances de commutation aussi bien pour les flux IPv4 que IPv6. Des débits IPv6 agrégés de 10G sont possibles dès maintenant. Pas de risques pris sur les routeurs et le réseau de cœur qui ne savent même qu'ils commutent de l'IPv6 (ni de l'IPv4 d'ailleurs);
- Introduction rapide : pour interconnecter deux îlots IPv6, seules deux routeurs sont à mettre à jour pour les passer en double pile (pour les interfaces natives IPv6 des clients) et configurer MP-BGP afin d'échanger les routes clientes IPv6. Or ces deux opérations sont de toutes façons indispensables, quel que soit le mode utilisé (double pile, tunnels statiques IPv4, 6PE) ;
- En cas de panne de liens ou de routeurs de cœur de réseau, les flux IPv6 peuvent utiliser l'intégralité des liens et des routeurs du réseau et donc être re-routés facilement grâce à MPLS.

Tout en supprimant de nombreux inconvénients des tunnels IPv4 statiques :

- Performance du plan de transfert: MPLS permet une encapsulation très rapide quelque soit le débit de l'interface. C'est l'interface cliente qui peut limiter les débits si les interfaces ne sont pas encore assez rapide, mais elle est à plus faible débit qu'une interface backbone. A priori, le surcoût d'encapsulation est plus faible d'où un coût de transport moins élevé pour l'opérateur et une MTU moins réduite pour le client ;

- Lourdeur de configuration des tunnels: aucun tunnel n'est à configurer manuellement. Ils sont établis automatiquement et re-routés dynamiquement en cas de panne.

Le mode 6PE a toutefois quelques inconvénients par rapport à un réseau entièrement double pile :

- Les paquets IPv6 ne sont pas transportés nativement par le réseau, or cela peut être une exigence du client (il est néanmoins possible de masquer les tunnels MPLS au client en ne recopiant pas le TTL des paquets clients dans les datagrammes MPLS).
- Certaines fonctions peuvent ne pas être disponibles une fois le trafic encapsulé dans MPLS.

C) Exemple de mise en œuvre de 6PE

Dans l'exemple suivant, la mise en œuvre de la fonctionnalité 6PE est effectuée sur une plate-forme comprenant 3 routeurs MPLS : deux PE-routeurs et un P-routeur. La fonctionnalité 6PE est introduite de façon incrémentale :

- mise en œuvre du routage interne IS-IS,
- ajout du protocole de distribution des labels LDP,
- ajout des peering BGP,
- et finalement activation de la fonctionnalité 6PE.

Le schéma de la plate-forme est donné dans la figure plateforme MPLS-6PE.

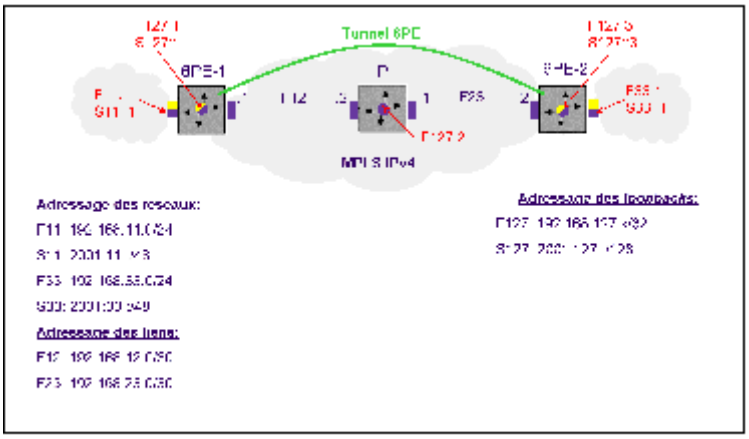


Figure 9-7 . plateforme MPLS-6PE

Les routeurs sont de marque Cisco et les versions du système d'exploitation sont donnés dans le See Versions des IOS pour la plate-forme 6PE.

Versions des IOS pour la plate-forme 6PE		
Routeur	**version**	**Note**
R1	12.2(15)T	6PE aware, DS
R2	12.3(1)	Juste MPLS
R3	12.2(15)T	6PE aware, DS

La première étape consiste à activer les technologies suivantes :

- routage interne IPv4 avec IS-IS ;
- MPLS avec LDP comme protocole de distribution de labels.

Les configurations des routeurs sont les suivantes :

```
6PE-1#sh run
version 12.2
hostname 6PE-1
boot system disk0:c7200-js-mz.122-15.T.bin
ip cef
clns routing
mpls label protocol ldp
mpls ldp logging neighbor-changes
!
interface Loopback6
    ip address 192.168.127.1 255.255.255.255
!
interface Ethernet0/0
    ip address 192.168.12.1 255.255.255.0
    ip router isis
    mpls label protocol ldp
    tag-switching ip
!
interface GigabitEthernet0/0
    ip address 192.168.11.1 255.255.255.0
!
router isis
    net 49.0001.1921.6812.7001.00
    is-type level-2-only
    metric-style wide
    redistribute connected
    passive-interface GigabitEthernet0/0
    passive-interface Loopback6
!
ip route 192.168.111.0 255.255.255.0 GigabitEthernet0/0
6PE-1#
6PE-2#sh run
version 12.2
hostname 6PE-2
boot system disk0:c7200-js-mz.122-15.T.bin
ip cef
clns routing
mpls label protocol ldp
no mpls ldp logging neighbor-changes
!
interface Loopback6
    ip address 192.168.127.3 255.255.255.255
```

```
!
interface Ethernet0/0
    ip address 192.168.23.2 255.255.255.0
    ip router isis
    mpls label protocol ldp
    tag-switching ip
!
interface GigabitEthernet0/0
    ip address 192.168.33.1 255.255.255.0
!
router isis
    net 49.0001.1921.6812.7003.00
    is-type level-2-only
    metric-style wide
    redistribute connected
    passive-interface GigabitEthernet0/0
    passive-interface Loopback6
!
ip route 192.168.133.0 255.255.255.0 GigabitEthernet0/0
6PE-2#
P#sh run
version 12.3
hostname P
boot system flash:C2600-JS-MZ.123-1.BIN
ip cef
clns routing
mpls label protocol ldp
mpls ldp logging neighbor-changes
!
interface Loopback0
    ip address 192.168.127.2 255.255.255.255
!
interface FastEthernet0/0
    ip address 192.168.12.2 255.255.255.0
    ip router isis
    mpls label protocol ldp
    tag-switching ip
!
interface FastEthernet0/1
    ip address 192.168.23.1 255.255.255.0
    ip router isis
    mpls label protocol ldp
    tag-switching ip
!
router isis
    net 49.0001.1921.6812.7002.00
    is-type level-2-only
    metric-style wide
    redistribute connected
    passive-interface Loopback0
!
P#
```

Pour vérifier que les configurations des routeurs sont correctes, il est possible de tester l'apprentissage des routes par IS-IS. Sur le routeur 6PE-2, la commande suivante permet de vérifier que les routes sont bien apprises :

```
6PE-2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
```

```
P - periodic downloaded static route
Gateway of last resort is not set
i L2 192.168.12.0/24 [115/20] via 192.168.23.1, Ethernet0/0
S 192.168.133.0/24 is directly connected, GigabitEthernet0/0
192.168.127.0/32 is subnetted, 3 subnets
C 192.168.127.3 is directly connected, Loopback6
i L2 192.168.127.2 [115/10] via 192.168.23.1, Ethernet0/0
i L2 192.168.127.1 [115/20] via 192.168.23.1, Ethernet0/0
i L2 192.168.11.0/24 [115/20] via 192.168.23.1, Ethernet0/0
C 192.168.23.0/24 is directly connected, Ethernet0/0
C 192.168.33.0/24 is directly connected, GigabitEthernet0/0
6PE-2#
```

De même, pour l'apprentissage des labels MPLS par LDP. Sur le routeur P la commande suivante permet de vérifier l'activation du protocole LDP sur les interfaces :

```
P#sh mpls interfaces detail
Interface FastEthernet0/0:
IP labeling enabled (ldp)
LSP Tunnel labeling not enabled
BGP tagging not enabled
Tagging operational
Fast Switching Vectors:
IP to MPLS Fast Switching Vector
MPLS Turbo Vector
MTU = 1500
Interface FastEthernet0/1:
IP labeling enabled (ldp)
LSP Tunnel labeling not enabled
BGP tagging not enabled
Tagging operational
Fast Switching Vectors:
IP to MPLS Fast Switching Vector
MPLS Turbo Vector
MTU = 1500
P#
```

Enfin sur le routeur 6PE-2, la commande suivante permet d'afficher la table de commutation MPLS :

```
6PE-2#sh mpls forwarding-table
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
16 Pop tag 192.168.12.0/24 0 Et0/0 192.168.23.1
17 Pop tag 192.168.127.2/32 0 Et0/0 192.168.23.1
18 17 192.168.127.1/32 0 Et0/0 192.168.23.1
19 18 192.168.11.0/24 0 Et0/0 192.168.23.1
6PE-2#
```

Ainsi :

- pour la FEC `192.168.127.1/32`, le label de sortie sera 17 ;
- pour la FEC `192.168.12.0/24`, 6PE-2 fait un POP du label (car il est le PHP pour le next-hop de ce préfixe, i.e. le router P).

La commande traceroute vers 192.168.127.1 montre que le flux passe sur MPLS et que le tag de sortie est effectivement 17 :

```
6PE-2#traceroute 192.168.127.1
Tracing the route to 192.168.127.1
1 192.168.23.1 [MPLS: Label 17 Exp 0] 0 msec 4 msec 0 msec
2 192.168.12.1 0 msec * 0 msec
```

```
6PE-2#
```

La capture d'un PING vers 192.168.127.1 confirme également que le trafic emprunte le LSP (cf. figure Capture d'un paquet de ping).

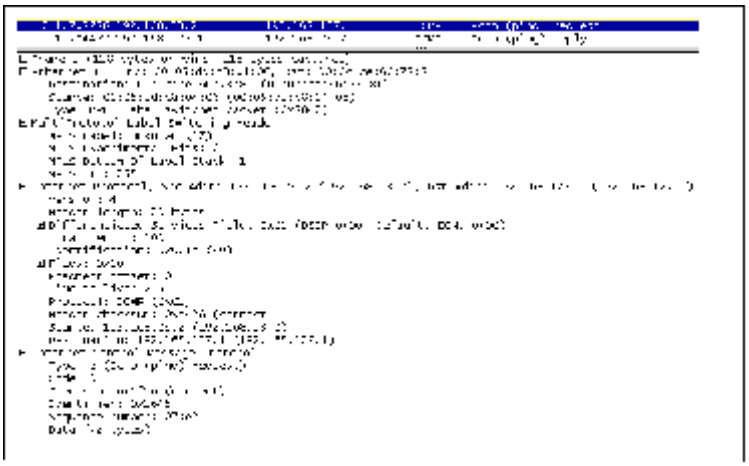


Figure 9-8 . Capture d'un paquet de ping

La seconde étape consiste à ajouter de la fonction 6PE. Le routeur P n'est pas concerné par cette fonction, par contre les routeurs 6PE-1 et 6PE-2 doivent établir une session i-BGP entre eux afin de pouvoir s'échanger les préfixes IPv6 avec MP-BGP. Les configurations des routeurs sont alors les suivantes (seuls les éléments nouveaux par rapport aux configurations précédentes sont listés) :

```
6PE-1#sh run
[..]
ipv6 unicast-routing
mpls ipv6 source-interface Loopback6
!
interface Loopback6
    ip address 192.168.127.1 255.255.255.255
    ipv6 address 2001:127::1/128
!
interface Ethernet0/0
!
interface GigabitEthernet0/0
    ipv6 address 2001:11::1/48
    ipv6 enable
!
router isis
[..]
!
router bgp 106
[..]
    neighbor 192.168.127.3 remote-as 106
    neighbor 192.168.127.3 update-source Loopback6
!
address-family ipv6
    neighbor 192.168.127.3 activate
    neighbor 192.168.127.3 soft-reconfiguration inbound
    neighbor 192.168.127.3 send-label
    redistribute connected
    redistribute static
    exit-address-family
!
address-family ipv4
    redistribute connected
    redistribute static
```

```
    neighbor 192.168.127.3 activate
    neighbor 192.168.127.3 soft-reconfiguration inbound
    exit-address-family
!
ipv6 route 2001:111::/32 GigabitEthernet0/0
6PE-1#

6PE-2#sh run
[..]
ipv6 unicast-routing
mpls ipv6 source-interface Loopback6
!
interface Loopback6
    ip address 192.168.127.3 255.255.255.255
    ipv6 address 2001:127::3/128
!
interface Ethernet0/0
[..]
!
interface GigabitEthernet0/0
[..]
    ipv6 address 2001:33::1/48
    ipv6 enable
!
router isis
[..]
!
router bgp 106
[..]
    neighbor 192.168.127.1 remote-as 106
    neighbor 192.168.127.1 update-source Loopback6
!
    address-family ipv6
        neighbor 192.168.127.1 activate
        neighbor 192.168.127.1 soft-reconfiguration inbound
        neighbor 192.168.127.1 send-label
        redistribute connected
        redistribute static
        exit-address-family
!
    address-family ipv4
        redistribute connected
        redistribute static
        neighbor 192.168.127.1 activate
        neighbor 192.168.127.1 soft-reconfiguration inbound
        exit-address-family
!
ipv6 route 2001:133::/32 GigabitEthernet0/0
6PE-2#
```

La commande suivante permet de vérifier cette configuration en testant le peering BGP sur 6PE-2 :

```
6PE-2#sh bgp ipv6 neighbor
BGP neighbor is 192.168.127.1, remote AS 106, internal link
BGP version 4, remote router ID 192.168.127.1
BGP state = Established, up for 00:34:04
Last read 00:00:04, hold time is 180, keepalive interval is 60 seconds
Neighbor capabilities:
Route refresh: advertised and received(old & new)
Address family IPv4 Unicast: advertised and received
Address family IPv6 Unicast: advertised and received
ipv6 MPLS Label capability: advertised and received
[..]
For address family: IPv6 Unicast
BGP table version 3, neighbor version 3
```

```
Index 1, Offset 0, Mask 0x2
Inbound soft reconfiguration allowed
Sending Prefix & Label
Sent Rcvd
Prefix activity: ---- ----
Prefixes Current: 1 1 (Consumes 72 bytes)
Prefixes Total: 2 2
[..]
6PE-2#sh bgp ipv6 2001:127::1/128
BGP routing table entry for 2001:127::1/128, version 3
Paths: (1 available, best #1, table Global-IPv6-Table)
Not advertised to any peer
Local, (received & used)
::FFFF:192.168.127.1 (metric 20) from 192.168.127.1 (192.168.127.1)
Origin incomplete, metric 0, localpref 100, valid, internal, best
6PE-2#
```

Sur le routeur 6PE-2, les labels utilisés par MP-BGP pour le transport d'IPv6 sur MPLS peuvent être visualisés :

```
6PE-2#sh bgp labels
Network Next Hop In label/Out label
2001:111::/32 ::FFFF:192.168.127.1
nolabel/22
2001:127::1/128 ::FFFF:192.168.127.1
nolabel/21
2001:127::3/128 :: 21/nolabel
2001:133::/32 :: 22/nolabel
2003::/16 ::FFFF:192.168.127.1
nolabel/23
2005:1234::/32 ::FFFF:192.168.127.1
nolabel/24
6PE-2#
```

Pour cet exemple, des routes statiques IPv6 supplémentaires a été ajoutées sur le routeur 6PE-1 afin de montrer que chaque préfixe se voit attribué un nouveau label. La capture (cf. figure Capture d'une annonce MP-BGP) décode l'annonce MP-BGP qui résulte de l'ajout de la route statique IPv6 2005 :1234 ::/32 (message BGP UPDATE). Ce message BGP UPDATE annonce à la fois le préfixe IPv6 (2005:1234::/32) et le label MPLS associé (18 en hexadécimal, soit 24 en décimal, avec le bit S positionné à 1) (RFC 3107).

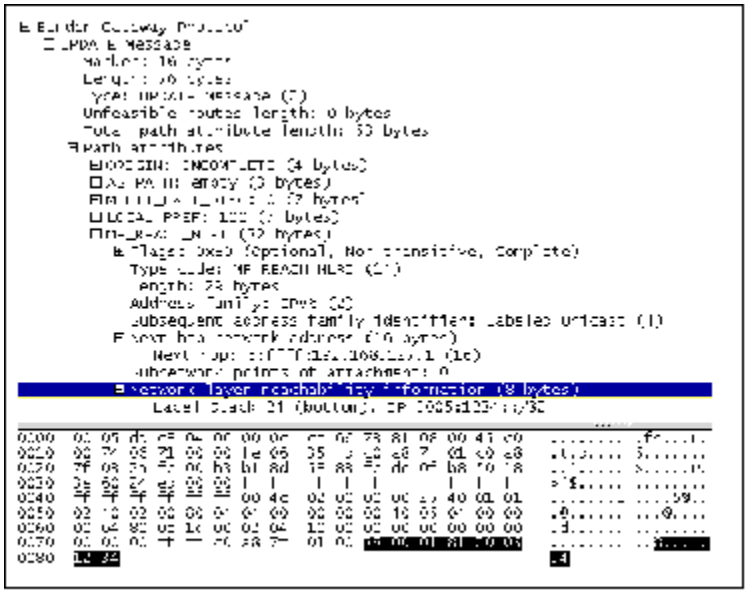


Figure 9-9 . Capture d'une annonce MP-BGP

Sur le routeur 6PE-2, il est possible de visualiser comment sont apprises les routes :

```
6PE-2#sh ipv6 route
IPv6 Routing Table - 4 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
U - Per-user Static route
I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
B 2001:127::1/128 [200/0]
via ::FFFF:192.168.127.1, IPv6-mpls
LC 2001:127::3/128 [0/0]
via ::, Loopback6
L FE80::/10 [0/0]
via ::, Null0
L FF00::/8 [0/0]
via ::, Null0
6PE-2#
```

Le préfixe IPv6 `2001:127::1/128` est appris par MP-BGP via l'adresse IPv6 IPv4 mappée `::FFFF:192.168.127.1`. L'indication IPv6-mpls montre que le flux IPv6 correspondant est acheminé sur MPLS (c'est la fonction 6PE).

Sur le routeur 6PE-2, la table de commutation MPLS peut être affichée :

```
6PE-2#sh mpls forwarding-table
Local Outgoing Prefix Bytes tag Outgoing Next Hop
tag tag or VC or Tunnel Id switched interface
16 Pop tag 192.168.12.0/24 0 Et0/0 192.168.23.1
17 Pop tag 192.168.127.2/32 0 Et0/0 192.168.23.1
18 17 192.168.127.1/32 0 Et0/0 192.168.23.1
19 18 192.168.11.0/24 0 Et0/0 192.168.23.1
21 Aggregate IPv6 1040
6PE-2#
```

Le label attribué à la fonction 6PE possède la valeur 21. La capture d'un ECHO REQUEST vers `2001:127::1` (cf. figure Capture d'un message "echo request") montre que le flux emprunte le LSP 6PE :

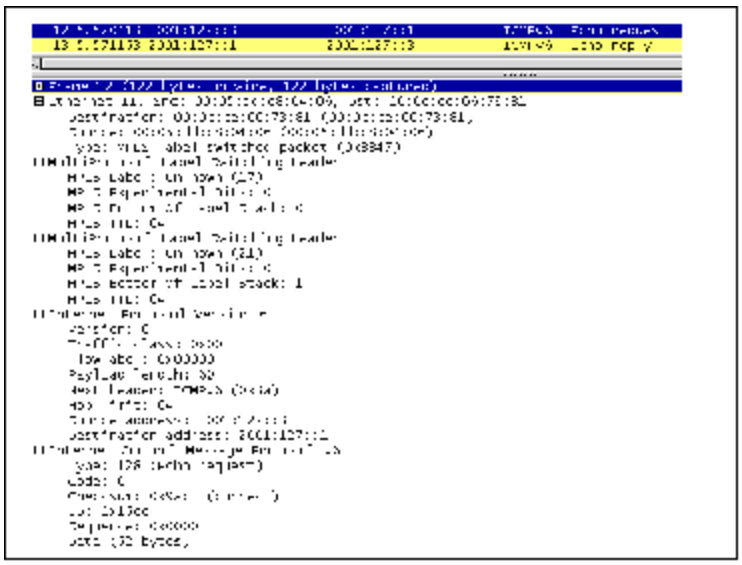


Figure 9-10 . Capture d'un message "echo request"

On remarque qu'il y a deux labels :

- le label « normal » qui assure la commutation MPLS (et l'on voit alors que le routeur P ignore qu'il commute de l'IPv6) ;
- le label 6PE qui permet ensuite d'établir la correspondance avec IPv6 sur le routeur PE de sortie.

La capture de la figure Capture de la réponse est la réponse à cette requête.

Figure 9-11 . Capture de la réponse

On remarquera qu'il n'y a que le label 6PE car le routeur précédent (routeur P) a déjà décapsulé le label « normal » de commutation MPLS.

D) Réseaux privés virtuels IPv6 sur MPLS

Cette technique crée des VPN IPv6 en utilisant les LSP MPLS d'un cœur de réseau IPv4 MPLS (le cœur de réseau n'est pas IPv6 MPLS mais bien IPv4 MPLS), offrant ainsi l'avantage d'utiliser le cœur de réseau MPLS déjà existant.

Du point de vue service réseau, cette technique est à IPv6 ce que les VPN MPLS "standards" sont à IPv4. Elle s'appuie sur les extensions BGP afin d'échanger les préfixes de VPN IPv6 et les labels MPLS associés. Ainsi, le raccordement entre le routeur PE (alors appelé 6VPE) et le routeur CE est en IPv6 natif (comme avec la technique 6PE).

La terminologie des L3 VPN MPLS IPv4 est reprise: routeurs CE, PE (6VPE en fait), P, communauté BGP étendue, Site Of Origin, Route Target, Route Distinguisher, adresses VPN-IPv6 (constituée des 64 bits du RT et des 128 bits de l'adresse IPv6), VRF.

L'architecture utilise les propositions relatives aux VPN IP BGP/MPLS RFC 2547bis25) et 6PE.

Le plan de transfert est le même que celui de la technique 6PE (c.f. Plan de transfert 6PE entre deux clients IPv6). Nous détaillons donc plutôt le plan de commande qui est plus complexe que dans la technique 6PE :

- Les sites clients sont en IPv6 (ou double pile). Ils ne savent pas qu'ils font partie d'un VPN. Les routeurs CE et 6VPE s'échangent les routes IPv6 soit par routage statique, interne ou externe. Les routeurs 6VPE s'échangent les préfixes IPv6 VPN entre eux par MP-iBGP.
- Les routeurs 6VPE implémentent des tables de routage séparées :
    - La table de routage globale: elle est constituée de préfixes IPv4 distribués par le protocole de routage interne IPv4 du cœur de réseau MPLS. Eventuellement, il peut y avoir des préfixes IPv6 si un protocole de routage interne IPv6 est implémenté dans le cœur de réseau;
    - Les VRF: elles sont associées aux interfaces (physiques ou logiques) connectant les sites du (des) VPN(s) au routeur 6VPE. Les routes VPN IPv6 apprises par MP-iBGP sont placées dans les VRF associées.
- Lorsqu'un site IPv6 VPN annonce son préfixe à son routeur 6VPE, cette annonce est transformée en un message MP-iBGP UPDATE incluant le champ MP_REACH_NLRI26 qui a les valeurs suivantes:
    - AFI (address family id.) = 2 (IPv6); SAFI (subsequent address family id.) = 128, RD (route distinguisher) = R1.
    - Le next-hop est remplacé par l'adresse de loopback du routeur 6VPE et codée sous forme d'une adresse IPv4-mapped (MP-BGP impose qu'il soit codé dans la même famille d'adresse que le préfixe annoncé).
    - Un label MPLS est attribué pour ce préfixe IPv6 VPN. C'est celui-là qui devra être ajouté au label de transport MPLS "classique" qui sert à établir le LSP. Ce label IPv6 VPN reste inchangé tout au long du parcours dans le LSP (alors que le label LDP IPv4 peut changer de proche en proche).

a) Accès Internet en IPv6 depuis le VPN

Il est possible de définir (dans la configuration globale) une route IPv6 dans la VRFv6 avec un next-hop dans la table de routage globale. Dans le cas d'un VPN multi-sites cela permet de définir un des sites comme passerelle vers l'internet IPv6 ouvert (par exemple, une route par défaut permet de joindre l'ASBR vers l'Internet (l'ASBR n'est pas dans un VPN)).

Pour le trafic retour on définit, pour les routes de chaque site, un next-hop routable dans une VRF donnée (à condition que chaque site attribue un préfixe IPv6 publique à ses utilisateurs de l'internet IPv6 ouvert).

On peut noter que cette fonctionnalité n'est pas implémentée en VPN IPv4 car les adresses IPv4 utilisées dans ce cas sont privées (RFC 1918) et une route statique ne serait pas suffisante (il serait nécessaire de faire du NAT en plus).

Certains constructeurs (comme Juniper ou Cisco) disposant de la fonction L3 VPN MPLS pour IPv4 implémentent également cette technique en IPv6.

6) BGP

BGP4 est le protocole de routage externe actuellement utilisé pour le routage global de l'Internet IPv4 (la version 4, identique pour BGP et IP, est pure coïncidence). Compte tenu de sa criticité, ce protocole est l'objet d'évolutions constantes. L'une d'entre elles est le RFC 2858 qui rend BGP4 multi-protocole en introduisant la notion de famille d'adresse (ex. IPv4, IPv6, IPX...) et de sous-famille d'adresse (ex. unicast, multicast). Le RFC 2545 précise l'usage des extensions multi-protocoles pour le cas d'IPv6.

L'adaptation multi-protocole de BGP4 est assez simple car elle ne concerne que les trois attributs dont le format dépend de l'adresse soit :

- `NLRI` : Network Layer Reachability Informations (suite de préfixes);
- `NEXT_HOP` : Adresse IP où il faut router les NLRI;
- `AGGREGATOR` : Adresse IP du routeur qui a fait une agrégation de préfixes.

Pour réaliser pratiquement cette adaptation, BGP4+ introduit deux nouveaux attributs :

- `MP_REACH_NLRI` : Multiprotocol Reachable NLRI;
- `MP_UNREACH_NLRI` : Multiprotocol Unreachable NLRI.

qui indiquent que l'on annonce des informations de routage autres que les routes unicasts IPv4. Ces attributs codent en premier le type de famille et de sous-famille d'adresse, puis les attributs dont le format est spécifique. Les autres attributs (comme le chemin d'AS Autonomous Systems) sont codés et annoncés sans changement.

Les implémentations du RFC 2858 sont souvent appelées MBGP (pour faire référence à leur capacité de traitement des routes multicast) ou BGP4+ (pour faire référence à leur capacité de traitement de routes IPv6). Pour l'anecdote, le numéro de version du protocole n'a pas été modifié (en BGP5 par exemple) car le passage de BGP3 à BGP4 rappelle trop de souvenirs douloureux à ceux qui l'ont mis en œuvre. Les numéros d'AS utilisés pour IPv4 servent aussi pour IPv6.

A) Règles d'annonce et d'agrégation

Pour assurer un bonne gestion du routage en IPv6, il est nécessaire de définir une politique de routage cohérente, qui agrège les sous-réseaux en un réseau global à chaque frontière de domaine (site, opérateur,...). Le RFC 2772 donne les règles d'agrégation souhaitables pour le 6bone. En particulier :

- il ne faut pas annoncer les différents sous-réseaux d'un site à l'extérieur de ce site, mais au contraire annoncer une route unique pour tout le site ;
- aux différentes frontières du plan d'adressage agrégé, il faut regrouper les différents NLA en un seul préfixe ;
- et naturellement les adresses non globales (lien local, site local) ne doivent pas être annoncées.

Avec les plans d'adressages actuels, cela se traduit par les règles d'annonces suivantes :

- Un site ne doit pas annoncer de préfixe plus long que /48.
- Dans le routage entre nœuds d'interconnexion, seuls existent des annonces de préfixe de longueur /24 ou /28 pour les préfixes "6bone" (`3ffe:xxx`, cf. Adresses de test), de longueur /35 pour les préfixes "plan autorités régionales" (`2001:xxx`, cf. Adresses globales : plan d'adressage agrégé), et `2002::/16` (cf. 6to4).

Bien entendu ces règles n'interdisent pas d'échanger des routages plus spécifiques entre deux organisations qui ont des collaborations.

X) Configuration des routeurs

Ce chapitre donne des exemples de configurations de routage IPv6. Il commence par décrire comment configurer des routeurs spécialisés ; dans ces exemples, la logique suivante a été adoptée pour faciliter la lecture : démarrer de zéro et configurer les différentes fonctions au fur et à mesure afin d'arriver à terme à un réseau IPv6 dont l'architecture irait du LAN au WAN. Pour ce faire, les différentes étapes sont les suivantes:

- configuration des interfaces (adresse IPv6, tunnel, annonce de préfixe sur un lien) ;
- configuration du routage interne (ISIS, OSPF, RIP) ;
- configuration du routage externe (BGP).

Les configurations présentées concernent des équipements des industriels Alcatel, Cisco et Juniper.

Une seconde partie décrit comment se passer d'équipements spécialisés et utiliser un ordinateur comme routeur. Cette partie décrit les commandes spécifiques pour différents systèmes Unix classiques. Enfin on décrit le logiciel Zebra/Quagga qui est un programme de routage dynamique utilisable sur de nombreux systèmes.

1) Configuration des interfaces : adresse et préfixe

Cette première étape indique explicitement au routeur qu'il devra activer sa pile de protocole IPv6 sur certaines interfaces. Quelques tests élémentaires de connectivité (typiquement le ping) permettent de vérifier que des routeurs voisins se voient et que les piles protocolaires sont bien en place.

Le premier besoin de configuration IPv6 concerne l'adressage des interfaces. On trouvera ci-dessous des exemples concernant :

- la configuration d'adresse IPv6 fixe,
- l'annonce d'un préfixe sur une interface afin que les hôtes raccordés puissent se configurer automatiquement (stateless auto configuration).

Les types d'interfaces abordés sont :

- Ethernet : en mode natif ou avec VLAN (802.1Q):
  - OmnisSwitch Alcatel
  - Cisco
  - Juniper
- Packet Over Sonet (POS):
  - Cisco
  - Juniper
- Asynchronous Transfert Mode (ATM):
  - Juniper
- les tunnels IPv6 dans IPv4.
  Ces tunnels étant considérés comme une interface logique (ils sont d'ailleurs configurés au niveau interface), ils ont donc logiquement leur place ici.
  Les exemples ci-dessous permettent de configurer une tête de tunnel IPv6 dans IPv4. Ce type de tunnel est le premier outil de transition IPv4 vers IPv6 puisqu'il permet d'encapsuler un paquet IPv6 dans un paquet IPv4 de façon manuel (le tunnel doit être préconfiguré de part et d'autre, ce qui n'est pas le cas pour les tunnels 6to4 par exemple).
  - OmnisSwitch Alcatel
  - Cisco
  - Juniper

2) Annonce de préfixe sur un lien

L'une des fonctions les plus intéressantes d'IPv6 est l'auto-configuration passive d'adresses. Cette caractéristique est implémentée dans le protocole de découverte des voisins (neighbor discovery) en émettant le préfixe réseau sur le lien dans un paquet de type RA (router advertisement). Les exemples ci-dessous montrent comment émettre un préfixe réseau sur une interface.

- OmnisSwitch Alcatel
- Cisco
- Juniper

3) Configuration du Routage

Une fois les interfaces configurées (adresses, tunnels, annonce de préfixe), une connectivité de niveau 3 existe uniquement entre les routeurs directement connectés. Pour étendre cette connectivité, il faut mettre en œuvre le routage IP. Il existe 3 types de routage :

- le routage statique : les routes sont configurées manuellement et ne nécessitent pas la mise en œuvre d'un protocole de routage;
  - OmnisSwitch Alcatel

    - Cisco
    - Juniper
- le routage interne : cantonné à un domaine administratif (ou système autonome, AS), il permet de distribuer des préfixes du domaine à des routeurs du domaine. On trouvera ci-dessous des exemples pour les protocoles de routage internes (IGP):
    - RIP,
        - OmnisSwitch Alcatel
        - Cisco
    - ISIS et
        - Cisco
        - Juniper
    - OSPF
        - Cisco
        - Juniper

- le routage externe : il permet à un AS d'échanger des préfixes réseaux avec des AS voisins. Ces préfixes, une fois redistribués à l'intérieur de l'AS, permettent d'avoir la connectivité vers l'Internet mondial. On trouvera ci-dessous des exemples pour le protocole EGP le plus répandu aujourd'hui : BGP4.
    - Cisco
    - Juniper

4) Utilisation d'un ordinateur comme routeur

En IPv6 les fonctions de machine simple («node») et de routeur ont été nettement séparées. Il est donc courant d'utiliser de matériels spécialisés comme routeurs, et de configurer les machines normales comme des nœuds simples, sans fonction de routage (cf. Installation d'un équipement), et ce d'autant plus qu'un routeur dédié peut devenir complexe (plusieurs protocoles de routage, filtrage des paquets, interfaces spécialisées, VLANs, MPLS...).

Il est malgré cela toujours possible d'utiliser un ordinateur universel comme routeur, en configurant des routes à la main ou en lançant des logiciels de routage. Il ne faut pas oublier que dans ce cas l'auto-configuration «sans état» ne peut plus être utilisée, et qu'il faut donner explicitement des adresses aux différentes interfaces.

- Routeur Solaris
- Routeur Linux
- Routeur FreeBSD
- Routeur NetBSD

5) Zebra – Quagga

Zebra est un logiciel de routage disponible librement (licence GPL) qui permet de transformer une machine Unix en un véritable routeur. Les protocoles RIP, OSPF, BGP sont implémentés dans leurs instances IPv4 et IPv6. Ce logiciel est architecturé de façon très modulaire (cf. figure Organisation du logiciel de routage Zebra). Chaque protocole de routage est supporté par un processus, et dialogue avec un processus maître (zebra) qui, lui, est chargé de gérer la table de routage de la machine en fonction des informations que lui transmettent les processus correspondants aux protocoles activés. Chaque protocole peut être paramétré en éditant un fichier de configuration spécifique ou de façon interactive au moyen d'un telnet sur un port associé. L'interface de commande et de configuration est calquée sur celle de l'IOS de Cisco. Quagga est un logiciel de routage plus récent, dérivé de Zebra, son interface et son utilisation sont très proches de celles de Zebra.

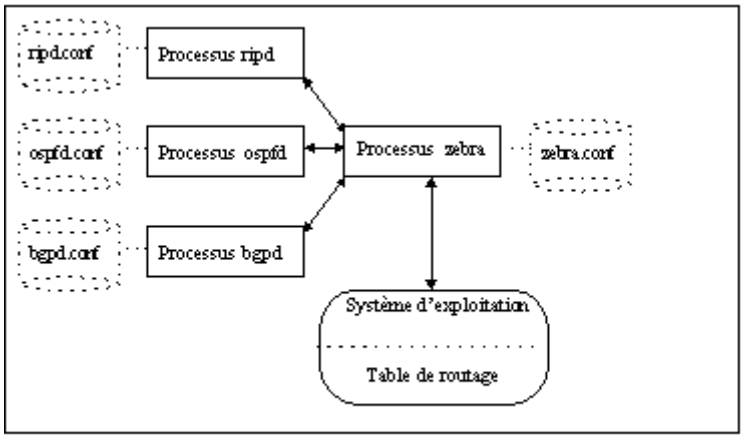


Figure 10-1. Organisation du logiciel de routage Zebra

A) Installation de Zebra ou Quagga

En général, et selon les versions du système, l'un ou l'autre de ces logiciels est disponible avec le support IPv6. En FreeBSD 5.x et NetBSD 2.x, les deux logiciels sont disponibles comme paquetage binaire, il suffit de l'installer par la commande `pkg_add`. En Linux FedoraCore, Quagga est un paquetage RPM standard. En Solaris 10, Zebra est disponible dans les CDs de distribution ; toutefois la version binaire ne connait pas IPv6, il faut récupérer le paquetage source (`SUNWzebraS`) et le compiler pour avoir le support IPv6.

Un autre solution est de récupérer le source sur les sites de référence et de l'installer. Lancement et configuration initiale

L'installation a créé des binaires, des scripts de lancement et des fichiers d'exemples de configuration. Dans la suite, nous prendrons l'exemple du paquetage Zebra sur FreeBSD ; pour les autres systèmes il faut adapter les chemins d'accès et les noms des scripts.

La première étape consiste à créer le fichier de configuration zebra.conf du processus maître Zebra, puis à lancer le processus. Deux méthodes sont possibles :

- éditer le fichier (si le processus zebra n'est pas lancé)

```
>cd /usr/local/etc/zebra
>cp zebra.conf.sample zebra.conf
>vi zebra.conf
>zebractl start
```

- lancer le processus et utiliser l'interface telnet

```
>cp /usr/local/etc/zebra/zebra.conf.sample /usr/local/etc/zebra/zebra.conf
>zebractl start
>telnet localhost zebra
Password: zebra
Router> enable
Password: zebra
Router# show running-config
Router# conf t
^Z
Router# write
```

La syntaxe est inspirée de celle de l'IOS de Cisco.

Le lancement et l'arrêt de tous les processus zebra en FreeBSD se fait par la commande `zebractl start` (ou `stop`). Pour un lancement au démarrage de la machine, il suffit de faire :

```
>ln -s /usr/local/sbin/zebractl /usr/local/etc/rc.d/zebractl.sh
```

La configuration du processus Zebra décrit les interfaces, les routes statiques et les annonces de préfixe et de routeur. Il faut se méfier des interférences entre Zebra et la configuration globale du système, qui peut faire des choix ou lancer des démons (annonces, RIPng) sans tenir compte du lancement de Zebra ; de même pour l'activation du relayage des paquets, et le lancement de la configuration automatique des interfaces. Une bonne précaution est de laisser dans la configuration système la déclaration du mode routeur (`ipv6_gateway_enable=YES` dans `/etc/rc.conf` pour FreeBSD), et des adresses des interfaces, et de mettre dans le fichier de configuration Zebra les routes statiques et les annonces de préfixes et routeur (`rtadvd_enable=NO`).

Considérons l'exemple d'une machine thieste qui héberge le logiciel Zebra et qui a un interface Ethernet et un tunnel (voir Tunnel sous FreeBSD).

La configuration IPv6 des interfaces se fait dans `/etc/rc.conf` par :

```
ipv6_enable=YES
ipv6_gateway_enable=YES
ipv6_router_enable=NO
rtadvd_enable=NO
ipv6_prefix_xl0="2001:660:282:1"
gif_interfaces="gif1"
gif_config_gif1="192.108.119.167 192.1098.119.189"
ipv6_ifconfig_gif1="3ffe:305:1014:1::1 3ffe:305:1014:1::2"
```

Le résultat de cette configuration se vérifie par la commande `ifconfig` :

```
>ifconfig xl0
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet 192.108.119.167 netmask 0xffffffc0 broadcast 192.108.119.191
inet6 fe80::204:76ff:fe0b:9bf4%xl0 prefixlen 64 scopeid 0x2
inet6 2001:660:282:1:204:76ff:fe0b:9bf4 prefixlen 64
```

```
ether 00:04:76:0b:9b:f4
media: autoselect (100baseTX <full-duplex>) status: active
supported media: autoselect 100baseTX <full-duplex> 100baseTX 10baseT/UTP <full-duplex>
  10baseT/UTP  100baseTX <hw-loopback>
>ifconfig gif1
gif1: flags=8011<UP,POINTOPOINT,MULTICAST> mtu 1280
inet6 fe80::203:47ff:fe22:9cd8%gif1 --> :: prefixlen 64
inet6 3ffe:305:1014:1::1 --> 3ffe:305:1014:1::2 prefixlen 128
physical address inet 192.108.119.167 --> 192.108.119.189
```

B) Configuration de l'annonce de préfixe par Zebra

Sur le schéma de l'exemple figure Session BGP sous Zebra, on veut provoquer l'annonce du préfixe `3ffe:305:1014::/48` par le routeur `thieste` sur le tunnel vers le PC Linux. Cette annonce se traduit par l'extrait du fichier de configuration de Zebra ci -après :

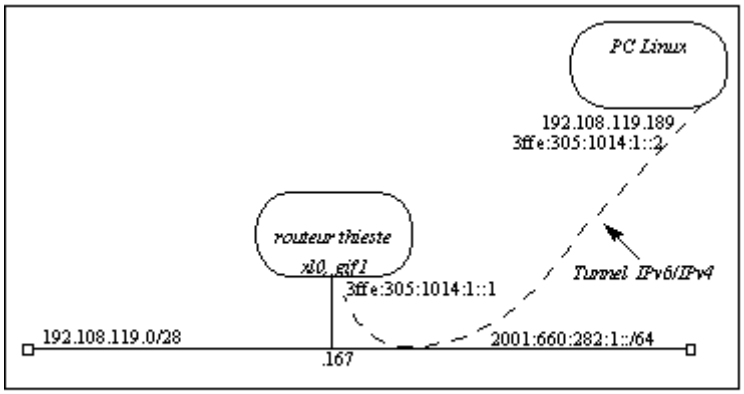


Figure 10-2 . Tunnel sous FreeBSD

```
interface gif1
ipv6 nd send-ra
ipv6 nd prefix-advertisement 3ffe:305:1014::/48
```

Pour vérifier que l'annonce a bien produit son effet sur la machine Linux qui est à l'autre extrémité du tunnel on trouvera ci-dessous un extrait de la commande `ifconfig` donnée sur cette machine :

```
testG6 Lien encap:IPv6-dans-IPv4
adr inet6: fe80::c06c:77bd/128 Scope:Lien
adr inet6: 3ffe:305:1014:1::2/0 Scope:Global
adr inet6: 3ffe:305:1014::c06c:77bd/64 Scope:Global
UP POINTOPOINT RUNNING NOARP MTU:1480 Metric:1
RX packets:13 errors:0 dropped:0 overruns:0 frame:0
TX packets:333 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:0
RX bytes:1144 (1.1 Kb) TX bytes:41292 (40.3 Kb)
```

C) Configuration de routage dynamique

Une fois le processus maître configuré, il faut configurer les différents processus de routage dynamiques, un par protocole. Pour chaque processus (ripngd, ospf6d, bgpd) il faut d'abord créer un fichier de configuration propre, qui contient les règles d'annonce, d'import et d'export de routes, et les contrôles (mot de passe et filtres de telnet par exemple). Il faut bien entendu s'assurer que le démon pour RIPng (route6d) n'est pas lancé (`ipv6_router_enable=NO`). Une fois les fichiers de configuration créés, la commande `zebractl restart` lancera tous les démons.

Voici deux exemples de configuration minimales pour RIPng et pour OSPFv3 ; des prototypes sont fournis avec la distribution Zebra dans `/usr/local/etc/zebra` :

```
>cat /usr/local/etc/zebra/ripng.conf
! Zebra ripngd configuration
!
hostname Router
password 8 xxxxxxxxx
enable password 8 xxxxxxxxxx
service password-encryption
!
router ripng
network xl0
network gif1
>cat /usr/local/etc/zebra/ospf6d.conf
! Zebra ospfv3 configuration
!
hostname Router
password zebra
!
router ospf6
router-id 0.0.0.1
interface xl0 area 200.1.0.0
```

D) Configuration BGP4+

Dans l'exemple figure Session BGP sous Zera, le routeur thieste appartenant à l'AS 65400 établi une session BGP4+ avec le routeur horace de l'AS 1938. Sur la machine thieste où le processus zebra a été activé lors des exemples précédents, il faut maintenant activer le processus bgpd après avoir installé son fichier de configuration (`bgpd.conf`). Cette opération est réalisée par la séquence de commandes suivante :

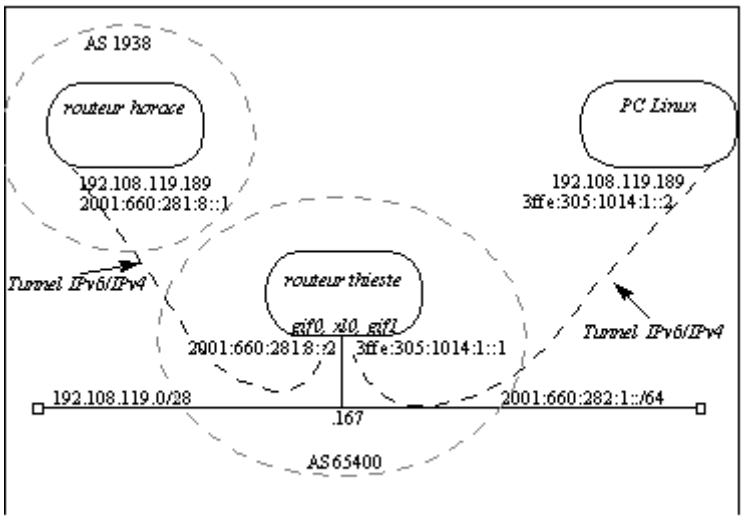


Figure 10-3. Session BGP sous Zebra

```
>cp /usr/local/etc/zebra/bgpd.conf.sample /usr/local/etc/zebra/bgpd.conf
>zebractl restart
>telnet localhost bgpd
```

La dernière commande permet de contacter le processus `bgpd` pour le configurer interactivement comme on le ferait avec un routeur dédié. La configuration correspondant à l'exemple ci-dessous est listée par la commande `write` :

```
bgpd# write t

Current configuration:
!
hostname bgpd
password xxxxxx
log stdout
!
router bgp 65400
bgp router-id 192.108.119.167
ipv6 bgp neighbor 2001:660:281:8::1 remote-as 1938
!
line vty
!
End
```

On peut avoir un état sommaire des sessions BGP4+ par une commande `show` :

```
bgpd# show ipv6 bgp summary
BGP router identifier 192.108.119.167, local AS number 65400
175 BGP AS-PATH entries
0 BGP community entries

Neighbor AS MsgRcvd MsgSent Up/Down State/PfxRcd
2001:660:281:8::1 1938 209 26 00:15:36 264

Total number of neighbors 1
```

Ou avoir plus d'informations sur les sessions BGP par :

```
bgpd# show ipv6 bgp neighbors
BGP neighbor is 2001:660:281:8::1, remote AS 1938, external link
BGP version 4, remote router ID 131.254.200.10
BGP state = Established, up for 00:14:50
Last read 00:00:49, hold time is 180, keepalive interval is 60 seconds
```

```
Neighbor capabilities:
Route refresh: advertised and received(old and new)
Address family IPv4 Unicast: advertised and received
Received 200 messages, 8 notifications, 0 in queue
Sent 25 messages, 0 notifications, 0 in queue
Route refresh request: received 0, sent 0
Minimum time between advertisement runs is 0 seconds

For address family: IPv6 Unicast
Community attribute sent to this neighbor
264 accepted prefixes

Connections established 1; dropped 0
Local host: 2001:660:281:8::2, Local port: 179
Foreign host: 2001:660:281:8::1, Foreign port: 11681
Nexthop: 192.108.119.167
Nexthop global: 2001:660:281:8::2
Nexthop local: ::
BGP connection: non shared network
Read thread: on Write thread: off
```

La liste des préfixes et leurs attributs est donnée par :

```
bgpd# show ipv6 bgp
BGP table version is 0, local router ID is 192.108.119.167
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

Network Metric LocPrf Weight Path
*> ::194.182.135.0/120 0 1938 2200 1103 766 278 6435 6939 513 8933 2852 3263 i
2001:660:281:8::1(fe80::83fe:c80a)
*> 2001:200::/35 0 1938 2200 3425 2500 i
2001:660:281:8::1(fe80::83fe:c80a)
*> 2001:200:12a::/48 0 1938 2200 5511 3549 ?
2001:660:281:8::1(fe80::83fe:c80a)
*> 2001:200:600::/40 0 1938 2200 5511 7667 i
[....]
*> 3ffe:8240::/28 0 1938 2200 1103 8954 109 6342 i
2001:660:281:8::1(fe80::83fe:c80a)

Total number of prefixes 264
```

E) Filtrage d'annonce BGP4+

Pour terminer cet exemple de configuration Zebra, on installe trois filtres sur les annonces de préfixes reçus par `thieste` en provenance de son voisin BGP `horace`.

Le filtre nommé `filtre_nlri` porte sur les préfixes eux-même, et rejette les annonces de préfixes contenus dans `3ffe:305:1014::/48` ou dans `2002::/16`.

Le filtre nommé `filtre_as` porte sur les chemins d'AS, il rejette les annonces de préfixes dont l'attribut `as_path` commence par la séquence d'AS : `1938 2200 5511`.

Le filtre nommé `filtre_map` modifie l'attribut `local_pre`. Ce dernier prend la valeur 200 si l'attribut `as_path` du préfixe annoncé commence par la séquence d'AS : `1938 2200 1103`. Dans tous les autres cas l'attribut `local_pref` prend la valeur 100.

La partie concernant BGP du fichier de configuration devient :

```
router bgp 65400
bgp router-id 192.108.119.167
ipv6 bgp neighbor 2001:660:281:8::1 remote-as 1938
ipv6 bgp neighbor 2001:660:281:8::1 prefix-list filtre_nlri in
ipv6 bgp neighbor 2001:660:281:8::1 filter-list filtre_as in
ipv6 bgp neighbor 2001:660:281:8::1 route-map filtre_map in
!
ipv6 prefix-list filtre_nlri description on ne veut pas se faire annoncer son reseau ni
le  2002::/16
ipv6 prefix-list filtre_nlri seq 5 deny 3ffe:305:1014::/48 le 128
ipv6 prefix-list filtre_nlri seq 10 deny 2002::/16 le 128
ipv6 prefix-list filtre_nlri seq 15 permit any
!
ip as-path access-list filtre_as deny 1938 2200 5511 *
ip as-path access-list filtre_as permit .*
!
ip as-path access-list as_prefere permit 1938 2200 1103 *
ip as-path access-list as_prefere deny .*
!
route-map filtre_map permit 10
match as-path as_prefere
set local-preference 200
!
route-map filtre_map permit 20
set local-preference 100
```

XI) Multicast

Une communication multicast est une communication dans laquelle un même paquet de données peut être envoyé à un groupe de récepteurs, quelque soit leur localisation. Dans le modèle Internet IPv6, une station peut potentiellement émettre un paquet multicast vers n'importe quel groupe. Comparé aux communications point à point (unicast), le multicast évite la duplication des paquets de données au niveau de la source, et minimise l'utilisation de la bande passante au niveau du réseau. De plus, il offre un service insensible à l'augmentation du nombre et la localisation des membres d'un groupe. Le multicast peut être utilisé pour la distribution de logiciels, la téléconférence, les applications d'enseignement à distance, la radio ou la télévision sur Internet, les simulations interactives distribuées, les jeux multimédia interactifs, les applications militaires, etc. Ce chapitre insistera sur les différences par rapport au multicast IPv4, tout en donnant une vue d'ensemble des protocoles mis en jeu.

Pour le multicast, on distingue deux modèles de communication : le modèle ASM (*Any-Source Multicast*) et le modèle SSM (*Source-Specific Multicast*). Dans le modèle ASM, un récepteur s'abonne à un groupe, et reçoit les données émises par n'importe quelles sources pour ce groupe. Ce modèle s'applique par exemple dans le cas de visioconférences avec de nombreux participants qui ne sont pas connus à l'avance. Dans le modèle SSM, les sources sont connues à l'avance et les récepteurs s'abonnent à un groupe et un ensemble de sources. Ce modèle s'applique par exemple à la diffusion de la télévision ou radio sur Internet, où il n'y a qu'une seule source connue de tous.

Les étapes suivantes interviennent dans l'établissement d'une session multicast IPv6 :

- Choix de l'adresse multicast pour la session. Dans cette section sont aussi présentés les mécanismes permettant l'allocation des adresses multicast.
- Description et annonce de la session multicast à tous les participants :
    - Gestion des membres du groupe sur le lien-local : Elle est réalisée par le protocole MLD (*Multicast Listener Discovery*).
    - Construction de l'arbre multicast : Elle est assurée par le protocole PIM (*Protocol Independant Multicast*).

La section, Multicast IPv6 inter-domaine, de ce chapitre est consacrée au multicast inter-domaine IPv6. L'état actuel du déploiement du multicast IPv6 est ensuite sommairement décrit. Les deux sections suivantes portent sur les applications multicast IPv6 et la coexistence avec le multicast IPv4. La dernière section présente un cas pratique.

1) Adressage multicast

Pour initier une session multicast, le groupe de récepteurs intéressés, appelé aussi groupe multicast, doit être formé. Un groupe multicast est identifié par une adresse IP multicast. Chaque adresse a une portée spécifique, qui limite la propagation du trafic multicast.

Dans ce chapitre, nous commençons par détailler le format des adresses multicast IPv6 Nous examinons ensuite successivement l'allocation des adresses multicast IPv6 puis l'annonce des sessions.

Contents

A) Format des adresses multicast IPv6

a) Généralités

Cette section décrit le système d'adressage multicast IPv6. La figure Structure de l'adresse IPv6 Multicast donne le format de l'adresse IPv6 de multicast décrite dans le RFC 3513.

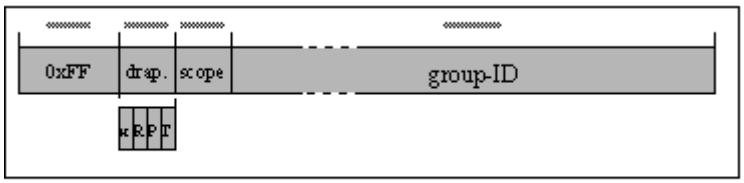


Figure 11-1 . Structure de l'adresse IPv6 Multicast

Les adresses multicast IPv6 sont dérivées du préfixe `FF00::/8`. Le champ drapeaux de 4 bits est défini de la manière suivante :

- Seul le bit `T` (comme *Transient*) du champ drapeaux est initialement décrit dans le RFC 3513. La valeur `0` indique une adresse multicast bien connue gérée par une autorité. La valeur `1` indique une valeur temporaire.
- Les bits `P` et `R` sont décrits dans le RFC 3306 et le draft Internet sur embedded-RP (RFC 3956).
- Le bit de poids fort du champ drapeaux n'est pas encore attribué.

Le champ drapeaux permet de définir plusieurs types d'adresses multicast IPv6 qui seront décrits dans les sections suivantes.

Le champ scope de l'adresse multicast IPv6 permet d'en limiter la portée (*scope* en anglais). En IPv4, la portée d'un paquet est limitée par le champ TTL (*Time To Live*), de même des préfixes peuvent être définis pour identifier des adresses à portée réduite. Les valeurs suivantes sont définies :

- 1 - node-local
- 2 - link-local
- 3 - subnet-local
- 4 - admin-local
- 5 - site-local
- 8 - organisation-local
- E - global
- Les portées 0 et F sont réservés.

b) Adresses multicast IPv6 permanentes

Une adresse multicast IPv6 avec le bit `T` du champ drapeaux à `0` correspond à une adresse multicast permanente, allouée par l'IANA.

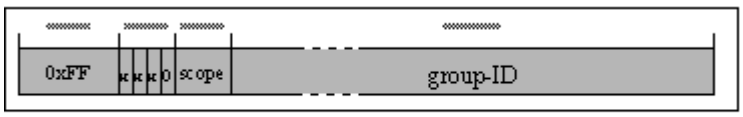


Figure 11-2 . Structure des adresses IPv6 Multicast permanentes

Lorsque le multicast IPv6 sera déployé à grande échelle, certains organismes pourraient avoir des émissions permanentes. Des chaînes de télévision ou stations de radio pourront par exemple se voir attribuer des adresses permanentes par l'IANA dans le préfixe `FF00::/12`.

Le RFC 2375 définit déjà certaines adresses IPv6 multicast. Deux types d'adresses multicast permanentes sont à distinguer :

- des adresses correspondant à des services de niveau réseau (comme NTP, DHCPv6, cisco-rp-announce, SAP,...) et
- des adresses correspondant d'avantage à des services applicatifs commerciaux permanents comme la distribution des chaînes de télévision. Le RFC 3307 définit des procédures pour l'allocation des adresses multicast permanentes. Celles-ci seront décrites par la suite.

c) Adresses temporaires

Les adresses temporaires sont des adresses multicast IPv6 dont le bit T est positionné à 1. Il existe plusieurs types d'adresses temporaires :

- générales : Ce sont des adresses avec tous les bits du champ flag à 0 sauf le bit T positionné à 1. Il n'y a pas de recommandations pour l'utilisation de ces adresses. Des scénarios d'utilisation peuvent être, par exemple, les visioconférences ponctuelles.
- dérivées d'un préfixe unicast IPv6. Le RFC 3306 définit une méthode pour dériver une adresse multicast IPv6 à partir d'un préfixe unicast :

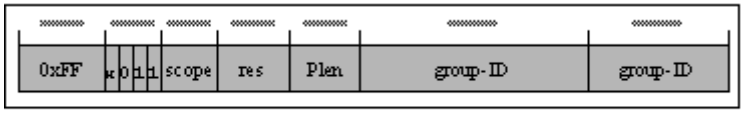


Figure 11-3 : Structure d'une adresse IPv6 Multicast dérivée d'un préfixe unicast

-
  - `res` (*reserved*) : tous les bits de ce champ doivent être positionnés à `0`.
  - `Plen` (*prefix length*) : ce champ contient la longueur du préfixe unicast utilisé pour en dériver une adresse multicast.
  - `prefix` : ce champ contient la valeur du préfixe du réseau utilisé pour en dériver une adresse multicast.
  - `group-ID` : ce champ de 32 bits contient l'identifiant de groupe, détaillé au chapitre Identifiant de groupe.

Par exemple, une adresse multicast peut être dérivée à partir du préfixe de RENATER (`2001:660::/32`). Le champ `prefix` prend la valeur `2001:0660:0000:0000` et le champ `Plen`, la valeur `0x20` (32 en décimal). Les adresses multicast IPv6 à choisir seront de type `FF3X:20:2001:660::aabb:ccdd` (`aabb:ccdd` étant le group-ID choisi dans l'exemple).

Cette méthode permet la création potentielle de 232 adresses par préfixe.

- les adresses multicast "Embedded-RP" voir le RFC 3618 définit une méthode pour inclure l'adresse du RP (Point de Rendez-Vous servant à la construction de l'arbre multicast) dans l'adresse multicast IPv6. Le schéma Structure d'une adresss IPv6 Multicast "embedded RP" montre la structure d'une telle adresse, aussi appelée adresse "embedded-RP" :

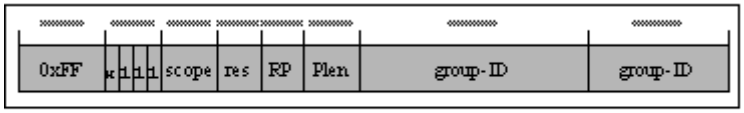


Figure 11-4 . Structure d'une adresss IPv6 Multicast "embedded RP"

Ainsi pour un point de rendez-vous qui possède l'adresse `2001:660:3307:125::3`, une adresse multicast correspondante peut être dérivée de la façon suivante :

- `res` (Reservé) : Les 4 bits de ce champ sont positionnés à `0`.
- `RPad` : Ce champ contient les 4 derniers bits de l'adresse du RP. Dans cet exemple, `RPad` prend la valeur 3.
- `Plen` (Longueur du préfixe) : Ce champ contient la longueur du préfixe réseau du RP à prendre en compte. Dans cet exemple, la valeur est de `0x40` (soit 64 en décimal),
- `prefix` (Préfixe) : Ce champ contient le préfixe réseau du RP. Ici, cette valeur est `2001:660:3007:125`
- `group-ID` : ce champ de 32 bits contient l'identifiant de groupe, détaillé au chapitre Identifiant de groupe.

Une adresse multicast dérivée de ce point de rendez-vous sera donc de la forme `FF7X:340:2001:660:3007:125:aabb:ccdd` (`aabb:ccdd` étant le group-ID choisi dans cet exemple).

- Les adresses SSM (*Source Specific Multicast*) sont décrites également dans le RFC 3306. Si le préfixe `FF3X::/32` a été réservé pour les adresses multicast SSM, seules les adresses dérivées du préfixe `FF3X::/96` doivent être utilisées dans un premier temps. Ce sont des adresses multicast basées sur le préfixe unicast où les champs `Plen` et `prefix` sont positionnés à 0 (cf. figure Structure des adresses IPv6 Multicast SSM).

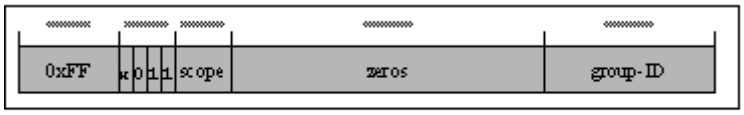


Figure 11-5 . Structure des adresses IPv6 Multicast SSM

Le tableau Récapitulatif des types d'adresses multicast définis récapitule les préfixes associés aux différents types d'adresses multicast décrit précédemment.

Récapitulatif des types d'adresses multicast définis	
Préfixe	**Usage**
FF0X::/16	Adresses IPv6 multicast permanentes
FF1X::/16	Adresses IPv6 multicast temporaires générales
FF3X::/16	Adresses multicast dérivées d'un préfix unicast (temporaires)
FF3X::/96	Adresses SSM (temporaires)
FF7X::/16	Adresses IPv6 multicast "Embedded-RP" (temporaires)

d) Identifiant de groupe

Le RFC 3307 décrit des procédures de création d'un identifiant de groupe (`Group-ID`) et le RFC 3513 fixe la taille du champ `Group-ID` à 112 bits. Le RFC 3307 précise également la correspondance entre les adresses IPv6 multicast et les adresses de niveau 2 : les 32 derniers bits de l'adresse multicast IPv6 sont ajoutés au préfixe MAC 33-33.

Par exemple, l'adresse FF0E:30:2001:660:3001:4002:**AE45:2C56** correspondra à l'adresse MAC 33-33-**AE-45-2C-56**. La probabilité que deux adresses multicast IPv6 utilisées sur un même lien correspondent à la même adresse MAC existe mais est très faible et les conséquences minimes. Restreindre le champ group-ID à 32 bits a toutefois un intérêt car cela apporte une homogénéité entre les différents types d'adresses décrits précédemment. En effet, dans le cas des adresses dérivées d'un préfixe unicast, ce champ a une longueur de 32 bits.

Le RFC 3307 définit aussi les adresses IPv6 multicast et identifiants de groupe qui seront gérés par l'IANA, où réservés pour des allocations dynamiques.

Récapitulatif des usages des identifiants de groupe			
	Description	**Valeur minimale de l'identifiant de groupe**	**Valeur maximale de l'identifiant de groupe**
Adresse multicast permanente	C'est une adresse allouée par l'organisme IANA. Les bits P et T doivent être initialisés à zéro.	0x00000001	0x3FFFFFFF
Identifiant de groupe permanent	Le but de ces identifiants de groupe est de pouvoir identifier un service donné dans un réseau. Ces services sont définis par des Group-ID alloués par l'IANA et devraient être utilisées pour des adresses IPv6 multicast dérivées d'un préfixe unicast (RFC 3306). Avec cette méthode, il est théoriquement possible d'atteindre un service donné dans n'importe quel réseau.	0x40000000	0x7FFFFFFF
Adresse multicast dynamique	Les adresses multicast allouées dynamiquement doivent avoir un group-ID compris entre 0x80000000 et 0xFFFFFFFF. Ces adresses ont le bit T du champ drapeaux positionné à 1.	0x80000000	0xFFFFFFFF

2) Le multicast IPv6 sur le lien-local

A) Gestion des abonnements sur le lien-local : MLD

Pour offrir un service de distribution multicast, deux composants sont nécessaires : un protocole de gestion de groupe multicast et un protocole de construction d'arbre multicast. Le protocole de gestion de groupe multicast réalise la signalisation entre l'hôte et son routeur d'accès à l'Internet. En IPv6, ce protocole est MLD (*Multicast Listener Discovery*). Il est utilisé par un routeur de bordure IPv6 pour découvrir la présence de récepteurs multicast sur ses liens directement attachés, ainsi que les adresses multicast concernées.

MLD est un protocole asymétrique qui spécifie un comportement différent pour les hôtes et les routeurs multicast. Toutefois, pour les adresses multicast sur lesquelles un routeur lui-même écoute, il doit exécuter les deux parties du protocole et répondre à ses propres messages.

Comme MLD est un sous-protocole d'ICMPv6, les messages MLD sont des messages ICMPv6 particuliers. Ils sont envoyés avec :

- une adresse source IPv6 lien-local ;

- le champ "nombre de sauts" fixé à 1 ;
- l'option "IPv6 Router Alert" activée.

Cette dernière option est nécessaire afin de contraindre les routeurs à examiner les messages MLD envoyés à des adresses multicast par lesquelles les routeurs ne sont pas intéressés. La version d'origine du protocole MLD (RFC 2710) (que nous appellerons également MLDv1) présente les mêmes fonctionnalités que le protocole IGMPv2 en IPv4.

Trois types de messages sont utilisés. Leur format est donné sur la figure Format générique d'un message ICMP pour MLD:

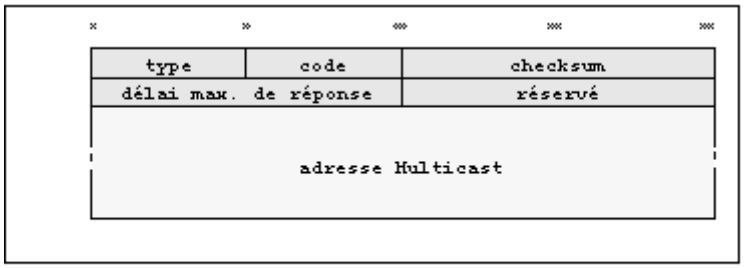


Figure 11-6. Format générique d'un message ICMP pour MLD

- recensement des récepteurs multicast (type = 130) avec deux sous-types de messages :
- recensement général émis à l'adresse de diffusion générale sur le lien (`FF02::1`)
- recensement spécifique à une adresse multicast, l'adresse de destination est l'adresse multicast du groupe en question
- rapport d'abonnement multicast (type = 131), l'adresse de destination est l'adresse multicast du groupe en question
- résiliation d'abonnement multicast (type = 132), émis à l'adresse du groupe multicast "tous les routeurs du lien local" (`FF02::2`).

Les champs ont la signification suivante :

- type : prend la valeur 130, 131 ou 132.
- code : mis à zéro par l'émetteur et ignoré par les récepteurs
- checksum : celui du protocole ICMPv6 standard, couvrant tout le message MLD auquel s'ajoutent les champs du pseudo-en-tête IPv6
- délai maximal de réponse :
    - utilisé seulement dans les messages de recensement. Il exprime le retard maximal autorisé (en millisecondes) pour l'arrivée des rapports d'abonnement
    - dans les messages de rapport ou de résiliation d'abonnement ce champ est mis à zéro par l'émetteur et ignoré par les récepteurs
- inutilisé : mis à zéro par l'émetteur et ignoré par les récepteurs
- adresse multicast :
    - pour un message de recensement général ce champ est mis à zéro
    - pour un message de recensement spécifique il contient l'adresse multicast en question
    - pour les messages de rapport et de résiliation d'abonnement, le champ contient l'adresse multicast sur laquelle l'hôte souhaite écouter ou cesser d'écouter

a) Messages de recensement et rapports d'abonnement périodiques MLD

Le routeur envoie régulièrement des messages de recensement général à l'adresse de diffusion générale sur le lien (`FF02::1`). Les hôtes arment un temporisateur pour chaque adresse multicast qui les concerne. Si un temporisateur expire sans que l'hôte ait entendu une réponse d'un de ses voisins concernant la même adresse, il envoie un rapport d'abonnement à l'adresse multicast du groupe. Ce système de temporisateurs permet aux hôtes de surveiller les rapports des autres hôtes sur le lien et d'annuler leurs propres rapports concernant les mêmes adresses. Ainsi la quantité du trafic MLD peut être minimisée.

b) Rapports d'abonnements MLD non-sollicités

Les changements d'état des hôtes sont notifiés par des messages non-sollicités :

- Pour souscrire à une adresse multicast spécifique, un hôte envoie un rapport d'abonnement non-sollicité ;
- Pour cesser d'écouter sur une adresse multicast, l'hôte peut simplement ne plus répondre aux messages de recensement du routeur. S'il est le seul récepteur de cette adresse multicast sur le lien, après un certain temps l'état du routeur concernant cette adresse expire. Le routeur arrêtera de faire suivre les paquets multicast envoyés à l'adresse en question, s'il s'avère que l'hôte était le dernier concerné par l'adresse multicast sur le lien;
- La résiliation rapide est aussi une possibilité offerte par MLDv1. L'hôte envoie un message de résiliation d'abonnement à l'adresse multicast de "tous les routeurs du lien local" (`FF02::2`). Le routeur répond avec un message de recensement spécifique à l'adresse en question. S'il n'y a plus de récepteur pour répondre à ce recensement, le routeur efface l'adresse multicast de sa table de routage.

Il est possible d'avoir plusieurs routeurs multicast sur le même lien local. Dans ce cas un mécanisme d'élection est utilisé pour choisir le routeur recenseur. Celui-ci sera le seul responsable pour l'envoi des messages de recensement.

c) Exemples de fonctionnement de MLDv1

Les paquets suivants ont été capturés lors de l'exécution d'un programme (multi2out6, dont le code est donné dans le chapitre Utilisation du multicast). Ce programme prend comme arguments une interface de la machine et une adresse multicast. Dans cet exemple, l'adresse choisie `ff12::1234:5678`, représente un groupe éphémère (valeur 0x1 du drapeau) sur le lien local (valeur `0x02`).

L'interface se joint à ce groupe multicast et commence par émettre un rapport d'abonnement :

```
En-tête IPv6
Version : 6 Classe : 00 Label : 00000
Longueur : 32 octets (0x0020) Proto. : 0 (0x0) "Proche-en-proche"
```

```
Nombre de sauts : 1
Source : fe80::0a00:20ff:fe18:964c
Desti. : ff12::1234:5678 (adresse du groupe multicast)
Proche-en-proche
En-tête Suivant : 58 (0x3a) ICMPv6/MLD
Type : 5 (0x5) Router Alert longueur : 2 valeur : 0
ICMPv6/MLD
Type : 131 (0x83) rapport d'abonnement
Code : 0
Checksum : 0xef48
Délai maximal de réponse : 0
Adresse multicast : ff12::1234:5678 (adr du grp multicast en question)

0000: 60 00 00 00 00 20 00 01 fe 80 00 00 00 00 00 00
0010: 0a 00 20 ff fe 18 96 4c ff 12 00 00 00 00 00 00
0020: 00 00 00 00 12 34 56 78 3a 00 05 02 00 00 00 00
0030: 83 00 ef 48 00 00 00 00 ff 12 00 00 00 00 00 00
0040: 00 00 00 00 12 34 56 78
```

En arrêtant le programme, l'interface en question se désabonne du groupe multicast et en s'apercevant qu'elle est la dernière à avoir envoyé un rapport concernant ce groupe, elle émet un message de fin d'abonnement :

```
En-tête IPv6
Version : 6 Classe : 00 Label : 00000
Longueur : 32 octets (0x0020) Proto. : 0 (0x0) "Proche-en-proche"
Nombre de sauts : 1
Source : fe80::0a00:20ff:fe18:964c
Desti. : ff12::1234:5678
Proche-en-proche
En-tête Suivant : 58 (0x3a) ICMPv6/MLD
Type : 5 (0x5) Router Alert longueur : 2 valeur : 0
ICMPv6/MLD
Type : 132 (0x84) Fin d'abonnement
Code : 0
Checksum : 0x5703
Délai maximal de réponse : 0
Adresse multicast : ff12::1234:5678 (adr du grp multicast en question)

0000: 60 00 00 00 00 20 00 01 fe 80 00 00 00 00 00 00
0010: 0a 00 20 ff fe 18 96 4c ff 02 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 02 3a 00 05 02 00 00 00 00
0030: 84 00 57 03 00 00 00 00 ff 12 00 00 00 00 00 00
0040: 00 00 00 00 12 34 56 78
```

B) Gestion des abonnements sur le lien-local : MLD v2

La nouvelle version du protocole de gestion de groupe multicast, MLDv2 est décrite dans le RFC 3810. Elle implante les fonctionnalités du protocole IGMPv3 défini pour IPv4, la plus importante étant l'introduction du filtrage des sources. Un hôte peut désormais spécifier les sources qu'il veut ou qu'il ne veut pas écouter pour une adresse multicast donnée. Cette information peut être utilisée par les protocoles de routage multicast afin d'éviter l'acheminement des paquets multicast provenant de certaines sources vers des liens où il n'y a pas de récepteur intéressé.

Pour être en mesure de supporter les fonctionnalités de MLDv2, l'API de l'hôte doit permettre l'opération suivante (ou un équivalent logique de celle-ci) :

```
EcouteIPv6Multicast (socket, interface,
 adresse multicast IPv6, mode de filtrage,
 liste de sources)
```

Par cet appel, une application demande, pour une certaine adresse multicast, la réception de paquets sur une certaine interface, en tenant compte du mode de filtrage et de la liste des sources spécifiées. Le mode de filtrage peut être soit `INCLUDE`, soit `EXCLUDE` :

- En mode `INCLUDE`, la réception des paquets envoyés à l'adresse multicast spécifiée est demandée seulement pour ceux en provenance des sources présentes dans la liste qui suit
- En mode `EXCLUDE`, la réception des paquets est demandée pour toutes les sources, à l'exception de celles spécifiées dans la liste de sources

Il existe deux types de messages MLDv2 :

- recensement des récepteurs multicast (type=130)
- rapport d'abonnement multicast version 2 (type=143)

Pour garder l'interopérabilité avec la version précédente de MLD, les messages de rapport d'abonnement multicast version 1 et de résiliation d'abonnement multicast sont également supportés.

a) Messages de recensement MLDv2

Un message de recensement des récepteurs en MLDv2 est donné sur la figure Format d'un message de recensement MLDv2. Les champs ont la signification suivante :

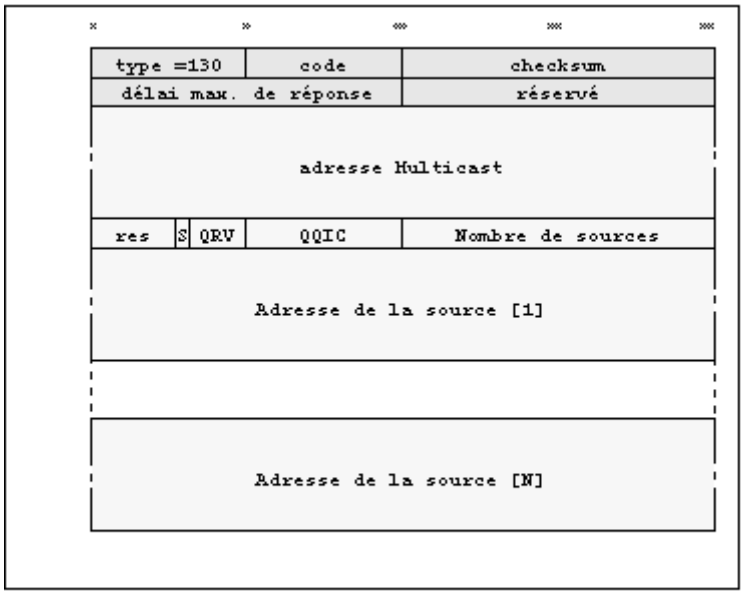


Figure 11-7. Format d'un message de recensement MLDv2

- `type` : le même type qu'en MLDv1
- `code` : mis à zéro par l'émetteur et ignoré par les récepteurs
- `checksum` : calculé de la même façon que pour la version précédente du protocole

- `délai max. de réponse` : utilisé pour calculer le délai maximal de réponse durant lequel le récepteur doit envoyer éventuellement son rapport d'abonnement
- `inutilisé` : mis à zéro par l'émetteur et ignoré par les récepteurs,
- `adresse multicast`,
- `réservé` : mis à zéro par l'émetteur et ignoré par les récepteurs,
- drapeau `S` : indique aux routeurs multicast qui reçoivent ce message s'ils doivent ou pas supprimer la mise à jour des temporisateurs, effectuée normalement au moment de la réception d'un message de recensement,
- `QRV` : contient la variable de robustesse utilisée par le recenseur (le nombre de fois qu'un récepteur envoie un rapport pour être robuste aux pertes dans le réseau),
- `QQIC` : code utilisé pour calculer l'intervalle de recensement,
- `nombre de sources`,
- `adresse de la source [N]`, vecteur contenant la liste éventuelle des sources.

Trois types de messages de recensement de récepteurs multicast sont utilisés :

- recensement général envoyé par un routeur multicast afin de découvrir les adresses multicast pour lesquelles il y a des récepteurs sur ses liens directs. Dans un tel message les champs "adresse multicast" et "nombre de sources" sont mis à zéro
- recensement spécifique à une adresse multicast envoyé par un routeur multicast afin de découvrir l'existence de récepteurs pour une adresse multicast spécifique. Le champ "adresse multicast" contient l'adresse en question, tandis que le champ "nombre de sources" est mis à zéro
- recensement spécifique à une adresse multicast et à une source envoyé par un routeur multicast afin de découvrir l'existence de récepteurs pour une adresse multicast et une source spécifiques. Le champ "adresse multicast" contient l'adresse en question, tandis que les champs "adresse source [i]" forment un vecteur de N adresses unicast (valeur spécifiée dans le champ "nombre de sources").

Les messages de recensement général sont envoyés à l'adresse de diffusion générale sur le lien (`FF02::1`). Les autres messages de recensement sont envoyés à l'adresse multicast spécifiée dans l'en-tête MLDv2.

b) Rapports d'abonnement MLDv2

Un rapport d'abonnement multicast en MLDv2 est donné sur la figure Format d'un message de rapport d'abonnement MLDv2. Les champs ont la signification suivante :

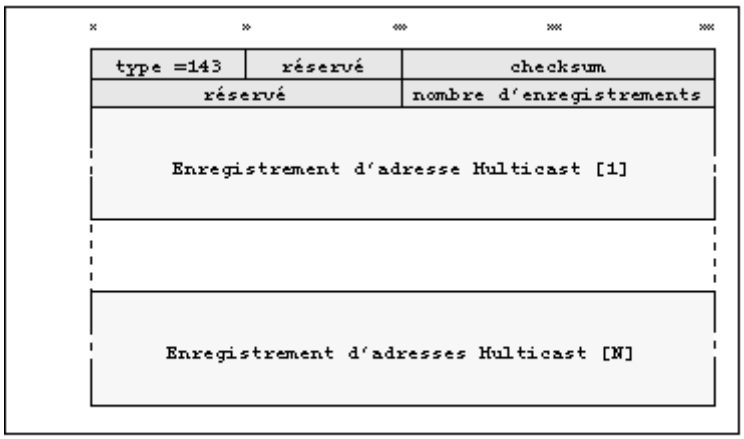


Figure 11-8 . Format d'un message de rapport d'abonnement MLDv2

- `type`, type=143
- `réservés`, mis à zéro par l'émetteur et ignorés par les récepteurs
- `checksum`, calculé de la même façon que pour la version précédente du protocole
- `nombre d'enregistrements d'adresse multicast`
- `enregistrement d'adresse multicast` : chaque enregistrement d'adresse multicast a la forme donnée sur la figure Format d'un message de recensement MLDv2 : Les champs ont la signification suivante :

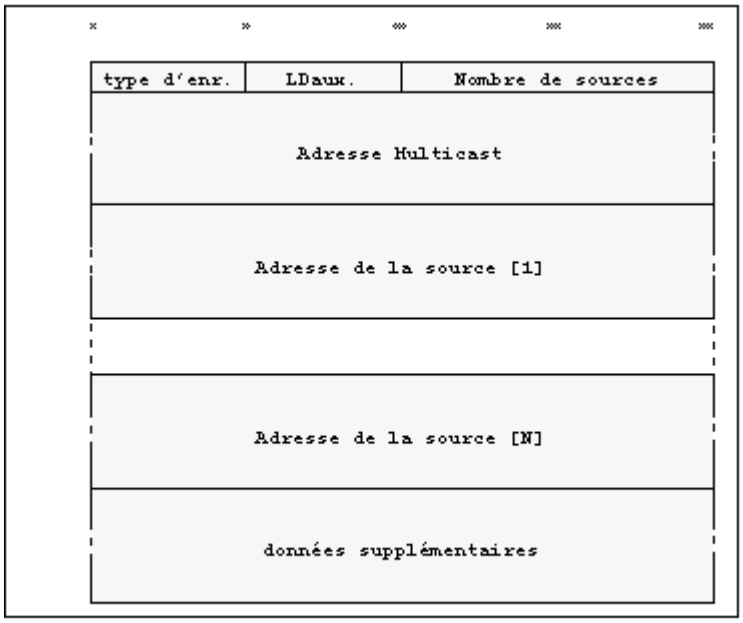


Figure 11-9 . Format d'un message de recensement MLDv2

  - type d'enregistrement : plusieurs types d'enregistrements d'adresse multicast peuvent être inclus dans un rapport d'abonnement :
    - un "enregistrement d'état actuel" est envoyé par un hôte en réponse à un message de recensement. Il décrit l'état de l'hôte concernant une adresse multicast spécifique .Le champ "type d'enregistrement" peut dans ce cas avoir les valeurs `MODE_IS_INCLUDE` ou `MODE_IS_EXCLUDE`,
    - un "enregistrement de changement de mode de filtrage" est envoyé par un hôte chaque fois qu'un appel `EcouteIPv6Multicast` modifie son mode de filtrage pour

une adresse multicast précise. Le champ "type d'enregistrement" peut dans ce cas avoir les valeurs `CHANGE_TO_INCLUDE_MODE` ou `CHANGE_TO_EXCLUDE_MODE`,

- un "enregistrement de changement de la liste des sources" est envoyé par un hôte quand un appel `EcouteIPv6Multicast` modifie la liste des sources qu'il souhaite ou ne souhaite pas écouter pour une adresse multicast précise. Le champ "type d'enregistrement" peut dans ce cas avoir les valeurs `ALLOW_NEW_SOURCES` ou `BLOCK_OLD_SOURCES`.

  - `LDAux` contient la longueur du champ données supplémentaires
- `adresse multicast`
- `nombre de sources`
- `adresse source [i]`, vecteur contenant la liste des sources qui doivent être désormais autorisées ou bloquées
- `données supplémentaires`, si elles sont présentes, peuvent contenir des informations supplémentaires concernant l'enregistrement d'adresse multicast en question.

Les rapports d'abonnement sont envoyés par les hôtes à l'adresse "tous les routeurs MLDv2" (`ff02::16`). Ainsi les récepteurs ne reçoivent pas les rapports des autres, chacun étant obligé d'envoyer son propre rapport. Le mécanisme d'économie d'émission des rapports du protocole MLDv1 n'est donc plus utilisé. Le risque de surcharge du routeur à cause du trop grand nombre de rapports reçus est toutefois évité en fixant des temporisateurs avec des valeurs différentes pour chaque rapport.

c) Fonctionnement de MLDv2

Comme dans le cas du MLDv1, s'il existe plusieurs routeurs multicast sur le même lien local, un seul routeur recenseur va être désigné à l'aide d'un mécanisme spécifique. Le recenseur envoie régulièrement des messages de recensement général auxquels les récepteurs répondent avec des rapports d'abonnement contenant des enregistrements d'état actuel.

Chaque hôte, ainsi que chaque routeur, gardent un état contenant le mode de filtrage et une liste des sources pour chaque adresse multicast. Dans le cas d'un hôte ce sont les adresses multicast sur lesquelles il écoute. Dans le cas d'un routeur ce sont les adresses multicast que ses récepteurs écoutent. Une application sur une machine hôte demande la modification du mode de filtrage ou de la liste des sources pour une adresse multicast précise à travers d'un appel `EcouteIPv6Multicast`. L'hôte inclut par conséquent ces modifications dans un rapport d'abonnement non-sollicité. Ce rapport contient des enregistrements de changement de mode de filtrage ou de la liste des sources, en conformité avec les changements qui ont eu lieu dans l'état interne de l'hôte.

Au moment de la réception des changements communiqués, le routeur met à jour son état pour l'adresse multicast concernée. Si, suite à ces changements, il constate qu'une certaine source ne doit plus être acceptée, le routeur envoie un message de recensement spécifique pour cette source, afin de vérifier l'existence d'éventuels récepteurs qui souhaitent toujours l'écouter. Un temporisateur est déclenché, et s'il expire sans que le routeur ait reçu un rapport d'abonnement concernant la source, celle-ci est éliminée de l'état local du routeur. Si le routeur détecte que plus aucune source n'est sollicitée pour une certaine

adresse, il envoie un message de recensement spécifique pour cette adresse. Si des rapports d'abonnement concernant l'adresse en question ne sont pas reçus en temps dû, l'adresse est effacée de l'état du routeur.

d) Exemples de fonctionnement de MLDv2

Les exemples suivants illustrent le fonctionnement du protocole MLDv2.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 36 octets (0x0024)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::240:95ff:fe49:ba9
Adresse destination : ff02::1 (adresse de diffusion générale sur le lien)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
PadN : 0x01 Longueur : 0x00 (ce qui revient à 2 octets de bourrage)
Router alert : 0x05 Longueur : 0x02 Valeur : 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 130 (0x82) - message de recensement
Code : 0 (0x00)
Somme de contrôle : 0xb464
Code de réponse maximal : 10000 (Ox2710) Réservé : 0x0000
Adresse multicast : 0::0 Réservé : 0x0
Drapeau S : 0
QRV : 2
QQIC : 125 (0x7d)
Nombre de sources: 0 (il s'agit d'un recensement général)

0x0000 6000 0000 0024 0001 fe80 0000 0000 0000
0x0010 0240 95ff fe49 0ba9 ff02 0000 0000 0000
0x0020 0000 0000 0000 0001 3a00 0100 0502 0000
0x0030 8200 b464 2710 0000 0000 0000 0000 0000
0x0040 0000 0000 0000 0000 027d 0000
```

Le routeur recenseur envoie un message de recensement général.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 76 octets (0x004c)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::203:47ff:fe7c:b9c5
Adresse destination : ff02::16 (tous les routeurs MLDv2 sur le lien)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
PadN : 0x01 Longueur : 0x00 (ce qui revient a 2 octets de bourrage)
Router alert : 0x0502 Valeur: 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 143 (0x8f) - rapport d'abonnement
Réservé : 0x00
Somme de contrôle : 0x9454
Réservé : 0x0000
Nombre d'enregistrements : 0x0003
Type d'enregistrement : 0x02 (MODE_IS_EXCLUDE)
Longueur des données auxiliaires : 0x00
```

```
Nombre de sources : 0x0000
Adresse multicast : ff02::9
Type d'enregistrement : 0x02 (MODE_IS_EXCLUDE)
Longueur des données auxiliaires : 0x00
Nombre de sources : 0x0000
Adresse de la source : ff02::2:816a:9e88
Type d'enregistrement : 0x02 (MODE_IS_EXCLUDE)
Longueur des données auxiliaires : 0x00
Nombre de sources : 0x0000
Adresse multicast : ff02::1:ff7c:b9c5

0x0000 6000 0000 004c 0001 fe80 0000 0000 0000
0x0010 0203 47ff fe7c b9c5 ff02 0000 0000 0000
0x0020 0000 0000 0000 0016 3a00 0100 0502 0000
0x0030 8f00 9454 0000 0003 0200 0000 ff02 0000
0x0040 0000 0000 0000 0000 0000 0009 0200 0000
0x0050 ff02 0000 0000 0000 0000 0002 816a 9e88
0x0060 0200 0000 ff02 0000 0000 0000 0000 0001
0x0070 ff7c b9c5
```

Un hôte envoie un rapport d'abonnement avec des enregistrements d'état actuel.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 52 octets (0x0034)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::2e0:29ff:fe3e:db03
Adresse destination : ff02::16 (tous les routeurs MLDv2 sur le lien)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
PadN : 0x01 Longueur : 0x00 (ce qui revient a 2 octets de bourrage)
Router alert : 0x0502 Valeur: 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 143 (0x8f) - rapport d'abonnement
Réservé : 0x00
Somme de contrôle : 0x6b59
Réservé : 0x0000
Nombre d'enregistrements : 0x0001
Type d'enregistrement : 0x05 (ALLOW_NEW_SOURCES)
Longueur des données auxiliaires : 0x00
Nombre de sources : 0x0001
Adresse multicast : ff34::17
Adresse source : 2001:660:10d:4105:50:fcff:fe0b:9966

0x0000 6000 0000 0034 0001 fe80 0000 0000 0000
0x0010 02e0 29ff fe3e db03 ff02 0000 0000 0000
0x0020 0000 0000 0000 0016 3a00 0100 0502 0000
0x0030 8f00 6b59 0000 0001 0500 0001 ff34 0000
0x0040 0000 0000 0000 0000 0000 0017 2001 0660
0x0050 010d 4105 0050 fcff fe0b 9966
```

Un hôte rajoute une source dans la liste des sources qu'il veut écouter.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 52 octets (0x0034)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::2e0:29ff:fe3e:db03
Adresse destination : ff02::16 (tous les routeurs MLDv2 sur le lien)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
PadN : 0x01 Longueur : 0x00 (ce qui revient a 2 octets de bourrage)
```

```
Router alert : 0x0502 Valeur: 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 143 (0x8f) - rapport d'abonnement
Réservé : 0x00
Somme de contrôle : 0x6a59
Réservé : 0x0000
Nombre d'enregistrements : 0x0001
Type d'enregistrement : 0x06 (BLOCK_OLD_SOURCES)
Longueur des données auxiliaires : 0X00
Nombre de sources : 0x0001
Adresse multicast : ff34::17
Adresse source : 2001:660:10d:4105:50:fcff:fe0b:9966
0x0000 6000 0000 0034 0001 fe80 0000 0000 0000
0x0010 02e0 29ff fe3e db03 ff02 0000 0000 0000
0x0020 0000 0000 0000 0016 3a00 0100 0502 0000
0x0030 8f00 6a59 0000 0001 0600 0001 ff34 0000
0x0040 0000 0000 0000 0000 0000 0017 2001 0660
0x0050 010d 4105 0050 fcff fe0b 9966
```

Un hôte ne désire plus écouter une source donnée.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 52 octets (0x0034)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::240:95ff:fe49:ba9
Adresse destination : ff34::17 (adresse multicast concernée)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
PadN : 0x01 Longueur : 0x00 (ce qui revient a 2 octets de bourrage)
Router alert : 0x0502 Valeur: 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 130 (0x82) - message de recensement
Code : 0 (0x00)
Somme de contrôle : 0xdab1
Code de réponse maximal : 1000 (0x03e8)
Réservé : 0x0000
Adresse multicast : ff34::17
Réservé : 0x0
Drapeau S : 0
QRV : 2
QQIC : 125 (0x7d)
Nombre de sources : 0x0001
Adresse de la source : 2001:660:10d:4105:50:fcff:fe0b:9966

0x0000 6000 0000 0034 0001 fe80 0000 0000 0000
0x0010 0240 95ff fe49 0ba9 ff34 0000 0000 0000
0x0020 0000 0000 0000 0017 3a00 0100 0502 0000
0x0030 8200 dab1 03e8 0000 ff34 0000 0000 0000
0x0040 0000 0000 0000 0017 027d 0001 2001 0660
0x0050 010d 4105 0050 fcff fe0b 9966
```

Un routeur envoie un message de recensement spécifique à une adresse multicast et à une source.

```
En-tête IPv6 :
Version : 6 Classe de trafic : 0x00 Identifiant de flux : 0x00000
Longueur des données : 52 octets (0x0034)
En-tête suivant : extension proche-en-proche (0x00) Nombre de sauts : 0x01
Adresse source : fe80::2e0:29ff:fe3e:db03
Adresse destination : ff02::16 (tous les routeurs MLDv2 sur le lien)
Extension proche-en-proche :
En-tête suivant : ICMPv6 (0x3a)
Longueur : 0x00 (nombre de mots de 64 bits -1)
```

```
PadN : 0x01 Longueur : 0x00 (ce qui revient a 2 octets de bourrage)
Router alert : 0x0502 Valeur: 0x0000 (pour les messages MLD)
ICMPv6 :
Type: 143 (0x8f) - rapport d'abonnement
Réservé : 0x00
Somme de contrôle : 0x6c49
Réservé : 0x0000
Nombre d'enregistrements : 0x0001
Type d'enregistrement : 0x04 (CHANGE_TO_EXCLUDE_MODE)
Longueur des données auxiliaires : 0x00
Nombre de sources : 0x0001
Adresse multicast : ff44::17
Adresse source : 2001:660:10d:4105:50:fcff:fe0b:9966

0x0000 6000 0000 0034 0001 fe80 0000 0000 0000
0x0010 02e0 29ff fe3e db03 ff02 0000 0000 0000
0x0020 0000 0000 0000 0016 3a00 0100 0502 0000
0x0030 8f00 6c49 0000 0001 0400 0001 ff44 0000
0x0040 0000 0000 0000 0000 0000 0017 2001 0660
0x0050 010d 4105 0050 fcff fe0b 9966
```

Un hôte envoie un rapport d'abonnement contenant des enregistrements de changement de mode de filtrage.

C) MLD Fowarding Proxy

Afin d'éviter de déployer des routeurs multicast dans un domaine donné, l'IETF a récemment proposé l'utilisation de proxy MLD [Fenner-id]. Comme le montre la figure Gestion de groupe avec des Proxy MLD, les proxy peuvent former un arbre de gestion de groupe enraciné sur un routeur multipoint. Seul le routeur `R1` est responsable de joindre le groupe et établir la branche multipoint vers l'arbre. Chaque proxy (`Proxy 2` et `Proxy 3`) collecte localement les informations de gestion de groupe sur ses propres liens et transmet un rapport d'abonnement (*Host Membership Report*) vers son proxy hiérarchiquement supérieur (`Proxy 1`). `Proxy 1` se charge de transmettre un autre rapport d'abonnement qui reflète exactement l'information de gestion de groupe des proxy 2 et 3.

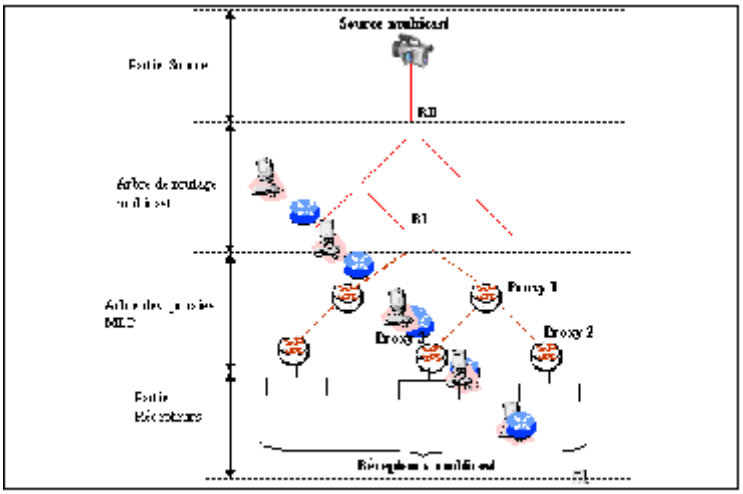


Figure 11-10. Gestion de groupe avec des Proxies MLD

Pour maintenir une base d'information des abonnements, chaque proxy maintient un ensemble d'enregistrement d'abonnement pour chacune de ses interfaces. Chaque enregistrement a la forme suivante :

- adresse multicast IPv6,
- mode de filtrage,
- liste de sources

L'avantage du recours à des proxy est d'éviter l'utilisation d'un protocole de construction d'arbre multicast dans certains types de topologies de réseau simples comme les DSLAM (*Digital Subscriber Line Acess Multiplexer*). Dans une telle topologie, seul le routeur de bordure du réseau est censé implémenter les fonctionnalités de construction d'arbre multicast ce qui simplifie l'architecture et le coût des équipements d'accès. De même la charge du réseau est considérablement réduite grâce à la suppression du trafic de signalisation pour la maintenance de l'arbre multicast.

Cependant, l'utilisation de proxy peut avoir des inconvénients. Par exemple, les proxy ne permettent pas une tolérance des défaillances de liens ou de routeurs puisque les proxy ne peuvent pas reconstruire un arbre multicast en fonction de l'état du réseau. La panne d'un proxy d'une hiérarchie donnée, entraîne la panne de tous les proxy du niveau inférieur. Par conséquent les récepteurs multicast ne peuvent plus recevoir du trafic multicast ni envoyer les rapports d'abonnements au routeur multicast.

D) La diffusion du multicast IPv6 sur le lien-local

Sur le lien-local, un segment Ethernet par exemple, les paquets IPv6 multicast sont encapsulés dans une trame Ethernet ayant une adresse MAC multicast (le 8ème bit de poids fort des adresses MAC multicast est positionné à 1, ce qui les distingue des adresses MAC unicast). Cette adresse MAC est la concaténation de 2 octets ayant pour valeur `33-33` et des 32 bits de poids faible de l'adresse IPv6 multicast. Si l'on considère l'adresse multicast IPv6 `FF1E::12:AC21:6521`, l'adresse MAC correspondante sera `33-33-AC-21-65-21`.

Les mécanismes qui existent en IPv4 comme IGMP snooping ou CGMP (*Cisco Group Multicast Protocol*) qui permettent de limiter la diffusion des paquets multicast aux hosts abonnés n'existent pas encore pour IPv6. MLD snooping n'est toujours pas implémenté sur les commutateurs. La conséquence directe est que si le réseau n'est pas segmenté correctement, à l'aide de VLANs par exemple, le flux multicast inondera tout le réseau et chaque host recevra des paquets multicast non souhaités.

3) La construction d'arbre multicast – PIM

Sur le lien-local, le protocole MLD permet aux stations de travail d'exprimer leur intérêt pour un groupe multicast (et d'un ensemble de sources pour MLDv2). Il reste ensuite à acheminer les paquets multicast IPv6 entre les sources et les abonnés. Ceci est réalisé par le protocole PIM (*Protocol Independant Multicast*). Le fonctionnement du protocole PIM en IPv6 est le même que pour IPv4. Aussi l'objectif de

cette section est d'expliquer les bases du protocole de construction d'arbre multicast PIM pour permettre une meilleure compréhension du chapitre multicast.

Contents

A) Le protocole PIM SM - Sparse-Mode

Le protocole PIM-SM (*Protocol Independent Multicast - Sparse Mode*) permet la construction d'arbres multicast (RFC 2362). Ce protocole peut utiliser la base d'information de routage unicast sous-jacente, ou une autre base d'information de routage multicast comme BGP IPv6 multicast SAFI. Dans ce sens il est indépendant. Il construit pour chaque groupe un arbre de diffusion unidirectionnelle, chaque arbre prenant racine sur un noeud spécifique appelé point de rendez-vous ou RP (*Rendez-vous Point*). Lorsqu'il y a plusieurs sources alimentant le même groupe, les paquets en provenance des différentes sources convergent vers le RP associé au groupe, puis à partir de celui-ci les paquets empruntent (et donc partagent) l'arbre associé au groupe, ce qui leur permet d'atteindre tous les destinataires membres du groupe. La construction de l'arbre de diffusion peut se décomposer en 3 étapes.

a) Etape 1 : l'arbre partagé

Une station de travail exprime son désir de recevoir le trafic multicast associé à un groupe en utilisant le protocole MLD vu dans la section précédente. Le routeur PIM en charge du lien-local ou DR (*Designated Router*), envoie un message `PIM (*,G) Join` vers le RP associé à ce groupe. On utilise la notation `(*,G)` car cela concerne n'importe quelle source pour le groupe `G`.

Ce message va être propagé de routeur en routeur vers le RP de ce groupe. A chaque routeur traversé un état associé à l'arbre multicast du groupe `G` est créé. Finalement le message `PIM (*,G) Join` va atteindre soit le RP, soit un routeur possédant déjà un état `(*,G)` associé au groupe. Quand plusieurs récepteurs adhèrent à un groupe, les messages `PIM Join` envoyés par les DR convergent vers le RP de ce groupe, ce qui forme l'arbre multicast pour ce groupe. Cet arbre est appelé RPT (*Rendez-vous Point Tree*) et il est qualifié d'arbre partagé puisqu'il sera utilisé pour atteindre tous les destinataires du groupe quel que soit l'émetteur du paquet multicast (cf. figure Arbre partagé).

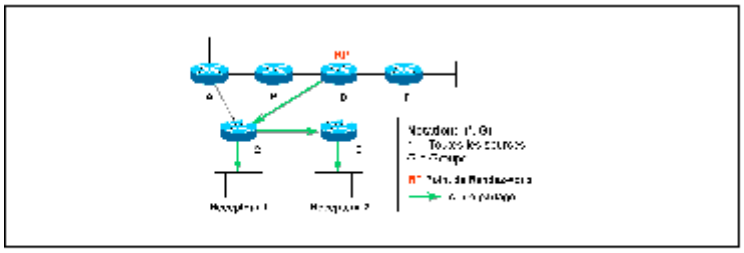


Figure 11-11 . Arbre partagé

Des messages PIM d'adhésion sont envoyés périodiquement tant qu'au moins un destinataire est membre du groupe. Quand tous les destinataires situés sur un lien-local quittent un groupe, le DR peut envoyer un message PIM d'élagage (`PIM Prune`). Une durée limite de validité étant associée à chaque adhésion, si aucun message PIM ne parvient, l'adhésion sera résiliée.

Dès qu'une station émet un paquet multicast vers un groupe, le DR de son lien-local encapsule ce paquet dans un datagramme unicast ayant pour adresse de destination le RP associé au groupe. Lorsque le RP reçoit ce datagramme, il décapsule le paquet multicast, et le propage sur le RPT associé au groupe. Le paquet est dupliqué aux nœuds qui forment de nouvelles branches, et donc parvient à l'ensemble des destinataires membres du groupe. Les datagrammes ayant encapsulé les paquets multicast sont appelés messages PIM Register (cf. figure Envoi des messages PIM Register).

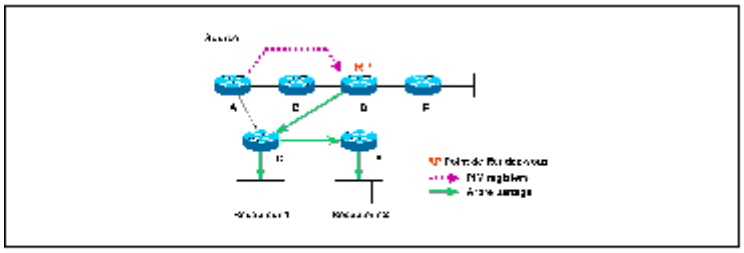


Figure 11-12 . Envoi des messages PIM Register

b) Etape 2 : l'acheminement spécifique

L'encapsulation dans les messages PIM Register est doublement inefficace :

- L'encapsulation et la décapsulation sont des opérations qui sont coûteuses pour un routeur surtout s'il ne possède pas de matériel spécifique pour accélérer ces opérations.

Le chemin de l'émetteur vers le RP puis à travers l'arbre RPT pour les récepteurs qui sont placés près de l'émetteur peut engendrer un large détour, produisant des délais et pouvant surcharger inutilement le réseau.

Le RP pourra choisir de basculer vers un acheminement natif (sans encapsulation) entre la source et le RP. Dans ce cas, lorsque le RP reçoit un message `PIM Register` contenant un paquet multicast provenant d'un émetteur `S` pour un groupe `G`, il peut envoyer un message `PIM (S,G) Join` vers `S`.

Ce message provoque dans les routeurs traversés la création d'un état multicast spécifique `(S,G)`. Ces états ne seront utilisés que pour transmettre les paquets multicast émis par S vers le groupe `G`. Finalement, ce message `PIM (S,G) Join` arrivera au DR associé à la source.

Dès que le `PIM (S,G) Join` arrive au DR de la source, ce dernier émet les données à la fois nativement vers le RP et en les encapsulant dans les messages `PIM Register`. Quand des paquets multicast natifs arrivent simultanément avec des paquets multicast encapsulés en provenance de la même source et pour le même groupe, le RP reçoit alors deux copies de chaque message. À partir de cet instant le RP doit détruire la copie des paquets multicast encapsulés, et il envoie un message `PIM Register-Stop` vers le DR de la source. Lorsque le DR reçoit un message `PIM Register-Stop`, il cesse d'encapsuler les paquets multicast dans des messages `PIM Register`.

A la fin de cette phase, le trafic émis par la source `S` pour le groupe `G` suit l'arbre centré sur la source jusqu'au RP, puis utilise l'arbre RPT (associé au groupe `G`) pour atteindre tous les destinataires du groupe G (cf. figure Le RP rejoint l'arbre centré sur la source).

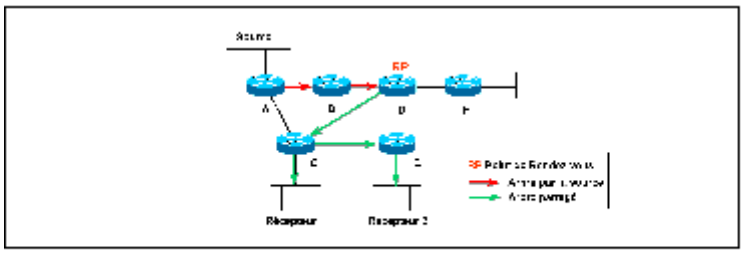


Figure 11-13 . Le RP rejoint l'arbre centré sur la source

On notera que l'émetteur peut commencer à émettre avant ou après qu'un destinataire s'abonne à un groupe, et donc que cette deuxième phase peut être mise en place avant que l'arbre RPT soit construit.

c) Etape 3 : l'arbre des plus courts chemins

L'étape 2 supprime le surcoût introduit par l'encapsulation entre l'émetteur et le RP ; cependant cela n'optimise pas complètement le chemin suivi par les paquets multicast. Pour de nombreux destinataires, le transit par le RP provoque un détour important si on compare ce chemin avec le chemin le plus court entre l'émetteur et chaque destinataire (cf. figure Arbre centré sur la source).

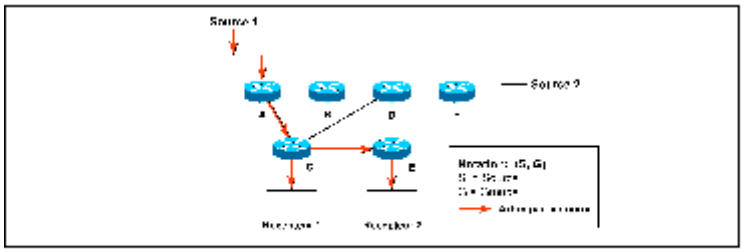


Figure 11-14 . Arbre centré sur la source

Une fois que le DR d'un récepteur reçoit les paquets émis par la source `S` vers le groupe `G`, il peut rejoindre l'arbre centré sur la source. On désignera cet arbre par SPT (*Shortest Path Tree*).

Dans ce cas, le DR émet un message `PIM (S,G) Join` vers l'émetteur. Cela crée des états spécifiques `(S,G)` dans les routeurs rencontrés sur le chemin vers l'émetteur. Le message atteint le DR de la source ou un autre routeur ayant déjà l'état `(S,G)`. Les paquets multicast dorénavant émis par l'émetteur S suivront les états `(S,G)`.

À partir de cet instant le DR du destinataire peut recevoir deux copies de chaque paquet multicast : une provenant du RP et ayant suivi l'arbre RPT associé, l'autre provenant directement de l'émetteur en ayant emprunté l'arbre centré sur la source (SPT). Dès que le premier paquet multicast est reçu en provenance de l'arbre centré sur la source, le DR du destinataire détruit les paquets qui arrivent via le RPT. Il envoie un message d'élagage `PIM (S,G) Prune` qui est propagé de routeur en routeur vers le RP. Dans chaque routeur rencontré ce message place un état indiquant que le trafic multicast `(S,G)` ne doit plus être propagé sur l'arbre partagé.

B) Le protocole PIM SSM - Source Specific Multicast

MLDv2 permet à un récepteur de spécifier le groupe auquel il veut s'abonner ainsi qu'un ensemble de sources pour ce groupe. La détermination d'un RP dans ce cas n'est pas nécessaire puisque les sources sont connues à l'avance par les destinataires. Prenons l'exemple d'une station qui s'abonne au groupe `G` en indiquant son intérêt pour les sources `S1` et `S2` uniquement. Le DR du récepteur peut envoyer directement des messages `PIM (S1,G) Join` vers `S1` et `PIM (S2,G) Join` vers `S2` et joindre ainsi les 2 arbres par la source associés. PIM SSM ne définit aucun nouveau message, c'est un sous-ensemble de PIM Sparse-Mode. Cependant, des adresses multicast dédiées pour PIM SSM doivent être utilisées. Ce sont des adresses dérivées du préfixe `FF3X::/96`.

Il est à noter qu'il y a une forte préférence vers le modèle SSM pour le multicast. En effet, il permet de pallier tous les problèmes rencontrés dans le modèle et non résolus de manière complètement satisfaisante à ce jour : inter-domaine multicast, allocation des adresses multicast, annonce des sessions...

Les principales différences avec IPv4 sont les suivantes :

- Les messages PIM sont échangés avec l'adresse de destination `FF02::D` (adresse de tous les routeurs PIM du lien)
- L'adresse source utilisée pour les messages PIM est l'adresse lien-local de l'interface d'où est émis le message. La conséquence directe est que cette adresse ne permet pas de réaliser le RPF check sur les messages PIM. Ainsi une option a été définie pour les messages `PIM Hello` afin de pouvoir spécifier toutes les adresses globales de l'interface. Ce sont ces adresses globales qui seront utilisées pour faire le RPF check sur le message. La structure de cette option qui est décrite figure Option "Address list" dans les messages PIM Hello est détaillée dans [Fenner2-id] :

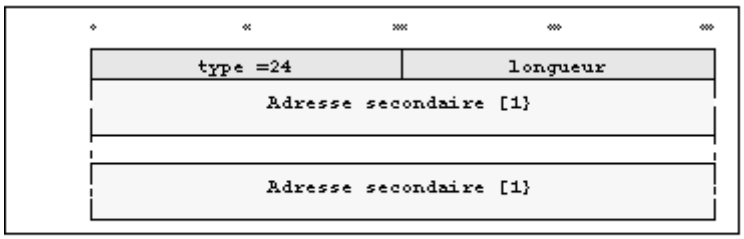


Figure 11-15 . Option "Address list" dans les messages PIM Hello

- La technologie embedded-RP est une différence majeure avec IPv4 en ce qui concerne le protocole PIM SM.

4) Multicast IPv6 inter-domaine

L'Internet est comme son nom l'indique une interconnexion de réseaux sous la direction d'entités administratives différentes (AS : *Autonomous System*) qu'on appelle domaines. Il faut définir des mécanismes qui permettent à ces domaines de dialoguer, tout en préservant leur autonomie. Ces mécanismes sont déjà pleinement déployés pour l'unicast, mais sont encore en plein développement pour ce qui concerne le multicast.

Aujourd'hui, les protocoles multicast inter-domaine pour IPv4 sont considérés comme non extensibles comme on le verra dans la section suivante sur MSDP. Ainsi pour IPv6, de nouveaux mécanismes ont été définis, prenant en compte l'existence de deux modèles de diffusion pour les applications : ASM et SSM. Alors que le modèle ASM inter-domaine était implémentée par MSDP en IPv4, la solution qui semble privilégiée aujourd'hui est embedded-RP. Pour le modèle SSM, l'utilisation du protocole MLDv2 est indispensable afin d'informer le réseau des sources d'intérêt pour la construction des arbres. Dans cette partie, nous présentons deux solutions qui pourraient être déployées à grande échelle pour le multicast IPv6 : PIM-SM associé à embedded-RP pour l'ASM et PIM-SSM associé à MLDv2 pour SSM.

a) ASM

En IPv4, un domaine PIM correspond à un ensemble de routeurs PIM gérés par une même entité. Tous les routeurs du domaine PIM sont configurés avec le même ensemble de points de rendez-vous qui appartiennent aussi à ce domaine. Pour permettre à des sources et des récepteurs répartis sur différents domaines de participer à une session multicast, un protocole été standardisé et déployé : il s'agit de MSDP (*Multicast Source Discovery Protocol*) [RFC 3618].

Des peerings MSDP sont déployés entre les RP des différents domaines PIM comme indiqué sur la See Peering MSDP. Ils permettent aux RP de s'échanger les informations quant aux sources actives dans les différents domaines. Chaque RP envoie à ses peers MSDP les sources qui émettent et les groupes destinataires.

Des filtres peuvent être appliqués sur les peerings pour permettre les annonces de certaines sources pour certains groupes uniquement.

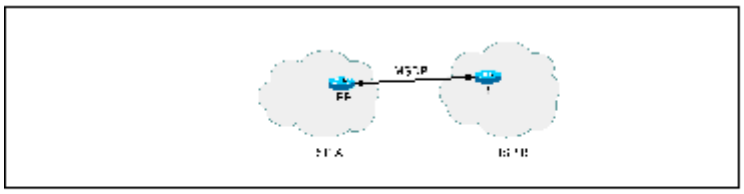


Figure 11-16 . Peering MSDP

Ce protocole classé expérimental ne permet pas une utilisation massive de la technologie multicast puisque les RP doivent s'échanger toutes les sources actives sur l'Internet. De plus, il s'agit d'un protocole compliqué, peu implémenté et difficile à administrer. Aussi, depuis plusieurs années, l'IETF recommande l'utilisation du modèle SSM afin de rendre le modèle plus simple, même si le service rendu est différent avec SSM. Pour toutes ces raisons MSDP n'est pas défini pour IPv6 et il n'existe pas de protocole équivalent.

b) Embedded-RP

En l'absence de MSDP, la construction de l'arbre multicast avec PIM-SM nécessite que tous les routeurs PIM soient configurés avec le même ensemble de RP. Un groupe doit correspondre à un seul RP unique dans tout l'Internet puisque les RP ne peuvent pas s'échanger les informations sur les sources actives en IPv6.

Il est difficile d'imaginer un protocole permettant d'échanger les informations sur les RP existants et les adresses multicast qu'ils gèrent. Un tel protocole ne serait pas meilleur que MSDP.

Une proposition simple a émergé : embarquer l'adresse du RP dans l'adresse multicast (ou embedded-RP). Ceci semble impossible car les deux adresses ont une taille identique (de 128 bits). Cependant, en faisant certaines hypothèses sur l'identifiant d'interface du RP, il est possible de parvenir à cette solution. La méthode de construction d'une adresse embedded-RP impose quelques changements :

- Modifications sur le protocole PIM SM.

  Embedded-RP [RFC 3618] nécessite une modification de l'algorithme de correspondance entre les adresses multicast et les RP (ou *group-to-RP mapping*). Pour un paquet à destination d'une adresse dérivée du préfixe `FF70::/12`, l'adresse du RP doit être retrouvée à l'aide des mécanismes décrits ici.
  Embedded-RP doit être supporté sur tous les routeurs de l'arbre partagé, le RP et le DR des sources et des récepteurs. Le support d'embedded-RP sur l'arbre centré sur la source et sur les routeurs entre la source et le RP n'est pas nécessaire puisque ce sont des messages `PIM (S,G) prune/join` qui sont utilisés. Cependant, il est à noter qu'une implémentation sur tous les routeurs PIM SM du réseau simplifie l'utilisation et la gestion de la technologie embedded-RP.

- Impact sur le modèle multicast.

  L'inter-domaine multicast en IPv6 apparaît donc très différent de ce qui est réalisé en IPv4. Notamment la notion de domaine PIM disparaît et le terme inter-domaine multicast ne correspond plus vraiment. L'Internet IPv6 multicast est un unique domaine PIM dans lesquels sont configurés des multiples points de rendez-vous (cf. figure Le modèle embedded-RP).

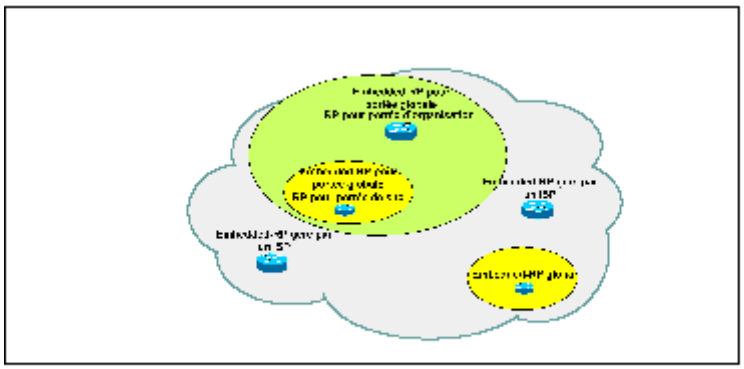


Figure 11-17. Le modèle embedded-RP

Il est encore difficile à la date de rédaction de ce chapitre de déterminer si ce modèle va être adopté. Si des tests ont montré que la technologie embedded-RP fonctionnait, il reste néanmoins des questions sur les impacts causés par les différences avec le modèle connu et déployé à ce jour pour IPv4.

C) Déploiement de SSM sur plusieurs domaines

Le modèle SSM, implémenté par PIM-SSM constitue un sous-ensemble simplifié du modèle ASM. En effet, il a été défini historiquement pour répondre aux problèmes de l'inter-domaine ASM n'ayant pas été résolus en IPv4. Par conséquent, les mécanismes permettant de réaliser l'inter-domaine SSM en IPv6 sont très similaires à ceux utilisés en IPv4.

Au niveau du protocole PIM-SSM, il n'y a aucune disposition à prendre pour permettre à ce protocole de fonctionner entre plusieurs domaines. Les messages PIM Join sont acheminés de routeur en routeur entre les DR des récepteurs et les sources spécifiées par les récepteurs. Peu importe donc que les sources soient dans le même domaine que les récepteurs.

Le problème du déploiement du protocole de construction d'arbre multicast PIM-SSM sur plusieurs domaines se situera donc au niveau du routage.

5) Déploiement du multicast

A) Le M6Bone

Le M6Bone est un réseau de test multicast IPv6. Le projet a débuté en Juillet 2001, sous l'impulsion en France de l'association Aristote, du G6 et de RENATER. Le but du projet est d'offrir une connectivité multicast IPv6 aux sites voulant expérimenter cette technologie. Le M6Bone permet aussi de valider des applications ou matériels relatifs au multicast IPv6.

a) Topologie du M6Bone

Les figures Carte européenne du M6bone et Carte mondiale du M6bone présentent des cartes de la topologie actuelle du M6Bone, avec les sites et réseaux connectés.

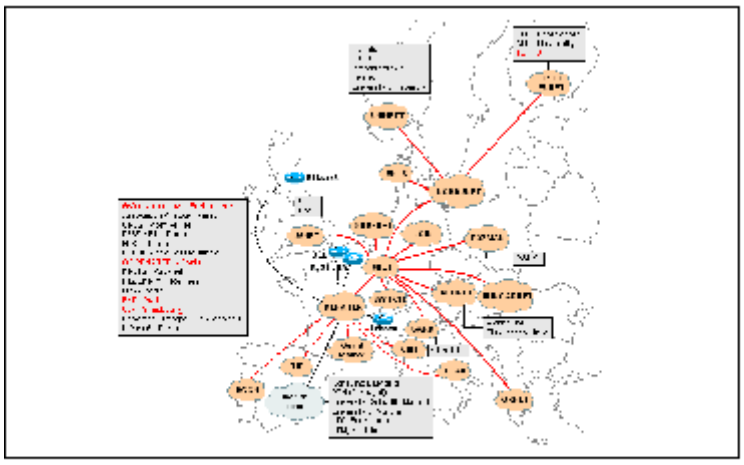


Figure 11-18 . Carte européenne du M6bone

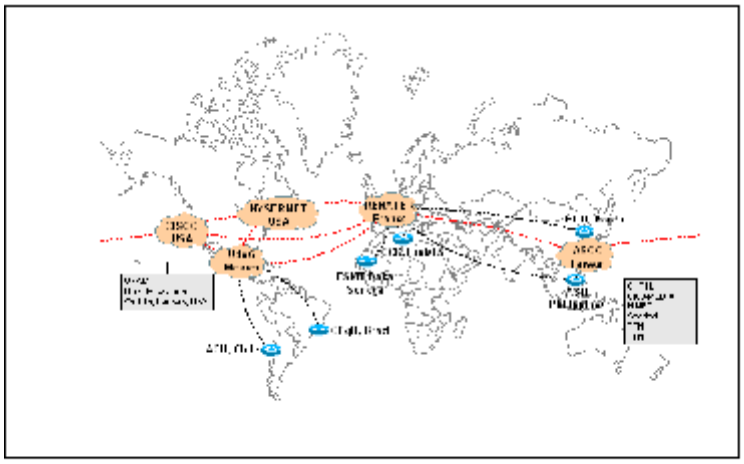


Figure 11-19 . Carte mondiale du M6bone

La majeure partie des liens du M6Bone est constituée de tunnels (IPv6 multicast dans IPv6 unicast ou alors IPv6 multicast dans IPv4). Si, au départ, des équipements différents devaient être utilisés pour le multicast

et l'unicast (absence de table de routage IPv6 multicast), l'implémentation de MBGP et de PIM sur certains routeurs commerciaux permet aujourd'hui de considérer un déploiement à grande échelle.

Le protocole utilisé dans le M6Bone est PIM sparse-mode. Le point de rendez-vous global est géré par RENATER. Des questions sur l'inter-domaine multicast subsistent. MSDP ne verra pas le jour pour IPv6 et des solutions sont à l'étude. Le M6Bone permet de tester à grande échelle les solutions envisagées.

b) Services disponibles grâce au M6Bone

Presque tous les systèmes d'exploitation supportent IPv6 aujourd'hui et permettent d'utiliser des applications multicast. Les outils du MBone (vic, rat, sdr, nte, whiteboard...) supportent aujourd'hui IPv6 et il est possible de réaliser des visioconférences sur le M6Bone. Des outils comme freeamp permettent aussi de diffuser des stations de radio sur le M6Bone à haut débit. Des passerelles (ou réflecteurs) avec le réseau IPv4 multicast ont aussi été développées. Il est ainsi possible pour des personnes sur le M6Bone de joindre des sessions IPv4 et vice-versa. Des réflecteurs unicast/multicast permettent à des personnes ne disposant que d'une connectivité unicast de rejoindre des sessions multicast. Tous ces outils sont régulièrement utilisés pour la diffusion d'événements (conférences, causeries de Renater, séminaires Aristote, ...)

c) Communauté M6Bone

Une mailing-list libre (m6bone@ml.renater.fr), permet aujourd'hui à plus de 180 personnes d'échanger leurs connaissances sur le multicast IPv6 (routage, applications...) Un site web (http://www.m6bone.net) collecte les informations principales pour tout savoir sur les technologies multicast et les évolutions du réseau M6Bone. La configuration des différents équipements est également détaillée.

B) 6NET

Le projet 6NET est un projet IST (*Information Society Technology*) du 5ème programme cadre de la Commission Européenne, regroupant principalement des partenaires du monde académique d'une quinzaine de pays européens. Il a entre autres permis le déploiement d'un backbone de test pan-européen IPv6 permettant d'interconnecter les réseaux de recherche partenaires. La figure suivante présente la topologie du réseau 6NET.

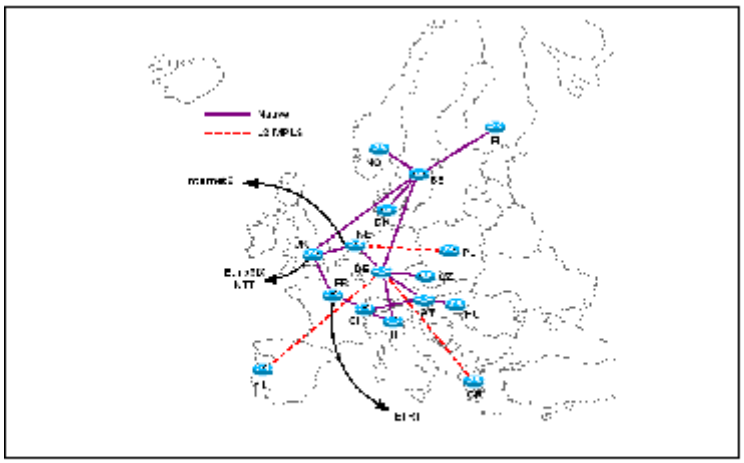


Figure 11-20 . Topologie du réseau 6NET

Dès mars 2003, il a été possible de déployer le multicast IPv6 sur l'ensemble des routeurs du réseau 6NET et d'interconnecter nativement tous les réseaux nationaux de la recherche partenaires du projet. Les protocoles PIM SM, PIM SSM, MBGP (SAFI multicast IPv6) ont été déployés. BSR (*Bootstrap Router*) a été configuré sur les routeurs de cœur afin de permettre l'échange d'information quant aux points de rendez-vous configurés. De plus, tous les nœuds du réseau supportent Embedded-RP.

Le réseau 6NET a été interconnecté au M6Bone, ce qui a d'une part permis l'établissement de sessions multicast avec des entités non partenaires du projet et d'autre part facilité la dissémination de l'expérience acquise à travers le projet vers tous les acteurs du M6Bone. Applications multicast IPv6

Il existe un certain nombre d'applications qui fonctionnent déjà en multicast IPv6. La liste suivante n'est pas exhaustive mais montre l'étendue des services déjà disponibles :

- DVTS (Digital Video Transport System) permet la diffusion de flux vidéo de très bonne qualité à travers un réseau multicast IPv6. Des débits de plus de 20 Mbit/s sont utilisés pour transmettre la vidéo. Du matériel dédié est nécessaire comme des caméras numériques NTSC avec un port IEEE 1394.
- VideoLAN. Cette application GNU est un projet de l'Ecole centrale de Paris. VLC (VideoLan Client) supporte un grand nombre de formats audio et vidéo (MPEG-1, MPEG2, MPEG-4, DivX, mp3..), et aussi les DVDs, VCDs et de nombreux protocoles de streaming. Cette application peut tout aussi bien être source ou récepteur de flux multicast IPv6.
- Windows Media Player 9. Les versions 9 et suivantes du client Windows Media Player permettent de recevoir des flux audio et vidéos multicast IPv6.
- Freeamp. Freeamp est une application libre qui permet de recevoir des flux audio diffusés en streaming. Le format MP3 est le plus utilisé avec cette application qui correspond parfaitement à l'écoute de radio sur Internet. Le multicast IPv6 est supporté ce qui peut permettre aux sources de diffuser la radio avec une excellente qualité sans saturation des ressources du réseau.
- Isabel. L'application Isabel est entièrement conçue pour le télé-enseignement. L'application permet la transmission de vidéo, d'audio et de transparents entre les multiples participants d'une session. Ceux-ci peuvent demander la parole pour poser des questions, et les intervenants peuvent clarifier certains points en rajoutant des éléments sur les transparents à l'aide d'un stylo virtuel. Il est aussi possible d'utiliser des modes simplifiés pour des visioconférences simples.
- VIC est l'application GNU traditionnellement la plus utilisée en multicast pour la visioconférence. Cet outil permet a plusieurs participants de s'échanger du trafic vidéo de manière simple, avec différents formats permettant une optimisation des débits disponibles pour la session. Il est

possible avec VIC de créer des sessions avec un très grand nombre de participants comme on peut le voir sur la copie d'écran donnée See Applications utilisant le multicast.

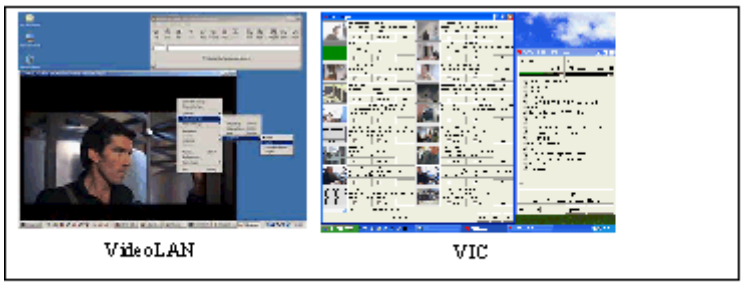


Figure 11-21. Applications utilisant le multicast

- RAT (Robust Audio Tool) est l'équivalent de l'application VIC mais permet l'échange de l'audio. Pour assurer une visioconférence, il faut utiliser VIC et RAT en parallèle.
- WB (White Board) est un tableau blanc partagé. Cette application peut permettre lors de visioconférences effectuées avec VIC et RAT de clarifier certains points à l'aide de schémas explicatifs, pouvant inclure du texte.
- NTE (Network Text Exchange) permet à des utilisateurs de communiquer en échangeant des messages. Chaque personne du groupe a une couleur personnelle ce qui permet de distinguer les messages écrits par chacun.
- MAD-FLUTE permet le transfert de fichiers en utilisant la technologie multicast. MAD-FLUTE est une implémentation du protocole FLUTE [RFC 3926]. A l'aide de cette application, il est possible de s'abonner à un groupe de diffusion de fichiers. MAD-FLUTE peut permettre la mise à jour de logiciels en téléchargeant les nouvelles versions régulièrement.
- SDR permet de créer et d'annoncer des sessions multicast en utilisant les protocoles SDP et SAP. Les personnes désirant participer à une session n'ont qu'à retrouver l'annonce dans la liste créée par SDR. Les bonnes applications sont ensuite automatiquement lancées, avec les paramètres associés adresse multicast IPv6, numéro de port, codecs...

6) Coexistence avec le multicast IPv4

L'objectif des passerelles IPv6/IPv4 est de permettre à des utilisateurs répartis dans les réseaux IPv6 et IPv4 multicast de participer à une même session multicast. Il existe des passerelles statiques aussi appelées réflecteurs qui permettent à des groupes IPv4 et IPv6 prédéfinis de communiquer. Une passerelle dynamique a aussi été conçue et implémentée permettant une complète interaction entre les réseaux multicast IPv6 et IPv4.

A) Réflecteurs

Ce type de passerelle permet d'assurer la correspondance entre un groupe IPv4 et un groupe IPv6 multicast. Sur chaque passerelle, les adresses des groupes IPv4 et IPv6 sont configurées statiquement. Ces réflecteurs peuvent donc être utilisés pour des sessions périodiquement utilisées sur des adresses IPv4 et IPv6 qui ne changent pas. La passerelle déployée sur RENATER traduit d'IPv4 à IPv6 (et vice versa) des sessions d'enseignement à distance, ainsi que des conférences organisées sur les thèmes des réseaux.

Une passerelle s'abonne aux groupes IPv4 et IPv6 multicast qui sont rentrés en paramètre. Pour cela, elle envoie un message IGMP report sur l'interface vers le réseau multicast IPv4 et un message MLD report vers le réseau multicast IPv6. Il est bien évidemment possible d'utiliser une seule interface physique lorsque le multicast IPv4 et IPv6 sont configurés sur le même lien-local.

Une fois les messages IGMP et MLD report envoyés (cf. figure Réflecteur IPv6/IPv4 multicast), la passerelle reçoit les flux multicast correspondants et n'a plus qu'à faire la traduction des en-têtes.

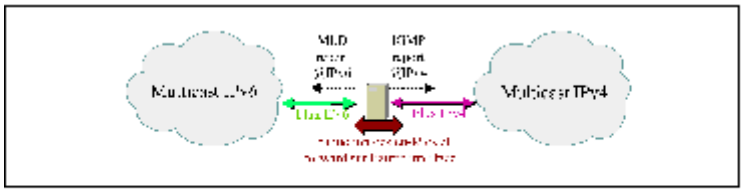


Figure 11-22 . Réflecteur IPv6/IPv4 multicast

Dans le paquet traduit, l'adresse source est l'adresse (IPv4 ou IPv6) de la passerelle et l'information de la source est perdue. Cependant, l'expérience acquise montre que beaucoup d'applications utilisent la couche applicative pour identifier les différents participants de la session. L'utilisation d'une passerelle devient donc transparente pour les différents utilisateurs.

Un problème peut survenir lors du déploiement de réflecteurs si deux passerelles sont configurées pour les mêmes groupes. Une boucle est alors créée dans le réseau ce qui sature les liens du réseau. Pour limiter la probabilité d'avoir ce genre de problème, il est possible de configurer une passerelle avec les ports UDP utilisés. La probabilité d'avoir deux passerelles pour les mêmes adresses et numéros de ports devient quasi nulle. Il ne faut pas non plus voir ce problème comme une faille de sécurité car une personne malveillante peut émettre une forte quantité de trafic dans les groupes considérés pour saturer les liaisons et les

équipements, et n'a pas besoin de passerelle pour cela. Un contrôle du débit maximal utilisable sur chaque groupe est une bonne solution pour éviter des conséquences trop importantes de ces attaques par déni de service.

B) Passerelles dynamiques

L'intérêt d'une telle passerelle est de pouvoir traduire n'importe quelle session IPv4 en IPv6 et vice-versa sans configuration préalable. Un utilisateur peut déployer le multicast IPv6 et arrêter le multicast IPv4 car il pourra avoir accès à tous les services IPv4 multicast grâce à la passerelle.

Le principe de base de cette passerelle est d'utiliser l'adresse IPv4 multicast comme identifiant de groupe de l'adresse multicast IPv6. La passerelle dynamique est donnée figure Principe de la passerelle dynamique IPv6/IPv4 multicast.

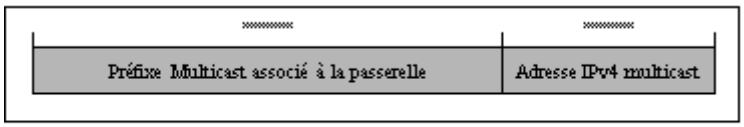


Figure 11-23 . Structure des adresses utilisées par les passerelles dynamiques

- Un point de rendez-vous dans le réseau multicast IPv6 pour le préfixe qui lui est associé,
- Un client terminal dans le réseau multicast IPv4.

Quand un client multicast IPv6 veut recevoir du flux multicast IPv4, il envoie un message MLD report pour l'adresse IPv6 multicast dérivée du préfixe multicast associé à la passerelle et de l'adresse IPv4 multicast. Le DR du lien envoie alors un message PIM join pour cette adresse qui sera propagé jusqu'à la passerelle puisqu'elle est configurée comme RP pour le préfixe utilisé. La passerelle retrouvera l'adresse IPv4 dans le paquet et enverra un message IGMP report pour cette adresse IPv4. Le trafic IPv4 multicast atteindra ensuite la passerelle qui traduira les paquets comme cela est expliqué dans la partie précédente concernant les passerelles statiques.

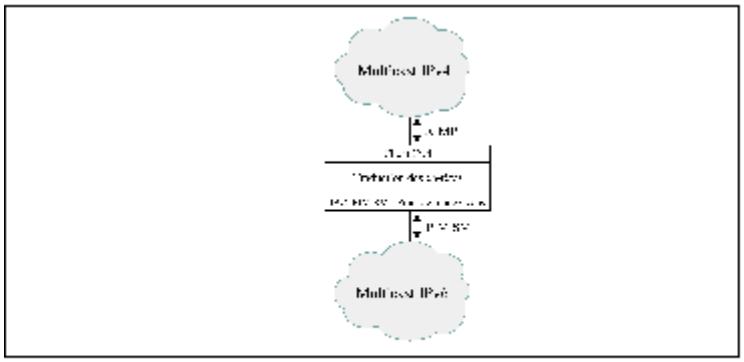


Figure 11-24 . Principe de la passerelle dynamique IPv6/IPv4 multicast

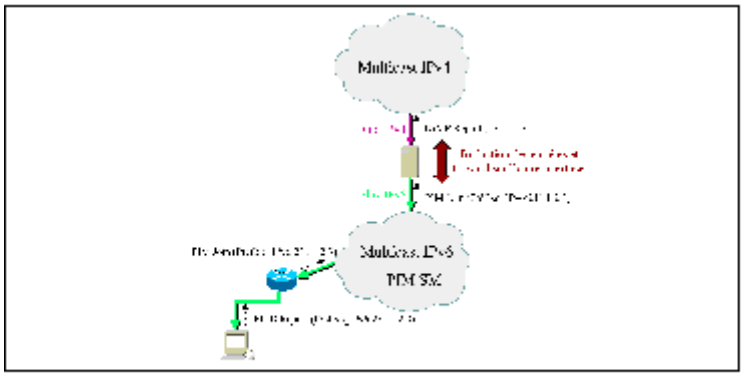


Figure 11-25 . Schéma explicatif de la passerelle dynamique

Lorsqu'une source émet du trafic multicast IPv6 avec des adresses associées à la passerelle, le DR sur le lien-local encapsule ce trafic vers la passerelle puisqu'elle est configurée comme point de rendez-vous pour le préfixe utilisé. La passerelle va traduire ensuite les en-têtes de la même manière qu'une passerelle statique en utilisant l'adresse IPv4 embarquée dans l'adresse IPv6. Les paquets traduits sont envoyés sur l'interface multicast IPv4. Comme expliqué dans la section consacrée à PIM, les paquets pourront être transmis de manière native entre la source et le RP afin d'éviter le coût lié à l'encapsulation dans les messages PIM register.

Pour rendre l'usage de ces passerelles encore plus facile, il est possible de traduire d'IPv4 à IPv6 les annonces SAP de toutes les sessions IPv4. Ainsi une session IPv4 annoncée par SAP sera visible par les clients multicast IPv6, avec des adresses automatiquement traduites pour permettre d'utiliser la passerelle. Il est ainsi possible pour un site de se passer entièrement du multicast IPv4, en gardant la certitude d'avoir accès à tous les services associés.

7) Etude pratique du déploiement du multicast IPv6

L'objectif est ici de détailler les différentes étapes de la mise en place d'un service multicast IPv6 dans un réseau existant. Pour avoir d'avantage de détails sur la configuration des équipements, le lecteur pourra lire le chapitre détaillant la configuration des routeurs.

Contents

A) Service et applications

La première chose est de comprendre quelles sont les caractéristiques principales du service multicast que l'on souhaite mettre en place dans le réseau. Celles-ci dépendent principalement des applications désirées.

Comme expliqué dans une section précédente de ce chapitre, il existe deux modèles pour le multicast IPv6: le modèle ASM (*Any Source Multicast*) et le modèle SSM (*Source Specific Multicast*). Pour des applications de vidéoconférence multi-utilisateurs, le modèle ASM doit être déployé. Si les applications désirées sont de type diffusion de contenu d'une source connue vers un ensemble de récepteurs (par exemple la radio ou la télévision sur internet), le modèle choisi sera SSM. Il est bien évidemment possible de déployer les deux modèles multicast dans un même réseau.

Le protocole PIM SM/SSM permet l'implémentation de ces deux modèles :

- Utilisé avec MLDv1 ou MLDv2 et configuré avec un certain nombre de points de rendez-vous, PIM permet de fonctionner en mode ASM.
- Utilisé avec MLDv2, PIM peut fonctionner en mode SSM, et dans ce cas il n'est pas nécessaire de configurer des points de rendez-vous. Les clients grâce au protocole MLDv2 peuvent spécifier les sources multicast et la configuration des RP devient inutile.

B) Topologie du réseau

La topologie du réseau multicast dépend du support des protocoles multicast choisis dans les équipements du réseau. Il est plus simple de déployer le multicast IPv6 lorsque ce service est supporté par tous les équipements du réseau. Dans ce cas, les topologies unicast et multicast sont congruentes ce qui simplifie le design et l'administration du réseau. Lorsque des routeurs ne sont pas capables de gérer le multicast IPv6, plusieurs solutions peuvent être envisagées : tunnels multicast, VLANs Ethernet dédiés, changement des équipements réseaux, mise à jour du logiciel des routeurs, changement de la topologie du réseau, PVC ATM ou LSP MPLS dédiés...

a) Topologies unicast et multicast congruentes

Dans le cas où tous les équipements du réseau supportent les protocoles multicast nécessaires (PIM SM/SSM, MLDv2...) la topologie multicast pourra être la même que la topologie unicast déployée : on parle alors de topologies unicast et multicast congruentes. Cela signifie que les routes entre les sources et récepteurs multicast sont identiques aux routes unicast.

Le protocole PIM pourra alors utiliser la table de routage unicast. L'implantation de MBGP (SAFI IPv6 multicast) ou de routes statiques multicast n'est alors pas nécessaire.

b) Topologies unicast et multicast non congruentes

Dans ce cas, le déploiement et l'administration du service multicast deviennent plus complexes. Les cas de topologies non-congruentes sont multiples : déploiement d'équipements spécifiques pour le multicast, routage du multicast sur des liens dédiés...

L'administrateur du réseau devra alors utiliser une base d'information de routage différente de la base unicast. Le protocole PIM devra reposer sur des informations de routage correspondant à la topologie multicast déployée. Des routes statiques multicast ou alors MBGP IPv6 multicast SAFI doivent alors être considérés.

La figure Multicast dans une topologie simple suivante montre le cas d'un site de petite taille (un seul routeur et un seul lien local) connecté à l'Internet v6 unicast et au multicast (M6Bone) par deux liens différents. La connexion vers le M6Bone peut être par exemple de type tunnel. Si le site souhaite déployer du routage statique, une route par défaut devra être configurée vers le réseau unicast. Pour le service multicast, le protocole PIM utilise des informations de routage pour connaître la topologie multicast et envoyer les messages protocolaires sur les bonnes interfaces. Si aucune autre information n'est ajoutée, le routeur du site va envoyer tous les messages PIM Join/Prune vers le réseau unicast car la route par défaut pointe dans cette direction. Il faut donc indiquer à ce routeur la topologie multicast. Ceci est fait en ajoutant une route statique multicast qui ne sera utilisée que pour le multicast. Cette route va peupler la table de routage multicast du routeur ou MRIB (Multicast Routing Information Base). Cette route n'est pas utilisée pour le forwarding des paquets unicast mais sera utilisée pour la construction de l'arbre et pour le routage des paquets multicast. En effet, le RPF check sera fait en utilisant les informations de routage contenues dans la MRIB. MBGP peut aussi être utilisé pour peupler la MRIB. La sub-address Family IPv6 multicast permet d'échanger les informations de routage pour le multicast.

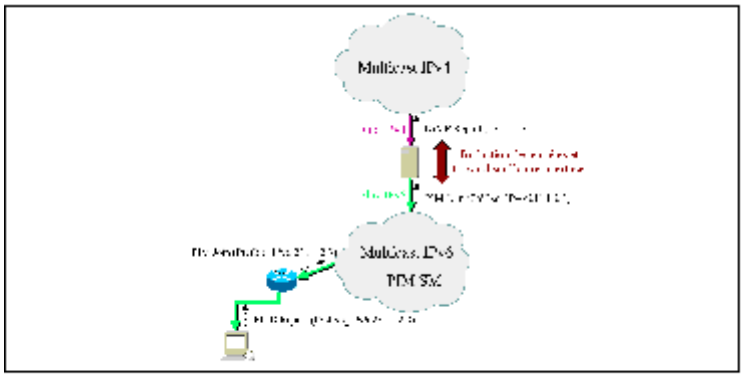


Figure 11-25 . Schéma explicatif de la passerelle dynamique

C) Déploiement d'un service fiable et efficace

Afin de déployer le protocole de routage multicast, il est important de déterminer quelle base d'information de routage unicast doit être utilisée par les routeurs multicast pour la vérification RPF. Si le service multicast n'est déployé qu'à l'intérieur d'un seul domaine (intra-domaine), la table de routage

unicast sous-jacente peut alors être utilisée dans la mesure où les topologies unicast et multicast sont congruentes. Si toutefois le service multicast doit couvrir plusieurs domaines (inter-domaine) les routeurs multicast, tels que les routeurs PIM-SM, peuvent s'appuyer sur une autre base d'information de routage qui sera alors utilisée uniquement pour la vérification RPF du routage multicast (on non pour la routage unicast). MBGP (ou BGP IPv6 multicast SAFI, voir [RFC 2858]) peut être utilisé pour générer de telles bases.

Lors du déploiement du routage multicast, il est aussi important de s'assurer de l'efficacité du routage mis en place. Par exemple dans le cas de PIM-SM, le placement des points de Rendez-vous (RP) peut avoir un impact important sur la charge du réseau. Afin d'éviter les surcharges sur un nœud particulier, il est possible de configurer plusieurs RP, chacun gérant uniquement un sous-ensemble d'adresses multicast.

La robustesse du service mis en place doit aussi être considérée. Par exemple dans le cas de PIM-SM, il est aussi possible de configurer plusieurs RP pour servir le même ensemble d'adresses multicast. Si le point de rendez-vous principal devient indisponible, un point de rendez-vous secondaire prend alors le relais afin de maintenir le service.

D) Services supplémentaires

Le routage multicast à lui seul ne suffit pas pour assurer un service multicast complet. Certains services supplémentaires comme l'annonce des sessions ou encore la gestion des adresses multicast peut être nécessaire.

L'annonce des sessions multicast peut se faire par simple publication sur page web, ou avec le protocole SAP et l'application SDR décrits plus haut dans ce chapitre.

La gestion des adresses multicast IPv6 dans un réseau est comme nous l'avons vu plus haut un problème complexe et il n'existe pas encore de solutions aujourd'hui pour permettre d'allouer des adresses multicast IPv6 pour les applications.

E) Interconnection à l'Internet multicast IPv6

Plusieurs réseaux multicast IPv6 régionaux ou internationaux, tel le M6Bone, sont d'ores et déjà déployés. Toute organisation souhaitant expérimenter le multicast IPv6 à grande échelle peut s'y rattacher, en attendant l'extension du service multicast IPv6 à tous les ISP.

Comme discuté précédemment, le protocole PIM-SM semble actuellement la solution la plus appropriée pour supporter le déploiement massif du multicast IPv6. En effet, ce seul protocole, associé à l'utilisation d'adresses multicast IPv6 « embedded RP », offre une solution globale permettant de s'affranchir du modèle traditionnel « intra-domaine / inter-domaine » du multicast IPv4.

XII) Sécurité

Profitant de la définition du protocole IPv6, l'IAB a mis l'accent sur la nécessité d'intégrer des services de sécurité dans le protocole IP en vue de protéger le trafic IP. Le réseau IPv4 étant largement déployé, il est vite apparu intéressant de définir des mécanismes de sécurité qui soient communs à la fois à IPv4 et IPv6. Ces mécanismes sont couramment désignés par les protocoles IPsec (*IP SECurity*).

Le placement de mécanismes de sécurité au niveau de IP présente l'avantage d'être exploitable par bon nombre d'applications et de mécanismes (seule l'auto-configuration peut poser un problème) et d'offrir ainsi un moyen de protection unique. En particulier, IPsec peuvent servir à protéger des VPNs (*Virtual Private Network*) dans un contexte d'interconnexion de sites ou de connexion à distance depuis un nomade.

Toutes les implémentations IPv6 conformes doivent obligatoirement intégrer IPsec. Par contre, les protocoles IPsec sont optionnels pour IPv4 et ne sont pas fournis en standard sur la plupart des systèmes courants. Lorsqu'IPv6 sera en place, il sera donc possible à tout utilisateur désirant des fonctions de sécurité d'avoir recours à IPsec.

Dans la suite du chapitre, deux extensions IPsec sont présentées de façon détaillée ainsi que plusieurs aspects liés à la gestion de la sécurité sur les réseaux IP, c'est-à-dire la gestion des associations de sécurité et la gestion des clés publiques. Deux architectures classiques de l'utilisation d'IPsec sont décrites. Afin d'illustrer nos propos, des exemples de configuration des protocoles IPsec basés sur le système d'exploitation BSD sont proposés. Le chapitre se termine par une critique d'IPsec et une comparaison entre l'usage qu'on peut faire d'IPsec dans IPv4 et dans IPv6.

1) Généralités

Pour une bonne compréhension de ce chapitre, quelques notions de base de sécurité sont nécessaire. En cryptographie, deux familles d'algorithmes de chiffrement existent :

- les algorithmes à clés symétriques qui utilisent le même secret (ou clé de chiffrement) pour chiffrer et déchiffrer des données et
- les algorithmes à clés asymétriques qui sont basés sur un couple de clés privées et publiques. La clé privée ne doit être connue que d'une seule entité et la clé publique peut être largement diffusée sur le réseau. Ces clés sont complémentaires dans la mesure où le chiffrement avec l'une de ces clés nécessite le déchiffrement avec l'autre clé.

Les services de sécurité étudiés sont :

- La confidentialité des données, qui permet, grâce à des mécanismes de chiffrement, de protéger les données émises sur le réseau de telle sorte qu'elles ne soient compréhensibles que par les entités du réseau autorisées. Pour des raisons de performance, elle fait généralement appel à des

algorithmes à clés symétriques. La clé utilisée pour chiffrer les paquets IP est appelée clé de session. Comme cette clé est fréquemment utilisée et comme en sécurité, plus une clé est utilisée plus elle est vulnérable, il est nécessaire de régulièrement renouveler les clés de session. Pour cela, on distingue deux niveaux de clés, les clés de session qui sont de courte durée et qui sont exploitées de façon intensive pour chiffrer les données et les clés de longue durée qui sont utilisées avec parcimonie exclusivement pour renouveler les clés de session de façon protégée.

- La confidentialité du flux de données, qui garantit qu'aucune information portant sur une communication (quantité d'informations échangées, fréquence,...) ne peut être déduite par analyse de trafic.
- L'authentification de l'origine des données, qui garantit que les données reçues proviennent de l'entité déclarée. Ce service consiste à adjoindre aux données émises un authentificateur (ICV : *Integrity Check Value*), terme générique pour désigner soit un code d'authentification de message (MAC : *Message Authentication Code*), soit une signature numérique.

  Dans les deux cas, on utilise une fonction de hachage, c'est-à-dire, une fonction satisfaisant plusieurs propriétés (1), et qui, à partir d'un message de taille quelconque, retourne le condensat (ou empreinte) de ce message avec une taille de condensat fixe dépendant de la fonction de hachage. SHA-1 (*Secure Hash Standard*) en est un exemple avec une taille de condensat fixée à 160 bits.

- Dans le cas du MAC, l'authentificateur consiste en une suite de concaténations des données avec une clé symétrique et de calculs de condensats à l'aide d'une fonction de hachage. Ce type de MAC est appelé HMAC, et en particulier si la fonction de hachage SHA-1 est utilisée, on parle de HMAC-SHA1. Noter que seul le partenaire en possession de la clé symétrique utilisée sera capable de vérifier l'authenticité du MAC.
- L'utilisation de la cryptographie à clé publique pour produire une signature numérique est également possible. La signature est construite en chiffrant un condensat des données à l'aide de la clé privée. La cryptographie à clés publiques est avantageuse car elle offre en plus des services d'authentification et d'intégrité, le service de non répudiation (voir ci-dessous). Cependant, du fait qu'elle est gourmande en ressources, elle n'est pas utilisée en pratique pour protéger les volumineux échanges de données d'un réseau.
- L'authentification mutuelle, qui permet à deux entités de se prouver mutuellement leur identité. Ce service est généralement rendu par l'utilisation d'"échange d'authentification" qui implique un certain dialogue entre les entités.
- L'intégrité des données, qui garantit que les données reçues n'ont subi aucune modification au cours de leur transfert sur le réseau. Ce service fait lui aussi appel à des mécanismes d'authentificateur en général.
- La prévention contre le rejeu de données, qui assure que les données reçues n'ont pas été précédemment jouées, c'est-à-dire déjà reçues. Un rejeu pourrait se produire si un intrus réussissait à capturer des datagrammes et à les réémettre vers le même destinataire.
- La non répudiation, qui permet de prouver, en cas de litige, que des données ont bien été émises ou reçues par des entités.

Pour fournir ces services de sécurité, on a recours à des mécanismes cryptographiques dont la plupart nécessite de partager des clés secrètes, ce qui implique de mettre en œuvre un protocole d'échange de clé. Il existe de nombreux protocoles d'échange de clé, qui se différencient suivant les pré-requis qu'ils imposent et les propriétés qu'ils vérifient.

Parmi ces propriétés, celle dite de *Perfect Forward Secrecy* (PFS) est particulièrement intéressante. Elle garantit que la découverte par un adversaire d'une ou des clés de longue durée ne compromettra pas les clés de session générées précédemment. Ainsi le trafic chiffré avec ces clés de session restera confidentiel. On considère généralement que cette propriété assure également que la découverte d'une clé de session ne compromet pas les clés de longue durée.

D'autre part, on parle d'anonymat ou protection de l'identité (*Identity Protection*) lorsque le protocole garantit qu'aucune information sur les identités des entités en communication ne pourra être déduite des échanges effectués sur le réseau.

La plupart des protocoles d'échange de clé développés pour la protection des échanges sous IP se basent sur le protocole Diffie-Hellman. Inventé en 1976 par Diffie et Hellman, ce protocole permet à deux entités de générer un secret partagé sans partager d'information préalable. Chaque entité possède pour cela un couple (valeur publique, valeur privée) pour lequel la valeur privée ne peut être déduite de la valeur publique. Chacun envoie sa valeur publique à son interlocuteur. A l'aide de la valeur publique de son interlocuteur et de sa propre valeur privée, chaque entité calcule un secret, qui est également calculé par son interlocuteur. Le secret partagé ainsi généré peut être utilisé pour dériver une ou plusieurs clés secrètes. Une personne qui espionnerait les échanges (les valeurs publiques) sur le réseau ne pourrait pas déduire le secret partagé car cela nécessite la connaissance d'une des valeurs privées.

En revanche, Diffie-Hellman est vulnérable à l'attaque de l'intercepteur, qui consiste, pour une personne malveillante, à intercepter les échanges de valeurs publiques et à faire croire aux entités légitimes que la valeur publique de leur interlocuteur est la sienne. Ainsi, à la fin du protocole Diffie-Hellman, chaque entité partagera une clé propre avec l'intercepteur. Chaque message émis par une entité sera intercepté, déchiffré avec la clé partagée par la source et l'intercepteur, puis rechiffré avec la clé partagée par le destinataire et l'intercepteur. Les entités auront l'impression de communiquer ensemble directement de façon sûre alors que l'intercepteur pourra en fait lire tous les messages, voire même forger de faux messages. Une façon de résoudre ce problème de sécurité est d'authentifier les valeurs publiques utilisées pour la génération du secret partagé.

2) Analyse des risques

Par la suite, il est fait référence à un "intrus". Un intrus désigne en fait toute personne adoptant un comportement malveillant sur le réseau. Trois attaques classiques peuvent être réalisées dans un environnement de réseaux IP :

- IP sniffing : L'IP sniffing consiste pour un intrus à "écouter" le trafic en transit sur un réseau. Il peut ainsi parvenir à prendre connaissance du contenu de messages électroniques échangés sur le réseau ou des mots de passe utilisés par ses collègues.

  L'IP sniffing peut être réalisé de deux façons :

- L'intrus est placé sur un réseau permettant naturellement la diffusion de données (Token Ring ou Ethernet, par exemple). L'intrus peut alors, à partir de sa station de travail, récupérer l'ensemble du trafic en transit sur le réseau et l'analyser.
- L'intrus réussit à utiliser un analyseur de protocoles réseau à l'insu des administrateurs du réseau. Il peut alors capturer l'ensemble du trafic en transit. Bien entendu, cette analyse est limitée au trafic échangé sur le segment de réseau sur lequel se trouve branché l'analyseur.

- IP spoofing : L'IP spoofing consiste à usurper l'identité d'une personne. Le but de cette attaque est d'établir une communication avec une station en se faisant passer pour quelqu'un d'autre ou d'insérer des données dans une communication existante. Plusieurs techniques sont utilisables :
  - La modification de l'adresse matérielle de la station de l'intrus. Ainsi, la station de l'intrus peut récupérer les données à destination d'une autre station de même adresse matérielle.
  - La création de messages ICMP de toute pièce en vue de rediriger des paquets IP vers la station de l'intrus.
  - La compromission d'un serveur de nom (DNS) peut servir à rediriger une requête DNS vers une station contrôlée par un intrus qui peut alors retourner une mauvaise adresse IP à la station initiatrice de la requête DNS.
- IP flooding : Le but de l'IP flooding est d'envoyer une multitude de paquets IP vers une même destination de telle sorte que le traitement de ces paquets empêche une entité du réseau (un routeur ou la station destinatrice) de traiter les paquets IP légitimes.

  Dans le cas de l'utilisation de la pile TCP/IP, cette attaque peut prendre la forme du SYN flooding qui consiste pour une station cliente à émettre une multitude de messages SYN de demande d'ouverture de connexion TCP. Le serveur destinataire doit retourner, pour chaque message SYN reçu, un message SYN-ACK d'acquittement et conserver dans sa mémoire système l'ensemble des connexions en attente d'un message ACK d'acquittement de la part du client. Cette mémoire système étant de taille finie, l'envoi d'un grand nombre de messages SYN-ACK conduit à sa saturation et à l'impossibilité d'accepter d'autres demandes d'ouverture de connexion TCP.
  La meilleure façon de réaliser cette attaque est de la combiner avec de l'IP spoofing. C'est-à-dire, un intrus envoie des messages SYN en faisant croire qu'ils proviennent d'autres stations clientes. Les messages SYN apparaissent ainsi légitimes au serveur. Cependant les stations clientes dont l'identité est usurpée sont incapables de répondre aux messages SYN-ACK.

  Le serveur reste donc en attente de messages d'acquittement et si sa mémoire est pleine, il refuse toute nouvelle ouverture de connexion. Normalement, comme un temporisateur est associé à chaque connexion en attente d'un acquittement, la mémoire système est amenée à se vider progressivement, ce qui permet au serveur d'accepter de nouvelles ouvertures de connexions TCP. L'intrus peut cependant empêcher le serveur de servir ces nouvelles connexions en émettant des demandes d'ouverture de connexion TCP à un rythme plus soutenu que le temps d'expiration des temporisateurs.

  Si l'IP flooding (ou SYN flooding) est combiné à l'IP spoofing, il est impossible, pour le destinataire, de connaître l'adresse source exacte des paquets IP. De ce fait, à moins que le destinataire ne limite ses échanges avec certaines stations, il lui est impossible de contrer ce type d'attaques.

(1) Une fonction de hachage se doit de satisfaire les deux propriétés suivantes :

- La modification d'un seul bit d'un message aboutit à un condensat dont au moins la moitié des bits est différent ;

- A partir d'un condesat, il n'est pas possible de remonter au message correspondant.

3) Orientations choisies par l'IETF

Comme le montre le tableau Services de sécurité pour la protection des réseaux, pour se prémunir des attaques IP spoofing et IP sniffing (l'IP flooding étant très difficile à contrer), il est nécessaire d'introduire les services de sécurité suivants (RFC 2401) :

- confidentialité des données ;
- intégrité des données ;
- authentification de l'origine des données.

Services de sécurité pour la protection des réseaux		
Les attaques	**IP sniffing**	**IP spoofing**
Les services de sécurité	confidentialité	authentification/intégrité, confidentialité, détection de rejeu

Pour introduire ces services de sécurité et assurer la protection des échanges de paquets IP, l'IETF a été amené à définir deux nouvelles extensions IP de sécurité :

- L'extension d'authentification ou extension AH pour *Authentication Header*, rend les services d'authentification, intégrité, et optionnellement détection de rejeu. Selon la méthode d'authentification utilisée, il peut également offrir le service de non répudiation.
- L'extension de confidentialité ou extension ESP pour *Encapsulating Security Payload*, peut rendre les services de confidentialité, intégrité, authentification, et détection de rejeu et garantir de façon limitée la confidentialité du flux.

La définition de deux extensions au lieu d'une seule est judicieuse du fait que la législation appliquée pour le service de confidentialité est plus stricte que celle appliquée pour les services d'intégrité/authentification. Selon les pays, l'exportation, l'importation et l'utilisation de moyens cryptographiques sont réglementées. En France, par exemple, depuis le 27 juillet 1996 (1), la cryptographie peut être librement utilisée pour authentifier ou prouver l'intégrité d'un message. L'usage de la cryptographie à des fins de chiffrement de messages a toujours été plus strict et dépendant de la longueur des clés de chiffrement utilisées, mais d'année en année, la réglementation s'est assouplie, et depuis le 21 juin 2004, date à laquelle la loi sur la confiance en l'économie numérique (LCEN) a été promulguée, les utilisateurs peuvent librement utiliser tout matériel leur permettant d'effectuer du chiffrement, et ce, quelle que soit la longueur des clés de chiffrement. Par contre, les fournisseurs de ce type de matériel sont

tenus de déclarer ce matériel (si la longueur de clés de chiffrement reste inférieure à 128 bits), voire de demander une autorisation (si la longueur des clés dépasse 128 bits).

Grâce à la distinction de ces deux extensions pour les services de confidentialité et d'intégrité/authentification, deux utilisateurs qui ne sont pas autorisés à assurer la confidentialité de leurs messages du fait de la législation de leur(s) pays peuvent toujours les protéger en intégrité/authentification.

Ces extensions sont utilisées pour protéger des communications. Cette protection peut être assurée, comme le montre la figure Différents modes de protection :

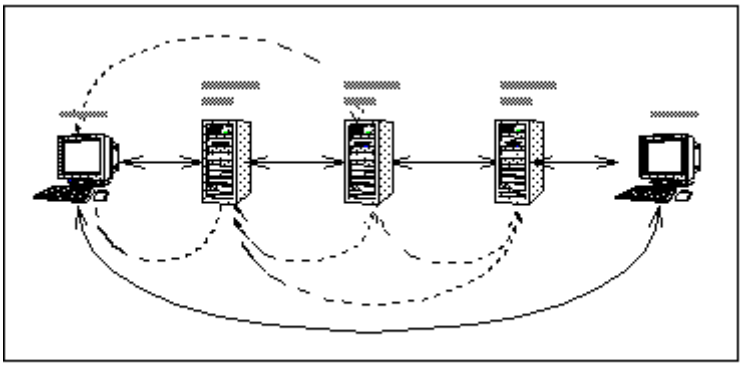


Figure 12-1 . Différents modes de protection

- de bout en bout, entre les deux stations extrémités en communication, auquel cas ce sont les stations qui introduisent et extraient les extensions de sécurité dans les paquets IP. Ce mode de protection est habituellement exploité dans des cas bien particuliers comme pour assurer la sécurité de la signalisation liée à la mobilité IPv6.
- sur des segments de réseaux entre deux équipements de réseau IP réalisant de la sécurité qui seront par la suite désignés par "passerelles de sécurité". Ces passerelles (*gateway* en anglais) peuvent être des routeurs ou des pare-feux (*firewall*). Les extensions de sécurité sont alors échangées entre passerelles de sécurité, c'est-à-dire introduites par une passerelle et extraites par l'autre passerelle. Ce type de protection est plus courant que le premier et correspond à un schéma classique de VPN (*Virtual Private Network*) où deux sites distants disposant chacun d'une passerelle sont interconnectés.
- entre une station et une ou plusieurs passerelle(s) de sécurité. Ce schéma est également couramment utilisé pour protéger les communications entre un nomade et son réseau privé d'entreprise.

Pour protéger les communications sur le réseau IP, deux modes de protection existent :

- Le mode transport permet de protéger la charge utile du paquet et certains champs de son en-tête ;
- Le mode tunnel assure la protection des communications sur un tunnel IP. C'est-à-dire, la protection porte sur tous les champs du paquet IP arrivant à l'entrée d'un tunnel et sur certains champs du nouveau paquet IP construit pour encapsuler le paquet IP entrant.

Le mode transport n'est utilisable qu'entre deux équipements terminaux en communication car, en cas d'utilisation sur des équipements intermédiaires, on courrait le risque, suivant les aléas du routage, que le paquet atteigne sa destination finale sans avoir traversé la passerelle chargée de le déchiffrer. Le mode

tunnel pallie ce problème en encapsulant chaque paquet IP dans un nouveau paquet, pour lequel les adresses source et destination sont celles des équipements IPsec.

Il faut noter que la protection d'une communication est toujours réalisée par des entités du réseau (stations ou passerelles) travaillant deux par deux. Cette protection nécessite que ces entités aient au préalable convenu de la liste des services de sécurité à appliquer à la communication ainsi que des mécanismes de sécurité adéquats (algorithmes de chiffrement, signature numérique). L'ensemble des services et mécanismes de sécurité choisis forme l'association de sécurité de la communication (voir la section suivante). Une fois l'une (des) association(s) de sécurité convenue(s), les entités réalisant la sécurité peuvent ensuite construire le (les) extension(s) de sécurité appropriée(s) et le(s) extraire en vue de leur vérification.

(1) article 17 de la loi de réglementation des télécommunications 96-659 du 26 juillet 1996, Journal Officiel du 27 juillet 1996

4) Association de sécurité

Une association de sécurité IPsec contient le type d'extension de sécurité à mettre en place sur une communication ainsi que les services et les paramètres de sécurité (algorithmes et clés de chiffrement, données de synchronisation,...) en vue de protéger les échanges de données effectués. Elle doit donc être connue des entités (par exemple : stations, passerelles de sécurité) chargées de la protection de la communication.

Une association de sécurité est identifiée de façon unique par un triplet comprenant un indice de paramètres de sécurité SPI (*Security Parameters Index*), parfois qualifié de SAID (*Security Association IDentifier*), l'adresse du destinataire d'un paquet IP et le protocole de sécurité AH ou ESP. Un équipement de sécurité IPv6 peut ainsi participer à plusieurs associations de sécurité et peut protéger une communication avec une ou plusieurs association(s) de sécurité. Si par exemple les deux extensions de sécurité AH et ESP sont utilisées simultanément entre deux équipements de sécurité pour une même communication, il est alors nécessaire que les équipements gèrent deux associations de sécurité distinctes.

Une association de sécurité est unidirectionnelle. Une communication bidirectionnelle passée entre deux stations A et B nécessite donc l'intervention de deux associations de sécurité, celle utilisée par A pour protéger les datagrammes de A vers B et celle utilisée par B pour la protection des datagrammes de B vers A.

A) Contenu d'une association de sécurité

Une association de sécurité contient entre autres les paramètres suivants :

- l'algorithme d'authentification, les clés de chiffrement,... utilisés pour générer l'extension AH ;
- l'algorithme de chiffrement, les clés de chiffrement, le mode de chiffrement, les paramètres de synchronisation,... utilisés pour générer l'extension ESP ;

- l'algorithme d'authentification, les clés de chiffrement,... utilisés pour générer l'extension ESP si le service d'authentification est sélectionné ;
- la durée de vie de l'association de sécurité : pour éviter que des clés de chiffrement ne soient utilisées trop longtemps, ce qui affaiblirait le niveau de sécurité des communications, il est nécessaire de convenir périodiquement d'une nouvelle association de sécurité ;
- le mode du protocole IPsec, à savoir : tunnel, transport ou wildcard. Le mode wildcard signifie que le choix du mode de protection est déterminé par l'application ; ce mode n'est utilisable qu'à partir d'une station de travail, mais il n'est pas disponible actuellement sur les implémentations courantes, car trop complexe de mise en œuvre.

Afin qu'une passerelle de sécurité puisse interpréter correctement les extensions de sécurité reçues, l'association de sécurité (c'est-à-dire le SPI) ayant servi à leur construction doit être mentionnée dans chacune des extensions de sécurité.

B) Choix d'une association de sécurité

Le choix de l'association de sécurité au niveau de la station émettrice ou d'une passerelle de sécurité peut théoriquement dépendre des paramètres suivants (appelés sélecteurs dans le RFC 2401) :

- l'adresse IP de la source locale qui peut être une adresse unicast, anycast, ou multicast, voire un ensemble d'adresses. Si le choix dépend exclusivement de cette adresse, la même association de sécurité est attribuée à deux utilisateurs connectés sur la même station et émettant vers le même destinataire. Cette approche souffre d'un handicap majeur. En effet, un utilisateur malveillant peut déduire la clé utilisée entre deux stations en se connectant sur cette station et en effectuant une attaque du type *Chosen Plaintext Attack* qui consiste à soumettre un texte en clair particulier à la station et à analyser le texte chiffré correspondant (celui émis sur le réseau) pour trouver la clé de chiffrement utilisée. La connaissance de cette clé lui permet ensuite de déchiffrer tout le trafic échangé entre les deux stations (dans un sens uniquement).
- l'identité de l'utilisateur (par exemple un nom DNS ou X.500). Cette approche est beaucoup plus fiable que l'approche précédente du fait que l'attaque *Chosen Plaintext Attack* n'est plus réalisable.
- l'identité de l'équipement utilisé (nom DNS ou X.500).
- les numéros de ports source et destination (e.g., TCP/UDP).
- le protocole de niveau transport obtenu par le champ en-tête suivant :
- l'adresse IP de l'équipement distant qui peut être une adresse unicast, anycast, ou multicast, voire un ensemble d'adresses. Les premières générations de passerelles IPsec se contentaient de ce type de sélecteurs, mais les nouvelles versions se sont enrichies en considérant, en plus de l'adresse de destination, les numéros de port de destination.
- le niveau de sensibilité des données (identifié par les étiquettes normalisées IPSO/CIPSO du RFC 1108)

La majorité des équipements IPsec aujourd'hui considère généralement que les sélecteurs sont les adresses IP de destination, les numéros de protocole et numéros de port.

C) Bases de données

L'IETF conseille aux constructeurs qui souhaiteraient implémenter les IPsec de définir deux bases de données de sécurité afin que l'équipement de sécurité puisse déterminer en deux étapes l'association de sécurité à appliquer pour construire ou extraire les extensions de sécurité. Du fait que ces bases de données dépendent beaucoup du choix de l'implémentation, les constructeurs sont libres de les implémenter ou pas. Les deux bases de données qui permettent, lors de la réception d'un paquet IP, de déterminer l'association de sécurité à appliquer, sont les suivantes :

- le SPD (Security Policy Database) précise, en fonction du "sélecteur" (le critère de sélection d'une association de sécurité) choisi, les actions à effectuer sur un paquet. Trois actions sont possibles :
  - Soit le trafic n'est pas autorisé à être émis par une station, à transiter via une passerelle de sécurité ou à être reçu par une application : il est donc supprimé (discard).
  - Soit le trafic est autorisé mais ne nécessite aucun service de sécurité ; il est alors ignoré par l'ensemble des équipements/logiciel implémentant les IPsec (bypass IPsec).
  - Soit le trafic nécessite une protection IPsec (apply IPsec) auquel cas le SPD spécifie pour chaque sélecteur l'association de sécurité ou les associations de sécurité à appliquer (précisée(s) dans la base de données SAD).
- le SAD (Security Association Database) contient l'ensemble des associations de sécurité et précise pour chacune d'elles, les services et mécanismes de sécurité à appliquer. Ainsi si d'après le SPD, un paquet nécessite une protection, il est facile de retrouver la (les) associations de sécurité à appliquer à l'aide du SPI, de l'adresse de destination et du protocole AH ou ESP sélectionné. Il reste alors à appliquer les services de sécurité adéquats.

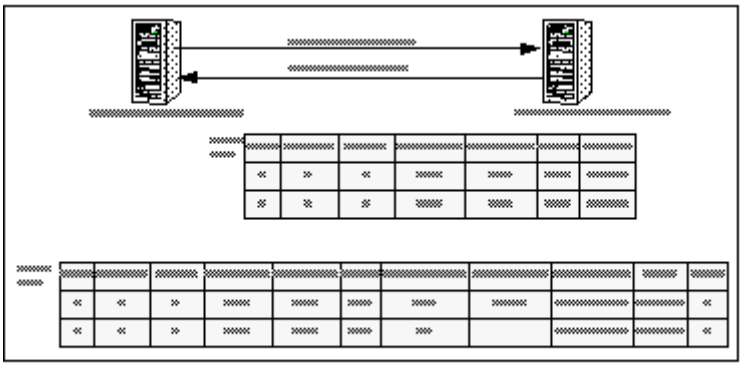


Figure 12-2 . Différents bases d'IPsec

La figure Différents bases d'IPsec montre un exemple de bases de données partagées par deux équipements IPsec A et B. La base SPD indique que le trafic émis de A vers B est protégé par ESP en mode tunnel et le trafic de B vers A par AH en mode tunnel. La base SAD précise les outils de mise en œuvre avec l'algorithme de chiffrement triple DES (3DES) pour ESP et la fonction de hachage HMAC-SHA1 utilisée par ESP et AH.

D) Évolutions attendues

Le RFC 2401 est en cours de révision avec les travaux en cours sur [RFC2401bis]. Le RFC2401bis ne change rien fondamentalement au standard, mais définit de façon plus fine les bases de données afin de permettre la mise en œuvre de scenarios IPsec plus complexes, d'améliorer les performances et de simplifier les implémentations. En effet, elle considère que la base SPD est divisée en trois sous bases de données, SPD-S (`S` pour *Secure*) qui répertorie l'ensemble des flux devant être protégés par IPsec, SPD-O (`O` pour *Outbound*) qui liste l'ensemble des flux sortants qui ne nécessitent pas de protection ou qui sont interdits et SPD-I (`I` pour *Inbound*) qui renseigne sur l'ensemble des flux entrants qui ne nécessitent pas de protection ou qui sont interdits. Ainsi suivant que les paquets entrent ou sortent du réseau, le traitement sera optimisé car il sera fait appel à une base SPD spécialisée (SPD-O ou SPD-I) et au besoin à SPD-S. En plus des bases SPD et SAD, la base PAD (*Peer Authorization Database*) est définie pour permettre de faire le lien entre le protocole de gestion des associations de sécurité comme IKE (voir le chapitre Gestion des associations et des clés) et la base SPD. Chaque entrée de SPD peut être exprimée à l'aide d'un ou plusieurs ensemble(s) de sélecteurs associés à une liste de plages de valeurs, contrairement au RFC 2401 où un seul sélecteur à la fois était possible. Enfin, une architecture logicielle modulaire est proposée ; elle introduit la notion de module forwarding indépendant du traitement IPsec qui a pour finalité de déterminer l'interface d'acheminement des paquets et ainsi de mettre en œuvre des traitements IPsec plus complexes sur les paquets. En particulier, il est possible d'appliquer deux traitements IPsec sur un même paquet.

5) Extension d'authentification AH

L'extension d'authentification (AH : Authentication Header) décrite dans le RFC 2402 permet de s'assurer que l'émetteur du message est bien celui qu'il prétend être. Elle sert aussi au contrôle d'intégrité pour garantir au récepteur que personne n'a modifié le contenu d'un message lors de son transfert sur le réseau. Elle peut optionnellement être utilisée pour détecter les rejeux.

Le principe de l'authentification est relativement simple. L'émetteur calcule un authentificateur sur un datagramme et l'émet avec le datagramme sur lequel il porte. Le récepteur récupère cette valeur et vérifie qu'elle est correcte. C'est-à-dire, si un MAC est utilisé, il lui suffit de calculer de son côté le MAC sur le même datagramme à l'aide de la clé symétrique partagée et de le comparer avec le MAC reçu. Si le mécanisme de signature numérique est employé, le récepteur doit alors récupérer la signature, la déchiffrer avec la clé publique de l'émetteur et comparer le condensat ainsi obtenu avec celui calculé de son côté sur le datagramme reçu. Si les deux MAC, ou les deux condensats diffèrent, soit l'émetteur ne possède pas la bonne clé, soit le message a subi des modifications en chemin.

A) Positionnement de l'extension AH

Le positionnement de l'extension AH dans les datagrammes IP est étudié ici, avec l'étendue de la protection assurée dans les modes transport et tunnel.

- L'extension AH en mode transport permet entre autres de rendre le service d'intégrité sur les paquets IP. En fait, l'intégrité du paquet n'est assurée que pour les données de niveau transport et les champs -- en-tête du paquet et extensions -- qui ne sont pas amenés à subir de modifications de la part du réseau lors de leur transfert. L'extension AH doit être placée après l'en-tête IPv6, les extensions proche-en-proche, routage, fragmentation et avant les données de niveau transport, comme le montre la figure Positionnement de l'extension AH en modes transport et tunnel. Seule l'extension destination peut être introduite avant ou après l'extension AH.

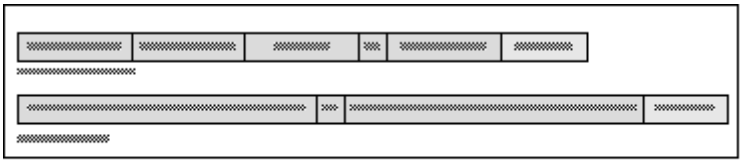


Figure 12-3 . Positionnement de l'extension AH en modes transport et tunnel

- Le mode tunnel consiste à encapsuler le paquet IP original constitué d'un en-tête original et du champ données dans un nouveau paquet IP, dont les adresses source et destination sont celles des équipements mettant en œuvre IPsec. L'extension AH permet alors de protéger en intégrité/authentification/détection de rejeu la totalité du paquet IP original et la partie du nouvel en-tête/extensions du paquet IP qui n'est pas amenée à subir de modifications en cours de transit. Il est placé après les extensions propres au nouveau paquet IP formé (cf. figure Positionnement de l'extension AH en modes transport et tunnel).

B) Contenu de l'extension AH

L'extension AH est composée des champs suivants (cf. figure Format de l'extension d'authentification AH) :

Image:CS130.gif

- Le champ longueur de l'extension indique la longueur du champ authentificateur exprimée en nombre de mots de 32 bits.
- Le champ réservé est réservé pour une utilisation future et est mis à 0 à l'émission.
- Le champ indice des paramètres de sécurité (SPI) combiné à l'adresse du (des) destinataire(s) et du protocole IPsec (AH ou ESP), identifie l'association de sécurité utilisée pour construire cette extension.
- Le numéro de séquence sur 32 bits permet de détecter les rejeux de paquet IP.
- Le champ authentificateur garantit l'intégrité du paquet ; il est calculé sur le datagramme IP à l'aide de l'algorithme et de la clé correspondant à l'association de sécurité (SPI + adresse(s) de destination + AH). C'est également l'association de sécurité qui précise la taille de ce champ.

L'extension AH n'offre pas le service de confidentialité. Elle ne permet pas de chiffrer les données transportées dans le paquet et ne protège donc pas ces données contre d'éventuelles écoutes effectuées sur le réseau.

La détection de rejeu est optionnelle du fait que l'équipement de sécurité qui produit l'extension AH doit remplir le champ numéro de séquence correctement tandis que l'équipement qui extrait l'extension n'est pas tenu de vérifier la bonne valeur de ce champ. La détection consiste en un numéro de séquence incrémenté à chaque paquet. Le récepteur peut vérifier sa croissante stricte ou mieux, gérer une fenêtre en rejetant tous les paquets avec un numéro de séquence déjà reçu. L'association de sécurité doit être renégociée quand le numéro de séquence atteint la valeur maximale. Une détection de rejeu n'est donc possible que si le récepteur est doté d'un module de gestion des associations de sécurité IKE qui renégociera automatiquement une association de sécurité le moment venu.

Les algorithmes utilisés pour le calcul de l'authentificateur doivent être particulièrement robustes. L'IETF fait la distinction entre les communications point-à-point et point-à- multipoint et propose, pour protéger les communications point-à-point, que les équipements de sécurité disposent au moins du MAC (*Message Authentication Code*) basé sur des algorithmes symétriques (par exemple DES) et des fonctions de hachage (RFC 2403, RFC 2404) du type MD5 (*Message Digest 5*) ou SHA-1 (*Secure Hash Standard*). On parle alors des algorithmes HMAC-MD5 et HMAC-SHA-1. Noter que la préférence est donnée à l'algorithme HMAC-SHA1 car SHA-1 fournit un résultat sur 20 octets (contre 16 octets pour MD5) et de ce fait rend plus difficile l'attaque visant à partir d'un condensat à trouver une chaîne binaire satisfaisante. Pour la protection des communications point-à-multipoint, l'IETF préconise l'usage de fonctions de hachage combinées à des algorithmes asymétriques, ceci pour échapper au difficile problème de gestion des clés de groupe (symétriques).

Pour générer l'extension AH, un équipement de sécurité calcule un authentificateur sur un datagramme sous la forme qu'il aurait à la réception finale (une extension de routage aura donc les adresses dans un ordre particulier) avec tous les champs pouvant changer en chemin mis à zéro (c'est-à-dire les champs classe, identificateur de flux, et nombre de sauts). Le champ authentificateur est également mis à zéro pour ce calcul. La vérification se fait par le même procédé.

Du fait que les champs considérés comme "non modifiables" par le schéma d'authentification n'incluent pas les adresses IP, la traduction d'adresses (NAT : *Network Address Translator*) et l'utilisation de l'extension AH ne sont pas toujours exploitables simultanément, en particulier lorsque les équipements IPsec communiquent au travers d'un traducteur d'adresses. En effet, l'authentificateur est calculé sur les adresses IP des paquets. Comme le traducteur d'adresses modifie l'adresse source ou destination des paquets IP, l'authentificateur reçu par l'un ou l'autre des équipements IPsec sera toujours invalide et le paquet sera tout le temps rejeté. Une architecture avec deux équipements IPsec de part et d'autre d'un traducteur d'adresses est courante aujourd'hui. Typiquement, il peut s'agir d'un nomade IPsec qui se connecte depuis un aéroport équipé d'un NAT à son réseau d'entreprise via un VPN IPsec.

Face à ce problème d'incompatibilité avec le NAT et à l'impossibilité de faire du chiffrement de contenu avec AH, les entreprises bien souvent préfèrent configurer leurs équipements IPsec avec l'extension ESP. Ainsi l'extension AH se trouve confiner à certains usages particuliers comme la protection des messages de découverte de routeurs (*router advertisement*).

C) Évolutions prévues

Dans sa prochaine version du [RFC2402bis], l'extension AH distingue l'identification qui doit être faite d'une association de sécurité suivant qu'elle est d'usage unicast ou multicast. Contrairement au RFC 2402, une association de sécurité unicast peut être identifiée par la seule valeur du SPI ou bien par le couple : SPI et protocole IPsec. Quant à l'association de sécurité multicast, elle est identifiée par la valeur du SPI, l'adresse de destination et optionnellement l'adresse source.

Pour les communications à très haut-débits, un numéro de séquence de 32 bits peut s'avérer insuffisant car un cycle sur ce numéro de séquence peut être atteint rapidement. Pour exemple, au débit de 1Gb/s nécessiterait qu'une nouvelle association de sécurité soit négociée toutes les heures. Pour éviter un renouvellement fréquent d'associations de sécurité, il a été décidé dans la prochaine version de AH d'avoir la possibilité d'avoir un numéro de séquence sur 64 bits au lieu de 32. Noter que la taille du numéro de séquence est négociée sous la forme d'une extension de numéro de séquence (ESN : *Extended Sequence Number*) grâce au protocole de gestion des associations de sécurité. Noter qu'astucieusement, la taille du numéro de cette extension ESN n'a aucun impact sur le format de l'extension AH. En effet, dans le cas de la sélection de ESN, seuls les 32 bits de poids faible sont transmis dans l'extension AH ; les 32 bits de poids fort sont en effet maintenus de part et d'autre sur les équipements IPsec comme compteur local et servent au calcul de l'authentificateur.

6) Extension de confidentialité ESP

L'extension ESP (*Encapsulating Security Payload*) décrite dans le RFC 2406 permet de chiffrer l'ensemble des paquets ou leur partie transport et de garantir l'authentification et l'intégrité de ces paquets. Cette extension permet optionnellement de détecter les rejeux (à condition que le service d'authentification soit assuré) et garantit de façon limitée la confidentialité du flux.

Pour obtenir ces services de sécurité, il est nécessaire, avant d'émettre un paquet IP sur le réseau, de chiffrer les données à protéger, de calculer un authentificateur et d'encapsuler ces informations dans l'en-tête de confidentialité. Cela nécessite bien entendu l'existence d'une association de sécurité précisant entre autres le(s) algorithme(s) de chiffrement, la (les) clé(s) et un indice de paramètres de sécurité.

A) Deux modes de protection

L'extension ESP est composée d'un en-tête ESP, d'une queue ESP et d'un authentificateur ESP. Suivant le mode de protection sélectionné, l'étendue des champs protégés diffère :

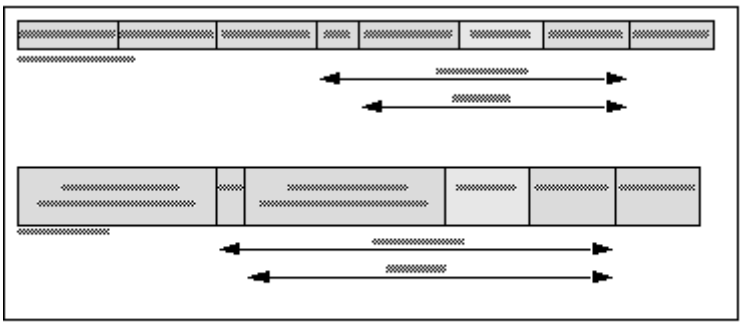


Figure 12-5 . Positionnement de l'extension ESP en modes transport et tunnel

- En mode transport, seules les données de niveau transport du paquet IP (de type TCP, UDP, ICMP) sont protégées. Plus précisément, le chiffrement porte sur les données et sur la queue ESP avec la possibilité de porter sur l'extension destination à condition que celle-ci soit placée dans l'extension ESP (cf. figure Positionnement de l'extension ESP en modes transport et tunnel). La protection en intégrité/authentification porte sur toute l'extension ESP excepté l'authentificateur placé dans le champ authentificateur. Elle assure ainsi la protection des données de niveau transport du paquet IP. L'extension ESP est insérée dans le paquet IP juste après l'en-tête du paquet IP et les extensions avec la possibilité d'avoir l'extension destination placée juste avant l'extension ESP ou encapsulée dans l'extension ESP en première position.

- En mode tunnel, la protection porte sur tout le paquet IP original. C'est-à-dire, le paquet IP original est chiffré avant d'être encapsulé dans l'extension ESP. Cette extension est alors placée dans un nouveau paquet IP dont les en-têtes IP sont en clair (non chiffrés) pour permettre au réseau IP de correctement acheminer le paquet (cf. figure Positionnement de l'extension ESP en modes transport et tunnel). Il s'agit du classique chiffrement des artères. La protection en intégrité/authentification porte sur l'extension ESP (comprenant le paquet IP original) excepté l'authentificateur ESP. L'extension ESP rend le service de confidentialité du flux de façon limitée en ce sens que ce service n'est rendu qu'en mode tunnel et ne porte que sur la confidentialité des adresses IP. En effet, l'extension ESP en mode tunnel réalise le chiffrement de l'intégralité des paquets IP originaux, y compris leurs adresses IP. De ce fait, si les équipements qui mettent en oeuvre IPsec ne sont pas l'émetteur et le destinataire final des paquets, leurs adresses masqueront celles de ces derniers. Un intrus placé en écoute sur le réseau au niveau d'un tunnel IP (entre les passerelles de sécurité) ne pourra pas dans ce cas là connaître les adresses des stations en communication.

B) Contenu de l'extension ESP

L'extension ESP est composée des champs suivants (cf. figure Format de l'extension d'ESP) :

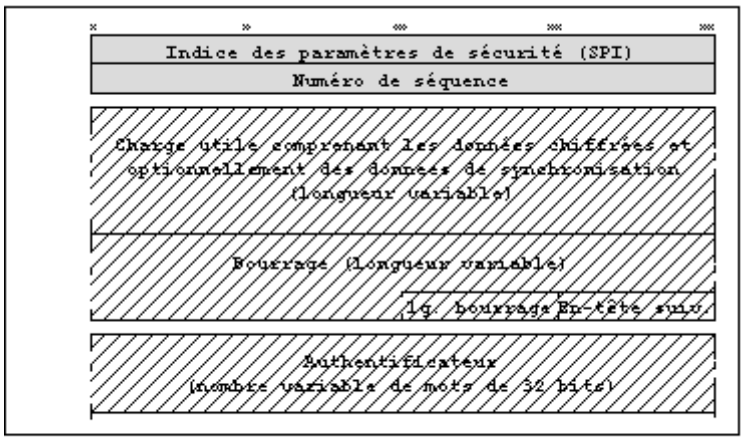


Figure 12-6 . Format de l'extension d'ESP

- Le champ indice de paramètres de sécurité (SPI) combiné à l'adresse du (des) destinataire(s), identifie l'association de sécurité utilisée pour construire cette extension.
- Le numéro de séquence permet de détecter les rejeux de paquet IP.
- Le champ charge utile peut contenir soit un paquet IP complet (en-tête IP + extensions + données de niveau transport), soit l'extension destination suivie de données de niveau transport, soit des données de niveau transport uniquement.

  Ce champ peut également contenir des données de synchronisation selon les algorithmes de chiffrement utilisés.

- Le champ bourrage permet d'ajouter des bits de bourrage pour aligner les données à chiffrer sur un nombre d'octets dépendant de l'algorithme de chiffrement utilisé. Le champ bourrage est optionnel. Sa présence est précisée par l'association de sécurité utilisée.
- Le champ longueur du bourrage indique la longueur en octets du champ bourrage.
- Le champ en-tête suivant identifie le type de données contenues dans le premier champ du champ charge utile.
- Le champ authentificateur est optionnel et sa présence dépend de l'association de sécurité utilisée (SPI + adresse(s) de destination + ESP). Il est calculé à l'aide de l'algorithme et de la clé correspondant à l'association de sécurité.

Notez que le champ en-tête ESP inclut l'indice des paramètres de sécurité et le numéro de séquence, tandis que le champ queue ESP comprend les champs bourrage, longueur du bourrage et en-tête suivant.

La détection de rejeu est optionnelle car lors de la génération d'une extension ESP, il est nécessaire de préciser un numéro de séquence adéquat dans le champ numéro de séquence tandis qu'à l'extraction de l'extension ESP, la vérification du numéro de séquence n'est pas obligatoire.

Le RFC 2406 précise les algorithmes de chiffrement que doivent obligatoirement implémenter les équipements IPsec pour rendre le service de confidentialité. Si dans le RFC 2406, seul l'algorithme DES dans le mode d'utilisation CBC (*Cipher Block Chaining*) est mentionné obligatoire, au fil du temps, l'IETF a

fait évoluer la liste en ajoutant en 1999 le triple DES et plus récemment l'AES (*Advanced Encryption Stantard*). Quant à la génération de l'authentificateur, les mêmes mécanismes que pour l'extension AH sont préconisés par défaut, à savoir HMAC- MD5, HMAC-SHA-1.

Le chiffrement des données dans l'extension ESP n'est pas obligatoire. Dans ce cas, seuls les services d'intégrité/authentification sont rendus, mais ils ne portent que sur l'extension ESP, contrairement à l'extension AH qui porte sur tout le paquet IP excepté certains champs "modifiables" par le réseau. Dans certains cas, suivant le niveau de protection en intégrité souhaité, il peut être nécessaire d'utiliser les deux extensions AH et ESP dans le même paquet, l'extension ESP ne rendant alors que le service de confidentialité.

L'extension ESP souffre tout comme l'extension AH d'incompatibilités avec le mécanisme de traduction d'adresse, et ce, bien que l'authentification réalisée ne porte que sur le contenu de l'en-tête ESP ; cependant contrairement à AH, il existe une solution palliative présentée ci-dessous. Le problème d'incompatibilité de ESP en mode transport avec la traduction d'adresses seules est lié aux protocoles TCP et UDP qui définissent un champ de contrôle dont le calcul fait intervenir les adresses IP ; ainsi la modification d'une adresse suppose la modification de ce contrôle, ce qui n'est pas réalisable avec ESP en mode transport ; en effet, soit le chiffrement est activé dans ESP et le champ de contrôle est inaccessible car chiffré ; soit les services d'authentification/intégrité sont activés, et la modification du champ de contrôle conduira à la détection d'un problème d'intégrité du paquet et au rejet du paquet par l'équipement IPsec récepteur.

Le fait de traduire en plus des adresses les numéros de port ajoute encore davantage d'incompatibilités. Pour ESP, c'est l'encapsulation elle-même qui pose problème car elle rend les numéros de port soit inaccessibles (chiffrement activé), soit interdits de modification (car protégés en intégrité). Enfin, noter que pour ESP en mode tunnel, les numéros de port sont inexistants.

Une solution est aujourd'hui définie pour pallier à cette incompatibilité. Elle consiste à n'utiliser que le protocole ESP (mode transport ou tunnel) et à réaliser une encapsulation UDP supplémentaire. Plus précisément, un tunnel UDP doit être établi entre les deux équipements IPsec ; les données encapsulées dans ce tunnel sont protégées par le protocole ESP. Ainsi, la traduction des adresses/numéros de port donne lieu à la modification du champ de contrôle de UDP sans porter atteinte à l'intégrité de l'en-tête ESP et la traduction de numéros de port est faite sur les numéros de port UDP qui sont accessibles ici. Afin de détecter la traversée d'un équipement de traduction d'adresse et n'activer l'encapsulation UDP que lorsque nécessaire, les passerelles IPsec peuvent faire appel au mécanisme de NAT-traversal [RFC 3947] [RFC 3948] lors de l'établissement dynamique d'une association de sécurité.

C) Évolutions prévues

La prochaine version de l'extension ESP intègre les mêmes évolutions que AH concernant l'identification des associations de sécurité et les numéros de séquence. Tout comme pour AH, il est prévu que la liste des algorithmes obligatoires soit précisée dans un document indépendant et tenu à jour.

Il est également prévu de permettre l'usage d'un plus grand nombre d'algorithmes, comme les algorithmes dits "combinés" qui permettent simultanément de chiffrer et de protéger en intégrité. Ces algorithmes combinés nécessitent un aménagement du format de ESP. En effet, certains d'entre eux n'assurent l'intégrité que sur les données chiffrées ; d'autres peuvent étendre la protection en intégrité sur des champs extérieurs. Compte tenu que l'indice de paramètres de sécurité et le numéro de séquence ne sont pas chiffrés mais nécessitent d'être protégés en intégrité, il peut être nécessaire de les faire apparaître deux fois dans le paquet : dans l'en-tête ESP et dans la charge utile. De façon à rendre l'extension ESP ouverte à un grand nombre d'algorithmes, le champ authentificateur n'est plus tenu de tenir sur un nombre entier de 32 mots.

Afin de se prémunir des analyses statistiques portant sur les flots de données chiffrés, la prochaine version de la RFC 2406 prévoit d'assurer le service de confidentialité du flux. La technique utilisée consiste à introduire des données de bourrage (sans signification) dans les paquets IP. À cette fin, un nouveau champ optionnel - *Traffic Flow Confidentiality* - est défini et se trouve positionner dans la charge utile.

Enfin, trois configurations de ESP sont distinguées suivant que sont rendus le service d'intégrité seul, le service de confidentialité seul ou simultanément les services d'intégrité et de confidentialité. Cette distinction permet d'optimiser le traitement des paquets, et de permettre l'usage d'algorithmes combinés. Noter que le service de confidentialité est optionnel et peut donc ne pas être implémenté dans ESP.

7) Gestion des associations et des clés

Les extensions AH et ESP ont recours à des mécanismes cryptographiques pour protéger les échanges et ont donc besoin de clés de chiffrement. L'un des problèmes fondamentaux d'utilisation de la cryptographie est la gestion de ces clés. Dans le cadre d'IPsec, les clés de chiffrement sont gérées, au même titre que les autres paramètres relatifs à la protection des échanges, par le biais des associations de sécurité.

Il existe deux approches de gestion des associations de sécurité. La plus simple est la gestion manuelle, qui consiste à laisser les individus configurer manuellement chaque équipement de sécurité avec les paramètres appropriés et notamment les clés. Si cette approche s'avère relativement pratique dans un environnement statique et de petite taille, elle ne convient plus pour un réseau de taille importante. De plus, cette méthode implique une définition totalement statique des associations de sécurité et un non-renouvellement des clés.

La seconde approche est la gestion automatique des associations de sécurité au moyen d'un protocole appelé ISAKMP (*Internet Security Association and Key Management Protocol*) définit dans le RFC 2408 qui fournit un cadre générique pour la négociation, la modification et la destruction des SA, ainsi que pour le

format de leurs attributs. Comme le montre la See Architecture d'ISAKMP et IKE, ISAKMP est indépendant du protocole d'échange de clés et du protocole pour lequel on souhaite négocier une association.

Ainsi, dans le cadre de la standardisation IPsec, ISAKMP est associé en partie aux protocoles d'échanges de clés SKEME et Oakley pour donner un protocole final du nom de *Internet Key Exchange* ou IKE [RFC 2409]. Ce protocole couvre pour le moment uniquement les situations d'unicast, mais l'IETF travaille à définir une distribution de clés multicast.

Plus précisément, pour gérer les associations de sécurité et les clés de chiffrement dans un environnement IPsec, le protocole IKE fait appel aux éléments suivants :

- un protocole de gestion des associations de sécurité comme ISAKMP (Internet Security Association and Key Management Protocol) [RFC 2408], qui définit des formats de paquets permettant de créer, modifier et détruire des associations de sécurité. Ce protocole sert également de support pour l'échange de clés préconisé par les protocoles de gestion de clés. Il assure aussi l'authentification des partenaires d'une communication ;
- un protocole d'échange de clés de session basé sur SKEME et Oakley qui repose sur la génération de secrets partagés Diffie-Hellman. Plus exactement, IKE utilise certains des modes définis par Oakley et emprunte à SKEME son utilisation du chiffrement à clé publique pour l'authentification et sa méthode de changement de clef rapide par échanges d'aléas ;
- un domaine d'interprétation ou DOI (*Domain of Interpretation*) [RFC 2407] qui définit tous les paramètres propres à l'environnement IPsec, à savoir les protocoles d'échanges de clés, les paramètres d'associations de sécurité à négocier,... ;
- les clés utiles lors de l'authentification mutuelle des équipements IPsec qui intervient en préalable à toute négociation d'association de sécurité. Ces clés peuvent être des clés partagées (*pre-shared key*) préconfigurées par l'administrateur, ou bien des clés privées/publiques personnelles à chaque équipement IPsec et préchargées dans les équipements, ou bien encore un certificat électronique géré par une infrastructure à clés publiques (PKI : *Public Key Infrastructure*) telle que décrite See PKI : principe et lacunes actuelles.

Deux versions de IKE vont bientôt coexister et une description de ces deux versions est proposée ci-après.

A) Architecture

ISAKMP offre un cadre générique de gestion des associations de sécurité en ce sens qu'il s'applique à n'importe quel protocole implémentant de la sécurité : IPsec, TLS (*Transport Layer Security*), SSL (*Secure Socket Layer*). Ceci est possible car ISAKMP est prévu pour fonctionner au-dessus d'IP si bien que les informations relatives à la gestion des associations de sécurité et des clés sont transportées au moyen de paquets ISAKMP indépendamment du protocole en faveur duquel la négociation a lieu. ISAKMP peut être placé directement au-dessus d'IP ou au-dessus de tout protocole de niveau transport. Il s'est notamment vu attribuer le port 500 sur UDP par l'IANA.

Les messages d'ISAKMP sont constitués d'un en-tête suivi d'un nombre variable de blocs. Ces blocs (payloads) sont les briques de base d'ISAKMP et incluent des informations relatives :

- paramètres de sécurité à négocier (bloc Security Association ou SA, Proposal ou P, Transform ou T),

- aux clés de session à convenir (Key Exchange ou KE),
- aux identités des entités (ID), aux certificats (CERT, Certificate Request ou CERTREQ),
- à l'authentification (HASH, SIG, NONCE où NONCE contient un nombre généré aléatoirement),
- aux messages d'erreurs (notification ou N),
- aux associations de sécurité à supprimer (Delete) et
- au constructeur d'équipement/logiciel de sécurité (Vendor ID).

ISAKMP définit un cadre pour négocier les associations de sécurité, mais n'impose rien quant aux paramètres qui les composent. Un document appelé "domaine d'interprétation" (DOI : *Domain of Interpretation*) définit la syntaxe des paramètres, les types d'échanges et les conventions de nommage relatifs à l'emploi d'ISAKMP dans un cadre particulier. Un identifiant de DOI est par conséquent utilisé pour interpréter et synthétiser la charge utile des messages ISAKMP dans le contexte d'un DOI donné (cf. See Présentation symbolique du domaine d'interprétation IPsec). Le DOI IPsec documenté dans le RFC 2407 est identifié par le numéro 1.

ISAKMP offre l'avantage d'être indépendant de la méthode de génération des clés et des algorithmes de chiffrement et d'authentification utilisés. Il est donc indépendant de tout protocole d'échange de clé, ce qui permet de séparer clairement les détails de la gestion des associations de sécurité des détails de l'échange de clé. Dans le contexte IPsec, les protocoles d'échange de clés utilisés sont SKEME/Oakley.

A la fin de la négociation, comme le montre la figure Architecture d'ISAKMP et IKE, ISAKMP communique le résultat au module intéressé par le biais d'une API (Application Programming Interface). Même si aujourd'hui, ISAKMP est exclusivement utilisé pour configurer le module IPsec - ESP/AH, il pourrait tout à fait servir à la configuration de modules SSL/TLS ou autres.

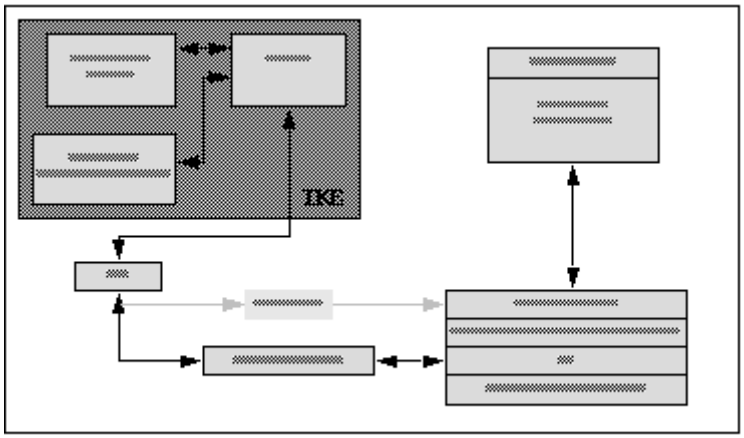


Figure 12-7 . Architecture d'ISAKMP et IKE

B) Echanges ISAKMP/IKEv1

Enfin, ISAKMP comporte deux phases qui permettent une séparation nette de la négociation des associations de sécurité pour IPsec et de la protection du trafic propre à ISAKMP :

- Durant la phase 1, un ensemble d'attributs relatifs à la sécurité est négocié, les identités des entités sont authentifiées et des clés sont générées. Ces éléments constituent une première association de sécurité, dite SA ISAKMP. Contrairement aux associations de sécurité IPsec, le SA ISAKMP est bidirectionnel. Il servira à protéger l'ensemble des échanges ISAKMP futurs.

- La phase 2 permet de négocier les paramètres de sécurité relatifs à une association de sécurité à établir pour le compte d'un protocole de sécurité donné (par exemple AH ou ESP). Les échanges de cette phase sont protégés en confidentialité et intégrité/authentification grâce au SA ISAKMP. Ce dernier peut bien sûr être utilisé pour négocier plusieurs associations de sécurité destinées à d'autres protocoles de sécurité.

ISAKMP définit des types d'échanges (exchange types). Un type d'échange est une spécification de l'ensemble des messages constituant un type d'échange donné (nombre, contenu). Chaque type d'échange est conçu pour fournir un certain nombre de services de sécurité spécifiques, comme l'anonymat, la propriété de *Perfect Forward Secrecy* (PFS) et l'authentification mutuelle.

IKE comprend quatre modes :

- le mode principal (*Main Mode*),
- le mode agressif (*Aggressive Mode*),
- le mode rapide (*Quick Mode*) et
- le mode nouveau groupe (*New Group Mode*).

Comme le montre la figure Phases d'IKEv1, *Main Mode* ou *Aggressive Mode* sont utilisés pour la phase 1 de la négociation ISAKMP ; *Quick Mode* est un échange de la phase 2 avec deux variantes suivant que la propriété de PFS est rendue ou non.

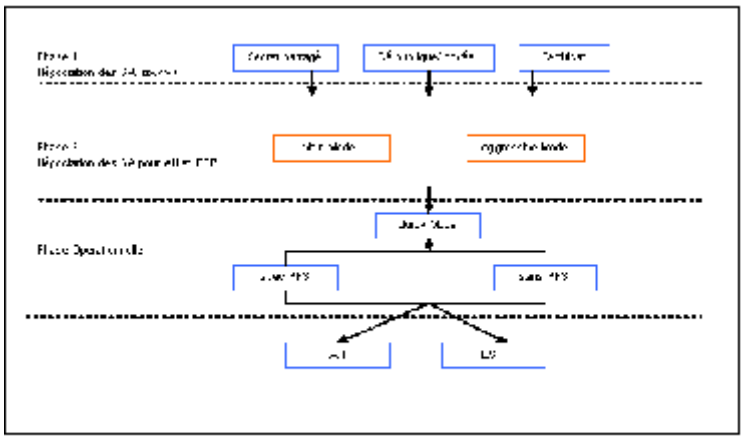


Figure 12-9 . Phases d'IKEv1

New Group Mode est un peu à part : ce n'est ni un échange de la phase 1, ni un échange de la phase 2, mais il ne peut avoir lieu qu'après l'établissement d'une SA ISAKMP. Il sert à convenir d'un nouveau groupe pour de futurs échanges Diffie-Hellman.

b) Phase 1 de IKEv1

Durant la phase 1, IKE négocie les attributs -- algorithme de chiffrement, fonction de hachage, méthode d'authentification et groupe mathématique pour les calculs relatifs à Diffie-Hellman -- et produit trois clés qui serviront pour chiffrer, authentifier et dériver d'autres clés. Ces attributs et ces clés permettent de protéger les messages de la phase 2 en authentification et confidentialité. L'authentification des messages est assurée par l'ajout d'un bloc HASH après l'en-tête ISAKMP, et la confidentialité est assurée par le chiffrement de l'ensemble des blocs du message. L'une des trois clés sert à produite deux autres clés utiles à la protection des échanges IPsec en authentification/intégrité et confidentialité.

Main Mode se compose de six messages comme le montre la figure Echanges - Phase 1 d'IKEv1 (*main mode*) :

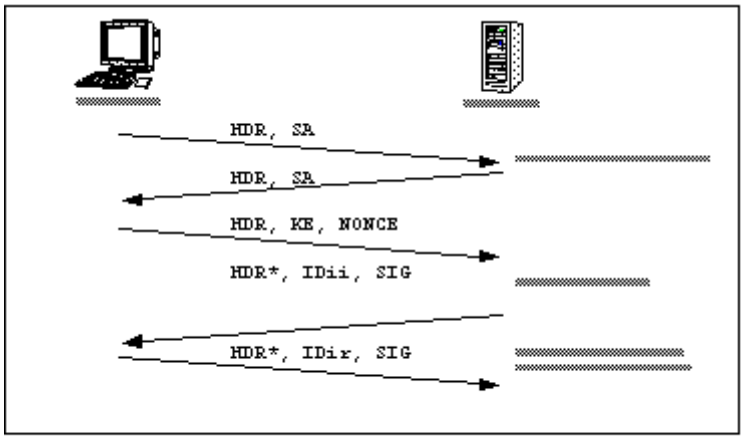


Figure 12-10. Echanges - Phase 1 d'IKEv1 (main mode)

- les deux premiers messages servent à négocier l'association de sécurité ISAKMP, ou en d'autres termes les paramètres propres à IKE : algorithme de chiffrement, fonction de hachage, méthode d'authentification des tiers et groupe pour Diffie- Hellman. Les blocs ISAKMP transportés sont de type *Security Association*, *Proposal* et *Transform*. L'initiateur propose plusieurs combinaisons d'algorithmes, mécanismes, et méthodes, et le partenaire IKE en choisit un. Pour information, la notation HDR sur les figures Echanges - Phase 1 d'IKEv1 (*main mode*) et Echanges - Phase 2 d'IKEv1 (*quick mode*) signifie "header" pour l'en-tête des messages IKE comprenant entre autres l'index SPI, le numéro de version.
- Pour s'authentifier, les équipements IPsec utilisent soit un secret partagé, soit des clés publique/privée, soit un certificat.
- les deux suivants permettent l'établissement d'un secret partagé via l'utilisation d'un échange de valeurs publiques Diffie-Hellman.
- Le secret partagé sert à dériver des clés de session, deux d'entre elles étant utilisées pour protéger la suite des échanges avec les algorithmes de chiffrement et de hachage négociés précédemment. Les blocs ISAKMP sont de type *Key Exchange*, *Nonce* et optionnellement *Certificate Request*.
- les deux derniers messages permettent aux tiers d'échanger leurs identités et servent à l'authentification de l'ensemble des données échangées (et donc des tiers en présence). Les blocs ISAKMP sont de type ID, SIG et optionnellement CERT. Le bloc SIG peut être formé selon trois méthodes d'authentification basées soit sur des clés partagées, soit des certificats X.509, soit un chiffrement avec des clés publiques. Noter que les HDR* signifient que la charge utile du message est chiffrée.

Main Mode fournit la propriété de *Perfect Forward Secrecy* (grâce à Diffie-Hellman) et assure l'anonymat des entités en présence (grâce au chiffrement des deux derniers messages).

Aggressive Mode ne présente pas ces avantages mais est plus rapide car il combine les échanges ci-dessus de façon à ramener le nombre total de messages à trois.

c) Phase 2 de IKEv1

Une fois une association de sécurité ISAKMP mise en place grâce à des échanges de phase 1, Quick Mode est utilisé pour négocier des associations de sécurité IPsec (cf. figure Echanges - Phase 2 d'IKEv1 (*quick mode*)). Chaque négociation aboutit à deux SA symétriques, une dans chaque sens de la communication. Plus précisément, les échanges composant ce mode ont les rôles suivants :

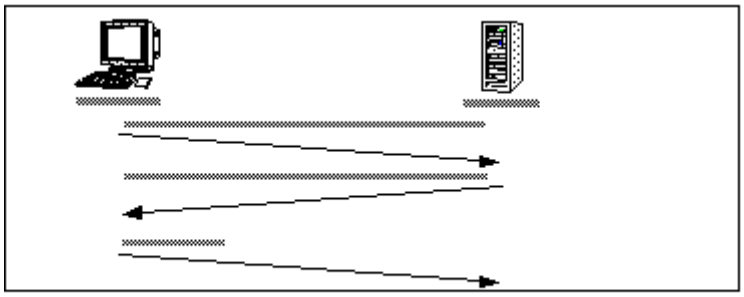


Figure 12-11 . Echanges - Phase 2 d'IKEv1 (quick mode)

- Négocier un ensemble de paramètres IPsec (paquet de SA).
- Échanger des nombres aléatoires, utilisés pour générer une nouvelle clé qui dérive de celle du SA ISAKMP. De façon optionnelle, il est possible d'avoir recours à un nouvel échange Diffie-Hellman, afin d'accéder à la propriété de *Perfect Forward Secrecy*, qui n'est pas fournie si on se contente de générer une nouvelle clé à partir de l'ancienne et des aléas.
- Optionnellement, identifier le trafic que ce paquet de SA protégera, au moyen de sélecteurs (blocs optionnels `IDi` et `IDr` ; en leur absence, les adresses IP des interlocuteurs sont utilisées).

Noter que les négociations de phase 1 visant à se réauthentifier entre équipements et à s'échanger de nouvelles clés secrètes sont rarement faites, tandis que celles de phase 2 sont beaucoup plus courantes. Pour exemple, une phase 1 peut être exécutée une fois par jour, voire une fois par semaine, tandis qu'une phase 2 est exécutée une fois toutes les quelques minutes.

d) Interactions entre IKE et IPsec

Les interactions entre IKE et IPsec sont représentées par le schéma de la figure Interactions de IKEv1 avec IPsec. Deux types de traitement sont envisagés :

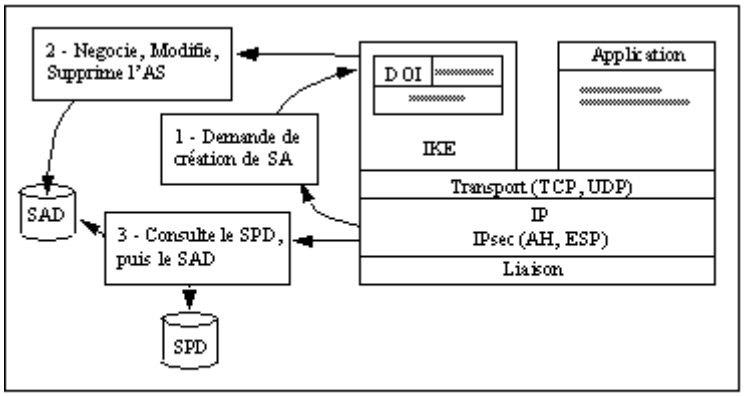


Figure 12-12 . Interactions de IKEv1 avec IPsec

- Trafic sortant

  Lorsque la couche IPsec reçoit des données à envoyer, elle consulte ses bases de données SPD et SAD pour savoir comment traiter ces données et si l'association de sécurité existe déjà. Si l'association de sécurité existe, elle est utilisée pour traiter le trafic en question. Dans le cas contraire, IPsec fait appel à IKE pour établir une nouvelle association de sécurité avec les caractéristiques requises.

- Trafic entrant

  Lorsque la couche IPsec reçoit un paquet en provenance du réseau, elle examine les extensions du paquet. Elle déduit si le paquet est protégé par IPsec et si oui, les indices du ou des associations de sécurité à appliquer. Elle consulte alors le SAD pour connaître les paramètres à utiliser pour la vérification et/ou le déchiffrement du paquet. Une fois le paquet authentifié et déchiffré, le SPD est consulté pour vérifier que le paquet en question répondait bien aux critères du SA qui le protégeait. Dans le cas où le paquet reçu est un paquet IP classique, le SPD permet également de savoir s'il a néanmoins le droit de passer. En particulier, les paquets IKE doivent avoir le droit de passer afin de pouvoir être traités par IKE, qui sera seul juge de leur validité.

8) IKE Version 2

Pour faire face à la complexité de configuration de IKEv1 et à son manque d'efficacité, une seconde version de IKE [Kau-id] a été définie dans le but d'être facilement utilisable sur des VPNs mettant en jeu des fournisseurs d'équipements différents. Pour faciliter sa lecture, le protocole IKEv2 se trouve décrit dans un document unique qui regroupe les informations précédemment réparties dans les RFC 2407 (domaine

d'interprétation), RFC 2408 (ISAKMP), RFC 2409 (IKEv1), RFC 3947 RFC 3948 (NAT-traversal). Ce protocole n'a plus le caractère générique de IKEv1 et supprime donc la notion de domaine d'interprétation.

IKEv2 a été défini avec comme objectifs de simplifier le protocole IKEv1, d'enlever les conditions inutiles d'ISAKMP et IKEv1 et d'incorporer de nouvelles fonctionnalités dans le protocole IPsec. L'objectif général est de clarifier au maximum l'utilisation du protocole et de rendre son implémentation plus facile. A ce titre, IKEv2 est moins flexible que IKEv1 ; de plus, IKEv2 introduit une refonte importante, puisqu'il propose de passer à une négociation des phases 1 et 2 en quatre messages. De ce fait, le nouveau protocole apparaît tellement différent de l'actuel qu'il ne sera pas compatible, d'où le changement de numéro de version majeure.

IKEv2 est semblable à IKEv1 dans l'authentification mutuelle des entités et l'établissement des associations de sécurité ISAKMP et de clés partagées. Ainsi :

- les deux premiers messages de IKEv2 présentés à la figure Echanges de IKEv2 (*main mode*) sont équivalents aux quatre premiers messages de IKEv1 en Main Mode. Les SA ISAKMP unidirectionnels apparaissent sous la notation SAi1 et SAr1.

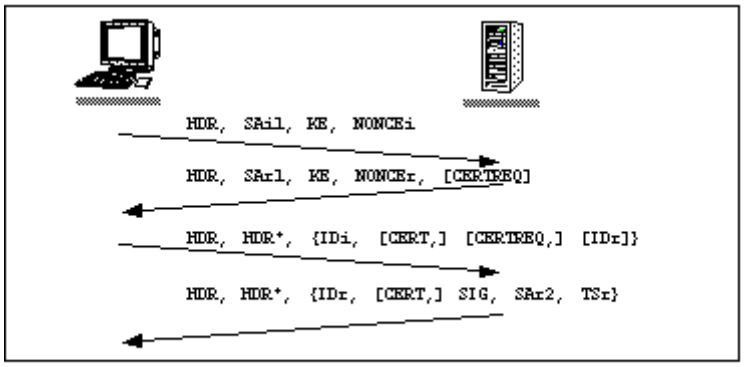


Figure 12-13. Echanges de IKEv2 (main mode)

- Les deux derniers messages sont équivalents aux deux derniers messages de phase 1 de IKEv1 en *Main Mode* et aux deux messages de phase 2 de IKEv1 en *Quick Mode*. En particulier, les deux derniers messages permettent aux équipements IPsec de prendre connaissance de leurs identifiants respectifs (ce qui est particulièrement utile en cas d'entités multiples hébergées sur une même machine), de s'authentifier grâce au bloc AUTH, de négocier des associations de sécurité IPsec unidirectionnelles (`SAi2` et `SAr2`) qui sont appelées dans IKEv2 CHILD_SA et d'identifier le trafic protégé par ces SA IPsec en précisant les sélecteurs choisis dans les blocs `TSi` et `TSr` (`TS` pour Traffic Selector).

Le bloc CERT consiste en une liste de certificats, avec le premier certificat de la liste utilisé pour générer le bloc AUTH. Le partenaire peut également forcer un équipement à retourner une liste de certificats en émettant un bloc CERTREQ contenant la liste des certificats préférés. Cette liste comprend en fait la liste des autorités de certification reconnues de confiance dont les identités sont hachées avant d'être placées dans le bloc CERTREQ.

9) Clés publiques : infrastructures et certificats

L'utilisation de la cryptographie à clé publique à grande échelle nécessite de pouvoir gérer des listes importantes de clés publiques, pour des entités souvent réparties dans un réseau. Pour cela, on a recours à des infrastructures à clés publiques (PKI : *Public Key Infrastructures*) qui sont des systèmes de gestion de clés publiques.

Ces PKI gèrent bien souvent les clés publiques sous forme de certificats, mais il est tout à fait possible de se passer des certificats pour assurer la gestion des clés publiques. Après une présentation des certificats et des listes de certificats révoqués (CRL : *Certificate Revocation List*), l'organisation et le fonctionnement général des PKIs sont décrits.

Contents

A) Certificats X.509 et CRL

Un certificat de clé publique permet d'associer une clé publique à une entité (serveur, PC, individu...) de façon sûre, c'est-à-dire certifiée par une autorité de certification (AC : Autorité de Certification) habilitée qui signe chacun des certificats qu'elle a émis. La grande majorité des certificats aujourd'hui utilisés se base sur la norme X.509V3 qui définit plusieurs champs incluant :

- Version : le numéro de version de la norme X.509 à laquelle se rapporte le certificat ;
- Numéro de série : le numéro unique correspondant à un certificat généré par une autorité de certification ;
- Emetteur : l'identifiant de l'autorité de certification émettrice du certificat ;
- Période de validité : les dates de début et de fin de validité du certificat ;
- Sujet : l'identifiant du propriétaire du certificat ;
- Clé publique : la clé publique associée au propriétaire du certificat ;
- Signature : la signature électronique de l'autorité de certification portant sur tout le contenu du certificat pour en assurer l'authenticité ;
- Extensions : de nombreuses extensions peuvent être ajoutées au certificat par l'autorité de certification comme l'usage qui doit être fait de la clé publique.

Avant même que le certificat n'arrive à expiration, de nombreuses raisons comme la compromission de la clé privée associée au certificat, l'impossibilité pour l'autorité de certification de continuer à enregistrer le

certificat, peuvent amener le certificat à être révoqué. La technique la plus répandue aujourd'hui consiste pour l'autorité de certification à générer périodiquement une liste CRL contenant les numéros de série des certificats révoqués, ainsi que la date de génération de la CRL et la date attendue pour la prochaine mise à jour de la CRL. L'authenticité de cette liste est assurée par l'AC par le biais de sa signature.

B) PKI : principe et lacunes actuelles

Une PKI (*Public Key Infrastructure*) ou IGC (Infrastructure de Gestion de Clés) permet de définir des entités fonctionnelles, chacune avec un rôle bien précis, le but étant d'offrir un environnement de confiance ainsi que des garanties quant à l'authenticité des certificats de clés publiques. La gestion des certificats est opérée au sein de la PKI par trois entités :

- Autorité de certification (AC) : l'organisme qui émet des certificats électroniques (par exemple CertPlus, Certinomis, Verisign) ;
- Autorité d'enregistrement : l'entité réalisant l'interface entre les entités demandeuses de certificats et l'autorité de certification chargée de vérifier l'identité du demandeur de façon plus ou moins forte (par exemple présentation d'une carte d'identité) ;
- Système de publication et de distribution de certificats et CRL : l'entité chargée par l'autorité de certification de publier et distribuer des certificats électroniques et des CRL, ce qui peut être réalisé classiquement par le biais d'un annuaire LDAP (Lightweight Directory Access Protocol) ou d'un serveur Web.

Une PKI est généralement organisée en une hiérarchie d'autorité de certifications, chacune de ces autorité étant responsable de la gestion d'un sous-ensemble de certificats et de CRL. Dans cette hiérarchie, une autorité de certification habilite une autorité de niveau inférieur à gérer les certificats de la PKI en lui générant un certificat pour sa propre clé publique ; elle lui prouve ainsi sa confiance. L'AC racine n'ayant aucune AC au dessus certifie elle-même sa propre clé publique ; le certificat de l'AC racine est appelé certificat auto-signé.

En fait, une PKI doit être capable de rendre les quatre fonctions suivantes :

- L'enrôlement qui consiste optionnellement à générer une clé publique/privée, à vérifier l'identité de l'entité demandeuse, et de façon obligatoire à distribuer des clés privées et certificat ;
- La publication de certificats de clé publique, c'est-à-dire à maintenir disponible les certificats pour des entités tiers demandeuses ;
- La validation de certificats qui doit permettre de renseigner n'importe quelle entité sur l'état de validité d'un certificat, et ce à tout moment ;
- La révocation de certificats qui consiste généralement à publier une CRL.

Si aujourd'hui la distribution et la publication de clés sont aisément résolues avec la distribution et la publication de clés/certificats par le biais d'un support comme une clé USB, ou une carte à puce, voire la publication de certificats via un annuaire LDAP par exemple, en revanche la révocation et la vérification de clés ne le sont pas ou imposent certaines limitations d'usage. En effet, de nombreuses PKIs existent actuellement et s'ignorent de sorte que pour vérifier la validité d'un certificat, il est nécessaire de connaître et d'avoir confiance dans l'AC émettrice. Pour la plupart des équipements IPsec, un certificat est

considéré valide uniquement s'il est émis par la même autorité de certification dont il dépend ; au mieux, il est possible d'étendre cette validité à des certificats émis par une autre autorité de certification à la condition de précharger le certificat dans l'équipement. Noter que pour toutes ces raisons, l'établissement d'un VPN (*Virtual Private Network*) sécurisé ne peut pas s'improviser. Certaines solutions tentant de rendre les VPNs dynamiques et appelées « chiffrement opportuniste » voient cependant le jour à l'IETF.

C) Outils et protocoles de mise en oeuvre des PKIs

Plusieurs outils logiciels opensource permettent de gérer des certificats de clé publique :

- L'enrôlement et la révocation de certificats sont assurés par le logiciel OpenCA développé par The OpenCA Labs. OpenCA se base sur Apache et permet donc aux administrateurs/utilisateurs de réaliser toute la gestion des certificats au travers d'un navigateur classique. Suivant le répertoire auquel on accède sur le serveur OpenCA, un utilisateur jouera le rôle de l'autorité de certification ou de l'autorité d'enregistrement.

  OpenCA bénéficie de plusieurs avantages vis-à-vis d'autres logiciels de gestion de certificats comme IDEALX. En effet, il est relativement simple d'installation et de plus il s'interface avec OpenLDAP ce qui permet, une fois les certificats générés, de les publier de façon automatique dans la base LDAP.

- La publication des certificats et CRL est réalisée par openLDAP développé par The OpenLDAP Project. Depuis 1999, avec les nouveaux attributs de LDAP définis dans le RFC 2587, il est possible de publier un certificat mais aussi une CRL dans une entrée de LDAP.

Quant à l'aspect validation, toute la difficulté est d'obliger les applications utilisatrices de certificats de vérifier la validité d'un certificat avant usage. Aujourd'hui, certaines applications comme Netscape 7 implémentent le protocole OCSP (*Online Certificate Status Protocol*) [RFC 2560] qui permet à une application d'interroger un serveur OCSP en ligne pour connaître la validité d'un certificat. Le serveur OCSP attaché à une autorité de certification se contente en fait de vérifier le statut d'un certificat stocké en local dans un de ces répertoires ; le certificat précise dans un champ l'adresse du serveur OCSP à contacter. Un autre protocole SCVP (*Simple Certificate Validation Protocol*) [MHF-id] est en cours de définition à l'IETF. Le serveur SCVP qui est local à l'application demandeuse est capable de vérifier la validité de n'importe quel certificat pour peu qu'il ait confiance dans la PKI concernée.

D) DNSSEC (Domain Name System Security)

DNSSEC [RFC 2535] est un standard définissant des extensions de sécurité pour le DNS. Il vise à assurer l'intégrité et l'authenticité des enregistrements DNS (`RR` : *Registration Record*) en introduisant des signatures électroniques (`RRSIG RR`) calculées par la zone sécurisée DNSSEC sur des enregistrements DNS. Plus exactement, chaque zone dispose de deux paires de clés publiques/privées appelées Key Signing Key (`KSK`) et Zone Signing Key (`ZSK`). La clé publique de KSK est connue de la zone parent et lui permet donc

d'authentifier sa zone fille, tandis que les clés ZSK permettent de signer les enregistrements gérés par la zone elle-même.

Il est tout à fait possible de superposer une PKI sur la hiérarchie DNS. En effet, chaque zone DNSSEC peut jouer le rôle d'une autorité de certification. Il lui suffit de certifier la clé publique de sa zone fille en signant le condensat de cette clé et en publiant ce condensat dans l'enregistrement `DS RR` (*Delegation Signer*) [RFC 3658]. La zone fille publie sa clé publique KSK dans l'enregistrement DNSKEY RR prévu à cet effet et signe cet enregistrement avec sa clé privée KSK. Ainsi, il suffit donc d'avoir confiance dans la zone parent pour récupérer en toute confiance la clé publique KSK de la zone fille. Il est alors possible d'avoir confiance dans la clé ZSK publiée dans un `DNSSKEY RR` et signée par la zone fille avec sa clé privée KSK (`RRSIG RR`). De proche en proche, on peut mettre en place une PKI, avec chaque zone définissant une autorité de certification intermédiaire et une autorité de certification racine correspondant à la zone DNSSEC de plus haut niveau dans la hiérarchie.

Pour publier des clés publiques associées à des noms DNS ou des adresses IP, DNSSEC prévoit l'usage de l'enregistrement `DNSKEY RR` protégé par les enregistrements `RRSIG RR`, ainsi que l'usage de l'enregistrement CERT RR, lui aussi protégé par RRSIG RR et qui peut inclure un certificat ou une CRL. La confiance dans la clé publique est à chaque fois garantie par la PKI DNS, mais dans le cas du CERT RR, la clé publique publiée est en plus gérée par une autorité de certification extérieure au DNS.

Il serait techniquement possible de gérer des CRLs et des clés publiques d'équipements IPsec par le biais de DNSSEC, mais cette utilisation de DNSSEC n'est pas recommandée car ces informations ont une durée de vie relativement courte (déplacement d'équipements, renouvellement fréquent des CRLs) et DNSSEC ne permet pas leur prise en compte immédiate du fait de la gestion des caches DNS.

E) Host Identity Protocol (HIP)

HIP [Mos-id] est une solution qui peut servir en remplacement des PKIs en prenant une clé publique comme identifiant d'un équipement IPsec. Plus généralement, HIP repose sur le principe de distinguer l'identité d'une machine (son identifiant) et le moyen de l'atteindre (son localisateur). Aujourd'hui, l'identifiant et le localisateur sont confondus dans l'adresse IP, mais HIP suggère de garder pour localisateur l'adresse IP et de considérer un nouvel identifiant HI (Host Identifier).

Il est recommandé de prendre pour HI d'un équipement IPsec la ou l'une des clés publiques associées à l'équipement. Ainsi, son exploitation sera immédiate lors de la phase d'authentification lors des échanges IKE. Bien entendu, il est nécessaire de publier le lien entre le localisateur et l'identifiant HI, en particulier, pour gérer la mobilité des équipements IPsec. Une solution consiste à associer au nom canonique d'un équipement son adresse IP ainsi que son HI. Noter que le HI étant de longueur trop importante, les équipements IPsec vont s'échanger leurs identifiants sous la forme du condensat du HI ; ce condensat qui fournit une valeur hachée sur 128 bits est appelé Host Identity Tag (HIT).

10) Architectures classiques de mise en oeuvre des IPsec

Aujourd'hui, les protocoles IPsec sont devenus d'un usage très courant sur Internet. Ils répondent au besoin des entreprises de protéger leurs communications passées au travers d'un réseau public, voire d'un réseau Wi-Fi (*Wireless Fidelity*) privé. Pour illustrer les différents usages des IPsec, plusieurs architectures sont décrites ci-dessous. En particulier, seront étudiés le cas où deux réseaux privés d'entreprises sont interconnectés par un tunnel protégé par IPsec, le cas du nomade qui se connecte à distance sur son réseau privé d'entreprise de façon protégée et enfin, le cas d'une entreprise qui souhaite protéger son accès Wi-Fi.

A) Interconnexion de réseaux privés

Le premier débouché commercial des IPsec (en 1999-2002) dans un environnement IPv4 fut sans conteste l'interconnexion de réseaux privés distants au travers d'un réseau public. D'une part, les entreprises voyaient là un moyen d'abaisser leurs coûts en communications de façon importante en remplaçant leurs lignes spécialisées par la configuration de réseaux privés virtuels (VPN : *Virtual Private Network*). D'autre part, la première génération de passerelles IPsec souffrait de limitations et se prêtait donc bien à ce type d'architecture où la configuration des passerelles est relativement statique et où une configuration manuelle par l'administrateur est possible.

Comme le montre la figure Architecture VPN : Interconnexion de réseaux locaux, cette architecture consiste à placer une passerelle de sécurité à la frontière entre les sites et le réseau public et à protéger les communications sur le segment de réseau séparant les deux passerelles. En fait, un tunnel IP protégé par IPsec est configuré sur les deux passerelles. La passerelle du côté du terminal émetteur est chargée d'insérer une extension AH ou ESP dans le paquet ; la passerelle réceptrice est chargée de vérifier la validité du paquet, de décapsuler le paquet et de transmettre le paquet d'origine au terminal destinataire en local. Dans cette architecture, seul le mode tunnel des protocoles IPsec est exploitable entre passerelles.

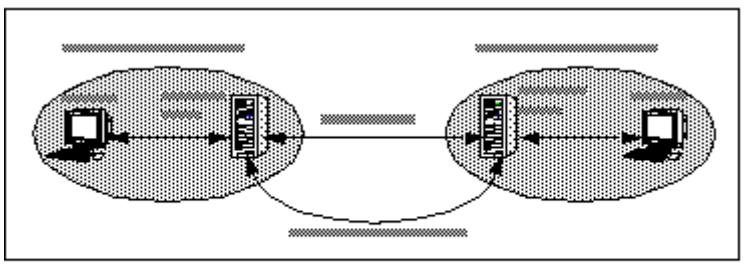


Figure 12-14 . Architecture VPN : Interconnexion de réseaux locaux

Noter que l'en-tête extérieur des paquets acheminés sur le réseau public fait intervenir les adresses publiques des passerelles (extrémités du tunnel), ce qui permet leur routage sur le réseau public, tandis

que l'en-tête des paquets IP encapsulés fait intervenir les adresses IP des réseaux privés qui sont généralement des adresses privées. Ainsi, deux terminaux même distants sont capables de s'adresser des paquets IP à l'aide de leurs adresses privées. Cette caractéristique leur permet donc de se comporter exactement comme s'ils étaient connectés sur le même réseau local. Le fait d'interconnecter de cette façon des réseaux privés distants est bien connu sous le nom de réseau privé virtuel ou VPN. Noter qu'un VPN n'est rien d'autre qu'un tunnel et n'est pas nécessairement protégé (par exemple par IPsec).

Actuellement ce type d'architecture nécessite l'intervention des administrateurs au moins pour permettre aux équipements de se connaître a priori et de pouvoir s'authentifier mutuellement lors de la phase I de IKE. Cela nécessite donc de charger un certificat ou de configurer un secret partagé. Dans un avenir plus ou moins proche, les passerelles IPsec seront moins contraignantes et permettront à des réseaux privés distants ne se connaissant pas de monter un tunnel dynamiquement. Cela améliorera grandement l'usage des IPsec puisqu'il sera alors possible de configurer un VPN pour une heure, une journée... le temps de la collaboration.

B) Nomadisme

L'architecture associée au nomadisme (cf. figure Architecture VPN : nomadisme) répond au besoin des utilisateurs qui partent en déplacement et souhaitent se connecter sur le réseau privé de leur entreprise pour accéder à leur boîte à lettre, des bases données, ou certaines applications [LAU03]. La plupart du temps, l'équipement nomade dont ils disposent leur est fourni par leur entreprise.

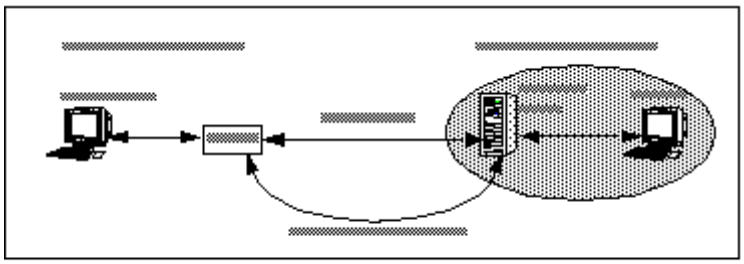


Figure 12-15 . Architecture VPN : nomadisme

L'usage d'IPsec dans un contexte du nomadisme s'avère beaucoup plus complexe que l'interconnexion de réseaux distants. En effet, comme un nomade peut se connecter depuis n'importe où, il dispose donc d'une adresse IP (IPv4 ou IPv6) temporaire qui est propre au réseau d'accès sur lequel il se trouve connecté ; il n'est donc plus possible, contrairement aux passerelles IPsec dans le cas de l'interconnexion de réseaux distants, d'authentifier un équipement en fonction de son adresse IP ; l'identifiant considéré dans le cas du nomadisme correspond à une chaîne de caractères de type `nom_machine@nom_entreprise` où `nom_machine` correspond au nom de machine et `nom_entreprise` au nom réseau de domaine de son réseau privé. De plus, en cas de vol d'un équipement nomade, il faut empêcher que le voleur ne puisse se connecter sur le réseau de l'entreprise et télécharger des informations confidentielles. En fait, l'objectif principal pour les entreprises est de n'autoriser l'accès à son réseau que depuis l'équipement nomade fourni à un employé et qu'à ce seul employé ; il faut donc pouvoir contrôler à la fois l'équipement et la personne.

C'est pour cela que les entreprises préconisent la double authentification, celle de l'équipement nomade et celle de l'utilisateur. La seule solution qui permette aujourd'hui de réaliser cette double authentification est la combinaison du protocole L2TP (*Layer Two Tunneling Protocol*) avec IPsec issue d'une collaboration entre Microsoft et Cisco [RFC 3193]. Les protocoles IPsec permettent en fait de protéger le tunnel L2TP, qui lui n'assure aucune protection des échanges en confidentialité, intégrité...

Plus précisément, le nomade doit établir une connexion avec la passerelle IPsec. Le mode de protection IPsec utilisé est ici le mode transport puisque les extrémités de la connexion sont le nomade et la passerelle. Lors de l'établissement de cette connexion, c'est le nomade qui s'authentifie auprès de la passerelle ; tous deux négocient une association de sécurité IPsec par le biais du protocole IKE. Ensuite, ils montent un tunnel L2TP ; noter que tous les échanges entre le nomade et le réseau privé sont protégés par IPsec sur le réseau public ; lors de l'établissement de ce tunnel, c'est cette fois-ci l'utilisateur qui est amené à s'authentifier auprès de la passerelle ; la passerelle envoie alors des paramètres de configuration au nomade ; en particulier, le nomade peut se voir affecter une adresse privée. Cette adresse privée est utile car il permet aux équipements du réseau privé de se comporter à l'égard du nomade exactement de la même façon que si le nomade était connecté dans l'enceinte du réseau privé.

a) Usage des VPNs dans le but de protéger des accès Wi-Fi

Cet usage des VPNs est apparu en attendant la commercialisation de produits Wi-Fi respectant la norme IEEE 802.11i. En effet, les réseaux Wi-Fi peuvent facilement faire l'objet d'écoutes du fait du mode de diffusion utilisé. Il est donc primordial d'assurer la confidentialité des communications au moins entre le mobile et le réseau d'accès.

Les VPNs peuvent être utilisés dans cette optique. La passerelle IPsec se trouve à ce moment là juste derrière le point d'accès Wi-Fi (cf. figure Architecture VPN : protection de l'accès Wi-Fi). D'une part, elle permet de rendre confidentiels les échanges sur le réseau sans-fils ; pour cela elle met en œuvre un tunnel ESP (avec chiffrement des échanges activé) avec le mobile IPsec. D'autre part, si l'accès au réseau filaire est restreint, elle permet d'authentifier les mobiles et ainsi de filtrer les demandes d'accès au réseau.

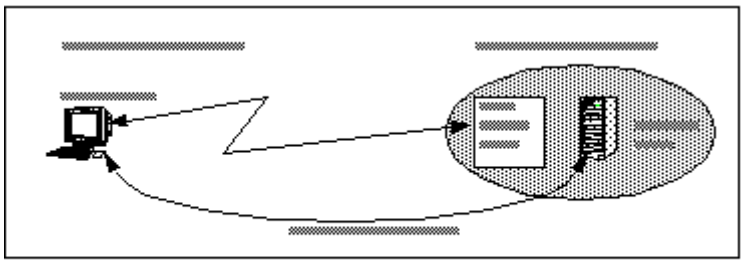


Figure 12-16 . Architecture VPN : protection de l'accès Wi-Fi

11) Mise en place d'une paire de SA IPsec

Deux machines A et B utilisent le système d'exploitation FreeBSD et on souhaite configurer une paire de SA IPsec entre ces deux machines. Afin d'utiliser IPsec, il est nécessaire que l'option IPSEC soit compilée dans le noyau41. Dans un premier temps une paire de SA IPsec sera mise en place en utilisant le protocole ESP. Pour cela, il faut configurer d'une part la base de données de politique de sécurité IPsec (SPD) et d'autre part la base de données des associations de sécurité (SAD).

Contents

A) Configuration de la SPD sur A et B

Imaginons que la politique de sécurité consiste à protéger tous les échanges entre `A` et `B` par le protocole ESP en mode transport.

Sur A, la SPD peut alors être configurée à l'aide du script présent à le Script de configuration de la SPD sur A. La commande `setkey` permet de manipuler les bases de données SPD et SAD (cf. `man setkey`). En particulier, l'option « `-FP` » permet de nettoyer la SPD et l'option « `-F` » de nettoyer la SAD.

L'une des opérations de `setkey` est `spdadd`. Cette opération permet d'ajouter une politique de sécurité dans la SPD. Elle prend les paramètres suivants :

- « `any` » signifie que n'importe quel trafic doit être protégé par ESP
- « `-P out` » indique que le trafic sortant est pris en considération
- « `-P in` » indique que le trafic entrant est pris en considéréation
- "`require`" indique que si la politique ne peut pas être appliquée (pas de SA correspondante), alors on ne doit ni envoyer ni accepter des paquets IP de `B`.

Dans le script, la première règle indique que tout le trafic (`any`) ayant pour source A et pour destination `B` (`$A $B`), qui sort de `A` (`-P out`), doit être traité par IPsec (`ipsec`). Ce trafic doit être protégé par ESP en mode transport.

Script de configuration de la SPD sur A

```
#!/bin/csh
set A = 2001:0660:3203:1020:210:b5ff:feab:4cc4
set B = 2001:0660:3203:1010:210:b5ff:feab:2dc2
#nettoie la SPD
setkey -FP
#nettoie la SAD
```

```
setkey -F
#exécute les opérations indiquées jusqu'à EOF
setkey -c << EOF
# policy to use between A and B
spdadd $A $B any -P out ipsec esp/transport//require;
spdadd $B $A any -P in ipsec esp/transport//require;
EOF
```

Script de configuration de la SPD sur B

```
#!/bin/csh
set A = 2001:0660:3203:1020:210:b5ff:feab:4cc4
set B = 2001:0660:3203:1010:210:b5ff:feab:2dc2

setkey -FP
setkey -F
setkey -c << EOF

# policy to use between A and B

spdadd $A $B any -P in ipsec esp/transport//require;
spdadd $B $A any -P out ipsec esp/transport//require;

EOF
```

Pour rendre ce script exécutable, il faut utiliser la commande `chmod +x nom_du_script`.

Pour définir la SPD sur la machine `B`, un script similaire (voir Script de configuration de la SPD sur B) doit être considéré. Noter que seule la direction de la politique change.

Afin de vérifier que les politiques de sécurité ont été correctement installées sur les deux machines, on peut utiliser la commande `setkey -DP` qui nous renseigne de plus sur la date de création ainsi que la date de dernière utilisation.

```
$(sur B) setkey -DP
2001:660:3203:1020:210:b5ff:feab:4cc4[any]   2001:660:3203:1010:210:b5ff:feab:2dc2[any]
any
in ipsec
esp/transport//require
created: Jan 12 14:46:07 2005 lastused: Jan 14 16:05:48 2005
lifetime: 0(s) validtime: 0(s)
spid=16408 seq=1 pid=21781
refcnt=1
2001:660:3203:1010:210:b5ff:feab:2dc2[any]   2001:660:3203:1020:210:b5ff:feab:4cc4[any]
any
out ipsec
ah/transport//require
created: Jan 12 14:46:07 2005 lastused: Jan 15 03:01:06 2005
lifetime: 0(s) validtime: 0(s)
spid=16409 seq=0 pid=21781
refcnt=1
```

a) Variantes possibles de la SPD

Nous proposons de donner d'autres exemples de configuration possibles. Pour ne protéger que le trafic ICMPv6 entre A et B (dans le sens A vers B) avec le protocole AH, on utilisera l'opération suivante :

spdadd $A $B icmp6 -P out ipsec ah/transport//require;

Pour protéger le trafic UDP entre A et B (de A vers B) avec ESP :

spdadd $A $B udp -P out ipsec esp/transport//require;

Pour protéger tout le trafic entre A et B (de B vers A) avec AH :

spdadd $B $A any -P out ipsec ah/transport//require;

On peut aussi vouloir utiliser le mode tunnel. Par exemple, on peut dire à A que ses paquets à destination d'une machine C doivent utiliser un tunnel entre A et B. B est alors chargé de transmettre les paquets non protégés à C. On utilisera pour cela la commande suivante :

spdadd $A $C any -P out ipsec esp/tunnel/$A-$B/require;

B) Configuration manuelle de la SAD

Pour que le trafic soit effectivement protégé par IPsec, il faut que les machines A et B partagent une ou des associations de sécurité IPsec. Dans notre exemple, A et B doivent partager deux SAs : une de A vers B et une de B vers A. Ces associations de sécurité peuvent être configurées soit manuellement soit dynamiquement en utilisant le protocole IKE. Dans cette partie, nous allons configurer ces deux SAs manuellement à l'aide du script de la See Script de configuration de la SAD sur A et B.

La commande `add` permet d'ajouter un SA dans la SAD. La première règle add permet de créer la SA utilisée entre A et B et de A vers B et la seconde crée la SA utilisée entre A et B mais de B vers A. Les paramètres utilisés ici sont :

- le SPI, qui vaut pour la première règle "`0x20001`",
- l'algorithme de chiffrement "`-E rijndael-cbc`",
- la clef utilisée pour le chiffrement "`GISELEGISELEGISELEGISELE`",
- l'algorithme d'authentification "`-A hmac-sha1`"
- la clef correspondante "`KAMEKAMEKAMEKAMEKAME`".

Vous pouvez vérifier que cette configuration fonctionne correctement à l'aide des commandes `tcpdump` et `ping6` par exemple, `tcpdump` permettant de voir le trafic arrivant et sortant d'une interface et `ping6` d'envoyer des paquets ICMPv6 ECHO_REQUEST vers une adresse IPv6.

Script de configuration de la SAD sur A et B

```
#!/bin/csh
set A = 2001:0660:3203:1020:210:b5ff:feab:4cc4
```

```
set B = 2001:0660:3203:1010:210:b5ff:feab:2dc2
setkey -F
setkey -c << EOF


#set SAD between A and B

add $A $B esp 0x20001 -E rijndael-cbc "GISELEGISELEGISELEGISELE" -A hmac-sha1
"KAMEKAMEKAMEKAMEKAME";
add $B $A esp 0x20002 -E rijndael-cbc "GISELEGISELEGISELEGISELE" -A hmac-sha1
"MEKAMEKAMEKAMEKAMEKA";

EOF
```

C) Configuration dynamique de la SAD

L'utilisation du protocole IKE permet la mise en place automatique des associations de sécurité. Il consulte la SPD afin de configurer les SAs correspondantes. Le logiciel `racoon` est une implémentation de ce protocole. C'est ce logiciel que nous allons utiliser pour automatiser la configuration de la SAD. Si le mode d'authentification choisi repose sur un secret pré-partagé, alors il est nécessaire d'éditer deux fichiers de configuration pour racoon. Ces deux fichiers se trouvent (après installation du logiciel) dans le répertoire `/usr/local/etc/racoon`. Le fichier `racoon.conf` est le fichier de configuration de racoon. Un exemple de configuration utilisée sur `A` et `B` est donné dans l'encadré suivant. Voici quelques éléments clés du fichier permettant de mieux en comprendre le contenu (le lecteur intéressé pourra obtenir des informations plus précises en tapant `man racoon.conf`) :

Fichier racoon.conf

```
# "path" must be placed before it should be used.
# You can overwrite which you defined, but it should not use due to confusing.
path include "/usr/local/etc/racoon" ;

# search this file for pre_shared_key with various ID key.
path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;

# "padding" defines some parameter of padding. You should not touch these.
padding
{
   maximum_length 20; # maximum padding length.
   randomize off; # enable randomize length.
   strict_check off; # enable strict check.
   exclusive_tail off; # extract last one octet.
}

# if no listen directive is specified, racoon will listen to all
# available interface addresses.
listen
{
   isakmp 2001:0660:3203:1020:210:b5ff:feab:4cc4 [500];
}

# Specification of default various timer.
timer
{
   # These value can be changed per remote node.
   counter 5; # maximum trying count to send.
   interval 20 sec; # maximum interval to resend.
   persend 1; # the number of packets per a send.
```

```
   # timer for waiting to complete each phase.
   phase1 30 sec;
   phase2 15 sec;
}

remote anonymous
{
   exchange_mode aggressive,main;
   doi ipsec_doi;
   situation identity_only;

   my_identifier address;

   nonce_size 16;
   lifetime time 1 min; # sec,min,hour
   initial_contact on;
   proposal_check obey; # obey, strict or claim

   proposal {
      encryption_algorithm 3des;
      hash_algorithm sha1;
      authentication_method pre_shared_key ;
      dh_group 2 ;
   }
}

sainfo anonymous
{
   pfs_group 1;
   lifetime time 30 sec;
   encryption_algorithm 3des ;
   authentication_algorithm hmac_sha1;
   compression_algorithm deflate ;
}
```

- `my_identifier` précise le type de l'identifiant des machines `A` et `B`. Ici, les machines sont identifiées par leurs adresses IPv6, donc le fichier contient :
  - `my_identifier address;`
- `authentication_method` précise la méthode d'authentification de la phase I. Il s'agit ici du mode `pre_shared_key` dans la partie `remote anonymous`
- `path pre_shared_key` indique le fichier (`/usr/local/etc/racoon/psk.txt`) contenant la clé

Le fichier `psk.txt` contient le secret pré-partagé (cf. Fichier `psk.txt<tt>), c'est-à-dire, il précise l'adresse de la machine distante suivie du secret partagé à savoir ici : 123456789abcde.`

Fichier psk.txt sur A :

```
2001:660:3203:1010:210:b5ff:feab:2dc2 123456789abcde
```

Fichier psk.txt sur B :

```
2001:660:3203:1020:210:b5ff:feab:4cc4 123456789abcde
```

`Une fois les fichiers correctement configurés, nous pouvons lancer le démon IKE à l'aide de la commande <tt>racoon` sur les machines A et B (cf. `man racoon` pour plus d'informations sur cette commande) :

```
#racoon -f /usr/local/etc/racoon/racoon.conf
```

Pour tester que la configuration fonctionne correctement, on peut utiliser la commande `ping6` entre `A` et `B`.

12) Critique des IPsec

Dès lors que l'on introduit de la sécurité dans un réseau, que ce soit pour filtrer les communications, ou pour protéger le trafic en confidentialité ou en authentification, les performances d'un réseau en termes de débit et temps de transit sont affectées. Cette diminution de performances est en partie due aux opérations de chiffrement/déchiffrement et de génération/vérification de l'authentificateur qui sont très gourmandes en temps de calcul.

Il faut également savoir que l'introduction de l'IPsec dans un réseau a un impact sur le fonctionnement même du réseau et sur l'usage que l'on souhaite en faire [Pui02-A] [Pui02-B] [Pui04-B]. En effet, IPsec actuel souffre des lacunes suivantes :

- Protéger une communication multicast avec IPsec n'est pas possible actuellement. Plus exactement, les protocoles AH et ESP autorisent le multicast. Cependant, le protocole de gestion des associations de sécurité IKE ne prévoit pas de négocier des associations de sécurité entre plus de deux équipements de sécurité.
- Le renouvellement de l'association de sécurité n'est pas transparent en ce sens que généralement pendant toute la durée du renouvellement, aucun trafic ne peut être échangé. A noter qu'un renouvellement d'association de sécurité est prévisible, car il est déclenché à chaque fois que l'association de sécurité arrive à expiration ou dès que le numéro de séquence a effectué un cycle (ceci pour éviter tout rejeu de paquet). Des méthodes de renouvellement "anticipées" existent donc, mais elles s'avèrent encore bien souvent problématiques.
- La fragmentation de paquets IP qui peut survenir dans un réseau IP a un impact sérieux sur les performances d'un réseau en termes de débit et de temps de transit. En effet, si un paquet se trouve fragmenté, il est nécessaire de le réassembler avant de procéder à son déchiffrement et/ou de vérifier la validité de son authentificateur. Or, ce réassemblage est très coûteux car il fait entre autres appel à des opérations de bufferisation. De plus, du fait de cette opération de bufferisation, il est possible qu'en période de forte charge, des paquets soient perdus.
- Le protocole DHCP permet entre autre d'allouer dynamiquement une adresse à un terminal. La difficulté est de savoir, au niveau d'une passerelle, quelle politique de sécurité appliquée au trafic en provenance de ces terminaux.
- Les IPsec et le DNS peuvent s'affecter de plusieurs façons. En particulier, l'exploitation d'une faille du DNS empêchant par exemple la résolution de nom peut bloquer le fonctionnement des IPsec du fait qu'IPsec peut identifier un correspondant en fonction de son nom de domaine. D'autre part, les réponses DNS de trop grande taille peuvent être bloquées par une passerelle IPsec si la fragmentation est interdite sur ces paquets ; en effet, la passerelle IPsec mettant en œuvre un tunnel IPsec voudra fragmenter les paquets avant de les encapsuler.
- La majorité des logiciels IPsec existants ne vérifient pas qu'un paquet reçu est correctement protégé [Pui03]. Ainsi, si un tunnel AH ou ESP est établi entre deux équipements IPsec et n'assure que les services d'intégrité et d'authentification, il est tout à fait possible d'usurper l'un de ces équipements et d'injecter du trafic dans les échanges. Bien entendu, cette attaque suppose que le terminal espion se trouve sur le même réseau local que l'équipement usurpé. Grâce à une attaque ARP (*Address Resolution Protocol*), il peut empoisonner le cache ARP des machines environnantes

et ainsi se voir rediriger tout le trafic destiné à l'équipement IPsec. Le tunnel n'étant pas chiffré, le terminal espion peut alors modifier ou forger des paquets en supprimant l'extension IPsec (AH ou ESP) et envoyer ces paquets non protégés vers l'équipement IPsec destinataire. Ce dernier ne vérifiant pas le niveau de protection attendue, il accepte les paquets et les traitent.

- Les extensions ESP et AH ne sont pas compatibles avec la traduction d'adresses/ports. Cependant, une solution palliative [RFC 3947] [RFC 3948] est en cours de définition pour ESP à l'IETF, ce qui permet d'offrir aux utilisateurs un accès VPN depuis un plus grand nombre de points d'accès au réseau. Noter que pour cette raison, mais également pour son service de confidentialité, l'extension ESP a la faveur des utilisateurs pour protéger les échanges IP.
- Actuellement, il n'existe pas d'infrastructure PKI mondiale. Les certificats de clés publiques n'ont le plus souvent qu'une portée nationale faute d'accord entre autorités de certification. Il est vrai que cette notion de confiance entre autorités de certification n'est pas anodine puisqu'elle est à la base de toute la sécurité. L'enjeu ici n'est pas technique, mais bien politique et juridique.
- Le manque de dynamicité dans la mise en œuvre des IPsec. Pour que les IPsec soient d'un usage plus courant, non limité à un domaine administratif défini, il faudrait que n'importe quel couple d'équipements IPsec puisse entrer en communication de façon improvisée, sans aucun arrangement initial sur des secrets cryptographiques, des certificats, sans faire appel à des services spécifiques... Aujourd'hui, cela n'est pas faisable car dans le meilleur des cas, si des certificats sont utilisés, il est nécessaire qu'ils soient reconnus de part et d'autre comme de confiance. Cela suppose dans les faits, que les équipements IPsec soient enregistrés auprès de la même autorité de certification, voire pour les équipements IPsec les plus tolérants, que le certificat du partenaire soit pré-chargé dans l'équipement.

13) Comparaison entre VPN IPsec et VPN SSL

Le terme VPN SSL (*Secure Socket Layer*) désigne simplement une encapsulation du trafic d'une application particulière (HTTP ou IMAP par exemple) par SSL/TLS afin d'atteindre les ressources informatiques de l'entreprise. Le VPN SSL ne concurrence pas directement les VPNs IPsec. Les constructeurs fournissant des VPNs SSL les positionnent comme une solution complémentaire à IPsec dans le but de répondre aux nouveaux besoins des entreprises. Le VPN SSL est une solution qui pallie aux difficultés rencontrées avec IPsec et les accès distants (notamment dues à la translation d'adresse et au filtrage d'application dans le réseau distant). Le principal avantage est de permettre aux utilisateurs nomades (télétravailleurs par exemple) de pouvoir se connecter au réseau de leur entreprise depuis n'importe quel ordinateur (PDA, cybercafé, réseau protégé par un firewall laissant passé les flux HTTP ...) sans modification à apporter sur le poste distant.

Ainsi on parle souvent de VPN Clientless : il n'y a pas besoin d'installer de logiciel spécifique sur le client pour atteindre le réseau de l'entreprise, contrairement au VPN IPsec. Ceci est possible avec des sessions HTTPS car SSL/TLS est aujourd'hui répandu sur tous les systèmes d'exploitation et supporté par n'importe quel navigateur Web.

Par ailleurs, cette proposition reste loin d'aboutir à un vrai standard VPN qui permet de multiplexer les données de plusieurs clients ainsi que celles des applications dans une seule session SSL/TLS.

XIII) Mobilité dans IPv6

Ce chapitre a pour objectif de décrire comment la mobilité a tiré partie d'IPv6 pour améliorer les mécanismes définis dans IPv4. Il introduit la vision IETF de la mobilité et les différents éléments du réseau qui en découlent. Sur la base de scénario, il décrit les flux de signalisation et de données, échangés lors des épisodes de mobilité. Afin de ne pas trop alourdir la présentation, tout en donnant une idée précise du protocole, seuls quelques formats de paquets sont donnés en exemple. Les mécanismes de sécurité mis en œuvre dans l'optimisation de la signalisation sont également décrits. Le chapitre aborde ensuite des sujets encore à l'étude comme certaines améliorations de performances et la mobilité des réseaux eux-mêmes. Il s'achève sur la présentation d'une implémentation expérimentale de la mobilité.

1) Introduction

Depuis quelques années déjà les terminaux informatiques deviennent moins encombrants et par conséquent plus mobiles. Par ailleurs les possibilités de se connecter à l'Internet se multiplient. Il s'en suit une mobilité induite par l'utilisation de plusieurs technologies d'accès (Ethernet, Wi-Fi, GPRS,...) sur un même terminal dans la même journée. Les études actuellement conduites par les constructeurs et les opérateurs pour fournir des infrastructures mobiles utilisant de nouvelles technologies radio, Wi-Fi et WiMax notamment, ont pour objet d'offrir la continuité des services en cours de déplacement, comme le permet le GSM dans le cas de la téléphonie mobile. Cela nécessite que les applications ne soient pas interrompues lors des épisodes de mobilité.

Enfin, les sociétés de transport désirant offrir un service de connexion à leurs clients en déplacement et les fabricants de véhicules, qui interconnectent de plus en plus d'équipements à bord, envisagent la question de la mobilité d'un réseau entier et non plus uniquement celle d'équipements isolés.

Le problème de la mobilité dans IP peut se décomposer en trois sous-problèmes distincts :

- pouvoir communiquer ;
- être joignable ;
- conserver les communications en cours lors des déplacements.

Le premier problème est élégamment résolu par le mécanisme d'auto configuration d'IPv6, en effet, dès que le terminal a réussi à construire une adresse IPv6 globale, il est capable de communiquer avec toute autre station sur l'Internet. Mobile IPv6 (MIPv6) modifie très peu ces mécanismes. Il ne requiert que de nouvelles directives de configuration permettant d'accélérer le processus. Le délai d'acquisition d'une adresse globale routable est en effet critique dans les situations de mobilité.

Le second problème est résolu pour les postes IP fixes par le DNS qui établit la relation entre un nom logique connu de tous et une adresse IP (Nommage). Dans le contexte de la mobilité, la fréquence

d'attribution d'une nouvelle adresse est incompatible avec la mise à jour du DNS distribué. D'autres mécanismes ont été proposés.

Le troisième problème est plus difficile à résoudre. Il a pour origine la dualité des fonctions d'une adresse IPv6. Elle identifie de manière unique sur l'Internet un terminal, ou pour être plus précis une interface réseau d'un terminal. Elle permet aussi de localiser un nœud dans la topologie de l'Internet. Ainsi chaque fois qu'un nœud se déplace, ce dernier doit changer d'adresse pour que la nouvelle adresse corresponde à sa nouvelle localisation (fonction de localisation). Malheureusement son identification change aussi ce qui pose des problèmes aux couches supérieures. En effet, TCP utilise le quintuplé : Adresse IPv6 source, Adresse IPv6 destination, Port source, Port destination et numéro de protocole pour identifier une connexion. Lorsqu'un de ces éléments change, il ne s'agit plus pour lui de la même session et les communications en cours sont interrompues.

Dès 1998, l'IETF a standardisé une solution de mobilité IP pour IPv4 [RFC 3344]. Les contraintes liées à la modification d'un protocole très largement déployé ont limité le travail à la modification du comportement du mobile lui-même (sans implication du correspondant pour qui la mobilité devait être transparente) et à l'ajout de nouvelles entités dans le réseau. IPv6 offrait l'opportunité du déploiement d'un nouveau protocole intégrant dès l'origine la mobilité. Les correspondants peuvent ainsi être mis à contribution pour des traitements liés à la mobilité. De plus la conception plus moderne d'IPv6 permet d'alléger les mécanismes d'encapsulation et de profiter des mécanismes d'auto configuration.

Des désaccords concernant la sécurisation de Mobile IPv6 (c.f. Les risques induits par la mobilité et leur limitation) et les différentes optimisations possibles, ont rendu la standardisation de Mobile IPv6 longue et laborieuse, les RFCs n'ayant été publiés qu'en juin 2004. La gestion de la mobilité dans IPv6 est maintenant définie dans le RFC 3775 pour ses aspects fonctionnels. Le RFC 3776 traite pour sa part des aspects liés à la sécurité de la signalisation de la mobilité.

Si les travaux dans le domaine de la mobilité IP se sont dans un premier temps exclusivement consacrés au support des stations mobiles, le besoin de fournir un accès Internet permanent aux routeurs mobiles et aux stations situées dans un réseau en mouvement (réseau mobile) est aujourd'hui clairement identifié. Les problèmes spécifiques posés par ce type de mobilité sont traités à l'IETF au sein du groupe de travail NEMO (NEtwork MObility) récemment créé. Ces travaux ont abouti à l'édition du RFC 3963 qui spécifie des fonctionnalités semblables à celles de MIPv6 dédiées aux routeurs mobiles.

2) La gestion de la mobilité IPv6

A) Le choix de l'IETF

Plusieurs alternatives sont possibles pour résoudre les problèmes introduits par la mobilité. La première d'entre-elles consiste à changer le fonctionnement même d'IP en modifiant les principes de l'adressage. Des propositions existent pour définir deux espaces d'adressage. Le premier serait dédié à la localisation et le second à l'identification. Mais il s'agit là d'un travail de très longue haleine. La seconde d'entre-elles consiste à ne rien changer au niveau d'IP et à laisser les couches supérieures gérer le problème. Il serait par exemple possible de modifier TCP pour gérer la mobilité (c'est-à-dire le changement d'adresse) au niveau transport. Malheureusement cela ne peut se faire qu'en modifiant la pile TCP ce qui est impensable étant donné le nombre de stations concernées. Par contre des propositions existent pour SCTP qui est un nouveau protocole de transport et n'est donc pas encore déployé à grande échelle. Le niveau applicatif peut aussi prendre en charge la mobilité en gérant la rupture et le ré-établissement automatique des connexions interrompues lors des handovers (épisodes de mobilité).

Toutes ces solutions supposent d'importants travaux de développement et sont difficiles à mettre en œuvre. Dans la mesure ou le développement d'IPv6 ne s'accompagne pas nécessairement de la modification en profondeur des niveaux protocolaires supérieurs, l'IETF a eu la volonté de gérer la mobilité au niveau de la pile IP et de la rendre transparente aux niveaux supérieurs utilisant le service IP.

Le groupe de travail Mobile IP s'est donc appuyé sur une solution basée sur deux adresses IP et sur le routage « normal » des paquets pour assurer la gestion de la mobilité des nœuds. Des améliorations apportées par la version 6 d'IP et des éléments spécifiques à MIPv6 ont été utilisés pour assurer au mieux la transparence des déplacements. MIPv6 utilise ou définit l'emploi des éléments suivants :

- les en-têtes d'extension protocolaire (protocole 135) ;
- les en-têtes de routage (nouveau type 2) ;
- les en-têtes destination ;
- les mécanismes d'annonce des routeurs (ICMPv6) ;
- la gestion de l'obsolescence des adresses ;
- la sécurisation des paquets (IPsec).

B) Vue générale de la gestion de la mobilité IPv6

Les mécanismes d'IPv6 vus dans les chapitres précédents offrent une très bonne base à la gestion de la mobilité. En effet, ils résolvent un certain nombre de problèmes qu'avaient à résoudre les solutions de mobilité IPv4. Ainsi, le mécanisme de configuration sans état permet au terminal mobile (MN : *Mobile Node*) en déplacement d'acquérir une adresse IPv6 globale topologiquement valide. Il peut dès lors communiquer sans contrainte. Le mécanisme d'annonce des routeurs facilite quant à lui la détection du mouvement qui est essentielle à la gestion de la mobilité.

Le principe à la base de la mobilité IPv6 est de séparer les fonctions d'identification et de localisation toutes deux traditionnellement assurées par l'adresse IP. Il s'en suit que lorsque le mobile se déplace, il doit changer d'adresse IP puisque celle-ci le localise dans le réseau. De ce fait, il perd son identité et n'est plus directement joignable à l'adresse connue de ses correspondants et du service de nom. Il est toujours possible d'enregistrer la nouvelle adresse dans un service de nom dynamique (DDNS) mais cela induit un délai très important et ne résout qu'une partie du problème. En effet, l'adresse IP est aussi utilisée par les couches transport (UDP et TCP) pour identifier une connexion. Lorsque l'adresse IP du mobile change, les connexions TCP en cours sont donc rompues.

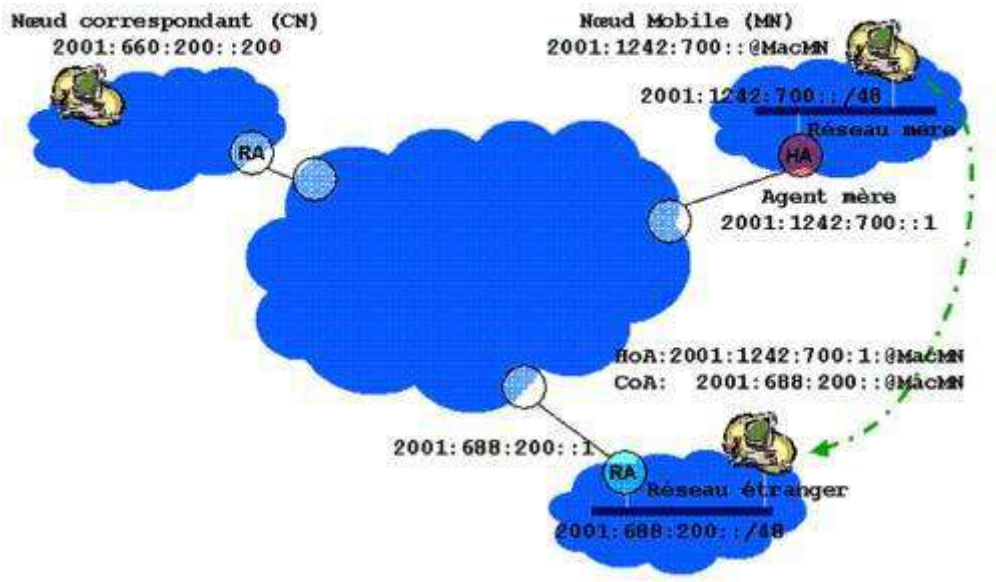


Figure 13-1 *Différentes adresses utilisées dans la mobilité IPv6*

Pour éviter cela, mobile IP permet au mobile de conserver l'adresse utilisée dans son réseau d'attachement. A des fins d'identification, on parle de home address (HoA) ou adresse mère et de home network ou réseau mère. Ainsi, du point de vue des couches supérieures, le mobile conserve son adresse mère (HoA) quelle que soit sa localisation. Par ailleurs, il acquiert une adresse nouvelle temporaire appelée care-of address (CoA) ou adresse temporaire, locale celle-ci, dans chacun des réseaux qu'il visite, dits réseaux étrangers. Ce second type d'adresses est utilisé à des fins de localisation. Du point de vue de la pile IPv6 le noeud mobile communique toujours avec l'adresse temporaire sauf lorsqu'il est attaché à son réseau mère. Du point de vue des couches protocolaires supérieures le mobile communique toujours avec son adresse mère (cf. figure Différentes adresses utilisées dans la mobilité IPv6).

Une nouvelle entité, le home agent (HA) ou agent mère, localisé dans le réseau mère est chargé d'assurer la correspondance entre la HoA et la CoA du mobile lorsque celui-ci est attaché à un réseau étranger. Cet agent est également chargé de réacheminer les paquets IP à destination de l'adresse mère du mobile vers son adresse temporaire dans son réseau visité.

a) Le mobile dans son réseau mère.

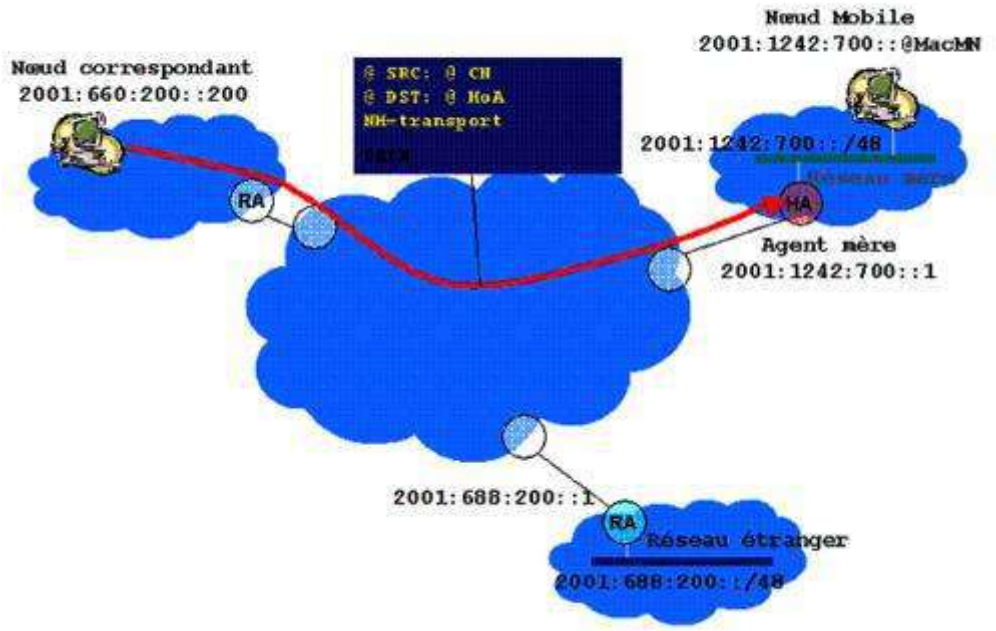


Figure 13-2 *Envoi de paquets à un mobile situé dans son réseau mère*

Lorsque le mobile est attaché au réseau mère (cf. figure Envoi de paquets à un mobile situé dans son réseau mère), il dispose de son adresse mère et communique normalement en utilisant sa HoA comme adresse source. Les paquets qui lui sont destinés comprennent l'adresse mère comme adresse destination et sont routés en fonction du préfixe du réseau mère. L'agent mère est inactif. Il n'y a pas de problème de sécurité supplémentaire induit par la gestion de la mobilité puisque le mobile communique de la même manière que n'importe quel noeud IPv6 sur l'Internet.

Le réseau mère n'est pas nécessairement un réseau sur lequel le mobile peut s'attacher, son rôle principal étant d'accueillir le home agent et les adresses mères des mobiles qu'il gère. Il y a toutefois une relation administrative forte entre le mobile et son réseau mère.

b) Le mobile dans un réseau étranger

Lorsque le mobile est attaché à un réseau étranger, il dispose, en plus de son adresse mère, d'une ou plusieurs adresses temporaires routables acquises par les mécanismes d'auto-configuration avec ou sans états. Une de ces adresses est choisie comme adresse temporaire primaire et est transmise à l'agent mère pour créer une association entre la HoA et cette CoA primaire.

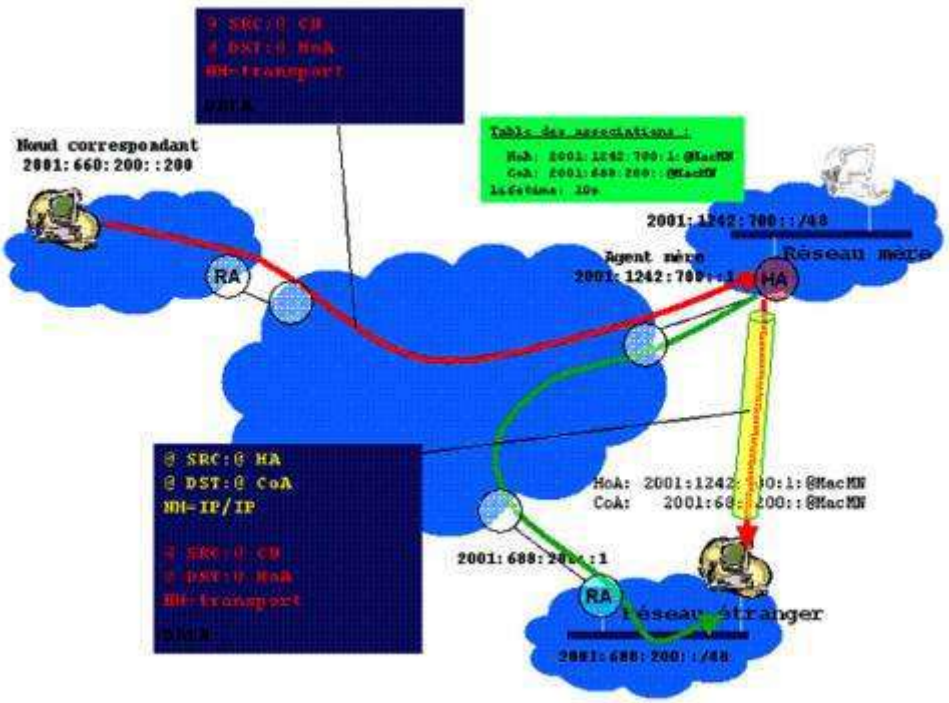


Figure 13-3 *Envoi de paquets à un mobile situé hors de son réseau mère - tunnelage*

L'agent mère maintient une "table des associations" contenant les associations de tous les mobiles qu'il gère et qui sont en visite dans un réseau étranger (cf. figure Envoi de paquets à un mobile situé hors de son réseau mère - tunnelage). Grâce à ces informations, il peut faire suivre les paquets à destination de la HoA d'un des mobiles vers la CoA primaire de ce dernier. Il encapsule pour cela les paquets en utilisant l'extension d'en-tête IP-IP d'IPv6. Les paquets ainsi tunnelés sont protégés par IPsec.

Le paquet IP retransmis vers le mobile comporte comme adresse source celle du HA et comme adresse destination la CoA primaire du mobile. Le paquet atteint le réseau étranger puisque la CoA primaire a pour préfixe un de ceux du réseau étranger. Le mobile réceptionne ce paquet et découvre l'extension d'en-tête IP dans IP. Il supprime l'en-tête extérieur et remet le paquet aux couches supérieures comme si le mobile avait réceptionné le paquet dans son réseau mère. Ce paquet à donc pour adresse destination l'adresse mère du mobile et pour adresse source l'adresse IPv6 du correspondant émetteur du message. Le quintuplé TCP d'identification d'une session n'est pas modifié. La communication n'est plus rompue lors d'un épisode de mobilité sans qu'il ait été nécessaire de modifier le protocole TCP.

Les paquets issus du mobile dans un réseau étranger et à destination du correspondant utilisent un principe similaire. Cependant dans le cas présent les paquets sont reverse tunnelés via le HA. le paquet IP ainsi transmis comporte comme adresse source la CoA primaire du mobile et comme adresse destination l'adresse du HA. le paquet IP encapsulé comporte comme adresse source la Home Address et comme adresse destination celle du correspondant. Les paquets ainsi transmis sont protégés par IPSEC, traversent les routeurs jusqu'au HA qui supprime l'en-tête extérieur et forwarde le paquet résultant au correspondant.

De cette manière le correspondant voit la communication comme venant directement du réseau mère du mobile. Pour effectuer ce traitement, le correspondant ne maintient aucun état spécifique aux mobiles avec qui il communique. Ainsi un mobile peut communiquer avec des correspondants ayant un support

minimum de la mobilité. Ce support minimum de la mobilité ne monopolise aucune ressource particulière au niveau du correspondant, et n'induit pas de nouveaux problèmes de sécurité ou de performances.

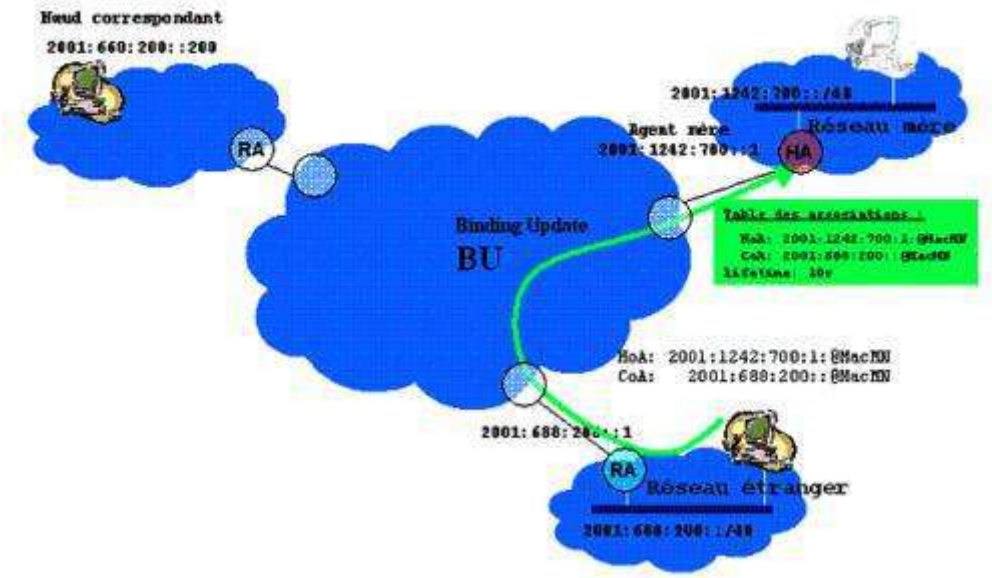


Figure 13-4 *Mise à jour d'association entre le mobile et l'agent mère*

Pour créer une association, ou la mettre à jour lorsqu'il change de réseau étranger, un mobile utilise une signalisation propre à la mobilité IPv6 appelée Binding Update (BU) ou mise à jour d'association. Cette signalisation utilise un nouveau protocole (135) basé sur les extensions d'en-tête IPv6 (cf. figure Mise à jour d'association entre le mobile et l'agent mère).

Toutefois, même avec ce mécanisme très simple, les applications, ou les utilisateurs, n'ont pas conscience de la mobilité et par conséquent du fait que le terminal n'est pas dans son réseau mère. Si bien, qu'elles peuvent faire confiance à un environnement réseau connu entre le correspondant et le noeud mobile, par exemple entre deux bâtiments d'un même campus. Le fait de se trouver dans un réseau visité à l'insu des applications peut donc causer des problèmes de sécurité puisque les paquets vont circuler sur une portion de réseau inconnu et potentiellement non sûr. Pour éviter cela, il est possible d'utiliser le tunnel sécurisé entre le mobile et le "home agent" dans les deux sens. La sécurité est ainsi assurée sans que le correspondant local n'ait à supporter la mobilité.

c) Optimisation dans le cas du mobile dans un réseau étranger

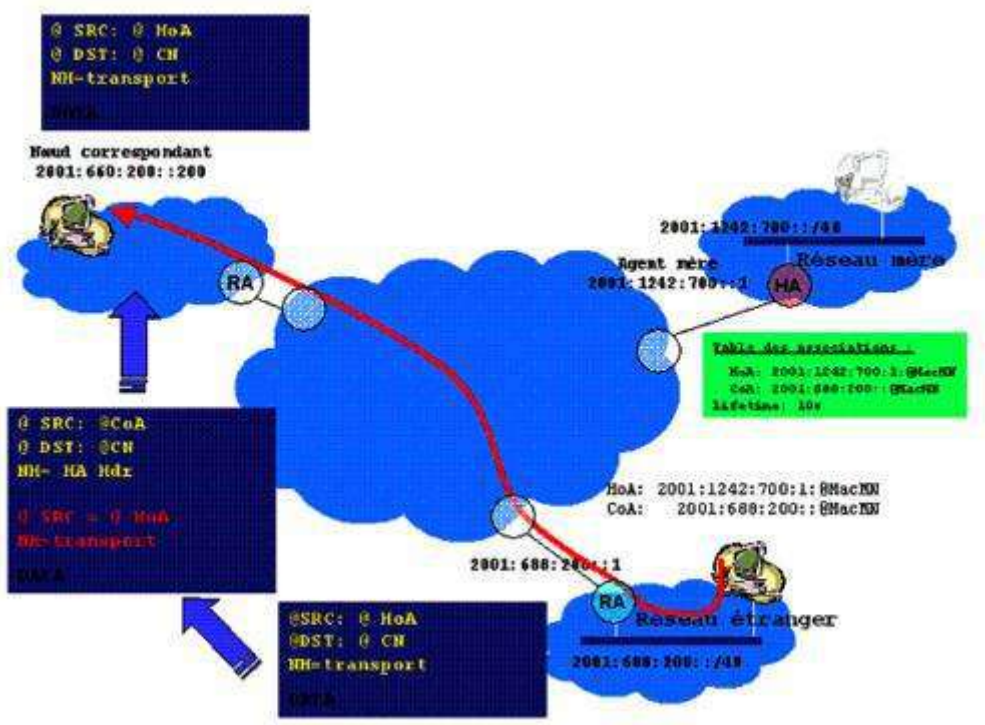


Figure 13-5 *Envoi de paquets par un mobile situé hors de son réseau mère*

Le routage systématique par l'agent mère du mobile est simple à mettre en oeuvre et très sûr puisque la communication entre l'agent mère et le mobile est sécurisée par IPsec (cf. figure Envoi de paquets à un mobile situé dans son réseau mère - tunnelage inverse). Par contre, elle est particulièrement inefficace au niveau du routage. En effet, si le mobile est en déplacement loin de son réseau mère et qu'il communique avec un serveur proche de lui, il est plus efficace de communiquer directement que de passer par l'agent mère (cf. figure Envoi de paquets par un mobile situé hors de son réseau mère). Cela permet d'économiser des ressources dans l'Internet et au niveau du réseau mère qui pourrait avoir des difficultés à monter en charge si les communications de tous les mobiles passent par lui. C'est pourquoi, l'optimisation de routage, qui était une option de mobile IPv4, a été intégrée à Mobile IPv6.

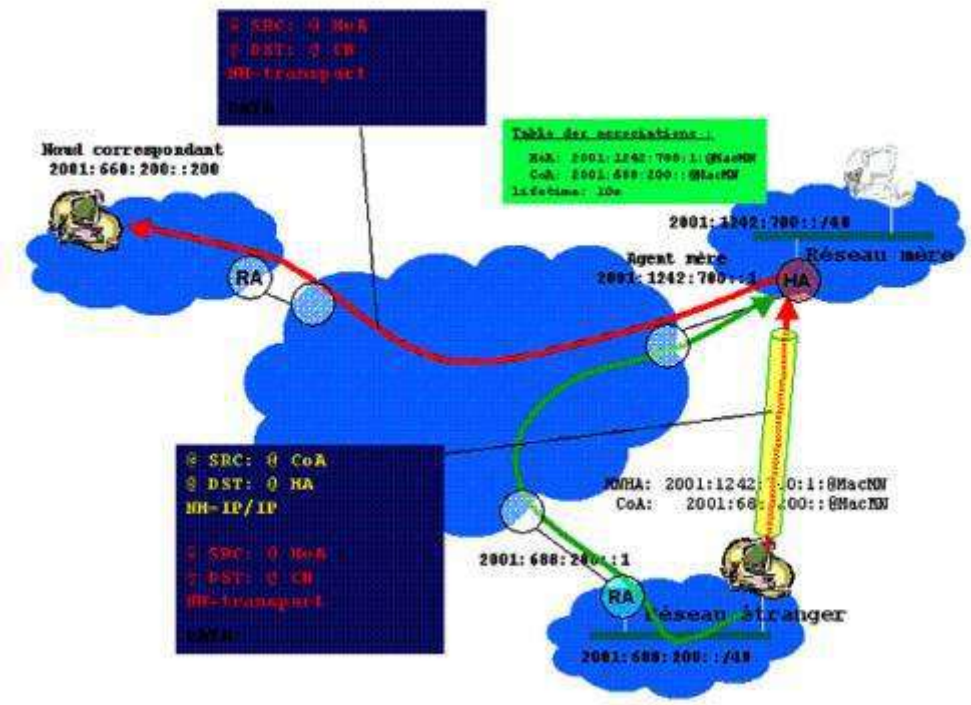


Figure 13-6 *Envoi de paquets à un mobile situé dans son réseau mère - tunnelage inverse*

L'optimisation de routage de Mobile IPv6 profite du fait que IPv6 est un nouveau protocole et que des mécanismes liés à la mobilité peuvent lui être intégrés lors de son déploiement. Ainsi pour supporter l'optimisation de routage les nœuds correspondants doivent intégrer des fonctions spécifiques liées à la mobilité.

Lorsqu'un correspondant supporte l'optimisation de routage, il maintient comme l'agent mère une table des associations pour tous les mobiles utilisant l'optimisation de routage avec lesquels il est en communication. Le mobile met à jour l'association qui le concerne en envoyant un message de mise à jour d'association (BU : Binding Update) au correspondant sitôt après qu'il a informé l'agent mère de sa nouvelle localisation. Il doit pour cela maintenir une table des mises à jour d'associations contenant toutes les associations qu'il doit entretenir auprès des correspondants et de l'agent mère. Cet entretien se fait à chaque déplacement et lorsqu'une mise à jour d'association arrive à échéance en échangeant des messages de type BU (cf. figure Mise à jour d'association entre le mobile et le correspondant - route optimisée).

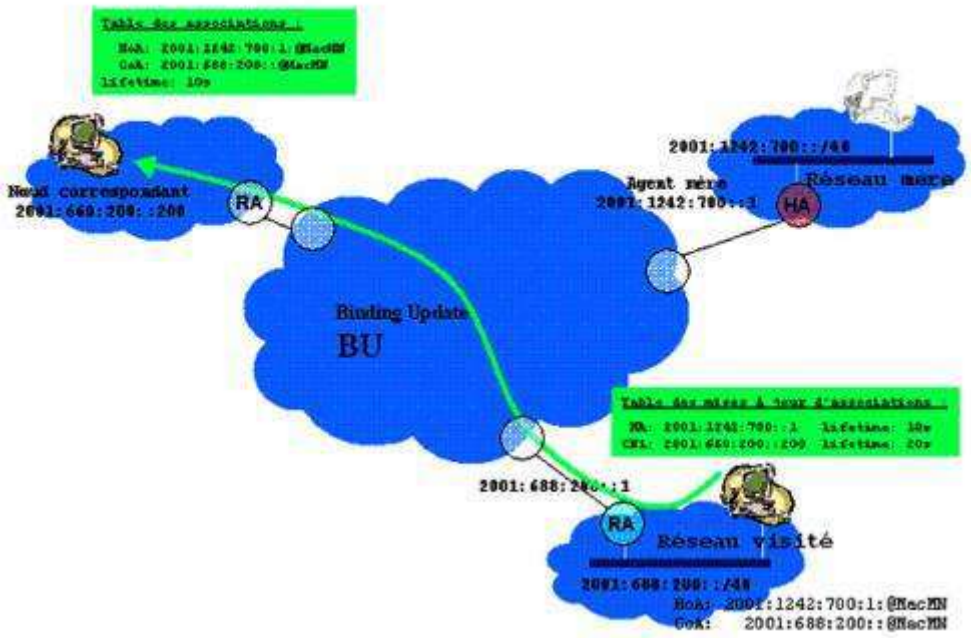


Figure 13-7 *Mise à jour d'association entre le mobile et le correspondant - route optimisée*

Le correspondant qui émet un paquet à destination d'un mobile et qui utilise l'optimisation de routage, trouve dans sa table des associations une association entre la HoA et une CoA. Il remplace alors l'adresse de destination par la CoA et ajoute une extension d'en-tête de routage particulière (de type 2) contenant la HoA du mobile comme adresse de destination finale (cf. figure Envoi de paquets à un mobile situé hors de son réseau mère - route optimisée).

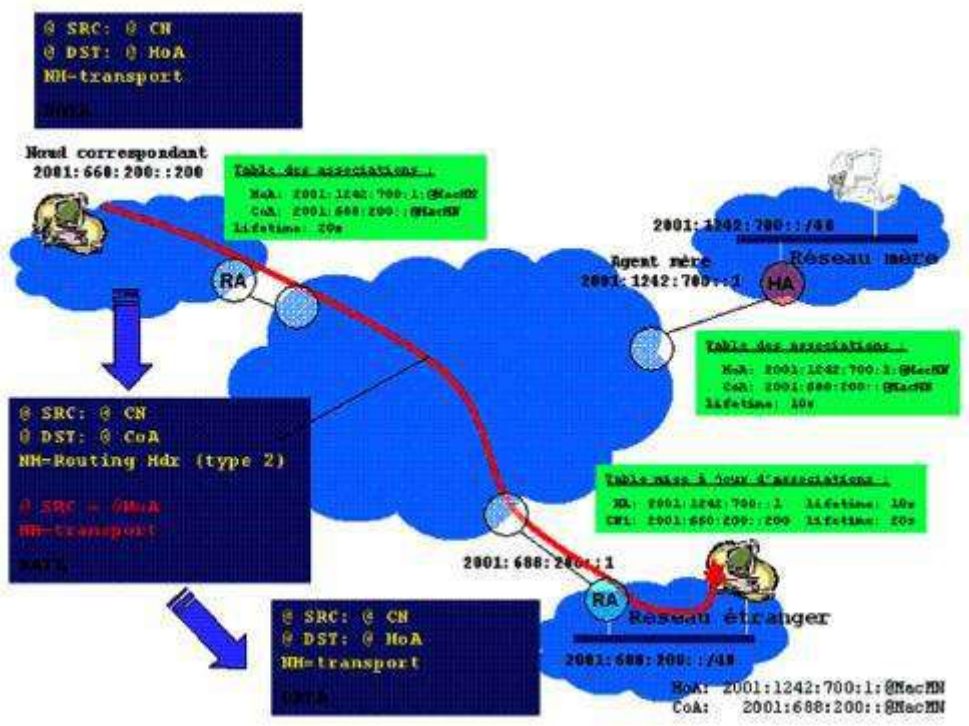


Figure 13-8 *Mise à jour d'association entre le mobile et le correspondant - route optimisée*

Les extensions d'en-tête de routage peuvent être filtrées pour des raisons de sécurité et pour qu'elles ne soient pas utilisées pour contourner les politiques de sécurité. Le type de l'extension d'en-tête de routage utilisé par Mobile IPv6 est différent pour permettre aux passerelles de sécurité d'appliquer des règles spécifiques aux paquets contenant un en-tête de routage liés à la gestion de la mobilité.

Tous les correspondants IPv6 potentiels ne supportent pas forcément l'optimisation de routage. Dans ce cas, un correspondant répond qu'il ne comprend pas la mise à jour d'association et les communications continuent à passer à travers l'agent mère. Pour, ce faire il utilise un message ICMPv6 spécifique qui informe le nœud mobile.

Lorsque le mobile veut envoyer un paquet à un correspondant, il vérifie au préalable si une optimisation de route a été initialisée. Dans ce cas précis uniquement, il émet les paquets à destination du correspondant en utilisant sa CoA et en ajoutant une extension d'en-tête spéciale appelée HoA. Cette extension d'en-tête est ajoutée de façon transparente aux couches supérieures pour qui la communication est toujours entre le correspondant et la HoA du mobile. Elle comprend la HoA du mobile, la CoA du mobile étant utilisée normalement pour émettre le paquet. Ainsi l'adresse source du paquet dispose d'un préfixe valide dans le réseau étranger et les paquets ne seront pas filtrés par le routeur de sortie. Lorsqu'il reçoit ce paquet, le correspondant supprime l'extension d'en-tête "home address" et remplace l'adresse source du paquet par la HoA trouvée dans l'extension. Il remet ensuite le paquet aux couches supérieures qui le considèrent comme venant directement du réseau mère du mobile.

Le mécanisme de mise à jour d'association pose d'importants problèmes de sécurité. En effet, il est aisé de protégé les échanges de signalisation entre le mobile et l'agent mère du fait de la relation administrative qui permet par exemple l'utilisation d'un secret partagé. C'est beaucoup plus compliqué en ce qui concerne les correspondants et pourtant la sécurité des mises à jour d'association est vitale. Sans protection, il serait possible de détourner les communications d'un mobile en redirigeant le trafic pour l'espionner ou de mener une attaque par déni de service. C'est pourquoi une procédure appelée "test de routage retour" est spécifiée (voir Sécurisation de la signalisation avec les nœuds correspondants).

d) Traitement du multicast

Pour la gestion du multicast, il faut considérer séparément les différents types de groupes multicast auxquels un nœud mobile adhère : il y a d'une part les groupes multicast auxquels toute station IPv6 adhère (par exemple le groupe multicast de sollicitation) et ceux auxquels une station adhère pour des besoins applicatifs spécifiques. Enfin il faut considérer la portée des groupes multicast qui peut être globale, locale ou correspondre à un site donné.

Pour ce qui est de la portée, le problème est relativement simple, les paquets multicast de portée locale ne doivent pas être retransmis à un nœud mobile qui se trouve dans un réseau visité et d'une manière générale, ceux qui n'ont pas une portée globale ne devraient pas l'être.

Lorsqu'un nœud mobile se trouve dans son réseau d'origine (réseau mère) il se comporte comme toute station IPv6 et traite normalement les paquets multicast. Par contre lorsqu'il se trouve dans un réseau étranger, il y a deux manières de gérer les flux multicast :

- soit l'agent mère retransmet vers le mobile les paquets à destination des groupes auxquels ce dernier a adhéré,
- soit le mobile ré-adhère aux groupes multicasts qui l'intéressent à chaque fois qu'il se déplace.

La première méthode suppose que l'agent mère dispose de la plupart des fonctions d'un routeur multicast. Il doit pouvoir comprendre et initier les échanges concernant l'appartenance du nœud mobile aux différents groupes et lui retransmettre les paquets à travers le mécanisme de tunneling.

Dans le second cas, l'agent mère n'assure pas le relayage des paquets multicast et le mobile qui souhaite écouter un groupe après un changement de réseau IPv6 doit se réabonner au groupe en question. Cette solution suppose que l'application prenne en compte la mobilité pour relancer la procédure d'adhésion avec la nouvelle CoA. Ceci constitue un des problèmes que MIPv6 souhaite éviter. Une autre difficulté vient de ce que l'interruption dans la réception d'un flux multicast entre le changement de réseau et la réception du premier paquet multicast à la nouvelle localisation peut être très longue du fait de la procédure d'adhésion.

Lorsque l'agent mère assure la gestion du multicast, le nœud mobile peut choisir pour chaque groupe multicast auquel il adhère, s'il passe par l'agent mère ou par une adhésion directe. Dans ce cas l'agent mère doit supporter le protocole de gestion des groupes multicast (i.e. MLD). De son côté, le mobile qui veut pouvoir profiter de cette fonctionnalité, doit implémenter la partie du protocole de gestion des groupes qui concerne l'hôte et être capable de traiter les paquets multicast encapsulés par l'agent mère.

Lorsque le mobile souhaite recevoir les paquets multicast destiné à un groupe particulier, il émet un paquet MLD d'adhésion au groupe avec pour destination l'adresse multicast des routeurs MLD et pour portée la valeur 1. Ce paquet est transmis à l'agent mère par l'intermédiaire du tunnel inverse entre le mobile et l'agent mère. De la même manière lorsqu'il ne souhaite plus recevoir les paquets de ce groupe, il informe l'agent mère qu'il quitte le groupe par le même moyen. De son côté, l'agent mère agit comme un routeur multicast standard et interroge régulièrement les mobiles en visite sur leur appartenance à des groupes multicast, en utilisant des messages MLD transmis dans les tunnels vers les mobiles.

Ce paquet est transmis à l'agent mère par l'intermédiaire du tunnel inverse entre le mobile et l'agent mère. De la même manière lorsqu'il ne souhaite plus recevoir les paquets de ce groupe, il informe l'agent mère qu'il quitte le groupe par le même moyen. De son côté, l'agent mère agit comme un routeur multicast standard et interroge régulièrement les mobiles qui sont dans des réseaux visités sur leur appartenance à des groupes multicast en utilisant des messages MLD transmis dans les tunnels vers les mobiles. De cette manière, il maintient, pour chaque mobile, une vue à jour des groupes multicast auxquels il appartient et lui fait suivre les paquets multicast par l'intermédiaire du tunnel.

Pour l'émission de paquet multicast, le mobile possède également deux solutions lorsqu'il est hors de son réseau mère :

- Emission des paquets en utilisant le routeur multicast local (celui du réseau visité). Dans ce cas le mobile utilise l'adresse temporaire locale (CoA) comme adresse source et ne doit pas utiliser l'option home address dans les paquets multicast. Dans ce cas, cela implique que l'application doit gérer l'existence d'une adresse locale.

- Utilisation du tunnel pour émettre les paquets à partir du réseau d'origine du mobile à travers l'agent mère. Dans ce cas les déplacements restent transparents aux applications émettant du trafic multicast au niveau du mobile.

La première alternative pose des problèmes insolubles aux protocoles de routage multicast chargés de construire les arbres de diffusion dans le réseau. En effet, à chaque fois que le mobile change de localisation dans l'Internet, les arbres de diffusion déjà construits sont soit inutilisables soit ne correspondent plus au meilleur choix possible. De plus les membres du groupe adhèrent souvent à un groupe et à une source spécifique, ce qui suppose qu'à chaque changement de localisation du mobile, tout le processus d'adhésion recommence avec une nouvelle adresse de source et que l'arbre de diffusion spécifique à la source soit reconstruit.

La gestion du multicast pose d'importantes difficultés pour la réception aussi. Elle suppose un important travail de la part de l'agent mère qui doit déjà assurer le transfert des paquets unicasts lorsque l'optimisation de routage n'est pas utilisée. En effet, lorsque l'agent mère transformé pour l'occasion en routeur multicast IPv6 doit émettre une requête de demande d'appartenance aux groupes, il ne peut pas la transmettre en multicast comme un routeur multicast IPv6 normal, il doit au contraire envoyer une copie de la requête dans chacun des tunnels correspondant aux mobiles qui se sont enregistrés auprès de lui. De la même manière, il va devoir traiter autant de réponses qu'il y a de mobiles et ceux-ci ne peuvent pas utiliser les mécanismes de MLD qui permettent de limiter le trafic de signalisation.

C) Déplacements des mobiles

La raison d'être de Mobile IPv6 est de gérer les déplacements des mobiles. Pour cela il faut être capable de détecter les changements de réseau, d'obtenir une nouvelle CoA, et informer l'agent mère et les correspondants du changement de localisation (i.e. de CoA). L'ensemble de ces opérations sont regroupées dans le "handover".

Contents

a) Détection du changement de réseau

La phase de détection de mouvement est cruciale. Elle représente une bonne part du délai d'interruption observé lors des changements de réseau. Le mobile utilise la gestion de voisinage pour détecter qu'un voisin, en l'occurrence son routeur par défaut, n'est plus accessible. Le défaut de ce mécanisme est que le mobile ne détecte la perte du routeur par défaut que lorsqu'il a des données à transmettre, ce qui retarde la détection du changement de réseau et augmente d'autant la durée des interruptions.

Une solution alternative suppose la coopération du routeur IPv6 qui ajoute dans les annonces de routeur une option indiquant le délai entre deux annonces. Le mobile qui écoute en permanence les annonces émises peut alors déduire de la perte de plusieurs annonces successives qu'il vient probablement de changer de réseau. Une autre adaptation demandée au routeur IPv6 concerne la réduction du délai entre deux annonces successives pour améliorer encore la vitesse de détection, ainsi il est conseillé d'émettre une annonce de routeur non sollicité toutes les 50 ms en moyenne (au lieu de 3s). Évidemment une telle configuration induit un trafic non négligeable pour certains types de réseau (réseau locaux sans fil).

Il est important de réduire le nombre de *handover* car ceux-ci induisent une rupture temporaire des communications et une signalisation importante dans le réseau. Le mobile peut utiliser d'autres informations pour décider de la nécessité d'effectuer un handover, mais il doit le faire prudemment. Par exemple, le mobile ne peut pas utiliser la découverte d'un nouveau réseau (une annonce de routeur avec un nouveau préfixe) pour décider qu'il a perdu l'accès à son ancien réseau. Il est, en effet, possible de recevoir simultanément des annonces en provenance de plusieurs routeurs annonçant des préfixes différents sur un même réseau.

Lorsqu'il reçoit une annonce de routeur, le mobile doit prendre en compte le préfixe annoncé et non l'adresse source des annonces de routeur pour détecter un changement, car des routeurs appartenant à des réseaux différents peuvent utiliser la même adresse lien-local. Dans ce cas, le bit `R` qui permet d'indiquer une adresse globale du routeur peut lever l'ambiguïté.

Notons que les mécanismes de détection de changement de réseau décrits sont complètement indépendants du niveau liaison. Il est possible de prendre en compte les informations connues du niveau liaison, comme le fait que le mobile vient de perdre ou d'acquérir un nouveau réseau sans-fils ou une nouvelle interface, pour déclencher la procédure du handover IPv6. Emettre une sollicitation de routeur aussitôt qu'un changement de réseau d'accès est soupçonné, permet de gagner un temps important mais suppose une implémentation plus complexe, puisque dépendante d'un niveau liaison particulier. En contrepartie, il est possible d'éviter l'envoi fréquent d'annonces de routeur non sollicitées.

b) Configuration de l'adresse temporaire

Une fois que le mobile a détecté la perte du routeur par défaut, il doit acquérir une nouvelle adresse en sollicitant un routeur. A réception d'une annonce de routeur sur le nouveau réseau il peut découvrir le préfixe du réseau et configurer une adresse globale appartenant à ce préfixe qui sera la nouvelle adresse temporaire (CoA). Lors de cette configuration, le mobile doit effectuer une procédure de détection d'adresse dupliquée (DAD) pour la nouvelle adresse. Sous certaines conditions, on cherchera à réduire la durée de cette procédure en émettant la sollicitation de voisin sans attendre le délai aléatoire habituel. D'autres propositions ont été faites pour ne pas attendre la seconde réglementaire qu'une éventuelle station défendant l'adresse réponde à la sollicitation de voisin.

Notons que le mobile peut configurer une adresse pour chaque préfixe annoncé sur le lien local. Toutes ces adresses seront des care-of address. Mais il devra choisir l'une d'entres-elles pour mettre à jour les associations au niveau du HA et des correspondants. Elle sera dite primary care-of address ou adresse temporaire primaire.

c) Avertissement de l'agent mère

Dès que le mobile a changé de care-of address principale, il doit en informer l'agent mère en envoyant une mise à jour d'association (*binding update*). Cette mise à jour peut être la première, dans ce cas, l'agent mère crée l'association. Dans le cas contraire, il met à jour l'association courante. Une mise à jour d'association, sera par la suite envoyée régulièrement, avant le délai d'expiration, pour maintenir l'association.

Lorsqu'il crée une nouvelle association, l'agent mère effectue la procédure de détection de duplication d'adresse (DAD) pour la home address avant d'acquitter l'association au mobile. Si un autre mobile ou une station sur le réseau mère possède cette adresse il répond au mobile que l'adresse est déjà utilisée et ce dernier doit essayer une autre adresse.

d) Découverte dynamique de l'agent mère

Il peut arriver que le mobile ne connaisse pas l'adresse d'un agent mère sur son réseau mère ou que l'agent mère qu'il connaissait ne réponde plus. Dans ce cas, le mobile doit tenter de découvrir l'adresse d'un agent mère apte à assumer ce rôle pour lui. Il envoie pour cela un paquet ICMP à l'adresse anycast des "Agents mère IPv6" pour le préfixe de son réseau mère.

Lorsque le mobile reçoit une réponse celle-ci contient une liste ordonnée des adresses globales des HAs du réseau mère. Il essaye ensuite dans l'ordre les adresses des agents mère de la liste jusqu'à recevoir un acquittement positif à sa demande de mise à jour d'association.

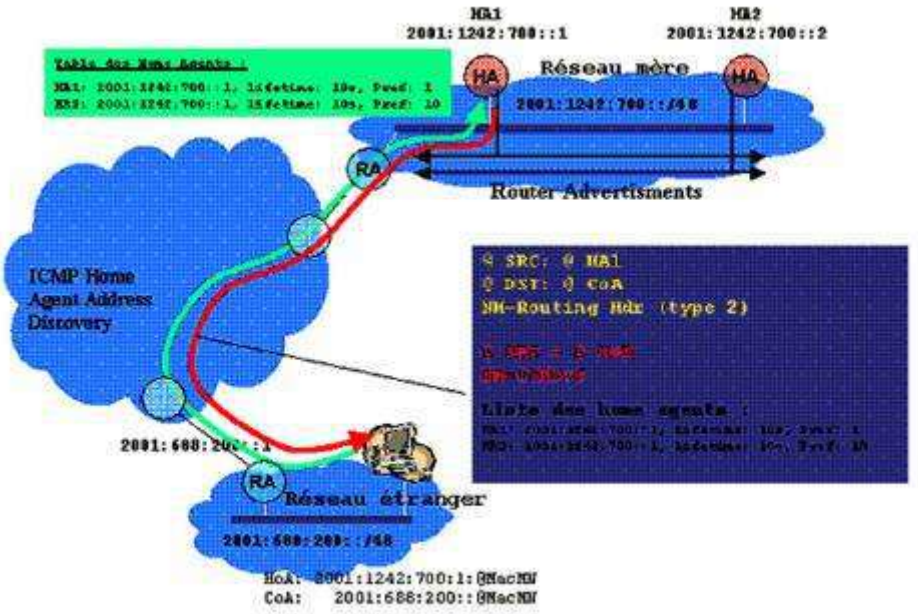


Figure 13-9 *Découverte dynamique de l'agent mère*

Pour supporter ce service, chaque HA doit être capable de découvrir les autres HAs sur le réseau mère pour en maintenir la liste. Il écoute pour cela les annonces de routeur émises sur le lien. Celles qui annoncent offrir le service d'agent mère (bit H : *Home Agent* validé) sont ajoutées à la liste. L'annonce de HA peut contenir d'autres informations utiles dans l'option home agent advertisement option : le délai de validité (*lifetime*), une adresse globale et une préférence. Cette dernière est utilisée pour ordonner la liste transmise au mobile.

Dans l'exemple, figure Découverte dynamique de l'agent mère, la requête ICMP du mobile atteint l'agent mère 1 mais ce denier renvoie une liste indiquant que l'agent mère 2 est plus prioritaire. Le mobile continuera donc la procédure en demandant à l'agent mère 2 d'enregistrer l'association.

Ce mécanisme pourra être implémenté pour distribuer la fonction d'agent mère sur plusieurs serveurs et pour répartir dynamiquement la charge entre les différents serveurs. La charge des agents mère est un point très critique de la mobilité IP et il est nécessaire de trouver des solutions permettant de résister au facteur d'échelle.

e) Interception des paquets par l'agent mère

L'agent mère doit assurer l'interception des paquets à destination du mobile dès qu'il dispose d'une association entre l'adresse mère (HoA) et une adresse temporaire (CoA) valide. Pour cela il diffuse une annonce de voisin non sollicitée sur le réseau mère. Cette annonce indique aussi que toutes les tables de voisinage doivent être mise à jour pour associer la HoA du mobile avec l'adresse de niveau 2 de l'agent mère. Comme le multicast n'est pas fiabilisé ce message est généralement émis plusieurs fois.

Ensuite, à chaque fois qu'une sollicitation de voisin concerne la HoA d'un mobile qu'il gère, le HA répond en lieu et place du mobile, pour associer la HoA du mobile avec son adresse de niveau 2. Il assure ainsi la défense de la HoA enregistrée dans l'association lors des procédures de détection de duplication d'adresse. Il répondra qu'il possède l'adresse si un autre mobile ou une autre station du réseau mère tente de la configurer.

Lorsque le HA a des paquets à transmettre au mobile, il doit agir comme un correspondant. Il utilise donc un en-tête de routage de type 2. Si par contre il s'agit de paquet qu'il intercepte pour le compte du mobile, ils sont encapsulés en utilisant l'encapsulation IP dans IP et envoyés à destination de l'adresse temporaire du mobile. Ce dernier traite ces paquets comme tout paquet disposant d'un en-tête de tunnelage. C'est-à-dire qu'il supprime l'en-tête externe et traite le paquet IP contenu comme s'il était arrivé directement.

L'agent mère ne fait pas suivre les paquets émis à destination de l'adresse lien local du mobile. Ceux-ci sont détruits et un message ICMP annonçant l'indisponibilité du mobile est envoyé à la source, sauf dans le cas du multicast ou le paquet est silencieusement écarté.

f) Information des correspondants

En acquittant la mise à jour d'association l'agent mère informe le mobile qu'il possède une association en règle et ce dernier peut informer ses correspondants. Pour cela il effectue la procédure RR (Return Routability Procedure - Procédure de test de Routage Retour) qui sera vue plus loin puis la mise à jour d'association. Il doit le faire pour tous les correspondants qui sont dans la liste des associations qu'il maintient.

g) Gestion de l'adresse mère

La validité de l'adresse mère, comme celle des autres adresses IPv6, est limitée dans le temps. La limite vient de la durée de validité annoncée dans l'annonce de routeur contenant le préfixe. Lorsqu'une adresse mère approche de sa date de péremption, le mobile ne peut pas envoyer tout simplement de sollicitation de routeur s'il n'est pas dans le réseau mère. Dans ce cas il émet un message appelé "sollicitation de préfixe mobile", directement vers l'agent mère. Ce message est semblable à une sollicitation de routeur, mais il contient l'option d'en-tête destination home address. Il doit être protégé par IPsec comme la plupart des échanges entre le mobile et l'agent mère. Ce dernier répond à la sollicitation par une annonce de préfixe mobile ressemblant à une annonce de routeur et dont les éléments seront traités par le mobile comme si l'annonce avait été reçue sur le réseau mère. En particulier le préfixe du réseau mère permet de mettre à jour la durée de validité de l'adresse mère.

Ce mécanisme permet de supporter la renumérotation du réseau mère. En effet, lorsque le mobile reçoit un nouveau préfixe, il peut configurer une nouvelle adresse mère en utilisant les mécanismes habituels d'auto configuration sans état.

h) Retour dans le réseau mère

Lorsqu'il détecte qu'il est de retour dans le réseau mère, le mobile doit en informer l'agent mère pour que ce dernier cesse de faire suivre les paquets à l'ancienne localisation du mobile. Il utilise pour détecter son retour dans le réseau mère une annonce de routeur contenant le préfixe de sa home address.

Pour envoyer la mise à jour d'association à l'agent mère, le mobile doit connaître l'adresse de niveau 2 de l'agent mère ce qui peut être déduit de l'annonce de routeur. Toutefois, il peut y avoir plusieurs agents mère sur le réseau. Dans ce cas le mobile doit découvrir l'adresse de niveau 2 de l'agent mère sans utiliser sa home address puisque l'agent mère est configuré pour la défendre dans les procédures de détection d'adresse dupliquée. Il demande donc l'adresse de niveau 2 de l'agent mère en émettant une sollicitation de voisin avec comme adresse source l'adresse non définie (::). Par contre il doit utiliser la home address comme adresse source de la mise à jour d'association et être en mesure de recevoir l'acquittement qui sera transmis par l'agent mère à cette adresse. Il doit donc configurer préalablement son interface avec la home address sans effectuer de procédure de détection d'adresse dupliquée.

Dès que la procédure de mise à jour d'association est terminée le mobile peut diffuser une annonce de voisin indiquant qu'il reprend possession de sa home address à toutes les autres stations sur le réseau local. Le bit indiquant la sollicitation devra être à zéro, tandis que celui indiquant que toutes les stations doivent mettre à jour leur cache avec la nouvelle association sera mis à 1. Ce message sera émis plusieurs fois pour prévenir les pertes éventuelles sur le réseau local. Une fois cette procédure terminée il doit supprimer les associations maintenues par les correspondants pour toutes les associations qu'il maintient.

3) Les en-têtes de mobilité

A) Format général du paquet

Nous avons vu que les en-têtes de mobilité sont utilisés pour transporter la signalisation de la gestion des associations de mobilité entre le nœud mobile, son agent mère et le nœud correspondant.

Un en-tête de mobilité ne doit jamais être utilisé en même temps qu'un en-tête de routage de type 2, excepté dans le seul cas du transport d'un acquittement d'une demande de BU. Il ne doit jamais non plus être utilisé en même temps qu'une extension de destination, sauf dans certains cas de Binding Update avec l'agent mère ainsi qu'avec des nouds correspondants déjà identifiés.

Le format général d'un en-tête de mobilité est donné figure Format de l'extension de mobilité :

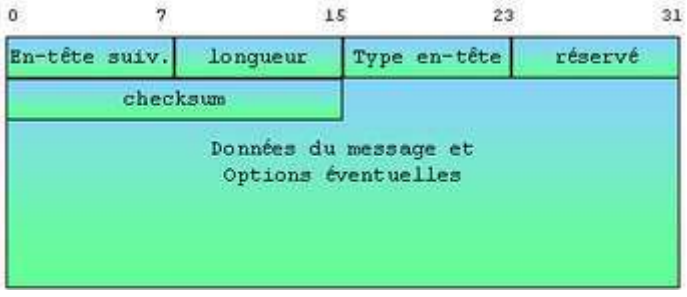


Figure 13-10 *Format de l'extension de mobilité*

- Le champ en-tête suivant est pris dans le même espace de numérotation que les en-têtes d'extension d'IPv6 (cf. Valeurs du champ en-tête suivant). Dans le cas de la signalisation de mobilité, il doit valoir 59 (pas d'en-tête suivant).
- Le champ longueur de l'en-tête, en octets, ne prend pas en compte les 8 premiers octets de l'en-tête.
- Le champ type d'en-tête décrit les messages de signalisation donné au tableau Type des en-têtes de mobilité.

Type des en-têtes de mobilité	
0	Demande de rafraîchissement émise par le noeud correspondant
1	Initialisation de test d'adresse mère (HoTI)
2	Initialisation de test d'adresse temporaire (CoTI)
3	Test d'adresse mère (HoT)
4	Test d'adresse temporaire (CoT)
5	Mise à jour d'association (émise depuis le noeud mobile)
6	Acquittement de mise à jour d'association
7	Erreur de mise à jour d'association

La structure et la longueur du message varient en fonction du numéro de l'en-tête. De plus, MIPv6 définit également des options de mobilité associées à ces messages. Comme la longueur des messages associés à chaque numéro d'en-tête est connue, la présence d'une option est déduite d'une longueur de l'en-tête plus grande que ce qui est suffisant pour le message. Elle se trouve forcément à la suite du message.

Comme tous les en-têtes IPv6, ces en-têtes de mobilité doivent être alignés sur des frontières de 8 octets. Des champs réservés seront éventuellement insérés pour respecter cette contrainte. Un nœud ne sachant pas interpréter une option doit l'ignorer. Actuellement MIPv6 ne définit d'option que pour les messages de BU (type 5) et leur acquittement (type 6).

a) Format des messages et options des différents types d'en-têtes

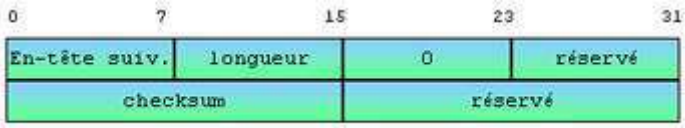


Figure 13-11 *Format de l'extension de mobilité rafraîchissement d'association*

La demande de rafraîchissement d'association ne requiert aucune information spécifique. Le message est vide, il n'y a pas d'option. (cf. figure Format de l'extension de mobilité rafraîchissement d'association).

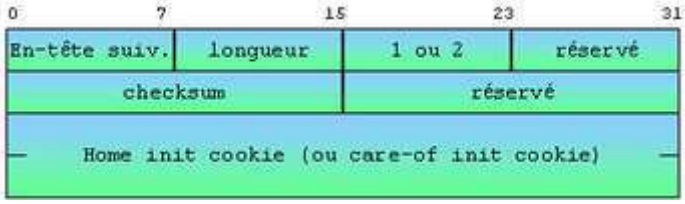


Figure 13-12 *Format de l'extension de mobilité HoTI ou CoTI*

Les messages HoTI, CoTI ne contiennent que le nombre aléatoire émis par le nœud mobile. Il ne contient pas d'option. (cf. figure Format de l'extension de mobilité HoTI ou CoTI).

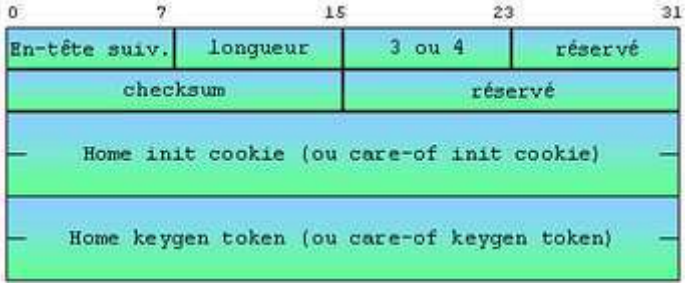


Figure 13-13 *Format de l'extension de mobilité HoT ou CoT*

Les messages HoT, CoT contiennent, l'index du nombre aléatoire (nonce) choisi par le nœud correspondant, le nombre aléatoire émis par le nœud mobile (*cookie*), (pour le test home address ou care-of address) et le jeton chiffré (`keygen token`) émis par le nœud correspondant. Il ne contient pas d'option (cf. figure Format de l'extension de mobilité HoT ou CoT).

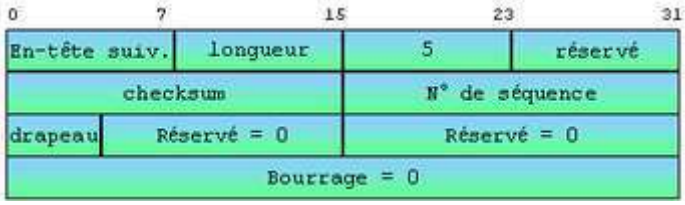


Figure 13-14 *Format de l'extension de mobilité mise à jour d'association*

Les messages de notification de mise à jour d'association, émis par le nœud mobile, peuvent contenir des options mobiles. Si elles ne sont pas présentes, le paquet doit se terminer par 4 octets de bourrage. (cf. figure Format de l'extension de mobilité mise à jour d'association)

Les options possibles sont :

- Les données d'autorisation de mise à jour d'association. L'option est obligatoire pour les mises à jour émises vers le nœud correspondant puisqu'elles ne sont pas protégées par IPsec.
- L'indice du "nonce" choisi par le nœud correspondant.
- Une "care-of address" alternative.

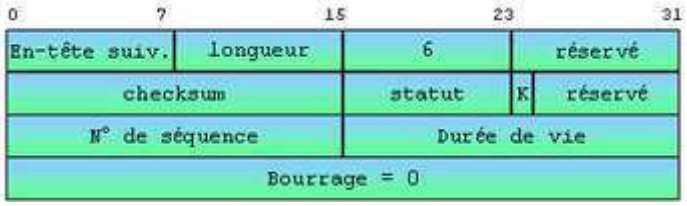


Figure 13-15 *Format de l'extension de mobilité acquittement de mise à jour d'association*

Les messages d'acquittement de mise à jour d'association peuvent contenir des options mobiles. Si elles ne sont pas présentes, le paquet doit être terminé par 4 octets de bourrage. (cf. figure Format de l'extension de mobilité acquittement de mise à jour d'association) :

- Une valeur du champ statut inférieure à 128 indique un acquittement, et une valeur supérieure un rejet. Le motif du rejet est codé par la valeur du statut.
- Le bit `K` = 1 indique que l'association de sécurité IPsec suivra les mouvements du noeud mobile. Le noeud correspondant doit le positionner à 0.
- Les options possibles sont :
  - Les données d'autorisation de mise à jour d'association.
  - Les informations de fréquence de rafraîchissement des associations.

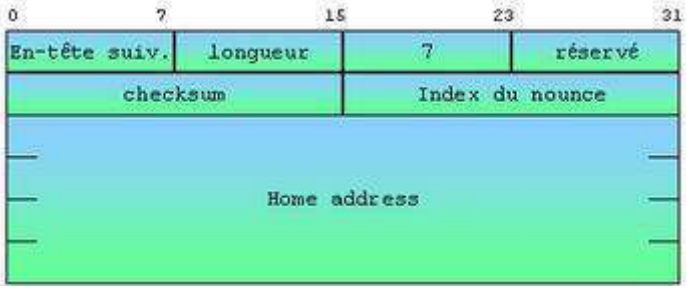


Figure 13-16 *Format de l'extension de mobilité message d'erreur*

Les messages d'erreur de mise à jour d'association contiennent un statut sur 8 bits codant l'erreur, ainsi que la home address de la mise à jour en erreur pour le cas où le nœud mobile aurait établi plusieurs associations avec le nœud correspondant (cf. figure Format de l'extension de mobilité message d'erreur).

4) Les risques induits par la mobilité et leur limitation

La mobilité doit satisfaire les deux contraintes de sécurité suivantes :

- L'introduction de la mobilité dans IP ne doit pas introduire de nouvelles vulnérabilités dans le réseau,
- Une communication dans un contexte mobile ne doit pas être plus risquée que dans un contexte fixe.

La sécurisation de MIPv6 est prévue dans le cadre standard de l'Internet où l'infrastructure de routage est réputée correcte, c'est-à-dire où un paquet destiné à un nœud A est effectivement acheminé vers ce noeud A. Cette sécurisation vise à obtenir un niveau de confiance des communications mobiles, égal à ou proche de celui offert aux communications fixes.

A) Les risques pour le nœud mobile

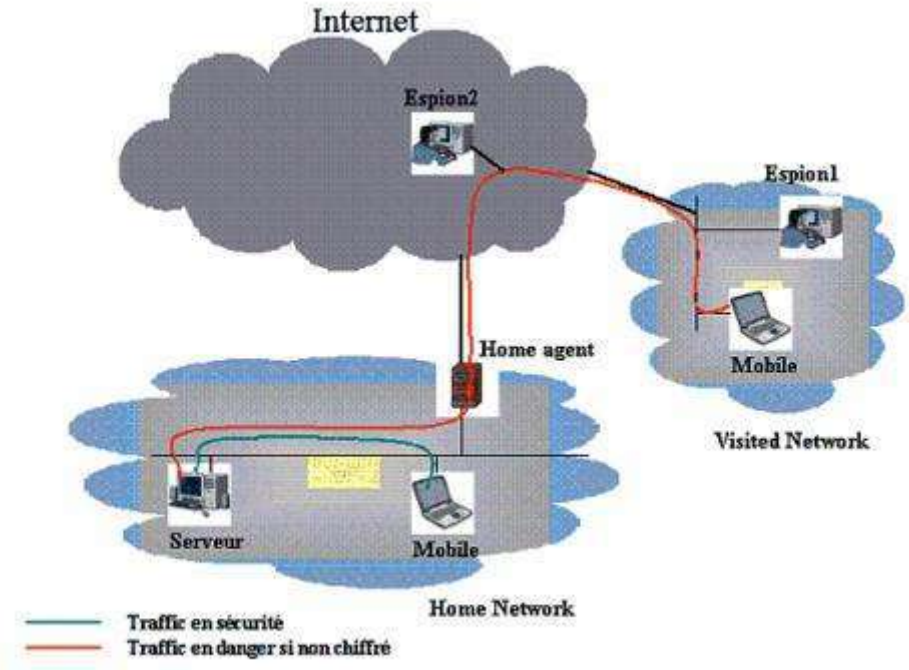


Figure 13-17 *Chiffrement nécessaire des données*

Un nœud dans son réseau mère (*home network* dans les figures) considère généralement son environnement comme amical, surtout lorsqu'il communique avec des correspondants également situés dans son réseau mère. En visite dans un autre réseau il doit se montrer plus circonspect. En particulier, il devra assurer la confidentialité des données transmises, par exemple en utilisant IPsec, afin d'éviter qu'elles ne soient épiées, au niveau du réseau visité, mais également sur le chemin entre le réseau visité et son réseau mère (cf. figure Chiffrement nécessaire des données).

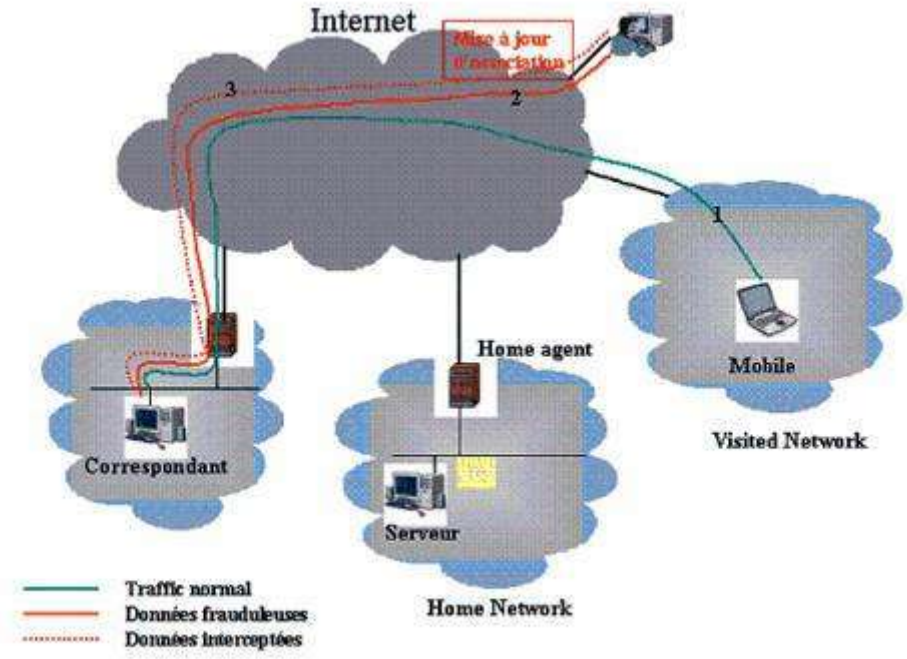


Figure 13-18 *Détournement de trafic*

Un nœud mobile peut également souffrir de déni de service si les mises à jour d'association avec son agent mère sont falsifiées. Un nœud hostile dans le réseau visité, ou partout ailleurs dans le réseau, pourrait envoyer une fausse demande de mise à jour d'association afin de détourner le trafic de son véritable destinataire. Il est donc important pour le nœud mobile de sécuriser ces mises à jour (cf. figure Détournement de trafic).

L'IETF a décidé d'utiliser IPsec pour la signalisation entre le mobile et son agent mère, spécifiée dans le RFC 3776.

B) Les risques pour les autres nœuds

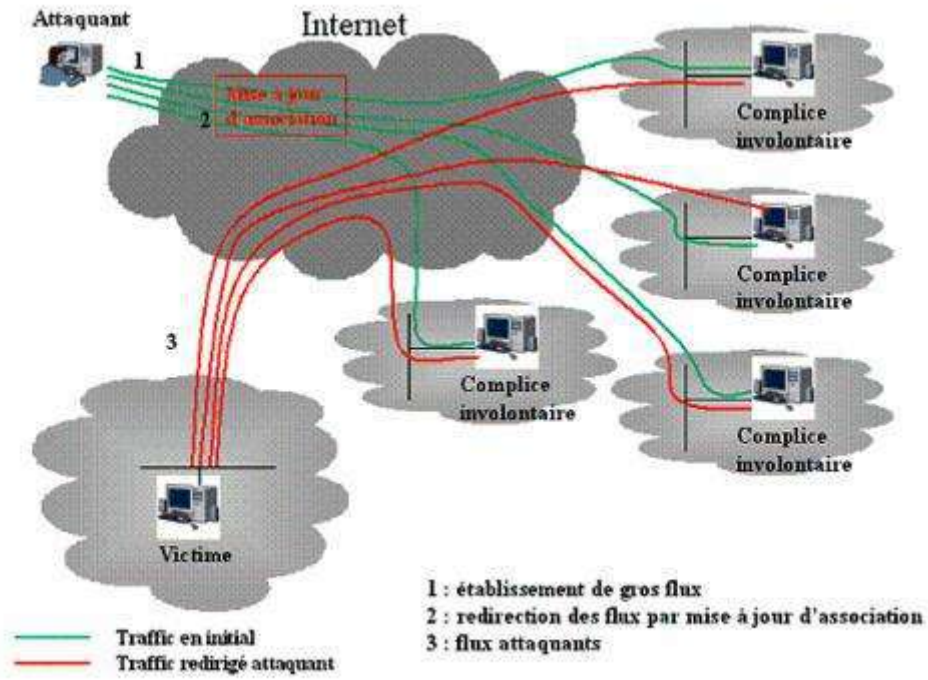


Figure 13-19 *Déni de service envers un nœud tiers*

Un nœud malveillant peut utiliser des messages de mise à jour d'association frauduleux pour détourner de ses clients légitimes les flux d'un serveur mettant en œuvre la mobilité. Ce cas est similaire au cas du détournement du trafic destiné à un nœud mobile. Un nœud malveillant peut également utiliser des nœuds mettant en œuvre la mobilité pour inonder un autre nœud, de trafic non sollicité, de façon à engorger ses liens de communication, ceci sans même que la victime ne mette en œuvre la mobilité (cf. figure Déni de service envers un nœud tiers).

On notera que ces risques sont exclusivement liés à la mise en œuvre de l'optimisation du routage. Afin de les diminuer jusqu'à un niveau acceptable, sans pénaliser les performances du protocole, MIPv6 prévoit la procédure de test de "routage retour" entre le nœud mobile et son correspondant. Cette procédure est décrite au paragraphe suivant et spécifiée dans le RFC 3775.

C) Sécurisation de la signalisation avec les nœuds correspondants

a) La procédure de test de routage retour

Les mises à jour d'association étant fréquentes, il est important que cette procédure soit la plus légère possible. Un nœud mobile et un nœud correspondant ne se connaissent pas à priori. Ils ne partagent donc pas de secret susceptible de chiffrer leurs échanges lors des différentes mises à jour d'association nécessaires pendant toute la durée de la communication. L'utilisation d'IPsec et d'une procédure d'échange sécurisé des clefs aurait été trop lourde. La procédure choisie a pour premier but de générer ce secret partagé.

Afin d'éviter l'attaque en déni de service à l'encontre d'un nœud tiers, elle garantit au nœud correspondant que, pour une certaine adresse temporaire et pour une certaine adresse mère, il y a effectivement un nœud mobile prêt à recevoir un paquet.

La procédure est conçue de façon à ce que le nœud correspondant ne puisse pas subir lui-même une attaque en déni de service par la simple exécution répétée de la procédure. A cette fin, elle consomme peu de ressources de calcul, et les ressources mémoires nécessaires ne dépendent pas du nombre de demande d'association. Enfin, pour ne pas surcharger le réseau, le nœud correspondant n'émet pas plus de paquets qu'il n'en reçoit.

La procédure est constituée de deux phases préliminaires, dont l'une teste la home address et l'autre teste la care-of address. Ensuite toute demande de mise à jour, ou de destruction d'association sera assujettie à l'exécution correcte des ces deux phases préliminaires.

Les deux phases sont menées parallèlement l'une de l'autre, à l'initiative du nœud mobile. Le nœud correspondant répond à leurs requêtes indépendamment l'une de l'autre. Il demeure sans état jusqu'à ce qu'une association soit établie. Les messages doivent donc être autosuffisants pour que la procédure puisse se dérouler.

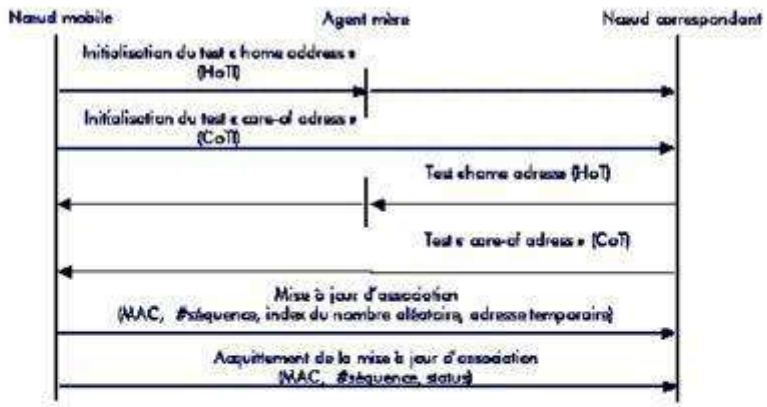


Figure 13-20 *Routage retour*

La procédure (cf. figure Routage retour) nécessite que le nœud mobile dispose d'un ensemble de nombres aléatoires secrets (*nonces*), tels que l'index d'un de ces nombres suffise à le retrouver dans cet ensemble. Elle nécessite également que le nœud correspondant dispose d'une clef secrète notée `Kcn`.

Un message HoTI, émis depuis la home address du nœud mobile vers le nœud correspondant (donc via l'agent mère), contient une valeur aléatoire sur 64 bits, le *home init cookie* identifiant cette home address.

Un message HoT, émis par le nœud correspondant en réponse au message HoTI à destination de la home address du mobile, donc toujours via l'agent mère contient trois valeurs :

- le *home init cookie* obtenu dans le message HoTI,
- l'index sur 16 bits d'un nonce, choisi par le nœud correspondant,
- le home keygen token, calculé par le nœud correspondant :

```
premiers (64, HMAC_SHA1 (Kcn, (home address | nonce | 0 )))
```

Un message CoTI, émis depuis la care-of address du nœud mobile, directement vers le nœud correspondant, contient une seconde valeur aléatoire sur 64 bits, le *care-of init cookie* identifiant cette care-of address.

Un message CoT, émis par le nœud correspondant en réponse au message CoTI du nœud mobile, directement vers le nœud mobile contient trois valeurs :

- le *care-of init cookie* obtenu dans le message CoTI,
- l'index sur 16 bits d'un second nonce, choisi par le nœud correspondant,
- le care-of keygen token, calculé par le nœud correspondant :

```
premiers (64, HMAC_SHA1 (Kcn, (care-of address| nonce | 1 )))
```

Lorsque le nœud mobile a reçu les messages HoT et CoT, la procédure de test du routage retour est terminée. Arrivé à ce point, le nœud mobile calcule `Kbm`, la clef de gestion des associations (*key binding management*) telle que :

```
Kbm = SHA1 ( "home keygen token" | "care-of keygen token")
```

pour la mise à jour d'une association, ou pour sa destruction.

```
Kbm = SHA1 ( "home keygen token")
```

Pour demander une nouvelle association, le nœud mobile envoie, depuis sa care-of address courante, à destination du nœud correspondant, une demande d'association contenant cinq informations :

- Sa home address
- Un numéro de séquence d'association
- Les deux index des home et care-of nonces
- Une valeur chiffrée
- `Premier( 96, HMAC_SHA1(Kbm , ("home address" | adresse du nœud correspondant | donnée de la mise jour d'association)))`

On notera que puisque les indexes des nombres aléatoires secrets sont fournis par le nœud mobile, le nœud correspondant peut recalculer `Kbm`. `Kbm` est donc bien une clef partagée utilisable dans une procédure `HMAC_SHA1` pour vérifier la légalité d'une demande de mise à jour d'association.

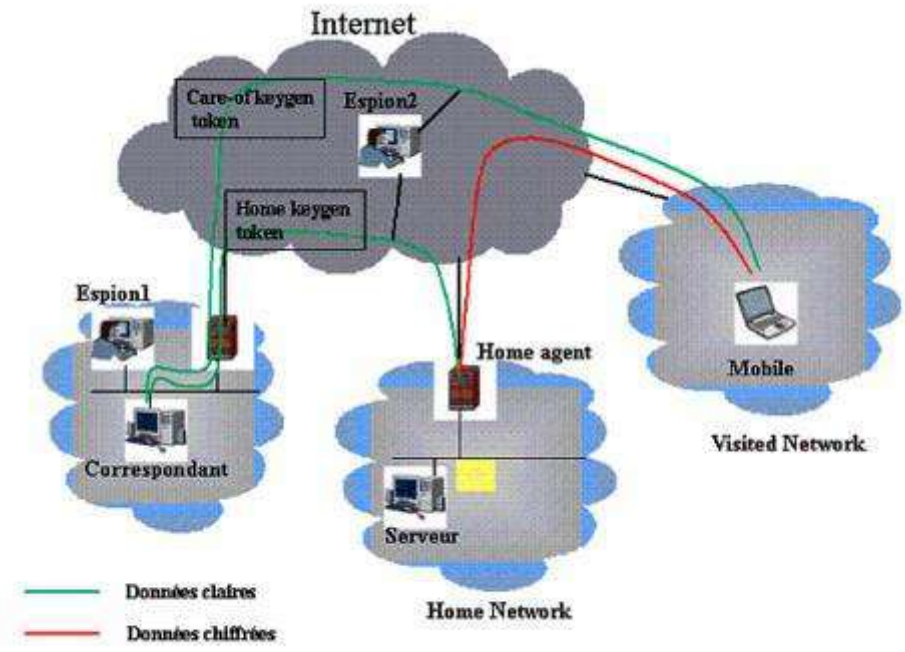


Figure 13-21 *Attaque contre le routage retour*

La correction de la procédure repose sur l'hypothèse qu'aucun intrus ne peut écouter à la fois les messages home test et care-of test ni connaître `Kbm`. Ces messages utilisant deux chemins différents pour joindre le nœud correspondant (cf. figure Attaque contre le routage retour), il faudrait donc que l'intrus se trouve dans le réseau du nœud correspondant. Dans ce cas, la mobilité n'introduit pas plus de risque que IP fixe lui-même.

Par contre, si un nœud malveillant obtient les deux home et care-of keygen tokens, il pourra par la suite envoyer de fausses demandes de mise à jour d'association. Pour cela, ce nœud doit écouter des données circulant en clair sur les 2 chemins conduisant du nœud mobile au nœud correspondant, directement et via l'agent mère. La probabilité que cette écoute soit possible augmente si le nœud malveillant est proche du nœud correspondant. L'écoute est définitivement possible lorsque l'on est connecté au réseau du nœud correspondant.

Les message HoTI, CoTI, HoT et CoT sont transportés dans des en-têtes d'extension de mobilité (numéro de protocole 135). Chacun possède un numéro de type d'en-tête de message particulier (respectivement 1,2,3,4).

La procédure conduit au partage d'un secret entre les nœuds mobile et correspondant. Il est nécessaire de rafraîchir régulièrement ce secret. Le rafraîchissement est laissé à l'initiative du nœud correspondant. Il est mise en œuvre en expirant la validité du nonce utilisé dans la clef `Kbm`. Une demande de modification d'association arrivant avec un nonce expiré sera refusée via le message d'acquittement. Le nœud mobile relancera alors la procédure pour obtenir une nouvelle `Kbm` basée sur un nonce valide.

La clef `Kcn` doit elle même être régulièrement régénérée. Elle le sera en particulier à chaque redémarrage du nœud correspondant et préférablement lors de chaque régénération de nonce. Il est en effet nécessaire que chaque nonce soit associé à la bonne `Kcn` dans la vérification de la clef `Kbm` d'une demande de mise à jour d'association.

La vérification par le nœud correspondant d'une clef `Kbm` s'effectue en vérifiant que :

- Les nonces HoA et CoA ne sont pas expirés,
- Le re-calcul de `Kbm`, sur la base des indices des nonces et de la `Kcn` associée, est cohérent avec la valeur `Kbm` contenue dans la demande d'association.

Un message d'erreur est envoyé en cas d'expiration d'une au moins des nonces. Aucun message d'erreur n'est émis dans le cas ou le re-calcul de la `Kbm` échoue. Dans le cas de nonce expiré, il est nécessaire de procéder au re-calcul sur la base d'une éventuelle ancienne valeur de `Kcn` pour n'envoyer de message d'expiration que si la `Kbm` est valide par rapport à l'ancienne `Kcn`.

Seul le nœud mobile maintient l'état des données de sécurité de chaque association. Les informations home init cookie et care-of init cookie peuvent être supprimées dès réceptions des nonces et keygen tokens. Les demandes de création d'association sont à l'initiative du nœud mobile. Il n'est donc pas sujet à une attaque en déni de service par consommation excessive de ressources mémoire.

Le nœud correspondant maintient un état de sécurité indépendant du nombre d'associations en cours. Il n'est donc pas sujet non plus à une attaque en déni de service par consommation excessive de ressources mémoire.

D) Protection de la signalisation par le protocole IPsec

a) Utilisation du protocole IPsec dans MIPv6

Le protocole IPsec nécessite la connaissance réciproque des clefs entre les correspondants. Le nœud mobile et son agent mère appartenant à la même organisation, il est raisonnable d'admettre qu'ils auront pu échanger leurs clefs en toute sécurité sans la mise en place d'une lourde infrastructure de gestion de clefs. L'utilisation d'IPsec entre le nœud mobile et son agent mère est donc tout à fait adaptée. Elle n'utilise que ESP en mode tunnel ou transport. La position des en-tête ESP sera choisie de façon à protéger également les informations de routage sans avoir recours à IPsec en mode AH, d'où une simplification de l'implémentation de la mobilité.

Le protocole IPsec est utilisé dans trois types de communications entre le nœud mobile et son agent mère :

- Les messages de la procédure de routage retour,
- La mise à jour des association de mobilité et leur acquittement,
- L'encapsulation du flux des données entre le mobile et son nœud correspondant sur le tronçon nœud mobile-agent mère. (Ce qui n'est somme toute qu'un cas d'utilisation "standard" d'IPsec dans IPv6 entre un nœud (le nœud mobile) et une passerelle de sécurité (l'agent mère).)

L'efficacité d'une procédure de sécurité repose largement sur la qualité des politiques mises en œuvre. Cet ouvrage n'étant pas un ouvrage sur la sécurité il n'est pas possible de détailler les politiques recommandées pour la gestion de la mobilité (voir RFC 3776).

b) IPsec dans les mises à jour d'association entre le nœud mobile et son agent mère

L'essentiel du problème consiste à garantir l'origine de la demande de mise à jour ainsi que sa valeur. ESP sera utilisé en mode transport. C'est-à-dire que le packet ESP ne contient qu'une signature des données qu'il encapsule (cf. figure IPsec en mode tranport pour les mises à jour d'association de mobilité).

Lors d'une demande de mise à jour, la care-of address est répétée dans l'option *alternate care-of address* de la demande de mise à jour. Comme sa valeur est certifiée par ESP, l'agent mère considèrera cette adresse plutôt que l'adresse source de l'en-tête IPv6. L'adresse de l'agent mère de l'en-tête n'est pas protégée par ESP, elle est garantie par les règles d'utilisation de l'adresse de l'agent mère lors de l'établissement de l'association de sécurité entre le nœud mobile et l'agent mère.

Enfin, lorsqu'un nœud mobile est dans son réseau mère, l'en-tête option destination ainsi que l'en-tête de routage peuvent être omises puisque le nœud mobile peut utiliser son adresse mère. Ce sera le cas notamment quand le nœud mobile de retour dans son réseau mère, demande la destruction de son association.

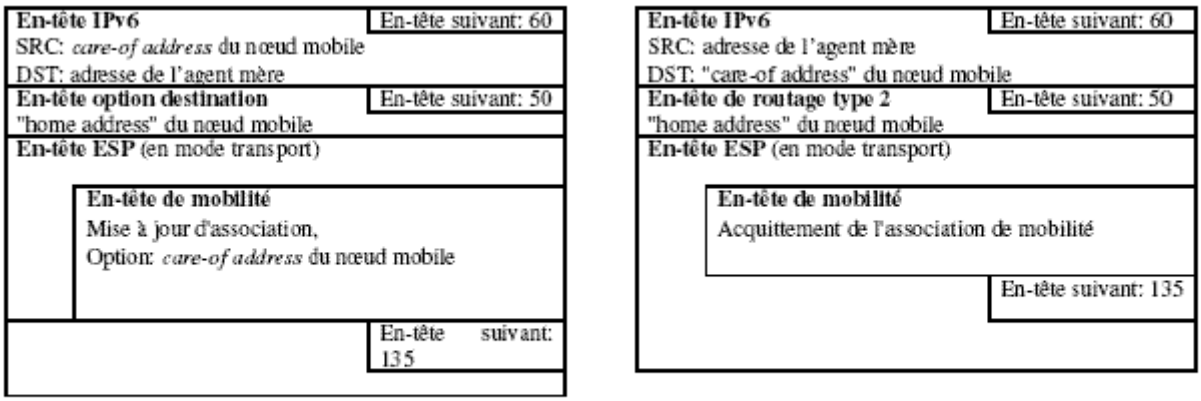


Figure 13-22 . IPsec en mode tranport pour les mises à jour d'association de mobilité

c) IPsec dans la procédure de "retour de routage" via l'agent mère

L'essentiel du problème est d'éviter qu'un nœud malveillant puisse écouter les échanges conduisant au nœud mobile à calculer la clef `Kbm` avec le nœud correspondant. Les informations *home init cookie* et *home keygen token* doivent donc être chiffrées. ESP est ici utilisé en mode tunnel. L'algorithme de chiffrement utilisé dans l'association ne doit donc pas être nul (cf. figure IPsec en mode tunnel pour la procédure de retour de routage).

On notera également que lorsque le nœud mobile change d'adresse temporaire, l'agent mère devra mettre à jour l'association de sécurité pour prendre en compte cette nouvelle adresse destination du nœud mobile.

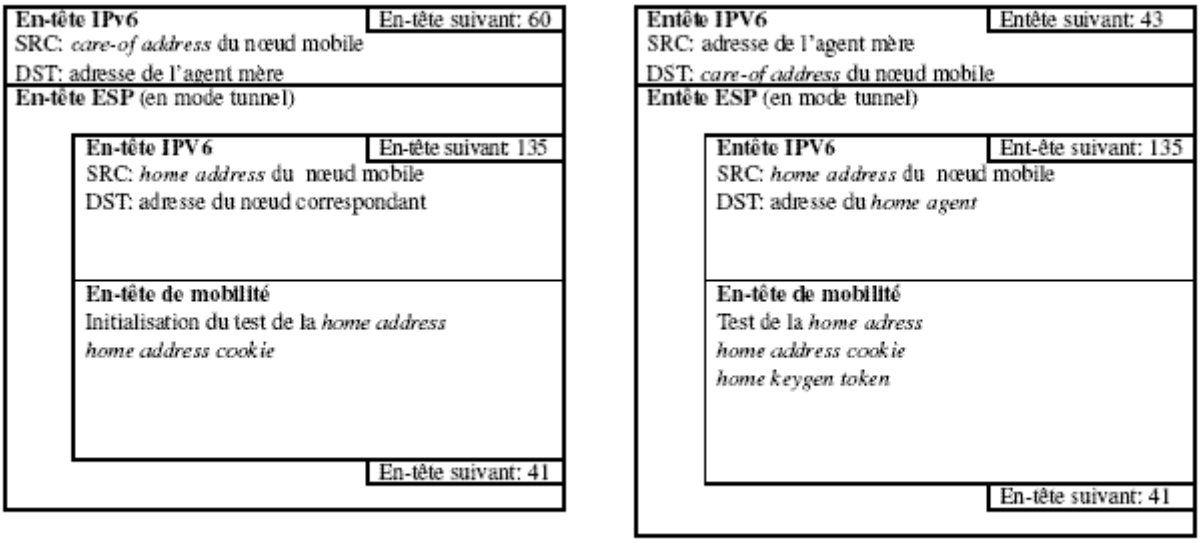


Figure 13-23 . IPsec en mode tunnel pour la procédure de retour de routage

5) Amélioration de la gestion de la mobilité IPv6

La gestion de la mobilité telle qu'elle a été définie pour IPv6 a l'avantage de pouvoir fonctionner dans de nombreux cas de figure sans supposer la collaboration des réseaux visités, ni même celle des nœuds correspondants lorsque le tunnelage inverse est utilisé. Par contre elle entraîne souvent des délais inacceptables pour la plupart des applications (plusieurs secondes) et n'offre aucun moyen à l'opérateur du réseau visité de contrôler la mobilité des terminaux. Il existe plusieurs manières d'améliorer le délai de handover et la quantité de signalisation induite dans le réseau.

A) Les approches de micro-mobilité

Plusieurs approches permettent une gestion plus fine des handovers à l'intérieur d'un domaine et réduisent la signalisation dans le réseau. Elles ont en commun de rendre les déplacements des mobiles à l'intérieur d'un domaine, dit de micro-mobilité, transparent au HA et aux correspondants. La mobilité entre les domaines de micro-mobilité est alors gérée normalement par Mobile IPv6 et est alors appelée macro-mobilité.

Dans les différentes approches de micro-mobilité, le mobile acquiert une adresse temporaire (CoA) lorsqu'il entre dans un domaine de micro-mobilité et enregistre cette adresse auprès de l'agent mère et des correspondants. À l'intérieur du domaine, les paquets adressés à cette CoA sont dirigés vers le point d'attachement du mobile suivant deux grandes techniques :

- La première consiste à modifier le routage à l'intérieur du domaine. Une route spécifique correspondant au mobile est maintenue jusqu'à la passerelle d'entrée du domaine. Cette solution n'était pas envisageable à l'échelle de l'Internet mais offre de bons résultats dans un environnement contrôlé. Cellular IP, qui est un exemple de protocole basé sur cette approche, apporte en outre des fonctionnalités utiles aux terminaux mobiles comme le support d'un mode veille.
- La seconde technique consiste à reproduire le fonctionnement de Mobile IPv6 sur un ou plusieurs niveaux de hiérarchie, chacun des niveaux masquant la mobilité aux niveaux supérieurs de la hiérarchie. HMIPv6 est un protocole fonctionnant sur ce modèle et développé à l'origine par l'INRIA. Une solution permettant d'introduire un contrôle par le réseau, NCHMIPv6, est développé par France Télécom R&D.

B) L'amélioration du handover : Fast Mobile IPv6

Les principaux problèmes de performance de Mobile IPv6 viennent de ce que le délai de latence du handover est trop important pour de nombreuses applications, il entraîne à la fois des interruptions de communication et des pertes de paquets perceptibles pour les utilisateurs. Cette latence est composée de plusieurs délais :

- le délai de handover de niveau 2 qui est irréductible si on se cantonne à IP ;
- le délai de découverte du changement de réseau et d'acquisition d'une nouvelle adresse temporaire ;
- le délai d'enregistrement auprès de l'agent mère et des correspondants.

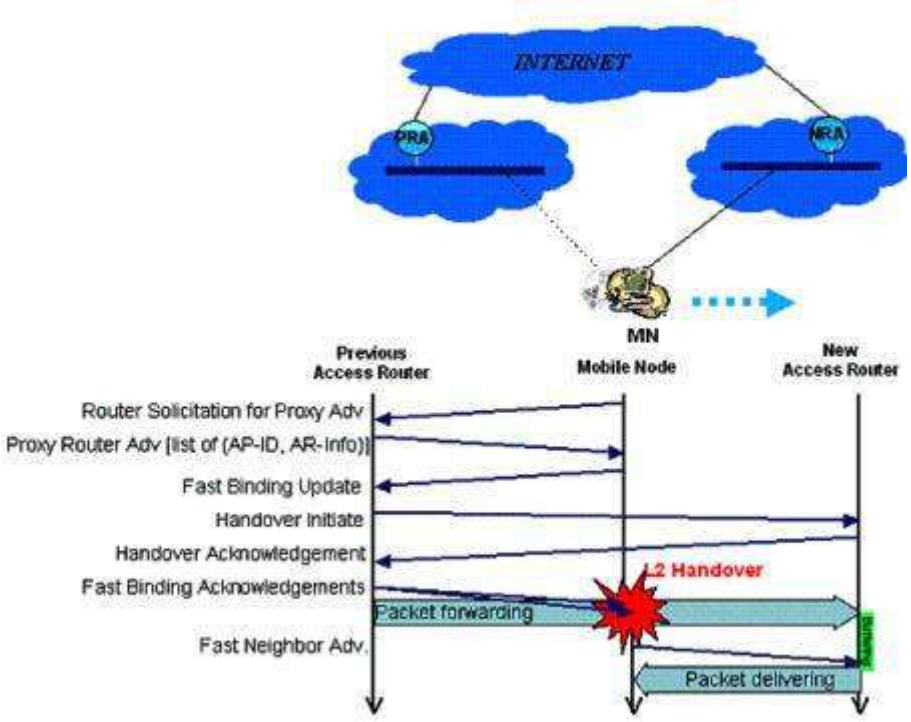


Figure 13-24 *Fonctionnement de Fast Handover pour Mobile IPv6*

Les approches de micro-mobilité permettent de réduire le temps d'enregistrement puisqu'il n'est plus nécessaire de s'enregistrer auprès de l'agent mère et des correspondants dans la mesure ou la mobilité locale leur est cachée. Le temps de handover de niveau 2 doit être traité spécifiquement pour chaque technologie. Plusieurs solutions ont été proposées pour réduire le temps nécessaire à la détection du changement de réseau et à l'acquisition d'une nouvelle adresse temporaire (NCoA).

Parmi les solutions proposées celle de handover rapide pour Mobile IPv6 est déjà bien aboutie (cf. figure Fonctionnement de Fast Handover pour Mobile IPv6). Il s'agit de réduire le délai nécessaire à l'acquisition d'une nouvelle adresse temporaire lors d'un changement de réseau d'accès et donc d'un changement de routeur d'accès et de limiter la perte des paquets en retransmettant les paquets entre l'ancien et le nouveau routeur d'accès (*Previous Acces Router* : PAR et *New Access Router* : NAR).

Le principe de fonctionnement est le suivant : le mobile acquiert une connaissance des réseaux voisins de son réseau d'accès actuel en interrogeant son routeur d'accès (PAR) à l'aide d'une sollicitation de routeur

demandant à ce dernier d'annoncer les voisins qu'il connaît. En retour, le mobile reçoit une liste des points d'accès voisins (un identifiant de niveau 2 par exemple) et les informations concernant les routeurs d'accès associés (adresse IP, préfixe de réseau, ...).

Lorsque le mobile voit la qualité de son signal diminuer, il tente de trouver dans son voisinage d'autres points d'accès et sélectionne celui qui lui offre la meilleure qualité. Cette partie est spécifique à chaque technologie de niveau 2 (par exemple le WiFi). Il obtient un identifiant du point d'accès et peut construire une nouvelle CoA (NCoA) grâce aux informations obtenues en réponse à sa sollicitation.

Le mobile informe alors sont routeur d'accès courant (PAR) qu'il va effectuer un changement de réseau d'accès en émettant une mise à jour d'association rapide : *Fast Binding Update* (FBU). Ce FBU contient la nouvelle CoA du mobile et l'identité du nouveau routeur d'accès (NAR). Le PAR informe alors le NAR qu'un handover va avoir lieu (*Handover Initiate*) et lui transmet la NCoA pour qu'il puisse en vérifier la validité. Cette procédure permet au mobile d'éviter d'effectuer une détection de duplication d'adresse lorsqu'il effectuera effectivement le handover de niveau 2.

A réception de l'acquittement du NAR (HAck), le PAR informe le mobile en acquittant la mise à jour d'association rapide (FBU) et commence à faire suivre les paquets destiné à l'ancienne CoA dans un tunnel vers la nouvelle CoA. Le nouveau routeur d'accès (NAR) stocke les messages en attendant l'arrivée du mobile.

Lorsque le mobile effectue le handover de niveau 2, il émet instantanément une annonce de voisin rapide (*Fast Neighbor Announcement* : FNA) pour informer le NAR de sa présence et ce dernier peut retransmettre les paquets en cours de transit. Il n'a plus alors qu'à effectuer la mise à jour d'association vers le HA et les correspondants. C'est seulement lorsque cette procédure sera terminée que la nouvelle CoA sera utilisée directement par les correspondants et par le HA.

La procédure est au final assez complexe. Elle suppose une coopération assez forte entre le niveau 2 et le niveau IP pour la détection du voisinage et le contrôle du handover de niveau 2. Enfin, elle fait l'hypothèse que les routeurs d'accès communiquent entre eux et ont une connaissance des réseaux d'accès voisins. Cela ne peut donc être mis en œuvre que dans un domaine restreint au même titre que la micro-mobilité.

6) Support de la Mobilité des Réseaux

A) Les réseaux mobiles

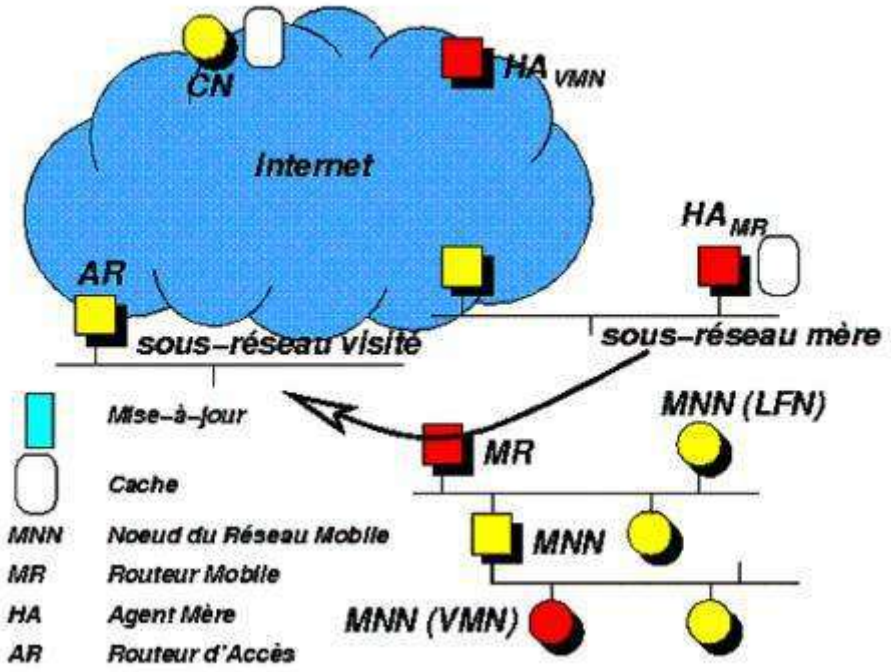


Figure 13-25 *Terminologie pour les réseaux mobiles*

Un réseau mobile est défini comme un ensemble de sous-réseaux connectés à l'Internet par l'intermédiaire d'un ou plusieurs routeurs mobiles (MR : *Mobile Router*) qui changent leurs points d'ancrage (AR : *Access Router*) à l'Internet. Les termes MNNs (nœud du réseau mobile) et CN (nœud correspondant) désignent respectivement tout nœud localisé à l'intérieur du réseau mobile et tout nœud communiquant avec un ou plusieurs MNNs. Les interfaces d'un MR connectées sur un sous-réseau mère ou un sous-réseau étranger sont nommées interfaces externes tandis que toutes les autres interfaces sont nommées interfaces internes. Toute interface devant obtenir une adresse sur le lien auquel elle se raccroche, le préfixe de l'interface externe sera le même que celui du sous-réseau mère ou celui du sous-réseau étranger tandis que celui de l'interface interne et de tous les MNNs sera le même que celui du réseau mobile, nommé MNP (*Mobile Network Prefix*). Ces termes sont illustrés sur la figure Terminologie pour les réseaux mobiles montrant un réseau mobile se déplaçant de son sous-réseau mère vers un autre sous-réseau.

B) Les Usages

Les usages possibles des réseaux mobiles sont très variés. Ceux-ci incluent entre autres les réseaux de capteurs déployés dans les véhicules (avions, trains, bateaux, voitures) qui ont besoin d'interagir avec des serveurs dans l'Internet, par exemple pour assurer la transmission de données nécessaires à la navigation; les réseaux d'accès déployés dans les transports publics (bus, trains et taxis) offrant une borne d'accès à l'Internet aux passagers; ou encore les réseaux personnels (*Personal Area Networks* : PANs) constitués d'un ensemble d'appareils électroniques de petite taille (cardio-fréquence mètre, montre, téléphone cellulaire, assistant personnel, appareil photo numérique, etc) portés par les personnes.

C) De Mobile IP à NEMO

La problématique des réseaux mobiles a fait sommairement son apparition à l'IETF à plusieurs reprises avant de véritablement prendre son envol à partir de 2000.

Les concepteurs de MIPv6 proposent de gérer la mobilité des réseaux de manière similaire à celle des stations, mais ceci est présenté de manière très succincte, en partant de l'observation qu'un réseau mobile n'est rien d'autre qu'un réseau rattaché à un routeur mobile, c'est-à-dire un nœud comme une autre. A chacun de ses déplacements, il suffirait donc au MR d'obtenir une adresse temporaire MR*CoA* et de l'enregistrer auprès de son HA comme dans le cas d'une station mobile. Cette analyse n'a cependant pas été suffisamment poussée par leurs auteurs pour considérer les caractéristiques et les problèmes spécifiques à la mobilité des réseaux. De nombreux problèmes subsistent donc.

Il s'est en effet avéré que MIPv6 n'est pas adapté au support de la mobilité des réseaux comme cela a été démontré dans [Ernst01f-fr] et [Ernst03f]. Le document qui en a été extrait pour être soumis au groupe de travail Mobile IP en août 2000 a, en particulier, mis en avant les insuffisances de MIPv6 pour supporter les stations situées derrière le routeur mobile. D'une part, la spécification ne permet pas au HA de rediriger les paquets destinés aux nœuds situés derrière le MR, et d'autre part le mécanisme d'optimisation du routage est inadéquat. Le support des réseaux mobiles nécessite donc une solution spécifique, mais dont le concept n'est pas forcément très éloigné.

La communauté IETF a donc pris conscience du besoin de traiter le cas des réseaux mobiles comme un sujet à part entière. Pour éviter les interférences avec le développement de Mobile IP, elle a créé, en octobre 2002, un nouveau groupe de travail nommé NEMO (NEtwork MObility). Les contours de son champ de travail ont été difficiles à établir notamment à cause de la confusion souvent faite entre réseaux mobiles et réseaux ad-hoc.

D) Le groupe de travail NEMO de l'IETF

Le groupe de travail NEMO a décidé lors de sa création d'aborder le problème en deux étapes afin de produire une solution déployable rapidement :

- Support de Base (Basic Support) : Dans un premier temps, le groupe a standardisé dans le RFC 3963 une solution simple permettant de maintenir les sessions pour l'ensemble des MNNs, sans optimisation de routage.
- Support Étendu (Extended Support): Dans un second temps, le groupe se doit d'étudier les problèmes d'optimisation, en particulier l'optimisation du routage. Un document de synthèse décrivant la problématique et les approches potentielles (ne reposant pas nécessairement sur le modèle MIPv6) sera publié. A l'issue de ce document, le groupe devra décider s'il continue ses travaux dans le but de standardiser une ou plusieurs solutions pour l'optimisation du routage, ou déclarer sa fermeture. Dans le premier cas, la charte devra être préalablement redéfinie, à la vue des conclusions du document de synthèse.

Le lecteur intéressé se référera sur le site web du groupe de travail pour retrouver l'ensemble des documents en cours l'étude à l'IETF.

E) NEMO support de base

La solution pour le support de base est définie sur le modèle MIPv6 (protocole de gestion de la mobilité des stations) selon des règles préalablement édictées par le groupe de travail dans un document dressant la liste des fonctions requises [Ernst-id]. La règle fondamentale est de ne pas imposer de modifications sur les nœuds localisés derrière le routeur mobile (MNNs) et de maintenir les sessions, sans optimisation de routage.

Cette solution permet la seule redirection des paquets destinés aux MNNs vers la position courante du MR. Elle consiste à établir un tunnel bidirectionnel entre le HA et le MR. Le principe de base est que tous les nœuds du réseau mobile partagent le (ou les) même préfixe d'adresse IP (MNP).

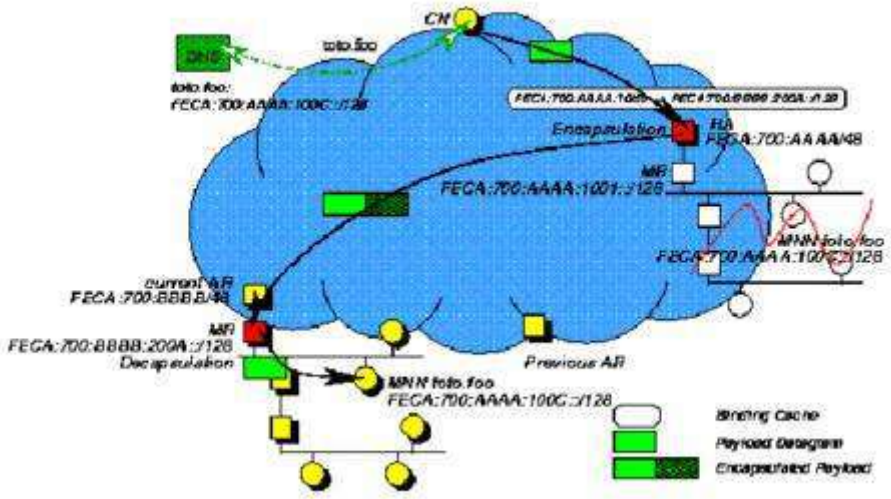


Figure 13-26 *Support de base de NEMO*

Comme dans MIPv6, le support de base gère le problème de la mobilité en allouant deux adresses à chaque interface externe du MR (ou des MRs dans le cas où il y en aurait plusieurs). La première MR*HoA* est une adresse permanente qui identifie le MR dans le sous-réseau mère. Elle identifie soit l'interface externe et a pour préfixe celui du sous-réseau mère, soit l'interface interne du MR [RFC 3810], et elle a pour préfixe MNP comme chacun des MNNs du même réseau mobile. La seconde (MR*CoA*) est temporaire (CoA) et est obtenue dans le sous-réseau visité sur lequel l'interface externe du MR prend ancrage. Le protocole établit ainsi une relation entre le préfixe MNP utilisé comme identificateur, et l'adresse temporaire MR*CoA*, utilisée pour le routage. Seuls les MRs qui changent leur point d'ancrage obtiennent cette nouvelle adresse, les autres MNNs conservent leur seule adresse MNN*MNP* ; la gestion de la mobilité leur est ainsi transparente (cf. figure Support de base de NEMO).

Le MR fait ensuite parvenir l'adresse temporaire primaire MR*CoA* au moyen d'un message de mise-à-jour des préfixes (PBU) à son agent mère (HA). Les PBUs sont des paquets spéciaux contenant une en-tête d'extension Mobility Header. Lorsque HA reçoit un PBU valide (i.e. obéissant aux tests de conformité liés à la sécurité, particulièrement l'authentification de l'émetteur par son destinataire), l'entrée correspondante au MNP est ajoutée ou mise à jour dans son cache (Binding Cache). Elle instruit le HA d'encapsuler les paquets à destination des stations résidants dans le réseau mobile vers la destination effective du réseau mobile (i.e. MRc) dans la mesure où le préfixe de l'adresse de destination correspond à celui enregistré dans le cache.

Lors d'une communication entre un MNN et un CN, le CN n'a pas connaissance de l'adresse de routage temporaire MR*CoA*. Les paquets sont donc envoyés normalement vers l'adresse MNN*MNP* du MNN et routés jusqu'au sous-réseau ayant pour préfixe MNP. Ils parviennent ainsi sur le sous-réseau mère du MR. Les paquets y sont interceptés par le HA puis encapsulés vers MR*CoA* comme cela est montré sur la figure Support de base de NEMO. A la réception d'un paquet encapsulé, le MR le décapsule et le transmet sur son interface interne. Le paquet que reçoit le MNN ne contient donc plus MR*CoA* ; l'opération lui est ainsi transparente. Dans le sens inverse, les paquets sont également encapsulés du MR à son HA.

Le groupe a récemment décidé de débattre de la question de la multi-domiciliation. Un document commun a été publié [NPE-id] dans le but d'analyser le problème et de décider des configurations qui devront être supportées dans le support de base. Le groupe de travail doit également produire quelques documents annexes au protocole NEMO Basic Support, en particulier, la délégation des préfixes, une MIB, et les usages [Thubert-id].

7) Un exemple de mise en œuvre de la pile LIVSIX

Cette section décrit un cas pratique de mobilité IPv6 telle qu'elle est mise en oeuvre dans la souche LIVSIX. Cette souche implémente la plupart des protocoles IPv6 nécessaires à un terminal mobile tels que la découverte de voisins, TCPv6, UDPv6, une grande partie des fonctionnalités de Mobile IPv6, ainsi que l'interface de programmation de type socket. Le but de cette expérimentation sera d'illustrer la continuité de fonctionnement de l'application ping lorsque le terminal mobile s'attache à différents points d'accès lors d'épisodes de mobilité, sans qu'il soit nécessaire de reconfigurer sa couche réseau et de redémarrer l'application comme ce serait le cas sans MIPv6.

L'application ping, fréquemment utilisée pour les tests de connectivité de réseaux, implique une succession d'échanges de paquets symétriques et s'exécute sur le MN. Elle envoie un nombre paramétrable de paquets ICMP de type echo-request, de taille également paramétrable, et attend que le correspondant réponde à chaque paquet echo-request par un paquet ICMP de type echo-reply.

A) Topologie

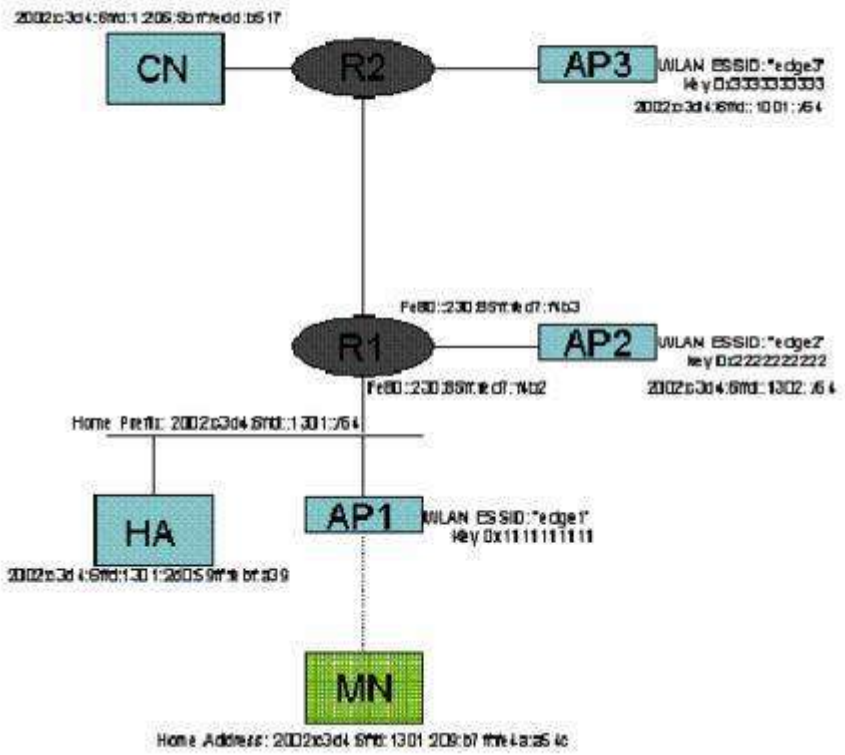


Figure 13-27 *Plate-forme de test*

Une topologie minimale pour tester la mobilité IP requiert l'utilisation de plusieurs systèmes. Il est en effet nécessaire d'utiliser au moins trois ordinateurs différents pour le MN, le CN et le HA. Ensuite, comme chacun de ces systèmes doit être positionné dans des sous-réseaux différents, quelques routeurs seront nécessaires pour constituer un réseau cœur. Finalement, pour que le terminal soit effectivement mobile, l'utilisation d'une technologie lien sans fil s'impose, donc au moins deux points d'accès, en l'occurrence WiFi (802.11b).

Sur la figure plateforme de test, les deux routeurs centraux R1 et R2 forment un réseau cœur, au bord duquel les HA, MN et CN sont positionnés. Le réseau mère interconnecte le HA, le MN et le AP1. Les réseaux visités, offrent l'accès WiFi 802.11b au moyen de deux AP's (Access Points) différents : AP1 et AP2. Finalement, le CN est positionné dans un réseau entièrement séparé.

B) Configuration Initiale

Initialement, le MN est positionné dans son réseau mère et configuré sans Mobile IPv6, c'est-à-dire comme un système IPv6 pourvu d'une adresse auto-configurée dynamiquement ainsi que d'une route par défaut. Dans la description suivante, le code source de la souche LIVSIX à été téléchargé et installé sur le MN et le HA dans le répertoire noté `<l>`, les noyaux de ces deux systèmes ont été configurés pour accepter LIVSIX, les sources respectives ont été compilées et R1 et R2 sont configurés pour envoyer des RA's sur leur liens avec une fréquence élevée (typiquement 50ms au lieu de 3 secondes). Les instructions d'installation détaillées se trouvent dans le fichier `INSTALL` de la distribution LIVSIX.

La configuration la plus simple de la souche du MN passe par la modification du fichier livsix.sh dans le répertoire `<l>/userspace`. Il s'agit d'indiquer seulement le paramètre `MCONF`, pour spécifier MN :

```
#!/bin/sh
# Copyright Emmanuel Riou, Alexandru Petrescu,
# Motorola Labs 2000, 2001, 2002, 2003, 2004
#
# Load LIVSIX module
#
# Automatically loads Livsix kernel module and configures it. This
# Script works only on Linux: hasn't been tested on other System.
LOCKDIR=/var/lock/subsys
# ISROUTER=1 means the machine forwards packets according to the
# routing table. ISROUTER=0, or commented, will not forward packets.
# ISROUTER=0
# To set a default interface, normally we have to check its validity
# first by sending a router sollicitation \ (cf: sysctl entry :
# rs_device) But the default interface can be set directly by writing
# into sysctl entry defint please make sure the chosen default
# interface is up and connected to the network ! # DEFINT=eth0
# MCONF: Mobility configuration (mandatory pour activer la mobilité)
# MCONF = 1 pour configurer le n?ud en « Home Agent »
# MCONF = 0 pour configurer le n?ud en « mobile node »
# A noter que si MCONF n'est pas défini, la mobilité sera désactivée
MCONF=0
[...]
# HOMEAGENT address
```

```
# Should be commented in case LIVSIX is acting as a HA.
#
# HOMEAGENT=2002:c3d4:6ffd:1201:2D0:59FF:FEAB:E83D
[...]
```

Une fois cette configuration faite, il est nécessaire de vérifier par `ifconfig`, qu'aucune autre souche IPv6 n'est déjà lancée avant de démarrer LIVSIX (aucune adresse IPv6 n'est déjà associée à l'interface) :

```
[root@MN userspace]# ifconfig eth1
eth1 Link encap:Ethernet HWaddr 00:09:B7:4A:A5:4C
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:465 errors:12532 dropped:0 overruns:0 frame:12532
TX packets:27 errors:9 dropped:0 overruns:0 carrier:9
collisions:0 txqueuelen:1000
RX bytes:32172 (31.4 Kb) TX bytes:4518 (4.4 Kb)
Interrupt:3 Base address:0x100
```

Le lancement de la souche se fait en exécutant depuis le répertoire d'installation le fichier `livsix.sh`:

```
[root@MN userspace]# ./livsix.sh start
Starting LIVSIX: [ OK ]
eth1:
FE80::209:B7FF:FE4A:A54C
eth0:
FE80::2D0:59FF:FECC:A14A
lo:
::0.0.0.1
```

À la fin du lancement de la souche, la commande `livconfig` est appelé pour afficher certains paramètres de la souche. `livconfig` permet également de contrôler les différents paramètres d'IPv6, comme la HoA, ou même les délais TCPv6. La commande standard ifconfig peut elle être utilisée pour observer l'apparition des adresses IPv6 sur l'interface :

```
[root@MN userspace]# ifconfig eth1
eth1 Link encap:Ethernet HWaddr 00:09:B7:4A:A5:4C
inet6 addr: 2002:c3d4:6ffd:1301:209:b7ff:fe4a:a54c/64 Scope:Global
inet6 addr: fe80::209:b7ff:fe4a:a54c/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:758 errors:18926 dropped:0 overruns:0 frame:18926
TX packets:39 errors:9 dropped:0 overruns:0 carrier:9
collisions:0
RX bytes:51972 (50.7 Kb) TX bytes:5358 (5.2 Kb)
```

Dans ce cas particulier, la souche a auto-configuré une adresse IPv6 locale (`fe80::209:b7ff:fe4a:a54c`) basée sur l'adresse MAC de l'interface ainsi qu'une adresse globale (`2002:c3d4:6ffd:1301:209:b7ff:fe4a:a54c`) basée sur la même adresse MAC et sur le préfix `2002:c3d4:6ffd::/64` reçu du RA du R1. En plus, la souche a auto-configuré une route par défaut qui peut être visualisé avec la commande `livconfig` :

```
[root@MN userspace]# ./livconfig -r
./livconfig: Default Router List:
FE80::230:85FF:FED7:F4B2 00:30:85:d7:f4:b2 (eth1)
```

Une fois la souche IPv6 configurée, il est déjà possible d'exécuter les applications qui supportent IPv6, par exemple ping, utilisée ici pour tester la connectivité entre MN et CN :

```
[root@MN userspace]# ping6 2002:c3d4:6ffd:1:206:5bff:fedd:b517
PING 2002:c3d4:6ffd:1:206:5bff:fedd:b517(2002:c3d4:6ffd:1:206:5bff:fedd:b517) 56 data
bytes
```

```
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=1 time=10.1 ms
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=2 time=5.05 ms
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=3 time=5.08 ms
--- 2002:c3d4:6ffd:1:206:5bff:fedd:b517 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2018ms
rtt min/avg/max/mdev = 5.055/6.761/10.143/2.392 ms
```

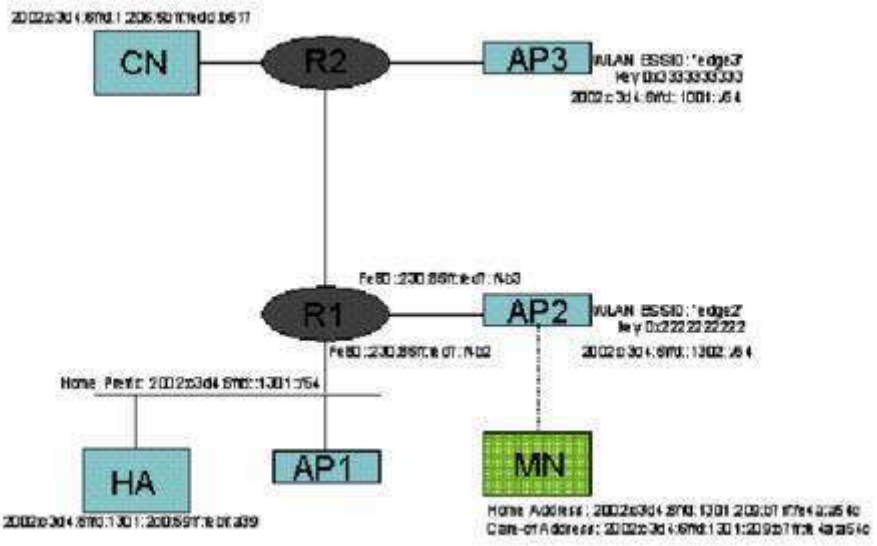


Figure 13-28 *Nœud mobile déplacé*

Cet échange de requêtes/réponses continue aussi longtemps que la connexion sans fil du MN à AP1 est maintenue. Si la connexion au AP1 est interrompue en attachant MN au AP2 (cf. figure Nœud mobile déplacé), l'échange est arrêté (on n'utilise pas Mobile IPv6 pour l'instant). Cette interruption est due au changement d'adresse IPv6 du MN.

```
[root@MN userspace]# ping6 2002:c3d4:6ffd:1:206:5bff:fedd:b517
PING 2002:c3d4:6ffd:1:206:5bff:fedd:b517(2002:c3d4:6ffd:1:206:5bff:fedd:b517) 56 data
bytes
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=1 time=9.61 ms
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=2 time=5.20 ms
64 bytes from 2002:c3d4:6ffd:1:206:5bff:fedd:b517: icmp_seq=3 time=10.3 ms
[changement d'attachement du AP1 vers AP2]
[block]
^C
--- 2002:c3d4:6ffd:1:206:5bff:fedd:b517 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3029ms
rtt min/avg/max/mdev = 5.201/8.388/10.355/2.276 ms
```

On remarquera que l'interface a acquis une nouvelle adresse valable sous AP2 et qu'une nouvelle route par défaut a été configurée :

```
[root@MN userspace]# ifconfig eth1
eth1 Link encap:Ethernet HWaddr 00:09:B7:4A:A5:4C
inet6 addr: 2002:c3d4:6ffd:1302:209:b7ff:fe4a:a54c/64 Scope:Global
inet6 addr: 2002:c3d4:6ffd:1301:209:b7ff:fe4a:a54c/64 Scope:Global
inet6 addr: fe80::209:b7ff:fe4a:a54c/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2324 errors:50553 dropped:0 overruns:0 frame:50553
TX packets:327 errors:9 dropped:0 overruns:0 carrier:9
collisions:1
```

```
RX bytes:167860 (163.9 Kb) TX bytes:32542 (31.7 Kb)
[root@MN userspace]# more /proc/net/livsix_drlist
FE80::230:85FF:FED7:F4B3 00:30:85:d7:f4:b3 (eth1)
FE80::230:85FF:FED7:F4B2 00:30:85:d7:f4:b2 (eth1)
```

Le MN a envoyé 4 paquets Echo Request au CN et en a reçu seulement 3. La réponse perdue a été en fait envoyé au AP1 et, comme MN ne se trouve plus sur son réseau mère (sous AP1) mais dans un réseau visité (sous AP2), toutes les autres réponses du CN sont perdues.

C) Mouvement

Pour pouvoir gérer dynamiquement ce changement d'adresse du MN et rediriger les réponses arrivées a AP1 vers AP2, il est nécessaire de configurer le HA sur le réseau mère et spécifier au MN l'adresse du HA. Le fichier `livsix.sh` du HA contiendra au moins le paramètre `MCONF=1` et dans le fichier `livsix.sh` du MN le paramètre

```
HOMEAGENT=2002:c3d4:6ffd:1301:2d0:59ff:febf:a39
```

est spécifié.

```
# MCONF: Mobility configuration (mandatory)
# HA = 1
# MN = 0
MCONF=1
```

Ensuite le script livsix.sh du HA est lancé :

```
[root@HA userspace]# ./livsix.sh start
Starting LIVSIX: [ OK ]
Initial Refresh Interval set to 8
LIVSIX box configured as Home Agent
eth0:
FE80::2D0:59FF:FEBF:A39
lo:
::0.0.0.1
```

Dans le fichier livsix.sh du MN le paramètre `HOMEAGENT = 2002:c3d4:6ffd: 1301:2d0:59ff:febf:a39` est spécifié, livsix.sh est relancé sur le MN positionné cette fois à la maison. Après avoir acquis une adresse valable dans le réseau mère (qui devient en effet la HoA), la commande ping vers le CN est redémarrée. Ensuite le MN est déplacé du AP1 vers AP2. On remarquera qu'après un court délai (entre 50ms et plusieurs secondes, dépendant de la fréquence de RA's), les réponses du CN vont commencer à arriver à AP2 et par conséquent au MN.

Ces réponses sont initialement interceptées par HA grâce à son cache d'adresses et ensuite encapsulées vers AP2 et la CoA du MN. Pour remplir son *Binding Cache*, le HA utilise la mise à jour d'association envoyée par MN une fois sa nouvelle CoA configurée. À la réception du BU, HA répond avec l'acquittement BAck (*Binding Acknowledgement*). Un exemple de capture de paquets BU et BAck réalisée avec le logiciel Ethereal sur HA est montré :

```
Internet Protocol Version 6
Version: 6
```

```
Traffic class: 0x00
Flowlabel: 0x00000
Payload length: 40
Next header: IPv6 destination option (0x3c)
Hop limit: 254
Source address: 2002:c3d4:6ffd:1302:209:b7ff:fe4a:a54c
Destination address: 2002:c3d4:6ffd:1301:2d0:59ff:febf:a39
Destination Option Header
Next header: Mobile IPv6 (0x87)
Length: 2 (24 bytes)
PadN: 4 bytes
Option Type: 201 (0xc9) - Home Address Option
Option Length : 16
Home Address : 2002:c3d4:6ffd:1301:209:b7ff:fe4a:a54c (
Mobile IPv6
Payload protocol: IPv6 no next header (0x3b)
Header length: 1 (16 bytes)
Mobility Header Type: Binding Update (5)
Reserved: 0x00
Checksum: 0x3d51
Binding Update
Sequence number: 0
1... .... = Acknowledge (A) flag: Binding Acknowledgement requested
.1.. .... = Home Registration (H) flag: Home Registration
..1. .... = Link-Local Compatibility (L) flag: Link-Local Address Compatibility
...1 .... = Key Management Compatibility (K) flag: Key Mngnt Mobility Compatib.
Lifetime: 65535 (262140 seconds)
Mobility Options
PadN: 4 bytes
Internet Protocol Version 6
Version: 6
Traffic class: 0x00
Flowlabel: 0x00000
Payload length: 40
Next header: IPv6 routing (0x2b)
Hop limit: 255
Source address: 2002:c3d4:6ffd:1301:2d0:59ff:febf:a39
Destination address: 2002:c3d4:6ffd:1302:209:b7ff:fe4a:a54c
Routing Header, Type 2
Next header: Mobile IPv6 (0x87)
Length: 2 (24 bytes)
Type: 2
Segments left: 1
Home Address : 2002:c3d4:6ffd:1301:209:b7ff:fe4a:a54c
Mobile IPv6
Payload protocol: IPv6 no next header (0x3b)
Header length: 1 (16 bytes)
Mobility Header Type: Binding Acknowledgement (6)
Reserved: 0x00
Checksum: 0xebd2
Binding Acknowledgement
Status: Binding Update accepted (0)
1... .... = Key Management Compatibility (K) flag: Key Mngt Mobility Compatib.
Sequence number: 0
Lifetime: 16383 (65532 seconds)
Mobility Options
PadN: 4 bytes
```

Suite à cet échange, la BC du HA peut être consultée ainsi que la " BU list " du MN :

```
[root@HA userspace]# ./livconfig -b
./livconfig: Binding Cache:
HOME ADDRESS CARE-OF ADDRESS lt
2002:C3D4:6FFD:1301:209:B7FF:FE4A:A54C 2002:C3D4:6FFD:1302:209:B7FF:FE4A:A54C 65529
[root@MN userspace]# cat /proc/net/livsix_bulist
```

```
Hoa\Coa\HA\lifetime
2002:C3D4:6FFD:1301:209:B7FF:FE4A:A54C          2002:C3D4:6FFD:1302:209:B7FF:FE4A:A54C
2002:C3D4:6FFD:1301:2D0:59FF:FEBF:A39 65535
```

D) Conclusions

Cet exemple démontre l'utilisation de la souche LIVSIX et les bénéfices du protocole Mobile IPv6. Avec Mobile IPv6, une application continuera a fonctionner sans interruption quand un terminal mobile change sont point d'attachement. Plusieurs autres fonctionnalités peuvent être expérimentées avec cette souche. Par exemple, il est possible de commencer les mouvements à partir d'un réseau visité (et non pas du réseau mère) en spécifiant la variable `HOMEADDRESS` sur MN ; une manière encore plus simple serait de ne spécifier sur MN que le préfix du réseau mère (et pas l'adresse entière du HA) ensuite, pour la configuration des autres paramètres de mobilités, utiliser:

- DHAAD (*Dynamic Home Agent Address Discovery*) qui permet à un nœud mobile distant de découvrir son HA ou
- MPD (*Mobile Prefix Discovery*) qui permet à un nœud mobile distant de reconfigurer ses adresses dans les cas ou, pendant son déplacement, les préfixes du lien mère changent. .

XIV) Intégration d'IPv6 et des applications

L'étape de standardisation des protocoles de base de IPv6 (*core specs*) est achevée, le développement de techniques assurant la transition devient un point clé dans le déploiement à grande échelle du protocole IPv6. Pour certaines catégories d'applications comme le mail, le web, le succès d'IPv6 est fortement lié à l'interopérabilité avec IPv4 puisque jusqu'à présent la majorité des informations et des utilisateurs ne sont accessibles qu'avec cette version du protocole. Il faut donc pouvoir amorcer un cercle vertueux qui permettra de passer à IPv6.

La phase de transition doit être simple, ou au minimum moins compliquée qu'une utilisation prolongée d'IPv4. Elle doit être également souple pour permettre un étalement dans le temps de la transition. Il n'y a pas de jour J pour le passage d'IPv4 à IPv6, il n'y a également pas d'échéance pour la disparition du protocole IPv4.

Généralement, l'identification d'une *killer application* est recherchée pour justifier un passage rapide vers IPv6. Ce fut le cas avec IPv4 quand le Web est apparu. Les sites sont massivement passés de protocoles propriétaires (IPX, NETBUI) vers IPv4 pour accéder aux informations par un navigateur, ce qui à conduit au concept d'intranet. On ne connaît pas actuellement d'applications particulières pouvant forcer massivement le passage vers IPv6. Les fonctionnalités sont les mêmes, puisqu'il ne s'agit que d'une nouvelle version du protocole IP. La qualité de service est souvent évoquée, mais il s'agit d'un leurre, car les mécanismes de réservation ou de différenciation se trouvent indifféremment dans les deux versions du protocole.

Les raisons qui vont nécessiter le passage à IPv6 vont être fortement liées à la pénurie d'adresses pour les nouveaux réseaux et aux fonctionnalités de configuration automatique que requièrent un grand espace d'adressage :

- le réveil des pays de la zone Asie Pacifique (en particulier la Chine et l'Inde) qui ont actuellement un taux d'adresses par habitant très faible,
- la téléphonie mobile avec GPRS puis les services de 3ème génération où chaque téléphone pourra se voir attribuer une adresse, voire un préfixe,
- les réseaux domotiques ou personnels, où la simplicité de configuration va être un critère important. Dans ces réseaux, on prévoit la généralisation de nouveaux types d'applications, qui sans être LES "killer applications" ne se contentent pas d'un fonctionnement en mode client/serveur :
- les applications de type peer-to-peer (comme la téléphonie sur IP, les jeux répartis,...) nécessitant des connexions entrantes, contrairement aux architectures clients/serveurs autorisées par l'emploi de l'adressage privé,
- les connexions permanentes où les mécanismes d'allocation d'adresses temporaires (PPP, DHCP,...) sont inefficaces.
- les grands parcs de machines où les mécanismes de configuration automatique vont simplifier la gestion du réseau. Avec IPv6, le réseau peut se gérer uniquement au niveau des routeurs, les stations construisant leur adresses automatiquement, alors qu'avec IPv4, chaque équipement doit être configuré.

A l'inverse, plusieurs autres facteurs vont freiner le déploiement d'IPv6 pour rester dans une situation de statu quo ou d'emploi d'autres mécanismes au-dessus d'IPv4. Ces facteurs limitant le déploiement d'IPv6 sont de plusieurs ordres :

- techniques :
    - la disponibilité du code IPv6 dans les équipements terminaux (PC, stations de travail, imprimantes,...) et dans les routeurs. Cette phase est achevée pour la plupart des équipementiers. Les fabricants d'équipements d'interconnexion ont intégré le code IPv6 dans leur nouveaux systèmes d'exploitation. Le monde Unix est également à la pointe pour la disponibilité d'une pile IPv6. Windows dispose aussi nativement d'une pile IPv6 aisément configurable pour les versions XP.
    - si les piles IPv6 commencent à se généraliser, nombre d'applications spécialisées ne sont pas encore prêtes (environnement de travail intégré, ...) surtout dans les environnements où le code source n'est pas disponible.
      Aujourd'hui , plus aucune application ne devrait être programmée sans pouvoir être utilisée dans un environnement IPv6. Cela implique, soit l'utilisation de l'API IPv6, soit tout simplement l'usage de règles de programmation, comme éviter de considérer l'adresse IP comme un entier, et de s'en servir comme d'un identificateur.
      C'est actuellement le facteur le plus bloquant pour un déploiement massif d'IPv6.
    - le problème lié à la phase de démarrage, dit de la poule et l'oeuf : les sites ne vont pas migrer vers IPv6 puisque les opérateurs n'offrent pas de connectivité IPv6 et les opérateurs ne vont pas offrir de réseaux IPv6 puisqu'il n'y a pas de client.
      En réalité il s'agit d'un faux problème. Un réseau IPv6 peut être déployé en intranet, les communications avec l'Internet restant en IPv4.
      De plus les opérateurs commencent à déployer des réseaux IPv6, les autorités régionales (RIPE-NCC, APNIC, ARIN,...) allouent des préfixes officiels, et les réseaux se mettent en place.
      Un site voulant acquérir une expérience en IPv6 peut se raccorder relativement facilement à

un de ces réseaux. Si cette démarche est impossible, l'emploi de 6to4 permet de construire son propre préfixe IPv6 et de s'interconnecter au reste du monde IPv6, D'autres mécanismes, comme Tunnel Broker permettent de se connecter tres simplement à l'Internet IPv6. Ces mécanismes sont passés en revue au long de ce chapitre.

- psychologiques :
  - IPv6 peut sembler mystérieux au premier abord et la taille des adresses peut décourager un ingénieur réseau habitué à manipuler les netmasks et les adresses IPv4. En réalité, les principes sont très similaires, et après quelques heures d'entraînement, leur emploi est relativement simple. Par ailleurs, ce livre participe, nous l'espérons, à ce transfert de connaissance et à la "dédramatisation" de cette nouvelle version du protocole IP.
  - le risque de casser quelque chose qui fonctionne correctement en changeant le protocole. Cette possibilité peut exister, mais la migration en douceur du réseau, sans jour J, permet de se focaliser sur les nouveaux réseaux tout en laissant les anciens fonctionner sous IPv4.

La suite de ce chapitre va décrire quelques mécanismes de transition, leurs principes et leurs limites. Le choix repose sur l'expérience de ces mécanismes et l'intérêt qu'ils offrent. Il ne semble pas qu'il soit nécessaire, voire possible, de définir un mécanisme unique et universel de transition entre les deux mondes. Dans certains cas, comme par exemple l'utilisation d'adresses IPv6 au sein d'une entreprise, seules les connexions sortantes doivent être autorisées. Chaque mécanisme répond à un besoin particulier :

- la construction d'une infrastructure IPv6 même si les équipements d'interconnexion ne gèrent que le protocole IPv4. Ces mécanismes sont principalement à base du tunnels statiques où les paquets IPv6 sont encapsulés dans des paquets IPv4.
- l'accès à un réseau IPv6 existant. Ces mécanismes sont principalement axés autour de la création dynamique de tunnels (6to4, Tunnel Broker). Le chapitre Déploiement IPv6 des fournisseurs d'accès (ISP) décrit l'utilisation de ces mécanismes.
- les mécanismes de traduction permettant la communication entre les deux versions du protocole pour que des applications conçues pour IPv4 et celles pour IPv6 puissent échanger des données. Cette traduction peut se faire à différents niveaux de l'architecture réseau :
- au niveau applicatif avec des relais (ALG : *Application Level Gateway*) ou proxy. Si l'application est connue, comme pour le web, le DNS les serveurs d'impressions ou le web, la traduction est relativement simple à faire. Cette méthode de migration devrait permettre de traiter la majorité des flux.
- au niveau transport avec des relais UDP et TCP [RFC 3142],
- au niveau IP, avec la création d'une interface encapsulant des paquets IPv4 dans des paquets IPv6 et se voyant allouer de manière temporaire une adresse IPv4 (DSTM : *Dual Stack Transition Mechanism*),
- au niveau de l'équipement de sortie d'un site avec des mécanismes de traduction d'en-tête. Cette traduction peut se faire sans installer d'état dans le routeur d'un site avec SIIT (Stateless IP/ICMP Translation), le paquet IPv4 est construit à partir d'informations déjà contenues dans l'en-tête IPv6, en particulier un format d'adressage particulier permet de véhiculer une adresse IPv4 dans les adresses IPv6.
  Par contre NAT-PT utilise les mêmes mécanismes que pour le passage d'une adresse IPv4 privée vers une adresse IPv4 publique, un pool d'adresses IPv4 est alloué au boîtier traducteur.
- La difficulté d'assurer la compatibilité entre les deux mondes n'est pas symétrique. Avec les mécanismes comme NAT-PT, SIIT ou DSTM, il est beaucoup plus facile d'initier une session partant du monde IPv6 pour aller vers le monde IPv4. En effet, il est possible d'ajouter des fonctionnalités au monde IPv6 pour faciliter la cohabitation. De plus comme une adresse IPv6 est quatre fois plus

grande, elle peut contenir une adresse IPv4.
Dans le sens inverse, il est impossible de modifier l'existant en IPv4. Ces différents mécanismes de transition s'appuient sur le DNS pour déclencher l'établissement d'un contexte (NAT-PT) ou l'allocation d'adresses temporaires (SIIT, DSTM). Le nom de l'équipement devient la référence commune entre les mondes IPv4 et IPv6.

Le déploiement progressif de ces mécanismes permet d'introduire graduellement et indépendamment le protocole IPv6 dans tous les segments du réseau. Chaque mécanisme a une portée bien définie (terminal, site, réseau). Il apporte une pièce au puzzle que constitue le passage d'IPv4 à IPv6.

En contrepartie chaque mécanisme de transition introduit une complexité supplémentaire dans le réseau. Ces mécanismes dits de transition n'ont pas pour vocation d'exister durablement. Ils devraient décroître dans le temps en fonction du nombre d'équipements IPv6 présents sur le réseau. De plus, les mécanismes de cohabitation (SIIT, DSTM, NAT-PT), ne visent que les applications existantes. Les nouvelles applications en particulier pour la domotique pourraient directement démarrer en IPv6.

1) Etat de la standardisation à l'IETF

A) Working Group ngtrans : approche "boite à outils"

D'une façon générale, l'une des clés de l'adoption d'une nouvelle technologie repose sur la facilité avec laquelle il est possible d'abandonner l'ancienne au profit de la nouvelle. Aussi, le groupe de travail IETF ngtrans a été créé, dès l'origine et en parallèle du groupe de travail IETF ipng, pour traiter des aspects liés à la transition des réseaux et applications d'IPv4 vers IPv6. La principale contrainte à respecter étant qu'il ne doit pas y avoir de jour J pour le basculement de l'ensemble de l'Internet vers IPv6 (à l'image du passage à l'an 2000 par exemple), et qu'au contraire il doit être possible de passer graduellement et progressivement d'IPv4 à IPv6, à tout moment et indépendamment de l'infrastructure réseau considérée.

Ngtrans a donné le jour à nombre de mécanismes de transition, permettant chacun de résoudre une problématique de transition particulière (interconnexion de réseaux IPv6 isolés, communication entre applications IPv4 et IPv6, transport de flux IPv6 dans les réseaux IPv4, etc...). Finalement une boîte à outils très complète a été définie, apportant des solutions a un vaste ensemble problèmes liés à la transition, soit localement à un terminal, soit plus globalement à l'échelle d'un réseau ou même de l'Internet dans son ensemble.

Cette mission remplie pour sa majeure partie, le groupe de travail ngtrans a finalement été clos pour laisser la place à IPv6ops traitant plus globalement de l'ensemble des aspects opérationnels liés au déploiement d'IPv6. En outre, il a souvent été reproché à ngtrans d'avoir spécifié une large boîte à outils sans en avoir décrit le mode d'emploi, et sans discernement des cas de transition concrets ou théoriques. Ainsi l'adoption d'IPv6 aurait été rendue en apparence plus complexe, contrairement au but initialement recherché.

B) Working Group IPv6ops : de la transition à la coexistence (déploiement opérationnel)

Au-delà de la critique faite à ngtrans et qui est encore aujourd'hui matière à débats, le groupe de travail IPv6ops (IPv6 operations), créé en septembre 2002, s'inscrit dans une dynamique nouvelle visant à traiter de l'ensemble des problèmes opérationnels liés au déploiement d'IPv6. Fort du principe selon lequel le passage d'IPv4 à IPv6 ne peut se réaliser que progressivement selon les infrastructures concernées et graduellement dans le temps, et sur la base du constat de l'absence d'imminence véritable d'une pénurie d'adresses IPv4, la transition IPv4/IPv6 devient essentiellement une question de coexistence des deux versions du protocole IP. Entre autres, l'approche en scénarios d'intégration d'IPv6 à l'existant, initialisée par ngtrans, est reprise par IPv6ops et se trouve déclinée selon quatre grandes familles : les réseaux de "mobiles" (UMTS, 3G), les réseaux d'ISP, les réseaux d'entreprise et les réseaux SOHO/domestiques. Cette approche en scénarios, a été achevée fin 2004, selon les objectifs fixés actuellement par le groupe IPv6ops.

2) Utilisation des mécanismes d'intégration d'IPv6

Les principaux mécanismes d'intégration d'IPv6 -aussi dénommés mécanismes de transition- spécifiés par l'IETF sont regroupés dans le tableau ci-dessous. Leur utilisation possible dans différents segments du réseau est mentionnée. La difficulté est de pouvoir distinguer les différents usages de ces mécanismes, par exemple la fonction de serveur du tunnel broker, installée dans le réseau de l'ISP, et la fonction client de ce service installée chez un usager -entreprise ou particulier. Cette distinction est détaillée dans le paragraphe où le mécanisme est décrit (cf. tableau Mécanismes de transition).

Mécanismes de transition				
Mécanismes de transition	**Coeur de réseau**	**ISP**	**Entreprises**	**Particuliers**
Double pile	X	X	X	X
6PE (MPLS)	X	X	X	
6to4		X	X	X
Tunnel Broker		X	X	X
Tunnels configurés	?	X	X	X
TSP		X	X	X
ISATAP			X	

TEREDO		X	X	X
Relais applicatifs		X	X	X
NAT-PT		X	X	X
DSTM		X	X	X
SOCKS			X	X
VPN		X	X	X
L2TP		X	X	X

3) Déploiement d'IPv6 et mécanismes d'intégration

Plusieurs champs de déploiement d'IPv6 sont présentés dans la suite de ce chapitre : le déploiement d'IPv6 dans le coeur de réseau en premier lieu, dans les réseaux de fournisseurs d'accès ensuite, et pour finir, l'accès des réseaux d'entreprises et de particuliers à la connectivité IPv6.

A) Déploiement d'IPv6 dans le coeur du réseau

a) Double pile

Le mécanisme de double pile IP consiste à doter chaque équipement du réseau d'une double pile protocolaire (*Dual Stack*) et d'affecter une adresse IPv4 et/ou IPv6 à chaque interface réseau. Il s'applique à tous les segments d'un réseau. En contrepartie, ce mécanisme ne prend pas en compte les problèmes de pénurie d'adresses IPv4.

Tous les équipementiers de coeur de réseaux supportent ce mécanisme, qui permet rapidement d'acheminer du trafic IPv6 dans une infrastructure IPv4 existante. Le déploiement de ce mécanisme peut être progressif et ne concerner qu'une partie du coeur de réseau dans un premier temps. Lorsque le déploiement est partiel, une attention particulière doit être portée au protocole de routage utilisé. Dans ce cas en effet, l'activation de fonctions permettant de gérer plusieurs topologies peut s'avérer nécessaire (cf. ISIS).

Pour les équipements terminaux, ce mécanisme de transition a été défini et a été très largement employé dès le début de la standardisation d'IPv6. De la même manière, il consiste à doter chaque équipement d'une double pile protocolaire et d'affecter une adresse IPv4 et/ou IPv6 à chaque inteface réseau. Cela ne résoud pas le problème de la pénurie d'adresses IPv4, mais permet dans un premier temps d'acheminer indifférement du trafic IPv4 ou IPv6 sur un équipement donné (station, routeur).

La figure Compatibilité grâce à la double pile illustre ce principe. La station B peut parler en IPv4 avec la station A et en IPv6 avec la station C. Le listing suivant donne un exemple de configuration d'une double pile dans un environnement Unix.

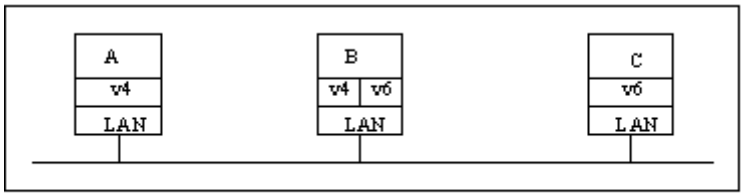


Figure 14-1 . Compatibilité grâce à la double pile

```
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet 192.108.119.134 netmask 0xffffff00 broadcast 192.108.119.255
inet6 3ffe:305:1002:1:2b0:d0ff:fe5c:4aee/64
inet6 fe80::2b0:d0ff:fe5c:4aee/64
ether 00:b0:d0:5c:4a:ee
media: 10baseT/UTP <half-duplex>
supported media: autoselect 100baseTX
```

Reste le problème des applications. Une application écrite avec l'API socket IPv6 (L'interface de programmation "socket" IPv6), utilisant en particulier des `struct sockaddr_storage` et la fonction `getaddrinfo`, peut dialoguer indifférement en IPv4 et en IPv6. Simplement, pour un serveur, deux sockets sont nécessaires, l'une pour IPv4 et l'autre pour IPv6. La station B devrait, dans l'exemple de la figure ci-dessus, posséder des serveurs pour chacune des versions de IP, ou au moins des serveurs écoutant sur plusieurs ports en parallèle. Cela peut être évité en utilisant des adresses mappées, qui permettent à une application de voir l'espace d'adresses IPv4 comme une partie de l'espace d'adressage IPv6.

Comme le montre la figure Adresse IPv4 mappée, l'adresse IPv4 est contenue dans l'adresse IPv6.

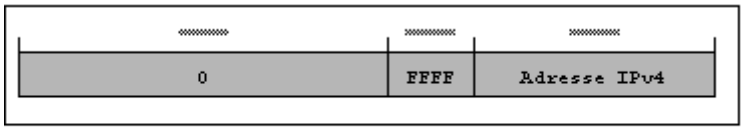


Figure 14-2 . Adresse IPv4 mappée

Quand la pile IPv4 d'un équipement reçoit un paquet et qu'une application s'est enregistrée via une socket IPv6 (famille de protocoles `PF_INET6`), les adresses IPv4 mappée source et destination sont construites à partir des informations contenues dans l'en-tête du paquet. Réciproquement quand une application IPv6 émet des paquets avec une adresse IPv4 mappée, ceux-ci sont aiguillés vers la pile IPv4.

L'exemple suivant illustre ce fonctionnement. Le client telnet, compilé en IPv6 peut contacter les équipements IPv4 en utilisant l'adresse mappée.

```
>telnet rhadamanthe
Trying 3ffe:305:1002:1:2b0:d0ff:fe5c:4aee...
Connected to rhadamanthe.ipv6.rennes.enst-bretagne.fr.
Escape character is '^]'.

FreeBSD/i386 (rhadamanthe.ipv6.rennes.enst-br) (ttyp3)

login:^D
>telnet bloodmoney
```

```
Trying ::ffff:193.52.74.211...
Connected to bloodmoney.rennes.enst-bretagne.fr.
Escape character is '^]'.


SunOS UNIX (bloodmoney)

login:
```

Le mécanisme de double pile permet de résoudre tous les problèmes d'interopérabilité liés à l'introduction de la pile IPv6. Quand cela est possible, la communication se fait en utilisant la nouvelle version du protocole. Dès qu'un des éléments n'est pas compatible (réseau, système d'exploitation, application), le protocole IPv4 est utilisé. Le principal intérêt vient du fait qu'il est possible d'écrire des applications en IPv6 qui restent compatibles avec les applications IPv4 existantes. Pourtant ce mécanisme n'est pas suffisant :

- Il ne résout pas le problème de la pénurie d'adresses puisque chaque machine doit disposer d'une adresse IPv4 et d'une adresse IPv6. Cela complique aussi les mécanismes de configuration automatique.
- Il implique que tous les routeurs soient aussi configurés pour router les deux types de paquets.
- Les applications doivent être compilées pour IPv6, ce qui implique la disponibilité du code source et un "effort" de reprogrammation.

Les applications recompilées avec l'API IPv6 ne fonctionnent que sur des équipements pourvus d'un système récent (et d'une pile IPv6 si on utilise les adresses IPv4 mappées), ce qui peut poser des problèmes de compatibilité entre les différentes versions d'un système.

b) 6PE (MPLS)

Ce mécanisme profite de la commutation de MPLS (*Multi Protocol Label Switching*) selon l'étiquette insérée dans un paquet, pour rendre un réseau capable de transporter des paquets IPv6 sans avoir à en modifier tous les équipements. Le coeur du réseau MPLS (les équipements P : *Provider*) reste inchangé. 6PE permet à un opérateur / ISP, dont le coeur de réseau s'appuie sur la technologie MPLS pour acheminer le trafic IPv4, de ne faire évoluer que la partie périphérique de son réseau (les équipements de périphérie : PE : *Provider Edge*) pour pouvoir transporter aussi le trafic IPv6 de ses usagers. Le routage IPv6 est réalisé par les équipements de périphérie (PE) qui attribuent une étiquette à chaque paquet IPv6.

La commutation de paquets IPv6 à haut débit peut poser des problèmes si les composants électroniques chargés de la commutation ne prennent pas en compte le nouveau format de paquets. MPLS peut être une solution pour véhiculer le trafic dans un réseau ne connaissant pas IPv6 puisque la décision de commuter une trame MPLS est faite en fonction d'une étiquette placée avant le paquet.

6PE, (La technique 6PE), propose l'utilisation de BGP pour créer automatiquement des tunnels dans un système autonome. Nous allons présenter ici sommairement une des possibilités appliquée à MPLS.

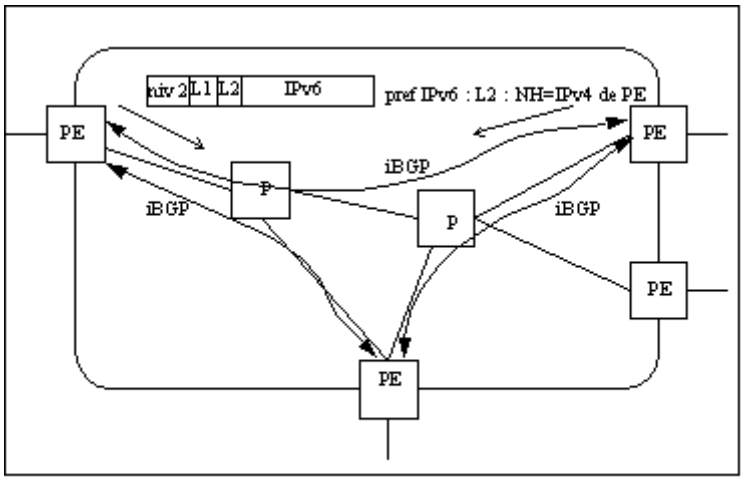


Figure 14-3 . Architecture d'un réseau MPLS

La figure Architecture d'un réseau MPLS illustre succintement l'architecture d'un réseau MPLS avec les routeurs en bordure de réseau (PE) qui insèrent les étiquettes en fonction de la destination et les routeurs en coeur de réseau (P) qui commutent rapidement les trames MPLS en fonction de l'étiquette.

Si la commutation de trame ne pose pas de problème, il faut pouvoir assigner les étiquettes en fonction de leur destination dans le réseau de l'opérateur. Or, pour ce faire, il faut modifier le processus de signalisation et par conséquent les équipements du coeur de réseau.

Les routeurs de bordure doivent être capables de prendre en compte les spécificités d'IPv6 en particulier insérer l'étiquette MPLS en fonction de l'adresse IPv6 de destination. Par contre, il existe une possibilité de ne pas modifier les équipements du coeur de réseau en utilisant le protocole de routage iBGP. Le peering BGP est fait en utilisant IPv4 comme protocole de transport. Grâce aux extensions multi-protocole de BGP, il est possible de transporter des préfixes IPv6 et les étiquettes MPLS associées à ces préfixes. Le routeur de bordure en entrée du réseau peut donc associer le préfixe IPv6 et le champ Next Hop correspondant à l'adresse IPv4 du routeur ayant fait l'annonce iBGP (48).

Le protocole de routage interne et le protocole de distribution d'étiquette permettent de construire des chemins pour les préfixes IPv4 internes. Quand un routeur MPLS reçoit un paquet IPv6, il ajoute deux étiquettes :

- la première (L1 dans l'exemple figure Architecture d'un réseau MPLS) permet de joindre le routeur de sortie. Cette étiquette est déterminée à partir de la valeur du champ *Next Hop* associé au préfixe de l'adresse de destination du paquet.
- la seconde (L2) contient l'étiquette annoncée par iBGP correspondant au préfixe IPv6.

B) Déploiement IPv6 des fournisseurs d'accès (ISP)

Les mécanismes disponibles pour ce segment de réseau, contrairement à ceux décrits précédemment, mettent en oeuvre une architecture type client/serveur. Sauf exception ou cas particulier, la partie cliente est localisée côté utilisateur et la partie serveur côté ISP.

Le point fort des mécanismes présentés ici est de permettre une mise en oeuvre de solutions dites automatiques, où l'intervention de l'administrateur est réduite à une phase de configuration/ initialisation du service.

C) 6to4

Le mécanisme 6to4 permet d'interconnecter entre eux des sites IPv6 isolés en créant des tunnels automatiques IPv6 dans IPv4 en fonction du destinataire des données. Le mécanisme définit plusieurs composants.

- la machine terminale 6to4 (dépendante de l'implantation dans le système d'exploitation)
- le routeur de bordure (ou gateway), qui doit encapsuler les paquets IPv6 dans des paquets IPv4, est connecté à IPv4 et IPv6
- le relais 6to4 est un équipement réseau dont l'adresse est bien connue (adresse anycast). Il assure la connexion à l'Internetv6.

6to4 permet à un ISP de fournir de la connectivité IPv6 à ses usagers en installant une machine unique connectée aux deux mondes IP. Il peut aussi permettre à un usager de router du trafic IPv6 même si son prestataire ne fournit qu'une connectivité et des adresses IPv4. Il faut noter que le routage entre les machines distantes a de bonnes probabilités d'être asymétrique notamment si le routeur de bordure utilise un relais 6to4 et que l'utilisation des tunnels peut conduire à des délais de propagation élevés.

On peut envisager l'usage de la technique 6to4 de deux manières :

- Comme un moyen de pression envers les ISP. Un site dont le fournisseur de service refuse d'offrir un service IPv6, n'est pas bloqué. Il dispose d'une méthode simple pour construire ses adresses IPv6 et la création de tunnels. Il suffit de trouver l'adresse IPv4 d'un routeur passerelle qui traitera les paquets.
  Cette approche conduit à un routage sous-optimal, comme indiqué précédemment, et à une anarchie dans le réseau en terme d'administration.
- Pour permettre aux ISP d'offrir un service IPv6 minimum à leurs clients. Cette approche est acceptable dans la période de déploiement d'IPv6. Le fournisseur de services met en place un routeur passerelle uniquement pour ses clients et le place par exemple dans un point d'interconnexion. D'une part, la charge administrative et technique est réduite puisque l'ISP n'a pas à gérer un nouveau plan d'adressage ou la création de nombreux tunnels. D'autre part, le routage est plus optimal puisque le relais est proche des clients et du réseau IPv6 auquel l'ISP est relié.

On pourrait envisager l'installation de relais 6to4 sur les points d'échange de l'Internet pour accélérer le déploiement et l'usage d'IPv6 par les ISP. Mais il n'y a pas de demande dans ce sens actuellement et ce mécanisme est actuellement peu déployé par les ISP. La question de sa résistance au facteur d'échelle, et des aspect liés à la sécurité, sont posées depuis longtemps. Ils n'ont pas encore trouvé de réponse.

Le préfixe `2002::/16` a été alloué par l'IANA à ce type d'adressage (cf. figure Adresse 6to4). Le gestionnaire d'un site peut aisément créer un préfixe de longueur 48 en y concaténant l'adresse IPv4 (convertie en hexadécimal) d'un routeur en bordure des réseaux IPv4 et IPv6.

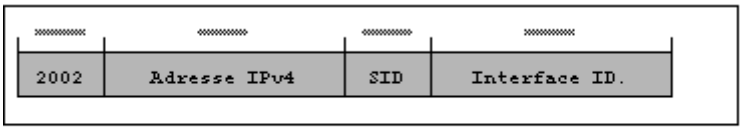


Figure 14-4 . Adresse 6to4

La figure Exemple de numérotation en utilisant le préfixe de 6to4 illustre le mécanisme d'attribution de préfixes. Le routeur RB se trouve en bordure du réseau. Il est connecté à la fois à l'Internet v4 et à un (ou des) réseau(x) IPv6. Le routeur possède obligatoirement une adresse IPv4 sur le réseau de l'ISP. Il va s'en servir pour construire les 48 premiers bits de l'adresse IPv6. C'est ce préfixe de 48 bits qui va être utilisé par l'ensemble des équipements 6to4 du site. Ce préfixe identifie un site donné. On peut remarquer que ce plan d'adressage est conforme aux plans d'adressage globaux actuellement en vigueur, puisqu'il réserve 16 bits pour numéroter les réseaux du site et 64 bits pour les identifiants d'interface.

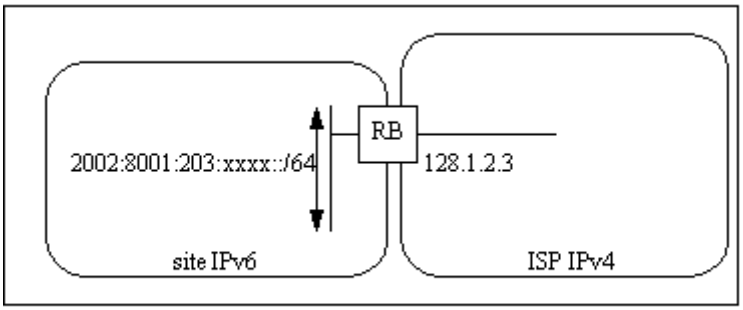


Figure 14-5 . Exemple de numérotation en utilisant le préfixe de 6to4

La figure Exemple de routage des paquets explique comment les paquets sont routés quand l'équipement A veut envoyer un paquet IPv6 à l'équipement B. Dans un premier temps, A interroge le DNS pour connaître l'adresse IPv6 de B. La réponse est une adresse du type `2002:C0C0:C0C0:....` La machine A émet un paquet vers cette destination. Les paquets dont l'adresse destination commence par le préfixe `2002::/16`, correspondant au plan d'adressage 6to4, sont routés vers un routeur tunnelier pour 6to4. Ce dernier analyse l'adresse IPv6 de destination et trouve l'adresse IPv4 de l'autre extrémité du tunnel (`192.192.192.192` dans l'exemple). Le paquet est reçu par le routeur RB qui retire l'encapsulation IPv4 et le route normalement vers la destination B en utilisant le routage interne.

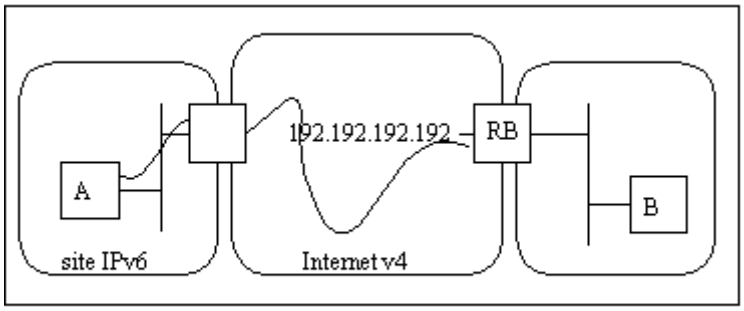


Figure 14-6. Exemple de routage des paquets

On peut remarquer dans cet exemple que l'adresse de la source peut être aussi bien une adresse IPv6 6to4 qu'une adresse IPv6 globale. Mais le dialogue dans le sens opposé est plus complexe et montre les limites de cette technique.

Un site utilisant 6to4 n'est pas, par définition, connecté à l'Internet v6. Il doit donc exister dans l'Internet v4 des routeurs servant de passerelle vers le réseau Internet IPv6. Un routeur de bordure faisant l'encapsulation des paquets IPv6 dans des paquets IPv4 peut être configuré de la manière suivante :

- si l'adresse du destinataire commence par le préfixe `2002::/16`, effectuer l'encapsulation du paquet vers le destinataire IPv4 dont l'adresse est incluse dans l'adresse IPv6 de destination,
- sinon, il s'agit d'une adresse IPv6 globale et le paquet doit être tunnelé à l'adresse IPv4 d'un routeur servant de passerelle vers le réseau IPv6.

De même, dans la figure Exemple de routage des paquets, nous avons supposé que le routeur faisant l'encapsulation était situé en bordure du réseau du site où se trouve la machine A. C'est probable si le site utilise également un plan d'adressage 6to4. Par contre si le site n'utilise que des adresses globales, voire n'a pas de connexion IPv4, l'encapsulation peut être déléguée à un routeur passerelle. Ce routeur passerelle peut en utilisant les protocoles de routage interne et externe annoncer aux équipements IPv6 cette fonctionnalité.

Le danger est d'engorger les tables de routage IPv6 avec une complexité lié à l'adressage IPv4. Pour éviter cet écueil, un routeur passerelle ne doit pas annoncer un préfixe IPv6 autre que `2002::/16`. En conséquence, les paquets émis à destination d'une adresse 6to4 seront traités par le routeur le plus proche au sens des protocoles de routage.

Il est important de respecter cette règle au niveau des annonces BGP, comme le montre l'exemple de configuration des routeurs See Règles d'annonce et d'agrégation.

Si 6to4 est une technique intéressante pour relier deux nuages IPv6 à travers un nuage IPv4, elle se complique et n'est pas optimale lorsqu'il s'agit de communiquer avec une machine dont l'adresse est issue d'un plan de numérotation global. Le routage n'est pas toujours optimal et presque assurément asymétrique :

- le site 6to4 peut avoir choisi un routeur passerelle loin du destinataire,
- le site ayant un plan d'adressage global envoie les paquets vers le routeur passerelle le plus proche au sens du routage.

Pour réduire la taille des tunnels, une adresse IP anycast a été proposée pour automatiser et simplifier la phase de configuration de l'adresse du relais. Le préfixe `192.88.99.0/24` a été attribué à ce propos [RFC 3068] et le relais prend l'adresse `2002:c058:6301::`, ou `192.88.99.1` en utilisant l'adresse IPv4. Un site offrant ce service peut annoncer ce préfixe dans le routage global de l'Internetv4 et les paquets à destination d'un relais iront vers l'équipement le plus proche.

a) Tunnel Broker

Un serveur de tunnels (IPv6 dans IPv4) permet de connecter à l'Internetv6 une machine double pile isolée dans l'Internetv4. Dans certaines versions de ce service un réseau local peut être ainsi connecté, quel que soit le nombre de machines qu'il comporte. La configuration du tunnel entre le serveur et la machine cliente est automatique et repose sur le protocole TSP. La demande de connexion au serveur est réalisée par une page HTML dont l'URL est connue à l'avance.

Ce mécanisme/service permet de fournir de la connectivité IPv6 à des équipements/réseaux locaux isolés dans l'Internetv4. Cette connectivité est en générale fournie à titre provisoire (soit en attendant que l'offre des ISP soit disponible soit pour faire des tests de validation, par exemple). Elle peut aussi être une première étape pour un prestataire de service pour procurer de la connectivité IPv6 à ses usagers.

Le service Tunnel Broker repose sur une architecture à base de client/serveur. Côté usager l'installation d'un simple client permet de faire la demande de tunnels au serveur. Ce client est en général authentifié. Pour le prestataire, il faut mettre en oeuvre un serveur qui a plusieurs fonctions : l'interface HTML pour accueillir les demandes de tunnels des usagers et la « comptabilité » qui peut l'accompagner, le configurateur de tunnels qui envoie les paramètres d'extrémité du tunnel entre l'équipement de concentration et celui de l'usager d'une part et le concentrateur de tunnels d'autre part.

De nombreuses évolutions de ce mécanisme sont en cours :

- L'authentification du client demandant à [r]établir une connexion au serveur de tunnels permet de disposer d'une fonction VPN quel que soit le lieu où se trouve l'usager dans l'Internet.
- Les implantations s'appuyant sur des tunnels UDP permettent la traversée de NAT, fonction indispensable aux terminaux (ou réseaux) situés dans un plan d'adressage privé.
- Le découpage de l'espace d'adresse pour numéroter les extrémités de tunnels et les réseaux d'interconnexion, nécessite un peu de doigté. Là aussi des évolutions sont en cours pour simplifier les implantations actuelles et mieux coller à l'expérience de déploiement des réseaux IPv6 acquise ces dernières années.

b) TSP : tunnel setup protocol

Le tunnel setup protocole [BP-id] a été défini en complément du Tunnel Broker afin de permettre une négociation automatisée des différents paramètres entrant en jeu lors de l'établissement d'un tunnel. En effet, nombre d'implémentations de Tunnel Broker sont basées aujourd'hui sur une interface Web qui permet de saisir ou de récupérer implicitement les paramètres nécessaires à l'établissement du tunnel entre le terminal et le Tunnel serveur. L'architecture d'un Tunnel broker implémentant TSP est donné figure Configuration d'un Tunnel Broker avec TSP.

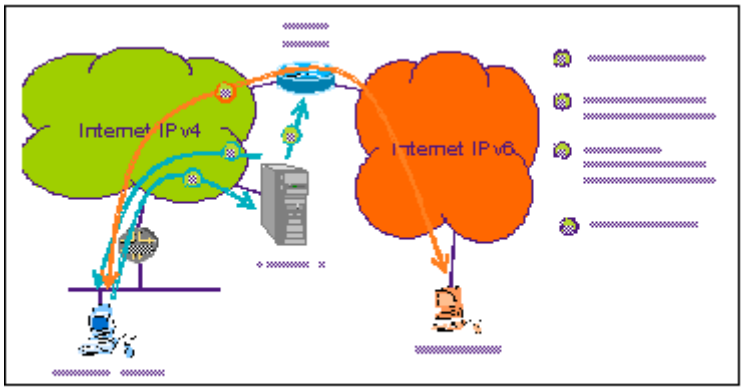


Figure 14-7. Configuration d'un Tunnel Broker avec TSP

TSP permet la négociation automatique et transparente à l'utilisateur de tout ou partie des paramètres suivants :

- le mécanisme d'authentification utilisateur utilisé,
- le type d'encapsulation utilisée : IPv4 dans IPv6, IPv6 dans IPv4, IPv6 dans UDP IPv4
- l'adresse IPv6 assignée lorsque le client TSP est un terminal
- le préfixe IPv6 alloué lorsque le client TSP est un routeur
- l'enregistrement DNS dans le cas d'un terminal
- la résolution DNS inverse dans le cas d'un routeur

La disponibilité du type d'encapsulation IPv6 dans UDP IPv4, permet d'offrir une solution de traversée de NAT, alternative à celle proposée par exemple par Teredo. Dans ce cas, le client TSP met en oeuvre un processus de découverte de NAT qui consiste simplement à envoyer au TSP serveur du Broker, un message UDP contenant l'adresse IP du terminal. Le serveur TSP serveur compare simplement l'adresse contenue dans le message avec l'adresse source du paquet UDP. Si elles sont différentes alors le terminal est situé derrière un NAT.

TSP s'appuie sur l'échange de simples messages XML dont un exemple est donné ci-dessous. Cet exemple correspond à la demande de création d'un tunnel simple par un client TSP :

```
-- Successful TCP Connection --
C:VERSION=2.0.0 CR LF
S:CAPABILITY TUNNEL=V6V4 AUTH=ANONYMOUS CR LF
C:AUTHENTICATE ANONYMOUS CR LF
S:200 Authentication successful CR LF
C:Content-length: 123 CR LF
<tunnel action="create" type="v6v4">
<client>
```

```
<address type="ipv4">1.1.1.1</address>
</client>
</tunnel> CR LF
S: Content-length: 234 CR LF
200 OK CR LF
<tunnel action="info" type="v6v4" lifetime="1440">
<server>
<address type="ipv4">206.123.31.114</address>
<address type= "ipv6">3ffe:b00:c18:ffff:0000:0000:0000:0000</address>
</server>
<client>
<address type="ipv4">1.1.1.1</address>
<address type= "ipv6">3ffe:b00:c18:ffff::0000:0000:0000:0001</address>
<address type="dn">userid.domain</address>
</client>
</tunnel> CR LF
C: Content-length: 35 CR LF
<tunnel action="accept"></tunnel> CR LF
```

D) Accès des entreprises et des particuliers à l'Internet IPv6

a) ISATAP

ISATAP (qui se prononce ice-a-tap) est en quelque sorte la déclinaison des principes de 6to4 à un réseau local, de façon à permettre un tunneling automatique et l'échanges de flux IPv6 entre terminaux double pile interconnectés via un réseau local IPv4 uniquement. ISATAP permet le déploiement de terminaux dual-stack et d'applications IPv6 sur une infrastructure locale IPv4, comme typiquement celle d'un réseau d'entreprise. ISATAP peut être déployé de plusieurs façons : soit simplement au sein des terminaux concernés afin de leur permettre d'échanger des flux IPv6 alors qu'ils sont connectés sur une infrastructure IPv4, soit en combinaison avec des routeurs 6to4 de façon à échanger des flux IPv6 localement et avec des sites distants.

ISATAP s'appuie sur un format d'adresse spécifique décrit ci-dessous, qui intègre dans la partie identifiant de terminal l'adresse IPv4 du terminal et donc la fonction d'encapsulation/désencapsulation associée.

ISATAP (*Intra-Site Automatic Tunnel Addressing Protocol*) [Templin-id] permet de connecter des équipements terminaux IPv6 isolés dans un réseau IPv4, en gérant de manière automatique l'encapsulation des paquets IPv6 dans des paquets IPv4. Contrairement à 6to4, cette technique s'applique à l'intérieur d'un domaine.

ISATAP repose sur une particularité de construction des identifiants d'interface proposée par l'IEEE. La figure Identifiant d'interface pour ISATAP montre comment un identifiant d'interface est construit à partir d'une adresse MAC en ajoutant la valeur `0xFFFE`. Or l'IEEE a prévu qu'un identifiant d'interface pouvait contenir une adresse IPv4, la valeur insérée étant alors `0xFE`, comme le montre la figure . La partie sur trois octets indiquant le constructeur prend la valeur de l'OUI (*Organisational Unit identifier*) attribué à l'IANA, c'est-à-dire `00-00-5E`.

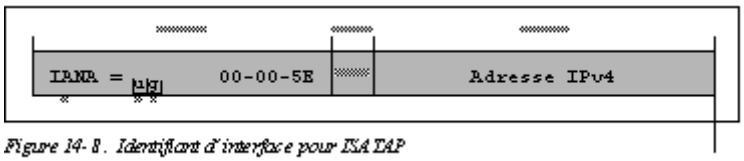


Figure 14-8. Identifiant d'interface pour ISATAP

Quand un routeur mettant en oeuvre ISATAP reçoit un paquet IPv6 dont l'identifiant d'interface commence par la séquence `00-00-5E-FE`, il en déduit que le paquet est destiné à une machine isolée et encapsule le paquet IPv6 dans un paquet IPv4 dont l'adresse de destination est celle contenue dans la partie identifiant d'interface.

L'intérêt de cette méthode est de respecter la structure actuelle des adresses IPv6 actuellement en vigueur puisque l'identifiant d'interface a toujours une longueur de 64 octets. Les stations isolées appartiennent au même lien IPv6 et partagent en conséquence le même préfixe IPv6. Mais pour construire complètement l'adresse, il faut pouvoir connaître les préfixes utilisés. Le réseau IPv4 servant à connecter les équipements IPv6 isolés est un réseau NBMA (*Non Broadcast Multiple Access*). Neighbor Discovery possède un mode adapté pour ce type de réseaux : les routeurs ne peuvent donc pas émettre de messages Router Advertisement de manière spontanée. Ces messages ne seront émis qu'en réponse à des Router Sollication.

L'algorithme de configuration d'un équipement isolé qui utilise ISATAP est le suivant :

- dans un premier temps, l'équipement doit connaître l'adresse IPv4 du routeur gérant ISATAP. Cette adresse pourrait être apprise par le DNS, ou en utilisant une adresse anycast IPv4 pour joindre le routeur le plus proche.
- l'équipement envoie un message IPv6 Router Sollicitation au routeur en utilisant comme adresse de la source, son adresse lien-local (`fe80::5e:fe:IPv4`) et comme adresse de destination l'adresse de multicast des routeurs (`FF02::02`). Ce message est encapsulé dans un paquet IPv4 dont l'adresse destination est l'adresse IPv4 du routeur.
- le routeur répond au message IPv6 Router Sollicitation en renvoyant en point-à-point, toujours encapsulé dans un paquet IPv4, la liste des préfixes IPv6 utilisés pour joindre les équipements isolés (Router Advertisement).

Il est à noter que ISATAP est compatible avec 6to4. Le préfixe global peut contenir l'adresse IPv4 du routeur d'accès et la partie identifiant d'interface l'adresse IPv4 privée de l'équiment. Deux tunnels seront nécessaire (le premier entre le routeur 6to4 de la source et le routeur d'accès du site et le second entre le routeur d'accès et le destinataire). un équipement, même s'il ne possède qu'une adresse privée en IPv4, peut de cette manière disposer une adresse IPv6 globale. Malheureusement, cette solution ne peut pas être déployée quand le routeur d'accès n'est pas configuré pour le protocole IPv6, cas généralement rencontré dans les réseaux ADSL.

b) TEREDO

Le principal objectif de Teredo (RFC 4380) est de fournir automatiquement une connectivité IPv6 à un terminal situé derrière un NAT et ne disposant donc pas d'une adresse IPv4 globale. Ce mécanisme de type client/serveur s'appuie sur des serveurs et des relais de façon à optimiser le chemin parcouru par les paquets IPv6 encapsulés qui transiteront via le relais le plus "proche" en non plus sytématiquement par un point unique comme avec le mécanisme de Tunnel Broker. Cependant, l'optimisation du chemin parcouru par les paquets IPv6 encapsulés conduit à une certaine complexité dans les échanges client/serveur/relais, en particulier lors de chaque phase d'initialisation d'une communication. Teredo est un outil de dernier recours, destiné à être utilisé en l'absence de connectivité IPv6 native, ou de tunnel configuré, ou de la mise oeuvre de 6to4. Ce mécanisme concerne exclusivement des terminaux individuels ne disposant pas d'une adresse IPv4 publique ; il ne s'applique pas à des sous-réseaux.

Teredo s'appuie sur un format d'adresse particulier (cf. figure Format des adresse Teredo) qui intègre dans la partie préfixe l'adresse IPv4 du serveur Teredo, et dans la partie identifiant les adresse et numéro de port (en sortie de NAT) du terminal client Teredo. Cette dernière infomation est brouillée afin de ne pas être modifiée par certains NAT qui systématiquement modifient les séquences binaires ressemblant à une adresse.

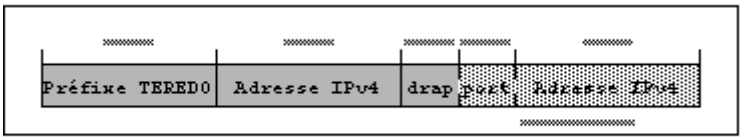


Figure 14-9 . Format des adresse Teredo

L'objectif de Teredo est de rendre entièrement automatique la configuration de tunnels IPv6-dans-IPv4 pour des terminaux situés derrière un ou plusieurs NAT, et utilisant par conséquent des adresses IPv4 privées. Pour cela, Teredo met en ?uvre une encapsulation UDP. Teredo s'appuie sur un serveur et des relais externes au réseau auquel est connecté le client Teredo, de façon à optimiser autant que possible le routage IPv6, en évitant un point d'encapsulation unique tel qu'on peut le rencontrer par exemple avec une solution de type tunnel broker. De plus, Teredo utilise un format d'adresse IPv6 spécifique qui ne requiert aucune allocation de la part des organismes officiels.

Le préfixe Teredo de longueur 64 bits inclus l'adresse du serveur Teredo auquel le terminal est rattaché. A la date d'édition de cet ouvrage le préfixe Teredo a la valeur `3FFE:831F::/32` mais l'IANA pourrait assigner une valeur définitive.

L'architecture globale et les différents éléments mise en oeuvre dans Teredo sont décrits figure architecture Teredo. Le but recherché est que chaque client Teredo soit rattaché au serveur Teredo le plus proche, et que le trafic IPv6 transite par le relais Teredo lui aussi le plus proche au sens routage IPv6.

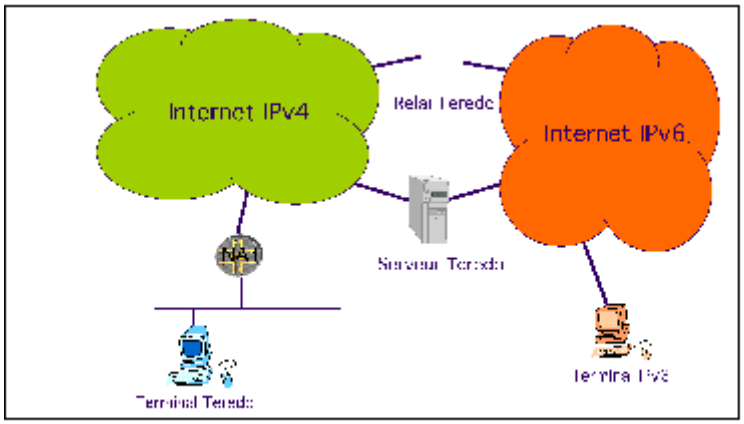


Figure 14-10 : architecture Teredo

Cependant, il existe plusieurs types de NAT, selon la politique de traduction adoptée (en particulier pour le port UDP), qui peuvent être classés selon deux grandes familles :

- celle des "cone NATs" (qui regroupe en fait trois type différents NAT) et
- celle des "symmetric NATs".
- Il est important de noter que Teredo tel qu'il est défini actuellement, ne peut pas fonctionner au travers d'un "symmetric NAT". En effet, contrairement aux "cone NATs", un "symmetric NAT" associe un couple adresse/port externe différent selon l'adresse et l'application destinatrices du datagramme, lors de la traduction de ce dernier. Ainsi, les caractéristiques du tunnel UDP d'un terminal Teredo ne sont pas globalement uniques, alors que le mécanisme d'initialisation et de maintien de contexte de Teredo requiert cette unicité.

Globalement le fonctionnement de Teredo est plutôt complexe puisqu'il nécessite la coopération de trois types de noeuds réseau (client, serveur et relais Teredo), afin de :

- de déterminer le type de NAT traversé et d'assigner une adresse IPv6 au client Teredo
- de maintenir ouvert dans le NAT, l''association entre adresse/port internes et adresse/port externes

La phase d'initialisation d'un client Teredo a en particulier pour but de déterminer le type de NAT derrière lequel se trouve le client Teredo. A l'issue de cette initialisation, et dès lors qu'aucun "symmetric NAT" n'est traversé, il est alors nécessaire de maintenir l'association ouverte dans le NAT, par l'envoi périodique d'un message spécifique vers le serveur Teredo qui a pour effet de ré-initialiser le time-out d'inactivité du NAT.

De plus, l'initialisation d'une communication IPv6 sera différente, d'une part selon le type de cone NAT traversé, et d'autre part selon la destination : client Teredo sur le même lien, client Teredo sur un site différent, terminal IPv6 externe.

L'initialisation typique d'une communication entre un client Teredo et un terminal uniquement IPv6, est illustrée dans l'exemple figure exemple d'initialisation d'une communication.

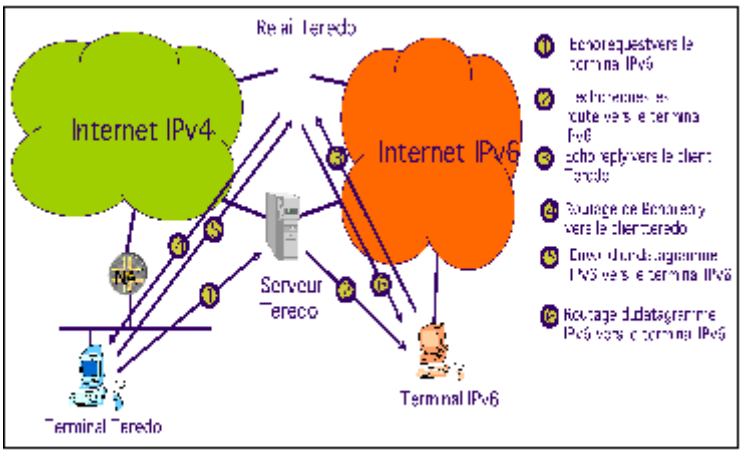


Figure 14-11 . exemple d'initialisation d'une communication

4) Mécanismes d'interopérabilité

A) Relais applicatifs

Les relais applicatifs ou ALG (*Application Level Gateway*) représentent le moyen le plus simple pour assurer une relation entre le monde IPv4 et le monde IPv6. Il s'agit de machines avec une double pile (cf. figure Exemple de relais applicatif pour le courrier électronique) configurées pour accéder aux deux versions du protocole. Les équipements IPv6 émettent leur requête vers le relais applicatif qui interprète le contenu de la requête et la retransmet en IPv4.

Un ou plusieurs relais peuvent être installés en fonction des services rendus disponibles sur le réseau (par exemple serveur d'impression, serveur de messagerie, relais http, ...). Les machines clientes doivent être configurées pour adresser leurs requêtes applicatives à ces relais.

L'usage de ces techniques est très fréquent dans les réseaux privés pour communiquer avec l'extérieur. Tous les protocoles ne peuvent pas utiliser les relais applicatifs, par exemple telnet. Mais comme la liste précédente l'indique, les ALG concernent des applications courantes qui représentent une partie importante du trafic. Cela permet également d'alléger le travail d'autres mécanismes de transition qui sont plus complexes à mettre en œuvre.

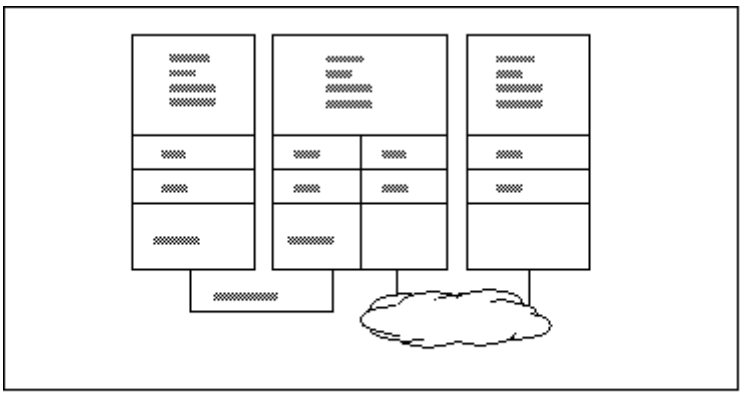


Figure 14-12 . Exemple de relais applicatif pour le courrier électronique

Les relais applicatifs regroupent :

- les proxies et les caches web,
- les spoolers d'impression,
- les serveurs de courrier électronique,
- les serveurs DNS,
- ...

a) Configuration d'un relais applicatif pour le Web

Le listing suivant donne un extrait de la configuration d'un serveur apache pour que celui-ci serve de relais aux requêtes émises par des navigateurs. Aucune configuration n'est relative au protocole IPv6. Il suffit d'activer la fonction de proxy.

```
#cat /usr/local/etc/apache/httpd.conf
#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
<IfModule mod_proxy.c>
ProxyRequests On
<Directory proxy:*>
Order deny,allow
Allow from all
</Directory>
#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server ver.;"Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
ProxyVia On
</IfModule>
# End of proxy directives.
```

B) SOCKS

Cette technique est aussi issue des solutions pour passer d'un adressage privé à un adressage public. SOCKS ajoute un "canal de signalisation" aux données transportées, ce qui permet de piloter à distance l'ouverture de connexion. En IPv4, cette technique permet d'utiliser un adressage privé en interne pour atteindre un relais SOCKS qui se trouve dans une zone démilitarisée. Le relais SOCKS se charge d'ouvrir une connexion normale avec l'équipement distant. Le relais SOCKS se plaçant dans le modèle architectural en dessous de l'application, il est relativement indépendant de celle-ci, il n'est pas nécessaire de recourir à différents types de relais. De plus, SOCKS permet aussi bien les communications entrantes que sortantes.

SOCKS s'applique à tout type de réseau, ainsi qu'à des réseaux utilisant l'adressage privé IPv4.

Le RFC 3089 étend ce mécanisme en permettant les deux versions du protocole IP. Quatre scénarios sont possibles, comme l'indique la figure suivante :

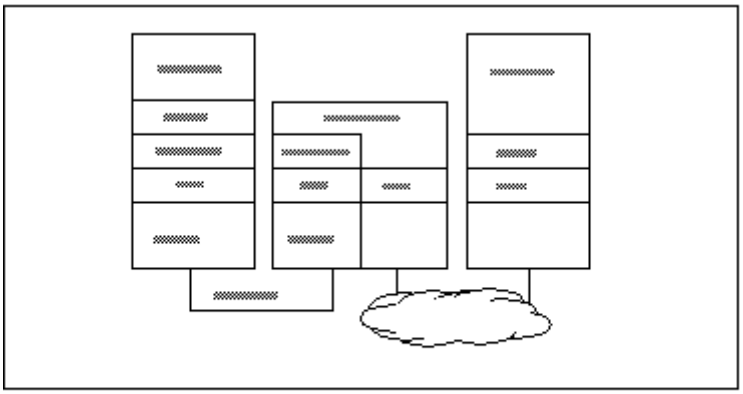


Figure 14-13. Exemple d'utilisation de SOCKS

- les protocoles X et Y sont IPv4. On est dans le cadre d'une utilisation classique de SOCKS telle qu'elle est prévue dans le RFC 1928 initial pour permettre à des équipements utilisant un plan d'adressage privé d'accéder aux ressources de l'Internet et de contrôler l'accès au réseau.
- le protocole X est IPv4 et le protocole Y est IPv6. Ce scénario permet de v6fier une application ne connaissant qu'IPv4.
- le protocole X est IPv6 et le protocole Y est IPv4. Ce scénario permet à une application IPv6 d'accéder à des ressources uniquement disponibles en IPv4
- les protocoles X et Y sont IPv6. Vu la taille des adresses IPv6, il est peu probable qu'il y ait toujours un plan d'adressage privé, non routable sur l'Internet. L'utilisation de SOCKS dans ce cas serait surtout pour contrôler l'usage des ressources réseau.

Le RFC 3089 prévoit aussi l'enchaînement de plusieurs relais SOCKS.

La principale difficulté introduite par l'utilisation de SOCKS pour la transition provient de l'interrogation du DNS. Dans le cas du deuxième scénario, où X vaut 4 et Y vaut 6, l'application ne peut que manipuler des adresses IPv4. En particulier les appels aux serveurs DNS pour la résolution de nom ne portent que sur des *Resource Records* de type `A`.

Le RFC 1928 prévoit que dans la partie "signalisation", le nom de l'équipement distant peut être envoyé de trois manières différentes :

- une adresse IPv4,
- un nom de machine,
- une adresse IPv6.

Si l'application utilise un nom de machine (FQDN : *Fully Qualified Domain Name*), SOCKS intercepte l'appel procédural pour la résolution et retourne à l'application une fausse adresse IP prise dans un poule. Quand l'application ouvre une connexion en utilisant cette adresse, le nom de la machine distante peut être retrouvé et est envoyé au relais SOCKS qui pourra, si le DNS retourne un enregistrement `AAAA`, ouvrir la connexion en utilisant le protocole IPv6.

C) DSTM

L'objectif de DSTM (*Dual Stack Transition Mechanism*) est de donner une connectivité IPv4 temporaire à un terminal dual-stack connecté à un réseau uniquement IPv6. La connectivité IPv4 n'est disponible que le temps nécessaire à une communication avec un terminal distant qui ne possède qu'une pile IPv4. Dans le principe, DSTM est en quelque sorte la réciproque du Tunnel Broker.

En général DSTM est identifié comme un mécanisme intervenant en fin de période de cohabitation IPv4/IPv6, lorsque les réseaux IPv6 sont dévenus majoritaires. Dans ce cas, DSTM peut typiquement s'appliquer soit à un réseau d'entreprise, soit à un réseau d'ISP.

DSTM se place dans la situation où une partie d'un site est passée en IPv6, mais où certaines applications continuent à utiliser IPv4. DSTM utilise une interface spéciale DTI (*Dynamic Tunneling Interface*) qui permet l'encapsulation des paquets IPv4 dans des paquets IPv6. Du point de vue de l'application, ce mécanisme est relativement transparent, puisque l'interface DTI est considérée comme une interface réseau classique.

Un routeur particulier appelé TEP (*Tunnel End Point*) possédant la connectivité entre le monde IPv4 et le monde IPv6 effectue l'encapsulation et la désencapsulation des données.

Initialement, les interfaces sont actives, mais ne disposent pas d'adresses IPv4, l'attribution d'une adresse n'est faite que lorsqu'un premier paquet est routé vers l'interface DTI. Dans ce cas, une demande d'allocation est envoyée par la machine à un serveur et une adresse IPv4 est allouée pendant la durée d'utilisation de l'interface DTI. Les paquets IPv4 sont envoyés à un TEP dont l'adresse IPv6 est généralement obtenue en même temps que l'adresse IPv4 temporaire. Le TEP garde en mémoire la correspondance entre les adresses sources IPv4 et IPv6 des paquets qu'il reçoit. De cette manière quand le destinataire sur le réseau IPv4 répond, le TEP peut déterminer vers quel équipement il peut tunneler les paquets.

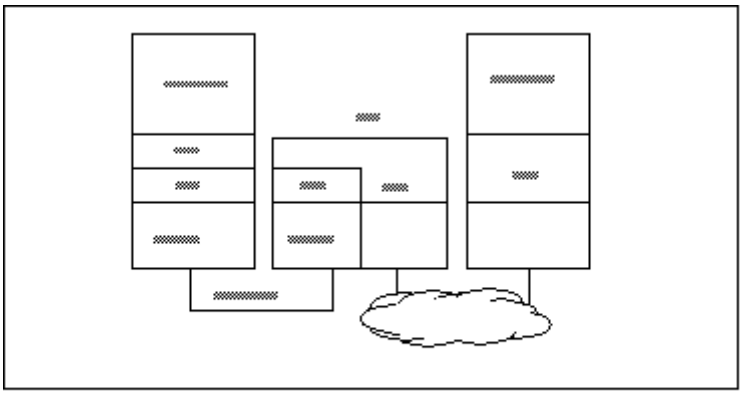


Figure 14-14 . Exemple d'utilisation de DSTM

L'usage de DSTM permet de ne configurer que la pile IPv6 d'un équipement. La pile IPv4 est configurée à la demande en fonction des besoins applicatifs. Le travail de configuation, de routage et de supervision de l'administrateur est donc simplifié.

Le fait d'utiliser des tunnels facilite l'allocation des adresses IPv4, car celles-ci perdent leur fonction de localisation, elles doivent juste conserver leur propriété d'unicité, ce qui est relativement facile à garantir.

D) NAT-PT

Le principe reste identique à celui de NAT pour IPv4. Il s'agit ici de traduire les en-têtes des datagrammes IPv6 en en-têtes de datagrammes IPv4. NAT-PT permet le déploiement de réseaux uniquement IPv6 et offrir une « compatibilité » avec le monde IPv4. Les boitiers NAT-PT sont des routeurs réalisant la traduction des paquets IPv6 en paquets IPv4 et servant de proxy DNS. Les autres équipements du réseau (derrière le boitier NAT-PT) n'ont besoin de rien de spécial pour que ce mécanisme puisse être utilisé.

- NAT-PT permet de s'affranchir d'IPv4 tout en permettant de continuer à bénéficier des services qui lui sont encore associés. Ce mécanisme introduit apres la phase de mise en œuvre (qui n'est pas très simple) un réel allègement de la tâche de l'administrateur.
- Le mécanisme NAT-PT est en discussion dans le groupe v6ops de l'IETF pour être classé historique, son usage semble devoir être limité (voire confiné) à des situations qui ne peuvent être gérées autrement. Un document de travail inventorie les scénarios qui en ont « absolument » besoin et tente de proposer des mécanismes de remplacement.

Le principe de fonctionnement de NAT-PT pour passer d'une version à l'autre du protocole IP est le même que pour passer d'un adressage privé à une adressage public, avec également les mêmes limitations, par exemple, si les paquets transportent des adresses. Par contre, ce mécanisme offre une certaine transparence au niveau du réseau.

Les paquets émis vers un équipement uniquement IPv4 utilisent l'adresse IPv4 mappée. Par contre, l'adresse source est une des adresses IPv6 globale que possède la machine émettrice. Le préfixe des adresses IPv4 mappées est routé vers un boîtier de traduction. Ce dernier dispose d'un poule d'adresses IPv4 officielles. Quand une nouvelle session est initiée par une machine IPv6, le boîtier alloue à la nouvelle adresse IPv6, une adresse IPv4 du poule, et construit un paquet IPv4 en extrayant de l'adresse IPv4, l'adresse IPv4 mappée de destination.

XV) Programmation d'applications

L'évolution de IPv4 vers IPv6 a été conçue pour minimiser les changements visibles. Un grand nombre de concepts n'ont pas changé : les noms, les "ports", l'envoi et la réception de données,... Un certain nombre de points ont malgré tout dû être modifiés. Le principal est lié à la taille de l'adresse : en IPv4, une adresse a une longueur de 32 bits (et de nombreux programmes confondent les types adresse et entier) alors qu'en IPv6 une adresse a une longueur de 128 bits ; les types liés aux adresses doivent donc être modifiés. En fait l'effet est plus profond : les nouvelles structures sont plus grandes, et certaines réservations de mémoire avec conversion de type implicite (en particulier : un entier pour une adresse, une `struct sockaddr` pour une `struct sockaddr_in`, un tampon de 16 octets pour afficher une adresse sous forme numérique) doivent être corrigés sous peine de débordement de mémoire.

L'interface de programmation réseau ("API") la plus connue est l'interface "socket" (dite aussi interface "BSD"). Le but de ce chapitre est de présenter pour cette interface de programmation les modifications introduites pour supporter IPv6, et notamment de donner une brève description des nouvelles primitives d'appel au DNS et de conversion d'adresses.

Ces modifications ont été définies pour être aussi transparentes que possible, et, s'il est en pratique toujours nécessaire de modifier un programme pour le porter de IPv4 à IPv6, un programme conçu avec des règles de typage strict est portable sans grandes modifications.

Ce chapitre illustrera l'interface de programmation "socket" pour IPv6 en présentant plusieurs exemples de programmes. Plus précisément, il détaillera successivement :

- un programme combinant les différentes fonctions de conversion d'adresse ;
- un client/serveur TCP calculant le nombre d'utilisateurs connectés sur une machine cible. En particulier, on aura soin de comparer les codes IPv4 et IPv6 de ce client/serveur, ce qui amènera à constater qu'à ce niveau de programmation, la migration vers IPv6 n'offre aucune difficulté ;
- un "mini ping" qui permettra de se familiariser avec le protocole ICMPv6 qui présente de notables différences avec son prédécesseur le protocole ICMPv4 ;
- un exemple qui génère un trafic multicast, avec abonnement et désabonnement ;
- un programme illustrant l'utilisation de l'API socket avancée.

1) L'interface de programmation "socket" IPv6

A) Ce qui a changé

Les changements opérés de façon à intégrer IPv6 concernent les quatre domaines suivants :

- les structures de données d'adresses ;
- l'interface socket ;
- les primitives de conversion entre noms et adresses ;
- les fonctions de conversions d'adresses.

Ces changements ont été minimisés autant que possible de manière à faciliter le portage des applications IPv4 existantes. En outre, et ce point est important, cette nouvelle API doit permettre l'interopérabilité entre machines IPv4 et machines IPv6 grâce au mécanisme de double pile décrit ci-après.

L'API décrite ici est celle utilisée en Solaris, Linux et systèmes *BSD. Elle correspond à celle définie dans le RFC 3493 avec quelques modifications nécessaires pour prendre en compte les dernières évolutions des protocoles sous-jacents. Cette API est explicitement conçue pour fonctionner sur des machines possédant la double pile IPv4 et IPv6 (cf. See Double pile IPv4/IPv6 pour le schéma d'implémentation d'une telle double pile sous UNIX 4.4BSD). Cette API "socket" est celle disponible dans de nombreux environnements de programmation tels que Java, perl, python, ruby, ...

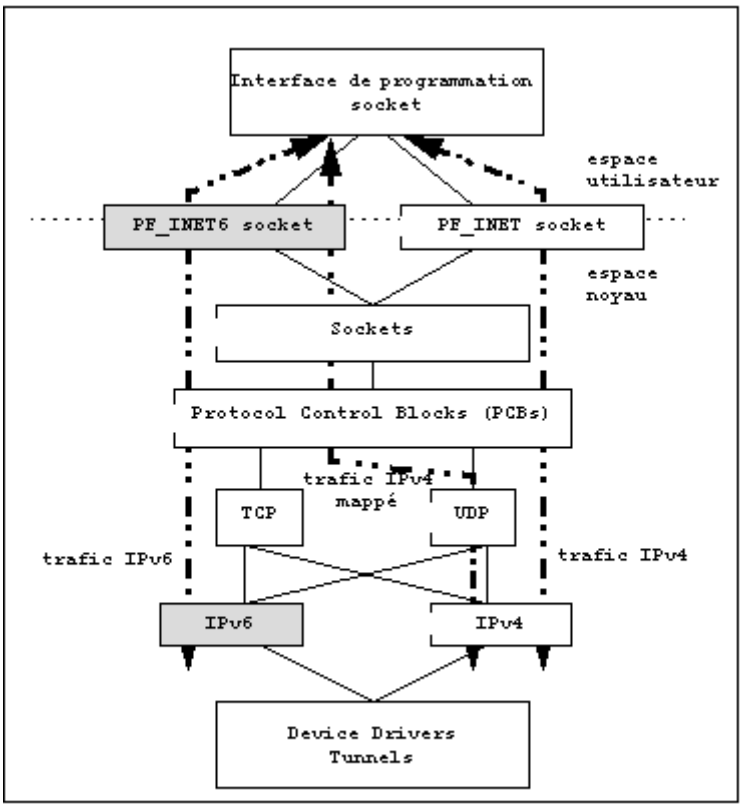


Figure 15-1. Double pile IPv4/IPv6

Une API "avancée", décrite dans le RFC 3542 permet de programmer les échanges réseaux de manière très précise. Elle sera également utilisée mais de manière succinte et essentiellement par le biais de l'exemple one_ping6.

B) Les structures de données d'adresses

Une nouvelle famille d'adresses ayant pour nom `AF_INET6` et dont la valeur peut varier d'une implémentation à l'autre, a été définie (dans `sys/socket.h`). Également, une nouvelle famille de protocoles ayant pour nom `PF_INET6` a été définie (dans `sys/socket.h`). En principe, on doit avoir :

```
#define PF_INET6 AF_INET6
```

La structure de données destinée à contenir une adresse IPv6 est définie comme suit (dans `netinet/in.h`) :

```
struct in6_addr {
   uint8_t s6_addr[16];
};
```

les octets constituant l'adresse étant rangés comme d'habitude dans l'ordre réseau (*network byte order*).

La structure de données IPv6 `struct sockaddr_in6`, est équivalente à la structure `struct sockaddr_in` d'IPv4. Elle est définie comme suit (dans `netinet/in.h`) pour les systèmes dérivés d'UNIX 4.3BSD :

```
struct sockaddr_in6 {
   sa_family_t sin6_family;    /* AF_INET6 */
   in_port_t sin6_port;        /* numéro de port */
```

```
   uint32_t sin6_flowinfo;    /* identificateur de flux */
   struct in6_addr sin6_addr; /* adresse IPv6 */
   uint32_t sin6_scope_id;    /* ensemble d'interfaces correspondant
                               * à la portée de l'adresse */
};
```

Il faut noter que cette structure a une longueur de 28 octets, et est donc plus grande que le type générique `struct sockaddr`. Il n'est donc plus possible de réserver une `struct sockaddr` si la valeur à stocker peut être une `struct sockaddr_in6`. Afin de faciliter la tâche des implémenteurs, une nouvelle structure de données, `struct sockaddr_storage`, a été définie. Celle-ci est de taille suffisante afin de pouvoir prendre en compte tous les protocoles supportés et alignée de telle sorte que les conversions de type entre pointeurs vers les structures de données d'adresse des protocoles supportés et pointeurs vers elle-même n'engendrent pas de problèmes d'alignement. Un exemple d'utilisation pourrait être le suivant :

```
struct sockaddr_storage ss;

struct sockaddr_in *sin = (struct sockaddr_in *) &ss;
struct sockaddr_in6 *sin6 = (struct sockaddr_in6 *) &ss;
```

Dans la version 4.4 d'UNIX BSD, la longueur du champ `sin6_family` est passée de 2 octets à 1 octet. L'octet ainsi récupéré contient la taille de la structure `sockaddr_in6` et sert à effectuer correctement la conversion de type vers la structure de données générique `sockaddr` utilisée par bon nombre de primitives de l'interface socket.

La macro-définition `SIN6_LEN`, présente dans toute implémentation 4.4BSD, permet alors de distinguer les versions. Les autres champs restant inchangés, cette structure est presque identique à celle de la précédente version :

```
#define SIN6_LEN

struct sockaddr_in6 {
   u_int8_t sin6_len;         /* la longueur de cette structure */
   sa_family_t sin6_family;   /* AF_INET6 */
   in_port_t sin6_port;       /* numéro de port */
   uint32_t sin6_flowinfo;    /* identificateur de flux */
   struct in6_addr sin6_addr; /* adresse IPv6 */
   uint32_t sin6_scope_id;    /* ensemble d'interfaces correspondant
                               * à la portée de l'adresse */
};
```

Si le champ `sin6_len` existe (ce qui est testable par le fait que le symbole `SIN6_LEN` est défini), il doit être initialisé par la taille de la structure `sockaddr_in6`.

On notera la présence de deux nouveaux champs (ils n'ont pas d'équivalents dans la structure `sockaddr_in`) dans la structure de données `sockaddr_in6`, les champs `sin6_flowinfo` et `sin6_scope_id`. Le premier, en réalité structuré, est décrit dans le RFC 2460 et Identificateur de flux. Le second désigne un ensemble d'interfaces en adéquation avec la portée de l'adresse contenue dans le champ `sin6_addr`. Par exemple, si l'adresse en question est de type lien local, le champ `sin6_scope_id` devrait être un index d'interface.

C) L'interface socket

a) *From Livre IPv6*

La création d'une socket se fait comme auparavant en appelant la primitive socket. La distinction entre les protocoles IPv4 et IPv6 se fait sur la valeur du premier argument passé à `socket`, à savoir la famille d'adresses (ou de protocoles), c'est-à-dire ici `PF_INET` ou `PF_INET6`. Par exemple, si on veut créer un socket IPv4/UDP, on écrira :

```
sock = socket(PF_INET, SOCK_DGRAM, 0);
```

tandis qu'une création de socket IPv6/UDP se fera ainsi :

```
sock = socket(PF_INET6, SOCK_DGRAM, 0);
```

Une erreur de programmation classique consiste à utiliser `AF_INET` à la place de `PF_INET`. Cela n'a pas d'effet en général car rares sont les systèmes pour lesquels ces deux constantes diffèrent. Pour éviter en IPv6 des problèmes liés à cette erreur, il est demandé que les deux constantes `PF_INET6` et `AF_INET6` soient identiques.

Quant aux autres primitives constituant l'interface socket, leur syntaxe reste inchangée. Il faut simplement leur fournir des adresses IPv6, en l'occurrence des pointeurs vers des structures de type `struct sockaddr_in6` au préalable convertis en des pointeurs vers des structures génériques de type `struct sockaddr`.

Donnons pour mémoire une liste des primitives les plus importantes :

```
bind()     connect()      sendmsg()
sendto()   accept()       recvfrom()
recvmsg()  getsockname()  getpeername()
```

D) L'adresse "wildcard"

Lors du nommage d'une socket via la primitive `bind`, il arrive fréquemment qu'une application (par exemple un serveur TCP) laisse au système la détermination de l'adresse source pour elle. En IPv4, pour ce faire, elle passe à bind une structure `sockaddr_in` avec le champ `sin_addr.s_addr` ayant pour valeur la constante `INADDR_ANY`, constante définie dans le fichier `netinet/in.h`.

En IPv6, il y a deux manières de faire cela, à cause des règles du langage C sur les initialisations et affectations de structures. La première est d'initialiser une structure de type `struct in6_addr` par la constante `IN6ADDR_ANY_INIT` :

```
struct in6_addr any_addr = IN6ADDR_ANY_INIT;
```

Attention, ceci ne peut se faire qu'au moment de la déclaration. Par exemple le code qui suit est incorrect (en C il est interdit d'affecter une constante complexe à une structure) :

```
struct sockaddr_in6 sin6;
```

```
sin6.sin6_addr = IN6ADDR_ANY_INIT; /* erreur de syntaxe !! */
```

La seconde manière utilise une variable globale :

```
extern const struct in6_addr in6addr_any;
struct sockaddr_in6 sin6;

sin6.sin6_addr = in6addr_any;
```

Cette méthode n'est pas possible dans une déclaration de variable globale ou statique.

La constante `IN6ADDR_ANY_INIT` et la variable `in6addr_any` sont toutes deux définies dans le fichier `netinet/in.h`.

E) L'adresse de bouclage

En IPv4, c'est la constante `INADDR_LOOPBACK`. En IPv6, de manière tout à fait similaire à l'adresse "wildcard", il y a deux façons d'affecter cette adresse. Ceci peut se faire au moment de la déclaration avec la constante `IN6ADDR_LOOPBACK_INIT` :

```
struct in6_addr loopback_addr = IN6ADDR_LOOPBACK_INIT;
```

ou via la variable globale `in6addr_loopback` :

```
extern const struct in6_addr in6addr_loopback;
struct sockaddr_in6 sin6;
sin6.sin6_addr = in6addr_loopback;
```

Cette constante et cette variable sont définies dans le fichier `netinet/in.h`.

F) Les primitives de conversion entre noms et adresses

Les primitives `gethostbyname`, `gethostbyaddr`, `getservbyname` et `getservbyport` ont été remplacées par les deux primitives indépendantes de la famille d'adresses et normalisées par la RFC 3493 `getaddrinfo` et `getnameinfo` :

```
#include <sys/socket.h>
#include <netdb.h>

int
getaddrinfo(const char *nodename, const char *servname,
const struct addrinfo *hints, struct addrinfo **res);

void freeaddrinfo(struct addrinfo *res);

const char *gai_strerror(int errcode);
```

Le type `struct addrinfo` est défini comme suit :

```
struct addrinfo {
```

```
   int ai_flags;             /* AI_PASSIVE, AI_CANONNAME, ... */
   int ai_family;            /* PF_xxx */
   int ai_socktype;          /* SOCK_xxx */
   int ai_protocol;          /* 0 ou IPPROTO_xxx pour IPv4 et IPv6 */
   size_t ai_addrlen;        /* la taille de l'adresse binaire ai_addr */
   char *ai_canonname;       /* le nom complétement qualifié */
   struct sockaddr *ai_addr; /* l'adresse binaire */
   struct addrinfo *ai_next; /* la structure suivante dans la liste chaînée */
};
```

`getaddrinfo` prend en entrée le nom d'une machine (`nodename`) et le nom d'un service (`servname`). S'il n'y a pas d'erreur, `getaddrinfo` rend 0 et res pointe sur une liste dynamiquement allouée de `struct addrinfo`. Chaque élément de cette liste contient la description et l'adresse d'une `struct sockaddr` initialisée pour fournir l'accès au service servname sur nodename. Les champs `ai_family`, `ai_socktype` et `ai_protocol` ont la valeur utilisable dans l'appel système socket.

Lorsque la liste de résultat n'est plus nécessaire, la mémoire allouée peut être libérée par la primitive `freeaddrinfo`. En cas d'erreur, `getaddrinfo` rend un code d'erreur non nul qui peut être imprimé par la fonction `gai_strerror`.

`getaddrinfo` peut donner des réponses de la famille d'adresses IPv4 ou IPv6, et des réponses pour les protocoles connectés ou non (`ai_socktype` peut valoir `SOCK_DGRAM` ou `SOCK_STREAM`). L'argument `hints` permet de choisir les réponses souhaitées. Un argument égal à `NULL` signifie que la liste des réponses doit contenir toutes les adresses et tous les protocoles. Sinon `hints` doit pointer sur une structure dont les champs `ai_family`, `ai_socktype` et `ai_protocol` définissent les types de résultat attendus. Une valeur de `PF_UNSPEC` du champ `ai_family` signifie que toutes les familles d'adresse (IPv4 et IPv6) sont admises, un `0` dans les champs `ai_socktype` (resp. `ai_protocol`) signifie que tous les types de socket (resp. protocole) sont admis. Le champ `ai_flags` permet de préciser des options supplémentaires.

L'argument `servname` peut être le nom d'un service ou un nombre décimal. De même, l'argument `nodename` peut être un nom (au format DNS habituel) ou une adresse sous forme numérique IPv4 ou IPv6 (si `ai_flags` contient le bit `AI_NUMERICHOST`, `nodename` doit être sous forme numérique et aucun appel au serveur de nom n'est fait). De plus l'un ou l'autre des arguments `servname` et `nodename` peut être un pointeur `NULL`, mais pas tous les deux. Si `servname` est `NULL`, le champ `port` des réponses ne sera pas initialisé (il restera égal à `0`). Si `nodename` est `NULL`, l'adresse réseau dans les réponses est mis à "non initialisé" (`INADDR_ANY` en IPv4, `IN6ADDR_ANY_INIT` en IPv6) si `ai_flags` contient le bit `AI_PASSIVE`, et à l'adresse de "loopback" (`INADDR_LOOPBACK` ou `IN6ADDR_LOOPBACK_INIT`) sinon. Le cas `AI_PASSIVE` sert donc à obtenir des réponses utilisables par un programme serveur dans un *bind* pour recevoir des requêtes. Enfin si le bit `AI_CANONNAME` est positionné, le champ `ai_canonname` de la réponse contient le nom canonique de `nodename`.

La primitive `getnameinfo` remplace les primitives `gethostbyaddr` et `getservbyport`. Elle effectue la traduction d'une adresse vers un nom :

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr *sa, socklen_t salen,
char *host, size_t hostlen,
char *serv, size_t servlen, int flags);
```

En entrée l'argument sa pointe vers une structure d'adresse générique (de type `sockaddr_in` ou `sockaddr_in6`) et salen contient sa longueur. Le champ `host` (resp. `serv`) doit pointer sur une zone de longueur `hostlen` (resp. `servlen`) caractères. `getnameinfo` retourne la valeur `0` si tout est correct et un code d'erreur non nul si une erreur est détectée. S'il n'y a pas d'erreur, le champ `host` (resp. `serv`) reçoit en sortie le nom de la machine (resp. du service) correspondant. Les arguments `host` et `serv` peuvent être `NULL` si la réponse est inutile. Deux constantes sont définies pour permettre de réserver des zones de réponses de longueur raisonnable :

```
# define NI_MAXHOST 1025
# define NI_MAXSERV 32
```

Le champ `flags` permet de modifier la réponse : si flags contient le bit `NI_NUMERICHOST` (resp. `NI_NUMERICSERV`) la réponse sera l'adresse et non le nom de la machine (resp. le numéro et non le nom du service) ; si on ne sait pas trouver dans le serveur de nom le nom de la machine, `getnameinfo` rendra une erreur si le bit `NI_NAMEREQD` est positionné et l'adresse numérique sinon ; le bit `NI_DGRAM` indique si le service est sur UDP et non sur TCP.

G) Les fonctions de conversion numériques d'adresses

Elles sont l'analogue des fonctions `inet_addr` et `inet_ntoa` d'IPv4, la seule véritable différence étant qu'elles ont un argument précisant la famille d'adresse et peuvent donc aussi bien convertir les adresses IPv4 que les adresses IPv6. Comme la plupart des programmes manipulent des `struct sockaddr*`, il est souvent préférable d'utiliser les fonctions `getaddrinfo` et `getnameinfo`, au besoin avec le flag `AI_NUMERICHOST`.

```
#include <sys/socket.h>
#include <arpa/inet.h>

int
inet_pton(af, src, dst)
int af;              /* AF_INET ou AF_INET6 */
const char *src;     /* l'adresse (chaine de caract.) à traiter */
void *dst;           /* le tampon où est rangé le résultat */

char *
inet_ntop(af, src, dst, size)
int af;              /* AF_INET ou AF_INET6 */
const void *src;     /* l'adresse binaire à traiter */
char *dst;           /* le tampon où est rangé le résultat */
size_t size;         /* la taille de ce tampon */
```

La primitive `inet_pton` convertit une adresse textuelle en sa forme binaire. Elle retourne `1` lorsque la conversion a été réussie, `0` si la chaine de caractères qui lui a été fournie n'est pas une adresse valide et `-1` en cas d'erreur, c'est-à-dire lorsque la famille d'adresses (premier argument) n'est pas supportée. Actuellement, les deux seules familles d'adresses supportées sont `AF_INET` et `AF_INET6`.

La primitive duale `inet_ntop` convertit une adresse sous forme binaire en sa forme textuelle. Le troisième argument est un tampon destiné à recevoir le résultat de la conversion. Il doit être d'une taille suffisante, à

savoir 16 octets pour les adresses IPv4 et 46 octets pour les adresses IPv6. Ces deux tailles sont définies dans le fichier `netinet/in.h` :

```
#define INET_ADDRSTRLEN 16
#define INET6_ADDRSTRLEN 46
```

Si la conversion est réussie, `inet_ntop` retourne un pointeur vers le tampon où est rangé le résultat de la conversion. Dans le cas contraire, `inet_ntop` retourne le pointeur nul, ce qui se produit soit lorsque la famille d'adresses n'est pas reconnue, soit lorsque la taille du tampon est insuffisante.

2) La commande haah (host-address-address-host)

L'exemple proposé n'est autre qu'une sorte de nslookup (très) simplifié. Si par exemple on lui donne en argument une adresse numérique (IPv4 ou IPv6), il imprime le nom complètement qualifié correspondant lorsque la requête DNS aboutit. L'extrait de session qui suit illustre l'utilisation de cette commande.

```
$ haah bernays
Canonical name:
bernays.ipv6.logique.jussieu.fr
Adresses:
2001:660:101:101:200:f8ff:fe31:17ec
3ffe:304:101:1:200:f8ff:fe31:17ec
$ haah 134.157.19.71
Canonical name:
bernays.logique.jussieu.fr
Adresses:
134.157.19.71
$
```

Le programme réalisant la commande `haah` ne présente aucune difficulté. C'est une simple application des primitives précédemment décrites.

```
 1| #include <stdio.h> </tt>
 2|
 3| #include <string.h>
 4| #include <errno.h>
 5| #include <sys/types.h>
 6| #include <sys/socket.h>
 7| #include <netinet/in.h>
 8| #include <netdb.h>
 9| #include <arpa/inet.h>
10|
11|
12| int main(int argc, char **argv)
13| {
14|    int ret;
15|    struct addrinfo *res, *ptr;
16|    struct addrinfo hints = {
17|       AI_CANONNAME,
18|       PF_UNSPEC,
19|       SOCK_STREAM,
20|       0,
21|       0,
22|       NULL,
23|       NULL,
24|       NULL
```

```
25|    };
26|
27|    if (argc != 2) {
28|       fprintf(stderr, "%s: usage: %s host | addr.\n", *argv, *argv);
29|       exit(1);
30|    }
31|    ret = getaddrinfo(argv[1], NULL, &hints, &res);
32|    if (ret) {
33|       fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(ret));
34|       exit(1);
35|    }
37|    for (ptr = res; ptr; ptr = ptr->ai_next) {
38|       if (ptr->ai_canonname)
39|          fprintf(stdout,"Canonical name:\n%s\nAdresses:\n", ptr->ai_canonname);
40|       switch (ptr->ai_family) {
41|          case AF_INET:
42|          {
43|             char dst[INET_ADDRSTRLEN];
44|             struct in_addr *src = &((struct sockaddr_in *) ptr->ai_addr)->sin_addr;
45|
46|             if(!inet_ntop(AF_INET, (const void *) src, dst, sizeof(dst))) {
47|                fprintf(stderr, "inet_ntop: %s\n", strerror(errno));
48|                break;
49|             }
50|             fprintf(stdout, "%s\n", dst);
51|             break;
52|          }
53|          case AF_INET6:
54|          {
55|             char dst[INET6_ADDRSTRLEN];
56|             struct in6_addr *src=&((struct sockaddr_in6 *)ptr->ai_addr)->sin6_addr;
57|
58|             if (!inet_ntop(AF_INET6, (const void *) src, dst, sizeof(dst))) {
59|                fprintf(stderr, "inet_ntop: %s\n", strerror(errno));
60|                break;
61|             }
62|             fprintf(stdout, "%s\n", dst);
63|             break;
64|          }
65|          default:
66|             fprintf(stderr, "getaddrinfo: %s\n", strerror(EAFNOSUPPORT));
67|       }
68|    }
69|    freeaddrinfo(res);
70|    exit(0);
71| }
```

3) Exemple de client/serveur TCP

A) Vue d'ensemble

Le client/serveur choisi est particulièrement simple quant à sa fonction de service, de façon à privilégier l'aspect réseau dans la présentation. Il calcule le nombre d'utilisateurs connectés sur une machine donnée. Plus précisément :

```
$ nbus bernays
3 user(s) logged on bernays
$
```

Il se compose de cinq fichiers sources :

- `nbus.h` le fichier d'en-tête commun aux fichiers sources du client et du serveur
- `nbus.c` le fichier source principal du client
- `open_conn.c` le fichier source qui gère les connexions du côté client
- `nbusd.c` le fichier source du serveur propre au service
- `serv_daemon.c` le fichier source qui gère les connexions du côté serveur

Le client, prend en argument un nom de machine (ou une adresse numérique), le convertit en une adresse IPv4 ou IPv6 et ainsi envoie ses requêtes à un serveur.

Le fichier source du serveur, selon les options de compilation, génère un code IPv6 ou un code IPv4. Dans le premier cas, le serveur est à même de satisfaire les requêtes d'un client IPv6 mais aussi d'un client IPv4. Dans le second cas, seuls les clients IPv4 pourront être pris en compte.

Il faut noter qu'il existe deux modes de fonctionnement pour une machine à double pile IPv4 et IPv6, la figure Le client/serveur nbus résume la situation. Soit les espaces d'adresses sont disjoints, et dans ce cas il faut deux serveurs, un qui écoute les requêtes IPv4 (`nbus4d`) et un qui écoute les requêtes IPv6 (`nbus6d`) (ou un seul serveur, `nbusd`, avec deux sockets IPv4 et IPv6 séparées). Soit l'espace d'adresse IPv4 est inclus dans l'espace IPv6 et dans ce cas il suffit d'un serveur IPv6 (`nbus6d`) qui recevra les requêtes venant en IPv4 comme une requête IPv6 venant d'une adresse "IPv4 mappée".

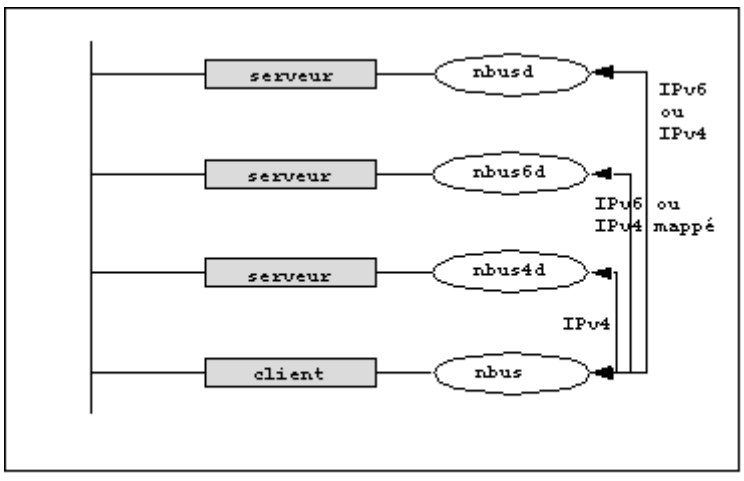


Figure 15-2. Le client/serveur nbus

Suivant les systèmes le mode de fonctionnement est prédéfini, configurable globalement dans le noyeau, ou configurable séparément pour chaque socket IPv6. Ainsi en FreeBSD le mode par défaut est "espace partagé" (choix modifiable par "`sysctl -w net.inet6.ip6.v6only=1`") et modifiable pour chaque socket, en précisant tcp4, tcp6 ou tcp46 dans le fichier de configuration d'inetd, ou en utilisant "`setsockopt(fd, IPPROTO_IPV6, IPV6_V6ONLY,&val)`" dans le code du serveur.

Deux mêmes programmes serveur ne peuvent s'exécuter en même temps sur une machine : chaque serveur réserve le port nbus (par un bind) et par suite si un serveur est lancé, tout autre serveur échouera avec une erreur "port en service" (`EADDRINUSE`). Cela peut être aussi vrai entre les deux types de serveurs, nbus4d et nbus6d sur une machine "double pile avec espace partagé", car les espaces de service TCP et UDP sont communs à IPv4 et IPv6 et un port en service l'est aussi bien en IPv4 qu'en IPv6 ; simplement si le port correspond à une socket `PF_INET`, une requête de connexion IPv6 sera rejetée avec une erreur "port inaccessible". Dans la réalité le comportement est plus complexe. Même en mode "double pile avec espace partagé", on peut avoir deux sockets, l'une PF_INET et l'autre PF_INET6, comme le montre le programme nbusd.

Le code des différents serveurs est très semblable, le choix est fait en donnant une option à la compilation : `nbusd` par défaut, `nbus4d` si on ajoute l'option -DIPV4,et `nbus6d` si on ajoute l'option `-DIPV6`.

Le fichier d'en-tête `nbus.h` ne contient que le nom du service correspondant.

```
#define SERVICE "nbus"
```

Ainsi doit-on trouver, par exemple, dans le fichier `/etc/services` des machines concernées, la ligne suivante :

```
nbus 20000/tcp
```

Le programme client nbus établit tout d'abord une connexion TCP avec la machine cible (donnée en argument à `nbus`) via la fonction `open_conn` (cette fonction sera décrite ci-après). Il lit ensuite un entier court, résultat du calcul du nombre d'utilisateurs connectés sur la machine cible, puis l'imprime.

On notera la présence de la macro `ntohs` rendue nécessaire du fait des représentations différentes des entiers selon les machines.

```
 1| #include <stdio.h>
 2| #include <unistd.h>
 3|
 4| #include "nbus.h"
 5|
 6| extern open_conn(char *, char *);
 7|
 8| int main(int argc, char **argv)
 9| {
10|    int sock;
11|    short nu;
12|
13|    if (argc != 2) {
14|       fprintf(stderr, "Usage: %s host\n", argv[0]);
15|       exit(1);
16|    }
17|    if ((sock = open_conn(argv[1], SERVICE)) < 0)
18|       exit(1);
19|    read(sock, (char *) &nu, sizeof(nu));
```

```
20|    nu = ntohs(nu);
21|    if (nu == -1) {
22|       fprintf(stderr, "Can't read \"utmp\" on %s\n", argv[1]);
23|       exit(1);
24|    }
25|    if (nu) {
26|       fprintf(stdout, "%d user(s) logged on %s\n", nu, argv[1]);
27|       exit(0);
28|    }
29|    fprintf(stdout, "Nobody on %s\n", argv[1]);
30|    exit(0);
31| }
```

Le serveur `nbusd`, quant à lui, lance en tâche de fond un processus "démon" qui exécutera la fonction de service nbus (lignes 28 à 60) à chaque requête d'un client. Ce processus démon est réalisé par la fonction `serv_daemon`.

```
 1| #include <stdio.h>
 2| #include <unistd.h>
 3| #include <fcntl.h>
 4| #include <utmp.h>
 5| #include <sys/socket.h>
 6|
 7| #include "nbus.h"
 8|
 9|
10| #if defined(IPV6)
11| #define FAMILY PF_INET6
12| #elif defined(IPV4)
13| #define FAMILY PF_INET
14| #else
15| #define FAMILY PF_UNSPEC
16| #endif
17|
18| extern serv_daemon(int, char *, void (*)(int), char *);
19|
20| void nbus(int);
21|
22| int main(void)
23| {
24|    serv_daemon(FAMILY, SERVICE, nbus, NULL);
25|   exit(0);
26| }
27|
28| void nbus(int sock)
29| {
30|    short nu = -1;
31| #ifdef USER_PROCESS /* Solaris/Linux, use getutent */
32|    struct utmp *up;
33|
34|    up = getutent();
35|    if (up != NULL) {
36|       for (nu = 0; up != NULL; up = getutent())
37|       if (up->ut_type == USER_PROCESS)
38|       nu++;
39|    }
40|    endutent();
41| #else /* *BSD read directly utmp file */
42| #ifndef UTMP
43| #define UTMP "/var/run/utmp" /* for FreeBSD/NetBSD */
44| #endif
```

```
45|
46|    struct utmp ut;
47|    int fd;
48|
49|   if ((fd = open(UTMP, O_RDONLY)) >= 0) {
50|       nu = 0;
51|       while (read(fd, (char *) &ut, sizeof(ut)) == sizeof(ut))
52|       if (ut.ut_name[0])
53|       nu++;
54|    }
55|    close(fd);
56| #endif
57|    nu = htons(nu);
58|    write(sock, (char *) &nu, sizeof(nu));
59|    return;
60| }
```

B) L'établissement d'une connexion TCP, côté client

Comme on l'a vu plus haut dans le code du client `nbus.c`, l'établissement de la connexion TCP se fait au moyen de la fonction `open_conn`. Cette fonction prend en premier argument le nom de la machine avec laquelle on va établir la connexion TCP et en deuxième argument le nom du service tel qu'il apparaît dans le fichier `/etc/services`. La valeur retournée par `open_conn` est soit `-1` en cas d'erreur, soit le descripteur associé à la socket réalisant la connexion. La figure Algorithme du client visualise l'algorithme employé.

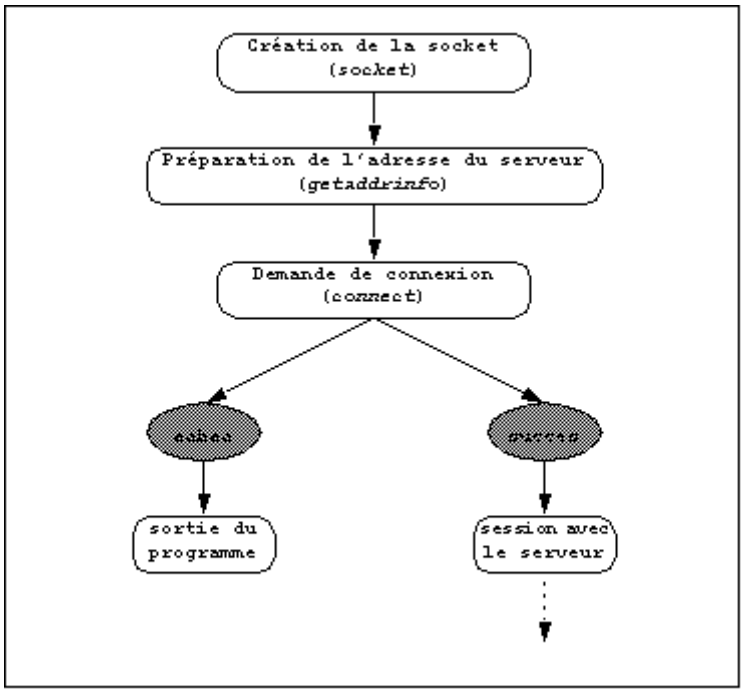


Figure 15-3. Algorithme du client

La première étape sera la construction de l'adresse de la socket distante, ceci via la primitive getaddrinfo. On remarquera que le champ `ai_family` de la structure `hints` a été initialisé à la valeur `PF_UNSPEC`, ce qui signifie que, suivant que l'on donne en argument à la commande nbus un nom de machine (ou une adresse numérique) IPv4 ou IPv6, on travaillera avec une socket soit dans la famille des protocoles `PF_INET`, soit dans la famille des protocoles `PF_INET6`. Si on avait fait le choix de forcer la famille des protocoles à la valeur `PF_INET6`, il aurait fallu initialiser les champs `ai_flags` et `ai_family` respectivement aux valeurs

`AI_V4MAPPED` et `PF_INET6` (les adresses IPv4 seraient dans ce cas "mappées"). Si, comme dans l'exemple proposé, on ne fait pas ce choix, on notera qu'il n'y a aucune différence entre le code IPv4 et le code IPv6.

Ensuite, après avoir créé une socket, on la connecte au serveur de la machine cible (primitive `connect`). Là encore, le code est le même pour IPv4 et IPv6. Remarquons que `getaddrinfo` peut avoir rendu plusieurs valeurs. Un programme plus évolué devrait essayer de se connecter à chaque adresse rendue jusqu'à obtenir une réponse.

```
 1| #include <stdio.h>
 2| #include <unistd.h>
 3| #include <sys/socket.h>
 4| #include <netdb.h>
 5|
 6| int open_conn(char *host, char *serv)
 7| {
 8|    int sock, ecode;
 9|    struct addrinfo *res;
10|    struct addrinfo hints = {
11|       0,
12|       PF_UNSPEC,
13|       SOCK_STREAM,
14|       0,
15|       0,
16|       NULL,
17|       NULL,
18|       NULL
19|    };
20|
21|    ecode = getaddrinfo(host, serv, &hints, &res);
22|
23|    if (ecode) {
24|       fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(ecode));
25|       exit(1);
26|    }
27|
28|    if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0) {
29|       freeaddrinfo(res);
30|       perror("socket");
31|       return -1;
32|    }
33|
34|   if (connect(sock, res->ai_addr, res->ai_addrlen) < 0) {
35|       close(sock);
36|       freeaddrinfo(res);
37|       perror("connect");
38|       return -1;
39|    }
40|    freeaddrinfo(res);
41|    return sock;
42| }
```

C) Le serveur

Le serveur proprement dit est réalisé par une unique fonction. C'est la fonction `serv_daemon` qui a quatre arguments. Le premier est la famille d'adresse gérée, `PF_INET` ou `PF_INET6`, ou `PF_UNSPEC` pour écouter dans les deux familles. Le deuxième est, comme dans le cas de `open_conn`, le nom du service tel qu'il apparaît dans le fichier `/etc/services`. Le troisième argument est un pointeur vers la fonction de service dont le type `(int(*)(void))` sera commenté ultérieurement. Enfin le dernier argument est le nom passé au démon `syslogd` lors de l'impression des messages d'erreur (ligne See). Si le dernier argument est le pointeur nul, ce nom est par défaut le nom du service. Un aperçu du déroulement de la fonction `serv_daemon` est donné par la figure Algorithme du serveur.

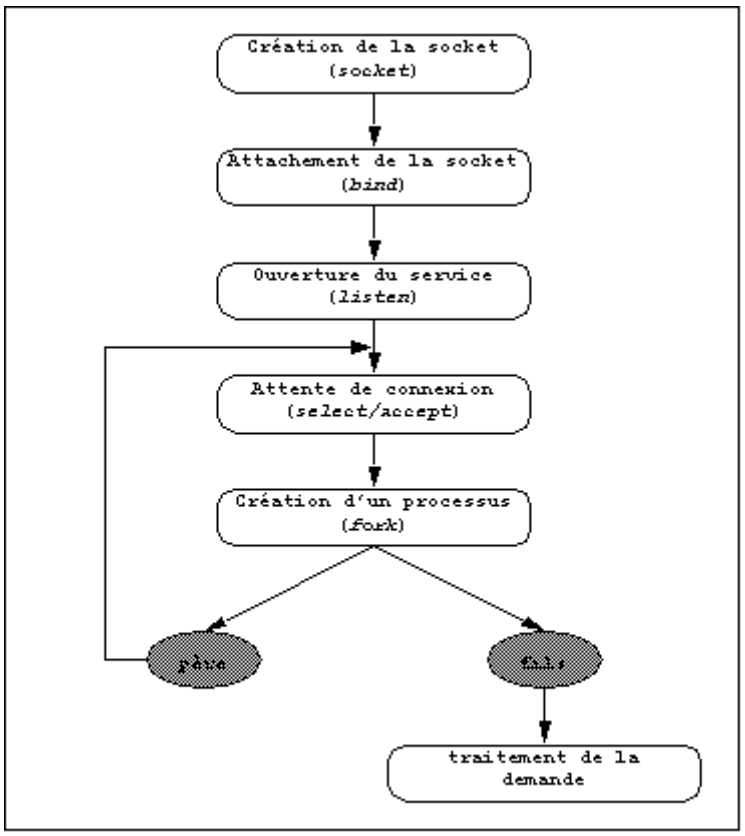


Figure 15-4 . Algorithme du serveur

```
 1| #include <stdio.h>
 2| #include <stdlib.h>
 3| #include <unistd.h>
 4| #include <errno.h>
 5| #include <sys/socket.h>
 6| #include <netdb.h>
 7| #include <signal.h>
 8| #include <syslog.h>
 9| #include <sys/select.h>
10| #include <sys/wait.h>
11|
12| static void reap_child(int);
13|
14|  void serv_daemon(int family, char *serv, void (*serv_funct)(int), char
*serv_logname)
```

```
15| {
16|    int sock[2], ecode, n = 0;
17|    struct addrinfo *res, *rres, hints;
18|
19|    memset(&hints, 0, sizeof hints) ;
20|    hints.ai_flags = AI_PASSIVE;
21|    hints.ai_socktype = SOCK_STREAM;
22|    hints.ai_family = family;
23|    ecode = getaddrinfo(NULL, serv, &hints, &rres);
24|    if (ecode) {
25|       fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(ecode));
26|       exit(1);
27|    }
28|    for (res = rres; res; res = res->ai_next) {
29|       if (n == 2) { /* au plus 2 : anyaddr PF_INET et anyaddr PF_INET6 */
30|         fprintf(stderr, "erreur interne: trop d'adresses\n");
31|         exit(1);
32|       }
33|       sock[n] = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
34|       if (sock[n] < 0) {
35|          perror("socket");
36|          exit(1);
37|       }
38|       if (bind(sock[n], res->ai_addr, res->ai_addrlen) < 0) {
39|          perror("bind");
40|          exit(1);
41|       }
42|       listen(sock[n], SOMAXCONN);
43|       n++;
44|       #ifdef __linux__
45|                      /* En Linux, utiliser seulement la premiere
46|                       * reponse, sinon on a un conflit sur bind */
47|       break;
48|       #endif
49|    }
50|    freeaddrinfo(rres);
```

La première étape consiste en la préparation de l'adresse de la socket d'écoute du serveur en faisant appel, comme d'habitude, à la primitive `getaddrinfo`. Comme il s'agit d'une socket d'écoute, le champ `ai_flags` de la structure `hints` a été initialisé à la valeur `AI_PASSIVE` tandis que le champ `ai_family` de cette même structure a lui été initialisé à la valeur passé en argument à l'appel.

Pour chaque réponse de getaddrinfo (une seule si la famille est définie, deux si la famille est `PF_UNSPEC`), on crée une socket d'écoute, puis effectue son attachement en faisant appel à la primitive `bind`. La socket d'écoute créée et attachée à une adresse, le serveur doit signifier au système qu'il est prêt à accepter les demandes de connexions. C'est l'objet de la primitive listen dont le deuxième argument `SOMAXCONN` (macro-définition que l'on trouve dans le fichier `sys/socket.h`) est le nombre maximal de connexions pendantes.

Dans les lignes qui suivent (51 à 58), le processus est détaché du terminal de contrôle et est lancé en tâche de fond (processus "démon"), sauf en Solaris, le code étant trop différent, si le fichier source n'est pas compilé avec l'option `-DDEBUG` (dans le cas contraire, cela peut être bien utile lors de la phase de mise au point).

```
51|    #ifndef DEBUG
```

```
52|    #ifndef sun /* no daemon function in Solaris */
53|    if (daemon(0, 0) < 0) {
54|       perror("daemon");
55|       exit(1);
56|    }
57|    #endif
58|    #endif
```

Dans la boucle sans fin (lignes See à See), on attend les connexions. Comme il peut y avoir plusieurs sockets actives, on utilise select (lignes See à See) pour attendre sur toutes les sockets ouverts. Au retour de `select`, on teste les sockets qui ont des connexions pendantes (ligne See), chaque connexion pendante dans la file d'attente associée à la socket d'écoute est extraite par le serveur et le circuit virtuel avec la socket du client est établi via la création d'une nouvelle socket. Ce travail est effectué par la primitive `accept` dont la valeur de retour est le descripteur d'E/S de la socket nouvellement créée. Ensuite le serveur crée un processus fils (ligne See) qui exécutera la fonction de service à laquelle on passe en argument la valeur retournée par la primitive accept.

Il faut également veiller à ce que chaque processus fils, lors de sa terminaison (qui se produit à la fin de l'exécution de la fonction de service), ne devienne un processus "zombie", ce qui à terme peut provoquer une saturation de la table des processus. Pour cela il faut, sous UNIX BSD, capter le signal `SIGCHLD` (ligne See , fonction reap_child). Notons qu'en SYSTEM V, on pourrait simplement ignorer le signal `SIGCLD`.

```
59|       signal(SIGCHLD, reap_child);
60|       if (!serv_logname)
61|          serv_logname = serv;
62|       openlog(serv_logname, LOG_PID | LOG_CONS, LOG_USER);
63|       for (;;) {
64|          int a, f, len, fd, m = -1;
65|          struct sockaddr_storage from;
66|          fd_set fdset;
67|
68|          FD_ZERO(&fdset);
69|          for (fd = 0; fd < n; fd++) {
70|             if (m < sock[fd])
71|                m = sock[fd];
72|             FD_SET(sock[fd], &fdset);
73|          }
74|          ecode = select(m+1, &fdset, NULL, NULL, NULL);
75|          if (ecode < 0)
76|             syslog(LOG_ERR, "%s: select: %m", serv);
77|          if (ecode <= 0)
78|             break;
79|          for (fd = 0; fd < n; fd++) {
80|            if (FD_ISSET(sock[fd], &fdset)) {
81|               len = sizeof from;
82|               a = accept(sock[fd], (struct sockaddr *)&from, &len);
83|               if (a < 0) {
84|                  if (errno != EINTR)
85|                   syslog(LOG_ERR, "%s: accept: %m", serv);
86|                   continue;
87|               }
88|               f = fork();
89|               if (f == 0) {
80|                  /* Par correction, il faudrait fermer dans le fils
90|                   * tous les descripteurs hérités (i.e. tous sauf a) */
91|                  serv_funct(a);
92|                  exit(0);
```

```
 93|                }
 94|                close(a);
 95|                if (f == -1) {
 96|                    syslog(LOG_ERR, "%s: fork: %m", serv);
 97|                    continue;
 98|                }
 99|            }
100|        }
101|    }
102| }
103|
104| static void reap_child(int sig)
105| {
106|    int status;
107|
108|    while (wait3(&status, WNOHANG, NULL) > 0);
109| }
```

Une dernière remarque toujours à propos de la similitude des codes IPv6 et IPv4 : la seule différence dans le code de `serv_daemon` entre IPv4 et IPv6 est la valeur du premier argument, qui définit la famille d'adresses écoutée.

4) Mini-ping

A) Description

La commande proposée est une version très simplifiée de `ping6`. Néanmoins, cela permettra de comprendre l'essentiel du fonctionnement de cette commande. Son principe est le suivant, on émet un paquet ICMPv6 du type ECHO_REQUEST et on active une temporisation. Si, le délai étant expiré, on n'a pas reçu de paquet ICMPv6 de type ECHO_REPLY en provenance de la machine cible, on imprime un message d'erreur. Dans le cas contraire, on imprime le nom de la machine émettrice de l'ECHO_REPLY. Par exemple, si le nom donné à cette commande est one_ping6 :

```
$ one_ping6 peirce
Sending ECHO REQUEST to: peirce.ipv6.logique.jussieu.fr
Waiting for answer (timeout = 5s)...
Got answer from 2001:660:101:201:200:f8ff:fe31:1942 (seq = 0)
$
```

Remarque : ICMP étant un protocole non fiable, il peut arriver qu'un premier paquet soit perdu, par exemple à cause du temps passé à exécuter le protocole de "recherche de voisins". Il suffit en général de relancer la commande pour que la réponse apparaisse la seconde fois.

one_ping6 accepte les options suivantes :

- `-d données`. Ces données seront incluses dans le paquet ECHO_REQUEST.
- `-s numéro de séquence`. La valeur défaut est zéro.
- `-t durée de la temporisation`. La valeur par défaut est fixée lors de la compilation via la macro-définition TIMEOUT.

Par exemple,

```
$ one_ping6 -d 'Un petit essai' -s 12 -t 3 peirce
Sending ECHO REQUEST to: peirce.ipv6.logique.jussieu.fr
Waiting for answer (timeout = 3s)...
Got answer from 2001:660:101:201:200:f8ff:fe31:1942 (seq = 12)
with data [
Un petit essai
] (end of data)
$
```

Les sources de ce programme se composent de trois fichiers : le programme principal, le source de la fonction assurant l'émission du paquet ECHO_REQUEST et le source de la fonction ayant en charge la gestion de la temporisation et la réception du paquet ECHO_REPLY.

B) Envoi du paquet ECHO_REQUEST

Rappelons tout d'abord que le nouveau protocole ICMPv6 est une refonte presque complète d'ICMP (sur IPv4). Néanmoins, le format des paquets ECHO_REQUEST et ECHO_REPLY est inchangé excepté la valeur du champ type (cf. figure format d'un message ICMPv6 demande et réponse d'écho).

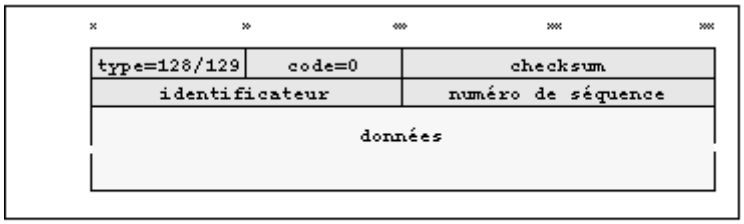


Figure 4-16. Format d'un message ICMPv6 demande et réponse d'écho

La préparation d'un paquet ECHO_REQUEST est similaire en ICMP(v4) ou ICMPv6. La seule différence est que le calcul du checksum n'est maintenant plus à la charge du programmeur mais effectué par le noyau. Plus précisément, ainsi qu'il est spécifié dans l'API "avancée", pour toutes les sockets de type `SOCK_RAW` et de protocole `IPPROTO_ICMPV6`, c'est le noyau qui doit calculer le checksum des paquets ICMPv6 sortants (dans le cas des Linux anciens, il faut activer le calcul du checksum, comme on le voit en lignes 81 à 95 du fichier ping.c).

Le paquet ICMPv6 de type ECHO_REQUEST, étant ainsi constitué, on l'expédie, via la primitive sendto à la machine cible.

```
 1| #include <stdio.h>
 2| #include <string.h>
 3| #include <sys/types.h>
 4| #include <sys/socket.h>
 5| #include <netinet/in.h>
 6| #include <netinet/ip6.h>
 7| #include <netinet/icmp6.h>
 8| #include <arpa/inet.h>
 9| #include <netdb.h>
```

```
10|
11| #ifndef MAX_DATALEN
12| #define MAX_DATALEN (1280 - sizeof(struct ip6_hdr) - sizeof(struct icmp6_hdr))
13| #endif
14|
15| static u_char buf[sizeof(struct icmp6_hdr) + MAX_DATALEN];
16|
17| int send_echo_request6(int sock, struct sockaddr_in6 *dst, uint16_t id,
18|                        uint16_t seq, char *opt_data, int opt_data_size)
19| {
20| int noc, icmp_pkt_size = sizeof(struct icmp6_hdr);
21| struct icmp6_hdr *icmp;
22|
23|    if (opt_data && opt_data_size > MAX_DATALEN) {
24|       fprintf(stderr, "send_echo_request6: too much data (%d > %d)\n",
25|       opt_data_size, MAX_DATALEN);
26|       return -1;
27|    }
28|
29|    memset((void *) buf, 0, sizeof(buf));
30|    icmp = (struct icmp6_hdr *) buf;
31|    icmp->icmp6_type = ICMP6_ECHO_REQUEST;
32|    icmp->icmp6_id = id;
33|    icmp->icmp6_seq = seq;
34|    if (opt_data) {
35|       memcpy(buf + sizeof(struct icmp6_hdr), opt_data, opt_data_size);
36|       icmp_pkt_size += opt_data_size;
37|    }
39|    noc = sendto(sock, (char *) icmp, icmp_pkt_size, 0,
39|                   (struct sockaddr *) dst, sizeof(struct sockaddr_in6));
40|    if (noc < 0) {
41|       perror("send_echo_request6: sendto");
42|       return -1;
43|    }
44|    if (noc != icmp_pkt_size) {
45|       fprintf(stderr, "send_echo_request6: wrote %d bytes, ret=%d\n",
46|               icmp_pkt_size, noc);
47|       return -1;
48|    }
49|    return 0;
50| }
```

Une dernière remarque avant de clore cette section. On a vu que l'on pouvait inclure des données dans le paquet ICMPv6 émis. La taille maximale de celles-ci a été choisie (ligne 12) pour que les paquets ne soient jamais fragmentés (le protocole IPv6 exigeant une taille de paquet minimale de 1280 octets, en-têtes comprises). Une taille plus grande serait possible, les paquets ICMP ECHO pouvant parfaitement être fragmentés.

C) La réception du paquet ECHO_REPLY

C'est la fonction wait_for_echo_reply6 qui gère la réception du paquet ECHO_REPLY. Cette fonction tout d'abord (lignes 32 à 35) utilise le mécanisme de filtrage des paquets ICMPv6, mécanisme défini dans l'API "étendue", afin que seuls les paquets ICMPv6 de type ECHO_REPLY soient reçus sur la socket d'écoute.

On trouve ensuite une boucle sans fin dont on sort soit sur réception du signal SIGALRM (armé juste avant l'entrée de la boucle à la ligne 36), c'est-à-dire lorsque le délai de temporisation (argument timeout) est expiré, soit lorsque la fonction `recv_icmp_pkt`, qui analyse tous les paquets ICMPv6 de type ECHO_REPLY reçus sur la socket d'écoute (argument sock) par l'émetteur, retourne 0, c'est-à-dire lorsque le paquet ECHO_REPLY en provenance de la machine cible a été détecté.

```
 1| #include <stdio.h>
 2| #include <unistd.h>
 3| #include <string.h>
 4| #include <sys/types.h>
 5| #include <sys/socket.h>
 6| #include <netinet/in.h>
 7| #include <netinet/ip6.h>
 8| #include <netinet/icmp6.h>
 9| #include <arpa/inet.h>
10| #include <errno.h>
11| #include <signal.h>
12| #include <setjmp.h>
13|
14| #ifndef MAX_DATALEN
15| #define MAX_DATALEN (1280 - sizeof(struct ip6_hdr) - sizeof(struct icmp6_hdr))
16| #endif
17|
18| static void on_timeout(int);
19| static int recv_icmp_pkt(int, struct sockaddr_in6 *, uint16_t, uint16_t);
20|
21| static u_char buf[sizeof(struct icmp6_hdr) + MAX_DATALEN];
22| static jmp_buf j_buf;
23|
24| void wait_for_echo_reply6(int sock, struct sockaddr_in6 *from, uint16_t id,
25|                           uint16_t seq, int timeout)
26| {
27| struct icmp6_filter filter;
28| char from_ascii[INET6_ADDRSTRLEN];
29|
30|    inet_ntop(AF_INET6, &from->sin6_addr, from_ascii, INET6_ADDRSTRLEN);
31|
32|    ICMP6_FILTER_SETBLOCKALL(&filter);
33|    ICMP6_FILTER_SETPASS(ICMP6_ECHO_REPLY, &filter);
34|    setsockopt(sock, IPPROTO_ICMPV6, ICMP6_FILTER, (const void *) &filter,
35|               sizeof(filter));
36|    signal(SIGALRM, on_timeout);
37|    alarm(timeout);
38|    for (;;) {
39|       int noc, from_len = sizeof(struct sockaddr_in6);
40|
41|       if (setjmp(j_buf) == SIGALRM) {
42|          fprintf(stderr, "No answer from %s\n", from_ascii);
43|          break;
44|       }
```

```
45|       noc = recvfrom(sock, buf, sizeof(buf), 0,
46|                       (struct sockaddr *) from, &from_len);
47|       if (noc < 0) {
48|          if (errno == EINTR)
49|             continue;
50|          perror("wait_for_echo_reply6: recvfrom");
51|          continue;
52|       }
53|       if (recv_icmp_pkt(noc, from, id, seq) == 0)
54|          break;
55|    }
56|    alarm(0);
57|    signal(SIGALRM, SIG_DFL);
58|    return;
59| }
60|
61| static void on_timeout(int sig)
62| {
63|    longjmp(j_buf, sig);
64| }
```

Contrairement à ce qui se passait en IPv4, l'entête IPv6 n'est pas incluse lors de la réception d'un paquet ICMPv6 (sauf si l'option `IP_HDRINCL` est positionnée). Ainsi dans la fonction `recv_icmp_pkt`, on commence directement par tester le champ identificateur et le numéro de séquence (lignes 84 et 85). Si ce test a été passé avec succès, c'est-à-dire que l'on a bien reçu le paquet attendu, la fonction `recv_icmp_pkt` retourne `0` après avoir, s'il y en a, imprimé les données incluses dans le paquet. Dans le cas contraire, la valeur retournée est 1.

```
65| static int recv_icmp_pkt(int noc, struct sockaddr_in6 *from, uint16_t id,
66|                          uint16_t seq)
67| {
68| int opt_data_size;
69| char from_ascii[INET6_ADDRSTRLEN];
70| struct icmp6_hdr *icmp;
71|
72|    if (inet_ntop(AF_INET6, &from->sin6_addr, from_ascii,
73|                  INET6_ADDRSTRLEN) == NULL) {
74|       perror("inet_ntop");
75|       return -1;
76|    }
77|    if (noc < sizeof(struct icmp6_hdr)) {
78|       fprintf(stderr, "recv_icmp_pkt: packet too short from %s\n",
79|               from_ascii);
80|       return -1;
81|    }
82|    opt_data_size = noc - sizeof(struct icmp6_hdr);
83|    icmp = (struct icmp6_hdr *) buf;
84|    if (icmp->icmp6_id != id || icmp->icmp6_seq != seq)
85|       return 1;
86|    fprintf(stdout, "Got answer from %s (seq = %d)\n", from_ascii, seq);
87|    if (opt_data_size > 0) {
88|       fprintf(stdout, "with data [\n");
89|       fflush(stdout);
90|       if (opt_data_size > MAX_DATALEN) {
91|          fprintf(stderr,
92|                  "recv_icmp_pkt: received too much data from %s\n",
93|                  from_ascii);
94|       }
95|       else
```

```
 96|          write(1, (char *) icmp + sizeof(struct icmp6_hdr), opt_data_size);
 97|       fprintf(stdout, "\n] (end of data)\n");
 98|    }
 99|    return 0;
100| }
```

D) Programme principal

Le programme principal ne présente pas de difficulté particulière puisqu'il est une application directe des fonctions décrites dans les deux sections précédentes.

La première partie est triviale : elle concerne le traitement des (éventuelles) options.

```
  1| #include <stdio.h>
  2| #include <stdlib.h>
  3| #include <unistd.h>
  4| #include <string.h>
  5| #include <sys/socket.h>
  6| #include <netinet/in.h>
  7| #include <netinet/icmp6.h>
  8| #include <arpa/inet.h>
  9| #include <netdb.h>
 10| #ifdef __linux__
 11| #include <linux/version.h>
 12| #if LINUX_VERSION_CODE < KERNEL_VERSION(2,4,19)
 13| #define LINUX_CKSUM_CALCUL_EXPLICITE
 14| #endif
 15| #endif
 16|
 17| #ifndef TIMEOUT
 18| #define TIMEOUT 5
 19| #endif
 20|
 21| extern int send_echo_request6(int, struct sockaddr_in6 *, uint16_t,
 22|                               uint16_t, char *, int);
 23| extern void wait_for_echo_reply6(int, struct sockaddr_in6 *, uint16_t,
 24|                                  uint16_t, int);
 25|
 26| static void usage(char *);
 27|
 28| int main(int argc, char **argv)
 29| {
 30| int sock, timeout = TIMEOUT, a, ecode;
 31| char *opt_data = NULL, *dst_ascii;
 32| int opt_data_size = 0;
 33| uint16_t id, seq = 0;
 34| struct sockaddr_in6 *dst;
 35| struct addrinfo *res;
 36| struct addrinfo hints = {
 37|                          AI_CANONNAME,
 38|                          PF_INET6,
 39|                          SOCK_RAW,
 40|                          IPPROTO_ICMPV6,
 41|                          0,
 42|                          NULL,
 43|                          NULL,
 44|                          NULL
 45|                         };
```

```
 46|
 47|    while((a = getopt(argc, argv, "d:s:t:")) != EOF)
 48|       switch(a) {
 49|       case 'd':
 50|          opt_data = optarg;
 51|          opt_data_size = strlen(optarg) + 1;
 52|          break;
 53|       case 's':
 54|          seq = (uint16_t) atoi(optarg);
 55|          break;
 56|       case 't':
 57|          timeout = atoi(optarg);
 58|          break;
 59|       default:
 60|          usage(*argv);
 61|       }
 62|       argc -= optind;
 63|       if (argc != 1)
 64|          usage(*argv);
 65|       argv += optind;
```

Ensuite c'est la préparation de l'adresse de la socket distante, opération qui est devenue maintenant familière. Noter que l'on a affecté au champ `ai_family` de la structure `hints` la valeur `PF_INET6` lors de sa déclaration (ligne 38) : on doit s'assurer que la machine cible est une machine IPv6 (il n'existe pas de mode double pile avec utilisation d'adresse IPv4 mappé pour le protocole ICMP, car celui-ci a fortement changé entre IPv4 et IPv6). On s'est interdit des adresses destination de type multicast (lignes 73 à 76) car, comme l'on ne traite qu'un paquet en réception, cela n'aurait guère d'intérêt.

On crée la socket qui servira à l'émission du paquet ECHO_REQUEST et à la réception du paquet ECHO_REPLY en provenance de la machine cible.

À la ligne 96, la valeur du champ identificateur du paquet ICMPv6 est calculée en fonction du numéro de processus en prenant les 16 premiers bits. C'est une technique sûre (et simple) quant à la garantie de l'unicité de l'identificateur. Enfin le paquet ECHO_REQUEST est émis (`send_echo_request6`) puis on attend la réponse éventuelle (`wait_for_echo_reply6`).

```
 66|       ecode = getaddrinfo(*argv, NULL, &hints, &res);
 67|       if (ecode) {
 68|          fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(ecode));
 69|          exit(1);
 70|       }
 71|       dst_ascii = res->ai_canonname ? res->ai_canonname : *argv;
 72|       dst = (struct sockaddr_in6 *) res->ai_addr;
 73|       if (IN6_IS_ADDR_MULTICAST(&dst->sin6_addr)) {
 74|          fprintf(stderr, "%s multicast address not supported\n", dst_ascii);
 75|          exit(1);
 76|       }
 77|       if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0)
{
 78|          perror("socket (RAW)");
 79|          exit(1);
 80|       }
 81| #ifdef LINUX_CKSUM_CALCUL_EXPLICITE
 82|       {
 83|       /*
 84|        * Pour linux avant 2.4.19, il faut demander le calcul des checksums
 85|        * sur les sockets raw, meme pour des paquets icmpv6
```

```
 86|         */
 87| #define OFFSETOF(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
 88|         int off = OFFSETOF(struct icmp6_hdr, icmp6_cksum);
 89|
 90|            if (setsockopt(sock, SOL_RAW, IPV6_CHECKSUM, &off, sizeof off) < 0) {
 91|               perror("setsockopt (IPV6_CHECKSUM)");
 92|               exit(1);
 93|            }
 94|       }
 95| #endif
 96|       id = (uint16_t) (getpid() & 0xffff);
 97|       fprintf(stdout, "Sending ECHO REQUEST to: %s\n", dst_ascii);
 98|       if (send_echo_request6(sock, dst, id, seq, opt_data,
 99|                              opt_data_size) < 0)
100|          exit(1);
101|       fprintf(stdout, "Waiting for answer (timeout = %ds)...\n", timeout);
102|       wait_for_echo_reply6(sock, dst, id, seq, timeout);
103|       close(sock);
104|       exit(0);
105| }
106|
107| static void usage(char *s)
108| {
109|    fprintf(stderr, "Usage: %s [-d data] [-s seq] [-t timeout] host | addr\n", s);
110|    exit(1);
111| }
```

5) Utilisation du multicast

La programmation avec les groupes multicast n'est pas standardisée en IPv4. La nouvelle API "socket" propose un ensemble de structures et appels systèmes pour étendre l'interface de programmation sockets aux applications utilisant le multicast. Cet exemple va illustrer ce point.

Le but des deux programmes est d'échanger des données par multicast. Le programme `in2multi6` diffuse les données lues en entrée ("standard input") vers un groupe multicast donné. Le programme `multi2out6` écoute les paquets transmis dans ce groupe et les copie sur la sortie standard ("stdout").

A) multi2out6

Pour ce faire `multi2out6` va faire des appels systèmes qui vont produire des paquets d'abonnement (et de désabonnement) à un groupe multicast. L'abonnement sera réalisé grâce à l'envoi de deux messages ICMPv6 successifs de type 131, c'est-à-dire des "rapports d'abonnement". Puis les messages émis dans le groupe sont reçus par le programme. Lorsque le programme s'arrête, le code de l'interface va automatiquement provoquer l'émission d'un message de réduction d'un groupe de multicast (132).

Le programme multi2out6 est appelé de la manière suivante :

```
multi2out6 [-i interface] <adresse de groupe multicast>
```

Voici le code complet du programme. Le port utilisé (ligne 15) est quelconque mais ne doit bien sûr pas correspondre à un service déjà existant.

```
 1| #include <sys/types.h>
 2| #include <sys/socket.h>
 3| #include <netinet/in.h>
 4| #include <arpa/inet.h>
 5| #include <stdio.h>
 6| #include <signal.h>
 7| #include <unistd.h>
 8|
 9| #ifndef SO_REUSEPORT
10| #define SO_REUSEPORT SO_REUSEADDR
11| #endif
12|
13| struct sockaddr_in6 sin6;
14|
15| #define IPPORT 54321
16|
17| void Perror(const char *c)
18| {
19|    perror(c);
20|    exit(1);
21| }
22|
23| void Usage ()
24| {
25|    fprintf(stderr, "%s\n", "Usage: multi2out6 [-i interface] addr");
26|    exit(1);
27| }
28|
29| void BrokenPipe(int Signal)
30| {
31|    signal(SIGPIPE, BrokenPipe);
32|    return;
33| }
34|
```

La partie principale du programme traite les options éventuelles. En fait, il n'y en a qu'une, le choix de l'interface à abonner au groupe multicast. Une fois ces operations effectuées, il faut créer et attacher une socket à une adresse. Ces opérations sont réalisées par l'utilisation des fonctions classiques socket et bind.

On utilise une structure de données spéciale pour stocker l'adresse multicast du groupe (définie dans `netinet/in.h`) :

```
struct ipv6_mreq {
struct in6_addr ipv6mr_multiaddr; /* IPv6 mcast address of group */
unsigned int ipv6mr_interface; /* local IPv6 address of interface */
};
```

```
35| int main(int argc, char **argv)
36| {
37| struct ipv6_mreq mreq;
38| int cc, ccb, ch, s;
39| char buf[10240];
40| u_int one = 1;
41| u_int ifi = 0;
42|
43|    signal(SIGPIPE, BrokenPipe);
44|    while ((ch = getopt(argc, argv, "i:")) != -1)
45|       switch(ch) {
46|       case 'i':
47|          if (sscanf(optarg, "%u\0", &ifi) != 1 &&
48|                      (ifi = if_nametoindex(optarg)) == 0)
49|             Usage();
50|          break;
51|       default:
52|          Usage();
53|       }
54|    argc -= optind;
55|    argv += optind;
56|    if (argc != 1)
57|       Usage();
58|
59|    if ((s = socket(AF_INET6, SOCK_DGRAM, IPPROTO_UDP)) < 0)
60|       Perror("socket");
61|    setsockopt(s, SOL_SOCKET, SO_REUSEPORT, &one, sizeof(one));
62|
63| #ifdef SIN6_LEN
64|    sin6.sin6_len = sizeof(sin6);
65| #endif
66|    sin6.sin6_family = AF_INET6;
67|    sin6.sin6_port = htons(IPPORT);
68|    if (bind(s, (struct sockaddr *)&sin6, sizeof(sin6)) < 0)
69|       Perror("bind");
```

La fonction `inet_pton` va permettre la conversion du nom de groupe passé en option sous une forme textuelle (par exemple `ff12::1234:5678`) en forme numérique. Le résultat est directement stocké dans la variable `mreq` qui sera utilisée par la commande `setsockop`. On passe en paramètre à cette fonction l'option `IPV6_JOIN_GROUP` avec la variable `mreq`. À partir de ce moment, il y a émission de deux messages d'abonnement. La boucle qui suit va permettre la lecture des informations envoyées sur le groupe auquel on vient de s'abonner et les afficher sur la sortie standard ainsi que leur longueur sur la sortie erreur standard.

```
70|    if (inet_pton(AF_INET6, *argv, &mreq.ipv6mr_multiaddr) != 1)
71|       Usage();
72|    mreq.ipv6mr_interface = ifi;
73|    if (setsockopt(s,IPPROTO_IPV6, IPV6_JOIN_GROUP, &mreq, sizeof(mreq)) < 0)
74|       Perror("setsockopt IPV6_JOIN_GROUP");
75|    for (;;) {
76|       cc = read(s, buf, 10240);
77|       if (cc < 0)
78|          Perror("read socket");
79|       if (cc == 0) {
80|          fprintf(stderr, "..\n");
81|          exit (0);
82|       }
83|       ccb = write(1, buf, cc);
84|       if (ccb != cc)
```

```
85|          Perror("write file");
86|       fprintf(stderr, "<-%d-\n", cc);
87|    }
88| }
```

Lorsque le programme s'arrête, un `close(s)` implicite a lieu, et le code de l'interface va envoyer un message de réduction de groupe si elle est la dernière à avoir envoyé un rapport d'abonnement au groupe.

B) in2multi6

Le programme est appelé de la manière suivante :

`in2multi6 [-i interface][-h max-hop-count][-l loop] <adresse de groupe multicast>`

Le code est relativement simple, principalement une analyse des arguments, le positionnement d'option et une boucle lecture--émission. En effet il n'est pas nécessaire de s'abonner pour faire de l'émission multicast.

Il y a quatre arguments, trois optionnels qui sont l'interface d'émission (nom ou index numérique), le "ttl" mis dans les paquets multicast (voir le manuel de la primitive `readv`), et un drapeau qui sert à dire si la machine émettrice reçoit ou non les paquet émis. Le dernier argument est l'adresse du groups sous forme numérique.

Voici le code complet du programme. Le port utilisé (ligne 10) est naturellement celui de `in2multi6`.

```
 1| #include <sys/types.h>
 2| #include <sys/socket.h>
 3| #include <netinet/in.h>
 4| #include <arpa/inet.h>
 5| #include <stdio.h>
 6| #include <unistd.h>
 7|
 8| struct sockaddr_in6 sin6;
 9|
10| #define IPPORT 54321
11|
12| void Perror(const char *c)
13| {
14|    perror(c);
15|    exit(1);
16| }
17|
18| void Usage ()
19| {
20|       fprintf(stderr, "%s\n", "Usage: in2multi6 [-i interface][-h hop][-l loop]
addr");
21|    exit(1);
22| }
23|
24| int main(int argc, char **argv)
25| {
26| u_int hops = 1,       /* as defined in rfc2553 */
27|       loop = 1,       /* as defined in rfc2553 */
28|       ifi = 0;
```

```
29| int s, cc, ch;
30| char buf[1024];
31| struct in6_addr addr6;
32| extern char *optarg;
33| extern int optind;
34|
35|    addr6 = in6addr_any;
36|    if ((s = socket(AF_INET6, SOCK_DGRAM, IPPROTO_UDP)) < 0)
37|       Perror("socket");
38|    while ((ch = getopt(argc, argv, "h:t:l:i:")) != -1)
39|       switch(ch) {
40|       case 'h':
41|       case 't':
42|          hops = atoi(optarg);
43|          break;
44|       case 'l':
45|          loop = atoi(optarg);
46|          break;
47|       case 'i':
48|          if (sscanf(optarg, "%u\0", &ifi) != 1) {
49|             ifi = if_nametoindex(optarg);
50|             if (ifi == 0)
51|                Usage();
52|          }
53|          break;
54|       default:
55|          Usage();
56|    }
57|    argc -= optind;
58|    argv += optind;
59|    if (argc != 1 || inet_pton(AF_INET6, *argv, &addr6) <= 0)
60|       Usage();
61|    if (setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_HOPS,
62|                  &hops, sizeof(hops)) < 0)
63|       Perror("setsockopt IPV6_MULTICAST_HOPS");
64|    if (setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_LOOP,
65|                  &loop, sizeof(loop)) < 0)
66|       Perror("setsockopt IPV6_MULTICAST_LOOP");
67|    if (ifi && (setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_IF,
68|                           &ifi, sizeof(u_int)) < 0))
69|       Perror("setsockopt IPV6_MULTICAST_IF");
70|
71| #ifdef SIN6_LEN
72|    sin6.sin6_len = sizeof(sin6);
73| #endif
74|    sin6.sin6_family = AF_INET6;
75|    sin6.sin6_addr = addr6;
76|    sin6.sin6_port = htons(54321);
77|
78|    for (;;) {
79|       cc = read(0, buf, 1024);
80|       if (cc < 0)
81|          Perror("read file");
82|       if (cc == 0) {
83|          fprintf(stderr, ".\n", cc);
84|          exit (0);
85|       }
86|       if (sendto(s, buf, cc, 0,
87|                (struct sockaddr *)&sin6, sizeof(sin6)) < 0)
88|          Perror("sendto");
89|       fprintf(stderr, "-%d->\n", cc);
90|    }
91| }
```

6) Programmation avancée

L'API « avancée » (*Advanced API*), définie par le RFC 3542, a pour objet la standardisation de la manipulation en émission/réception des datagrammes IPv6. Elle permet notamment au programmeur d'écrire des applications utilisant les nouvelles fonctionnalités proposées par le protocole IPv6 et ce de façon portable.

Cette API avancée concerne essentiellement les sockets de type `SOCK_DGRAM` (UDP) ou de type `SOCK_RAW` (ICMPv6,...). En effet, comme il n'y a pas de correspondance biunivoque entre les opérations de réception (respectivement d'émission) et les segments TCP reçus (respectivement émis), la plupart des options proposées ne sont pas applicables ou voire dénuées de sens pour une socket de type `SOCK_STREAM`. L'API avancée est utile pour programmer des applications comme `ping`, `traceroute`, des implémentations de protocoles de routage et, de manière générale, toute application construite avec des sockets de type `SOCK_RAW` et devant accéder aux champs des en-têtes IPv6 ou ICMPv6.

La standardisation des appels systèmes et de fonctions a pour but de fournir un interface uniforme, évitant ainsi l'hétérogénéité qui existe en IPv4.

Les opérations disponibles sont les suivantes :

- Calcul/vérification des checksums par le noyau (pour les sockets de type `SOCK_RAW`)
- Filtrage des réceptions des paquets ICMPv6
- Modification des caractéristiques du datagramme IPv6 (packet information)
- Manipulation des en-têtes d'extension IPv6
- proche-en-proche (hop-by-hop)
- routage par la source (routing header)
- destination (destination)
- Gestion du MTU et du mécanisme de découverte du PMTU (Path MTU discovery)

En outre, des fonctions facilitant le traitement des en-têtes d'extension IPv6 ont été définies ainsi qu'une interface étendant les primitives `rresvport`, `rcmd` et `rexec` à IPv6. Cette API avancée ne prend pas en compte les en-têtes d'extension IPv6 liés à IPsec.

L'implémentation de ce nouveau standard est réalisé à l'aide des primitives `sendmsg` et `recvmsg`, les données en émission/réception étant traitées via les données auxiliaires (ancillary data) associées à la socket et gérées par ces primitives. Également, de nouvelles options ont été définies pour les sockets.

A) L'implémentation

Comme indiqué précédemment, l'implémentation de l'API avancée repose principalement sur les primitives `sendmsg` et `recvmsg` dont les prototypes sont les suivants :

```
int sendmsg(int s, const struct msghdr *msg, int flags);
int recvmsg(int s, struct msghdr *msg, unsigned int flags);
```

Le premier paramètre s désigne le descripteur d'E/S associée à la socket et le dernier paramètre flags est identique au 3ème paramètre des primitives sendto et recvfrom. Le second paramètre est une structure définie (dans `<sys/socket.h>`) comme suit :

```
struct msghdr {
   void *msg_name;           /* pointeur vers l'adresse de la socket */
   socklen_t msg_namelen;    /* longueur de l'adresse de la socket */
   struct iovec *msg_iov;    /* tampon mémoire vectoriel (scatter/gather array) */
   int msg_iovlen;           /* nombre d'éléments de msg_iov */
   void *msg_control;        /* données auxiliaires */
   socklen_t msg_controllen; /* longueur des données auxiliaires */
   int msg_flags;            /* drapeaux des messages reçus */
};
```

Les deux premiers champs spécifient pour `sendmsg` (respectivement `recvmsg`) l'adresse de destination (respectivement d'origine). Le premier champ peut être le pointeur `NULL` en mode connecté. Les deux champs suivants contiennent le tampon mémoire vectoriel en émission ou en réception suivant le cas (voir le manuel de la primitive `readv`).

Les champs `msg_control` et `msg_controllen` spécifient le tableau des données auxiliaires reçues ou émises, le champ `msg_control` pouvant être le pointeur `NULL` s'il n'y a aucune donnée auxiliaire à émettre ou recevoir. Chaque donnée auxiliaire se présente sous la forme d'une structure de type struct `cmsghdr` définie (dans `sys/socket.h`) :

```
struct cmsghdr {
   socklen_t cmsg_len; /* longueur en octet, en-tête inclus */
   int cmsg_level;     /* protocole (IPPROTO_IPV6, ...) */
   int cmsg_type;      /* sous-type dans le protocole (IPV6_RTHDR, ...) */
                       /* suivi par unsigned char cmsg_data[]; */
};
```

En raison de problèmes d'alignement (cf. figure Structure des données auxiliaires), l'accès au tableau des données auxiliaires ainsi que la manipulation de ces dernières ne doivent se faire qu'au moyen de cinq macros appropriées, définies dans `<sys/socket.h>` :

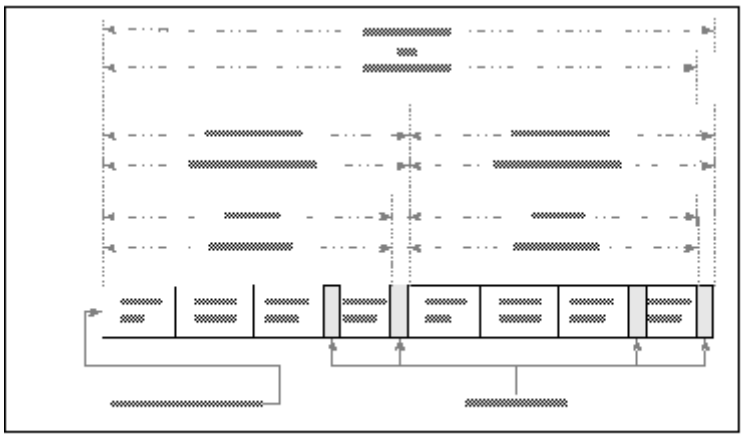


Figure 15-6. Structure des données auxiliaires

- `struct cmsghdr *CMSG_FIRSTHDR(const struct msghdr *msgh);`
  `CMSG_FIRSTHDR` renvoie un pointeur vers la première donnée auxiliaire contenue dans la structure de type `struct msghdr` pointée par `msgh`.
- `struct cmsghdr *CMSG_NXTHDR(const struct msghdr *msgh, const struct cmsghdr *cmsg);`
  `CMSG_NXTHDR` renvoie un pointeur vers la donnée auxiliaire qui suit celle pointée par `cmsg` ou le pointeur `NULL` s'il n'y en a pas. Si `cmsg` est le pointeur `NULL`, `CMSG_NXTHDR` renvoie un pointeur vers la première donnée auxiliaire. Ainsi, `CMSG_NXTHDR(msgh, NULL)` est équivalent à `CMSG_FIRSTHDR(msgh)`.
- `socklen_t CMSG_SPACE(socklen_t length);`
  `CMSG_SPACE` renvoie le nombre d'octets occupés par une donnée auxiliaire dont la taille des données transmises est `length`, tout en tenant compte des alignements.
- `socklen_t CMSG_LEN(socklen_t length);`
  `CMSG_LEN` retourne la valeur à stocker dans le champ `cmsg_len` de la structure (de type `struct cmsghdr`) associée à une donnée auxiliaire dont la taille des données transmises est `length`, ceci en tenant compte des alignements.
- `unsigned char *CMSG_DATA(const struct cmsghdr *cmsg);`
  `CMSG_DATA` retourne un pointeur vers les données contenues dans la donnée auxiliaire pointée par le paramètre `cmsg`.

Le dernier champ `msg_flags` de la structure msghdr est rempli au retour de `recvmsg()`. Plusieurs drapeaux peuvent avoir été levés dont le drapeau `MSG_TRUNC` pour indiquer que les données ont été tronquées ou le drapeau `MSG_CTRUNC` pour indiquer que les données auxiliaires ont été tronquées.

Afin de recevoir toute donnée auxiliaire sur une socket, il faut auparavant le demander en positionnant l'option correspondante. Plus précisément, le RFC 3542 liste de manière exhaustive des options disponibles et comment les positionner :

```
int on = 1;
/* interface de réception / adresse destination */
setsockopt(s, IPPROTO_IPV6, IPV6_RECVPKTINFO, &on, sizeof(on));
/* nombre de sauts */
setsockopt(s, IPPROTO_IPV6, IPV6_RECVHOPLIMIT, &on, sizeof(on));
/* en-tête de routage */
setsockopt(s, IPPROTO_IPV6, IPV6_RECVRTHDR, &on, sizeof(on));
/* options proche-en-proche */
setsockopt(s, IPPROTO_IPV6, IPV6_RECVHOPOPTS, &on, sizeof(on));
/* option destination */
setsockopt(s, IPPROTO_IPV6, IPV6_RECVDSTOPTS, &on, sizeof(on));
/* classe de trafic */
```

```
setsockopt(s, IPPROTO_IPV6, IPV6_RECVTCLASS, &on, sizeof(on));
```

En ce qui concerne l'émission d'une donnée auxiliaire, deux possibilités s'offrent au programmeur :

- soit il fait appel à la primitive `setsockopt` pour positionner l'option correspondante avec les données adéquates. Ce sont alors des options dites permanentes (*sticky*) car elles s'appliquent à tous les paquets transmis par la suite et ce jusqu'à un nouvel appel à `setsockopt` ou une surcharge par une donnée auxiliaire.
- soit il utilise `sendmsg` et les données auxiliaires affectent uniquement le datagramme concerné (non applicable au socket de type `SOCK_STREAM`)

Le tableau Options de données auxiliaires en émission, extrait du RFC 3542, donne la liste des options disponibles en émission (avec leur type de données associées) :

Options de données auxiliaires en émission		
opt level / cmsg_level	**optname / cmsg_type**	**optval / cmsg_data[]**
IPPROTO_IPV6	IPV6_PKTINFO	structure in6_pktinfo
IPPROTO_IPV6	IPV6_HOPLIMIT	int
IPPROTO_IPV6	IPV6_NEXTHOP	structure sockaddr_in6
IPPROTO_IPV6	IPV6_RTHDR	structure ip6_rthdr
IPPROTO_IPV6	IPV6_HOPOPTS (prochain saut / next hop)	structure ip6_hbh
IPPROTO_IPV6	IPV6_DSTOPTS	structure ip6_dest
IPPROTO_IPV6	IPV6_RTHDRDSTOPTS	structure ip6_dest
IPPROTO_IPV6	IPV6_TCLASS	int

Les options proposées par cette API avancée, ne seront pas toutes détaillées dans ce chapitre. Nous recommandons au lecteur intéressé de se reporter au RFC 3542. L'exemple simple qui suit, met en oeuvre ces notions. Il s'agit d'un extrait de programme qui, outre les données habituelles, souhaite recevoir également deux données auxiliaires :

- index de l'interface de réception du paquet / adresse destination du paquet reçu (option `IPV6_RECVPKTINFO`) et
- le nombre de sauts (hop limit) du paquet reçu (option `IPV6_RECVHOPLIMIT`).

La première partie de ce programme, où les variables sont déclarées et initialisées ne présente aucune difficulté. On notera l'usage de la macro `CMSG_SPACE` afin d'initialiser la variable `cmsg_buf` destinée à accueillir les données auxiliaires demandées.

```
int noc, o = 1, s;
struct sockaddr_storage from;
```

```
char buf[1024];
struct iovec iov = {buf, sizeof(buf)};
char cmsg_buf[CMSG_SPACE(sizeof(struct in6_pktinfo)) + CMSG_SPACE(sizeof(int))];
struct cmsghdr *cmsg;
struct msghdr msg = {(void *) &from, sizeof(from), &iov, 1,
                     (void *) cmsg_buf, sizeof(cmsg_buf), 0};
```

Actuellement, un grand nombre d'implémentations ne sont pas à jour du RFC 3542 (bien que publié en mai 2003). En particulier, certaines implémentations ne distinguent toujours pas entre les options de réception et les options d'émission. Si bien qu'il peut être nécessaire d'ajouter les lignes suivantes :

```
#ifndef IPV6_RECVHOPLIMIT
#define IPV6_RECVHOPLIMIT IPV6_HOPLIMIT
#endif
#ifndef IPV6_RECVPKTINFO
#define IPV6_RECVPKTINFO IPV6_PKTINFO
#endif
```

Ensuite on indique par l'intermédiaire de la primitive `setsockopt` que les données auxiliaires mentionnées plus haut doivent être reçues. La variable `s` est un descripteur d'entrées/sorties associé à une socket `PF_INET6`.

```
if (setsockopt(s, IPPROTO_IPV6, IPV6_RECVPKTINFO, &o, sizeof(o)) ||
   setsockopt(s, IPPROTO_IPV6, IPV6_RECVHOPLIMIT, &o, sizeof(o))) {
     /* traitement de l'erreur */
}
```

La primitive `recvmsg` est exécutée et les erreurs éventuelles sont traitées :

```
if ((noc = recvmsg(s, &msg, 0)) < 0) {
/* traitement de l'erreur */
}
if (msg.msg_flags & MSG_TRUNC) {
/* traitement de l'erreur (les données sont tronquées) */
}
if (msg.msg_flags & MSG_CTRUNC) {
/* traitement de l'erreur (les données auxiliaires sont tronquées) */
}
```

Finalement, au moyen des macros `CMSG_FIRSTHDR` et `CMSG_NXTHDR` précédemment décrites, une boucle traite les données auxiliaires reçues :

```
for (cmsg = CMSG_FIRSTHDR(&msg); cmsg; cmsg = CMSG_NXTHDR(&msg, cmsg)) {
   if ((cmsg->cmsg_level == IPPROTO_IPV6) && (cmsg->cmsg_type == IPV6_PKTINFO)) {
      struct in6_pktinfo *pi = (struct in6_pktinfo *) CMSG_DATA(cmsg);
      /* suite du traitement */
   }
   if ((cmsg->cmsg_level == IPPROTO_IPV6) && (cmsg->cmsg_type == IPV6_HOPLIMIT)) {
       int hlim = *(int *)CMSG_DATA(cmsg);
       /* suite du traitement */
   }
/* suite du programme */
}
```

B) L'exemple « mini-ping » revisité

Le programme `one_ping6.c` va être repris afin de lui ajouter deux fonctionnalités dont l'implémentation s'appuiera sur l'usage de données auxiliaires. On souhaite d'une part afficher le nombre de sauts (*hop limit*) du paquet ECHO_REPLY (éventuellement) reçu et d'autre part de permettre, à l'instar de la commande `ping6`, de passer une liste de relais par lesquels le paquet ECHO_REQUEST devra transiter avant d'être envoyé à l'hôte destinataire (routage par la source).

Par exemple, pour envoyer un paquet ECHO_REQUEST à la machine `ipv6.imag.fr` tout en transitant tout d'abord par les machines `www.kame.net` et `relai.imag.fr`, la commande `xapi_ping6` sera :

```
$ xapi_ping6 www.kame.net relais.imag.fr ipv6.imag.fr
Sending ECHO REQUEST to: ipv6.imag.fr via:
www.kame.net
relais.imag.fr
Waiting for answer (timeout = 5s)...
Got answer from 2001:660:9510:25::632 (seq = 0, hoplimit = 241)
```

L'affichage du nombre de sauts a déjà été en grande partie traité dans l'exemple du paragraphe consacré à l'implémentation de l'API avancée. Nous indiquerons donc seulement les changements significatifs par rapport à la version originale. Ces changements concernent essentiellement la routine `wait_for_echo_reply6`. La première tâche à effectuer est, comme dans l'exemple précédent, de positionner l'option `IPV6_RECVHOPLIMIT`, juste après avoir mis en place le filtrage ICMPv6. L'instruction :

```
noc = recvfrom(sock, buf, sizeof(buf), 0, (struct sockaddr *) from, &from_len);
```

est remplacée par la nouvelle instruction :

```
noc = recv_data(sock, buf, sizeof(buf), 0, (struct sockaddr *) from, &from_len,
&hoplimit);
```

où `hoplimit` est un entier qui a été précédemment déclaré dans le corps de la fonction `wait_for_echo_reply6` et `recv_data` a pour texte :

```
 1| #ifdef sun /* For Solaris */
 2| #define _XOPEN_SOURCE 500 /* correct recvmsg/sendmsg/msg/CMSG_xx syntax */
 3| #define __EXTENSIONS__
 4| #endif
 5| #include <stdio.h>
 6| #include <stdlib.h>
 7| #include <unistd.h>
 8| #include <sys/uio.h>
 9| #include <sys/types.h>
10| #include <sys/socket.h>
11| #include <netinet/in.h>
12| #include <netinet/ip6.h>
13| #ifndef CMSG_SPACE /* Solaris <= 9 */
14| #define CMSG_SPACE(l) ((size_t)_CMSG_HDR_ALIGN(sizeof (struct cmsghdr) + (l)))
15| #define CMSG_LEN(l) ((size_t)_CMSG_DATA_ALIGN(sizeof (struct cmsghdr)) + (l))
16| #endif
17|
18| int recv_data(int sock, void *buf, int len, unsigned int flags,
19|               struct sockaddr *from, socklen_t *fromlen, int *hoplimit)
20| {
21| int ret, found = 0, cmsgspace = CMSG_SPACE(sizeof(int));
```

```
22| struct iovec iov = {buf, len};
23| struct cmsghdr *cmsg = (struct cmsghdr *) malloc(cmsgspace), *ptr;
24| struct msghdr msg = {
25|    (caddr_t) from, *fromlen, &iov, 1,
26|    (caddr_t) cmsg, cmsgspace
27| };
28|
29| if (cmsg == NULL) {
30|    perror("recv_data: malloc");
31|    return -1;
32| }
33| ret = recvmsg(sock, &msg, flags);
34| if (ret < 0) {
35|    perror("recv_data: recvmsg");
36|    goto done;
37| }
38| if (msg.msg_flags & MSG_TRUNC) {
39|    fprintf(stderr, "recv_data: recvmsg: data discarded before delivery\n");
40|    goto bad;
41| }
42| if (msg.msg_flags & MSG_CTRUNC) {
43|    fprintf(stderr,
44|            "recv_data: recvmsg: control data lost before delivery\n");
45|    goto bad;
46| }
47| if (msg.msg_controllen)
48|    for (ptr = CMSG_FIRSTHDR(&msg); ptr; ptr = CMSG_NXTHDR(&msg, ptr)) {
49|       if (ptr->cmsg_level==IPPROTO_IPV6 && ptr->cmsg_type==IPV6_HOPLIMIT) {
50|          if (ptr->cmsg_len != CMSG_LEN(sizeof(int))) {
51|             fprintf(stderr,
52|                     "recvmsg: ancillary data with invalid length\n");
53|             goto bad;
54|          }
55|          *hoplimit = *((int *) CMSG_DATA(ptr));
56|          goto done;
57|       }
58|    }
59|    fprintf(stderr,
60|            "recv_data: recvmsg: hoplimit not found in ancillary data\n");
61|  bad:
62|    ret = -1;
63|  done:
64|    free(cmsg);
65|    return ret;
66| }
```

Le code de la fonction `recv_data` ne sera pas commenté car la réception du nombre de sauts via une donnée auxiliaire a été étudiée dans un précédent exemple, la seule différence étant que la gestion des erreurs est ici plus détaillée.

Il faut enfin modifier trivialement le code de la routine `recv_icmp_pkt` afin que celle-ci imprime le nombre de sauts du paquet ECHO_REPLY (éventuellement) reçu. Nous en laissons le soin au lecteur.

Pour l'autre donnée auxiliaire (cette fois ci en émission), à savoir le routage par la source, il faut naturellement tout d'abord modifier la fonction `send_echo_request6` et en premier lieu son prototype qui devient :

```
int send_echo_request6(int sock, struct sockaddr_in6 *dst, uint16_t id,
                       uint16_t seq, char *opt_data, int opt_data_size,
                       struct in6_addr *seg, int nseg)
```

La routine `send_echo_request6` modifié possède deux arguments supplémentaires ajoutés à la fin. Le premier de ces nouveaux arguments est un pointeur vers un tableau contenant les adresses IPv6 des relais par lesquels on souhaite effectuer le routage par la source. Le dernier argument est le nombre d'éléments de ce tableau, c'est-à-dire le nombre de relais.

Il faut ensuite substituer dans le corps de la fonction `send_echo_request6` l'instruction suivante :

```
noc = sendto(sock, (char *) icmp, icmp_pkt_size, 0, (struct sockaddr *) dst,
```

par :

```
noc = send_data(sock, (void *) icmp, icmp_pkt_size, 0,
                (struct sockaddr *) dst, sizeof(struct sockaddr_in6),
                seg, nseg);
```

Si la variable `seg` est le pointeur `NULL`, la liste des relais est vide. On fait appel à la fonction `send_data`, dont le code va être commenté en détails :

```
 1| #ifdef sun /* For Solaris */
 2| #define _XOPEN_SOURCE 500 /* correct recvmsg/sendmsg/msg/CMSG_xx syntax */
 3| #define __EXTENSIONS__
 4| #endif
 5| #include <stdio.h>
 6| #include <stdlib.h>
 7| #include <unistd.h>
 8| #include <sys/uio.h>
 9| #include <sys/types.h>
10| #include <sys/socket.h>
11| #include <netinet/in.h>
12| #include <netinet/ip6.h>
13| #ifndef CMSG_SPACE /* Solaris <= 9 */
14| #define CMSG_SPACE(l) ((size_t)_CMSG_HDR_ALIGN(sizeof (struct cmsghdr) + (l)))
15| #define CMSG_LEN(l) ((size_t)_CMSG_DATA_ALIGN(sizeof (struct cmsghdr)) + (l))
16| #endif
17| #ifndef IPV6_RECVHOPLIMIT
18| #define IPV6_RECVHOPLIMIT IPV6_HOPLIMIT
19| #endif
20|
21| extern void * inet6_rth_init(); /* sometimes not in ip6.h */
22|
23| int send_data(int sock, void *buf, int len, unsigned int flags,
24|               struct sockaddr *to, socklen_t tolen,
25|               struct in6_addr *seg, int nseg)
26| {
27| int ret = -1, rthsp, cmsgspace;
28| void *data;
29| struct in6_addr *in6;
30| struct iovec iov = {buf, len};
31| struct cmsghdr *cmsg = NULL;
32| struct msghdr msg = {
33|                     (caddr_t) to, tolen, &iov, 1,
34|                     NULL, 0, 0
35|                     };
36|
37|    if (seg != NULL) {
38|       rthsp = inet6_rth_space(IPV6_RTHDR_TYPE_0, nseg);
39|       cmsgspace = CMSG_SPACE(rthsp);
40|       msg.msg_control = cmsg = (struct cmsghdr *) malloc(cmsgspace);
41|       if (cmsg == NULL) {
```

```
42|          perror("recv_data: malloc");
43|          goto bad;
44|       }
45|       cmsg->cmsg_level = IPPROTO_IPV6;
46|       msg.msg_controllen = cmsg->cmsg_len = CMSG_LEN(rthsp);
47|       cmsg->cmsg_type = IPV6_RTHDR;
48|       data = CMSG_DATA(cmsg);
49|       data = (void *)inet6_rth_init(data, rthsp, IPV6_RTHDR_TYPE_0, nseg);
50|       if (!data) {
51|          fprintf(stderr, "send_data: inet6_rth_init failed\n");
52|          goto bad;
53|       }
54|       for (in6 = seg; in6 - seg < nseg; in6++)
55|          if (inet6_rth_add(data, in6) == -1) {
56|             fprintf(stderr, "send_data: inet6_rth_add failed\n");
57|             goto bad;
58|          }
59|    }
60|    ret = sendmsg(sock, &msg, flags);
61|    if (ret < 0) {
62|       perror("send_data: sendmsg");
63|       goto bad;
64|    }
65| bad:
66|    if (cmsg)
67|    free(cmsg);
68|    return ret;
69| }
```

Les six premiers paramètres de la fonction `send_data` sont identiques à ceux de la primitive système `sendto`, les deux derniers étant quant à eux identiques aux deux derniers arguments de la routine `send_echo_request6`.

Si la liste de relais est vide, on appelle `sendmsg` sans données auxiliaires (`msg.msg_control` est nul). Sinon on alloue (ligne 40) un tampon pour contenir les données auxiliaires.

La routine `inet6_rth_space` est l'une des six nouvelles routines proposées par l'API avancée afin de faciliter la tâche du programmeur lors de la manipulation des en-têtes de routage. Elle prend en arguments le type de l'extension de routage (en l'occurrence la constante `IPV6_RTHDR_TYPE_0` dont la valeur numérique est 0 est qui est définie dans `<netinet/in.h>`) et le nombre de relais contenus dans cette extension (pour ce type d'extension, ce nombre doit être compris entre 0 et 127 inclus). Elle retourne la taille en octets nécessaire pour contenir cette en-tête de routage. Ici cette routine va permettre d'initialiser, via la variable `rthsp` et à l'aide de la macro `CMSG_SPACE`, la variable `cmsgspace` à la taille en octets de la donnée auxiliaire associée à cette extension de routage.

En lignes 45 à 47, la longueur des données auxiliaires et la structure `cmsg` sont initialisés au moyen de la macro `CMSG_LEN` pour le champ `cmsg_len`.

Il faut maintenant initialiser les données transmises par la donnée auxiliaire avec l'en-tête routage (lignes 48 à 53). Nous allons nous servir de la routine `inet6_rth_init` fournie par l'API avancée. Celle-ci prend en premier argument un pointeur vers la zone mémoire qui contiendra l'en-tête de routage, le deuxième argument étant la taille en octets de cette zone mémoire. Les deux derniers arguments sont identiques à ceux de la routine `inet6_rth_space`. `inet6_rth_init` retourne un pointeur vers cette zone mémoire ou le pointeur `NULL` si la taille de celle-ci est insuffisante.

Après ces diverses initialisations, la donnée auxiliaire est représentée à la figure Initialisation de l'en-tête de routage où l'on a supposé, afin de fixer les idées, que l'on est en présence d'une architecture 32 bits et que l'alignement se fait sur 32 bits également (autrement dit il n'y a pas de bourrage entre la structure cmsg et le début des données transmises, cf. figure Structure des données auxiliaires).

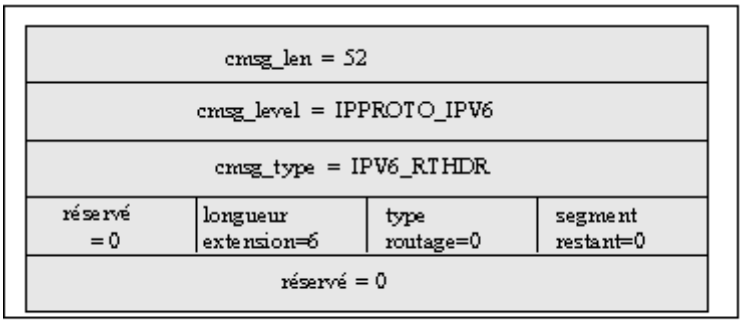


Figure 15-7. Initialisation de l'en-tête de routage

Dans la boucle qui suit (lignes 54 à 58), l'initialisation de l'en-tête de routage se termine en ajoutant successivement les adresses IPv6 des relais du routage par la source. Ces ajouts se font au moyen de la fonction `inet6_rth_add` qui prend en premier argument la zone mémoire contenant l'en-tête de routage et en deuxième argument un pointeur (de type `struct in6_addr *`) vers l'adresse du relais à ajouter.

A l'issue de cette boucle, si l'on reprend l'exemple qui nous a servi à présenter la nouvelle version de la commande one_ping6 :

```
$ xapi_ping6 www.kame.net relais.imag.fr ipv6.imag.fr
```

la donnée auxiliaire sera maintenant comme représentée à la figure Adjonction des deux relais dans l'en-tête de routage, (avec les mêmes hypothèses sur l'architecture et l'alignement). Le message ainsi construit est expédié tout en gérant les erreurs éventuelles (nous laissons le soin au lecteur l'adaptation de la fonction main afin de prendre en compte les nouveaux arguments (optionnels) du programme `one_ping6`).

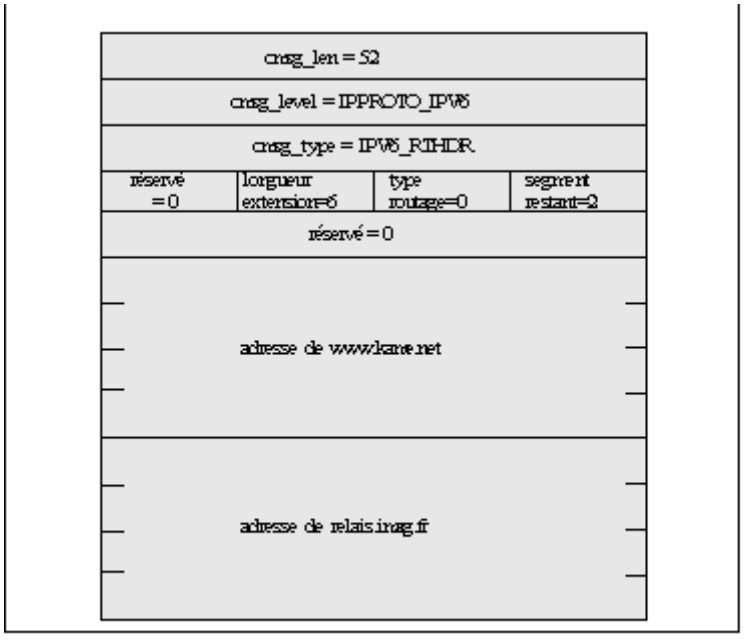


Figure 15-8. Adjonction des deux relais dans l'en-tête de routage

On remarque que la donnée auxiliaire contient les adresses des relais intermédiaires, alors que dans un paquet IPv6, l'en-tête de routage contient les adresses à partir du deuxième relais et l'adresse destination finale, l'adresse du premier relais étant dans l'en-tête IPv6. Le noyau lors du `sendmsg` va permuter les adresses pour rétablir l'ordre correct.

a) Portabilité du code

Solaris définit des prototypes de `sendmsg` et `recvmsg` variables selon les modes de compilation. De plus, jusqu'à la version 9 incluse, il ne définit pas les macros `CMSG_SPACE` et `CMSG_LEN`. Les lignes 1 à 4 et 13 à 19 du programme servent à éviter ces problèmes de compatibilité.

D'autre part, les fonctions `inet6_rth_xxx`, définies dans le RFC 3542 sont encore souvent absentes de la librairie système (c'est le cas pour Solaris 9, FreeBSD4.x, NetBSD1.x, et Linux). Le lecteur peut les remplacer par un codage à la main, ou récupérer leur texte, par exemple dans la distribution KAME.

XVI) Supervision

L'administration des réseaux se décompose en un ensemble de tâches, chacune ayant une fonction bien particulière qui peuvent brièvement être décrite de la manière suivante :

- Supervision (*Monitoring*) : La supervision est essentielle à la bonne administration d'un réseau. Elle consiste à vérifier qu'un ensemble d'équipements constituant un réseau (Routeurs, switchs, serveurs, PCs) fonctionne correctement. Elle permet de connaître à tout moment la disponibilité mais aussi les performances qu'offre le réseau.
- Métrologie : Elle consiste à surveiller et analyser le trafic véhiculé par les équipements réseaux. Elle permet donc d'évaluer le type et la quantité de trafic dans le réseau. La métrologie a pris ainsi une place importante dans le monde de l'administration des réseaux. Parmi les utilisateurs de ce service se trouvent les opérateurs qui l'emploient entre autres pour suivre la consommation de leurs clients et vérifier qu'elle est conforme à leur contrat (*Accounting - Billing*). De même, ils vérifient que leurs liens physiques n'atteignent pas la capacité limite. Grâce aux fonctions de quelques outils, il est possible de faire du "reporting". Il est ainsi plus facile de prévoir le dimensionnement du réseau lors de migrations.
- Sécurité: Il est indispensable d'avoir une politique de sécurité dans un réseau afin d'assurer principalement l'intégrité et la confidentialité des données. Pour cela, il faut mettre en place plusieurs niveaux de sécurité :
    - il est nécessaire de sécuriser l'accès physique aux équipements réseaux (routeurs, commutateurs) et aux serveurs (collecteurs, serveur d'authentification...).
    - pour restreindre l'accès aux utilisateurs non autorisés, des filtres sur les adresses IP et les ports (Access List, IPTables, ACL) ainsi que sur les services applicatifs (Certificats de sécurité, PKI) sont mis en place.

- l'information transmise sur le réseau doit être chiffrée pour éviter que des informations confidentielles comme les mots de passe circulent en clair.

La plus part des outils d'administration offrent ces fonctionnalités :

- Découverte de la Topologie: La topologie permet de connaître à travers l'élaboration de schémas l'architecture du réseau. Il est donc très important de maintenir à jour ces schémas pour avoir une vision correcte du réseau.
- Inventaire: L'inventaire permet de recenser l'ensemble des équipements qui constitue un réseau. Il a aussi pour but de tenir à jour la configuration de ces équipements. Si un inventaire n'est pas fait, il se peut, par exemple, que certains équipements du réseau ne soient pas supervisés. C'est pour cela qu'il est important de maintenir à jour l'inventaire, en même temps que le réseau évolue.

1) MIBs IPv6

Figure 16-1 *Architecture d'administration - Interrogation des MIB*

La croissance rapide des réseaux et la diversité des systèmes pendant la dernière décennie a entraîné le besoin d'une gestion globale des infrastructures réseaux. SNMP (*Simple Network Management Protocol*) s'est avéré être une bonne solution proposée et standardisée par l'IETF. Ce protocole permet de collecter diverses informations stockées dans les équipements du réseau dans des bases de données appelées MIBs (*Management Information Base*). Les équipements sont aussi capables de faire remonter des alertes (*traps*) au collecteur SNMP via ce protocole (cf. figure Architecture d'administration - Interrogation des MIB.).

Le protocole SNMP et les MIBs sont devenus les deux éléments principaux du modèle de supervision de réseaux. Le protocole SNMP étant indépendant de l'architecture du protocole IP, son évolution vers IPv6 n'a pas représenté un problème majeur. La première implémentation de SNMP pour le protocole IPv6 a été réalisée par l'équipe du Loria de l'université de Nancy (NET-SNMP OpenSource) et a été disponible dans la version 5.0.3 de net-snmp, en mai 2002. Avant cette première version, l'administration des réseaux IPv6 était possible du fait que les équipements avaient une double pile configurée IPv4 et IPv6. En effet, il n'était possible d'accéder en SNMP sur un équipement qu'en IPv4 et de récupérer des champs de la MIB concernant IPv6. Aujourd'hui, certains réseaux projets ne véhiculent que du trafic IPv6 d'où la nécessité de disposer d'une version IPv6 pour le transport du protocole SNMP. L'évolution des MIB est plus complexe. Depuis les spécifications initiales du protocole IPv6, en 1995, la définition de la MIB-2 pour l'administration des réseaux IPv6 a été modifiée à deux occasions en 1998 et en 2000 (cf. figure Evolution de la MIB.). Le problème principal était de définir le type du champ "IP ADDRESS". En 1996, une représentation initiale du

champ `IP ADDRESS` est décrite dans le RFC 1902 où la longueur réservée pour ce champ est de 4 octets. Avec ce champ ne peuvent être représentées que les adresses IPv4 puisque les adresses IPv6 ayant une longueur de 128 bits. C'est pourquoi, en 1998, le RFC 2465 introduit la définition d'un champ "Ipv6Address" sur 16 octets.

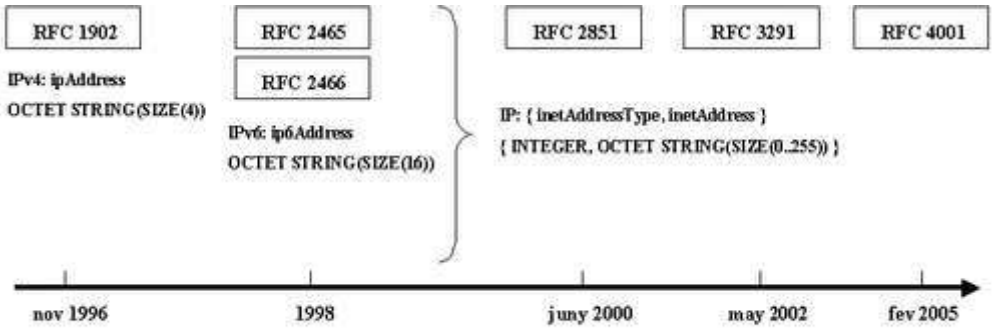


Figure 16-2 *Evolution de la MIB*

Ainsi de nombreuses RFCs ont été affectées par ces modifications. D'abord, avec la première approche de créer des MIBs IPv4 et IPv6 indépendantes, de nouveaux RFC ont vu le jour. Principalement, les RFCs décrivant les MIBs IPv6 sur transport TCP ou UDP (RFC 2452 et RFC 2454) ont été publiés en 1998. De même, une MIB concernant le protocole ICMPv6 a été décrite dans le RFC 2466.

Cependant, cette approche implique une gestion indépendante des protocoles IPv4 et IPv6. Ainsi, l'IETF a décidé de définir une MIB-2 unifiée, permettant de superviser les réseaux IPv4 et IPv6 (RFC 2851 - Juin 2000). Ce RFC définit le champ `IP ADDRESS` comme une structure avec deux champs. Le premier champ permet de différencier le type d'adresses (IPv4 ou IPv6). Le deuxième champ est une chaîne de caractères de longueur variable qui peut contenir des valeurs de longueur égale à 4 ou 16 octets (IPv4 ou IPv6). Il devient possible de définir dans les MIBs des tables compatibles avec les deux versions d'IP.

Afin d'améliorer la description de la MIB, en mai 2002, le RFC4001 obsolète les RFC 2851 et RFC 3291. Des informations supplémentaires y sont décrites comme le préfixe réseau, le numéro de port utilisé par la couche transport ou le numéro d'AS (*Automous System*) correspondant à l'adresse IPv4 ou IPv6.

Cette volonté d'avoir une MIB unifiée entraîne aussi la mise à jour des MIB déjà existantes. Actuellement, l'IETF publie des drafts de mise à jour pour les RFC 2011, RFC 2012, RFC 2013 et RFC 2096. Les dernières propositions publiées sont :

draft-ietf-ipv6-rfc2011-update-10.txt (mai 2004) (RFC Editor Queue),

RFC 4022 (Mars 2005),

RFC 4113 (Juin 2005),

draft-ietf-ipv6-rfc2096-update-07.txt (février 2004) (RFC Editor Queue).

Ces drafts concernent respectivement les MIBs IP, TCP, UDP et IP Forwarding table. Pour les MIBs concernant le routage BGP4+ et OSPFv3, les derniers drafts proposés sont `draft-ietf-idr-bgp4-mibv2-05.txt` (Juillet 2005) et `draft-ietf-ospf-ospfv3-mib-10.txt` (Décembre 2005) mais le premier de ces

drafts a expiré en janvier 2006, ce qui laisse les MIBs concernant le routage externe IPv6 en suspens. En raison de toutes ces modifications, les MIBs ne sont pas entièrement disponibles aujourd'hui. En effet, uniquement quelques réalisations ont été effectuées par les constructeurs. Nous verrons par la suite les implémentations des différents constructeurs.

2) Administration des réseaux IPv6

A) Administration d'un réseau Double Pile

Le fait d'administrer des réseaux IPv6 ne se traduit pas toujours par l'emploi d'outils qui transportent l'information en IPv6. En effet, la plupart des équipements sur le réseau étant en double pile (IPv4 – IPv6), l'accès peut se faire en IPv4 puis les informations concernant la supervision IPv6 sont récupérées. Cette solution est retenue très souvent car certaines applications ne supportent pas encore le transport en IPv6.

Cela permet aussi aux NOC (*Network Operations Center*) n'ayant pas déployé encore un plan d'adressage IPv6 pour la supervision, d'administrer des réseaux aussi bien IPv4 qu'IPv6.

B) Administration d'un réseau uniquement en IPv6

L'administration d'un réseau uniquement en IPv6 est différente de l'administration d'un réseau ayant une double pile. En effet, du fait que les équipements ne sont accessibles qu'en IPv6, toute l'information de supervision doit être récupérée via le protocole IPv6. Il est donc indispensable dans ce cas là que toutes les applications utilisées pour l'administration du réseau supportent ce protocole. L'administration d'un réseau IPv6 est donc semblable à celle d'un réseau IPv4. Les informations récupérées sont souvent les mêmes dans les deux cas. Le point essentiel reste les constructeurs ainsi que les éditeurs de logiciels qui doivent rendre disponible les protocoles ainsi que les applications permettant l'administration d'un réseau IPv6.

3) Mise en œuvre par les constructeurs

Dans cette partie, nous avons choisi un ensemble de constructeurs dont leurs équipements offrent des fonctionnalités pour la supervision des réseaux IPv6. A noter que cette liste n'est pas exhaustive. Dans un premier temps, certains constructeurs implémentant des MIBs IPv6 seront passés en revue, ensuite, dans un second temps ceux proposant l'utilisation de SNMP en IPv6 seront listé. Nous présenterons ceux qui offrent la possibilité d'exporter des flux IPv6 vers un collecteur.

Contents

A) Les MIBs

Nous avons vu précédemment que le modèle SNMP est très utilisé ; il est donc important que les constructeurs implémentent les MIBs définies par l'IETF :

- Juniper a conçu une première réalisation en mettant en œuvre une MIB privée en se basant sur le RFC 2465. Cette MIB permet en autre de faire de la métrologie sur le nombre de paquets entrants et sortants. D'autre part, ce constructeur a implémenté une MIB privée basée sur le draft JH-id qui permet de récupérer des informations sur le routage BGP4.
- Cisco a mis en œuvre sur ses équipements une MIB privée (CISCO-IETF-IP-MIB) permettant de récupérer un certain nombre d'informations IPv6 de base (interfaces, messages ICMP). Par contre, il n'a pas encore développé une MIB permettant de différencier le trafic sur les interfaces transportant les deux versions IP le trafic IPv6.
- Hitachi a développé sur ces routeurs (GR2000/GR4000) et sur les switchs (GS4000) les MIBs décrites dans les RFC 2452, RFC 2454, RFC 2465 et RFC 2466. Par contre, les MIBs unifiées ne sont pas encore prêtes.
- 6Wind a implémenté sur ses équipements les RFC 2465 et RFC 2466.

Le tableau suivant récapitule les MIBs implémentés par les principaux constructeurs.

RFC/draft de référence	Nom de la MIB
Cisco	RFC 2466 cisco-ietf-ip-mib (privé)
Juniper	RFC 2465 RFC 2466 draft ietf-idr-bgp4-mibv2-0 mib-jnx-ipv6.txt (privé)

	mib-jnx-ipv6.txt (privé) mib-jnx-bgpmib2.txt (privé)
Hitachi	RFC 2452 RFC 2454 RFC 2465 RFC 2466 MIBs standard implémentées
6wind	RFC 2465 RFC 2466 MIBs standard implémentées

B) Le transport IPv6 du protocole SNMP

Afin que le transport de SNMP puisse se faire en IPv6, il est nécessaire que le serveur SNMP ainsi que les agents situés sur les équipements supportent la pile IPv6. En ce qui concerne les serveurs, il existe un nombre important d'application qui supporte déjà le transport IPv6. De même, chez un grand nombre de constructeurs, les agents SNMP sont capables de répondre à des requêtes IPv6.

Constructeur	Agent SNMP IPv6
Cisco	A partir de la version d'IOS 12.0 (27)S
Juniper	Disponible
Hitachi	Disponible
6wind	Disponible

C) L'export des flux IPv6

Figure 16-3 *Exportation des flux*

Un autre élément très important de nos jours dans la supervision des réseaux est l'export des flux. En effet, les flux permettent de caractériser le trafic acheminé sur le réseau. Il est donc essentiel de disposer de cette fonctionnalité d'export pour les flux IPv6.

Le groupe de travail IPFIX de l'IETF travaille pour définir un format standard pour l'export des flux. Cependant, aujourd'hui chaque constructeur a développé sa propre technologie. Voici un tableau indiquant les différentes technologies utilisées par chaque constructeur et si elles supportent l'export de flux IPv6.

Constructeur	technologie pour l'export des flux	Export des flux IPv6
Cisco	Netflow (version 9)	A partir de la version d'IOS 12.3(7)T
Juniper	Cflowd	Non disponible
Hitachi	sflow	Disponible
6wind	Netflow	Non disponible

4) Les plates-formes.

Les plates-formes de supervision disponibles dans le marché visent à intégrer un ensemble d'outils permettant d'administrer de façon globale le réseau. Vu que tout les éléments de base ne sont pas disponibles, il est difficile de trouver aujourd'hui une plateforme qui permette de superviser un réseau IPv6 de la même façon qu'IPv4. Cependant, certaines versions de ces plateformes disposent de quelques fonctionnalités de supervision IPv6. Voici un tableau récapitulant un ensemble de fonctionnalités avec les différentes plateformes d'administration.

Plate-forme	outil	Fonctionnalités disponibles
HP OpenView	Network Node Manager (NMM 7.5)	Implémentation de MIBs IPv6 standard Découverte de réseau IPv6
CiscoWorks	Campus manager 4.0	Transport IPv6 : trace d'équipements terminaux IPv6 En phase de test
	Resource Manager Essentials (RME)	Inventaire et configuration d'équipements En phase de test
	Cisco View	En cours de développement
Tivoli-Netview		Pas de fonctionnalité IPv6 disponible

Infovista		Pas de fonctionnalité IPv6 disponible et aucune prévision d'en intégrer.

5) Les applications spécifiques

La plupart des constructeurs n'ayant implémenté qu'une partie des MIBs définis par l'IETF et les plateformes de supervision n'ayant pas encore intégré la totalité des fonctionnalités nécessaires pour administrer convenablement un réseau IPv6, la supervision de certains services restent impossible avec les applications existantes. Aujourd'hui, pratiquement tous les langages de programmation supportent la pile IPv6, il est donc facile de développer ses propres scripts pour répondre à ces besoins. Par exemple, pour connaître à intervalle régulier le nombre de routes IPv6 reçus sur les peerings BGP d'un équipement, vu que les MIBs BGP4+ ne sont quasiment pas implémentées par les constructeurs, il est difficile de récupérer cette information via SNMP (ce qui est fait généralement en IPv4). Une façon de palier ce manque est de mettre en place un script qui se connecte régulièrement sur l'équipement en Telnet ou SSH et via des CLI (*Command Line Interface*) récupère cette information (par exemple, "`show ipv6 bgp summary`"). Il est clair que cette méthode est moins optimale qu'une simple récupération par SNMP (charge de la CPU plus importante, temps de réponse plus lent avec SSH ou Telnet que SNMP) mais elle permet cependant d'obtenir l'information souhaitée.

A) Exemples d'outils existants

Actuellement, il existe un éventail d'outils permettant d'administrer des réseaux IPv4. Il est nécessaire de disposer également d'un certain nombre d'instruments pour administrer les réseaux IPv6. Quelques outils existants en IPv4 sont déjà disponibles en IPv6 et de nouveaux outils sont développés pour répondre à des besoins spécifiques au protocole IPv6. Les chapitres suivants donnent une liste non exhaustive d'outils sont disponibles aussi bien pour les sites terminaux que pour les réseaux d'opérateurs. Certaines applications utilisées pour superviser les réseaux multicast en IPv6 sont également listées.

Outils pour réseaux Locaux :

- Argus est un logiciel libre permettant de superviser les différents équipements d'un réseau local (switch, routeurs, serveurs, stations de travail) ainsi que les applications associées (connectivité, applications TCP, UDP comme ping, ssh, ftp, http, dns,...) via une interface web. Ce logiciel supporte IPv6 depuis la version 3.2. Son fonctionnement modulaire permet à un administrateur d'ajouter des fonctionnalités qui ne seraient pas disponibles dans un premier temps. Le logiciel est aussi capable de générer des notifications d'alertes (mail, pager).

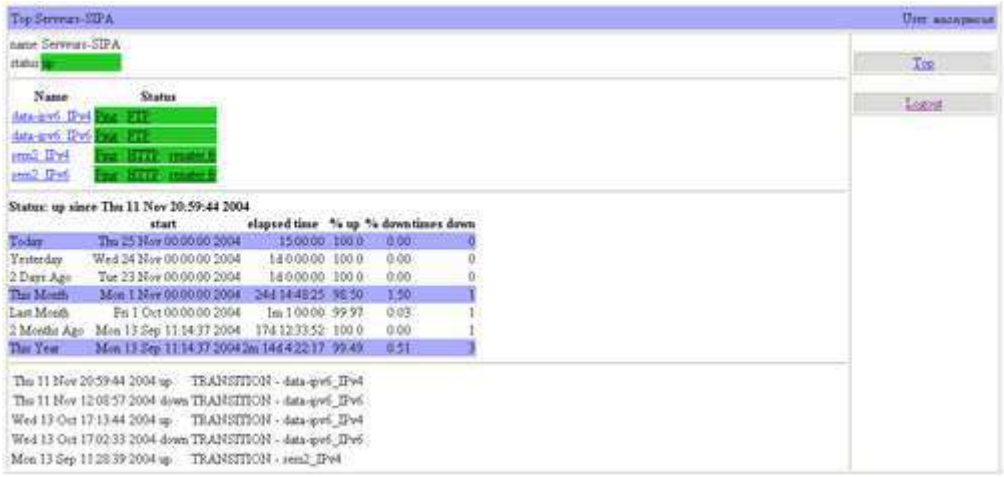


Figure 16-5 *Argus*

- Ethereal est un logiciel libre qui permet d'analyser le trafic en temps réel en distinguant le trafic IPv4 et IPv6 sur une interface. Il intègre un grand nombre de protocoles des couches réseau et transport mais aussi des couches applicatives.

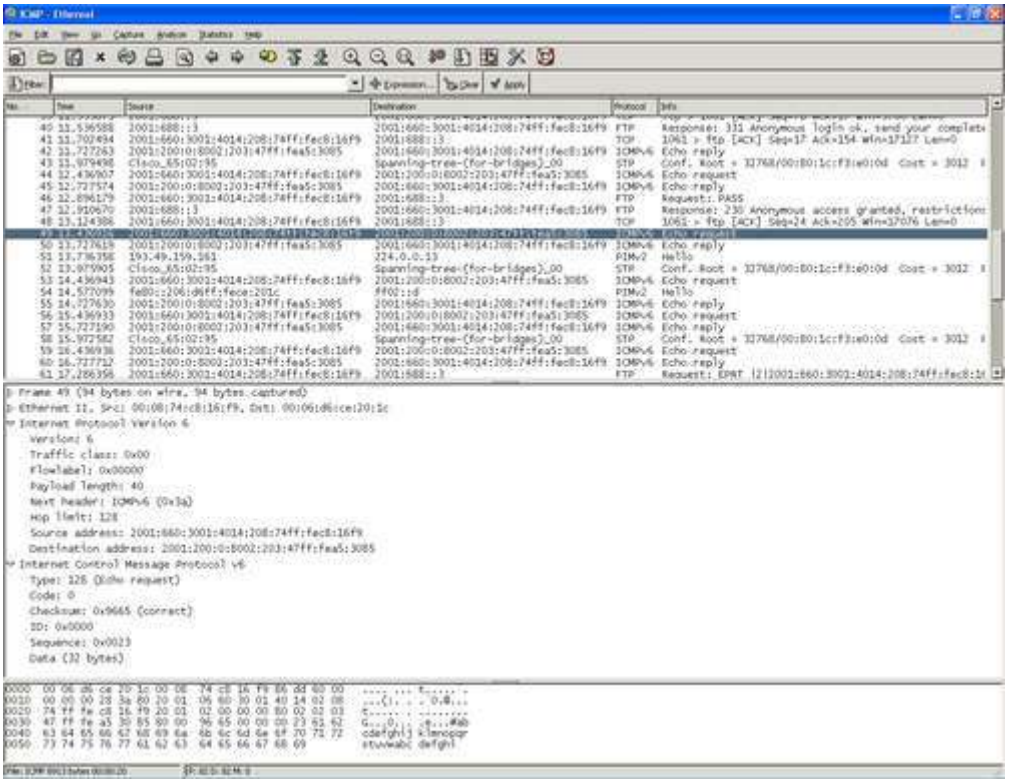


Figure 16-6 *Ethereal*

- NDPMon (Neighbor Discovery Protocol Monitor) est un outil permettant de superviser les activités du protocole Neighbor Discovery sur un réseau afin de détecter les annonces erronées ou malicieuses.

Le protocole Neighbor Discovery est utilisé par les tous noeuds IPv6 pour s'autoconfigurer puis interagir entre eux grâce aux fonctionnalités de découverte inhérentes au protocole. Une mauvaise configuration (volontaire ou non) d'un hôte peut entraîner la diffusion d'annonces erronées, qui peuvent facilement conduire à perturber le réseau en simulant de différentes manières des attaques de type DoS, flooding ou redirection. Certaines annonces perturbent la découverte des voisins anciennement ARP) conduisant à des

problèmes d'identification des noeuds du segment (table des voisins erronnée, non détection d'un noeud hors ligne, usurpation d'identité). D'autres annonces peuvent perturber le mécanisme de routage en empêchant le trafic d'un segment d'être acheminé vers le routeur légitime.

NDPMon est un outil de surveillance pour IPv6 fonctionnant sur le modèle d'Arpwatch. Il répond au besoin des administrateurs réseau de disposer d'un outil facilement déployable permettant de surveiller les faiblesses du protocole Neighbor Discovery. NDPMon détecte les paquets ICMPv6 maliciceux en analysant les paquets capturés sur le réseau et en comparant ces annonces avec sa base de connaissance. Celle-ci est constituée d'une base de données des voisins établie par une phase d'apprentissage et d'un fichier de configuration établi en partie par l'administrateur et pouvant être complété durant cette phase préliminaire. Le logiciel peut détecter différents types d'anomalies et déclencher des alertes spécifiques (email, syslog ou autres alertes scriptées).

Le logiciel a été développé au sein de l'équipe Madynes du LORIA, par Thibault Cholez et Frederic Beck, et est diffusé sous license LGPL.

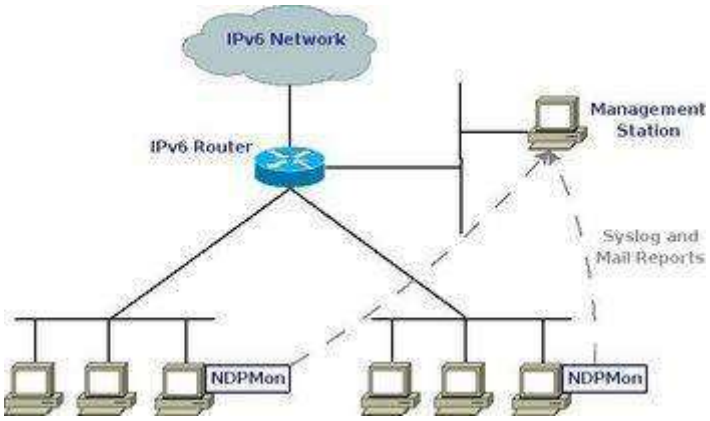


Figure 16-7 *Déploiement de NDPMon*

Outils pour réseaux d'opérateurs :

- ASPath-Tree est basé sur des captures régulières de la table BGP IPv6, ASpath-tree produit automatiquement un ensemble de pages HTML fournissant une vue graphique des chemins pour atteindre les autres AS sous forme d'arbre.
  A la base conçu pour être employé par des sites IPv6 impliqués dans l'expérimentation du protocole BGP à l'intérieur du 6Bone, il peut être aujourd'hui utilisé dans n'importe quel réseau IPv6 opérationnel qui utilise BGP comme protocole de routage. De plus, il permet la détection des entrées anormales de routes annoncées par BGP (les préfixes interdits ou non agrégés), des numéros d'AS erronés (réservés ou privés) et fournit un ensemble d'information complémentaire comme:
  - le nombre de routes (valid/total/suppressed/damped/history),
  - le nombre d'AS dans la table (total, originating only, originating/transit, transit only, private and reserved),
  - le nombre de voisins actifs BGP (c'est-à-dire annonçant des routes),
  - une analyse de la taille de réseau, en termes de distance inter-AS,
  - le nombre de préfixes circulant dans le réseau (total, 6Bone pTLAs, sTLAs, 6to4, autres).

Un des grands avantages d'ASPath-Tree est qu'il permet de connaître rapidement les chemins empruntés par les paquets pour atteindre un AS en se basant uniquement sur un routeur du réseau (même AS). Ceci simplifie beaucoup la mise en place de l'outil. Cependant, ceci peut aussi présenter un inconvénient : Normalement, l'arbre devrait être identique quel que soit le routeur interrogé au sein du réseau. Or, si la configuration est mal faite (voisin mal déclaré,...), l'arbre ne sera plus le même selon le routeur interrogé.
ASPath-tree facilite donc la mise en place d'une politique de routage au niveau d'un backbone.

Cet outil est spécifique à IPv6. En effet, le nombre de routes présentes dans les tables de routage en IPv6 est de l'ordre de 500 alors qu'en IPv4 ce nombre dépasse 150 000, rendrant la gestion de l'outil beaucoup plus lourde.

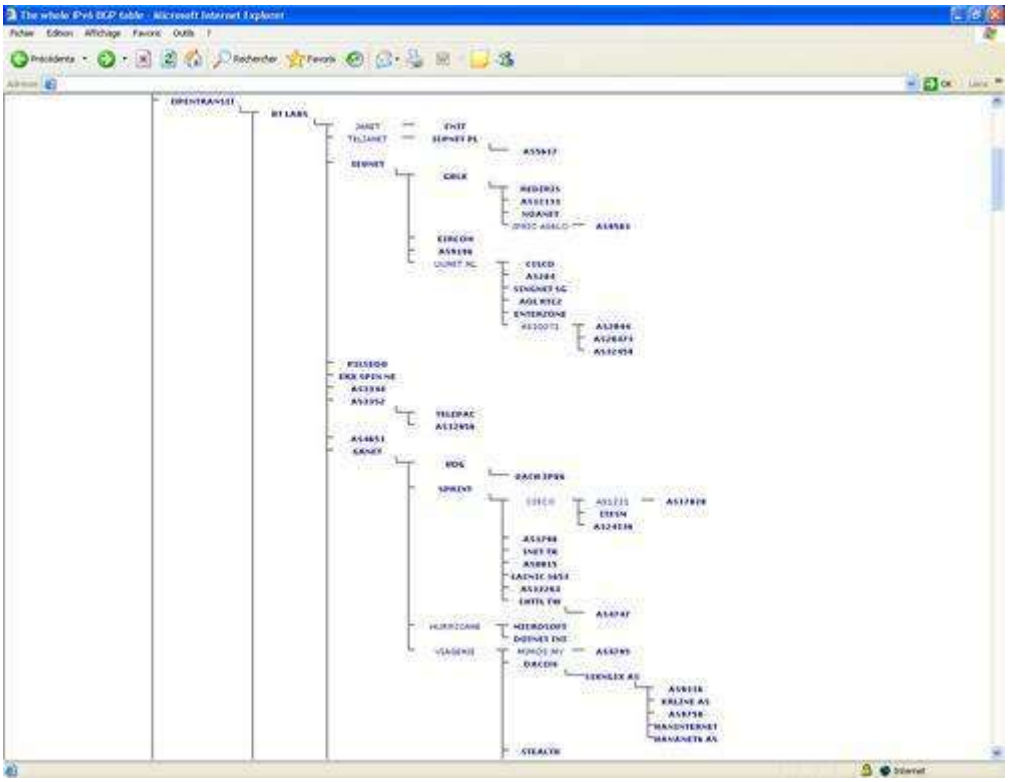


Figure 16-8 *ASTree*

- Looking Glass : Cet outil permet, à travers une interface WEB, d'avoir un accès direct sur les routeurs ou les commutateurs. L'utilisateur dispose d'une liste de commandes qui peuvent être exécutées. Des scripts CGI permettant la connexion Telnet ou SSH (en IPv4 ou IPv6) récupèrent l'information désirée et l'affichent sur une page WEB.
  Il permet d'obtenir des informations directement sur des équipements réseaux en temps réel sans avoir un compte dédié sur ces équipements. Cependant, le fait de donner des informations sur les équipements réseaux oblige à mettre en place un certain nombre de mesures de sécurité pour préserver la confidentialité des informations (accès restreint au serveur web…).

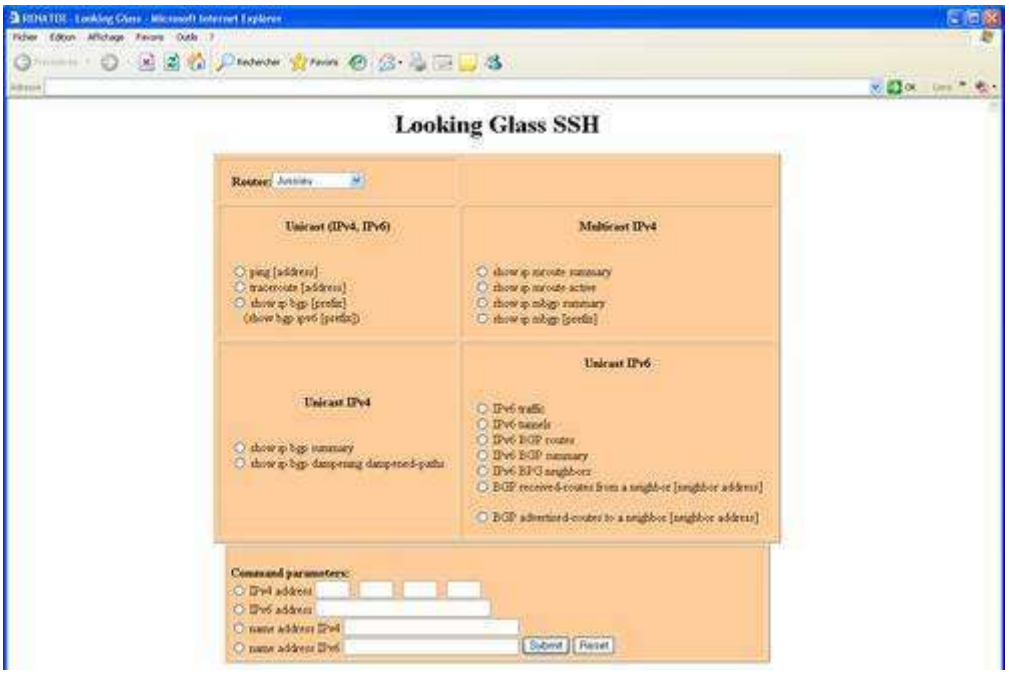


Figure 16-9 *Looking Glass*

- MRTG permet de générer des graphes sur le trafic réseau. Une version IPv6 de MRTG a été développée par l'université de Rome-3. Elle permet d'interroger les équipements via SNMP sur un transport IPv6. D'autres outils comme RRDtool, disponible en IPv6 permettent de faire de la métrologie.

- Weather map permet de présenter une carte topologique du réseau avec son trafic actuel. Il se base généralement sur des applications comme MRTG pour pouvoir afficher les graphes de trafic entre deux liens du réseau. Pour obtenir une weather map représentant uniquement le trafic IPv6, il est nécessaire que les MIB des routeurs distinguent bien le trafic IPv4 et IPv6. Or, actuellement, tous les constructeurs ne proposent pas cela. Des réseaux expérimentaux où le trafic est exclusivement IPv6, comme 6NET, peuvent disposer de cartes affichant la charge de trafic IPv6. En ce qui concerne le transport, il peut être réalisé entièrement en IPv6.

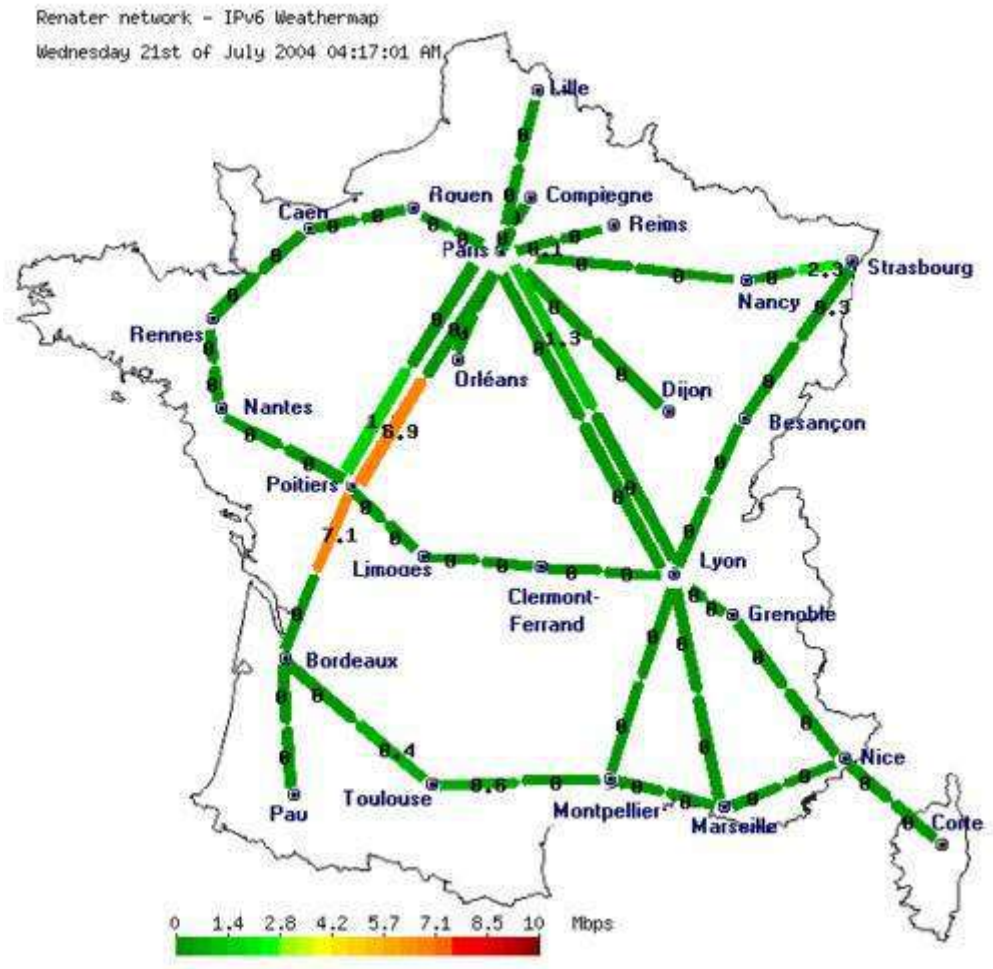


Figure 16-10 *Weather map*

Outils pour les réseaux multicast :

- Mping est un outil qui se base sur les fonctionnalités de ping6. Il permet de tester la connectivité entre plusieurs interfaces IPv6 et peut réaliser des mesures de performances entre elles. Il réalise une collecte de données qui lui permet de générer des rapports et des histogrammes.
  Cet outil reste pratique tant que la taille du réseau n'est pas très importante. Lorsque le réseau grandit, le nombre de requêtes est multiplié. Ainsi, le trafic ICMP généré est de plus en plus important, ce qui peut faire augmenter la CPU des équipements réseaux.
- Multicast beacon est une application client/serveur, réalisée en java, donnant des matrices de mesures sur le trafic Multicast en IPv4 et IPv6. Chaque client possède un démon beacon. Le démon envoie un message périodiquement aux autres membres du groupe multicast pour mesurer les pertes, le délai, la gigue, les doublons et le mauvais séquencement des paquets. Ces informations sont envoyées au serveur beacon qui peut afficher ces informations dans une table.

6) Conclusion

Aujourd'hui, un administrateur dispose d'un ensemble d'outils permettant d'administrer des réseaux IPv6. Cependant, un effort reste à faire pour atteindre le même niveau de supervision qu'en IPv4. Les récentes normalisations réalisées par des organismes comme l'IETF commence à être mises en œuvre par les constructeurs d'équipements et par les éditeurs de logiciels de supervision. Ceci permet de se rapprocher d'un modèle de supervision standard.

XVII) Historique de la standardisation d'IPv6

Cette annexe décrit les principales étapes qui jalonnent la standardisation d'IPv6, depuis la prise en compte de la nécessité d'un nouveau protocole jusqu'à la production de standards. Ce processus est toujours en cours même si une grande partie du chemin a déjà été parcourue.

Contents

1) La standardisation de l'Internet

Il est difficile de comprendre comment est né IPv6, et comment le protocole continue d'évoluer, si l'on ne connaît pas le processus de standardisation des protocoles utilisés dans l'Internet. Les organismes qui pilotent la standardisation de l'Internet sont l'IETF, l'IESG, l'IAB, l'ISOC et l'ICANN. Le résultat final de l'activité de standardisation est la production de documents appelés RFC (Request for Comments) dont certains sont des standards. L'ensemble du processus de standardisation est décrit dans le RFC 2026.

A) L'IETF (*Internet Engineering Task Force*)

L'activité au sein de l''ETF est organisée en groupes de travail (*working group*). Les groupes agissant autour d'un même thème sont regroupés dans un Domaine (Area), comme le routage, la gestion des réseaux, la sécurité... Chaque Domaine est coordonné par un directeur (Area Director). L'ensemble des directeurs de Domaines compose l'IESG (voir ci-après). Les groupes de travail de l'IETF sont constitués de personnes volontaires et bénévoles qui sont motivées pour faire évoluer les standards de l'Internet sur la base de leur expérience acquise à partir de mises en œuvre et d'essais concrets. Toute personne intéressée (et ayant des disponibilités) peut faire partie d'un groupe de travail IETF. La participation à l'IETF et à ses groupes de travail est donc le fait d'individus proposant des contributions techniques plutôt que de représentants formels d'organisations. Chaque groupe de travail définit ses objectifs dans une charte, qu'il soumet à l'IESG lors de son processus de constitution. Le groupe est dirigé par un ou plusieurs présidents (*Working Group Chairs*). Le fonctionnement des groupes de travail de l'IETF est décrit en détail dans le RFC 2418. Dans un groupe le travail, les échanges de points de vue et d'expériences visant à élaborer les projets de standards (*Internet Drafts*), se font essentiellement par courrier électronique. Trois réunions annuelles permettent aux membres des groupes de se rencontrer pour une interaction plus directe. Lorsqu'un consensus est atteint, le groupe en demande la publication en tant que RFC (*Request For Comments*). Quand un groupe considère que ses objectifs sont atteints, il se dissout. Un certain nombre de groupes et sous-groupes IETF travaillent actuellement sur IPv6 ; on peut citer notamment:

- IPng (aujourd'hui IPv6) : principal groupe de travail sur la production de protocoles liés à IPv6.
- NGtrans (aujourd'hui v6ops) : étudie les modalités de la migration d'IPv4 vers IPv6. Ce groupe gère le déploiement du 6bone, un réseau expérimental IPv6 interconnectant des sites de test à travers le monde.

Les groupes de travail de l'IETF doivent faire preuve d'esprit de coopération, ainsi que d'un haut degré de maturité technique ; les participants à l'IETF reconnaissent que les plus grands bénéfices pour tous les membres de la communauté de l'Internet proviennent du développement coopératif des protocoles et services (voir http://www.ietf.org pour plus de détails).

B) L'IESG (*Internet Engineering Steering Group*)

L'IESG est responsable de la direction des activités techniques de l'IETF. Il est composé des directeurs de Domaines (il y en a 7 actuellement) et du président de l'IETF qui est aussi le président de l'IESG. L'IESG administre le processus de standardisation de l'Internet suivant des règles et procédures détaillées dans le RFC 2028. C'est donc l'IESG qui décide de l'évolution du statut des spécifications techniques émises par les groupes de travail, ce qui peut élever certaines d'entre elles au rang de Standard Internet. L'IESG approuve également la constitution des nouveaux groupes de travail.

C) Les RFC (*Request For Comments*)

Les RFC sont les documents officiels de l'Internet ; ils sont disponibles gratuitement sur le réseau. En France, ils sont notamment disponibles sur le site miroir officiel primaire ftp://www.imag.fr/pub/archive/IETF/rfc/. Un RFC est complètement identifié par le numéro qui lui est attribué lors de sa publication. Ce numéro n'a pas de signification ; il est attribué par ordre chronologique de publication. Les premiers RFC sont sortis en 1969 ; début 2006, nous en sommes au numéro 4400, pour un flux supérieur à 200 RFC par an. Il existe plusieurs types de RFC d'importance différente pour la communauté IETF. On distingue deux catégories principales : ceux faisant partie du processus de standardisation (*Standard Track*) et les autres.

Les documents entrant dans la première catégorie suivent un parcours bien défini. Ils sont d'abord publiés comme "Internet Drafts" par le groupe de travail. Ce sont des documents de travail à durée de vie limitée, disponibles en ligne sur les mêmes serveurs que les RFC. Lorsqu'un consensus semble atteint, le responsable lance un appel aux derniers commentaires dans le groupe. Il transmet ensuite le document à l'IESG qui, à son tour, lance un appel aux derniers commentaires à tout l'IETF. S'il n'y a pas d'objections majeures, le document est alors publié comme RFC avec un statut de (*Proposed Standard*).

Après une période où se succèdent mises en œuvre et retours d'informations sur le protocole décrit, le groupe de travail auteur revoit le document.

Si seuls des changements mineurs sont nécessaires, il demande alors à l'IESG de le faire avancer en le publiant en tant que nouveau RFC mais avec un statut de Draft Standard) après un nouvel appel à commentaires. Si les changements sont majeurs, le nouveau RFC garde le statut de Proposed Standard.

Suit alors une nouvelle période de mises en œuvre et de retours d'informations où l'on doit faire la preuve de l'interopérabilité de deux implantations indépendantes. Puis le responsable du groupe de travail demande à l'IESG de faire avancer le document en le publiant (après un dernier appel à commentaires) en tant que RFC avec un statut de type "Standard". Il faut noter qu'à chaque étape, un nouveau numéro de RFC est attribué. Un RFC peut passer à l'état "historique" si son utilisation est devenue déconseillée. S'il est remplacé par un autre il devient "obsolète".

La publication d'un RFC hors du processus de standardisation est plus facile. L'IESG peut directement faire publier un RFC avec un statut de type "Information", "Experimental" ou "*Best Current Practice*". Un RFC

"Information" documente un protocole ou une approche particulière d'un problème. Il peut aussi donner des informations d'intérêt général. Il n'a aucune valeur de standard. Un RFC "Expérimental" décrit un protocole nécessitant que l'on mène des expériences avant de décider ou non d'entrer dans le processus de standardisation. Un RFC "*Best Current Practice*" documente une pratique d'ingénierie recommandée. La répartition par types des RFC édités en 2001 est donnée dans figure 17-2.

D) L'ISOC (Internet Society)

L'ISOC est une organisation internationale s'occupant de la croissance et de l'évolution de l'Internet à l'échelle mondiale et des aspects sociaux, politiques et techniques de son utilisation. Les membres de l'ISOC sont des personnes physiques ou morales. L'ISOC est dirigée par un Bureau des Administrateurs élu par les membres individuels. La standardisation de l'Internet est une activité chapeautée par l'ISOC, le Bureau des Administrateurs étant responsable de l'approbation des procédures et des règles du processus de standardisation de l'Internet. Le moyen selon lequel les membres du Bureau des Administrateurs de l'ISOC sont choisis, et les autres aspects concernant le fonctionnement de l'Internet Society sont décrits dans le document ISOC By-Laws [RFC 2135] et http://www.isoc.org.

E) L'IAB (Internet Architecture Board)

L'IAB est chargé par les Administrateurs de l'ISOC de fournir une supervision de l'architecture de l'Internet et de ses protocoles. L'IAB désigne le président de l'IETF et approuve les autres candidats pour l'IESG, présentés par le comité de nomination. L'IAB est aussi responsable de l'examen et de l'approbation des chartes des nouveaux groupes de travail proposés.

L'IAB supervise le processus utilisé pour créer des Standards Internet et sert de cour d'appel pour les réclamations comme par exemple une violation des procédures de standardisation, ou pour un conflit entre les groupes de travail de l'IETF et l'IESG. L'IAB conseille l'IETF et l'ISOC en matière technique, politique, d'architecture, de procédure se rapportant à l'Internet et aux technologies associées. Les membres de l'IAB sont nommés par un comité (le Nomcom) et approuvés par le conseil d'administration de l'ISOC.

F) L'ICANN (Internet Corporation for Assigned Names and Numbers)

L'ICANN est un organisme international créé en octobre 1998 pour remplacer l'IANA (Internet Assigned Numbers Authority) qui n'était contrôlé que par le gouvernement américain. Beaucoup de spécifications de protocoles comportent des nombres, mots clés et paramètres qui doivent être affectés de manière unique, par exemple les numéros de versions, les numéros de protocoles, les numéros de ports et de MIB (Management Information Base). C'est l'ICANN qui affecte les valeurs de ces paramètres pour l'Internet. L'ICANN publie aussi les tables de toutes les valeurs affectées dans des RFC nommés Assigned Numbers. L'ICANN sert aussi de "sommet de la pyramide" pour la gestion des domaines de noms (DNS) et l'affectation des adresses et en établit les règles.

G) Les RIR (Regional Internet Registries)

Les RIR ou RIAR (*Regional Internet Address Registries*) reçoivent une délégation de l'ICANN pour distribuer les adresses IPv4 et IPv6. Ils sont au nombre de quatre actuellement, répartis sur 4 continents pour assurer une gestion de "proximité" à cette échelle. Ce sont :

- APNIC : Asia Pacific Network Information Centre,
- ARIN : American Registry for Internet Numbers,
- RIPE-NCC : Réseaux IP Européen (Network Coordination Centre),
- LACNIC : Réseaux d'Amérique Latine et des Caraïbes.

Un cinquième, l'AFRINIC destiné à assurer la couverture du continent Africain est en cours de gestation (l'Afrique dépend actuellement du RIPE et de l'APNIC). En 2001, sept préfixes IPv4 de longueur 16 ont été affectés aux RIR pour être distribués aux opérateurs

2) La standardisation d'IPv6

Contents

Dans cette première phase, commencée en août 1990 lors de la réunion IETF de Vancouver, le taux de croissance continu de l'Internet met en évidence le gaspillage des adresses dû à la structure en classes de l'adressage IPv4. L'activité a surtout consisté à quantifier le problème et à proposer quelques solutions à plus ou moins court terme. Un certain nombre de sites ne peuvent plus se contenter d'un simple réseau de classe C pour les équipements qu'ils ont à mettre en réseau. Leurs besoins d'adressage sont intermédiaires entre un réseau de classe C (254 adresses disponibles) et un réseau de classe B (65534 adresses disponibles). Une simulation montre alors que si chacun de ces sites fait une demande de réseau de classe B, compte tenu de la vitesse de croissance de l'Internet, il n'y aurait plus eu de réseaux de classe B disponibles pour de nouvelles allocations vers mars 1994.

La solution adoptée, qui consiste à attribuer plusieurs réseaux de classe C à un même site va générer un autre problème qui est la saturation de la mémoire disponible pour maintenir les tables de routage sur les routeurs principaux de l'Internet.

La question de fond commence alors à émerger : faut-il conserver le protocole IP en l'adaptant tant bien que mal aux exigences de l'évolution de l'Internet et en acceptant les contraintes (en particulier la limitation de la croissance à cause de la pénurie prévisible d'adresses), ou bien opter pour la modification de la structure des adresses et refondre le standard IP en acceptant le bouleversement que cela va provoquer ?

En novembre 1991, lors de la réunion IETF de Santa Fe, les groupes de travail ROAD (*Routing and Addressing*) et ALE (*Address Lifetime Expectations*) sont créés. Ils sont chargés d'étudier les trois problèmes suivants pour guider l'IETF dans ses choix :

- la pénurie des réseaux de classe B,
- la croissance des tables de routage des routeurs principaux,
- la pénurie des adresses de machines.

A) L'émergence des premières idées

En mars 1992, lors de la réunion IETF de San Diego, le groupe ROAD rend compte de ses travaux et propose pour le court terme d'adopter CIDR (*Classless Inter-Domain Routing*) qui règle provisoirement le problème de la croissance des tables de routage par l'agrégation des préfixes contigus.

Pour le long terme, il propose de faire un appel à propositions et de former des groupes de travail chargés d'étudier différentes approches possibles pour un nouveau protocole IP doté d'un espace d'adressage plus grand qu'IPv4. Quatre groupes sont alors au travail avec des propositions sérieuses, il s'agit de :

- PIP : The 'P' Internet Protocol, qui introduit la notion d'identificateur et de localisateur,
- TUBA : TCP and UDP with Bigger Addresses ([RFC 1347]) basée sur CLNS,
- IPAE : IP Address Encapsulation, basée sur une extension des adresses IPv4,
- SIP : Simple Internet Protocol, version simplifiée de IPv4 avec des adresses à 64 bits.

Parallèlement à ce travail, l'IAB produit une version 7 du protocole IP basée sur CNLP (*Connectionless Protocol*) de l'ISO.

D'autre part, plusieurs propositions indépendantes ont vu le jour. On peut citer notamment CATNIP [RFC 1707], proposant une standardisation de l'interface entre les couches réseau et transport. Après discussions, l'IETF décide de rejeter la proposition de l'IAB et adopte les propositions du groupe ROAD.

En juillet 1992, lors de la réunion IETF de Cambridge, un appel à propositions est donc lancé à la communauté mondiale pour définir les caractéristiques de l'IP de nouvelle génération (IPng).

B) Définition du cahier des charges du nouvel IP

En novembre 1992, lors de la réunion IETF de Washington, CIDR est adopté par l'IESG pour répondre au problème de la croissance trop rapide des tables de routage. Il sera rapidement inclus dans les protocoles de routage externe comme BGP4.

L'IESG publie le RFC 1380 "Délibérations de l'IESG pour le routage et l'adressage". Ce document fixe un premier cahier des charges que devront respecter les propositions pour le nouveau standard IP, et une grille d'évaluation qui sera utilisée pour les comparer. Ainsi, le nouveau standard devra être capable :

- d'adresser au moins un milliard de réseaux,
- de proposer un plan de transition "sans jour J",
- de prendre en compte à terme la mobilité, la réservation de ressources, les hauts débits, etc.

Les propositions devront préciser les changements induits sur :

- les machines,
- les routeurs intérieurs et extérieurs,
- la sécurité et l'authentification,
- la gestion du réseau et les outils (ping, traceroute, etc.),

- les modes opératoires et les procédures d'administration,
- les autres protocoles (ARP, RARP, IP, ICMP...).

Et étudier :

- l'impact sur les performances et la sécurité,
- un schéma complet de routage et d'adressage,
- un plan de déploiement.

Tous les groupes sont invités à présenter leurs propositions pour la réunion IETF de mars 1993 à Columbus. Lors de cette réunion, les groupes IPAE, PIP, SIP, TUBA présentent leurs premiers travaux et parfois des premières implémentations de leurs protocoles. Les groupes SIP/IPAE fusionnent en gardant le nom de SIP. Mais la compétition entre les groupes qui défendent chacun leur solution est très forte et tend à disperser les efforts tout en ralentissant le processus d'émergence du nouveau standard. Aussi en juillet 1993, lors de la réunion IETF d'Amsterdam, et sous la pression des acteurs industriels, l'objectif est redéfini : il faut arriver rapidement à produire un standard minimum fondé sur tous les éléments techniques pour lesquels il y a consensus. L'IESG est également chargé de clarifier le processus de sélection. Pour limiter la dispersion des efforts due à la concurrence de groupes de travail rivaux, un domaine IPng (IP next generation) est formellement créé pour coordonner leur action [RFC 1454]. En novembre 93, lors de la réunion IETF de Houston, le groupe ALE présente des projections de croissance de l'Internet et propose les mesures suivantes pour prolonger la vie d'IPv4 :

- récupération des numéros de réseaux assignés non utilisés ou sous-utilisés,
- durcissement de la politique d'allocation de réseaux,
- incitation des sites largement pourvus à renuméroter pour restituer une partie de leurs numéros de réseaux ou pour revenir dans l'espace d'adressage de leur fournisseur d'accès.

Lors de cette même réunion, les groupes de travail PIP et SIP fusionnent dans le groupe SIPP (Simple Internet Protocol Plus [RFC 1710]). L'IESG publie un livre blanc ([RFC 1550]) qui est un appel à propositions pour définir les critères qui serviront à apprécier les propositions des différents groupes de travail sur le nouveau protocole IP. En mars 1994, lors de la réunion IETF de Seattle, le groupe ALE dans son étude sur la croissance du nombre de machines connectées à l'Internet prédit qu'il n'y aura plus d'adresses disponibles en 2008 (à 3 ans près) sur la base du taux de croissance actuel. Il propose de standardiser quelques numéros de réseaux non routables pour les utiliser dans les réseaux IP privés (Intranets). Parallèlement, une forte incitation sera lancée pour utiliser l'agrégation de routes au moyen du *Classless Interdomain Routing* (CIDR).

L'appel du RFC 1550 sera entendu, et un document définissant les critères de sélection du nouveau protocole IP parmi les candidats restants a pu être élaboré à partir des réponses obtenues. Un appel à commentaires est lancé avec comme objectif de pouvoir disposer d'une version définitive lors de la prochaine réunion de l'IETF en juillet 1994 à Toronto.

C) Le choix d'IPv6

Lors de la réunion IETF de Toronto, les projets de protocoles TUBA, CATNIP, SIPP qui ont été finalisés courant 1994 sont analysés. Il en ressort les principaux éléments suivants :

- CATNIP est bien considéré, mais trop jeune. Le risque lié à un protocole très nouveau et le manque de plan de transition le font rejeter.
- TUBA est assez globalement rejeté à cause de la dualité IETF/ISO.
- SIPP, qui est devenu SIPP+ quand la taille de ses adresses a été portée à 16 octets, est adopté : la philosophie principale d'IPv4 est conservée et le nombre de champs de l'en-tête est moindre.

Recommandation "impérative" est faite d'adopter le protocole SIPP+ comme successeur d'IPv4.

Le choix de SIPP+ comme nouveau protocole a aussi le mérite d'arrêter la compétition entre groupes de travail rivaux, et de concentrer les efforts autour d'un même projet. Car si le format du paquet IP nouveau est défini, beaucoup de choses restent à faire, et notamment définir la structure de l'adressage qui reste un problème majeur non résolu à cette époque.

Fin 1994, le nom définitif du protocole est adopté, ce sera IPv6 (la version 5 ayant déjà été attribuée à un protocole expérimental : ST-II, [RFC 1819] et la version 7 réservée au transport sur CLNP). L'IESG approuve alors la création de deux nouveaux groupes de travail :

- IPng (*IP next generation*) va devoir définir complètement les spécifications du nouveau protocole sur les bases de SIPP, en respectant le cahier des charges élaboré.
- NGTRANS (*IPng Transition*) qui s'attèle au problème de la migration de l'Internet d'IPv4 vers IPv6.

En même temps, les premières implémentations du protocole vont permettre des tests à plus grande échelle, en interconnectant les plates-formes de test IPv6 intra- et inter- continent.

D) Des premières implémentations au démarrage du 6bone

Après la publication d'une nouvelle version des recommandations pour le nouveau protocole IP [RFC 1752], en janvier 1995 et la création de l'enregistrement de type AAAA pour supporter les adresses IPv6 dans le DNS d'IPv4 en février, les trois premières souches IPv6 sont produites par DIGITAL, l'INRIA, et le NRL (*US Naval Research Laboratory*). Le 30 mars 1995, les premiers paquets IPv6 sont échangés entre des machines utilisant ces codes.

Les choses vont s'accélérer à partir de la réunion IETF de fin avril 1995 à Denver qui donne lieu à première diffusion des souches IPv6 existantes. Les implémentations d'IPv6 se multiplient, et début juin 1995, la liste des systèmes supportant IPv6 est la suivante : NetBSD (INRIA), BSDI (NRL), DIGITAL Unix, HP-UX (SICS).

La fin de l'année 1995 va voir un grand nombre de tests et démonstrations d'interopérabilité d'IPv6 se dérouler. En particulier :

- juillet 1995 : interopérabilité démontrée à la réunion IETF de Stockholm.
- 8-10 août 1995 : TCP/IP expo show (SICS, Mitre, INRIA, HP, DIGITAL).
- 27-29 septembre 1995 : démonstration publique de machines échangeant du trafic IPv6 à l'Atlanta User Interop.
- 7 décembre 1995 : rencontre des implémenteurs IPv6 à la réunion IETF de Dallas.

La plupart de ces tests d'interopérabilité ont eu lieu sur un même réseau local. La suite logique est de vouloir étendre ces tests dans un contexte de réseau étendu pour tester les équipements et protocoles de routage.

C'est ainsi que naît l'idée du G6 en France. Ce groupe se donne un double objectif :

- regrouper les expérimentateurs d'IPv6 en France en les aidant à partager leur expérience et en coordonnant des actions communes.
- faire connaître et promouvoir le proctocole IPv6.

Il tient sa première réunion le 20 décembre 1995.

Au plan international, c'est à la réunion IETF de Los Angeles en mars 1996 que germe l'idée de créer un réseau mondial de machines implémentant IPv6. Il est appelé le 6bone. Lors de la réunion IETF suivante à Montréal en Juillet 1996, il est imaginé de construire le 6bone à partir d'un ensemble de tunnels reliant entre eux des îlots de machines.

Le 15 juillet 1996 comme prévu à Montréal, a lieu le démarrage officiel du 6bone reliant l'IMAG en France pour le G6, Uni-C au Danemark et le projet WIDE au Japon. En Décembre 1996, lors de la réunion IETF de San José, le 6bone devient un groupe de travail de l'IETF. Il est décidé de structurer géographiquement le réseau, tous les tunnels d'un même pays arrivant sur un ou quelques noeuds qui échangent eux-même leurs trafics.

Pendant toute l'année 1997, le 6bone raccorde de plus en plus de sites, jouant son rôle de terrain d'expérimentation pour IPv6. Dans le domaine du routage, il utilise d'abord RIPng [RFC 2080] qui est une adaptation à IPv6 de RIP et IDRPv6, puis BGP4+ qui est une adaptation à IPv6 de BGP4.

Il permet surtout de déployer et de tester à grande échelle les deux plans d'adressage qui se succèdent en cette année 1997.

a) Format des adresses de test

Les expérimentations d'IPv6 sur un réseau devaient commencer avant que les règles d'attribution des préfixes soient complètement finalisées. La valeur 0x1FFE a été attribué par l'IANA au 6bone dans le plan d'adressage agrégé pour les expérimentations (RFC 3701). Il correspond au préfixe `3FFE::/16` pour l'ensemble du 6bone.

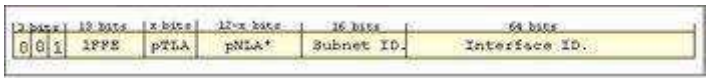


Figure 3-4 *Adresse de test du plan agrégé*

Les 48 bits restant avant le champ Subnet ID recréent les niveaux hiérarchiques d'un réseau IPv6 défini dans le RFC 3587, d'où le terme pseudo accolé au nom du champ. La taille réduite n'étant pas un facteur limitant dans la phase expérimentale. Des pseudo-TLA d'une taille initialement de 8, mais portées à 12 bits par la suite, sont attribués à des opérateurs voulant expérimenter le protocole. Les 24 ou 20 bits suivants sont utilisés pour numéroter les sites.

Les pseudo-TLA ont été alloués jusqu'en décembre 2003 aux opérateurs qui en faisaient la demande. La liste complète est disponible sur le serveur Web http://www.6bone.net/6bone_pTLA_list.html. Le tableau Pseudo TLA attribués par le groupe ngtrans reprend quelques unes de ces valeurs.

Pseudo TLA attribués par le groupe ngtrans			
Organismes/Pays	**Préfixe**	**Organismes/Pays**	**Préfixe**
ROOT66/US-CA	3FFE:0000::/24	TRUMPET/AU	3FFE:8000::/28
TELEBIT/DK	3FFE:0100::/24	ICM-PL/PL	3FFE:8010::/28
SICS/SE	3FFE:0200::/24	IIJ/JP	3FFE:8020::/28
G6/FR	3FFE:0300::/24	QTPVSIX/EU	3FFE:8030::/28
JOIN/DE	3FFE:0400::/24	APAN-KR	3FFE:8040::/28
WIDE/JP	3FFE:0500::/24	MIBH	3FFE:8050::/28
SURFNET/NL	3FFE:0600::/24	ATNET-AT	3FFE:8060::/28

L'expérimentation lié au 6bone s'est terminée symboliquement le mardi 6 juin 2006 RFC 3701. En effet, si ce réseau au début de l'introduction d'IPv6 palier à l'absence d'opérateurs officiels, il a vite montré ses limites. L'utilisation de tunnels pour créer la connectivité, a conduit à un trop fort maillage, à des routes relativement longues et par conséquence à une faible qualité de service.

E) Mise au point du plan d'adressage d'IPv6

Comme on l'a vu précédemment, la structure de l'adressage n'a pas été définie lors de l'adoption du nouveau format de paquet IP en juillet 1994. La seule chose de sûre est que les adresses IPv6 ont 128 bits de long. Une des premières tâches a donc été de définir comment ces 128 bits allaient être utilisés.

Cela va donner lieu à beaucoup d'activité et à des échanges passionnés tant les enjeux sont importants. En effet, les principaux acteurs que sont les fournisseurs d'accès au réseau, leurs clients, et les constructeurs d'équipements routage ou de postes de travail ont des intérêts et donc des points de vue qui peuvent être sensiblement opposés. Par exemple les fourniseurs d'accès sont plutôt enclins à garder captifs leurs clients en maîtrisant l'espace d'adressage, alors que les clients souhaitent garder la possibilité de changer de

fournisseurs facilement. Les constructeurs quant à eux souhaitent que les protocoles soient simples à implémenter.

Plusieurs plans d'adressages ont été étudiés pour arriver finalement au plan d'adressage dit agrégé basé sur 3 niveaux d'agrégation.

En juillet 1995, lors de la réunion IETF de Stockholm, un premier débat oppose les partisans d'un plan d'adressage géographique aux fournisseurs d'accès. Ces derniers souhaitent pouvoir faire de l'agrégation d'adresses (type CIDR) indépendamment des critères géographiques. Les premières spécifications d'IPv6 sont publiées sous forme d'une suite de RFC ([RFC 1883], [RFC 1884], [RFC 1885], [RFC 1886], [RFC 1887]) lors de la réunion IETF de Dallas en décembre 1995. Ils spécifient complètement la structure des en-têtes du paquet IPv6, le plan d'adressage est structuré par fournisseur d'accès à l'Internet.

En décembre 1996, lors de la réunion IETF de San José la proposition "8+8" qui découpe les 16 octets de l'adresse en 8 octets fournisseur d'accès et 8 octets utilisateur est discutée. Elle a pour objectif de limiter la croissance des tables de routage et de rendre l'utilisateur indépendant de son fournisseur en effectuant une traduction partielle d'adresse sur les 8 octets réservés au fournisseur d'accès en sortie de site. En mars 1997, lors de la réunion IETF de Palo Alto, la proposition 8+8 qui est devenue GSE (*Global, Site and End-system*) est à nouveau discutée, mais est rejetée.

a) Plan d'adressage géographique (geographic-based unicast address)

C'est l'un des premiers plans d'adressage proposés. Il découpe hiérarchiquement les adresses en entités géographiques (continents, pays, région, ville, etc.). Ce plan est aujourd'hui abandonné, car difficile à mettre en œuvre pour des raisons d'ordre technique notamment. Il n'est valable que dans des situations monopolistiques où les opérateurs contrôlent une partie de territoire.

Les adresses étaient caractérisées par le préfixe 8000::/3.

b) Plan d'adressage fournisseur (provider-based unicast address)

Ce type d'adresse (cf. figure 17-4) est décrit dans le RFC 2073, classé historique. Une adresse de ce type avait pour préfixe `4000::/3` et possèdait 5 niveaux hiérarchiques :

Figure 17-4 . Adresse de type plan fournisseur

Allocation des valeurs des registry ID	
registry ID	**Nature**
10000	IANA (Internet Assigned Numbers Authority)
01000	RIPE NCC (Réseaux IP Européens, Network Coordination Center)
11000	InterNIC (Internet Network Information Center)
10100	APNIC (Asia-Pacific Network Information Center)

- Le premier niveau avait une longueur fixée à 5 bits (`i=5`) et ses valeurs possibles étaient donné par le tableau précédent.
- Un fournisseur d'accès à l'Internet était identifié par un numéro qu'il obtient d'un des organismes mentionnés ci-dessus.

De même, un souscripteur obtenait son identificateur de son fournisseur. C'est un mot de 24 bits suivi de 8 bits laissés à zéro (donc `k=32`).

Les deux derniers niveaux, gérés par le souscripteur, étaient respectivement le numéro de sous-réseau (`l=16`) et l'identificateur de l'interface (48 bits), typiquement son adresse MAC. Dans ce plan d'adressage, les adresses appartiennent aux fournisseurs de connectivité IP et non pas aux utilisateurs. Tout changement de fournisseur implique donc pour un client de renuméroter toutes ses machines. De plus, si un client est abonné à un "petit" fournisseur lui-même revendant de la connectivité IP d'un plus gros fournisseur, et si le fournisseur intermédiaire décide de changer de fournisseur principal, le client final doit renuméroter tout son réseau. Ceci a été jugé inacceptable.

c) Adresses de test dans le plan d'adressage fournisseur

Ces adresses (cf. figure 17-5) sont décrites dans le RFC 1897. L'intérêt principal de ces adresses de test est de fournir un algorithme simple basé sur les adresses IPv4 existantes afin de construire des adresses IPv6 pour des équipements expérimentaux sans avoir à demander de délégation d'autorité. Ces adresses ont été largement utilisées lors de la création du 6bone:

Figure 17-5 . Adresse de type de test dans le plan fournisseur

d) Plan d'adressage GSE

Le plan d'adressage fournisseur a été largement critiqué, surtout par les opérateurs, qui tiennent essentiellement aux problèmes de renumérotation.

L'approche GSE (*Global, Site and End-system*) propose une construction dynamique des adresses avec une partie basse fixe d'identificateur globalement unique (au niveau de l'Internet) sur 64 bits et une partie haute variable identifiant le fournisseur d'accès, insérée à la volée par le routeur de sortie du site.

Tant qu'un paquet reste à l'intérieur du site, la partie haute des adresses sources des paquets émis depuis le site est forcée à zéro. Il en va de même pour les adresses destinations des paquets reçus depuis l'extérieur à destination d'une machine du site. La partie haute des paquets sortants est positionnée à la "bonne" valeur par le routeur de sortie. Si le site est connecté à plusieurs fournisseurs d'accès, ce routeur sélectionne cette valeur en fonction du fournisseur choisi. Ceci facilite le changement de fournisseur en éliminant la nécessité de renumérotation manuelle. En outre, la gestion d'un site dépendant de plusieurs fournisseurs est facilitée.

Le plan d'adressage proposé par GSE présente des avantages :

- GSE facilite la renumérotation du réseau puisqu'il suffit de changer l'adresse dans le routeur en frontière ;
- GSE supporte facilement les réseaux à attachements multiples (multihomed). Les réseaux ayant plusieurs attachements à des fournisseurs d'accès différents posent un problème d'adressage dans les plans d'adressages classiques~: si une station choisit l'une des adresses possibles, des exceptions doivent être introduites dans les tables de routage de l'Internet. Si la station prend toutes les adresses possibles, elle ne sait quelle adresse source optimale utiliser. Avec GSE l'adresse de la source est construite dynamiquement au moment de la traversée du routeur en frontière, elle est donc la meilleure pour une destination donnée.

Par contre GSE présente plusieurs inconvénients graves :

- TCP associe un contexte en fonction des adresses IP. Si celles-ci changent, la connexion doit quand même être maintenue, d'où l'existence d'un identifiant unique au niveau mondial ; cela implique une refonte de TCP qui n'est pas à l'ordre du jour à l'IETF ;
- le calcul des checksums doit être modifié puisque la station ne connaît qu'une partie de l'adresse ;
- la question de l'enregistrement des adresses dans le DNS inverse, faisant la correspondance entre une adresse et un nom de machine, n'est pas résolue ;
- la partie identificateur est supposée "globalement" unique ; on ne sait pas s'il faut une garantie absolue, et si oui, comment la vérifier ;

les mécanismes de mascarade (utiliser l'adresse source d'une autre machine) sont mal compris, et il semble que l'utilisation de sécurité active (IPsec) soit quasiment obligatoire.

F) Mise au point finale du cœur des spécifications d'IPv6

En août 1997, lors de la réunion IETF de Munich, le protocole BGP4+ est choisi de préférence à BGP5 (la possibilité d'avoir des numéros de systèmes autonomes codés sur 4 octets au lieu de 2 dans la version IPv4 n'est pas retenue).

En décembre 1997, lors de la réunion IETF de Washington, le rapport d'un test d'interopérabilité fait par un laboratoire de l'UNH (Université du New Hampshire) est présenté au groupe IPng. Ce test qui met en oeuvre 10 machines et 6 routeurs, montre que 90% des machines et 70% des routeurs interopèrent bien (en RIPng ou BGP4+ pour ces derniers). Lors de cette même réunion, les règles d'attribution des TLA et NLA et les critères définissant les niveaux d'agrégation donnent lieu à beaucoup de controverses, ces éléments ne pourront être finalisées que mi 1998.

Pendant cette année 1998, une intense activité de test et de déploiements de réseaux expérimentaux, permet d'affiner les propositions de standards, les implémentations (machines et routeurs) et de valider les plans d'adressage expérimentaux et de production.

En octobre 1998, ESnet (*Energy Sciences Network*) établit une connexion IPv6 sur ATM avec les réseaux CAIRN, Internet2/vBNS et CA*net2, puis, en décembre avec WIDE (Japon). ESnet crée aussi l'initiative 6REN (*IPv6 Research and Education Networks Initiative*) qui a pour but de donner au monde de l'enseignement et la recherche américain un service IPv6 de production en complément du 6bone qui est un réseau d'expérimentation et ne peut assurer la continuité de service necessaire à un réseau de production. Le 6REN est à considérer comme un équivalent du NANOG dédié à IPv6. C'est aussi ESnet qui assure la connecticité 6BONE/6REN. On peut aussi noter à cette époque des ébauches de projets similaires en australie (AARNET: *Australian Academic and Research Network*) et en chine (CERNET: *China Education and Research Network*). En France, le G6bone effectue sa migration dans le nouveau plan d'adressage de test.

Pendant la réunion IETF de décembre 1998 à Orlando, le groupe IPng décide de fusionner les codes IPv6 INRIA, NRL, et KAME dans la souche KAME regroupant ainsi sur une seule souche les efforts d'implémentation sous système d'exploitation BSD.

Cette fin d'année 1998 marque une nouvelle étape dans l'histoire d'IPv6, en effet, 15 RFC ont été publiés (dont 4 sont passés en Draft Standard), fixant ainsi le cœur des spécifications d'IPv6 et définissant un premier plan d'adressage aggrégé. Quatre ans après l'étape décisive de fin 1994 qui avait vu fixer les éléments consensuels d'alors (les 128 bits d'adresse, et le format d'en-tête simplifié), cette étape fixe de la même manière ce qui peut l'être, laissant à la suite du processus le soin de résoudre les nombreux problèmes encore en suspens (migration, mobilité, DNS, autoconfiguration, etc...).

G) Les schémas de migration et d'intégration IPv4/IPv6

Bien que la standardisation d'IPv6 ne soit pas achevée, loin s'en faut, les membres de l'IETF, considèrent qu'il faut maintenant s'attaquer à un nouveau problème de taille : celui de la transition d'IPv4 vers IPv6. Lors de la réunion intérimaire de l'IETF de février 1999 (qui a lieu à l'IMAG à Grenoble) les propositions de schémas de migration et d'intégration IPv4/IPv6 sont si nombreux qu'il est décidé d'en faire une taxonomie pour y voir plus clair et faire naître des convergences.

Au premier trimestre 2000, la plupart des propositions sont formalisées par des RFC. On peut les répartir dans les trois grandes familles suivantes :

Des outils au niveau réseau comme :

- les adresses IPv4 compatibles : utilisées au début quand il n'y avait que très peu de machines IPv6,
- les adresses IPv4 mappées : qui permettent à des machines IPv4 de communiquer avec des machines IPv6,
- les adresses '6to4' : qui permettent de joindre un site IPv6 à travers un réseau IPv4 sans configuration de tunnels,
- les tunnels configurés : utilisés pour le déploiement des réseaux de test, mais comme tous les tunnels, ils réduisent le MTU et empilent deux routages,
- les générateurs automatiques de tunnels (tunnels brokers) : solution permettant à une machine d'obtenir une connectivité IPv6 sans avoir intervention manuelle du gestionnaire du site offrant cette connectivité.

Des protocoles de traduction d'adresses comme :

- SIIT [RFC 2765] : qui permet à des machines IPv6 de communiquer avec des machines IPv4 au moyen de traducteurs d'adresses, situés en bordure des domaines IPv4/IPv6. Ces traducteurs traitent les paquets IP et ICMP, sans introduire d'état dans le réseau,
- NAT-PT [RFC 2766] : semblable à SIIT, il utilise des adresses globales IPv6 (et non pas des adresses dérivées d'IPv4). Les traducteurs de bordure modifient les adresses et les en-tête, ils doivent disposer d'un préfixe IPv4,
- DSTM [BTM-id] : qui permet de résoudre tous les cas de communication entre les mondes IPv6 et IPv4 sans nécessiter de portage des applications IPv4. Les traducteurs de bordure doivent disposer d'un préfixe IPv4.

Des protocoles intervenant au niveau TCP ou des applicatifs comme :

- BIS [RFC 2767] : qui permet à des aplications IPv4 de fonctionner sur les machines IPv6 en effectuant les traductions d'adresses nécessaires dans le noyau du système d'exploitation,
- SOCKS [RFC 3089] : fonctionnellement similaire à BIS, c'est une extension du protocole SOCKS (défini dans le RFC 1928) à la communication entre des piles TCP/IPv4 et TCP/IPv6.

Durant le reste de l'année 2000 et toute l'année 2001, ces protocoles sont retravaillés, notamment pour prendre mieux en compte les aspects sécurité et dénis de service ainsi que les mécanismes de mise à jour du DNS qui peuvent ou doivent leur être associés.

H) La longue marche vers un Internet IPv6

Le 14 juillet 1999, presque 3 ans jour pour jour après le démarrage du 6bone, l'IANA annonce que la première délégation officielle de préfixes IPv6 vient d'être effectuée auprès du RIPE-NCC, de l'ARIN, et de l'APNIC. On peut considérer que cette date marque la naissance du nouvel Internet IPv6. Ces organismes vont alors commencer à distribuer des sTLA à un rythme tel qu'indiqué dans le tableau suivant.

Allocation des sTLA IPv6 par les RIR			
APNIC	**ARIN**	**RIPE-NCC**	
mai 2000	13 sTLA	3 sTLA	14 sTLA
décembre 2000	21 sTLA	10 sTLA	23 sTLA
décembre 2001	48 sTLA	20 sTLA	51 sTLA

Comme les chiffres le montrent, ce déploiement, s'il est bien réel, reste limité en volume et assez hétérogène géographiquement. L'hétérogénéité géographique s'explique par le fait que les utilisateurs nord américains ne sont pas en situation de pénurie d'adresses IPv4, et donc ne sollicitent que faiblement leurs opérateurs en connectivité et adresses IPv6. Le déploiement limité sur les autres continents s'explique par la non complétude de la standardisation des protocoles IPv6. C'est le deuxième axe de travail de l'IETF (en plus de celui sur les mécanismes de transition/intégration) durant les années 2000 et 2001, car cette complétude constitue un pré-requis à un développement à grande échelle d'IPv6.

En décembre 1999, le RFC 2740 standardise la version IPv6 d'OSPF, et en juillet 2000, lors de la réunion IETF de Pittsburgh, une communication dans le groupe de travail OSPF informe les participants que le logiciel de routage Zebra dispose d'une première version IPv6 d'OSPF implémentée et testée avec succès.

En juillet et août 2000, les RFC 2874 et RFC 2894 sont publiés, ils concernent des extensions au DNS pour l'aggrégation et la renumérotation ainsi que la renumérotation de routeurs. En fin d'année 2000, les RFC 3007 et RFC 3008 concernant la sécurisation des DNS sont publiés. Ce sont les compléments indispensables pour gérer la dynamicité introduite par les mécanismes de renumérotation.

Début 2001, le RFC 3041 améliore la protection de la vie privée ou professionnelle qui pouvait être mis en cause par le mécanisme d'autoconfiguration. En effet, ce dernier, en construisant un identifiant d'interface unique et permanent à partir de l'adresse Ethernet aurait pu permettre la tracabilité dans le temps ou dans l'espace du possesseur d'une machine IPv6. L'introduction d'une durée de vie limitée pour les identifiants d'interface et leur remplacement régulier par d'autres règle cette question.

Mi 2001, le groupe de travail IPng constate qu'il a presque terminé sa tâche. Il décide de changer de nom ; un nouveau groupe est créé, IPv6, qui a pour but de terminer les documents en attente et de traiter de points encore en débat (renumérotation automatique, adressage de site, ...). Un autre groupe, multi6, est créé pour s'occuper du problème de la multi-domiciliation.

En août 2001, le RFC 3152 met en place la délégation de la racine (`ip6.arpa`) de l'arbre de nommage inverse pour IPv6, complétant ainsi le dispositif pour le DNS.

Fin 2001, les travaux de standardisation sont principalement centrés sur les mécanismes de transition comme DSTM et BIA (*Bump In the API*) ainsi que sur la gestion du multicast IPv6. Des liens sont aussi établis avec la téléphonie mobile dont la version UMTS a choisi IPv6 comme technologie de transport pour pallier les insuffisances d'IPv4 en matière d'espace d'adressage.

Début 2002, l'ensemble des systèmes d'exploitation sont équipés d'une double pile IPv4 et IPv6, la plupart des routeurs implémentent au moins un protocole de routage interne (RIPng) et un protocole de routage externe (BGP4+). Même si la suite des protocoles IPv6 n'est pas encore aussi complète qu'il serait souhaitable, l'Internet IPv6 est bien en cours de déploiement, et des noeuds importants comme le Startap, aux Etats-Unis, le Nspixp au Japon ou le Sfinx en France échangent désormais du trafic natif IPv6.

XVIII) Bases whois

1) finition et concepts de base

Les bases de type WHOIS contiennent des informations techniques et administratives permettant entre autres de savoir qui est le titulaire d'un nom de domaine, qui gère tel bloc d'adresses IP (v4 ou v6), quelle est la politique de routage de tel AS... Il s'agit de bases distribuées à accès public dont l'interrogation est possible en utilisant des outils disponibles en standard sur une grande partie des systèmes Unix (commande `whois` tout simplement). Si ce n'est pas le cas, il existe un certain nombre de clients dont l'un des plus usités est celui qu'il est possible de télécharger sur le site du RIPE.

Il existe de plus, sur les sites des organismes assurant la gestion de bases WHOIS, des outils de consultation via une interface Web. De telles interfaces peuvent être trouvées sur les sites du RIPE, de l'INTERNIC, du 6BONE, ou bien encore de l'AFNIC. En ce qui concerne la distribution de l'information, tout ce qui concerne l'adressage IP et les informations de routage se trouve réparti sur 3 bases. La base ARIN pour l'Amérique du nord et du sud, les Caraïbes et l'Afrique sub-saharienne; la base RIPE pour l'Europe et une partie de l'Afrique; enfin la base APNIC pour la zone Asie Pacifique. Pour les adresses IPv6 de test (préfixe `3FFE::/16`) il existe une base spécifique gérée par le 6BONE.

2) Types de données spécifiques à IPv6

Il existe 2 types de données qui ont été spécialement créés pour répondre aux besoins spécifiques à IPv6. Il s'agit des types inet6num et ipv6-site.

Les objets de type inet6num sont présents dans les bases RIPE, ARIN, APNIC et 6BONE, ceux de type ipv6-site ne sont utilisés que dans la base 6BONE car ils sont pour l'instant réservés aux adresses de test. D'ailleurs, ces deux types sont utilisés à titre expérimental et leurs format peut être amené à être modifié selon les besoins futurs.

3) Type inet6num

Ce type de données va permettre de définir les propriétés associées à un préfixe IPv6 donné. Le format qui est décrit ci-après ne concerne pas la base ARIN qui adopte un format minimaliste à l'image de la base INTERNIC pour les noms de domaines.

A) Format générique (template) d'un objet de type inet6num

Chaque ligne du template donne des informations sur un attribut. Après le nom de l'attribut, on indique sont statut (obligatoire, optionnel ou généré), s'il ne doit être présent qu'une seule fois ou non dans un objet et pour finir si cet attribut est une clef de recherche dans la base.

```
inet6num: [mandatory] [single] [primary/look-up key]
netname: [mandatory] [single] [lookup key]
descr: [mandatory] [multiple] [ ]
country: [mandatory] [multiple] [ ]
admin-c: [mandatory] [multiple] [inverse key]
tech-c: [mandatory] [multiple] [inverse key]
rev-srv: [optional] [multiple] [inverse key]
status: [generated] [single] [ ]
remarks: [optional] [multiple] [ ]
notify: [optional] [multiple] [inverse key]
mnt-by: [mandatory] [multiple] [inverse key]
mnt-lower: [optional] [multiple] [inverse key]
changed: [mandatory] [multiple] [ ]
source: [mandatory] [single] [ ]
```

- inet6num: Préfixe IPv6 dont on définit les attributs (format préfixe/taille du préfixe).
- netname: Le nom associé à la plage d'adresses définie par le préfixe.
- descr: Description succincte de l'objet.
- country: Code en 2 lettres (ISO 3166) identifiant le pays du contact administratif de l'organisme auquel appartient le préfixe.
- admin-c: Référence à un objet de type person/role identifiant un contact administratif pour l'objet inet6num. Cette référence est un NIC-Handle, c'est-à-dire un objet unique dans la base.
- tech-c: Référence à un objet de type person/role identifiant un contact technique pour l'objet inet6num. Cette référence est un NIC-Handle, c'est à dire un objet unique dans la base.
- rev-srv: Serveur de noms pour les reverses de la plage d'adresses définie par le préfixe.
- status: Type du préfixe (TLA, Sub-TLA, NLA, ...).
- remarks: Remarques divers et variées.
- notify: E-Mail de la personne qui est prévenue lors de toute modification effectuer sur l'objet.
- mnt-by: Référence à un objet de type mntner identifiant les personnes habilitées à modifier l'objet, ainsi que la méthode d'authentification.
- mnt-lower: Référence à un objet de type mntner identifiant les personnes habilitées à fournir des espaces d'adresses (avec ce préfixe) a des utilisateurs finaux.
- changed: E-Mail de la personne ayant effectuer la dernière mise à jour de l'objet, ainsi que la date de cette modification.
- source: Identification de la base où se trouve cet objet (6BONE, RIPE, APNIC, ARIN, ...)

B) Exemple d'objet de type inet6num

Si on utilise la commande suivante :

```
> whois -h whois.ripe.net 2001:0660::/35
```

On obtient le résultat suivant :

```
inet6num: 2001:0660::/35
netname: FR-RENATER-20000321
descr: Renater Sub-TLA block
descr: Réseau National de télécommunications pour la
descr: Technologie l'Enseignement et la Recherche
descr: National telecommunications network
descr: for Technology, Education and Research
country: FR
admin-c: BT261-RIPE
admin-c: GR1378-RIPE
tech-c: GR1378-RIPE
status: SUBTLA
mnt-by: RIPE-NCC-HM-MNT
mnt-lower: RENATER-MNT
changed: hostmaster@ripe.net 20000321
changed: hostmaster@ripe.net 20000322
source: RIPE
```

On peut donc en déduire que le préfixe `2001:0660::/35` appartient au réseau FR-RENATER-20000321, qu'il s'agit d'un sub-TLA français géré par Renater. On a aussi les NIC-handles des différents contacts techniques et administratifs.

4) Type ipv6-site

Ce type d'objet est uniquement utilisé, pour le moment en tout cas, dans la base 6BONE, c'est à dire sur les adresses de test IPv6 (de type `3FFE::/16`). Il sert a décrire les sites utilisant des adresses en IPv6.

A) Template d'un objet de type ipv6-site

Le type ipv6-site est décrit dans un document dont la validité a expiré le draft draft-ietf-ngtrans-6bone-registry-03.txt.

```
ipv6-site: [mandatory] [single]
origin: [mandatory] [single]
descr: [mandatory] [single]
location: [optional] [multiple]
country: [mandatory] [multiple]
prefix: [mandatory] [multiple]
application: [optional] [multiple]
tunnel: [optional] [multiple]
native: [optional] [multiple]
```

```
contact: [mandatory] [multiple]
remarks: [optional] [multiple]
url: [optional] [multiple]
notify: [optional] [multiple]
mnt-by: [optional] [multiple]
changed: [mandatory] [multiple]
source: [mandatory] [single]
```

- ipv6-site: Nom du site.
- origin: Numéro de l'AS (Autonomous System) dont fait partie le site.
- descr: Description succinte de l'objet.
- location: "Coordonnées terrestre" du site, pour plus de détails, se référer au RFC 1876.
- country: Code en 2 lettres (ISO 3166) identifiant le pays du contact administratif de l'organisme auquel appartient le site.
- prefix: Liste des préfixes IPv6 utilisés sur ce site.
- application: Liste des applications disponibles sur le site. En plus du nom de l'application, on indique l'équipement sur lequel elle est accessible. Par exemple "http://www.ipv6.toto.fr".
- tunnel: Les tunnels permettent d'encapsuler un protocole dans un autre. En l'occurrence, il s'agit principalement de tunnels IPv6 au-dessus d'une infrastructure en IPv4. On indique le protocole du tunnel, celui de l'infrastructure, le nom de la machine sur le site qui fait une des extrémités du tunnel, le nom de la machine distante qui fait l'autre bout du tunnel, le site où cette machine se trouve (optionnel), et pour finir le protocole de routage utilisé : STATIC, BGP4+... (optionnel). Par exemple:

```
IPv6 in IPv4 popeye.site1.fr -> olive.site2.fr OLIVE-SITE BGP4+
```

- native: Similaire a l'attribut "tunnel", à ceci près que cette fois-ci on décrit les connexion natives IPv6. La syntaxe est la même.
- contact: Référence à un objet de type "person" identifiant un contact pour le site.
- remarks: Remarques diverses et variées sur le fonctionnement général du site.
- url: Pointeurs sur quelques pages d'informations supplémentaires à propos du site.
- notify: E-Mail de la personne qui est prévenue lors de toute modification effectuée sur l'objet.
- mnt-by: Référence à un objet de type mntner identifiant les personnes habilitées à modifier l'objet, ainsi que la méthode d'authentification.
- changed: E-Mail de la personne ayant effectuée la dernière mise à jour de l'objet, ainsi que la date de cette modification.
- source: Identification de la base où se trouve cet objet (6BONE, RIPE, APNIC, ARIN, ...)

B) Exemple d'objet de type ipv6-site

Si on utilise la commande suivante :

```
> whois -h whois.6bone.net G6-UTBM
```

On obtient le résultat suivant :

```
ipv6-site: G6-UTBM
origin: AS1717
descr: Université de Technologie de Belfort-Montbéliard
country: FR
prefix: 3FFE:303:22::/48
```

```
tunnel: IPv6 in IPv4 testnm.utbm.fr ->
ipv6-cisco.ipv6.u-strasbg.fr G6-PIR-GE STTIC
contact: TN1-6BONE
remarks: This object is automatically converted from the RIPE181 registry
changed: noel@dpt-info.u-strasbg.fr 20000315
changed: auto-dbm@whois.6bone.net 20010117
source: 6BONE
```

5) Création, modification et suppression d'objets

Les différentes opérations possibles sur les objets contenus dans les bases de type WHOIS se passent toutes de la même manière, il suffit d'envoyer un mail a une adresse spécifique :

- auto-dbm@whois.6bone.net pour la base du 6BONE,
- auto-dbm@ripe.net pour la base RIPE.

Certains sites proposent des interfaces plus pratiques, comme sur le site de Viagenie65 qui permettent de réaliser tous les types d'opérations beaucoup plus facilement en évitant les habituelles erreurs de syntaxes qui ne manquent pas d'arriver en utilisant la technique des mails.

A) Création

Avant toute création, il est nécessaire d'avoir au moins un objet de type "person" ou "role" dans la base où l'on souhaite travailler afin d'obtenir un identifiant unique, plus communément appelé NIC-handle. Un objet de type "person" va être créé pour définir une personne physique.

L'objet de type role, lui, définit plus une fonction qu'un individu et son utilisation principale est de décrire par exemple une équipe technique d'un organisme. Lorsqu'un des membres de cette équipe s'en va ou qu'un nouveau arrive il suffit de modifier le contenu de l'objet de type role, il n'y a pas besoin de mettre à jour les autres objets qui référencent ce NIC-handle. Donc, la technique la plus approprié pour débuter tout enregistrement dans une base est de créer les objets "person" nécessaires afin d'obtenir des NIC-Handles, puis de créer le ou les objets "role" qui référencent les objets de types "person" précédemment créés et c'est ce dernier NIC-Handle obtenu que l'on choisira d'utiliser en priorité pour les contacts techniques ou administratifs. Si on choisit d'utiliser la technique de création par mail, il suffit d'envoyer un template correspondant à l'objet a créer correctement rempli. Pour avoir le format exact et la syntaxe de chaque attribut, il faut utiliser la commande suivante :

```
> whois -h whois.6bone.net -v ipv6-site
```

La commande précédente permet d'obtenir une documentation complète de l'objet de type ipv6-site tel qu'il est défini dans la base du 6BONE. Selon le même principe on peut demander la documentation concernant un objet de type person dans la base RIPE :

```
> whois -h whois.ripe.net -v person
```

Si jamais l'option "-v" n'est pas prise en compte par le client whois de votre système, il faut télécharger une version prenant quelques options avancées66.

B) Modification

Pour la modification par mail, il faut récupérer l'objet tel qu'il est stocké dans la base en utilisant tout simplement la commande whois, modifier le contenu et envoyer le mail ainsi compose.

C) Suppression

Pour la suppression, il faut récupérer l'objet dans la base et le renvoyer en ajoutant un attribut supplémentaire "delete" dans le template. Cela donnera quelque chose comme :

```
...
remarks: Ceci est un objet de test
delete: Un texte quelconque
changed: test@test.tt 20011111
source: G6BONE
```

Toutefois, si l'objet est protégé, il ne pourra être supprimé que si l'on connaît la méthode d'authentification.

6) Problèmes spécifiques à WHOIS

Le principal problème des bases WHOIS est qu'il existe deux grandes tendances au niveau du format des données, le style INTERNIC, minimaliste et à la lecture peu aisée et le style RIPE, plus complet et plus facile à comprendre.

Le second problème, qui en découle, est que les informations ne sont plus centralisées comme cela a été le cas à une époque. Si cela n'est pas trop dramatique en ce qui concerne les adresses IP puisqu'il n'y a que 3 bases (ARIN, APNIC et RIPE), cela est beaucoup plus gênant pour les noms de domaines puisque pratiquement chaque organisme d'enregistrement possède sa propre base. Et bien entendu, chacun utilise des templates plus ou moins différents dérivés des style RIPE et INTERNIC.

Le troisième problème, qui résulte des deux précédents, est qu'une même information stockée dans deux bases différentes n'aura pas du tout le même format et donc offrira plus ou moins de détails. Pour finir, il faut savoir que ces bases ne sont là qu'à titre d'information et leur contenu n'influence pas le fonctionnement de l'Internet. Il arrive donc que l'information que l'on obtient soit incomplète ou erronée.

Conclusion, si les bases de type WHOIS sont des alliés précieux pour obtenir des informations rapidement et facilement, une analyse technique un peu plus poussée est souvent nécessaire pour en vérifier la véracité et la complétude.

XIX) Bibliographie

Contents

1) Livres sur IPv6

[BM95] S. O. Bradner & A. Mankin ed : IPng, Internet Protocol Next Generation, Addison-Wesley (IPng Series), ISBN0201633957, Septembre 1995.

[Gai98] S. Gai, Internetworking IPv6 With Cisco Routers (Computer Communications), McGraw-Hill, ISBN: 0-070-22836-1, Février 1998.

[Hui97] C. Huitema: IPv6: The New Internet Protocol, Prentice Hall, ISBN: 0-138-50505-5, Octbre 1997.

[LL99] Peter Loshin & Pete Loshin: IPv6 Clearly Explained (Clearly Explained), Ap Professional, ISBN: 0-124-55838-0, Janvier 1999.

[MM00] Mark A. Miller & P. E. Miller: Implementing IPv6, second edition (Network Troubleshooting Library), IDG Books Worldwide, ISBN: 0-764-54589-2, Janvier 2000.

[Mil97] Stewart S. Miller: IPv6 : The Next Generation Internet Protocol, DigitalPress, ISBN: 1-555-58188-9, Décembre 1997.

[MK98] Marcus Goncalves & Kitty Niles: IPv6 Networks, McGraw-Hill, ISBN: 0-070-24807-9, Mai 1998.

[Sal00] Peter H. Salus: Big Book of IPV6 Addressing RFCs, Morgan Kaufmann Publishers, ISBN: 0-126-16770-2, Mars 2000.

[WR99] J. D. Wegner & Robert Rockwell: IP Addressing and Subnetting, Including IPv6, Syngress Media, ISBN: 1-928-99401-6, Décembre 1999.

2) Internet Drafts sur IPv6

Remarque : Il faut rappeler que les Internet drafts sont des documents de travail à durée de vie limitée. Ils ont vocation à devenir des RFC. Le lecteur est donc invité à vérifier quelle est la version la plus récente, ou si un RFC le remplace, en consultant l'index à jour (cf. Les RFC (Request For Comments)).

[BCP2-id]

J. Bound, M. Carney, C. Perkins: Extensions for the Dynamic Host Configuration Protocol for IPv6, draft-ietf-dhc-v6exts-12.txt, Internet Draft, 5 Mai 2000 - Obsolete.

[BKLSPTSDY-id]

W. Biemolt, M. Kaat, T. Larder, H. Steenman, R. van der Pol, G. Tsirtsis, Y. Sekiya, A. Durand, K. Yamamoto: On overview of the introduction of IPv6 in the Internet, draft-ietf-ngtrans-introduction-to-ipv6-transition-08.txt, Internet Draft, Février 2002. Working Group Concluded.

[BTM-id]

J. Bound, L. Toutain, O. Medina, F. Dupont, A. Durand, H Afifi,: Dual Stack Transition Mechanism (DSTM), draft-ietf-ngtrans-dstm-07.txt, Internet Draft, Aout 2002. Obsolete.

[BP-id]

M. Blanchet, F. Parent, IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP), draft-blanchet-v6ops-tunnelbroker-tsp-03.txt, Aout 2005. Work in progress.

[CMB-id]

Hesham Soliman, Claude Castelluccia, Karim Malki, Ludovic Bellier, Hierarchical Mobile IPv6 mobility management (HMIPv6), draft-ietf-mipshop-hmipv6-04.txt, Décembre 04. Obsolete - Experimental RFC 4140.

[Craw-id]

Matt Crawford: IPv6 Node Information Queries, [http://www.ietf.org/internet-drafts/draft-ietf-ipngwg-icmp-name-lookups-15.txt draft-ietf-ipngwg-icmp-name-lookups-15.txt, Internet Draft, 13 Février 2006. Work in progress.

[Drave-id]

R. Draves: Default Address Selection for IPv6, draft-ietf-ipngwg-default-addr-select-06.txt, Internet Draft, 28 Septembre 2001. Obsolete - RFC 3484.

[DHZ-id]

S. Deering, B. Haberman, B. Zill, T. Jinmei, E. Nordmark, A. Onoe: IP Version 6 Scoped Address Architecture, draft-ietf-ipngwg-scoping-arch-03.txt Internet Draft, 30 Novembre 2001. Obsolete - RFC 4007.

[DIS-id]

Alain Durand, Johan Ihren, Pekka Savola, Operational Considerations and Issues with IPv6 DNS, draft-ietf-dnsop-ipv6-dns-issues-12.txt, Internet Draft, 19 Octobre 2005,. Work in progress.

[Dupont-id]

F. Dupont, M. Laurent-Maknavicius: AAA for mobile IPv6, draft-dupont-mipv6-AAA-01.txt, Internet Draft, 20 Novembre 2001, Obsolete.

[Ernst-id]

Thierry Ernst, Network Mobility Support Requirements, draft-ietf-nemo-requirements-05.txt, Octobre 2005

[Fenner-id]

Bill Fenner, Haixiang He, Brian Haberman, Hal Sandick, IGMP/MLD-based Multicast Forwarding ('IGMP/MLD Proxying'), draft-ietf-magma-igmp-proxy-06.txt

[Fenner2-id]

Bill Fenner, Mark Handley, Hugh Holbrook, Isidor Kouvelas Protocol Independent Multicast - Sparse Mode (PIM -SM): Protocol Specification (Revised), IETF Internet Draft, draft-ietf-pim-sm-v2-new-11.txt, 4 Octobre 2004. Work in progress.

[Haber-id]

B. Haberman: Dynamic Allocation Guidelines for IPv6 Multicast Addresses, draft-ietf-malloc-ipv6-guide-01.txt, Internet Draft, 12 Octobre 2000. Obsolete - RFC 3307.

[HD-id]

R. Hinden, S. Deering: IP Version 6 Addressing Architecture, [draft-ietf-ipv6-addr-arch-v4-04.txt], Internet Draft, 20 Mai 2005. Work in progress.

[HH-id]

Robert Hinden, Brian Haberman, Centrally Assigned Unique Local IPv6 Unicast Addresses, draft-ietf-ipv6-ula-central-01.txt, Internet Draft, 21 Février 2005, Obsolete - See RFC 4193.

[Hopps-id]

C. E. Hopps: Routing IPv6 with IS-IS, draft-ietf-isis-ipv6-06.txt, Internet Draft, Octobre 2005. Work in progress.

[Huitéma-id]

C. Huitema, Teredo: Tunneling IPv6 over UDP through NATs, draft-huitema-v6ops-teredo-05.txt, Avril 2005, Obsolete - See RFC 4380.

[JP-id]

D. B. Johnson, C. Perkins: Mobility Support in IPv6, draft-ietf-mobileip-ipv6-15.txt, Internet Draft, 2 Juillet 2001. Obsolete - RFC 3775.

[NGF-id]

Tri Nguyen, Gerard Gastaud, Francois Le Faucheur, Dirk Ooms, Jeremy De Clercq, Stuart Prevost, Connecting IPv6 Domains across IPv4 Clouds with BGP, draft-ooms-v6ops-bgp-tunnel-06.txt, Janvier 2006. Proposed standard.

[NPE-id]

Chan Wah Ng, Eun Kyoung Paik, Thierry Ernst, Analysis of Multihoming in Network Mobility Support, draft-ietf-nemo-multihoming-issues-04.txt, 24 Octobre 2005. Work in progress

[Prz-id]

Tony Przygienda, M-ISIS: Multi Topology (MT) Routing in IS-IS, draft-ietf-isis-wg-multi-topology-11.txt, 21 Octobre 2005.

[Prigent-id]

N. Prigent, J. Marchand, F. Dupont, B. Cousin, M. Laurent-Maknavicius, J. Bournelle: DHCPv6 Threats, draft-prigent-dhcpv6-threats-00.txt, Internet Draft, 24 mai 2001, Expired.

[RFC2547bis]

Eric C. Rosen, Yakov Rekhter, BGP/MPLS IP VPNs, draft-ietf-l3vpn-rfc2547bis-03.txt, Internet Draft, October 2004, Obsolete - RFC 4364.

[Templin-id]

Fred Templin, T. Gleeson, M. Talwar D. Thaler, Intra-Site Automatic Tunnel Addressing Protocol (ISATAP), draft-ietf-ngtrans-isatap-24.txt, Juillet 2005, Obsolete - RFC 4214.

[Thubert-id]

Pascal Thubert, NEMO Home Network models, draft-ietf-nemo-home-network-models-06.txt, 17 Fevrier 2006.

3) Autres documents de travail

[FIPS-46]

Data Encryption Standard, Federal Information Processing Standards Publication 46, U.S. National Institute of Standards and Technology, U.S. Department Of Commerce, 15 Janvier 1977.

[FIPS-81]

DES Modes of Operation, Federal Information Processing Standards Publication 81, U.S. National Institute of Standards and Technology, U.S. Department Of Commerce, 2 Décembre 1980.

[FIPS-180-1]

Secure Hash Standard, Federal Information Processing Standards Publication 180-1, U.S. National Institute of Standards and Technology, U.S. Department Of Commerce, Avril 1995.

4) Autres Références

[AL00]

Mohammed Achemlal, Maryline Laurent, Analyse des fonctions des protocoles IPsec et leur intégration dans un réseau privé virtuel, Annales des télécommunications, 2000.

[AL06]

P. Albitz and C. Liu: DNS and BIND, 5th Edition, ISBN: 978-0596100575, O'Reilly Media, Inc., May 1, 2006.

[BTC02]

T. Bu, L. Gao, D. Towsley, On Characterizing Routing Table Growth, GlobalInternet 2002. http://www-unix.ecs.umass.edu/~lgao/globalinternet2002_tian.pdf

[Ernst01f-fr]

Ernst, Thierry, Le Support des Réseaux Mobiles dans IPv6, UniversitéJoseph Fourier, Octobre 2001, Grenoble, France, http://www.inria.fr/rrrt/tu-0714.html

[Ernst03f]

Thierry Ernst, Le Support des Réseaux Mobiles dans IPv6, CFIP: Colloque Francophone sur l'Ingenierie des Protocoles, Octobre 2003, Paris

[Fen-id]

Bill Fenner, Management Information Base for the User Datagram Protocol (UDP), draft-ietf-ipv6-rfc2013-update-04.txt, Internet Draft, 20 Octobre 2004, Work in progress.

[Hui95]

C. Huitema: Le routage dans l'Internet, Eyrolles, Octobre 1995.

[IEEE]

Protocol Independant Interfaces, IEEE Std 1003.1g, DRAFT~6.6, Mars 1997.

[JH-id]

Jeffrey Haas, Susan Hares, Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4), draft-ietf-idr-bgp4-mib-05.txt, Internet Draft, 31 Août 2004, Work in progress.

[JM-id]

Dan Joyal, Vishwas Manral, Management Information Base for OSPFv3, draft-ietf-ospf-ospfv3-mib-09.txt, Internet Draft, 13 Mai 2005, Work in progress.

[Kau-id]

Charlie Kaufman, Internet Key Exchange (IKEv2) Protocol, draft-ietf-ipsec-ikev2-17.txt, Internet Draft, 4 Octobre 2004, Work in progress.

[LAU03]

M. Laurent-Maknavicius, Le protocole IPsec, TE7545, Techniques de l'Ingénieur, Sécurité des systèmes d'information, Novembre 2003.

[LAU04-A]

M. Laurent-Maknavicius, M. Gardie, LDAP et la certification, Rapport de recherche du GET, 04001 LOR, 2004.

[LAU04-B]

M. Laurent-Maknavicius, La sécurité de l'accès distant, Technique et Science Informatiques (TSI), 2004.

[MHF-id]

Ambarish Malpani, Russ Housley, Trevor Freeman, Simple Certificate Validation Protocol (SCVP), draft-ietf-pkix-scvp-22.txt, Internet Draft, Octobre 2005, Work in progress.

[Mos-id]

Robert Moskowitz, Host Identity Protocol, draft-ietf-hip-base-04.txt, Internet Draft, 24 Octobre 2005, Work in progress

[Pui02-A]

J.J. Puig, M. Laurent-Maknavicius, Analyse critique d'IPsec, rapport de recherche 03 004 LOR, 2002.

[Pui02-B]

J.J. Puig, M. Laurent-Maknavicius, Analyse de l'impact de la mise en oeuvre d'IPsec dans les architectures de Communication, rapport de recherche 03 002 LOR, octobre 2002.

[Pui03]

J.J. Puig, M. Laurent-Maknavicius, Evaluation en environnement réel de la mise en oeuvre des Services de sécurité dans les architectures typiques (Aspects ARP), rapport de recherche 03 001 LOR, janvier 2003.

[Pui04-A]

J.J. Puig, M. Laurent-Maknavicius, Etude des interactions IPsec/DNS, rapport de recherche 04002 LOR, 2004.

[Pui04-B]

J.J. Puig, M. Laurent-Maknavicius, Evaluation en environnement réel de la mise en oeuvre des Services de sécurité dans les architectures typiques (Aspects liés à ICMP), rapport de recherche 04004 LOR, 2004.

[RFC2401bis]

Stephen Kent, Karen Seo, Security Architecture for the Internet Protocol, draft-ietf-ipsec-rfc2401bis-06.txt, Internet Draft, 1 Avril 2005, Work in progress

[RFC2402bis]

Stephen Kent, IP Authentication Header, draft-ietf-ipsec-rfc2402bis-11.txt, Internet Draft, 21 Mars 2005, Work in progress.

[Rou-id]

Shawn Routhier, Management Information Base for the Internet Protocol (IP), draft-ietf-ipv6-rfc2011-update-10.txt; Internet Draft, 24 Mai 2004, Work in progress.

[Sch95]

B. Schneier: Applied Cryptography : protocols, algorithms, and source code in C, (second edition), John Wiley & Sons, ISBN: 0-471-12845-7, 1996.

[Sta99]

William Stallings: Snmp, Snmpv2, Snmpv3, and Rmon 1 and 2, Addison Wesley, ISBN: 0-201-48534-6, Janvier 1999.

[Tout03]

Laurent Toutain: Réseaux Locaux et Internet: des protocoles à l'interconnexion, Troisième édition revue et augmentée, Hermès, ISBN: 2-7462-0670-6, 2003.

[Uni31]

ATM Forum: ATM User-Network Interface (UNI) Specification Version 3.1, Prentice Hall, Englewood Cliffs, NJ, Juin 1995.

[WH-id]

Margaret Wasserman, Brian Haberman, IP Forwarding Table MIB, draft-ietf-ipv6-rfc2096-update-07.txt, Internet Draft, 12 février 2004, Work in progress.

[WK99]

M. Welsh et L. Kaufman: Le système Linux, 2e édition révisée, Éditions O'Reilly, ISBN: 2-84177-033-8, Janvier 1999.

5) Sites Web sur IPv6

[IETF]

http://www.imag.fr/pub/archive/IETF: Mirroir français du serveur de l'IETF.

[6bone]

http://www.6bone.net: Réseau 6bone.

[G6]

http://www.g6.asso.fr/ Groupe et association G6.

[hs247]

http://hs247.com/ Informations sur le 6bone et IPv6 en général.

[ipv6.org]

http://www.ipv6.org/ pages d'information sur le protocole IPv6

[ipv6forum]

http://www.ipv6forum.org/ Groupement d'industriels pour promouvoir IPv6.

[playground]

http://playground.sun.com/pub/ipng/html/ipng-main.html liste des mises en oeuvre d'IPv6 dans les équipements.