# Architecture de l'Internet : Technologie de l'Internet

## – M2103 –

Camille Coti
camille.coti@lipn.univ-paris13.fr

Département R&T, IUT de Villetaneuse, Université de Paris XIII

# Plan du cours

# Plan du cours

# Plan du cours

# Contexte d'Internet

## Étymologie

Internet = **Inter** connection **Net** works

- Souvent appelé "le réseau des réseaux"
- Interconnecte des réseaux **hétérogènes** !

Exemple :

- faire communiquer des réseaux ATM, X25, Frame Relay, IP...
- ... utilisant des ondes radio, de la fibre optique, des câbles RJ45...

## Conséquences

Hétérogénéité : différentes technologies réseaux

- Caractère indispensable des **normes et standards**
- **Protocoles** de communication

# Modèle OSI

## Modèle en 7 couches

Normalisé par l'ISO (International Standards Organisation)

Principes de base du découpage en couches du modèle OSI :

- une couche doit être créée lorsqu'un nouveau **niveau d'abstraction** est nécessaire,
- chaque couche a des **fonctions bien définies** ,
- les fonctions de chaque couche doivent être choisies dans l' **objectif de la normalisation internationale des protocoles** ,
- les frontières entre couches doivent être choisies de manière à **minimiser le flux d'information aux interfaces** ,
- le nombre de couches doit être tel qu'il n'y ait **pas cohabitation de fonctions très différentes** au sein d'une même couche et que l'architecture ne soit pas trop difficile à maîtriser.

Une bonne référence : `http://www.frameip.com/osi`

# Modèle OSI

| | |
|---|---|
| 7. | Application |
| 6. | Présentation |
| 5. | Session |
| 4. | Transport |
| 3. | Réseau |
| 2. | Liaison de données |
| 1. | Physique |

Couches 1 à 4 : acheminement des informations

- Couches 1 et 2 : couches matérielles
- Couches 1 à 3 : couches entre machines voisines

Couches 5 à 7 : traitement de l'information par les hôtes

- Couches 4 à 7 : entre hôtes potentiellement distants

# Exemples

Source : Wikipedia

- **7 Couche application**
  - Gopher • SSH • FTP • NNTP • DNS • SNMP • XMPP • Telnet • SMTP • POP3 • IMAP • IRC • RTP• WebDAV • SIMPLE • HTTP • Modbus • CLNP • SIP • DHCP• CANOpen• TCAP• RTSP • BGP
- **6 Couche de présentation**
  - SMB • ASCII • Videotex • Unicode • TDI • ASN.1 • XDR • UUCP • NCP • AFP • SSP
- **5 Couche de session**
  - • AppleTalk • NetBios
- **4 Couche de transport**
  - TCP • UDP • SCTP • SPX • DCCP
- **3 Couche de réseau**
  - IP • NetBEUI • IPv4 • IPv6 • DHCP (en tant que service) • IPX • ICMP • IGMP • WDS • RIP10 • OSPF • IS-IS • EIGRP
- **2 Couche de liaison de données**
  - Ethernet • CSMA/CD • CSMA/CA • Anneau à jeton • LocalTalk • FDDI • X.21 • X.25 • Frame Relay • BitNet • CAN • PPP • PPPoE • HDLC• ATM • ARP • MPLS "2,5"
- **1 Couche physique**
  - Codage NRZ • Codage Manchester • Codage Miller • RS-232 • RS-449 • V.21-V.23 • V.42-V.90 • Câble coaxial • 10BASE2 • 10BASE5 • Paire torsadée • 10BASE-T • 100BASE-TX • 1000BASE-T • RNIS • PDH • SDH • T-carrier • EIA-422 • EIA-485 • SONET • ADSL • SDSL • VDSL • DSSS • FHSS • HomeRF • IrDA • USB • IEEE 1394 (FireWire) • Thunderbolt • Wireless USB, Bluetooth • Wi-Fi

# Les couches du modèle OSI

Entité A

| | |
|---|---|
| 7. | Application |
| 6. | Présentation |
| 5. | Session |
| 4. | Transport |
| 3. | Réseau |
| 2. | Liaison de données |
| 1. | Physique |

*Contribution de Laure Petrucci*

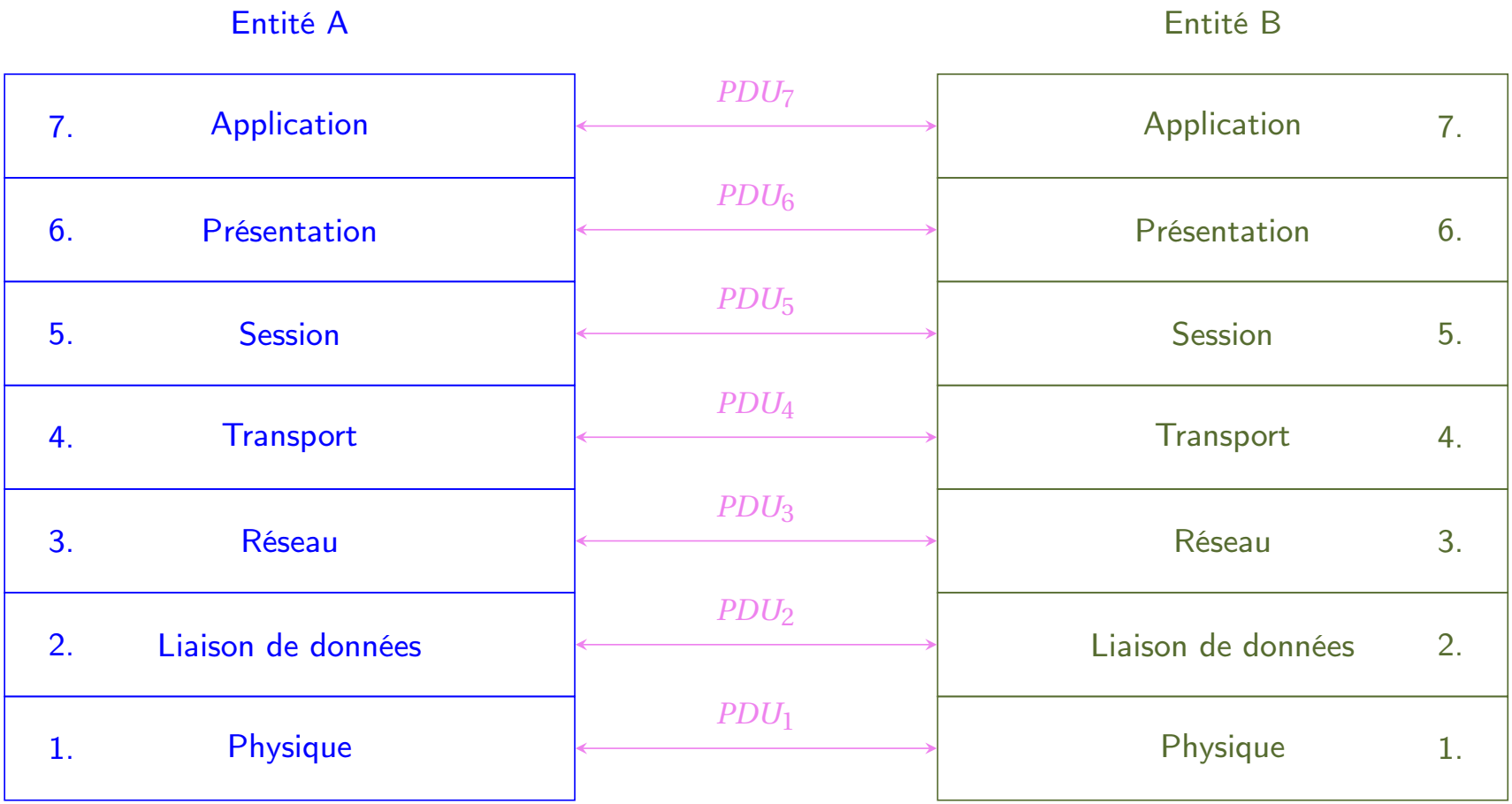
# Les couches du modèle OSI

| Entité A | |
|---|---|
| 7. | Application |
| 6. | Présentation |
| 5. | Session |
| 4. | Transport |
| 3. | Réseau |
| 2. | Liaison de données |
| 1. | Physique |

| Entité B | |
|---|---|
| Application | 7. |
| Présentation | 6. |
| Session | 5. |
| Transport | 4. |
| Réseau | 3. |
| Liaison de données | 2. |
| Physique | 1. |

*Contribution de Laure Petrucci*

# Les couches du modèle OSI




**PDU — Protocol Data Unit**

$PDU_n$ : protocole d'échange entre deux couches de niveau $n$

*Contribution de Laure Petrucci*

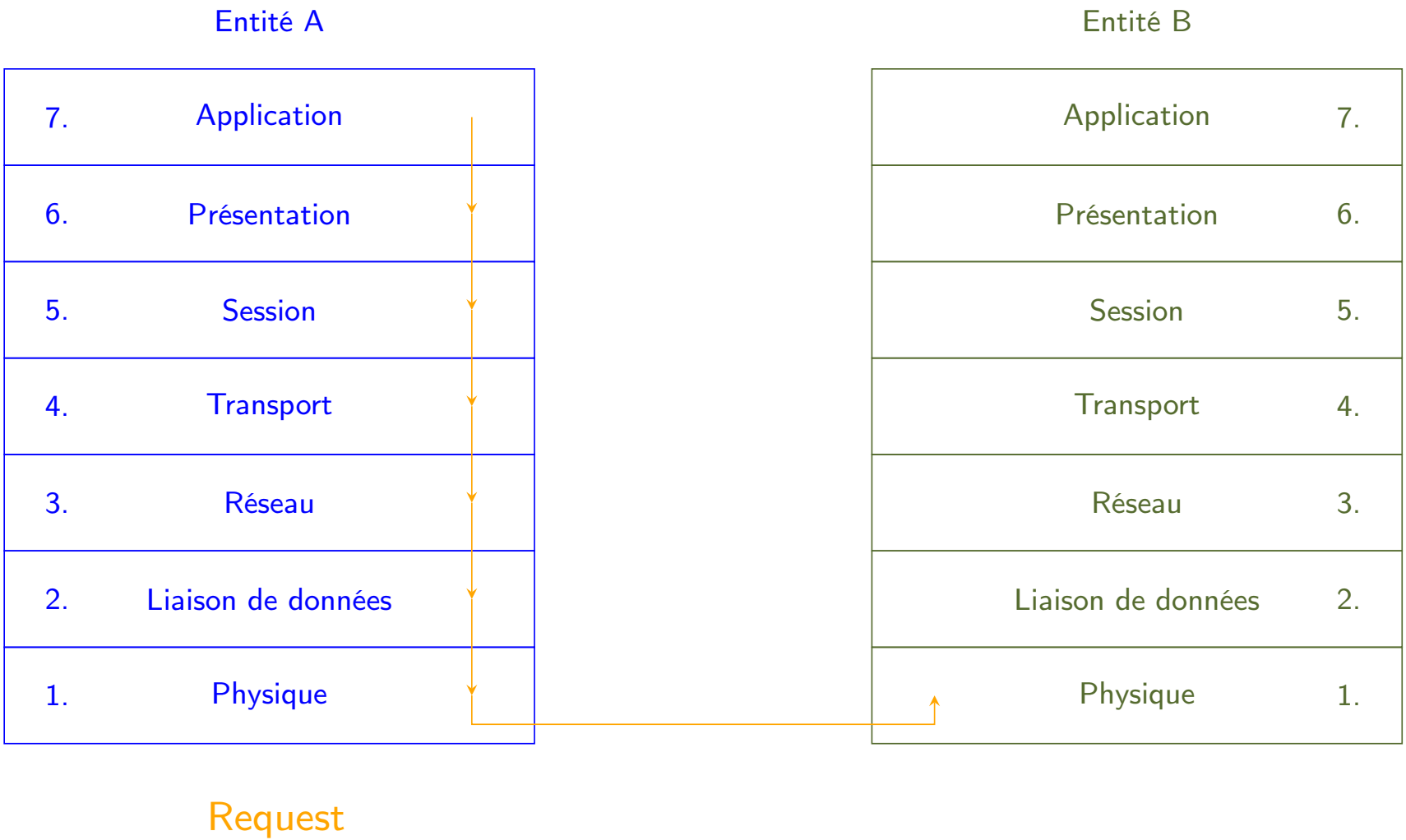
# Les couches du modèle OSI


**SDU — Service Data Unit**

$SDU_n$ : service fourni par la couche $n$ à la couche $n+1$

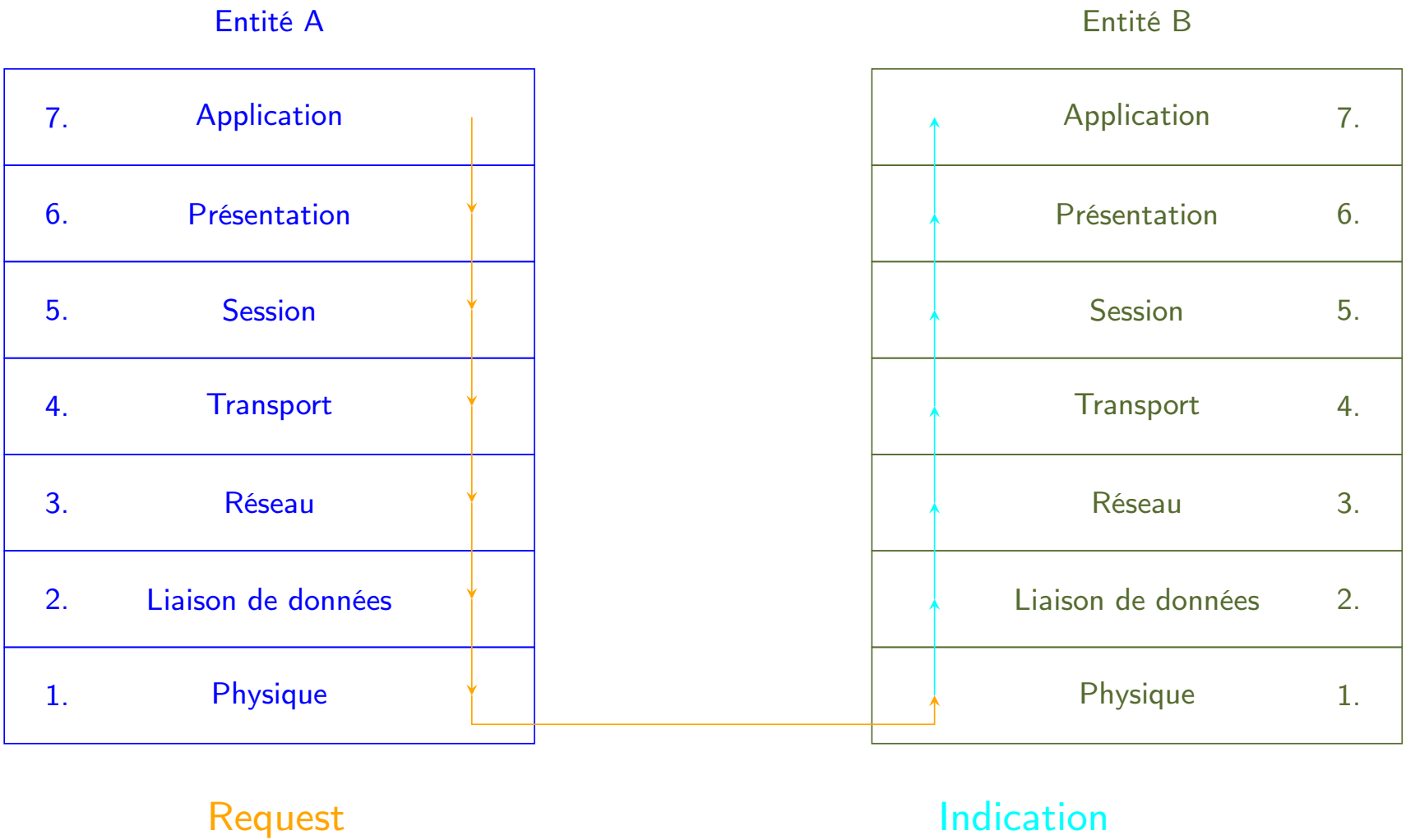
*Contribution de Laure Petrucci*

# Les couches du modèle OSI



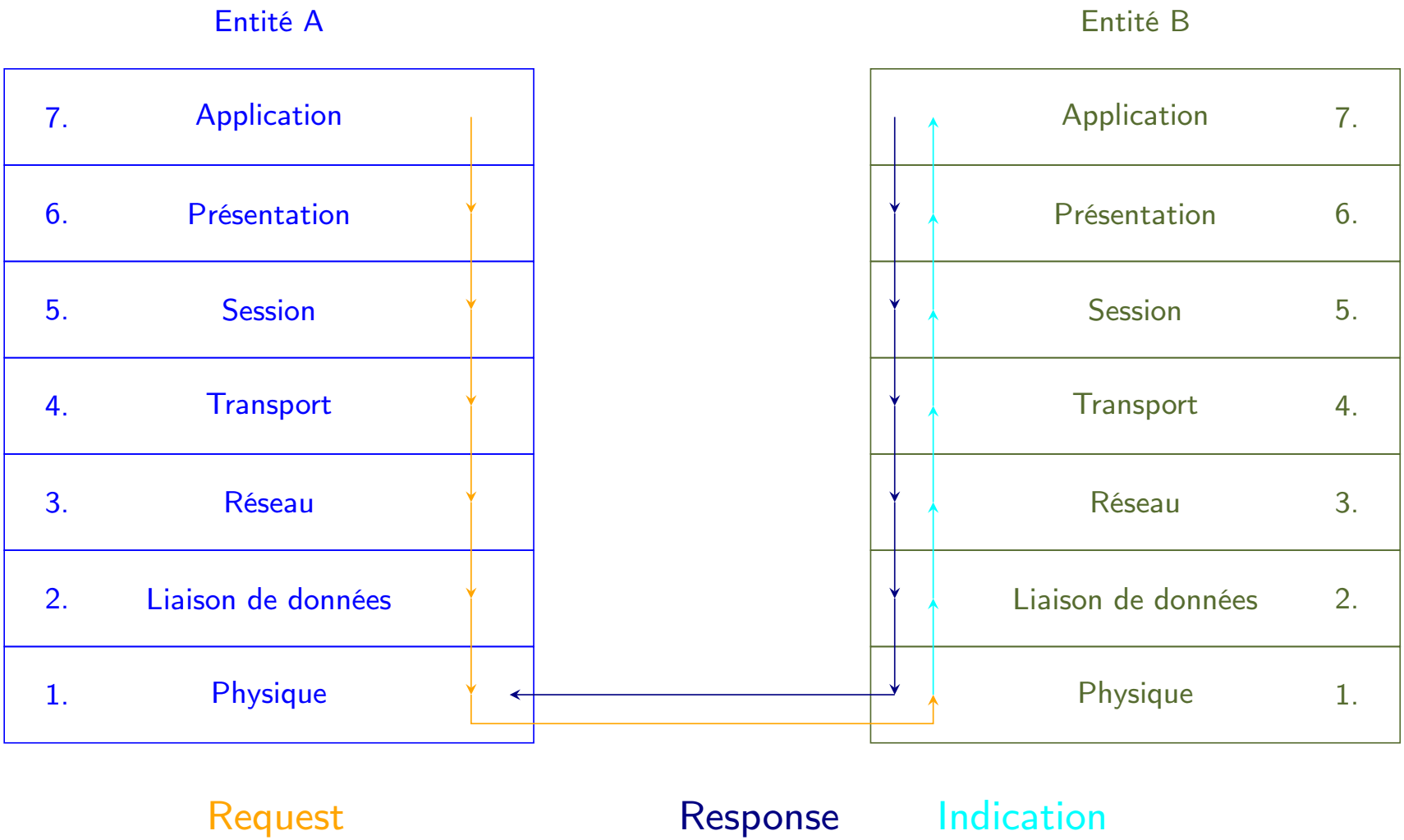

*Contribution de Laure Petrucci*

# Les couches du modèle OSI



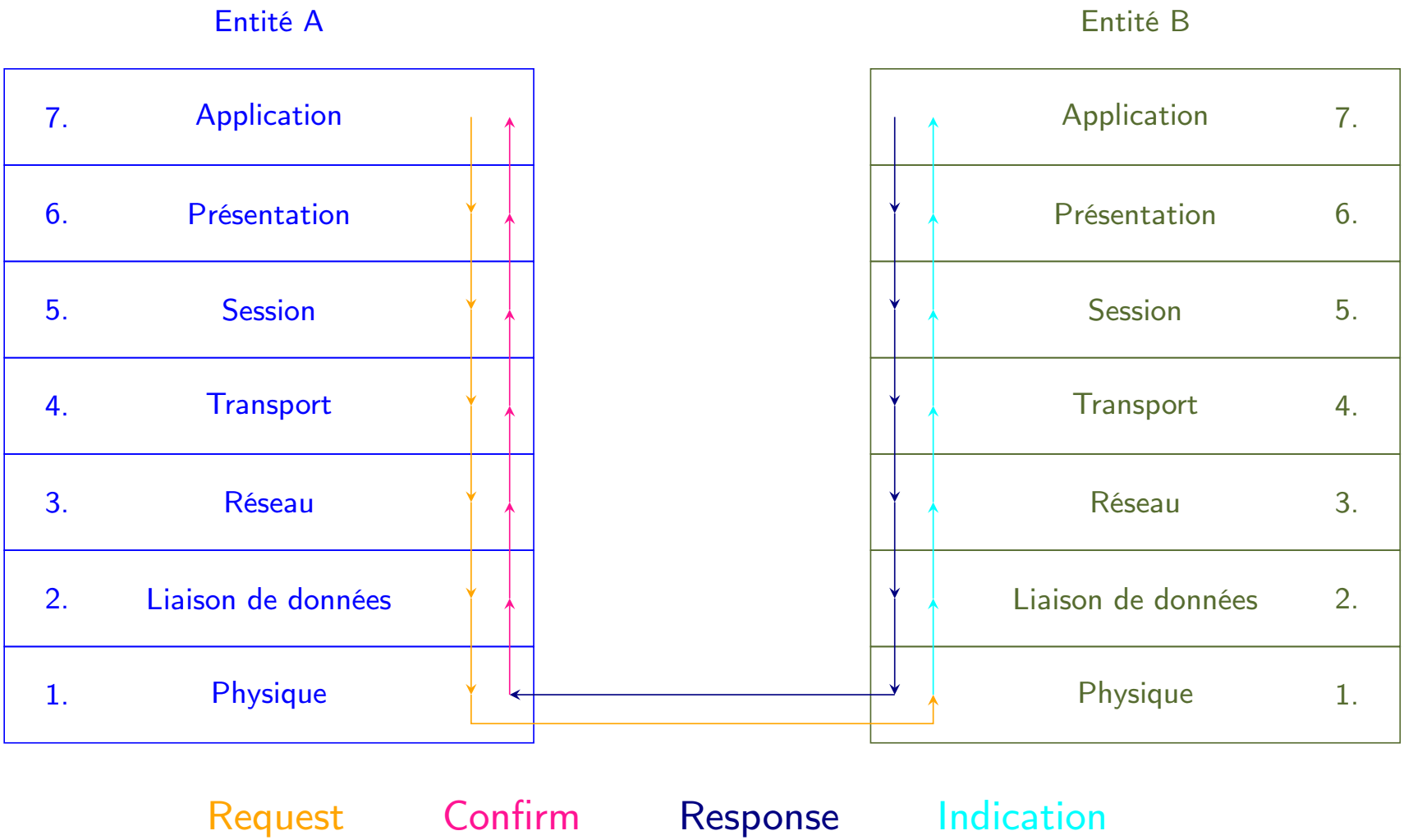

*Contribution de Laure Petrucci*

# Les couches du modèle OSI




*Contribution de Laure Petrucci*

# Les couches du modèle OSI




*Contribution de Laure Petrucci*

# Plan du cours

# Encapsulation

## Communications traversant les couches

Quand un message descend le long du modèle OSI, chaque couche :

- Récupère les données qui lui arrivent de la couche de niveau supérieur
- Ajoute son **enveloppe protocolaire**
- Transmet le tout à la couche de niveau inférieur

En pratique : l'enveloppe est un en-tête pour les couches 1 et 3 à 7, un en-tête et une séquence de fin pour la couche 2.

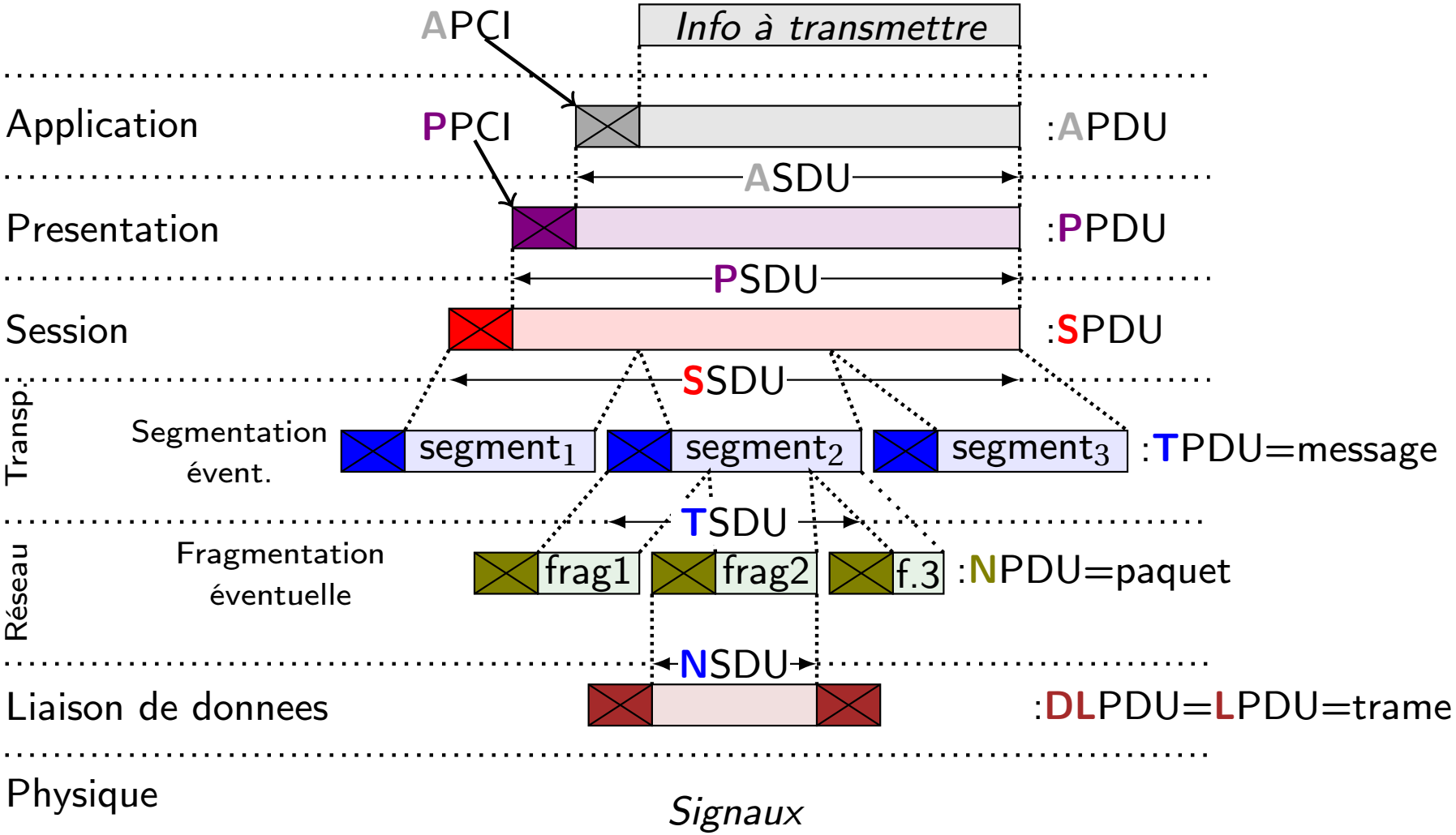
## Communications remontant la pile

Quand un message monte le long du modèle OSI, chaque couche :

- Récupère les données qui lui arrivent de la couche de niveau inférieur
- Retire son **enveloppe protocolaire**
- Transmet le tout à la couche de niveau supérieur

Données de la couche $n$ + en-tête de la couche $n$ = données de la couche $n+1$

# Encapsulation




*Contribution de Franck Butelle*

# Interconnexion

**Au niveau 1** : le **hub** (concentrateur)

- Utilisé pour des réseaux locaux
- "Multiprise" : amplifie et renvoie le signal sur toutes ses sorties
- Réservé à des réseaux de petite taille

**Au niveau 2** : le **switch** (commutateur)

- Distribue les données à leur destinataire
- Fonctionne avec ARP
- Élimine les collisions

**Au niveau 3** : le **routeur**

- Interconnecte des réseaux
- Permet de créer des sous-réseaux
- Ou de relier des réseaux
- Échelle de réseau supérieure : situé à la frontière entre les réseaux

# Plan du cours

# Plan du cours

# Histoire et développement

Développement des réseaux longues distances

- ARPANET aux États-Unis, Cyclades en France...

Besoin :

- Interconnecter des hôtes **hétérogènes**
- Sur des liens physiques **hétérogènes**
- Avec une infrastructure **décentralisée** et **résiliente**

Chronologie :

- 1957 : Lancement du 1er Spoutnik, création de l'Advanced Research Projects Agency (ARPA)
- 1962 : l'US Air Force demande la création d'un réseau capable de résister à toute attaque massive
- 1969 : création d'ARPANET
- 1971 : invention du courrier électronique
- 1973 : création de TCP/IP, adopté pour Arpanet en 1974
- 1983 : invention du système DNS, mise en place des TLD en 1984
- 1989 : création du World Wide Web

# Gouvernance d'Internet

À l'origine : militaire

- ARPANET : DoD, DARPA

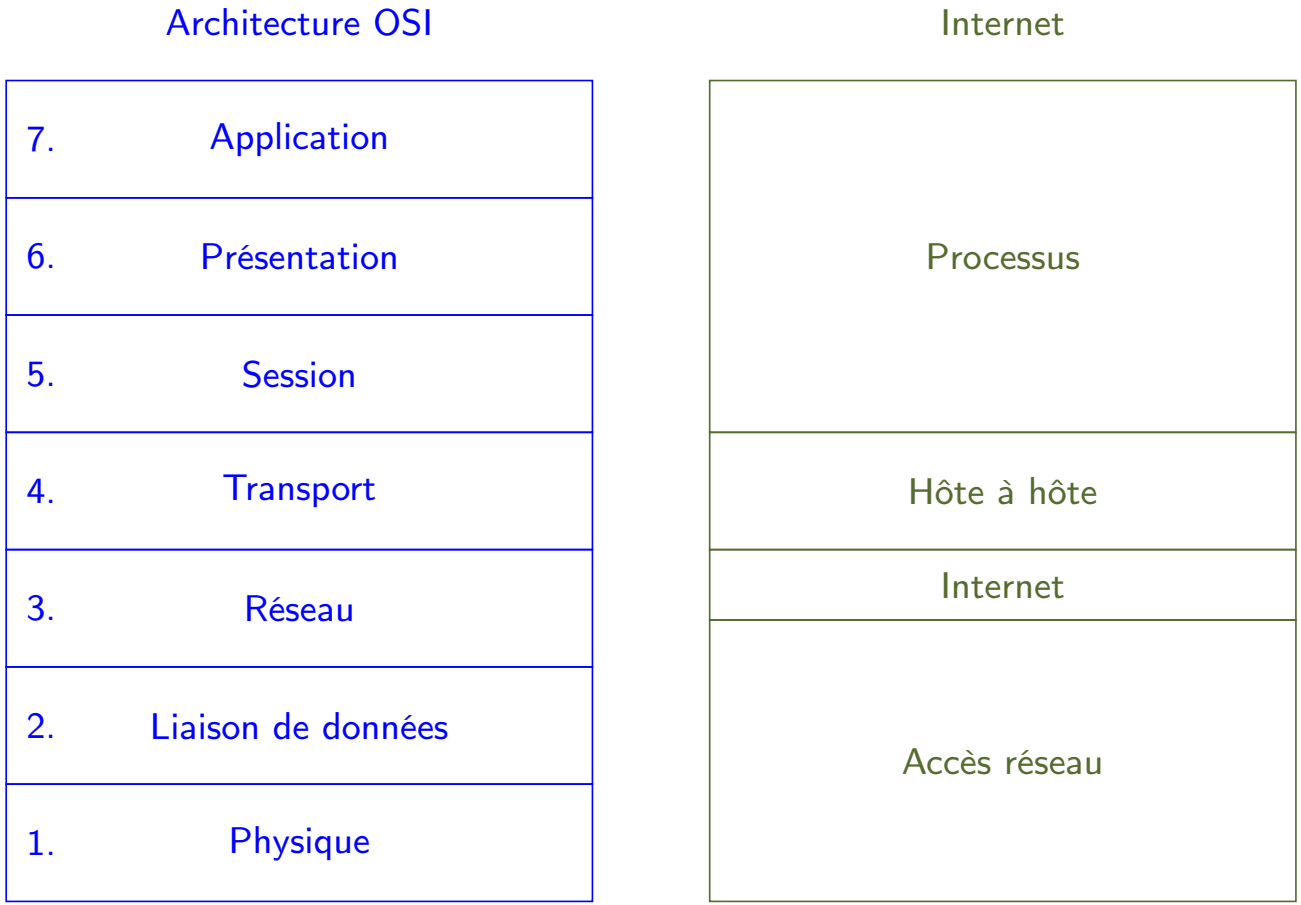
Développement/coordination assurés par :

- 1979 – 1983 : ICCB (Internet Control and Configuration Board)
- 1983 – 1989 : IAB (Internet Activities Board)
- 1989 – . . . : IRTF (Internet Research Task Force) et IETF (Internet Engineering Task Force)

Les protocoles sont définis dans des **RFC** (Requests For Comments) et non pas dans des normes

- 1ere RFC : 7 avril 1969 (date considérée comme la naissance d'Internet)
- RFC 791 : Internet Protocol
- RFC 793 : Transmission Control Protocol
- RFC 821 : Simple Mail Transfer Protocol
- RFC 882 et 883 : DNS
- RFC 959 : File Transfer Protocol
- RFC 2324 : Hyper Text Coffee Pot Control Protocol
- RFC 2549 puis 6214 : IP over Avian Carriers

# Architecture Internet




*Contribution de Laure Petrucci*

# Protocoles

## Processus

Telnet, FTP, SMTP, NFS, SNMP, Ping, ...

*Contribution de Laure Petrucci*

# Protocoles

## Processus

Telnet, FTP, SMTP, NFS, SNMP, Ping, ...

## Hôte à hôte

TCP, UDP, ICMP

*Contribution de Laure Petrucci*

# Protocoles

## Processus

Telnet, FTP, SMTP, NFS, SNMP, Ping, ...

## Hôte à hôte

TCP, UDP, ICMP

## Internet

IP, ARP, RARP

*Contribution de Laure Petrucci*

# Protocoles

## Processus

Telnet, FTP, SMTP, NFS, SNMP, Ping, ...

## Hôte à hôte

TCP, UDP, ICMP

## Internet

IP, ARP, RARP

## Accès réseau

FDDI, X25, Xerox Ethernet, IEEE 802 (couches LLC et MAC), Arpanet, ...

*Contribution de Laure Petrucci*

# Plan du cours

# Structure d'un paquet IP

| v. IP (4) | lg. entête (4) | DSCP (6) | ECN (2) | lg. tot. en octets (16) | | | |
|---|---|---|---|---|---|---|---|
| Identification (16) | | | | 0 | DF | MF | offset (13) |
| Durée de vie — TTL (8) | | protocole (8) | | ctrl. erreur entête (16) | | | |
| Adresse IP émetteur (32) | | | | | | | |
| Adresse IP destinataire (32) | | | | | | | |
| Options éventuelles | | | | | | | |
| bourrage éventuel pour les options | | | | | | | |
| Données | | | | | | | |

Les tailles des champs sont en nombre de bits.

- v. IP : version du protocole (IPv4, IPv6)
- lg entête : longueur de l'entête en paquets de 4 octets.
- DSCP (Differenciated Service Code Point) : permet aux routeurs de traiter au mieux le paquet.
- ECN (Explicit Congestion Notification) : gestion de congestions
- DF (Don't Fragment) : pas de fragmentation
- MF (More Fragments) : il y a d'autres fragments
- Offset : position du fragment dans le message, en nombre de blocs de 8 octets

*Contribution de Laure Petrucci*

# TCP — Transmission Control Protocol

## Objectif

Fournir un service de transport fiable

## Comment ?

- indépendant des protocoles inférieurs
- connexions bi-directionnelles
- processus serveurs et processus clients
- communication identifiable de manière unique par :
  - adresse source
  - port source
  - adresse destination
  - port destination

*Contribution de Laure Petrucci*

# Structure d'un *segment* TCP

| port source (16) | | | port destination (16) |
|---|---|---|---|
| numéro de séquence (32) | | | |
| numéro d'acquittement (32) | | | |
| Lg. entête (4) | réservé (4) | drapeaux (8) | taille fenêtre (16) |
| checksum (16) | | | pointeur urgent (16) |
| Options éventuelles | | | |
| Données | | | |

# Structure d'un *segment* TCP

| port source (16) | | | port destination (16) |
|---|---|---|---|
| numéro de séquence (32) | | | |
| numéro d'acquittement (32) | | | |
| Lg. entête (4) | réservé (4) | drapeaux (8) | taille fenêtre (16) |
| checksum (16) | | | pointeur urgent (16) |
| Options éventuelles | | | |
| Données | | | |

## Drapeaux

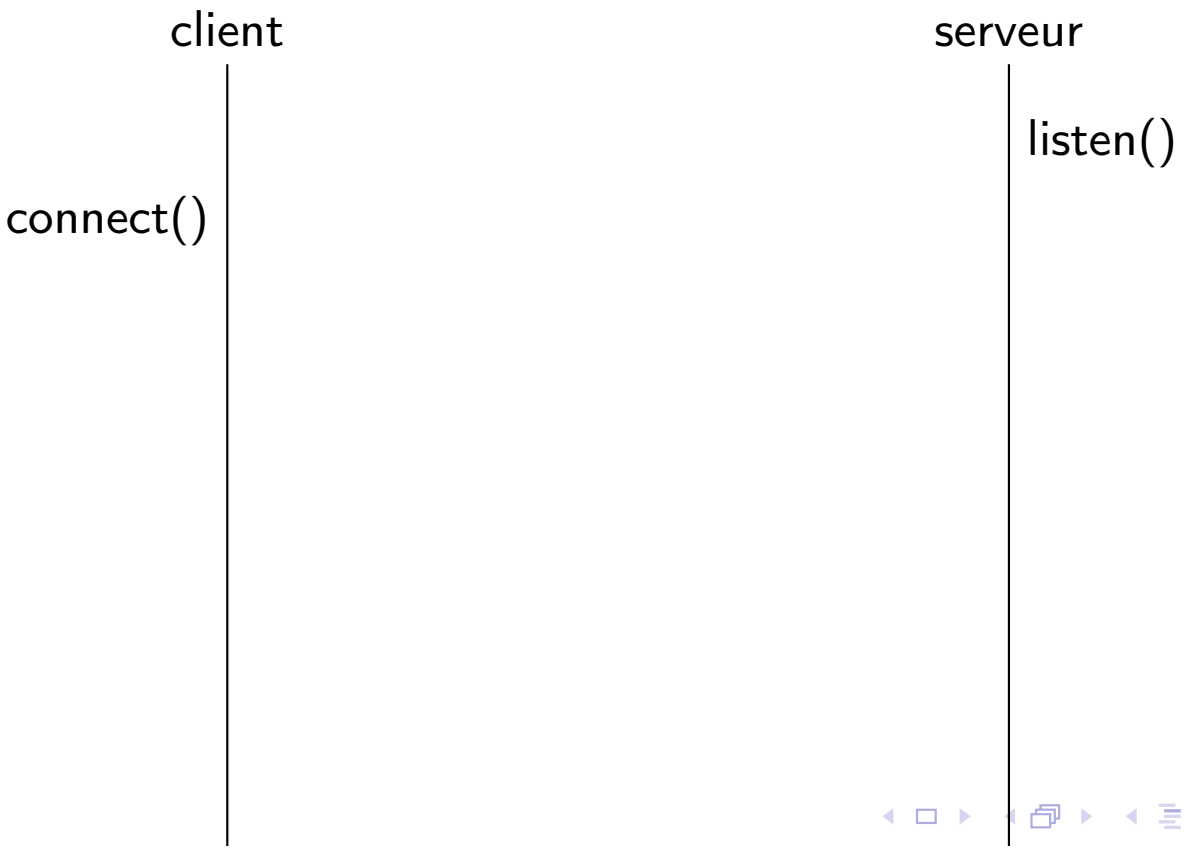
- CWR, ECE (Congestion Window Reduced) : gestion de la congestion
- URG (urgent) : utilisation du pointeur urgent
- ACK (acknowledgement) : utilisation du numéro d'acquittement, ou réponse à l'ouverture de connexion
- PSH (push) : demande de transmission des données à la couche supérieure
- RST (reset) : réinitialisation de la connexion
- SYN (synchronise) : synchronisation des numéros de séquence et établissement de la connexion
- FIN (finished) : demande de fermeture de la connexion

# Protocole TCP

## Établissement de la connexion

Triple poignée de main :

- Le client envoie une requête de connexion : paquet SYN
- Le serveur accepte : paquet SYN/ACK
- Le client acquitte la réception de l'acceptation : paquet ACK

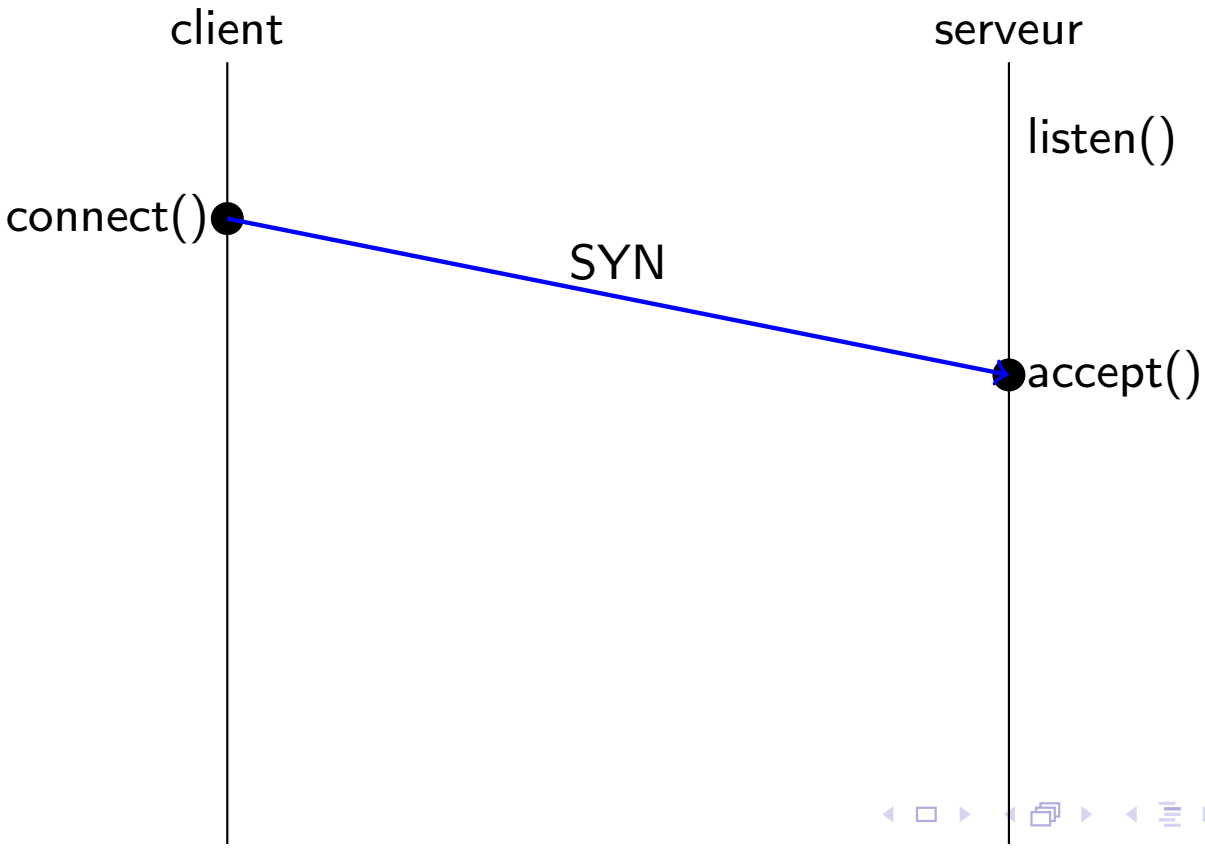


# Protocole TCP

## Établissement de la connexion

Triple poignée de main :

- Le client envoie une requête de connexion : paquet SYN
- Le serveur accepte : paquet SYN/ACK
- Le client acquitte la réception de l'acceptation : paquet ACK

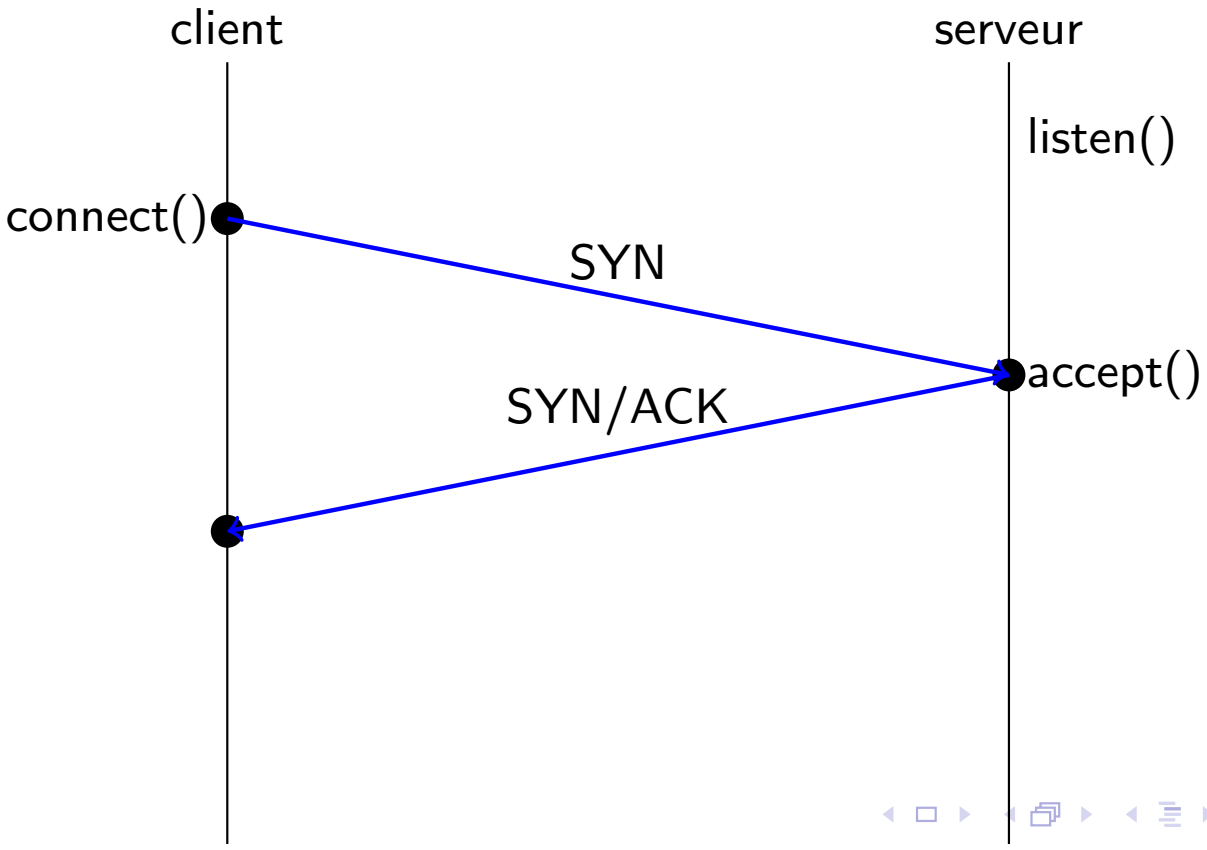


# Protocole TCP

## Établissement de la connexion

Triple poignée de main :

- Le client envoie une requête de connexion : paquet SYN
- Le serveur accepte : paquet SYN/ACK
- Le client acquitte la réception de l'acceptation : paquet ACK

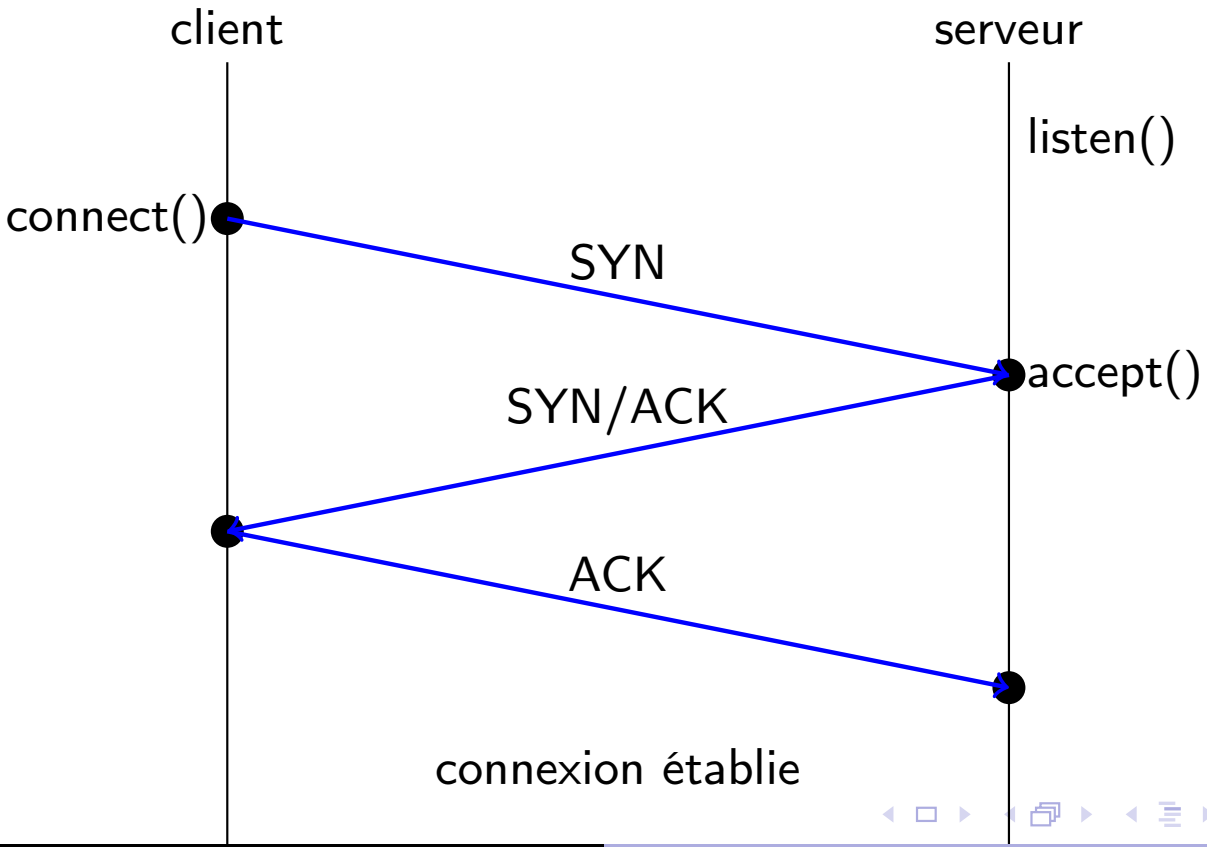


# Protocole TCP

## Établissement de la connexion

Triple poignée de main :

- Le client envoie une requête de connexion : paquet SYN
- Le serveur accepte : paquet SYN/ACK
- Le client acquitte la réception de l'acceptation : paquet ACK



# Protocole TCP

## Communications

Les envois sont acquités : messages ACK

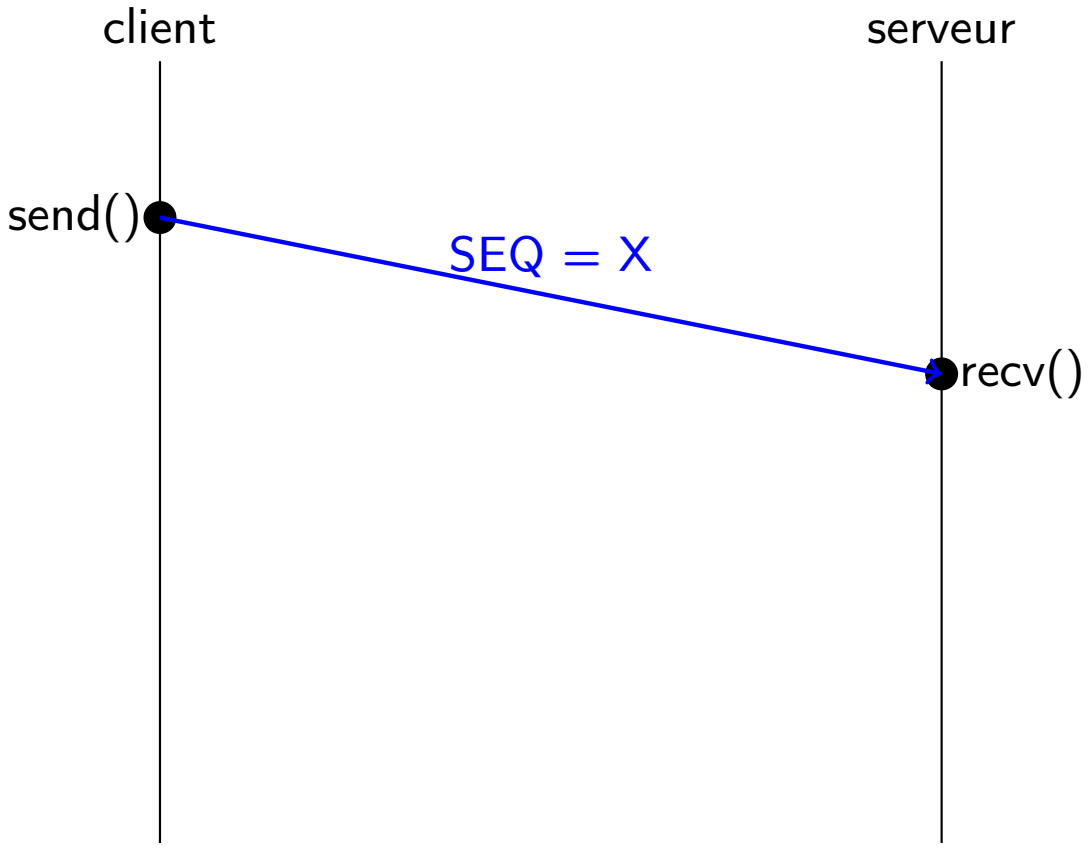
- Problèmes : messages supplémentaires, risque d'engorgement du réseau, bloquage en attente de l'acquittement

Technique de la fenêtre glissante

- Anticipation : l'émetteur continue d'envoyer des trames sans bloquer sur l'attente de l'acquittement
- Il en envoie un certain nombre avant de s'assurer qu'une trame est acquitée
- On n'acquitte pas *toutes* les trames mais seulement toutes les X trames
- Utilisation du numéro de séquence pour l'acquittement

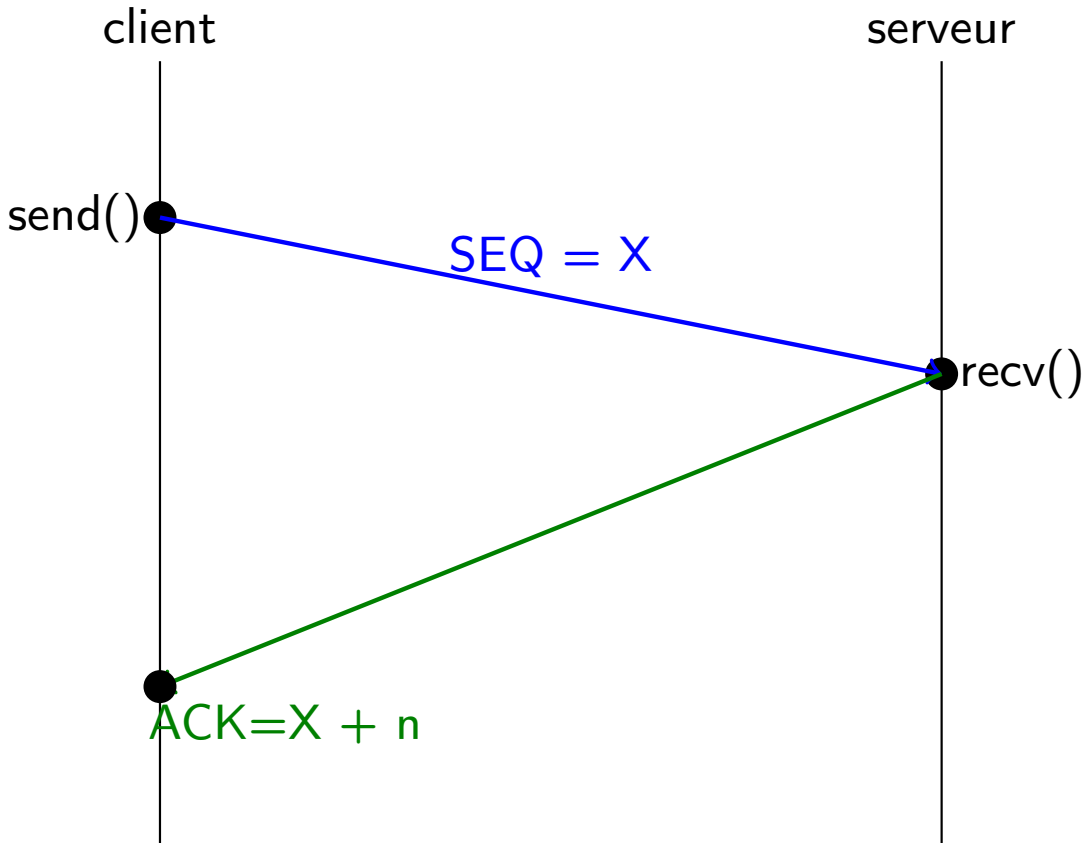
# Protocole TCP

## Communications



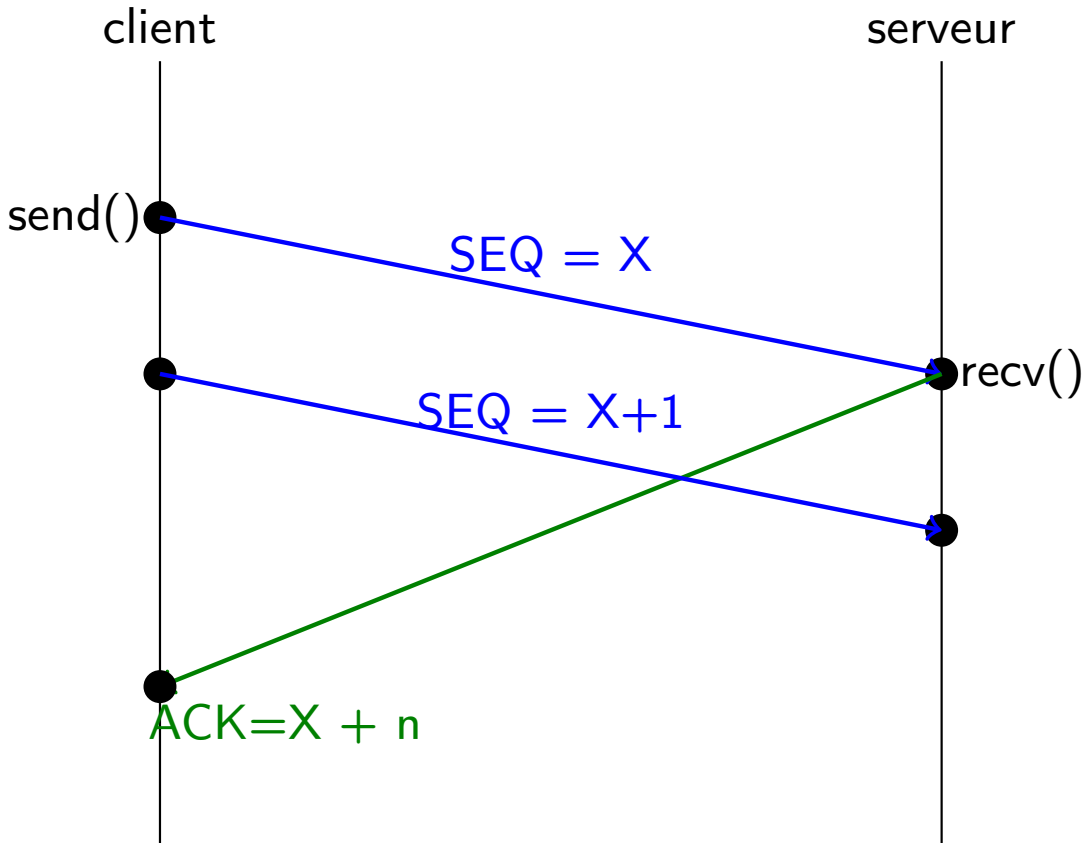
# Protocole TCP

## Communications



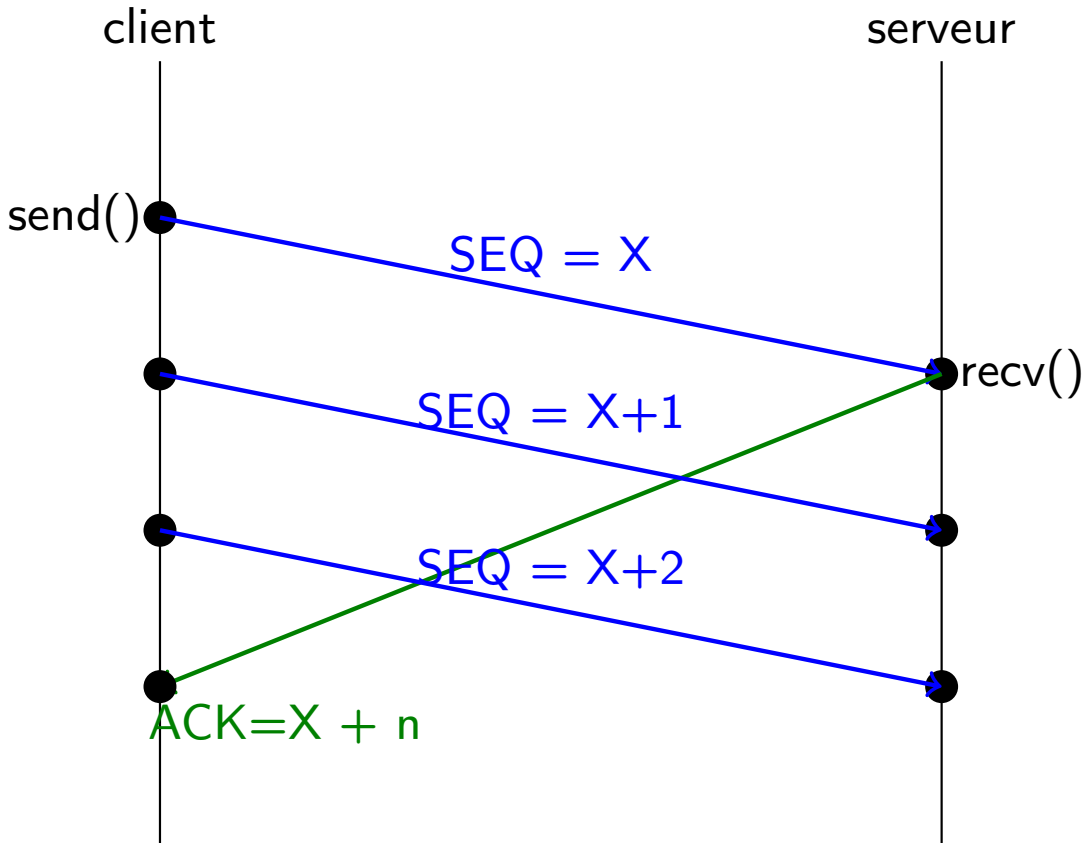
# Protocole TCP

## Communications



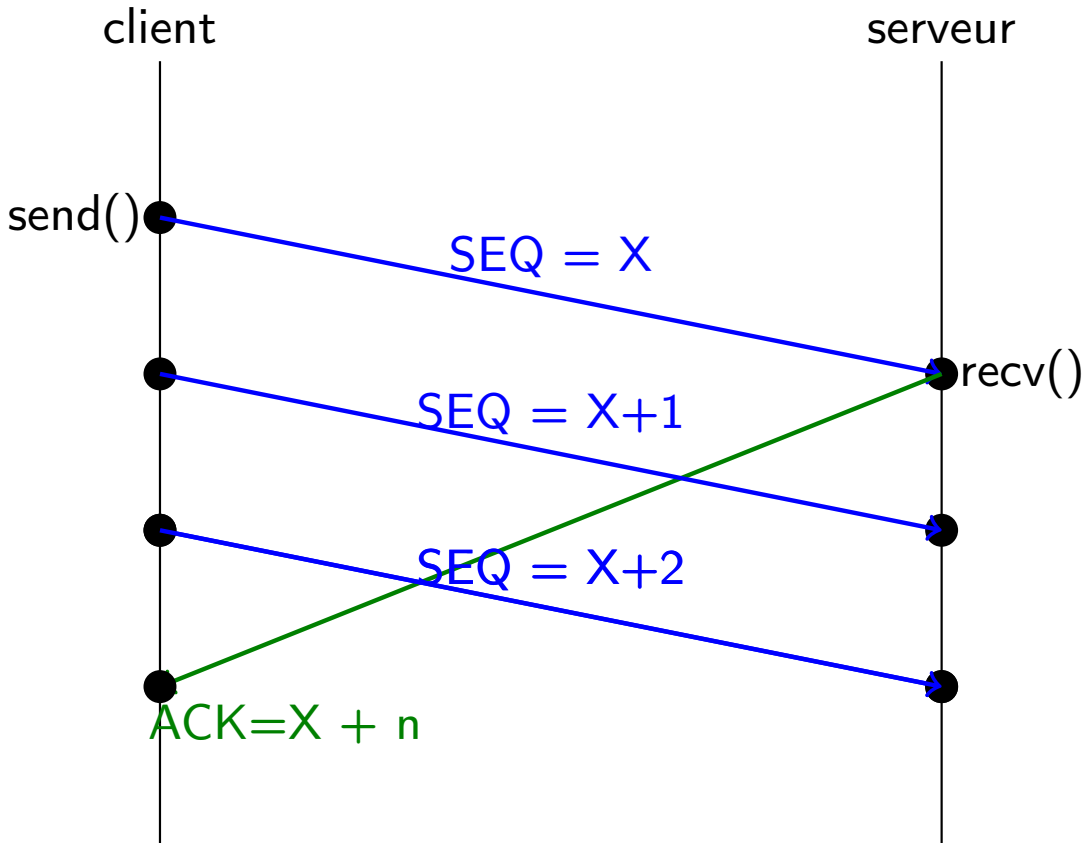
# Protocole TCP

## Communications



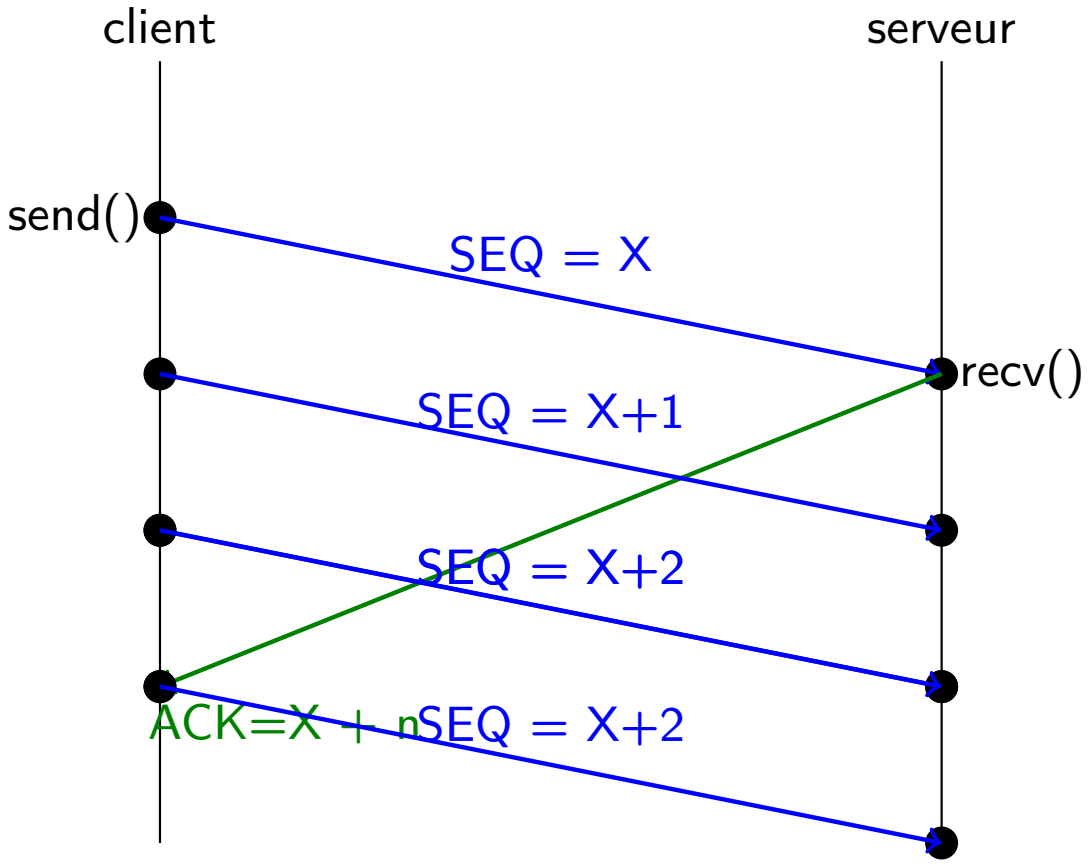
# Protocole TCP

## Communications



# Protocole TCP

## Communications



# Protocole TCP

## Correspondance séquence / acquittement

- **Données** : numéro de séquence = X, numéro d'acquittement = Y
- **Acquittement** : bit ACK à 1, numéro de séquence = Y, numéro d'acquittement = X + taille des données reçues

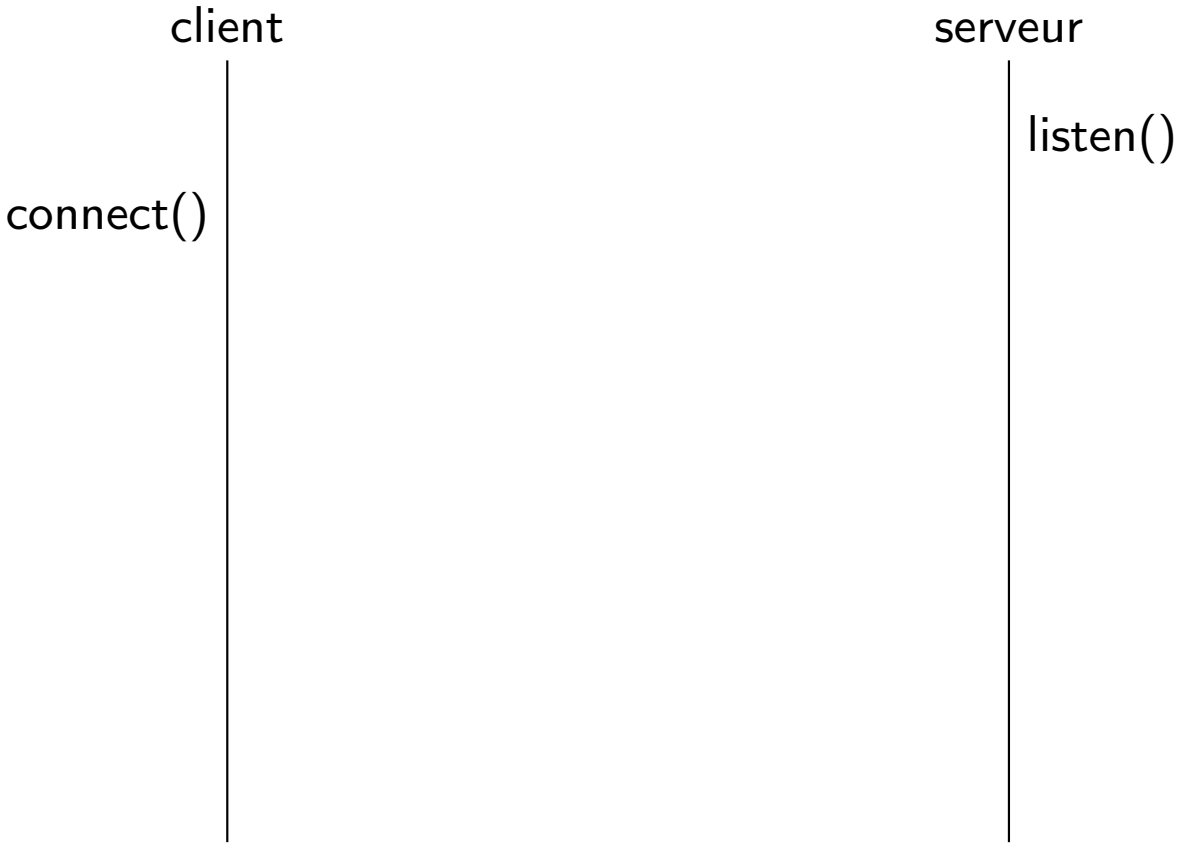
Comment est choisi le numéro de séquence ?

- **Initialisation** : aléatoire, pour éviter les attaques par prédiction du numéro de séquence.
- À chaque **envoi de données** : numéro de séquence précédent + taille des données envoyées dans le segment précédent

# Protocole TCP

## Synchronisation des numéros de séquence à l'ouverture de connexion

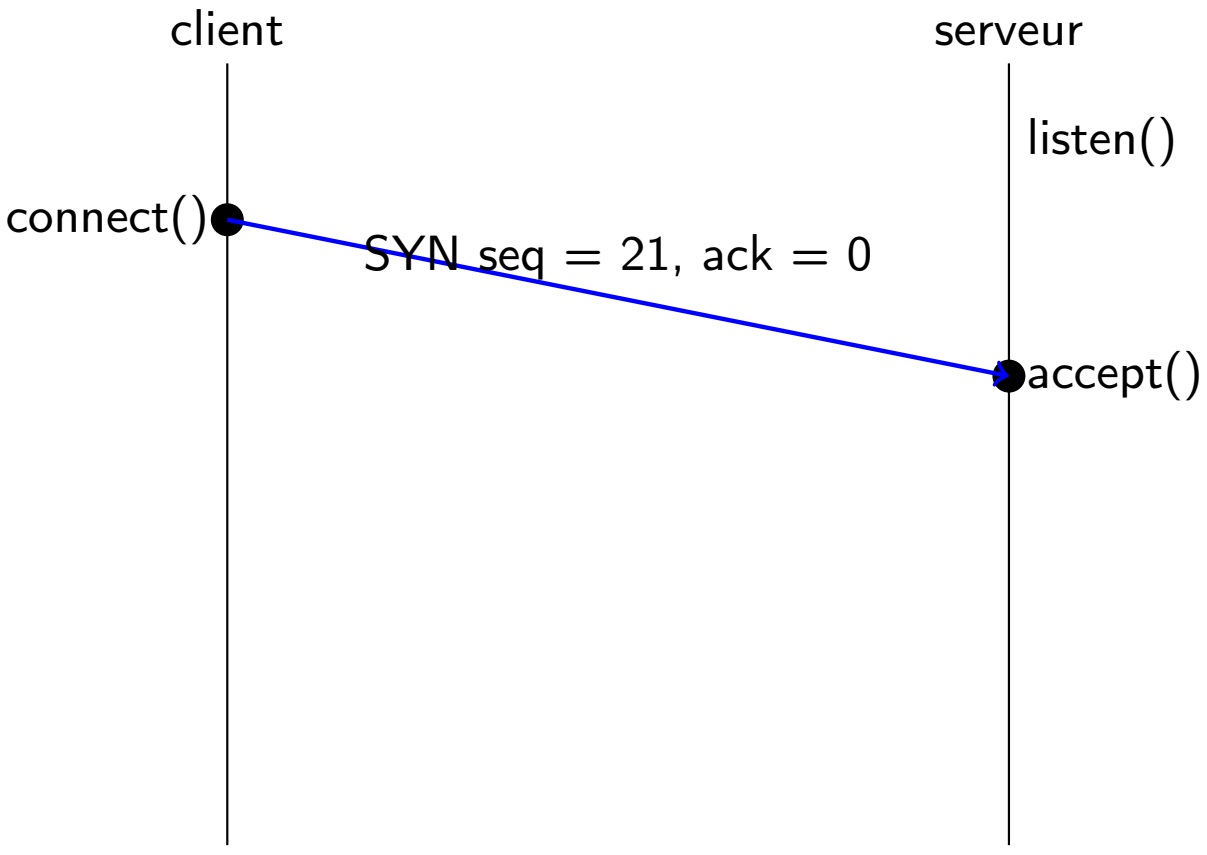
Le client a généré aléatoirement 21, le serveur a généré 86



# Protocole TCP

## Synchronisation des numéros de séquence à l'ouverture de connexion

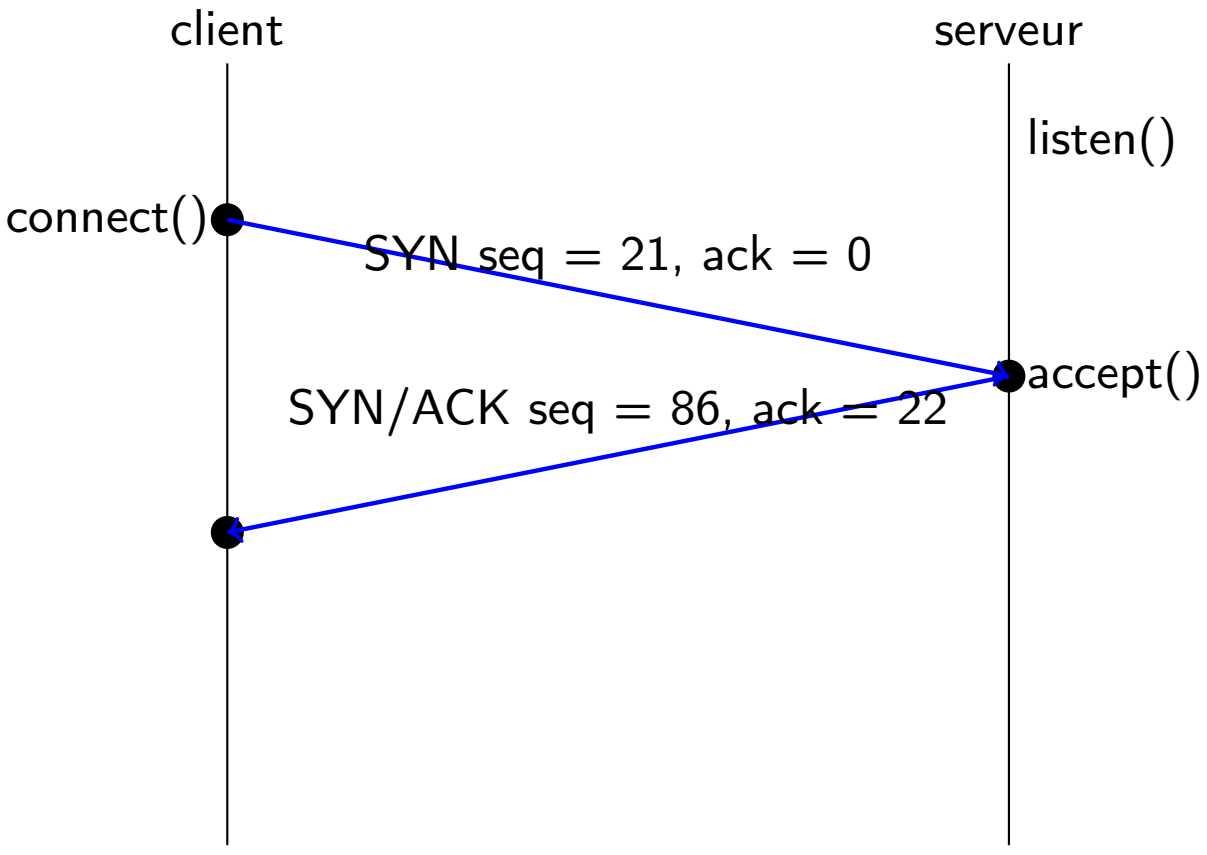
Le client a généré aléatoirement 21, le serveur a généré 86



# Protocole TCP

## Synchronisation des numéros de séquence à l'ouverture de connexion

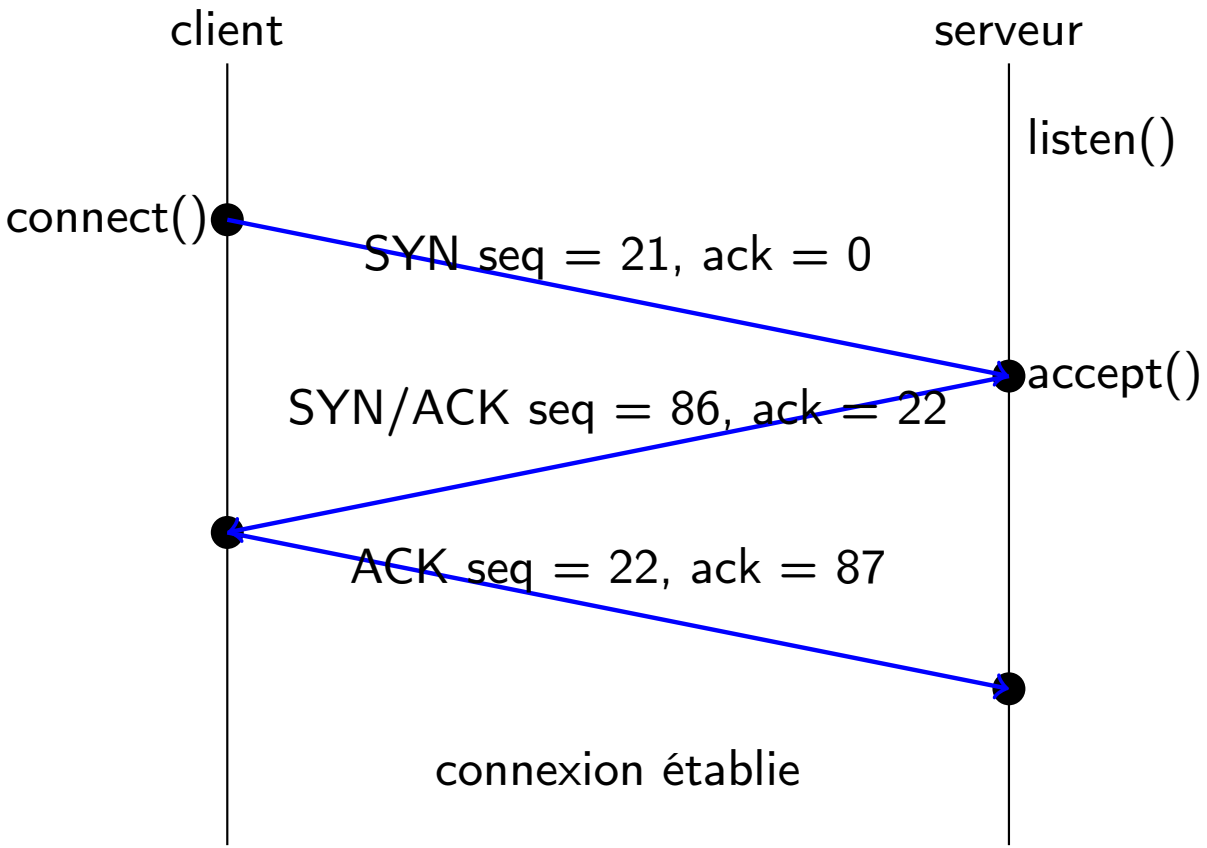
Le client a généré aléatoirement 21, le serveur a généré 86



# Protocole TCP

## Synchronisation des numéros de séquence à l'ouverture de connexion

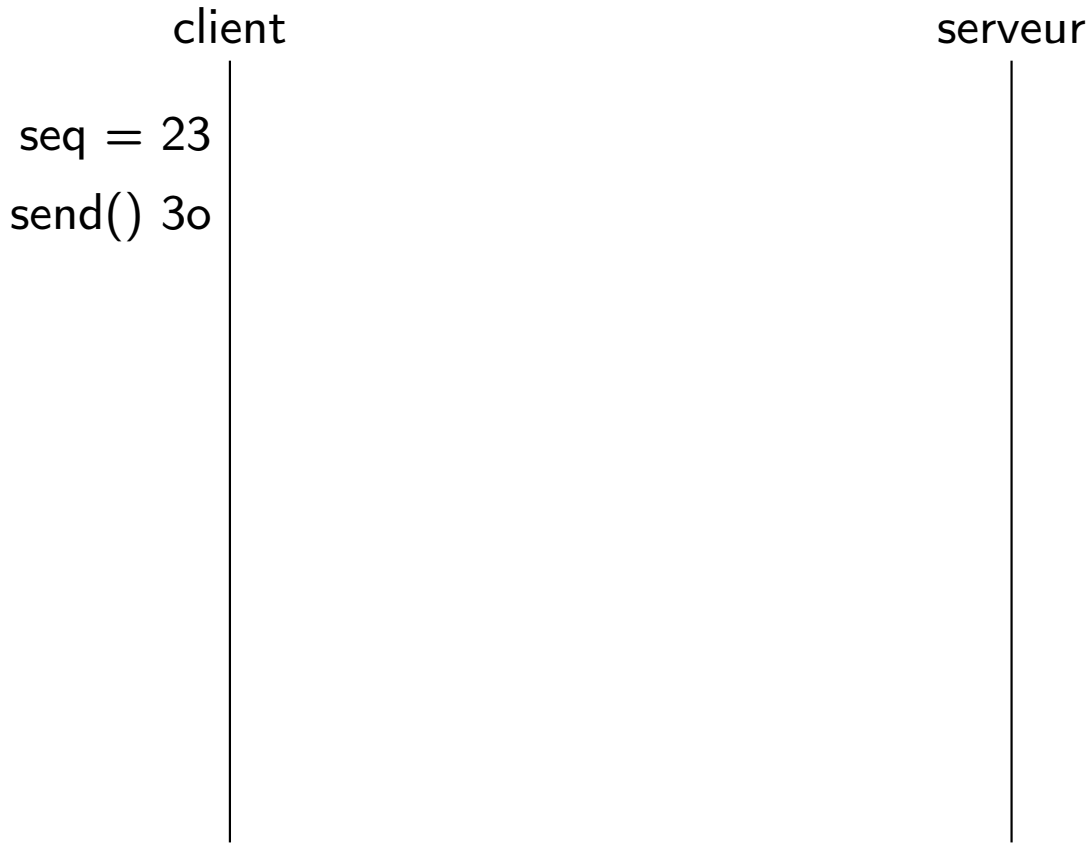
Le client a généré aléatoirement 21, le serveur a généré 86



# Protocole TCP

## Envoi de données

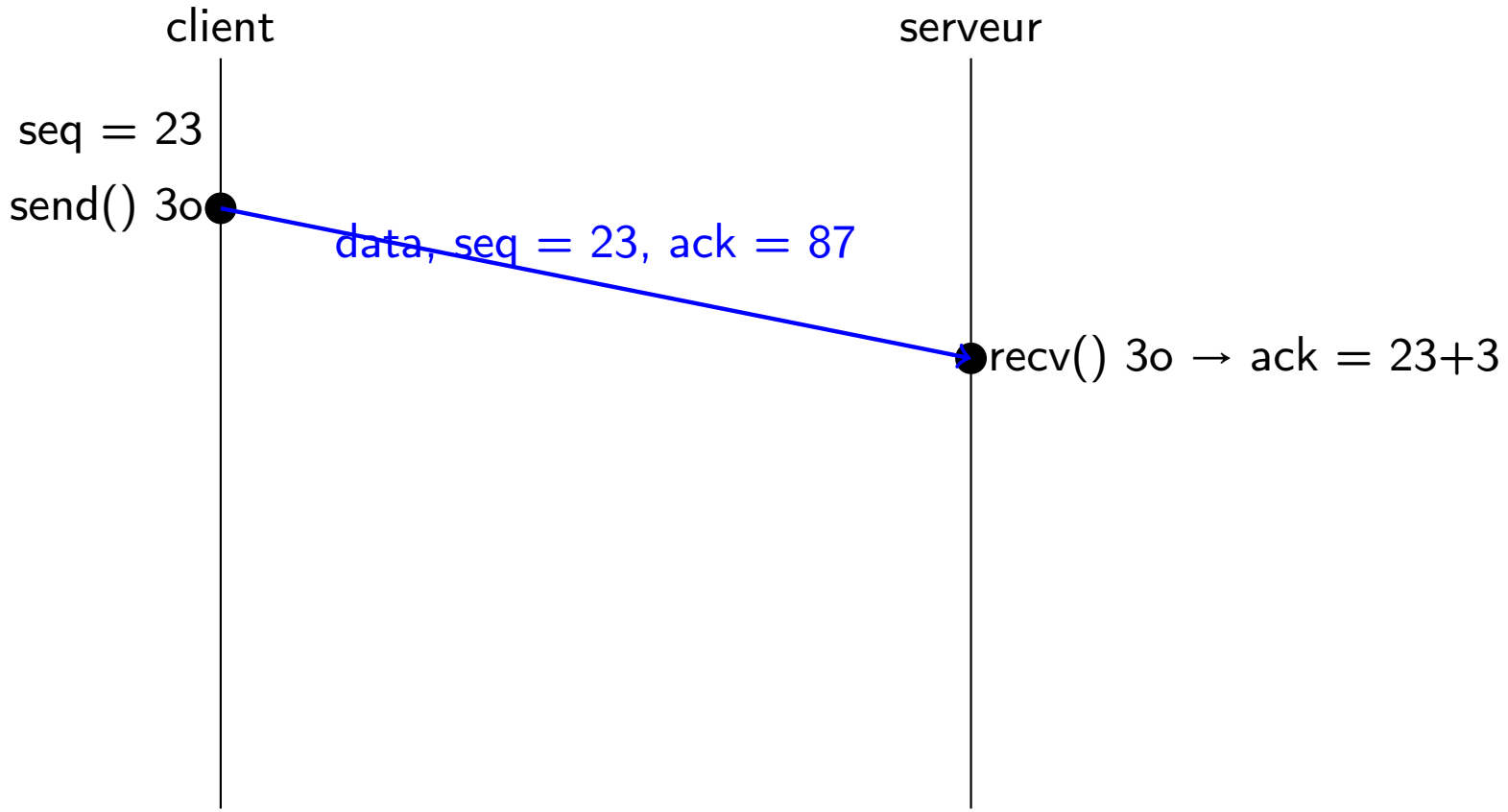
- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22



# Protocole TCP

## Envoi de données

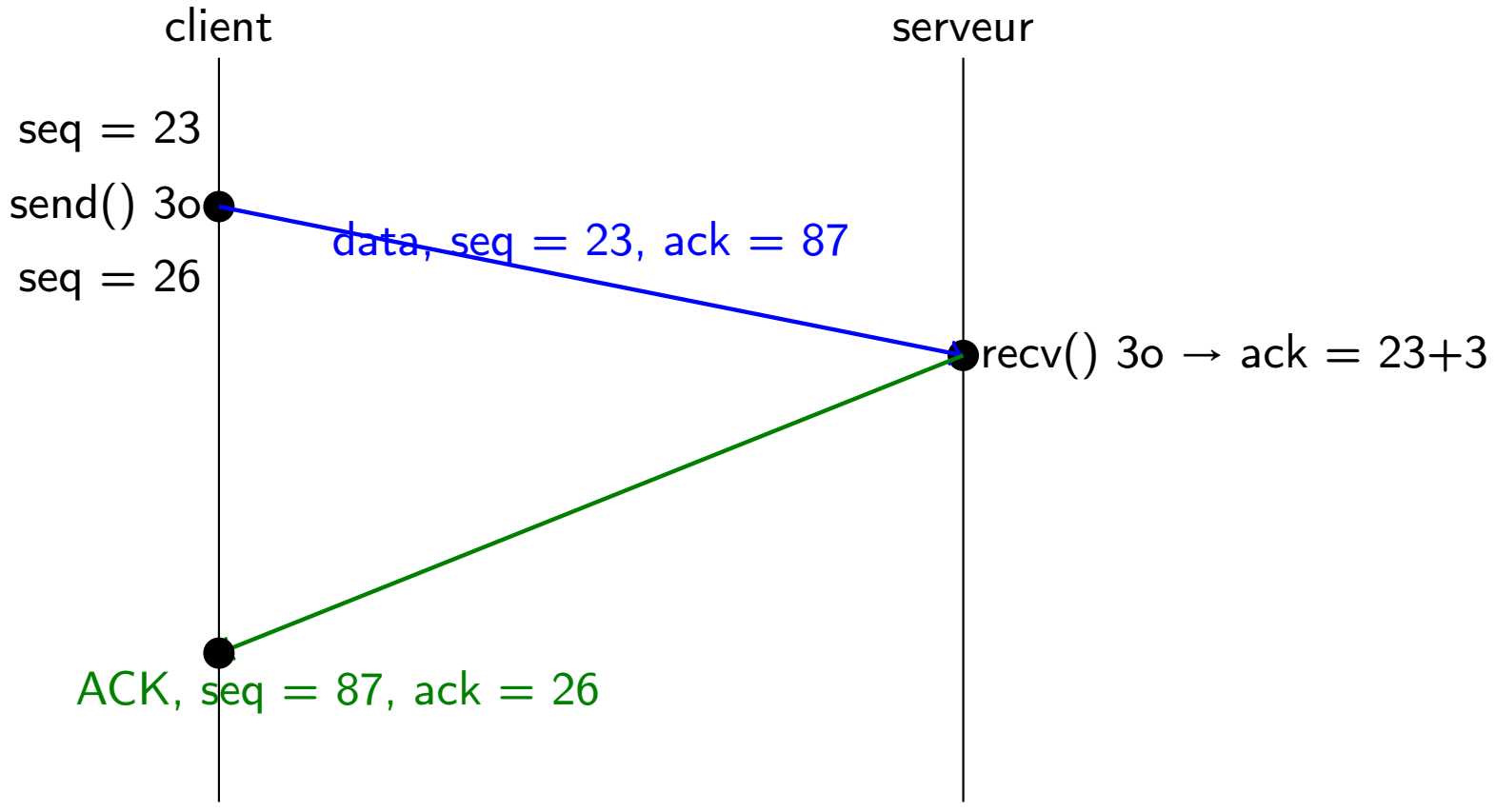
- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22



# Protocole TCP

## Envoi de données

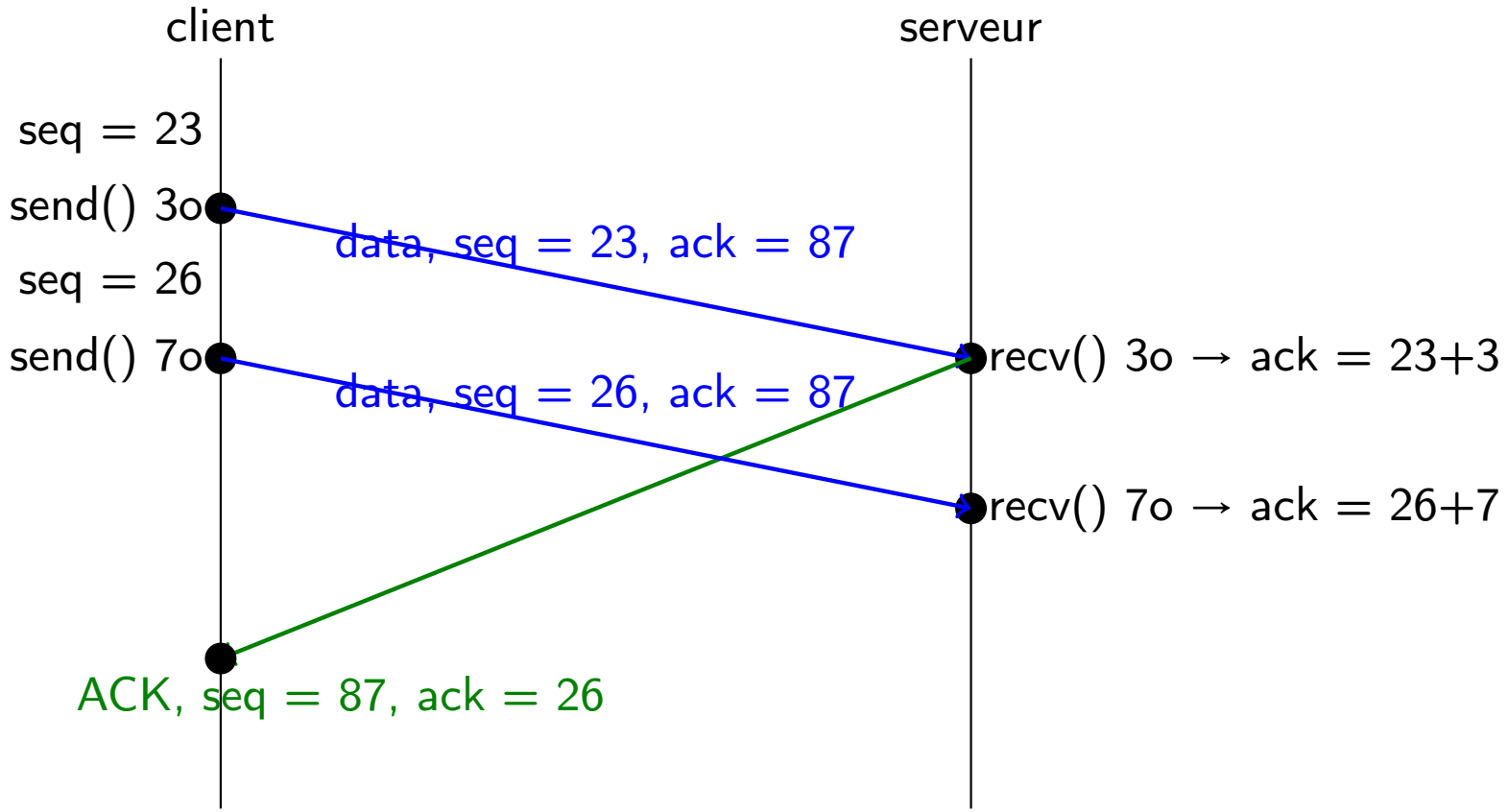
- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22



# Protocole TCP

## Envoi de données

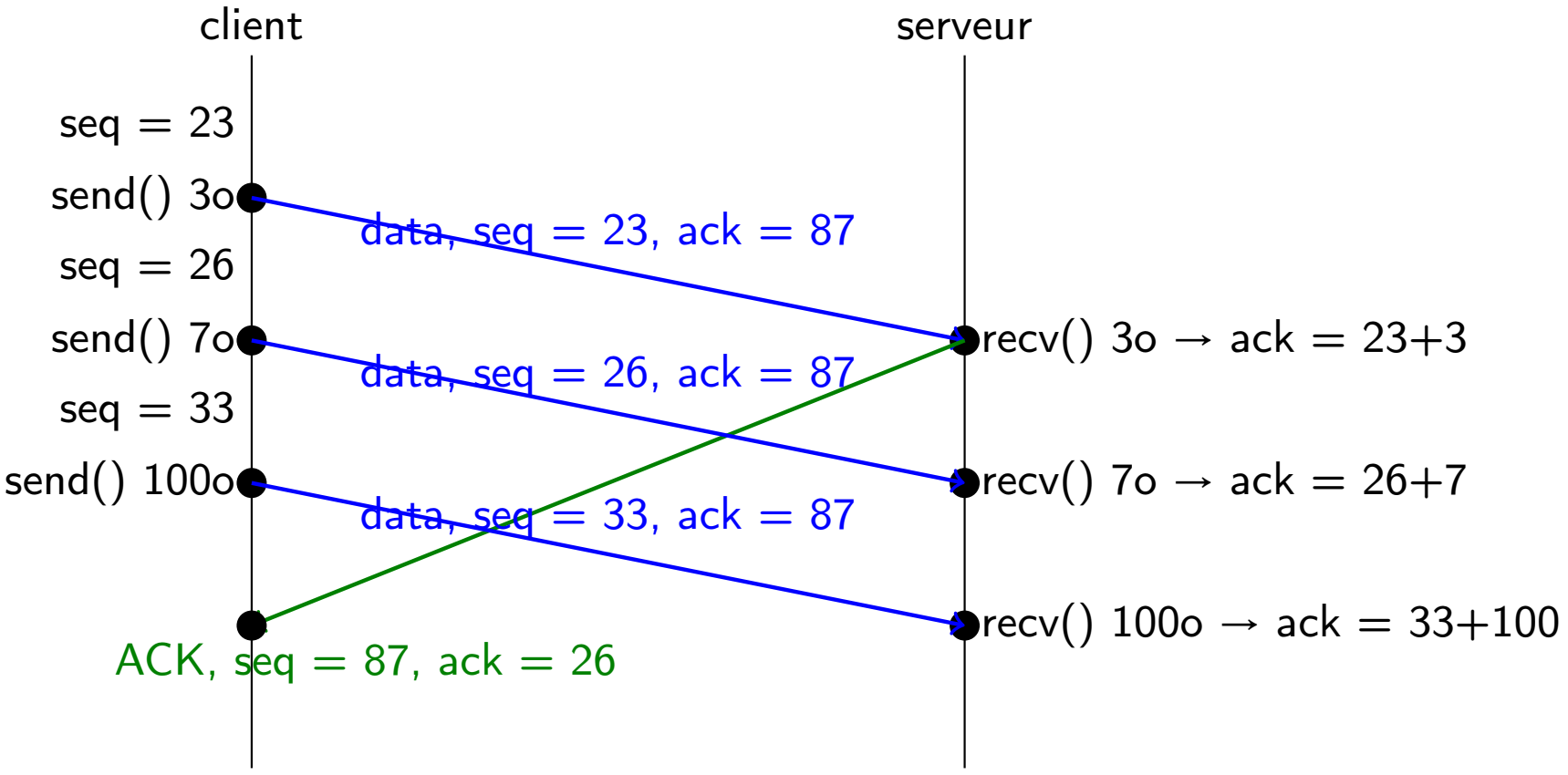
- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22



# Protocole TCP

## Envoi de données

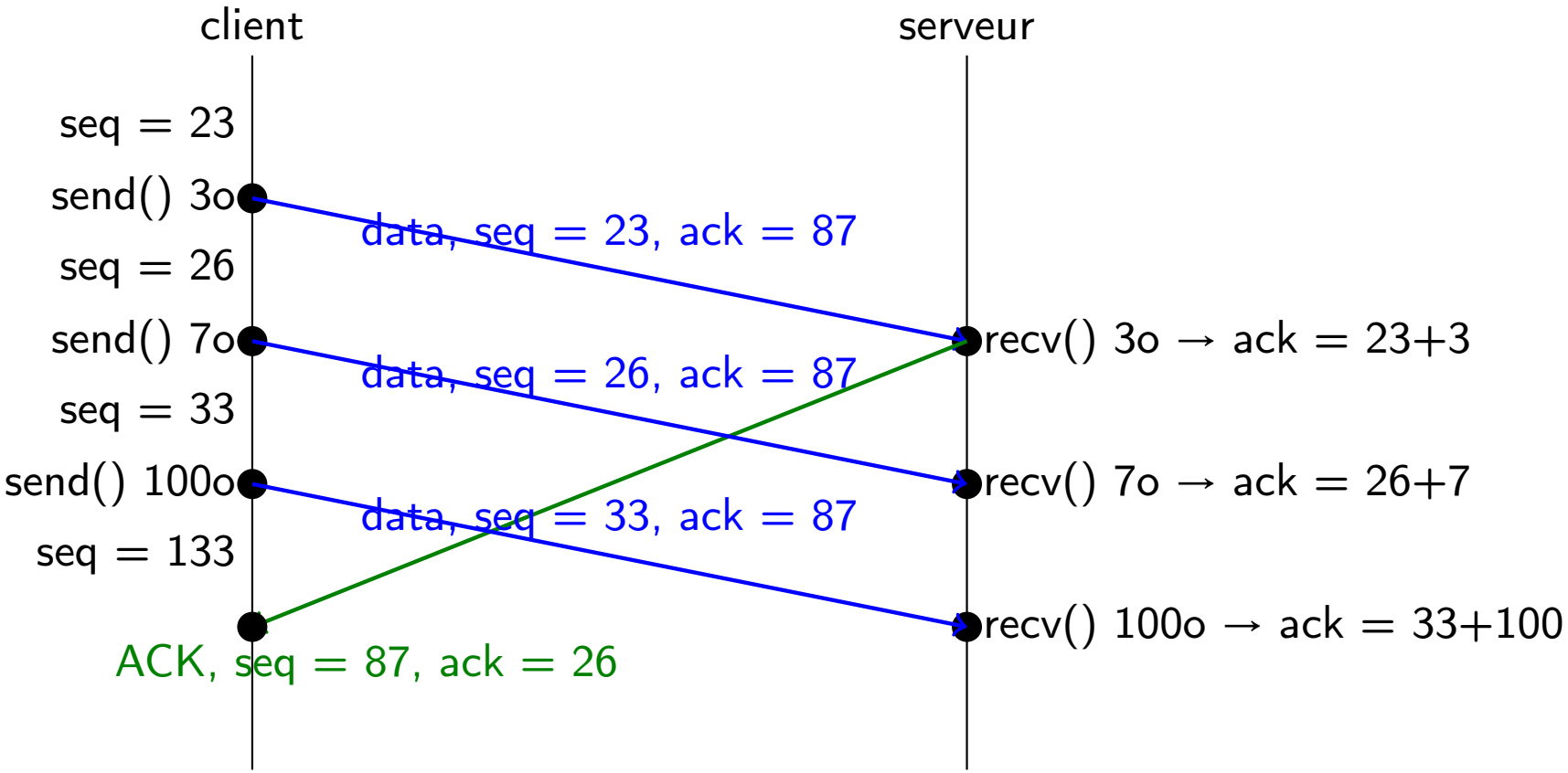
- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22



# Protocole TCP

## Envoi de données

- Le client vient d'envoyer 1 octet avec seq = 22, ack = 87
- Le serveur vient d'envoyer 1 octet avec seq = 86, ack = 22

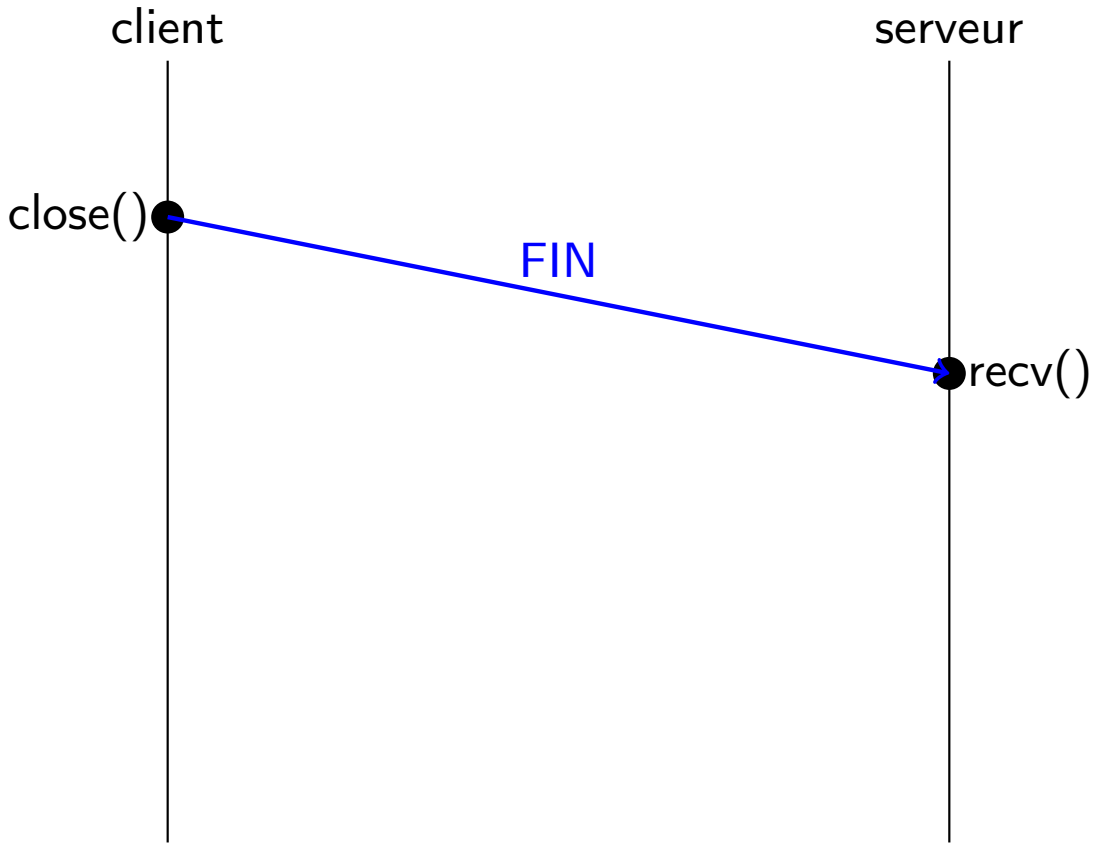


# Protocole TCP

## Fermeture de la connexion

Les deux machines doivent fermer la connexion de leur côté

- *two-way handwhake* (poignée de main bidirectionnelle)

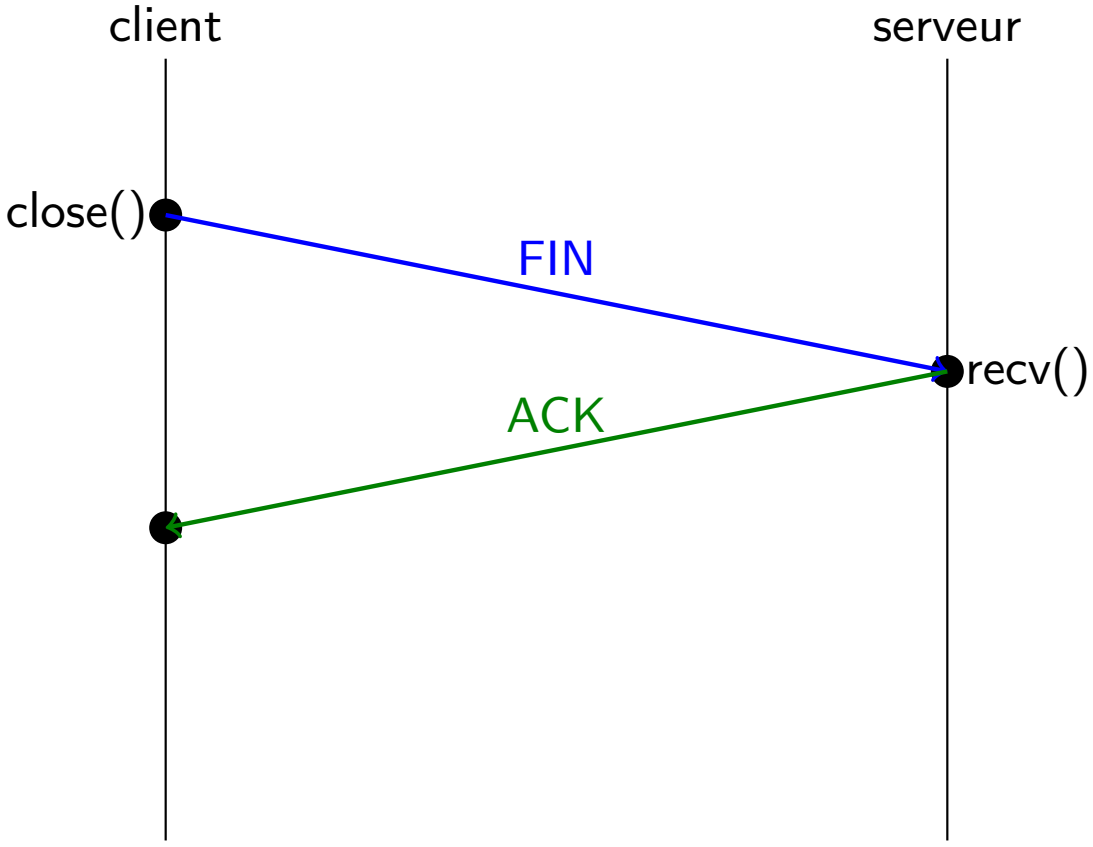


# Protocole TCP

## Fermeture de la connexion

Les deux machines doivent fermer la connexion de leur côté

- *two-way handwhake* (poignée de main bidirectionnelle)

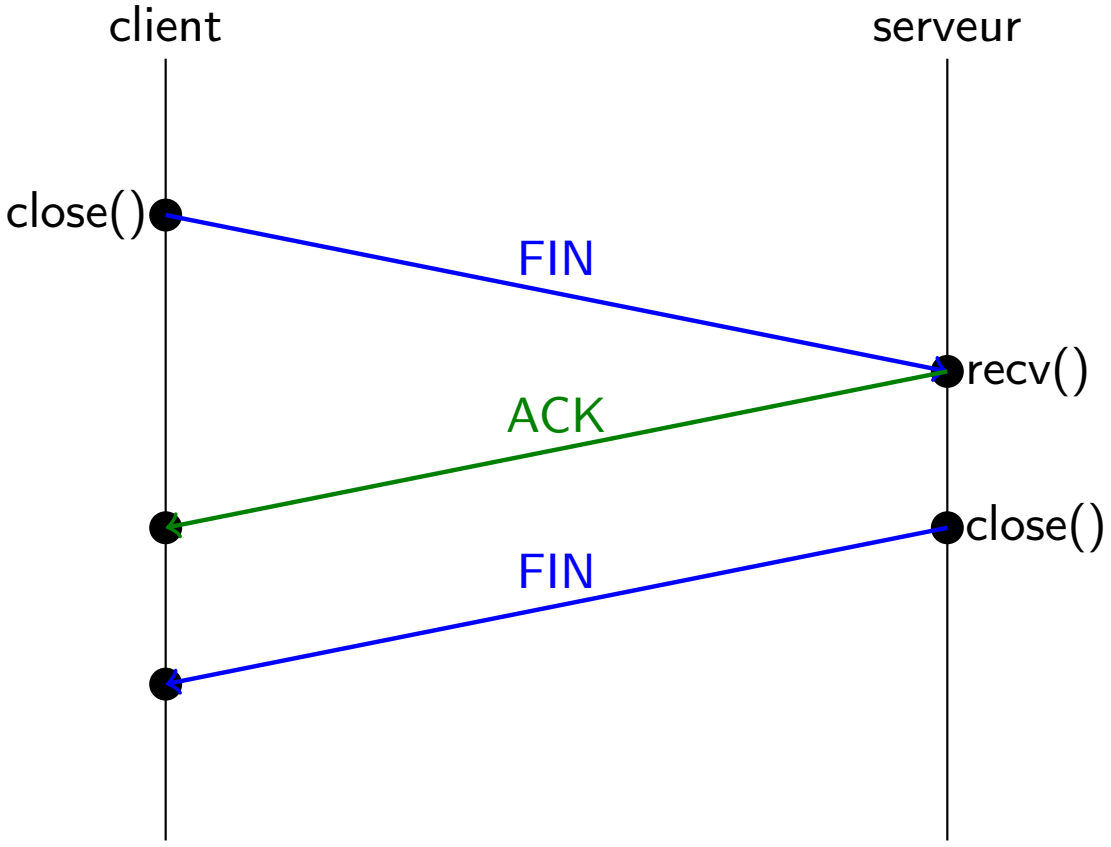


# Protocole TCP

## Fermeture de la connexion

Les deux machines doivent fermer la connexion de leur côté

- *two-way handwhake* (poignée de main bidirectionnelle)

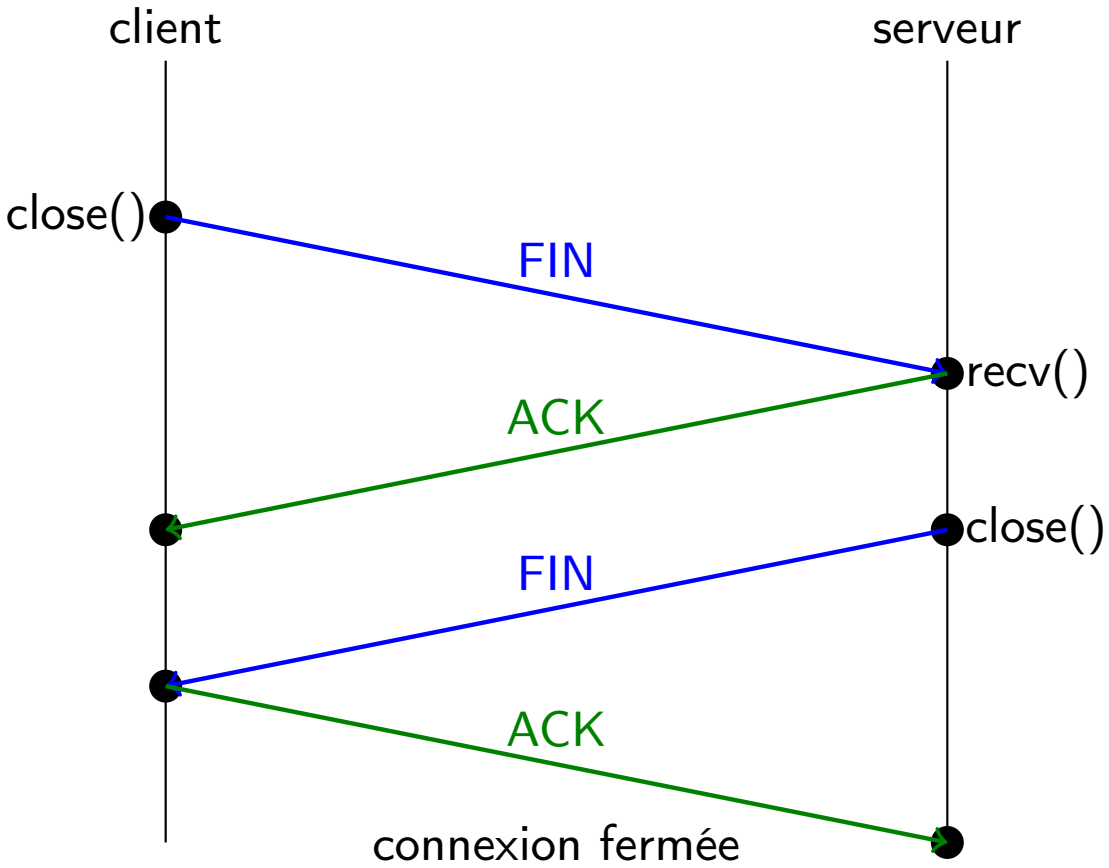


# Protocole TCP

## Fermeture de la connexion

Les deux machines doivent fermer la connexion de leur côté

- *two-way handwhake* (poignée de main bidirectionnelle)



# UDP — User Datagram Protocol

- Protocole très simple
- sans connexion
- sans acquittement
- utilisation des numéros de ports

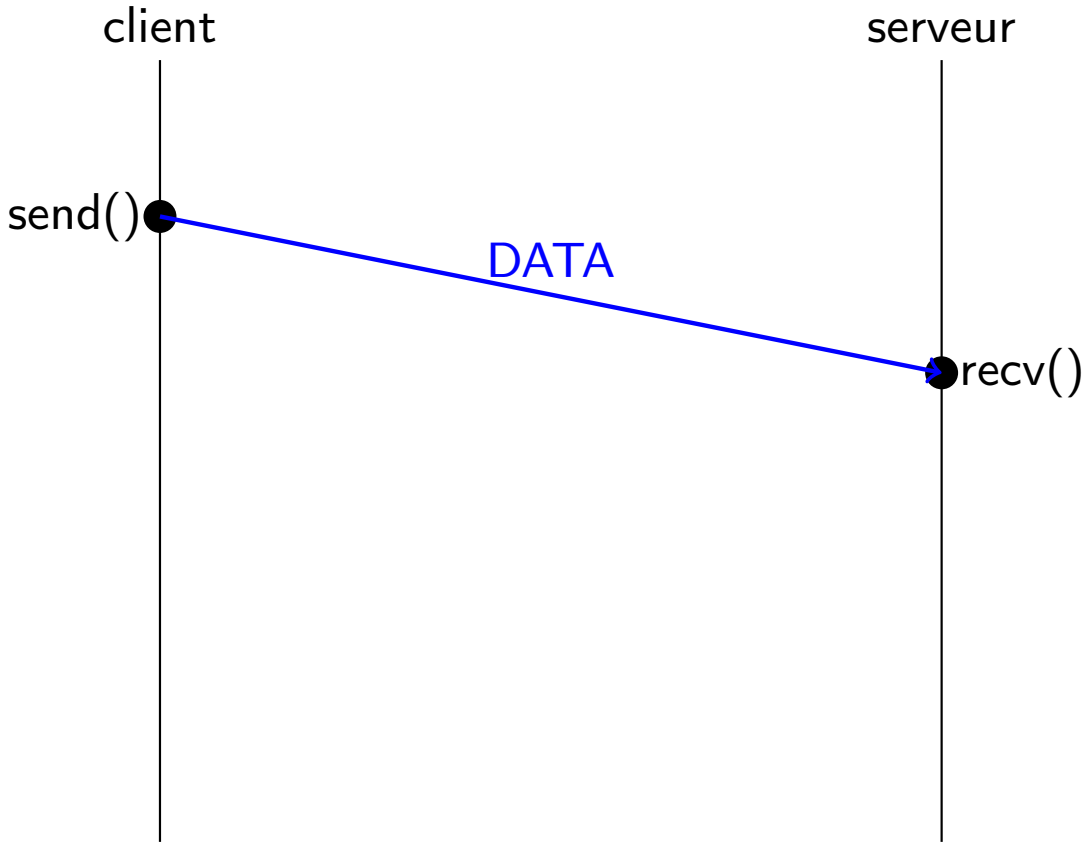
| port source (16) | port destination (16) |
|---|---|
| longueur (16) | checksum (16) |
| Données | |

# Protocole UDP

## Protocole de communications

Très simple, non fiable

- Non-connecté : pas d'établissement de connexion
- Non-fiable : pas d'acquittements

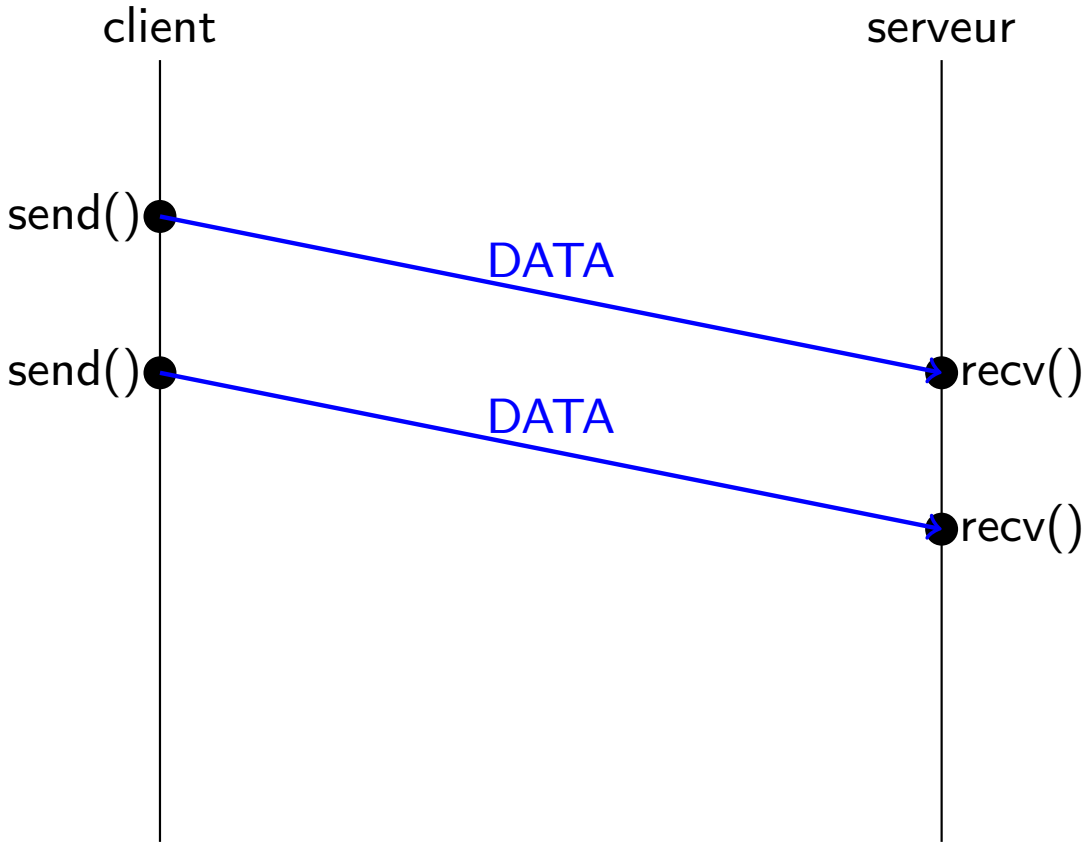


# Protocole UDP

## Protocole de communications

Très simple, non fiable

- Non-connecté : pas d'établissement de connexion
- Non-fiable : pas d'acquittements

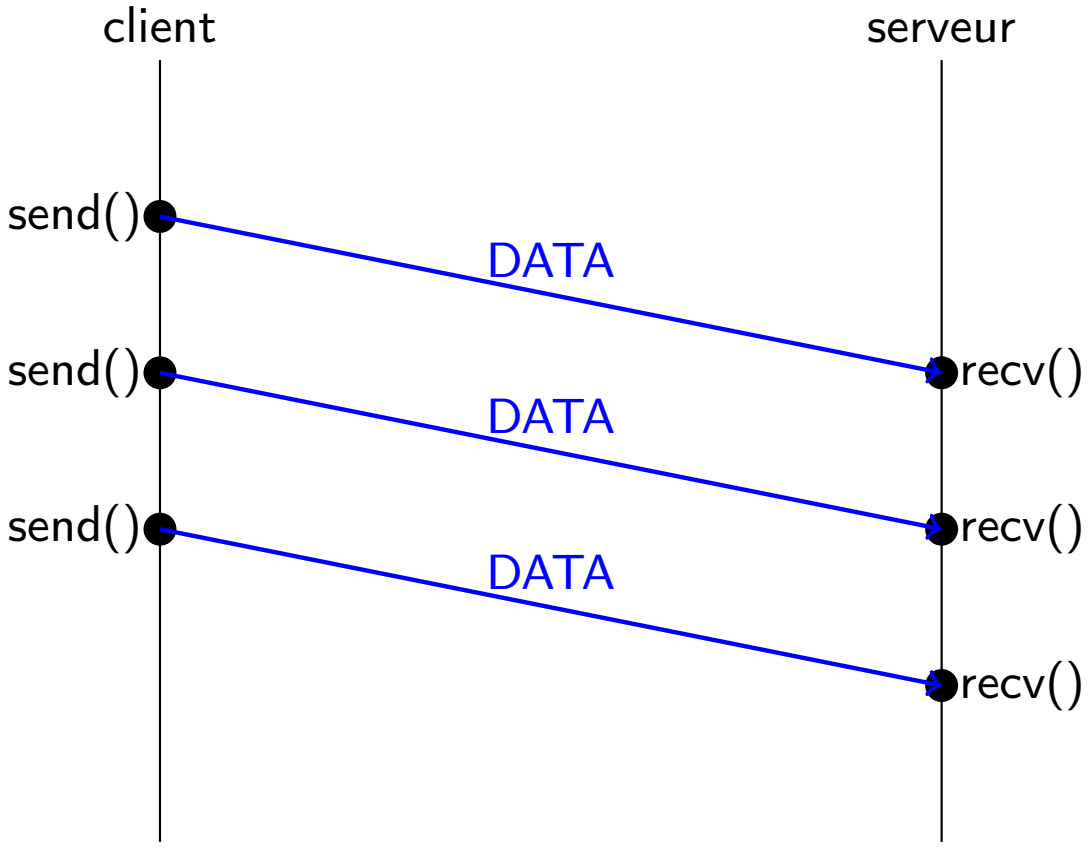


# Protocole UDP

## Protocole de communications

Très simple, non fiable

- Non-connecté : pas d'établissement de connexion
- Non-fiable : pas d'acquittements

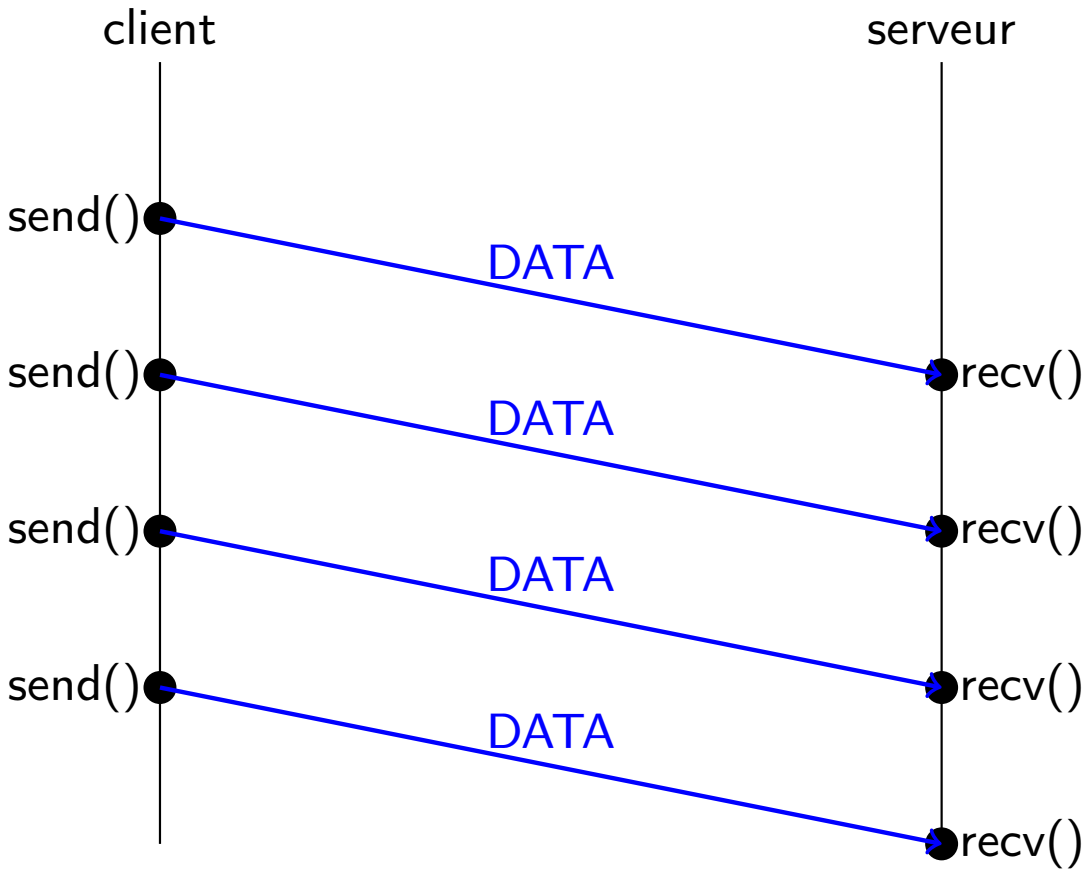


# Protocole UDP

## Protocole de communications

Très simple, non fiable

- Non-connecté : pas d'établissement de connexion
- Non-fiable : pas d'acquittements



# Plan du cours

# Composition d'une adresse IPv4

L'adresse IP identifie à la fois **la machine** et **le réseau** auquel elle appartient

- Toujours associée au **masque de sous-réseau**

Adresse composée de 4 octets

- En décimal, on les sépare par des points : 192.168.0.10

Composée de deux parties :

- La **partie réseau**
- La **partie locale**

| NET ID | HOST ID |
|---|---|

# Masque de sous-réseau

Le masque de sous-réseau **donne la taille de l'adresse réseau**

- c'est-à-dire le nombre de bits la constituant

Deux notations :

- **Notation décimale** : 4 octets séparés par des points
- **Notation CIDR** : '/' puis le nombre de bits

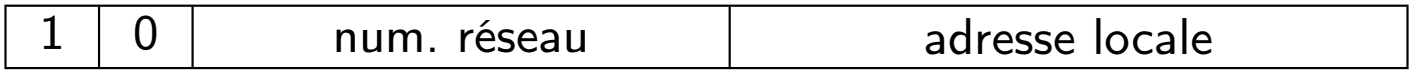
Exemples :

- Adresse réseau sur 8 bits : /8 ou 255.0.0.0
- Adresse réseau sur 20 bits : /20 ou 255.255.240.0
- Adresse réseau sur 26 bits : /26 ou 255.255.255.192

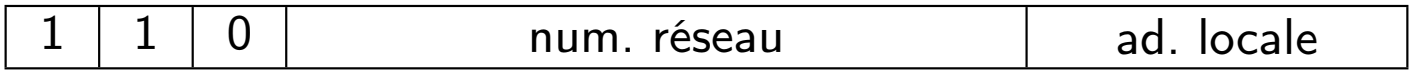
# Classes d'adresses

Classe A (grands réseaux) :

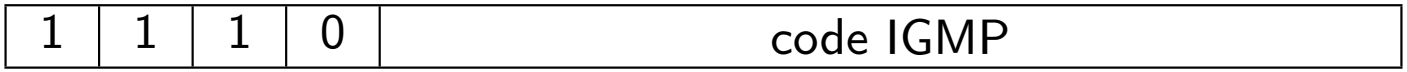

Classe B (réseaux moyens) :




Classe C (petits réseaux) :




Classe D (utilisé par IGMP) :




*Contribution de Laure Petrucci*

# Routage en fonction de l'adresse IP

Rappel : un routeur sert à **interconnecter plusieurs réseaux**

- Plusieurs interfaces réseaux, une dans chaque réseau qu'il relie

Problématique du routeur : **vers quel réseau envoyer un paquet ?**

- Le paquet arrive : pour quelle adresse est-il destiné ?
- Le routeur connaît les adresses de ses réseaux et les masques associés
- Pour chaque réseau, il applique le masque à l'adresse destination
- L'hôte de destination est-il dans ce réseau ?

Exemple :

- On a le réseau 192.168.10.0/25. L'hôte 192.168.10.130 est-il dedans ?

# Routage en fonction de l'adresse IP

Rappel : un routeur sert à **interconnecter plusieurs réseaux**

- Plusieurs interfaces réseaux, une dans chaque réseau qu'il relie

Problématique du routeur : **vers quel réseau envoyer un paquet ?**

- Le paquet arrive : pour quelle adresse est-il destiné ?
- Le routeur connaît les adresses de ses réseaux et les masques associés
- Pour chaque réseau, il applique le masque à l'adresse destination
- L'hôte de destination est-il dans ce réseau ?

Exemple :

- On a le réseau 192.168.10.0/25. L'hôte 192.168.10.130 est-il dedans ?

On applique le masque à l'adresse de l'hôte :

$$\begin{array}{rr} & 192.168.10.130 \\ \& & 255.255.255.128 \\ \hline & 192.168.10.128 \end{array}$$

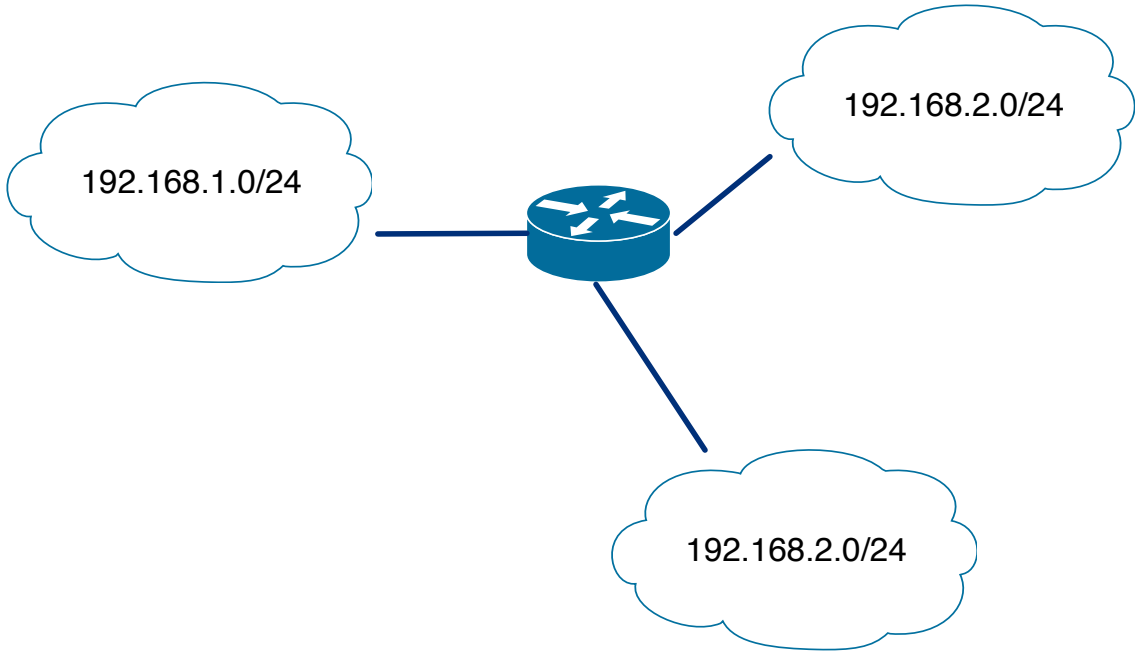
L'hôte est dans le réseau 192.168.10.128/25, donc il n'est pas dans 192.168.10.0/25.

# Plan du cours

# Plan du cours

# Interconnexion de réseaux : exemple 1

Exemple vu en TP M2106 (TP n°1) :



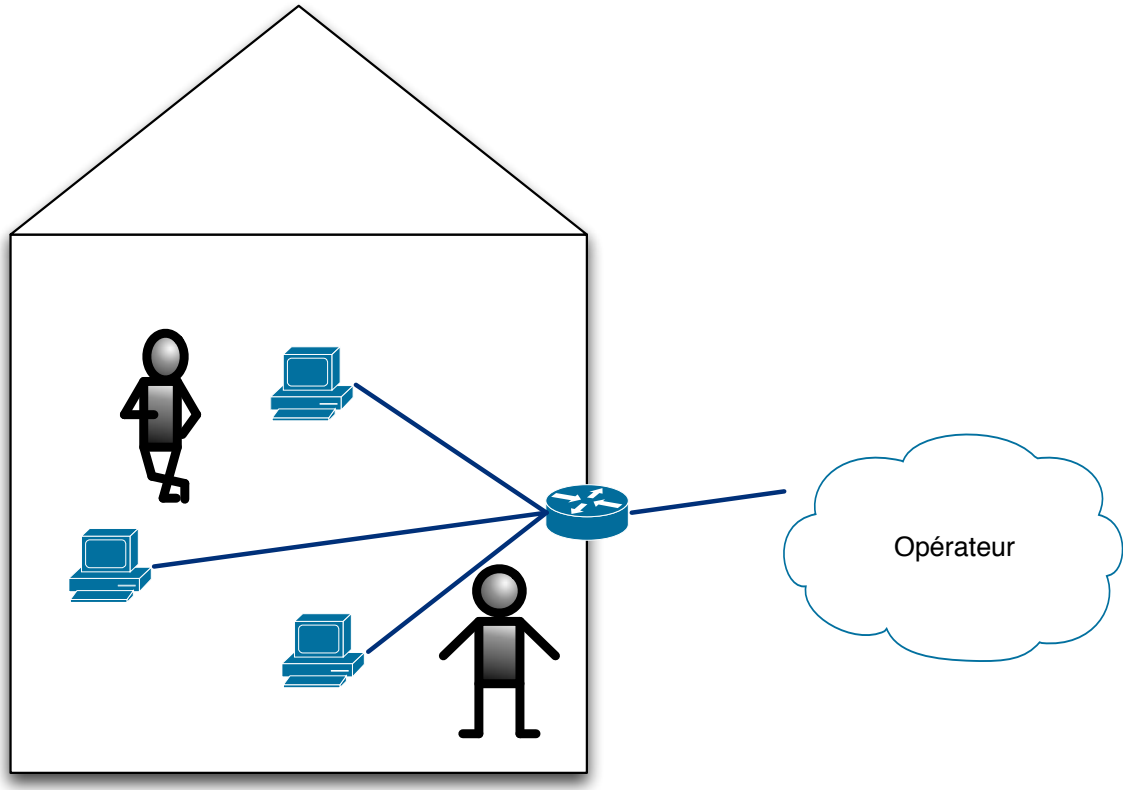

Réseaux :

- 192.168.1.0/24
- 192.168.2.0/24
- 192.168.3.0/24

Ces réseaux sont **interconnectés** par un **routeur** .

# Interconnexion de réseaux : exemple 2

Utilisation d'une box ADSL :




Réseaux :

- Réseau local de la maison, souvent 192.168.0.0/16
- Réseau de l'opérateur ADSL

La box contient un **routeur** qui assure l' **interconnexion** entre le réseau de la maison et le réseau de l'opérateur.

# Utilisation d'un routeur

Un routeur assure l' **interconnexion entre plusieurs réseaux**

- Il a plusieurs interfaces réseaux
- Une interface dans chaque réseau qu'il relie
- Il achemine les paquets d'un réseau à l'autre

Exemple :

- 2 réseaux : 192.168.1.0/24 et 192.168.2.0/24
- interconnectés par un routeur
- Machine M1 192.168.1.82/24 envoie un message à M2 192.168.2.231/24
- Les machines **ne sont pas dans le même réseau** : le message **transite par le routeur**

# Utilisation d'un routeur

Un routeur assure l' **interconnexion entre plusieurs réseaux**

- Il a plusieurs interfaces réseaux
- Une interface dans chaque réseau qu'il relie
- Il achemine les paquets d'un réseau à l'autre

Exemple :

- 2 réseaux : 192.168.1.0/24 et 192.168.2.0/24
- interconnectés par un routeur
- Machine M1 192.168.1.82/24 envoie un message à M2 192.168.2.231/24
- Les machines **ne sont pas dans le même réseau** : le message **transite par le routeur**
- Concrètement :
  - M1 détermine que M2 n'est pas dans le même réseau qu'elle
  - Le routeur relie M1 au réseau de M2
  - M1 envoie le message au routeur
  - Le routeur envoie le message à M2

# Le routage : définition

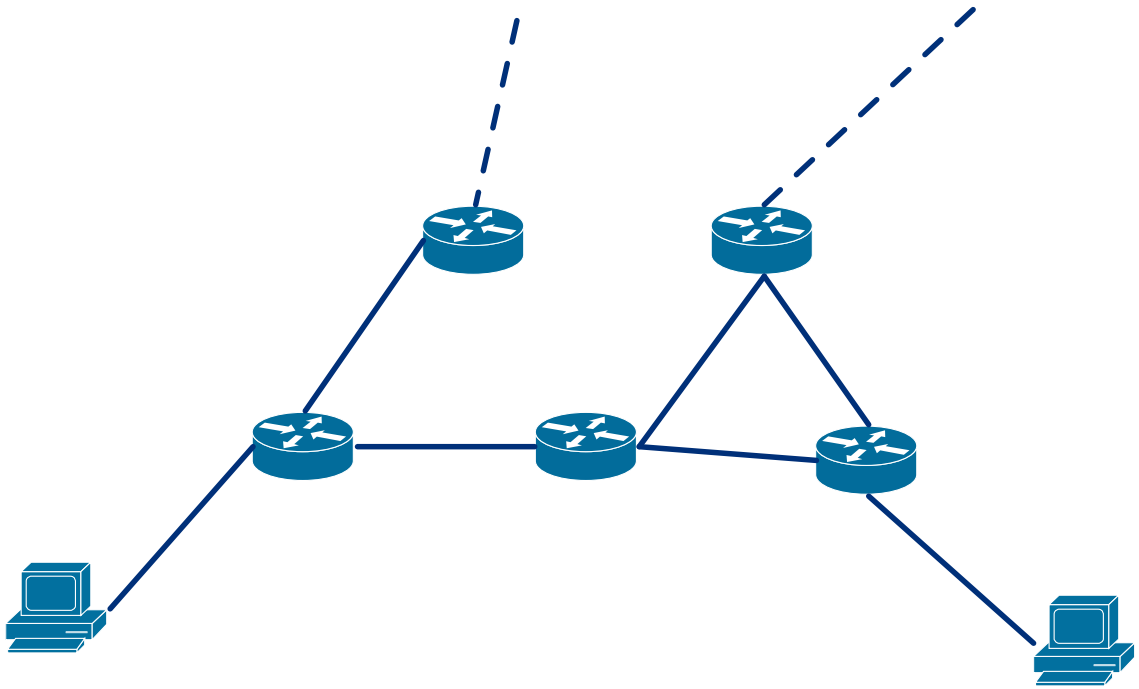
**Définition (Wikipedia)**

Le routage est le mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires.

# Le routage : définition

## Définition (Wikipedia)

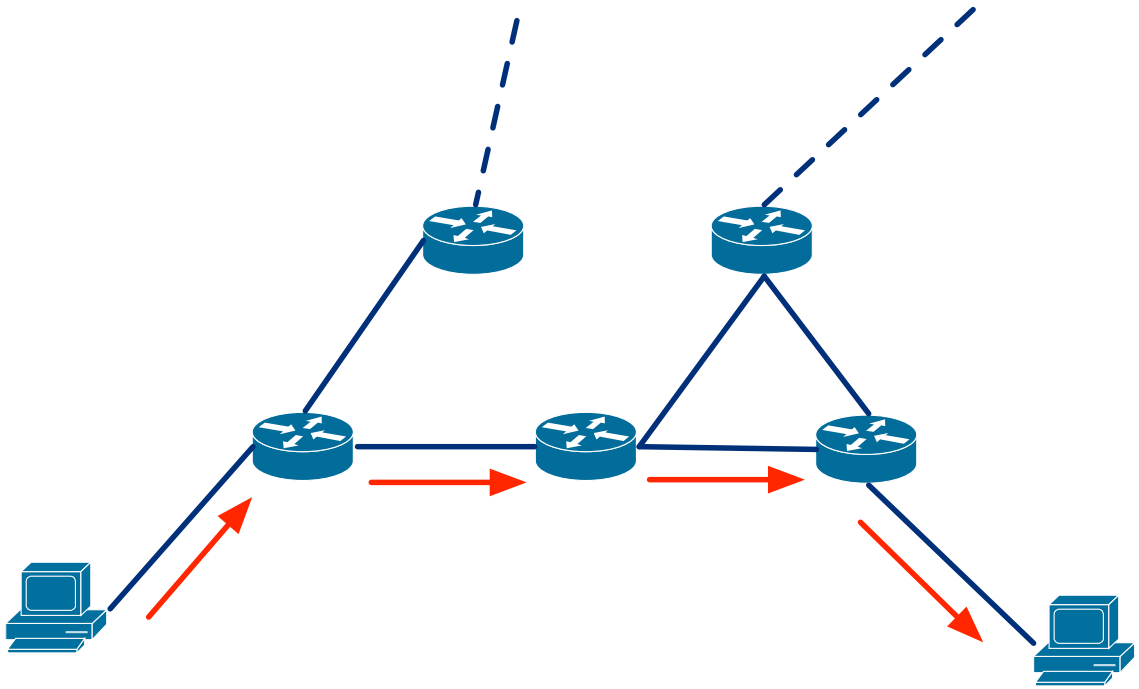
Le routage est le mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires.



# Le routage : définition

## Définition (Wikipedia)

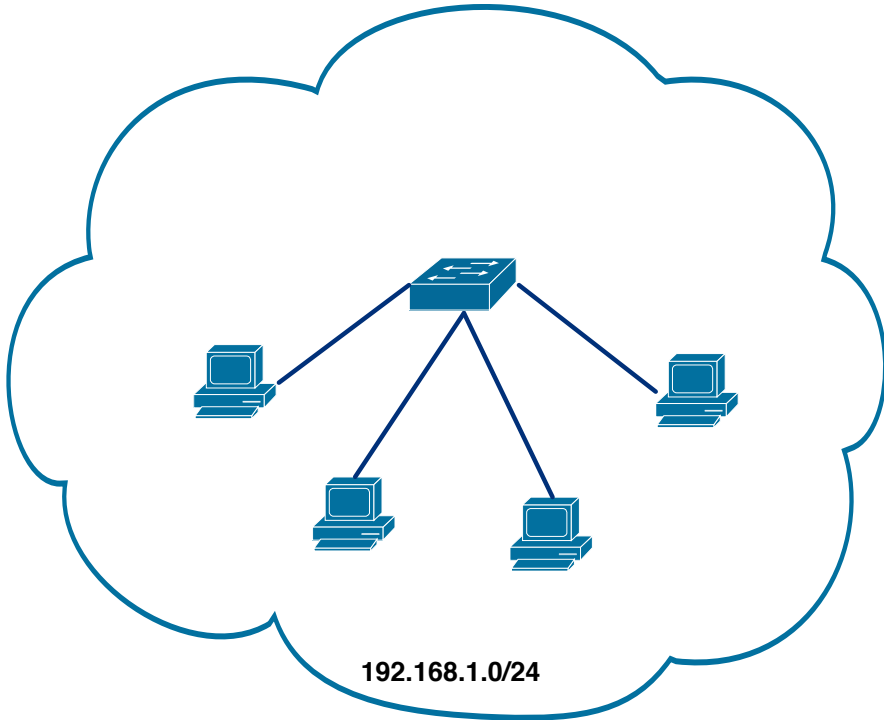
Le routage est le mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires.



# Plan du cours

# Passerelle d'un réseau local

Considérons un réseau local :




Éléments de ce réseau :

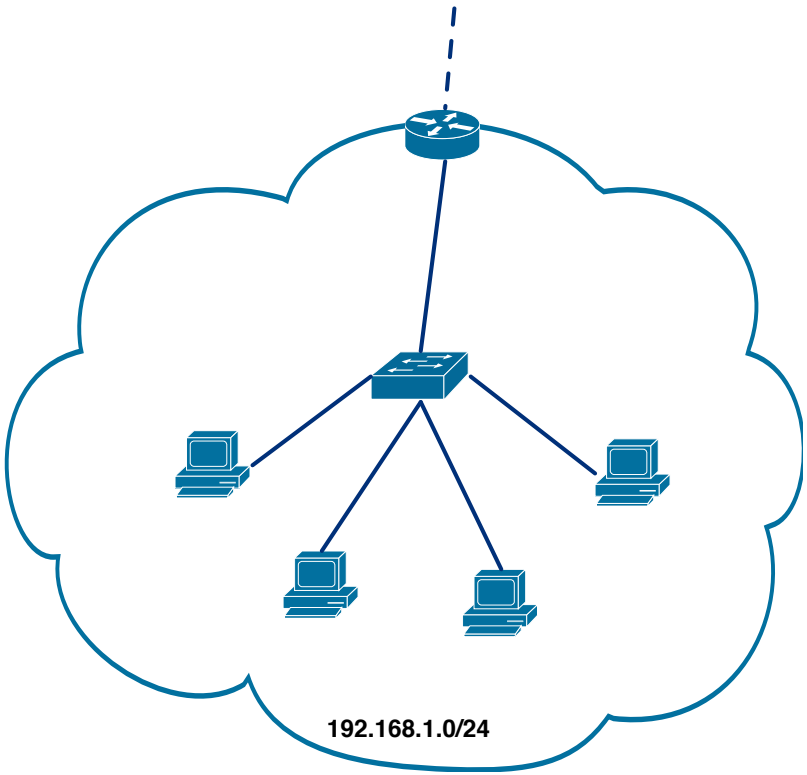
- 4 machines
- 1 switch

Les machines du réseau 192.168.1.0/24 :

- peuvent communiquer entre elles
- ne peuvent pas sortir du réseau

# Passerelle d'un réseau local

Pour sortir d'un réseau local : **utilisation d'une passerelle**




Deux possibilités de communications :

- Communication à l'intérieur du réseau : envoi du message directement au destinataire
- Communication vers l'extérieur du réseau : envoi du message au routeur qui l'achemine

# Passerelle d'un réseau local

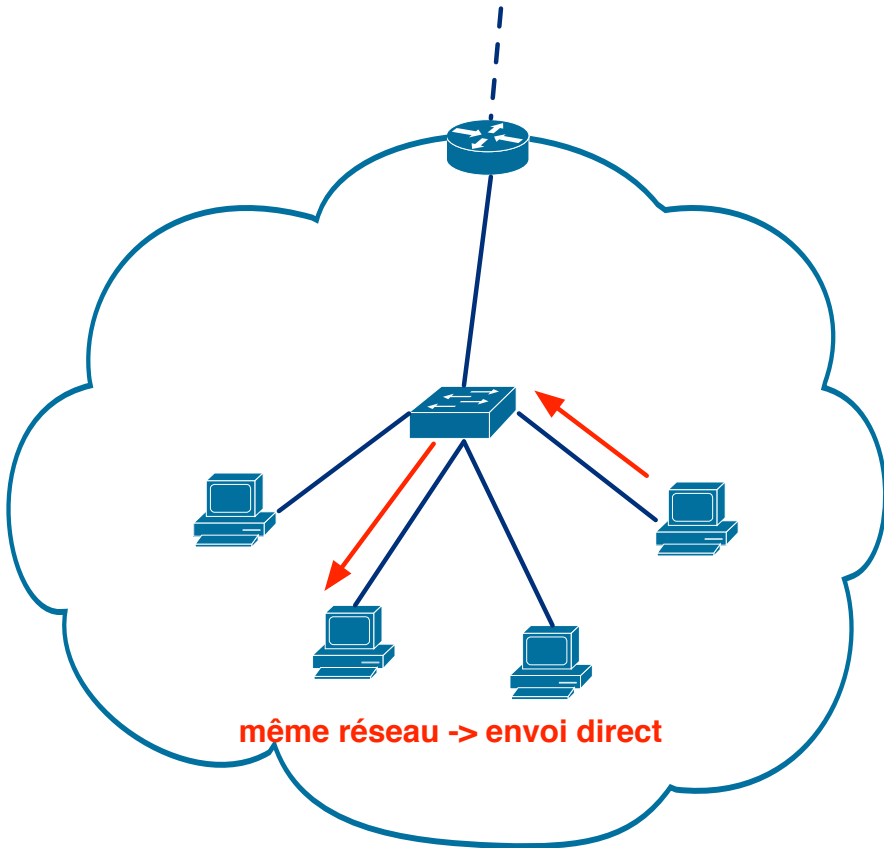
Rôle du routeur en sortie d'un **réseau local** : **connecter le réseau local avec l'extérieur**

- C'est une **passerelle**
- Les paquets **sortants** passent par le routeur
- Les paquets **entrants** passent par le routeur

# Passerelle d'un réseau local

Rôle du routeur en sortie d'un **réseau local** : **connecter le réseau local avec l'extérieur**

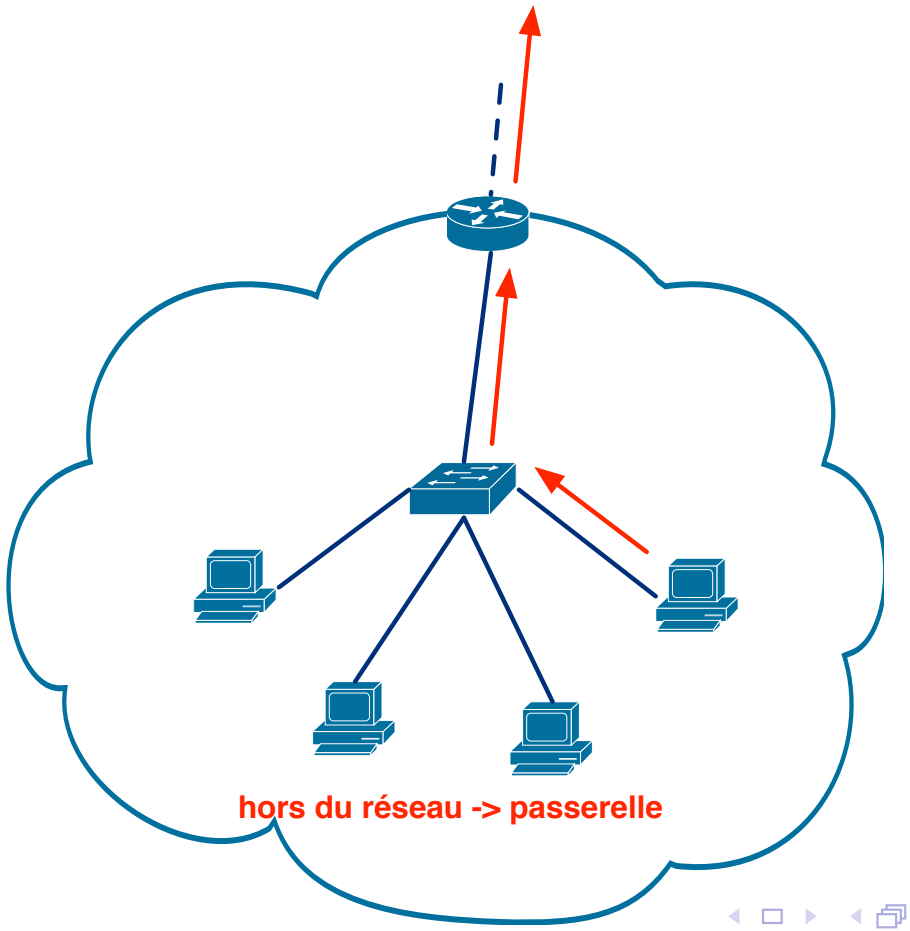
- C'est une **passerelle**
- Les paquets **sortants** passent par le routeur
- Les paquets **entrants** passent par le routeur



# Passerelle d'un réseau local

Rôle du routeur en sortie d'un **réseau local** : **connecter le réseau local avec l'extérieur**

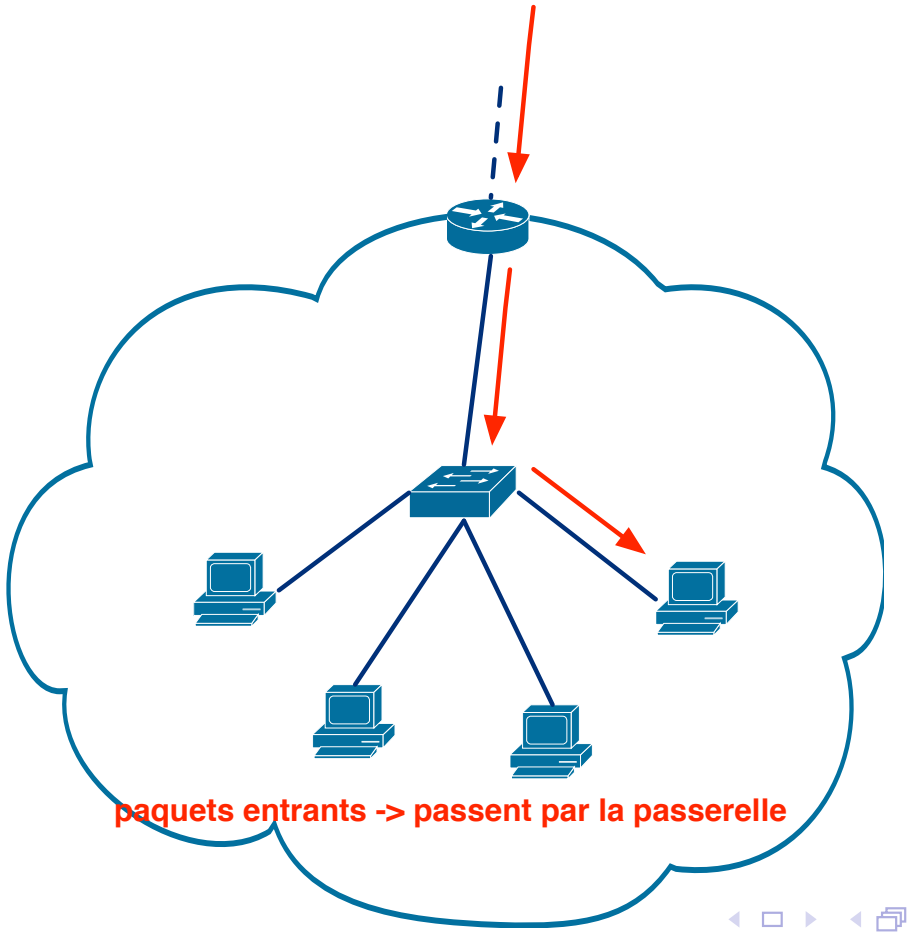
- C'est une **passerelle**
- Les paquets **sortants** passent par le routeur
- Les paquets **entrants** passent par le routeur



# Passerelle d'un réseau local

Rôle du routeur en sortie d'un **réseau local** : **connecter le réseau local avec l'extérieur**

- C'est une **passerelle**
- Les paquets **sortants** passent par le routeur
- Les paquets **entrants** passent par le routeur



# Décision de routage dans un réseau local

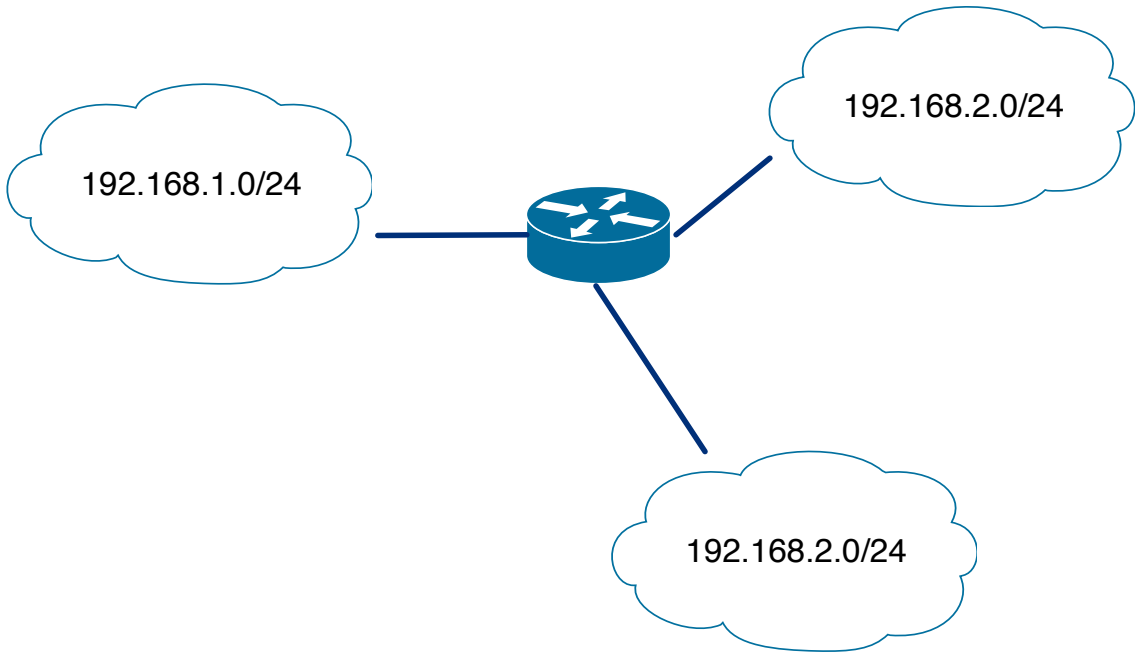
Envoi d'un paquet → décision de routage

- Pour qui le paquet est-il destiné ?
- Le destinataire *fait-il partie du même réseau que la source* ?
  - Oui → envoi direct
  - Non → envoi à la passerelle, qui se charge de *router le paquet*

# Plan du cours

# Routeur entre plusieurs réseaux 1/3

Retour sur notre premier exemple :




R1 interconnecte 3 réseaux :

- 192.168.1.0/24
- 192.168.2.0/24
- 192.168.3.0/24

# Routeur entre plusieurs réseaux 2/3

Interfaces réseau du routeur : **une dans chaque réseau !**

Exemple :

- 192.168.1.254/24
- 192.168.2.254/24
- 192.168.3.254/24

Conséquence au niveau IP : le routeur **appartient à chacun des réseaux** qu'il interconnecte.

Exemple d'application:

- 192.168.1.1/24 peut envoyer un message à 192.168.1.254/24
- 192.168.2.254/24 peut envoyer un message à 192.168.2.1/24
- Le message peut être acheminé de 192.168.1.1/24 à 192.168.2.1/24 **en passant par le routeur**

# Routeur entre plusieurs réseaux 3/3

**Décision de routage** au niveau du routeur :

- Un paquet arrive → à qui faut-il l'envoyer ?

Méthode :

- On regarde réseau par réseau **si le destinataire en fait partie**
- Importance du **masque de sous-réseau** !

# Routeur entre plusieurs réseaux 3/3

**Décision de routage** au niveau du routeur :

- Un paquet arrive → à qui faut-il l'envoyer ?

Méthode :

- On regarde réseau par réseau **si le destinataire en fait partie**
- Importance du **masque de sous-réseau** !

Exemple : un routeur est connecté à 4 sous-réseau :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/25
- LAN3 = 192.168.2.128/26
- LAN4 = 192.168.2.192/26

Un paquet arrive pour 192.168.2.152. Sur quel sous-réseau l'envoie-t-on ?

# Routeur entre plusieurs réseaux 3/3

**Décision de routage** au niveau du routeur :

- Un paquet arrive → à qui faut-il l'envoyer ?

Méthode :

- On regarde réseau par réseau **si le destinataire en fait partie**
- Importance du **masque de sous-réseau** !

Exemple : un routeur est connecté à 4 sous-réseau :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/25
- LAN3 = 192.168.2.128/26
- LAN4 = 192.168.2.192/26

Un paquet arrive pour 192.168.2.152. Sur quel sous-réseau l'envoie-t-on ?

- 192.168.2.152/24 → réseau 192.168.2.0/24 → on n'envoie pas sur LAN1

# Routeur entre plusieurs réseaux 3/3

**Décision de routage** au niveau du routeur :

- Un paquet arrive → à qui faut-il l'envoyer ?

Méthode :

- On regarde réseau par réseau **si le destinataire en fait partie**
- Importance du **masque de sous-réseau** !

Exemple : un routeur est connecté à 4 sous-réseau :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/25
- LAN3 = 192.168.2.128/26
- LAN4 = 192.168.2.192/26

Un paquet arrive pour 192.168.2.152. Sur quel sous-réseau l'envoie-t-on ?

- 192.168.2.152/24 → réseau 192.168.2.0/24 → on n'envoie pas sur LAN1
- 192.168.2.152/25 → réseau 192.168.2.128/25 → on n'envoie pas sur LAN2

# Routeur entre plusieurs réseaux 3/3

**Décision de routage** au niveau du routeur :

- Un paquet arrive → à qui faut-il l'envoyer ?

Méthode :

- On regarde réseau par réseau **si le destinataire en fait partie**
- Importance du **masque de sous-réseau** !

Exemple : un routeur est connecté à 4 sous-réseau :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/25
- LAN3 = 192.168.2.128/26
- LAN4 = 192.168.2.192/26

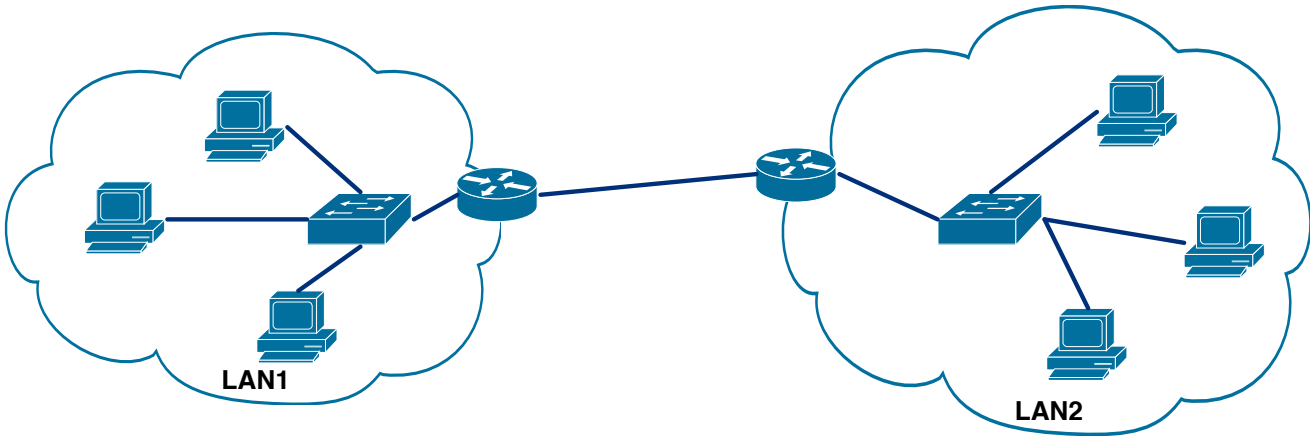
Un paquet arrive pour 192.168.2.152. Sur quel sous-réseau l'envoie-t-on ?

- 192.168.2.152/24 → réseau 192.168.2.0/24 → on n'envoie pas sur LAN1
- 192.168.2.152/25 → réseau 192.168.2.128/25 → on n'envoie pas sur LAN2
- 192.168.2.152/26 → réseau 192.168.2.128/26 → on envoie sur LAN3

# Plan du cours

# Routage et interconnexion 1/2

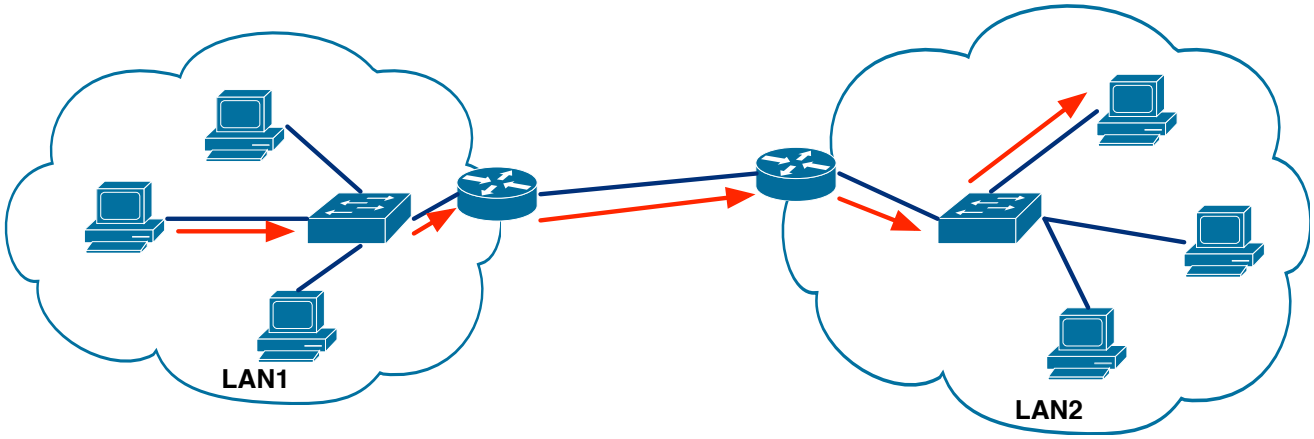
Interconnexion de plusieurs réseaux : on relie les routeurs entre eux




Les messages transitent **entre les routeurs**

# Routage et interconnexion 2/2

Exemple : envoi d'un message de LAN1 à LAN2




Chemin pris par le message :

- Le destinataire est extérieur à LAN1 : envoi au routeur (passerelle) de LAN1
- Le destinataire est dans LAN2 : envoi au routeur (passerelle) de LAN2
- Envoi au destinataire

# Réseau d'interconnexion : liaison directe 1/2

Concrètement, comment interconnecter les routeurs ?

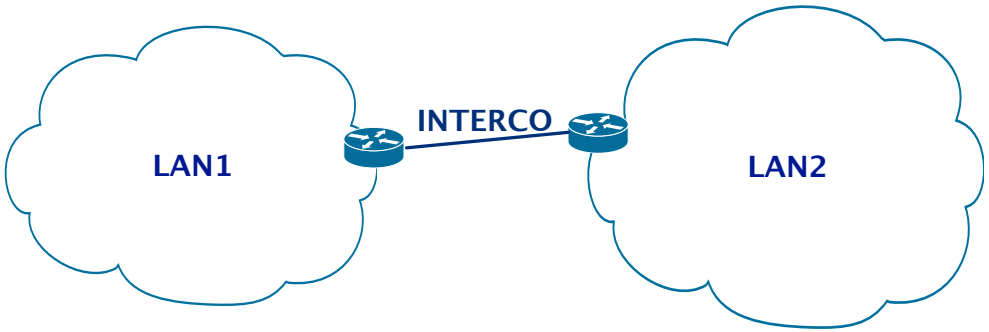
- Physiquement : on assure la continuité du réseau physique (câble direct, câble + switch...)
- Besoin d'un adressage IP !

On crée un petit **sous-réseau d'interconnexion** entre les routeurs.

Les routeurs assurent alors le lien entre :

- Les réseaux auquels ils sont reliés d'une part
- Le réseau d'interconnexion auquel ils appartiennent d'autre part

Le réseau d'interconnexion sert alors aux routeurs à **communiquer entre eux** pour relayer les messages.



# Réseau d'interconnexion : liaison directe 2/2

Exemple : On a deux réseaux :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/24

Chacun a un routeur de sortie :

- R1 → eth0 = 192.168.1.254/24
- R2 → eth0 = 192.168.2.254/24

On relie eth1 de R1 avec eth1 de R2 → quelles adresses IP utiliser ?

# Réseau d'interconnexion : liaison directe 2/2

Exemple : On a deux réseaux :

- LAN1 = 192.168.1.0/24
- LAN2 = 192.168.2.0/24

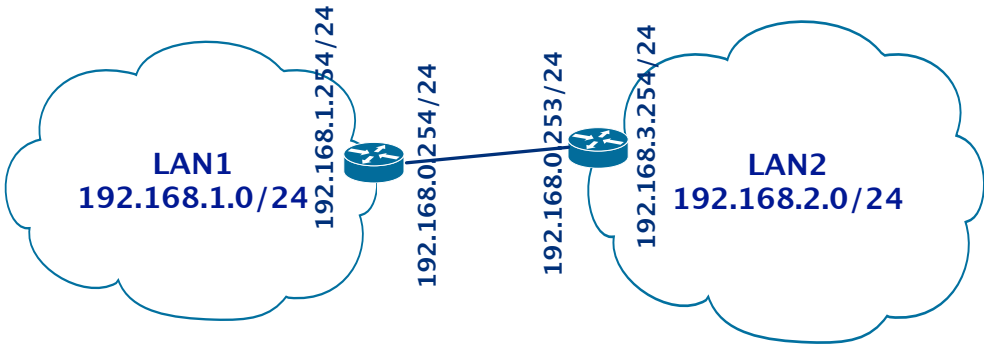
Chacun a un routeur de sortie :

- R1 → eth0 = 192.168.1.254/24
- R2 → eth0 = 192.168.2.254/24

On relie eth1 de R1 avec eth1 de R2 → quelles adresses IP utiliser ?

On crée le sous-réseau 192.168.0.0/24 pour relier R1 et R2

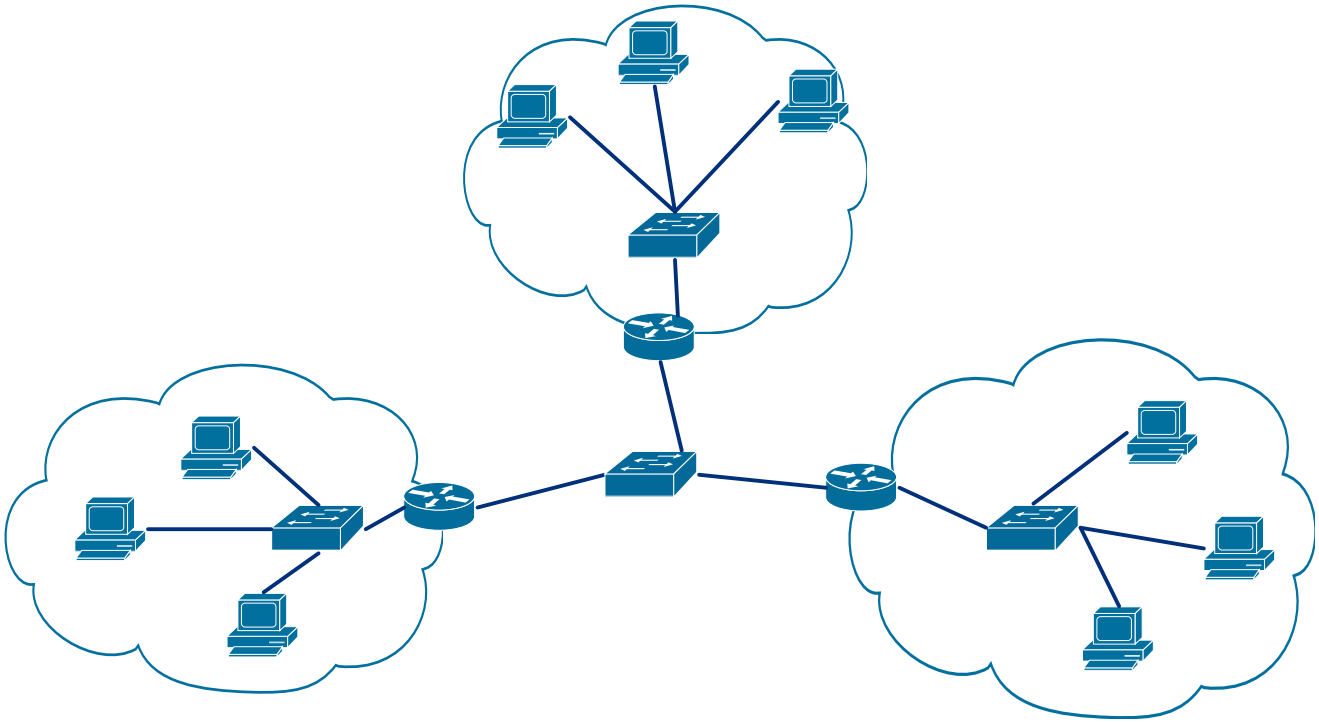
- R1 → eth1 = 192.168.0.254/24
- R2 → eth1 = 192.168.0.253/24



# Réseau d'interconnexion : backbone

Épine dorsale du réseau : routeurs connectés entre eux

- c'est le **backbone**



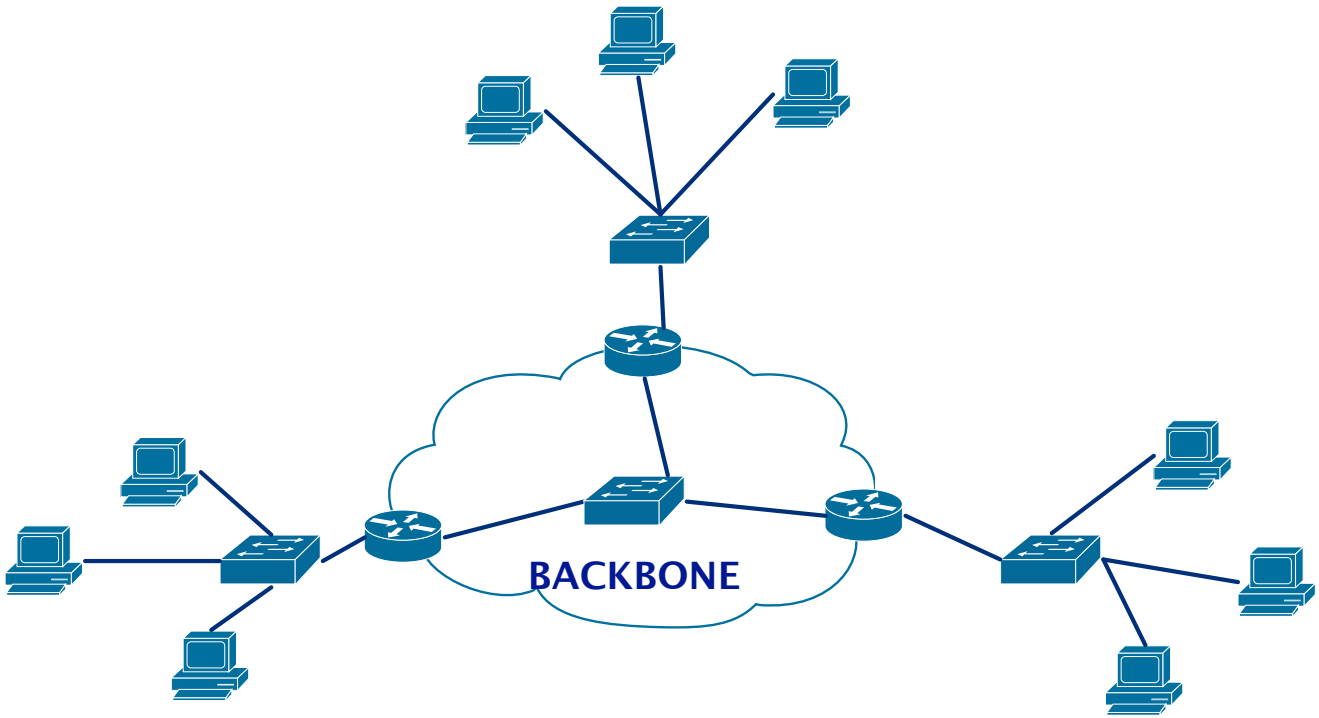
On a un **réseau spécifique pour les routeurs**

- Les passerelles des réseaux assurent l'interconnexion entre les réseaux locaux et le backbone
- Les paquets transitent dans le backbone pour être acheminés entre les réseaux locaux

# Réseau d'interconnexion : backbone

Épine dorsale du réseau : routeurs connectés entre eux

- c'est le **backbone**

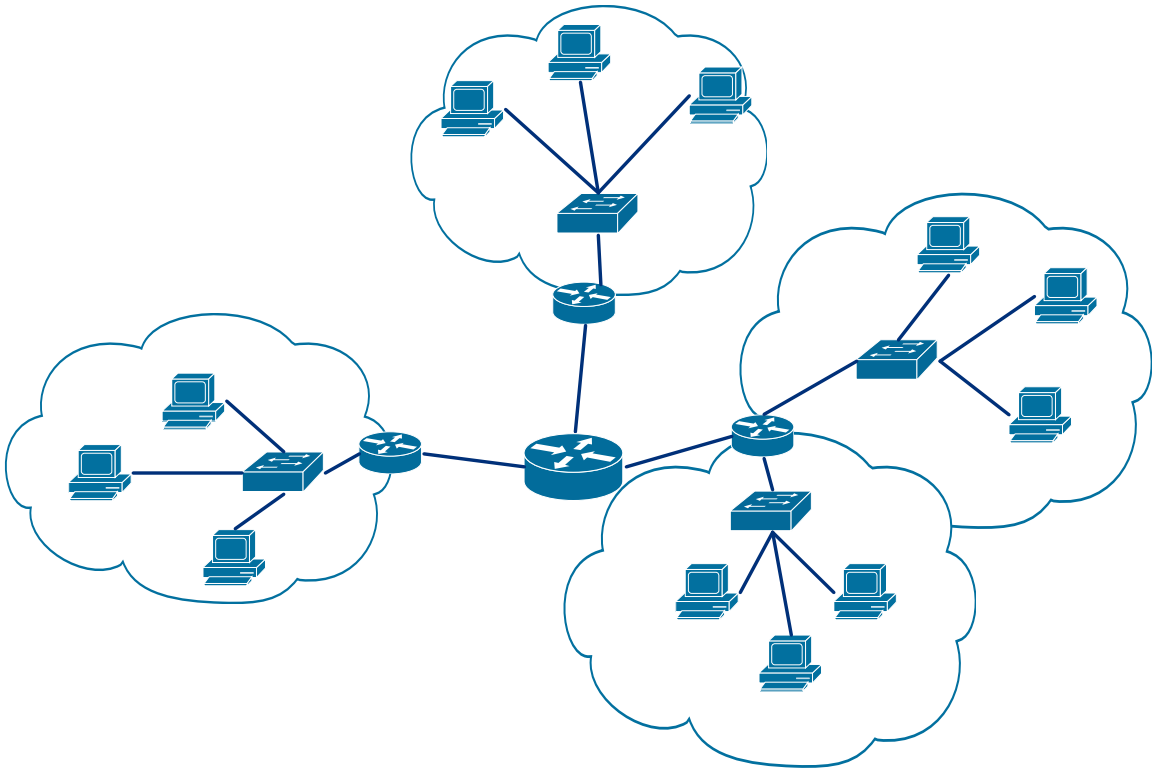



On a un **réseau spécifique pour les routeurs**

- Les passerelles des réseaux assurent l'interconnexion entre les réseaux locaux et le backbone
- Les paquets transitent dans le backbone pour être acheminés entre les réseaux locaux

# Réseau d'interconnexion : routage hiérarchique

Les routeurs sont reliés à un **gros routeur central**



Les paquets à acheminer entre les routeurs passent par le routeur central

- Les passerelles des réseaux font le lien entre les réseaux locaux et le routeur central
- Les paquets transitent par ce routeur central qui prend la décision de routage

# Décision de routage dans un réseau d'interconnexion

Besoin de connaître (presque) toute la topologie du réseau

- Problématique : à qui envoyer ce paquet qui vient de m'arriver ?
- Question sous-jacente : où est le réseau auquel il appartient ?
  - Un des réseaux auxquels le routeur est relié : envoi direct
  - Sinon : besoin de faire transiter par un autre routeur
  - → besoin de **savoir à quels réseaux sont reliés les autres routeurs**

Configuration d'un routeur = **connaissance du prochain saut pour atteindre tous les autres réseaux**

- À qui doit-on envoyer un paquet si on veut qu'il arrive à chaque réseau

# Plan du cours

# Table de routage

La **table de routage** rassemble les infos permettant de **prendre la décision de routage**

- Quels réseaux sont accessibles
- Par quel moyen ils sont accessibles

Quand le routeur reçoit un pqauet, il regarde dans la table de routage

- Réseau par réseau, il regarde si le destinataire est dedans
- Une fois qu'il a trouvé le bon, il l'envoie sur l'interface concernée
- Si il ne sait pas où l'envoyer, le paquet est perdu (*no route to host*)

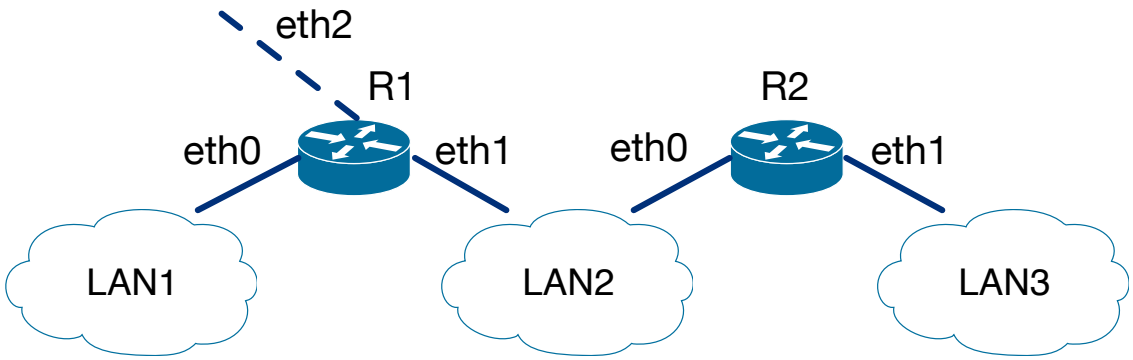
Pour les réseaux structurés, il peut s'agit du **prochain saut vers la destinataion**

# Table de routage

Que trouve-t-on dans une table de routage ?

- **Pour chaque réseau auquel le routeur est directement relié** : l'adresse et le masque du réseau, l'interface à lqauelle il est connecté
- **Pour chaque réseau distant connu** : l'adresse du prochain saut pour l'atteindre, l'interface par laquelle on va l'atteindre
  - Ce prochain saut *doit être* dans un réseau relié directement au routeur
- Éventuellement : une **route par défaut** , pour tous les destinataires qu'on ne trouve pas dans les réseaux connus

# Table de routage : exemple



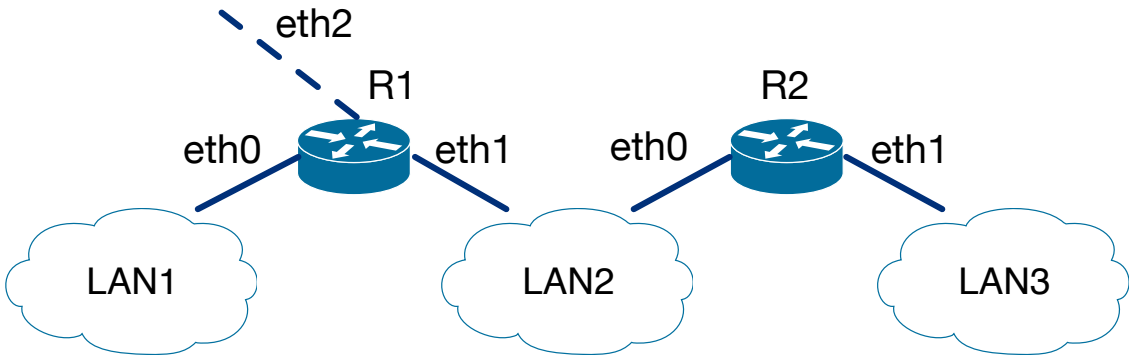

R1 est relié à LAN1 et LAN2 :

- LAN1 sur eth0
- LAN2 sur eth1
- Le reste du monde sur eth2

R2 est relié à LAN2 et LAN3 :

- LAN2 sur eth0
- LAN3 sur eth1

# Table de routage : exemple




R1 est relié à LAN1 et LAN2 :

- LAN1 sur eth0
- LAN2 sur eth1
- Le reste du monde sur eth2

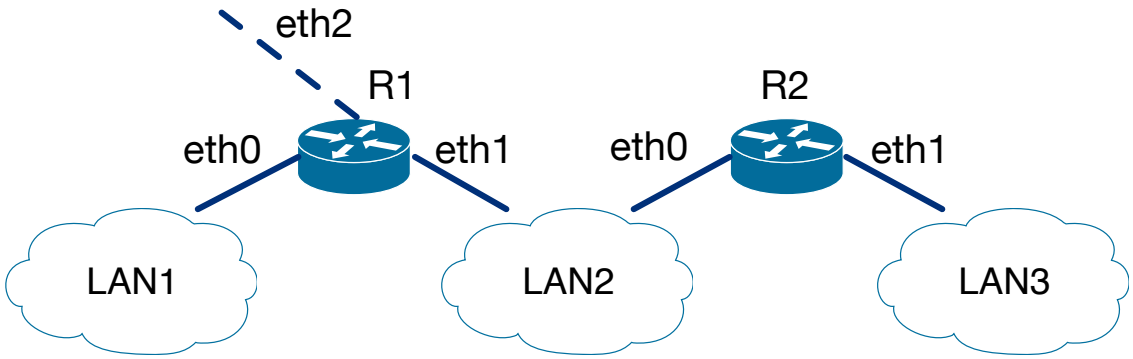
R2 est relié à LAN2 et LAN3 :

- LAN2 sur eth0
- LAN3 sur eth1

Depuis R1 :

- On atteint LAN1 directement sur eth0
- On atteint LAN2 directement sur eth1
- On atteint l'extérieur directement sur eth2
- On doit passer par R2 pour atteindre LAN3 sur eth1

# Table de routage : exemple




R1 est relié à LAN1 et LAN2 :

- LAN1 sur eth0
- LAN2 sur eth1
- Le reste du monde sur eth2

R2 est relié à LAN2 et LAN3 :

- LAN2 sur eth0
- LAN3 sur eth1

Depuis R2 :

- On atteint LAN2 directement sur eth0
- On atteint LAN3 directement sur eth1
- On doit passer par R1 pour atteindre LAN1 sur eth0
- On doit passer par R1 pour atteindre l'extérieur sur eth0

# Plan du cours

# Configurer les tables de routage

Sous Linux : `route`

- **Destination** : réseau (plus rarement hôte) de destinataion
- **Gateway** : passerelle par laquelle faire transiter les paquets
- **Netmask** : masque de sous-réseau
- **Flags** : Drapeaux sur la route. U = ligne effective, G = passerelle, H = hôte
- **Interface** : interface réseau sur laquelle envoyer les paquets

# Configurer les tables de routage

Sous Linux : `route`

- **Destination** : réseau (plus rarement hôte) de destinataion
- **Gateway** : passerelle par laquelle faire transiter les paquets
- **Netmask** : masque de sous-réseau
- **Flags** : Drapeaux sur la route. U = ligne effective, G = passerelle, H = hôte
- **Interface** : interface réseau sur laquelle envoyer les paquets

Table de routage de R1 :

| Destination | Gateway | Netmask | Flags | Interface |
|---|---|---|---|---|
| 192.168.10.0 | | 255.255.255.0 | U | eth0 |
| 192.168.20.0 | | 255.255.255.0 | U | eth1 |
| 192.168.30.0 | 192.168.20.254 | 255.255.255.0 | UG | eth1 |
| default | 10.0.2.2 | 0.0.0.0 | UG | eth2 |

# Configurer les tables de routage

Sous Linux : `route`

- **Destination** : réseau (plus rarement hôte) de destinataion
- **Gateway** : passerelle par laquelle faire transiter les paquets
- **Netmask** : masque de sous-réseau
- **Flags** : Drapeaux sur la route. U = ligne effective, G = passerelle, H = hôte
- **Interface** : interface réseau sur laquelle envoyer les paquets

Table de routage de R2 :

| Destination | Gateway | Netmask | Flags | Interface |
|---|---|---|---|---|
| 192.168.20.0 | | 255.255.255.0 | U | eth0 |
| 192.168.30.0 | | 255.255.255.0 | U | eth1 |
| default | 192.168.20.253 | 0.0.0.0 | UG | eth0 |

# Utilisation de route

- **Affichage** : `route`

# Utilisation de route

- **Affichage** : `route`
- **Ajout d'un réseau** : `add -net`

```
route add -net 10.0.2.0 netmask 255.255.255.0
```

# Utilisation de route

- **Affichage** : `route`
- **Ajout d'un réseau** : `add -net`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0
  ```
- Ajout d'une **passerelle** vers un réseau : `add -net .... gw ...`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.1.254
  ```

# Utilisation de route

- **Affichage** : `route`
- **Ajout d'un réseau** : `add -net`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0
  ```
- Ajout d'une **passerelle** vers un réseau : `add -net .... gw ...`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.1.254
  ```
- Ajout d'une **passerelle par défaut** : `add default gw`
  ```
  route add default gw 10.0.2.2
  ```

# Utilisation de route

- **Affichage** : `route`
- **Ajout d'un réseau** : `add -net`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0
  ```
- Ajout d'une **passerelle** vers un réseau : `add -net .... gw ...`
  ```
  route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.1.254
  ```
- Ajout d'une **passerelle par défaut** : `add default gw`
  ```
  route add default gw 10.0.2.2
  ```
- **Supression** d'un réseau : `del -net`
  ```
  route del -net 10.0.2.0 netmask 255.255.255.0
  ```

# Voir la route empruntée

Pour voir la **route empruntée par un paquet** : `traceroute`

```
coti@maximum:~$ traceroute magi.univ-paris13.fr
traceroute to magi.univ-paris13.fr (81.194.43.250), 30 hops max, 60
        byte packets
 1  fw.lipn.univ-paris13.fr (10.10.0.1)  2.642 ms  2.682 ms  2.676 ms
 2  gws2-r.math.univ-paris13.fr (194.254.165.254)  2.433 ms  2.495 ms
 3  magi.univ-paris13.fr (81.194.43.250)  2.286 ms  2.286 ms  2.339 ms
```

Utile notamment :

- Pour voir la route empruntée : vérifier le routage
- Pour voir où se situe un éventuel problème

# Plan du cours

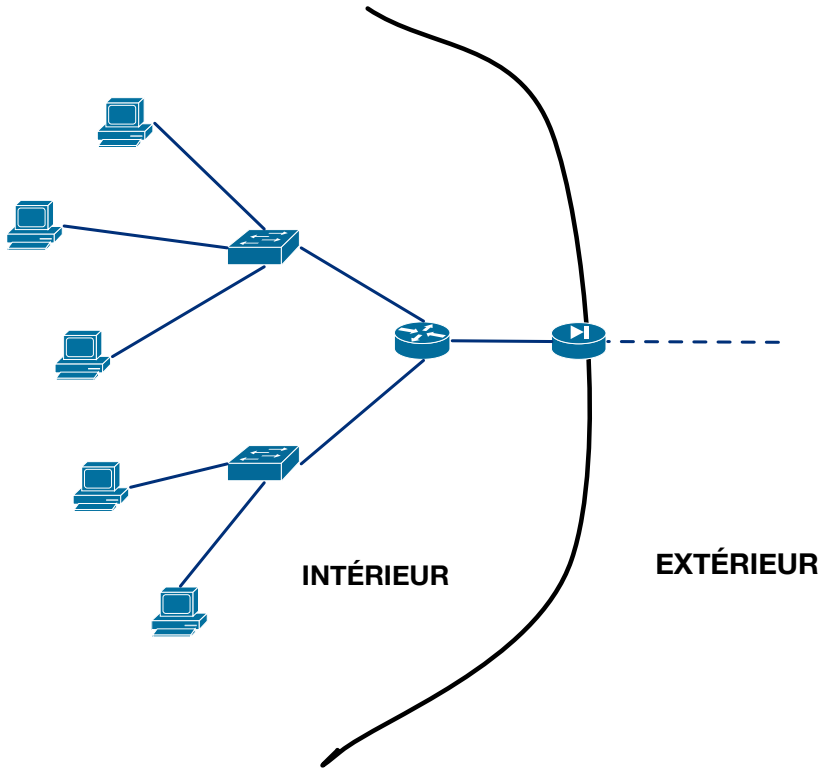
# Plan du cours

# Pare-feu : but et rôle

## Pare-feu

Élément réseau situé **entre le réseau interne et le réseau externe** qui examine les communications entrantes et sortantes, leur applique des **règles** définies par la **politique de sécurité** et effectue un **filtrage** ou des **modifications** sur les paquets réseaux le traversant.

Concrètement :



# Qu'est-ce qu'un pare-feu

Le pare-feu dispose d'au moins deux interfaces réseaux :

- Une interface vers l'*extérieur* du réseau
- Une interface vers l'*intérieur* du réseau

Les paquets traversent le pare-feu

→ Le pare-feu décide de ce qu'il en fait (modificaiton, suppression...)

Notion de **zone de confiance**

- Le pare-feu contrôle le trafic entre différentes zones de confiance

Filtrage selon différents critères :

- Type de protocoles, ports utilisés, contenu, utilisateur...

Application de règles :

- Accepter : **accept**
- Bloquer : **deny**
- Rejeter (sans message d'erreur) : **drop**

# Zones de confiance ?

Quelles sont ces **zones de confiance** ?

- Le **monde extérieur** (Internet) : confiance *nulle*
- Le **réseau interne** : confiance plus élevée
- Des serveurs, devant être **accessibles de l'extérieur**

But du firewall : **contrôler la connectivité entre zones de niveau de confiance différents**

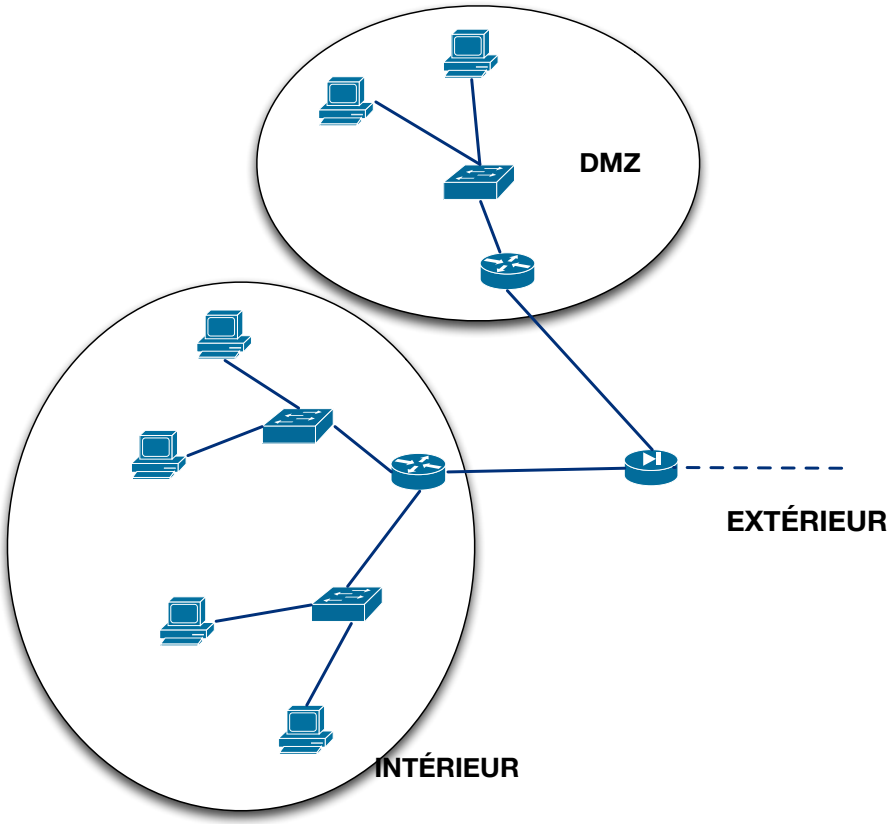
La politique à définir pour les serveurs n'est pas la même que celle pour l'intérieur du réseau

→ Protection différente pour cette zone particulière

# Zone démilitarisée

On met les serveurs devant être accessibles dans une **zone démilitarisée (DMZ)**

- Politique de protection différente du reste de l'intérieur du réseau
- Doivent être accessibles depuis l'extérieur
  - Contrairement à l'intérieur du réseau lui-même
- Attention : ne signifie pas aucune protection !
- Étanchéité avec le reste de l'intérieur du réseau



# Politique définie

- **Authorisations explicites uniquement** : tout ce qui n'est pas autorisé est interdit
  - On définit a prori une liste de ce qui est autorisé
  - Contraignant, mais peu de mauvaises surprises
- **Interdictions explicites uniquement** : tout ce qui n'est pas interdit est autorisé
  - On définit une liste exhaustive de ce qui est interdit
  - Besoin de dénombrer ce qui est interdit !
  - Moins contraignant mais peu sûr

Choix à faire en fonction de beaucoup de paramètres...

- Contraintes (d'utilisateurs, d'applications...)
- Risques identifiés
- ...

# Plan du cours

# Types de pare-feux

Suivant le niveau où le pare-feu agit, plusieurs types :

- **Stateless Packet Inspection** (ou Stateless Packet Filtering)
- **Stateful Packet Inspection** (ou Stateful Packet Filtering)
- **Deep Packet Inspection**

Différences :

- Efficacité, protection apportée
- Demande en ressources

Compromis à faire : pas de solution universelle.

# Stateless Packet Inspection

Filtrage simple, **niveau 3 et 4** du modèle OSI

- Analyse des en-têtes des paquets entrants
  - Possibilité de filtrer par IP source, IP destination, protocole (TCP, UDP, ICMP...), port utilisé...

→ Rapide !

# Stateless Packet Inspection

Filtrage simple, **niveau 3 et 4** du modèle OSI

- Analyse des en-têtes des paquets entrants
  - Possibilité de filtrer par IP source, IP destination, protocole (TCP, UDP, ICMP...), port utilisé...

→ Rapide !

Typiquement :

- Blocage de certaines adresses IP entrantes, repérées comme dangereuses
- Blocage des ports non-indispensables (23 = telnet, non sécurisé)
- Blocage en sortie de certaines adresses IP (sites à interdire aux employés)
- Autorisation uniquement des services indispensables (port 80 = HTTP pour un serveur HTTP en DMZ...)

# Stateless Packet Inspection

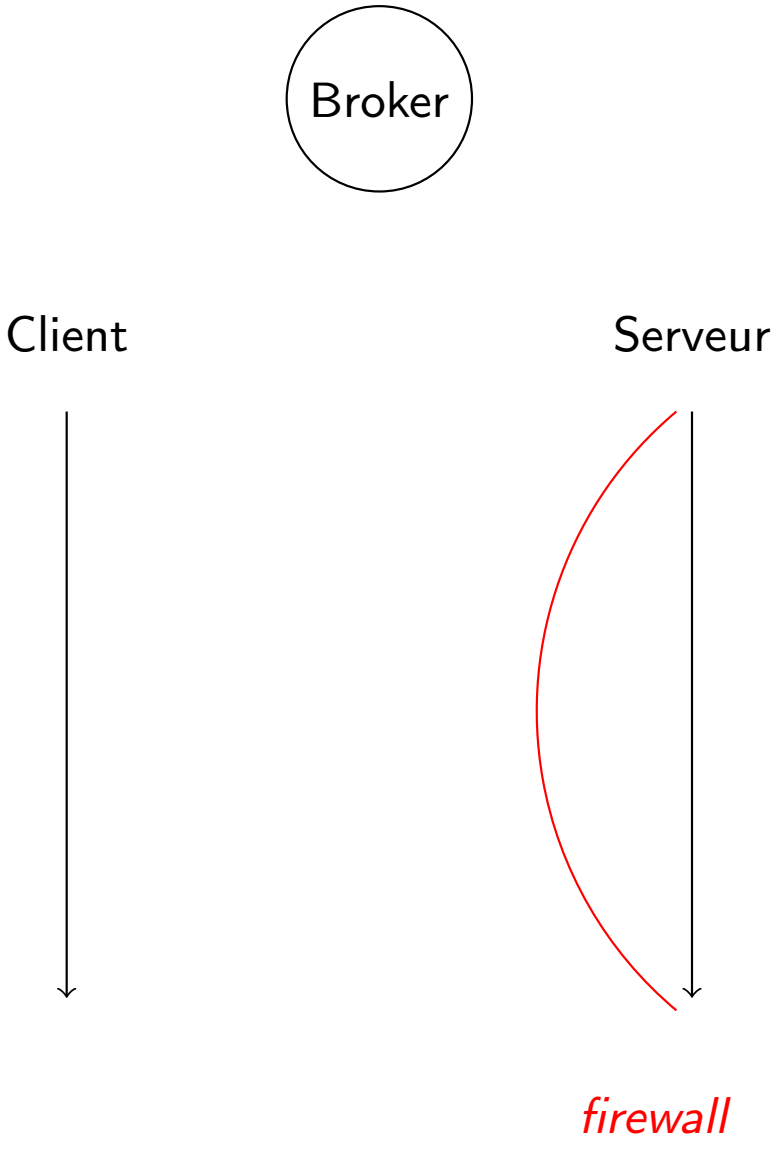
Problème de la Stateless Packet Inspection : **rudimentaire**

- Par exemple, les connexions TCP sont interdites en bloquant simplement les paquets SYN entrants
- Vulnérable à beaucoup d'attaques : spoofing, TTCP...

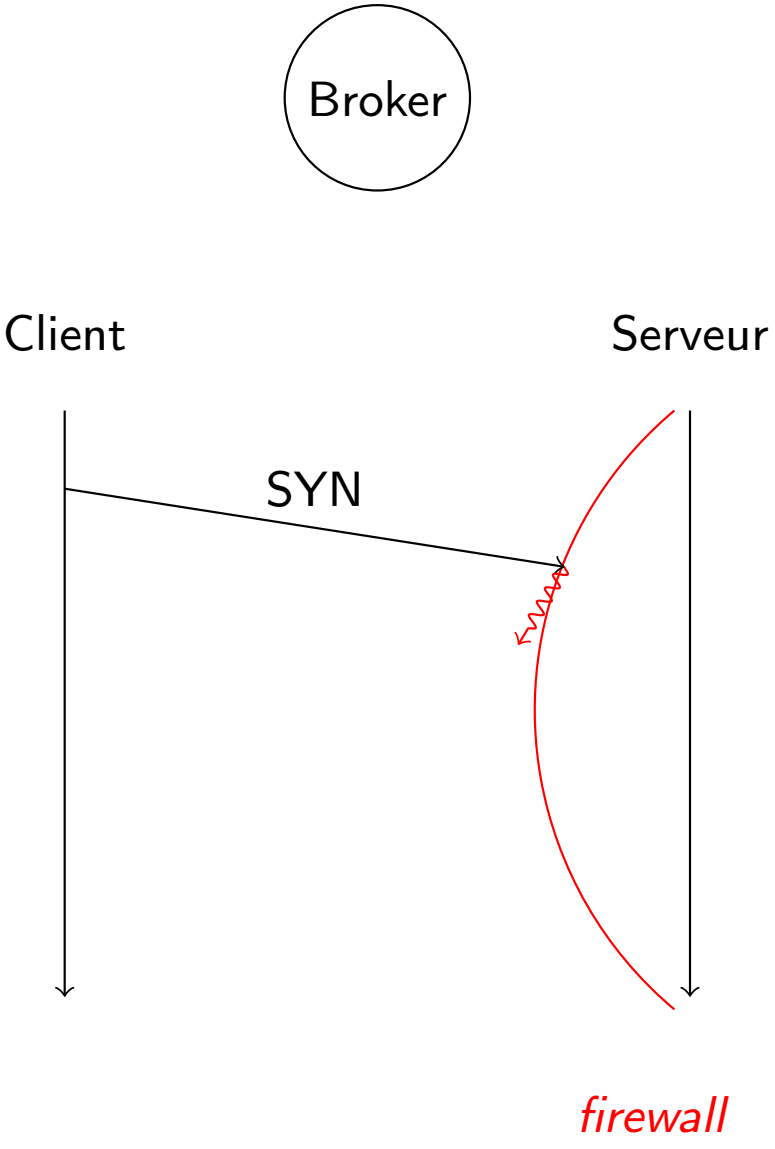
Mais : **rapide**

- Ne regarde que l'en-tête du paquet
- Pas besoin de conserver des données en mémoire

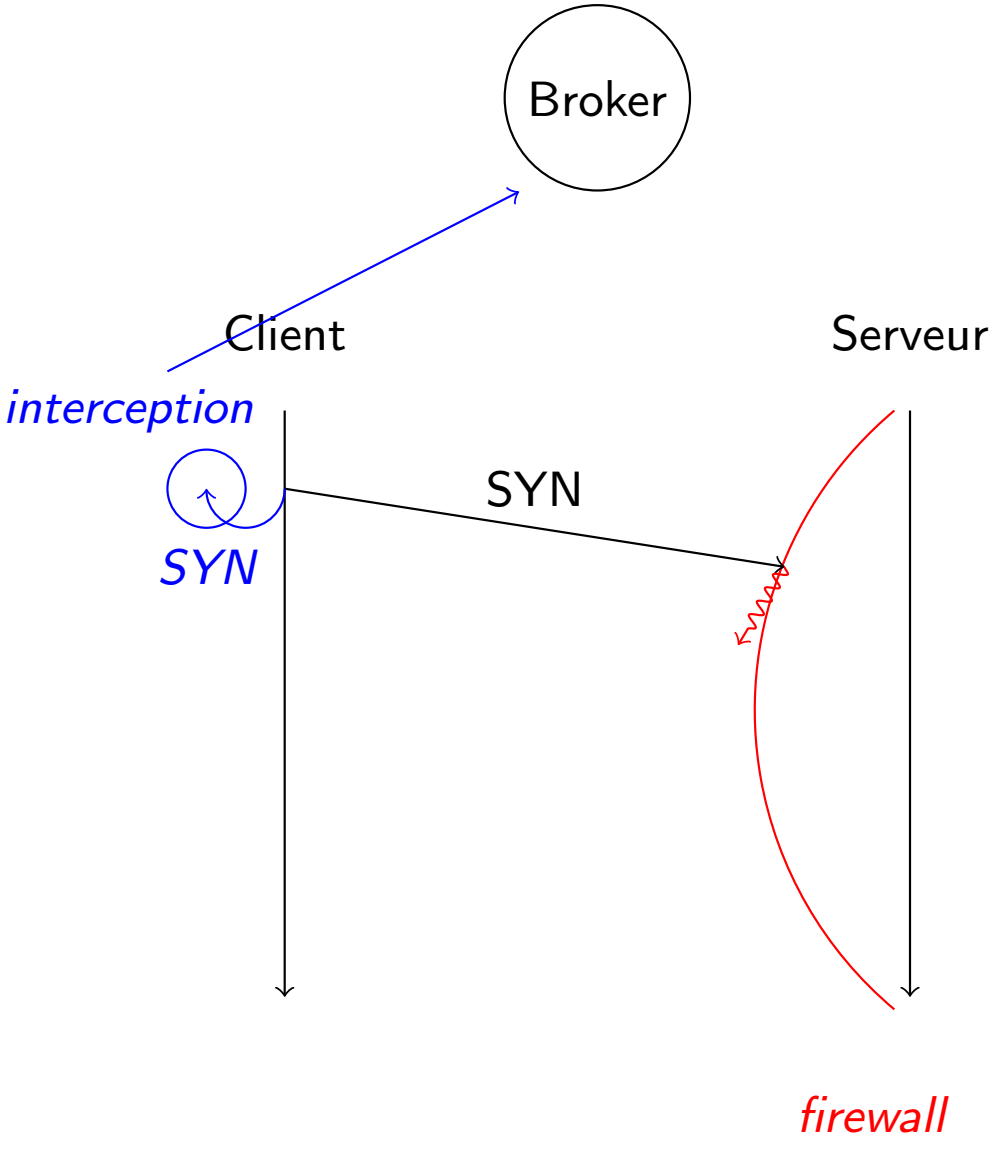
# Traversée de firewall *stateless* : protocole TCP traversant



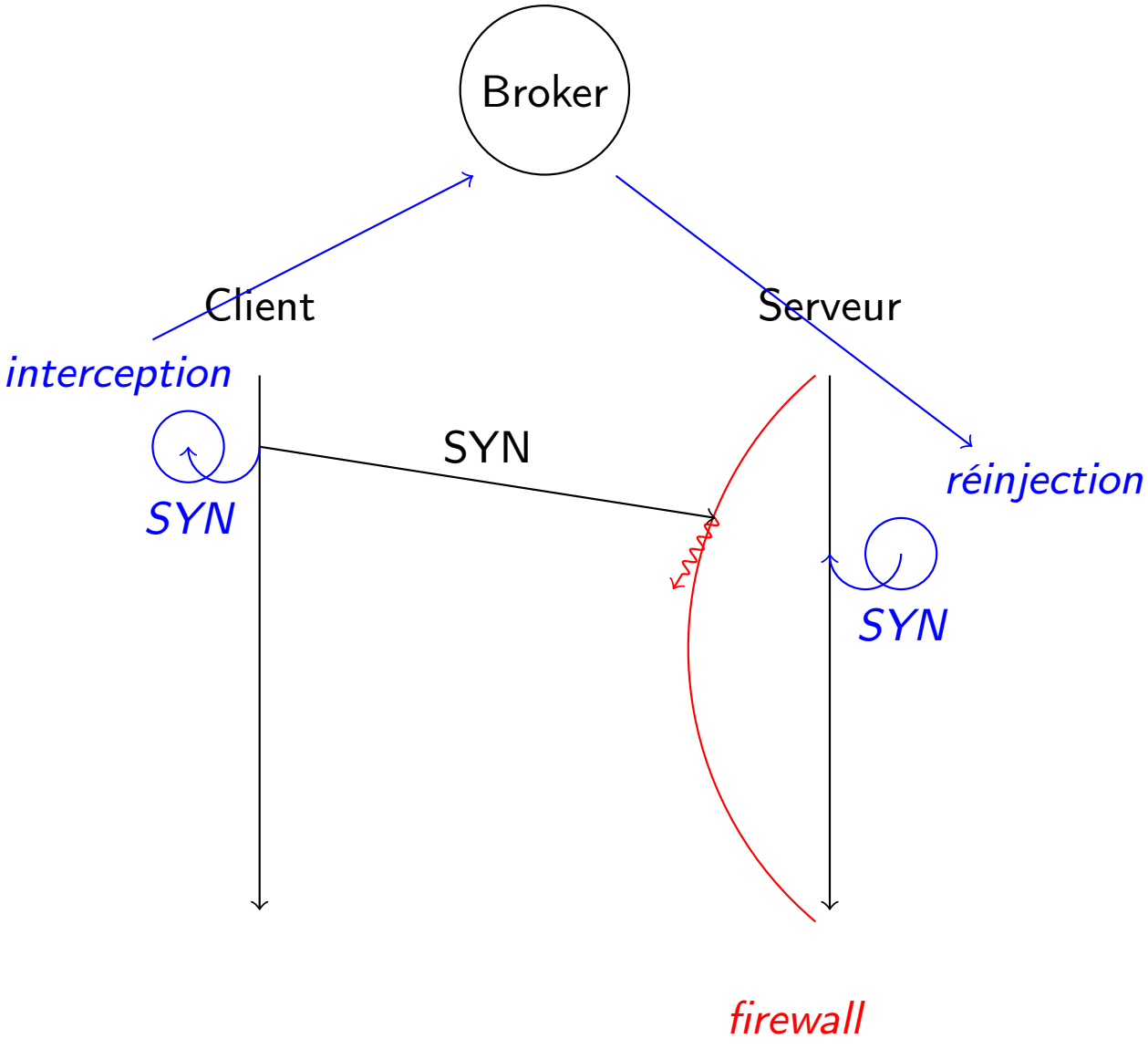
# Traversée de firewall *stateless* : protocole TCP traversant



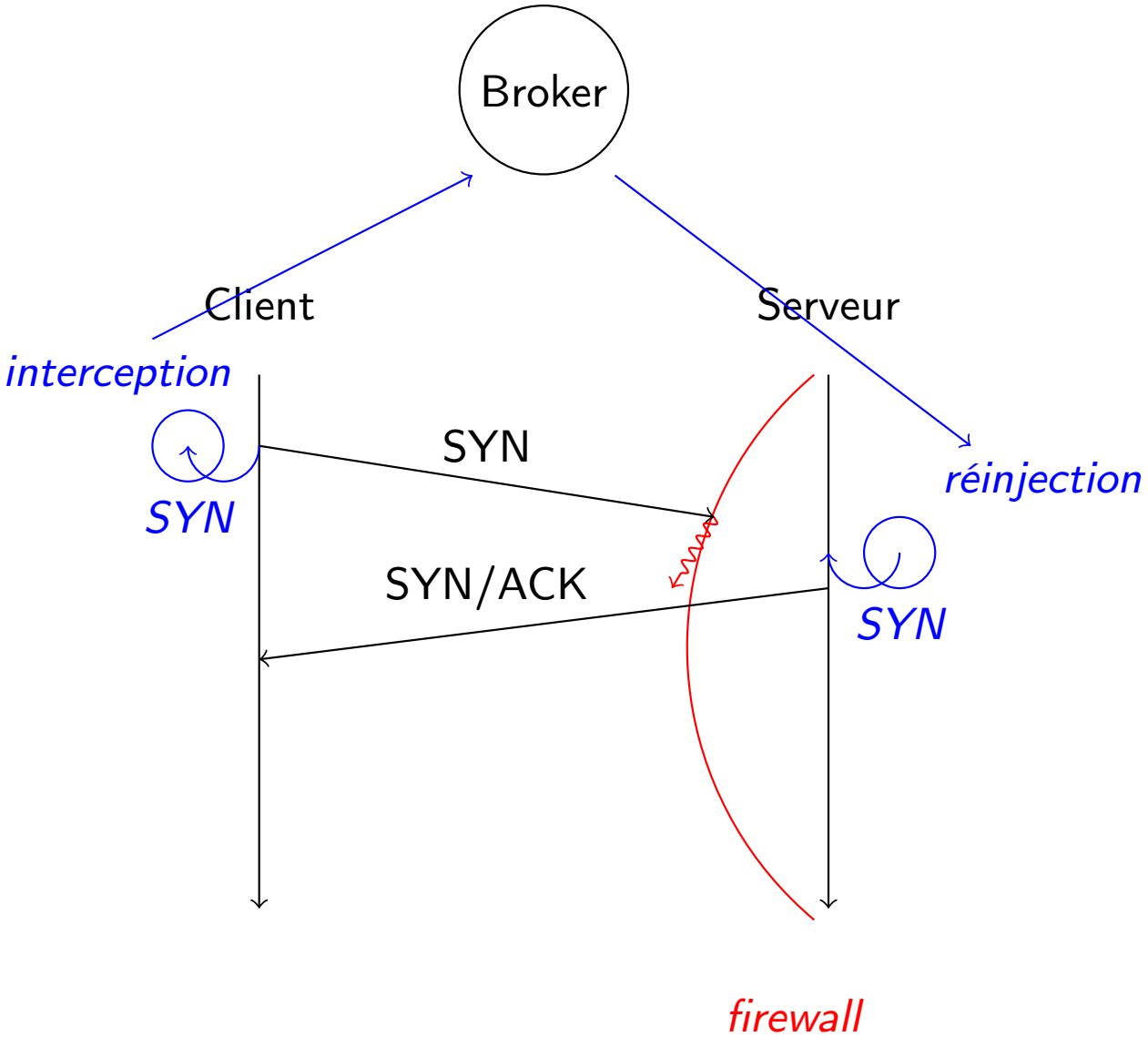
# Traversée de firewall *stateless* : protocole TCP traversant



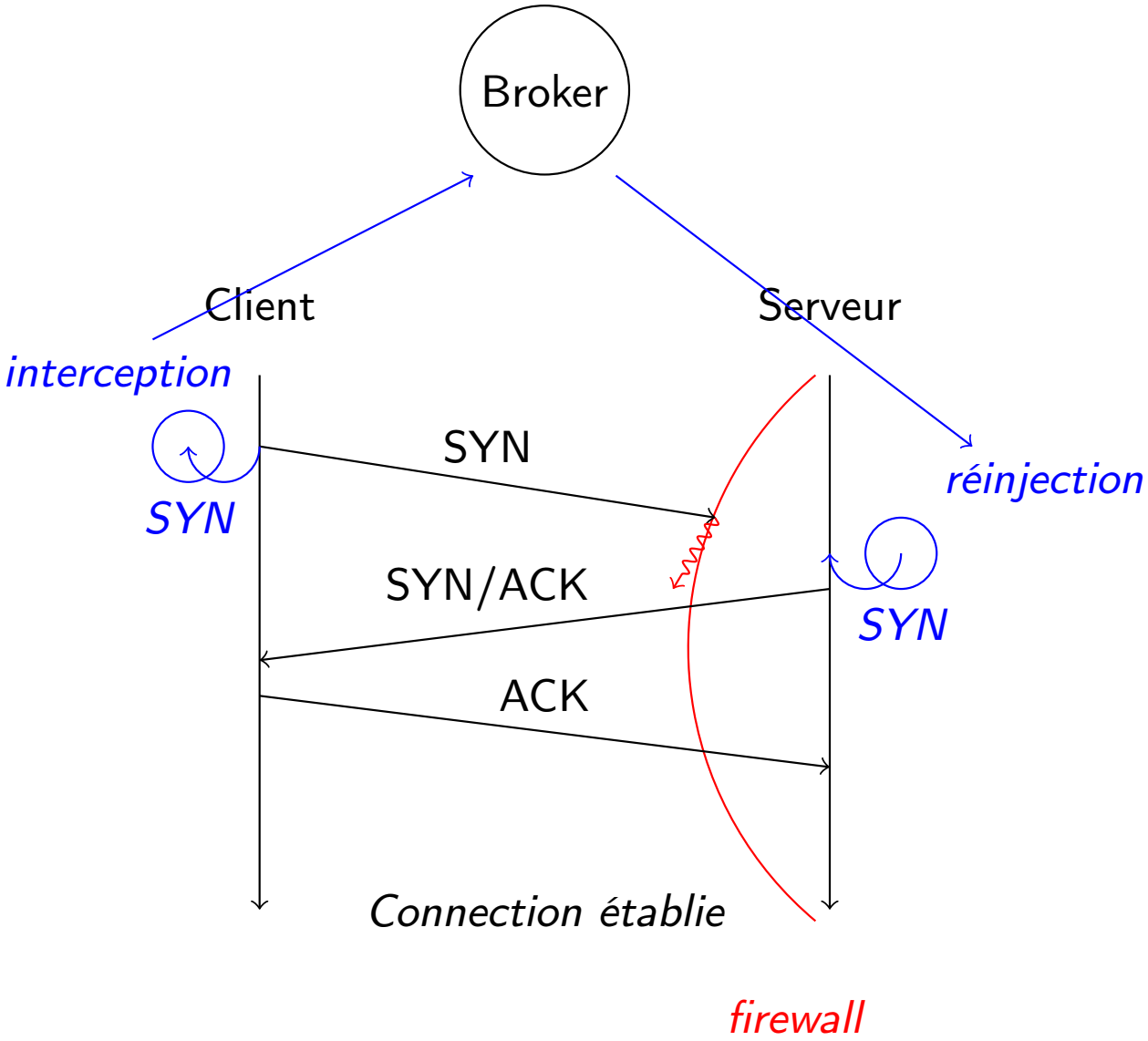
# Traversée de firewall *stateless* : protocole TCP traversant



# Traversée de firewall *stateless* : protocole TCP traversant



# Traversée de firewall *stateless* : protocole TCP traversant



# Stateful Packet Inspection

Filtrage **niveau 3 et 4** du modèle OSI

- Regarde l'en-tête des paquets
- **Conserve en mémoire l'état des connexions**
- Attention : pas parfait ! Possibilité de passer outre (connexions externes, Javascript).

Par exemple : suivi de l'état d'une connexion TCP

- Dans l'exemple précédent, le ACK n'a rien à faire ici puisqu'aucune connexion n'est en cours
- Donc il est éliminé et la connexion n'est pas établie

Pour les protocoles sans connexion :

- Le premier paquet ouvre un passage (un trou) dans le firewall
- Les paquets suivants soivent emprunter ce passage
- Expiration du passage par timeout
- Maintenance du passage par *UDP hole punching* : petit paquet envoyé de temps en temps pour ne pas laisser expiré le timeout

# Deep Packet Inspection

Deep Packet Inspection = firewall applicatif

- Niveau 7 du modèle OSI

On va regarder **le contenu des paquets**

- Les protocoles sont-ils respectés ?
- Les paquets contiennent-ils des éléments interdits ? (mot-clés, etc)
- Permet d'éviter l'utilisation de failles dans les protocoles

Notamment : ne permet pas l'utilisation de proxy pour tromber de pare-feu sur l'adresse distante

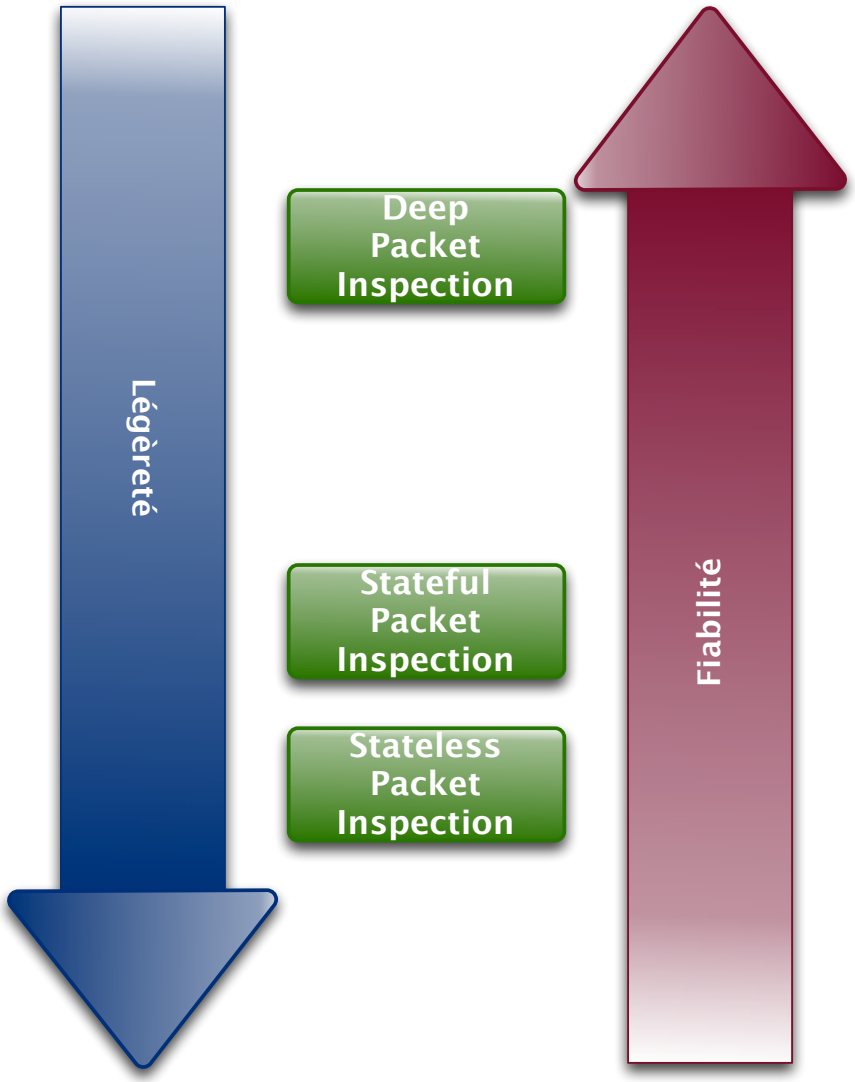
Exemple : Eagle de Amesys

- Surveille environ 300 protocoles : HTTP, mail (IMAP, POP3, SMTP), VoIP, recherches dans des moteurs, analyse sémantique du contenu...

Très très gourmand en ressources !!

- Chaque paquet est analysé selon un grand nombre de critères

# Firewalls : comparaison



# Plan du cours

# Plan du cours

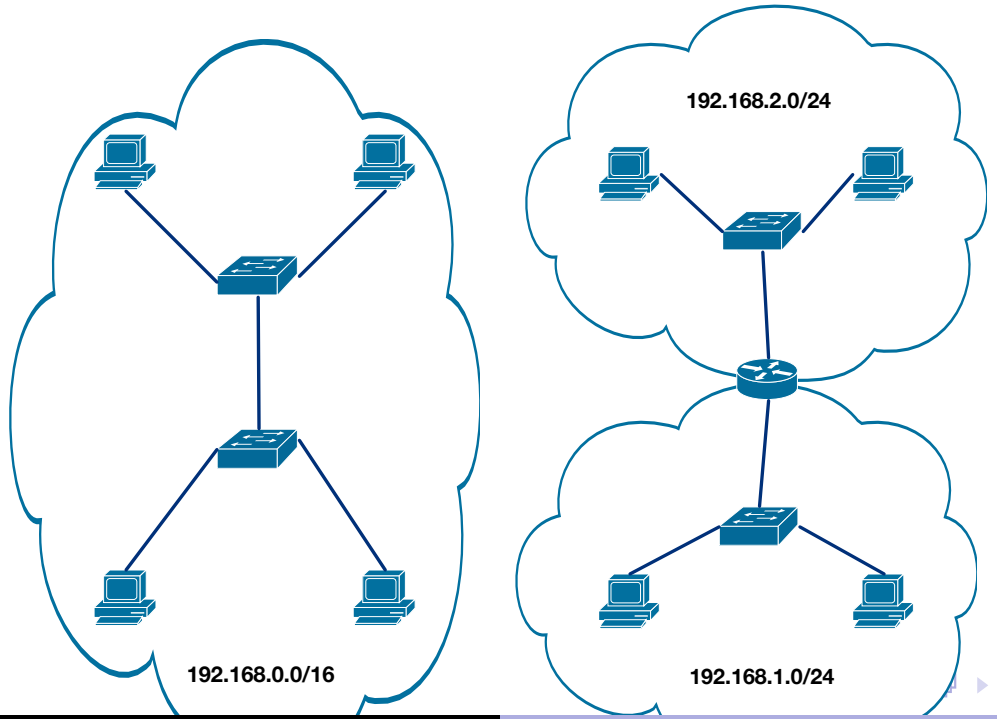
# Plan du cours

# Segmentation des réseaux locaux

## Segmentation de réseaux

Deux règles de base :

- Une interface de **switch** délimite un domaine de collision
  - → Le **switch** décide d'envoyer un message sur l'un ou l'autre de ses ports **à partir de l'adresse MAC** de destination.
- Une interface de **routeur** délimite un domaine de diffusion
  - → Le routeur décide d'envoyer un message sur l'un ou l'autre de ses ports **à partir de l'adresse IP** de destination.



# Rappels sur les VLAN

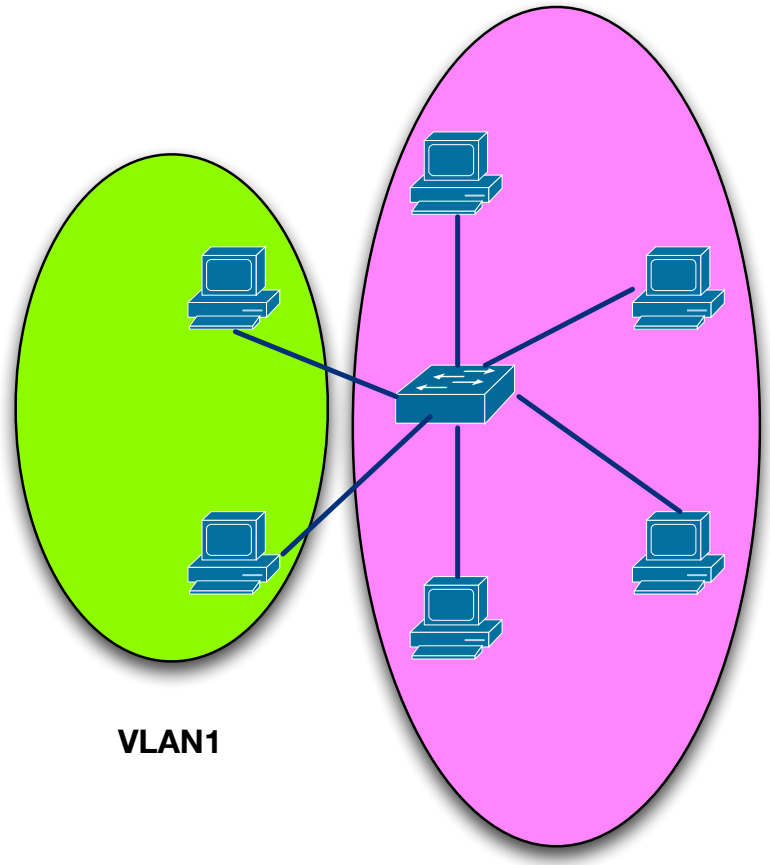
Principe : découper le réseau en sous-réseaux virtuels

- Les machines de deux VLANs différents ne peuvent pas s'échanger de trames *directement*
- Réseaux logiques séparés
- Réseaux physiques quelconques (séparés ou non)

# Rappels sur les VLAN

Principe : découper le réseau en sous-réseaux virtuels

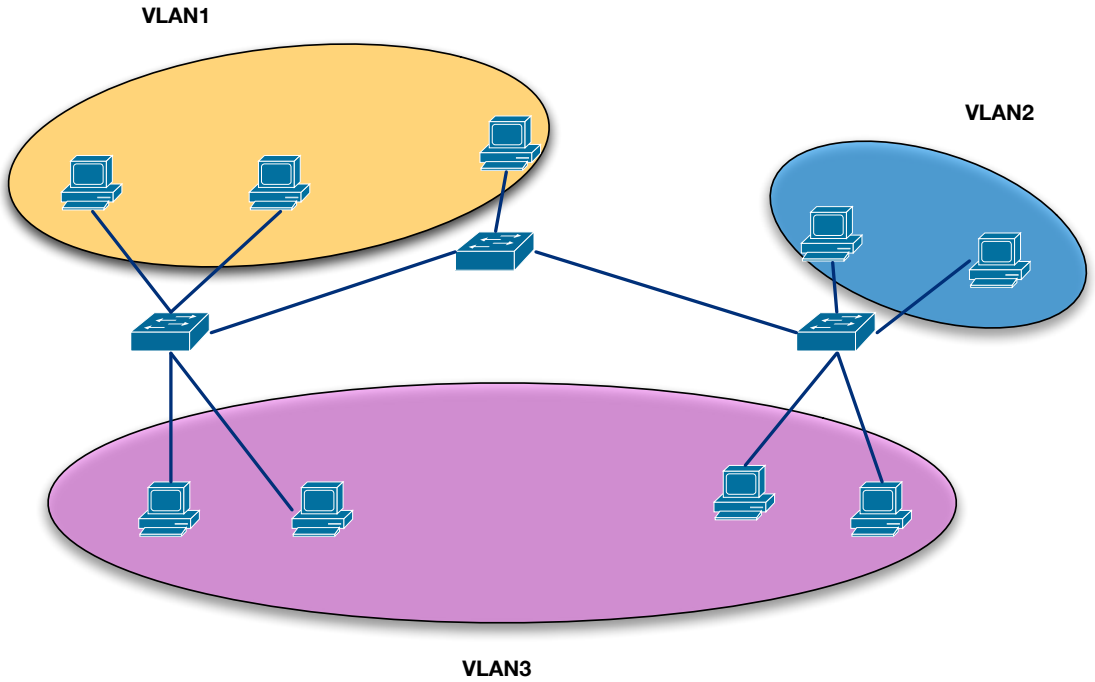
- Les machines de deux VLANs différents ne peuvent pas s'échanger de trames *directement*
- Réseaux logiques séparés
- Réseaux physiques quelconques (séparés ou non)



# Rappels sur les VLAN

Principe : découper le réseau en sous-réseaux virtuels

- Les machines de deux VLANs différents ne peuvent pas s'échanger de trames *directement*
- Réseaux logiques séparés
- Réseaux physiques quelconques (séparés ou non)

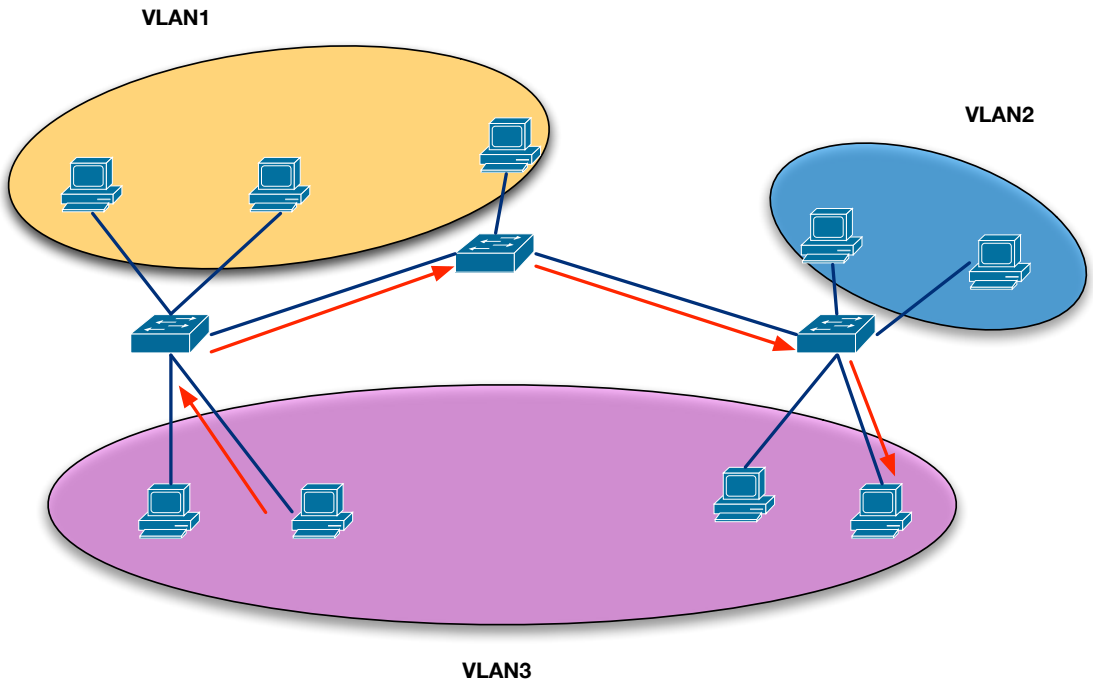


# Rappels sur les VLAN

Un seul VLAN peut s'étendre sur **plusieurs switches**

→ Notion de **trunk**

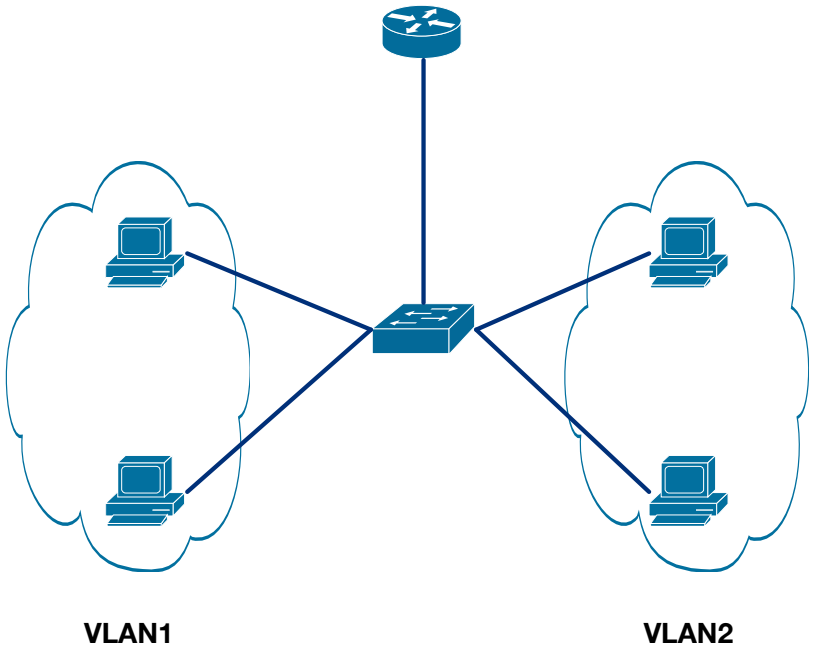
→ Les trames circulent entre les switches



# Problématique de routage

Dans cette situation :

- VLAN couvrant plusieurs interfaces du switch
- Routage entre VLANs

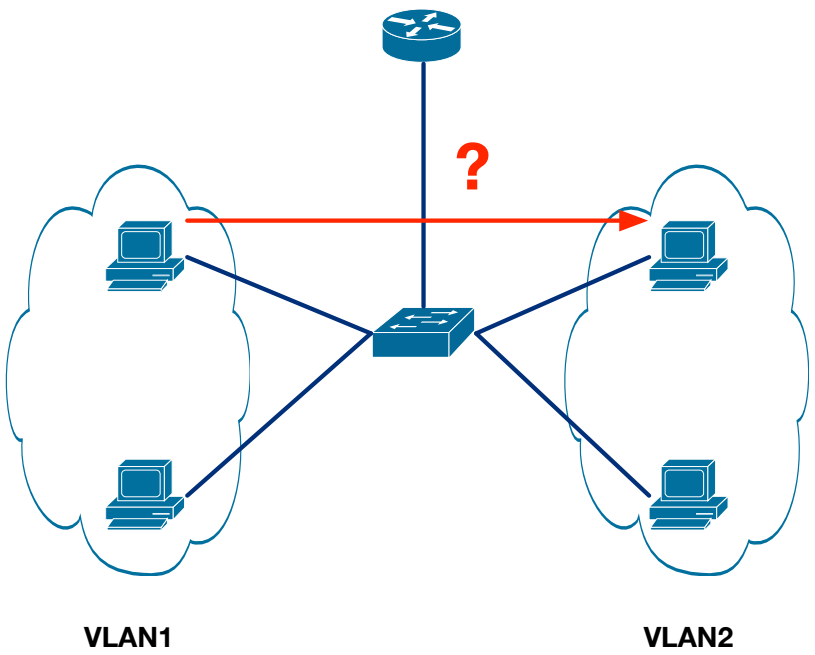


# Problématique de routage

Dans cette situation :

- VLAN couvrant plusieurs interfaces du switch
- Routage entre VLANs

**Question** : comment communiquer **entre deux VLANs** entre deux interfaces du routeur ?



# Routage inter-VLANs 1/2

Rappel de la définition d'un routeur : *interconnecter plusieurs réseaux*. Ici : **interconnecter plusieurs réseaux virtuels** (VLANS).

→ Chaque VLAN = une interface **virtuelle** au niveau du routeur

**Routage inter-VLANs** : faire passer une trame d'un VLAN à un autre

- Interconnexion de deux réseaux virtuels

# Routage inter-VLANs 1/2

Rappel de la définition d'un routeur : *interconnecter plusieurs réseaux*. Ici : **interconnecter plusieurs réseaux virtuels** (VLANS).

→ Chaque VLAN = une interface **virtuelle** au niveau du routeur

**Routage inter-VLANs** : faire passer une trame d'un VLAN à un autre

- Interconnexion de deux réseaux virtuels

Mise en œuvre niveau 3 :

- Le routeur détag les trames du VLAN source
- puis il ajoute le tag des trames du VLAN de destination

Exemple :

- VLAN1 → routeur → VLAN2
- Les trames arrivent sur le routeur avec le tag de VLAN1
- Le routeur les modifie pour remplacer le tag VLAN1 par le tag VLAN2
- Les trames repartent avec le tag du VLAN2 → elles passent dans le VLAN2

# Routage inter-VLANs 2/2

Mise en œuvre en utilisant un routeur :

- On relie les VLANs au routeur

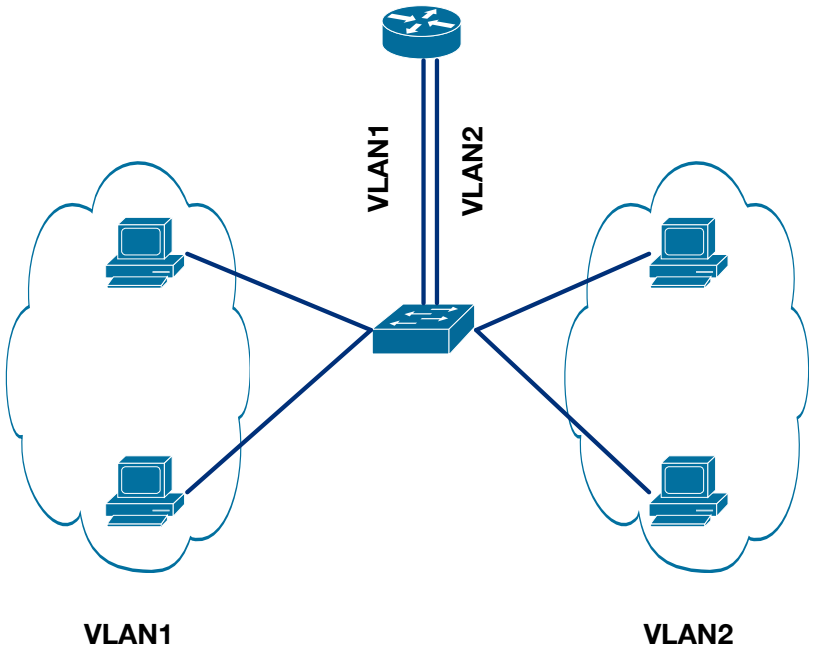
# Routage inter-VLANs 2/2

Mise en œuvre en utilisant un routeur :

- On relie les VLANs au routeur

Deux possibilités :

- Une interface du routeur **par VLAN**
  - → Physiquement, on branche **plusieurs câble** entre le switch et le routeur



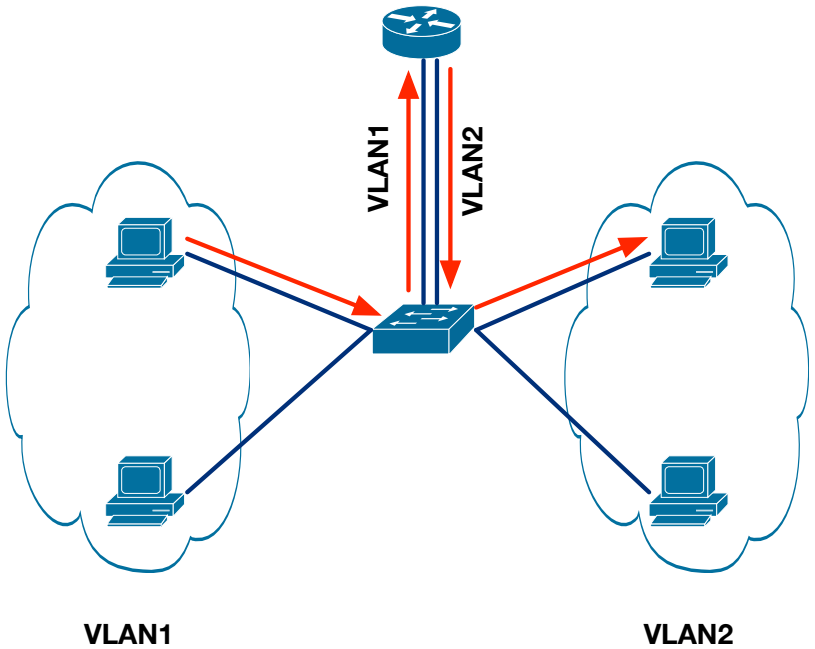
# Routage inter-VLANs 2/2

Mise en œuvre en utilisant un routeur :

- On relie les VLANs au routeur

Deux possibilités :

- Une interface du routeur **par VLAN**
  - → Physiquement, on branche **plusieurs câble** entre le switch et le routeur



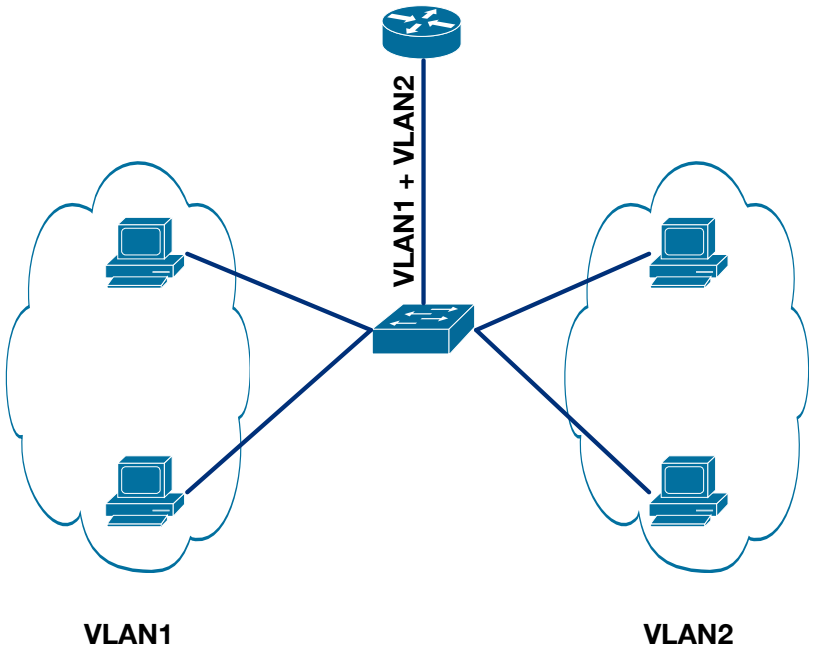
# Routage inter-VLANs 2/2

Mise en œuvre en utilisant un routeur :

- On relie les VLANs au routeur

Deux possibilités :

- Une seule interface physique pour **plusieurs VLANs**
  → Utilisation de **plusieurs interfaces virtuelles** sur cette interface du routeur



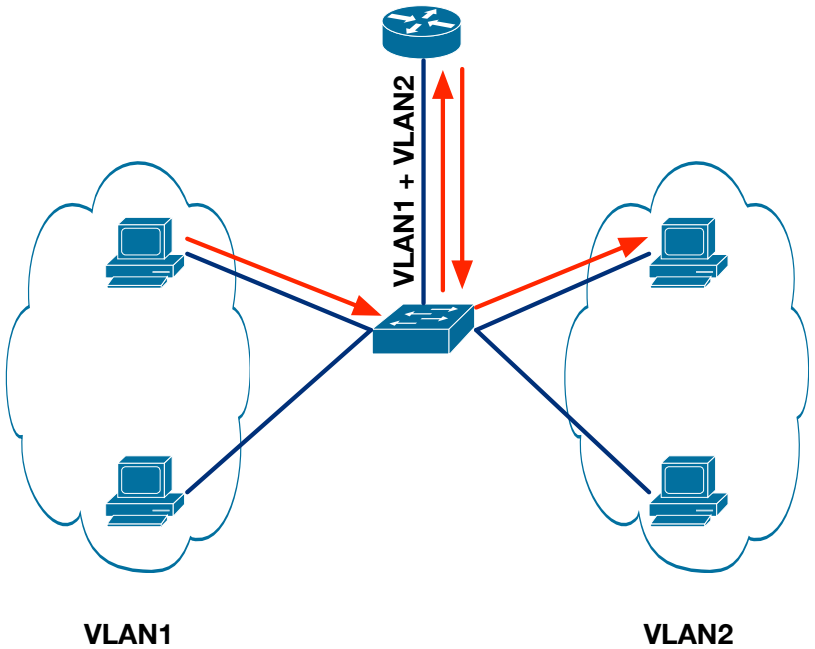
# Routage inter-VLANs 2/2

Mise en œuvre en utilisant un routeur :

- On relie les VLANs au routeur

Deux possibilités :

- Une seule interface physique pour **plusieurs VLANs**
  → Utilisation de **plusieurs interfaces virtuelles** sur cette interface du routeur



# Utilisation d'une interface virtuelle

Sur la plupart des routeurs on peut définir **plusieurs interfaces virtuelles** sur une interface physique

- Commande : `ifconfig ethX:Y ... netmask ....`

Exemple :

- `ifconfig eth0:0 192.168.10.254 netmask 255.255.255.0`
- `ifconfig eth0:1 192.168.20.254 netmask 255.255.255.0`

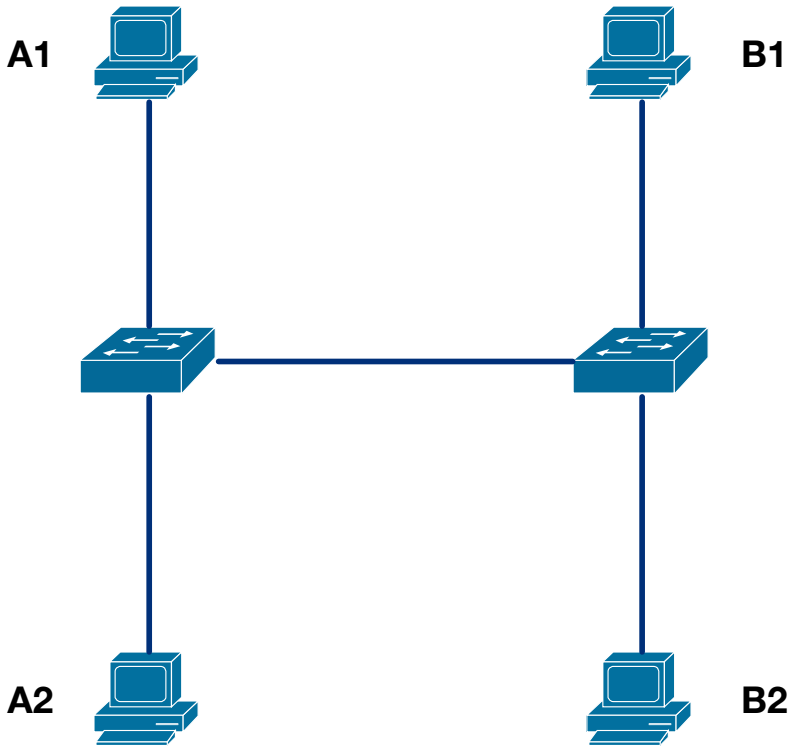
Définition de la table de routage associée :

| Destination | Gateway | Netmask | Flags | Interface |
|---|---|---|---|---|
| 192.168.10.0 | | 255.255.255.0 | U | eth0 |
| 192.168.20.0 | | 255.255.255.0 | U | eth0 |

# Utilisation de VLAN

Intérêt des VLANs :

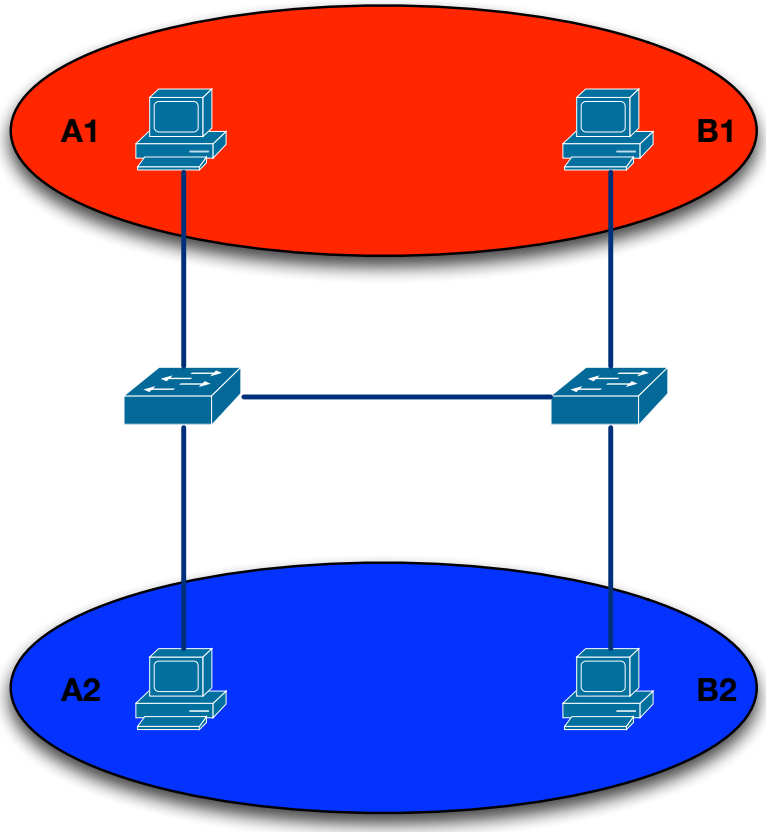
- Confinement, séparation de réseaux reliés physiquement les uns aux autres, sécurité élémentaire... (cf M2101)
- Regrouper dans **le même VLAN** des éléments **géographiquement distants**



# Utilisation de VLAN

Intérêt des VLANs :

- Confinement, séparation de réseaux reliés physiquement les uns aux autres, sécurité élémentaire... (cf M2101)
- Regrouper dans **le même VLAN** des éléments **géographiquement distants**



# Utilisation de VLAN : exemples

Exemple 1 :

- Une entreprise sur plusieurs bâtiments
- Une fibre optique reliant les bâtiments, via un switch dédié
- Plusieurs switches dans chaque bâtiment : switch d'étage, etc.
- Nécessité d'avoir plusieurs réseaux : réseau d'administration système/réseau, réseau compta, réseau de la direction, réseau des machines-outil...
- Mais on retrouve des postes dnas chaque catégorie dans *tous les bâtiments*, tous les étages...

→ Utilisation d'un VLAN par réseau !

Exemple 2 :

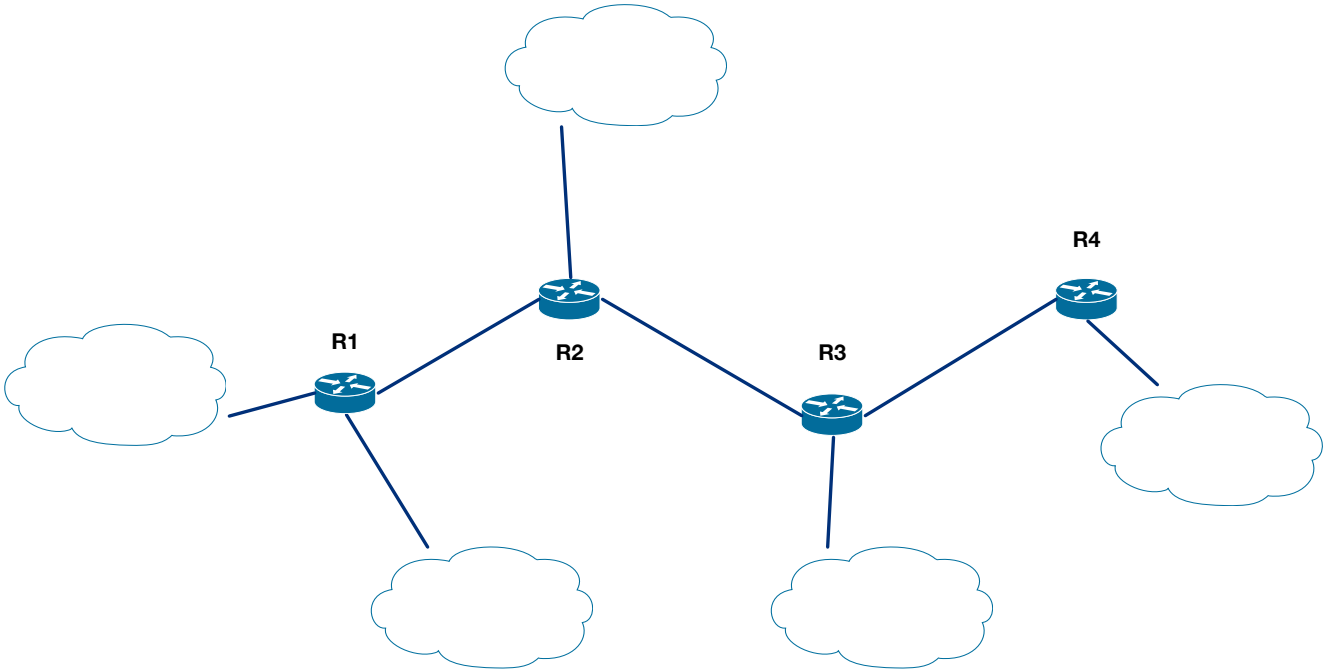
- Une entreprise disposant de données critiques, à ne surtout pas perdre
- Que faire si tout le bâtiment est détruit ?
- Sauvegarde physiquement à l'extérieur du bâtiment principal
- Un switch reliant les deux bâtiments

→ VLAN reliant les serveurs principaux et les sauvegardes

# Plan du cours

# VLAN d'interconnexion

Rappel d'infrastructure réseau : liaison directe entre les routeurs

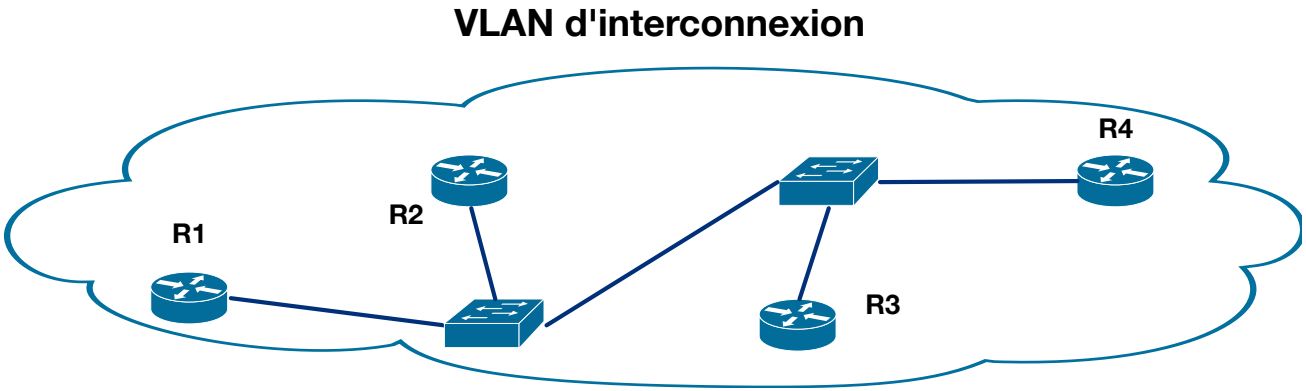



→ Création d'un réseau R1-R2, R2-R3, R3-R4

→ **Un sous-réseau par paire** de routeurs !

# VLAN d'interconnexion

Rappel d'infrastructure réseau : liaison directe entre les routeurs

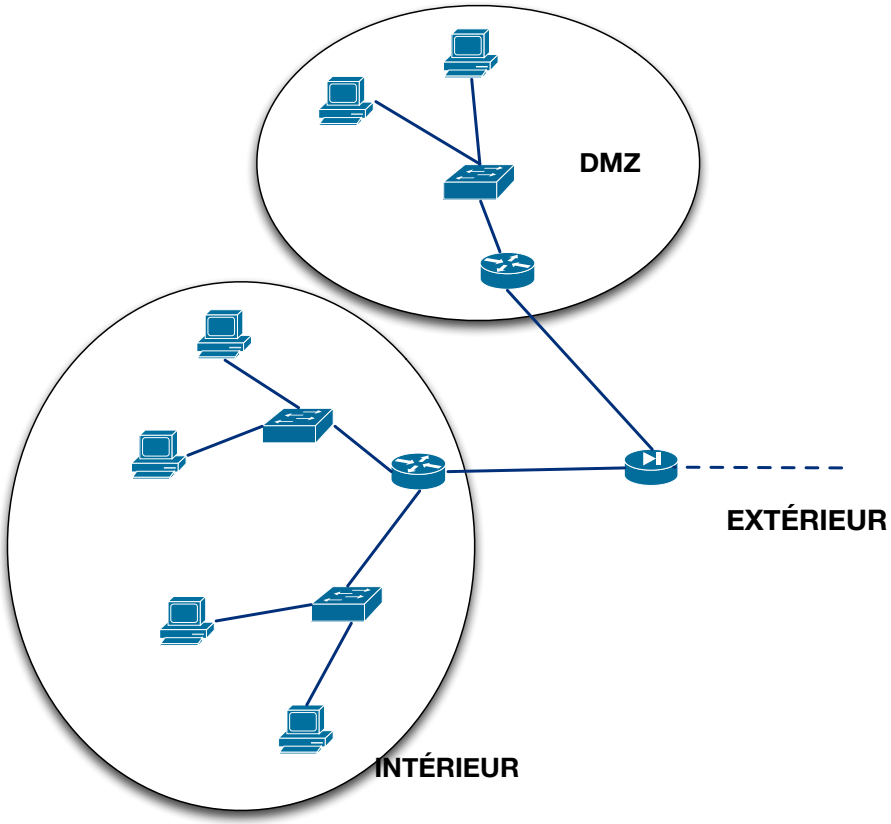



→ **VLAN d'interconnexion** comprenant R1, R2, R3 et R4

→ Un seul sous-réseau pour tous les routeurs

# Plan du cours

# Zone démilitarisée




Contraintes :

- Au moins un port du firewall est **dédié à la DMZ**
- Nécessité de **regrouper géographiquement** les serveurs dans la DMZ

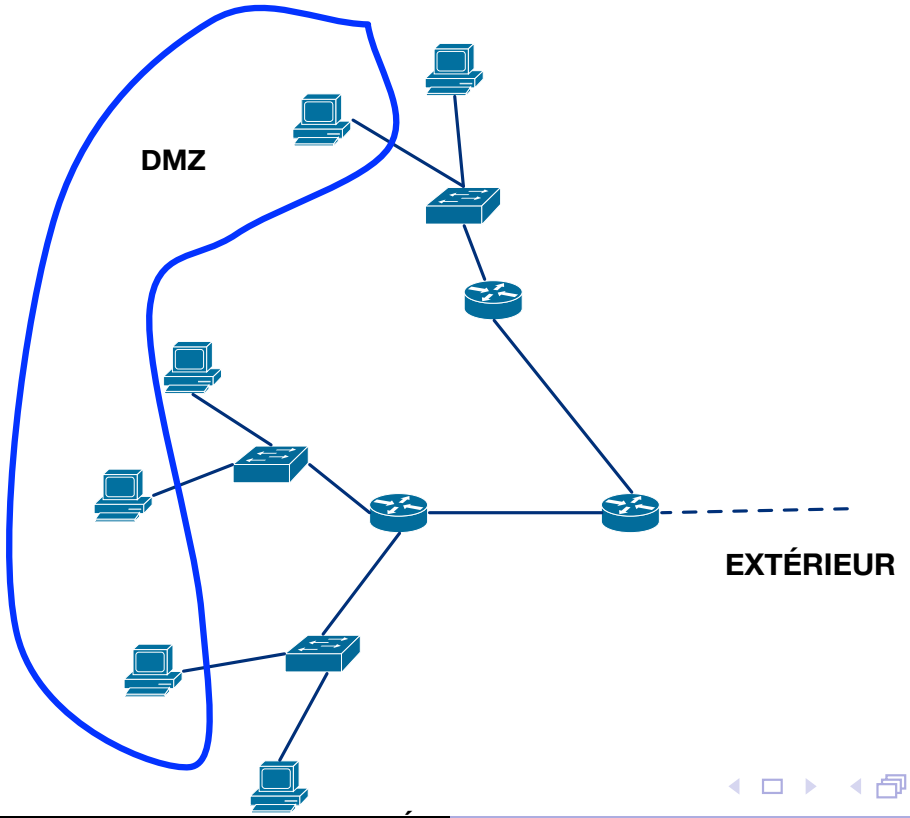
# DMZ en VLAN

Utilisation d'un **VLAN pour la DMZ**

- La DMZ est **séparée** du reste du réseau grâce au confinement en VLAN
- Segmentation de niveau 2
- On s'affranchit de la contrainte géographique

Définition de **règles de filtrage** entre les VLANs

- Via un **routeur filtrant** (équipement spécifique)
- Via un firewall au niveau du routeur inter-VLANs



# Plan du cours

# Plan du cours

# Routage statique vs dynamique

## Routage statique

Tables de routage entrées "manuellement" dans les machines

- Configuration simple, rapide, une fois pour toutes
- Routage simple à calculer
- Tout est local : pas besoin de connaître le reste de l'état du réseau

Mais toute modification du réseau nécessite une mise à jour de toutes les tables de routage.

## Routage dynamique ou adaptatif

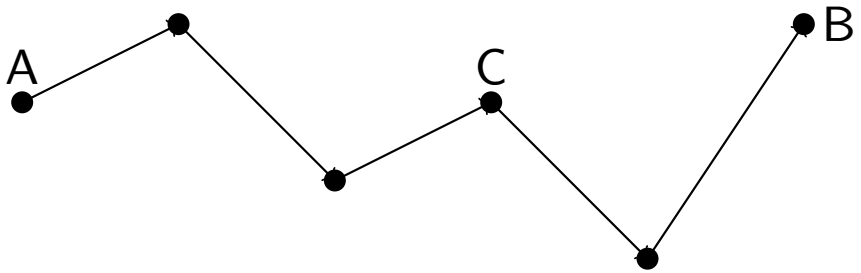
Pas de connaissance a priori du réseau

- Décisions de routage fondées sur la topologie, le trafic...
- Possibilité de modifier les décisions de routage en cas de modifications de la topologie (pannes, modification du réseau)

# Principes d'optimalité

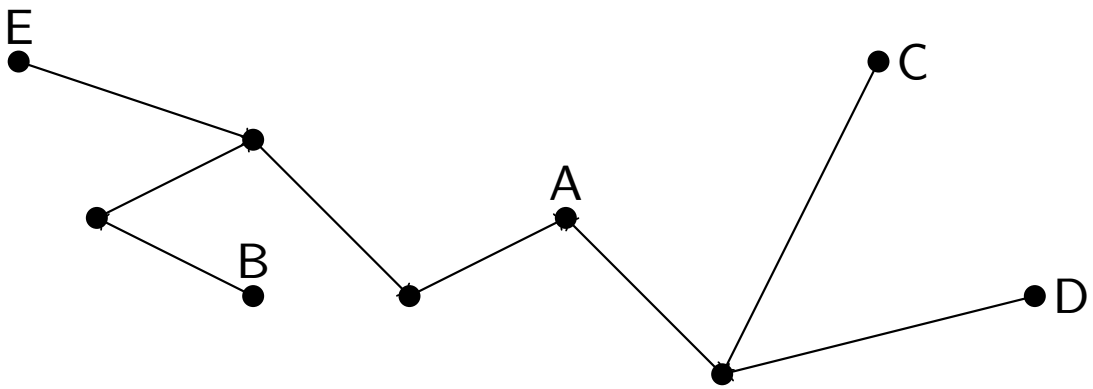
## Principe d'optimalité

Si un chemin de A vers B est optimal, alors si C est sur le chemin entre A et B la portion $[AC]$ et la potion $[CB]$ de $[AB]$ sont optimales.




## Arbre collecteur

Un arbre collecteur rassemble tous les chemins optimaux de toutes les sources vers une destination donnée.



# Métriques pour mesurer la performance d'un algorithme de routage

**Objectif**

Router les messages d'une source vers une destination à moindre coût.

Comment évaluer la performance d'un algorithme de routage ? Détermination d'une métrique (ou notion de coût)

- Nombre de sauts : par combien d'intermédiaires le message doit transiter
- Temps : somme des délais de transmission
- Robustesse, résilience : le message peut-il arriver en cas de pannes
- Nombre de messages générés, congestion provoquée par la propagation du message
- ...

Un algorithme ne peut pas être optimal partout !

- Suivant le contexte, on choisira un algorithme plutôt qu'un autre.

# Plan du cours

# Algorithme glouton : par inondation

Principe : on envoie le paquet à tous ses voisins, sauf le lien d'origine

- De proche en proche, le paquet finira bien par arriver à son destinataire...

## Avantages

- Très *robuste* : si il y a des pannes sur le réseau, le paquet passe quand même tant que le réseau reste connexe
- Le plus court chemin est toujours trouvé

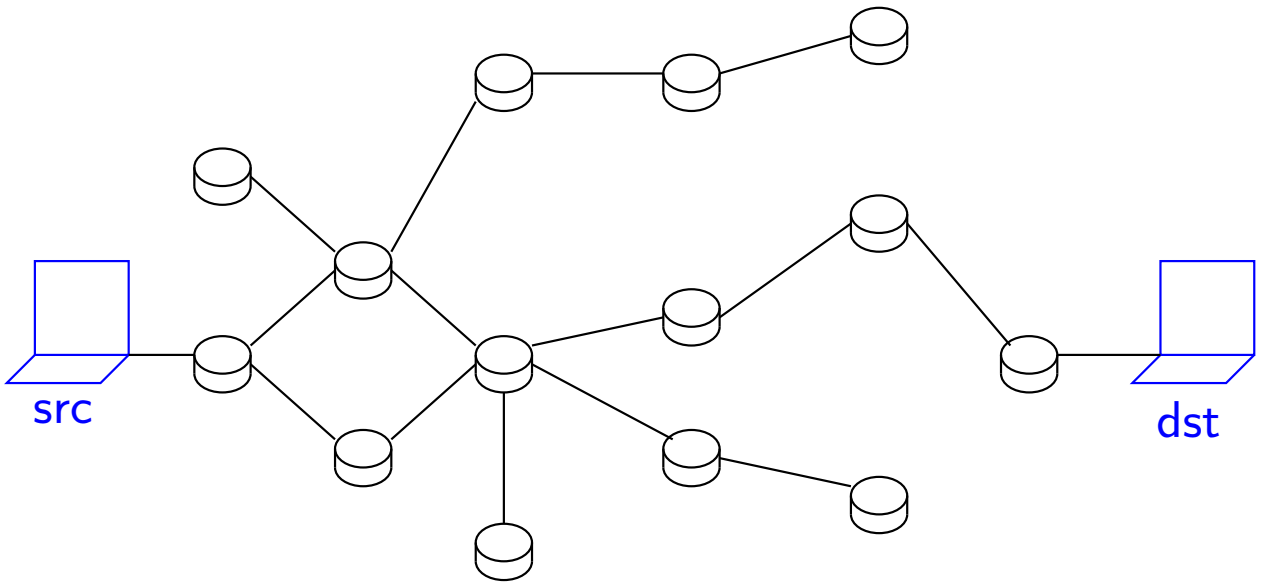
## Inconvénients

- Génère un *grand* nombre de messages
  - Infini puisque chaque noeud qui reçoit le message le transmet à tous ses voisins
  - Pour éviter ce problème : utilisation du TTL
- Envoi sur plusieurs routes en même temps : messages non-indispensables (beaucoup !)

Utilisation : systèmes critiques

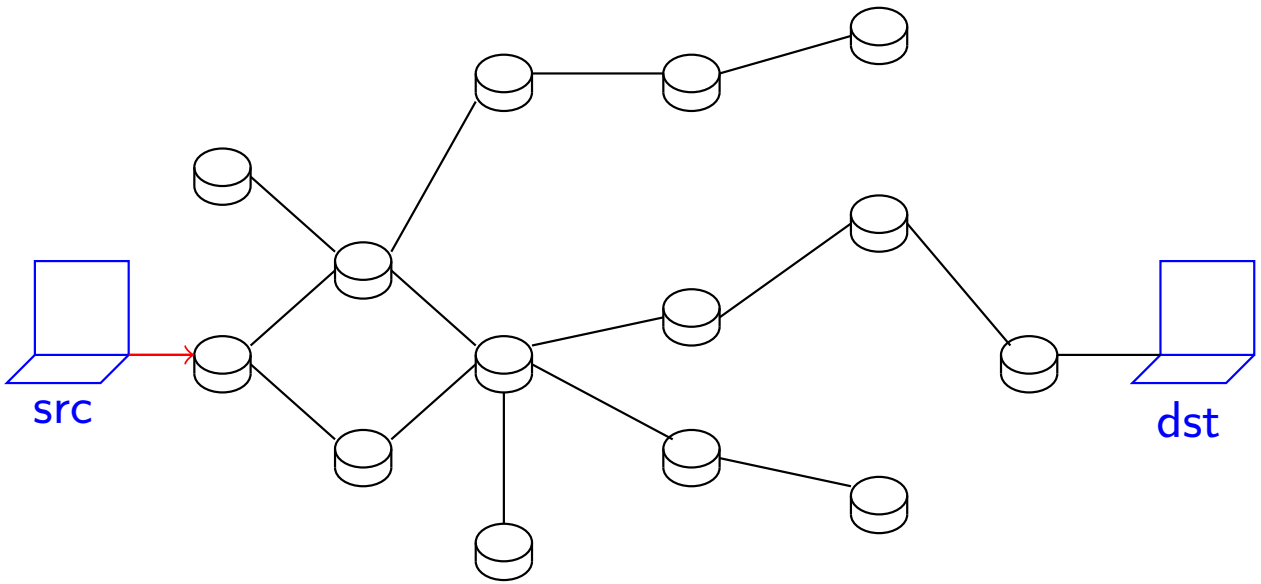
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



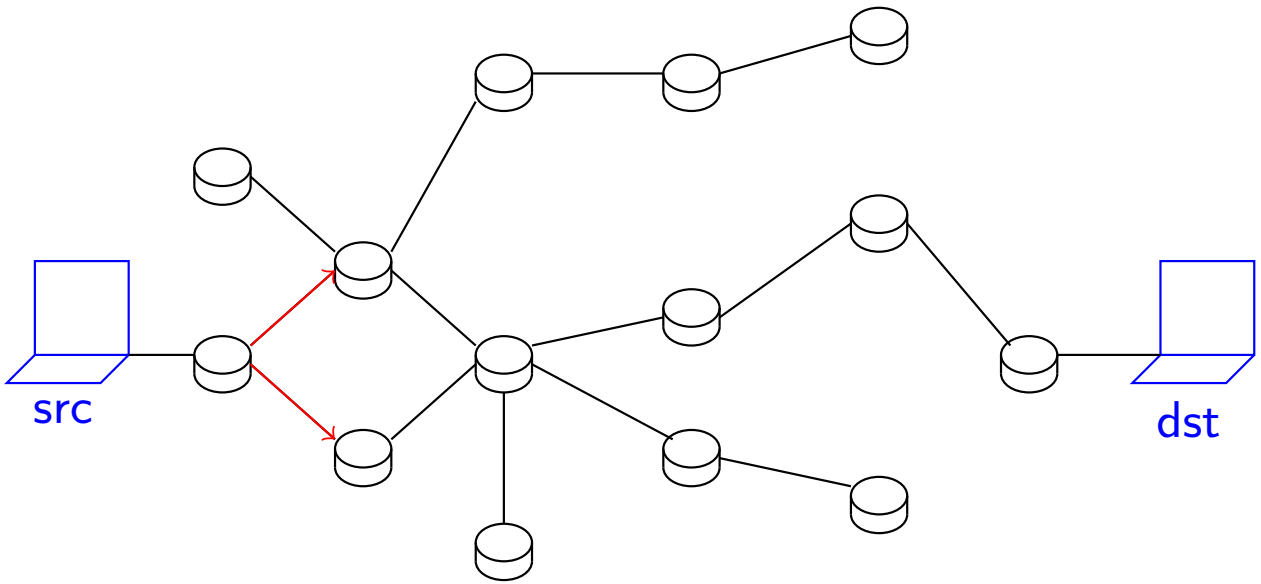
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



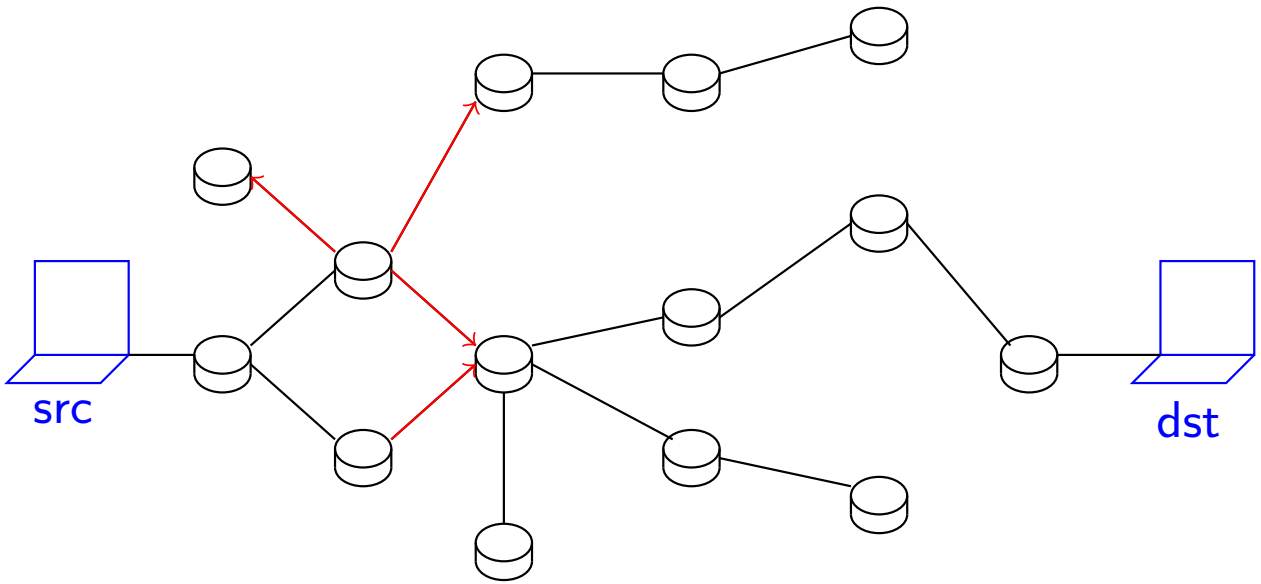
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



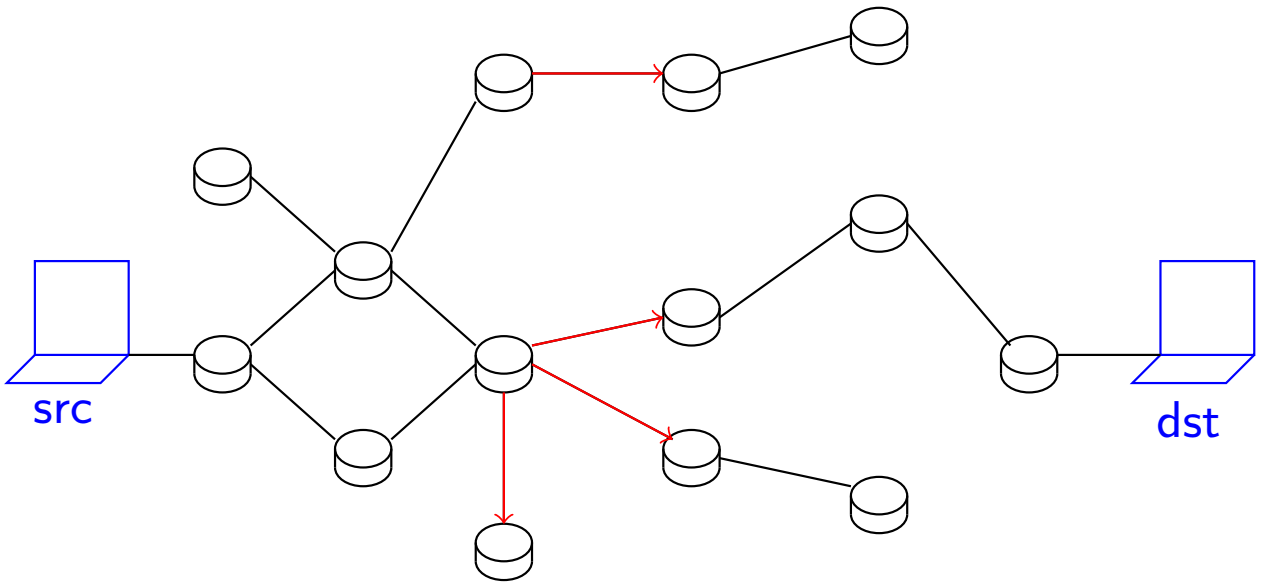
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



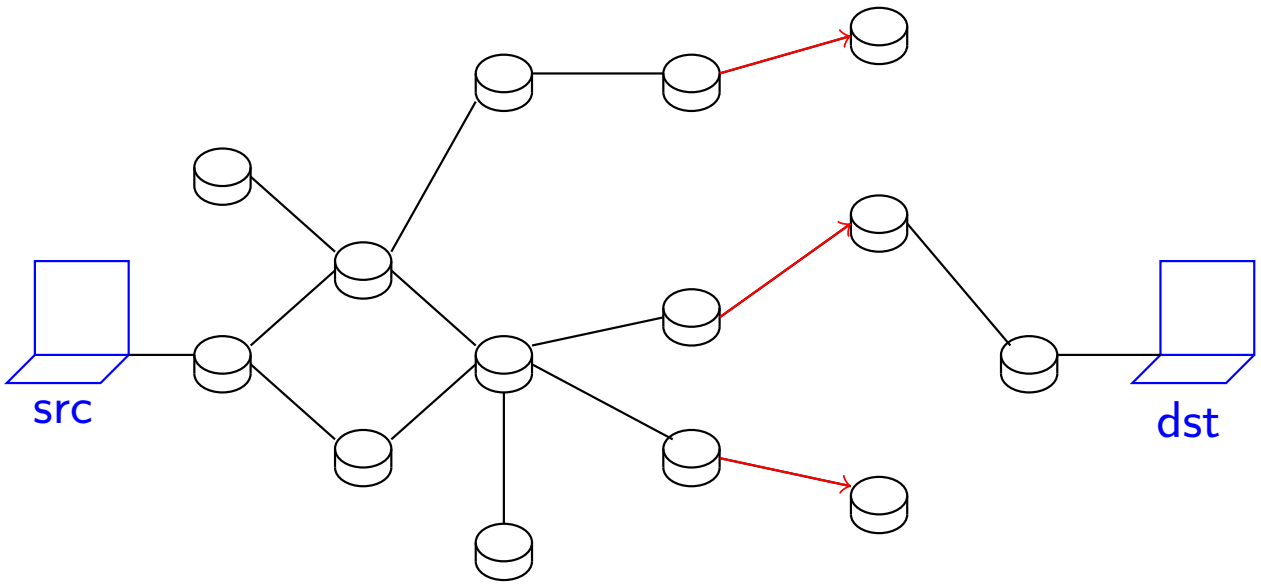
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



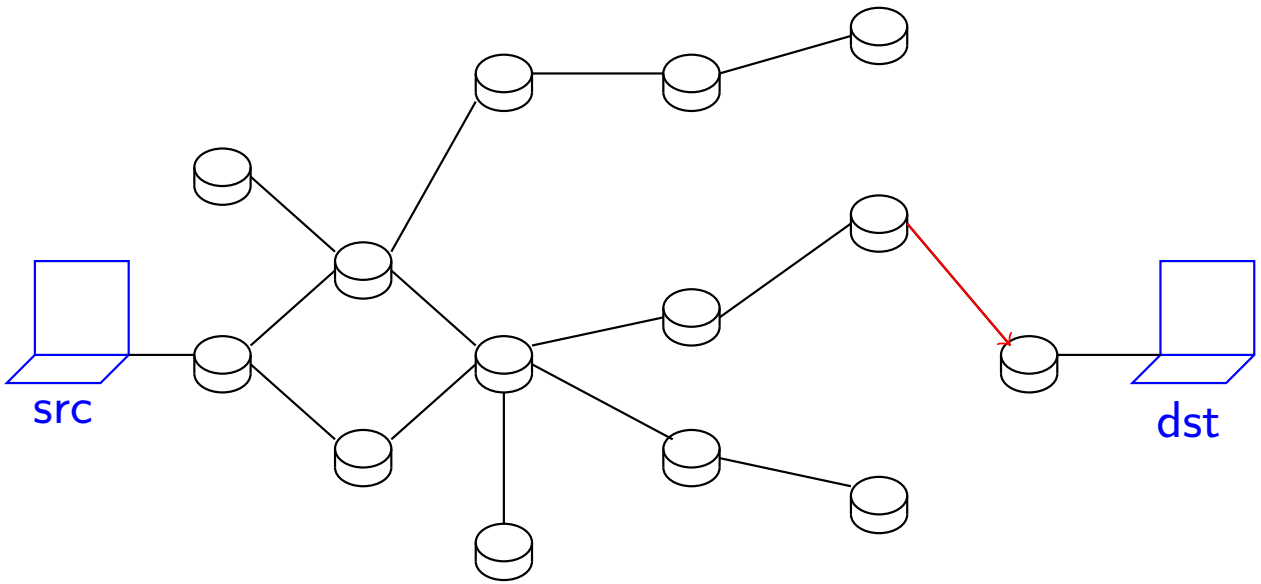
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



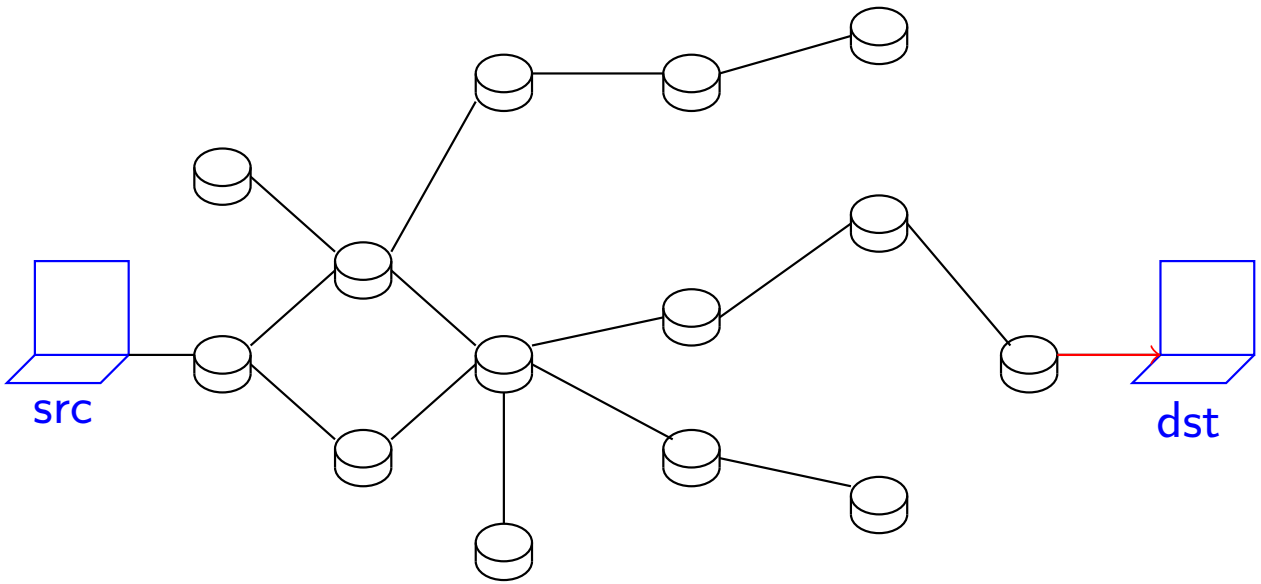
# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



# Exemple

Envoi d'un message de la machine *src* vers la machine *dst*



# Algorithme de la patate chaude

Principe : on se débarrasse le plus vite possible du paquet

- Envoi vers le lien le moins chargé
  - à l'exclusion du lien d'origine
- Algorithme *adaptatif* : tient compte de l'état du réseau

## Avantages

- Adaptatif
- Vise à diminuer la charge (on garde le paquet le moins longtemps possible)

## Inconvénients

- Non optimal en nombre d'étapes pour atteindre un destinataire
- L'arrivée n'est même pas garantie !
- Possibilité de création de boucles

## Évolution : routage réparti

Chaque noeud diffuse à ses voisins l'état de ses liens

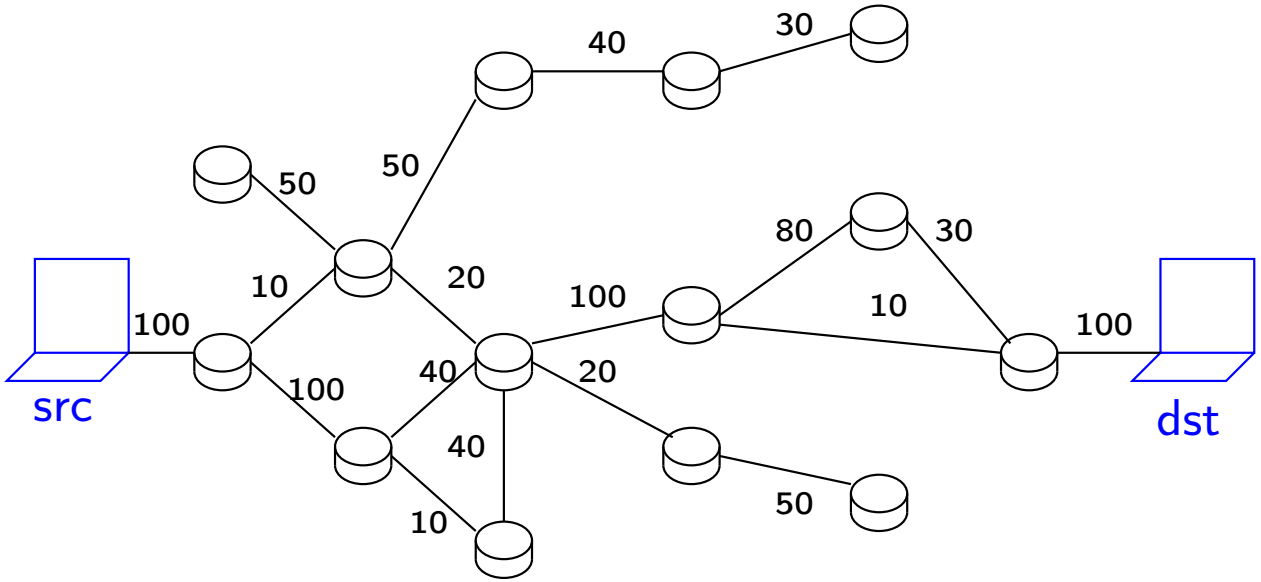
- Permet de calculer la route avec plus de visibilité

Utilisation : souvent combiné à un algorithme statique qui mémorise les routes

- Plus qu'un algorithme de routage, c'est un algorithme de sélection de route

# Exemple

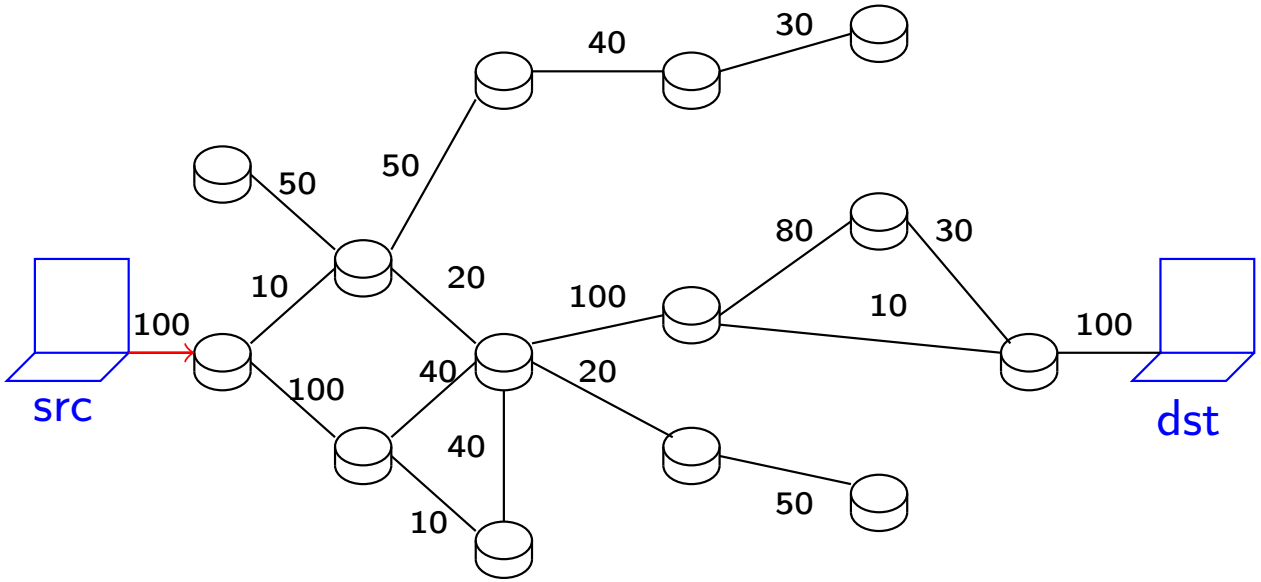
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

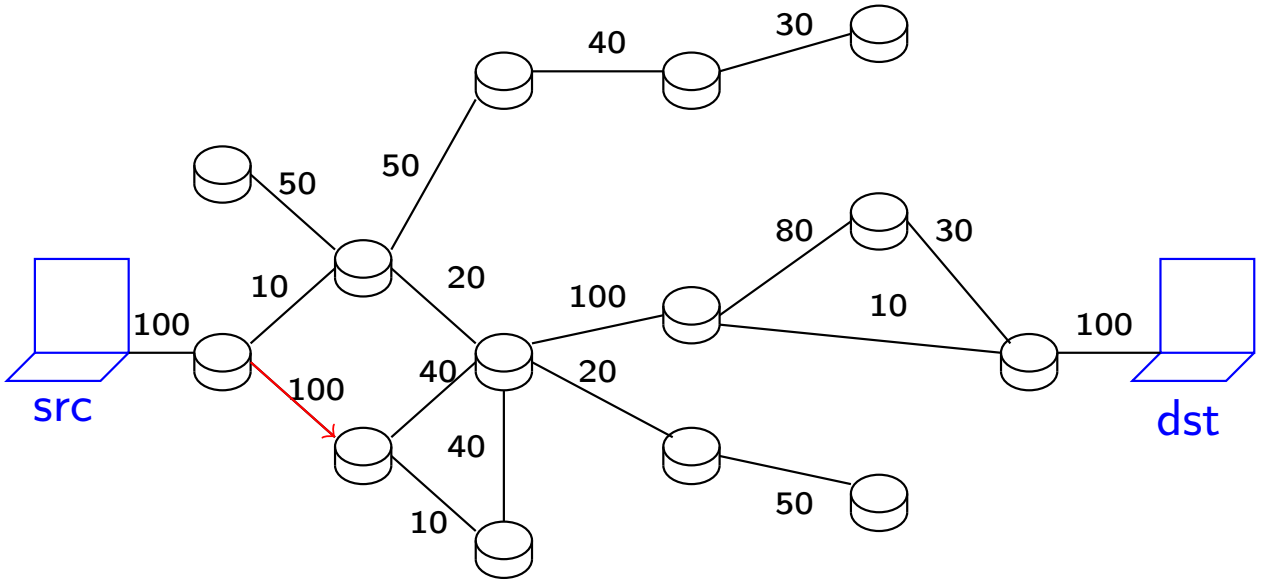
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

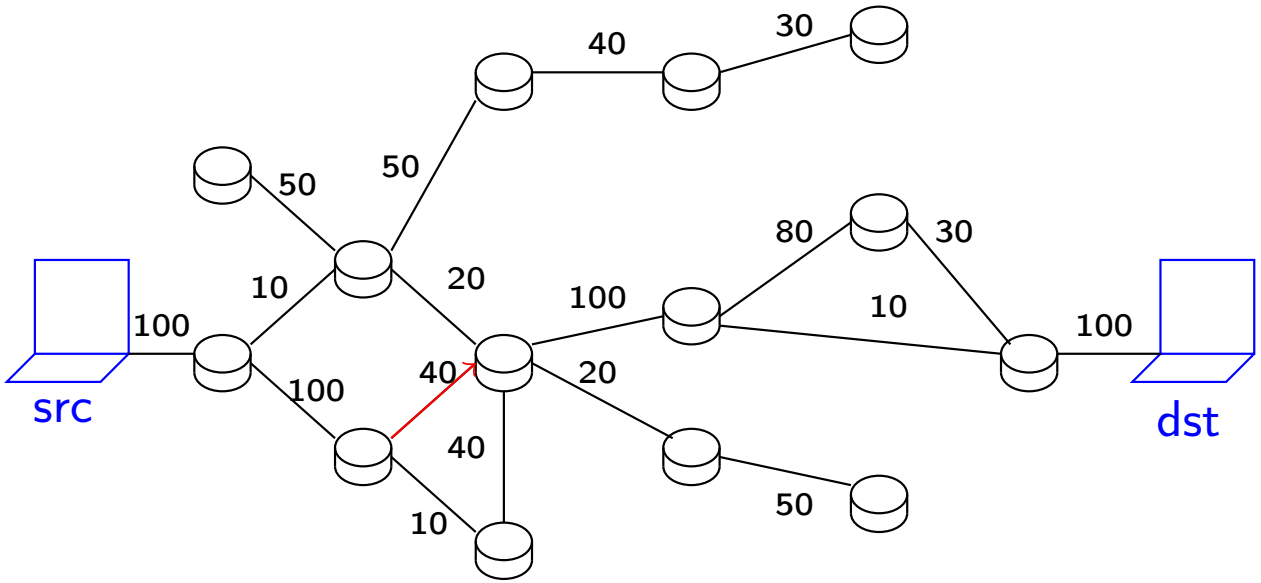
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

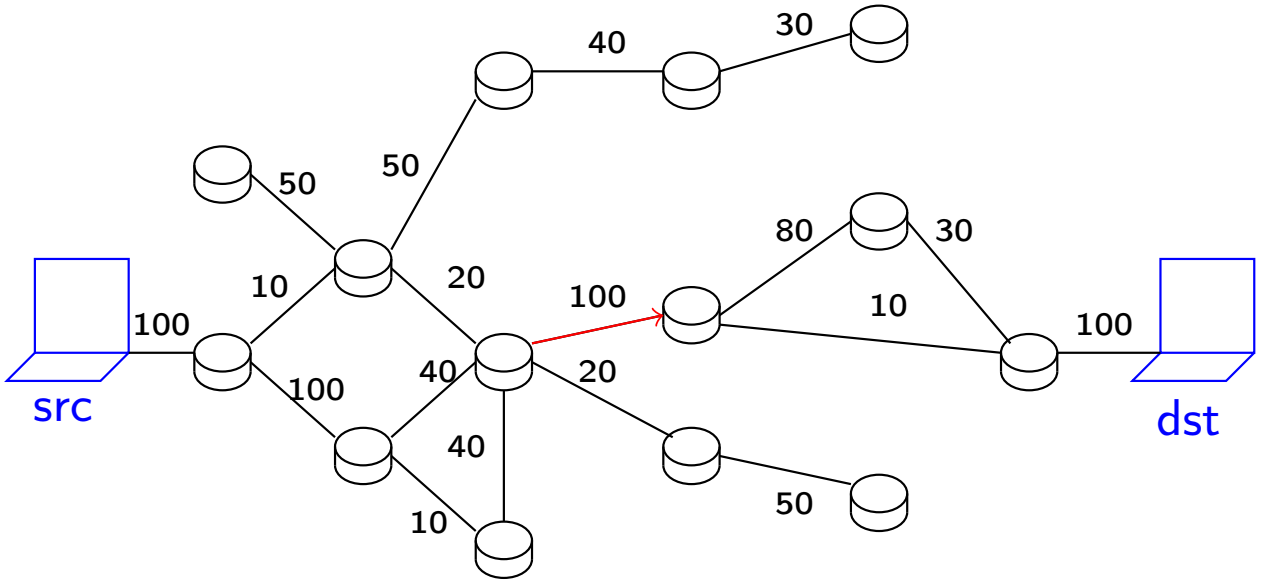
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

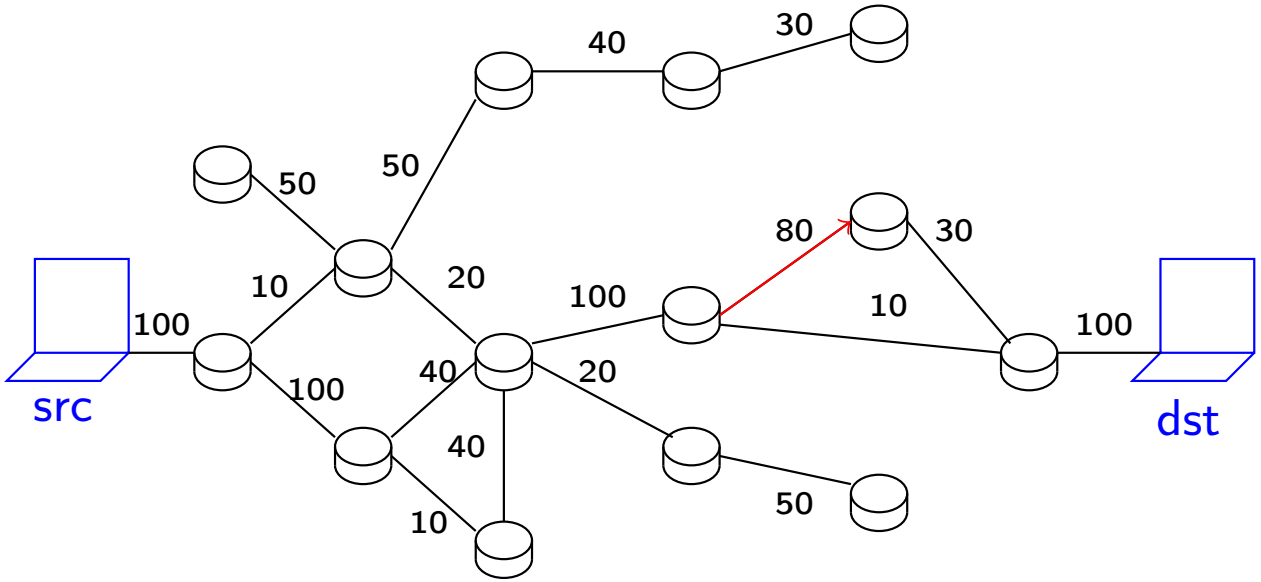
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

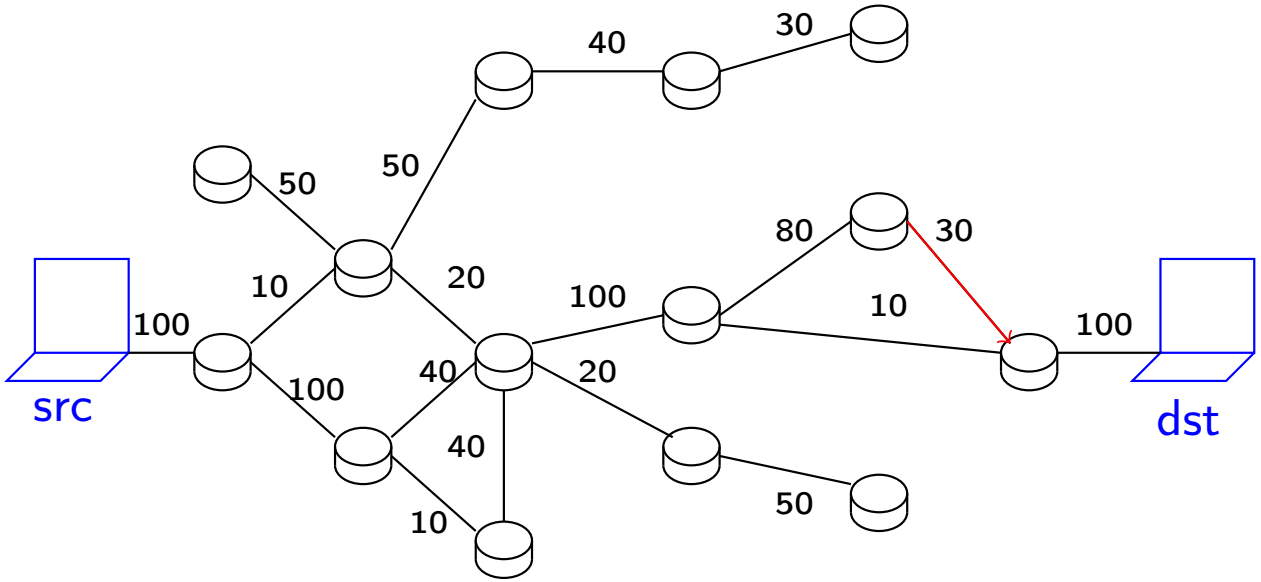
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

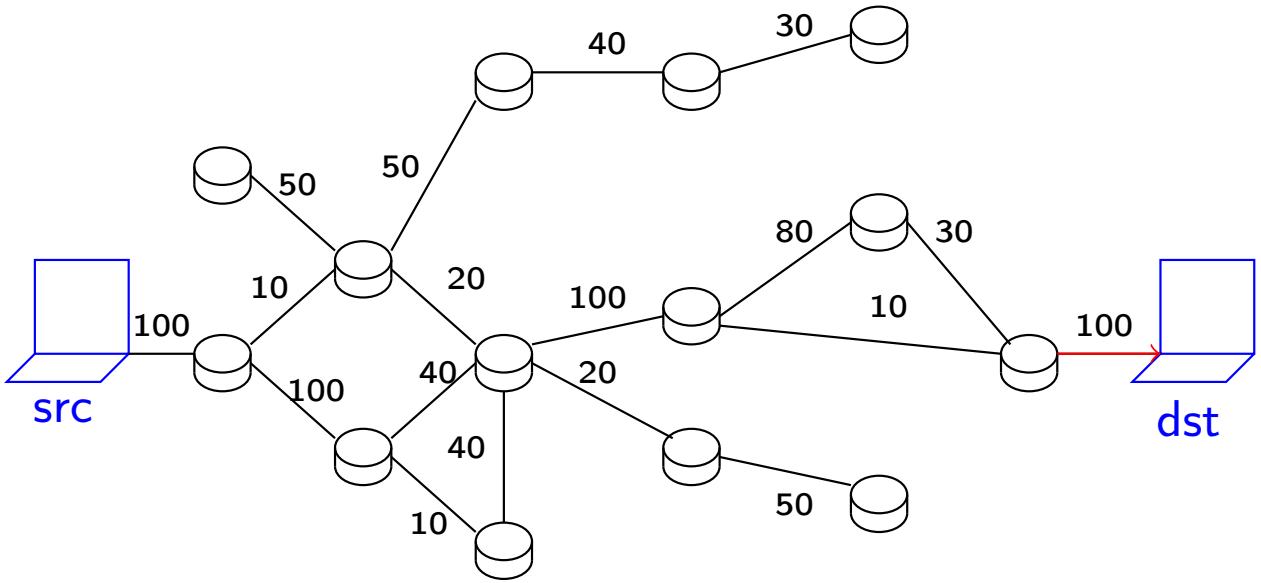
Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Exemple

Considérons les débits associés aux routes du réseau suivantes. La machine *src* envoie un message vers la machine *dst*.




NB : il s'agit dans cet exemple de débits associés aux liens. Il peut aussi s'agit de chargements des liens (on va alors vers le lien le moins chargé), de latence, etc.

# Algorithme à vecteurs de distance : Bellman-Ford

## Principe

Chaque routeur échange avec ses voisins des informations sur les tables de routage dont il dispose.

Informations : couple (destination, coût)

- À partir de ces informations, construction d'une table de routage selon le principe d'optimalité
- Coût = généralement nombre de sauts

Quand on ne connaît pas de route vers un destinataire :

- Coût initialisé à l'infini

Idem en cas de panne :

- Tous les destinataires ayant une route passant par un bouteur tombé en panne ont un coût mis à l'infini
- Adaptation des routes selon l'algorithme

Informations diffusées

- De proche en proche, de voisin en voisin
- À l'initialisation d'un routeur qui rejoint le réseau
- À intervalles réguliers (adaptativité)

# Algorithme à états de liens : Dijkstra

**Principe**

Chaque routeur échange avec ses voisins des informations sur l'état de ses liens.

Mesure des coûts de transmission d'un message avec ses voisins

- Généralement : latence

Diffusion des coûts à ses voisins

- Construction petit à petit d'une matrice des coûts sur le réseau
- Calcul du plus court chemin sur le réseau

Informations découvertes et diffusées

- De proche en proche, de voisin en voisin
- À l'initialisation d'un routeur qui rejoint le réseau et à intervalles réguliers (adaptativité)
- Ponctuelles : ne tiennent pas compte du trafic

# Plan du cours

# Protocoles internes

Routage au sein d'un réseau, entre les routeurs.

- Un protocole donné au sein d'un réseau
- Différents réseaux reliés entre eux peuvent ne pas utiliser le même protocole

**Routing Information Protocol (RIP) :**

- Routage s'appuyant sur l'algorithme de *Bellman-Ford*
- Objectif : minimiser le nombre de sauts effectués entre la source et le destinataire
- RFC 1058 (RIP 1), RFC 2080 (RIPng, support de IPv6), RFC 2453 (RIP 2)

Plutôt pour des petits réseaux : maximum 15 routeurs.

**Open Shortest Path First (OSPF) :**

- Routage s'appuyant sur l'algorithme de *Dijkstra*, s'affranchissant de la limite de RIP à 15 routeurs
- Objectif : minimiser le coût de communication entre 2 routeurs
- Routage *hiérarchique* : segmentation des ensemble de routeurs, notion d'aire
- ORFC 1131 (OSPF v1), RFC 2328 (OSPF v2), RFC 5340 (OSPF v3, support de IPv6)

# Routing Information Protocol

Chaque routeur annonce **périodiquement** (toutes les 30 secondes environ) tous ses réseaux et le **nombre de sauts** pour les atteindre

- Chaque machine et chaque routeur écoute les annonces de ses passerelles et actualise sa table de routage.
- Timeout : si au bout d'un certain temps un réseau n'est plus annoncé, il est supprimé de la table de routage
- Les modifications sont annoncées immédiatement.

# Routing Information Protocol

Chaque routeur annonce **périodiquement** (toutes les 30 secondes environ) tous ses réseaux et le **nombre de sauts** pour les atteindre

- Chaque machine et chaque routeur écoute les annonces de ses passerelles et actualise sa table de routage.
- Timeout : si au bout d'un certain temps un réseau n'est plus annoncé, il est supprimé de la table de routage
- Les modifications sont annoncées immédiatement.

Timers utilisés :

- **Entre deux annonces** : 30 secondes, plus un petit nombre de secondes tiré aléatoirement pour éviter que tous les routeurs s'annoncent en même temps.
- Pour considérer une route comme **invalide** : 90 secondes d'inactivité (aucune mise à jour ni annonce). Le routeur annonce à ses voisins que la route est invalide.
- Pour **retirer complètement** de la liste des routes : 240 secondes.

# Paquets RIP 1

Protocole utilisant UDP sur le port 520

*Taille des champs en bits*

| Commande (8) | Version (8) | Zéros (16) |
|---|---|---|
| ID de famille d'adresse (16) | | Zéros (16) |
| Adresse IP (32) | | |
| Zéros (32) | | |
| Zéros (32) | | |
| Métrique (32) | | |
| *Payload* | | |

- Commande :
  1. Request : demande tout ou partie d'une mise à jour de table d'un autre routeur RIP
  2. Response : réponse à une request. Toutes les mises à jour de routes utilisent cette commande
  3. Traceon : obsolète et ignoré
  4. Traceoff : obsolète et ignoré
  5. Reserved : Utilisé par les routeurs Sun
- Version : version du protocole RIP (1 ou 2)
- ID de famille d'adresse : identifie le protocole d'adressage utilisé (CLNS, IPX, IP...)
- Métrique : indique combien de routeurs ont été traversés. 1 à 15 : route valide, 16 = route impossible à atteindre

# Paquets RIP 2

*Taille des champs en bits*

| Commande (8) | Version (8) | Zéros (16) |
|---|---|---|
| ID de famille d'adresse (16) | | Route tag (16) |
| Adresse IP (32) | | |
| Masque de sous-réseau (32) | | |
| Saut suivant (32) | | |
| Métrique (32) | | |
| *Payload* | | |

- Route tag : permet de distinguer les routes entre elles, notamment interne et externes
- Saut suivant : adresse IP du prochain routeur auquel le paquet doit être envoyé
- Adresse IP, masque : ceux de l'entrée dont il est question
- Payload : Entrées de la table de routage, au maximum 25

# Fonctionnement de RIP

Algorithme à **vecteur de distance**

- *Itératif* : continue tant qu'il y a des informations à échanger
- *Asynchrone* : chaque noeud (routeur) est indépendant
- *Distribué* : aucun noeud n'a la vision complète du réseau

# Fonctionnement de RIP

Algorithme à **vecteur de distance**

- *Itératif* : continue tant qu'il y a des informations à échanger
- *Asynchrone* : chaque noeud (routeur) est indépendant
- *Distribué* : aucun noeud n'a la vision complète du réseau

Basé sur l'algorithme de **Bellman-Ford**

- Chaque noeud calcule une route depuis **chaque autres noeud** qui **minimise le coût**
- C'est un **arbre collecteur** pour chaque noeud

# Fonctionnement de RIP

Algorithme à **vecteur de distance**

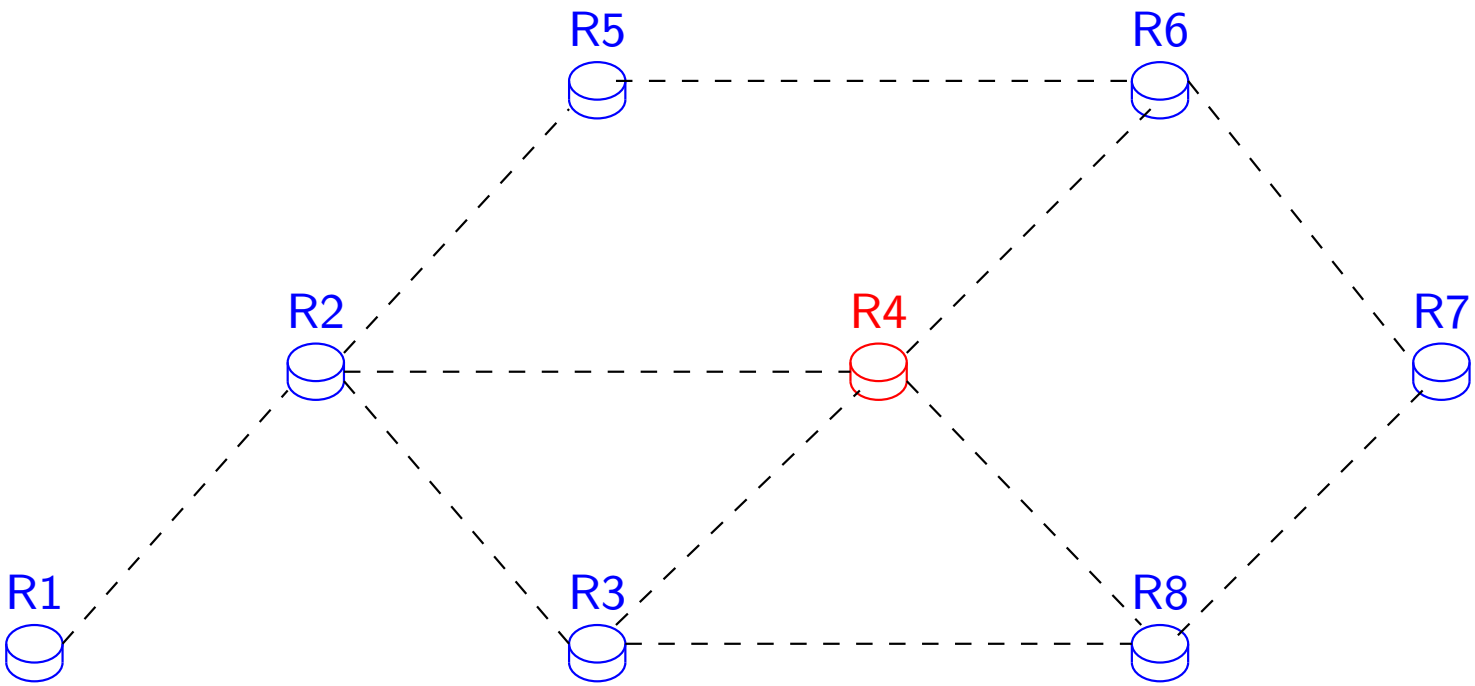
- *Itératif* : continue tant qu'il y a des informations à échanger
- *Asynchrone* : chaque noeud (routeur) est indépendant
- *Distribué* : aucun noeud n'a la vision complète du réseau

Basé sur l'algorithme de **Bellman-Ford**

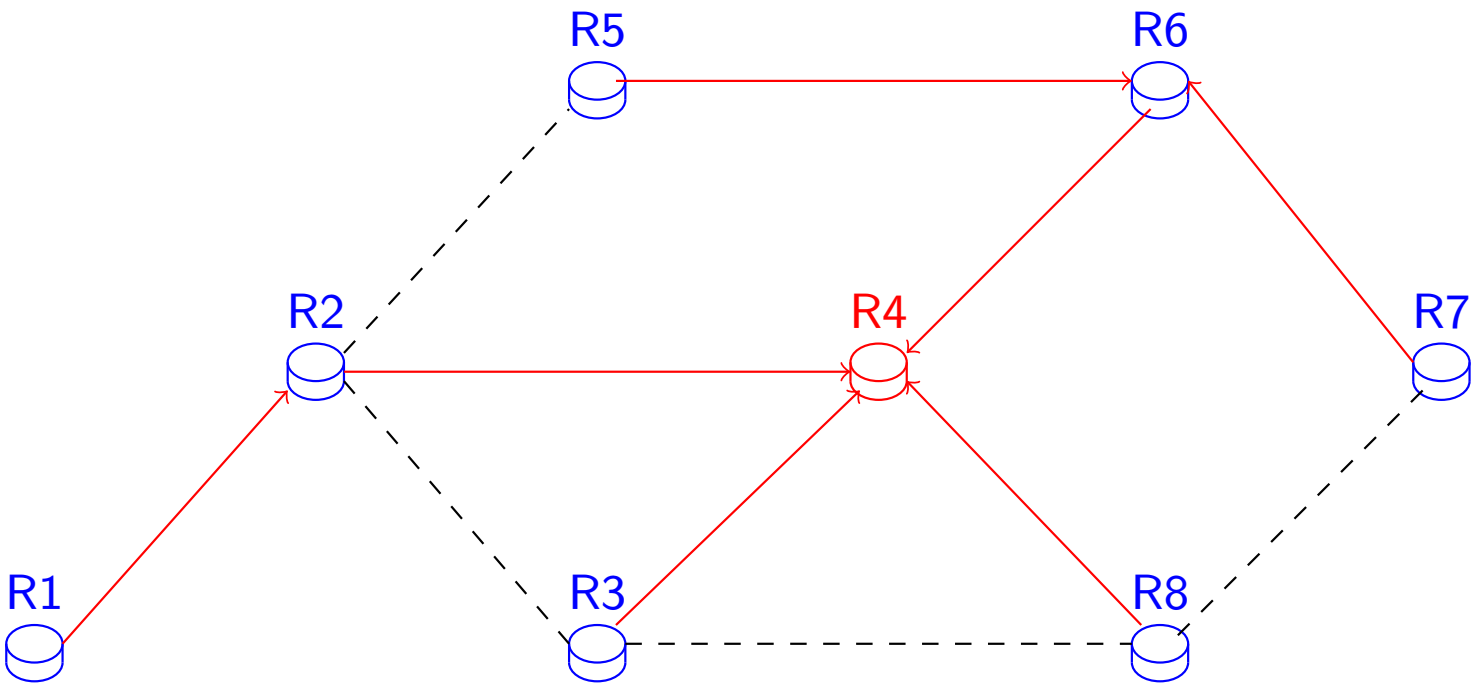
- Chaque noeud calcule une route depuis **chaque autres noeud** qui **minimise le coût**
- C'est un **arbre collecteur** pour chaque noeud

→ Chaque noeud du réseau connaît le **plus court chemin** pour atteindre chaque autre noeud

# RIP : arbre collecteur



# RIP : arbre collecteur



# Fonctionnement de RIP

**Au démarrage** :

- Un routeur RIP diffuse une requête pour demander à ses voisins leurs tables de routage

# Fonctionnement de RIP

**Au démarrage** :

- Un routeur RIP diffuse une requête pour demander à ses voisins leurs tables de routage
- Les voisins envoient une réponse contenant leurs tables de routage
  - Si on reçoit une nouvelle route : on l'insère dans la table de routage
  - Si on reçoit une meilleure route : on remplace la route existante par la nouvelle

# Fonctionnement de RIP

**Au démarrage** :

- Un routeur RIP diffuse une requête pour demander à ses voisins leurs tables de routage
- Les voisins envoient une réponse contenant leurs tables de routage
  - Si on reçoit une nouvelle route : on l'insère dans la table de routage
  - Si on reçoit une meilleure route : on remplace la route existante par la nouvelle
- Le routeur qui vient de démarrer envoie alors une mise à jour contenant sa table de routage
  - Ses voisins peuvent mettre à jour leurs tables de routage en cas de nouvelle ou de meilleure route

# Fonctionnement de RIP

**Fonctionnement normal** :

- Chaque routeur annonce périodiquement les réseaux auxquels il est relié
- Ces messages servent à annoncer qu'on est toujours en vie
- Les messages ne sont pas acquittés, RIP fonctionne sur UDP

# Fonctionnement de RIP

**Fonctionnement normal** :

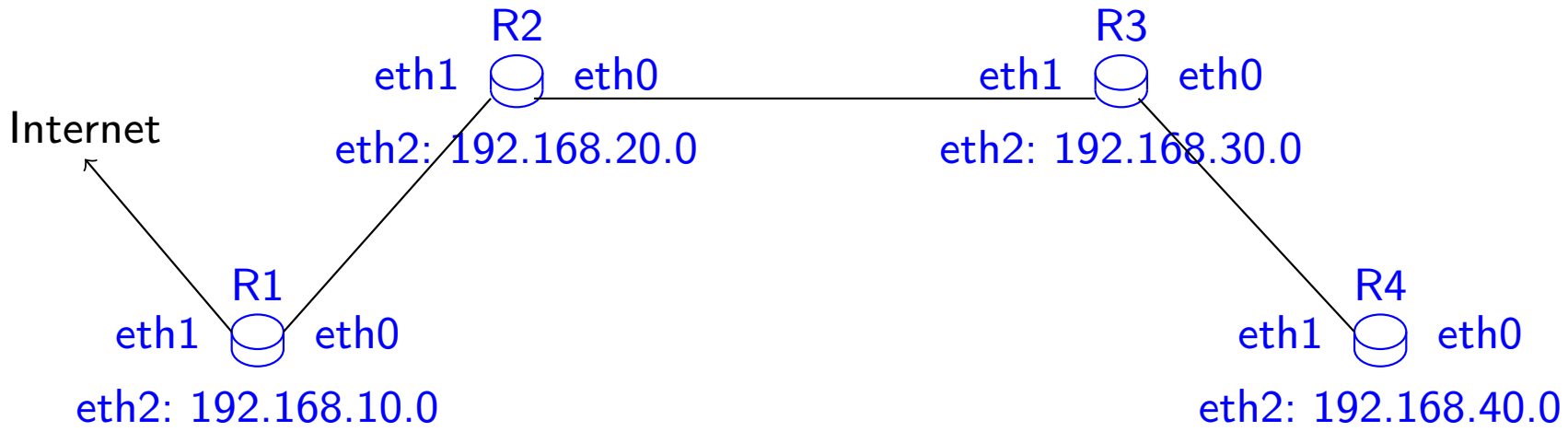
- Chaque routeur annonce périodiquement les réseaux auxquels il est relié
- Ces messages servent à annoncer qu'on est toujours en vie
- Les messages ne sont pas acquittés, RIP fonctionne sur UDP

**En cas de panne** :

- Si un voisin ne donne plus de signe de vie :
  - On ne reçoit plus ses annonces
  - 1er timeout : la route est considérée comme invalide
  - 2eme timeout : la route est complètement supprimée
- Attention : ceci est une modification de la table
  → il faut propager la mise à jour

# Exemple

Considérons le réseau suivant :




Extraits des tables de routage (simplifiées) :

R1 :

| Adresse | Interface |
|---|---|
| 192.168.10.0 | eth2 |
| 192.168.20.0 | eth0 |
| 192.168.30.0 | eth0 |
| 192.168.40.0 | eth0 |
| * | eth1 |

R2 :

| Adresse | Interface |
|---|---|
| 192.168.20.0 | eth2 |
| 192.168.30.0 | eth0 |
| 192.168.40.0 | eth0 |
| * | eth1 |

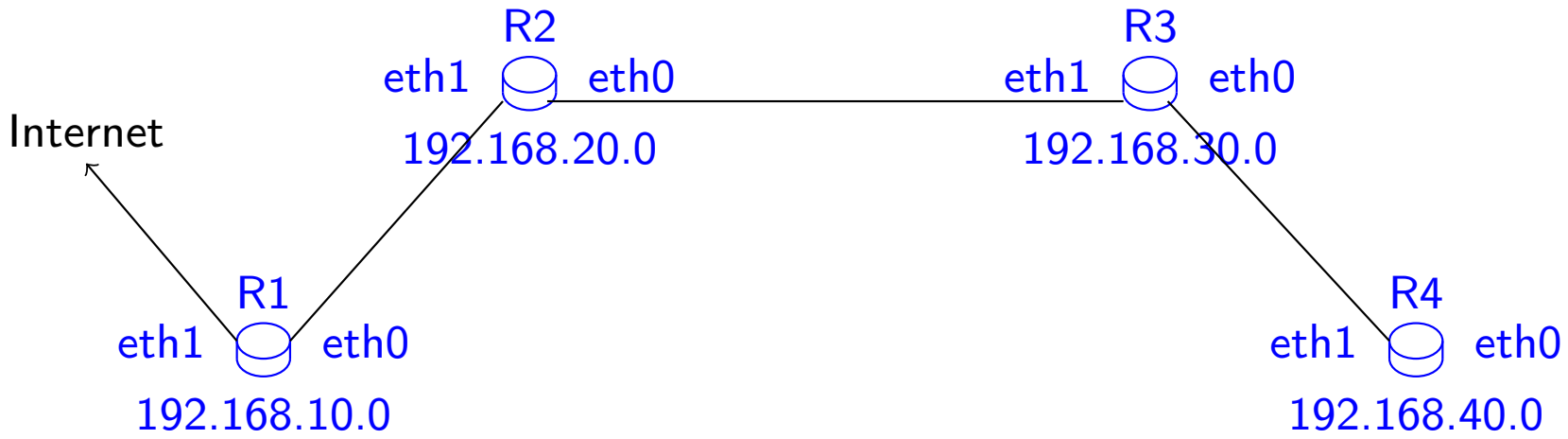
R3 :

| Adresse | Interface |
|---|---|
| 192.168.30.0 | eth2 |
| 192.168.40.0 | eth0 |
| * | eth1 |

R4 :

| Adresse | Interface |
|---|---|
| 192.168.40.0 | eth2 |
| * | eth1 |

# Exemple

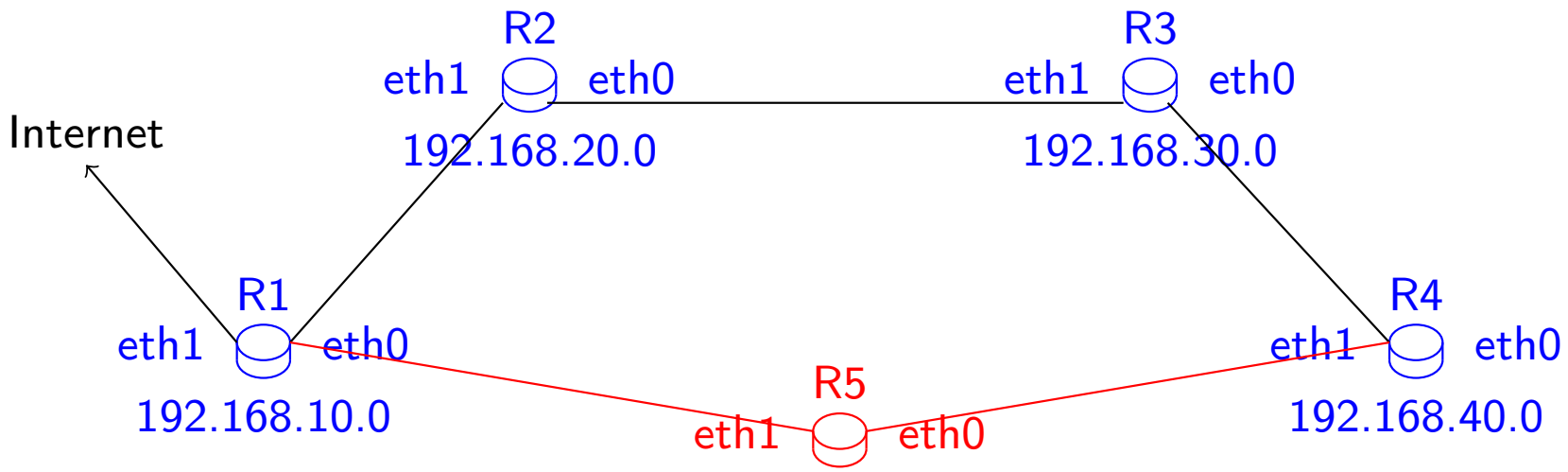



Nombre de sauts nécessaires entre deux routeurs :

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| R1 | 0 | 1 | 2 | 3 |
| R2 | 1 | 0 | 1 | 2 |
| R3 | 2 | 1 | 0 | 1 |
| R4 | 3 | 2 | 1 | 0 |

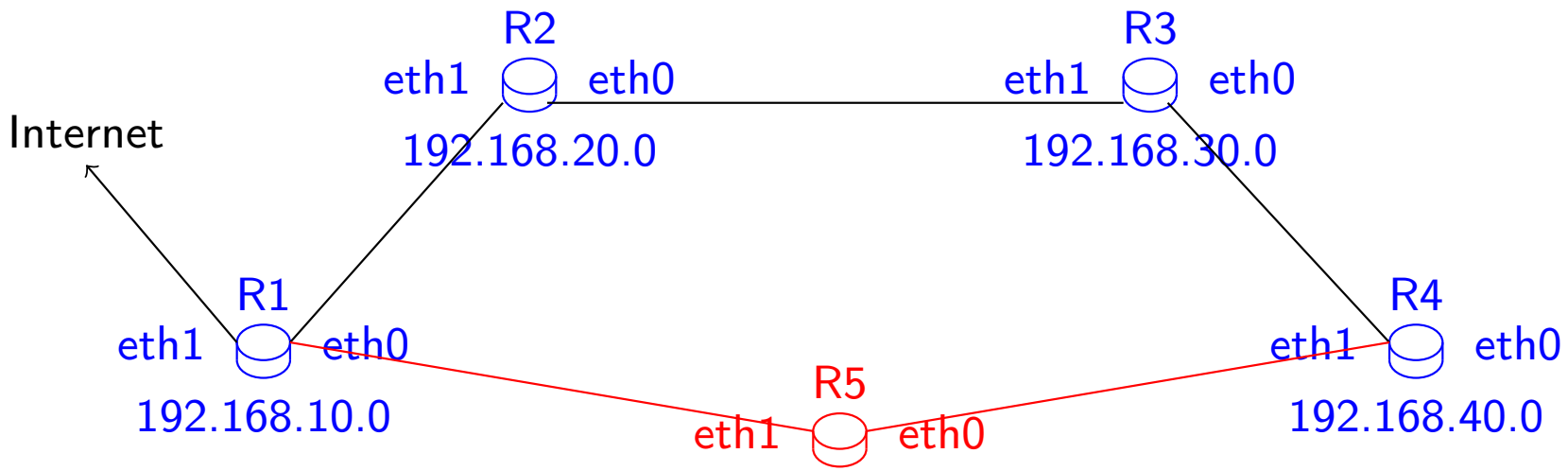
# Exemple

On ajoute un routeur R5 dans le réseau :



# Exemple

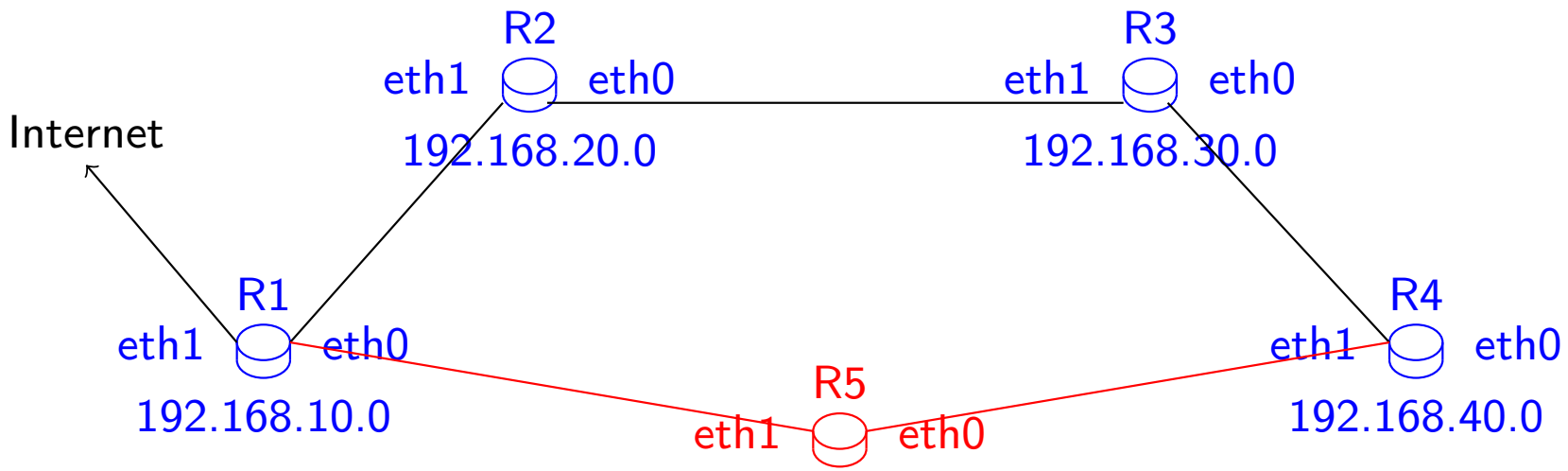
On ajoute un routeur R5 dans le réseau :




- R5 envoie une requête à ses voisins : R1 et R4

# Exemple

On ajoute un routeur R5 dans le réseau :




- R5 envoie une requête à ses voisins : R1 et R4
- R1 et R4 lui envoient une réponse contenant leurs tables de routage avec le nombre de sauts nécessaires, qui est incrémenté de 1

Reçu de R1 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 1 | R1 |
| 192.168.20.0 | 2 | R1 |
| 192.168.30.0 | 3 | R1 |
| 192.168.40.0 | 4 | R1 |
| * | 2 | R1 |

Reçu de R4 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.40.0 | 1 | R4 |
| 192.168.30.0 | 2 | R4 |
| 192.168.20.0 | 3 | R4 |
| 192.168.10.0 | 4 | R4 |
| * | 5 | R4 |

# Exemple

R5 choisit les routes les plus courtes pour atteindre chaque réseau :

| Réseau | Nb de sauts par R1 | Nb de sauts par R4 |
|---|---|---|
| 192.168.10.0 | 1 | 4 |
| 192.168.20.0 | 2 | 3 |
| 192.168.30.0 | 3 | 2 |
| 192.168.40.0 | 3 | 1 |
| * | 2 | 5 |

# Exemple

R5 choisit les routes les plus courtes pour atteindre chaque réseau :

| Réseau | Nb de sauts par R1 | Nb de sauts par R4 | Passerelle choisie |
|---|---|---|---|
| 192.168.10.0 | 1 | 4 | → R1 (1 saut) |
| 192.168.20.0 | 2 | 3 | → R1 (2 sauts) |
| 192.168.30.0 | 3 | 2 | → R4 (2 sauts) |
| 192.168.40.0 | 3 | 1 | → R4 (1 saut) |
| * | 2 | 5 | → R1 (2 sauts) |

Et R5 envoie sa table de routage avec le nombre de sauts à ses voisins R1 et R4 pour qu'ils actualisent les leurs :

- De **nouvelles routes** ont été découvertes en passant par R5

# Exemple

R5 choisit les routes les plus courtes pour atteindre chaque réseau :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 1 | R1 |
| 192.168.20.0 | 2 | R1 |
| 192.168.30.0 | 2 | R4 |
| 192.168.40.0 | 1 | R4 |
| * | 2 | R1 |

Et R5 envoie sa table de routage avec le nombre de sauts à ses voisins R1 et R4 pour qu'ils actualisent les leurs :

- De **nouvelles routes** ont été découvertes en passant par R5

# Exemple

Table des distances de R1 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 0 | – |
| 192.168.20.0 | 1 | R2 |
| 192.168.30.0 | 2 | R2 |
| 192.168.40.0 | 3 | R2 |
| * | 1 | gw |

Reçu de R5 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 2 | R5 |
| 192.168.20.0 | 3 | R5 |
| 192.168.30.0 | 3 | R5 |
| 192.168.40.0 | 2 | R5 |
| * | 3 | R1 |

# Exemple

Table des distances de R1 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 0 | – |
| 192.168.20.0 | 1 | R2 |
| 192.168.30.0 | 2 | R2 |
| 192.168.40.0 | 3 | R2 |
| * | 1 | gw |

Reçu de R5 :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 2 | R5 |
| 192.168.20.0 | 3 | R5 |
| 192.168.30.0 | 3 | R5 |
| 192.168.40.0 | 2 | R5 |
| * | 3 | R1 |

On **compare** les distances correspondant aux routes et on prend **la plus courte** pour atteindre **chaque réseau** :

| Réseau | Nombre de sauts | Passerelle |
|---|---|---|
| 192.168.10.0 | 0 | – |
| 192.168.20.0 | 1 | R2 |
| 192.168.30.0 | 2 | R2 |
| 192.168.40.0 | 2 | R5 |
| * | 1 | gw |

En passant par R5, le routeur R1 peut atteindre R4 en deux sauts (et inversement)

- Même chose pour R4
- R1 et R4 ont modifié leurs tables de routage
  - → il faut **propager la mise à jour**
  - → l'algorithme s'arrête (on dit qu'il *converge*) une fois qu'il n'y a **plus de mises à jour à propager**
- Comme R1 et R4 modifient leurs tables de routage, ils **envoient une mise à jour** à leurs voisins, respectivement R2 et R3.

# Plan du cours

# Agrégation de réseaux

Internet est fait d'une agrégation de réseaux de différents opérateurs :

- Chaque opérateur route en interne ce qui reste dans son réseau
- Protocole de routage entre les réseaux de différents opérateurs

AS : **Autonomous System**

- Ensemble de réseaux contrôlés par une seule autorité
- C'est un réseau, connecté à d'autres AS : ainsi est formé Internet
- AS = l'unité de routage sur Internet
- Une autorité contrôle un grand nombre d'AS

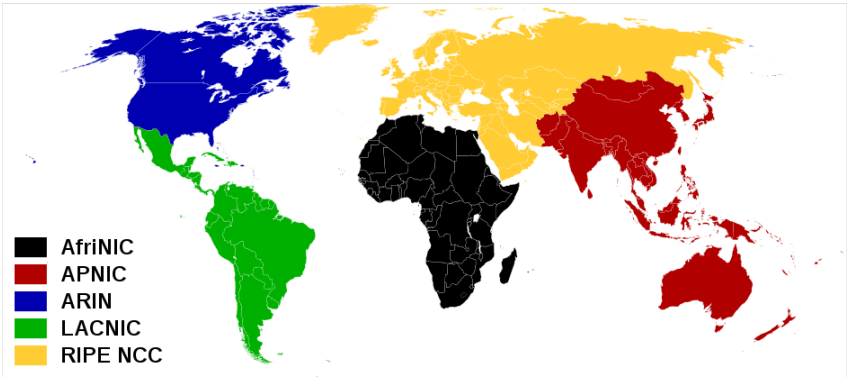
Cette autorité est **régionale** : il y en a 5, appelées **Regional Internet Registry (RIR)**

- Europe élargie : RIPE NCC (Réseaux IP Européens Network Coordination Centre)
- Afrique : AfriNIC
- Amérique du nord : ARIN
- Amérique du sud, caraïbes : LACNIC
- Asie-Pacifique : APNIC

# Rôle d'un Regional Internet Registry

Allocation des ressources dans sa région

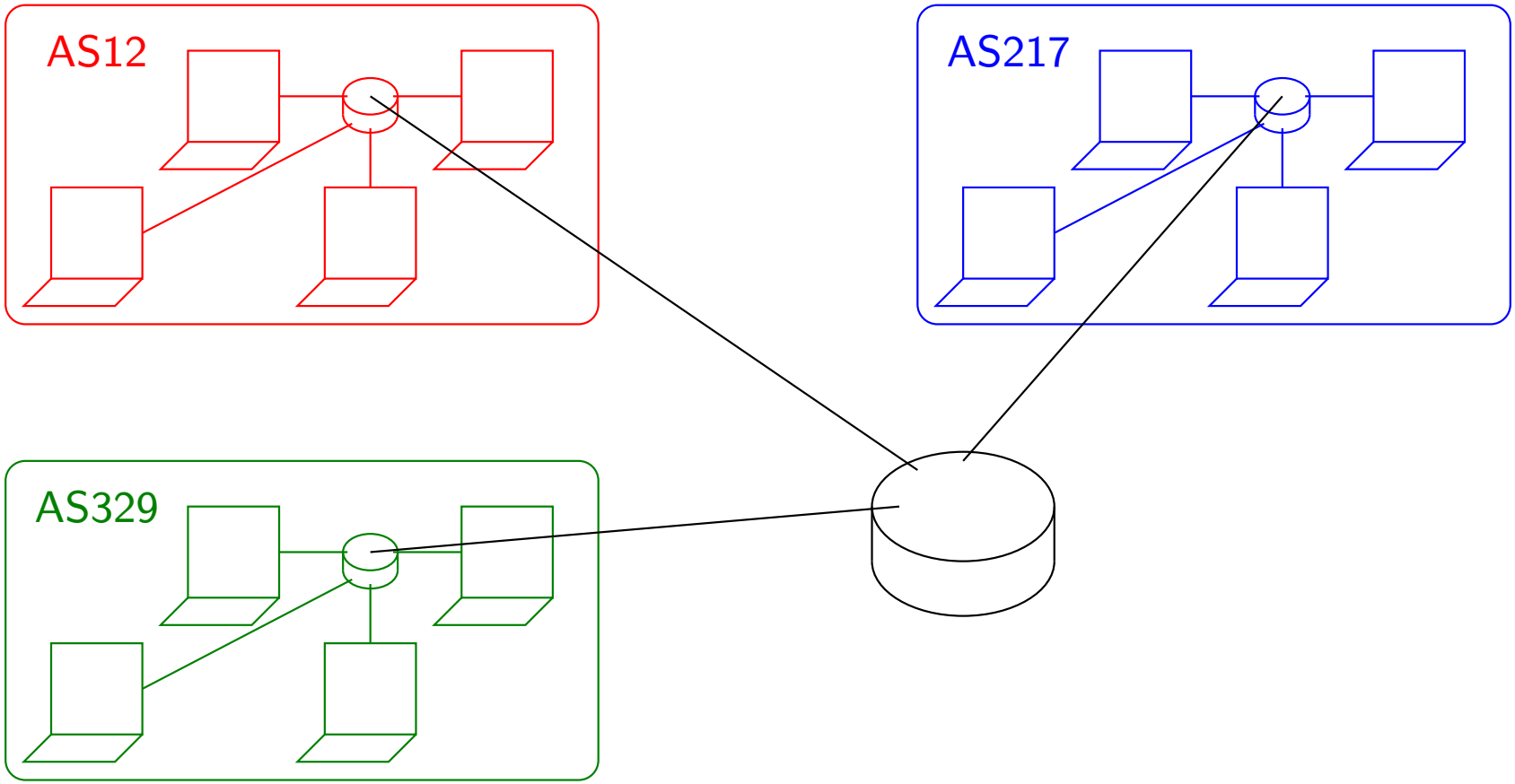
- Adresses IPv4 et IPv6
- Numéros d'AS



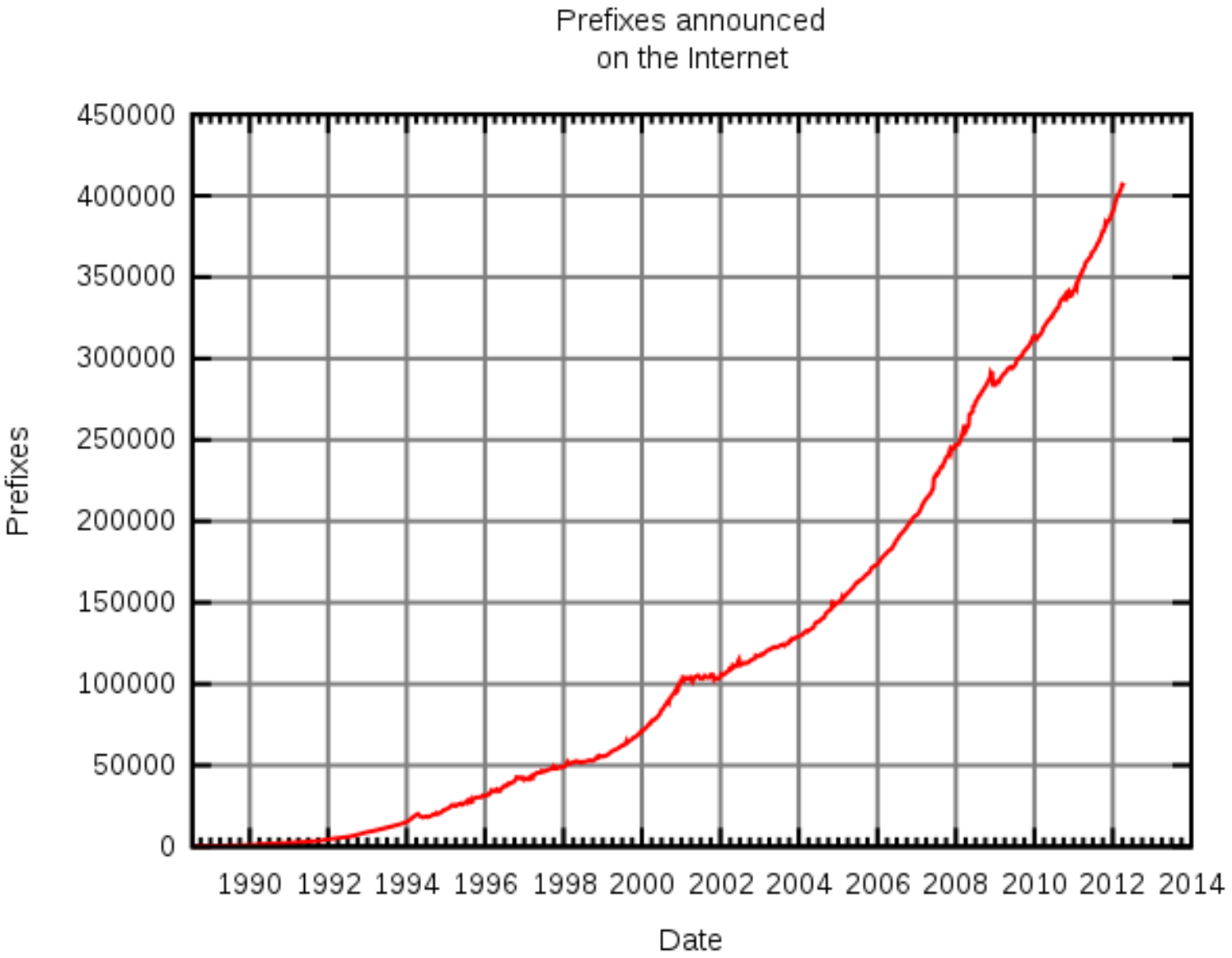

Autres activités :

- Maintenance de la base de données d'infos sur les réseaux de sa zone
- Maintenance des tables de routage à l'intérieur de sa zone
- Serveurs de noms racine (en Europe : K-root)
- Statistiques sur le réseau, ses performances et son développement

# Agrégation de réseaux



# Taille des tables de routage BGP sur Internet




*Source : Wikipedia*

# Architecture d'Internet

Internet est découpé en systèmes autonomes

- Routage entre les systèmes autonomes : **Exterior Gateway Protocol (EGP)**
- Puis routage au sein d'un système autonome : **Interior Gateway Protocol (IGP)**

Les systèmes autonomes (AS) sont identifiés par un *numéro de système autonome*

- 2 octets puis 4 octets
- Défini par la RFC 1771 (mars 1995), mis à jour par la RFC 4893 (mai 2007)
- Alloué par le RIR

Routage sur Internet : combinaison de protocoles

- Protocoles externes (EGP) entre les AS (BGP)
  - Granularité : l'AS
- Protocoles internes (IGP) au sein de chaque AS (RIP, OSPF)
  - Granularité : le routeur

# Administration : l'AS

```
coti@thorim:~$  whois -h whois.ripe.net AS1303
% Information related to 'AS1299 - AS1309'

as-block:       AS1299 - AS1309
descr:          RIPE NCC ASN block
remarks:        These AS Numbers are assigned to network operators in the RIPE NCC service
mnt-by:         RIPE-NCC-HM-MNT
source:         RIPE # Filtered

% Information related to 'AS1303'

% No abuse contact registered for AS1303

aut-num:        AS1303
as-name:        FR-IDRIS-ORSAY
descr:          FR
import:         from AS2200 action pref=100; accept ANY
export:         to AS2200 announce AS1303
default:        to AS2200 action pref=10; networks ANY
admin-c:        VA401-RIPE
tech-c:         GG93-RIPE
mnt-by:         RENATER-MNT
source:         RIPE # Filtered

person:         ........
address:        IDRIS-CNRS
```

# Internet Exchange Point

Internet Exchange Point = Network Access Point (ancien terme)

- Points d'accès aux liens haut débit d'Internet (longues distances)
- Généralement : relient les fournisseurs d'accès (Autonomous Systems) à Internet
- Plusieurs AS reliés à un IXP → relie des AS entre eux

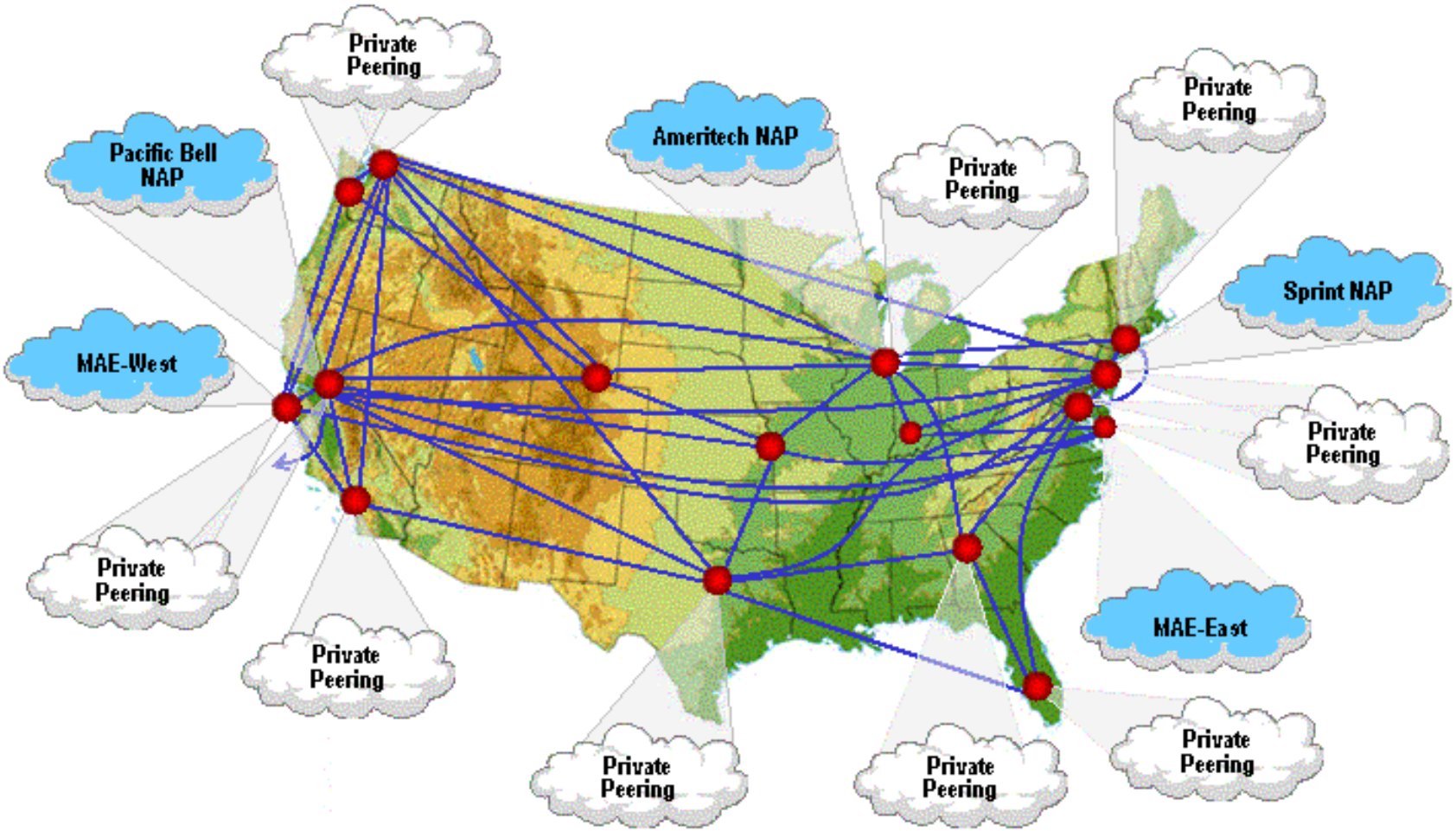
Historiquement, les IXP reliaient un réseau au backbone d'Internet

Plus gros IXP (par débit) :

| Nom | Pays | Création | Nombre de membres |
|---|---|---|---|
| DE-CIX | Allemagne | 1995 | > 600 |
| AMS-IX | Pays Bas | 1997 | 649 |
| LINX | Royaume Unis | 1994 | 503 |
| Equinix | USA, Europe, Asie | 1998 | 768 |
| MSK-IX | Russie | 1995 | 605 |

*Source : Wikipedia* `http://en.wikipedia.org/wiki/List_of_Internet_exchange_points_by_size`

# Internet Exchange Point




*Source : http://www.infocellar.com/networks/internet/nap-ixp.htm*

# Communications aux IXP

Communication entre AS : **peering**

- Deux AS qui ont besoin de communiquer peuvent être **connectés au même IXP**
- On dit alors que l'IXP est un **point de peering**
- **Modèle économique :**
  - Soit le trafic est symétrique : alors pas de facturation
  - Soit le trafic va plutôt d'un AS vers l'autre : possibilité de facturation (ou pas...)
- Important : aux points de peering sont aussi échangées des **informations de routage**

# Communications aux IXP

Communication entre AS : **peering**

- Deux AS qui ont besoin de communiquer peuvent être **connectés au même IXP**
- On dit alors que l'IXP est un **point de peering**
- **Modèle économique :**
  - Soit le trafic est symétrique : alors pas de facturation
  - Soit le trafic va plutôt d'un AS vers l'autre : possibilité de facturation (ou pas...)
- Important : aux points de peering sont aussi échangées des **informations de routage**

Communication à travers un AS : **transit**

- Les AS ne sont **pas tous connectés directement**
- Si deux AS non connectés directement doivent communiquer, le trafic peut **transiter par un autre AS**
- **Modèle économique :**
  - Le transit est un **service** fourni par l'opérateur à ses clients
  - L'AS (l'opérateur) par lequel transite le trafic facture à l'AS source (c'est lui qui a besoin de faire transiter)

# Type d'opérateurs Internet

Il existe trois types d'opérateurs Internet, suivant leur niveau de participation :

## Opérateurs Tier I

- N'achètent pas de transit
- Servent uniquement à acheminer du trafic entre AS sur Internet (opérateurs Tier II et Tier III)
- Plutôt des liaisons longue distance, haut débit
- Exemples : Sprint, AT&T, Cogent... (environ 14 en tout)

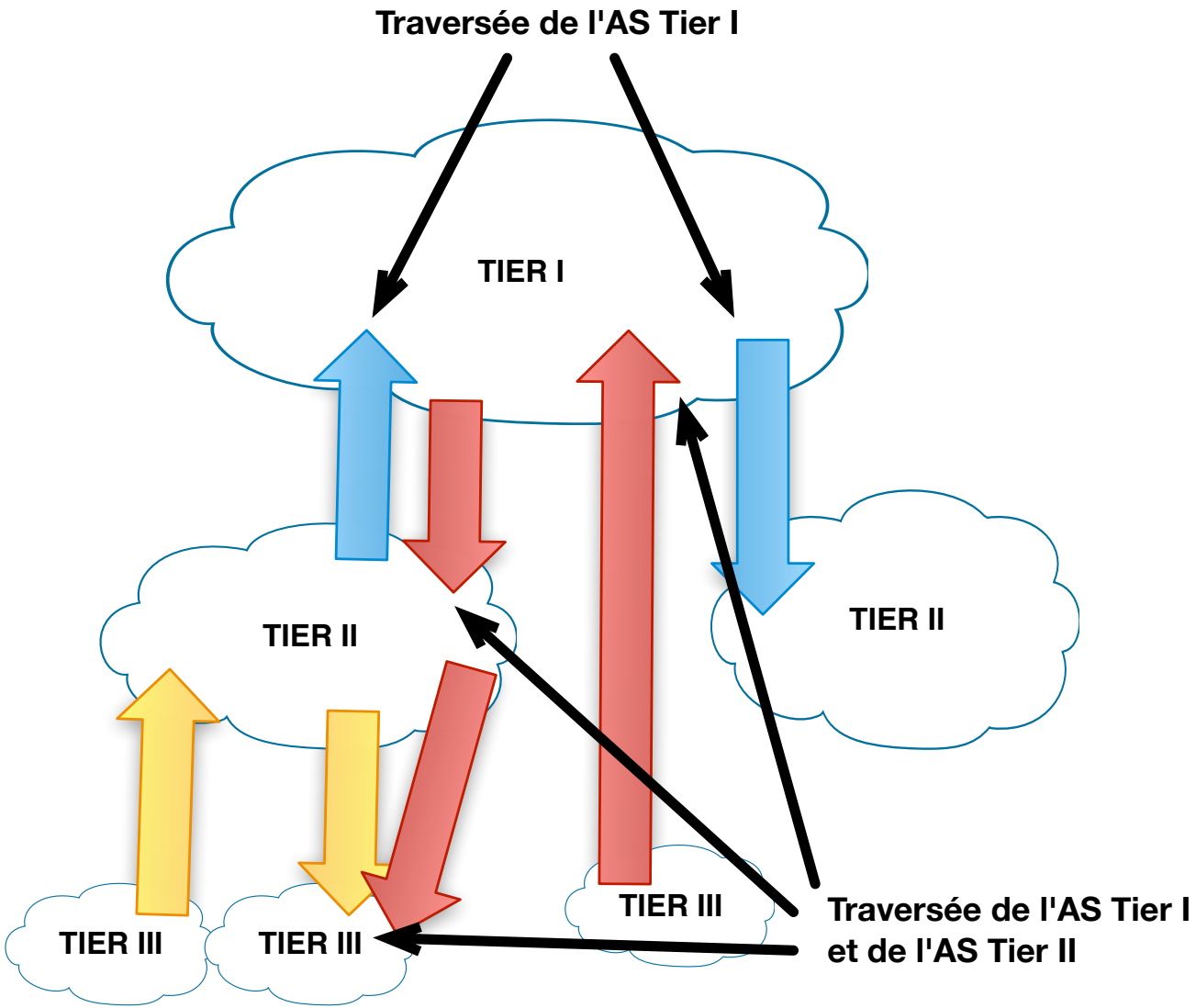
## Opérateurs Tier II

- Achètent du transit aux opérateurs Tier I
- Acheminent du trafic entre AS et opérateurs Tier III
- Exemples : British telecom, Vodafone, Tele2, Comcast...

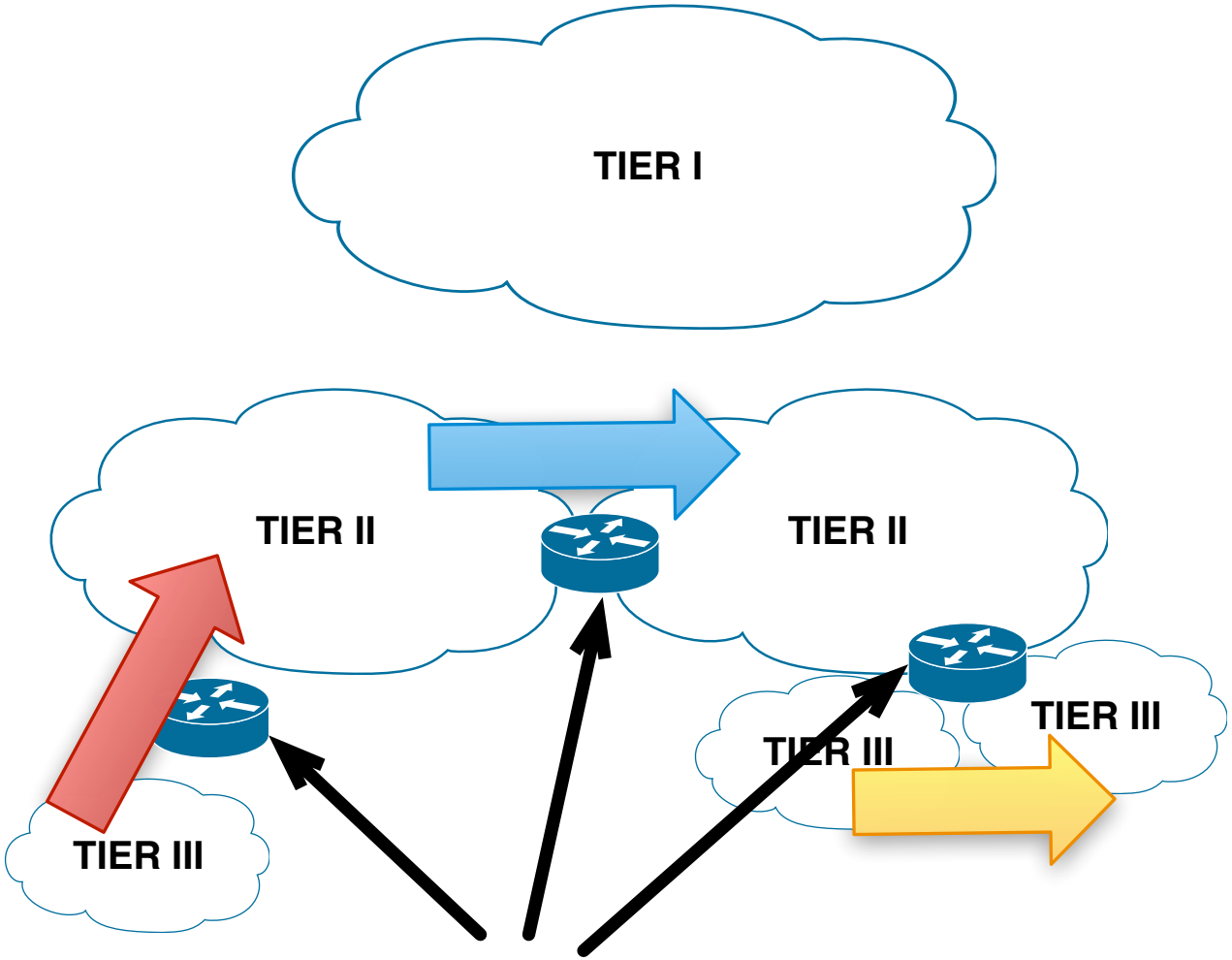
## Opérateurs Tier III

- Ne font qu'acheter du transit à des opérateurs Tier I et Tier II
- Beaucoup plus nombreux et plus petits

# Transit


Traversée de l'AS Tier I
TIER I
TIER II
TIER II
TIER III
TIER III
TIER III
Traversée de l'AS Tier I
et de l'AS Tier II

# Peering




Points de peering entre les AS

# Backbone d'Internet

Internet est un réseau **décentralisé** et **résilient**

- Pas de point central de défaillance
- Pas de point central de congestion
- Pas de gestion centralisée

# Backbone d'Internet

Internet est un réseau **décentralisé** et **résilient**

- Pas de point central de défaillance
- Pas de point central de congestion
- Pas de gestion centralisée

Infrastructure :

- Réseau physique redondant
- Fibres optiques agrégées
- Gestion de congestion, équilibrage de charge
- Le moins possible de traitements et d'opérations possible sur les paquets transitant sur ces fibres : tout est fait aux extrémités

# Backbone d'Internet

Internet est un réseau **décentralisé** et **résilient**

- Pas de point central de défaillance
- Pas de point central de congestion
- Pas de gestion centralisée

Infrastructure :

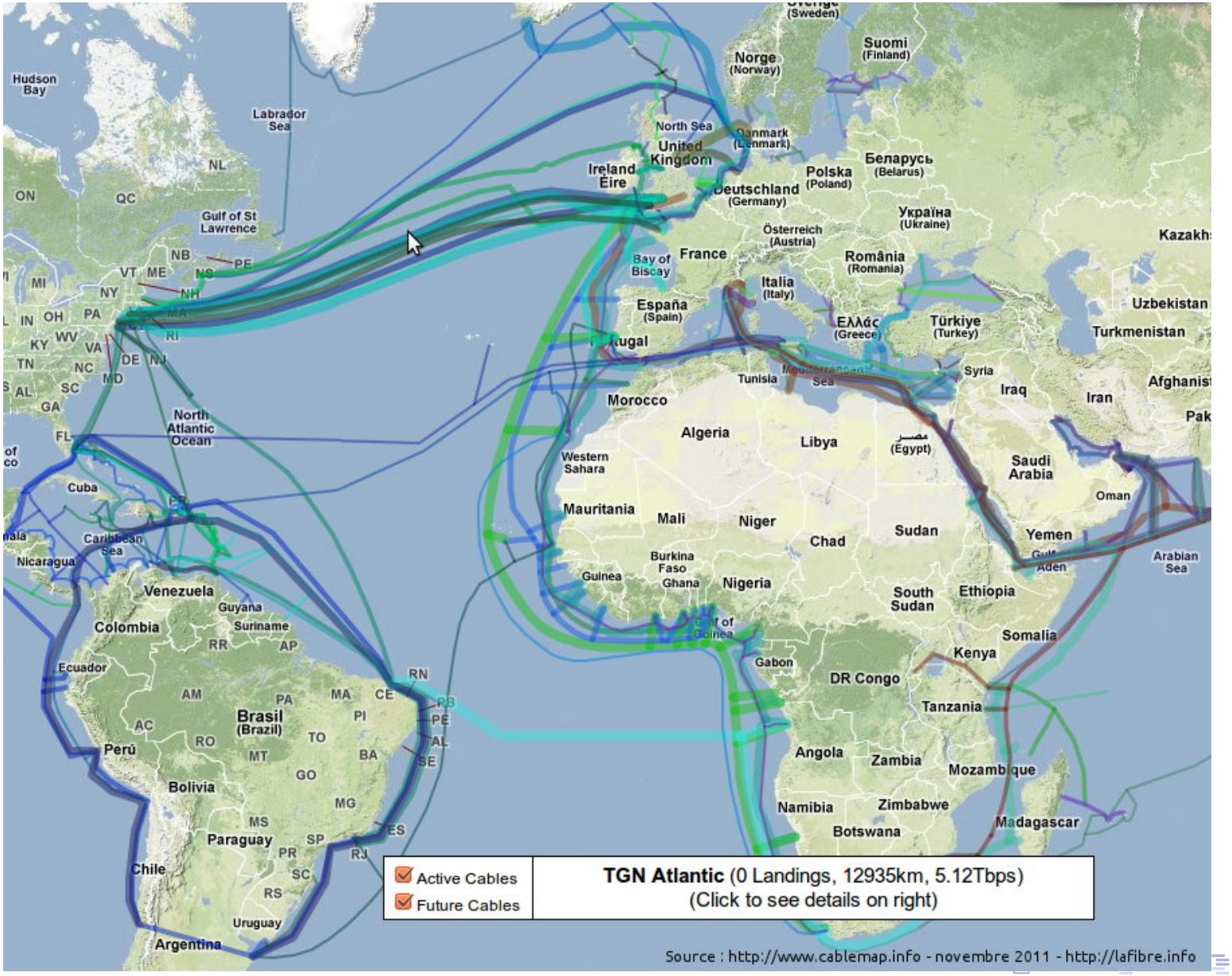
- Réseau physique redondant
- Fibres optiques agrégées
- Gestion de congestion, équilibrage de charge
- Le moins possible de traitements et d'opérations possible sur les paquets transitant sur ces fibres : tout est fait aux extrémités

Réseau **rapide** :

- 45 Mb/s en 1998 aux États-Unis
- Jusqu'à 31 Tb/s aujourd'hui

# Backbone d'Internet

Cartes en temps réel : `http://www.cablemap.info`



# Default-Free Zone

La **Default-Free Zone** (DFZ) est l'ensemble des AS d'Internet qui n'ont pas besoin d'avoir une *route par défaut* pour router leurs paquets.

- Les routeurs de la DFZ ont une **table BGP complète**
- ... à peu près : changements rapides, par exemple

Ils ont une **route explicite vers tous les réseaux d'Internet**

# Plan du cours

# Notion de préfixe

## Petit rappel sur IP

- Une adresse IP est constituée de deux parties :
  - la partie réseau
  - la partie locale
- Les bits les plus à gauche identifient le réseau. C'est le **préfixe**
- Les bits les plus à droite identifient le machine.

Par exemple : adresse IPv4 80.67.160.1/27

- Préfixe de longueur 27 bits
- La machine est désignée par les 5 derniers bits

# Affichage des préfixes connus

```
coti@thorim:~$ netstat -rn -finet
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            192.168.0.254      UGSc           20        0     en1
127                127.0.0.1          UCS             0        0     lo0
127.0.0.1          127.0.0.1          UH              2     1598     lo0
169.254            link#5             UCS             0        0     en1
192.168.0          link#5             UCS             2        0     en1
192.168.0.10       127.0.0.1          UHS             0        0     lo0
192.168.0.254      0:24:d4:c1:a3:f6   UHLWIir        21      297     en1   1190
192.168.0.255      ff:ff:ff:ff:ff:ff  UHLWbI          0       40     en1
```

→ On voit les **préfixes** connus

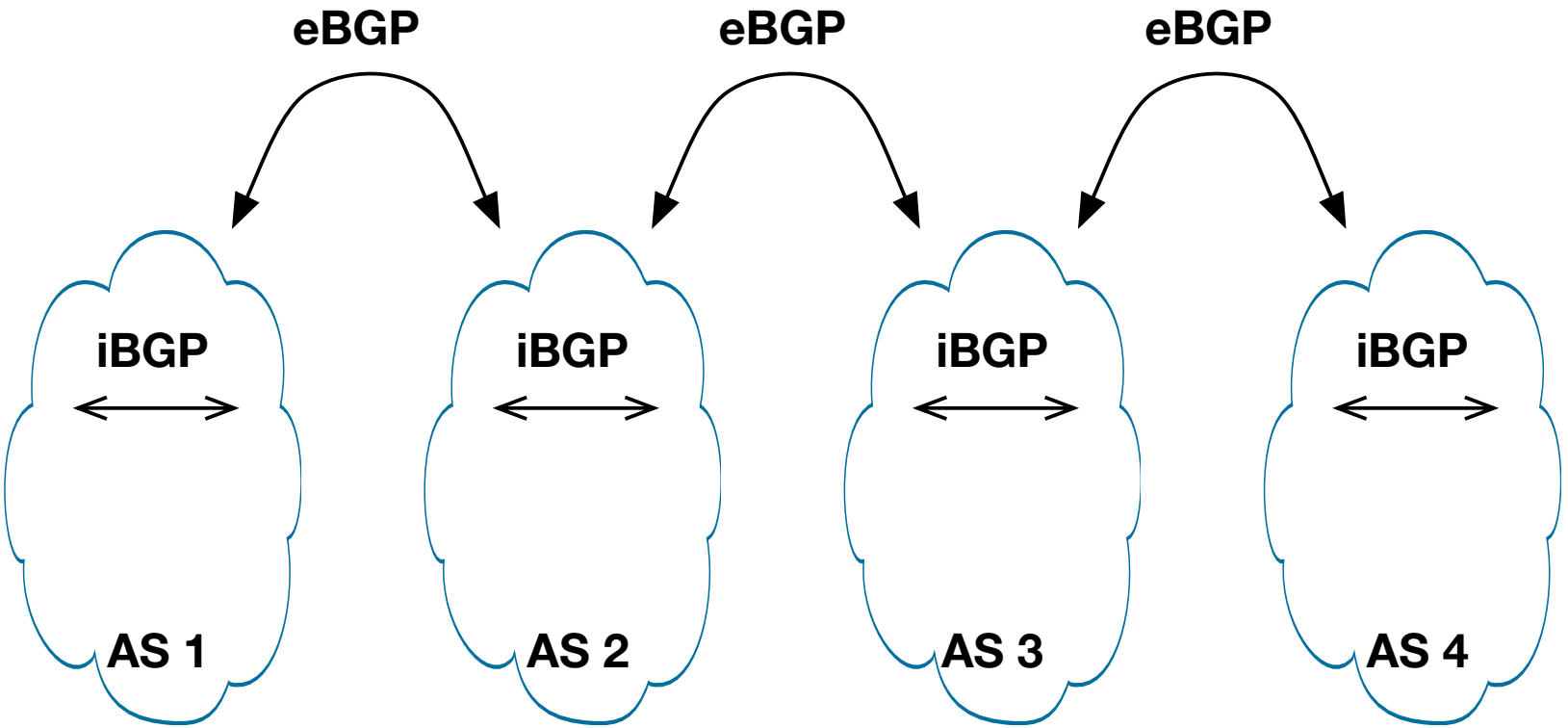
# Fonctionnement global de BGP

Deux parties :

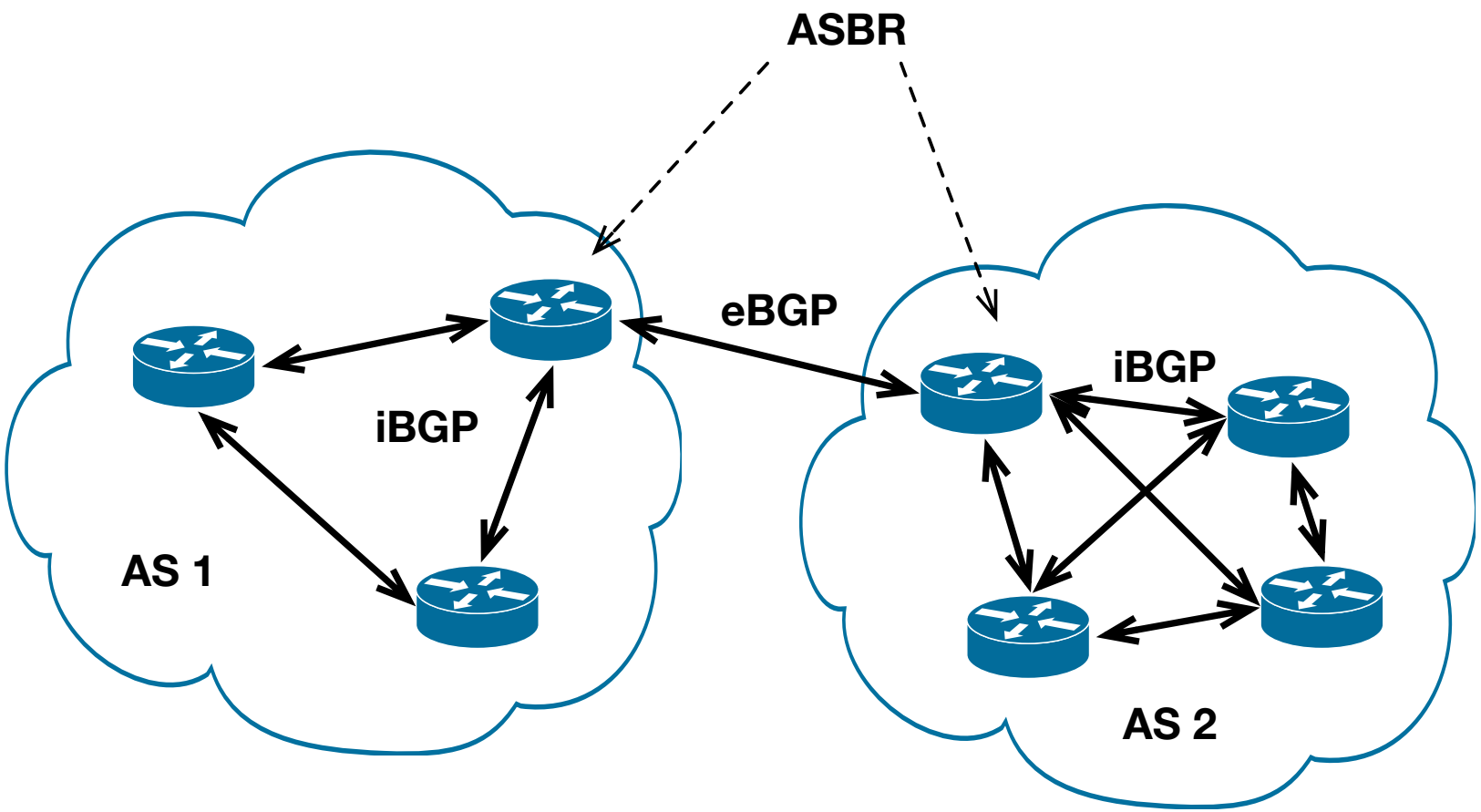
- **Internal BGP** (iBGP)
  - Au sein d'un AS
  - Deux voisins iBGP ne sont pas forcément physiquement connectés directement
  - Transporte des préfixes d'Internet et les préfixes internes à l'AS à travers l'AS
- **External BGP** (eBGP)
  - Entre les AS
  - Des voisins eBGP doivent être connectés directement
  - Échange de préfixes entre AS
  - Mise en place des politiques de routage entre AS
- eBGP ou iBGP ?
  - Si le numéro d'AS est le même pour deux routeurs, BGP sait qu'il fait de l'iBGP ; sinon, eBGP

- Apprend les routes possibles grâce aux annonces iBGP et eBGP
- Sélectionne les meilleures routes et les insère dans sa table de routage
- Les meilleurs routes sont envoyées à ses voisins en External BGP
- Application de politiques de routage en influant sur la sélection de la "meilleure route"

# External BGP / Internal BGP



# External BGP / Internal BGP




Les routeurs à la frontière avec d'autres AS sont des **ASBR** : Autonomous System Boundary Routers

# Hiérarchie du protocole

Le protocole est alors **hiérarchique** :

- Les passerelles des AS parlent entre eux
- Les informations sont transmises par les passerelles à l'intérieur de leur AS

Avantages :

- Réduction de la taille des tables de routage (factorisation)
- Réduction du nombre de connexions entre routeurs !

Inconvénients :

- Temps de transmission des mises à jour des routes
- Updates toutes les 90 secondes : si une route a été modifiée, on peut ne pas le savoir
- Certains routeurs peuvent être accessibles ou non par intermittence. Les temps de propagation des mises à jour continuelles rend le système incohérent : on parle de *route flapping*

# En-tête BGP

*Taille des champs en bits*

| *Marqueur (128)* | | |
|---|---|---|
| Longueur (16) | Type (8) | Bourrage (8) |

- Marqueur : généralement tous les bits mis à 1 (compatibilité)
- Longueur : longueur totale du message (incluant l'en-tête) en octets
- Type : type de message BGP
  - 1 : open
  - 2 : update
  - 3 : notification
  - 4 : keepalive
  - 5 : route-refresh

# Messages open

Type de message **1** .

Utilisés à l'ouverture d'une session BGP (*i.e.*, lorsqu'un routeur est connecté) pour s'annoncer Format d'un message d'erreur :
*Taille des champs en bits*

| Version (8) | Num. d'AS (16) | HDT (16) |
|---|---|---|
| ID BGP | Lg options (8) | |
| Paramètres optionnels (lg variable) | | |

- HDT : Hold Down Timer
- Lg options : longueur des paramètres optionnels
- ID BGP : identifiant BGP du message, donnant l'identifiant (sur 32 bits) du routeur (= adresse IP)

# Messages update

Type de message **2** .

Contiennent les routes elles-mêmes :

- Au démarrage : toutes les routes jusqu'à ce que toute la table de routage soit passée
- En fonctionnement : modifications du réseau (nouvelles routes et routes inaccessibles)

Format : champs de longueurs variables

- Routes inaccessibles
  - Longueur (16 bits)
  - Routes inaccessibles
- Attributs de chemin
  - Longueur (16 bits)
  - Chemin
- Informations d'accessibilité réseau
  - Longueur (8 bits)
  - Préfixe, en notation 204.129.10/24
  - ...

# Messages notification

Type de message **3** .

Servent à prévenir que quelque chose s'est mal passé

- Une option non supportée a été envoyée dans un message open
- Un routeur n'a pas envoyé de keepalive ni d'update

Format d'un message d'erreur :
*Taille des champs en bits*

| Code d'erreur (8) | Sous-code (8) | *Données (16)* |
|---|---|---|
| *Données...* | | |

Codes d'erreur :

- 1 : Erreur dans un en-tête
- 2 : Erreur dans un message OPEN
- 3 : Erreur dans un message d'update
- 4 : Timer Hold Down expiré
- 5 : Erreur d'un état
- 6 : fin

# Messages keepalive

Type de message **4** .

- Envoyés à intervalles réguliers
- Maintiennent la session ouverte en l'absence de modifications
- Le timer FIT (Keepalive Interval Timer) spécifie cet intervale dépend de la configuration de chaque routeur (par défaut 60 secondes)

Si un routeur BGP rate trois messages keepalive (par défaut, 180 secondes de silence)

- Il est suspecté d'être HS ou inatteignable
- On attend l'écoulement du timer HDT (Hold Down Timer) pour supprimer toutes les routes en provenance de ce routeur

# Algorithme de sélection de route

BGP est un protocole à **vecteur de chemins**

- Variante de l'algorithme de **Bellman-Ford**
- Distance = nombre de sauts **entre AS** (pas entre routeurs)
- Attributs appliqués aux préfixes pour déterminer la meilleure route

# Algorithme de sélection de route

BGP est un protocole à **vecteur de chemins**

- Variante de l'algorithme de **Bellman-Ford**
- Distance = nombre de sauts **entre AS** (pas entre routeurs)
- Attributs appliqués aux préfixes pour déterminer la meilleure route

Supporte le routage inter-domaine sans distinction de classe :

- C'est la **notion de préfixe** qui prévaut

# Algorithme de sélection de route

BGP est un protocole à **vecteur de chemins**

- Variante de l'algorithme de **Bellman-Ford**
- Distance = nombre de sauts **entre AS** (pas entre routeurs)
- Attributs appliqués aux préfixes pour déterminer la meilleure route

Supporte le routage inter-domaine sans distinction de classe :

- C'est la **notion de préfixe** qui prévaut

Notion de **préférence locale**

- C'est le paramètre local qui permet d'influer sur les **choix des routes**
  - → Par exemple, on préfèrera de passer par un lien de peering (gratuit) qu'un lien de transit (payant)
- Les **attributs** permettent de sélectionner une route parmi plusieurs selon cette préférence locale

# Échange de routes

Important : un AS doit être **présent dans la DFZ**

- La DFZ connaît tout, donc l'AS doit être dedans

Pas de diffusion globale !

- Échange avec ses voisins (directs), deux à deux
- À intervalle régulier

Les routeurs s'échangent des **informations d'accessibilité**

- Liste des réseaux accessibles via chaque routeur voisin (routeurs pairs = peer routers)

Un routeur annonce à ses voisins les **routes qu'il veut qu'ils connaissent**

# Internal BGP

Utilisé **à l'intérieur d'un AS**

- En charge de la connectivité entre les routeurs BGP internes
- Assure la cohérence entre les routeurs BGP

Le réseau interne doit être **connexe**

- Les connexions entre les routeurs iBGP doivent former un **graphe complet** (maillage complet)
  - → Tous les routeurs sont connectés directement deux à deux
- Physiquement, pas forcément connectés les uns aux autres
  - → Indépendant de la topologie physique

Utile si on a un AS grand ou étendu géographiquement

- Répartition de charge
- Lien avec plusieurs autres AS

# External BGP

Revenons à l'information obtenue sur un AS donné :

```
coti@thorim:~$  whois -h whois.ripe.net AS1303
[...]
% Information related to 'AS1299 - AS1309'
import:         from AS2200 action pref=100; accept ANY
export:         to AS2200 announce AS1303
default:        to AS2200 action pref=10; networks ANY
[...]
```

On voit ici les **informations sur la politique de routage** de l'AS :

- Échange des routes avec l'AS 2200 (RENATER)
- Informations sur la politique de sélection de routes

# External BGP

Revenons à l'information obtenue sur un AS donné :

```
coti@thorim:~$  whois -h whois.ripe.net AS1303
[...]
% Information related to 'AS1299 - AS1309'
import:         from AS2200 action pref=100; accept ANY
export:         to AS2200 announce AS1303
default:        to AS2200 action pref=10; networks ANY
[...]
```

On voit ici les **informations sur la politique de routage** de l'AS :

- Échange des routes avec l'AS 2200 (RENATER)
- Informations sur la politique de sélection de routes

Même chose avec l'AS 2200 :

```
import:         from AS20965 action pref=300; accept ANY
import:         from AS7500  action pref=190; accept AS7500
import:         from AS1273  action pref=300; accept ANY
import:         from AS3257  action pref=300; accept ANY
export:         to AS-RENATER announce ANY
export:         to AS-SFINX-MEMBERS  announce AS-RENATER
export:         to AS20965   announce AS-RENATER  AS7500
export:         to AS12654   announce AS-RENATER
export:         to AS21357   announce AS-RENATER
export:         to AS1273    announce AS-RENATER
export:         to AS3257    announce AS-RENATER
```

# External BGP

Quand un routeur BGP est connecté à un autre routeur BGP dans un autre AS :

- Il se connecte à cet autre routeur (rappel : on utilise TCP)
- Les routeurs s' **identifient** l'un avec l'autre (possibilité de rejeter)
- Chaque routeur **annonce les routes** qu'il veut que l'autre connaisse

# External BGP

Quand un routeur BGP est connecté à un autre routeur BGP dans un autre AS :

- Il se connecte à cet autre routeur (rappel : on utilise TCP)
- Les routeurs s' **identifient** l'un avec l'autre (possibilité de rejeter)
- Chaque routeur **annonce les routes** qu'il veut que l'autre connaisse

En cours de fonctionnement :

- Chaque routeur annonce à ses voisins les **changements** ayant eu lieu dans sa table
- Si pas de modifications toutes les X secondes : message **keepalive** pour garder la route active (généralement toutes les 30 secondes)

# Informations de BGP

## Informations reçues par un routeur

- Réception des informations de routeurs des voisins
- Sélection de chemins, placement dans la table des chemins de BGP
- Détermination du "meilleur chemin"

## Informations envoyées par le routeur

- Chaque routeur annonce à ses voisins ses meilleurs chemins vers les préfixes qu'il connaît

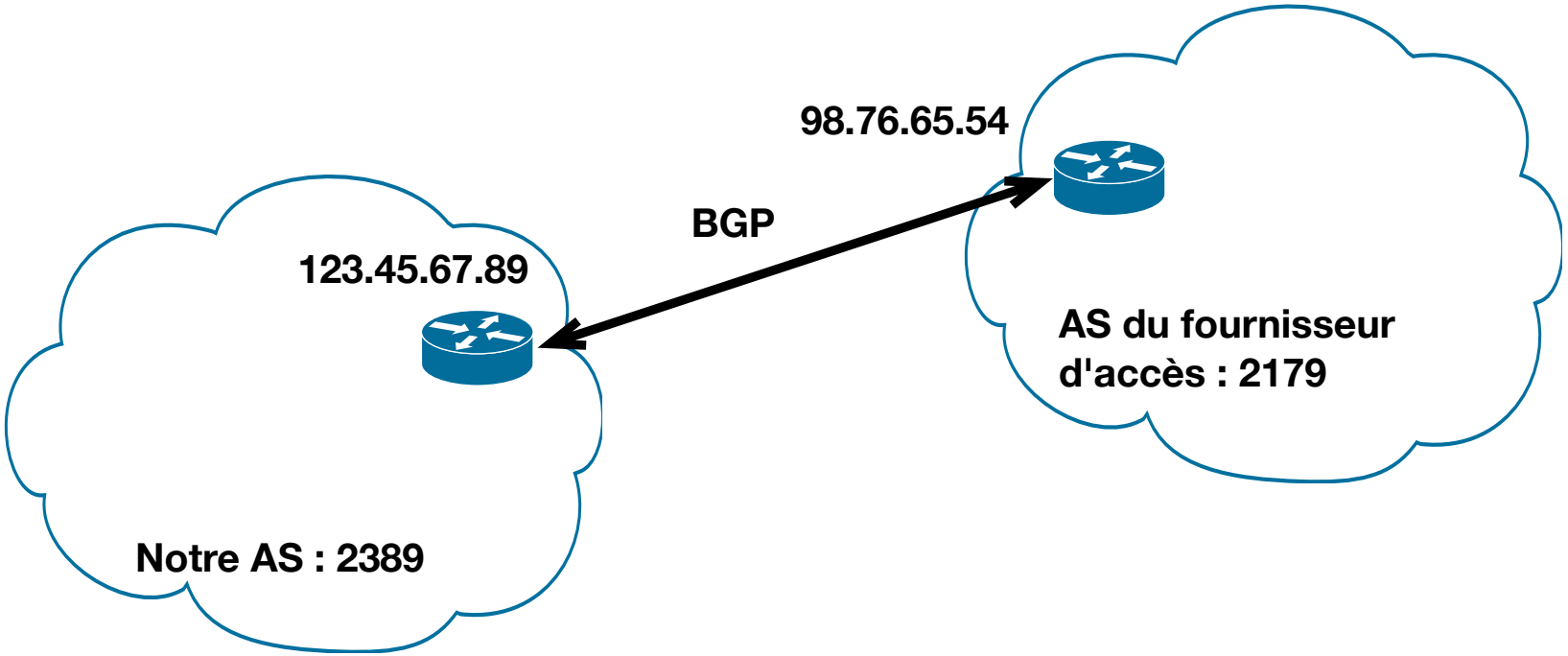
Les meilleurs chemins sont stockés dans la **table de routage** (RIB)

- Les meilleurs chemins de la RIB sont stockés dans la **table de transmission** (FIB)
  - Dans la FIB : unicité des préfixes/longueurs de préfixes
  - Plus petite distance protocolaire

# Raccordement d'un AS sur Internet

Avec **un seul routeur ASBR** (une seule connexion avec un autre AS) :

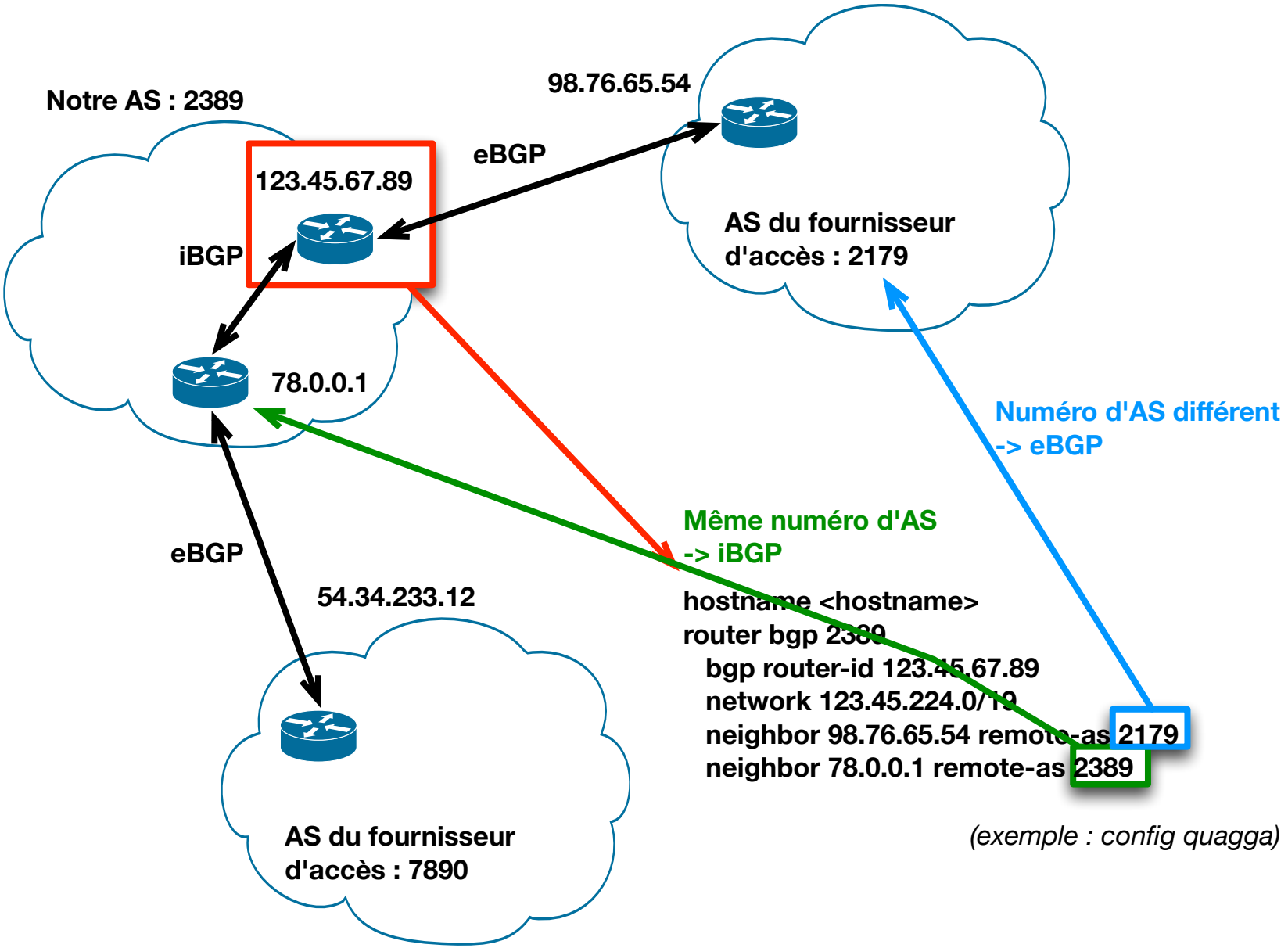
- Notre ASBR connaît ses voisins, accessibles directement, sans routage
- Il leur déclare
  - Son numéro d'AS
  - Son id de routeur (adresse IP du routeur)
  - Les réseaux que l'on annonce (notation CIDR)




```
hostname <hostname>
router bgp 2389
  bgp router-id 123.45.67.89
  network 123.45.224.0/19
  neighbor 98.76.65.54 remote-as 2179
```

*(exemple : config quagga)*

# Raccordement d'un AS sur Internet




*(exemple : config quagga)*

# Détection de boucles

Mécanisme de **détection de boucles**

- Les mises à jour contiennent le **chemin** par lequel le paquet de mise à jour a transité (numéros d'AS)
- On ne transfère pas à un AS dont le numéro est déjà présent dans le chemin

Attention : fonctionne uniquement entre AS, pas à l'intérieur d'un AS

- Pour éviter les problèmes en interne : les routeurs BGP au sein d'un AS sont **tous connectés** les uns aux autres (= topologie de graphe complet)

# IP hikacking

Principe : annoncer à tous les routeurs une route comme étant la meilleure pour une destination

- Évidemment, vers une de ses machines
- Tout le trafic vers cette destination est alors dirigé vers cette machine
- Possibilité de faire ce que l'on veut de ce trafit

Exemple : incident YouTube de 2007

- Pakistan Telecom s'est annoncé comme étant la meilleure route vers YouTube
- Ensuite, trou noir : le trafic était perdu

→ Conséquence : YouTube inaccessible !

# Pour aller plus loin

Un petit peu de lecture sur BGP :

- `http://www.bortzmeyer.org/files/bgp.pdf`
- `http://training.apnic.net/docs/eROU03_BGP_Basics.pdf`

# Plan du cours

# Autres protocoles de routage dynamique

Intermediate System to Intermediate System (IS-IS)

- À états de liens
- Définit des systèmes terminaux (les machines des utilisateurs), des systèmes intermédiaires (les routeurs), des zones (groupes locaux) et des domaines (regroupement de plusieurs zones)
- Hiérarchique
- Utilise son propre système d'adressage

Interior Gateway Routing Protocol (IGRP)

- Protocole propriétaire CISCO
- Supporte plusieurs métriques pour chaque route : bande passante, latence, charge... Combinaison de plusieurs critères par une somme pondérée
- Créé à l'origine pour outrepasser les limites de RIP (15 sauts)

Enhanced Interior Gateway Routing Protocol (EIGRP)

- Protocole propriétaire CISCO
- Utilise l'algorithme DUAL (Diffusing Update ALgorithm)
- Évolution de IGRP, compatible
- Algorithme hybride vecteur de distance / état de liens
- Consomme peu de bande passante : pas de mises à jour régulières, pas de diffusions, seules les modifications sont communiquées

# Plan du cours

# Plan du cours

# Pourquoi IPv6 ?

- Saturation des adresses IPv4
  - l'IANA a donné ses 5 derniers blocs libres le 3/02/2011 aux RIR
  - les RIR (Registres Internet Régionaux) distribuent aux RIL ;
  - les RIL (Registres Internet Locaux, 355 en France en 2011) manquent d'adresses IPv4 depuis 2012.

- Plus d'adresses
  - IPv4 : 4 octets : $2^{32} \approx 4.10^9$ adresses.
  - IPv6 : 16 octets : $2^{128} \approx 3,4.10^{38}$ adresses
    - $\approx 10^{30}$ /personne sur terre.
    - Surface de la terre = 510 067 420 $km^2$ $\rightarrow$ $6.67 * 10^{29}$ adresses/$km^2$ donc $6.67 * 10^{26}$ adresses/$m^2$

# Avancées de IPv6

- mécanismes de configuration et de renumérotation automatique
- IPsec, QoS et le multicast (présents en IPv4 par ajouts ultérieurs)
- simplification des en-têtes de paquets IP

« IPv6 » existe depuis 1995 (RFC 1883), finalisée en 1998 (RFC 2460).
Ex.: Google accessible en IPv6 depuis 2008.

# Perspectives ouvertes par IPv6

IPv6 ouvre de nouvelles perspectives de communications réseau et d'applications :

## Internet des objets

- Réseaux de capteurs communicants
- Domotique / Home appliances : consultation et contrôle à distance d'appareils domestiques
- Gestion et entretien de bâtiments : gestion de la consommation énergétique
- Automatisation Industrielle : M2M, Monitoring automatique, reporting intelligent
- Gestion de la ville : Gestion des parkings, de la circulation, de la pollution

# Perspectives ouvertes par IPv6

IPv6 ouvre de nouvelles perspectives de communications réseau et d'applications :

## Réseaux Persistants et Mobilité

- Persistance de la connectivité Internet (adresse IP unique permanente)
- Solutions de ToIP et VoIP persistantes / itinérantes
- Ubiquité du contexte IT utilisateur : environnement informatique utilisateur disponible tout le temps, n'importe où sur divers supports

# Perspectives ouvertes par IPv6

IPv6 ouvre de nouvelles perspectives de communications réseau et d'applications :

## Diffusion multi-canal de flux

- Multicasting de flux audio / vidéo : conférences, manifestations sportives
- Solutions d'e-Learning / e-Education : classes virtuelles, diffusion libre du savoir
- Nouvelles solutions de services P2P : échanges de don, jeux en ligne (type Second Life)

# Perspectives ouvertes par IPv6

IPv6 ouvre de nouvelles perspectives de communications réseau et d'applications :

## Monétique Sécurité

- M-Payment : Solution de paiement NFC (Near-Field communications)
- Sécurisations des échanges end-to-end : IPSec natif sur les échanges de flux
- M-Banking : Extensions des fonctionnalités de M-Banking

# Comparatif IPv4 vs IPv6

| Version | IPv4 | IPv6 |
|---|---|---|
| Déployé en | 1981 | 1999 |
| Taille des adresses | 32 bits | 128 bits |
| Format de notation | Décimale à point | Hexadécimale |
| Nombre d'adresses | $2^{32}$ =4 294 967 296 | $2^{128}$ = 340 282 366 920 938 463 463 374 607 431 768 211 456 |
| Exemples de préfixes | `192.0.2.0/24` | `2001:0DB8:0234::/48` |

# Plan du cours

# Adressage IPv6

Normalisée: en hexa, 8 groupes de 2o séparés par des « : », pas d'espace.

**Exemple**

`fe80:0000:0000:0000:0004:06ff:fed0:0629`

Peut être simplifiée en (suppression des 0 non significatifs et `::` utilisé au plus une fois, pour une suite de $2n$ octets à 0 contigus):

**Exemples**

`fe80::4:6ff:fed0:629` `::1` `ff02::1`

Notation pour NAT64 (adresse IPV6 intégrant une adresse IPv4) :

**Exemples**

`64:FF9B::192.168.0.1` = `64:FF9B::C0A8:0001`

# Adressage IPv6
## Notion de préfixe

Notion de « netmask » devient longueur de préfixe (voir CIDR):

**Exemple**

`fe80:db8::/32` ici le préfixe est de 32 bits donc représente les adresses commençant par `fe80:db8`

A la fois l'adresse et le préfixe:

**Exemple**

`fe80:db8::1/32` signifie la machine `fe80:db8::1` sur le réseau `fe80:db8::/32`

# Types d'adresses IPv6

| Préfixe | Description | Subdivision | Description |
|---|---|---|---|
| ::/0 | Route par défaut (en IPv4 0.0.0.0 netmask 0.0.0.0) | | |
| ::/8 | Adresses réservées | ::/128<br>::1/128<br>64:FF9B::... | 0.0.0.0 en IPv4<br>ip6-localhost<br>adresses NAT64 |
| 2000::/3 | Adresses routables sur Internet | 2001::/16 | adresses permanentes ouvertes à réservation depuis 1999 |
| | | 2001:db8::/32 | adresses pour de la doc! |
| | | 2002::/16 | adresses 6to4: trafic IPv6 via réseaux IPv4 |
| fc00::/7 | Adresses locales (à un site) uniques grâce à 40 bits aléa. qui suivent fc (obsolète?) | | |
| fe80::/10 | Adresses locales (à un lien) auto-attribuées, nécessaires | | |
| ff00::/8 | Adresses multicast | ff02::1:ff00:0/104<br>ff02::1 | Solicited-Node adresses multicast<br>all-nodes : recherche d'hôtes |

# Types d'adresses IPv6

Structure des adresses unicast globales:

| *champ :* | **préfixe** | **sous-réseau** | **interface** |
|---|---|---|---|
| *nbre bits :* | 48 | 16 | 64 |

**Notes:**

Le FAI free propose des adresses IPv6 avec des préfixes de 64 bits, cela fait tout de même $2^{64} \approx 1,8.10^{19}$ adresses par abonné!

⚠ pas d'adresse de diffusion en IPv6.

# Notion de "scope"

**Scope** (portée) = Domaine de validité et d'unicité.

- Adresses **unicast** : *de 1 vers 1*
  - l'adresse boucle locale ::1/128 a une validité limitée à l'hôte,
  - les adresses locales à un lien, uniques sur un lien donné (VLAN, sous-réseau),
  - locales à un site: voir adresses privées IPv4.
  - les autres adresses ont un scope global, elles sont uniques et peuvent être utilisées pour communiquer avec d'autres adresses globalement uniques, ou des adresses locales à des liens directement connectés
- Les adresses **anycast** (*de 1 vers le plus proche*)
  - idem unicast
- Les adresses **multicast** `ff00::/8`

⟶ Une interface réseau possède plusieurs adresses IPv6.

# Adresses Multicast

| *8bits* | *4bits* | *4bits* | *112 bits* |
|---|---|---|---|
| `ff` | flags | scope | groupid |

- flags: 0RPT (R=0: n'embarque pas d'adresse de point de rdv, T=0: permanently-assigned ("well-known"), P=1 adresse basée sur le préfixe)
- scope : 1= local à l'hôte, 2= local au lien, 5= local au site, E=global.

**Exemples**

`ff02::1` recherche de noeuds (groupid=1) en IPv6 sur le "brin" local
`ff05::2` recherche de routeurs (groupid=2) IPv6 sur le site local.

# Attribution des adresses IPv6

- Manuelle
- Configuration automatique
  - Attribution par un serveur DHCPv6 (*stateful*)
  - Autoconfiguration (*stateless*) basée sur l'adresse MAC (et éventuellement *ND: Neighbor Discovery*)
    puis 8 octets formés comme suit : +2 au premier octet (bit U/L à 1) et insérer `fffe` pour les octets 4 et 5.

**Exemples**

`00:04:06:08:0A:0C` → `fe80::0204:06ff:fe08:0a0c`
`00:04:06:08:0A:0C` *et par ex. préfixe* `3ffe:302::/64` *(fourni par ND)*
→ `3ffe:302::0204:06ff:fe08:0a0c`

# Résolution d'adresse IPv6

- *Resource Record* de résolution est `AAAA` au lieu de `A`.
- Résolution inverse: suffixe=`ip6.arpa.` (au lieu de `in-addr.arpa.`) avec des points séparant chaque quartet.

**Exemple**

b.8.6.0.0.0.1.c.0.0.0.0.0.0.0.0.2.2.0.0.0.4.2.0.0.1.6.0.1.0.0.2.ip6.arpa. PTR www.ipv6.ripe.net.

- Configuration de Bind9 pour qu'il réponde à des requêtes IPv6 : dans les *options* : `listen-on-v6 { any; }`

# Neighbor Discovery Protocol (ND)

- basé sur ICMPv6
- associe les adresses IPv6 à des adresses MAC sur un « brin » (voir ARP pour IPv4)
- découvre les routeurs et les préfixes routés (possible en IGMPv4)
- découvre le MTU (voir `tracepath` en IPv4)
- détecte les adresses dupliquées (voir `arping` pour IPv4)
- détecte hôtes devenus inaccessibles
- autoconfigure les adresses d'interface (voir adresses "link-local" pour IPv4)
- fournit éventuellement passerelle par défaut
- recherche éventuellement des serveurs DNS récursifs
- pas de diffusion mais du multicast au niveau Ethernet avec des adresses MAC débutant par `33:33:`

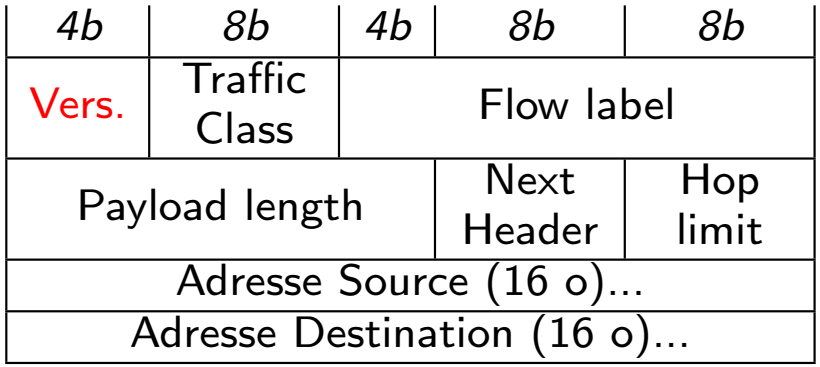
# Plan du cours

# Entête IPv6

| *4b* | *8b* | *4b* | *8b* | *8b* |
|---|---|---|---|---|
| Vers. | Traffic Class | Flow label | | |
| Payload length | | Next Header | Hop limit | |
| Adresse Source (16 o)... | | | | |
| Adresse Destination (16 o)... | | | | |

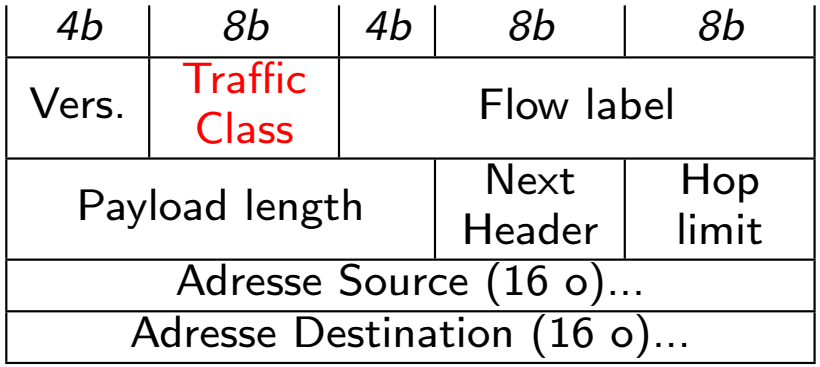
- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6




- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6




- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6

| *4b* | *8b* | *4b* | *8b* | *8b* |
|---|---|---|---|---|
| Vers. | Traffic Class | Flow label | | |
| Payload length | | Next Header | Hop limit | |
| Adresse Source (16 o)... | | | | |
| Adresse Destination (16 o)... | | | | |

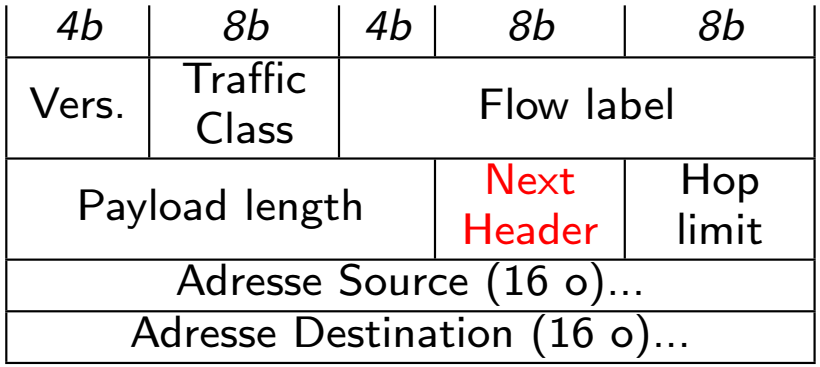
- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6

| *4b* | *8b* | *4b* | *8b* | *8b* |
|---|---|---|---|---|
| Vers. | Traffic Class | Flow label | | |
| Payload length | | | Next Header | Hop limit |
| Adresse Source (16 o)... | | | | |
| Adresse Destination (16 o)... | | | | |

- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

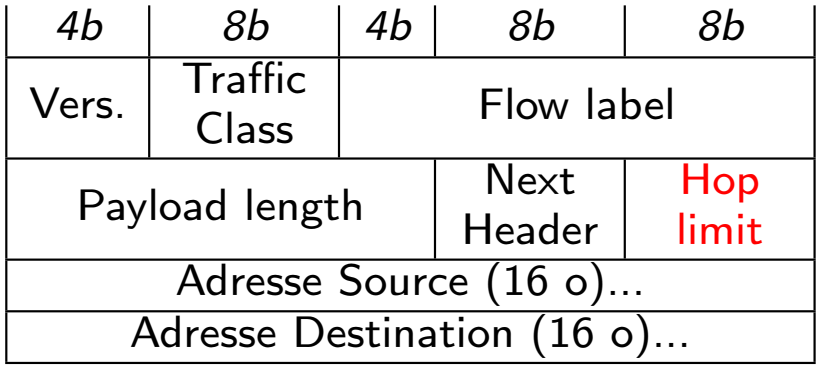
# Entête IPv6




- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets

- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6




- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6

| *4b* | *8b* | *4b* | *8b* | *8b* |
|---|---|---|---|---|
| Vers. | Traffic Class | Flow label | | |
| Payload length | | | Next Header | Hop limit |
| Adresse Source (16 o)... | | | | |
| Adresse Destination (16 o)... | | | | |

- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

# Entête IPv6

| *4b* | *8b* | *4b* | *8b* | *8b* |
|---|---|---|---|---|
| Vers. | Traffic Class | Flow label | | |
| Payload length | | Next Header | Hop limit | |
| Adresse Source (16 o)... | | | | |
| Adresse Destination (16 o)... | | | | |

- Version: 6
- Traffic Class: gestion QoS
- Flow Label: marquage de flux pour traitement différencié
- Payload length: taille données (charge utile) en octets
- Next Header: comme no proto. dans IPv4 ou chaînage d'options.
- Hop Limit: voir TTL de IPv4.

**Remarques**

⚠ Pas de checksum, pas de longueur d'entête.

⚠ Fragmentation par routeurs intermédiaires interdite: ICMPv6 "Packet Too Big".

⚠ Il existe une option Jumbo permettant de passer de $2^{16}$ à 4Go la taille max des données.

# Quelques codes de "Next Header" (extensions)

| Nom | Type | Taille | Description |
|---|---|---|---|
| Options Hop-By-Hop | 0 | var. | Options à propager à tous les routeurs |
| TCP | 6 | var. | ... |
| Routage | 43 | var. | Voir routage stricte et lâche IPv4 |
| Fragment | 44 | 64 bits | Fragmentation |
| AH | 51 | var. | Authentification en-tête, voir IPsec |
| ESP | 50 | variable | Chiffrement du contenu, voir IPsec |
| Options de dest. | 60 | var. | Options à traiter à destination |
| No Next Header | 59 | vide | Pas de paquet d'extension à suivre et pas de données |

# Chaînage d'entêtes

Exemple d'enchaînement d'entêtes:

| **Entête IPv6** | **Entête Routage** | **Entête ESP** | **Entête TCP** |
|---|---|---|---|
| next header=Routage | next header=ESP | next header=TCP | data TCP |

# Plan du cours

# Transition IPv4 → IPv6

## Schémas de cohabitation

⚠ IPv4 et IPv6 sont **incompatibles** .

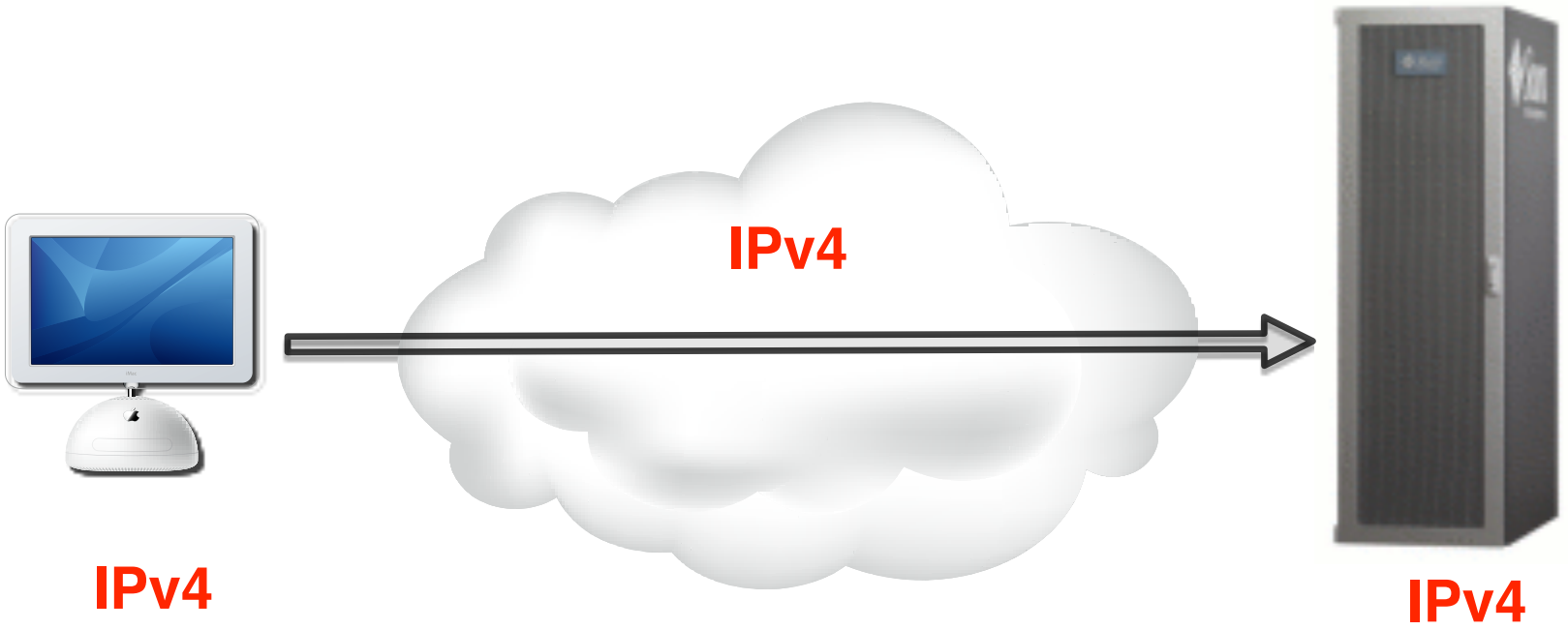
→ En attendant: **mécanismes de cohabitation d'IPv6 avec IPv4**

| Schéma | Description du schéma | Complexité |
|---|---|---|
| Schéma 1 | Un système IPv4 se connecte à un système IPv4 à travers un réseau IPv4 | Faible |
| Schéma 2 | Un système IPv6 se connecte à un système IPv6 à travers un réseau IPv6 | Faible |
| Schéma 3 | Un système IPv4 se connecte à un système IPv4 à travers un réseau IPv6 | Moyenne |
| Schéma 4 | Un système IPv6 se connecte à un système IPv6 à travers un réseau IPv4 | Moyenne |
| Schéma 5 | Un système IPv4 se connecte à un système IPv6 | Forte |
| Schéma 6 | Un système IPv6 se connecte à un système IPv4 | Forte |

# Transition IPv4 → IPv6

## Schémas de cohabitation

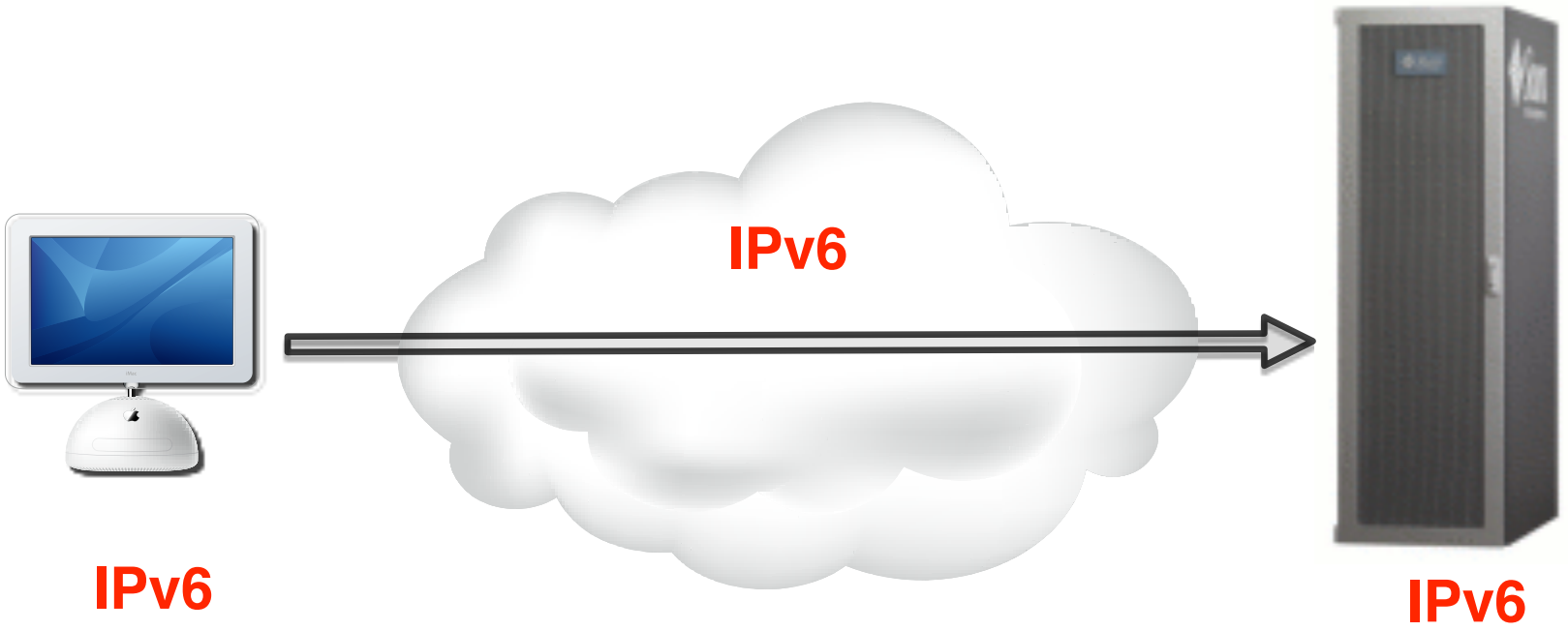
Schéma 1 : **Un système IPv4 se connecte à un système IPv4 à travers un réseau IPv4**



# Transition IPv4 → IPv6

## Schémas de cohabitation

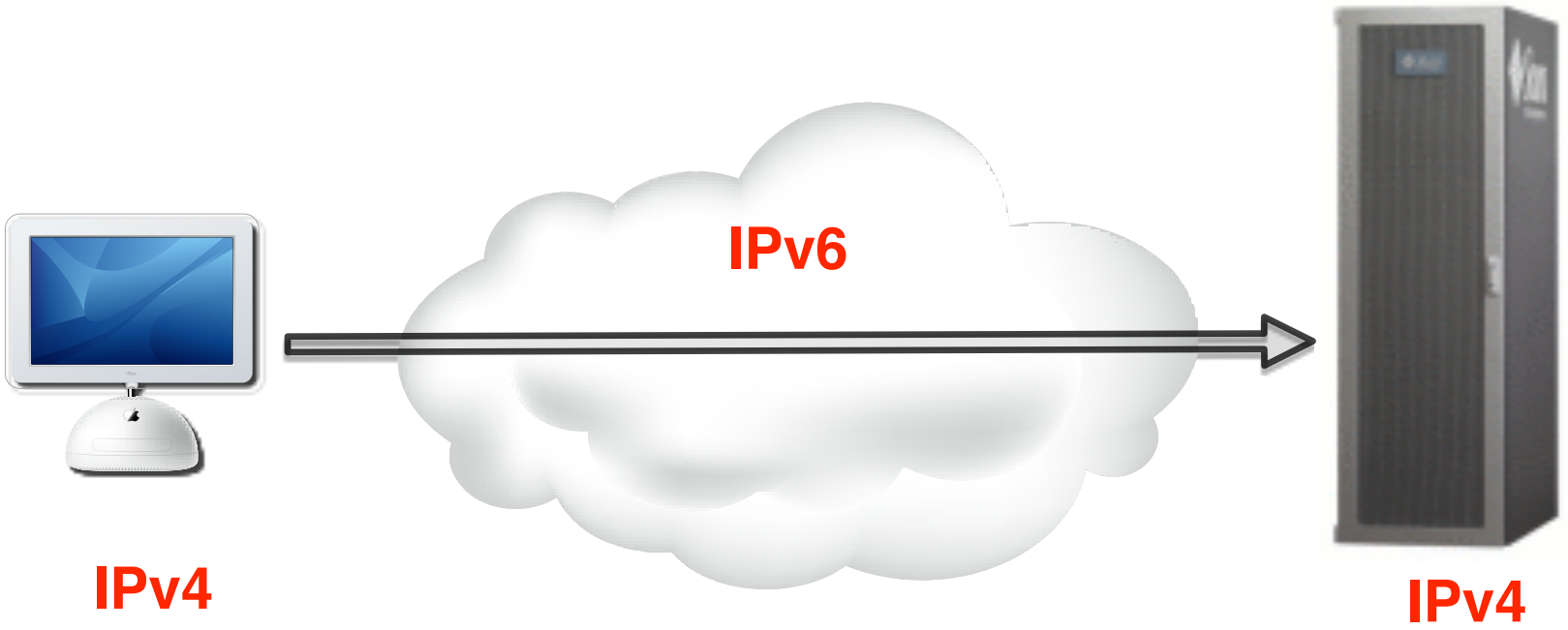
Schéma 2 : **Un système IPv6 se connecte à un système IPv6 à travers un réseau IPv6**



# Transition IPv4 → IPv6

## Schémas de cohabitation

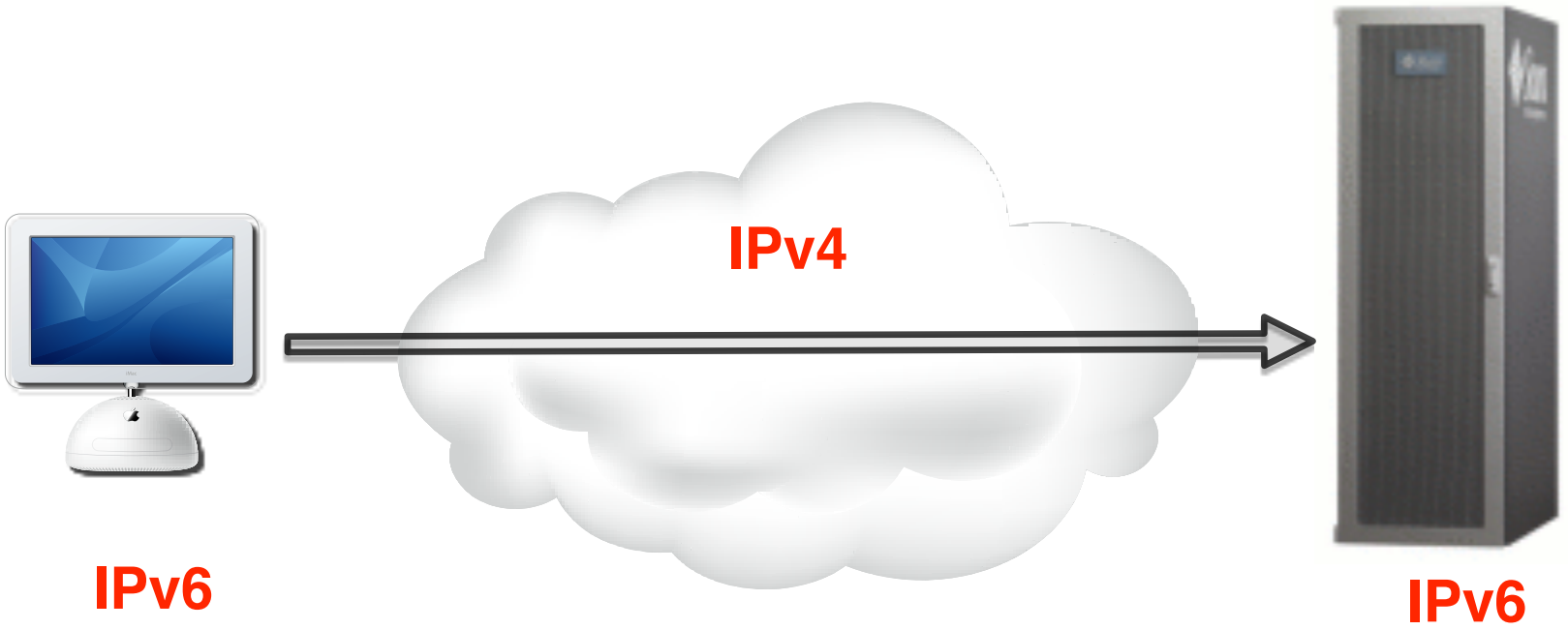
Schéma 3 : **Un système IPv4 se connecte à un système IPv4 à travers un réseau IPv6**



# Transition IPv4 → IPv6

## Schémas de cohabitation

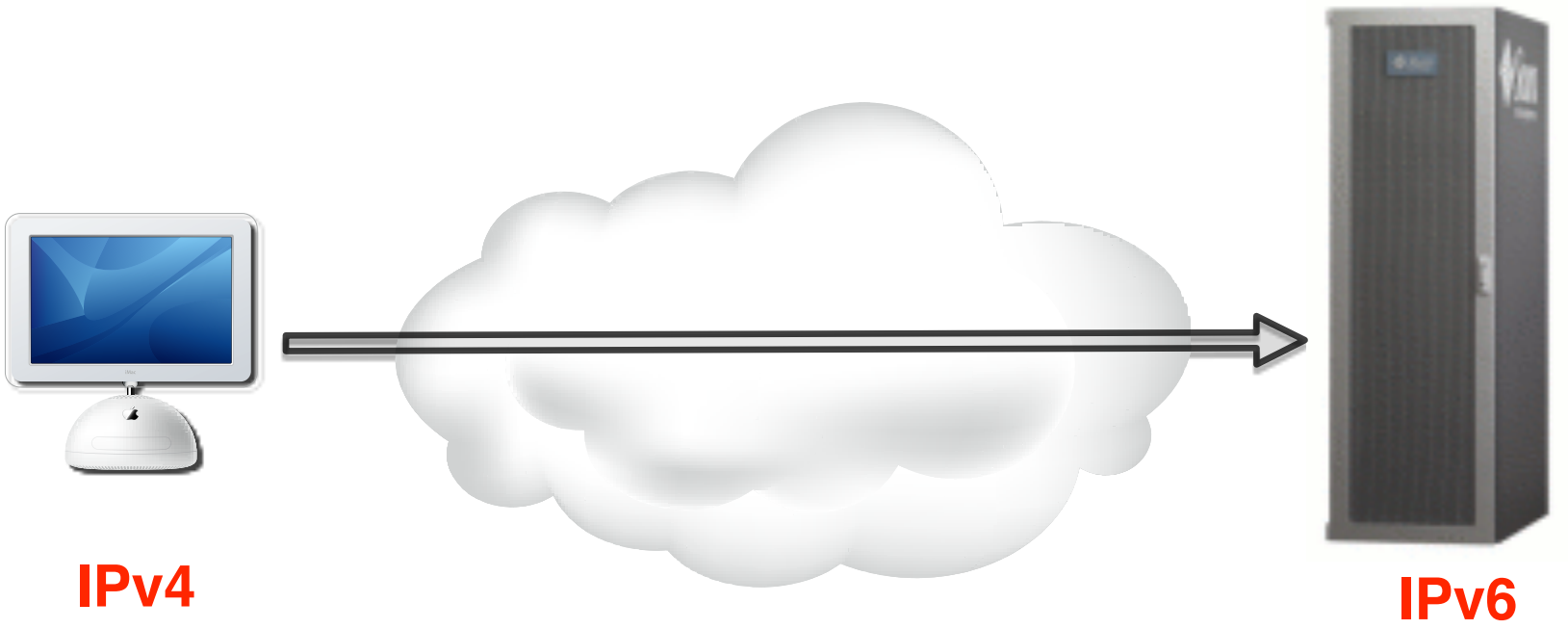
Schéma 4 : **Un système IPv6 se connecte à un système IPv6 à travers un réseau IPv4**



# Transition IPv4 → IPv6

## Schémas de cohabitation

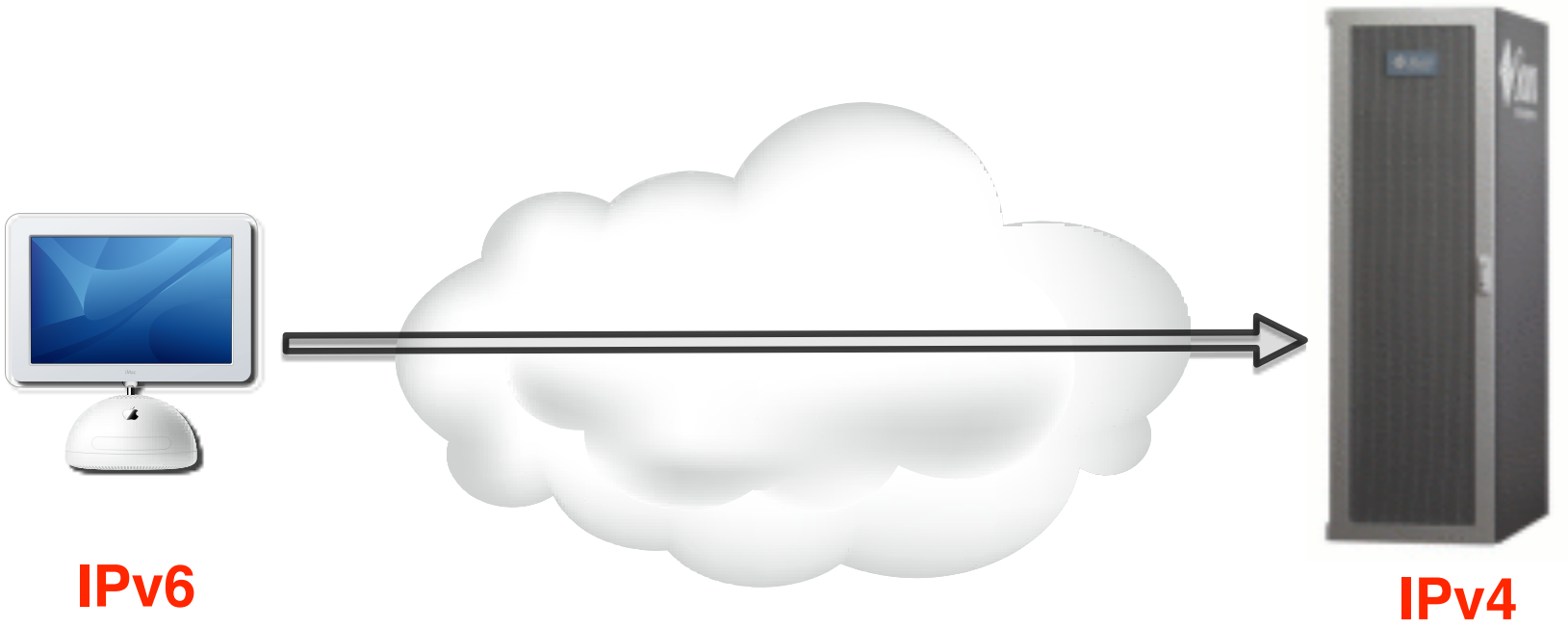
Schéma 5 : **Un système IPv4 se connecte à un système IPv6**



# Transition IPv4 → IPv6

## Schémas de cohabitation

Schéma 6 : **Un système IPv6 se connecte à un système IPv4**



# Cohabitation sur un réseau IPv4

## Sur un réseau IPv4

| | Hôte IPv4 | Hôte IPv6 |
|---|---|---|
| Hôte IPv4 | *Aucune adaptation* | Translation<br>Dual stack |
| Hôte IPv6 | Translation<br>Dual stack | Tunneling<br>Dual stack |

Tant que les **réseaux d'interconnexion sont en IPv4** , la migration vers IPv6 implique la mise en place de mécanismes. Deux cas de figure sont possibles :

- Déploiement d'un **réseau uniquement IPv6** . Ce qui néccessite un double mécanisme
  - Tunneling / 6to4
  - Translation
- Déploiement d'un **réseau IPv6 compatible IPv4** . La méthode utilisée est le **Dual Stack**

# Cohabitation sur un réseau IPv6

**sur un réseau IPv6**

| | Hôte IPv4 | Hôte IPv6 |
|---|---|---|
| Hôte IPv4 | Tunneling<br>Dual stack | Translation |
| Hôte IPv6 | Translation<br>Dual stack | *Aucune adaptation* |

Si la migration se fait sur la base d'un écosystème basé sur un **réseau d'interconnexion IPv6** :

- Mécanisme de **Translation**

Permet de traiter la seule problématique qui se pose : la **communication avec les systèmes IPv4**

# Familles de mécanismes de translation

## Dual stack

- RFC 4213
- Pas de variante

## Tunneling

- Manuel
  - Tunnel broker
  - Tunnel point-à-point
- Automatique
  - 6to4 (RFC3056)
  - ISATAP (RFC4214)
  - Toredo (RFC 4380)
  - 6over4
  - Dual Stack Transition Mechanism (DTSM)

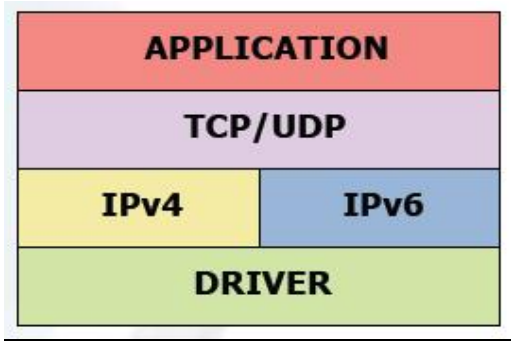
## Translation

- NAT-PT (RFC2765)
- Stateless IP/ICMP Translation (SIIT) (RFC6145)
- Transport Relay Translator (TET) (RFC3142)

# Dual stack (RFC4213)

**Principe** : utilisation de IPv4 et IPv6 **sur le même système**

→ douple pile IPv4 et IPv6




- Pour un réseau : nécessite la compatibilité de **tous les composants**
  - Hôtes, routeurs, applications et services
- L'application **choisit le protocole** (et l'adresse) IP à utiliser
  - Dans certains cas : version par défaut. Par exemple IE utilise IPv6 par défaut
- Ne doit pas avoir d' **impact sur les performances**
  - Activer IPv6 ne doit pas ralentir IPv4
  - Ajout d'une pile supplémentaire = traitements supplémentaires au niveau du routeur

# Dual stack (RFC4213)

**Avantages :**

- Assure la compatibilité entre les 2 protocoles pour les liens sélectionnés
- Permet de déployer globalement un réseau IPv6 en parallèle d'un réseau IPv4
- Peut être utilisé pour déployer un réseau IPv6 dans un environnement (réseau d'interconnexion) majoritairement IPv4
- Peut être utilisé pour garder opérationnels des réseaux IPv4 dans un environnement IPv6

**Inconvénients :**

- L'application doit choisir le protocole (et l'adresse) IP à utiliser
  - Les DNS retournent des adresses IPv4 (A record) et des adresses IPv6 (AAAA records)
  - Certaines applications basculent par défaut sur une version du protocole
- Durant la phase de transition, chaque machine a toujours besoin d'une adresse IPv4
  - Ne résout pas seul le problème d'épuisement des adresses IPv4

# Le Tunneling

**Principe** : encapsuler des paquets formatés suivant un protocole IPvX dans des paquets IPvY

- Les paquets peuvent alors être transportés sur des réseaux uniquement compatibles IPvY

Deux modes de tunneling :

- **Manuel** : le tunnel est créé manuellement
- **Automatique** : le tunnel est créé automatiquement sans intervention de l'utilisateur final

Migration IPv4 → IPv6 en deux temps :

- Tout d'abord, tunneling IPv6 sur un réseau IPv4
- Ensuite, une fois le réseau migré, tunneling IPv4 (vieilles applications) sur le réseau IPv6

**Mais... attention**

- Peut créer des canaux non-contrôlés / sécurisés par l'administrateur réseau
- L'usage de tunneling masque aux opérateurs/FAI l'existence d'une demande en IPv6
  - → Déficit d'incitation des opérateurs à passer à IPv6 malgré une demande existante

# Quelques tunnels

## Tunnels statiques

- 6in4 : code protocole 41 d'IP => attention parfois filtré.
- Anything In Anything ou AYIYA : IPv6 dans TCP ou UDP !
- GRE : code protocole 47 d'IP (connu).
- Tunnel broker : connexions entre brokers, méthode la plus utilisée de tunneling manuel

## Tunnels automatiques

- 6to4 : nécessite une adresse IPv4 publique (préfixe 2002 ajouté à l'adresse IPv4, puis des 0), code proto 41 d'IP. + adresse dédiée 192.88.99.1 réservée pour routeurs relais.
- 6rd (Rapid Dev.) comme 6to4 mais préfixe diff. suivant FAI (Free depuis 2007)
- ISATAP Intra-Site Automatic Tunnel Addressing Protocol : utilise IPv4 comme une couche 2 virtuelle (version améliorée de 6over4): génère une adresse IPv6 link-local à partir d'une IPv4 et fait du ND sur IPv4.
- TEREDO (windows), Miredo (Linux) : utilisable avec des IPv4 privées (encapsule IPv6 dans UDP/IPv4).
- Passerelles applicatives à double pile
  - exemple : serveurs web relais
- Traduction de protocole dans l'hôte
  - applications à code fermé en dur IPv4 : couche logicielle additionnelle (*Bump in the stack, bump in the API*, SOCKS).

# Translation

Permettent la **communication entre deux versions du protocole**

- **Traduit** en quelque sorte l'IPv4 en IPv6 et vice versa
- La Translation **modifie les paquets** de données IP

Peut se faire à différents niveaux :

- Au niveau **applicatif** : relais (Application Level Gateway, ALG) ou proxy
- Au niveau **transport** : relais TCP/UDP - Transport Relay Tanslator (TRT)
- Au niveau **réseau** : NAT-PT

**Mais... attention**

- Mécanismes complexes à mettre en place
- L'adresse IP est incluse dans le payload (données du paquet)
  - → Nécessite des traitements supplémentaires pour traiter et router les paquets
  - → Deep Packet Inspection

Ce sont des mécanismes **temporaires**

# Plan du cours

# Sous Linux

## Exemple

```
>$ /sbin/ifconfig -a |grep "adr inet6:"
adr inet6: ::1/128 Scope : Hôte
adr inet6: fe80::224:d7ff:fe1f:6624/64 Scope : Lien
```

## Notes

- Interfaces virtuelles IPv6-in-IPv4 : sit$n$ (sit0 réservée) : solution théoriquement obsolète.
- Préférence vers commande `ip` à la place de `ifconfig` et `route`.

# Commandes de base sous Linux

Ajout d'une adresse IPv6 à une interface :

```
/sbin/ifconfig eth0 inet6 add 3ffe:ffff:0:f101::1/64
```

⚠ Par défaut préfixe en /128 !

Suppression: `del` au lieu de `add`

## Nouvelles commandes

ping6, traceroute6, tracepath6, ip6tables,...

## Exemple

```
>$ ping6 -c 1 ::1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.024 ms
-- ::1 ping statistics --
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.024/0.024/0.024/0.000 ms
```

Table de routage : `route -A inet6` ensuite `add`, `del` comme avec IPv4.

# Commande ip

- Contrairement à `ifconfig` ne positionne pas le netmask automatiquement et ne fait pas appel à DNS.
- Syntaxe: ip [ OPTIONS ] OBJECT [ COMMAND [ ARGUMENTS ]]
- OBJECT := { link | addr | route | rule | neigh | tunnel | maddr | monitor... }

```
ip addr add 10.0.0.1/24 brd + dev eth0      # brd + pour calcul broadcast
        ip -s link show eth0                # affiche les stats
            ip link show                    # presque comme ifconfig -a
ip -6 addr add 2001:db8::1/32 dev eth0
            ip route show                   # presque comme route -n
            ip neigh show                   # visualise cache arp: arp -n
          ip -6 neigh show                  # Découverte de voisinage IPv6
          ip -6 addr flush                  # Purge toutes les adresses IPv6
ip -6 route add 2000::/3 dev eth0 metric 1  # ajout route, ⚠metric=1024 par déf.
```

# Plan du cours