INFO3 – S5
Introduction aux réseaux

Vincent Ricordel

vincent.ricordel@univ-nantes.fr
Bureau B217

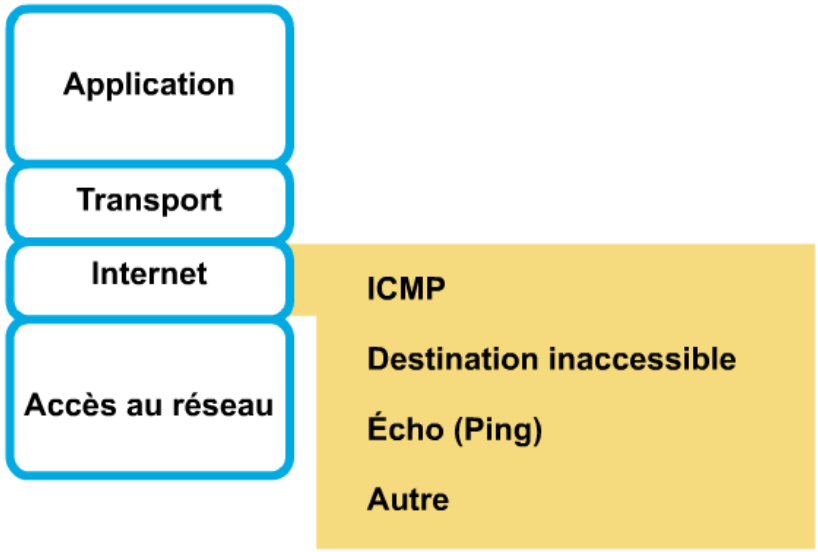
INFO3 – S5 Réseaux informatiques

- Accueil : 5 CM
- Introduction aux réseaux
  - 3 CM
  - 6 TD
  - 5 TP de 3h
  - 1 exam

Plan

- Bases
  - Réseaux LAN / WAN
  - Modèles
    - OSI
    - TCP/IP
    - Unités LAN et couches
- Réseaux locaux
  - Couche 1
  - Couche 2
  - Couche 3 Protocoles
  - Couches 4, 5, 6, 7

ICMP



«Internet Control Message Protocol »

Envoi de messages d'erreur et de contrôle

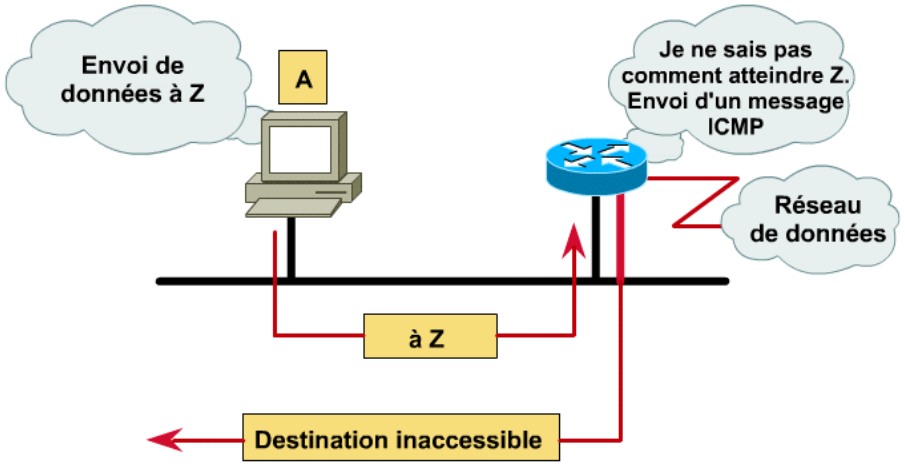
Ex. de messages ICMP :

- Destination inaccessible
- Durée de vie dépassée
- Problème de paramètres
- Source éteinte
- Réponse à la demande d'information
- Réponse à la demande d'adresse
- Redirection
- Écho / Réponse d'écho
- Horodatage / Réponse d'horodatage
- Demande d'information
- Demande d'adresse

Principe : à chaque message de demande (request) correspond un message de réponse (response)

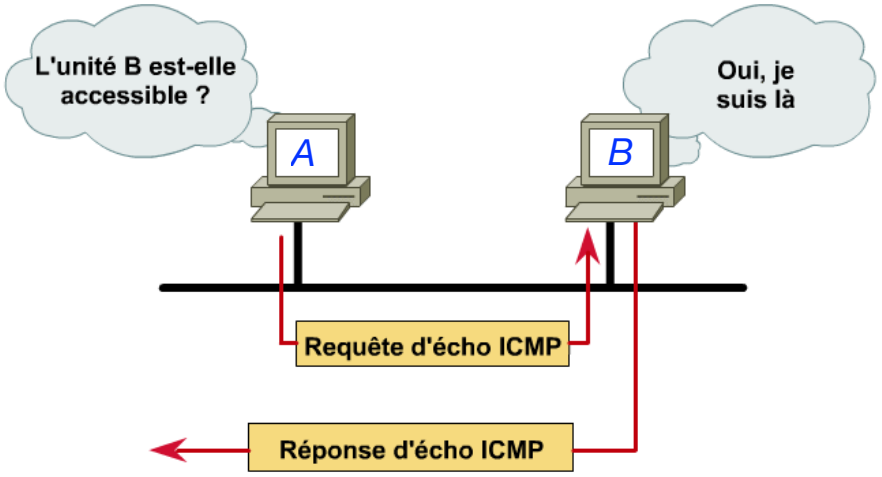
ICMP

Ex. Message d'erreur



Destination inaccessible
- Machine ou port inaccessible
- Réseau inaccessible

Ex. Messages (requête/réponse) de contrôle

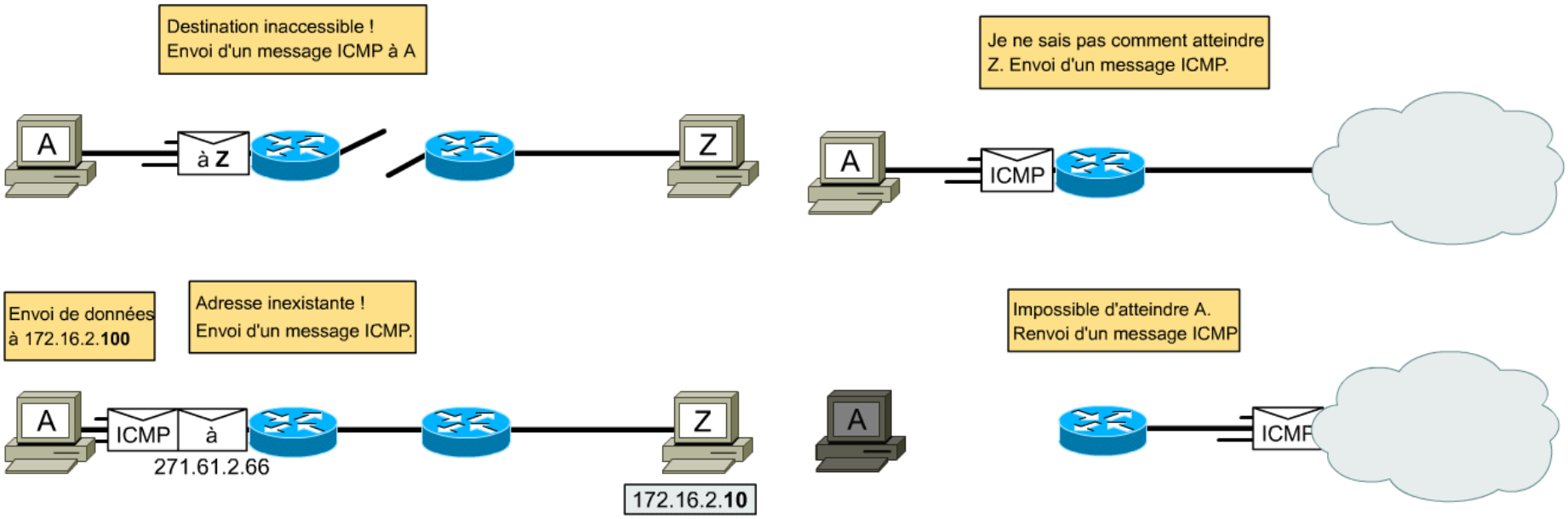


◆ Généré par la commande ping

Destination inaccessible

« Ping » :
echo request / echo reply

ICMP



Ex. Messages d'erreur
Paquet impossible à remettre

1 message ICMP qui n'a pu être remis est supprimé
=> Pour éviter les flots ininterrompus de messages d'erreur

ICMP

Messages de contrôle

Messages de type : **requête-réponse**

pour vérifier la présence

- type=0: code=0 - réponse echo - *reply*
- type=8: code=0 - requête echo - *request*

pour établir les tables de routage

- type=9: code=0 - réponse du routeur
- type=10: code=0 - requête au routeur

pour calibrer le *time-out*

- type=13: code=0 - requête *timestamp*
- type=14: code=0 - réponse *timestamp*

ICMP

Messages d'erreurs

Messages de type : **erreurs**

pour annoncer
l' inaccessibilité

type=3: code=0 - réseau inaccessible

type=3: code=1 - station inaccessible

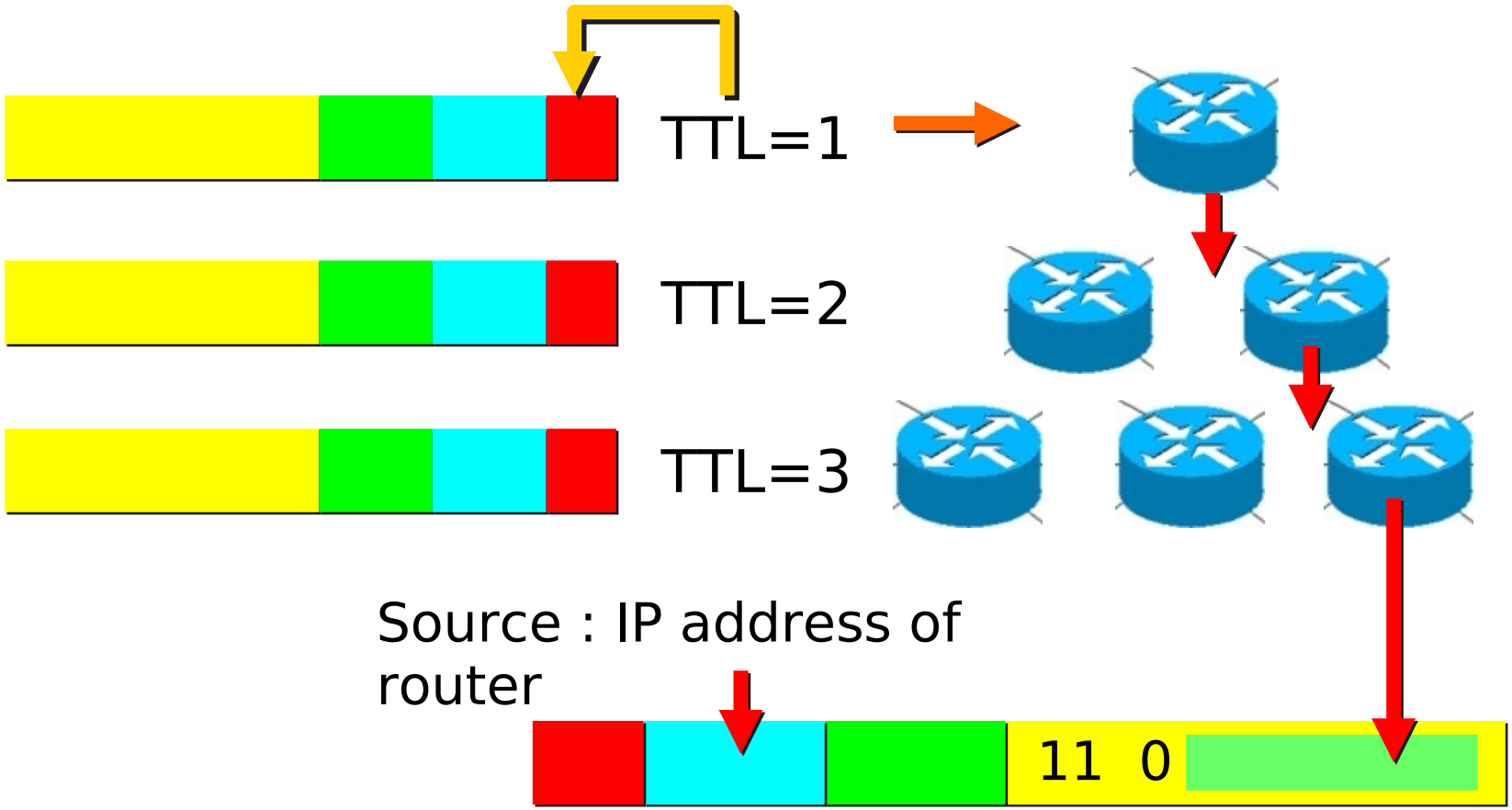
type=3: code=2 - protocole inaccessible

type=3: code=4 - port inaccessible

Identificateur contient les 8 premiers octets
de l'en-tête du datagramme IP ayant provoqué l'erreur
=> identification de ce datagramme

ICMP

Pour connaître un RTT « round trip time »
ou une route empruntée



Utilisé par
traceroute

ICMP: TTL expired

Message d'erreur

Broadcasting / multicasting

- Broadcasting - **diffusion** :

  envoi des trames & datagrammes à
  toutes les stations du même lien
  (dans un même réseau)
  => une adresse de diffusion

- Multicasting - **distribution** :

  envoi des trames & datagrammes
  vers de multiples destinations (sur différents réseaux)
  => une adresse de « multicast » (groupe)

Broadcasting - adressage

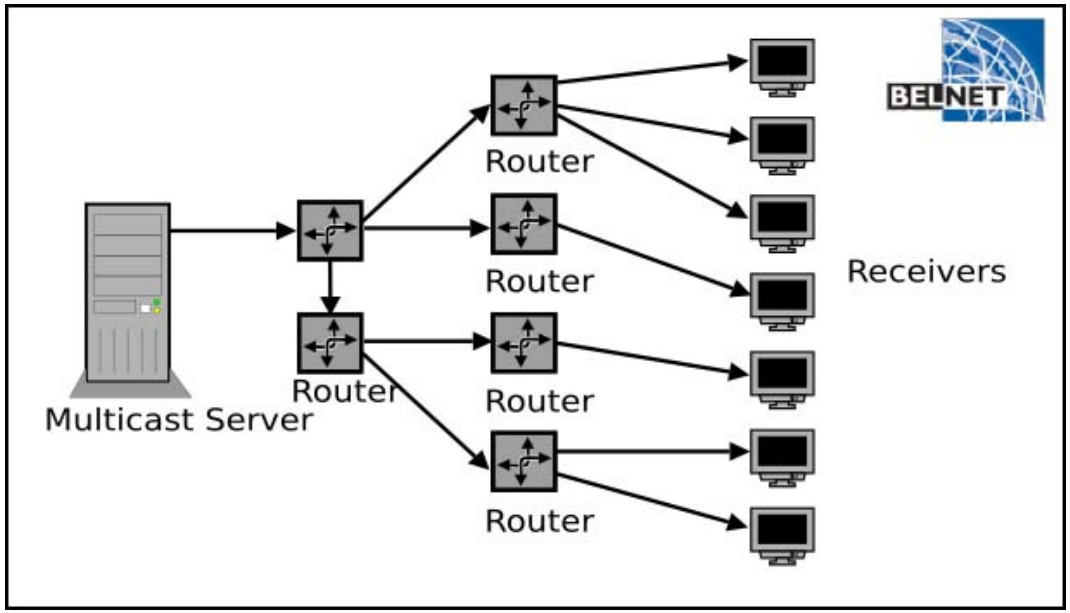
- Adresse de diffusion (niveau 3) :
  - @IP dont les bits du Host_ID sont à « 1 »
  - Ex. (en classe B) Net_ID.255.255
  - La couche 2 doit être capable aussi de diffuser les trames
    => une **@Mac_diffusion**

    Cf. Ethernet :
    @Mac_de_diffusion = FF:FF:FF:FF:FF:FF

Multicast

- Émetteur envoie 1 seul paquet
  => prise en charge de la diffusion au groupe par les routeurs multicast (une diffusion de niveau 3)

Diffusion multicast …



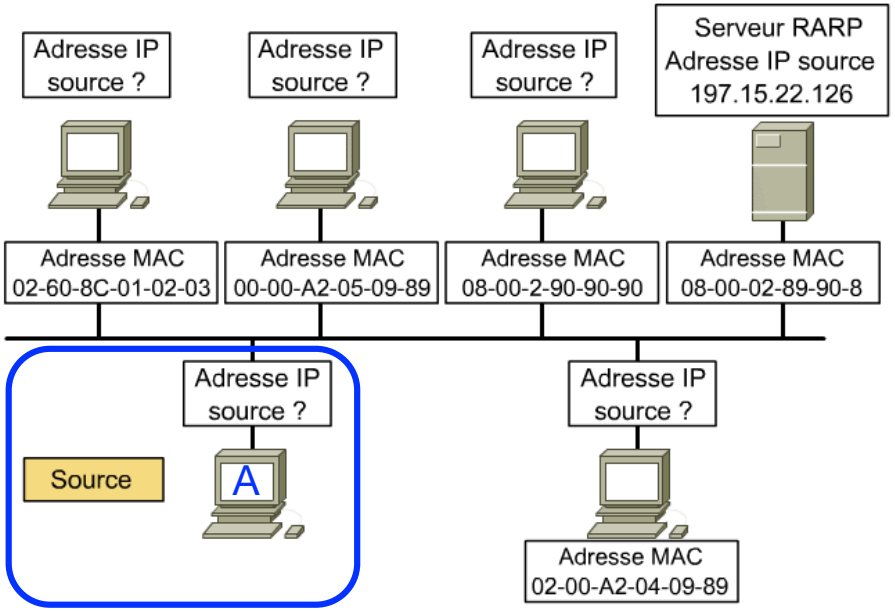
Multicasting - adressage

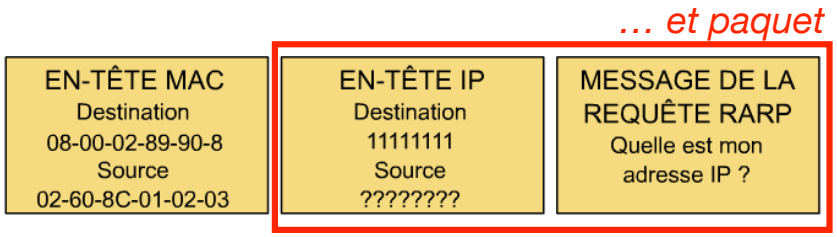
- @IP de groupes multicast
  entre : 224.0.0.0 à 239.255.255.255
- Groupe = ensemble des stations associées
  dynamiquement à l'adresse multicast
- Hist. : @IP_multicast assignées
  par l' IANA (*Internet Assigned Numbers Authority*)
  pour des groupes permanents

Attribution d'@IP

- Méthodes statiques :
  (association @MAC <-> @IP statique [fixée])
  - Directe
  - RARP, BOOTP (BOOTstrap) ...
- Méthodes dynamiques :
  (association @MAC <-> @IP dynamique)
  - DHCP

RARP





La requête (trame)

- « Reverse Address Resolution Protocol »
  Résolution inverse d'adresse
- Usage : terminaux sans HD
- Associer
  @MAC_connue <=> @IP_inconnue
- Comment :
  - Code RARP en ROM
    => @MAC_serveur_RARP connue
  - Démarrage station :
    - Requête RARP :
      @MAC_serveur | @IP_broadcast
    - Serveur RARP répond à l'hôte
      @MAC_hôte | @IP_broadcast

DHCP

- « Dynamic Host Configuration Protocol »
  => complète BOOTP
- Serveur DHCP a :
  - 1 @IP fixe propre
  - 1 plage d'@IP_hôte à attribuer dynamiquement
    (nombre @IP_hôte) **parfois <** (nombre des hôtes)
- Hôte :
  - fait 1 requête DHCP
    => Broadcast (+ type de requête)
  - Serveur répond par Broadcast
    pour attribuer @IP et configuration (masque, @IP_passerelle...)
- Système de baux :
  - @IP avec 1 date_début et 1 date_fin de validité
  - Client peut demander prolongation de son bail
  - Serveur peut demander au client s'il veut prolonger son bail

DHCP

Serveur

Optimisation => bonne gestion durée des baux

(si aucune @IP n'est libérée,
aucune requête d'DHCP ne peut être satisfaite)
+ ne pas multiplier les diffusions inutiles

Gestion des baux :

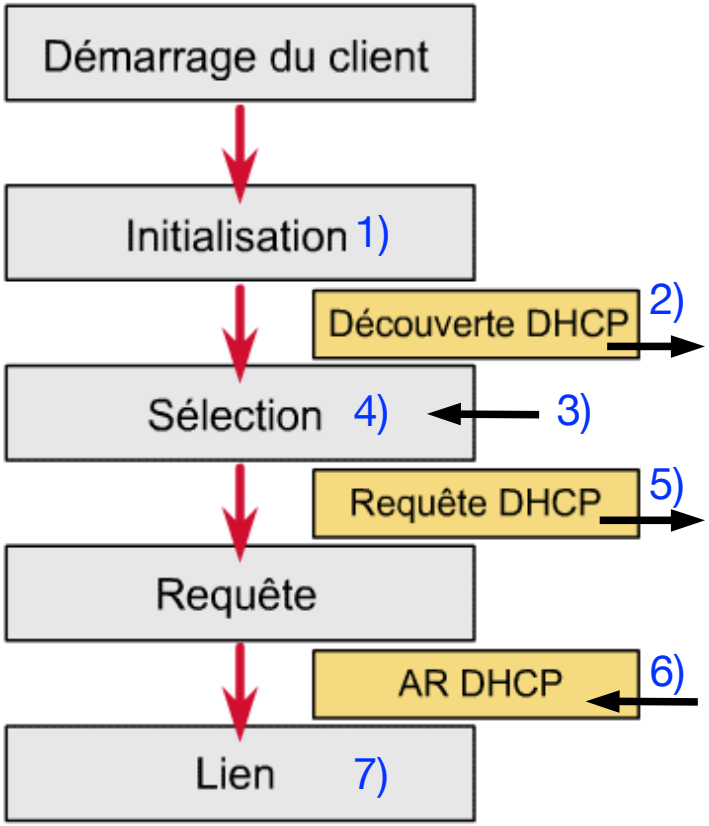
- Courte durée : si beaucoup de PC « se branchent & se débranchent »
- Longue durée : si machines « fixes »

DHCP

Plusieurs types de paquets (de **requêtes**) :

- Client :
  - DHCP_DISCOVER : localisation du ou des serveurs DHCP
  - DHCP_REQUEST : diverses requêtes (ex. prolongement d'un bail)
  - DHCP_DECLINE : car a déjà 1 @IP
  - DHCP_RELEASE : libération de son @IP
  - DHCP_INFORM : demande d'information (@IP déjà obtenue)
- Serveur :
  - DHCP_OFFER : 1ère réponse (et 1ers paramètres) du serveur
  - DHCP_ACK : accusé réception (+ @IP du client & paramètres)
  - DHCP_NACK : annonce échéance d'un bail ou mauvaise configuration

DHCP



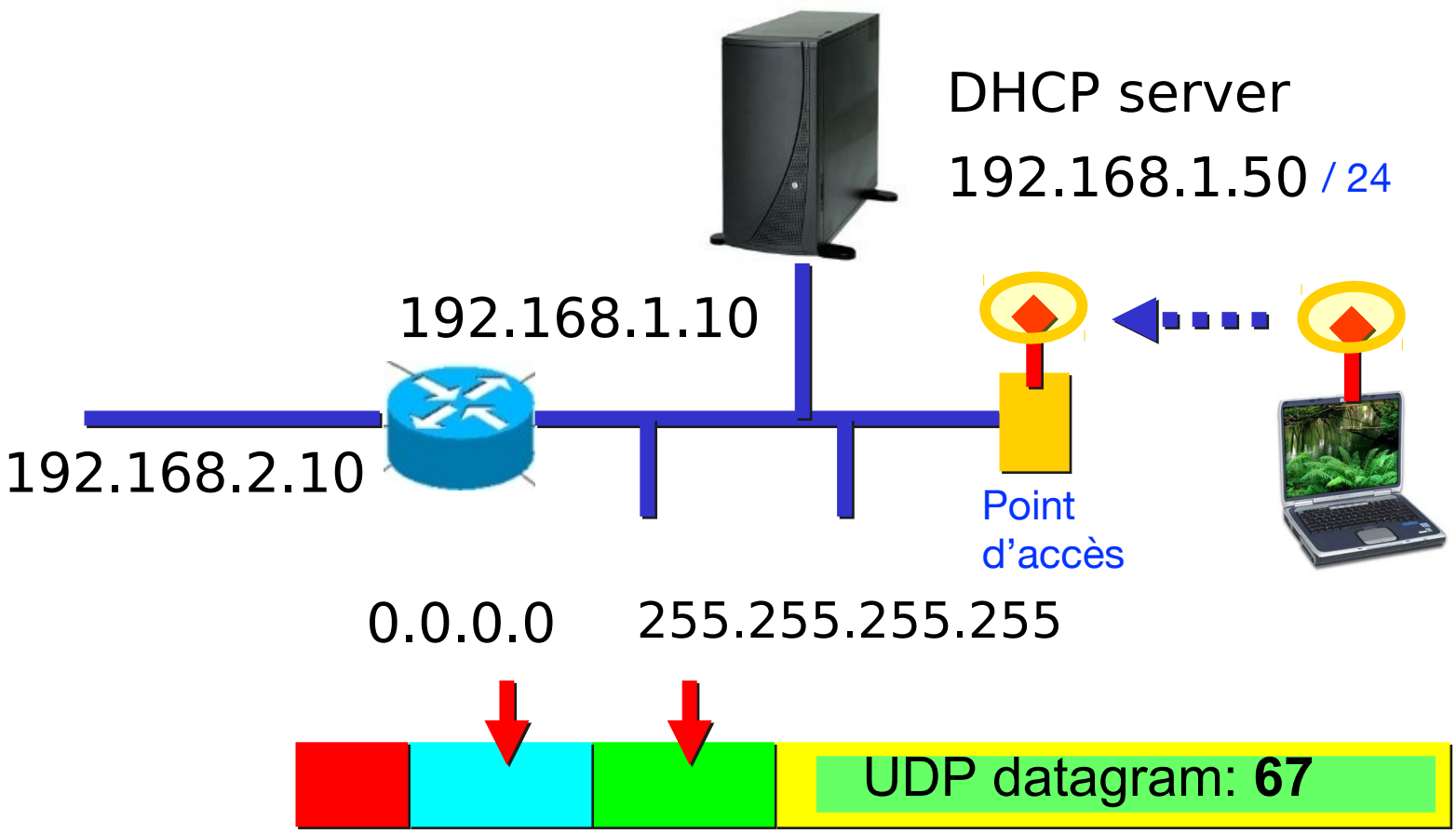
Coté client

- 1) Client (**initialisation**) :
  2) émet en broadcast **DHCP_DISCOVER**
  -> segment UDP
  (vers n°port 67 du serveur)
- 3) Serveur :
  fait une offre DHCP_OFFER en Broadcast
  (le client n'a pas encore d'@IP)
- 4) Client (**sélection**) :
  Recueille les offres du (des) serveur(s)
  => sélectionne la 1ère réponse
  5) + négocie durée d'utilisation de l'@IP
  (**DHCP_REQUEST**)
- 6) Serveur (accusé réception) :
  Envoie **DHCP_ACK**
- 7) Client :
  Peut utiliser @IP reçue

DHCP

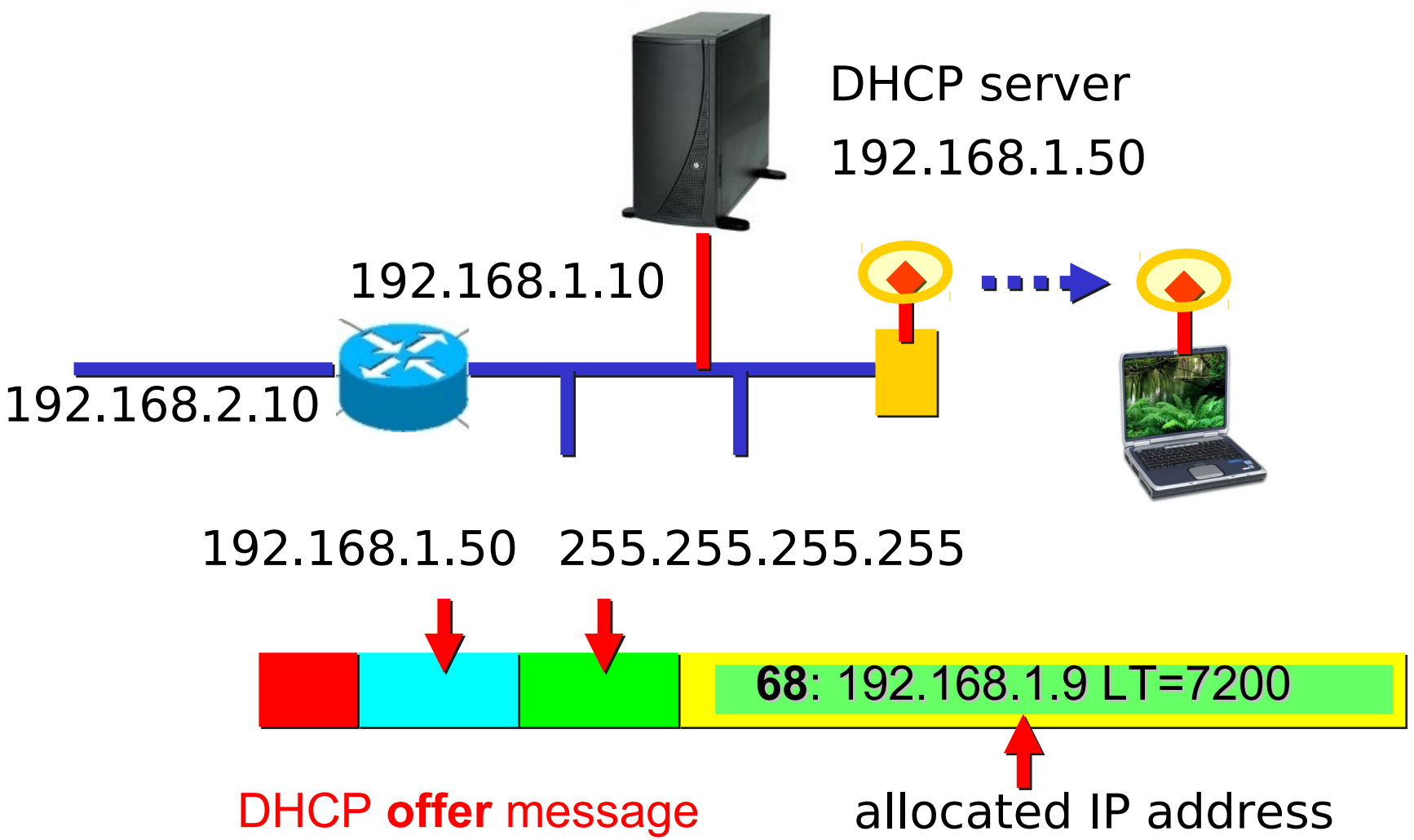
```
[dupont@pc] $ sudo dhclient wlan0

Listening on LPF/wlan0/00:0d:88:99:12:e1
Sending on   LPF/wlan0/00:0d:88:99:12:e1
Sending on   Socket/fallback/fallback-net
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPOFFER from 192.168.0.254
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.254
bound to 192.168.0.1 -- renewal in 432000 seconds.
```

DHCP

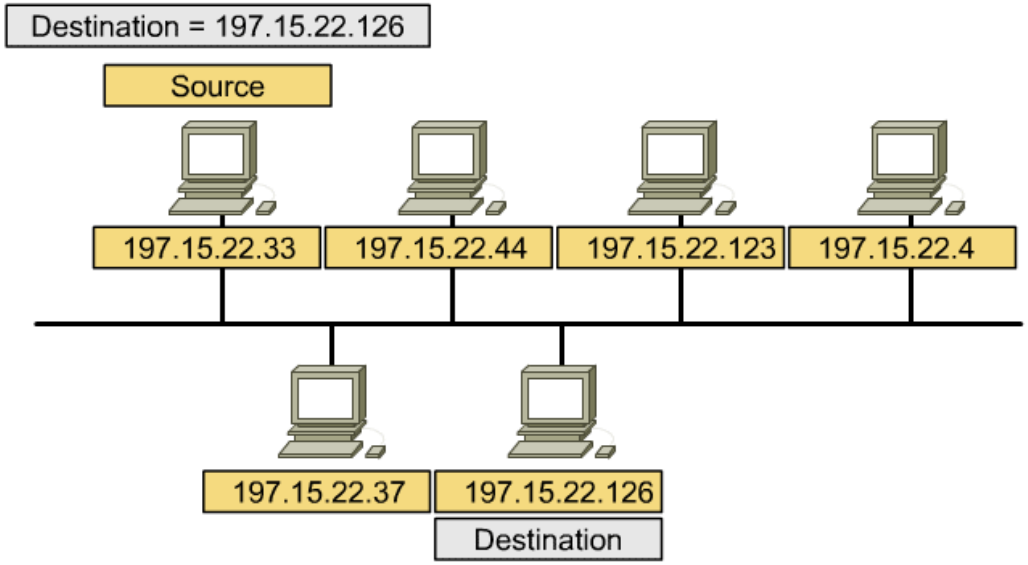


server **discovery** message

DHCP


DHCP server
192.168.1.50
192.168.1.10
192.168.2.10
192.168.1.50 255.255.255.255
68: 192.168.1.9 LT=7200
DHCP offer message
allocated IP address

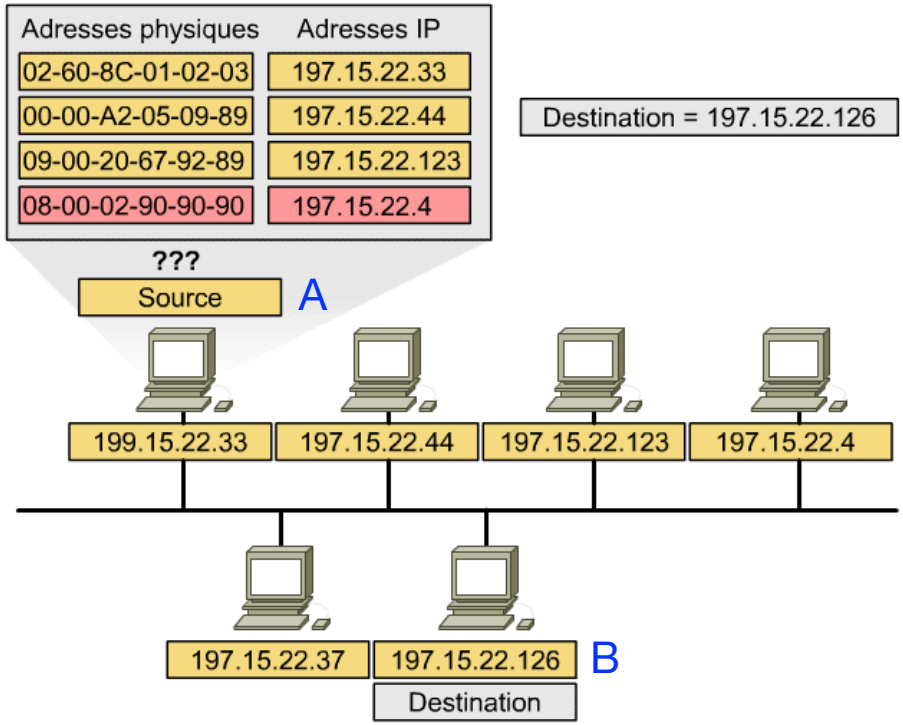
ARP - Rôle



« Address Resolution Protocol »
Paquet encapsulé dans une trame :
➔ Prochain saut de la trame ??
=> @MAC_destination ??

ARP

Table ARP de la machine A



Chaque hôte a sa table ARP

Correspondance « mappage » :
@MAC <-> @IP

Commande :
arp (linux, win.)
Table : /proc/net/arp (linux)
ip neighbour

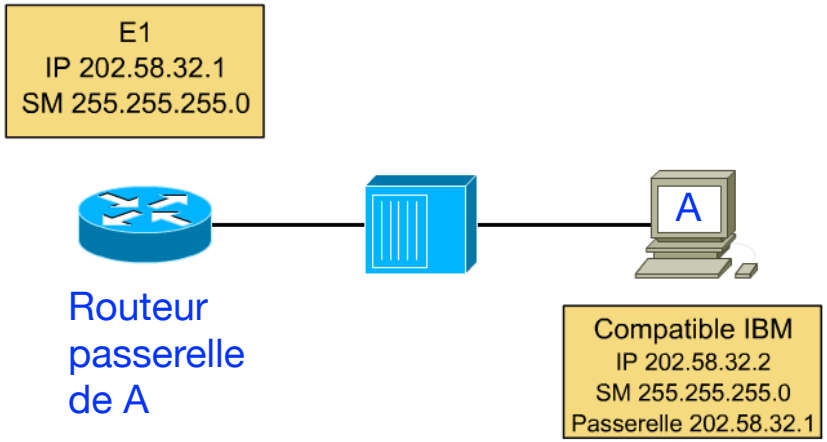
Si table ARP à jour
=> pas de problème

Cas simple :

Source & destination appartenant au même réseau local

ARP

Cas complexe :
Source & destination appartenant à des réseaux différents (communication inter-réseau)



Routeur bloque les diffusions
=> Impossibilité d'interroger au delà du routeur

➔ Transmission via « passerelle par défaut » (puis routage)

ARP

Comment Source sait si Destination est dans le même réseau ?

En **comparant @IP_réseau_source & @IP_réseau_destination**

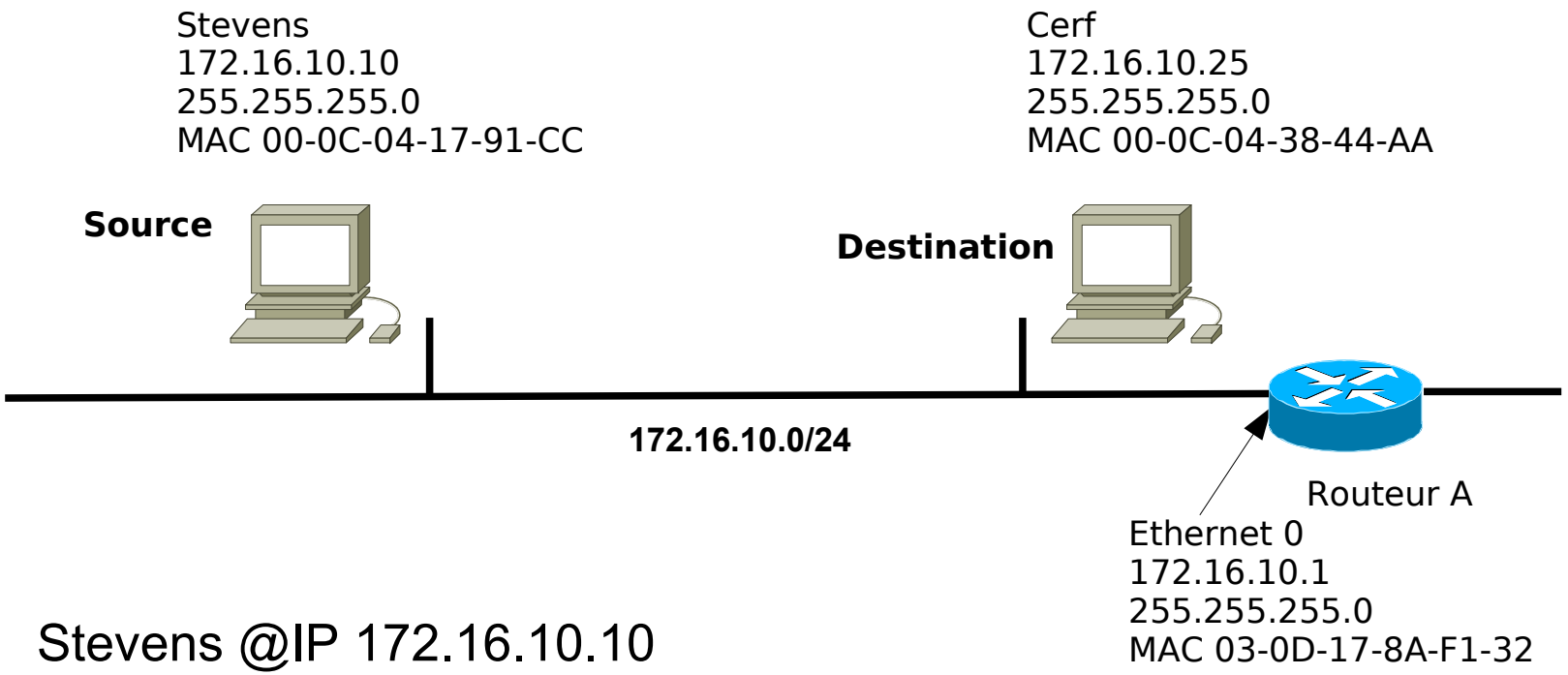
@IP_réseau_source = @IP_source & Masque_source
@IP_réseau_destination = @IP_destination & Masque_source

2 cas :

- Égalité => Hôtes dans le même réseau
  => Envoi direct des données
- Sinon (différence) => Hôtes dans des réseaux différents
  => Source envoie ses données à sa passerelle

ARP

Exemple simple : Hôtes appartenant au même réseau



Stevens @IP 172.16.10.10
souhaite envoyer paquet à
Cerf @IP 172.16.10.25

ARP

@IP_Stevens	172.16.10.10
Masque réseau Stevens	255.255.255.0
---	---
Réseau Stevens	**172.16.10.0**

@IP_Cerf	172.16.10.25
Masque réseau Steven	255.255.255.0
---	---
Réseau Cerf	**172.16.10.0**

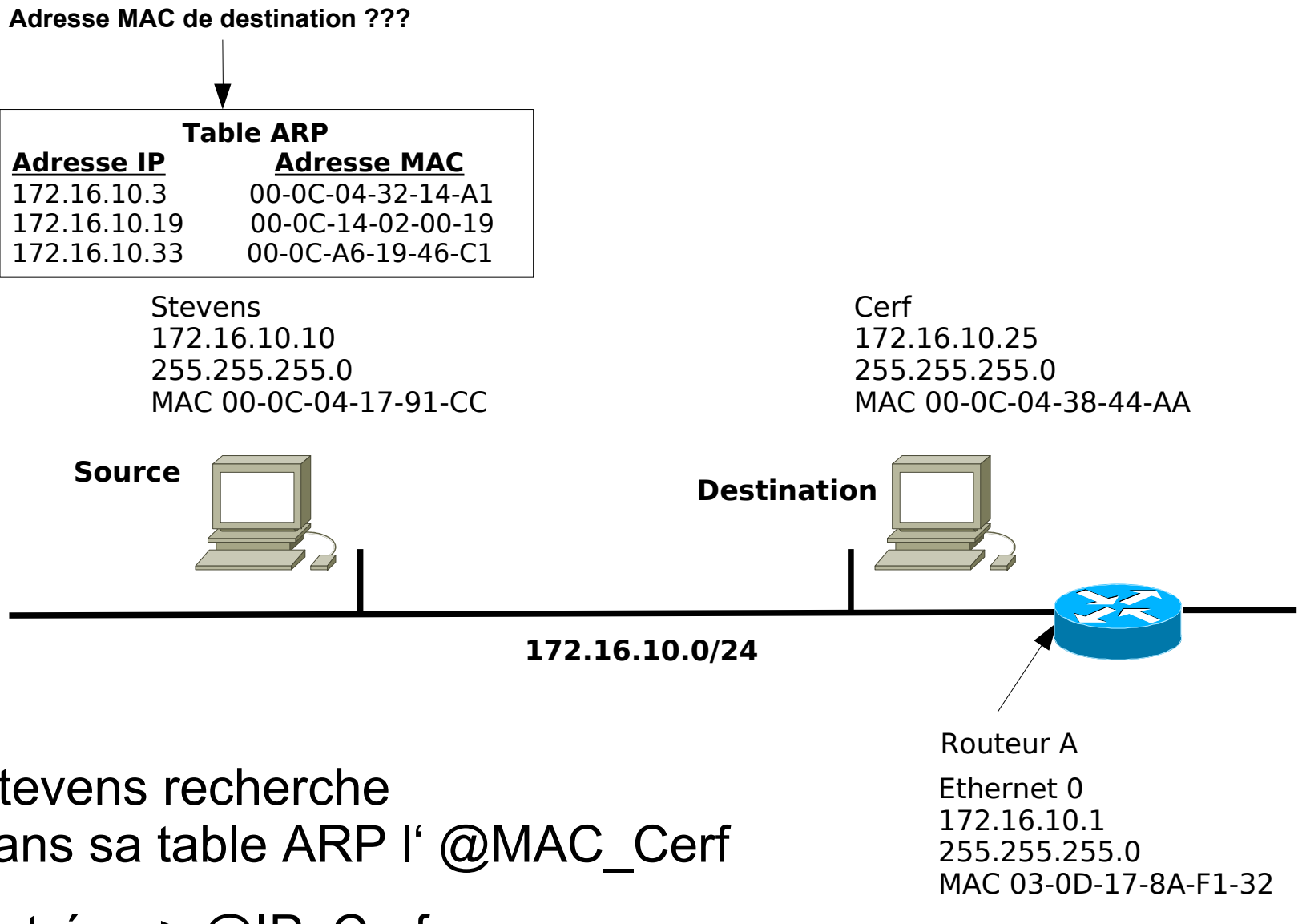
Stevens détermine que Cerf
est sur le **même réseau**

=> encapsulation paquet et envoi direct de la trame

=> recherche @IP_Cerf dans sa table ARP

... Ex. 1

ARP



Stevens recherche
dans sa table ARP l‘ @MAC_Cerf

Entrée => @IP_Cerf

ARP

Si @IP_Cerf n'est pas dans
table ARP de Stevens :

=> Stevens envoie 1 requête ARP
sur l‘ @IP_Cerf 172.16.10.25

Stevens peut émettre directement
la requête ARP pour Cerf
car sait que les 2 sont dans le même réseau

ARP

Requête ARP émise par Stevens @IP 172.16.10.10

" Diffusion : Who has 172.16.10.25 ?
je veux joindre cette @IP et
j'aimerais que cet hôte me renvoie son @MAC"

Requête ARP en provenance de 172.16.10.10

Paquet

En-tête Ethernet			**Données Ethernet – requête/réponse ARP 28 octets**				
Adresse de destination Ethernet (MAC)	Adresse d'origine Ethernet (MAC)	Type de trame	En-têtes ARP (champ op)	Adresse Ethernet (MAC) de l'émetteur	Adresse IP de l'émetteur	Adresse Ethernet (MAC) du destinataire	Adresse IP du destinataire
FF-FF-FF-FF-FF-FF	00-0C-04-17-91-CC	0x806	op = 1	00-0C-04-17-91-CC	172.16.10.10	??	172.16.10.25

Trame

champ op – Requête ARP = 1
Réponse ARP = 2
Requête RARP = 3
Réponse RARP = 4

ARP

Réponse ARP provenant de Cerf @IP 172.16.10.25

" Émetteur de la requête ARP,
voici l‘ @MAC dont tu as besoin. "

Réponse ARP en provenance de 172.16.10.25

Paquet

En-tête Ethernet			**Données Ethernet – requête/réponse ARP 28 octets**				
Adresse de destination Ethernet (MAC)	Adresse d'origine Ethernet (MAC)	Type de trame	En-têtes ARP (champ op)	Adresse Ethernet (MAC) de l'émetteur	Adresse IP de l’émetteur	Adresse Ethernet (MAC) du destinataire	Adresse IP du destinataire
00-0C-04-17-91-CC	00-0C-04-38-44-AA	0x806	op = 2	00-0C-04-38-44-AA	172.16.10.25	00-0C-04-17-91-CC	172.16.10.10

Trame

C'est celle-ci !

ARP

- Stevens
  * reçoit réponse ARP
  * ajoute @MAC et @IP Cerf dans sa table ARP
- Stevens peut ensuite
  * encapsuler paquet IP dans la trame
  * envoyer directement le paquet à Cerf

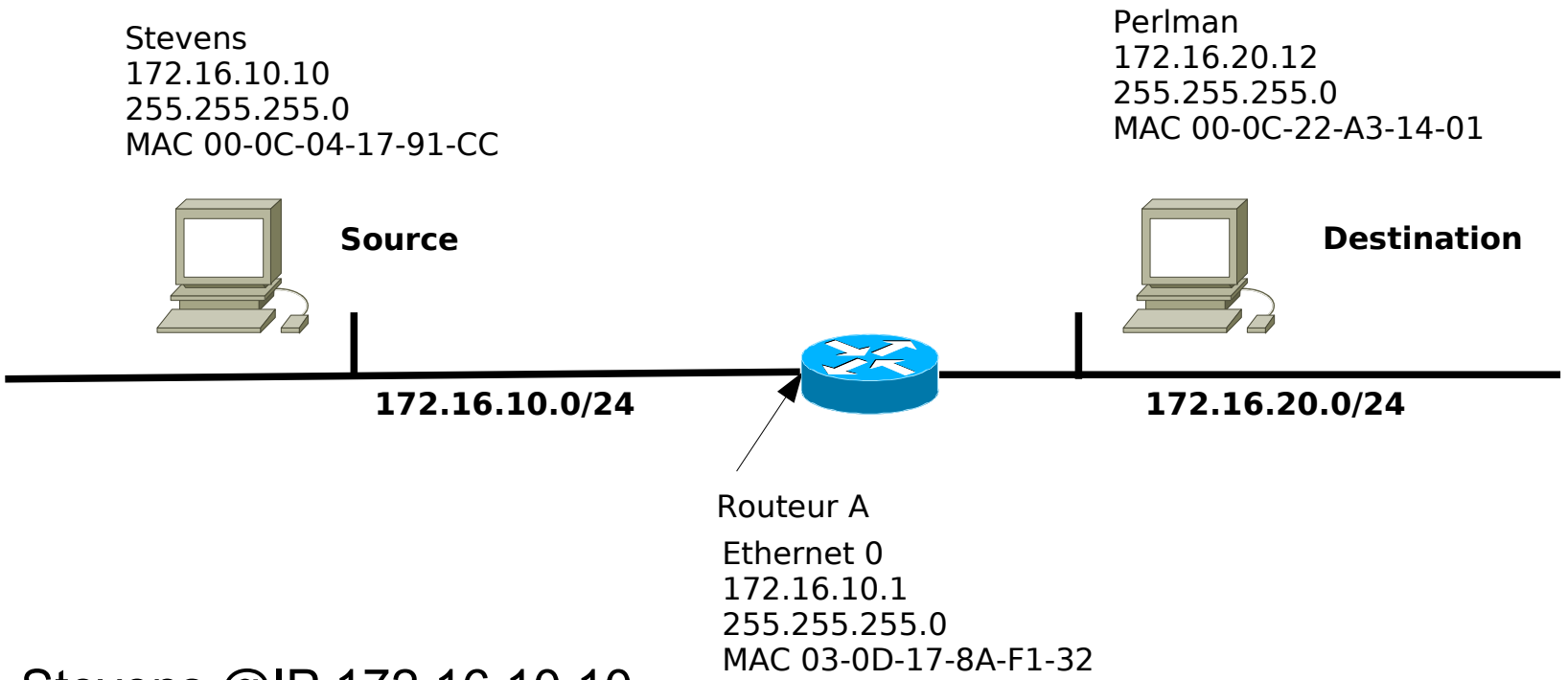
Trame Ethernet

Paquet

En-tête Ethernet			Datagramme IP du niveau supérieur				En-queue Ethernet
Adresse MAC de destination **00-0C-04-38-44-AA**	Adresse MAC d'origine 00-0C-04-17-91-CC	Autres informations d'en-tête	Informations d'en-tête IP	Adresse IP d'origine 172.17.10.10	Adresse IP destination finale **172.16.10.25**	Données	FCS

ARP

Exemple complexe : hôtes dans des réseaux différents



Stevens @IP 172.16.10.10
souhaite envoyer un paquet IP à
Perlman @IP 172.16.20.12

ARP

@IP Stevens	172.16.10.10
Masque Stevens	255.255.255.0
---	---
Réseau Stevens	**172.16.10.0**

@IP Perlman	172.16.20.12
Masque Stevens	255.255.255.0
---	---
Réseau Perlman	**172.16.20.0**

=> 2 réseaux différents

ARP

➔ **2 réseaux différents**

➔ Stevens

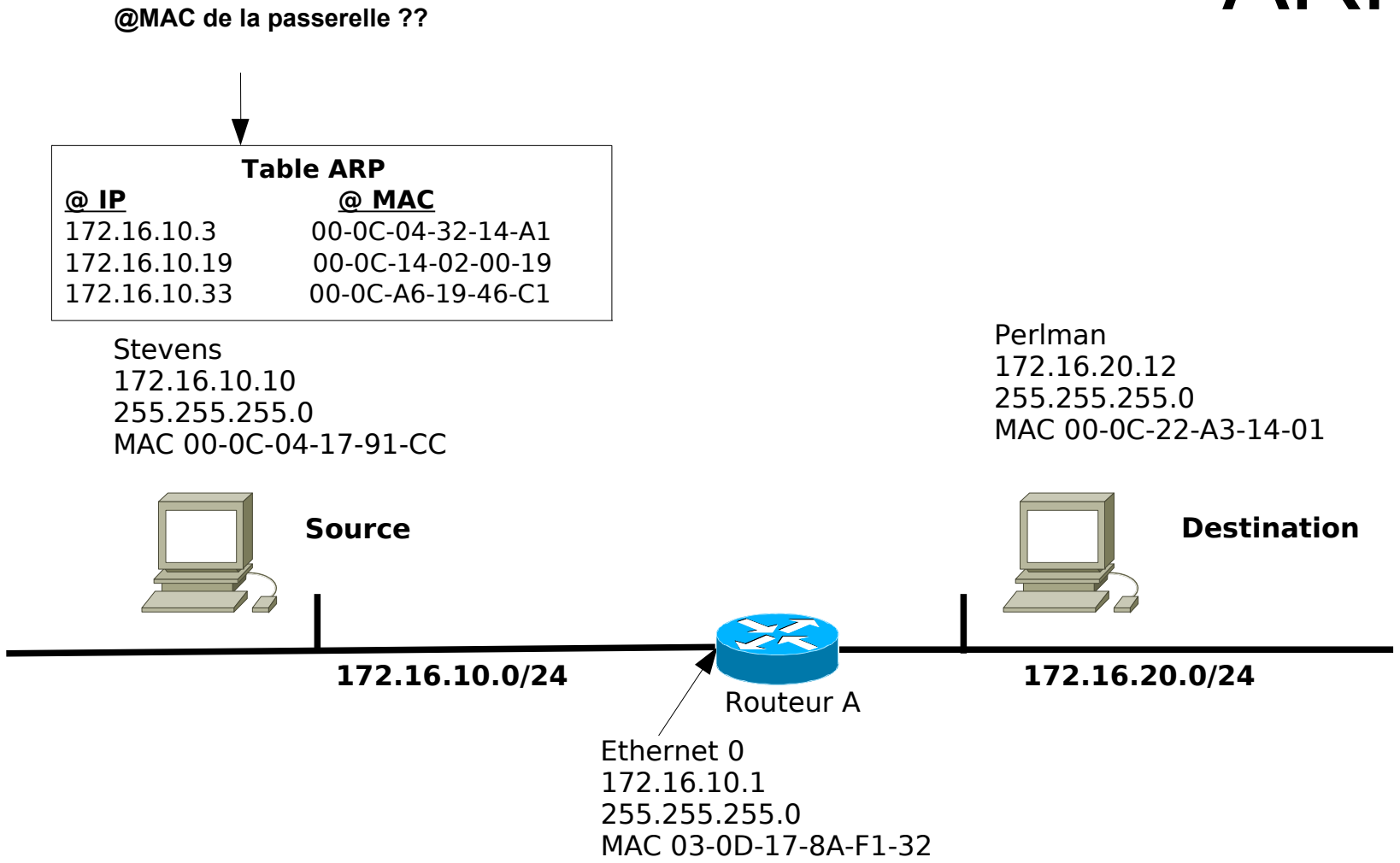
- ne peut **pas** envoyer le paquet directement
- doit envoyer le paquet à sa passerelle
  « routeur passerelle par défaut »

=> recherche @MAC_passerelle
dans sa table ARP

(@IP_routeur déjà saisie manuellement
ou obtenue via serveur DHCP)

... Ex. 2

ARP



Stevens recherche @MAC_routeur
dans sa table ARP
=> Entrée @IP_routeur

ARP

- **Si @IP_passerelle n'est pas** dans table ARP de Stevens

  => Stevens doit envoyer
  1 requête ARP portant sur
  @IP 172.16.10.1 du routeur A

ARP

Requête ARP émise par Stevens @IP 172.16.10.10

" Diffusion :
je veux joindre l‘ @IP 172.16.10.1 et
j'aimerais que cette unité me renvoie son @MAC "

Who has 172.16.10.1 ?

Requête ARP en provenance de 172.16.10.10

Paquet

En-tête Ethernet			**Données Ethernet – requête/réponse ARP 28 octets**				
Adresse de destination Ethernet (MAC)	Adresse d'origine Ethernet (MAC)	Type de trame	En-têtes ARP (champ op)	Adresse Ethernet (MAC) de l'émetteur	Adresse IP de l'émetteur	Adresse Ethernet(MAC) du destinataire	Adresse IP du destinataire
FF-FF-FF-FF-FF-FF	00-0C-04-17-91-CC	0x806	op = 1	00-0C-04-17-91-CC	172.16.10.10		**172.16.10.1**

Trame

champ op – Requête ARP = 1
Réponse ARP = 2
Requête RARP = 3
Réponse RARP = 4

ARP

Réponse ARP du routeur A ayant l' @IP 172.16.10.1

" Émetteur de la requête ARP,
voici l‘ @MAC dont tu as besoin ".

Réponse ARP en provenance de 172.16.10.1

Paquet

En-tête Ethernet			**Données Ethernet – requête/réponse ARP 28 octets**				
Adresse de destination Ethernet (MAC)	Adresse d'origine Ethernet (MAC)	Type de trame	En-têtes ARP (champ op)	Adresse Ethernet(MAC) de l'émetteur	Adresse IP de l'émetteur	Adresse Ethernet(MAC) du destinataire	Adresse IP du destinataire
00-0C-04-17-91-CC	03-0D-17-8A-F1-32	0x806	op = 2	03-0D-17-8A-F1-32	172.16.10.1	00-0C-04-17-91-CC	172.16.10.10

Trame

C'est celle-ci !

ARP

- Stevens :
  * reçoit réponse ARP
  * ajoute @MAC et @IP du routeur A dans sa table ARP
- Stevens peut ensuite :
  * encapsuler paquet IP dans trame Ethernet
  * envoyer ce paquet au routeur A.

Ethernet Frame

Paquet

En-tête Ethernet			Datagramme IP du niveau supérieur				En-queue Etherne t
Adresse MAC de destination **03-0D-17-8A-F1-32**	Adresse MAC d'origine 00-0C-04-17-91-CC	Autres informations d'en-tête	Informations d'en-tête IP	Adresse IP d'origine 172.17.10.10	Adresse IP de destination finale **172.16.20.12**	Données	FCS

- Ce sera au tour du routeur A d'acheminer le paquet (1er saut du routage).

ARP synthèse

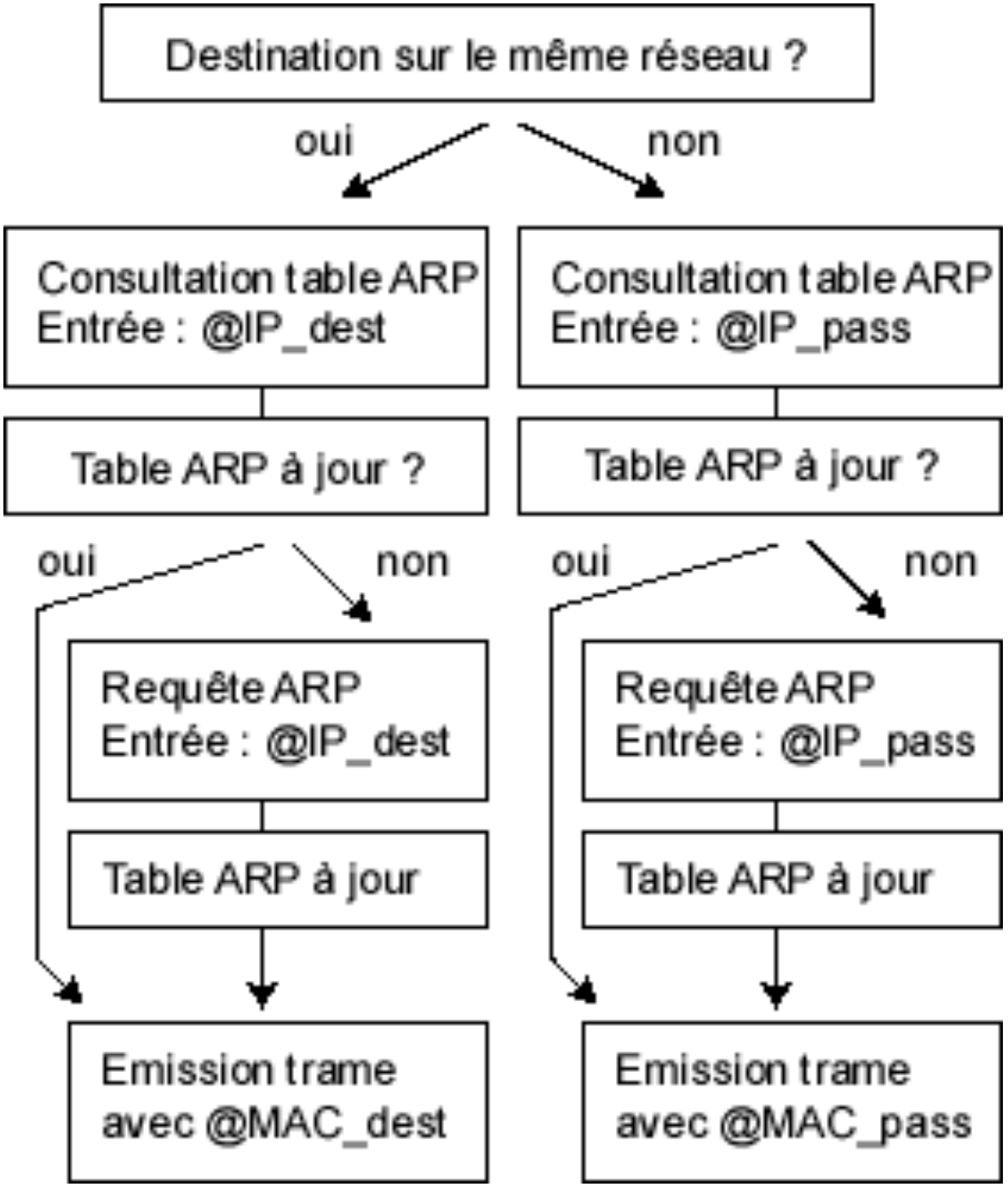
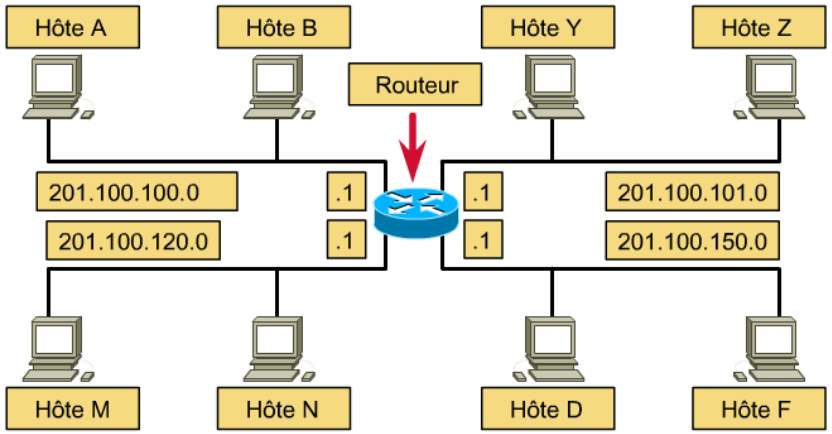


Table ARP d'un routeur



Réseau de destination	Port du routeur
201.100.100.0	201.100.100.1
201.100.101.0	201.100.101.1
201.100.120.0	201.100.120.1
201.100.150.0	201.100.150.1

Pour chaque réseau auquel il est relié :

- @MAC et @IP des ports du routeur
- @MAC et @IP des hôtes

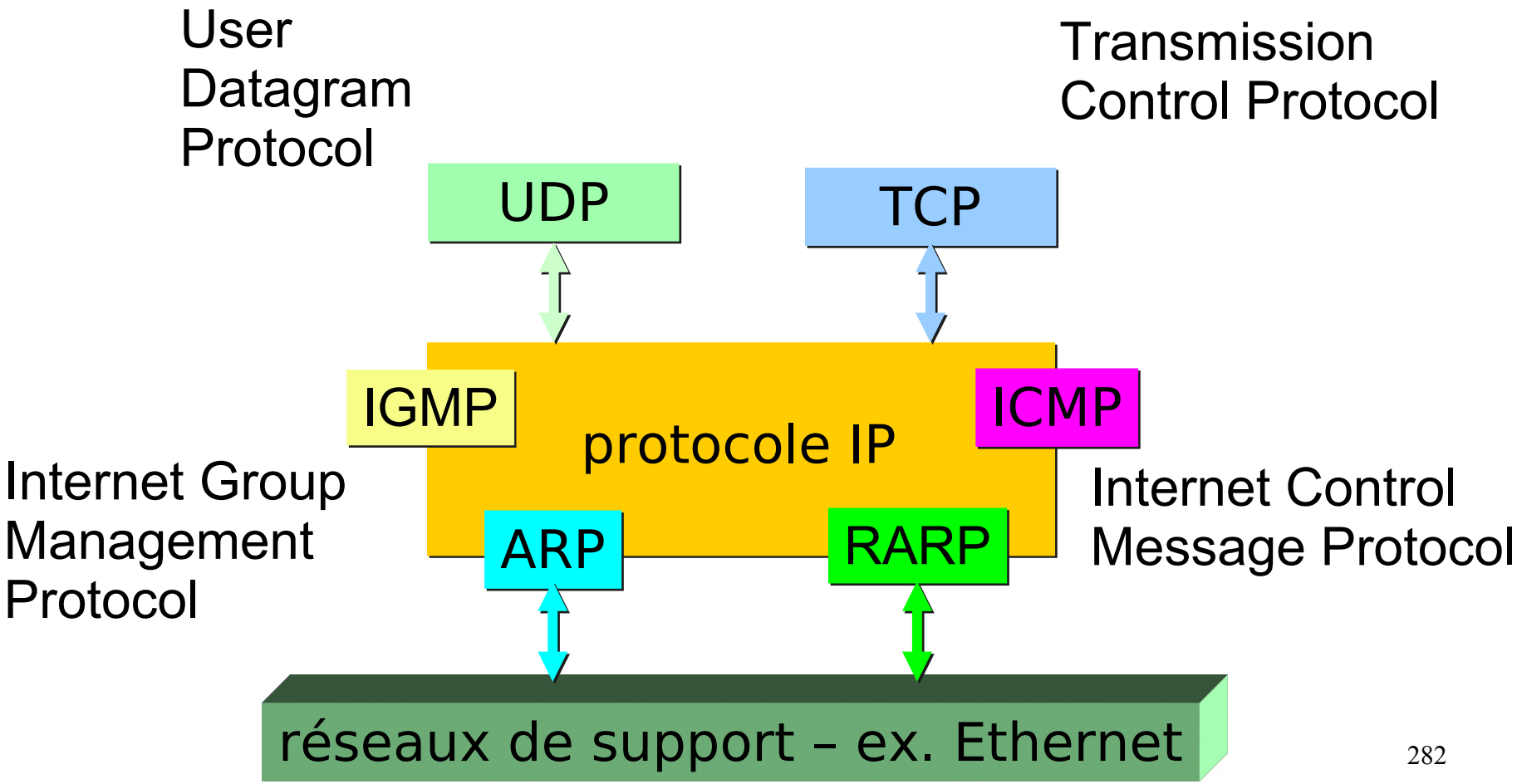
Pour les autres routeurs auxquels il est relié :

leur @MAC et @IP

=> vers une hiérarchie complexe de routeurs interconnectés

(requêtes ARP entre routeurs, routeurs par défaut, proxy ARP…)

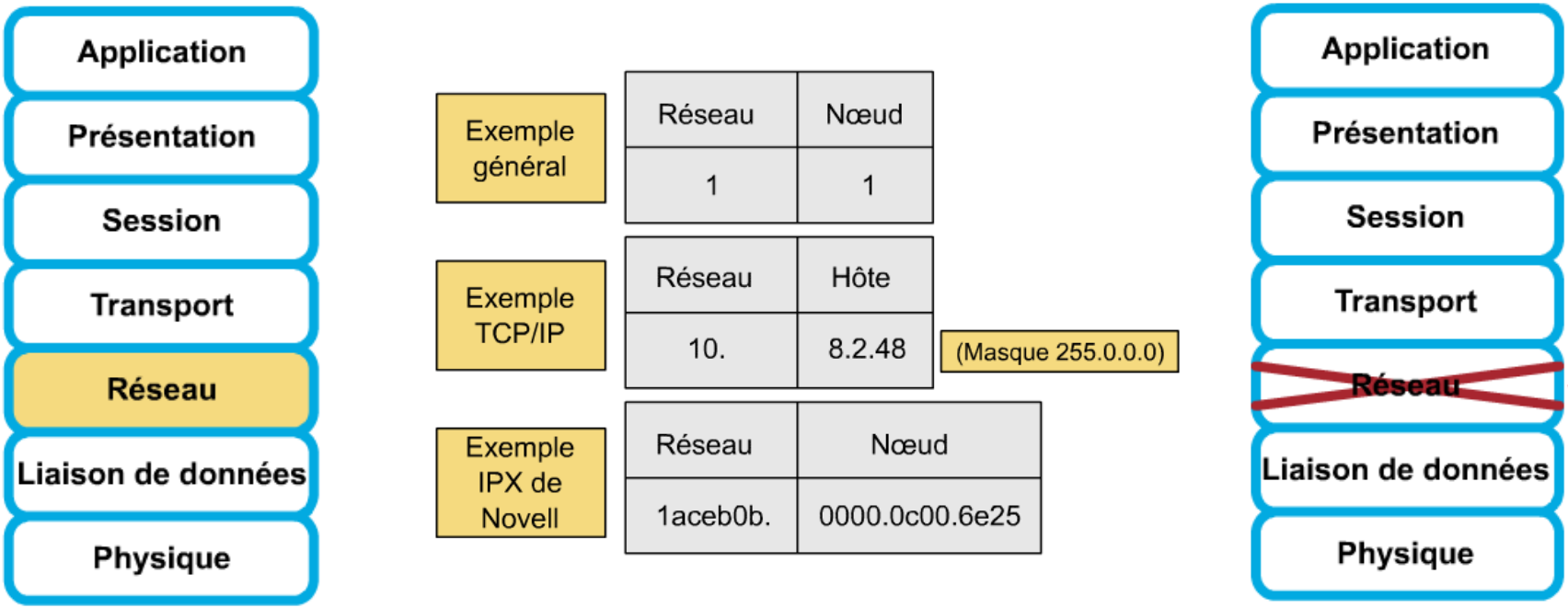
IP et sa “famille”



Plan

- Bases
  - Réseaux LAN / WAN
  - Modèles
    - OSI
    - TCP/IP
    - Unités LAN et couches
- Réseaux locaux
  - Couche 1
  - Couche 2
  - Couche 3 Routage
  - Couche 4, 5, 6, 7

Protocole routable / non routable



Protocole routable :

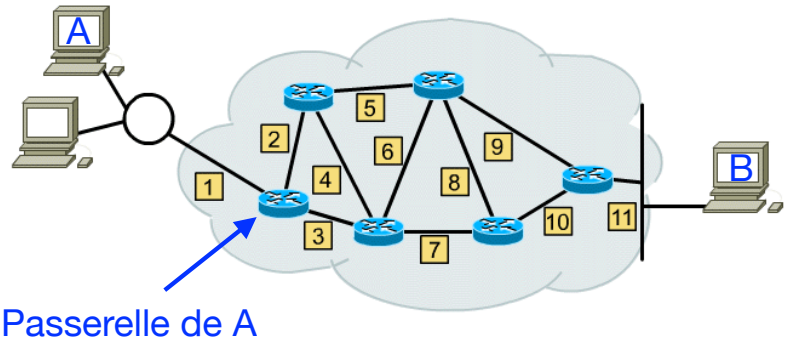
- Ex. : IP (TCP/IP), AppleTalk, IPX (IPX/SPX de Novell)
- 1 couche 3 définie
  => Protocole routé

Protocole non routable :

- Ex. : NetBEUI
- Pas de prise en charge de la couche 3
  => Non routé

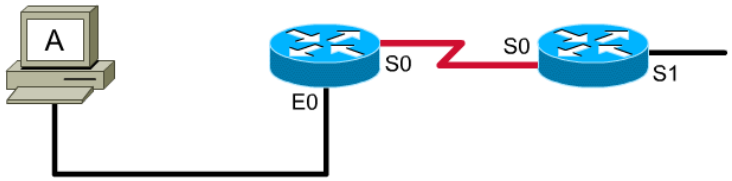
Routage - Principes

Fonctions de base du routeur :



Les adresses représentent le chemin des connexions du média.

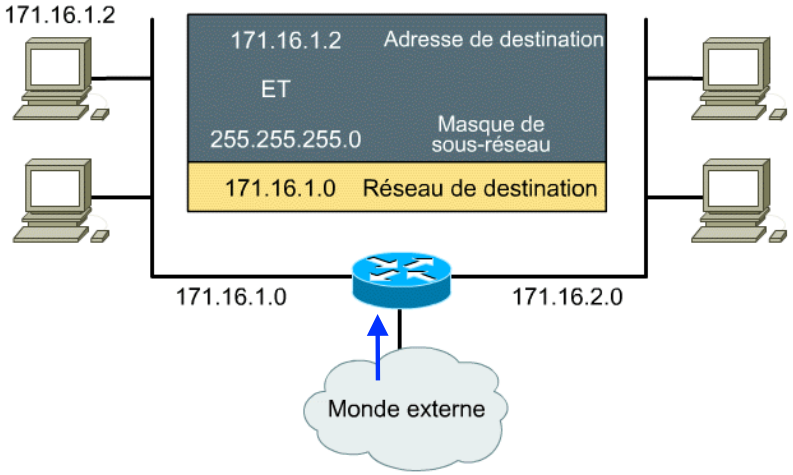
	Réseau de destination	Interface (saut suivant)
	172.16.0.0	S0
	172.18.0.0	--
	192.168.24.0	S0
Aucune correspondance	Routeur par défaut	S1

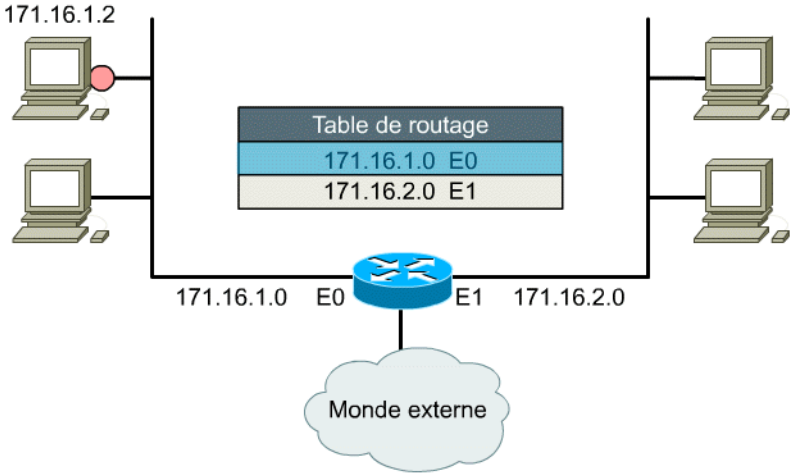


Détermination du chemin pour les paquets
&
commutation des paquets

Utilise sa table de routage
=> prochain saut

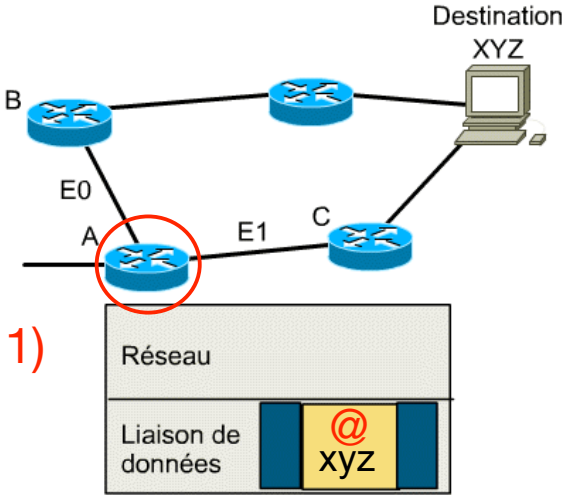
Routage - Principes

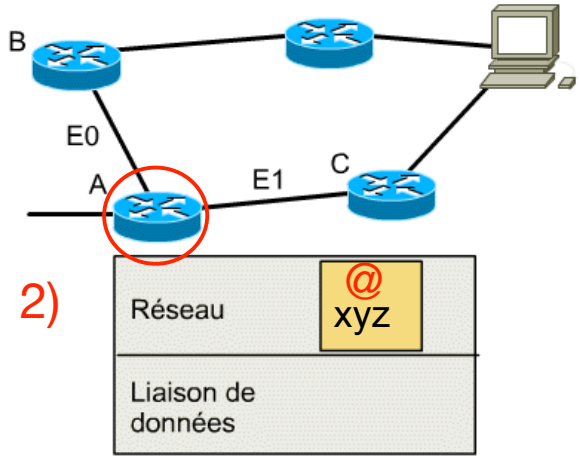




Détermination de l'adresse du réseau destination
=> par masquage

Routage - Principes





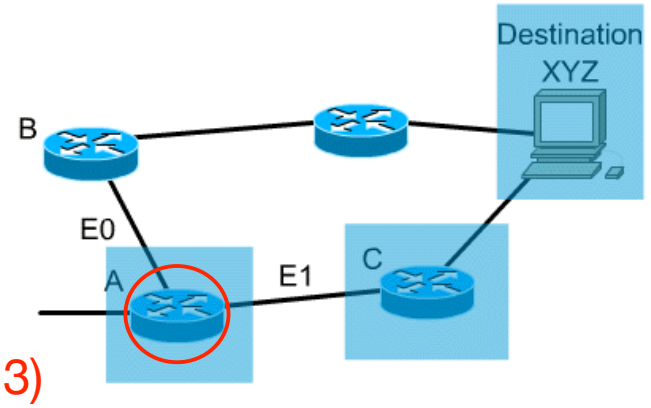
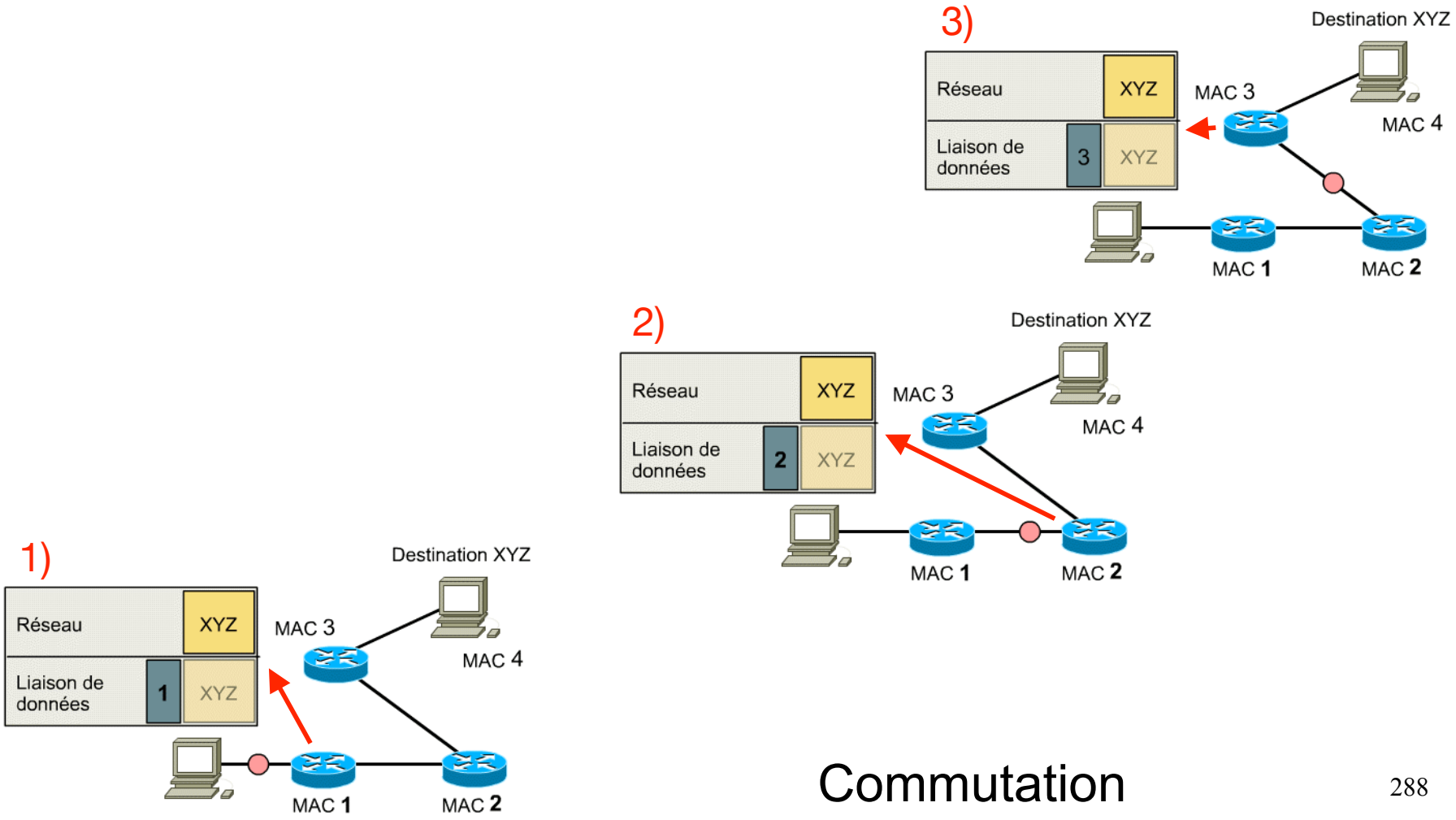


Table de routage de A

Interface	Avantages	Saut suivant	Destination
E1	+	Routeur C	XYZ
E0	-	Routeur B	XYZ

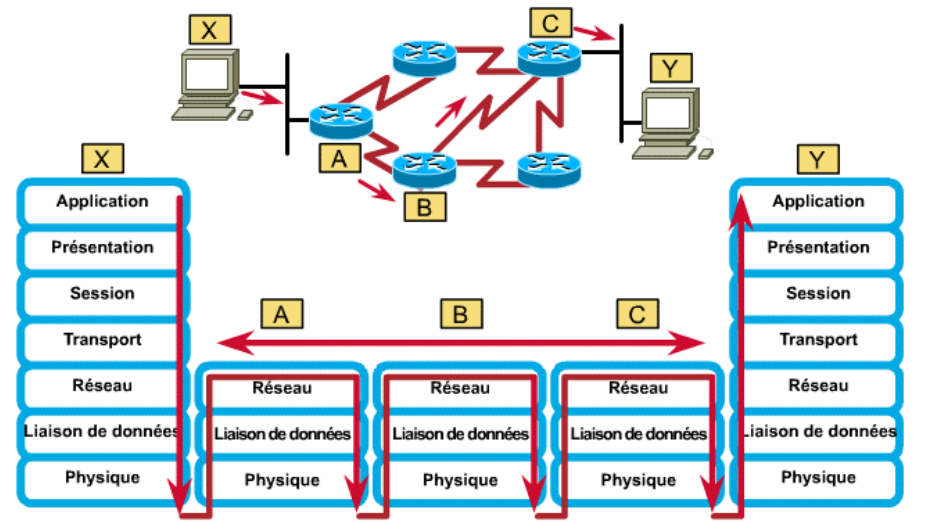
Routage de paquets
→ saut de proche en proche
jusqu'à destination

Routage



Commutation

Routage



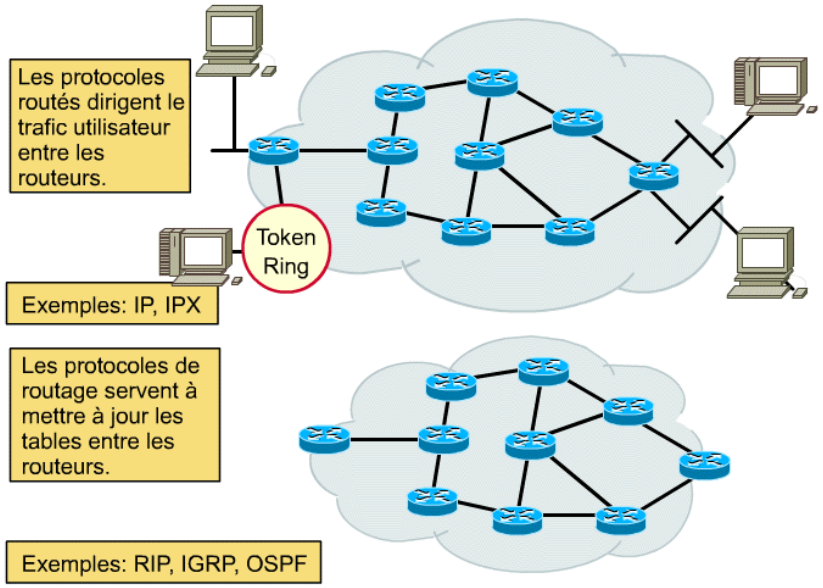
- Chaque routeur fournit ses services pour la prise en charge des fonctions de la couche supérieure.

4 Trames 1 Paquet

Source	Dest.	Source	Dest.	
@MAC_X	@MAC_A1	@IP_X	@IP_Y	data
@MAC_A2	@MAC_B1	@IP_X	@IP_Y	data
@MAC_B2	@MAC_C1	@IP_X	@IP_Y	data
@MAC_C2	@MAC_Y	@IP_X	@IP_Y	data

Couches :
4 => Liaison de bout en bout
3 => Routage du paquet
2 => Saut de trames entre 2 machines adjacentes (sur un même lien)

Protocoles routés / de routage



Ex. protocoles de routage :

- RIP (*Routing Information Protocol*)
- IGRP (*Interior Gateway Routing Protocol*)
- EIGRP (*Enhanced Interior Gateway Routing Protocol*)
- SPF ouvert

=> servent à actualiser les tables de routage

Protocoles de routage

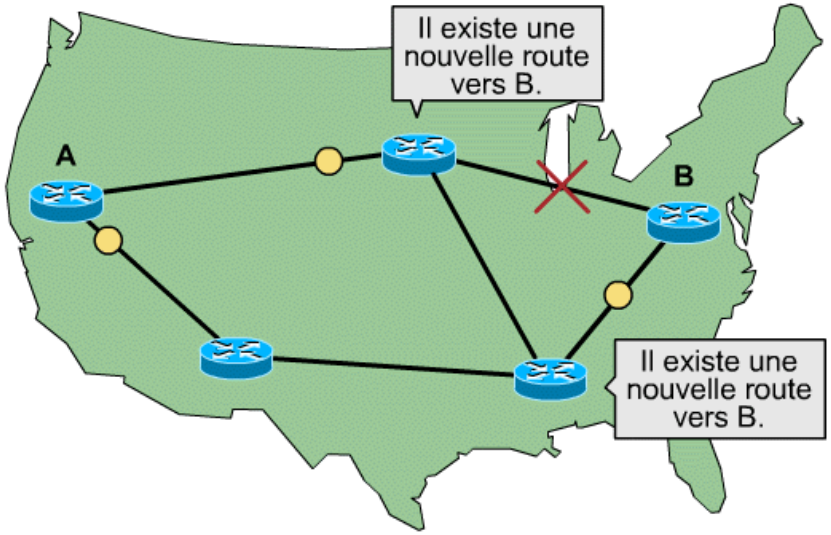
- Routeurs doivent déterminer **dynamiquement** le meilleur chemin vers les réseaux destination
- Table de routage :
  - Intégrée dans le routeur
  - « Échangée » entre routeurs interconnectés
  - Utilisée pour cartographier les chemins vers les réseaux destination

Routage statique / dynamique

=> Routes **statiques** configurées manuellement
Saisie par l'administrateur réseau dans le routeur.

=> Routes **dynamiques** acquises automatiquement via le protocole de routage.
Ajustement automatique des routes à emprunter en fonction des modifications de topologie ou de trafic.

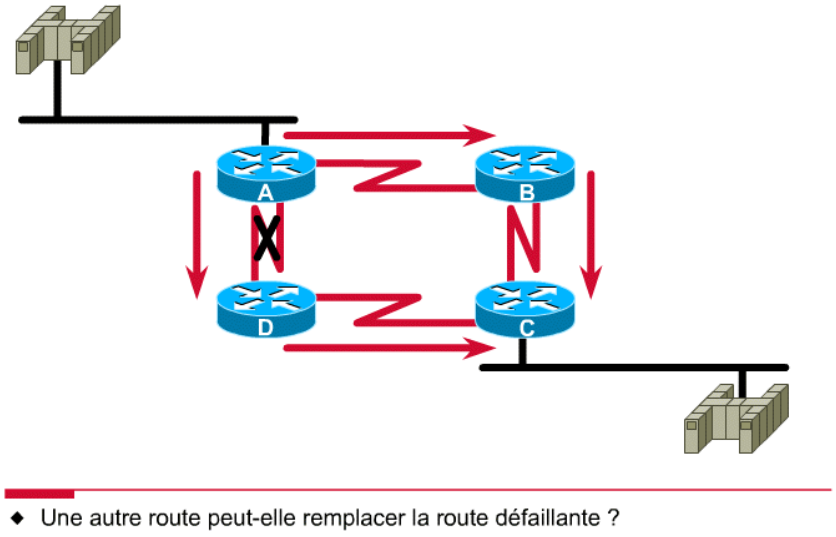
Routage dynamique



Le routage dynamique ou adaptatif permet aux routeurs de tenir compte des modifications des conditions réseau. Grâce au routage dynamique, les routeurs échangent des messages périodiques de mise à jour du routage. Si un routeur reçoit un message de mise à jour d'un autre routeur, indiquant un changement sur le réseau, il calcule une nouvelle route et inclut les nouvelles informations dans le message de mise à jour suivant qu'il envoie. Les protocoles de routage tels que RIP et IGRP facilitent le routage dynamique.

=> Nécessité pour les grands réseaux

Routage dynamique

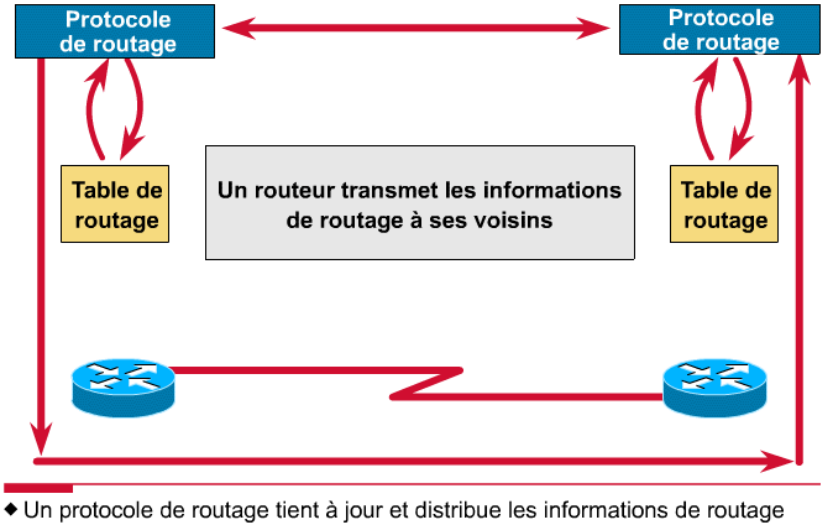


Adaptation aux modifications topologiques

Fonctions de base :

- Gestion d'une table de routage
- Distribution des informations aux autres routeurs pour les mises à jour de routage

Routage dynamique



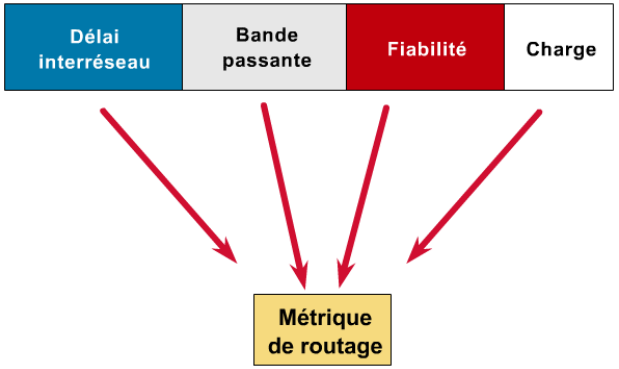
Ex. d'actions définies
par le protocole de routage :

Informations des MaJ :
Que doivent-elles contenir ?

Quand envoyer MaJ :
A quel instant ?
Ou suite à quel événement ?

Destinataires des MaJ :
A qui ?

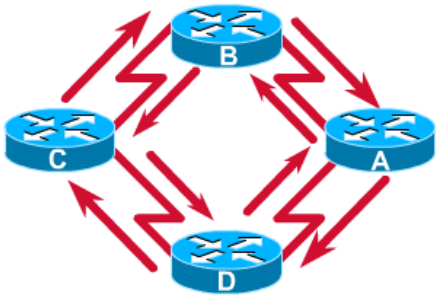
Métriques



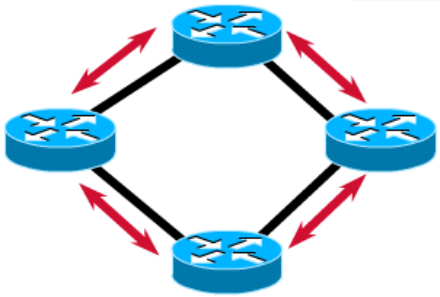
- **Bande passante** :
  Débit d'1 liaison [bit/s]
- **Délai** :
  Temps requis pour acheminer paquet de la source à la destination
- **Charge** :
  Quantité de trafic sur 1 ressource (routeur ou lien) du réseau
- **Fiabilité** :
  Ex. : taux d'erreurs sur les liaisons du réseau
- **Nombre de sauts** :
  Nombre de routeurs pour arriver à destination
- **Tics** :
  Intervalle de temps estimé sur la liaison
- **Coût** :
  Valeur arbitraire (basée sur la BP, un coût monétaire, etc)
  attribuée à un lien par l'administrateur

Classes

Vecteur de distance



Routage hybride



État de lien

3 classes :

- **A vecteur de distance** :
  Routeurs échangent directement sur les « distances » les séparant (ex. : RIP & nombre de sauts)
- **A état de lien** :
  Routeurs échangent sur l' « état des liaisons » les séparant
  => à partir de ces informations, chacun calculera les distances
- **Hybride**

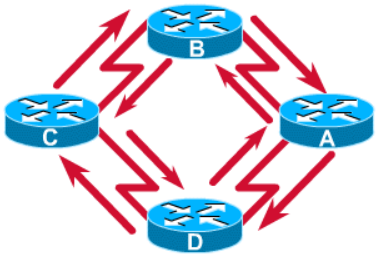
Temps de convergence

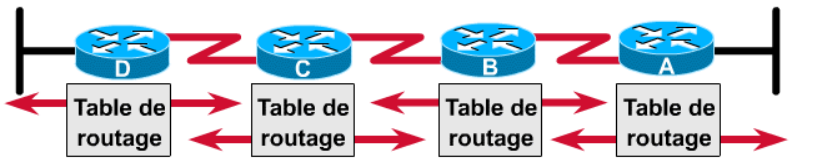
Inter-réseau est *convergent* :

Quand tous les routeurs « voient »
les mêmes informations

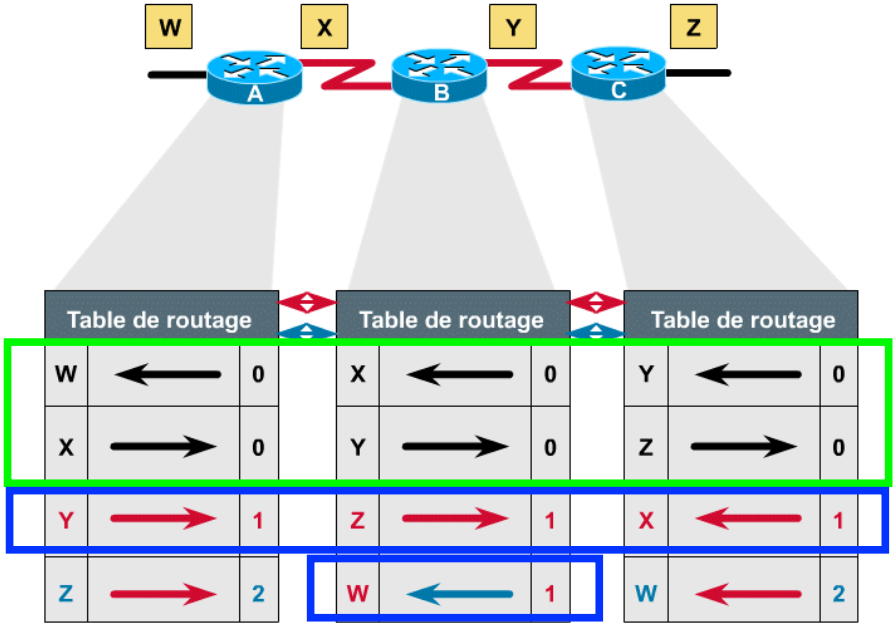
=> Une convergence rapide (après chaque MaJ)
est nécessaire

Convergence avec routage à vecteur de distance





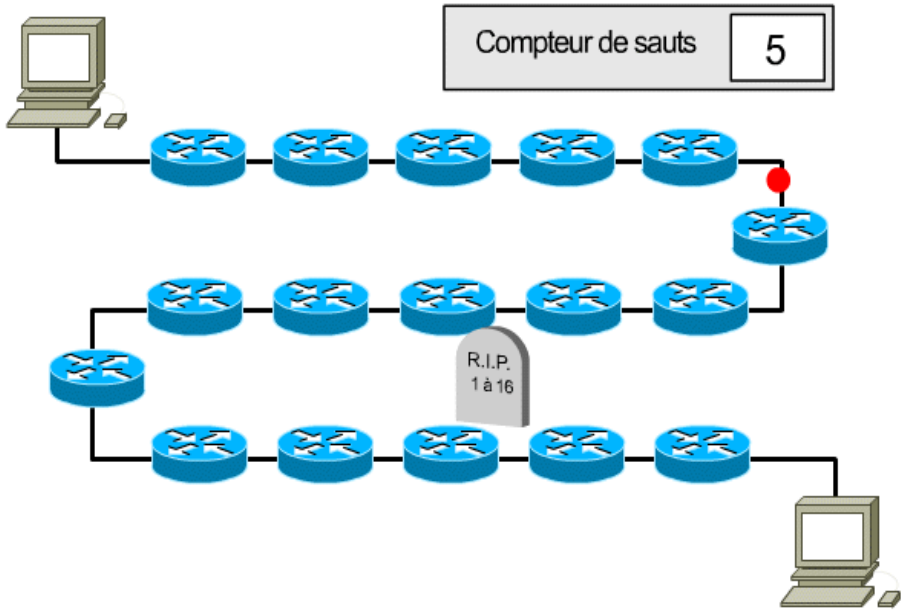
◆ Un routeur transmet périodiquement des copies de sa table de routage aux routeurs voisins et cumule les vecteurs de distance



1. Initialisation de chaque routeur :
   Configuration avec identification des voisins (adjacents)
2. Puis découverte du réseau
   => pas de vue de la topologie entière
   Convergence au bout de plusieurs itérations

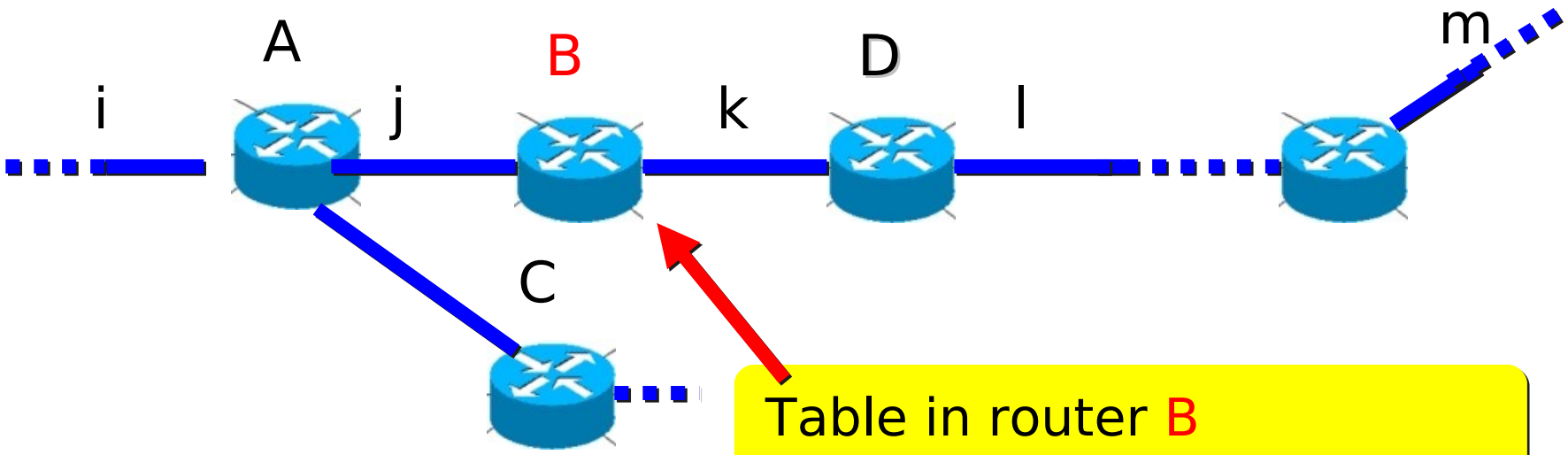
Routage dynamique

« Routing Information Protocol »



- **Ex. RIP :**
  - Le plus ancien
  - Usage : réseau local
  - Distance entre 2 hôtes,
    Métrique : « nombre de sauts »
    Limite : max. 15 sauts
  - Mise à jour des tables
    typ. toutes les 30s
    => systématique et périodique (coût : accroissement du trafic)
  - Meilleur chemin => celui comptant le moins de sauts
    « **vecteur de distance** » (mais pas toujours le plus rapide)
  - Un message RIP contient jusqu'à 25 réseaux destination

Ex. : RIP



destination network	next router	hops to destination
i	A	1
j	-	0
l	D	1
m	D	7

Ex. : RIP

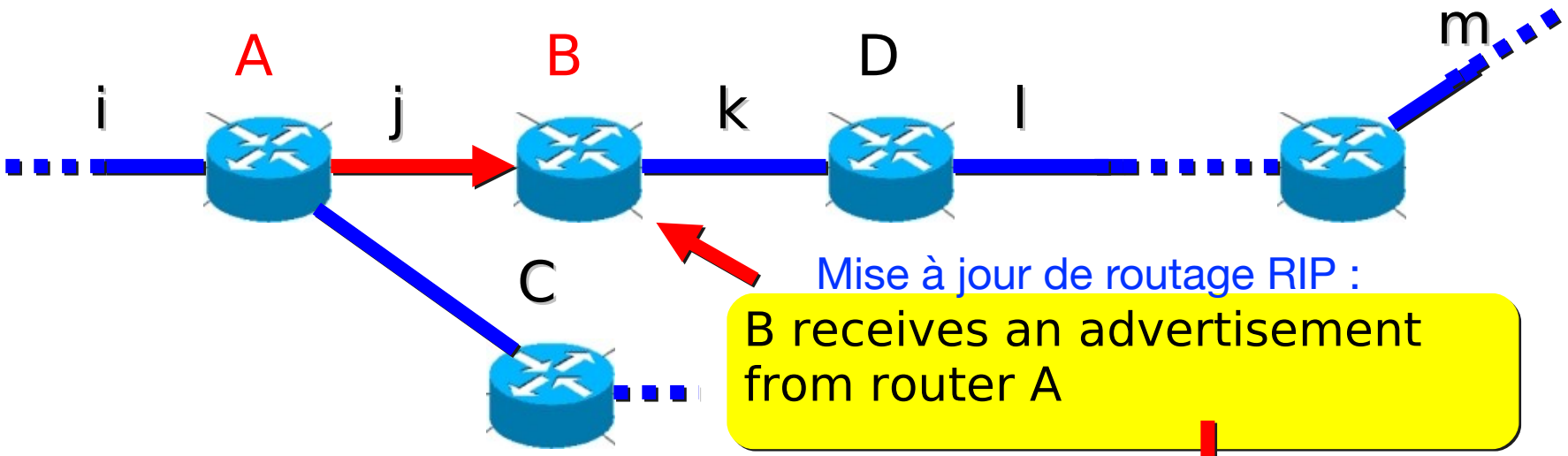
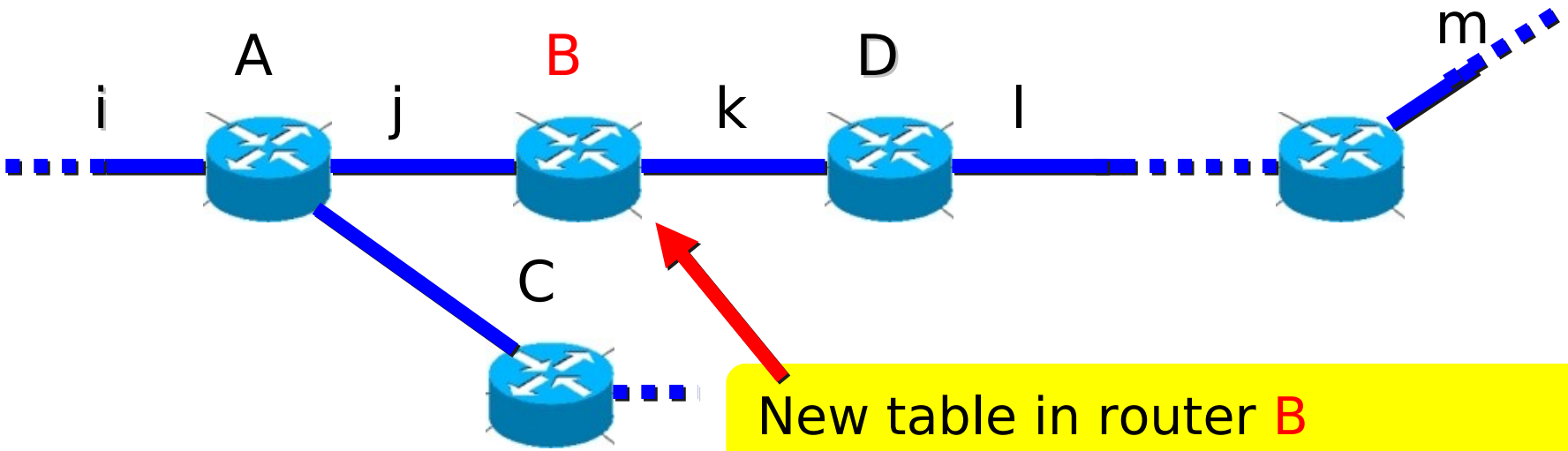


Table de routage de A

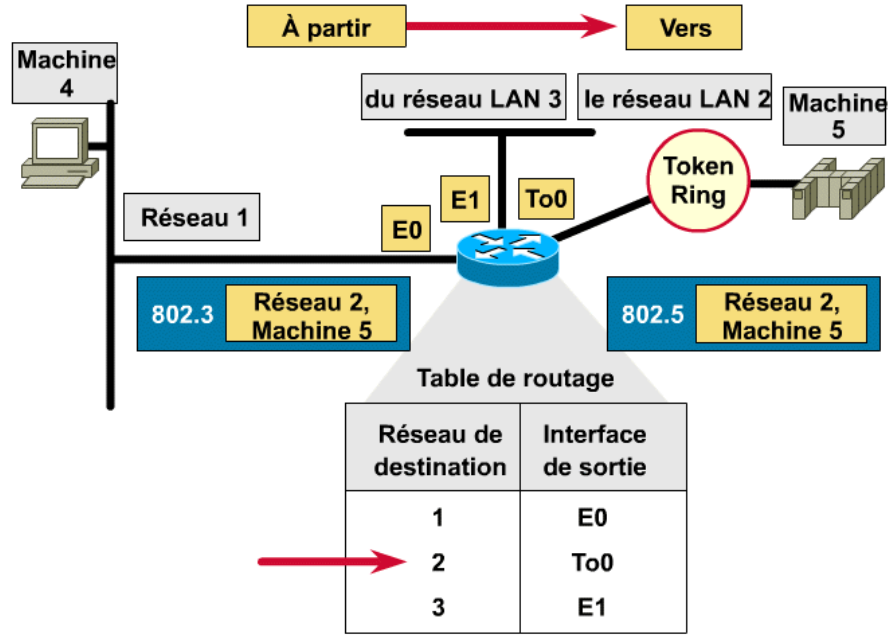
destination network	next router	hops to destination
i	-	0
j	-	0
m	C	3

Ex. : RIP



destination network	next router	hops to destination
i	A	1
j	-	0
l	D	1
m	A	4

Routage entre LAN

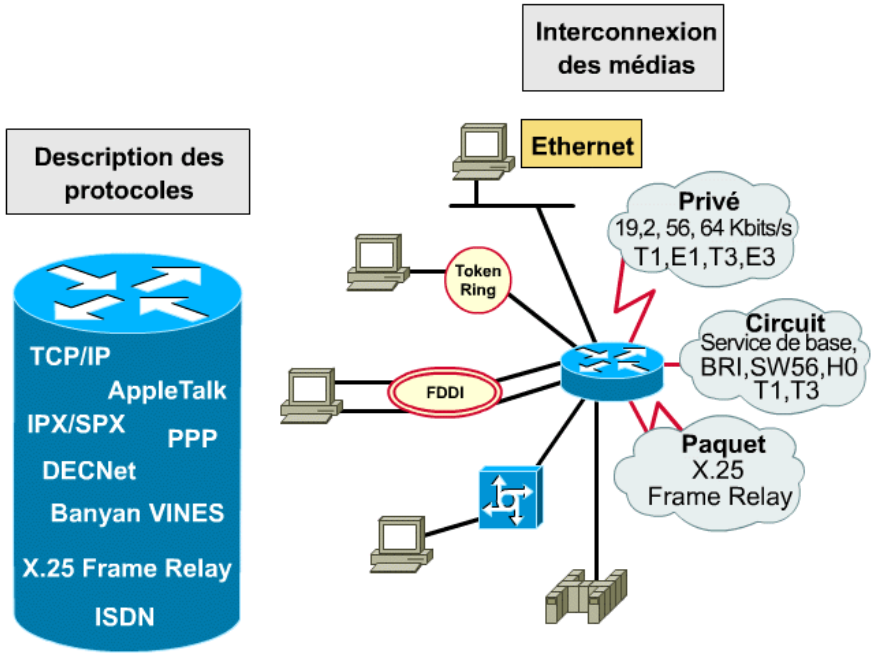


→ couche réseau doit communiquer avec couches inférieures

=> routeur doit être capables d'encapsuler les paquets dans différentes trames (différentes technologies de niveau 2)

Ex.
Paquets
de la machine 4 (réseau1, Ethernet)
vers machine 5 (réseau 2, Token Ring).

Routage entre LAN et WAN



Routeurs = « noeuds actifs et intelligents »

=> fonctions de base :
commutation & routage

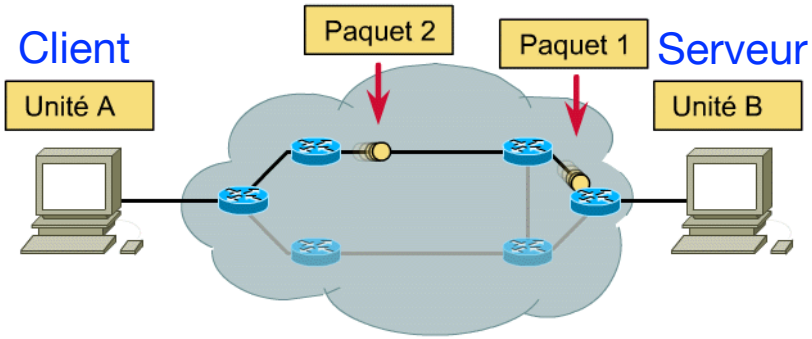
=> connectivité inter-réseaux :
éventail de liaisons, débits et protocoles (routés & de routage)

=> autres :
Gestion & contrôle du trafic (priorités, filtrage, pontage, concentration, …)
Sécurité, ...

Plan

- Bases
  - Réseaux LAN / WAN
  - Modèles
    - OSI
    - TCP/IP
    - Unités LAN et couches
- Réseaux locaux
  - Couche 1
  - Couche 2
  - Couche 3
  - Couche 4 Protocoles

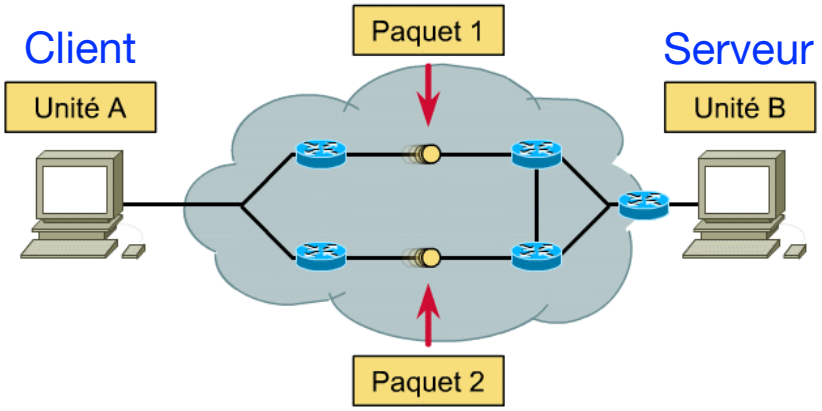
Services réseau orientés connexion



Pour une connexion fiable,
3 phases :

- **Établissement** de la connexion
  - 1 Chemin (virtuel) unique entre source et destination (commutation de circuits)
  - + Allocation de ressources (destinataire contacté avant envoi, QoS assurée)
- **Transfert** des données
  - Données transmises séquentiellement via chemin établi
  - Données arrivent dans l'ordre d'émission
- **Fermeture** de la connexion

Services réseau non orientés connexion



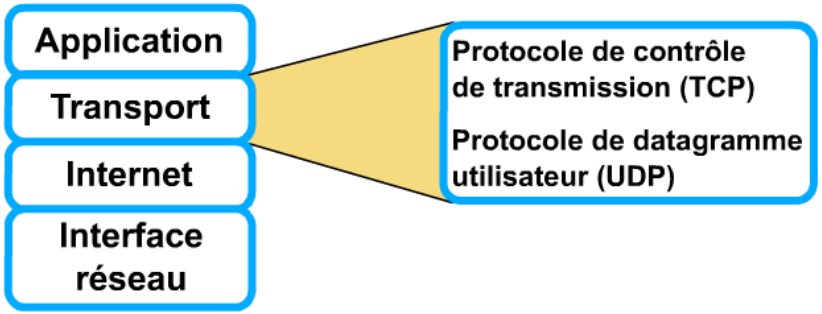
Pas de connexion :

- « commutation de paquets »
- Destinataire non contacté avant envoi
- Chaque paquet :
  - traité séparément
  - peut emprunter 1 chemin différent
    => paquets peuvent arriver dans le désordre
- Paquets doivent être ré-assemblés & ré-ordonnés à destination

- Ex. Internet est non orienté connexion
  - Pour distribution fiables des **segments**
    => Ajout de services orientés connexion par la couche « Transport » avec TCP (vérification pertes de segments, ré-ordonnancement, ...)

Couche transport

2 protocoles : **TCP** et **UDP**



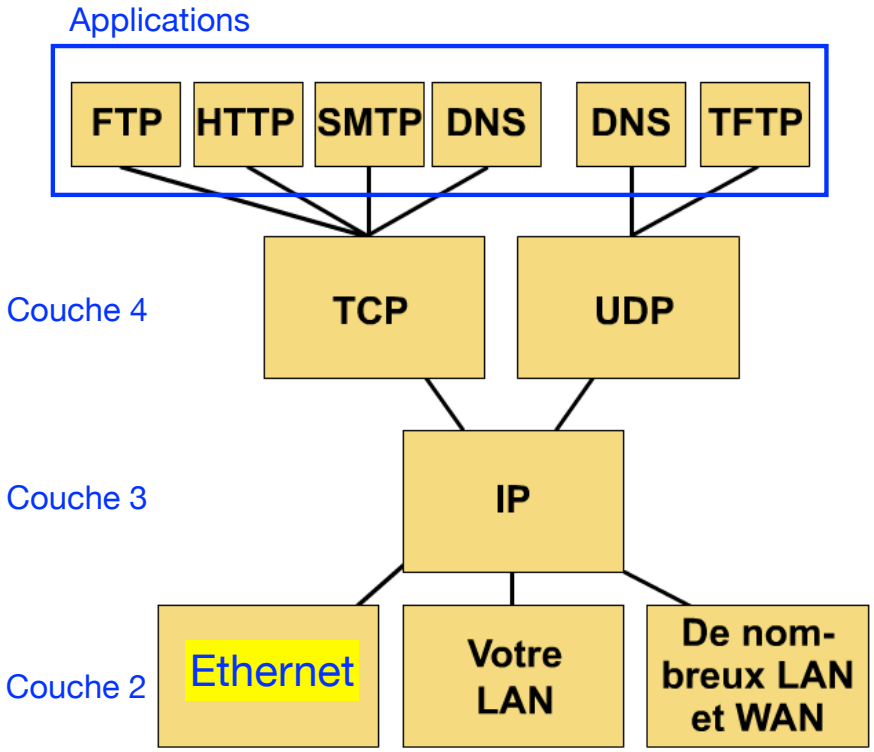
« Qualité de service » ajoutée par TCP :

- Fiabilité de l'acheminement de bout en bout (de la source à destination) des données
- Contrôle de flux

- Mécanisme de fenêtres glissantes avec n° de séquence et accusé de réception

TCP / UDP

Modèle TCP/IP



TCP : « circuit virtuel »

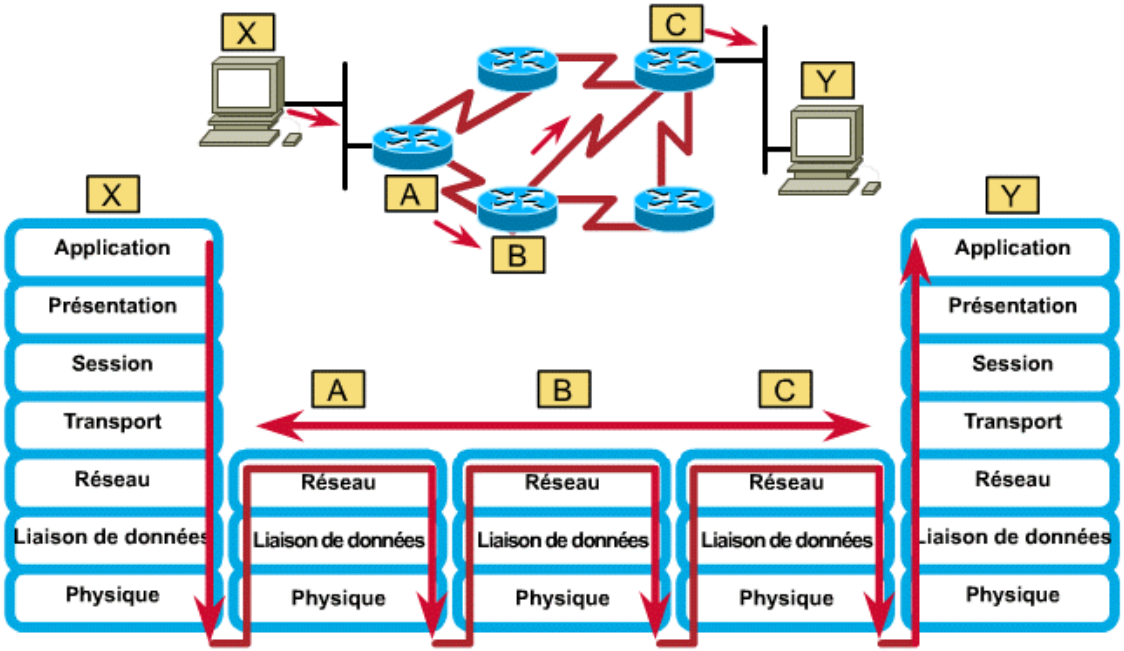
- Orienté connexion (fiabilité)
- Transmission en mode full-duplex
- Source : divise les messages en **segments**
- Destination:
  - ré-assemble les segments
  - demande les segments non reçues

UDP :

- Non orienté connexion
  (peu fiable mais rapide)
- Transmission de **datagrammes**
- Pas de vérification logicielle de la livraison des datagrammes
- Pas d'accusé de réception, ni de fenêtrage
- Pas de contrôle de flux
- => applications doivent assurer la fiabilité

Couche transport

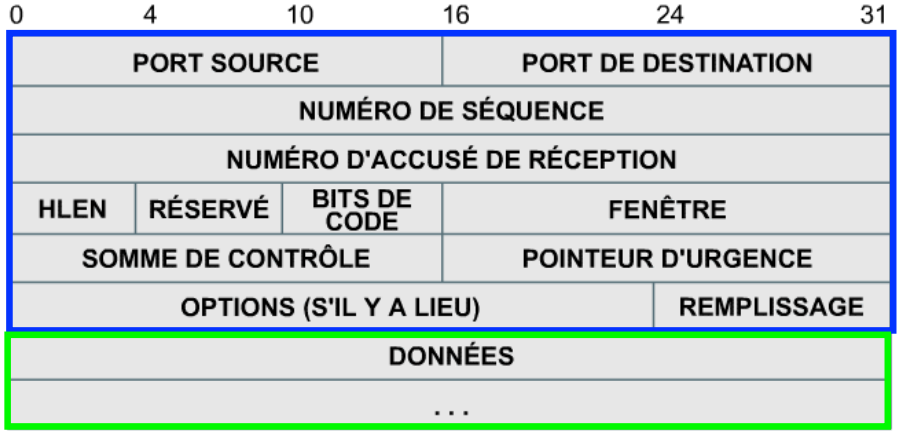
TCP : Fiabilité de l'acheminement
de bout en bout des données



- Chaque routeur fournit ses services pour la prise en charge des fonctions de la couche supérieure.

Segment TCP / Datagramme UDP

Segment TCP

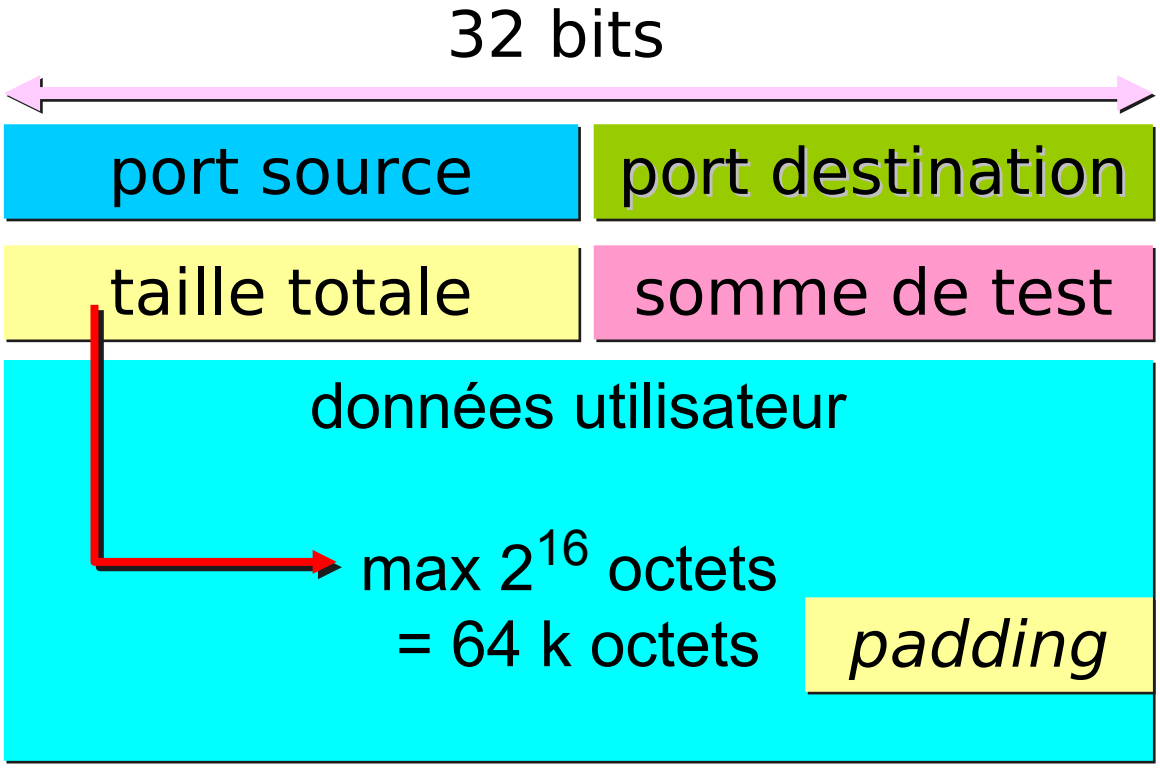


- *Port d'origine* - numéro du port appelant
- *Port de destination* - numéro du port appelé
- *Numéro de séquence* - numéro utilisé pour assurer le séquençage correct des données entrantes
- *Numéro d'accusé de réception* - prochain octet TCP attendu
- *HLEN* - nombre de mots de 32 bits contenus dans l'en-tête
- *Réservé* - défini sur zéro
- *Bits de code* - fonctions de contrôle (telles l'ouverture et la fermeture d'une session)
- *Fenêtre* - nombre d'octets que l'émetteur est prêt à accepter
- *Somme de contrôle* - somme de contrôle calculée des champs d'en-tête et de données
- *Pointeur d'urgence* - indique la fin des données urgentes
- *Option un* - taille maximale d'un segment TCP
- *Données* - données du protocole de couche supérieure

Entête Données

Datagramme UDP

Datagramme UDP



Somme de test :
- sur l'ensemble du datagramme
(+ le pseudo en-tête IP : les @IP)

- complément à 1 de la somme des mots de 16 bits

UDP – Somme de test

Exemple : somme de 3 mots de 16 bits

0110 0110 0110 0110
0101 0101 0101 0101
0000 1111 0000 1111

1100 1010 1100 1010

Complément à 1 (“checksum”) :

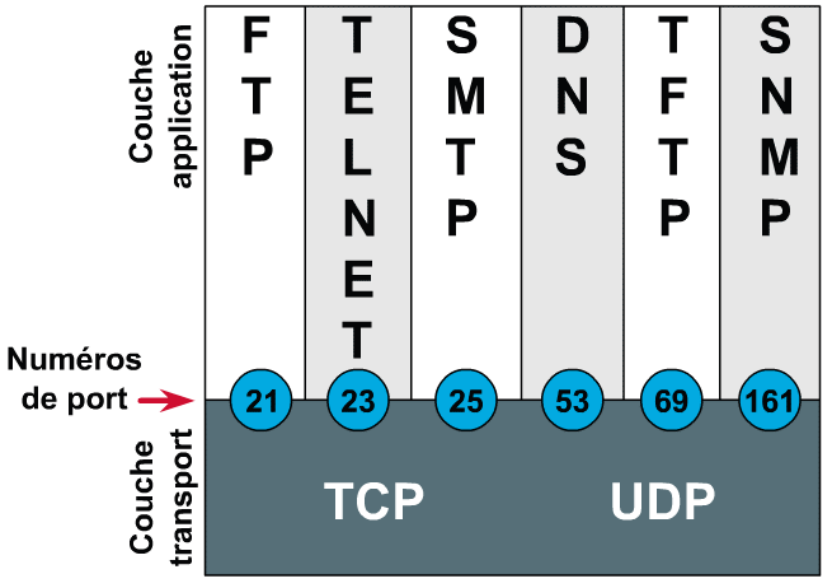
0011 0101 0011 0101

Somme de contrôle transmise

Si aucune erreur => somme au récepteur :

1111 1111 1111 1111

Notion de « Socket » (prise)



Socket :

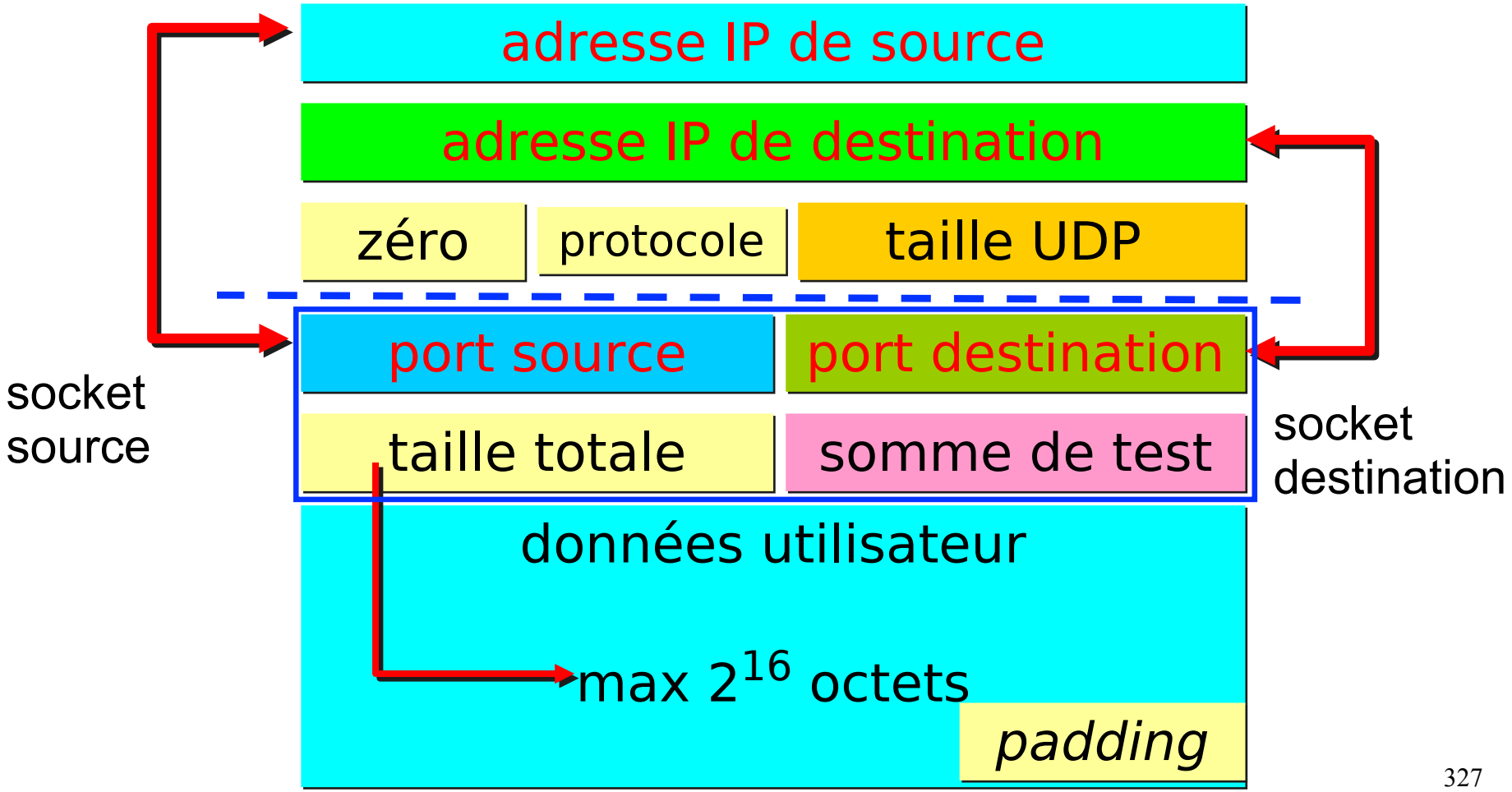
Association de 3 éléments :
@IP + protocole (UDP ou TCP) + n°port

- n°port :
  identification de l'application utilisant le service réseau
  => cf. plusieurs applications en //
- Certains n°port sont réservés

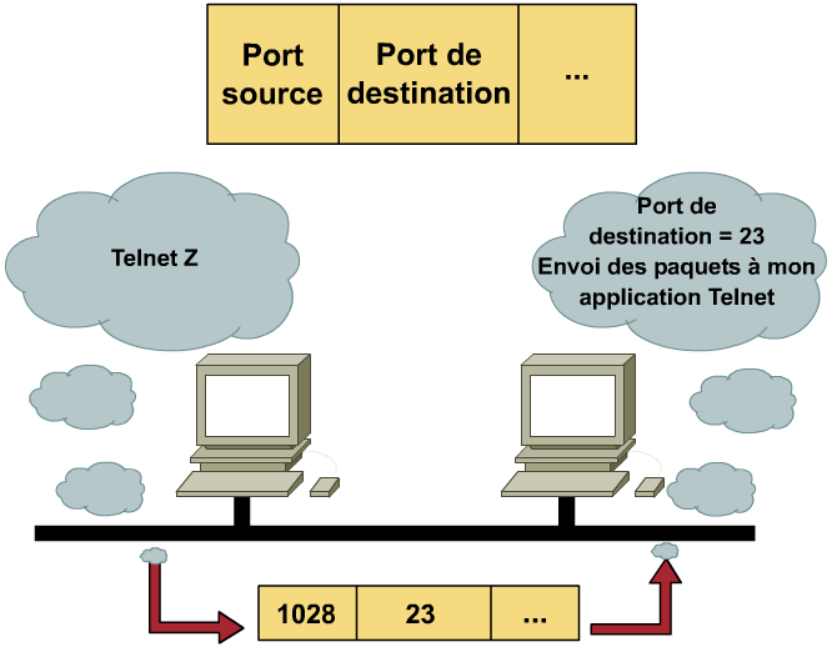
- Numéros inférieurs à 255 - réservés aux applications publiques.
- Numéros entre 255 et 1023 - attribués aux entreprises pour les applications commercialisables.
- Numéros supérieurs à 1023 - ne sont pas attribués.

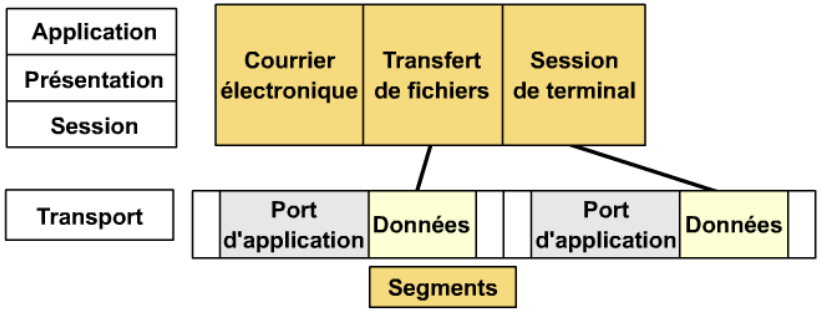
utilisés de façon dynamique

Socket et UDP (datagramme)
Pseudo entête



Socket





◆ Les segments de transport partagent le flux de trafic.

=> Plusieurs applications en //

Client

Serveur (Telnet)

n° port réservés

aux applications « bien connues »

TCP

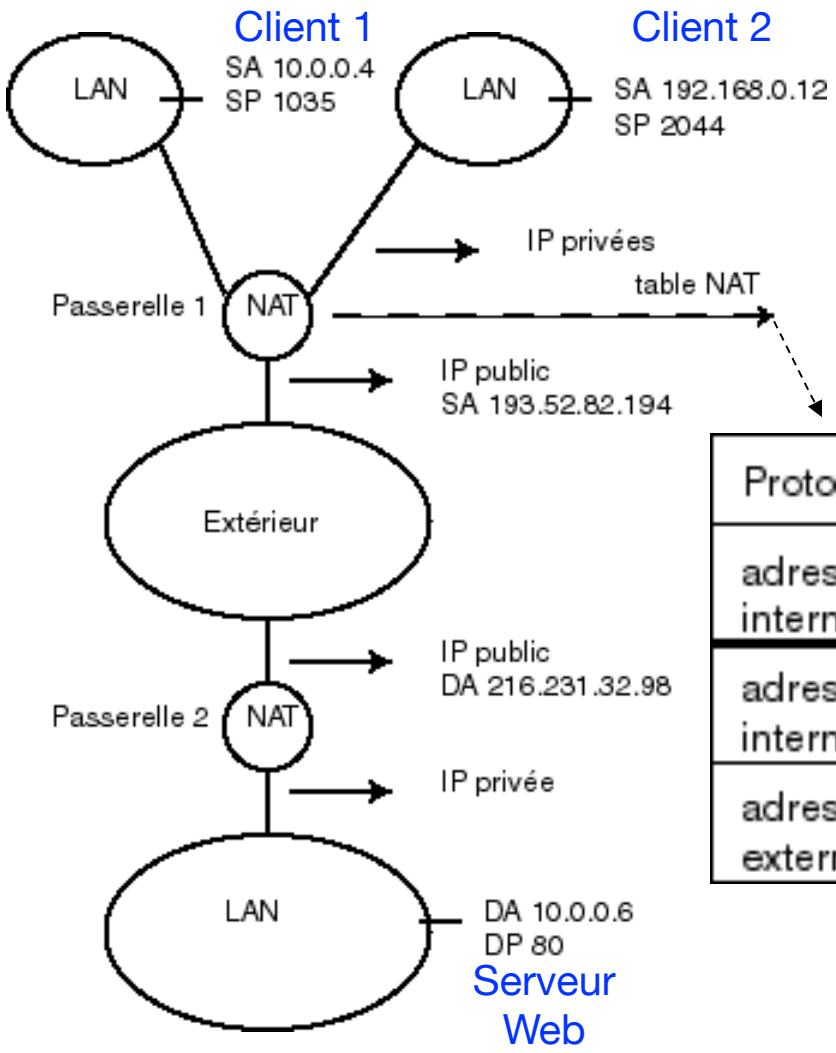
Décimal	Mot-clé	Description
0		Réservé
1-4		Non attribué
5	RJE	Soumission de travaux à distance
7	ECHO	Écho
9	DISCARD	Abandon
11	USERS	Utilisateurs actifs
13	DAYTIME	Heure du jour
15	NETSTAT	Qui est actif ou NETSTAT
17	QUOTE	Citation du jour
19	CHARGEN	Générateur de caractères
20	FTP-DATA	Protocole FTP (données)
21	FTP	Protocole FTP
23	TELNET	Connexion en mode terminal
25	SMTP	Protocole SMTP
37	TIME	Heure du jour
39	RLP	Protocole RLP
42	NAMESERVER	Serveur de noms d'hôte
43	NICNAME	Qui est
53	DOMAIN	Serveur de noms de domaine
67	BOOTPS	Serveur de protocole Bootstrap
68	BOOTPC	Client de protocole Bootstrap
69	TFTP	Protocole TFTP
75		Tout service de sortie privé
77		Tout service RJE privé
79	FINGER	Finger
80	HTTP	Protocole HTTP
95	SUPDUP	Protocole SUPDUP
101	HOSTNAME	Serveur de noms d'hôte NIC
102	ISO-TSAP	ISO-TSAP
113	AUTH	Service d'authentification
117	UUCP-PATH	Service de chemin UUCP
123	NTP	Protocole NTP
133-159		Non attribué
160-223		Réservé
224-241		Non attribué
242-255		Non attribué

UDP

Décimal	Mot-clé	Description
0		Réservé
1-4		Non attribué
5	RJE	Soumission de travaux à distance
7	ECHO	Écho
9	DISCARD	Abandon
11	USERS	Utilisateurs actifs
13	DAYTIME	Heure du jour
15	NETSTAT	Qui est actif ou NETSTAT
17	QUOTE	Citation du jour
19	CHARGEN	Générateur de caractères
20	FTP-DATA	Protocole FTP (données)
21	FTP	Protocole FTP
23	TELNET	Connexion en mode terminal
25	SMTP	Protocole SMTP
37	TIME	Heure du jour
39	RLP	Protocole RLP
42	NAMESERVER	Serveur de noms d'hôte
43	NICNAME	Qui est
53	DOMAIN	Serveur de noms de domaine
67	BOOTPS	Serveur de protocole Bootstrap
68	BOOTPC	Client de protocole Bootstrap
69	TFTP	Protocole TFTP
75		Tout service de sortie privé
77		Tout service RJE privé
79	FINGER	Finger
123	NTP	Protocole NTP
133-159		Non attribué
160-223		Réservé
224-241		Non attribué
242-255		Non attribué

Cf. linux : /etc/services

NAT overloading - IP masquerading



nbre(@IP_internes_locales utilisées simultanément **) >**
nbre(@IP_internes_globales **)**

=> table NAT « étendue »

Hyp. :
2 clients font une requête au même serveur web.

Protocole	tcp	tcp
adresse.port interne & local	10.0.0.4.1035	192.168.0.12.2044
adresse.port interne & global	193.52.82.194.1035	193.52.82.194.2044
adresse.port externe & global	216.231.32.98.80	216.231.32.98.80

Serveur NAT peut orienter les réponses du serveur web.

TCP

- TCP & UDP => le même protocole réseau (IP)
- Mais TCP

=> service « amélioré » pour la couche applicative :

« un flux fiable d'octets avec connexion »

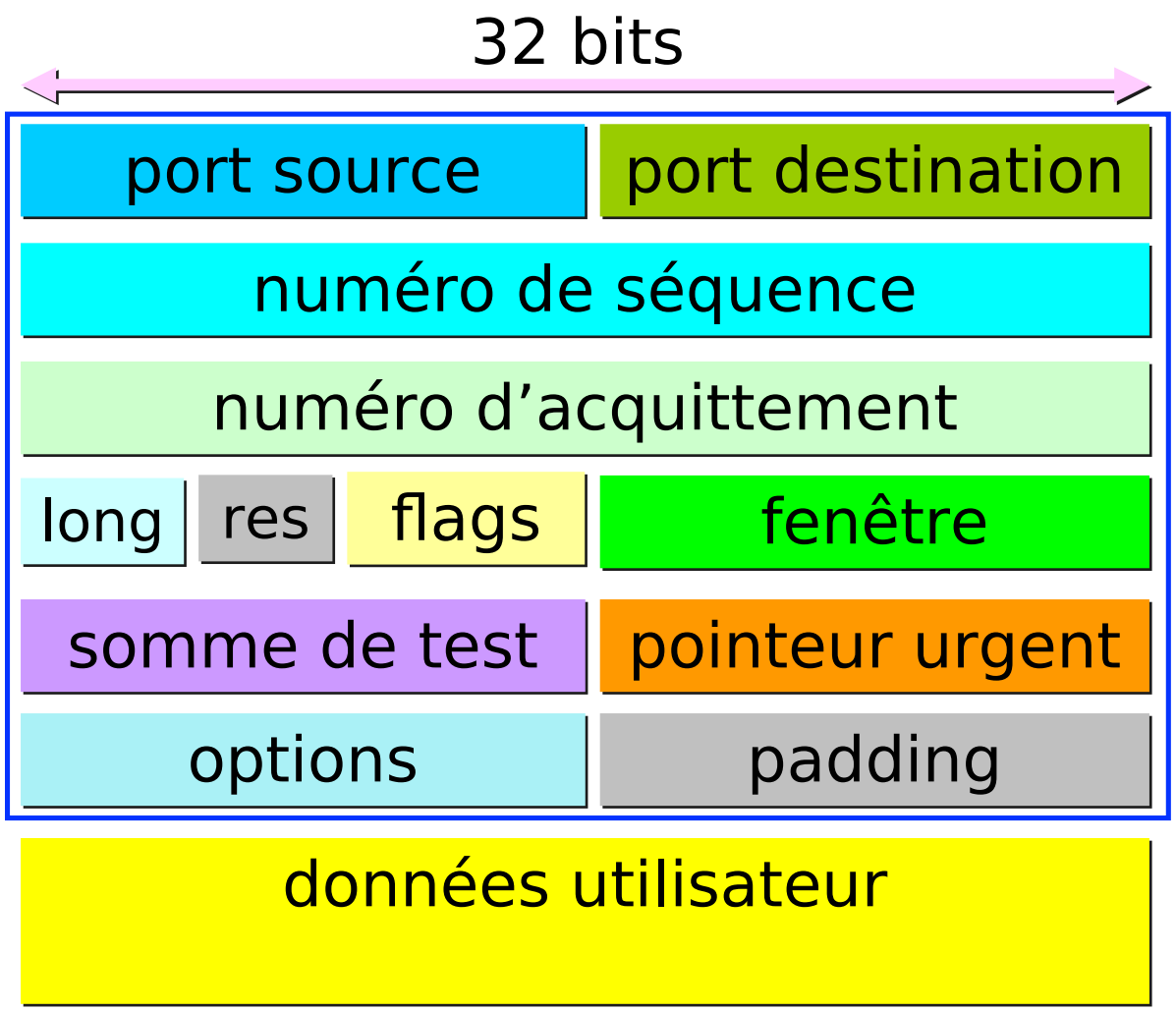
- "Orienté connexion" => les 2 correspondants établissent une connexion logique avant d' échanger des données

TCP

Fiabilité => via des mécanismes :

- Données découpées : segments
- Détection d'erreurs : somme de contrôle
- Retransmission : *time-out* adaptatif
- Re-séquencement : numéro de segment
- Contrôle de flux : fenêtre d'émission
- Acquittements positifs des données (segments) reçues
  - Avec retransmission des segments non acquittés après un délai « *time-out* »

Socket & Segment TCP



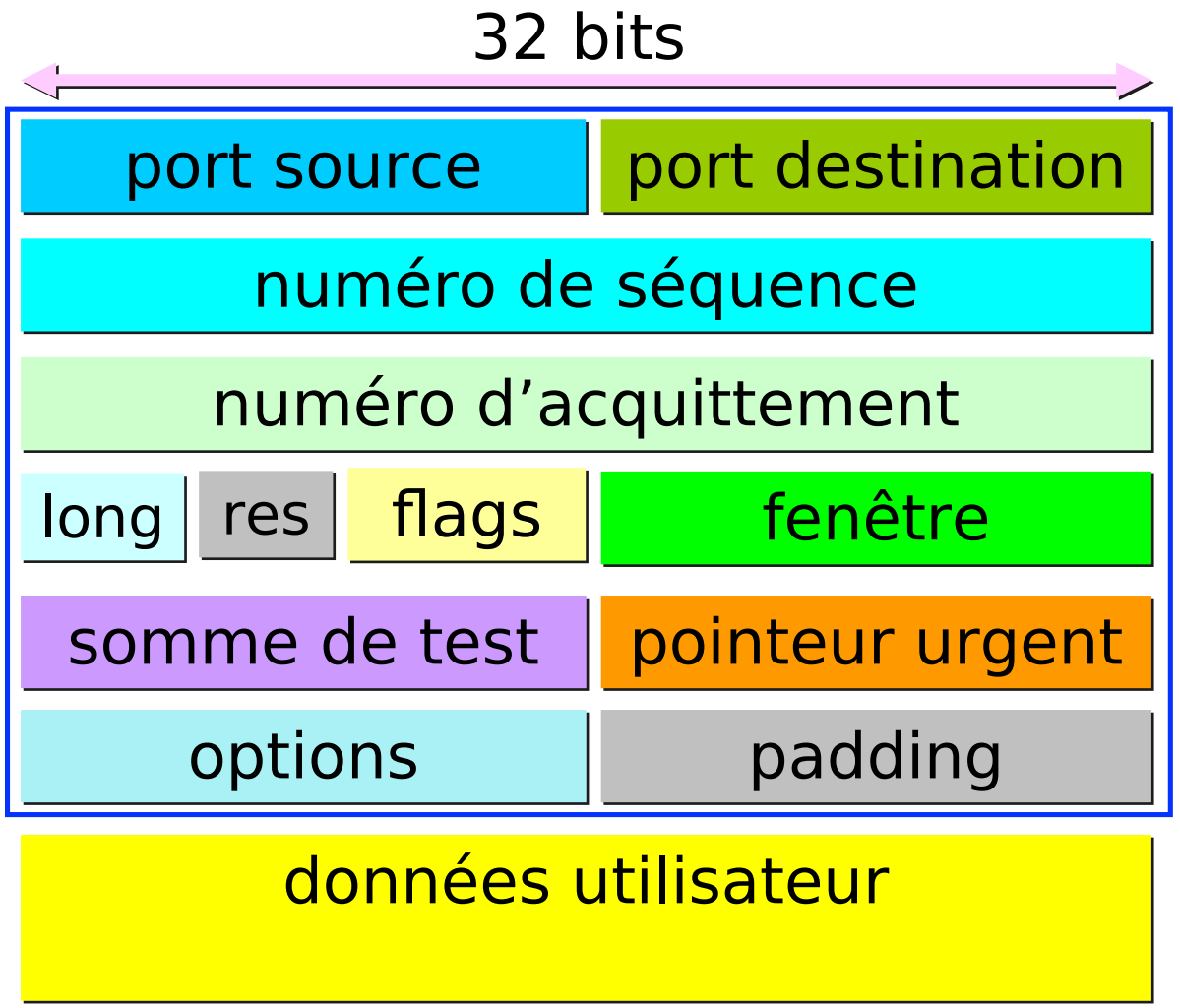
Pseudo-entête IP :

n°port source
et n°port destination
=> identification des applications

+

@IP_source et @IP_destination
=> identification applications & stations
=> Sockets

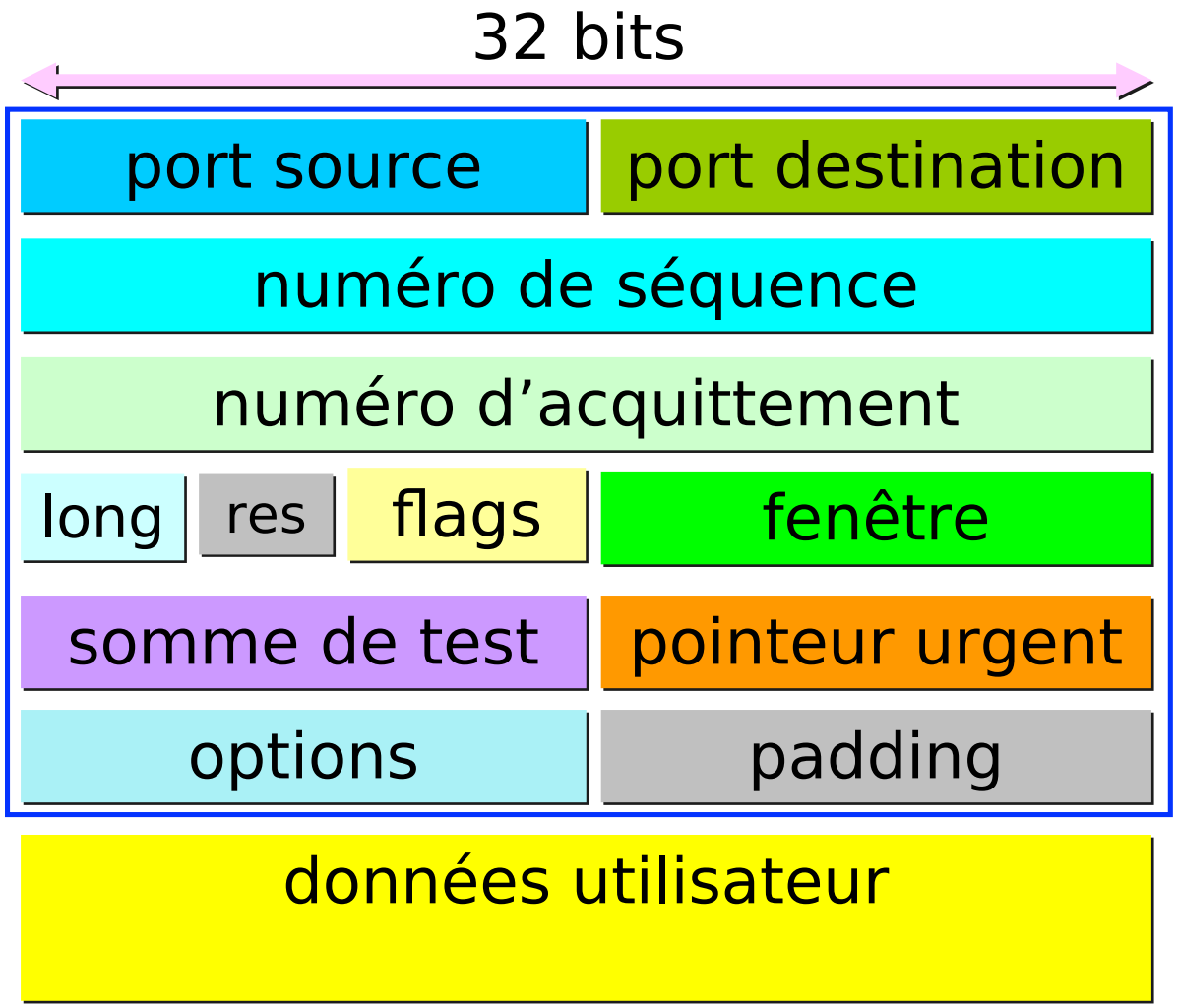
Segment TCP



n° de séquence :
n° du segment émis

n° d'acquittement :
acquittement du segment reçu
=> n° du prochain segment attendu

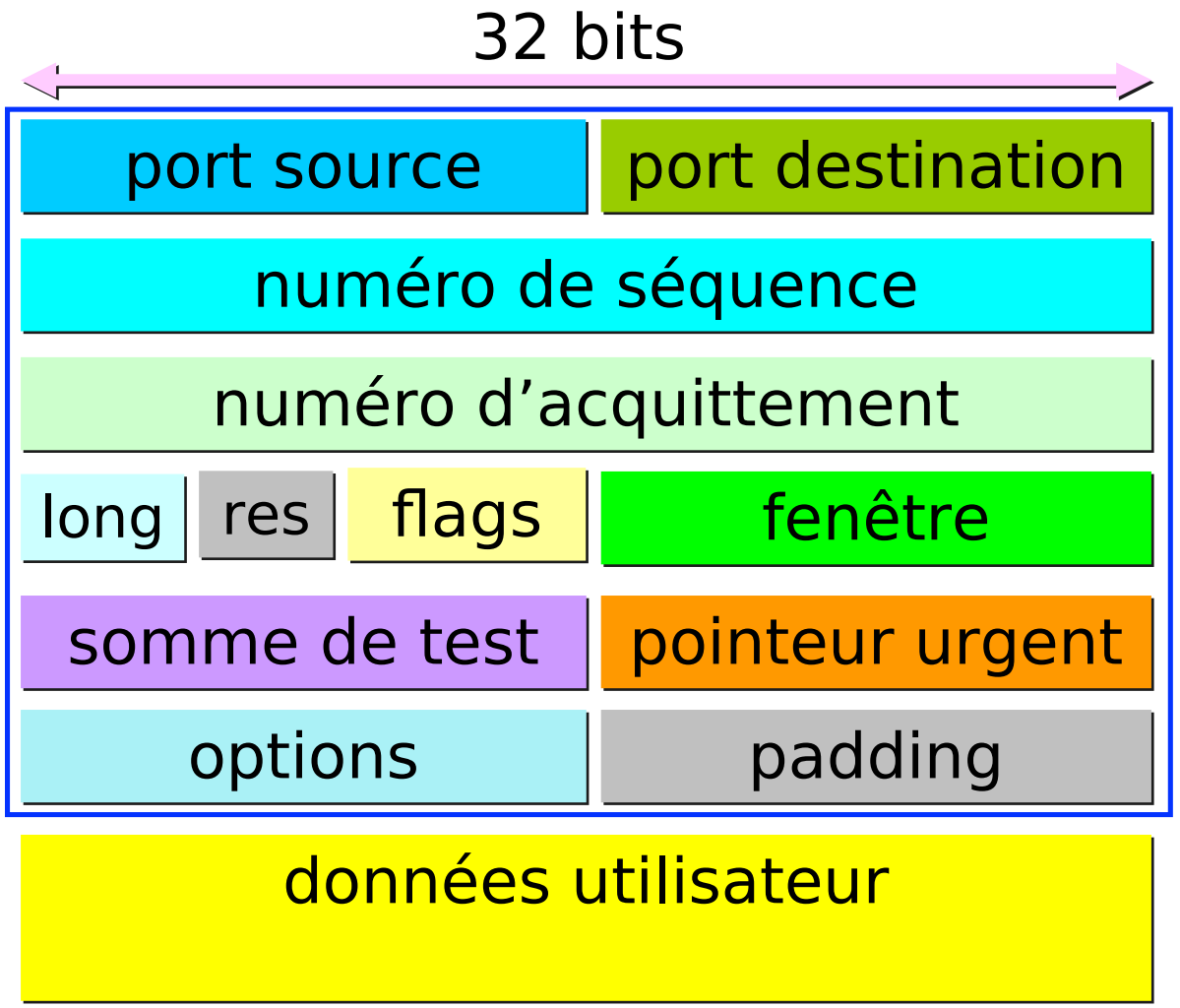
Segment TCP



long (4 bits) :
longueur de
l'en-tête
en mots de 32 bits

flags :
codes de contrôle
(annonce une
phase ou un
contenu)

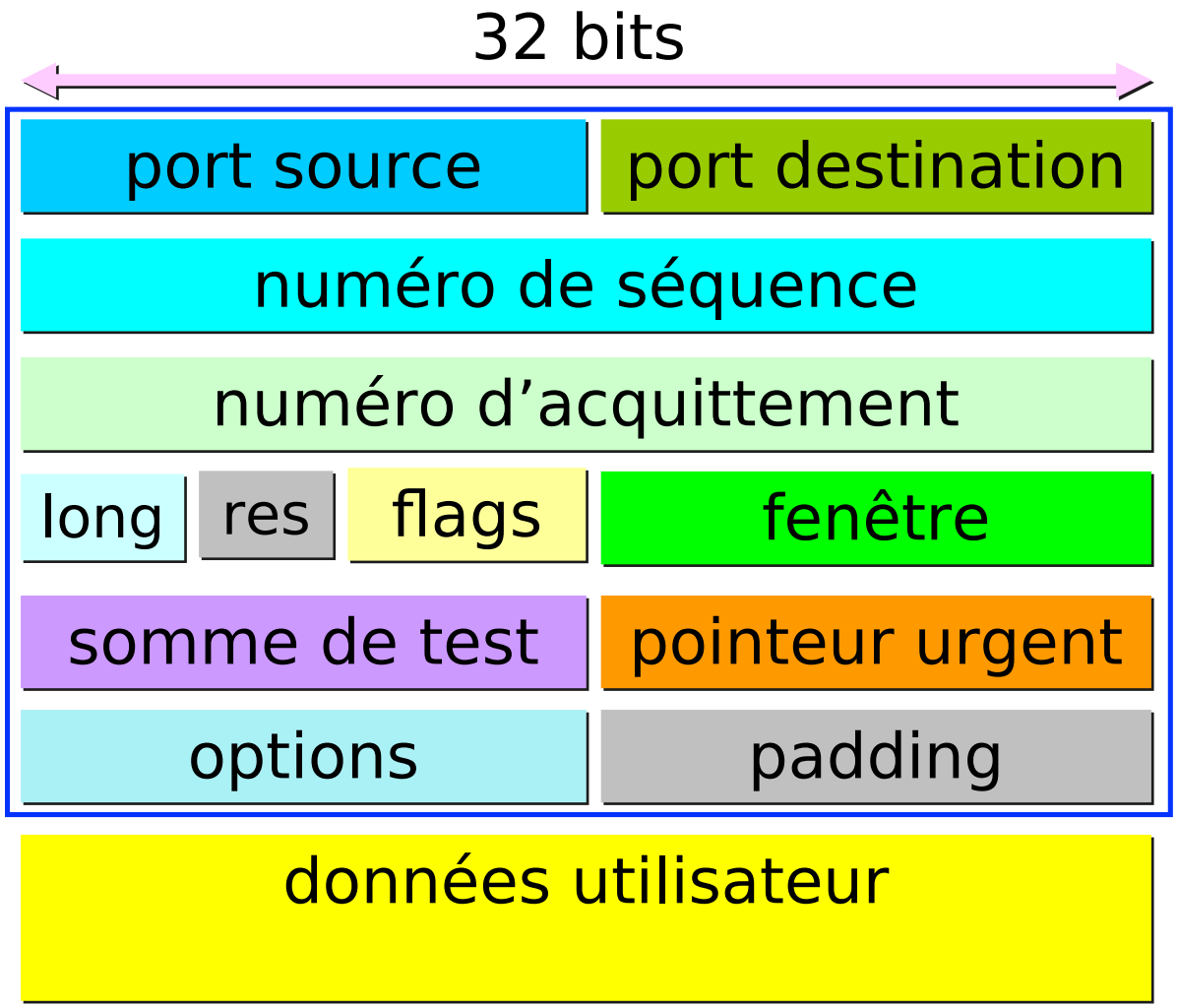
Segment TCP



fenêtre :
pour le contrôle de flux

Taille des données (fenêtre) que le récepteur souhaite recevoir sans AR (fenêtre s'ajuste dynamiquement)

Segment TCP



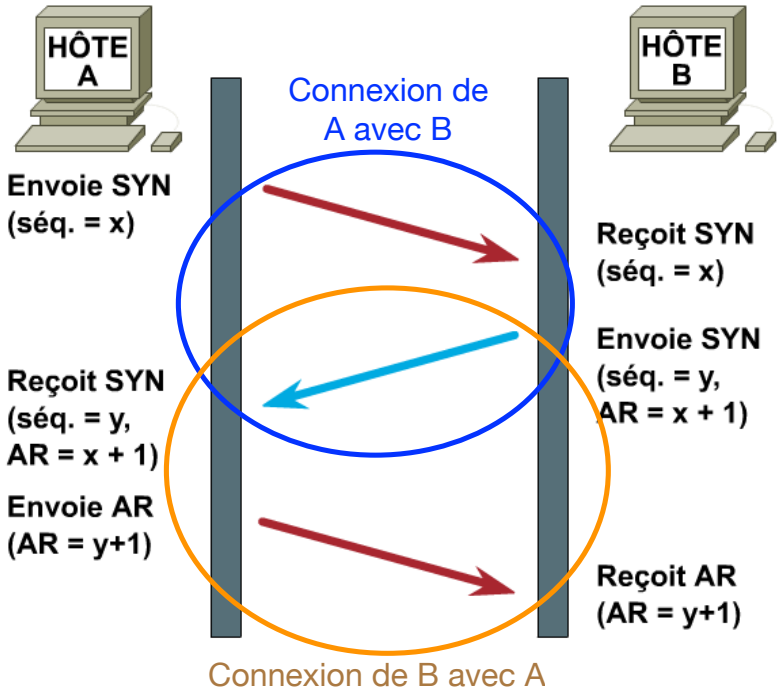
somme de test :
de l'en-tête et des données
(comme pour UDP)

pointeur urgent :
(est validé par le flag URG)
présence de données urgentes

Segment TCP – les « flags »

URG	Segment présente de données urgentes
ACK	Segment contient un accusé de réception
PSH	Segment doit être envoyé immédiatement
RST	Ré-initialisation de la connexion
SYN	Demande de synchronisation ou initialisation d'une connexion
FIN	Demande de fin de connexion

TCP - Établissement



Protocole d'échange en
« **3 étapes** » ou
« à connexion ouverte »

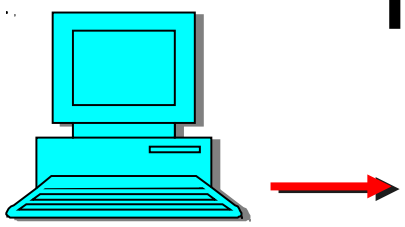
➔ Synchronisation de la connexion avant transfert

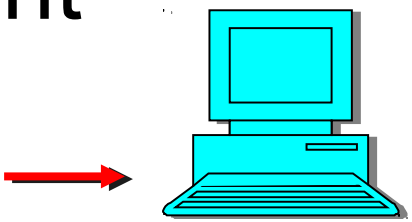
→ Échange des n° de séquences

- A demande la connexion
- « x » n°séquence initial de A
- « y » n°séquence initial de B
- **AR prévisionnel** « en avant » : indique le n° du prochain segment attendu

n° du segment = n° du 1er octet qu'il contient
(OU) = tiré au hasard

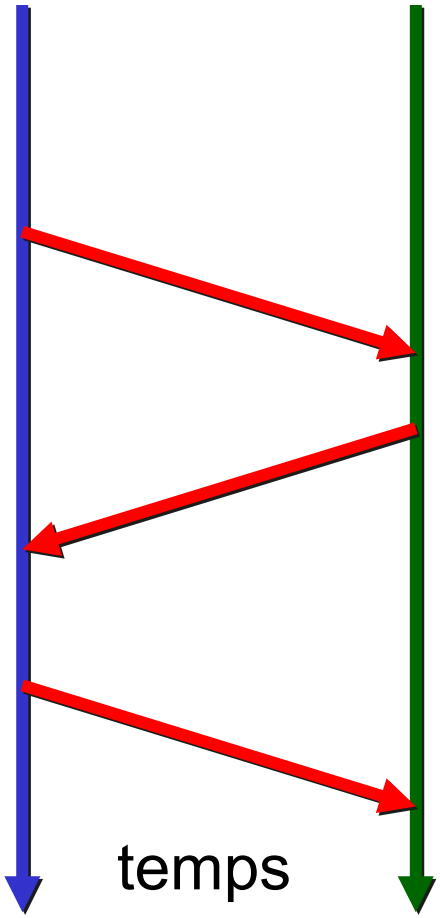
TCP - Etablissement





émetteur :
ouverture active
@IP=193.83.51.10

call_request
ISN=5670
Initial Sequence Number



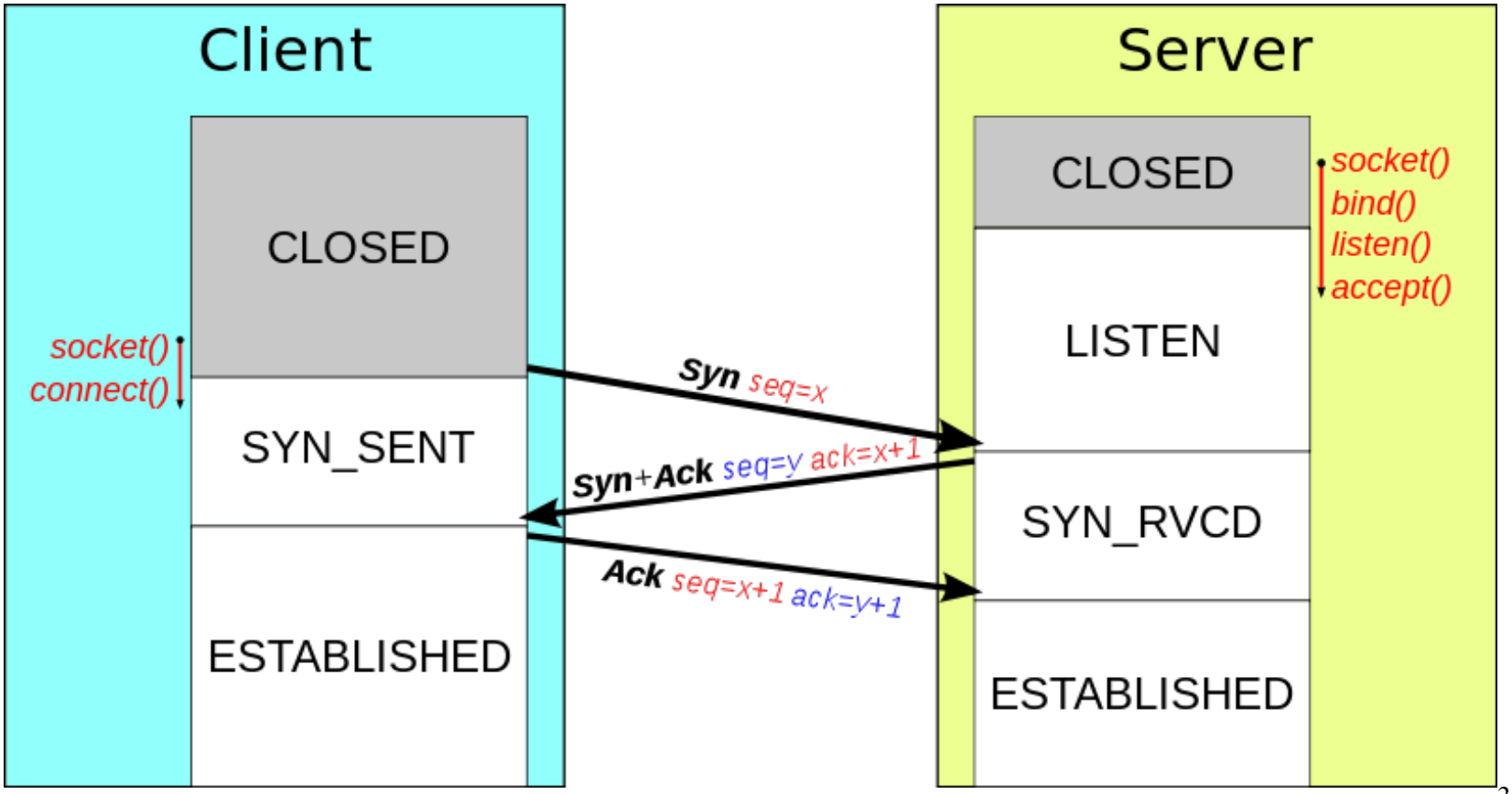
récepteur :
ouverture passive
@IP=193.83.56.82

call_request
ISN=7820
ACK=5671

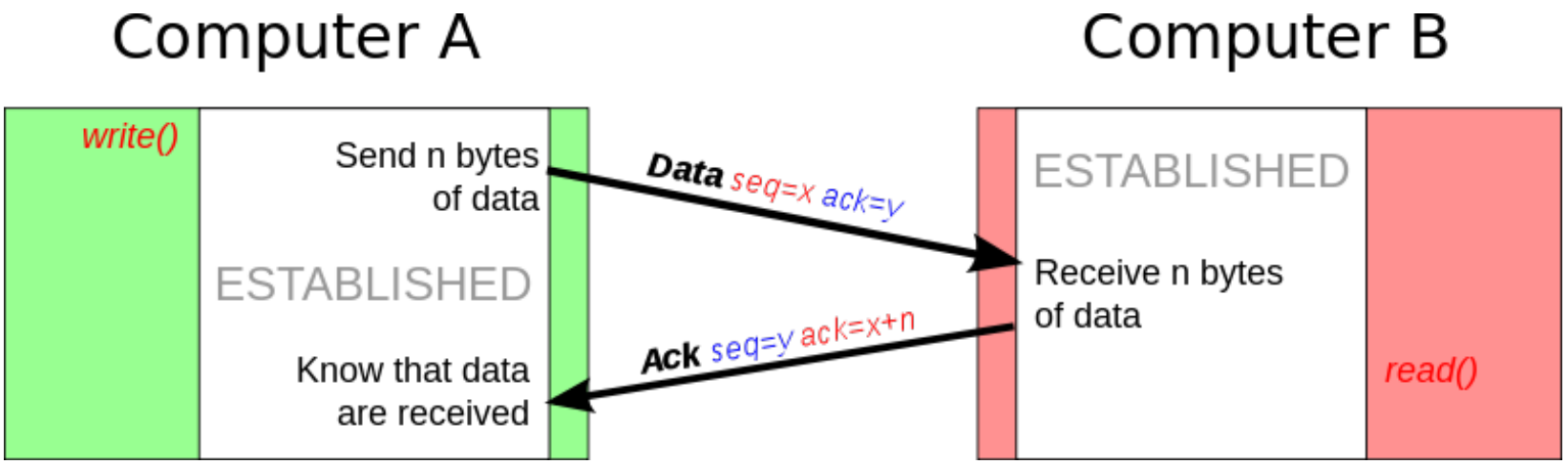
Connecté
envoi de
ACK=7821

connecté

TCP - établissement



TCP – transfert



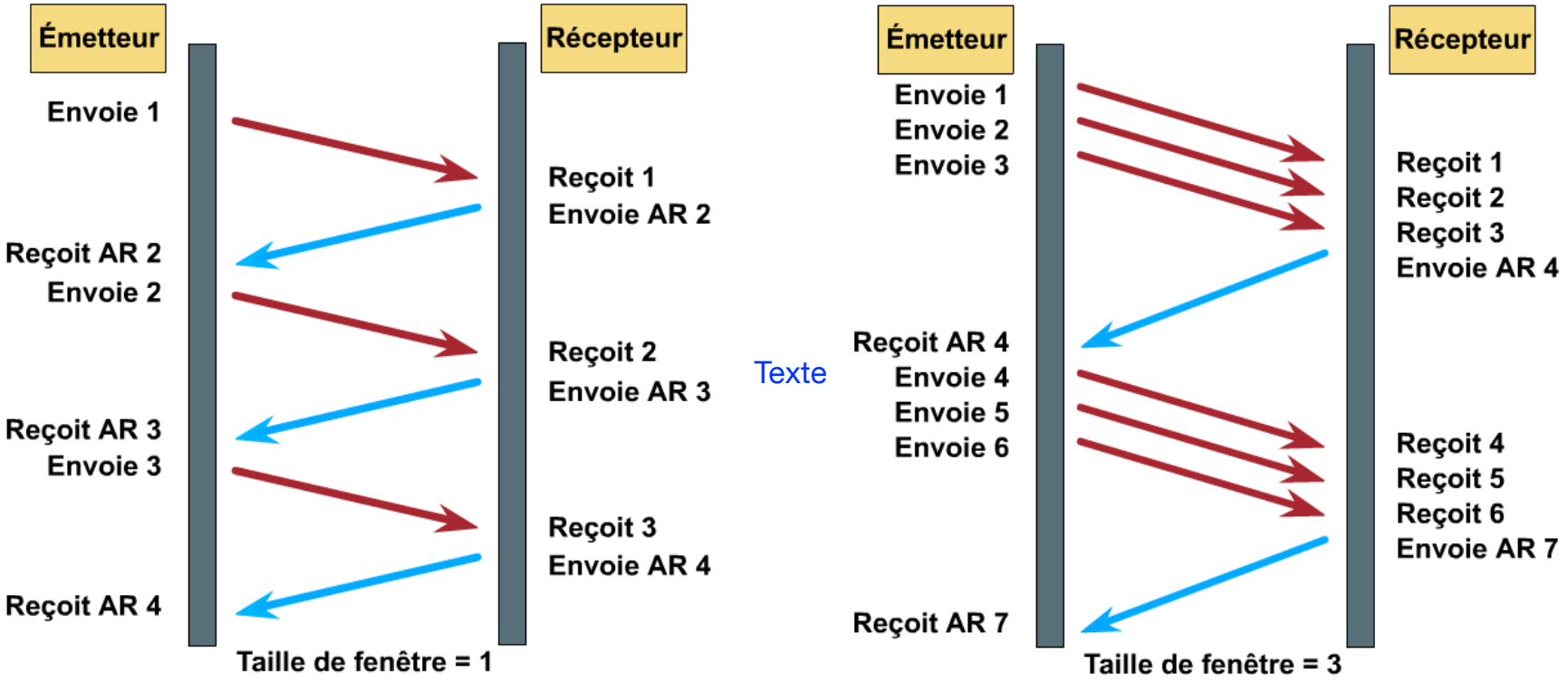
Transfert

Technique **PAR** (fiabilité) :

- « Positive Acknowledgement and Retransmission »
  Accusé de réception positif et retransmission

- Source :
  - Envoie un segment
    + Déclenche un compteur
  - Attend l' AR (accusé de réception)
    avant d'envoyer segment suivant
    - Si compteur arrive à expiration :
      - Retransmet le segment
      - Redémarre le compteur

Fenêtrage – Contrôle de flux



Coté Emetteur

- Taille de fenêtre : nombre de segments émis avant réception d'un AR
- Fenêtre « **glissante** » : taille s'adapte dynamiquement

=> utilisation efficace de BP

Taille de fenêtre contrôlée (indirectement) par le Récepteur

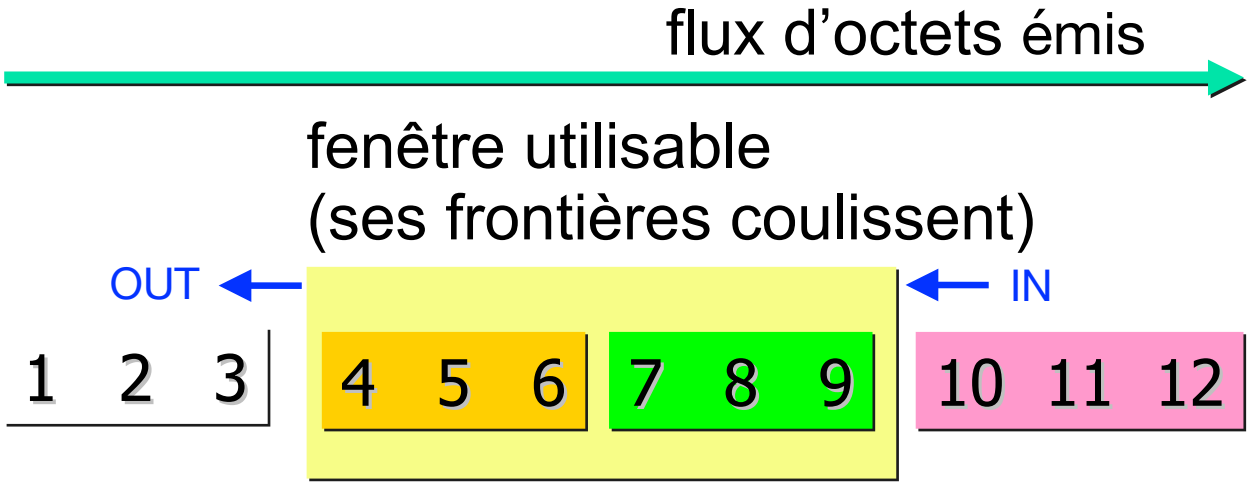
TCP – Contrôle de flux

Contrôle via une fenêtre glissante.

Taille de la fenêtre donnée par le récepteur (indirectement)

(par défaut : 4096 octets)

Ex. coté émetteur :



segments émis et acquittés

segments envoyés et non acquittés

segments prêts à être envoyés

TCP – Time-out

Calculé par approximation du temps aller-retour (RTT – *Round Trip Time*).

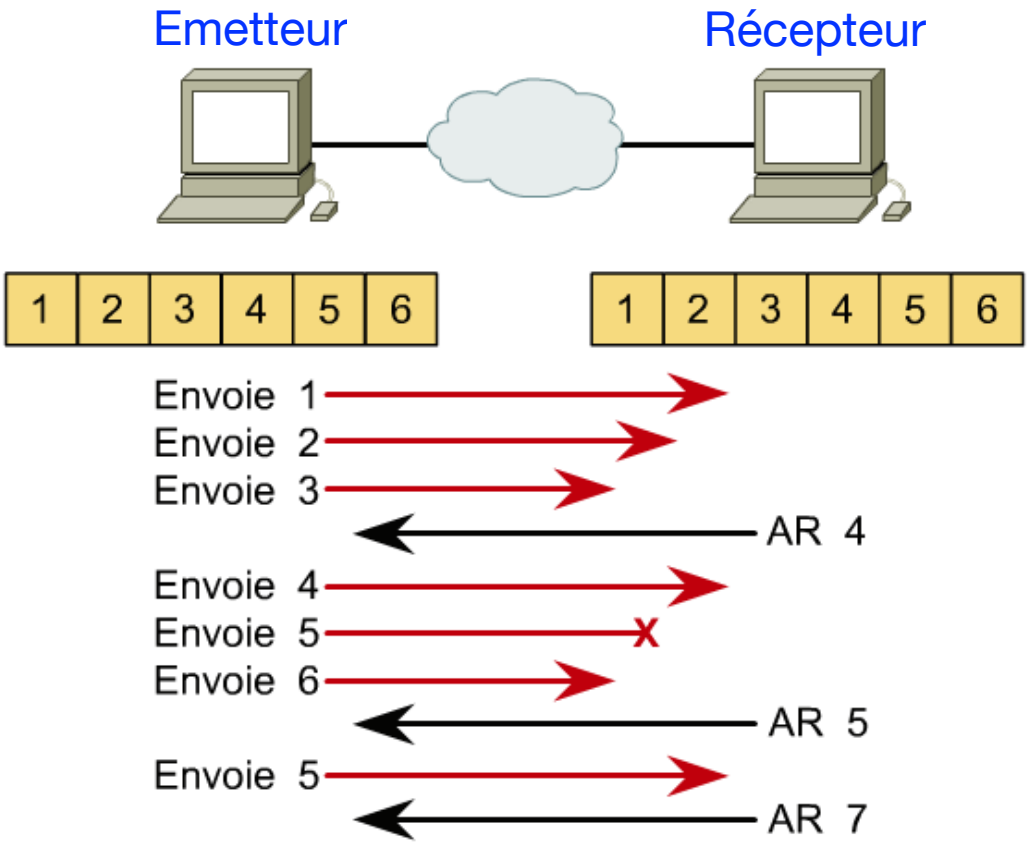
Nouveau RTT calculé chaque fois qu'un time-out expire (avant l'arrivée de l'acquittement) :

new_RTT = 2 * old_RTT

Valeur maximale fixée : new_RTT < max_RTT

Au delà de max_RTT
=> réseau considéré congestionné.

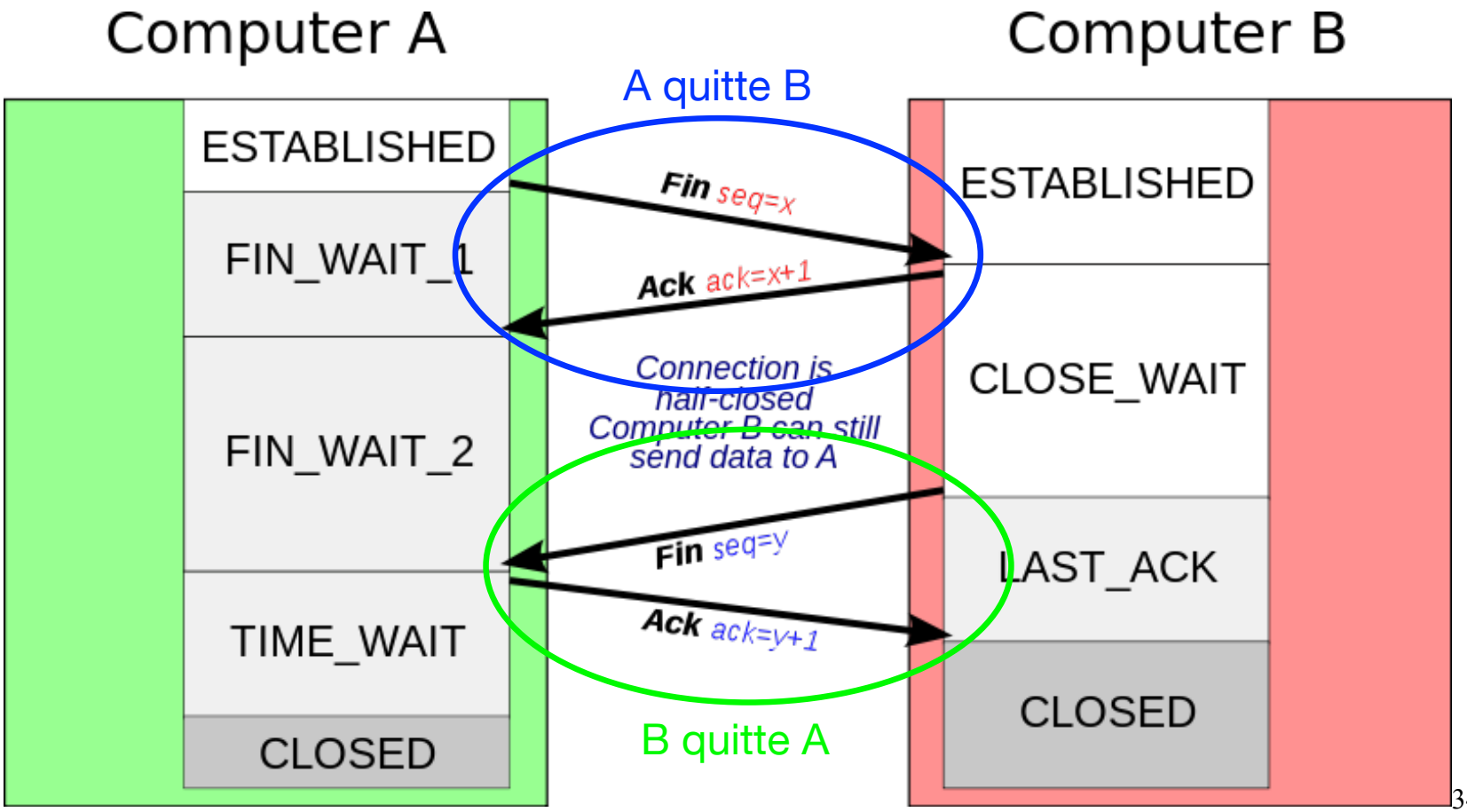
Séquençage des segments



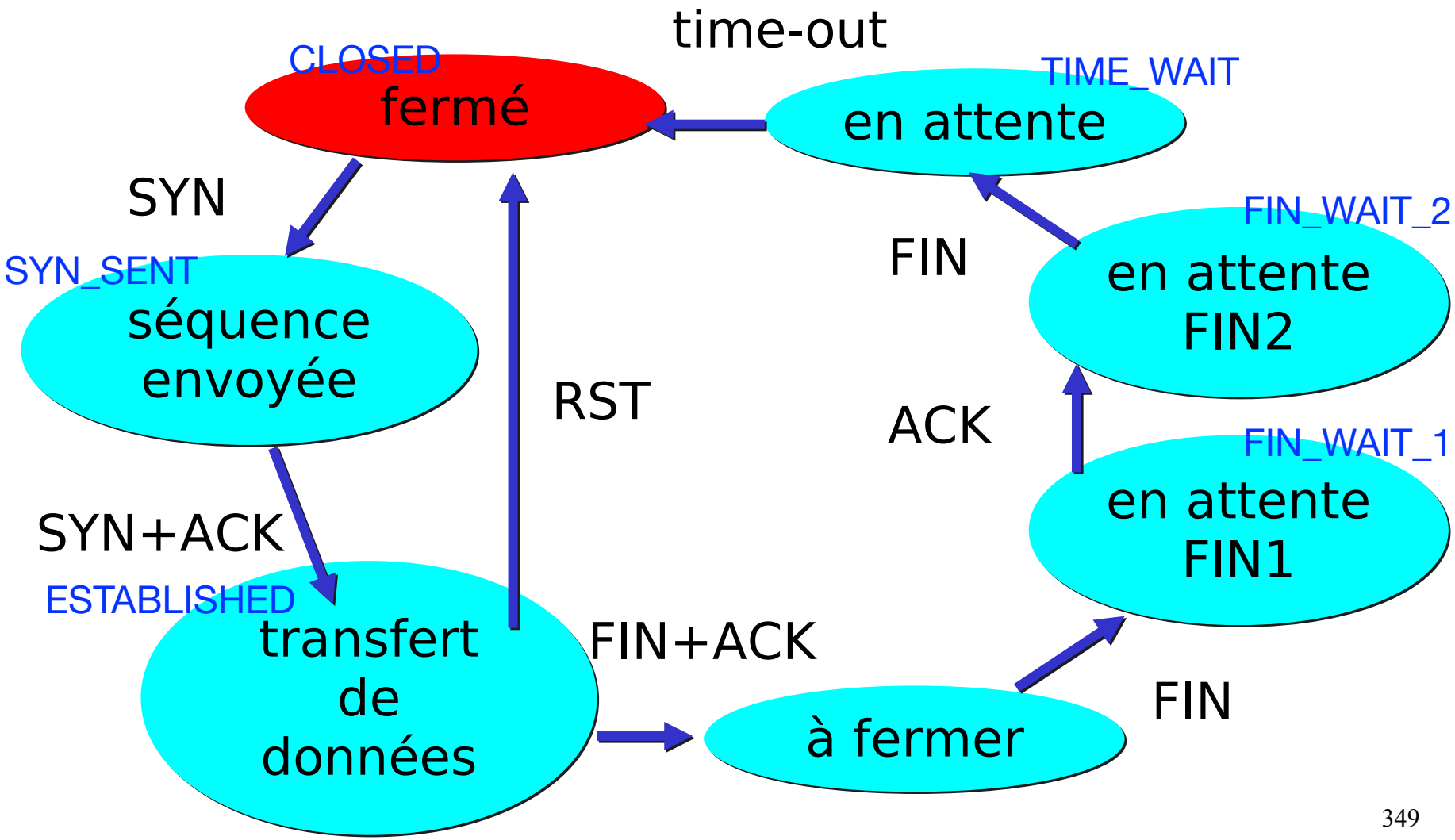
- A l' émission :
  - Chaque segment est numéroté
- A la réception :
  - TCP ré-assemble les segments
  - Si 1 n° de segment est absent
    => Demande de retransmission
    de ce segment

Technique d'accusé de réception (PAR)

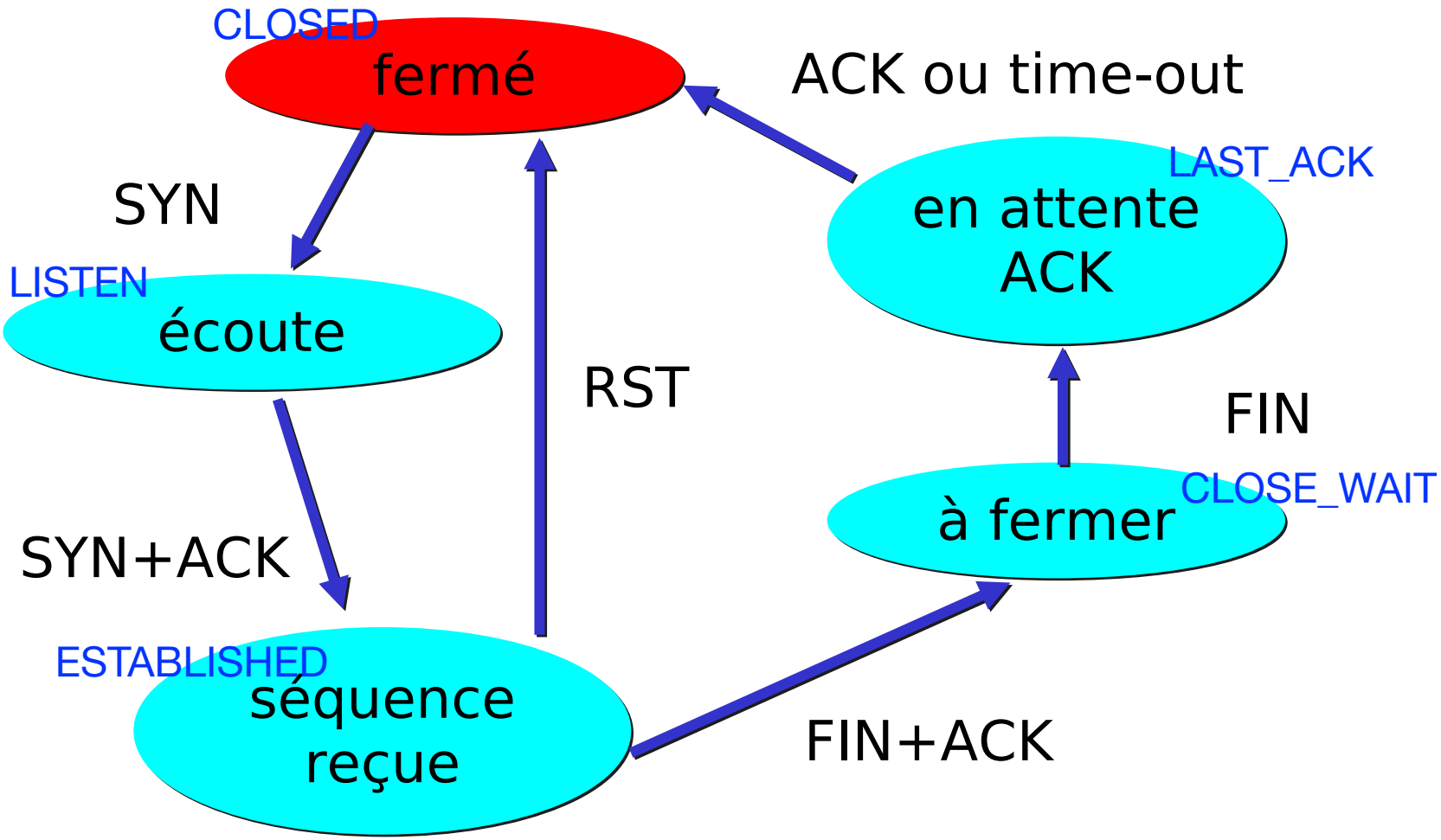
TCP - libération



TCP – états de l'émetteur

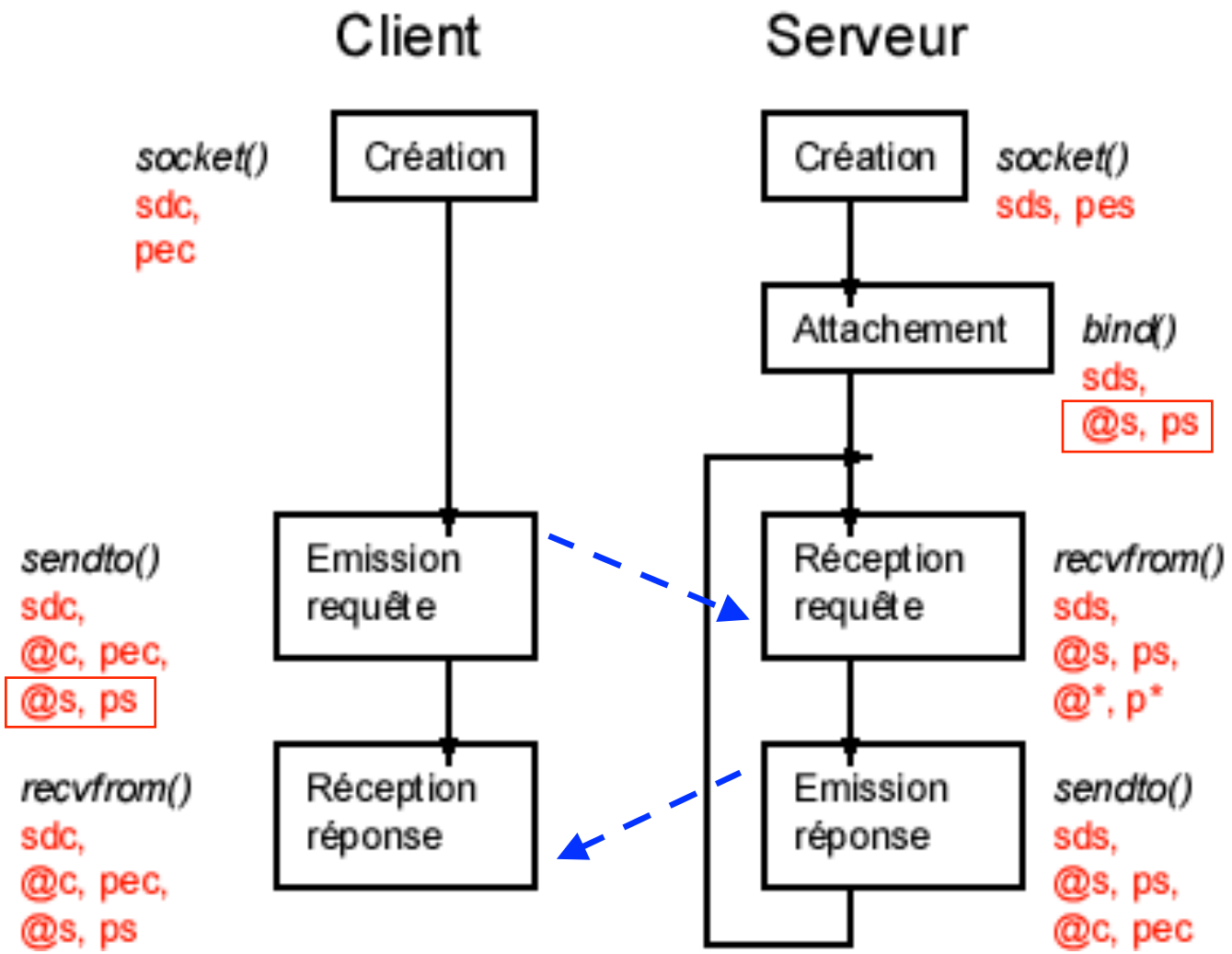


TCP – états du récepteur



UDP
Mode
datagramme
=> aucun
contrôle

Modélisation
Codes



API (interface de programmation)
Socket en mode non connecté (UDP)

options :
- u : UDP
- n : format numérique
- l : état LISTEN

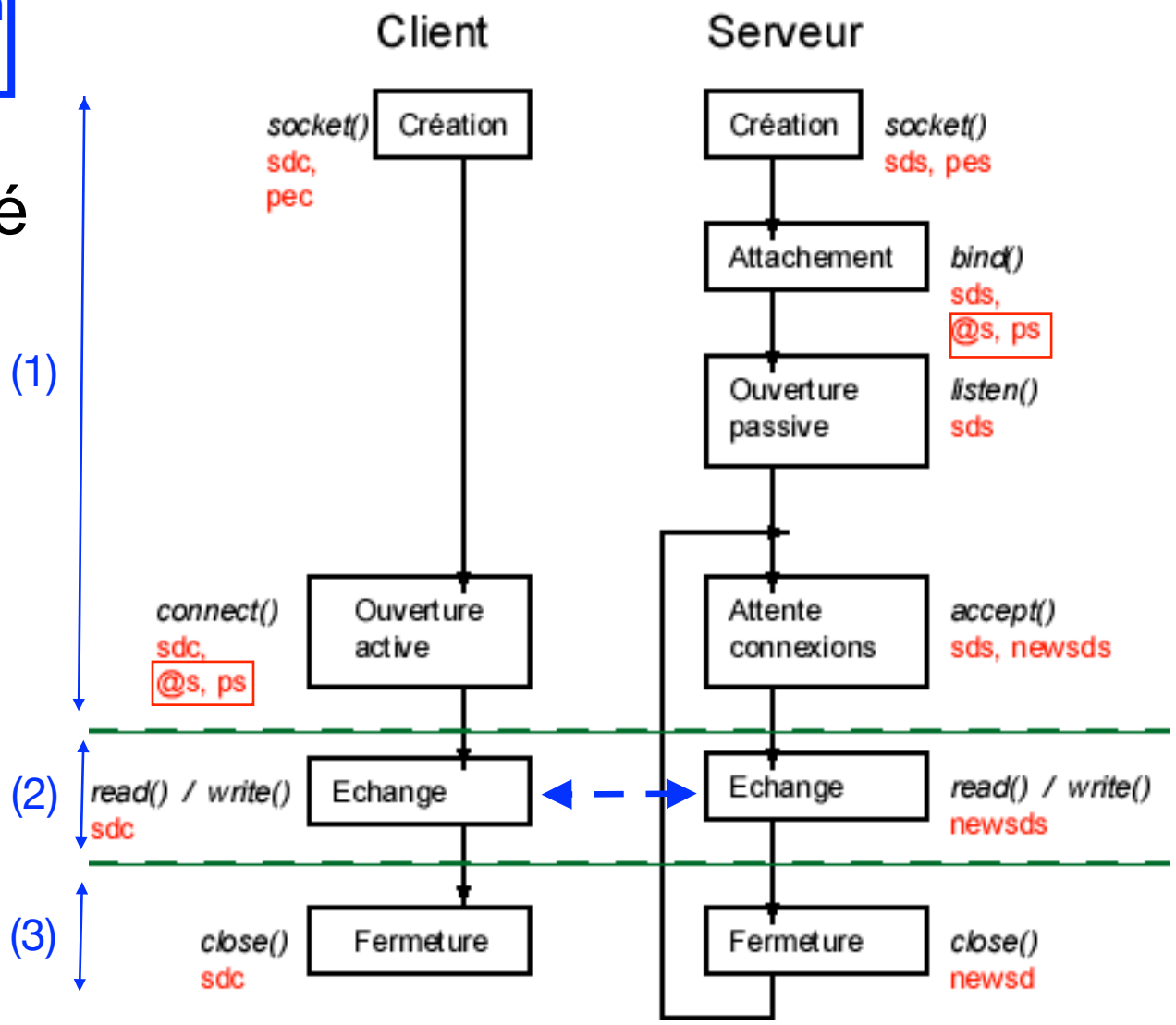
```
[ricordel@irlivc03: ] $ netstat -unl
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale        Adresse distante       Etat
udp        0      0 0.0.0.0:32779         0.0.0.0:*
udp        0      0 0.0.0.0:8000          0.0.0.0:*
udp        0      0 0.0.0.0:68            0.0.0.0:*
udp        0      0 0.0.0.0:111           0.0.0.0:*
udp        0      0 0.0.0.0:631           0.0.0.0:*
```

Netstat :
état des sockets « en écoute »
(coté serveur)

Modélisation
Codes

Socket
en mode connecté (TCP)

Serveur séquentiel



3 phases :
connexion / échanges / libération
(1) (2) (3)

TCP

options :
- t : TCP
- n : format numérique
- l : état LISTEN

```
[ricordel@irlivc03:] $ netstat -tnl
Connexions Internet actives (seulement serveurs)
```

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat
tcp	0	0	0.0.0.0:8000	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:704	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:631	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:25	0.0.0.0:*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::25	:::*	LISTEN

Netstat : (coté serveur)
Attente de connexion

TCP

options :
- t : TCP
- n : format numérique

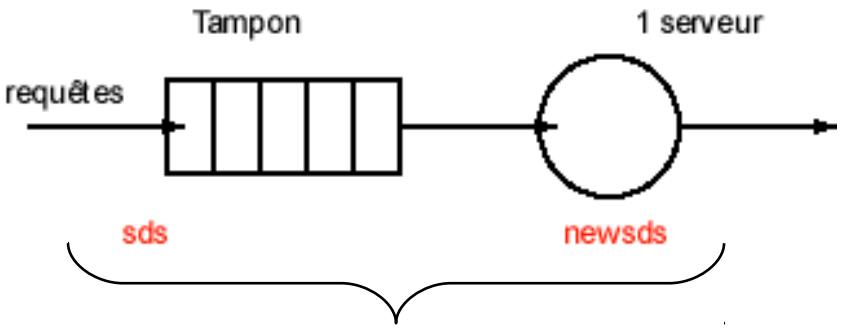
```
[ricordel@irlivc03:] $ netstat -tn
Connexions Internet actives (sans serveurs)
```

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat	
tcp	0	0	127.0.0.1:33056	127.0.0.1:8000	ESTABLISHED	Client
tcp	0	0	127.0.0.1:33052	127.0.0.1:631	TIME_WAIT	
tcp	0	0	127.0.0.1:33053	127.0.0.1:631	TIME_WAIT	
tcp	0	0	127.0.0.1:33054	127.0.0.1:631	TIME_WAIT	
tcp	0	0	127.0.0.1:33055	127.0.0.1:631	TIME_WAIT	
tcp	0	0	127.0.0.1:8000	127.0.0.1:33056	ESTABLISHED	Serveur

Cas particulier : Client & Serveur sont sur la boucle locale (127.0.0.1)
et connectés

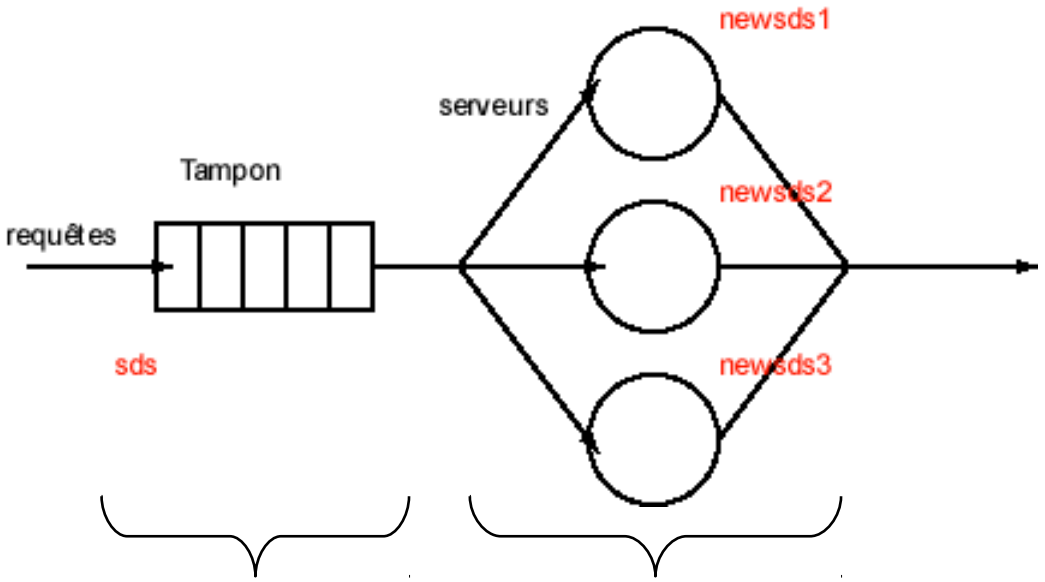
Netstat : (coté client ou serveur)
Connexion établie

Serveur TCP séquentiel



1 seul processus => attendre fin du service avant le client suivant

Serveur TCP parallèle



1 processus

3 processus en //

Bibliographie

- A. Tanenbaum, « Réseaux »
  éd. Prentice Hall, 2003
- G. Pujolle, « Les Réseaux »,
  éd. Eyrolles, 1995
- J. Kurose & K. Ross, « Analyse structurée des réseaux », éd. Pearson, 2003
- Documentation Cisco
  (source de nombreuses illustrations de ce cours)

& Cours « Computer Networks and Internet » de P. Bakowski