HOWTO IPv6 Linux (fr)

Peter Bieringer

pb chez bieringer point de

Historique des versions

Version 0.49.fr.1	26−02−2006	Revu par : MB
Voirl'historique des révisions pour plus de détails		
Version 0.48.1.fr.1	20−01−2005	Revu par : MB
Voir l'historique des révisions pour plus de détails		
Version 0.47.fr.1	05−09−2004	Revu par : MB
Voir l'historique des révisions pour plus de détails		
Version 0.44.fr.1	05−09−2003	Revu par : MB
Voir l'historique des révisions pour plus de détails		

L'objet de cet HOWTO IPv6 Linux est de répondre à la fois aux questions basiques et avancées au sujet d'IPv6 sur le système d'exploitation Linux. Cet HOWTO fournira au lecteur assez d'information pour installer, configurer et utiliser les applications IPv6 sur les machines Linux.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Chapitre 1. Généralités

Vous trouverez les informations concernant les différentes traductions disponibles dans la section Traductions.

1.1. Copyright, licence et autres

1.1.1. Copyright

1.1.2. Licence

1.1.3. A propos de l'auteur

1.1.3.1. L'auteur, Internet et IPv6

- 1993: J'ai pris contact avec l'Internet par la pratique du mél et des news sur un client en mode texte (par exemple, rechercher "e91abier" sur groups.google.com, c'est moi).
- 1996: J'ai été sollicité pour concevoir un cours sur IPv6, incluant des travaux pratiques sur le système d'exploitation Linux.
- 1997: Début de la rédaction d'un guide sur la façon d'installer, de configurer et d'utiliser IPv6 sur les systèmes Linux, appelé HowTo − IPv6 & Linux (voir IPv6 & Linux − HowTo/History pour plus d'information).
- 2001: Début de la rédaction de cet HOWTO IPv6 Linux.

1.1.3.2. Contact

L'auteur peut être contacté par mél à <pb chez bieringer point de> mais aussi *via* sa page personnelle.

Il vit actuellement à Munich [dans la partie nord du Schwabing] / Bavière / Allemagne (sud) / Europe (centrale) / Terre (surface/continent).

1.2. Catégorie

Cet HOWTO relève de la catégorie "*Réseau/Protocoles*".

1.3. La version, l'historique et ce qu'il reste à faire

1.3.1. La version

La version actuelle est visible dès le début de ce document.

En ce qui concerne les autres versions/traductions, voir égalementhttp://www.bieringer.de/linux/IPv6/.

1.3.2. L'historique

1.3.2.1. L'essentiel de l'historique

30−11−2001:Début de la conception du nouvel HOWTO.

02−01−2002: Une quantité importante du contenu est achevée, publication de la première version du premier chapitre (version 0.10).

14−01−2002: Plus achevé, avec relectures, publication de la première version complète du document (version 0.14).

16−08−2002: La traduction polonaise est en cours

31−10−2002: La traduction chinoise est disponible (voir les traductions pour en savoir plus)

10−11−2002: La traduction allemande est en cours

10−02−2003: La traduction allemande est disponible

09−04−2003: La traduction francophone est en cours

09−05−2003: La traduction francophone est disponible

15−08−2003: La traduction espagnole est en cours

16−10−2003: La traduction italienne est en cours

12−03−2004: La traduction italienne est disponible

18−06−2004: La traduction grecque est en cours

1.3.2.2. L'historique complet

Voir l'historique des révisions à la fin de ce document.

1.3.3. Ce qu'il reste à faire

- Rédiger les contenus manquants
- Achever la correction orthographique

1.4. Les traductions

Les traductions doivent toujours contenir l'URL, le numéro de version et le copyright du document original (le vôtre aussi). Merci de ne pas traduire le journal original des modifications, ce n'est vraiment pas utile. Il apparaît que la fréquence des modifications apportées à ce document est, la plupart du temps, inférieure à une fois par mois. Depuis la version 0.27, il apparaît aussi que la plus grande part du contenu fourni par moi−même a été rédigée. Les traductions doivent toujours prendre comme source la version anglo−saxonne.

1.4.1. Traductions disponibles

1.4.1.1. En langue française

La traduction francophone par Michel Boucey a été mise en chantier le 9 avril 2003, à partir de la révision 0.41.1. Elle est disponible depuis le 9 mai 2003 sur Deep Space 6, avec pour URL original mirrors.deepspace6.net / Linux+IPv6−HOWTO−fr. Je (*Michel Boucey*) remercie par avance toute personne qui aidera, de quelque façon, à améliorer cette traduction. On peut me contacter à l'adresse mél <mboucey chez free point fr>.

1.4.1.2. Les autres traductions disponibles

L'information concernant les traductions disponibles en d'autres langues que l'anglais et le français peut être trouvée dans le document original: TLDP / Linux+IPv6−HOWTO / Translations

1.5. Un peu de technique

1.5.1. Le document original de cet HOWTO

Cet HOWTO est actuellement rédigé avec la version 1.2.0 de LyX sur un système Linux Red Hat 7.3 avec un patron SGML (livre DocBook). Il est disponible en vue des contributions à l'URLTLDP−CVS / users / Peter−Bieringer.

1.5.1.1. Modification des lignes de code propres à LyX

Les modifications des lignes de code propres à LyX sont réalisées par un script "maison", "lyxcodelinewrapper.pl", que vous pouvez obtenir par CVS pour votre propre compte: TLDP−CVS / users / Peter−Bieringer (*NdT*: ces lignes ne gênent pas la génération au format SGML, mais celles aux formats PS et PDF à partir du SGML généré couramment, *i.e.* sans ce script).

1.5.1.2. La génération du SGML

Le code SGML est généré en utilisant la fonction d'exportation de LyX.

Des solutions ont été apportées afin de créer un code SGML plus propre (voir aussi ici pour le programme Perl, TLDP−CVS / users / Peter−Bieringer):

- L'exportation du document LyX ne créait pas proprement les balises "colspan" − l'outil qui règle le problème: "sgmllyxtabletagfix.pl" (le problème est définitivement réglé depuis la version 1.2.0 de LyX)
- LyX utilise parfois des entités spéciales gauche/droite, à la place des guillemets habituels, qui seront présentes dans le code HTML. Certains navigateurs n'interprètent pas très bien ces balises (Opéra 6 TP 2 ou Konquéror sont connus pour ce problème) − l'outil qui règle le problème: "sgmllyxquotefix.pl"

1.5.2. Les références en ligne à la version HTML de cet HOWTO (lien/ancrage)

1.5.2.1. La page d'index maître

Généralement, une référence vers la page d'index maître est recommandée.

1.5.2.2. Les pages dédiées

Parce que les pages HTML sont générées à partir du fichier SGML, le nommage des fichiers HTML prend une tournure aléatoire. Et cependant, certaines pages ont des balises assignées par LyX, dont il résulte un nommage constant. Ces balises sont très utiles aux références et ne devraient pas être changées à l'avenir.

Si vous pensez que j'ai oublié une balise, merci de me le faire savoir, et je l'ajouterai.

1.6. Préface

Quelques petites choses d'abord:

1.6.1. Combien se promène−t−il de versions de l'HOWTO Linux & IPv6?

En incluant celui−ci, il y a trois documents HOWTO disponibles. Mes excuses si cela vous semble de trop ;)

1.6.1.1. La FAQ/HOWTO IPv6 Linux (obsolète)

Le premier document relatif à IPv6 a été écrit par *Eric Osborne,* et s'appelle FAQ/HOWTO IPv6 Linux (merci de ne l'utiliser que pour des raisons historiques). La dernière version fut la 3.2.1, publiée le 14 juillet 1997.

Merci de m'aider: si quelqu'un connaît la date anniversaire de cet HOWTO, merci de m'envoyer un mél (cette information est nécessaire à "l'historique").

1.6.1.2. L'HowTo − IPv6 & Linux (maintenu)

Il existe une seconde version appelée HowTo − IPv6 & Linux − écrite par moi−même (*Peter Bieringer*) en pur HTML. Elle est née en avril 1997 et la première version anglo−saxonne a été publiée en juin 1997. Je continuerais à la maintenir, mais cela déclinera lentement (mais pas complètement) en faveur de l'HOWTO

IPv6 Linux que vous lisez en ce moment.

1.6.1.3. L'HOWTO IPv6 Linux (ce document)

Parce que l'HowTo – IPv6 & Linux est écrit en HTML pur, il n'est vraiment pas compatible avec le Projet de Documentation Linux (*Linux Documentation Project,* ou TLDP). J'ai (*Peter Bieringer*) reçu une demande fin novembre 2001 de réécriture de l'HowTo –IPv6 & Linux en SGML. Cependant, à cause de la discontinuité de cet HOWTO (le future de l'HowTo – IPv6 & Linux), et de la standardisation croissante d'IPv6, je décidais d'écrire un nouveau document couvrant aussi bien les questions simples ou avancées qui resteront importantes dans les toutes prochaines années. Plus dynamique, un contenu plus avancé s'y trouvera en plus, par rapport au second HOWTO (HowTo – IPv6 & Linux).

1.7. Termes employés, glossaire et abréviations

1.7.1. Relatifs aux réseaux

Base 10
: Le système bien connu des nombres décimaux, représentant n'importe quelle valeur avec les chiffres 0–9.

Base 16
: Habituellement utilisée dans les langages de programmation de bas et haut niveaux, connue encore en tant que système numérique hexadécimal, représentant les valeurs avec les chiffres 0–9 et les caractères A–F (insensible à la casse).

Base 85
: Représentation d'une valeur grâce à 85 différents chiffres/caractères, cela permet des chaînes de caractères plus courtes mais jamais vue dans la pratique.

Bit
: Unité minimale de stockage, allumée(*on*)/vraie (1) ou éteinte(*off*)/fausse (0).

Byte
: Le plus souvent, une collection de 8 bits (mais ce n'est pas réellement une nécessité – regardez les systèmes des anciens ordinateurs).

Périphérique
: ici, matériel de connexion réseau, voir aussi NIC.

Hôte à double résidence
: Un hôte à double résidence est un noeud ayant deux interfaces réseau (physique ou virtuelle) sur deux liens différents, mais qui ne réalise pas de renvoi de paquets entre les interfaces.

Hôte
: Généralement, un hôte simple résident, présent sur un lien. Normalement, il n'a seulement qu'une interface réseau active, par exemple Ethernet ou (non pas *et*) PPP.

Interface
: quasi–synonyme de "périphérique", voir aussi NIC.

En–tête IP
: En–tête d'un paquet IP (chaque paquet réseau a un en–tête, son type dépendant de la couche réseau).

Lien
: Un lien est un médium de transport de paquet réseau de la couche 2, des exemples en sont Ethernet, PPP, SLIP, ATM, RNIS, Frame Relay, *etc.*

Noeud
: Un noeud est soit un hôte, soit un routeur.

Octet
: Une collection véritable de 8 bits, aujourd'hui synonyme de "byte".

Port
: Information destinée au distributeur TCP/UDP (couche 4) afin de transporter l'information à la couche supérieure.

Protocole
: Chaque couche réseau contient la plupart du temps un champ "protocole" facilitant la distribution de l'information transportée à la couche supérieure, comme cela peut se voir dans la couche 2 (MAC) et 3 (IP)

Routeur
: Un routeur est un noeud possédant une ou plusieurs interface(s) réseau, capable d'envoyer les paquets entre ses interfaces.

Socket
: Une socket IP est définie par ses adresses source et destination, ses ports et (association)

Pile
: Une collection de couches relative au réseau.

Masque de sous−réseau
: Les réseaux IP utilisent un masque de bits afin de distinguer le réseau local de ceux qui sont distants.

Tunnel
: Un tunnel est typiquement une connexion point−à−point sur laquelle les paquets échangés transportent les données d'un autre protocole, un tunnel IPv6−in−IPv4 en est un exemple.

1.7.1.1. Abréviations

ACL
: *Access Control List*, Liste de Contrôle d'Accès

API
: *Application Programming Interface*, Interface de Programmation d'Application

ASIC
: *Application Specified Integrated Circuit*, Circuit Intégré d'Application Spécifique

BSD
: *Berkeley Software Distribution*, Distribution des Logiciels Berkeley

Bus CAN
: Système de bus physique contrôlant un réseau (*NdT*: voir par exemple ici pour plus d'information)

ISP
: *Internet Service Provider*, Fournisseur d'Accès à Internet (FAI)

KAME
: Projet − effort conjoint de six entreprises au Japon pour fournir, mondialement et dans le cadre du logiciel libre, une pile IPv6 et IPsec (pour IPv4 et IPv6) pour les variantes de BSD www.kame.net

LIR
: *Local Internet Registry*, Bureau local d'enregistrement Internet

NIC
: *Network Interface Card*, Carte d'interface réseau

RFC
: *Request for comments*, Appel à commentaires − jeu de notes techniques et organisationnelles au sujet d'Internet.

USAGI
: Projet "UniverSAl playGround for IPv6" − travaille à rendre disponible une pile protocolaire IPv6 destinée au système Linux qui soit d'une qualité apte à la production.

1.7.2. Relatifs à ce document

1.7.2.1. Balisage en vue de l'encodage PDF/PS

Le caractère "¬" est utilisé pour signaler que le code est enveloppé en vue d'un meilleur affichage dans les fichiers PDF et PS.

1.7.2.2. Conventions

Dans les exemples génériques vous trouverez parfois ce qui suit:

```
<monadresseip>
```

Pour une utilisation réelle sur votre système, en ligne de commande ou dans des scripts, cela doit être remplacé par le contenu adéquate (ôtez bien sûr les chevrons), et le résultat devrait être par exemple

```
1.2.3.4
```

1.7.2.3. Les commandes dans l'interpréteur de commandes (le *shell*)

Les commandes exécutables en tant qu'utilisateur non−root commencent avec un $, par exemple

```
$ whoami
```

Les commandes exécutables en tant qu'utilisateur root commencent avec un #, par exemple

```
# whoami
```

1.8. Pré−requis à l'usage de cet HOWTO

1.8.1. Pré−requis personnels

1.8.1.1. Une expérience des outils Unix

Vous devriez être familiarisé avec les outils essentiels d'Unix comme *grep*, *awk*, *find*, *etc*, et connaître les options de ligne de commande les plus communément employées.

1.8.1.2. Une expérience de la théorie des réseaux

Vous devriez connaître les notions de couche, de protocole, d'adresse, de câble, de socket, *etc*. Si vous êtes nouveau, voici un bon point de départ pour vous: linuxports/howto/intro_to_networking

1.8.1.3. Une expérience de la configuration IPv4

Vous devriez absolument avoir quelque expérience de la configuration IPv4, sinon ce sera difficile pour vous de comprendre ce qui se passe réellement.

1.8.1.4. Une expérience du Système des Noms de Domaine (DNS)

Vous devriez aussi comprendre en quoi consiste le Système des Noms de Domaine (DNS), ce qu'il fournit et comment s'en servir.

1.8.1.5. Une expérience des stratégies de déboguage réseau

Vous devriez au moins savoir comment utiliser *tcpdump* et avoir connaissance de ce qu'il peut vous montrer. Sinon, le déboguage réseau sera très difficile pour vous.

1.8.2. Le matériel compatible avec le système d'exploitation Linux

Vous espérez certainement pouvoir expérimenter tout cela avec du vrai matériel, et pas seulement lire cet HOWTO jusqu'à tomber de sommeil. ;−7)

Chapitre 2. Les bases

2.1. Qu'est–ce qu'IPv6?

IPv6 est un nouveau protocole de la couche 3 (voir le modèle OSI) qui supplantera à terme IPv4 (plus connu sous le nom d'IP). IPv4 a été conçu il y a déjà un certain de temps (RFC 760 / Le protocole Internet à partir de janvier 1980), et, dès le début, il y a eu de nombreuses demandes pour accroître la quantité d'adresses disponible et augmenter les capacités. Le RFC le plus récent est le RFC 2460 / spécification du protocole Internet version 6 (*NdT*: une version francophone de ce RFC). Le changement essentiel apporté par IPv6 est la nouvelle conception de l'en–tête, incluant une augmentation de la taille de l'adresse, passant de 32 à 128 bits. Parce que la couche 3 est responsable de bout en bout du transport des paquets dont le routage est basé sur des adresses, elle doit inclure les nouvelles adresses IPv6, comme pour IPv4.

Pour en savoir plus sur l'histoire d'IPv6, jetez un oeil aux anciens RFC concernant IPv6, par exemple dans le Guide / Références IPv6 SWITCH .

2.2. Historique d'IPv6 pour Linux

Les années 1992, 1993 et 1994 de l'histoire d'IPv6 (dans ses généralités) sont couvertes par le document suivant: IPv6 ou IPng (IP nouvelle génération).

A faire: plus de détails historiques, plus de contenu...

2.2.1. Au début

Le premier code réseau relatif à IPv6 a été ajouté au noyau Linux 2.1.8 en novembre 1996 par Pedro Roque. Il était fondé sur l'API BSD:

```
diff –u --recursive --new-file v2.1.7/linux/include/linux/in6.h
¬ linux/include/linux/in6.h
--- v2.1.7/linux/include/linux/in6.h Thu Jan 1 02:00:00 1970
+++ linux/include/linux/in6.h Sun Nov 3 11:04:42 1996
@@ –0,0 +1,99 @@
+/*
+ * Types and definitions for AF_INET6
+ * Linux INET6 implementation
+ * + * Authors:
+ * Pedro Roque <******>
+ *
+ * Source:
+ * IPv6 Program Interfaces for BSD Systems
+ * <draft-ietf-ipngwg-bsd-api-05.txt>
```

Les lignes présentées sont copiées du patch–2.1.8 (l'adresse mél a été effacée au copier&coller).

2.2.2. Après

A cause du manque de bras, l'implémentation d'IPv6 dans le noyau était incapable de suivre les projets discutés ou les RFC nouvellement mis à jour. En novembre 2000, un projet débute au Japon, appelé USAGI, dont le but était d'implémenter dans Linux tout le support IPv6 manquant ou obsolète. Ce projet suit en cela la trace de l'implémentation courante d'IPv6 pour FreeBSD, réalisée par le projet KAME . De temps à autre, ils

créaient des archives de développement (*snapshots*) à partir des sources courantes du noyau Linux.

2.2.3. Actuellement

Malheureusement, le patch USAGI est très volumineux, à tel point que les personnes s'occupant actuellement de maintenir les fonctionnalités réseau de Linux sont incapables de l'inclure dans les sources, aptes à la production, de la série des noyaux Linux 2.4.x. En conséquence, la série 2.4.x manque de certaines (et même de nombreuses) extensions, et elle n'applique pas non plus les brouillons et RFC courants (voir le groupe de travail IP Version 6 (ipv6)). Cela peut poser des problèmes d'interopérabilité avec les autres systèmes d'exploitation.

2.2.4. A l'avenir

USAGI fait maintenant usage de la série des noyaux de développement Linux 2.5.x afin d'incorporer toutes les extensions actuelles dans cette version de développement; dans l'espoir que la série des noyaux 2.6.x comprenne une véritable implémentation à jour d'IPv6.

2.3. A quoi ressemblent les adresses IPv6?

Comme cela a été mentionné précédemment, les adresses IPv6 ont une longueur de 128 bits. Ce nombre de bits génère de très grands nombres, dont la quantité de chiffres est supérieure à 39:

```
2^128-1: 340282366920938463463374607431768211455
```

De tels nombres ne sont vraiment pas des adresses pouvant être mémorisées. L'adresse IPv6 en elle−même est faite à partir d'une collection de bits (comme pour IPv4, bien que cela soit rarement su). Il y a une meilleure notation pour de si grands nombres, qui est l'hexadécimal. En hexadécimal, 4 bits (mot aussi connu sous la dénomination de "nibble") sont représentés par un chiffre ou un caractère de 0−9 et a−f (10−15). Ce format réduit la longueur de l'adresse IPv6 à 32 caractères.

```
2^128-1: 0xffffffffffffffffffffffffffffffff
```

Cette représentation est encore peu praticable (possibilité de confusion ou de perte d'un simple chiffre hexadécimal), c'est pourquoi les concepteurs d'IPv6 ont choisi un format hexadécimal scindé en blocs de 16 bits, avec comme séparateur le caractère ":". De plus, le préfixe "0x" (le marqueur des valeurs hexadécimales utilisé dans les langages de programmation) est ôté:

```
2^128-1: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

Une adresse utilisable (nous verrons les différents types d'adresse plus tard) est par exemple:

```
3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
```

Dans un but de simplification, les zéros non significatifs de chaque bloc de 16 bits sont omis:

```
3ffe:ffff:0100:f101:0210:a4ff:fee3:9566  ->
¬ 3ffe:ffff:100:f101:210:a4ff:fee3:9566
```

Une séquence de blocs de 16 bits ne comprenant que des zéros peut être remplacée par "::". Mais pas plus d'une fois par adresse, sinon il ne s'agirait plus d'une représentation unique.

```
3ffe:ffff:100:f101:0:0:0:1  ->  3ffe:ffff:100:f101::1
```

La plus importante réduction qui peut être observée est celle de l'adresse localhost d'IPv6:

```
0000:0000:0000:0000:0000:0000:0000:0001  ->  ::1
```

Il existe aussi une représentation dite *compacte*, encodée en base85 (RFC 1924 / A Compact Representation of IPv6 Addresses, publié le 1er avril 1996), jamais vue véritablement employée, sans doute une blague de 1er avril; en voici cependant un exemple:

```
# ipv6calc --addr_to_base85 3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
Itu&-ZQ82s>J%s99FJXT
```

> Info: *ipv6calc* est un programme de formatage d'adresse IPv6 et de conversion pouvant être trouvé ici: ipv6calc (miroir)

2.4. FAQ (Les bases)

2.4.1. Pourquoi IPv6 et non pas IPv5 comme successeur d'IPv4?

Dans tout en−tête IP, les 4 premiers bits sont réservés à la version du protocole. C'est ainsi qu'un numéro de protocole entre 0 et 15 est théoriquement possible:

- 4: est déjà pris pour IPv4
- 5: est réservé au protocole de flux (*Stream Protocol*, ou STP – RFC 1819 / Internet Stream Protocol Version 2) (qui n'a jamais véritablement conquis le public)

Le prochain numéro libre était 6. Et voilà comment IPv6 était né!

2.4.2. L'adresse IPv6: pourquoi un tel nombre de bits?

Lors de la conception d'IPv4, les gens pensaient que 32 bits seraient suffisants pour le monde, dans sa globalité. Rétrospectivement, 32 bits ont été jusqu'à maintenant suffisants, et seront sans doute suffisants pour encore quelques années. Cependant, 32 bits seront insuffisants à fournir dans le futur une adresse globale à chaque périphérique réseau. Pensez aux téléphones mobiles, aux voitures (incluant les périphériques électroniques sur bus CAN), aux grille−pain, aux réfrigérateurs, aux interrupteurs d'éclairage, *etc*.

Les concepteurs ont alors choisi 128 bits, 4 fois plus en longueur et une quantité 2^96 fois plus importante qu'IPv4 aujourd'hui.

La quantité utilisable est cependant inférieure à ce qu'il semble. La raison en est que, dans le schéma d'adresse défini actuellement, 64 bits sont utilisés pour l'identifiant d'interface, les 64 autres bits sont utilisés pour le routage. Compte tenu des niveaux stricts actuels d'agrégation (/48, /32, ...), il est encore possible d'"épuiser" cette quantité, mais bien heureusement, pas dans un avenir proche.

Voir aussi pour plus d'information le RFC 1715 / The H Ratio for Address Assignment Efficiency et le RFC 3194 / The Host−Density Ratio for Address Assignment Efficiency.

2.4.3. L'adresse IPv6: pourquoi un si petit nombre de bits pour sa nouvelle conception?

Pendant ce temps, il y a (c'est possible) des gens sur Internet (je n'en connais qu'un, Jim Fleming...) qui pensent déjà à IPv8, et même jusqu'à IPv16, dont les conceptions sont loin d'être couramment reçues et implémentées. En attendant, 128 bits était le meilleur choix qui pouvait être fait au regard de l'en−tête placé au−dessus des données transportées. En considérant le minimum de la taille de l'Unité Maximale de Transfert (*Maximum Transfer Uni*t, ou MTU), la longueur de l'en−tête en IPv4 est de 20 octets (c'est le minimum, car elle peut monter à 60 octets avec les options IPv4), et en IPv6, elle est de 48 octets (longueur constante). C'est 3,4 % de la MTU en IPv4 et 3,8 % de la MTU en IPv6. Cela signifie que le surplus de taille dû à l'en−tête est quasiment le même. Plus de bits dans les adresses auraient réclamé un en−tête de plus grande taille, et par conséquent, un plus grand surplus. Et si l'on prend aussi en compte la MTU maximale sur un lien courant (tel Ethernet aujourd'hui): soit 1500 octets (dans des cas particuliers: 9 Ko pour de grosses trames). Finalement, cela n'aurait pas été d'une conception correcte si 10% ou 20% des données transférées dans un paquet de la couche 3 avaient été utilisés pour les adresses et non pas pour la charge utile.

Chapitre 3. Les types d'adresse IPv6

Comme pour IPv4, l'adresse IPv6 peut être scindée en une partie réseau et une partie hôte, par l'usage d'un masque de sous−réseau.

IPv4 a montré que parfois cela serait bien si plus d'une adresse IP pouvaient être assignées à une interface, chacune à un but bien précis (alias, multi−cast). Afin de demeurer ouvert à l'avenir, IPv6 offre davantage en permettant à plus d'une adresse IPv6 d'être assignées à une interface. Il n'y a actuellement aucune limite définie par aucun RFC, mais seulement par l'implémentation de la pile IPv6 (afin de prévenir les attaques DoS).

Pour employer le grand nombre de bits constitutifs de son adresse, IPv6 définit des types d'adresse basés sur certains regroupements de ces bits, qui, avec un peu de chance, ne devraient pas être modifiés à l'avenir (à la différence d'aujourd'hui avec IPv4, et l'histoire des classes A, B et C).

C'est ainsi que la totalité des bits est divisée en une partie réseau (les 64 supérieurs) et en une partie hôte (les 64 inférieurs), afin de faciliter l'auto−configuration.

3.1. Les adresses sans préfixe spécial

3.1.1. L'adresse localhost

C'est une adresse spéciale pour l'interface de bouclage (*loopback*), similaire à IPv4 avec sa "127.0.0.1".

```
0000:0000:0000:0000:0000:0000:0000:0001
```

ou compressée:

```
::1
```

Les paquets ayant cette adresse comme source ou destination ne devraient jamais quittés l'hôte émetteur.

3.1.2. L'adresse non spécifiée

C'est une adresse spéciale telle que "n'importe laquelle" ("any") ou "0.0.0.0" en IPv4 . Il s'agit pour IPv6 de:

```
0000:0000:0000:0000:0000:0000:0000:0000
```

ou:

```
::
```

Ces adresses sont essentiellement utilisées/vues dans les sockets d'écoute (à toute adresse IPv6) ou dans les tables de routage.

Note: l'adresse non spécifiée ne peut pas être utilisée comme adresse de destination.

3.1.3. L'adresse IPv6 avec adresse IPv4 intégrée

Il y a deux types d'adresse contenant une adresse IPv4

3.1.3.1. L'adresse IPv6 mappée IPv4

Les adresses IPv6 compatibles seulement avec IPv4 sont parfois utilisées/vues pour la création de socket par un démon disposant d'IPv6, mais à l'écoute d'une adresse IPv4.

Ces adresses sont définies par un préfixe spécial d'une longueur de 96 (a.b.c.d est l'adresse IPv4):

```
0:0:0:0:0:ffff:a.b.c.d/96
```

ou en format compressé:

```
::ffff:a.b.c.d/96
```

Par exemple, l'adresse IPv4 1.2.3.4 ressemble à ceci:

```
::ffff:1.2.3.4
```

3.1.3.2. L'adresse IPv6 compatible IPv4

Utilisée pour le tunnelage automatique (RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers), en cours de remplacement par le tunnelage 6to4.

```
0:0:0:0:0:0:a.b.c.d/96
```

ou en format compressé:

```
::a.b.c.d/96
```

3.2. La partie réseau, aussi appelée préfixe

Les concepteurs ont défini certains types d'adresse et laissé un vaste champ libre à de futures définitions, telles que l'émergence de nouvelles exigences encore aujourd'hui inconnues. L'architecture d'adressage IPv6 (RFC 2373 de juillet 1998) définit le schéma d'adressage actuel, mais il y a déjà un nouveau brouillon disponible: draft−ietf−ipngwg−addr−arch−*.txt.

Jetons maintenant un coup d'oeil aux différents types de préfixe (et par conséquent aux différents types d'adresse IPv6):

3.2.1. Le type d'adresse lien−local

Ce sont des adresses particulières qui n'auront de validité que sur le lien d'une interface. En utilisant cette adresse comme adresse de destination le paquet ne devrait jamais franchir un routeur. C'est utile pour des communications sur un lien telles que:

- Y a−t−il quelqu'un d'autre sur ce lien?

- Y a−t−il quelqu'un d'autre sur ce lien ayant une adresse spéciale (on cherche par exemple à détecter la présence d'un routeur)?

Elles commencent par (où *"x"* est n'importe quel caractère hexadécimal, couramment *"0"*)

```
fe8x:  <- actuellement le seul en usage.
fe9x:
feax:
febx:
```

Ce type d'adresse se trouve sur chaque interface disposant d'IPv6 après une auto−configuration sans état (ce qui est couramment le cas).

3.2.2. Le type d'adresse site−local

Ces adresses sont similaires à ce que le RFC 1918 (RFC 1918 / Address Allocation for Private Internets) définit aujourd'hui pour IPv4, avec en plus l'avantage que celui qui utilise ce type d'adresse a la capacité d'utiliser les 16 bits fournis pour un maximum de 65536 sous−réseaux. Comparable au 10.0.0.0/8 aujourd'hui en IPv4.

Autre avantage: parce qu'il est possible avec IPv6 d'assigner plus d'une seule adresse par interface, vous pouvez assigner une telle adresse site−local en plus de l'adresse globale.

Il commence par:

```
fecx:  <- le plus couramment utilisé.
fedx:
feex:
fefx:
```

(où *"x"* est n'importe quel caractère hexadécimal, couramment *"0"*)

Notez que des discussions sont en cours concernant la dépréciation de ce type d'adresse en raison de nombreux problèmes. Pour en savoir plus, lire: draft−ietf−ipv6−deprecate−site−local−XY.txt.

Pour des tests en laboratoire, de telles adresses restent un bon choix, à mon humble avis.

3.2.3. Le type d'adresse "unicast globale (agrégeable) "

Aujourd'hui, il y a un type d'adresse globale de défini (la première conception, appelée "basée sur le fournisseur" a été abandonnée il y a déjà quelques années (RFC 1884 / IP Version 6 Addressing Architecture [obsolete]), vous en trouverez des traces dans des sources anciennes du noyau Linux).

Il commence par (les *x* étant des caractères hexadécimaux)

```
2xxx:
3xxx:
```

Note: la dénomination "agrégeable" est abandonnée dans les brouillons actuels. Il y a quelques sous−types définis en plus, ci−dessous:

3.2.3.1. Les adresses de test 6bone

Elles ont été les premières adresses globales à être définies et mises en usage. Elles commencent toutes par

```
3ffe:
```

Exemple:

```
3ffe:ffff:100:f102::1
```

Une adresse spéciale de test 6bone, qui ne sera jamais globalement unique, commence par

```
3ffe:ffff:
```

Et elle est la plupart du temps montrée dans les exemples passés, car si des adresses réelles sont montrées, il est possible à quelqu'un de les copier/coller dans ses propres fichiers de configuration. Ce type d'inadvertance cause des duplications d'adresse globalement unique. Cela pose de graves problèmes à l'hôte d'origine (par exemple recevoir des paquets en réponse de requêtes qu'il n'a pas émises). Parce qu'IPv6 est maintenant en production, ce préfixe ne sera plus délégué et probablement retiré du routage après 6 juin 2006 (voirRFC 3701 / 6bone Phaseoutpour plus d'information).

3.2.3.2. Les adresses 6to4

Ce type d'adresse, conçu pour un mécanisme précis de tunnelage (RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds et RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers), encode une adresse IPv4 donnée et un sous−réseau possible. Il commence par

```
2002:
```

Par exemple, pour représenter 92.168.1.1/5:

```
2002:c0a8:0101:5::1
```

Une petite ligne de commande peut vous aider à générer une telle adresse à partir d'une adresse IPv4 donnée:

```
ipv4="1.2.3.4"; sla="5"; printf "2002:%02x%02x:%02x%02x:%04x::1" `echo $ipv4
¬ | tr "." " "` $sla
```

Voir aussi le tunnelage utilisant 6to4 et information concernant le relayage de 6to4 par les routeurs.

3.2.3.3. Les adresses assignées par un fournisseur dans la hiérarchie de routage

Ces adresses sont déléguées aux Fournisseurs d'Accès à Internet (FAI) et commencent par

```
2001:
```

Les préfixes fournis aux FAI (aussi connus en tant que LIR) les plus importants (propriétaires de backbone) sont délégués par les centres locaux d'enregistrement (*local registries*) et ils possèdent actuellement un préfixe d'une longueur de 32.

Tout client peut obtenir de son FAI un préfixe d'une longueur de 48.

3.2.3.4. Adresses réservées aux exemples et à la documentation

Actuellement, deux blocs d'adresses sont réservés aux exemples et à la documentation:

```
3ffe:ffff::/32
2001:0DB8::/32   EXAMPLENET-WF
```

Ces blocs d'adresses devraient être filtrés sur la base des adresses source et, si possible, NE devraient PAS être acheminés par les routeurs en bordure d'Internet vers ce dernier.

3.2.4. Les adresses multicast

Les adresses multicast sont utilisées pour les services y afférents.

Elles commencent par (xx est la valeur de la portée)

```
ffxy:
```

Elles se répartissent en différentes portées et types:

3.2.4.1. La portée multicast

La portée multicast est un paramètre spécifiant la distance maximale qu'un paquet multicast peut prendre vis−à−vis de son entité émettrice.

Actuellement, les régions suivantes (portées) sont définies:

- ffx1: noeud−local, ces paquets ne quittent jamais le noeud.
- ffx2: lien−local, ces paquets ne sont jamais transmis par les routeurs, ils ne quittent par conséquent jamais le lien spécifié.
- ffx5: site−local, ces paquets ne quittent jamais le site.
- ffx8: organisation−locale, ces paquets ne quittent jamais l'organisation (pas si simple à implémenter, cela doit être par le protocole de routage).
- ffxe: portée globale.
- les autres sont réservées.

3.2.4.2. Les types multicast

Il y a déjà de nombreux types définis/réservés (voir le RFC 2373 / IP Version 6 Addressing Architecture pour les détails). Quelques exemples en sont:

- Adresse de tous les noeuds: ID = 1h, correspond aux adresses de tous les hôtes présents sur le noeud local (ff01:0:0:0:0:0:0:1) ou au lien connecté (ff02:0:0:0:0:0:0:1).
- Adresse de tous les routeurs: ID = 2h, correspond aux adresses de tous les routeurs présents sur le noeud local (ff01:0:0:0:0:0:0:2), sur le lien connecté (ff02:0:0:0:0:0:0:2), ou encore sur le site local (ff05:0:0:0:0:0:0:2).

3.2.4.3. L'adresse multicast de sollicitation du lien−local

Adresse multicast spéciale utilisée comme adresse de destination dans la découverte de voisinage, car à la différence d'IPv4, ARP n'existe plus dans IPv6.

Un exemple de cette adresse ressemble à ceci

```
ff02::1:ff00:1234
```

Le préfixe utilisé montre qu'il s'agit d'une adresse multicast lien−local. Le suffixe est généré à partir de l'adresse de destination. Dans cet exemple, un paquet devrait être envoyé à l'adresse "fe80::1234", mais la pile réseau ne connaît pas l'actuelle adresse MAC de la couche 2. Elle remplace les 104 bits supérieurs par "ff02:0:0:0:0:1:ff00::/104" et laisse les 24 bits inférieurs inchangés. Cette adresse est maintenant utilisée 'sur le lien' afin de trouver le noeud correspondant, lequel va devoir émettre une réponse contenant son adresse MAC de couche 2.

3.2.5. Les adresses anycast

Les adresses anycast sont des adresses spéciales utilisées pour couvrir des besoins tels que déterminer le serveur DNS le plus proche, le serveur DHCP le plus proche, ou tout groupe dynamique similaire. Les adresses sont prises dans l'espace d'adressage unicast (agrégeable ou site−local pour le moment). Le mécanisme anycast (au regard du client) sera pris en compte par un protocole de routage dynamique.

Note: Les adresses anycast ne peuvent être utilisées comme adresse source, elles sont utilisables uniquement comme adresse de destination.

3.2.5.1. L'adresse anycast de routeur de sous−réseau

Un simple exemple d'une adresse anycast est celle d'un routeur de sous−réseau. Soit un noeud avec l'adresse IPv6 suivante assignée:

```
3ffe:ffff:100:f101:210:a4ff:fee3:9566/64  <- L'adresse du noeud
```

L'adresse anycast de routeur de sous−réseau sera créée en laissant totalement blanc le suffixe (les 64 bits inférieurs):

```
3ffe:ffff:100:f101::/64  <- l'adresse anycast de routeur de sous-réseau
```

3.3. Les types d'adresse (partie hôte)

En ce qui concerne les questions d'auto−configuration et de mobilité, Il a été décidé d'utiliser les 64 bits inférieurs de la partie hôte de l'adresse pour la plupart des types d'adresse actuels. Conséquemment, chaque sous−réseau détient une grande quantité d'adresses.

Cette partie hôte peut être différemment considérée:

3.3.1. L'adresse calculée automatiquement (dite aussi "sans état")

Avec l'auto−configuration, la partie hôte de l'adresse est calculée en convertissant l'adresse MAC d'une interface (si disponible), avec la méthode EUI−64, en une adresse IPv6 unique. Si aucune adresse MAC n'est disponible pour le périphérique en question (ce qui arrive par exemple sur les périphériques virtuels), quelque chose d'autre (comme l'adresse IPv4 ou l'adresse MAC d'une interface physique) est utilisée à la place.

Considérons à nouveau le premier exemple:

```
3ffe:ffff:100:f101:210:a4ff:fee3:9566
```

ici,

```
210:a4ff:fee3:9566
```

est la partie hôte calculée à partir de l'adresse MAC de la NIC

```
00:10:A4:E3:95:66
```

en utilisant IEEE EUI−64 conçue pour les identifiants EUI−48.

3.3.1.1. Le problème d'incursion possible dans la sphère privée (*privacy problem*) avec les adresses automatiquement calculées, et une solution

Parce que la partie hôte "automatiquement calculée" est globalement unique (sauf lorsqu'un fabriquant de NIC utilise la même adresse MAC sur plus d'une NIC), la traque grâce à un client (*client tracking*) est possible sur l'hôte, dès lors qu'aucun proxy d'aucune sorte n'est utilisé.

C'est un problème connu, et une solution a été apportée: l'extension "sphère privée", définie dans le RFC 3041 (RFC 3041 / Privacy Extensions for Stateless Address; il y a déjà aussi un brouillon plus récent disponible: draft−ietf−ipngwg−temp−addresses−*.txt). Le principe est d'utiliser une valeur aléatoire et une valeur statique à partir desquelles un nouveau suffixe est généré à intervalle régulier. Note: ce n'est raisonnable que pour des connexions client sortantes, et n'est pas vraiment utile pour des machines réputées être des serveurs.

3.3.2. La configuration manuelle

Pour les serveurs, il est probablement plus aisé de se rappeler d'adresses plus simples; cela peut aussi se faire. Il est possible d'assigner une adresse IPv6 additionnelle à une interface, par exemple

```
3ffe:ffff:100:f101::1
```

Pour les suffixes tels que "::1", montré dans l'exemple ci−dessus, il est requis que le septième bit le plus significatif soit positionné à 0 (le bit universel/local d'un identifiant automatiquement généré). Certaines autres (à part celles qui n'ont pas étaient choisies) combinaisons de bits sont réservées aux adresses anycast.

3.4. La longueur de préfixe nécessaire au routage

Dans les premières phases de la conception, il était prévu d'utiliser une approche intégrale de routage hiérarchique, et ce, afin de réduire au maximum la taille des tables de routage. A la base du raisonnement sous−tendu par cette approche, il y a la prise en compte du nombre grandissant des entrées de routage IPv4 au coeur des routeurs (supérieur à 104 000 en mai 2001), la nécessité de réduire ce nombre afin de diminuer le

besoin en mémoire du matériel (piloté par Circuit Intégré d'Application Spécifique, *Application Specified Integrated Circuit*, ou ASIC) maintenant les tables de routage, et, en conséquence, d'accroître la vitesse (dans l'espoir que moins d'entrées génèrent des recherches plus rapides).

Aujourd'hui, le point de vue est que le routage sera conçu quasi−hiérarchiquement pour les réseaux ayant seulement un fournisseur de service. Pour plus d'une connexion à un ISP, ce n'est pas possible, et cela relève du problème de la multi−résidence (des informations sur la multi−résidence:drafts*multi6* IPv6 Multihoming Solutions

3.4.1. La longueur du préfixe (aussi connue en tant que "masque de réseau")

Comme pour IPv4, la notion de chemin de réseau routable nécessaire au routage a ici sa place. Parce que la notation standard d'un masque réseau n'est pas très agréable pour un adressage sur 128 bits, les concepteurs ont employé le schéma du Routage Inter−Domaines IPv4 Sans Classe (*IPv4 Classless Inter Domain Routing*, ou CIDR, défini dans le RFC 1519 / Classless Inter−Domain Routing), dans lequel est spécifié le nombre de bits de l'adresse devant être utilisé pour le routage. Il est aussi connu comme notation "slash".

Un exemple:

```
3ffe:ffff:100:1:2:3:4:5/48
```

De cette notation seront extraits:

- le réseau:

```
3ffe:ffff:0100:0000:0000:0000:0000:0000
```

- le masque de réseau:

```
ffff:ffff:ffff:0000:0000:0000:0000:0000
```

3.4.2. La correspondance à une route

Dans des conditions normales (*i.e.* sans QoS), de la recherche dans une table de routage résulte la route ayant le plus grand nombre de bits d'adresse significatifs; autrement dit, la route avec le plus grand préfixe correspond la première.

Par exemple, si une table de routage affiche les entrées suivantes (la liste est incomplète):

```
3ffe:ffff:100::/48     ::            U  1 0 0 sit1
2000::/3               ::192.88.99.1 UG 1 0 0 tun6to4
```

Ci−dessous, les adresses de destination des paquets IPv6 dont le trafic sera routé au travers du périphérique désigné

```
3ffe:ffff:100:1:2:3:4:5/48  ->  trafic routé au travers du périphérique sit1
3ffe:ffff:200:1:2:3:4:5/48  ->  trafic routé au travers du périphérique tun6to4
```

Chapitre 4. La vérification d'un système prêt pour IPv6

Avant de commencer à utiliser IPv6 sur votre hôte Linux, vous avez à tester si votre système est prêt pour IPv6. Pour ce faire, vous aurez peut−être d'abord un peu de travail.

4.1. Un noyau prêt pour IPv6

Les distributions contemporaines de Linux comportent déjà un noyau prêt pour IPv6, les capacités IPv6 sont en général compilées dans un module, mais il est possible que ce module ne soit pas chargé automatiquement au démarrage.

Voir la page IPv6+Linux−status−distributions pour obtenir les informations les plus à jour.

Note: vous ne devriez plus utiliser les noyaux de la série 2.2.x; car ils ne sont pas assez à jour vis−à−vis d'IPv6.

4.1.1. Vérifier la présence du support IPv6 dans le noyau actuellement en cours d'utilisation

Afin de vérifier si oui ou non votre actuel noyau supporte IPv6, jetez un coup d'oeil dans votre système de fichiers /proc. Les entrées qui suivent doivent être présentes:

```
/proc/net/if_inet6
```

Un bref test automatique ressemble à:

```
# test −f /proc/net/if_inet6 && echo "Running kernel is IPv6 ready"
```

Si cela échoue, cela peut être parce que le module IPv6 n'est pas chargé.

4.1.2. Essayer de charger le module IPv6

Vous pouvez tenter de charger le module IPv6 en exécutant

```
# modprobe ipv6
```

Si c'est un succès, la présence de ce module sera testée comme par magie par la ligne suivante:

```
# lsmod |grep −w 'ipv6' && echo "IPv6 module successfully loaded"
```

Et la vérification montrée plus haut devrait maintenant se faire elle aussi avec succès.

Note: enlever le module n'est actuellement pas supporté et peut aboutir, sous certaines conditions, au *crash* du noyau.

4.1.2.1. Le chargement automatique du module

Il est possible d'automatiser le chargement du module IPv6 à la demande. Vous avez juste à ajouter les lignes qui suivent dans le fichier de configuration du chargeur de modules du noyau (normalement /etc/modules.conf ou /etc/conf.modules):

```
alias net-pf-10 ipv6  # chargement automatique du module IPv6 à la demande
```

Il est aussi possible de mettre hors service le chargement automatique du module IPv6 en utilisant la ligne suivante

```
alias net-pf-10 off   # rend indisponible le chargement automatique du module IPv6
```

Note additionnelle: pour les noyaux de la série 2.6,le mécanisme du chargeur de modules a été repensé. Le nouveau fichier de configuration s'appellera /etc/modprobe.conf au lieu de /etc/modules.conf. Pour de plus amples détails voir le module−init−tool

- Compiler un noyau à partir des seules sources brutes (facile, si vous connaissez les options dont vous avez besoin)
- Recompiler les sources du noyau fournies par votre distribution Linux (parfois, ce n'est si simple que ça)
- Compiler un noyau avec l'extension USAGI

Si vous vous décidez à compiler un noyau, vous devriez avoir une certaine expérience dans la compilation de noyau et lire l'HOWTO sur le noyau Linux.

La comparaison pratiquement la plus à jour entre un noyau original et un noyau comprenant USAGI est disponible dans IPv6+Linux−status−kernel.

4.1.2.2. Compiler un noyau uniquement à partir des sources originales (vanille)

Plus d'éléments concernant la compilation d'un noyau disposant d'IPv6 peuvent par exemple être trouvés dans IPv6−HOWTO−2#kernel.

Note: vous devriez autant que possible utiliser les noyaux de la série 2.6.x ou supérieures,car le support IPv6 de la série 2.4.x n'obtiendra qu'un portage partielet celui de la série 2.2.x est désespérément obsolète.).

4.1.2.3. Compiler un noyau avec l'extension USAGI

Comme pour le noyau vanille, seulement recommandé aux utilisateurs avancés, déjà familiarisés avec IPv6 et la compilation noyau. Voir aussi la FAQ du projet USAGI et comment obtenir le meilleur support IPv6 avec Linux (article) (miroir).

4.1.3. Les périphériques réseau prêts pour IPv6

Les périphériques réseau n'ont pas tous déjà (ou n'auront jamais, pour certains) la capacité de transporter des paquets IPv6. L'état actuel de la situation quant à ce sujet peut être trouvé ici.

A cause de l'implémentation de la structure de la couche réseau du noyau, un problème majeur est qu'un paquet IPv6 n'est pas réellement reconnu par son numéro d'en−tête IP (6 au lieu de 4). Il est reconnu par le numéro de protocole de la couche transport 2. En conséquence, tout protocole n'utilisant pas un tel numéro de

protocole ne peut pas distribuer les paquets IPv6. Note: le paquet est bien encore transporté sur le lien, mais, côté récepteur, la distribution ne fonctionne pas (vous pouvez observer cela par exemple avec tcpdump).

4.1.3.1. Actuellement connus pour ne jamais être "capables de lien IPv6"

- IP sur Ligne Série (*Serial Line IP*, SLIP, RFC 1055 / SLIP), serait aujourd'hui mieux dénommé SLIPv4, noms de périphérique: slX
- IP sur Ligne Parallèle, comme pour SLIP, noms de périphérique: plipX
- RNIS avec encapsulation *rawip*, noms de périphérique: isdnX

4.1.3.2. Actuellement connu pour ne pas être "capable de lien IPv6"

- RNIS avec encapsulation *syncppp*, noms de périphérique: ipppX (au sujet de la conception de ipppd, il fusionnera dans une couche PPP plus abstraite dans la série des noyaux 2.5.x)

4.2. Les outils de configuration réseau prêts pour IPv6

Vous n'irez pas loin si vous faites tourner un noyau prêt pour IPv6 mais sans avoir d'outils pour configurer IPv6. Il existe plusieurs paquetages pouvant servir à cette tâche.

4.2.1. Le paquetage net−tools

Le paquetage net−tools inclut certains outils tels que ifconfig et route qui vous aideront à configurer IPv6 sur une interface. Regardez la sortie d'ifconfig? ou celle de route?, et si vous y voyez quelque chose comme IPv6, ou inet6, c'est que l'outil est prêt pour IPv6.

Vérification magique:

```
# /sbin/ifconfig −? 2>& 1 | grep −qw 'inet6' && echo "utility 'ifconfig' is
¬ IPv6−ready"
```

La même vérification peut être réalisée pour route:

```
# /sbin/route −? 2>& 1 | grep −qw 'inet6' && echo "utility 'route' is IPv6−ready"
```

4.2.2. Le paquetage iproute

Alexey N. Kuznetsov (actuellement la personne qui maintient le code réseau de Linux) a créé un jeu d'outils qui configure le réseau à travers le périphérique netlink. Vous aurez plus de fonctionnalités en utilisant ce jeu d'outils que n'en fournit net−tools, mais il n'est pas très bien documenté et n'est pas vraiment fait pour les êtres pusillanimes.

```
# /sbin/ip 2>&1 | grep −qw 'inet6' && echo "utility 'ip' is IPv6−ready"
```

Si le programme /sbin/ip n'est pas trouvé, je vous recommande alors d'installer le paquetage iproute.

- Vous pouvez le récupérer à partir de votre distribution Linux (s'il s'y trouve)
- Vous pouvez télécharger l'archive tar et recompiler: sources sur le FTP d'origine et miroir (manquant)
- Vous pouvez rechercher le bon paquetage RPM ici, RPMfind pour iproute (parfois la reconstruction d'un paquetage SRPM est recommandée)

4.3. Les programmes de test/déboguage prêts pour IPv6

Après avoir préparé votre système pour IPv6, vous voudrez établir des communications en utilisant IPv6. Vous devriez d'abord apprendre comment examiner les paquets IPv6 avec un programme dit "renifleur" (un *sniffer*). Cela est fortement conseillé, car cela peut aider à fournir très rapidement un diagnostic en cas de déboguage/dépannage.

4.3.1. ping IPv6

Ce programme est normalement inclus dans le paquetage *iputils*. Il est conçu pour réaliser de simples tests du transport en émettant des paquets de requête d'écho (*echo−request*) ICMPv6 et en attendant les paquets de réponse en écho (*echo−reply*) ICMPv6.

Usage

```
# ping6 <hôteavecadresseipv6>
# ping6 <adresseipv6>
# ping6 [-I <périphérique>] <adresseipv6-lien-local>
```

Exemple

```
# ping6 -c 1 ::1
PING ::1(::1) from ::1 : 56 data bytes
64 bytes from ::1: icmp_seq=0 hops=64 time=292 usec
--- ::1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.292/0.292/0.292/0.000 ms
```

Info: ping6 a besoin d'un accès brut à la socket, il faut donc les permissions root. Par conséquent, s'il n'y a pas d'utilisateur root pouvant utiliser ping6, deux problèmes peuvent se poser ici:

1. ping6 n'est pas sur le chemin de l'utilisateur (probablement, car ping6 est généralement stocké dans /usr/sbin −> ajouter au chemin (pas vraiment recommandé)
2. ping6 ne s'exécute pas proprement, généralement, c'est qu'il y a des permissions root manquantes −> chmod u+s /usr/sbin/ping6

4.3.1.1. Spécifier une interface à ping IPv6

En spécifiant uniquement une adresse lien−local à ping IPv6, le noyau ne sait pas par quel périphérique (physique ou virtuel) il doit émettre le paquet − chaque périphérique a une adresse lien−local. Un essai aura pour résultat un message d'erreur:

```
# ping6 fe80::212:34ff:fe12:3456
connect: Invalid argument
```

Dans ce cas vous devez en plus spécifier l'interface comme ci−dessous:

```
# ping6 -I eth0 -c 1 fe80::2e0:18ff:fe90:9205
PING fe80::212:23ff:fe12:3456(fe80::212:23ff:fe12:3456) from
¬ fe80::212:34ff:fe12:3478 eth0: 56 data bytes
64 bytes from fe80::212:23ff:fe12:3456: icmp_seq=0 hops=64 time=445 usec
--- fe80::2e0:18ff:fe90:9205 ping statistics ---
```

```
1 packets transmitted, 1 packets received, 0% packet loss round-trip
¬ min/avg/max/mdev = 0.445/0.445/0.445/0.000 ms
```

4.3.1.2. Ping6 et les adresses multicast

Un mécanisme intéressant pour détecter les hôtes IPv6 actifs sur un lien est de lancer ping6 sur l'adresse multicast lien−local tous−noeuds (*all−node*):

```
# ping6 -I eth0 ff02::1
PING ff02::1(ff02::1) from fe80:::2ab:cdff:feef:012356 eth0: 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.104 ms
64 bytes from fe80::212:34ff:fe12:3450: icmp_seq=1 ttl=64 time=0.549 ms (DUP!)
```

A la différence d'IPv4, où les réponses à un ping sur l'adresse de diffusion (*broadcast*) peuvent être rendues indisponibles, en IPv6, ce comportement ne peut pas être actuellement rendu indisponible, sauf par un pare−feu IPv6 local.

4.3.2. traceroute6 IPv6

Ce programme est normalement inclus dans le paquetage *iputils*. C'est un programme similaire au traceroute d'IPv4. En voici un exemple:

```
# traceroute6 www.6bone.net
traceroute to 6bone.net (3ffe:b00:c18:1::10) from 3ffe:ffff:0000:f101::2, 30
¬ hops max, 16 byte packets
 1 localipv6gateway (3ffe:ffff:0000:f101::1) 1.354 ms 1.566 ms 0.407 ms
 2 swi6T1-T0.ipv6.switch.ch (3ffe:2000:0:400::1) 90.431 ms 91.956 ms 92.377 ms
 3 3ffe:2000:0:1::132 (3ffe:2000:0:1::132) 118.945 ms 107.982 ms 114.557 ms
 4 3ffe:c00:8023:2b::2 (3ffe:c00:8023:2b::2) 968.468 ms 993.392 ms 973.441 ms
 5 3ffe:2e00:e:c::3 (3ffe:2e00:e:c::3) 507.784 ms 505.549 ms 508.928 ms
 6 www.6bone.net (3ffe:b00:c18:1::10) 1265.85 ms * 1304.74 ms
```

Note: à la différence de certaines versions contemporaines du traceroute d'IPv4, qui peuvent utiliser les paquets de requête d'écho ICMPv4 aussi bien que les paquets UDP (défaut), l'actuel traceroute IPv6 ne peut qu'émettre des paquets UDP. Comme vous le savez peut−être, les paquets de requête d'écho ICMP sont mieux acceptés par les pare−feu ou les ACL sur les routeurs intermédiaires que les paquets UDP.

4.3.3. tracepath6 IPv6

Ce programme est normalement inclus dans le paquetage *iputils*. C'est un programme comme traceroute6, il trace le chemin vers une destination donnée, découvrant la MTU le long de ce chemin. En voici un exemple:

```
# tracepath6 www.6bone.net
 1?: [LOCALHOST] pmtu 1480
 1: 3ffe:401::2c0:33ff:fe02:14 150.705ms
 2: 3ffe:b00:c18::5 267.864ms
 3: 3ffe:b00:c18::5 asymm 2 266.145ms pmtu 1280
 3: 3ffe:3900:5::2 asymm 4 346.632ms
 4: 3ffe:28ff:ffff:4::3 asymm 5 365.965ms
 5: 3ffe:1cff:0:ee::2 asymm 4 534.704ms
 6: 3ffe:3800::1:1 asymm 4 578.126ms !N
Resume: pmtu 1280
```

4.3.4. tcpdump IPv6

Sur Linux, tcpdump est l'outil majeur pour la capture de paquets. Vous allez trouver ci−dessous quelques exemples. Le support IPv6 est normalement intégré aux éditions actuelles de la version 3.6.

tcpdump utilise des expressions pour filtrer les paquets, minimisant le bruit:

- icmp6: filtre le trafic ICMPv6 natif
- ip6: filtre le trafic IPv6 natif (incluant ICMPv6)
- proto ipv6: filtre le trafic IPv6−in−IPv4 tunnelé
- not port ssh: supprime l'affichage des paquets SSH, pour lancer tcpdump à partir d'une session distante SSH

Certaines options en ligne de commande sont très utiles pour capter et afficher plus d'information concernant les paquets, essentiellement intéressant pour approfondir l'information des paquets ICMPv6:

- "−s 512": augmente la quantité d'information capturée pour un paquet à 512 octets
- "−vv": sortie vraiment verbeuse
- "−n": ne pas résoudre les adresses en noms, utile si la résolution inversée ne fonctionne pas proprement

4.3.4.1. Ping IPv6 vers l'adresse native 3ffe:ffff:100:f101::1 sur un lien−local

```
# tcpdump −t −n −i eth0 −s 512 −vv ip6 or proto ipv6
tcpdump: listening on eth0
3ffe:ffff:100:f101:2e0:18ff:fe90:9205 > 3ffe:ffff:100:f101::1: icmp6: echo
¬ request (len 64, hlim 64)
3ffe:ffff:100:f101::1 > 3ffe:ffff:100:f101:2e0:18ff:fe90:9205: icmp6: echo
¬ reply (len 64, hlim 64)
```

4.3.4.2. Ping IPv6 vers 3ffe:ffff:100::1 routée au travers d'un tunnel IPv6−in−IPv4

1.2.3.4 et 5.6.7.8 sont les extrémités du tunnel (toutes les adresses sont des exemples)

```
# tcpdump −t −n −i ppp0 −s 512 −vv ip6 or proto ipv6
tcpdump: listening on ppp0
1.2.3.4 > 5.6.7.8: 2002:ffff:f5f8::1 > 3ffe:ffff:100::1: icmp6: echo request
¬ (len 64, hlim 64) (DF) (ttl 64, id 0, len 124)
5.6.7.8 > 1.2.3.4: 3ffe:ffff:100::1 > 2002:ffff:f5f8::1: icmp6: echo reply (len
¬ 64, hlim 61) (ttl 23, id 29887, len 124)
1.2.3.4 > 5.6.7.8: 2002:ffff:f5f8::1 > 3ffe:ffff:100::1: icmp6: echo request
¬ (len 64, hlim 64) (DF) (ttl 64, id 0, len 124)
5.6.7.8 > 1.2.3.4: 3ffe:ffff:100::1 > 2002:ffff:f5f8::1: icmp6: echo reply (len
¬ 64, hlim 61) (ttl 23, id 29919, len 124)
```

4.4. Les programmes prêts pour IPv6

Les distributions actuelles comportent déjà les clients et les serveurs IPv6 les plus couramment utilisés. Allez d'abord voir sur IPv6 & Linux / l'état actuel des distributions. Si ce que vous cherchez n'y est pas encore, vous pouvez vérifier sur IPv6 & Linux / l'état actuel des applications disponibles, où sont répertoriés les programmes déjà portés sur IPv6 et utilisables sous Linux. Pour les programmes les plus communément

employés, il y a quelques éléments disponibles dans la troisième partie et la quatrième partie de l'HowTo – IPv6 & Linux.

4.5. Les programmes client prêts pour IPv6 (une sélection)

Pour lancer les tests qui vont suivre, il est nécessaire que votre système dispose d'IPv6, et certains exemples montrent des adresses ne pouvant être atteintes que si une connexion au 6bone est disponible.

4.5.1. Vérifier la résolution DNS des adresses IPv6

A cause des mises à jour de sécurité ces dernières années, tout serveur du Système des Noms de Domaine (DNS) devrait fonctionner avec un logiciel récent comprenant déjà le type (intermédiaire) d'adresse IPv6 AAAA (le nouveau, nommé A6 n'est pas encore assez répandu pour le moment, car uniquement supporté par BIND9 et supérieurs, mais aussi à cause de la non existence de support du domaine racine IP6.ARPA). Un simple test pour savoir si le système utilisé peut résoudre les adresses IPv6 est

```
# host -t AAAA www.join.uni-muenster.de
```

et cela devrait affiché quelque chose comme ce qui suit:

```
www.join.uni-muenster.de. is an alias for tolot.join.uni-muenster.de.
tolot.join.uni-muenster.de. has AAAA address 2001:638:500:101:2e0:81ff:fe24:37c6
```

4.5.2. Le client telnet prêt pour IPv6

Des clients telnet prêts pour IPv6 sont disponibles. Un simple test peut être effectué par

```
$ telnet 3ffe:400:100::1 80
Trying 3ffe:400:100::1...
Connected to 3ffe:400:100::1.
Escape character is '^]'.
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Sun, 16 Dec 2001 16:07:21
GMT Server: Apache/2.0.28 (Unix)
Last-Modified: Wed, 01 Aug 2001 21:34:42 GMT
ETag: "3f02-a4d-b1b3e080"
Accept-Ranges: bytes
Content-Length: 2637
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Connection closed by foreign host.
```

Si le client telnet ne comprend pas l'adresse IPv6 et dit quelque chose comme "ne peut résoudre le nom d'hôte" ("*cannot resolve hostname*"), IPv6 n'est alors pas disponible.

4.5.3. Les clients ssh prêts pour IPv6

4.5.3.1. openssh

Les versions actuelles d'openssh sont prêtes pour IPv6. Selon la configuration précédant la compilation, il y a deux comportements possibles.

- −−without−ipv4−default: le client essaie automatiquement une connexion IPv6 en premier et revient à IPv4 en cas d'échec.
- −−with−ipv4−default: la connexion par défaut est IPv4, la connexion IPv6 doit être forcée comme dans l'exemple qui suit:

```
$ ssh -6 ::1
user@::1's password: ******
[user@ipv6host user]$
```

Si votre client ssh ne comprend pas l'option "−6", c'est qu'il n'a pas IPv6 de disponible, comme la plupart des paquetages de ssh version 1.

4.5.3.2. ssh.com

Le client et le serveur SSH de chez SSH.com sont aussi prêts pour IPv6, et gratuits pour les machines Linux et FreeBSD selon l'usage – commercial ou personnel – qui en est fait.

4.5.4. Les navigateurs web prêts pour IPv6

L'état actuel de la liste des navigateurs web IPv6 est disponible.

La plupart ont des problèmes irrésolues pour le moment

1. Si un seul proxy IPv4 est utilisé dans les réglages, les requêtes IPv6 seront bien envoyées vers le proxy, mais celui−ci échouera à comprendre la requête, laquelle échouera. Solution: mettre à jour le logiciel proxy (à voir plus tard).
2. Les réglages de configuration automatique de proxy (*.pac) ne peuvent être étendus afin de prendre en charge différemment les requêtes IPv6 (par exemple ne pas utiliser le proxy) à cause de leur nature (écrits en Java−script et bien encodés en dur dans les sources, comme cela peut être observé pour le code source de Maxilla).

C'est ainsi que les anciennes versions ne comprennent pas un URL avec une adresse encodée en IPv6 comme http://[3ffe:400:100::1]/ (cet URL ne fonctionne qu'avec un navigateur disposant d'IPv6!).

Un bref test est d'essayer l'URL fourni avec un navigateur donné, sans utiliser de proxy.

4.5.4.1. Un URL de test

Un bon point de départ pour tester la navigation IPv6 est http://www.kame.net/. Si la tortue sur la page est animée, la connexion se fait *via* IPv6, sinon la tortue est statique.

4.6. Les programmes serveur prêts pour IPv6

Dans cette partie, de nombreuses questions concernant des clients spécifiques ont été mentionnées. En conséquence, les éléments pour les serveurs prêts pour IPv6 sont fournis plus bas dans la section Eléments d'installation des démons prêts pour IPv6.

4.7. FAQ (vérification d'un système prêt pour IPv6)

4.7.1. Utiliser les outils

4.7.1.1. Q: impossible d'utiliser ping6 avec des adresses lien−local

Message d'erreur: "*connect: Invalid argument*"

Le noyau ne sait pas sur quel lien (physique ou virtuel) vous voulez l'utiliser et envoyer des paquets ICMPv6. C'est pourquoi est affiché un message d'erreur.

Solution: spécifier l'interface de cette façon: "ping6 −I eth0 fe80::2e0:18ff:fe90:9205", voir aussi l'usage du programme ping6.

4.7.1.2. Q: impossible d'utiliser ping6 ou traceroute en tant qu'utilisateur courant

Message d'erreur: "*icmp socket: Operation not permitted*"

Ces utilitaires créent des paquets spéciaux ICMPv6 et les émettent en dehors. Ceci est réalisé par l'emploi des sockets brutes du noyau. Ces dernières ne peuvent être utilisées que par l'utilisateur "root". C'est pourquoi les utilisateurs courants obtiennent un tel message d'erreur.

Solution: s'il est vraiment nécessaire que tous les utilisateurs puissent utiliser ces utilitaires, vous pouvez ajouter le bit "suid" en faisant "chmod u+s /chemin/vers/le/programme", voir aussi l'usage du programme. Si tous les utilisateurs ne doivent pas en être capables, vous pouvez changer ce programme de groupe, par exemple au profit du groupe "wheel", ajouter les utilisateurs nécessaires à ce groupe et ôter le bit d'exécution aux autres utilisateurs par "chmod o−rwx /chemin/vers/le/programme", ou bien configurer "sudo" pour mettre en place votre politique de sécurité.

Chapitre 5. Configurer les interfaces

5.1. Les différents périphériques réseau

Sur un noeud, il existe différents périphériques réseau. Ils peuvent être

- Physiquement rattachés, comme eth0, tr0
- Virtuellement existants, comme ppp0, tun0, tap0, sit0, isdn0, ippp0

5.1.1. Physiquement rattachés

Les interfaces physiquement rattachées, comme Ethernet ou Token−Ring, sont la norme et n'ont pas besoin d'un traitement particulier.

5.1.2. Virtuellement existants

Les interfaces virtuellement rattachées ont toujours besoin d'un traitement particulier.

5.1.2.1. Les interfaces de tunnelage IPv6−in−IPv4

Ces interfaces sont normalement dénommées sit*x*. *sit* est l'abréviation mise pour Simple Transition Internet (*Simple Internet Transition*). Ce périphérique a la capacité d'encapsuler les paquets IPv6 à l'intérieur de paquets IPv4 et de les tunneler vers une extrémité étrangère.

sit0 a une signification particulière et ne peut être utilisée pour des tunnels dédiés.

5.1.2.2. Les interfaces PPP

Les interfaces PPP acquièrent leur capacité IPv6 grâce à un démon PPP disposant d'IPv6.

5.1.2.3. Les interfaces RNIS HDLC

La capacité IPv6 pour HDLC avec encapsulation ip est déjà intégrée au noyau.

5.1.2.4. Les interfaces PPP RNIS

Les interfaces PPP RNIS (ippp) ne sont pas disponibles pour IPv6 dans le noyau. Il n'est pas prévu que cela se fasse, puisqu'elles seront remplacées par une couche d'interface ppp plus générique.

5.1.2.5. SLIP + PLIP

Comme il a déjà été dit, ces interfaces ne supportent pas le transport IPv6 (l'émission est OK, mais la distribution à la réception ne fonctionne pas).

5.1.2.6. Le périphérique Ether−tap

Les périphériques Ether−tap sont prêts pour IPv6, et sont de plus configurables sans état. Pour être utilisés, le module "ethertap" doit être chargé au préalable.

5.1.2.7. Les périphériques tun

Actuellement, je ne les ai pas encore testés par moi−même.

5.1.2.8. ATM

01/2002: non supporté par l'actuel noyau vanille, supporté par l'extension USAGI.

5.1.2.9. Autres

Ai−je oublié une interface?...

5.2. (dé)Montage des interfaces

Deux méthodes peuvent être utilisées pour (dé)monter les interfaces.

5.2.1. Utiliser "ip"

Usage:

```
# ip link set dev <interface> up
# ip link set dev <interface> down
```

Exemple:

```
# ip link set dev eth0 up
# ip link set dev eth0 down
```

5.2.2. Utiliser "ifconfig"

Usage:

```
# /sbin/ifconfig <interface> up
# /sbin/ifconfig <interface> down
```

Exemple:

```
# /sbin/ifconfig eth0 up
# /sbin/ifconfig eth0 down
```

Chapitre 6. Configurer les adresses IPv6

Il y a différentes façons de configurer une adresse IPv6 sur une interface. Vous pouvez utiliser "ifconfig" ou "ip".

6.1. Affichage des adresses IPv6 existantes

Vous devriez d'abord vérifier s'il existe des adresses IPv6 configurées, et combien (peut−être y en a−t−il qui l'ont été, comme par magie, pendant l'auto−configuration sans état).

6.1.1. Utiliser "ip"

Usage:

```
# /sbin/ip −6 addr show dev <interface>
```

Exemple pour un hôte configuré statiquement:

```
# /sbin/ip −6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP&gt; mtu 1500 qdisc pfifo_ fast qlen 100
inet6 fe80::210:a4ff:fee3:9566/10 scope link
inet6 3ffe:ffff:0:f101::1/64 scope global
inet6 fec0:0:0:f101::1/64 scope site
```

Exemple pour un hôte auto−configuré

Ici vous pouvez voir des adresses auto−configurées comme par magie et leurs durées de vie.

```
# /sbin/ip −6 addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,PROMISC,UP&gt; mtu 1500 qdisc pfifo_fast qlen
¬ 100
inet6 2002:d950:f5f8:f101:2e0:18ff:fe90:9205/64 scope global dynamic
valid_lft 16sec preferred_lft 6sec
inet6 3ffe:400:100:f101:2e0:18ff:fe90:9205/64 scope global dynamic
valid_lft 2591997sec preferred_lft 604797sec inet6 fe80::2e0:18ff:fe90:9205/10
¬ scope link
```

6.1.2. Utiliser "ifconfig"

Usage:

```
# /sbin/ifconfig <interface>
```

Exemple (la sortie est filtrée avec grep pour n'afficher que les adresses IPv6). vous pouvez voir ici des adresses IPv6 ayant des portées différentes.

```
# /sbin/ifconfig eth0 |grep "inet6 addr:"
inet6 addr: fe80::210:a4ff:fee3:9566/10 Scope:Link
inet6 addr: 3ffe:ffff:0:f101::1/64 Scope:Global
inet6 addr: fec0:0:0:f101::1/64 Scope:Site
```

6.2. Ajouter une adresse IPv6

Ajouter une adresse IPv6 est similaire au mécanisme des adresses "ALIAS IP" sur les interfaces configurées par IPv4 Linux.

6.2.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 addr add <adresseipv6>/<longueurdupréfixe> dev <interface>
```

Exemple:

```
# /sbin/ip -6 addr add 3ffe:ffff:0:f101::1/64 dev eth0
```

6.2.2. Utiliser "ifconfig"

Usage:

```
# /sbin/ifconfig <interface> inet6 add <adresseipv6>/<longueurdupréfixe>
```

Exemple:

```
# /sbin/ifconfig eth0 inet6 add 3ffe:ffff:0:f101::1/64
```

6.3. Ôter une adresse IPv6

Rarement nécessaire, prenez garde de ne pas ôter une adresse IPv6 n'existant pas, il en résulte parfois un *crash* sur les anciens noyaux.

6.3.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 addr del <adresseipv6ipv6address>/<longueurdupréfixe> dev <interface>
```

Exemple:

```
# /sbin/ip -6 addr del 3ffe:ffff:0:f101::1/64 dev eth0
```

6.3.2. Utiliser "ifconfig"

Usage:

```
# /sbin/ifconfig <interface> inet6 del <adresseipv6>/<longueurdupréfixe>
```

Exemple:

```
# /sbin/ifconfig eth0 inet6 del 3ffe:ffff:0:f101::1/64
```

Chapitre 7. Configurer les routes IPv6 courantes

Si vous voulez quitter votre lien et voulez émettre des paquets vers l'Internet mondial IPv6, vous avez besoin de routage. S'il existe déjà un routeur disposant d'IPv6 sur votre lien, il est possible que cela soit suffisant pour ajouter des routes IPv6.

7.1. Afficher les routes IPv6 existantes

Vous devriez d'abord vérifier s'il existe des routes IPv6 configurées, et combien (peut–être y en a–t–il qui l'ont été, comme par magie, pendant l'auto–configuration sans état).

7.1.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 route show [dev <périphérique>]
```

Exemple:

```
# /sbin/ip -6 route show dev eth0
3ffe:ffff:0:f101::/64 proto kernel metric 256 mtu 1500 advmss 1440
fe80::/10             proto kernel metric 256 mtu 1500 advmss 1440
ff00::/8              proto kernel metric 256 mtu 1500 advmss 1440
default               proto kernel metric 256 mtu 1500 advmss 1440
```

7.1.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6
```

Exemple (la sortie est filtrée sur l'interface eth0). Ici vous pouvez voir différentes routes IPv6 pour différentes adresses sur une même interface.

```
# /sbin/route -A inet6 |grep -w "eth0"
3ffe:ffff:0:f101 ::/64 :: UA  256 0 0 eth0 <- Route de l'interface de portée globale
¬ address
fe80::/10        ::       UA  256 0 0 eth0 <- Route de l'interface de portée lien-local
¬ address
ff00::/8         ::       UA  256 0 0 eth0 <- Route de l'interface destiné à tout le trafic multi
¬ addresses
::/0             ::       UDA 256 0 0 eth0 <- Route automatique par défaut
```

7.2. Ajouter une route IPv6 traversant une passerelle

Nécessaire la plupart du temps pour atteindre l'extérieur grâce à IPv6 en utilisant un routeur IPv6 sur votre lien.

7.2.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 route add <réseauipv6>/<longueurdupréfixe> via <adresseipv6>
¬ [dev <périphérique>]
```

Exemple:

```
# /sbin/ip -6 route add 2000::/3 via 3ffe:ffff:0:f101::1
```

7.2.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6 add <réseauipv6>/<longueurdupréfixe> gw
¬ <adresseipv6> [dev <périphérique>]
```

Un périphérique peut être nécessaire également, si l'adresse IPv6 de la passerelle est un lien−local.

Suivre l'exemple montré ajoute une route à toutes les adresses globales actuelles (2000::/3) à travers la passerelle 3ffe:ffff:0:f101::1

```
# /sbin/route -A inet6 add 2000::/3 gw 3ffe:ffff:0:f101::1
```

7.3. Ôter une route IPv6 traversant une passerelle

Rarement nécessaire manuellement, la plupart du temps effectué par les scripts configurant le réseau à l'extinction (totale ou par interface)

7.3.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 route del <réseauipv6>/<longueurdupréfixe> via <ipv6address>
¬ [dev <périphérique>]
```

Exemple:

```
# /sbin/ip -6 route del 2000::/3 via 3ffe:ffff:0:f101::1
```

7.3.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6 del <réseau>/<longueurdupréfixe> [dev <périphérique>]
```

Exemple pour de nouveau ôter la route précédemment ajoutée:

```
# /sbin/route -A inet6 del 2000::/3 gw 3ffe:ffff:0:f101::1
```

7.4. Ajouter une route IPv6 traversant une interface

Pas si fréquent, parfois en cas de création de lien point–à–point.

7.4.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 route add <réseauipv6>/<longueurdupréfixe> dev <périphérique>
¬ metric 1
```

Exemple:

```
# /sbin/ip -6 route add 2000::/3 dev eth0 metric 1
```

La distance (*metric*) "1" est utilisée ici par soucis de compatibilité avec la distance utilisée par route, car la distance par défaut fixée par "ip" est "1024".

7.4.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6 add <réseau>/<longueurdupréfixe> dev <périphérique>
```

Exemple:

```
# /sbin/route -A inet6 add 2000::/3 dev eth0
```

7.5. Ôter une route IPv6 traversant une interface

Rarement utiliser manuellement, les scripts de configuration font cela à l'extinction.

7.5.1. Utiliser "ip"

Usage:

```
# /sbin/ip -6 route del <réseauipv6>/<longueurdupréfixe> dev <périphérique>
```

Exemple:

```
# /sbin/ip -6 route del 2000::/3 dev eth0
```

7.5.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6 del <réseau>/<longueurdupréfixe> dev <périphérique>
```

Exemple:

```
# /sbin/route -A inet6 del 2000::/3 dev eth0
```

7.6. FAQ concernant les routes IPv6

7.6.1. Support d'une route par défaut IPv6

Une idée d'IPv6 était le routage hiérarchique, avec pour conséquence une quantité moindre d'entrées dans les tables de routage nécessaires aux routeurs.

Il y a certains problèmes dans les noyaux Linux actuels:

7.6.1.1. Les clients (ne routent aucun paquet!)

Les clients peuvent installer une route par défaut avec pour préfixe "::/0", ils peuvent aussi apprendre une telle route par auto−configuration, en utilisant par exemple radvd s'il est présent sur le lien, comme le montre ce qui suit:

```
# ip -6 route show | grep ^default
default via fe80::212:34ff:fe12:3450 dev eth0 proto kernel metric 1024 expires
¬ 29sec mtu 1500 advmss 1440
```

7.6.1.2. Les routeurs en cas de renvoi de paquets

Dans ses grandes lignes, l'actuel noyau Linux (au moins <= 2.4.17) ne supporte pas les routes par défaut. Vous pouvez les installées, mais la recherche échouera quand un paquet devra être renvoyé (une intention normale pour un routeur).

Pour l'heure, le "routage par défaut" peut être installé en utilisant l'actuel et unique préfixe d'adresse globale "2000::/3".

Le projet USAGI supporte déjà cela dans leurs extensions grâce à une astuce de programmation (*NdT*: *a hack*, *i.e.* littéralement, une"bidouille").

Note: prenez garde au routage par défaut sans filtrage d'adresse sur les routeurs de bordure, sinon du trafic multicast ou site−local quittera l'environnement.

Chapitre 8. La découverte de voisinage

La découverte de voisinage est le successeur IPv6 de ARP (*Address Resolution Protocol*, protocole de résolution d'adresse) pour IPv4. Vous pouvez récupérer l'information concernant le voisinage actuel, de plus, vous pouvez fixer ou détruire des entrées. Le noyau garde la trace de la détection d'un voisin (comme ARP pour IPv4). Vous pouvez faire des recherches dans la table apprise, en utilisant "ip".

8.1. Afficher le voisinage en utilisant "ip"

Avec la commande qui suit vous pouvez afficher les voisins IPv6 appris ou configurés

```
# ip -6 neigh show [dev <périphérique>]
```

L'exemple suivant montre un voisin, qui est un routeur pouvant être atteint

```
# ip -6 neigh show
fe80::201:23ff:fe45:6789 dev eth0 lladdr 00:01:23:45:67:89 router nud reachable
```

8.2. Manipuler la table de voisinage en utilisant "ip"

8.2.1. Ajouter manuellement une entrée

La commande suivante vous permet d'ajouter manuellement une entrée

```
# ip -6 neigh add <adresseIPv6> lladdr <adressedelacouche-lien> dev <périphérique>
```

Exemple:

```
# ip -6 neigh add fec0::1 lladdr 02:01:02:03:04:05 dev eth0
```

8.2.2. Détruire manuellement une entrée

De même qu'une entrée peut être ajoutée, une entrée peut être détruite:

```
# ip -6 neigh del <adresseipv6> lladdr <adressedelacouche-lien> dev <périphérique>
```

Exemple:

```
# ip -6 neigh del fec0::1 lladdr 02:01:02:03:04:05 dev eth0
```

8.2.3. Pour plus de réglages avancés

L'outil "ip" est sous–documenté, mais il est très puissant. Voir l'aide en ligne pour en savoir plus

```
# ip -6 neigh help
Usage: ip neigh { add | del | change | replace } { ADDR [ lladdr LLADDR ]
          [ nud { permanent | noarp | stale | reachable } ]
```

```
          | proxy ADDR } [ dev DEV ]
      ip neigh {show|flush} [ to PREFIX ] [ dev DEV ] [ nud STATE ]
```

Il semble que certaines options soient uniquement pour IPv4... si vous pouvez contribuer à en dire plus sur les drapeaux et l'emploi avancé, merci d'envoyer vos informations.

Chapitre 9. Configurer les tunnels IPv6–in–IPv4

Si vous souhaitez quitter votre lien incapable d'accéder à IPv6 à partir de votre réseau local, vous avez besoin d'un tunnelage IPv6–in–IPv4 afin de rejoindre l'Internet mondial IPv6.

Il y a différents mécanismes de tunnelage, et conséquemment, différentes façons d'installer des tunnels.

9.1. Les types de tunnel

Il y a plus d'une façon de tunneler des paquets IPv6 sur des liens uniquement IPv4.

9.1.1. Tunnelage statique point–à–point: 6bone

Un tunnel point–à–point est un tunnel dédié à un point de connexion terminal, qui connaît votre réseau IPv6 (pour le routage en retour) et l'adresse IPv4 de votre point de connexion (terminale), comme défini dans la RFC 2893 / Transition Mechanisms for IPv6 Hosts and Routers. Pré–requis:

- L'adresse IPv4 de votre point de connexion terminal doit être globalement unique, statique, et accessible à partir de l'autre point de connexion terminal distant
- Un préfixe IPv6 vous est assigné (voir le bureau d'enregistrement 6bone)
- Une extrémité distante du tunnel capable de router votre préfixe IPv6 jusqu'à votre extrémité locale du tunnel (la plupart du temps, une configuration manuelle distante est requise)

9.1.2. Le tunnelage automatique

Le cas du tunnelage automatique se présente quand un noeud se connecte directement à un autre noeud en ayant obtenu au préalable l'adresse IPv4 de l'autre noeud.

9.1.3. Le tunnelage 6to4

Le tunnelage 6to4 (RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds) utilise un mécanisme simple pour créer des tunnels automatiques. Tout noeud ayant une adresse unique globale IPv4 est capable d'être le point de connexion terminal d'un tunnel 6to4 (si aucun pare–feu IPv4 ne prohibe ce trafic). Foncièrement, le tunnelage 6to4 n'est pas un tunnel en binôme (*one–to–one tunnel*). Ce tunnelage se subdivise en un tunnelage d'un flux montant et d'un flux descendant. Une adresse IPv6 spéciale indique que ce noeud utilisera un tunnelage 6to4 pour se connnecter au réseau mondial IPv6.

9.1.3.1. La génération d'un préfixe 6to4

Une adresse 6to4 est définie comme suit (le schéma provient du RFC 3056 / Connection of IPv6 Domains via IPv4 Clouds):

```
|   3+13   |    32     |    16  |            64 bits             |
+---+------+-----------+--------+--------------------------------+
|  FP+TLA  |  V4ADDR   | SLA ID |           Interface ID         |
|  0x2002  |           |        |                                |
+---+------+-----------+--------+--------------------------------+
```

FP et TLA ensemble (16 bits) ont la valeur 0x2002. V4ADDR est l'adresse IPv4 globale et unique du noeud (en notation hexadécimale). SLA est l'identifiant de sous–réseau (65536 sous–réseaux locaux possibles). Ils

sont utilisés pour représenter la structure locale de votre réseau.

Pour les passerelles, un tel préfixe est généré en utilisant normalement pour SLA "0000", et pour suffixe "::1" (ce n'est pas une nécessité, il peut être déterminé arbitrairement, mais d'une portée locale) et assigné à l'interface de tunnelage 6to4. Notez que Windows Microsoft utilise aussi V4ADDR comme préfixe.

9.1.3.2. Le flux de tunnelage ascendant 6to4

Le noeud doit savoir à quel point de connexion terminal étranger ses paquets IPv6 dans IPv4 doivent être envoyés. Aux tout premiers jours du tunnelage 6to4, des routeurs dédiés au tunnelage de flux ascendant avaient été définis. Voir l'information 6to4 de NSayer pour une liste de ses routeurs.

De nos jours, les routeurs de flux ascendant 6to4 peuvent être découverts comme par magie par l'emploi de l'adresse anycast 192.88.99.1. Les protocoles de routage s'occupent de cela en arrière−plan, voir le RFC 3068 / An Anycast Prefix for 6to4 Relay Routers pour les détails.

9.1.3.3. Le flux de tunnelage descendant 6to4

La méthode servant au flux descendant (du 6bone vers votre noeud disposant de 6to4) n'est pas vraiment bien fixée et peut varier selon l'hôte étranger vers qui sont envoyés les paquets originaux. Il existe deux possibilités:

- l'hôte étranger utilise 6to4 et émet directement en retour les paquets à votre noeud (voir plus bas)
- l'hôte étranger émet les paquets en retour vers le réseau mondial IPv6 et selon le routage dynamique qui a lieu alors, un routeur relais créera un tunnel automatique de retour vers votre noeud.

9.1.3.4. Le trafic possible avec 6to4

- de 6to4 vers 6to4: est normalement tunnelé directement entre chacun des hôtes disposant de 6to4
- de 6to4 vers un trafic non 6to4: est émis *via* le flux ascendant du tunnelage
- un trafic non 6to4 vers 6to4: est émis *via* le flux descendant du tunnelage

9.2. Afficher les tunnels existants

9.2.1. Utiliser "ip"

Usage:

```
# /sbin/ip −6 tunnel show [<périphérique>]
```

Exemple:

```
# /sbin/ip −6 tunnel show
sit0: ipv6/ip remote any local any ttl 64 nopmtudisc
sit1: ipv6/ip remote 195.226.187.50 local any ttl 64
```

9.2.2. Utiliser "route"

Usage:

```
# /sbin/route -A inet6
```

Exemple (la sortie est filtrée afin de ne laisser apparaître que les tunnels empreintant l'interface sit0):

```
# /sbin/route -A inet6 | grep "\Wsit0\W*$"
::/96      ::               U   256  2  0  sit0
2002::/16  ::               UA  256  0  0  sit0
2000::/3   ::193.113.58.75  UG    1  0  0  sit0
fe80::/10  ::               UA  256  0  0  sit0
ff00::/8   ::               UA  256  0  0  sit0
```

9.3. Montage d'un tunnel point–à–point

Il y a 3 possibilités pour ajouter ou ôter un tunnel point–à–point.

Une bonne source d'information additionnelle à propos de l'installation de tunnel grâce à "ip" est configurer les tunnels avec iproute2 (article) (miroir).

9.3.1. Ajouter un tunnel point–à–point

9.3.1.1. Utiliser "ip"

La méthode la plus commune actuellement pour une petite quantité de tunnels.

Usage en vue de créer un périphérique de tunnelage (mais il n'est pas monté pour autant, une TTL doit également être spécifiée, car la valeur par défaut est 0)

```
# /sbin/ip tunnel add <périphérique> mode sit ttl <ttlpardéfaut> remote
¬ <adresseipv4dutunnelétranger> local <adresseipv4locale>
```

Usage (exemple générique pour trois tunnels):

```
# /sbin/ip tunnel add sit1 mode sit ttl <ttlpardéfaut> remote
¬ <adresseipv4dutunnelétranger1> local <adresseipv4locale>
# /sbin/ip link set dev sit1 up
# /sbin/ip -6 route add <préfixepourlaroute1> dev sit1 metric 1
# /sbin/ip tunnel add sit2 mode sit ttl <ttlpardéfaut>
¬ <adresseipv4dutunnelétranger2> local <adresseipv4locale>
# /sbin/ip link set dev sit2 up
# /sbin/ip -6 route add <préfixepourlaroute2> dev sit2 metric 1
# /sbin/ip tunnel add sit3 mode sit ttl <ttlpardéfaut>
¬ <adresseipv4dutunnelétranger3> local <adresseipv4locale>
# /sbin/ip link set dev sit3 up
# /sbin/ip -6 route add <préfixepourlaroute3> dev sit3 metric 1
```

9.3.1.2. Utiliser "ifconfig" et "route" (méthode dépréciée)

Ce n'est véritablement pas une méthode recommandée pour ajouter un tunnel, car elle est plutôt étrange. Pas de problème lors de l'ajout d'un seul tunnel, mais si vous en montez plus d'un, il ne vous est pas possible facilement de démonter le premier tout en laissant les autres fonctionner.

Usage (exemple générique pour trois tunnels):

```
# /sbin/ifconfig sit0 up
# /sbin/ifconfig sit0 tunnel <adresseipv4dutunnelétranger1>
# /sbin/ifconfig sit1 up
# /sbin/route -A inet6 add <préfixepourlaroute1> dev sit1
# /sbin/ifconfig sit0 tunnel <adresseipv4dutunnelétranger2>
# /sbin/ifconfig sit2 up
# /sbin/route -A inet6 add <préfixepourlaroute2> dev sit2
# /sbin/ifconfig sit0 tunnel <adresseipv4dutunnelétranger3>
# /sbin/ifconfig sit3 up
# /sbin/route -A inet6 add <préfixepourlaroute3> dev sit3
```

Important: NE JAMAIS FAIRE DE LA SORTE, car cette façon de faire rend implicitement disponible le "tunnelage automatique" à partir de n'importe où dans l'Internet, c'est un risque, et cela ne devrait jamais être préconisé.

9.3.1.3. Utiliser seulement "route"

Il est aussi possible d'installer des tunnels dans le style Accès Multiple Sans Diffusion (*Non Broadcast Multiple Access*, ou NBMA), c'est un moyen facile d'ajouter de nombreux tunnels en une fois.

Usage (exemple générique pour trois tunnels):

```
# /sbin/ifconfig sit0 up
# /sbin/route -A inet6 add <préfixepourlaroute1> gw
¬ ::<adresseipv4dutunnelétranger1> dev sit0
# /sbin/route -A inet6 add <préfixepourlaroute2> gw
¬ ::<adresseipv4dutunnelétranger2> dev sit0
# /sbin/route -A inet6 add <préfixepourlaroute3> gw
¬ ::<adresseipv4dutunnelétranger3> dev sit0
```

Important: NE JAMAIS FAIRE DE LA SORTE, car cette façon de faire rend implicitement disponible le "tunnelage automatique" à partir de n'importe où dans l'Internet, c'est un risque, et cela ne devrait jamais être préconisé.

9.3.2. Ôter des tunnels point–à–point

Rarement réalisé manuellement, mais utilisé par les scripts pour une extinction propre ou un redémarrage de la configuration IPv6.

9.3.2.1. Utiliser "ip"

Pour ôter un périphérique de tunnelage:

```
# /sbin/ip tunnel del <périphérique>
```

Usage (exemple générique pour trois tunnels):

```
# /sbin/ip -6 route del <préfixepourlaroute1> dev sit1
# /sbin/ip link set sit1 down
# /sbin/ip tunnel del sit1
# /sbin/ip -6 route del <préfixepourlaroute2> dev sit2
# /sbin/ip link set sit2 down
# /sbin/ip tunnel del sit2
# /sbin/ip -6 route del <préfixepourlaroute3> dev sit3
# /sbin/ip link set sit3 down
# /sbin/ip tunnel del sit3
```

9.3.2.2. Utiliser "ifconfig" et "route" (méthode dépréciée parce qu'elle n'est pas très drôle)

Ce n'est pas seulement la création qui est étrange, mais l'extinction aussi... vous devez ôter les tunnels dans l'ordre inverse, ce qui signifie que le premier créé doit être le premier ôté.

Usage (exemple générique pour trois tunnels):

```
# /sbin/route -A inet6 del <préfixepourlaroute3> dev sit3
# /sbin/ifconfig sit3 down
# /sbin/route -A inet6 del <préfixepourlaroute2> dev sit2
# /sbin/ifconfig sit2 down
# /sbin/route -A inet6 add <préfixepourlaroute1> dev sit1
# /sbin/ifconfig sit1 down
# /sbin/ifconfig sit0 down
```

9.3.2.3. Utiliser "route"

Comme pour ôter des routes IPv6 courantes

Usage (exemple générique pour trois tunnels):

```
# /sbin/route -A inet6 del <préfixepourlaroute1> gw
¬ ::<adresseipv4dutunnelétranger1> dev sit0
# /sbin/route -A inet6 del <préfixepourlaroute2> gw
¬ ::<adresseipv4dutunnelétranger2> dev sit0
# /sbin/route -A inet6 del <préfixepourlaroute3> gw
¬ ::<adresseipv4dutunnelétranger3> dev sit0
# /sbin/ifconfig sit0 down
```

9.3.3. Attribution d'une adresse (*numbered*) à un tunnel point–à–point

Il est parfois nécessaire de configurer un tunnel point–à–point avec des adresses IPv6 comme pour IPv4 aujourd'hui. C'est seulement possible avec la première méthode (ifconfig+route – dépréciée) et la troisième méthode (ip+route) d'installation de tunnel. Dans de tels cas, vous pouvez ajouter l'adresse IPv6 à l'interface de tunnelage comme montré dans la configuration d'interface.

9.4. Installation des tunnels 6to4

Prenez garde au fait que le support des tunnels 6to4 est actuellement manquant sur la série des noyaux vanille 2.2.x (voir la vérification du système / noyau pour plus de détails). Notez aussi que la longueur du préfixe d'une adresse 6to4 est de 16, car, du point de vue du réseau, tous les autres hôtes 6to4 sont sur la même

couche 2.

9.4.1. Ajouter un tunnel 6to4

Vous avez premièrement à calculer votre préfixe 6to4 en utilisant votre adresse IPv4 routable assignée localement (si votre hôte n'a pas d'adresse IPv4 routable, dans des cas précis, NAT sur une passerelle est possible):

En considérant que votre adresse IPv4 soit

```
1.2.3.4
```

le préfixe 6to4 généré sera

```
2002:0102:0304::
```

Les passerelles locales 6to4 devraient (mais cela n'est pas une nécessité, vous pouvez choisir un préfixe arbitraire de portée locale, si cela vous sied mieux) toujours assigner le suffixe "::1", ce qui vous donnera comme adresse 6to4 locale

```
2002:0102:0304::1
```

Utiliser par exemple ce qui suit pour une génération automatique:

```
ipv4="1.2.3.4"; printf "2002:%02x%02x:%02x%02x::1" `echo $ipv4 | tr "." " "`
```

Il y a maintenant deux façons possibles de mettre en place un tunnelage 6to4.

9.4.1.1. Utiliser "ip" et un périphérique tunnel dédié

C'est dorénavant la façon de faire qui est recommandée (une TTL doit être spécifiée, car le défaut est 0).

Créez un nouveau périphérique tunnel

```
# /sbin/ip tunnel add tun6to4 mode sit ttl <ttlpardéfaut> remote any local <adresseipv4locale>
```

Montez l'interface

```
# /sbin/ip link set dev tun6to4 up
```

Ajouter une adresse 6to4 locale à l'interface (note: la longueur du préfixe, 16, est importante!)

```
# /sbin/ip -6 addr add <adresse6to4locale>/16 dev tun6to4
```

Ajouter une route (par défaut) au réseau global IPv6 en utilisant l'adresse anycast tous-routeurs-6to4 (*all-6to4-routers*)

```
# /sbin/ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric 1
```

Il a été rapporté que certaines versions de "ip" (par exemple Linux SuSe 9.0) ne prennent pas en charge les adresses IPv6 compatibles IPv4 pour les passerelles. Dans ce cas, l'adresse IPv6 correspondante doit être employée:

```
# /sbin/ip -6 route add 2000::/3 via 2002:c058:6301::1 dev tun6to4 metric 1
```

9.4.1.2. Utiliser "ifconfig", "route" et le périphérique de tunnelage "sit0" (méthode dépréciée)

Cela est déprécié car le périphérique de tunnel générique sit0 ne permet pas de spécifier un filtrage par périphérique.

Monter l'interface de tunnelage générique sit0

```
# /sbin/ifconfig sit0 up
```

Ajouter une adresse 6to4 locale à une interface

```
# /sbin/ifconfig sit0 add <adresse6to4locale>/16
```

Ajouter une route (par défaut) au réseau global IPv6 en utilisant l'adresse anycast IPv4 tous-relais-6to4 (*all-6to4-relays*)

```
# /sbin/route -A inet6 add 2000::/3 gw ::192.88.99.1 dev sit0
```

9.4.2. Ôter un tunnel 6to4

9.4.2.1. Utiliser "ip" et un périphérique de tunnelage dédié

Ôter toutes les routes traversant ce périphérique de tunnelage spécifique

```
# /sbin/ip -6 route flush dev tun6to4
```

Démonter l'interface

```
# /sbin/ip link set dev tun6to4 down
```

Ôter un périphérique tunnel

```
# /sbin/ip tunnel del tun6to4
```

9.4.2.2. Utiliser "ifconfig", "route" et un périphérique de tunnel générique "sit0" (déprécié)

Ôter une route (par défaut) traversant une interface tunnel 6to4

```
# /sbin/route -A inet6 del 2000::/3 gw ::192.88.99.1 dev sit0
```

Ôter une adresse locale 6to4 d'une interface

```
# /sbin/ifconfig sit0 del <adresse6to4locale>/16
```

Démontage d'un périphérique de tunnelage générique (prenez garde, peut-être est-il utilisé...)

```
# /sbin/ifconfig sit0 down
```

Chapitre 10. Configurer les tunnels IPv4–in–IPv6

Cela sera complété à l'avenir. Pour le moment, de tels tunnels sont essentiellement employés en environnement de test, mais il semble que le support soit actuellement manquant pour linux (03/2004).

Pour l'heure, plus d'information dans le RFC 2473 / Generic Packet Tunneling in IPv6 Specification

Chapitre 11. Les réglages du noyau dans le système de fichiers /proc

Note: la source de cette section est essentiellement le fichier "ip−sysctl.txt", qui est inclus dans les sources du noyau actuel, dans le répertoire "Documentation/networking". Le crédit va à Pekka Savola qui maintient la partie de ce fichier relative à IPv6. D'autres textes sont aussi plus ou moins copier/coller dans cette partie de document.

11.1. Comment accéder au système de fichiers /proc

11.1.1. Utiliser "cat" et "echo"

Utiliser "cat" et "echo" est le moyen le plus simple d'accéder au système de fichiers /proc, mais certains pré−requis sont nécessaires à cela

- Le système de fichiers /proc doit être rendu disponible dans le noyau, ce qui signifie qu'à la compilation le commutateur suivant doit avoir été positionné

```
CONFIG_PROC_FS=y
```

- Le système de fichiers /proc doit être auparavant monté, ce qui peut être testé en faisant

```
# mount | grep "type proc"
none on /proc type proc (rw)
```

- Vous devez pouvoir lire le système de fichiers /proc et parfois aussi y écrire (normalement seul root le peut)

Normalement, seules les entrées dans /proc/sys/* sont en écriture, les autres sont en lecture seule et servent seulement à la récupération de l'information.

11.1.1.1. Récupérer une valeur

La valeur de l'entrée peut être récupérée en utilisant "cat":

```
# cat /proc/sys/net/ipv6/conf/all/forwarding
0
```

11.1.1.2. Fixer une valeur

Une nouvelle valeur peut être fixée (si l'entrée est en écriture) en utilisant echo:

```
# echo "1" >/proc/sys/net/ipv6/conf/all/forwarding
```

11.1.2. Utiliser "sysctl"

Utiliser le programme "sysctl" pour accéder aux commutateurs du noyau est une méthode moderne aujourd'hui. Vous pouvez aussi l'utiliser même si le système de fichiers /proc n'est pas monté. Mais vous n'avez alors accès qu'à /proc/sys/*!

Le programme "sysctl" est compris dans le paquetage "procps" (sur le système Red Hat).

- L'interface sysctl doit être disponible dans le noyau, ce qui signifie qu'à la compilation le commutateur suivant a à être fixé

```
CONFIG_SYSCTL=y
```

11.1.2.1. Récupérer une valeur

La valeur de l'entrée peut maintenant être récupérée:

```
# sysctl net.ipv6.conf.all.forwarding
net.ipv6.conf.all.forwarding = 0
```

11.1.2.2. Fixer une valeur

Une nouvelle valeur peut être fixée (si l'entrée est en écriture):

```
# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
```

Note: n'utilisez pas d'espaces autour du signe "=" lorsque vous fixez les valeurs. De même pour une valeur multiple sur une même ligne, mettez des guillemets comme ceci

```
# sysctl -w net.ipv4.ip_local_port_range="32768 61000"
net.ipv4.ip_local_port_range = 32768 61000
```

11.1.2.3. En plus

Note: il existe dans la pratique certaines versions de sysctl qui affichent "/" au lieu de "."

Pour plus de détails jetez un coup d'oeil dans la page de manuel de sysctl.

une astuce: pour une recherche rapide parmi les réglages, utiliser "−a" (afficher toutes les entrées) en conjonction avec "grep".

11.1.3. Les types de valeur trouvés dans le système de fichiers /proc

IL y a plusieurs formats observés dans le système de fichiers /proc:

- BOOLÉEN: simple "0" (faux) ou "1" (vrai)
- ENTIER: une valeur entière, peut être également non signée

- Des lignes plus sophistiquées avec plusieurs valeurs: parfois un en−tête est aussi affiché, sinon, jetez un coup d'oeil aux sources du noyau pour savoir quel sens possède telle ou telle valeur...

11.2. Les entrées de /proc/sys/net/ipv6/

11.2.1. conf/default/*

Changer les réglages par défaut spécifiques à chaque interface.

11.2.2. conf/all/*

Changer tous les réglages spécifiques aux interfaces.

Exception: "conf/all/forwarding" a une signification différente ici

11.2.2.1. conf/all/forwarding

- Type: BOOLÉEN

Ceci rend disponible le renvoi global IPv6 entre toutes les interfaces.

En IPv6, vous ne pouvez contrôler le renvoi par périphérique, le contrôle du renvoi doit être réalisé en utilisant les jeux de règles de netfilter−IPv6 (contrôlés grâce à ip6tables) en spécifiant les périphériques d'entrée et de sortie (voir comment mettre en place un pare−feu/Netfilter6 pour plus d'information); à la différence d'IPv4, où vous pouvez contrôler le renvoi périphérique par périphérique (la décision est prise sur l'interface qui reçoit des paquets).

Ceci fixe aussi le réglage du renvoi Hôte/Routeur de toutes les interfaces à la valeur spécifiée. Voir plus bas pour plus de détails. Tout ceci relève du renvoi global.

Si cette valeur est à 0, aucun renvoi IPv6 n'est disponible, jamais aucun paquet ne part vers une autre interface, ni physique, ni logique, comme par exemple un tunnel.

11.2.3. conf/interface/*

Changer les réglages spécifiques à chaque interface.

Le comportement fonctionnel de certains réglages est dépendant du positionnement du renvoi local, disponible ou non.

11.2.3.1. accept_ra

- Type: BOOLÉEN
- Défaut fonctionnel: disponible si le renvoi local est disponible; indisponible si le renvoi local est disponible.

Accepter les annonces de routeur, et auto−configurer cette interface avec les données reçues.

11.2.3.2. accept_redirects

- Type: BOOLÉEN
- Défaut fonctionnel: disponible si le renvoi local est indisponible. Indisponible si le renvoi local est disponible.

Accepter les redirections émises par un routeur IPv6.

11.2.3.3. autoconf

- Type: BOOLÉEN
- Défaut: VRAI

Configurer les adresses lien−local (voir aussi Les types d'adresse) utilisant les adresses matérielles L2. Par exemple, ceci génère, comme par magie, une adresse telle que "fe80::201:23ff:fe45:6789" sur une interface ayant une adresse MAC−L2.

11.2.3.4. dad_transmits

- Type: ENTIER
- Défaut: 1

Quantité de message de détection d'adresse dupliquée à émettre.

11.2.3.5. forwarding

- Type: BOOLÉEN
- Défaut: FAUX si le renvoi global est indisponible (défaut), sinon VRAI

Configurer le comportement spécifique à chaque interface Hôte/Routeur.

Note: Il est recommandé d'avoir le même réglage sur toutes les interfaces; mélanger les scénarii routeur/hôte est plutôt atypique.

- Valeur FAUX: Par défaut, le comportement d'hôte est assumé. Cela signifie que:

1. Le drapeau IsRouter n'est pas positionné dans les annonces de voisinage.
2. Les sollicitations de routeur sont envoyées dès que nécessaires.
3. Si accept_ra est VRAI (défaut), accepte les annonces de routeur (et réalise une auto−configuration).
4. Si accept_redirects est VRAI (défaut), accepte les redirections.

- Valeur VRAI: si le renvoi local est disponible, le comportement d'un routeur est assumé. Ceci signifie l'opposé de ce qui précéde:

1. Le drapeau IsRouter est positionné dans les annonces de voisinage.
2. Les sollicitations de routeur ne sont pas émises.
3. Les annonces de routeur sont ignorées.
4. Les redirections sont ignorées.

11.2.3.6. hop_limit

- Type: ENTIER
- Défaut: 64

Nombre limite de sauts par défaut.

11.2.3.7. mtu

- Type: ENTIER
- Défaut: 1280 (minimum requis pour IPv6)

Unité de transfert maximum par défaut

11.2.3.8. router_solicitation_delay

- Type: ENTIER
- Défaut: 1

Nombre de secondes à attendre après le montage d'une interface avant d'émettre des sollicitations de routeur.

11.2.3.9. router_solicitation_interval

- Type: ENTIER
- Défaut: 4

Nombre de secondes d'attente entre les émissions de sollicitations de routeur.

11.2.3.10. router_solicitations

- Type: ENTIER
- Défaut: 3

Nombre de sollicitation(s) de routeur à émettre avant de considérer qu'aucun routeur n'est présent.

11.2.4. neigh/default/*

Changer les réglages par défaut pour la détection de voisinage et certaines valeurs d'intervalle global et de déclenchement (*threshold*):

11.2.4.1. gc_thresh1

- Type: ENTIER
- Défaut: 128

A remplir plus avant.

11.2.4.2. gc_thresh2

- Type: ENTIER
- Défaut: 512

A remplir plus avant.

11.2.4.3. gc_thresh3

- Type: ENTIER
- Défaut: 1024

Paramètre de réglage de la taille de la table du voisinage.

Augmenter cette valeur si vous avez de nombreuses interfaces et un problème avec des routes qui commencent à mystérieusement s'activer et échouer. Ou si un démon de routage Zebra en cours d'activité rapporte cette erreur:

```
ZEBRA: netlink-listen error: No buffer space available, type=RTM_NEWROUTE(24), seq=426, pid=0
```

11.2.4.4. gc_interval

- Type: ENTIER
- Défaut: 30

A remplir plus avant.

11.2.5. neigh/interface/*

Changez ces réglages spécifiques à chaque interface pour la détection de voisinage.

11.2.5.1. anycast_delay

- Type: ENTIER
- Défaut: 100

A remplir plus avant.

11.2.5.2. gc_stale_time

- Type: ENTIER
- Défaut: 60

A remplir plus avant.

11.2.5.3. proxy_qlen

- Type: ENTIER
- Défaut: 64

A remplir plus avant.

11.2.5.4. unres_qlen

- Type: ENTIER
- Défaut: 3

A remplir plus avant.

11.2.5.5. app_solicit

- Type: ENTIER
- Défaut: 0

A remplir plus avant.

11.2.5.6. locktime

- Type: ENTIER
- Défaut: 0

A remplir plus avant.

11.2.5.7. retrans_time

- Type: ENTIER
- Défaut: 100

A remplir plus avant.

11.2.5.8. base_reachable_time

- Type: ENTIER
- Défaut: 30

A remplir plus avant.

11.2.5.9. mcast_solicit

- Type: ENTIER
- Défaut: 3

A remplir plus avant.

11.2.5.10. ucast_solicit

- Type: ENTIER
- Défaut: 3

A remplir plus avant.

11.2.5.11. delay_first_probe_time

- Type: ENTIER
- Défaut: 5

A remplir plus avant.

11.2.5.12. proxy_delay

- Type: ENTIER
- Défaut: 80

A remplir plus avant.

11.2.6. route/*

Changer les réglages globaux du routage.

11.2.6.1. flush

Retiré des nouvelles versions du noyau .

11.2.6.2. gc_interval

- Type: ENTIER
- Défaut: 30

A remplir plus avant.

11.2.6.3. gc_thresh

- Type: ENTIER
- Défaut: 1024

A remplir plus avant.

11.2.6.4. mtu_expires

- Type: ENTIER
- Défaut: 600

A remplir plus avant.

11.2.6.5. gc_elasticity

- Type: ENTIER
- Défaut: 0

A remplir plus avant.

11.2.6.6. gc_min_interval

- Type: ENTIER
- Défaut: 5

A remplir plus avant.

11.2.6.7. gc_timeout

- Type: ENTIER
- Défaut: 60

A remplir plus avant.

11.2.6.8. min_adv_mss

- Type: ENTIER
- Défaut: 12

A remplir plus avant.

11.2.6.9. max_size

- Type: ENTIER
- Défaut: 4096

A remplir plus avant.

11.3. Les entrées relatives à IPv6 dans /proc/sys/net/ipv4/

Pour le moment (et cela sera valable jusqu'à ce qu'IPv4 soit complètement converti en un module indépendant du noyau), certains commutateurs IPv4 sont aussi utilisés par IPv6.

11.3.1. ip_*

11.3.1.1. ip_local_port_range

Ce réglage sont aussi utilisé par IPv6.

11.3.2. tcp_*

Ces réglages sont aussi utilisés par IPv6.

11.3.3. icmp_*

Ces réglages ne sont pas utilisés par IPv6. Pour réaliser une limitation du trafic ICMPv6 (ce qui est très recommandé compte tenu de possibles engorgements ICMPv6), des règles netfilter−v6 doivent être utilisées.

11.3.4. autre(s)

Inconnu(s), mais probablement inutilisé(s) par IPv6.

11.4. Les entrées relatives à IPv6 dans /proc/net/

Dans /proc/net il y a plusieurs entrées disponibles en lecture seule. Vous ne pouvez pas utiliser ici "sysctl" afin de récupérer des informations, utiliser "cat".

11.4.1. if_inet6

- Type: Une ligne par adresse comporte plusieurs valeurs

Ici toutes les adresses IPv6 configurées sont montrées sous un format particulier. L'exemple affiche seulement l'interface loopback. Sa signification est détaillée ci−dessous (voir "net/ipv6/addrconf.c" pour en savoir plus).

```
# cat /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80 lo
+--------------------------------+ ++ ++ ++ ++ ++
|                                  |  |  |  |  |
1                                  2  3  4  5  6
```

1. L'adresse IPv6 affichée grâce à 32 caractères hexadécimaux sans le séparateur ":"
2. Numéro de périphérique Netlink (index d'interface) in hexadécimal (voir aussi "ip addr")
3. La longueur du préfixe en hexadécimal
4. La valeur de la portée (voir les sources du noyau "include/net/ipv6.h" et "net/ipv6/addrconf.c" pour plus de détails)
5. Les drapeaux de l'interface (voir "include/linux/rtnetlink.h" et "net/ipv6/addrconf.c" pour en savoir plus)
6. Le nom du périphérique

11.4.2. ipv6_route

- Type: Une ligne par route comporte plusieurs valeurs

Ici toutes les routes IPv6 configurées sont montrées dans un format particulier. L'exemple affiche seulement l'interface loopback. Sa signification est détaillée ci−dessous (voir "net/ipv6/route.c" pour en savoir plus).

```
# cat /proc/net/ipv6_route
00000000000000000000000000000000 00 00000000000000000000000000000000 00
+--------------------------------+ ++ +--------------------------------+ ++
|                                  |  |                                  |
1                                  2  3                                  4
¬ 00000000000000000000000000000000 ffffffff 00000001 00000001 00200200 lo
¬ +--------------------------------+ +------+ +------+ +------+ +------+ ++
¬ |                                  |        |        |        |        |
¬ 5                                  6        7        8        9        10
```

1. Le réseau de destination IPv6 affiché grâce à 32 caractères hexadécimaux sans le séparateur ":"
2. La longueur du préfixe de destination IPv6 en hexadécimal

3. Le réseau source IPv6 affiché grâce à 32 caractères hexadécimaux sans le séparateur ":"
4. La longueur du préfixe de la source IPv6 en hexadécimal
5. Le prochain saut IPv6 affiché grâce à 32 caractères hexadécimaux sans le séparateur ":"
6. La distance en hexadécimal
7. Compteur de référence
8. Compteur d'utilisation
9. Les drapeaux
10. Nom du périphérique

11.4.3. sockstat6

- Type: Une ligne par protocole avec description et valeur

Statistiques à propos de l'utilisation des sockets IPv6. Exemple:

```
# cat /proc/net/sockstat6
TCP6: inuse 7
UDP6: inuse 2
RAW6: inuse 1
FRAG6: inuse 0 memory 0
```

11.4.4. tcp6

A remplir.

11.4.5. udp6

A remplir.

11.4.6. igmp6

A remplir.

11.4.7. raw6

A remplir.

11.4.8. ip6_flowlabel

A remplir.

11.4.9. rt6_stats

A remplir.

11.4.10. snmp6

- Type: Une ligne par description et valeur SNMP

Statistiques SNMP, peuvent être récupérées par un serveur SNMP et mis en rapport à une tableau MIB grâce à un logiciel d'administration réseau.

11.4.11. ip6_tables_names

Tables netfilter6 disponibles

Chapitre 12. L'interface de netlink vers le noyau

A remplir... je n'ai en cela pas d'expérience...

Chapitre 13. Le déboguage réseau

13.1. Les sockets d'écoute de serveur

13.1.1. Utiliser "netstat" pour vérifier les sockets d'écoute de serveur

C'est toujours intéressant de savoir quelles sockets de serveur sont actives à un moment donné sur un noeud. Utiliser "netstat" est le moyen le plus court pour obtenir une telle information:

options employées: –nlptu

Exemple:

```
# netstat -nlptu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
¬ PID/Program name
tcp        0      0 0.0.0.0:32768           0.0.0.0:*               LISTEN
¬ 1258/rpc.statd
tcp        0      0 0.0.0.0:32769           0.0.0.0:*               LISTEN
¬ 1502/rpc.mountd
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
¬ 22433/lpd Waiting
tcp        0      0 1.2.3.1:139             0.0.0.0:*               LISTEN
¬ 1746/smbd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
¬ 1230/portmap
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
¬ 3551/X
tcp        0      0 1.2.3.1:8081            0.0.0.0:*               LISTEN
¬ 18735/junkbuster
tcp        0      0 1.2.3.1:3128            0.0.0.0:*               LISTEN
¬ 18822/(squid)
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
¬ 30734/named
tcp        0      0 ::ffff:1.2.3.1:993      :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 :::13                   :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 ::ffff:1.2.3.1:143      :::*                    LISTEN
¬ 6742/xinetd-ipv6
tcp        0      0 :::53                   :::*                    LISTEN
¬ 30734/named
tcp        0      0 :::22                   :::*                    LISTEN
¬ 1410/sshd
tcp        0      0 :::6010                 :::*                    LISTEN
¬ 13237/sshd
udp        0      0 0.0.0.0:32768           0.0.0.0:*
¬ 1258/rpc.statd
udp        0      0 0.0.0.0:2049            0.0.0.0:*
¬ -
udp        0      0 0.0.0.0:32770           0.0.0.0:*
¬ 1502/rpc.mountd
udp        0      0 0.0.0.0:32771           0.0.0.0:*
¬ -
udp        0      0 1.2.3.1:137             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:137             0.0.0.0:*
```

```
¬ 1751/nmbd
udp        0      0 1.2.3.1:138             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:138             0.0.0.0:*
¬ 1751/nmbd
udp        0      0 0.0.0.0:33044           0.0.0.0:*
¬ 30734/named
udp        0      0 1.2.3.1:53              0.0.0.0:*
¬ 30734/named
udp        0      0 127.0.0.1:53            0.0.0.0:*
¬ 30734/named
udp        0      0 0.0.0.0:67              0.0.0.0:*
¬ 1530/dhcpd
udp        0      0 0.0.0.0:67              0.0.0.0:*
¬ 1530/dhcpd
udp        0      0 0.0.0.0:32858           0.0.0.0:*
¬ 18822/(squid)
udp        0      0 0.0.0.0:4827            0.0.0.0:*
¬ 18822/(squid)
udp        0      0 0.0.0.0:111             0.0.0.0:*
¬ 1230/portmap
udp        0      0 :::53                   :::*
¬ 30734/named
```

13.2. Des exemples de dump provenant de tcpdump

Suivent quelques exemples de paquets capturés, cela sera peut−être utile pour vos propres déboguages...

...plus d'info à venir...

13.2.1. La découverte de routeur

13.2.1.1. Une annonce de routeur

```
15:43:49.484751 fe80::212:34ff:fe12:3450 > ff02::1: icmp6: router
¬ advertisement(chlim=64, router_ltime=30, reachable_time=0,
¬ retrans_time=0)(prefix info: AR valid_ltime=30, preffered_ltime=20,
¬ prefix=2002:0102:0304:1::/64)(prefix info: LAR valid_ltime=2592000,
¬ preffered_ltime=604800, prefix=3ffe:ffff:0:1::/64)(src lladdr:
¬ 0:12:34:12:34:50) (len 88, hlim 255)
```

Un routeur, avec pour adresse lien−local "fe80::212:34ff:fe12:3450", émet une annonce à l'adresse multicast tous−les−noeuds−du−lien (*all−node−on−link*) "ff02::1", contenant deux préfixes, "2002:0102:0304:1::/64" (d'une durée de vie de 30 s) et "3ffe:ffff:0:1::/64" (d'une durée de vie de 2592000 s), incluant sa propre adresse MAC de couche 2, "0:12:34:12:34:50".

13.2.1.2. Une sollicitation de routeur

```
15:44:21.152646 fe80::212:34ff:fe12:3456 > ff02::2: icmp6: router solicitation
¬ (src lladdr: 0:12:34:12:34:56) (len 16, hlim 255)
```

Un noeud, avec pour adresse lien−local "fe80::212:34ff:fe12:3456" et comme adresse de couche 2 "0:12:34:12:34:56", est en quête d'un routeur présent sur le lien, en conséquence il émet cette sollicitation à l'adresse multicast tous−routeurs−présents−sur−le−lien (*all−router−on−link*) "ff02::2".

13.2.2. La découverte de voisinage

13.2.2.1. Une sollicitation de découverte de voisinage afin de détecter une possible duplication d'adresse

Les paquets suivants sont émis par un noeud sur la couche 2, adresse MAC "0:12:34:12:34:56", pendant l'auto−configuration, afin de vérifier si une adresse potentielle est déjà employée ou non par un autre noeud sur le lien permettant d'émettre ces paquets, *via* l'adresse multicast lien−local du noeud sollicité.

- Le noeud veut configurer son lien−local avec l'adresse "fe80::212:34ff:fe12:3456", il est en train de vérifier s'il y a duplication

```
15:44:17.712338 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ fe80::212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32, hlim 255)
```

- Le noeud veut configurer son adresse globale "2002:0102:0304:1:212:34ff:fe12:3456" (après avoir reçu l'annonce montrée plus haut), il est en train de vérifier s'il y a ou non duplication

```
15:44:21.905596 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ 2002:0102:0304:1:212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32,
¬ hlim 255)
```

- Le noeud veut configurer son adresse globale "3ffe:ffff:0:1:212:34ff:fe12:3456" (après avoir reçu l'annonce montrée plus haut), il est en train de vérifier s'il y a ou non duplication

```
15:44:22.304028 :: > ff02::1:ff12:3456: icmp6: neighbor sol: who has
¬ 3ffe:ffff:0:1:212:34ff:fe12:3456(src lladdr: 0:12:34:12:34:56) (len 32, hlim
¬ 255)
```

13.2.2.2. Une sollicitation de découverte de voisinage à la recherche d'hôte(s) ou de passerelle(s)

- Un noeud veut émettre des paquets à "3ffe:ffff:0:1::10" mais il n'a aucune adresse MAC de la couche 2 vers laquelle il pourrait émettre, il émet alors maintenant une sollicitation

```
13:07:47.664538 2002:0102:0304:1:2e0:18ff:fe90:9205 > ff02::1:ff00:10: icmp6:
¬ neighbor sol: who has 3ffe:ffff:0:1::10(src lladdr: 0:e0:18:90:92:5) (len 32,
¬ hlim 255)
```

- Ce noeud recherche maintenant "fe80::10"

```
13:11:20.870070 fe80::2e0:18ff:fe90:9205 > ff02::1:ff00:10: icmp6: neighbor
¬ sol: who has fe80::10(src lladdr: 0:e0:18:90:92:5) (len 32, hlim 255)
```

Chapitre 14. Support à la configuration persistante IPv6 dans les distributions Linux

Certaines distributions Linux contiennent déjà un support à la configuration persistante IPv6 utilisant une configuration nouvelle ou préexistante, des fichiers de script, et des accroches dans les fichiers de script IPv4.

14.1. Linux Red Hat et ses "clones"

Depuis que j'ai commencé à écrire l'Howto –IPv6 & Linux , il était dans mon intention de rendre disponible une configuration convenant aux cas les plus fréquents tels que hôte simple, routeur simple, hôte à double résidence, routeur avec un second tronçon réseau, tunnel typique, tunnel 6to4, *etc*. De nos jours, il existe des fichiers de configuration et des scripts qui font très bien ce travail (je n'ai jamais entendu parler de vrais problèmes, mais je ne sais pas s'ils sont beaucoup utilisés). Parce que cette configuration et ces scripts augmentent régulièrement en volume, ils ont leur propre page HOWTO: initscripts–ipv6 (miroir). Parce que j'ai commencé mon expérience IPv6 sur un clone de la Linux Red Hat 5.0, mes développements concernant IPv6 sont encore essentiellement basés sur Linux Red Hat, il est par conséquent un peu logique que ces scripts soient développés sur ce type de distribution (on appelle ça une raison historique). Il est ainsi très facile d'étendre certains de ces fichiers de configuration, d'en créer de nouveaux et de créer de simples accroches d'appel à l'installation d'IPv6 à partir de l'installation d'IPV4.

Depuis la Red Hat 7.1, une archive de mes scripts y est incluse. Cela est dû, et cela sera encore vrai à l'avenir, à l'assistance de Pekka Savola.

La Mandrake, depuis la version 8.0, inclut aussi un paquetage initscript prêt pour IPv6, cependant un bogue mineur retient de l'employer (il manque 'inet6" à "ifconfig" avant "add").

14.1.1. Tester la présence des scripts de configuration IPv6

Vous pouvez tester si votre distribution Linux contient le support pour la configuration persistante IPv6 utilisant mon jeu d'outils. Le script de la bibliothèque devrait exister:

```
/etc/sysconfig/network-scripts/network-functions-ipv6
```

Un test magique:

```
# test -f /etc/sysconfig/network-scripts/network-functions-ipv6 && echo "Main
¬ IPv6 script library exists"
```

La version de la bibliothèque est importante s'il vous manque certaines fonctionnalités. Vous pouvez l'obtenir en exécutant ce qui suit (ou d'une façon encore plus aisée en regardant le haut du fichier):

```
# source /etc/sysconfig/network-scripts/network-functions-ipv6 &&
¬ getversion_ipv6_functions
20011124
```

Dans l'exemple montré, la version utilisée est la 20011124. Vérifiez cela par rapport à l'information la plus à jour sur la page initscripts–ipv6 (miroir) afin de voir ce qui a changé. Vous y trouverez aussi un journal des modifications.

14.1.2. Quelques éléments pour rendre disponible IPv6 sur les actuelles RHL 7.1, 7.2, 7.3,...

- Vérifiez si votre système a déjà le module IPv6 chargé

```
# modprobe -c | grep net-pf-10
alias net-pf-10 off
```

- Si le résultat est "off", alors rendez disponible la mise en réseau IPv6 en éditant /etc/sysconfig/network, ajoutez la nouvelle ligne

```
NETWORKING_IPV6=yes
```

- Redémarrez la machine, ou simplement le réseau par

```
# service network restart
```

- Maintenant le module IPv6 devrait être chargé

```
# modprobe -c | grep ipv6
alias net-pf-10 ipv6
```

Si votre système est sur un lien fournissant l'annonce de routeur, la configuration sera réalisée automatiquement. Pour plus d'information sur les réglages supportées, voir /usr/share/doc/initscripts-$version/sysconfig.txt.

14.2. Linux SuSE

Dans les nouvelles versions, il n'y a véritablement qu'un support rudimentaire disponible, voir /etc/rc.config pour les détails.

A cause de sa configuration très différente et de la structure de ses scripts, il est difficile (voire impossible) d'utiliser le jeu d'outils de Linux Red Hat et de ses clones avec cette distribution. Dans les versions 8.x, SuSE va complétement modifier l'installation de sa configuration.

14.2.1. Linux SuSE 7.3

- Comment installer IPv6 6to4 pour la SuSE 7.3

14.2.2. Linux SuSE 8.0

14.2.2.1. Configuration d'adresse IPv6

Editez le fichier /etc/sysconfig/network/ifcfg-<nom-de-l'interface> et fixez la valeur suivante

```
IP6ADDR="<Adresse-ipv6>/<préfixe>"
```

14.2.2.2. Information supplémentaire

Voir le fichier /usr/share/doc/packages/sysconfig/README

14.2.3. Linux SuSE 8.1

14.2.3.1. Configuration d'adresse IPv6

Editez le fichier /etc/sysconfig/network/ifcfg−<nom−de−l'interface> et fixez la valeur suivante

```
IPADDR="<Adresse-ipv6>/<préfixe>"
```

14.2.3.2. Information supplémentaire

Voir le fichier /usr/share/doc/packages/sysconfig/Network

14.3. Linux Debian

Les informations qui suivent sont une contribution de Stéphane Bortzmeyer <bortzmeyer chez nic point fr>

1. Assurez−vous qu'IPv6 soit chargé; soit il est compilé dans le noyau, soit il est chargé comme module. Dans ce dernier cas, trois solutions, l'ajouter à /etc/modules, utiliser la configuration ci−dessous, ou utiliser kmod (non détaillé ici).
2. Configurez votre interface. Par exemple, ici, nous considérons la prise en compte de eth0, avec pour adresse "3ffe:ffff:1234:5::1:1". Editez /etc/network/interfaces :

```
iface eth0 inet6 static
        pre-up modprobe ipv6
        address 3ffe:ffff:1234:5::1:1
        # Pour rendre complètement indisponible l'auto-configuration:
        # up echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
        netmask 64
        # Le routeur est auto-configuré, et n'a pas d'adresse fixe.
        # Il est déterminé comme par magie
        # (/proc/sys/net/ipv6/conf/all/accept_ra). Sinon:
        # gateway 3ffe:ffff:1234:5::1
```

Puis vous rebootez, ou alors vous faites juste

```
# ifup --force eth0
```

Et vous avez votre adresse statique.

14.3.1. Plus d'information

- IPv6 sur Linux Debian par Craig Small
- HowTo pour Freenet6 & les utilisateurs Debian de Jean−Marc Liotier (annoncé le 24.12.2002 sur la liste de diffusion users@ipv6.org)

Chapitre 15. L'auto−configuration et la mobilité

15.1. L'auto−configuration sans état

Est supportée et observée sur l'adresse lien−local assignée après le montage d'une interface sur laquelle IPv6 est disponible.

Exemple:

```
# ip −6 addr show dev eth0 scope link
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qlen1000
    inet6 fe80::211:d8ff:fe6b:f0f5/64 scope link
         valid_lft forever preferred_lft forever
```

15.2. L'auto−configuration avec état utilisant le Démon d'Annonce de Routeur

A compléter. Voir plus bas l'auto−configuration par le démon radvd (*Router Advertisement Daemon*).

15.3. Le Protocole de Configuration Dynamique d'Hôte version 6 (DHCPv6)

Après de longues discussions concernant les difficultés, le RFC 3315 / Dynamic Host Configuration Protocol for IPv6 (DHCPv6) a finalement vu le jour. Au moment de la mise à jour de ce passage (10/2005), il existe deux implémentations:

- Dibbler par Tomasz Mrugalski <thomson chez klub point com point pl>
- DHCPv6 chez Sourceforge

15.4. La mobilité

A compléter.

Pour le moment, voir la page d'accueil sur la mobilité IPv6 sur Linux (MIPL) pour plus de détails. Plus d'information peut être trouvée ici (merci de signaler les liens brisés):

- draft−oneill−mipv6−cao−??.txt / MIPv6 Care of Address Option
- draft−mccann−mobileip−80211fh−??.txt / Mobile IPv6 Fast Handovers for 802.11 Networks
- draft−haberman−ipv6−anycast−rr−??.txt / IPv6 Anycast Binding using Return Routability
- draft−mun−aaa−localkm−mobileipv6−??.txt / Localized Key Management for AAA in MobileIPv6
- draft−thubert−nemo−ro−taxonomy−??.txt / Taxonomy of Route Optimization Models in the NEMO Context
- draft−le−aaa−diameter−mobileipv6−??.txt / Diameter Mobile IPv6 Application
- draft−wakikawa−manet−globalv6−??.txt / Global Connectivity for IPv6 Mobile Ad Hoc Networks
- draft−ietf−mobileip−fast−mipv6−??.txt / Fast Handovers for Mobile IPv6
- draft−ietf−mobileip−ipv6−??.txt / Mobility Support in IPv6
- draft−ohnishi−mobileip−v6vpngateway−??.txt / Mobile IPv6 VPN using Gateway Home Agent

- draft−ietf−mobileip−hmipv6−??.txt / Hierarchical MIPv6 mobility management (HMIPv6)
- draft−mkhalil−ipv6−fastra−??.txt / IPv6 Fast Router Advertisement
- draft−okazaki−mobileip−abk−??.txt / Securing MIPv6 Binding Updates Using Address Based Keys (ABKs)
- draft−vriz−mobileip−hbhlmap−??.txt / Hop−by−Hop Local Mobility Agents Probing for Mobile IPv6
- draft−thubert−nemo−reverse−routing−header−??.txt / IPv6 Reverse Routing Header and its application to Mobile Networks
- draft−ietf−mobileip−mipv6−ha−ipsec−??.txt / Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents
- draft−suh−rmm−??.txt / Regional Mobile IPv6 mobility management
- draft−mccann−mobileip−ipv6mipv4−??.txt / IPv6 over Mobile IPv4
- draft−kempf−mobileip−fmipv6−sem−??.txt / Improving the Architectural Alignment for FMIPv6
- draft−le−aaa−mipv6−requirements−??.txt / Mobile IPv6 Authentication, Authorization, and Accounting Requirements
- draft−hwang−rohc−mipv6−??.txt / RObust Header Compression (ROHC): A Compression Profile for Mobile IPv6
- LANCASTER MOBILE IPv6 PACKAGE
- Testbed for MIND project on IPv6
- Mobile IPv6 Issue List

Chapitre 16. Mettre en place le pare−feu

Mettre en place un pare−feu IPv6 est très important, tout spécialement si IPv6 est utilisé sur un intranet avec des adresses IPv6 globales. Car, à la différence des réseaux IPv4 où les hôtes internes courants sont protégés par l'usage d'adresses IPv4 privées comme défini par le RFC 1918 / Address Allocation for Private Internets ou l'adressage IP privée automatique (*Automatic Private IP Addressin*g, ou APIPA) recherche Google "Microsoft + APIPA", en IPv6, les adresses globales sont normalement utilisées, et quelqu'un possédant une connectivité IPv6 peut atteindre tous les noeuds propres à un intranet disposant d'IPv6.

16.1. Mettre en place un pare−feu grâce à netfilter

La mise en place d'un pare−feu IPv6 est nativement supportée par les noyaux dont la version est supérieure à 2.4. Avec les anciennes versions inférieures à 2.2, vous pouvez seulement filtrer IPv6−in−IPv4 par le protocole 41.

Attention: il n'y a aucune garantie que les règles décrites ou les exemples fournis puissent protéger votre système!

Faites un audit de votre jeu de règles après son installation, voir l'audit de sécurité sur IPv6 pour en savoir plus.

Notez aussi que le projet USAGI finalise actuellement son travail sur la traque de connexion pour IPv6! Cela rendra la création de jeu de règles plus simple et plus sûre à l'avenir!

16.1.1. Plus d'information

- Le projet Netfilter
- maillist archive of netfilter users
- maillist archive of netfilter developers
- Information non officielle concernant l'état de netfilter

16.2. Préparation

16.2.1. Récupérer les sources

Récupérez les dernières sources du noyau: http://www.kernel.org/

Récupérez le dernier paquetage d'iptables:

- Les sources en archive tar (pour patcher le noyau): http://www.netfilter.org/
- Les sources en RPM pour reconstruire les binaires (pour les systèmes RedHat): ftp://ftp.redhat.com/redhat/linux/rawhide/SRPMS/SRPMS/ ou peut−être encore dans http://www.netcore.fi/pekkas/linux/ipv6/

16.2.2. Extraire les sources

Déplacez−vous dans le répertoire des sources:

```
# cd /chemin/vers/les/sources
```

Décompactez et renommez les sources du noyau

```
# tar z|jxf kernel-version.tar.gz|bz2
# mv linux linux-version-iptables-version+IPv6
```

Décompactez les sources d'iptables

```
# tar z|jxf iptables-version.tar.gz|bz2
```

16.2.3. Appliquer les derniers patchs relatifs à iptables/IPv6 aux sources du noyau

Déplacez−vous dans le répertoire iptables

```
# cd iptables-version
```

Appliquez les patchs en attente

```
# make pending-patches KERNEL_DIR=/chemin/vers/les/sources/linux-version-iptables-version+IPv6/
```

Appliquez les patchs additionnels relatifs à IPv6 (pas encore inclus dans le noyau vanille)

```
# make patch-o-matic KERNEL_DIR=/path/to/src/linux-version-iptables-version/
```

Répondez par l'affirmative aux options suivantes (iptables−1.2.2)

- ah−esp.patch
- masq−dynaddr.patch (nécessaire seulement sur les systèmes ayant une adresse IP dynamique à la connexion au WAN, comme pour PPP ou PPPoE)
- ipv6−agr.patch.ipv6
- ipv6−ports.patch.ipv6
- LOG.patch.ipv6
- REJECT.patch.ipv6

Vérifier la présence des extensions IPv6

```
# make print-extensions
Extensions found: IPv6:owner IPv6:limit IPv6:mac IPv6:multiport
```

16.2.4. Configurer, construire et installer un nouveau noyau

Déplacez−vous dans les sources du noyau

```
# cd /chemin/vers/les/sources/linux-version-iptables-version/
```

Editez Makefile

```
- EXTRAVERSION =
+ EXTRAVERSION = -iptables-version+IPv6-try
```

Lancez configure, avec IPv6 de disponible

```
Code maturity level options
      Prompt for development and/or incomplete code/drivers : yes
Networking options
      Network packet filtering: yes
      The IPv6 protocol: module
           IPv6: Netfilter Configuration
                 IP6 tables support: module
                 All new options like following:
                       limit match support: module
                       MAC address match support: module
                       Multiple port match support: module
                       Owner match support: module
                       netfilter MARK match support: module
                       Aggregated address check: module
                       Packet filtering: module
                            REJECT target support: module
                            LOG target support: module
                       Packet mangling: module
                       MARK target support: module
```

Configurez aussi tout ce qui concerne votre système

Compilez et installez: voir ici même la section noyau et autres HOWTO

16.2.5. Reconstruire et installer les binaires d'iptables

Assurez–vous que l'arborescence des sources du noyau existe aussi dans /usr/src/linux/

Renommez l'ancien répertoire

```
# mv /usr/src/linux /usr/src/linux.old
```

Créez un nouveau lien symbolique

```
# ln -s /chemin/vers/src/linux-version-iptables-version /usr/src/linux
```

Reconstruisez le SRPM

```
# rpm --rebuild /chemin/vers/SRPM/iptables-version-release.src.rpm
```

Installez les nouveaux paquetages iptables (iptables + iptables–ipv6)

- Sur les systèmes RH 7.1, normalement, une ancienne version est installée, en conséquence utiliser "freshen"

```
# rpm -Fhv /chemin/vers/RPMS/cpu/iptables*-version-release.cpu.rpm
```

- Si elle n'était pas installée, utiliser "install"

```
# rpm -ihv /chemin/vers/RPMS/cpu/iptables*-version-release.cpu.rpm
```

- Sur les systèmes RH 6.2, normalement, aucun noyau 2.4.x n'est installé, conséquemment les pré−requis ne correspondent pas. Utiliser "−−nodeps" pour l'installer

```
# rpm -ihv --nodeps /chemin/vers/RPMS/cpu/iptables*-version-release.cpu.rpm
```

Il sera peut−être nécessaire de créer un lien symbolique vers le lieu où les bibliothèques iptables sont

```
# ln -s /lib/iptables/ /usr/lib/iptables
```

16.3. Utilisation

16.3.1. Vérifier le support

Chargez le module, s'il est compilé

```
# modprobe ip6_tables
```

Vérifiez si le noyau courant prend en charge iptables

```
# [ ! -f /proc/net/ip6_tables_names ] && echo "Current kernel doesn't support
¬ 'ip6tables' firewalling (IPv6)!"
```

16.3.2. Apprendre à utiliser ip6tables

16.3.2.1. Lister toutes les entrées netfilter IPv6

- de façon abrégée

```
# ip6tables -L
```

- de façon détaillée

```
# ip6tables -n -v --line-numbers -L
```

16.3.2.2. Lister un filtre spécifique

```
# ip6tables -n -v --line-numbers -L INPUT
```

16.3.2.3. Insérer une règle de journal au filtre entrant, avec des options

```
# ip6tables --table filter --append INPUT  -j LOG --log-prefix "INPUT:"
¬ --log-level 7
```

16.3.2.4. Insérer une règle de destruction (*drop rule*) au filtre entrant

```
# ip6tables --table filter --append INPUT  -j DROP
```

16.3.2.5. Détruire une règle par son numéro

```
# ip6tables --table filter --delete INPUT 1
```

16.3.2.6. Autoriser ICMPv6

Avec les plus anciens noyaux (noyau non patché 2.4.5 et iptables−1.2.2), aucun type ne peut être spécifié

- Accepter le trafic ICMPv6 entrant dans les tunnels

```
# ip6tables -A INPUT -i sit+ -p icmpv6 -j ACCEPT
```

- Autoriser le trafic ICMPv6 sortant des tunnels

```
# ip6tables -A OUTPUT -o sit+ -p icmpv6 -j ACCEPT
```

Les nouveaux noyaux permettent de spécifier les types ICMPv6:

```
# ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -j ACCEPT
```

16.3.2.7. La limitation du débit

Il peut arriver (l'auteur l'a déjà vu) qu'un engorgement ICMPv6 se produise, c'est pourquoi vous devriez utiliser la limitation de débit, puisqu'elle est disponible, et ce, au moins pour le jeu de règles ICMPv6. De plus, des règles de journalisation devraient aussi être mises en place pour garder trace d'attaques DoS, grâce à syslog et au stockage des fichiers de log. Un exemple de limitation du débit ICMPv6 ressemble à ceci:

```
# ip6tables -A INPUT --protocol icmpv6 --icmpv6-type echo-request
¬ -j ACCEPT --match limit --limit 30/minute
```

16.3.2.8. Permettre le trafic entrant SSH

Ici l'exemple montré est un jeu de règles permettant les connexions entrantes SSH par une adresse IPv6 donnée

- Autoriser le trafic entrant SSH provenant de 3ffe:ffff:100::1/128

```
# ip6tables -A INPUT -i sit+ -p tcp -s 3ffe:ffff:100::1/128 --sport 512:65535
¬ --dport 22 -j ACCEPT
```

- Autoriser les paquets réponse (pour le moment, la traque du trafic IPv6 n'est pas au coeur de l'implémentation de netfilter)

```
# ip6tables -A OUTPUT -o sit+ -p tcp -d 3ffe:ffff:100::1/128 --dport 512:65535
¬ --sport 22 ! --syn j ACCEPT
```

16.3.2.9. Rendre disponible le trafic tunnelé IPv6–in–IPv4

Pour accepter les paquets tunnelés IPv6–in–IPv4, vous devez insérer des règles dans votre installation de pare–feu IPv4 relatives à de tels paquets, pour exemple

- Accepter le trafic entrant IPv6–in–IPv4 sur l'interface ppp0

```
# iptables -A INPUT -i ppp0 -p ipv6 -j ACCEPT
```

- Permettre au trafic IPv6–in–IPv4 de sortir par l'interface ppp0

```
# iptables -A OUTPUT -o ppp0 -p ipv6 -j ACCEPT
```

Si vous avez seulement un tunnel statique, vous pouvez aussi spécifier l'adresse IPv4, comme ici

- Accepter le trafic entrant IPv6–in–IPv4 sur l'interface ppp0 et provenant de l'extrémité du tunnel 1.2.3.4

```
# iptables -A INPUT -i ppp0 -p ipv6 -s 1.2.3.4 -j ACCEPT
```

- Autoriser le trafic sortant IPv6–in–IPv4 vers l'interface ppp0 pour l'extrémité du tunnel 1.2.3.4

```
# iptables -A OUTPUT -o ppp0 -p ipv6 -d 1.2.3.4 -j ACCEPT
```

16.3.2.10. Protection contre les requêtes de connexion entrante TCP

TRÈS RECOMMANDÉ! Pour des questions de sécurité, vous devriez vraiment insérer une règle qui bloque les requêtes de connexion TCP entrante. Ajouter l'option "–i", si d'autres noms d'interface sont utilisés!

- Bloquer les requêtes de connexion entrante TCP vers cet hôte

```
# ip6tables -I INPUT -i sit+ -p tcp --syn -j DROP
```

- Bloquer les requêtes de connexion entrante TCP allant vers les hôtes placés derrière ce routeur

```
# ip6tables -I FORWARD -i sit+ -p tcp --syn -j DROP
```

Peut–être ces règles doivent–elles être placées sous d'autres, mais ça, c'est votre travail. La meilleure façon de faire est de créer un script et d'exécuter les règles d'une manière spécifique.

16.3.2.11. Protection contre les requêtes de connexion entrante UDP

TRÈS RECOMMANDÉ AUSSI! Comme il a été dit dans la section concernant la mise en place d'un pare−feu, il est possible de contrôler les ports des sessions sortantes UDP/TCP. Si tous vos systèmes IPv6 locaux utilisent les ports locaux, par exemple de 32768 à 60999, vous êtes aussi capables de filtrer les connexions UDP (jusqu'à ce que la traque des connexions fonctionnent) comme suit:

- Bloquer les paquets entrants UDP qui ne peuvent être des réponses de requêtes sortantes de cet hôte

```
# ip6tables -I INPUT -i sit+ -p udp ! --dport 32768:60999 -j DROP
```

- Bloquer les paquets entrants UDP qui ne peuvent être des réponses de requêtes d'hôtes placés derrière ce routeur, et transitant à travers lui

```
# ip6tables -I FORWARD -i sit+ -p udp ! --dport 32768:60999 -j DROP
```

16.3.3. Un exemple plus conséquent

Les lignes qui suivent montrent en exemple une installation plus sophistiquée. Bonne création de jeux de règles netfilter6...

```
# ip6tables -n -v -L
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 extIN      all      sit+   *       ::/0                 ::/0
    4   384 intIN      all      eth0   *       ::/0                 ::/0
    0     0 ACCEPT     all      *      *       ::1/128              ::1/128
    0     0 ACCEPT     all      lo     *       ::/0                 ::/0
    0     0 LOG        all      *      *       ::/0                 ::/0
¬        LOG flags 0 level 7 prefix `INPUT-default:'
    0     0 DROP       all      *      *       ::/0                 ::/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 int2ext    all      eth0   sit+    ::/0                 ::/0
    0     0 ext2int    all      sit+   eth0    ::/0                 ::/0
    0     0 LOG        all      *      *       ::/0                 ::/0
¬        LOG flags 0 level 7 prefix `FORWARD-default:'
    0     0 DROP       all      *      *       ::/0                 ::/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 extOUT     all      *      sit+    ::/0                 ::/0
    4   384 intOUT     all      *      eth0    ::/0                 ::/0
    0     0 ACCEPT     all      *      *       ::1/128              ::1/128
    0     0 ACCEPT     all      *      lo      ::/0                 ::/0
    0     0 LOG        all      *      *       ::/0                 ::/0
¬        LOG flags 0 level 7 prefix `OUTPUT-default:'
    0     0 DROP       all      *      *       ::/0                 ::/0

Chain ext2int (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
```

```
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1:65535 dpts:1024:65535 flags:!0x16/0x02
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `ext2int-default:'
    0     0 DROP       tcp       *      *       ::/0                ::/0
    0     0 DROP       udp       *      *       ::/0                ::/0
    0     0 DROP       all       *      *       ::/0                ::/0

Chain extIN (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     tcp       *      *       3ffe:400:100::1/128  ::/0
¬        tcp spts:512:65535 dpt:22
    0     0 ACCEPT     tcp       *      *       3ffe:400:100::2/128  ::/0
¬        tcp spts:512:65535 dpt:22
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1:65535 dpts:1024:65535 flags:!0x16/0x02
    0     0 ACCEPT     udp       *      *       ::/0                ::/0
¬        udp spts:1:65535 dpts:1024:65535
    0     0 LOG        all       *      *       ::/0                ::/0
¬        limit: avg 5/min burst 5 LOG flags 0 level 7 prefix `extIN-default:'
    0     0 DROP       all       *      *       ::/0                ::/0

Chain extOUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     tcp       *      *       ::/0
¬ 3ffe:ffff:100::1/128tcp spt:22 dpts:512:65535 flags:!0x16/0x02
    0     0 ACCEPT     tcp       *      *       ::/0
¬ 3ffe:ffff:100::2/128tcp spt:22 dpts:512:65535 flags:!0x16/0x02
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1024:65535 dpts:1:65535
    0     0 ACCEPT     udp       *      *       ::/0                ::/0
¬        udp spts:1024:65535 dpts:1:65535
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `extOUT-default:'
    0     0 DROP       all       *      *       ::/0                ::/0

Chain int2ext (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     icmpv6    *      *       ::/0                ::/0
    0     0 ACCEPT     tcp       *      *       ::/0                ::/0
¬        tcp spts:1024:65535 dpts:1:65535
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `int2ext:'
    0     0 DROP       all       *      *       ::/0                ::/0
    0     0 LOG        all       *      *       ::/0                ::/0
¬        LOG flags 0 level 7 prefix `int2ext-default:'
    0     0 DROP       tcp       *      *       ::/0                ::/0
    0     0 DROP       udp       *      *       ::/0                ::/0
    0     0 DROP       all       *      *       ::/0                ::/0

Chain intIN (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     all       *      *       ::/0
¬ fe80::/ffc0::
    4   384 ACCEPT     all       *      *       ::/0                ff02::/16
```

```
Chain intOUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
¬
    0     0 ACCEPT     all      *      *       ::/0
¬ fe80::/ffc0::
    4   384 ACCEPT     all      *      *       ::/0                 ff02::/16
    0     0 LOG        all      *      *       ::/0                 ::/0
¬        LOG flags 0 level 7 prefix `intOUT-default:'
    0     0 DROP       all      *      *       ::/0                 ::/0
```

Chapitre 17. La sécurité

17.1. La sécurité d'un noeud

Il est très recommandé d'appliquer tous les patchs disponibles, de rendre indisponibles tous les services inutiles, d'associer les services nécessaires aux adresses IPv4/IPv6 et d'installer un pare−feu local.

A remplir plus avant...

17.2. Les limitations d'accès

De nombreux services utilisent la bibliothèque tcp_wrapper pour contrôler l'accès. Plus bas est décrite l'utilisation de tcp_wrapper.

A remplir plus avant...

17.3. L'audit de sécurité IPv6

Actuellement, il n'existe pas d'outil véritablement adéquate aux questions de sécurité IPv6 et capable de vérifier un système monté sur le réseau. Ni Nessus ni aucun autre scanner de sécurité provenant du commerce n'est capable, autant que je sache, de scanner les adresses IPv6.

17.3.1. Question d'ordre légal

ATTENTION: Prenez bien garde d'uniquement scanner vos propres systèmes, ou alors seulement après avoir reçu une autorisation écrite, sinon des problèmes d'ordre juridique risquent de vous arriver. VERIFIER A DEUX FOIS les adresses IPv6 avant de lancer un scan.

17.3.2. Audit de sécurité par l'emploi de netcat disposant d'IPv6

Avec netcat disposant d'IPv6 (voir IPv6+Linux−status−apps/security−auditing pour en savoir plus), vous pouvez lancer un scan de ports *via* un script qui balaiera un intervalle de ports, captera des bannières, *etc*. Un exemple d'utilisation:

```
# nc6 ::1 daytime
13 JUL 2002 11:22:22 CEST
```

17.3.3. Audit de sécurité par l'emploi de nmap disposant d'IPv6

NMap, l'un des meilleurs scanners de ports à travers le monde, supporte IPv6 depuis la version 3.10ALPHA1. Un exemple d'utilisation:

```
# nmap −6 −sT ::1
Starting nmap V. 3.10ALPHA3 ( www.insecure.org/nmap/ )
Interesting ports on localhost6 (::1):
(The 1600 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
53/tcp     open        domain
```

```
515/tcp    open        printer
2401/tcp   open        cvspserver
Nmap run completed -- 1 IP address (1 host up) scanned in 0.525 seconds
```

17.3.4. Audit de sécurité par l'emploi de strobe disposant d'IPv6

Strobe est (comparé à NMap) un scanner de ports pour les petits budgets, mais il y a un patch disponible pour le rendre prêt pour IPv6 (voir IPv6+Linux−status−apps/security−auditing pour plus d'information). Un exemple d'utilisation:

```
# ./strobe ::1 strobe 1.05 (c) 1995-1999 Julian Assange <proff@iq.org>.
::1 2401 unassigned unknown
::1 22 ssh Secure Shell - RSA encrypted rsh
::1 515 printer spooler (lpd)
::1 6010 unassigned unknown
::1 53 domain Domain Name Server
```

Note: strobe n'est plus véritablement en développement, le numéro de version montré n'est pas le bon.

17.3.5. Le résultat de l'audit

Si le résultat de l'audit ne correspond pas à votre politique de sécurité IPv6, mettez en place le pare−feu IPv6 pour combler les trous de sécurité, par exemple en utilisant netfilter6 (voir Mettre en place un pare−feu grâce à Netfilter6 pour plus de détails).

Info: une information plus détaillée concernant la sécurité IPv6 peut être trouvée ici:

- Firewalling Considerations for IPv6 / draft−savola−v6ops−firewalling−??.txt
- IPv6 Neighbour Discovery trust models and threats
- Security Considerations for 6to4
- Access Control Prefix Router Advertisement Option for IPv6
- Requirements for Plug and Play IPsec for IPv6 applications
- Security of IPv6 Routing Header and Home Address Options

Chapitre 18. L'encryptage et l'authentification

A la différence d'IPv4, l'encryptage et l'authentification sont des fonctionnalités que ne fournit pas IPv6 lui−même. Elles sont normalement implémentées par l'utilisation d'IPsec (qui peut également être employé par IPv4).

18.1. Les modes d'emploi de l'encryptage et de l'authentification

Deux modes d'encryptage et d'authentification sont possibles:

18.1.1. Le mode transport

Le mode transport est un mode de connexion réellement de bout−en−bout. Ici, seule la charge utile (généralement ICMP, TCP ou UDP) est encryptée avec son en−tête particulier, tandis que l'en−tête IP n'est pas encrypté (mais couramment inclus dans l'authentification).

Utilisant AES−128 pour l'encryptage et SHA1 pour l'authentification, ce mode diminue la MTU de 42 octets.

18.1.2. Le mode tunnel

Le mode tunnel peut être utilisé soit dans un mode de connexion de bout−en−bout soit dans un mode de connexion de passerelle−à−passerelle. Ici, le paquet IP complet est encrypté et prend un nouvel en−tête IP, le tout constituant un nouveau paquet (ce mécanisme étant connu sous le nom d'encapsulation).

Cependant, à cause de l'indépendance de l'encryptage et de l'authentification à l'égard du protocole d'échange de clés, il existe actuellement des problèmes d'interopérabilité. Ce mode diminue actuellement de 40 octets par rapport au mode transport. Utiliser AES−128 pour l'encryptage et SHA1 pour l'authentification diminue donc au total de 82 octets la MTU courante.

18.2. Son support dans le noyau (ESP et AH)

18.2.1. Son support dans les noyaux Linux vanille 2.4.x

Manquant à ce jour jusqu'au noyau 2.4.28 vanille, le problème était de garder les sources du noyau Linux éloignées des questions de contrôles légaux d'import/export concernant le code d'encryptage en général. C'est une des raisons pour lesquelles le projet FreeS/WAN (IPsec pour IPv4 seulement) n'était pas compris dans les sources vanille. Un rétro−portage à partir de 2.6.x sera peut−être réalisé un jour.

18.2.2. Son support dans les noyaux Linux vanille 2.6.x

Les versions actuelles (2.6.9 et supérieures, au moment de la rédaction) supportent nativement IPsec pour IPv4 et IPv6.

Le projet USAGI a aidé à l'implémentation.

18.3. Echange automatique de clés (IKE)

IPsec requière un échange de clés afin de partager un secret. Ceci est essentiellement réalisé de façon automatisée par les démons IKE. Ils prennent également en charge l'authentification des entités en présence, soit par un secret commun (nommé "secret pré−partagé"), soit par clés RSA (qui peuvent provenir de certificats X.509).

Actuellement, deux démons IKE sont disponibles pour Linux, lesquels différent totalement par la configuration et l'emploi.

Je préfère "pluto" à l'implémentation *S/WAN à cause de son installation plus simple et à son unique fichier de configuration.

18.3.1. Le démon IKE "racoon"

Le démon IKE "racoon" provient du projet KAME et a été porté sur Linux. Les distributions contemporaines de Linux comportent ce démon dans le paquetage "ipsec−tools". Deux exécutables sont requis pour bien installer IPsec. Jetez aussi un oeil à Linux Advanced Routing & Traffic Control HOWTO / IPSEC.

18.3.1.1. Manipulation de la base de données IPsec SA/SP grâce à l'outil "setkey"

Le rôle important de "setkey" est de définir la politique de sécurité (*SP*, *security policy*) pour le noyau.

Fichier: /etc/racoon/setkey.sh

- Exemple d'une connexion encryptée de boute−en−bout en mode transport

```
#!/sbin/setkey -f
flush;
spdflush;
spdadd 2001:db8:1:1::1 2001:db8:2:2::2 any -P out ipsec esp/transport//require;
spdadd 2001:db8:2:2::2 2001:db8:1:1::1 any -P in  ipsec esp/transport//require;
```

- Exemple d'une connexion encryptée de boute−en−bout en mode tunnel

```
#!/sbin/setkey -f
flush;
spdflush;
spdadd 2001:db8:1:1::1 2001:db8:2:2::2 any -P out ipsec
¬ esp/tunnel/2001:db8:1:1::1-2001:db8:2:2::2/require;
spdadd 2001:db8:2:2::2 2001:db8:1:1::1 any -P in  ipsec
¬ esp/tunnel/2001:db8:2:2::2-2001:db8:1:1::1/require;
```

Pour l'autre machine, vous avez juste à échanger "in" et "out".

18.3.1.2. La configuration du démon IKE "racoon"

Pour sa bonne exécution, "racoon" requière d'être configuré. Ceci inclus les réglages relatifs à la politique de sécurité, qui doit être précédemment mise en place grâce à "setkey".

Fichier: /etc/racoon/racoon.conf

```
# Fichier de configuration du démon IKE racoon.
# Voir 'man racoon.conf' pour une description du format et des entrées.
path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
listen
            {
                    isakmp 2001:db8:1:1::1;
            }

remote 2001:db8:2:2::2
{
        exchange_mode main;
        lifetime time 24 hour;
        proposal
        {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group 2;
        }
}
# De passerelle-à-passerelle
sainfo address 2001:db8:1:1::1 any address 2001:db8:2:2::2 any
{
        lifetime time 1 hour;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
sainfo address 2001:db8:2:2::2 any address 2001:db8:1:1::1 any
{
        lifetime time 1 hour;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Fixez aussi un secret pré−partagé:

Fichier: /etc/racoon/psk.txt

```
# Fichier des clés pré-partagées utilisées pour l'authentification IKE
# Le format est: 'identificateur' 'clé'
2001:db8:2:2::2 absolumentsecret
```

18.3.1.3. Démarrer IPsec grâce au démon IKE "racoon"

Il faut pour le moins que le démon soit démarré. Au premier démarrage, utiliser les modes déboguage et premier plan (*debug and foreground*). L'exemple suivant montre une négociation IKE réussie dans ses phases 1 (ISAKMP−SA, *Internet Security Association Key Management Security Association*) and 2 (IPsec−SA, *IPsec Security Association*):

```
# racoon -F -v -f /etc/racoon/racoon.conf
Foreground mode.
2005-01-01 20:30:15: INFO: @(#)ipsec-tools 0.3.3 (http://ipsec-tools.sourceforge.net)
2005-01-01 20:30:15: INFO: @(#)This product linked
¬ OpenSSL 0.9.7a Feb 19 2003 (http://www.openssl.org/)
2005-01-01 20:30:15: INFO: 2001:db8:1:1::1[500] used as isakmp port (fd=7)
```

```
2005-01-01 20:31:06: INFO: IPsec-SA request for 2001:db8:2:2::2
¬ queued due to no phase1 found.
2005-01-01 20:31:06: INFO: initiate new phase 1 negotiation:
¬ 2001:db8:1:1::1[500]<=>2001:db8:2:2::2[500]
2005-01-01 20:31:06: INFO: begin Identity Protection mode.
2005-01-01 20:31:09: INFO: ISAKMP-SA established
¬ 2001:db8:1:1::1[500]-2001:db8:2:2::2[500] spi:da3d3693289c9698:ac039a402b2db401
2005-01-01 20:31:09: INFO: initiate new phase 2 negotiation:
¬ 2001:6f8:900:94::2[0]<=>2001:db8:2:2::2[0]
2005-01-01 20:31:10: INFO: IPsec-SA established:
¬ ESP/Tunnel 2001:db8:2:2::2->2001:db8:1:1::1 spi=253935531(0xf22bfab)
2005-01-01 20:31:10: INFO: IPsec-SA established:
¬ ESP/Tunnel 2001:db8:1:1::1->2001:db8:2:2::2 spi=175002564(0xa6e53c4)
```

Chaque direction a sa propre IPsec−SA (comme définie dans le standard IPsec). Avec "tcpdump" à l'écoute de la bonne interface, vous devriez voir comme résultat d'un ping IPv6:

```
20:35:55.305707 2001:db8:1:1::1 > 2001:db8:2:2::2: ESP(spi=0x0a6e53c4,seq=0x3)
20:35:55.537522 2001:db8:2:2::2 > 2001:db8:1:1::1: ESP(spi=0x0f22bfab,seq=0x3)
```

Comme prévu, les SPI (*Security Parameter Index*) négociés sont utilisés ici.

Et en utilisant "setkey", les paramètres actifs courants:

```
# setkey -D
2001:db8:1:1::1 2001:db8:2:2::2
        esp mode=tunnel spi=175002564(0x0a6e53c4) reqid=0(0x00000000)
        E: 3des-cbc  bd26bc45 aea0d249 ef9c6b89 7056080f 5d9fa49c 924e2edd
        A: hmac-md5  60c2c505 517dd8b7 c9609128 a5efc2db
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Jan  1 20:31:10 2005   current: Jan  1 20:40:47 2005
        diff: 577(s)    hard: 3600(s)   soft: 2880(s)
        last: Jan  1 20:35:05 2005      hard: 0(s)      soft: 0(s)
        current: 540(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 3    hard: 0 soft: 0
        sadb_seq=1 pid=22358 refcnt=0
2001:db8:2:2::2 2001:db8:1:1::1
        esp mode=tunnel spi=253935531(0x0f22bfab) reqid=0(0x00000000)
        E: 3des-cbc  c1ddba65 83debd62 3f6683c1 20e747ac 933d203f 4777a7ce
        A: hmac-md5  3f957db9 9adddc8c 44e5739d 3f53ca0e
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Jan  1 20:31:10 2005   current: Jan  1 20:40:47 2005
        diff: 577(s)    hard: 3600(s)   soft: 2880(s)
        last: Jan  1 20:35:05 2005      hard: 0(s)      soft: 0(s)
        current: 312(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 3    hard: 0 soft: 0
        sadb_seq=0 pid=22358 refcnt=0
```

18.3.2. Le démon IKE "pluto"

Le démon IKE "pluto" est inclus dans les distributions des projets *S/WAN, qui ont pour origine FreeS/WAN. Le développement du projet FreeS/WAN a malheureusement été stoppé en 2004. A cause de la lenteur du développement dans le passé, deux projets en découlèrent: strongSwan et Openswan. Aujourd'hui, des paquetages d'installation sont disponibles, au moins pour Openswan (inclus dans Fedora Core 3).

Une différence importante par rapport à "racoon", un seul et unique fichier de configuration est requis. Il y a

bien sûr un script d'initialisation qui automatise le lancement au démarrage de la machine.

18.3.2.1. La configuration du démon IKE "pluto"

La configuration est très similaire à celle nécessaire pour IPv4, à part une importante et nécessaire option.

Fichier: /etc/ipsec.conf

```
# /etc/ipsec.conf - Fichier de configuration d'IPsec Openswan
#
# Manuel:     ipsec.conf.5
version 2.0     # conforme à la seconde version de la spécification d'ipsec.conf
# configuration de base
config setup
        # Contrôles du déboguage / journalisation :  "none" pour (presque) rien, "all" pour beauc
        # klipsdebug=none
        # plutodebug="control parsing"
#Rendre indisponible l'encryptage opportuniste
include /etc/ipsec.d/examples/no_oe.conf
conn ipv6-p1-p2
        connaddrfamily=ipv6       # Important pour IPv6!
        left=2001:db8:1:1::1
        right=2001:db8:2:2::2
        authby=secret
        esp=aes128-sha1
        ike=aes128-sha-modp1024
        type=transport
        #type=tunnel
        compress=no
        #compress=yes
        auto=add
        #auto=start
```

N'oubliez pas ici également de définir un secret pré−partagé.

Fichier: /etc/ipsec.secrets

```
2001:db8:1:1::1 2001:db8:2:2::2 : PSK      "absolumentsecret"
```

18.3.2.2. Démarrer IPsec grâce au démon IKE "pluto"

Si l'installation d'Openswan s'est achevée avec succès, un script d'initialisation doit exister permettant le démarrage d'IPsec, lancez simplement (sur chaque machine) par:

```
# /etc/rc.d/init.d/ipsec start
```

Ensuite, démarrez une connexion sur l'une des machines. Si vous pouvez voir la ligne "IPsec SA established", c'est que tout fonctionne parfaitement.

```
# ipsec auto --up ipv6-peer1-peer2
104 "ipv6-p1-p2" #1: STATE_MAIN_I1: initiate
106 "ipv6-p1-p2" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "ipv6-p1-p2" #1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "ipv6-p1-p2" #1: STATE_MAIN_I4: ISAKMP SA established
112 "ipv6-p1-p2" #2: STATE_QUICK_I1: initiate
004 "ipv6-p1-p2" #2: STATE_QUICK_I2: sent QI2,
```

```
¬ IPsec SA established {ESP=>0xa98b7710 <0xa51e1f22}
```

Parce que *S/WAN et setkey/racoon sont basés sur la même implémentation d'IPsec dans les noyaux 2.6.x,"setkey" peut être utilisé pour afficher les paramètres actifs courants:

```
# setkey -D
2001:db8:1:1::1 2001:db8:2:2::2
        esp mode=transport spi=2844489488(0xa98b7710) reqid=16385(0x00004001)
        E: aes-cbc  082ee274 2744bae5 7451da37 1162b483
        A: hmac-sha1  b7803753 757417da 477b1c1a 64070455 ab79082c
        seq=0x00000000 replay=64 flags=0x00000000 state=mature
        created: Jan  1 21:16:32 2005   current: Jan  1 21:22:20 2005
        diff: 348(s)    hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=1 pid=23825 refcnt=0
2001:db8:2:2::2 2001:db8:1:1::1
        esp mode=transport spi=2770214690(0xa51e1f22) reqid=16385(0x00004001)
        E: aes-cbc  6f59cc30 8d856056 65e07b76 552cac18
        A: hmac-sha1  c7c7d82b abfca8b1 5440021f e0c3b335 975b508b
        seq=0x00000000 replay=64 flags=0x00000000 state=mature
        created: Jan  1 21:16:31 2005   current: Jan  1 21:22:20 2005
        diff: 349(s)    hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=0 pid=23825 refcnt=0
```

18.4. Informations complémentaires

Pour les noyaux Linux 2.6.x, vous pouvez également obtenir la politique et l'état d'IPsec en utilisant "ip":

```
# ip xfrm policy
...
# ip xfrm state
...
```

Chapitre 19. La Qualité de Service (QoS)

IPv6 supporte QoS par l'utilisation des labels de flux et des classes de trafic. Ceci peut être contrôlé en utilisant "tc" (compris dans le paquetage "iproute").

Information complémentaire:

- RFC 3697 / IPv6 Flow Label Specification

A remplir plus avant...

Chapitre 20. Eléments d'installation des démons prêts pour IPv6

Ici quelques éléments d'installation des démons prêts pour IPv6 sont exposés.

20.1. Le Démon de Nom Internet Berkeley (named)

IPv6 est supporté depuis la version 9. Utilisez toujours la dernière version disponible. Il faut au moins utiliser la version 9, les versions plus anciennes peuvent contenir des trous de sécurité exploitables à distance.

20.1.1. A l'écoute des adresses IPv6

Note: à la différence d'IPv4, les versions actuelles ne permettent pas d'associer une socket de serveur à des adresses IPv6 données, par conséquent, seule l'alternative *toutes* ou *aucune* adresse(s) IPv6 est valide. Parce que cela peut poser un problème de sécurité, consultez aussi plus bas la section concernant la liste de contrôle d'accès (ACL)!

20.1.1.1. Rendre disponible l'écoute sur adresse IPv6

Pour rendre disponible à named l'écoute IPv6, les options suivantes demandent à être modifiées

```
options {
        # certainement que d'autres options sont aussi ici
        listen-on-v6 { any; };
};
```

Il doit en résulter après redémarrage

```
# netstat -lnptu |grep "named\W*$"
tcp 0 0 :::53         :::*      LISTEN 1234/named
¬ # incoming TCP requests
udp 0 0 1.2.3.4:53    0.0.0.0:*        1234/named
¬ # incoming UDP requests to IPv4 1.2.3.4
udp 0 0 127.0.0.1:53  0.0.0.0:*        1234/named
¬ # incoming UDP requests to IPv4 localhost
udp 0 0 0.0.0.0:32868 0.0.0.0:*        1234/named
¬ # dynamic chosen port for outgoing queries
udp 0 0 :::53         :::*             1234/named
¬ # incoming UDP request to any IPv6
```

Un test simple ressemble à

```
# dig localhost @::1
```

et doit vous afficher un résultat.

20.1.1.2. Rendre indisponible l'écoute sur adresse IPv6

Pour rendre indisponible l'écoute IPv6, l'option suivante demande à être modifiée

```
options {
        # certainement que d'autres options sont aussi ici
```

```
        listen-on-v6 { none; };
};
```

20.1.2. Les Listes de Contrôle d'Accès IPv6 (ACL)

Les ACL IPv6 sont disponibles et devraient être utilisées dès que possible. Un exemple ressemble à ce qui suit:

```
acl internal-net {
        127.0.0.1;
        1.2.3.0/24;
        3ffe:ffff:100::/56;
        ::1/128;
        ::ffff:1.2.3.4/128;
};
acl ns-internal-net {
        1.2.3.4;
        1.2.3.5;
        3ffe:ffff:100::4/128;
        3ffe:ffff:100::5/128;
};
```

Ces ACL peuvent être utilisées par exemple pour les requêtes des clients ou pour le transfert de zones aux serveurs de noms de domaine secondaires. Ceci prévient aussi contre l'utilisation de votre serveur cache de noms de domaine à partir de l'extérieur grâce à IPv6.

```
options {
        # certainement que d'autres options sont aussi ici
        listen-on-v6 { none; };
        allow-query { internal-net; };
        allow-transfer { ns-internal-net; };
};
```

Il est aussi possible de positionner les options *allow−query* et *allow−transfer* pour la plupart des définitions par fichier de zone.

20.1.3. Emettre des requêtes avec une adresse IPv6 dédiée

Cette option n'est pas requise, mais peut être nécessaire:

```
query-source-v6 address <adresseipv6|*> port <port|*>;
```

20.1.4. Adresses IPv6 dédiées définies par zone

Il est aussi possible de définir des adresses IPv6 par zone.

20.1.4.1. Adresse de la source de transfert

L'adresse de la source de transfert est utilisée pour aller chercher les zones transférées:

```
transfer-source-v6 <adresseipv6|*> [port port];
```

20.1.4.2. Adresse de la source à notifier

L'adresse de la source à notifier est utilisée pour les messages de notification:

```
notify-source-v6 <adresseipv6|*> [port port];
```

20.1.5. Des exemples de fichiers de zone DNS IPv6

Des informations peuvent être aussi trouvées dans cet article concernant l'information d'installation d'un DNS IPv6. Le constructeur de zone inverse IPv6 pour BIND 8/9 (outil web) peut aussi être d'une aide précieuse.

20.1.6. Servir des données DNS relatives à IPv6

Pour IPv6, de nouveaux types et la zone racine nécessaire à la recherche inversée sont définis:

- AAAA et IP6.INT inversée: specifiés dans le RFC 1886 / DNS Extensions to support IP version 6, utilisés depuis BIND version 4.9.6
- A6, DNAME (DORÉNAVANT DÉPRECIÉ!) et IP6.ARPA inversé: spécifiés dans leRFC 2874 / DNS Extensions to Support IPv6 Address Aggregation and Renumbering, utilisable depuis BIND 9, mais vous pouvez trouver de l'information sur l'état actuel dans draft−ietf−dnsext−ipv6−addresses−00.txt

Peut−être complété plus tard, pour le moment, jetez un coup d'oeil aux RFC fournis et

- AAAA et IP6.INT inversé: l'installation d'un DNS IPv6
- A6, DNAME (DORÉNAVANT DÉPRECIÉ!) et IP6.ARPA inversé: jetez un coup d'oeil aux chapitres 4 et 6 du manuel de référence de l'administrateur BIND 9 (ARM), distribué avec le paquetage bind, ou bien récupérez−le : BIND version 9 ARM (PDF)

Parce que IP6.INT est déprécié (mais encore en usage), un serveur DNS qui supportera l'information IPv6 aura à servir tous les types de zones inversées.

20.1.6.1. La meilleure pratique courante

Parce qu'il y a encore quelques problèmes qui existent lorsque les nouveaux formats sont utilisés, la meilleure pratique courante est:

Support de recherche:

- AAAA

La recherche inversée supporte:

- Le format réduit inversé (*reverse nibble format*) pour la zone ip6.int (POUR LA COMPATIBILITÉ ASCENDANTE)
- Le format réduit inversé (*reverse nibble format*) pour la zone ip6.arpa (RECOMMANDÉ)

20.1.7. Vérifier la connectivité IPv6

Pour vérifier si BIND est à l'écoute sur une socket IPv6 et sert des données, voir les exemples suivants.

20.1.7.1. Connecté *via* IPv6, mais refusé par les ACL

En spécifiant un serveur pour les requêtes, une connexion IPv6 peut être forcée:

```
$ host -t aaaa www.6bone.net 3ffe:ffff:200:f101::1
Using domain server:
Name: 3ffe:ffff:200:f101::1
Address: 3ffe:ffff:200:f101::1#53
Aliases:
Host www.6bone.net. not found: 5(REFUSED)
```

L'entrée relative dans le journal ressemble à ce qui suit:

```
Jan 3 12:43:32 gate named[12347]: client
¬ 3ffe:ffff:200:f101:212:34ff:fe12:3456#32770:
 query denied
```

Si vous observez de telles entrées dans le journal, vérifiez si les requêtes provenant de ce client doivent être autorisées, pour revoir, si nécessaire, votre configuration ACL.

20.1.7.2. Une connexion IPv6 réussie

Une connexion IPv6 réussie ressemble à ce qui suit:

```
$ host -t aaaa www.6bone.net 3ffe:ffff:200:f101::1
Using domain server:
Name: 3ffe:ffff:200:f101::1
Address: 3ffe:ffff:200:f101::1#53
Aliases:
www.6bone.net. is an alias for 6bone.net.
6bone.net. has AAAA address 3ffe:b00:c18:1::10
```

20.2. Le super démon Internet (xinetd)

IPv6 est supporté, approximativement, depuis la version 1.8.9 de xinetd. Utilisez toujours la version disponible la plus récente. Seules les versions antérieures à la version 2.3.3 doivent être utilisées, les versions plus anciennes peuvent contenir des trous de sécurité exploitables à distance.

Certaines distributions Linux contiennent un paquetage supplémentaire pour xinetd prêt pour IPv6, d'autres démarrent xinetd prêt pour IPv6 si la variable suivante est positionnée: NETWORKING_IPV6="yes", chose normalement réalisée par /etc/sysconfig/network (valide uniquement pour la distribution Red Hat et ses dérivées). Dans les nouvelles livraisons des distributions, un binaire supporte à la fois IPv4 et IPv6.

Si vous rendez disponible un service fourni avec xinetd, comme par exemple daytime, en modifiant la configuration dans le fichier /etc/xinetd.d/daytime comme suit

```
# diff -u /etc/xinetd.d/daytime.orig /etc/xinetd.d/daytime
--- /etc/xinetd.d/daytime.orig Sun Dec 16 19:00:14 2001
+++ /etc/xinetd.d/daytime Sun Dec 16 19:00:22 2001
```

```
@@ -10,5 +10,5 @@
        protocol = tcp
        user = root
        wait = no
-       disable = yes
+       disable = no
 }
```

vous devriez recevoir, après le redémarrage de xinetd, une réponse positive telle que:

```
# netstat -lnptu -A inet6 |grep "xinetd*"
tcp 0 0 ::ffff:192.168.1.1:993  :::*  LISTEN  12345/xinetd-ipv6
tcp 0 0 :::13                   :::*  LISTEN  12345/xinetd-ipv6 <- service
¬ daytime/tcp
tcp 0 0 ::ffff:192.168.1.1:143  :::*  LISTEN  12345/xinetd-ipv6
```

L'exemple montre aussi que xinetd écoute pour IMAP et IMAP−SSL sur IPv4 seulement.

Note: un serveur xinetd uniquement IPv4 ne démarrera pas sur un noeud disposant d'IPv6 et inversement, un serveur xinetd IPv6 ne démarrera pas sur un noeud uniquement IPv4. Ce problème est réputé réglé dans les versions postérieures, au moins à partir de la version 2.3.11.

20.3. Le serveur web Apache2 (httpd2)

Le serveur web Apache supporte nativement IPv6 depuis la version 2.0.14. Des patchs disponibles pour l'ancienne série 1.3.x ne sont pas courants et ne devraient pas être employés dans un contexte public, mais ils sont disponibles sur ce serveur ftp, KAME / Misc.

20.3.1. A l'écoute sur les adresses IPv6

Note: Les hôtes virtuels sur adresses IPv6 ne fonctionnent pas pour les versions inférieures à la 2.0.28 (un patch est disponible pour la 2.0.28). Mais en tout premier lieu, récupérez toujours la dernière version disponible, parce que les premières versions ont des problèmes de sécurité.

20.3.1.1. Un hôte virtuel écoute sur une adresse IPv6 uniquement

```
Listen [3ffe:ffff:100::1]:80
<VirtualHost [3ffe:ffff:100::1]:80>
        ServerName ipv6seul.votredomaine.votretld
        # certainement des lignes de configuration en plus...
</VirtualHost>
```

20.3.1.2. Un hôte virtuel écoute sur une adresse IPv6 et sur une adresse IPv4

```
Listen [3ffe:ffff:100::2]:80
Listen 1.2.3.4:80
<VirtualHost [3ffe:ffff:100::2]:80 1.2.3.4:80>
        ServerName ipv6etipv4.votredomaine.votretld
        # certainement des lignes de configuration en plus...
</VirtualHost>
```

Il devrait en résulter après redémarrage

```
# netstat -lnptu | grep "httpd2\W*$"
tcp 0 0 1.2.3.4:80          0.0.0.0:* LISTEN 12345/httpd2
tcp 0 0 3ffe:ffff:100::1:80 :::*      LISTEN 12345/httpd2
tcp 0 0 3ffe:ffff:100::2:80 :::*      LISTEN 12345/httpd2
```

Pour de simples tests, utiliser l'exemple telnet déjà montré.

20.3.1.3. Note additionnelle

- D'une part, Apache2 supporte une méthode appelée "sendfile", accélérant la fourniture des données. D'autre part, certains pilotes de NIC supportent la vérification différée des sommes de contrôle (*offline checksumming*). Dans certains cas, cela peut conduire à des problèmes de connexion et invalider les sommes de contrôle TCP. Il faut alors rendre indisponible "sendfile", ou bien en recompilant par l'utilisation de l'option de configure "−−without−sendfile", ou bien en utilisant la directive du fichier de configuration "EnableSendfile off".

20.4. Le Démon d'Annonce de Routeur (radvd)

Le Démon d'Annonce de Routeur est très utile sur un LAN, à partir du moment où les clients doivent être auto−configurés. Le démon lui−même doit tourner sur la passerelle par défaut IPv6 Linux (il n'est pas requis qu'elle soit aussi la passerelle IPv4, aussi prenez garde à qui émet des annonces de routeur sur votre LAN).

Vous avez à spécifier certaines informations et drapeaux qui doivent être compris dans l'annonce. Les plus employés sont

- Le préfixe (nécessaire)
- La durée de vie du préfixe
- La fréquence des envois d'annonce (optionnelle)

Après une configuration convenable, le démon émet des annonces au travers des interfaces spécifiées, dans l'espoir que les clients les reçoivent et auto−configurent comme par magie leurs adresses avec le préfixe reçu et le routeur par défaut.

20.4.1. Configurer radvd

20.4.1.1. Configuration simple

Le fichier de configuration de radvd est généralement /etc/radvd.conf. Un exemple simple ressemble à ce qui suit:

```
interface eth0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 3ffe:ffff:0100:f101::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

Ce qui a pour résultat côté client

```
# /sbin/ip -6 addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 3ffe:ffff:100:f101:2e0:12ff:fe34:1234/64 scope global dynamic
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::2e0:12ff:fe34:1234/10 scope link
```

Parce qu'aucune limite de vie n'a été définie, une très grande valeur est utilisée.

20.4.1.2. Configuration spéciale 6to4

Les versions à partir de la 0.6.2pl3 supportent la (ré)génération des préfixes dépendant d'une adresse IPv4 propre à une interface spécifique. Ceci peut être utilisé afin de distribuer les annonces dans un LAN après que le tunnelage 6to4 ait changé. Surtout employé derrière un routeur de connexion dynamique à la demande (*dial−on−demand*). Avec l'assurance d'un temps de vie très bref pour un tel préfixe (après chaque reconnexion*, dial−up*, un autre préfixe est valide), la durée de vie est configurée aux valeurs minimales:

```
interface eth0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 0:0:0:f101::/64 {
                AdvOnLink off;
                AdvAutonomous on;
                AdvRouterAddr on;
                Base6to4Interface ppp0;
                AdvPreferredLifetime 20;
                AdvValidLifetime 30;
        };
};
```

Il en résulte pour le client situé à l'intérieur (en considérant que ppp0 a actuellement 1.2.3.4 comme adresse IPv4 locale):

```
# /sbin/ip -6 addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
   inet6 2002:0102:0304:f101:2e0:12ff:fe34:1234/64 scope global dynamic
      valid_lft 22sec preferred_lft 12sec
   inet6 fe80::2e0:12ff:fe34:1234/10 scope link
```

Parce qu'un bref temps de vie a été défini, un tel préfixe sera rapidement jeté si aucune annonce s'y rapportant n'est reçue.

Note additionnelle: si vous n'utilisez pas de support spécifique 6to4 dans les scripts d'initialisation, vous avez à mettre en place une route spécifique sur l'interface interne du routeur, sinon vous aurez des problèmes de routage retour. En ce qui concerne l'exemple ci−après:

```
# /sbin/ip -6 route add 2002:0102:0304:f101::/64 dev eth0 metric 1
```

Cette route a besoin d'être remplacée chaque fois que le préfixe change, ce qui est le cas à chaque fois qu'une nouvelle adresse IPv4 est assignée à une interface numérotée automatiquement (*dial−up interface*).

20.4.2. Le déboguage

Un programme appelé "radvdump" peut vous aider à observer les annonces émises ou reçues. Simple à utiliser:

```
# radvdump
Router advertisement from fe80::280:c8ff:feb9:cef9 (hoplimit 255)
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        Prefix 2002:0102:0304:f101::/64
                AdvValidLifetime: 30
                AdvPreferredLifetime: 20
                AdvOnLink: off
                AdvAutonomous: on
                AdvRouterAddr: on
        Prefix 3ffe:ffff:100:f101::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
                AdvAutonomous: on
                AdvRouterAddr: on
        AdvSourceLLAddress: 00 80 12 34 56 78
```

Cette sortie vous montre que chaque paquetage d'annonce se présente dans un format lisible. Vous devriez voir à nouveau ici vos valeurs configurées, sinon, ce n'est peut−être pas votre radvd qui émet l'annonce... vérifiez s'il n'y a pas un autre routeur sur le lien (pour traquer ce routeur, prenez l'adresse LL, AdvSourceLLAddress, qui est son adresse MAC).

20.5. Le serveur de Configuration Dynamique d'Hôte v6 (dhcp6s)

DHCPv6 peut être utilisé pour réaliser des configurations avec état. Le démon par lui−même n'a pas nécessairement à tourner sur la passerelle routeur Linux par défaut.

Vous pouvez spécifier plus d'information qu'avec radvd. Il est plus similaire à un serveur DHCP IPv4.

Après une configuration correcte, le démon réagit lors de la réception de paquets multicast envoyés par un client à l'adresse ff02::16.

20.5.1. La configuration du serveur DHCPv6 (dhcp6s)

20.5.1.1. Une configuration simple

Le fichier de configuration de dhcp6s est normalement /etc/dhcp6s.conf. Un exemple simple ressemble à ce qui suit:

```
interface eth0 {
        server-preference 255;
        renew-time 60;
        rebind-time 90;
        prefer-life-time 130;
        valid-life-time 200;
```

```
        allow rapid-commit;
        option dns_servers 2001:db8:0:f101::1 sub.domain.example;
        link AAA {
                range 2001:db8:0:f101::1000 to 2001:db8:0:f101::ffff/64;
                prefix 2001:db8:0:f101::/64;
        };
};
```

20.5.2. La configuration du client DHCPv6 (dhcp6c)

20.5.2.1. Une configuration simple

Le fichier de configuration de dhcp6c est normalement /etc/dhcp6c.conf. Un exemple simple ressemble à ce qui suit:

```
interface eth0 {
        send rapid-commit;
        request domain-name-servers;
};
```

20.5.3. Usage

20.5.3.1. dhcpv6_server

Démarrage du serveur,

```
# service dhcp6s start
```

20.5.3.2. dhcpv6_client

Démarrage du client en premier plan,

```
# dhcp6c -f eth0
```

20.5.4. Déboguage

20.5.4.1. dhcpv6_server

Le serveur a un drapeau pour passer en premier plan et deux pour le déboguage (tout deux devraient être utilisés pour le déboguage). Voici un exemple:

```
# dhcp6c -d -D -f eth0
```

20.5.4.2. dhcpv6_client

Le client a un drapeau pour passer en premier plan et deux pour le déboguage. Voici un exemple:

```
   # dhcp6c -d -f eth0
   Oct/03/2005 17:18:16 dhcpv6 doesn't support hardware type 776
```

```
   Oct/03/2005 17:18:16 doesn't support sit0 address family 0
   Oct/03/2005 17:18:16 netlink_recv_rtgenmsg error
   Oct/03/2005 17:18:16 netlink_recv_rtgenmsg error
   Oct/03/2005 17:18:17 status code for this address is: success
   Oct/03/2005 17:18:17 status code: success
   Oct/03/2005 17:18:17 netlink_recv_rtgenmsg error
   Oct/03/2005 17:18:17 netlink_recv_rtgenmsg error
   Oct/03/2005 17:18:17 assigned address 2001:db8:0:f101::1002 prefix len isnot    in any RAs pre
   Oct/03/2005 17:18:17 renew time 60, rebind time 9
```

Notez que les messages d'erreur n'ont aucun impact.

20.6. tcp_wrapper

tcp_wrapper est une bibliothèque qui peut vous aider à protéger vos services contre les usages abusifs.

20.6.1. Les capacités de filtrage

Vous pouvez utiliser tcp_wrapper pour

- Le filtrage à partir des adresses source (IPv4 ou IPv6)
- Le filtrage des utilisateurs (réclame un démon ident en état de fonctionnement sur le client)

20.6.2. Les programmes utilisant tcp_wrapper

Les suivants sont connus:

- Chaque service appelé par xinetd (si xinetd est compilé en utilisant la bibliothèque tcp_wrapper)
- sshd (si compilé en utilisant tcp_wrapper)

20.6.3. Utilisation

tcp_wrapper est contrôlé par deux fichiers nommés /etc/hosts.allow et /etc/hosts.deny. Pour plus d'information voir

```
$ man hosts.allow
```

20.6.3.1. Exemple de fichier /etc/hosts.allow

Dans ce fichier, chaque service qui doit être positivement filtré (*i.e.,* dont les connexions doivent être acceptées) a besoin d'une ligne.

```
sshd:           1.2.3. [3ffe:ffff:100:200::]/64
daytime-stream: 1.2.3. [3ffe:ffff:100:200::]/64
```

Note: ils existent des implémentations cassées qui utilisent la mauvaise description réseau IPv6 suivante: [3ffe:ffff:100:200::/64].

Heureusement, de telles versions seront rapidement corrigées.

20.6.3.2. Exemple de fichier /etc/hosts.deny

Ce fichier contient toutes les entrées de filtre négative et devrait dénier l'accès à tout le reste en utilisant

```
ALL: ALL
```

Si ce noeud est très sensible, vous pouvez remplacer la ligne stantard ci−dessus par celle−ci, mais cela peut être cause d'attaque DoS (charge de serveur de courrier et répertoire spool), si trop de connexions sont réalisées en un temps très bref. Peut−être un observateur de journaux (*a logwatch*) serait−il meilleur dans de tels cas.

```
ALL: ALL: spawn (echo "Attempt from %h %a to %d at `date`"
 | tee -a /var/log/tcp.deny.log | mail root@localhost)
```

20.6.4. La journalisation

Selon l'entrée du fichier de configuration du démon syslog /etc/syslog.conf, la journalisation de tcp_wrapper se fait normalement dans /var/log/secure.

20.6.4.1. Connexion refusée

Une connexion refusée *via* IPv4 au service daytime, couvert par xinetd, produit des lignes telles que celles de l'exemple suivant

```
Jan 2 20:40:44 gate xinetd-ipv6[12346]: FAIL: daytime-stream libwrap
¬ from=::ffff:1.2.3.4
Jan 2 20:32:06 gate xinetd-ipv6[12346]: FAIL: daytime-stream libwrap
 from=3ffe:ffff:100:200::212:34ff:fe12:3456
```

Une connexion refusée *via* IPv4 à sshd en double écoute produit des lignes telles que celles de l'exemple suivant

```
Jan 2 20:24:17 gate sshd[12345]: refused connect from ::ffff:1.2.3.4
¬ (::ffff:1.2.3.4)
Jan 2 20:39:33 gate sshd[12345]: refused connect
 from 3ffe:ffff:100:200::212:34ff:fe12:3456
¬ (3ffe:ffff:100:200::212:34ff:fe12:3456)
```

20.6.4.2. Connexion autorisée

Une connexion autorisée *via* IPv4 vers le service daytime, couvert par xinetd, produit des lignes telles que celles de l'exemple suivant

```
Jan 2 20:37:50 gate xinetd-ipv6[12346]: START: daytime-stream pid=0
¬ from=::ffff:1.2.3.4
Jan 2 20:37:56 gate xinetd-ipv6[12346]: START: daytime-stream pid=0
 from=3ffe:ffff:100:200::212:34ff:fe12:3456
```

Une connexion autorisée *via* IPv4 vers sshd en double écoute produit des lignes telles que celles de l'exemple suivant

```
```

```
Jan 2 20:43:10 gate sshd[21975]: Accepted password for user from ::ffff:1.2.3.4
¬ port 33381 ssh2
Jan 2 20:42:19 gate sshd[12345]: Accepted password for user
 from 3ffe:ffff:100:200::212:34ff:fe12:3456 port 33380 ssh2
```

20.7. vsftpd

20.7.1. A l'écoute des adresses IPv6

Editer le fichier de configuration, couramment /etc/vsftpd/vsftpd.conf, et ajuster l'option "listen"

```
listen_ipv6=yes
```

C'est tout.

20.8. proftpd

20.8.1. A l'écoute des adresses IPv6

Editer le fichier de configuration, couramment /etc/proftpd.conf, mais prenez garde, tout n'est pas tout à fait logique dans la mise en place des hôtes virtuels

```
<VirtualHost 192.0.2.1>
        ...
        Bind 2001:0DB8::1
        ...
</VirtualHost>
```

C'est tout.

20.9. Autres démons

De nos jours c'est généralement simple, cherchez une ligne de commande d'option ou bien une valeur de configuration pour rendre disponible l'écoute IPv6. Cherchez dans la page du manuel du démon ou dans les FAQ concernées. Il peut arriver que vous ne puissiez lier le démon qu'à une adresse IPv6 "any" (::) et pas à une adresse IPv6 précise, par manque de support (ça dépend de la façon dont le développeur à implémenter...).

Chapitre 21. Programmer (en utilisant l'API)

Je n'ai aucune expérience de la programmation IPv6, peut−être que ce chapitre sera rempli par d'autres, ou déplacé vers un autre HOWTO

Plus d'information peut être trouvée ici:

- RFC 2553 / Basic Socket Interface Extensions for IPv6
- Draft / Advanced Sockets API for IPv6 / draft−ietf−ipngwg−rfc2292bis−XY.txt

Porting applications to IPv6 HowTo by Eva M. Castro

Chapitre 22. L'interopérabilité

Il y a à travers le monde quelques projets qui vérifient l'interopérabilité des différents systèmes d'exploitation vis−à−vis de l'implémentation des fonctionnalités d'IPv6: Voici un URL:

- Le projet TAHI

D'autres arriveront prochainement...

Chapitre 23. Plus d'information et d'URL

23.1. Livres en édition papier, articles, revues en ligne (mélangés)

23.1.1. Livres édités (en anglais)

23.1.1.1. Cisco

- Cisco Self−Study: Implementing IPv6 Networks (IPV6), par Regis Desmeules. Cisco Press; ISBN 1587050862; 500 pages; 1ère édition (11 avril 11 2003). Note: cet ouvrage sera publié le 11 avril 2003.
- Configuring IPv6 with Cisco IOS, par Sam Brown, Sam Browne, Neal Chen, Robbie Harrell, Edgar, Jr. Parenti (Editeur), Eric Knipp (Editeur), Paul Fong (Editeur) 362 pages; Syngress Media Inc; ISBN 1928994849; (12 juillet 2002).

23.1.1.2. Généraux

- IPv6 Essentials par Silvia Hagen, juillet 2002, O'Reilly, référence pour la commande: 1258, ISBN 0−5960−0125−8, 352 pages. Table des matières, index, exemple de chapitre, etc.; Les derniers livres sortis chez O'Reilly
- IPv6: The New Internet Protocol. Par Christian Huitema; Publié chez Prentice−Hall; ISBN 0138505055. Description: Ce livre, écrit par Christian Huitema − membre du Comité Architecture Internet (*Internet Architecture Board*, ou IAB), offre une excellente description d'IPv6, de ses différences d'avec IPv4, du comment et du pourquoi de son développement. Source: http://www.cs.uu.nl/wais/html/na−dir/internet/tcp−ip/resource−list.html
- IPv6 Networks par Niles, Kitty; (ISBN 0070248079); 550 pages; Date de publication 05/01/1998.
- Implementing IPV6. Supporting the Next Generation Internet Protocols par P. E. Miller, Mark A. Miller; éd. John Wiley & Sons; ISBN 0764545892; 2ème édition (15 mars 2000); 402 pages.
- Big Book of Ipv6 Addressing Rfcs par Peter H. Salus (compilateur), Morgan Kaufmann Publishers, avril 2000, 450 pages, ISBN 0126167702.
- Understanding IPV6 par Davies, Joseph; ISBN 0735612455; Date de publication 01/05/2001; 350 pages. Understanding IPV6 par Davies, Joseph; ISBN 0735612455; Date de publication 11/13/2002; 544 pages.
- Migrating to IPv6 − IPv6 in Practice. par Marc Blanchet; éd. John Wiley & Sons; ISBN 0471498920; 1ère édition (novembre 2002); 368 pages.
- Ipv6 Network Programming par Jun−ichiro Hagino; ISBN 1555583180
- Wireless boosting IPv6 par Carolyn Duffy Marsan, 23/10/2000.
- la recherche avec le mot clé IPv6 sur O'reilly réseau donne 29 résultats (au 28 Janvier 2002).

23.1.2. Livres édités (en allemand)

- Technik der IP−Netze (TCP/IP incl. IPv6) bei Amazon.de Anatol Badach, Erwin Hoffmann Carl Hanser Verlag München, Wien, 2001 ISBN 3−446−21501−8 Kap. 6: Protokoll IPv6 S.205−242 Kap. 7: Plug&Play−Unterstützung bei IPv6 S.243−276 Kap. 8: Migration zum IPv6−Einsatz S.277−294 Kap. 9.3.4: RIP für das Protokoll IPv6 (RIPng) S.349−351 Kap. 9.4.6: OSPF für IPv6 S.384−385 Kommentar: tw. nicht ganz up−to−date bzw. nicht ganz fehlerfreie Abbildungen Homepage des Buches und Tabelle mit Fixes
- Internet−Sicherheit (Browser, Firewalls und Verschlüsselung) bei Amazon.de Kai Fuhrberg 2. akt. Auflage 2000 Carl Hanser Verlag München, Wien, ISBN 3−446−21333−3 Kap.2.3.1.4. IPv6 S.18−22 Kurz angerissen werden: RFC 1825 − Security Association Konzept RFC1826 − IP authentication

Header RFC 1827 – IP Encapsulation Security Payload
- IPv6. Das neue Internet– Protokoll. Technik, Anwendung, Migration bei Amazon Hans Peter Dittler 2. akt. und erweiterte Auflage 2002 dpunkt.verlag, ISBN 3–89864–149–X
- Das neue Internetprotokoll IPv6 bei Amazon Herbert Wiese 2002 Carl Hanser Verlag, ISBN 3446216855

23.1.3. Articles, livres électroniques, revues en ligne (mélangés)

- Getting Connected with 6to4 par Huber Feyrer, 01/06/2001
- Transient Addressing for Related Processes: Improved Firewalling by Using IPv6 and Multiple Addresses per Host; écrit par Peter M. Gleiz, Steven M. Bellovin (version PDF pour PC; version PDF pour Palm; version PDB)
- IPv6, théorie et pratique (en français) 3ème édition, mars 2002, O'Reilly, ISBN 2–84177–139–3
- IPSec (en langue française)
- Internetworking IPv6 with Cisco Routers par Silvano Gai, McGrawHill Italia, 1997. Le chapitre 13 et les appendices A–D sont téléchargables au format PDF.
- Secure and Dynamic Tunnel Broker par Vegar Skaerven Wang, thèse de maîtrise en sciences informatiques, 2 Juin 2000, Faculté des Sciences, Département des sciences informatiques, Université de Tromso, Norvège.
- Aufbruch in die neue Welt – IPv6 in IPv4 Netzen par Dipl.Ing. Ralf Döring, TU Illmenau, 1999
- Migration and Co–existence of IPv4 and IPv6 in Residential Networks par Pekka Savola, CSC/FUNET, 2002

23.1.4. Publications scientifiques (résumés, bibliographies, ressources en ligne)

- Plan de travail du projet IPv6 GEANT
- A simulation study on the performance of Mobile IPv6 in a WLAN–based cellular network, par Perez Costa X.; Hartenstein H. –– Computer Networks, Septembre 2002, vol. 40, no. 1, pp. 191–204(14) –– Elsevier Science.
- Tests IPv6 sur le réseau universitaire britannique: Projet Bermudes 2 Août 2002: Participants – Se connecter – Project deliverables – Network topology – adresse assignments – Wireless IPv6 access – IPv6 migration – Project presentations – Internet 2 – Other IPv6 projects – IPv6 fora and standards Bermuda 2...
- http://www.ipv6.ac.uk/
- A scalable parallel internet router that enables the QoS through merging ATM with IPv6. par Song S. –– Computer Communications, 1 mai 2002, vol. 25, no. 7, pp. 647–651(5) –– Elsevier Science.
- Linux IPv6: Which One to Deploy? journal Linux, Vol. 96, p. 86, 88–90, April 2002. (pour plus d'information, voir aussiliinwww.ira.uka.de/ipv6)
- An overview and analysis of mobile Internet protocols in cellular environments. Chao H–C. –– Internet Research: Electronic Networking Applications and Policy, 24 Octobre 2001, vol. 11, no. 5, pp. 435–450(16) –– MCB University Press
- IPv6 for Future Wireless networksToftegaard Nielsen T. –– Wireless Personal Communications, Juin 2001, vol. 17, no. 2/3, pp. 237–247(11) –– éd. Kluwer Academic, Dordrecht, Les Pays–Bas
- IPv6 at the University of Southampton
- Seamless Support for Mobile Internet Protocol Based Cellular EnvironmentsChao H–C.; Chu Y–M. –– International Journal of Wireless Information Networks, Juillet 2001, vol. 8, no. 3, pp. 133–153(21) –– éd. Kluwer Academic/Plenum, New York, U.S.A.
- IPv6: The Solution for Future Universal Networks. Lecture Notes in Computer Science, Vol. 1818, p. 82–??, 2000.
- Modeling and performance analysis for IPv6 traffic with multiple QoS classes. Zhang L.; Zheng L.

−− Computer Communications, 1 octobre 2001, vol. 24, no. 15, pp. 1626−1636(11) −− Elsevier Science.
- Threshold−Based Registration (TBR) in Mobile IPv6. Lecture Notes in Computer Science, Vol. 1818, p. 150−??, 2000.
- IPv6 Performance Analysis on FreeBSD Workstation Using Simple Applications. Lecture Notes in Computer Science, Vol. 1961, p. 33−??, 2000.
- Microsoft Research IPv6 Implementation (MSRIPv6): MSRIPv6 Configuring 6to4 – Connectivity with MSR IPv6 – Our 6Bone Node...
- New frontiers in cybersegmentation: marketing success in cyberspace depends on IP address. Louvieris P.; Driver J. −−Qualitative Market Research: An International Journal, 27 juin 2001, vol. 4, no. 3, pp. 169−181(13) −− MCB University Press.
- QoS−Conditionalized Handoff for Mobile IPv6. Notes de conférence en sciences informatiques, Vol. 2345, p. 721−??, 2002.

23.1.5. Autres

Voir l'URL suivant pour en savoir plus: SWITCH Pilote IPv6 Pilot / Références

23.2. Conférences, rencontres, sommets

23.2.1. 2002

- Renater – Conférence IPv6 2002
- Sommet déploiement IPv6 à INET 2002

Quelque chose manque? Les suggestions sont les bienvenues!

23.2.2. 2003

Les suggestions sont les bienvenues!

23.3. L'information en ligne

23.3.1. Rejoindre le backbone IPv6

A remplir plus avant et plus tard... Les suggestions sont les bienvenues!

23.3.1.1. Les bureaux d'enregistrement global

- Backbone de test IPv6: 6bone, Comment rejoindre 6bone, Teilnahme am 6bone (en langue allemande), La participation au 6bone (en langue anglaise)

23.3.1.2. Les centres d'enregistrement de noms de domaine les plus importants, par régions

- Amérique: ARIN, ARIN / page d'enregistement, ARIN / IPv6 guidelines
- EMEA: Ripe NCC, Ripe NCC / page d'enregistrement, Ripe NCC / IPv6 registration
- Asie/Pacifique: APNIC, APNIC / guide de ressources IPv6
- Amérique latine et les Caraïbes: LACNIC, Service d'enregistrement IPv6, Politique d'allocation IPv6
- Afrique: AfriNIC

Il existe aussi une liste des principales allocations (préfixe de 32 bits) par bureau d'enregistrement régional ici: Ripe NCC / allocations IPv6.

23.3.1.3. Les fournisseurs de tunnel (*tunnel brokers*)

Note: une liste de fournisseurs de tunnel peut être trouvée plus bas dans l'information concernant les fournisseurs de tunnel.

- Code source utilisé dans une thèse de maîtrise dans le cadre du projet Vermicelli au sujet des fournisseurs de tunnels, Université de Tromso.
- Fondation IPng. Fournisseurs de tunnel et ressources IPv6, dorénavant migré au système SixXs.
- La page de Eckes Linux−avec−IPv6.
- tunnelc − client de tunnelage basé sur perl: freshmeat.net: détails sur ce client de tunnelage SourceForge: Projet Info − tunnelc (aussi ici)
- L'HOWTO Routage avancée Linux & contrôle du trafic, Chapitre 6: Le tunnelage IPv6 avec Cisco et/ou 6bone.

Voir aussi ici pour plus d'information et d'URL: ipv6−net.org.

23.3.1.4. 6to4

- information 6to4 de NSayer
- RFC 3068 / An Anycast Prefix for 6to4 Relay Routers

23.3.1.5. ISATAP

- ISATAP (Intra−Site Automatic Tunnel Access Protocol) Information byJOIN

23.3.2. Les dernières nouvelles

A remplir plus avant et plus tard... Les suggestions sont les bienvenues!

- ipv6−net.org, est aussi la page d'accueil du canal #IPv6 sur EFnet
- Nombreux URLs vers d'autres documents par Anil Edathara

23.3.3. Les références aux protocoles

23.3.3.1. Les appels à commentaires (RFC) relatifs à IPv6

La publication de la liste des RFC relatifs à IPv6 outrepasse la portée de ce document, mais les URL fournis vous guiderons vers de telles listes:

- Listes classées par Etat de la standardisation IPng ou Spécifications actuelles d'IPng par Robert Hinden
- Spécifications relatives à IPv6 on IPv6.org

23.3.3.2. Les brouillons actuels des groupes de travail

Les brouillons actuels concernant (aussi) IPv6 peuvent être trouvés ici:

- IP Version 6 (ipv6)

- Transition vers la nouvelle génération (ngtrans)
- Dynamic Host Configuration (dhc)
- Extension du Système des Noms de Domaine
- Mobile IP (mobileip)
- Obtenir toute l'information à propos d'IPv6, depuis des vues d'ensemble, en passant par les brouillons et les RFC, jusqu'aux implémentations (comprenant la disponibilité de la pile sur différentes plates−formes & le code source de la pile IPv6)

23.3.3.3. Autres

- Network Sorcery / IPv6, Protocole Internet IP version 6, l'en−tête du protocole IPv6
- Guide / Références IPv6 SWITCH, importante liste de références IPv6, maintenue par Simon Leinen

---

23.3.4. Plus d'information

A remplir plus avant et plus tard... les suggestions sont les bienvenues!

DeepSpace6 / plus de liens intéressants

23.3.4.1. Relative à Linux

- DeepSpace6 / Portail Linux IPv6 (pas uniquement) − Italie (miroir)
- IPv6−HowTo pour Linux par Peter Bieringer − Allemagne et son archive logiciel − Bieringer / IPv6
- L'état de Linux+IPv6 par Peter Bieringer − Allemagne (en cours d'obsolescence)
- DeepSpace6 / La page concernant l'état IPv6 − Italie (miroir) (remplacera la page ci−dessus)
- Projet USAGI − Japon, et son archive logiciel − projet USAGI
- HOWTO IPv6 Protocole de routage à état de lien optimisé (OLSR)

23.3.4.2. Relative à Linux, par distribution

PLD
: PLD Linux Distribution ("leader du marché" quant aux paquetages disposant d'IPv6)

Red Hat
: Linux Red Hat, les paquetages IPv6 de Pekka Savola

Debian
: Linux Debian, Etat et information IPv6 par Craig Small,HOWTO Connectivité globale d'un LAN IPv6

Novell/SuSE
: Linux Novell/SuSE

MandrivaMandriva

Pour en savoir plus voir la page état des distributions Linux+IPv6.

23.3.4.3. Général

- IPv6.org
- 6bone
- Centre de ressources britanique IPv6 − Royaume−Uni
- Projet WIDE − Japon
- SWITCH IPv6 Pilot − Suisse

- Le coin IPv6 de Hubert Feyrer – Allemagne
- Projet Vermicelli – Norvège
- IPv6 Forum – un consortium mondial d'importants fournisseurs Internet, Research & Education Networks...
- Playground.sun.com / Page d'info IPv6 – maintenu par Robert Hinden, Nokia. Obtenir toute information au sujet d'IPv6, depuis de simples vues d'ensemble, en passant par les RFC et brouillons, jusqu'aux implémentations (incluant la disponibilité des piles sur différentes plates–formes & le code source des piles IPv6).
- 6INIT – Initiative Internet IPv6 – le 5ème programme–cadre européenne R&D de l'IST.
- IPv6 Task Force (Union Européenne) (*NdT*: IPv6 Task Force France)
- Projet de Documentation IPv6 (langue japonaise)
- 6init – IniTiative INternet IPv6
- Vue d'ensemble d'IP Nouvelle Génération
- IPv6: La nouvelle version du protocole Internet, par Steve Deering.
- IPv6: Le protocole Internet Nouvelle Génération, par Gary C. Kessler.
- IPv6: Le protocole Internet Nouvelle Génération – 3Com
- Initiative Internet Nouvelle Génération
- internet || site et Groupe de travail internet2 – Presentation (HTML + PPT) de l'atelier IPv6: (auto–configuration sans état, adressage IPv6, USAGI, fournisseur d'adressage IPv6 indépendant et autres thèmes).
- NetworkWorldFusion: rechercher IPv6 (102 documents trouvés au 22.12.2002)
- The Register (la recherche pour IPv6 donne 30 documents, 22.12.2002)
- recherche chez ZDNet pour IPv6
- Recherche chez TechTarget pour IPv6
- Liste de resssources IPv6 & TCP
- Les outils IPv6 Klingon, les outils IPv6 Klingon (accessible seulement en IPv6 natif): exemples de pare–feu IPv6, test de bande passante et scanner de ports

Quelque chose manque? Les suggestions sont les bienvenues!

23.3.4.4. Etudes de marché

- A Tale of Two Wireless Technology Trends: Processor Development Outsourcing and IPv6Groupe Yankee – 1/4/2002 – 12 Pages – ID: YANL768881
- The World Atlas of the Internet: Americas; IDATE – 2/1/2002 – 242 Pages – ID: IDT803907. Les pays couverts: Amérique Centrale, Amérique du Nord, Amérique du Sud; Liste: Prix: $ 3,500.00; à l'exception: Panorama du marché des accès à l'Internet à travers le monde. Estimation du marché et prévions jusqu'en 2006 pour 34 pays: structure du marché: les principaux ISP et le partage du marché; nombre de souscripteurs, d'ISP.
- Earlier Interest Rising for IPv6 par IDC (Auteur); prix: $1,500.00; support: e–book (Acrobat Reader); éd. IDC; ISBN B000065T8E; (1 mars 2002)

23.3.4.5. Les brevets

- Base de données des brevets canadiens: Accueil, Recherche (Recherche simple, entrez juste "IPv6" dans le champs recherche ;–); 84 documents trouvés au 22.12.2002)
- Espacenet – information sur les brevets européens: Offices nationaux, membres d'Espacenet(IPv6: 84 documents, au 22.12.2002)
- Delphion: Recherche de brevets. Un simple enregistrement (gratuit) est nécessaire. Exemples trouvés au 21.12.2002 par une recherche sur l'expression IPv6: Méthode de communication entre terminal IPv4 et terminal IPv6, mécanisme de conversion IPv4–IPv6 Traducteur pour réseaux IP, système

réseau utilisant le traducteur, mais aussi une méthode de couplage de réseaux IP

23.3.5. Par pays

23.3.5.1. Europe

- www.ist−ipv6.org: IST IPv6 Cluster, recherche européenne IPv6 et développement de projets
- Euro6IX: Backbone européenne d'interconnexion Internet IPv6

23.3.5.2. Autriche

- IPv6@IKNnet et le groupe de recherche MIPv6: Vienne , Autriche (IPv6: projets, publications, diplômes / thèses de doctorat, actes de conférence, *etc.*)é

23.3.5.3. Australie

- Les pages IPv6 australiennes de Carl (contenu ancien)

23.3.5.4. Belgique

23.3.5.5. Brésil

- BR6bone
- IPv6 Summit in Brazil
- IPv6 do Brasil

23.3.5.6. Chine

23.3.5.7. Tchèque

23.3.5.8. Allemagne

- IPv6−net.org: Forum IPv6 allemand

23.3.5.9. France

- Renater: La page d'accueil du projet IPv6 Renater
- IPv6 − RSVP − ATM à l'INRIA
- Documentation IPv6 NetBSD IPv6

23.3.5.10. Hongrie

- Tester la technologie expérimentale IPv6 et ses services en Hongrie

23.3.5.11. Inde

23.3.5.12. Italie

23.3.5.13. Japon

- Groupe d'utilisateurs IPv6 Linux JP
- Yamaha IPv6 (désolès, tout en japonais...)

23.3.5.14. Corée

- ETRI: Institut de Recherche en Electronique et Télécommunications
- Forum IPv6 koréen: Projet de déploiement coréen d'IPv6

23.3.5.15. Mexique

- Mexique IPv6 (versions espagnole et anglaise): Accueil du projet IPv6 de l'Université nationale autonome du Mexique (UNAM)

23.3.5.16. Pays−Bas

- SURFnet: Backbone IPv6 SURFnet
- STACK, STACK (IPv6): Association d'étudiants en informatique de l' Université de Technologie, Pays−Bas
- IPng.nl: collaboration entre WiseGuys et Intouch

23.3.5.17. Portugal

- FCCN (Fondation Nationale pour le Calcul Scientifique)

23.3.5.18. Russie

- Forum IPv6 pour la Russie: Centre Internet de l'Université publique de Yaroslavl

---

23.3.5.19. Suisse

- SWITCH: L'éducation suisse & les recherches réseau

23.3.5.20. Royaume−Uni

- IPv6 au Royaume−Uni
- Centre de resources IPv6 britanique
- La page d'accueil de British Telecom IPv6: essai de BT en tant que fournisseur de service IPv6, premier point d'interconnexion Internet au Royaume−Uni, ...

23.3.6. Par systèmes d'exploitation

23.3.6.1. *BSD

- Le projet KAME (*BSD)
- FAQ de la mise en réseau IPv6 de NetBSD
- Le port FreeBSD d'Ipv6
- BUGAT − Groupe d'utilisateurs Australiens BSD − www.bugat.at: Tunnel IPv6 FreeBSD (langue allemande)

23.3.6.2. Cisco IOS

- Cisco IOS IPv6
- IPv6 pour IOS de Cisco, Fichier 2 sur 3: août 2002 −− Table des Matières: IPv6 pour IOS de Cisco; documentation des caractéristiques de configuration; Rendre disponible et configurer le routage IPv6; l'adressage IPv6; Rendre globalement le fonctionnement IPv6 disponible.
- Manuel de la mise en réseau Internet Cisco, chapitre IPv6

23.3.6.3. Compaq

- IPv6 chez Compaq − Présentations, livres blancs, documentation, ...

23.3.6.4. HPUX

- FAQ comp.sys.hp.hpux

23.3.6.5. IBM

- Maintenant c'est IBM qui annonce la disponibilité de z/OS V1.4, Quoi de neuf dans cette version? Cette question a été posée le 15 août 2002

23.3.6.6. Microsoft

- Microsoft Windows 2000 IPv6
- MSRIPv6 − Accueil IPv6 des recherches réseau de Microsoft
- Débuter avec la technologie IPv6 Microsoft prévue pour Windows 2000
- Le pare−feu servant à la connexion Internet ne bloque pas le trafic IPv6 (au 6.11.2001)
- Internet Protocol Numbers (au 8.10.2002)
- IPv6 Technology Preview Refresh (au 16.10.2002)
- Comment: installer et configurer IP version 6 pour Windows .NET Enterprise Server (au 26.10.2002)
- Le service de routage 6to4 du serveur Windows .NET quitte lorsque vous publiez une adresse 2002 sur une interface publique (au 28.10.2002)
- msdn − Microsoft Windows CE .NET − commandes IPv6
- msdn − recherche pour IPv6 (100 résultats, au 22.12.2002)

23.3.6.7. Solaris

- Solaris Sun Microsystems
- Solaris 2 Frequently Asked Questions (FAQ) 1.73

23.3.6.8. Sumitoma é

- Sumitomo Electric a implémenté IPv6 sur la famille des routeurs Suminet 3700

23.3.6.9. ZebOS

- IpInfusion's ZebOS Logiciel de serveur de routage

23.3.7. La sécurité IPv6

- Internet Security Systems: Centre Sécurité, recherche dans la base de donnée X−Force (21.12.2002 − 6 thèmes relatifs à IPv6)
- Projet IPsec NIST (Institut National des Standards et Technologie, NIST)
- Information Security
- NewOrder.box.sk (recherche pour IPv6) (Articles, exploits, files database, etc.)

23.3.8. Les listes d'applications

- IPv6.org / Ies applications disposant d'Pv6
- Freshmeat / recherche IPv6, actuellement (au 14 décembre 2002), 62 projets
- Forum IPv6 : IPv6 Router List

23.3.8.1. Les outils d'analyse

- Ethereal − Ethereal est un analyseur libre de protocoles réseaux pour Unix et Windows
- Radcom RC100−WL − Téléchargez l'analyseur de protocoles RC100−WL Radcom version 3.20

23.3.8.2. Les produits IPv6

- 6wind − solutions pour routeur IPv4/IPv6, QoS, Multicast, Mobilité, Sécurité/VPN/Pare−feu.
- Fefe's patches for IPv6 with djbdnsAoût 2002 −− Qu'est−ce que djbdns et a−t−il besoin d'IPv6? djbdns est un serveur DNS complet qui outrepasse les performances de BIND.
- Suite de serveurs de routage ZebOS
- Serveur de courrier SPA 2.21
- Inframail (Advantage Server Edition) 6.0
- HTTrack Website Copier
- CommView 5.0
- Posadis 0.50.6
- TCP Wrapper (prêt pour IPv6)

23.3.8.3. SNMP

- comp.protocpols.snmp SNMP FAQ Parties 1 of 2

23.4. L'infrastructure IPv6

23.4.1. Statistiques

- Histoire de la table de routage IPv6 créée par Gert Döring, Space.Net
- Official 6bone Webserver list Statisic
- IPv6 Allocation Data & Survey Results, IPv6 WG, Ripe 42, Ripe NCC

23.4.2. Points d'interconnexion Internet

Une autre liste de points d'interconnexion IPv6 peut être trouvée ici: Site web des points d'interconnexion IPv6 ou Statut IPv6 des IXP en Europe

---

23.4.2.1. Estonie

- TIX (point d'interconnexion Internet Tallinn avec support IPv6)

23.4.2.2. Europe

- Euro6IX, Backbone des points d'interconnexion Internet IPv6 européen

23.4.2.3. France

- Point d'interconnexion Internet IPv6 français (actif depuis le 1.11.2002). FNIX6 fournit une interconnexion haut débit FastEthernet gratuite et fiable entre ISP situés chez TeleCity, Paris.

23.4.2.4. Allemagne

- INXS: Munich et Hamburg (câble & sans fil)

23.4.2.5. Japon

- NSPIXP−6: point d'interconnexion Internet basé sur IPv6, à Tokyo
- JPIX, Tokyo

23.4.2.6. Korée

- 6NGIX

23.4.2.7. Les Pays−Bas

- AMS−IX: Point d'interconnexion à Amsterdam

23.4.2.8. Royaume−Uni

- UK6X: Londres
- XchangePoint: Londres

23.4.2.9. USA

- 6TAP: Chicago. Supporte le *peering* à travers tout le globe
- NY6IX: Point d'interconnexion IPv6 basé à New York
- PAIX: Palo Alto

23.4.3. Les fournisseurs de tunnel (*tunnelbrokers*)

Voir aussi: http://www.deepspace6.net/docs/tunnelbrokers.html

23.4.3.1. Belgique

- Wanadoo

23.4.3.2. Canada

- Freenet6 – délégation /48, Canada Accéder à IPv6 en utilisant Freenet6 sur Debian Création Freenet6

23.4.3.3. Chine

- CERNET−Nokia

23.4.3.4. Estonie

- Estpak

23.4.3.5. Europe

- Fournisseur de tunnel distribué XS26, USA & Europeé

23.4.3.6. Allemagne

- 6bone Knoten Leipzig Info bez. Hackangriff (2001)
- Berkom

23.4.3.7. Italie

- Centro Studi e Laboratory Telecomunicazioni (page de téléchargement TunnelBroker Version 2.1.) Fournisseur de tunnel IPv6: instructions d'installation
- Comv6
- Bersafe (langue italienne)
- Telecom Italia LAB (page de téléchargement du logiciel Tunnelbroker)

23.4.3.8. Japon

- Initiative Internet au Japon (en langue japonaise) – avec fourniture de lignes IPv6 natives et tunnelage IPv6

23.4.3.9. Malaisie

- Manis

23.4.3.10. Les Pays−Bas

- XS26 – "Accès à Six" – avec des POP (Points De Présence) en République slovaque, en République Tchèque, aux Pays−Bas, en Allemagne et en Hongrie.
- IPng Pays−Bas – Intouch, SurfNet, AMS−IX, UUNet, Cistron, RIPE NCC et AT&T sont connectés au AMS−IX. Il est possible (sous certaines conditions) d'obtenir un tunnel statique.
- Clients SURFnet

23.4.3.11. Norvège

- UNINETT, Guide concernant le service IPv6 (pour les clients): fournisseur de tunnel et allocation d'adresse Uninett−Autoupdate−HOWTO

23.4.3.12. Espagne

- Consulintel

23.4.3.13. Suisse

- Fournisseur de tunnel AS8758, Dolphins Network Systems (en ligne depuis le 20.12.2002)

23.4.3.14. Royaume−Uni

- NTT Europe, NTT, Royaume−Uni − essai IPv6. Tunnel IPv4 et IPv6 natif and native IPv6 leased Line connexions. Les POP sont situés à Londres au Royaume Uni, Düssenldorf en Allemagne, New Jersey aux USA (Côte Est), Cupertino aux USA (Côte Ouest), Tokyo au Japon
- Interface d'administration du fourniseur de tunnel IPv6 BtexacT
- Royaume−Uni IPng

23.4.3.15. USA

- ESnet, USA − Réseau des Sciences de l'Energie: Tunnel Registry & adresse Delegation for directly connected ESnet sites and ESnet collaborators.
- 6REN, USA − l'initiative 6ren est coordonnée par Réseau des Sciences de l'Energie (ESnet), the network for the Energy Research program of the US Dept. of Energy, located at the University of California's Lawrence Berkeley National Laboratory.
- XS26 Distributed Tunnel Broker, USA & Europe
- Hurricane Electric, backbone US; Fournisseur de tunnel Hurrican Electric (aussi disponible ici) Press Version: Hurricane Electric devient fournisseur de tunnel IPv6 (communiqué de presse) Mise à jour d'extrémité de tunnel, pour fournisseur de tunnel, script Perl

---

23.4.3.16. Singapour

- http://tunnel−broker.singnet.com.sg/, avec en option NAT et IPsec

---

23.4.3.17. Plus de fournisseurs de tunnel...

- Routeurs relais 6to4 publiques (boycott MS IIE!)

23.4.4. Services nativement accessibles par IPv6

Note: Ces services sont uniquement disponibles grâce à une connexion IPv6 valide!

23.4.4.1. Serveur de jeu

- Quake2 sur IPv6

23.4.4.2. IRC Server

- Cyconet (serveurs IRCnet Cyconet sur IPv6)

23.4.4.3. Stations Radio, flux de musique

- Flux IPv6 expérimental en direct!, Université de Leipzig, Allemagne

23.4.4.4. Serveur web

- la page d'accueil de l'HOWTO IPv6 Linux de Peter Bieringer

Quelque chose manque? Les suggestions sont les bienvenues!

23.5. Les listes de diffusion

Des listes de listes de diffusion sont disponibles:

- DeepSpace6 / liste de listes de diffusion

Les listes de diffusion essentielles sont listées dans le tableau suivant:

Centre d'intérêt	Adresse mél de la requête d'inscription	A quoi souscrire	Adresse mél de la liste de diffusion	Langue	accès par WWW
L'activité réseau du noyau Linux incluant IPv6	majordomo (chez) oss.sgi.com	netdev	netdev (chez) oss.sgi.com	Anglaise	Archive
Linux et IPv6 en général (1)	majordomo (chez) list.f00f.org	linux−ipv6	linux−ipv6 (chez) list.f00f.org (modérée)	Anglaise	
L'implémentation Linux du protocole IPv6	interface web, voir l'URL		project6 (chez) ferrara.linux.it	Anglaise	Info, Subscriptio
La mobilité IP(v6) sur Linux	majordomo (chez) list.mipl.mediapoli.com	mipl	mipl (chez) list.mipl.mediapoli.com	Anglaise	Info, Archive
Utilisateurs Linux IPv6 avec l'extension USAGI	usagi−users−ctl (chez) linux−ipv6.org		usagi−users (chez) linux−ipv6.org	Anglaise	Info / Recherche, Ar
IPv6 sur Debian Linux	interface web, voir l'URL		debian−ipv6 (chez) lists.debian.org	Anglaise	Info/Souscription/Ar
IPv6/6bone en Allemagne	majordomo (chez) atlan.uni−muenster.de	ipv6	ipv6 (chez) uni−muenster.de	German/English	Info, Archive
6bone	majordomo (chez) isi.edu	6bone	6bone (chez) isi.edu	Anglaise	Info, Archive
Discussions IPv6	majordomo (chez) sunroof.eng.sun.com	ipng	ipng (chez) sunroof.eng.sun.com	Anglaise	Info, Archive, Mirro archives
Les utilisateurs d'IPv6 en général	majordomo (chez) ipv6.org	users	users (chez) ipv6.org	Anglaise	Info, Archive
Recherche des bogues des applications	bugtraq−subscribe (chez) securityfocus.com		bugtraq (chez) securityfocus.com (moderated)	Anglaise	Info, Archive

Internet (2)					
IPv6 en général	interface web, voir l'URL		ipv6 (chez) ipng.nl	Anglaise	Info/Subscription Archive
majordomo (chez) mfa.eti.br	majordomo (chez) mfa.eti.br	ipv6	ipv6 (chez) mfa.eti.br	Portugaise	Info

(1) recommandé pour les questions d'ordre général Linux & IPv6.

(2) très recommandé si vous êtes fournisseur d'applications serveur.

Quelque chose manque? Les suggestions sont les bienvenues!

Les listes de diffusion et newsgroups suivants sont disponibles *via* le web:

- ipv6 (France) Description: Cette liste IPv6 permet de discuter d'IPv6 en langue française. Elle s'adresse aux personnes désirant démarrer dès aujourd'hui des tests IPv6. Ce n'est en aucun cas un substitut aux listes de l'IETF. Pour de plus amples informations: http://www.urec.fr/IPng
- Tunnelbroker Maillingliste (Allemagne)
- ipv6 (Hongrie) Description: ipv6 Az IPv6 protokoll listaja Konfiguracios es adminisztracios kerdesek az IPv6−al kapcsolatban. (Archivum)
- student−ipv6 (Inde) Description: groupe d'étudiants intéressé par IPv6
- IPV6−CNR@LISTSERV.CNR.IT (Italie) Description: Groupe IPv6 au CNR
- ipv6−jp (Japon)
- ipv6 (Japon)
- sun−ipv6−users Description: Merci de rapporter les problèmes/suggestions concernant l'implémentation IPng SUN Microsystems
- IPv6−BITS Description: Cette liste coordonnera le travail du Verebrae.
- openbsd−ipv6
- IPv6 Description: Cette liste de diffusion consiste en discussions techniques au sujet des possibilités d'IPv6/IPsec WRT OpenBSD.
- linux−bangalore−ipv6 Description: La liste concernant le déploiement d'IPv6 du groupe d'utilisateurs Linux Bangalore
- gab Description: L'intention est de discuter du plan géographique d'adressage IPv6.
- ipv6−bsd−user Description: Cette liste de diffusion concerne l''"implémentation INéRIA/IMAG d'IPv6. Elle est bilingue, Francais/Anglais. Si vous souhaitez contacter les implémenteurs, essayez ipv6−bsd−core@imag.fr
- gated−ipv6
- La commutation de paquets Description: cette liste de diffusion fournit un forum de discussion au sujet de l'implémentation, de la technologie et de la théorie de la commutation de paquets et l'application à tout domaine, LAPB, X.25, SDLC, P802.1d, LLC, IP, IPv6, IPX, DECNET, APPLETALK, FR, PPP, téléphonie IP, les systèmes PBX LAN, les protocoles d'administration comme SNMP, e−mail, système de fenêtre transparent au réseau, implémentation de protocoles, vérification de protocoles, tests de conformité et outils utilisés dans la maintenance ou dans le développement des systèmes de commutation de paquets.
- mumbaiinternetgroup Description: Ce forum discutera des problèmes et des développements actuels concernant Internet dans la région de l'Asie pacifique. Cela couvrira IPv4, IPv6, le DNS multilingue, les numéros de systèmes autonomes, la gouvernance Internet et bien plus...
- de.comm.protocols.tcp−ip Description: Umstellung auf IPv6 Source: Chartas der Newsgruppen in de.*
- Groupe Google: comp.protocols.tcp−ip

- Groupe Google: linux.debian.maint.ipv6
- Groupe Google: microsoft.public.platformsdk.networking.ipv6
- Groupe Google: fa.openbsd.ipv6

23.6. Outils en ligne

23.6.1. Outils de test

- finger, nslookup, ping, traceroute, whois: Centre de ressources IPv6 britanique / la page de test
- ping, traceroute, tracepath, whois 6bone, DNS: JOIN / outils de test (en langue allemande seulement, mais en l'occurrence cela ne devrait pas être un problème pour les non germanistes)
- traceroute6, whois: IPng.nl
- Vérificateur de résolution AAAA http://www.cnri.dit.ie/cgi−bin/check_aaaa.pl
- Outils divers : IPv6tools
- Outil d'analyse d'adresse IPv6 (assez similaire à l'option d'information d'ipv6calc)

23.6.2. Recherche d'information

- Le bureau d'enregistrement 6BONE
- Liste mondiale d'attribution de tout bloc d'adresses IPv6

23.6.3. Outils d'observation des réseaux IPv6

- Observer IPv6 chez SURRIEL
- Observer IPv6 chez DRENv6

23.6.4. Applications venant en aide

- Calculatrice de préfixe IPv6 par TDOI
- Vérificateur d'enregistrement DNS

23.7. Pratique, séminaires

- formation et atelier IPv6, AERAsec, Allemagne (en langue allemande pour l'instant)
- Migrer vers IPv6, Learning Tree International
- Formation professionnelle CIW la maintenance Internet CBT CD
- Pages concernant la formation, Royaume−Uni − recherche à partir du mot clé "IPv6" (13 cours, au 22.12.2002)

Quelque chose manque? Les suggestions sont les bienvenues!

23.8. 'La découverte en ligne'...

IPv6: Addressing The Needs Of the Future [DOWNLOAD: PDF] par le groupe Yankee (Auteur) Prix: $595.00 Edition: e−book (Acrobat Reader) Pages: 3 (trois) Editeur: MarketResearch.com; ISBN B00006334Y; (1 novembre 2001)

;−) Le nombre de copies serait intéressant à connaître...

Chapitre 24. Historique des Révisions / Crédits / La Fin

24.1. Historique des Révisions

24.1.1. Révisions 0.x

24.1.1.1. La version anglo−saxonne (document original de Peter Bieringer)

Un historique des modifications de la version anglo−saxonne originale peut être trouvé ici: TLDP / Linux+IPv6−HOWTO / Revision History.

24.1.1.2. La version francophone

0.49.fr.1
: 26−02−2006/MB: Mise à jour au profit de la révision 0.49.

0.48.1.fr.1
: 20−01−2005/MB: Mise à jour au profit de la révision 0.48.1.

0.47.fr.1
: 05−09−2004/MB: Mise à jour au profit de la révision 0.47.

0.45.1.fr.1
: 14−03−2004/MB: Mise à jour au profit de la révision 0.45.1. Corrections et améliorations diverses.

0.44.fr.1
: 05−09−2003/MB: Mise à jour au profit de la révision 0.44. A cette occasion, une révision non systématique est réalisée.

0.43.2.fr.2
: 17−07−2003/MB: Correction de quelques coquilles, amélioration de la traduction de quelques passages.

0.43.2.fr.1
: 20−06−2003/MB: Mise à jour au profit de la révision 0.43.2. A cette occasion, une révision non systématique est réalisée.

0.41.1.fr.2
: 06−06−2003/MB: Première révision générale; à savoir, correction de coquilles, bogues, fautes d'orthographe, *etc.*

0.41.1.fr.1
: 09−05−2003/MB: Cette version est la première. Son contenu, rédigé sur LyX version 1.3.2, est basé sur celui de la version anglo−saxonne 0.41.1. Cependant un certain nombre de mes suggestions (*Michel Boucey*) ayant été prises en compte par Peter Bieringer lors du travail de traduction, il existe d'ores et déjà, ça et là, des modifications mineures qui seront présentes dans les futures versions du document original. Le suffixe .fr.x indique le numéro de révision de la traduction francophone.

24.2. Crédits

Le moyen le plus rapide d'être ajouté à cette sympathique liste est de m'envoyer des corrections (de bogue), et/ou mises à jour ;−).

Si vous voulez réaliser un réexamen important, vous pouvez utiliser le fichier natif LyX (voir le document original) et envoyez les diffs s'y rapportant, car les diffs en rapport au code SGML ne sont pas d'une grande utilité (*NdT*: merci d'envoyer les diffs à l'adresse <mboucey chez free point fr>).

24.2.1. Crédits majeurs

- David Ranch <dranch chez trinnet point net>: pour m'avoir encouragé à écrire cet HOWTO, pour ses commentaires sur quelques premières des révisions, et ses contributions à de différents résultats de test IPv6 sur mon site web IPv6. Mais aussi pour ses relectures et suggestions.
- Pekka Savola <pekkas chez netcore point fi>: pour ses relectures essentielles, apports et suggestions.
- Martin F. Krafft <madduck chez madduck point net>: pour la vérification orthographique et sa relecture générale du document.
- John Ronan <j0n chez tssg point wit point ie>: pour la vérification orthographique.
- Georg Käfer <gkaefer chez gmx point at>: pour la détection de la création défectueuse au format PDF (problème réglé maintenant par Greg Ferguson, travaillant au LDP), les références de livres en langue allemande, une grande liste d'URL, leurs vérifications, une grande quantité de suggestions, de corrections, de contributions, et pour sa traduction en langue allemande.
- Michel Boucey <mboucey chez free point fr>: pour la correction orthographique, la découverte de liens brisés, sa contribution grâce à ses suggestions, ses apports de nouveaux liens, et pour sa traduction en langue française.
- Michele Ferritto <m dot ferritto at virgilio dot it>: pour avoir découvert des bogues, et pour sa traduction en langue italienne.
- Daniel Roesen <dr at cluenet dot de>: pour ses vérifications orthographiques.

24.2.2. Autres crédits

24.2.2.1. Crédits relatifs aux documents techniques

Ecrire, en étant débutant, un HOWTO LDP (dans LyX et en exportant le travail vers DocBook pour se conformer au SGML) n'est pas si facile que certains pourraient le dire. Il y a d'étranges pièges... malgré tout, merci:

- Aux auteurs du Guide de l'auteur LDP
- A B. Guillon: pour l'HOWTO DocBook avec LyX

24.2.2.2. Crédits relatifs à la traduction francophone

Les crédits concernant les corrections/suggestions apportées à la version francophone viendront ici (peut−être un jour...). Merci par avance pour vos contributions.

24.3. La Fin

Merci de m'avoir lu. Dans l'espoir que cela puisse aider!

Si vous avez des questions, souscrivez à la bonne liste de diffusion et décrivez votre problème en fournissant autant d'information que possible.