Migration IPv6

Migration IPv4/IPv6

GRILLE DE GESTION

Type de diffusion

☐ Livrable	☐ Consultable	☒ Privé
☐ Diffusion contrôlée		Exemplaire N°

Mode d'accès

Serveur DIS : \Stages\IPv6.ML\Documents\Dossiers\Migration.doc

Conservation

Responsable : CJ Lieu : DIS

E				
D				
C				
B				
A				
Ind.	Nom Visa	Nom Visa	Nom Visa	Date
	REDACTION	VERIFICATION	APPROBATION	

GRILLE DE REVISION

La présente Grille de révision indique l'objet et la localisation des modifications génératrices du changement d'indice.

N°	Objet	Localisation
1	Document initial	-
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

SOMMAIRE

1 Introduction

1.1 Objet du projet

Le projet porte sur l'étude de migration des réseaux IP de la version v4 à la version v6. Cette évolution (à prévoir dans les années à venir) va poser un certain nombre de problèmes, mais va également permettre de proposer de nouvelles fonctionnalités (mobilité, routage, sécurité, …). Le but de l'étude est de proposer des méthodes pour effectuer ce basculement, ainsi qu'une étude détaillée des nouvelles fonctionnalités.

La finalité du projet est de définir un certain nombre d'offres de service à proposer à nos clients pour qu'ils puissent basculer sans problème vers IPv6.

L'étude est réalisée sous la forme d'un Projet de Fin d'Etude (PFE) entre RSI et l'INSA de Lyon. Ce PFE est effectué en entreprise à raison de 2 jours par semaine pendant 6 mois (Novembre – Avril) et à temps plein pendant 2 mois (Mai – Juin).

1.2 Objet du document

Présentation et étude des différentes solutions pour effectuer la migration. Dans le cas de réseaux complexes, on étudiera la possibilité de migration échelonnée dans le temps. Migration des logiciels.

2 Techniques de cohabitation v4/v6 existantes

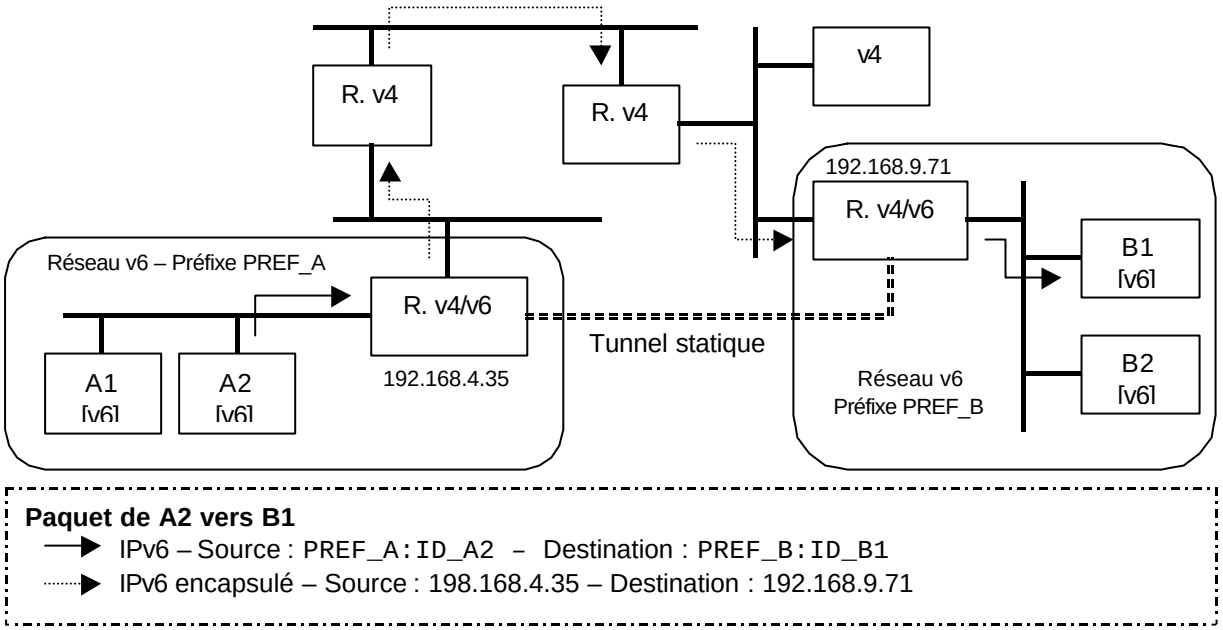
Le passage d'un Internet entièrement IPv4 à un Internet entièrement IPv6 est prévu pour durer très longtemps (plusieurs années). Il est donc nécessaire pendant cette période de transition de permettre aux machines IPv4 et IPv6 de cohabiter et de communiquer entre elles. Un certain nombre de mécanismes ont donc été étudiés pour réaliser cette cohabitation et ainsi faciliter la transition.

2.1 Encapsulation

Les techniques d'encapsulation sont utilisées dans le cas où l'on doit faire communiquer des machines IPv6 qui ne sont reliées que par un réseau IPv4. Les paquets IPv6 sont alors encapsulés dans des paquets IPv4 le temps de traverser ces points. Cette encapsulation peut permettre de réaliser des tunnels IPv6 entre deux points.

2.1.1 Tunnel statique [MECH]

Les tunnels statiques sont utilisés pour relier un réseau ou une machine IPv6 à un autre réseau IPv6 par l'intermédiaire d'un réseau IPv4. Ils sont configurés à la main et sont mis en place avec un durée de vie importante. Les machines qui sont aux extrémités du tunnel doivent avoir une double pile IPv4/IPv6 et disposer chacune d'une adresse IPv4 globale. Les autres machines du réseau IPv6 n'ont donc pas besoin de cette double pile pour communiquer avec les machines IPv6 situées de l'autre côté du tunnel, mais elle peut être utile pour communiquer avec des machines IPv4 (sans passer par le tunnel). Un tunnel statique est dans la plupart des cas utilisé pour se relier au backbone IPv6 (lui même constitué de tunnels dans les phases initiales de transition).



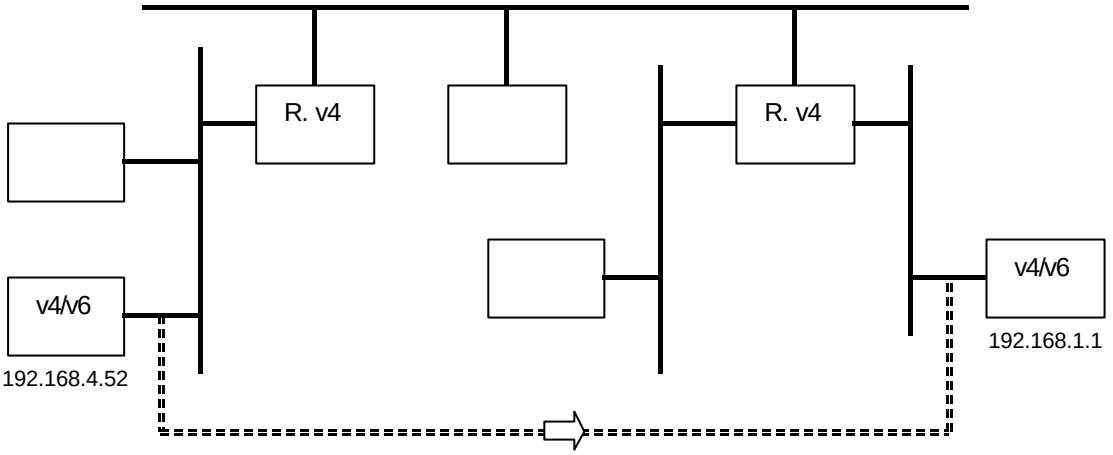
Note : L'adresse IP du point de sortie du tunnel peut éventuellement être une adresse IPv4 anycast[1] (!!). Cela permet par exemple d'avoir plusieurs routeurs d'accès au *6bone*, d'utiliser celui qui est le plus près et de basculer automatiquement sur un autre en cas de défaillance.

2.1.2 Tunnel automatique (avec une adresse IPv4-compatible) [MECH]

Les tunnels automatiques servent à communiquer en IPv6 avec une machine connecté sur un réseau IPv4. Cette méthode est souvent utilisé pour joindre une machine IPv6 'isolée'. Les deux machines établissant le tunnels doivent disposer d'une double pile IPv4/IPv6. La machine destination du tunnel doit être la machine destinataire du paquet, alors que la machine source du tunnel peut être la machine source du paquet ou un routeur qui a reçu le paquet sur son réseau IPv6. Dans ce dernier cas, il faudra que la machine source possède une adresse IPv4-compatible (pour le retour). Pour cela, elle pourra éventuellement l'obtenir par allocation dynamique dans une plage d'adresse IPv4.

Note : Les adresses IPv4-compatible sont des adresses IPv6 particulières qui sont formés en ajoutant les 32 bits d'une adresse IPv4 au préfixe `::/96`. Par exemple, `::192.168.1.1` est l'adresse IPv4-compatible de 192.168.1.1.

[1] Une adresse anycast est partagée par plusieurs machines mais seule la machine la plus proche reçoit les paquets.



Tunnel Automatique IPv6
`::192.168.4.52` → `::192.168.1.1`

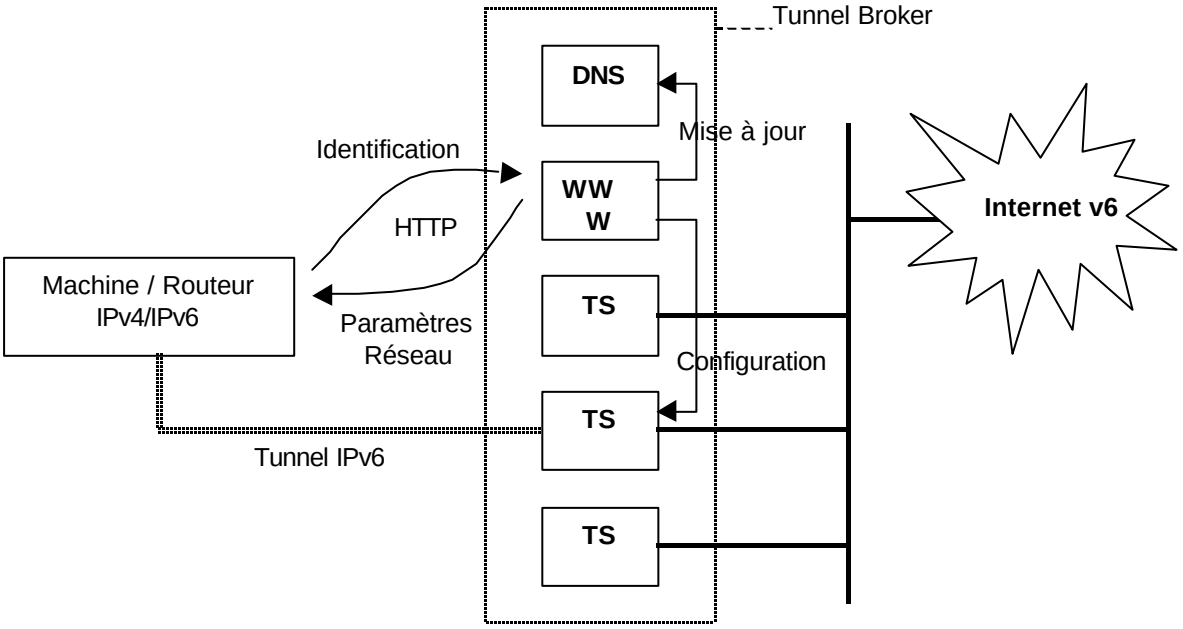
2.1.3 Tunnel Broker [BROKER]

Un *Tunnel Broker* (courtier en tunnels) est un service proposé sur Internet permettant la création et la configuration automatique d'un tunnel statique pour accéder à Internet v6 par l'intermédiaire d'un réseau IPv4. Cela peut être vu comme un fournisseur d'accès à IPv6.

L'accès à un *Tunnel Broker* est réalisé de la façon suivante :

- L'utilisateur s'enregistre sur le site Web en indiquant les informations suivantes :
  - Identification (nom, login/passwd, …).
  - Nombre de machines à connecter (une machine ou un site de x machines).
  - Système d'exploitation et adresse IPv4 de la machine qui met en place le tunnel (il faut disposer d'une adresse IPv4 globale).
  - Etc…
- Si la création du tunnel est autorisée (il peut éventuellement falloir avoir un compte, payer un abonnement, limitation du nombre de machines, …) :
  - Renvoi des paramètres réseau à utiliser (dont l'adresse IPv4 du *Tunnel Server* et le préfixe IPv6 à utiliser).
  - Configuration automatique du *Tunnel Server* en lui indiquant l'adresse IPv4 de la machine qui met en place le tunnel, la ou les adresses IPv6, la durée de vie du tunnel, …
  - Mise à jour du DNS pour que la ou les machines de l'utilisateur puissent être joignables par un nom long.
  - Configuration de la machine qui met en place le tunnel (soit automatiquement, soit manuellement par l'utilisateur).

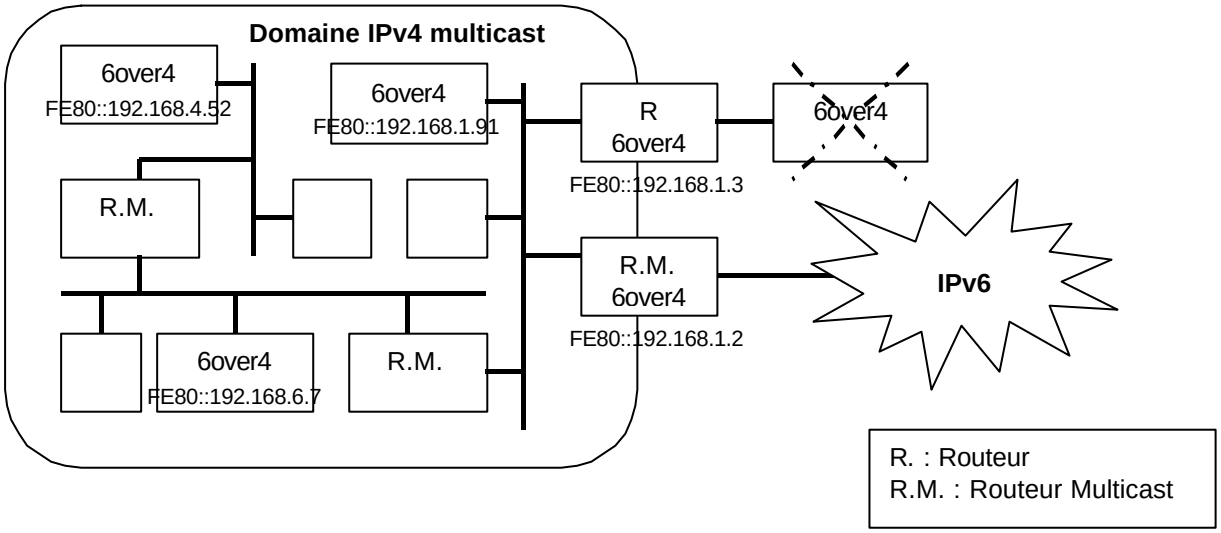
FreeNet6 (http://www.freenet6.net) propose déjà un service de *Tunnel Broker* gratuit permettant de se relier au 6bone (réseau expérimental de test IPv6).



2.1.4 6over4 [RFC2529]

Le mécanisme *6over4* permet à plusieurs machines IPv6 isolés (mais connectées par un réseau IPv4 supportant le multicast), de créer un réseau local IPv6 (comme si elles étaient situés sur le même lien) en s'appuyant sur un domaine multicast IPv4. De plus, si l'une de ces machines est un routeur IPv6 connecté à un autre réseau IPv6, toutes les machines peuvent avoir accès à ce réseau.

L'identifiant d'interface utilisé pour déterminer l'adresse d'une machine est l'adresse IP. L'adresse lien-local d'une machine utilisant le protocole *6over4* est donc `FE80::192.168.1.1` où 192.168.1.1 est son adresse IPv4. On peut également utiliser un préfixe IPv6 propre au site s'il y a un routeur IPv6 pour faire l'annonce du préfixe. Le domaine IPv4 doit obligatoirement pouvoir transmettre du trafic multicast car le multicast est utilisé pour transmettre les paquets multicast IPv6, lui même nécessaire pour le fonctionnement du lien local.



2.1.5 6to4 [6TO4]

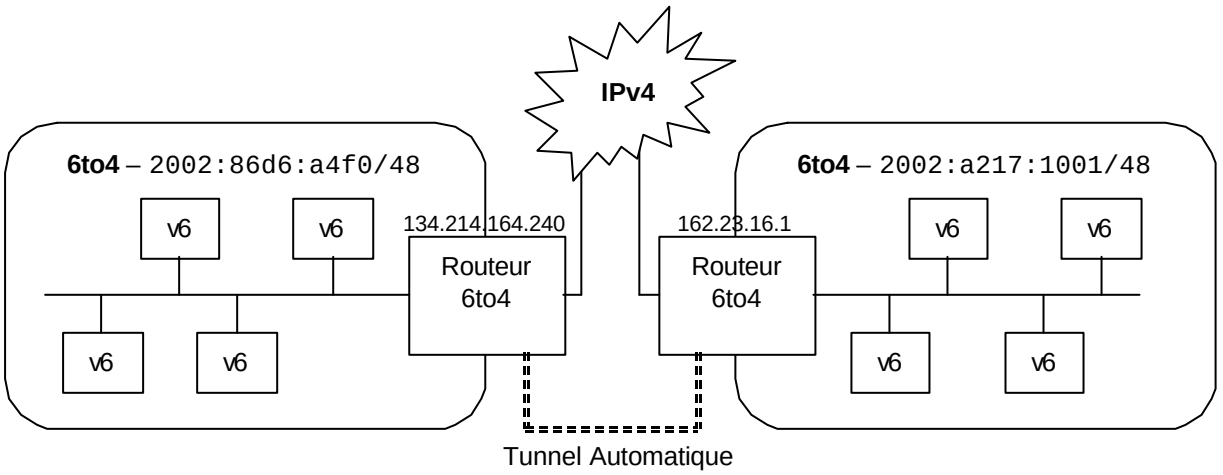
Le mécanisme *6to4* peut être utilisé pour interconnecter entre eux des sites *6to4* par l'intermédiaire d'un réseau IPv4. Les routeurs *6to4* ont besoin d'une double pile IP et d'une adresse IPv4 unique pour pouvoir relier un site entier.

Pour cela, chaque routeur se crée un préfixe IPv6 unique (sans avoir besoin de l'allouer auprès d'une quelconque instance) et l'utilise sur tous son site. Le préfixe est construit en ajoutant l'adresse IP du routeur au préfixe `2002::/16`. On obtient ainsi un préfixe de 48 bits auquel on peut ajouter un identifiant de réseau (16 bits) et l'identifiant de l'interface. Chaque machine dispose donc d'une adresse IPv6 unique sans avoir à en faire la demande. Ces adresses peuvent même (et c'est conseillé) être renvoyés par le serveur de DNS pour pouvoir être joint en IPv6 par les autres sites 6to4.

Quand une machine veut envoyer un paquet à une machine située sur un domaine *6to4*, elle effectue la démarche suivante :

- Requête DNS avec le nom long
- Le serveur de DNS renvoie l'adresse `2002::c001::0203::0001::ID_Interface`
- La machine source envoie le paquet sur son réseau IPv6
- Le paquet va arriver à un routeur *6to4* (qui peut router le préfixe `2002::/16`)
- Celui-ci va encapsuler le paquet IPv6 dans un paquet IPv4 et l'envoyer à l'adresse 192.1.2.3 (`c001::0203`)
- La machine 192.1.2.3 (qui est un routeur 6to4) desencapsule le paquet IPv6 et l'envoie sur son réseau IPv6

De cette manière, le simple fait de posséder une adresse IPv4 globale permet de construire un préfixe IPv6 permettant à toutes les machines d'un site d'être jointes par cette méthode. La seule contrainte est que les deux sites doivent utiliser le mécanisme *6to4*.



2.2 Double Pile [MECH]

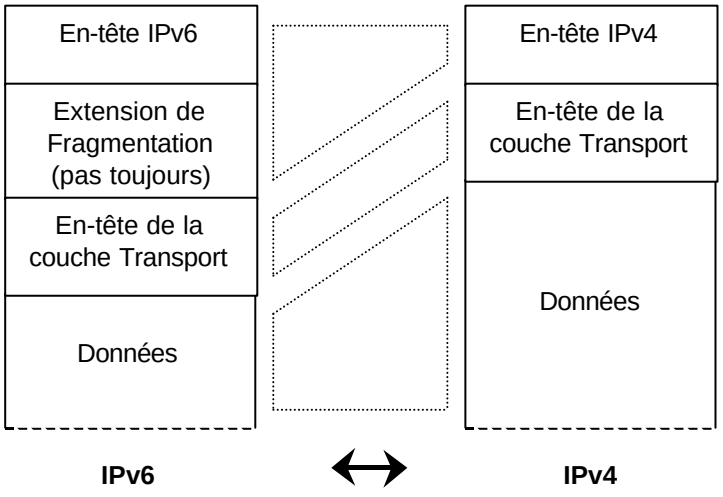
Dans le cas où l'on dispose d'une pile IPv4/IPv6, d'applications IPv4/IPv6 et que l'on est sur un réseau IPv4/IPv6, on n'a pas besoin de mécanismes supplémentaires pour accéder à la fois à des machines IPv4 et à des machines IPv6. Dans ce cas, les communications sont transmises par les couches IP correspondantes aux adresses utilisées et il n'y a aucun problème de conversion.

2.3 Traduction / Conversion

Pour faire communiquer des machines IP4 avec des machines IPv6, il est nécessaire d'implémenter des mécanismes de traduction ou de conversion de paquets. Comme il y a de grandes différences entre IPv4 et IPv6, ces mécanismes ne peuvent pas marcher dans toutes les circonstances. Il se peut donc que certains protocoles et certaines options (mobilité, qualité de service, …) ne marchent pas (ou de façon dégradé) avec des mécanismes de traduction.

2.3.1 Conversion des en-têtes [SIIT]

L'algorithme SIIT (*Stateless IP/ICMP Translation Algorithm*) a été défini pour permettre la conversion des en-têtes IPv4 en en-têtes IPv6 et inversement. Il est utilisé dans un certain nombre de mécanismes de communication entre des machines IPv4 et IPv6.



SIIT prend en charge les différents points nécessaire pour convertir un paquet IPv4 en IPv6 :

- Correspondance des champs de l'en-tête IP
- Correspondance entre les codes ICMP
- Gestion de la fragmentation (pas géré de la même façon dans les deux cas)
- Gestion du Path MTU (obligatoire pour IPv6)
- Etc…

SIIT permet de convertir à la fois les paquets TCP, UDP, et ICMP. Néanmoins, il peut y avoir des problèmes dans le cas où les données contenues dans les paquets contiennent des informations sur les couches inférieures (transport et réseau). En effet, la conversion des en-têtes risque de changer ces informations et la machine distante se retrouvera avec des informations contradictoires. Comme protocole connu et qui pose ce genre de problème, on peut citer FTP.

Voici les points à noter lorsque l'on veut faire de la traduction :

- Pas de possibilité de faire passer du trafic Multicast (`::ffff:224.1.2.3` n'est pas une adresse IPv6 Multicast alors que 224.1.2.3 est une adresse Multicast IPv4).
- Les paquets IGMP (Gestion des Groupes Multicast) sont détruits.
- IPSec:AH (Authentification) ne peut pas marcher à travers un traducteur (les adresses IP changent).
- IPSec:ESP (Confidentialité) peut marcher à travers un traducteur.
- Les paquets qui utilisent l'option de routage ne doivent pas être convertit (détruits). Renvoi d'une erreur.
- On détruit les paquets ICMP avec des options qui n'existent pas dans les deux versions.
- On ignore les options de IPv4 et de IPv6.
- On garde le champs DiffServ qui est utilisé à la fois dans IPv4 et IPv6.
- Il n'y a plus d'identifiant de flux (n'existe pas dans IPv4).

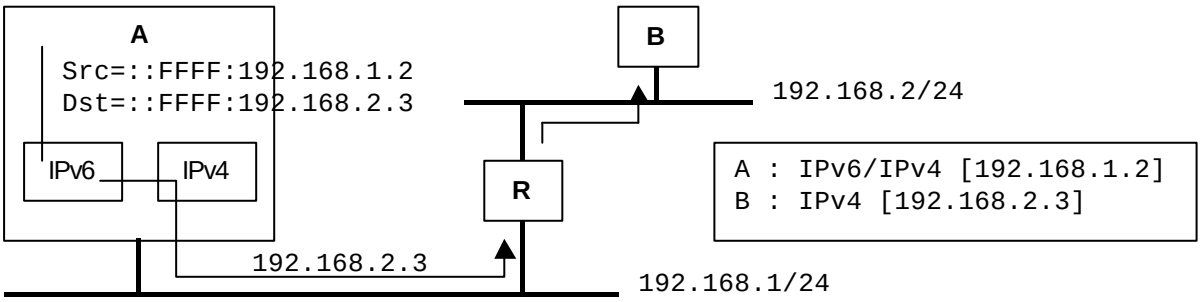
2.3.2 Utilisation d'adresses IPv4 mappées [SIIT]

Les adresses IPv4 mappées sont des adresses IPv6 utilisées pour indiquer que la machine destination est une machine IPv4 et qu'il faudra donc traduire les paquets (cf. § 2.3.1).

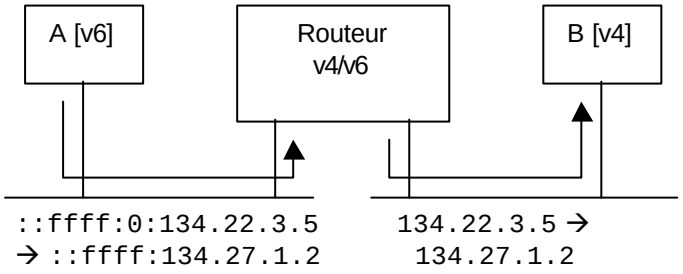
Note : Une adresse IPv4 mappées est construite en ajoutant les 32 bits de l'adresse IPv4 au préfixe `::FFFF/96`. Par exemple, `::FFFF:192.168.1.2`.

La traduction des paquets peut être réalisée soit en local, soit sur une machine distante :

- Si la traduction est faite en local, elle doit disposer d'une adresse IPv4 globale pour pouvoir recevoir les réponses. Les paquets sont envoyés vers la couche IPv6 avec des adresses IPv4 mappées comme source et comme destination. Ils sont convertis en IPv4 en utilisant les adresses extraites des adresses IPv4 mappées. Dans le sens inverse, on convertit les paquets IPv4 en IPv6 en utilisant également des adresses IPv4 mappées.



- Si la traduction est faite sur une machine distante, la machine source n'a pas besoin d'avoir une pile IPv4. Pour pouvoir envoyer des paquets, on a néanmoins besoin d'une adresse source IPv4. Celle-ci sera allouée dynamiquement dans une plage d'adresses disponible (le moyen de l'allouer n'est pas encore clairement défini) et sera utilisée pour construire une adresse IPv4-translated[2]. Le paquet sera ensuite envoyé avec cette adresse IPv4-translated comme source et l'adresse IPv4 mappée comme destination. La machine traductrice convertit ensuite le paquet et utilise les adresses IPv4 contenues dans les adresses IPv6 comme source et destination. Dans le sens inverse, le paquet IPv4 est convertit en IPv6 avec les mêmes adresses et envoyé vers la machine IPv6. La méthode utilisée pour router le paquet vers la machine IPv6 détenant l'adresse IPv4-translated n'est pas clair, mais il serait éventuellement possible de le faire en utilisant l'extension de routage.



Routeur v4/v6

Pool d'adresse v4 : 134.22.3/24
Adresses allouées :
- 134.22.3.5 → PREF:ID_A
- …

[2] Une adresse IPv4-translated est créé en ajoutant une adresse IPv4 au préfixe `::ffff:0:0:0/96`. Par exemple, `::ffff:0:192.168.1.2`. Elle correspond à une machine IPv6 dialoguant avec une machine IPv4.

2.3.3 Network Address Translation – Protocol Translation [NAT-PT]

NAT-PT est utilisé pour faire de la translation d'adresses entre un réseau IPv6 et un réseau IPv4. Le but est de permettre à un site qui est passé en IPv6-only de continuer à pouvoir communiquer avec des machines IPv4. Pour cela, le routeur NAT-PT dispose d'une plage d'adresses IPv4 qu'il utilise, lorsqu'il envoie des paquets, pour faire des associations avec des adresses IPv6.

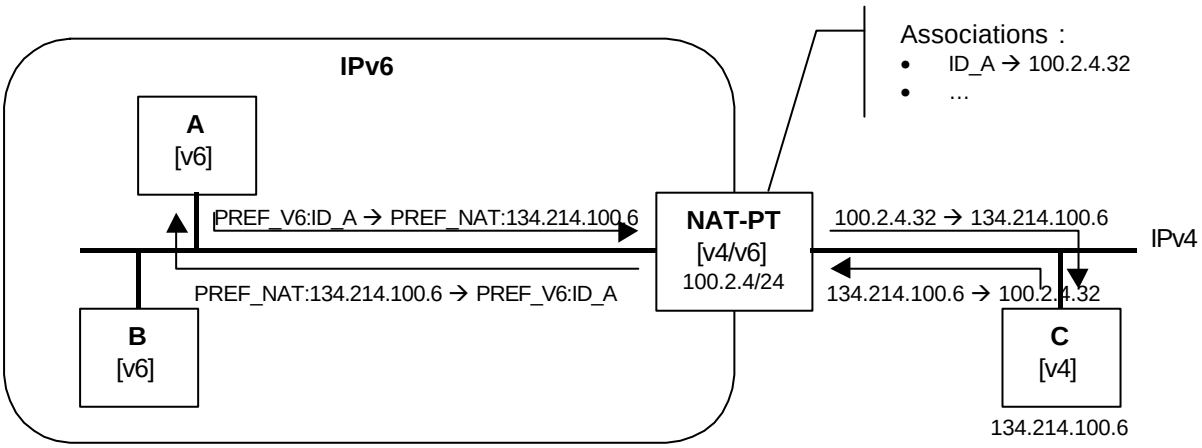
Pour chacune des 3 variantes de NAT-PT, on utilise la méthode de conversion des en-têtes SIIT (cf. § 2.3.1) pour faire passer les paquets d'un protocole à l'autre.

2.3.3.1 Variantes

- **NAT-PT Simple**

Dans ce cas, le routeur NAT-PT dispose d'une plage d'adresse IPv4 qu'il associe au fur et à mesure des connexions. Il convertit ensuite le paquet IPv6 en un paquet IPv4 et l'envoie sur le réseau IPv4.

L'adresse destination IPv6 utilisée sur le réseau IPv6 est constituée d'un préfixe défini par le NAT-PT auquel on ajoute l'adresse IPv4 de la machine destination. Pour construire cette adresse, il faut que la machine source connaisse ce préfixe ou bien qu'elle utilise des noms longs (cf. DNS-ALG - § 2.3.3.2).



- **NAPT-PT (Network Address Port Translation – Protocol Translation)**

NAPT-PT améliore le concept en permettant à plusieurs machines de partager la même adresse IPv4. En effet, en plus de faire la translation de l'adresse IP, on ajoute la translation du port (TCP/UDP). On peut également utiliser une plage d'adresse IPv4 mais ce n'est pas forcement nécessaire (on dispose déjà de 64k connections TCP et 64k connections UDP par adresse IPv4).

Lorsqu'un paquet arrive sur le routeur NAT-PT, une association est faite entre l'adresse IPv6 plus le port source et l'adresse IPv4 associée plus un port source. L'adresse destination IPv6 est formée de la même manière que le NAT-PT Simple.

- **NAT-PT Bidirectionnel**

Le NAT-PT Bidirectionnel permet à une machine située derrière un NAT-PT d'être jointe à partir de l'extérieur (par une machine IPv4). Le routeur NAT-PT associe une adresse IPv4 à la machine destination lorsqu'une machine source veut communiquer (détecté par une requête DNS). Cf. DNS-ALG - § 2.3.3.2 pour plus de détail sur le traitement du DNS. Dans ce cas là, on a besoin d'une adresse IPv4 par machine IPv6 qui est accédée de l'extérieur et au même moment.

2.3.3.2 ALG – Application Layer Gateway (Passerelle Applicative)

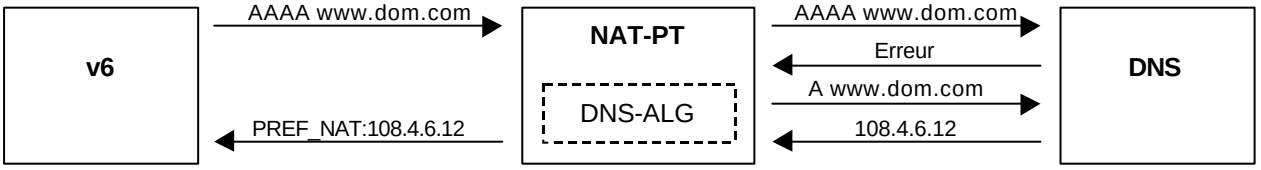
Pour que le NAT-PT marche au mieux, il est nécessaire de modifier certains paquets (dans les couches supérieures à 3). Pour cela, le routeur NAT-PT dispose de plusieurs passerelles applicatives lui permettant de modifier certains protocoles pour qu'ils s'insèrent mieux dans un système NAT-PT. Il n'y a donc pas de problèmes pour les protocoles connus (comme FTP) qui sont implémentés, par contre il est possible que des protocoles moins répandus posent problème.

- **DNS-ALG**

DNS-ALG permet d'intercepter les requêtes DNS pour permettre :

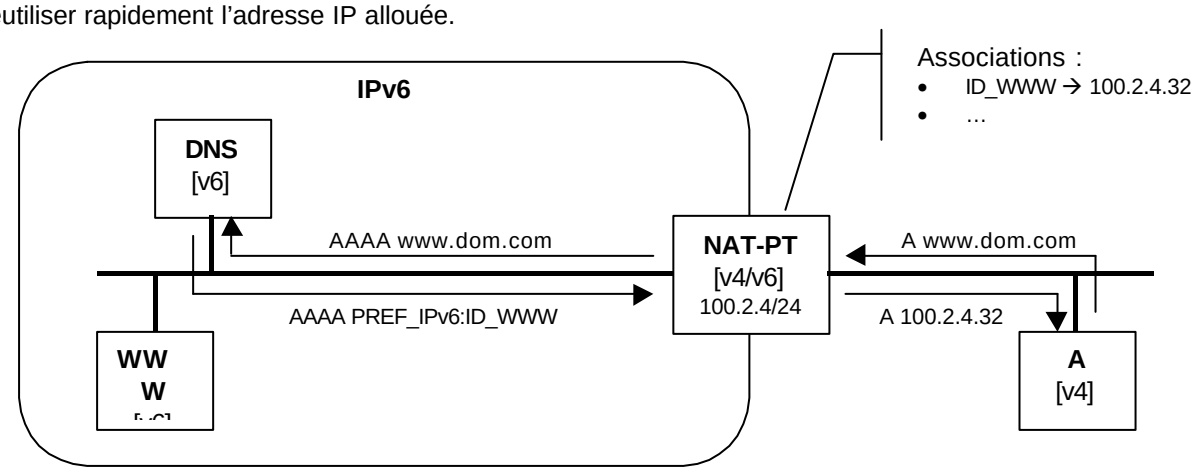
- Le renvoi d'une adresse IPv6 lorsque la machine destination est IPv4 pour pouvoir y accéder en traversant le NAT-PT.

Lorsqu'une requête DNS IPv6 échoue car la machine destination n'a pas d'adresse IPv6, le routeur NAT-PT réenvoie la requête pour récupérer l'adresse IPv4 de la machine correspondante. Il construit ensuite une adresse IPv6 constituée de son préfixe NAT-PT et de l'adresse IPv4. Puis il renvoie cette adresse comme étant la réponse à la requête DNS initiale. De cette façon, les communications avec des machines IPv4 ou IPv6 sont totalement transparentes.



- L'allocation d'une adresse IPv4 permettant à une machine IPv4 extérieure de communiquer avec une machine IPv6 située derrière un NAT-PT.

Lorsqu'une machine veut communiquer avec une autre machine située derrière un NAT-PT, elle est obligée d'utiliser son nom long. Elle commence donc par faire une requête DNS pour déterminer l'adresse IPv4 à utiliser. Cette requête est intercepté par le NAT-PT qui transforme l'adresse IPv6 en une adresse IPv4 associée (qu'il alloue dans un pool d'adresses). La durée de vie de la réponse DNS devra être faible pour permettre de réutiliser rapidement l'adresse IP allouée.



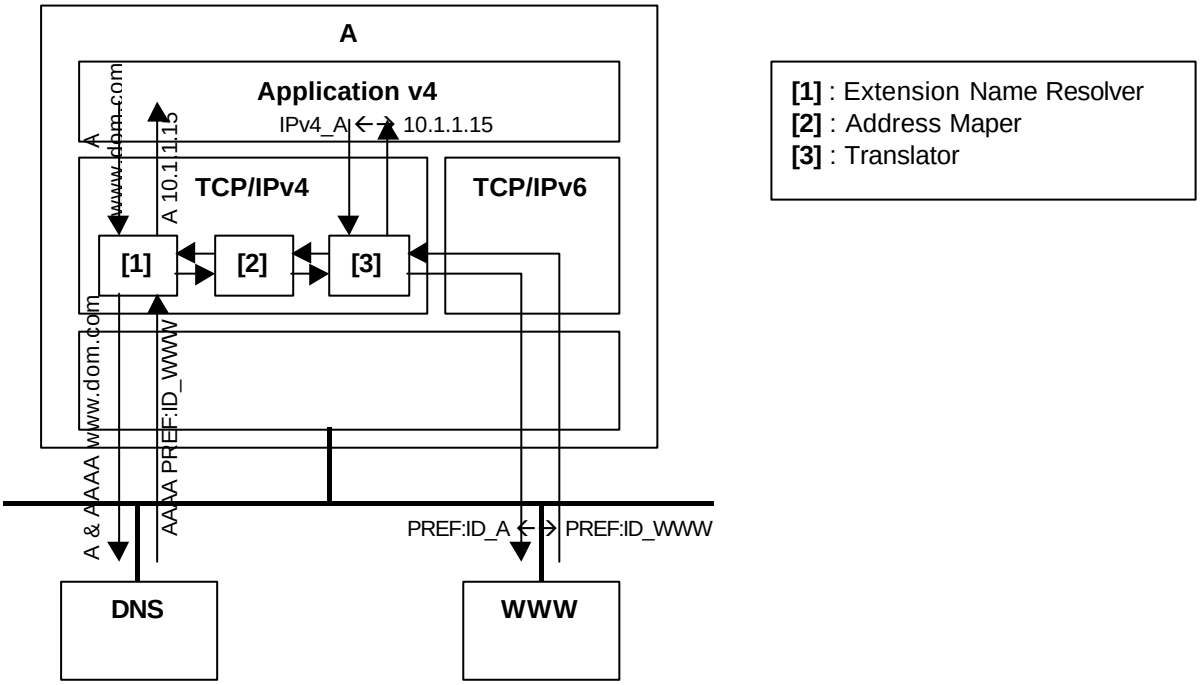
- **FTP-ALG**

Le protocole FTP envoie au niveau de la couche applicative des renseignements de la couche inférieure : des adresses IP et des ports. Comme ces paramètres sont modifiés pendant la translation d'adresse, il est nécessaire de les traiter et de les modifier à la volée pour permettre d'utiliser FTP derrière un NAT-PT.

2.3.4 Bump-In-The-Stack [BITS]

Le mécanisme de *Bump-In-The-Stack* peut être utilisé par une applications IPv4 (sur une machine IPv4/IPv6) pour communiquer avec une machine IPv6. Cela peut s'avérer utile pendant la phase de transition pour continuer à utiliser des applications IPv4 qui n'ont pas encore pu être mises à jour. Le principe de BITS est d'allouer dynamiquement des adresses IPv4 privées associées à des adresses IPv6.

Pour cela, trois modules sont ajoutés à la couche IPv4 :
- *Extension Name Resolver*
- *Address Maper*
- *Translator*



2.3.4.1 Requête DNS

Lorsque l'application IPv4 effectue une requête DNS, celle-ci passe par la couche IPv4 qui la transmet au module *Extension Name Resolver*. Ce dernier modifie la requête pour demander à la fois une adresse IPv4 (A) et une adresse IPv6 (AAAA) au serveur de DNS.
- Si le serveur renvoie une adresse IPv4, celle-ci est renvoyée à l'application et la communication s'effectue normalement en IPv4.
- Si le serveur renvoie une adresse IPv6, l'*Extension Name Resolver* fait une demande d'adresse IPv4 au module *Address Mapper*. C'est l'*Adress Mapper* qui gère les adresses allouées par BITS. Celui-ci renvoie une adresse IPv4 associée à l'adresse IPv6 de la machine à joindre (elle est allouée si elle n'existe pas déjà).

L'application peut ensuite communiquer avec cette adresse IP 'virtuelle', comme si elle dialoguait avec une machine IPv4.

2.3.4.2 Envoi d'un paquet

Lorsque l'application envoie un paquet à cette adresse IPv4 'virtuelle', elle est transmise au module *Translator*. Celui-ci récupère l'adresse IPv6 correspondante auprès de l'*Adress Mapper* et effectue la traduction du paquet IPv4 en un paquet IPv6 (cf. § 2.3.1) avant de l'envoyer à la couche IPv6.

2.3.4.3 Réception d'un paquet

Lorsqu'un paquet arrive au niveau de la couche IPv6 vers une application IPv4, il est transmis au module *Translator*. Celui-ci récupère l'adresse IPv4 'virtuelle' correspondante auprès de l'*Adress Mapper* (elle est allouée si elle n'existait pas) et effectue la traduction du paquet IPv6 en un paquet IPv4 (cf. § 2.3.1) avant de l'envoyer à l'application.

2.3.4.4 Notes

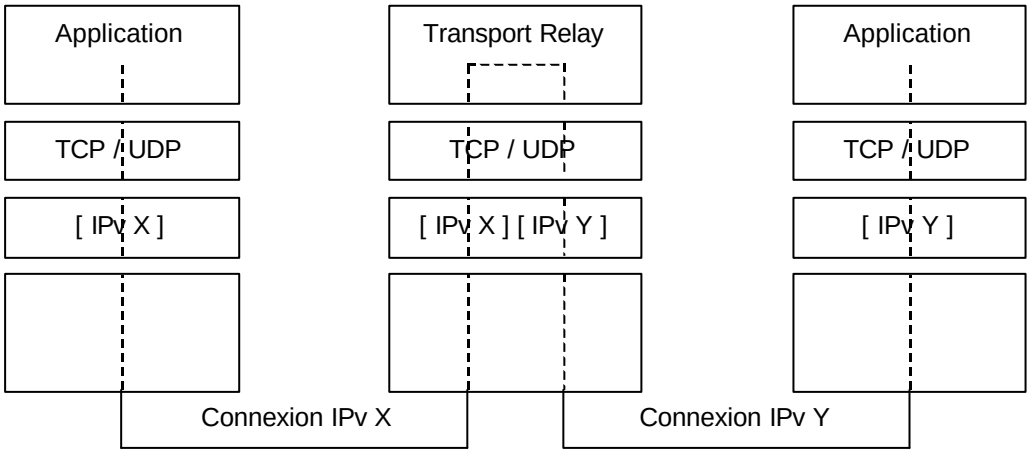
- Pour les remarques sur la traduction des paquets, cf. § 2.3.1.
- L'Adress Mapper utilise une plage d'adresses IP privées pour faire les associations. Ces adresses ne sont utilisées qu'en interne, on peut donc utiliser la même plage sur un ensemble de machines.
- Le mécanisme de BITS ne fonctionne pas avec les communications Multicast.

2.4 Passerelles

Les passerelles sont utilisés pour permettre la communication entre des machines IPv4 et des machines IPv6. Par contre, au lieu de convertir les paquets comme les traducteurs, elles font passer les données d'une connexion à l'autre.

2.4.1 Transport Relay [TRANS]

La machine qui joue le rôle de Transport Relay est utilisée pour relayer une session [TCP, UDP]/IPv6 vers une session [TCP, UDP]/IPv4 et inversement. Le Transport Relay est une machine disposant d'une double pile IPv4/IPv6. Il établit une connexion avec la source en IPv6 et une autre avec la destination en IPv4. Il transmet ensuite les données de la couche transport d'une connexion à l'autre.



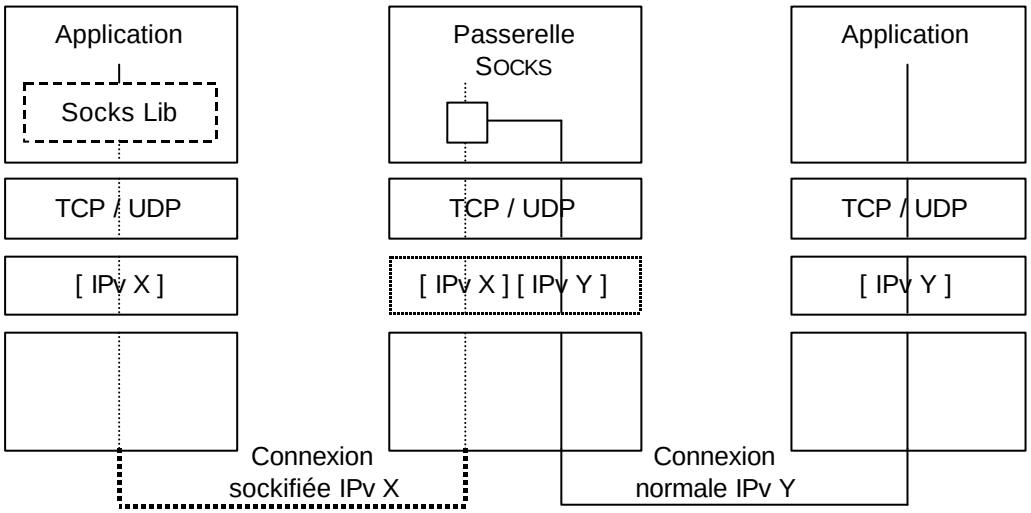
Le choix des adresses à utiliser pour faire la correspondance entre l'adresse IPv4 et l'adresse IPv6 est encore assez flou, mais il est possible que cela se fera en utilisant un préfixe spécial, routé par le Transport Relay (comme dans le cas de NAT-PT). Ce préfixe sera soit connu par les machines, soit envoyé dans les réponses du serveur DNS (dans le cas d'une demande d'adresse d'une machine IPv4).

Note : Comme dans le cas de la conversion d'en-tête, il peut y avoir des problèmes dans le cas où les données transmises contiennent des informations sur les couches inférieures (cas de FTP par exemple).

2.4.2 Socks [SOCKS]

SOCKS (http://www.socks.nec.com/translator.html) est un cas particulier de Transport Relay. Il nécessite que les applications qui l'utilisent soit 'Sockifiées'. Par contre, comme il utilise uniquement des noms longs (et pas des adresses IP), il n'y a pas de problème concernant l'adresse IPv4 à utiliser par la machine IPv6 et inversement.

Une fois les applications sur la machine cliente Sockifiées (procédure indiqué comme étant très simple : cela consiste à insérer une bibliothèque entre l'application et l'API des sockets pour intercepter les appels), la communication s'effectue entre la machine et la passerelle Socks en utilisant le protocole Socks (contenant à la fois des informations sur la connexion à réaliser et les données à transmettre). L'adresse IP utilisée n'est pas connu par la machine cliente qui n'utilise que des noms longs.



Socks peut donc être utile pour un réseau IPv4 qui veut communiquer avec une machine IPv6 et inversement. Dans le cas où le réseau utilise déjà Socks sur un réseau IPv4 (Socks est quelquefois utilisé comme Firewall), le seul fait d'installer une couche IPv6 sur la passerelle permet au réseau de communiquer avec des machines IPv6.

2.4.3 Application Proxy [TRANS]

Un proxy est souvent utilisé pour cacher les informations concernant un site privé (surtout s'il utilise un adressage privé) et également pour améliorer les performances en utilisant un mécanisme de cache. Mais si le proxy supporte les deux piles IP, il peut également servir à permettre la communication entre une machine IPv4 et une machine IPv6. Le seul inconvénient est qu'un proxy est spécifique à une application et qu'il ne traite donc que les paquets concernant cette application.

Exemples :
- Proxy Web
- Serveur Mail (MTA)

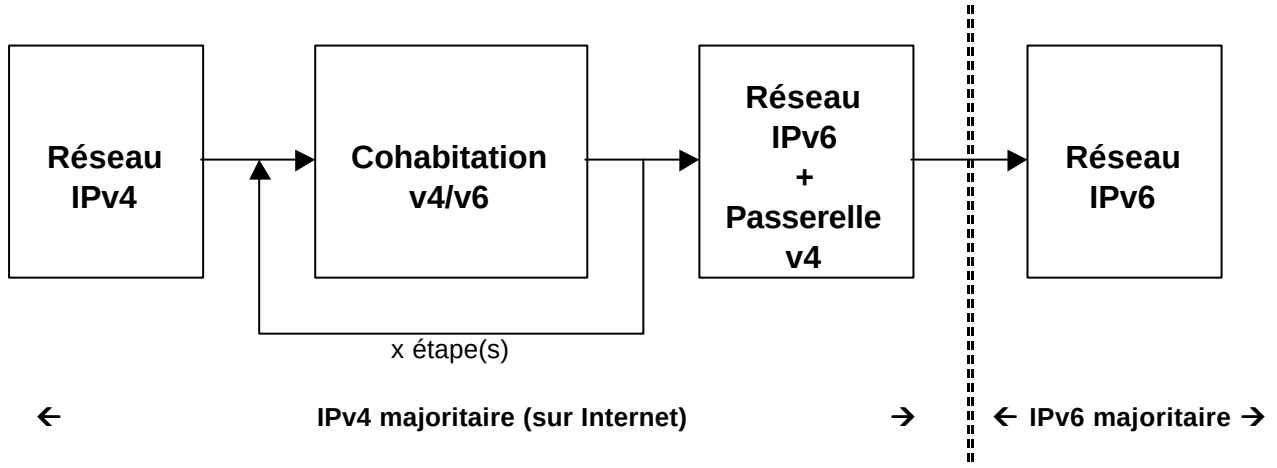
2.5 Résumé

	Mécanisme	Utilisation (la plus courante)
Encapsulation	Tunnel statique	Connexion de deux sites IPv6 par l'intermédiaire d'une architecture IPv4.
	Tunnel automatique	Connexion de deux machines IPv6 par l'intermédiaire d'une architecture IPv4.
	Tunnel Broker	Connexion d'un site IPv6 à un fournisseur d'accès IPv6 (par exemple pour se relier à un backbone IPv6).
	6over4	Réalisation d'un lien local IPv6 au dessus d'une architecture IPv4 Multicast.
	6to4	Connexion automatique de sites 6to4 par l'intermédiaire d'une architecture IPv4.
Double Pile	Double Pile	Communication soit en IPv4 soit en IPv6 si les applications, les systèmes et le réseau le permettent.
Traduction	Adresses IPv4 mappées	Communication entre une machine IPv6 et une machine IPv4 grâce à un traducteur d'en-tête (local ou distant).
	NAT-PT	Communication entre une machine IPv6 et une machine IPv4 grâce à un traducteur d'en-tête (distant).
	Bump-In-The-Stack	Utilisation d'une application IPv4 pour accéder à une machine IPv6 sur un réseau IPv6.
Passerelle	Transport Relay	Communication entre une machine IPv6 et une machine IPv4 en relayant les données de la couche transport.
	Socks	Communication entre une machine IPv6 et une machine IPv4 grâce à une passerelle Socks.
	Application Proxy	Communication entre une machine IPv6 et une machine IPv4 grâce à un proxy (dépendant de l'application).

3 Méthode de transition

3.1 Introduction

Le nombre de techniques de cohabitation v4/v6 étant importante, il n'est pas obligatoire de migrer vers IPv6 en une seule fois et en abandonnant définitivement IPv4. Dans la plupart des cas, on procédera par étape tout en gardant une possibilité de communiquer avec des machines IPv4-only. Le schéma ci-dessous montre les étapes nécessaire avant la migration complète.



Le but de cette partie est de donner une méthode générale permettant de faire la transition vers IPv6 en donnant tous les points délicats et les étapes à suivre. *Note :* Suivant l'étendue de la migration (qui peut aller du passage à IPv6 d'une simple machine à la migration complète d'un réseau multi-sites), tous les points ne seront pas forcement utiles.

3.2 Etude de l'existant (état source)

Avant de pouvoir choisir les techniques à utiliser, il faut connaître précisément le réseau actuel, les applications utilisées, les machines présentes sur le réseau, etc…

- **Plan du réseau – Plan d'adressage.** Il faut avoir une description détaillée du réseau (au moins de la partie à migrer) pour pouvoir prévoir les évolutions à effectuer et connaître la liste de tous les équipements.

- **Fournisseurs d'accès.** Se renseigner sur la position du ou des fournisseurs d'accès au sujet d'IPv6. Suivant leur état d'avancement, il sera éventuellement possible de se raccorder directement en IPv6 sur leur réseau, de bénéficier de passerelles de conversion avec IPv4, d'obtenir des préfixe IPv6, etc…

- **Equipements réseau.** Faire l'inventaire de tous les équipements réseau. Pour ceux qui montent jusqu'au niveau 3 (routeurs, quelques switchs, équipements configurables / interogeables avec un protocole au dessus d'IP), déterminer si leur fonctionnement en IPv6 est nécessaire (dont on se sert). Se renseigner sur les mises à jour éventuelles à effectuer pour leur permettre de gérer IPv6. Dans le cas, où il n'y a pas moyen de les mettre à jour, il faudra le prendre en compte dans la transition et prévoir le remplacement de l'équipement avant la fin de la migration.

- **Systèmes d'exploitation.** Faire l'inventaire de tous les systèmes d'exploitation (faire attention aux numéros de version mineures qui ont parfois une grande importance…). Pour chacun, se renseigner sur la possibilité d'installer une pile IPv6 ou éventuellement des solutions de remplacement (BITS, …). Pour les systèmes où l'on ne dispose d'aucune solution, il faudra le prendre en compte dans la transition et prévoir leurs remplacement avant la fin de la migration (Note : la durée de la migration peut être très longue et on peut rester très longtemps en phase de cohabitation pour éviter d'avoir à remplacer du matériel qui marche).

- **Logiciels utilisés.** Faire l'inventaire de tous les logiciels réseau utilisés. Pour chacun, vérifier l'existence de mises à jour / patches pour gérer IPv6. Pour les logiciels développés en interne, il faut prévoir d'effectuer les mises à jour nécessaires. S'il y a des logiciels essentiels qu'il ne sera pas possible de mettre à jour, il faudra utiliser d'autres solutions pour contourner le problème (BITS, Socks, …).

- **Méthodes de cohabitation.** Faire l'inventaire de toutes les méthodes de cohabitation possible (au cas où il en sorte de nouvelles). Pour chacune, vérifier qu'elle sera **effectivement implémentable** au moment où on compte faire la transition.

- **Protocoles.** Faire l'inventaire des protocoles qui circulent sur le réseau. Pour les protocoles 'réseau pur' (routage, DHCP, DNS, …), se renseigner sur les évolutions IPv6, sur les possibilités d'implémentation et les changements induits. Pour les protocoles 'applicatifs', vérifier qu'ils peuvent subir la traduction des paquets sans problème (pas comme FTP), où que l'on dispose de traduction spécifique (comme FTP). Pour le vérifier, on peut utiliser deux machines (une v4/v6 avec traduction des paquets et l'autre v4) sur lesquels on effectue les tests. En ce qui concerne l'inventaire des protocoles, on les classera en groupes suivant qu'ils sont utilisés en local, entre les sites ou vers l'Internet.

3.3 Détermination de l'état cible

Après avoir fait l'inventaire de ce dont on disposait et de ce qu'il était possible de faire, il faut déterminer la solution que l'on veut mettre en place. Pour cela, on commencera par éliminer toutes les solutions qui ne sont pas suffisamment avancées pour être appliquées sur des machines de production (où dont on n'est pas sûr qu'elles seront opérationnelles au moment où l'on en aura besoin).

- Si on veut communiquer avec l'Internet v6 et que le fournisseur d'accès ne nous fournit pas ce service, il faudra mettre en place des tunnels (statiques, automatiques, 6to4, …).

- Si on veut continuer à communiquer avec des machines IPv4 à partir de IPv6, il faudra prévoir soit des solutions sur les machines clientes (double pile, IPv4-mapped, …), soit des passerelles (NAT-PT, Socks, Proxy, …).

- De même, si on veut rester majoritairement IPv4, tout en communicant avec des machines IPv6 (éventuellement situées sur Internet), on passera également par des passerelles (NAT-PT, Socks, Proxy, …)

- Si on veut garder des applications IPv4, il faudra mettre en place un mécanisme comme BITS.

Note : L'état cible, n'est pas forcement un état où tout est passé en IPv6. On peut prévoir bon nombre de solutions permettant la cohabitation des deux protocoles.

3.4 Préparation de la transition

Une fois que l'état cible a été défini et que l'on a vérifié qu'il est techniquement possible et qu'il permettra de faire tout ce que l'on veut, on prépare la transition.

- **Détermination des besoins.** Déterminer et se procurer tous ce dont on a besoin (mises à jour, patchs, microcodes, logiciels, systèmes, documentation, …).

- **Vérification du bon fonctionnement.** Vérifier le bon fonctionnement des techniques et outils sur des machines de test (cette étape pourra être éventuellement sautée dans le cas de machines non critiques).

- **Déterminer le préfixe à utiliser.** Si le fournisseur de service le permet, demander un préfixe IPv6 global. Dans le cas contraire, on commencera avec les préfixes locaux ou les préfixes liés à certains mécanismes (6to4). L'absence d'un préfixe non global n'est pas gênant car les mécanismes de renumérotation permettront de renuméroter facilement tout le réseau le moment venu.

- **Définir le plan d'adressage.** Il ne s'agit pas de donner des adresses à chaque machine (plus nécessaire), mais de séparer le préfixe en plusieurs sous-réseaux. Dans le cas du réseau d'un opérateur, cette étape sera plus importante car il faudra réfléchir au découpage en sub-TLA, NLA et SLA et à l'affectation de ces préfixes aux éventuels clients. Il faudra éventuellement réserver des préfixes pour certains mécanismes (NAT-PT, Transport Relay, …).

3.5 Transition

Il faut définir les différents états stables permettant de passer (avec le moins d'interruption de service possible) de l'état source à l'état cible. L'ordre que l'on utilise pour migrer les machines n'est pas très important car les mécanismes de cohabitations permettent une migration dans n'importe quel ordre. Cependant, on préférera suivre un ordre logique (soit des routeurs vers les postes clients, soit l'inverse). Les étapes qui suivent donnent les différentes étapes pour passer d'un réseau totalement IPv4 à un réseau totalement IPv6. Il faudra l'adapter en fonction de la migration à effectuer.

- **Routeurs.** Installation d'une double pile IPv4/IPv6 sur les tous les routeurs de chaque site. On peut alors tester la communication IPv6 sur tous le site en utilisant des machines de test. Idem pour les switchs ayant des capacités de routage.

- **Passerelles et Traducteurs.** Installation des passerelles et des traducteurs que l'on a décidé d'utiliser. Installation également des liens vers les autres sites et vers l'extérieur en IPv6. Test de l'ensemble. Il pourra être nécessaire d'installer un serveur DNS gérant IPv6.

- **Equipements réseaux.** On installe également une double pile IPv4/IPv6 sur les équipements réseaux que l'on utilise en IP.

- **Serveurs.** On installe une double pile IPv4/IPv6 sur les serveurs et on migre les logiciels pour qu'ils gèrent à la fois IPv4 et IPv6. Si on a besoin de serveurs spécifiques (DHCPv6) ou ayant besoin d'une configuration spécifiques (DNS), on les met en place.

- **Postes Clients.** En fonction des besoins, on peut soit installer une double pile, soit une simple pile IPv6. On installe également certains mécanismes de cohabitation si on en a besoin (BITS, …). De même, suivant les besoins qu'ils ont, les nouveaux postes clients pourront n'être installés qu'en IPv6.

- **IPv6-only.** Petit à petit, on bascule tous les postes/serveurs/équipements en IPv6 only. On garde toujours les passerelles pour communiquer avec l'extérieur.

- **IPv6-only (bis).** Si Internet v6 est devenu très majoritaire, on peut abandonner les passerelles de conversion. Dans ce cas, ce seront les sites IPv4 qui chercheront à communiquer avec les sites IPv6 et non plus l'inverse…

4 Cas particuliers

Après la présentation d'une méthode générale, cette partie va donner les quelques mécanismes applicables dans chacun des cas. On présente les solutions qui seront les plus utilisées, sachant que l'on peut mettre en place de nombreuses solutions très spécifiques dans chacun des cas.

4.1 Réseau local

- Pour effectuer la migration d'un réseau local, la solution la plus simple est d'installer des doubles piles IPv4/IPv6 sur chaque machine désirant communiquer en IPv6. S'il faut faire communiquer des machines de plusieurs réseaux, il faudra que les routeurs intermédiaires disposent également d'une double pile IPv4/IPv6.

- On peut également utiliser des tunnels IPv6 pour relier des réseaux entre eux sans installer une double pile IPv4/IPv6 sur tous les routeurs intermédiaires.

- Dans le cas où les machines devant communiquer entre elles sont très éloignées les unes des autres, et que l'on ne désire pas mettre une double pile IPv4/IPv6 sur les routeurs, on peut utiliser le mécanisme de 6over4 (cf. § 2.1.4). Dans ce cas là, il faut que tous les routeurs supportent le Multicast pour pouvoir créer un lien-local IPv6 virtuel.

- Si on n'installe pas de double pile IPv4/IPv6 sur les routeurs, on peut également se servir de tunnels automatiques pour communiquer entre deux machines disposants chacune d'une adresse IPv4 accessible par l'autre.

- Si on utilise des VLAN et que la machine faisant le routage entre les VLAN ne supporte pas IPv6, on pourra installer une machine connectée à tous les VLAN pour faire le routage IPv6.

4.2 Connexion (directe) entre sites distants

- Pour relier deux sites distants en IPv6, la solution la plus simple est de construire un tunnel IPv6 entre les deux routeurs de sortie de site.

- Si on ne peut pas faire ce tunnel entre les deux routeurs de sortie, on peut également le faire entre deux autres machines (une de chaque côté), du moment qu'elles aient chacune une adresse IPv4 accessible par l'autre.

- Dans tous les cas, on peut également utiliser les solutions vues dans la partie précédente, mais il vaut mieux créer un unique lien entre les deux sites (sauf si la communication IPv6 est exceptionnelle).

4.3 Accès à Internet

- Pour accéder à des machines IPv6 sur Internet, tout dépend de la manière dont la machine est connectées au réseau. *Note :* Ces services pourront éventuellement être fournis par le fournisseur d'accès / Tous ces mécanismes ne sont pas forcement applicable derrière du NAT. :
  - Si elle est sur le 6bone, il suffit de se connecter à un point d'entrée du 6bone. On utilisera dans ce cas, soit un tunnel statique, soit un Tunnel Broker (cf. § 2.1.3).
  - Si elle est isolée sur un réseau IPv4, il faudra utiliser un tunnel automatique (utilisation d'une adresse IPv4-compatible).
  - Si elle est située sur un site 6to4 (cf. § 2.1.5) , il faudra avoir accès à un routeur 6to4 pour y accéder.

- Pour connecter deux sites en IPv6 et en passant par Internet, on pourra utiliser deux méthodes :
  - Un tunnel statique entre les deux routeurs de sortie.
  - L'installation de 6to4 sur les deux routeurs de sortie.

4.4 Cohabitation

- Si l'accès à des machines IPv4 à partir d'un réseau IPv6, ne se limite qu'à l'utilisation du mail et du web, on n'est pas obligé d'installer des passerelles de conversion, il suffit d'installer une double pile IPv4/IPv6 sur le serveur de mail et le proxy (on considère que l'on utilise un proxy). Cette solution est également utilisable pour accéder à des serveurs Web IPv6 alors que l'on est en IPv4.

- Si on utilise d'autres protocoles, il faudra installer des systèmes de conversion. *Note :* il faudra faire attention à ce que tous les protocoles utilisés supportent la conversion :
  - Conversion automatique en utilisant des adresses IPv4 mappées
  - Installation d'une passerelle NAT-PT (facile à mettre en place si on utilise déjà du NAT (ou une variante) sur le réseau).
  - Utilisation d'un Transport Relay.

- Si on utilise déjà Socks sur le réseau, il suffit d'installer une double pile IPv4/IPv6 sur la machine Socks, et toutes les machines pourront accéder aux deux réseaux sans problème et sans avoir besoin d'utiliser des logiciels IPv6.

- Si on veut continuer à utiliser des logiciels IPv4 sur un réseau IPv6, on utilisera soit une passerelle Socks (les logiciels devront cependant supporter Socks), soit le mécanisme de Bump-In-The-Stack sur la machine elle-même.

5 Migration de logiciels

Les changements induits par IPv6 sont suffisamment importants pour qu'il soit nécessaire de modifier les logiciels pour supporter IPv6. Cependant, pour la majorité des applications, il suffira de remplacer quelques fonctions par les nouvelles et de recompiler.

5.1 Applications avancées

Par applications avancées, on considère toutes les applications fortement liées à la structure des trames IP. Ces applications nécessitent donc un travail de portage plus important car elles accèdent aux trames IP, aux extensions, aux paquets ICMP, … On ne décrira pas tous les changements nécessaires au portage car ces applications sont souvent fournis avec le système (DHCP, ping, traceroute, routage, …). Pour plus d'information, on se reportera au RFC2292 – *Advanced sockets API for IPv6*.

5.2 Autres applications

L'accès à la couche IP se fait la plupart du temps en utilisant le mécanisme de sockets (que ce soit dans le monde Unix ou Windows). Il a donc été nécessaire de revoir l'API[3] pour intégrer l'utilisation des adresses IPv6. Ces modifications permettent d'intégrer IPv6 et ses nouvelles fonctionnalités tout en restant compatible avec l'ancien fonctionnement.

Note : Microsoft utilise sa propre API Winsock (la version 2.0 intègre IPv6), qui dérive de l'API d'origine (modèle BSD) et qui intègre des ajouts spécifiques. Les changements sont donc à peu de chose près équivalents.

Les changements au niveau de l'API sont les suivant :
- Changement de la structure contenant une adresse (on passe de 32 à 128 bits).
- Utilisation de nouveaux types pour indiquer que l'on travaille en IPv6.
- Nouvelle fonctions pour pouvoir gérer les conversions d'adresses.
- Nouvelles options pour ajouter les nouvelles fonctionnalités d'IPv6.

Normalement, il ne doit pas y avoir de changement à faire dans les données transmises par l'application. Le seul cas qui peut poser problème est le cas où l'on transmet des informations sur les adresses IP. Dans ce cas, il faudra revoir le protocole de la couche application.

5.2.1 Fonctions de base des sockets

5.2.1.1 Types d'adresse et de protocole

Comme on utilise un nouveau type d'adresse et de protocole, il a fallu définir de nouvelles constantes pour la création des sockets. Ces constantes sont définies dans le fichier `sys/socket.h`.
- `AF_INET6` : Adresse de type IPv6
- `PF_INET6` : Protocole IPv6

5.2.1.2 Structure d'une adresse IPv6

Le passage de la taille des adresses IPv6 à 128 bits nous oblige à utiliser des structures plus grande. On utilise donc `in6_addr` (défini dans `netinet/in.h`) au lieu de `in_addr` :

[3] Application Programming Interface.

```
struct in6_addr {
    uint8_t s6_addr[16]; /* IPv6 address */
};
```

5.2.1.3 Structure d'un socket

Une nouvelle structure de socket, `sockaddr_in6`, a été définie pour remplacer `sockaddr`. Cette structure est définie dans le fichier `netinet/in.h`.

Note : Il existe deux versions relativement proche des structures de socket (4.3BSD-based system, 4.4BSD-based system). Pour déterminer le type de système, on peut utiliser la constante SIN6_LEN qui n'est défini que dans le cas d'un système basé sur BSD 4.4.

- **BSD 4.3**

```
struct sockaddr_in6 {
   sa_family_t sin6_family;    /* AF_INET6 */
   in_port_t sin6_port;        /* transport layer port # */
   uint32_t sin6_flowinfo;     /* IPv6 traffic class & flow info */
   struct in6_addr sin6_addr;  /* IPv6 address */
   uint32_t sin6_scope_id;     /* set of interfaces for a scope */
};
```

- **BSD 4.4**

```
struct sockaddr_in6 {
    uint8_t sin6_len;           /* length of this struct */
    sa_family_t sin6_family;    /* AF_INET6 */
    in_port_t sin6_port;        /* transport layer port # */
    uint32_t sin6_flowinfo;     /* IPv6 traffic class & flow info */
    struct in6_addr sin6_addr;  /* IPv6 address */
    uint32_t sin6_scope_id;     /* set of interfaces for a scope */
};
```

5.2.1.4 Fonctions sur un socket

La fonction `socket()` (initialisation d'un socket) ne change pas de syntaxe. Par contre, on doit indiquer le type d'adresse à utiliser.

IPv4 :

```
s_tcp = socket(PF_INET, SOCK_STREAM, 0);
s_udp = socket(PF_INET, SOCK_DGRAM, 0);
```

IPv6 :

```
s_tcp = socket(PF_INET6, SOCK_STREAM, 0);
s_udp = socket(PF_INET6, SOCK_DGRAM, 0);
```

Toutes les autres fonctions ne changent pas non plus de syntaxe car on utilise toujours un pointeur sur un sockaddr, avec la longueur de l'adresse.

```
int bind (int sockfd, struct sockaddr *localaddr, int addrlen);
int accept (int sockfd, struct sockaddr *foreignaddr, int addrlen);
int connect (int sockfd, struct sockaddr *foreignaddr, int addrlen);
...
```

5.2.1.5 Adresses spéciales

- L'adresse wildcard permet de laisser le système déterminer l'adresse source à utiliser dans une communication. Comme l'adresse IPv6 est maintenant une structure (auparavant, c'était un mot de 32bits), on ne peut plus utiliser la constante dans tous les cas (seulement à l'initialisation d'une variable). On a donc ajouté une variable globale que l'on peut transmettre en paramètre d'une fonction.
  - **IPv4** : `INADDR_ANY` (constante)
  - **IPv6** : `IN6ADDR_ANY_INIT` (constante) et `in6addr_any` (variable globale)

- L'adresse de loopback permet d'envoyer des paquets en direction d'un socket situé sur la même machine.
  - **IPv4** : `INADDR_LOOPBACK` (constante)
  - **IPv6** : `IN6ADDR_LOOPBACK_INIT` (constante) et `in6addr_loopback` (variable globale)

5.2.1.6 Options d'un socket

Les fonctions `getsockopt()` et `setsockopt()` permettent respectivement de récupérer et de fixer des options sur un socket. La déclaration de ces fonctions est la suivante :

```
int getsockopt(int socket, int level, int option_name,
               void *option_value, socklen_t *option_len);
int setsockopt(int socket, int level, int option_name,
               const void *option_value, socklen_t option_len);
```

Le paramètre `level` permet d'indiquer à quel protocole l'option est attachée. Un certains nombre d'options ont été ajoutées pour le protocole IPv6. Pour ces options, on utilisera donc le level `IPPROTO_IPV6`.

- `IPV6_UNICAST_HOPS` : Nombre maximal de sauts avant destruction (hop limit).
- `IPV6_MULTICAST_IF` : Interface à utiliser pour les paquets Multicast.
- `IPV6_MULTICAST_HOPS` : Hop limit pour le trafic Multicast.
- `IPV6_MULTICAST_LOOP` : Indique si un paquet Multicast émis doit être envoyé à soi même.
- `IPV6_JOIN_GROUP` : Adhère à un groupe Multicast.
- `IPV6_LEAVE_GROUP` : Quitte un groupe Multicast.

5.2.2 Fonctions annexes

5.2.2.1 Conversion Adresse IP – Nom long

La fonction gethostbyname() a été remplacé par getipnodebyname() pour pouvoir traiter à la fois les adresses IPv4 et les adresses IPv6.

```
struct hostent *getipnodebyname(const char *name, int af,
                                int flags, int *error_num);
```

Le paramètre af indique la famille d'adresse que l'on désire (`AF_INET` ou `AF_INET6`). Dans la plupart des cas, il suffira de remplacer :

```
hptr = gethostbyname(name);
```

par

```
hptr = getipnodebyname(name, AF_INET6, AI_DEFAULT, &error_num);
```

Le nom est soit un nom long (www.ei-rsi.com), soit une adresse IP littérale (192.168.4.54 ou `FE80::34B:1239:43B1:23CD`). L'application n'a donc pas à se soucier de savoir sous quelle forme l'adresse a été fournie.

Le paramètre flag permet d'indiquer le comportement de la fonction en présence d'adresses IPv4 ou IPv6 pour le nom long donné en paramètre. Par défaut (0), on ne renvoie que l'adresse qui est du même type que af.

- `AI_V4MAPPED` : renvoie une adresse IPv4-mapped quand il y a une adresse IPv4.
- `AI_ALL` : employé avec `AI_V4MAPPED`, on renvoie les adresses IPv4-mapped et les adresses IPv6.
- `AI_ADDRCONFIG` : On renvoie une adresse d'un certain type (IPv4-mapped / IPv6) que si l'on dispose d'une adresse de ce type sur la machine.
- `AI_DEFAULT` : `(AI_ADDR_CONFIG | AI_V4MAPPED)`

Cette fonction peut donc être utilisé pour communiquer en toute transparence avec des machines IPv4 ou IPv6.

De la même façon, pour faire la résolution inverse, on a remplacé la fonction `gethostbyaddr()` par `getipnodebyaddr()` :

```
struct hostent *getipnodebyaddr(const void *src, size_t len, int af,
                                int *error_num);
```

Si af = AF_INET6, on peut soit donner une adresse IPv4-mapped, soit une adresse IPv6.

5.2.2.2 Conversion d'adresse

Les fonctions de conversion d'adresse permettent de transformer une adresse IP en une chaîne imprimable, et inversement. Les fonctions `inet_addr()` et `inet_ntoa()` ont été remplacées par les fonctions `inet_pton()` et `inet_ntop()`.

```
int inet_pton(int af, const char *src, void *dst);
const char *inet_ntop(int af, const void *src, char *dst, size_t size);
```

5.2.2.3 Macros

Des macros ont été définies pour permettre de tester le type des adresses. Il s'agit de :

- `int IN6_IS_ADDR_UNSPECIFIED (const struct in6_addr *);`
- `int IN6_IS_ADDR_LOOPBACK (const struct in6_addr *);`
- `int IN6_IS_ADDR_MULTICAST (const struct in6_addr *);`
- `int IN6_IS_ADDR_LINKLOCAL (const struct in6_addr *);`
- `int IN6_IS_ADDR_SITELOCAL (const struct in6_addr *);`
- `int IN6_IS_ADDR_V4MAPPED (const struct in6_addr *);`
- `int IN6_IS_ADDR_V4COMPAT (const struct in6_addr *);`

- `int IN6_IS_ADDR_MC_NODELOCAL(const struct in6_addr *);`
- `int IN6_IS_ADDR_MC_LINKLOCAL(const struct in6_addr *);`
- `int IN6_IS_ADDR_MC_SITELOCAL(const struct in6_addr *);`
- `int IN6_IS_ADDR_MC_ORGLOCAL (const struct in6_addr *);`
- `int IN6_IS_ADDR_MC_GLOBAL (const struct in6_addr *);`

5.3 Migration

Si l'application a été bien conçue, tous le code concernant l'utilisation des sockets est concentré à un seul endroit et la migration sera facile. Il suffit de faire attention aux points suivants :

- Vérifier que le protocole au niveau application est indépendant de la couche IP utilisée.
- Modifier les structures contenant les adresses (`in6_addr` au lieu de `in_addr`) et les sockets (`sockaddr_in6` au lieu de `sockaddr`).
- Vérifier tous les appels de fonctions de type socket en modifiant si besoin est les noms, paramètres, types, constantes, … pour gérer aussi bien IPv4 que IPv6.
- Vérifier qu'il n'y a pas d'adresse IPv4 en dur dans le code.

Note : Sun dispose d'un outil permettant de migrer des applications IPv4 vers IPv6. Il s'agit du *IPv6 Socket Scrubber* qui analyse le code source de l'application et détecte les appels spécifiques à IPv4.
Pour plus d'informations : http://www.sun.com/solaris/ipv6/

6 Conclusion

La migration vers IPv6 pose de nombreux problèmes, et c'est pour cela que l'IETF a constitué très tôt un groupe de travail exclusivement consacré à cela. Il en ressort un certains nombres de mécanismes qui vont faciliter cette migration.

De plus, comme on ne passera pas à IPv6 en un seul jour (mais plutôt en plusieurs années), il a fallu développer des techniques de cohabitation des deux couches IP. Ces techniques permettent également de commencer à migrer des réseaux à tous les niveaux : on n'est pas obligé d'attendre que son fournisseur d'accès commence à migrer pour effectuer cette transition. Cela va permettre aux entreprises de se lancer dans IPv6 même si les fournisseurs d'accès sont encore assez frileux dans le domaine.

En conclusion, on peut dire que si IPv6 devient incontournable dans les années à venir, la migration va occuper les informaticiens pendant plusieurs années. Que ce soit pour faire la migration des réseaux, des systèmes et des logiciels…

7 Annexe 1 : Bibliographie

7.1 Techniques de cohabitation

[INTRO] *A Guide to the Introduction of IPv6 in the IPv4 World* – W. Biemolt & M. Kaat & T. Larder & R. van der Pol & H. Steenman – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-introduction-to-ipv6-transition-02.txt.

[MECH] *Transition Mechanisms for IPv6 Hosts and Routers* – R. E. Gilligan & E. Nordmark – Mai 1999 – Work In Progress – draft-ietf-ngtrans-mech04.txt.

[BROKER] *IPv6 Tunnel Broker* – A. Durand & P. Fasano & I. Guardini & D. Lento – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-broker-02.txt.

[RFC2529] *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels* – B. Carpenter & C. Jung – Mars 1999 – Statut : Draft Standard.

[6TO4] *Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels* – B. Carpenter & K. Moore – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-6to4-03.txt.

[SIIT] *Stateless IP/ICMP Translation Algorithm (SIIT)* – E. Nordmark – Décembre 1999 – Work In Progress – draft-ietf-ngtrans-siit-08.txt.

[NAT-PT] *Network Address Translation – Protocol Translation (NAT-PT)* – G. Tsirtsis & P. Srisuresh – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-natpt-07.txt.

[BITS] *Dual Stack Hosts using the Bump-In-the-Stack Technique (BIS)* – K. Tsuchiya & H. Higuchi & Y. Atarashi – Juillet 1999 – Work In Progress – draft-ietf-ngtrans-bis-00.txt.

[TRANS] *Categorizing Translators between IPv4 and IPv6* – K. Yamamoto & M. Sumikawa – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-translator-02.txt.

[RELAY] *An IPv6-to-IPv4 transport relay translator* – J. Hagino & K. Yamamoto – Janvier 2000 – Work In Progress – draft-ietf-ngtrans-tcpudp-relay-00.txt.

[SOCKS] *A SOCKS-based IPv6/IPv4 Gateway Mechanism* – H. Kitamura & A. Jinzaki & S. Kobayashi – Juillet 1999 – Work In Progress – draft-ietf-ngtrans-socks-gateway-02.txt.

[DSTM] *Dual Stack Transition Mechanism (DSTM)* – J. Bound & L. Toutain & H. Afifi – Octobre 1999 – Work In Progress –draft-ietf-ngtrans-dstm-00.txt.

7.2 Méthode de transition

[INTRO] *A Guide to the Introduction of IPv6 in the IPv4 World* – W. Biemolt & M. Kaat & T. Larder & R. van der Pol & H. Steenman – Octobre 1999 – Work In Progress – draft-ietf-ngtrans-introduction-to-ipv6-transition-02.txt.

7.3 Migration de logiciels

[RFC2553] *Basic Socket Interface Extensions for IPv6* – R. Gilligan & S. Thomson & J. Bound & W. Stevens – Mars 1999 – Statut : Informational.

[RFC2292] *Advanced Sockets API for IPv6* – W. Stevens & M. Thomas – Février 1998 – Statut : Informational.