

# Principes de base du réseau



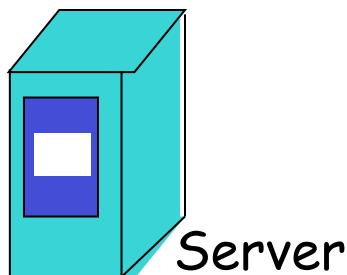
# Partie 1: Les concepts de réseau

# plan

- Généralités
- Classification
- Topologie
- Réseau cœur: mise en relation
- Le modèle de référence

Déf. Un réseau est un ensemble de matériels et de logiciels dispersés destinés à assurer le transport de données.

Matériels:

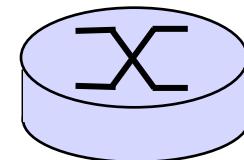


Logiciels: Protocoles

- la délimiation des blocs de données échangés
- le contrôle et organisation de l'échange



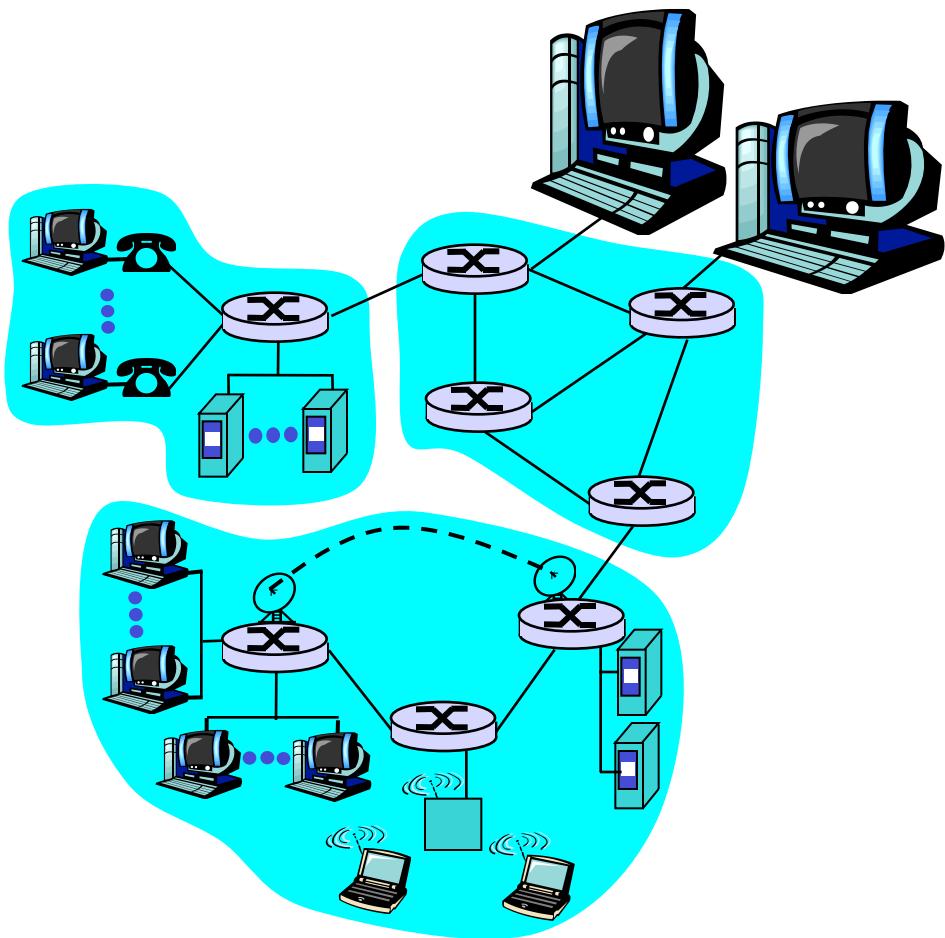
mobile



routeur/switch



workstation



# Classification des réseaux

- Réseaux Personnels PAN(Personal Area Network)
  - Couverture: de 10m à 100m
  - Débit: quelques Mbits/s
  - Bluetooth, HomeRF,...
- Réseaux locaux LAN(Local Area Network)
  - Couverture: de 100m à 1000m
  - Débit: quelques dizaines de Mbits/s
  - Ethernet, Token Ring, WiFi, HipperLan

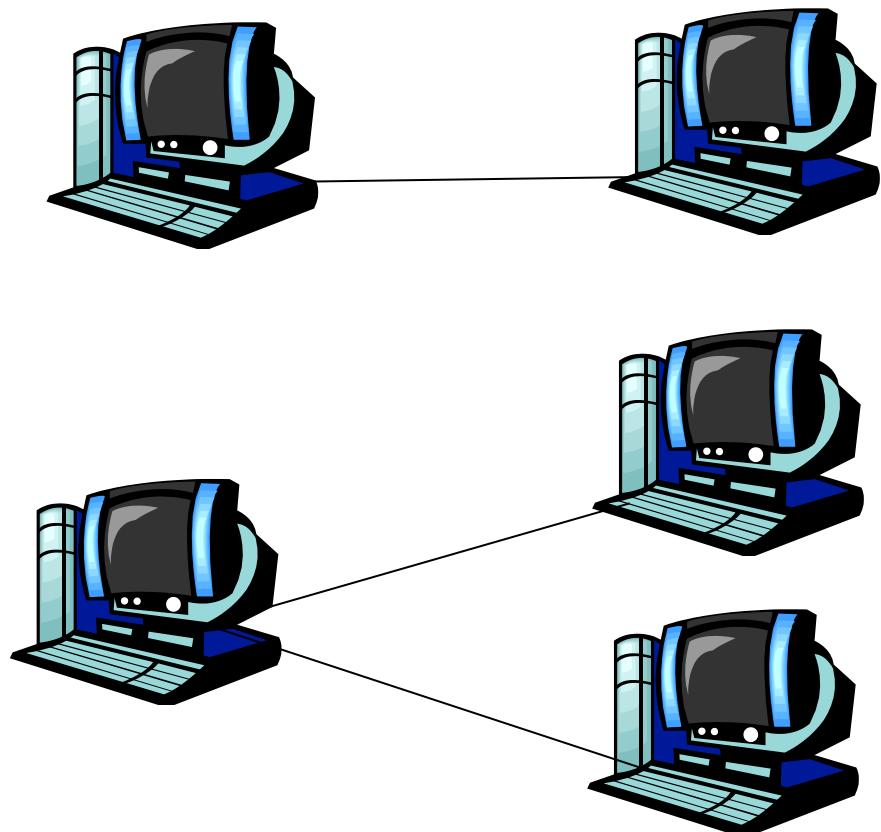
# Classification des réseaux

- Réseaux Métropolitains MAN(Metropolitain Area Network)
  - Couverture: la taille d'une ville
  - Débit: quelques dizaines de Mbits/s
  - FDDI, DQDB, ATM, WiMax,.....
- Réseaux locaux WAN(Wide Area Network)
  - Couverture: Mondiale
  - Débit: quelques Mbits/s
  - RNIS, IP, ATM, WATM, GSM,...

# Topologies

- Logique: le mode d'échange des messages dans le réseau
- Physique: raccordement des machines

Deux types de liaisons:  
point-à-point ou  
multipoints



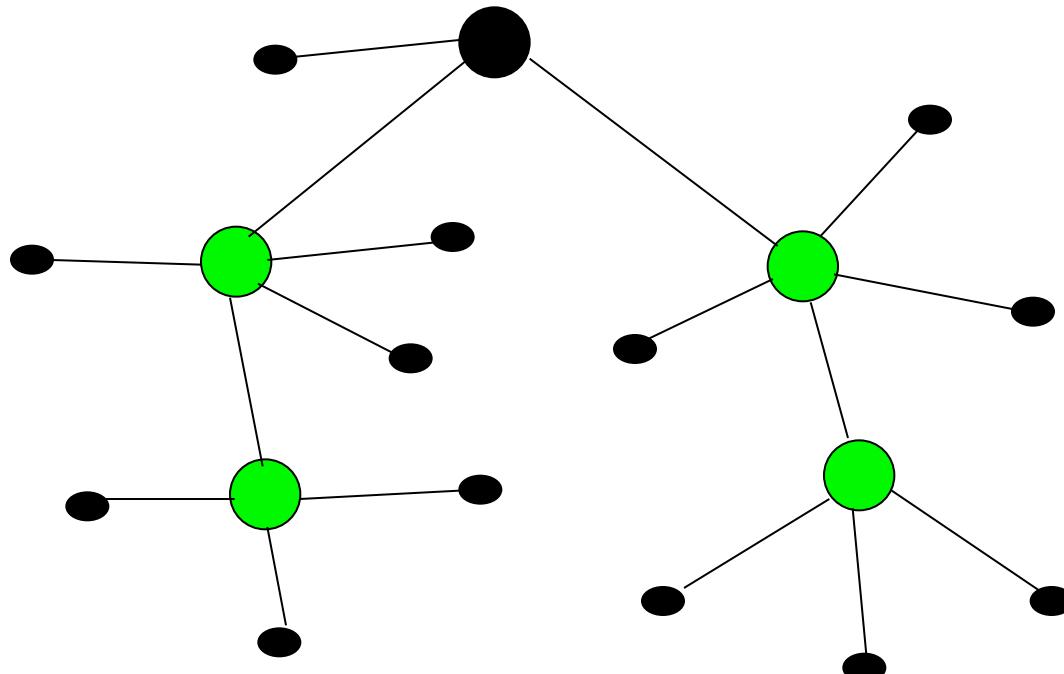
# Topologies (physique)

## Topologies de base

- Bus
- Etoile
- Anneau

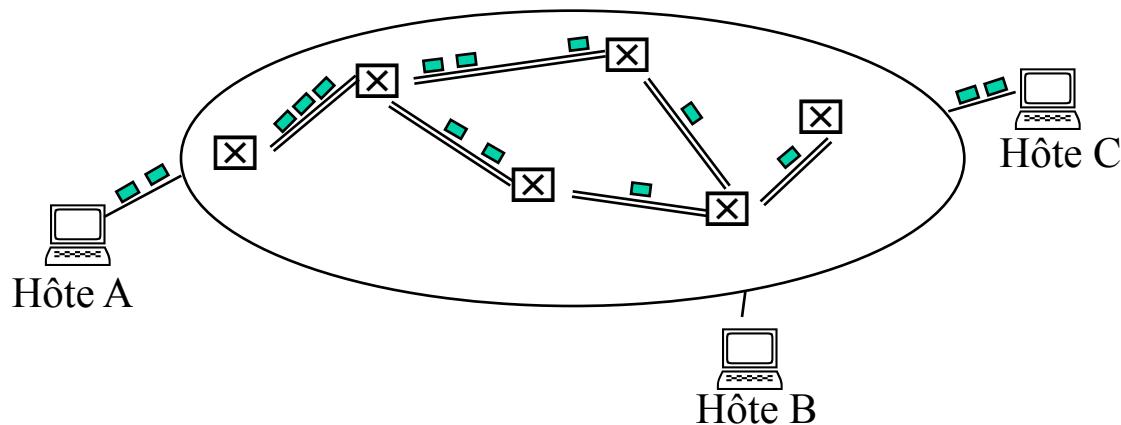
# Topologies (physique)

- Topologies construites: dérivés des réseaux en étoiles



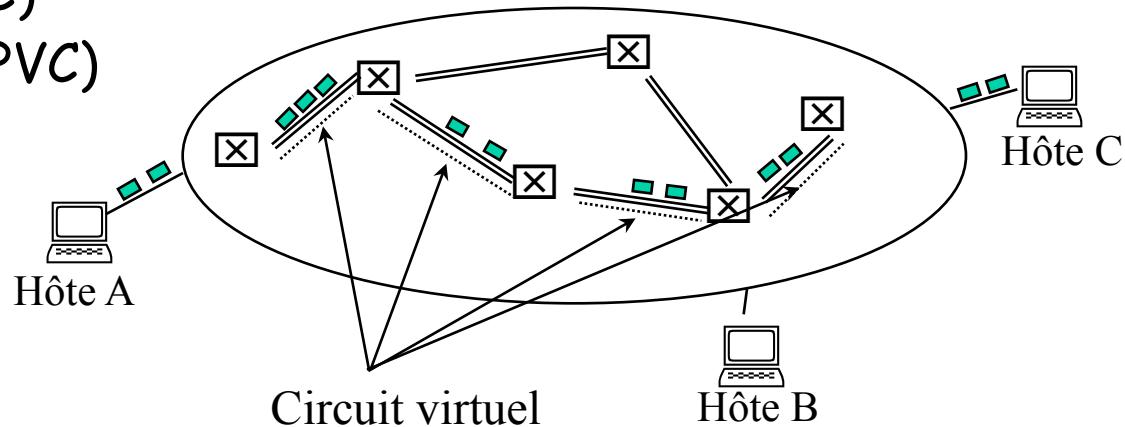
# Modes de mise en relation

- Deux modes de fonctionnement pour transiter les informations
- Mode non connecté (Datagramme)
  - Une seule phase (Transfert des données)
  - Simple
  - Plusieurs chemins possibles



# Modes de mise en relation

- Mode connecté (Circuit virtuel,CV/circuit physique)
  - Etablissement d'une connexion
  - Transfert des données
  - Libération de la connexion
  - Service fiable
  - Complexe
  - Chemin dédié
  - Circuits commutés (SVC)
  - ou Circuits permanent(PVC)



# Modes de mise en relation

- Technique de commutation  
la manière d'interconnecter 2 correspondants
- Le fonctionnement d'un nœud (routeur/switch)
- Nombre de liens= $N(N+1)/2$  (N: nombre de nœuds)
- Temps de traversée du réseau T<sub>p</sub>

$$T_p = (L + pH)(1 + N/p)/D$$

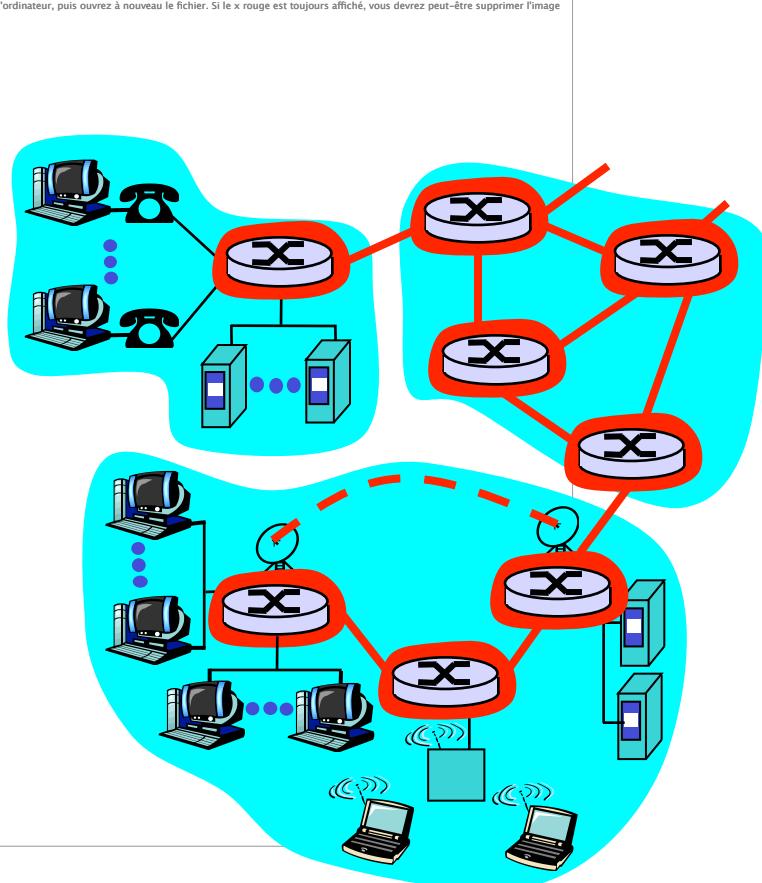
L: longueur de message, N:nombre de nœuds,  
p: nombre de paquets,H: entête protocole, D: débit

# Modes de mise en relation

- Techniques de commutation
  - Commutation de circuit ( $p=1, N=0$ )
  - Commutation de messages( $p=1, N>0$ )
  - Commutation de paquets
  - Commutation de trames
  - Commutation de cellules

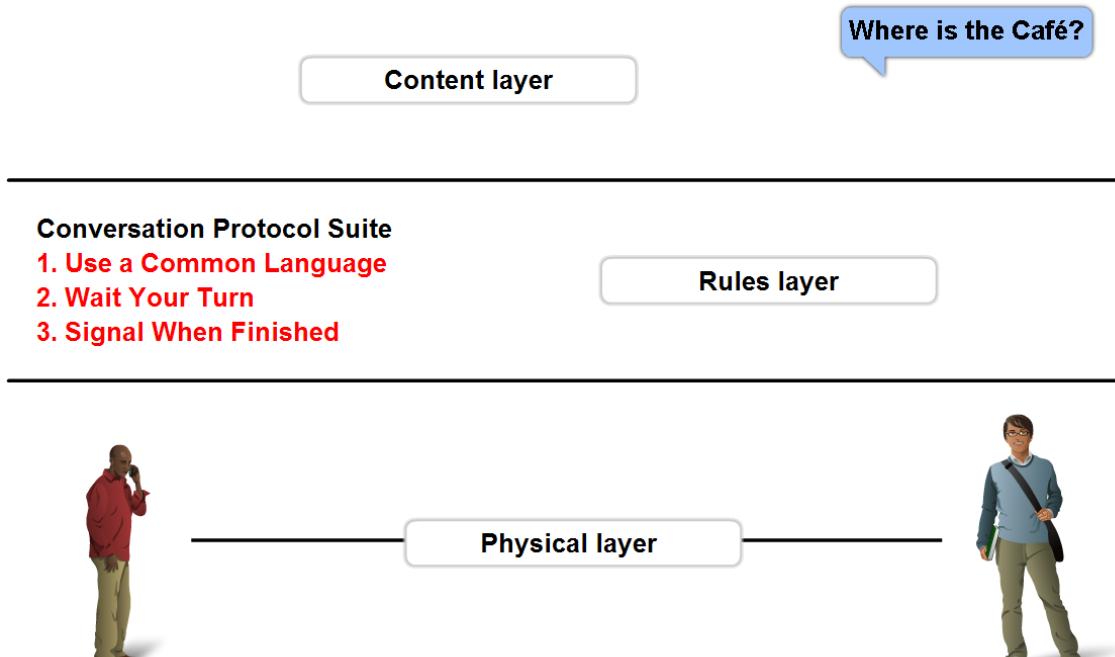
# Le réseau cœur

- Réseau maillés de routeurs
- La question fondamentale:***  
comment les données sont transférées à travers le réseau ?
  - Commutation de circuits:**  
circuit dédié par appel:  
réseau téléphonique
  - Commutation par paquets:**  
données envoyées sur le réseau par "morceaux"



# Fonction du Protocole dans le réseau de communication

- L'importance des protocoles et comment ils sont utilisés pour faciliter la communication sur les réseaux de données
  - Un protocole est un ensemble de règles prédéterminées



# Fonction du Protocole dans le réseau de communication

- Les protocoles réseau sont utilisés pour permettre aux périphériques de communiquer avec succès

## Protocols provide:

The format or structure of the message

The process by which networking devices share information about pathways to other networks

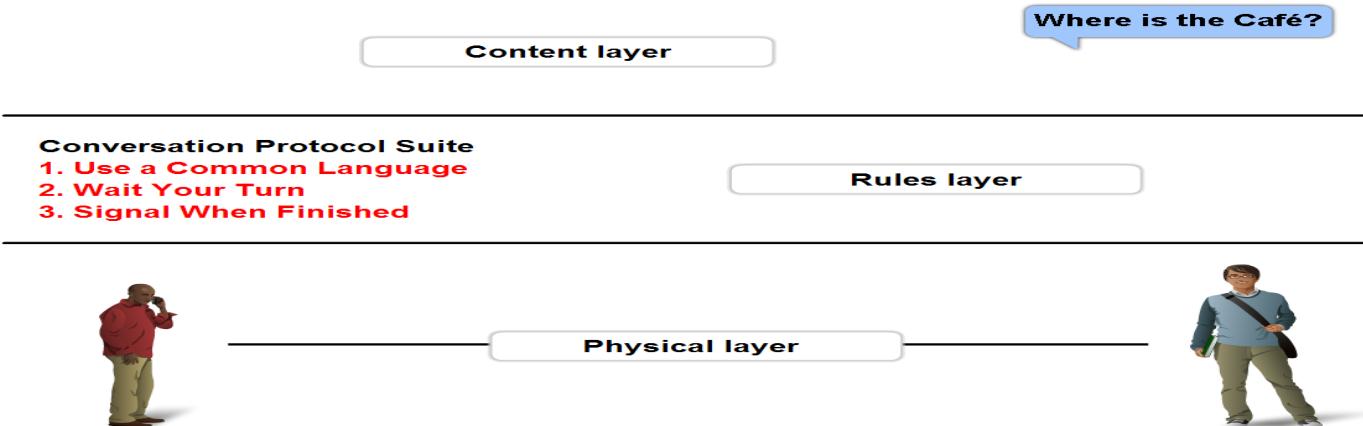
How and when error and system messages are passed between devices

The setting up and termination of data transfer sessions

# Fonction du Protocole dans le réseau de communication

## □ Les suites du protocole et les normes de l'industrie

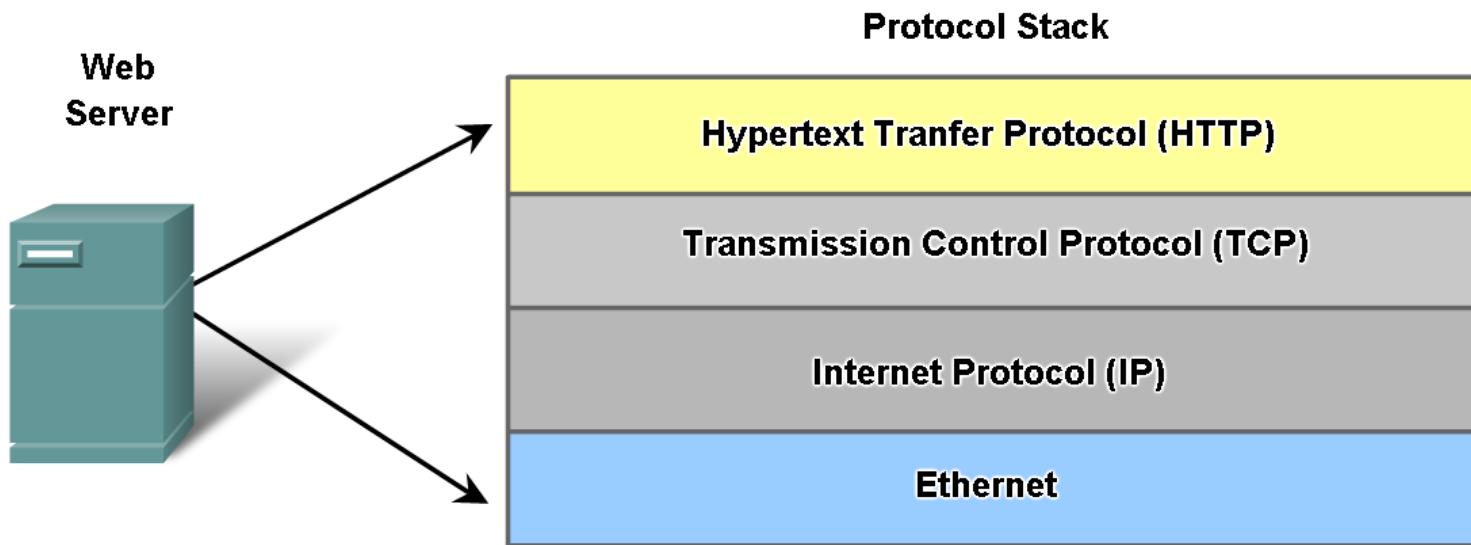
Protocol Suites are sets of rules that work together to help solve a problem.



- Une norme est un processus ou un protocole qui a été approuvé par l'industrie des réseaux et ratifié par un organisme de normalisation

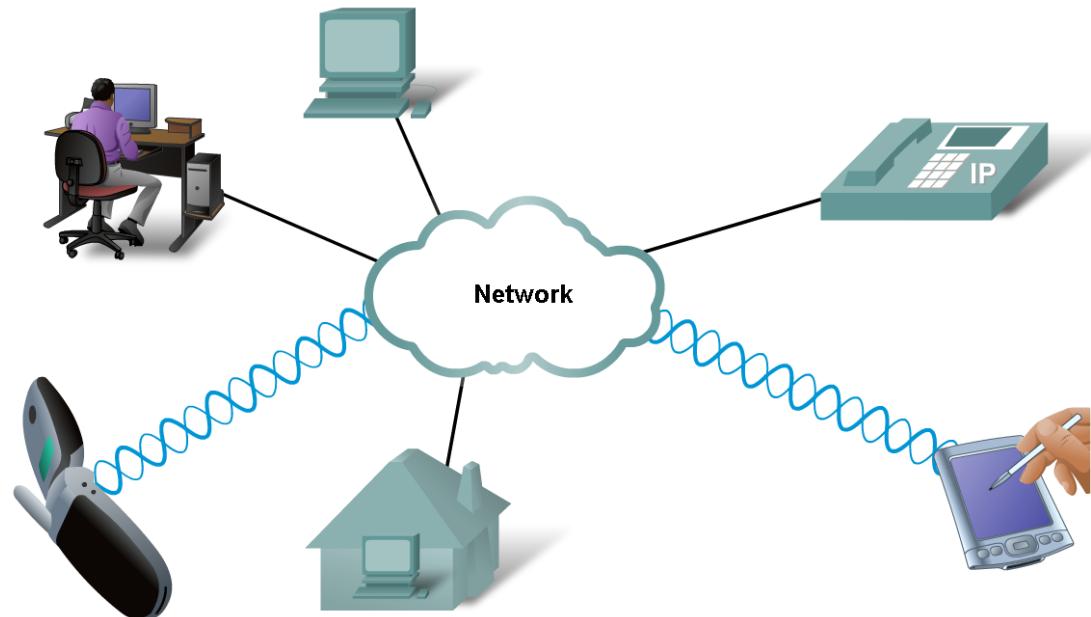
# Fonction du Protocole dans le réseau de communication

## □ Interactions entre protocoles



# Fonction du Protocole dans le réseau de communication

- Indépendance du matériel (dispositif)
  - Beaucoup de divers types de dispositifs peuvent communiquer en utilisant les mêmes ensembles de protocoles. C'est parce que les protocoles spécifient les fonctionnalités du réseau, pas la technologie sous-jacente à la charge cette fonctionnalité.



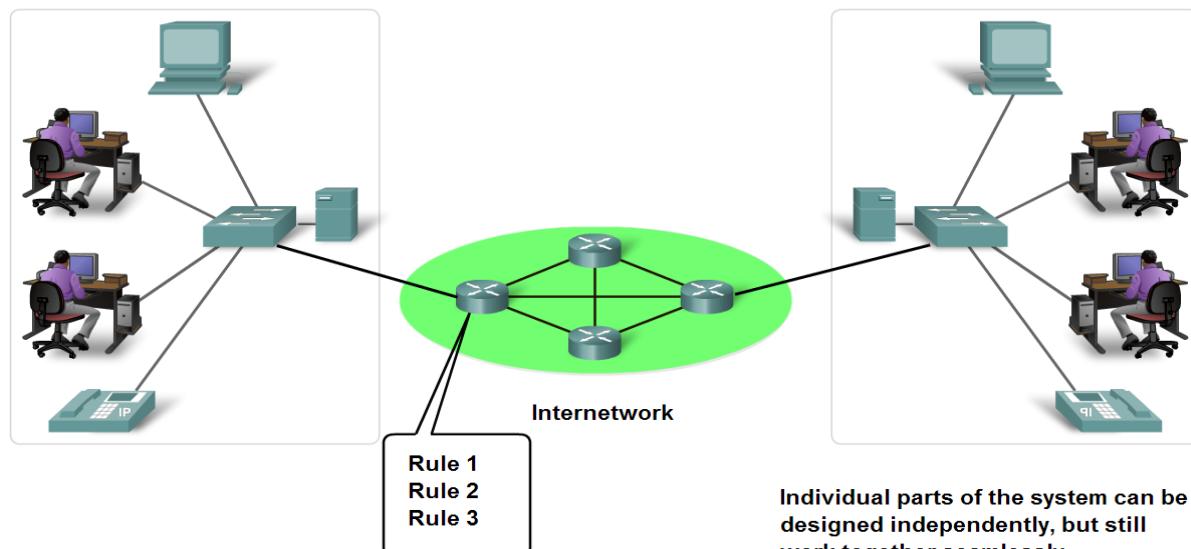
# Couches avec TCP/IP et le Modèle OSI

## □ Les avantages d'utiliser un modèle en couches

### ○ Les avantages comprennent

- aide à la conception du protocole
- favorise la concurrence
- des changements dans une couche n'affectent pas les autres couches
- fournit un langage commun

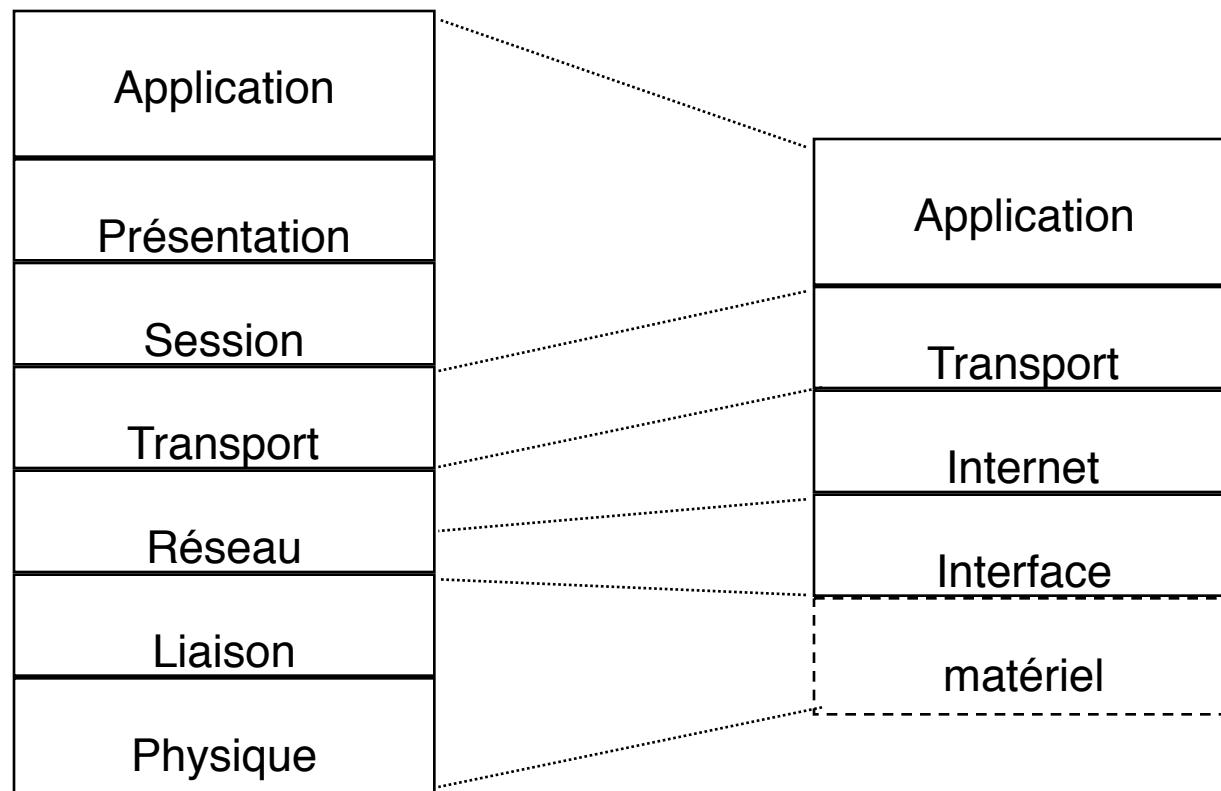
Using a layered model helps in the design of complex, multi-use, multi-vendor networks.



# Les couches de protocoles : modèle OSI

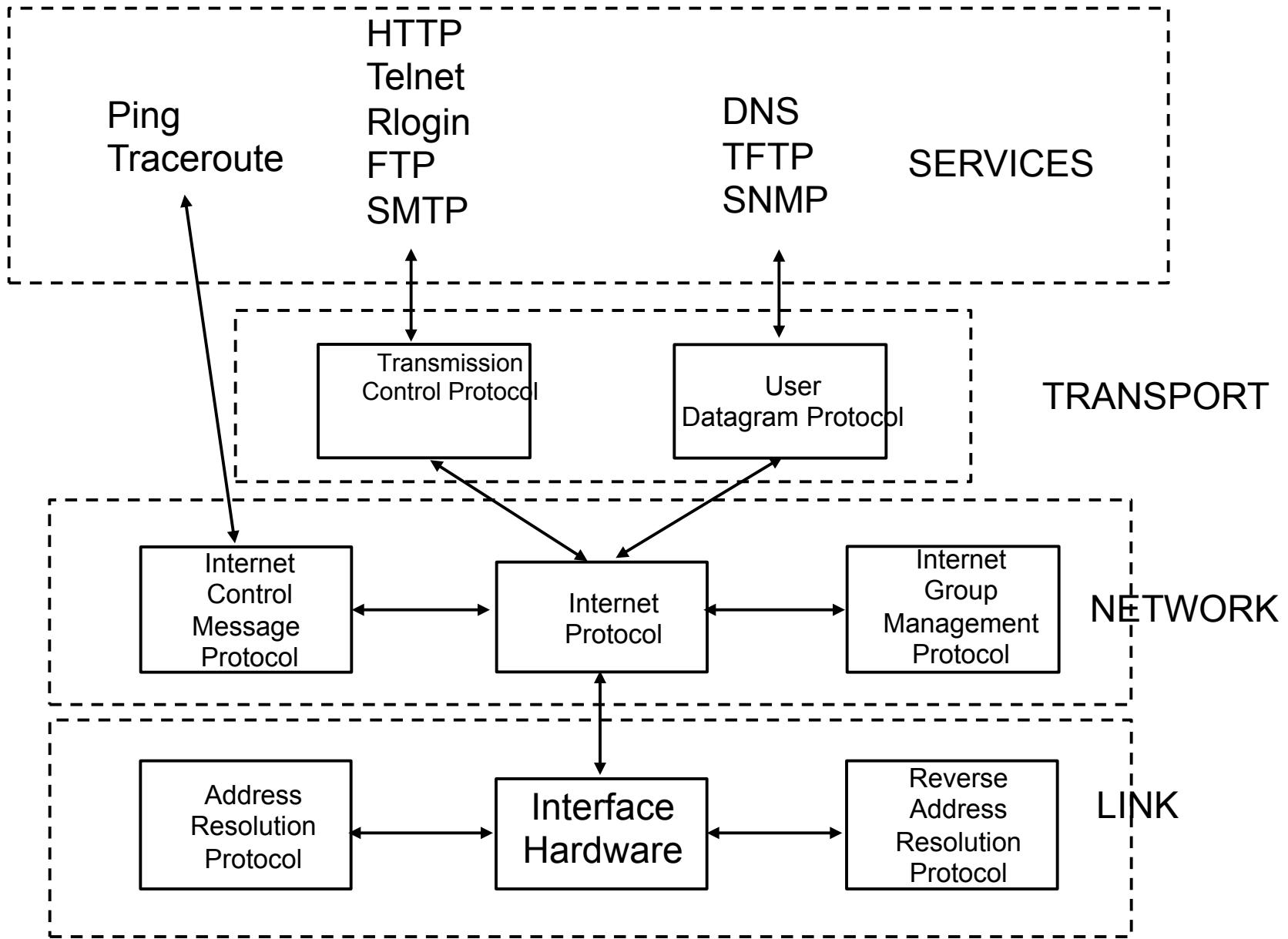
Couches OSI	Fonctions
Application	Applications réseaux : transfert de fichiers.
Présentation	Formatage et cryptage des données
Session	Etablissement et maintien des sessions
Transport	Transport de bout en bout, fiable et non fiable
Réseau	Envoi et routage des paquets de données
Liaison	Transfert des unités d'informations et contrôle d'erreurs
Physique	Transmission des données binaires

# Les couches de protocoles : TCP/IP et le modèle OSI



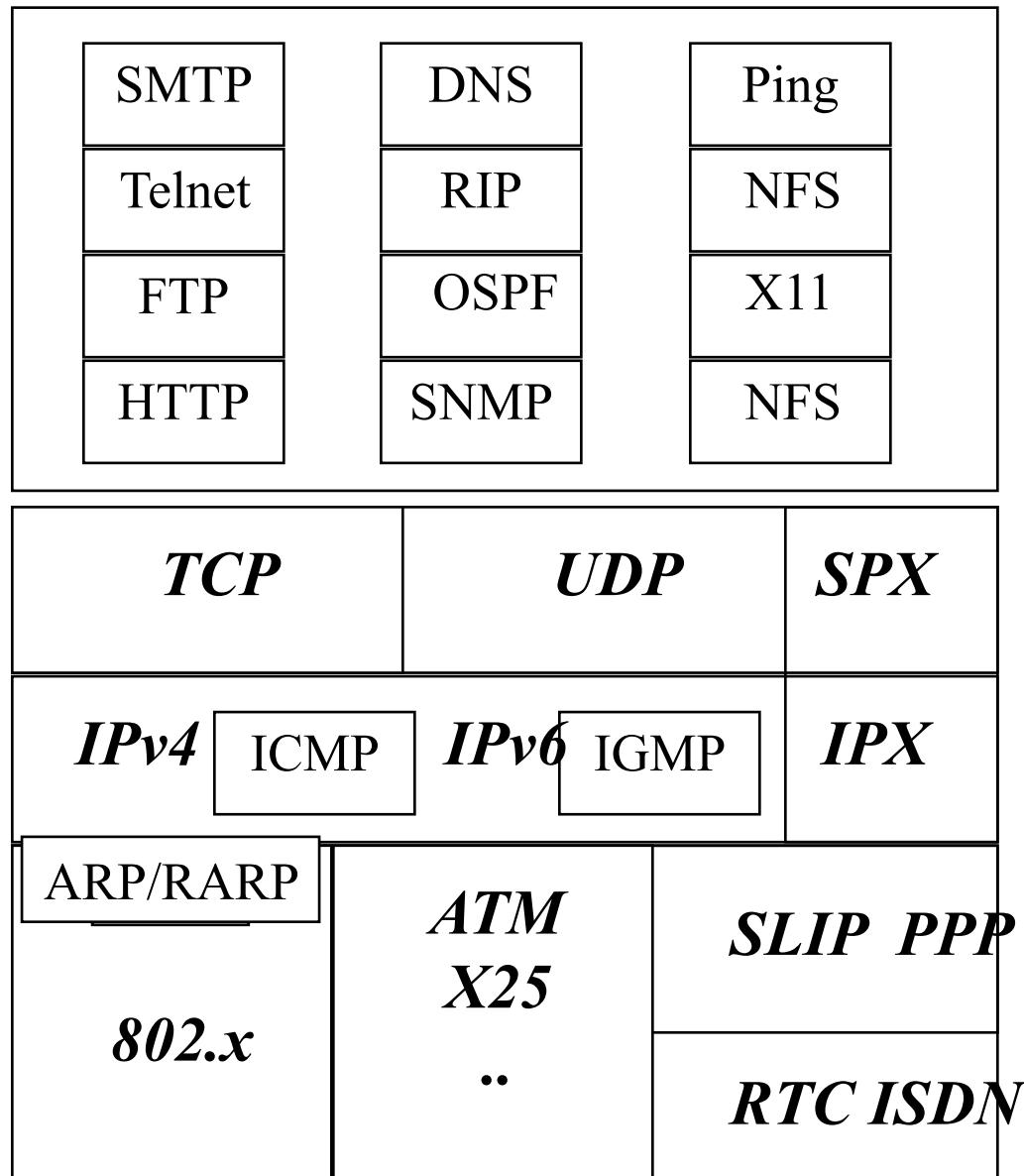
# Les couches de protocoles : TCP/IP et le modèle OSI

<u>Protocol Implementation</u>							<u>OSI</u>
File Transfer	Electronic Mail		Terminal Emulation	File Transfer	Client Server	Network Mgmt	Application
File Transfer Protocol (FTP) RFC 559	Simple Mail Transfer Protocol (SMTP) RFC 821	TELNET Protocol RFC 854	Trivial File Transfer Protocol (TFTP) RFC 783	Network File System Protocol (NFS) RFC 1024, 1057 and 1094	Simple Network Management Protocol (SNMP) RFC 1157	Presentation	Presentation
							Session
Transmission Control Protocol (TCP) RFC 793			User Datagram Protocol (UDP) RFC 768			Transport	
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791			Internet Control Message Protocol (ICMP) RFC 792		Network	
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS	Data Link	
Transmission Mode						Physical	
TP	STP	FO	Satellite	Microwave, etc			



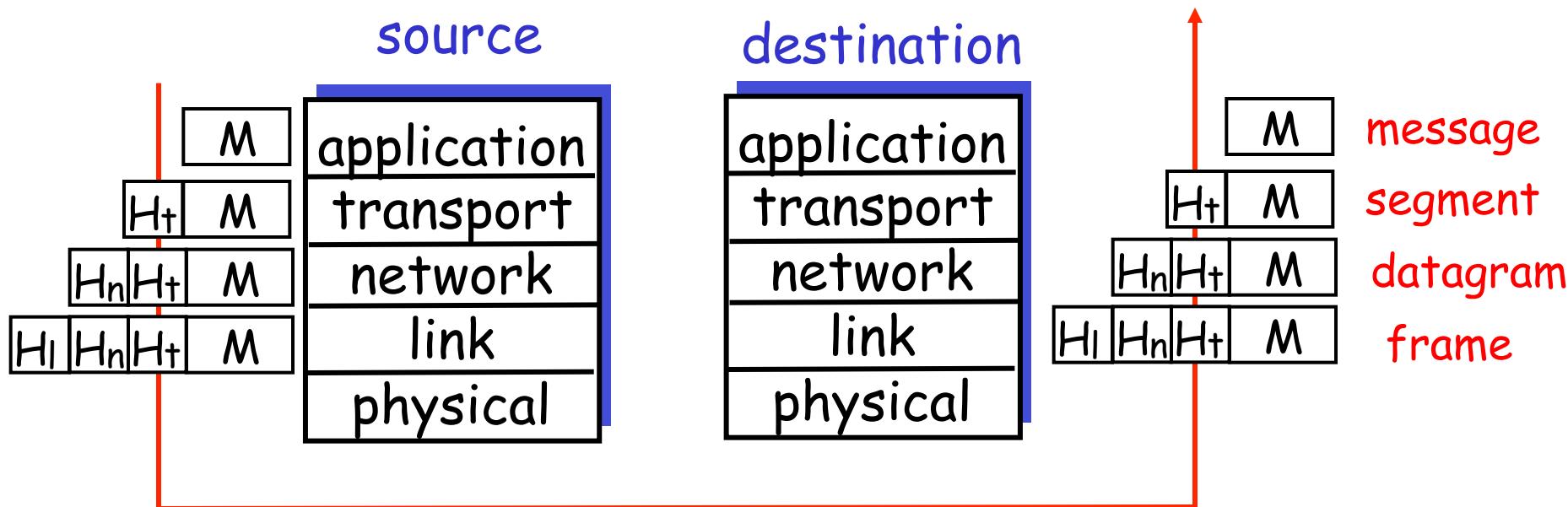
## TCP/IP Protocole Suite

# TCP/IP Protocole Suite



# Les couches et les données

## Encapsulation/Décapsulation



# Partie 2: Les couches réseau



# Couche Applications et Services

# Plan

- Principes des protocoles de la couche Applications
- DNS
- Electronic Mail
  - o SMTP, POP3, IMAP
  - Web et HTTP
  - FTP

# Les couches de protocoles : TCP/IP et le modèle OSI

<u>Protocol Implementation</u>							<u>OSI</u>
File Transfer File Transfer Protocol (FTP) RFC 559	Electronic Mail Simple Mail Transfer Protocol (SMTP) RFC 821	Terminal Emulation TELNET Protocol RFC 854	File Transfer Trivial File Transfer Protocol (TFTP) RFC 783	Client Server Network File System Protocol (NFS) RFC 1024, 105 and 1094	Network Mgmt Simple Network Management Protocol (SNMP) RFC 1157	Application Presentation	Session
Transmission Control Protocol (TCP) RFC 793				User Datagram Protocol (UDP) RFC 768		Transport	
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791	Internet Group Management Protocol (IGMP) RFC 2236		Internet Control Message Protocol (ICMP) RFC 792	Network		
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS	Data Link	
Transmission Mode TP STP FO Satellite Microwave, etc						Physical	

# Applications réseau

**Processus:** programme s'exécute sur une host.

- sur la même host, deux processus interagissent en utilisant **la communication interprocessus** (règle définie par l'OS)
- Processus sur deux machines différentes communiquent par message à travers un **protocole de la couche application**

**Agent utilisateur:** interfaces avec l'utilisateur au dessus et le réseau en dessous

- Implémentations interface utilisateur et le protocole du niveau application
  - Web: browser
  - E-mail: mail reader
  - streaming audio/vidéo: media player

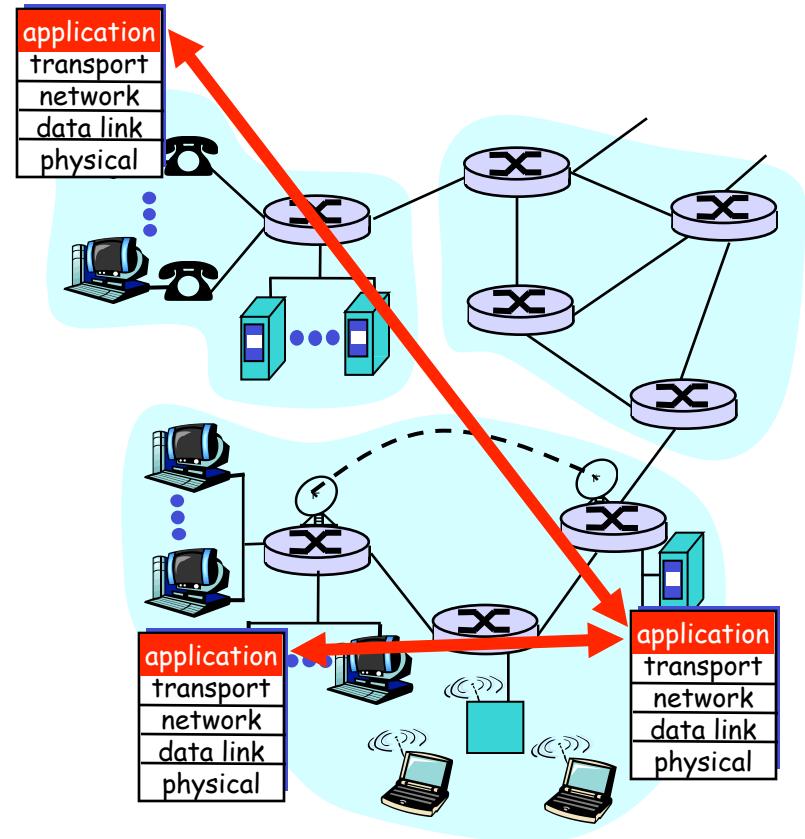
# Applications et protocoles de la couche application

## Application: processus distribués

- e.g., e-mail, Web, telnet
- s'exécutant sur les terminaux (hosts)
- échangent des messages pour implémenter l'application

## Protocoles de la couche application

- définissent des messages échangés par les applications et les actions prises
- utilisent les services de communication fournis par les protocoles de la couche inférieure (TCP, UDP)



# Protocole de la couche application

- Types de messages échangés, ex, messages de demande et de réponse
  - la syntaxe adoptée par les différents types de message: soit les différents champs qu'il contient et leur délimitation
  - la sémantique des différents champs, c'est à dire le sens des informations qu'ils renferment
- les règles utilisées pour déterminer quand et comment un processus doit envoyer ou répondre à un message
- Protocoles domaines publics:**
- définis dans les RFCs
  - Permettent l'interopérabilité
  - ex, HTTP, SMTP
- Protocoles propriétaires:**
- ex, KaZaA

# Paradigme Client-server

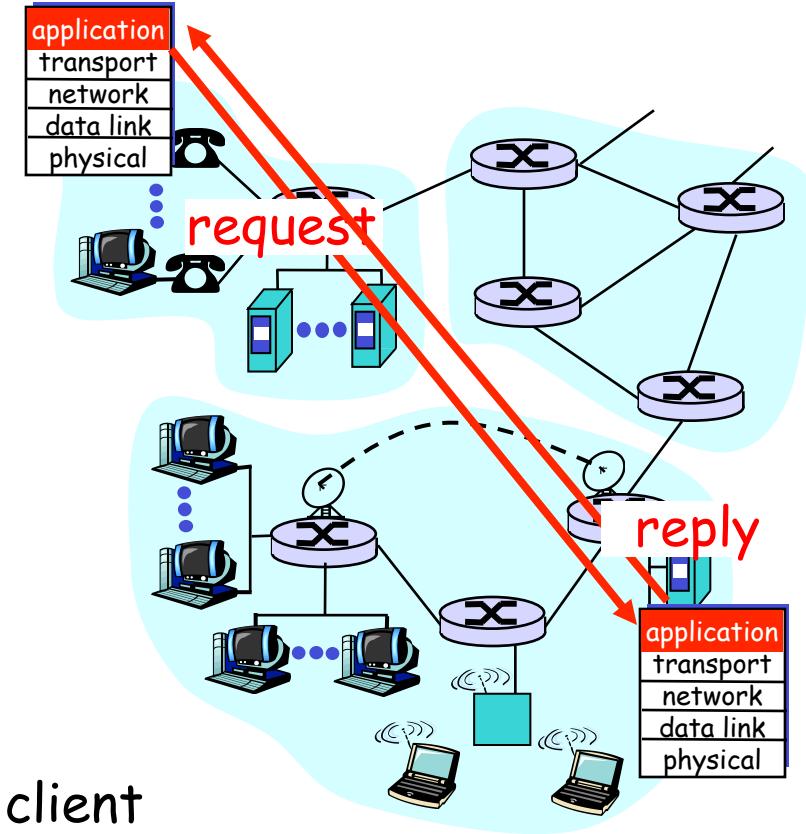
Application réseau contient deux parties: *client* et *serveur*

## Client:

- initialise le contact avec le serveur
- Demande de service de serveur
- Web: client implementé dans le browser; e-mail: dans mail reader

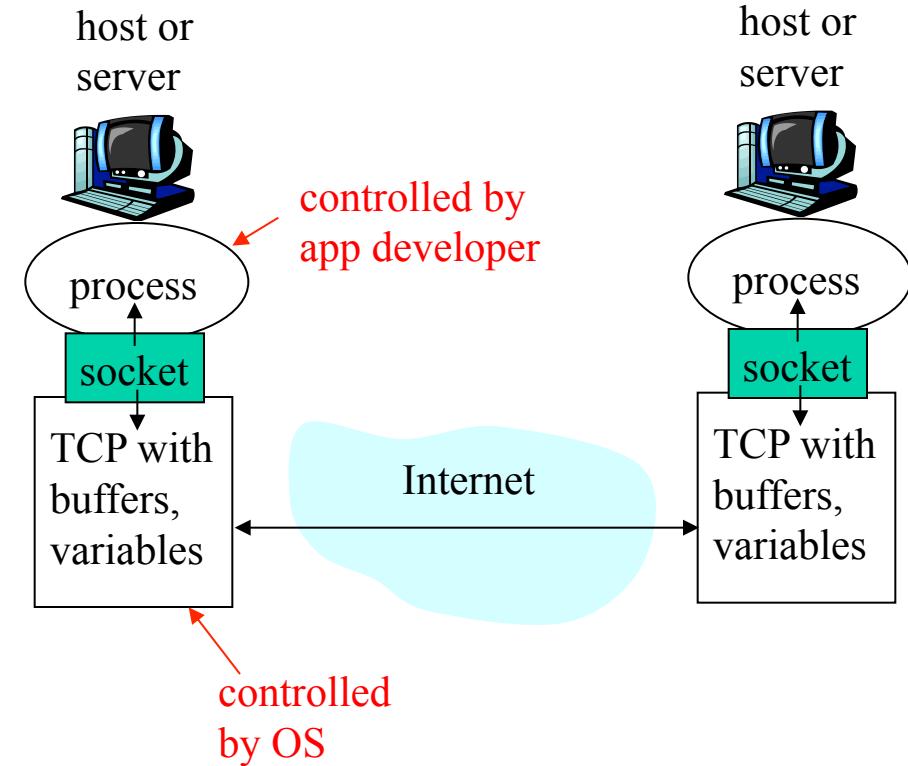
## Serveur:

- fournit le service demandé par le client
- e.g., Web server envoie la page Web demandée, mail server délivre e-mail



# Processus de communication à travers le réseau

- processus reçoit / envoie les messages à travers son socket
- socket analogue à une porte
  - l'envoie de message à travers cette porte (interface)
  - le processus d'envoi suppose qu'il y à une infrastructure de transport de l'autre coté de cette porte prête à prendre en charge le message et à l'emmener jusqu'à la porte de destinataire via destinataire



- API (1) le choix du protocole de transport; (2) la possibilité de définir quelques paramètres

# Processus d'adressage:

- Un processus local a besoin d'identifier le processus distant
- Chaque host a une unique adresse IP
- Q: l'adresse IP de la machine sur laquelle le processus tourne suffit-elle pour identifier le processus?
- Réponse: Non, plusieurs processus peuvent être exécutés sur la même machine
  - Identificateur inclut l'adresse IP et **le numéro de port associé** sur le host.
  - Exemple port numbers:
    - HTTP server: 80
    - Mail server: 25

# Services nécessaires à une application?

## Data loss (transfer fiable)

- certaines apps (e.g., audio) tolèrent la perte
- autres apps (e.g., file transfer, telnet) exigent 100% transfert fiable

## Contraintes de temps

- certaines apps (e.g., Internet telephony, interactive games) demandent un bas délai

## Bandwidth (débit)

- certaines apps (e.g., multimédia, téléphonie par internet) exigent un débit minimal disponible
- autres apps ("elastic apps, comme ftp et web") peuvent s'adapter aux débits disponibles

# Fonctionnement de quelques applications de réseau

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:	yes, 100's msec
stored audio/video	loss-tolerant	10kbps-5Mbps	yes, few secs
interactive games	loss-tolerant	same as above	yes, 100's msec
instant messaging	no loss	few kbps up elastic	yes and no

## Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary	typically UDP

# DNS: Domain Name System

Personne: plusieurs manières de l' identifier:

- NSS, nom, N.passeport

Internet hosts, routeurs:

- IP address (32 bit)
- "nom",  
gaia.cs.umass.edu  
utilisé par le humains

Q: correspondance entre adresses IP et nom?

Domain Name System:

- *database distribuée* implémentée en hiérarchie dans plusieurs *name servers*
- *protocole couche application* host, routeurs, name servers communiquent pour *résoudre* noms (translation adresse/ name)

# DNS name servers

Pourquoi le DNS n'est pas centralisé?

- Un seul point de rupture
- volume de trafic
- database centralisée distante
- maintenance

n'est pas *scalable!*

- Pas de serveur à toutes les correspondances nom-adresse IP

**name servers locaux:**

- chaque ISP, a son *local (default) name server*
- host DNS consulte en premier le name server local

**name server de source autorisée:**

- pour une host: stocke son adresse IP, nom
- peut accomplir la translation name/address IP

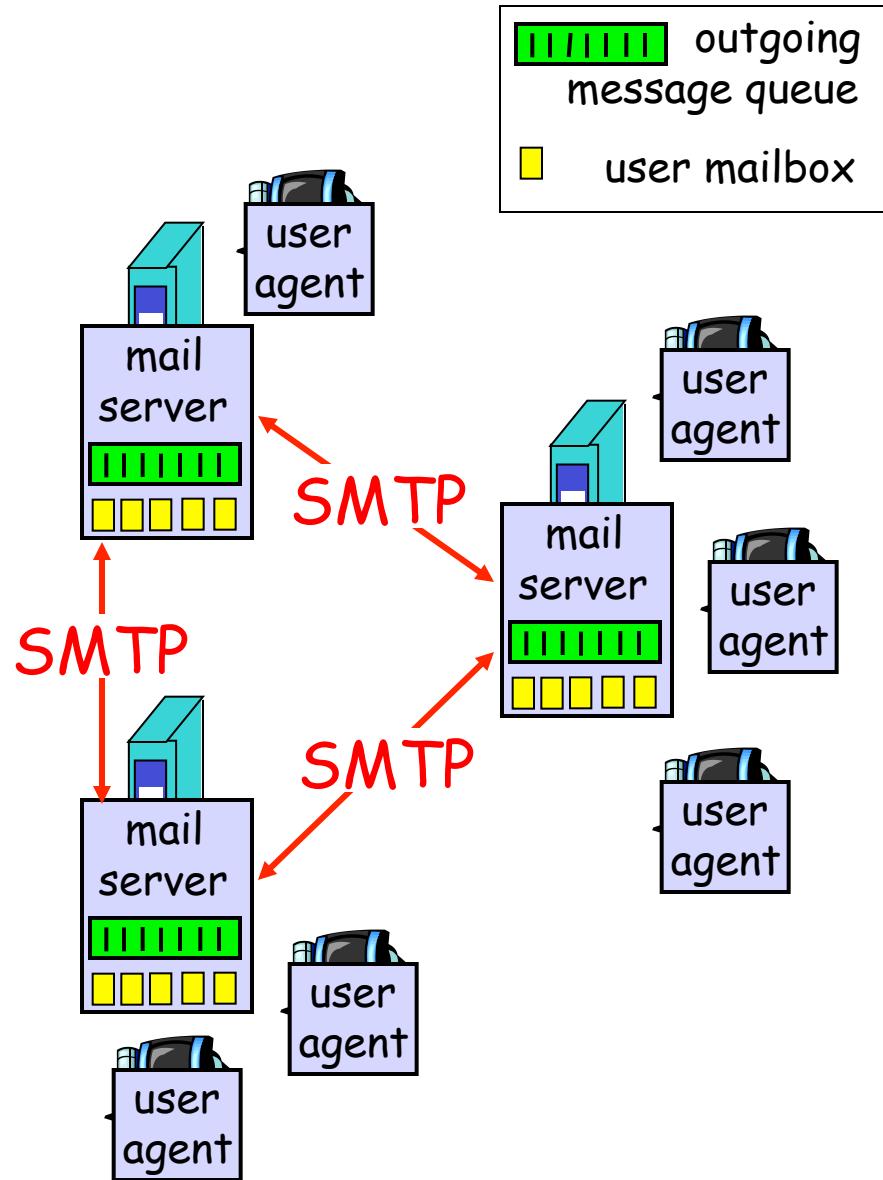
# Electronic Mail

Trois éléments fondamentaux:

- Agents utilisateurs
- serveurs de messagerie
- Le protocole SMTP: simple mail transfer protocol

Agent utilisateur

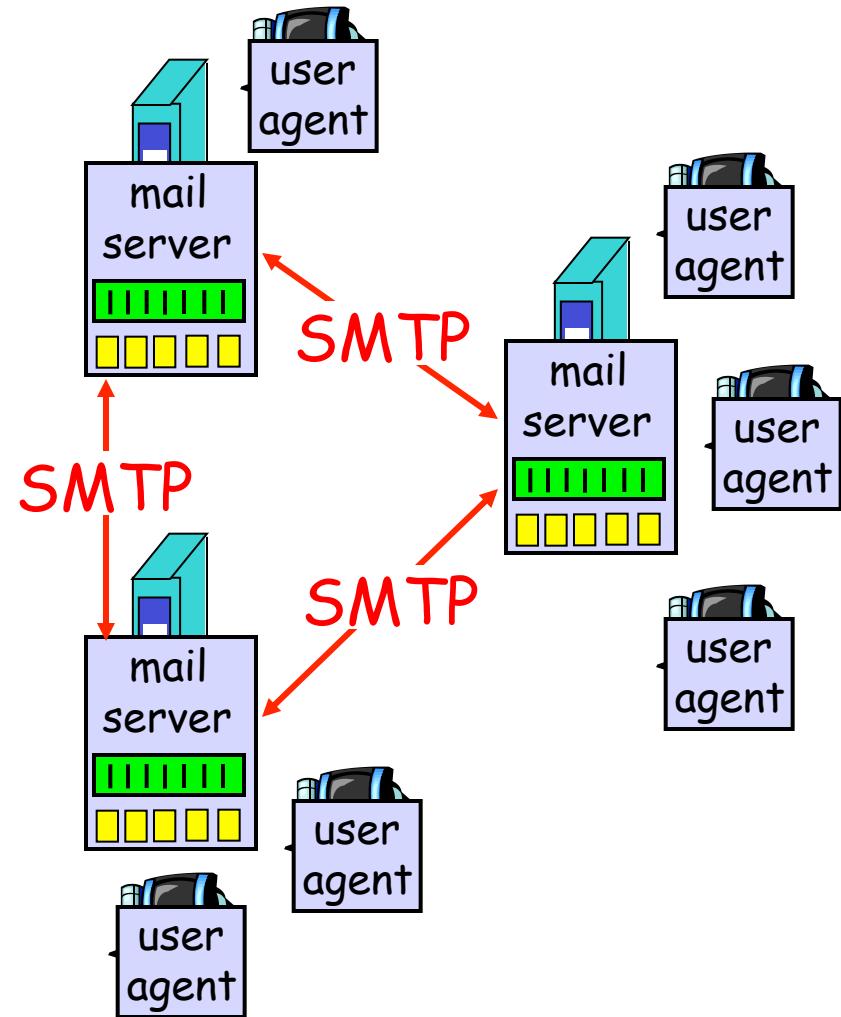
- Appelé aussi lecteur de messagerie, éditeur....
- exemples, Eudora, Outlook, elm, Netscape Messenger
- Messages sortant, entrant sont stockés sur le serveur



# Electronic Mail: mail servers

## Serveurs de messagerie

- **mailbox** contient des messages entrants pour l'utilisateur
- **File de messages** de messages sortants (d'être envoyés)
- **SMTP protocol** pour envoyer les messages entre les serveurs
  - client: sending mail server
  - "server": receiving mail server

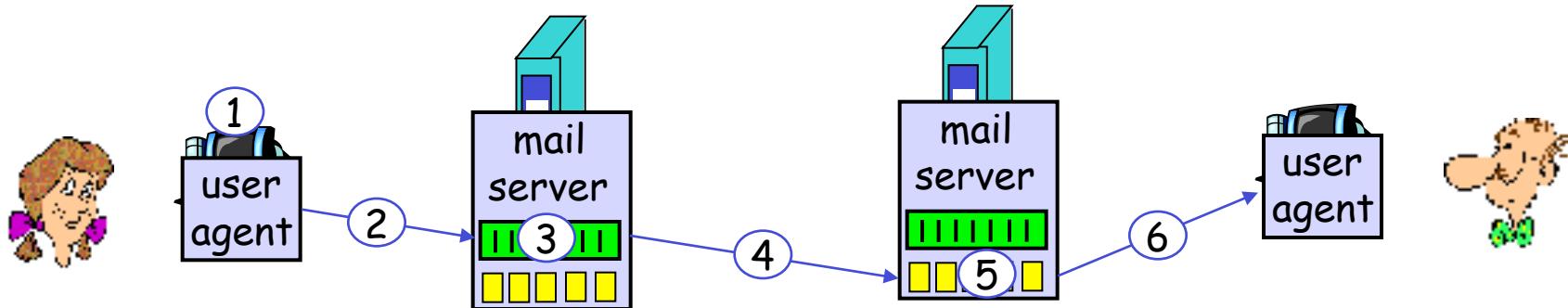


# Electronic Mail: SMTP [RFC 2821]

- utilise TCP pour transférer d'email de client au serveur, port 25
- Transfert direct: serveur d'envoi au serveur de réception
- trois phases de transfert
  - La connexion
  - Transfert de messages
  - fermeture
- Interaction commande/réponse
  - commandes: ASCII
  - réponses: code d'état et phrase
- messages en ASCII de 7 bits

# Scénario: Alice envoie un message à Bob

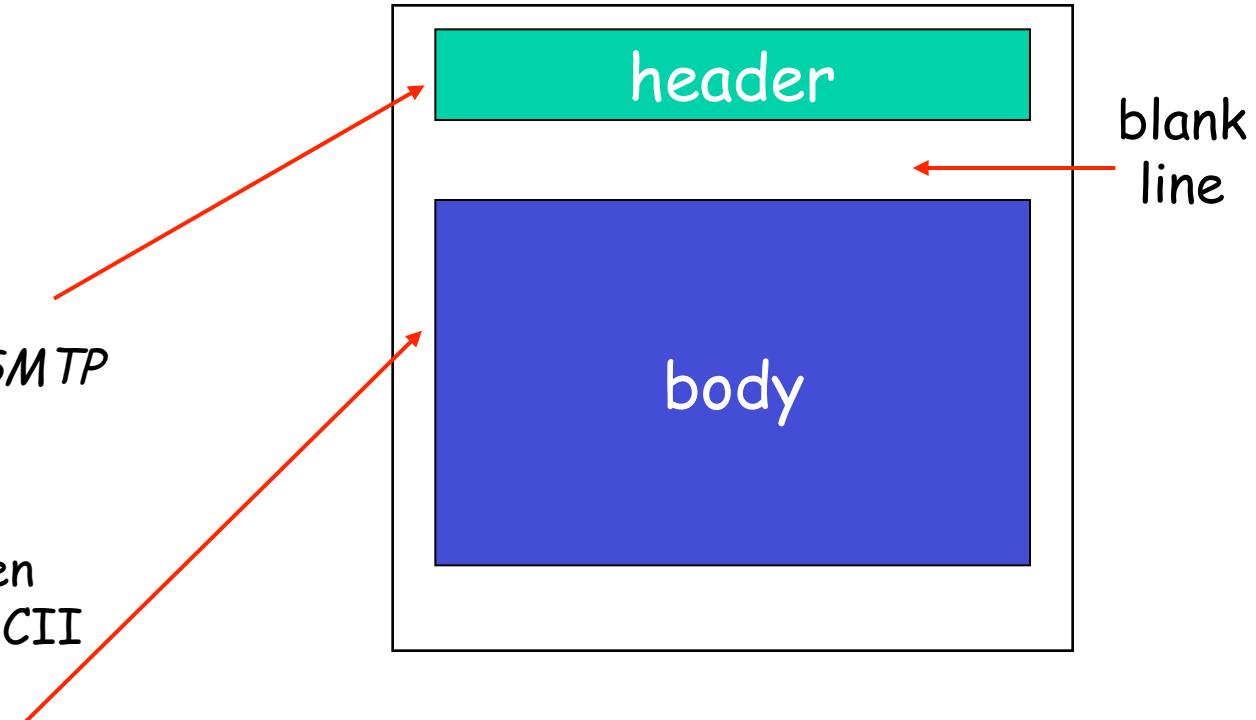
- 1) Alice utilise un UA pour composer le message "to" bob@someschool.edu
- 2) UA d'Alice envoie le message à son serveur de messagerie, où il est placé dans une file de messages
- 3) Le pôle client de SMTP opérant au niveau de serveur de messagerie d'Alice, aperçoit le message dans la file, ouvre une connection TCP avec le serveur mail de Bob
- 4) SMTP client envoie le message d'Alice sur la connexion TCP
- 5) Serveur mail de Bob place le message dans le mailbox de Bob
- 6) Bob ouvre son user agent pour lire le message



# Format de message Mail: texte

SMTP: RFC 822:

- Ligne d'en-tête,
  - To:
  - From:
  - Subject:  
*different from SMTP commands!*
- body
  - le "message", en caractères ASCII seulement



# Format de message: multimedia extensions

- ❑ MIME: multipurpose internet mail extension, RFC 2045, 2056
- ❑ additional lines in msg header declare MIME content type

MIME version  
method used to encode data  
multimedia data type, subtype, parameter declaration  
encoded data

From: alice@crepes.fr  
To: bob@hamburger.edu  
Subject: Picture of yummy crepe.  
MIME-Version: 1.0  
Content-Transfer-Encoding: base64  
Content-Type: image/jpeg

base64 encoded data .....  
.....  
.....base64 encoded data

# MIME types

Content-Type: type/subtype; paramètres

## Text

- exemple subtypes: plain, html

## Image

- exemple subtypes: jpeg, gif

## Audio

- exemple subtypes: basic (8-bit mu-law encoded), 32kadpcm (32 kbps coding)

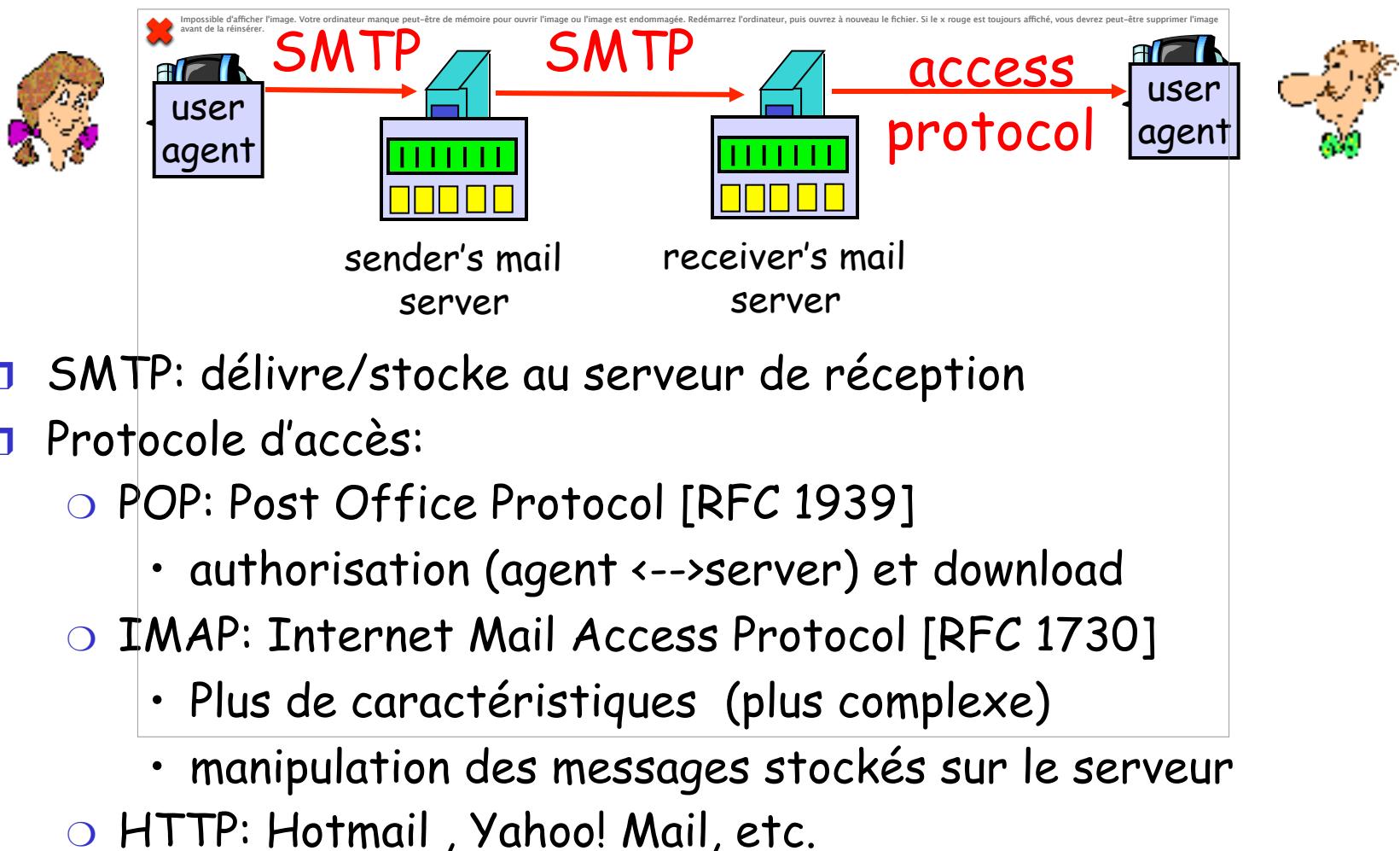
## Vidéo

- exemple subtypes: mpeg, quicktime

## Application

- Le type application est utilisé pour les données ne correspondant pas à aucune autre catégorie, notamment celle qui devait être soumises à une application particulière avant d'être accessibles à l'utilisateur
- exemple subtypes: msword, octet-stream

# Protocoles d'accès à la messagerie



# Web et HTTP

- Web page composée des objets
- Objet peut être fichier HTML, image JPEG, applet Java, fichier audio,...
- Web page composée de base HTML-file qui inclut plusieurs objets référencés
- Chaque objet est adressé par un URL
- Exemple URL:

www.someschool.edu/someDept/pic.gif

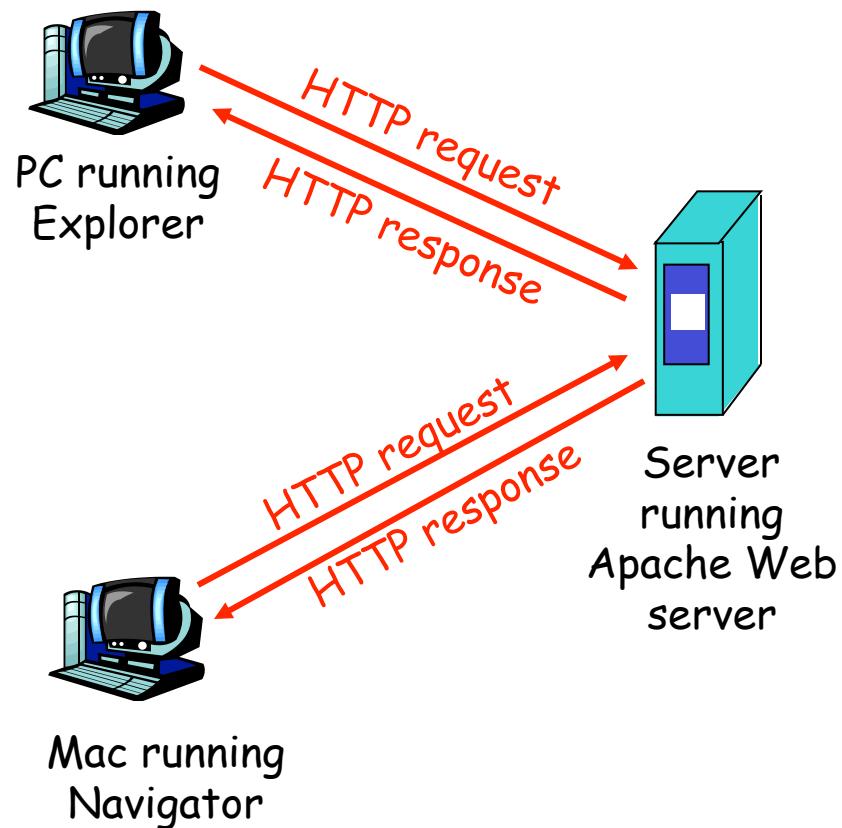
host name

path name

# HTTP

## HTTP: hypertext transfer protocol

- Protocole de la couche application Web
- Modèle client/server
  - *client*: browser qui demande et reçoit, "displays" les objets Web
  - *server*: Web serveur envoie les objets demandés
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



# HTTP

## Utilise TCP:

- client initialise TCP connection (crée une socket) au serveur, port 80
- serveur accepte TCP connection de client
- messages HTTP (application-layer protocol messages) échangés entre browser (HTTP client) et Web server (HTTP server)
- TCP ferme la connection

## HTTP protocole sans mémoire

- serveur ne maintient pas l'information concernant les demandes passées du client

## Protocoles avec mémoire sans complexes!

- Historique (state) doit être maintenu
- si server/client crashent, on garde une image de l'historique pour re-établir l'état d'avant

# HTTP connections

## Non-persistant HTTP

- Un seul objet Web peut être transféré à la fois sur une connexion TCP
- HTTP/1.0 utilise des connexions non persistantes

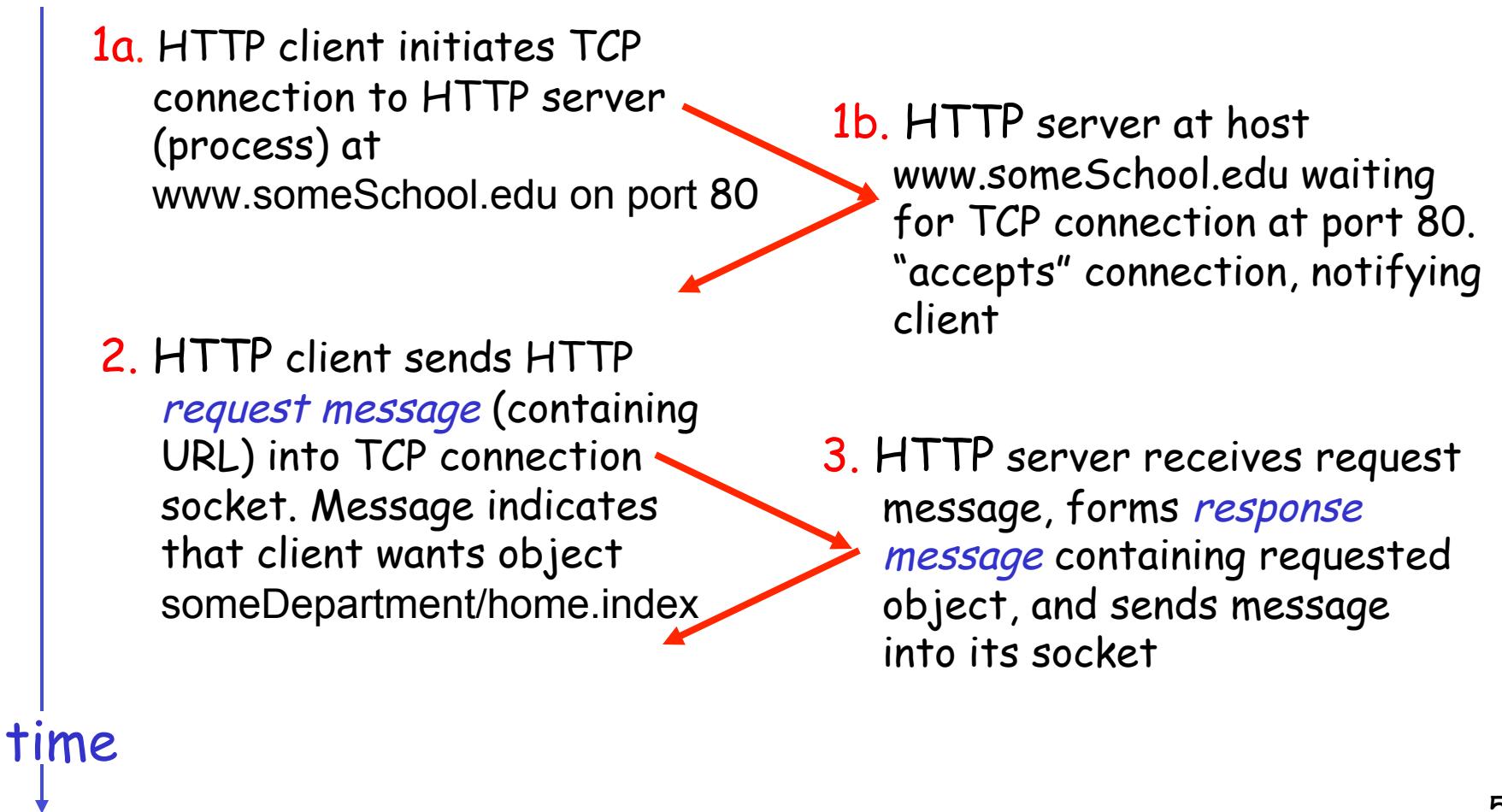
## Persistant HTTP

- Multiple objets peuvent emprunter la même connexion TCP
- HTTP/1.1 utilise les connexions persistants par défaut avec pipelining

# Non persistant HTTP

Supposons l'utilisateur entre l' URL suivant:  
www.someSchool.edu/someDepartment/home.index

(contient text,  
référence 10  
Image jpeg )



# Non persistant HTTP

time  
↓

4. HTTP server closes TCP connection.
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
6. Steps 1-5 repeated for each of 10 jpeg objects

# Persistant HTTP

## HTTP non persistant demande:

- 2 RTTs par objet
- Le système devrait travailler et alloue des ressources pour chaque TCP connection
- mais browser souvent ouvre des connections parallèles pour traiter des objets référencés

## Persistant HTTP

- serveur maintient la connexion ouvert après l'envoi de la réponse
- les messages successifs, entre le même couple client/server sont envoyés sur la connection

## Persistant sans pipelinage:

- client envoie une nouvelle demande seulement quand il reçoit une réponse de la requête précédente
- un RTT pour chaque objet référencé

## Persistant avec pipelinage:

- Par défaut dans HTTP/1.1
- client envoie des requêtes dès qu'il rencontre une référence objet
- un seul RTT pour tous les objets référencés

# HTTP: message de demande

- deux types de messages HTTP: *demande, réponse*
- **HTTP message de demande:**
  - ASCII (format humain lisible)

Ligne de demande  
(GET, POST,  
HEAD commands)

lignes  
d'en-tête

GET /somedir/page.html HTTP/1.1  
Host: www.someschool.edu  
User-agent: Mozilla/4.0  
Connection: close  
Accept-language:fr

# Types de méthodes

## HTTP/1.0

- GET
- POST
- HEAD
  - Demande au serveur de quitter l'objet demandé après la réponse

## HTTP/1.1

- GET, POST, HEAD
- PUT
  - Permet le chargement d'un objet vers un chemin spécifié dans le champ URL
- DELETE
  - Permet d'effacer un objet (fichier) hébergé sur un serveur web

# HTTP: message de réponse

Ligne d'état

Code d'état

Message d'état

HTTP/1.1 200 OK

Connection close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998 .....

Content-Length: 6821

Content-Type: text/html

Ligne  
d'en-tête

Squelette  
(corp)

data data data data data ...

# HTTP: codes d'état

Dans la première ligne dans le message de réponse  
server->client

quelques codes d'état:

## **200 OK**

- la requête est réussie et l'information est contenue dans la réponse

## **301 Moved Permanently**

- l'objet sollicité a été définitivement déplacé, une nouvelle URL est spécifiée dans la ligne d'en-tête (Location:)

## **400 Bad Request**

- demande de message n'a pas été comprise par le serveur

## **404 Not Found**

- document recherché est introuvable sur le serveur

## **505 HTTP Version Not Supported**

# Exemple de message de réponse

1. Telnet sur votre serveur Web favoris:

telnet www.eurecom.fr 80

Opens TCP connection to port 80  
(default HTTP server port) at www.eurecom.fr.  
Anything typed in sent  
to port 80 at www.eurecom.fr

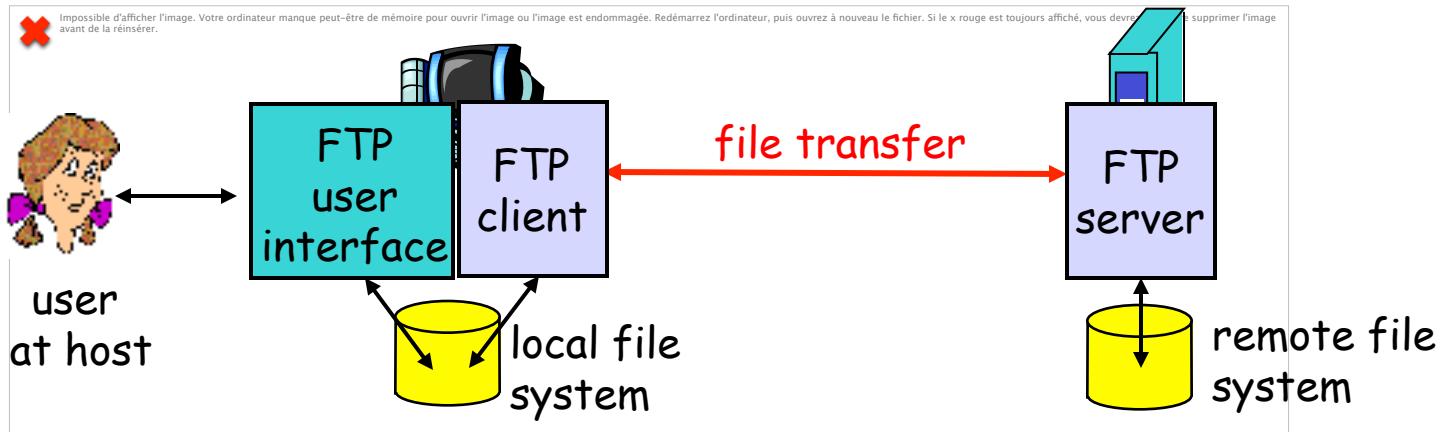
2. Taper la ligne suivante:

GET /~login/index.html HTTP/1.0

By typing this in (hit carriage  
return twice), you send  
this minimal (but complete)  
GET request to HTTP server

3. Observer la réponse du serveur!

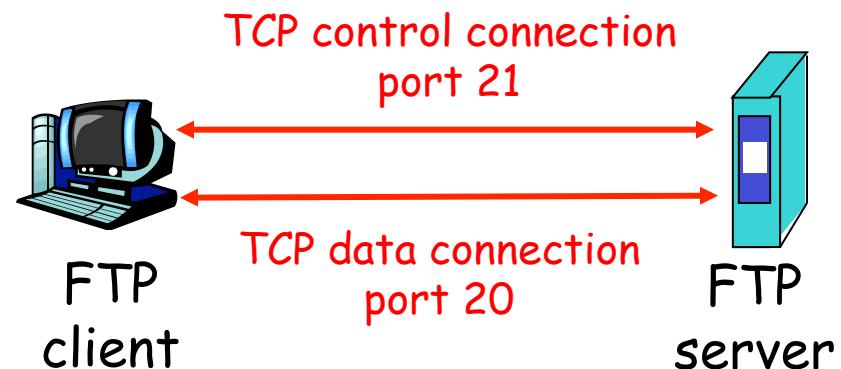
# FTP: the file transfer protocol



- Fichier de transfert de/à host distante
- modèle client/server
  - client*: partie qui initialise le transfert
  - server*: host distante
- ftp: RFC 959
- ftp serveur: port 21

# FTP: sépare les connexions de contrôle et de données

- FTP client contacte FTP server au port 21, spécifiant TCP comme protocole de transport
- Client obtient l' autorisation sur la connexion de contrôle
- Client navigue dans le répertoire distant en envoyant les commandes sur la connexion de contrôle
- Quand le serveur reçoit une commande pour un transfert de fichier, le serveur ouvre une connexion de données TCP au client
- Après le transfert d'un fichier, le serveur ferme la connection



- Serveur ouvre une seconde connection de données TCP pour transférer un autre fichier
- Connection de contrôle: "**hors bande**"
- FTP serveur garde trace des changements dans le répertoire tout le long de son exploration du répertoire distant

# FTP: commandes, réponses

## Simples commandes:

- Format as ASCII à 7 bits sur la connexion de contrôle
- USER** *username*
- PASS** *password*
- LIST** demande au serveur de transmettre une liste de tous les fichiers contenus dans le répertoire distant actuel
- RETR** *filename* = gets (obtenir un fichier à partir du répertoire distant)
- STOR** *filename* stocke (puts) le fichier dans le répertoire distant

## Simples codes de retour

- Code d'état et phrase (comme dans HTTP)
- 331 Username OK, password required**
- 125 data connection already open; transfer starting**
- 425 Can't open data connection**
- 452 Error writing file**

# Couche transport



# Plan

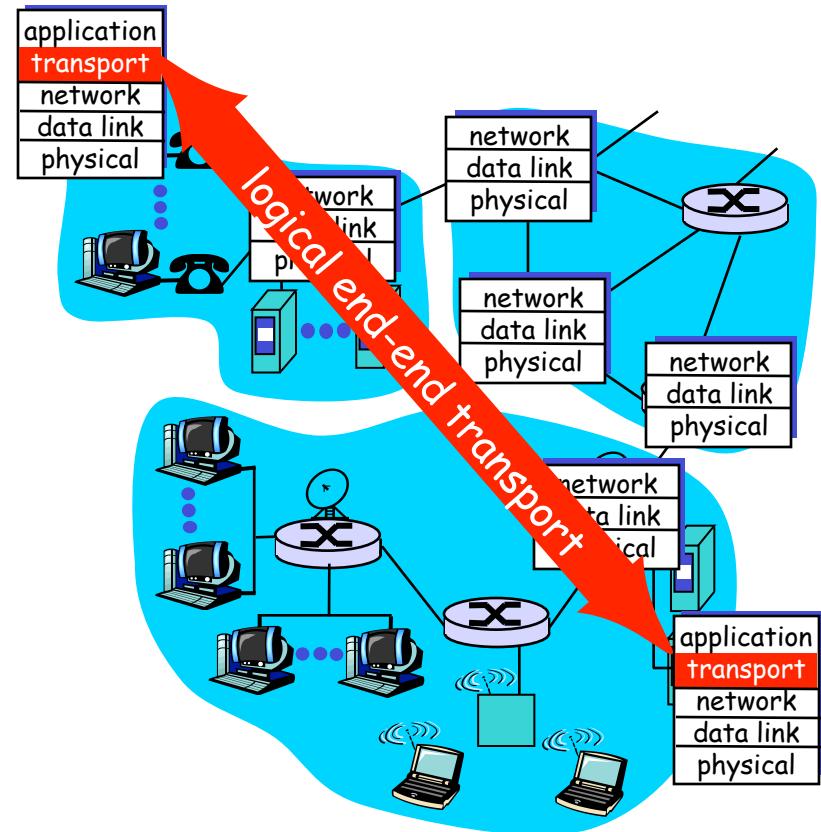
- Services de la couche Transport
- Multiplexage et démultiplexage
- Transport sans connexion: UDP
- Principes de transfert de données fiable
- Transport orienté connexion: TCP
  - structure de segment
  - transfert de données fiable
  - contrôle de flux
  - gestion de connections TCP

# Les couches de protocoles : TCP/IP et le modèle OSI

<u>Protocol Implementation</u>							<u>OSI</u>
File Transfer	Electronic Mail	Terminal Emulation	File Transfer	Client Server	Network Mgmt	Application	
File Transfer Protocol (FTP) RFC 559	Simple Mail Transfer Protocol (SMTP) RFC 821	TELNET Protocol RFC 854	Trivial File Transfer Protocol (TFTP) RFC 783	Network File System Protocol (NFS) RFC 1024, 1057 and 1094	Simple Network Management Protocol (SNMP) RFC 1157		Presentation
							Session
Transmission Control Protocol (TCP) RFC 793				User Datagram Protocol (UDP) RFC 768			Transport
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791	Internet Group Management Protocol (IGMP) RFC 2236		Internet Control Message Protocol (ICMP) RFC 792			Network
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS		Data Link
Transmission Mode TP STP FO Satellite Microwave, etc							Physical

# services de transport et protocoles

- fournit *une communication logique* sur des processus d'application exploités sur des serveurs différents
- les protocoles de transport s'exécutent aux extrémités
  - les messages sont découpés en *segments*, par l'émetteur
  - Le récepteur: reassemble ces segments en messages
- Plusieurs protocoles de transport disponibles pour les applications
  - Internet: TCP et UDP



# Relation entre couches Transport et réseau

- Couche réseau:*  
communication logique entre machines
- Couche transport:*  
communication logique entre processus

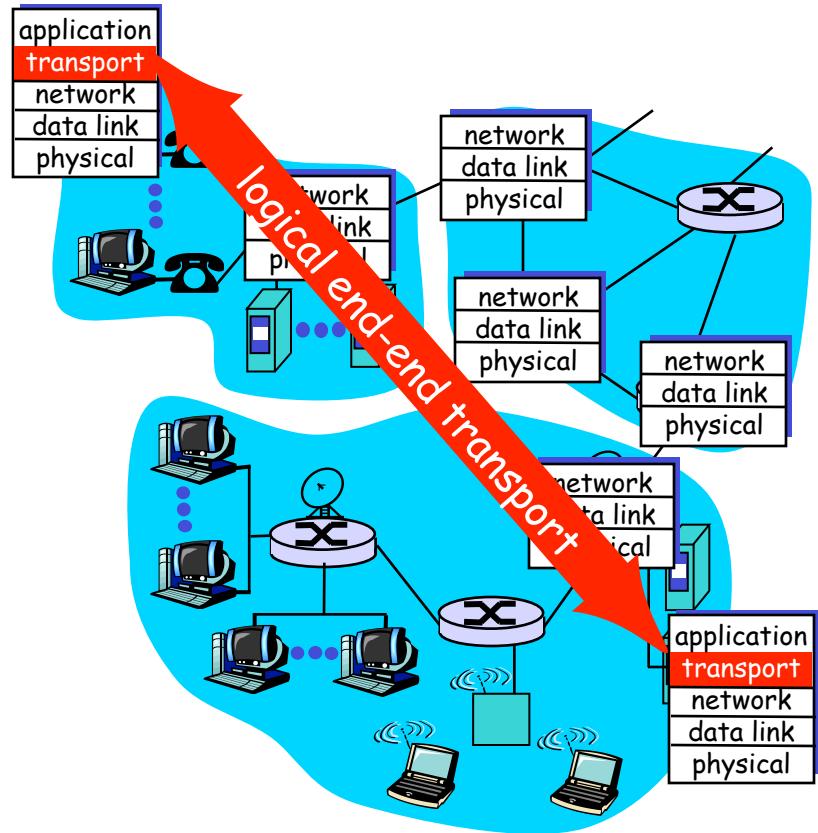
## analogie:

*12 enfants envoient des lettres à 12 autres enfants*

- processes = enfants
- app messages = lettres dans les enveloppes
- hosts = maisons
- transport protocol = Anne et Bill
- network-layer protocol = service postal

# Les protocoles de la couche transport

- Délivrance fiable, dans l'ordre (TCP)
  - Contrôle de congestion
  - Contrôle de flux
  - Initialisation de la connection
- Délivrance Non fiable, en désordre (UDP)
- services non disponibles:
  - délai garanti
  - bandwidth garantie



# Multiplexage/Démultiplexage

Démultiplexage à rcv host:

Orientation des segments  
reçus vers la bonne interface  
“socket”

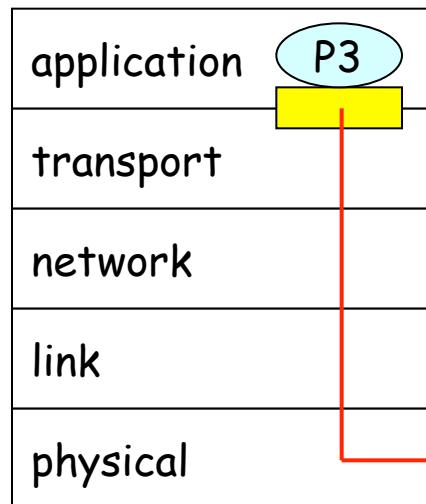
= socket



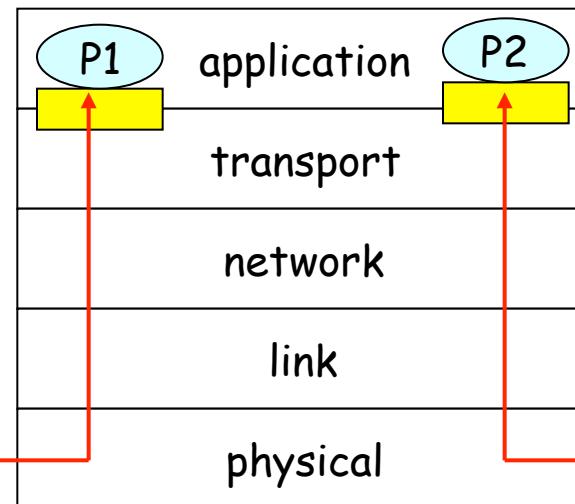
= process

Multiplexage à send host:

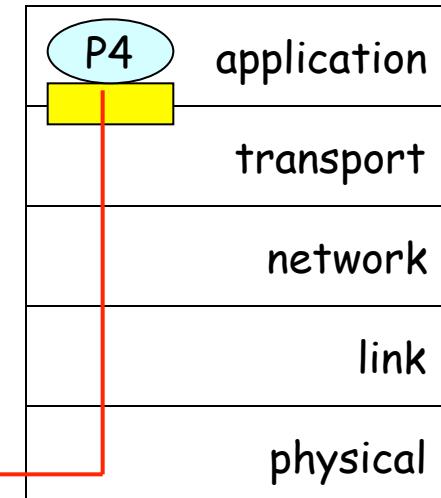
Rassemblement des données  
Création des segments en  
les associant des en-têtes



host 1



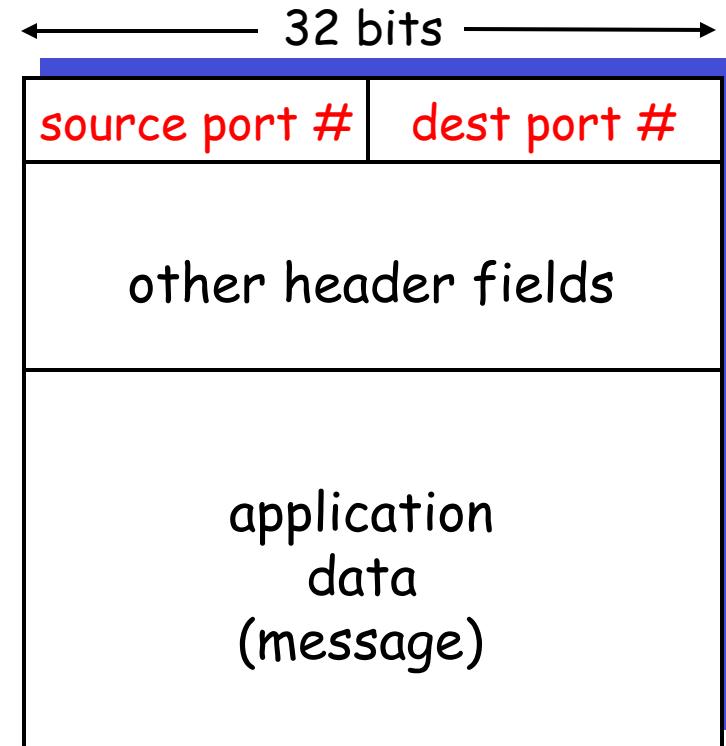
host 2



host 3

# Comment le démultiplexage fonctionne?

- host reçoit IP datagrams
  - chaque datagram a une source IP adresse, une destination IP adresse
  - chaque datagram contient un segment transport
  - chaque segment a des numéros de ports source, destination
  - La machine utilise l'adresse IP et le numéro de port pour diriger le segment au socket appropriée



TCP/UDP segment format

# Démultiplexage sans connexion

- Creation des sockets avec les numéros de port:

```
DatagramSocket mySocket1 = new  
DatagramSocket(09111);
```

```
DatagramSocket mySocket2 = new  
DatagramSocket(09222);
```

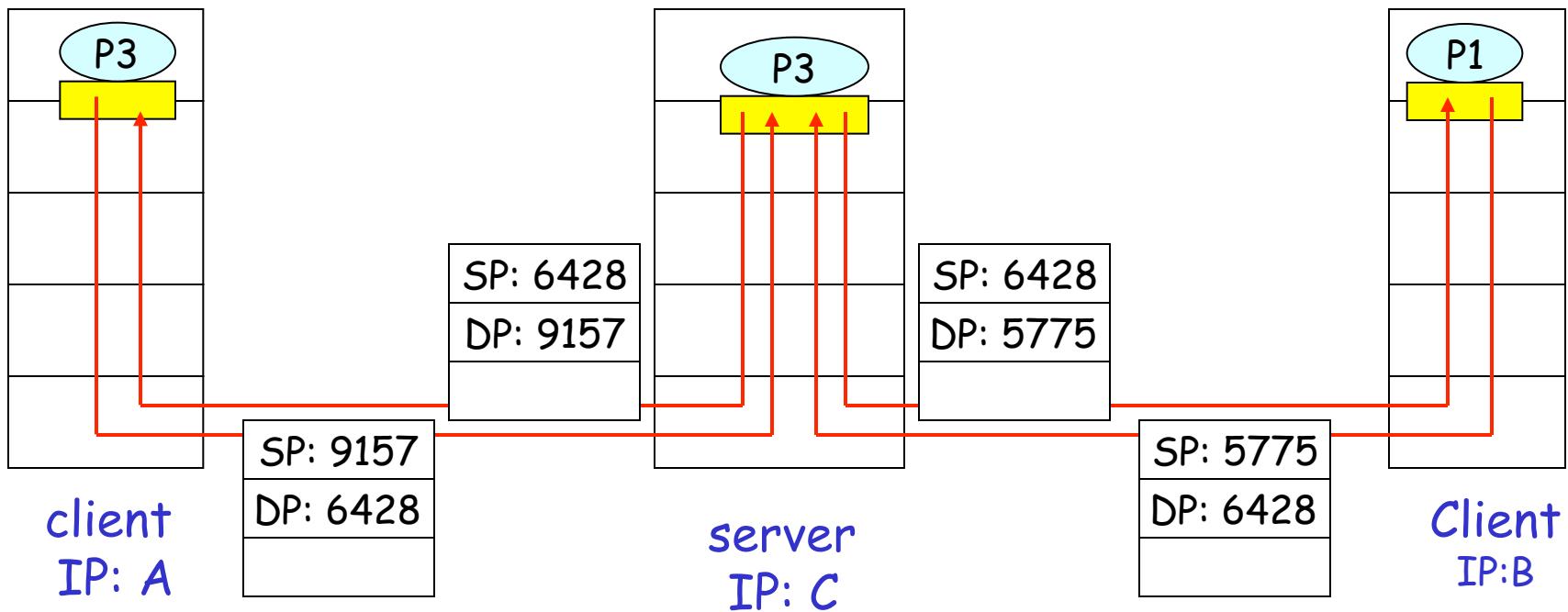
- UDP socket identifiée par le tuple :

(dest IP address, dest port number)

- Quand host reçoit UDP segment:
  - contrôle le numéro de port destination dans le segment
  - dirige UDP segment vers la socket avec un numéro de port

# Démultiplexage sans connection

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```

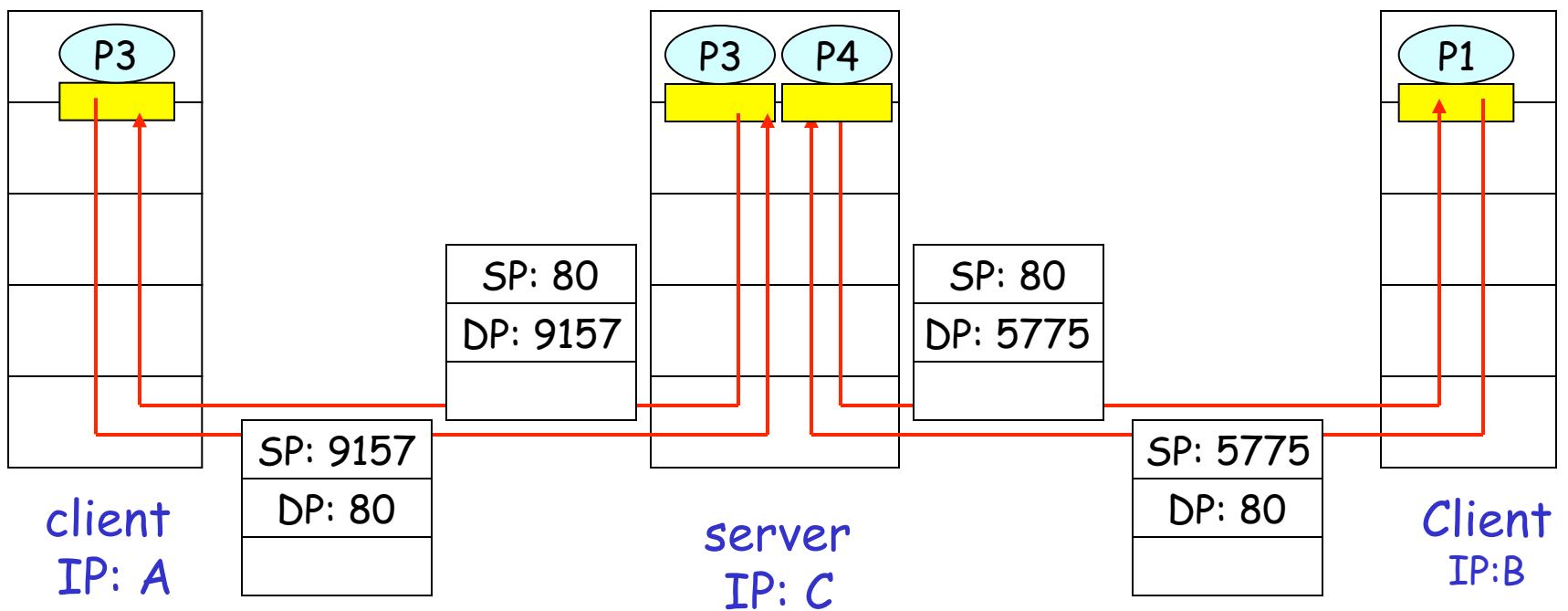


SP provides “return address”

# Démultiplexage avec Connection

- TCP socket identifiée par un tuple suivant:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- recv host utilise les 4 valeurs pour diriger le segment vers la socket appropriée
- Server host peut supporter plusieurs TCP sockets simultanées:
  - chaque socket identifiée par un tuple de 4 valeurs
- Web servers ont des différents sockets pour chaque connection client
  - HTTP ouvre différentes socket pour chaque demande

# Démultiplexage avec Connection



# UDP: User Datagram Protocol [RFC 768]

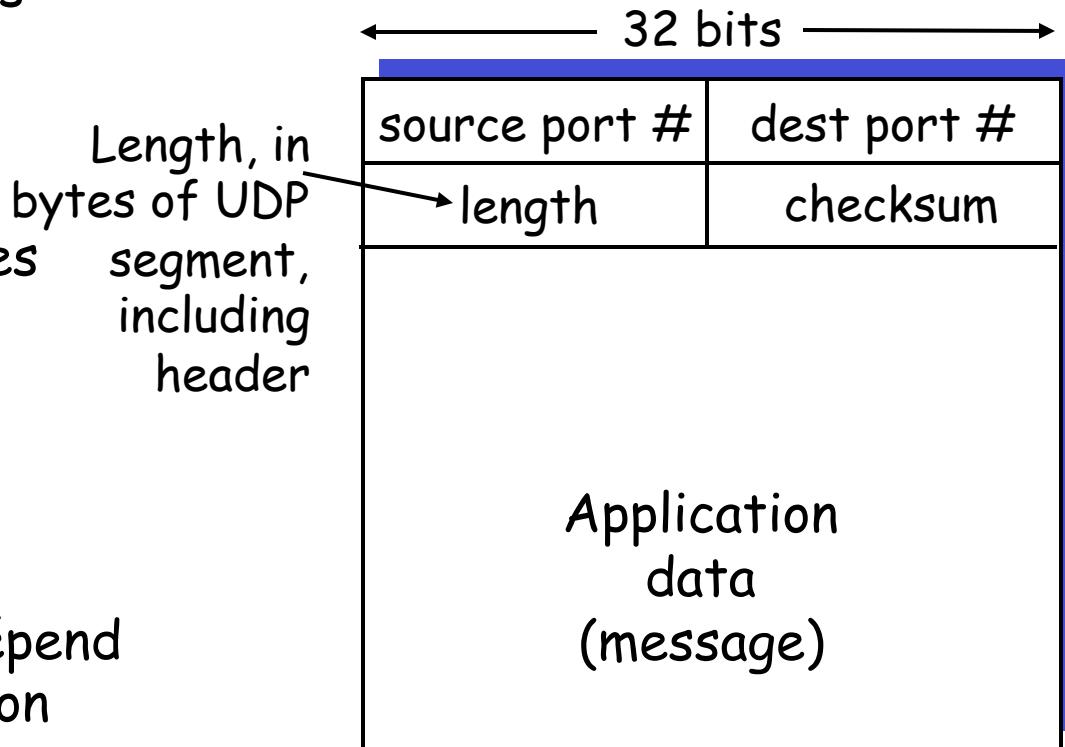
- UDP est un protocole “best effort”, UDP segments peuvent être:
  - perdus
  - Délivrés en désordre à la couche application
- *Sans connection:*
  - pas d'accord en avance entre UDP sender, receiver
  - chaque UDP segment est manipulé indépendamment des autres

## Pourquoi UDP?

- pas d'établissement de connection (qui peut ajouter un délai supplémentaire)
- simple: pas d'état de connection à l'émetteur, récepteur
- un en-tête plus court de segment (8 octets)
- pas de contrôle de congestion

# UDP: structure de segment

- souvent utilisé pour des applications comme streaming multimédia (audio)
  - tolérance aux pertes
  - sensible au débit
- autres applications
  - DNS
  - SNMP
- le transfer sur UDP dépend de la couches application



UDP segment format

# UDP: checksum

but: permet la détection d'erreur sur le segment par le récepteur

## Sender:

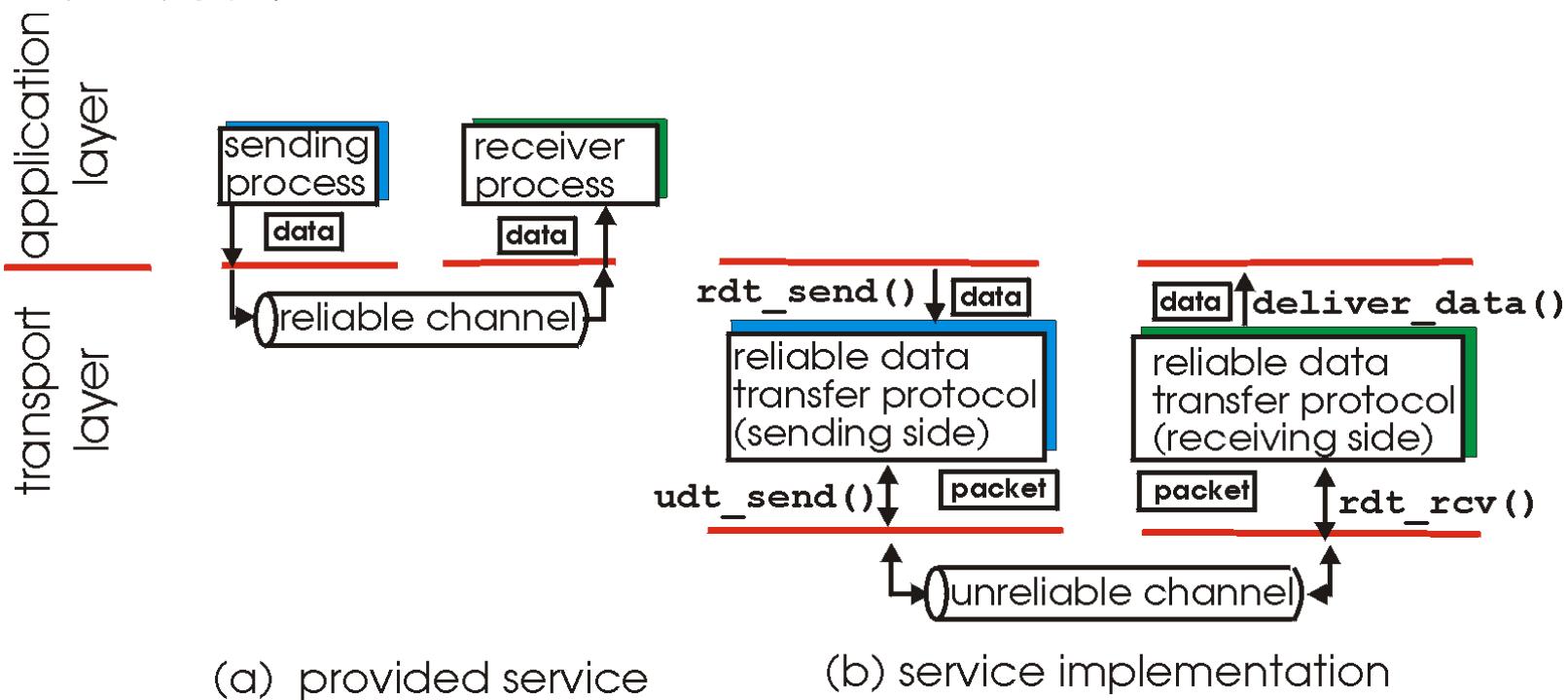
- fait la somme de tous les mots de 16 bits du segment en éliminant tout dépassement de capacité
- checksum: puis on fera le complément à 1 du résultat de la somme
- sender met cette valeur dans le champ checksum du segment UDP

## Receiver:

- fait la somme de tous les mots de 16 bits y compris le checksum
- si cette somme = "1111111111111111"
  - Non → error detected
  - Oui → no error detected.

# Principes du transfert de données fiable

- important dans les couches application, transport et liaison

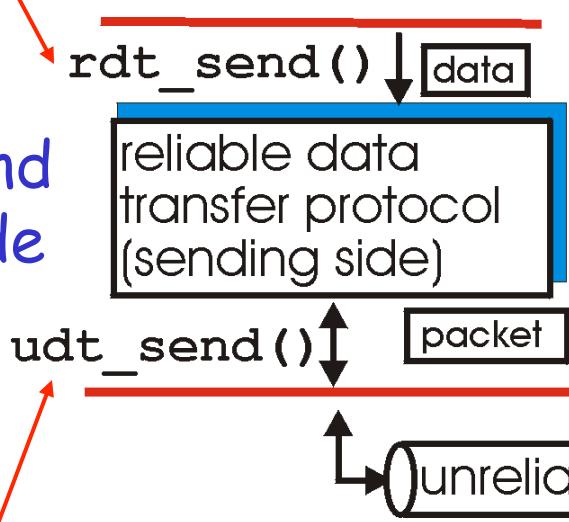


- Caractéristiques de canal non fiable détermineront la complexité du transfert de données fiable “ reliable data transfer protocol (rdt)/unreliablerdt (non fiable)”

# Reliable data transfer: initial

**rdt\_send()** : sollicité pour remettre les données à expédier à la couche supérieure du destinataire

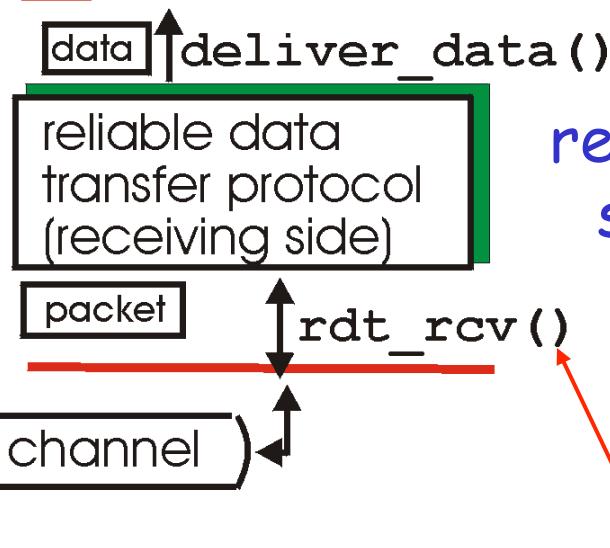
send side



**udt\_send()** : appelé par rdt, pour transférer le paquet sur un canal non fiable au destinataire

**deliver\_data()** : appelé par rdt pour délivrer les données à la couche supérieure

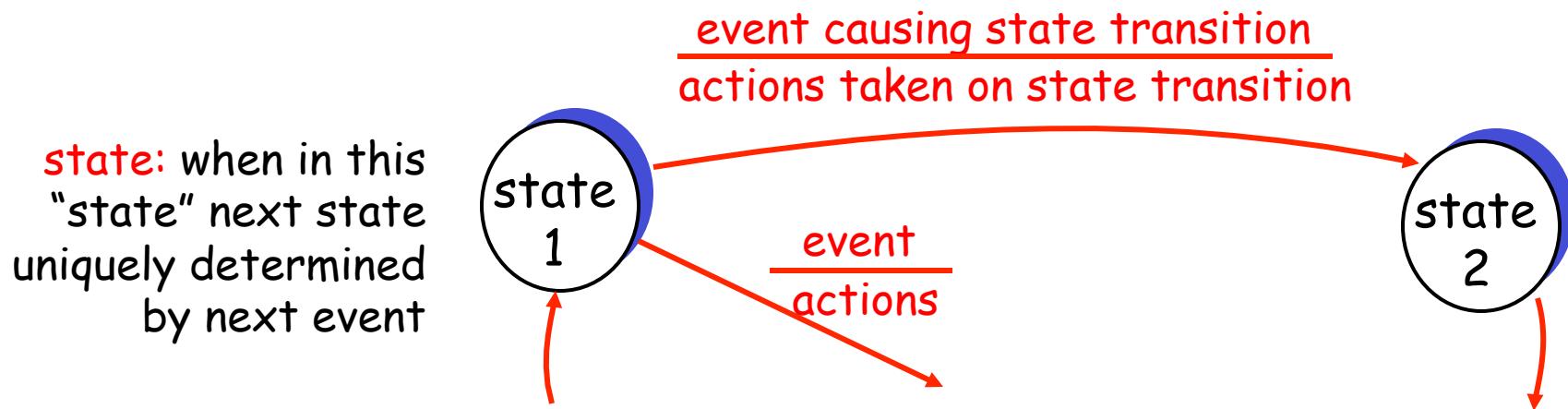
receive side



**rdt\_rcv()** : appelé quand le paquet arrive sur le canal de destination

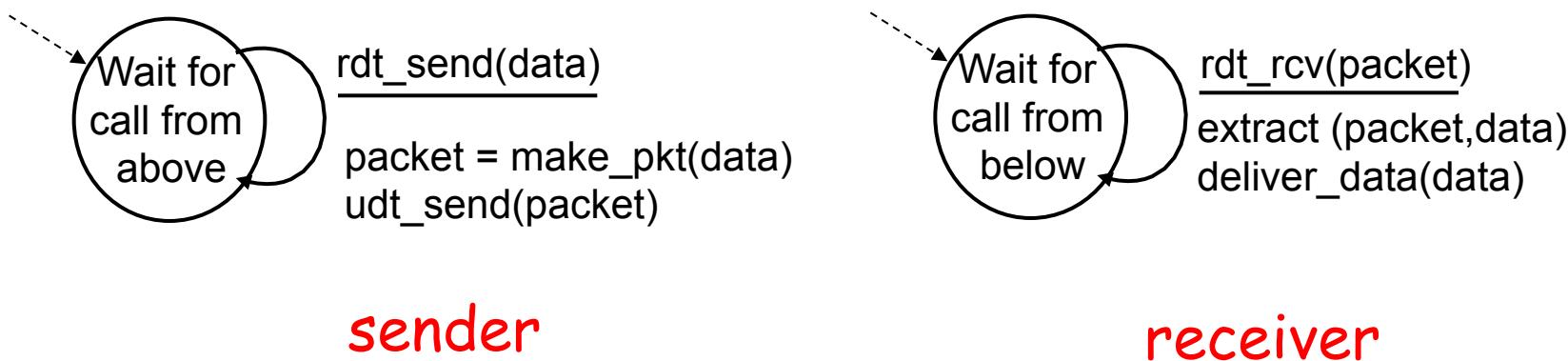
# Reliable data transfer: initial

- développement de façon indépendantes les rdt de l'émetteur et de récepteur
- on considère seulement unidirectionnel data transfer
  - mais le contrôle de flux est bidirectionnel
- utilise finite state machines (FSM) "automate à nombre d'états fini" pour spécifier sender, receiver



# Transfert fiable sur un canal fiable

- au dessous un canal parfaitement fiable
  - pas d' erreurs bit
  - pas de pertes de paquets
- séparation des FSMs de sender et receiver:
  - sender envoie des data sur le canal de dessous
  - receiver lit les data de canal de dessous



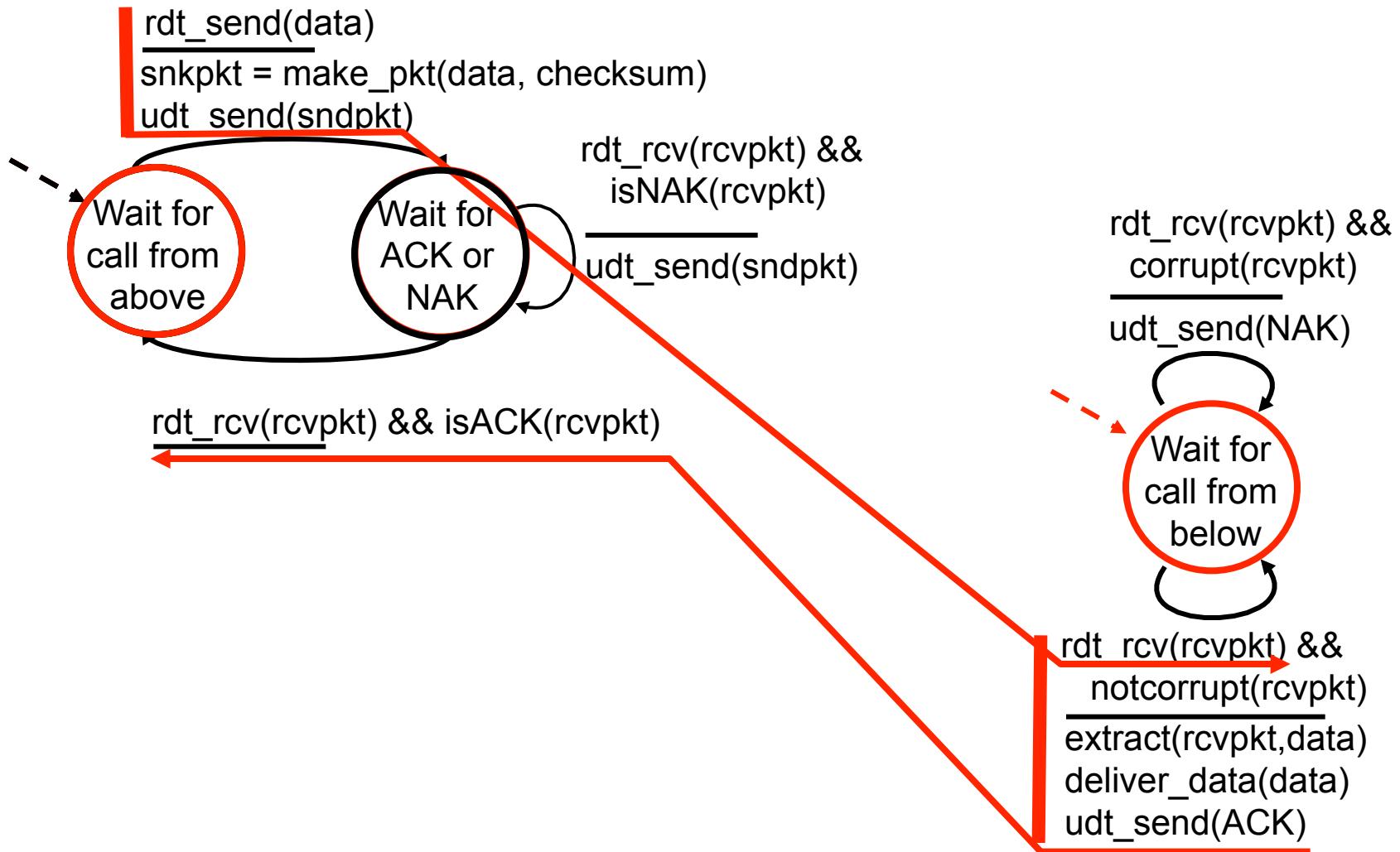
sender

receiver

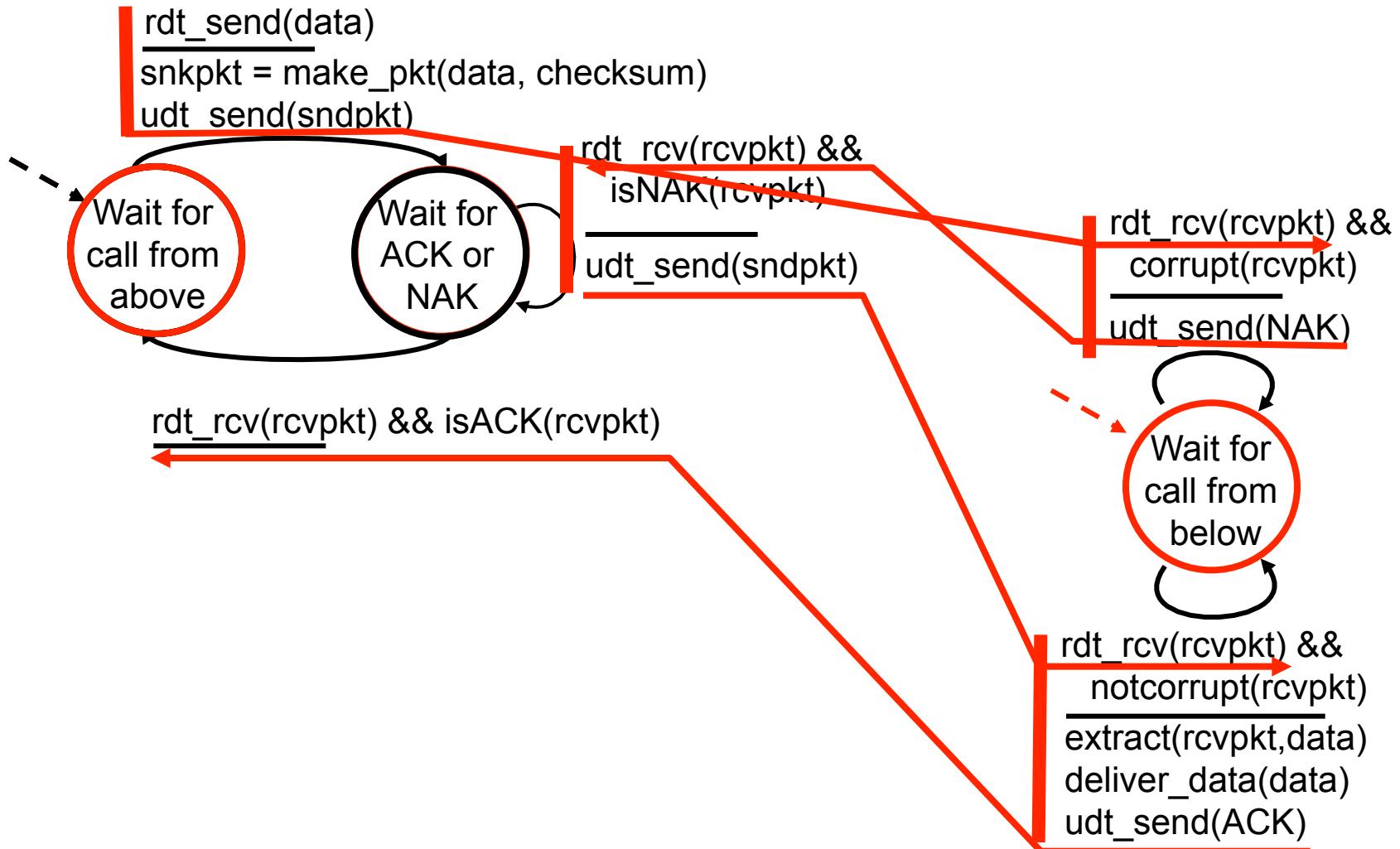
# Transfert sur un canal avec erreurs bit

- au dessous d'un canal qui laisse introduire des erreurs
- la question: comment détecter les erreurs:
  - **acknowledgements (ACKs)**: le destinataire notifie explicitement l'émetteur que le paquet est reçu
  - **negative acknowledgements (NAKs)**: récepteur envoie une notification explicitement à l'émetteur qu'il y a une erreur dans le paquet
  - sender retransmet les paquets sur réception d'un NAK

# Opération sans erreurs



# Scenario d'erreur



# Erreur fatale !

Qu'est ce que se passe si ACK/NAK erronés?

- l'émetteur ne sait pas qu'est ce que se passe chez le destinataire!
- ne peut pas juste retransmettre: duplication possible

Qu'est ce qu'il fait?

- le sender si ACK/NAK se perdent?

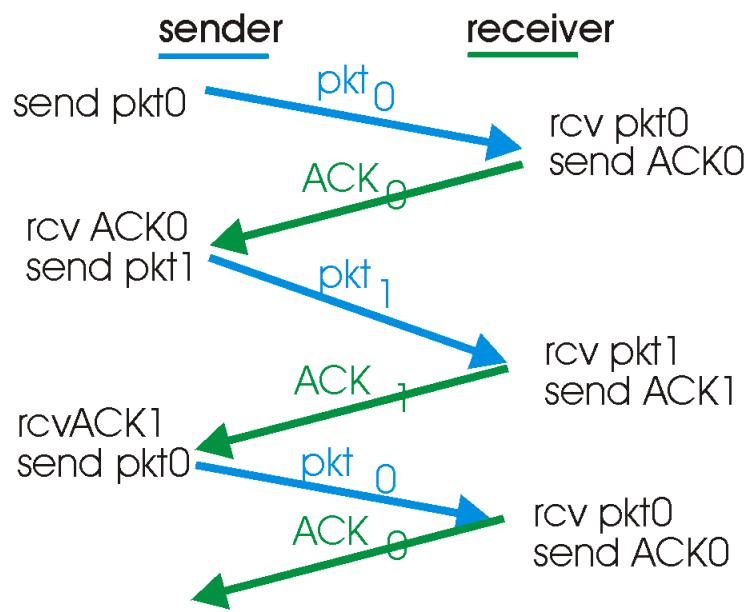
Eviter la duplication:

- sender ajoute *sequence number* pour chaque paquet
- sender retransmet le paquet courant si ACK/NAK corrompus
- le destinataire écarte le paquet dupliqué

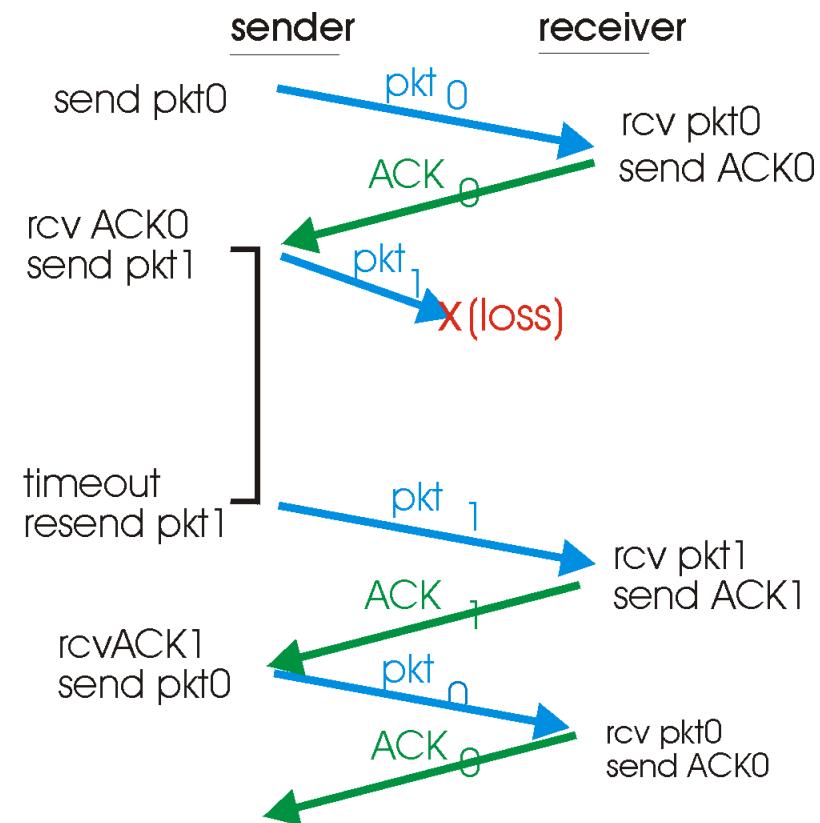
stop and wait

Sender envoie un paquet alors attends la réponse de récepteur

# Action: échanges des paquets

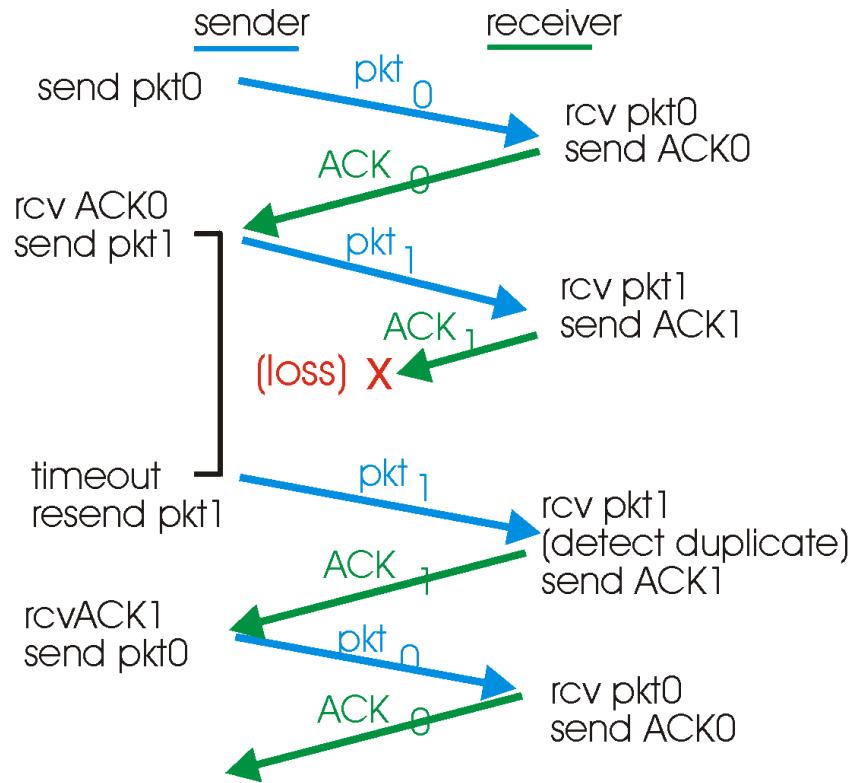


(a) operation with no loss

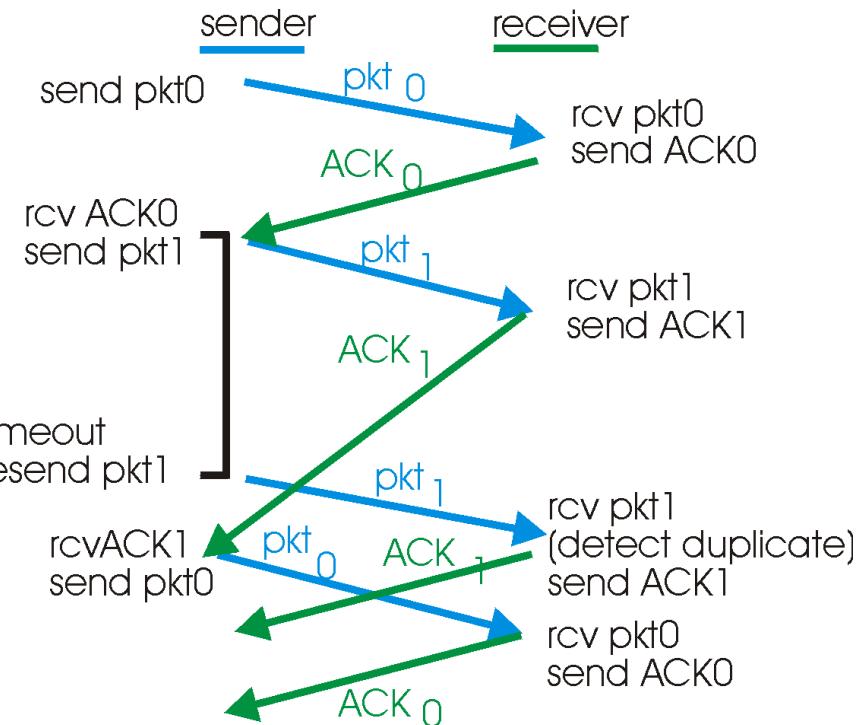


(b) lost packet

# Action: perte d'un ACK



(c) lost ACK

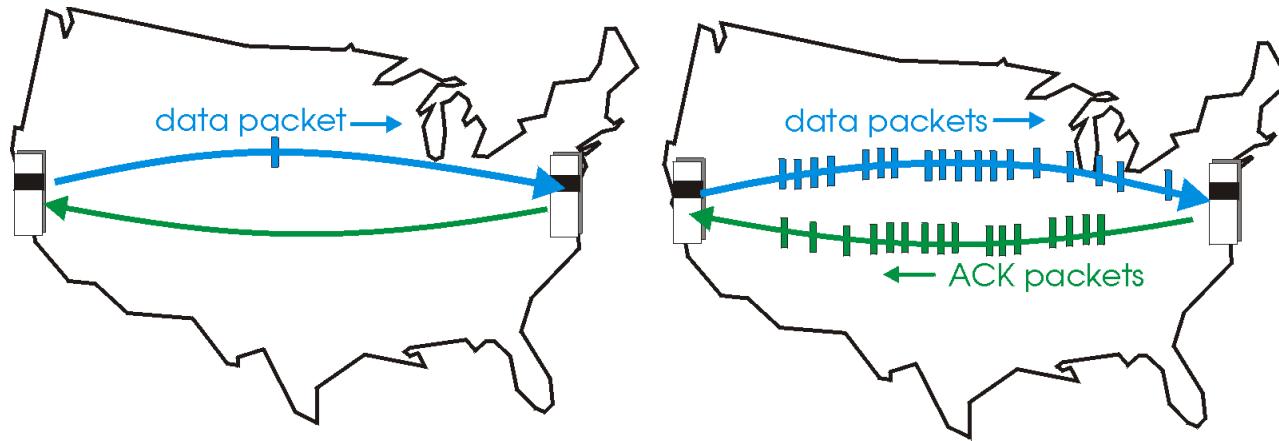


(d) premature timeout

# Protocole Pipeliné

Pipelining: envoie de plusieurs paquets avant de se mettre en attente

- L'intervalle de numéros de séquence doit être augmenté
- buffering à l'émetteur et/ou récepteur

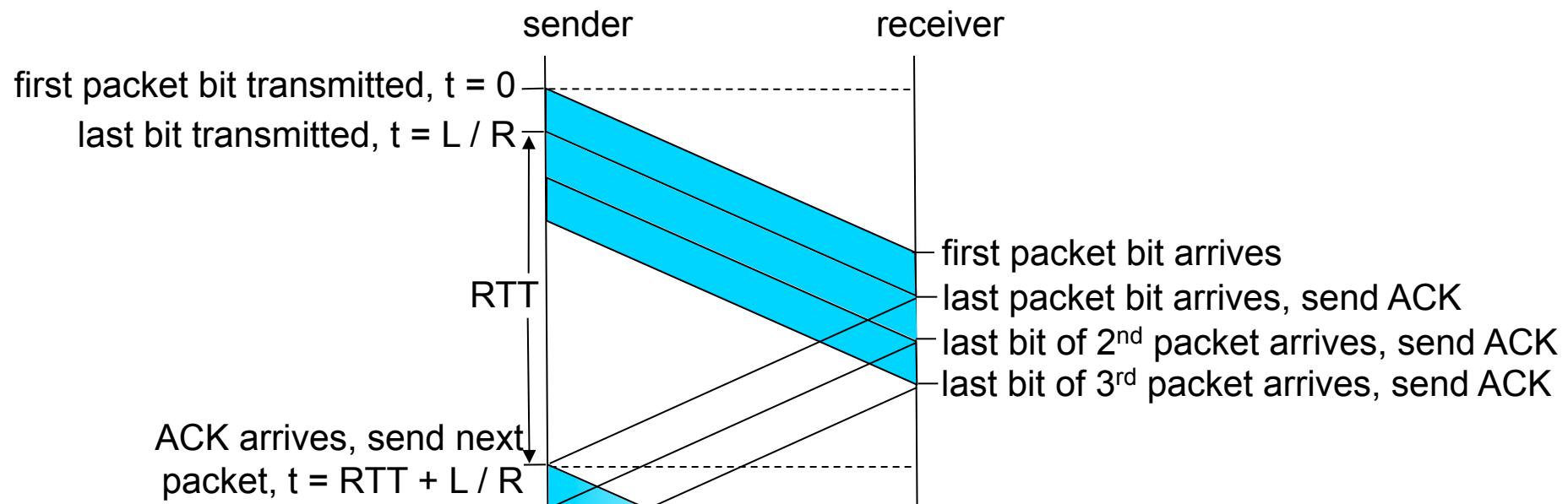


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Deux méthodes de correction d'erreurs en mode pipeline:  
*go-Back-N (aller à la trame N), répétition sélective*

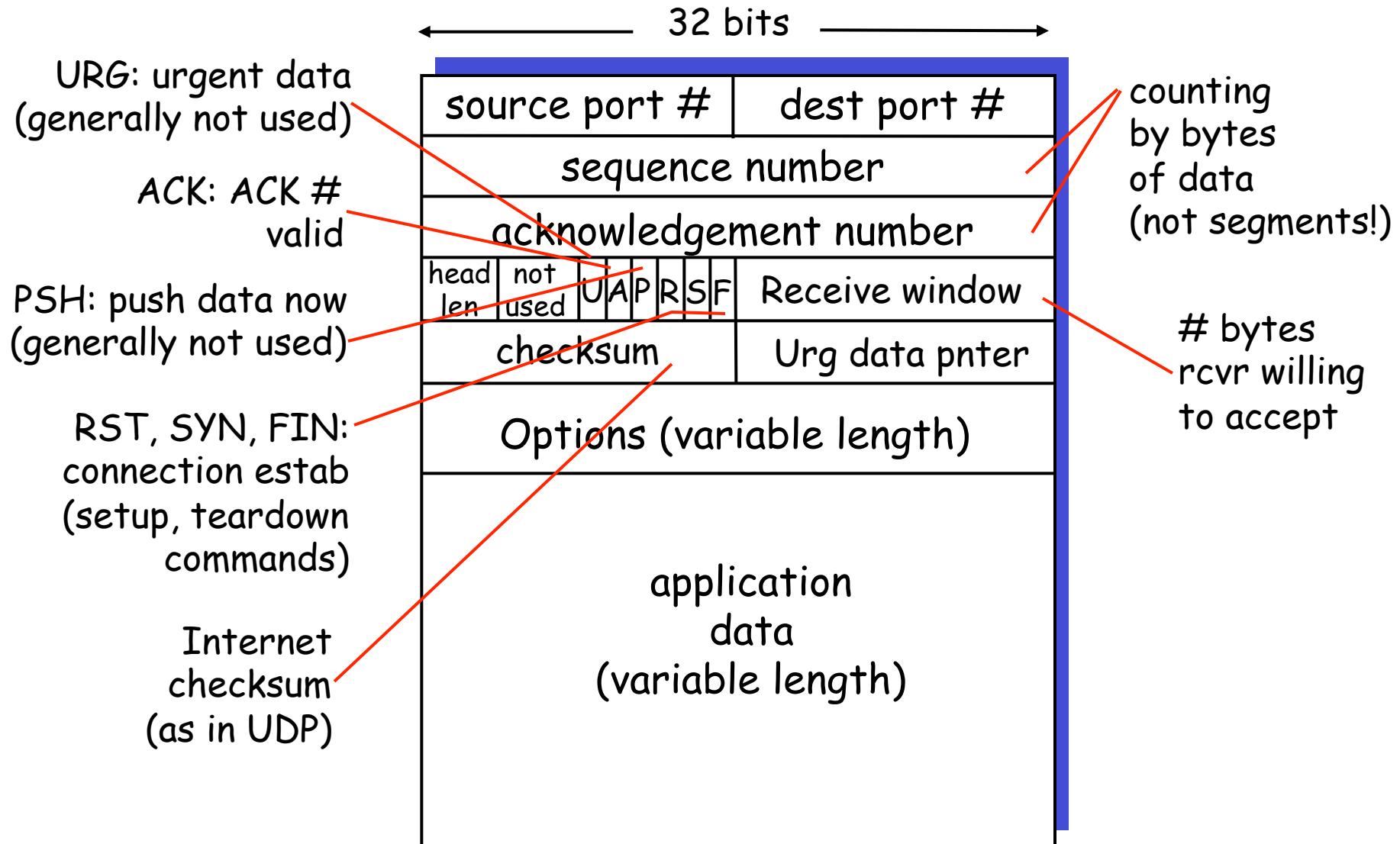
# Pipelining: augmente l'utilisation



Increase utilization  
by a factor of 3!

$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

# TCP segment structure



# TCP seq. number et ACKs

## Numéro de Séquence:

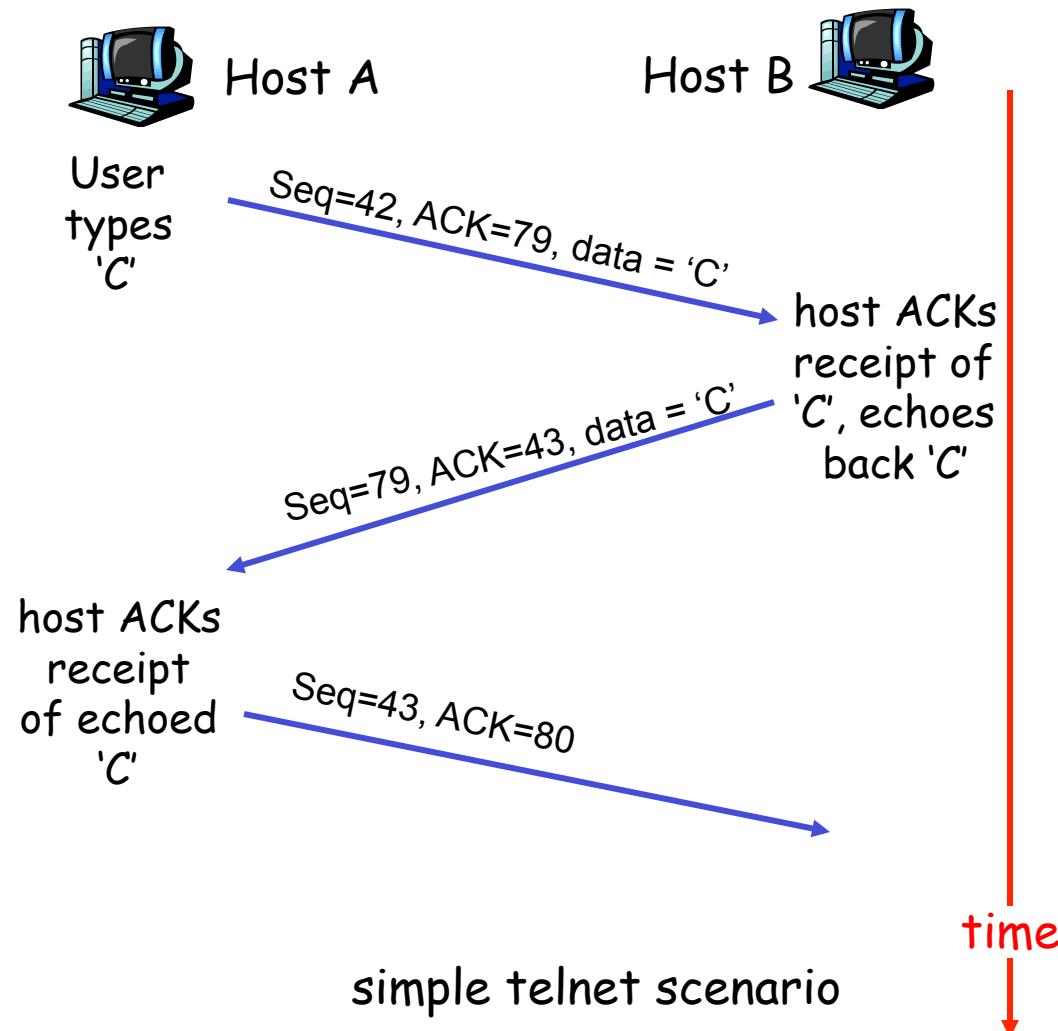
- Le numéro dans le flux d'octets du premier octet de chaque segment

## ACKs:

- seq. number du prochain octet expédié de l'autre côté
- en cas de perte d'un segment un cumulative ACK cumulatif est envoyé

Q: comment TCP résolve le problème de perte de segment hors séquence

- Le choix auprogrammeur de le détruire ou de le conserver



# TCP: Round Trip Time et Timeout

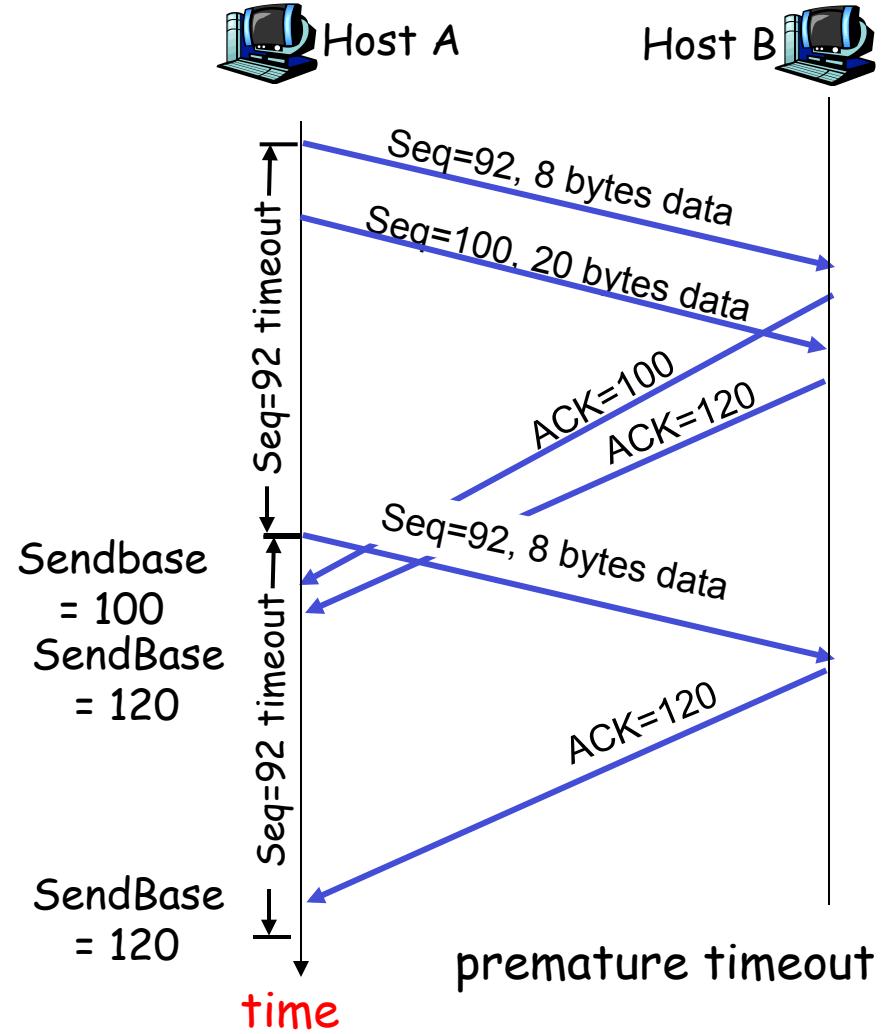
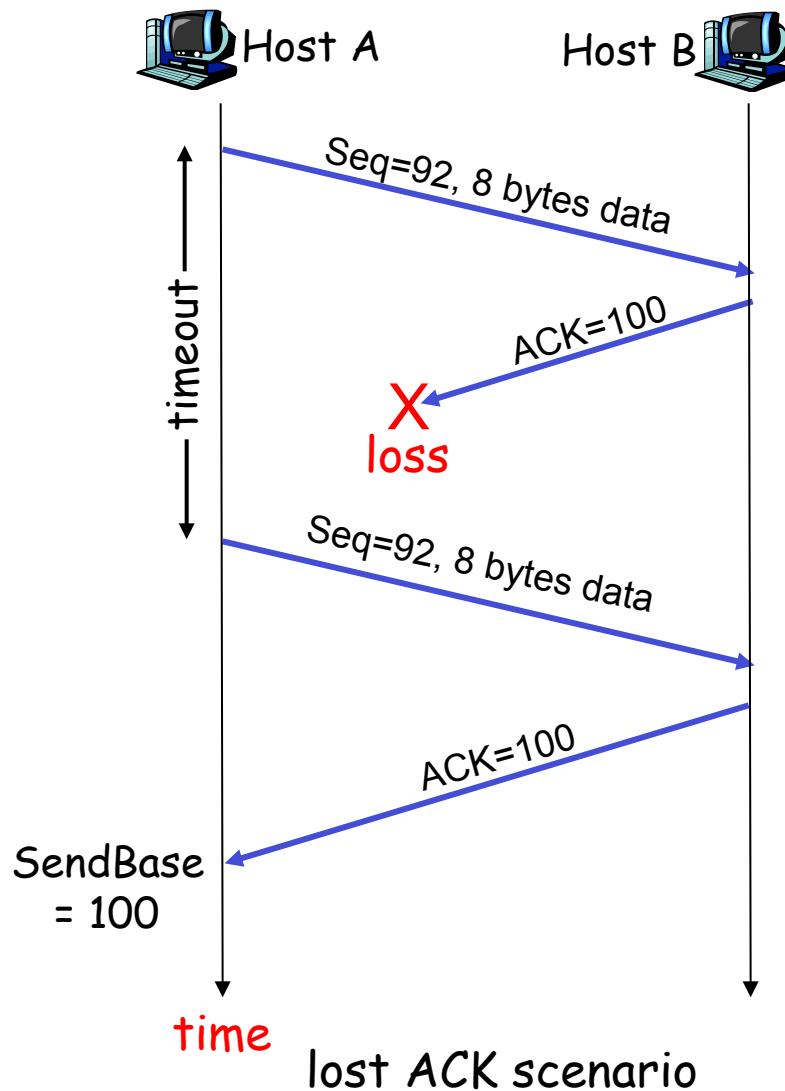
- Q:** comment fixer la valeur timeout?
- plus supérieur que le RTT
    - mais le RTT varie
  - court: timeout prématué
    - Retransmissions non nécessaire
  - long: TCP tardera à retransmettre le segments

- Q:** comment estimer le RTT?
- SampleRTT:** temps écoulé entre son envoi (c'est à dire son passage à l'IP) et l'ACK renvoyé par le destinataire
    - Ignore les segments retransmis
  - SampleRTT varié d'un segment à un autre
    - Une mesure moyenne s'avère nécessaire appelée EstimatedRTT

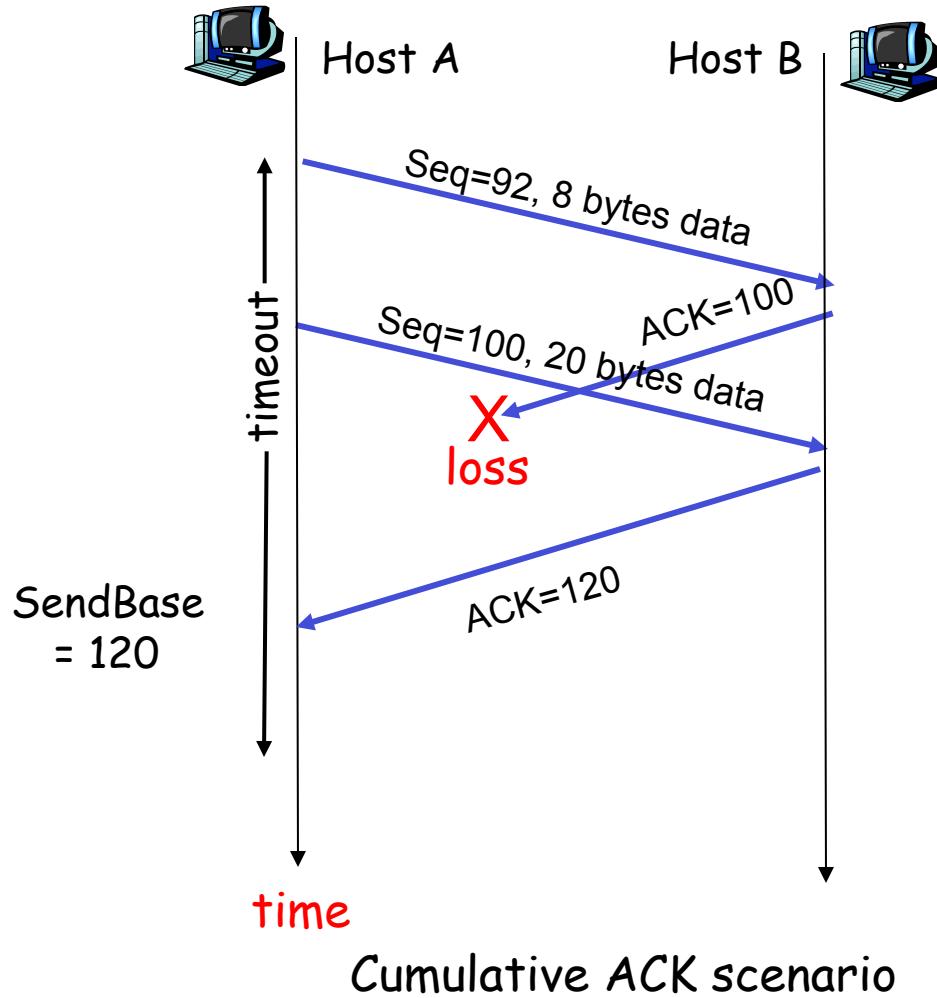
$$\text{EstimatedRTT (new)} = (1 - \alpha) * \text{EstimatedRTT (old)} + \alpha * \text{SampleRTT (new)}$$

Avec  $\alpha = 0,125$

# TCP: retransmission scenarios

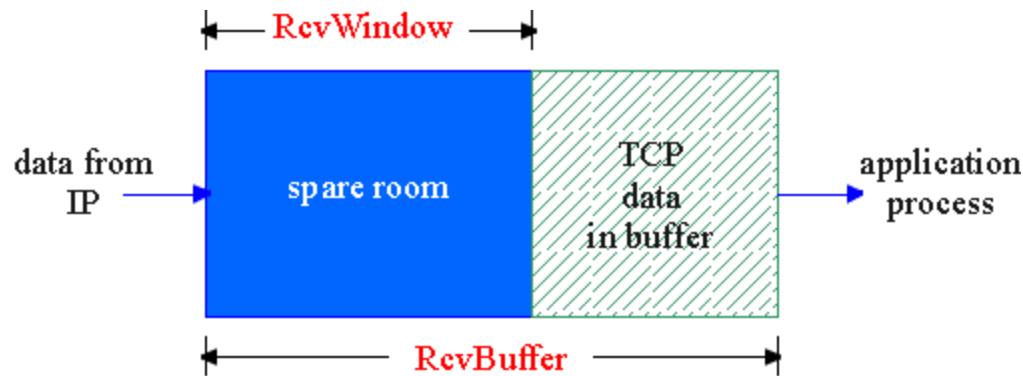


# TCP retransmission scenarios



# TCP: Flow Control

- tampon (buffer) de réception:

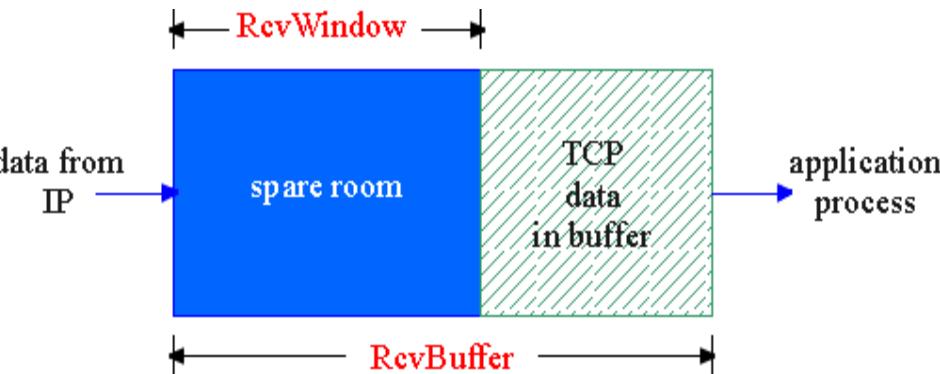


## flow control

L'objectif est d'équilibrer le rythme d'envoi au sender à la vitesse de lecture de destinataire

- le processus d'application peut être lent pour lire dans le buffer

# Flow control: fonctionnement



(Suppose que le receiver supprime tous les segments hors séquence)

- Espace mémoire disponible( dans le buffer) = **RcvWindow**

$$= \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

- Rcv informe le sender de l'espace disponible dans son tampon en insérant la valeur **RcvWindow** dans les segments
- Sender maintient les données non confirmées au-dessous de **RcvWindow**
  - Données non confirmées= **LastByteSent** - **LastByteAcked**
  - garantir le non débordement de buffer de réception

# TCP: Gestion de Connection

Rappel: sender, receiver établissent une "connection" avant d'échanger les segments de données

- Initialisent les variables:
  - seq. number
  - buffers, flow control info (RcvWindow)
- *client*: informe le récepteur par

```
Socket clientSocket = new  
Socket("hostname", "port  
number");
```

- *serveur*: contacté par le client

```
Socket connectionSocket =  
welcomeSocket.accept();
```

L'établissement se fait en 3 étapes:

- Step 1: client envoie un SYN segment au serveur
  - spécifie initial seq. number
  - pas de données
- Step 2: serveur reçoit SYN, répond par un SYNACK segment
  - serveur alloue les buffers
  - spécifie son initial seq. number

Step 3: client reçoit SYNACK, répond avec ACK segment, qui peut contenir des données

# TCP: Gestion de Connection

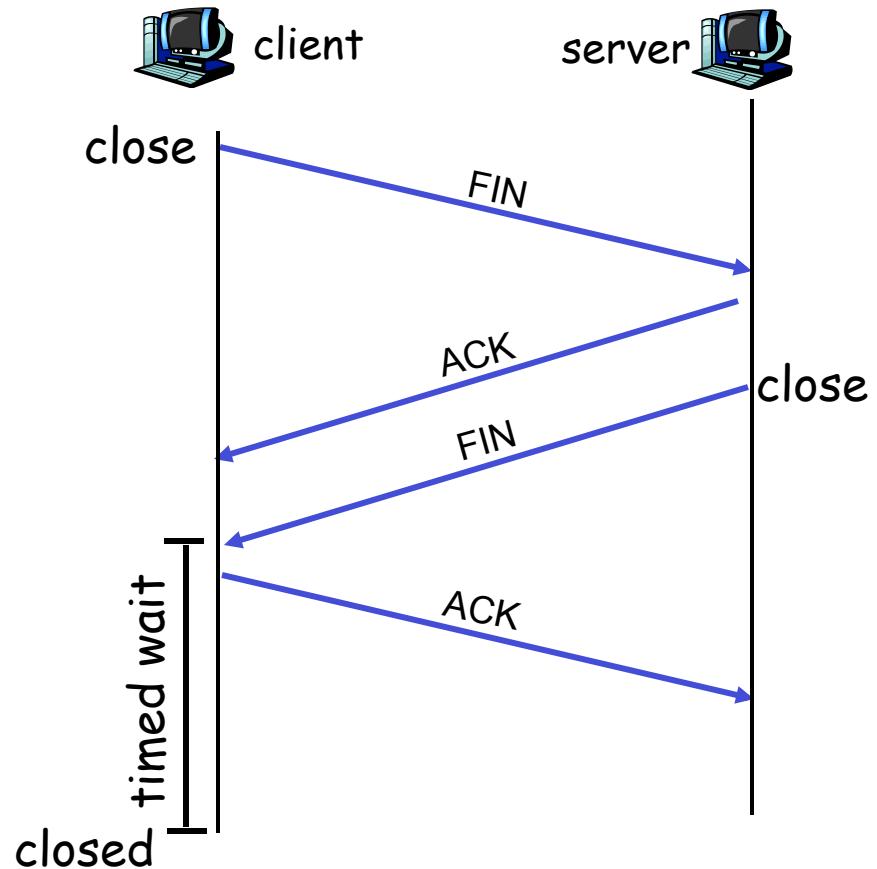
## Fermeture de connection:

client ferme la socket:

```
clientSocket.close();
```

Step 1: client envoie un segment FIN au serveur

Step 2: server reçoit le segment FIN, répond avec ACK. Ferme connection, envoie FIN.

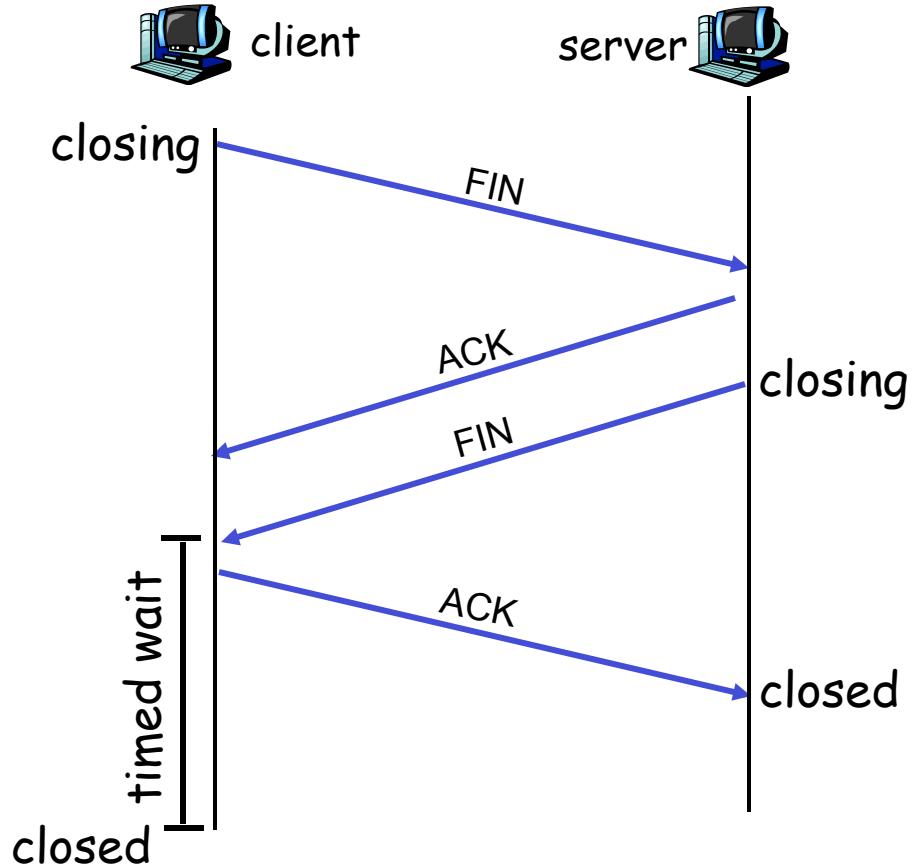


# TCP: Gestion de Connection

Step 3: client reçoit FIN,  
répond avec ACK.

- Déclenche un "timed wait" = 30 secondes qui lui permet de renvoyer l'ACK en cas de perte

Step 4: server, reçoit un ACK. La Connection est fermée.



# Couche Réseau



# plan

- Introduction
- IP: Adressage
- IP: Routage
- ARP Protocole
- ICMP Protocole

# Introduction

<u>Protocol Implementation</u>							<u>OSI</u>
File Transfer	Electronic Mail	Terminal Emulation	File Transfer	Client Server	Network Mgmt	Application	
File Transfer Protocol (FTP) RFC 559	Simple Mail Transfer Protocol (SMTP) RFC 821	TELNET Protocol RFC 854	Trivial File Transfer Protocol (TFTP) RFC 783	Network File System Protocol (NFS) RFC 1024, 1057 and 1094	Simple Network Management Protocol (SNMP) RFC 1157	Presentation	
						Session	
Transmission Control Protocol (TCP) RFC 793			User Datagram Protocol (UDP) RFC 768			Transport	
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791	Internet Group Management Protocol (IGMP) RFC 2236		Internet Control Message Protocol (ICMP) RFC 792		Network	
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS	Data Link	
Transmission Mode						Physical	
TP	STP	FO	Satellite	Microwave, etc			

# IP ?

- IP est un service simple pour l'envoi de datagrammes en mode non connecté.
- IP est sensible à l'adressage, il s'assure que le routeur sait ce qu'il doit faire, lorsque les données arrivent.

# Fonctions IP

- Acheminement de datagrammes vers un destinataire en mode non connecté (Datagramme)
  - ↳ @IP (Adresse IP)
- Routage : déterminer le chemin
  - ↳ Pour les paquets de la couche transport
  - ↳ Pour les trames de la couche liaison (Routeur)
  - ↳ Aucune connaissance complète de la carte du réseau
- Fragmentation/Réassemblage
  - ↳ Des micros aux gros systèmes
- Gestion des options IP
- Envoi et réception des messages de contrôle et d'erreur par ICMP

# Ce que ne fait pas IP !

IP n'est pas fiable : il ne fait pas

- ↳ Multiplexage
- ↳ Séquencement
- ↳ Détection des duplications
- ↳ Détection de perte et retransmission
- ↳ Contrôle de flux

→ IP est un protocole 'Best effort'

# Internet datagramme

- Unité de transfert basique
- Format de datagramme



0	4	8	16	19	24	31
Vers	Hlen	Type of serv.		Total length		
		Identification	Flags	Fragment offset		
TTL		Protocol		Header Checksum		
		Source IP address				
		Destination IP address				
		IP Options (if any)		Padding		
		Data				
		...				

# IP datagramme

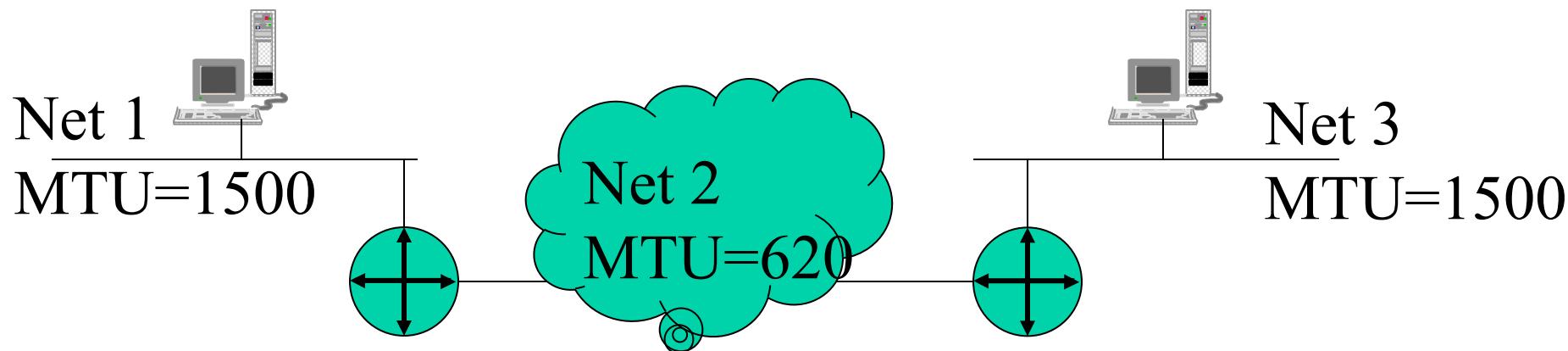
- **Vers** (4 bits): (IPv4=4)
- **Hlen** (4 bits): Header length (mot de 32 bits, sans options (cas général) = 20 octets)
- **Type of Service - TOS** (8 bits): n'est pas utilisé,
- **Total length** (16 bits): length de datagramme en octets en-tête inclus
- **identification, flags, fragmentation**
- **Time to live - TTL** (8bits): spécifie la durée de vie de datagramme
  - Routers decrement by 1
  - When TTL = 0 router discards datagram
  - Prevents infinite loops
- **Protocol** (8 bits): spécifie le format de la zone de données
  - Protocol numbers administered by central authority to guarantee agreement, e.g. TCP=6, UDP=17 ...

# IP Datagramme

- **Checksum**
- **Source & destination IP address (32 bits ):**  
contiennent les adresses IP source et destination
- **Options (variable):** infos. de routage et sécurité

# IP Fragmentation

- Comment nous pouvons envoyer 1400 bytes à travers un réseau de *Maximum Transfer Unit (MTU)* est de 620 bytes?
- La réponse: fragmenter le datagramme

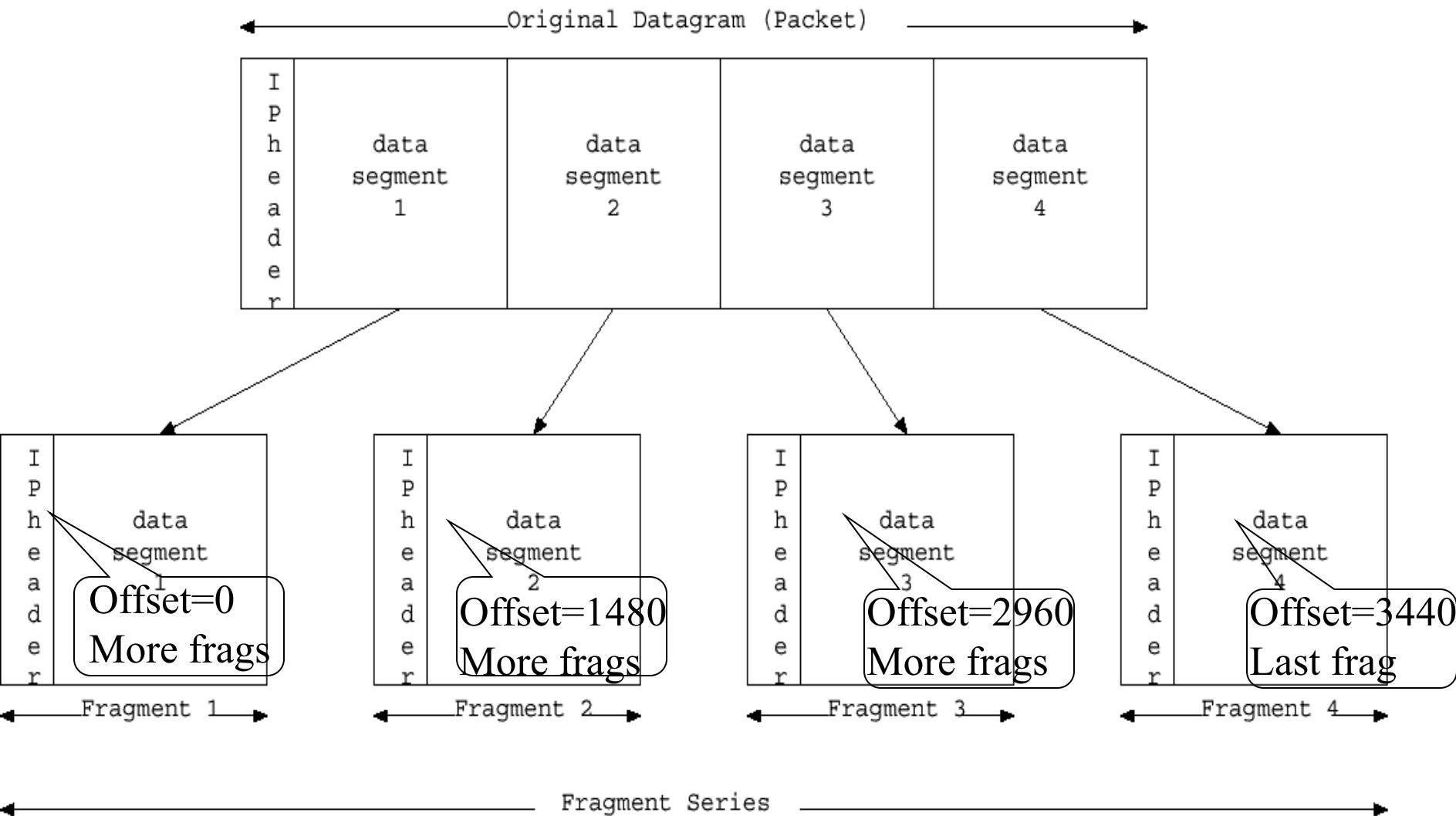


- Routeur fragmente les datagrammes de 1400 bytes
  - En 600 bytes, 600 bytes, 200bytes (20 bytes pour l'en-tête IP)
  - Routeurs ne re-assemblent pas les fragments

# Contrôle de Fragmentation

- **Identification:** permet à la destination de connaître l'origine de chaque fragment
- **Fragment Offset (13 bits):** permet la localisation des données transportées dans le fragment courant par rapport au datagramme initial
  - Mesuré en unités de 8 bytes commençant par 0
- **Flags (3 bits):** contrôle la fragmentation
  - Réservé (0 bit)
  - Don't Fragment - DF (1<sup>er</sup> bit):
    - 1 → n'est pas fragmenté
  - More Fragments - MF (2<sup>ième</sup> bit): 1 → données qui suivent
- Environ de 0.1% - 0.5% des paquets TCP sont fragmentés.

# Exemple de Fragmentation



# Internet Adressage

- 32 bits par adresse
  - L'adresse réfère une interface qu'une machine
  - Composée de deux parties
    - Identificateurs de network et host
  - Class A, B, C for unicast
  - Class D for multicast
  - Class E réservée
- 4 octets/bytes en format décimal
  - E.g. 134.79.16.1, 127.0.0.1

# Classes d'adresses d'Internet

- Class A: plus de hosts, peu de networks
  - Onnnnnnn hhhhhhhh hhhhhhhh hhhhhhhh
    - 7 bits network (0 et 127 réservés, soit 126 networks), 24 bits host (> 16M hosts/net)
- Class B: nombre de hosts et networks
  - 10nnnnnn nnnnnnnn hhhhhhhh hhhhhhhh
    - 16,384 class B networks, 65,534 hosts/network
    - Plage d'adressage 128-191 (décimal)
- Class C: grand nombre de networks
  - 110nnnnn nnnnnnnn nnnnnnnn hhhhhhhh
    - 2,097,152 networks, 254 hosts/network
    - Initial byte 192-223 (decimal)
- Class D: 224-247 (decimal) ( RFC 1112)
- Class E: 248-255 (decimal)

# Subnets: sous adressage

- Le mask du réseau est appliqué pour déterminer comment le réseau est sous adressé
- exemple :

137.138.28.228, et le mask sous réseau 255.255.255.0 alors 228 représente la machine est 137.138.28.0 le réseau

# Conversions de subnet mask

Prefix Length	Subnet Mask	Prefix Length	Subnet Mask
/1	128.0.0.0	/17	255.255.128.0
/2	192.0.0.0	/18	255.255.192.0
/3	224.0.0.0	/19	255.255.224.0
/4	240.0.0.0	/20	255.255.240.0
/5	248.0.0.0	/21	255.255.248.0
/6	252.0.0.0	/22	255.255.252.0
/7	254.0.0.0	/23	255.255.254.0
/8	255.0.0.0	/24	255.255.255.0
/9	255.128.0.0	/25	255.255.255.128
/10	255.192.0.0	/26	255.255.255.192
/11	255.224.0.0	/27	255.255.255.224
/12	255.240.0.0	/28	255.255.255.240
/13	255.248.0.0	/29	255.255.255.248
/14	255.252.0.0	/30	255.255.255.252
/15	255.254.0.0	/31	255.255.255.254
/16	255.255.0.0	/32	255.255.255.255

Decimal Octet	Binary Number
128	1000 0000
192	1100 0000
224	1110 0000
240	1111 0000
248	1111 1000
252	1111 1100
254	1111 1110
255	1111 1111

# Problèmes d' Adressage

- En 1991 IAB (Internet Architecture Board) identifié 3 dangers
  - Pénurie des adresses de classe B
  - Augmentation des réseaux va exploser les tables de routage
  - Augmentation des nets/hosts dépasse l'espace d'adressage de 32 bits
- Quatre stratégies d'adressage
  - Creative address space allocation {RFC 2050}
  - adresses privées{RFC 1918}, Network Address Translation (NAT) {RFC 1631}
  - Classless InterDomain Routing (CIDR) {RFC 1519}
  - IP version 6 (IPv6) {RFC 1883}

# Creative IP address allocation

- Trois organisations d'attribution d'adresses
  - APNIC - Asia & Pacific [www.apnic.net](http://www.apnic.net)
  - ARIN - N. & S. America, Caribbean & sub-Saharan Africa [www.arin.net](http://www.arin.net)
  - RIPE - Europe and surrounding areas  
[www.ripe.net](http://www.ripe.net)

# Adresses privées

- IP adresses non routables, utilisées par les entreprises en interne "ne sont pas assignées à des réseaux au sein de l'Internet global"
- Trois plages:
  - 10.0.0.0 - 10.255.255.255 a single class A net
  - 172.16.0.0 - 172.31.255.255 16 contiguous class Bs
  - 192.168.0.0 - 192.168.255.255 256 contiguous class Cs
- Connectivité fournit par Network Address Translator (NAT)
  - correspondre une adresse privée à une adresse IP publique (routable)
  - seulement pour les paquets TCP/UDP

# Classless InterDomain Routing (CIDR)

- Beaucoup d'organisation ont > 256 machines mais peu ont plusieurs milliers
- À la place d'attribuer une class B (16384 nets), on attribue des adresses de classe C
  - < 256 adresses → 1 class C
  - ...
  - < 8192 adresses → 32 Class C

# CIDR & Supernetting

- Blocs d'adresses CIDR représentés par un préfixe et préfixe long
  - Préfixe = seule adresse représente le bloc de réseaux
    - $192.32.136.0 = 11000000\ 00100000\ 10001000\ 00000000$
    - $192.32.143.0 = 11000000\ 00100000\ 10001111\ 00000000$
  - Préfixe long indique le nombre de bits de routage
    - $192.32.136.0/21 \rightarrow$  21 bits utilisés pour le routage
    - CIDR regroupe tous les réseaux entre 192.32.136.0 et 143.0 en une seule entrée dans le routeur- réduire les entrées dans la table de routeur
- Voir en détail RFC 1519

# IPV6

## □ Pourquoi un nouveau Protocole IP ?

Actuellement la taille de l'Internet double tous les 12 mois

- problèmes à résoudre l'épuisement des adresses IP (2008 +/- 3 ans)
- Et l'explosion de la taille des tables de routage
- le nouveau protocole doit permettre d'adresser un espace (beaucoup) plus grand (10 E+9 réseaux au minimum)
- un routage plus efficace

# de IPv4 à IPv6

Pour résoudre ces problèmes :

Nouvelle version de Internet Protocol : Version 6

- actuellement : IPv4
- IANA: Internet Assignment Numbers Authority (RFC 1700) Ce nouveau protocole (IPv6) :
  - garde ce qui a fait le succès de l'Internet
  - étend la fonction d'adressage et de routage
  - tend à résoudre les problèmes qui vont devenir critiques (applications temps réel, multipoint, sécurité...)

# IPV6 :Quelques Caractéristiques

- o Adresse plus longue : 128 bits (16 octets)
  - o Adressage de  $340 \times 10 \text{ e}36$  équipements
  - o Adressage hiérarchique
- 
- 3 types d'adresses :
    - o Unicast
    - o Multicast
    - o *Broadcast*

# IPV6 :Quelques Caractéristiques

- En-tête simplifié
  - nombre de champs réduit de moitié => augmente l'efficacité de commutation des équipements de routage
- Extension de l'en-tête pour les options
  - Les options IPv6 sont placées dans des en-têtes séparés, intercalés entre l'en-tête IPv6 et l'en-tête de la couche transport=> introduction aisée de nouvelles fonctionnalités
  - la longueur des options n'est plus limitée à 40 octets

# IPV6 : Nouvelles fonctionnalités

- Autoconfiguration : "*plug and play*"
  - Gestion de la mobilité
  - Renumérotation facile si changement de prestataire
  - Serveurs d'adresses (DHCP : *Dynamic Host Configuration Protocol*)
  - et SAA : *Stateless Address Autoconfiguration (RFC 1971)*
- Multipoint (*Multicast*) inclus de base pour les routeurs et les clients
  - "scope" = meilleur routage des paquets multicast => plus besoin de Mbone ni de mrouted

# IPV6: Nouvelles fonctionnalités

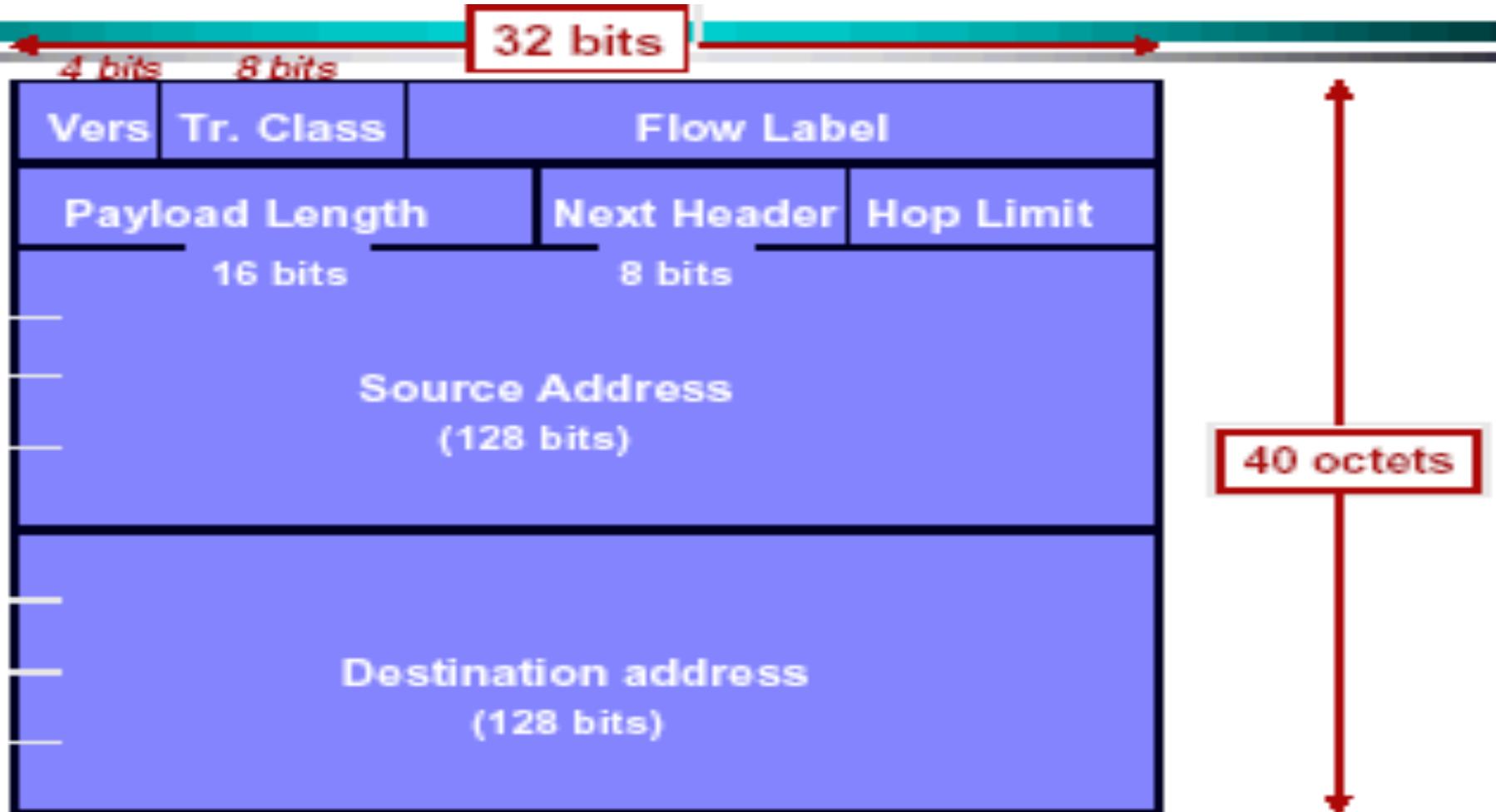
- "Marquage" des flux particuliers : (*Flow Label*)
  - applications temps réel, Qualité de Service ( QoS)
  - Priorité du trafic de contrôle
- Sécurité :
  - authentification et intégrité des données
  - *en option* : confidentialité
- Routage à partir de la source
  - Source Demand Routing Protocol

# IPv4 -> IPv6

## changements de l'en-tête

- o Header Length (IHL) : supprimé
- o ToS --> Flow Label
- o Total Length ( TL) --> Payload Length
- o ID, Flags et Fragment Offset (FO) : supprimés
- o TTL --> Hop Limit
- o Protocol --> Next header (mêmes valeurs que dans IPv4)
- o Header CS : supprimé
- o Adresses : 32 --> 128 bits (4 --> 16 octets)
- o Alignement 32 --> 64 bits

# IPV6 : En-tête



# IPV6: les champs de l'en-tête

- o Vers : Version Number (= 6)
- o Traffic Class : priorité ou classes de trafic (Differentiated Services)
- o Flow label : marquage des paquets «spéciaux»
- o Payload length : longueur du paquet après en-tête (en octets).  
autorise des paquets > 64 Koctets => Payload length = 0
- o Next header : indique le type d'entête suivant immédiatement l'entête IPv6  
On utilise les mêmes valeurs que dans le champ "Protocol" de IPv4 pour référencer les protocoles de niveau 4 (TCP=6 , UDP=17, ICMP=1)

# IPV6:les champs de l'en-tête

- Hop\_limit : -1 chaque fois que le paquet est commuté par un équipement si hop\_limit = 0 => le paquet est détruit.  
permet de réduire l'effet des boucles de routage.
- Source address : @ de l'émetteur initial du paquet
- Destination adress : Une adresse de destination ...  
peut-être différente de l'adresse destination finale si  
l'option "Routing Header" est présente.

# Concepts de routage

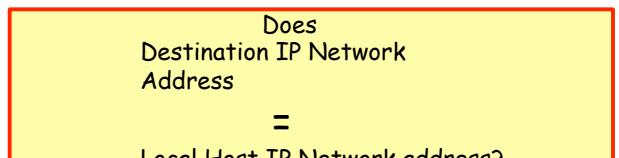
- Deux types basics de routage IP :
  - **Routage Direct.**
    - ➡ La machine destination est directement attachée au même support physique “réseau” comme la machine source
    - ✓ Le datagramme IP est encapsulé dans une trame physique (i.e. Ethernet).
  - **Routage Indirect.**
    - ➡ Attachement physique de la source et la destination n'est pas nécessaire
    - ➡ Un seul chemin pour rejoindre la destination via un ou plusieurs routeurs
    - ➡ L'adresse du premier routeur(default gateway) est la seule adresse exigée par la source

# Concepts de routage

## Table de routage IP

- Chaque host/router garde un ensemble de chemins avec **le routeur next** dans la direction de la Destination
- Ces chemins sont stockés dans la **table de Routage IP**. Cette table contiendra trois chemins différents:
  - ✓ **Routes directes** pour les réseaux attachés localement.
  - ✓ **Routes indirectes** pour les réseaux atteints via un ou plusieurs réseaux
  - ✓ Une **default route** dans le cas où le réseau destination n'a pas de table de routage
- Un Algorithme de routage

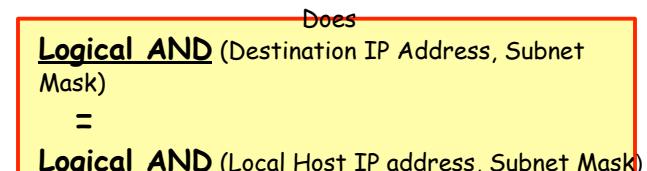
### IP Routing without Subnets



Yes  
Send IP Datagram to the **directly attached network**.

No  
Send IP Datagram to the **interface (gateway)** corresponding to the destination IP (sub)network address.

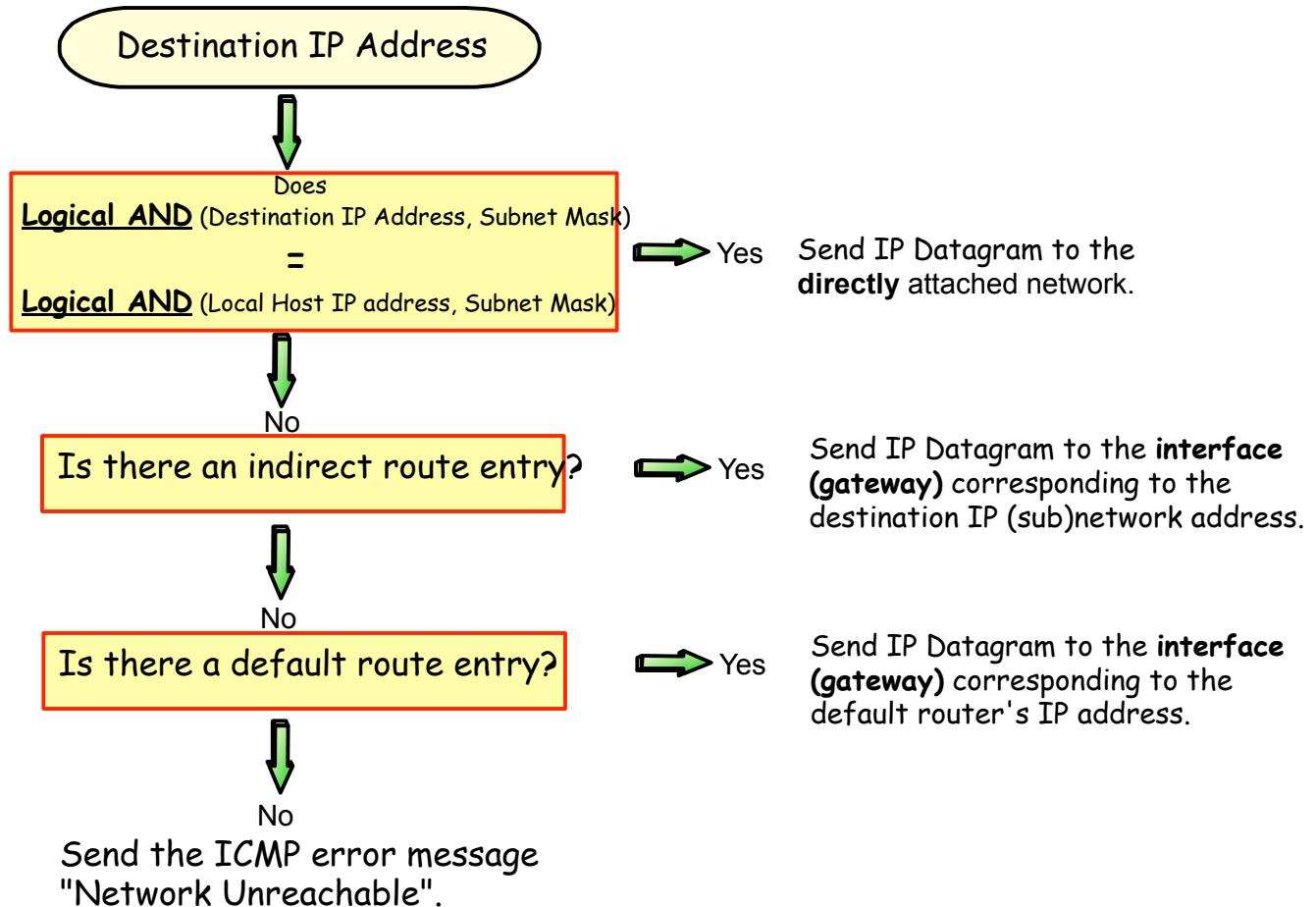
### IP Routing with Subnets



Yes  
Send IP Datagram to the **directly attached network**.

No  
Send IP Datagram to the **interface (gateway)** corresponding to the destination IP (sub)network address.

# Algorithme de routage w/ Subnets



# ADDRESS RESOLUTION PROTOCOL (ARP)/ REVERSE ARP(RARP)

# ARP

## □ Problème

- IP est une méthode d'adressage utilisée au niveau réseau pour identifier les machines et les réseaux

C'est une adresse logique et n'est pas physique/ MAC adresse.

Physique/ MAC adresses identifient les stations au niveau liaison

- méthode d'adressage TCP/IP n'était pas désignée pour les LANs  
LANs exigent que les stations connaissent chaque adresse physique pour établir la communication.

La station connaît l'adresse IP destination mais ne connaît pas l'adresse MAC destination

Comment la station obtiendra-t-elle l'adresse MAC destination?

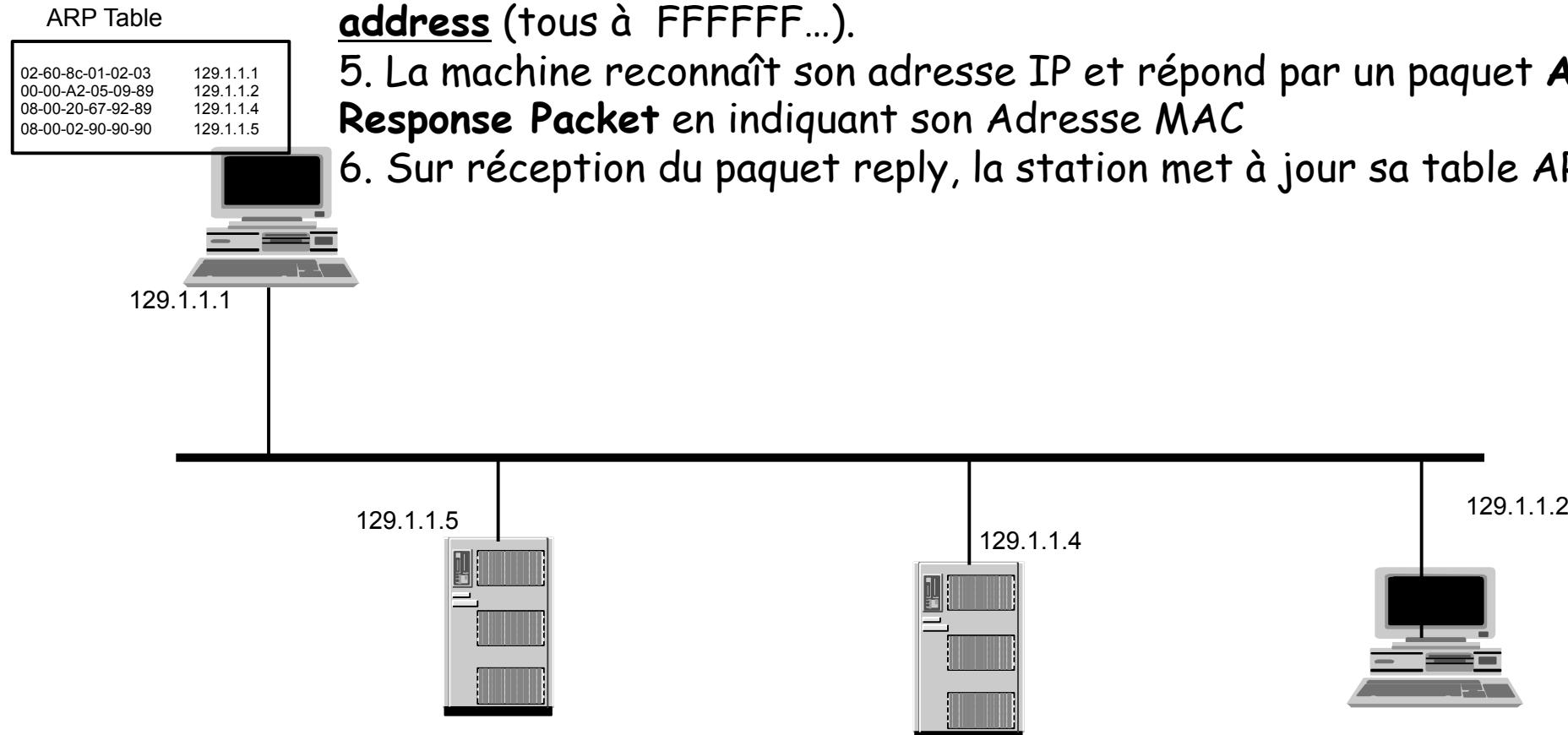
## □ Solution

- ✓ Mappage (correspondre) de l'address IP (32 bits) dans l'adresse MAC (48 bits)
- ✓ TCP/IP utilisera ARP pour trouver l'adresse physique de la station destination

# ARP : CONCEPT

- 1 IP demande une **MAC address** de la table **ARP**
2. Station recherche dans l'entrée dans la table ARP
3. Si l'adresse est dans la table alors retourne MAC address à IP
4. Si l'adresse MAC n'est pas dans la table, un paquet **ARP Request** généré sur le réseau en utilisant physical Broadcast address (tous à FFFFFF...).

5. La machine reconnaît son adresse IP et répond par un paquet **ARP Response Packet** en indiquant son Adresse MAC
6. Sur réception du paquet reply, la station met à jour sa table ARP



# ARP Table

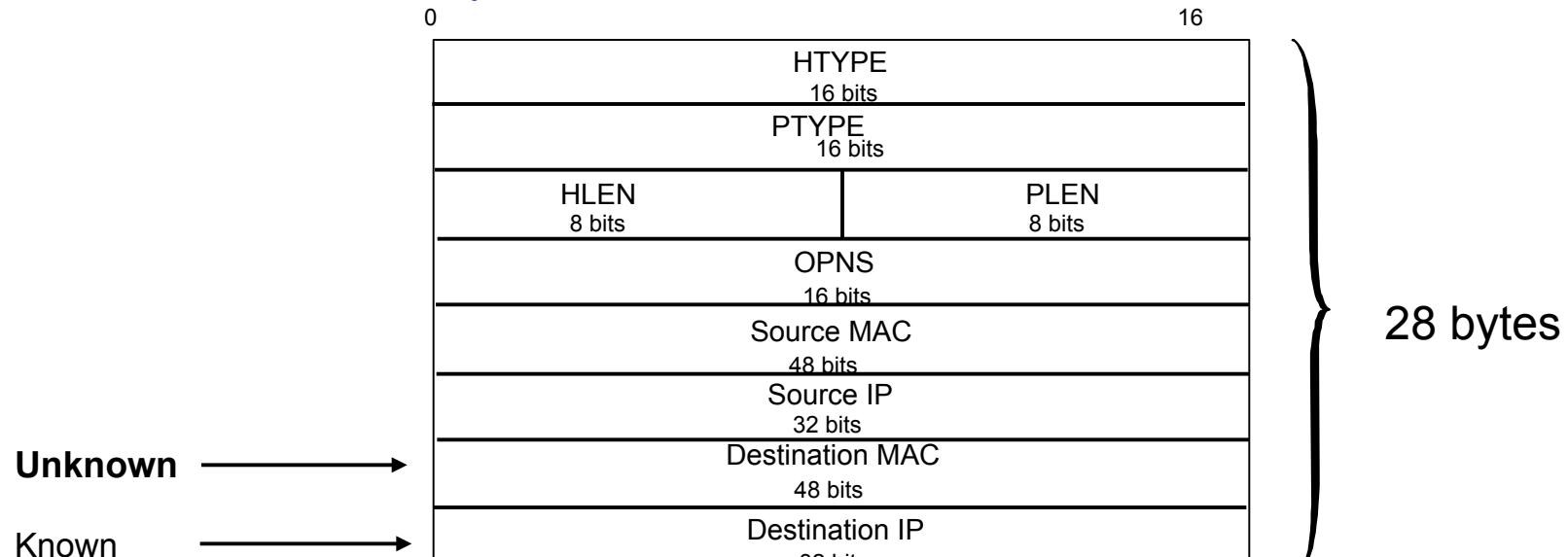
- ARP Table Size: adresses IP
  - Routeurs peut avoir plusieurs centaines d'entrées
  - les hosts quelques entrées.

- ARP Timeout

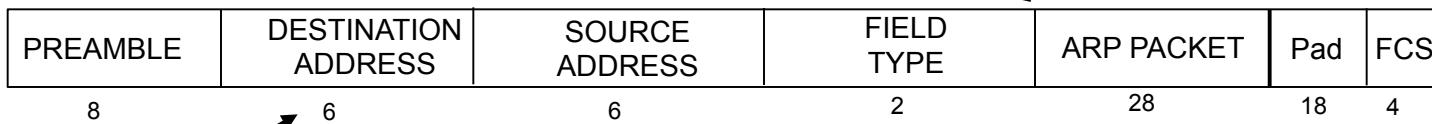
Variable entre 120s à 4heures, indique le temps d'expiration (effacement) de l'adresse de la table

- Adresses MAC

# Format de paquet ARP



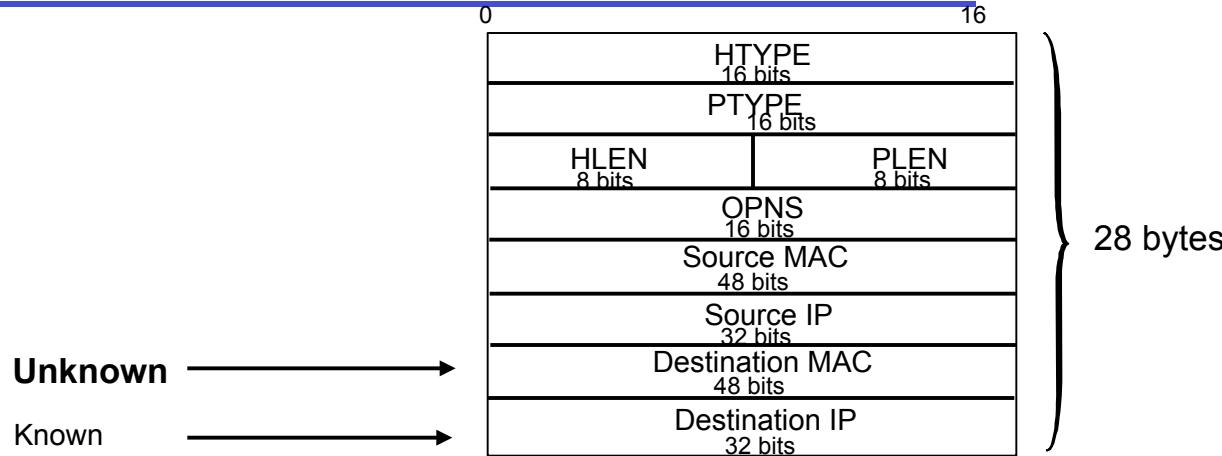
ETHERNET



- ✓ A **Broadcast** for an ARP Request.
- ✓ A **Unicast** for a ARP Reply

$0806_{16}$  = Ethernet type for ARP Request and Response.

# ARP PACKET FORMAT

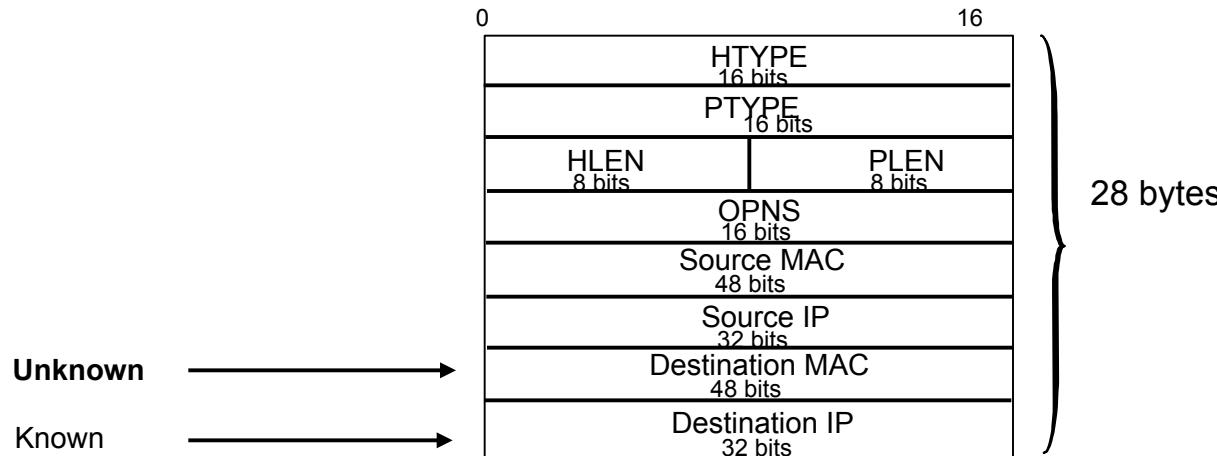


- **HTYPE.** The **hardware type field** indicates the network (hardware interface type) for which the sender seeks an answer. This will be **1** for Ethernet.
- **PTYPE.** The **protocol type field** specifies the type of high-level protocol address the sender has supplied. This is normally **0800h** for IP addresses.
- **HLEN.** The **hardware address length field** specifies the size in bytes of the hardware address. This field will normally contain the **value 6** (the length of the MAC address field is 48 bits).
- **PLEN.** The **protocol address length field** specifies the size in bytes of the high-level protocol address. This field will normally contain the **value 4** (the length of the IP address is 32 bits).
- **OPNS.** The **operations field** specifies whether the packet is an **ARP request** (a value of **1**) or an **ARP reply** (a value of **2**). The field is required since the **Ethernet field type** will contain the same value for both ARP request and ARP reply.

# Hardware Types

Hardware Type	Decimal Code
DIX -Ethernet and FDDI	1
IEEE 802 (802.3 and 802.5)	6
ARCnet	7
LocalTalk	11
SMDS	14
Frame Relay	15
ATM	19
Serial Line	20

# ARP : format de paquet



- **Source MAC.** The physical/MAC address of the source station requesting the address resolution.
- **Source IP.** The IP address of the source station requesting the address resolution.
- **Destination MAC.** The physical/MAC address of the destination station which will resolve the address mapping. This field will normally be set to all Os in the request packet because this field is unknown.
- **Destination IP.** The IP address of the destination station which will resolve the address mapping.

NOTE:

1. For an ARP Request all the fields are filled in except the destination MAC address.
2. When the host receives the ARP request, it fills in its hardware address and swaps the two sender addresses with the two destination address.
3. The **OPNS field** is set to 2(this indicates a reply) and a reply is sent back to the requesting station.
4. The reply can either be broadcast or unicast but is normally unicast.

# RARP

## □ Problème

- **MAC dépend du matériel** cependant **IP adresses sont indépendantes**.
- Les programmes d'Application utilisent **IP adresse** pour spécifier une destination. Cette adresse est enregistrée sur le disque de l'ordinateur, le système au démarrage lit cette adresse
- Stations sur le réseaux utilisent Physical/ **MAC adresses** pour spécifier la destination.

ARP est utilisé pour accomplir la correspondance entre deux adresses

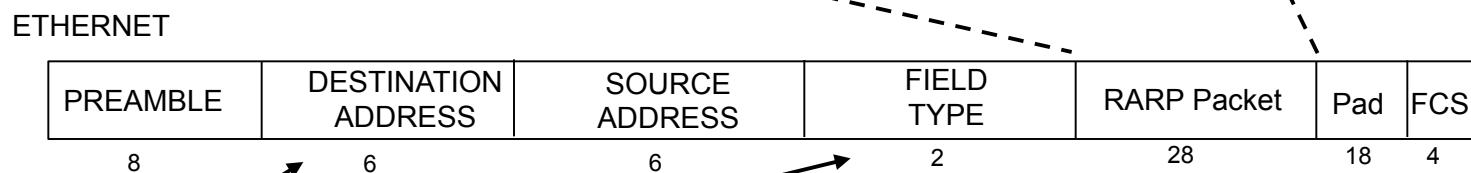
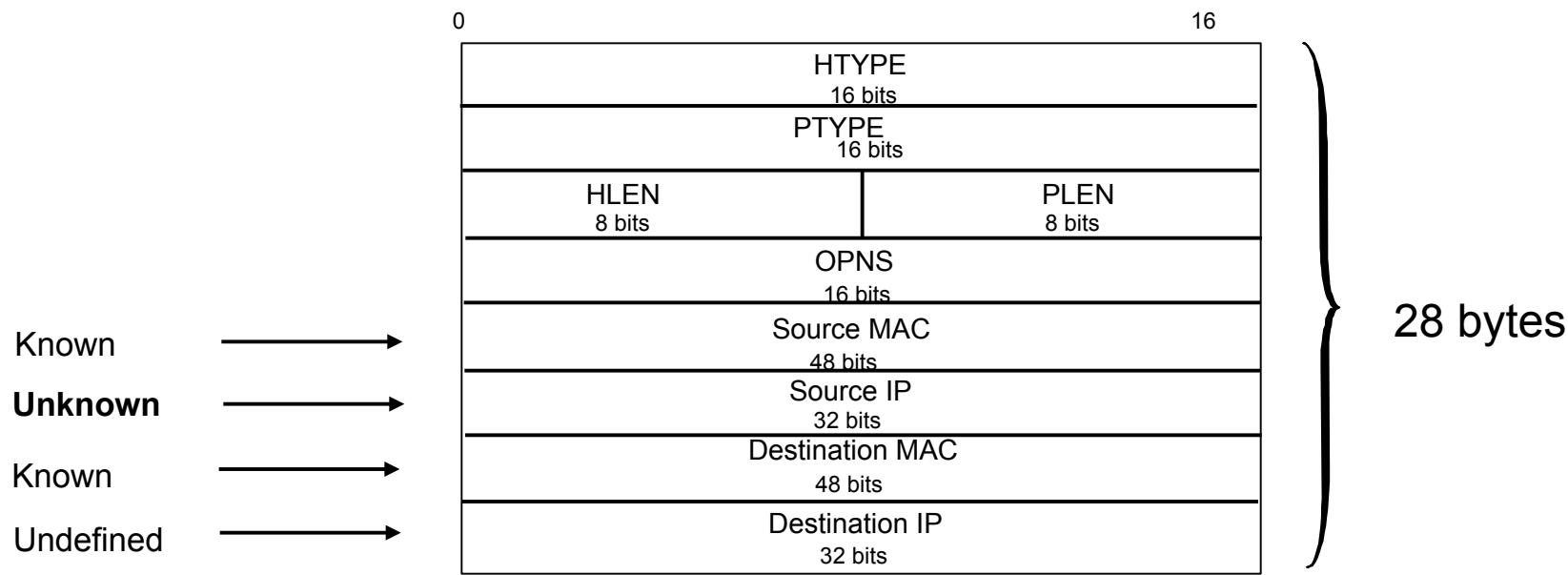
La station source connaît son **MAC address** mais pas son **IP address**

Comment une station du travail **diskless** obtient une IP address?

## □ Solution

- TCP/IP utilisera RARP pour trouver **Internet Address** à travers le réseau via **RARP server**
- Chaque réseau doit posséder au moins un **RARP server**
- IP fait correspondre une IP adresse avec une adresse MAC

# ARP/RARP PACKET FORMAT



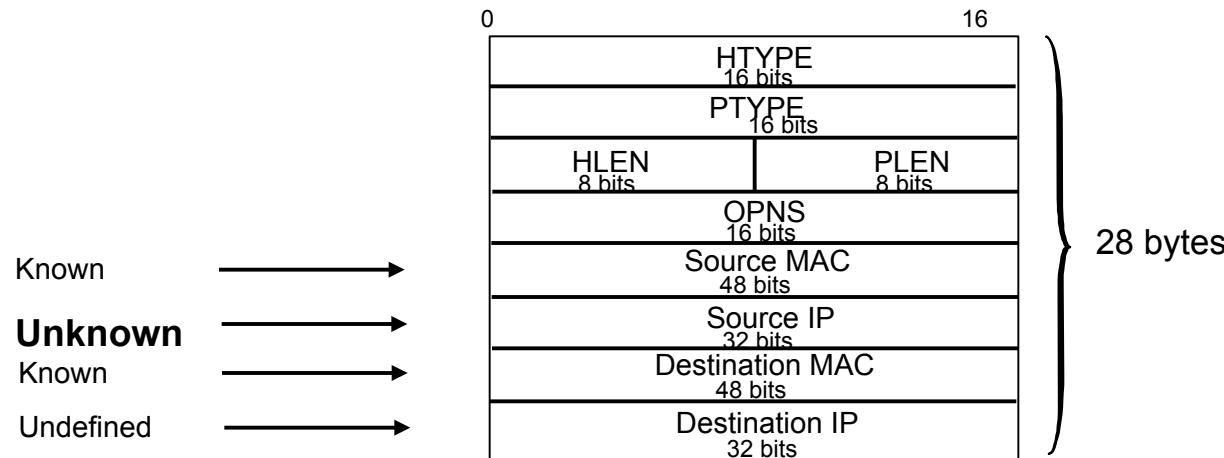
$8035_{16}$  = Ethernet type for RARP Request and Response.

A **Broadcast** for an RARP Request.

A **Unicast** for a RARP Reply

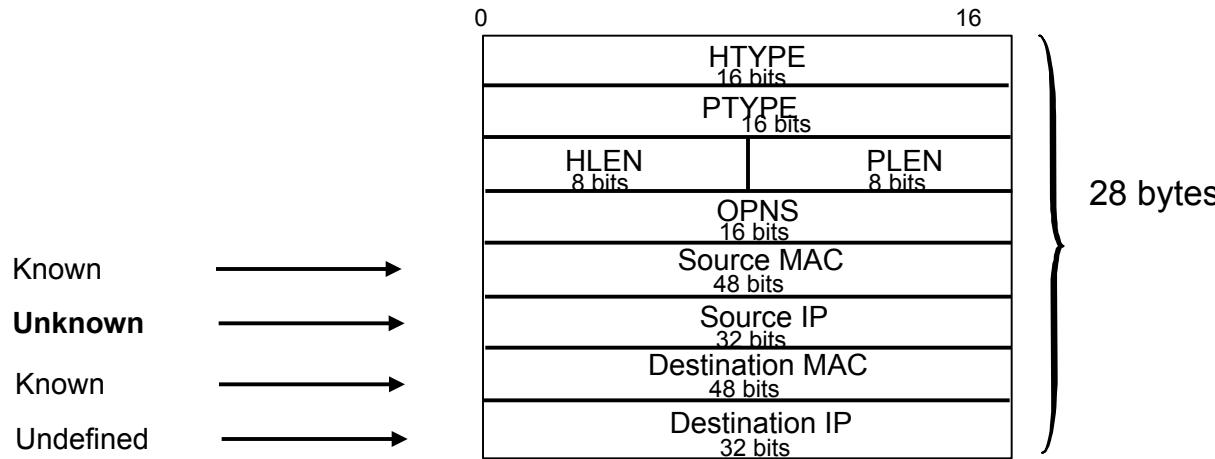
RARP will normally reside on a prom in the network adapter card.

# ARP/RARP PACKET FORMAT



- **HTYPE.** The **hardware type field** indicates the network(hardware interface type) for which the sender seeks an answer. This will be 1 for Ethernet.
- **PTYPE.** The **protocol type field** specifies the type of high-level protocol address the sender has supplied. This is normally 0800h for IP addresses.
- **HLEN.** The **hardware address length field** specifies the size in bytes of the hardware address. This field will normally contain the value 6(the length of the MAC address field is 48 bits).
- **PLEN.** The **protocol address length field** specifies the size in bytes of the high-level protocol address. This field will normally contain the value 4 (the length of the IP address is 32 bits).
- **OPNS.** The **operations field** specifies whether the packet is an **RARP request** (a value of 3) or an **RARP reply** (a value of 4). The field is required since the **Ethernet field type** will contain the same value for both RARP request and RARP reply.

# ARP/RARP PACKET FORMAT



- **Source MAC.** The Physical/MAC address of the source station requesting the address resolution. This field is known.
- **Source IP.** The IP address of the source station requesting the address resolution. This field is unknown and will normally be set to all 0s. It will be filled in by the responding RARP server.
- **Destination MAC.** For a RARP Request, the physical/MAC address of the source station requesting the address resolution.
- **Destination IP.** The IP address of the RARP server. This is unknown.

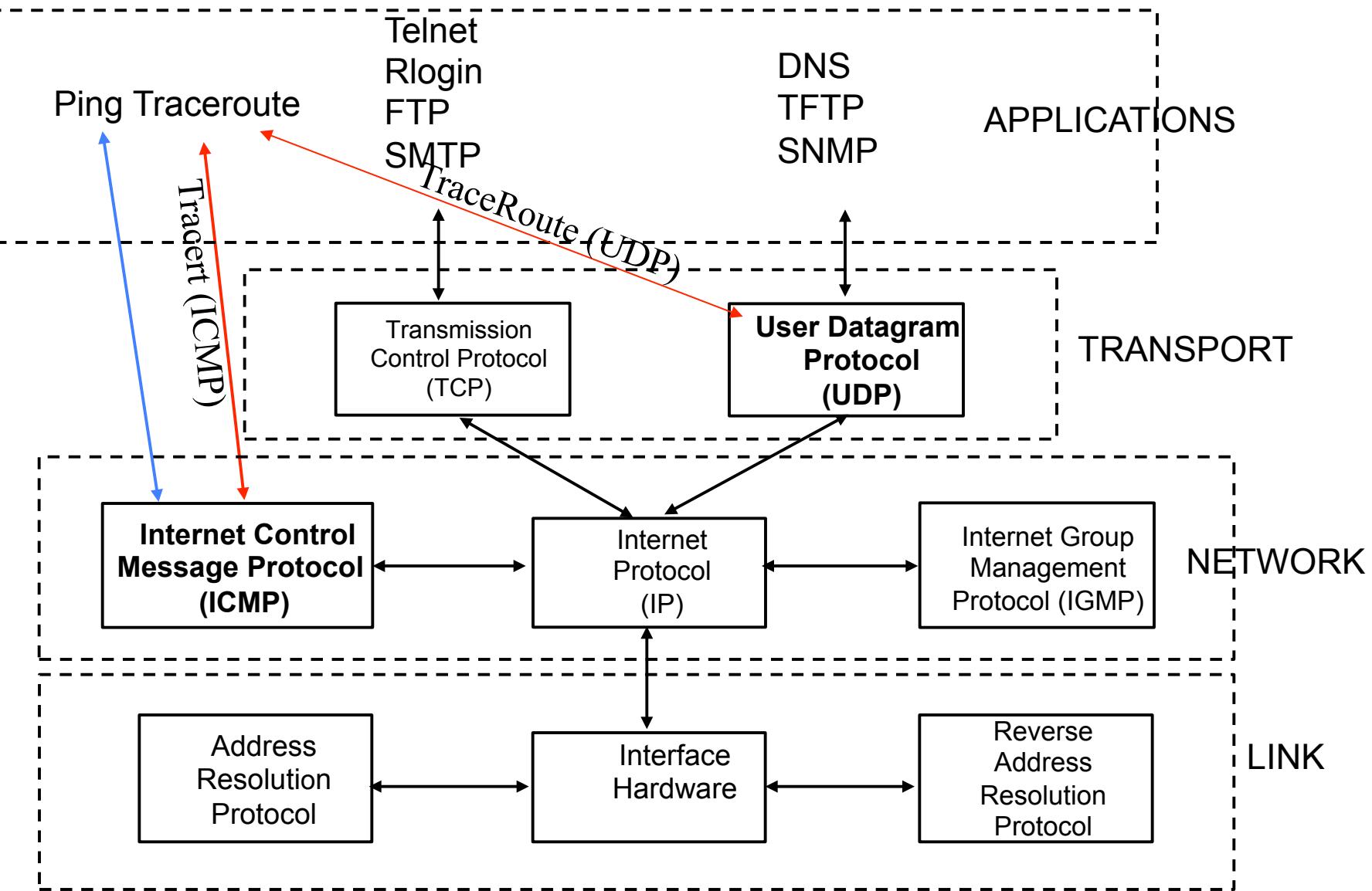
NOTE:

1. For an RARP Request, the **Source MAC** and **Destination MAC** addresses are filled in with the physical address of the requesting station. The **Source IP** is Unknown and will be filled in by RARP server. The **Destination IP** is unknown.
2. For an Rarp Response, the **Source MAC** and **Source IP** addresses contain the physical address and logical address of the RARP server. The **Destination MAC** contains the physical address of the requesting station while the **Destination IP** contains the IP address of the requesting station assigned by the RARP server.
3. The **OPNS** field is set to 4 to indicate a reply.

# INTERNET CONTROL MESSAGE PROTOCOL -ICMP -

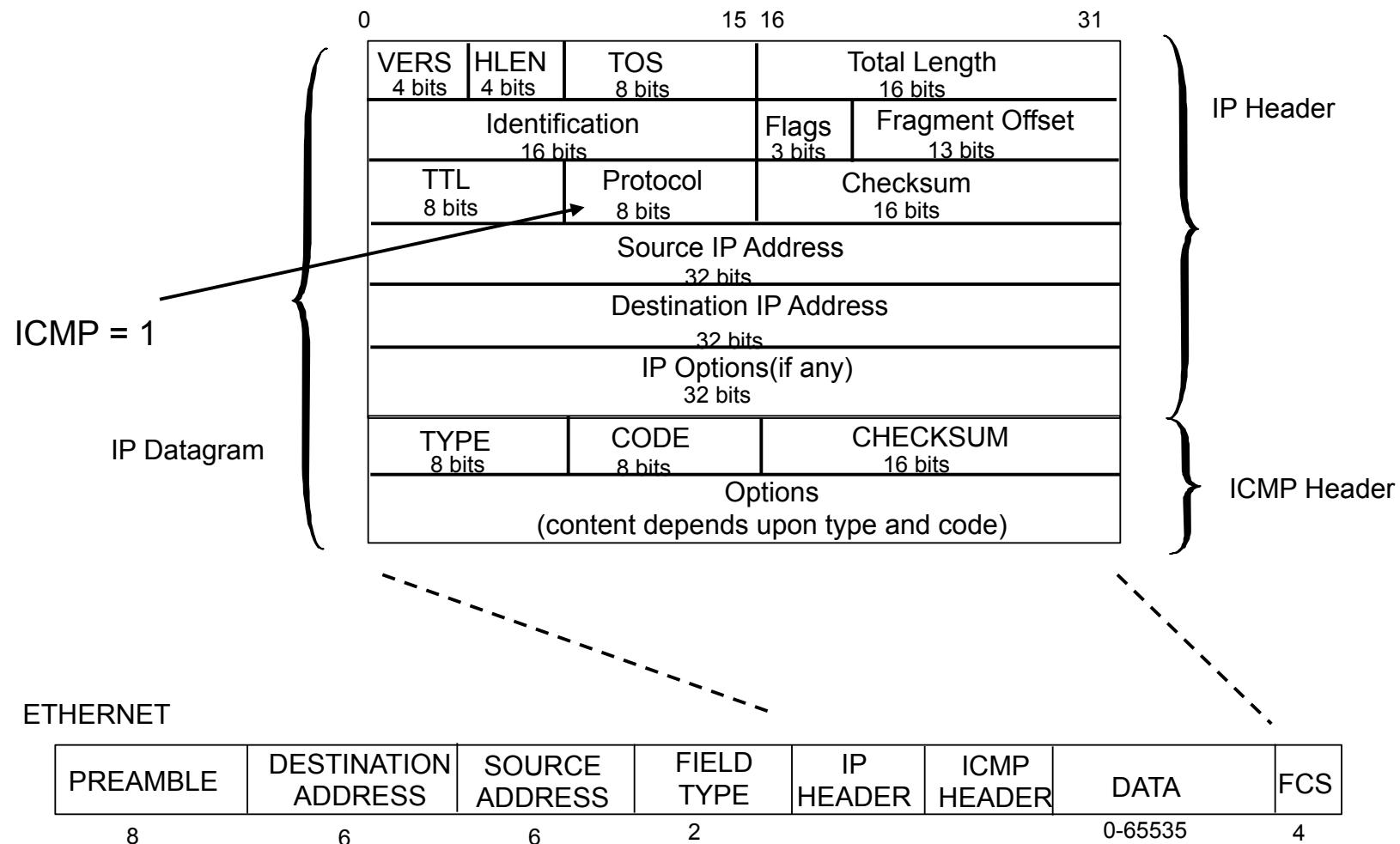
# ICMP

- ICMP fait un compte rendu sur les conditions d'erreur à la source
- ICMP permet aux routeurs d'envoyer des messages d'erreur ou de supervision à d'autres routeurs ou machines
- ICMP offre un mécanisme de communication entre le logiciel IP d'une machine et celui d'une autre machine
- Le message ICMP est **encapsulé** dans un datagramme IP et il est considéré comme une partie intégrante du protocole IP
- Le message d'erreur ICMP contient l'**en-tête IP** et **8 premiers bytes** de message
- Dans l'ordre d'éviter la congestion, un message ICMP n'est jamais envoyé en réponse au:
  - Un message d'erreur ICMP
  - Une adresse broadcast ou multicast
  - Un autre fragment que le premier
  - Une adresse source qui ne définit pas une seule machine

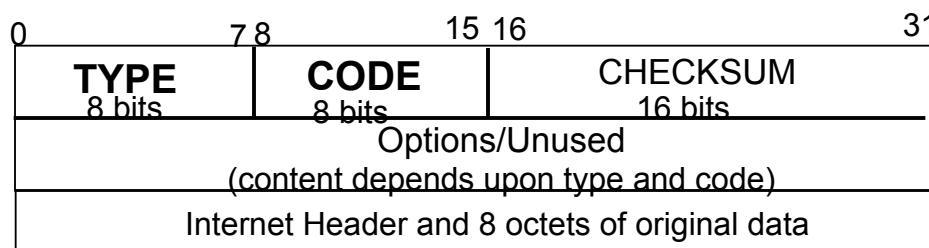


TCP/IP Protocole Suite

# ICMP ENCAPSULATION



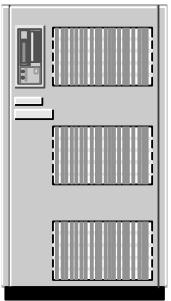
# ICMP: ENCAPSULATION



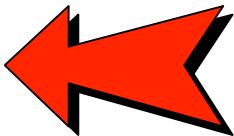
- **TYPE**. Type de message (**20 ICMP type messages**)
- **CODE**. Fournit des infos. Supplémentaires sur le type de message
- **CHECKSUM**. Le même comme IP, mais pour le message ICMP (y compris les données)
- **OPTION FIELD**. Le contenu dépend de **type de message** .

# ICMP: Messages d'erreurs

Host



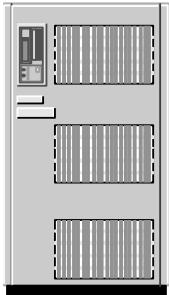
Router



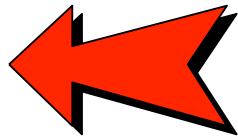
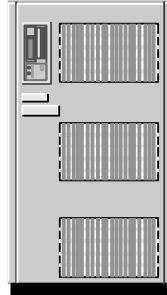
## Router Error Messages

- ✓ Destination is unreachable.
- ✓ Time-to-Live expired at router.
- ✓ Bad parameter in the IP header
- ✓ A better route is available

Host



Host



## Host Error Messages

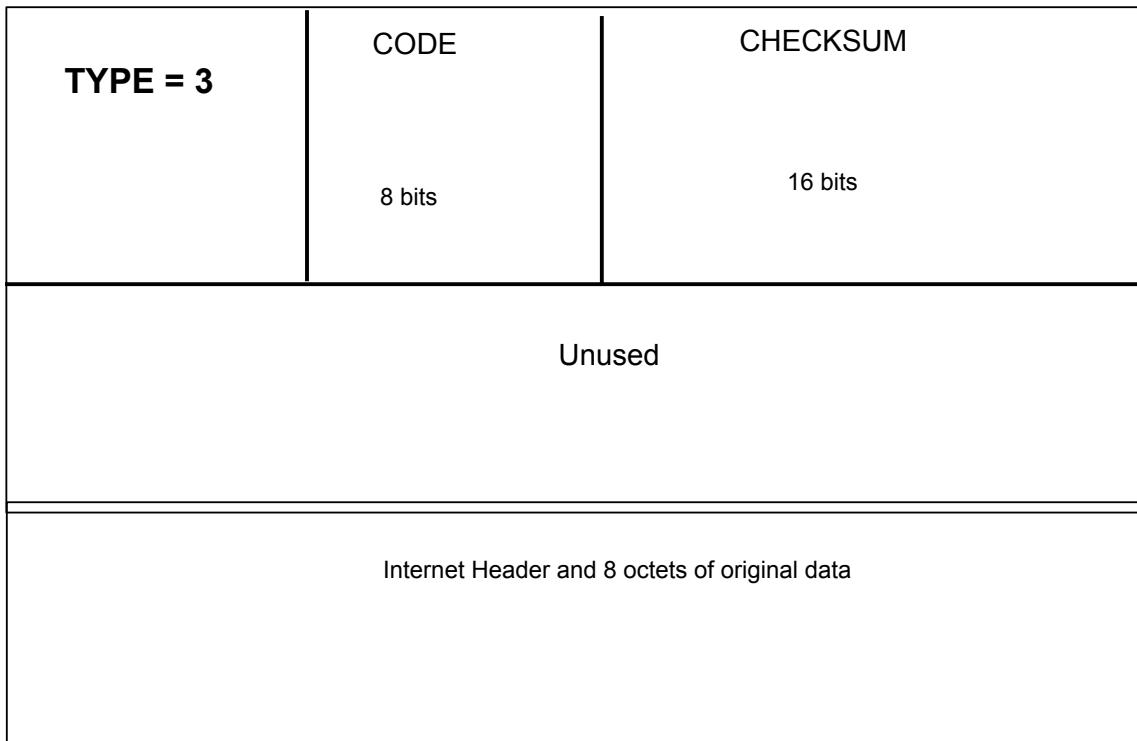
- ✓ Service is unreachable.
- ✓ Fragment reassembly time has expired.
- ✓ Bad parameter in the IP header

# ICMP : Types de messages

0	Echo Reply	RFC 792
3	Destination Unreachable	RFC 792
4	Source Quench	RFC 792
5	Redirect	RFC 792
8	Echo	RFC 72
9	Router Advertisement	RFC 1256
10	Router Solicitation	RFC 1256
11	Time Exceeded	RFC 792
12	Parameter Problem	RFC 792
13	Timestamp	RFC 792
15	Information Request	RFC 792
16	Information Reply	RFC 792
17	Address Mask Request	RFC 950
18	Address Mask Reply	RFC 950
30	Traceroute	RFC 1393
32	Mobile Host Redirect	
33	IPv6 Where-Are-You	
34	IPv6 I-Am-Here	
35	Mobile Registration Request	
36	Mobile registration Reply	

# Destination inaccessible

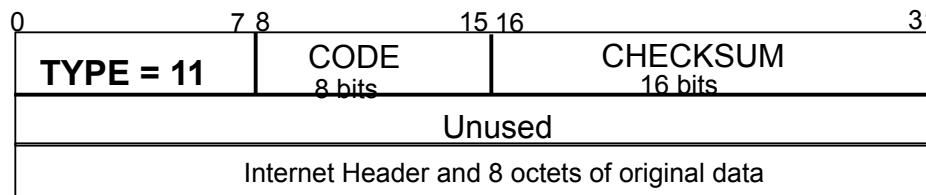
0                    7 8                    15 16                    31



# Destination inaccessible

Code	Description
0	Network is unreachable
1	Host is unreachable
2	Protocol is not supportable at destination
3	Port unreachable (application is probably not available)
<b>4</b>	<b>Fragmentation required but DF bit is set</b>
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated (obsolete)
9	Destination network administratively prohibited
10	Destination host administratively prohibited
11	Network unreachable for TOS
12	Host unreachable for TOS
13	Communication administratively prohibited by filtering
14	Host precedence violated
15	Precedence cutoff in effect

# Délai excessif



- **Time-to-Live** est expiré au routeur ou **Fragment Reassembly Time** est expiré à la destination avant l'arrivée des fragments → route circulaire ou excessivement longue
  - **Time-to-Live** field sets an upper bound on how many routers a datagram can transit.
    - ✓ initialisé par l'émetteur (32 ou 64) et décrémenté à chaque passage par un routeur
    - ✓ il est basiquement limité au lifetime de datagramme afin d'éviter la boucle
    - ✓ quand le TTL zéro ,le **datagramme est écarté** et l'émetteur est notifié ICMP message
  - A **Fragment Reassembly Timer** est positionné par le récepteur à la réception du premier fragment (**60 et 120 secondes**), quand le timer expire **datagramme est écarté** et l'émetteur est notifié ICMP message

Code	Description
0	<b>Time-to-Live</b> expired while the datagram was in transit. Generated by the router.
1	<b>Fragment Reassembly</b> Time has expired. Generated by the host.

# Couche Liaison

Protocol Implementation							<u>OSI</u>
File Transfer	Electronic Mail	Terminal Emulation	File Transfer	Client Server	Network Mgmt	Application	
File Transfer Protocol (FTP) RFC 559	Simple Mail Transfer Protocol (SMTP) RFC 821	TELNET Protocol RFC 854	Trivial File Transfer Protocol (TFTP) RFC 783	Network File System Protocol (NFS) RFC 1024, 1057 and 1094	Simple Network Management Protocol (SNMP) RFC 1157	Presentation	
						Session	
Transmission Control Protocol (TCP) RFC 793			User Datagram Protocol (UDP) RFC 768			Transport	
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791	Internet Group Management Protocol (IGMP) RFC 2236		Internet Control Message Protocol (ICMP) RFC 792		Network	
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS	Data Link	
Transmission Mode						Physical	
TP	STP	FO	Satellite	Microwave, etc			

# Couche Liaison

- Le niveau liaison a pour rôle de fiabiliser la transmission physique des données et ainsi de diminuer le taux d'erreurs

## Fonctionnalités

- Détection des erreurs
- Correction ou reprise sur erreurs
- Gestion du contrôle de flux
- Structure des données en blocs ou trames  
(PDU : Protocol Data Unit)

# Protocole

- Un protocole est un ensemble de règles régissant les échanges entre plusieurs entités communicantes.
- Il définit :
  - format des unités de données échangées et leur type (information ou contrôle)
  - déroulement de ces échanges dans le temps (timing)

# Services de la couche liaison

Offre des services classés en trois catégories :

- ❑ Service sans connexion et sans acquittement, adapté quand :
  - Le taux d'erreur est faible
  - La correction de transmission est au niveau supérieure
  - Exemple : trafic temps réel (parole)
- ❑ Service sans connexion et avec acquittement
  - Acquittement à chaque réception de trame
  - Sémantique au moins une fois
- ❑ Service orienté connexion
  - Établissement préalable d'une connexion
  - Sémantique exactement une fois

# Contrôle de flux

Le récepteur reçoit des trames de données dans des buffers. La taille des buffers est limité.

Si la vitesse d'arrivée des trames est plus grande que celle du traitement → *Saturation (engorgement) & perte de trames*

## But du contrôle de flux

Asservir ou ralentir le flux d'émission à la capacité mémoire

Deux types de mécanisme :

- trame particulière du récepteur pour stopper l'émetteur
- fenêtre de contrôle de flux permettant l'autorégulation de l'émetteur

# Notion de Trames

*Objectifs:* fourni un service à la couche réseau, utilise le service de la couche physique

*Idée :* découpage en trames des trains de bits, calcul une somme de contrôle d'erreur pour chaque trame

Plusieurs méthodes :

➤ Compter les caractères

➔ Utilisation d'un champ dans l'en-tête

➔ Pb : si erreur du compteur => perte de synchronisation

➤ Utiliser des caractères de début et de fin de trame et des caractères de transparence

➔ DLE (Data Link Escape) et STX (Start of TeXt) en début de trame, et DLE et ETX (End of TeXt) en fin de trame

➔ Pb: données peuvent contenir ces séquences

➤ Utiliser des fanions de début et de fin de trame et des bits de transparence

➔ Fanion : 0111110, ajout de 0 après cinq bits consécutifs à 1

➤ Violer le codage utiliser dans la Couche Physique

# Protocole Send and Wait

## *Hypothèses :*

- A, B deux stations
- Transmission unidirectionnelle de données (de A vers B)
- A possède toujours des informations à transmettre
- Les temps de traitement sont négligeables
- Les tampons sont de capacité infinie
- Le canal est sans erreur

```
type-evt = (Arrivee_trame);  
void Emetteur();  
void Recepteur():
```

# Protocole Send and Wait

```
Void Emetteur() {  
    trame t;  
    paquet tampon;  
    type_evt evenement;  
  
    While (!Fin) {  
        Recup_paquet_reseau(tampon);  
        t.info := tampon;  
        Envoi_trame_physique(t);  
        Attendre(evenement);  
    }  
}
```

```
Void Recepteur() {  
    trame tr, ta;  
    type_evt evenement;  
  
    While (!Fin) {  
        Attendre(evenement);  
        Recup_trame_physique(tr);  
        Envoi_paquet_reseau(tr.info)  
        ;  
        Envoi_trame_physique(ta);  
    }  
}
```

# Protocole de liaison simple pour un canal bruité

## *Hypothèses :*

- A, B deux stations
- Transmission unidirectionnelle de données (de A vers B)
- A possède toujours des informations à transmettre
- Les temps de traitement sont négligeables
- Les tampons sont de capacité infinie
- Le canal n'est pas parfait

```
#define maxseq 1  
  
type-evt = (Arrivee_trame, erreur, timeout);  
  
void Emeteur();  
  
void Recepteur();
```

# Protocole de liaison simple pour un canal bruité

```
void Emetteur() {  
    num_sequence num_trame;  
    trame t; paquet tampon;  
    type_evt evenement;  
  
    num_trame := 0;  
    Recup_paquet_reseau(tampon);  
    While (!Fin) {  
        t.info := tampon;  
        t.seq := num_trame;  
        Envoi_trame_physique(t);  
        Debut_timer(t.seq);  
        Attendre(evenement);  
        if (evenement == Arrivee_trame)  
        {  
            Recup_paquet_reseau(tampon);  
            num_trame := (num_trame +1)  
                Mod 2;  
        }  
    }  
}
```

```
void Recepteur() {  
    num_sequence num_trame;  
    trame tr, ta; paquet tampon;  
    type_evt evenement;  
  
    num_trame:= 0;  
    While (!Fin) {  
        Attendre(evenement);  
        if (evenement == Arrivee_trame)  
        {  
            Recup_trame_physique(tr);  
            if (tr.seq == num_trame) {  
                Envoi_paquet_reseau(tr.info);  
                num_trame := (num_trame +1)  
                    Mod 2;  
            }  
            Envoi_trame_physique(ta);  
        }  
    }  
}
```

# Protocole avec fenêtre d'anticipation

## *Hypothèses :*

- A, B deux stations
- Transmission bidirectionnelle de données
- Les temps de traitement sont négligeables
- Les tampons sont de capacité infinie
- Le canal n'est pas parfait

```
#define maxseq 1  
  
type_evt = (Arrivee_trame, erreur, timeout);  
  
void Emeteur();  
  
void Recepteur();
```

# Protocole avec fenêtre d'anticipation

```
void Emetteur() {  
    num_sequence num_trame_envoye,  
        num_trame_attendu;  
  
    trame te, ta; paquet tampon;  
    type_evt evenement;  
  
    num_trame_attendu := 0;  
    num_trame_envoye := 0;  
    Recup_paquet_reseau(tampon);  
    te.info := tampon;  
    te.seq := num_trame_envoye;  
    te.ack := 1 -  
        num_trame_attendu;  
    Envoi_trame_physique(te);  
    Debut_timer(t.seq);  
    While (!Fin) {  
        t.info := tampon;  
        t.seq := num_trame;  
        Envoi_trame_physique(t);  
    }  
}
```

```
    Attendre(evenement);  
    if (evenement == Arrivee_trame)  
    {  
        Recup_paquet_reseau(tampon);  
        if (te.seq ==  
            num_trame_attendu) {  
            Envoi_paquet_reseau(te.info);  
            num_trame_envoye :=  
                (num_trame_envoye +1)Mod 2;  
        }  
    }  
    ta.info := tampon;  
    ta.seq := num_trame_envoye;  
    ra.ack := 1 -  
        num_trame_attendu;  
    Envoi_trame_physique;  
    Debut_timer(s.seq);  
}  
}
```

# High Data Link Control

HDLC a été adopté par le CCITT:

- En 1980, sous le nom de LAP (Link Access Procedure)
- En 1988, sous le nom de LAP-B (Balanced) qui gère le niveau liaison de l'interface X.  
25

## *Caractéristiques*

- Procédure synchrone
- Orienté bit
- Protocole en mode connecté
- Gestion d'une connexion logique
- Le dialogue se déroule en trois phases
  - Établissement de la liaison
  - Transfert de données
  - Libération de la liaison
- Point à point (LAP-B)
- Numérotation modulo 8 ou étendue modulo 128
- Fenêtre d'anticipation maximale de 8 ou étendue modulo 128
- Transparence au drapeau

# Trame HDLC

- drapeau (0111 1110) : toutes les trames doivent commencer et finir par un drapeau
- adress : indique l'adresse du destinataire de la trame
- contrôl : permet de définir les différents types de trames et de mettre les numéros de séquence et d'acquittement
- Data : les données à transmettre
- FCS (Frame Control Sequence) : 16 bits de contrôle d'erreur calculés par un code polynomial de polynôme générateur  $x^{16} + x^{12} + x^5 + 1$

8	8	8	16	8
flag	adress	contrôl	data	FCS

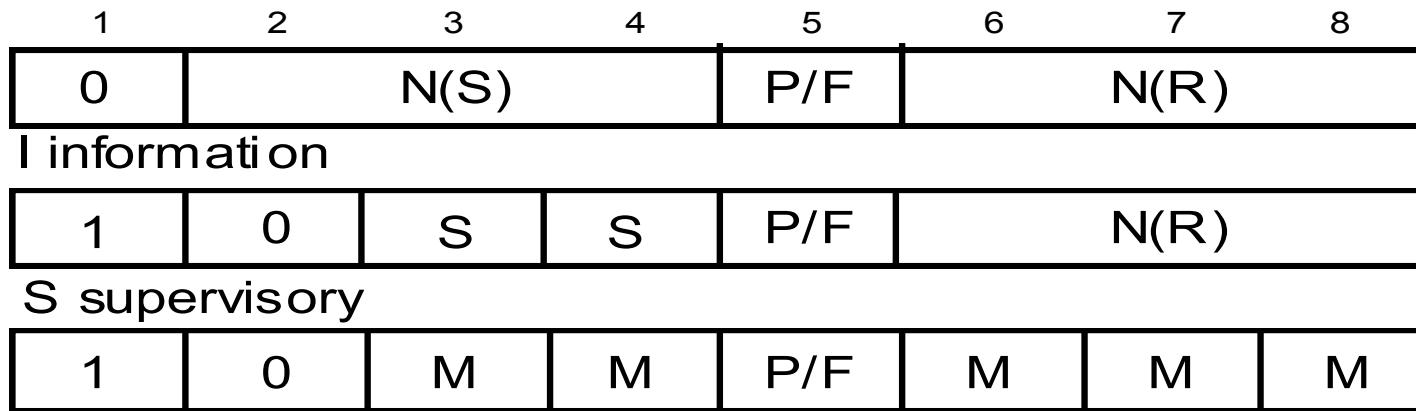
# Trame HDLC

- Transparence au drapeau
  - Eviter données = drapeau = 0111 1110
    - à l'émission, rajouter un bit 0 après cinq bits 1 consécutifs
    - à la réception, enlever un bit 0 après cinq bits 1 consécutifs
- Emission*                            *Réception*
- |                               |                                  |
|-------------------------------|----------------------------------|
| ➤ Construction de la trame    | ➤ Transparence (retrait)         |
| ➤ Calcul des bits de contrôle | ➤ Cohérence du contrôle d'erreur |
| ➤ Transparence (ajout)        | ➤ Examen de la trame             |

# Type de trame HDLC

## Trois types de trame HDLC

- I (Information) : trame d'information pour transporter les données
- S (Supervision) : trame de supervision pour la reprise sur erreur et le contrôle de flux
- U (Unnumbered) : trame non numérotée pour l'établissement et la libération de la liaison



U unnumbered  
N(S) : numéro de séquence en émission, N(R) : numéro de séquence en réception

S : élément de la fonction de supervision, M : élément de la fonction de modification, P/F : élément d'invitation à émettre / fin

# Etablissement/libération de la liaison

Etablissement en mode équilibré ou symétrique :

- États des stations : primaires
- Etablissement par la trame de commande SABM (Set Asynchronous Balanced Mode)
- Acquittement par une réponse UA (Unnumbered Acknowledge)
- Libération par la trame de commande DISC (DISConnect)
- Acquittement par une réponse UA

# Etablissement/libération de la liaison

Etablissement en mode non équilibré ou dissymétrique : états des stations : 1 primaire et n secondaires

## ***Mode normal***

- Une secondaire ne peut émettre que sur invitation du primaire
- Établissement par la trame de commande SNRM (Set Normal Response Mode)
- Acquittement par une réponse UA
- Libération de la liaison par DISC
- Acquittement par une réponse UA

## ***Mode asynchrone***

- Une primaire invite une fois la secondaire, ensuite la secondaire peut émettre quand elle désire
- Établissement par la trame de commande SABM (Set Asynchronous Balanced Mode)
- Acquittement par une réponse UA
- Libération de la liaison par DISC
- Acquittement par une réponse UA

# HDLC

*Le temporisateur* : dans le cas d'une

- erreur
- rupture de la ligne
- panne de la station distante
- perte d'acquittement

*la reprise* se fait à l'aide d'un temporisateur d'attente d'acquittement :

- un compteur à rebours initialisé à une certaine valeur
- décrémenter à chaque top d'horloge
- compteur = 0 ou expiration du temporisateur (timeout), génération d'une interruption

# Transfert de données

## *Numérotation modulo N*

- Numérotation cyclique (modulo N). les trames sont numérotées de 0 à  $N - 1$

## *Détection et correction d'erreurs*

- Détection : contrôle des bits contenus dans le champ FCS
- Correction : demande de retransmission

## *Contrôle de flux et fenêtre*

- Émission des trames I par anticipation dans la limite d'une fenêtre
- Une fenêtre est un crédit K alloué à l'émetteur en nombres de trames (en pratique  $K * \text{temps émission} > \text{temps de propagation}$ )
- Une fenêtre est un intervalle de numéros de séquence de trame que l'émetteur peut envoyer sans avoir reçu d'acquittement
- Une fenêtre est un intervalle à bornes glissantes (sliding window) de largeur K bonne inférieure = dernier acquittement  $N(R)$  reçu Borne supérieure = ( $\text{dernier acquittement } N(R) \text{ reçu} + K - 1$ ) MOD N

# Transfert de données

## *Remarque*

- K permet de définir le nombre de tampons nécessaires en réception comme en émission
- Au niveau liaison, K n'est pas négocié mais établi une fois pour toute à l'installation du contrôleur (pour Transpac c'est un paramètre d'abonnement)

## *MAJ des numéros de séquence portés par les trames*

- Deux compteurs
  - Dans la partie émission : N(S) indique le numéro de séquence de la prochaine trame I à transmettre, elle est incrémenté après chaque émission d'une trame I
  - Dans la partie réception : N(R) indique le numéro de séquence de la prochaine trame I attendue en réception, N(R) est incrémenté après chaque réception d'une trame I reçue sans erreur

# Différents type de trame de Supervision

## Supervision et contrôle de flux

### ***RR : Ready to Receive***

- ➔ Indique qu'on est prêt à recevoir une trame I
- ➔ Accuse réception des trames I reçues  $\leq N(R) - 1$
- ➔ Indique la sortie d'un état occupé

### ***RNR : Receiver Not Ready***

- ➔ Indique qu'on entre dans un état occupé
- ➔ Accuse réception des trames I reçues  $\leq N(R) - 1$

**Supervision et reprise sur erreurs** : Une trame erronée est toujours ignorée, la reprise sur erreur se fait lors d'un déséquencement ie  $N(S) > N(R)$  par:

### ***REJ : REject***

- ➔ Demande de retransmission de toutes les trames émises numérotées à partir de  $N(R)$
- ➔ Accuse réception des trames de numéro  $\leq N(R) - 1$

### ***SREJ : Selective REject***

- ➔ Demande de retransmission de la trame de numéro  $N(R)$
- ➔ Accuse réception des trames de numéro  $\leq N(R) - 1$

# Elément binaire P/F (Poll/Final)

Le bit P/F se trouvant dans le champ de contrôle d'une trame permet :

- Mode normal de réponse : primaire/secondaire
  - Le primaire positionne P à 1 sur ses trames lorsqu'il autorise le secondaire à émettre
  - Le secondaire émet alors ses trames et sur sa dernière trame, il positionne F à 1 pour indiquer au primaire qu'il a fini
  - Partage de la liaison en mode half-duplex
- Mode de réponse asynchrone
  - Lorsqu'une station reçoit une trame avec P à 1, elle répond avec F à 1 (acquittement de P)
  - Mécanisme servant de reprise sur erreur (REJ)

## Exemple : Protocole LAP-B (

### X.25.2)

- Protocole de niveau liaison ETTD/ETCD
- Mode équilibré : primaire/primaire
- 2 modes de fonctionnement : base modulo 8 ou étendu modulo 128
- Utilisation du bit P/F
- Reprise par REJ

# Principales trame de commande et de réponse

format	symbole	cmd	rép	champ de cmd	fonction
S	RR	X	X	1 0 0 0 P/F N(R)	<i>Ready to Receive</i> acquitte toutes les trames reçues de N(S) < N(R) et sert à la régulation de flux (sortie de l'état d'occupation)
	RNR	X	X	1 0 1 0 P/F N(R)	<i>Not Ready to Receive</i> acquitte toutes les trames reçues de N(S) < N(R) et sert à la régulation de flux (état d'occupation temporaire)
	REJ	X	X	1 0 0 1 P/F N(R)	<i>Reject</i> acquitte toutes les trames reçues de N(S) < N(R) et rejette toutes les trames à partir de N(R)
	SREJ	X	X	1 0 1 1 P/F N(R)	<i>Selective Reject</i> acquitte toutes les trames reçues de N(S) < N(R) et demande la retransmission de la trame de N(S) = N(R)
U	SARM	X		1 1 1 1 P 0 0 0	<i>Set ARM</i> demande l'établissement en mode ARM
	SNRM	X		1 1 0 0 P 0 0 1	<i>Set NRM</i> demande l'établissement en mode NRM
	SABM	X		1 1 1 1 P 1 0 0	<i>Set ABM</i> demande l'établissement en mode ABM
	SARME	X		1 1 1 1 P 0 1 0	<i>Set ARME</i> demande l'établissement en mode ARM étendu (mod 128)
	SNRME	X		1 1 1 1 P 0 1 1	<i>Set NRME</i> demande l'établissement en mode NRM étendu (mod 128)
	SABME	X		1 1 1 1 P 1 1 0	<i>Set ABME</i> demande l'établissement en mode ABM étendu (mod 128)
	DISC	X		1 1 0 0 P 0 1 0	<i>Disconnect</i> libère la liaison
	UA		X	1 1 0 0 F 1 1 0	<i>Unnumbered Acknowledge</i> indique la réception et l'acceptation d'une commande non numérotée
	CMDR FRMR		X	1 1 1 0 F 0 0 1	<i>Command (ARM, NRM) / Frame (ABM) Reject:</i> indique la réception d'une trame incorrecte et que la reprise ne peut s'effectuer par retransmission (avec la cause du rejet)
	DM		X	1 1 1 1 F 0 0 0	<i>Disconnect Mode</i> indique que la station se trouve en mode déconnecté

# SLIP (Serial Line Internet Protocol)

## Protocol)

- Principe
  - Ajout d'un octet en fin de paquet IP et utilise le byte stuffing. Il peut y avoir un octet au début ou fin de paquet IP
- Inconvénients
  - Ne fonctionne qu'avec IP
  - Pour les adresses fixes connues à l'avance
  - Pas de détection ou correction d'erreurs
  - Pas d'authentification de l'émetteur
  - Trop de versions existent il n'y a donc pas de standard de l'Internet

# PPP (Point to Point Protocol)

## ○ Principe

- Un format de trame défini permet la détection du début et de la fin de trame
- Un protocole de contrôle du lien LCP (Link Control Protocol) pour établir, tester et libérer le lien
- Un protocole NCP (Network Control Protocol) gère l'allocation des adresses IP

## ○ Avantages

- Fonctionne avec plusieurs protocoles réseaux, pour des adresses IP pas forcément connus à l'avance.  
Utilisation de mécanisme de détection d'erreurs et d'authentification de l'émetteur

# Couche Physique

<u>Protocol Implementation</u>							<u>OSI</u>
File Transfer	Electronic Mail	Terminal Emulation	File Transfer	Client Server	Network Mgmt	Application	
File Transfer Protocol (FTP) RFC 559	Simple Mail Transfer Protocol (SMTP) RFC 821	TELNET Protocol RFC 854	Trivial File Transfer Protocol (TFTP) RFC 783	Network File System Protocol (NFS) RFC 1024, 1057 and 1094	Simple Network Management Protocol (SNMP) RFC 1157	Presentation	
							Session
Transmission Control Protocol (TCP) RFC 793			User Datagram Protocol (UDP) RFC 768			Transport	
Address Resolution Protocols ARP: RFC 826 RARP: RFC 903	Internet Protocol (IP) RFC 791	Internet Group Management Protocol (IGMP) RFC 2236		Internet Control Message Protocol (ICMP) RFC 792		Network	
Ethernet	Token Ring	Starlan	Arcnet	FDDI	SMDS	Data Link	
Transmission Mode							Physical
TP	STP	FO	Satellite	Microwave, etc			

# Transmission de l'information

- Les informations sont représentées par des signaux électriques sur le support physique. Le support physique est constitué de câbles de différents types:
  - Mécanique
  - Coaxial
  - Fibre optique
  - Satellite

Le choix du support physique est influencé par les performances attendues du système à réaliser :

- Débit attendu
- Bande passante nécessaire
- Coût
- Le taux d'erreurs toléré
- Distance maximale
- Possibilité d'avoir des voies de secours en cas de panne

# Les supports physiques

## • La paire torsadée

- constituée d'une paire de fils électriques agencés en spirale
  - convient à la transmission analogique et numérique
  - adaptée à la transmission d'information de courte distance
  - support le plus utilisé : téléphonie
- 
- Performances :
    - Débits courants : 1 Mbit/s, 4 Mbit/s, 10 Mbit/s, 16 Mbit/s
    - Portée sans régénération : 100 à 250 m

# Support de transmission

- **Le cable coaxial**

- constitué de deux conducteurs cylindriques de même axe, séparés par un isolant
- convient à la transmission numérique et analogique
- adaptée à la transmission d'information de longue distance
- Les cables coaxiaux offerts sur le marché sont (cable 50 du type Ethernet, le cable de 75 du type CATV : Community Antenna Television)
- 
- Performances :
  - Débits courants : 2 Mbit/s à 100 M bit/s
  - Portée sans régénération : 3 Km, 4,5 Km
- 
- Coût : plus élevé
- 
-

# Support de transmission

## La fibre optique

- fibre de silicium (ou plastique)
- utilise un faisceau optique modulé
- permet une très large bande passante (de l'ordre de 1GHz pour 1 Km)
- permet une très bonne qualité de la transmission

Performances :

- Débit courant : qq Gbits/s par Km
- Portée sans régénération : 15 Km ( $L : 0.85 \mu m$ ), 500 Km ( $L : 1,3 \mu m$ )

- **Transmission terrestre sans support matériel**

- > Courte distance
  - Rayons infrarouges et rayons laser
  - Transmission numérique
- > Longue distance
  - Faisceaux hertziens, ondes radios magnétiques
  - Problèmes des interférences
  - Partage de la bande passante admissible (2-40 GHz)
  - Application : réseaux de radio-télévision
- 
- 
-

## Les satellites de communication

- Ondes à très hautes fréquences
- Répéteurs à transposition de fréquences
- Satellites géostationnaires (36000 Km par rapport à l'équateur)
- Bandes de fréquences utilisables (4/6 GHz, 12/14 GHz)
- Faible taux d'erreurs
- Problèmes de confidentialité (cryptographie)

- Support de transmission n'est pas parfait à cause :
  - 
  - Distorsion de phase et affaiblissement
  - Bruits :
    - ➔ bruits impulsifs : interférence entre signaux, diaphonie (transfert d'une partie de l'énergie du signal transmis d'un circuit à un autre)
    - ➔ bruit gaussien : phénomène thermique à l'intérieur des conducteurs (ou bruit blanc)

Signal reçu  $\neq$  signal émis -> Filtre

# Phénomènes perturbateurs

- Bruit blanc :
  - agitation thermique,
  - de faible puissance,
  - sur une large plage de fréquences.
- Bruit impulsif :
  - organes électromécaniques,
  - micro-coupures,
  - forte puissance,
  - durée faible,
  - peu présent dans les réseaux numériques.
- Diaphonie :
  - couplage parasite entre lignes voisines – influence électromagnétique
  - placement des câbles, blindage, fibre optique !
- Echo :
  - Réflexion du signal due à une désadaptation d'impédance
  - suppresseur d'écho.
  - Ex : câblage téléphonique 4 fils/2 fils

# Filtrage linéaire

**Définition** : Un filtrage linéaire est une opération qui transforme un signal d'entrée  $x(t)$  en un signal de sortie  $y(t)$ . Cette transformation est linéaire et homogène dans le temps

$$\begin{aligned}x(t) &\longrightarrow y(t) \\ax_1(t) + bx_2(t) &\longrightarrow ay_1(t) + by_2(t) \\x(t-t_0) &\longrightarrow y(t-t_0)\end{aligned}$$

On caractérise le filtre par sa réponse à certaines fonctions « types »

$$\begin{aligned}\delta(t) &\longrightarrow R(t) \text{ impulsion de Dirac} \longrightarrow \text{Réponse impulsionnelle} \\e^{-2i\pi f t} &\longrightarrow G(f) e^{-2i\pi f t}\end{aligned}$$

fonctions propres  $\longrightarrow$  fonction de transfert du filtre

$G(f)$  : facteur d'amplification ou d'affaiblissement, gain du filtre ou fonction de transfert

# Bande Passante

- La bande passante d'un support de communication correspond à la plage de fréquences où il présente les meilleures caractéristiques de transmission.
  - où le gain est non nul ! (gain = 1/affaiblissement)
  - malheureusement, en général, le gain n'est jamais tout-à-fait nul!
- La bande passante à n décibels (dB) est la plage de fréquences dans laquelle le rapport S/B (appelé le rapport signal sur bruit) vérifie :
  - $S/B = 10 \log_{10} (P_S / P_B) \leq n \text{ dB}$ ,
  - où  $P_S$  est la puissance du signal et  $P_B$  est la puissance du bruit.

•

S/B	$\frac{1}{4}$	$\frac{1}{2}$	1	2	4	10	100	1000
$10\log_{10}(S/B)$	-6	-3	0	3	6	10	20	30

# Bande Passante

- On définit également la bande passante comme la plage de fréquences telle que
- $|G(f)|^2$  reste supérieur à une valeur donnée.
  
- 
- $|G(f)|^2 = \frac{P_R}{P_E} = \frac{\text{(Puissance reçue en } f\text{)}}{\text{(Puissance émise en } f\text{)}}$

La bande passante à  $m$  décibels est la plage de fréquences telles que :

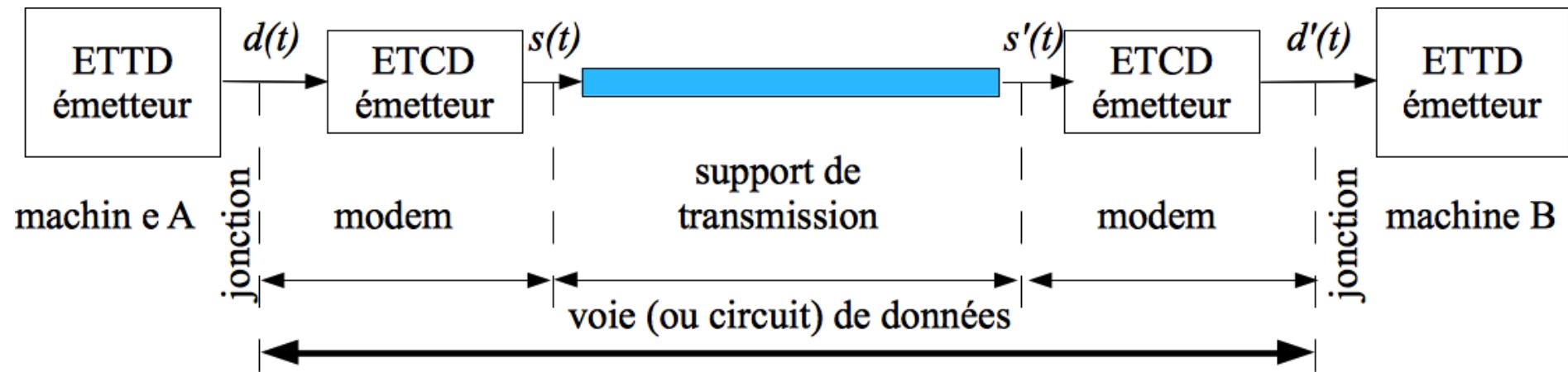
$$m = 10 \log_{10} \frac{P_E}{P_R} = 10 \log_{10} \frac{1}{|G(f)|^2}$$

# Débit

- Le débit théorique maximum d'un support est donné par :
- La formule de Nyquist [1942] d'un support sans bruit :  $D_{max} = 2H \log_2 V \text{ bit/s}$ ,
- $H$  bande passante
- $V$  niveaux significatifs
- Exemple : le débit permis sur une ligne exempte de bruit dont la bande passante égale à 3000 Hz est, dans le cas d'un signal à deux niveaux significatifs sur la ligne, limité à  $D = 6000 \text{ bit/s}$
- La formule de Shannon [1948] d'un support soumis à du bruit :  $D = W \cdot \log_2(1 + Ps/Pb) \text{ bit/s}$
- $W$ , exprimé en Hertz (Hz), représente la bande passante du support,
- $Ps$  la puissance du signal et  $Pb$  la puissance du bruit, est obtenu à l'aide du rapport signal sur bruit exprimé en décibel.
- Exemple : le débit maximum théorique d'une ligne téléphonique [300 à 3400 Hz] admettant un rapport signal sur bruit de 30 dB est de :  $(3400-300) \cdot \log_2(1 + 10^{30/10}) = +/- 30 \text{ Kbit/s}$

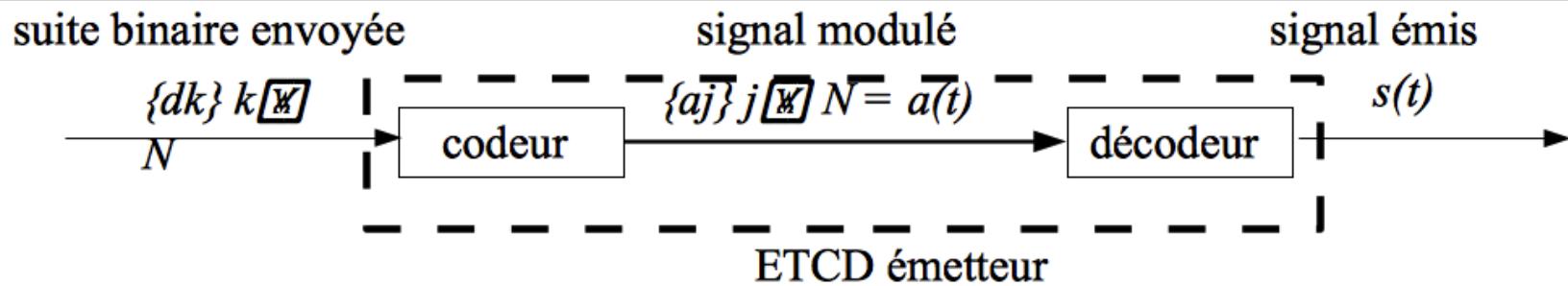
# Éléments intervenant dans la transmission

- *Les principaux éléments*
- L'ETCD (Equipement Terminal de Communication de Données)
- équipement spécifique chargé d'adapter les données à transmettre au support de communication
- L'ETTD (Equipement Terminal de Traitement de Données)
- l'ordinateur !
- *Le support de transmission*

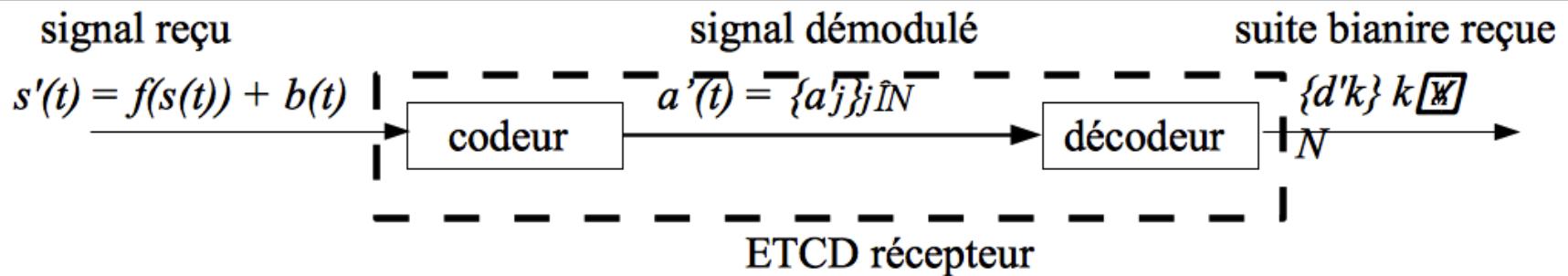


# Fonctions de l'ETCD

- Deux transformations fondamentales sont définies :
- Le codage : bits  $\rightarrow$  symboles
- la modulation : symboles  $\rightarrow$  signal
- les symboles peuvent être une fonction continue ou une suite de valeurs
- la transformation appliquée peut être très simple (pour le codage en BdB)
- A l'émission



A la réception



# Modulation

- Le modulateur transforme un signal initial quelconque  $a(t)$  en un signal  $s(t)$  adapté au support de communication employé
- Le signal  $s(t)$  est obtenu en faisant varier les paramètres d'une onde généralement sinusoïdale
  - ➔  $s(t) = A \cos (2 \pi f t - F)$  :
    - ✓ le signal sinusoïdal est centré autour d'une fréquence  $f_0$  appelée onde de référence ou porteuse
- Trois types élémentaires de modulation par transposition en fréquence:
  - ➔ modulation d'amplitude (lorsque les variations portent sur  $A$ )
  - ➔ modulation de fréquence (lorsque les variations portent sur  $f$ )
  - ➔ modulation de phase (lorsque les variations portent sur  $F$ )
- La transposition en fréquence autorise le multiplexage temporel

# Modulation

- Lorsque la fonction de modulation, existe la transmission est dite analogique. Dans ce cas, on a une transformation d'une fonction continue en une autre fonction continue.
- La transmission est dite en bande de base lorsque le signal ne subit pas (peu) de transposition en fréquence. Dans ce cas, le signal présente souvent un aspect rectangulaire car la fonction de modulation simple utilisée est rectangulaire.
- On peut transformer une fonction discrète  $\{d_k\}$  en fonction continue  $d(t)$  à l'aide de la relation suivante :

$$d(t) = \sum_{k=-N}^{+N} d_k * R_T(t - kT - t_0) / k = +N, k = -N$$

$t_0$  étant l'instant initial, la rapidité de modulation étant  $1/T$  et  $R_T(t)$  étant la fonction rectangulaire sur l'intervalle  $[0, T]$  définit ainsi :

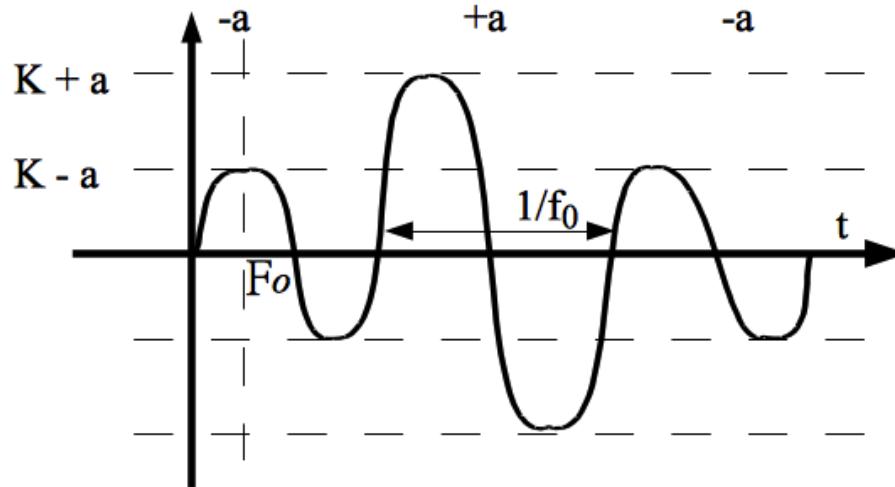
$$\begin{aligned} R(t) &= 1, t \in [0, T] \\ &0, t \notin [0, T] \end{aligned}$$

- Par abus de langage, on parle de transmission numérique lorsque une fonction discrète (suite binaire) est transformée en fonction continue lors de l'émission et réciproquement lors de la réception

# Modulation d'amplitude

Signal :  $s(t) = A(t) \cos(2\pi f_0 t - F_0)$

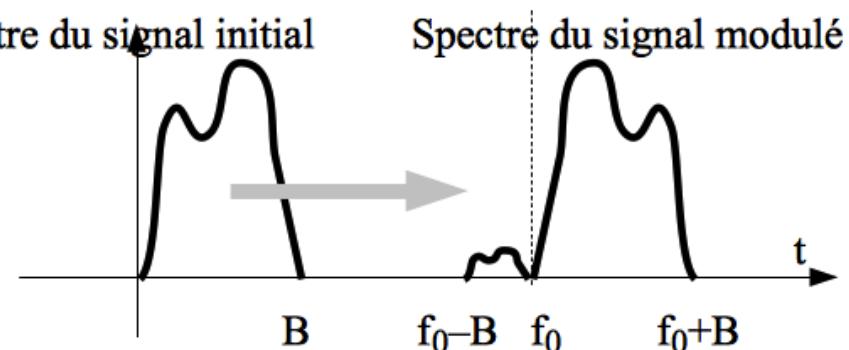
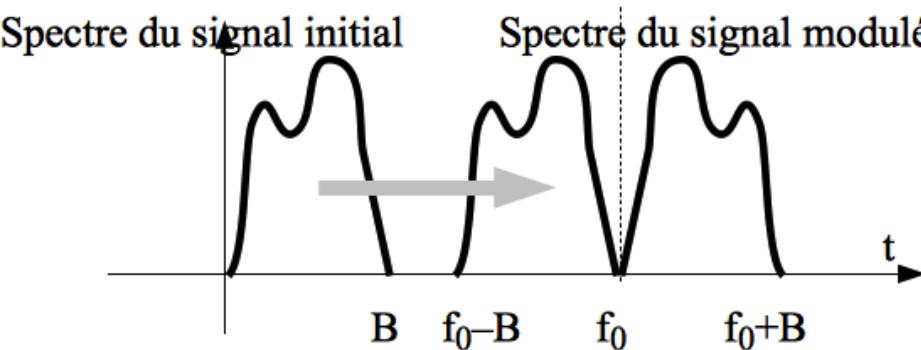
avec  $A(t) = K + a(t)$  et  $a(t) \in \{-a, +a\} \dots$  ou  $a(t) \in [-a, +a]$  !



Technique électroniquement simple mais sensible au bruit.

Modulation en double bande

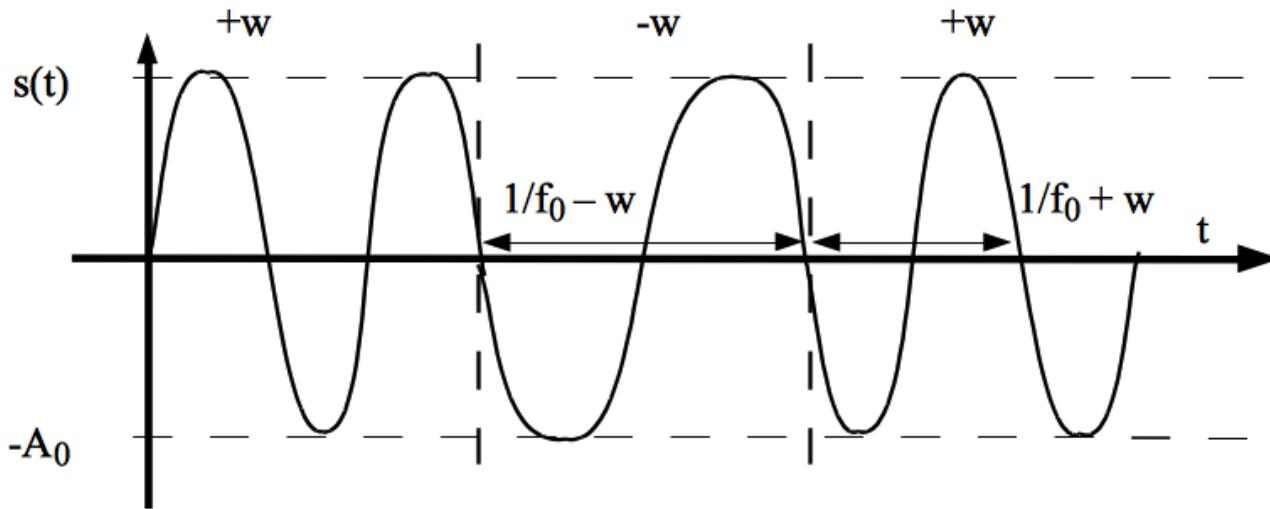
BLU : Bande latérale unique



# Modulation de Fréquence

Signal :  $s(t) = A_0 \cos (2 \pi f(t) t - F_0)$

avec  $f(t) = f_0 + a(t)$  et  $a(t) \in \{-w, +w\} \dots$  ou  $a(t) \in [-w, +w]$  !



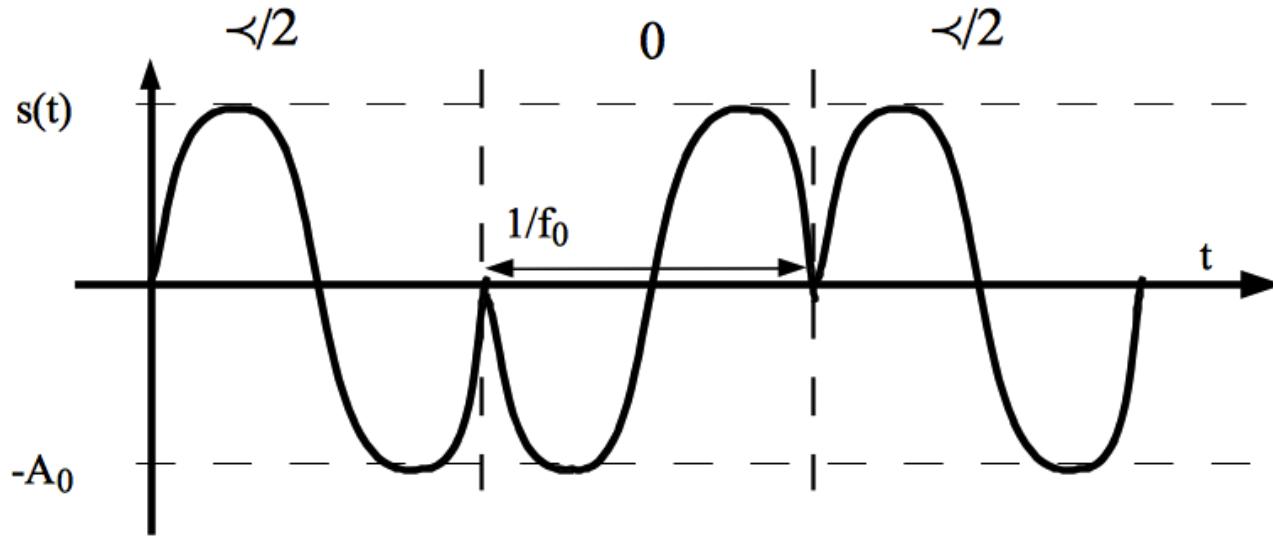
- Difficulté à maintenir la phase
- Utilisée par la technique de multiplexage fréquentiel

# Modulation de phase et complexe

## Modulation en phase

Signal :  $s(t) = A_0 \cos (2 \pi f_0 t - F(t))$

avec  $F(t) = F_0 + a(t)$  et  $a(t) \in \{P k / n\}$  pour  $n$  symboles ... ou  $a(t) \in [-P, +P]$  !



## Modulation complexe

Modulation en quadrature (MAQ)

- modulation en phase et en amplitude
- Par exemple :

# Codage

Le **codeur** transforme une suite  $\{d_k\}_{k=0}^n$  initiale généralement binaire (de bits) en une suite codée  $\{a_k\}_{k=0}^n$  (de symboles) généralement binaire ou ternaire

Le **décodeur** fait l'opération inverse.

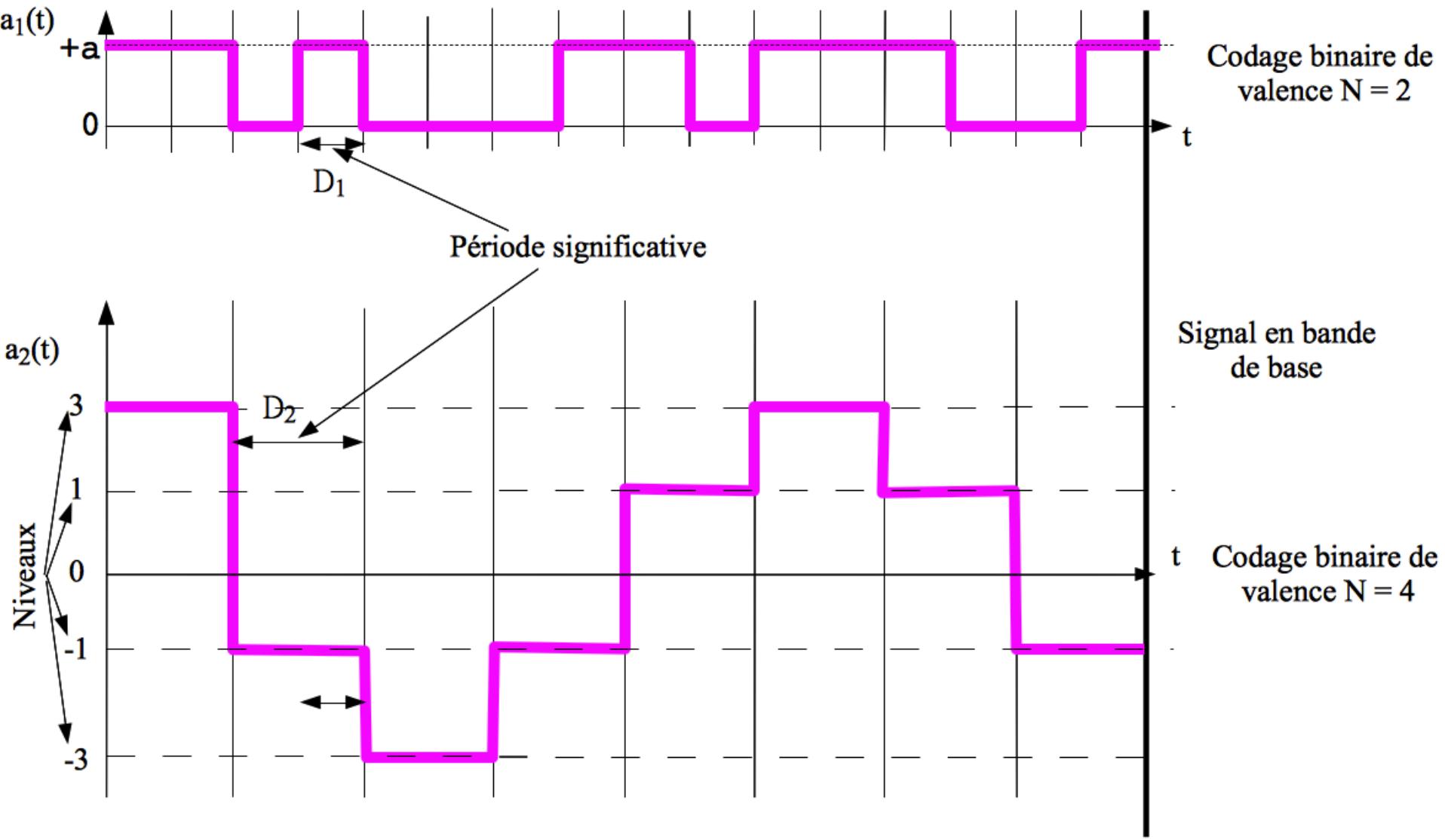
Le but du codage est d'adapter la suite de bits à transmettre aux caractéristiques de la transmission

S'il n'y a pas de modulation par transposition en fréquence, le codage est dit en bande de base :

- la plage de fréquences utilisée par le signal issu de la suite codée est la même que celle de la suite initiale.
- dans ce cas, le modulateur module à partir d'une fonction rectangulaire.
  - ✓  $\{a_k\}_{k=0}^n \Rightarrow a(t)$

# Exemples

$\{dk\} = (1 \cdots 1 \cdots 0 \cdots 1 \cdots 0 \cdots 0 \cdots 1 \cdots 1 \cdots 0 \cdots 1 \cdots 1 \cdots 1 \cdots 0 \cdots 0 \cdots 1)$  suite binaire initiale



## Débit binaire et rapidité de modulation

- Le **débit binaire**  $D$  d'une voie de données est le nombre maximum de bits  $d_i$  transmis par seconde sur cette voie.

$$D = 1/T \text{ bits/s}$$

- La **rapidité de modulation**  $R$  (exprimée en bauds) mesure le nombre maximum de symboles (éléments de modulation) transmis par seconde

$$R = 1/D \text{ bauds}$$

- Remarque* : Généralement,  $1/D$  est un multiple de  $1/T$  et le nombre de niveaux  $N$  est choisi de telle sorte que  $a(t)$  et  $d(t)$  aient le même débit d'information. On a alors :

$$D = \frac{1}{T} = \frac{(\log_2(N))}{\Delta} = R \log_2(N) \text{ bits/s}$$

# Principales qualités d'un code 1/2

- largeur de sa plage de fréquences
  - ➔ la plus étroite possible
- répartition fréquentielle de la puissance
  - ➔ peu de puissance aux faibles fréquences, aucune à la fréquence nulle
- codage de l'horloge
  - ➔ fréquence suffisante des transitions
  - ➔ synchronisation de l'horloge du récepteur sur le signal reçu
- résistance au bruit
  - ➔ espacement des niveaux
- complexité du codage
  - ➔ coût et vitesse de codage

# Principales qualités d'un code 2/2

- dépendance à la polarité
  - ➔ Facilité d'installation
- équilibrage
  - ➔ mesure approximative de l'influence du codage sur des symboles successifs
  - ➔ Running Digital Sequence :  $RDS(\{a_k\}) = \sum a_k$
  - ➔ DRDS( $\{a_k\}$ ) = max (abs{RDS( $\{a_j\}$ ) / { $a_j$ } sous suite valide de  $\{a_k\}$ })

# Codes usuels en bande de base

- **Les codes à deux niveaux :**
  - ➔ code **NRZ** (Non Return to Zero)
  - ➔ code **NRZI** (Non Return to Zero Invert)
  - ➔ code **biphase**
  - ➔ code **biphase différentiel**
  - ➔ code **de Miller**
- **Les codes à trois niveaux :**
  - ➔ code **RZ** (Return to Zero)
  - ➔ code **bipolaire** (simple)
  - ➔ code **bipolaire entrelacé d'ordre 2**
  - ➔ codes **bipolaires à haute densité d'ordre n** (BHDn)
- **Les codes par blocs :**
  - ➔ code **nB/mB**

# Code NRZ (Non Return to Zero)

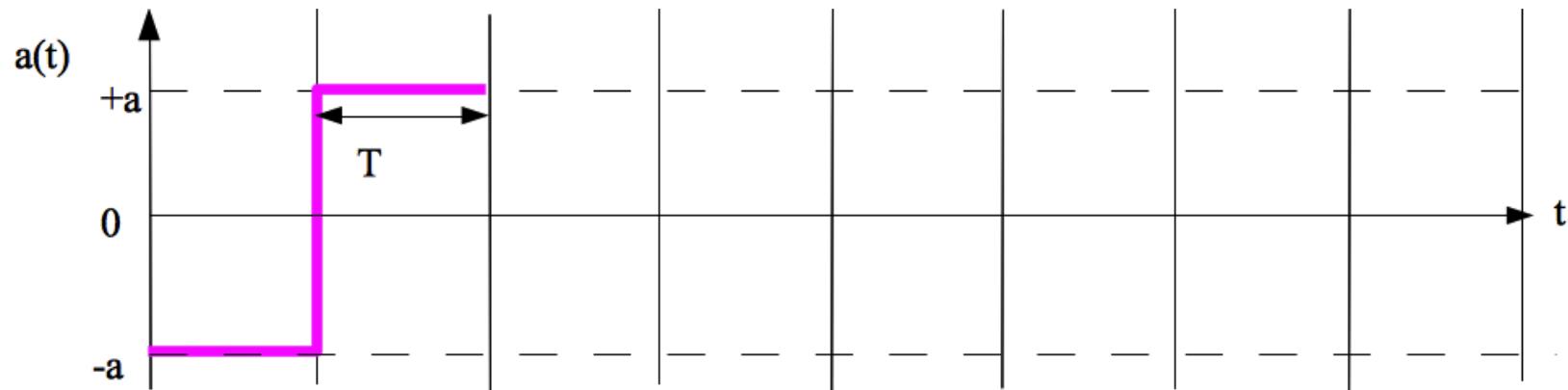
Code NRZ

$(d_k = 0) \rightarrow (a_k = [a])$

$(d_k = 1) \rightarrow (a_k = [-a])$

*Exemple :*

$\{d_k\} = (1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1)$  suite binaire initiale



$RDS(\{a_k, k < i\}) = -a, -2a, -a, -2a, -a, 0, +a, 0, -a, 0, -a, -2a, -3a, \dots$

Code simple, utilisé couramment entre l'ordinateur et ses périphériques.

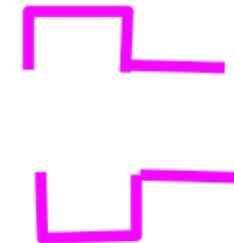
$DRDS(NRZ) = \mathbb{N}$

Spectre :  $X(f) = T (a \sin(Pft) / Pft)^2$

# Code RZ (Return to Zero)

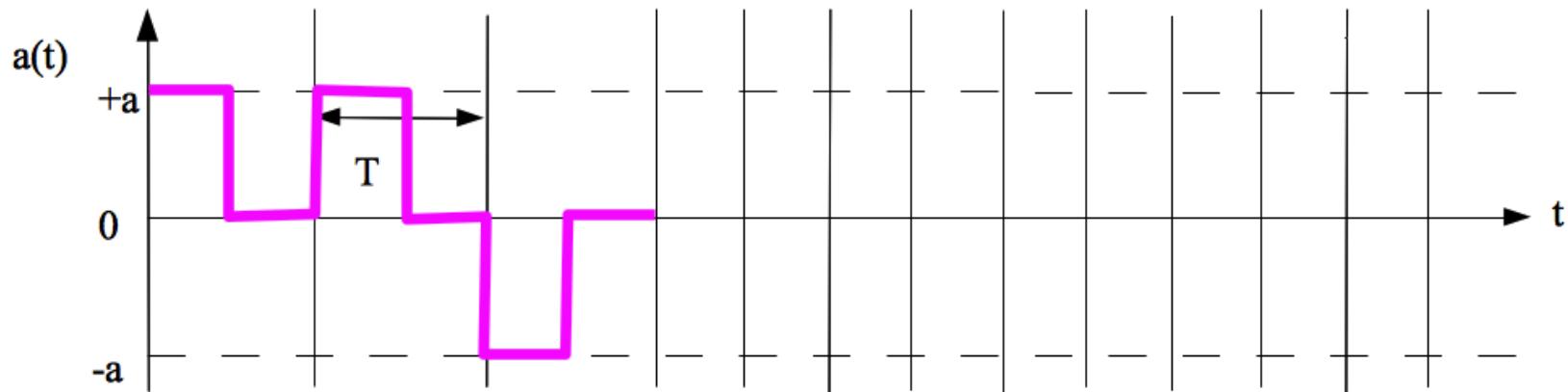
Code RZ

$$\left\{ \begin{array}{l} (d_k = 0) \Leftrightarrow (a_k = [-a, 0]) \\ (d_k = 1) \Leftrightarrow (a_k = [+a, 0]) \end{array} \right.$$



*Exemple :*

$\{d_k\} = (1 \cdots 1 \cdots 0 \cdots 1 \cdots 0 \cdots 0 \cdots 1 \cdots 1 \cdots 0 \cdots 1 \cdots 1 \cdots 1 \cdots 0 \cdots 0 \cdots 1)$  suite binaire initiale



Code ternaire simple, limite les interférences entre symboles, codage de l'horloge

DRDS(RZ) =  $\mathbb{N}/2$  !

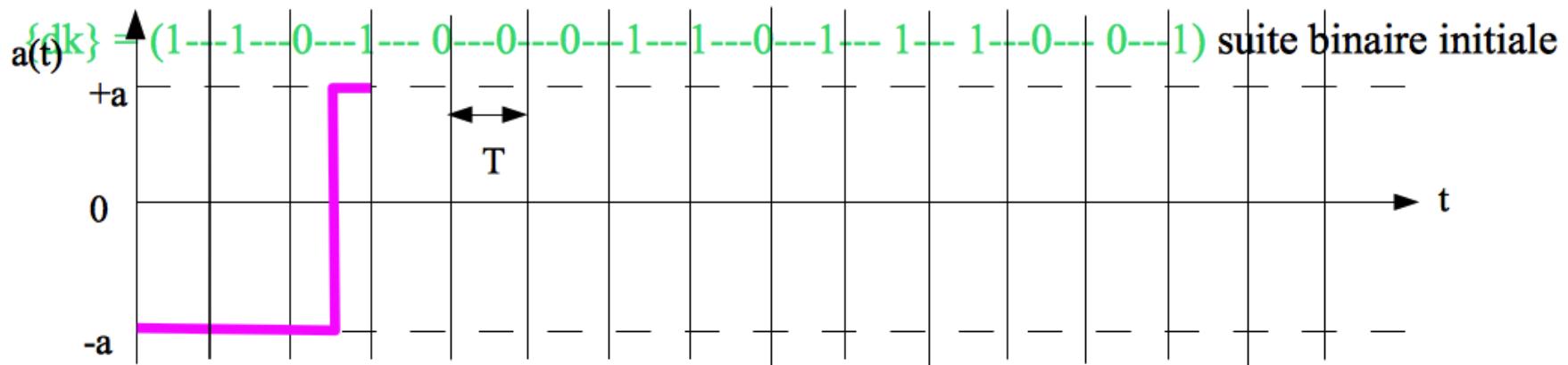
codage 1B/2T

# Code NRZI (Non Return to Zero Invert)

$$\left\{ \begin{array}{l} (d_k = 0) \Leftrightarrow (a_k = [\diamond_k, \text{zigzag}_k]) / ((\diamond_k \Leftrightarrow \text{zigzag}_k) \Leftrightarrow (\diamond_k = \text{zigzag}_{k-1})) \\ \text{ou} \\ (d_k = 1) \Leftrightarrow (a_k = [\diamond_k, \text{zigzag}_k]) / ((\diamond_k = \text{zigzag}_k) \Leftrightarrow (\diamond_k = \text{zigzag}_{k-1})) \end{array} \right. \quad \boxed{0} \quad \boxed{1}$$

$a_k \in \{a, -a\}, (\text{zigzag}_0 = -a)$

Exemple :



Code binaire, indépendant de la polarité, adapté à la transmission photonique

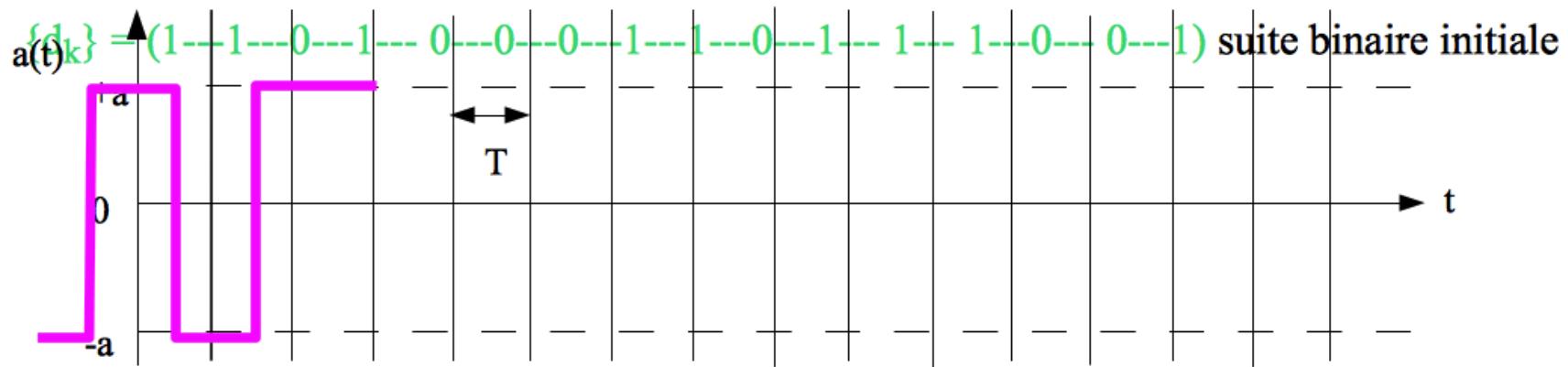
# Code Miller

$$\left\{ \begin{array}{l} (d_k = 0) \Leftrightarrow (a_k = [\diamond_k, \circlearrowleft_k]) / ((\diamond_k = \circlearrowright_k) \Leftrightarrow (\diamond_k \vee a_{k-1})) 0 \\ \text{ou} \\ (d_k = 1) \Leftrightarrow (a_k = [\diamond_k, \circlearrowright_k]) / ((\diamond_k \vee \circlearrowleft_k) \Leftrightarrow (\diamond_k = \circlearrowleft_{k-1})) 1 \end{array} \right.$$

$$\{a_k \in \{a, -a\}, (a_0 = -a), (\circlearrowleft_0 = a)\}$$

Autres dénominations : “modified FM”, DM : “Delay modulation”

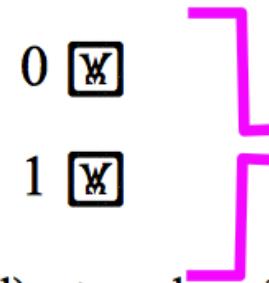
*Exemple :*



Code binaire dense, conservation de l'horloge et indépendance de la polarité

# Code biphase

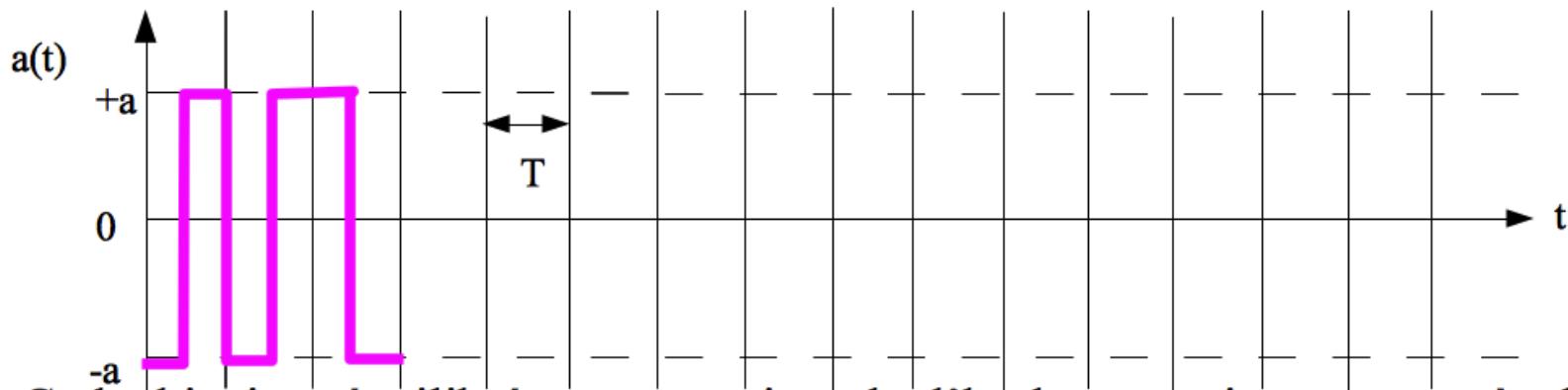
$$\left\{ \begin{array}{l} (d_k = 0) \Leftrightarrow (a_k = [a, -a]) \\ (d_k = 1) \Leftrightarrow (a_k = [-a, a]) \end{array} \right.$$



Autres dénominations : Manchester, biphase\_L(evel), => codage 1B/2B.

*Exemple :*

$\{d_k\} = (1---1---0---1---0---0---0---1---1---0---1---1---1---0---0---1)$  suite binaire initiale



Code binaire, équilibré, conservation de l'horloge, mais spectre très large (le double). Codage utilisé par Ethernet.

DRDS(biphase) = 0 !

$$\text{Spectre : } X(f) = T * ((2a) / Pft)^2 * \sin(Pft / 2)^4$$

# Code biphasé différentiel

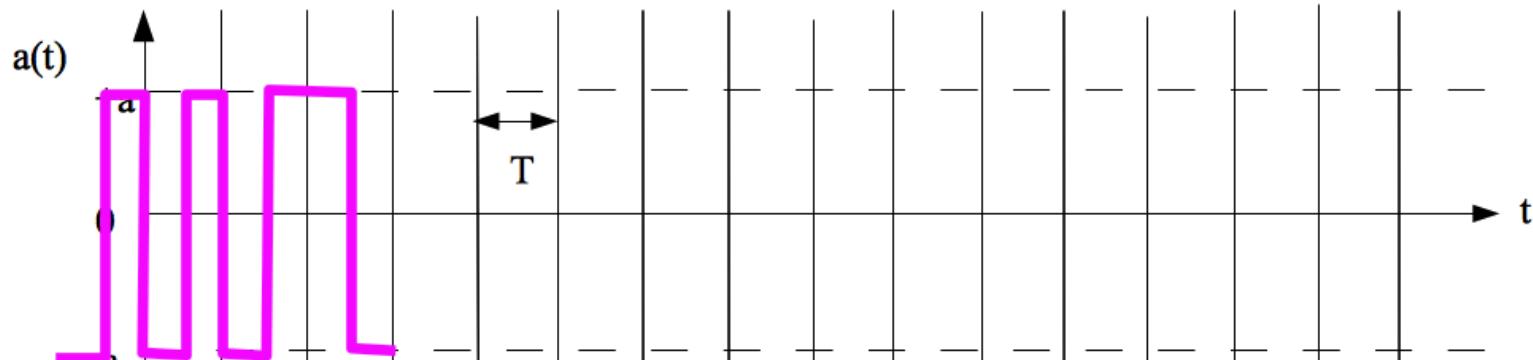
$$\left\{ \begin{array}{l} (d_k = 0) \Leftrightarrow (a_k = [\diamond_k, \circlearrowleft_k]) / ((\diamond_k \Leftrightarrow \circlearrowleft_k) \Leftrightarrow (\diamond_k \Leftrightarrow a_{k-1})) \\ (d_k = 1) \Leftrightarrow (a_k = [\diamond_k, \circlearrowright_k]) / ((\diamond_k \Leftrightarrow \circlearrowright_k) \Leftrightarrow (\diamond_k = a_{k-1})) \end{array} \right. \quad \begin{array}{l} 0 \\ 1 \end{array} \quad \text{ou} \quad \begin{array}{l} 1 \\ 0 \end{array} \quad \text{ou}$$

$\{a_k \in \{a, -a\}, (a_0 = -a), (\circlearrowleft_0 = a)\}$

Autres dénominations : Manchester différentiel, FSK : “frequency shift keying”, FM : “frequency modulation”, biphasé\_M(ark) ou biphasé\_S(pace)

*Exemple :*

$\{d_k\} = (1---1---0---1---0---0---0---1---1---0---1---1---1---0---0---1)$  suite binaire initiale



Identique au code Manchester + indépendance de la polarité

Problème s'il y a corruption d'un des symboles : la suite est mal décodée

DRDS(biphasé\_diff) = 0

Codage utilisé par Token Ring.

# Code bipolaire simple

Notation :  $d^1_j$  le  $j^{\text{ème}}$  bit de la sous-suite des bits à 1

$(d_k = 0) \rightarrow (a_k = 0)$

0

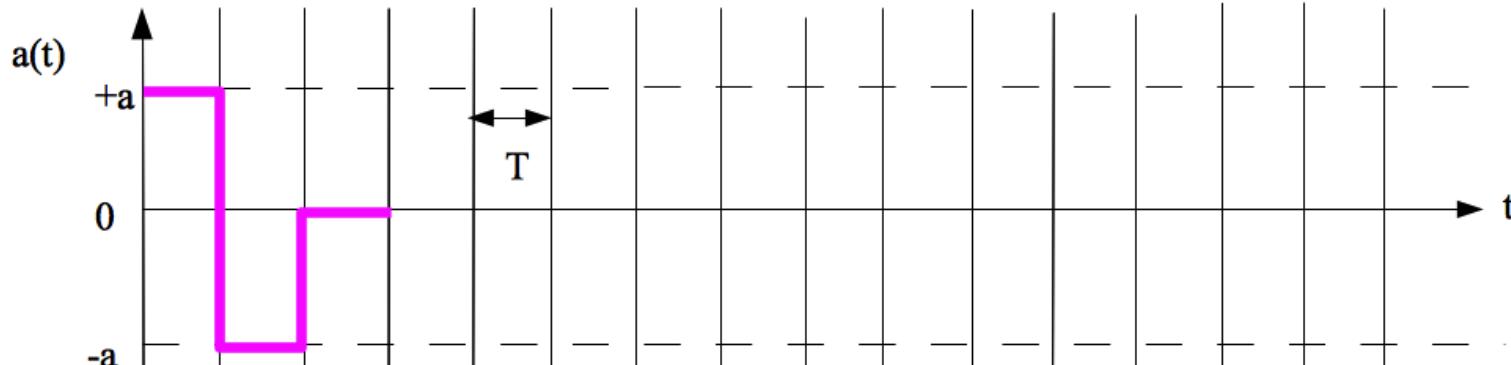
$(d_k = 1) \rightarrow d^1_m \rightarrow (a_k = [a]) / m = 2n + 1, (n \leq N)$  1 alternativement

{

Autres dénominations : AMI “Alternate Mark Inversion”

*Exemple :*

$\{d_k\} = (1---1---0---1---0---0---1---1---0---1---1---1---0---0---1)$  suite binaire initiale



Code ternaire, équilibré, indépendant de la polarité, dérive de l'horloge (suite de 0)

spectre :  $X(f) = T (a / (Pf))^2 \sin(Pft)^4$

DRDS(bipolaire) = a

Utilisé par le système de téléphonie numérique PCM sur la ligne de transmission T1.

# Code bipolaire entrelacé d'ordre 2

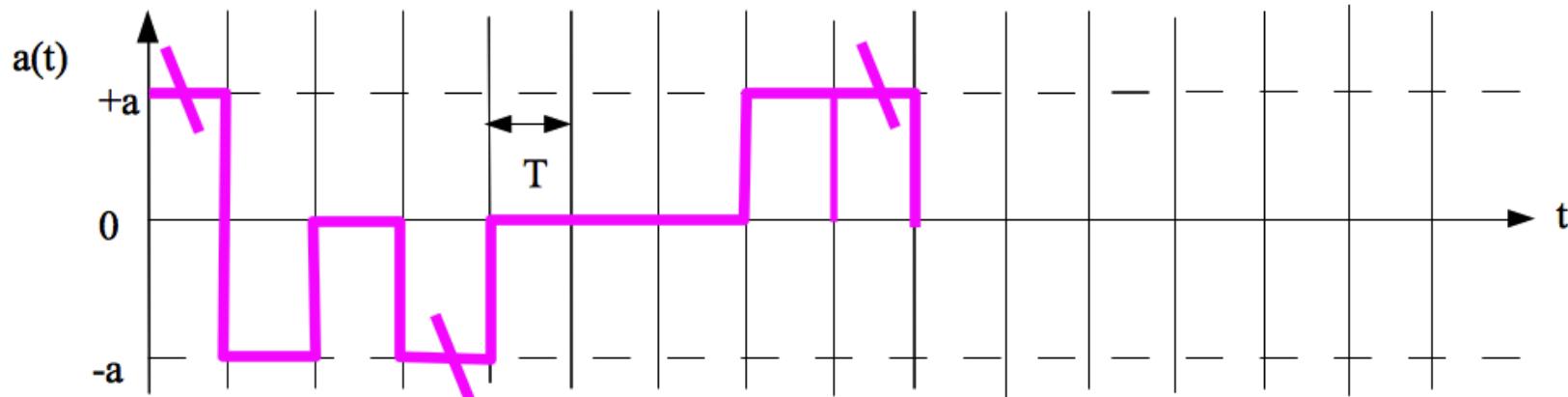
Construction de 2 sous-suites à partir de la sous-suite des bits à 1 :

- la sous-suite des 1 pairs et celle des 1 impairs.

Chaque sous-suite est indépendamment codée en alternance.

*Exemple :*

$\{d_k\} = (1 \text{---} 1 \text{---} 0 \text{---} 1 \text{---} 0 \text{---} 0 \text{---} 1 \text{---} 1 \text{---} 0 \text{---} 1 \text{---} 1 \text{---} 0 \text{---} 0 \text{---} 1)$  suite binaire initiale



Spectre très étroit, code complexe qui ne résout pas le problème lié aux longues suites de 0. Les longues suites de 1 présentent un battement dont la fréquence est réduite (de moitié) par rapport au codage bipolaire simple.

Généralisation possible aux codes bipolaires entrelacés d'ordre  $n$ .

# Code bipolaire haute densité d'ordre n (BHDn)

Même codage que le code bipolaire + une transformation des suites de plus de  $n$  zéros.

- basée sur la violation de l'alternance : **bit de viol** (noté V)

Une suite consécutive de  $n+1$  bits à 0 est codée par :

- (a) suite de  $n$  zéros suivis d'un bit de viol : [000...00] ® [000...0V]
- (b) suite formée d'un **bit de bourrage** (noté B),  $n-1$  zéros, suivis d'un bit de viol; les bits B et V ayant même polarité : [000...00] ® [B00...0V]

Pour assurer l'équilibrage :

- On choisit la forme (a) si le nombre de bits à 1 suivant le dernier bit de viol est impair, la forme (b) sinon.

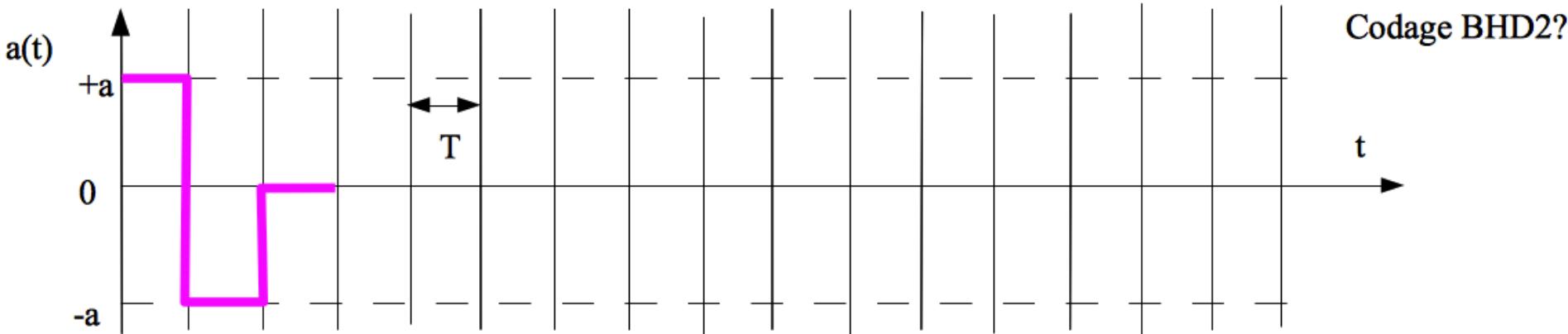
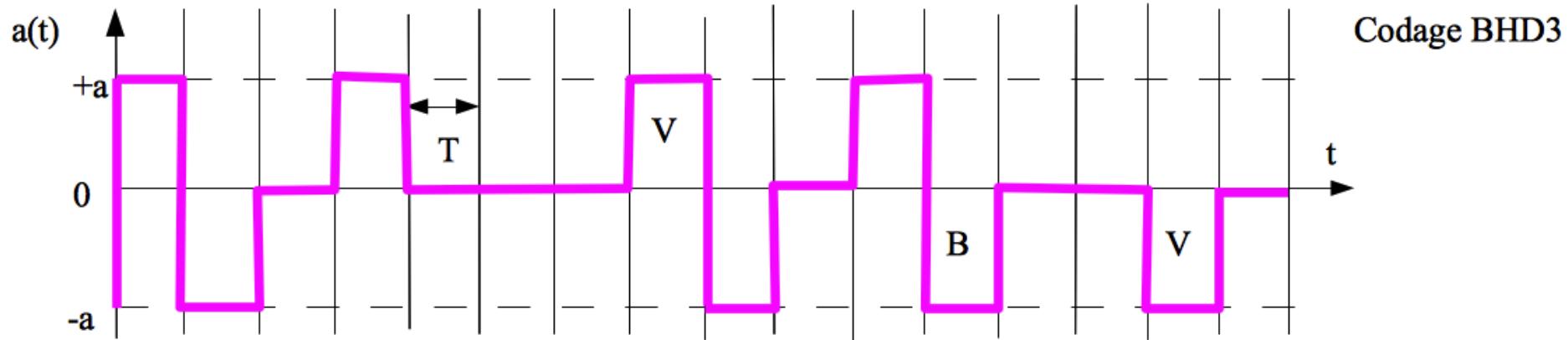
*Remarques :*

- le premier bit à un (suivant un bit de viol) est codé avec la valeur inverse du bit de viol qui le précède
- On considère que la suite est conventionnellement précédée d'un bit de viol.
- Dans une très longue suite de zéros tous les blocs successifs (sauf parfois le premier) sont codés dans la forme (b).

# Exemples code BHDn

*Exemple :*

$\{d_k\} = (1 \text{---} 1 \text{---} 0 \text{---} 1 \text{---} 0 \text{---} 0 \text{---} 0 \text{---} 1 \text{---} 0 \text{---} 1 \text{---} 0 \text{---} 0 \text{---} 0 \text{---} 0)$  suite binaire initiale



*Remarques :* La plupart de ces codes acceptent plusieurs variantes. Par exemple en inversant la convention de codage de la parité (0/1) ou en modifiant les conditions initiales.

# Codes par blocs

Code chaque bloc de  $k$  bits par un bloc de  $n$  symboles pris dans un alphabet de taille  $L$ . L'alphabet étant généralement binaire, ternaire, ou plus rarement quaternaire (noté resp. B, T, Q).

- on a la relation :  $2^k \leq L^n$
- notation :  $kB/nL$ , par exemple 4B/5B

0	1	2	3	4	5	...	F	Idle	J	K	R	S	Halt	Quiet
11110	01001	10100	10101	01010	01011	...	11101	11111	11000	10001	00111	11001	00100	00000

Tableau : quelques encodages de symboles FDDI

Certains codes précédents peuvent être perçus comme des codes par blocs (surtout si le bloc à coder est réduit à un seul bit).

- Exemple : RZ  $\hat{I}$  1B/2T, biphasé  $\hat{I}$  1B/2B

L'efficacité de ces codes peut être faible ( $2^k/L^n$ ).

Ces codes servent à éliminer les suites de symboles impropre à la transmission.

Lors de précodage :

- La modulation est généralement effectuée ultérieurement en utilisant un des codes simples précédents.
  - ✓ Exemple : FDDI = 4B/5B + NRZI

# Couche Physique : Protection contre les erreurs

# Contrôle et gestion

## Contrôle d'erreur

- Pb : comment s'assurer que le récepteur a reçu correctement et dans le bon ordre toutes les trames émises?
  - ➔ Informer l'émetteur de ce qui se passe : envoi d'une trame de contrôle (acquittement)
  - ➔ Perte d'une trame complète : utilisation d'un délai de garde
  - ➔ Plusieurs émissions de la même trame : numérotation des trames

## Contrôle de flux

- But : régulariser l'émission des trames sur la capacité du récepteur
- Ex: interdire à l'émetteur l'envoi de trames avant avoir reçu une permission du récepteur

## Gestion de la liaison

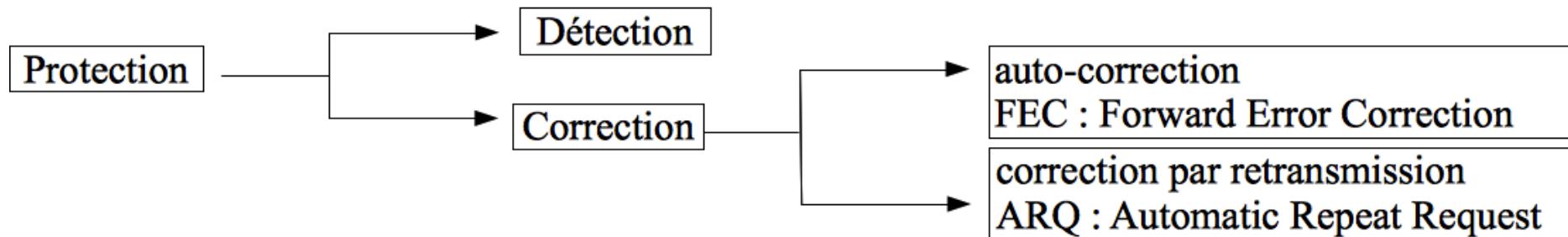
- Établir la connexion, initialiser des numéros de séquence, les ré-initialiser en cas d'erreur

# Erreurs

*Quelque soit la qualité des supports de communication et les performances des techniques de transmission utilisées, des perturbations vont se produire entraînant des erreurs sur les données transmises.*

Dans ces conditions, la suite binaire reçue ne sera pas identique à la suite émise. Il faut donc mettre en oeuvre :

- des techniques de protection contre les erreurs de transmission
- des stratégies de protection contre les erreurs de transmission



# Principe général de détection

Principe général pour la **détection** des erreurs de transmission

- un émetteur veut transmettre un message (suite binaire quelconque) à un récepteur
- l'émetteur transforme le message initial à l'aide d'un procédé de calcul spécifique qui génère une certaine redondance des informations au sein du message codé.
- le récepteur vérifie à l'aide du même procédé de calcul que le message reçu est bien le message envoyé grâce à ces redondances.

**Exemple** : la technique de détection par répétition

- le message codé est un double exemplaire du message initial, le récepteur sait qu'il y a eu erreur si les exemplaires ne sont pas identiques.

**Note** : certaines erreurs sont indétectables !

- ex. : une même erreur sur les deux exemplaires simultanément

# Principe général de correction

Principe général pour la **correction** des erreurs de transmission

- Après détection d'une erreur, la redondance est suffisante pour permettre de retrouver le message initial.

**Exemple** : la technique de correction par répétition

- le message codé est un triple exemplaire du message initial, le récepteur suppose que le message initial correspond aux deux exemplaires qui sont identiques.

**Note** : certaines erreurs détectées ne sont pas corrigibles !

- ex. : une erreur différente sur au moins deux exemplaires

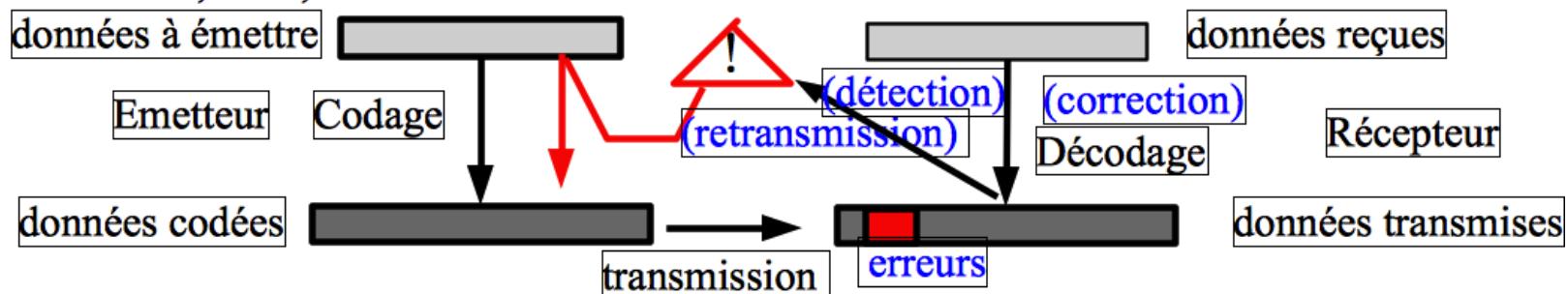
**Note** : certaines erreurs sont détectées et mal corrigées !

- ex. : une même erreur sur deux exemplaires simultanément

# Principe général de retransmission

Principe général pour la correction par **retransmission** des erreurs de transmission

- Après détection d'une erreur, le récepteur demande à l'émetteur, implicitement (temporisateur) ou explicitement (nack), de retransmettre une nouvelle fois le message (codé).
- Exemple : de très nombreux protocoles de télécommunication : HDLC, X25, TCP, TP.



La correction par retransmission est préférée dans les réseaux où le taux de perte est faible et le délai de retransmission tolérable, car son surcoût est généralement plus faible que celui induit par les codes auto-correcteurs.

Estimation du surcoût (message de longueur moyenne = 1000 bits) :

- taux d'erreur typique =  $10^{-9}$ , taux de retransmission  $\Rightarrow 1000 \cdot 10^{-9} = 10^{-6}$
- surcoût typique d'un code auto-correcteur : qq octets par message  $\Rightarrow 28 / 1000 = 10^{-2}$

# Codes de protection contre les erreurs

## Classification des codes

- Deux grandes familles de codes :
  - ➔ Les **codes en bloc** (linéaires, cycliques ou non) : le codage/décodage d'un bloc dépend uniquement des informations de ce bloc.
  - ➔ Les **codes en treillis** (convolutifs, récursifs ou non) : le codage/décodage d'un bloc dépend des informations d'autres blocs (généralement de blocs précédemment transmis).
    - ✓ Un code convolutif s'applique sur une suite infinie de symboles et produit une suite infinie.

On préfère généralement le codage par bloc dans les applications télé-informatiques classiques :

- ➔ Le codage/décodage est plus simple et il y a moins de délai

Par la suite, on ne va présenter que les codes par bloc :

- ➔ codes simples
- ➔ codes linéaires, de Hamming
- ➔ codes polynomiaux
- ➔ codes cycliques

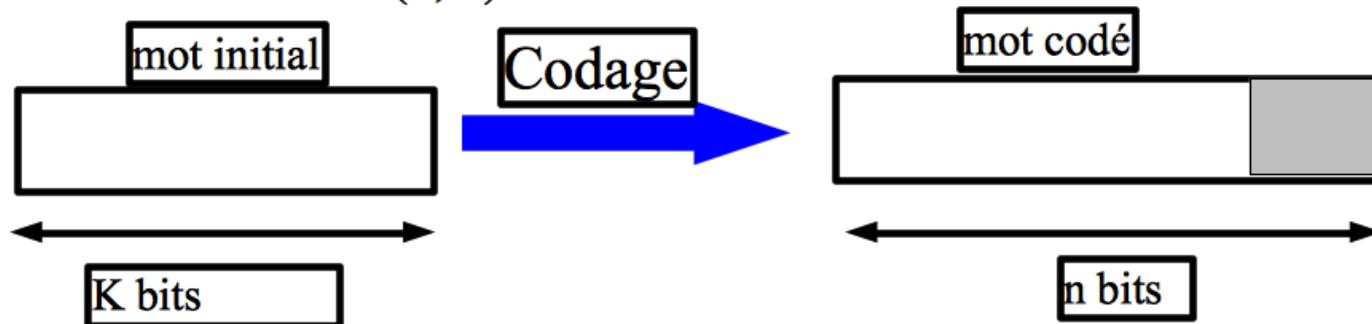
# Définitions générales

Un **code ( $k, n$ )** transforme (code) tout bloc initial de  $k$  bits d'information en un bloc codé de  $n$  bits.

Le code introduit une redondance puisque  $n \geq k$ .

Le code est **systématique** si les  $k$  premiers bits du bloc codé sont égaux aux bits du bloc initial. Alors les  $r$  ( $r=n-k$ ) derniers bits forment un champ de contrôle d'erreur.

Le **rendement** d'un code  $(k, n)$  est :  $R = k/n$



On appelle **mot du code**, la suite de  $n$  bits obtenue après un codage  $(k, n)$ . Le nombre  $n$  de bits qui composent un mot du code est appelé la **longueur du code**. La **dimension  $k$**  étant la longueur initiale des mots.

# Les codes linéaires

## Définitions

- Les **codes linéaires** sont des codes dont chaque mot du code (noté  $c$ ) est obtenu après transformation linéaire des bits du mot initial (noté  $i$ ).
- Un code est linéaire si et seulement si il contient le  $n$ -uplet  $0^n$  et si la somme (ou-exclusif) de deux mots de code est un mot de code.
- Ces codes sont caractérisés par leur matrice  $G(k, n)$  (appelée **matrice génératrice**) telle que :  $i \cdot G = c$
- 
- La matrice  $H_{(n-k, n)}$  (appelée **matrice de contrôle**) permet de savoir si un mot reçu est un mot du code, en calculant son **syndrome**. Si le syndrome du mot est nul, ce mot appartient au code.  $G \cdot H^T = 0$  et  $H \cdot G^T = 0$ 
  - ➔ Syndrome du mot reçu  $c'$  :  $c' \cdot H^T = 0_{(n-k)} \iff c' \in C_{(k, n)}$
- L'équation  $G \cdot H^T = 0$  définit la relation entre les deux matrices.
  - ➔ Preuve :  $c' \cdot H^T = i \cdot G \cdot H^T = i \cdot 0$

## exemple

*Exemples :*

Code de Hamming  $C_{4(4, 7)}$  tel que sa matrice de contrôle  $H_{4(3, 7)} =$

Calculez la matrice génératrice  $G_4$  puis les syndromes de

$c_1 = [0\ 0\ 1\ 0\ 1\ 1\ 1]$ ,

de  $c_2 = [0\ 0\ 1\ 0\ 1\ 1\ 0]$

et de  $c_3 = [0\ 0\ 0\ 1\ 1\ 1\ 1]$

Le code est systématique, la transposée de  $H_4$  :  $H_4^T =$

$$\begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

$$\begin{vmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}, \text{ donc } G_4 = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{vmatrix}$$

$$c_1 \cdot H_4^T = [0\ 0\ 0] \Rightarrow i_1 = [0\ 0\ 0]$$

$$c_2 \cdot H_4^T = [0\ 0\ 1] \Rightarrow \text{erreur}$$

$$c_3 \cdot H_4^T = [1\ 0\ 0] \Rightarrow \text{erreur}$$

**Conclusion :** La méthode matricielle impose de travailler sur des mots de taille fixe, ce qui est un inconvénient majeur pour les réseaux informatiques où les messages sont de taille variable.

## Poids de hamming

Le **poids de Hamming** d'un mot est le nombre de bits à 1 qu'il contient.

La **distance de Hamming** entre deux mots de même longueur est définie par le nombre de positions binaires qui diffèrent entre ces deux mots. On l'obtient par le poids de Hamming de la somme binaire des 2 mots.

La distance de Hamming d'un code est la distance minimum entre tous les mots du code.

*Exemple :* poids\_de\_Hamming(01001100) = 3

distance\_de\_Hamming(01001100, 01010101) = 3

La **capacité de détection** (de correction) d'un code est définie par les configurations erronées qu'il est capable de détecter (corriger).

- Une **erreur simple** (resp. double, ou d'ordre p) affecte une seule (resp. 2, ou p) position(s) binaire(s) d'un mot.
- Pour qu'un code ait une capacité de détection (resp. correction) des erreurs d'ordre  $e$ , il faut que sa distance de Hamming soit supérieure à  $1+e$  (resp.  $1 + 2e$ ).

*Exemple :* distance =3 ==> capacité de détection =< 2, capacité de correction =< 1.

# Contrôle de parité

Parité paire (impaire) : le poids de Hamming des mots du code est pair (impair)

C'est un code systématique ( $k, k+1$ ) dans lequel un bit (le bit de parité) est ajouté au mot initial pour assurer la parité. Son rendement est faible lorsque  $k$  est petit.

Exemple : Transmission de caractères utilisant un code de représentation (le code ASCII sur 7 bits)

<u>Lettre</u>	<u>Code ASCII</u>	<u>Mot codé (parité paire)</u>	<u>Mode codé (parité impaire)</u>
E :	1 0 1 0 0 0 1	; 1 0 1 0 0 0 1 <b>1</b>	; 1 0 1 0 0 0 1 <b>0</b>
V :	0 1 1 0 1 0 1	; 0 1 1 0 1 0 1 <b>0</b>	; 0 1 1 0 1 0 1 <b>1</b>
A :	1 0 0 0 0 0 1	; 1 0 0 0 0 0 1 <b>0</b>	; 1 0 0 0 0 0 1 <b>1</b>

Ce code est capable de détecter toutes les erreurs en nombre impair. Il ne détecte **pas** les erreurs en nombre pair !

# Parité longitudinale et transversale

Association d'un double codage de la parité :

- LRC : “Longitudinal Redundancy Check” et VRC : “Vertical ...”

Le bloc de données est disposé sous une forme matricielle ( $k=a.b$ ). On applique la parité (uniquement paire) sur chaque ligne et chaque colonne. On obtient une matrice  $(a+1, b+1)$

## VRC (parité paire)

$$\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

---

$$\text{LRC} = 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1$$

- Le rendement est faible :  $a.b / (a+1).(b+1)$ .
- Le délai de codage et décodage important : il faut recevoir tout le bloc

# Parité longitudinale et transversale

- Capacité de détection et d'auto-correction :
  - Principe : Une erreur simple modifie simultanément la parité d'un ligne et d'une colonne.
  - Correction : inverser le bit situé à l'intersection de la ligne et de la colonne ayant une parité incorrecte.

**Exemple :**

1	0	1	0	0	0	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	0	1	1
0	1	0	0	1	0	1	0

**Attention** : une erreur triple peut faire croire à une erreur simple et sa correction sera inadaptée !

**Exemple :**

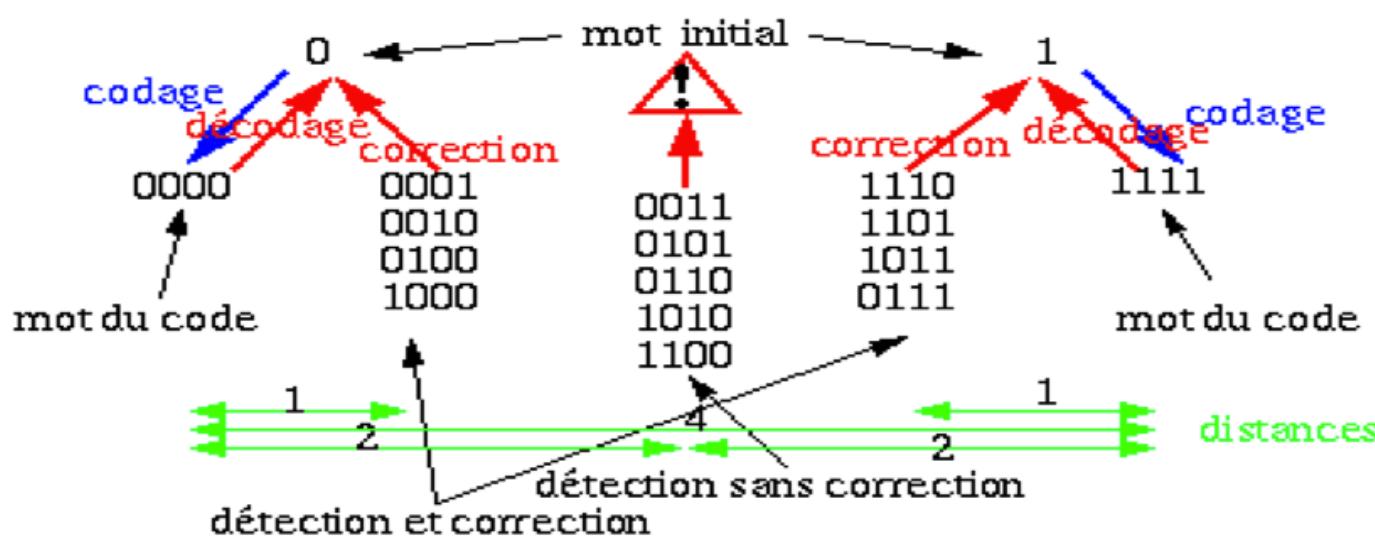
1	0	1	0	0	0	1	1
0	1	<u>1</u>	0	<u>1</u>	0	1	0
1	0	<u>0</u>	0	1	0	1	0
0	1	0	0	1	0	1	1

# Correction

- Correction par proximité : le procédé de correction transforme un mot reçu et détecté comme erroné dans le mot de code le plus proche au sens de la distance de Hamming.
  - Il peut exister plusieurs mots équidistants. On choisit un représentant
  - procédé de correction lourd

**Exemple :**

Le code à répétition  $C_3(1, 4)$ , distance de Hamming de  $C_3 = 4$ .



# Construction de code de détection d'erreurs simples

- $n = m + r$ ,  $n$  mots de code ayant  $m$  bits de données et  $r$  bits de contrôle, le nombre total de mots  $n$  bits est  $2^n$
- La relation suivante doit être vérifiée :  $(n + 1) 2^m \leq 2^n \Rightarrow (m + r + 1) \leq 2^r$ , ainsi connaissant  $m$ , il est possible de déterminer le nombre minimal de bits de contrôle
- Exemples de détection et correction des erreurs simple

Caractères	Code ASCII	R <sub>i</sub> bits de contrôle
H	1 0 0 1 0 0 0	0 0 1 1 0 0 1 0 0 0 0
a	1 1 0 0 0 0 1	1 0 1 1 1 0 0 1 0 0 1
m	1 1 0 1 1 0 1	1 1 1 0 1 0 1 0 1 0 1
m	1 1 0 1 1 0 1	1 1 1 0 1 0 1 0 1 0 1
i	1 1 0 1 0 0 1	0 1 1 0 1 0 1 1 0 0 1
n	1 1 0 1 1 1 0	0 1 1 0 1 0 1 0 1 1 0
g	1 1 0 0 1 1 1	1 1 1 1 1 0 0 1 1 1 1

# Les codes polynomiaux

**Notation** : tout vecteur peut être présenté sous une forme polynomiale :

$$\rightarrow U = \langle u_0, u_1, u_2, \dots, u_n \rangle \iff U(x) = u_0 + u_1 \cdot x + u_2 \cdot x^2 + \dots + u_n \cdot x^n$$

**Attention** : les opérations sont binaires (construits sur le corps  $Z/2Z$ ) :  $1 \cdot x + 1 \cdot x = 0 \cdot x$  !

→ Somme polynomiale :

- ✓ somme deux à deux de chaque coefficient (somme vectorielle)
- ✓ exemple :  $\langle x^3 + 1 \rangle + \langle x^4 + x^2 + 1 \rangle = \langle x^4 + x^3 + x^2 + 1 \rangle$

→ Produit polynomiale :

- ✓ produit vectoriel
- ✓ exemple :  $\langle x^3 + 1 \rangle \cdot \langle x + 1 \rangle = \langle x^4 + x^3 + x + 1 \rangle$

→ Division polynomiale

- ✓ idem
- ✓ Exemple :  $\langle x^4 + x^3 + x^2 + x + 1 \rangle / \langle x + 1 \rangle = \langle x^3 + x^2 + 1 \rangle$

→ **Définition** : Un **code polynomial** est un code linéaire systématique dont chacun des mots du code est un multiple du polynôme générateur (noté  $g(x)$ ).

# Les codes polynomiaux

☒ les lignes de la matrice génératrice sont engendrées par le polynôme générateur  
Le degré du polynôme définit la longueur du champ de contrôle d'erreur.

## *Exemple :*

- soit le polynôme générateur :  $g(x) = x+1$
- on s'intéresse à un code  $C_5(2,3)$  donc les calculs se feront modulo  $x^3+1$
- première ligne :  $g(x) \cdot 1 = x+1$
- deuxième :  $g(x) \cdot (x^2+1) \bmod (x^3+1) = (x^3+x^2+x+1) \bmod (x^3+1) = x^2+x$
- ce qui donne la matrice génératrice suivante :  $G_5(2,3) = \begin{matrix} 0 & 1 & 1 \\ & 1 & 1 & 0 \end{matrix}$

## *Exemples de codes polynomiaux :*

- L'avis V41 du CCITT conseille l'utilisation de codes polynomiaux (de longueurs  $n = 260, 500, 980$  ou  $3860$  bits) avec le polynôme générateur :
  - ✓  $g(x) = x^{16} + x^{12} + x^5 + 1$ .
- Le polynôme CRC-16 est utilisé par le protocole HDLC :
  - ✓  $g(x) = x^{16} + x^{15} + x^2 + 1$ .
- Le polynôme suivant est utilisé par Ethernet :
  - ✓  $g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$ .

# Cyclic Redundancy Code

- Implémentation :
  - ➔ L'émetteur et le récepteur se mettent d'accord sur le choix d'un polynôme générateur  $G(x)$
  - ➔ L'émetteur colle les bits de contrôle ( $r$ ) au bit du polynôme  $M(x)$  ( $m$ ) de manière à ce qu'elle soit divisible par  $G(x)$
  - ➔ Émission de la trame  $m$  bits +  $r$  bits de contrôle
  - ➔ À la réception de la trame, le récepteur la divise par  $G(x) \Rightarrow$  reste non nul, une erreur de transmission a eu lieu
- Application
  - ➔ Soit  $r$  le degré de  $G(x)$ , ajouter  $r$  zéros après le bit de poids faible du bloc. Il contient ainsi  $m+r$  bits correspondant au polynôme  $x^r M(x)$
  - ➔ Effectuer la division modulo 2 du polynôme  $x^r M(x)$  par  $G(x)$
  - ➔ Soustraire modulo 2 le reste de la division de la chaîne de bits correspondant au polynôme  $M(x)$ . Le résultat de cette opération est la trame transmise au destinataire
- Exemples :
  - ➔  $\text{CRC-12} = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$

## Cyclic Redundancy Code

- Exemples : Trame 1101011011, Générateur : 10011, Message après ajouts de 4 bits à 0 : 11010110110000

# Mise en pratique de la division polynomiale

- Soit  $g(x)$ , polynôme générateur de l'avis V41 du CCITT

$$g(x) = x^{16} + x^{12} + x^5 + 1$$

- On construit le circuit correspondant à  $g(x)$  :

- Soit  $g(x)$ , polynôme générateur de l'avis V41 du CCITT :

$$g(x) = x^{16} + x^{12} + x^5 + 1$$

- On construit le circuit correspondant à  $g(x)$  :

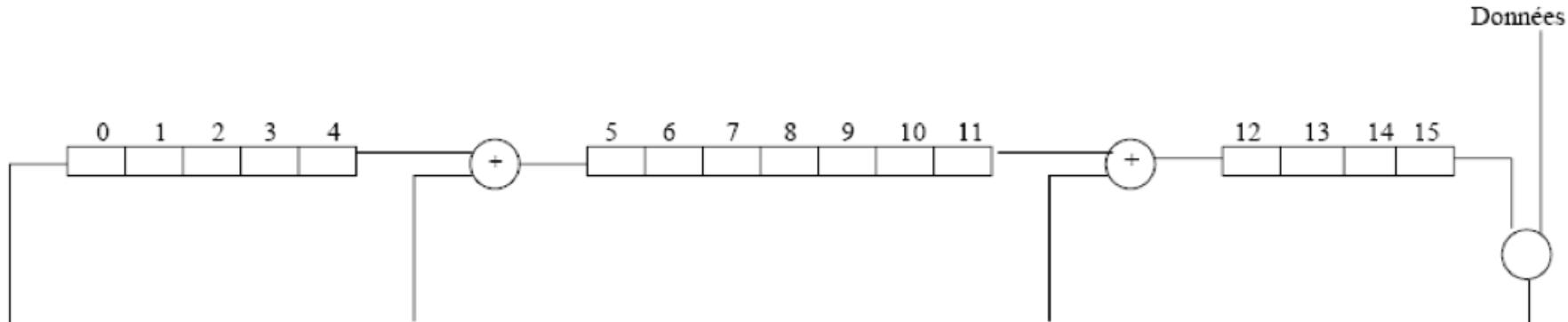


Figure 1: Circuit de calcul du CRC

# Capacité de détection des erreurs

Un code polynomial ( $k, n$ ) dont le polynôme générateur a plus d'un coefficient non-nul (donc il ne divise pas  $x^i$ ,  $i < n$ ) permet de détecter toutes les erreurs simples.

- idée de preuve :  $(C(x) + x^i) \bmod g(x) \neq 0$

Si le polynôme générateur d'un code polynomial ( $k, n$ ) a un facteur irréductible de trois termes (il ne divise ni  $x^i$  ni  $1 + x^{j-i}$ ,  $i < j < n$ ), le code permet de détecter les erreurs doubles.

- par exemple :  $g(x) = x^3 + x + 1$

Pour qu'un code polynomial détecte toutes les erreurs d'ordre impair, il suffit que son polynôme générateur ait  $(x+1)$  comme facteur.

- Par exemple : le code de polynôme générateur  $(x+1)$  qui est équivalent à la parité.

*Capacité de détection des paquets d'erreurs:*

Un code polynomial ( $k, n$ ) permet de détecter toutes les erreurs d'ordre  $1 \leq n-k$  (c'est-à-dire inférieur au degré du polynôme générateur).

Et la probabilité de ne pas détecter les erreurs d'ordre  $l > n-k$  est très faible et égale à<sub>1</sub> :  $2^{-(n-k)}$

## Principe du codage

Le mot de code  $m(x)$  d'un code polynomial  $(k, n)$  de polynôme générateur  $g(x)$  associé au mot initial  $i(x)$  est défini par :

- $m(x) = i(x) \cdot x^{n-k} + r(x)$ , où  $r(x)$  est le reste de la division de  $i(x) \cdot x^{n-k}$  par le polynôme générateur  $g(x)$  (noté :  $r(x) = (i(x) \cdot x^{n-k}) \bmod g(x)$  ).

*Remarques :*

- Les  $r = n-k$  bits de  $r(x)$  (de degré  $\leq n-k-1$ ) forment les bits du champ de contrôle.
- Les bits de poids fort (de degré  $> n-k-1$ ) forment le mot initial (→ code systématique)
- L'opération de codage effectuée à l'émission est ramenée à une division polynomiale, qui peut être réalisée simplement (électroniquement).

## Principe du décodage

A la réception, chaque mot reçu  $m'(x)$  est divisé par le polynôme générateur  $g(x)$ .

- Un reste **non-nul** indique qu'il y a eu **erreur** lors de la transmission.
- Syndrome de  $m'(x)$  :  $m'(x) \bmod g(x) \neq 0 \implies$  Erreur !

**Attention :** la réciproque est fausse ! Si le reste est nul cela ne veut pas dire qu'il n'y a pas eu d'erreurs --> subsistance d'erreurs dites **résiduelles**.

## Principe du décodage

A la réception, chaque mot reçu  $m'(x)$  est divisé par le polynôme générateur  $g(x)$ .

- Un reste **non-nul** indique qu'il y a eu **erreur** lors de la transmission.
- Syndrome de  $m'(x)$  :  $m'(x) \bmod g(x) \neq 0 \implies$  Erreur !

**Attention :** la réciproque est fausse ! Si le reste est nul cela ne veut pas dire qu'il n'y a pas eu d'erreurs --> subsistance d'erreurs dites **résiduelles**.

# Les codes cycliques

## Définitions

- Un **code cyclique** ( $k, n$ ) est un code linéaire ( $k, n$ ) tel que toute permutation circulaire d'un mot du code est encore un mot du code.

### *Exemple :*

- Un code cyclique (1, 2) possède les mots de code suivants : {01, 10} ou {00, 11}, mais pas {01,11}.
- Un code cyclique (1, 3) possède les mots de code suivants : {000, 111}.
- On appelle **période** du polynôme  $H(x)$  le plus petit entier  $u$  tel que  $H(x)$  divise  $x^u+1$ .
- Un polynôme est **irréductible** s'il ne possède aucun diviseur de degré supérieur à zéro.
- Si la période d'un polynôme irréductible est égale à  $n-k$  alors le polynôme est dit **primitif**.

# **Techniques de contrôle d'erreur par retransmission**

## ***Techniques de détection des erreurs***

L'écho permet de s'assurer que le récepteur a reçu **correctement** les informations émises.

Exemple : technique utilisée sur les liaisons (asynchrones) entre les ordinateurs et les périphériques par caractères !

Actuellement, peu utilisé car très mauvais rendement et délai important.

## ***Technique d'auto-correction par répétition***

Les informations à transmettre sont répétés plusieurs fois. Au moins un des exemplaires (redondants) sera correctement reçu.

Très mauvais rendement.

# Technique de correction par retransmission

## Principe de fonctionnement

- L'émetteur conserve une copie des données qu'il envoie.
- Le récepteur détecte les erreurs grâce à la présence d'un champ de contrôle d'erreur (code polynomial) dans les paquets de données.
- Le récepteur informe l'émetteur de la bonne (resp. mauvaise) réception en lui retournant un paquet spécifique :
  - ✓ **acquittement** positif (resp. négatif) souvent appelé ACK (resp. NACK)
- Dans le cas d'un acquittement négatif, l'émetteur doit ré-émettre le paquet erroné.
- Sinon il peut émettre le prochain paquet.

--> *Protocole "Send and wait"*

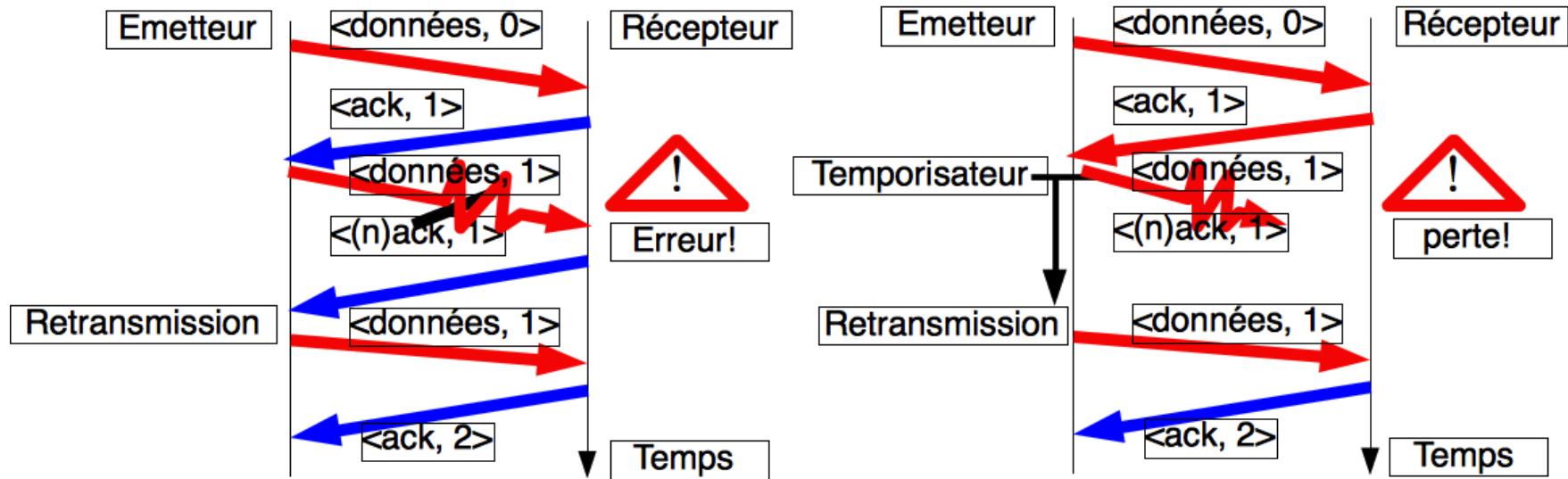
Un **temporisateur** bornant la durée d'attente des acquittements est nécessaire pour assurer la correction du mécanisme lors des pertes de paquets de données.

L'**identification** des paquets (de données et d'acquittement) est nécessaire pour assurer la correction du mécanisme lors des pertes d'acquittement : au moins numérotation modulo 2.

--> *Protocole "du bit alterné"*

# Exemple de fonctionnement

- Fonctionne à l'alternat : Emetteur --> Récepteur



Communication bidirectionnelle : x 2 !

Son rendement très faible :

- ➔ si le temps de transmission Tt est faible vis-à-vis du temps de propagation Tp.
- ➔ car la liaison est inutilisée lorsque l'émetteur attend l'acquittement.
- ➔  $R = Tt / (2Tp + Tt)$

--> *Optimisation : mécanisme de la fenêtre coulissante ("sliding window") !*

# Protocole de liaison simple pour un canal bruité

## ***Principe :***

- Emetteur émet une trame
- Après expiration du temporisateur, l'émetteur retransmet la même trame s'il n'a pas reçu d'acquittement
- Récepteur émet une trame d'acquittement si les données ont été correctement reçues

## ***Un scénario de ce protocole :***

- 1) A envoie une trame à B
- 2) B reçoit la trame de A et envoie l'acquittement à A
- 3) Perte de l'acquittement
- 4) Le temporisateur de A expire, A renvoie la trame à B qui arrive correctement

# Bibliographie

- TCP/IP architecture protocoles applications, Douglas Cormer
- Computer Networking A top-Down Approach Featuring the Internet, James Kurose et Keith Ross, second Edition
- Réseaux et Télécoms, Claude Servin. Dunod