GEI460 - Réseaux et Téléinformatique

Roch Lefebvre, Prof.

Objectifs

- Connaître le rôle et le fonctionnement des différentes couches des réseaux informatiques actuels
- Etre en mesure d'utiliser les fonctions de l'interface transport (Winsock API de Windows) pour transmettre des messages sur un réseau local
- Décrire et implanter un protocole de communication fiable pour une application distribuée (transfert de fichier)

Horaire des cours

- Cours :

Lundi 8h30-9h30	local SA-308
Mardi 8h30-10h30	local SA-308

- Laboratoires :

Jeudi	15h30-18h30	local SA-318
Vendredi	13h30-16h30	local SA-318

Bibliographie

- Andrew S. Tanenbaum, “Computer Networks”, 3e édition, Prentice Hall, 1996.

- Bob Quinn, Dave Shute, “Windows Sockets Network Programming”, Addison-Wesley, 1995. (optionnel)

- Notes de cours (présentation PowerPoint)

Evaluation

- 1 Examen Intra (25 %)
- 3 Laboratoires (45 %)
- 1 Examen Final (30 %)

Laboratoires

(15 % chacun)

- Laboratoire 1: Application distribuée simple ("*chat*")
- Laboratoire 2: Définition d'un protocole d'échange de fichiers (client ET serveur)
- Laboratoire 3: Implantation du laboratoire 2 (client ou serveur, interopérable)

Contenu

- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

Introduction

Couche Physique

Couche Liaison de données

Sous couche MAC (les réseaux locaux)

Couche Réseau

Couche Transport

Couche Application

Réseau

Interconnexion de machines (hôtes) avec la capacité de communiquer entre elles

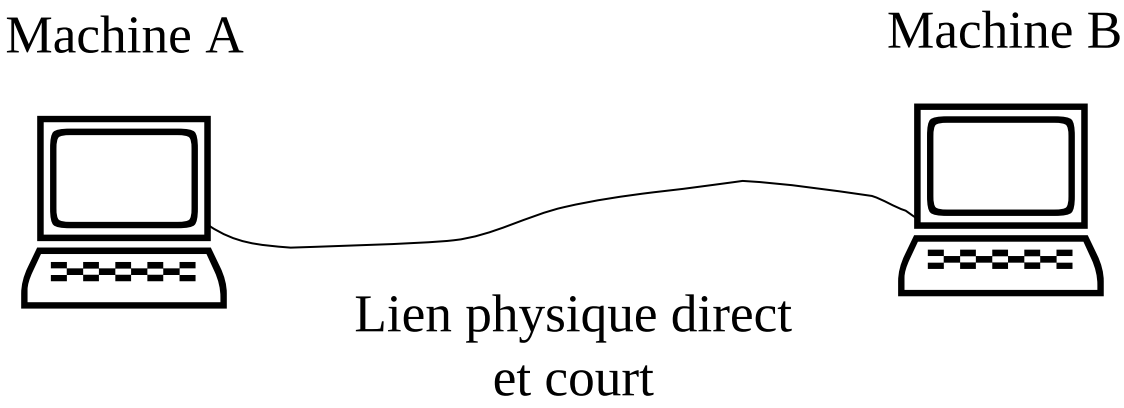
$\rightarrow$ Système de communication

Téléinformatique

Utilisation à distance de ressources informatiques

$\rightarrow$ Services

Réseau le plus simple



Problèmes d'un réseau réel

- Transmission (modulation, synchronisation, …)
- Récupération des erreurs (bruit, débordement de mémoire)
- Partage d'un canal (accès multiple)
- Résolution d'adresses
- Acheminement
- Sécurité
- Intégration des Services
- Qualité de Service
- ...

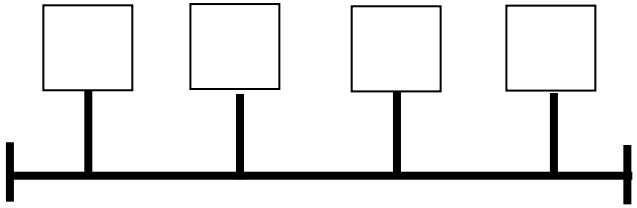
ARPANET: les premiers pas

- Commandé par le DoD américain (années 60)
- Premier réseau à commutation par paquets
- Basé sur le principe “store-and-forward”
- Evolution
  - Déc. 1969 : 4 réseaux (1 hôte par réseau…)
  - Juil. 1970 : 8 réseaux
  - Mars 1971 : 15 réseaux
  - Avril 1972 : 25 réseaux
  - Sept. 1972 : 34 réseaux
  - 1983 : plus de 200 réseaux
  - 1990 : R.I.P. (place à l’Internet et au WWW)

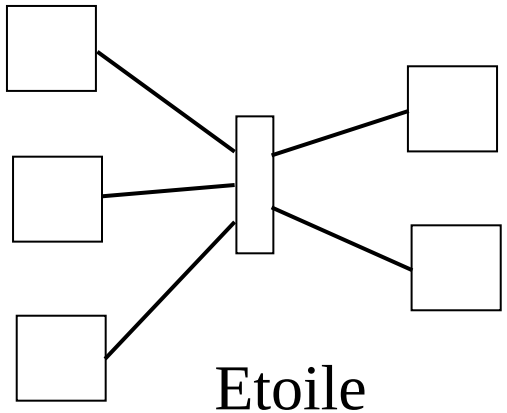
Echelles de grandeur

Distance entre les processeurs	Processeurs dans le (la) même	Exemple
0.1 m	Circuit	Proc. parallèles
1 m	Système	Multi-ordinateur
10 m	Pièce	LAN
100 m	Edifice	“
1 km	Campus	“
10 km	Ville	MAN
100 km	Pays	WAN
1 000 km	Continent	“
10 000 km	Planète	l’Internet

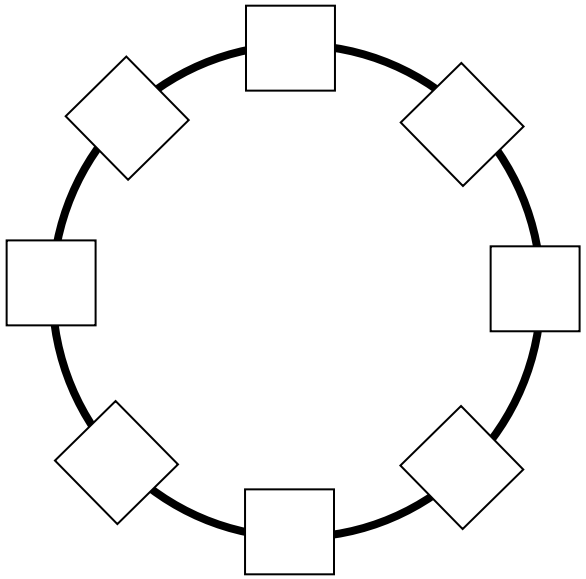
Réseaux locaux: configurations



Bus

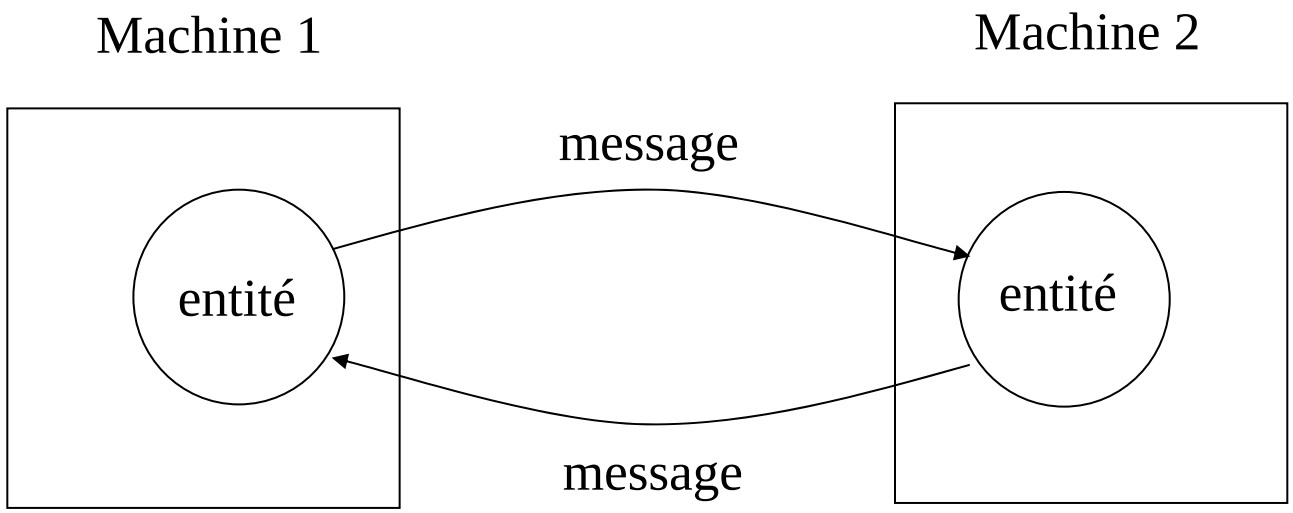


Etoile

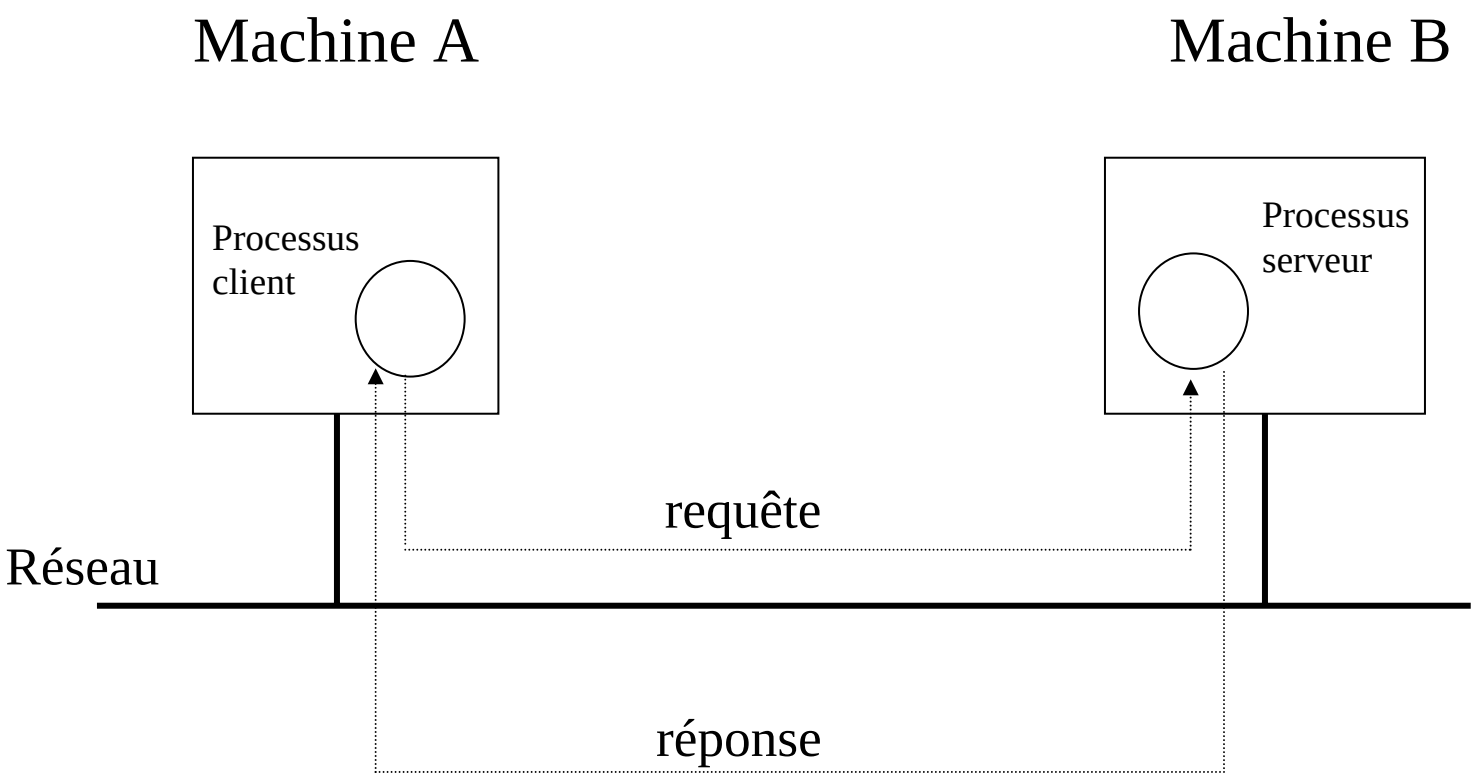


Anneau

Entités et messages



Modèle client/serveur



Primitives

Un service = ensemble de **primitives** (opérations)

Primitive	Signification
Requête	Une entité veut utiliser un service
Indication	Un service informe une entité qu'une action a été prise
Réponse	Une entité répond à une action (typiquement, une requête)
Confirmation	La réponse à une requête est arrivée

Deux grandes classes de services

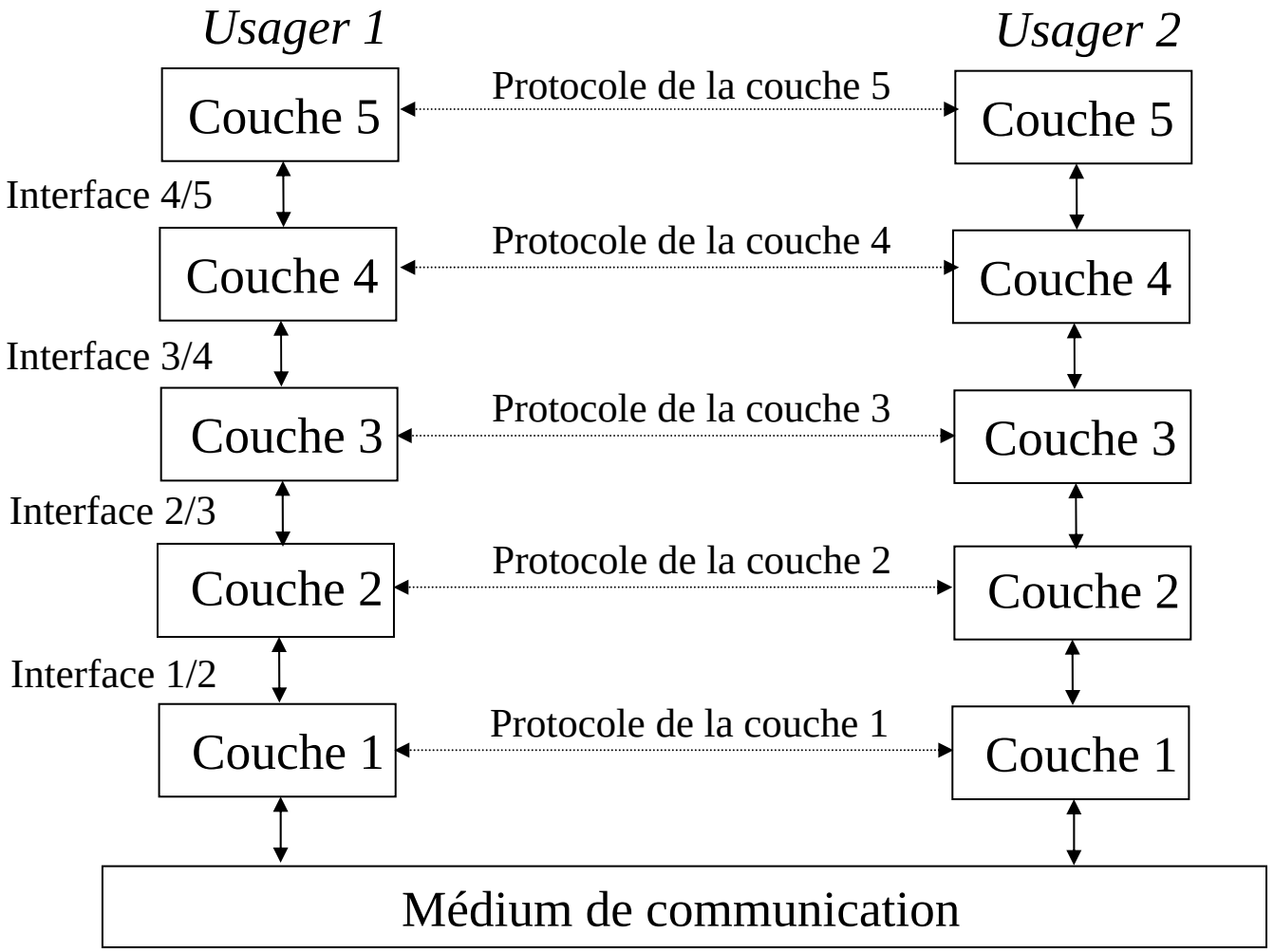
Service avec connexion

- Tous les paquets arrivent au récepteur
- L'ordre est le même à l'émetteur et au récepteur
- Exemple: réseau téléphonique

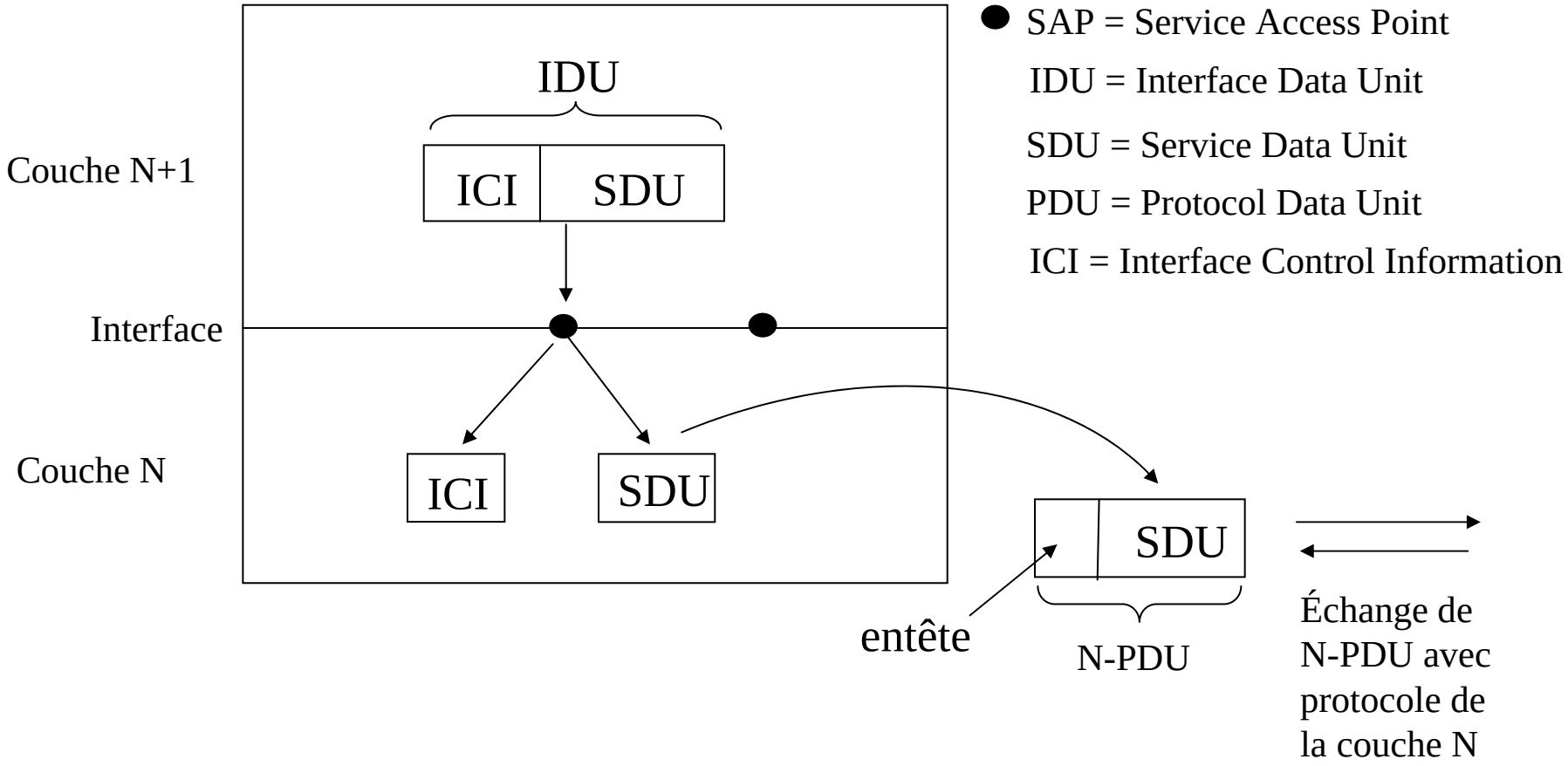
Service sans connexion

- Pas de guarantie de livraison au récepteur
- L'ordre des paquet n'est pas assuré
- Exemple: service postal

Hiérarchie de protocoles



Interface



- SAP = Service Access Point

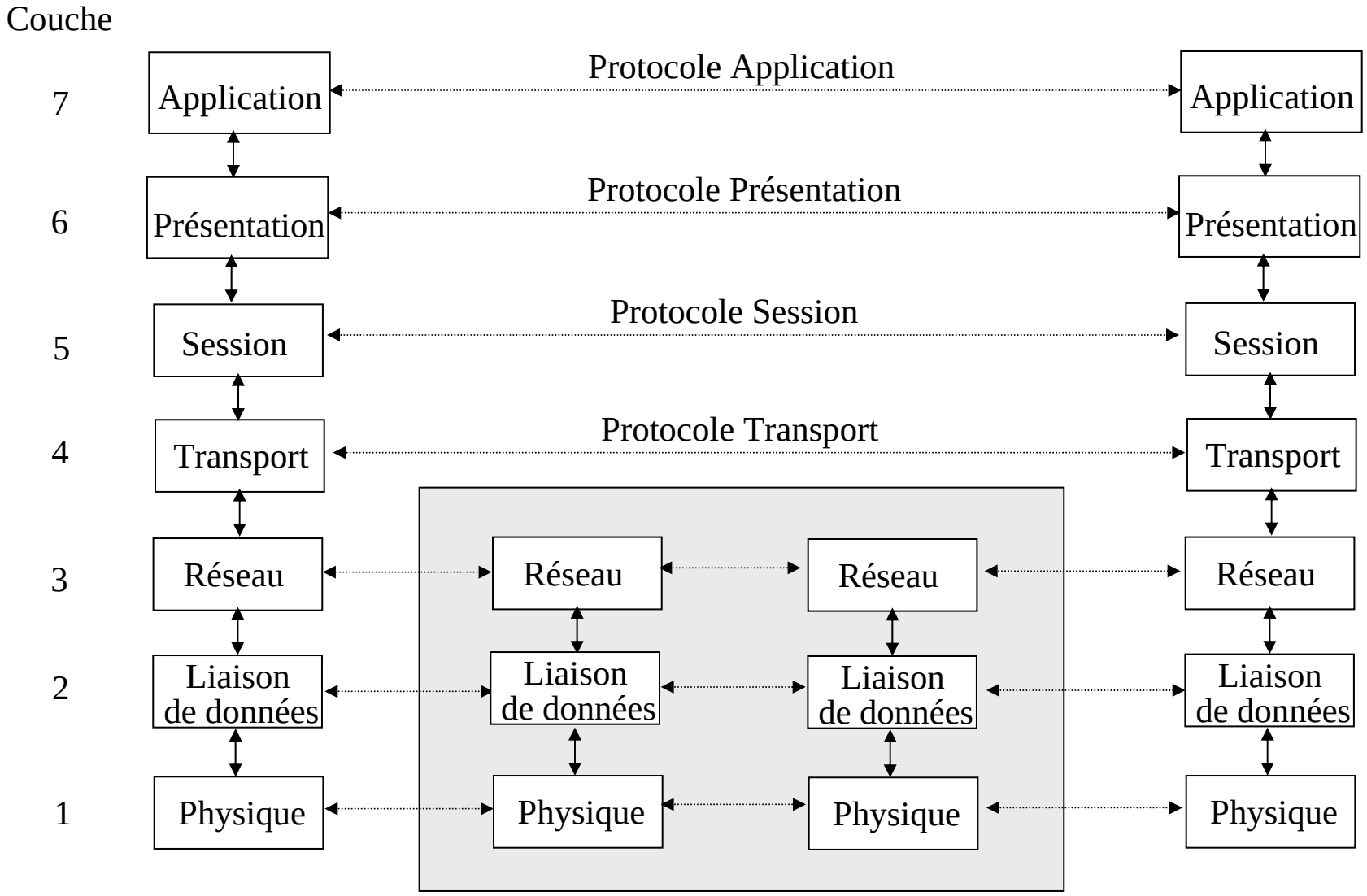
IDU = Interface Data Unit

SDU = Service Data Unit

PDU = Protocol Data Unit

ICI = Interface Control Information

Le modèle OSI



La couche physique

- Assure un service de **transmission de bits** sur un canal de communication
- Possibilités d'erreurs de transmission (bruit de canal)
- Les erreurs devront être récupérées par les couches supérieures

La couche liaison de données

- Assure un service **fiable** de transmission de données
- Décompose et transmet les données en **trames** qui sont transmises séquentiellement (confiées à la couche physique)
- Processus d'**acquiescements** entre récepteur et émetteur
- Possibilité de **retransmission** d'une trame erronée
- Ajustement du débit en fonction de la capacité du récepteur

La couche réseau

- **Achemine les paquets** de la source à la destination (possiblement à travers une séquence de “routeurs”)
- Mécanismes d’acheminement (“routing”) qui réagissent à la congestion et aux conditions du réseau
- Un paquet comporte une entête où sont inscrites (entre autres) les **adresses** source et destination
- La couche réseau peut **fragmenter** les paquets si leur taille dépasse la capacité d’un réseau

La couche transport

- Assure un service **fiable** de transmission de paquets
- Est à la couche réseau ce que la couche liaison de données est à la couche physique
- Mécanismes de **numérotation** et d'**acquiescement** des paquets (avec possibilité de retransmission des paquets)
- Possibilité de service fiable ou sans guarantie
- Ajustement du débit en fonction de la capacité du récepteur

La couche session

- Etablit une **session** entre deux hôtes (durée du dialogue)
- Gère la communication à haut niveau, par exemple à l'aide de **jetons** donnant le contrôle à un usager à la fois
- Permet la **synchronisation** de longues transmissions, en insérant des points de repère dans le message (reprise rapide après une panne)

La couche présentation

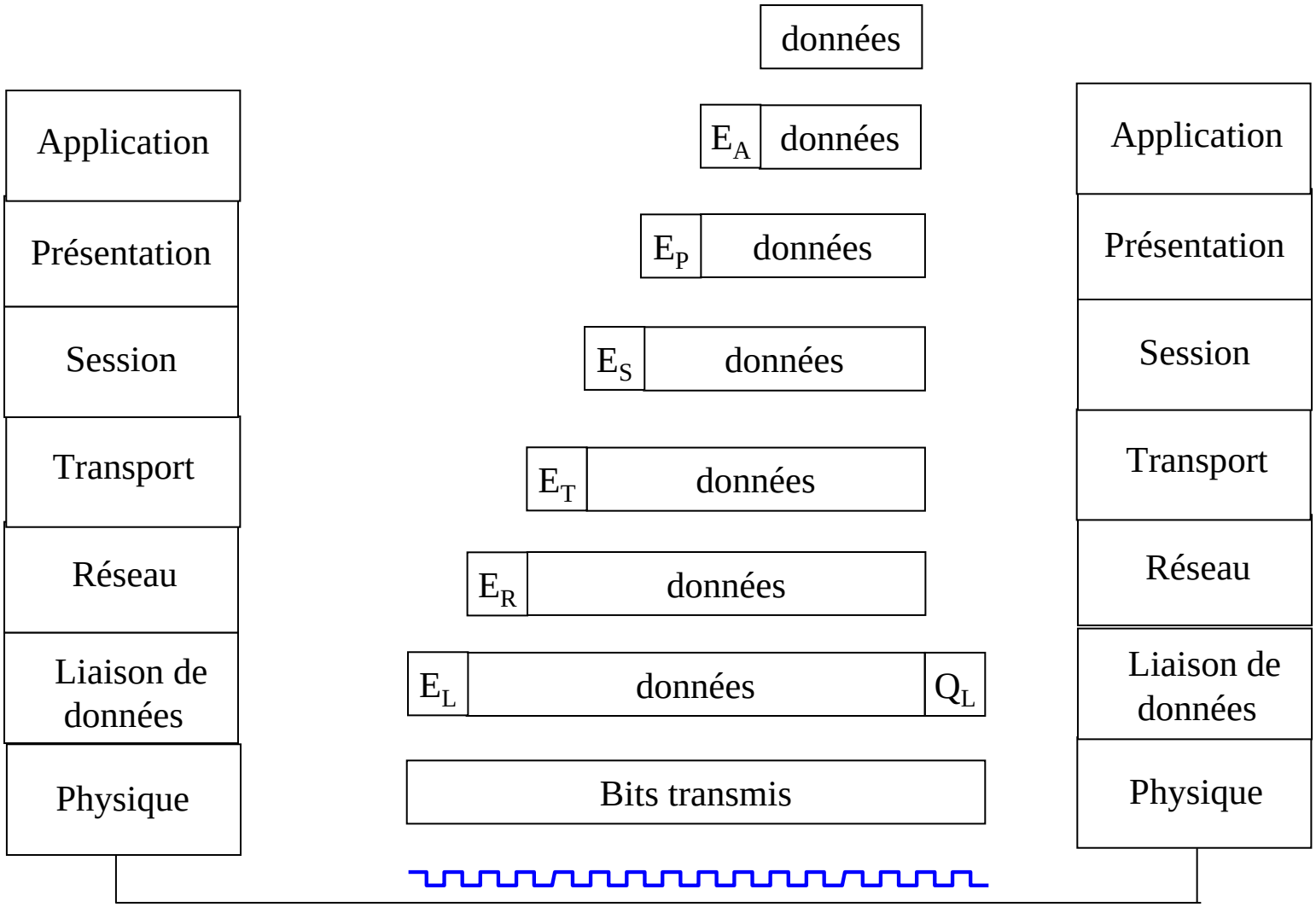
- Permet la **traduction** entre deux réseaux qui n'utilisent pas la même syntaxe de données

- Définition d'une structure abstraite de données: **ASN.1**

La couche application

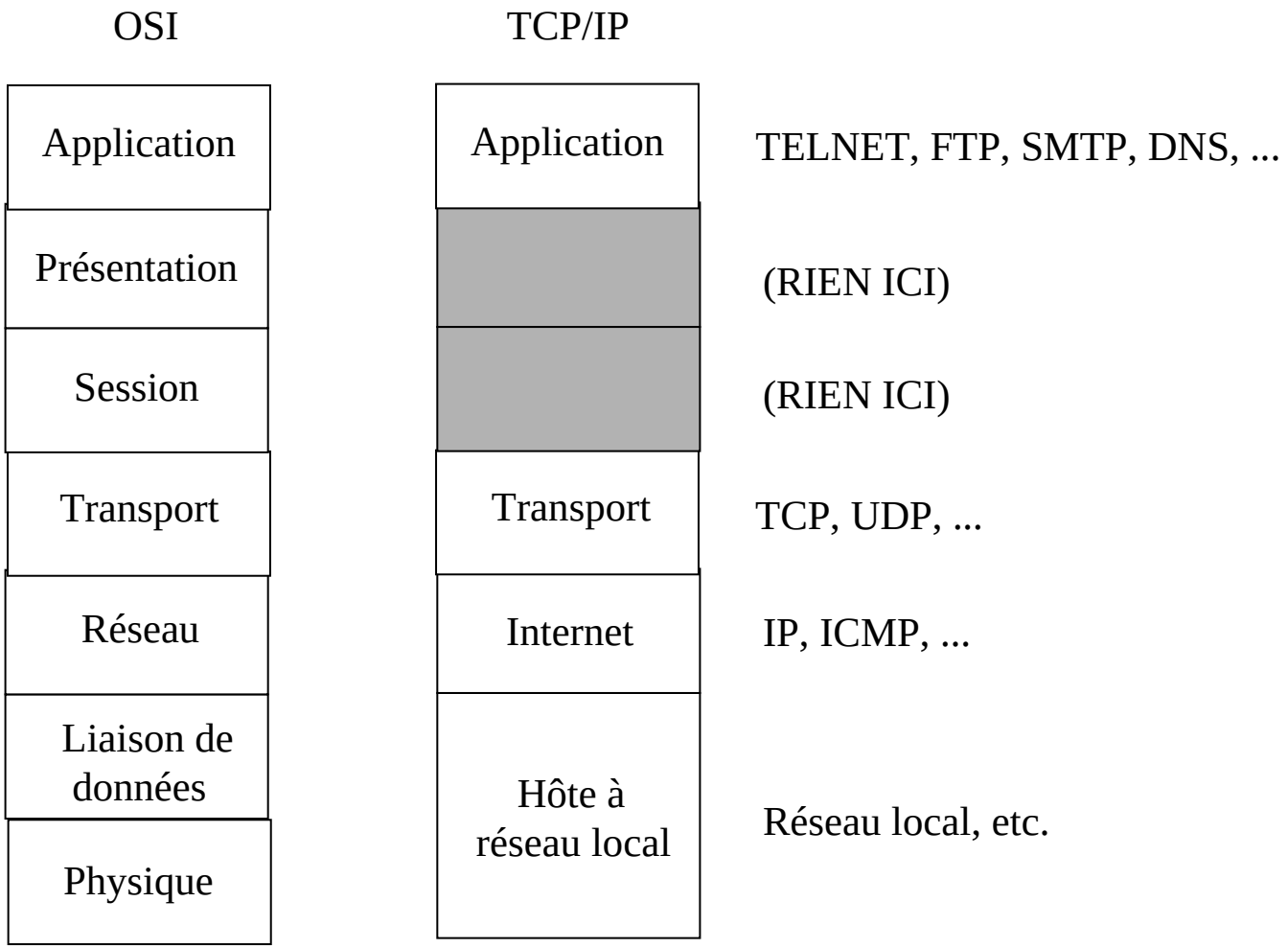
- Permet des traductions de haut niveau
- Par exemple, définition d'un **terminal virtuel**
- Permet aussi la traduction entre différentes conventions pour les noms de fichiers, la structure d'un répertoire, etc.
- Autres fonctions: courrier électronique, recherche de fichiers à distance, cryptographie, etc.

Encapsulation dans le modèle OSI

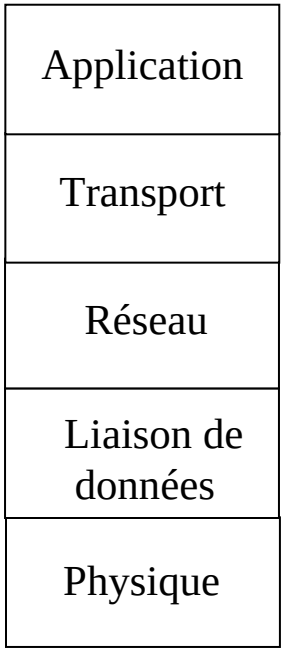


Le modèle TCP/IP

TCP/IP est devenu le protocole officiel d'ARPANET le 1er janvier 1983.



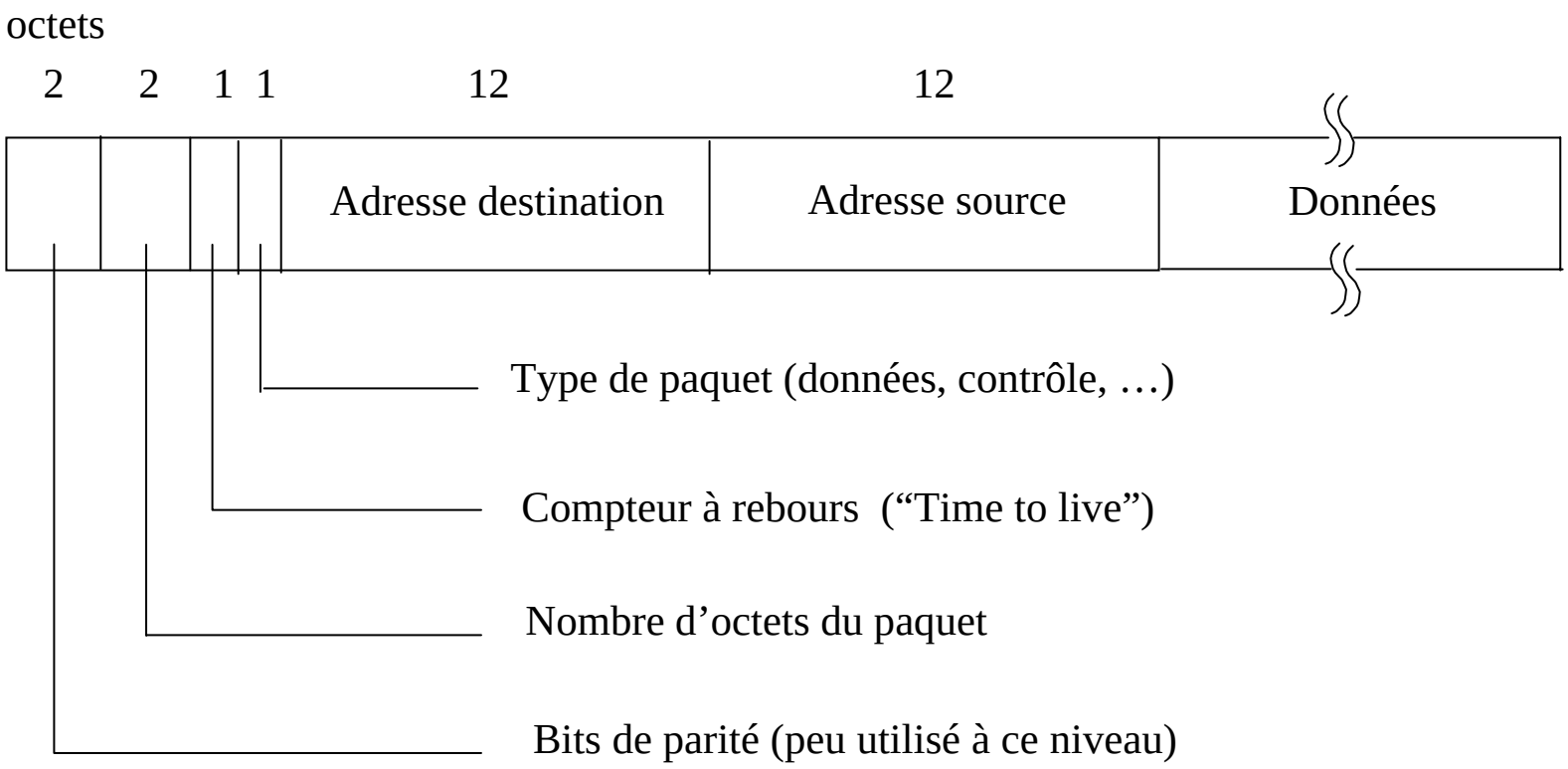
Modèle hybride plus réaliste



Empilement de protocoles sur le réseau Novell

Application	Serveur de fichiers	SAP (les serveurs s'affichent…)	. . .
Transport	NCP (orienté connexion)	SPX	. . .
Réseau	IPX (Internet Paquet eXchange) (adresses de 12 octets; service non orienté-connexion)		
Liaison de données	Ethernet	Anneau à jeton	ARCnet
Physique	Ethernet	Anneau à jeton	ARCnet

Structure d'un paquet IPX



Réseaux multi-Gb/s

- Plus de débit ne signifie pas toujours moins de délai (???)

- Exemple: transmission d'**un paquet** de 1000 bits
  de New-York à San Francisco

  Choix 1: canal à 1 Mb/s $\rightarrow$ 1 ms pour transmettre
  Choix 2: canal à 1 Gb/s $\rightarrow$ 0.001 ms pour transmettre

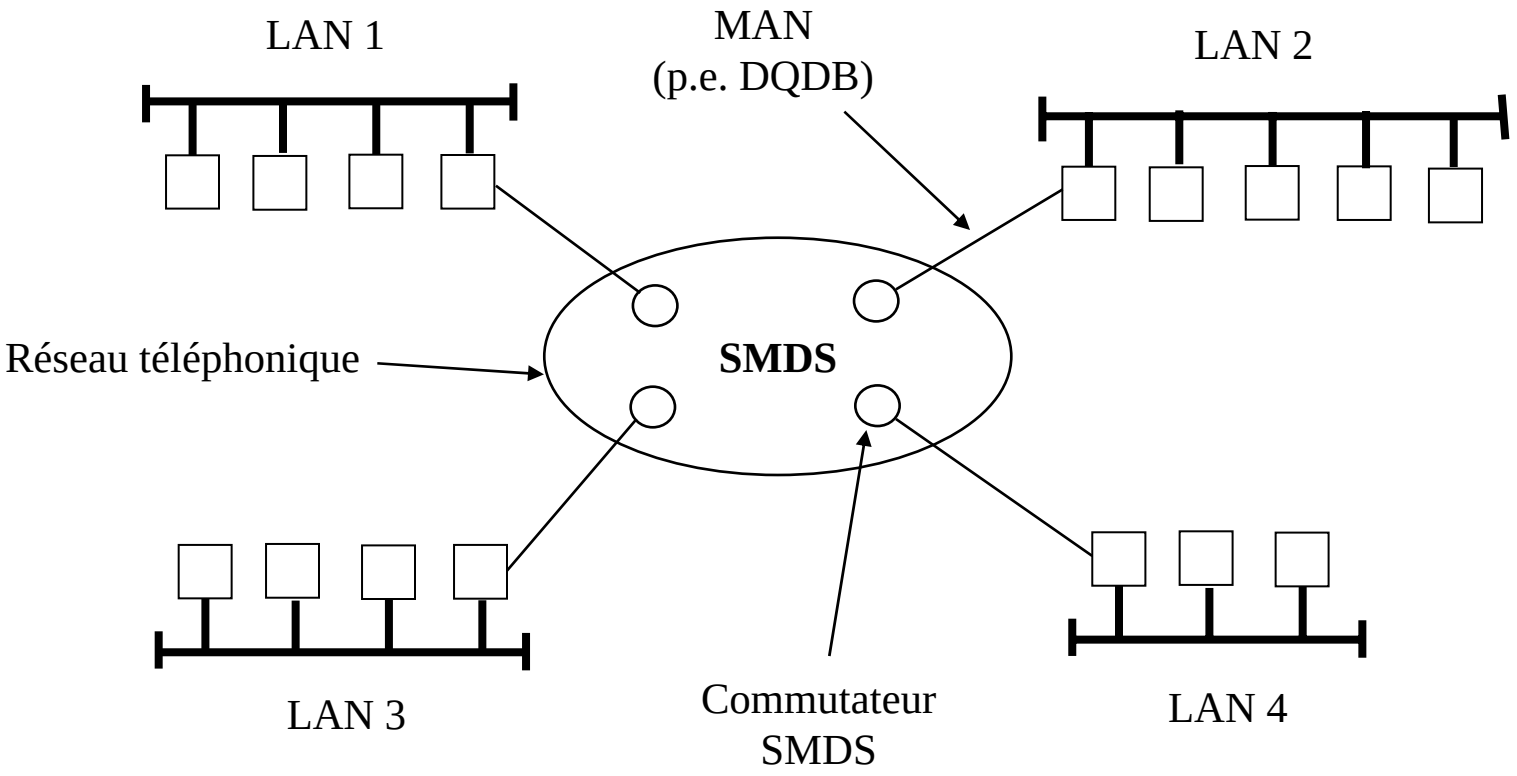
  MAIS dans les 2 cas, le délai de **propagation** est 20 ms
  $\rightarrow$ 4000 km / (200 000 km/s)

  Donc: délai total pour choix 1 = 21 ms
  délai total pour choix 2 = 20.001 ms

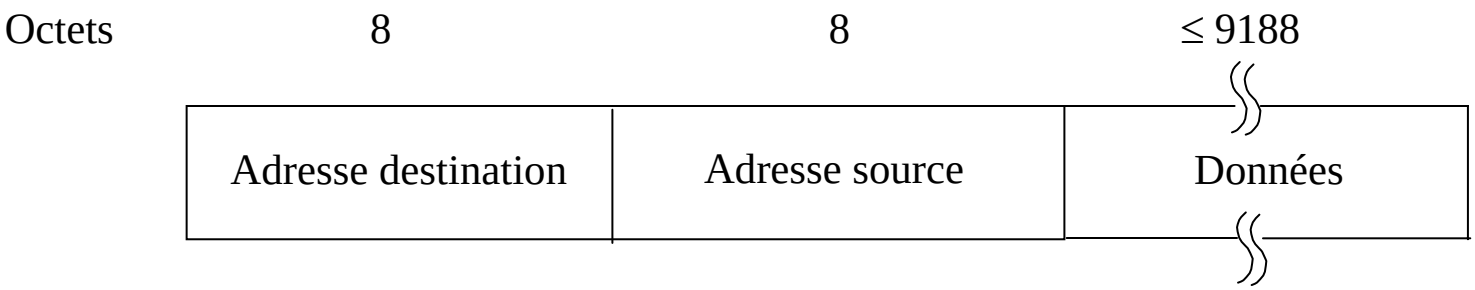
- Evidemment, en général, $\uparrow$ le débit = ☺

SMDS

Switched Multimegabit Data Service



Trame SMDS



- Service sans connexion
- Adresses basées sur numéros de téléphone

Les réseaux X.25

- Norme encore utilisée par certains réseaux publics
- Service orienté connexion (**circuits virtuels**)
  - aquiescement
  - retransmission des trames erronés
  - contrôle du flux des données (capacité du récepteur)
- Trames limitées à 128 octets de données
- Débits limités à 64 kb/s dans la plupart des cas

Frame relay

- Service **simple** de commutation par paquets
  - aucun acquiescement
  - aucun contrôle de flux des données

- **Circuits virtuels** permanents
  - le débit de pointe peut être important
  - le débit moyen doit être sous un certain seuil
  - moins cher qu'un lien téléphonique permanent

- Un lien virtuel = environ 1.5 Mb/s (une ligne T1)

B-ISDN et ATM

- Possiblement à la base de la future "autoroute de l'information"
- B-ISDN: Broadband Integrated Services Digital Network
- ATM: Asynchronous Transfert Mode
- Transport asynchrone de trames de taille fixe (**cellules**)
- Acheminement par circuits virtuels
- Plusieurs gammes de débits et QdS associée

Cellule ATM

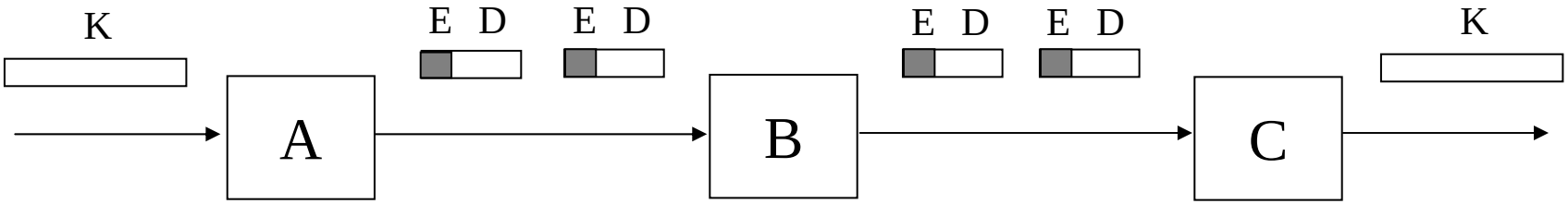
Octets: 5 48

- Identification du circuit virtuel (VCI, VPI)
- Contrôle entre un hôte et le réseau
- Type de données
- Sensibilité aux pertes de trames (1 bit)
- Parité pour détection d’erreur (1 octet)

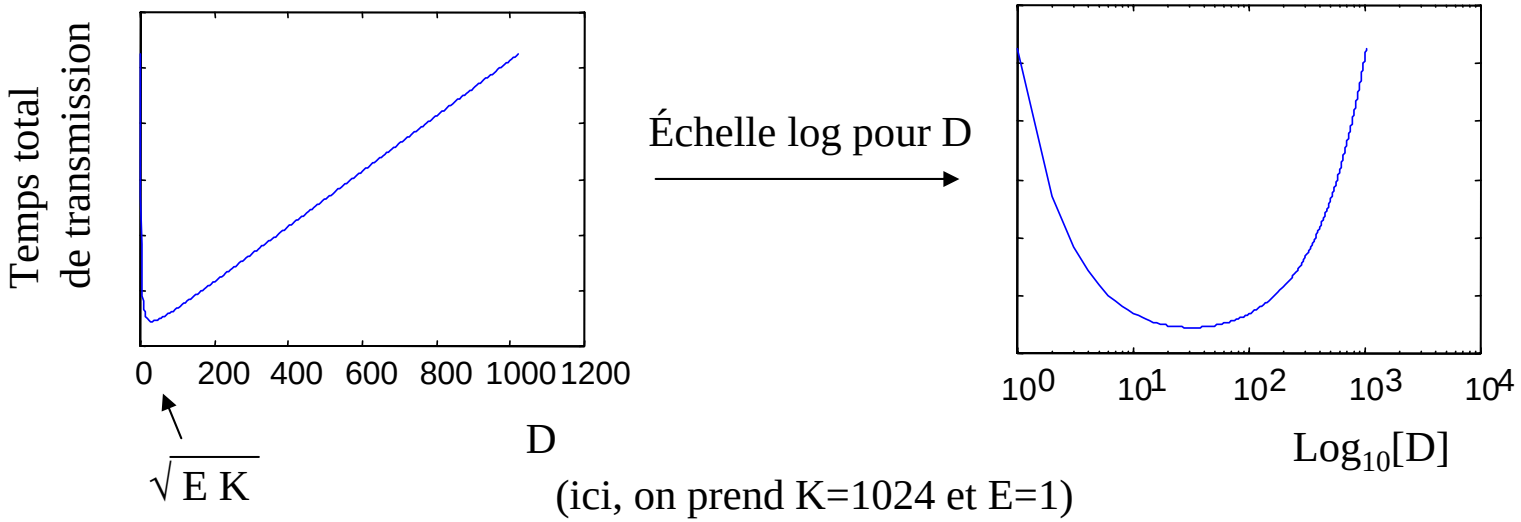
Avantages de la commutation de cellules

- Grande variété de débits (fixes ou variables)
- Commutation très rapide (hautement parallèle)
- Diffusion simultanée multi-usagers (“broadcasting”)
  - impossible à réaliser avec commutation de circuit

"Store-and-forward" taille optimale des paquets



D = K / N = taille d'un paquet (sans l'entête E)



(ici, on prend K=1024 et E=1)

Problème 14, page 75 (Tanenbaum)

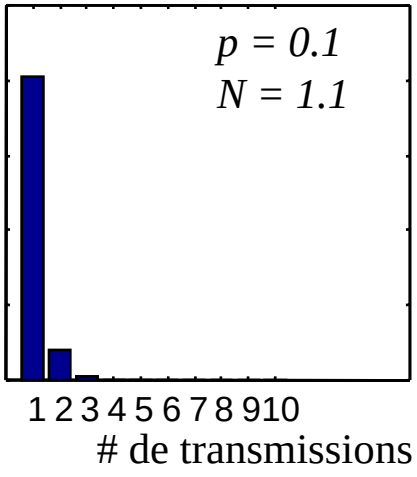
- Probabilité de retransmission d'un paquet : p
- Quelle est le nombre *moyen* de transmissions par paquet?

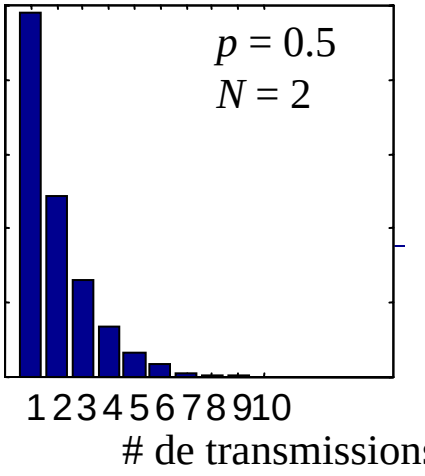
Soit N ce nombre moyen de transmissions. On a:

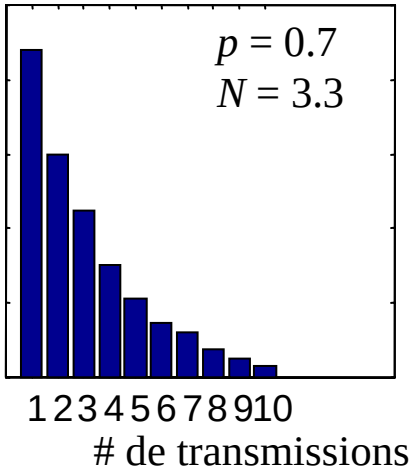
$$
\begin{aligned}
N &= 1\,(1-p) + 2\,p\,(1-p) + 3\,p^2\,(1-p) + 4\,p^3\,(1-p) + \ldots \\
&= 1 + (-p+2p) + (-2p^2+3p^2) + (-3p^3+4p^3) + \ldots \\
&= 1 + p + p^2 + p^3 + p^4 + \ldots \\
&= 1 / (1-p)
\end{aligned}
$$

(si $p < 1$)

Problème 14, page 75 (Tanenbaum)
Exemples







- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

La couche physique

- Assure un service de **transmission de bits** sur un canal de communication
- Possibilités d'erreurs de transmission (bruit de canal)
- Les erreurs devront être récupérées par les couches supérieures
- **Multiplexage** et **Commutation** $\rightarrow$ utilisation optimale

Capacité d'un canal

$$Capacité = H \log_2 (1 + S/B) \quad b/sec$$

H = largeur de bande du canal
S/B = rapport signal à bruit (linéaire)

Exemple

H = 3000 Hz (bande téléphonique)
S/B = 30 dB (1000 en échelle linéaire)

$$Capacité = 3000 \log_2 (1+1000) = 29\,900 \text{ b/sec}$$

Supports physiques

- Paire de **fils de cuivre** torsadés
  - connexions locales du réseau téléphonique
  - Capacité = plusieurs Mb/s (quelques km)

- **Câble coaxial**
  - Capacité = jusqu'à 2 Gb/s (quelques km)

- **Fibre optique**
  - Capacité = 1000 Gb/s et plus (en principe)
  - Limites dues à l'interface électrique/optique (1 Gb/s)
  - transmission unidirectionnelle

Transmissions sans fil

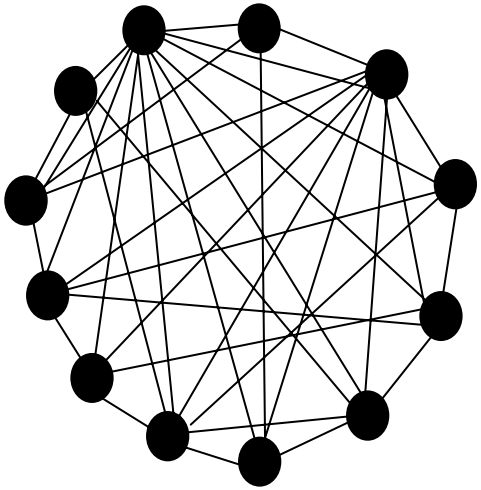
- **Ondes radio** (10 kHz - 100 MHz)
  - Omnidirectionnelles

- **Micro-ondes** (1-100 GHz)
  - Unidirectionnelles
  - MCI = “Microwave Communications Inc.” (originalement, réseau micro-ondes)

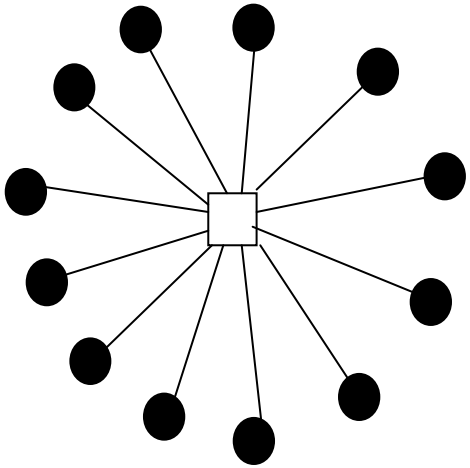
- Ondes **Infrarouges** (ex.: télécommande)

- Ondes **visibles** (“lightwave”)

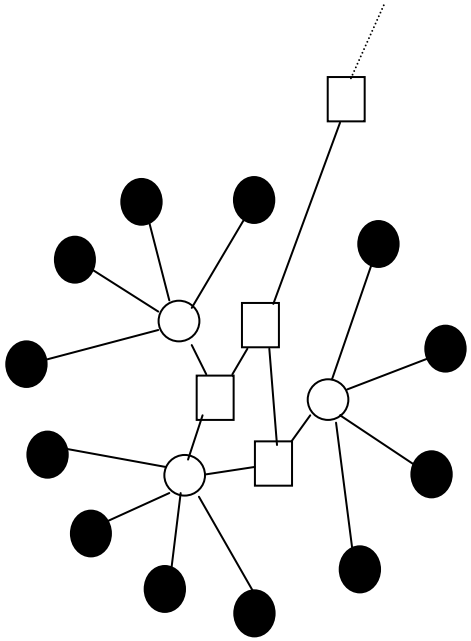
Connectivité



Point-à-point

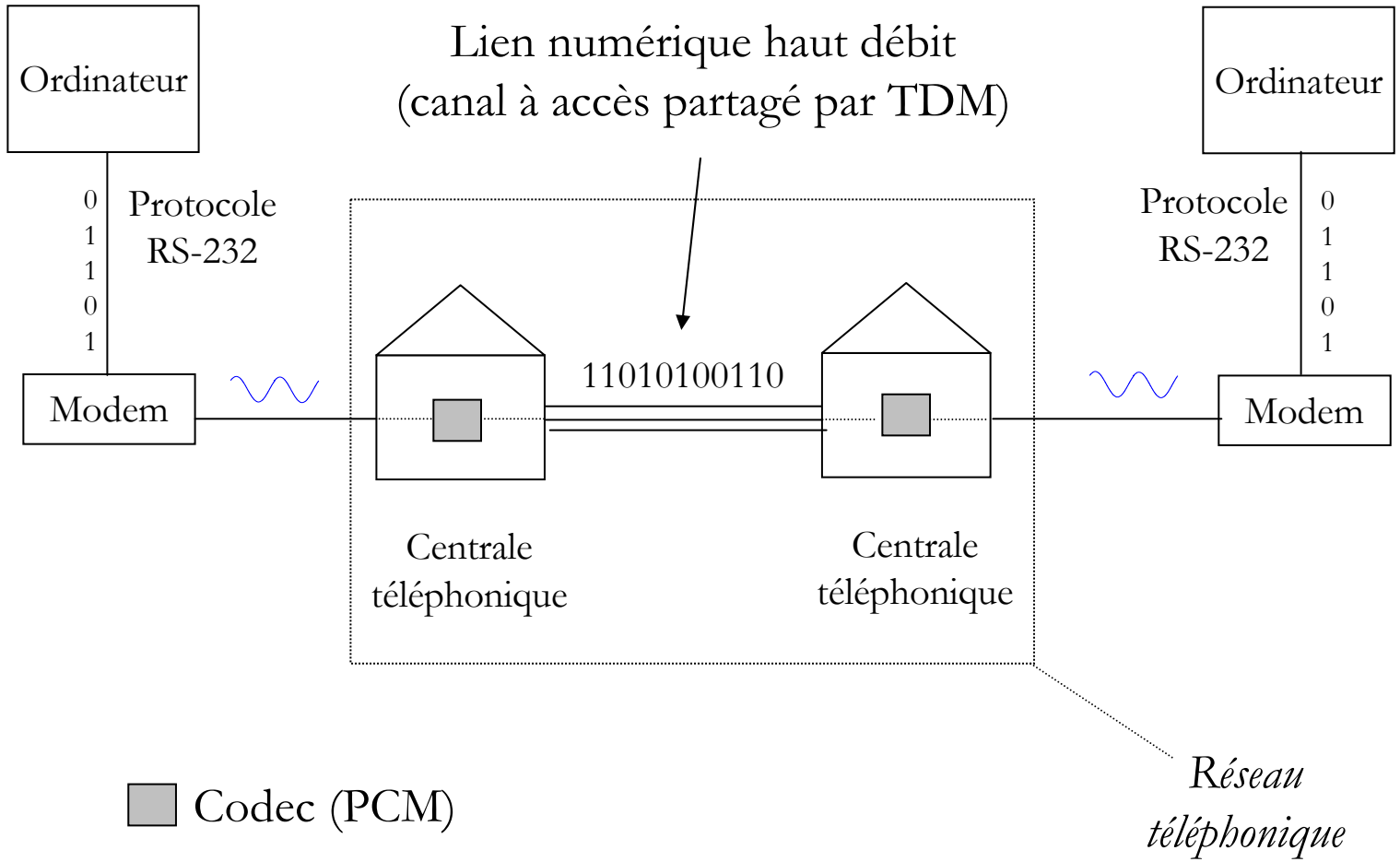


Centralisé



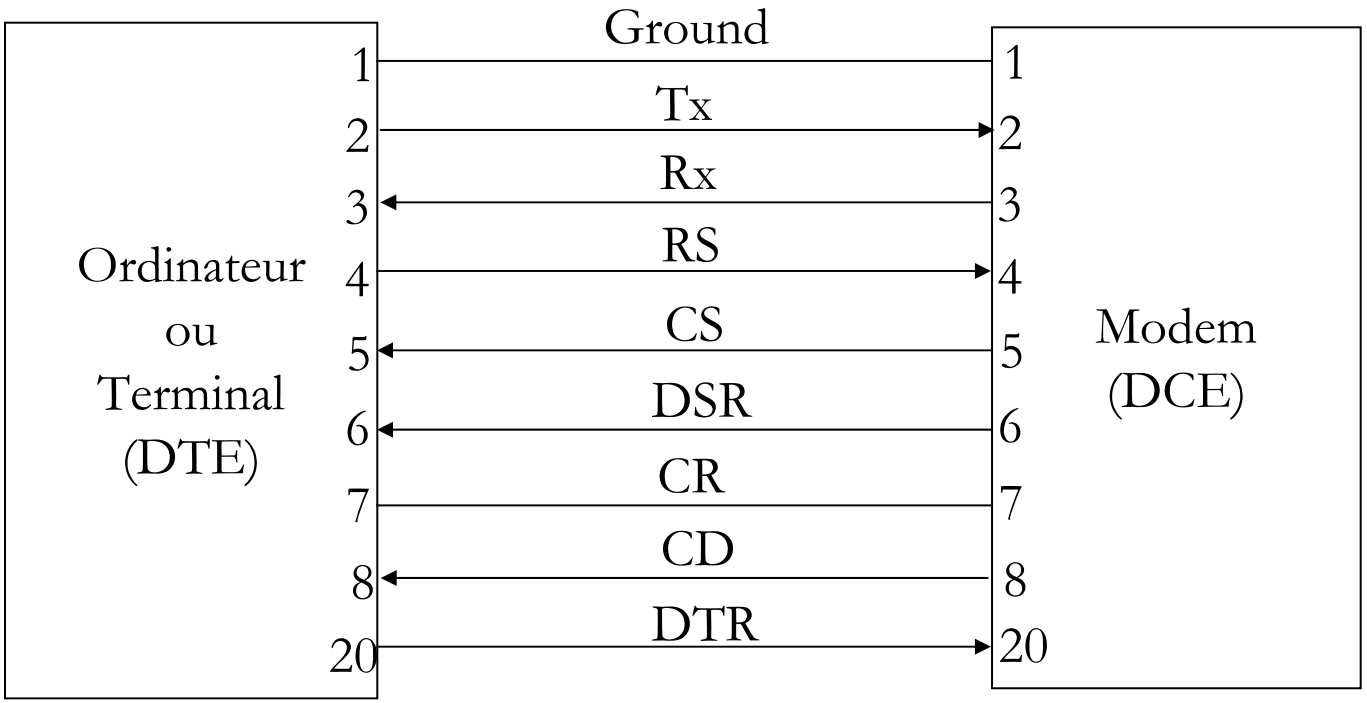
Hiérarchisé

Communication numérique sur le réseau téléphonique



RS-232-C

Interface standard entre ordinateur et modem (communication asynchrone orientée charactère)



RS-449 vs RS-232

RS-232	RS-449
Un seul mode - non balançé (un ground unique)	Deux modes - non balançé - balançé
Débit max = 20 kbit/s	Débit max = 2 Mbit/s
Distance max = 15 m	Distance max = 60 m

Les modems

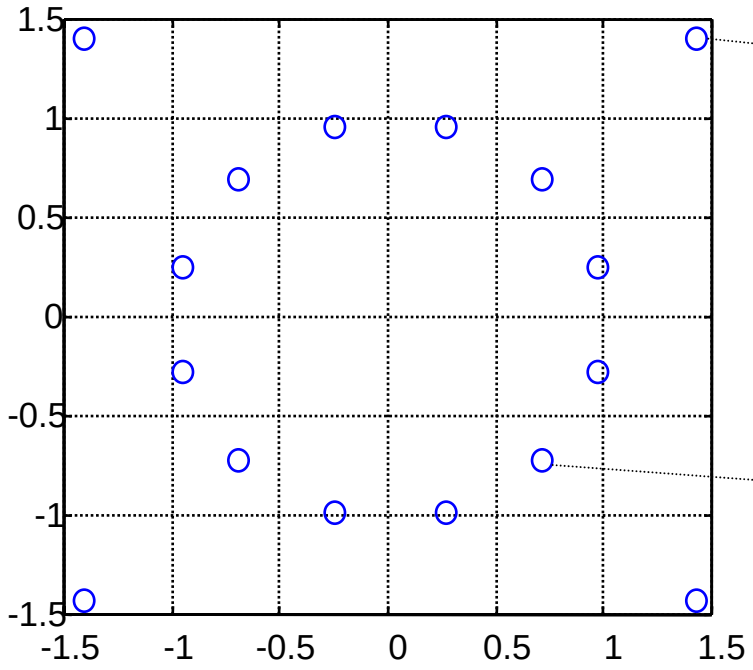
- Modulation d'un train binaire pour transmission sur canal analogique

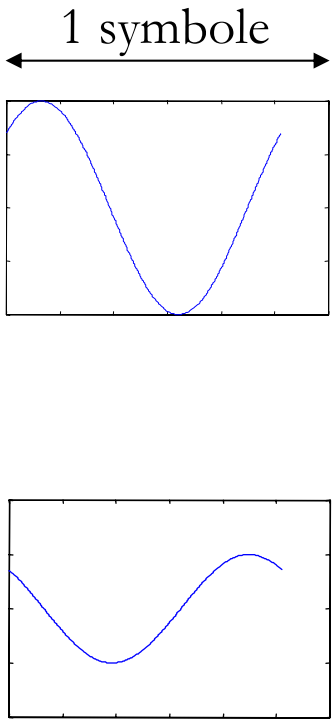
- Techniques principales de modulation:
  - Modulation d'amplitude
  - Modulation de fréquence
  - Modulation de phase

- Les modems haut débit utilisent une combinaison de ces techniques, pour coder *plusieurs bits par symbole*
  - Techniques complexes de démodulation: TCM

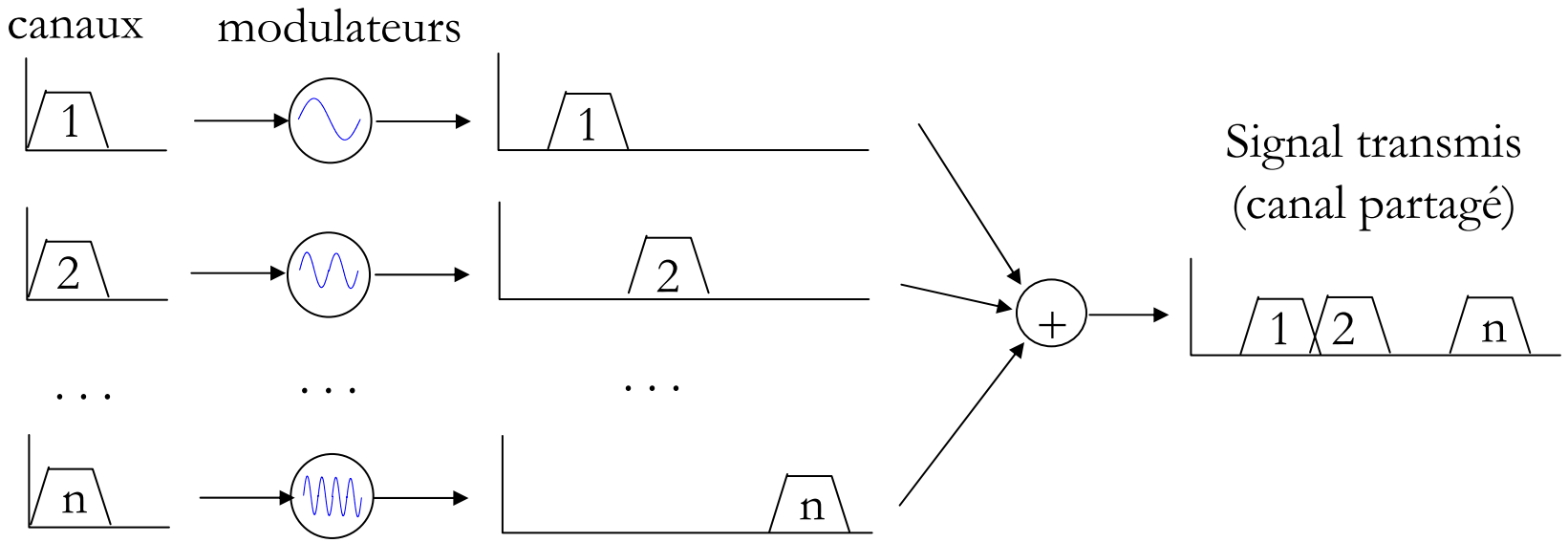
Exemple: QAM

Constellation de la norme ITU V.32 à 9600 bps



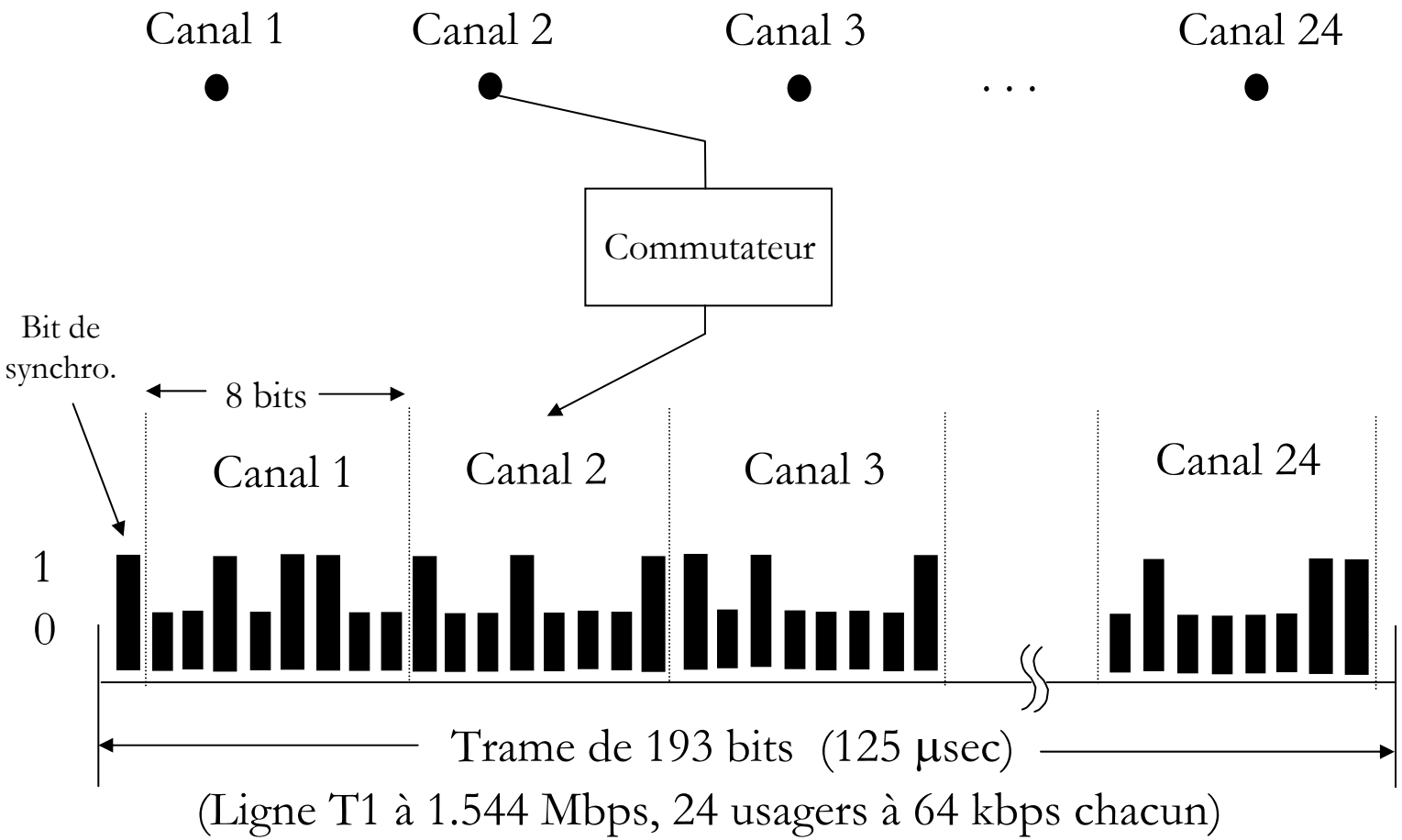


Multiplexage fréquentiel (FDM)



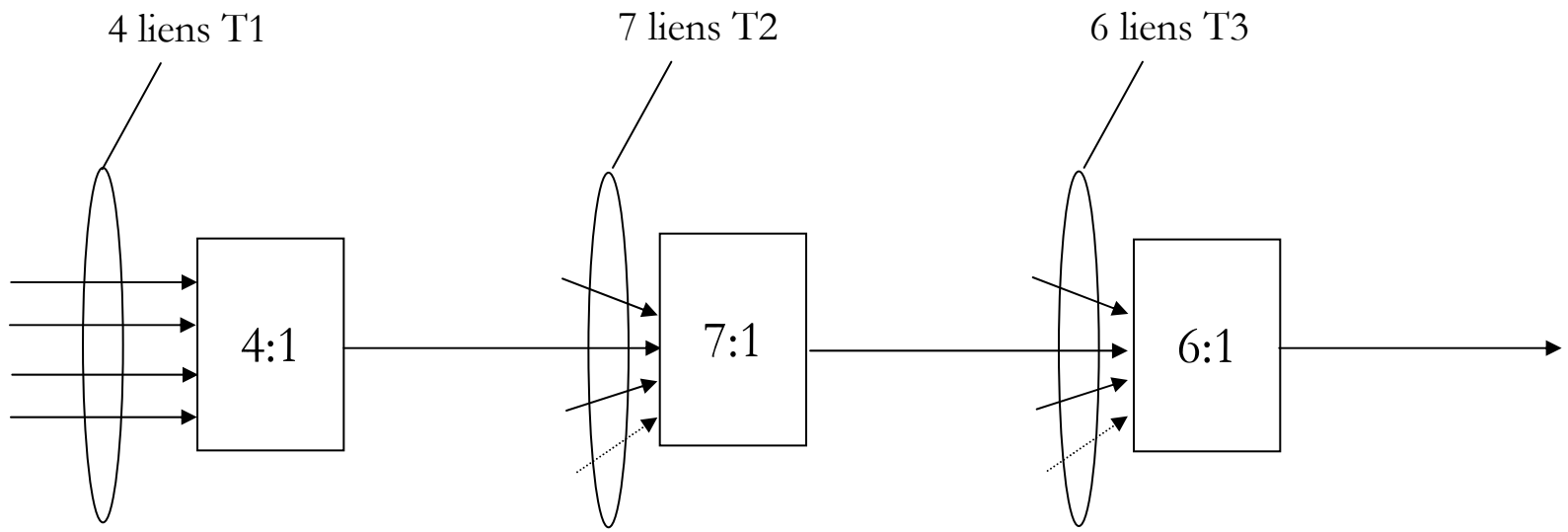
Multiplexage temporel (TDM)

(uniquement pour les signaux *numériques*)



(Ligne T1 à 1.544 Mbps, 24 usagers à 64 kbps chacun)

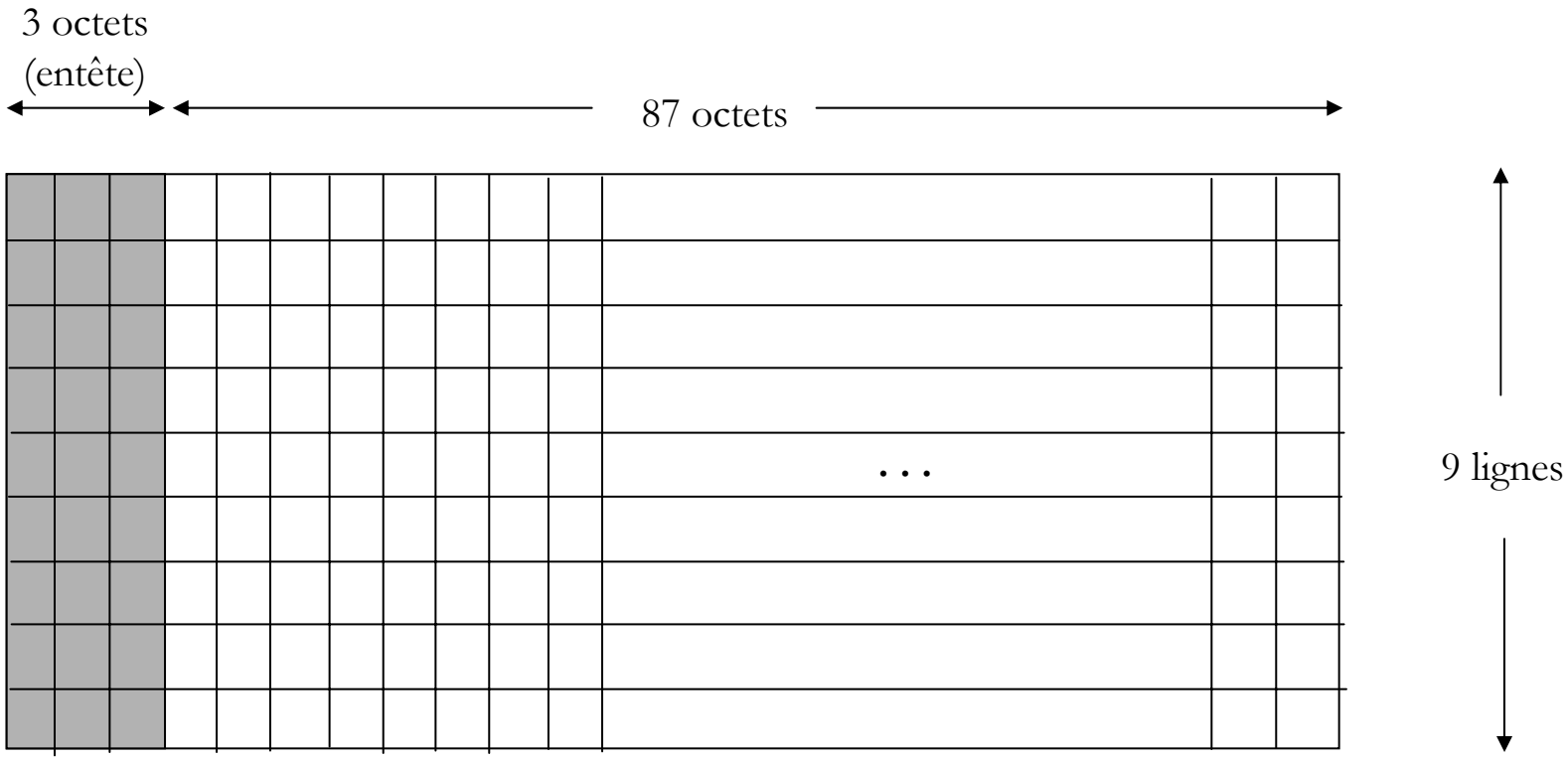
Hyérarchie PDH
(Plesiosynchronous Digital Hierarchy)



SONET

- Transmission numérique à haut débit sur *fibre optique*
- SONET = Synchronous Optical NETwork
  - **Hyérarchie Synchrone** (SDH)
  - Multiplexage temporel (TDM)
- Communications longues distances

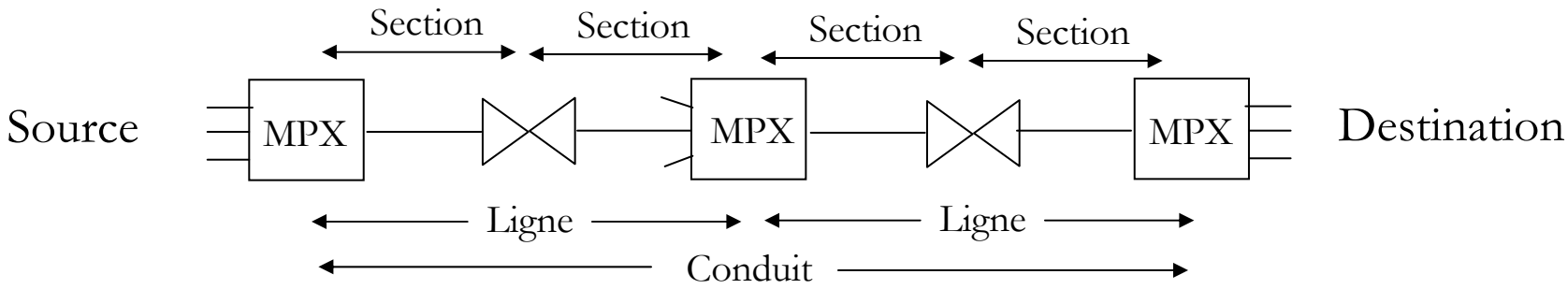
Trame SONET OC-1 (810 octets)

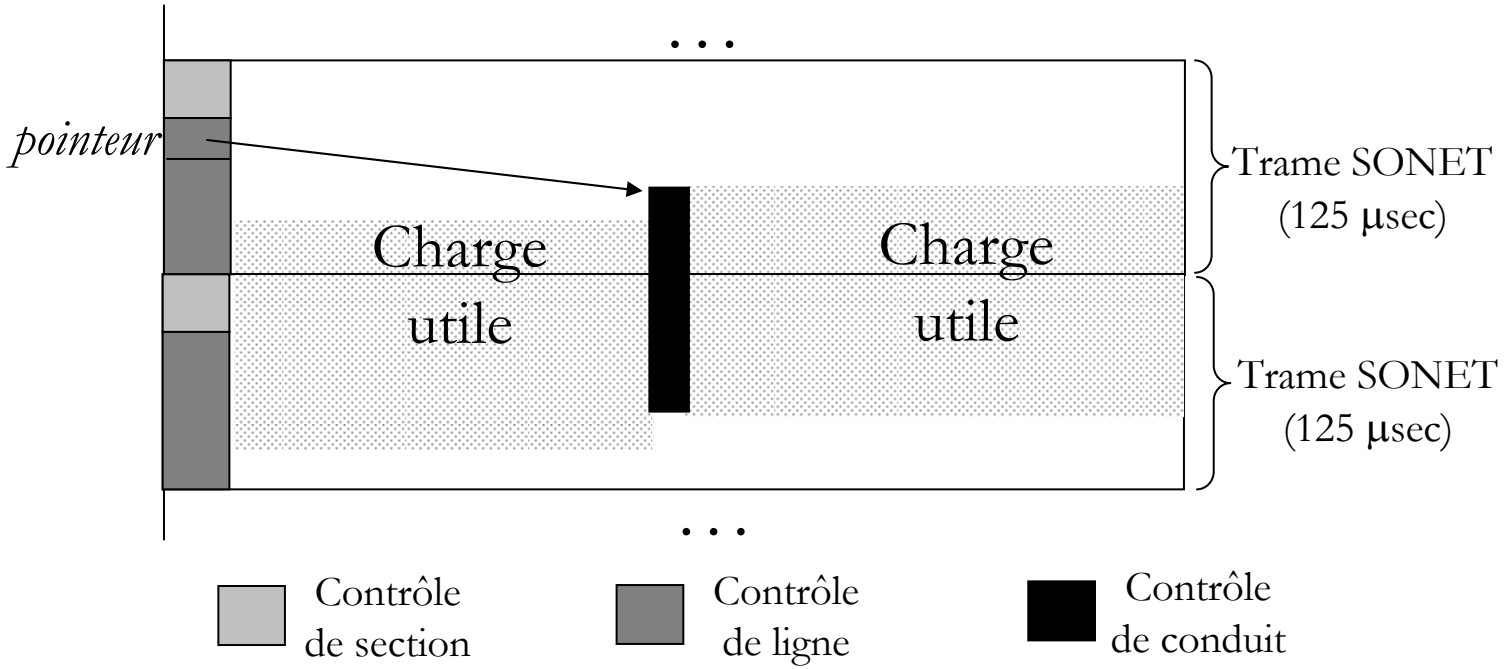


1 trame = 125 μsec (8000 trames/sec)

Débit = (90x9) octets * 8 bits/octet * 8000 trames/sec = **51.84 Mbps**

Transmission SONET





SONET: hiérarchie synchrone

- Hauts débits = multiples ENTIERS du débit de base
  - OC-1 $\rightarrow$ 51.84 Mb/s
  - OC-3 $\rightarrow$ 155.52 Mb/s
  - OC-12 $\rightarrow$ 622.08 Mb/s
- Multiplexage $\rightarrow$ entrelacement des colonnes
- ATM utilise le niveau OC-3

La commutation

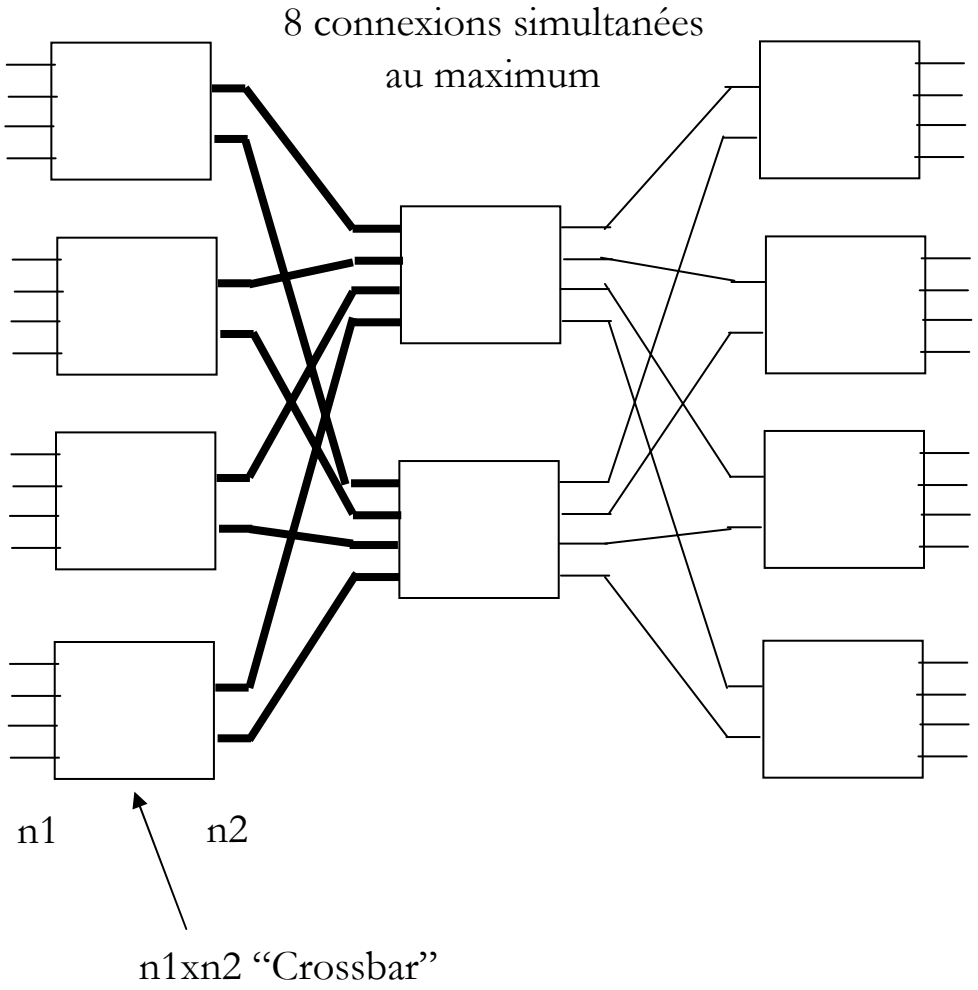
- Commutation par circuits

  - **connexion “physique”** établie à travers une séquence de centrales d’acheminement (“switching offices”)
  - **circuit dédié** pour la durée de la communication
  - aujourd’hui, la commutation est numérique, avec les signaux en format PCM

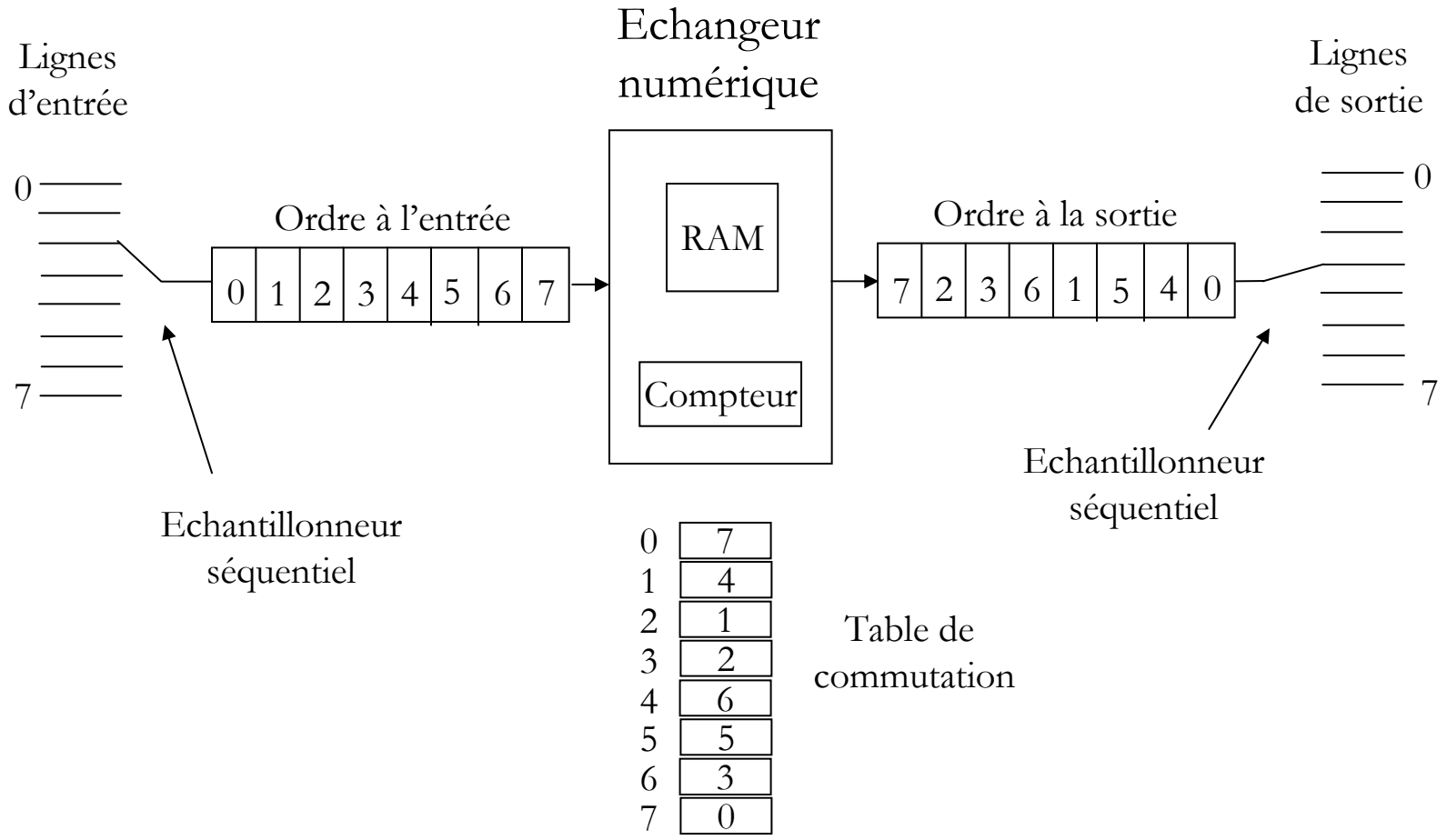
- Commutation par paquets

  - Acheminement à travers une séquence de **“routeurs”**
  - Approche **“store-and-forward”**

Commutateur à division spatiale



Commutateur à division temporelle



RNIS (ISDN)

- RNIS = Réseau Numérique à Intégration de Services
- Numérique de bout en bout
  - débit de base = 128 kb/s
- Commutation par circuits
- Types de services:
  - Intégration parole et données
  - Contrôle à distance
- Trop peu, trop tard ?...

B-ISDN

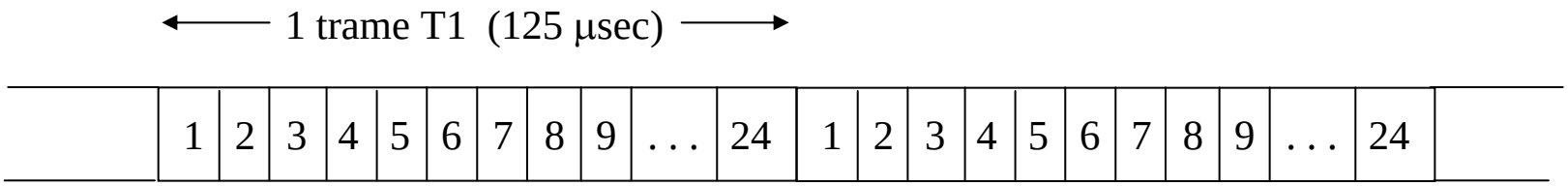
- B-ISDN = “Broadband-Integrated Services Digital Network”

- Commutation basée sur **circuits virtuels**
  - technologie ATM
  - compromis entre
    commutation par circuits et commutation par paquets

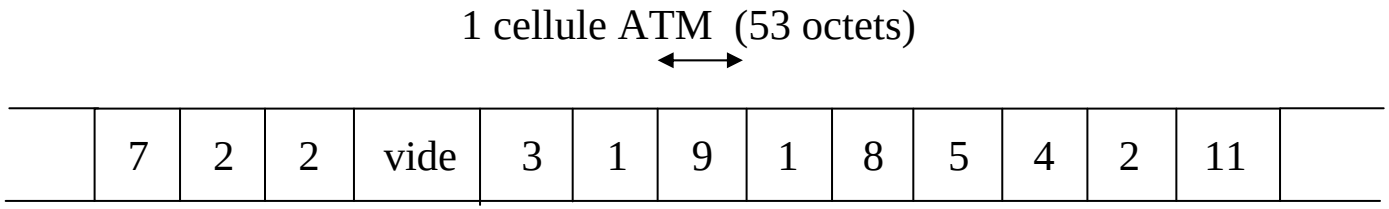
- Débit = 155 Mb/s (STS-3 ou OC-3)

- Demande des investissements importants
  - Presque tout le réseau téléphonique actuel à modifier

ATM: transmission asynchrone



Transmission (multiplexage) synchrone



Transmission (multiplexage) asynchrone → ATM

Commutateurs ATM

- Acheminement à bas niveau (couche physique)

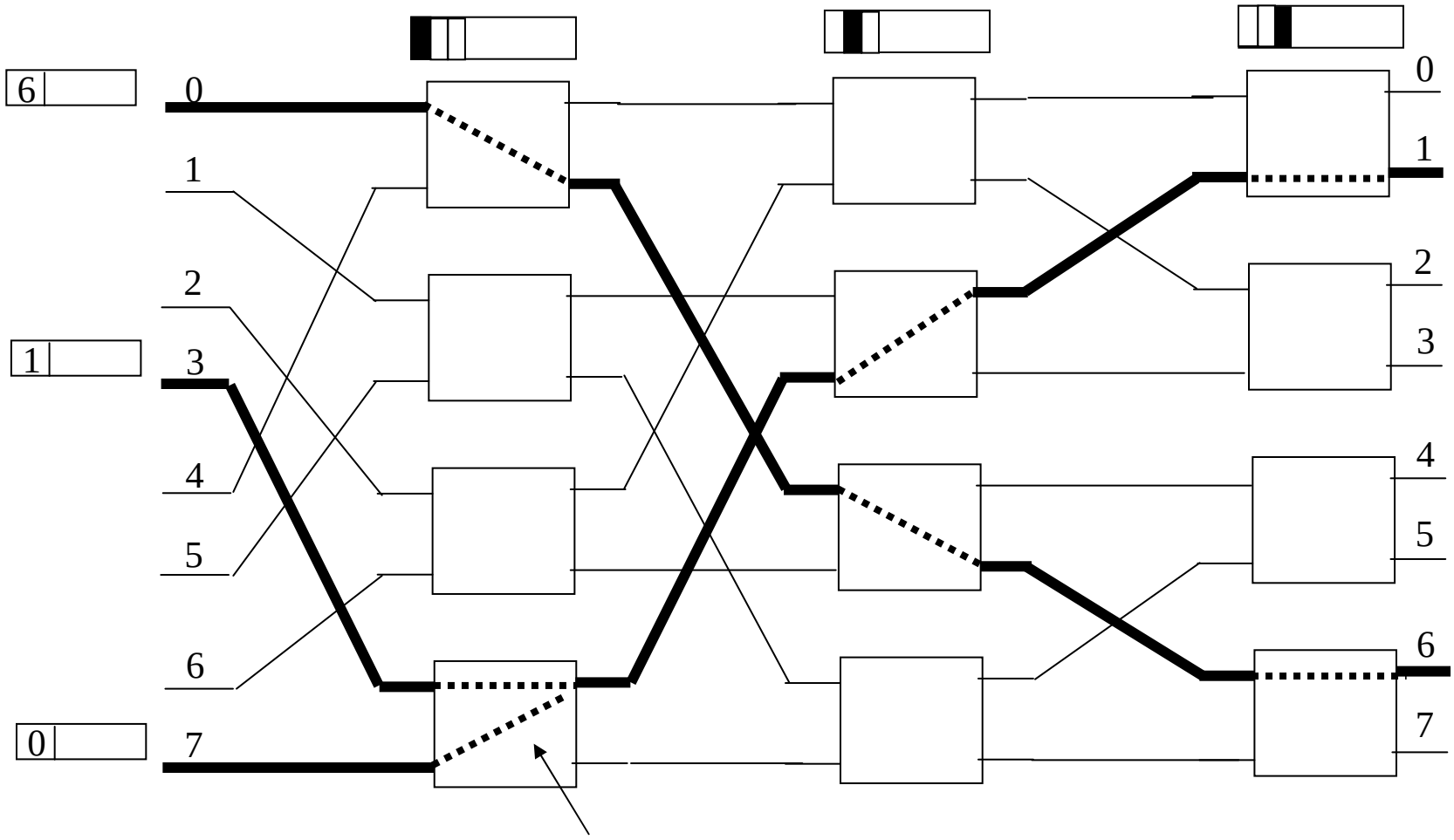
- Commutation très rapide
  - cellules **petites** (53 octets) et de **taille fixe**

- Contraintes de qualité:

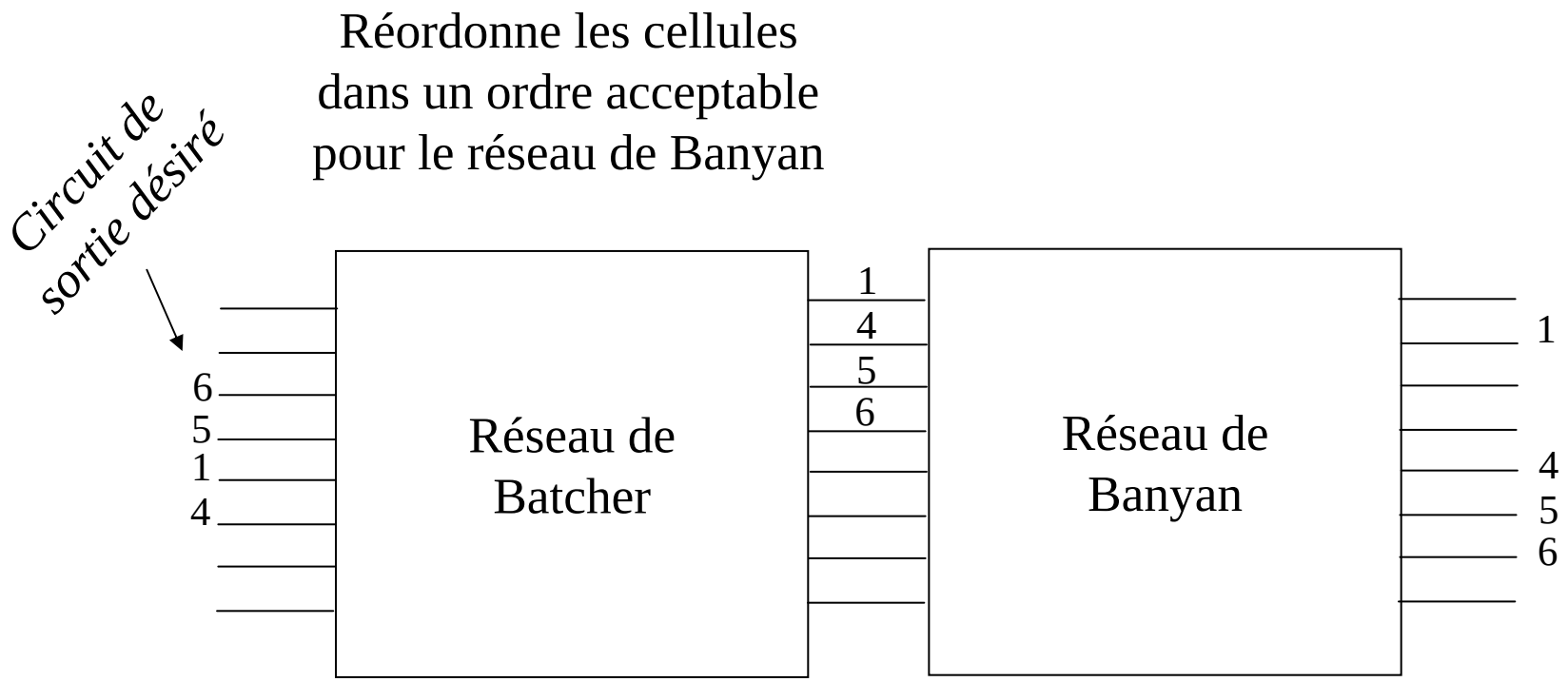
  (1) taux de pertes minime
  - ex.: 1 ou 2 cellules rejetées à l'heure…
  - files d'attentes pour les contingences

  (2) Préserver l'ordre des cellules sur un CV

Le commutateur Banyan

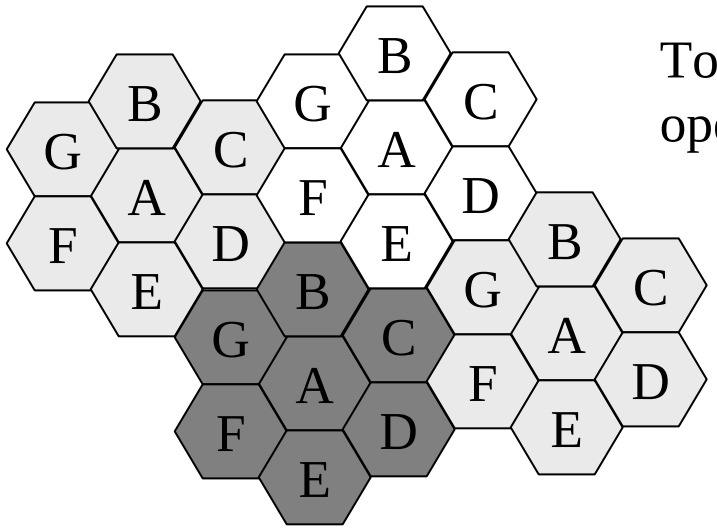


Batcher-Banyan
(solution aux collisions)



Réseaux cellulaires

- Inventés chez Bell Labs (AMPS, analogique)
- Cellules: réutilisation des fréquences
  → augmentation de la capacité du réseau



Toutes les cellules de même “nom” opèrent aux mêmes fréquences

Changement de cellule
= changement de fréquence
(“handoff”)

Réseaux cellulaires numériques

- 1ère génération = réseaux analogiques (AMPS)
- 2ème génération = réseaux numériques
- Aux Etats-Unis:
  IS-54 and IS-135 $\rightarrow$ compatibles avec le mode analogique
  IS-95 $\rightarrow$ CDMA (voir chapitre 4)
- En Europe:
  GSM $\rightarrow$ basé sur FDM *et* TDM (voir chapitre 4)

- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

La couche liaison de données

- Assure un service **fiable** de transmission de données entre deux noeuds typiquement sur le **même réseau local**. (“physiquement” reliés)
- Décompose et transmet les données en **trames** qui sont transmises séquentiellement (confiées à la couche physique)
- Processus d’**acquiescements** entre récepteur et émetteur
- Possibilité de **retransmission** d’une trame erronée
- Ajustement du débit en fonction de la capacité du récepteur

Problème 1, page 239 (Tanenbaum)

- Probabilité qu'une trame traverse sans erreur :

  $p_0 = 0.8$

- Probabilité que les **10 trames** du message traversent sans erreur:

  $1\text{-}p = 0.8^{10} = 0.1074$

  ($p = 0.8926$ est donc la probabilité qu'il faudra retransmettre le message -- i.e. encore une fois les 10 trames…)

- Nombre moyen de retransmissions (voir prob. 14, page 75);

  $N = 1 / (1 - p) = 9.31$

  (plus de 9 fois, en moyenne)

Problème 1, page 239 (suite)

- Combien de transmissions de trames seraient requises *au total* si les ACK se situaient à ce niveau (et non au niveau du message complet) ?

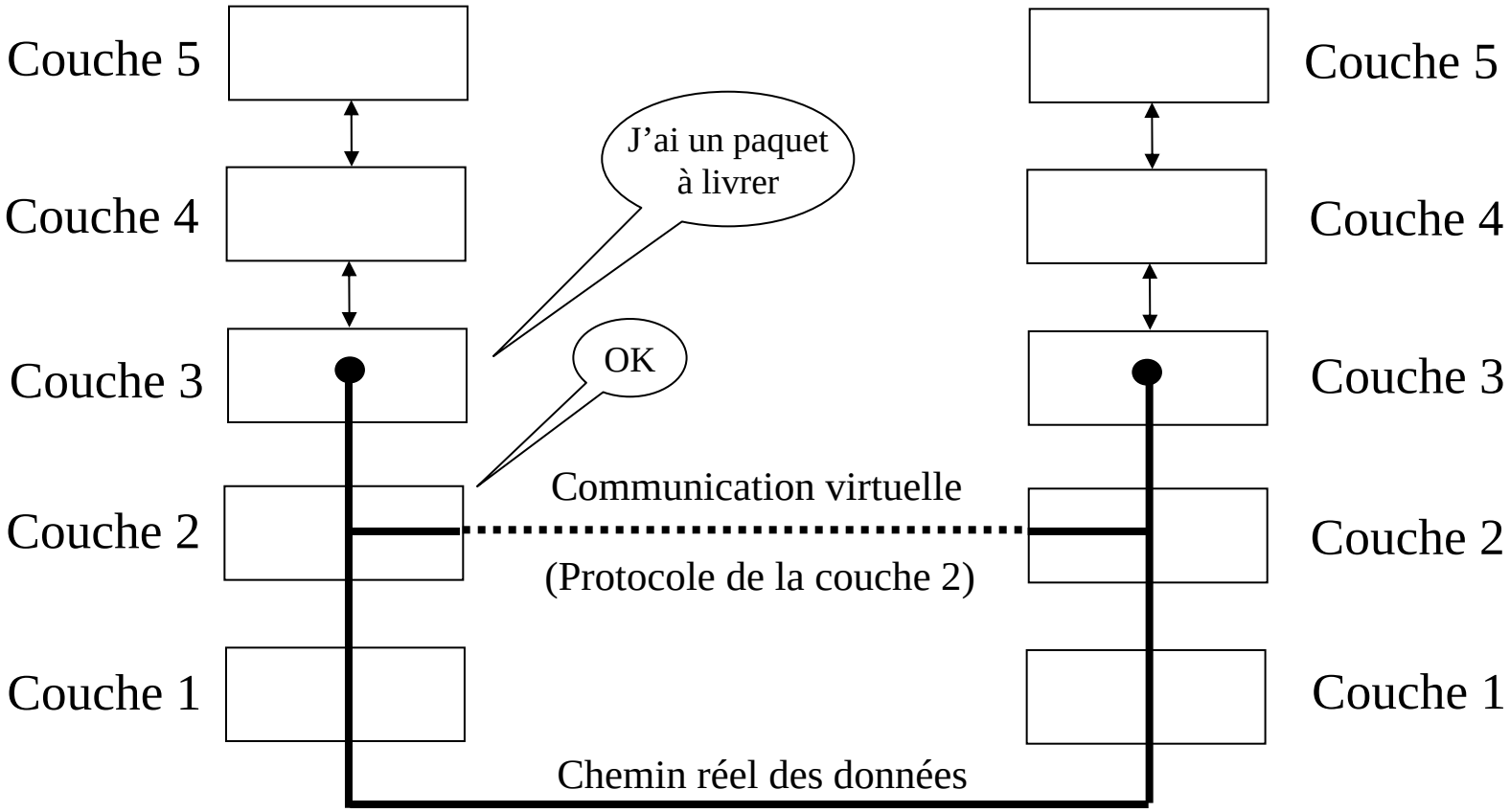
1- p_0 = 0.2 est la probabilité qu'une trame soit retransmise

Pour un message de 10 trames, on peut donc dire (en gros):

- 8 trames seront transmises avec succès
- 2 trames devront être retransmises une deuxième fois

$\rightarrow$ au total : 12 transmissions
(i.e. 1.2 fois le message, au lieu de 9.31 …)

Communication virtuelle



Services

- Sans connexion, sans aquiescement
  - pas de guarantie sur l'ordre et contre les copies multiples
  - pas de guarantie de livraison des trames
  - utilisé sur la plupart des *LAN* (peu d'erreurs)
  - la couche transport s'oocupera des reprises ...

- Sans connexion, avec aquiescement
  - toujours pas de guarantie sur l'ordre et copies, mais
  - guarantie de livraison (retransmission si pas d'ACK)
  - utilisé sur les *transmissions sans fils* (peu fiables)

- Avec connexion, avec aquiescement
  - emule un circuit physique ("*bitstream pipe*")

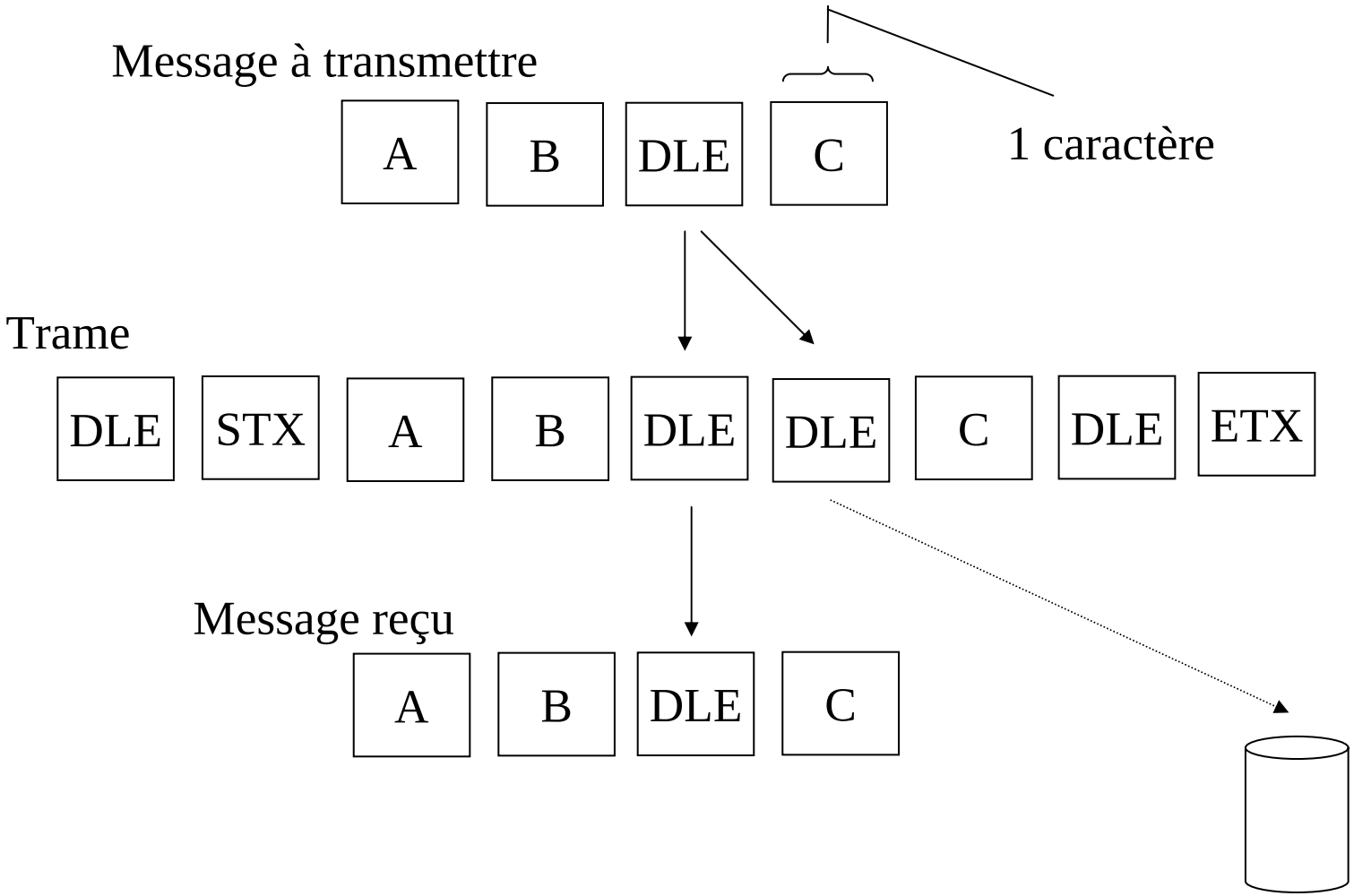
Découpage en TRAMES

- Permet de détecter des erreurs (bits de parité)

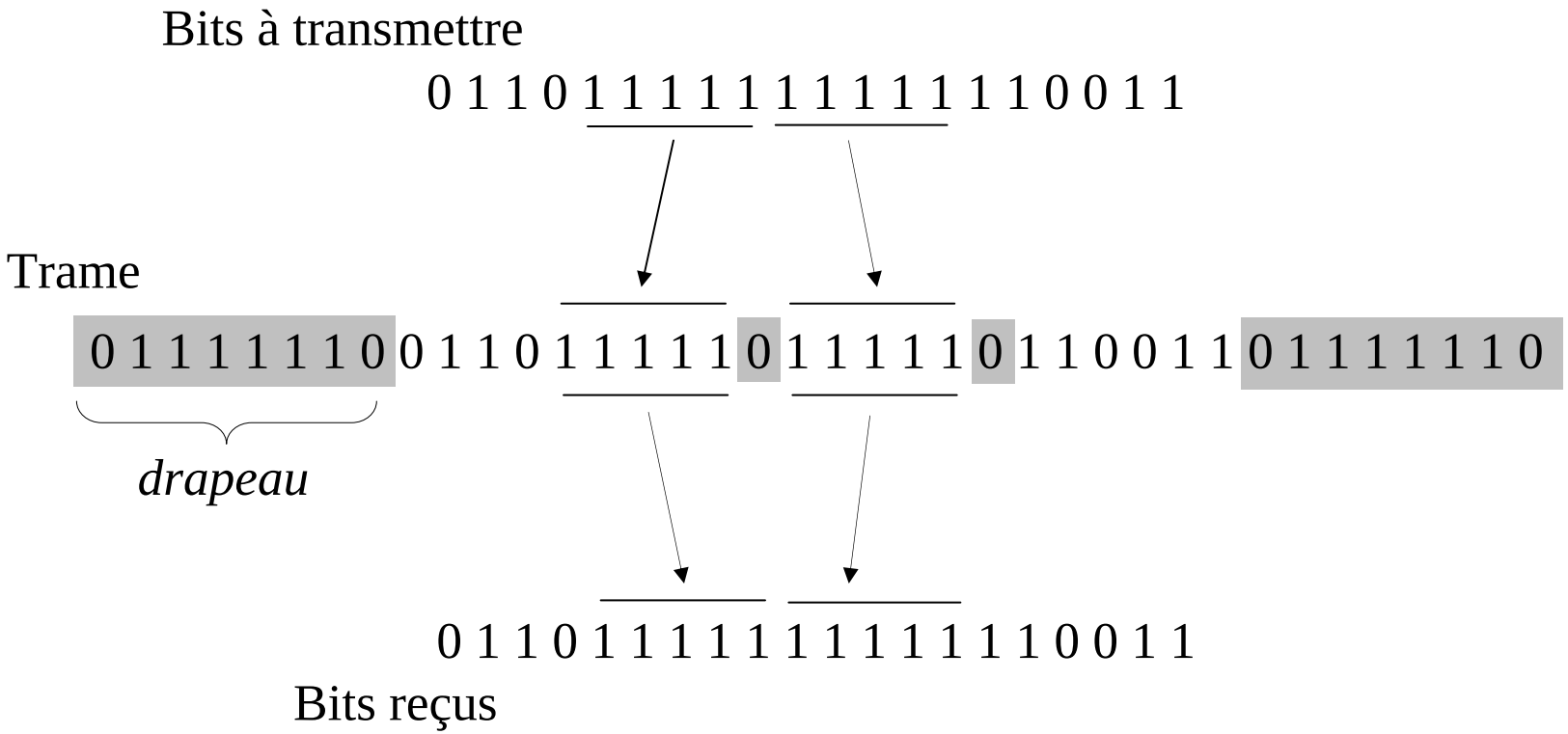
- Plusieurs approches
    1) entête (indicateur de taille) plus données
    2) délimiteurs de début et fin
        → attention si les données contiennent par accident le délimiteur
            - "character stuffing"
            - "bit stuffing"
    3) Utilisation des conventions de la couche physique

- En pratique, on utilise (1) avec (2) ou (3)

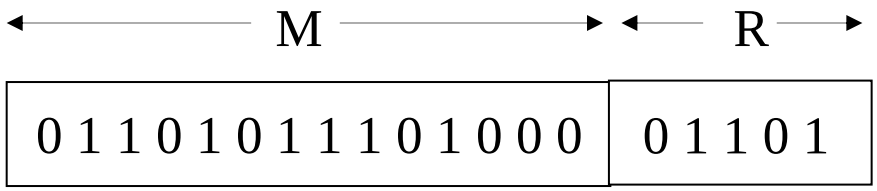
“Character stuffing”



“Bit stuffing”



Récupération des erreurs



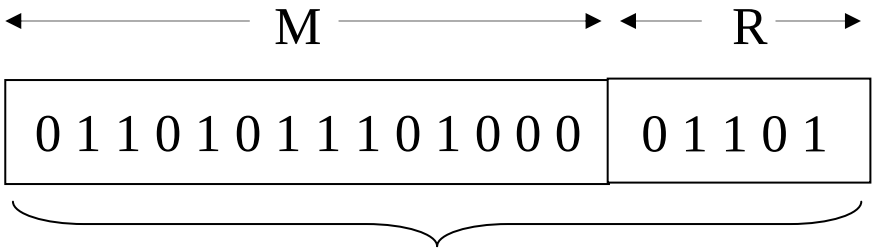
M = *m* bits du message
R = *r* bits de redondance

- Deux approches principales :
  (1) codes de *correction* d'erreurs
  (2) codes de *détection* d'erreurs

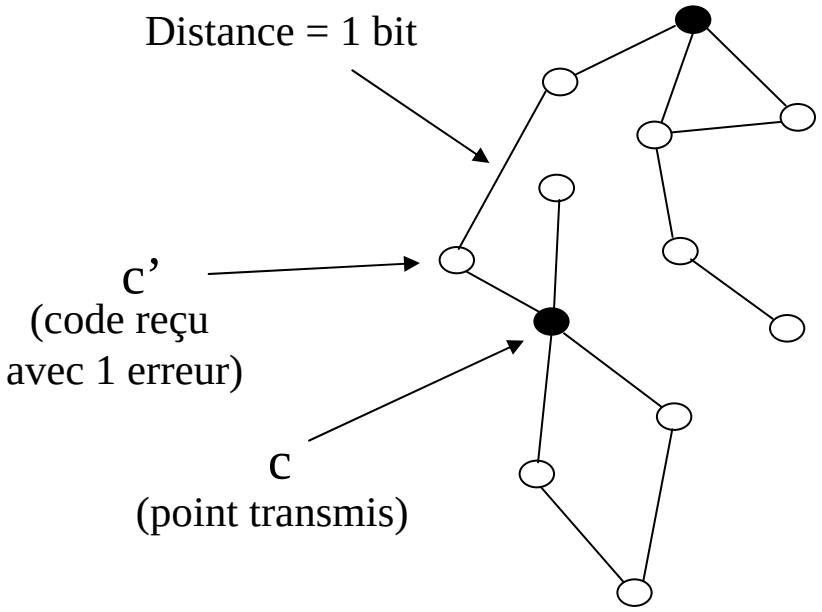
En géréral, l'approche (1) introduit davantage de redondance
→ diminue le débit utile du canal ...

(1) Correction des erreurs

- mot de code valide
- point dans l'espace



Code transmis = c
un point en (m+r) dimensions

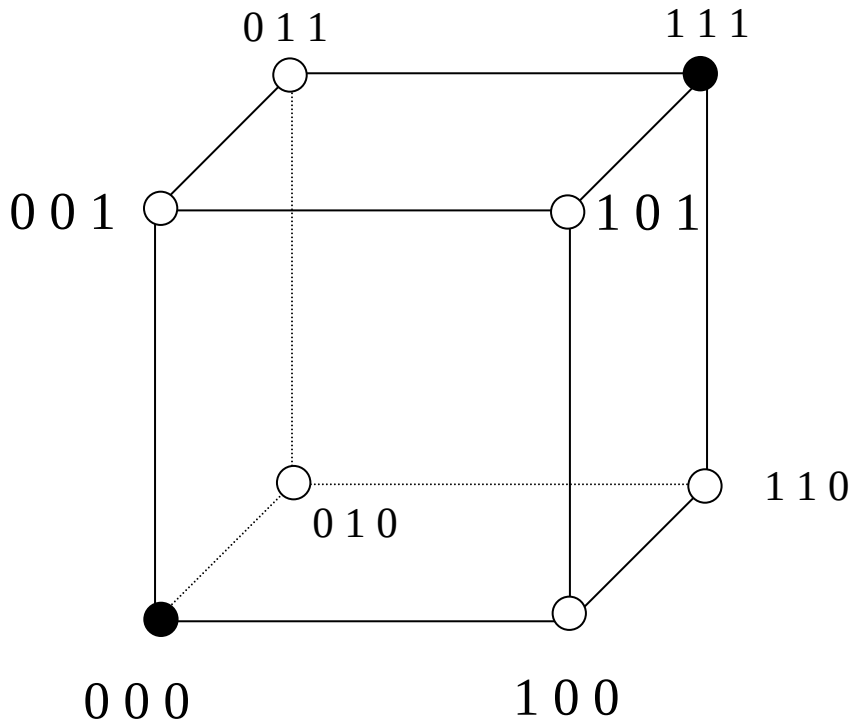


Au décodeur:
c est toujours le mot de code
le plus près du mot reçu **c'**
$\rightarrow$ on décode **c** sans erreur ...

Exemple: code à répétition

M	R
0	0 0
1	1 1

Peut corriger 1 erreur
Et détecter 2 erreurs ...



Code de Hamming

- Code transmis = Message + Bits de parité
  → parité: certaines combinaisons des bits d'info

- Correction d'*une* erreur

- Au décodeur
  → Calcul du *syndrôme* (S) en arithmétique modulo-2

- Position de l'erreur
  → Valeur du syndrôme S
  → Aucune erreur si S = 0

(2) Détection des erreurs

- Permet de vérifier l'intégrité d'une trame au récepteur
- Retransmission des trames corrompues
- Plus efficace que la correction
  → requiert moins de bits de redondance
- **CRC** : "Cyclic Redundancy Code"
  → codes de détection couramment utilisés

CRC

- 1 bit = coefficient d'un polynôme en x :

$$1\ 0\ 0\ 1\ 1 \quad = \quad \mathbf{1}\, x^4 + \mathbf{0}\, x^3 + \mathbf{0}\, x^2 + \mathbf{1}\, x^1 + \mathbf{1}\, x^0$$
$$= \quad x^4 + x + 1$$

- Arithmétique modulo-2, sans retenue : $1+0 = 0+1 = 1$
  $0+0 = 1+1 = 0$

- Polynôme générateur $G(x)$
  $\rightarrow$ r+1 bits, où r est le nombre de bits de parité

- R = reste de la division de $2^r M$ par G
  = bits du CRC (parité, ou "checksum")

CRC: un exemple

Message M = 1 1 0 0 1 1

= $1\,x^5 + 1\,x^4 + 0\,x^3 + 0\,x^2 + 1\,x^1 + 1\,x^0$

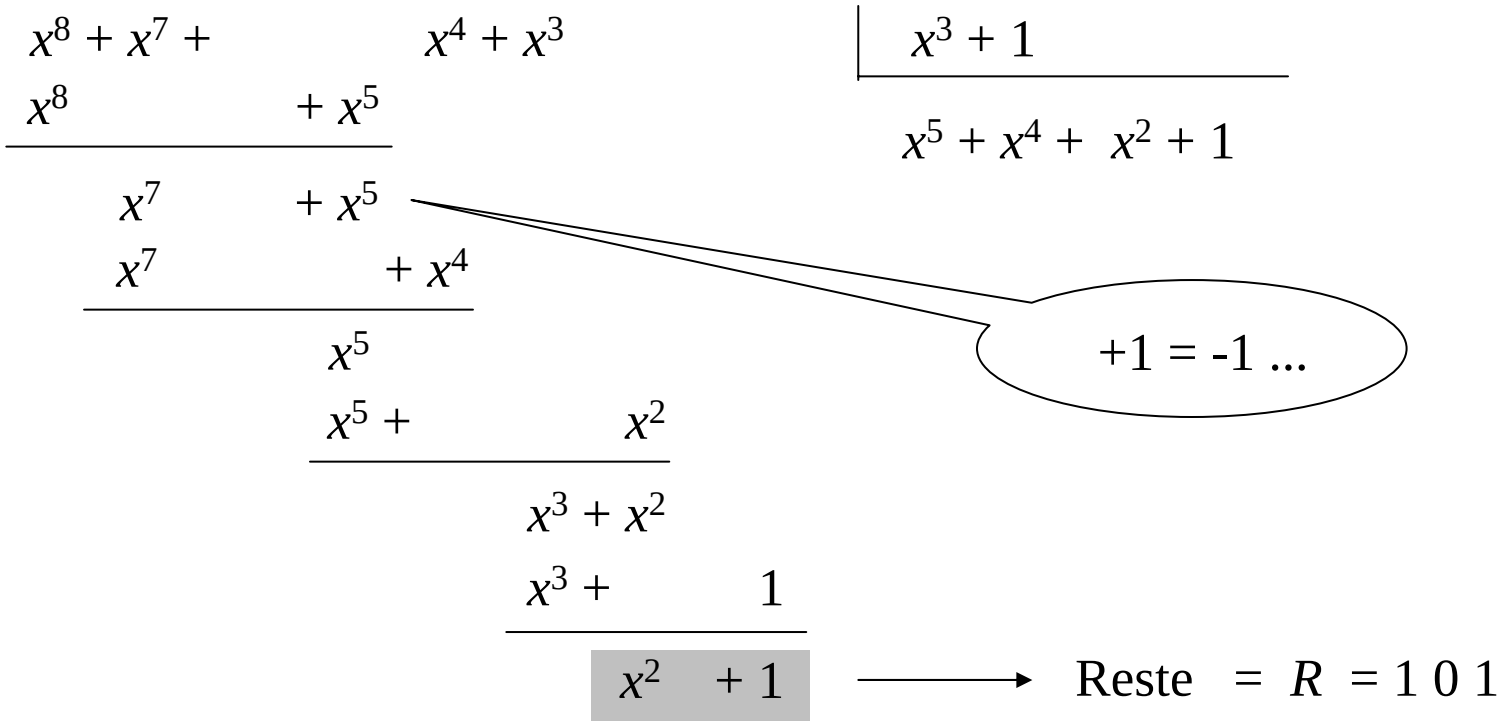
= $x^5 + x^4 + x + 1$

Polynôme G = 1 0 0 1 = $x^3 + 1$ $\rightarrow$ 4 bits (donc r = 3 bits)

2^rM = $x^8 + x^7 + x^4 + x^3$

Il faut donc diviser 2^rM par G. $\rightarrow$ page suivante

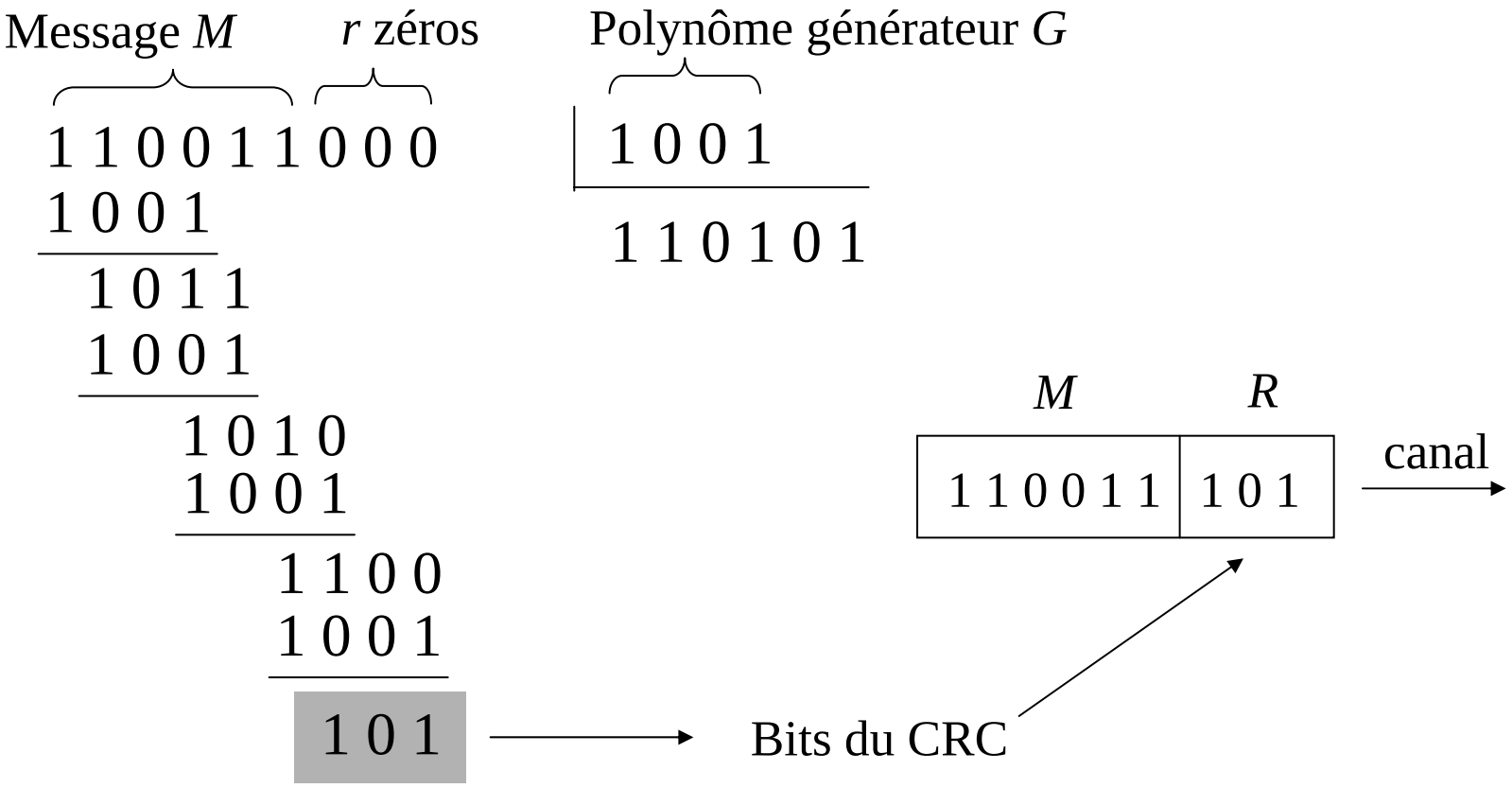
Calcul du CRC



Reste = R = 1 0 1

Ce sont les bits de parité (CRC)

CRC: calcul rapide



CRC: propriétés

- Peut *corriger* 1 erreur isolée
- Peut détecter 2 erreurs (certaines conditions sur G)
- Peut détecter *tous les patrons d'erreurs impairs*
  $\rightarrow$ si $(x + 1)$ est un facteur de $G(x)$
- Peut détecter r erreurs consécutives ("bursts")
- Trois polynômes standards :

CRC-12	:	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
CRC-16	:	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	:	$x^{16} + x^{12} + x^5 + 1$

Protocoles de liaison de données

Protocole unidir. le plus simple

```
typedef enum {data, ack, nak} frame_kind;
typedef  struct {
   frame_kind kind;
   seq_nr seq;
   seq_nr ack;
   packet info;
}  frame;
```

Emetteur

```
void sender(void)
{
   frame s;
   packet buffer;

   while (true)
   {
      from_network_layer(&buffer);
      s.info  =  buffer;
      to_physical_layer(&s);
   }
}
```

Récepteur

```
void  receiver(void)
{
   frame r;
   event_type event;

   while (true)
   {
      wait_for_event(&event);
      from_physical_layer(&r);
      to_network_layer(&r.info);
   }
}
```

Hypothèses du protocole simple

- La couche réseau à l'émetteur a toujours des paquets à transmettre
- La couche réseau au récepteur est toujours prête à recevoir un paquet
  → On suppose une mémoire infinie ...
- On ignore le temps (CPU) de traitement des paquets
- Aucunes erreurs sur le canal, aucune trame perdue

  → Protocole très peu réaliste ...

Protocole “stop-and-wait”

Emetteur

```
void sender(void)
{
   frame s;
   packet buffer;
   event_type event;

   while (true)
   {
      from_network_layer(&buffer);
      s.info  =  buffer;
      to_physical_layer(&s);
      wait_for_event(&event);
   }
}
```

Récepteur

```
void  receiver(void)
{
   frame r, s;
   event_type event;

   while (true)
   {
      wait_for_event(&event);
      from_physical_layer(&r);
      to_network_layer(&r.info);
      to_physical_layer(&s);
   }
}
```

① Seulement si event = SEND_NEXT_PAQUET

Limites du protocole “stop-and-wait”

- Ne prend pas en compte les erreurs de canal
  $\rightarrow$ une trame perdue = bloqué en wait_for_event( )

- Ajout d’un temporisateur (“timer”)
  $\rightarrow$ élimine les blocages (retransmission avant $t = \infty$)
  $\rightarrow$ MAIS possibilité de trames multiples au récepteur !

- Il faut donc numéroter les trames …

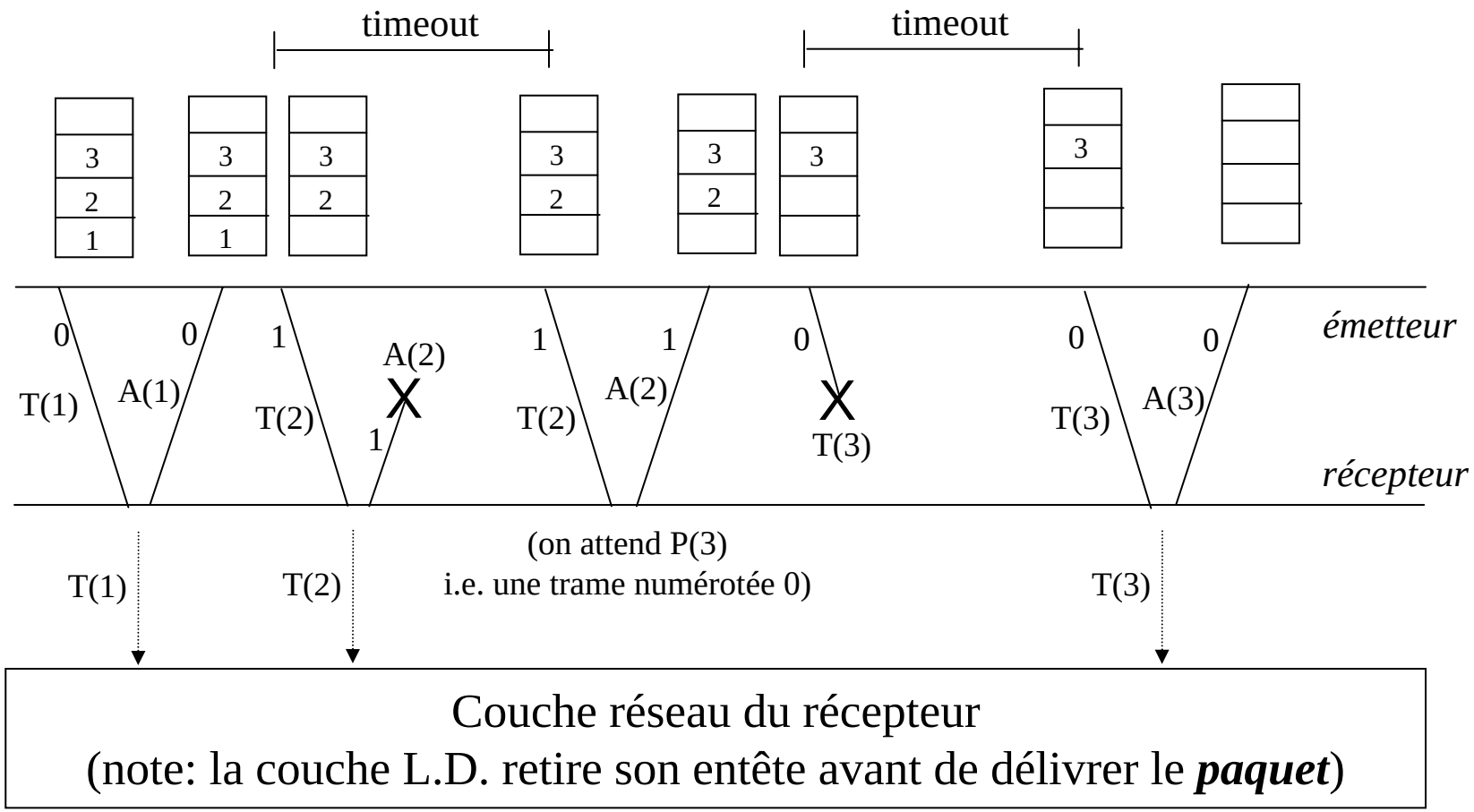
- Solution: protocoles ARQ, ABP, GO BACK N, SRP

Protocole ARQ

numérotation à 1 bit

- Numérotation des trames modulo-2 (0,1)
- Numérotation d'un acquiescement (ACK) modulo-2
  $\rightarrow$ même numéro que la trame reçue
- Utilisation d'un temporisateur
  $\rightarrow$ reprise si ACK perdu
- Seules les trames avec le numéro attendu sont livrées à la couche réseau du récepteur
- Voir le code à la page 201 de Tanenbaum

ARQ: séquence d'évènements



Protocoles à fenêtre glissante

- Plus d'une trame non-acquiesquées sont transmises
- Transmission bi-directionnelle ("full-duplex")
- Une trame contient à la fois :
  - un paquet pour transmission dans un sens
  - un ACK d'un autre paquet (transmission inverse)
  - introduit un léger délai pour les ACKs
    (doit attendre la prochaine trame)
  - ACK transmis seul si la prochaine trame tarde trop
- Requiert très peu de bits pour les ACK ("***piggybacking***")

ABP: fenêtre glissante à 1 bit

```
void ABP_protocol(void)
{
   seq_nr  next_frame_to_send;
   seq_nr  frame_expected;
   frame   r, s;
   packet  buffer;
   event_type  event;

   next_frame_to_send  = 0;
   frame_expected  = 0;
   from_network_layer(&buffer);
   s.info  =  buffer;
   s.seq   =  next_frame_to_send;
   s.ack   =  1 - frame_expected;
   to_physical_layer(&s);
   start_timer(s.seq);

//  suite à la deuxième colonne
```

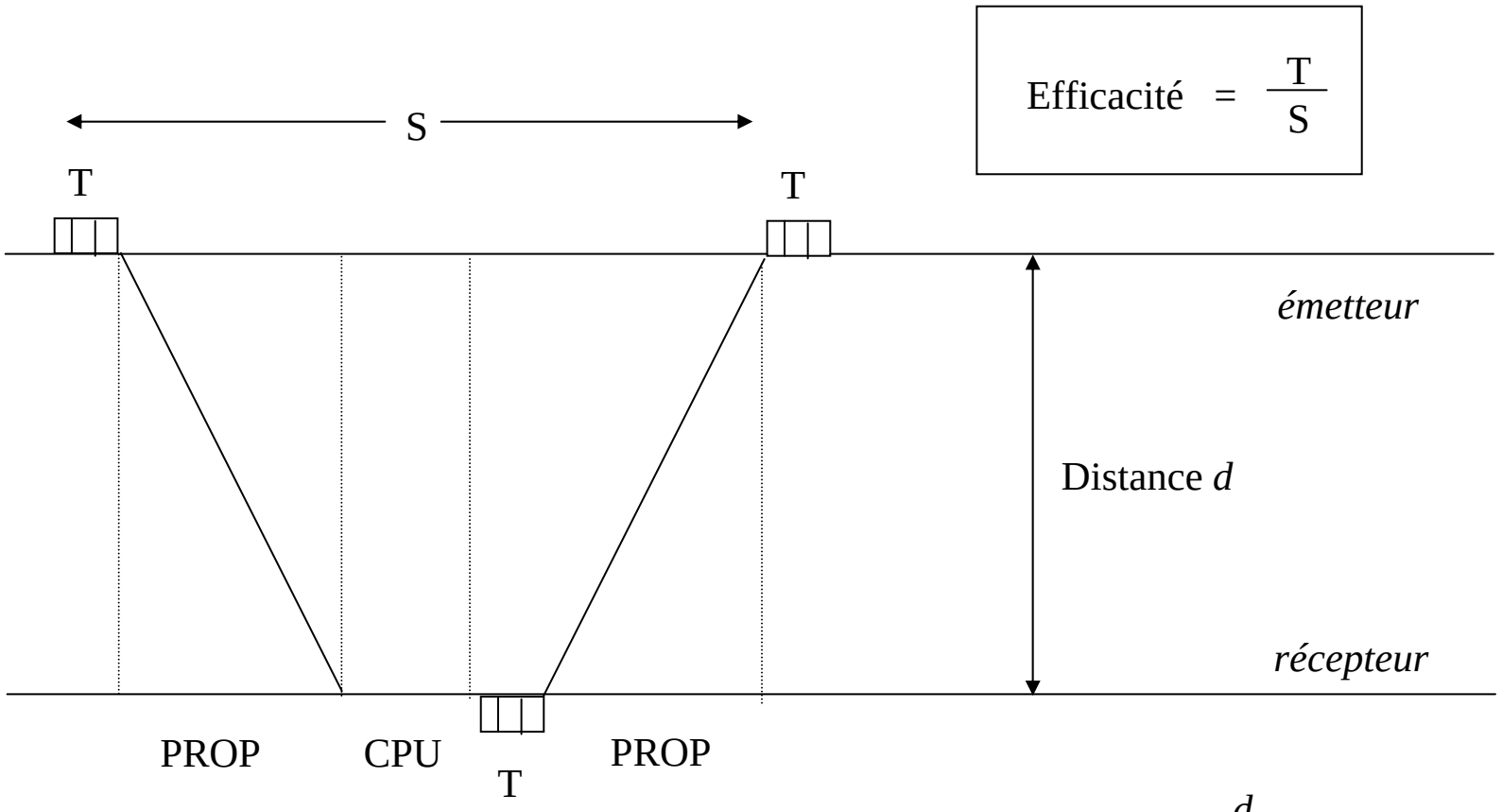
```
// suite de ABP_protocol

   while (true)
   {
      wait_for_event(&event);
      if (event == frame_arrival)
      {
        from_physical_layer(&r);
        if (r.seq == frame_expected)
        {
           to_network_layer(&r.info);
           inc(frame_expected);
        }
        if (r.ack == next_frame_to_send)
        {
           from_network_layer(&buffer);
           inc(next_frame_to_send);
        }
      }
      s.info = buffer;
      s.seq = next_frame_to_send;
      s.ack = 1 - frame_expected;
      to_physical_layer(&s);
      start_timer(s.seq);
   }
}  // fin de ABP_protocol
```

ABP: séquence d'événements

- Identique à ARQ
- Différence: les ACK sont transmis dans l'entête des trames transmises en sens inverse ("piggybacking")
- Protocole à efficacité réduite si
  - le débit est élevé
  - le temps de propagation (distance) est élevé
  - les trames sont courtes

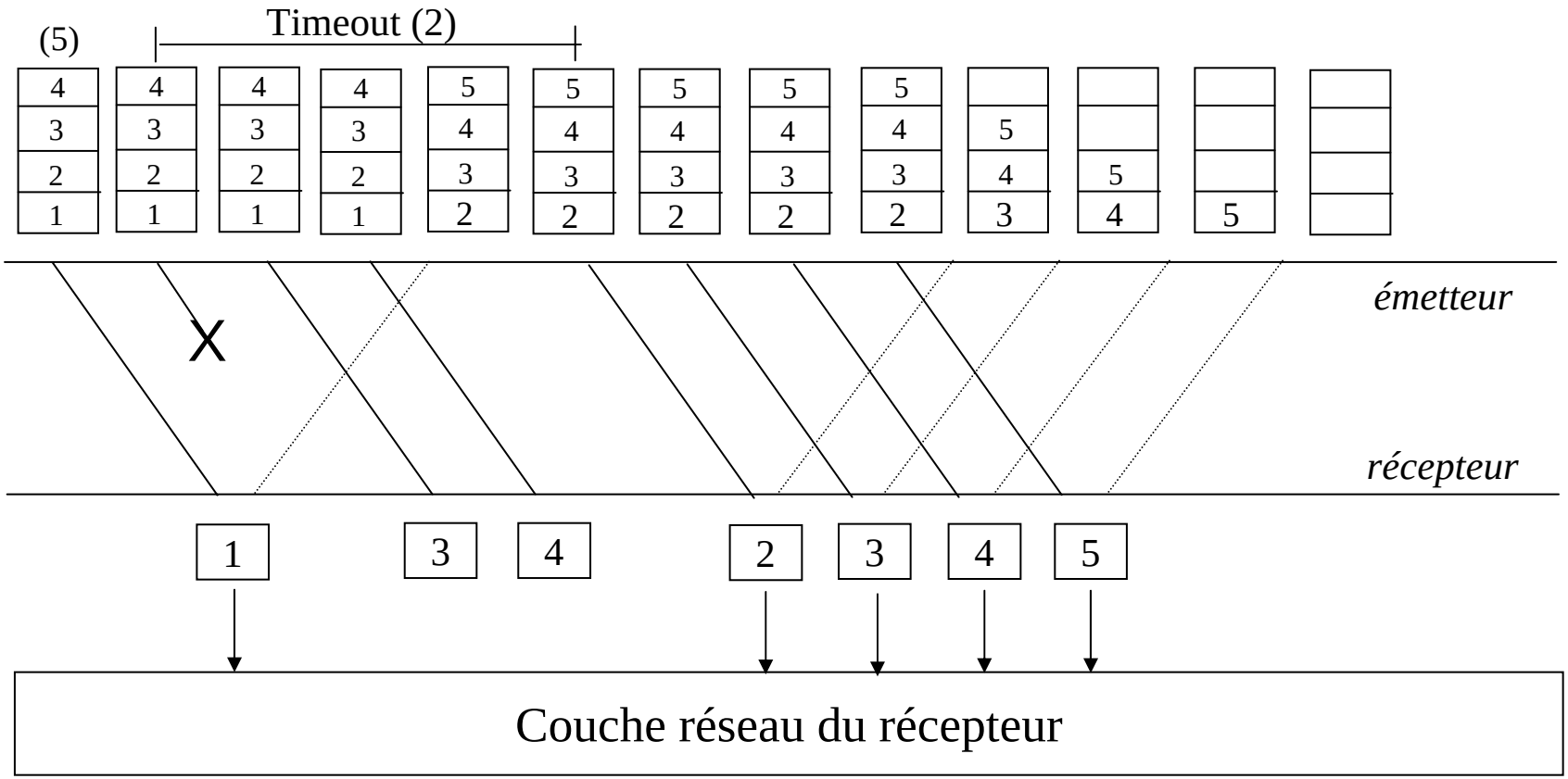
Impact du temps de propagation



Protocole GO BACK N

- Généralisation de ABP à une fenêtre de $w > 1$ trames
- Numérotation modulo w**+1** (sinon, ambiguité…)
- Optimalité: choisir $w = (S / T)$
- Au récepteur: fenêtre de taille 1
  (alors qu'elle est w à l'émetteur)
- A l'émetteur, retransmission d'une trame non-aquiescée, **et de toutes celles qui la suivent dans le fenêtre** (même celles qui auraient été correctement reçues…)

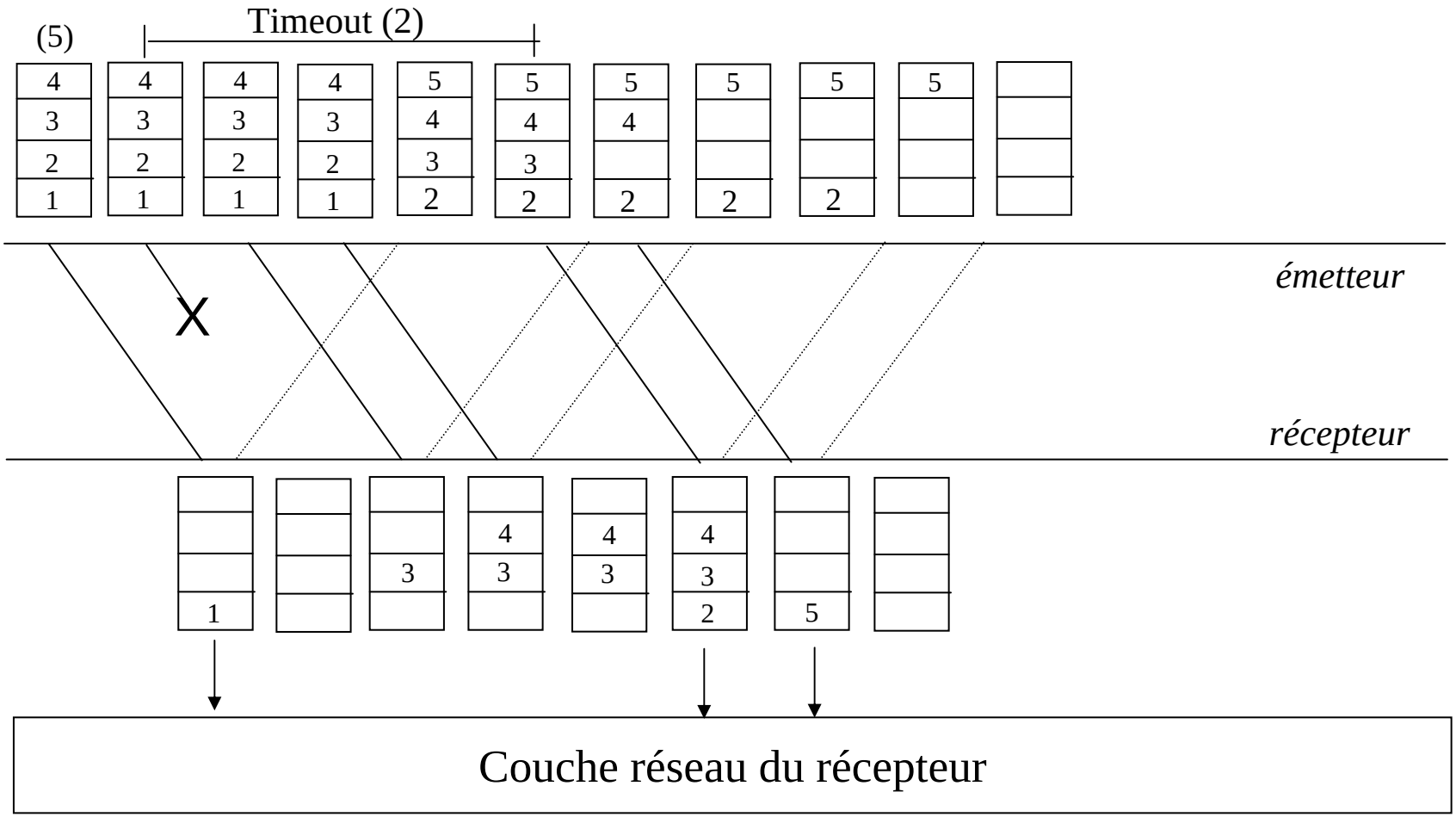
GO BACK N: séquence d'évènements



Protocole SRP

- Utilisation d'une fenêtre (w trames) au récepteur également
- Numérotation modulo $2w+1$
- Le récepteur mémorise les trames sans erreurs, jusqu'à ce qu'il puisse les livrer dans l'ordre à la couche réseau
- A l'émetteur, on ne retransmet **que les trames en erreur** (d'où le nom: Selective Repeat Protocol)
- Plus efficace que GO BACK N lorsque le taux d'erreur est important

SRP: séquence d'évènements



Vérification des protocoles

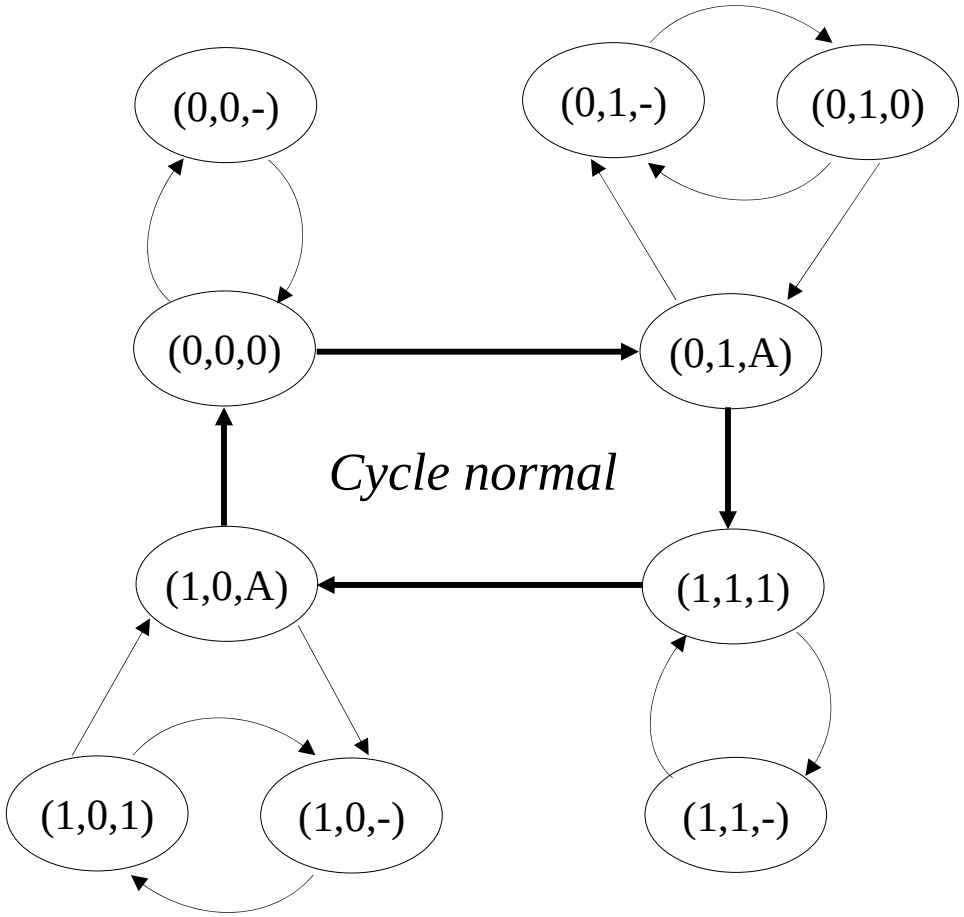
Machine à états finis

- Modélisation d'un protocole comme un ensemble d'**états**, et de **transitions** menant d'un état à un ou plusieurs autres

- Une transition est causée par un évènement
  - arrivée ou transmission d'une trame ou d'un ACK
  - expiration d'un temporisateur (perte d'une trame)
  - interruption
  - etc.

- Modèle complet = protocole + canal

Machine à états finis du protocole ARQ



(e, r, c)

e = état de l'émetteur

- 0: après transmission de la trame 0
- 1: après transmission de la trame 1

r = état du receveur

- 0: attend la trame 0 (a transmis ACK)
- 1: attend la trame 1 (a transmis ACK)

c = état du canal

- 0: trame 0 en transit
- 1: trame 1 en transit
- A: ACK en transit
- -: canal vide (perte…)

Exercice : déterminez pourquoi les états non montrés sont impossibles (ex.: (0,0,1))

Validation du protocole

- Pour que le protocole soit valide, la machine à états finis correspondante doit :

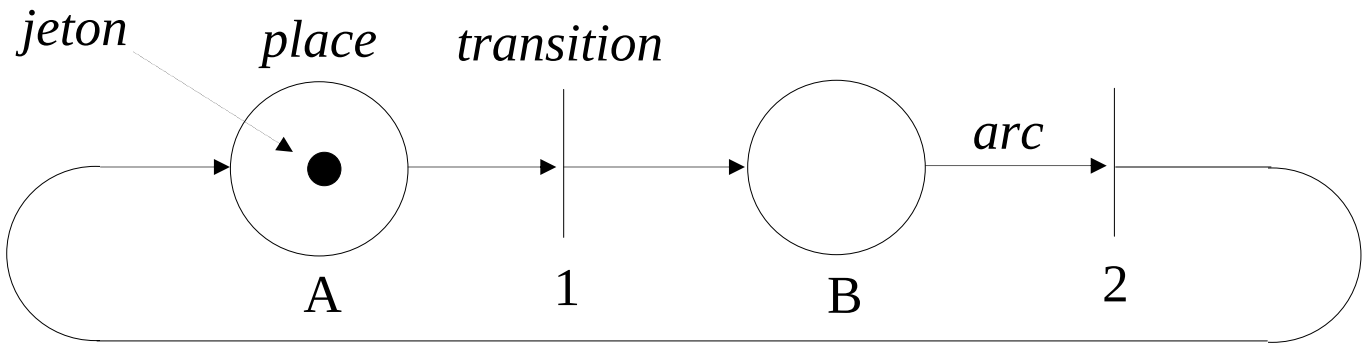
  (1) éviter les séquences d'états ambigues
  → par exemple, l'émetteur change d'état plus d'une fois alors que le receveur demeure dans le même état

  (2) éviter les bloquages ("deadlocks")
  → pris dans un sous-ensemble d'états, dans lequel aucune transition ne permet au protocole de progresser

Réseaux de Petri

Généralisation des machines à états finis

Exemple simple : Réseau à 2 états (A et B)



Réseaux de Petri
transitions

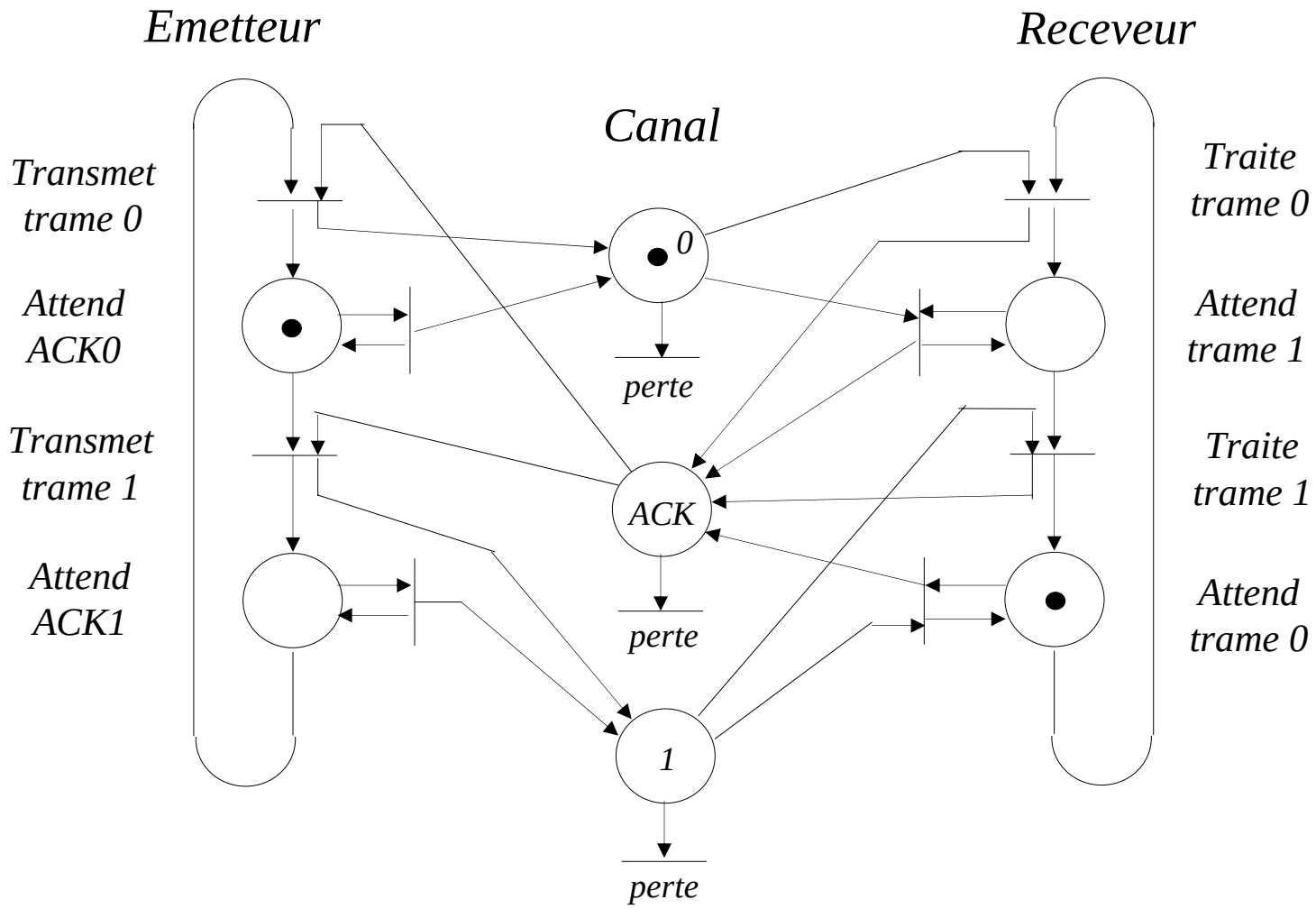
- Une transition peut s'exécuter ("fire") si

  il y a *au moins un jeton*
  dans *chacune des places en entrée*

- Lorsqu'une transition s'exécute, elle

  retire un jeton de *chaque place en entrée*
  et *ajoute un jeton* dans *chaque place de sortie*

Réseau de Petri du protocole ARQ

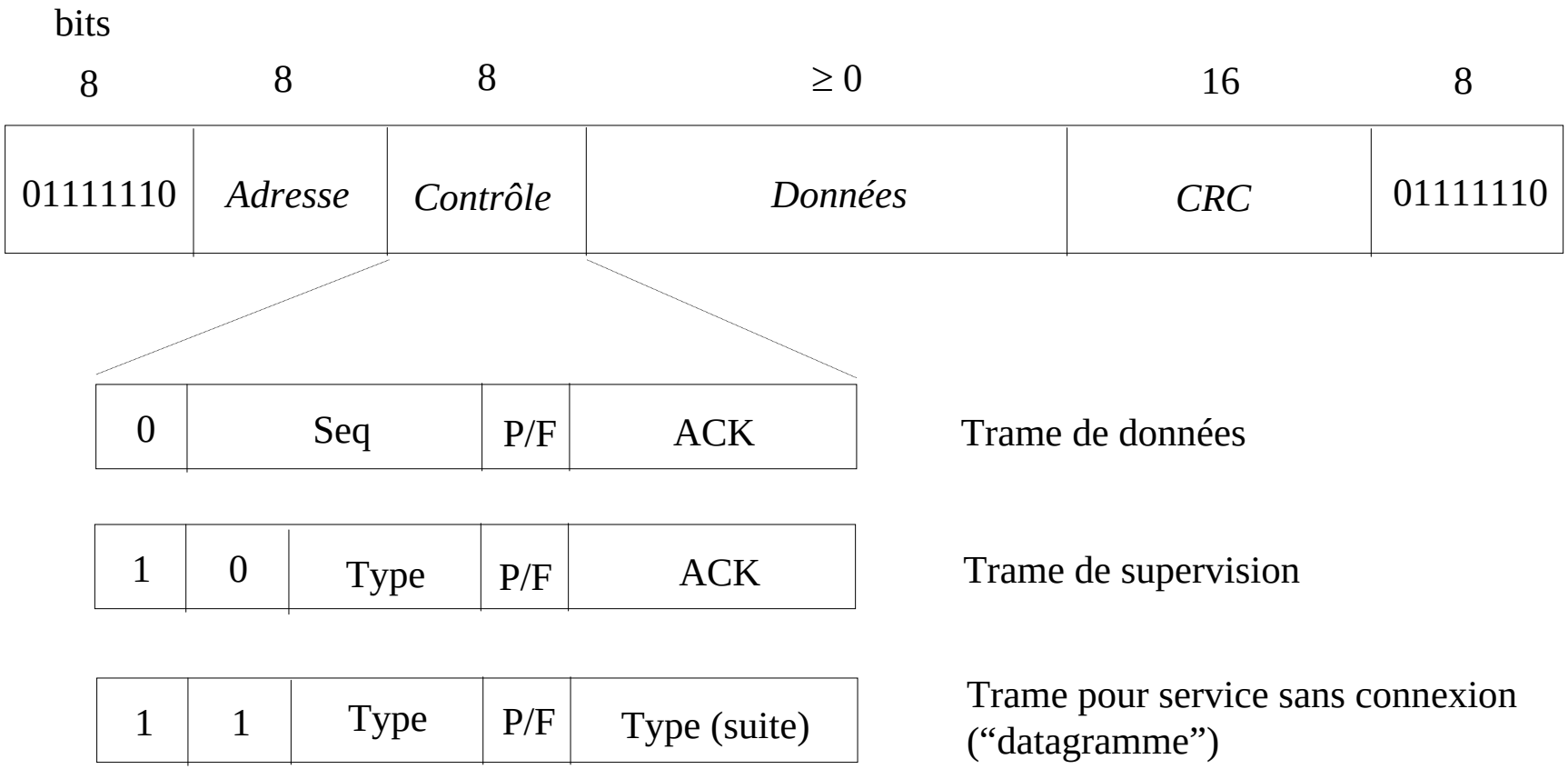


Exemples de protocoles de L. D.

HDLC
High-level Data Link Control

- Utilisé dans les réseaux X.25, entre autres
- Norme ISO
- Protocole orienté bit (“bit stuffing”)
- Utilise GO BACK N avec $W = 7$
  $\rightarrow$ numérotation à 3 bits

Trame HDLC


bits
8
8
8
≥ 0
16
8
01111110
Adresse
Contrôle
Données
CRC
01111110
0
Seq
P/F
ACK
Trame de données
1
0
Type
P/F
ACK
Trame de supervision
1
1
Type
P/F
Type (suite)
Trame pour service sans connexion
(“datagramme”)

Couche L. D. sur Internet

- Pour les connexions point-à-point

  (1) entre routeurs de haut niveau
  (WAN, épine dorsale)

  (2) entre un usager et un ISP via un modem

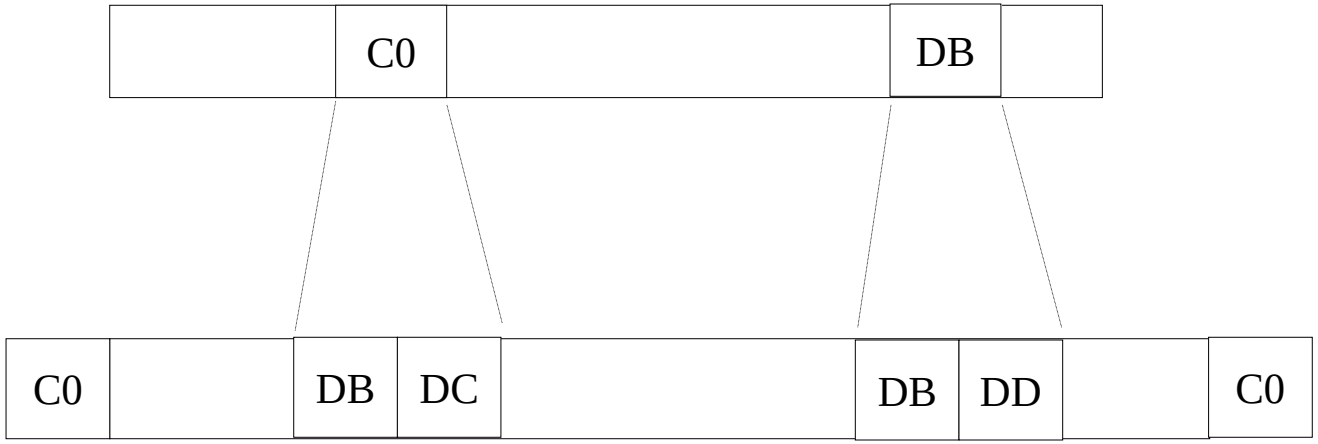
- Deux protocoles largement utilisés:
  - SLIP
  - PPP

SLIP
Serial Link Internet Protocol

- Décrit dans RFC 1055
- Encapsulation *simple* d'un datagramme IP
- Pas de détection d'erreurs
- Compression de l'entête (transmission différentielle)
  $\rightarrow$ RFC 1144
- Supporte uniquement IP

Encapsulation dans SLIP

Datagramme IP

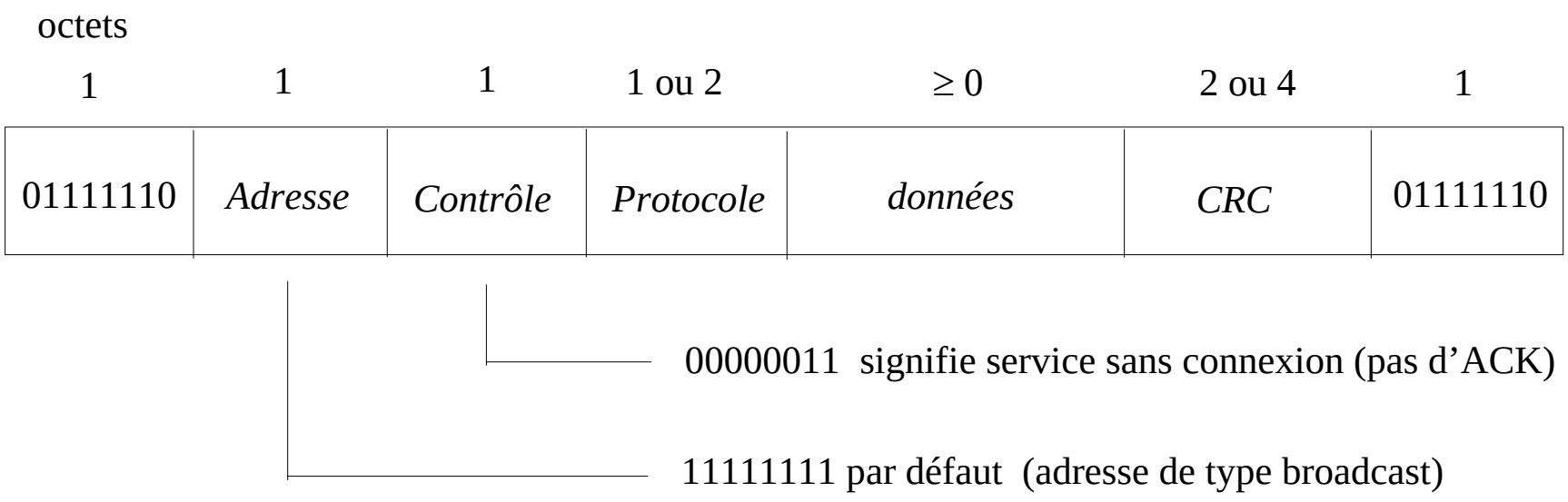


Trame transmise

PPP

- Décrit dans RFC 1661, 1662 et 1663
- Détection d'erreur
- Supporte IP et autres protocoles (IPX, AppleTalk, etc.)
  → indique aussi un paquet LCP (Link Control Prot.)
  ou NCP (Network Control Prot.)
- Permet de négotier une adresse IP lors de la connexion
- Gestion plus complète de la connexion

Trame PPP


octets
1
1
1
1 ou 2
≥ 0
2 ou 4
1
01111110
Adresse
Contrôle
Protocole
données
CRC
01111110
00000011 signifie service sans connexion (pas d'ACK)
11111111 par défaut (adresse de type broadcast)

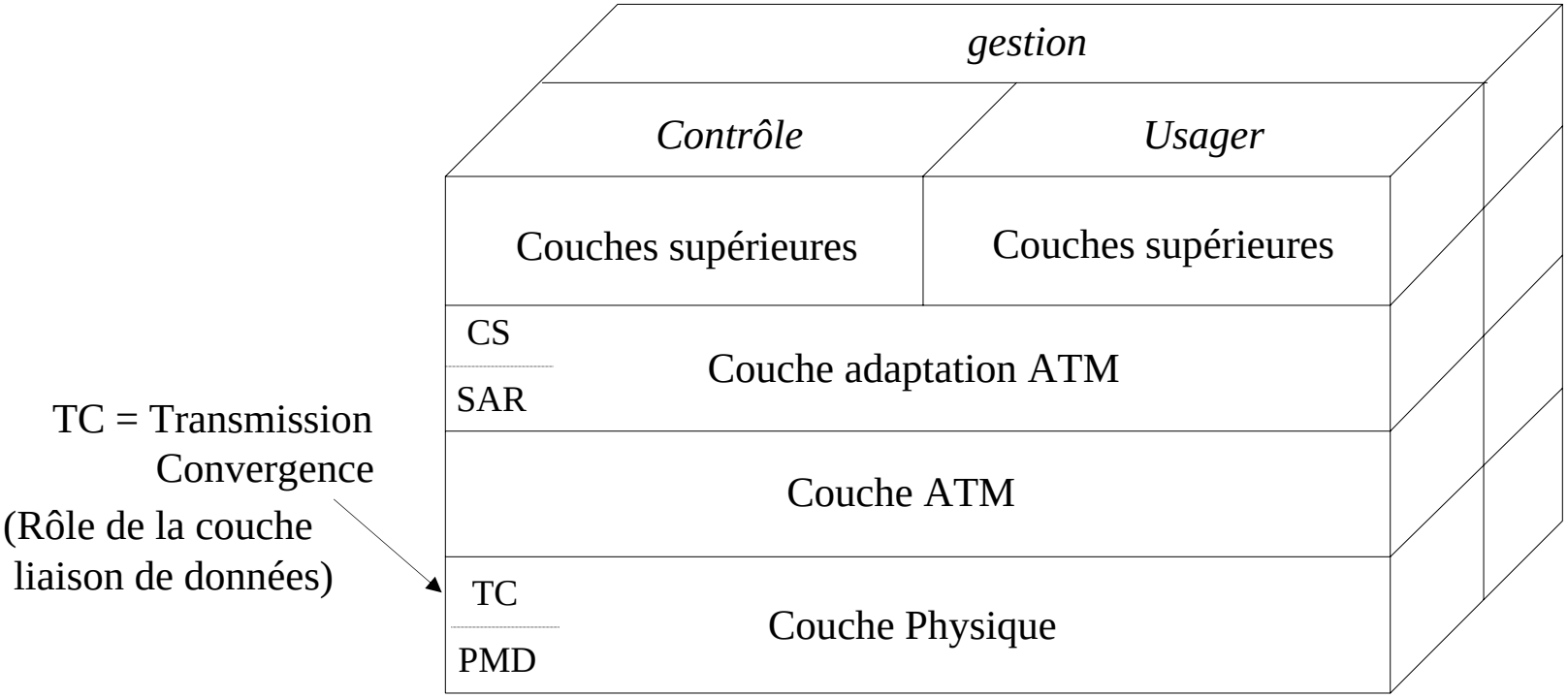
PPP: gestion de la connexion

(1) Connexion modem à modem (client → ISP)
(2) Echange de paquets LCP → paramètres PPP
(3) Echange de paquets NCP → configuration couche réseau
    - pour IP, obtention d'une adresse temporaire

(4) Session TCP/IP (par exemple)

(5) Echange de paquets NCP → libération de l'adresse IP
(6) Echange de paquets LCP → fermeture de conn. logique
(7) Libération de la ligne physique

La couche L.D. sur ATM

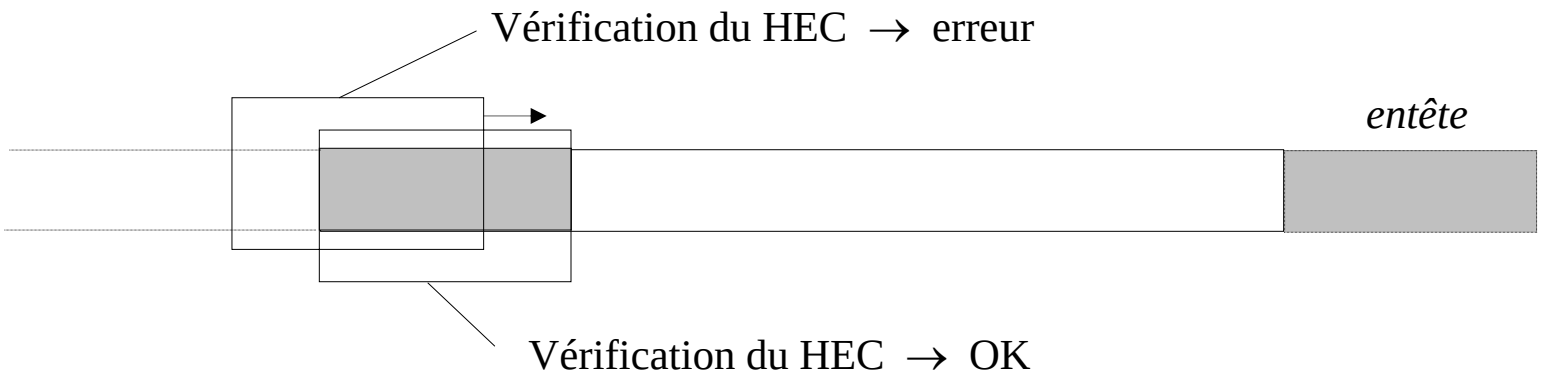


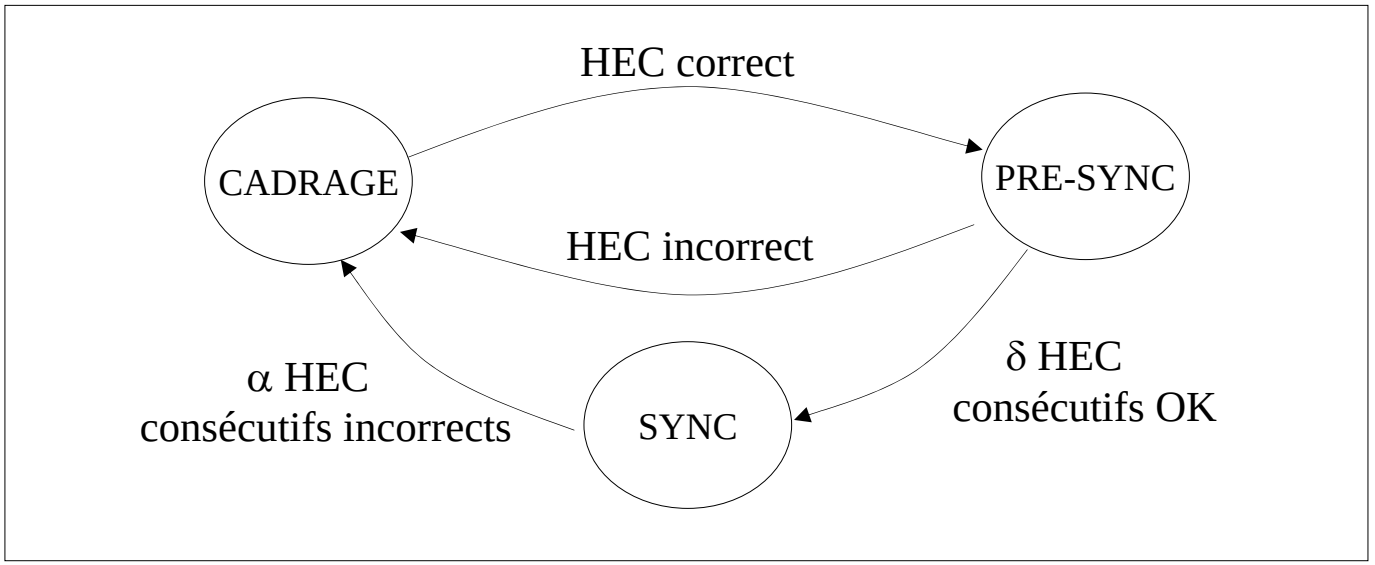
Sous-couche TC

"transmission convergence"

- Détection d'erreur sur l'*entête uniquement*
  - → entête = 4 octets d'adresse + 1 octet CRC (HEC)
  - → $g(x) = x^8 + x^2 + x + 1$
- Adaptation au débit de la couche physique
- Câdrage ("framing")
  - → à la réception
  - → tâche non-triviale
    (il n'y a pas de marqueur (i.e. drapeau), comme c'est le cas dans HDLC, PPP ou SLIP)

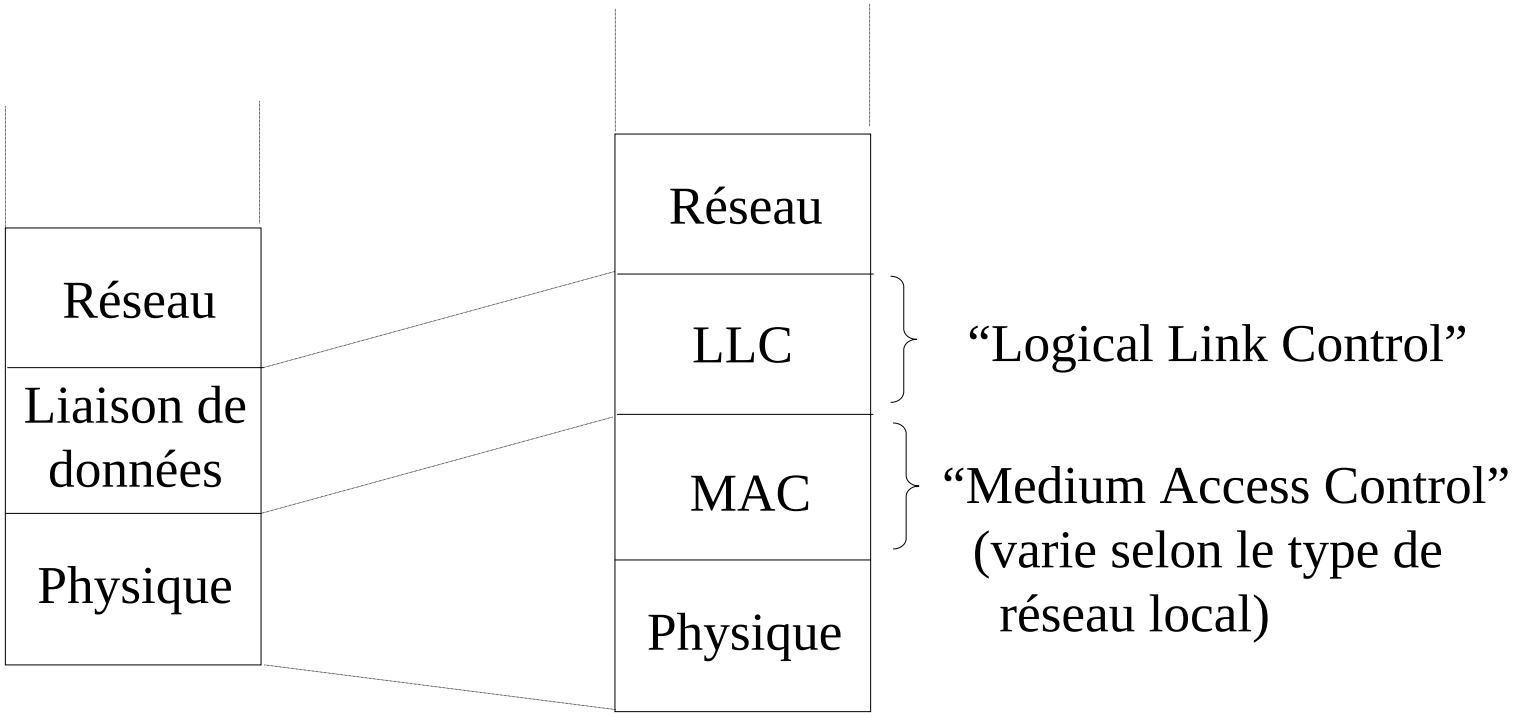
Câdrage dans ATM



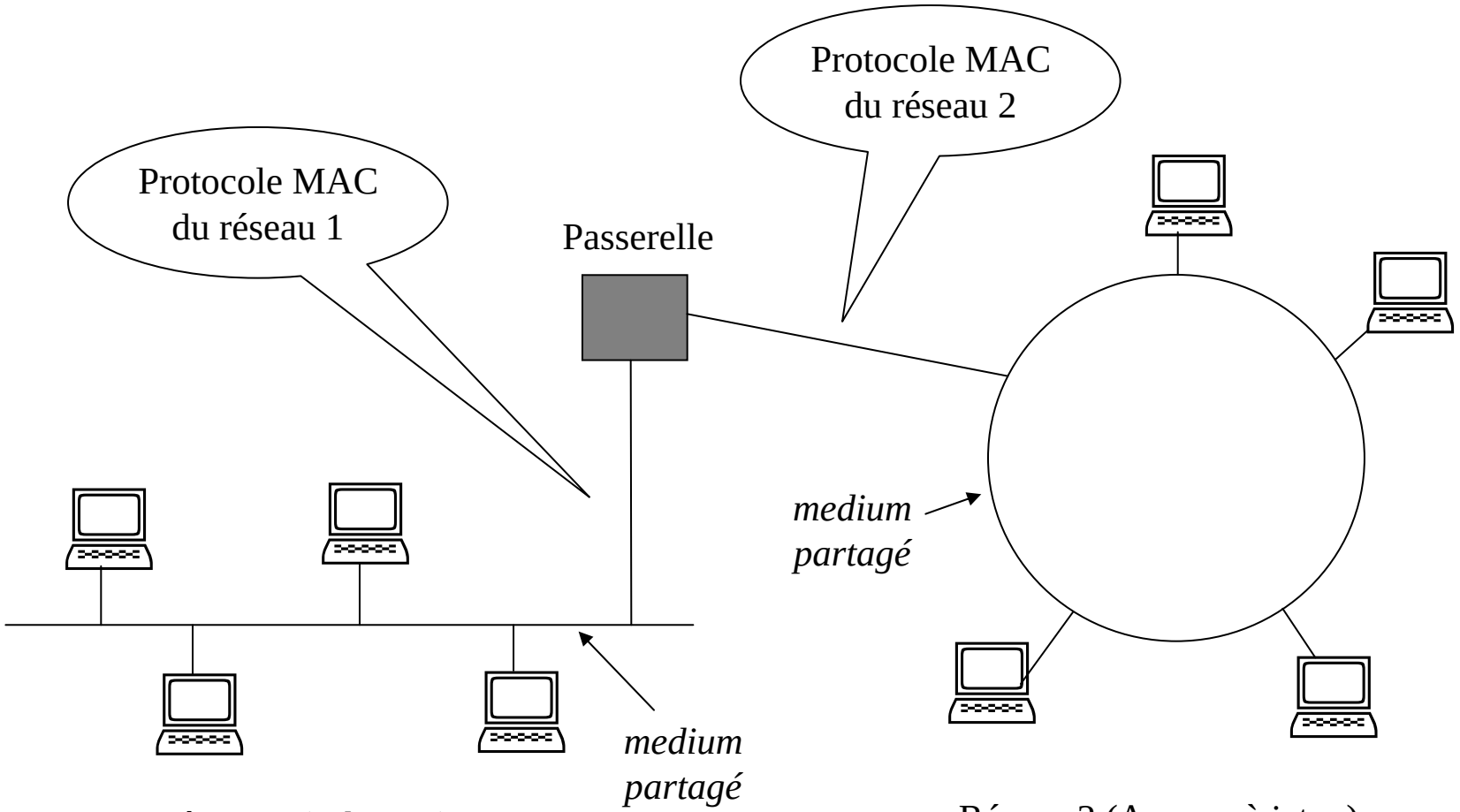


- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

La sous-couche MAC



Interconnexion de réseaux



Protocoles d'accès au medium

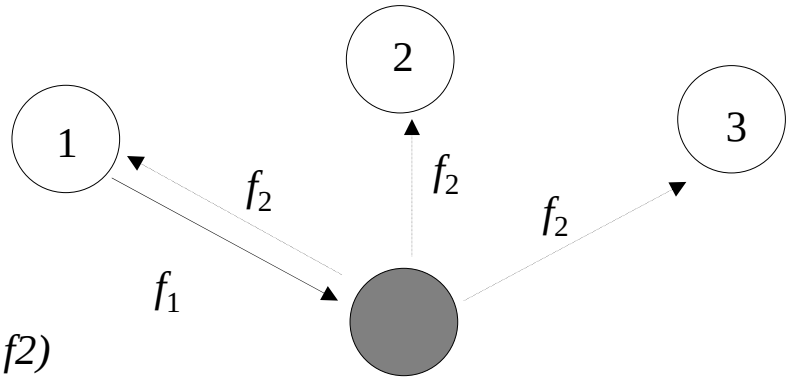
- Contexte: canal unique à accès multiple

- Allocation statique
  - Circuit fixe pour la durée de la communication
  - Faible performance à débit variable
  - Exemples: TDM, FDM

- Allocation dynamique
  - Multiplexage statistique
  - Efficace pour les sources à débit variable
  - Exemples: ALOHA, CSMA, Anneau à jeton, ...

ALOHA

- Chaque usager transmet *dès qu'une trame est prête*
- Trames simultanées, ou partiellement = *collision*
- Reprise après collision:
  - → attente d'une durée aléatoire
  - → retransmission de la trame (nouvelle tentative)
- Version originale: réseau sans fil centralisé ("broadcast")



L'usager 1 transmet (sur f1)

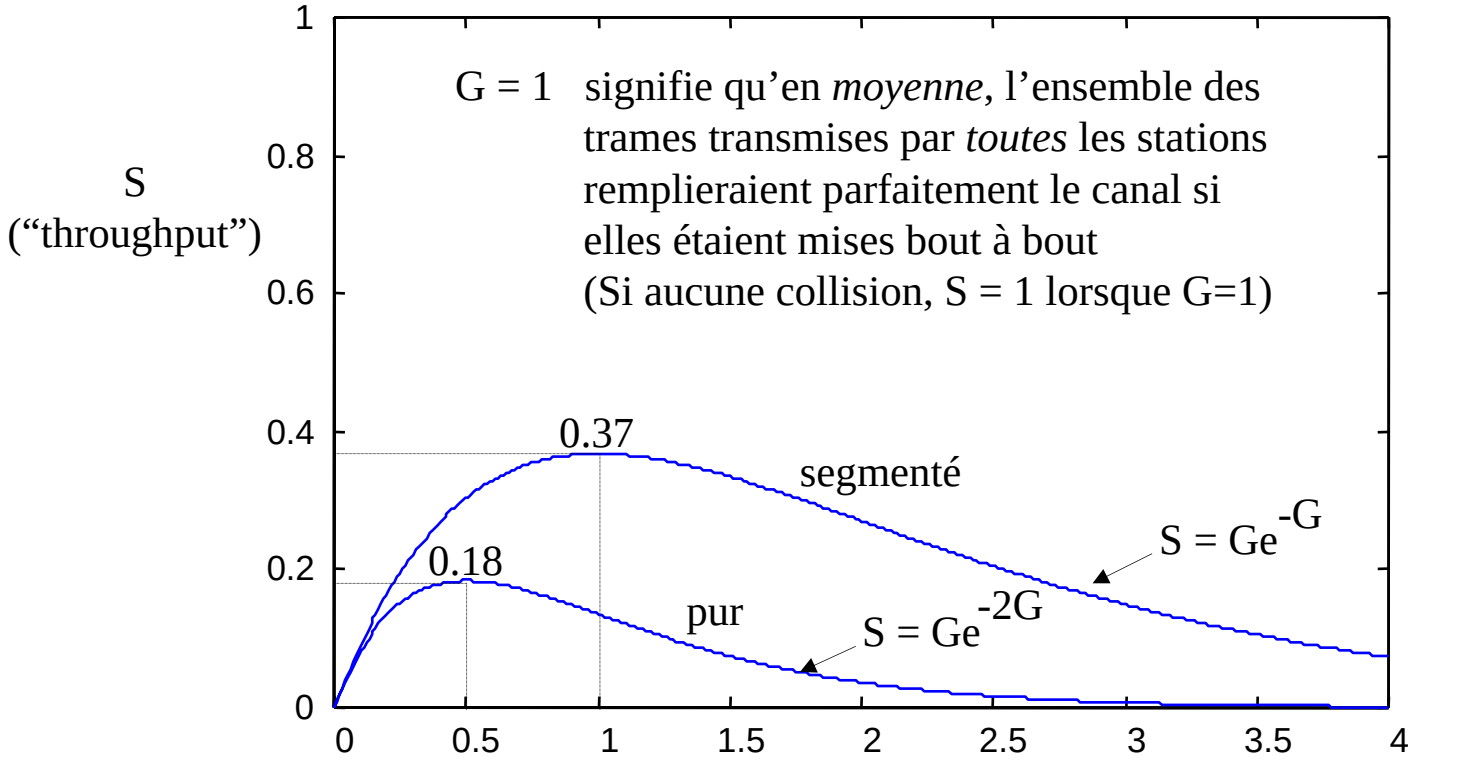
Le noeud central réémet (sur f2)

ALOHA pur et segmenté

- **ALOHA pur** :
  - → Aucune restriction sur le début de transmission d'une trame
  - → Détection de collision:
    comparaison entre trame émise et reçue (sur f_2)

- **ALOHA segmenté** :
  - → Horloge centrale
  - → Temps divisé en *segments* (durée = 1 trame)
  - → Transmission uniquement dans les segments
  - → Détection de collision:
    même procédure que ALOHA pur

ALOHA: performance



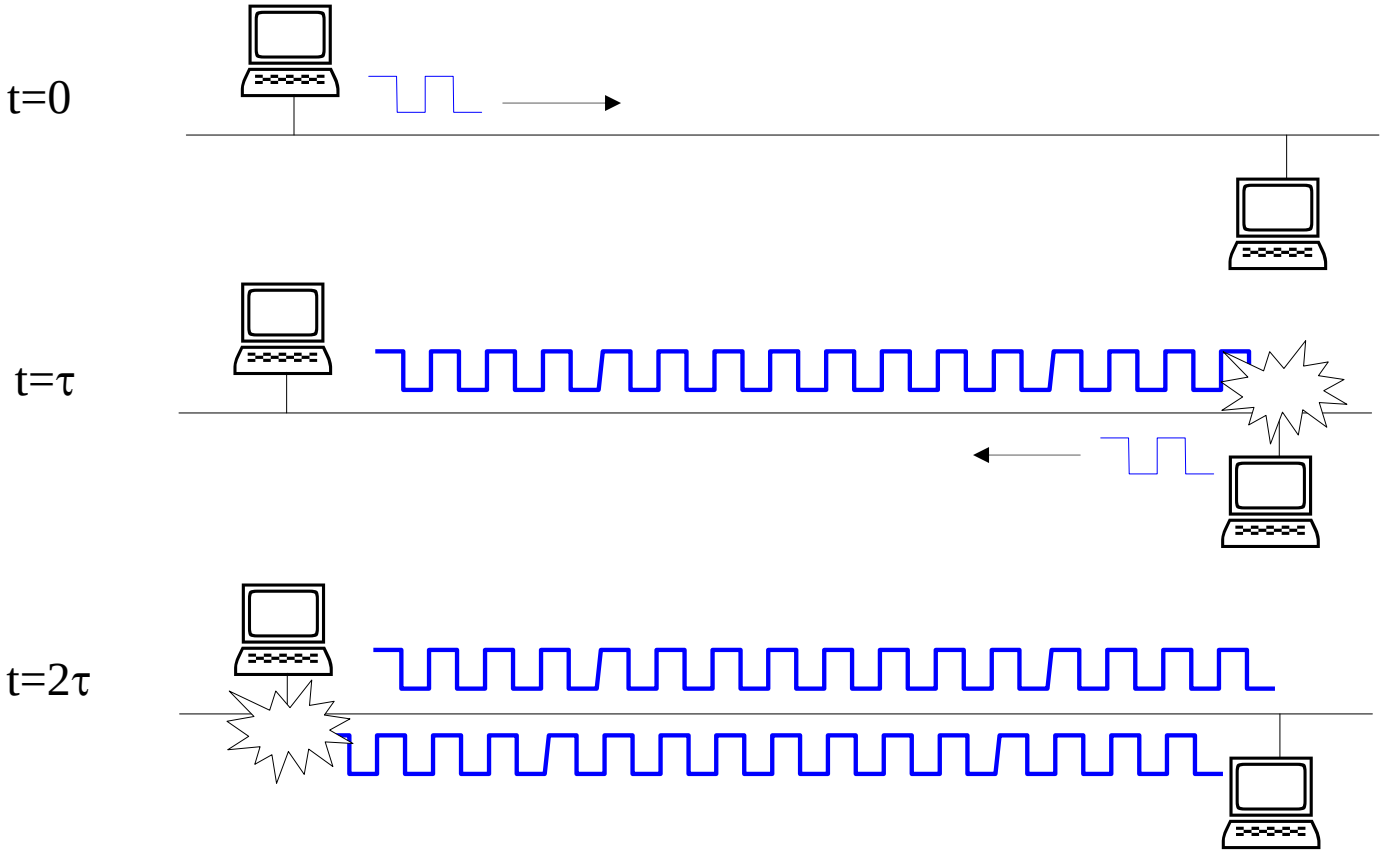
CSMA

"Carrier Sense Multiple Access"

- Version "polie" d'ALOHA
- Un noeud attend que le canal soit libre avant de transmettre
- Interruption immédiate si collision (CSMA/**CD**)
  $\rightarrow$ à la base des réseaux Ethernet
- Meilleure performance qu'ALOHA

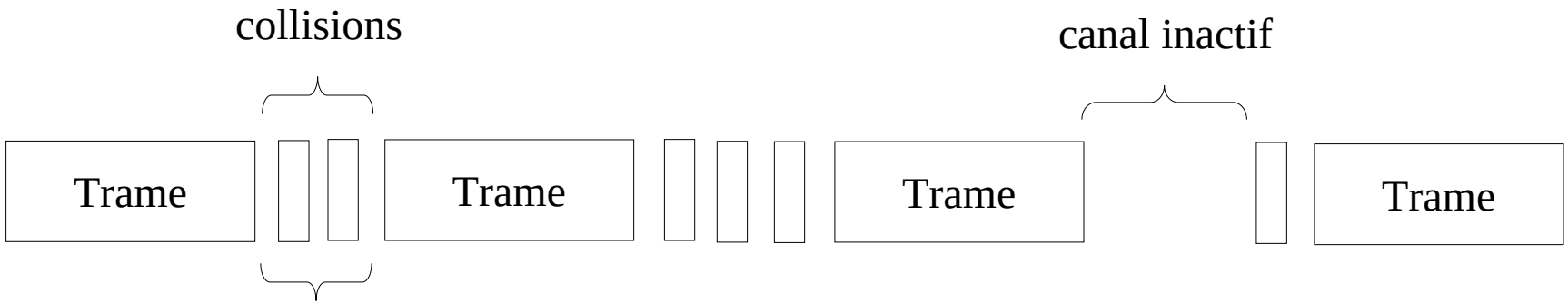
CSMA/CD

Détection des collisions



CSMA/CD

Etats du canal



On peut modéliser la durée de *contention* par la protocole ALOHA segmenté, où la durée d'un segment est 2τ, avec τ le temps de propagation d'un bout à l'autre du câble.

CSMA/CD

Action lors d'une collision

- Attente d'une durée aléatoire
- Retransmission si le canal est libre

Protocoles sans collision

- Mécanisme de *réservation*
- Alternance *réservation-transmission-réservation-…*
- Approche hybride:
  - charge faible $\rightarrow$ CSMA/CD
  - charge élevée $\rightarrow$ réservation
- CDMA: une classe à part
  - $\rightarrow$ un “canal” par usager
  - $\rightarrow$ transmission continue pour chaque usager
  - $\rightarrow$ résoud le problème des collisions

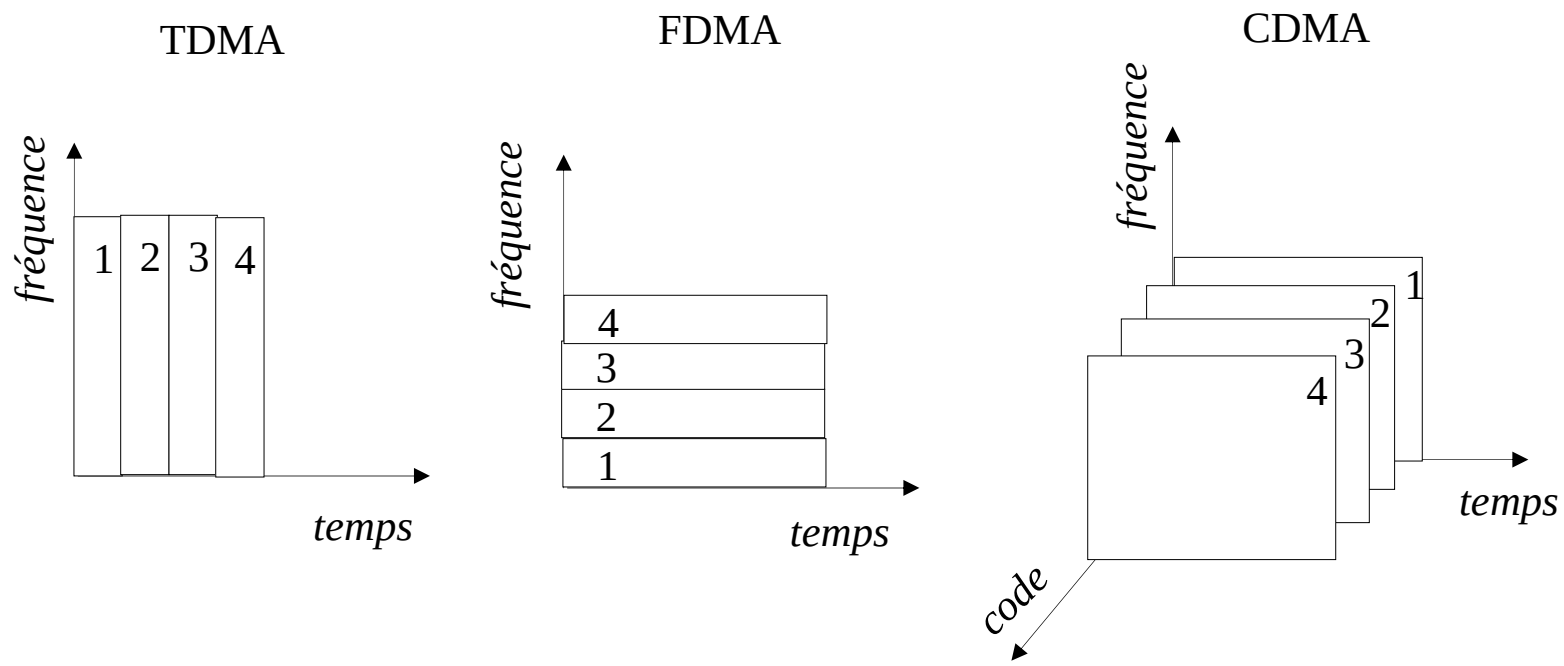
CDMA

- CDMA = “Code Division Multiple Access”
- Technique à bande élargie (“spread spectrum”)
- Allocation d’un canal par usager
- Chaque usager transmet *tout le temps*
  sur toute la bande

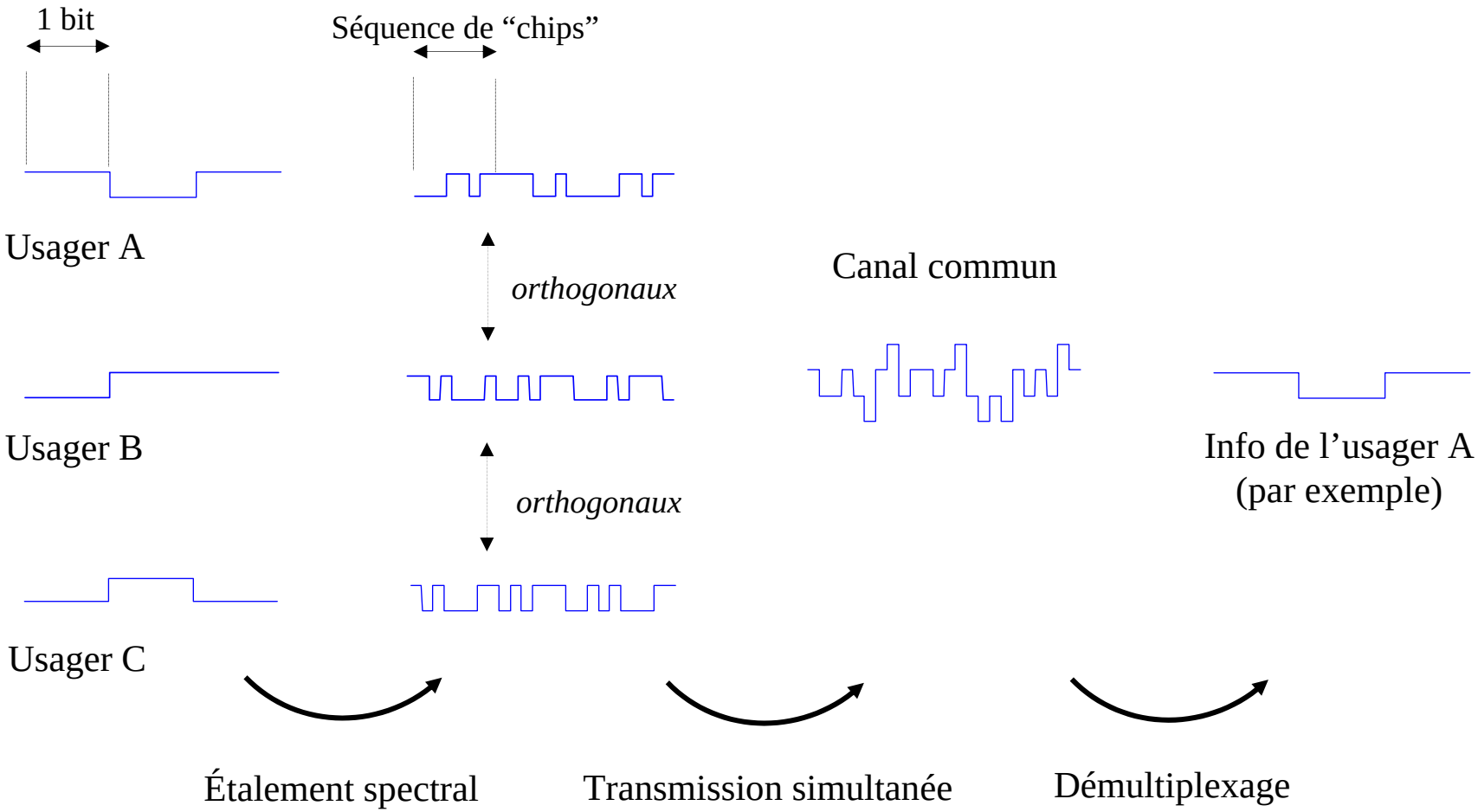
contrairement à FDMA → bandes de fréquences
et TDMA → multiplexage dans le temps

TDMA, FDMA et CDMA

Comparaison



CDMA: principe



CDMA: démultiplexage

Séquence de chips (binaire)

canal

A : 0 0 0 1 1 0 1 1
B : 0 0 1 0 1 1 1 0
C : 0 1 0 1 1 1 0 0

Séquence de chips (bi-polaire)

A : (-1 -1 -1 +1 +1 -1 +1 +1)
B : (-1 -1 +1 -1 +1 +1 +1 -1)
C : (-1 +1 -1 +1 +1 +1 -1 -1)

Démultiplexage pour le canal A

Information transmise

temps

A	B	C
1	-1	-1
-1	1	1
1	1	-1

Signal transmis = S
(Somme des chips pour chaque intervale d'un bit)

temps

(+1 -1 -1 +1 -1 -3 +1 +3)
(-1 +1 +1 -1 +1 +3 -1 -3)
(-1 -3 +1 -1 +1 -1 +3 +1)

(S • **A**) / 8

temps

1
-1
1

CDMA: limites

- La puissance reçue au récepteur doit être la même pour chaque canal
  → gestion dynamique de la puissance

- Nécessite un émetteur à haut débit (coûteux)
  → typiquement, 128 chips/bit

- Capacité moindre que TDM pour un canal très bruyant

Quelques réseaux locaux standards

La série IEEE 802

<table>
<tr><td></td><td colspan="4">Couches supérieures</td></tr>
<tr><td>LLC</td><td colspan="4">802.2</td></tr>
<tr><td>MAC</td><td>802.3
(Ethernet)</td><td>802.4
(Anneau logique)</td><td>802.5
(Anneau physique)</td><td>802.6
(DQDB)</td></tr>
<tr><td></td><td colspan="4">Couche physique</td></tr>
</table>

IEEE 802.3

(Ethernet)

- Basé sur le protocole CSMA/CD
- Après une collision:
  → attente d'une durée aléatoire
- Collisions multiples:
  → attente de durée croissante avant une nouvelle tentative ("exponential backoff")

IEEE 802.3 : reprise des collisions

"exponential backoff"

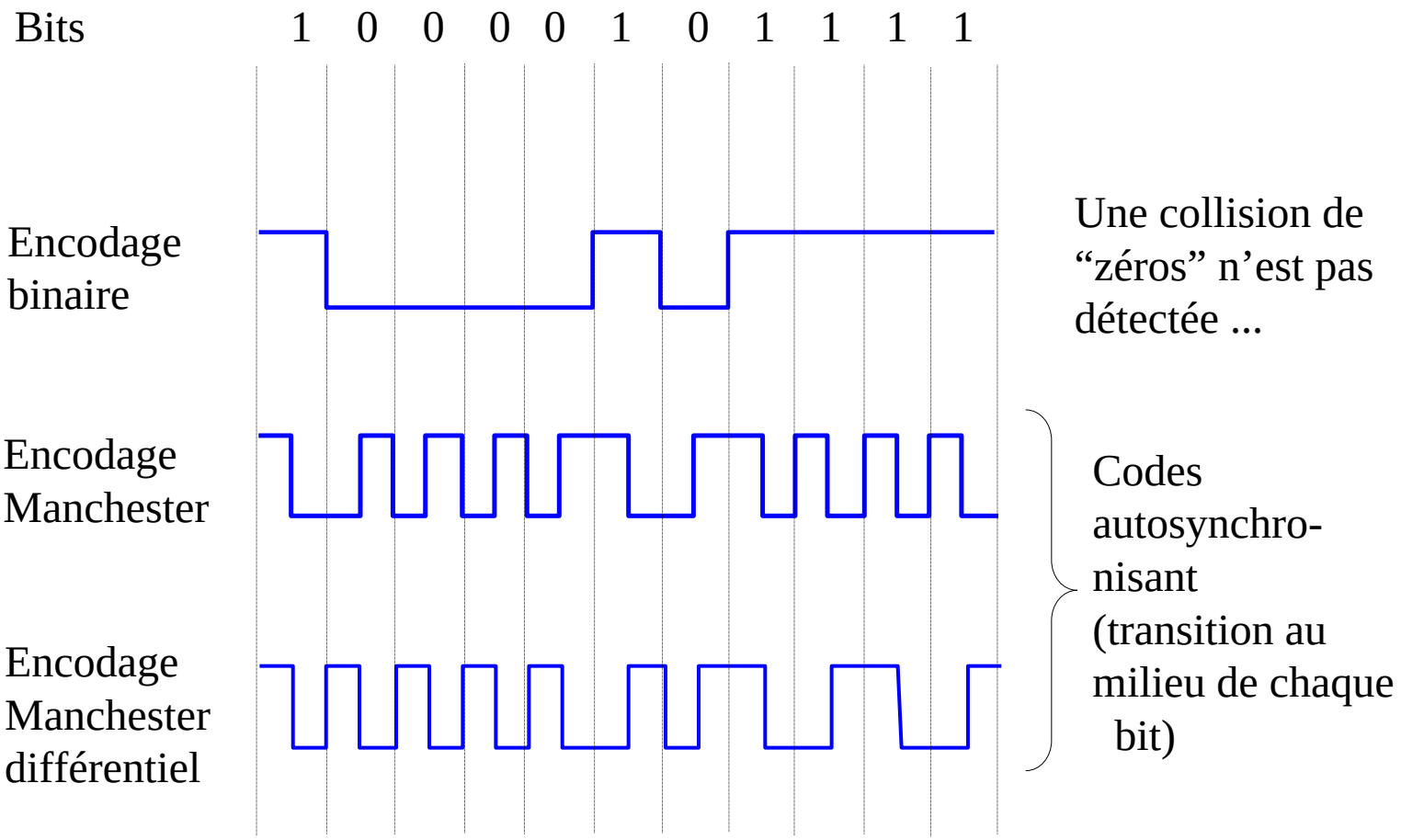
- Après la $i^{ème}$ collision (pour une même trame), un émetteur

  (1) choisit un nombre aléatoire *n* entre 0 et 2^i -1
  (2) attend *n* segments (typiquement, 1 segment = 512 bits)
  (3) fait une nouvelle tentative de transmission

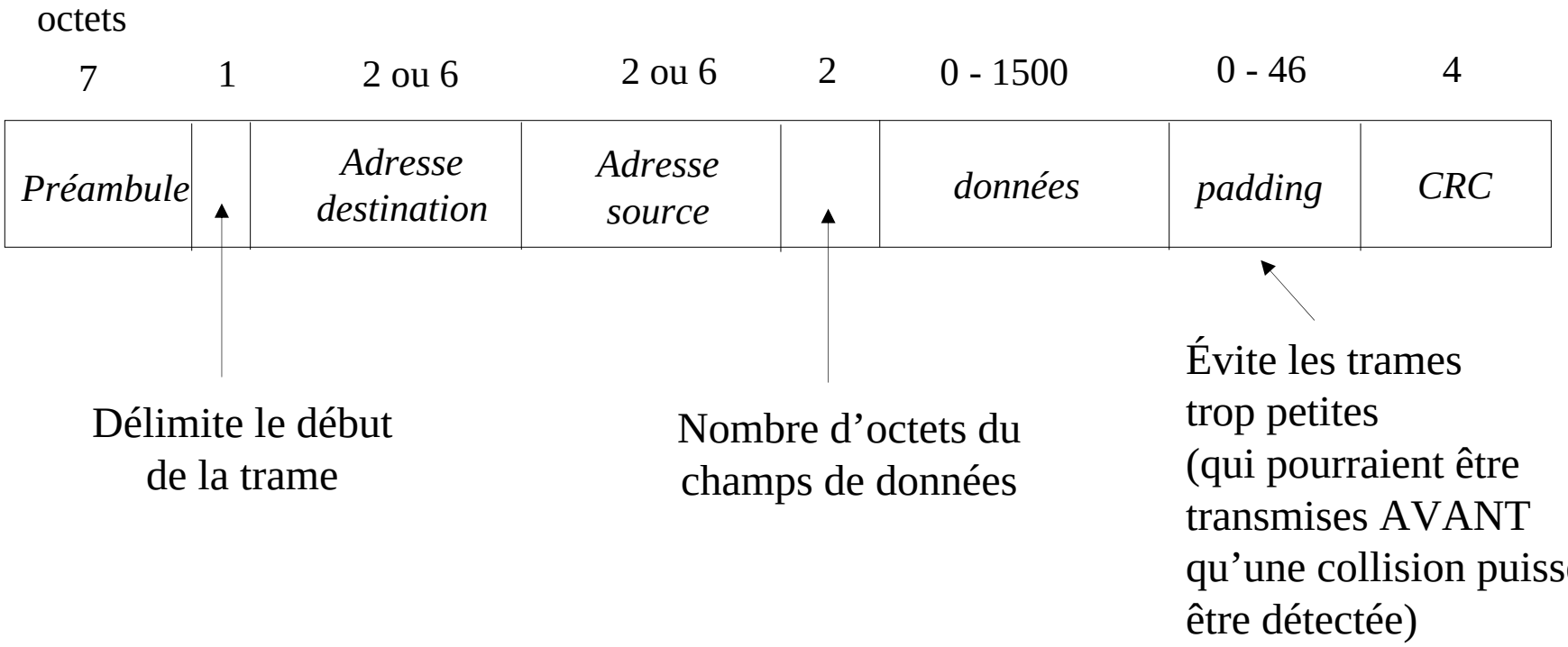
- Aprés 16 collisions

  → une erreur est rapportée (abandon)
  → c'est aux couches supérieures de décider de la suite

Encodage Manchester

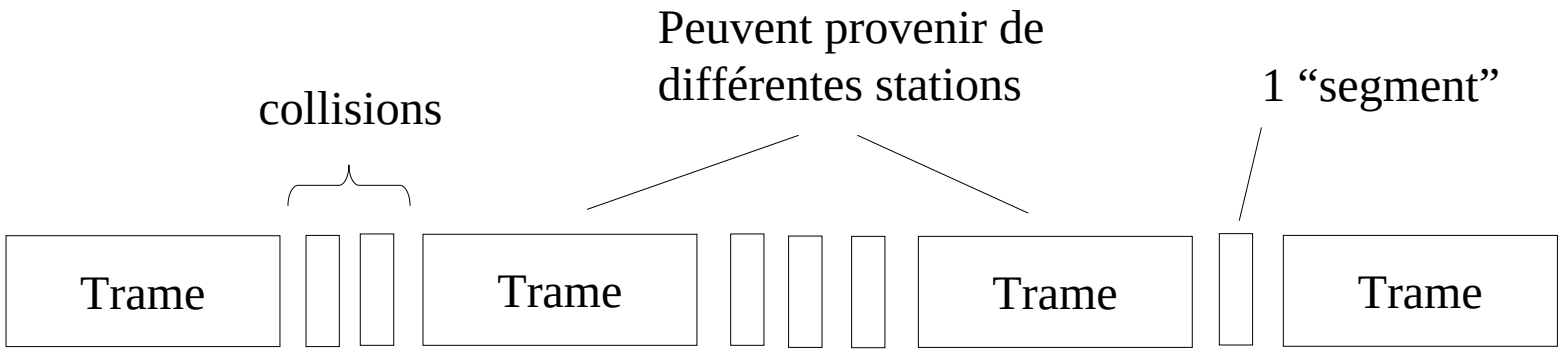


Trame IEEE 802.3



- Le préambule (1010…10) sert à synchroniser l'horloge du récepteur

IEEE 802.3 : efficacité



$$\text{Efficacité} = \eta = \frac{\text{Durée moyenne d'une trame}}{\text{Durée moyenne d'une trame + durée moyenne de contention}}$$

IEEE 802.3 : calcul de l'efficacité

- Hypothèses:
  - → k stations, toujours prêtes à transmettre
  - → Durant un "segment" dans la période de contention, une station transmet avec probabilité p

- Dans ce cas,

$A = kp(1-p)^{k-1}$ est la probabilité qu'une station gagne le canal (ie. elle seule a transmis durant le segment de contention donné)

IEEE 802.3 : calcul de l'efficacité

- Donc, la probabilité que la période de contention ait exactement *j* segments est

$$A\,(1\text{-}A)^{\,j\text{-}1}$$

- Et ainsi, le nombre moyen de segments dans une période de contention est

$$N = \sum_{j=0}^{\infty} j\,A\,(1\text{-}A)^{\,j\text{-}1} = \frac{1}{A}$$

IEEE 802.3 : calcul de l'efficacité

- On trouve donc la durée moyenne de contention:

$$Dc = 2\,\tau / A$$

où τ est le temps de propagation d'un bout à l'autre du réseau

- Avec *Dt* la durée moyenne de transmission d'une trame, on trouve finalement l'efficacité :

$$\eta = \frac{Dt}{Dt + (2\,\tau / A)}$$

IEEE 802.3 : calcul de l'efficacité

- En posant $\tau = BL / c$

  où B est la largeur de bande du canal (Hz)

  L est la longueur du câble

  c est la vitesse de la lumière (vit. de propagation)

  F est le nombre de bits par trame

et sachant que $Dt = F / B$

on trouve

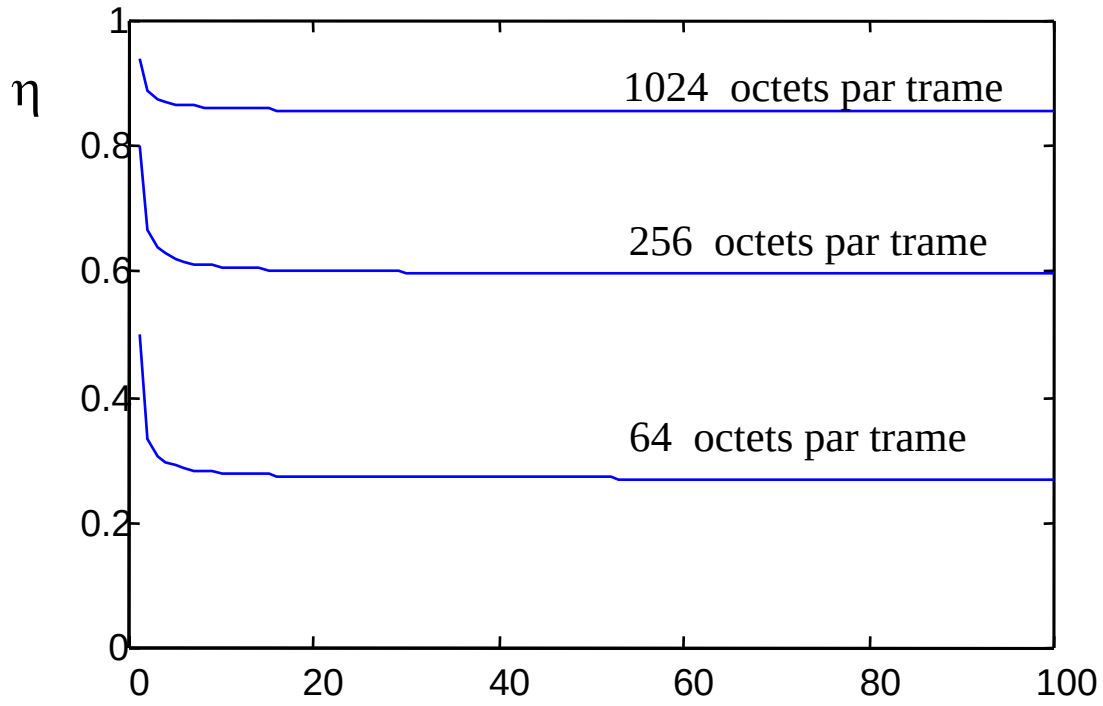
$$\eta = \frac{1}{1 + (2BL) / (cFA)}$$

L'efficacité
- diminue avec B et L
- augmente avec F

IEEE 802.3

Efficacité pour différentes conditions

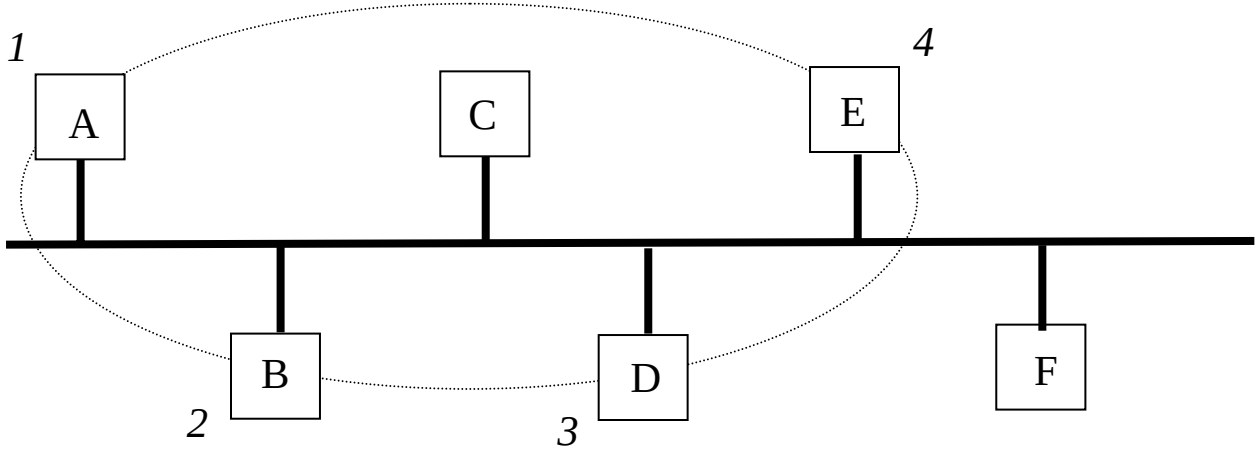
Réseau Ethernet à 10 Mbps
Segment de contention = 64 octets



IEEE 802.4
(Bus à jeton -- anneau logique)

- Une seule station peut transmettre à la fois
- Un *jeton* (trame spéciale) circule pour donner droit de parole
  → exemple: course à relais
- Lorsqu'une station est en possession du jeton
  → elle peut transmettre pendant un certain temps (THT)
  → elle passe ensuite le jeton au prochain sur la liste
- Réseau de type "broadcast" comme Ethernet, mais sans collision (sauf lorsqu'une station veut se joindre à l'anneau logique)

IEEE 802.4: principe

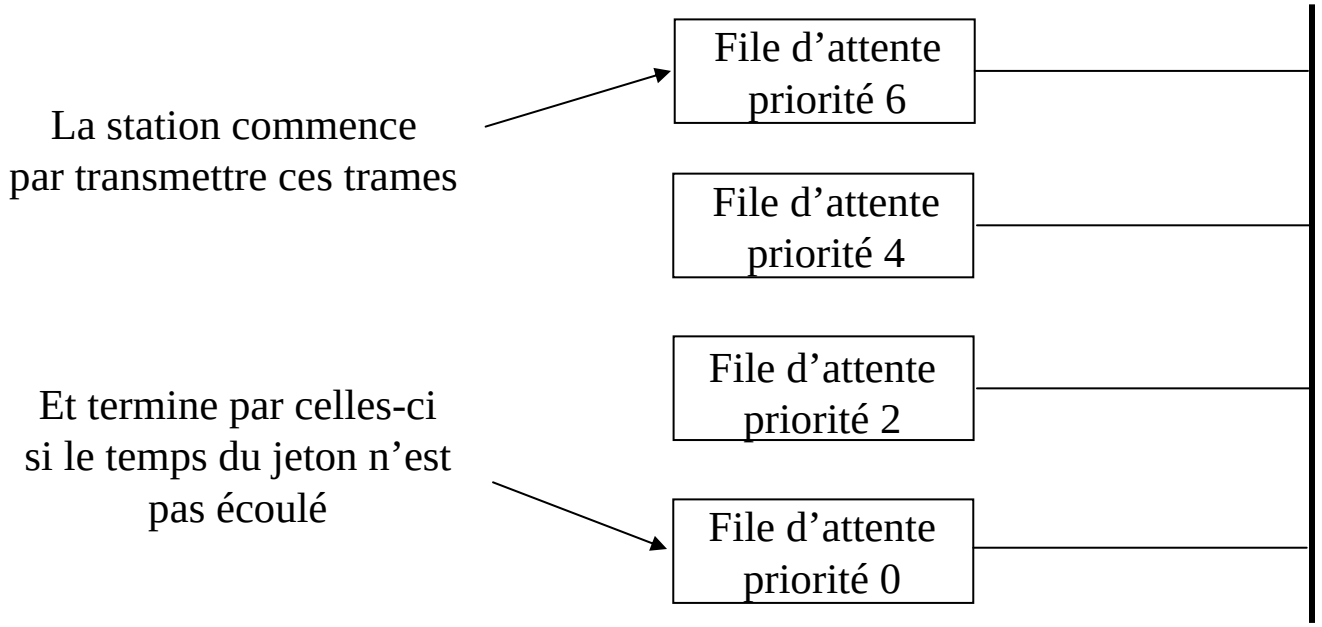


- Les chiffres indiquent l'ordre logique dans l'anneau
  - → D est le *successeur* de B
  - → A est le *prédécesseur* de B
  - → Chaque noeud connaît son prédécesseur et son successeur

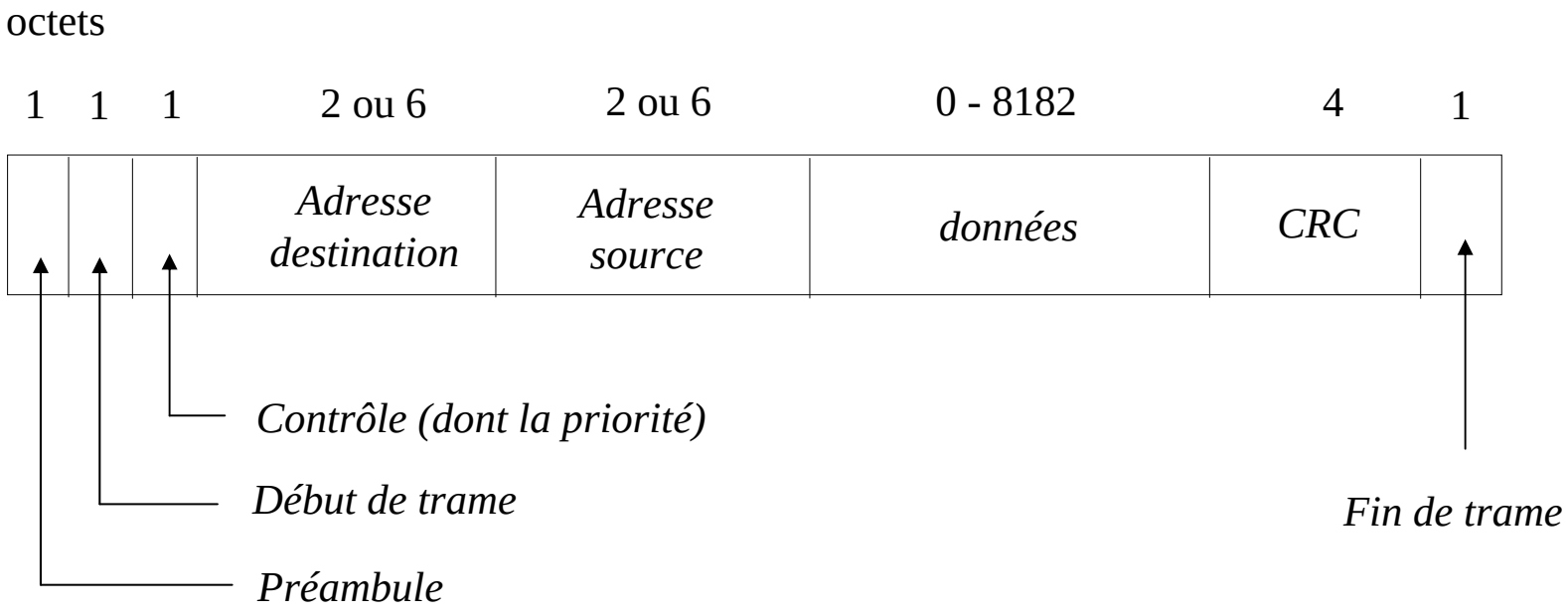
- C et F ne sont pas encore dans l'anneau (ils ne peuvent transmettre)
  - → Procédure spéciale pour entrer (ou sortir) de l'anneau

IEEE 802.4
Priorités

- Quatre classes de priorités (0, 2, 4, 6)
- Chaque classe a sa propre file d'attente pour transmission



Trame IEEE 802.4



IEEE 802.4

Gestion de l'anneau logique: décentralisée

Contrôle	*Nom*	*Signification*
00000000	Claim_Token	Une station demande le jeton lors de l'initialisation
00000001	Sollicit_Successor_1	Permet à une station d'entrer dans l'anneau
00000010	Sollicit_Successor_2	Permet à une station d'entrer dans l'anneau
00000011	Who_follows	Après perte du jeton
00000100	Resolve_contention	Gère les collisions (après Sollicit_Successor)
00001000	Token	Le jeton
00001100	Set_successor	Permet de quitter l'anneau

IEEE 802.5

(Anneau à jeton -- anneau physique)

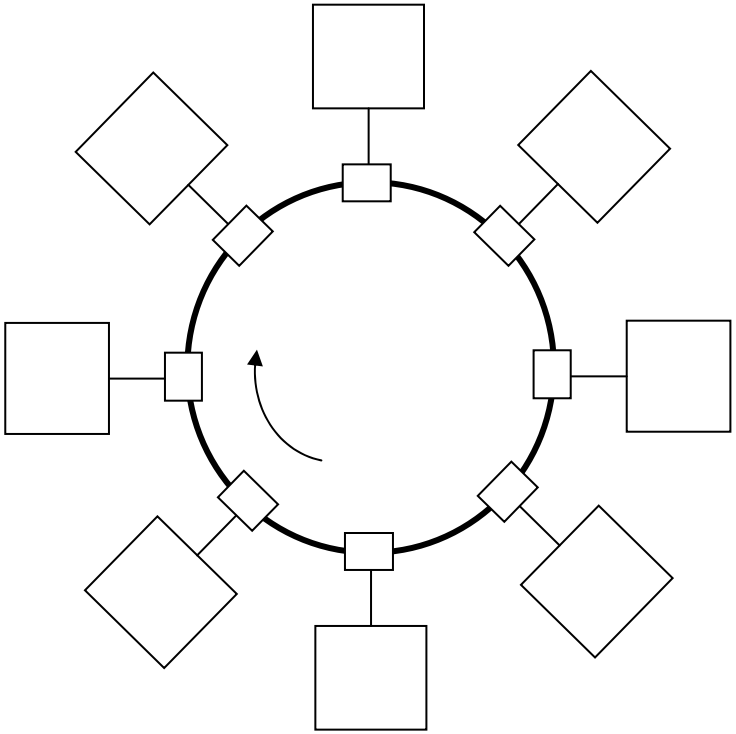
- Formé d'une succession de *liens point-à-point*
  - → élément de base = réseau le plus simple (…)
  - → Opération presque entièrement numérique
    (contrairement à Ethernet, par exemple,
    où la détection de collision est *analogique…)*

- Temps d'accès maximal = limite supérieure précise
  - → fonction du temps de rétention du jeton (THT)

- Les anneaux réalisent un *multiplexage temporel*
  - → mais PAS de gaspillage si une station est inactive

IEEE 802.5

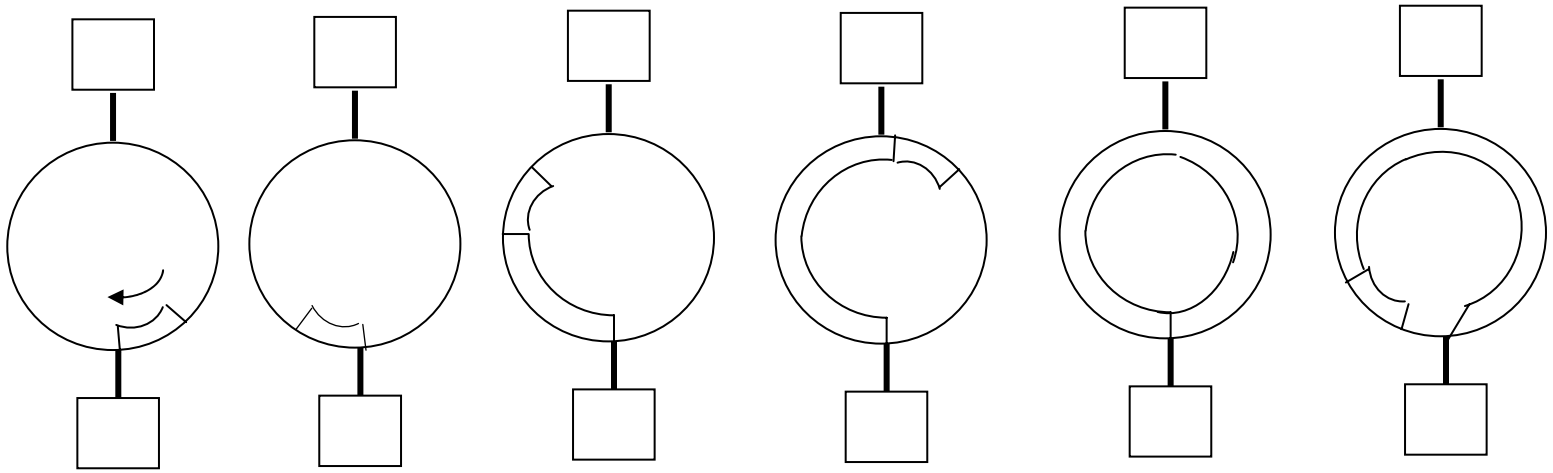
Topologie



- Un jeton gère le droit de transmission (le jeton circule *continuellement* lorsque les stations ne transmettent pas de trame)
- L'anneau est *unidirectionnel*
- Le prédécesseur et le successeur sont les stations auxquelles un noeud est connecté *physiquement*
- Lorsqu'une station n'a pas le jeton, elle est simplement un *répéteur* (avec un *délai de 1 bit*)

IEEE 802.5

Vie d'une trame sur l'anneau



La station reçoit le jeton

La jeton est modifié en début de trame

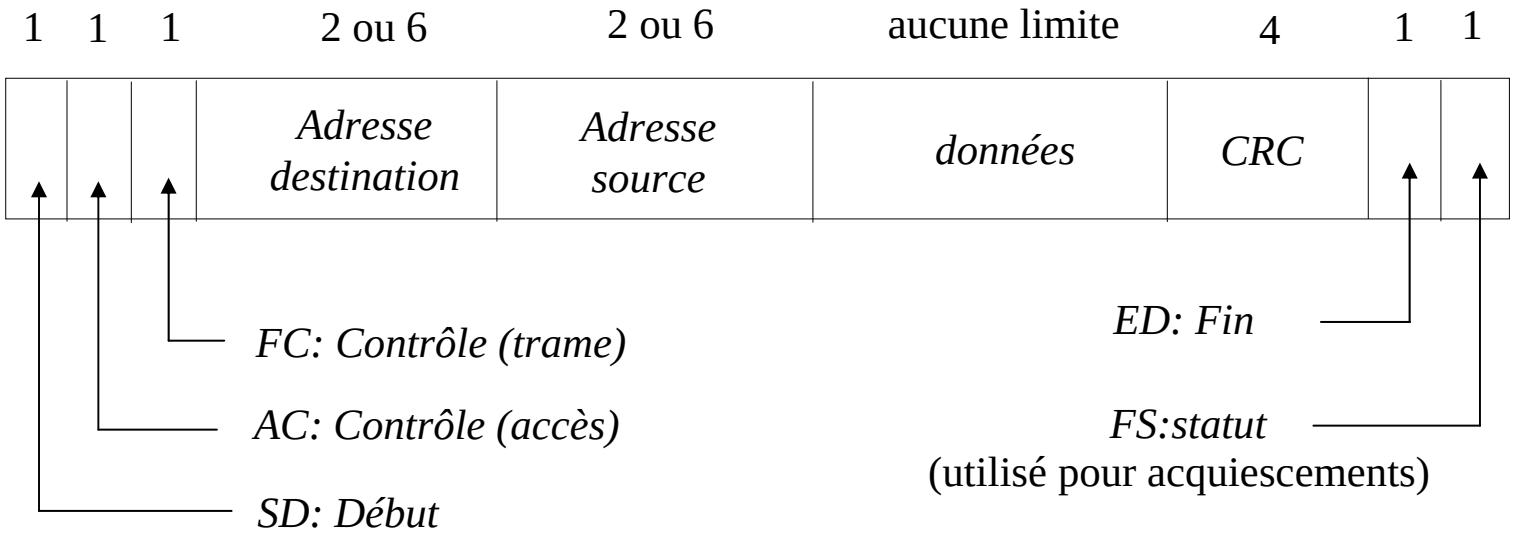
La transmission suit le délimiteur

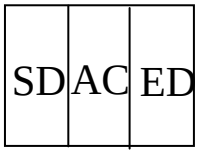
Les autres stations retransmettent (copie locale si destination)

L'émetteur "détruit" l'information qui a fait un tour complet

L'émetteur libère le jeton

Trame IEEE 802.5





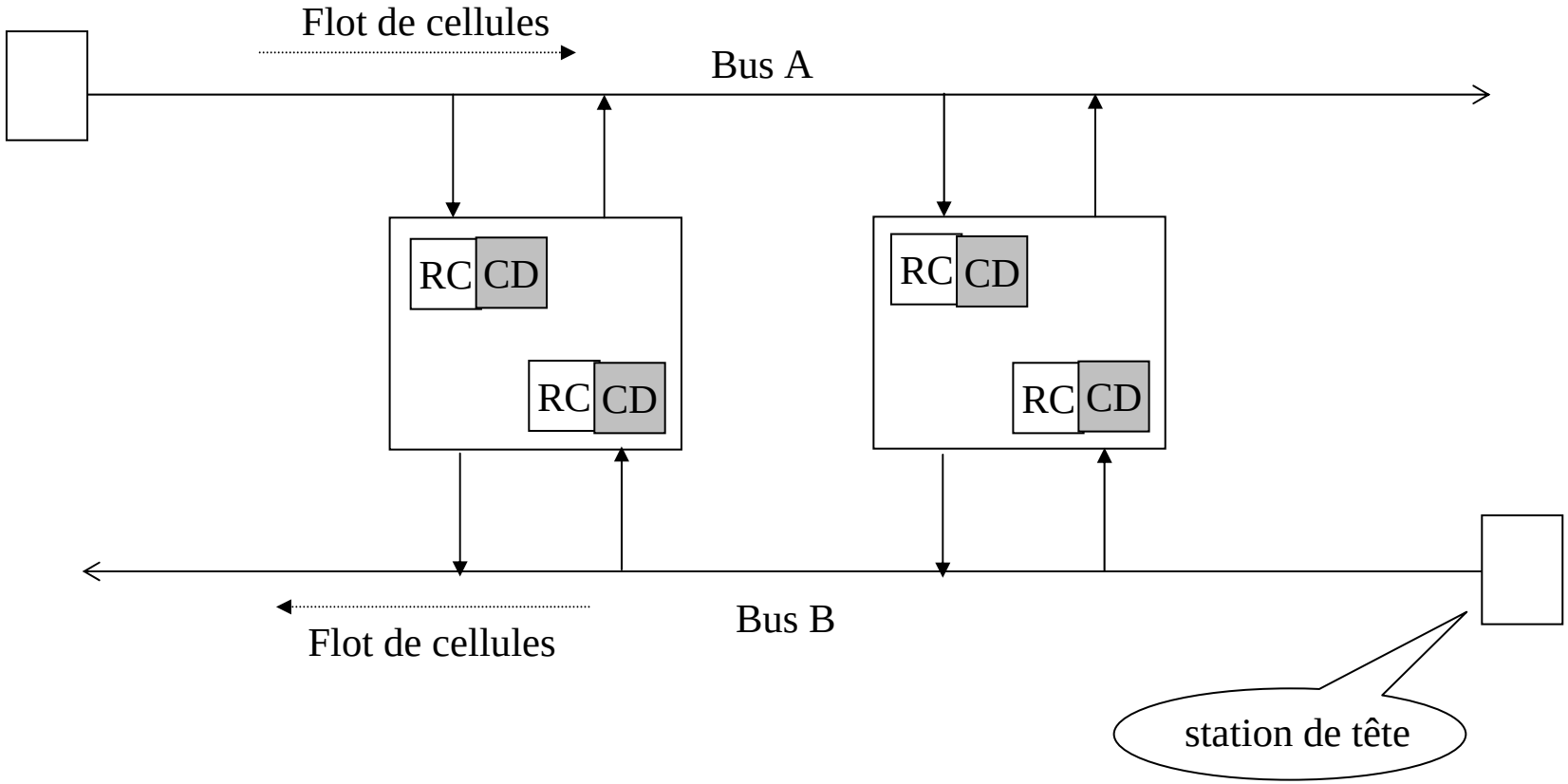
Jeton : 1 seul bit dans le champs AC le distingue d'un début de trame

IEEE 802.6
Distributed Queue Dual Bus

- Deux bus unidirectionnels
- Trames de taille fixe: *cellules* de 53 octets
- Réservation sur un bus pour transmettre sur l'autre bus
  → mécanisme *distribué* qui réalise une file d'attente *globale*
- Utilisé dans les réseaux métropolitains (MAN)

IEEE 802.6

Topologie



IEEE 802.6
Protocole

- Pour chaque bus, une station maintient 2 compteurs:
  - → RC: “Request counter”
  - → CD: “Countdown”

- Le compteur RC du bus A:
  - → fait un bilan pour toutes les trames sur bus A et B
  - → RC = RC+1 si trame contient une réservation sur bus B
  - → RC = RC-1 si trame occupée sur bus A

- Le compteur RC sert lors qu’on a une trame à transmettre

IEEE 802.6

Transmission d'une trame

- Lorsqu'une station veut transmettre sur le bus A (inverser pour le bus B):

  (1) Une réservation est placée sur le bus B
  → bit spécial mis à 1 dans la prochaine cellule

  (2) CD = RC (les deux compteurs du bus A)

  (3) Pour chaque nouvelle cellule libre sur le bus A
  → CD = CD-1

  (4) Lorsque CD=0
  → on transmet dans la prochaine cellule libre
  → on indique que la cellule est occupée (bit spécial)

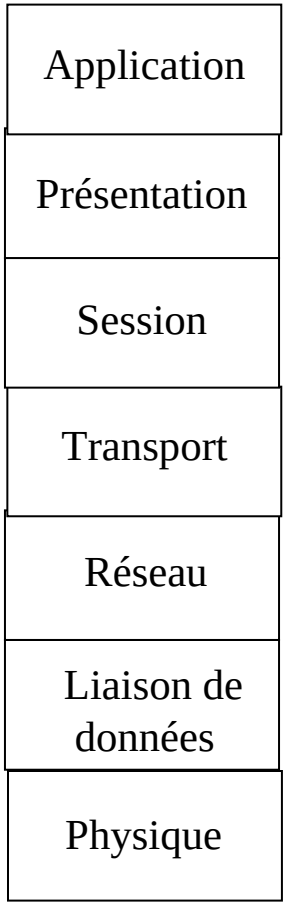
FDDI

Réseau local à haute vitesse

- "Fiber Distributed Dual Interface" (support à fibre optique)
- Protocole très proche de IEEE 802.5
- Double anneau à jeton, reconfigurable
  - → sens horaire sur le premier anneau
  - → sens anti-horaire sur le second
- Encodage 4/5
  - → 4 bits encodés en 5 bits
  - → il reste donc 16 codes possibles de 5 bits pour le contrôle

Les ponts

Interconnexion de réseaux locaux



Passerelle

Routeur

Pont

Répéteur

Le pont ne manipule que des *adresses physiques* (interface MAC)

Il permet de combiner plusieurs LANs en un seul *réseau logique*

→ permet de mieux gérer la charge

Les ponts permettent aussi de couvrir des distances plus importantes

Le Pont

Un répéteur intelligent

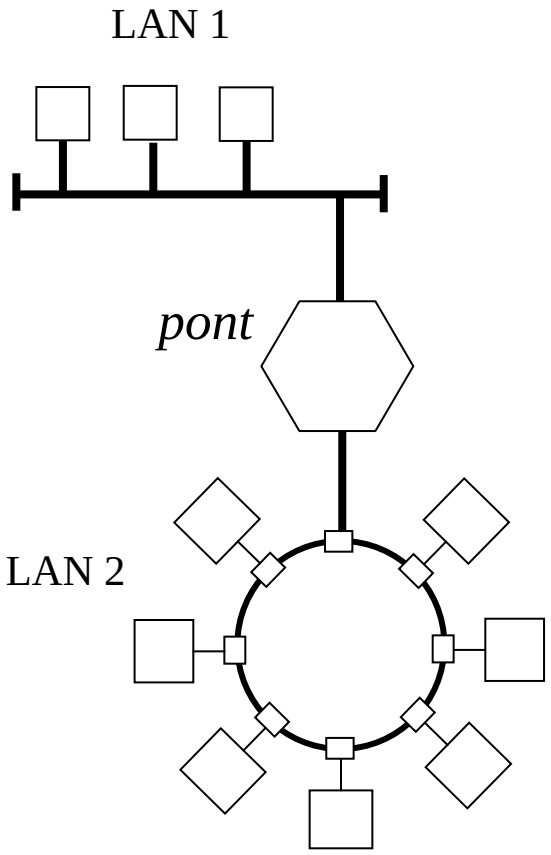


Table interne du pont

Adresses physiques sur LAN 1	Adresses physiques sur LAN 2
x x x x x x *y y y y y y* *z z z z z z*	*a a a a a a* *b b b b b b* *c c c c c c* *d d d d d d*

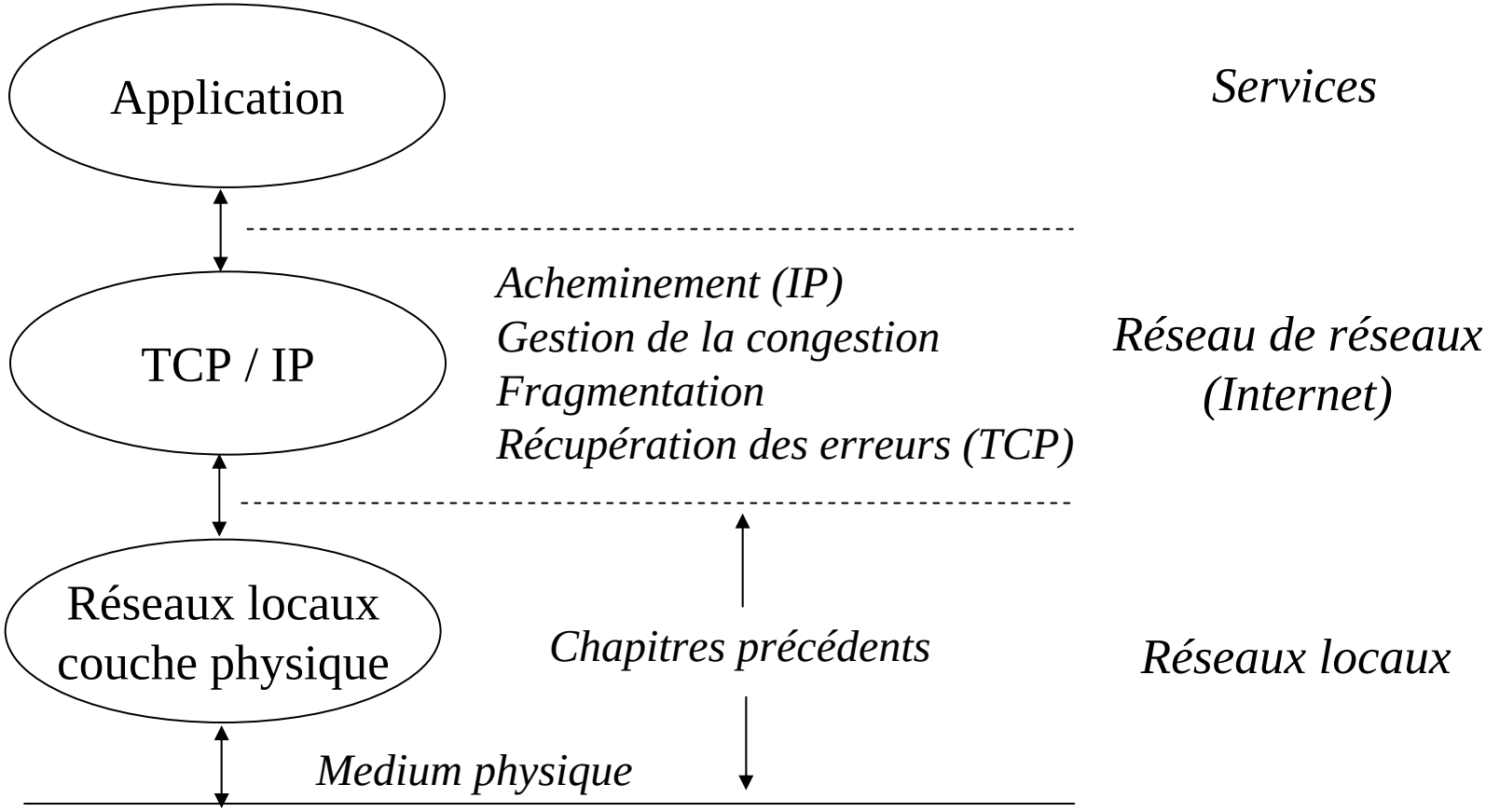
Le pont “observe” les trames qui circulent sur chaque LAN auquel il est connecté

Il ne transmet que les trames qui sont destinées à un autre LAN que celui d’où elles proviennent

Il peut mettre à jour sa table en observant les adresses *sources* des trames qui passent

- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

Un peu de recul



La couche réseau

- **Achemine les paquets** de la source à la destination (possiblement à travers une séquence de “routeurs”)
- Mécanismes d’acheminement (“routing”) qui réagissent à la congestion et aux conditions du réseau
- Un paquet comporte une entête où sont inscrites (entre autres) les **adresses** source et destination
- La couche réseau peut **fragmenter** les paquets si leur taille dépasse la capacité d’un réseau

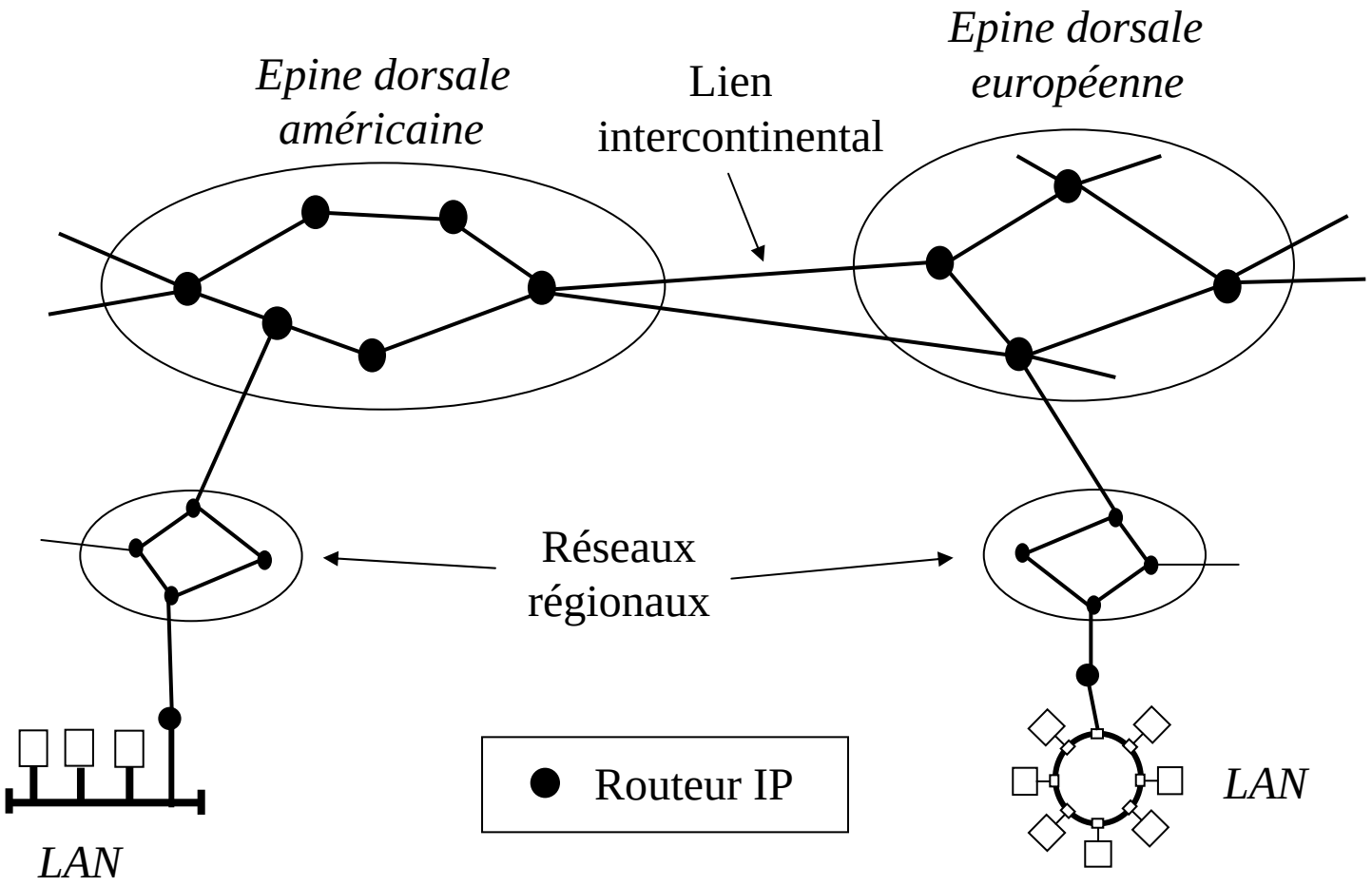
Types d'acheminement

- **Datagrammes**
  - → service sans connexion
  - → chaque paquet est acheminé indépendemment
  - → pas de guarantie de livraison
  - → Exemple: couche Réseau sur **Internet**

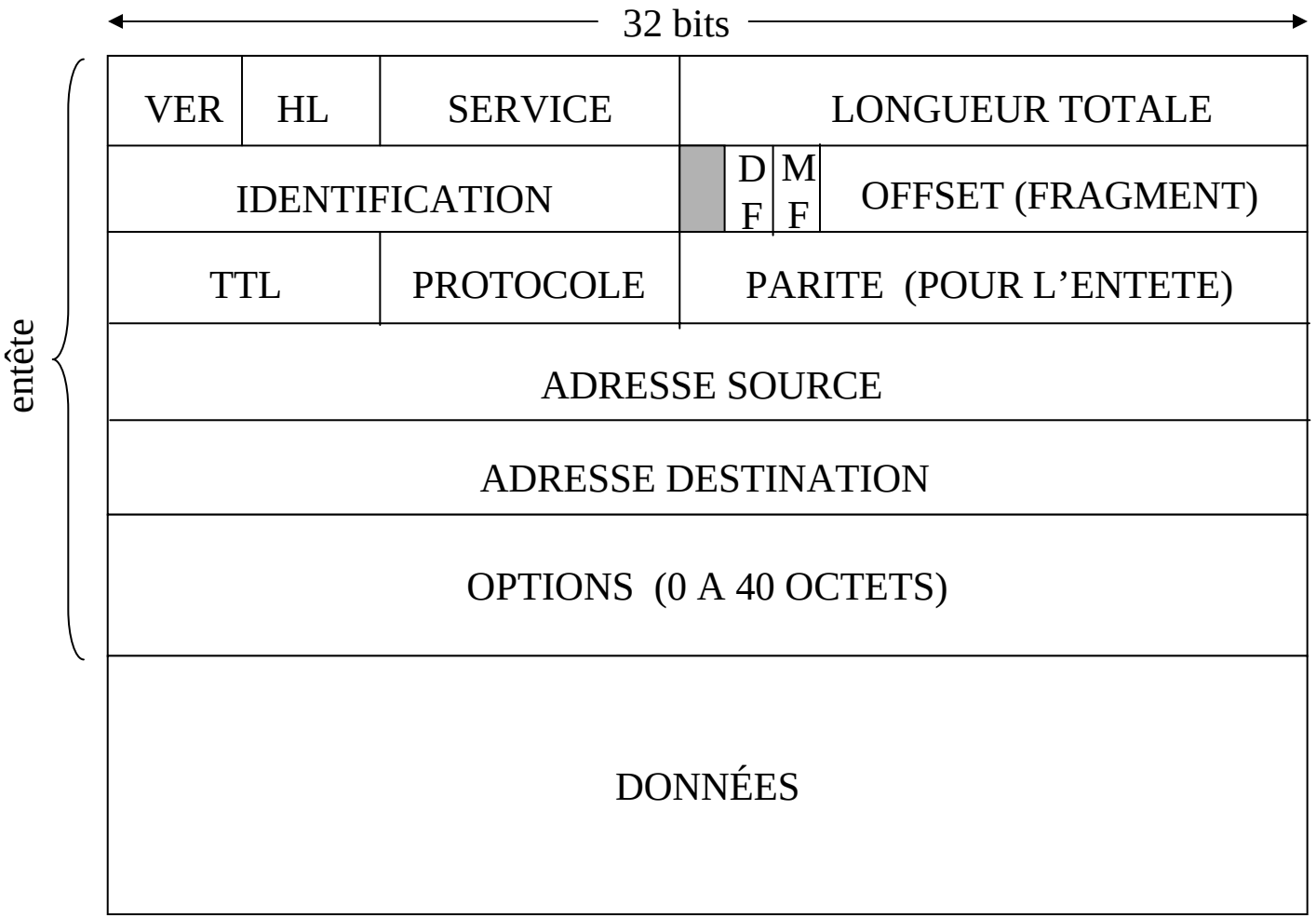
- **Circuits virtuels**
  - → service avec connexion
  - → nécessité d'établir (négocier) une connexion d'abord
  - → les paquets empruntent tous le même chemin
  - → les paquets sont livrés dans l'ordre
  - → Exemple: réseau **ATM**

La couche réseau sur Internet

Un réseau de réseaux



Le datagramme IP



Les champs de l'entête IP

- **VER** : (**4 bits**) version du protocole IP
  - → actuellement, version 4
  - → IP prochaine génération sera la version 6 (Ipv6 ou IPng)

- **HL** : (**4 bits**) longueur de l'entête IP, en mots de 32 bits
  - → minimum = 5
  - → maximum = 15 (limite le champs OPTIONS à 40 octets)

- **SERVICE** : (**8 bits**) Type de service
  - → niveaux de priorité
  - → les routeurs actuels n'utilisent PAS ce champs

- **LONGUEUR TOTALE** : (**16 bits**)
  - → nombre d'octets d'entête ET de données
  - → maximum = 65535 octets

Les champs de l'entête IP (suite)

- **IDENTIFICATION** : (**16 bits**)
  - → Permet de rassembler des fragments à la réception
  - → Tous les fragments d'un même datagramme ont le même no. d'identification

- **DF** : (**1 bit**) "don't fragment"
  - → Indique si on peut fragmenter ou non le datagramme

- **MF** : (**1 bit**) "more fragments"
  - → Tous les fragments d'un datagramme, sauf le dernier fragment, ont ce bit mis à *1* (ou *set*). Equivaut à un *EOF*.

- **OFFSET** : (**13 bits**)
  - → Indique le numéro d'un fragment dans un datagramme fragmenté.
  - → Nombre maximum de fragments = 8192.

Les champs de l'entête IP (suite)

- **TTL** : (**8 bits**) "Time to Live"
  - → Compte à rebours: nombre de sauts (routeurs) d'un datagramme
  - → Le datagramme est rejeté sur TTL = 0
    et une indication est envoyée à l'émetteur

- **PROTOCOLE** : (**8 bits**)
  - → Indique à quel protocole de transport (TCP, UDP, …) remettre le paquet à l'arrivée
  - → Numéros de protocoles: définis dans RFC 1700

- **PARITE** : (**16 bits**)
  - → Bits de détection d'erreur
  - → Appliqués à l'entête uniquement
  - → Somme complément-à-1 des mots de 16 bits de l'entête
  - → Calculé à chaque routeur

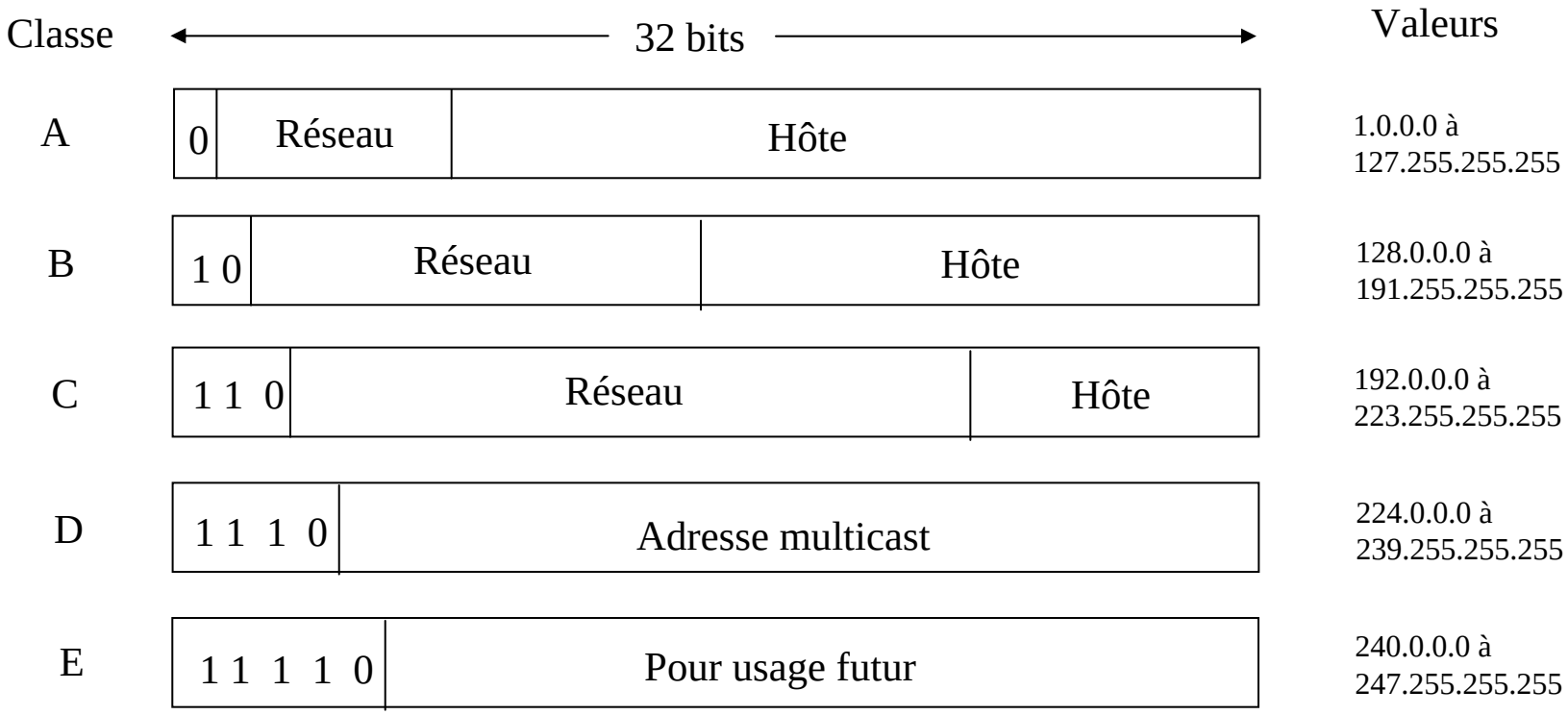
Les champs de l'entête IP (suite)

- **ADRESSE SOURCE** : (**32 bits**)
  → Adresse IP de la source

- **ADRESSE DESTINATION** : (**32 bits**)
  → Adresse IP de la destination

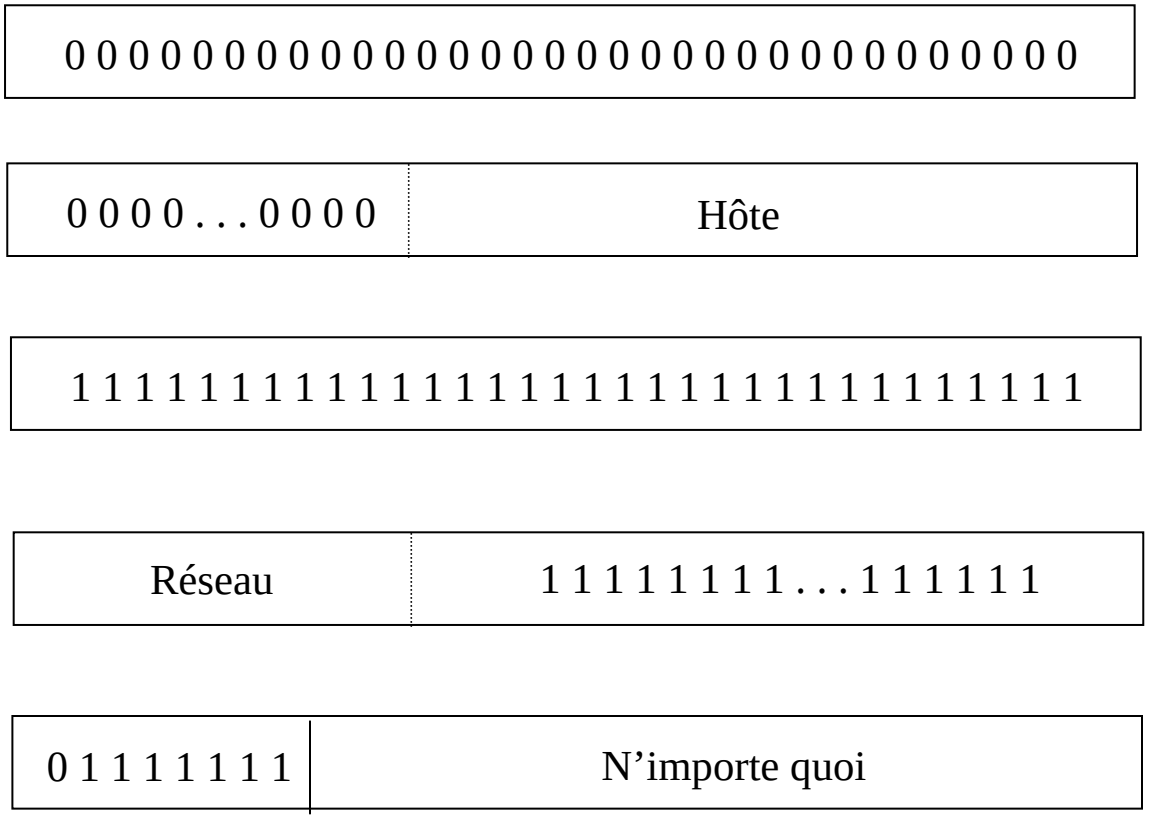
- **OPTIONS** : (**0 à 40 octets**)
  → Champs de longueur variable
  → Chaque option débute par un octet précis
  → Doit être un multiple entier de 4 octets
  → Permet de :
    - spécifier le niveau de sécurité
    - spécifier une route donnée (une séquence de routeurs)
    - enregistrer une séquence de routeurs (et l'heure de passage…)

Les adresses IP



⇒ Les adresses IP sont accordées par le NIC (Network Information Center)

Adresses spéciales



Machine locale
→ utilisé lors de la mise sous tension

Un hôte sur le réseau local

Broadcast sur le réseau local

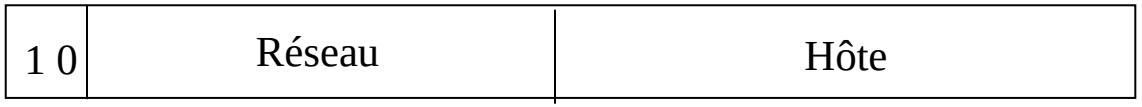
Broadcast sur le réseau spécifié

"Loopback"
→ aucune transmission

Les sous-réseaux

“subnets”

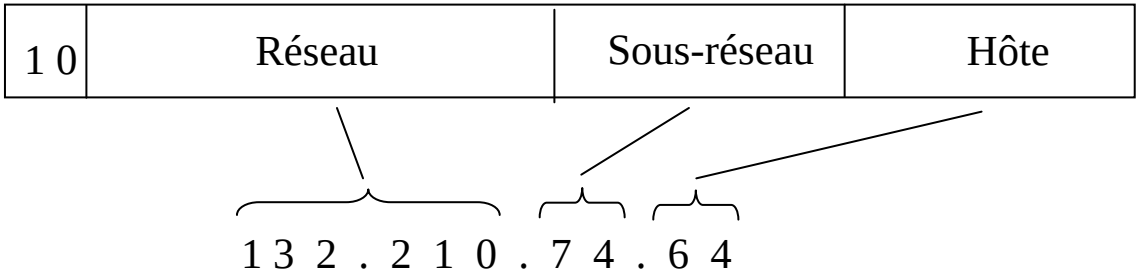
Adresses allouées par le NIC



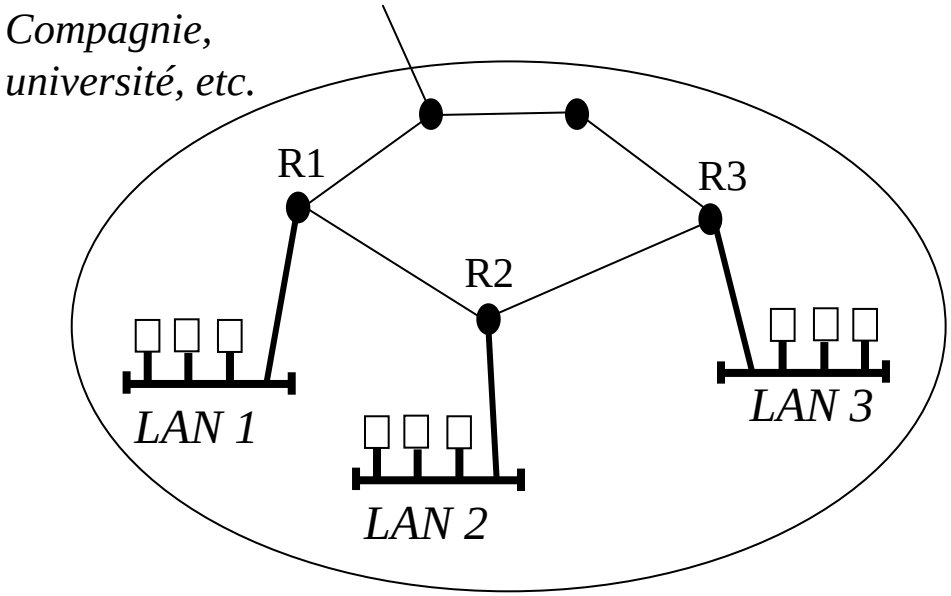
Masque de sous-réseau (géré par l’organisation elle-même)

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

Subdivision en sous-réseaux (l’adresse de sous-réseau n’a de sens qu’à l’intérieur de l’organisation)



Utilisation du masque



Vu de l'extérieur, tous ces réseaux locaux (LANs) ont la même adresse réseau (par exemple, 132.210.x.y)

Le routeur R3 a une table qui indique:

- comment rejoindre R1, R2
- l'adresse physique des hôtes sur LAN3

Lorsqu'un paquet arrive à R3, l'adresse destination et le masque de sous-réseau sont multipliés (ET bit à bit)

- le résultat est l'adresse de sous-réseau de la destination

ARP

Address Resolution Protocol (RFC 826)

- adresse IP $\Rightarrow$ adresse MAC
- L'adresse IP est une adresse *symbolique*
- Pour transmettre un paquet, la couche liaison de données a besoin d'une adresse *physique*
  - $\rightarrow$ adresse de l'interface MAC (6 octets sur Ethernet)
  - $\rightarrow$ l'adresse physique est une adresse *constante* déterminée par le manufacturier
  - $\rightarrow$ l'interface MAC ne manipule *jamais* d'adresse IP

ARP

Exemple pour hôtes sur le même LAN

(1) L'hôte 1 transmet un message broadcast spécial:
→ "Qui a l'adresse IP 132.210.74.64 ? "

(2) Ce message est vu par tous les hôtes sur le LAN
→ Chacun vérifie s'il a l'adresse IP 132.210.74.64

(3) L'hôte destination (disons l'hôte 2)
→ retourne son adresse physique *ADDR_PHYS* à l'hôte 1 (l'adresse physique du demandeur était dans le champs *adresse source* de la trame ARP)

(4) L'hôte 1 peut maintenant transmettre des trames à l'hôte 2

RARP

Reverse ARP (RFC 903)

- adresse MAC $\Rightarrow$ adresse IP
- un hôte veut connaître *sa propre adresse IP*
- Par exemple: démarrage d'une station de travail
  sans disque local
- BOOTP (RFC 951, 1048, 1084)
  - $\rightarrow$ utilise UDP (couche transport)
    au lieu de trames L.D.
  - $\rightarrow$ permet d'obtenir aussi les adresses IP de serveurs
    ainsi que le masque de sous-réseau.

Calcul du “meilleur chemin”

- Table d’un routeur :
  - liste de chemins possibles pour une destination donnée

- Question :
  - Comment déterminer automatiquement, et de façon dynamique, le *meilleur chemin* parmi un ensemble de routeurs ?

- Données du problème :
  - distances (débit, délai, etc.) *point-à-point* entre les routeurs du réseau donné

- Algorithmes: Bellman-Ford, Dijkstra

Algorithme de Bellman-Ford

- Algorithme itératif

- Deux versions :
  (1) distribuée (“distance vector routing”)
  - réagit plus lentement aux changements rapides de topologie

  (2) centralisée
  - demande plus d’échange d’information entre les routeurs

- version distribuée était utilisée dans ARPANET (RIP)

Bellman-Ford distribué

- Chaque routeur effectue le calcul *localement*, avec une information *limitée*

- Information disponible pour un routeur:
  - coût pour atteindre les routeurs auxquels il est *directement connecté*
  - coût total entre la destination et chaque routeur auquel il est connecté

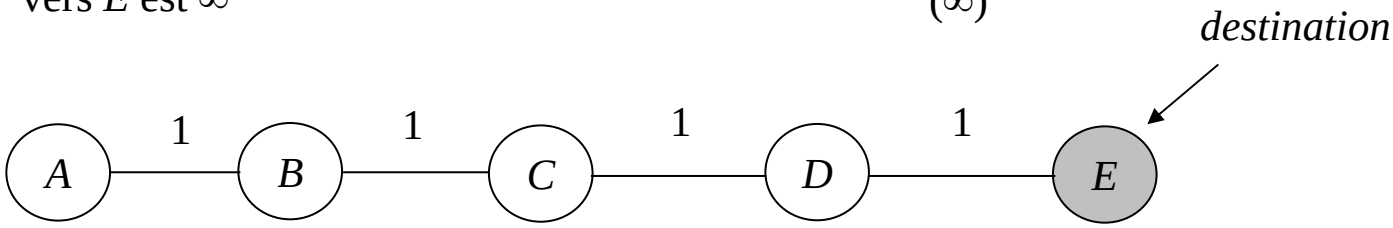
    $\rightarrow$ échange d'information sur les liens directs

Bellman-Ford distribué

exemple

Les autres routeurs estiment initialement que le coût vers *E* est ∞

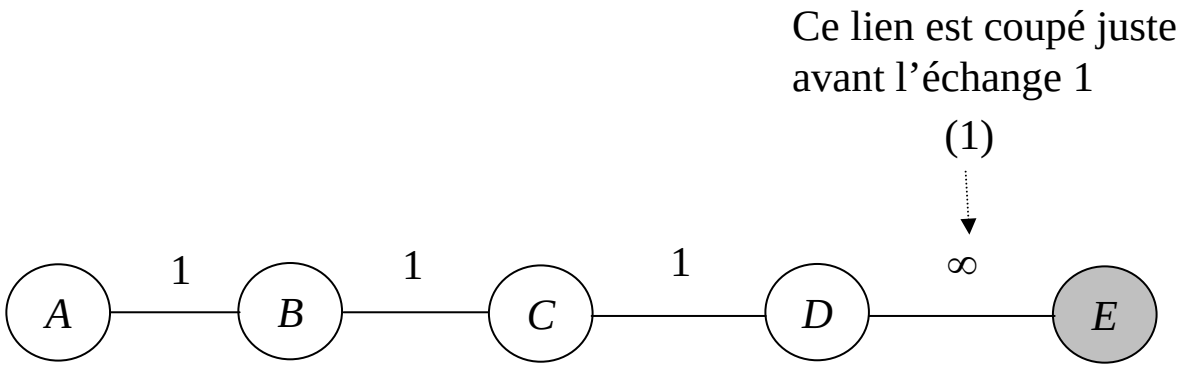
Supposons qu'initialement, *E* n'est pas en fonction (∞)



∞	∞	∞	∞	initialement
∞	∞	∞	1	après 1 échange
∞	∞	2	1	après 2 échanges
∞	3	2	1	après 3 échanges
4	3	2	1	après 4 échanges
4	3	2	1	après 5 échanges
. . .	. . .	. . .		. . .

Bellman-Ford distribué

réaction lente aux mauvaises nouvelles

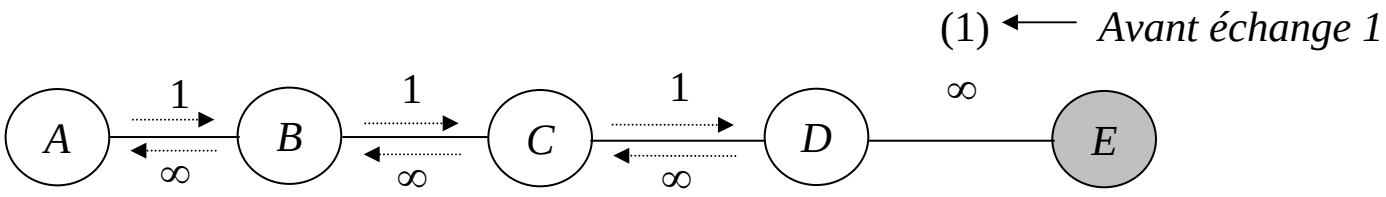


4	3	2	1	initialement
4	3	2	3	après 1 échange
4	3	4	3	après 2 échanges
4	5	4	5	après 3 échanges
6	5	6	5	après 4 échanges
6	7	6	7	après 5 échanges
8	7	8	7	après 6 échanges
. . .	. . .	. . .		. . .

(*Ca risque d'être long …)*

Bellman-Ford distribué

accélération de la convergence

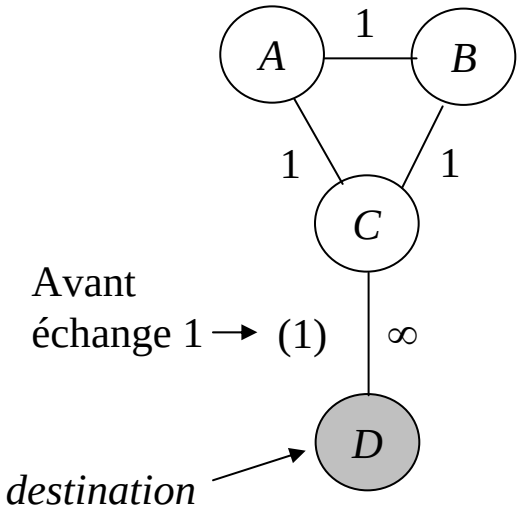


4	3	2	1	initialement
4	3	2	∞	après 1 échange
4	3	∞	∞	après 2 échanges
4	∞	∞	∞	après 3 échanges
∞	∞	∞	∞	après 4 échanges
∞	∞	∞	∞	après 5 échanges
∞	∞	∞	∞	après 6 échanges
. . .	. . .	. . .		. . .

La distance minimale vers *E* n'est pas rapportée par un routeur sur la ligne où il envoie les paquets pour rejoinre *E*… En fait, cette distance est rapportée comme étant ∞.

Bellman-Ford distribué

cas où la technique d'accélération échoue

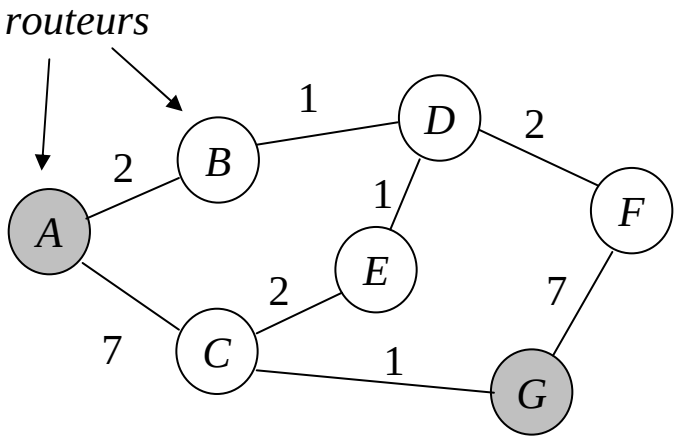


Distance vers *D*
vue de

A	*B*	*C*	
2	2	1	initialement
2	2	∞	après 1 échange
3	3	∞	après 2 échanges
4	4	∞	après 3 échanges
5	5	∞	après 4 échanges
. . .			. . .

réaction lente pour les noeuds *A* et *B*

Bellman-Ford centralisé
exemple



Notation :

d(A,B) : coût entre deux noeuds reliés directement (ici, A et B)

d(C) : plus courte distance entre le noeud C et la destination G

Le routeur *A* veut connaître le plus court chemin pour atteindre le routeur *G*.

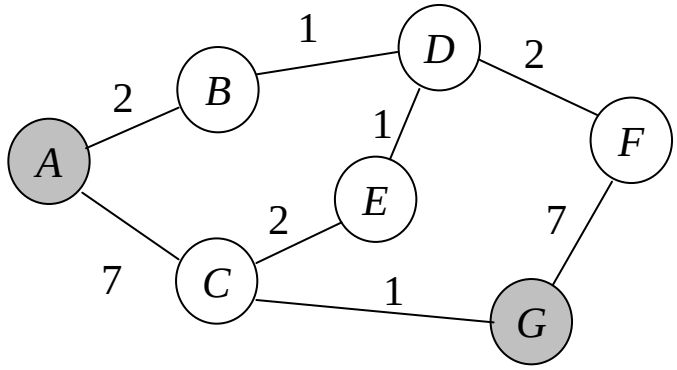
Les nombres indiquent le coût (par exemple, le délai) associé à chaque lien. On suppose des liens bi-directionnels avec le même coût dans les 2 sens.

Note: Chaque routeur envoit à *A* le coût observé avec chacun de ses voisins immédiats (d(*x*,*y*)) (p.e., *B*, *E* et *F* pour le routeur *D*)

Bellman-Ford centralisé

Mise en équations

Réseau:

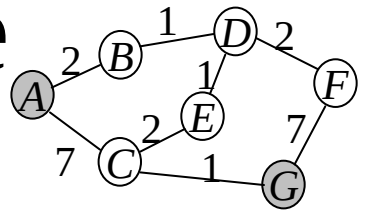


Equations:

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} d(A,B) + d(B) \ , \ d(A,C) + d(C) \\ d(B,A) + d(A) \ , \ d(B,D) + d(D) \\ d(C,A) + d(A) \ , \ d(C,E) + d(E) \ , \ d(C) \\ d(D,B) + d(B) \ , \ d(D,E) + d(E) \ , \ d(D,F) + d(F) \\ d(E,C) + d(C) \ , \ d(E,D) + d(D) \\ d(F,D) + d(D) \ , \ d(F) \end{pmatrix}$$

Initialement : $d(A) = d(B) = d(C) = d(D) = d(E) = d(F) = \infty$

Bellman-Ford centralisé



itérations

1

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+\infty\ ,\ 7+\infty \\ 2+\infty\ ,\ 1+\infty \\ 7+\infty\ ,\ 2+\infty\ ,\ 1 \\ 1+\infty\ ,\ 1+\infty\ ,\ 2+\infty \\ 2+\infty\ ,\ 1+\infty \\ 2+\infty\ ,\ 7 \end{pmatrix} = \begin{pmatrix} \infty \\ \infty \\ 1 \\ \infty \\ \infty \\ 7 \end{pmatrix}$$

2

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+\infty\ ,\ 7+1 \\ 2+\infty\ ,\ 1+\infty \\ 7+\infty\ ,\ 2+\infty\ ,\ 1 \\ 1+\infty\ ,\ 1+\infty\ ,\ 2+7 \\ 2+1\ ,\ 1+\infty \\ 2+\infty\ ,\ 7 \end{pmatrix} = \begin{pmatrix} 8 \\ \infty \\ 1 \\ 9 \\ 3 \\ 7 \end{pmatrix}$$

3

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+\infty\ ,\ 7+1 \\ 2+8\ ,\ 1+9 \\ 7+8\ ,\ 2+3\ ,\ 1 \\ 1+\infty\ ,\ 1+3\ ,\ 2+7 \\ 2+1\ ,\ 1+9 \\ 2+9\ ,\ 7 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \\ 1 \\ 4 \\ 3 \\ 7 \end{pmatrix}$$

4

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+10\ ,\ 7+1 \\ 2+8\ ,\ 1+4 \\ 7+8\ ,\ 2+3\ ,\ 1 \\ 1+10\ ,\ 1+3\ ,\ 2+7 \\ 2+1\ ,\ 1+4 \\ 2+4\ ,\ 7 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \\ 1 \\ 4 \\ 3 \\ 6 \end{pmatrix}$$

5

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+5\ ,\ 7+1 \\ 2+8\ ,\ 1+4 \\ 7+8\ ,\ 2+3\ ,\ 1 \\ 1+5\ ,\ 1+3\ ,\ 2+6 \\ 2+1\ ,\ 1+4 \\ 2+4\ ,\ 7 \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \\ 1 \\ 4 \\ 3 \\ 6 \end{pmatrix}$$

6

$$\begin{pmatrix} d(A) \\ d(B) \\ d(C) \\ d(D) \\ d(E) \\ d(F) \end{pmatrix} = \min \begin{pmatrix} 2+5\ ,\ 7+1 \\ 2+7\ ,\ 1+4 \\ 7+7\ ,\ 2+3\ ,\ 1 \\ 1+5\ ,\ 1+3\ ,\ 2+6 \\ 2+1\ ,\ 1+4 \\ 2+4\ ,\ 7 \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \\ 1 \\ 4 \\ 3 \\ 6 \end{pmatrix}$$

Bellman-Ford centralisé

après convergence

- Le routeur *A* connaît *pour chacun des autres routeurs*
  - la plus courte distance (coût) vers *G*
  - le prochain routeur pour y parvenir

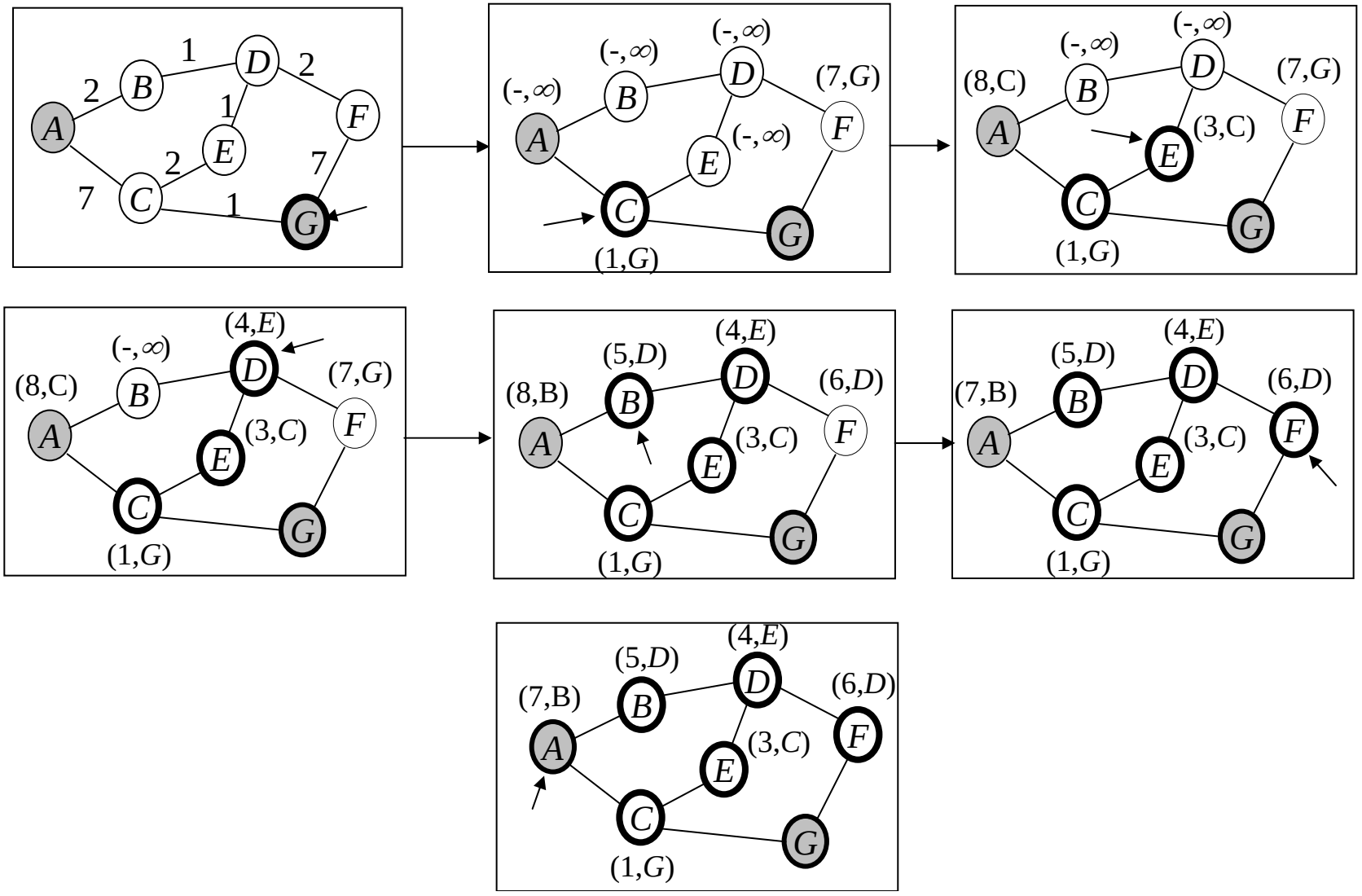
- *A* peut retransmettre ces résultats aux autres routeurs
  (ou encore, chaque noeud peut faire le calcul individuellement)

- Pour transmettre de *A* vers *G*:
  - Le meilleur chemin est

$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow G$$

  - avec le coût associé: 7

Algorithme de Dijkstra



Algorithme de Dijkstra
résumé

- Donne, pour chaque routeur, la distance minimale pour atteindre la destination (ici, *G*), ainsi que le prochain routeur sur la route.

- Doit être répété *avec chaque routeur comme destination*
  $\rightarrow$ permet de savoir comment atteindre chaque routeur

- Les routeurs s'échangent l'information locale
  $\rightarrow$ le coût (nombre) sur chaque lien point-à-point
  $\rightarrow$ base de données distribuée

- Même résultat que Bellman-Ford Centralisé

“Link state routing”

Périodiquement, chaque routeur d’un domaine:

(1) découvre qui sont ses voisins (*HELLO*)

(2) mesure le délai (ou le coût) sur les liens correspondants

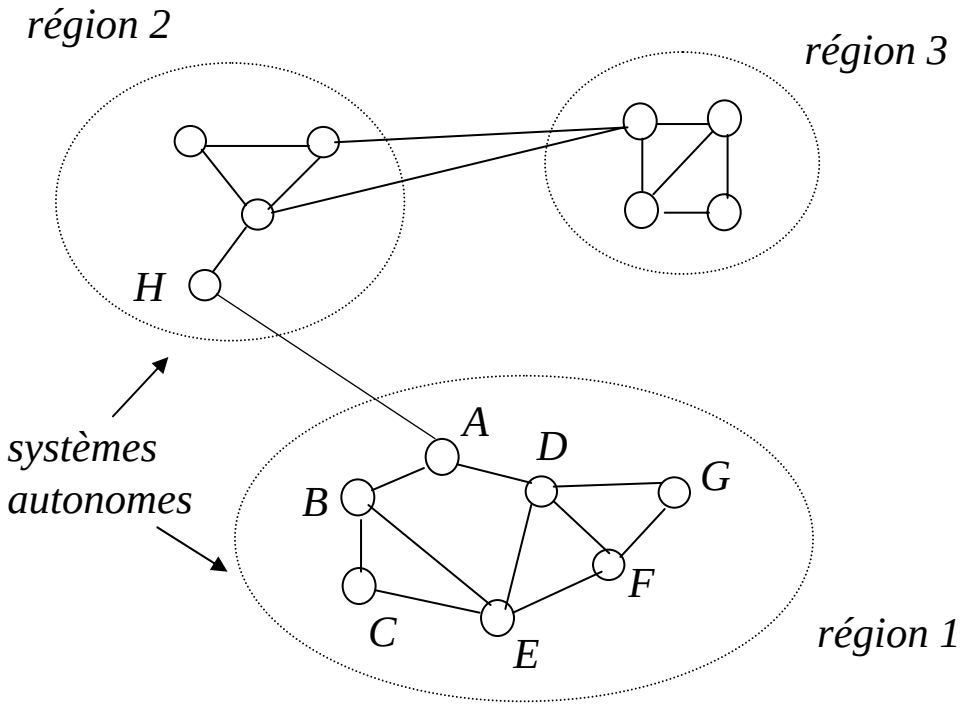
(3) construit un paquet contenant cette information

(4) transmet cette info aux autres routeurs du domaine (“*flooding*”)

(5) calcule le chemin le plus court vers les autres routeurs

Hiérarchie

- Le nombre de routeurs est *trop grand*
  - → les tables d'acheminement "explosent"
  - → le temps pour chercher la meilleure route aussi



Ici, le routeur *A* de la région 1
ne maintient une table d'acheminement
que pour les routeurs de sa région
(*B* à *G*), et pour le routeur *H*.

Pour N routeurs au total, le nombre
optimal de niveaux pour minimiser
la taille des tables est

$$\ln(N)$$

Protocoles d'acheminement

- Internet = collection de systèmes (réseaux) autonomes
  $\rightarrow$ ex.: Université, compagnie, ...

- Acheminement à l'intérieur d'un système autonome (SA) :
  "Interior Gateway Routing Protocol"
  - **OSPF**: "Open Shortest Path First" (norme depuis 1990)
  - Meilleures routes: algorithme de Dijkstra

- Acheminement entre des systèmes autonomes :
  "Exterior Gateway Routing Protocol"
  - **BGP**: "Border Gateway Protocol"

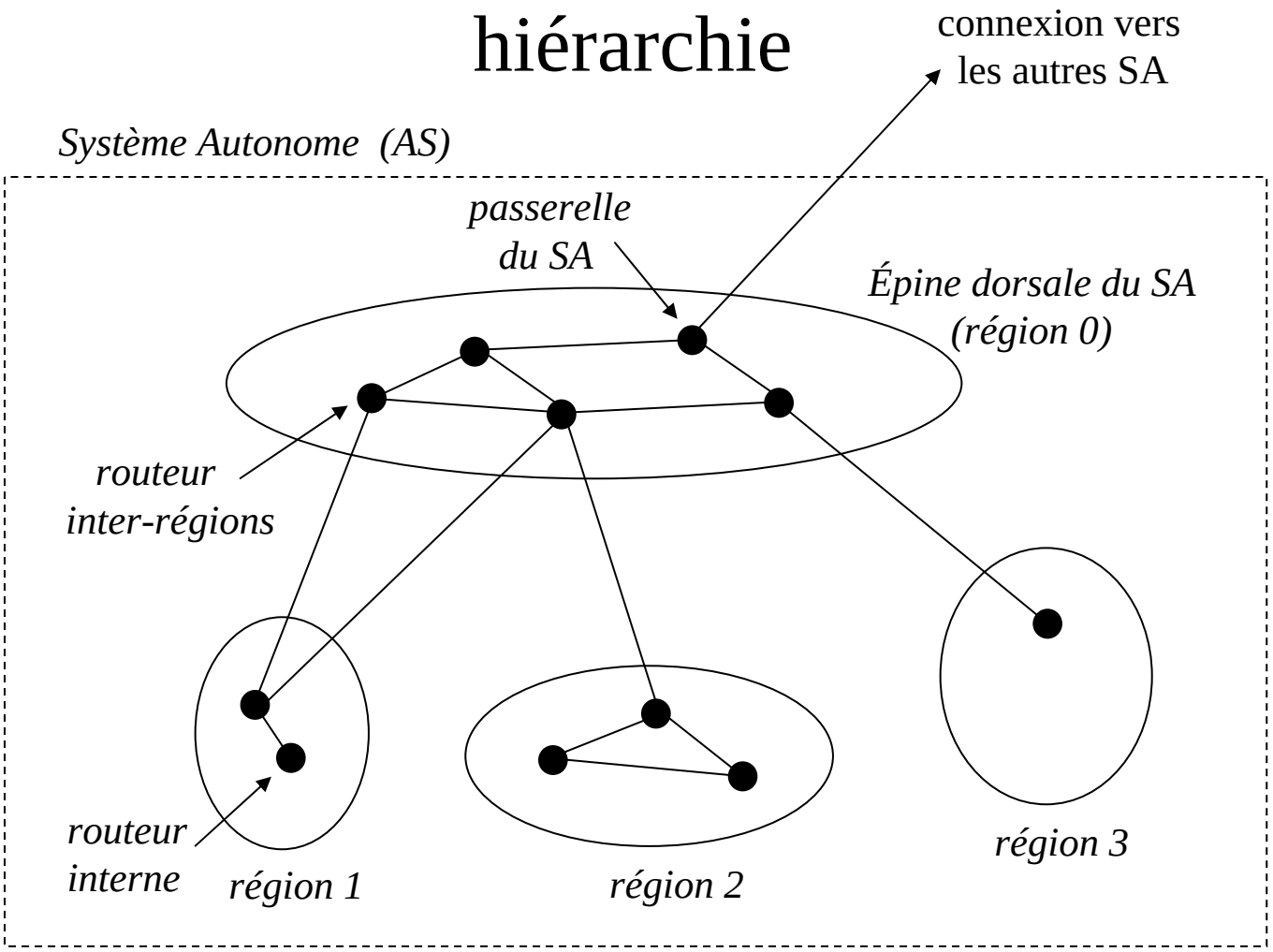
OSPF

Open Shortest Path First (RFC 1247)

- Acheminement selon distance, nombre de sauts, délai, etc.
- Adaptation rapide à la topologie (congestion, …)
- Différents types de services (temps-réel par exemple)
- Répartition de la charge
  → utilise les *n* meilleures routes
- Très hyérarchisé (à l'intérieur même d'un SA)
- Niveau accru de robustesse (sécurité)

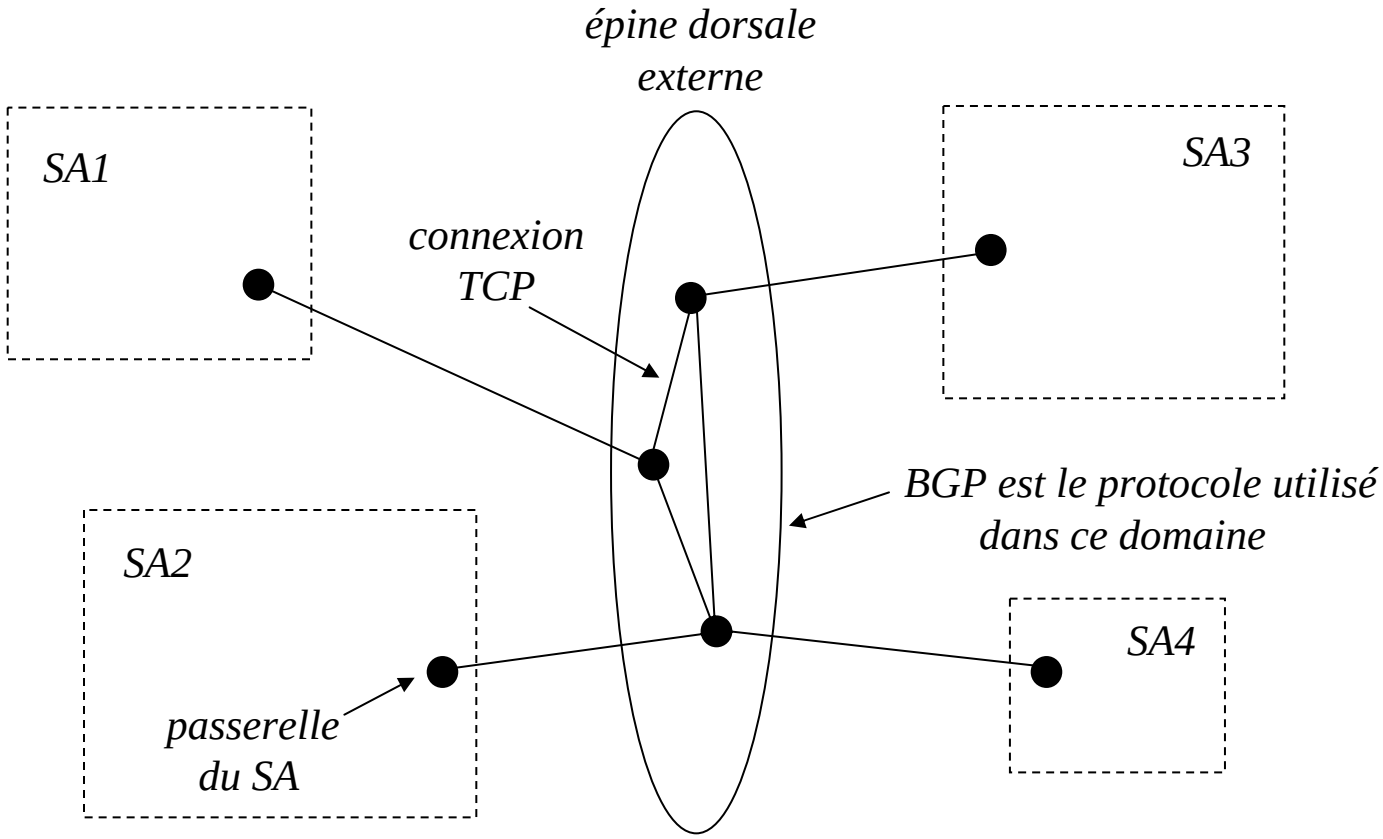
OSPF

hiérarchie



BGP

"Border Gateway Protocol" (RFC 1654)



BGP

- Doit prendre en compte diverses contraintes
  - politiques
  - économiques
  - relativement à la sécurité
  - etc.

- Ces politiques sont implémentées *manuellement*
  → elles ne font pas partie du protocole lui-même

- Optimisation d'une route: similaire à Bellman-Ford distribué
  → différence: les routeurs échangent les routes *complètes* (très robuste…)

Fragmentation

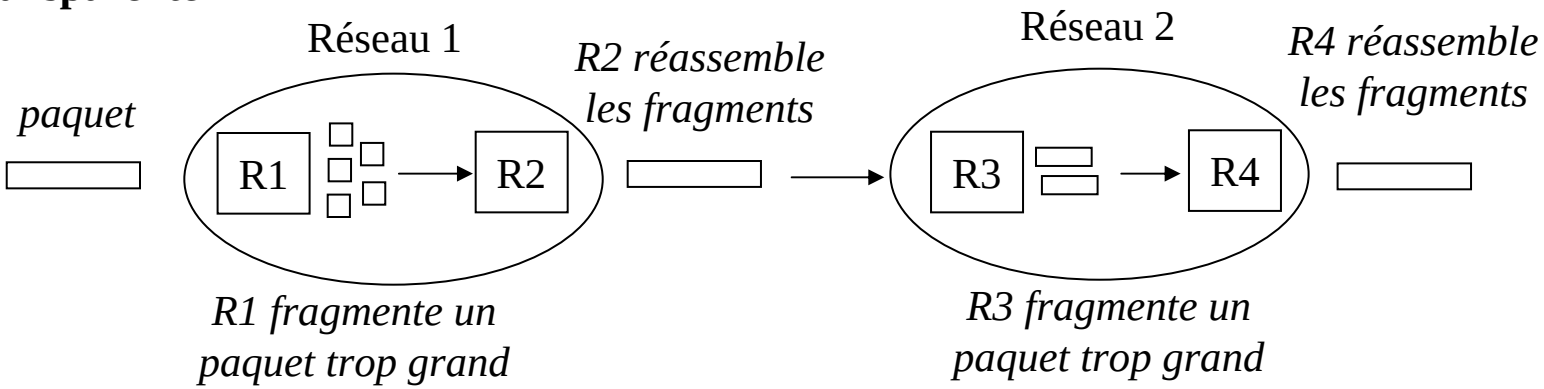
- Il est possible qu'un paquet transite à travers
  plusieurs types de réseaux

- Chaque réseau a ses contraintes, dont
  la *taille maximale d'une trame*
  (p.e. cellule ATM = 48 octets de données)

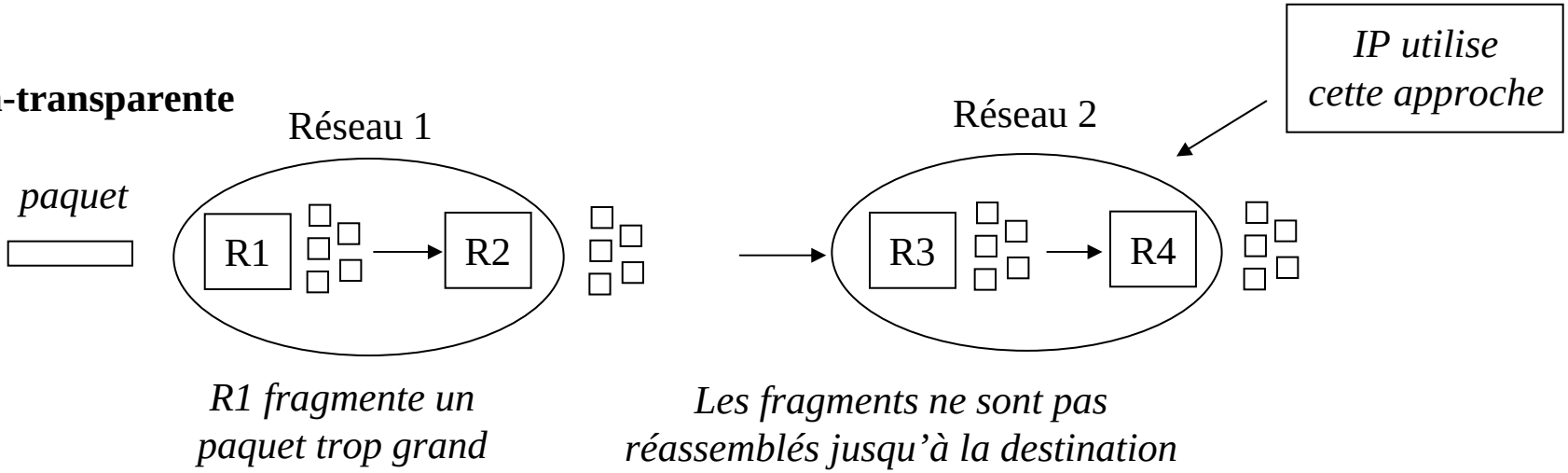
- Solution inévitable:
  $\rightarrow$ fragmentation des paquets

Fragmentation
deux approches

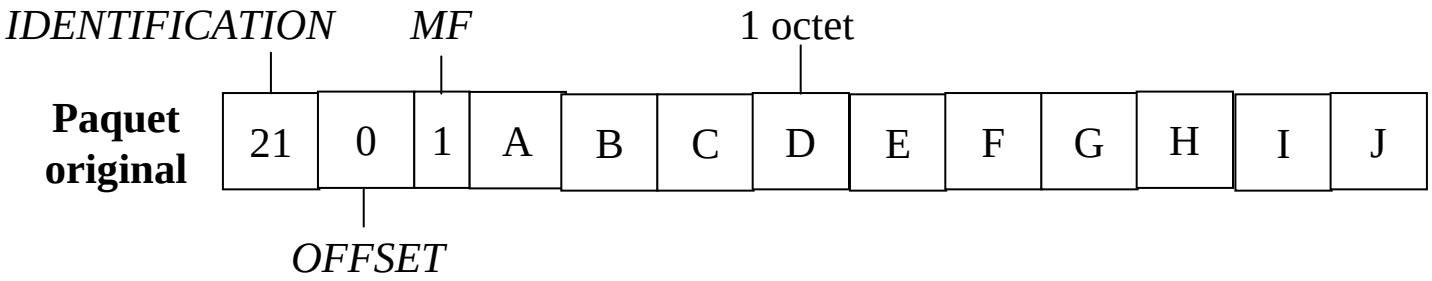
Transparente



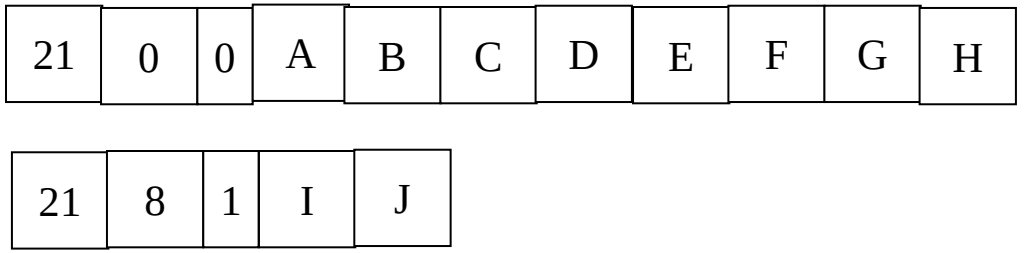
Non-transparente



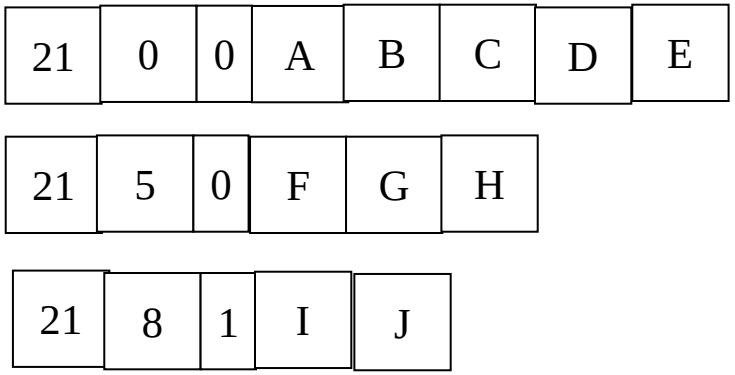
Numérotation des fragments



Fragments après passage dans un réseau avec une taille max = 8 octets de données



Fragments après passage dans un réseau avec une taille max = 5 octets de données



Des adresse limitées

- Le 100 000^{e} réseau (système autonome) a été connecté à l'Internet en 1996
- Problème: le nombre d'adresses IP disponibles diminue rapidement
- Les adresses de classe B sont les plus populaires (de loin)
- Cette division des adresses IP en *classes* est sous-optimale
  $\rightarrow$ des *millions* d'adresses ne seront jamais utilisées

CIDR

une solution temporaire

- CIDR = “Classless InterDomain Routing” (RFC 1519)

- **Idée**: allouer les quelques 2 millions de blocs d’adresses de classe C encore disponibles

- Plusieurs petits blocs *successifs* d’adresses de classe C
  → 256 hôtes par réseau → utilisation de *masques*

194.0.0.0 à 195.255.255.255	pour l’Europe
198.0.0.0 à 199.255.255.255	pour l’Amérique du Nord
200.0.0.0 à 201.255.255.255	pour l’Amérique centrale et du Sud
202.0.0.0 à 203.255.255.255	pour l’Asie et le Pacifique

IETF, 1990:
"Watch out, captain, she's gonna blow!"

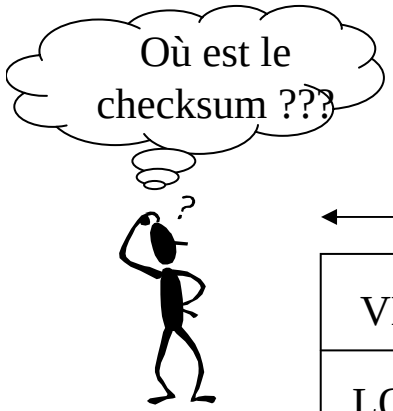
IPv6

SIPP
"Simple Internet Protocol Plus"

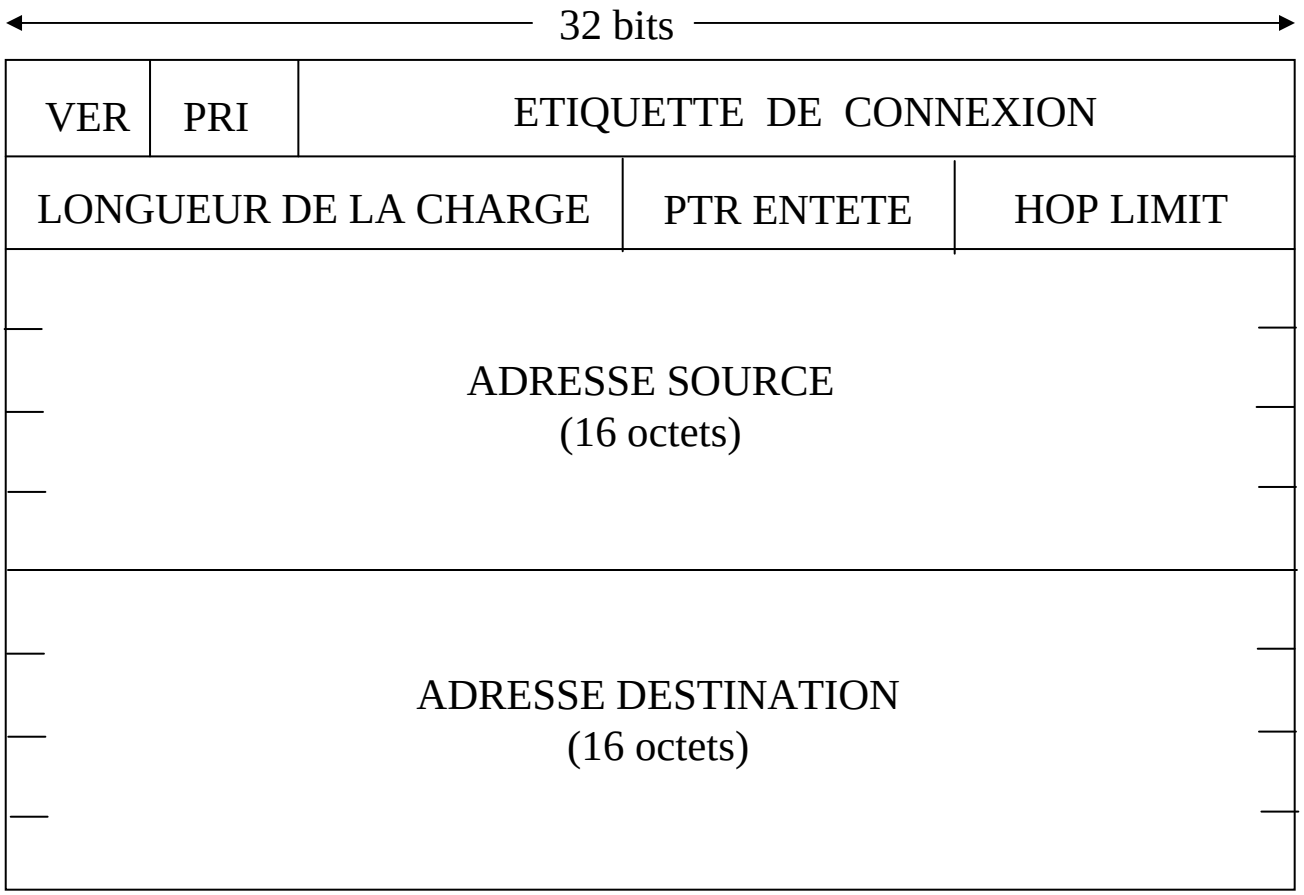
- Prochaine génération du protocole IP (actuellement, IPv4)

- Doit permettre à IPv4 d'exister pendant encore plusieurs années (transition douce…)

- Quelques particularités:
  - → adresses de 16 octets (au lieu de 4)
  - → permet de définir des *extensions* à l'entête
  - → protocole d'acheminement simplifié (plus rapide...)
  - → meilleure sécurité (identification, privacy)
  - → plusieurs types de services, dont temps-réel
  - → mobilité (hôte mobile / adresse fixe…)
  - → plus flexible pour les versions futures
  - → fragmentation permise *uniquement à la source*



Entête principale IPv6



Champs de l'entête IPv6

- **VER** : (**4 bits**) Version

- **PRI** : (**4 bits**) Priorité
  - → Indique le niveau de contôle qu'un routeur peut exercer sur ce paquet pour réguler la congestion sur le réseau
  - → 0..7 : faibles priorités, traffic qui n'est pas temps réel
  - → 8..15 : hautes priorités, traffic temps réel
  - → Exemples :
    - News : priorité 1
    - FTP : priorité 4
    - Telnet : priorité 6 (session à distance, interactive)

- **ETIQUETTE DE CONNEXION** (**24 bits**)
  - → Encore à l'étude
  - → Pour guarantir l'équivalent d'un circuit virtuel

Champs de l'entête IPv6 (suite)

- **LONGUEUR DE LA CHARGE** : (**16 bits**)
  → Nombre d'octets qui *suivent* l'entête principale

- **PTR ENTÊTE** (**8 bits**)
  → Indique comment interpréter ce qui suit immédiatement l'entête principale
    - 6 extensions à l'entête sont déjà définies
  → Si l'entête est la dernière entête IP (pas d'extension):
    - PTR ENTETE indique à quel protocole (TCP, UDP) remettre le paquet à l'arrivée (même rôle que PROTOCOLE dans IPv4)

- **HOP LIMIT** (**8 bits**)
  → Nombre de "sauts" maximum permis à un paquet avant qu'il ne soit retiré de la circulation

Adresses IPv6

Préfixe	Usage	Fraction
0 0 0 0 0 0 0 0	Réservé (incluant IPv4)	1 / 256
0 0 0 0 0 0 0 1	Non attribué	1 / 256
0 0 0 0 0 0 1	Adresses OSI NSAP	1 / 128
0 0 0 0 0 1 0	Adresses IPX (Novell Netware)	1 / 128
0 0 0 0 0 1 1	Non attribué	1 / 128
0 0 0 0 1	Non attribué	1 / 32
0 0 0 1	Non attribué	1 / 16
0 0 1	Non attribué	1 / 8
0 1 0	Attribuées à un ISP	1 / 8
0 1 1	Non attribué	1 / 8
1 0 0	Attribuées à une région géographique	1 / 8
…	. . .	…
1 1 1 1 1 1 1 0 1 0	Usage local *	1 / 1024
1 1 1 1 1 1 1 0 1 1	Usage local *	1 / 1024
1 1 1 1 1 1 1 1	Adresses multicast	1 / 256

* Ces paquets ne sont pas propagés à l'extérieur d'un SA

Adresses IPv6
notation

Notation complète (les 16 octets)

8 0 0 0 : 0 0 0 0 : 0 0 0 0 : 0 0 0 0 : 0 1 2 3 : 4 5 6 7 : 8 9 A B : C D E F

Notation abrégée (omission et compression de zéros au début d'un bloc)

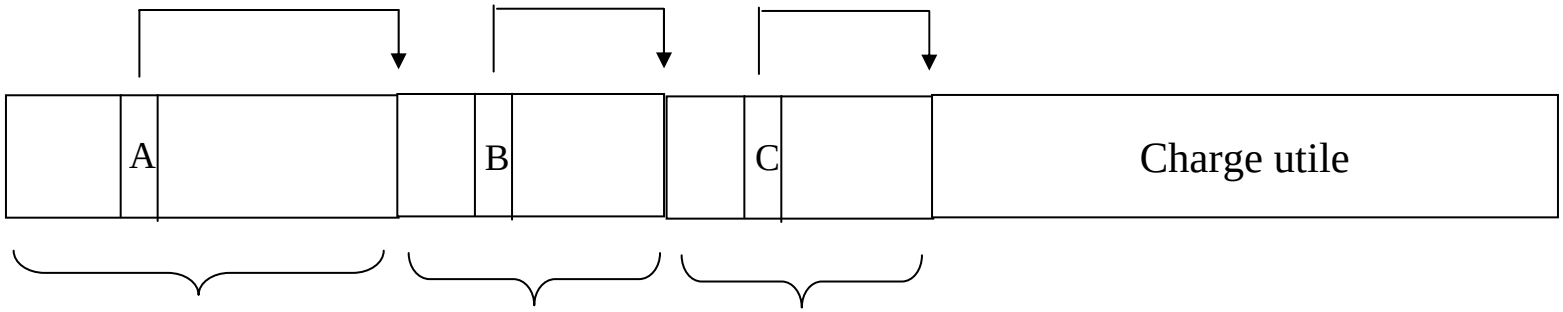
8 0 0 0 : : 1 2 3 : 4 5 6 7 : 8 9 A B : C D E F

Une adresse IPv4

: : 132 . 210 . 74 . 64

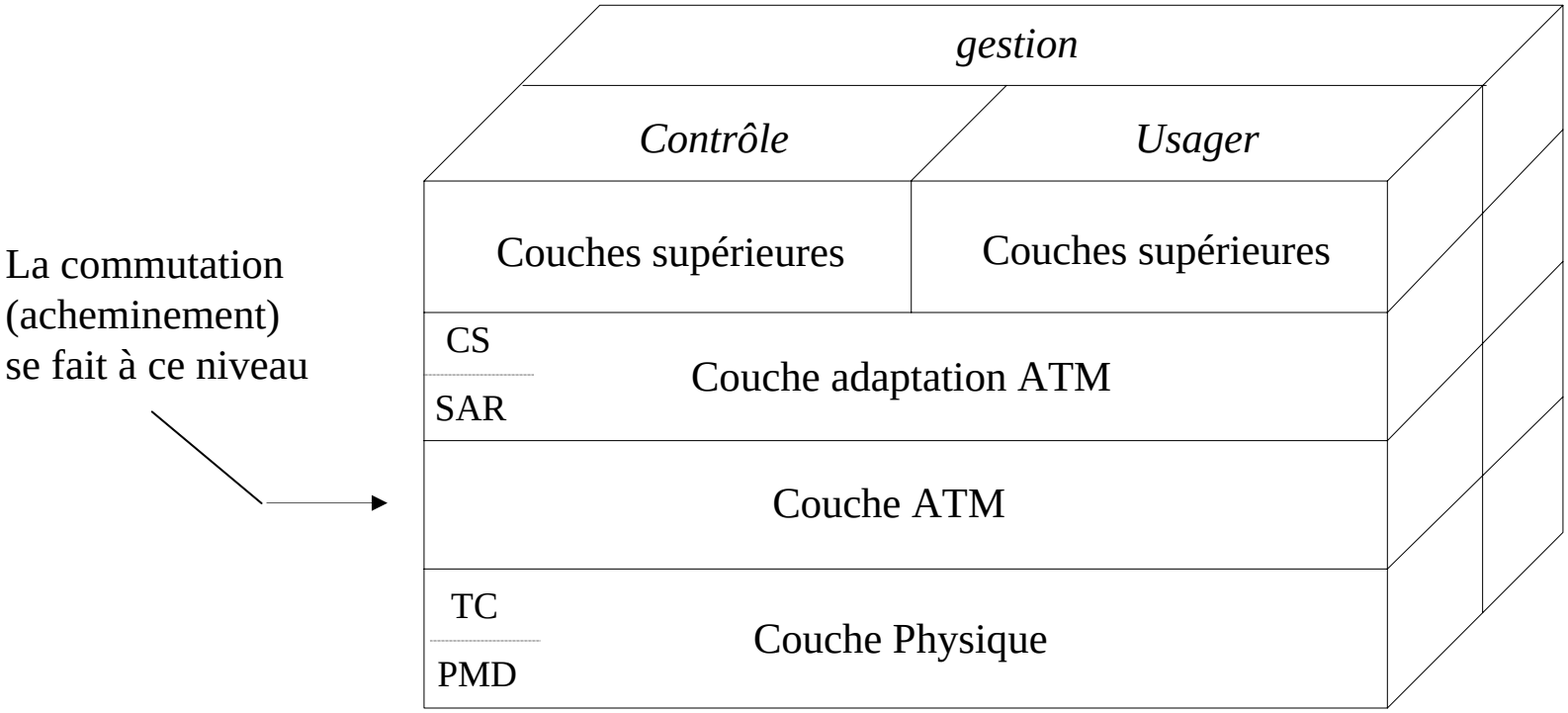
signifie qu'il n'y a que des zéros devant

IPv6
extension de l'entête



A : PTR ENTETE pour l'entête principale (pointe à l'extension 1)
B : PTR ENTETE pour l'extension 1 (pointe à l'extension 2)
C : PTR ENTETE pour l'extension 2 (décrit le protocole -- TCP, UDP, …)

La couche réseau sur ATM



Note: Il est difficile de comparer chaque couche du modèle ATM avec les couches du modèle OSI ou TCP/IP. Ce sont des modèles très différents...

ATM
un service orienté connexion

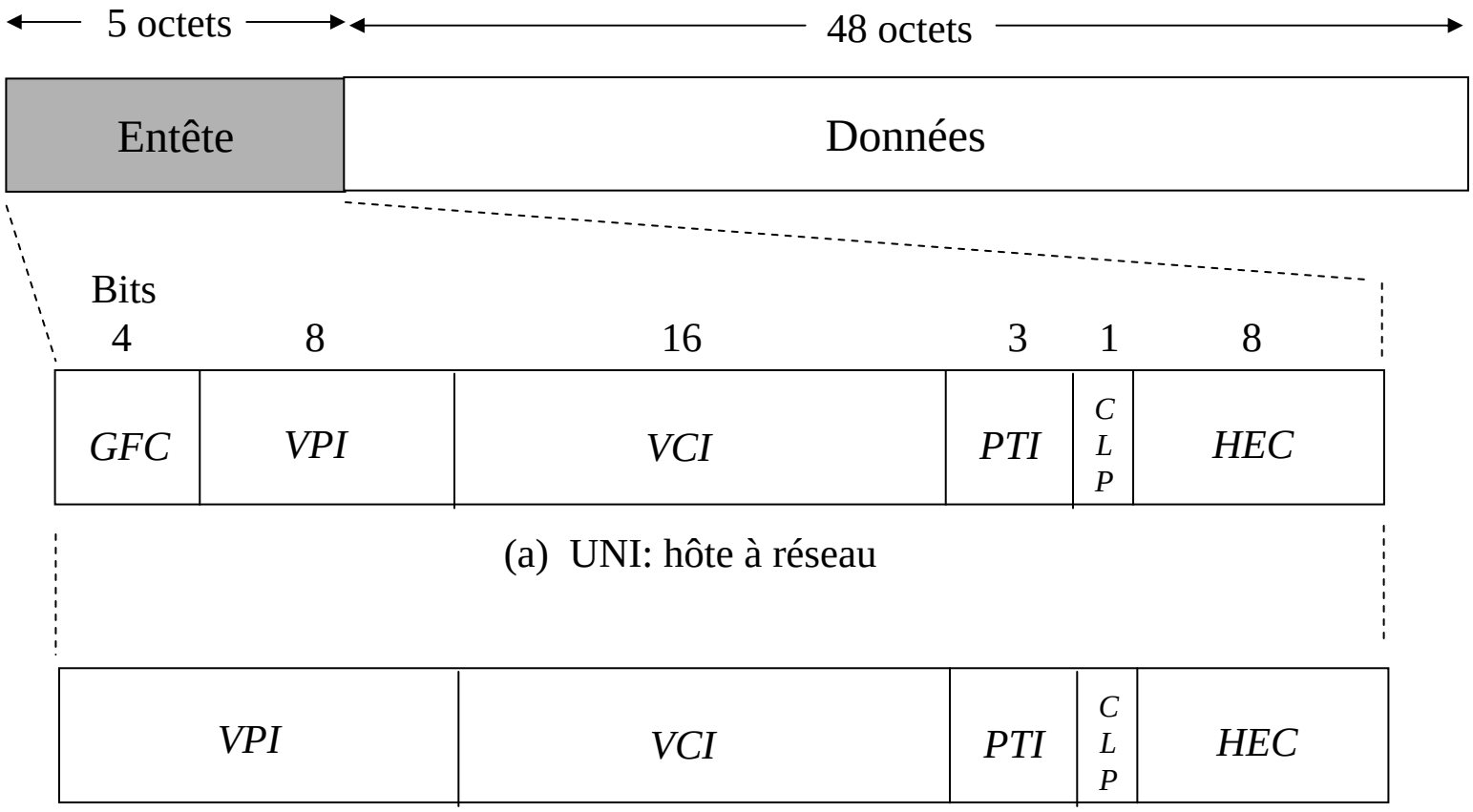
- Acheminement par *circuits virtuels*
  → il faut d'abord établir la connexion (réserver un CV)

- L'ordre des paquets (cellules) sur un circuit est préservé

- Protocole orienté connexion *sans acquiescements* …
  → le medium physique (fibre optique) est très fiable

- La récupération des erreurs est laissée aux couches supérieures

Cellule ATM


5 octets
48 octets
Entête
Données
Bits
4
8
16
3
1
8
GFC
VPI
VCI
PTI
CLP
HEC
(a) UNI: hôte à réseau
VPI
VCI
PTI
CLP
HEC
(b) NNI: réseau à réseau

Les champs de l'entête ATM

- **GFC** : (**4 bits**)
  - → seulement pour les connexions entre hôte et réseau
  - → aucune signification de bout-en-bout
  - → n'est pas délivré au récepteur
    (le champs est écrasé au premier routeur rencontré…)
  - → "… a bug in the [ATM] standard." A.S. Tanenbaum

- **VPI** : (**8 ou 12 bits**)
  - → Indicateur de chemin Virtuel ("Virtual Path")

- **VCI** : (**16 bits**)
  - → Indicateur de Circuit Virtuel ("Virtual Circuit")
  - → Un VP contient plusieurs VC

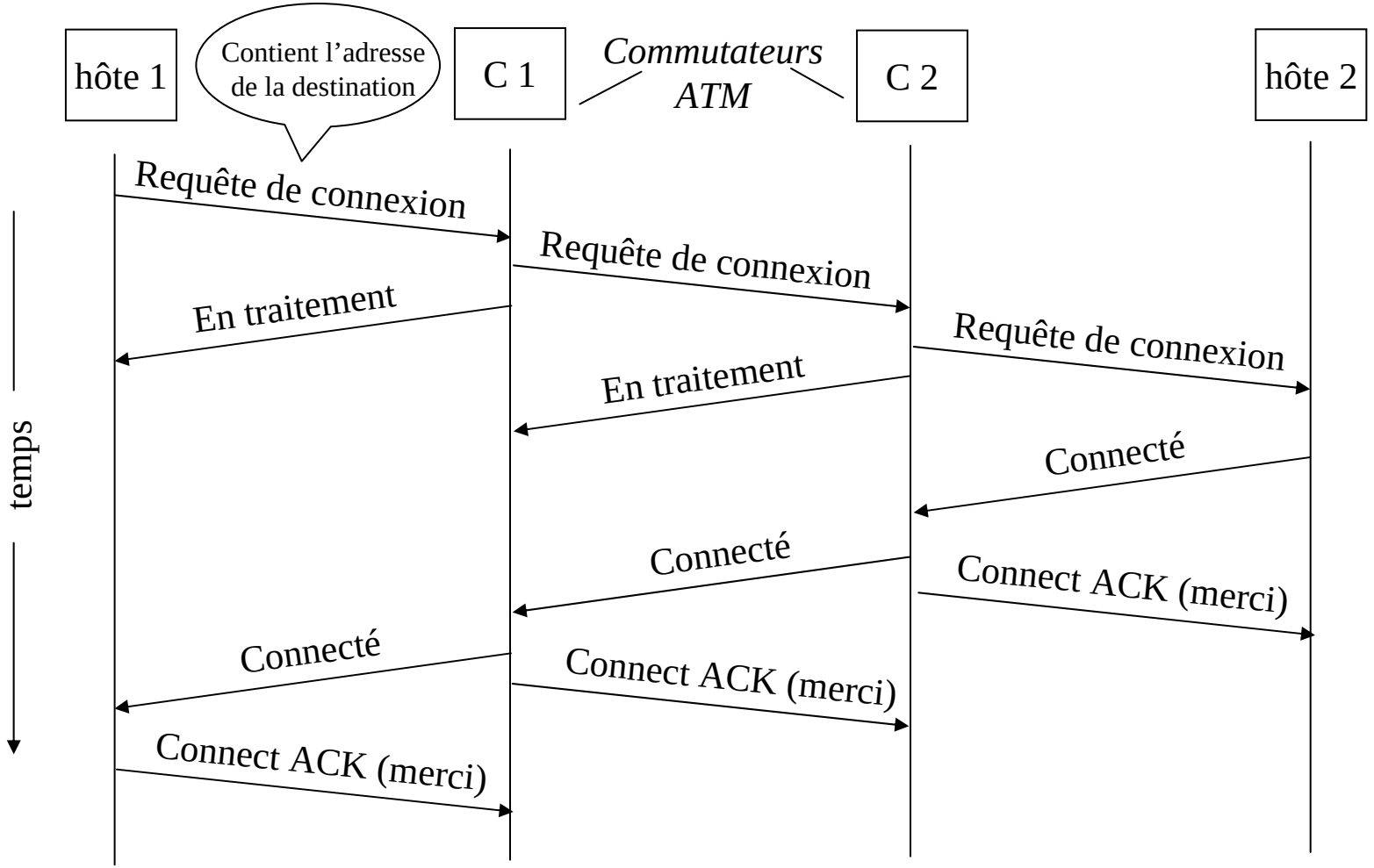
Champs de l'entête ATM (suite)

- **PTI** : (**3 bits**)
  - → Types de données de la charge utile
  - → Permet d'indiquer au récepteur le niveau de congestion

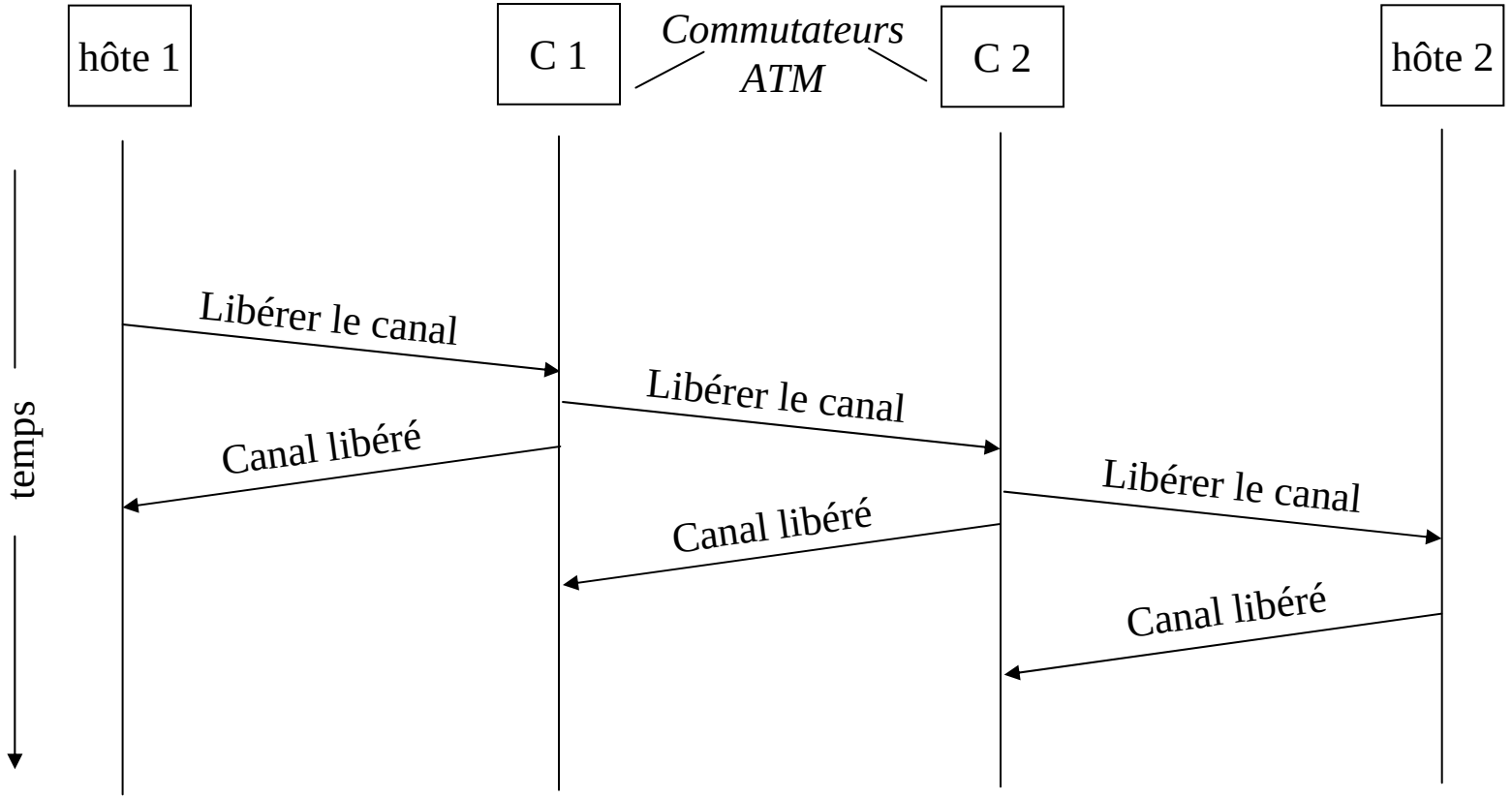
- **CLP** : (**1 bit**)
  - → Niveau de priorité ("Cell Loss Priority")
  - → Les commutateurs ATM laissent tomber une cellule avec CLP=1 avant une cellule avec CLP=0

- **HEC** : (**8 bit**)
  - → Bits de parité pour l'entête uniquement
  - → CRC de 8 bits, avec $g(x) = x^8 + x^2 + x + 1$
  - → peut corriger toutes les erreurs sur 1 bit
  - → peut détecter 90% de tous les autres patrons d'erreurs

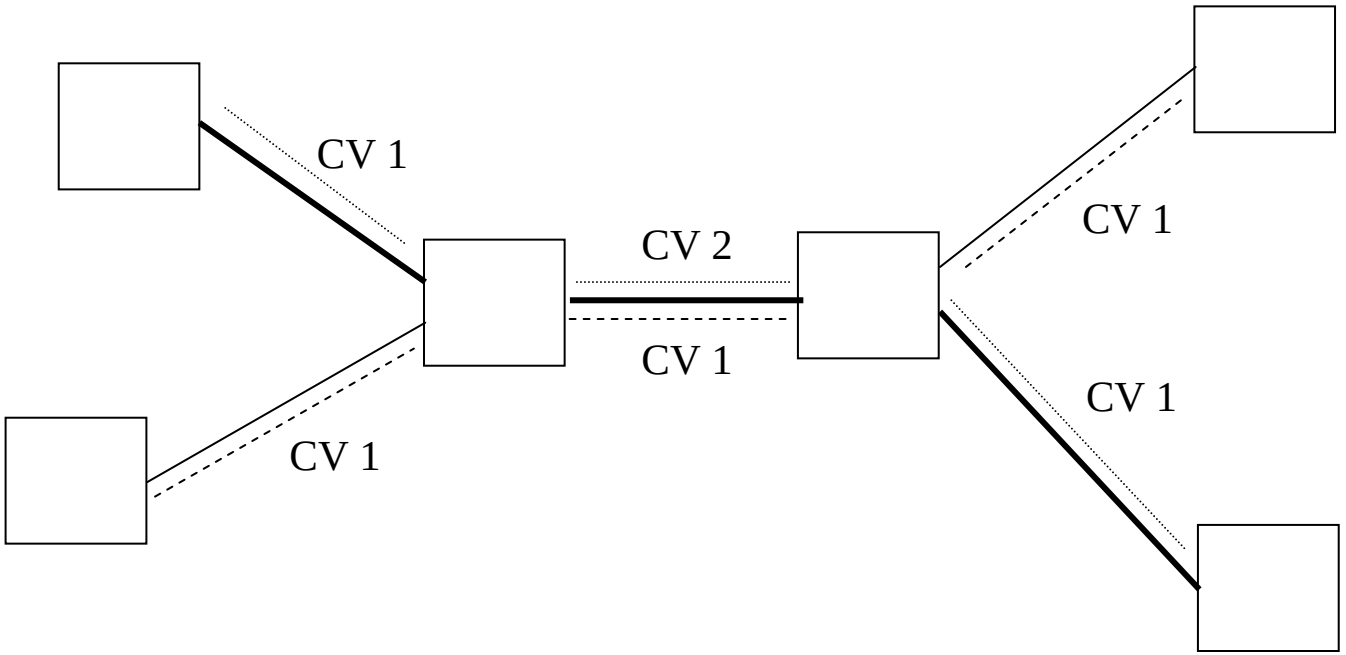
Etablissement de la connexion



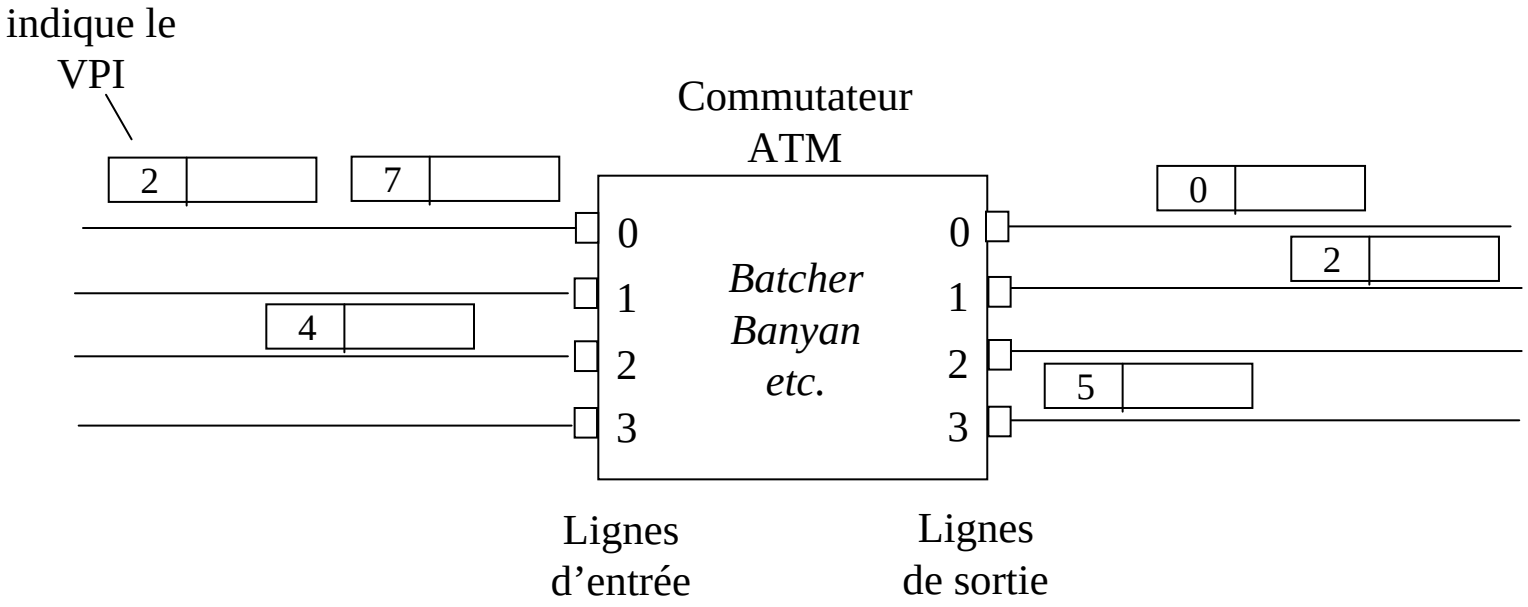
Libération de la connexion



Circuits virtuels



Commutation sur ATM



Ligne d'entrée	VPI	Ligne de sortie	Nouveau VPI
0	2	3	5
0	3	0	4
0	7	1	2
2	4	0	0

Contrôle de la congestion

- Indicateurs de la congestion:
  - → nombre moyen de paquets rejetés
  - → longueur moyenne des files d'attente
  - → nombre de retransmissions pour cause de "timeout"
  - → délai moyen par paquet
  - → distribution des délais (histogramme)
- Contrôle sans feedback
- Contrôle avec feedback
- Vitesse d'adaptation: optimisation *non-triviale*

- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

La couche transport

- Assure un service **fiable** de transmission de paquets
  → isole les couches supérieures (application) des diverses technologies et des imperfections du réseau

- Est à la couche réseau ce que la couche liaison de données est à la couche physique

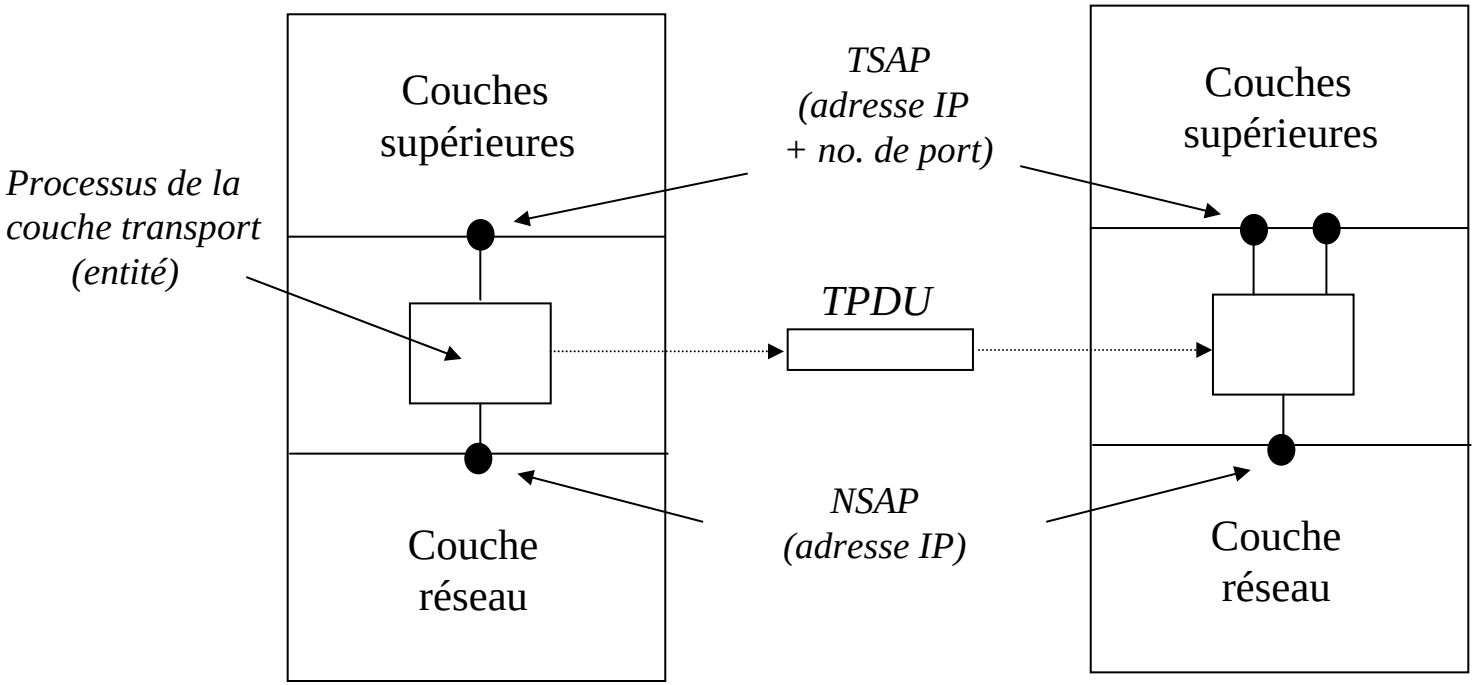
- Mécanismes de **numérotation** et d'**acquiescement** des paquets (avec possibilité de retransmission des paquets)

- Possibilité de service fiable ou sans guarantie

- Ajustement du débit en fonction de la capacité du récepteur

Préliminaires

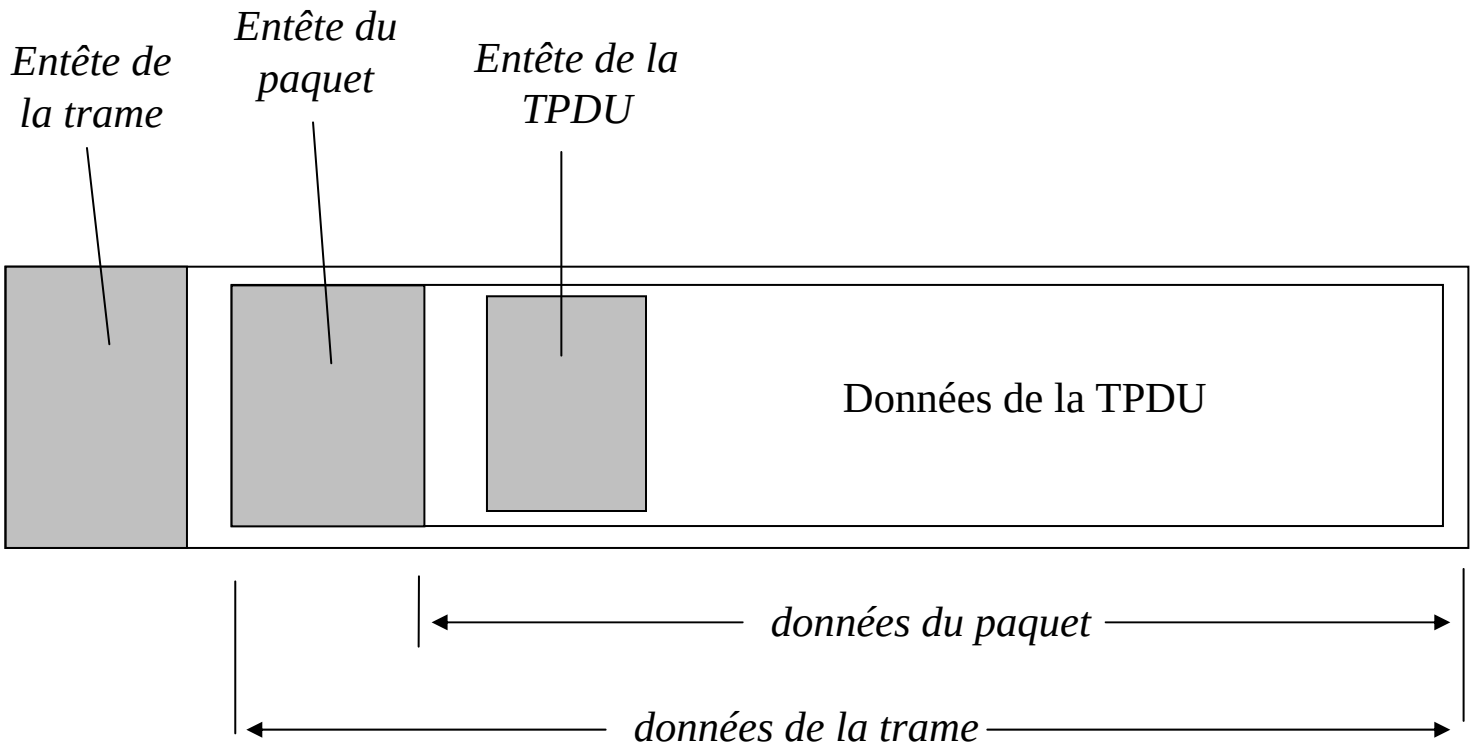
Communication *pair-à-pair*



Services de base
exemple plus simple que BSD

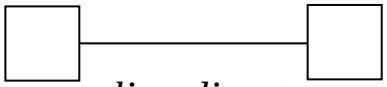
Primitive	Message (TPDU) envoyé	Signification
LISTEN	*aucun*	Le processus est bloqué jusqu'à ce qu'un autre processus se connecte
CONNECT	Requête de connexion	Le processus tente d'établir une connexion
SEND	Données	Transmission de données
RECEIVE	*aucun*	Le processus est bloqué jusqu'à ce qu'une TDPU de données arrive.
DISCONNECT	Requête de déconnexion	Le processus veut libérer la connexion

Encapsulation



Couche 4 *vs* couche 2

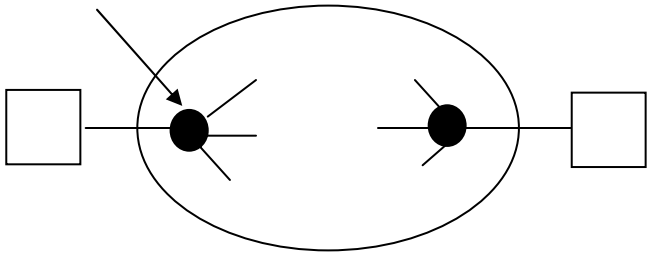
Couche 2
(Liaison de données)



lien direct

Couche 4
(Transport)

Routeur



réseau

Le lien physique n'a pas
de capacité de mémoire

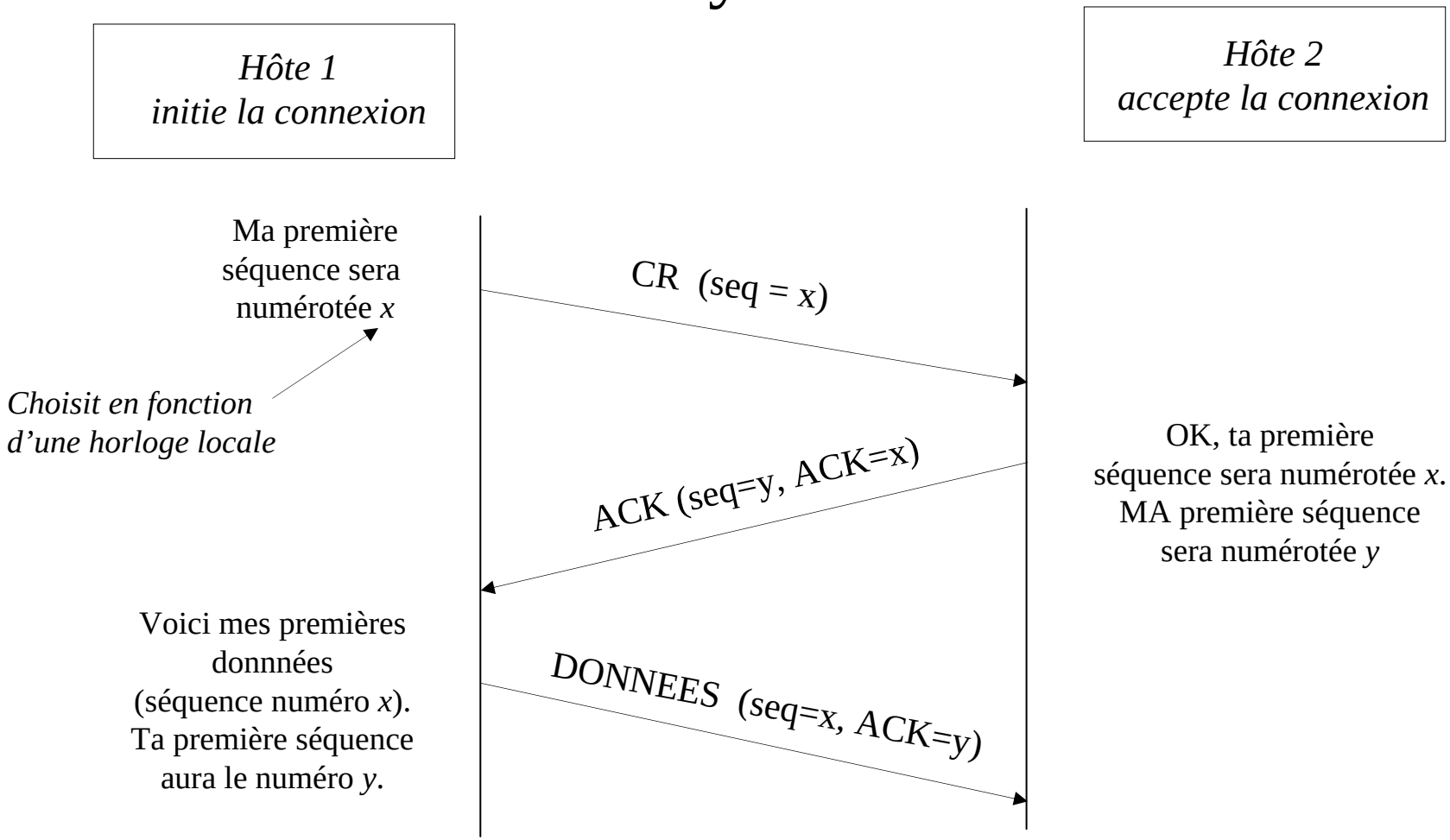
Un seul canal à gérer
→ une seule file d'attente
pour les retransmissions

Le réseau (ensemble des routeurs)
a une grande capacité de mémoire
(et de délais associés)

Plusieurs connexions simultanées possibles
→ plusieurs files d'attente
pour les retransmissions

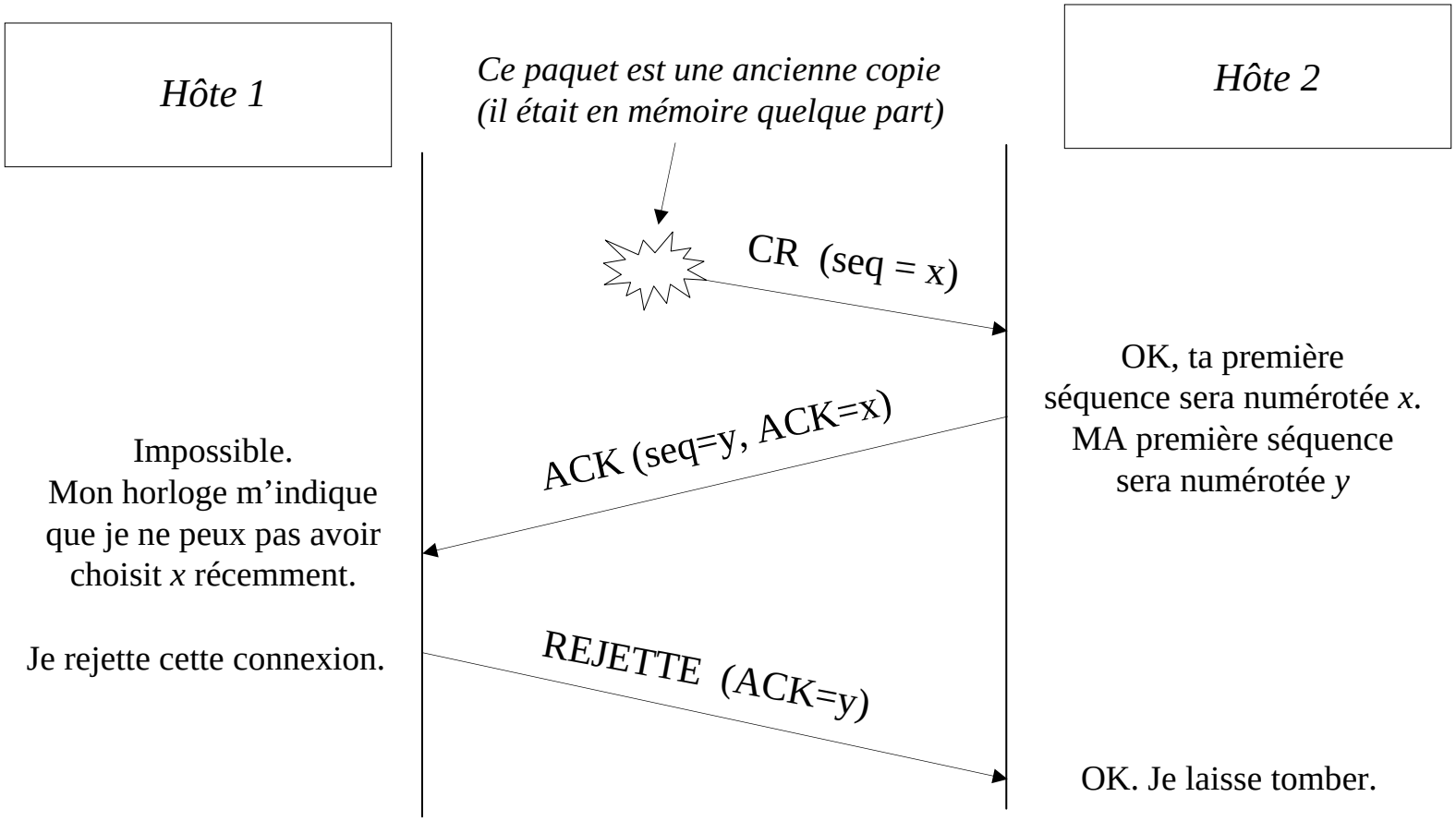
Etablissement de la connexion

“three-way handshake”



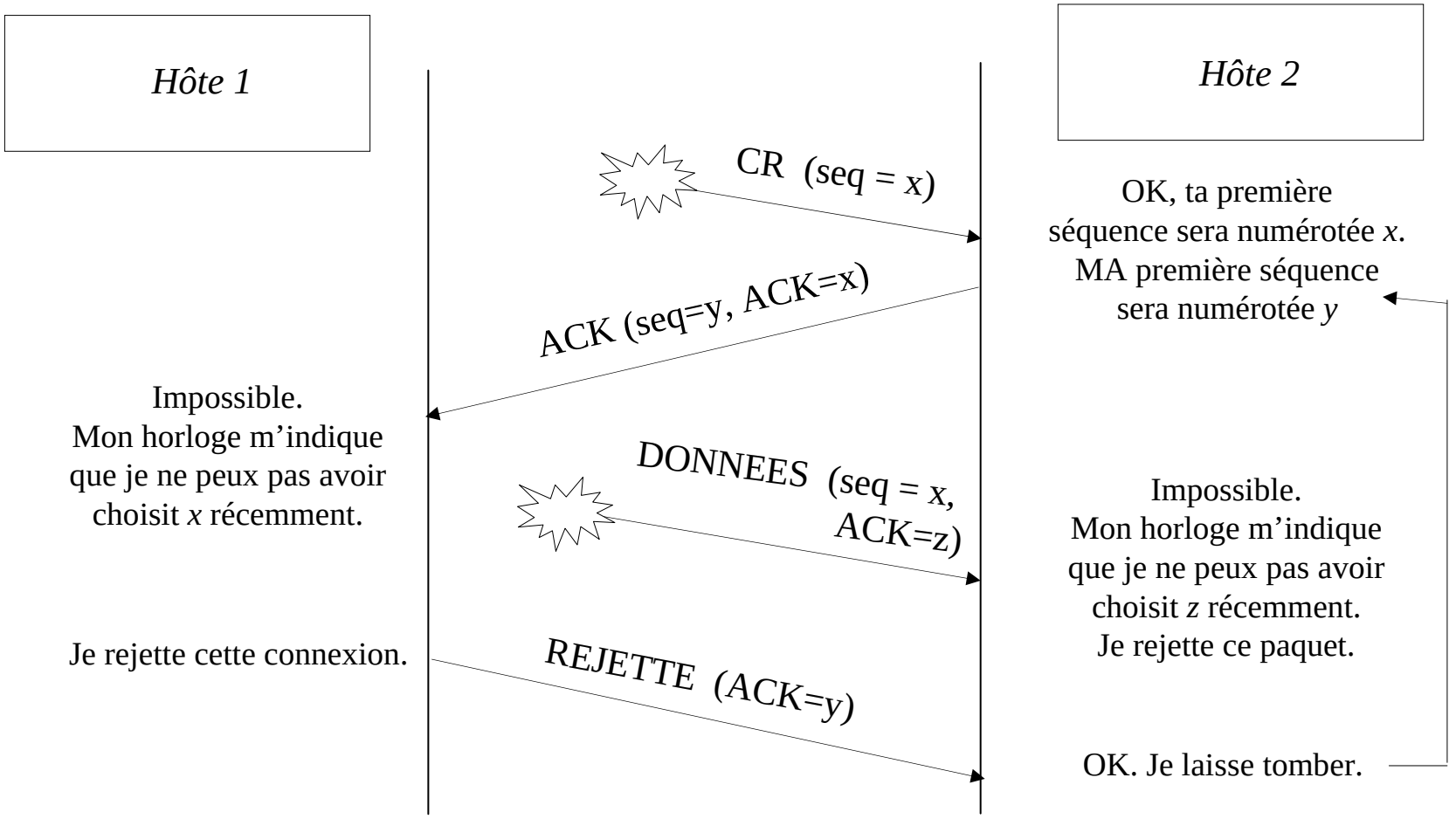
"Three-way handshake"

robustesse

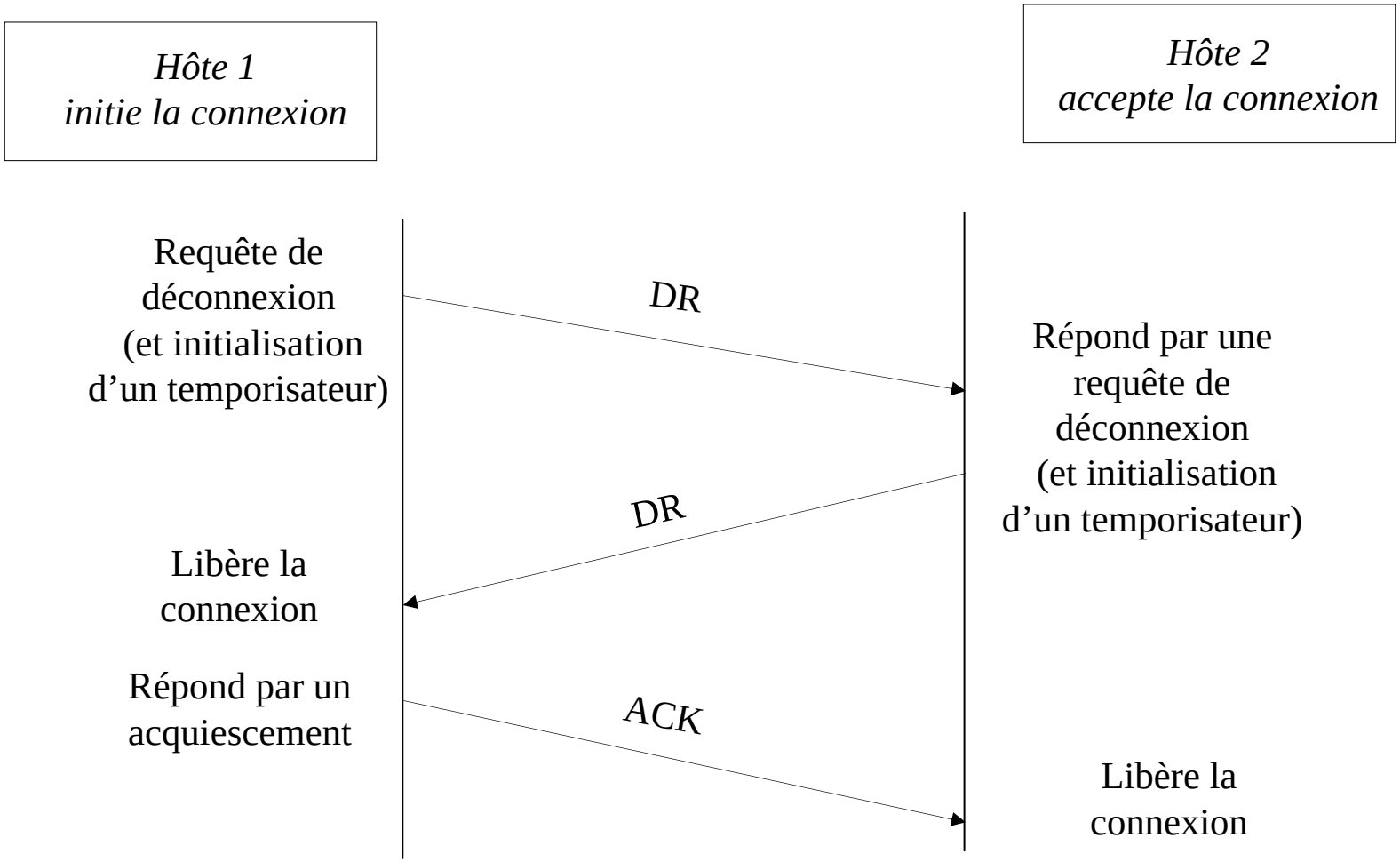


“Three-way handshake”

robustesse



Libération de la connexion



Contrôle du flux de données
selon les limites du receveur

- Protocoles à fenêtre glissante
  → similaire à la couche liaison de données

- Typiquement, l'émetteur *requiert* une taille mémoire au receveur
  → pour stocker les messages en attente d'être livrés *dans l'ordre* aux couches supérieures du receveur
  → le receveur alloue de façon dynamique à l'émetteur le nombre de blocs mémoire disponibles à ce moment

- La taille de la fenêtre glissante est gérée *par le receveur*
  → c'est donc le receveur qui maîtrise le flux de données

Contrôle du flux de données
selon les limites du réseau

- Un autre élément important à considérer:
  → la *capacité* du réseau

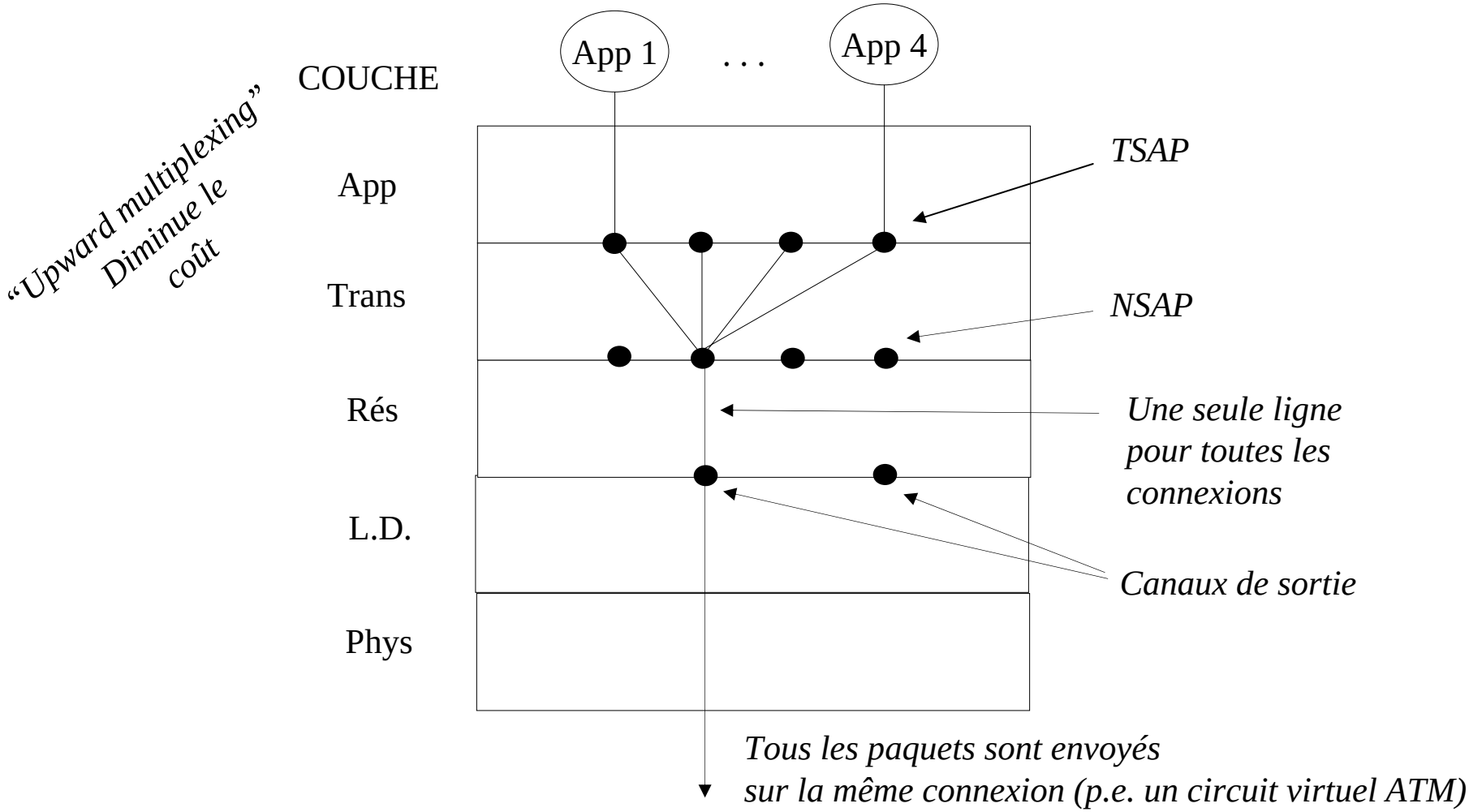
- Demande aussi un mécanisme de contrôle à l'*émetteur*

- Ajustement *dynamique* de la taille de la fenêtre à l'émetteur
  → Taille de la fenêtre = $c\ r$
  où c est la capacité du réseau en TPDU / sec
  et r est le temps aller-retour (TRANS - CPU - ACK)

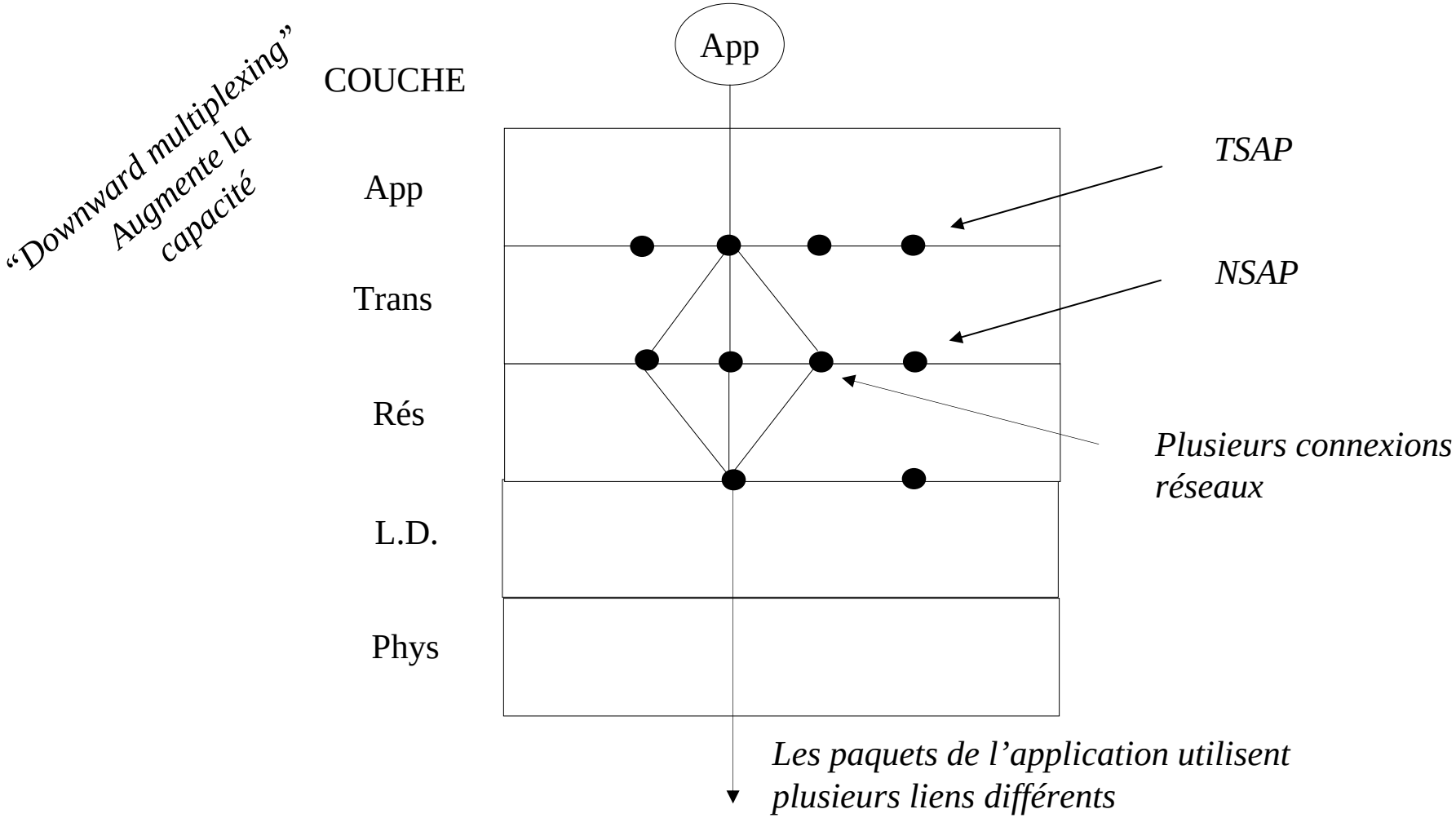
- La capacité est mesurée en comptant périodiquement
  le nombre de TPDUs acquiescées par unité de temps
  (l'émetteur transmet en continu lorsqu'il mesure…)

- La couche transport sur Internet (TCP) utilise ce mécanisme

Multiplexage des protocoles - 1



Multiplexage des protocoles - 2



Couche Transport sur Internet

TCP et UDP

- TCP : “Transmission Control Protocol”
  - → service orienté connexion
    - gère les ACK et temporisateurs
    - remet les paquets *dans l’ordre* aux couches supérieures
  - → s’adapte aux conditions du réseau
  - → fiable en cas de pannes diverses

- UDP : “User datagram Protocol”
  - → service de datagrammes non orienté connexion
  - → essentiellement, même service que IP

TCP

RFC 793, 1122, 1323

- L'émetteur et le receveur doivent créer un *socket*
  $\rightarrow$ socket = *adresse* IP + numéro de *port* + *service*

- Une *connexion* est définie par l'association d'un socket sur une machine et un socket sur une autre machine

- Numéros de port réservés: 0-255

- Numéros de ports bien connus: voir RFC 1700
  21 : FTP
  23 : Telnet
  25 : SMTP

TCP : un “byte stream”

- Les limites de messages ne sont pas préservées

→ TCP peut segmenter ou regrouper
→ exemple :
- l’émetteur transmet 4 blocs de 512 octets
- le receveur peut recevoir
4 blocs de 512 octets ou
2 blocs de 1024 octets ou
1 bloc de 2048 octets ou
…
→ analogie: lecture d’un fichier
à la lecture, on ne peut pas dire la dimension des blocs utilisés lors de *l’écriture*

Le segment TCP

32 bits

PORT SOURCE								PORT DESTINATION
NUMERO DE SEQUENCE								
NUMERO D'ACQUIESCEMENT								
HLEN		URG	ACK	PSH	RST	SYN	FIN	TAILLE DE LA FENETRE
CHECKSUM								POINTEUR (URG)
OPTIONS (0 ou nombre entier de mots de 32 bits)								
DONNEES								

Les champs du segment TCP

- **PORT SOURCE** : (**16 bits**)
  - → porte logique associée à la connexion locale de la source
  - → peut être alloué explicitement (si > 255)
  - → Adresse IP + port = TSAP unique (48 bits au total)

- **PORT DESTINATION** : (**16 bits**)
  - → porte logique associée à la connexion locale de la destination

- **NUMERO DE SEQUENCE** : (**32 bits**)
  - → Indique la position du premier octet du champs de données du segment TCP dans le message (par exemple, un fichier)

- **NUMERO D'ACQUIESCEMENT** : (**32 bits**)
  - → Indique le numéro du prochain octet attendu au récepteur (et non le numéro du dernier octet bien reçu)

Les champs du segment TCP (suite)

- **HLEN** : (**4 bits**)
  - → longueur de l'entête TCP (en mots de 32 bits)
  - → requis puisque le champs OPTIONS est à longueur variable

- **---** : (**6 bits**)
  - → 6 bits non-utilisés (réservés pour de futures modifications qui ne se sont jamais avérées nécessaires)

- **URG** : (**1 bit**)
  - → utilisé conjointement avec le champs POINTEUR (URG)
  - → ce bit est mis à 1 pour signifier que des données doivent être transmises immédiatement
    (rappelons que TCP ne préserve pas nécessairement les divisions entre les messages)

Les champs du segment TCP (suite)

- **ACK** : (**1 bit**)
  → mis à 1 si le champs NUMERO D'ACQUIESCEMENT est valide

- **PSH** : (**1 bit**)
  → indique une information urgente au receveur
  → le receveur doit passer les données reçues immédiatement à la couche application

- **RST** : (**1 bit**)
  → utilisé après un problème majeur (ex.: crash d'une machine)
  → utilisé aussi pour indiquer le refus d'un segment invalide ou d'une tentative de connexion.
  → en général, ce bit indique un problème majeur…

Les champs du segment TCP (suite)

- **SYN** : (**1 bit**)
  - → utilisé lors de la connexion
  - → pour la requête (*connect*), on a SYN=1 et ACK=0
  - → si on accepte (*accept*) on a SYN=1 et ACK=1

- **FIN** : (**1 bit**)
  - → utilisé pour libérer la connexion
  - → signifie que l'émetteur n'a plus d'info à transmettre
  - → puisque l'ordre des paquets n'est pas garantit sur le réseau, le receveur doit examiner les numéros de segments (NUMERO DE SEQUENCE) pour s'assurer d'avoir reçu toute l'information de l'émetteur

Les champs du segment TCP (suite)

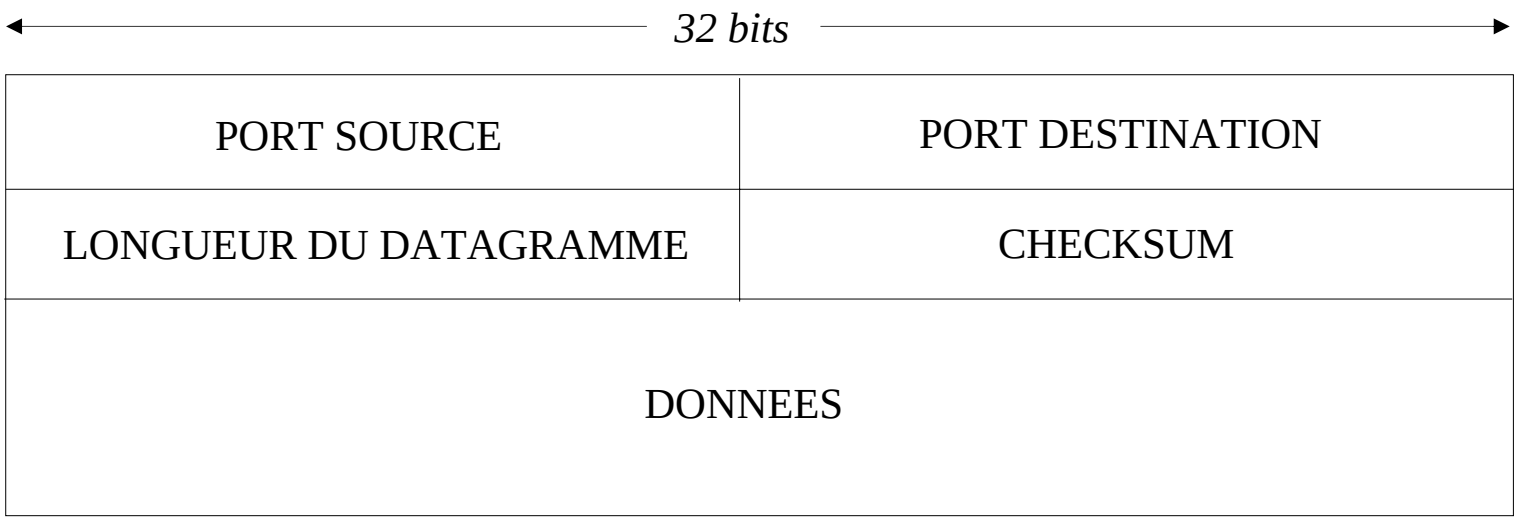
- **TAILLE DE LA FENETRE** : (**16 bits**)
  - → indique le nombre d'octets pouvant être transmis à partir du dernier octet acquiescé
  - → contrôlé par le receveur (contrôle de congestion)
  - → similaire à la fenêtre *W* dans les protocoles de liaison de données, sauf qu'ici on compte les octets directement
  - → un valeur de 0 est permise; elle indique que le receveur demande un repos
  - → en transmettant un aquiescement avec le même numéro et une valeur non-nulle de TAILLE DE LA FENETRE, le receveur peut permettre à l'émetteur de transmettre à nouveau

- **CHECKSUM** : (**16 bits**)
  - → bits de parité (sur l'entête et les données)
  - → somme complément-à-1 par mots de 16 bits

UDP
User Datagram Protocol

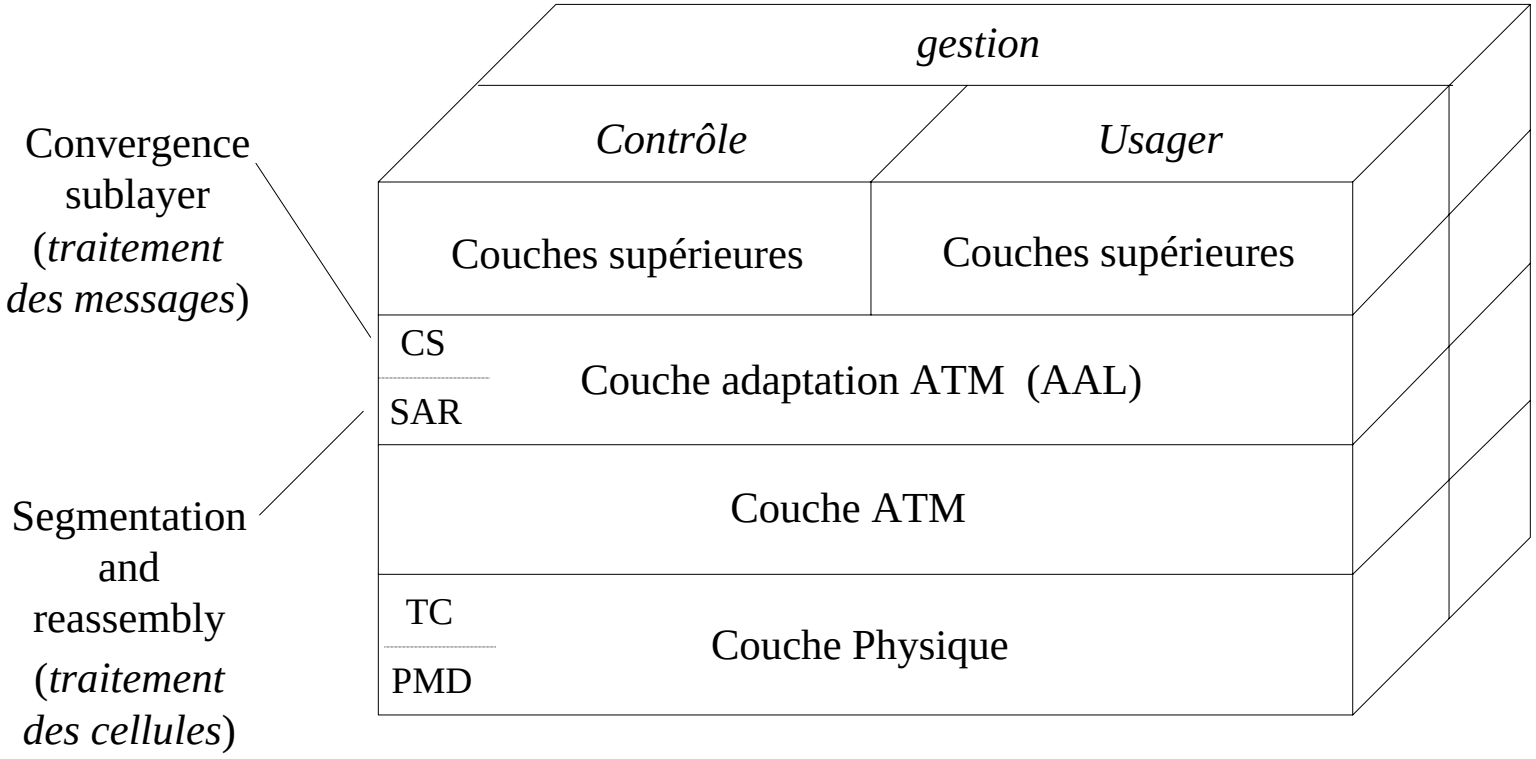
- Protocole non orienté connexion
- Aucune connexion requise avant de transmettre des données
- Décrit dans RFC 768
- Usages:
  - transmission audio/video temps réel
  - sessions interactives courtes (1 requête / 1 réponse)
  - etc.

datagramme UDP



Couche Transport sur ATM

Couche “Transport” ATM = AAL



Rôles de la couche AAL

- Isoler l'usager du fonctionnement de la couche ATM
  (i.e. découpage en cellules et réassemblage)

- Offrir différents types de services:

  - temps-réel ou non (TR/NTR)
  - débit fixe ou variable (DF/DV)
  - orienté connexion ou non (…) (OC/NOC)

	temps-réel	débit	orienté-connexion
Classe A :	OUI	fixe	OUI
Classe B :	OUI	variable	OUI
Classe C :	NON	variable	OUI
Classe D :	NON	variable	NON

AAL1

- Protocole pour le service de classe A
- Lissage du traffic (l'application voit un débit constant)
- N'utilise PAS de protocole à fenêtre glissante (aucun acquiescement)
- Les cellules manquantes sont tout de même rapportées à l'application
  → indique aussi les cellules "mal acheminées"
  (numéro de séquence dans chaque cellule)
- Détection d'erreurs uniquement sur l'entête et le num. de séquence
- Applications:
  - transmission temps-réel audio/video non compressé

AAL 2

- Protocole pour le service de classe B (débit variable)
- N'utilise PAS de protocole à fenêtre glissante (aucun acquiescement)
- Détection d'erreurs SUR TOUTE LA CELLULE
- Le standard n'inclut pas la taille des champs (no. de séq, type, etc.)
  → rend ce standard *inutilisable* …
- Applications possibles:
  - communications audio/video avec compression à débit variable

AAL 3/4

- Deux modes: service fiable, et service sans garantie

- Multiplexage
  - → exemple: plusieurs sessions sur le même circuit virtuel
  - → les transporteurs facturent *à la connexion (i.e. CV)*

- Quatre octets de contrôle pour chaque cellule
  - → en plus des 5 octets de l'entête …

- Permet de traiter des "messages" pouvant aller jusqu'à 65535 octets
  - → la sous-couche de convergence (CS) découpe le message en cellules, et ajoute 4 octets de contrôle à chaque cellule

AAL 5

Interface de choix
pour TCP/IP sur ATM

- La réaction (tardive) de l'industrie informatique

- Plusieurs types de services
  - service fiable, avec contrôle du flot de données
  - service sans garantie
    les cellules avec erreurs sont
    1) jetées
    ou 2) rapportées à l'application (avec indication de défaut)

- Comme pour AAL 3/4, supporte les modes
  - message (maximum = un datagramme de 65535 octets)
  - "stream" (ne préserve pas les limites de messages, comme TCP)

- Usage d'un CRC plus long (4 octets), mais sur TOUT le message

- Aucune information supplémentaire n'est ajoutée aux cellules

- Introduction
- Couche Physique
- Couche Liaison de données
- Sous couche MAC (les réseaux locaux)
- Couche Réseau
- Couche Transport
- Couche Application

La couche application

- Sécurité (encryptage)
- Résolution de noms: DNS
- Traductions de haut niveau
  → exemple: ASN.1 (Structure abstraite de données)
- Applications multiples: courrier électronique, recherche de fichiers à distance, le WWW, etc.

Encryptage

- Pour assurer *confidentialité* et *sécurité*
- Pour *authentifier* l'identité d'un correspondant
- Deux approches principales:

1) encryptage à clé privée (ex.: DES)
   - même clé pour encoder et décoder
2) encryptage à clé publique (ex.: RSA)
   - clés différentes pour encoder et décoder

Algorithmes à clé privée

- Basés sur
  - **substitutions**

    hello $\rightarrow$ *rjccb*

    (i.e. h $\rightarrow$ r, e $\rightarrow$ j, etc.)

  - **transpositions**

    la_vie_privee_est_sacree

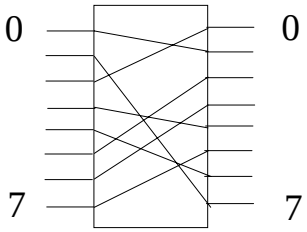
    l a _ v i e
    p r i v e e
    _ e s t _ s
    a c r e e

    $\rightarrow$ *lp_aarec_istvvteie_eees*

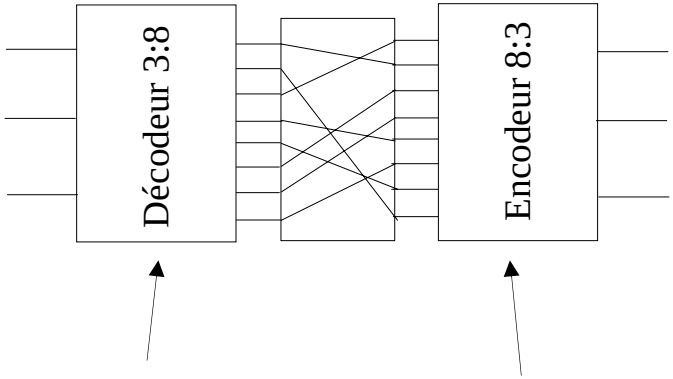
    (On lit les colonnes au lieu des lignes)

Algorithmes à clé privée
fonctions de base

Transposition
Permutation
("P-box")



Substitution
("S-box")



Choisit une des 8 lignes de sortie, selon le code binaire à l'entrée

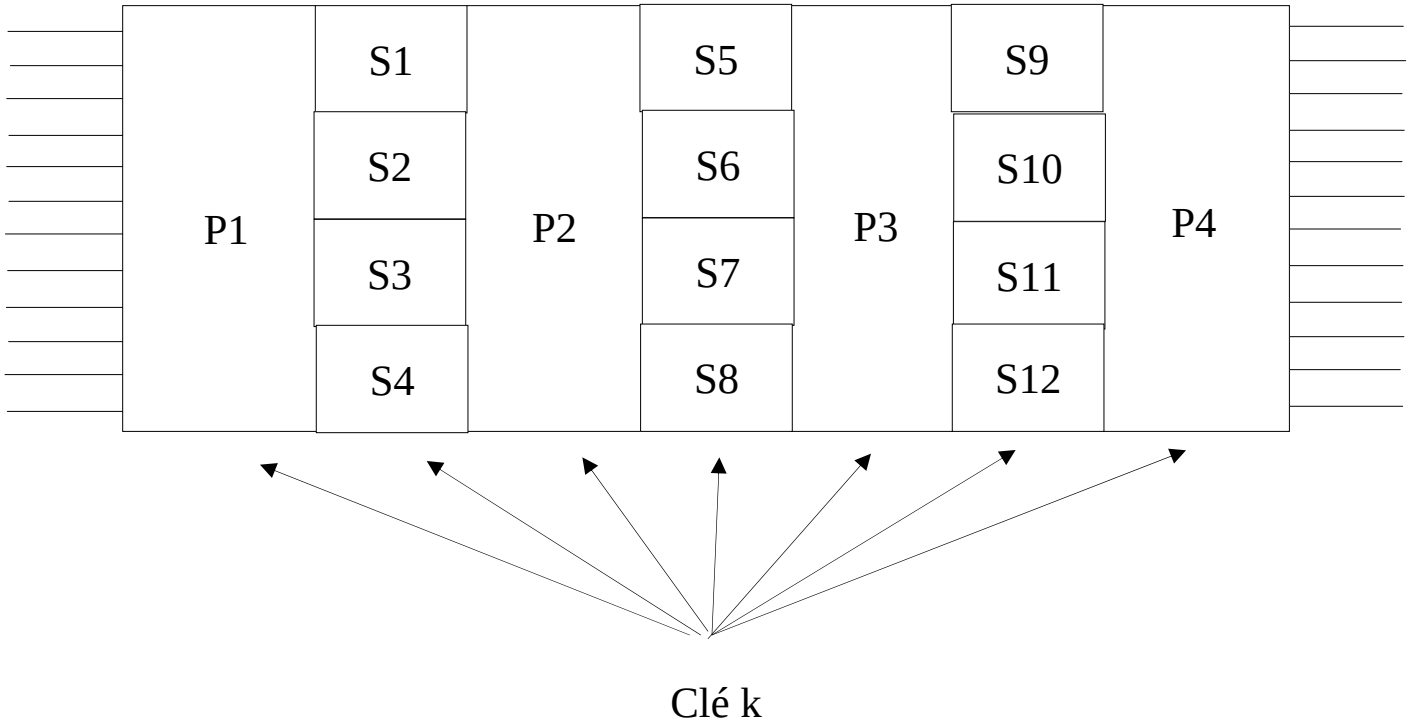
Le code binaire en sortie est le numéro de la ligne d'entrée "allumée"

Avec cette boîte, le code 010 devient 000

Algorithmes à clé privée
éléments en cascade

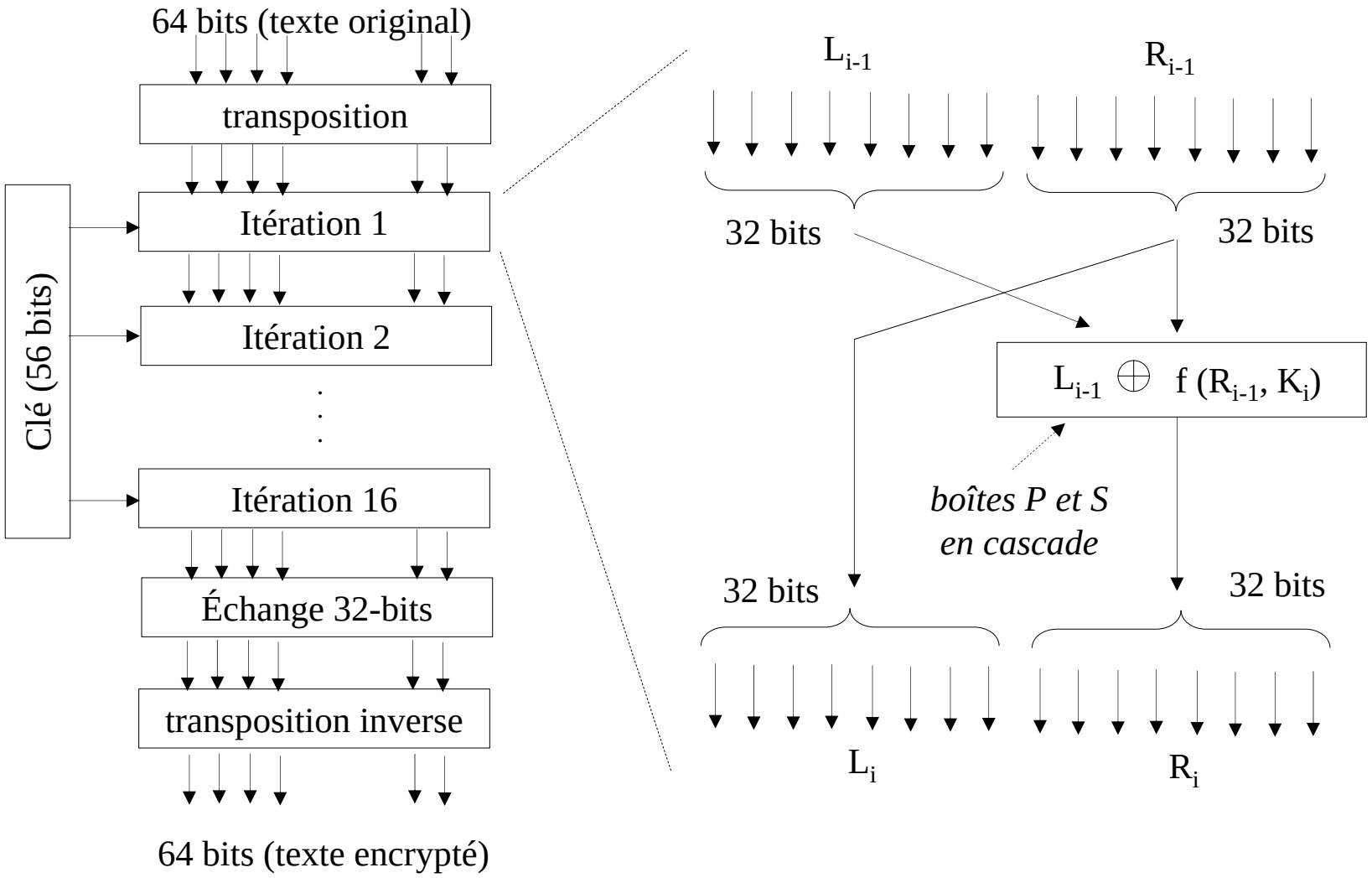
12 bits en entrée
(fichier original)

12 bits en sortie
(fichier encrypté)



DES

Data Encryption Standard



DES plus robuste

- Mécanisme de rétroaction
- Chaque mot de 64 bits est XOR avec le mot *encrypté* précédent

  Ainsi, le code pour un mot de 64 bits dépent du *contexte*

- Ce code est aussi inversible

Algorithmes à clé publique

- Clés différentes pour encoder et décoder
- Résout le problème de la distribution de clé
  → la clé d'encryptage peut être connue de tous…
- La clé (privée) de désencryptage ne peut être dérivée de la clé d'encryptage
- Le plus connu : RSA
  → basé sur la factorisation d'un entier (très grand) en ses facteurs permiers...

Algorithmes à clé publique

exigences de base

1) E = fonction d'encryptage
   D = fonction de désencryptage

2) $D(E(P)) = P$ où P est le message original

3) Il doit être TRÈS difficile de déduire D si on a E (clé publique)

4) On ne doit pas pouvoir briser le code en ayant un fichier suffisamment long de données encryptées

   Puisqu'on a la clé publique E, on peut générer "à volontée" des données encryptées et "expérienter"

RSA (Rivest, Shamir, Adleman: MIT)

un algorithme à clé publique

Bases mathématiques:

- Soient p et q deux nombres premiers (typiquement > 10^{100})
- Calculer $n = p \times q$ et $z = (p-1) \times (q-1)$
- Choisir un nombre d relativement premier p/r à z
  i.e. d est premier et n'est pas facteur de z
- Trouver e tel que $e \times d = 1 \ mod \ z$

- La clé d'encryptage est le couple (e, n)
- La clé de désencryptage est le couple (d, n)

RSA: un exemple

On fabrique les clés:

Prenons $p = 3$ et $q = 11$.
On a $n = 3 \times 11 = 33$
et $z = 2 \times 10 = 20$

Un choix pour d est $d = 7$
puisque 7 est premier et n'est pas facteur de z

Donc, e est la solution à $e \times d = 1 \ mod \ z$
i.e. $7e = 1 \ mod \ 20$

Et on trouve $e = 3$ (puisque $3 \times 7 = 21 = 1 \ mod \ 20$)

RSA: un exemple (suite)

La clé d'encryptage est le couple $(e, n) = (3, 33)$
La clé de désencryptage est le couple $(d, n) = (7, 33)$

Soit alors $P = 19$ le message à encrypter $(0 \leq P \leq n\text{-}1)$

Encryptage

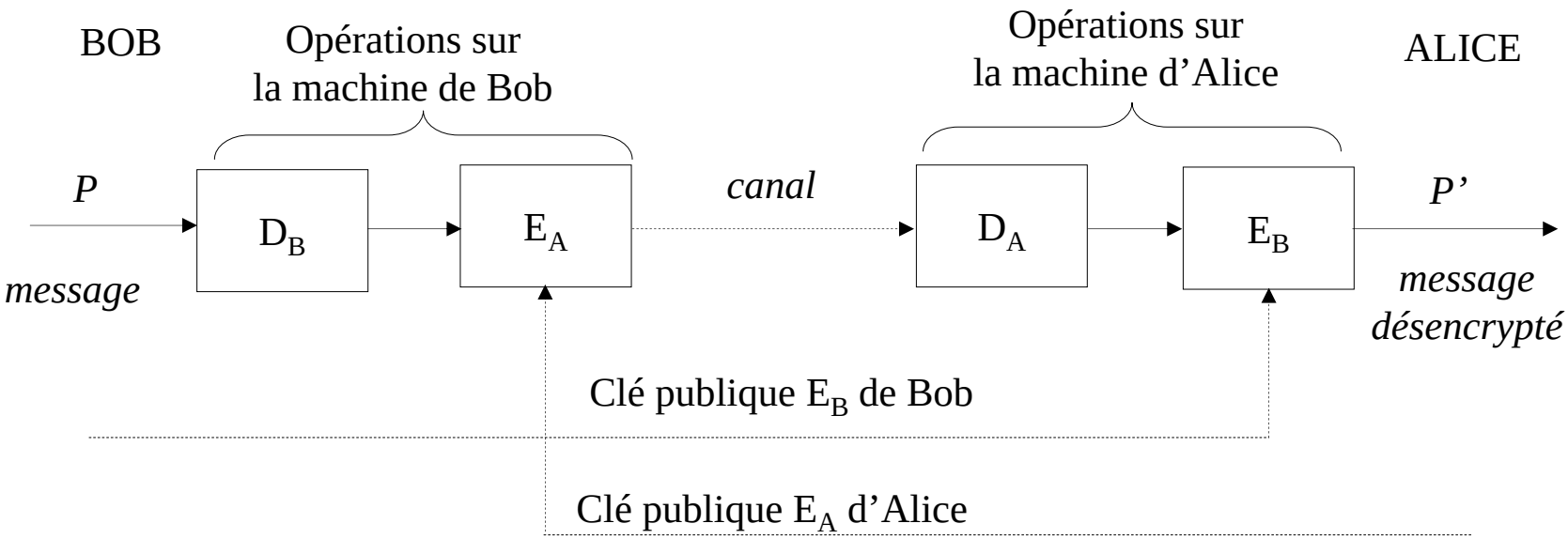
$$C = P^e \bmod n = 19^3 \bmod 33 = 6859 \bmod 33 = 28$$

Désencryptage

$$P = C^d \bmod n = 28^7 \bmod 33 = 13492928512 \bmod 33 = 19$$

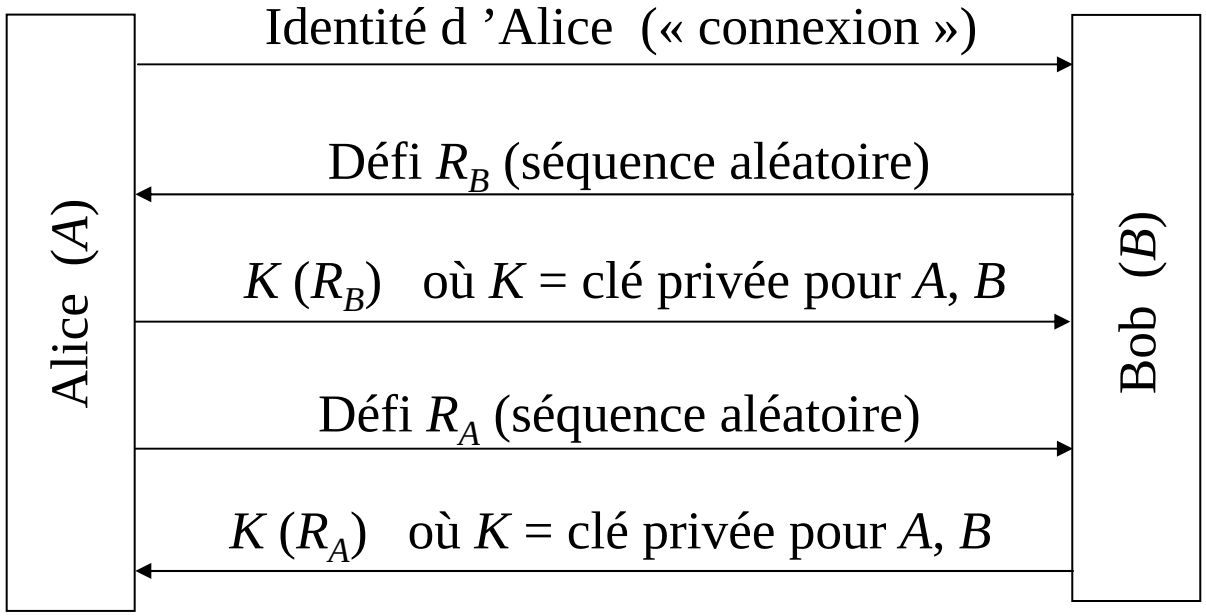
Note: On peut inverser le rôle des clés e et d (commutatif…)

Signature basée sur clé publique



$$P' = E_B\,(D_A\,(E_A\,(D_B\,(\,P\,)\,)\,)\,) = E_B\,(D_B\,(\,P\,)\,) = P$$

Authentification



Alice et Bob savent maintenant à qui ils s’adressent...

DNS
Domain Name System

- Les applications utilisent en général des noms symboliques comme *lefebvre@gel.usherb.ca*

- DNS est un service qui permet de faire le mapping entre un nom symbolique et (entre autres) une adresse IP

  → espace des noms très *hiérarchisé*
  → partie fixe (*usherb.ca*) allouée par une autorité centrale
  → base de données distribuée
  → recherche *récursive*à travers les serveurs de noms

Types de champs DNS

Une entrée dans la table d'un serveur DNS:

Nom du domaine *TTL* *Classe (IN)* *Type* *Valeur*

Type	Signification	contenu du champs *Valeur*
SOA	autorité (zone)	paramètres pour la zone du serveur
A	adresse IP	entier 32-bits (128 bits pour IPv6)
MX	trappe e-mail	priorité, nom du serveur de courrier élec.
NS	serveur de nom	nom d'un serveur pour ce domaine
CNAME	nom canonique	nom d'un domaine
PTR	pointeur	alias pour une adresse IP
HINFO	descr. d'un hôte	CPU et système d'opération en ASCII
TXT	texte	message ASCII

ASN.1
Abstract Syntax Notation 1

- Au coeur de SNMP (Simple Network Management Protocol)

- ASN définit :
  → un *format générique universel* pour les échanges de données
  → une *règle d'encodage* sur le canal

- Ce format générique permet les échanges entre des machines qui n 'utilisent pas la même représentation interne des données

- ASN est une norme ISO, complexe et relativement peu efficace
  → minimise le débit requis, au détriment de la complexité (CPU)

Applications majeures sur Internet

	Technologies reliées
• Courrier électronique	SMTP, MIME, PGP
• Usenet	NNTP (Network News Transfer Protocol)
• Le World Wide Web	HTML, Java, ...
• Multimedia	Compression audio/video

Courrier électronique

- RFC 822 pour les messages en ASCII
- Extension binaire/multimedia: MIME (RFC 1521)
- Deux entités distinctes:

    1) l'agent usager
        → écriture, lecture du courrier de la boîte postale
    2) l'agent de transfert
        → transfert des données source/destination (SMTP)
        → status des messages (délivrés, erreur, etc.)

MIME

Multipurpose Internet Mail Extensions

- Encodage de messages contenant de l'information autre que ASCII

- Règle d'encodage: base64
  - → 24 bits = 4 unités de 6 bits
  - → chaque unité est transmise en ASCII sur 7 bits

000000	=	'A'	110100	=	'0'
000001	=	'B'	110101	=	'1'
...			...		
011010	=	'a'	111110	=	'+'
011011	=	'b'	111111	=	'/'
...					

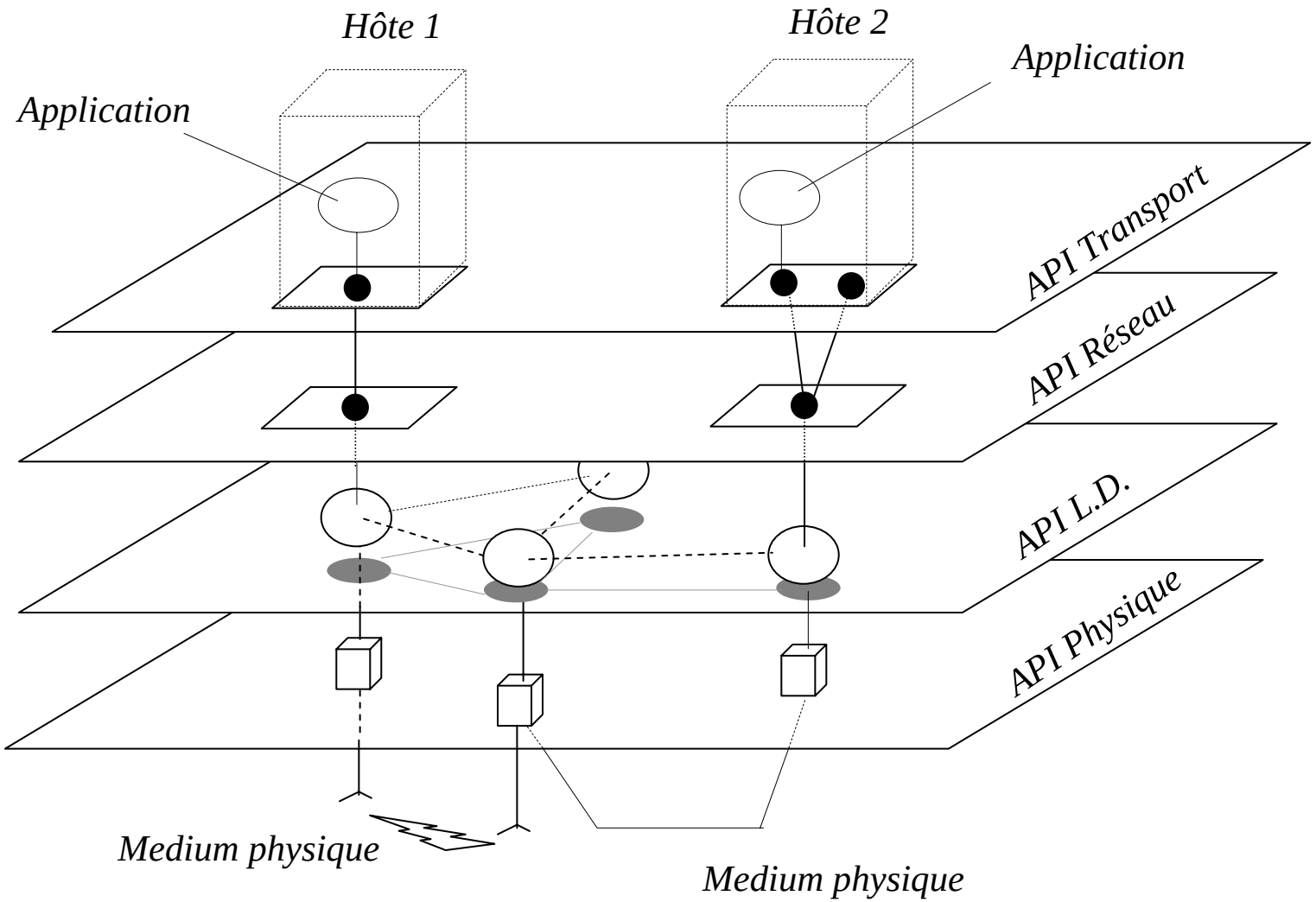
'= =' indique que le dernier groupe contenait 8 bits
'=' indique que le dernier groupe contenait 16 bits

SMTP
Simple Mail Transfer Protocol

- SMTP est un protocole simple orienté caractère (ASCII)
- Port standard pour le serveur = 25
- Le serveur SMTP
  - → reçoit et accepter les requêtes de transmission
  - → reçoit les messages du client (le courrier)
  - → copie les messages dans la boîte au lettre appropriée
  - → gère les erreur et les messages associés

Le WWW

- Système pour accéder à un ensemble de documents reliés entre eux sur l ’Internet
- Pour le commun des mortels, l’Internet C’EST le WWW
- Mis au point en 1989 par Tim Berners-Lee, au CERN
  → besoin de partager une base de documents distribués, variant continuellement dans le temps
- Première démonstration publique en 1991
- Basé sur le langage HTML
- Le consortium WWW: http://www.w3.org


Hôte 1
Hôte 2
Application
Application
API Transport
API Réseau
API L.D.
API Physique
Medium physique
Medium physique