IPv4

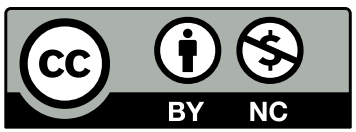
Pierre David
pda@unistra.fr

Université de Strasbourg – Master CSMI

2023 – 2024

Plan

Licence d'utilisation



Plan

Adressage

Adresse IPv4 = 32 bits

Adresse IPv4 = 4 nombres séparés par des points (exemple : 130.79.201.195)

Adressage

Une adresse IP est découpée en :

- un numéro de réseau
- un numéro de machine à l'intérieur du réseau

Historiquement (1983), 3 principales classes d'adresses :

Classe A	1.x.y.z $\rightarrow$ 127.x.y.z	premiers bits = 0
	127 réseaux / 16777216 machines	
Classe B	128.0.y.z $\rightarrow$ 191.255.y.z	premiers bits = 10
	16384 réseaux / 65536 machines	
Classe C	192.0.0.z $\rightarrow$ 223.255.255.z	premiers bits = 110
	2097152 réseaux / 256 machines	

... plus les classes D (*multicast*) et E (*expérimentale*)

$\Rightarrow$ classes obsolètes (depuis les années 1990)

Adressage

Dans tout réseau IP, deux adresses spéciales

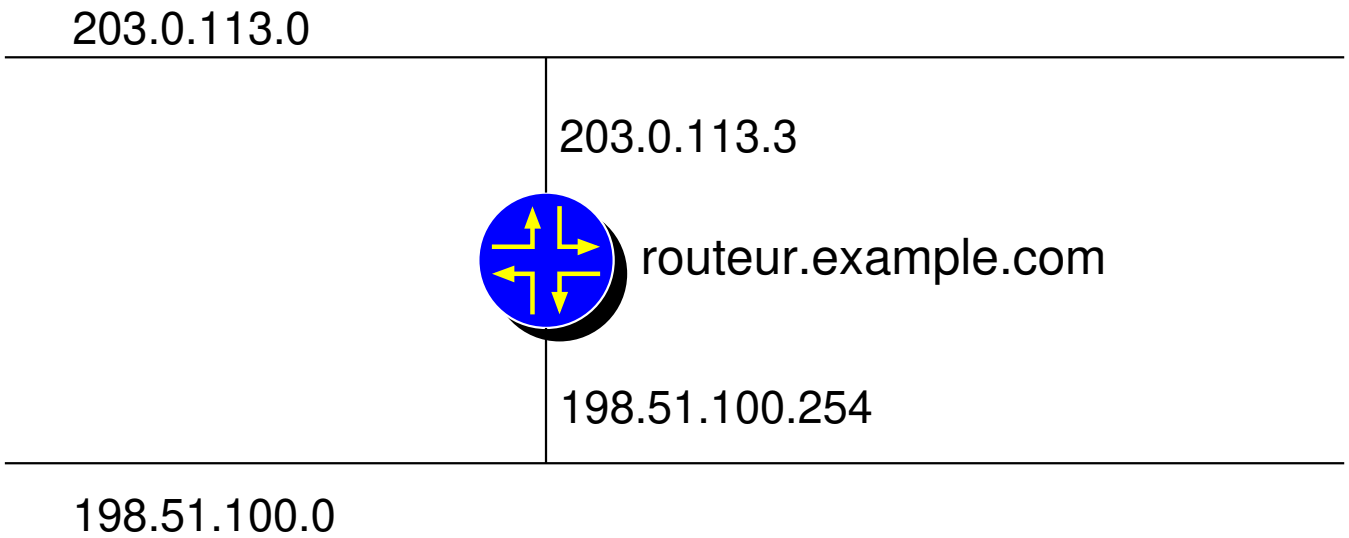
- numéro de machine = tous les bits à 0
  $\Rightarrow$ identifie le réseau lui-même
- numéro de machine = tous les bits à 1
  $\Rightarrow$ adresse de *broadcast*

Réseau 127 : *loopback*

Adressage

Une adresse IP est associée à une interface

Exemple : le routeur routeur.example.com a deux interfaces, il a donc deux adresses IP : 198.51.100.254 et 203.0.113.3



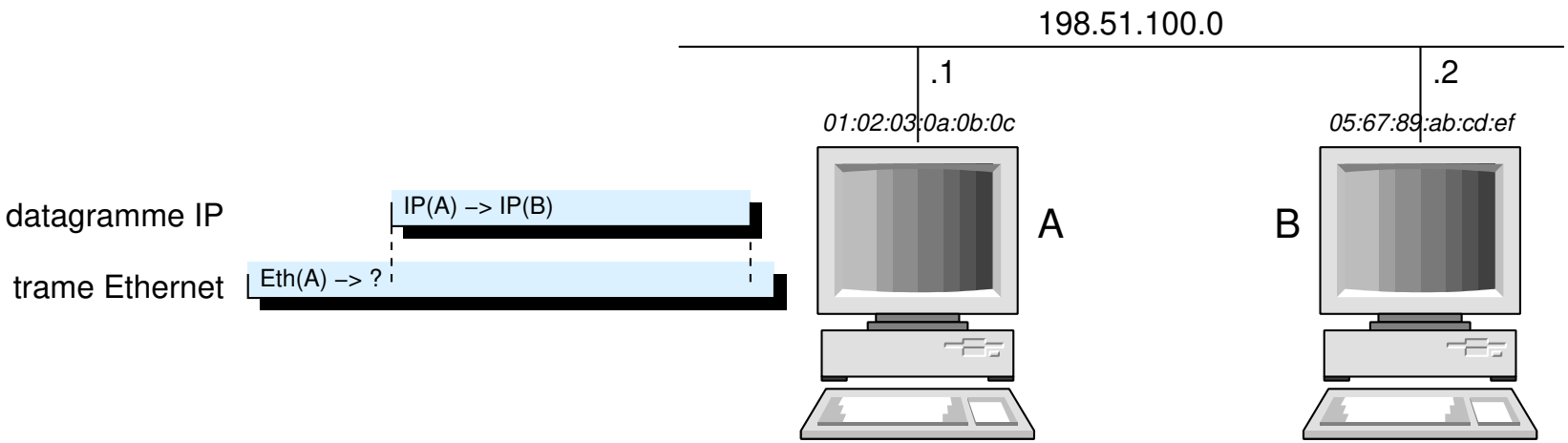
Plan

Résolution d'adresses

- adresse physique : généralement associée à l'interface matérielle
  - ex : l'adresse Ethernet ou 802.11 est codée par le constructeur
- adresse IP : attribuée par l'administrateur du réseau

Si A veut communiquer avec B :

- A construit un datagramme IP avec comme destinataire l'adresse IP de B
- pour construire la trame Ethernet, il faudrait connaître l'adresse Ethernet de B



Résolution d'adresses

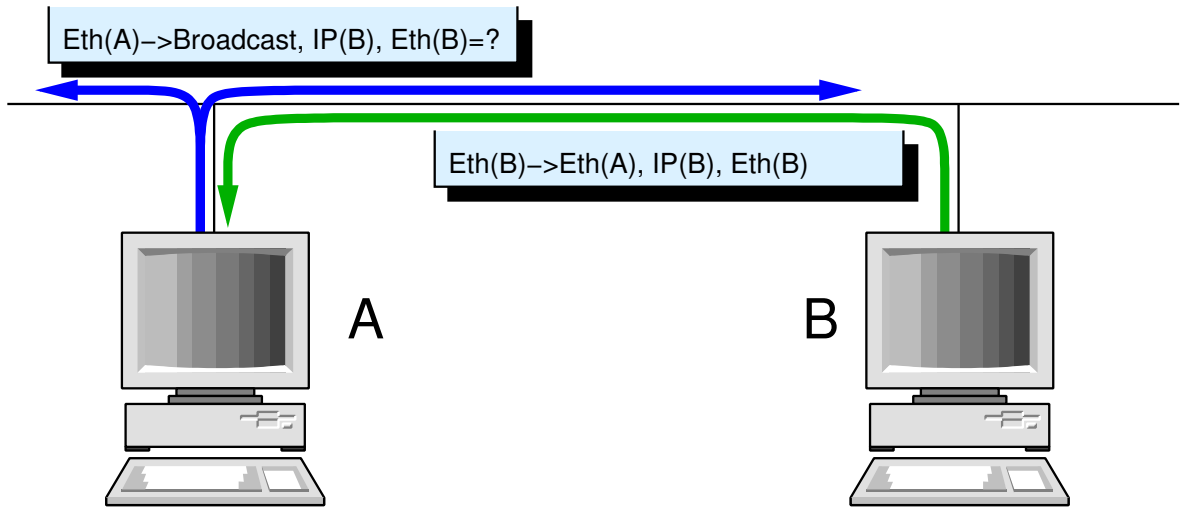
Résolution d'adresse = déterminer l'adresse physique correspondant à une adresse IP

Deux catégories de méthodes :

1. résolution statique (table)
2. résolution dynamique

Résolution d'adresses

Exemple de résolution dynamique : Ethernet, protocole ARP (*Address Resolution Protocol*)



1. Message (broadcast Ethernet) de A
   Question = étant donné IP(B), que vaut Eth(B) ?
2. Message de B à A
   Réponse = voici mon adresse Ethernet (Eth(B))

Résolution d'adresses

Naïvement : un paquet IP envoyé $\Rightarrow$ un échange ARP (1 broadcast + 1 réponse)

Problème : trafic énorme

Solution : chaque machine conserve les dernières transactions dans un cache

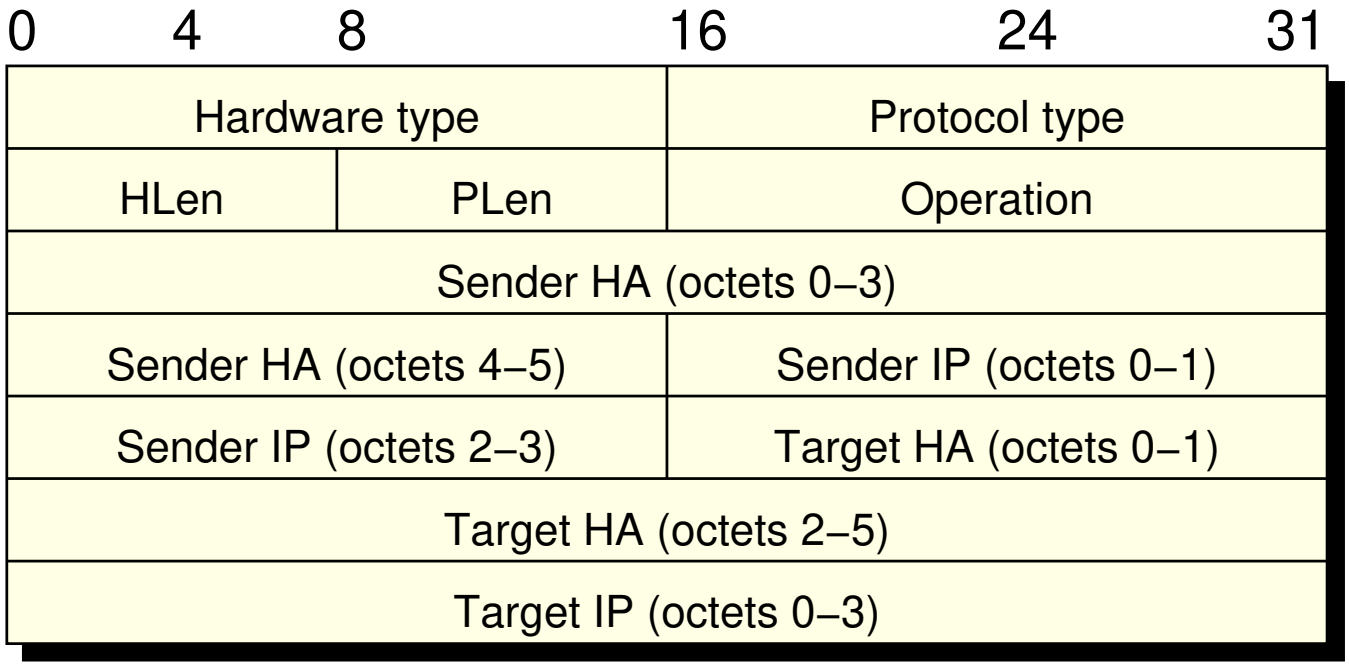
Commande `arp`

La commande `arp` affiche le cache ARP :

```
> /sbin/arp
Address                  HWtype  HWaddress           Flags Mask            Iface
fbsd.unistra.fr          ether   52:54:00:7c:a0:3a   C                     virbr0
elephant.unistra.fr      ether   00:11:32:5b:4a:bd   C                     eno1
fwicube.unistra.fr       ether   00:00:5e:00:2c:f0   C                     eno1
```

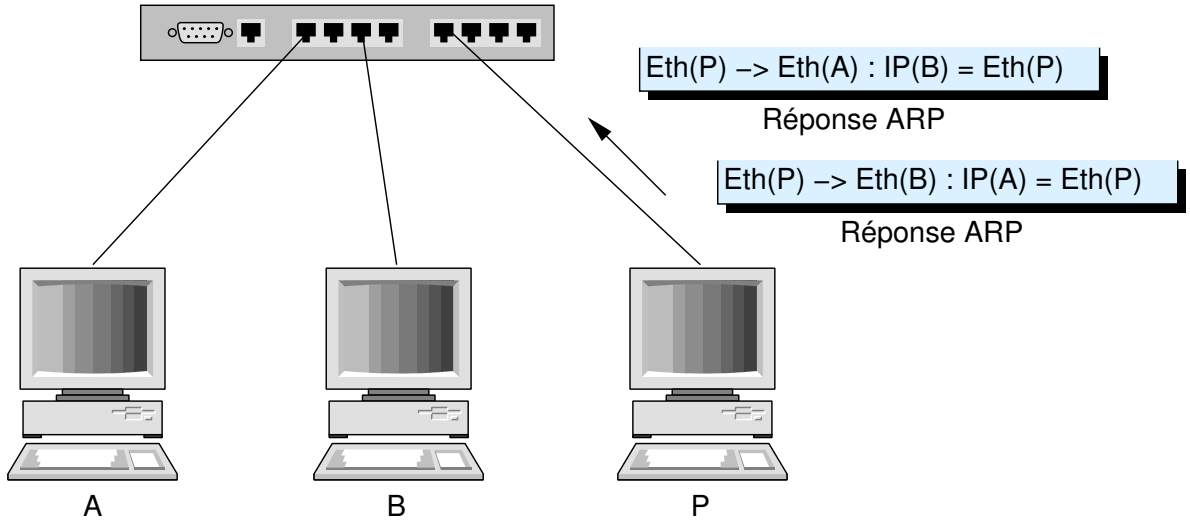
L'administrateur peut également modifier le cache (ajouter ou supprimer une entrée) avec cette commande.

Résolution d'adresses



L'écoute du trafic réseau

Duperie sur les résolutions d'adresses (« *ARP poisoning* ») :



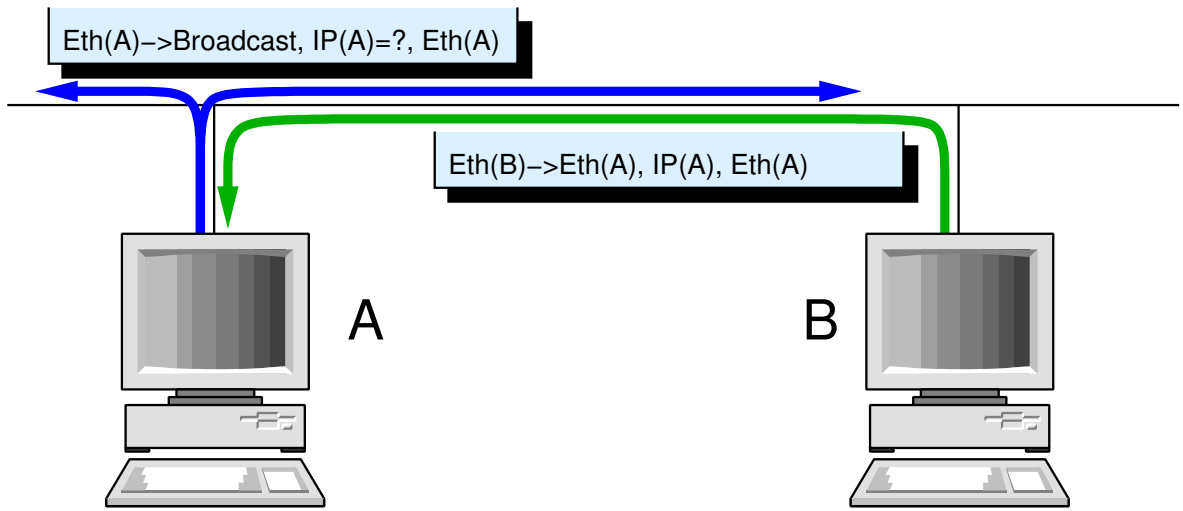
- le pirate envoie une réponse ARP à A
  $\Rightarrow$ réponse sans requête = « *gratuitous ARP* »
  $\Rightarrow$ A actualise son cache ARP : tout datagramme à destination de IP(B) sera envoyé à Eth(P)
- le pirate envoie une réponse ARP à B
  $\Rightarrow$ B actualise son cache ARP : tout datagramme à destination de IP(B) sera envoyé à Eth(P)

Résolution d'adresses

Problème inverse : "qui suis-je" ?

Comment associer une adresse IP à une adresse Ethernet (stations sans disque, terminaux X Window, serveurs de terminaux, etc.) ?

Protocole RARP (Reverse Address Resolution Protocol)



1. Message (broadcast Ethernet) de A
   Question = étant donné Eth(A), que vaut IP(A) ?
2. Message de B à A
   Réponse = voici ton adresse IP (Eth(A))

Résolution d'adresses

Problèmes de RARP :

- retourne peu d'information
  $\Rightarrow$ protocoles supplémentaires pour obtenir serveur DNS, de fichiers, de swap, routeur par défaut, etc.
- protocole pas basé sur IP
  $\Rightarrow$ pas d'utilisation des couches basses

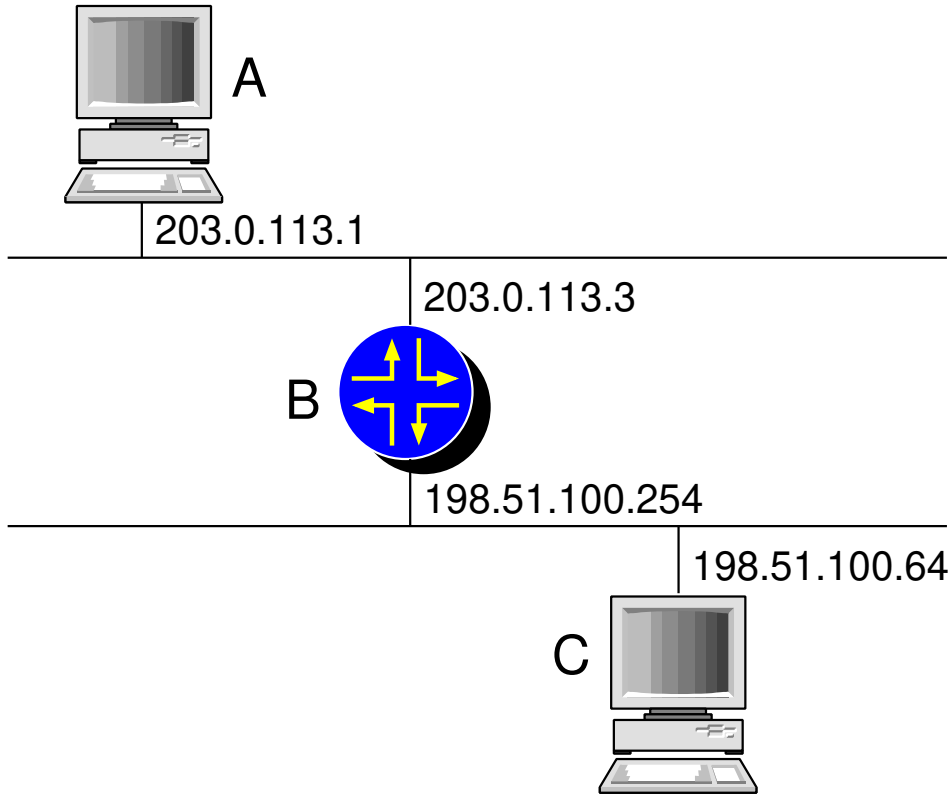
D'où :

- RARP est obsolète
- Protocoles plus récents
  (BOOTP et DHCP)

Plan

Routage

Comment A communique avec C ?

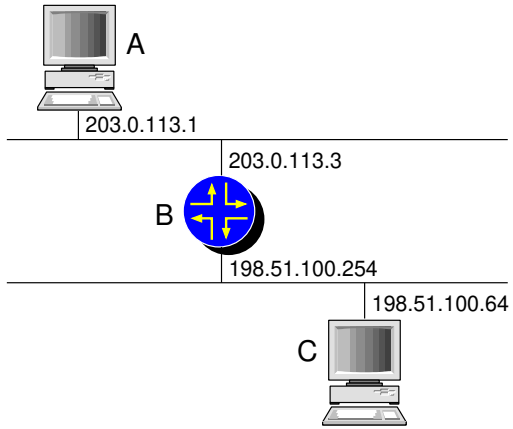


A ne peut pas utiliser ARP pour trouver l'adresse Ethernet de C

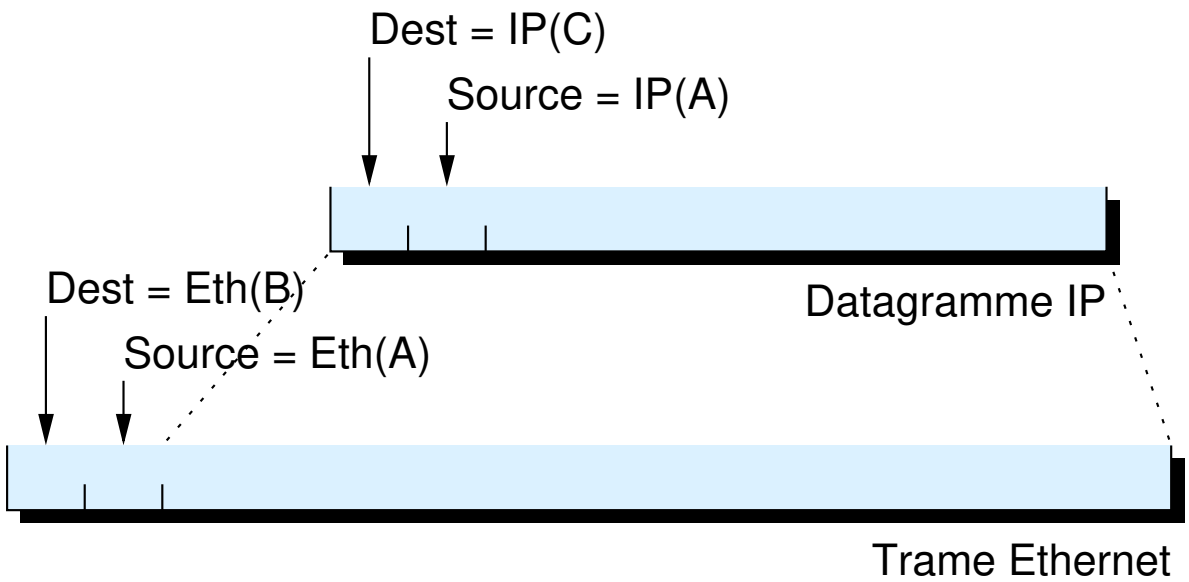
B sait communiquer avec C $\Rightarrow$ A envoie le message à B

Routage

Routage de A vers C via B



Message envoyé par A :



Routage

Décision : table de routage

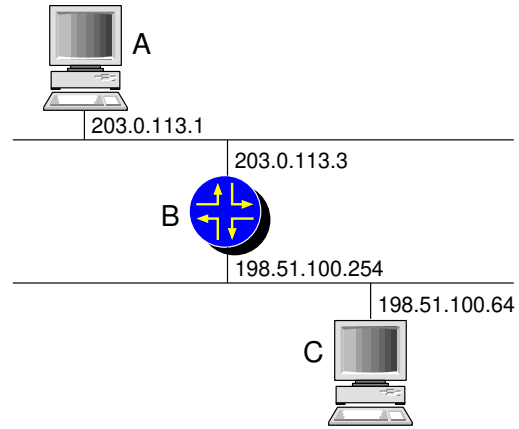


Table de routage de A :

Pour aller à	Passer par
203.0.113.0	(envoi direct)
198.51.100.0	203.0.113.3

Routage

Algorithme :

```
void envoyer (datagramme_t datagramme) {
    IPdest = adresse_destination (datagramme) ;
    if (reseau (IPdest) == reseau (IPmoi)) {
        Eth = ARP (IPdest) ;
    } else {
        IProuteur = chercher_route (reseau (IPdest))
        if (non_trouve (IProuteur)) {
            IProuteur = chercher_route (0.0.0.0) ;
            if (non_trouve (IProuteur))
                erreur () ;
        }
        Eth = ARP (IProuteur) ;
    }
    encapsuler_trame (Eth, datagramme) ;
}
```

Routage

Généralisation :

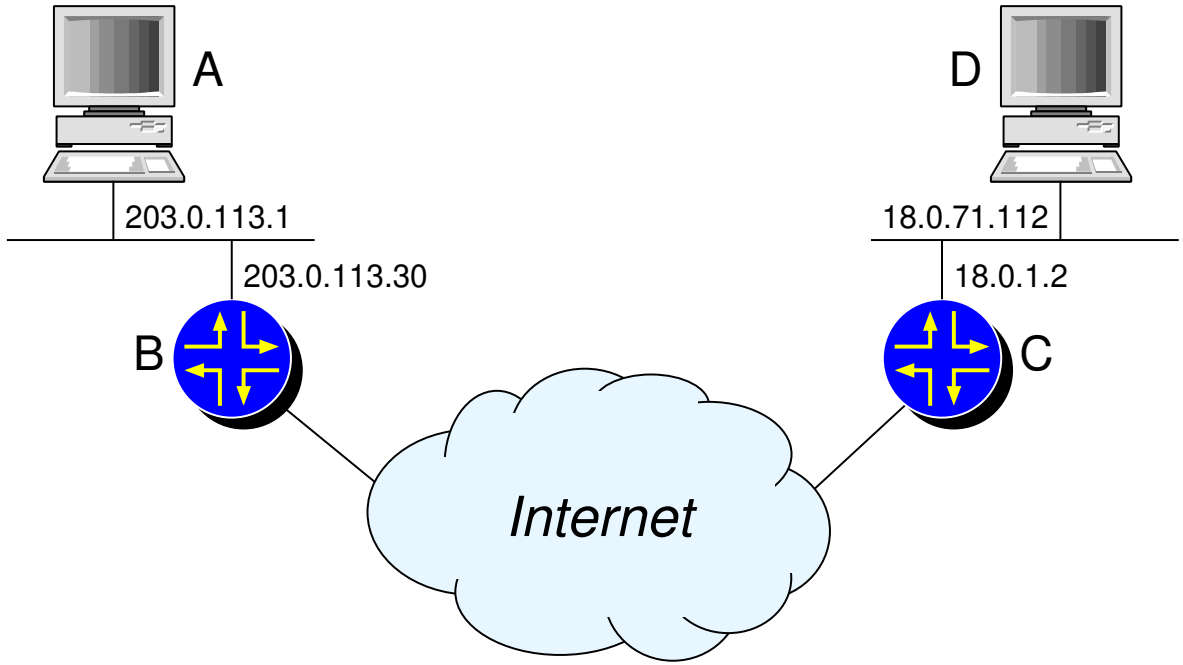


Table de routage de A :

Pour aller à	Passer par
203.0.113.0	(envoi direct)
18.0.0.0	203.0.113.30

Routage

Problèmes :

- explosion des tables de routage
- actualisation des tables de routage

$\Rightarrow$ route par défaut

Routage

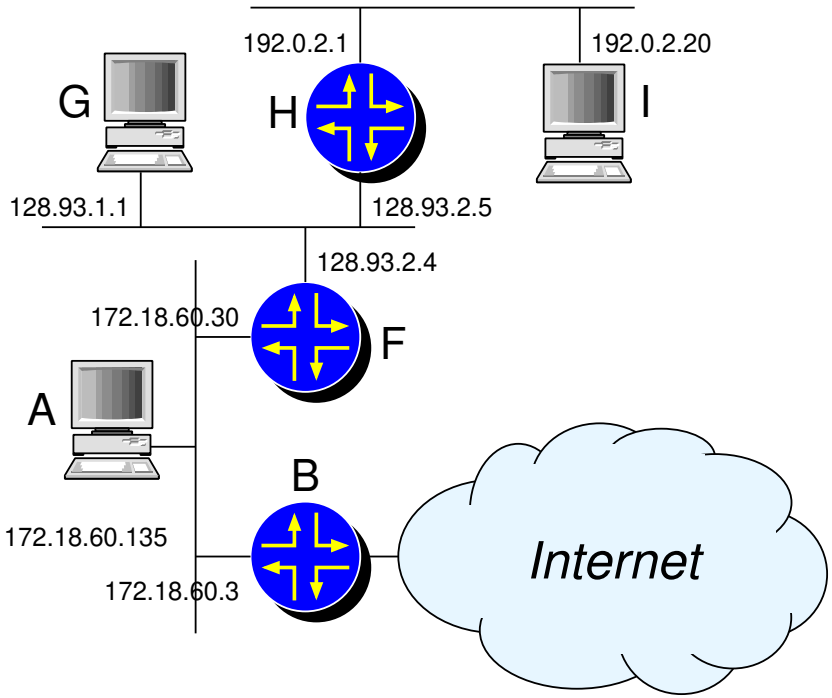


Table de routage de A :

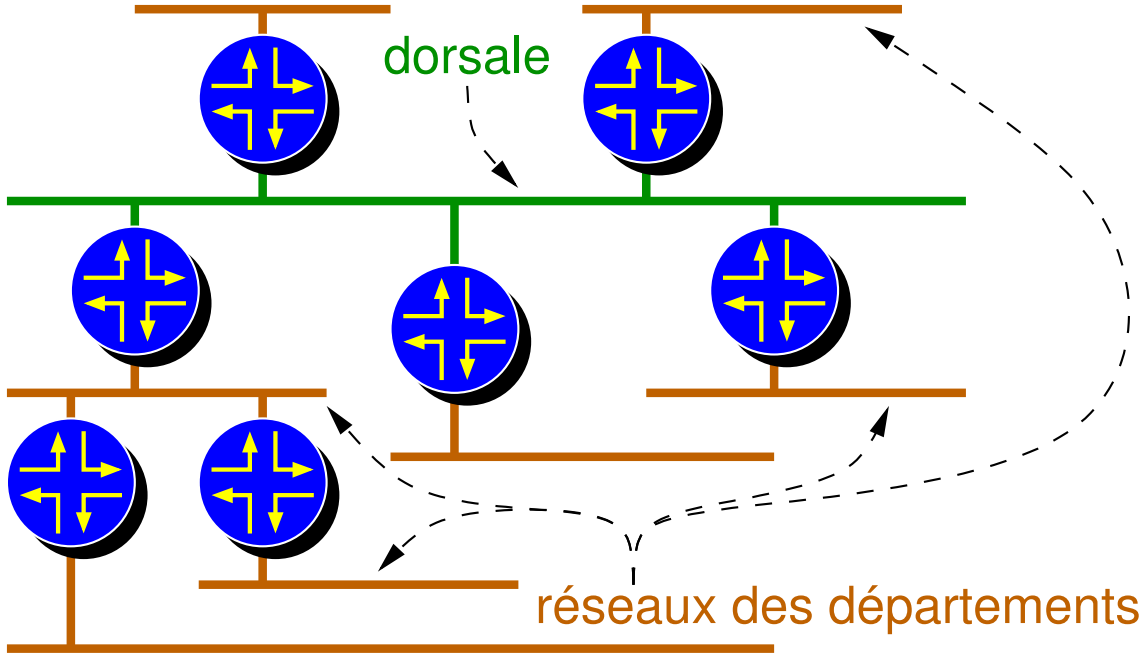
Pour aller à	Passer par
172.18.0.0	172.18.60.135
128.93.0.0	172.18.60.30
192.0.2.0	172.18.60.30
défaut	172.18.60.3

Plan

Subnetting

Constat : un site ne contient pas *un* réseau, mais un *ensemble* de réseaux

Exemple : un site est composé d'une dorsale fédérant un ensemble de réseaux de départements



Subnetting

Problèmes :

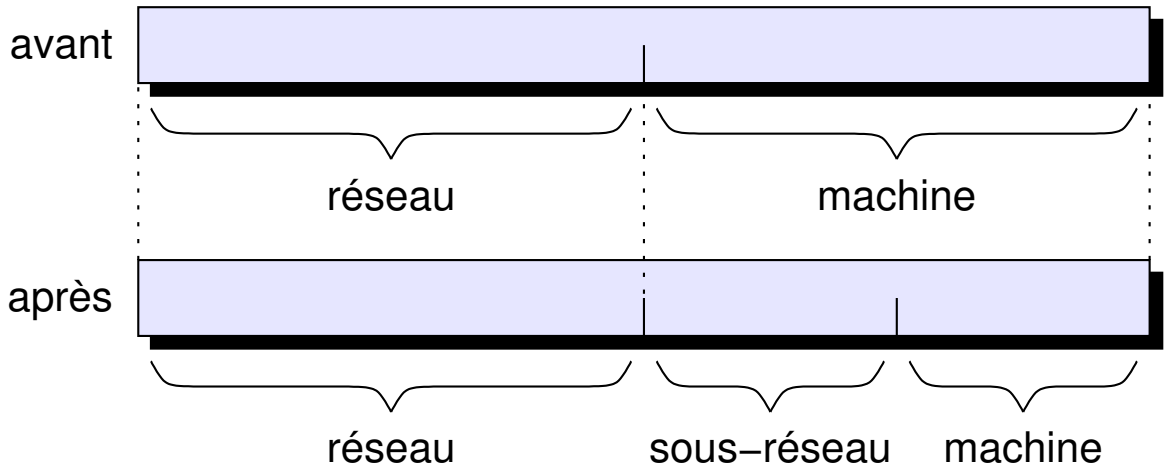
- classes A et B non adaptées
- inflation de classes C

Rappel :

- une adresse IP = 32 bits
- une adresse IP identifie une machine
- une adresse IP est constituée du numéro de réseau et du numéro de machine dans ce réseau

Subnetting

Solution : scinder une classe en sous-réseaux
$\Rightarrow$ la partie *numéro de machine* devient le numéro de sous-réseau et le numéro de machine dans ce sous-réseau



Nombre configurable de bits alloués au numéro de *sous-réseau*
$\Rightarrow$ *subnet mask*

Note : obsolète depuis CIDR (milieu des années 1990)

Subnetting

Subnet mask : masque binaire qui définit la séparation entre numéro de sous-réseau et numéro de machine
Fonctionnement : étant donnée une adresse *a*

- *a* & *subnet mask* = numéro de réseau
- *a* & ˜ *subnet mask* = numéro de machine

$\Rightarrow$ subnet mask = généralisation de la notion de classe

Exemple : *subnet-mask* = 0xffffff80 (= 255.255.255.128)
$\Rightarrow$ partie *sous-réseau* = 9 bits et partie *machine* = 7 bits

Subnetting

Routage en présence de subnetting :

- à l'extérieur du site :
  La décision de routage vers le site est fonction uniquement de la partie *réseau* de l'adresse
- à l'intérieur du site :
  La décision de routage vers une machine du site est fonction des parties *réseau* et *sous-réseau* de l'adresse
  La machine *a* envoie un paquet pour *b* :
  - directement à *b* si $a \,\&\, s = b \,\&\, s$
  - à la passerelle correspondante trouvée dans la table de routage si $a \,\&\, s \neq b \,\&\, s$

Subnetting

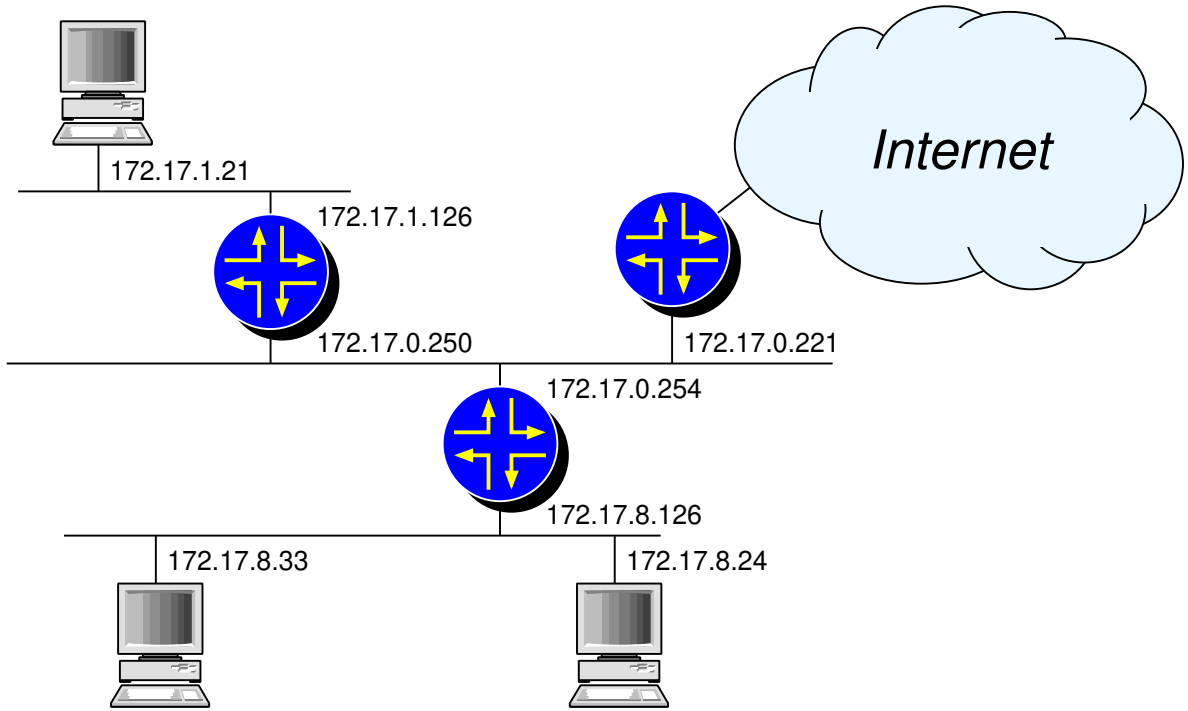
Algorithme :

```
void envoyer (datagramme_t datagramme) {
    IPdest = adresse_destination (datagramme) ;
    if ((IPdest & subnet_mask) == (IPmoi & subnet_mask)) {
        Eth = ARP (IPdest) ;
    } else {
        IProuteur = chercher_route (IPdest)
        if (non_trouve (IProuteur))
            erreur () ;
        Eth = ARP (IProuteur) ;
    }
    encapsuler_trame (Eth, datagramme) ;
}

adresse_t chercher_route (adresse_t IPdest) {
    masque = 0xffffffff ;
    do {
        a = chercher_table (IPdest & masque) ;
        if (trouve (a))
            return a ;
        masque <<= 1 ;
    } while (masque != 0) ;
    return -1 ;
}
```

Subnetting

Exemple (avec un subnet-mask = 0xffffff80) :



172.17.8.33 $\rightarrow$ 172.17.8.24 $\Rightarrow$ même réseau

172.17.8.33 $\rightarrow$ 172.17.1.21 $\Rightarrow$ réseau différent

172.17.8.33 $\rightarrow$ 18.71.0.38 $\Rightarrow$ réseau différent

Subnetting

Table de routage de 172.17.8.33 :

Pour aller à	Passer par
172.17.8.0	172.17.8.33
défaut	172.17.8.126

Table de routage du routeur 172.17.8.126/172.17.0.254 :

Pour aller à	Passer par
172.17.8.0	172.17.8.126
172.17.0.0	172.17.0.254
172.17.1.0	172.17.0.250
défaut	172.17.0.221

Subnetting

Adresses spéciales :

- numéro de machine = tous les bits à 0
  $\Rightarrow$ identifie le sous-réseau lui-même
- numéro de machine = tous les bits à 1
  $\Rightarrow$ adresse de *broadcast*

Plan

CIDR

CIDR = *Classless Inter-Domain Routing*

Au début des années 1990, réflexion sur l'épuisement des adresses IPv4 :

- solution à long terme : nouveau protocole IPng (IPv6)
- solution à court terme : CIDR

CIDR

CIDR était aussi appelé au début « *supernetting* »

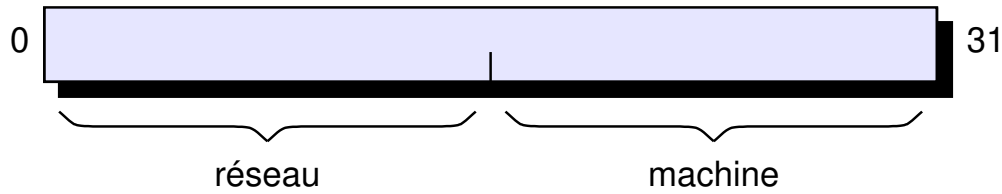
Pour la plupart des organismes :

- les classes B sont trop grands
  et il y a saturation des classes B
- les classes C sont trop petits
  et il reste suffisamment de classes C

$\Rightarrow$ allocation de plusieurs classes C
$\Rightarrow$ explosion des tables de routage

CIDR

Solution : routage par agrégation de réseaux
Techniquement : une adresse est divisée en deux parties (numéro de réseau, numéro de machine)



$\Rightarrow$ évolution de la notion de classe :

- traditionnellement : frontière fixe

classe A	8 / 24
classe B	16 / 16
classe C	24 / 8

- CIDR : frontière variable $\Rightarrow$ notion de classe obsolète

CIDR

Impact sur les tables de routage :

$\Rightarrow$ information supplémentaire : la longueur du préfixe

Pour aller à	Passer par
193.51.24.0 / 21	193.48.55.34

```
193.51.24.0     = 0xC1331800
193.51.31.0     = 0xC1331F00
255.255.248.0   = 0xFFFFF800
```

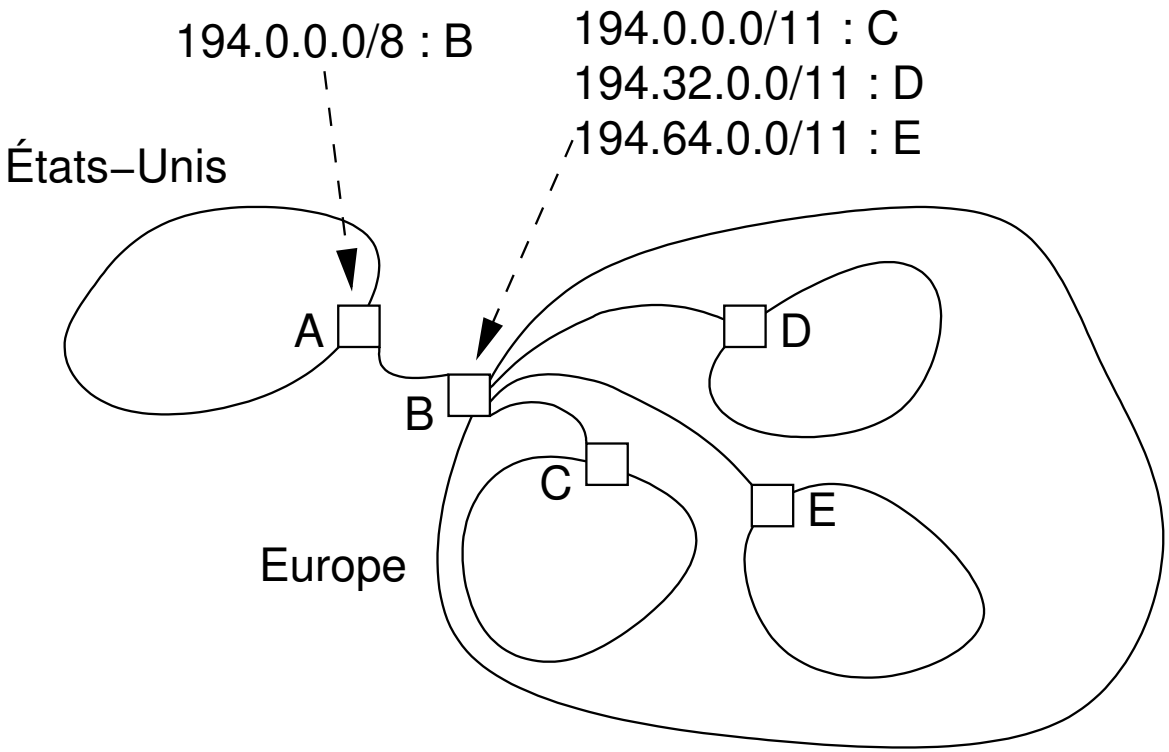
Algorithme :

```
si IP(dest) & table [n].masque = table[n].reseau
    alors router vers table [n].routeur
fin si
```

CIDR

CIDR permet le routage géographique et hiérarchique

Exemple : quelques tables de routage



Exemple théorique, jamais mis en pratique

CIDR

Subdivisions en espaces de routage :

Multi-regional	192.0.0.0	–	193.255.255.255
Europe	194.0.0.0	–	195.255.255.255
Others	196.0.0.0	–	197.255.255.255
North America	198.0.0.0	–	199.255.255.255
Central/South America	200.0.0.0	–	201.255.255.255
Pacific Rim	202.0.0.0	–	203.255.255.255
Others	204.0.0.0	–	205.255.255.255
Others	206.0.0.0	–	207.255.255.255

CIDR

- Aucun impact sur les réseaux de sites
- CIDR utilisable avec des longueurs de préfixe différentes à tous les niveaux
- Problème pour appliquer le routage géographique : quelle liaison choisir si plusieurs opérateurs IP pour la zone géographique ?
- Impact sur les protocoles de routage : propagation de la longueur de préfixe

Plan

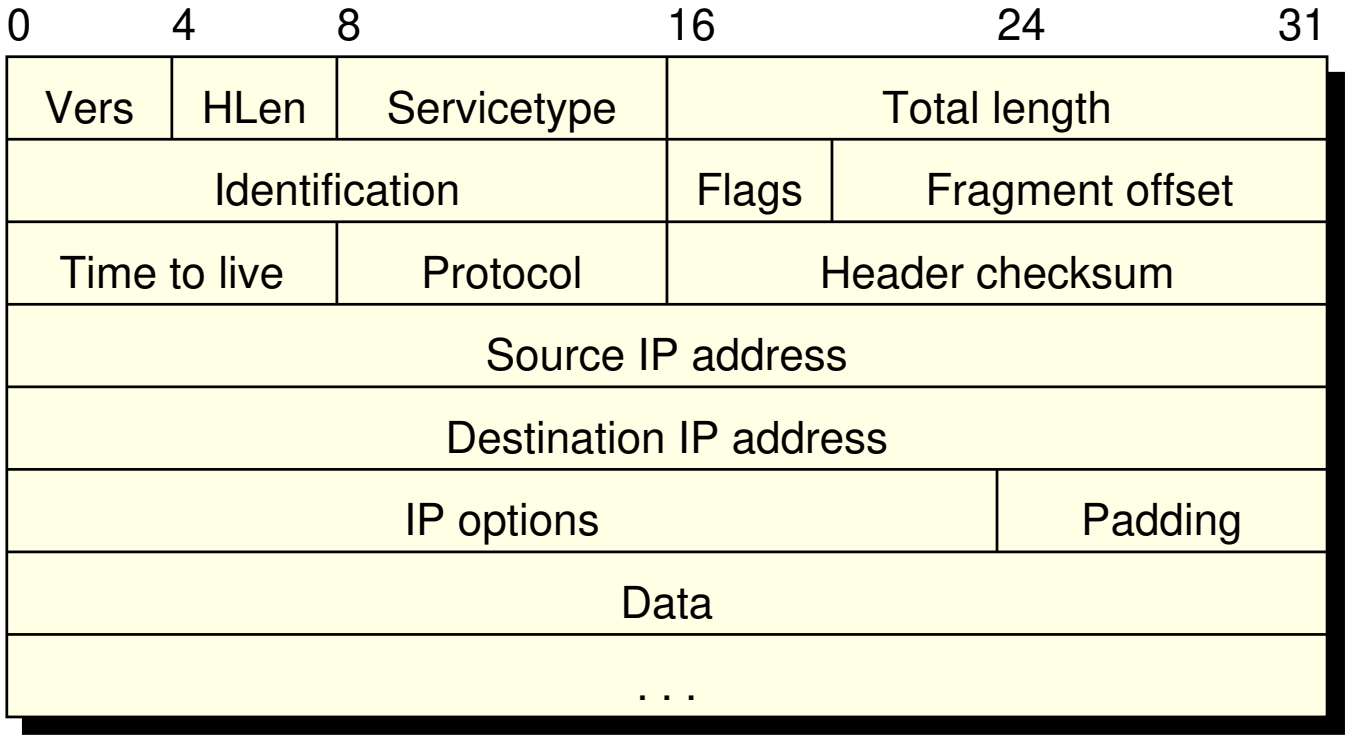
Datagrammes

Caractéristiques de IP :

- pas de connexion
  $\Rightarrow$ dialogue de machine à machine
- non fiable
  $\Rightarrow$ fiabilité assurée par les protocoles supérieurs

Le protocole IP est simple. Sa simplicité est la clef de sa robustesse

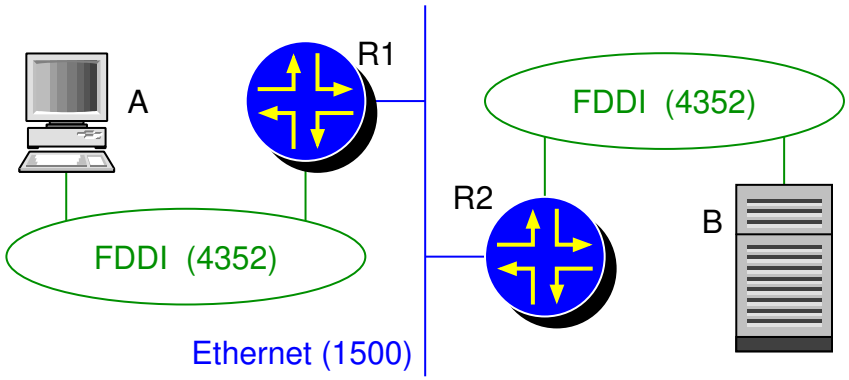
Datagrammes



Datagrammes – Fragmentation

- datagramme IP : limite = 65535 octets
- réseau sous-jacent : limite le plus souvent inférieure
  MTU = Maximum Transfert Unit

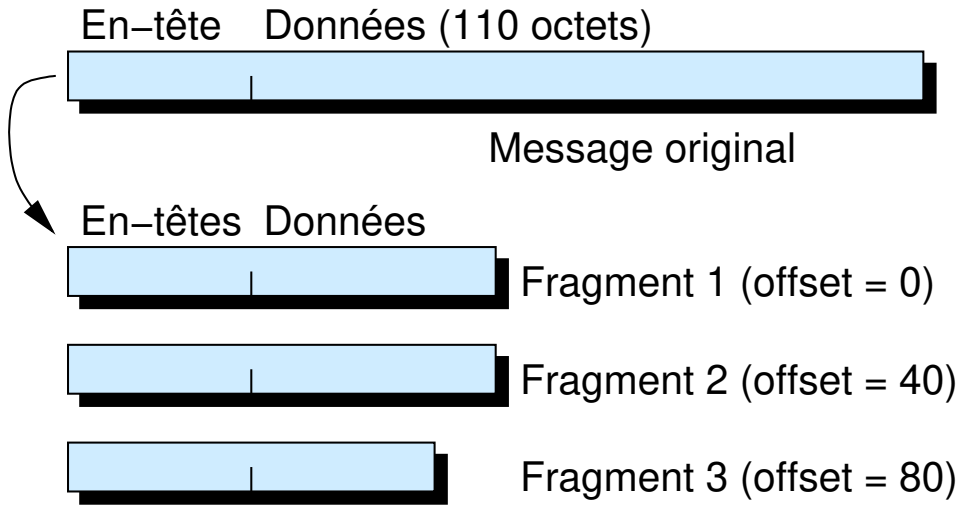
Passage par des réseaux de MTU inférieur $\Rightarrow$ fragmentation



Fragmentation : par un nœud intermédiaire

Assemblage des paquets : par la destination uniquement

Datagrammes – Fragmentation



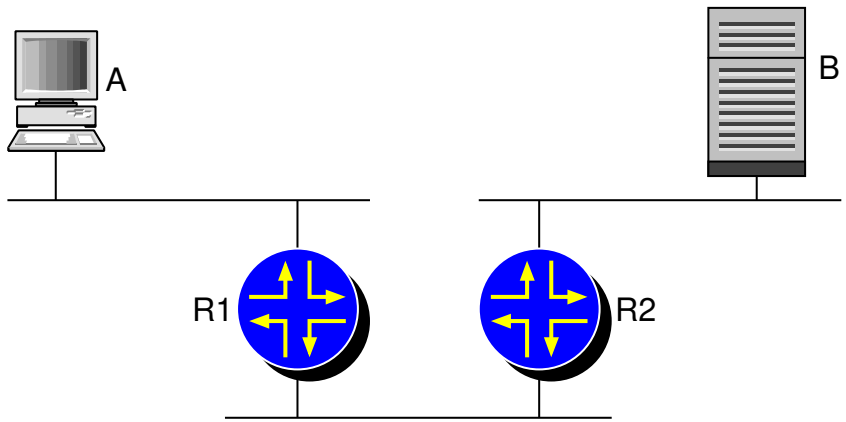
En-têtes des messages fragmentés identiques à l'en-tête du message original, sauf :

Num	Flag « More Fragment »	Offset
1	1	0
2	1	40
3	0	80

Fragmentation seulement si flag « Don't Fragment » = 0

Datagrammes – Time To Live

Les réseaux ne sont pas toujours exempts de problèmes... Exemple : boucle de routage



R_1 route B via R_2, R_2 route B via R_1 (erreur sur R_2)
$\Rightarrow$ boucle de routage
$\Rightarrow$ paquets circulent sans fin (*paquets fantômes*)

Solution : *Time To Live* (TTL). Compteur décrémenté à chaque routeur. Le paquet est détruit si TTL = 0 sans arriver à destination

IP – Network byte order

Problème : les machines ont des représentations physiques des entiers différentes (ex.: 80x86/Vax et MC68x00/IBM370)

Solution : *Network Byte Order*, octet le plus significatif en tête

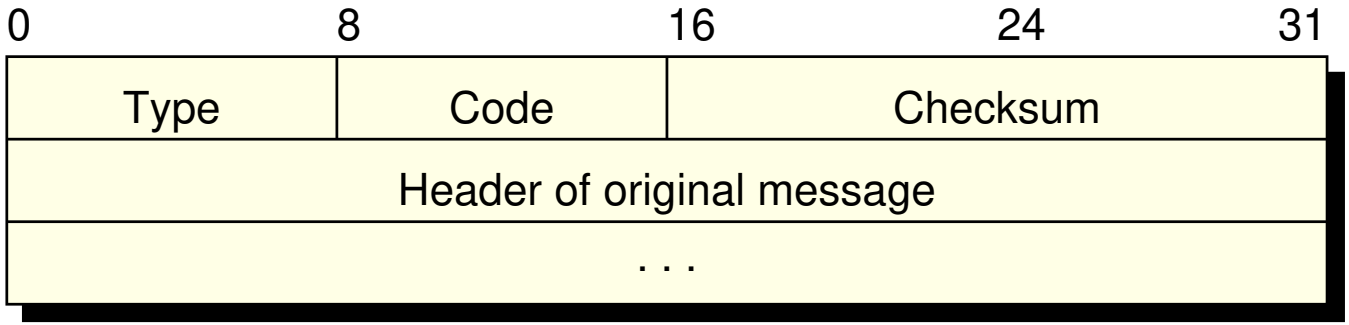
Plan

ICMP

ICMP = *Internet Control Message Protocol*

- protocole encapsulé dans des datagrammes IP
- générés par la destination ou un routeur intermédiaire
- indiquent une condition (d'erreur) à l'émetteur

ICMP



Type	Signification
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect
8	Echo Request
11	TTL Exceeded
12	Parameter Problem
13	Timestamp Request
14	Timestamp Reply
17	Address Mask Request
18	Address Mask Reply

ICMP – Echo Request, Echo Reply

Test de l'accessibilité de la machine distante :

- les tables de routage sont correctes
- tous les éléments intermédiaires fonctionnent
- IP sur la machine distante fonctionne
- ICMP sur la machine distante fonctionne

$\Rightarrow$ commande `ping`

ICMP – Echo Request, Echo Reply

Commande `ping`

La commande `ping` est un des outils de base de l'administrateur réseau :

```
> ping prep.ai.mit.edu.
PING prep.ai.mit.edu: 64 byte packets
64 bytes from 18.159.0.42: icmp_seq=0. time=187. ms
64 bytes from 18.159.0.42: icmp_seq=3. time=311. ms
64 bytes from 18.159.0.42: icmp_seq=4. time=233. ms
----prep.ai.mit.edu PING Statistics----
5 packets transmitted, 3 packets received, 40% packet loss
round-trip (ms)  min/avg/max = 187/243/311
```

Histoire de ping : `https://ftp.arl.army.mil/~mike/ping.html`

ICMP – Destination Unreachable

Envoyé lorsqu'un routeur ne peut router un datagramme (exemple : pas d'entrée dans la table de routage)

Variantes (selon le « code ») :

- réseau non accessible
- machine non accessible
- port non accessible
- fragmentation interdite par le flag « Don't Fragment »
- destination interdite par un pare-feu
- etc.

ICMP – Destination Unreachable

Extension : algorithme du *Path MTU discovery*
$\Rightarrow$ pour trouver le MTU minimum entre deux nœuds :

- l'émetteur envoie un datagramme D (taille m_1 = MTU du réseau local) avec le flag DF = 1
- si un routeur doit envoyer D sur un réseau de MTU m_2 inférieur à la taille de D, il renvoie un message ICMP *Destination Unreachable* avec le code 4 (*Too Big*) et la valeur de m_2
- l'émetteur recommence jusqu'à réception d'un message indiquant que D est arrivé à destination

Le chemin peut changer $\Rightarrow$ le MTU minimum aussi :

- diminution : l'émetteur reçoit *Destination Unreachable*
- augmentation : refaire l'algorithme périodiquement

ICMP - TTL Exceeded

Si TTL = 0, le message *TTL Exceeded* est envoyé à la source

Astuce : programme `traceroute` $\Rightarrow$ tracer le chemin suivi

Principe :

- l'émetteur envoie un datagramme avec TTL = 1
- le premier routeur renvoie un message ICMP signé
- l'émetteur affiche l'adresse du premier routeur
- l'émetteur recommence en incrémentant le TTL jusqu'à arriver à la destination

ICMP - TTL Exceeded

Commande `traceroute`

La commande `traceroute` affiche les routeurs vers une destination :

```
> traceroute concorde.inria.fr.
traceroute to concorde.inria.fr (192.93.2.39) from 0.0.0.0, 30 hops max, 20 byte packets
1  r-prism.reseau.uvsq.fr (198.51.100.254)  6 ms  3 ms  9 ms
2  r-uvsq.reseau.uvsq.fr (203.0.113.30)  3 ms  5 ms  3 ms
3  france-telecom.reseau.uvsq.fr (193.51.43.2)  4 ms  10 ms  3 ms
4  boulogne.rerif.ft.net (193.48.55.33)  6 ms  6 ms  6 ms
5  inria-rocquencourt.rerif.ft.net (193.48.55.58)  10 ms  8 ms  8 ms
6  rocq-gwr.inria.fr (192.93.122.2)  16 ms  9 ms  9 ms
7  concorde.inria.fr (192.93.2.39)  10 ms  9 ms  9 ms
```

Possibilité de choisir le type des paquets envoyés :

- `-I` : envoie des paquets ICMP echo
- `-T` : envoie des paquets TCP SYN
- `-U` : envoie des paquets UDP pour un port invraisemblable

$\Rightarrow$ permet de contourner le filtrage de certains routeurs