Quelques points durs d'IPv6

Matthieu Herrb
CNRS-LAAS

JRES 2005 - Marseille

Introduction

Avancées d'IPv6 :

- Espace d'adressage quasiment infini
- Simplification de la structure des paquets
- Auto-configuration des adresses
- Sécurité intégrée
- Fonctions avancées : mobilité, multicast, etc.

Mais le déploiement tarde ...

> Using IPv6 in three stages :
>
> 1. believe.
> 2. hack and slash because it only has to work for me.
> 3. believe.
>
> IPv6 is dead. Because noone cares to make it REALLY WORK.
>
> – *Theo de Raadt*

Quelques (bonnes) nouvelles

17 juin 2005 : sortie du noyau Linux 2.6.12 : IPv6 enfin sorti de l'état expérimental

7 novembre 2005 : annonce de la fin du projet KAME : *mission accomplie !*

Manipulation des adresses

`2001:1234:5678:9abc:def0:1234:5678:9abc`

Qui arrive à mémoriser ça ? !

Mécanismes d'aide :
- auto-configuration basée sur l'adresse MAC (RFC 2462).
- tirage pseudo-aléatoire (Microsoft)
- DHCPv6 (RFC 3315) : toujours pas d'implémentation libre utilisable.

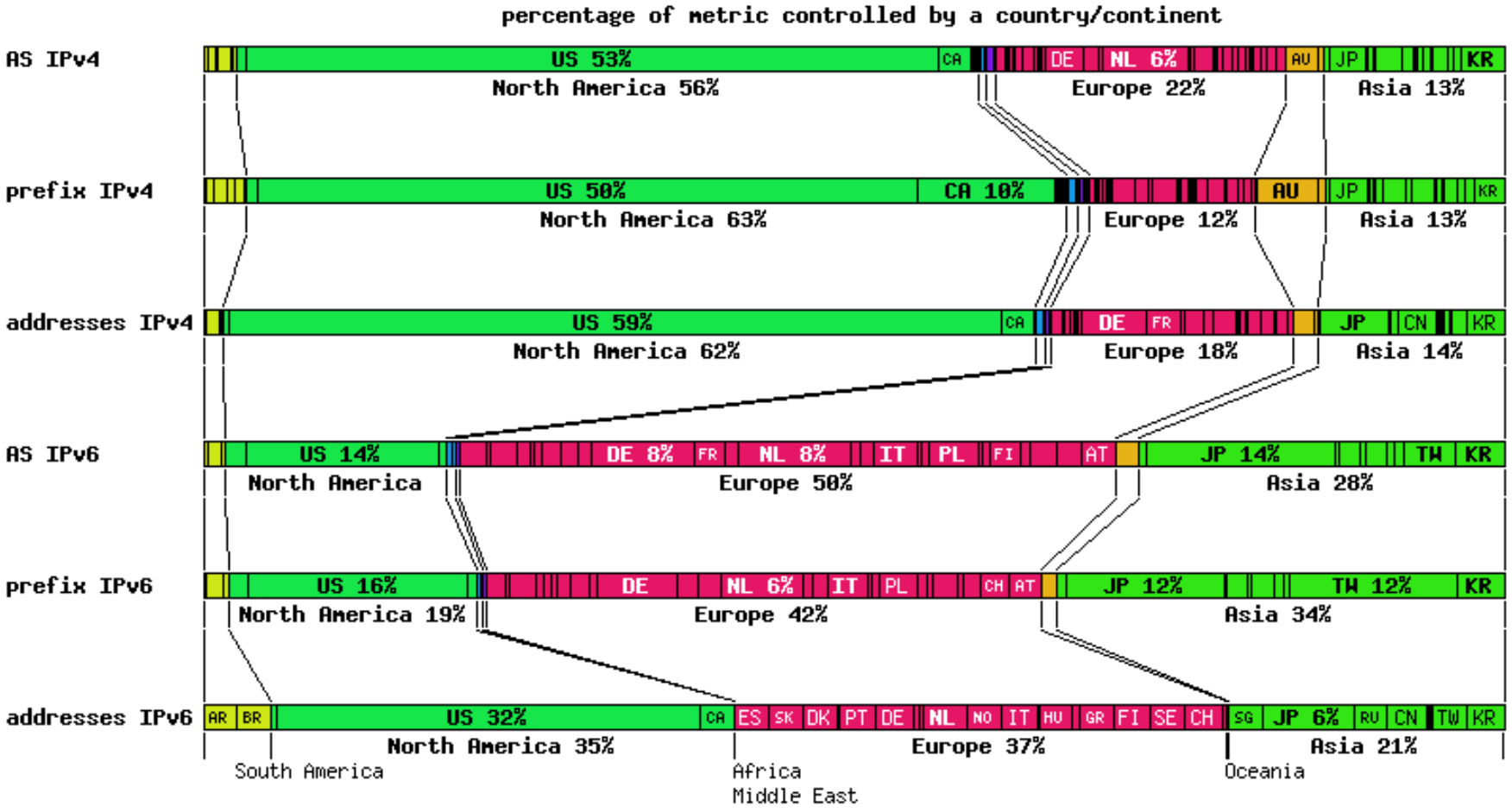
Besoin d'adresses globales ?

IPv6 : plusieurs milliers d'adresses par mètre-carré sur terre.

Mais l'internet d'aujourd'hui est composé de :
- serveurs Webs (72 millions selon Netcraft)
- pare-feux avec translation d'adresses et/ou proxys,
- réseaux privés d'opérateurs (ex. téléphones mobiles),
- ...

La pression sur les adresses existe, mais n'est pas aussi forte qu'on le prédisait.

Répartition des adresses IPv6



Source : http://www.caida.org/analysis/geopolitical/bgp2country/ipv6.xml

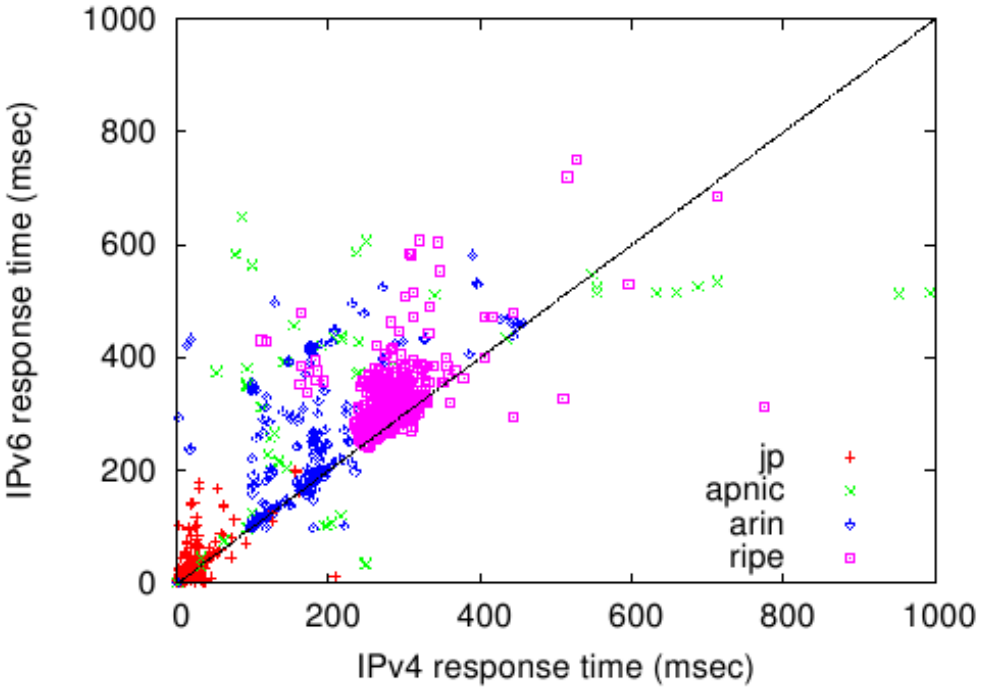
Infrastructure défaillante

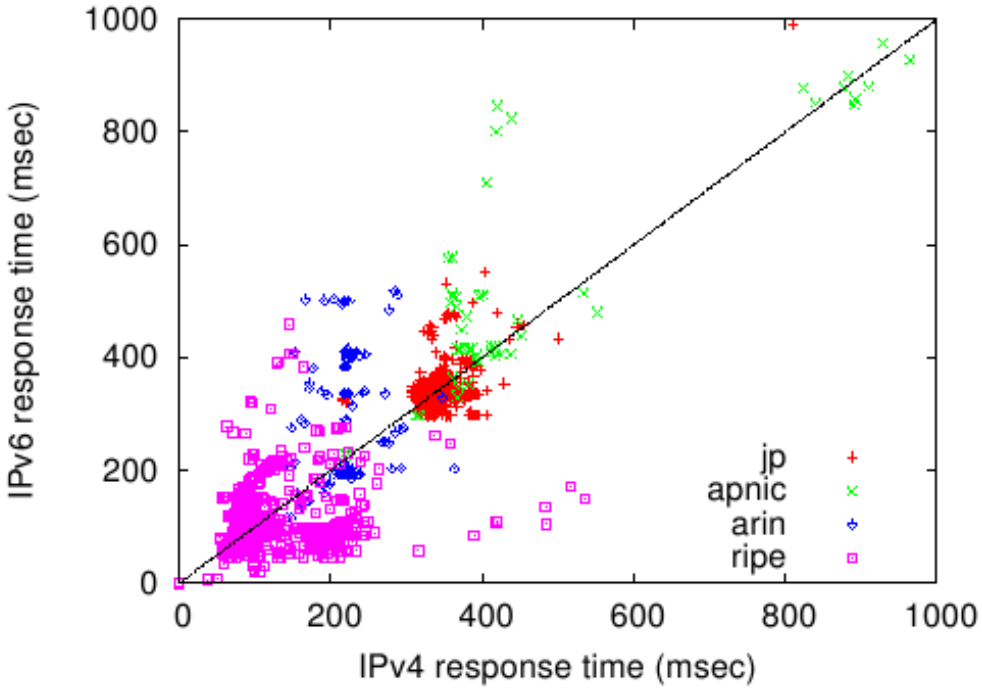
Phase expérimentale, donc :
- routeurs expérimentaux,
- instabilité des normes : problèmes d'interopérabilité,
- moins de supervision : délais d'intervention plus grands,
- routes IPv6 en moyenne plus longues.

Seuls opérateurs IPv6 en France : Renater et Nerim.

Infrastructure défaillante (2)

Comparaison des RTT pour IPv4 et IPv6, à partir du Japon et de l'Espagne [CHO-2004].





Mais les applications choisissent IPv6 de préférence $\longrightarrow$ problèmes.

Problèmes de DNS

A cause de la complexité des adresses, un bon DNS est indispensable.

Mais problèmes rencontrés avec des serveurs existants :

- *Pas de résolution DNS inverse* : erreur de paramétrage, ou impossible mettre à jour le serveur de la zone.
- *Réponse à une requête AAAA (IPv6) par un enregistrement A (IPv4)*.
- *Trou noir 1* : le serveur ne répond pas du tout à la requête AAAA : au bout de quelques dizaines de secondes d'attente le client abandonne la requête (ex. : doubleclick.net).

- *Trou noir 2* : au lieu de répondre qu'il n'y a pas d'adresse IPv6 correspondant au nom demandé, le serveur répond par une erreur indiquant que le domaine auquel appartient le nom n'existe pas ou est en erreur.

Une mesure du déploiement

Requêtes de type AAAA reçues par les serveurs DNS du LAAS en un semaine, par service :

Hôte	requêtes (total)	requêtes AAAA (%)
`www`	24507	3.9
`ftp`	691	6.7
`ntp`	12493	18.3
tous	397538	22.7

Beaucoup plus de machines capables de faire de l'IPv6 que de machines qui l'utilisent réelleement.

IPv6 dans les applications

Tous les systèmes d'exploitation majeurs supportent IPv6 :
Windows XP (SP2), Linux (noyau ≥ 2.6), Solaris (≥ 8), Mac OS X et les BSDs.

Il faut également modifier les applications :
Support dépendant du choix des développeurs.

Compatibles IPv6	Non compatibles IPv6
Internet Explorer	Squid
Mozilla	XDMCP
Firefox	NFS (autre que Solaris)
Thunderbird	Samba
Apache (≥ 2)	
OpenSSH	
X Window System	

Interface Socket

RFC 2553 étend l'API socket pour la rendre indépendante du protocole.

- nouvelle interface pour la résolution des noms : `getaddrinfo()` & cie
- famille d'adresses `AF_INET6` et extension des structures de type `sockaddr_in`

Encore trop peu enseignées / utilisées.

(`http://www.koders.com` : 3709 `gethostbyname()` / 464 `getaddrinfo()`)

Exemple de code pour double pile

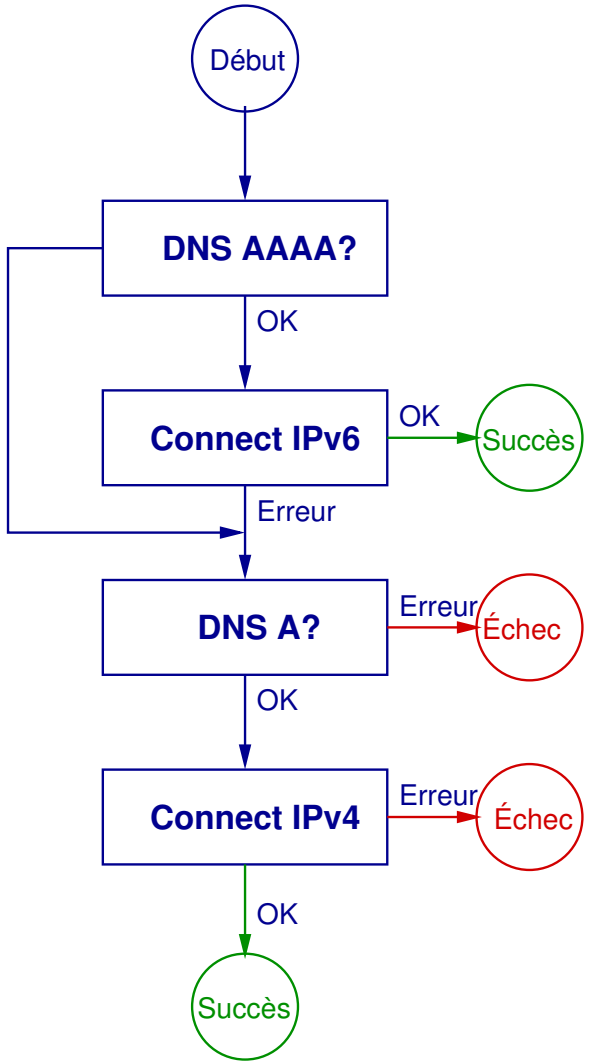
```
struct addrinfo hints, *res, *res0;
int error, s = -1;

memset(&hints, 0, sizeof(hints));
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
getaddrinfo("www.laas.fr", "80", &hints, &res0);
for (res = res0; res != NULL; res = res->ai_next) {
        s = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
                if (s < 0) continue;
        if (connect(s, res->ai_addr, res->ai_addrlen) < 0) {
                close(s);
                s = -1;
                continue;
        }
        break;
}
freeaddrinfo(res0);
```

Gestion de la double pile IPv4/IPv6

Quelle stratégie lorsque les 2 systèmes disposent d'une double pile ?

La stratégie est implémentée localement dans chaque application.

En général c'est le schéma ci-contre ⟶
(qui ne correspond pas à l'exemple précédent)



Exemple vécu : support d'IPv6 dans X11

X11 a été conçu pour être multi-protocoles (Decnet).

Extensions IPv6 proposées par Sun.

Intégrées par XFree86 et maintenant X.Org

- Adresses (RFC 3986) : `DISPLAY=[2001:660:6602:1:2]:0.0` .
- Deux sockets pour double-pile
- Pas de gestion des adresses lien-locales
- Extensions du protocole d'authentification
- Remplacer XDM-AUTHORIZATION-1 (transporte des adresses de 4 octets au plus).
- Difficultés avec XDMCP.

Malgré la bonne prédisposition, travail beaucoup plus complexe que ce qui avait été prévu.

Autres exemples

OpenSSH

Réécrit pour utiliser exclusivement l'API de la RFC 2553. Fournit une implémentation compatible pour les systèmes qui ne supportent qu'IPv4.
Un seul socket sur les systèmes qui supportent les adresses IPv4 mappées, deux sinon.
Choix du protocole par option ou fichier de configuration.

Apache

Profité de la réécriture pour la version 2. Spécifie explicitement les adresses sur lesquelles écouter pour forcer un protocole. (Difficulté avec les hôtes virtuels par adresse IP)

Squid

Aurait pu aider à rendre accessible des sites IPv6 aux machines IPv4.
Mais toujours pas de support officiel :-(

Politique du choix du protocole

Le choix du protocole par les applications pose de problèmes :

- Comportements incohérents.
- Impossible pour certaines applications de forcer la connexion avec un protocole donné.
- Vulnérabilité au problème des trous noirs du DNS.
- Complique le diagnostic en cas de mauvais fonctionnement.

Sécurité

IPv6 prend plus la sécurité en compte que IPv4 :

- simplification de la structure des paquets
- IPsec

Mais cela n'est pas suffisant :
- Implémentation beaucoup plus complexe
- Ne corrige pas un certain nombre de problèmes connus dans les protocoles TCP et UDP
- Problèmes mal connus faute de recul
- Code peu testé $\Rightarrow$ vulnérabilités et bugs dormants.

Adresses mappées

Permettent de représenter une adresse IPv4 au format IPv6 :
forme `::ffff:a.b.c.d`

Un serveur IPv6 peut accepter des connexions d'une machine IPv4 seule.

Problèmes :
- Comment distinguer un paquet IPv4 d'un paquet IPv6 contenant des adresses mappées ?
  $\longrightarrow$ ne pas laisser circuler ces adresses sur le réseau !
- risque de trous dans la politique de sécurité. Au moins trois adresses à bloquer : IPv4, IPv6 et IPv4 mappée dans IPv6. $\longrightarrow$ Source de complexité supplémentaire.

Recommandation : préférer la séparation complète des deux protocoles.

IPsec

Qui utilise IPsec sur IPv6 ?

Déjà complexe avec IPv4

Pas de démon IKE compatible IPv6 disponible

Implémentations expérimentales = pas encore vraiment testées.

$\Rightarrow$ pas de sécurité sur IPv6.

Conclusions

L'état actuel est peu satisfaisant pour déployer IPv6 aujourd'hui.

Les problèmes font fuir les téméraires testeurs.

Il faut sortir de la logique "hack and slash bacause it only has to work for me".

Mais IPv6 est indispensable pour l'accès internet des pays émergeants et pour le maintien de la forme ouverte de l'internet actuel.

Renforcer les soutiens aux projets qui améliorent la qualité d'IPv6 !