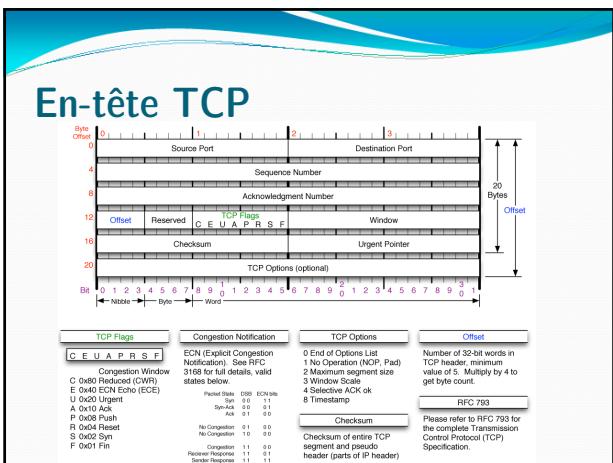




# TCP

- TCP offre un service de flux de données orienté connexion
  - **Orienté connexion** : *les deux partenaires de la communication doivent établir une connexion avec de pouvoir échanger des données*
- TCP est un protocole de transport fiable
- TCP assure le séquencement des données

- Les données sont fractionnées en **fragments** dont la taille est fixée par TCP
  - (contrairement à UDP qui effectue une écriture pour chaque ensemble de données fourni par l'application)
- A chaque émission d'un segment TCP est associé un **temporisateur** (timer) qui s'écoule en attendant un accusé de réception
- TCP maintient également une **somme de contrôle** qui permet de détecter si une modification a eu lieu pendant la transmission
- Chaque segment TCP est émis dans un datagramme IP
- Les datagrammes IP peuvent arriver à destination dans un ordre quelconque
  - TCP a pour tâche de réordonner les données avant de les transmettre à l'application
- TCP est capable de détecter les duplications de paquets
- TCP fournit un contrôle de flux. Chaque extrémité TCP a une taille finie d'espace de stockage (buffers)
  - attente et remise en ordre



**En-tête TCP****Détail des champs**

- Flags
  - **CWR\*** (*congestion window reduced*): réduction débit suite à réception flag ECE
  - **ECE\*** (*ECN echo*) : voir *explicit congestion notification*
  - **URG** (*urgent*) : le segment contient un pointeur urgent valide
  - **ACK** (*acknowledgment*) : le segment contient un n° d'acquittement valide
  - **PSH** (*push*) : les données en attente dans les buffers doivent être remontées à l'application
  - **RST** (*reset*) : arrêt de la connexion
  - **SYN** (*synchronize*) : synchronisation des nos de séquence initiaux
  - **FIN** (*final*) : dernier segment de données

---



---



---



---



---



---



---

**En-tête TCP****Détail des champs**

- Taille de la fenêtre (*window size*) : annonce de la taille de la fenêtre de réception
- Somme de contrôle (*TCP checksum*) : vérification en-tête et données
- Pointeur urgent (*urgent pointer*) : les données urgentes sont positionnées dès le début de la zone de données ; le pointeur urgent désigne la position de la suite des données (si URG = 1)

---



---



---



---



---



---



---

**En-tête TCP****Détail des champs**

- Options
  - optionnelles ! (voir data offset)
  - TLV
  - exemples
    - **MSS** : taille maximale du segment de données applicatives que l'émetteur accepte de recevoir.
    - **timestamp** : utilisé notamment pour suivre la durée d'aller-retour (RTT).
    - **wscale** : facteur d'échelle ("shift") pour augmenter la taille de la fenêtre au delà des 16 bits du champ WINDOW (> 65535).
      - Quand cette valeur n'est pas nulle, la taille de la fenêtre est de  $65535 \times 2^{shift}$
    - **nop** Les options utilisent un nombre quelconque d'octets ; par contre les paquet TCP sont toujours alignés sur une taille de mot de quatre octets; complétion des mots.

---



---



---



---



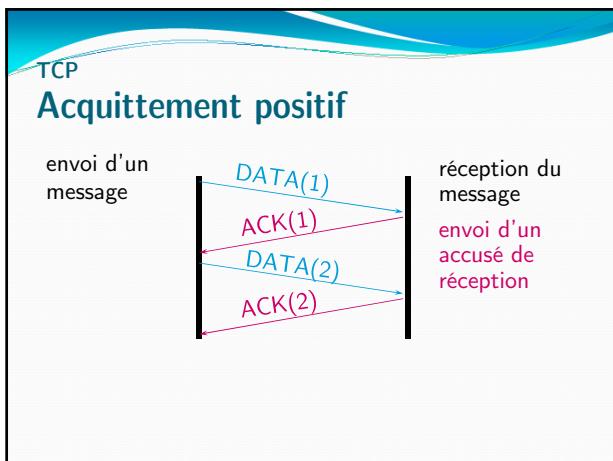
---



---



---




---



---



---



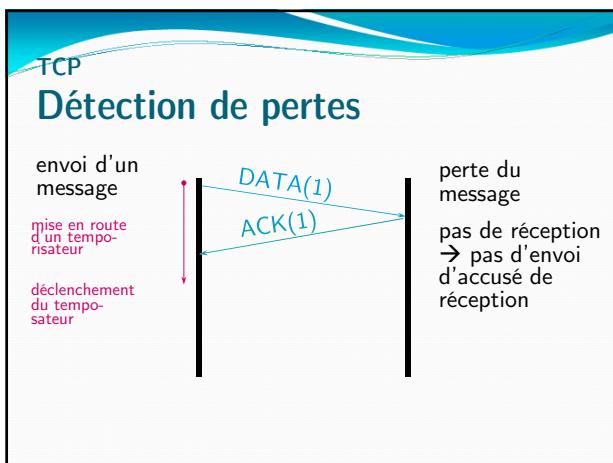
---



---



---




---



---



---



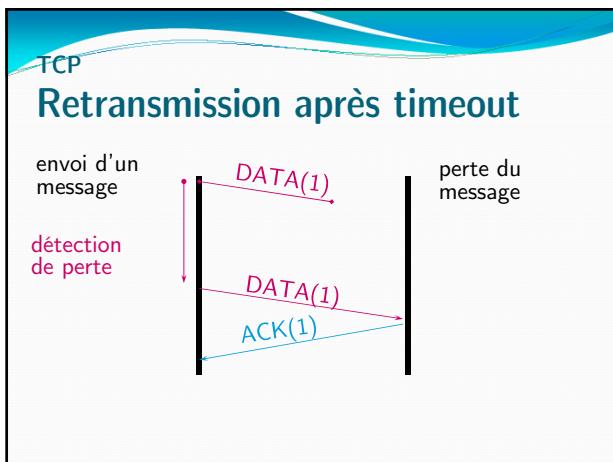
---



---



---




---



---



---



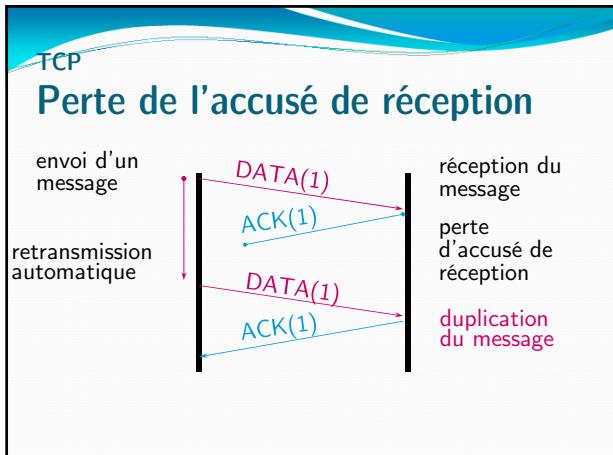
---



---



---




---



---



---



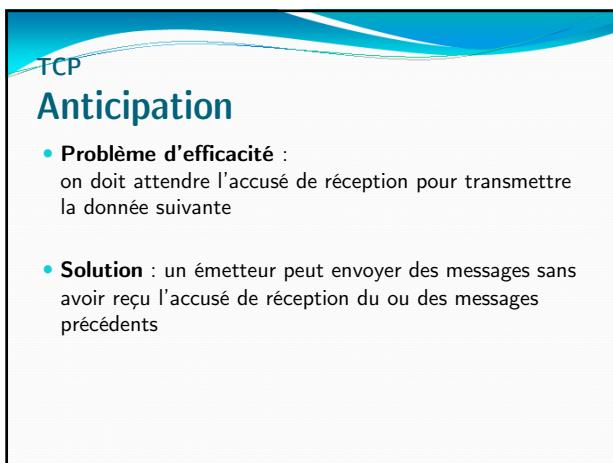
---



---



---




---



---



---



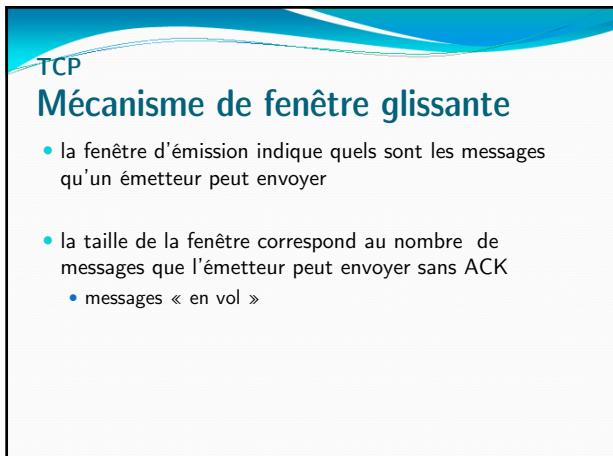
---



---



---




---



---



---



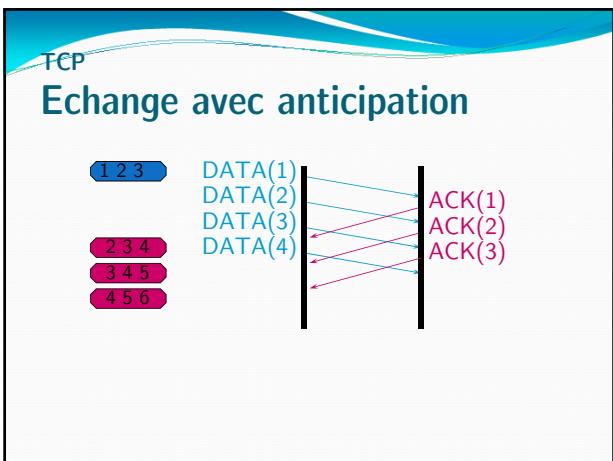
---



---



---




---

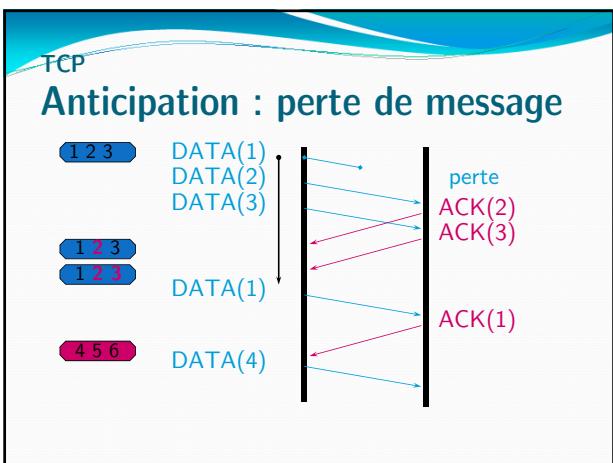
---

---

---

---

---




---

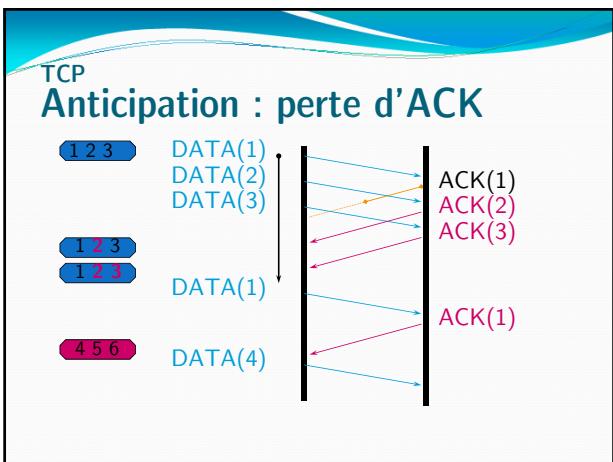
---

---

---

---

---




---

---

---

---

---

---

TCP

## Acquittement cumulatif

- l'accusé de réception *cumulatif* indique :
  - le numéro du premier message que l'on a pas encore reçu
  - le numéro du prochain message que l'on attend
  - que l'on a reçu tous les messages précédents jusqu'au numéro indiqué (exclu)

---



---



---



---



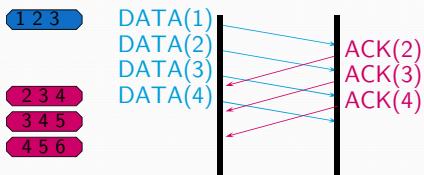
---



---

TCP

## Anticipation et acquittement cumulatif




---



---



---



---



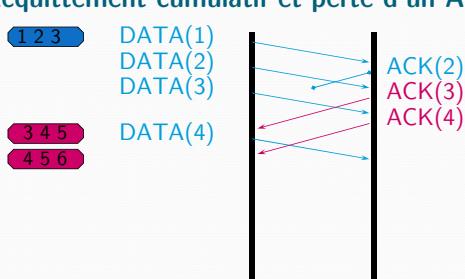
---



---

TCP

## Acquittement cumulatif et perte d'un ACK




---



---



---



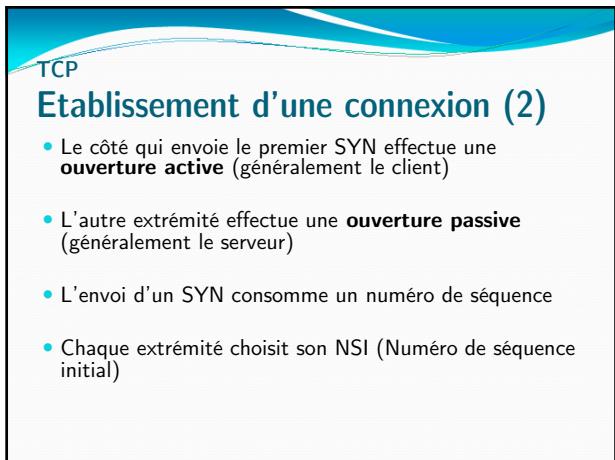
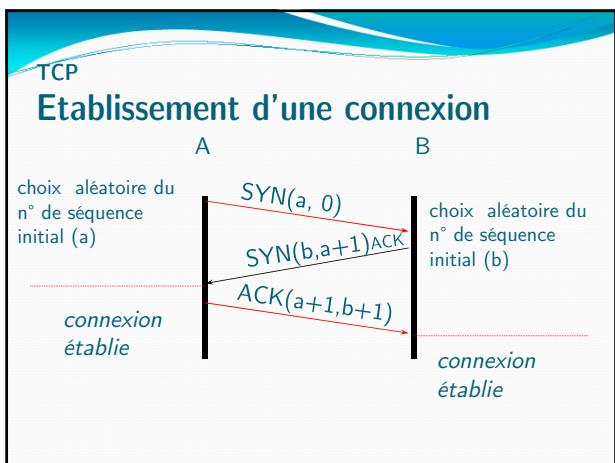
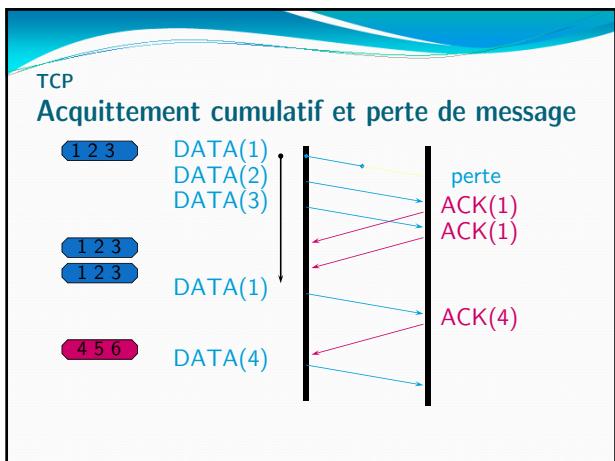
---



---



---



TCP

## Taille maximale d'un segment

- La taille maximale de segment (MSS) est le plus grand « morceau » de données que TCP enverra à l'autre extrémité
- Lors de l'établissement d'une connexion chaque extrémité peut annoncer sa MSS
- L'option MSS ne peut apparaître que dans un segment SYN
- La valeur par défaut est 536
- En règle générale les implémentations de TCP choisissent un MSS le plus grand possible tout en évitant la fragmentation

---



---



---



---



---



---

TCP

## Fin d'une connexion

- Il faut 3 segments TCP pour ouvrir une connexion
- Il faut 4 segments TCP pour fermer une connexion
- On parle de **semi-fermeture**
  - Une connexion TCP est full-duplex, chaque direction peut-être arrêtée indépendamment
  - Chaque extrémité envoie un segment FIN
  - La réception d'un segment FIN signifie qu'il n'y aura plus de données émises dans cette direction
- L'extrémité qui effectue en premier la fermeture fait une **fermeture active**
- Alors que l'autre extrémité effectue une **fermeture passive**

---



---



---



---



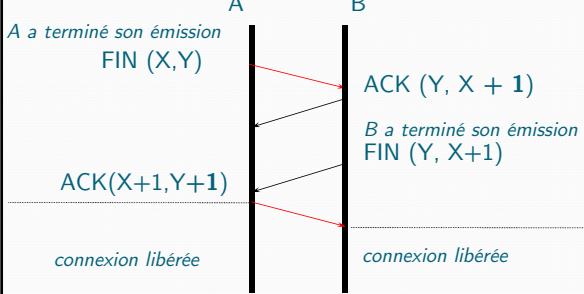
---



---

TCP

## Fin d'une connexion (2)




---



---



---



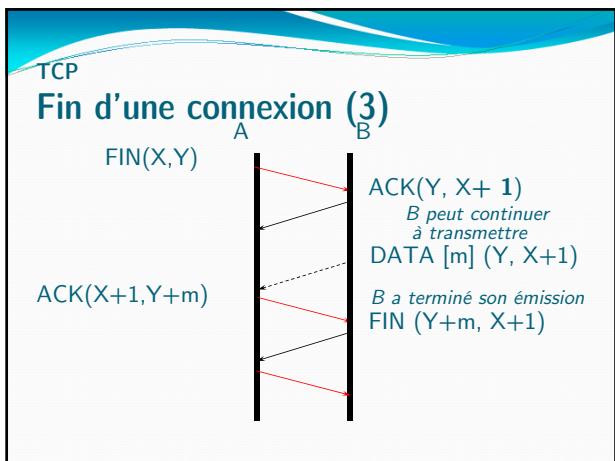
---



---



---




---



---



---



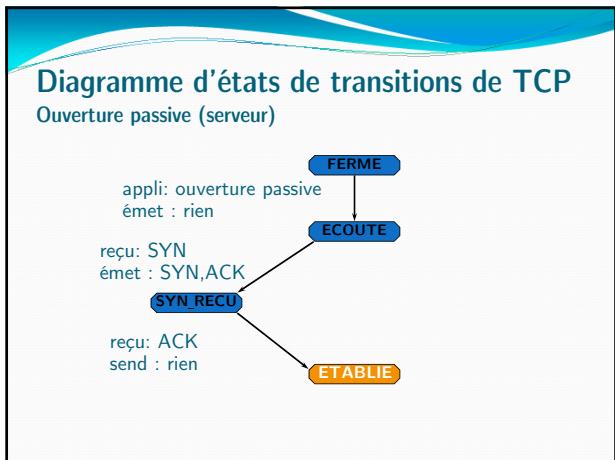
---



---



---




---



---



---



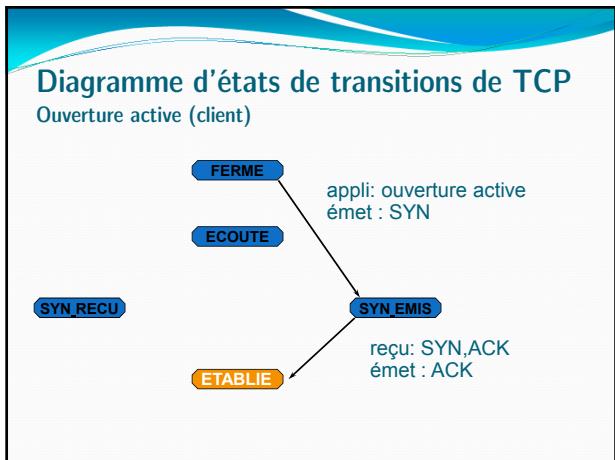
---



---



---




---



---



---



---



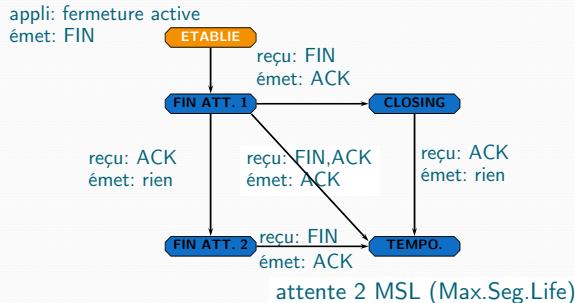
---



---

## Diagramme d'états de transitions de TCP

Fermeture active

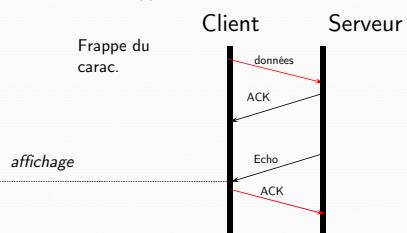


## Flux de données interactif

- Cas du telnet ou du rlogin : l'utilisateur tape une commande
- Chaque frappe de caractères génère un paquet de données
- Dans le cas de telnet et rlogin il faut également gérer l'écho du caractère tapé :
  - 1) frappe du caractère (envoyé au serveur)
  - 2) Acquittement de la frappe
  - 3) Echo de la frappe envoyé par le serveur
  - 4) Acquittement de l'écho provenant du serveur

## Flux de données interactif

Schéma de la frappe d'un caractère



Les segments 2 et 3 sont généralement combinés

## TCP

### Démarrage lent

- L'émetteur peut-il émettre des données jusqu'à remplir la fenêtre annoncée par le destinataire ?
- Hypothèse valable si les deux machines se trouvent sur le même lien
- Dans l'internet les routeurs situés entre l'émetteur et le récepteur peuvent mettre dans des file d'attentes les paquets
- Les routeurs pourraient même être saturés et détruire certains paquets

---



---



---



---



---



---

## TCP

### Démarrage lent

- Le démarrage lent
  - Permet à un émetteur d'observer la vitesse à laquelle les nouveaux paquets doivent être injectés dans le réseau
  - Permet d'observer à quelle vitesse les acquittements reviennent
- Le démarrage lent ajoute une nouvelle fenêtre au TCP de l'émetteur : la fenêtre de congestion (cwnd)
- Lorsqu'une congestion est constatée la fenêtre de congestion est initialisée à la taille du segment annoncée par l'autre extrémité
- A chaque ACK reçu la fenêtre est incrémentée d'une taille de segment supplémentaire
- L'émetteur ne peut émettre plus que le min(cwnd, et la fenêtre annoncée)
- La fenêtre annoncée par le récepteur est un contrôle de flux imposé par le récepteur
- La fenêtre de congestion est un contrôle de flux imposé par l'émetteur

---



---



---



---



---



---



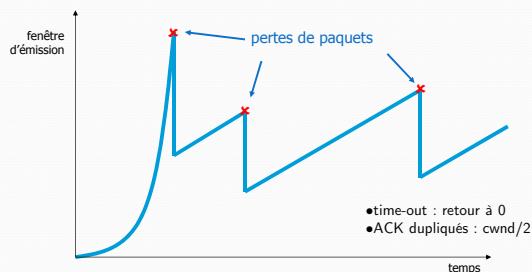
---



---

## TCP

### Démarrage lent + évitement de congestion




---



---



---



---



---



---



---



---

## A suivre...

- Acquittements TCP
- Etude du rfc2001
- Implémentations TCP

---

---

---

---

---

---