Cours de Réseaux

5 – La couche réseau

Rappel : le modèle hybride

5	Couche application
4	Couche transport
3	Couche réseau
2	Couche liaison de données
	(Sous-couche contrôle d'accès au canal)
1	Couche physique

La couche réseau

- Gère les communications de bout en bout, généralement entre machines
- Transmission de *paquets*
- Problèmes de routage
- Gestion des adresses de machines
- C'est la couche la plus haute qui est réalisée sur les routeurs.

Adressage de machines dans Internet + Ethernet

	Couche	*Exemple*
Nom	Applications	`avocat.lsv.ens-cachan.fr`
Adr. IP	Réseau	`138.231.81.68`
Adr. Ethernet	Liaison Données	`00 :13 :72 :75 :C2 :8B`

En vérité il s'agit des adresses des *interfaces*

Une machine peut en avoir plusieures.

Adressage

Pourquoi des niveaux d'adressage différents ?

- Plus de flexibilité :
  changer une carte réseau ou une machine et garder le nom.
- Les adresses sont spécifiques à la technologie utilisée dans une couche.

Traduction entre adresses de niveaux différent

Dans la combinaison Internet + Ethernet :

- Nom $\rightarrow$ Adresse IP :
  Domain Name Service (DNS) dans la couche Applications
  (voir plus tard)
- Adresse IP $\rightarrow$ Adresse Ethernet :
  Protocoles ARP (et RARP) de la couche Réseau
  (voir plus tard dans ce chapitre)

Service avec ou sans connexion (1)

Service avec connexion :

- Exemple : téléphone
- Au début d'une connexion : trouver un chemin entre émetteur et destinataire (problème de routage), réserver des capacités sur ce chemin
- À la fin : relâcher les ressources réservées
- Avantage : ressource réservée, qualité de service
- Inconvénient : Gaspillage de capacités, pas robuste, lourd pour des petits volumes de données

Service avec ou sans connexion (2)

Service sans connexion :

– Exemple : internet
– Envoi de paquet, chacun portant l'adresse de destination
– Les paquets sont transmis indépendamment un de l'autre
– Avantage : robuste contres des pannes, meilleure exploitation des capacité, pas de phase d'établissement de connexion
– Inconvénients : coût supplémentaire pour chaque paquet

Dans la suite de ce chapitre : service avec connexion.

Défis de la couche Réseau

– Routage : trouver le chemin entre émetteur et destinataire
– Qualité der service : exigences différentes selon l'application
– Contrôle de congestion
– Propager l'information sur la topologie du réseau, gérer des pannes
– Réseaux mobiles et ad-hoc
– Taille maximale des trames différentes, et différentes de la taille des paquets

Routage

Différents métriques possibles dans le critère d'optimalité :

- Nombre d'étapes
- Délai moyen mesuré
- Longueur de la file d'attente d'un routeur en moyenne
- Distance géographique

Décision à prendre par un routeur

Routeur R est censé envoyer un paquet à l'adresse IP i :

– Si i se trouve sur sur un LAN connexe à R :
  obtenir le numéro MAC correspondant, former une trame et envoyer sur le LAN locale.

– Si non :
  déterminer le routeur S suivant sur la route optimale
  (les routeurs R et S sont connexes)
  former une trame et envoyer à S

Un cas particulier important

Utiliser un *gateway* pour tous les envois. Exemple d'une machine connecté à un modem ADSL :

```
% /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Use Iface
82.227.67.0     *               255.255.255.0   eth0
default         82.227.67.254   0.0.0.0         eth0
```

(quelques colonnes supprimées)

Routage par vecteur de distance (1)

Aussi appelé routage de *Bellman-Ford*, ou aussi routage de *Ford-Fulkerson*.

Utilisé dans l'ARPAnet original

Soit

– V_r l'ensemble des voisins de r

– $Addr$ l'ensemble des adresses IP

Routage par vecteur de distance (2)

– Tout routeur connaît le coût d'envoi vers chacun de ses voisins

$$v_r \colon V_r \to C$$

– Tout routeur r maintient une table qui associe à chaque adresse le premier routeur sur le meilleur chemin qu'il connaît, et le coût de ce chemin :

$$t_r \colon Addr \to V_r \times C$$

(fonction partielle)

Routage par vecteur de distance (3)

Tous les routeurs envoient régulièrement leur t à leurs voisins

Calcul d'un nouveau t

$$t_r(a) := (s, c)$$

tel que $c = v_r(s) + t_s(a)$ minimal parmi tous les voisins s de r.

Routage par vecteur de distance (3)

Problème : Comptage à l'infini

A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

(b)

Routage par informations d'états de lien (1)

Link State Routing : Utilisé actuellement par internet (protocole OSPF)

La table de routage dans l'algorithme de Bellman-Ford ne contient pas suffisamment d'information.

$\Rightarrow$ il faut prendre en compte la topologie du réseau.

Routage par informations d'états de lien (2)

- Chaque routeur connaît (ou mesure régulièrement) le coût de connexion avec chacun de ses voisin.
- Chaque routeur envoie régulièrement cette information à *tous* les routeurs du réseau.
- Chaque routeur calcule la table de routage en utilisant l'algorithme du plus court chemin.

Routage par informations d'états de lien (3)

Problème : comment contrôler la diffusion des informations (boucles dans le réseau) ?

– utiliser des compteurs pour détecter des paquets en boucle : cher et pas robuste
– utiliser une horloge dans le paquet qui est décrementée à chaque étape (*hop counting*), initialisée à une valeur > diamètre du réseau.

Routage hiérarchique

Problème : comment maîtriser la taille des tables de routage ?

Solution : hiérarchiser le réseau :

- Structurer le réseau en *régions*, avec un *routeur principal* par région
- Les routeurs sur le WAN ne connaissent des régions que leurs routeurs principales.
- Les routeurs principales des régions connaissent la topologie de leur région.
- Plusieurs niveaux de hiérarchie possible.

Contrôle de congestion

– Les routeurs disposent des files d'attente pour gérer des variations de flux
– Possibilité : envoyer des paquet de rétention (*choke paquet*) à la source
– Possibilité : délestage de charge (*load shredding*)
fonctionne mieux si le routeur peut utiliser des informations sur la priorité du paquet, ou la qualité de service souhaitée
– Aujourd'hui plutôt traitée par la couche Transport.

Qualité de service (1)

4 critères de qualité importants :

– Fiabilité (*Reliability*) : taux de perte / erreur
– Délai : temps de transmission (moyen) pour un paquet donné
– Gigue (*Jitter*) : variation du temps de transmission des paquets
– Bande passante (*Bandwidth*) : nombre de paquets transmis par unité de temps

Qualité de service (2)

Exigences différentes pour des applications différentes :

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Fragmentation (1)

Problème : un paquet doit tenir dans une trame.

- un paquet peut être plus grand que la taille maximale d'une trame.
- les liens différents sur un chemin peuvent avoir une taille maximale différente pour les trames.

Par exemple Ethernet :

Taille maximale 1518 octets (dont 1500 pour les données).

Fragmentation (2)

Stratégies possibles

– Interdire (possible que dans des cas restreints)

– Erreur : demander la source de fabriquer des paquets plus petits (solution choisie dans IPv6)

– Hacher une trame dans des trames plus petites :

  – assemblage par routeur dès que possible

  – assemblage de morceaux par le destinataire (solution choisie dans IPv4)

IPv4 : Entêtes (1)

Protocole actuellement utilisé par internet

Informations contenues dans une entête IPv4 :

- Structure du paquet :
  - Version du protocole IP (4 ou 6)
  - Taille de l'entête ($\leq$ 60 octets)
  - Taille totale du paquet ($\leq$ 65.535 octets)
  - Checksum de l'entête (méthode particulière qui permet de recalculer rapidement la checksum quand l'entête change lors du passage au routeur)

IPv4 : Entêtes (2)

Informations contenues dans une entête IPv4 (2) :

- Informations relatives au routage :
  - Adresse émetteur
  - Adresse destinataire
  - Qualité de service (très peu utilisé)
  - Durée de vie (TTL = *Time to Live*) compteur pour le nombre maximal de sauts.

    Décrémentée par le routeur $\Rightarrow$ nécessité de recalculer la checksum.

IPv4 : Entêtes (3)

Informations contenues dans une entête IPv4 (3) :

- Informations relatives à la fragmentation :
  - Drapeau « interdiction de fragmenter »
  - Drapeau « ceci est le dernier fragment »
  - Identificateur du paquet original
  - Décalage par rapport au début du paquet original
- Options (plutôt utilisées pour la maintenance)
- Protocole de la couche Transport

IPv4 : Adresses (1)

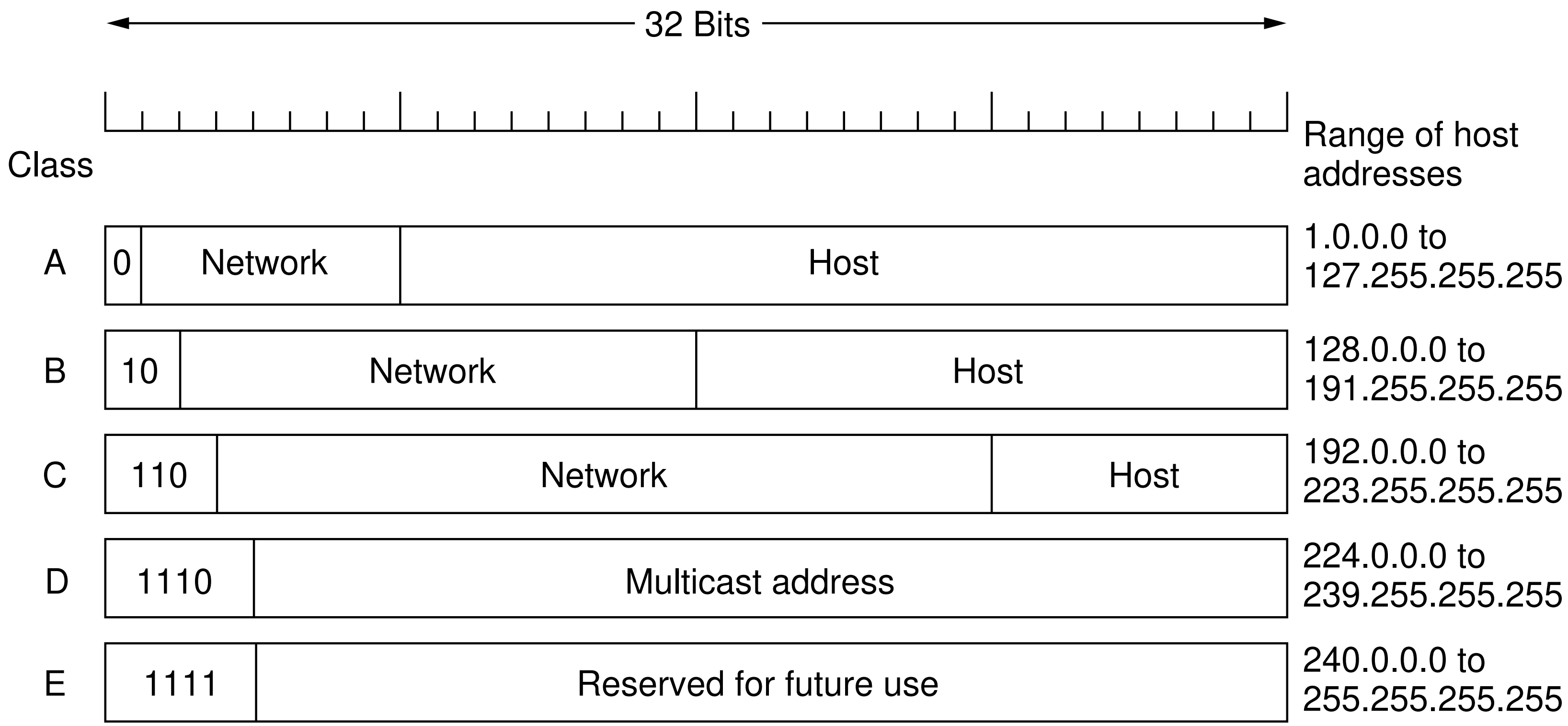
Format (original) des adresses :

– Longueur des adresses 4 octets, noté $n.n.n.n$
  par exemple `138.231.81.68`
– Hiérarchisation en deux niveaux : *network* et *host*
– Trois classes de *networks* :

Classe	Nombre de networks	Hosts par network
A	128	16.777.216
B	16.384	65.536
C	524.288	256

IPv4 : Adresses (2)

Format (original) des adresses :



IPv4 : Adresses (3)

Adresses particulières :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		This host
0 0 . . . 0 0	Host	A host on this network
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		Broadcast on the local network
Network	1 1 1 1 . . . 1 1 1 1	Broadcast on a distant network
127	(Anything)	Loopback

IPv4 : ARP

Étant donnée l'adresse IP d'une machine sur le même LAN, comment peut-on trouver son adresse MAC ?

Protocole ARP (*Address Resolution Protocol*) :
– Envoyer en multi-diffusion une demande avec l'adresse IP cherchée
– La machine avec la bonne adresse répond
– Optimisation : stocker les informations, et aussi les questions/réponses des autres vues sur le réseaux

IPv4 : La commande `traceroute`

La commande UNIX `traceroute` ***`adr`*** permet d'observer la route d'une machine vers *adr*.

Implémentation : envoyer succesivement des paquets à *adr*, avec le *time-to-live* mis à $1, 2, 3 \ldots$.

Récupérer les messages d'erreur renvoyés par les routeurs.

Certains routeurs ne renvoient pas de message d'erreur.

IPv4 : Exemple d'utilisation de `traceroute`

```
% traceroute ftp.jp.debian.org
 1  82.227.67.254  38.372 ms  49.192 ms  44.218 ms
 3  cbv-6k-1-po3.intf.routers.proxad.net 44.527 ms * *
 4  if-4-0.core2.PG1-Paris.teleglobe.net 43.802 ms  42.144 ms *
 6  if-2-0.core2.FR1-Frankfurt.teleglobe.net  47.411 ms  48.990 ms  46.995 ms
 8  ge-2-1.r01.frnkge03.de.bb.gin.ntt.net  54.464 ms  55.879 ms  53.997 ms
11  as-0.r22.londen03.uk.bb.gin.ntt.net 53.662 ms  54.276 ms  53.024 ms
14  p64-1-3-0.r21.tokyjp01.jp.bb.gin.ntt.net 321.136 ms  320.565 ms  346.830 ms
19  ve-7.hitachi2.nara.wide.ad.jp 331.672 ms  366.349 ms  337.638 ms
20  ftp.nara.wide.ad.jp 328.251 ms  352.343 ms  327.657 ms
```

IPv4 : sous-réseaux

- En principe, chaque réseau correspond à un LAN.
- Les réseaux classes B et A sont trop grands pour former un seul LAN.
- Solution : introduction d'un niveau hiérarchique supplémentaire de *sous-réseau* : une partie initiale de l'adresse de host, indiquée par une *masque*, dénote le sous-réseau.

IPv4 : CIDR (1)

Problème : mauvais découpage en classes.
Il y a trop de réseaux C, et les réseaux B sont souvent trop grands.

Solution : Routage interdomaine sans classe
(CIDR - *Classless InterDomain Routing*)
introduit en 1993
aussi appelé *VLSM - variable length subnet masking*

IPv4 : CIDR (2)

- permet de définir des régions de taille (nombre d'adresses) arbitraires
- idée : une table de routage contient pour toute région
  - adresse de début
  - longueur
  - adresse du routeur suivant
  - interface vers le routeur suivant
- implémentation : utiliser des masques de bits au lieu de la longueur
- permet de fusionner des entrées dans la table de routage

IPv4 : nombre d'adresses (1)

Problème : on a pas suffisamment d'adresses en IPv4 !

Nombre maximale d'adresses IPv4 (adresses particulières comprises) :

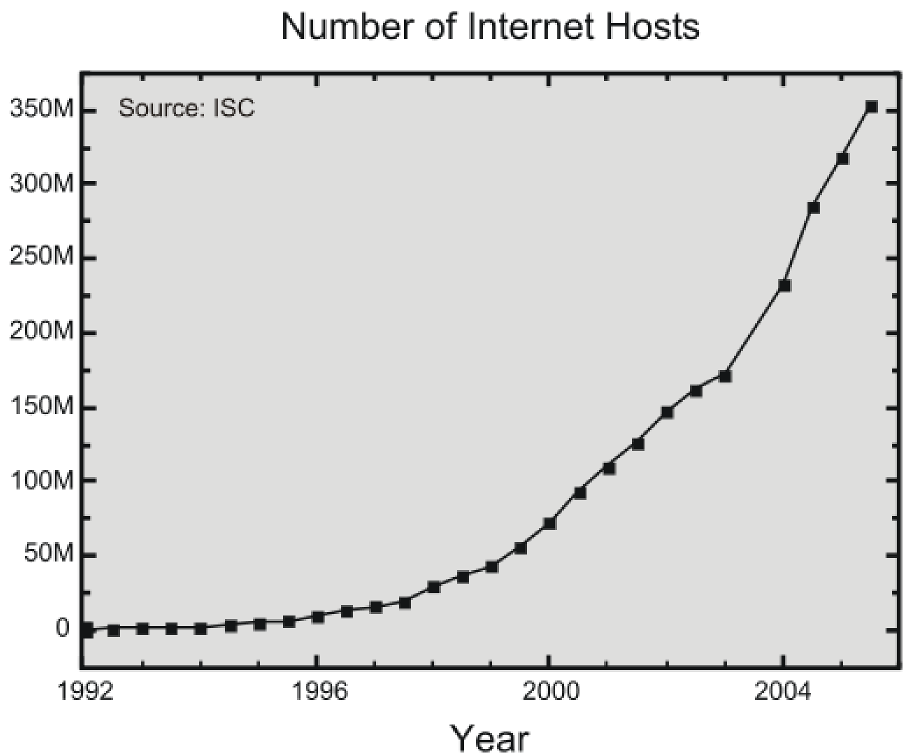
$$2^{32} = 4 * 10^9$$

Nombre de machines connectées à internet en juillet 2006 :

439.286.364

IPv4 : nombre d'adresses (2)

Évolution du nombre de machines connectées à internet :

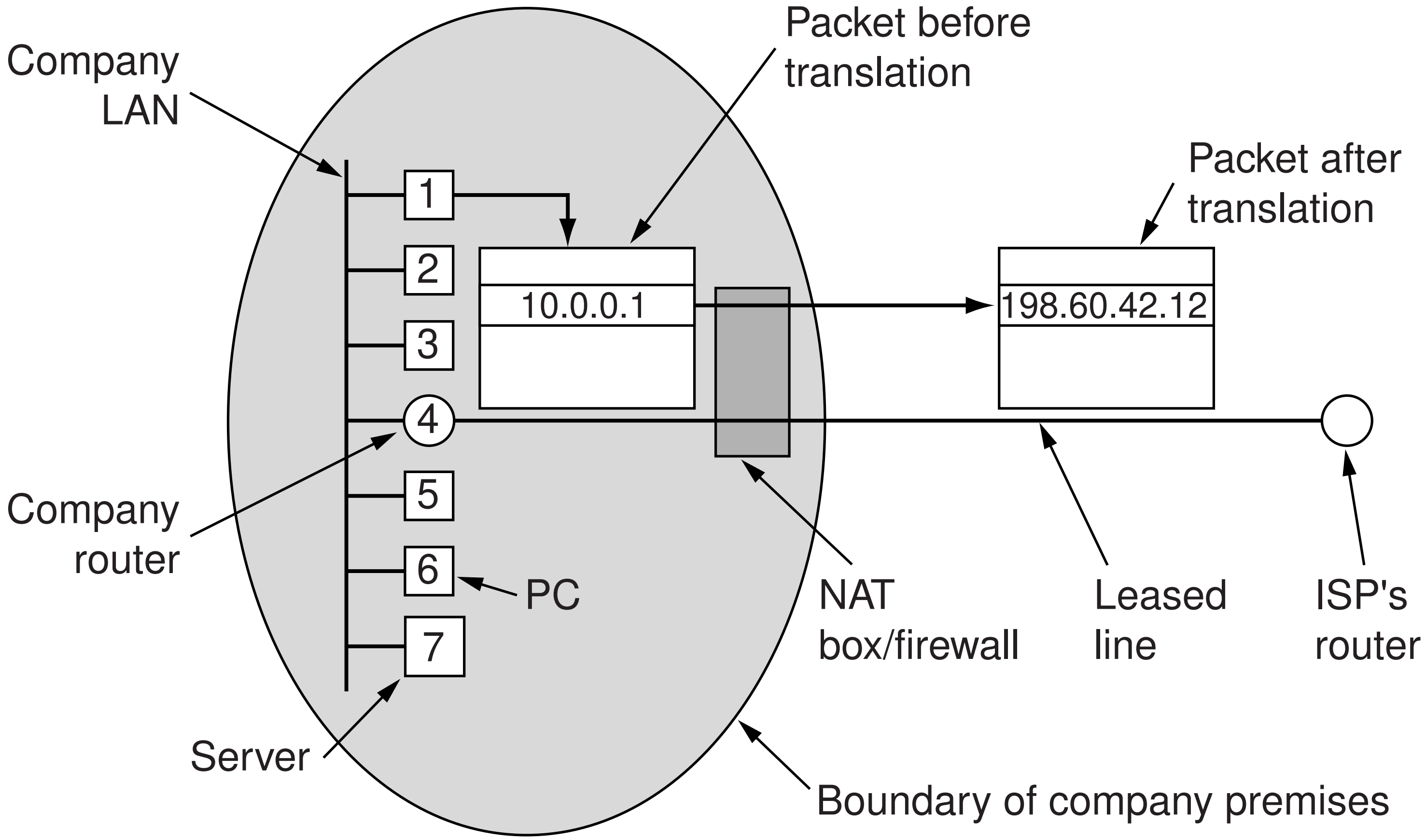


IPv4 : nombre d'adresses (3)

Solutions (temporaires) à la pénurie d'adresses IP :

- Affectation d'adresses IP seulement à la demande (par exemple pour des accès à internet par modem téléphonique)
- Traduction d'adresses de réseaux (NAT - *Network Address Translation*)
  Vu de l'extérieur il n'y a qu'une seule adresse IP
  Traduction des adresses par une boîte NAT qui utilise (ou abuse) les ports de la couche Transport.

IPv4 : Network Address Translation (1)



IPv4 : Network Address Translation (2)

– Une seule adresse par réseau
– Les machines individuelles ne sont plus « visibles » de l'extérieur
  ⇒ avantage pour la sécurité réseau
– Inconvénient : protocoles qui mentionnent les adresses IP aussi dans les données
  Exemple `ftp` : en linux, nécessite le module `ip_nat_ftp`
– Inconvénient majeur : utilise infrastructure de la couche Transport (les ports)

Le protocole IPv6

(aussi IPng : *IP next generation*)

- Adresse sur 16 octets :
  - solution définitive à la pénurie d'adresses
  - possibilité de coder les adresses MAC dans les adresse IP
- Pas de fragmentation en route
- Entête simplifiée
- Cryptage et authentification dans la couche réseau

À quand IPv6 ?

- Standardisé en 1994
- Passage à v6 retardé par NAT et CIDR
- Support pour IPv6 dans Linux (depuis les années 90), *BSD, Windows, Apple
- La Chine a un plan très ambitieux pour le passage à IPv6
- USA : tous les *backbones* en IPv6 à partir de 2008 (prévu)
- Période de transition : co-existence de IPv4 et IPv6