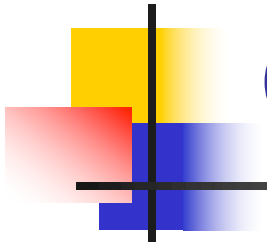
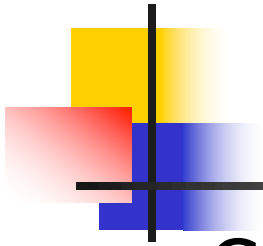

# Partie 7 : Internet et l'architecture TCP/IP

Olivier GLÜCK
Université LYON 1 / Département Informatique
Olivier.Gluck@univ-lyon1.fr
http://perso.univ-lyon1.fr/olivier.gluck

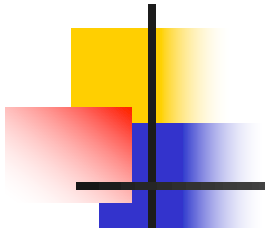


# Copyright



# Remerciements

- Certains transparents sont basés sur des supports de cours de :
  - Danièle DROMARD (PARIS 6)
  - Andrzej DUDA (INP Grenoble/ENSIMAG)
  - Shivkumar KALYANARAMAN (RPI/ECSE)
  - Alain MILLE (LYON 1)
  - CongDuc PHAM (LYON 1)
  - Michel RIVEILL (Université de Nice/ESSI)
  - L'Institut National des Télécommunications (INT)
  - Cisco Networking Academy
- Des figures sont issues des livres cités en bibliographie

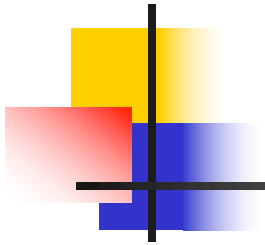


# Bibliographie


- « *Réseaux* », 4ième édition, Andrew Tanenbaum, Pearson Education, ISBN 2-7440-7001-7
- « *Réseaux et Télécoms* », Claude Servin, Dunod, ISBN 2-10-007986-7
- « *Analyse structurée des réseaux* », 2ième édition, J. Kurose et K. Ross, Pearson Education, ISBN 2-7440-7000-9
- « *TCP/IP Illustrated Volume 1, The Protocols* », W. R. Stevens, Addison Wesley, ISBN 0-201-63346-9
- « *TCP/IP, Architecture, protocoles, applications* », 4ième édition, D. Comer, Dunod, ISBN 2-10-008181-0
- « *An Engineering Approach to Computer Networking* », Addison-Wesley, ISBN 0-201-63442-6
- Internet…
  - http://www.guill.net/
  - http://www.courseforge.org/courses/
  - http://www.commentcamarche.net/ccmdoc/
  - http://www.rfc-editor.org/ (documents normatifs dans TCP/IP)

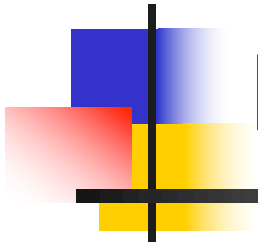
# Bibliographie


- « *Réseaux* », 4ième édition, Andrew Tanenbaum, Pearson Education, ISBN 2-7440-7001-7
- « *Réseaux et Télécoms* », Claude Servin, Dunod, ISBN 2-10-007986-7
- « Réseaux locaux et Internet, des protocoles à l'interconnexion », 3ième édition, Laurent Toutain, Hermes Science, ISBN 2-7462-0670-6
- « *An Engineering Approach to Computer Networking* », Addison-Wesley, ISBN 0-201-63442-6
- Internet…
  - http://www.guill.net/
  - http://www.courseforge.org/courses/
  - http://www.commentcamarche.net/ccmdoc/
  - http://www.protocols.com/
  - http://dir.yahoo.com/Computers_and_Internet/
  - http://www.rfc-editor.org/ (documents normatifs dans TCP/IP)



# Plan de la partie 7

- Description générale
- Adressage dans l'Internet (IPv4)
- Le protocole IP (IPv4)
- Les protocoles de routage de l'Internet
- Protocoles de contrôle de l'Internet et les utilitaires réseaux
- Le protocole IPv6
- Les protocoles de transport
- Exemples de connexion à Internet



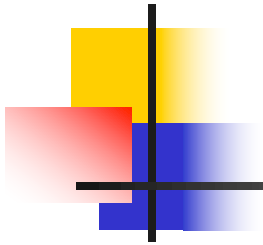
# Description générale

Visage de l'Internet

Architecture TCP/IP

Protocoles et applications

Identification des protocoles et applications

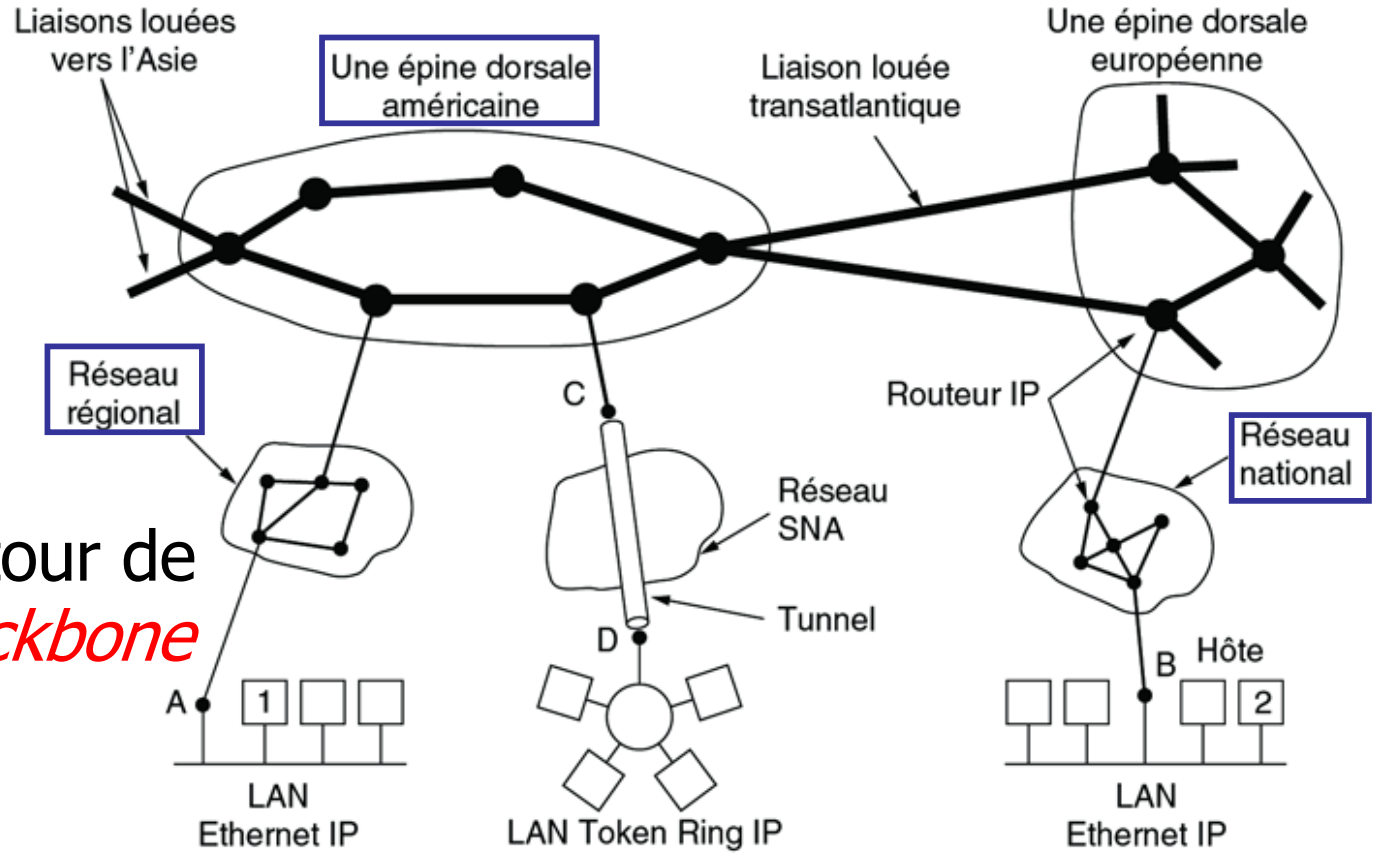


# Historique et acronymes

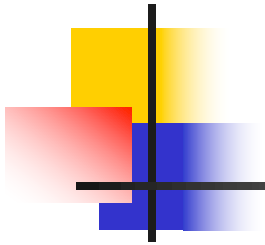
- Architecture développée par la DARPA (*Defence Advanced Research Project Adgency*), milieu des années 1970
- IP : *Internet Protocol* - résout les problèmes d'interconnexion en milieu hétérogène (1974)
- TCP : *Transmission Control Protocol* - protocole de transport de l'Internet (de bout en bout)
- TCP/IP est intégré à Unix BSD 4 (Berkeley) en 1980
- TCP/IP est intégré à ARPANET en 1983
- Aujourd'hui, TCP/IP est devenu le standard d'Internet (Internet pour Inter-Networking)

# Le visage de l'Internet (1)

- Un ensemble de sous-réseaux indépendants (*Autonomous System*) et hétérogènes qui sont interconnectés (organisation hiérarchique)




S'articule autour de plusieurs *backbone*

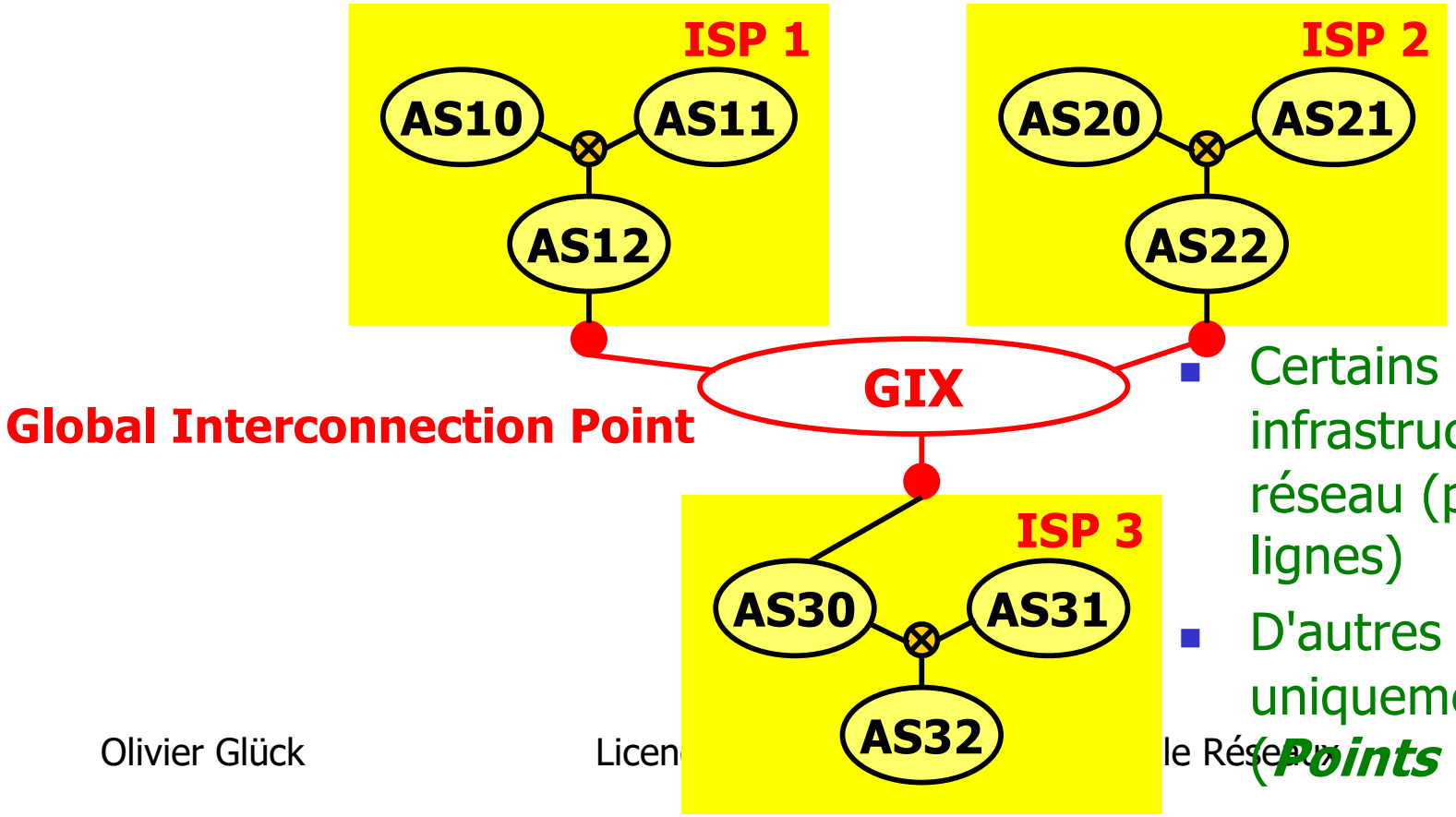


# Le visage de l'Internet (2)

- Trois types de systèmes autonomes
  - les AS de transit (*backbone*) (réseaux régionaux, nationaux, ...) qui acceptent de faire transiter des paquets d'autres AS
    - parfois avec certaines restrictions
    - souvent moyennant finance
  - les puits (*stubs*) : réseaux sans issue qui ne peuvent acheminer aucun trafic externe
  - les AS multi-connectés qui peuvent être utilisés pour du transit, sauf indication contraire (mais ce n'est pas leur rôle premier)
- ***Peering*** : accords de transit entre ISP -> points d'interconnexion privés

# Le visage de l'Internet (3)

- ISP - Internet Service Provider
  - un ou plusieurs systèmes autonomes
  - un AS = ensemble de réseaux/routeurs sous la même autorité d'administration (entreprise, campus, ...)

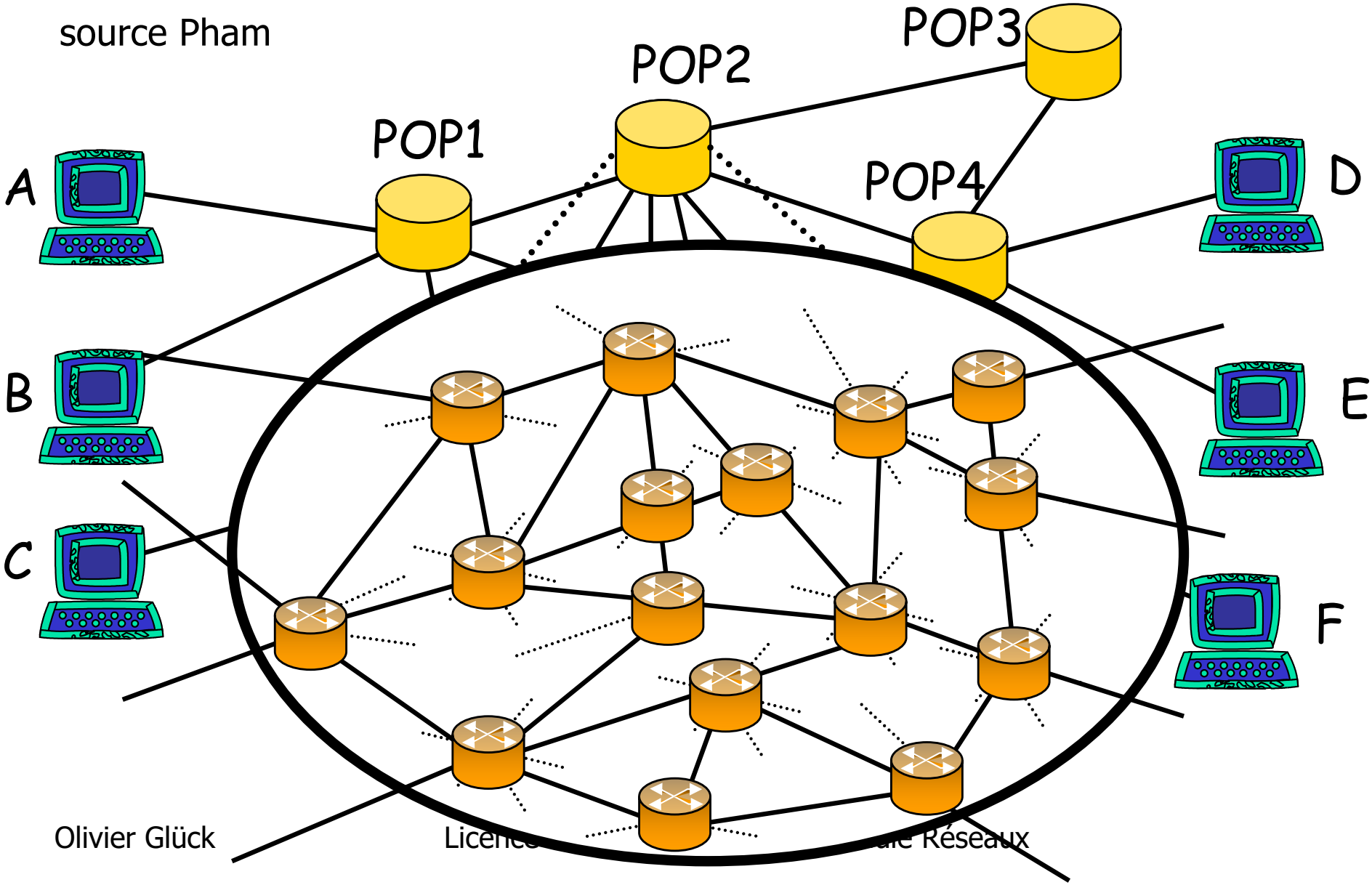



- Certains ISP ont une infrastructure physique de réseau (possèdent des lignes)
- D'autres proposent uniquement des POPs (***Points of Presence***)

# Le visage de l'Internet (4)

**POP = interface entre le réseau d'accès et le réseau de transit**

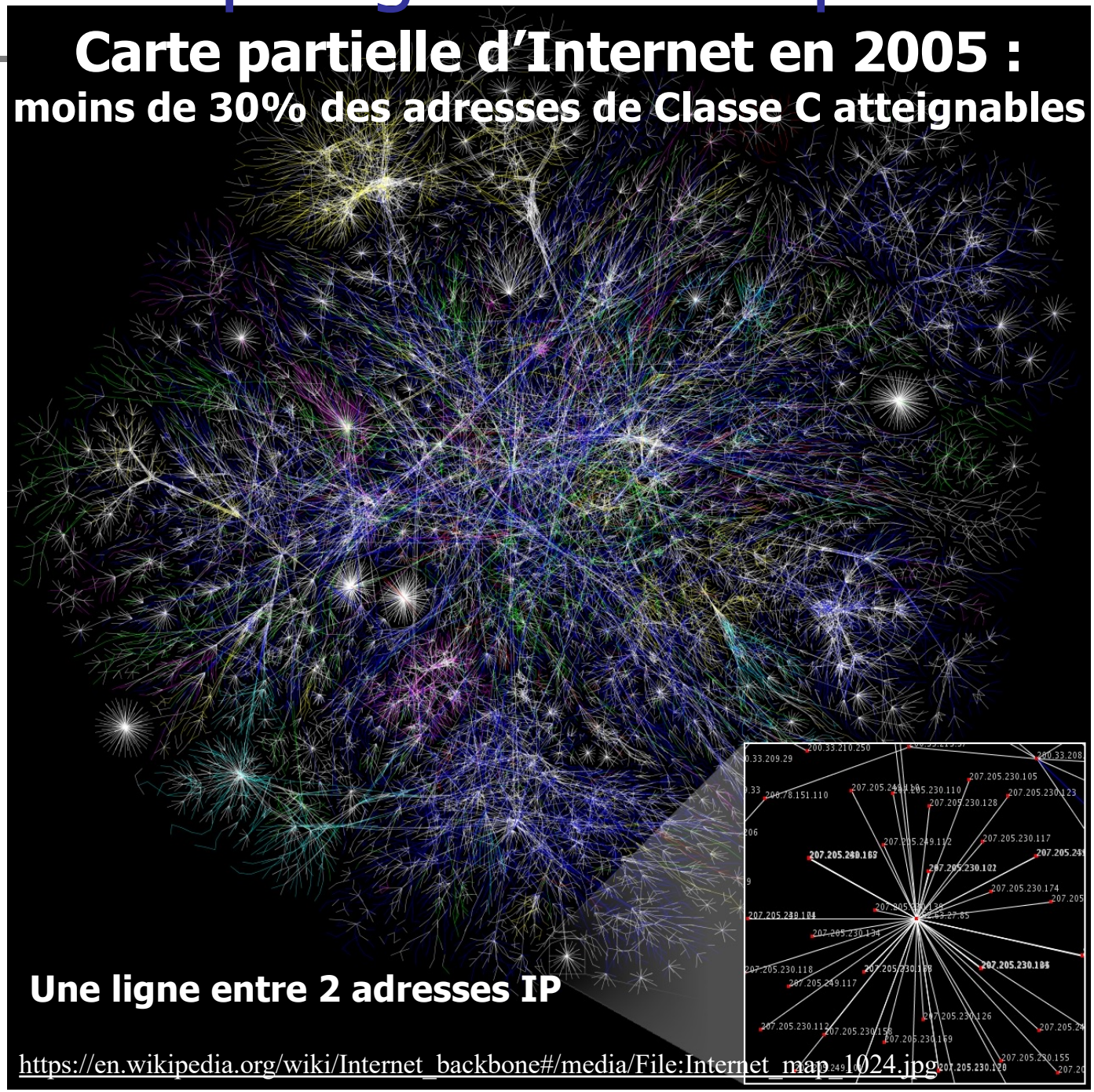
source Pham



# Internet : une topologie très complexe

**La cartographie d'Internet n'existe pas !**

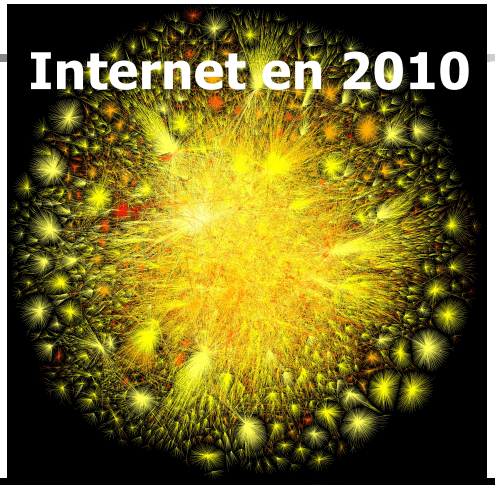
**Carte réalisée grâce à des ping entre deux adresses IP**



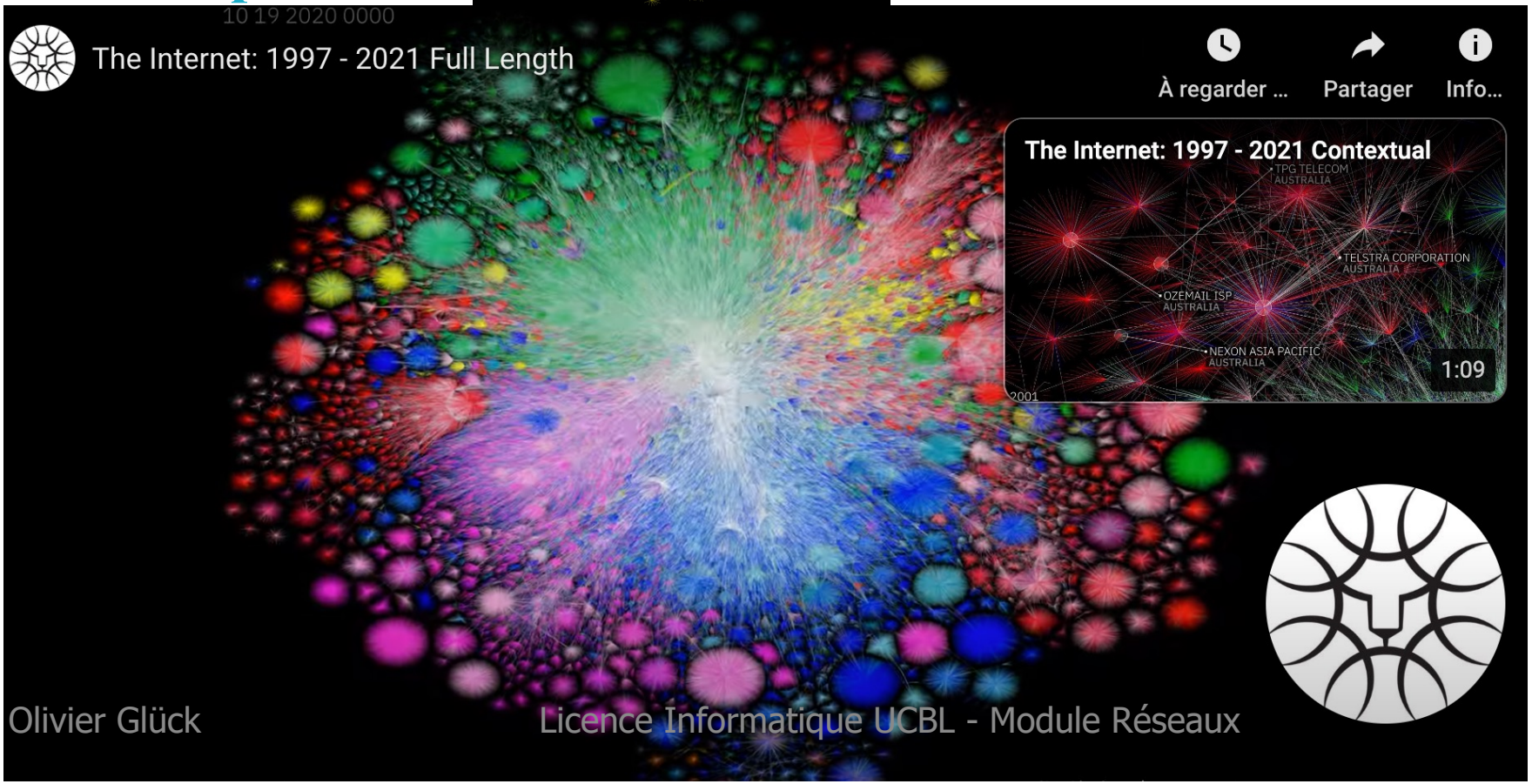

By The Opte Project - Originally from the English Wikipedia; description page is/was here., CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1538544

# Internet : une topologie très complexe

**La cartographie d'Internet n'existe pas !**

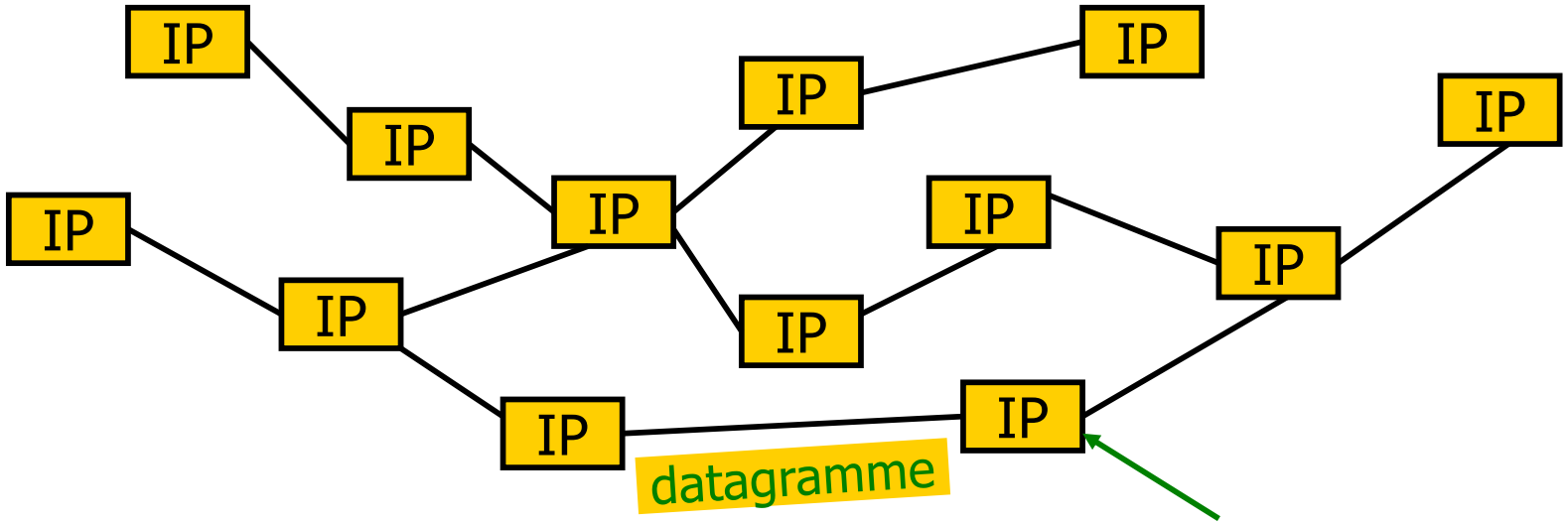



By The Opte Project - https://www.opte.org/the-internet



# Fonctionnement de l'Internet (1)

## **Couche réseau** : communications entre machines

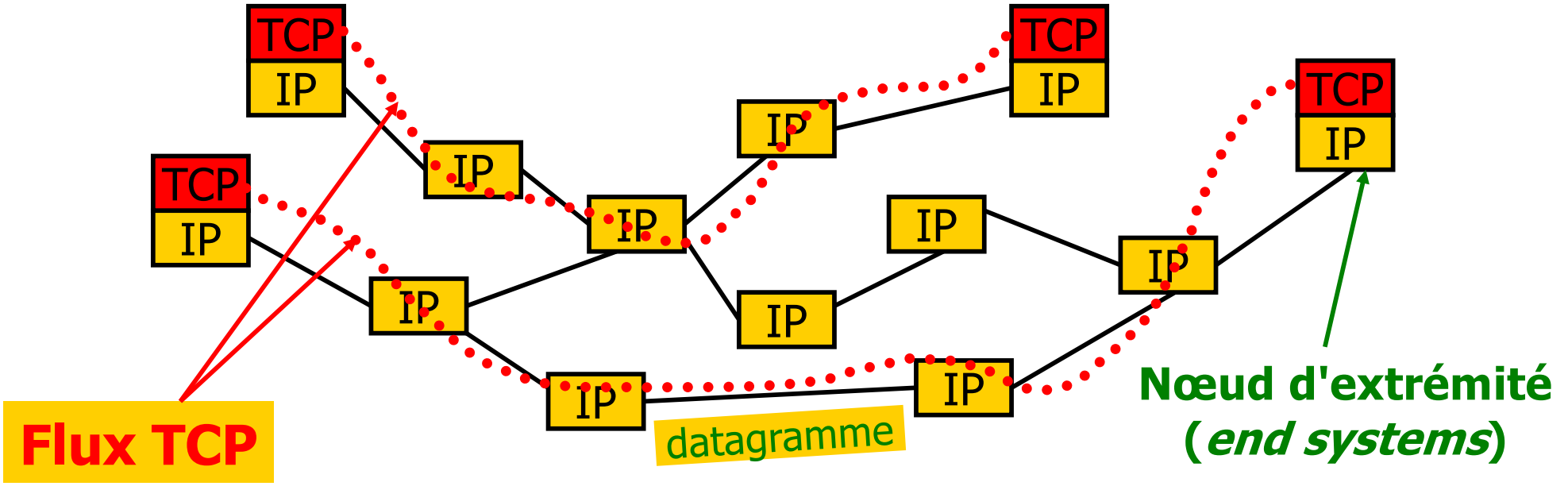



**Nœud intermédiaire : routeur (matériel ou logiciel)**

- IP - protocole d'interconnexion, best-effort
  - acheminement de **datagrammes** (mode non connecté)
  - peu de fonctionnalités, pas de garanties
  - simple mais robuste (défaillance d'un nœud intermédiaire)

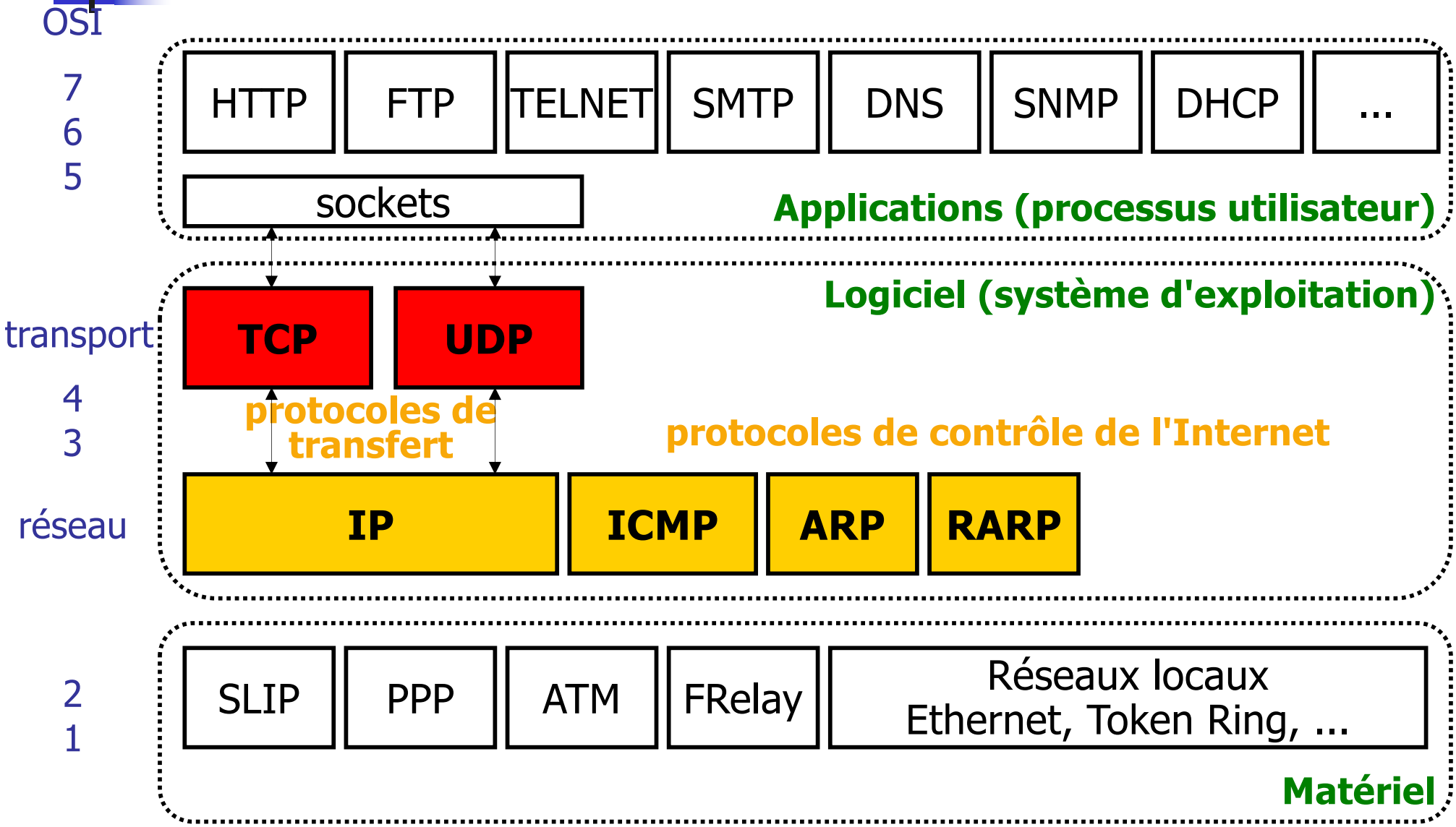
# Fonctionnement de l'Internet (2)

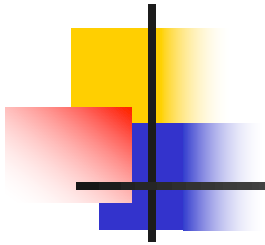
**Couche transport** : communications entre applis




- TCP - protocole de transport **de bout en bout**
  - uniquement présent aux extrémités
  - transport **fiable** de **segments** (mode connecté)
  - protocole complexe (retransmission, gestion des erreurs, séquencement, ...)

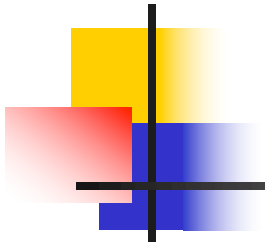
# Architecture de TCP/IP





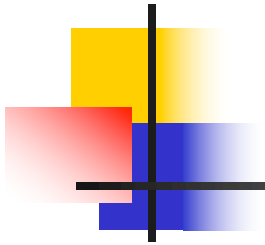
# Protocoles et applications (1)

- Niveau applicatif
  - HTTP - HyperText Transport Protocol
    - protocole du web
    - échange de requête/réponse entre un client et un serveur web
  - FTP - File Transfer Protocol
    - protocole de manipulation de fichiers distants
    - transfert, suppression, création, ...
  - TELNET - TELetypewriter Network Protocol
    - système de terminal virtuel
    - permet l'ouverture d'une session distante



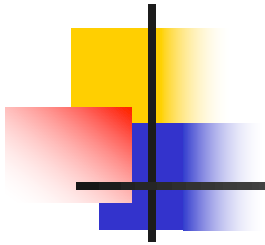
# Protocoles et applications (2)

- Niveau applicatif
  - SMTP - Simple Mail Transfer Protocol
    - service d'envoi de courrier électronique
    - réception (POP, IMAP, IMAPS, ...)
  - DNS - Domain Name System
    - assure la correspondance entre un nom symbolique et une adresse Internet (adresse IP)
    - bases de données réparties sur le globe
  - SNMP - Simple Network Management Protocol
    - protocole d'administration de réseau (interrogation, configuration des équipements, ...)
  - Les sockets - interface de programmation permettant l'échange de données (via TCP ou UDP)



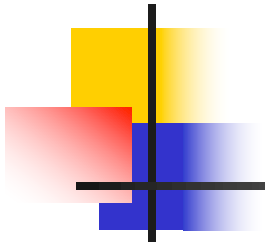
# Protocoles et applications (3)

- Protocoles de transfert de données
  - TCP/IP : transfert fiable de données en mode connecté
  - UDP/IP : transfert non garanti de données en mode non connecté
- Protocoles de contrôle de l'Internet
  - ICMP - Internet Control and error Message Protocol
    - assure un dialogue IP<-->IP (entre routeurs par ex.) pour signaler les congestions, synchroniser les horloges, estimer les temps de transit, …
    - utilisé par l'utilitaire *ping* permettant de tester la présence d'une station sur le réseau



# Protocoles et applications (4)

- Protocoles de contrôle de l'Internet
  - ARP - Address Resolution Protocol
    - protocole permettant d'associer une adresse MAC (adresse physique utilisée dans les réseaux locaux) à une adresse IP (adresse logique Internet)
  - RARP - Reverse ARP
    - permet à une station de connaître son adresse IP à partir de son adresse MAC (interrogation d'un serveur RARP)
    - phase de démarrage d'équipements ne possédant pas de configuration initiale (imprimante, terminal X)

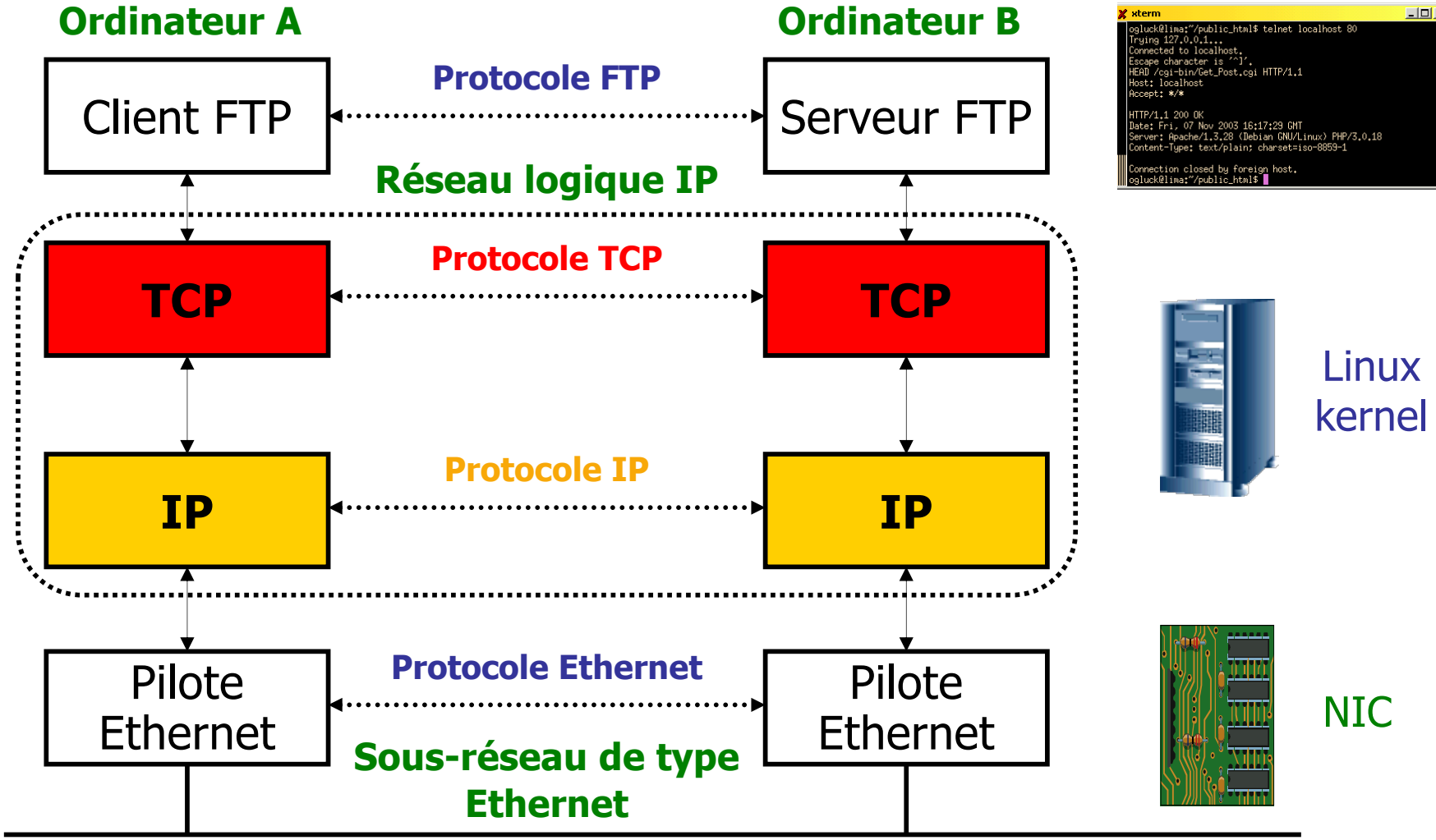


# Protocoles et applications (5)

- Protocoles de contrôle de l'Internet
  - BOOTP - Boot Protocol
    - permet à une station de connaître sa configuration réseau lors du démarrage par interrogation d'un serveur *bootp*
    - au-dessus d'UDP (ports 67 et 68)
  - DHCP - Dynamic Host Configuration Protocol
    - extension du protocole BOOTP
    - meilleure gestion du plan d'adressage IP avec attribution dynamique des adresses IP pour une certaine durée (bail ou *lease time*)
    - au-dessus d'UDP (ports 67 et 68)

# Communications sans routeur

- Deux machines sur un même sous réseau









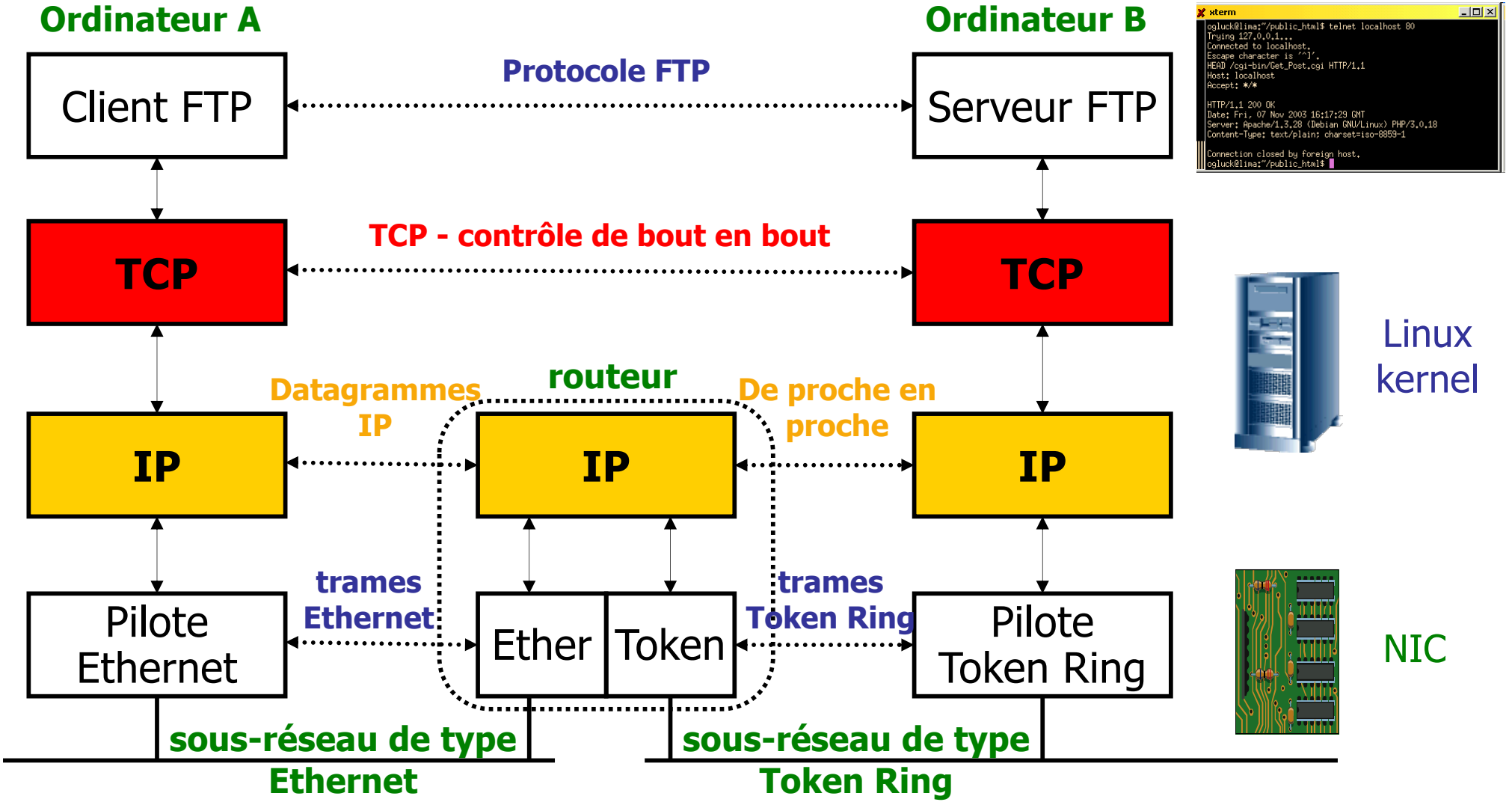
Linux kernel



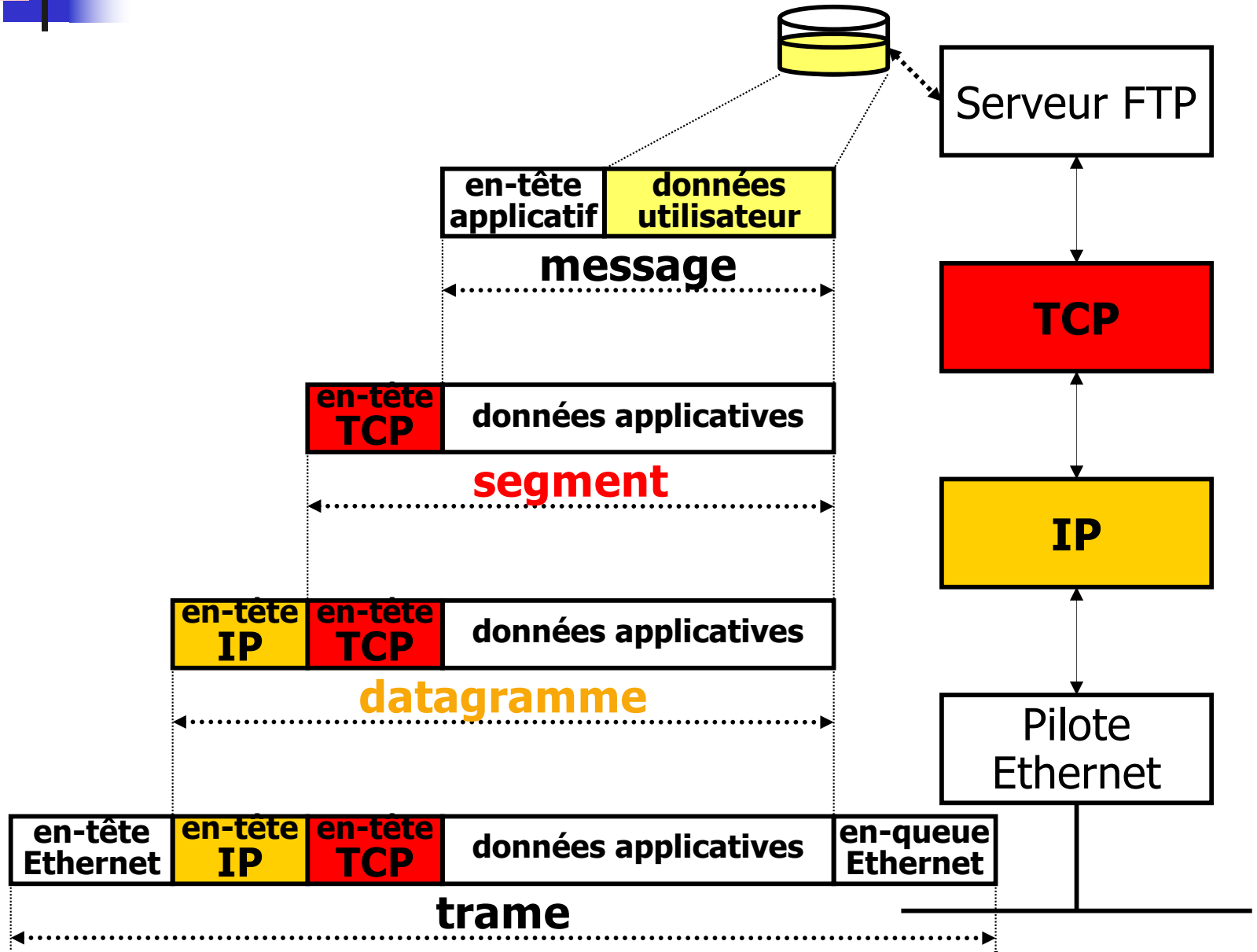
NIC

# Communications avec routeur(s)

- Prise en compte de l'hétérogénéité

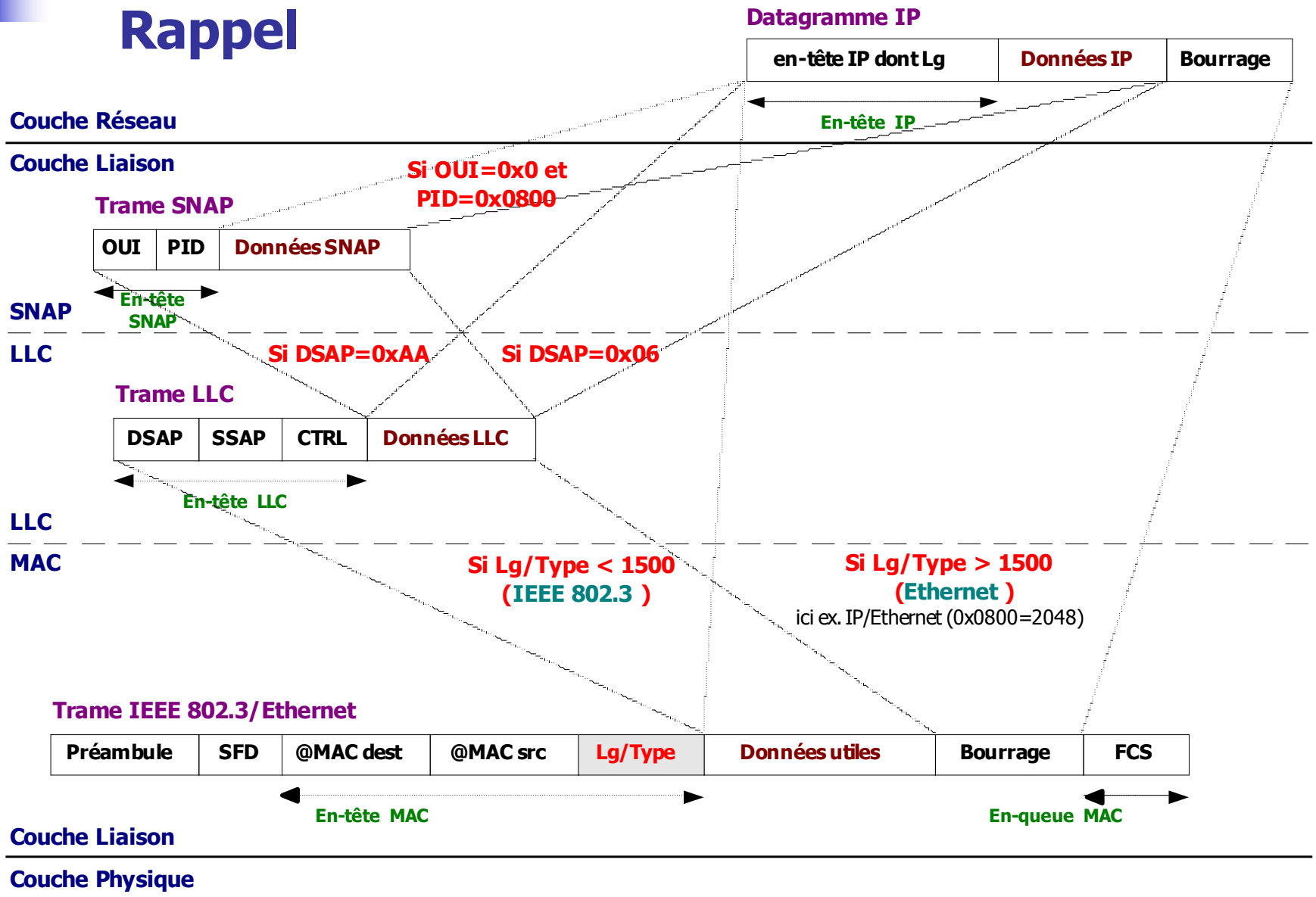


# Encapsulation

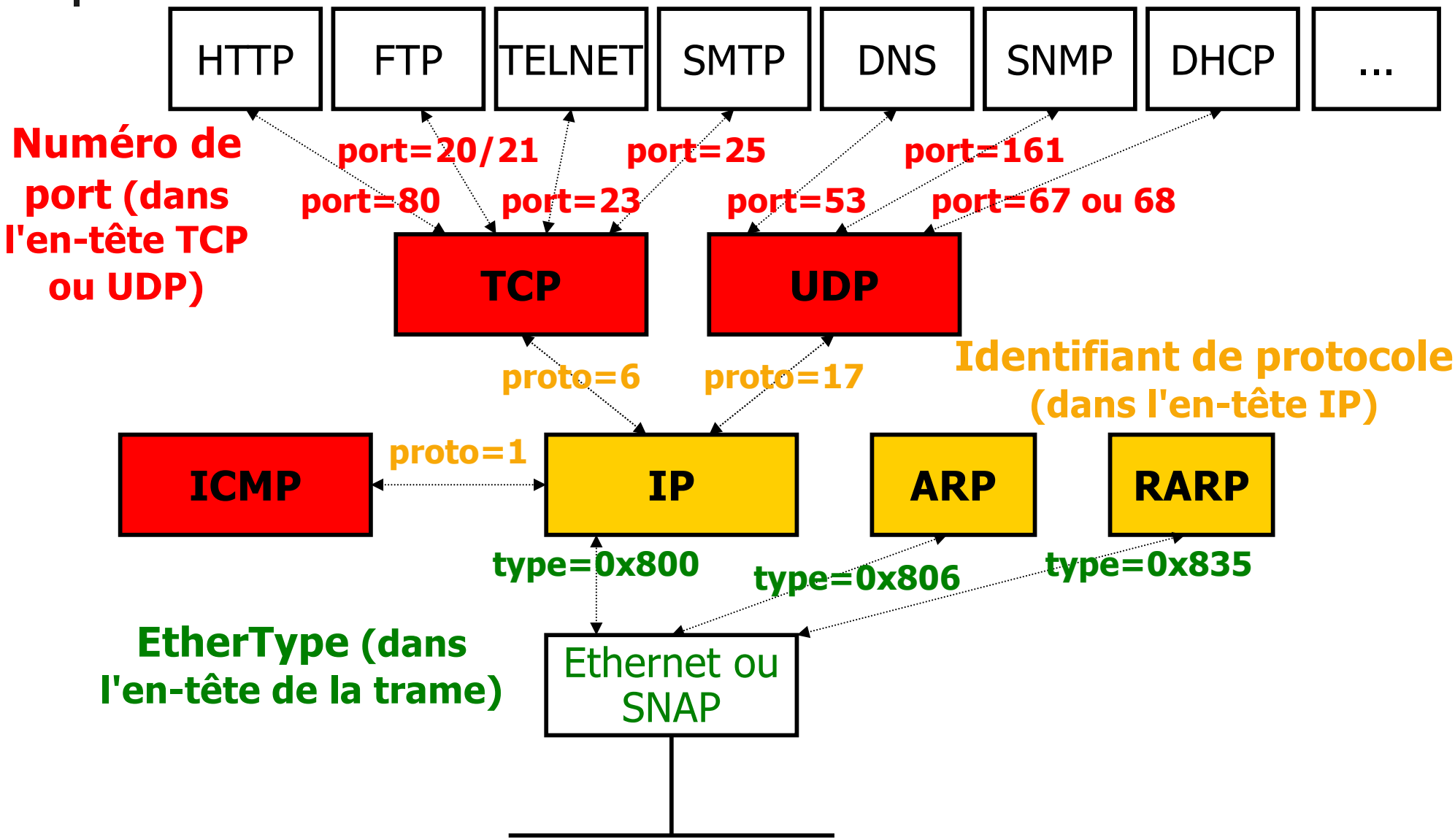


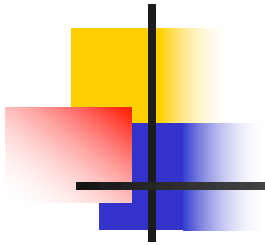
# Identification des protocoles (1)

## Rappel



# Identification des protocoles (2)



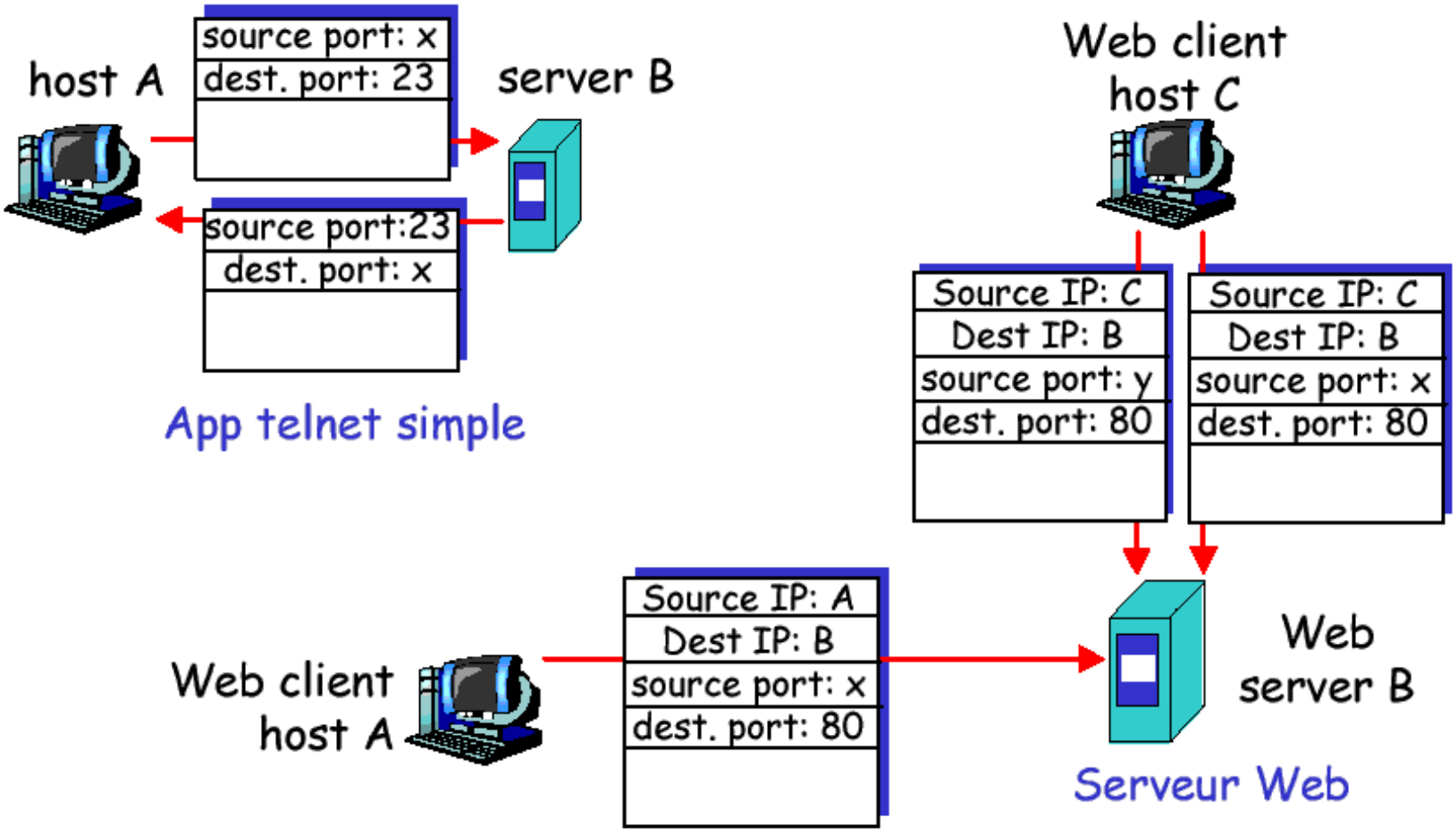


# Identification des protocoles (3)

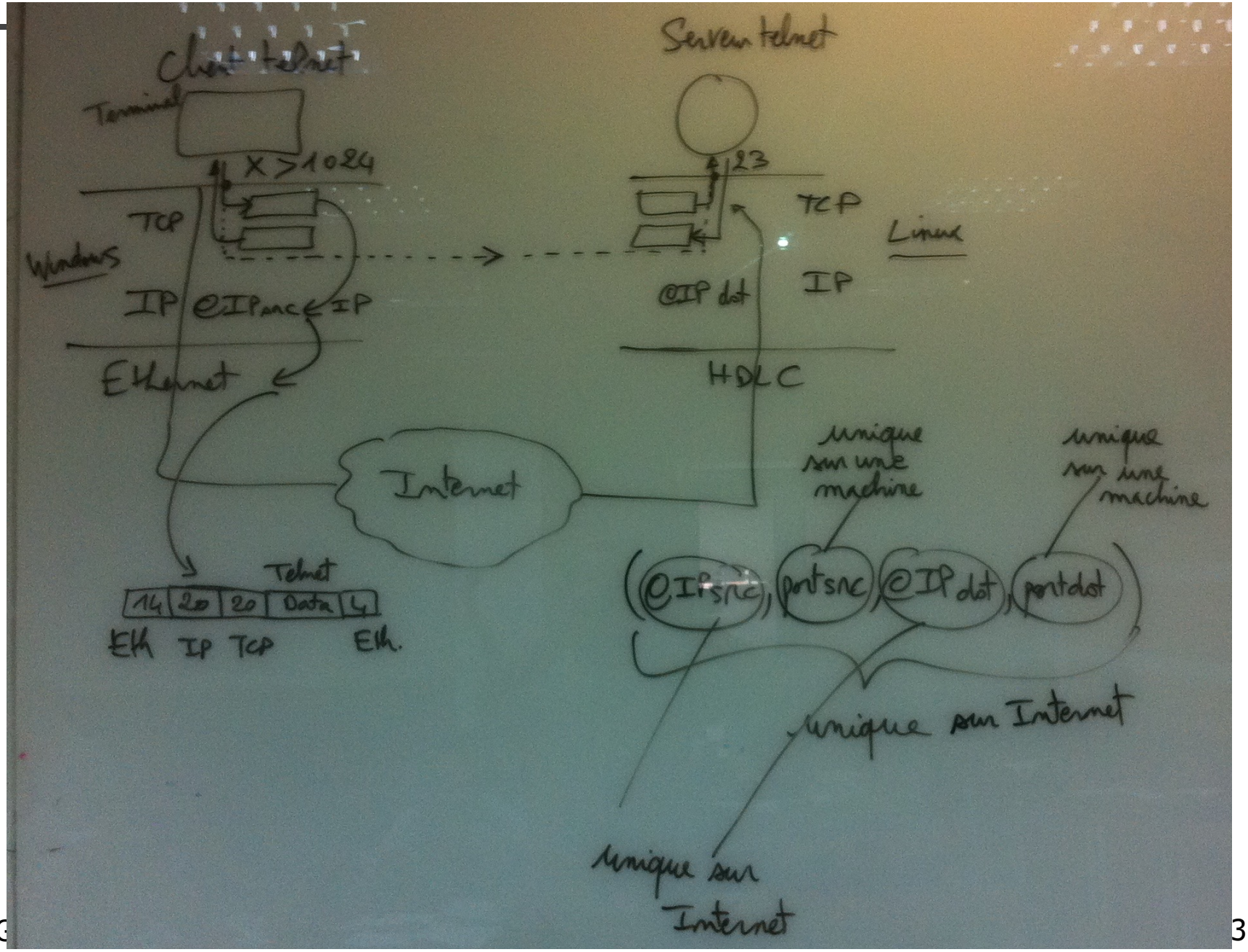
- Une adresse de transport = une adresse IP + un numéro de port (16 bits) + TCP ou UDP -> adresse de socket
- Une connexion TCP s'établit entre une socket source et une socket destinataire -> une connexion = un quadruplé (@IPsrc, port src, @IPdest, port dest)
- Deux connexions peuvent aboutir à la même socket
- Les ports permettent un multiplexage ou démultiplexage de connexions au niveau transport
- Les ports inférieurs à 1024 sont appelés **ports réservés**

# Identification des protocoles (4)

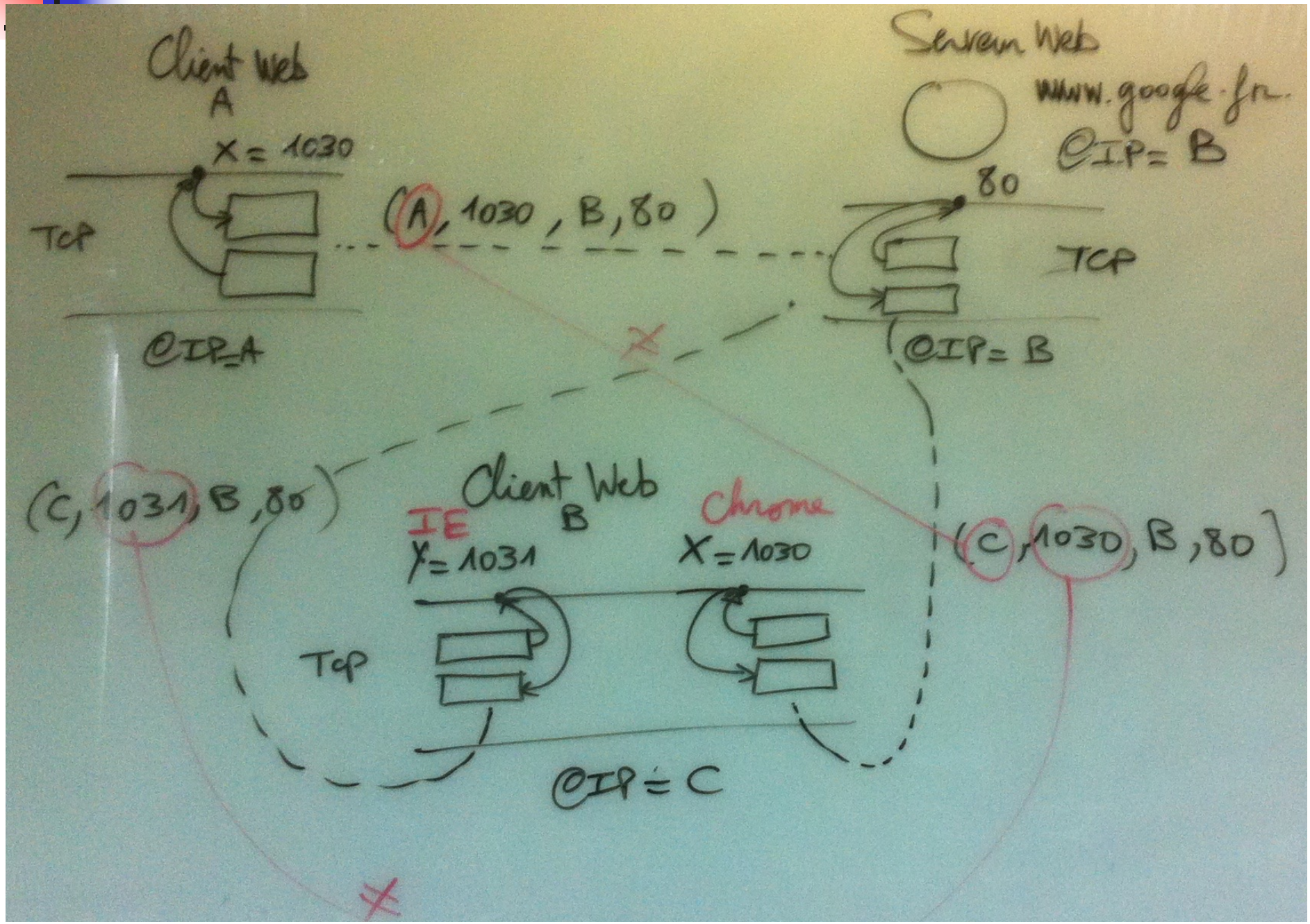
## Multiplexage/demultiplexage: exemples



# Identification des protocoles (4)



# Identification des protocoles (4)




# Adressage dans l'Internet (IPv4)

Format de l'adresse IPv4

Les classes d'adressage

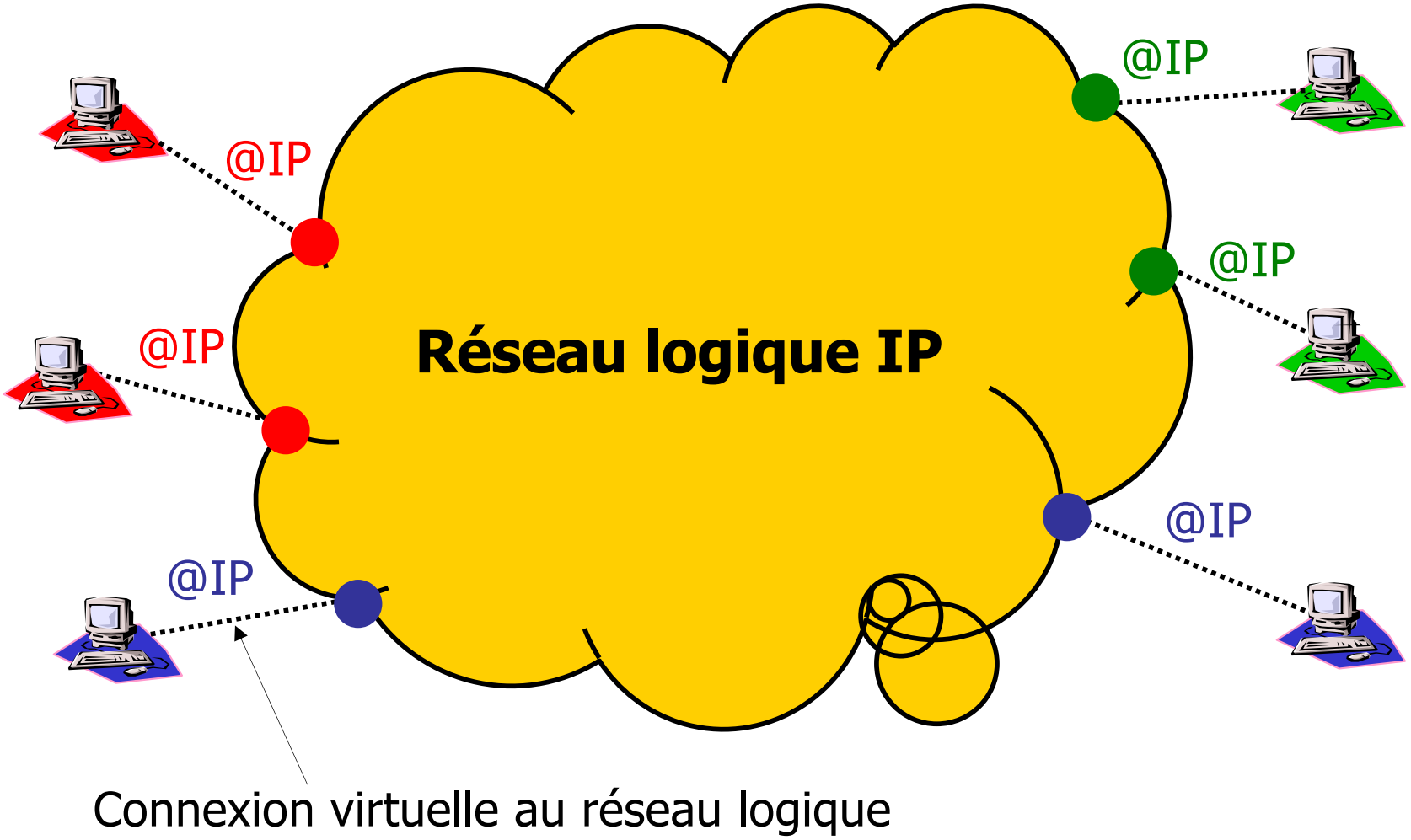
Adresses IP particulières

Adresses privées et NAT

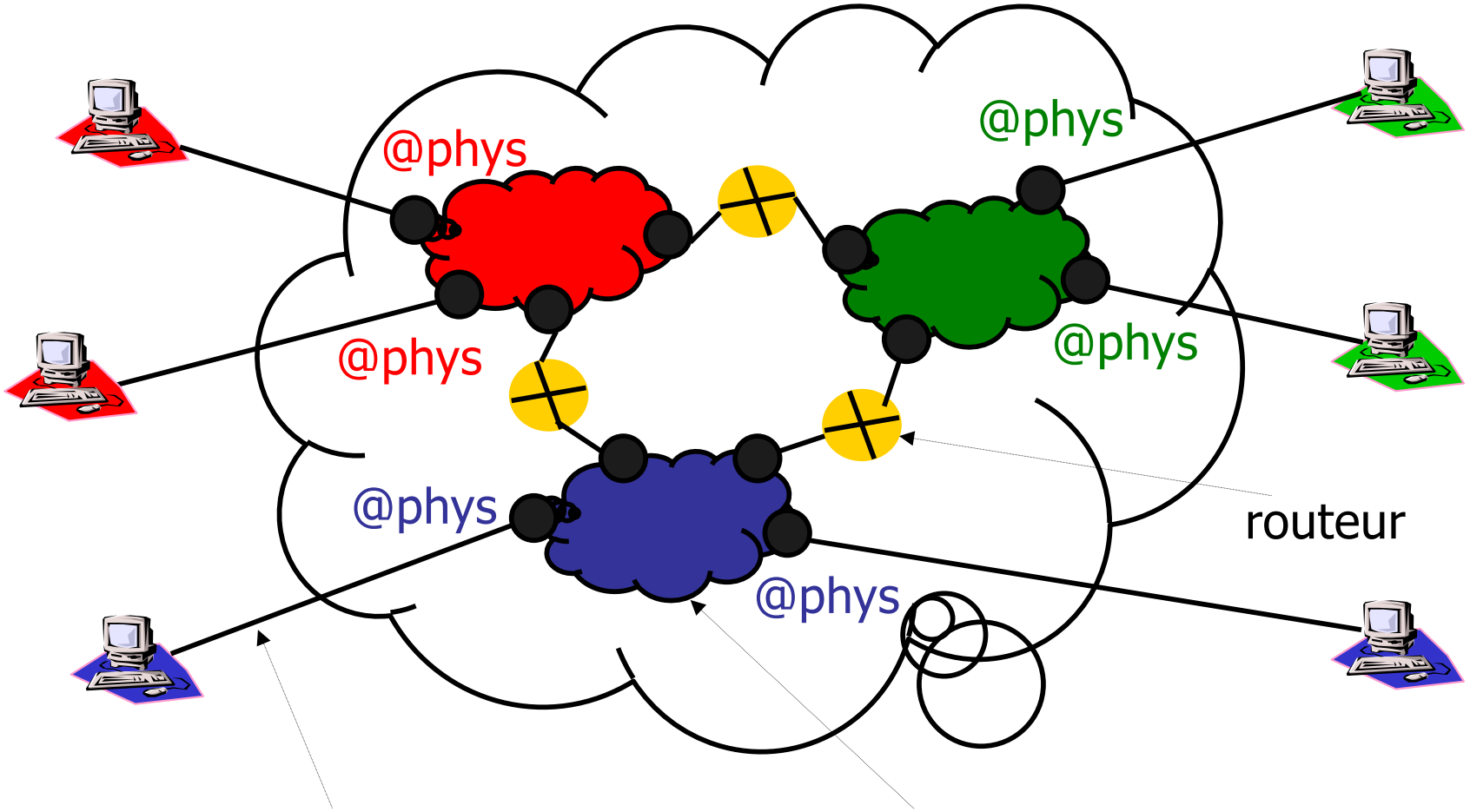
Les sous-réseaux

Adressage géographique (CIDR)

# L'Internet du point de vue utilisateur

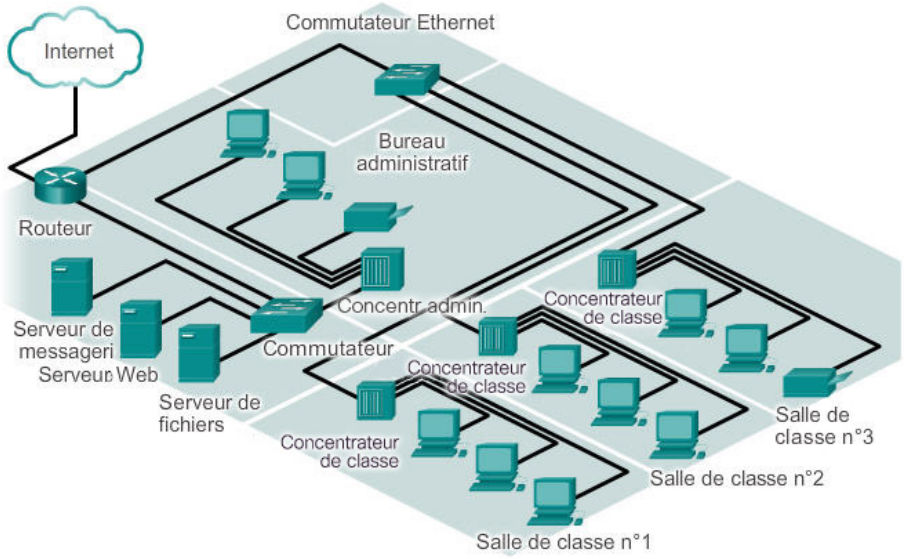


# L'Internet du point de vue réel

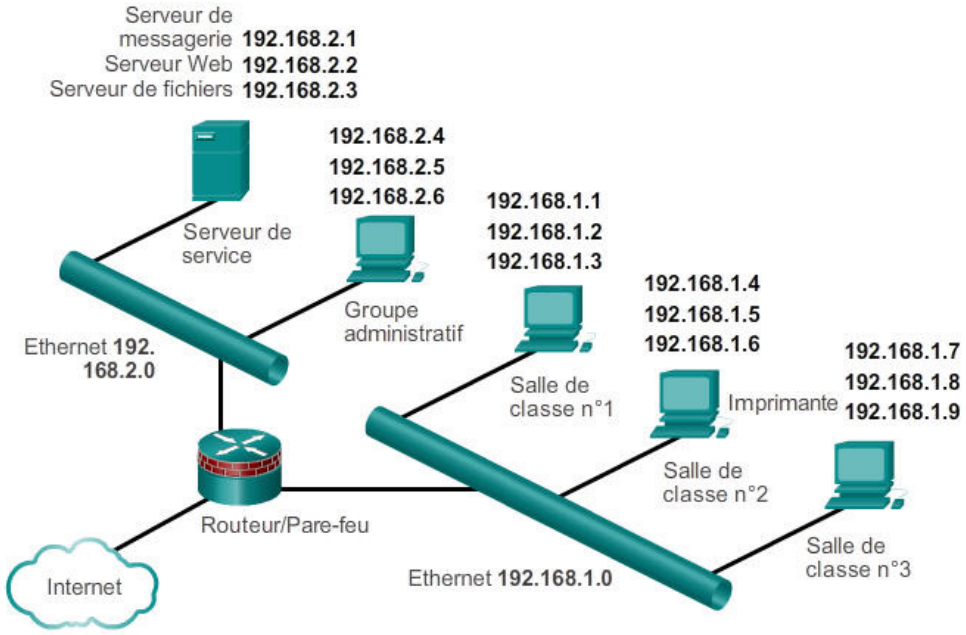


# Topologie physique/logique

**Topologie physique**

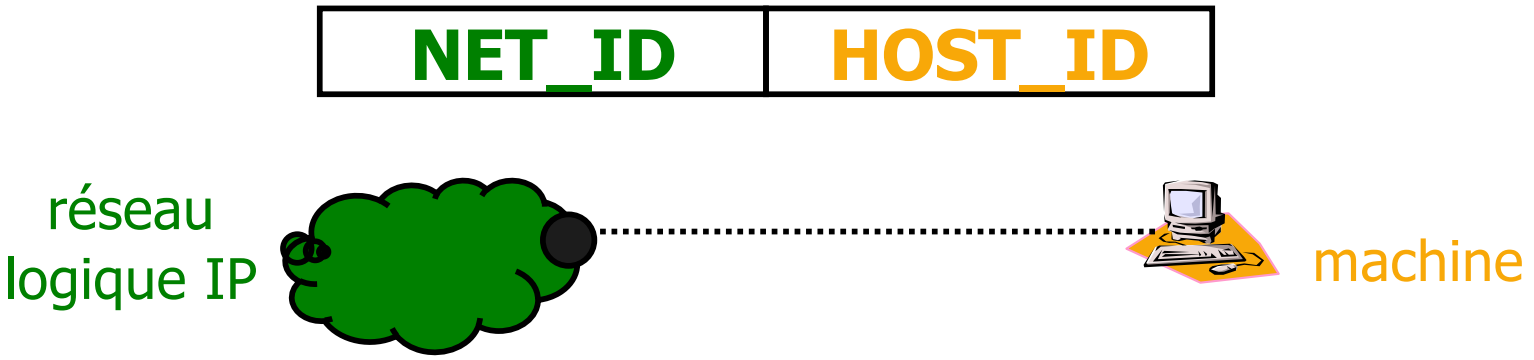



**Topologie logique**



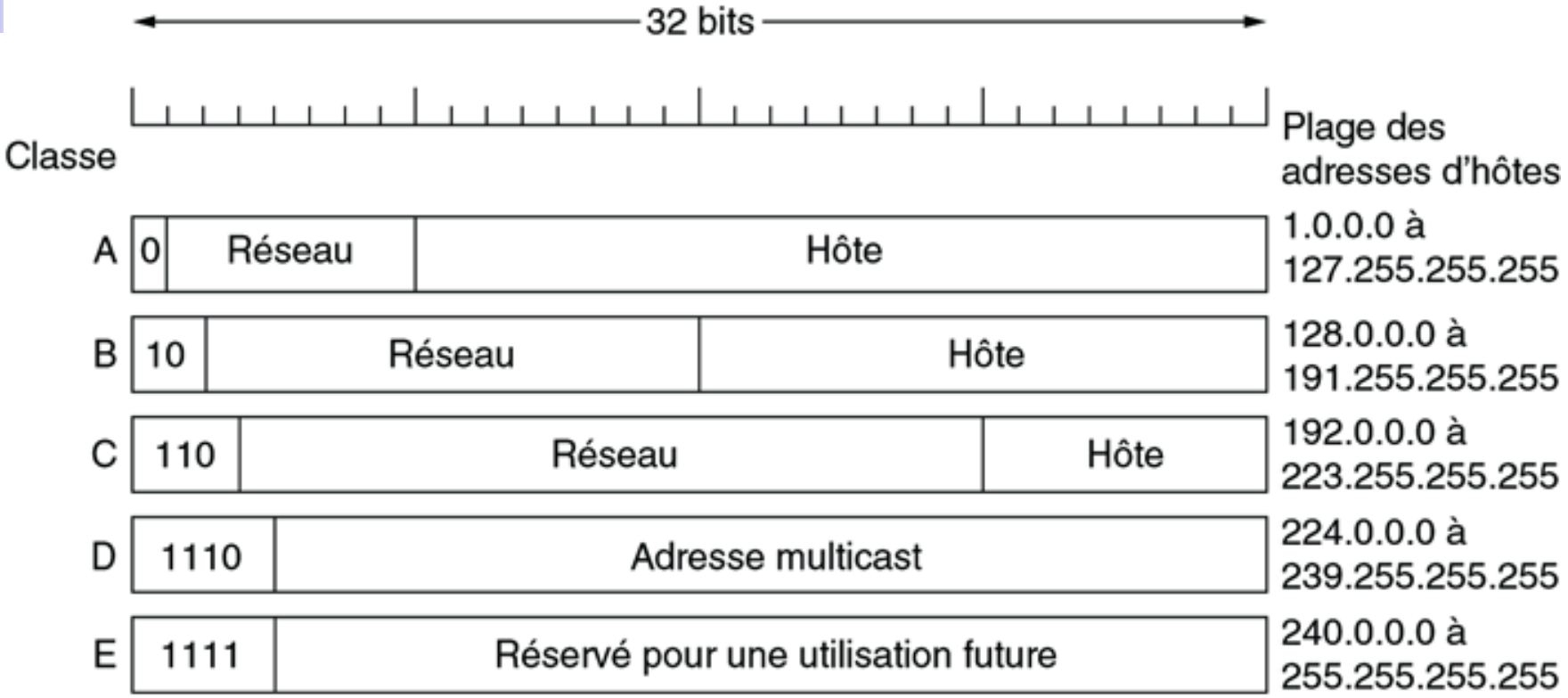
# Format de l'adresse IP

- L'internet se décompose en plusieurs réseaux logiques IP
- L'adresse IP est composée de deux champs
  - NET_ID : identifiant du réseau IP (utilisé pour le routage)
  - HOST_ID : identifiant de la machine dans le réseau IP




- Adresse IP = 32 bits = 4 octets (représentée par 4 valeurs décimales [0-255] séparées par un .)

# Les classes d'adressage

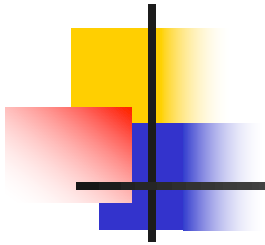





- Les adresses réseaux sont distribuées par un organisme international à but non lucratif : ICANN (Internet Corporation for Assigned Names and Numbers) puis décentralisé au niveau de chaque pays

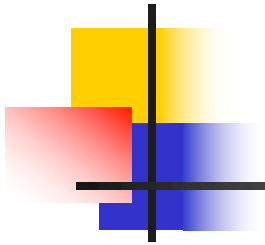
# Adresses IP particulières

- Diffusions locale et distante
  - 255.255.255.255 : adresse de broadcast sur le réseau IP local (ne passe pas le routeur, traduit en broadcast ARP)
  - <NET_ID><111…111> : adresse de broadcast dirigée vers le réseau de numéro NET_ID (exemple : 132.227.255.255 = diffusion dans le réseau 132.227.0.0 traduit en broadcast ARP par le routeur destination)
- Rebouclage local (*loopback*) : 127.x.y.z
  - généralement 127.0.0.1 (*localhost*)
  - permet de tester la pile TCP/IP locale sans passer par une interface matérielle
- l'adresse 0.0.0.0
  - utilisée par le protocole RARP (@IP de démarrage)
  - adresse de la route par défaut dans les routeurs



# Les adresses privées et le NAT (1)

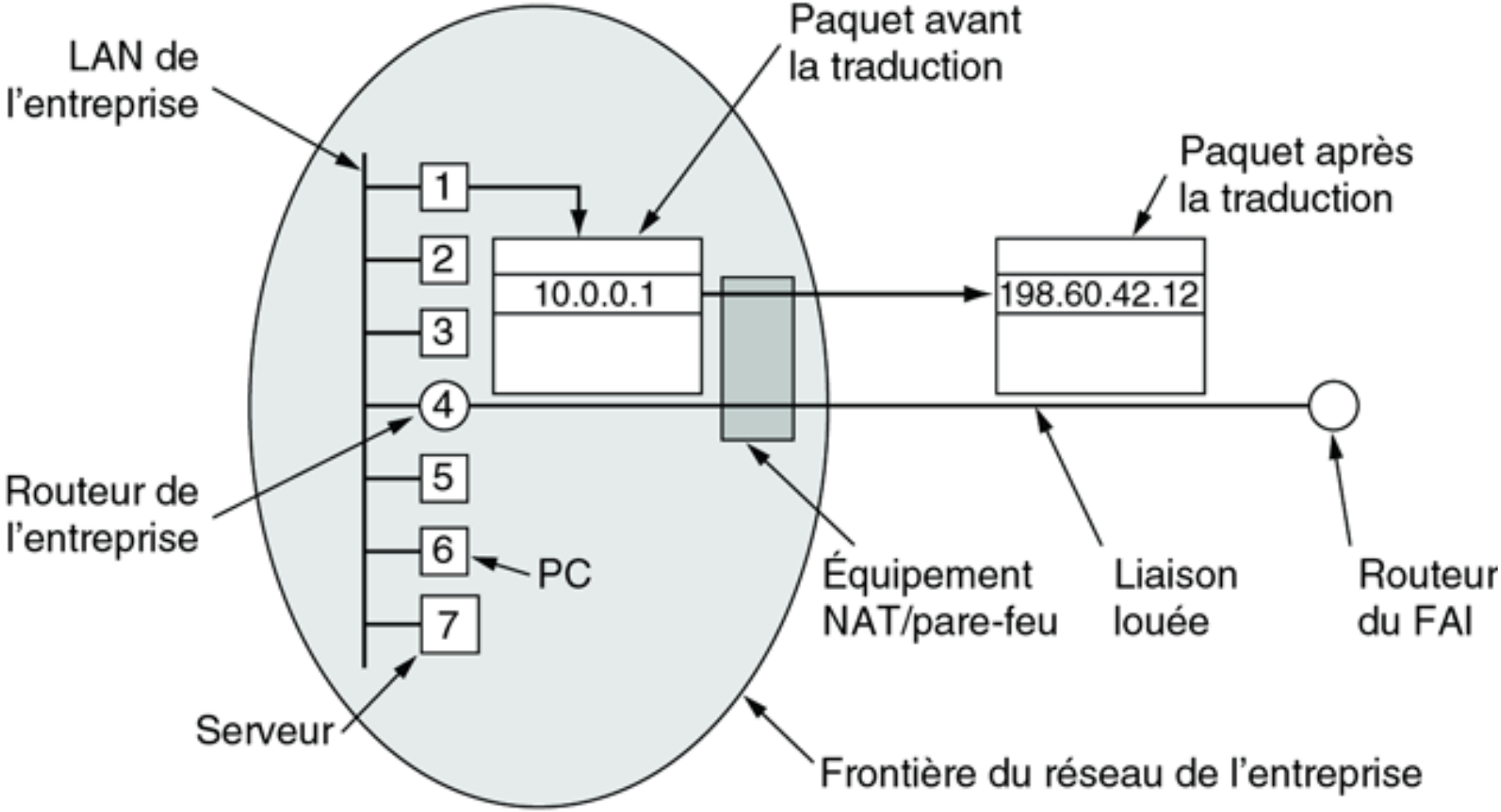
- Adresses privées (RFC 1918)
  - des adresses qui ne seront jamais attribuées (adresses illégales) et qui ne sont pas routables sur l'Internet
  - classe A : de 10.0.0.0 à 10.255.255.255
  - classe B : de 172.16.0.0 à 172.31.255.255
  - classe C : de 192.168.0.0 à 192.168.255.255
- Si une entreprise qui utilise des adresses privées souhaitent tout de même disposer d'une connexion à l'Internet, il faut
  - demander une adresse publique
  - faire des conversions adresse privée <--> adresse publique



# Les adresses privées et le NAT (2)

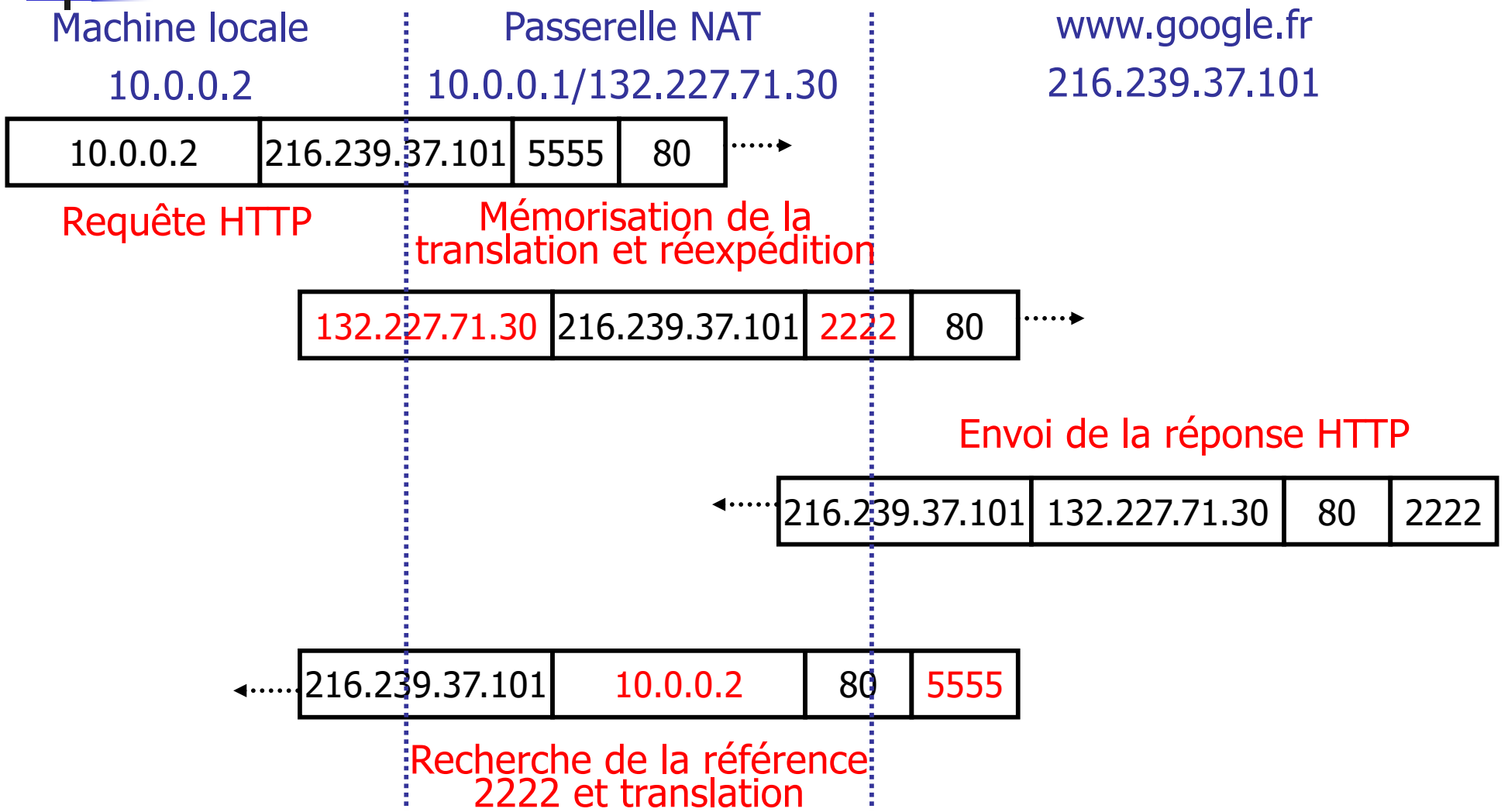
- NAT (RFC 3022) - Network Address Translator
  - mise en correspondance d'une adresse privée et d'une adresse publique
  - traduction statique ou dynamique (lors de la connexion)
  - une solution au manque d'adresses IP publiques : quelques adresses IP publiques pour beaucoup d'adresses IP privées mais le NAT est coûteux en perf.
- Fonctionnement du NAT
  - une table stockée dans le NAT fait la correspondance entre (@IP_src privée, port_src) et une @IP_publique
  - quand le paquet part : @IP_src devient @IP_publique, port_src devient la référence de l'entrée dans la table
  - quand la réponse revient : port_dest du paquet permet de retrouver dans la table @IP et port_src

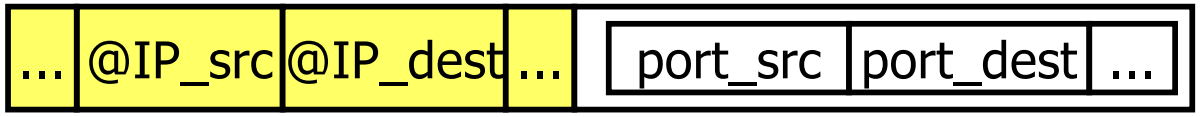
# Les adresses privées et le NAT (3)



# NAT - IP masquerading

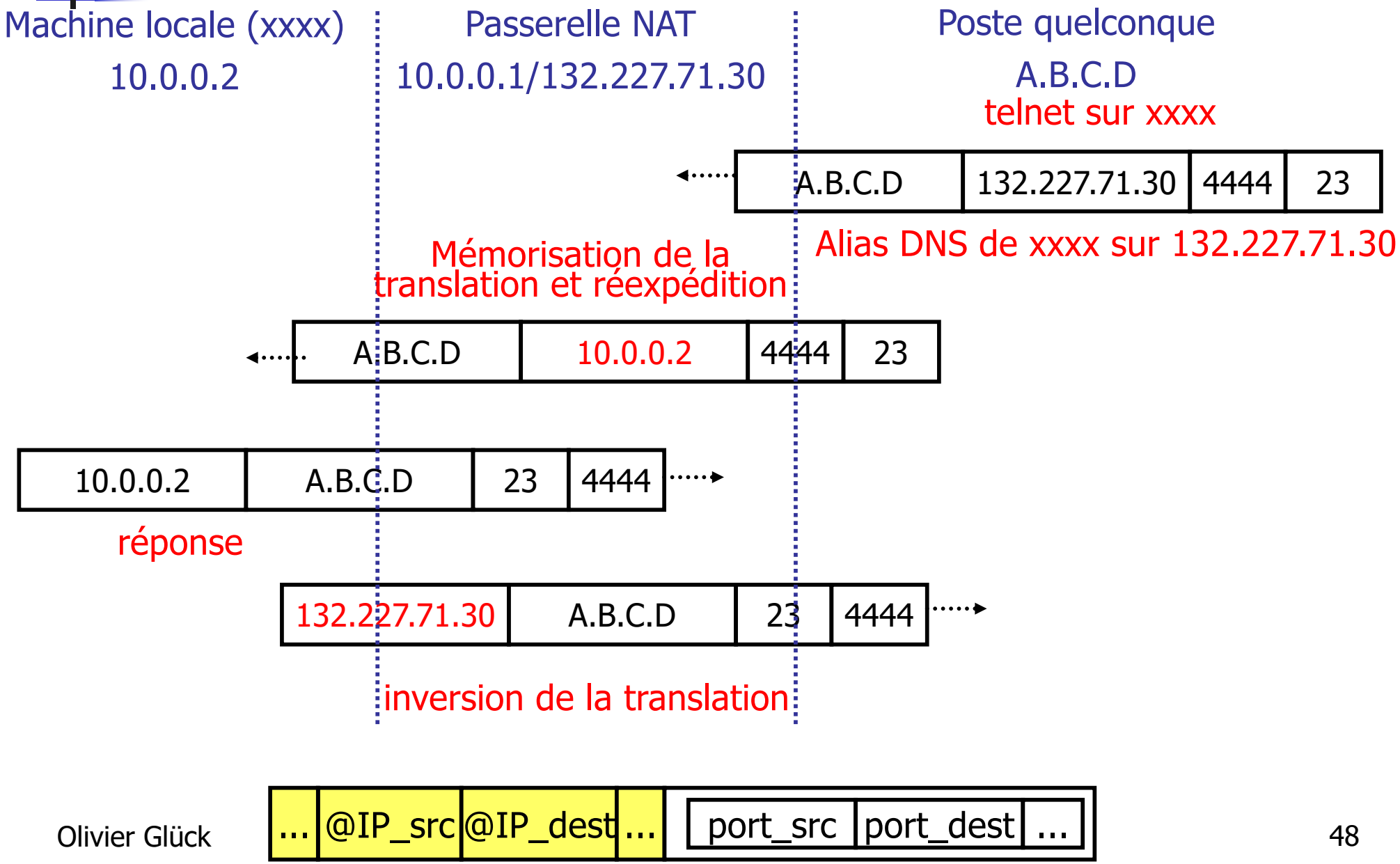
Exemple de requête **sortante**



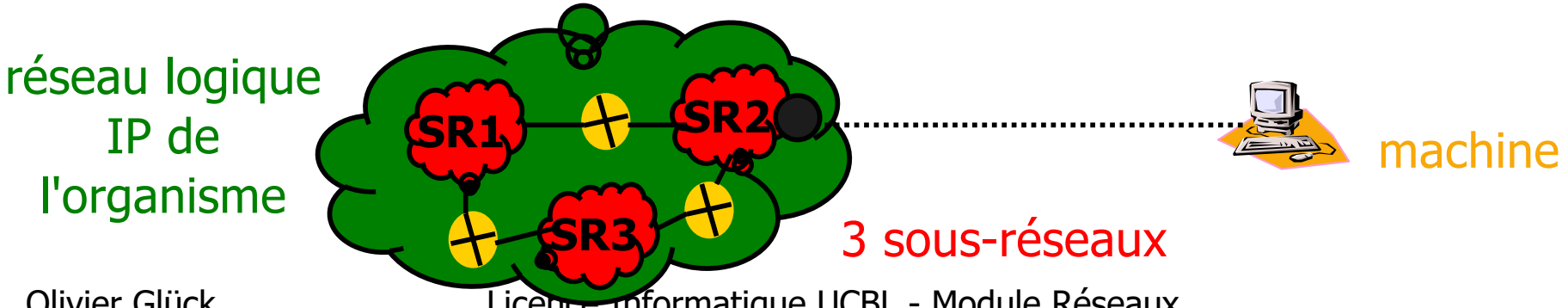
# NAT - port forwarding

Exemple de requête **entrante**



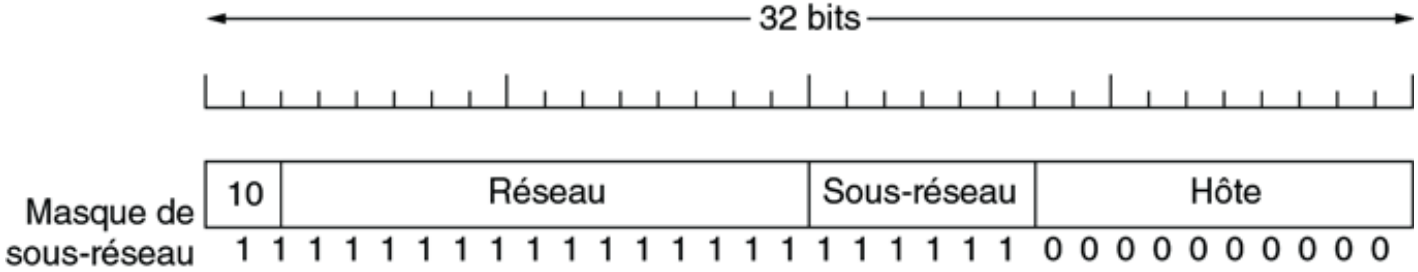
# Les sous-réseaux (1)

- Une organisation dispose généralement d'une seule adresse de réseau IP mais est composée de plusieurs sites/départements
  - -> diviser un réseau IP en plusieurs sous-réseaux
  - -> prendre quelques bits de la partie <HOST_ID> de l'adresse IP pour distinguer les sous-réseaux
  - -> transparent vis à vis de l'extérieur

# Les sous-réseaux (2)

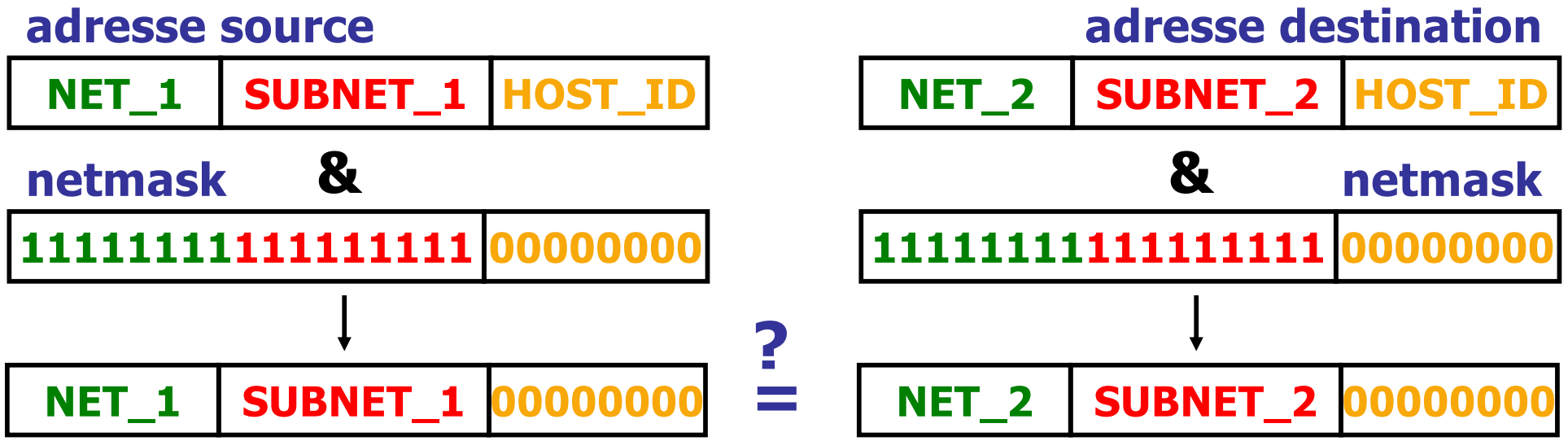
- Masque de sous-réseau (*Netmask*)
  - l'acheminement se fait en fonction de <NET_ID> et <SUBNET_ID> mais taille de <SUBNET_ID> inconnue
  - -> information donnée par le netmask : tous les bits à 1 correspondent à <NET_ID><SUBNET_ID>
- Exemple : 134.214.0.0 attribuée à l'UCBL
  - divisée en 64 sous-réseaux : 134.214.0.0, 134.214.4.0, 134.214.8.0, …, 134.214.248.0, 134.214.252.0
  - netmask = 255.255.252.0 = /8+8+6 = /22




Adresse de classe B dont 6 bits sont réservés à la numérotation des sous-réseaux

# Les sous-réseaux (3)

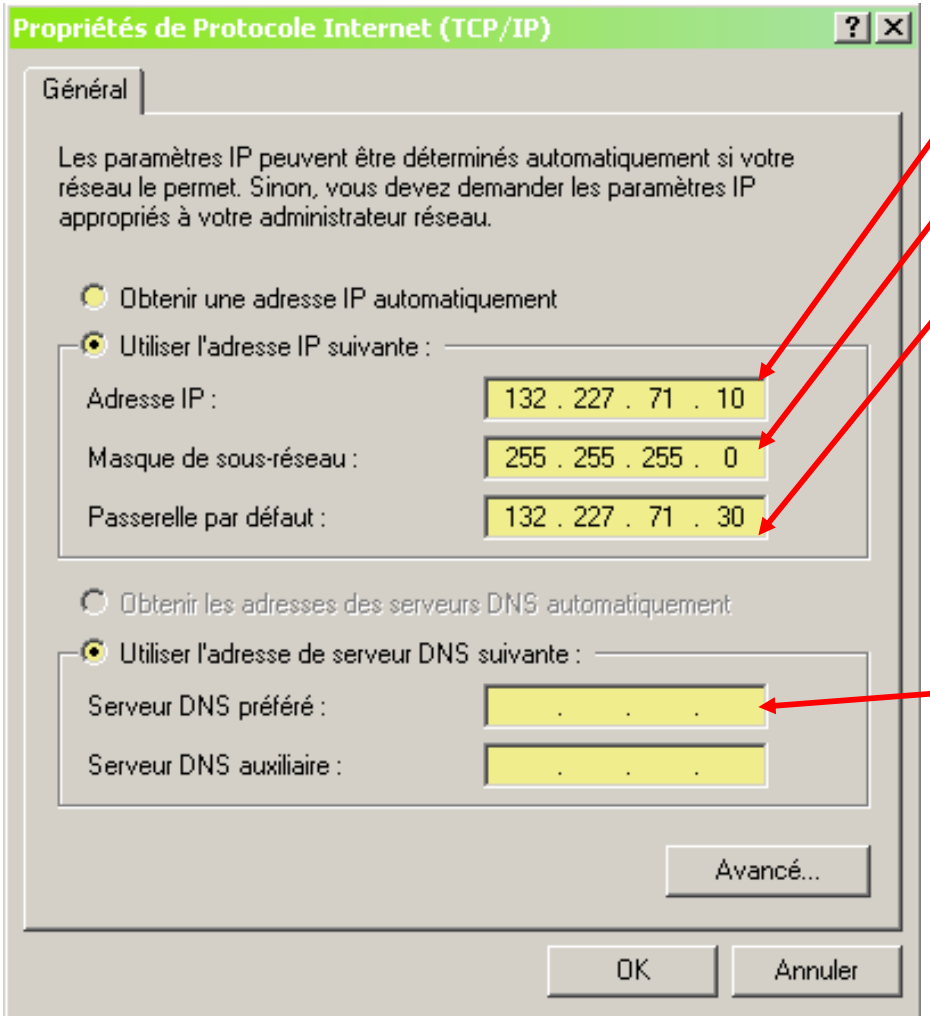
- Détermination du sous-réseau : ET logique avec le netmask




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

# Configuration réseau

- Pour une machine d'extrémité, il suffit d'indiquer




- son adresse IP
- le masque de sous-réseau
- l'adresse IP du routeur par défaut (tous les paquets qui ne sont pas à destination du même sous-réseau sont envoyés vers ce routeur)
- éventuellement, un serveur de noms

# L'adressage géographique - CIDR (1)

- Routage inter-domaine sans classe - *Classless InterDomain Routing* - RFC 1519, 1466
  - pour répondre (partiellement) aux problèmes de pénurie d'adresses de classe B et d'explosion des tables de routage
  - idée : allouer les adresses IP restantes sous la forme de blocs de taille variable (sans considération de classe) en tenant compte de la localisation géographique
  - -> évite le gaspillage : si un site a besoin de 2000 adresses, 2048 lui sont attribuées
  - -> agrégation de routes (plusieurs réseaux peuvent être regroupés sous le même identifiant)
  - -> les tables de routage doivent alors contenir un masque de sous-réseau pour l'acheminement (il n'y a pas de masque implicite indiqué par la classe)

# L'adressage géographique - CIDR (2)

- Exemple d'agrégation de 2 adresses de classe C :
  - une entreprise a besoin de 510 adresses IP -> deux adresses de classe C

    193.127.32.0 netmask 255.255.255.0

    193.127.33.0 netmask 255.255.255.0
  - les réseaux 193.127.32.0 et 193.127.33.0 sont agrégés en 193.127.32.0 netmask 255.255.254.0
  - ce qui se note 193.127.32.0/24 + 193.127.33.0/24 = 193.127.32.0/23 (préfixe/nb_bits du masque à 1)
  - dans une table de routage, cela représente les deux réseaux 193.127.32.0 et 193.127.33.0

| | |
|---|---|
| 193.127.32.0 | 11000001.01111111.00100000.00000000 |
| 193.127.33.0 | 11000001.01111111.00100001.00000000 |
| 193.127.32.0 / 23 | 11000001.01111111.0010000 0.00000000 |

# L'adressage géographique - CIDR (3)

- Allocation géographique des adresses restantes

  Europe (194-195), Amérique du nord (198-199), Amérique du sud (200-201), Pacifique (202-203), Afrique (99-?)

  -> 194 et 195 ont les 7 premiers bits identiques donc il suffit d'indiquer aux routeurs (hors Europe) : 194.0.0.0/7

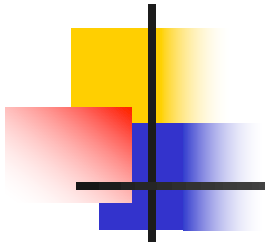
- Autres exemples (source L. Toutain)

| société | nb d'adresses | nb de classe C | adresse de début | adresse de fin | adresse de début | netmask | netmask en binaire | |
|---|---|---|---|---|---|---|---|---|
| A | < 2048 | 8 | 192.24.0 | 192.24.7 | 192.24.0 | 255.255.248.0 | 255.255. 1111 1000 .0 | 192.24.0.0/21 |
| B | < 4096 | 16 | 192.24.16 | 192.24.31 | 192.24.16 | 255.255.240.0 | 255.255. 1111 0000 .0 | 192.24.16.0/20 |
| C | < 1024 | 4 | 192.24.8 | 192.24.11 | 192.24.8 | 255.255.252.0 | 255.255. 1111 1100 .0 | 192.24.8.0/22 |
| D | < 1024 | 4 | 192.24.12 | 192.24.15 | 192.24.12 | 255.255.252.0 | 255.255. 1111 1100 .0 | 192.24.12.0/22 |
| E | < 512 | 2 | 192.24.32 | 192.24.33 | 192.24.32 | 255.255.254.0 | 255.255. 1111 1110 .0 | 192.24.32.0/23 |
| F | < 512 | 2 | 192.24.34 | 192.24.35 | 192.24.34 | 255.255.254.0 | 255.255. 1111 1110 .0 | |

# L'adressage géographique - CIDR (4)

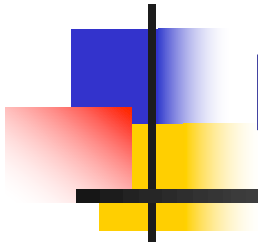
- Conclusions
  - il n'y a plus de notion de classes et de sous-réseaux
  - une plage d'adresses est désignée par
    - un "*network-prefix*" : des bits désignants le réseau
    - un "*host-number*" : des bits désignants la machine

  --> un réseau est désigné par une adresse IP et une longueur de préfixe réseau
    - 132.227.0.0 n'a pas de sens
    - 132.227.0.0/16 ou 132.227.0.0/23 ont un sens
  - une table de routage peut contenir les deux destinations précédentes : la route avec le préfixe le plus long ("*the longest matching network prefix*") est choisie si une destination correspond aux deux entrées (ici 132.227.0.0/23)



# Numérotation des sous-réseaux

- Peut-on mettre dans SUBNET_ID tous les bits à 1 ou tous les bits à 0 ?
  - exemple : 10.0.0.0 avec *netmask* 255.192.0.0 (2 bits pour numéroter les sous-réseaux)
    --> 10.0.0.0, 10.64.0.0, 10.128.0.0, 10.192.0.0
- La RFC 950 (1985 - définition des SR) dit que cela n'est pas conseillé car
  - 10.0.0.0 désigne t-il le réseau 10.0.0.0 ou le sous-réseau ?
  - 10.255.255.255 désigne t-il le *broadcast* sur le réseau 10.0.0.0 ou sur le sous-réseau 10.192.0.0 ?
- Il n'y a plus d'ambiguïté avec CIDR (RFC 1812 -1995)
  - 10.0.0.0/8 et 10.0.0.0/10 ne désignent pas la même chose
- En pratique, on peut utiliser 10.0.0.0 et 10.192.0.0 !



# Le protocole IP (IPv4)

Datagramme IPv4

Fragmentation dans IP

Routage dans IP

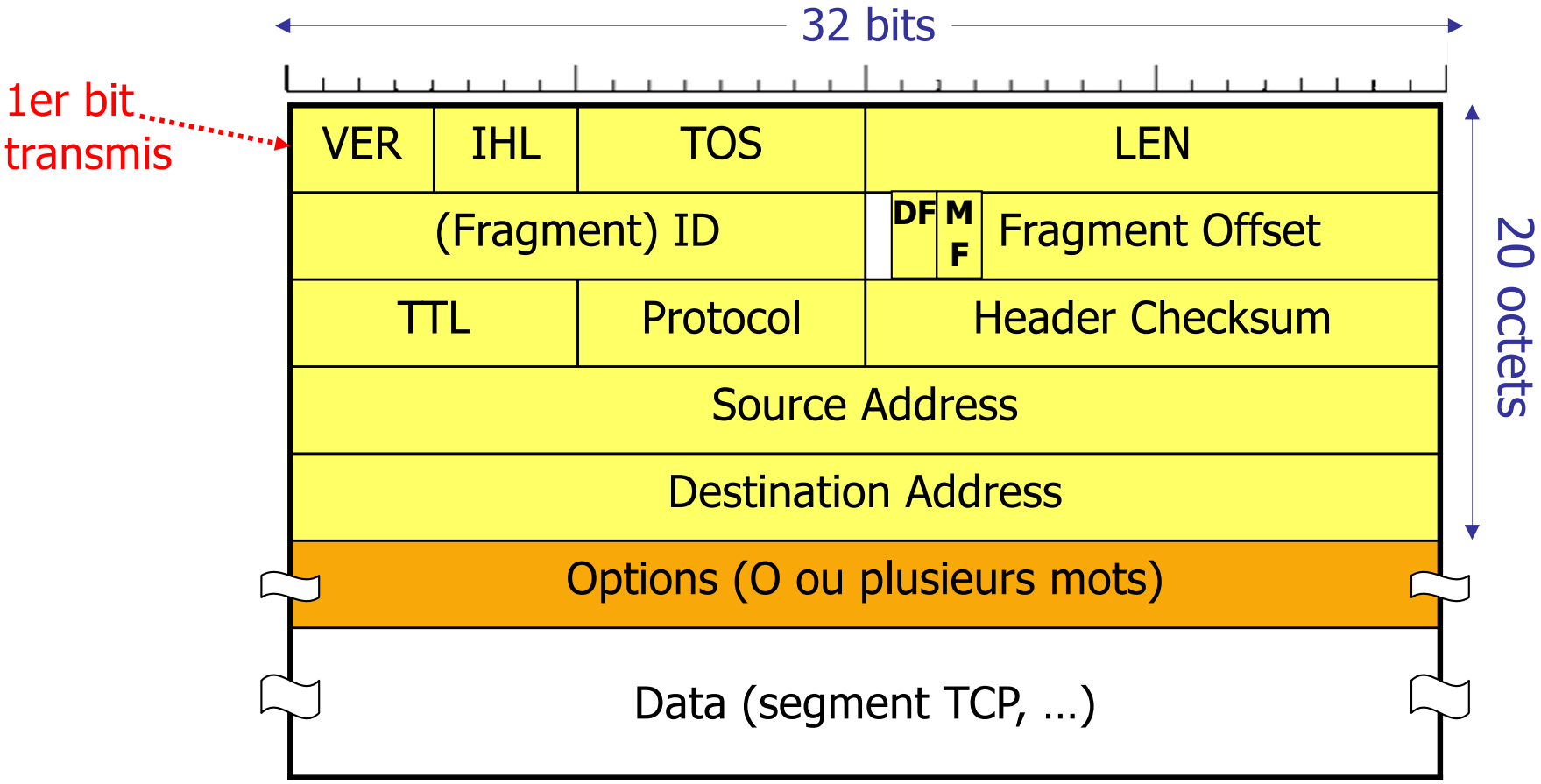
Routage statique

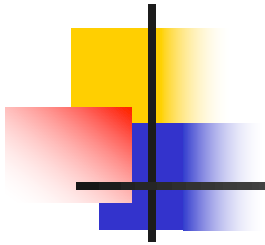
# L'Internet Protocol

- IP (RFC 791) : protocole d'interconnexion de l'Internet
  - conçu pour transporter des datagrammes d'une certaine source A vers une destination B
  - A et B peuvent être sur le même réseau ou séparés par d'autres réseaux de nature très différentes
  - livraison au mieux - **best-effort delivery** : aucune garantie quant au service d'acheminement (délai, taux de perte, ...), aucune variable d'état
  - IP n'accomplit que trois tâches élémentaires :
    - adaptation des datagrammes IP à la MTU du réseau physique traversé
    - acheminement dans le réseau logique
    - désignation des nœuds (adressage IP)

# Le datagramme IPv4 (1)

- Un en-tête de 20 octets + une partie facultative de longueur variable (options)

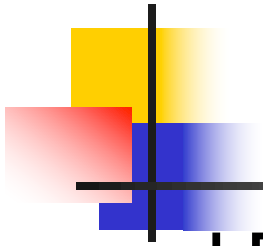




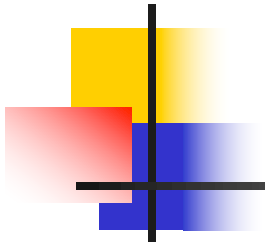
# Le datagramme IPv4 (2)

- VER - Version - 4 bits
  - numéro de version d'IP (généralement 4 pour IPv4)
  - permet de faire cohabiter plusieurs versions (transition)
- IHL - Internet Head Length - 4 bits
  - longueur de l'en-tête du datagramme (en nombre de mots de 32 bits, 4 octets) -> 5 si pas d'option
  - valeur maximale = 15 -> 40 octets d'options maximum
- TOS - Type Of Service - 6+2 bits
  - pour distinguer différentes classes de services (niveaux de priorités) -> compromis entre fiabilité, délai et débit
  - champ ignoré par la plupart des routeurs

# Le datagramme IPv4 (3)

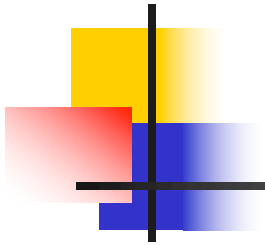


- LEN - total LENgth field - 16 bits
  - longueur totale du datagramme en octets
  - au maximum 65535 octets
- ID - Identification - 16 bits
  - identifiant de datagramme (ou paquet)
  - tous les fragments d'un même paquet ont le même ID
- DF (1bit) et MF (1 bit)
  - DF - Don't Fragment : ordre au routeur de ne pas fragmenter (autre route ou destruction)
  - MF - More Fragment : indique qu'un fragment suit
- Fragment Offset - 13 bits
  - position du premier bit du fragment dans le datagramme d'origine, en multiple de 8 octets



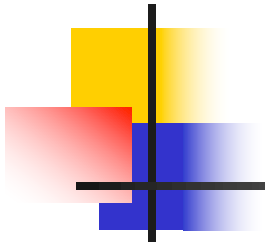
# Le datagramme IPv4 (4)

- TTL - Time To Live - 8 bits
  - compteur qui sert à limiter la durée de vie du datagramme
  - 255 au départ puis décrémenté à chaque nouveau saut
  - datagramme éliminé s'il atteint zéro
  - évite les paquets perdus (erreurs de routage)
- Protocol - 8 bits
  - numéro du protocole destinataire (RFC 1700)
- Header Checksum - 16 bits
  - CRC sur l'en-tête uniquement
  - complément à 1 de la somme des demi-mots de 16 bits
  - doit être recalculé dès qu'une valeur change (ex. TTL) !



# Le datagramme IPv4 (5)

- Le champ Options
  - prévu pour des expérimentations mais peu utilisé dans la pratique
  - codé : <code option (1 octet)>,<longueur (1 octet)>, <données>
  - longueur variable, plusieurs options possibles
  - exemples d'options :
    - sécurité : degré de confidentialité du datagramme (route plus sécurisée que d'autres !)
    - routage strict par la source : suite d'@ IP décrivant le chemin pour atteindre la destination
    - enregistrement de route : les routeurs traversés insèrent chacun leur @IP



# La fragmentation des datagrammes IP

- Caractéristiques :
  - fragmentation non-transparente : réassemblage uniquement sur le destinataire
  - chaque fragment est acheminé de manière indépendante
  - temporisateur de réassemblage sur le destinataire quand le premier fragment arrive (décrémentation de TTL)
  - la perte d'un fragment IP provoque la retransmission de l'ensemble du datagramme

  S' il y a une perte, elle ne sera détectée qu' au niveau TCP où la notion de fragments n' existe pas

  Un routeur IP ne s' encombre pas de fragments qu' il ne peut réassembler

# La fragmentation des datagrammes IP

http://wps.aw.com/aw_kurose_network_2/0,7240,227091-,00.html.

## Exemple (valeurs en décimal) :

MTU de 128 octets (soit 108 octets de données IP par fragment), l'offset devant être un multiple de 8 octets -> 13*8=104 octets

Datagramme origine

| 4 | 5 | 00 | LEN=**368** |
|---|---|---|---|
| ID=11425 | | 0 0 | Offset=**0** |
| TTL | Pro=6 | Checksum | |
| Source Address | | | |
| Destination Address | | | |
| Data (348 octets) | | | |

F1

| 4 | 5 | 00 | LEN=**124** |
|---|---|---|---|
| ID=11425 | | 0 1 | Offset=**0** |
| TTL | Pro=6 | Checksum | |
| Source Address | | | |
| Destination Address | | | |
| Data (104 octets) | | | |

F2

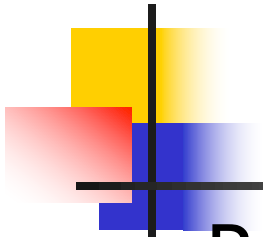
| 4 | 5 | 00 | LEN=**124** |
|---|---|---|---|
| ID=11425 | | 0 1 | Offset=**13** |
| TTL | Pro=6 | Checksum | |
| Source Address | | | |
| Destination Address | | | |
| Data (104 octets) | | | |

F3

| 4 | 5 | 00 | LEN=**124** |
|---|---|---|---|
| ID=11425 | | 0 1 | Offset=**26** |
| TTL | Pro=6 | Checksum | |
| Source Address | | | |
| Destination Address | | | |
| Data (104 octets) | | | |

F4

| 4 | 5 | 00 | LEN=**56** |
|---|---|---|---|
| ID=11425 | | 0 0 | Offset=**39** |
| TTL | Pro=6 | Checksum | |
| Source Address | | | |
| Destination Address | | | |
| Data (36 octets) | | | |



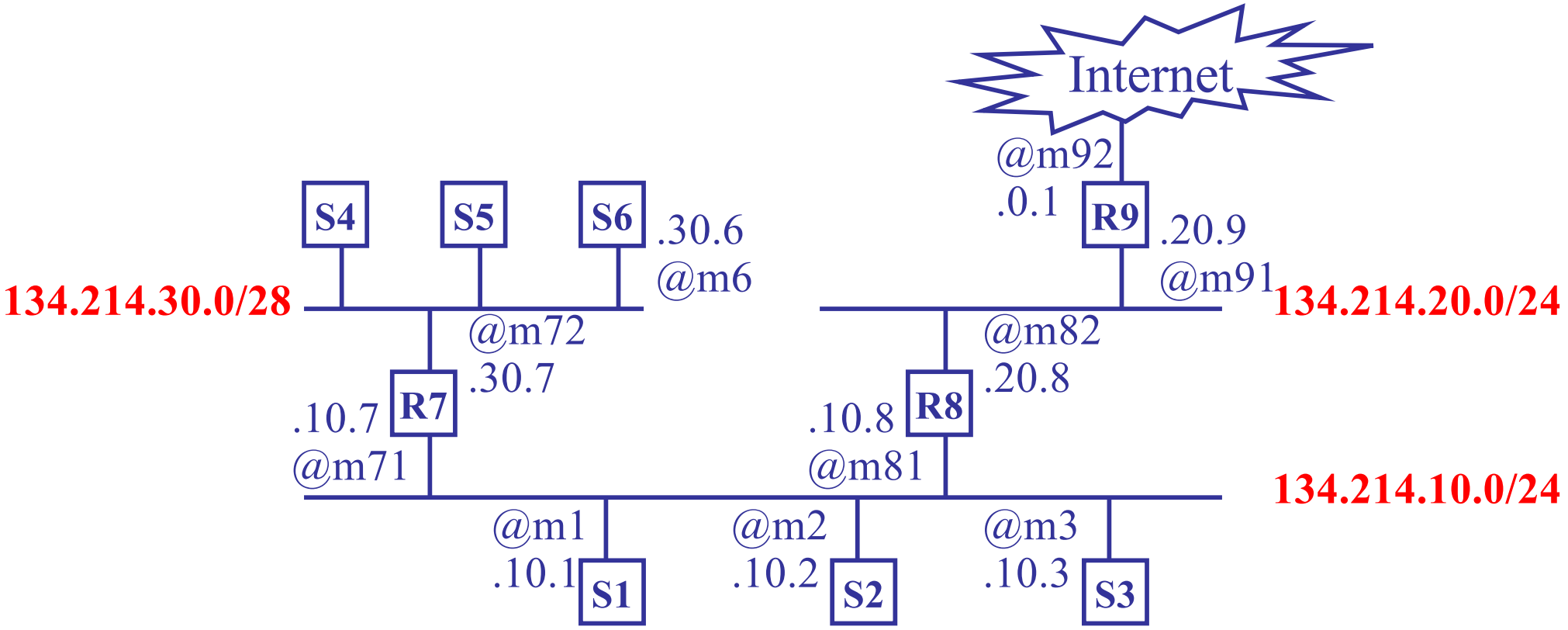
# Le routage dans IP

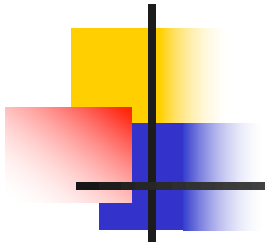
- Routeur :
  - passerelle entre sous-réseaux
  - une adresse IP par interface (par sous-réseau)
  - communications à l'intérieur d'un même sous-réseau sans passer par un routeur
  - acheminement à partir de l'@ destination (& logique avec le netmask de chaque entrée de la table de routage)
- Mise à jour de la table de routage :
  - Manuelle = **routage statique**
    - commande "route" des stations unix
    - langage de commande des routeurs (ip route ...)
  - Automatique = **routage dynamique**
    - processus sur les stations et les routeurs
    - échanges d'informations de routage : protocoles de routage

# Le routage dans IP - exemple

Table de routage de S2

| destination | netmask | gateway | int | cost |
|---|---|---|---|---|
| 134.214.10.0 | 255.255.255.0 | - | eth0 | 0 |
| 134.214.30.0 | 255.255.255.240 | 134.214.10.7 | - | 1 |
| default | 0.0.0.0 | 134.214.10.8 | - | - |



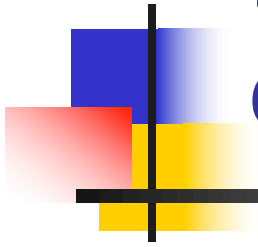


# Routage statique

- La commande **`ip route`** permet d'indiquer une route :
  - vers un réseau (net) ou vers un équipement (host)
  - ou une route par défaut (default)
- Syntaxe :

```
ip route add |delete [net|host]
  destination |default gateway metric
```

- En général, sur les équipements non routeur, on définit uniquement une route par défaut



# Protocoles de contrôle de l'Internet et utilitaires réseaux

ICMP

ping et traceroute

ARP et RARP

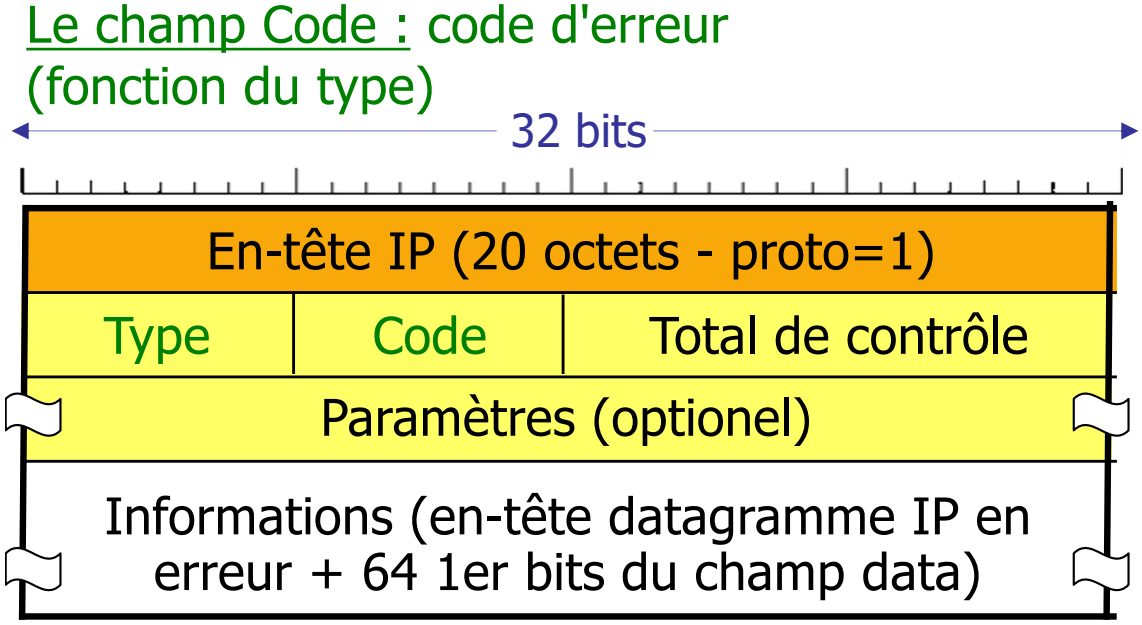
BOOTP et DHCP

Fichiers de config. et commandes UNIX

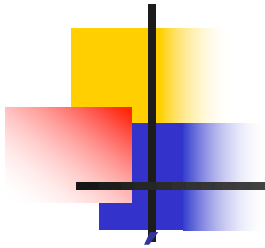
# ICMP - Internet Control Message Protocol

RFC 792

- Protocole de messages de contrôle de l'Internet
  - échange de messages entre routeurs : signaler une erreur réseau, demande d'information d'état, tests
  - utilisé par des utilitaires (*ping*, *traceroute*, *Network Time Protocol*)
  - permet de pallier au manque de service d'IP

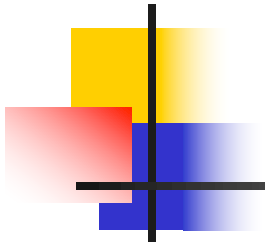



Le champ Type :
0 : réponse d'Echo
3 : destination inconnue
4 : limitation du débit par la source
5 : redirection (ICMP redirect)
8 : demande d'Echo
11 : expiration de délai (TTL=0)
12 : en-tête IP invalide
13/14 : requête/réponse d'horodatage
17/18 : requête/réponse de netmask



# ICMP - Types de message

- réponse/demande d'Echo : utilisé par *ping*
- réponse/demande d'horodate : idem mais heures incluses pour mesures de performances
- destination inconnue : un routeur ne parvient pas à localiser la destination, problème de fragmentation (bit DF=1), ...
- délai expiré : paquet éliminé car TTL a atteint 0 (boucle, congestion, ...)
- en-tête IP invalide : la valeur d'un champ IP a une valeur illégale
- ICMP redirect : envoyé par un routeur à un nœud d'extrémité pour signaler une meilleure route (évite la mise à jour manuelle de toutes les tables de routage quand ajout d'un routeur...)
- ralentissement de la source : contrôle de congestion (mais quasiment plus utilisé car génère du trafic supplémentaire -> congestion au niveau TCP)
- autres messages : *www.iana.org/assignments/icmp-parameters*



# L'utilitaire ping

- Ping : envoi d'un écho, attente de réponse, mesure du temps aller-retour
  - teste l'accessibilité d'une destination **de bout en bout**
  - évaluation de performances
  - la réponse doit parvenir avant 20 secondes
- Exemples :

  `ping 127.0.0.1` : permet de tester la pile TCP/IP locale (en loopback)

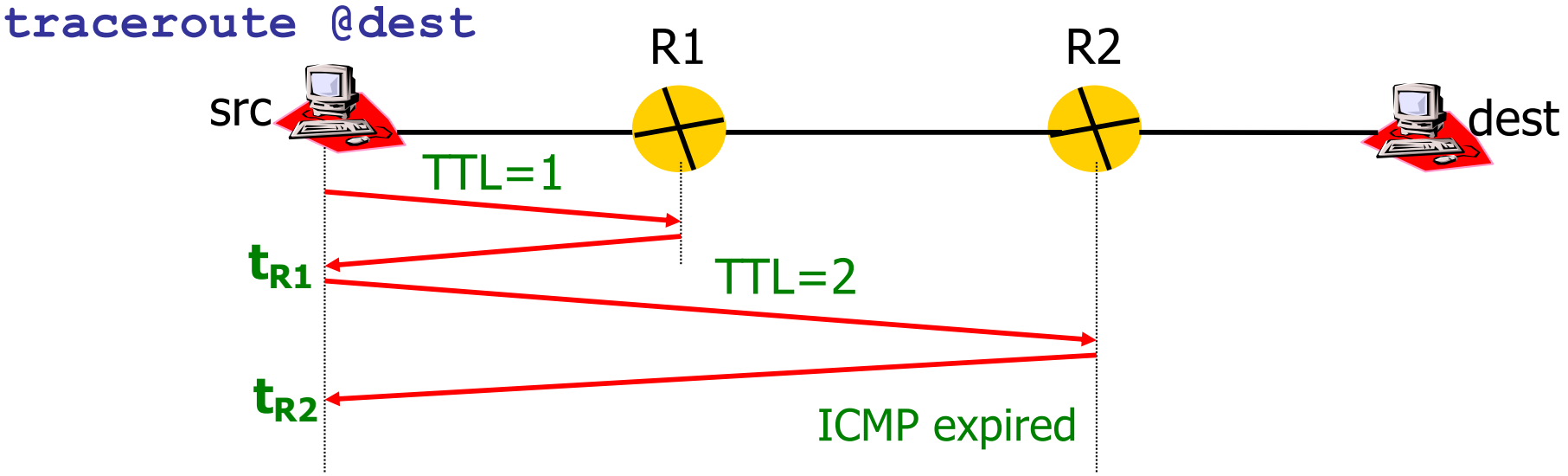
  `ping mon@IP` : permet de vérifier la configuration réseau locale de la station

  `ping @default-routeur` : permet de tester la configuration du sous-réseau et de la passerelle

  `ping @dest` : permet de tester un chemin de bout en bout

# L'utilitaire traceroute

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



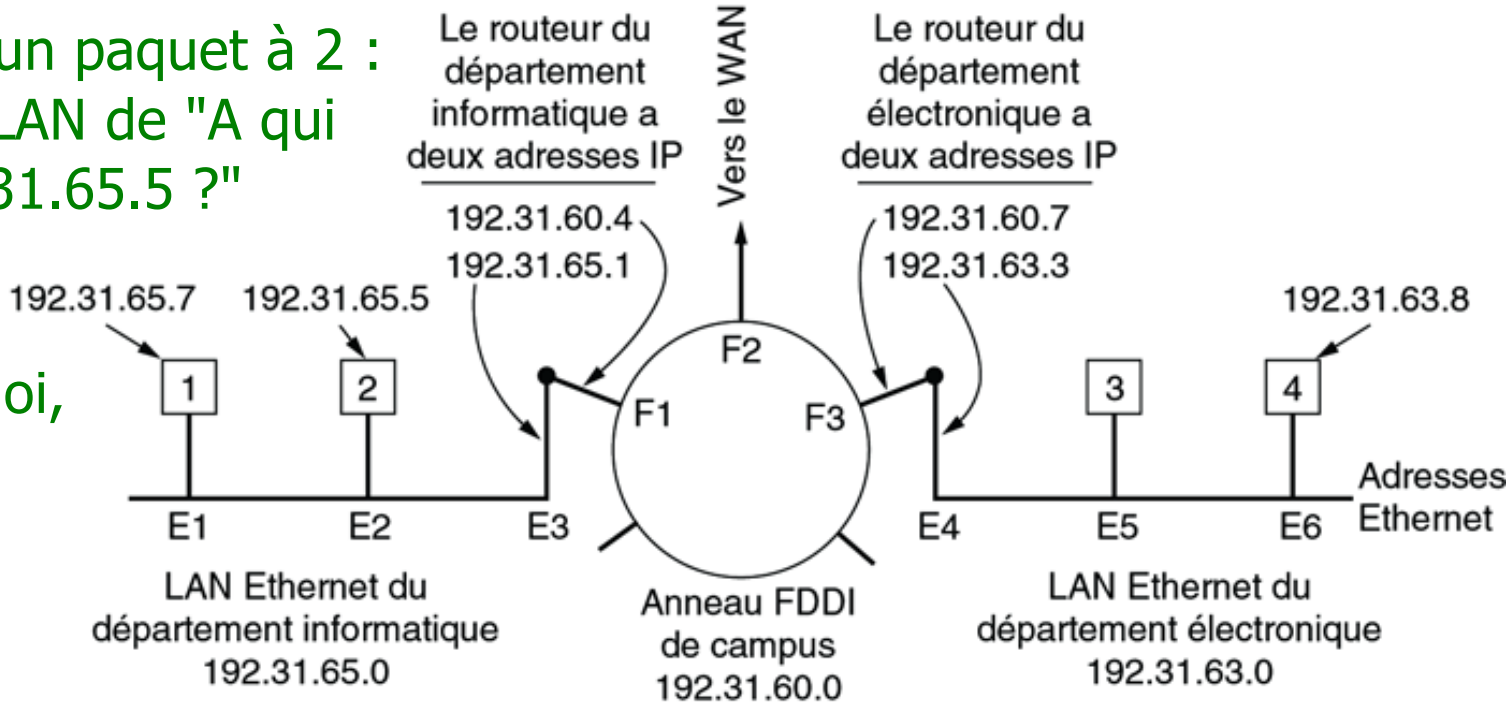
# ARP - Address Resolution Protocol

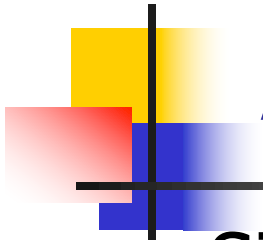
RFC 826

- Problème : les équipements de liaison (cartes réseau…) ne comprennent pas les adresses IP mais utilisent des adresses physiques (MAC)
- Besoin d'associer @MAC <--> @IP

1 veut envoyer un paquet à 2 : diffusion sur le LAN de "A qui appartient 192.31.65.5 ?"

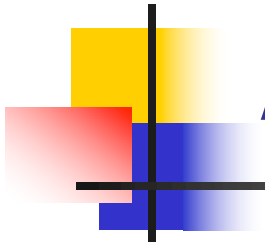
2 répond : "A moi, je suis E2"





# ARP - Fonctionnement (1)

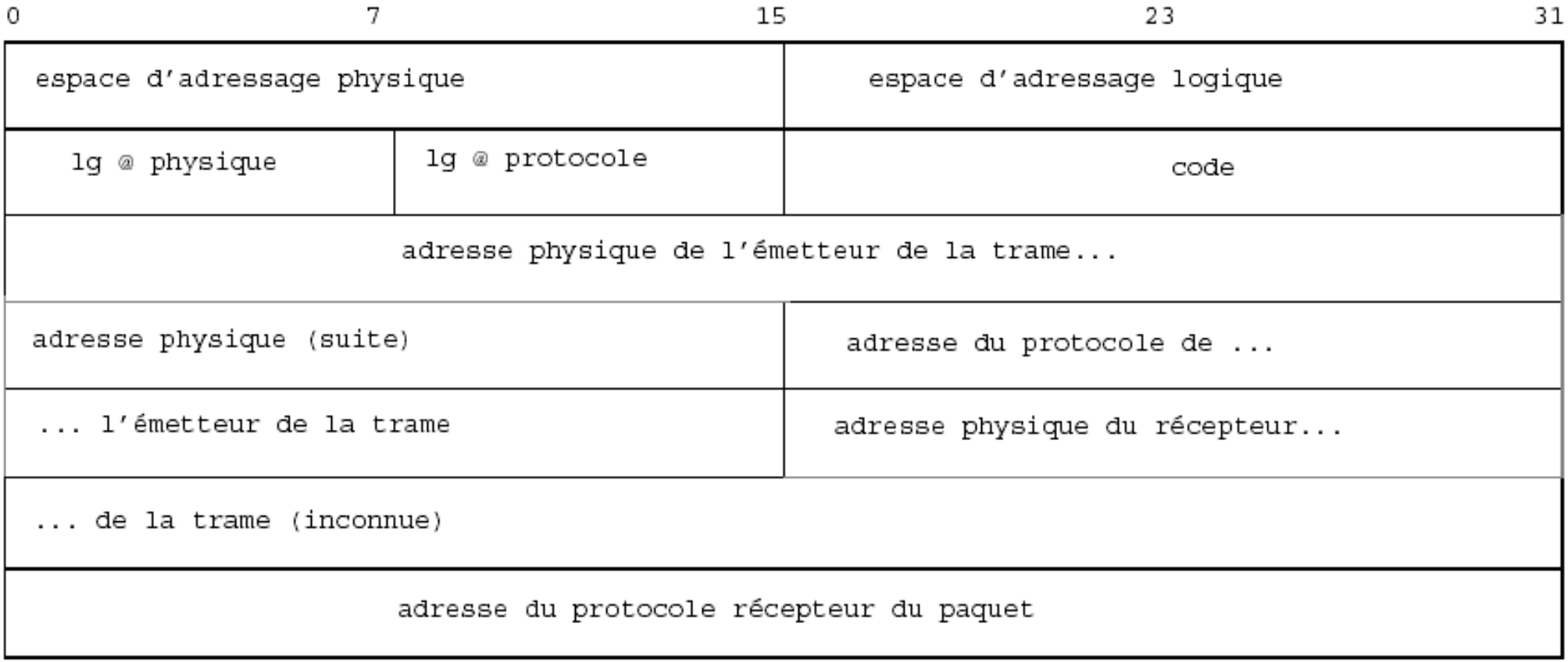
- Si la machine source et destinataire sont sur le même réseau (par ex. de 1 vers 2)
  - 1 - requête ARP (broadcast MAC)
  - 2 - réponse ARP (le destinataire a reçu le broadcast et s'est reconnu, il envoie son @MAC)
  - 3 - la source peut envoyer ses données vers le destinataire (adresse MAC destination connue)
- Si elles ne sont pas sur le même réseau (par ex. de 1 vers 4)
  - la diffusion ne passe pas le routeur
  - résolution de proche en proche : 1 envoie les données à 192.31.65.1 (ARP pour trouver E3), le routeur info envoie les données à 192.31.60.7 (ARP pour trouver F3), le routeur élec envoie les données à 4 (ARP pour E6)



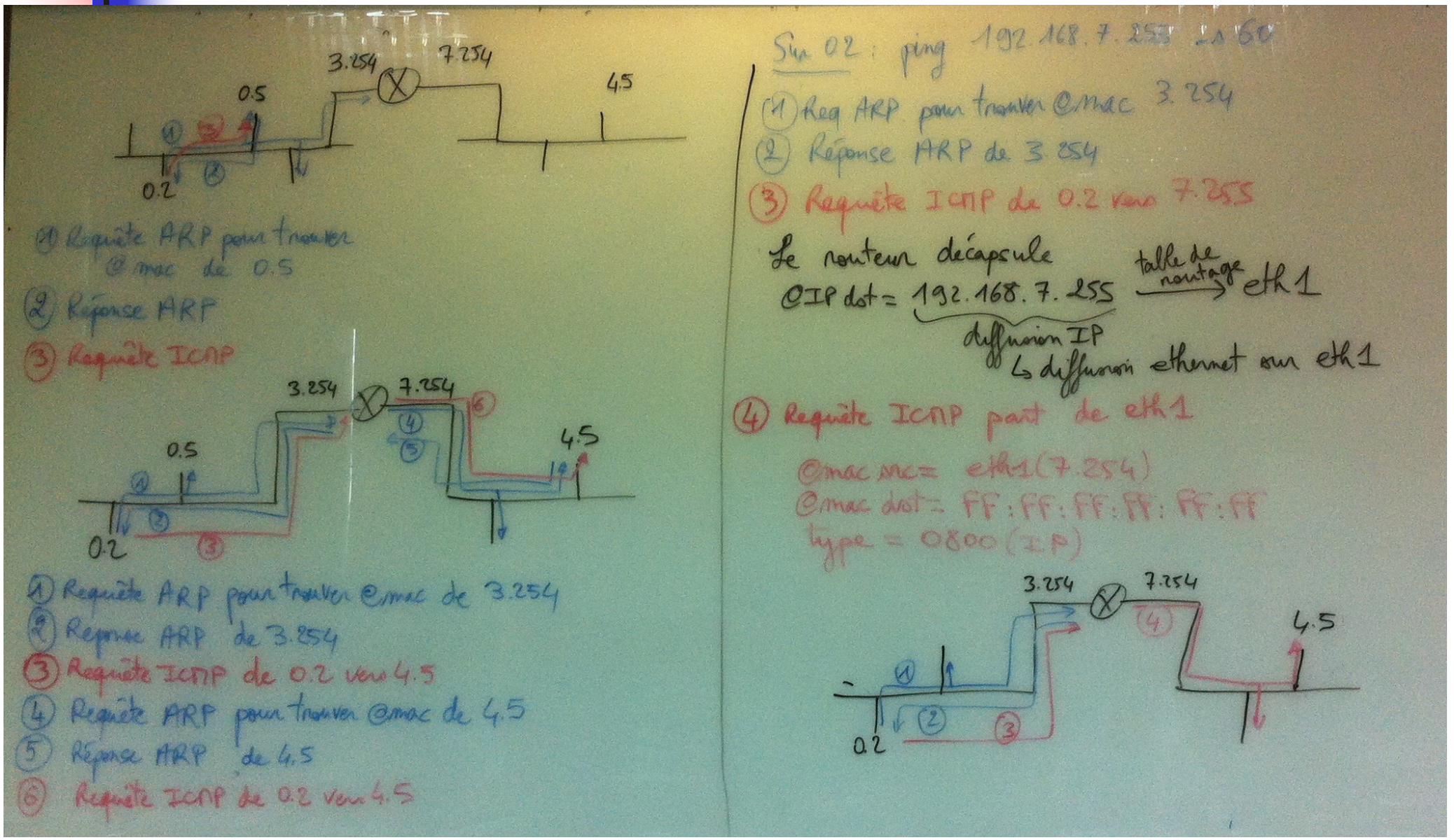
# ARP - Fonctionnement (2)

- Optimisations
  - Cache ARP : le résultat de chaque résolution est conservé localement pour les émissions suivantes
  - la correspondance (@IP, @MAC) de l'émetteur sont inclus dans la requête ARP pour que le récepteur, voire toutes les machines qui reçoivent le broadcast, mettent à jour leur cache
- Proxy ARP : une machine qui répond à une requête à la place du destinataire (qui ne reçoit pas le broadcast)
  - nécessaire si la route (adresse de la passerelle) pour atteindre le destinataire n'est pas connue

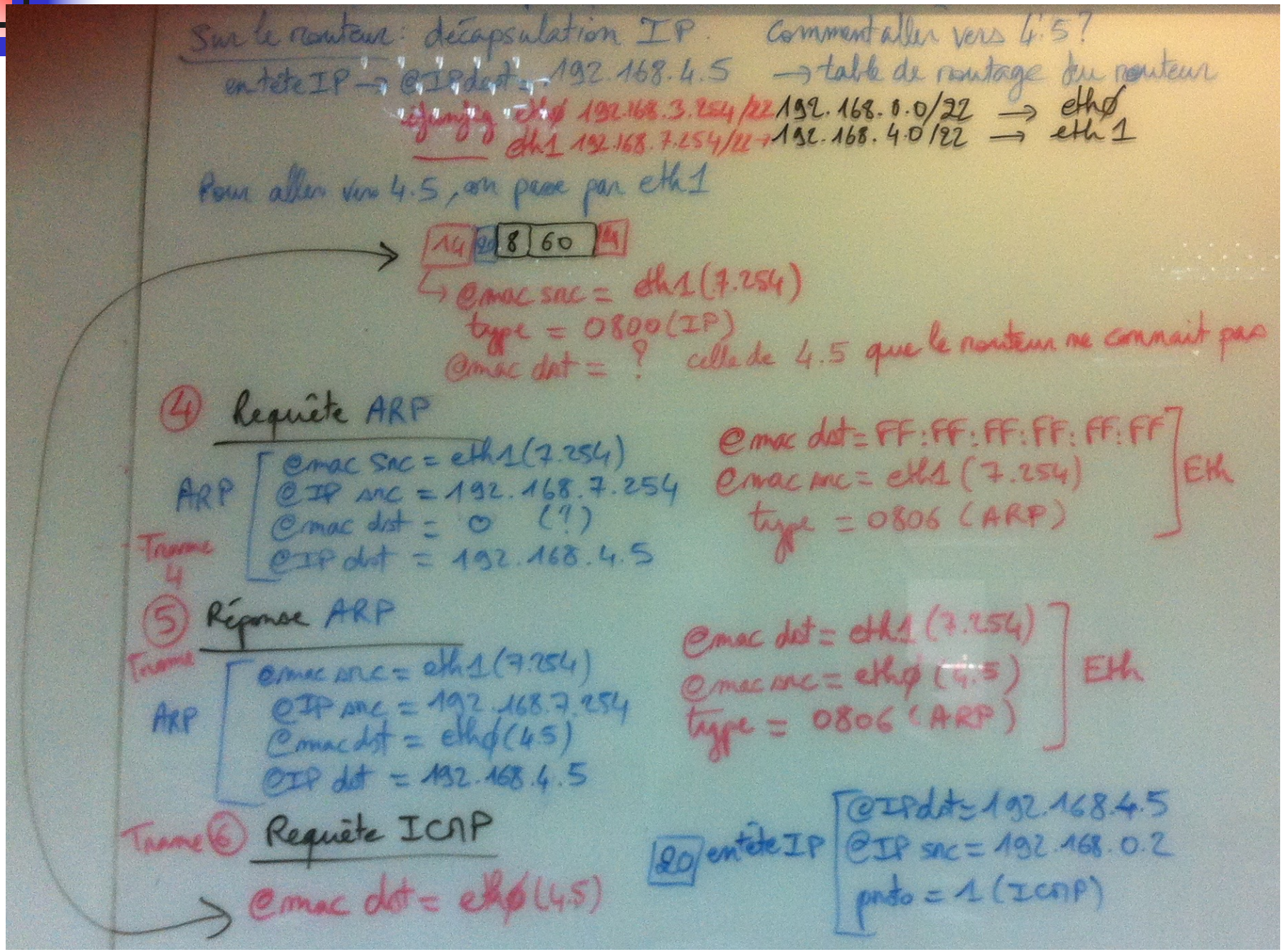
# ARP - Format du paquet



# ARP - ICMP


3.254
7.254
0.5
4.5
0.2
① Requête ARP pour trouver @mac de 0.5
② Réponse ARP
③ Requête ICMP
3.254
7.254
0.5
4.5
0.2
① Requête ARP pour trouver @mac de 3.254
② Réponse ARP de 3.254
③ Requête ICMP de 0.2 vers 4.5
④ Requête ARP pour trouver @mac de 4.5
⑤ Réponse ARP de 4.5
⑥ Requête ICMP de 0.2 vers 4.5
Sur 0.2 : ping 192.168.7.255 -s 60
① Req ARP pour trouver @mac 3.254
② Réponse ARP de 3.254
③ Requête ICMP de 0.2 vers 7.255
Le routeur décapsule
@IP dst = 192.168.7.255
diffusion IP
table de routage
eth1
↳ diffusion ethernet sur eth1
④ Requête ICMP part de eth1
@mac src = eth1(7.254)
@mac dst = FF:FF:FF:FF:FF:FF
type = 0800 (IP)
3.254
7.254
4.5
0.2

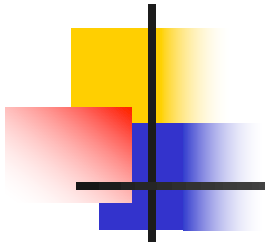
# ARP - ICMP


Sur le routeur : décapsulation IP. Comment aller vers 4.5 ?
entête IP → @IP dst = 192.168.4.5 → table de routage du routeur
eth0 192.168.3.254/22
eth1 192.168.7.254/22
192.168.0.0/22 → eth0
192.168.4.0/22 → eth1
Pour aller vers 4.5, on passe par eth1
14 20 8 60 4
@mac src = eth1 (7.254)
type = 0800 (IP)
@mac dst = ? celle de 4.5 que le routeur ne connait pas
④ Requête ARP
ARP
@mac src = eth1 (7.254)
@IP src = 192.168.7.254
@mac dst = ♡ (?)
@IP dst = 192.168.4.5
Trame 4
@mac dst = FF:FF:FF:FF:FF:FF
@mac src = eth1 (7.254)
type = 0806 (ARP)
ETH
⑤ Réponse ARP
Trame
ARP
@mac src = eth1 (7.254)
@IP src = 192.168.7.254
@mac dst = eth0 (4.5)
@IP dst = 192.168.4.5
@mac dst = eth1 (7.254)
@mac src = eth0 (4.5)
type = 0806 (ARP)
ETH
Trame ⑥ Requête ICMP
@mac dst = eth0 (4.5)
20 entête IP
@IP dst = 192.168.4.5
@IP src = 192.168.0.2
proto = 1 (ICMP)

# RARP - Reverse ARP

RFC 903

- ARP : @IP->@MAC RARP : @MAC->@IP
- "Mon @MAC est xx:xx:xx:xx:xx:xx. Quelqu'un connaît-il mon @IP ?"
- permet à un hôte de récupérer son @IP au démarrage par interrogation d'un serveur RARP
  - stations sans disque
  - imprimantes,…
- Même fonctionnement, même format de paquet
- Obsolète car désormais remplacé par BOOTP ou DHCP qui peuvent rendre le même service et ne nécessite pas un serveur RARP sur chaque réseau (*broadcast* MAC limité)



# BOOTP (*bootstrap*) - Principe

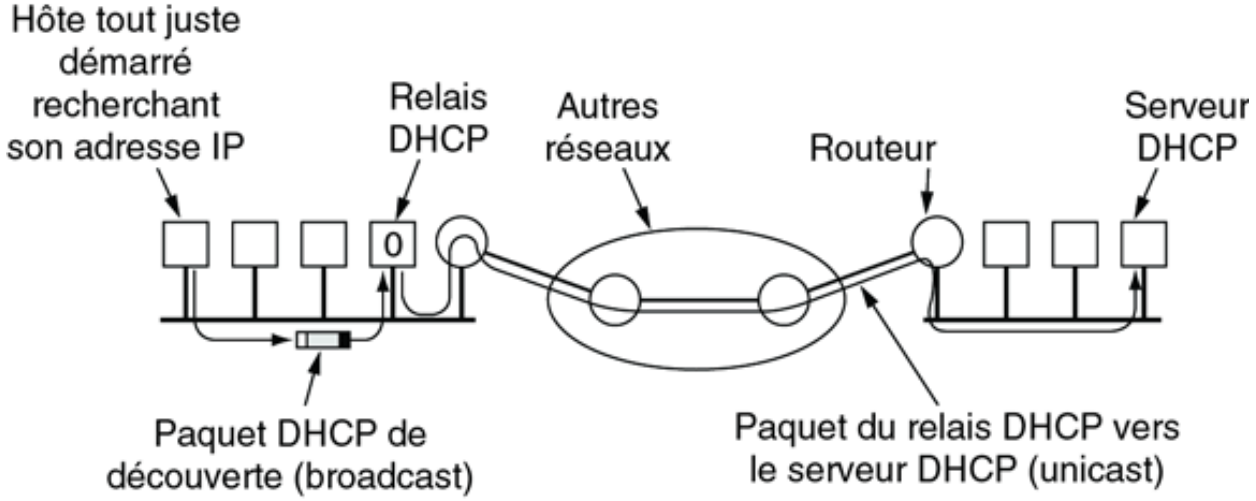
RFC 951, 1048, 1084

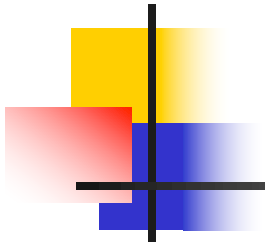
- Protocole d'amorçage du réseau au dessus de UDP (les diffusions passent les routeurs)
- le serveur informe la machine qui démarre de
  - son @IP, @IP du serveur de fichiers qui contient son image disque, @IP du routeur par défaut, masque de sous-réseau
- Inconvénient : les tables de correspondances sont statiques (configurées manuellement)
- Pour y remédier, BOOTP est devenu DHCP : *Dynamic Host Configuration Protocol*

# DHCP - Principe

RFC 2131, 2132

- Configuration manuelle ou assignation dynamique des adresses IP
- Un serveur spécifique s'occupe d'assigner des configurations réseaux aux hôtes qui en font la demande
- Le serveur n'est pas nécessairement sur le même réseau (passage par un relais DHCP)

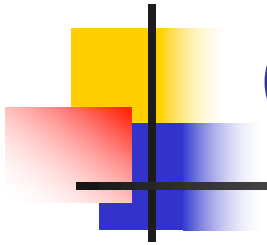




# DHCP - Fonctionnement

- DHCP - économie d'adresses IP : quand un hôte quitte le réseau, il restitue son adresse
- Les messages DHCP (au dessus d'UDP)
  - **DHCPDiscover** : diffusion du client pour que les serveurs DHCP actifs répondent en fournissant une @IP
  - **DHCPOffer** : offre des serveurs (réponse à DHCPDiscover)
  - **DHCPRequest** : après avoir sélectionné une offre, le client émet une requête d'affectation d'@ au serveur élu
  - **DHCPAck** : le serveur renvoie une config. Réseau et une durée de validité (*lease time*)
  - **DHCPNAck** : refus d'un renouvellement par le serveur
  - **DHCPRelease** : résiliation du bail avant échéance par le client

# Quelques fichiers de config. UNIX



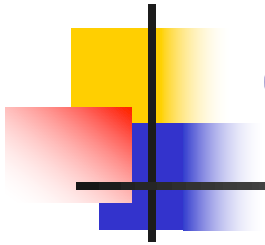
- `/etc/hosts` : association locale nom/@IP
- `/etc/resolv.conf` : @ des serveurs de noms, noms de domaines
- `/etc/protocols` : association nom de protocole, numéro de protocole, liste d'alias

```
icmp        1       ICMP
tcp         6       TCP
```

- `/etc/services` : association nom de service, numéro de port/protocole, liste d'alias

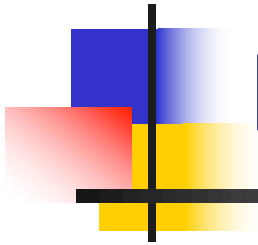
```
ftp         21/tcp          FTP
ssh         22/UDP
```

- `/etc/inetd.conf` : association entre nom de service et exécutable réalisant le service

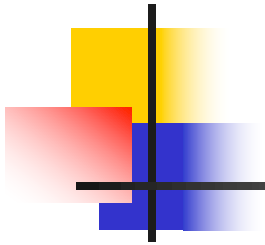


# Quelques commandes UNIX

- `ping` : teste l'accessibilité d'une destination
- `traceroute` : renvoie la route prise par les paquets pour atteindre une destination
- `arp` : visualiser/modifier le cache ARP
- `host` : interroger un serveur de noms
- `netstat` : obtenir des statistiques sur le nombre de paquets, les erreurs, les collisions, une interface, une table de routage, les sockets ouvertes, …
- `tcpdump` : visualiser des informations qui passent par l'interface réseau d'une machine

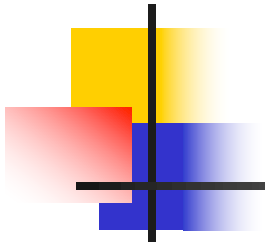


# Le protocole IPv6



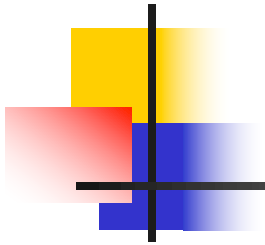
# Pourquoi IPv6 ?

- La fin d'IPv4 est proche
  - pénurie d'adresses IP et explosion des tables de routage
  - prolongement de quelques années grâce au routage CIDR et au NAT (solutions transitoires)
  - besoin d'un nouveau protocole mais suppose de le déployer sur tous les nœuds de l'Internet actuel !
- L'IETF, en 1990, élabore les souhaits d'un nouveau protocole et fit un appel à propositions
- 1993 : IPv6 est née de propositions combinées (Deering et Francis) - RFC 2460 à 2466



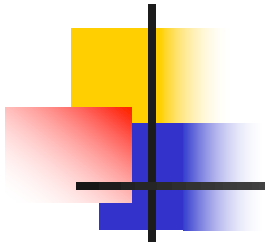
# Un nouveau protocole IP

- IETF - 1990 - objectifs d'une nouvelle version d'IP
  - supporter des milliards d'hôtes
  - réduire la taille des tables de routage
  - simplifier encore le protocole pour routage plus rapide des paquets
  - offrir une meilleure sécurité
  - accorder plus d'importance à la QoS (trafic temps-réel)
  - améliorer la diffusion multicast
  - permettre la mobilité des hôtes sans changer d'@ IP
  - rendre le protocole plus évolutif
  - permettre au nouveau protocole et à l'ancien de coexister pendant quelques années



# IPv6 - Caractéristiques (1)

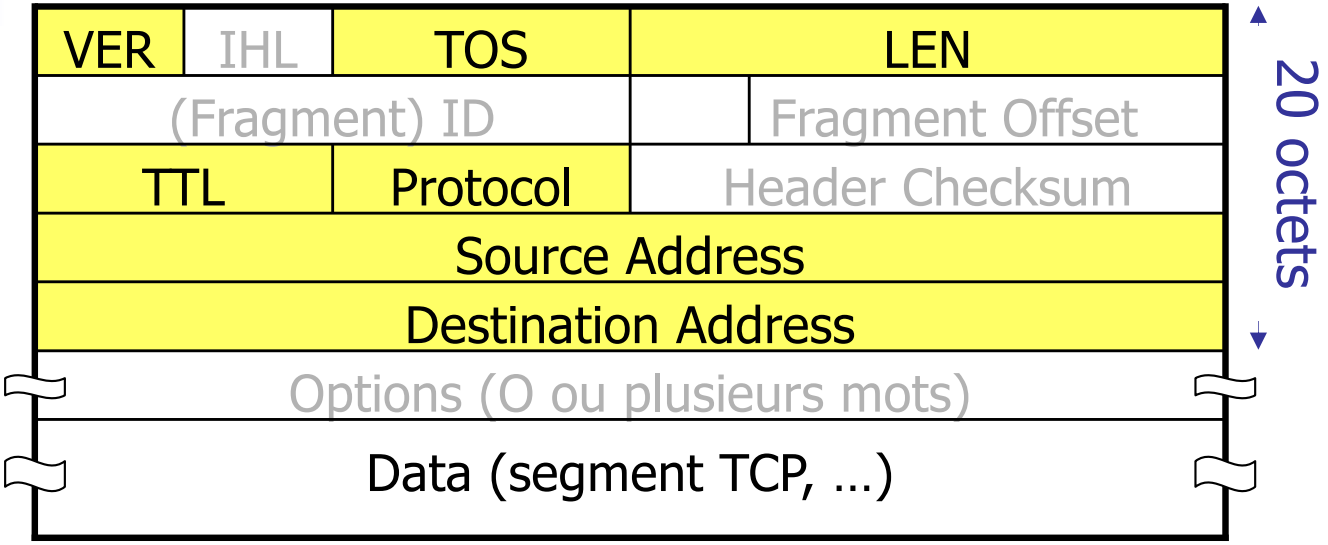
- IPv6 est compatible avec
  - non seulement IPv4
  - mais aussi TCP, UDP, ICMP, OSPF, BGP, DNS, … (ou quelques modifications mineures)
- Supporte un format d'adresses plus longues
  - 16 octets au lieu de 4 (quasiment inépuisable)
- Simplification de l'en-tête
  - 7 champs au lieu de 13 (accélère le traitement dans les routeurs)
  - meilleure gestion des options avec une taille fixe (accélère le temps de traitement des paquets)



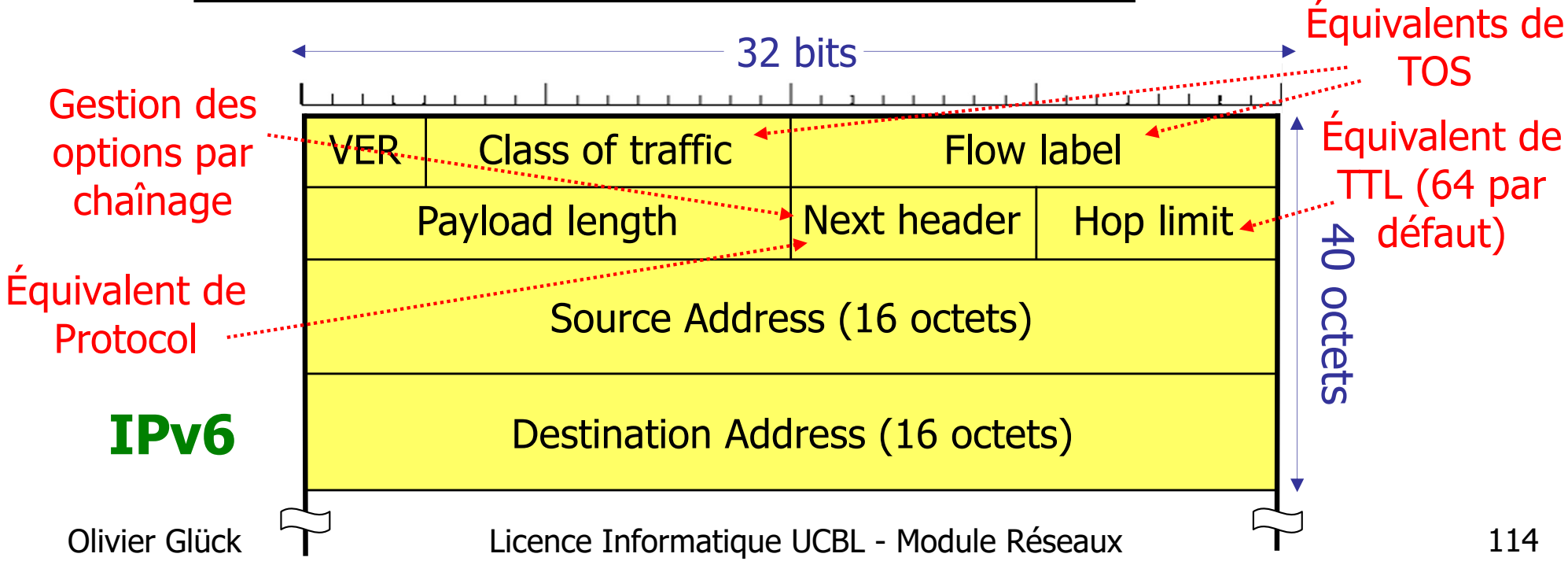
# IPv6 - Caractéristiques (2)

- Sécurité accrue
  - intégrité des données
  - mécanismes d'authentification et de cryptographie
- Plus de fragmentation dans les nœuds intermédiaires
  - mécanisme de découverte du MTU optimal (envoi de paquets ICMP en diminuant la taille jusqu'à recevoir une réponse)
  - fragmentation par la source uniquement
- Plus de champ *checksum*
  - allège considérablement le travail des routeurs
- Amélioration des aspects de diffusion (multicast)

# Le datagramme IPv6

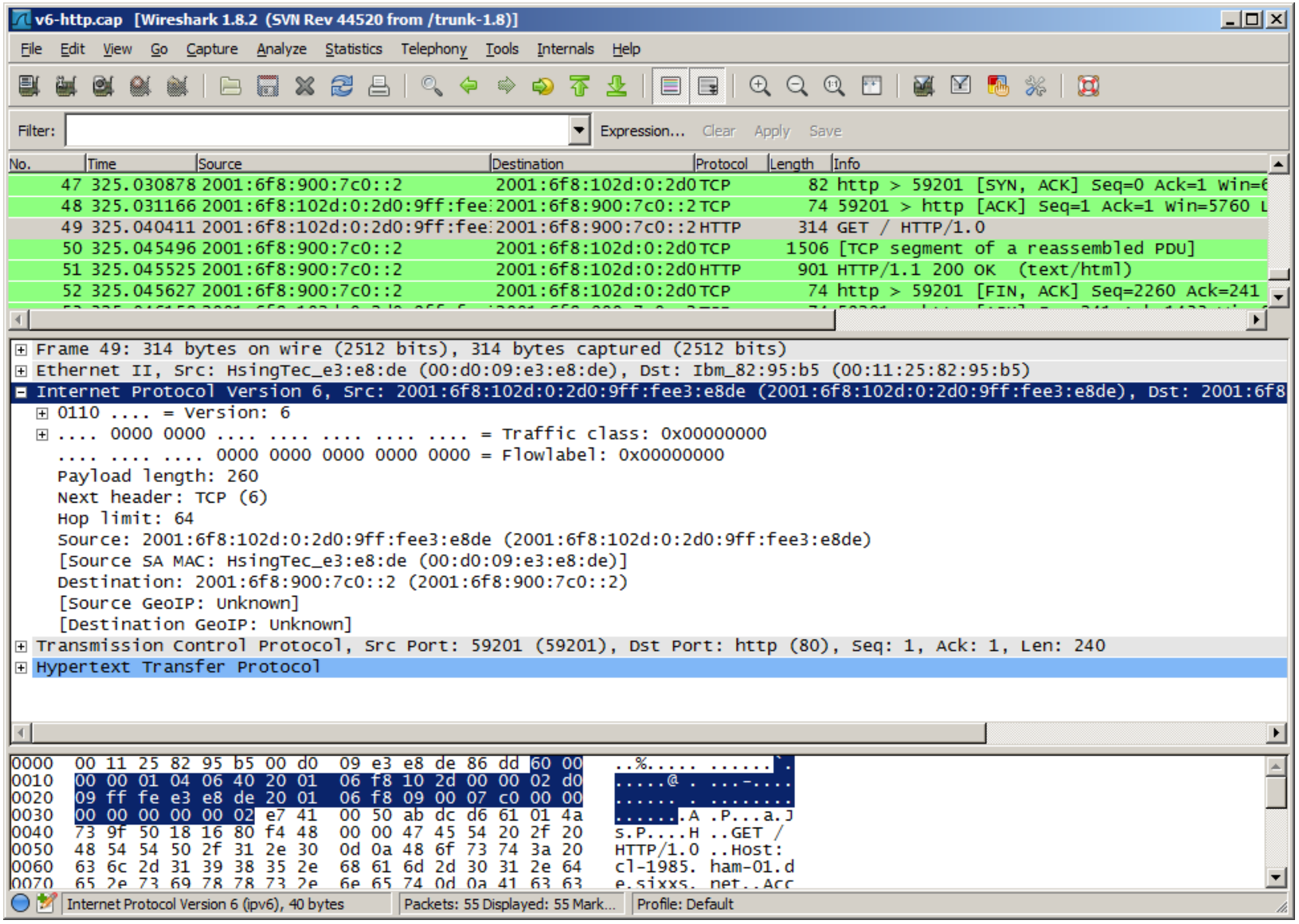



**IPv4**




**IPv6**

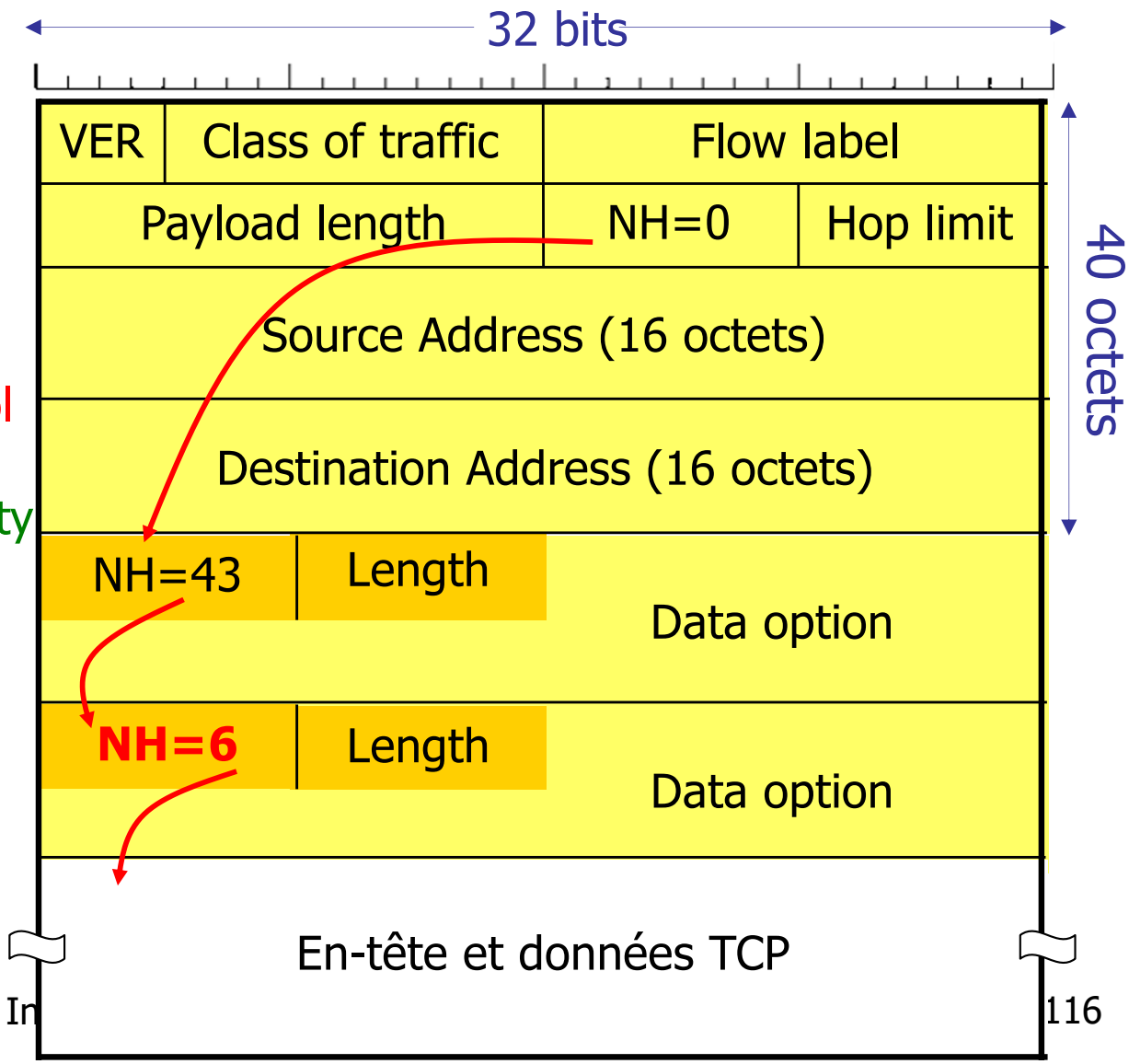
# Exemple d'en-tête IPv6

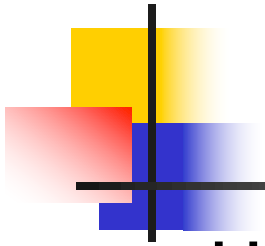


# Le chaînage des options

Le champ *Next Header* (NH) :

0 : option "Hop-by-Hop"
4 : IPv4
6 : TCP
17 : UDP
43 : option "Routing Header"
44 : option "Fragment Header"
45 : Interdomain Routing Protocol
46 : RSVP
50 : option "Encapsulation Security Payload" (IPsec)
51 : option "Authentification Header" (IPsec)
58 : ICMP
59 : No next header
60 : option "Destination Options Header"





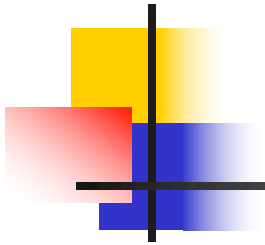
# Exemples d'options

- Hop by Hop
  - la seule qui doit être traitée par tous les routeurs traversés (les autres extensions sont traitées comme un protocole de niveau 4 !)
  - transport d'informations <type d'option, longueur, valeur> dont on sait qu'elle sera examinée par tous les routeurs (par ex. support de datagramme de taille supérieure à 64Ko : *jumbogram*)
- Routing Header
  - liste de routeurs à traverser obligatoirement
- Fragmentation Header
  - pour permettre au destinataire de réassembler les fragments (reprend les champs de IPv4)
- Destination Options Header
  - informations additionnelles pour la destination

# L'adressage (1)

- Adressage hiérarchique pour alléger les tables de routage


  - un préfixe de localisation - public - 48 bits
  - un champ de topologie locale (subnet) - 16 bits
  - un identifiant de désignation de l'interface (basé sur l'@MAC) sur 64 bits (équivalent HOST_ID) qui garantie l'unicité de l'adresse
- Notation : groupes de 4 chiffres hexadécimaux séparés par :
  - ex : 8000:0000:0000:0000:0123:4567:89AB:CDEF
  - :: représente un ou plusieurs groupes de 0000
  - ex : 8000::123:4567:89AB:CDEF



# L'adressage (2)

- Trois types
  - adresses unicast : désigne une interface
  - adresses multicast (FF00::/8) : désigne un ensemble d'interfaces (localisées n'importe où)
  - adresses anycast :
    - restriction du multicast
    - désigne un ensemble d'interfaces partageant un même préfixe réseau
    - n'est délivré qu'à une interface du groupe (celle dont la métrique est la plus proche du nœud source)
  - plus d'adresses de broadcast, remplacée par FF02::1

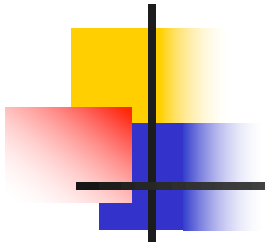
# L'adressage (3)

- Adresses particulières
  - :: (unspecified address) : équivalent de 0.0.0.0, interface en cours d'initialisation
  - ::1 (loopback address) : équivalent de 127.0.0.1
  - adresses de site local (adresses privées) : commençant par FD00::
  - adresses de lien : commençant par FE80::
- Construction de Interface_ID

  @ MAC 00:A0:24:E3:FA:4B

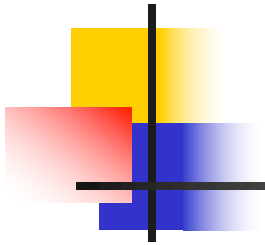
  Interface_ID 02A0:24FF:FEE3:FA4B (U/L=1, I/G=0)
- Adressage agrégé
  - la partie publique est découpée en différents sous-champs ; un sous-champ est attribué par l'organisme qui s'est vu affecter le champ précédent (assignation hiérarchique)


# Les protocoles de transport : UDP et TCP



# Le protocole UDP

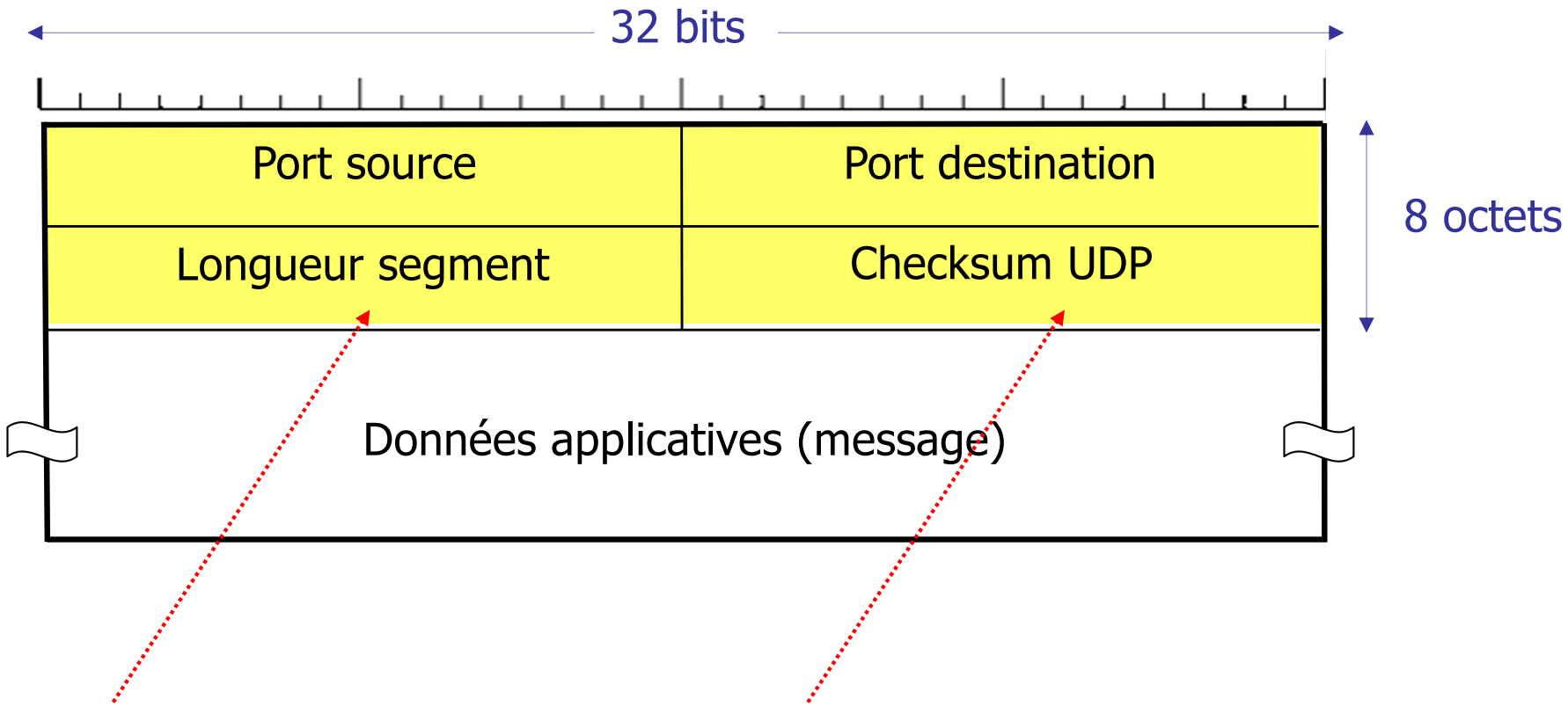
- UDP (RFC 768) - User Datagram Protocol
  - protocole de transport le plus simple
  - service de type best-effort (comme IP)
    - les segments UDP peuvent être perdus
    - les segments UDP peuvent arriver dans le désordre
  - mode non connecté : chaque segment UDP est traité indépendamment des autres
- Pourquoi un service non fiable sans connexion ?
  - simple donc rapide (pas de délai de connexion, pas d'état entre émetteur/récepteur)
  - petit en-tête donc économie de bande passante
  - sans contrôle de congestion donc UDP peut émettre aussi rapidement qu'il le souhaite



# Les utilisations d'UDP

- Performance sans garantie de délivrance
- Souvent utilisé pour les applications multimédias
  - tolérantes aux pertes
  - sensibles au débit
- Autres utilisations d'UDP
  - applications qui envoient peu de données et qui ne nécessitent pas un service fiable
  - exemples : DNS, SNMP, BOOTP/DHCP
- Transfert fiable sur UDP
  - ajouter des mécanismes de compensation de pertes (reprise sur erreur) au niveau applicatif
  - mécanismes adaptés à l'application

# Le datagramme UDP




Taille totale du segment
(en-tête+données)

Total de contrôle du segment
(en-tête+données)

optionnel : peut être à 0

UDP = IP + multiplexage (adresse de transport) !!

# Le protocole TCP

- Transport Control Protocol (RFC 793, 1122, 1323, 2018, 2581)

**Attention: les RFCs ne spécifient pas tout - beaucoup de choses dépendent de l'implantation du protocole**

- Transport fiable en mode connecté
  - point à point, bidirectionnel : entre deux adresses de transport (@IP src, port src) --> (@IP dest, port dest)
  - transporte un flot d'octets (ou flux)
    - l'application lit/écrit des octets dans un tampon
  - assure la délivrance des données en séquence
  - contrôle la validité des données reçues
  - organise les reprises sur erreur ou sur temporisation
  - réalise le contrôle de flux et le contrôle de congestion (à l'aide d'une fenêtre d'émission)

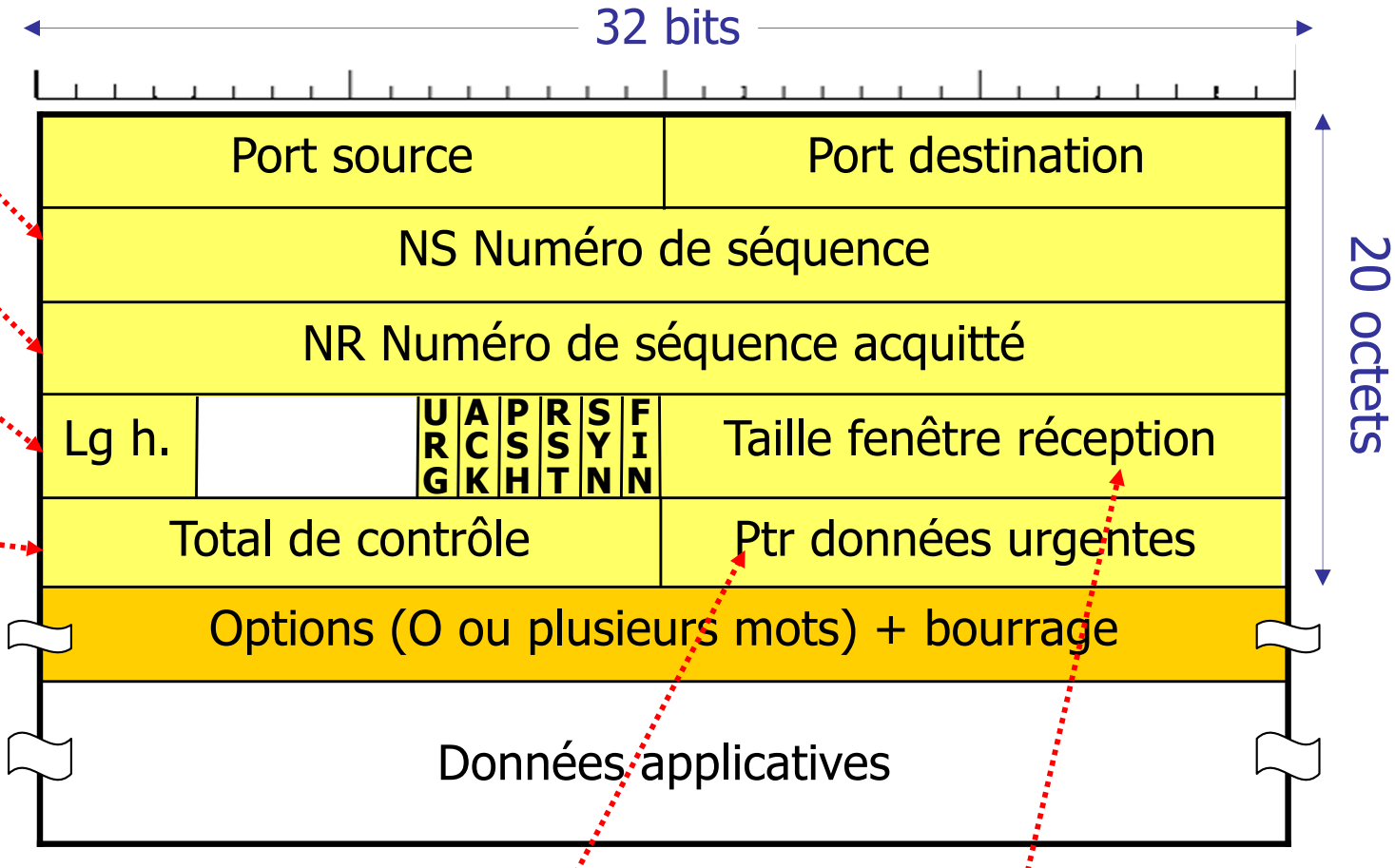
# Le segment TCP (1)

Numéro du premier octet du segment

Numéro du prochain octet attendu

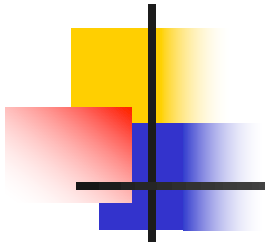
Longueur en-tête en multiple de 4 octets

Checksum sur tout le segment (cf. UDP)



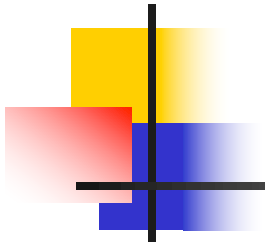

Les données comprises entre le premier octet DATA et la valeur du Ptr sont urgentes : TCP interrompt l'application pour forcer la lecture

Nb d'octets que le récepteur peut recevoir



# Le segment TCP (2)

- Numéro de séquence NS (émission)
  - comptabilise les **octets** depuis le début de la connexion
  - ISN : numéro de séquence initial, valeur "aléatoire" acquittée lors de l'établissement de la connexion
  - le numéro de séquence du premier octet transmis est ISN+1 puis NS=ISN+nb_octets_transmis+1
- Numéro de séquence NR (réception)
  - le récepteur renvoie le numéro du prochain octet attendu soit NS_reçu+taille_données_reçues

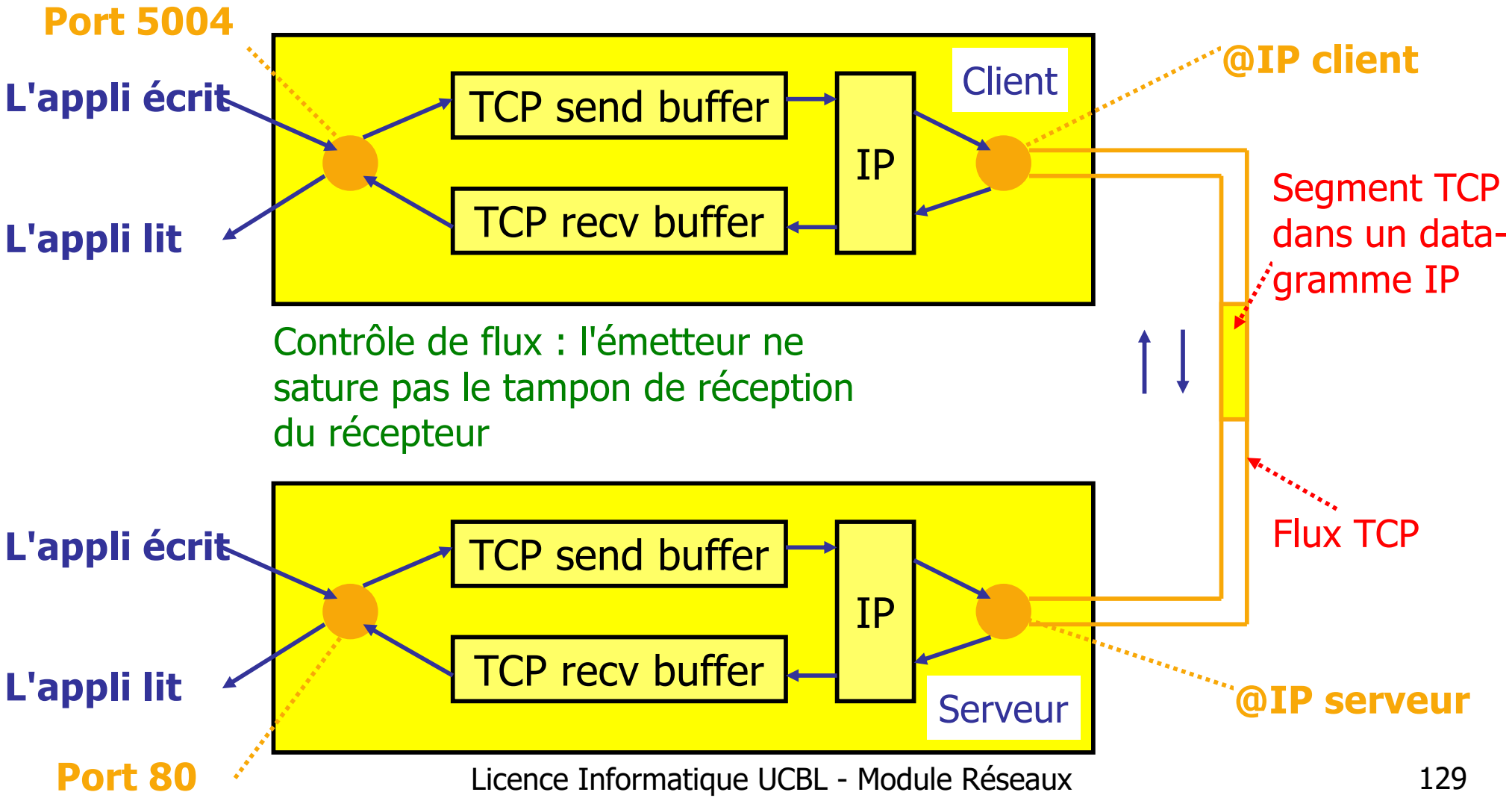


# Le segment TCP (3)

- Les 6 indicateurs
  - **URG** : valide le champ "Ptr données urgentes"
  - **ACK** : valide le champ NR
  - **PSH** : PUSH indique au récepteur de délivrer immédiatement les données en attente sur le récepteur
    - TCP peut attendre d'avoir suffisamment de données avant de constituer un fragment (efficacité du protocole)
    - exemple : retour chariot (CR) dans un terminal virtuel
  - **RST** : demande au destinataire de réinitialiser la connexion ou rejet d'une demande de connexion
  - **SYN** : demande de connexion (échange des ISN)
  - **FIN** : demande de déconnexion (le destinataire n'est pas obligé de s'exécuter : fermeture négociée)

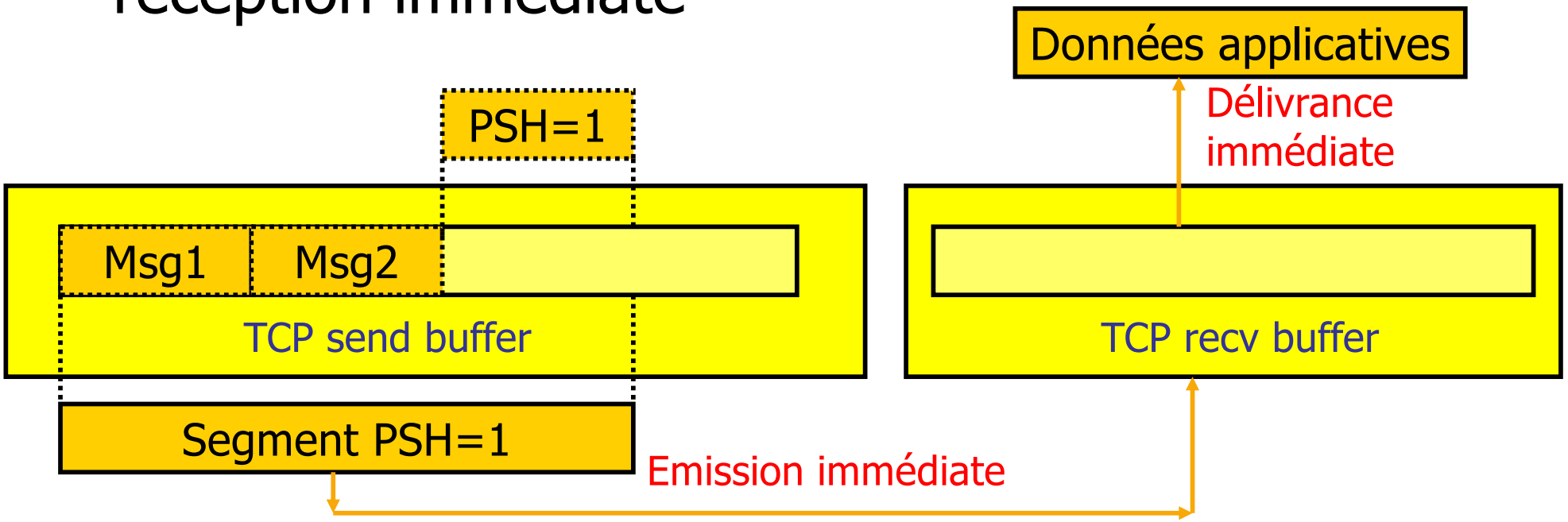
# Une connexion TCP

- Une connexion = **(@IP_src,port_src,@IP_dest,port_dest)**



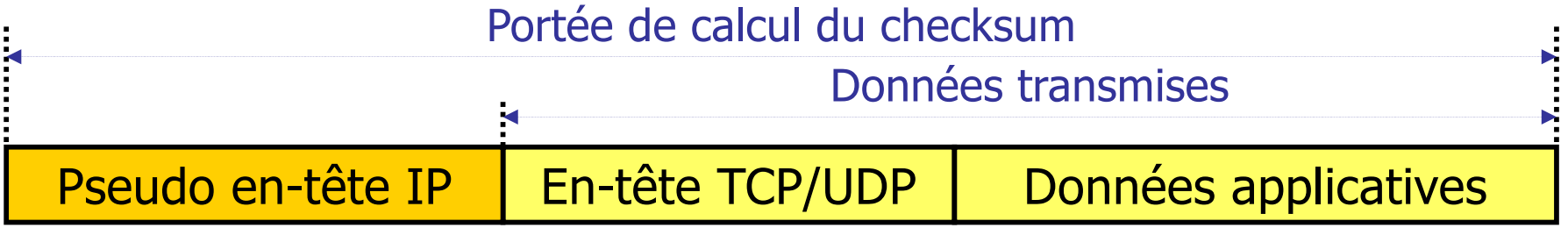
# Le bit PUSH

- Pour optimiser la transmission, par défaut TCP attend que le tampon d'émission soit plein pour constituer un segment (groupage de messages)
- Le bit PUSH sert à demander la transmission et réception immédiate



# Le contrôle d'erreur dans UDP et TCP

- Deux objectifs
  - vérifier que les données transmises n'ont pas été altérées
  - garantir que les données sont transmises au bon destinataire --> rajout d'un pseudo en-tête IP pour le calcul du checksum (qui est non transmis)



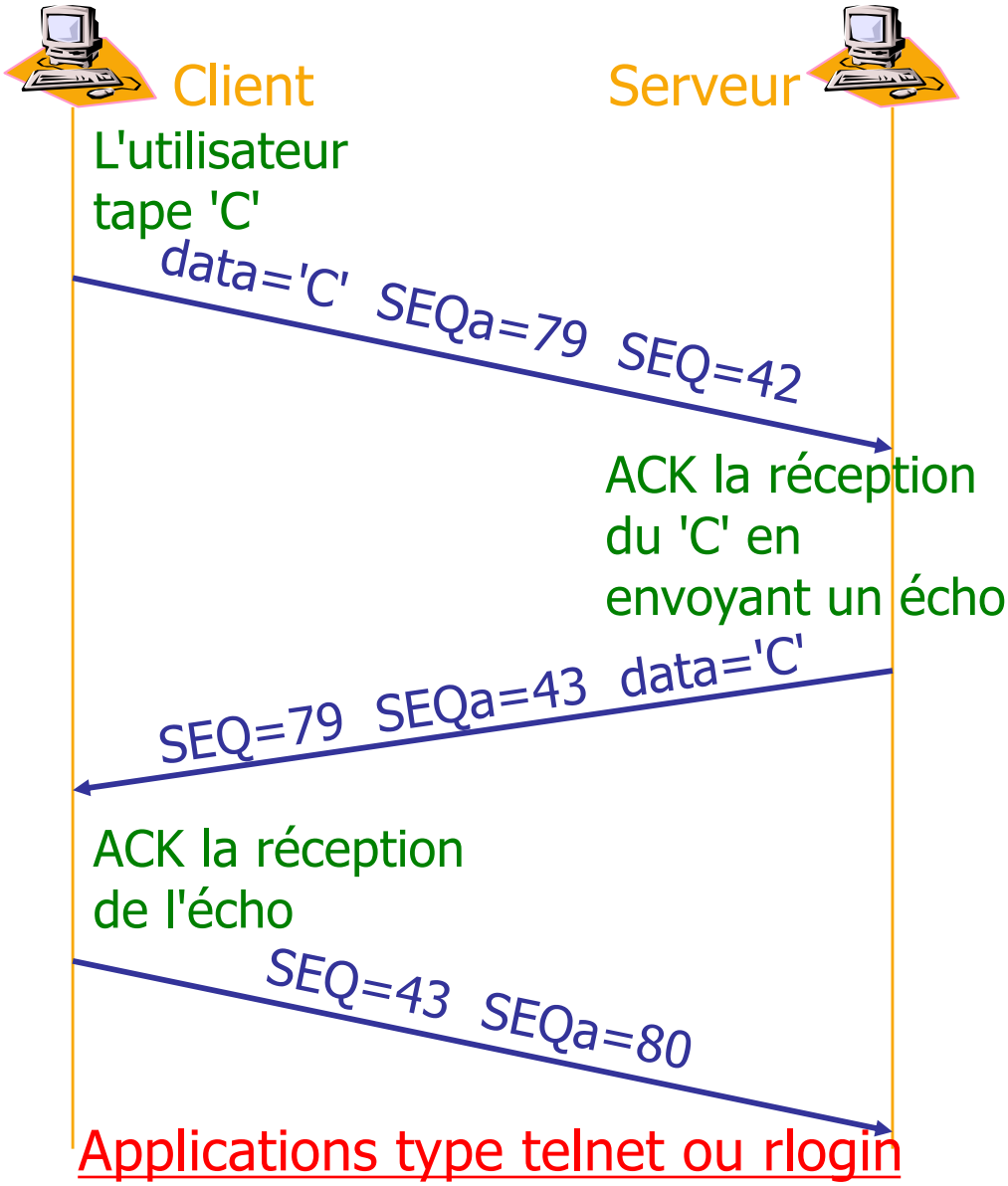

Calcul du checksum=complément à 1 de l'addition des mots de 16 bits

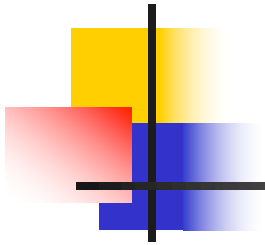
# Exemple de calcul de checksum

- Calcul du checksum par additions des mots de 16 bits complémentées à 1
- Exemple : checksum sur 3 mots de 16 bits
  - 0110011001100110
  - 0101010101010101
  - 0000111100001111
- Somme des deux premiers mots
  - 1011101110111011
- Addition du troisième mot
  - 1100101011001010
- Complément à 1
  - 0011010100110101 (= le champ checksum)
- Si pas d'erreur, la somme de tous les mots de 16 bits reçus doit faire 1111111111111111

# Numéro de séquence et ACK

- Principe du *piggybacking* : un segment peut contenir des données et acquitter un segment précédent
- SEQ=numéro du premier octet dans le segment depuis l'ouverture de la connexion
- SEQa=numéro du prochain octet attendu
- L'acquittement d'un octet acquitte tous les octets précédents



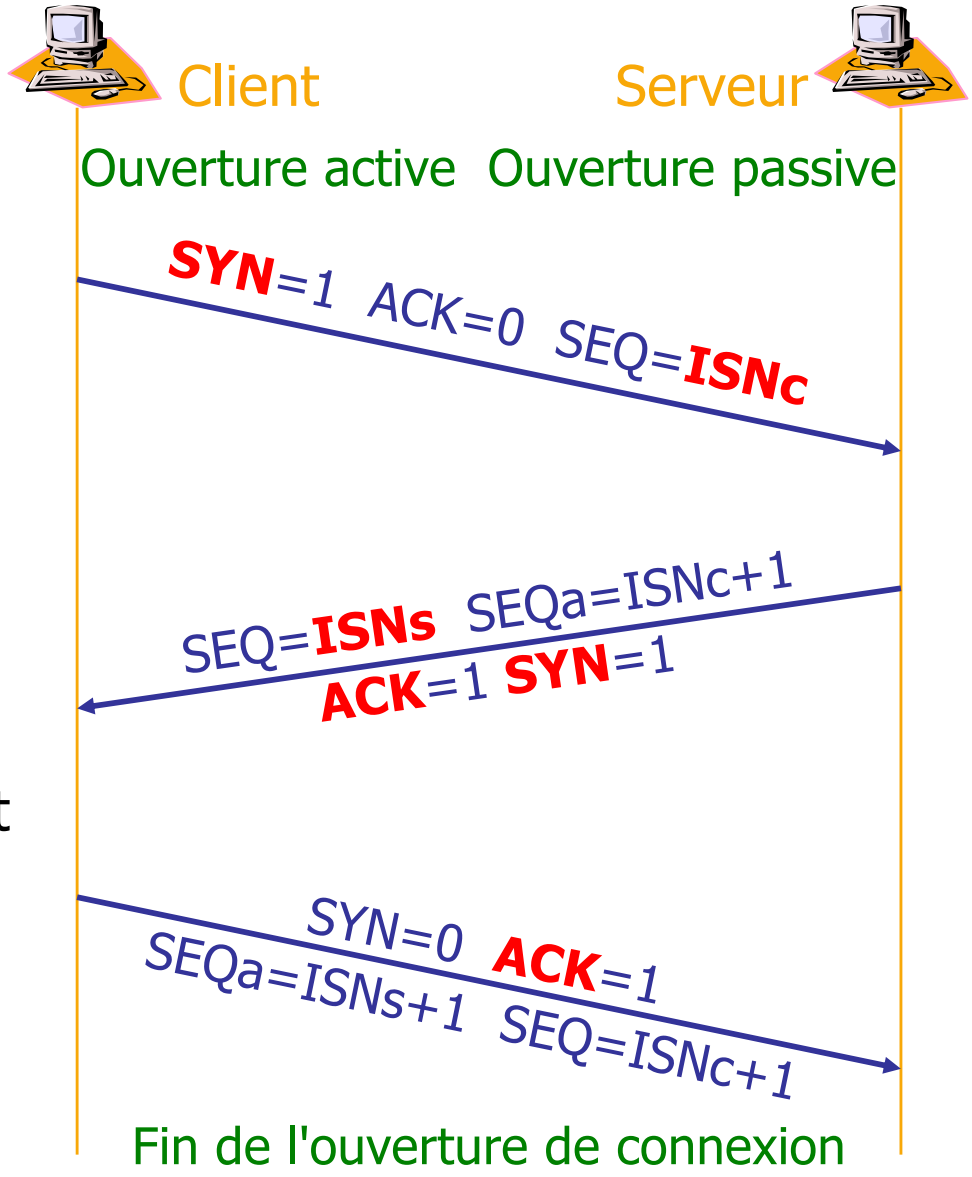


# Le numéro de séquence initial (ISN)

- Chaque entité communique son ISN à l'autre à l'ouverture de la connexion
- Il permet de distinguer les octets de deux connexions successives utilisant les mêmes adresses de transport
  - ouverture de la connexion (@IP1,p1,@IP2,p2)
  - échanges de segments
  - fermeture de la connexion (@IP1,p1,@IP2,p2)
  - ouverture de la connexion (@IP1,p1,@IP2,p2)
  - …
- Un même ISN est tiré toutes les 4h30 environ

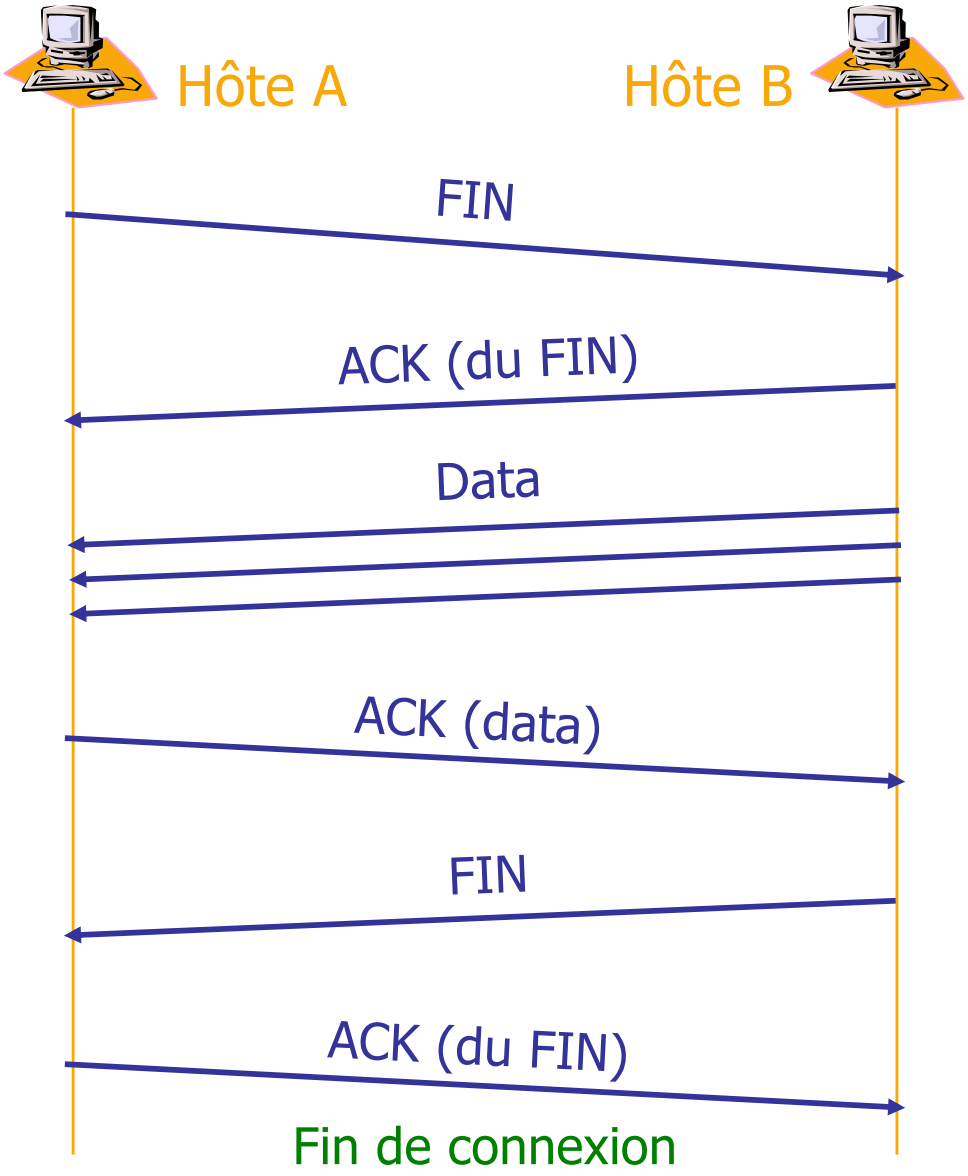
# Etablissement d'une connexion TCP

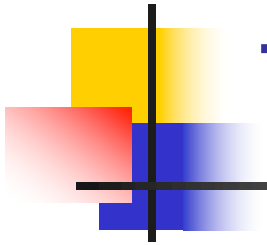
- Connexion en trois phases
  - 1 - demande d'ouverture par le client (SYN), choix ISNc
  - 2 - acceptation par le serveur (SYN+ACK), allocation des tampons, choix ISNs
  - 3 - le client acquitte l'acceptation (ACK)
- Modes d'ouverture
  - ouverture passive : le serveur est en attente de demande de connexion
  - ouverture active : TCP adresse une demande de connexion à une entité identifiée



# Fermeture d'une connexion TCP

- Fermeture négociée
  - 1 - demande de fin de connexion (FIN) par une des extrémités
  - 2 - acquittement du FIN (ACK) mais mise en attente de la demande (B a encore des données non transmises)
  - 3 - B envoie ses données en attente
  - 4 - A acquitte les données (ACK)
  - 5 - acceptation de la fin de connexion par B (FIN)
  - 6 - acquittement de la fin de connexion (ACK)



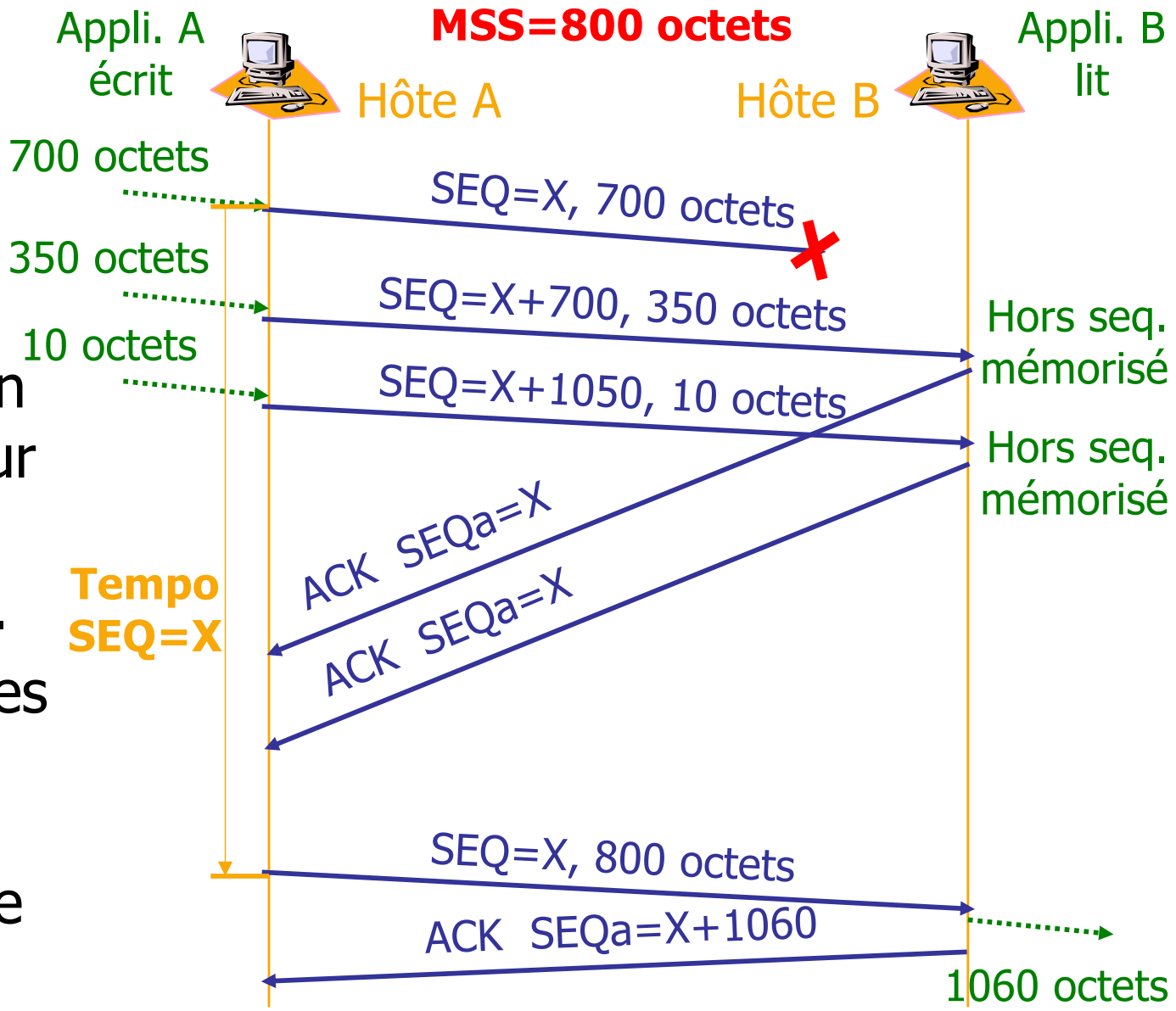


# Taille des segments TCP

- Souplesse de TCP :
  - l'application lit/écrit dans un tampon
  - TCP décide quand envoyer/restituer un segment (si PSH n'est pas positionné)
- Idée :
  - TCP a intérêt d'envoyer des segments de taille maximale (limitation de l'overhead lié à la taille de l'en-tête)
  - fragmentation IP coûteuse --> éviter la fragmentation
  - la taille max. d'un segment TCP est de 64Ko (à cause d'IP)
- MSS : Maximum Segment Size (sans en-tête)
  - à l'ouverture de la connexion, chaque entité peut annoncer (option TCP) son MSS à l'autre, en fonction de la MTU de son réseau (MSS=MTU-40) (20,IP+20,TCP)
  - par défaut, MSS=536 octets

# Politique de retransmissions

- Les pertes de segment sont détectées par absence d'ack positif à expiration d'un temporisateur sur l'émetteur
- 1 tempo. par seg.
- Toutes les données reçues hors séquence sont mémorisées par le destinataire



# Envois des ACK (RFC 1122, 2581)

- Idée : essayer de ne pas envoyer d'ACK sans données, faire des acquittements cumulés

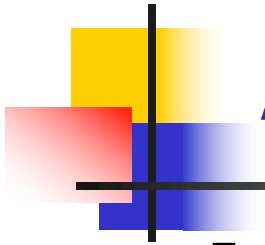
| Evénement sur le récepteur | Action du récepteur |
|---|---|
| Arrivée d'un segment, dans l'ordre (sans trou), les octets précédents ayant étés acquittés | ACK mis en attente. Envoi de l'ACK au bout de 500ms s'il n'y a pas eu de données à envoyer entre temps |
| Arrivée d'un segment, dans l'ordre (sans trou), un ACK d'un segment précédent est déjà en attente | Envoi immédiat d'un ACK qui acquitte l'ensemble (ACK cumulé) |
| Arrivée d'un segment, dans le désordre (avec un numéro de séquence supérieur à celui attendu : création d'un trou) | Envoi d'un ACK dupliqué : si le récepteur attendait 120, il renvoie un ACK avec SEQa=120 |
| Arrivée d'un segment qui remplit partiellement ou complétement un trou | Envoi immédiat d'un ACK |

# Valeur du temporisateur RTO

- RTO : Retransmission Time Out
- Impact de cette valeur sur les performances
  - valeur trop petite : des retransmissions inutiles ont lieu
  - valeur trop grande : attente trop importante entre deux retransmissions
- TCP fait une estimation du RTT (temps aller/retour) et ajuste **dynamiquement** la valeur du temporisateur en fonction de cette estimation
  - utilisation d'une option TCP :
    - l'émetteur met la valeur de son horloge dans l'option
    - le récepteur fait écho de cette valeur dans l'ACK
    - l'émetteur fait la différence entre son horloge et cette valeur à la réception de l'ACK

# Retransmissions rapides *(Fast Retransmit)*

- Idée : il est de toute façon pénalisant d'attendre l'expiration du RTO pour retransmettre
  - quand le récepteur reçoit des données hors-séquence, il renvoie immédiatement un ACK indiquant les données qu'il attend
  - si l'émetteur a reçu 3 ACK dupliqués (4 ACK avec le même numéro de séquence attendu), il retransmet sans attendre l'expiration du RTO
- Suppositions de cas de perte :
  - expiration du RTO --> pas d'ACK dupliqués, congestion dans le réseau (plusieurs segments sont perdus)
  - ACK dupliqués --> peu de segments sont perdus (un ACK dupliqué signifie la réception d'un segment)

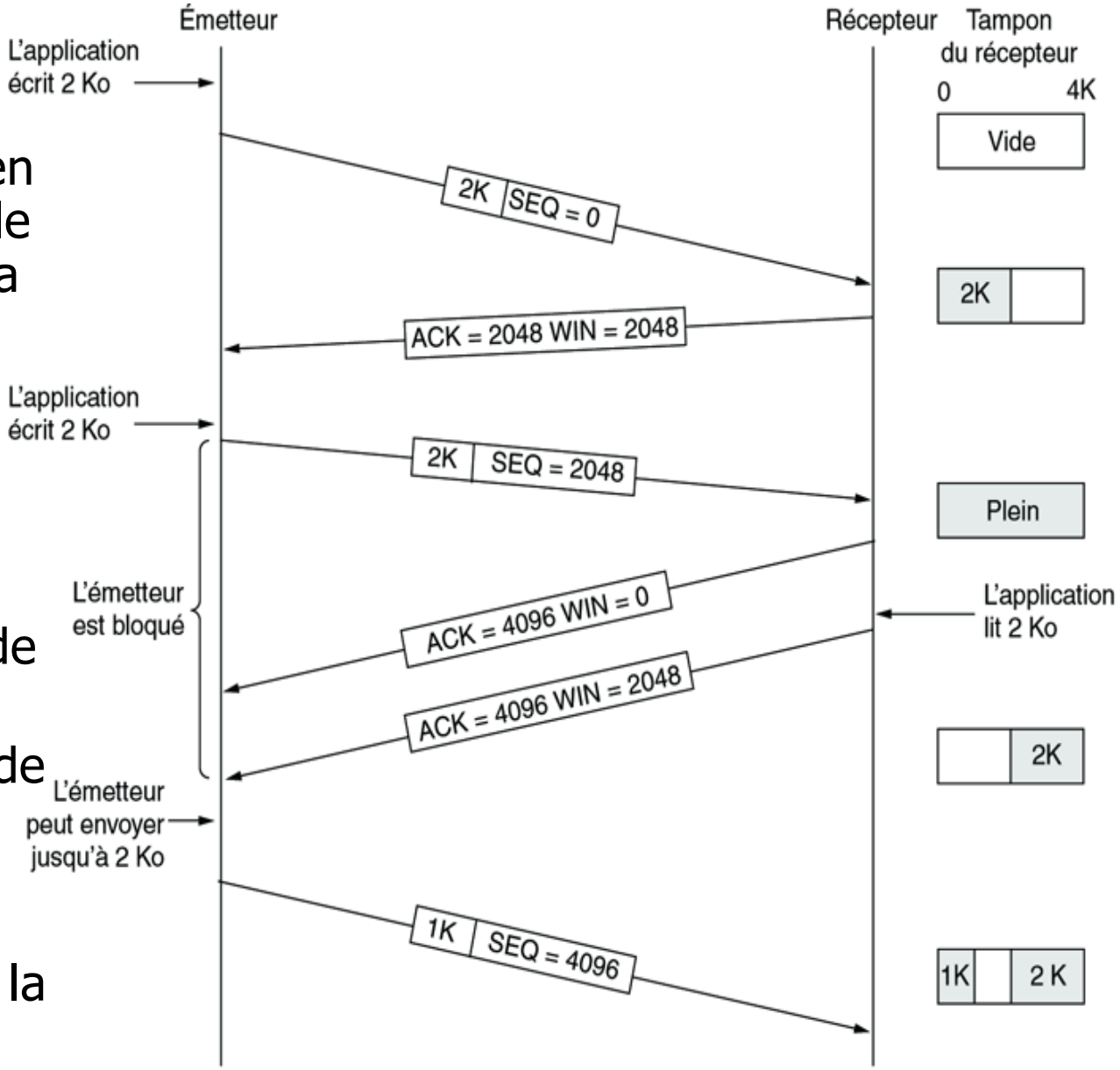


# Algorithme de Nagle

- Idée : il est pénalisant d'envoyer des segments contenant 1 seul octet de données (par ex. terminal virtuel)
- principe de Nagle : si l'appli. écrit octet par octet
  - envoi du premier octet dans un segment
  - accumulation dans le tampon des octets suivants tant que le premier octet n'est pas acquitté
  - envoi des octets accumulés dans un seul segment
  - attente de l'acquittement pour envoyer le segment suivant…
- Peut être désactivé dans certains cas (X-Window : mouvements de souris saccadés)

# Gestion de la fenêtre

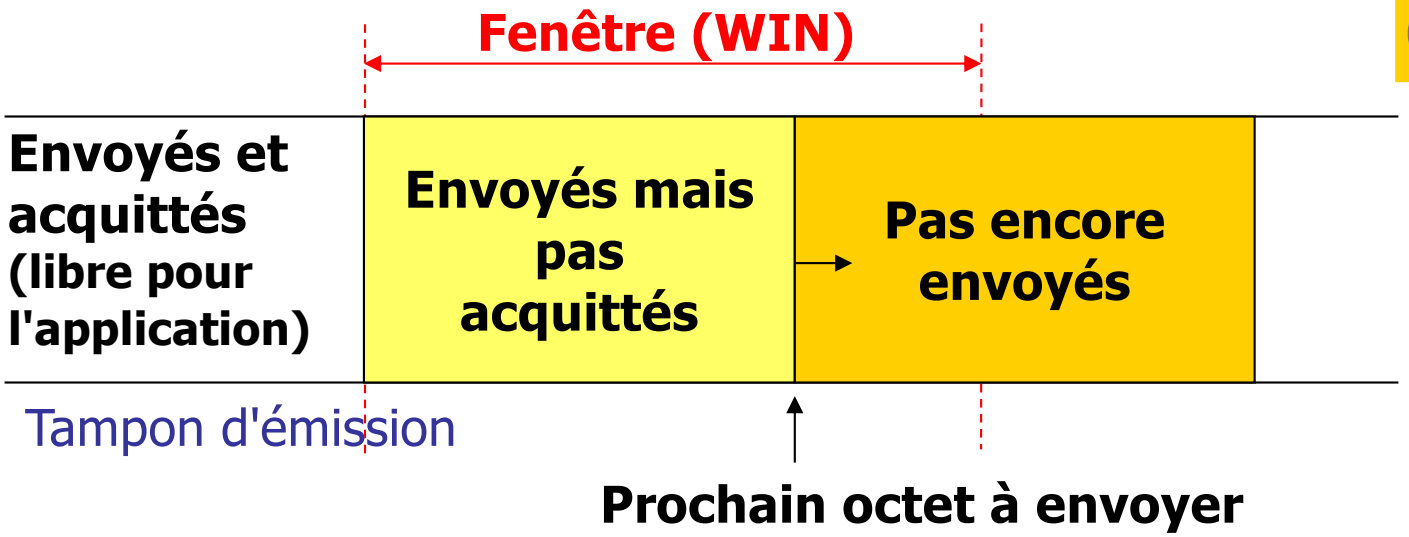
- Contrôle de flux TCP :
  - l'émetteur ne doit pas en envoyer de données si le tampon de réception n'a pas l'espace libre correspondant
- Quand l'émetteur est bloqué, il peut :
  - envoyer des données urgentes (interruption de l'application réceptrice)
  - envoyer des segments de 1 octet pour obliger le récepteur à envoyer SEQa et WIN et maintenir l'état actif de la connexion



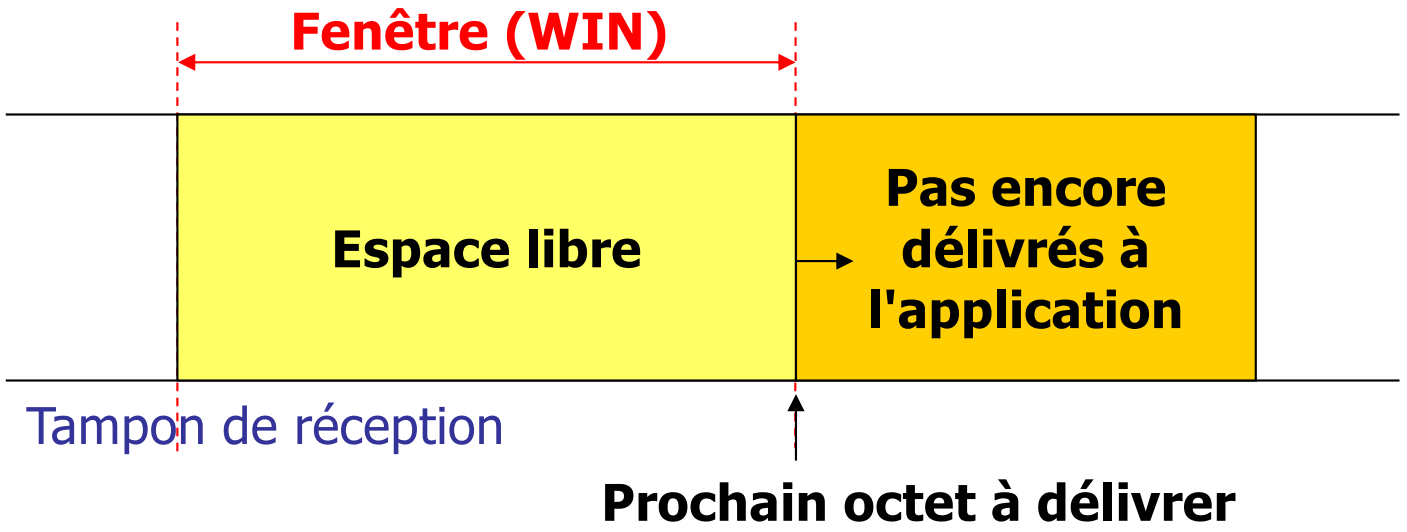
# Contrôle de flux TCP

http://wps.aw.com/aw_kurose_network_2/0,7240,227091-,00.html.

**Côté émetteur**

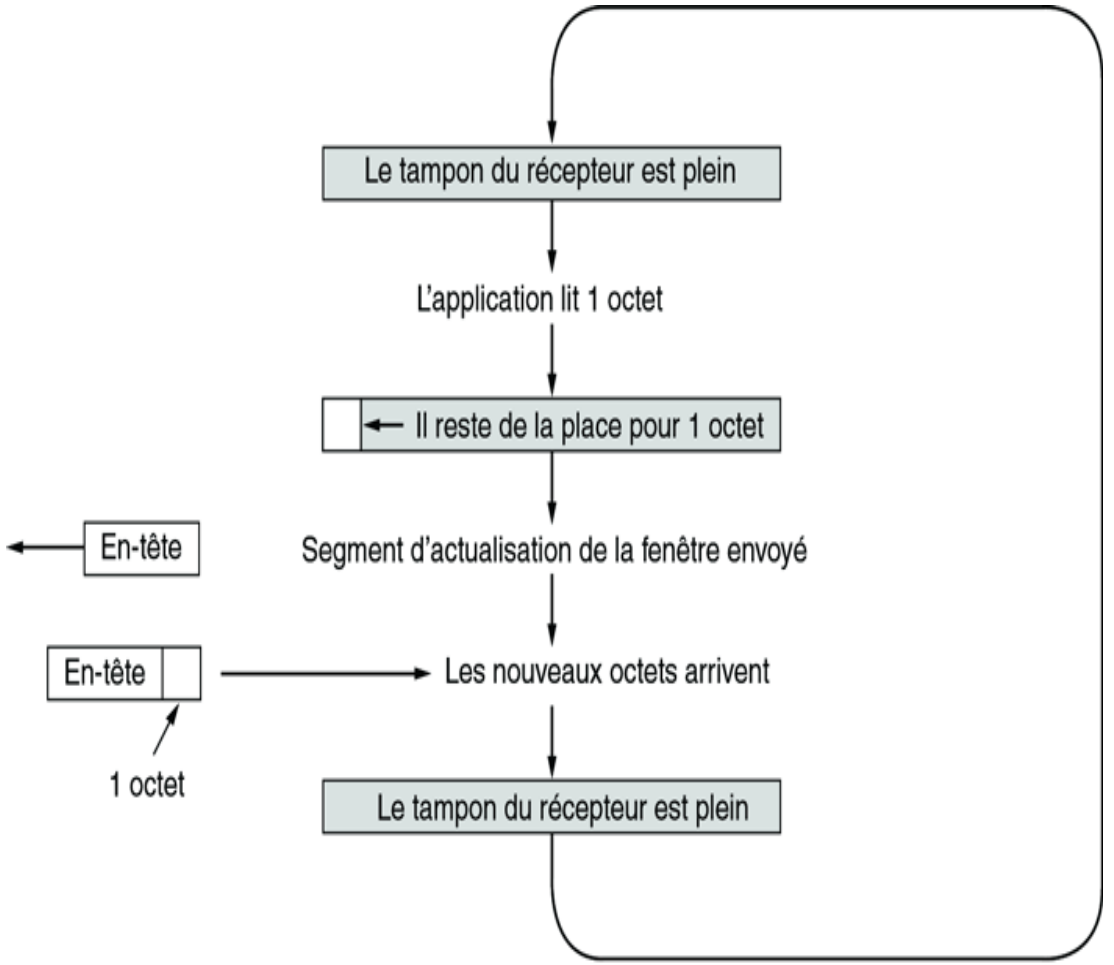



**Côté récepteur**

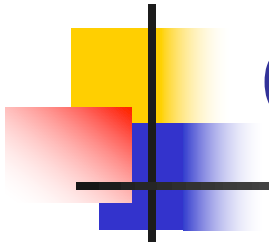


# Syndrome de la fenêtre stupide

- Problème lié au fait que l'application réceptrice lit (vide le tampon) octet par octet
  - l'émetteur ne peut alors envoyer que des petits segments
- Solution de Clark :
  - le récepteur n'annonce la réouverture de la fenêtre que lorsqu'une taille suffisante est disponible --> minimum entre MSS et taille_tampon/2

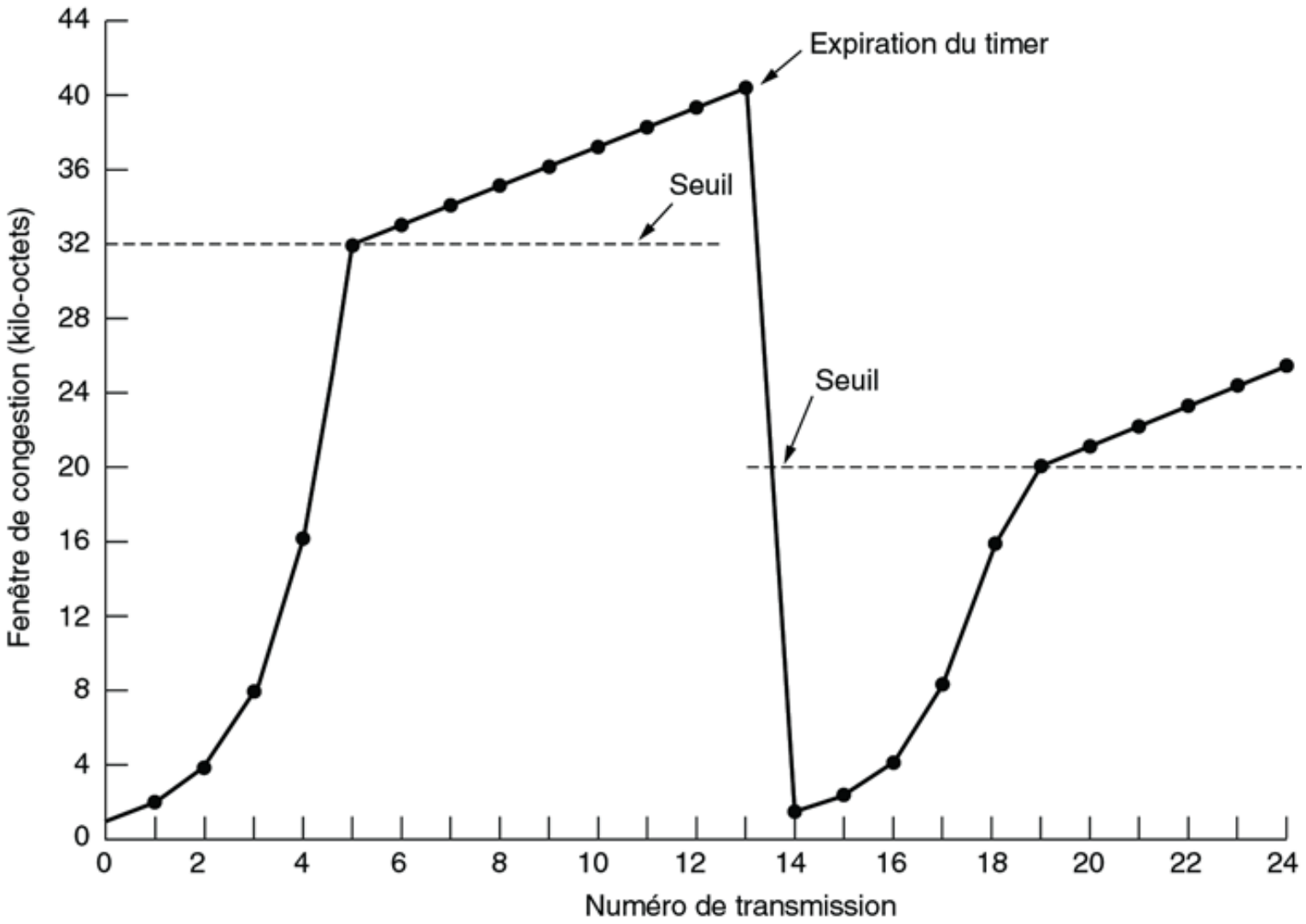


# Contrôle de congestion dans TCP

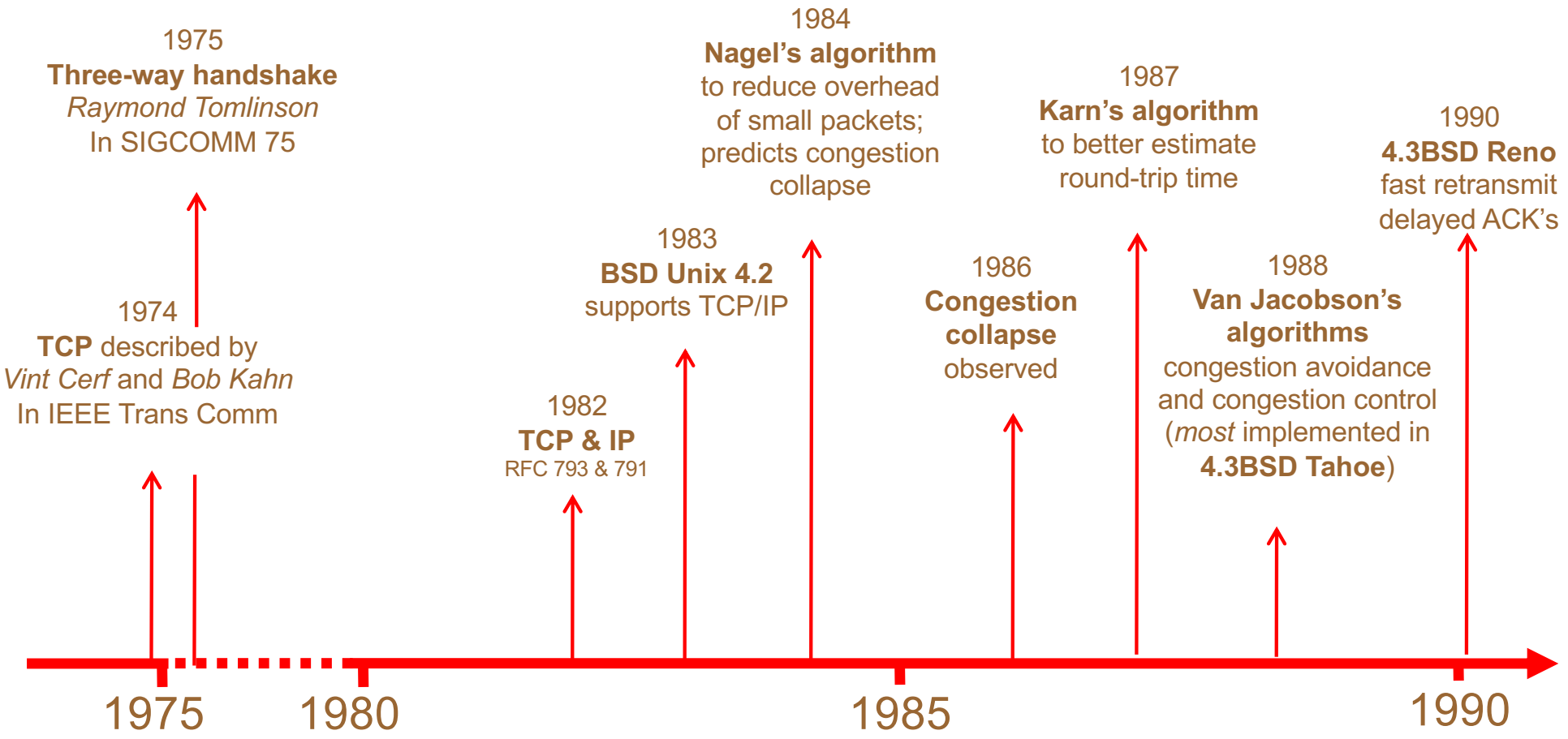


- *Congwin* : taille de la fenêtre de congestion
- *Recvwin* : taille de la fenêtre de réception
- La fenêtre d'émission (quantité d'octets que l'émetteur peut envoyer) est *Min(Recvwin,Congwin)*
- *Threshold* : seuil d'évitement de congestion qui définit la limite entre les deux phases suivantes
  - *slow start* : *Congwin* augmente exponentiellement tant que
    - *Threshold* n'est pas atteint
    - une perte se produit
  - *congestion avoidance* : Congwin augmente linéairement tant que pas de perte
- au départ, Threshold=64Ko et Congwin=1*MSS

# Contrôle de congestion dans TCP



# Evolutions de TCP (1)



# Evolutions de TCP (2)