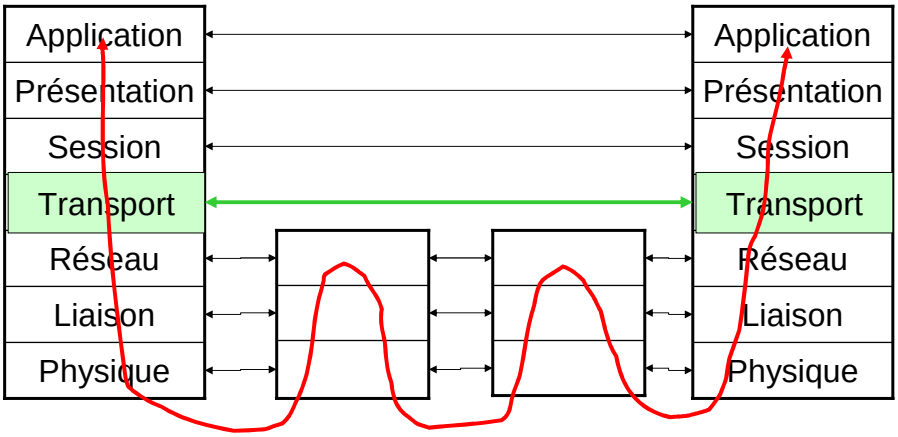
Couche Transport
TCP et UDP

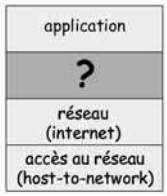
Le Modèle OSI



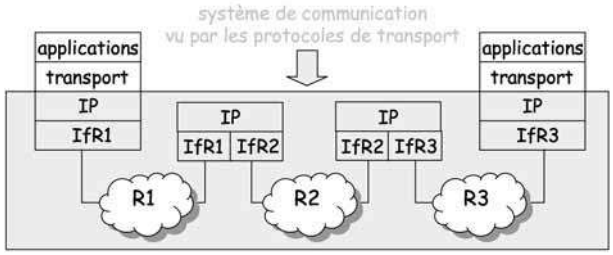
Mode de transport des données

Problématique ?

- Réseau :
  - service de remise de paquets de machine à machine
- Comment passer de ce service à un canal de communication de processus à processus ?



- La couche transport permet de faire communiquer directement deux processus (applications, programmes) qui s'exécutent sur deux machines quelconques du réseau

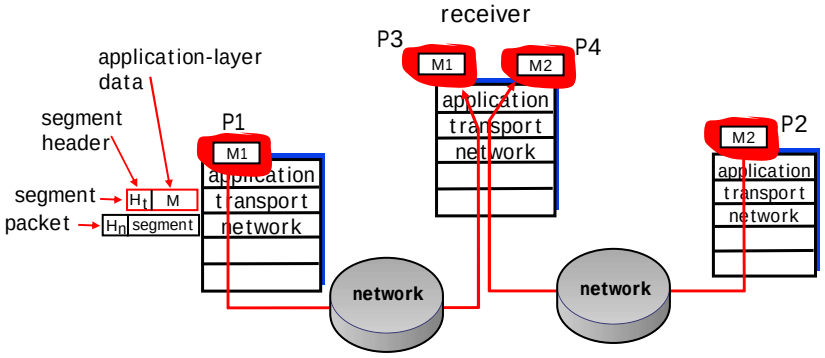


Services offerts par la couche Transport

- Créer des connexions logiques entre applications s'exécutant sur des hôtes distants
- Fragmentation et réassemblage des messages
- Unité de données de protocole (PDU) appelé segment (TCP) ou datagramme (UDP)
- Protocoles de la couche Transport s'exécutent uniquement aux extrémités.

Services offerts par la couche Transport

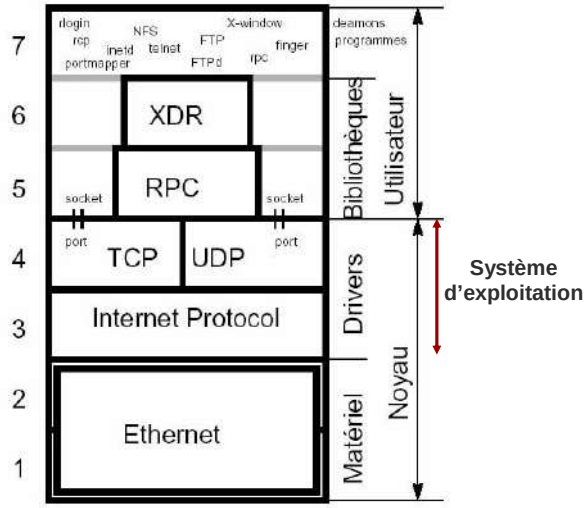
- Multiplexage des données issues de différentes applications sur une connexion de niveau réseau avec démultiplexage au moyen de paramètres dans l'en-tête des segments (n° de ports)
- Service Transport VS Service Réseau
  - Couche réseau: transfert de données entre machines
  - Couche transport : transfert de données entre processus applicatifs (Pi)



Internet - Couche Transport -

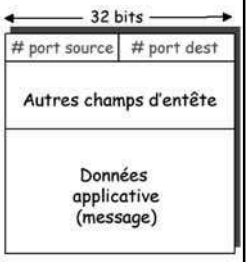
1. Mode connecté : **TCP** (Transport Control Protocol)
   - Transport fiable
   - Phase de connexion / de transfert des données / déconnexion
   - Contrôle de flux
   - Contrôle des erreurs + acquittements + retransmission
   - Garantie du séquencement
   - Segmentation des messages
   - Usage : Applications critiques
2. Mode non connecté : **UDP** (User Datagram Protocol)
   - Transport non fiable
   - Overhead réduit par rapport à TCP (8 octets / 20 octets)
   - Envoi direct des informations
   - Support du multi-point
   - Pas de garantie de séquencement
   - Contrôle des erreurs
   - Usage : Applications multimédias

Architecture d'un terminal IP



UDP / TCP - Identification des services -

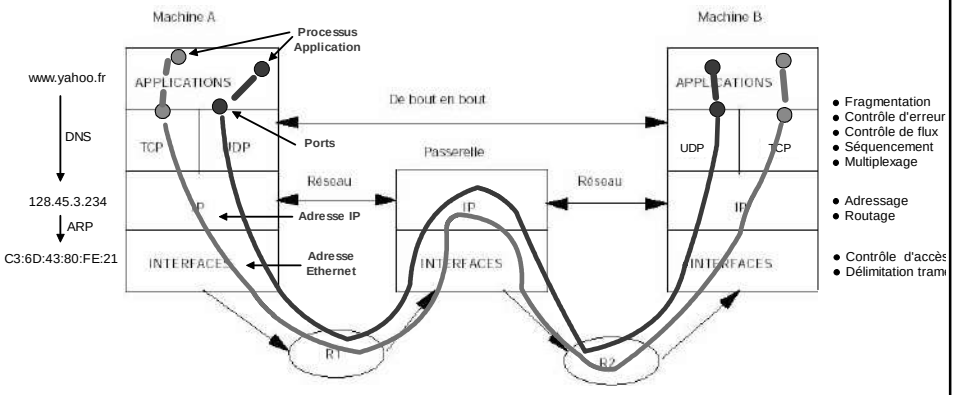
- les adresses IP désignent les machines entre lesquelles les communications sont établies. Lorsqu'un processus désire entrer en communication avec un autre processus, il doit adresser le processus s'exécutant sur cette machine.
- L'adressage de ce processus est effectué selon un concept abstrait, les ports, indépendant du système d'exploitation des machines car :
  - les processus sont créés et détruits dynamiquement sur les machines,
- L'émission d'un message se fait sur la base d'un port source et d'un port destinataire.
- Les processus disposent d'une interface système leur permettant de spécifier un port ou d'y accéder (socket).
- Les accès aux ports sont généralement synchrones, les opérations sur les ports sont tamponnés (files d'attente).



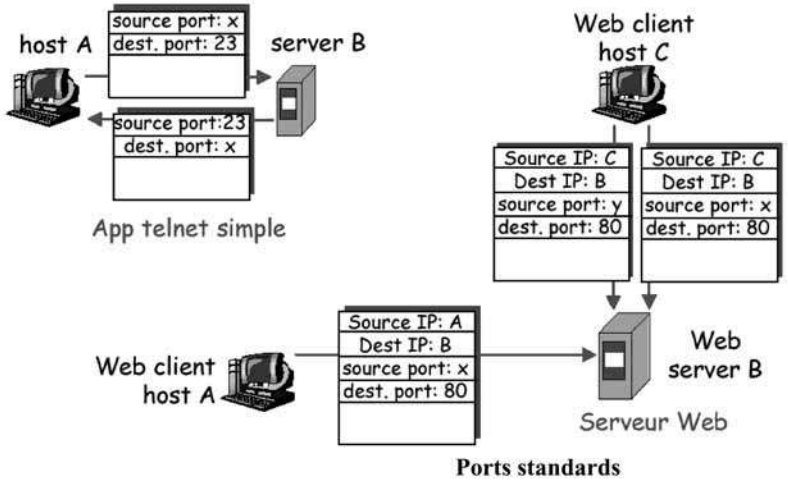
format de segment TCP/UDP

Ports

Communication client/serveur



UDP / TCP – les ports



Ports

UDP / TCP: les ports
RFC 1700

- 3 catégories
  - ports well-known : de 0 à 1023
    - alloués par l'IANA (Internet Assigned Numbers Authority)
    - sur la plupart des systèmes, ne peuvent être utilisés que par des processus système (ou root) ou des programmes exécutés par des utilisateurs privilégiés
  - ports registered : de 1024 à 49 151
    - listés par l'IANA
    - sur la plupart des systèmes, peuvent être utilisés par des processus utilisateur ordinaires ou des programmes exécutés par des utilisateurs ordinaires
  - ports dynamic/private : de 49 152 à 65 535
    - alloués dynamiquement

Well-known ports

UDP : les ports standards

- Certains ports sont réservés (*well-kown port assignements*) :

No port	Mot-clé	Description
7	**ECHO**	**Echo**
11	**USERS**	**Active Users**
13	**DAYTIME**	**Daytime**
37	**TIME**	**Time**
42	**NAMESERVER**	**Host Name Server**
53	**DOMAIN**	**Domain Name Server**
67	**BOOTPS**	**Boot protocol server**
68	**BOOTPC**	**Boot protocol client**
69	**TFTP**	**Trivial File transfert protocol**
123	**NTP**	**Network Time Protocol**
161	**SNMP**	**Simple Network Management prot**.

- D'autres numéros de port (non réservés) peuvent être assignés dynamiquement aux applications.

Notion de multiplexage

TCP : ports standards

No port	Mot-clé	Description
20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
37	TIME	Time
42	NAMESERVER	Host Name Server
43	NICNAME	Who Is
53	DOMAIN	Domain Name Server
79	FINGER	Finger
80	HTTP	WWW
110	POP3	Post Office Protocol - Version 3
111	SUNRPC	SUN Remote Procedure Call

UDP : User Datagram Protocol

UDP: User Datagram Protocol [RFC 768]

- Protocole de transport le plus simple
- Service "au mieux", les segments UDP peuvent être:
  - Perdus
  - Délivrés dans le désordre
- *Sans connexion:*
  - Sans handshaking entre l'émetteur et le récepteur
  - Chaque segment UDP est traité indépendamment des autres

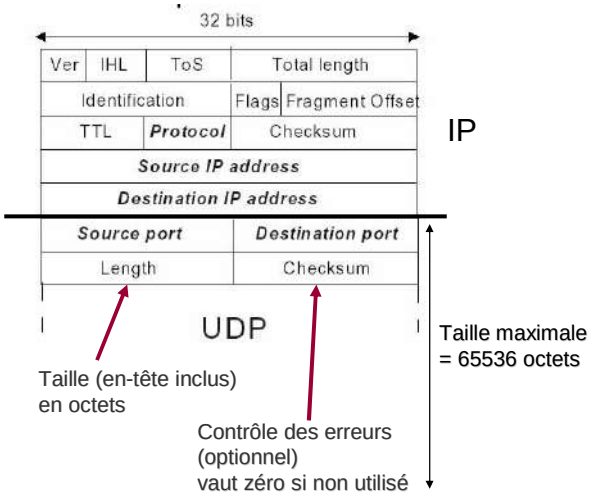
Pourquoi UDP?

- sans délai de connexion
- simple: sans nécessité d'un état dans l'émetteur et le récepteur
- petit entête
- sans contrôle de congestion: UDP peut émettre aussi rapidement qu'il le souhaite

UDP

UDP : format du datagramme

- Les messages UDP sont appelés datagrammes UDP.



Datagramme UDP

UDP : Multiplexage

- UDP multiplexe et démultiplexe les datagrammes en sélectionnant les numéros de ports :
  - Le **Port source** indique la session créatrice du paquet (une application obtient un numéro de port de la machine locale; dès lors que l'application émet un message via ce port ➔ le champ PORT SOURCE du datagramme UDP contient ce numéro de port),
  - une application connaît (ou obtient) un numéro de **Port distant** afin de communiquer avec le service désiré.
- Lorsque UDP reçoit un datagramme, il vérifie que celui-ci est un des ports actuellement actifs (associé à une application) et le délivre à l'application responsable (mise en queue)
- si ce n'est pas le cas, il émet un message ICMP *port unreachable*, et détruit le datagramme.

Inconvénients de TCP

Inconvénients de UDP

- Manque de fiabilité (pas nécessaire pour les applications de vidéo diffusion ou de VoIP)
- Aucune gestion de la congestion (les routeurs sont la seul solution pour réduire cet effet)
- UDP représente un pourcentage très réduit du trafic qui traverse le réseau
  - Quelques applications utilisant UDP: DNS, SNMP, DHCP et RIP.

Transmission Control Protocol

- RFCs : 793, 1122, 1323, 2018, 2581
- point-à-point:
  - Un émetteur, un récepteur
- Fiable, réception dans l'ordre du flot d'octet:
  - Pas de « frontières de message »
- Fenêtre d'émission:
  - Les mécanismes de contrôle de flot et de congestion règlent la taille de la fenêtre
- Tampons de réception et d'émission

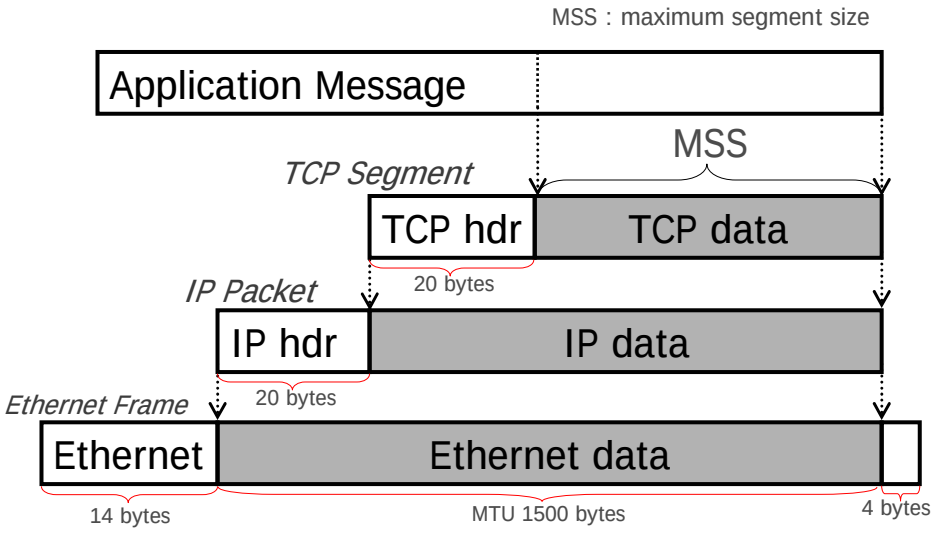
Transmission Control Protocol

- Transfert bidirectionnel des données:
  - Transfert bi-directionel de données sur une connexion
  - MSS: *maximum segment size*
- Orienté connexion:
  - handshaking (échange de msgs de control) initialise l'état de l 'émetteur et du récepteur avant l'émission de données
- Contrôle de flot :
  - L'émetteur ne submerge pas le récepteur
- Contrôle de congestion :
  - l'émetteur adapte son débit d'envoi de paquets à l'état du réseau

Transmission Control Protocol

- TCP est orienté flux d'octets
  - le processus émetteur "écrit" des octets sur la connexion TCP
  - le processus récepteur "lit" des octets sur la connexion TCP
- TCP ne transmet pas d'octets individuels
  - en émission
    - TCP "bufferise" les octets jusqu'à en avoir un nombre raisonnable
    - TCP fabrique un segment et l'envoie

TCP: Fragmentation & Encapsulation



TCP segmentation

TCP : construction du segment

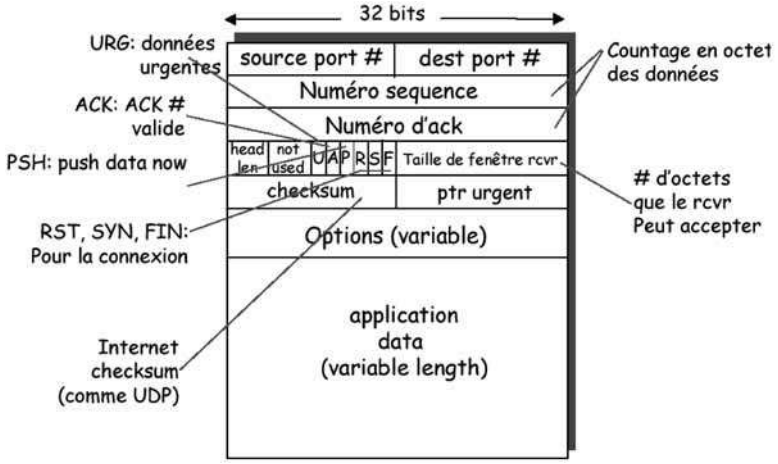
- Quand est-ce que TCP décide d'envoyer un segment ?

1. il a MSS (MaximumSize Segment) octets de données à envoyer
   - MSS = MTU - en-tête IP - en-tête TCP
2. le processus lui demande explicitement
   - fonction *push*
3. un temporisateur expire
   - pour éviter d'attendre trop longtemps MSS octets

- En pratique :
  - Le "Minimum Reassembly buffer" = 576 octets
    - 576 est la valeur recommandée pour le MTU
    - MSS = 536 octets

segment TCP

TCP : format du segment



segment TCP

TCP : format du segment (suite)

- **Numéro de séquence : indique le numéro de séquence du premier octet (NS) de ce segment.**
  - **Généralement à la suite d'octets O1, O2, ..., On (représentant les données du message à envoyer) est associée la suite de numéro de séquence NS, NS+1, ..., NS+n.**

Il existe deux exceptions à cette règle :

- **lorsque le bit SYN (voir CODE BITS) est positionné, le NS représente cette donnée de contrôle et par conséquent la suite NS, NS+1, NS+2, ..., NS+n+1, associe la suite de données SYN, O1, O2, ..., On.**
- **lorsque le bit FIN (voir CODE BITS) est positionné, le NS+n représente cette donnée de contrôle et par conséquent la suite NS, NS+1, NS+2, ..., NS+n, associe la suite de données O1, O2, ..., On, FIN.**

- **Numéro d'acquittement : le prochain numéro de séquence NS attendu par l'émetteur de cet acquittement. Acquitte implicitement les octets NS-1, NS-2, etc.**
- **Fenêtre: la quantité de données que l'émetteur de ce segment est capable de recevoir; ceci est mentionné dans chaque segment (données ou acquittement).**

Format du segment TCP (suite)

TCP : Format du segment

- **CODE BITS : indique la nature du segment :**
  - **URG : le pointeur de données urgentes est valide (exemple : interrupt en remote login), les données sont émises sans délai, les données reçues sont remises sans délai.**
  - **SYN : utilisé à l'initialisation de la connexion pour indiquer où la numérotation séquentielle commence. SYN occupe lui-même un numéro de séquence bien que ne figurant pas dans le champ de données. Le Numéro de séquence inscrit dans le datagramme (correspondant à SYN) est alors un *Initial Sequence Number* (ISN) produit par un générateur garantissant l'unicité de l'ISN sur le réseau (indispensable pour identifier les duplications).**
  - **FIN : utilisé lors de la libération de la connexion;**
  - **PSH : fonction « push ».**
    - **Normalement, en émission, TCP reçoit les données depuis l'Application, les transforme en segments à sa guise puis transfère les segments sur le réseau;**
    - **un récepteur TCP décodant le bit PSH, transmet à l'application réceptrice, les données correspondantes sans attendre plus de données de l'émetteur.**
    - **Exemple : émulation terminal, pour envoyer chaque caractère entré au clavier (mode caractère asynchrone).**
  - **RST : utilisé par une extrémité pour indiquer à l'autre extrémité qu'elle doit réinitialiser la connexion. Ceci est utilisé lorsque les extrémités sont désynchronisées.**

Format du segment TCP (suite)

TCP format du segment

- **CHECKSUM : calcul du champ de contrôle (CRC: Code cyclique de vérification) et s'applique à la totalité du segment obtenu (Entête et données)**
- **Champ OPTIONS**
  - **Permet de négocier la taille maximale des segments échangés. Cette option n'est présente que dans les segments d'initialisation de connexion (avec bit SYN).**
  - **TCP calcule une taille maximale de segment de manière à ce que le datagramme IP résultant corresponde au MTU du réseau. La recommandation est de 536 octets.**
  - **La taille optimale du segment correspond au cas où le datagramme IP n'est pas fragmenté mais :**
    - **il n'existe pas de mécanisme pour connaître le MTU,**
    - **le routage peut entraîner des variations de MTU,**
    - **la taille optimale dépend de la taille des en-têtes (options).**

Acquittement dans TCP

TCP : La connexion

- une connexion de type circuit virtuel est établie avant que les données ne soient échangées : appel + négociation + transferts

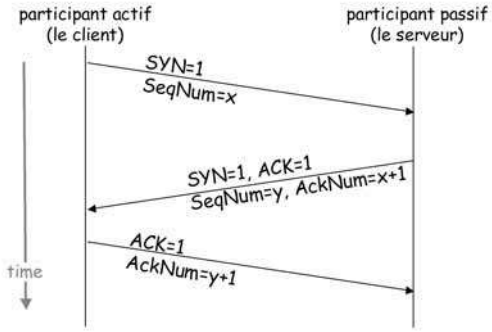
- Une connexion = une paire d'extrémités de connexion
- Une extrémité de connexion = couple (adresse IP, port)
- Exemple de connexion : ((124.32.12.1, 1034), (19.24.67.2, 21))

- Une extrémité de connexion (Serveur) peut être partagée par plusieurs autres extrémités de connexions (Clients) : multi-instanciation
- La mise en oeuvre de la connexion se fait en deux étapes :
  - une application (extrémité) effectue une ouverture passive en indiquant qu'elle accepte une connexion entrante,
  - une autre application (extrémité) effectue une ouverture active pour demander l'établissement de la connexion.

TCP connexion

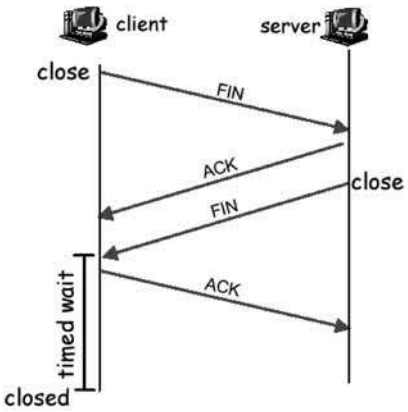
TCP : ouverture connexion

- Les 3 segments échangés



TCP opening

TCP : fermeture connexion



Le client ferme la socket:
`clientSocket.close();`

1: le client envoit un segment TCP FIN au serveur

2: le serveur reçoit le FIN, repond par un ACK. Ferme la connexion et envoi un FIN.

3: le client reçoit un FIN, et répond par un ACK.

- Passe en attente et acquitte to les FINs qu'il reçoit

4: le serveur, reçoit l' ACK. La connexion est fermé.

TCP closing

TCP : n° de séq. & Acquittement

#' Seq.:

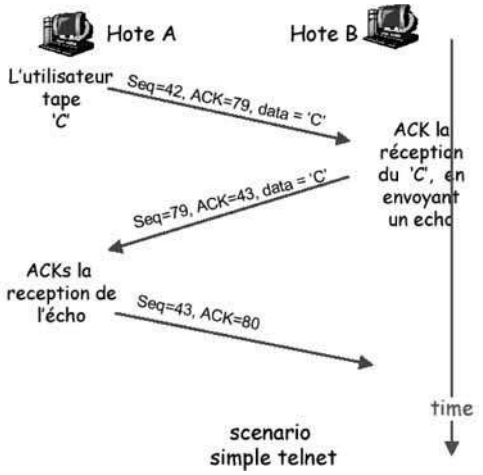
- « Numéro » du premier octet dans le segment

ACKs:

- # de seq du prochain octet attendu
- ACK cumulé

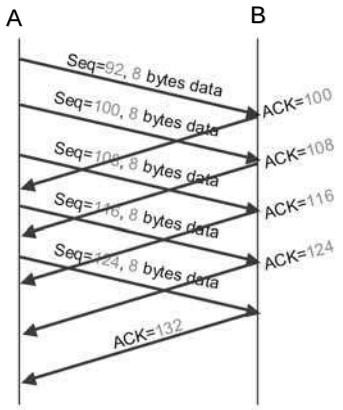
Q: Comment traiter les segments arrivé en désordre

- R: La spec TCP ne le dit pas- laissé au concepteur

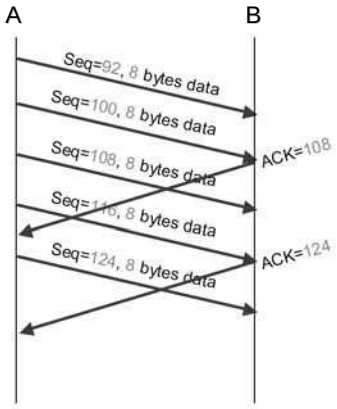


TCP acquittement

TCP : ACKs cumulés

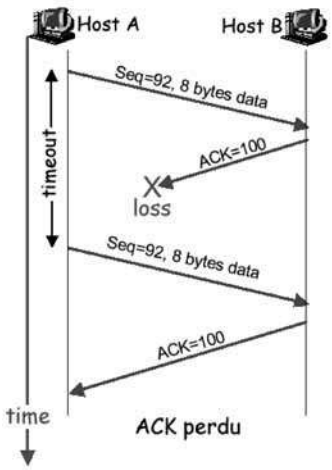


ACK cumulés

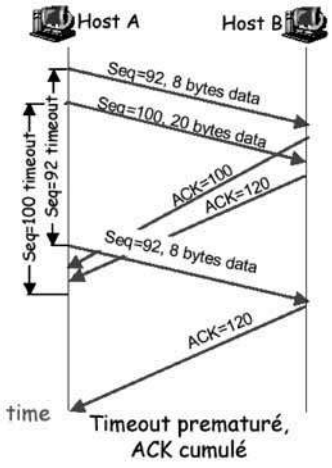


ACK cumulés
ACK retardés

TCP : problèmes avec ACKs



ACK perdu



Timeout prématuré,
ACK cumulé

TCP : envoi des ACK

Evenement	Action du récepteur
Arrivée d'un segment dans l'ordre, sans trou, Tout le reste a été ACKé	ACK mis en attente. Attendre jusqu'à 500ms pour le segment suivant. Sinon envoyer l'ACK
Arrivée d'un segment dans l'ordre, sans trou, Un ACK en attente	Envoyer immédiatement un ACK cumulé
Arrivée d'un segment dans le désordre avec un seq. # supérieur à l'attente	Envoyer un ACK dupliqué, indiquant le seq. # Du prochain octet attendu
Arrivée d'un segment qui remplit partiellement ou complètement le trou	Envoyer immédiatement un ACK si le segment rempli un trou

TCP acquittement

TCP : retransmission

Acquittements et retransmissions

- Le mécanisme d'acquittement de TCP est cumulatif :
  - il indique le numéro de séquence du prochain octet attendu : tous les octets précédents cumulés sont implicitement acquittés
  - Si un segment a un numéro de séquence supérieur au numéro de séquence attendu (bien que dans la fenêtre), le segment est conservé mais l'acquittement référence toujours le numéro du séquence attendu.
- Pour tout segment émis, TCP s'attend à recevoir un acquittement :
  - Si le segment n'est pas acquitté, le segment est considéré comme perdu et TCP le retransmet.
  - Or un réseau d'interconnexion offre des temps de transit variables nécessitant le réglage des temporisations;
  - TCP gère des temporisations variables pour chaque connexion en utilisant un algorithme de retransmission adaptative,

Acquittement dans TCP (suite)

TCP : Retransmissions

Algorithme de retransmission adaptative

1. **enregistre la date d'émission d'un segment,**
2. **enregistre la date de réception de l'acquittement correspondant,**
3. **calcule l'échantillon de temps de boucle A/R écoulé (New_RTT),**
4. **détermine le temps de boucle moyen RTT (Round Trip Time) :**
   - **RTT = (a * old_RTT) + ((1-a) * New_RTT))**

     avec 0<= a < 1

     a proche de 1 : RTT insensible aux variations brèves,

     a proche de 0 : RTT très sensible aux variations rapides,

5. **calcule la valeur du temporisateur en fonction de RTT.**
   - **Les premières implémentations de TCP ont choisi un coefficient constant B pour déterminer cette valeur :**
     - **Temporisation = B * RTT avec B >1 (généralement B=2).**
   - **Aujourd'hui de nouvelles techniques sont appliquées pour affiner la mesure du RTT : l'algorithme de Karn.**

Gestion de la congestion

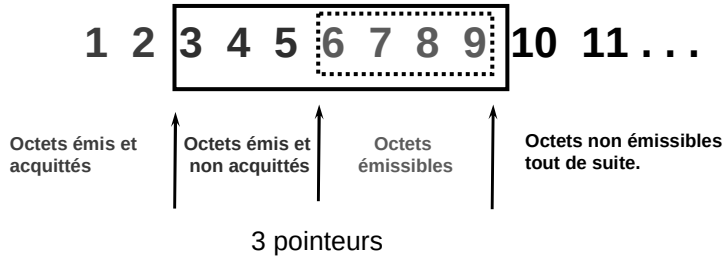
TCP : Contrôle de flux par fenêtrage

- La technique acquittement simple pénalise les performances puisqu'il faut attendre un acquittement avant d'émettre un nouveau message. Le fenêtrage améliore le rendement des réseaux.
- La technique du fenêtrage : une fenêtre de taille T, permet l'émission d'au plus T messages "*non acquittés*" avant de ne plus pouvoir émettre :
- fenêtrage glissante permettant d'optimiser la bande passante
- permet également au destinataire de faire diminuer le débit de l'émetteur donc de gérer le contrôle de flux.

Fenêtrage: suite

TCP : Contrôle de flux par fenêtrage (2)

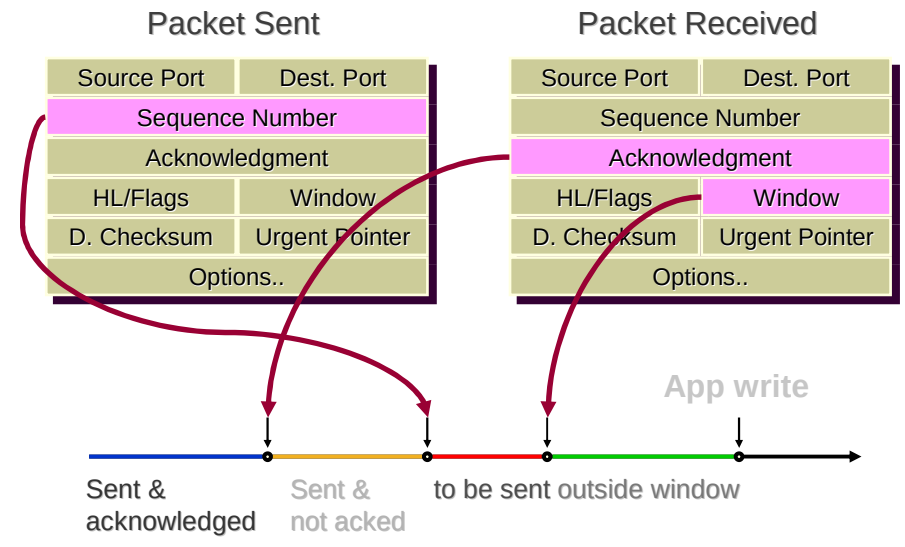
- Le mécanisme de fenêtrage mis en oeuvre dans TCP opère au niveau de l'octet et non pas au niveau du segment; il repose sur :
  - la numérotation séquentielle des octets de données à envoyer,
  - la gestion de trois pointeurs par fenêtrage :



3 pointeurs

Format du segment TCP

TCP : Contrôle de flux - Sender Side



TCP : Contrôle des congestions

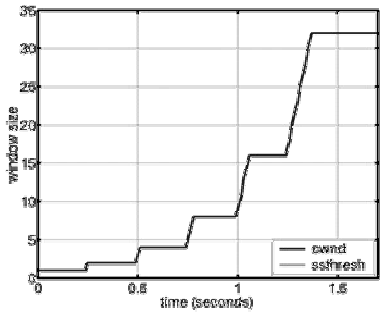
- TCP gère :
  - le contrôle de flux de bout en bout (saturation des tampons des terminaux récepteurs) par une fenêtre de transmission;
  - mais également les problèmes de congestion des tampons des routeurs au moyen d'une fenêtre de congestion.
- Le contrôle de congestion est basée sur 3 mécanismes :
  - Le Slow Start (SS)
  - Le Congestion Avoidance (CA)
  - Le Multiplicative Decrease (MD)
- Utilise 3 variables :
  - Une fenêtre de congestion : cnwnd
  - Un temporisateur fonction du RTT (délai aller-retour)
  - Un seuil : ssthresh

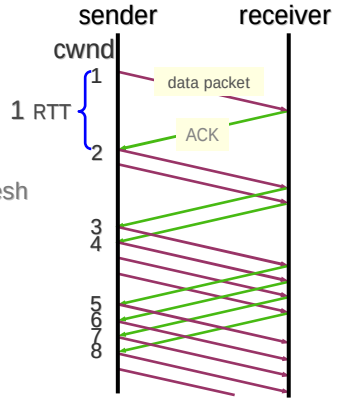
Slow Start

- Start with cwnd = 1 (slow start)
- On each successful ACK increment cwnd

  cwnd ← cnwd + 1
- Exponential growth of cwnd

  each RTT: cwnd ← 2 x cwnd
- Enter Congestion Avoidance when cwnd >= ssthresh



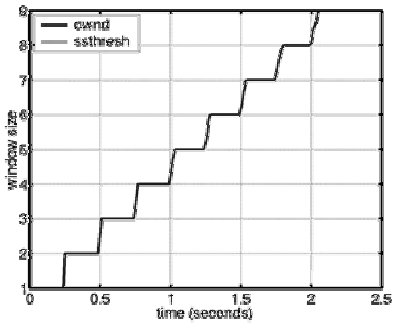


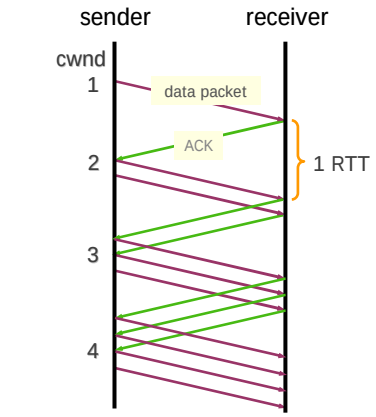
cwnd ← cwnd + 1 (for each ACK)

Congestion Avoidance

- Starts when cwnd ≥ ssthresh
- On each successful ACK:

  cwnd ← cwnd + 1/cwnd

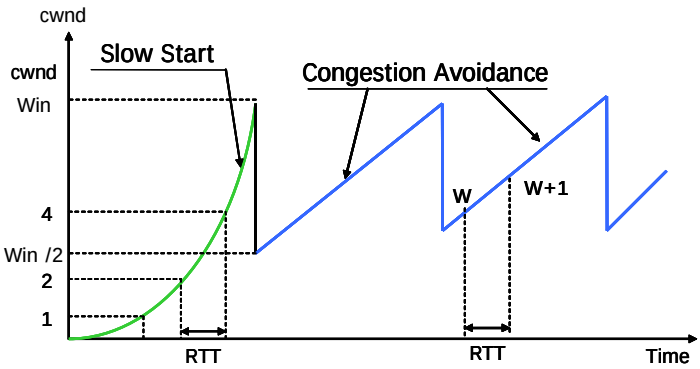




cwnd ← cwnd + 1 (for each group of ACKs per cwnd)

Multiplicative Decrease

- On each packet drop ➔ decrease cwnd:
- cwnd ← Win / 2



TCP summary

```
Initially:
    cwnd = 1;
    ssthresh = infinite;
New ack received:
    if (cwnd < ssthresh)
        /* Slow Start*/
        cwnd = cwnd + 1;
    else
        /* Congestion Avoidance */
        cwnd = cwnd + 1/cwnd;
Timeout: (loss detection)
    /* Multiplicative decrease */
    ssthresh = window/2;
    cwnd = 1;
```

SS: Slow Start
CA: Congestion Avoidance
MD : Multiplicative Decrease