M1 RES - Architecture des réseaux 4/10
Couche transport

Olivier Fourmaux

olivier.fourmaux@lip6.fr

Version 4.c, septembre 2004

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

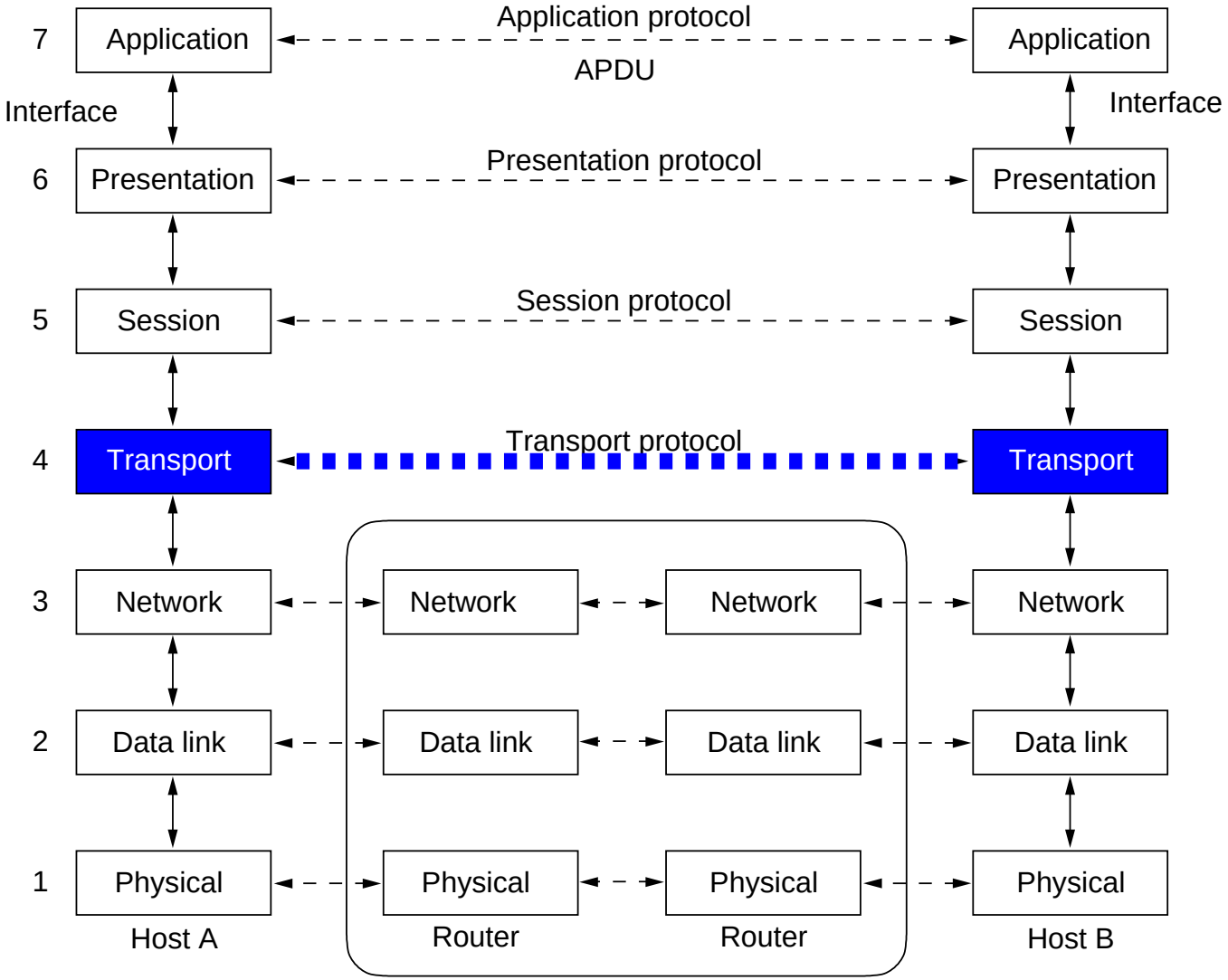
TCP : un protocole en mode orienté connexion

Couche Transport

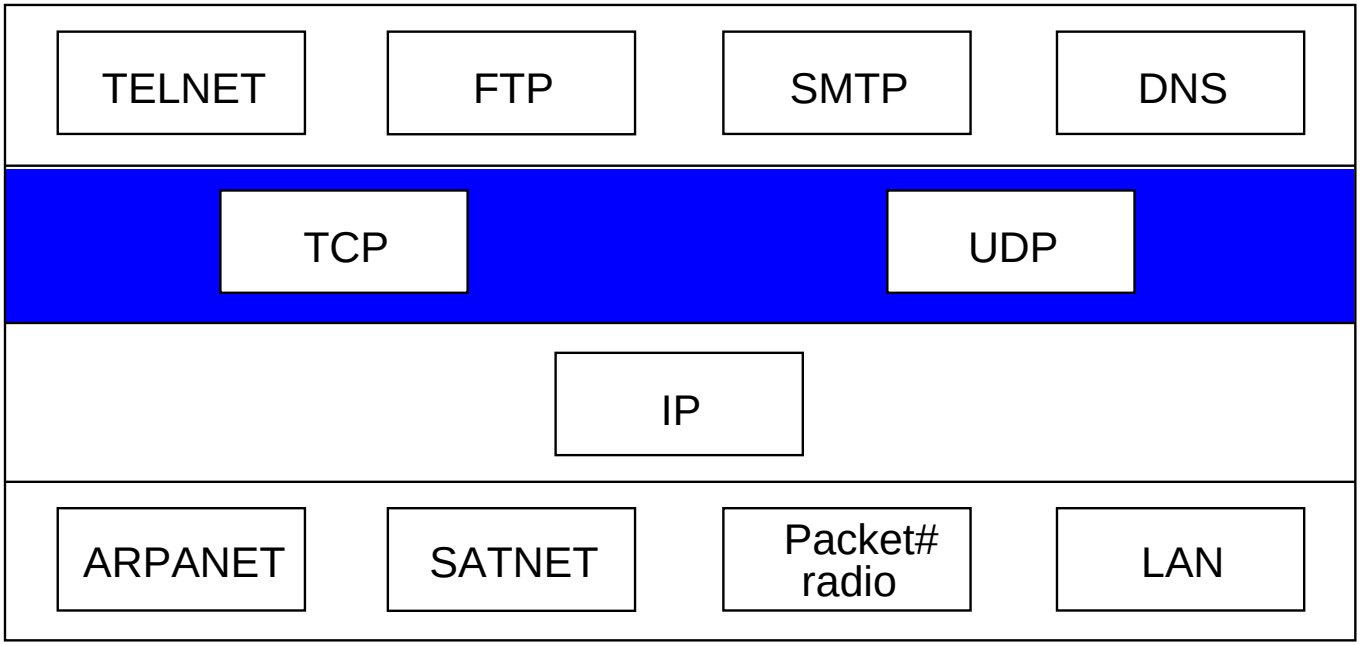
La **Couche Transport** permet de faire **communiquer directement** deux ou plusieurs entités sans avoir à se préoccuper des différents éléments de réseaux traversés :

- associations virtuelles
- de bout-en-bout (*end-to-end*)
- abstraction de la **topologie** et des **technologies** sous-jacentes
- services génériques potentiellement attendus par les applications :
  - ✓ contrôle d'erreur
  - ✓ fiabilité
  - ✓ ordonnancement
  - ✓ contrôle de flux
  - ✓ contrôle de congestion

Couche Transport : OSI

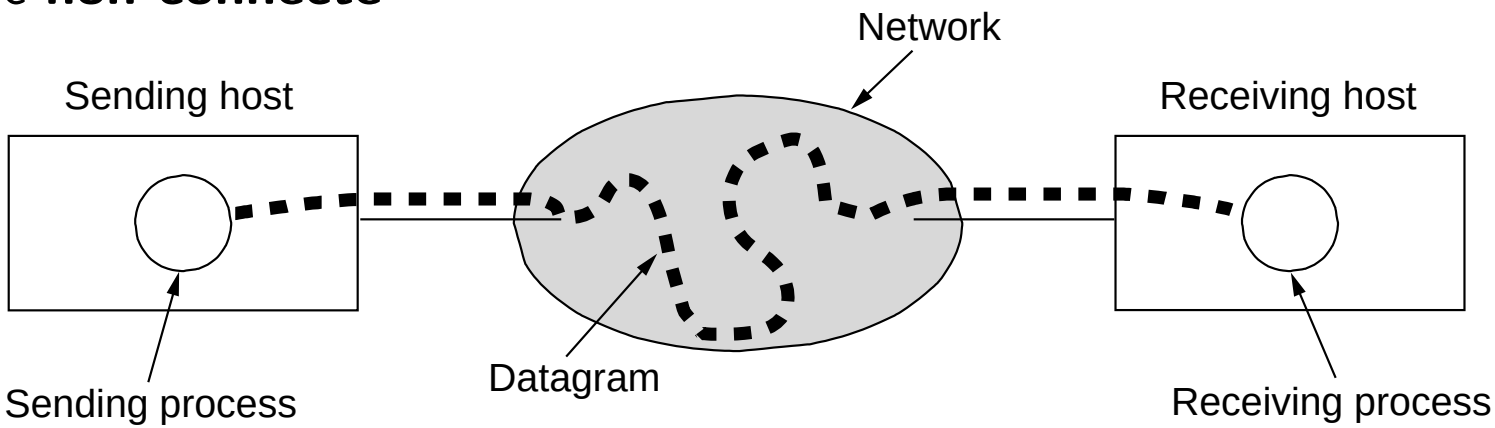


Couche Transport : TCP/IP

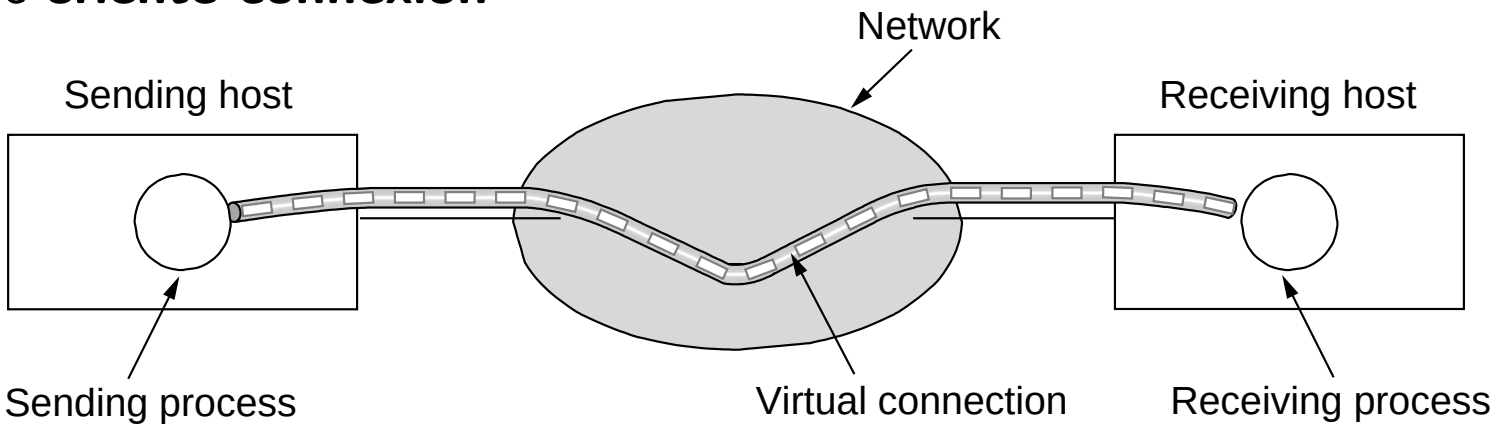


Couche Transport : 2 approches

Mode **non connecté**



Mode **orienté connexion**

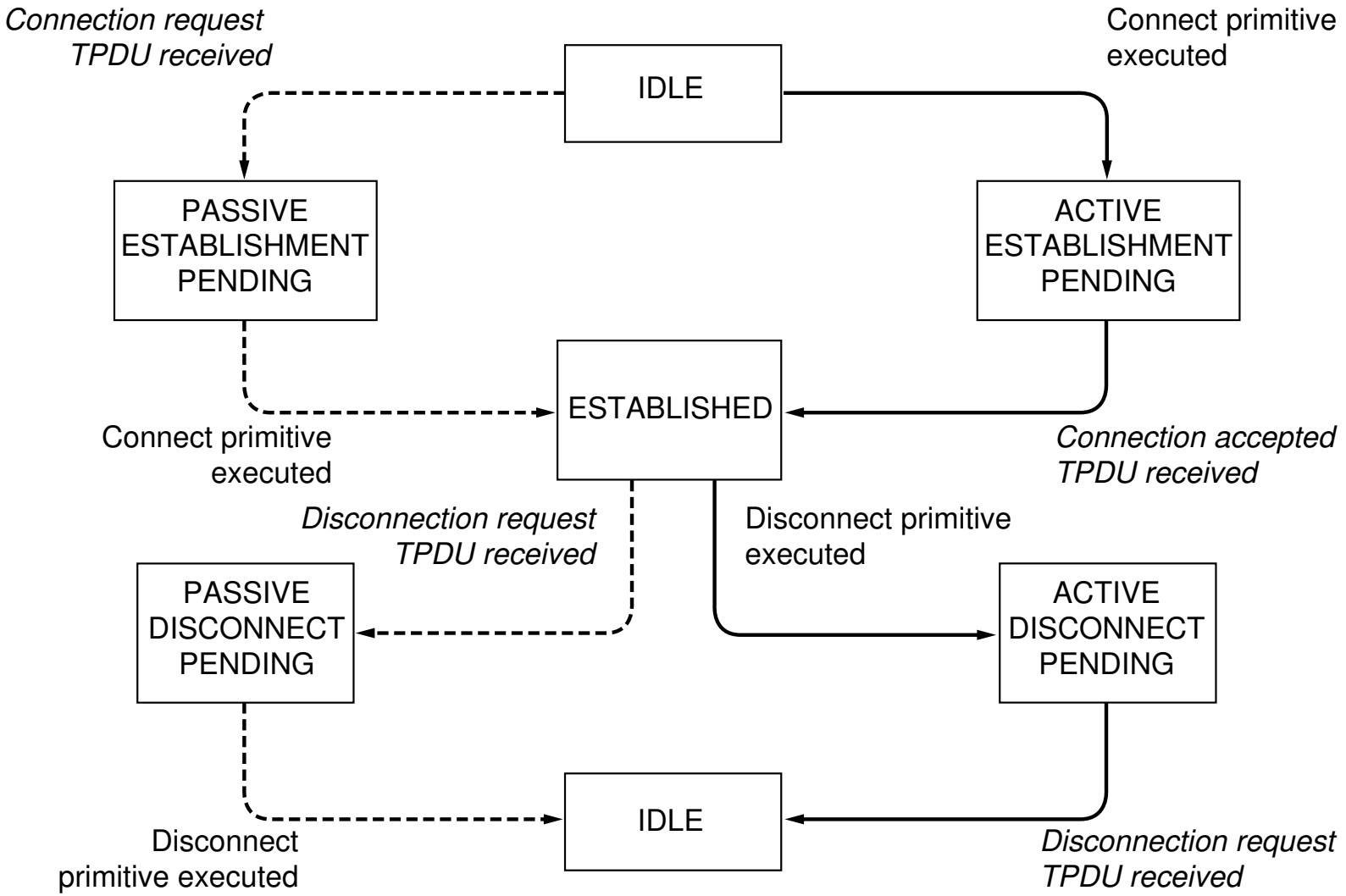


Couche Transport : Primitives

Interface de **programmation** (applications ou developpeurs)

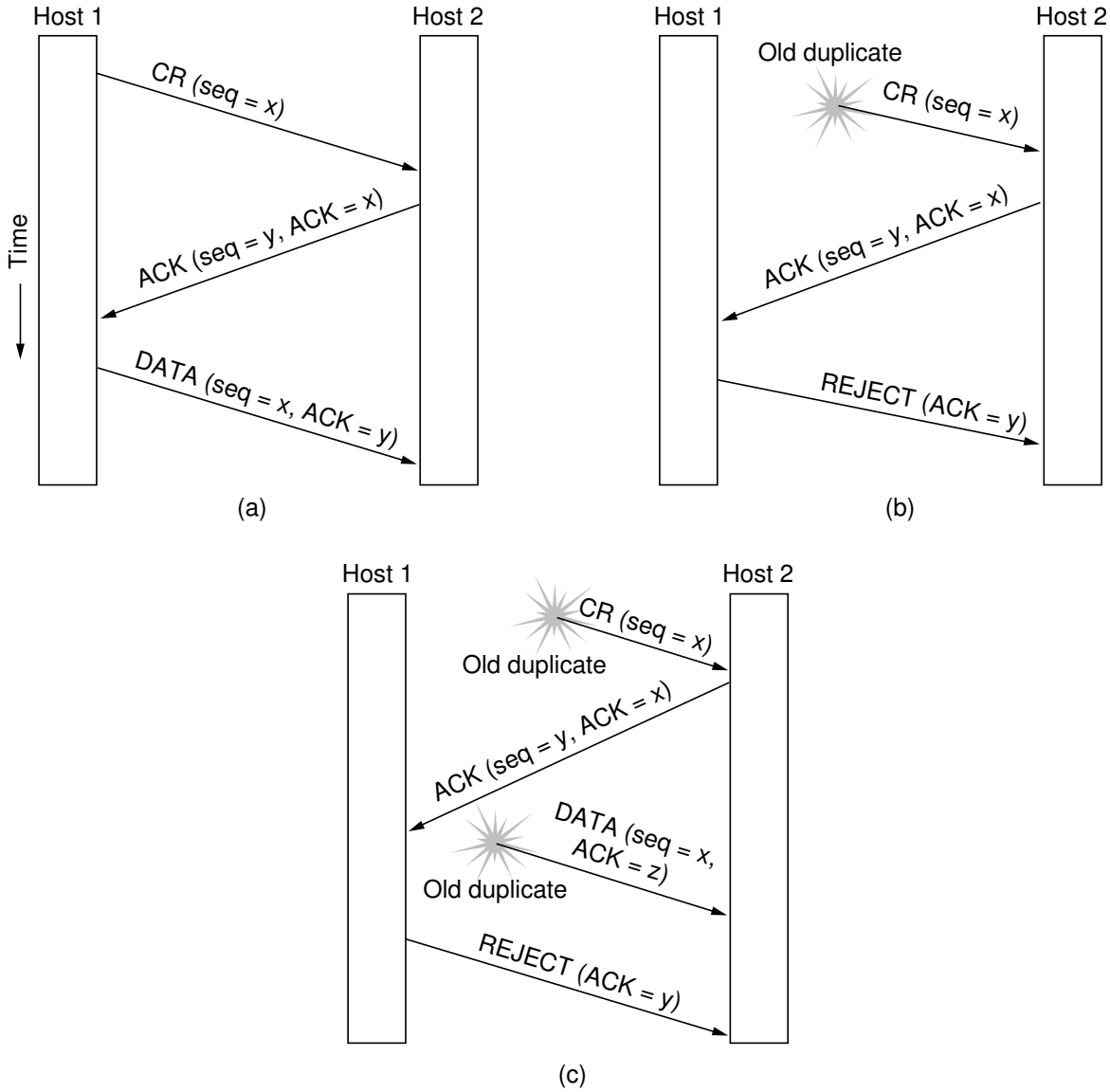
- exemple de primitives de base en mode connecté :
  - ✓ LISTEN
  - ✓ CONNECT
  - ✓ SEND
  - ✓ RECEIVE
  - ✓ DISCONNECT

Couche Transport : Automate de connexion

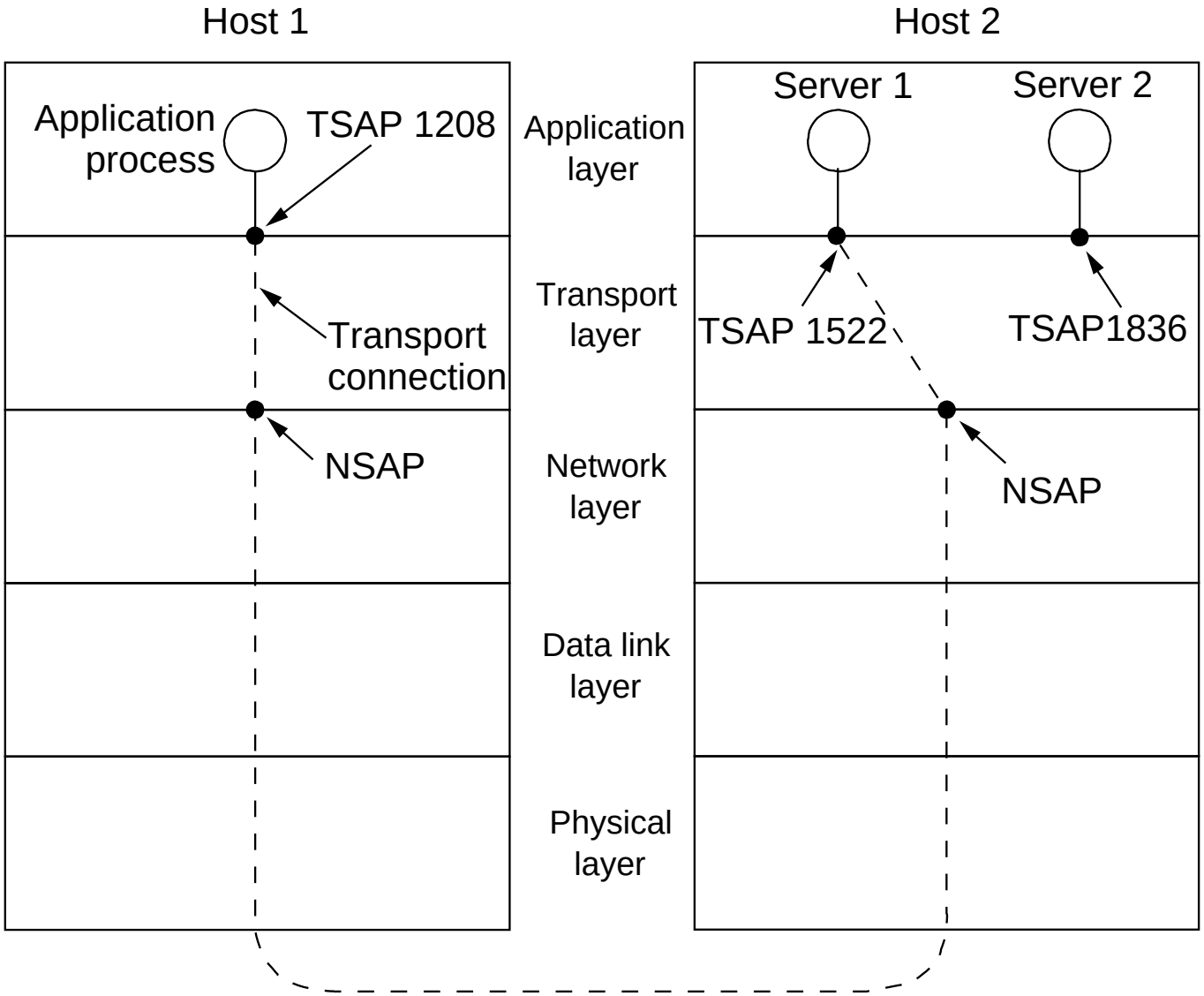


pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Couche Transport : Etablissement (*call setup*)



Couche Transport : Multiplexage (1)



Couche Transport : Multiplexage (2)

```
Unix> cat \etc\services

tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp
discard         9/udp
systat          11/tcp
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp
chargen         19/tcp
chargen         19/udp
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp
ssh             22/udp
telnet          23/tcp
smtp            25/tcp
time            37/tcp
time            37/udp
whois           43/tcp
domain          53/tcp
domain          53/udp
bootps          67/tcp
bootps          67/udp
bootpc          68/tcp
bootpc          68/udp
tftp            69/udp
gopher          70/tcp
gopher          70/udp
finger          79/tcp
www             80/tcp
www             80/udp
kerberos        88/tcp
kerberos        88/udp
hostnames       101/tcp
iso-tsap        102/tcp
...
```

Plan

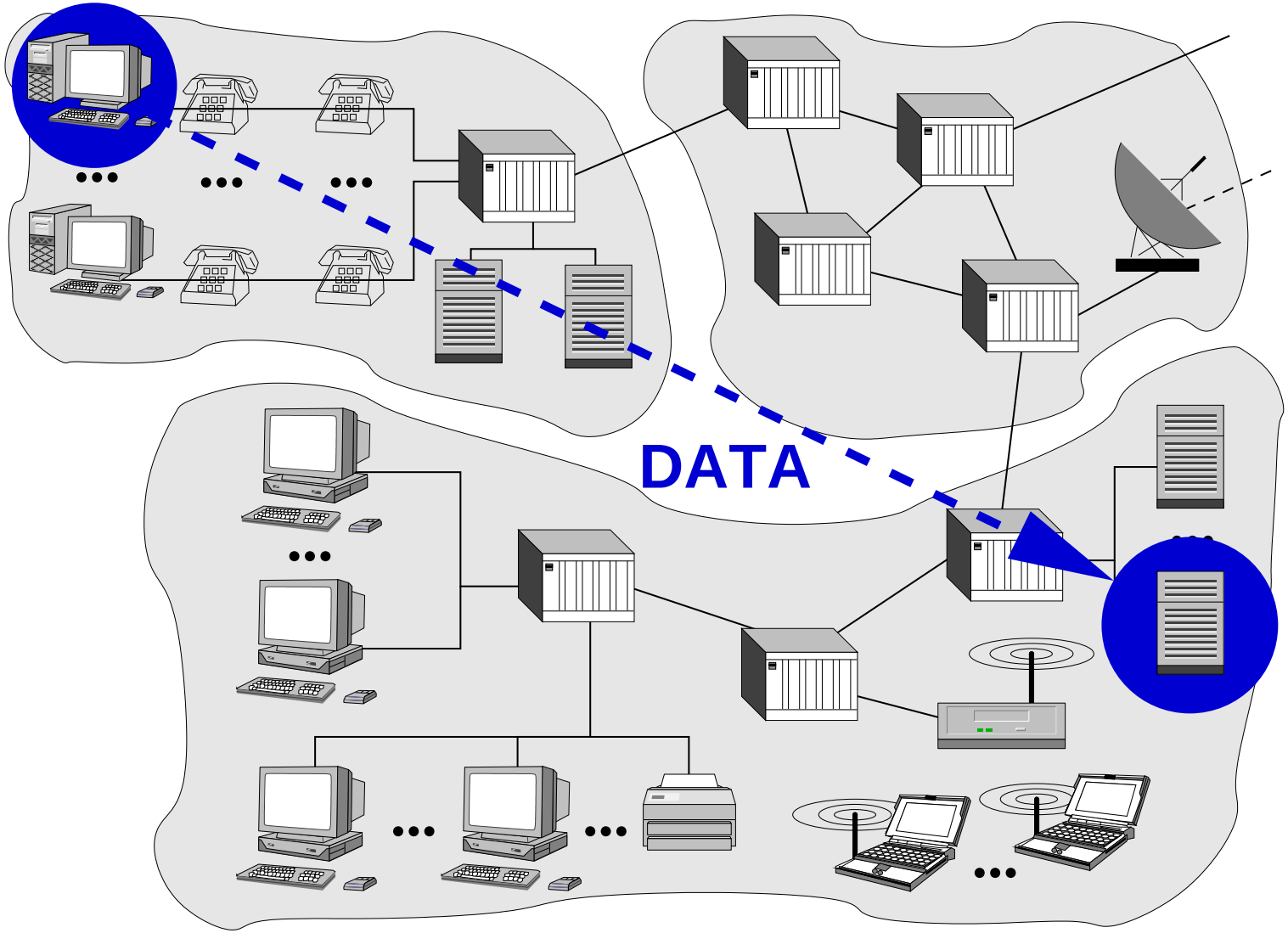
Rappels sur la couche transport

UDP : un protocole en mode non connecté

- format du datagramme UDP
- utilisation d'UDP

TCP : un protocole en mode orienté connexion

UDP



UDP : arguments pour un transport sans connexion

Le choix d'un service transport non connecté peut être motivé par :

- pas d'**établissement** de connexion
- absence d'**état** dans les hôtes
- **entête réduit**
- pas de **retransmission**
- pas de **contrôle** du débit d'émission

Plan

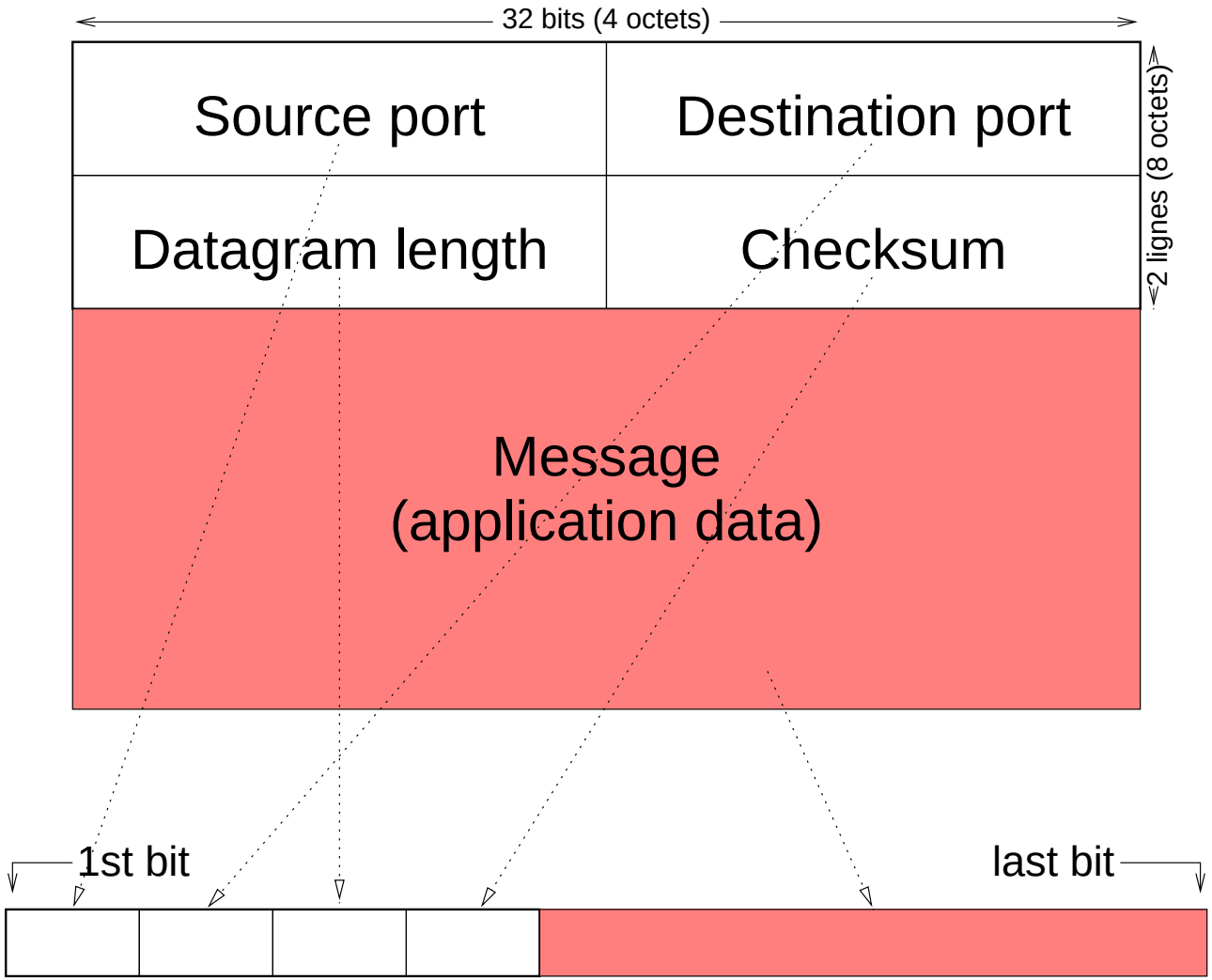
Rappels sur la couche transport

UDP : un protocole en mode non connecté

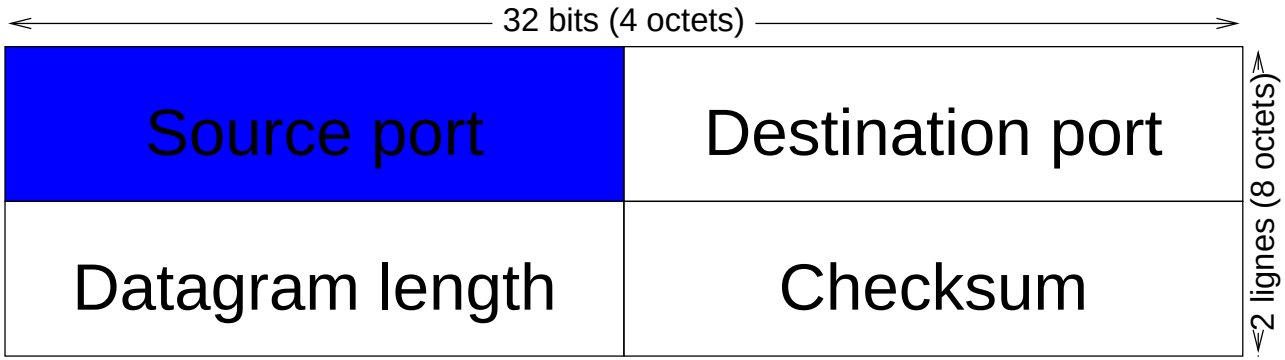
- **format du datagramme UDP**
- utilisation d'UDP

TCP : un protocole en mode orienté connexion

Datagramme UDP

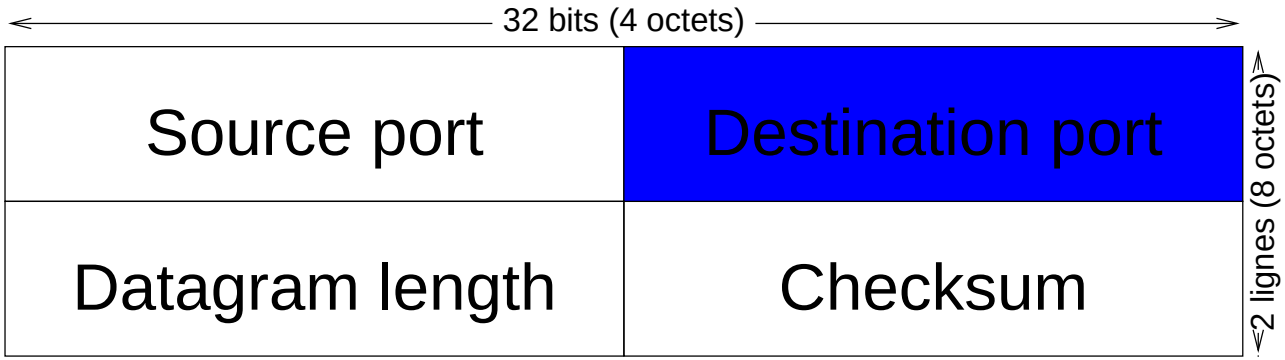


UDP : Port source



- multiplexage à la source (plusieurs clients vers un serveur)
- 16 bits (65535 ports)
  - ✓ `7` `echo`
  - `13` `daytime`
  - `53` `domain`
  - `69` `tftp`
  - `161` `snmp`
  - `162` `snmp-trap`
  - `1027` `(allocation fixe ou dynamique d'un client)`
  - `...`

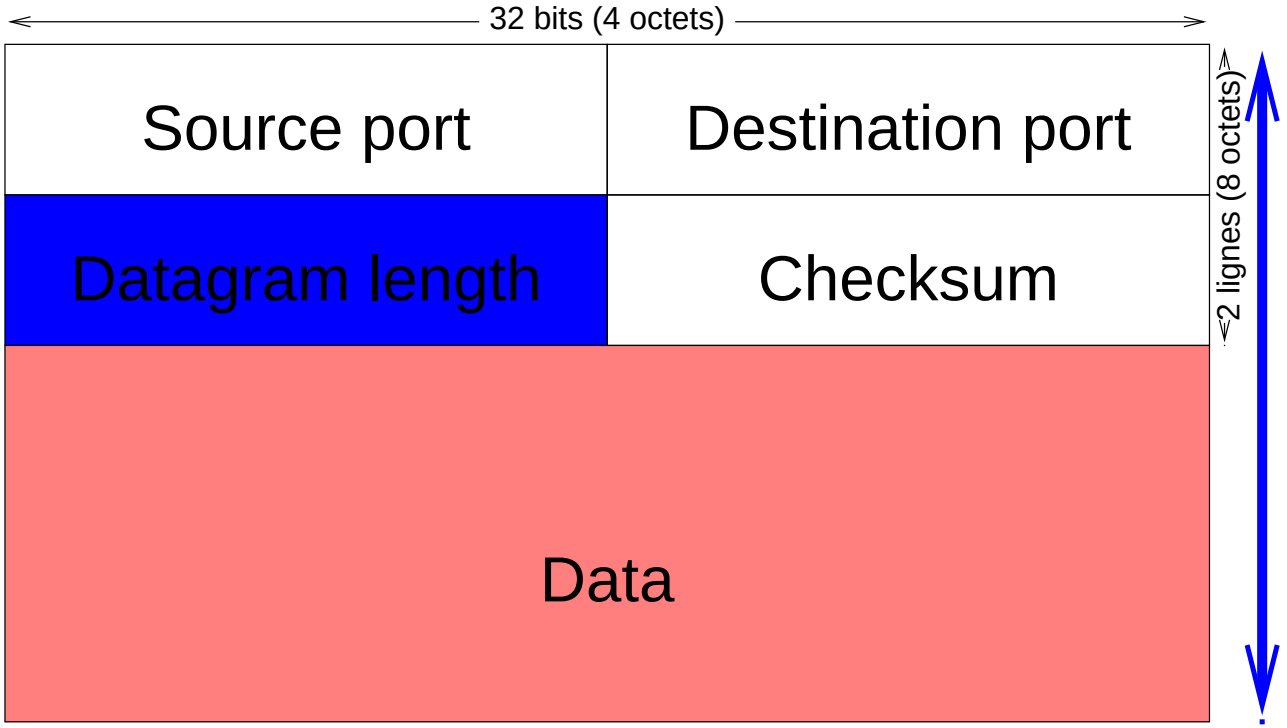
UDP : Port destination



- multiplexage à la destination (plusieurs serveurs sur un hôte)
- le serveur doit être en écoute sur ce port pour accepter le datagramme
- 16 bits (65535 ports)

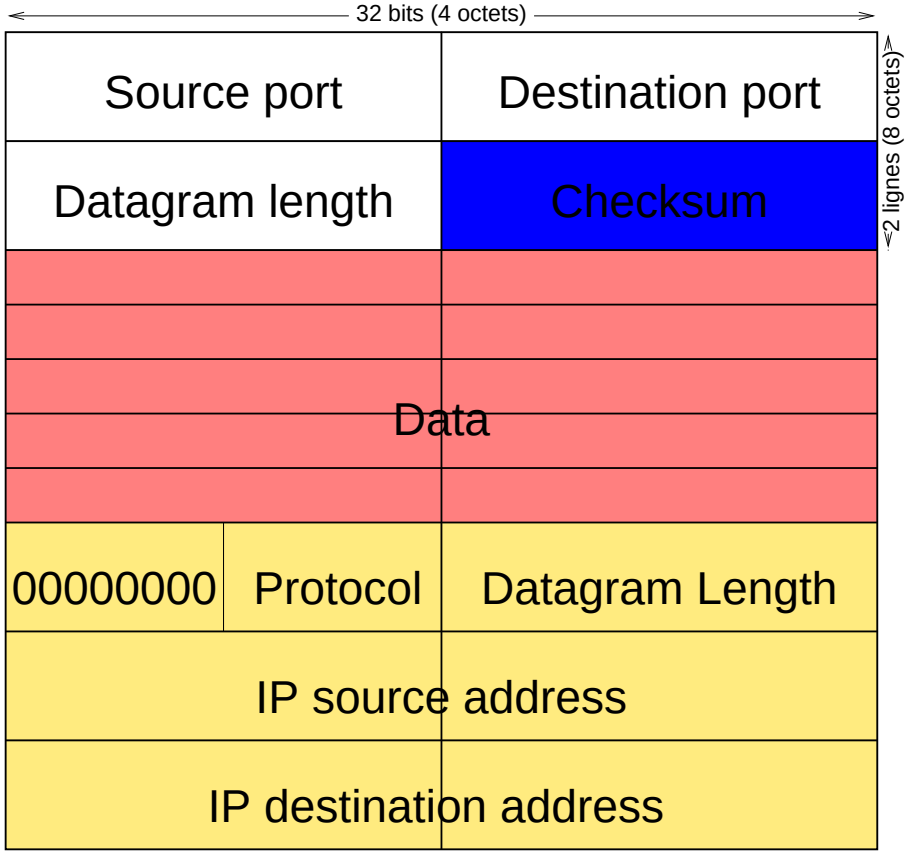
✓	7	echo
	13	daytime
	53	domain
	69	tftp
	161	snmp
	162	snmp-trap
	2345	(port fixé à l'avance et connu du client)
	...	

UDP : Longueur du datagramme



- longueur totale avec les données exprimée en **octets**
- 16 bits (64 Koctets maximum)

UDP : Somme de contrôle du datagramme



- contrôle d'erreur **facultatif**
- nécessaire si les couches sous-jacentes n'effectuent pas de contrôle
- *pseudo-header*
- 16 bits
- calcul rudimentaire : $\overline{\sum^{\overline{1}} mot_{16\text{bits}}}$

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

- format du datagramme UDP
- **utilisation d'UDP**

TCP : un protocole en mode orienté connexion

UDP : exemples d'applications

Les applications suivantes reposent typiquement sur UDP :

- résolution de noms (DNS)
- serveur de fichiers distants (NFS)
- protocole de routage (RIP)
- administration du réseau (SNMP)
- diffusion multimédia, *streaming* (propriétaire)
- téléphonie sur Internet (propriétaire)
- visioconférence, (propriétaire)
- ...

et toutes les applications multicast

UDP : Interface socket

```
#include <sys/types.h>
#include <sys/socket.h>

# Create a descriptor
int socket(int domain, int type, int protocol);
#   domain : PF_INET for IPv4 Internet Protocols
#     type : SOCK_DGRAM Supports datagrams (connectionless, unreliable msg of a fixed max length)
# protocol : UDP (/etc/protocols)

# Bind local IP and port
int bind(int s, struct sockaddr *my_addr, socklen_t addrlen);

# Send an outgoing datagram to a destination address
int sendto(int s, const void *msg, size_t len, int flags,
                                        const struct sockaddr *to, socklen_t tolen);

# Receive the next incoming datagram and record is source address
int recvfrom(int s, void *buf, size_t len, int flags,
                                        struct sockaddr *from, socklen_t *fromlen);

# End : dealocate
int close(int s);
```

Plan

Rappels sur la couche transport

UDP : un protocole en mode non-connecté

TCP : un protocole en mode orienté connexion

- format du segment TCP
- gestion de la connexion
- gestion de la fiabilité
- contrôle de flux
- contrôle de congestion
- implémentation : voyage au Nevada
- utilisation de TCP

Mode orienté connexion

Caractéristiques associées au mode orienté connexion :

- état (mémoire)
- protocole de gestion de la connexion :
  - ✓ création
  - ✓ maintien
  - ✓ suppression
- bidirectionnel
- services avancés :
  - ✓ fiabilité
  - ✓ ordonnancement
  - ✓ contrôles d'émissions
  - ✓ ...

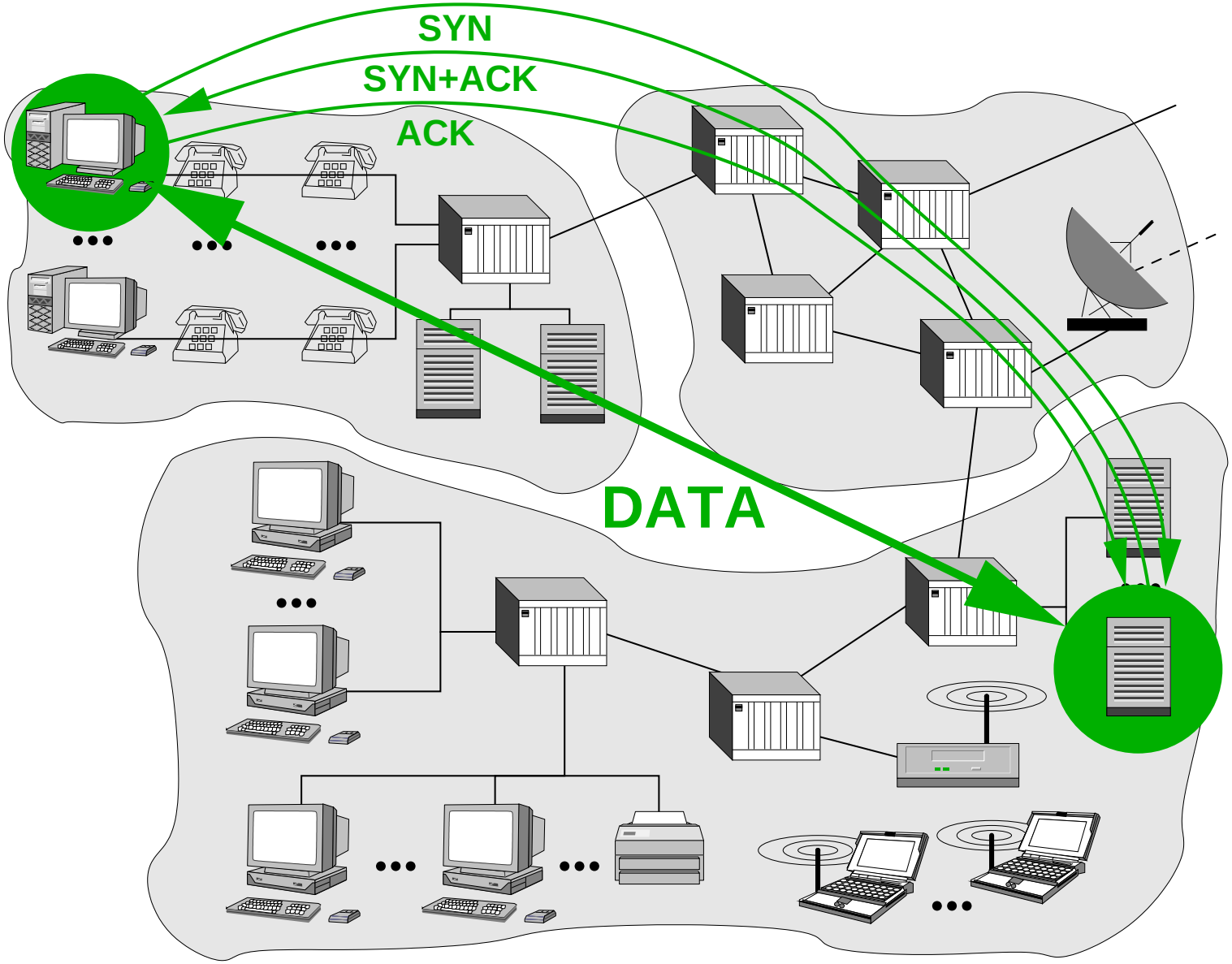
Vers un service fiable

La couche transport doit assurer la fiabilité afin d'être indépendante des couches inférieures.

Mécanismes associés à la fiabilisation de la connexion :

- **ARQ** (*Automatic Repeat reQuest*)
  - ✓ **acquittement** positif (ACK) ou négatif (NACK)
  - ✓ **détection** d'erreur
  - ✓ **retour** d'information (*receiver feedback*)
  - ✓ **retransmission**
- fenêtre d'anticipation
  - ✓ protocole *stop-and-wait*
  - ✓ protocole *pipeline*
    - ☞ numéros de séquence
    - ☞ acquittements cumulatifs ou sélectifs
    - ☞ retransmissions *Go-Back-N* ou sélectives

TCP (1)



TCP (2)

Caractéristiques de base :

- service fiable
- connexion **bidirectionnelle** (*full duplex*)
- **point-à-point**
- ouverture en trois échanges (*three-way handshake*)
- fermetures courtoise ou brutale
- orienté **octet**
- taille maximum de segments (bloc d'octets)
  - ✓ MSS (*Maximun Segment Size*) = taille des *data* !
  - ✓ MSS typiques : 1460, 1448 et 512
- structure...

Plan

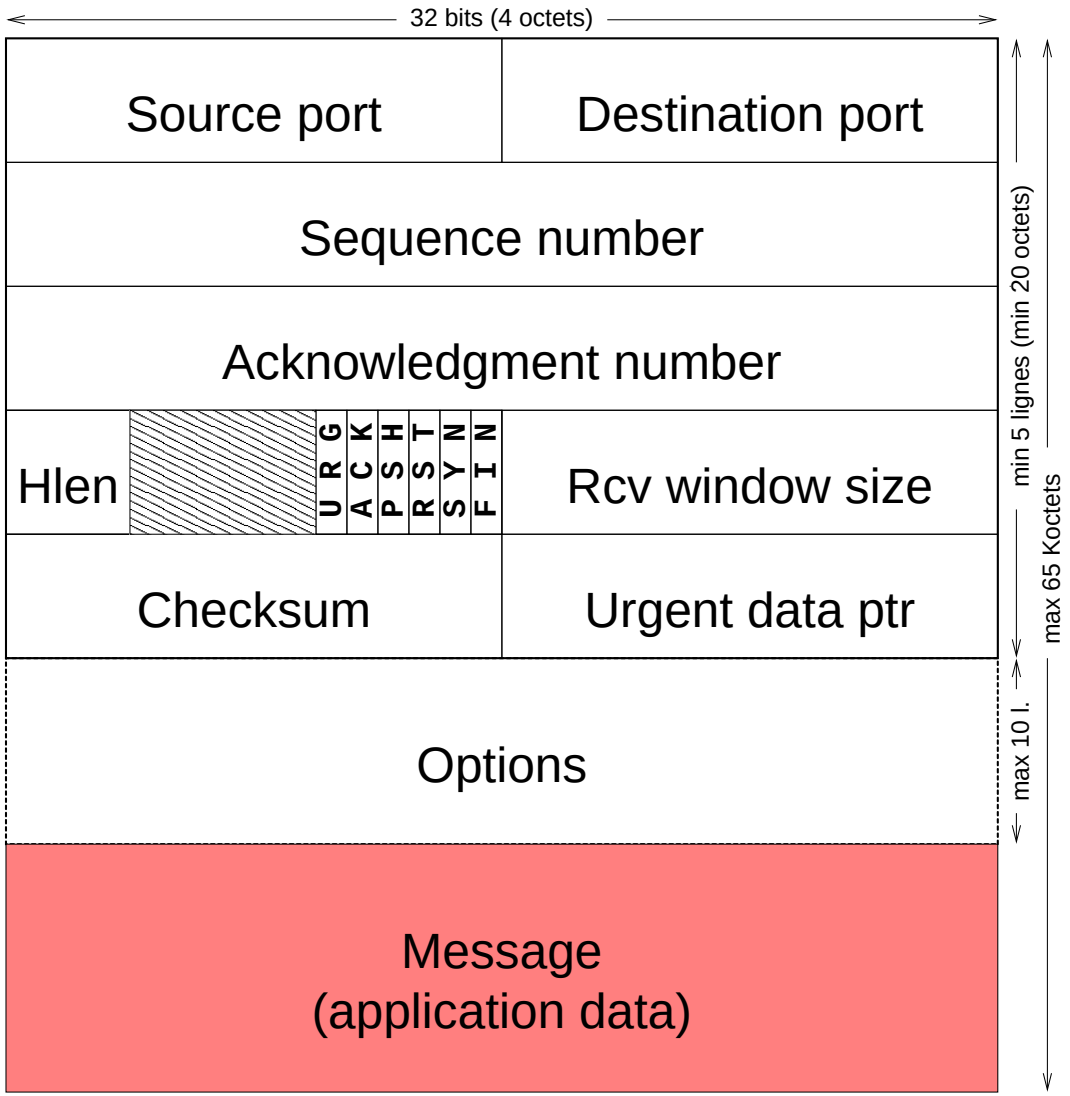
Rappels sur la couche transport

UDP : un protocole en mode non connecté

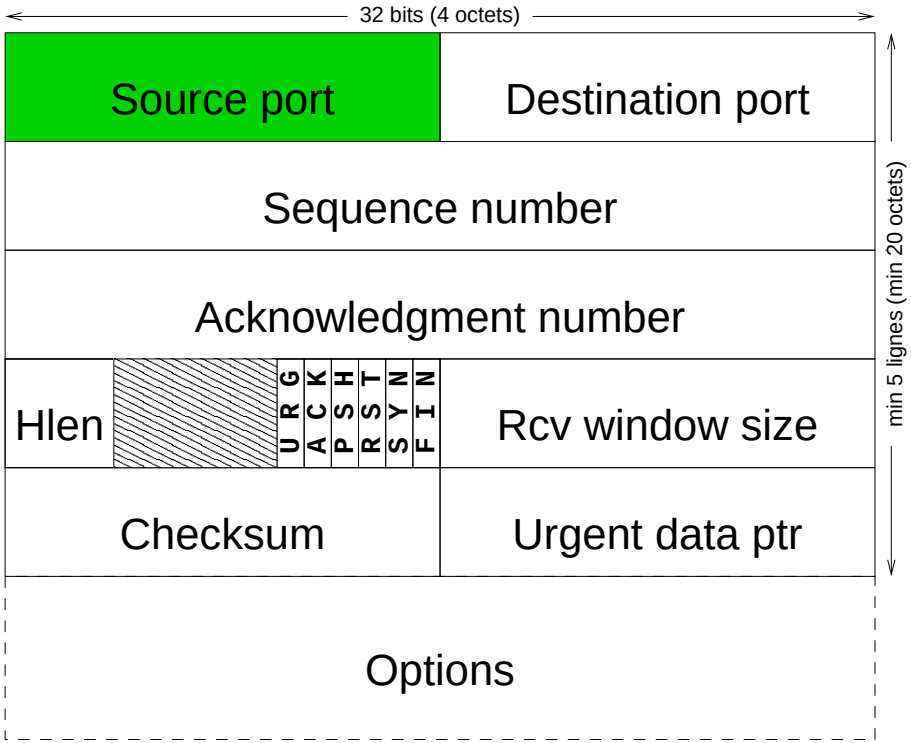
TCP : un protocole en mode orienté connexion

- **format du segment TCP**
- gestion de la connexion
- gestion de la fiabilité
- contrôle de flux
- contrôle de congestion
- implémentation : voyage au Nevada
- utilisation de TCP

Segment TCP

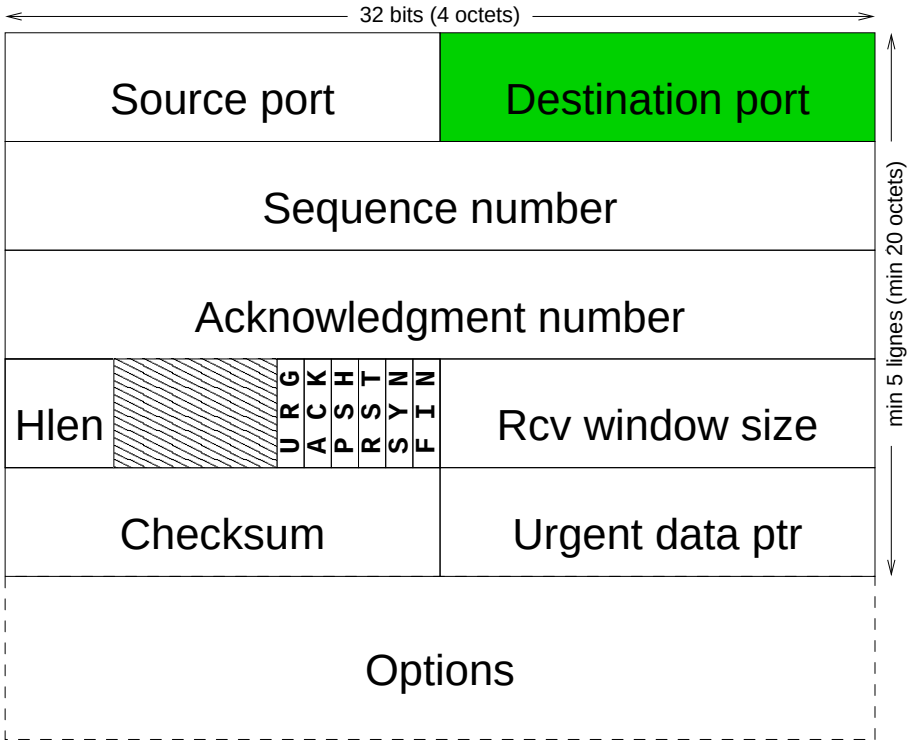

32 bits (4 octets)
Source port
Destination port
Sequence number
Acknowledgment number
Hlen
URG
ACK
PSH
RST
SYN
FIN
Rcv window size
Checksum
Urgent data ptr
Options
Message
(application data)
min 5 lignes (min 20 octets)
max 10 l.
max 65 Koctets

TCP : Port source



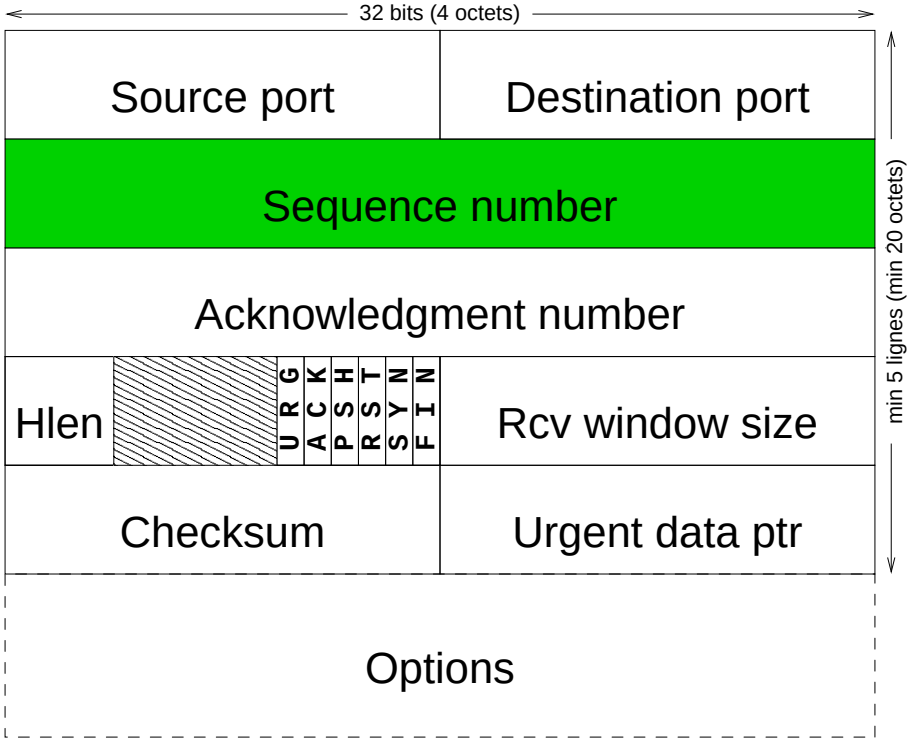
- identification unique de l'application **source**
- **multiplexage** à la source (plusieurs clients vers un serveur)
- 16 bits (65535 ports)
  - ✓ `20` `ftp`
  - `80` `www`
  - `1027` `(port utilisateur fixe ou dynamique)`

TCP : Port destination



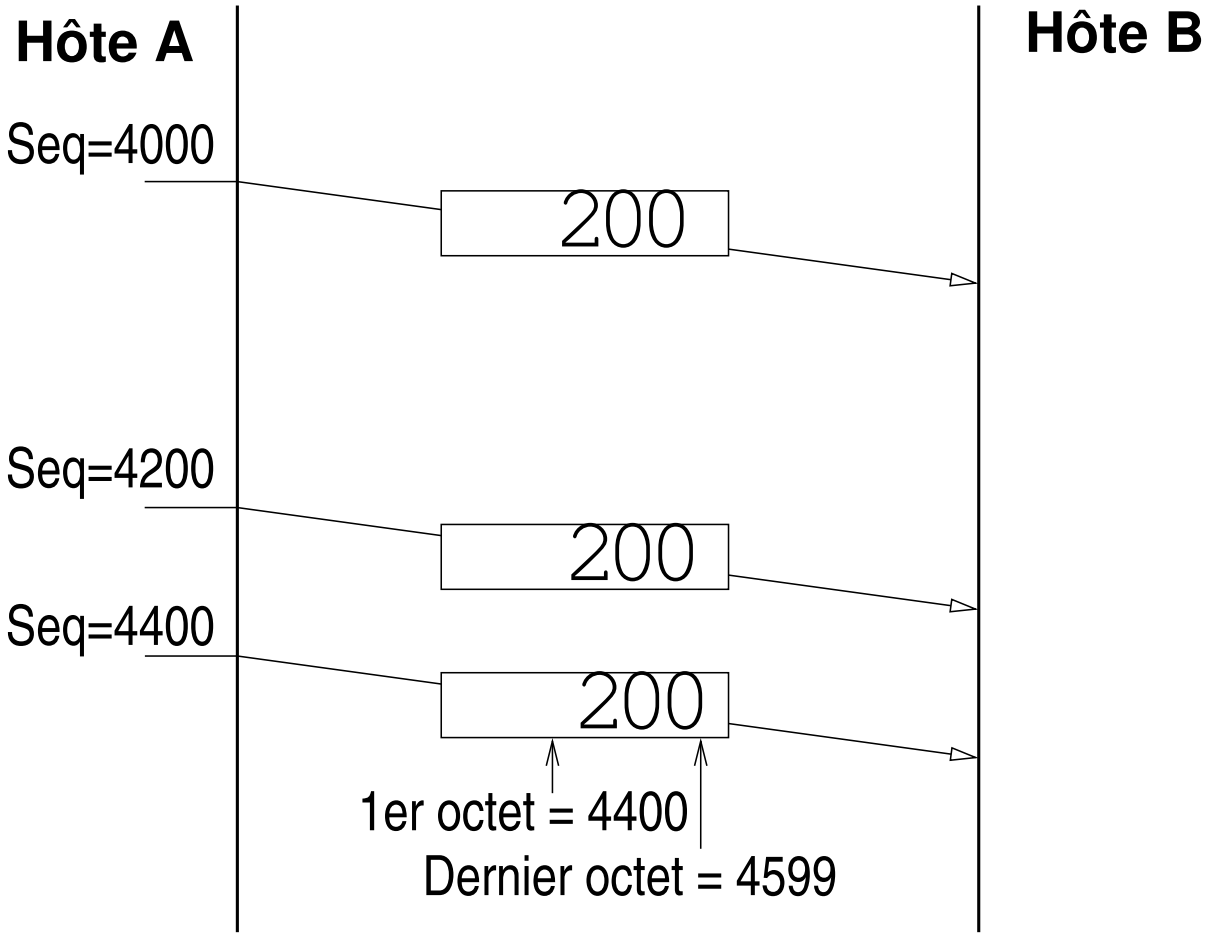
- identification unique de l'application **destination**
- **multiplexage** à la destination (plusieurs serveurs sur un hôte)
- 16 bits (65535 ports)
  - ✓ `20` `ftp`
  - `80` `www`
  - `2345` `(port utilisateur fixe)`

TCP : Numéro de séquence (1)

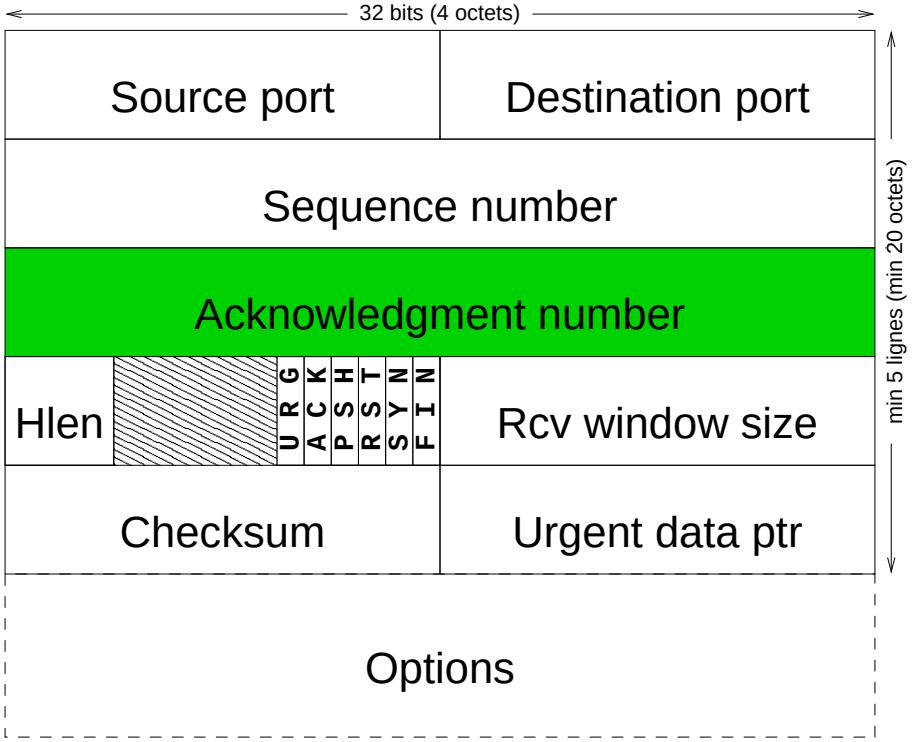


- associé à chaque **octet** (et non pas à un segment)
- numérote le **premier** octet des *data*
- numérotation implicite des octets suivants
- détection des **pertes**
- **ordonnancement**
- 32 bits (boucle au bout de 4 Goctets)

TCP : Numéro de séquence (2)

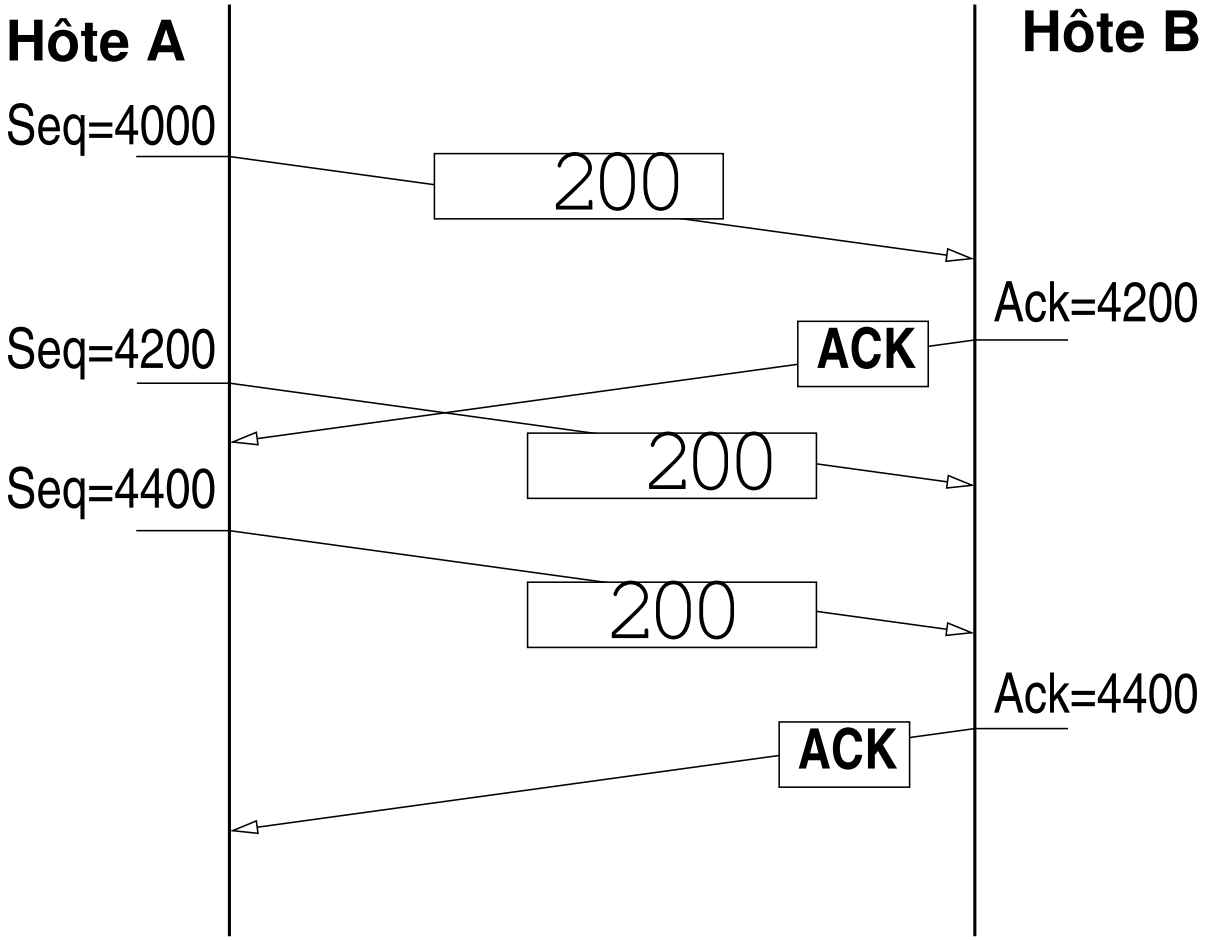


TCP : Numéro d'acquitement (1)



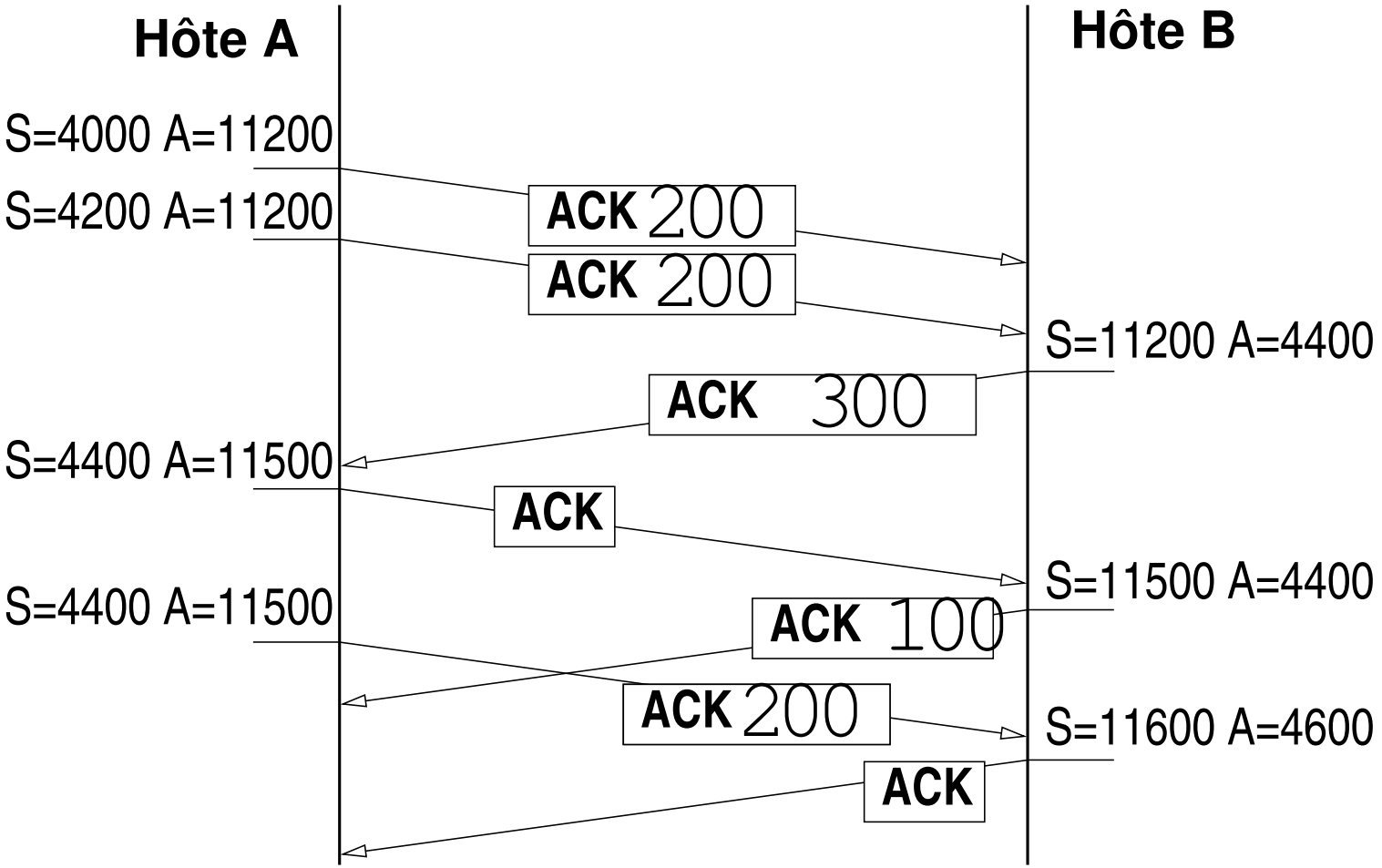
- indique le numéro du prochain octet attendu
- **cumulatif**, indique le premier octet non reçu (d'autres peuvent avoir été reçus avec des numéros de séquence supérieurs)
- associé à une retransmission *Go-Back-N*
- *piggybacking*
- 32 bits

TCP : Numéro d'acquitement (2)

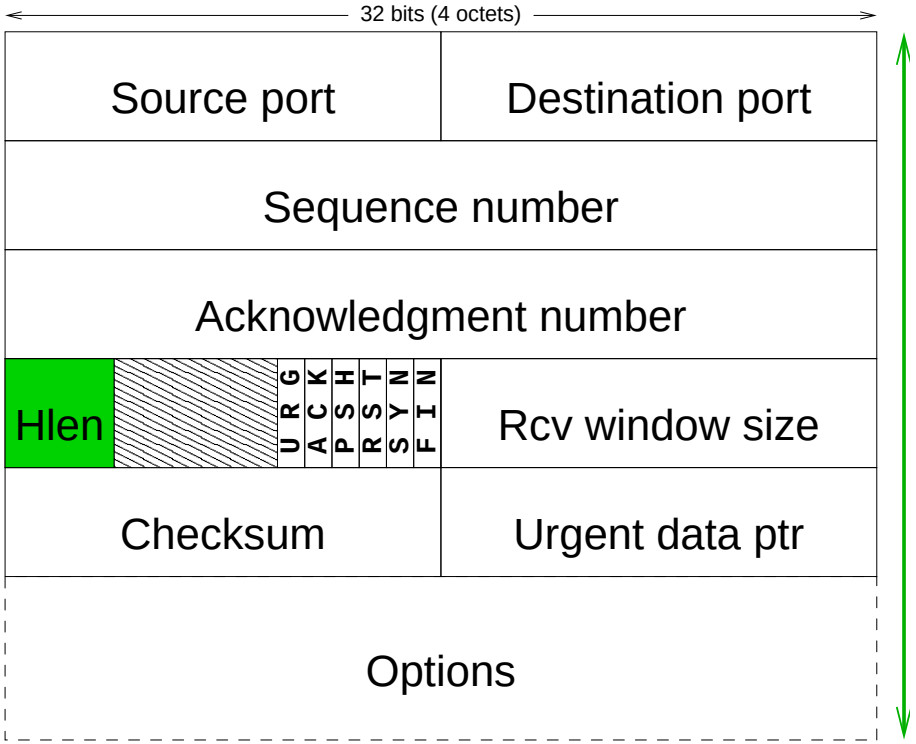


TCP : Numéro d'acquitement (3)

Piggybacking

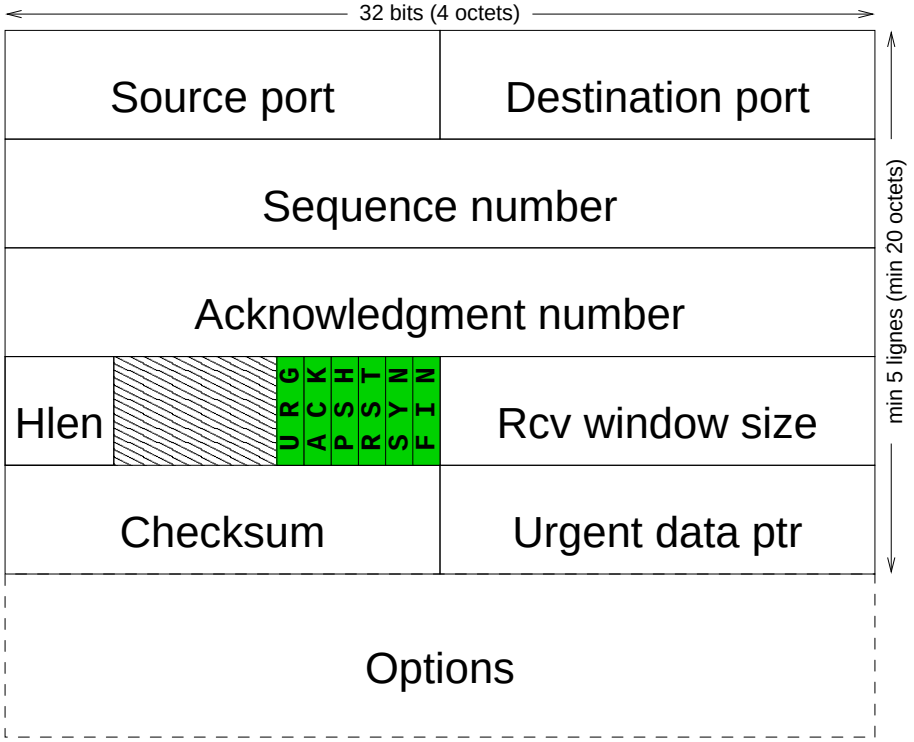


TCP : Longueur de l'entête



- nombre de lignes de 32 bits dans l'entête TCP
- nécessaire car le champ option est de longueur variable (20 à 60 octets)
- valeur de 5 (pas d'options) à 15 (10 lignes d'options, soit 40 octets)
- 4 bits (valeur "0" à 15 maximum)

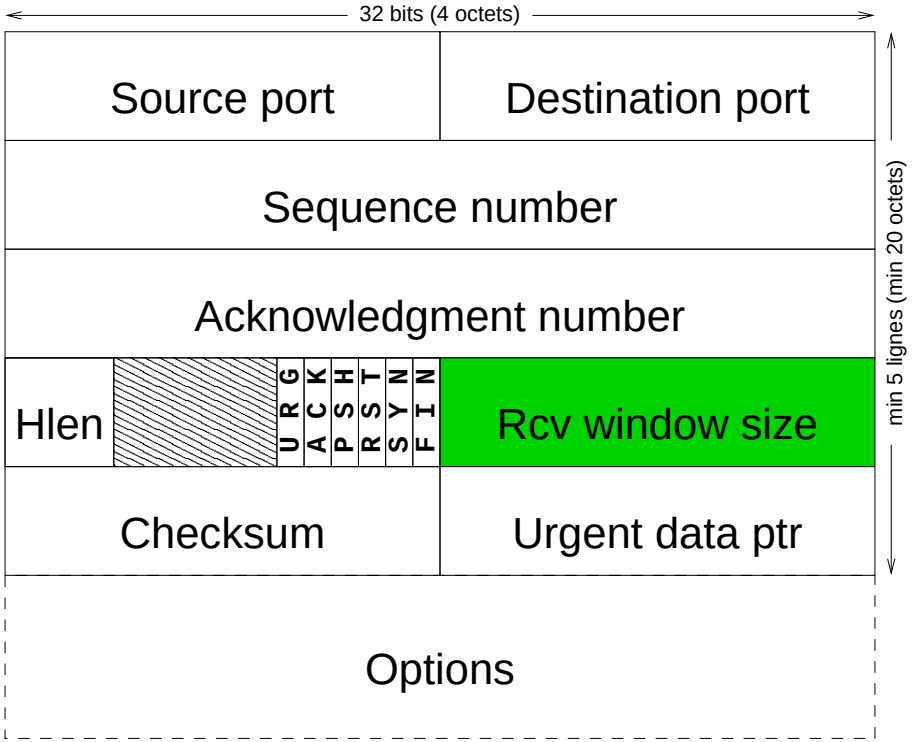
TCP : Indicateurs (*flags*)



Chacun sur 1 bit indique :

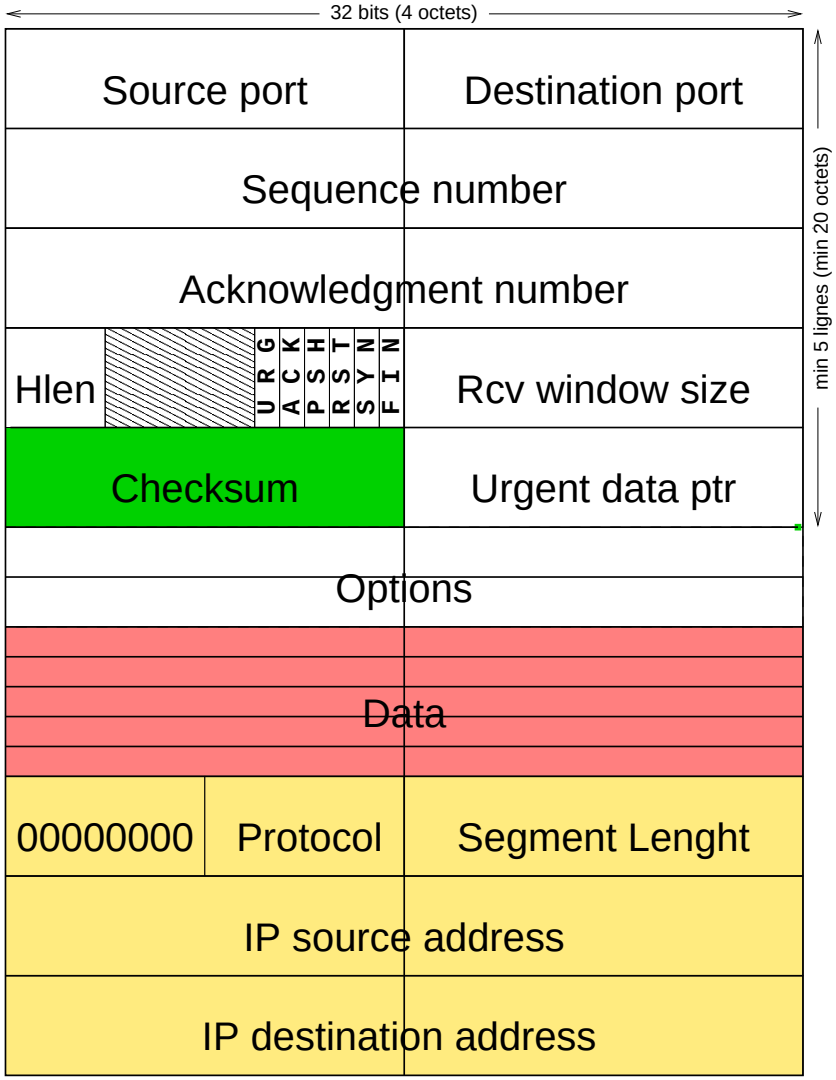
- `URG` : présence de données **urgentes**
- `ACK` : champ **acquittement** valide
- `PSH` : envoi **immédiat** avec vidage des tampons
- `RST` : **terminaison** brutale de la connexion
- `SYN` : synchronisation lors de l'**ouverture**
- `FIN` : **fermeture** courtoise

TCP : Taille de la fenêtre de réception



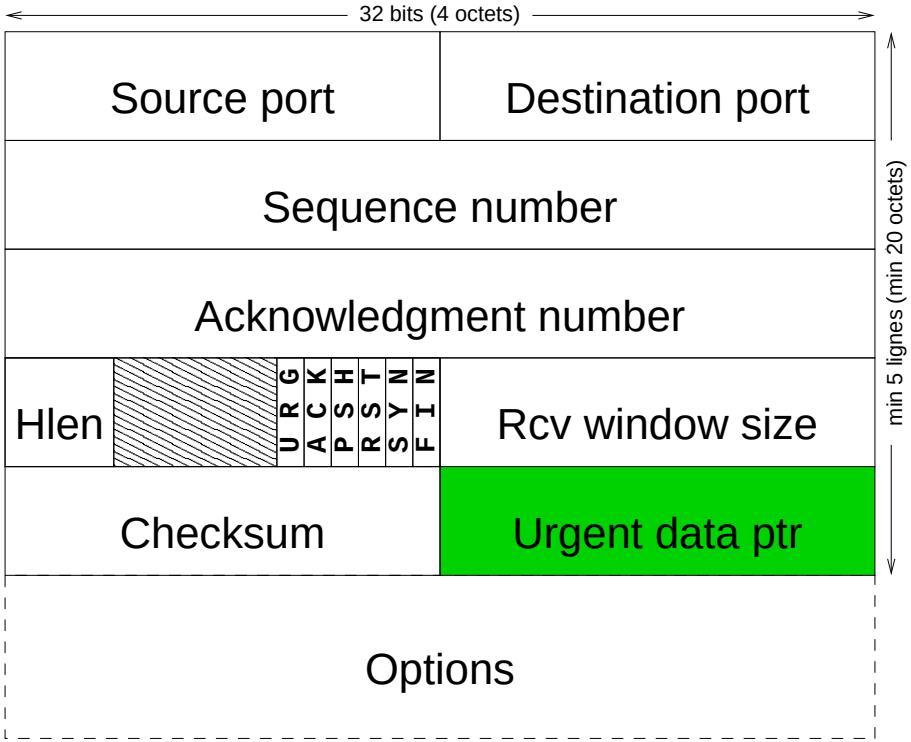
- indique le nombre d'octets disponibles du coté du récepteur
- dimentionne la taille de la fenêtre d'anticipation (coulissante) de l'émetteur
- **contrôle de flux**
- *piggybacking*
- 16 bits (le récepteur annonce 64 Koctets maximum)

TCP : Somme de contrôle du segment



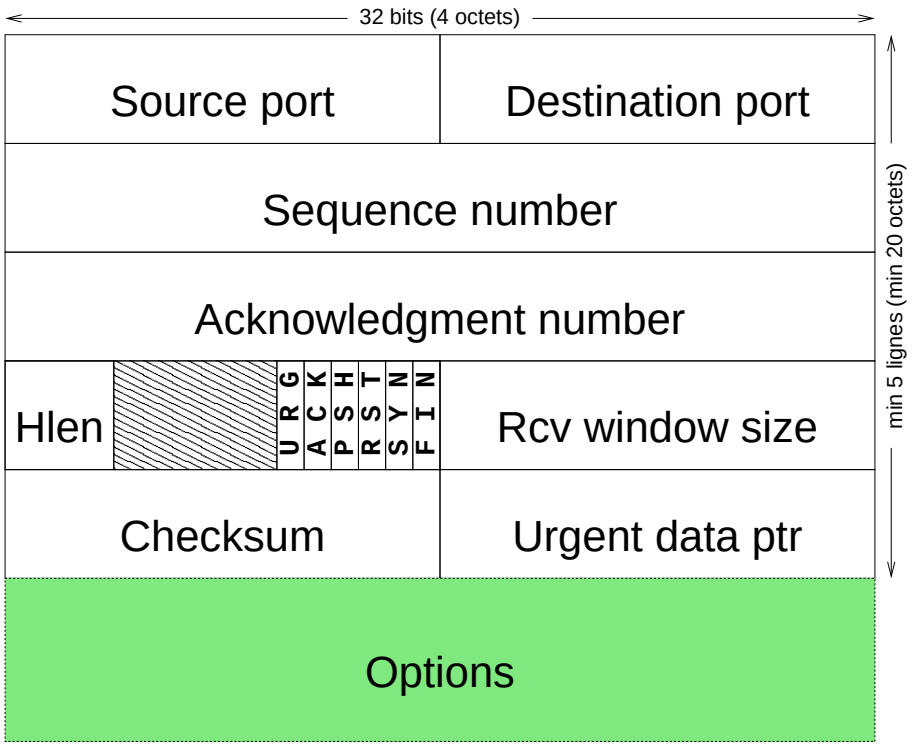
- contrôle d'erreur
- nécéssaire si les couches sous-jacentes n'effectuent pas de contrôle
- *pseudoheader*
- 16 bits
- calcul rudimentaire : $\overline{\sum^{\overline{1}} mot_{16\text{bits}}}$

TCP : Pointeur sur les données urgentes



- permet l'envoi de données spéciales (et non **hors bande**)
- délimite des données traitées en priorité
- indique la fin des données urgentes
  - ✓ interprétation de la quantité de données et de leur rôle par l'application
- 16 bits

TCP : Options



Les options sont de la forme TLV ou *Type, Length (*octets*), Value* :

- `END` : fin de liste d'options (T=0)
- `NOOP` : pas d'opération, bourrage (T=1)
- `MSS` : négociation du MSS (T=2, L=4, V=MSS)
- `WSIZE` : mise à l'échelle de la fenêtre par une puissance de deux (T=3, L=3, V=puissance)
- `SACK` : demande d'acquittement sélectif à l'ouverture (T=4, L=2)
- `SACK` : acquittement sélectif de n blocs (T=5, L=$2 + 8n$, $2n$ numéros de séquences)
- ...

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

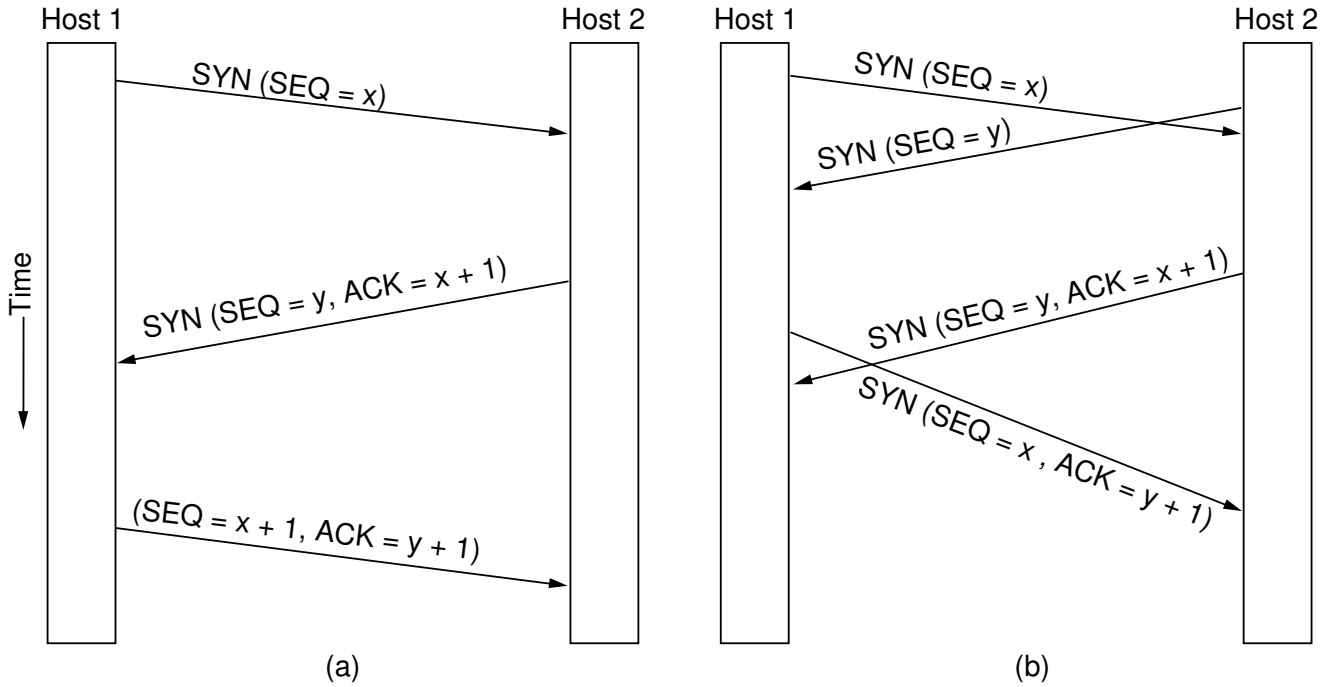
TCP : un protocole en mode orienté connexion

- format du segment TCP
- **gestion de la connexion**
- gestion de la fiabilité
- contrôle de flux
- contrôle de congestion
- implémentation : voyage au Nevada
- utilisation de TCP

TCP : Gestion de la connexion

Création de la connexion (*call setup*) :

- *three-way handshake*

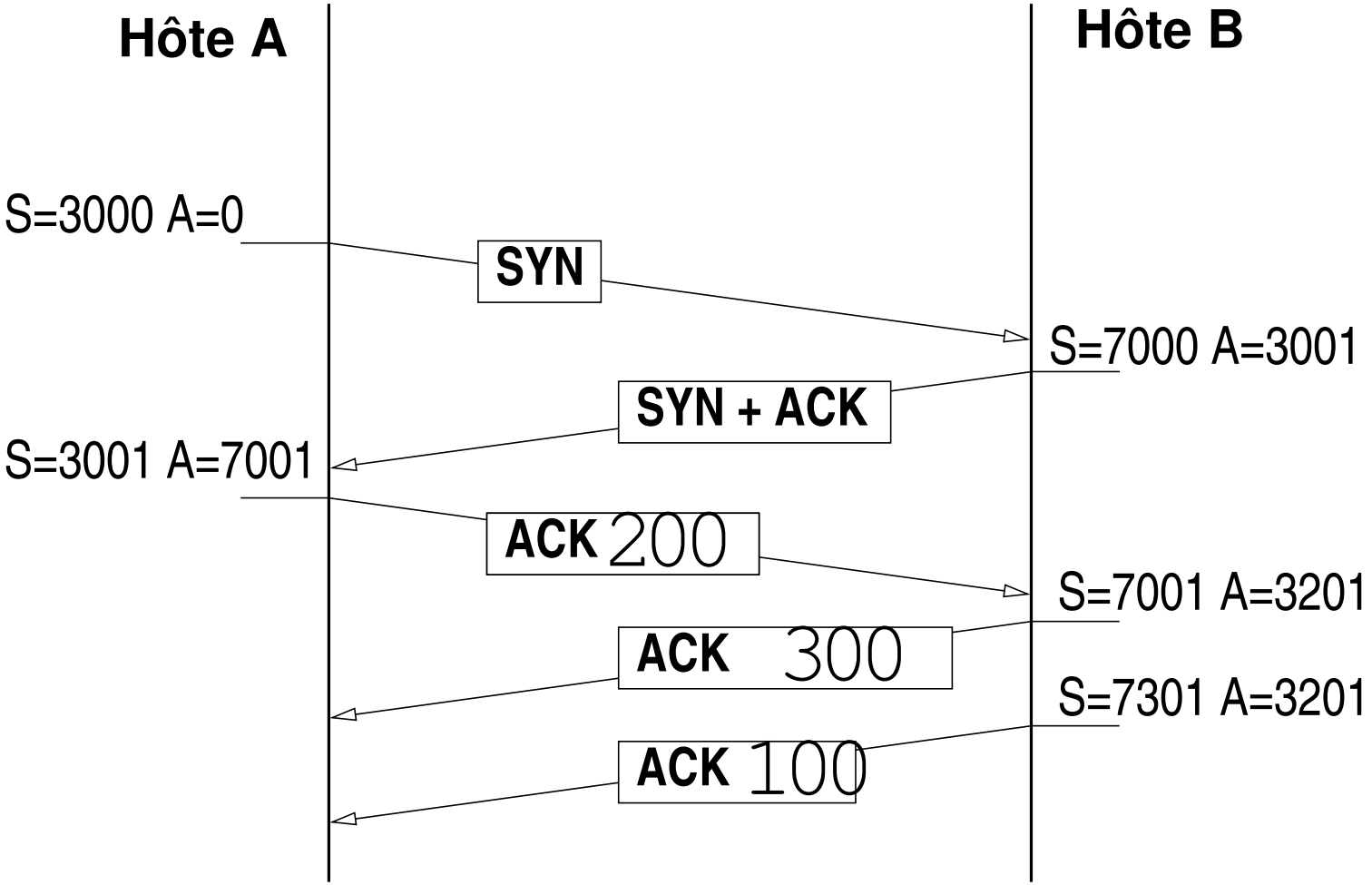


pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

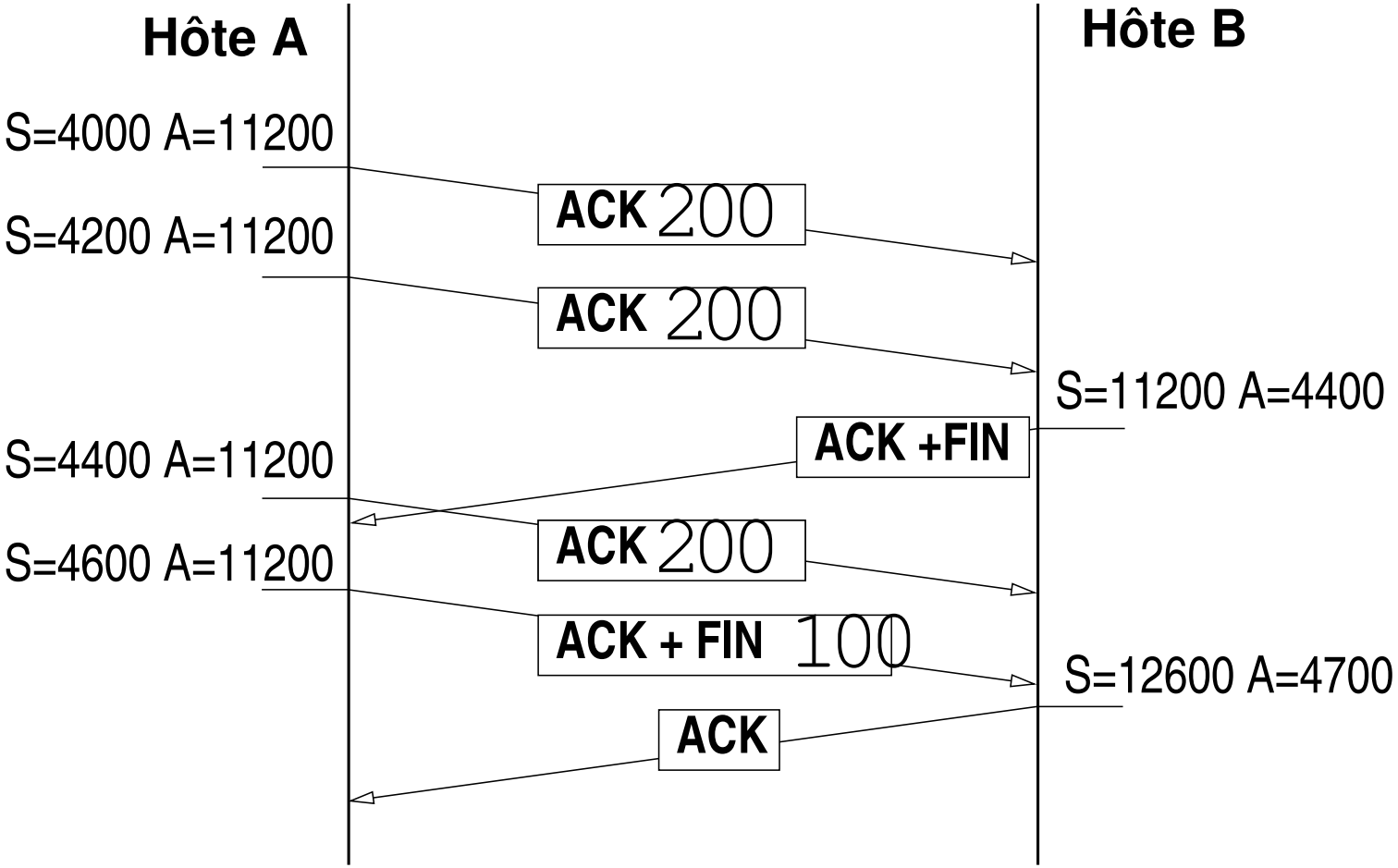
Déconnexion (*release*) :

- fermeture unilatérale (*shutdown*)
- fermeture courtoise (*gracefull release*)

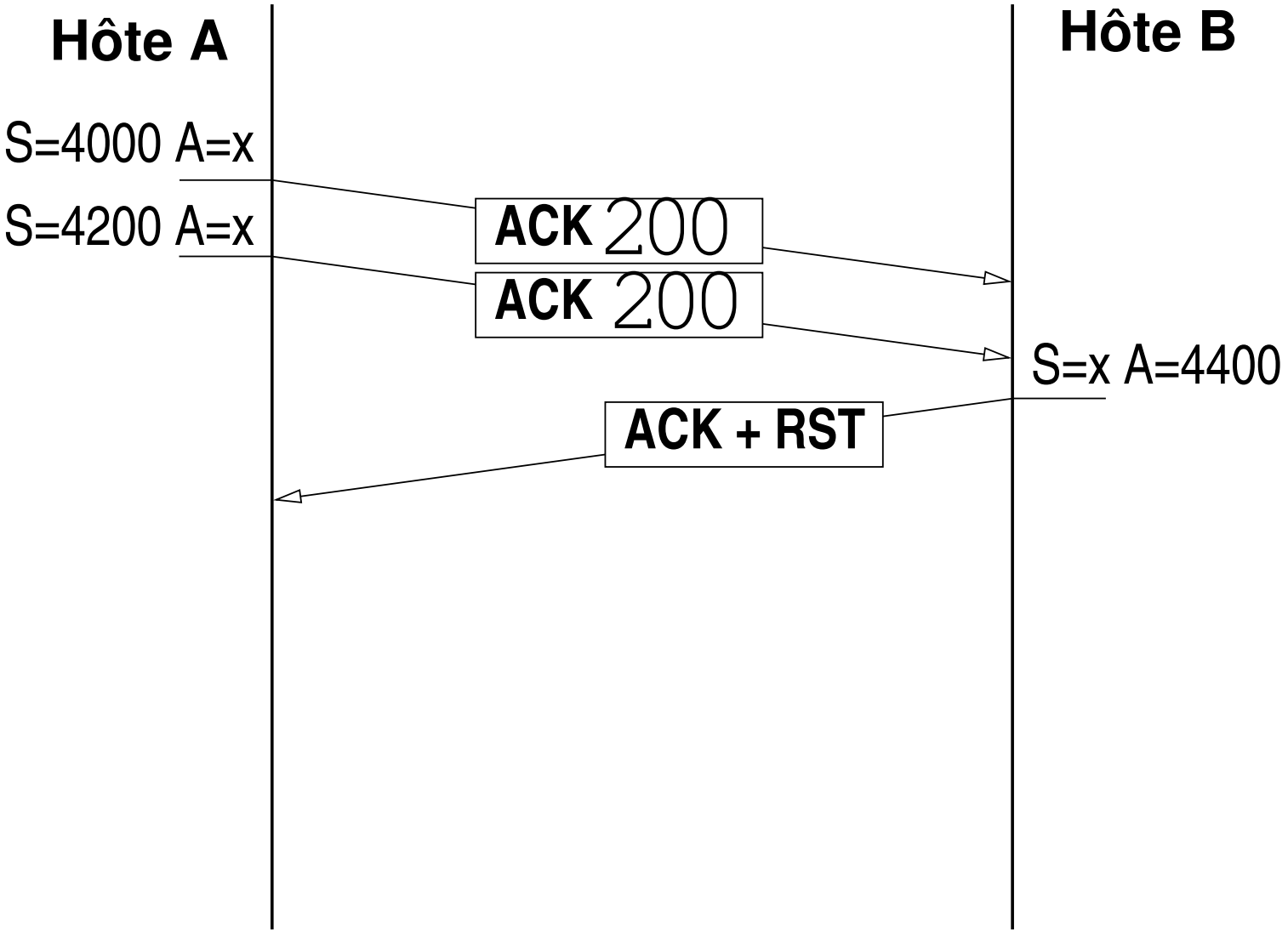
TCP : *Three-Way Handshake*



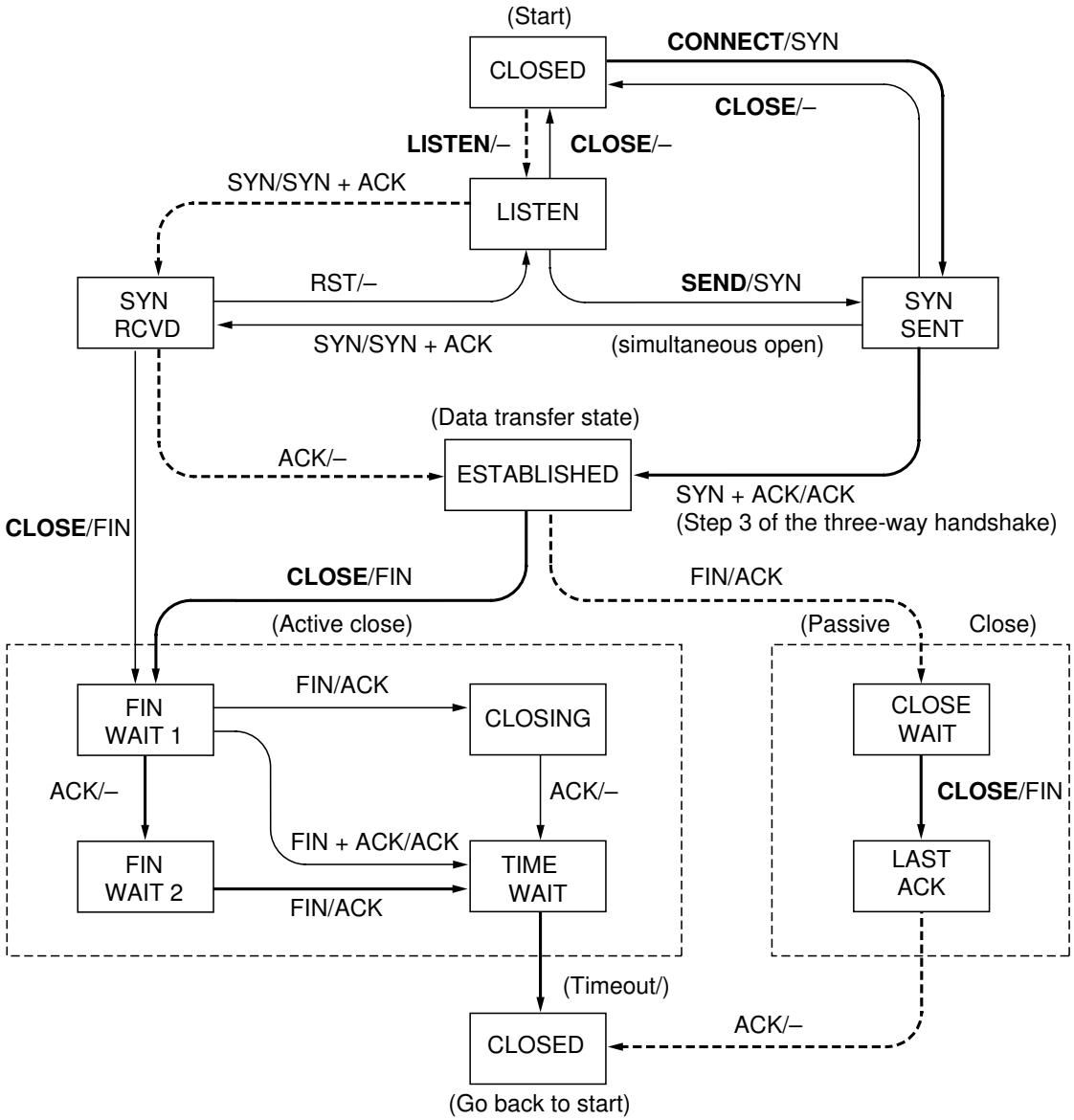
TCP : *Gracefull Release*



TCP : *Shutdow*



TCP : Automate d'états finis



Plan

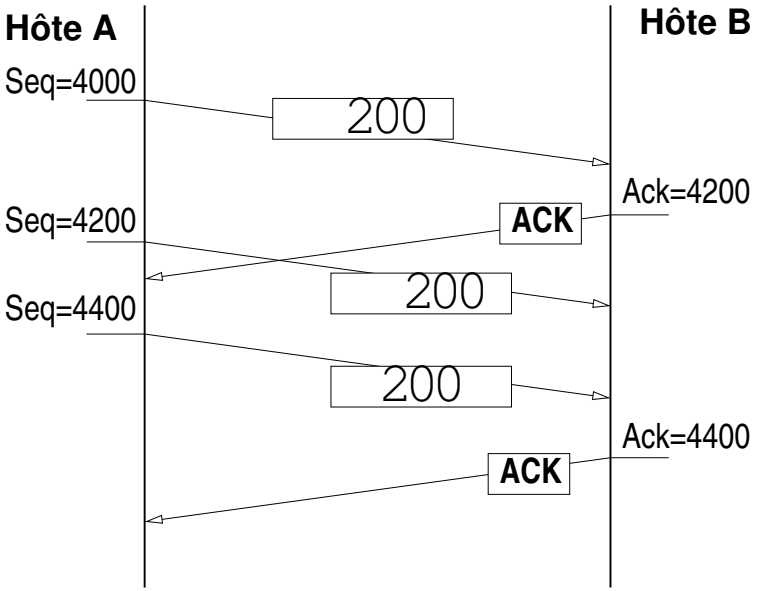
Rappels sur la couche transport

UDP : un protocole en mode non connecté

TCP : un protocole en mode orienté connexion

- format du segment TCP
- gestion de la connexion
- **gestion de la fiabilité**
- contrôle de flux
- contrôle de congestion
- implémentation : voyage au Nevada
- utilisation de TCP

Politique d'acquittement



Mécanismes de fiabilisation de TCP :

- **ARQ** (*Automatic Repeat reQuest*)
  - ✓ **acquittement** positif (ACK)
  - ✓ **retransmissions** temporisées...
- fenêtre d'anticipation
  - ✓ protocole *pipeline*
    - ☞ numéros de séquence
    - ☞ acquitements cumulatifs
    - ☞ retransmissions *Go-Back-N*

➠ nécessite une évaluation des :

- RTT (*Round Trip Time*)
- RTO (*Retransmission TimeOut*)

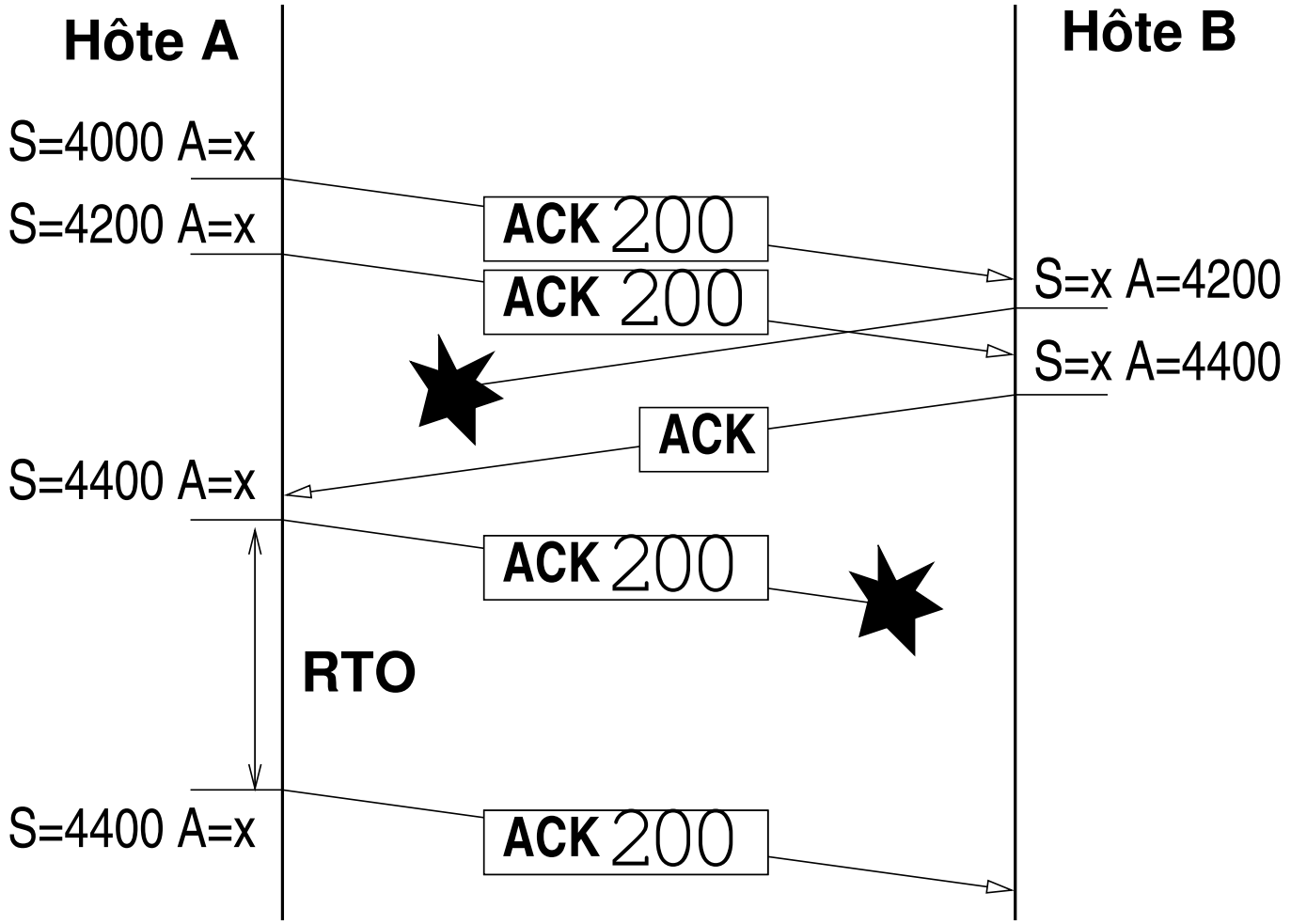
TCP : Temporisations

Gestion de multiples temporisations (*timers*) :

- *retransmission timer* (détecte les pertes)
  - ✓ $RTO = RTT + \delta D$
    - ☞ avec $\delta = 4$ et une valeur initiale du RTT élevée (3 secondes)
  - ✓ $RTT = \alpha RTT_{mesure} + (1 - \alpha) RTT_{ancien}$
    - ☞ calcul d'une moyenne avec α usuel $= 1/8$
  - ✓ $D = \beta(|RTT_{mesure} - RTT_{ancien}|) + (1 - \beta) D_{ancien}$
    - ☞ calcul de l'écart moyen avec β usuel $= 1/4$
  - ✓ **algorithme de Karn**
    - ☞ ne pas tenir compte des paquets retransmis et doubler le RTO à chaque échec (*exponential backoff*)
- *persistence timer* (évite les blocages)
  - ✓ envoi d'un acquittement avec une fenêtre à 0
- *keep alive timer* (vérifie s'il y a toujours un destinataire)
- *closing timer* (terminaison)

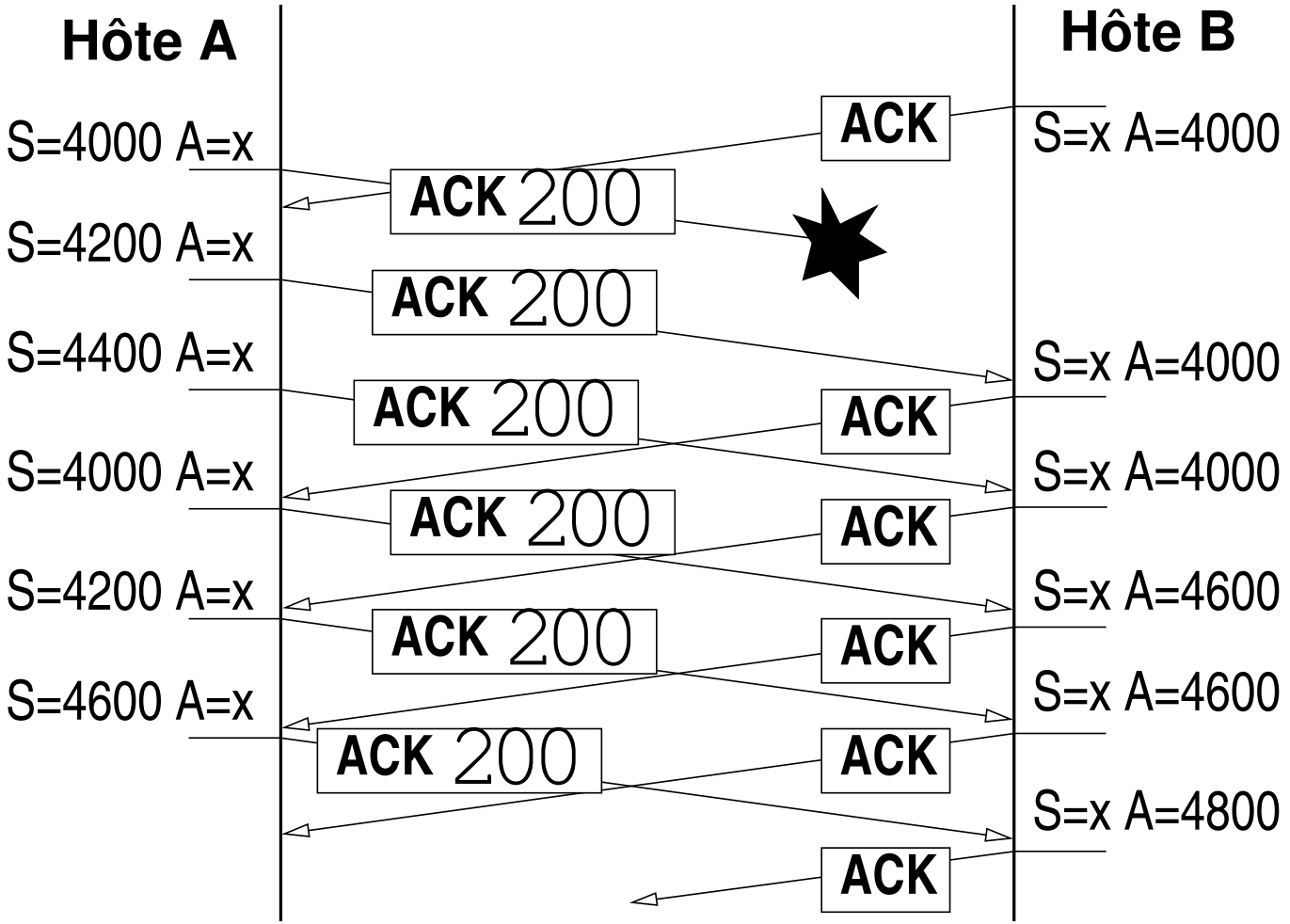
Politique de retransmission

Acquittements cumulatifs, temporisation (RTO) et retransmission :



Politique de retransmission

Acquittements cumulatifs et politique *Go-back-N* :



Plan

Rappels sur la couche transport

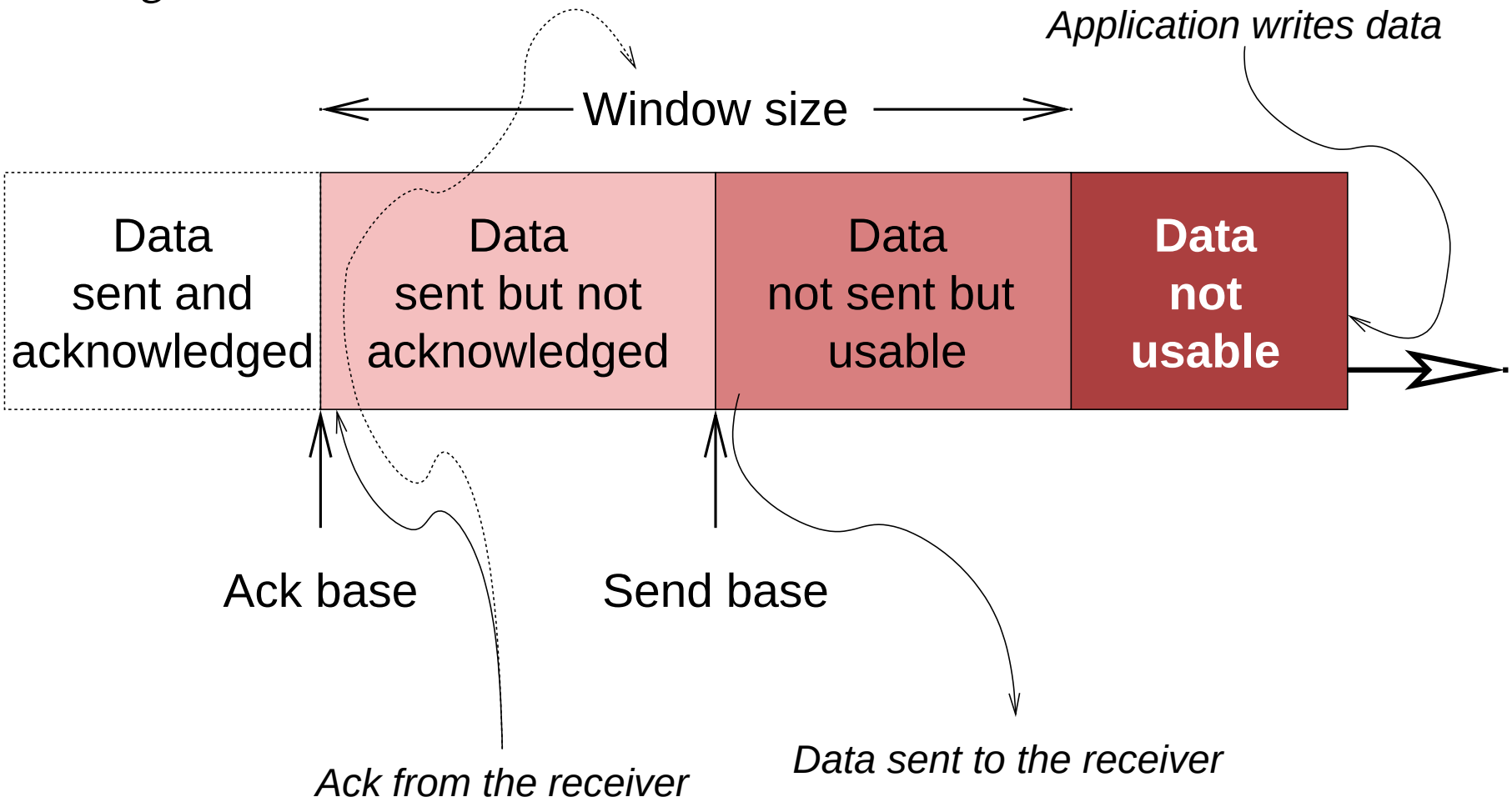
UDP : un protocole en mode non connecté

TCP : un protocole en mode orienté connexion

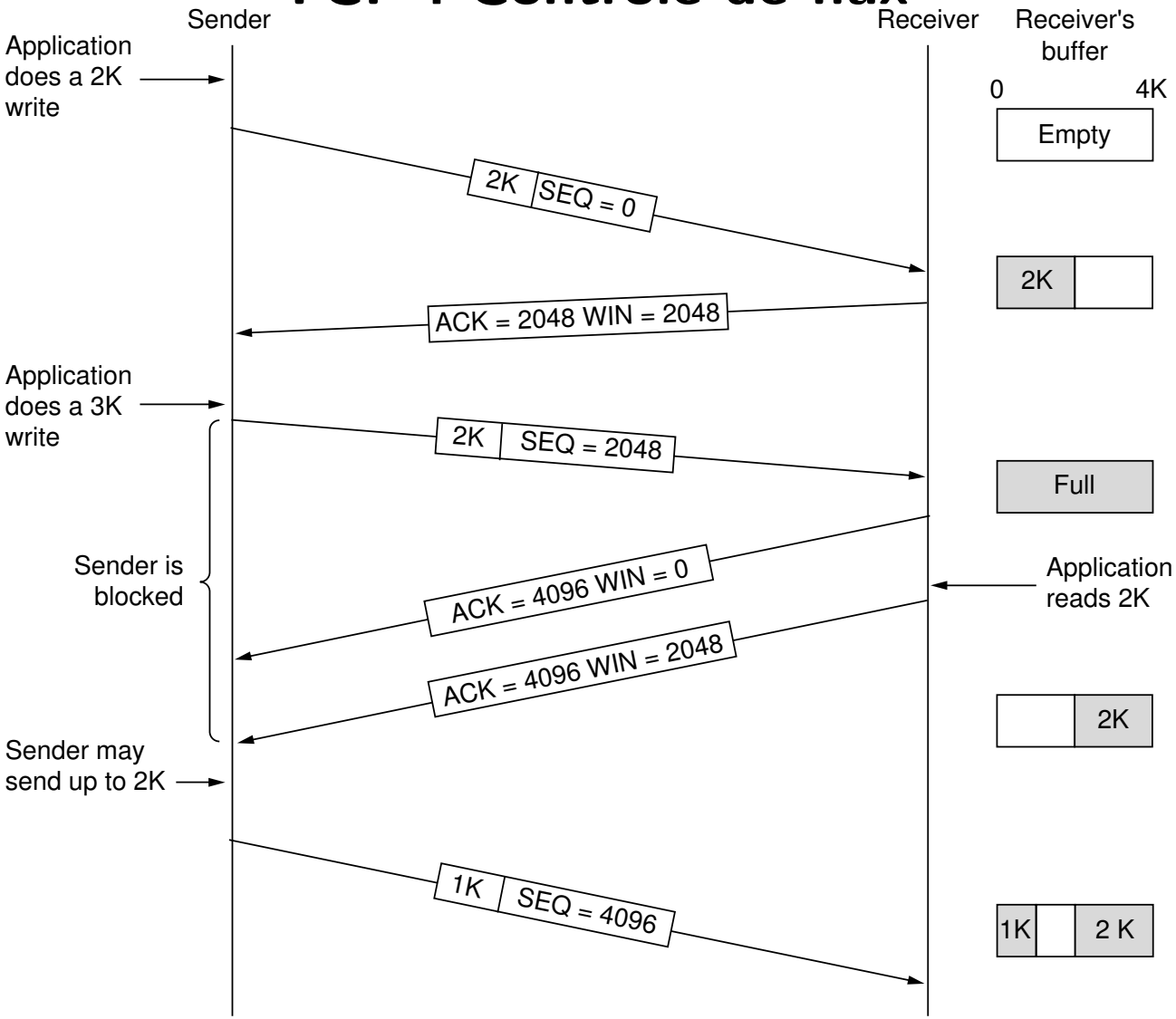
- format du segment TCP
- gestion de la connexion
- gestion de la fiabilité
- **contrôle de flux**
- contrôle de congestion
- implémentation : voyage au Nevada
- utilisation de TCP

TCP : 1ère fenêtre coulissante

Sliding window :



TCP : Contrôle de flux



pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

TCP : un protocole en mode orienté connexion

- format du segment TCP
- gestion de la connexion
- gestion de la fiabilité
- contrôle de flux
- **contrôle de congestion**
- implémentation : voyage au Nevada
- utilisation de TCP

TCP : Pertes

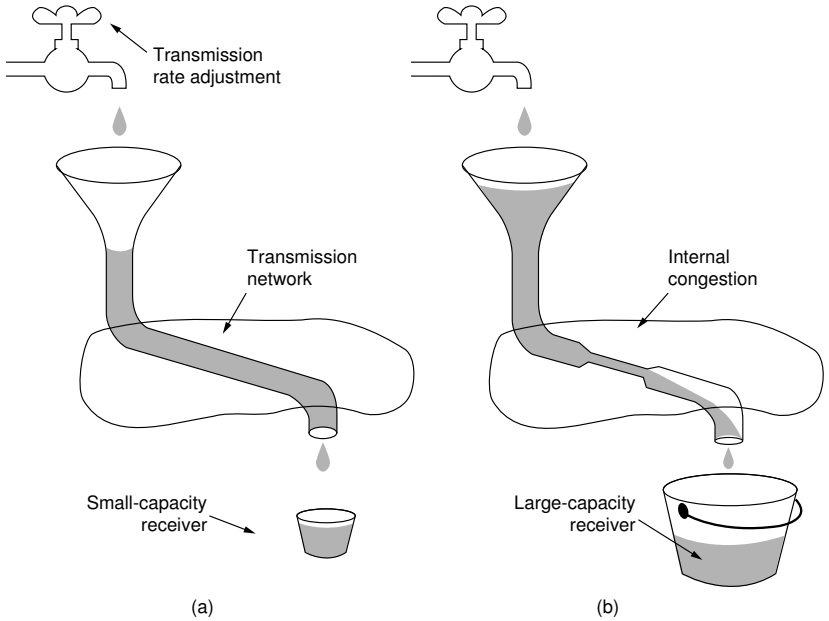
L'absence d'un paquet au temps voulu :

- principalement lié à la *congestion*
- rarement à une erreur de transmission

➠ Comment évaluer une perte ?

➠ Quelle politique de retransmission ?

TCP : Contrôle de congestion (1)



pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Algorithmes de contrôle :

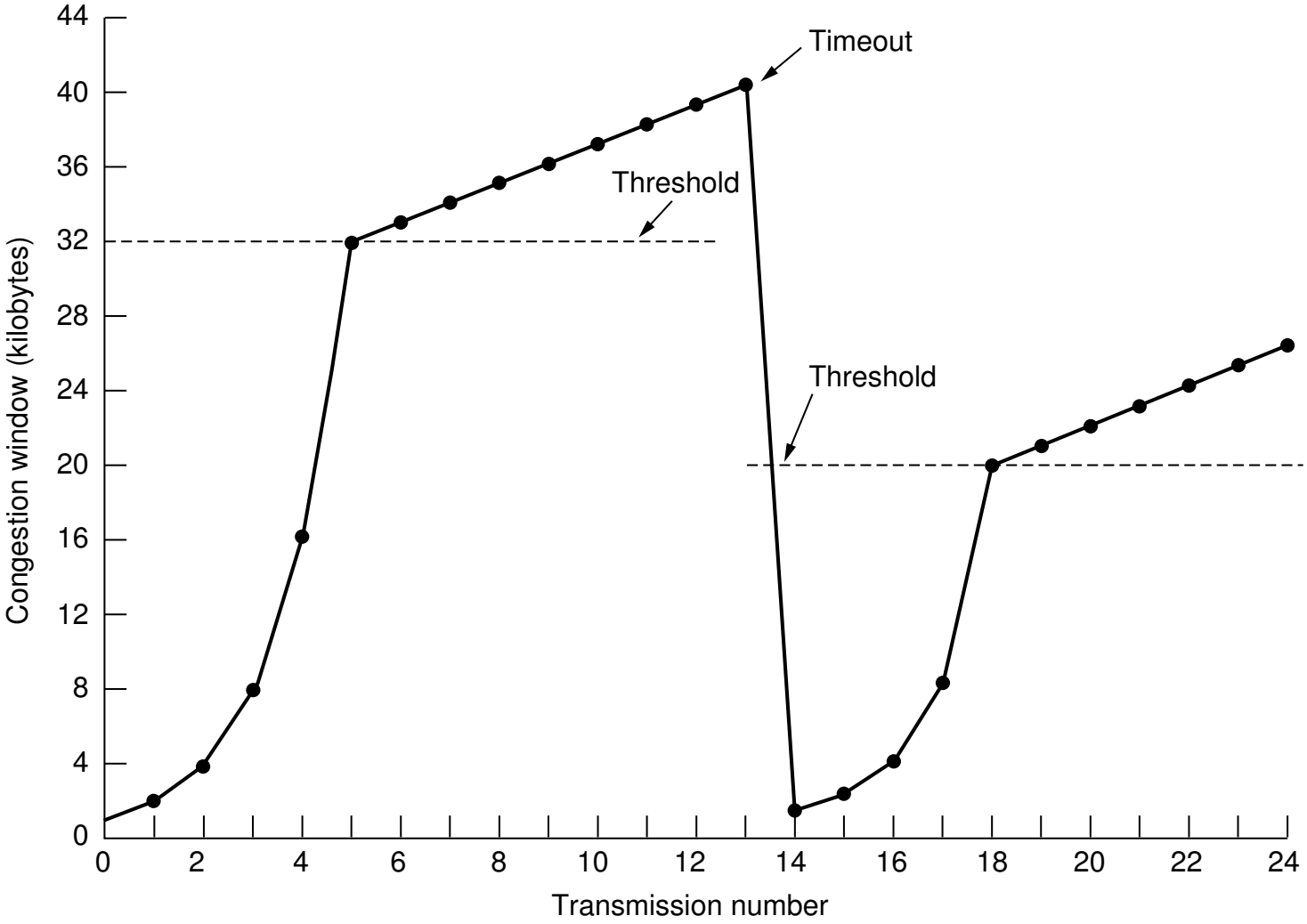
- superposition d'une seconde fenêtre coulissante ($cwin$)
- algorithmes de contrôle du débit :

✓ **slow start** (*exponential increase*)
- ☞ $cwin \mathrel{+}= MSS$ pour chaque **nouvel** acquittement
- ☞ de $cwin = 1MSS$ jusqu'au *ss_threshold*

✓ **congestion avoidance** (*additive increase*)
- ☞ $cwin \mathrel{+}= MSS$ pour chaque RTT
- ☞ de $ss\_threshold$ jusqu'à l'expiration de RTO

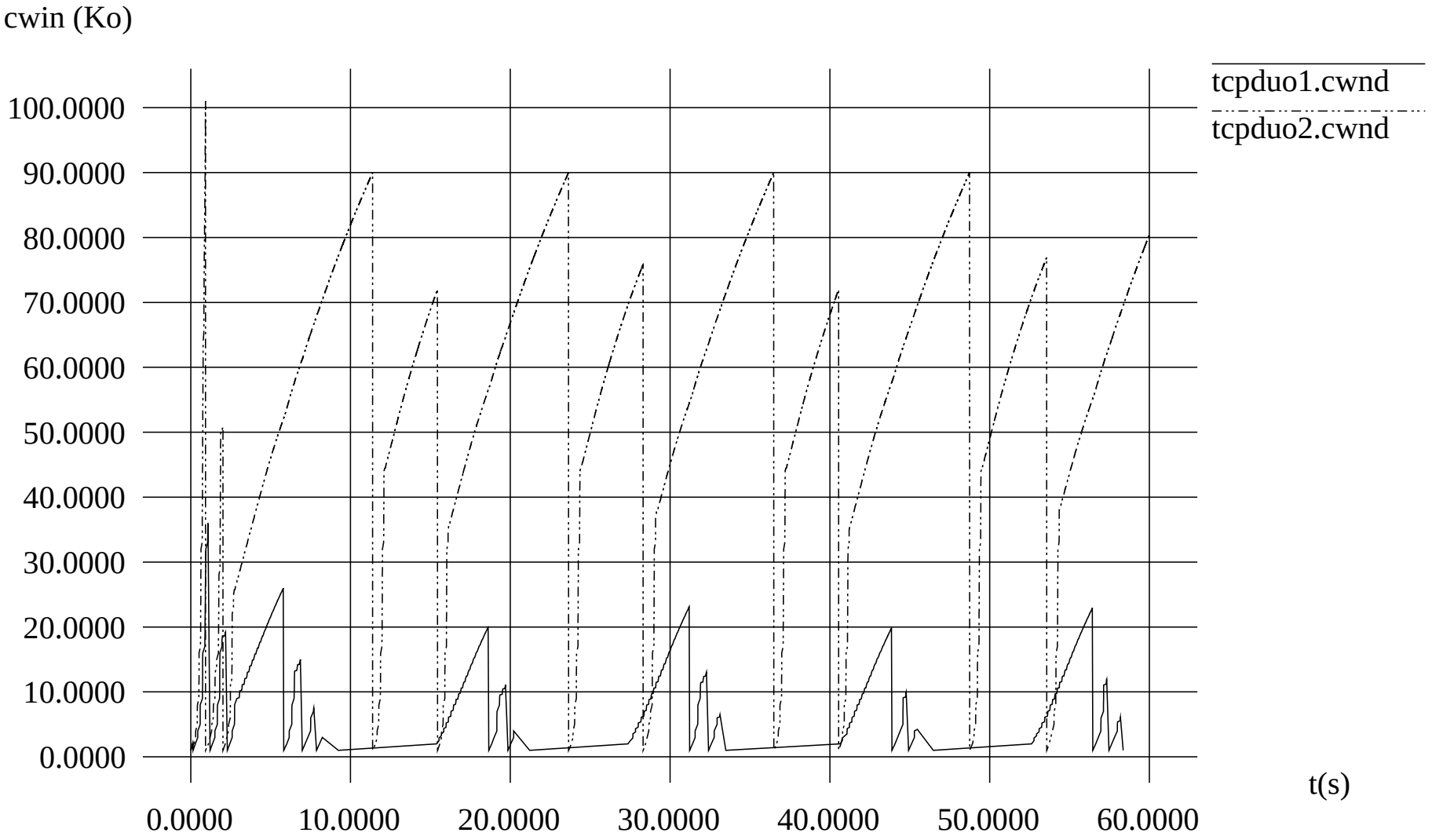
✓ **multiplicative decrease**
- ☞ $ss\_threshold = cwin_{RTOexp}/2$
- ☞ $cwin = 1MSS$
  - ➠ *slow start*

TCP : Contrôle de congestion (2)



pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

TCP : équité entre flots ?



- oscillation de deux flots en phase de congestion

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

TCP : un protocole en mode orienté connexion

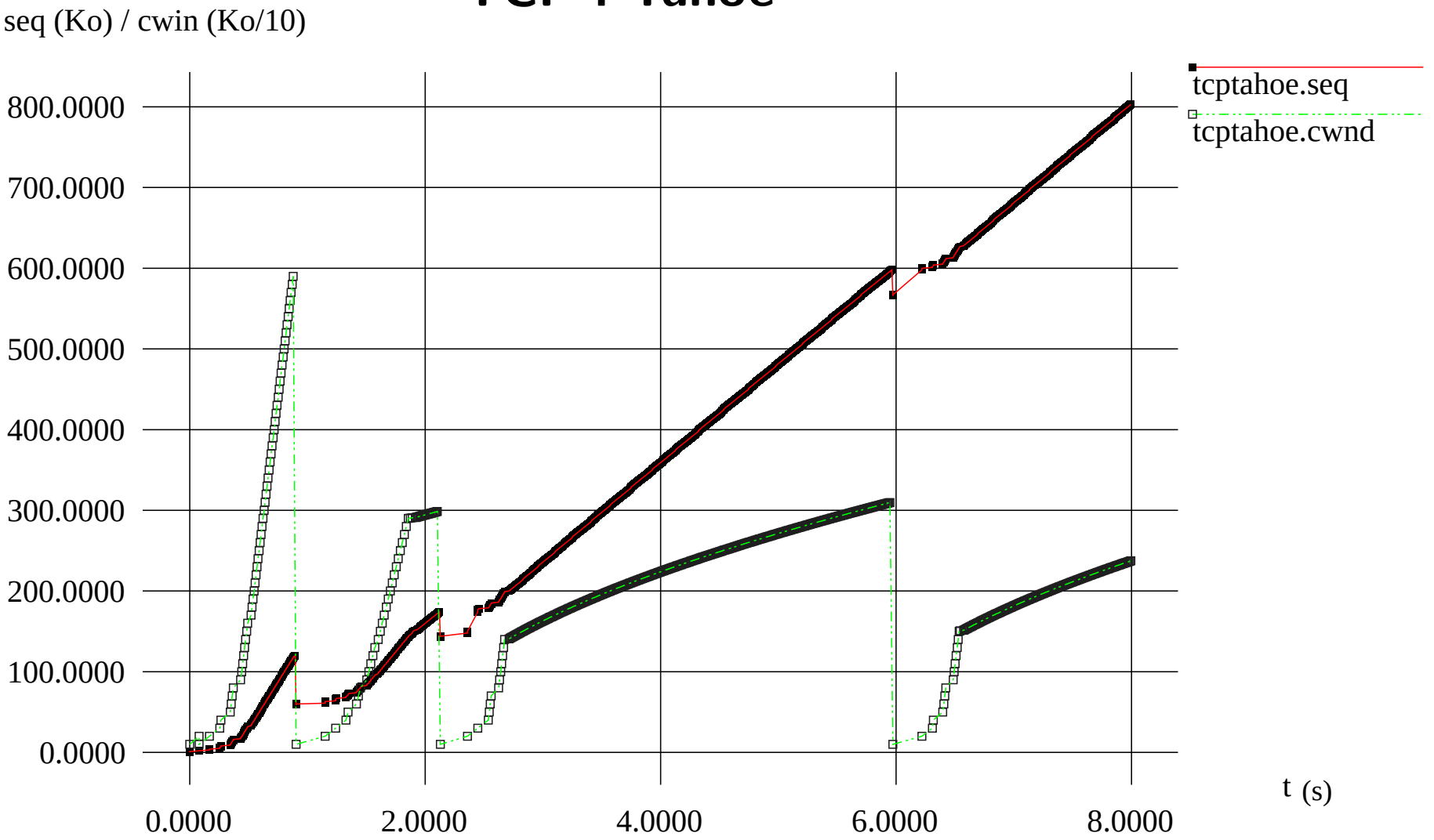
- format du segment TCP
- gestion de la connexion
- gestion de la fiabilité
- contrôle de flux
- contrôle de congestion
- **implémentation : voyage au Nevada**
- utilisation de TCP

Implémentation

A trip to Nevada :

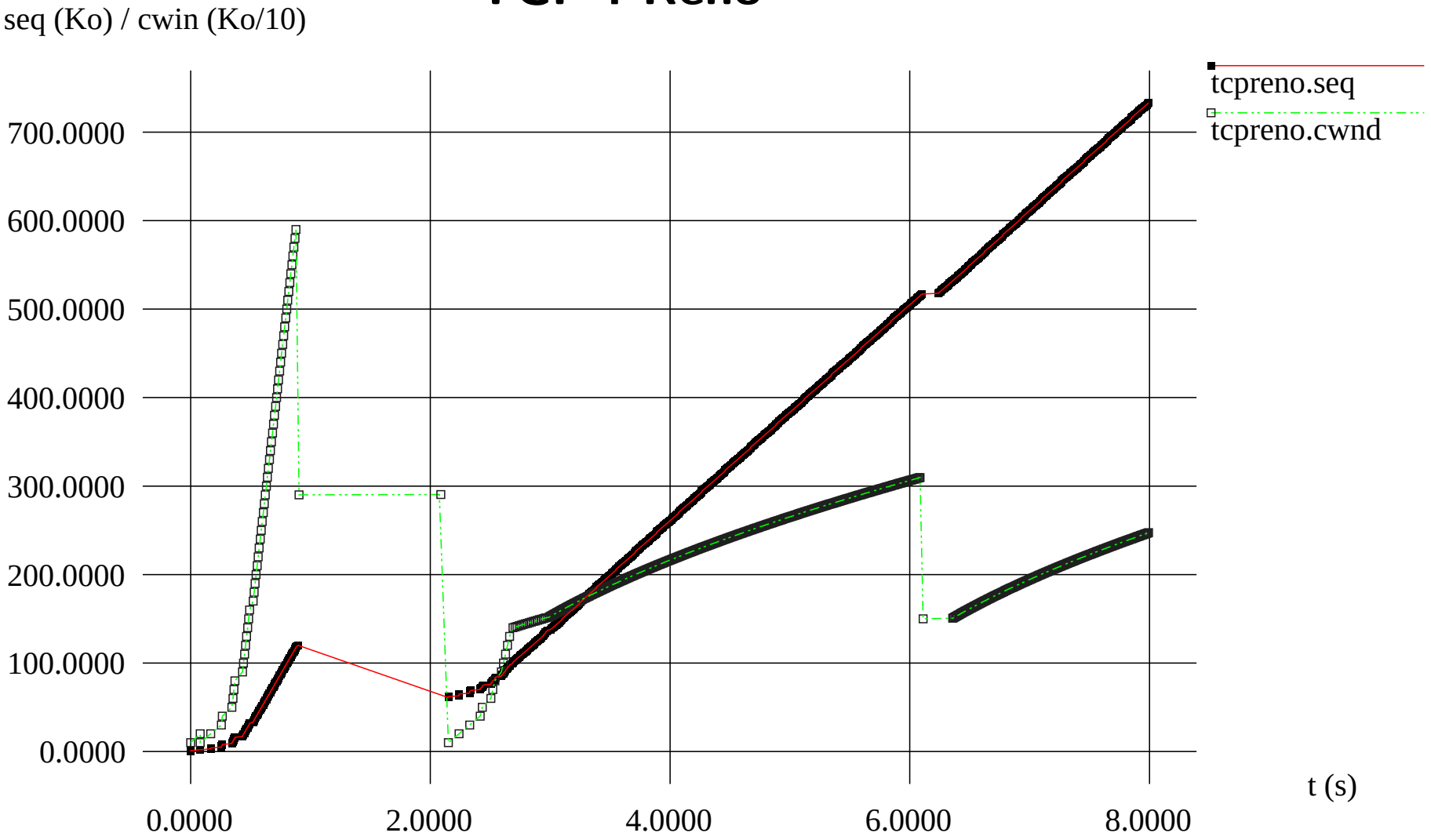
- **TCP Tahoe** 1988
  - ✓ *slow start* + *congestion avoidance* + *multiplicative decrease*
  - ✓ + **fast retransmit** (déclenche la retransmission d'un segment après trois acquittements dupliqués, avant le *timeout*)
- **TCP Reno** 1990 (RFC 2581)
  - ✓ idem TCP Tahoe
  - ✓ + **fast recovery** (pas de *slow start* après un *fast retransmit*)
- **TCP newReno** 1996 (RFC 2582)
  - ✓ idem TCP Reno
  - ✓ + pas de *slow start* à la première congestion et ajustement de $cwin$
  - ✓ + SACK (RFC 2018)
- **TCP Vegas...**
  - ✓ évite la congestion en **anticipant** les pertes
  - ✓ réduction du débit en fonction des variations du RTT

TCP : Tahoe



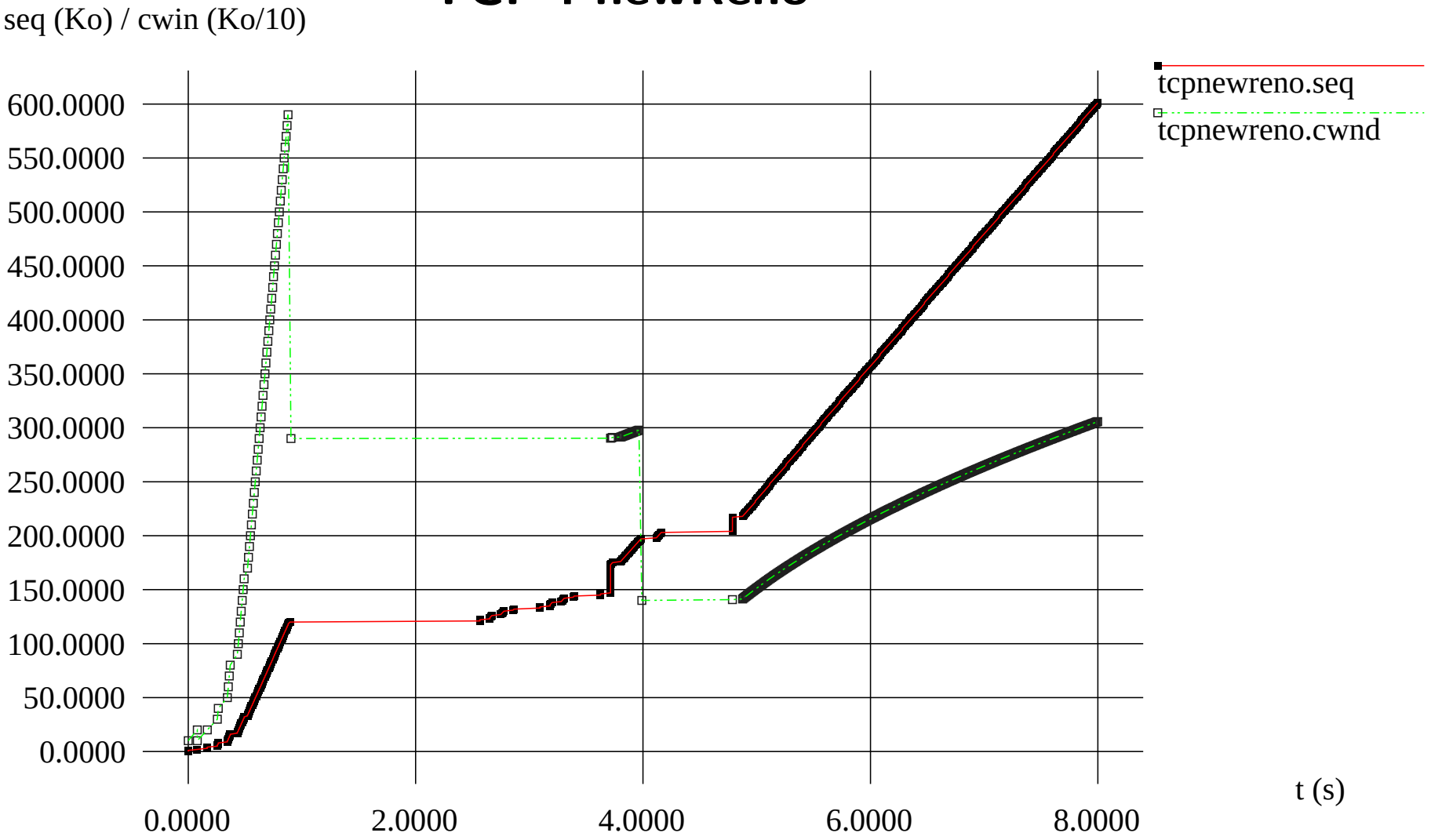
- *slow start + congestion avoidance + multiplicative decrease*
- + **fast retransmit** : vers quel débit converge-t-on ?

TCP : Reno



- **fast recovery** (pas de *slow start* après un *fast retransmit*)
- meilleurs résultats qu'avec Tahoe ?

TCP : newReno



- pas de *slow start* à la première congestion et ajustement de $cwin$
- meilleurs résultats qu'avec Reno ?

Plan

Rappels sur la couche transport

UDP : un protocole en mode non connecté

TCP : un protocole en mode orienté connexion

- format du segment TCP
- gestion de la connexion
- gestion de la fiabilité
- contrôle de flux
- contrôle de congestion
- implémentation : voyage au Nevada
- **utilisation de TCP**

TCP : exemples d'applications

Les applications suivantes reposent typiquement sur TCP :

- connexion à distance (TELNET, rlogin et ssh)
- transfert de fichiers (FTP, rcp, scp et sftp)
- protocole de routage externe (BGP)
- messageries instantanées (IRC, ICQ, AIM...)
- ...

TCP : utilisation spécifiques

TCP doit s'adapter à des flots de qqs **bps** à plusieurs **Gbps** :

- LFN (*Long Fat Network*)
  - ✓ capacité du réseau = **bande passante * délai de propagation**
    - ☞ limitation de taille de la fenêtre (option WSIZE, jusqu'à un facteur 2^{14})
    - ☞ rebouclage des numéros de séquence (PAWS, *Protect Against Wrapped Sequence*, utilise l'option TIMESTAMP)
    - ☞ acquittements sélectifs pour éviter des retransmissions importantes inutiles (option SACK)
  - ✓ satellites
  - ✓ fibres transatlantiques
- réseaux asymétriques (ADSL, Cable)
  - ✓ sous-utilisation du lien rapide

TCP : Interface socket

```
#include <sys/types.h>
#include <sys/socket.h>

# create a descriptor and bind local IP and port
int socket(int domain, int type, int protocol);
#   domain : PF_INET for IPv4 Internet Protocols
#     type : SOCK_STREAM Provides sequenced, reliable, 2-way, connection-based byte streams.
#              An out-of-band data transmission mechanism may be supported.
# protocol : TCP (/etc/protocols)
int bind(int s, struct sockaddr *my_addr, socklen_t addrlen);

# Server : passive queuing mode and connection acceptance
int listen(int s, int backlog);
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);

# Client : active connection
int  connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);

# Send and receive data
int send(int s, const void *msg, size_t len, int flags);
int recv(int s, void *buf, size_t len, int flags);

# End : dealocate
int close(int s);
```

Fin

Document réalisé avec LaTeX.
Généré le 14 octobre 2004.
Classe de document `foils`.
Dessins réalisés avec `xfig`.

Olivier Fourmaux, `olivier.fourmaux@lip6.fr`
`http://www-rp.lip6.fr/~fourmaux`

Ce document est disponible en postscript compressé avec gzip à
`http://www-rp.lip6.fr/~fourmaux/res/res4c4c-.pdf`