

# Cours de Réseaux

Karim Sehaba

Maître de conférences

Université Lumière Lyon 2

[karim.sehaba@univ-lyon2.fr](mailto:karim.sehaba@univ-lyon2.fr)

<http://liris.cnrs.fr/~ksehaba>

# Organisation

Supports de CM et TD ici : <http://liris.cnrs.fr/ksehaba/Reseaux>

## ■ CM

- 5 séances d'1h45
- Examen

### Plan

**Modèles en couches : OSI & TCP/IP**

Couche internet (IP, ICMP, ARP/RARP)

Couche transport (TCP, UDP)

Couche application (HTTP, DNS, SMTP, FTP...)

Programmation Réseaux

## ■ TD

- 7 séances d'1h45
- Projet

### Plan

Modèles en couches (OSI et TCP)

Installation et configuration d'un serveur Web – Apache

L'adressage et le routage IP

Programmation par sockets – MiniTchat (3 séances)

Évaluation

# Références

---

## ■ Livres :

- *Réseaux* de Andrew Tanenbaum
- *Les Réseaux (edition2005)* de Guy Pujolle
- *TCP/IP : Architecture protocoles et applications* de Douglas Comer
- *TCP/IP pour les nuls* de Candance Leiden et Marshall Wilensky

## ■ Net :

- <http://www.commentcamarche.net/>
- Certains transparents sont de Olivier Glück (Lyon1)

# Introduction

---

## Qu'est-ce qu'un réseau ?

Un ensemble d'entités (objets, personnes, machines, etc.) interconnectées les unes avec les autres

### *Exemples :*

- réseau de transport
- réseau téléphonique
- réseau de neurones
- ...

## Réseaux informatique

Un ensemble d'ordinateurs reliés entre eux grâce à des lignes physiques et échangeant des informations sous forme de données numériques

## Intérêts des réseaux informatique

- communication (personnes, processus...)
- partage de ressources (fichiers, applications, matériels...)
- minimisation des coûts
- ...

# Introduction

---

## Les topologies

### En bus

Tous les ordinateurs sont reliés à une même ligne de transmission (*bus*)

### En étoile

Tous les ordinateurs sont reliés à un *concentrateur* (*hub*)

### En anneau

Les ordinateurs sont situés sur une boucle

### Câblage en maille

Chaque machine est reliée à toutes les autres par un câble

**Quels sont les avantages et inconvénients de chaque topologie ?**

# Introduction

---

## Les types de réseaux

### **LAN (local area network - Réseau Local)**

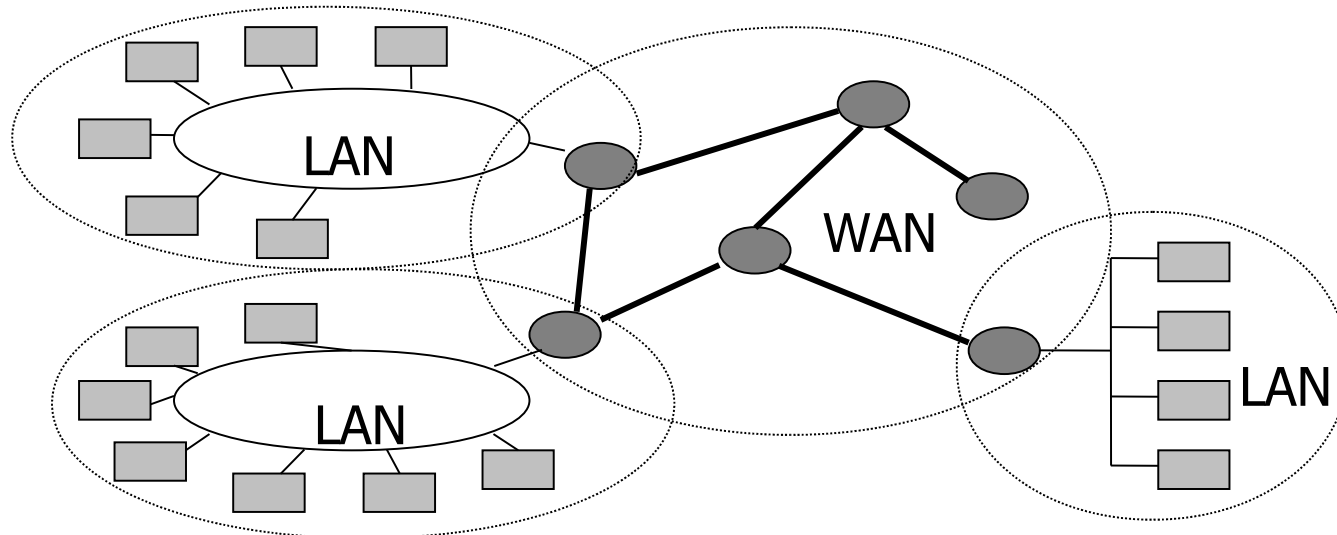
même organisation dans une petite aire géographique,  
débit : [10 Mbps, 1 Gbps], utilisateurs : [100 à 1000]

### **MAN (Metropolitan Area Network – Réseau métropolitain)**

interconnexion de plusieurs LAN géographiquement proches (dizaines de km)

### **WAN (Wide Area Network – Réseau étendu)**

interconnexion de plusieurs LAN à travers de grandes distances géographiques



# Introduction

---

## Les techniques de transfert

### La commutation de circuits

- Circuit entre les deux entités qui communiquent
- Le circuit reste ouvert jusqu'au moment où l'un des deux participants interrompt la communication

### Le transfert de paquets

#### Routage

- Le paquet qui arrive doit posséder l'adresse complète du destinataire
- Le routeur consulte sa table de routage pour choisir la meilleure ligne de sortie

#### Commutation

- Les commutateurs acheminent les paquets vers le récepteur en utilisant des références, de circuit
- Les tables de commutation sont des tableaux, qui, à une référence, font correspondre une ligne de sortie
- Seules les communications actives entre utilisateurs comportent une entrée dans la table de commutation.

# **Modèles en couches**

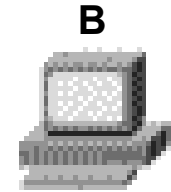
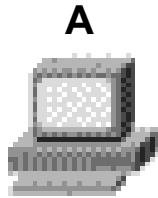
Notion de protocole

Modèle de référence : OSI de ISO

Modèle utilisé sur Internet : TCP/IP

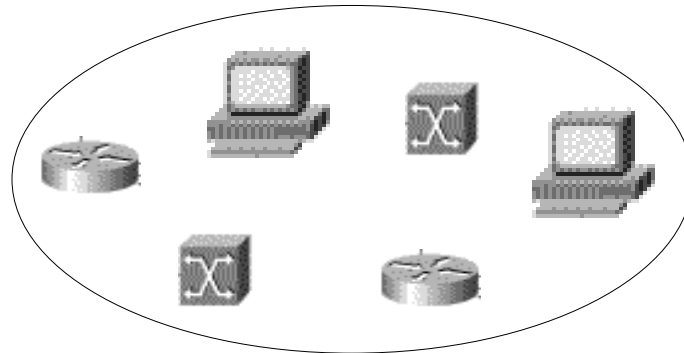
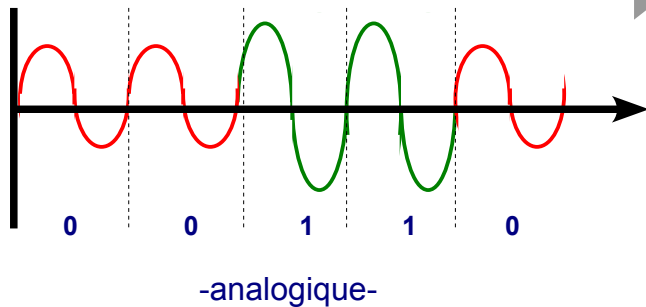
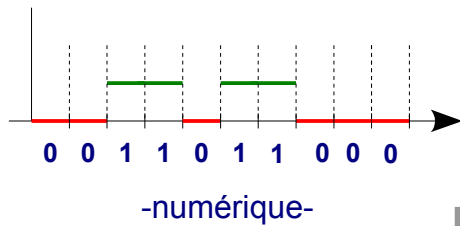


# Problématiques des réseaux



- 1. Données      Xyab lkd...
- 2. Code binaire    00110110001...
- 3. Signal

- 3. Données      Xyab lkd...
- 2. Code binaire    00110110001...
- 1. Signal



# Protocole

---

- Un ensemble de convention préétablies pour réaliser un échange de données entre deux entités
- Il définit le format des données et les règles d'échanges
  - syntaxes et sémantique de message...
- En particulier :
  - format des données et les règles d'échange
  - délimitation des blocs de données échangés
  - organisation et contrôle de l'échange
  - contrôle de la liaison

Communication en réseau :  
**protocoles** organisés en plusieurs **couches**

# Modèle de référence OSI

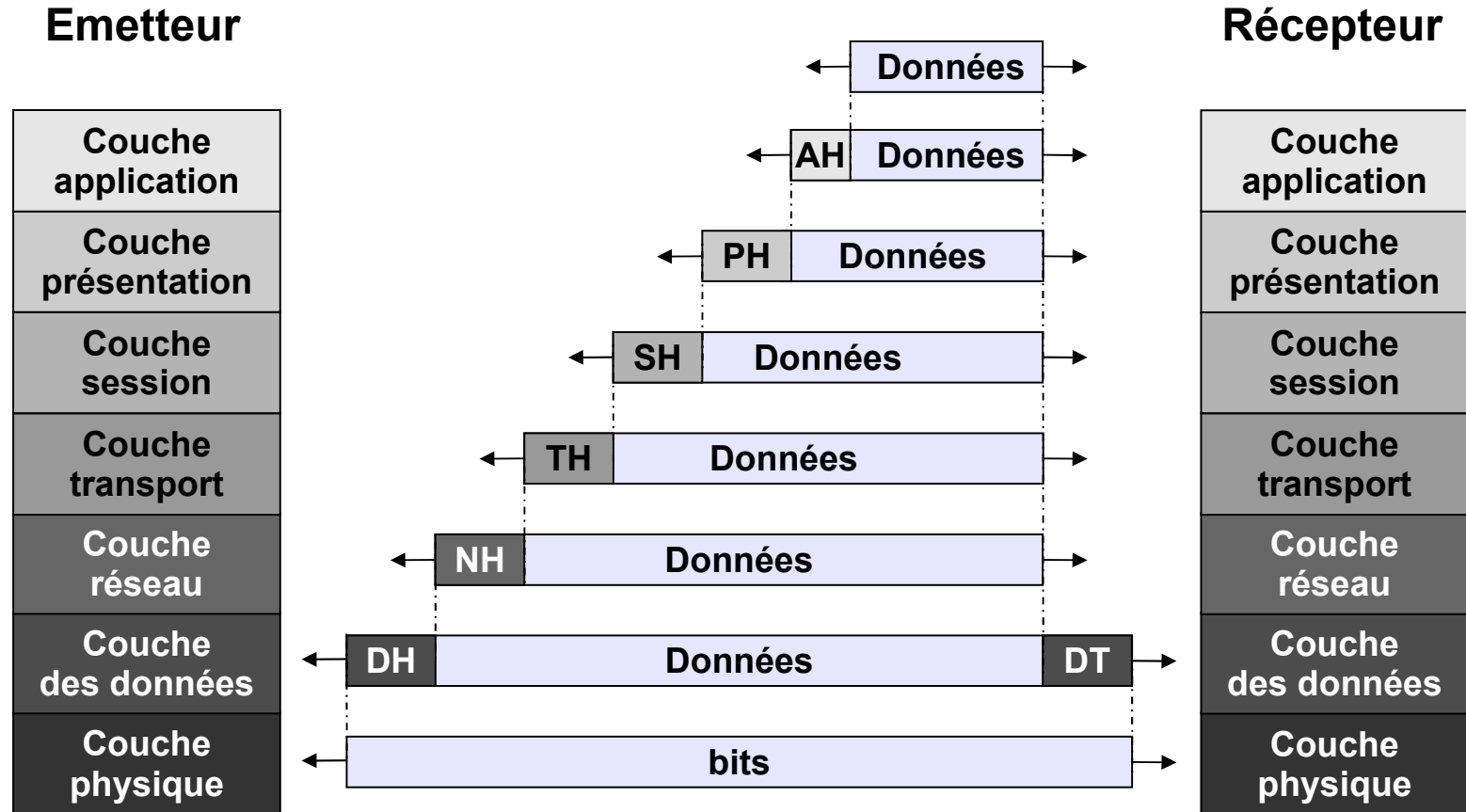
---

- *Open Systems Interconnection* (Interconnexion des Systèmes Ouverts)
- Définit par "*International Standard Organisation*"
  - organisation non gouvernementale
  - centaine de pays membres
  - édite des normes dans tous les domaines

## **Idée fondamentale :**

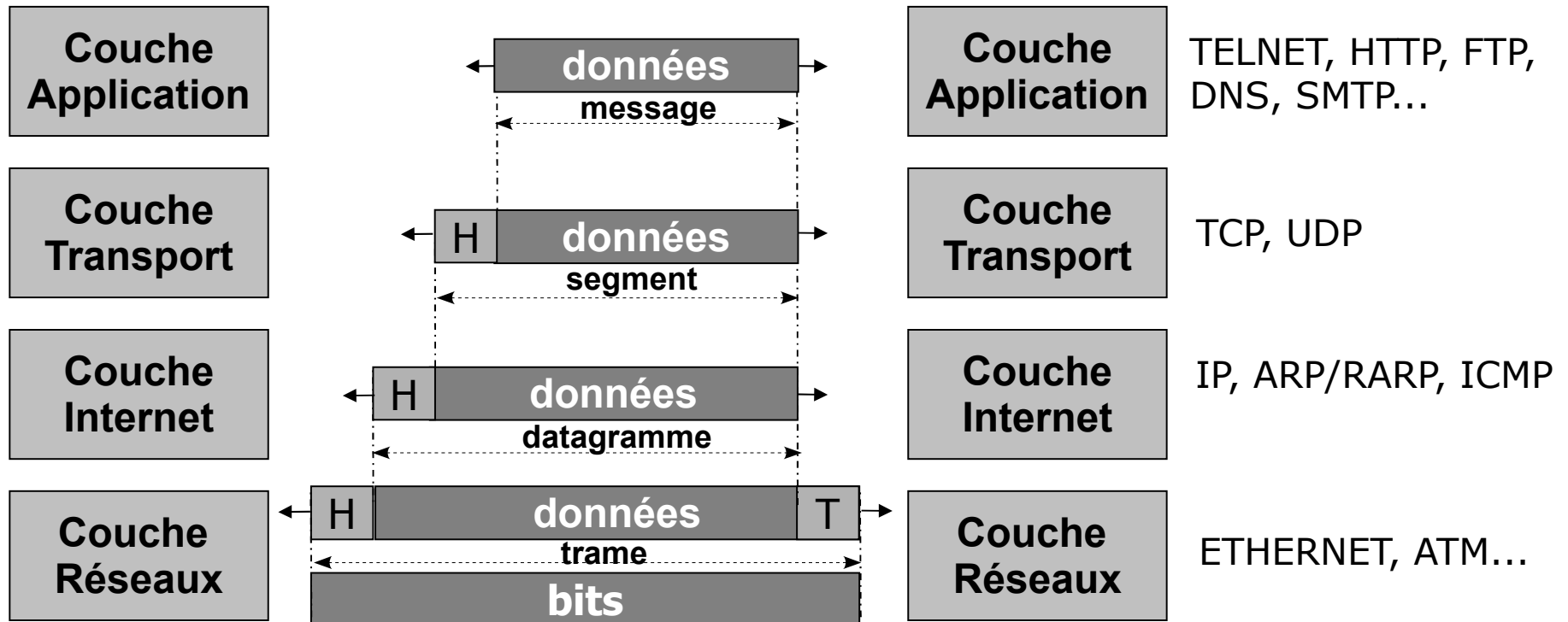
- Modèle en couches
- « une couche » : un ensemble homogène destiné à accomplir une tâche ou rendre un service
- Le découpage en couche permet de
  - dissocier des problèmes de natures différentes
  - rendre l'architecture évolutive
  - faire de la réutilisation

# Modèle de référence OSI



# Modèle TCP/IP

## Protocoles



# Modèle TCP/IP

---

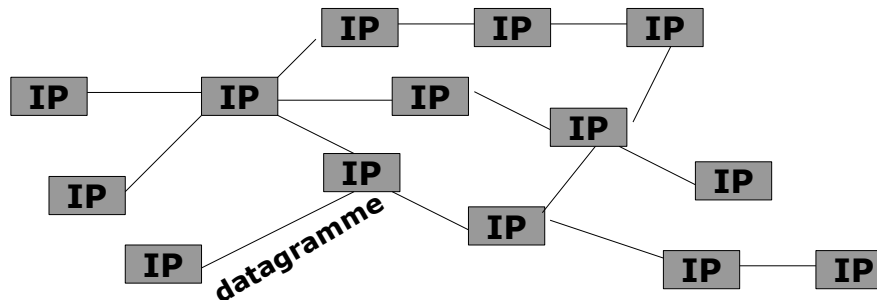
## Les couches TCP/IP

### ■ Réseau

- Acheminement des données sur la liaison
- Coordination de la transmission de données (synchronisation)
- Conversion des signaux (analogique/numérique)

### ■ Internet : communication entre machine

- Adressage IP
- Acheminement de datagrammes
- Peu de fonctionnalité, pas de garanties
- Gestion de la fragmentation et assemblage

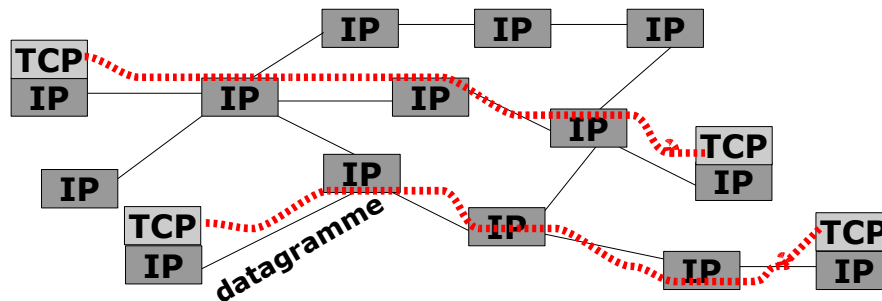


# Modèle TCP/IP

## Les couches TCP/IP

### ■ Transport : communication entre applications

- Protocole de transport de bout en bout
- Présent uniquement en extrémités
- Transport fiable de segments (en mode connecté)
- Protocole complexe (transmission, gestion des erreurs, séquençement...)



### ■ Application

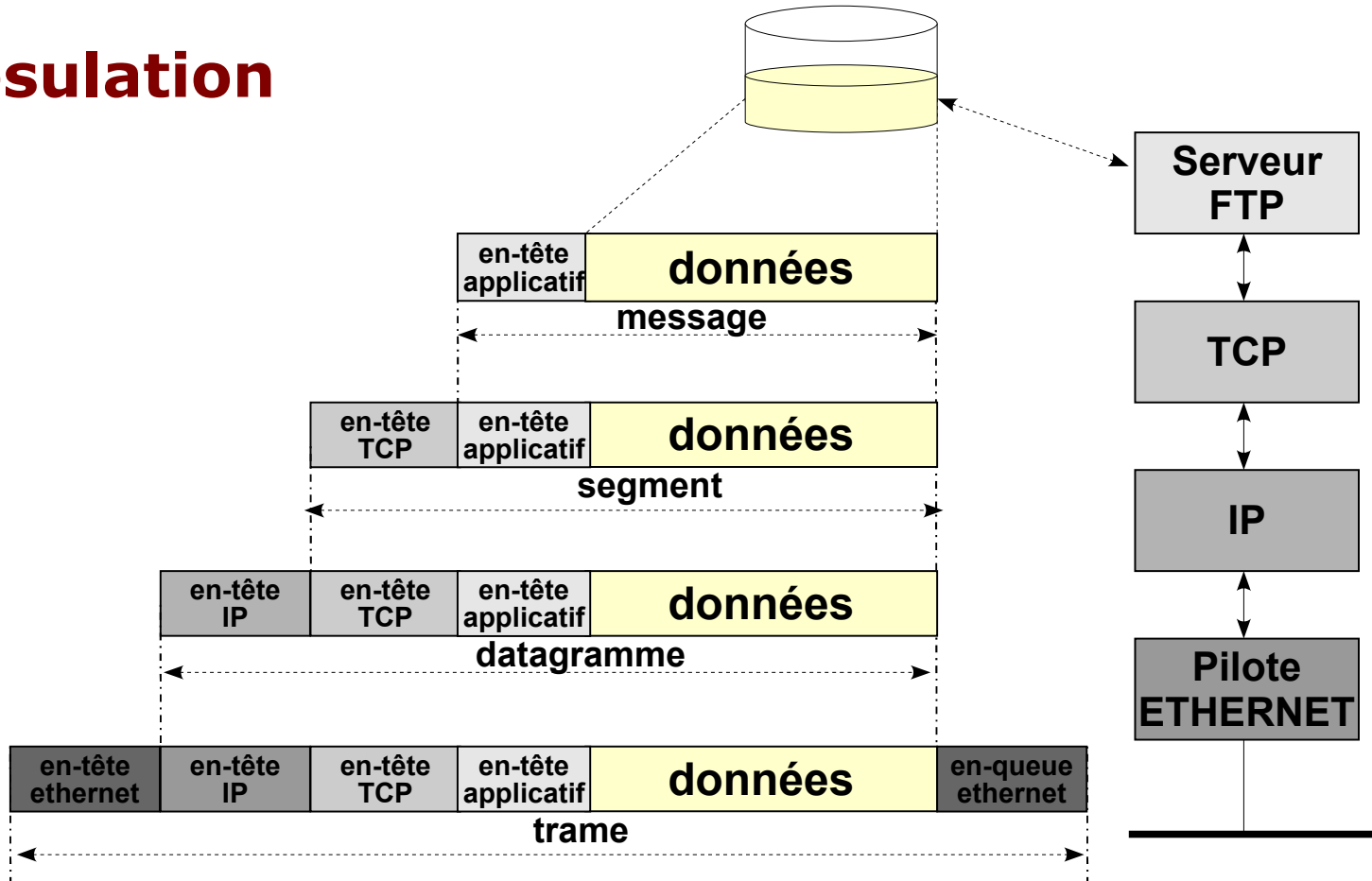
- services de gestion (transfert) de fichier et d'impression
- services de connexion au réseau
- services de connexion à distance
- utilitaires Internet divers

# Modèle TCP/IP

## Efficacité du transfert :

Efficacité du transfert = données utiles / données totales

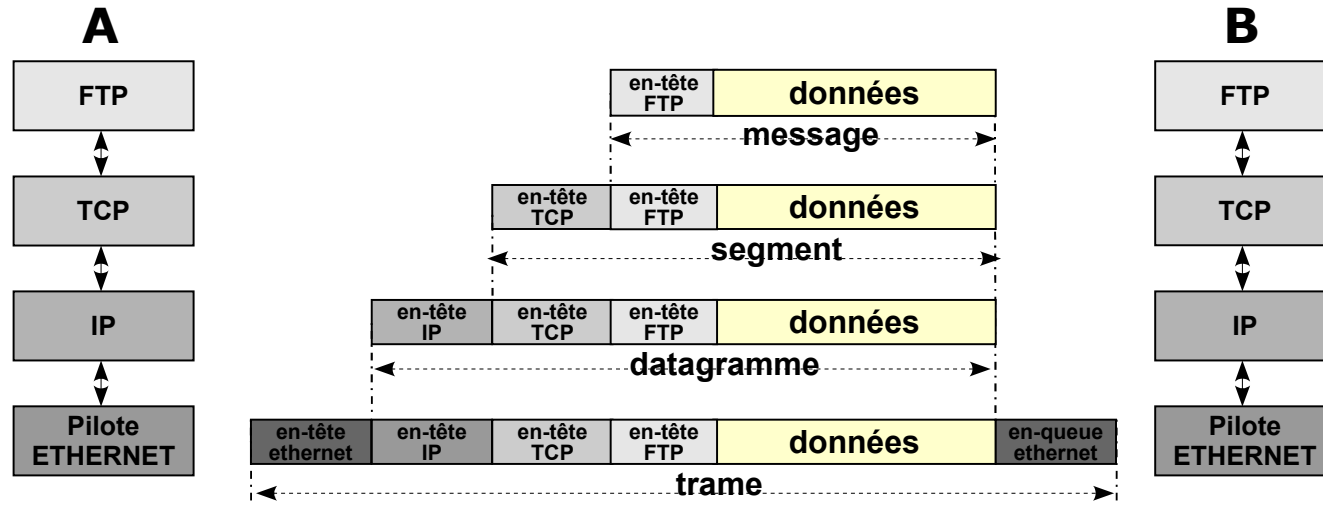
## Encapsulation





# Modèle TCP/IP

## Efficacité du transfert :



**A** veut envoyer **1024 octets de données** à **B** en utilisant le protocole FTP. On suppose que :

- l'entête **FTP** a une taille de **70 octet**
- l'entête **TCP** a une taille fixe de **20 octets**
- l'entête **IP** a une taille fixe de **20 octets**
- l'entête plus l'en-queue des trames **ETHERNET** est de **18 octets**
- Taille max d'une trame Ethernet est de **1518 octets**

Quelle est l'efficacité du transfert ?

# **Délimitation des données**

Notion de fanion

Notion de transparence

# Notion de fanion

---

- Lors d'une transmission de données, il faut pouvoir reprer le début et la fin de la séquence des données transmises
- Fanion en transmission synchrone
  - Un caractère spécial
  - Une séquence de bits particuliers



**Exemple de fanion** : 01111110

- Fonctions du fanion
  - délimite les données
  - maintien de la synchronisation de l'horloge de réception

Quelles sont les problèmes que pose le fanion ?

# Notion de transparence

---

- Les caractères « spéciaux » comme le fanion ne sont pas délivrés aux couches supérieures, ils sont interprétés pour les besoins du protocole
- Les caractères « spéciaux » doivent pouvoir être transmis en tant que données et donc délivrés en tant que tel
  - mécanisme de transparence
  - définition d'un autre caractère spéciale ; le **caractère d'échappement**
- Fonctionnement
  - **côté émission** : Insertion du caractère d'échappement devant le caractère à protéger
  - **côté réception** : L'automate examine chaque caractère pour découvrir le fanion de fin ; s'il rencontre le caractère d'échappement, il l'élimine et n'interprète pas le caractère suivant -> il le délivre au système

# La technique du bit de bourrage

---

- Seul le fanion (01111110) peut contenir plus de 5 bits consécutifs à "1"
- **Côté émission** : si 5 bits consécutifs sont à "1", l'automate insère un "0"
- **Côté réception** : si 5 bits consécutifs sont à "1", l'automate regarde le bit suivant :
  - s'il est à "1", il s'agit d'un fanion
  - s'il est à "0", le "0" est enlevé de la séquence

## ■ Exemple

Fanion	Séquence originale	Fanion
01111110	000111111001011111001101100	01111110

Fanion	Séquence transmise	Fanion
01111110	0001111101001011111001101100	01111110

# Organisation

## ■ CM

### Plan

Modèles en couches : OSI & TCP/IP

**Couche internet (IP, ICMP, ARP/RARP)**

Couche transport (TCP, UDP)

Couche application (HTTP, DNS, SMTP, FTP...)

Programmation Réseaux

# Couche Internet (1)

## Protocole IP

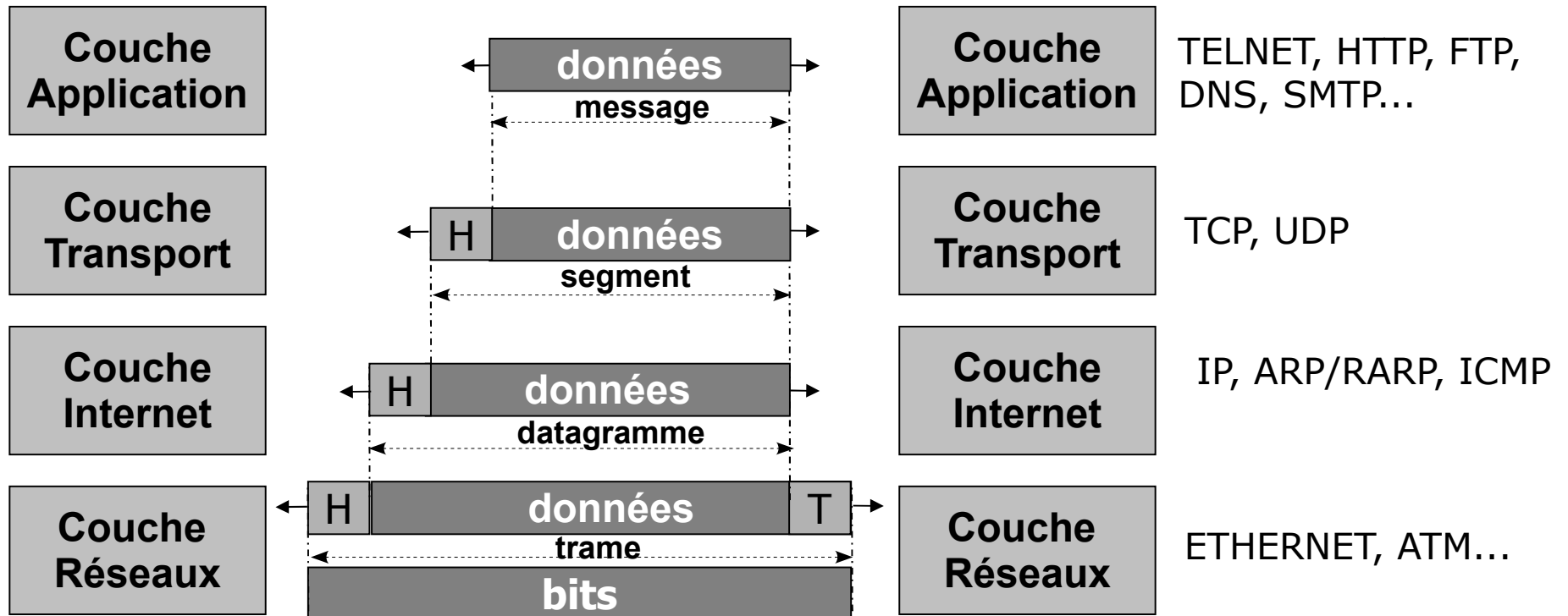
Gestion de la fragmentation

ARP, ICMP

IPv6

# Modèle TCP/IP

## Protocoles





# Couche internet

---

## ■ **Functionalité : communication entre machines**

- Adressage IP
- Acheminement de datagrammes (en mode **non connecté**)
- Peu de fonctionnalité, pas de garanties
- Gestion de la fragmentation et assemblage

## ■ **Protocoles de la couche**

- IP - Internet Protocol
- ARP - Address Resolution Protocol -
- ICMP - Internet Control and error Message Protocol -
- ...

# Adresse IP

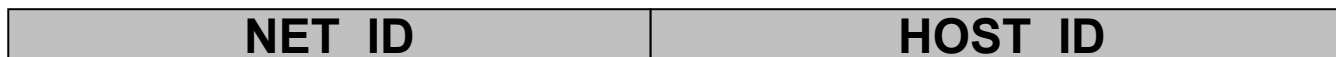
---

- Permet d'identifier les machines sur le réseau
- Distribuées par ICANN -Internet Corporation for Assigned Names and Numbers
  - Adresse IP = 32 bits (4 octets)



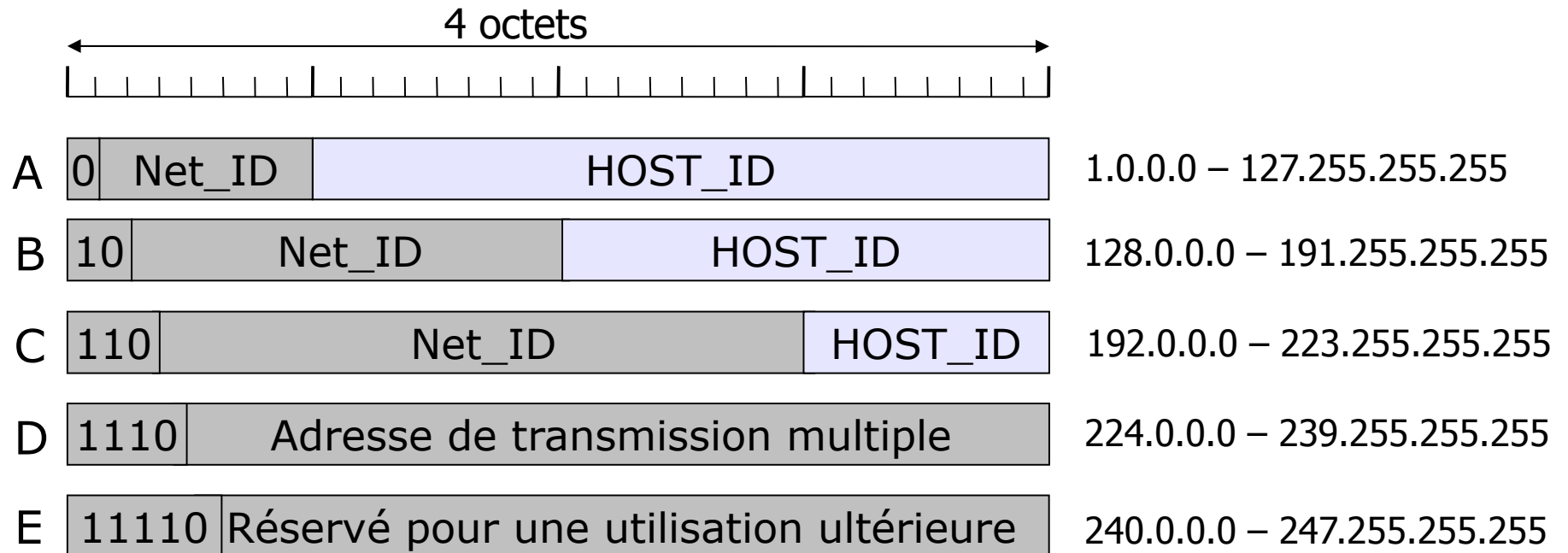
Représentation décimale : W.X.Y.Z

- Chaque adresse est composée de deux champs :
  - **NET\_ID** : identifiant du réseau IP (utilisé pour le routage)
  - **HOST\_ID** : identifiant de la machine dans le réseau IP



# Adresse IP

## Classes réseaux



Quel est le nombre d'ordinateurs supporté pour chaque classe ?

# Adresse IP spéciales

---

## Diffusion locale et distante

- 255.255.255.255 : adresse de broadcast sur le réseau IP local
- <NET\_ID><111...111> : adresse de broadcast dirigée vers le réseau de numéro NET\_ID

## Rebouclage local : 127.y.x.z

- Généralement 127.0.0.1 (*localhost*)
- Permet de tester la pile TCP/IP locale sans passer par une interface matérielle

## Adresse 0.0.0.0

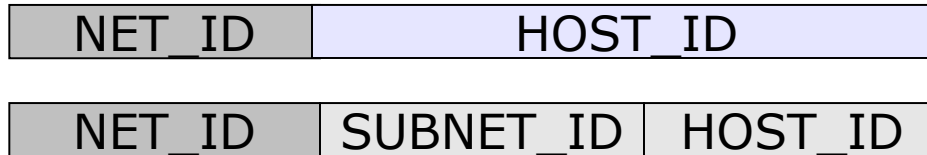
- Utilisée par le protocole RARP
- Adresse de la route par défaut dans les routeurs

# Sous-réseaux

---

Comment diviser un réseau IP en plusieurs sous-réseaux ?

Prendre quelques bits de la partie <HOST\_ID> de l'@IP pour le sous-réseau



## Utilisation des masques *Netmask*

L'acheminement se fait en fonction de <NET\_ID> et <SUBNET\_ID>

Mais la taille de <SUBNET\_ID> est inconnue !

Information donnée par le **netmask** ; tous les bits à 1 correspond à <NET\_ID><SUBNET\_ID>

# Sous-réseaux

Comment déterminer l'adresse de sous-réseau ?

Adresse source

NET_1	SUBNET_1	HOST_ID
-------	----------	---------

Netmask **ET Logique**

1.....1	1.....1	0.....0
---------	---------	---------



NET_1	SUBNET_1	0.....0
-------	----------	---------

?  
=

Adresse destination

NET_2	SUBNET_2	HOST_2
-------	----------	--------

Netmask **ET Logique**

1.....1	1.....1	0.....0
---------	---------	---------



NET_2	SUBNET_2	0.....0
-------	----------	---------

- Le netmask permet de savoir si la machine source et destination sont sur le même sous-réseau
- La classe d'adressage permet de savoir si elles sont sur le même réseau

# Sous-réseaux

---

Un réseau de classe B dispose du masque de sous-réseau 255.255.240.0

Les machines 159.84.146.236, 159.87.178.23 et 159.84.158.23 trouvent-elles sur le même sous-réseaux ?

**Netmask 255.255.240.0 =**

11111111.11111111.11110000.00000000

**IP1 :** 159.84.146.236 = 10011111.01010100.10010010.11101100

**IP2 :** 159.87.178.23 = 10011111.01010111.10110010.00010111

**IP3 :** 159.84.158.23 = 10011111.01010100.10011110.00010111

# Couche Internet (1)

Protocole IP

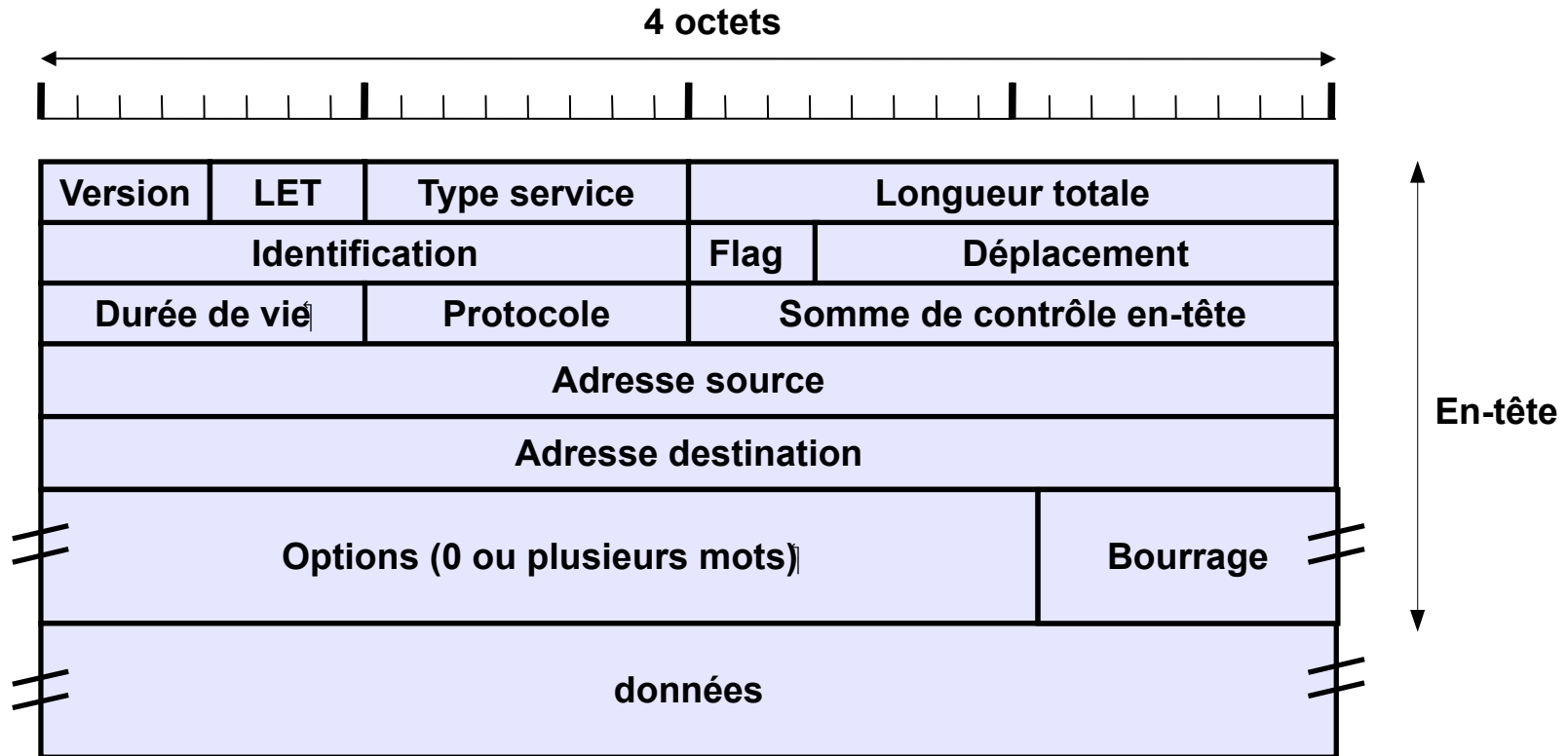
**Gestion de la fragmentation**

ARP, ICMP

IPv6



# Format d'un datagramme IP



**Version (4 bits):** la version d'IP utilisée

**LET (4 bits) :** Longueur d'En-Tête en nombre de mots de 32 bits

**Type de service (8 bits) :** qualité de service

**Longueur Totale (16 bits) :** taille (entête + données) en octet

**Identification (16bits) :** id des fragments d'un même paquet

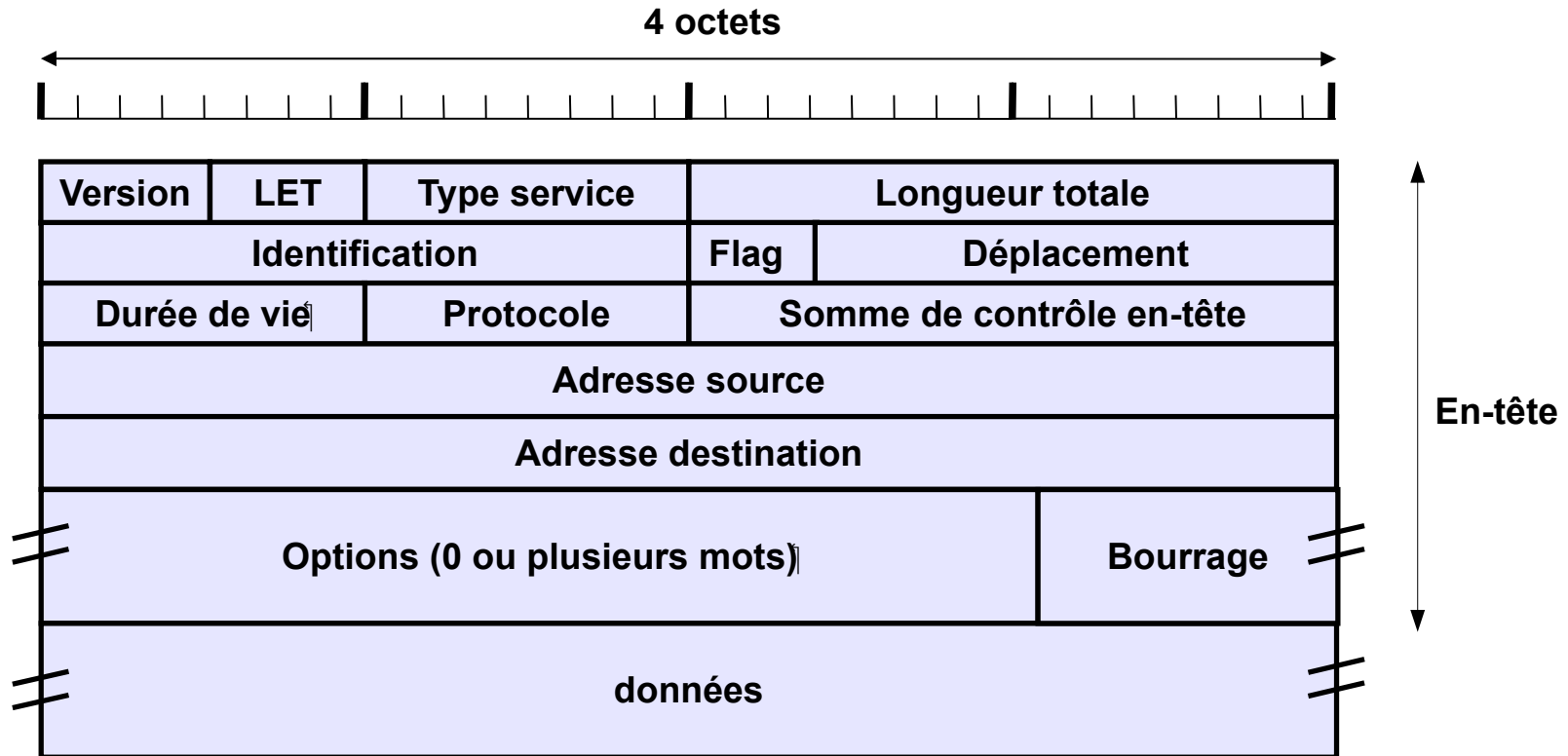
**Flags (3 bits):**

Bit 0: réservé, doit être à zéro ;

Bit 1: (AF) 0 = Fragmentation possible, 1 = Non fractionnable

Bit 2: (DF) 0 = Dernier fragment, 1 = Fragment intermédiaire

# Format d'un datagramme IP



**Déplacement (13 bits)** : Position du fragment par rapport au paquet de départ, en nombre de mots de 8 octets

**Durée de vie (TTL) 8 bits** : pour que le paquet peut rester dans le réseau

**Protocole (8 bits)** : type du protocole de niveau supérieur

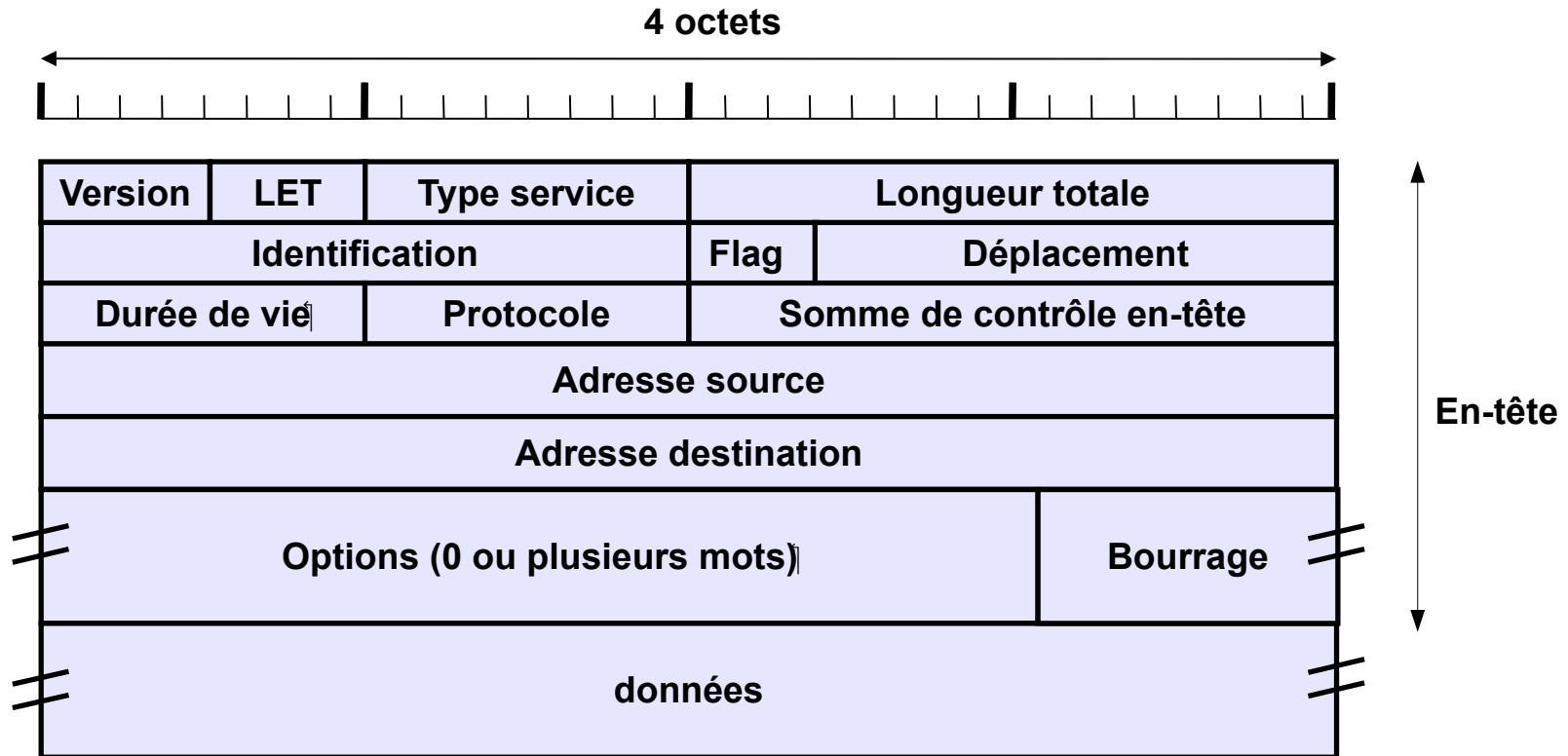
- 1 -> ICMP
- 17 -> UDP
- 6 -> TCP

**Checksum d'en-tête (16 bits)** : code de contrôle d'erreur pour l'entête

**Adresse source (32 bits)** : IP de la machine source

**Adresse destination (32 bits)** : IP de la machine destination

# Format d'un datagramme IP



**Quel est la taille maximale d'un datagramme ?**

# Fragmentation des datagrammes IP

---

Taille maxi d'un datagramme :  $2^{16} - 1 = \mathbf{65535}$  octet

Taille maximale d'une trame - **MTU** (Maximum Transfer Unit)

Type de réseau	MTU(oct)
Arpanet	<b>1000</b>
Ethernet	<b>1500</b>
FDDI	<b>4470</b>

**Problème** : pas de capacité pour envoyer de si gros paquets

**Solution** : découper en fragements les trames ==> **la fragmentation**

# Fragmentation des datagrammes IP

---

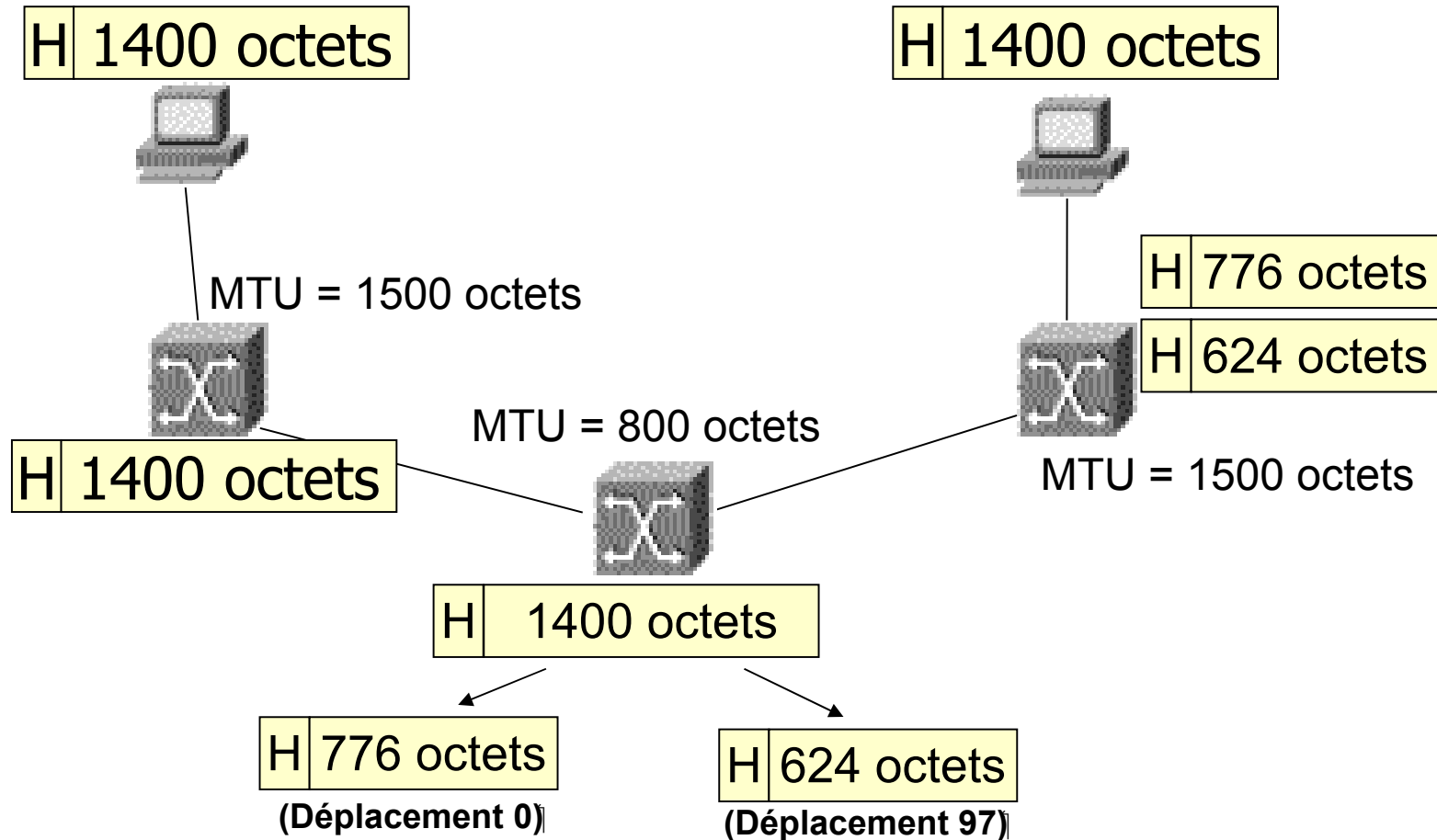
- Se fait au niveau des routeurs

## ■ Fonctionnement

- Découper en fragments de tailles inférieures au MTU du réseau et de telle façon que la **taille du fragment soit un multiple de 8 octets**
- Ajouter des informations afin que la machine de destination puisse réassembler les fragments dans le bon ordre
- Envoyer ces fragments de manière indépendante et les réencapsuler de telle façon à tenir compte de la nouvelle taille du fragment.


# Fragmentation des datagrammes IP

## Exemple





# Fragmentation des datagrammes IP

Version	LET	Type service	Longueur totale = 1420	
ID = 77		0 0	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (1400 oct)				

 MTU = 1500 octets

Version	LET	Type service	Longueur totale = 1420	
ID = 77		0 0	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (1400 oct)				

 MTU = 1500 octets


 MTU = 800 octets

Version	LET	Type service	Longueur totale = 796	
ID = 77		0 1	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (776 oct)				


Version	LET	Type service	Longueur totale = 644	
ID = 77		0 0	Déplacement = 97	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (624 oct)				

# Fragmentation des datagrammes IP

Version	LET	Type service	Longueur totale = 1420	
ID = 77		0 0	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (1400 oct)				


 MTU = 1500 octets

Version	LET	Type service	Longueur totale = 1420	
ID = 77		0 0	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (1400 oct)				

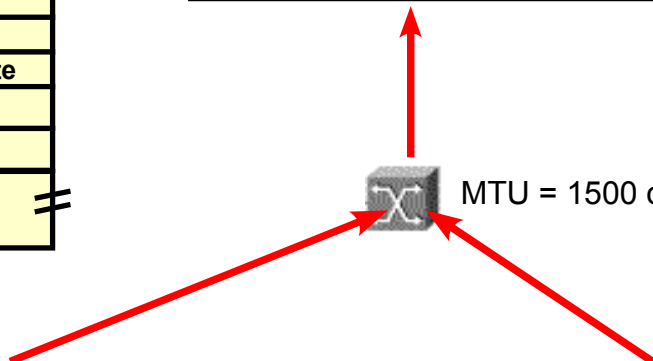
 MTU = 800 octets

Version	LET	Type service	Longueur totale = 796	
ID = 77		0 1	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (776 oct)				

Version	LET	Type service	Longueur totale = 1420	
ID = 77		0 0	Déplacement = 0	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (1400 oct)				

 MTU = 1500 octets

Version	LET	Type service	Longueur totale = 644	
ID = 77		0 0	Déplacement = 97	
Durée de vie	Protocole	Somme de contrôle en-tête		
Adresse source				
Adresse destination				
Données (624 oct)				





# Couche Internet (1)

Protocole IP

Gestion de la fragmentation

**ARP, ICMP**

IPv6

# Protocole ARP

---

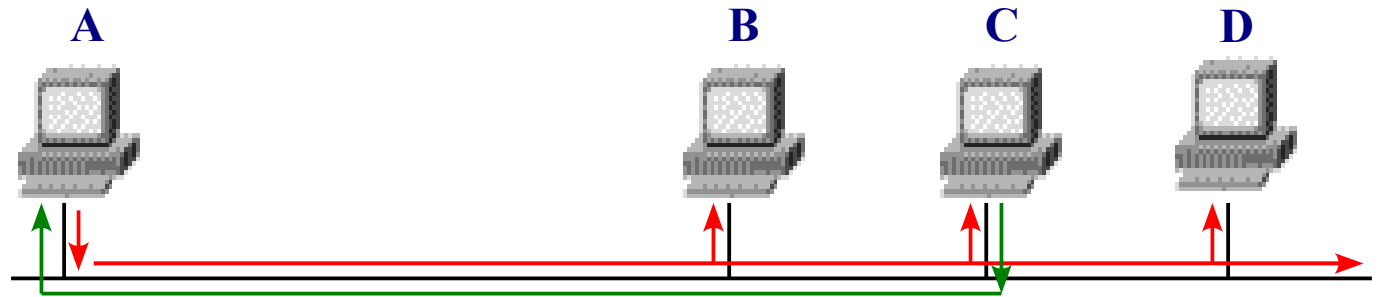
- ARP (*Address Resolution Protocol*) – Protocol de résolution d'adresse
- **@MAC (adresse physique)**: 48 bits  
fixée par le fabricant  
exemple :
- **@IP (adresse logique)**: 32 bits  
fixée par l'administrateur réseau ou ICANN  
exemple :
- **Rôle du protocole ARP**  
Faire la correspondance entre une @IP et une @MAC.

Les applications ne manipulent que des @IP (**pourquoi pas des @MAC?**)  
Le système doit retrouver l'@ MAC correspondante.  
Les systèmes construisent une table de correspondance (cache ARP).

# Protocole ARP

Table de correspondance	
@IP C	@MAC C

-Cache ARP-



**A** veut envoyer un message à **C** (**A** et **C** sur le même réseau)

## Principe du protocole

- **A** consulte sa table de correspondance IP->MAC
- **A** émet une requête ARP (contenant l'@IP de **C**) en *broadcast*
- **B**, **C** et **D** comparent cette adresse logique à la leur
- **C** répond en envoyant son adresse MAC
- **A** met à jour son cache ARP
- **A** envoie le message

# Protocole ICMP

---

- ICMP - *Internet Control and error Message Protocol*
  - Encapsulé dans un datagramme IP (champ protocole = 1)
  - Sert à contrôler le bon déroulement du protocole IP
  - Utilisé par l'utilitaire ping, traceroute...
- 
- **Utilitaire ping**
    - Teste l'accessibilité d'une destination de bout en bout
    - Évaluation de performances (mesure de temps aller-retour)
    - La réponse doit parvenir avant 20 secondes

## Exemple

Ping 127.0.0.1 : tester la pile TCP/IP locale

Ping mon@IP : vérifier la configuration réseau local

Ping @default-routeur : tester la configuration du sous-réseau et la passerelle

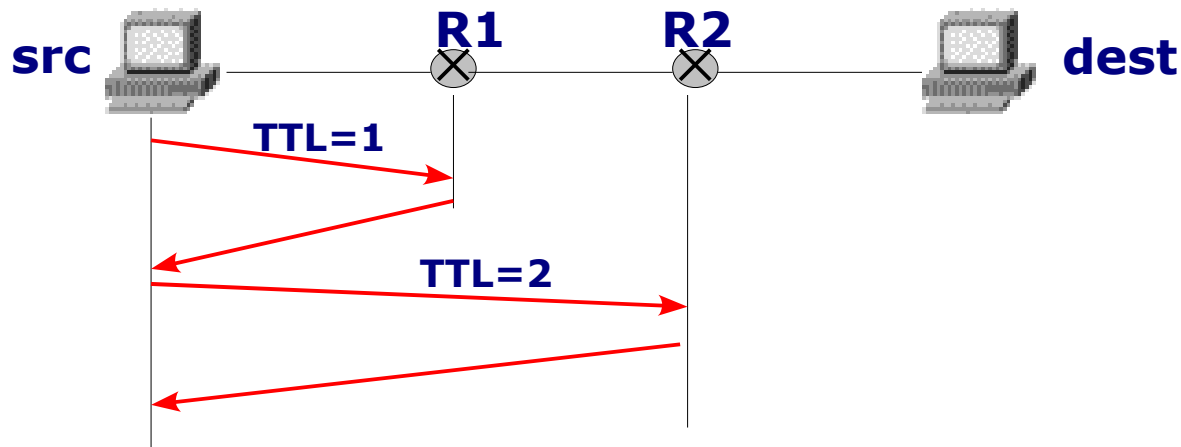
Ping @dest : tester un chemin de bout en bout

# Protocole ICMP

---

## Utilitaire Traceroute/Tracert

- Permet de trouver pas à pas le chemin pour atteindre une destination
  - Envoie d'un paquet IP avec TTL = 1
  - Attend ICMP délai expiré
  - Envoi d'un paquet IP avec TTL = 2
  - ...



# Couche Internet (1)

Protocole IP

Gestion de la fragmentation

ARP, ICMP

**IPv6**

# Protocole IPv6

---

## Pourquoi IPv6

- La fin d'IPv4 est proche
  - Pénurie d'adresses IP et explosion des tables de routage
  - Besoin d'un nouveau protocole mais suppose de le déployer sur tous les noeuds de l'Internet actuel !
- L'IETF, en 1990, élabore les souhaits d'un nouveau protocole et fit un appel à propositions
- 1993 : IPv6 est née de propositions combinées (Deering et Francis)

## Objectifs du protocole

- Supporter des milliards d'hôtes
- Réduire la taille des tables de routage
- Simplifier encore le protocole pour un routage plus rapide des paquets
- Offrir une meilleure sécurité
- Permettre au nouveau protocole et à l'ancien de coexister pendant quelques années

# Protocole IPv6

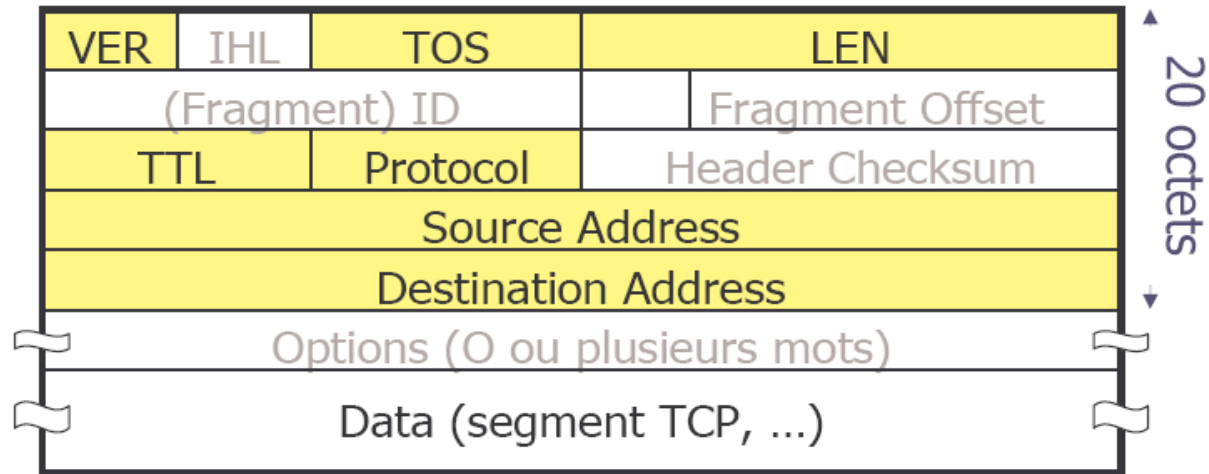
---

## Caractéristiques

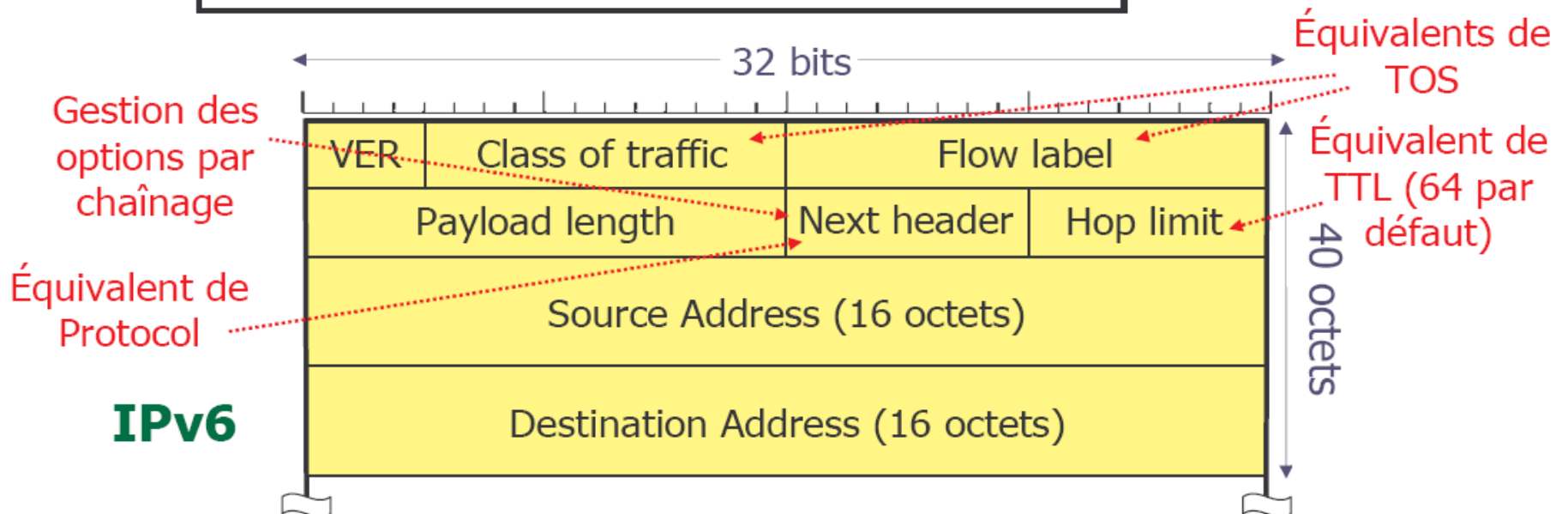
- IPv6 est compatible avec IPv4, TCP, UDP, ICMP, DNS... (ou quelques modifications mineures)
- Supporte un format d'adresses plus longues (16 octets au lieu de 4)
- Simplification de l'entête (7 champs au lieu de 13 et une taille fixe des "options" pour accélérer le traitement dans les routeurs)



# Protocole IPv6



**IPv4**



**IPv6**

# Protocole IPv6

Le champ *Next Header* (NH) :

0 : option "Hop-by-Hop"

4 : IPv4

6 : TCP

17 : UDP

43 : option "Routing Header"

44 : option "Fragment Header"

45 : Interdomain Routing Protocol

46 : RSVP

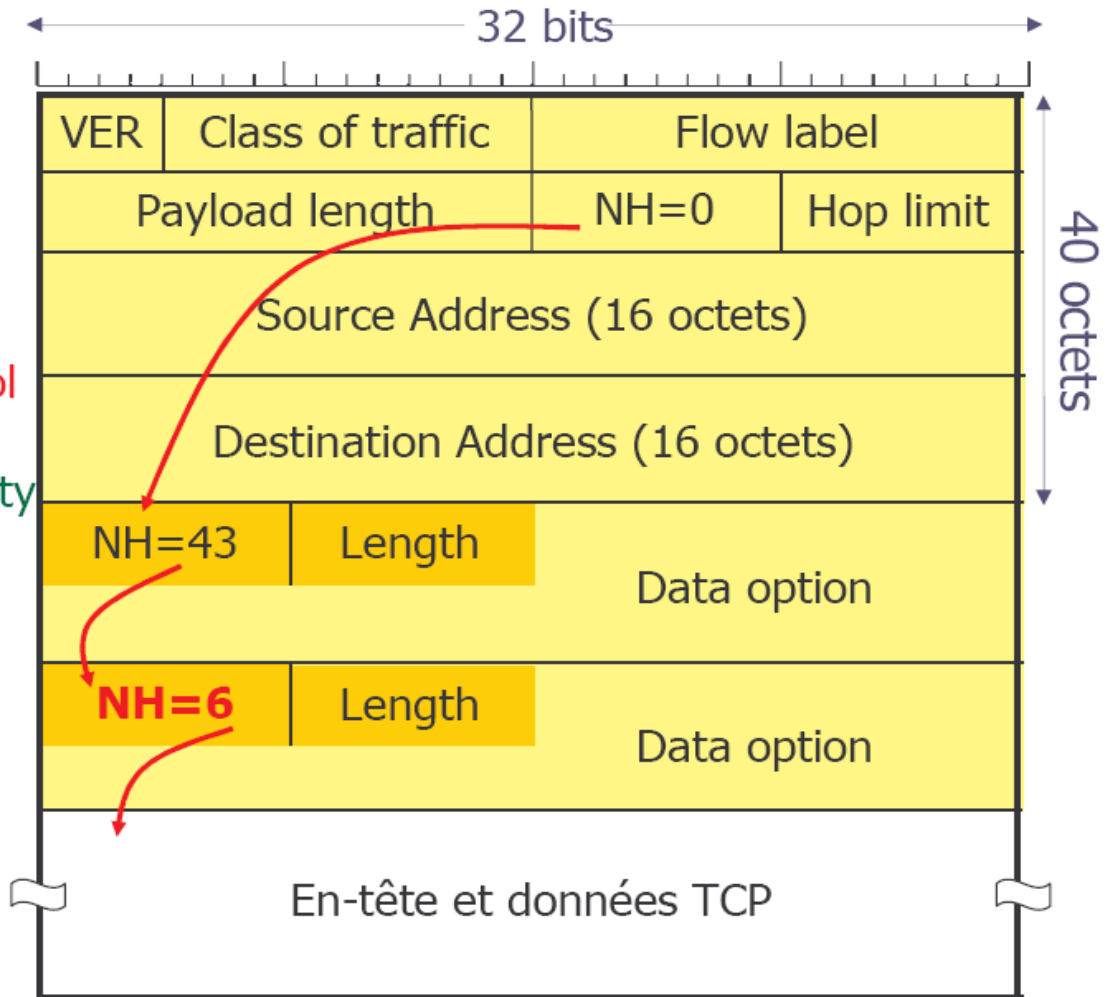
50 : option "Encapsulation Security Payload" (IPsec)

51 : option "Authentication Header" (IPsec)

58 : ICMP

59 : No next header

60 : option "Destination Options Header"



# Protocole IPv6

---

## Exemples d'options

- Routing Header
  - Liste de routeurs à traverser obligatoirement
- Fragmentation Header
  - Pour permettre au destinataire de réassembler les fragments (reprend les champs de IPv4)
- Destination Options Header
  - Informations additionnelles pour la destination

## L'adressage



- Adressage hiérarchique pour alléger les tables de routage
- Un préfixe de localisation – public – 48 bits
- Un champ de topologie locale (subnet) – 16 bits
- Un identifiant de désignation de l'interface (basé sur l'@MAC) sur 64 bits (équivalent HOST\_ID qui garantie l'unicité de l'adresse)

# Organisation

## ■ CM

### Plan

Modèles en couches : OSI & TCP/IP

**Couche internet (IP, ICMP, ARP/RARP)**

Couche transport (TCP, UDP)

Couche application (HTTP, DNS, SMTP, FTP...)

Programmation Réseaux

# **Couche Internet (2)**

## **Routage**

Routage statique (Commandes)

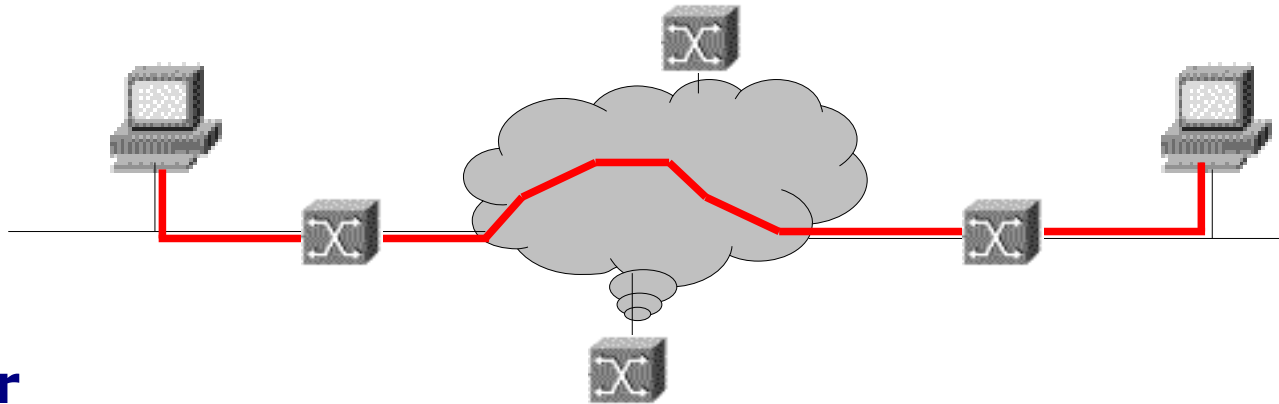
Routage dynamique (Protocoles RIP, OSPF, EGP, BGP)

# Routage

---

Quel chemin empruntent les datagrammes pour arriver à destination ?

**Routage** : mécanisme par lequel les données d'un équipement expéditeur sont acheminées jusqu'à leur destinataire



## ■ Routeur

- dispositif permettant de **choisir le chemin** que les datagrammes vont emprunter
- possède **plusieurs cartes réseau** dont chacune est reliée à un réseau différent
- utilise la **table de routage** qui définit le chemin à emprunter pour une adresse donnée

# Routage

---

## Table de routage

Définit la correspondance entre l'adresse de la machine visée et le noeud suivant auquel le routeur doit délivrer le message

### Contenu de la table de routage

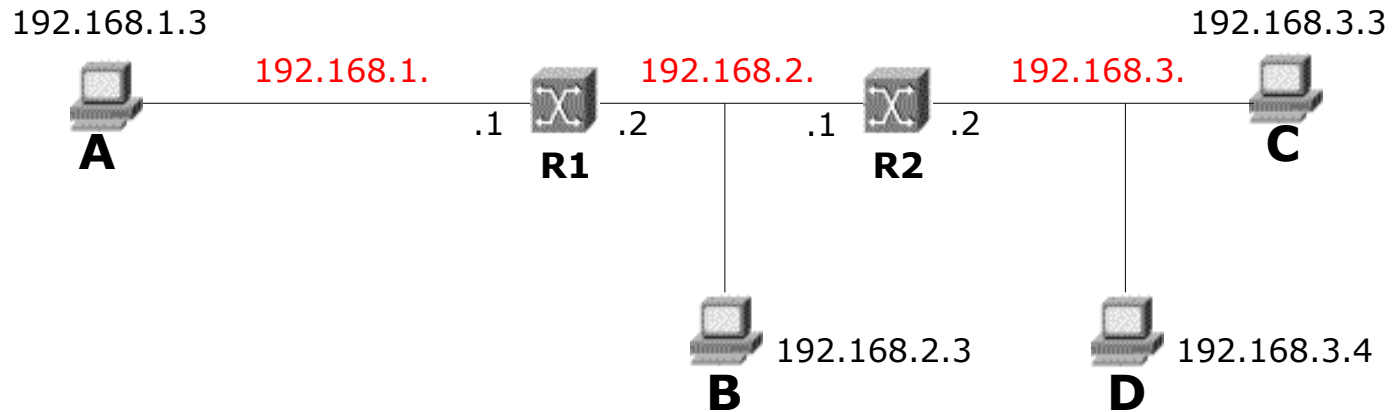
<b>destination</b>	IP d'une machine ou d'un réseau de destination
<b>passerelle (gateway)</b>	IP du prochain routeur vers lequel il faut envoyer le datagramme
<b>masque (mask)</b>	masque associé au réseau de destination
<b>interface</b>	interface physique par laquelle le datagramme doit réellement être expédié
<b>métrique (cost)</b>	utilisé pour le calcul du meilleur chemin

### Commandes pour afficher le contenu de la table

netstat -r , route print

# Routing

## Exemple



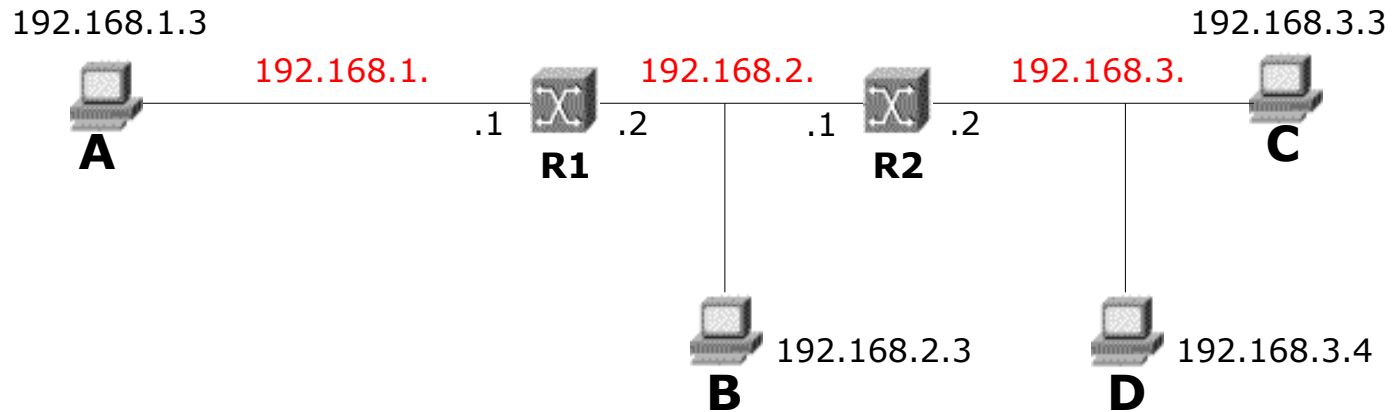
## Table de routage de A

Destination	Netmask	Gateway	Interface	Cost
192.168.1.0	255.255.255.0	-	192.168.1.3	0
192.168.2.0	255.255.255.0	192.168.1.1	192.168.1.3	1
0.0.0.0	0.0.0.0	192.168.1.1	192.168.1.3	0



# Routage

## Exemple



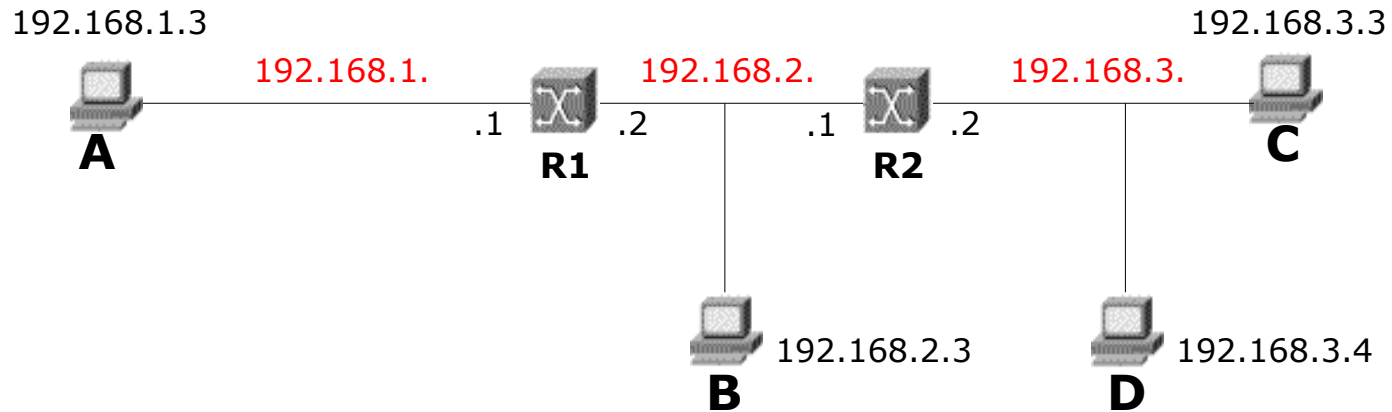
## Table de routage de B

Destination	Netmask	Gateway	Interface	Cost
192.168.1.0	255.255.255.0	192.168.2.2	192.168.2.3	1
192.168.3.0	255.255.255.0	192.168.2.1	192.168.2.3	1
192.168.2.0	255.255.255.0	-	192.168.2.3	0

**Quelle doit être la table de routage de C pour qu'il puisse communiquer avec les autres réseaux ?**

# Routing

## Exemple

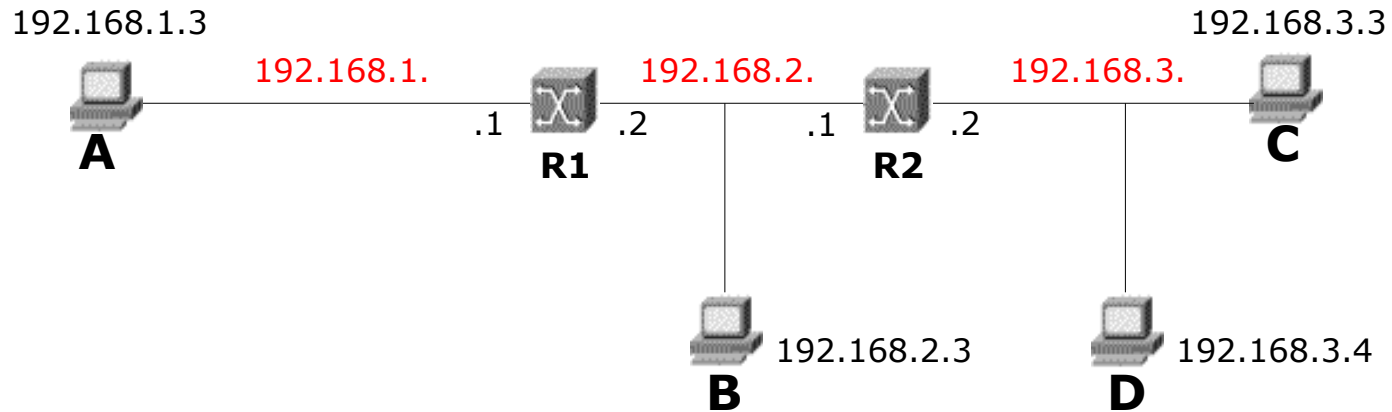


## Table de routage de R1

Destination	Netmask	Gateway	Interface	Cost
192.168.1.0	255.255.255.0	-	192.168.1.1	0
192.168.2.0	255.255.255.0	-	192.168.2.2	0
192.168.3.0	255.255.255.0	192.168.2.1	192.168.2.2	1

# Routage

## Exemple



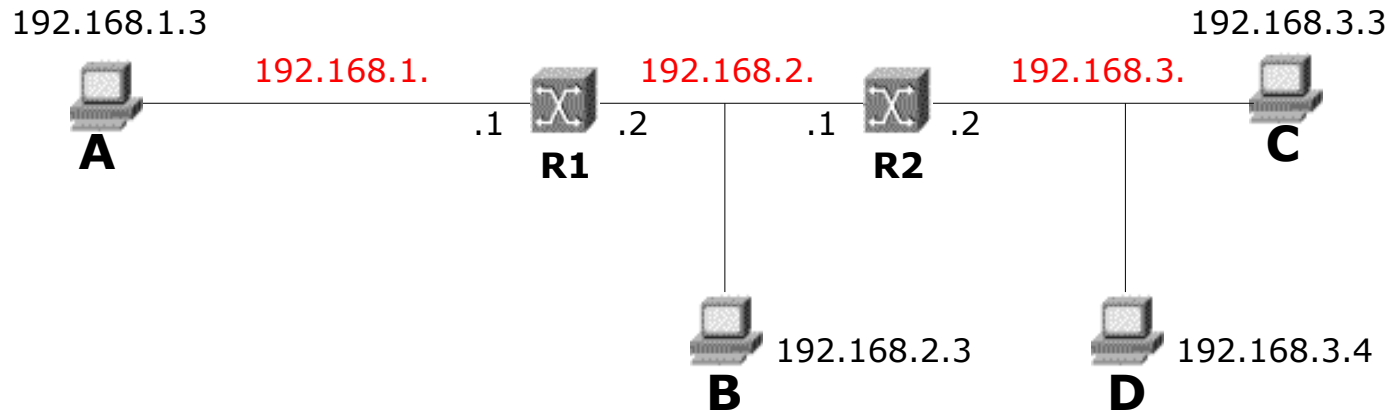
## Table de routage de R2

Destination	Netmask	Gateway	Interface	Cost
192.168.2.0	255.255.255.0	-	192.168.2.1	0
192.168.3.0	255.255.255.0	-	192.168.3.2	0

**Modifier la table de routage de R2 pour que le réseau 192.168.3.0 puisse communiquer avec le réseau 192.168.1.0**

# Routing

## Exemple



## Table de routage de R2

Destination	Netmask	Gateway	Interface	Cost
192.168.2.0	255.255.255.0	-	192.168.2.1	0
192.168.3.0	255.255.255.0	-	192.168.3.2	0
192.168.1.0	255.255.255.0	192.168.2.2	192.168.2.1	1

# Routage

---

## Mise à jour de la table de routage

### ■ Manuelle « routage statique »

- table de routage entrée manuellement par l'administrateur
- commande « route » des stations unix
- langage de commande des routeurs (ip route...)

### ■ Automatique « dynamique »

- table de routage mis à jour dynamiquement par le routeur
- processus sur les stations et les routeurs
- échanges d'informations de routage : **protocoles de routage**
  - Routage basé sur un vecteur de distance
  - Routage basé sur l'état des liens

# Couche Internet (2)

Routage

**Routage statique** (Commandes)

Routage dynamique (Protocoles RIP, OSPF, EGP, BGP)

# Routage statique

---

- La commande **route** permet d'indiquer une route vers:

- un réseau (NET\_ID)
- une machine (HOST\_ID)
- ou une adresse par défaut (default)

- **Syntaxe :**

- route add | del [net | host] destination | netmask | gw | metric

# Routage statique

---

## Route vers une machine

route add 192.168.0.36 netmask 255.255.255.240 eth0 -----> **ajouter**

route del 192.168.0.36 netmask 255.255.255.240 eth0 -----> **supprimer**

## Route vers un réseau

route add -net 192.168.0.0 netmask 255.255.255.240 eth0 -----> **ajouter**

route del -net 192.168.0.0 netmask 255.255.255.240 eth0 -----> **supprimer**

## Route vers un réseau via une passerelle

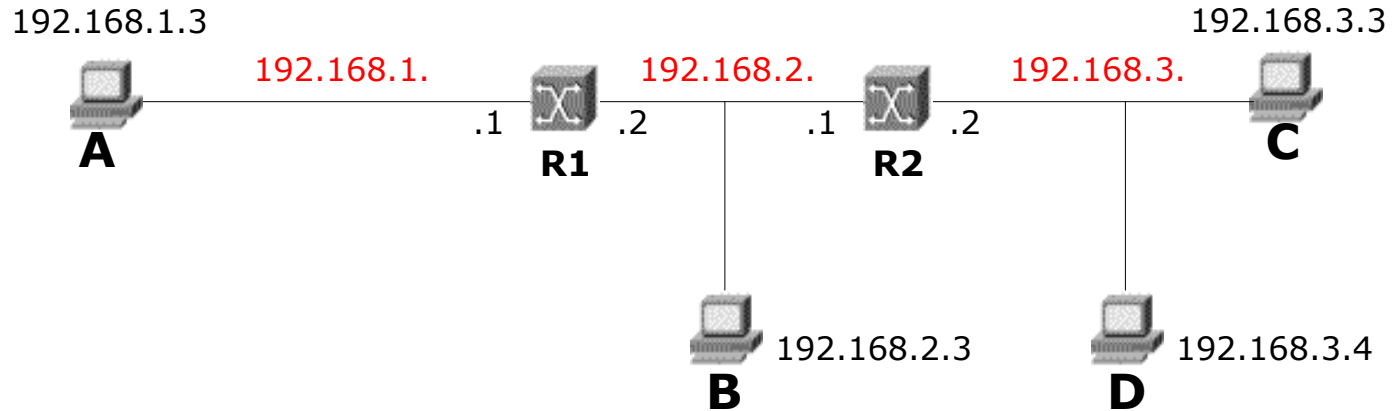
route add -net 192.168.0.0 netmask 255.255.255.240 gw 192.168.0.7 eth0 ---> **ajouter**

route del -net 192.168.0.0 netmask 255.255.255.240 gw 192.168.0.7 eth0 --> **supprimer**



# Routage statique

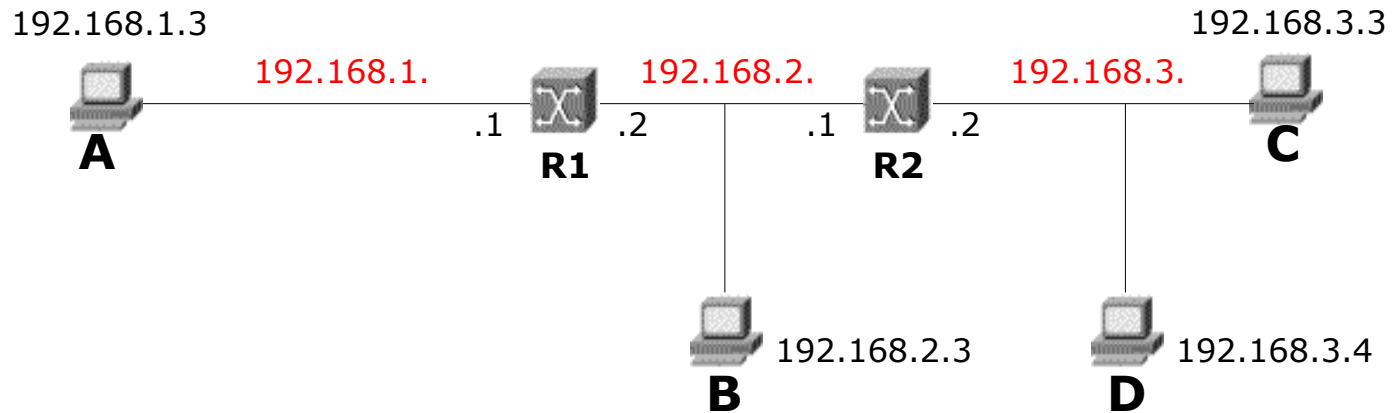
## Exemple



**Configurer la table de routage des différentes machines**

# Routage statique

## Exemple



### Étape 1 : configuration des IP

**A** : `ifconfig eth0 192.168.1.3 netmask 255.255.255.0`

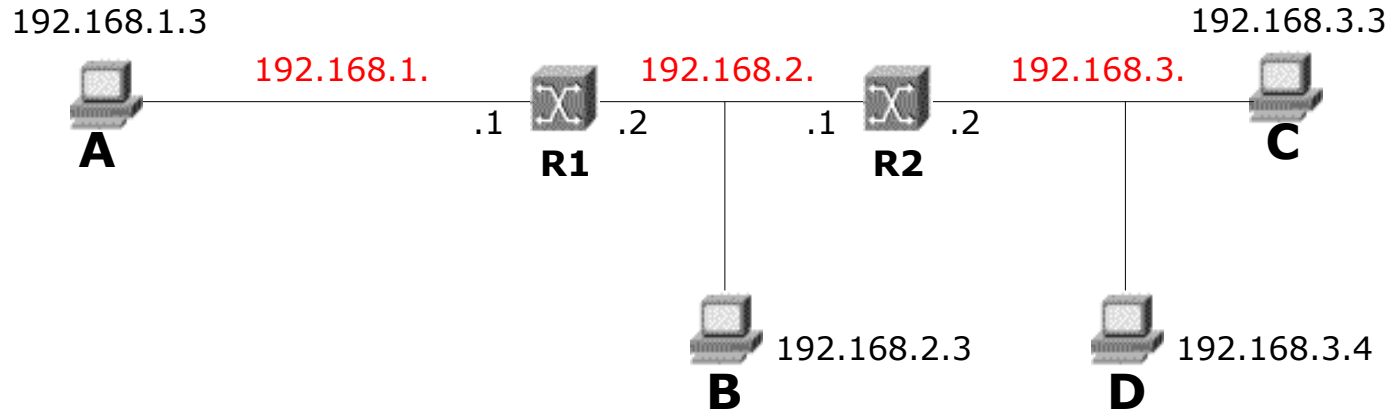
**B** : `ifconfig eth0 192.168.2.3 netmask 255.255.255.0`

**C** : `ifconfig eth0 192.168.3.3 netmask 255.255.255.0`

**D** : `ifconfig eth0 192.168.3.4 netmask 255.255.255.0`

# Routage statique

## Exemple



### Étape 1 : configuration des IP

**R1 :** `ifconfig eth0 192.168.1.1 netmask 255.255.255.0`

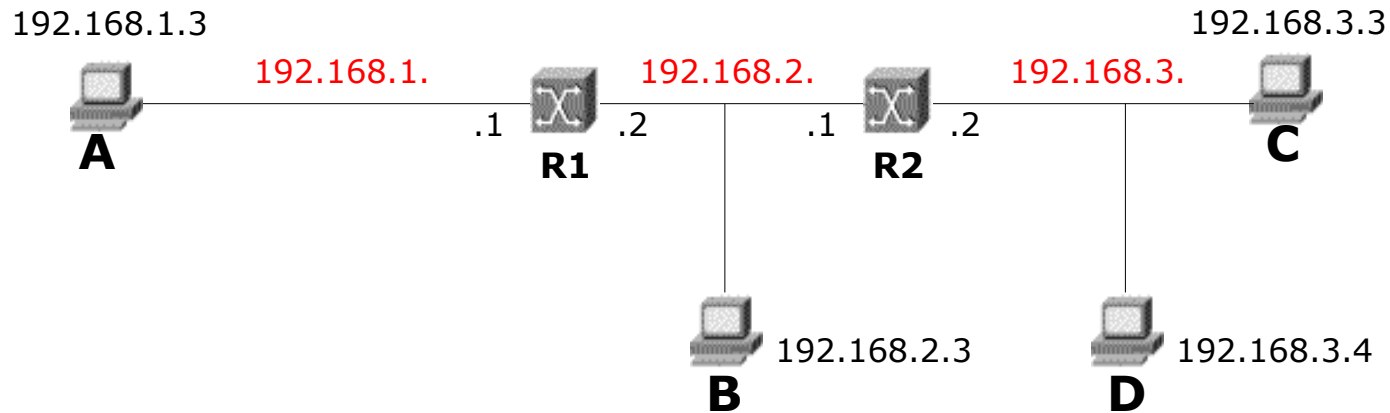
`ifconfig eth1 192.168.2.2 netmask 255.255.255.0`

**R2 :** `ifconfig eth0 192.168.2.1 netmask 255.255.255.0`

`ifconfig eth1 192.168.3.2 netmask 255.255.255.0`

# Routage statique

## Exemple



## Étape 2 : configuration des tables de routage

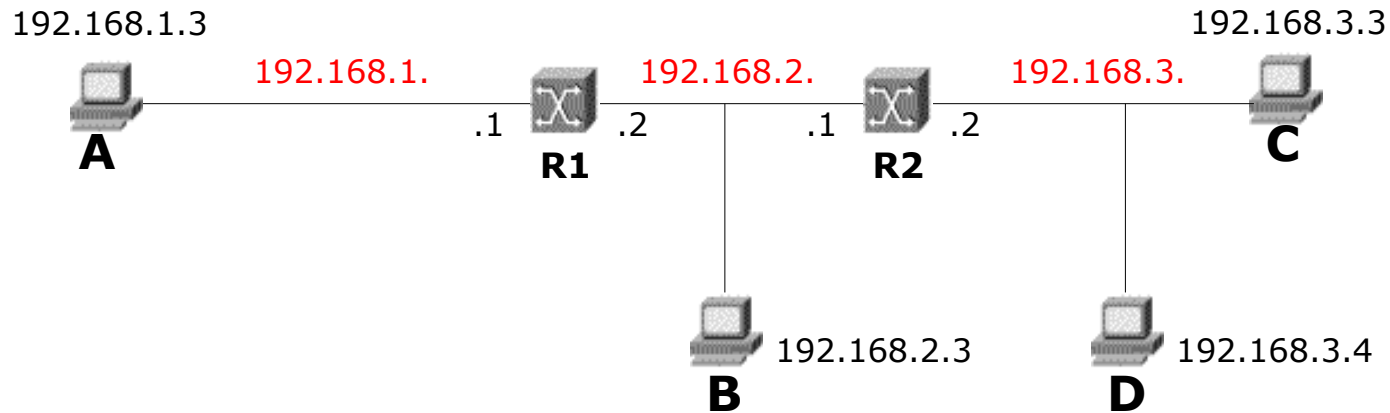
**A :** `route add default gw 192.168.1.1`

**B :** `route add -net 192.168.3.0 gw 192.168.2.1`

`route add -net 192.168.1.0 gw 192.168.2.2`

# Routage statique

## Exemple



## Étape 2 : configuration des tables de routage

**R1 :**

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0 eth0 : 192.168.1.1
route add -net 192.168.2.0 netmask 255.255.255.0 eth1 eth1 : 192.168.2.2
route add -net 192.168.3.0 gw 192.168.2.1
```

**R2 :**

```
route add -net 192.168.2.0 netmask 255.255.255.0 eth0 eth0 : 192.168.2.1
route add -net 192.168.3.0 netmask 255.255.255.0 eth1 eth1 : 192.168.3.2
```

# Couche Internet (2)

Routage

Routage statique (Commandes)

**Routage dynamique** (Protocoles RIP, OSPF, EGP, BGP)

# Routage Dynamique

---

## Mise à jour de la table de routage

### ■ Manuelle « routage statique »

- table de routage entrée manuellement par l'administrateur
- commande « route » des stations unix
- langage de commande des routeurs (ip route...)

### ■ Automatique « dynamique »

- table de routage mis à jour dynamiquement par le routeur
- processus sur les stations et les routeurs
- échanges d'informations de routage : **protocoles de routage**
  - Routage basé sur un vecteur de distance
  - Routage basé sur l'état des liens

# Routage Dynamique

---

## Deux algorithmes clefs

### **Vecteur de distance** – Bellman-Ford

Chaque noeud stocke un "vecteur" pour toutes destinations

- Ce vecteur contient la distance à chacune d'entre elles
- Distance = coût

*Pré condition*

- Chaque noeud connaît la distance vers tous ses voisins directs

### **État des liens** – Dijkstra

Chaque noeud possède un carte complète du réseau

- Contrairement au "vecteur de distance", chaque noeud ne connaît que les états voisins



# Routage Dynamique

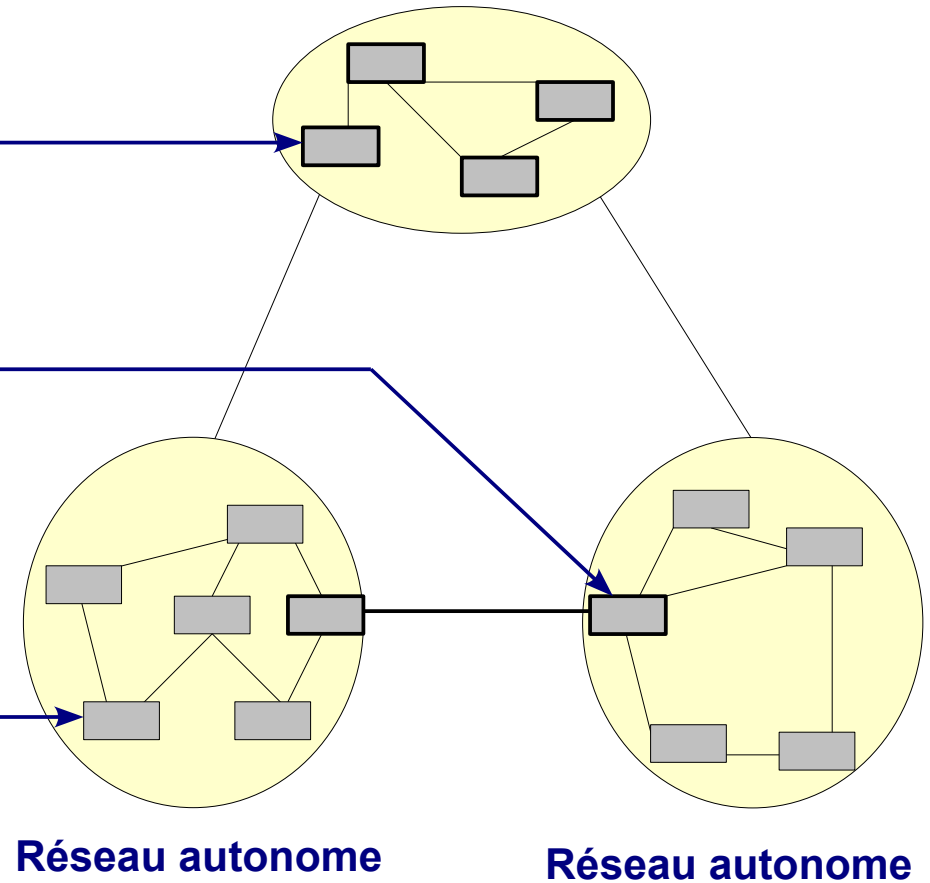
## Types de routeurs

**Routeurs noyaux**  
relient les réseaux

**Routeurs externes**  
permettent une liaison des  
réseaux autonomes entre eux

Protocoles : **EGP** (Exterior Gateway Protocol)  
**BGP** (Border Gateway protocol)

**Routeurs internes**  
routage à l'intérieur d'un réseau autonome  
Protocole : **IGP** (Interior Gateway Protocol)  
**RIP, OSPF, EIGRP**



# Protocoles de routages

---

## RIP - Routing information protocol

- Protocole de type **Vecteur de Distance**
- Chaque 30 seconde le routeur diffuse à ses voisins ses vecteurs de distance
  - vecteur de distance : (destination, nombre de sauts)
  - nombre de sauts maximum = 16 (pour éviter les boucles)
  - utilisable uniquement à l'intérieur de domaines peu étendus
- Si aucun message pendant 180s, route inaccessible
- un noeud construit sa table de routage en fonction des vecteurs de distance reçus de ses voisins

# Protocoles de routages

---

## RIP - Routing information protocol

### ■ Avantages

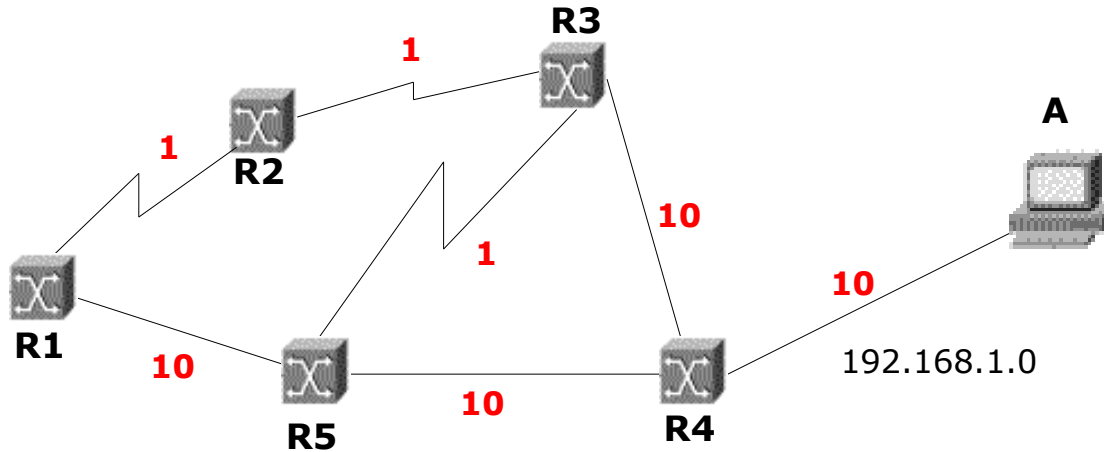
- très utilisé et très répandu sur tous les équipements
- s'adapte automatiquement (panne, ajout de réseau...)

### ■ Inconvénients

- la distance ne tient pas compte de l'**état de la liaison** (la charge, débit, coût des lignes...)
- distance maximale = 15 (ne peut pas aller plus que 15 routeurs)
- trafic important (toutes les 30s un message)
- pas d'authentification des messages (attaques de routeurs en générant des faux messages RIP)

protocole très efficace dans un petit réseau que l'on contrôle  
mais pas adapté aux grands domaines

# Protocoles de routages



**Route de R1 à A ?**

**RIP** : R1 -> R5 -> R4 -> A

**Supposons deux types de liens**  
«rapides» et «lents»

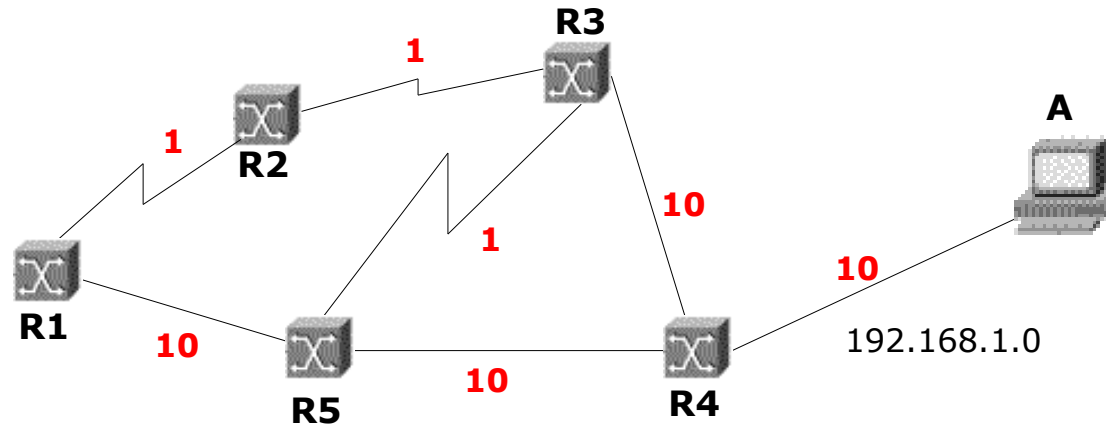
**RIP** n'est plus pertinent !



Protocole de type **état des liens**  
**OSPF -Open Shortest Path First**

# Protocoles de routages

## OSPF - Open Shortest Path First



**Base de données topologique**

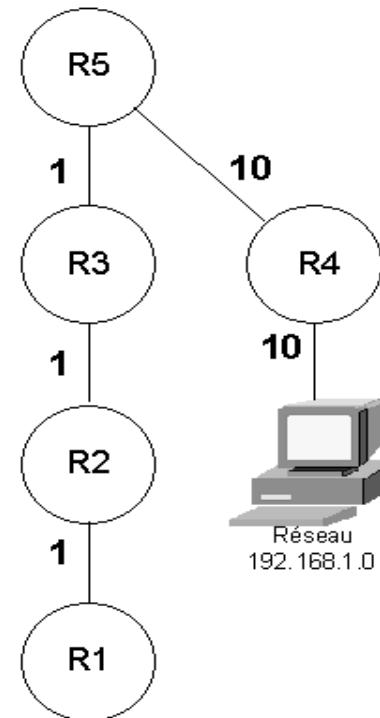
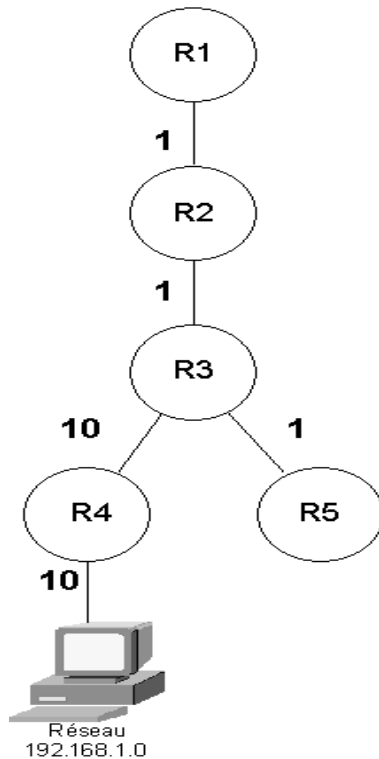
Arc	Coût
R1, R2	1
R1, R5	10
R2, R3	1
R3, R4	10
R3, R5	1
R4, R5	10
R4, 192.168.1.0	10

# Protocoles de routages

## OSPF - Open Shortest Path First

### Élection des meilleures routes

- Algorithme de Dijkstra -



Meilleure route entre R1 et A passe par R2, R3 et R4 pour un coût total de  $1 + 1 + 10 + 10 = 22$

# Protocoles de routages

---

## OSPF - Open Shortest Path First

### Détermination de la table de routage

#### Extrait de la table de R1

Réseau de destination	Moyen de l'atteindre	Coût
192.168.1.0	R2	22

#### Extrait de la table R5:

Réseau de destination	Moyen de l'atteindre	Coût
192.168.1.0	R4	20

# Protocoles de routages

## OSPF - Open Shortest Path First

### Recherche du plus court chemin

#### Algorithme de Dijkstra : Plus courte chaîne du sommet $\alpha$ au sommet $\omega$

Étape 1	Placer tous les sommets du graphe dans la première ligne d'un tableau. La deuxième ligne du tableau est obtenue en écrivant le coefficient $0$ sous l'origine $\alpha$ et le coefficient $\infty$ sous tous les autres sommets. Sélectionner le sommet origine $\alpha$ . Son coefficient est $0$ .
Étape 2	Soit $X$ le sommet de plus petit coefficient. Commencer une nouvelle ligne dite ligne courante. Rayer toutes les cases vides de la colonne $X$ .
Étape 3	Pour tous les sommets adjacents à $X$ répéter Soit $Y$ le sommet courant $p = \text{Coef}(X) + \text{poids de l'arête reliant } X \text{ à } Y$ . <b>SI</b> $p < \text{Coef}(Y)$ <b>ALORS</b> écrire « $p X$ » dans la case correspondante <b>SINON</b> recopier le contenu de la ligne précédente <b>S'il reste des sommets adjacents à } X</b> passer au suivant <b>sinon</b> aller à l'étape 4
Étape 4	Compléter la ligne courante en recopiant les valeurs de la ligne précédente dans les cases vides.
Étape 5	S'il reste des sommets non sélectionnés, retourner à l'étape 2 Sinon aller à l'étape 6.
Étape 6	La plus courte chaîne de $\alpha$ à $\omega$ est obtenue en écrivant de droite à gauche le parcours en partant de $\omega$



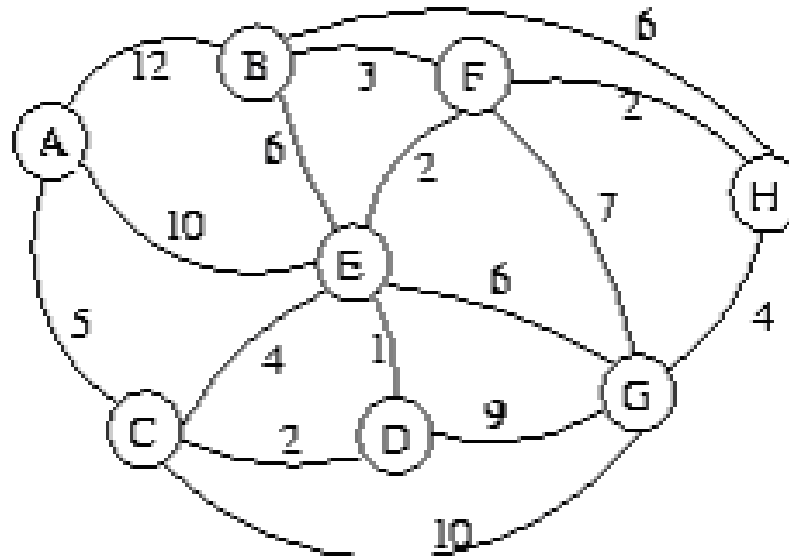
# Protocoles de routages

---

## OSPF - Open Shortest Path First

Recherche du plus court chemin

Calculez les plus courts chemins de A à destination de H, puis de G en utilisant l'algorithme de Dijkstra



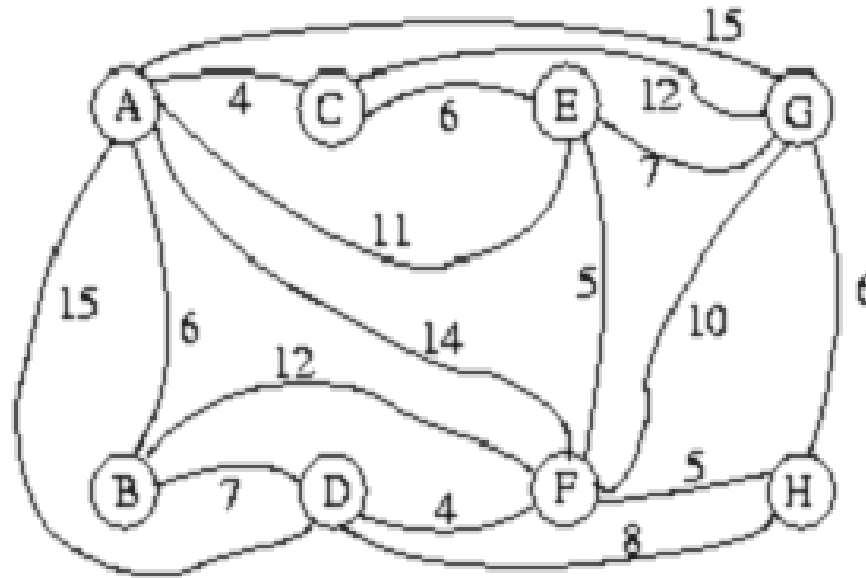
# Protocoles de routages

---

## OSPF - Open Shortest Path First

Recherche du plus court chemin

Calculez les plus courts chemins de A à destination de H, puis de G en utilisant l'algorithme de Dijkstra



# Protocoles de routages

---

## OSPF - Fonctionnement

### 1. État initial

- Le processus de routage OSPF est inactif sur tous les routeurs

### 2. Établir la liste des routeurs voisins

- Chaque routeur se présente et fait connaissance avec ses voisins –  
**message « HELLO, my name is R1 and I'm an OSPF router »**
- Les voisins sauvegardent l'@IP de l'émetteur dans leurs **bases d'adjacences**
- Les voisins répondent en envoyant leurs IP à l'émetteur
- L'émetteur ajoute les adresses IP de ses voisins dans sa **base d'adjacence**

# Protocoles de routages

---

## OSPF - Fonctionnement

### 3. Élire le routeur désigné (DR) et le routeur désigné de secours (BDR)

#### - DR -

- routeur avec la plus grande priorité (=1 par défaut)
- routeur avec la plus grande adresse IP en cas de conflit

#### - BDR -

- routeur avec plus grande priorité juste après le DR

### 4. Découvrir les routes

- Le DR transmet aux routeurs un résumé de sa base de données topologique
- Si les données des routeurs sont plus récentes, ils demandent une information plus complète
- Le DR envoie l'intégralité de l'information demandée
- Les routeurs (non DR ou BDR) transmettent les routes meilleures ou les routes inconnues du DR.

# Protocoles de routages

---

## OSPF - Fonctionnement

### 5. Élire les routes à utiliser

- Création de la table de routage -> Algorithme de Dijkstra

### 6. Maintenir la base topologique

- Chaque routeur envoie ses états de liens au routeur désigné avec des messages « mises à jour d'état de lien »
- Le DR et le BDR intègrent cette information à leur base topologique
- Ces messages sont acquittés pour plus de fiabilité
- Toute modification de la topologie déclenche une nouvelle exécution de l'algorithme Dijkstra et une nouvelle table de routage est constituée

# Protocoles de routages

---

## OSPF - Résumé

- Protocole de type **état des liens**
- chaque noeud évalue le coût pour rejoindre ses voisins selon une certaine métrique (**plusieurs métrique peuvent être utilisées simultanément**)
  - construit un paquet contenant les informations relatives à chacun de ses liens (voisins)
  - le diffuse à tout le monde (par inondation)
  - calcule la route de moindre coût pour atteindre chaque entité du réseau
  - ensuite, les routeurs s'échangent uniquement les changements détectés dans la topologie
  - chaque noeud a une vision globale de la cartographie du réseau

# Protocoles de routages

---

## OSPF - Conclusion

- Protocole complexe encore peu mis en oeuvre mais
  - remédie aux inconvénient de RIP
    - temps de convergence
    - OSPF adapté aux grands domaines
    - OSPF prend en compte plusieurs métriques
- autres avantages d'OSPF
  - permet de router les sous-réseaux
  - permet l'équilibrage de charge entre différentes routes de même coût
  - inclut un système d'authentification des messages

# Routage Dynamique

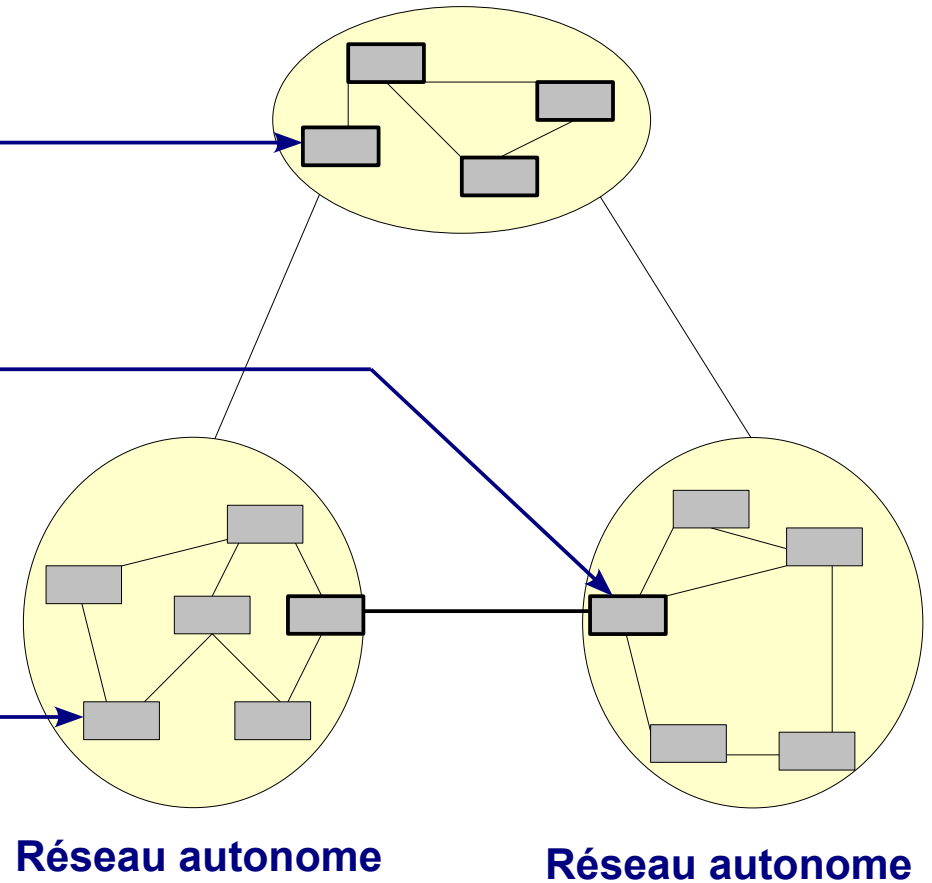
## Types de routeurs

**Routeurs noyaux**  
relient les réseaux

**Routeurs externes**  
permettent une liaison des  
réseaux autonomes entre eux

Protocoles : **EGP** (Exterior Gateway Protocol)  
**BGP** (Border Gateway protocol)

**Routeurs internes**  
routage à l'intérieur d'un réseau autonome  
Protocole : **IGP** (Interior Gateway Protocol)  
**RIP, OSPF, EIGRP**

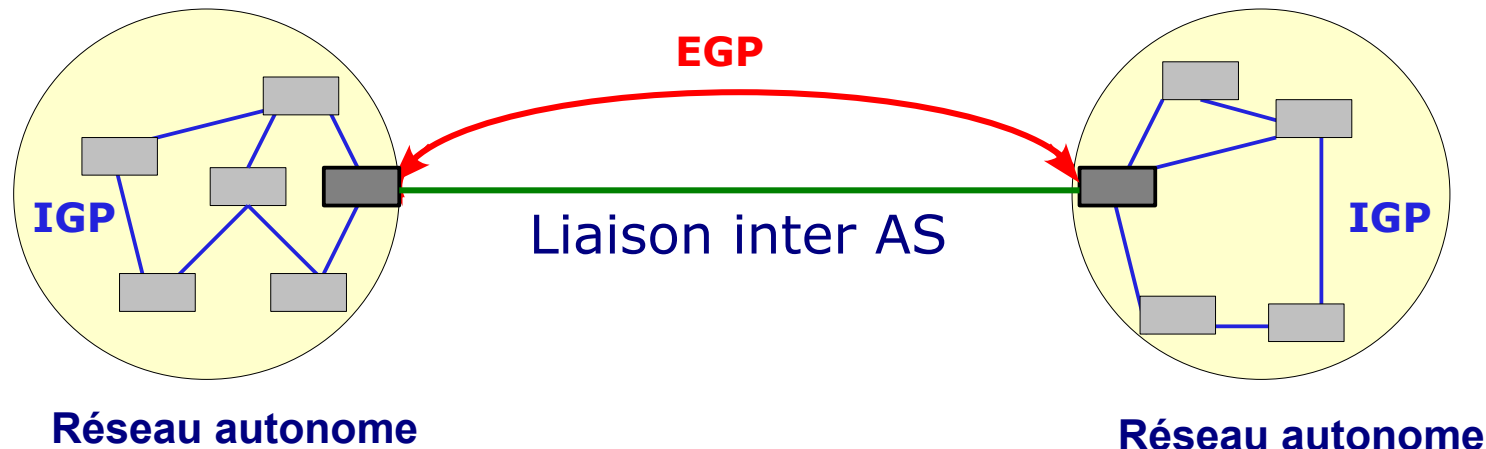




# Protocoles de routages

## EGP – Exterior Gateway Protocol

- Premier protocole externe utilisé dans internet (désormé remplacé par BGP)
- Echange entre routeurs déclarés comme «pairs »
  - deux routeurs de bordure s'échangent à l'intervalles réguliers **la liste des réseaux accessible** dans leurs AS respectives
  - tout le trafic entre 2 AS passe par le même chemin physique



# Protocoles de routages

---

## BGP – Border Gateway Protocol

### Stratégie de routage

- Besoin de prendre en compte dans les stratégies de routage des considérations d'ordres
  - **politique** : certain AS peuvent refuser de faire transiter du trafic externe ou le trafic sortant de tel AS préfère transiter par tel AS que tel autre...
  - **de sécurité** : trafic en provenance de tel AS ne doit pas transiter par tel AS
  - **économique** : la traversée d'une AS peut être payante...
- La prise en compte de ces stratégies ne fait pas partie du protocole (configuration manuelle des routeurs à l'aide de scripts)

# Protocoles de routages

---

## BGP – Border Gateway Protocol

### Principe de routage

Deux routeurs BGP établissent une connexion TCP pour s'échanger des infos de routage:

- numéro de l'AS
- liste des sous-réseaux de l'AS
- distance relative vers chacun des sous-réseaux de l'AS
- adresse IP du routeur (interne) d'accès à ces réseaux

### Quatre types de messages

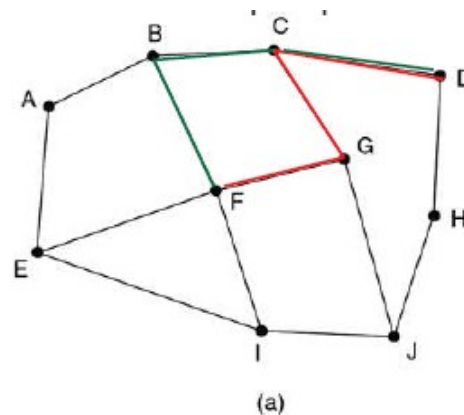
- message d'ouverture : ouverture d'une session BGP entre deux routeurs
- message de mise à jour : signaler à un peer router le changement d'état d'une route interne à l'AS
- message de notification : clore une session BGP
- message « hello » message signalant que tout va bien au routeur voisin (keep Alive)

# Protocoles de routages

## BGP – Border Gateway Protocol

### Principe de routage

- Type vecteur de distance mais les paires s'échangent le chemin complet correspondant à chaque destination (pas uniquement le coût)
- **Exemple** : pour la destination D, F utilise actuellement FGCD et apprend d'autres routes de ses voisins (il peut alors choisir celle qu'il préfère selon la stratégie choisie)



Informations reçues par F de la part de ses voisins à propos de D

De la part de B : «J'utilise BCD»  
De la part de G : «J'utilise GCD»  
De la part de I : «J'utilise IFGCD»  
De la part de E : «J'utilise EFGCD»

(b)

# Organisation

## ■ CM

### Plan

Modèles en couches : OSI & TCP/IP

Couche internet (IP, ICMP, ARP/RARP)

**Couche transport (TCP, UDP)**

Couche application (HTTP, DNS, SMTP, FTP...)

Programmation Réseaux

# Couche Transport

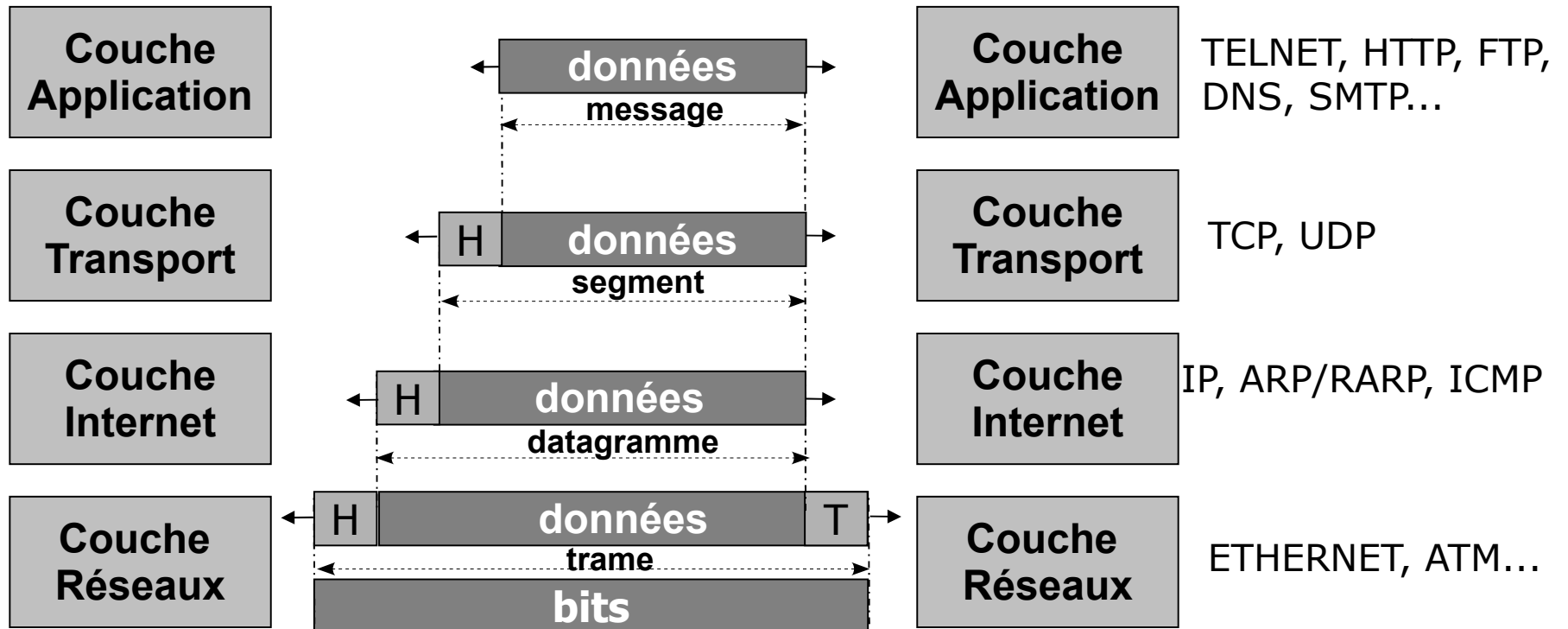
## Services de la couche

Protocole UDP

Protocole TCP

# Modèle TCP/IP

## Protocoles



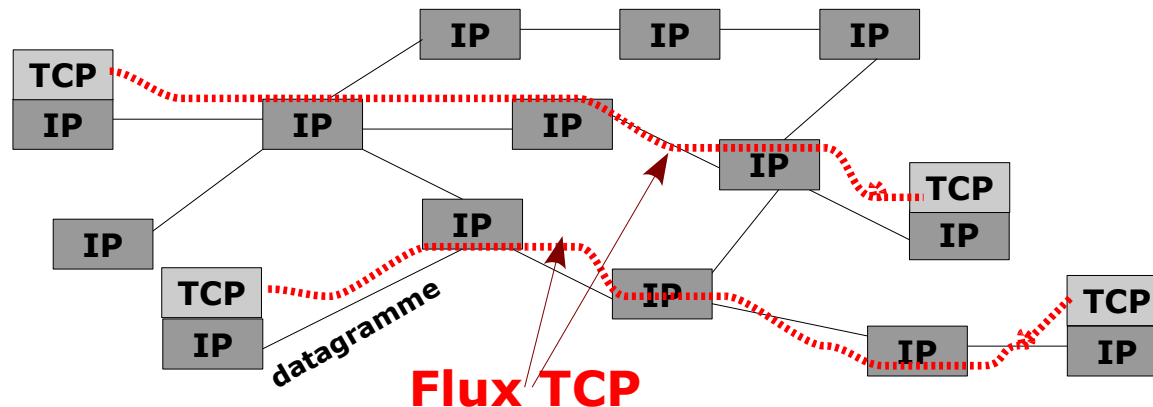
# Couche transport

## Services et protocoles de la couche

- Crée un circuit de communication logique entre des applications s'exécutant sur des hôtes distants
- Les protocoles de la couche transport ne s'exécutent qu'au extrémités

## Services transport vs réseau :

- **Couche réseau** : Transfert de données entre machines
- **Couche transport** : Transfert de données entre applications
  - Se fonde sur les services de la couche réseau et les améliore





# Couche transport

---

## Services de transport

### Livraison fiable (TCP)

- réception des segments dans l'ordre
- contrôle de congestion
- contrôle de flot
- mise en place de connection

### Livraison non fiable (UDP)

- sans garantie d'ordre
- peut s'étendre au multicast

### Services non disponibles

- temps-réel, garantie de délai
- garantie de bande passante
- multicast fiable

# Couche transport

## Multiplexage

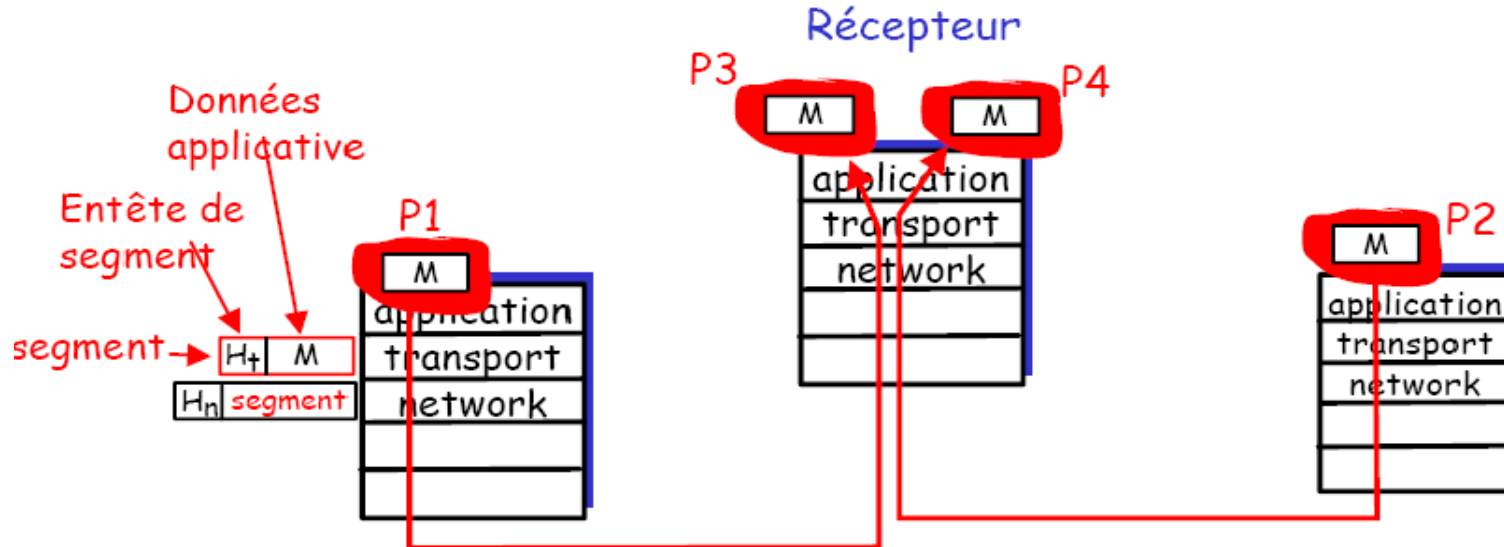
Encapsuler les données de plusieurs applications dans un même segment avec un en-tête qui permettra le démultiplexage

## Demultiplexage

Distribuer chaque segment à l'application à laquelle il est destiné

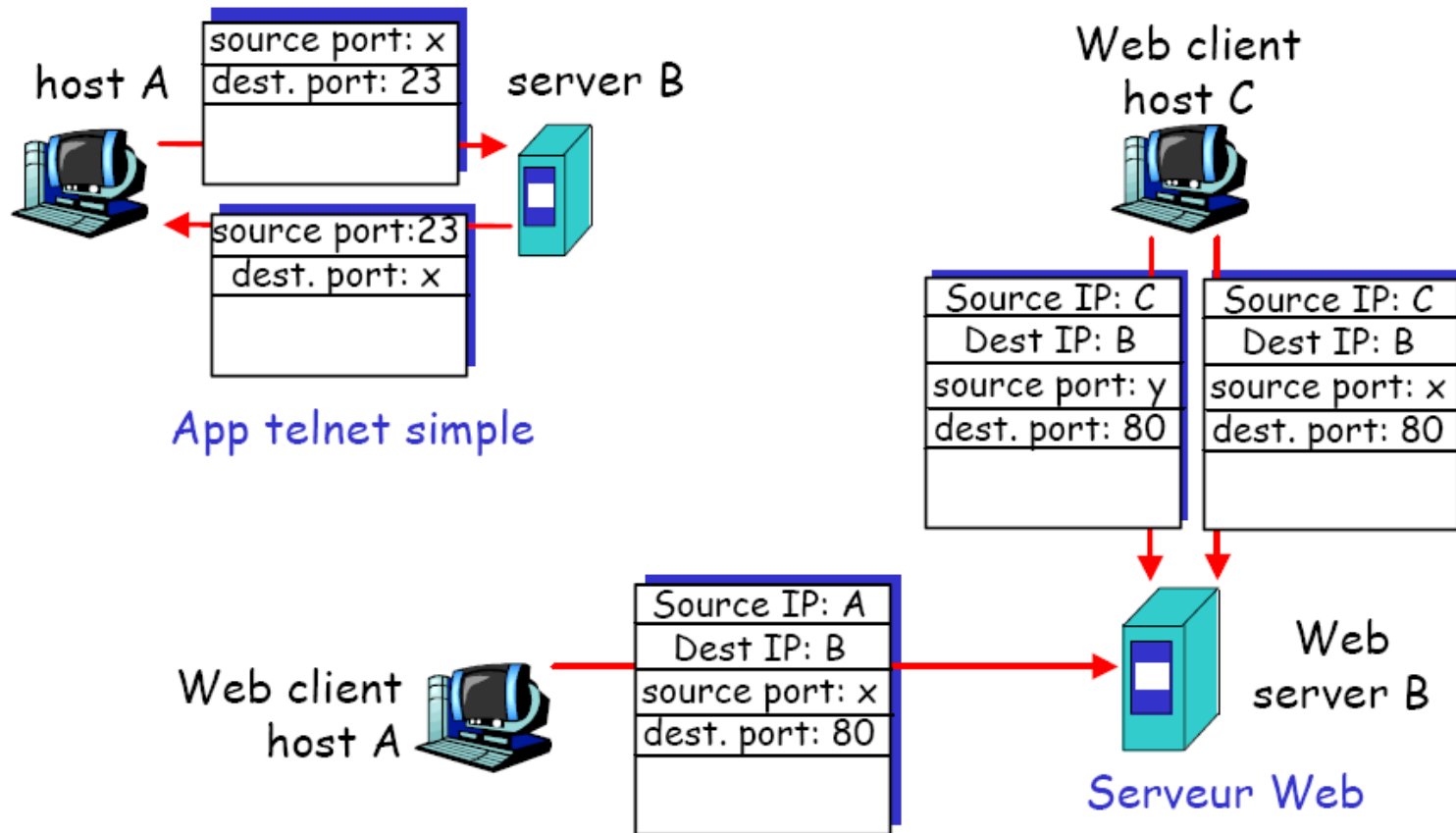
## Multiplexage/demultiplexage

- Fondé sur le **port de réception**, le **port d'émission** et les **adresses IP**
- Les ports source et destination sont répétés dans chaque segment
- Certaines applications utilisent des ports spécifiques



# Couche transport

## Multiplexage/demultiplexage: exemples



# Couche transport

---

- Le numéro de port est un entier de 16 bits [0, 65535]
- Le port 0 n'est pas exploitable
- Distribués par l'IANA "Internet Assigned Numbers Authority"

## Catégories des ports

### 1. Le segment [1, 1023]

- réservés aux services bien connus
- désignés par l'IANA
- mis en oeuvre par des applications avec des droits privilégiés

### 2. Le segment [1024, 49151]

- services enregistrés
- énumérés par l'IANA
- peuvent être employés par des processus ayant des droits ordinaires

### 3. Le segment [49152, 65535]

- attributions dynamiques

# Couche transport

---

## ■ Visualisation des services

- Unix : /etc/services
- Windows : C:\Winnt\System32\drivers\etc\services

Nom	Port	Proto	Commentaire
echo	7	tcp	
echo	7	udp	
ftp-data	20	tcp	#File Transfer [Default Data]
ftp-data	20	udp	#File Transfer [Default Data]
ftp	21	tcp	#File Transfer [Control]
ftp	21	udp	#File Transfer [Control]
telnet	23	tcp	
telnet	23	udp	
smtp	25	tcp	mail #Simple Mail Transfer
smtp	25	udp	mail #Simple Mail Transfer
domain	53	tcp	#Domain Name Server
domain	53	udp	#Domain Name Server
http	80	tcp	www www-http #World Wide Web HTTP
http	80	udp	www www-http #World Wide Web HTTP
pop3	110	tcp	#Post Office Protocol - Version 3
pop3	110	udp	#Post Office Protocol - Version 3
sunrpc	111	tcp	rpcbind #SUN Remote Procedure Call
sunrpc	111	udp	rpcbind #SUN Remote Procedure Call
nntp	119	tcp	usenet #Network News Transfer Protocol
nntp	119	udp	usenet #Network News Transfer Protocol

# Couche Transport

Services de la couche

**Protocole UDP**

Protocole TCP

# Protocole UDP

---

- UDP – User Datagram Protocol
- Protocole simple
- Service “au mieux”, les segments UDP peuvent être:
  - Perdus
  - Delivrés dans le désordre
  - Aucun contrôle de flux ou de récupération d’erreurs
- Mode sans connexion
  - Sans handshaking entre l’émetteur et le récepteur
  - Chaque segment UDP est traité indépendamment des autres
- Sans contrôle de congestion
  - UDP peut émettre aussi rapidement qu’il le souhaite

o  
d  
e  
S  
a  
n  
s  
c  
o  
n  
n  
e  
x  
i  
o  
n

# Protocole UDP

---

## ■ Pourquoi UDP?

- sans délai de connexion
- simple donc plus rapide (pas de délai, pas d'état entre émetteur/récepteur)
- Petit en-tête
- Souvent utilisé pour **les applications multimédias** (streaming multimedia)
  - Tolérance aux pertes
  - Sensible au débit

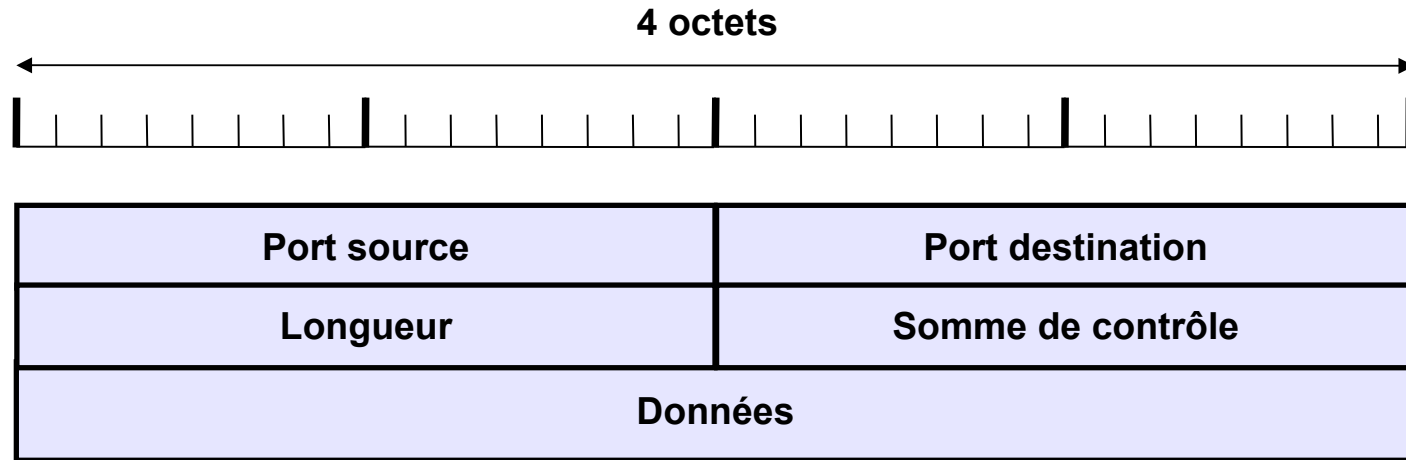
## ■ Autre utilisation d'UDP

- DNS
- SNMP
- Transfert fiable sur UDP
  - ajouter des mécanismes de compensation de pertes à niveau applicatif
  - compensation de pertes adaptée à chaque application



# Segment UDP

---



**Port source** : numéro de port de l'application émettrice du segment

**Port destination** : numéro de port de l'application destinataire

**Longueur** : longueur de l'en-tête + données

- La longueur minimal est 8
- La longueur maximale est 65 535

**Somme de contrôle** : code de contrôle d'erreurs

# Couche Transport

Services de la couche

Protocole UDP

**Protocole TCP**

# Protocole TCP

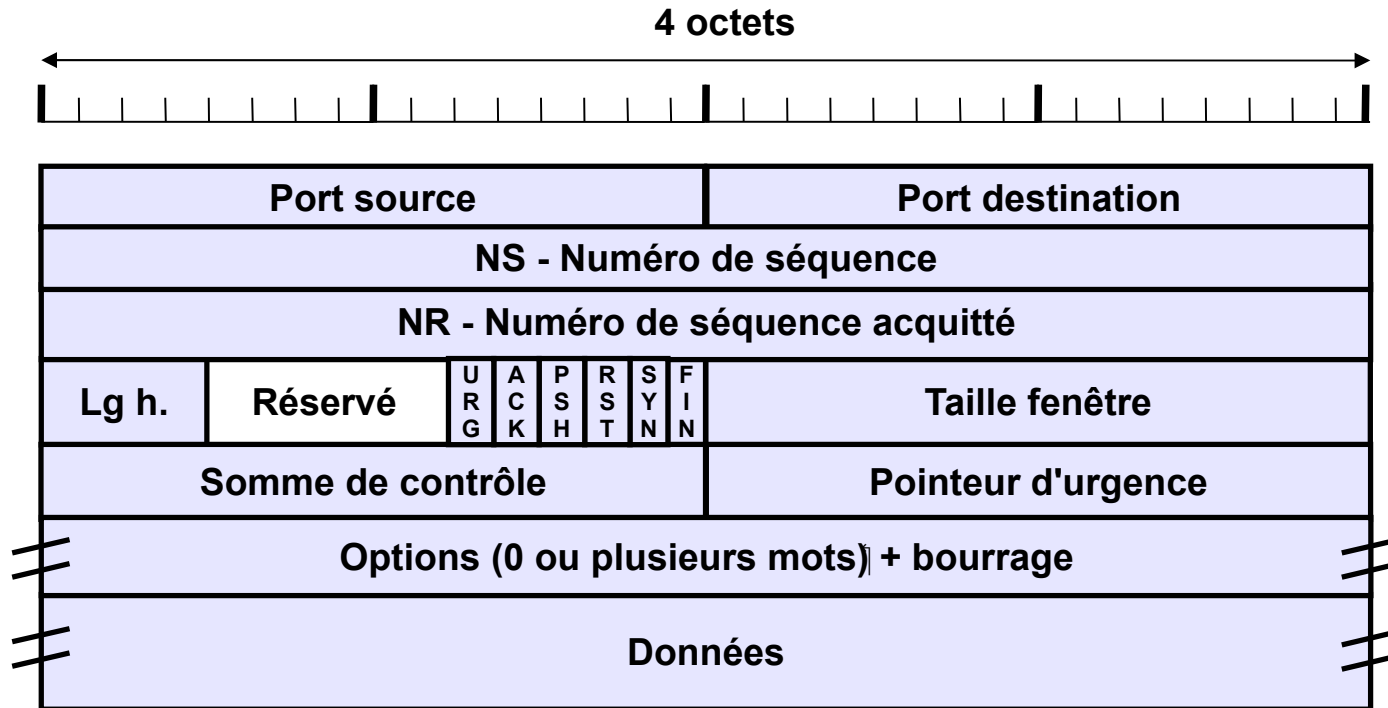
---

- TCP - Transport Control Protocol (*Protocole de Contrôle de Transmission*)

## Caractéristiques du protocole

- Arrivée garantie des données
- Récupération des erreurs par réémission
- Re-assemblage des données dans le bon ordre
- Vérification du flot de données afin d'éviter une saturation du réseau
- Multiplexage/démultiplexage des données
- Initialisation et fin d'une communication
- Communication en mode connecté
  - ✓ Ouverture d'un canal
  - ✓ Communication Full-Duplex
  - ✓ Fermeture du canal

# Segment TCP



**Port Source (2 oct)** : numéro de port de l'application émettrice

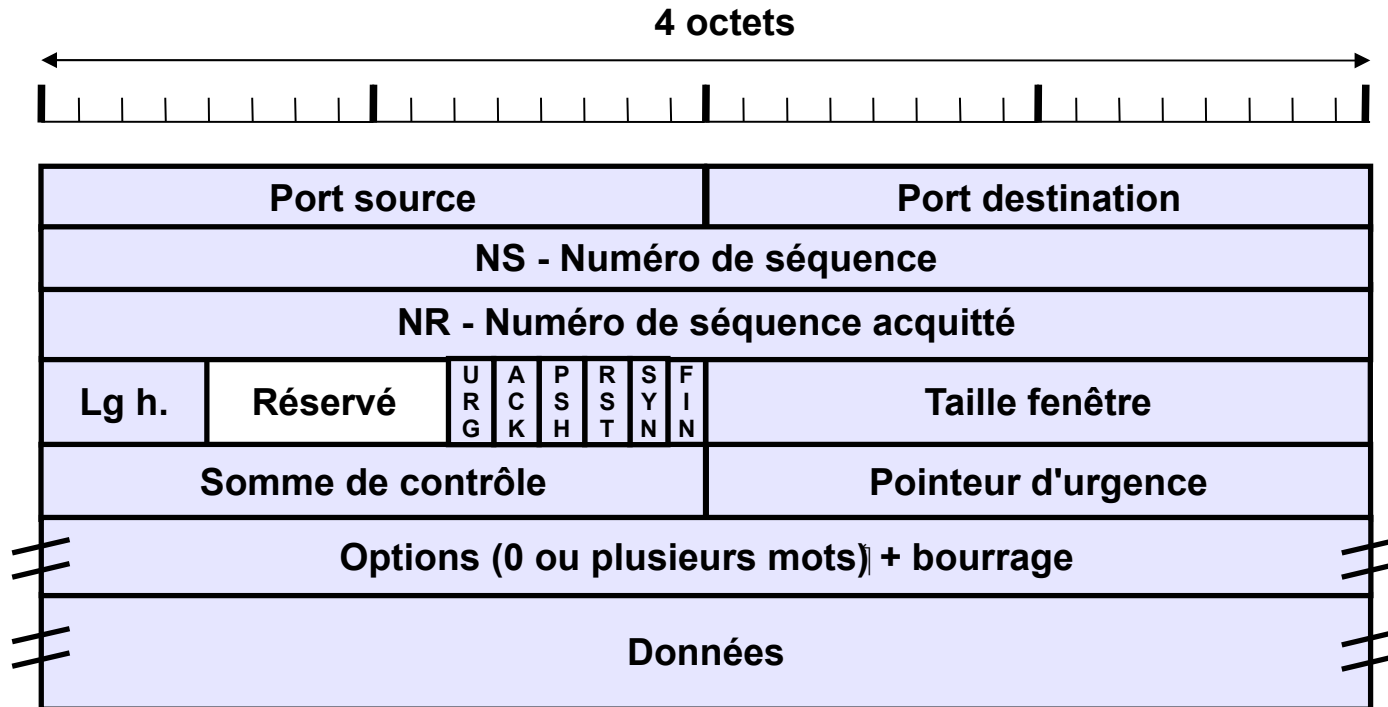
**Port Destination (2 oct)** : numéro de port de l'application distante

**NS (4 oct)** : position du premier octet du segment à transmettre  
Si SYN = 1, NS = ISN (Initial Sequence Number)

**NR (4 oct)** : numéro du prochain octet attendu

**Lg h. (4 bits)** : longueur de l'en-tête en multiple de 4 octets

# Segment TCP



**Réservé (6 bits)** : champ inutilisé actuellement

**CODE (6 bits) URG**: si 1 le paquet doit être traité de façon urgente

**ACK**: si 1 le paquet est un accusé de réception

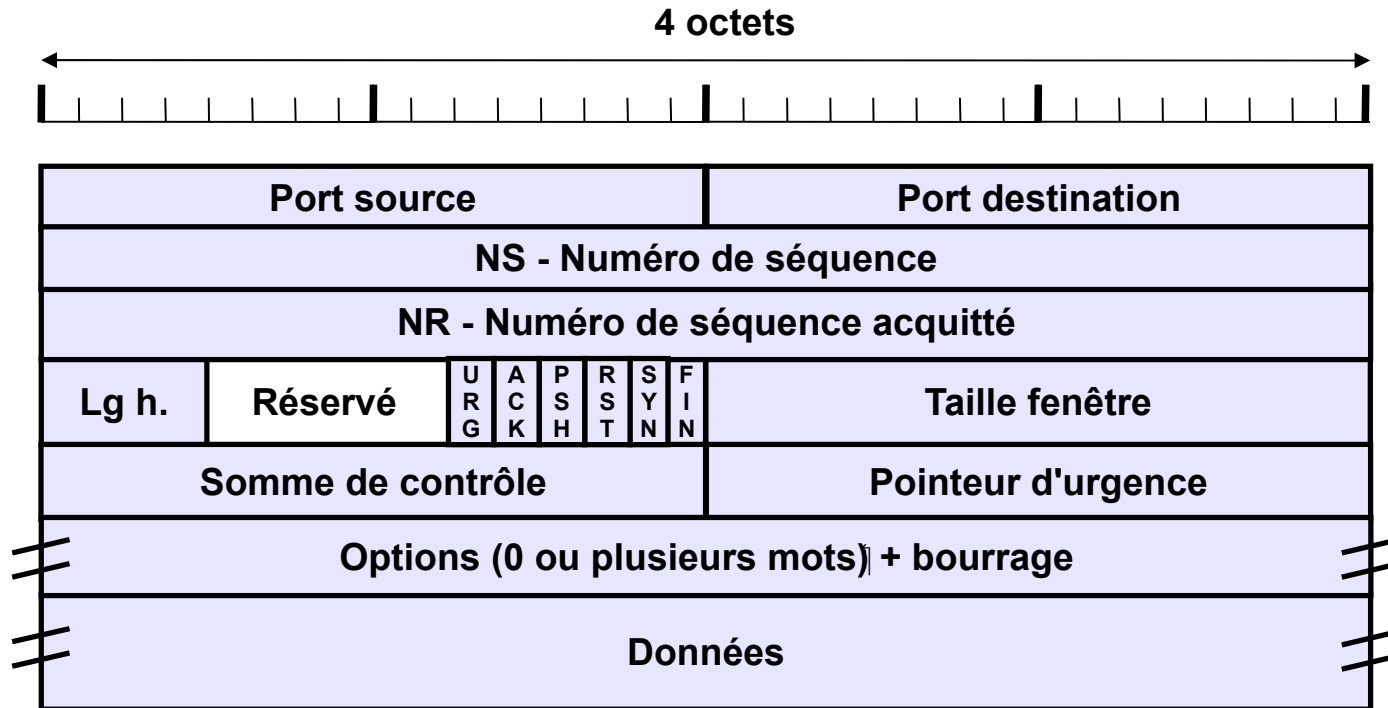
**PSH (PUSH)**: si 1 les données collectées doivent être transmises à l'application sans attendre les données qui suivent

**RST**: si 1 la connexion est réinitialisée

**SYN**: si 1 indique une demande d'établissement de connexion

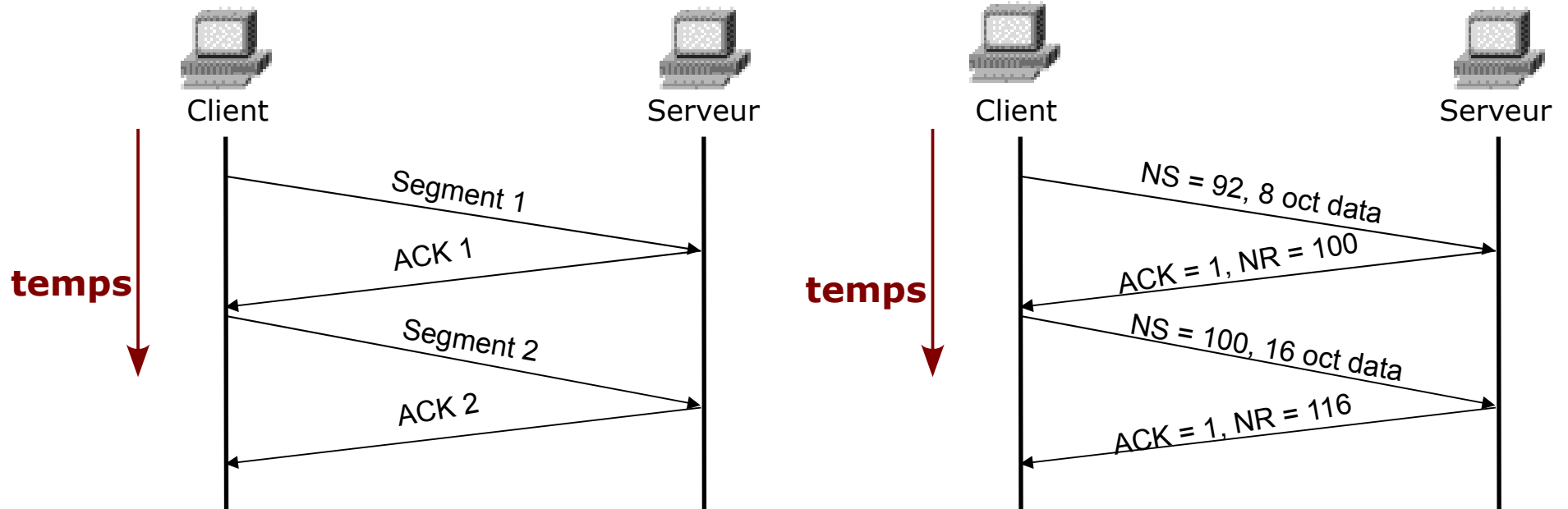
**FIN**: si 1 la connexion s'interrompt

# Segment TCP



- Taille fenêtre (2 oct) :** nb d'octets que le récepteur peut recevoir sans acquittement
- Somme de contrôle (2 oct) :** permet de vérifier l'intégrité de l'en-tête
- Pointeur d'urgence (2 oct):** numéro d'ordre à partir duquel l'information devient urgente
- Options (Taille variable):** diverses options
- Bourrage (Taille variable) :** bits à zéro pour avoir une longueur en-tête multiple de 32 bits

# Fiabilité des transferts



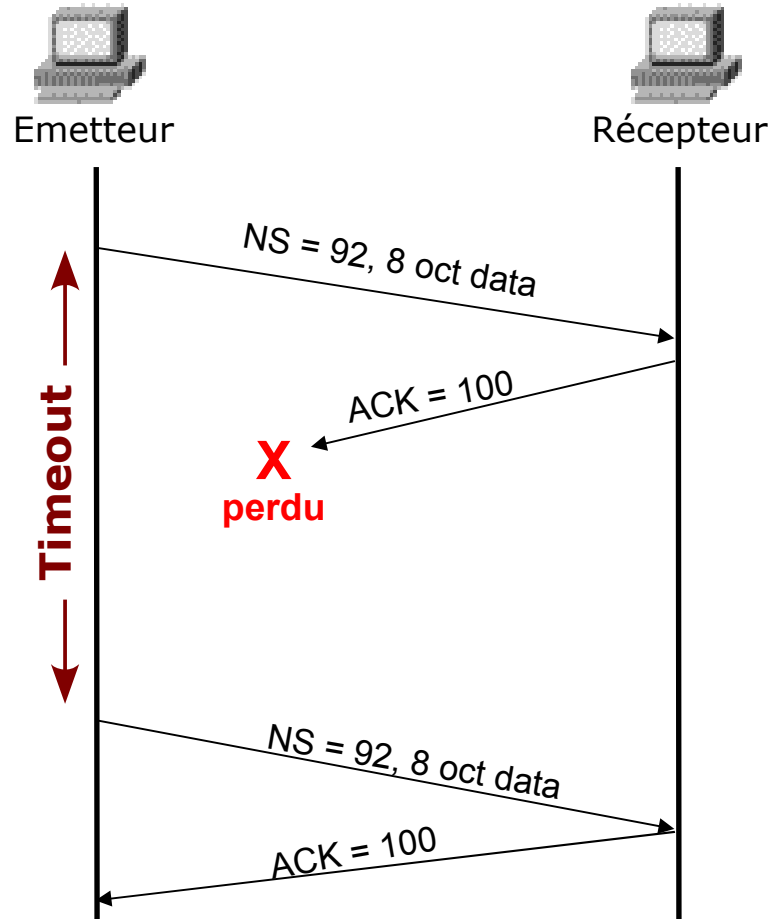
**Notation :**

$(ACK = x) \Leftrightarrow (ACK = 1 \text{ et } NR = x)$

# Fiabilité des transferts

Les pertes de segment sont détectées par absence d'ack positif à expiration d'un temporisateur sur l'émetteur

**Exemple :** Perte d'ACK

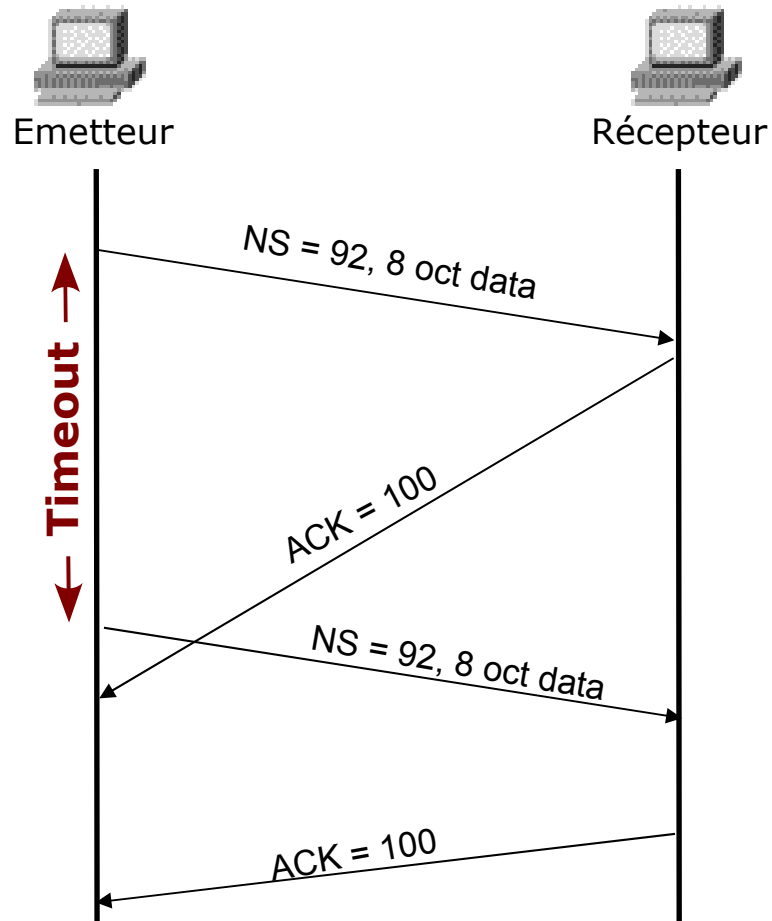




# Fiabilité des transferts

Les pertes de segment sont détectées par absence d'ack positif à expiration d'un temporisateur sur l'émetteur

## Exemple :



# Établissement de connexion

---

## Schéma de connexion

- Ports TCP doivent être ouverts
- Application sur le serveur à l'écoute (en attente d'une connexion)
- Application sur le client fait une requête de connexion

## Les paquets pour ouvrir la connexion

1. **Client** : Demande de connexion

$SYN=1, ACK=0, ISN_{\text{client}} = x$  (séquence initiale du client)

2. **Serveur** : Acceptation de connexion

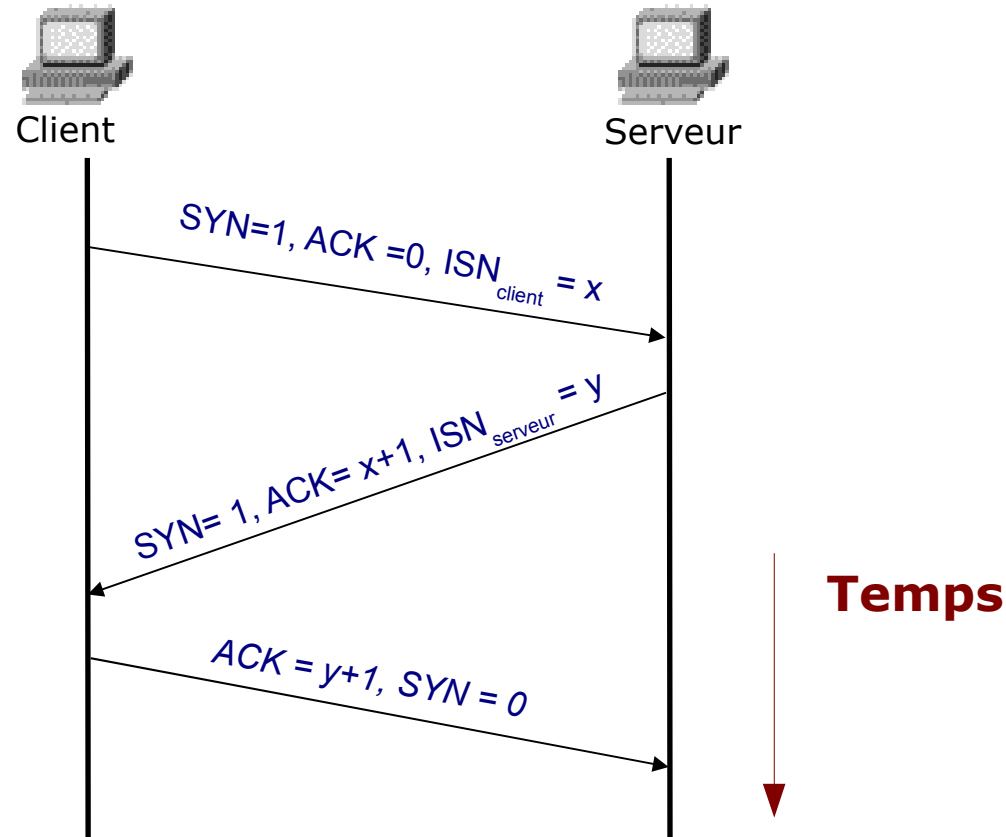
$SYN=1, ACK=x+1, ISN_{\text{serveur}} = y$  (séquence initiale du serveur)

Options possibles (Taille maximale des paquets, convention pour le contrôle de flux sur réseau haut débit... )

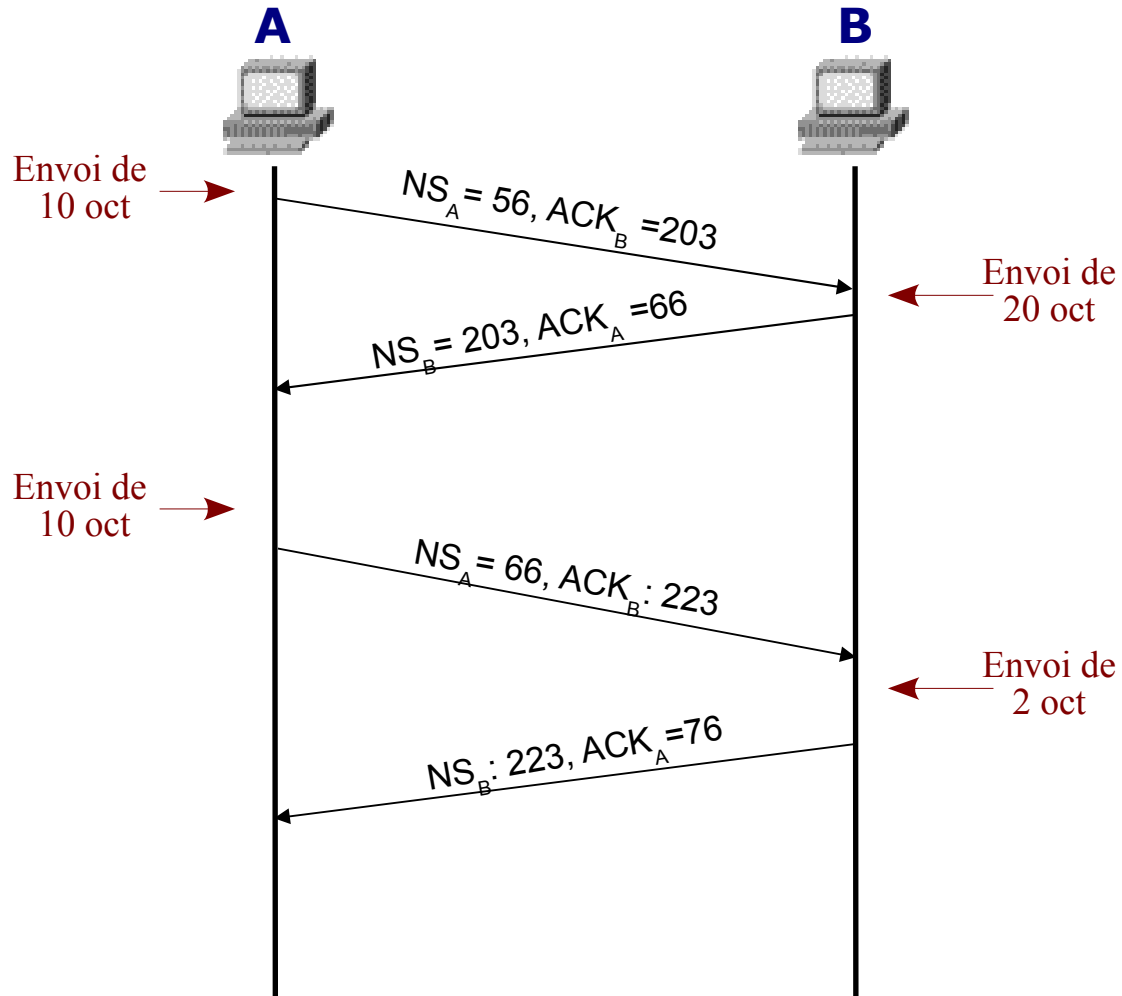
3. **Client** : Accusé de réception  $ACK = y+1, SYN = 0$

# Établissement de connexion

---



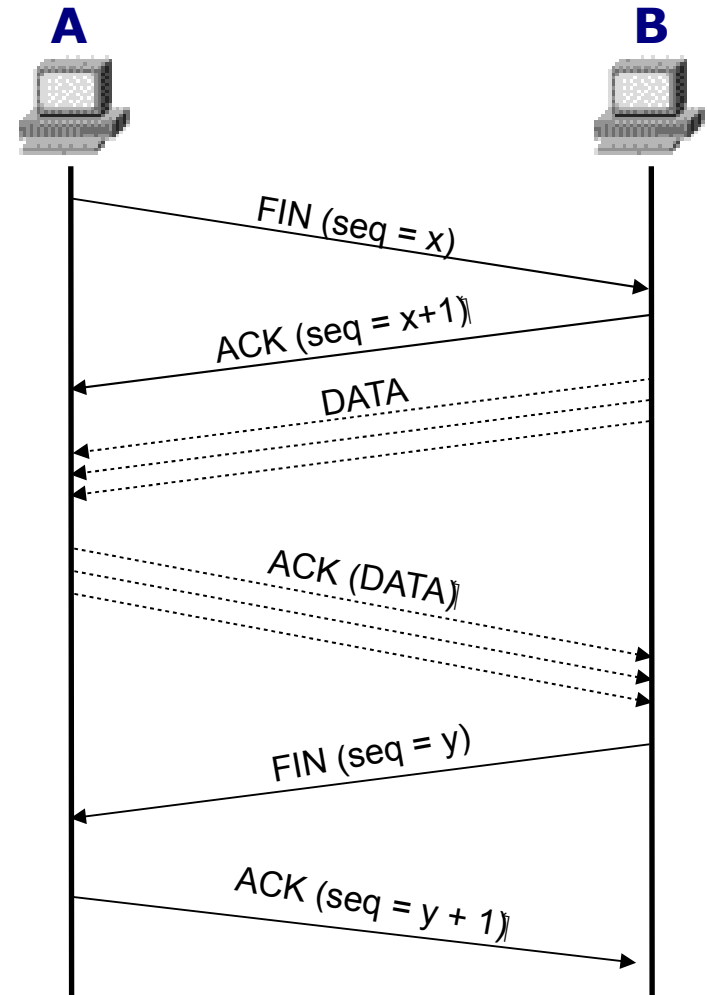
# Transfert de données



# Fin de connexion

## Fermeture négociée

- A** demande la fin de connexion (**FIN=1**)
- B** acquitte la demande de A (**ACK**)
- B** envoie ses données en attente
- A** acquitte les données (**ACK**)
- B** accepte de la fin de connexion (**FIN**)
- A** acquitte de la fin de connexion (**ACK**)



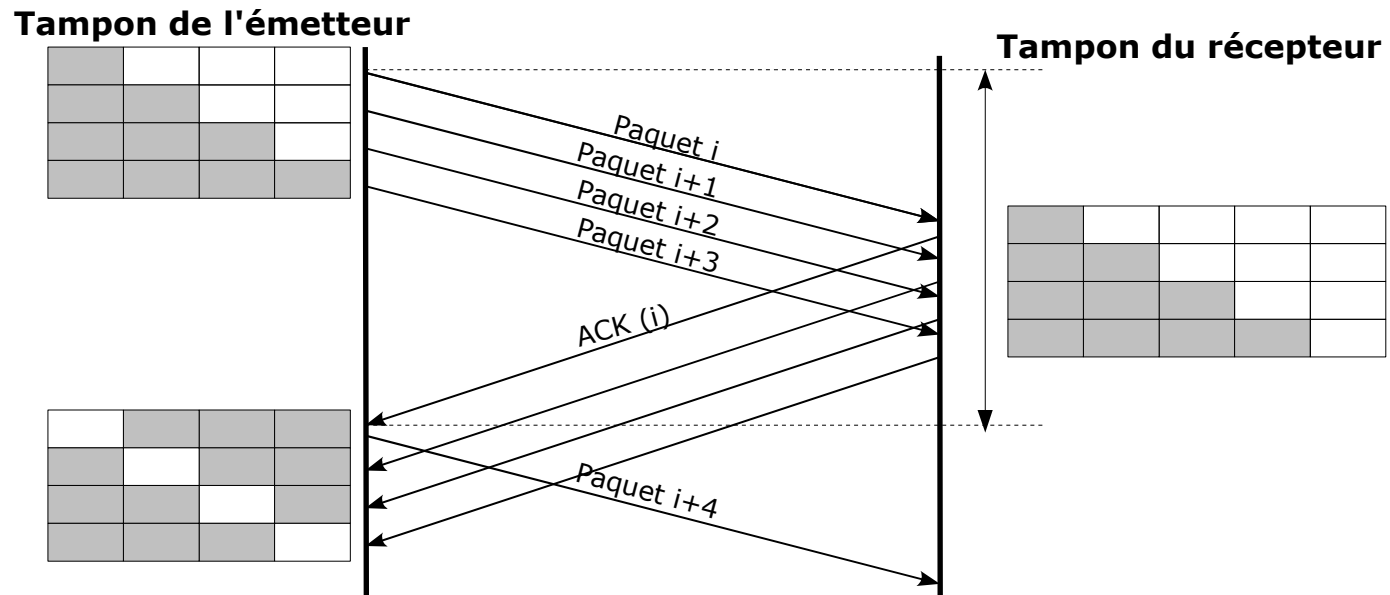
# Utilisation du réseau

- Gaspillage de bande passante !
  - Envoi de données
  - Attente
  - Envoi d'acknowledge
  - Attente
  - ...
- On peut faire mieux !
  - fenêtrage

# Fenêtre glissante

## Principe

- prendre de l'avance sur les réponses --> fenêtre glissante
- fixer un nombre de séquence au bout duquel un accusé de réception est nécessaire



- Le nombre de séquence est stocké dans le champ *fenêtre* de l'en-tête TCP/IP
- La taille de la fenêtre n'est pas fixe
- Le serveur peut demander une augmentation de la taille de la fenêtre

# Fenêtre glissante

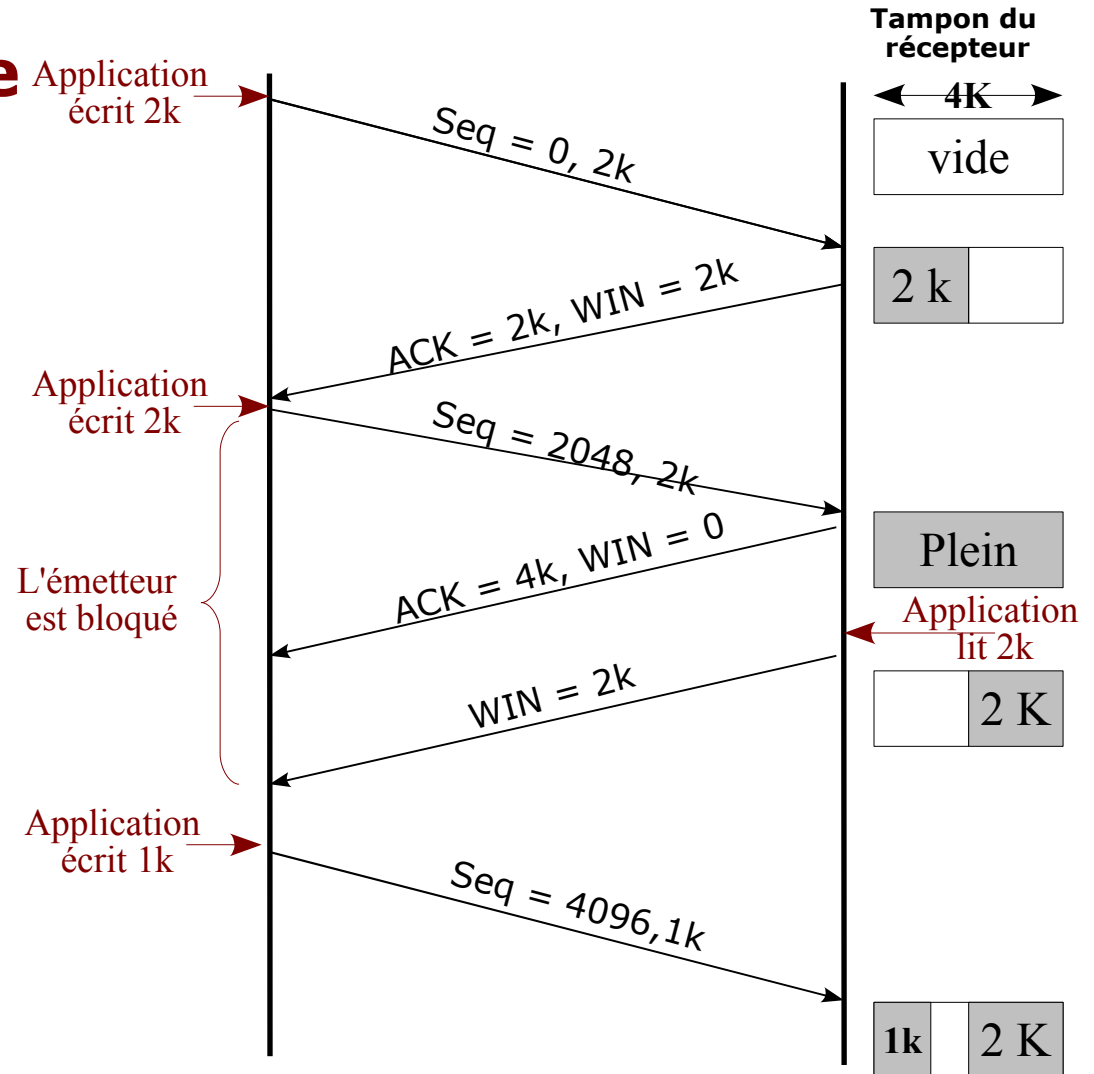
## Gestion de la fenêtrage

### Contrôle de flux TCP :

- l'émetteur ne doit pas envoyer de données si le tampon de réception n'a pas l'espace libre correspondant

### Quand l'émetteur est bloqué, il peut :

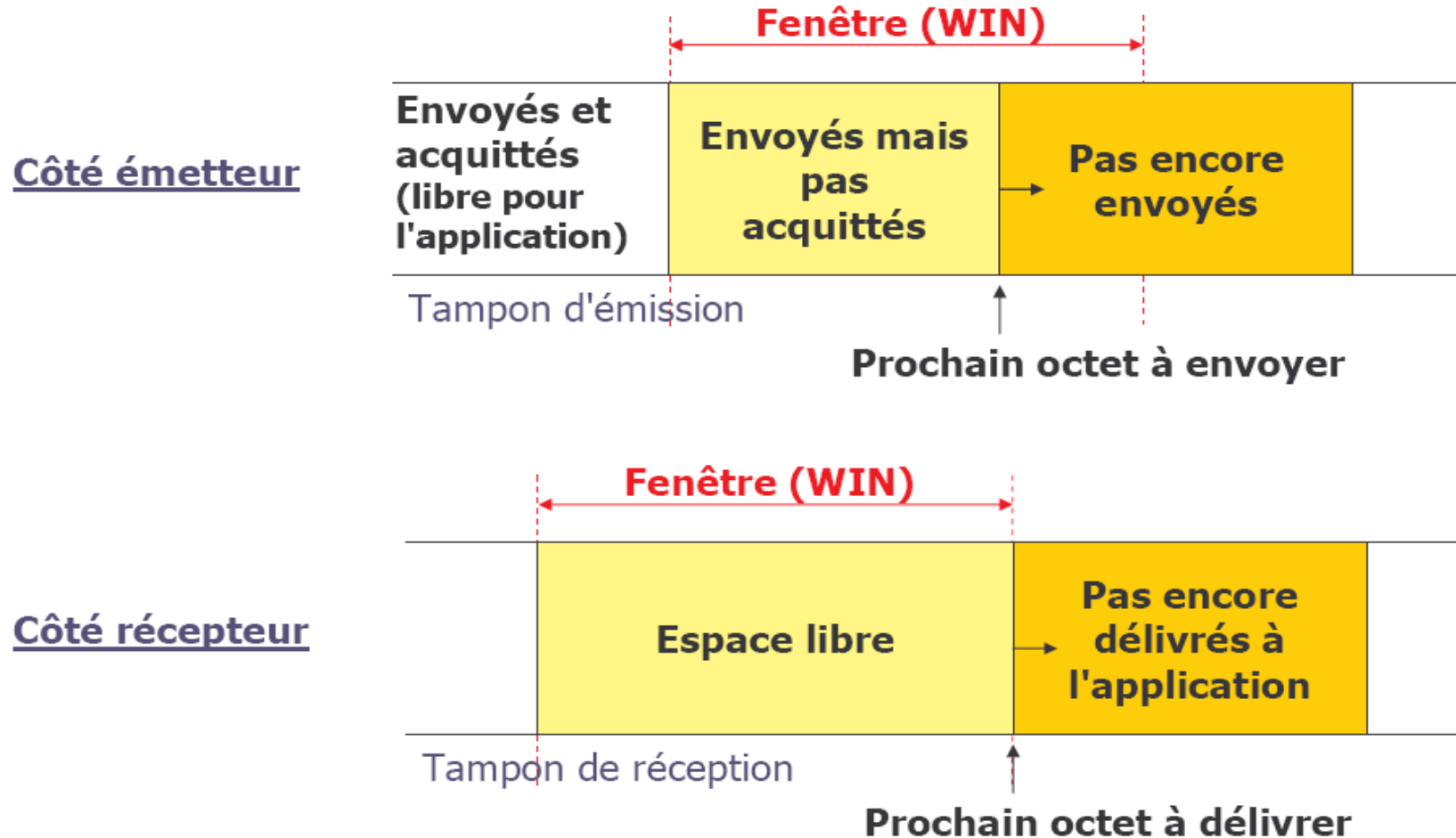
- envoyer des données urgentes (interruption de l'application réceptrice)
- envoyer des segments de 1 octet pour obliger le récepteur à envoyer SEQa et WIN et maintenir l'état actif de la connexion





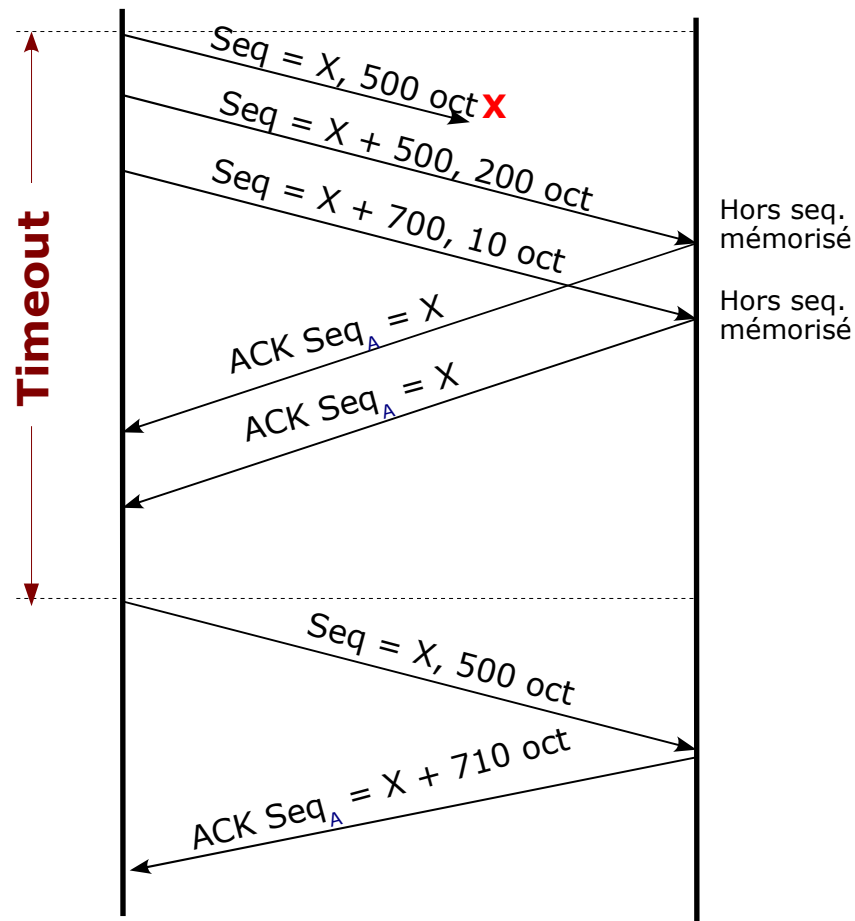
# Fenêtre glissante

## Gestion de la fenêtrage



# Fenêtre glissante

**Exemple :** expiration du timeout



# Organisation

## ■ CM

### Plan

Modèles en couches : OSI & TCP/IP

Couche internet (IP, ICMP, ARP/RARP)

Couche transport (TCP, UDP)

**Couche application (HTTP, DNS, SMTP, FTP...)**

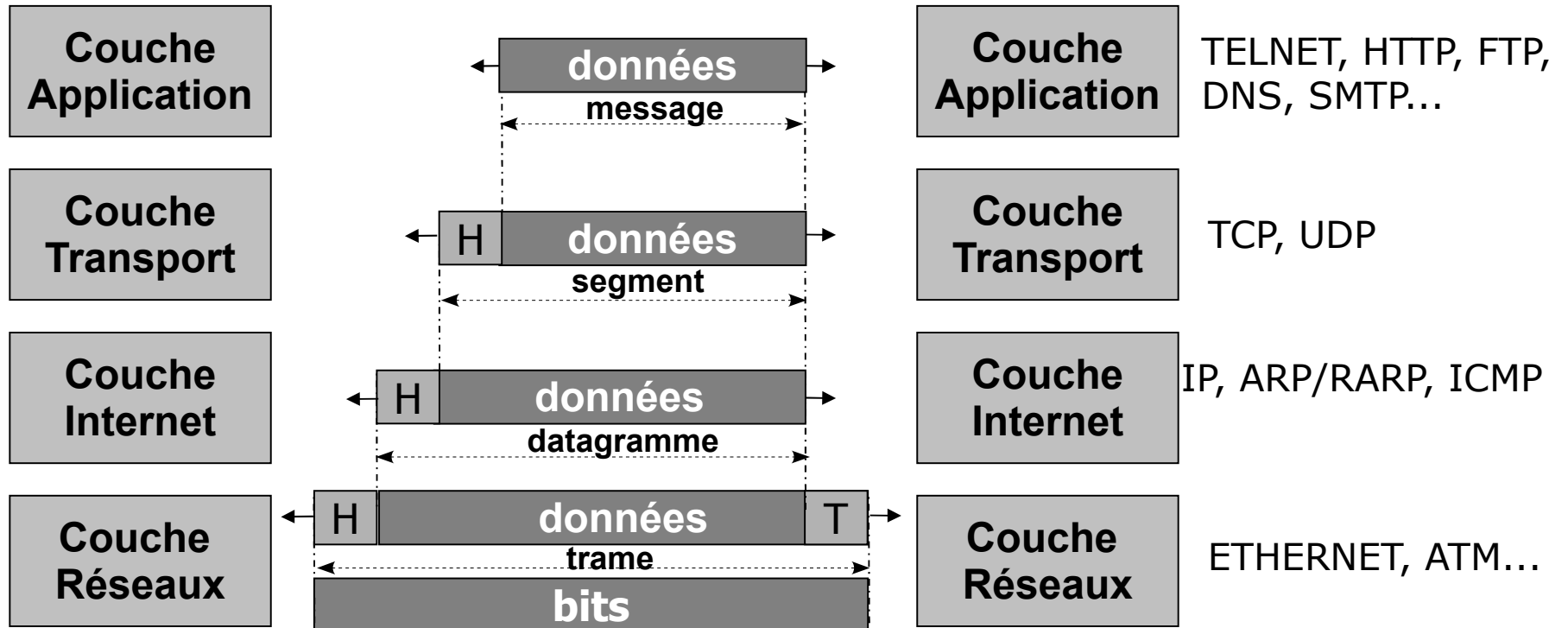
Programmation Réseaux

# **Couche Applications**

Protocoles dns, telnet, ssh, ftp, smtp, pop et http

# Modèle TCP/IP

## Protocoles



# Protocole DNS

---

- Les adresses IP représentent une partie fondamentale de l'Internet
  - Adressage IP
  - Routage IP (le paquet est routé selon sa destination)
  - Les utilisateurs préfèrent des **noms symboliques** que des nombres
  - Comment joindre des sites Web (URL) ?
  - Comment échanger du courrier électronique ?
  - Comment mémoriser les adresses IP de tous les sites Internet à joindre ?
  - Comment gérer les changements d'adresses IP ?

## Solution

Fichier */etc/hosts* pour faire la correspondance noms/adresses

```
# Exemple de fichier /etc/hosts
# This file should contain the addresses and aliases
# for local hosts that share this file.
132.214.1.3 routerquebec
132.212.1.5 routermontreal
```

# Protocole DNS

---

Fichier */etc/hosts* pour faire la correspondance noms/adresses

## Problèmes

- collision des noms
- déploiement à grande échelle
- changements d'adresses et mise à jour

## **DNS - *Domain Name System (résolution de noms de domaines)***

- Base de données hiérarchique et distribuée dans Internet
- Contient des correspondances entre :
  - Nom ordinateur -> adresse IP
  - Passerelle de courrier d'un domaine -> adresse IP
  - Traduction inverse : IP -> nom ordinateur
  - ..., bien plus que seulement nom/adresse IP

# Protocole DNS

---

## Le DNS est constitué de

### Espace de noms de domaines

- base de données qui associe de manière structurée des noms à des @IP

### Serveurs de noms

- hôtes sur lesquels tournent un daemon (port 53)
- Hôtes compétents pour répondre sur une ou plusieurs zones

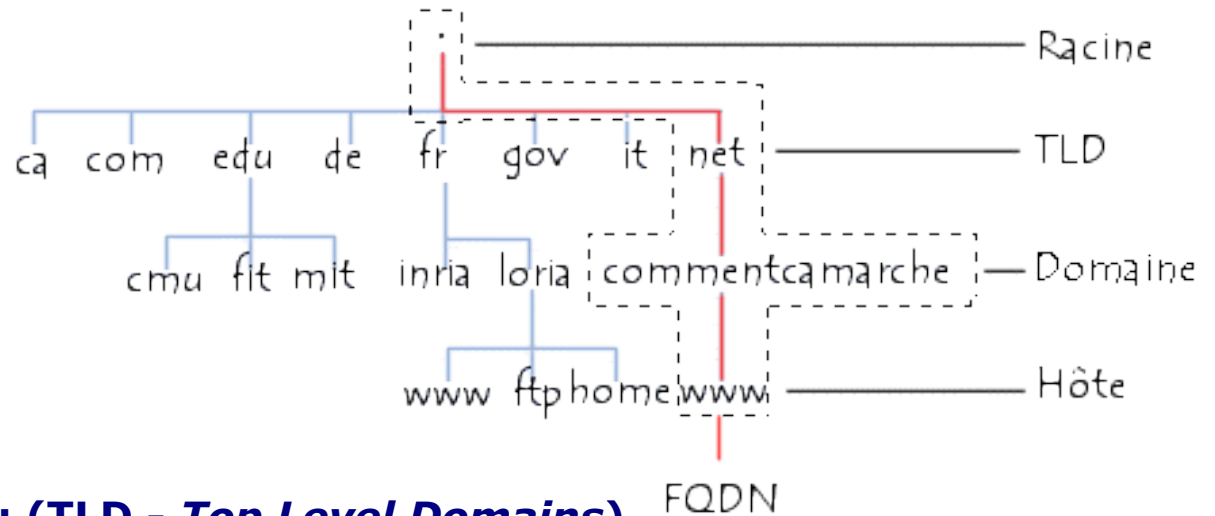
### Resolvers

- fonctions applicatives qui sollicitent la résolution d'une information (nom, adresse) auprès d'un serveur de nom



# Protocole DNS

## Espace de noms



### Domaines de plus haut niveau (TLD - *Top Level Domains*)

- rattachés à un noeud racine représenté par un point

### Nom de domaine

- chaque noeud de l'arbre, ex : univ-lyon2, liris...

### Domaine

- ensemble des étiquettes de noeuds d'une arborescence à l'exception de l'hôte

### Hôte

- extrémité d'une branche, correspond à une machine ou une entité du réseau

### Adresse FQDN (*Fully Qualified Domain Name*) :

- ensemble des étiquettes des noeuds d'une arborescence

### Zone

- ensemble de domaine (concerne un sous arbre du DNS)

# Protocole DNS

---

## Serveurs de noms

- La hiérarchie du DNS est partagée en zone
- L'information contenue dans une zone est implantée dans au moins deux serveurs de nom

## DNS Serveur Primaire

- définit une zone, ensemble de domaines sur lequel le serveur a autorité
- il est mis à jour par l'administrateur système/réseaux

## DNS Serveur Secondaire

- Pour des raisons de fiabilité
- Il recopie périodiquement ses informations d'un serveur primaire
- *Recommandation*: serveur primaire et secondaire sur des sites différents

## Serveur racine

- serveur ayant autorité sur la racine de l'espace de nommage (TLD)
- Actuellement il y a 13 serveurs de ce type

# Protocole DNS

---

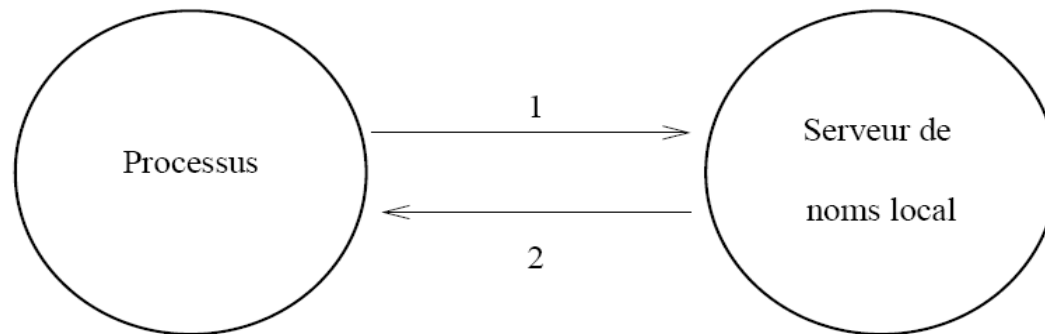
## Résolution de noms de domaine

*Resolver* : un ensemble de fonctions qui font l'interface entre les applications et les serveurs de noms.

## Stratégie de fonctionnement

### Interrogation locale

1. Le processus demande l'@ IP d'une machine. Le " resolver " envoie la demande au serveur local
2. Le serveur local reçoit la demande et répond directement au " resolver "

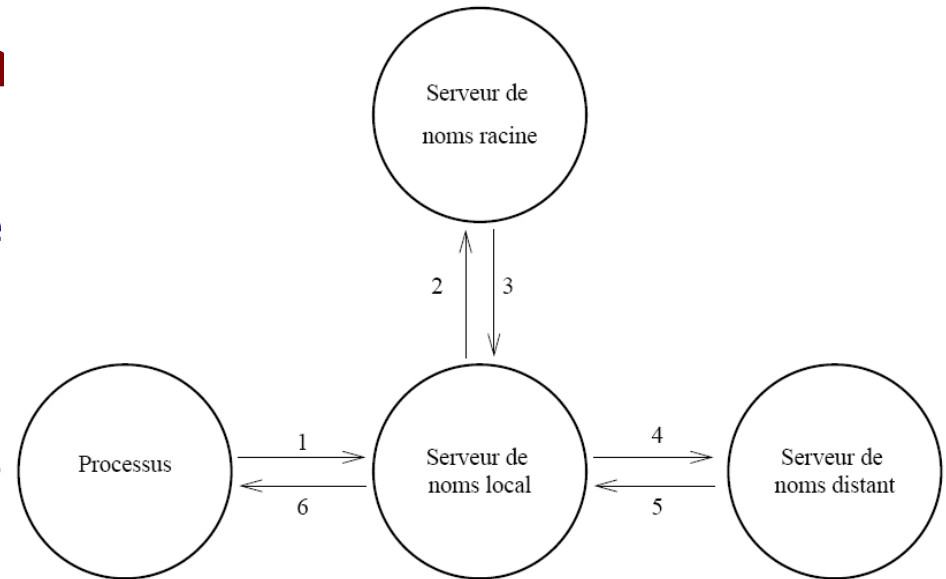


# Protocole DNS

## Résolution de noms de don

### Interrogation distante

1. Le processus demande l'adresse IP d'une machine. Le "resolver" envoie sa requête au serveur local.
2. Le serveur local reçoit la requête, il interroge le serveur racine pour avoir l'adresse d'un serveur qui a l'autorité sur la zone demandée par le processus.



3. Le serveur racine renvoie l'adresse du serveur qui a l'autorité sur la zone.
4. Le serveur local interroge ce nouveau serveur distant.
5. Le serveur distant renvoie l'information demandée au serveur local.
6. Le serveur local retourne la réponse au "resolver".

### Utilisation des caches

Si un processus redemande la même machine distante, le serveur local interroge alors directement le serveur distant sans passer par le serveur racine

# Protocole Telnet

---

- TELeypewrite Network Protocol
- Système de terminal virtuel (Travailler sur un système distant comme sur un système local)
- Connexion TCP
- Utiliser pour ouvrir un émulateur de terminal à distance et pour communiquer avec d'autres services réseau (HTTP, SMTP, FTP...)
- Attention : données transmises en clair !

## Commande telnet

telnet <nom ou adresse IP du serveur> <numéro de port>

Exemple : émulation d'un navigateur web :

```
telnet liris.univ-lyon2.fr 80
```

# Protocole SSH

---

- SSH - Secure SHell
- Port 22
- Remplace Telnet pour l'émulation de terminal
- Comme Telnet, protocole et logiciel
- Utilise un chiffrement asymétrique
- Commande :  
`ssh -l <login> <nom ou adresse IP du serveur>`

Connexion au serveur : `ssh -l ksehaba liris.cnrs.fr`

# Protocole FTP

---

- FTP - *File Transfer Protocol* (protocole de transfert de fichier)
- Port 21
- permet un partage de fichiers entre machines distantes
- permet une indépendance aux systèmes de fichiers des machines clientes et serveur
- permet de transférer des données de manière efficace
- Transmission sur deux canaux :
  - Un canal pour les commandes (canal de contrôle), géré par l'interpréteur de commandes (**PI** : *Protocol Interpreter*)
  - Un canal pour les données, géré par un processus de transfert de données (**DTP** : *Data Transfer Process*)

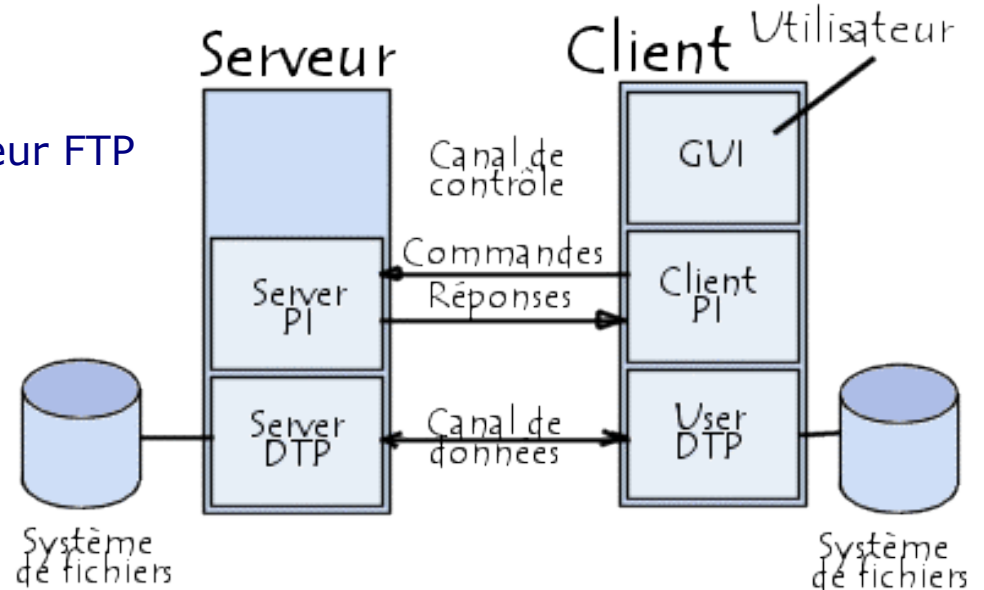
## Notations :

Processus côté client : USER-PI, USER-DTP

Processus côté serveur : SERVER-PI , SERVER-DTP

# Protocole FTP

- **DTP** (*Data Transfer Process*): chargé d'établir la connexion et de gérer le canal de données
- **PI** (*Protocol Interpreter*): interpréteur de protocole permettant de commander le DTP
- **SERVER-PI :**
  - ◆ chargé d'écouter les commandes provenant d'un USER-PI
  - ◆ d'établir la connexion pour le canal de contrôle
  - ◆ de recevoir sur celui-ci les commandes FTP de l'USER-PI
  - ◆ de répondre et de piloter le SERVER-DTP
- **Le USER-PI**
  - ◆ chargé d'établir la connexion avec le serveur FTP
  - ◆ d'envoyer les commandes FTP
  - ◆ de recevoir les réponses du SERVER-PI
  - ◆ de contrôler le USER-DTP si besoin





# Protocole FTP

---

## Types de commandes FTP :

### Commandes de contrôle d'accès

USER, PASS, CWD... QUIT

### Commandes du paramétrage de transfert

PORT, TYPE, PASV...

### Commandes de service FTP

DELE, RMD, MKD, PWD...

## Réponses FTP

Connexion au serveur : <ftp.liris.cnrs.fr> 21

Code à 3 chiffres accompagné d'un texte

**1**yz : Réponse préliminaire positive

**2**yz : Réponse positive de réalisation

**3**yz : Réponse intermédiaire positive

**4**yz : Réponse négative de réalisation

**5**yz : Réponse négative permanente

# Protocole SMTP

---

- SMTP - Simple Mail Transfert Protocol (*Protocole Simple de Transfert de Courrier*)
- Service d'envoi de courrier électronique
- Port 25

## Commandes SMTP

EHLO <machine> :	présentation du client
MAIL FROM: <exp> :	spécifie l'adresse de l'expéditeur
RCPT TO: <dest> :	spécifie l'adresse du destinataire
DATA <message> :	données à envoyer (terminées par .)
QUIT :	termine la session SMTP

# Protocole SMTP

---

## Champs d'en-tête

- To: toto@wanadoo.fr
- Cc: titi@yahoo.fr
- Bcc: tata@gmail.com
- From: pmg@univ-lyon2.fr
- Reply-to: pmg@univ-lyon2.fr
- User-Agent: Mozilla/5.0
- Return-Path:
- Date:
- Subject:
- ...

Envoyez-vous un email en vous connectant via telnet  
au serveur mail de Lyon 2 (smtp.univ-lyon2.fr)  
et en utilisant les commandes du protocole SMTP.

# Protocole POP

---

- POP - *Post Office Protocol (protocole de bureau de poste)*
- permet de récupérer les courriers sur un serveur distant (le serveur POP)
- deux versions du protocole (POP2 port 109 et POP3 port 110)

## Commandes POP2

HELLO : Identification à l'aide de l'adresse IP de l'ordinateur expéditeur  
FOLDER : Nom de la boîte à consulter  
READ : Numéro du message à lire  
RETRIEVE: Numéro du message à récupérer  
SAVE : Numéro du message à sauvegarder  
DELETE : Numéro du message à supprimer  
QUIT : Sortie du serveur POP2

## Commandes POP3

USER : nom de l'utilisateur, doit précéder la commande *PASS*.  
PASS : mot de passe de l'utilisateur  
STAT : Information sur les messages contenus sur le serveur  
RETR : Numéro du message à récupérer  
DELE : Numéro du message à supprimer  
LIST : Numéro du message à afficher  
QUIT : sortie du serveur POP3.

# Protocole HTTP

---

- HTTP : HyperText Transfer Protocol
- Transfert de fichiers (essentiellement au format HTML)
- Port 80

## Version 1.0/1.1 : Modes de communication

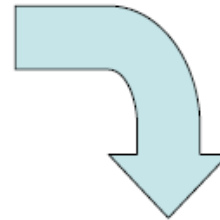
### HTTP 1.0 :

Pour chaque requête :

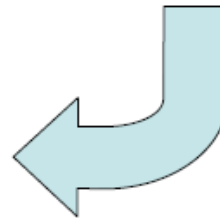
- Réaliser une connexion TCP
- Envoyer la requête
- Recevoir la réponse
- Libérer la connexion

### HTTP 1.1 :

- Réaliser une connexion TCP
- Pour chaque requête :
  - Envoyer la requête
  - Recevoir la réponse
- Libérer la connexion



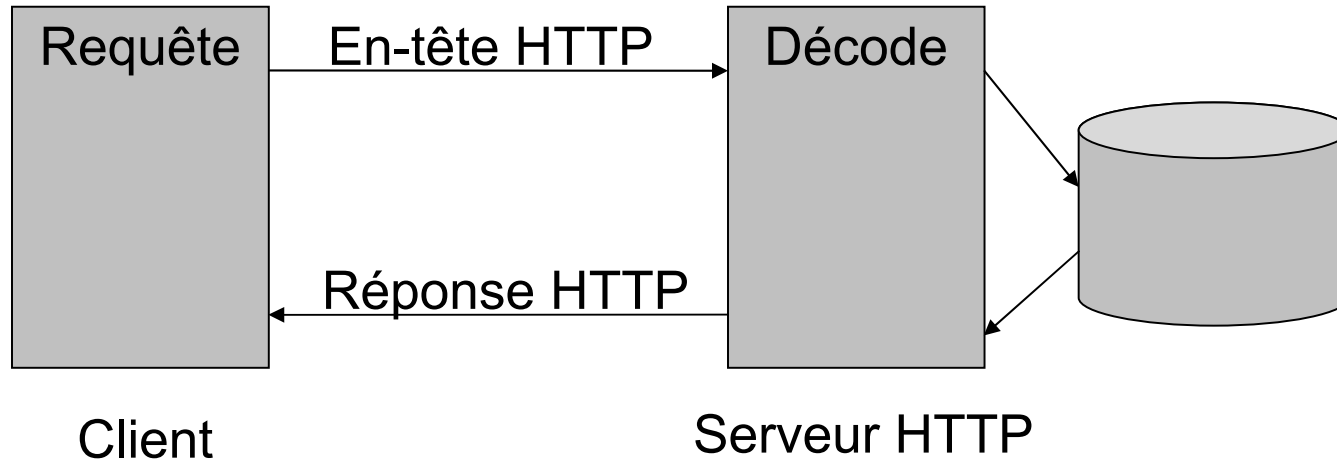
Plusieurs requêtes par site Web  
(par exemples, les fichiers images)  
**-> Problème de performances**



# Protocole HTTP

---

- Communication en deux temps :



## Structure des Requêtes/Réponses

- 1. Ligne de statut :** précise le type de document demandé
- 2. Lignes d'en-tête :** informations supplémentaires sur la requête et/ou le client
- 3. Lignes de Corps**

# Protocole HTTP

---

## Requête HTTP

### 1. Ligne de requête

#### Méthodes

GET : Requête de la ressource située à l'URL spécifiée

HEAD: Requête de l'en-tête de la ressource située à l'URL spécifiée

POST : Envoi de données au programme situé à l'URL spécifiée

DELETE: Suppression de la ressource située à l'URL spécifiée

#### URL - Uniform Resource Locator

**Syntaxe url avec http :** "http ://<host> :<port>/<path>?<searchpath>"

host : nom de machine ou une adresse IP

Port : numéro de port (80 par défaut)

path : sélecteur au sens du protocole http

Searchpath : la chaîne d'interrogation

#### Version du protocole :

Forme *HTTP-Numéro de version*. Par exemple HTTP/1.1

# Protocole HTTP

---

## Requête HTTP

### 2. Champs d'en-tête (facultatif)

- Informations supplémentaire sur la requête et/ou le navigateur
- *Syntaxe* **Type en-entête : valeur en-tête**

#### Exemple d'en-tête

User-Agent : informations sur le client (nom et la version du navigateur, système d'exploitation...)

Accept-Language: Langage attendu par le navigateur

### 3. Corps

- Lignes optionnelles séparées des lignes précédentes par une ligne vide
- Permettent l'envoi des données au serveur (commande POST)



# Protocole HTTP

---

## Requête HTTP

### Exemples:

Ligne de requête { GET http://www.google.com HTTP/1.0

Champs d'en-tête { Accept : text/html  
If-Modified-Since : Saturday, 15-January-2000 14:37:11 GMT  
User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

Renvoie le contenu du document par défaut

```
$ telnet clx.anet.fr 80
```

```
GET /  
\n  
\n
```

Renvoie le document mentionné dans le chemin complet

```
$ telnet 10function.kicks-ass.org 80
```

```
GET /books/ruhacker.txt
```

```
\n
```

```
\n
```

```
$ telnet liris.cnrs.fr 80
```

```
GET /ksehaba/Reseaux/Test.html
```

```
\n
```

```
\n
```

# Protocole HTTP

---

## Réponse HTTP

### 1. Ligne de statut

- **Version du protocole utilisé**

*HTTP-Numéro de version*

- **Code de statut :**

valeur numérique (xxx) qui décrit le statut de la réponse

**1xx:** N'est pas utilisé, "Futur Use"

**2xx:** Succès, l'action exécutée correctement

**3xx:** Redirection, reprendre l'interrogation avec une autre formulation

**4xx:** Erreur coté client, erreur de syntaxe ou ne peut être acceptée

**5xx:** Erreur coté serveur, erreur interne due à l'OS

- **Signification du code**

### 2. En-tête

Informations supplémentaires sur la réponse et/ou le serveur

### 3. Corps

Le document demandé

# Protocole HTTP

## Réponse HTTP

### Exemple :

```
$ telnet localhost 80
```

```
GET / HTTP/1.0
```

```
\n
```

```
\n
```

```
C:\WINDOWS\system32\cmd.exe

HTTP/1.1 200 OK
Date: Thu, 18 Oct 2007 14:05:03 GMT
Server: Apache/1.3.14 (Win32)
Last-Modified: Sat, 16 Feb 2002 07:15:38 GMT
ETag: "0-629-3c6e071a"
Accept-Ranges: bytes
Content-Length: 1577
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>PHPGeek.com Presents PHPTriad</TITLE>
</HEAD>
<BODY>
<h2>Welcome</h2>
Congratulations on choosing PHPTriad for your Windows PHP needs. We're always str
iving toward making the package better, so be sure to check in at PHPGeek.com t
o be sure you received the latest version and additional extensions.<p>
While PHPTriad is free both financially and in the sense of freedom, it takes mo
ney and time to run this project. If this product is useful to you consider cont
ributing to the ongoing development of this project. There are several ways to g
ive back.
<ul>
<li>Giving money directly through my "Give Something Back" box at <a href="http:
//www.phpgeek.com">PHPGeek.com</a>.
<li>Buying me something off of my Amazon.com wish list linked in that same box a
t PHPGeek.com.
<li>Buying T-shirts and other products from the shirt gallery linked at PHPGeek.
com.
<li>Buying products and services from the advertisers and endorsements at PHPGe
ek.com
</ul>
<h2>What's New in This Version?</h2>
PHPTriad 2.2 contains the following changes:
<ul>
<li>Updated packages for Apache(1.3.23), MySQL(3.23.48) and PHP(4.1.1).
<li>Updated PHPMyAdmin.
<li>Added the PHPTriad Control Panel (written in PHP) to manage PHPTriad.
<li>Added Backup module to
</ul>
<p>
This file can be found at c:\apache\htdocs\index.html and is the default page fo
r your site. Edit that file or replace it in order to change your site. To contr
ol your site, use the PHPTriad Control Panel
</BODY>
</HTML>
```

# Organisation

## ■ CM

### Plan

Modèles en couches : OSI & TCP/IP

Couche internet (IP, ICMP, ARP/RARP)

Couche transport (TCP, UDP)

Couche application (HTTP, DNS, SMTP, FTP...)

**Programmation Réseaux**

# **Programmation Réseau**

**Applications Client/serveur**

**Mode connecté/mode non connecté**

Programmation en C

Programmation en java

# Modèle client-serveur définition

---

## Application client/serveur

- application qui fait appel à des services distants à travers d'un échange de messages (les requêtes) plutôt que par un partage de données (mémoire ou fichiers)
- **Serveur**
  - programme offrant un service sur un réseau (par extension, machine offrant un service)
- **Client**
  - programme qui émet des requêtes (ou demandes de service). Il est toujours l'initiateur du dialogue

# Exemple de client/serveur

---

- Un serveur de fichiers
- Des clients qui demandent des fichiers
- Comment gérer la concurrence ?
  - ◆ un processus serveur => un client servi
  - ◆ plusieurs clients => il va falloir faire des fork() !!!
- Quel protocole utiliser ?
  - ◆ le client envoie le nom du fichier
  - ◆ le serveur renvoie la taille puis les données
  - ◆ comment gère-t-on les erreurs ?

# Mode connecté/nom connecté

---

## ■ Mode connecté (TCP) :

- ◆ problèmes de communications gérés automatiquement,
- ◆ primitives simples d'émission et de réception,
- ◆ gestion de la connexion coûteuse,
- ◆ pas de délimitation des messages dans le tampon.

## ■ Mode non connecté (UDP) :

- ◆ consomme moins de ressources systèmes,
- ◆ permet la diffusion,
- ◆ gestion de toutes les erreurs à la main : il faut réécrire la couche transport !!!



# Programmation Réseau

Applications Client/serveur

Mode connecté/mode non connecté

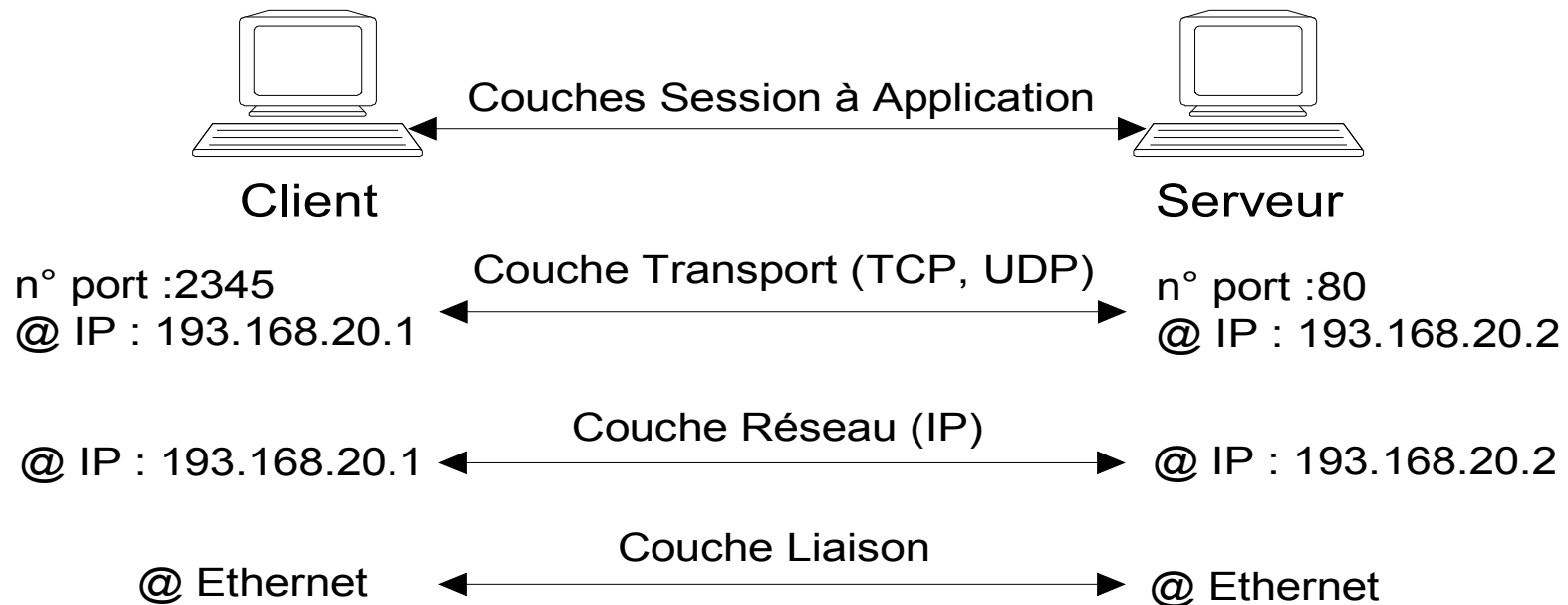
**Programmation en C**

Programmation en java

# Les sockets

---

- Interface d'accès au réseau
- développé dans Unix BSD (Berkeley Software Distribution)
- n° port, @ IP, protocole (TCP, UDP, ...)

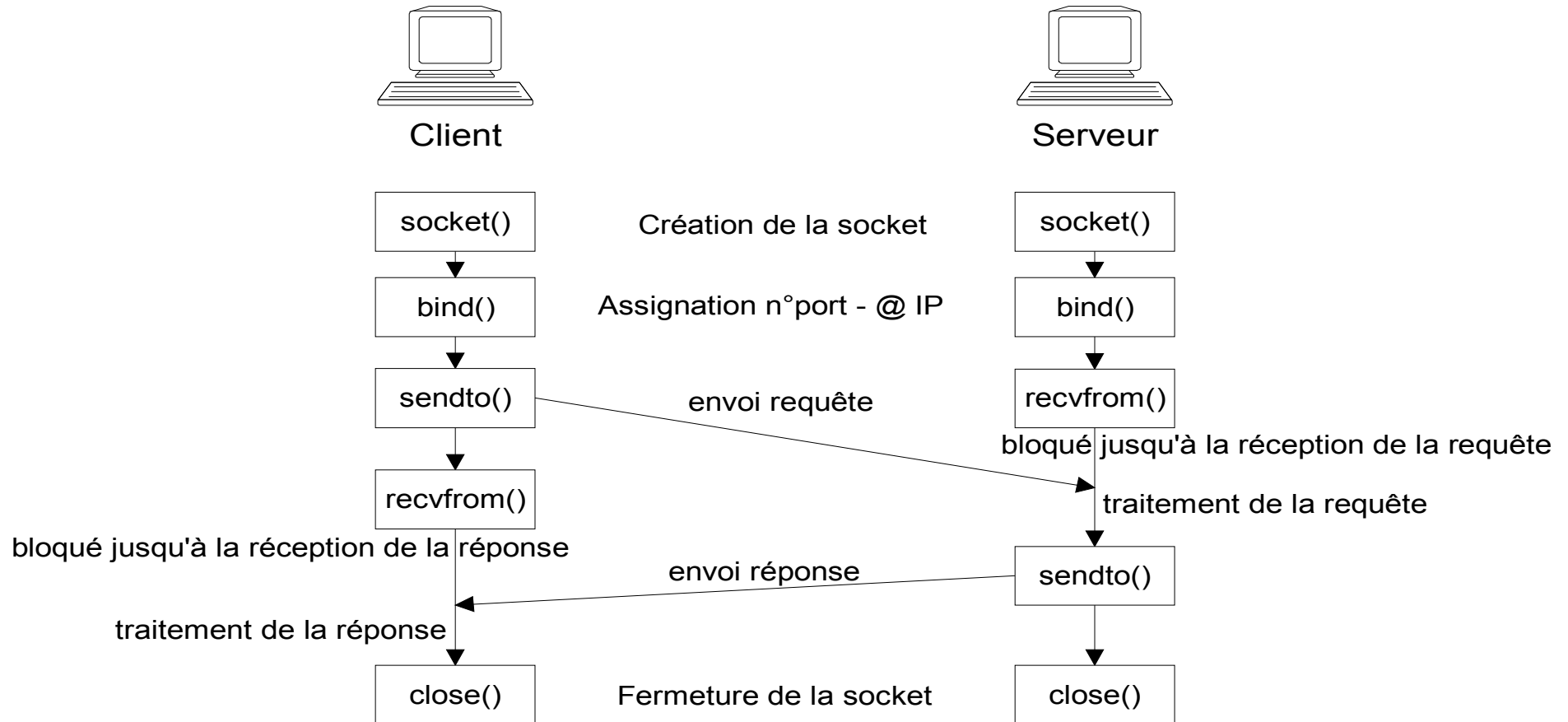


# L'API socket

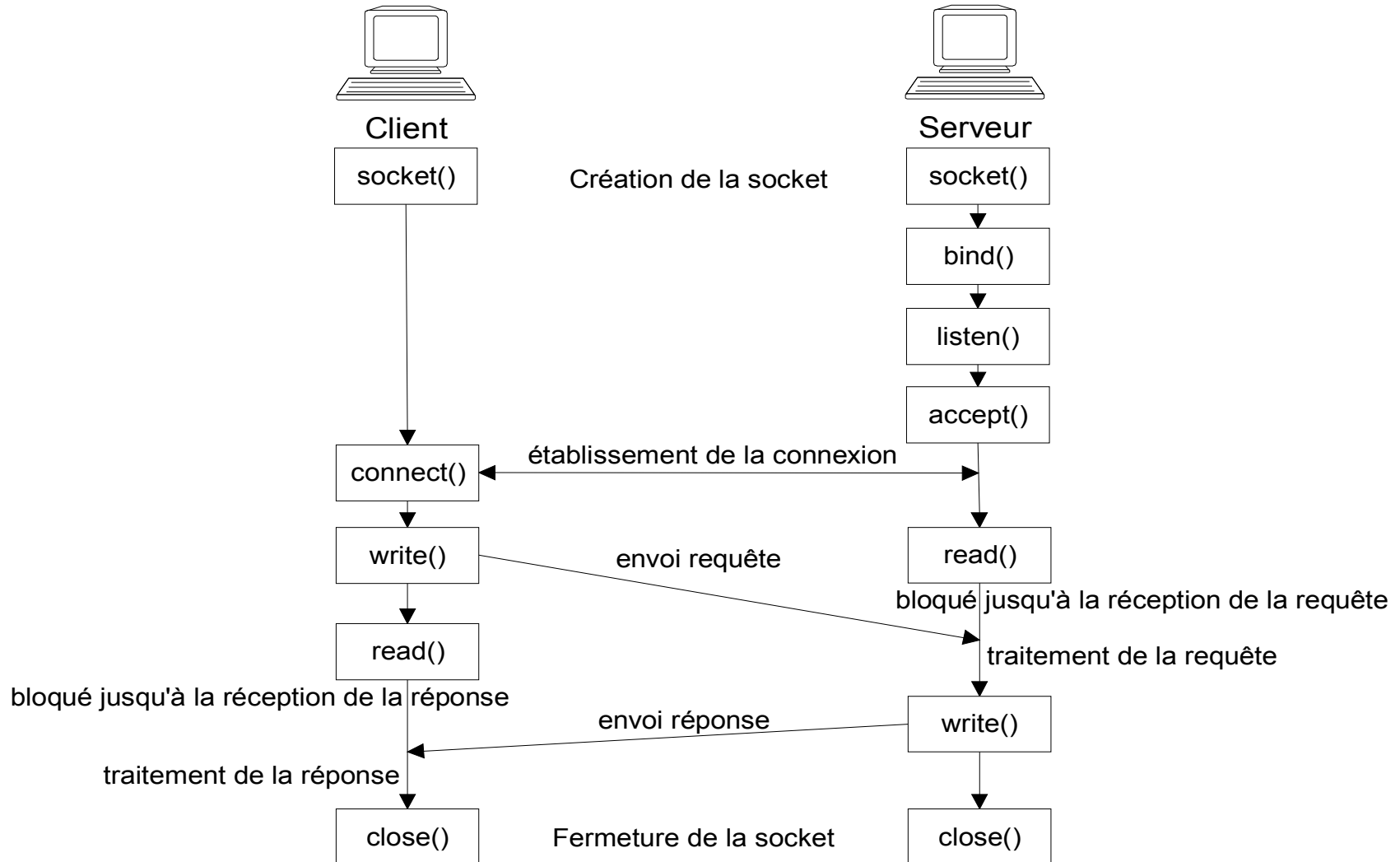
---

- Création de socket : `socket(family, type, protocol)`
- Ouverture de dialogue :
  - ◆ client : `connect(...)`
  - ◆ serveur : `bind(..), listen(...), accept(...)`
- Transfert de données :
  - ◆ mode connecté : `read(...), write(...), send(...), recv(...)`
  - ◆ mode non connecté : `sendto(...), recvfrom(...), sendmsg(...), recvmsg(...)`
- Clôture du dialogue :
  - ◆ `close(...), shutdown(...)`

# Client/Serveur en mode non connecté



# Client/Serveur en mode connecté



# La primitive socket()

---

**int socket(int family, int type, int protocol)**

■ **family :**

- ◆ AF\_INET : pour des communications Internet
- ◆ AF\_UNIX : pour des communications locales

■ **type ou mode de fonctionnement :**

- ◆ SOCK\_STREAM : mode connecté (TCP)
- ◆ SOCK\_DGRAM : mode déconnecté (UDP)
- ◆ SOCK\_RAW : accès direct aux couches basses (IP)

■ **protocol :**

- ◆ IPPROTO\_UDP (protocole UDP avec SOCK\_DGRAM)
- ◆ IPPROTO\_TCP (protocole TCP avec SOCK\_STREAM)
- ◆ IPPROTO\_ICMP (protocole ICMP avec SOCK\_RAW)
- ◆ IPPROTO\_RAW (accès direct IP avec SOCK\_RAW)

# La primitive connect()

---

**int connect(int sock\_desc, struct sockaddr \* @\_serveur, int lg\_@)**

- **sock\_desc** : descripteur de socket retourné par socket()
- **@\_serveur** : adresse IP et n° de port du serveur distant
- **Exemple de client :**

```
int sd;
struct sockaddr_in serveur; // @IP, n° port, mode
struct hostent  remote_host; // nom et @IP

sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
serveur.sin_family = AF_INET;
serveur.sin_port = htons(13);
remote_host = gethostbyname("brassens.upmf-grenoble.fr");
bcopy(remote_host->h_addr, (char *)&serveur.sin_addr,
      remote_host->hlength); // Recopie de l'adresse
connect(sd, (struct sockaddr *)&serveur, sizeof(serveur));
```

# La primitive bind()

---

**int bind(int sock\_desc, struct sockaddr \*my\_@, int lg\_@)**

- **sock\_desc** : descripteur de socket retourné par socket()
- **my\_@** : adresse IP et n° de port auxquels le serveur veut répondre
- **Exemple de serveur :**

```
int sd;
struct sockaddr_in serveur; // @IP, n° port, mode

sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
serveur.sin_family = AF_INET;
serveur.sin_port = 0; // Laisse le système choisir un port
serveur.sin_addr.s_addr = INADDR_ANY;
// Autorise des connexions de n'importe où
bind(sd, (struct sockaddr *)&serveur, sizeof(serveur));
```



# La primitive listen()

---

## **int listen(int sock\_desc, int backlog)**

- **sock\_desc** : descripteur de socket retourné par socket()
- **backlog** : nombre maximum de connexions en attente d'être acceptées
- **Exemple** de serveur :

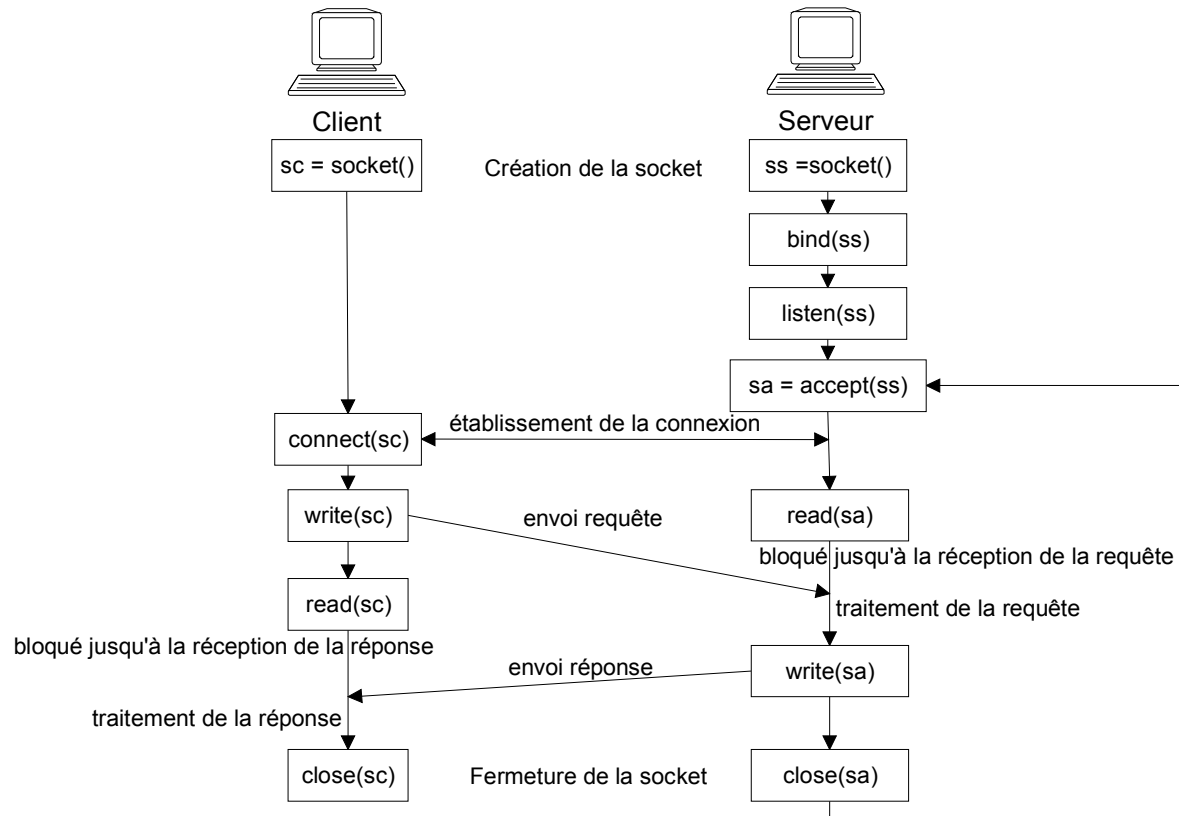
```
int sd;
struct sockaddr_in serveur; // @IP, n° port, mode

sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
serveur.sin_family = AF_INET;
serveur.sin_port = 0; // Laisse le système choisir un port
serveur.sin_addr.s_addr = INADDR_ANY;
// Autorise des connexions de n'importe où
bind(sd, (struct sockaddr *)&serveur, sizeof(serveur));
listen(sd, 5);
```

# La primitive accept()

**int accept(int sock\_desc, struct sockaddr \*client, int lg\_@)**

- **sock\_desc** : descripteur de socket retourné par socket()
- **client** : identité du client demandant la connexion
- accept renvoie le descripteur de la nouvelle socket créée



# Les primitives d'envoi/réception

---

- `int write(int sock_desc, char *tampon, int lg_tampon);`
  - `int read(int sock_desc, char *tampon, int lg_tampon);`
  - `int send(int sock_desc, char *tampon, int lg_tampon, int drap);`
  - `int recv(int sock_desc, char *tampon, int lg_tampon, int drap);`
  - `int sendto(int sock_desc, char *tampon, int lg_tampon, int drap, struct sockaddr *to, int lg_to);`
  - `int recvfrom(int sock_desc, char *tampon, int lg_tampon, int drap, struct sockaddr *from, int lg_from);`
- `drap` : options de contrôle de la transmission (consulter le man)

# Programmation Réseau

Applications Client/serveur

Mode connecté/mode non connecté

Programmation en C

**Programmation en java**

# Canaux bidirectionnels

- Communication entre deux applications distantes par des canaux de bidirectionnels (sockets).
- deux types de sockets en Java :
  - Stream sockets (TCP) : communication en mode connecté, caractérisées par le caractère continu du flux de données (ordre correct - données non corrompues).
  - Datagram sockets (UDP) : communication en mode non-connecté : plus grande rapidité, mais fiabilité moindre (perte, duplication ou désordre possibles).

# C l a s s e s d e j a v a . n e t

- ContentHandler                   : *base pour gérer contenus M I M E*
- Datagram Packet                }
- Datagram Socket                }        U D P
- InetAddress                    : *gestion des adresses*
- ServerSocket                    }
- Socket                            }        T C P
- SocketImpl
- URL
- URLConnection                 }
- URLEncoder                     }        U R L
- URLStreamHandler

# InetAddress

```
public final class InetAddress {

    public String getHostName ()
    public byte[] getAddress ()
    public String getHostAddress ()
    public int hashCode ()
    public boolean equals (Object obj)
    public String toString ()
    public static synchronized InetAddress
        getByName (String host) throws UnknownHostException
    public static synchronized InetAddress
        getAllByName (String host)[] throws UnknownHostException
    public static InetAddress
        getLocalHost () throws UnknownHostException
}
```

# ServerSocket

```
public final class ServerSocket {
    public ServerSocket(int port)
                                throws IOException
    public ServerSocket(int port, int backlog)
                                throws IOException
    public InetAddress getInetAddress()
    public int getLocalPort()
    public Socket accept()
                                throws IOException
    public void close()
                                throws IOException
    public String toString()
    public static synchronized void
        setSocketFactory(SocketImplFactory fac)
                                throws IOException
}
```



# Socket

```
public final class Socket {
    public Socket(String host, int port)
        throws UnknownHostException, IOException
    public Socket(String host, int port, boolean stream)
        throws IOException
    public Socket(InetAddress address, int port)
        throws IOException
    public Socket(InetAddress address, int port, boolean stream)
        throws IOException

    public InetAddress getInetAddress()
    public int getPort()
    public int getLocalPort()
    public InputStream getInputStream() throws IOException
    public OutputStream getOutputStream() throws IOException
    public synchronized void close() throws IOException
    public String toString()
    public static synchronized void
        setSocketImplFactory(SocketImplFactory fac)
        throws IOException
}
```

# Utilisation des sockets TCP / IP

Client

Serveur

1- Création d'un client et tentative de connexion

Construction d'un objet de type **Socket** (paramètres : hôte et n° de port)

2- Echange  
**OutputStream**  
**InputStream**

3- Fermeture (**close()**)

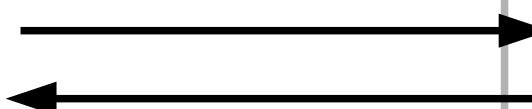
1- Enregistrement du service Construction d'un objet de type **ServerSocket** (paramètres : n° de port et nombre connexions)

2- Attente d'une demande de connexion du client

**accept()** retourne, lorsqu'une connexion est acceptée, un objet de type **Socket**

3- Echange  
**InputStream**  
**OutputStream**

4- Fermeture (**close()**)



# Exemple de code : Serveur (1)

```
import java.net.*;
import java.io.*;

public class Serveur {
    ServerSocket srvk;
    public Serveur (int port) throws IOException {
        srvk = new ServerSocket (port);
    }
    public void go () {
        while (true) {
            try {
                Socket sck = srvk.accept ();
                OutputStream os = sck.getOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(os);
                String msg = "REPONSE DE SERVEUR";
                Object obj = (Object) msg;
                oos.writeObject(obj);
                sck.close ();
            }
            catch (IOException e) {}
        }
    }
}
```

## Exemple de code : Serveur (2)

```
public static void main (String args [])
                        throws IOException {
    Serveur s = new Serveur (Integer.parseInt (args[0]));
    s.go ();
}
}
```

# Exemple de code : Client (1)

```
import java.net.*;
import java.io.*;

public class Client {
    Socket so;
    public Client (String host, int port) throws IOException {
        so = new Socket (host, port);
    }

    public void go () {
        try {
            InputStream is = so.getInputStream ();
            ObjectInputStream ois = new ObjectInputStream (is);
            Object obj = ois.readObject();
            String msg = (String) obj;
            System.out.println(msg);
        } catch (IOException e) {}
        catch (ClassNotFoundException e){}
    }

    public void stop () {
        try { so.close ();}
        catch (IOException e) { }
    }
}
```

# Exemple de code : Client (2)

```
public static void main (String args [])  
    throws IOException {  
    Client c = new Client (args[0], Integer.parseInt (args[1]));  
    c.go ();  
    c.stop ();  
    }  
}
```

Bilan

# **Modèles en couches**

Notion de protocole

Modèle de référence : OSI de ISO

Modèle utilisé sur Internet : TCP/IP

# **Délimitation des données**

Notion de fanion

Notion de transparence



# Couche Internet (1)

## Protocole IP

Gestion de la fragmentation

ARP, ICMP

IPv6

# **Couche Internet (2)**

## **Routage**

Routage statique (Commandes)

Routage dynamique (Protocoles RIP, OSPF, RGP, BGP)

# Couche Transport

Services de la couche

Protocole UDP

Protocole TCP

# **Couche Applications**

Protocoles dns, telnet, ssh, ftp, smtp, pop et http

# **Programmation Réseau**

Applications Client/serveur

Mode connecté/mode non connecté

Programmation en C

Programmation en java