Compléments sur l'introduction aux principes généraux des réseaux

Plan

- Equipements d'interconnexion
- Lien entre couche 2 (liaison) et couche 3 (réseau)
- Adressage IPv4
- NAT
- IPv6

Equipements d'interconnexion

- **Niveau 1 (physique)**
  - **Répéteur : régénération/amplification du signal sur un lien**
  - **Concentrateur / *Hub* : raccordement de plusieurs liens**
    - Un signal est retransmis sur tous les liens
    - Des collisions sont possibles, comme sur un unique lien partagé
- **Niveau 2 (liaison)**
  - **Notion de pont (*bridge*) / commutateur (*switch*)**
  - **Interconnexion de plusieurs liens avec aiguillage**
    - Principe : une trame entrante n'est (si possible) retransmise que sur le lien qui mène au destinataire
    - Permet d'éviter les problèmes de collisions et d'obtenir de meilleures performances
  - **Comment aiguiller les trames ?**
    - À partir de l'adresse de niveau 2 (souvent appelée adresse MAC) du destinataire
    - En l'absence de connaissances, retransmettre sur les (N-1) liens.
    - Apprendre au fur et à mesure les chemins en observant l'adresse de l'émetteur de chaque trame arrivant sur chacun des liens

Equipements d'interconnexion (suite)

- **Niveau 2 (liaison) - suite**
  - **Il est possible de construire des topologies avec plusieurs commutateurs interconnectés**
  - **... et éventuellement avec des chemins redondants entre nœuds**
    - Meilleure robustesse en cas de panne d'un lien
    - Mais introduit un problème de boucles de diffusion (saturation du réseau)
    - … car les en-têtes de niveau 2 ne fournissent pas de champ Time-To-Live (durée de vie d'une trame)
    - Solution : protocole d'arbre couvrant (*spanning tree protocol*)
      - Synchronisation entre commutateurs d'un même réseau via un dialogue périodique
      - Résultat : désactivation de certains liens pour éviter les boucles (mais ces liens restent utilisés par le protocole de synchronisation, qui ne nécessite pas de diffusion). On obtient donc une topologie fonctionnelle de type arbre couvrant plutôt que de type graphe arbitraire.
      - Adaptation automatique aux modifications du réseau (pannes de liens ou changement de topologie/câblage).

Equipements d'interconnexion (suite)

- **Niveau 3 (réseau)**
  - **Notion de routeur (*router*) ou de passerelle (*gateway*)**
  - **Interconnexion de plusieurs réseaux avec aiguillage**
    - L'aiguillage des paquets se fait à partir de l'adresse de niveau 3 (typiquement adresse IP) du destinataire
    - Les adresses de niveau 2 ne sont pas prises en compte par un routeur ... mais il s'appuie sur la couche de niveau 2 pour ses communication point-à-point entrantes et sortantes (extraction du paquet de la trame entrante, aiguillage puis retransmission du paquet encapsulé dans une nouvelle trame)
  - **Contrairement à un commutateur (switch), un routeur**
    - Doit être configuré explicitement (définition des routes)
    - Peut disposer des plusieurs chemins redondants simultanément actifs vers une même destination
  - **Des protocoles de routage permettent à des routeurs d'échanger des informations afin que chaque routeur puisse prendre des décisions d'aiguillage appropriées**

Equipements d'interconnexion (suite)

- **Niveaux supérieurs (transport et/ou application)**
  - **L'idée d'aiguillage vue précédemment peut se transposer à des niveaux conceptuels supérieurs via la notion de passerelle protocolaire/applicative.**
  - **Exemple 1 : conversion de protocoles de transport**
    - La machine A ne connaît que le protocole de transport T_A (par exemple TCP)
    - La machine B ne connaît que le protocole de transport T_B (par exemple UDP)
    - Utilisation d'un nœud intermédiaire C (qui connaît T_A et T_B) comme relais pour les communications entre A et B
  - **Exemple 2 : proxy (mandataire) applicatif**
    - Entité intermédiaire/relais entre une application cliente et une application serveur
    - Illustration : proxy web (voir détails dans le cours sur le Web)

Commutateur ou routeur ?

- **À l'échelle d'un réseau local de moyenne envergure, on a généralement le choix entre :**
  - un schéma d'interconnexion de machines via un ensemble de commutateurs
  - un schéma d'interconnexion de petits « sous-réseaux » via des routeurs (chaque sous-réseau utilisant un ou plusieurs commutateurs pour son organisation interne)
- **Commutateurs**
  - Ne nécessitent pas de configuration pour l'aiguillage des trames
  - Bon rapport performances/prix
- **Routeurs**
  - Permettent d'avoir des chemins redondants simultanément actifs
  - Fournissent des leviers de contrôle sur le trafic réseau (notamment filtrage selon des critères de charge et de source/destination)
- **L'utilisation de routeurs devient incontournable à partir d'une certaine échelle (~1000 machines)**

Lien entre niveau 3 et niveau 2

- **Considérons les deux cas suivants:**
  - **Cas 1 : Un paquet (IP) envoyé par la machine A et destiné à la machine B arrive au niveau du dernier routeur R du chemin**
    - C'est-à-dire le routeur qui relie le réseau de la machine B à l'« extérieur »
  - **Cas 2 : Communication entre deux machines sur un même réseau**
    - Machines A et B (reliées directement ou interconnectées par un commutateur C)
    - Même si les machines peuvent communiquer directement au niveau 2, les applications utilisent des protocoles de transport, qui reposent eux-mêmes sur la couche réseau
- **Dans les deux cas, on a besoin de faire le lien entre une adresse de niveau 3 et une adresse de niveau 2**
  - **Cas 1 : le routeur R doit connaître l'adresse de niveau 2 de B**
  - **Cas 2 : la machine A doit connaître l'adresse de niveau 2 de B**
  - **Remarque : les adresses de niveau 2 sont généralement appelées *adresses MAC (Medium Access Control)***
- **Comment cette correspondance est-elle déterminée ?**

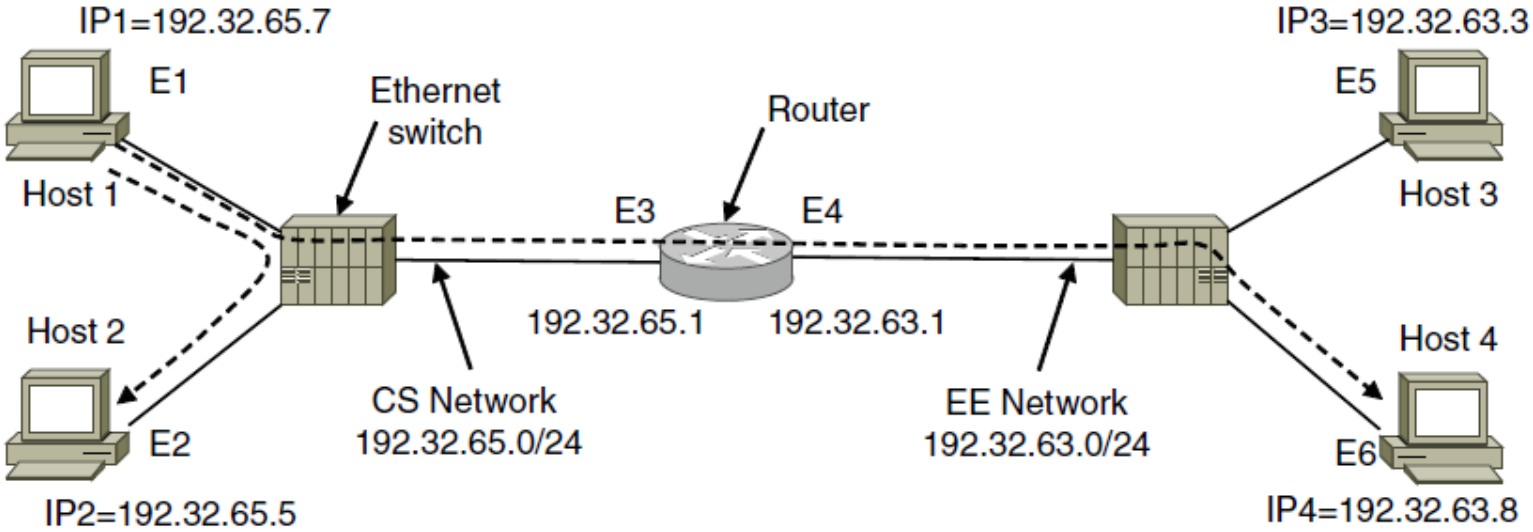
Lien entre niveau 3 et niveau 2
Protocole ARP (*Address Resolution Protocol*)

- **Le nœud N (c'est-à-dire le routeur R dans le cas 1, la machine A dans le cas 2), qui doit émettre une trame pour envoyer le paquet à B, envoie au préalable une requête ARP sur le lien qui le relie à B**
  - **Une requête ARP contient les adresses IP et MAC du demandeur et l'adresse IP du destinataire dont on cherche à connaître l'adresse MAC**
  - **Une requête ARP est envoyée à une adresse MAC spéciale qui correspond à l'ensemble des machines du réseau local**
    - C'est-à-dire les machines connectées directement à N (par un lien point-à-point ou un lien partagé) et les machines connectées indirectement via des concentrateurs (*hubs*) ou des commutateurs (*switches*)
- **La machine B concernée (si elle existe) répond en indiquant son adresse MAC**
  - **La réponse est adressée spécifiquement au demandeur N (contrairement à la requête, qui était diffusée à l'attention de toutes les machines du réseau)**
- **Remarque : le protocole ne nécessite aucune configuration particulière et s'adapte aux modifications du réseau local**

Lien entre niveau 3 et niveau 2
Protocole ARP (*Address Resolution Protocol*)

- **Chaque nœud gère une « table ARP », qui joue un rôle de cache**
  - Une requête ARP n'est envoyée sur le réseau local que si la réponse à la question ne peut être trouvée dans la table locale
  - La table d'un nœud est mise à jour lorsque celui-ci reçoit une requête (même s'il n'est pas concerné) ou reçoit une réponse
  - La validité d'une entrée expire au bout d'un temps borné
  - Une machine nouvellement connectée (ou reconfigurée) peut se faire connaître en envoyant une requête concernant sa propre adresse de niveau 2
- **Fonctionnement d'un commutateur vis-à-vis d'ARP**
  - Il dispose d'un cache mémorisant l'association entre une adresse MAC et le lien correspondant
  - Il met à jour le cache en observant le trafic (au fil des aiguillages de trames)
  - Lorsqu'il reçoit une trame à destination d'une machine d'adresse MAC A
    - S'il ne trouve pas le lien concerné dans le cache, il diffuse la trame sur tous ses ports
    - Sinon, il ne retransmet la trame que sur le lien indiqué par le cache
  - Lorsqu'il reçoit une requête ARP, il la retransmet sur tous ses liens (sémantique de diffusion)
  - La taille du cache impacte les performances (notamment pour les réseaux qui englobent de nombreuses machines via une grande hiérarchie de commutateurs)

Lien entre niveau 3 et niveau 2
Un exemple plus complet



Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6

Deux réseaux locaux (ethernet) interconnectés par un routeur

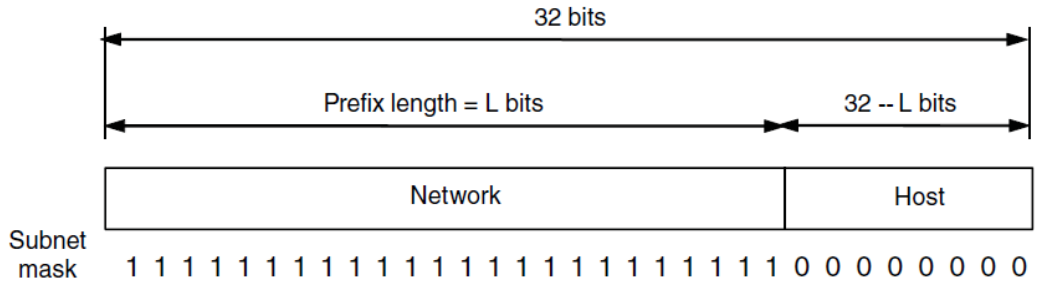
(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Adressage IPv4

- Une adresse IP est une valeur numérique codée sur 32 bits
- Chaque interface réseau est associée à une adresse IP
  - Une machine a au moins une interface réseau
  - Un routeur a de nombreuses interfaces réseau
- Une adresse IP est généralement écrite sous forme décimale pointée
  - 4 nombres (1 par octet) dans l'intervalle 0 à 255
  - Exemple : 0x80D00297 → 128.208.2.151
- Une adresse IP est hiérarchique (contrairement à une adresse MAC)
  - Elle est composée d'un identifiant de réseau (de taille variable) dans les bits de poids fort, et d'un identifiant d'interface (ou d' « hôte », par abus de langage) au sein d'un réseau dans les bits de poids faible
  - Un réseau correspond donc à un bloc contigu d'adresses. Un tel bloc est appelé « préfixe »

Adressage IPv4

- **Un préfixe est généralement écrit en prenant la plus basse adresse du bloc correspondant et en précisant la taille du bloc**
  - La taille est déterminée par le nombre de bits de la partie « hôte »
  - Cette taille est donc une puissance de 2
  - Exemple: si l'adresse 128.208.2.151 appartient à un réseau contenant 2^8 adresses d'interfaces, alors 24 bits sont consacrés à l'identifiant du préfixe, que l'on notera 128.208.2.0/24
  - La taille d'un préfixe correspond au nombre de bits à 1 dans un masque binaire (appelé *masque de sous-réseau* ou *subnet mask*)



  - La taille d'un préfixe ne peut pas être inférée directement à partir d'une adresse IP seule (voir détails plus loin)

(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Adressage hiérarchique

Avantages

- Un routeur peut aiguiller un paquet (IP) à partir du préfixe du destinataire (dès lors que chaque réseau dispose d'un préfixe unique)
- Cela permet de réduire nettement la taille des tables de routage

Inconvénients

- L'adresse d'un hôte dépend du réseau auquel il appartient et donc de sa localisation courante. Cela complique la gestion de la mobilité (ce problème ne sera pas étudié ici)
- L'allocation d'adresses par blocs peut conduire à des problèmes de gaspillage d'adresses (pénurie d'adresses dans certains réseaux alors que d'autres en ont en surplus)

Structuration de l'espace d'adressage

- **Un organisme gère l'attribution des plages d'adresses : *Internet Corporation for Assigned Names and Numbers* (ICANN)**
  - **La gestion de certaines plages est déléguée à des organismes régionaux**

- **Comment l'espace d'adresses IPv4 est-il structuré ?**
  - **Nous allons étudier deux aspects de cette question**
    - Structuration interne (au sein d'un réseau) : le principe des sous-réseaux (*subnets*)
    - Structuration au niveau de l'Internet : le principe des *classes* de réseaux (et ses améliorations)

Principe des sous-réseaux

Problème de départ

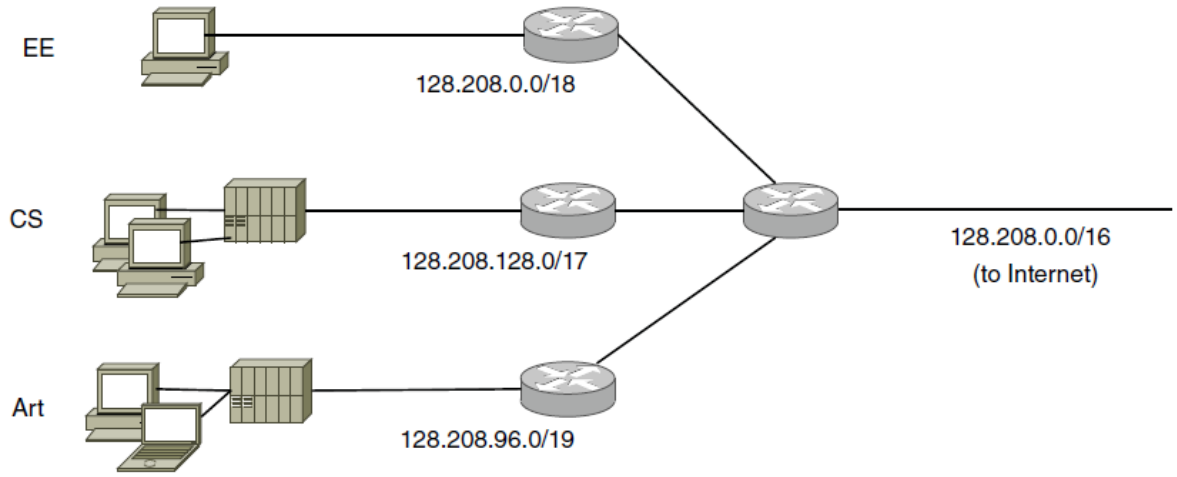
- On dispose déjà d'un réseau (relié à l'Internet) et on souhaite en déployer un second
- On dispose d'un large bloc d'adresses dont une partie est disponible et on voudrait éviter de réserver une nouvelle plage d'adresses

Solution

- Découper la plage de numéros d'hôtes au sein du réseau en deux dimensions : un numéro de sous-réseau et un numéro d'hôte au sein du sous réseau
- Vu de l'« extérieur » (les routeurs et machines des autres réseaux), rien ne change
- Au niveau du routeur d'entrée de notre réseau, on définit des règles d'analyse du préfixe étendu (basée sur des masques de sous-réseau) pour identifier le sous-réseau et aiguiller un paquet vers le routeur correspondant

Sous-réseaux : Exemple

- On dispose du préfixe 128.208.0.0/16 que l'on veut subdiviser en 3 sous-réseaux (de tailles diverses) pour les départements d'une université
- On choisit le découpage suivant
  - Computer Science : la moitié du réseau (128.208.128.0/17)
  - Electrical engineering : un quart du réseau (128.208.0.0/18)
  - Art : un huitième du réseau (128.208.96.0/19)
  - Un huitième du réseau non alloué



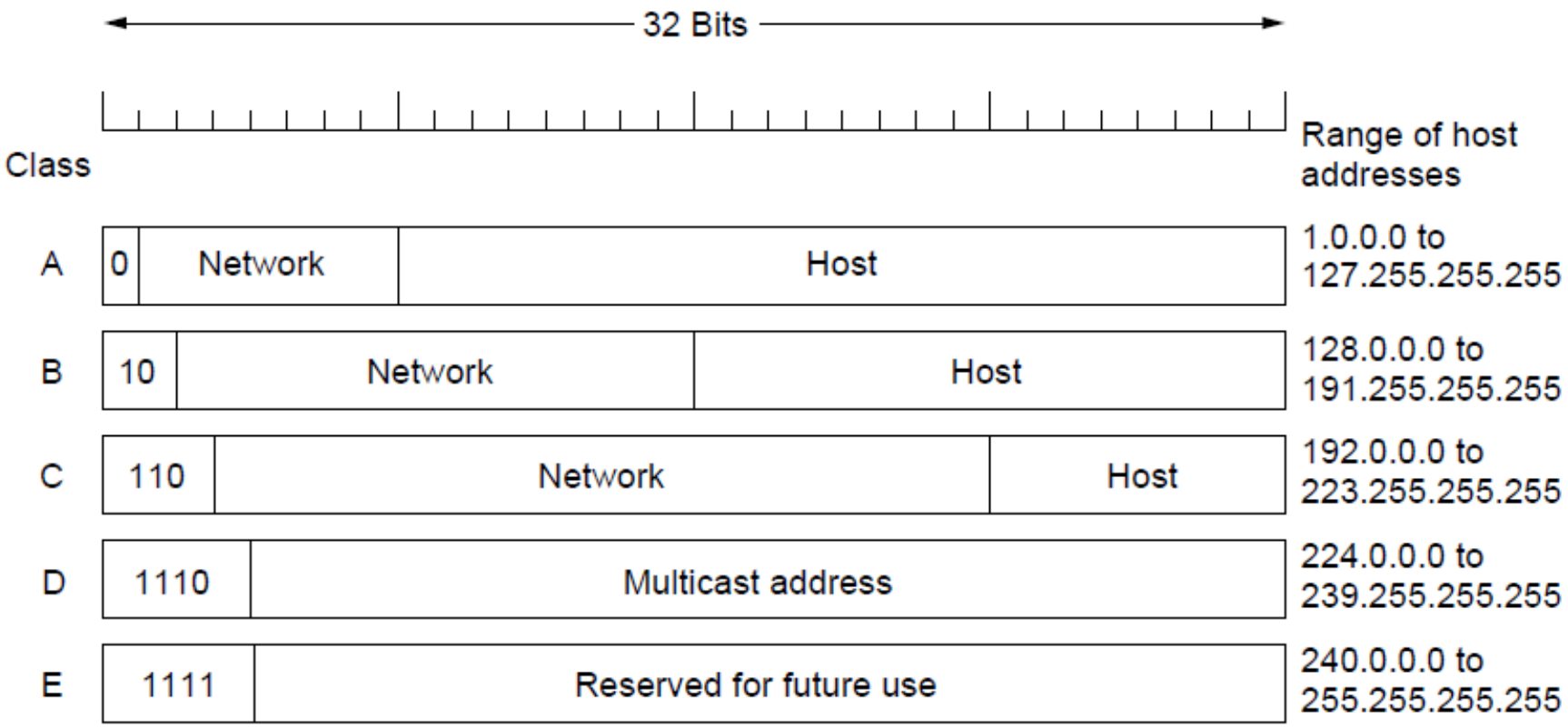
(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Sous-réseaux : Exemple (version détaillée)

- On choisit le découpage suivant
  - Computer Science : la moitié du réseau (128.208.128.0/17)
    - 128.208.128.0 à 128.208.255.255
    - 16 derniers bits en hexadecimal : 0x8000 à 0xFFFF
    - 2^{15} adresses
  - Electrical engineering : un quart du réseau (128.208.0.0/18)
    - 128.208.0.0 à 128.208.63.255
    - 16 derniers bits en hexadecimal : 0x0000 à 0x3FFF
    - 2^{14} adresses
  - Art : un huitième du réseau (128.208.96.0/19)
    - 128.208.96.0 à 128.208.127.255
    - 16 derniers bits en hexadecimal : 0x6000 à 0x7FFF
    - 2^{13} adresses
  - Un huitième du réseau non alloué
    - 128.208.64.0 à 128.208.95.255
    - 16 derniers bits en hexadecimal : 0x4000 à 0x5FFF
    - 2^{13} adresses

Structuration de l'espace d'adressage au niveau de l'Internet

- **Historiquement, découpage de l'espace d'adressage en 5 classes d'adresses (*classful addressing*)**



(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Structuration de l'espace d'adressage au niveau de l'Internet

- **Classe A**
  - 128 réseaux de 16 millions d'adresses chacun
- **Classe B**
  - 16384 réseaux de 65536 adresses chacun
- **Classe C**
  - 2 millions de réseaux avec 256 adresses chacun
- **Classe D**
  - Adresses de groupes de diffusion (*multicast*), non étudiés ici
- **Classe E**
  - Adresses réservées (usages expérimentaux …)

- **Certaines plages d'adresses au sein des classes A, B, C sont particulières (voir détails plus loin)**

Structuration de l'espace d'adressage au niveau de l'Internet

- **Avantage du découpage à base de classes**
  - **Simplicité de l'algorithme et des règles de routage des paquets**
- **Inconvénients**
  - **Rigide (et peut-être mal dimensionné)**
    - Les tailles des blocs sont pré-déterminées
    - Peut conduire à un gaspillage (exemple : la capacité d'un réseau de classe C est trop contrainte pour de nombreux besoins mais celle d'un réseau de classe B est trop importante)
    - Les sous-réseaux ne permettent pas de gérer toutes les situations
  - **Manque de passage à l'échelle**
    - Les routeurs du cœur de l'Internet (contrairement à ceux des bordures) doivent connaître les règles d'aiguillage vers l'ensemble des réseaux
    - Le découpage à base de classes conduit à l'explosion des tables de routage de ces routeurs « critiques »

Structuration de l'espace d'adressage au niveau de l'Internet

- **Un refonte nécessaire : CIDR (1993)**
  - ***Classless InterDomain Routing***

- **Principes**
  - **On n'utilise plus la notion de classes de réseaux pour le routage des paquets**
  - **On utilise un principe contraire (et complémentaire) à celui des sous-réseaux : l'agrégation de routes (on parle parfois de *supernet*)**
    - On factorise les informations de routage vers plusieurs préfixes
    - Différents routeurs (à différents niveaux d'un chemin) peuvent voir une même adresse IP comme appartenant à différents préfixes (de tailles différentes)
    - Les règles de configuration des routeurs sont plus complexes que celles de l'approche à base de classes

Structuration de l'espace d'adressage au niveau de l'Internet
CIDR - Exemple

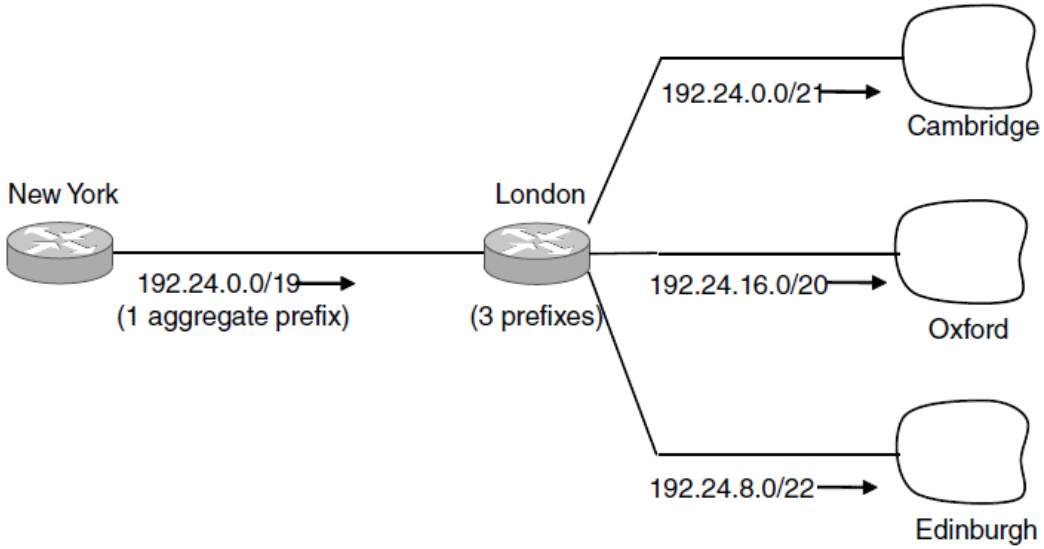
- **Considérons un bloc de 8192 adresses IP allouées à partir de l'adresse 194.24.0.0 selon l'historique suivant :**
  - **1) Besoin d'une plage de 2048 adresses pour Cambridge**
    - **On choisit 194.24.0.0/21**
  - **2) Plus tard, besoin d'une plage de 4096 adresses pour Oxford**
    - **Toutes les adresses dans cette plage doivent nécessairement avoir leurs 20 bits de poids forts en commun**
    - **On ne peut donc pas commencer à l'adresse 194.24.8.0**
    - **On choisit la plage 194.24.16.0/20**
  - **3) Encore plus tard, on a besoin d'une plage de 1024 adresses pour Edinburgh**
    - **On choisit 194.24.8.0/22**

University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

CIDR – Exemple (suite)

- **Du point de vue d'un routeur à New York, tous les paquets ayant pour destinataire une adresse comprise dans l'un des trois préfixes doivent être transférés vers Londres**
  - **Le routeur de NY peut donc se contenter d'une seule règle de routage pour les trois universités en l'associant au préfixe 194.24.0.0/19**



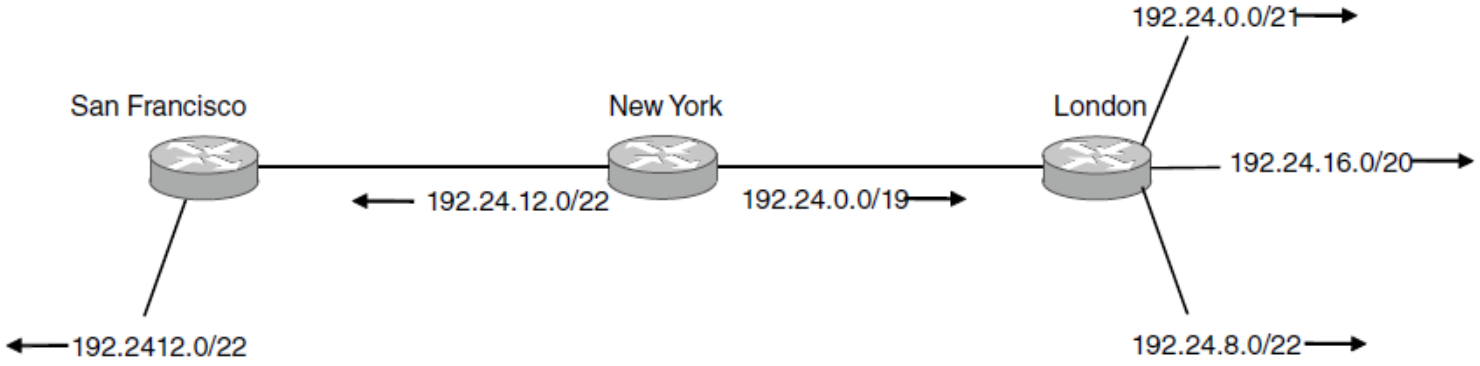
- **Remarque : l'agrégation est effectuée de manière automatique par les routeurs**

(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

CIDR – Exemple (suite et fin)

- **Optimisation supplémentaire :**
  - **Une même table de routage peut contenir des préfixes qui se recouvrent**
  - **Règle : si plusieurs préfixes correspondants sont trouvés dans une table de routage pour un destinataire, c'est le plus long préfixe qui s'impose**

- **Exemple : si le bloc précédemment libre est alloué à un réseau situé à San Francisco**



(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Compléments sur le fonctionnement d'un routeur

- **Le travail d'un routeur se décompose en deux aspects principaux, qui sont complémentaires**
- ***Forwarding***
  - **Retransmission d'un paquet à partir d'un port d'entrée vers un port de sortie.**
  - **Le choix de l'interface réseau à utiliser en sortie se base sur le contenu de tables d'aiguillage.**
- ***Routing* :**
  - **Construction et maintenance des tables nécessaires pour le forwarding.**
  - **Ces tables sont ajustées au cours du temps, en fonction de l'évolution de la structure du réseau et du trafic.**
  - **Nécessite des dialogues entre routeurs**
  - **Cet aspect ne sera pas étudié dans ce cours. Se référer à la bibliographie.**

Compléments sur le fonctionnement d'un routeur (suite)

- **Algorithme de *forwarding* (très simplifié) d'un routeur IPv4**
- **Version 1 : sans CIDR**
  - **Identifier le réseau de destination à partir de l'adresse de destination (contenue dans le paquet IP)**
    - Possible grâce à la connaissance des classes d'adresses (et des différents réseaux au sein de chaque classe)
  - **Si l'une des interfaces de sortie du routeur appartient au réseau de destination, alors envoyer le paquet sur cette interface**
  - **Sinon, consulter la table d'aiguillage**
    - Chercher une entrée qui correspond au réseau de destination
    - Si entrée trouvée, utiliser l'interface de sortie indiquée dans l'entrée
    - Sinon, utiliser l'interface de sortie associée à la route par défaut

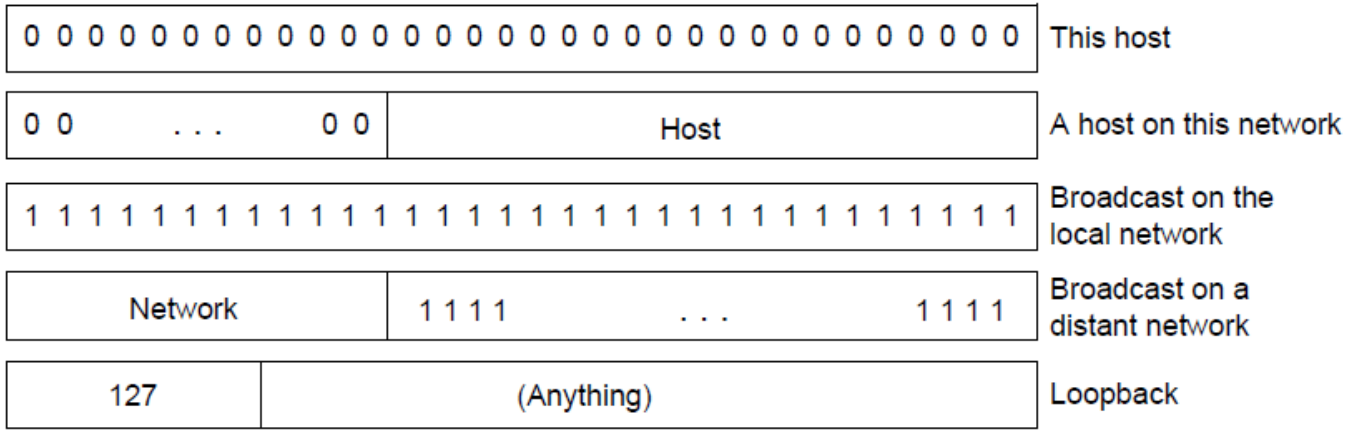
Compléments sur le fonctionnement d'un routeur (suite)

- **Algorithme de *forwarding* (très simplifié) d'un routeur IPv4**
- **Version 2 : avec CIDR**
- **Particularités**
  - **Les préfixes peuvent être de tailles très variables**
  - **Recouvrement de préfixes possible**
- **Principe**
  - **Récupérer l'adresse de destination du paquet entrant**
  - **Rechercher dans la table toutes les définitions de préfixes qui peuvent convenir**
  - **Si au moins une définition convient**
    - Choisir l'entrée qui est associée au préfixe convenable le plus long
    - Et utiliser l'interface de sortie indiquée dans cette entrée
  - **Si aucune entrée de la table ne convient, utiliser l'interface de sortie associée à la route par défaut**

Compléments sur le fonctionnement d'un routeur (suite)

- **Algorithme de *forwarding* (très simplifié) d'un routeur IPv4**
- **Version 2 : avec CIDR**
- **Exemple (simplifié)**
  - **Table avec 3 entrées**
    - Entrée 1 : 171.69.0.0/16
    - Entrée 2 : 171.69.10.0/24
    - Entrée 3 : route par défaut
  - **Cas 1 : adresse de destination 171.69.10.5**
    - Les entrées 1 et 2 conviennent
    - On choisit la route indiquée dans l'entrée 2 (plus long préfixe)
  - **Cas 2 : adresse de destination 171.69.20.5**
    - On choisit la route indiquée dans l'entrée 1
  - **Cas 3 : adresse de destination 199.28.39.157**
    - On choisit la route par défaut

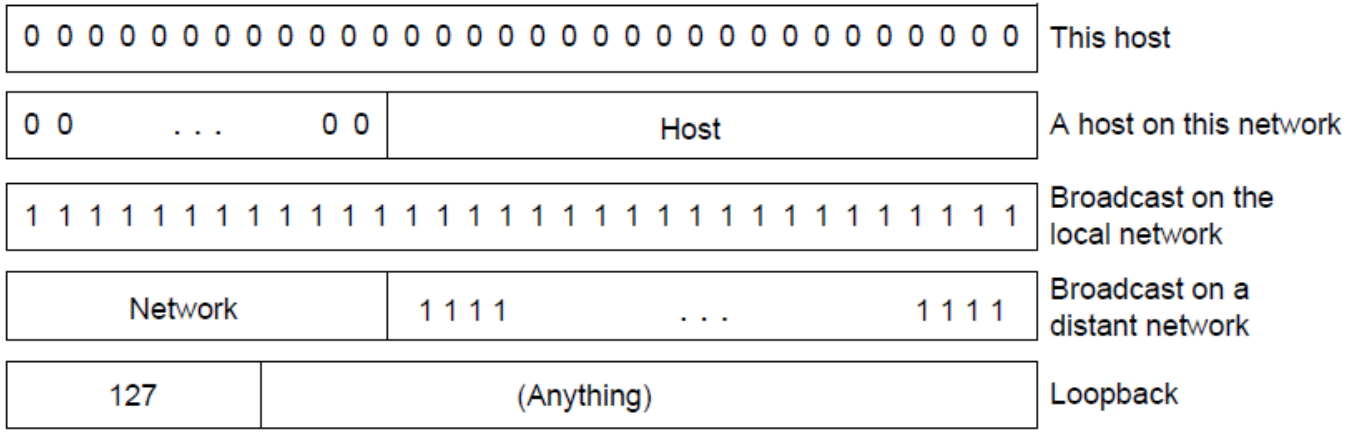
Adresses IPv4 particulières



(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

- **Et aussi :**
  - **Plages d'adresses privées (non prises en charge par les routeurs), utiles pour les réseaux locaux (voir plus loin) :**
    - 10.0.0.0/8 (capacité : 16 777 216 adresses)
    - 172.16.0.0/12 (capacité : 1 048 576 adresses)
    - 192.168.0.0/16 (capacité : 65 536 adresses)
  - **Il existe quelques plages réservées à des besoins spécifiques**
    - Par exemple la plage 169.254.0.0/16 pour certaines allocations automatiques d'adresses (non routables) dans les réseaux locaux

Adresses IPv4 particulières



(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

- **Et aussi :**
  - **Plages d'adresses privées (non prises en charge par les routeurs), utiles pour les réseaux locaux (voir plus loin) :**
    - 10.0.0.0/8 : 10.0.0.0 à 10.255.255.255
    - 172.16.0.0/12 : 172.16.0.0 à 172.31.255.255
    - 192.168.0.0/16 : 192.168.0.0 à 192.168.255.255
  - **Il existe quelques plages réservées à des besoins spécifiques**
    - Par exemple la plage 169.254.0.0/16 pour certaines allocations automatiques d'adresses (non routables) dans les réseaux locaux

Adresses IPv4 particulières (suite)

- **Remarque : Dans la plupart des cas, on ne peut pas attribuer une adresse se terminant par .0 ou .255 à une interface**
  - **Pour éviter les ambiguïtés (entre identifiant de réseau et adresse d'une interface), les adresses ayant leur octet de poids faible à 0 ne sont généralement pas utilisées**
  - **Pour la plupart des réseaux, une adresse ayant son octet de poids faible à 255 correspond à une adresse de diffusion sur l'ensemble du réseau (cf. diapositive précédente)**

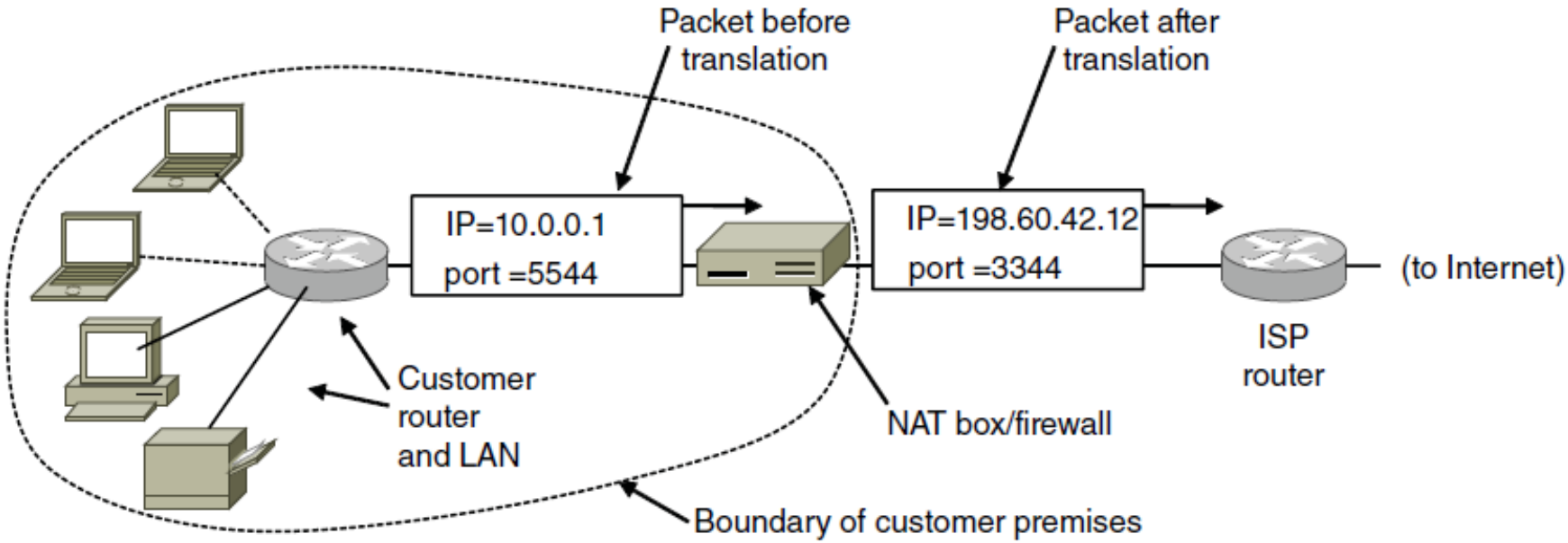
- **Contre-exemples :**
  - **Dans le réseau 192.168.0.0/16 (attention, la taille a son importance), on peut attribuer les adresses suivantes à des interfaces :**
    - 192.168.1.0, 192.168.2.0, etc. (mais pas 192.168.0.0)
    - 192.168.0.255, 192.168.1.255, etc. (mais pas 192.168.255.255)
  - **À éviter dans la mesure du possible car sources de confusion (et problématiques avec certains logiciels)**

Network Address Translation (NAT)

- **Problème de départ : rareté/raréfaction des adresses IP (v4)**
  - **Comment permettre à M machines d'un même réseau de communiquer (en même temps) avec des machines sur l'Internet alors que l'on ne dispose pas d'autant d'adresses IP publiques ?**

- **Solution: réseau privé + passerelle NAT**
  - **On attribue des adresses IP privées aux machines du réseau local**
  - **On utilise une passerelle NAT comme point de passage du trafic arrivant de (et sortant vers) l'Internet**
  - **La passerelle NAT peut être vue comme un routeur (elle dispose d'une adresse IP dans le réseau privé et d'une adresse IP publique)**
  - **Contrairement à un « simple » routeur, la passerelle NAT modifie le contenu des paquets qu'elle reçoit**
    - Paquet sortant : elle remplace l'@ IP privée de l'émetteur par l'@ IP publique du NAT
    - Paquet entrant : elle remplace l'@ IP destinatrice (l'@ IP publique du NAT par celle du vrai destinataire dans le réseau privé)

Network Address Translation (NAT)



(source : A. Tanenbaum et D. Wetherall. Computer Networks, 5th edition, Pearson Education)

Network Address Translation (NAT)

- **Question**
  - Comment la passerelle NAT peut-elle déterminer vers quelle adresse privée rediriger un paquet entrant ?
  - En fait, la passerelle NAT ne consulte/remplace pas seulement les adresses IP mais également les numéros de ports (TCP/UDP) des messages qu'elle retransmet
- **1er cas : Un paquet entrant P_2 arrive en réponse à un paquet sortant P_1**
  - Lors de la retransmission de P_1, la passerelle NAT ne modifie pas seulement l'@ IP source mais également le numéro de port (TCP/UDP) source et elle mémorise la correspondance entre, d'une part, le numéro de port modifié et, d'autre part, l'@ IP et le numéro de port de la machine émettrice sur le réseau privé
  - Lors de la réception de P_2, la passerelle NAT peut, grâce au numéro de port de destination, retrouver vers quelle machine et quel numéro de port rediriger P_2
  - Remarque : la passerelle NAT doit nécessairement modifier le numéro de port de P_1 car plusieurs machines du réseau privé peuvent envoyer des paquets avec le même numéro de port source

Network Address Translation (NAT)

- **2ème cas : Un paquet entrant P_2 n'arrive pas en réponse à un paquet sortant P_1**
  - **Dans ce cas, la passerelle NAT doit être configurée avec des règles, basées sur le type de protocole de transport (TCP/UDP) et le numéro de port destination, lui permettant de savoir vers quelle machine (et éventuellement port) du réseau interne le paquet doit être redirigé (on parle de redirection de port – *port forwarding*)**
  - **Remarques**
    - Cette technique est nécessaire si l'on a une application serveur sur une machine du réseau privée que l'on souhaite rendre joignable par des clients provenant de l'Internet
    - Un numéro de port destination (TCP ou UDP) ne peut être redirigé que vers une seule machine du réseau privé
- **La technique du NAT est très utilisée en pratique pour des cas simples … mais elle présente de nombreuses restrictions qui la rendent critiquable**
  - **Sur le plan pratique (performances, fiabilité, fonctionnalités avancées)**
  - **Sur le plan conceptuel (non respect du modèle en couches de l'Internet)**

IPv6 : Une nouvelle version du protocole IP

- **Motivations**
  - Résoudre la pénurie d'adresses IPv4 (inéluctable, malgré l'aide de mécanismes comme CIDR et NAT) et fournir un espace d'adressage adapté à l'explosion du nombre de machines sur l'Internet
  - D'autres aspects (non détaillés ici) : améliorer la gestion du routage, de la sécurité, des types de services différenciés (notamment pour les applications temps-réel), de la diffusion (multicast), de la mobilité ...

- **Protocole développé dans les années 1990 et standardisé en 1998**
  - Les implémentations nécessaires au niveau matériel et logiciel sont disponibles
  - ... mais IPv6 est encore très faiblement déployé
  - L'état du déploiement est très hétérogène selon les régions du monde et les domaines d'applications

IPv6 : Une nouvelle version du protocole IP

- IPv6 peut cohabiter avec IPv4 mais n'est pas directement compatible/interopérable. Il existe cependant des mécanismes de transition
- Au niveau des systèmes d'exploitation
  - Piles protocolaires doubles ou hybrides
  - Une pile hybride offre une API unifiée aux applications pour manipuler des adresses IP (v4 ou v6), en s'alignant sur l'adressage IPv6
- Au niveau du réseau
  - Mécanismes de « tunnels » pour encapsuler un paquet IPv6 dans un paquet IPv4 afin de pouvoir traverser des portions de chemins ne gérant qu'IPv4 (et vice versa)
  - Passerelles (au niveau applicatif ou au niveau réseau/NAT) pour permettre des interactions entre une machine utilisant uniquement IPv4 et une machine utilisant uniquement IPv6

Adressage IPv6

- **Une adresse IPv6 est codée sur 128 bits (16 octets)**
- **Notation**
  - **Différente de la notion (décimale pointée) utilisée par IPv4**
  - **8 groupes de 4 chiffres hexadécimaux chacun, séparés par « : »**
  - **Exemple : 47cd:1234:4422:0000:0000:ac02:a456:0124**
  - **Optimisations**
    - On peut omettre les chiffres 0 au début d'un groupe
    - On peut omettre un groupe (ou plusieurs groupes consécutifs) dont tous les bits sont à 0 (en laissant cependant les « : » adjacents)
    - Exemple revisité : 47cd:1234:4422::ac02:a456:124
    - Remarque : on ne peut pas avoir plus d'une seule occurrence de « :: » (risque d'ambiguïté)
- **DNS**
  - **Définition d'un nouveau type d'enregistrement pour les adresses IPv6 : « AAAA » (par opposition à « A » pour les adresses IPv4)**

API hybrides pour la programmation réseau IPv4/IPv6 (Unix / langage C)

- Les structures de données et primitives étudiées dans les cours/TP précédents gèrent uniquement IPv4. On liste ci-après les API utiles pour manipuler des adresses IPv6 (ou aussi v4)

- Sockets
  - `struct sockaddr_in6` (au lieu de `struct sockaddr_in`)
  - domaine `AF_INET6` (au lie d'`AF_INET`)
- Conversion entre codage binaire et codage textuel d'une adresse
  - `inet_pton()` (au lieu d'`inet_aton()`) et `inet_ntop()` (au lieu d'`inet_ntoa()`)
- Correspondance entre nom d'hôte et adresse IP (DNS)
  - `getaddrinfo()` (au lieu de `gethostbyname()`) et `getnameinfo()` (au lieu de `gethostbyaddr()`)

- Voir la documentation pour plus de détails
  - Pages de manuel (notamment : man 7 socket, man 7 ip, man 7 tcp ...)
  - http://tldp.org/HOWTO/Linux+IPv6-HOWTO/chapter-section-using-api.html
  - M. Kerrisk, The Linux Programming Interface, No Starch Press (chap. 58-59)
  - Ian Burell, IPv6 for programmers. http://opensourcebridge.org/wiki/2014/IPv6_for_Programmers