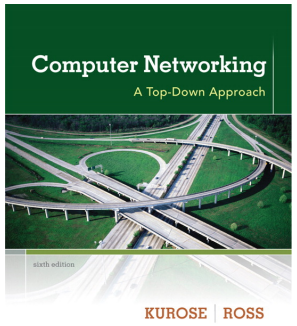
Chapitre 3
La couche Transport

LOG100/GTI100 Programmation et Réseautique en génie logiciel/des TI

- Le contenu de cette présentation est basé sur le livre de Kurose et Ross et de la documentation y jointe :

  Computer Networking: A Top Down Approach, 6ème édition.
  Jim Kurose, Keith Ross
  Addison-Wesley, Mars 2012,
  ISBN-13: 978-0132856201



Chapitre 3 : la couche Transport

Objectifs:

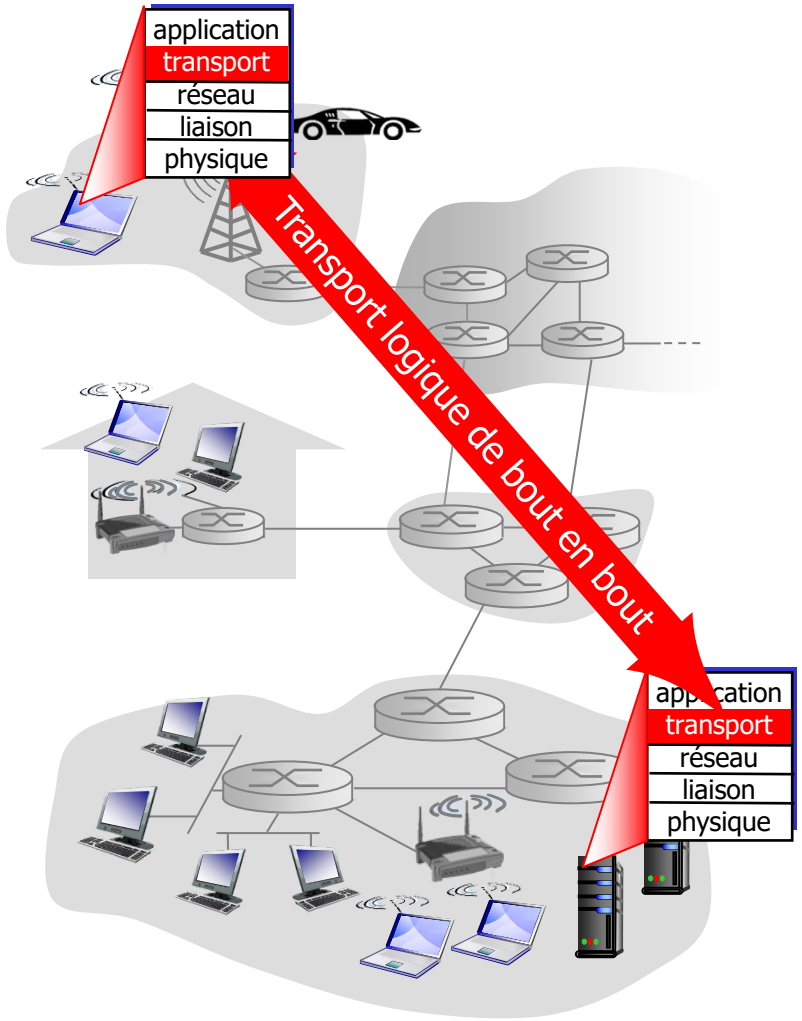
- Comprendre les principes des services offerts par la couche transport:
  - multiplexage et démultiplexage
  - transfert fiable de données
  - contrôle de flux
  - contrôle de congestion
- Maîtriser les protocoles de la couche transport utilisés dans l'internet
  - UDP : transport sans connexion
  - TCP : transport fiable orienté connexion

Chapitre 3 : la couche transport

1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

Services de transport et protocoles

- Offrir *une communication logique* entre les processus exécutés sur différents hôtes
- les protocoles de transport roulent sur les terminaux
  - côté émetteur : découper le message en segments, les passer à la couche réseau
  - côté récepteur : réassembler les segments en messages, les passer à la couche application
- il y a plusieurs protocoles disponibles:
  - Internet : TCP et UDP



Couche transport vs. réseau

- *Couche réseau:* communication logique entre les hôtes
- *Couche transport:* communication logique entre les processus
  - Utilise les services de la couche réseau

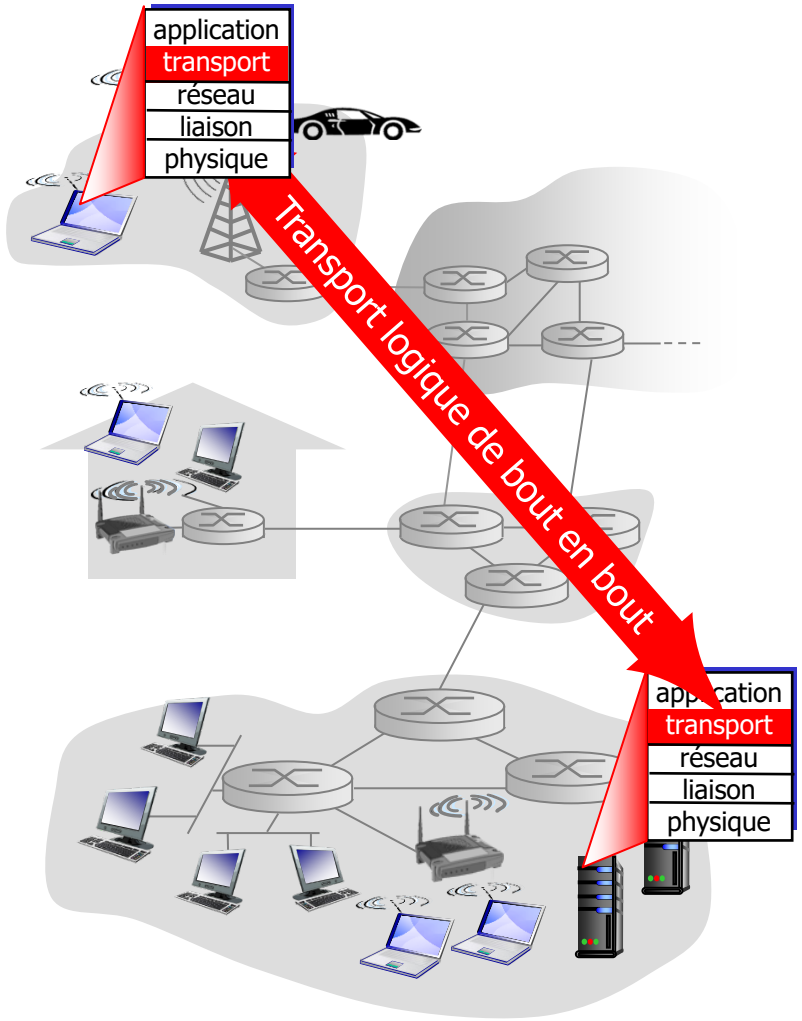
Analogie maisons:

12 enfants à la maison de Anne envoyant des lettres à 12 enfants de la maison de Bill

- hôtes = maisons
- processus = enfants
- messages app = lettres dans des enveloppes
- protocole de transport = Anne et Bill
- protocole de la couche réseau = service postal

Les protocoles transport d'Internet

- TCP: fiable, livre les segments dans l'ordre
  - contrôle de congestion
  - contrôle de flux
  - établissement de connexion
- UDP: non fiable, ne livre pas les segments dans l'ordre
  - extension à IP "best-effort"
- services non disponibles:
  - garantie de délai
  - garantie de bande passante



Chapitre 3 : la couche transport

1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

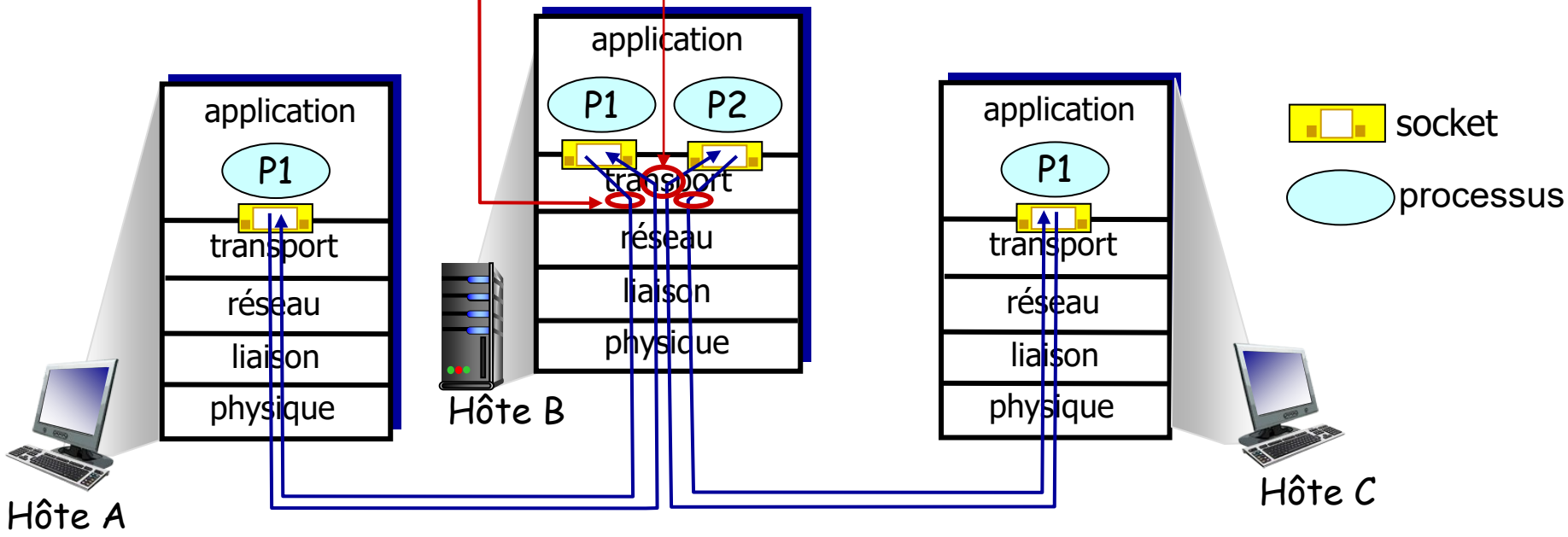
Multiplexage et démultiplexage

multiplexage à l'émetteur :

traiter les données provenant des sockets, et ajouter l'entête transport (utilisé plus tard pour le démultiplexage)

démultiplexage au récepteur :

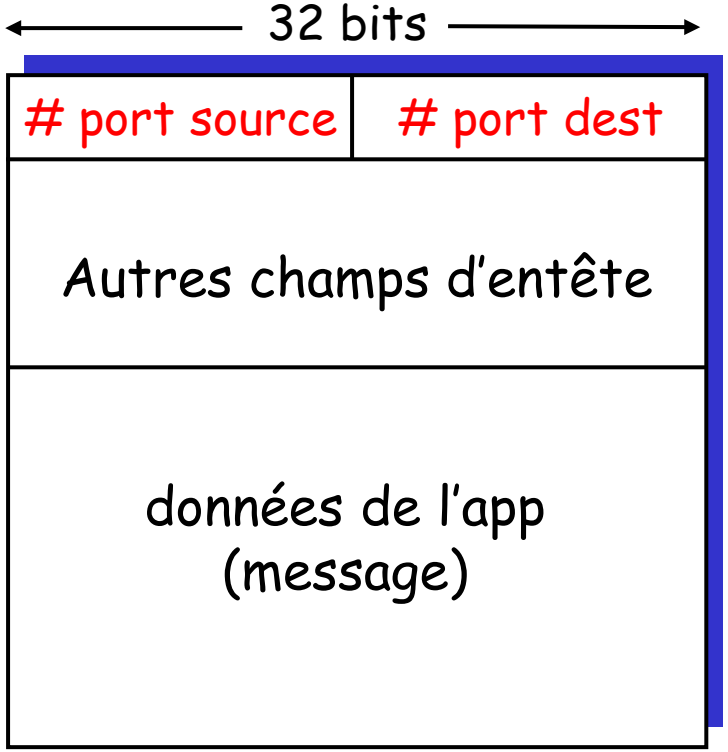
délivrer les segments reçus aux sockets correspondants



Comment fonctionne le démultiplexage

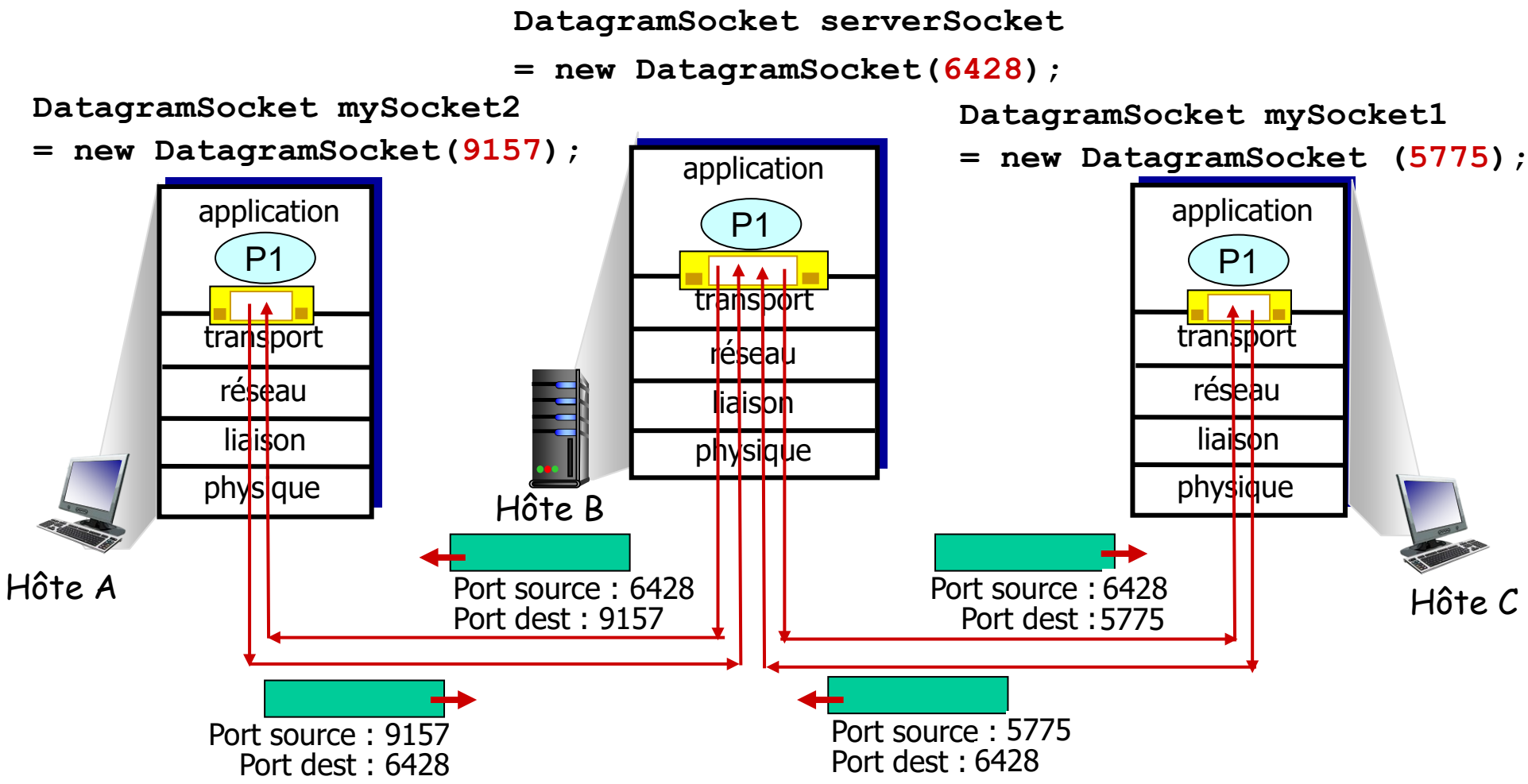
- La destination reçoit des datagrammes IP
  - chaque datagramme contient les adresses IP source et destination
  - chaque datagramme porte 1 segment de la couche transport
  - chaque segment contient les numéros de port source et destination

- La destination utilise les adresses IP et les numéros de port pour diriger le segment vers le socket approprié



format du segment TCP/UDP

Démultiplexage UDP : Exemple



Démultiplexage UDP

- *Créer un socket client* :

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
```

- *Créer un socket serveur* :

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

- *Pour envoyer un datagramme dans un socket UDP, on doit spécifier :* ***l'adresse IP destination et le # port destination***

```
DatagramPacket Packet = new DatagramPacket(Data, Data.length, DstIPAddr, DstPort);
mySocket1.send(Packet);
```

- Le socket UDP est identifié par un doublet
  - Adresse IP destination
  - Numéro de port destination
- Quand un hôte reçoit un segment UDP :
  - Il vérifie le numéro de port destination dans le segment
  - Il dirige le segment UDP au socket ayant ce # port
- Les datagrammes IP avec le même numéro port destination, mais des adresses IP source différentes et/ou des numéros de port source différents seront dirigés vers le même socket à la destination

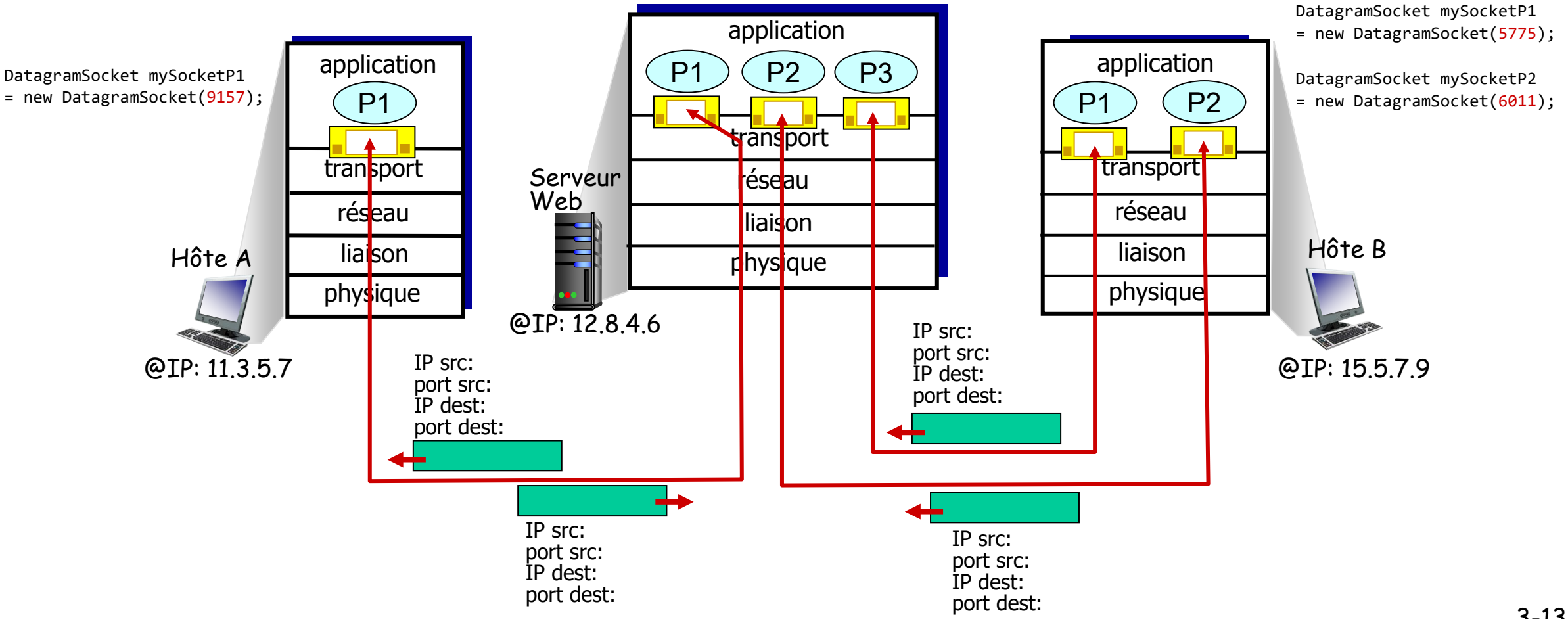
Démultiplexage TCP

- Le socket TCP est identifié Par un quadruplet:
  - adresse IP source
  - numéro de port source
  - adresse IP destination
  - numéro de port destination
- Le récepteur utilise les quatre valeurs pour diriger le segment vers le socket approprié

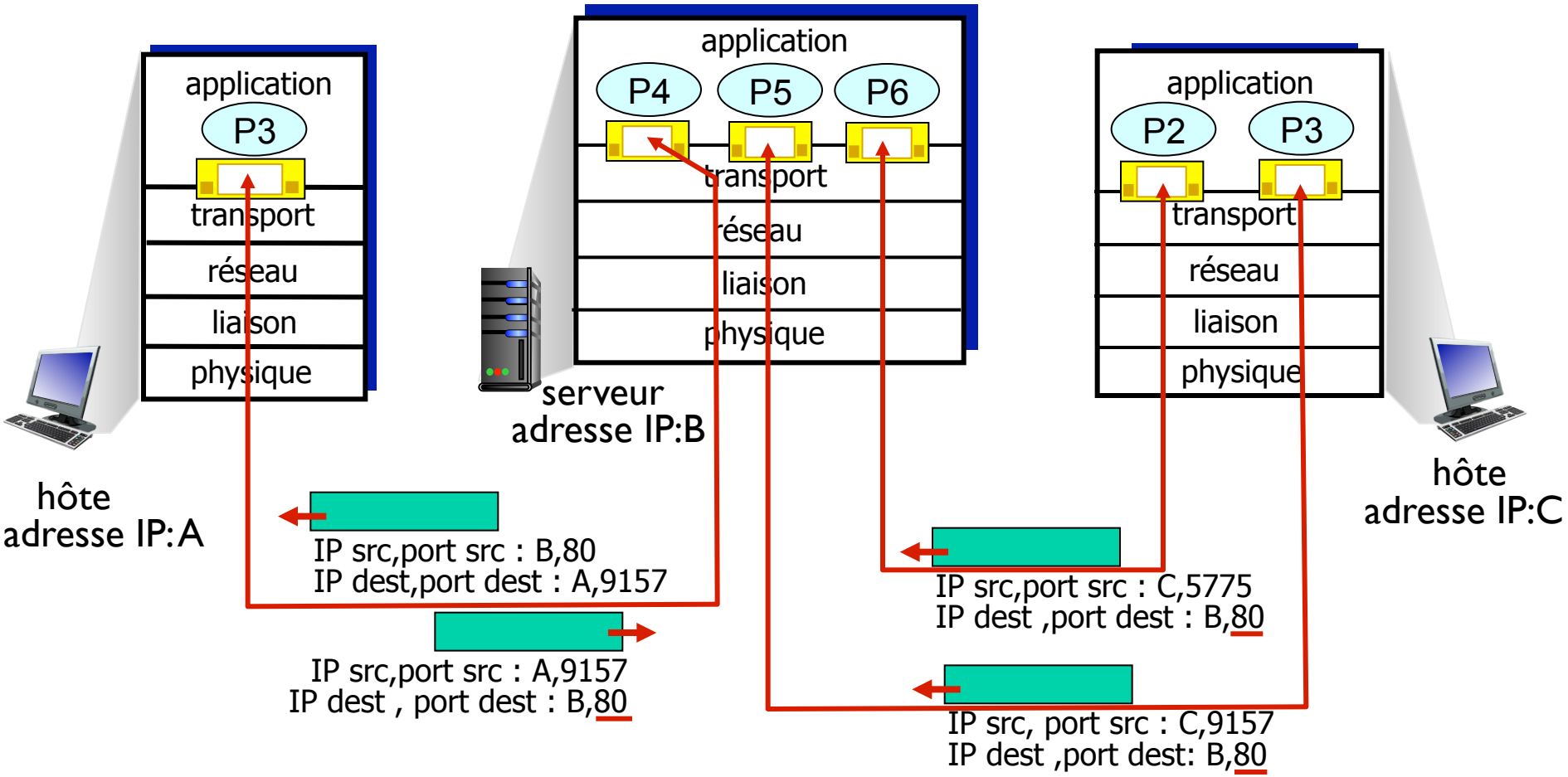
- Le serveur peut supporter plusieurs sockets TCP simultanément:
  - chaque socket est identifié par son propre quadruplet
- Les serveurs Web ont un socket pour chaque client connecté
  - HTTP non-persistant va avoir un socket pour chaque requête

Démultiplexage TCP : Exemple

3 segments destinés au serveur web qui sont démultiplexés vers des sockets différents

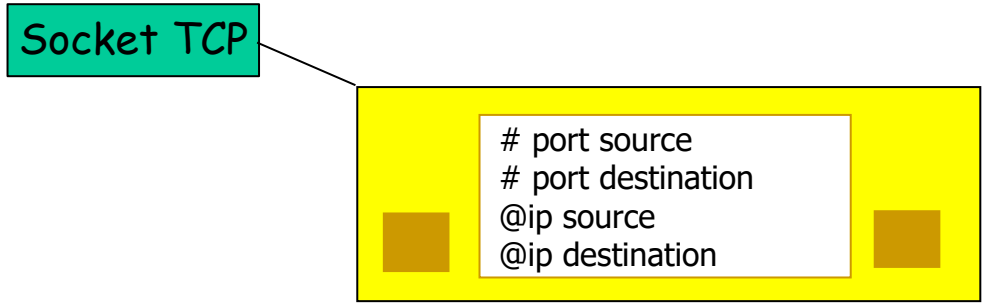


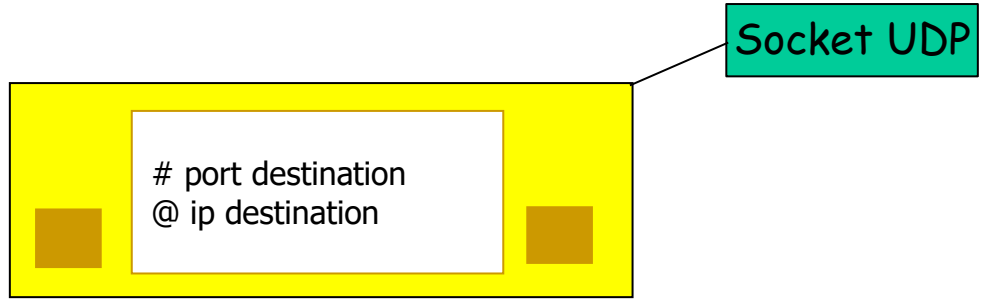
Démultiplexage TCP : Exemple



Example de trois segments destinés à (l'adresse IP : B, port destination: 80) qui sont demultiplexés vers des sockets différents

Résumé: socket TCP- socket UDP





Chapitre 3 : la couche transport

1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

UDP: User Datagram Protocol [RFC 768]

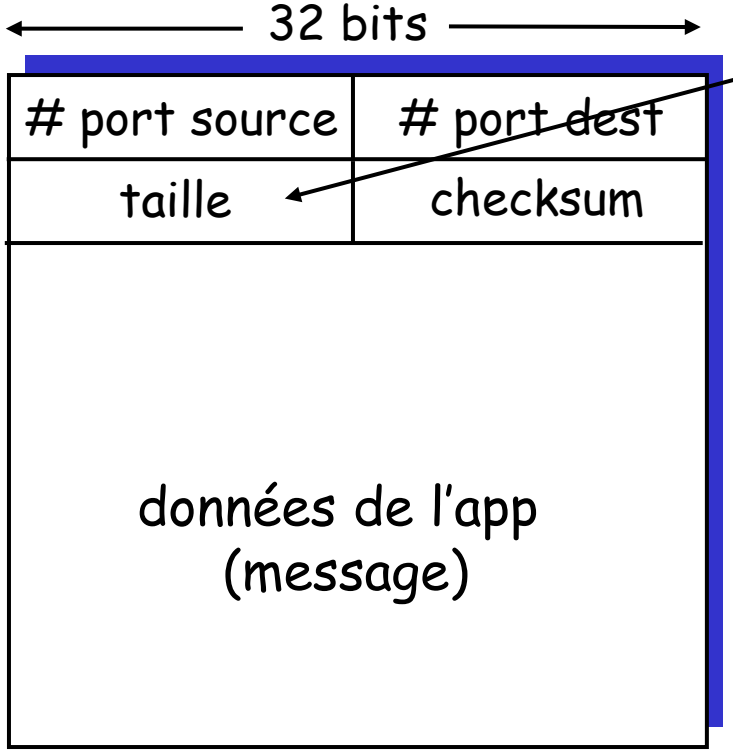
- Protocole de transport basique
- Service "best effort". Les segments UDP peuvent être:
  - perdus
  - livrés en désordre à l'application

- *Sans connexion:*
  - Pas d'établissement de connexion entre l'émetteur et le récepteur UDP
  - Chaque segment UDP est traité indépendamment des autres

- Utilisation d'UDP:
  - applications de streaming multimédia (tolérance aux pertes, sensibilité à la variation de débit)
  - DNS
  - SNMP
- Transfert fiable sur UDP : Comment faire?

UDP: entête UDP

Taille en octets d'un segment UDP, y compris l'entête



format d'un segment UDP

Pourquoi UDP?

- pas d'établissement de connexion (ça ajoute des délais)
- simple: pas d'état de connexion
- un petit entête
- pas de contrôle de congestion: UDP peut transmettre aussi rapidement qu'il veut

UDP somme de contrôle (Checksum)

objectif: détecter les "erreurs" (par ex., bits inversés) dans les segments (reçus).

émetteur:

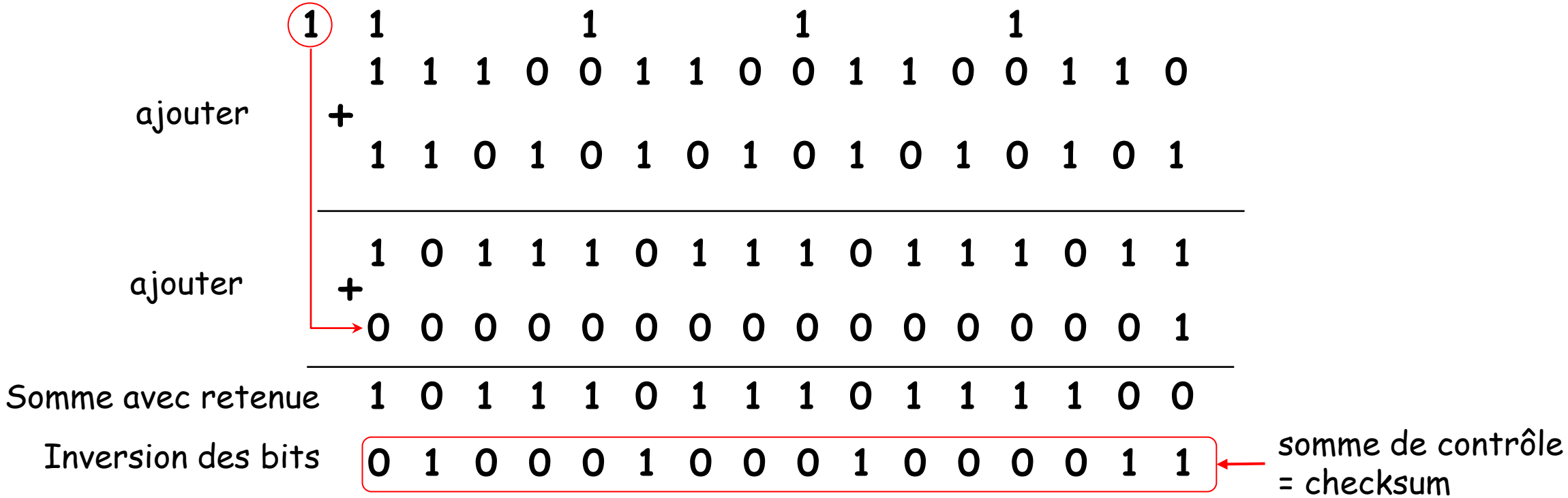
- traite le contenu du segment comme une séquence d'entiers de 16-bits (mot)
- calcule la somme de contrôle (checksum): le complément à 1 de la somme de tous les mots
- met la valeur de la somme de contrôle dans le champ checksum d'UDP

Récepteur:

- calcule la somme de contrôle du segment reçu
- vérifie si la somme de contrôle calculée égale à la valeur du champ checksum:
  - NON - erreur détectée
  - OUI – pas d'erreurs détectées. Est-ce qu'il peut y avoir des erreurs?

Exemple de checksum Internet

- Note: quand on additionne des nombres, le reste dans le bit le plus significatif doit être ajouté au résultat
- Exemple: addition de deux entiers de 16-bit



Chapitre 3 : la couche transport

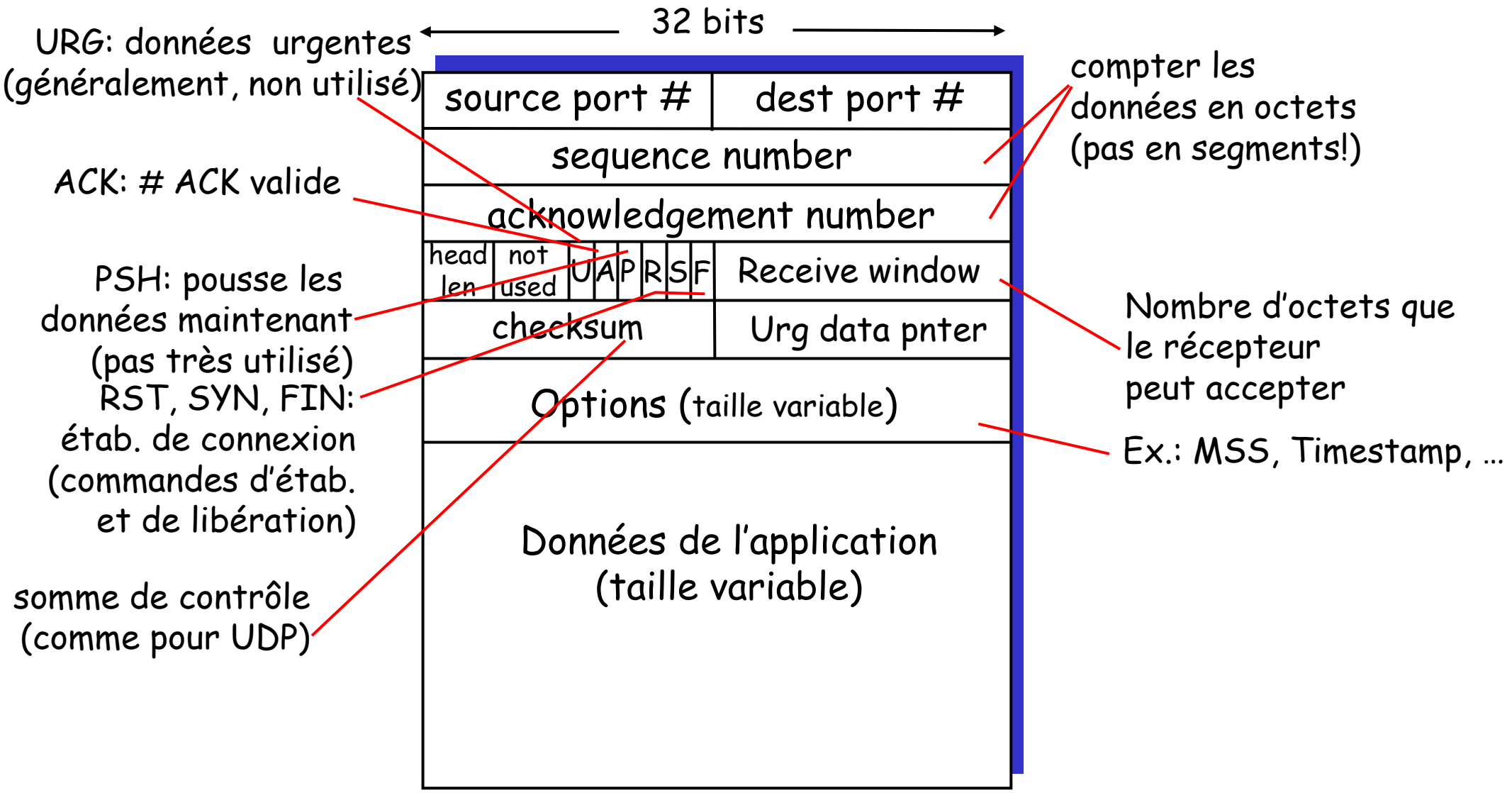
1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

TCP: une vue d'ensemble

RFCs: 793, 1122, 1323, 2018, 2581

- point-à-point:
  - 1 émetteur, 1 récepteur
- fiabilité de transmission
  - Checksum
  - Retransmission
- flux d'octets reçus dans l'ordre
- mécanismes de contrôle de flux et de congestion TCP:
  - Ces mécanismes décident de la taille de données que l'émetteur peut envoyer (appelée fenêtre)
  - l'émetteur ne submerge pas le récepteur et tient compte de l'état du réseau
- données en duplex:
  - flux de données bidirectionnel au sein de la même connexion
  - MSS (Maximum Segment Size): taille maximale du champs de données du segment TCP.
- orienté connexion:
  - échange de messages de contrôle (*handshaking*) pour initialiser les états de transmission et de réception avant l'échange de données

structure d'un segment TCP



Chapitre 3 : la couche transport

1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

séq. et ACKs dans TCP

Numéro de séquence:

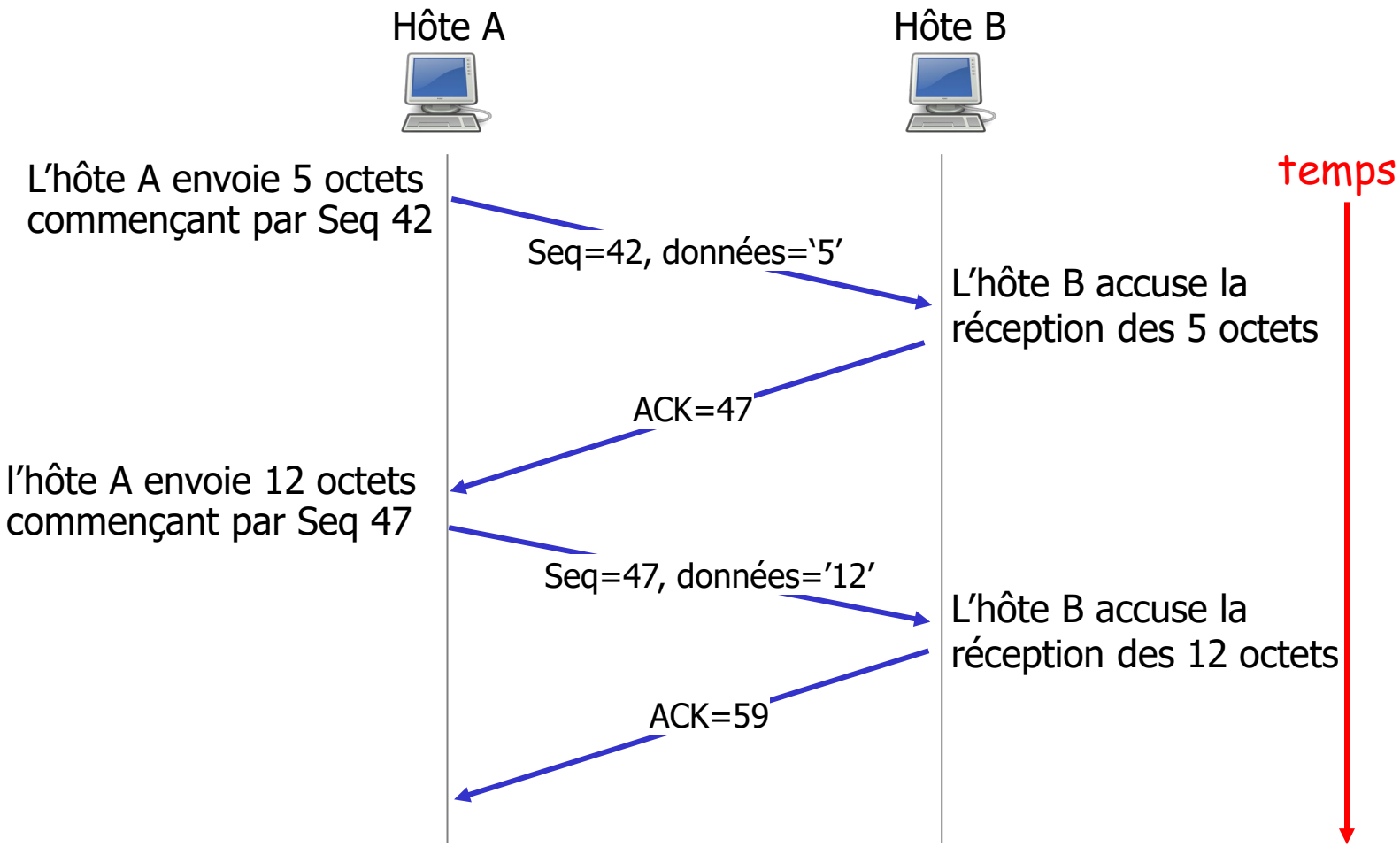
- le numéro du premier octet dans les données du segment

ACKs: Acquittement

- # de séq du prochain octet attendu
- ACK cumulatif

Q: comment le récepteur traite les segments non ordonnés ?

- TCP ne le précise pas : cela dépendra de l'implémentation



Scénario simple: seul l'hôte A envoie des données

séq. et ACKs dans TCP

Numéro de séquence:

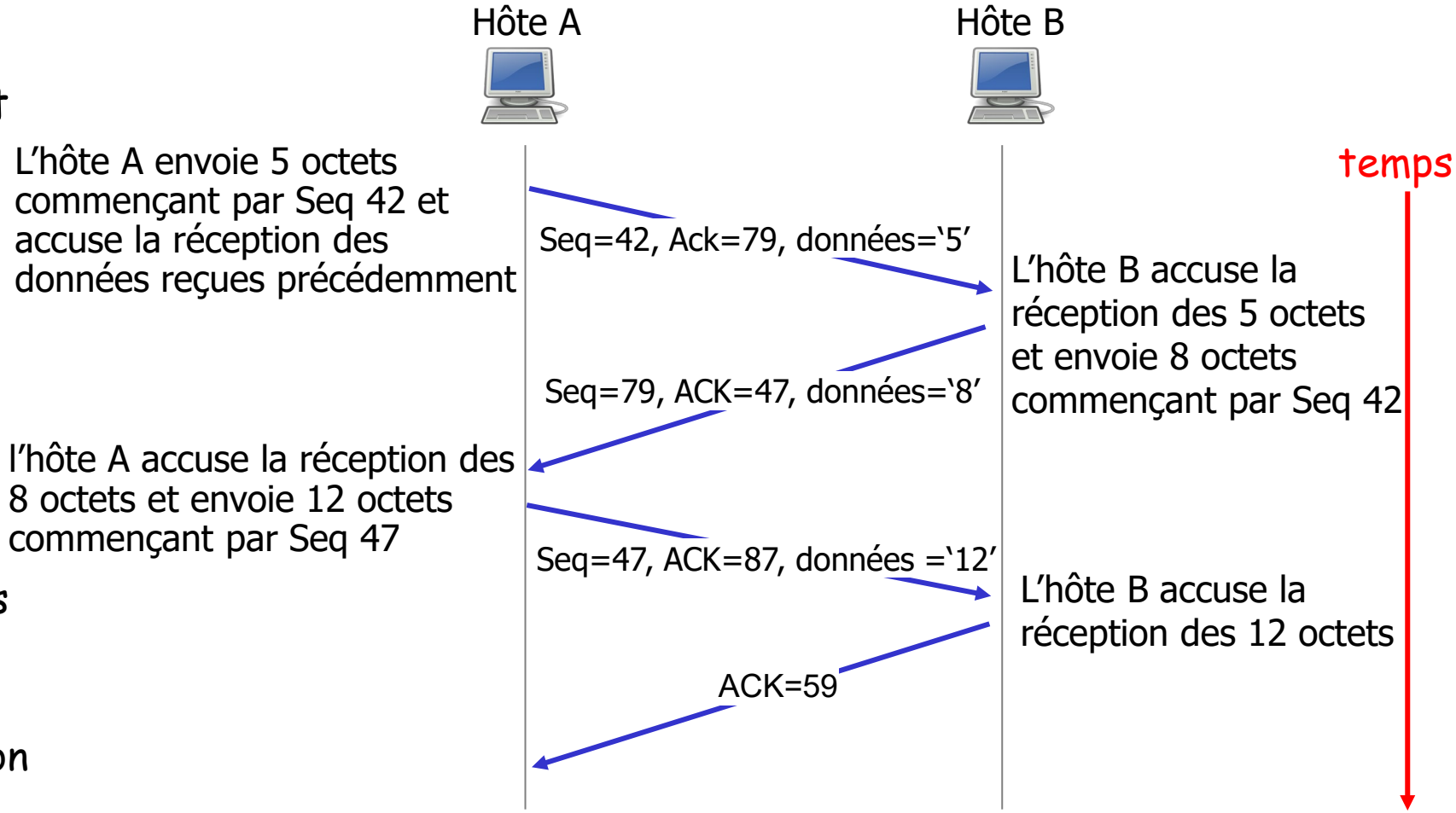
- le numéro du premier octet dans les données du segment

ACKs: Acquittement

- # de séq du prochain octet attendu
- ACK cumulatif

Q: comment le récepteur traite les segments non ordonnés ?

- TCP ne le précise pas : cela dépendra de l'implémentation

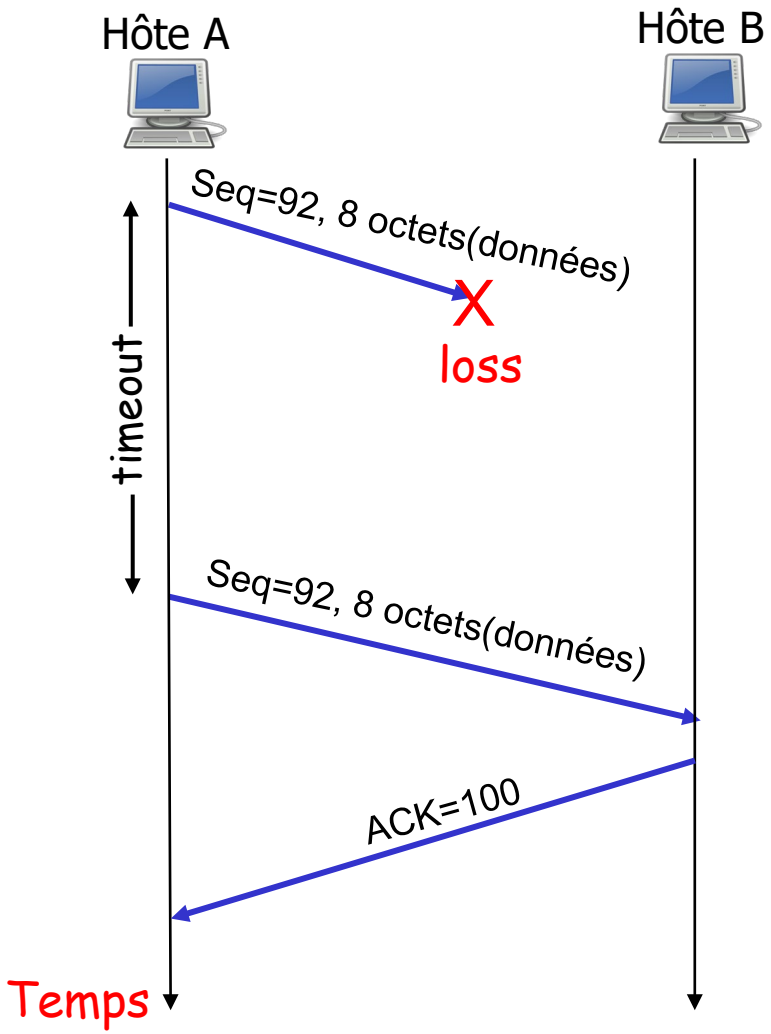


Scénario simple: les deux hôtes envoient des données

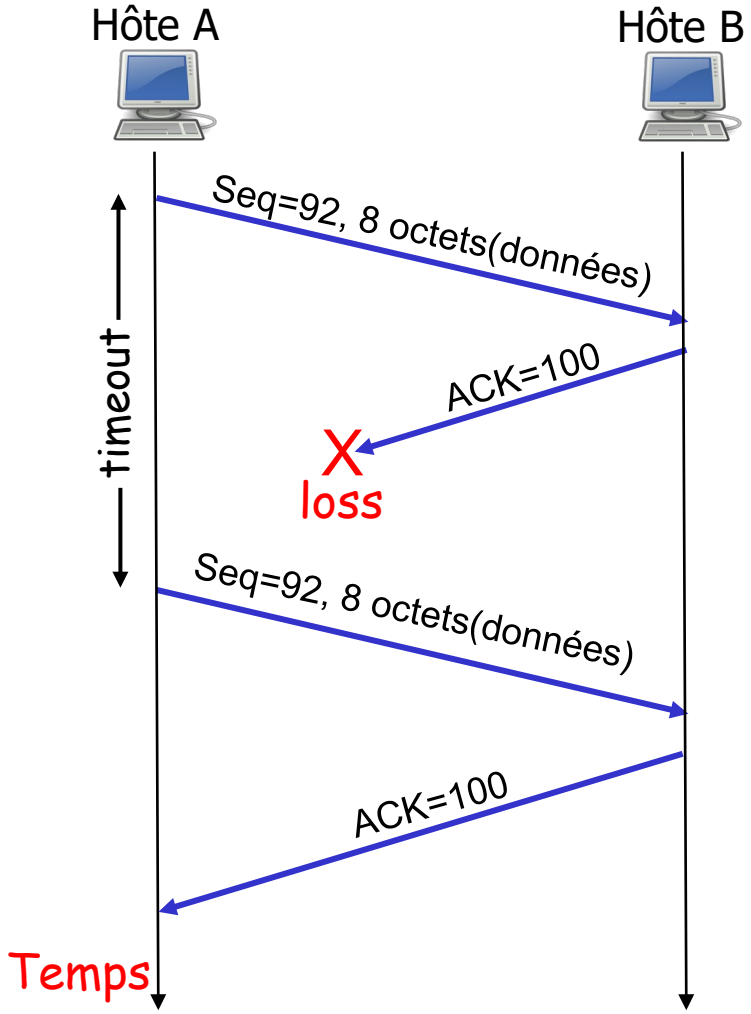
Fiabilité de transmission dans TCP

- TCP crée un service fiable au dessus d'un service non fiable (le protocole IP)
- Détection des erreurs
  - Somme de contrôle (*Checksum*)
- Acquittement des données reçues sans erreur
  - Acquittement (ACK) Cumulatifs
- Retransmission des paquets perdus:
  - Les retransmissions sont déclenchées par:
    - Timeout
    - ACKs dupliqués
  - TCP utilise un seul temporisateur de retransmission

TCP: scénarios de retransmission

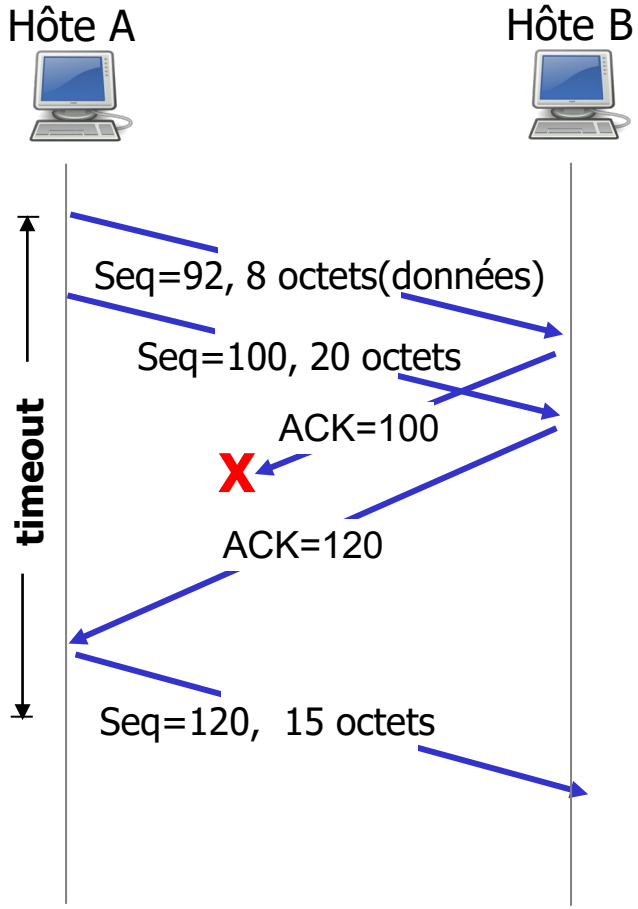


Scénario: un segment est perdu

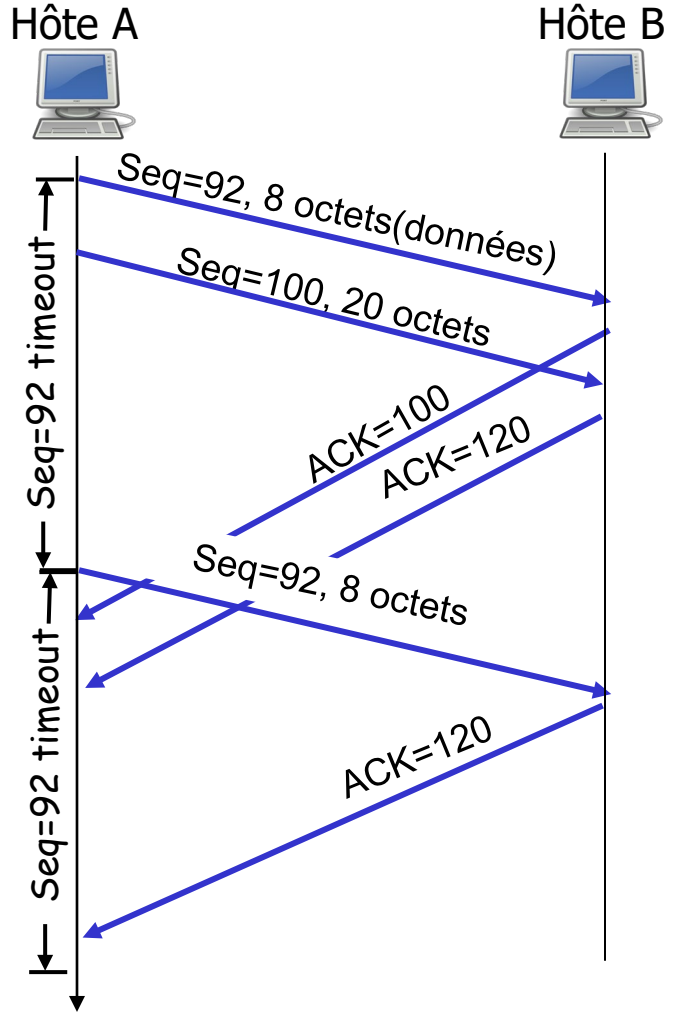


Scénario : un Ack est perdu

TCP: scénarios de retransmission

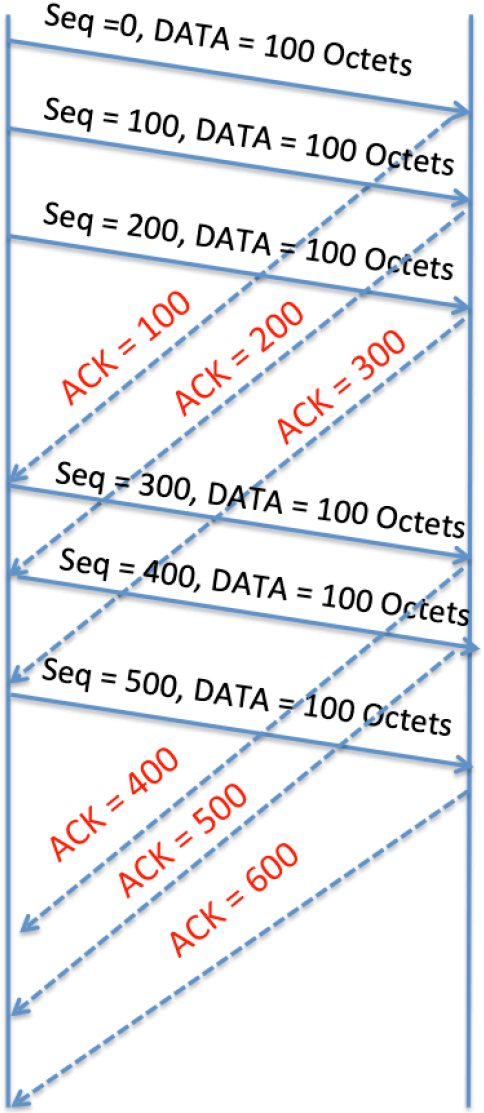


Scénario : Ack cumulatif



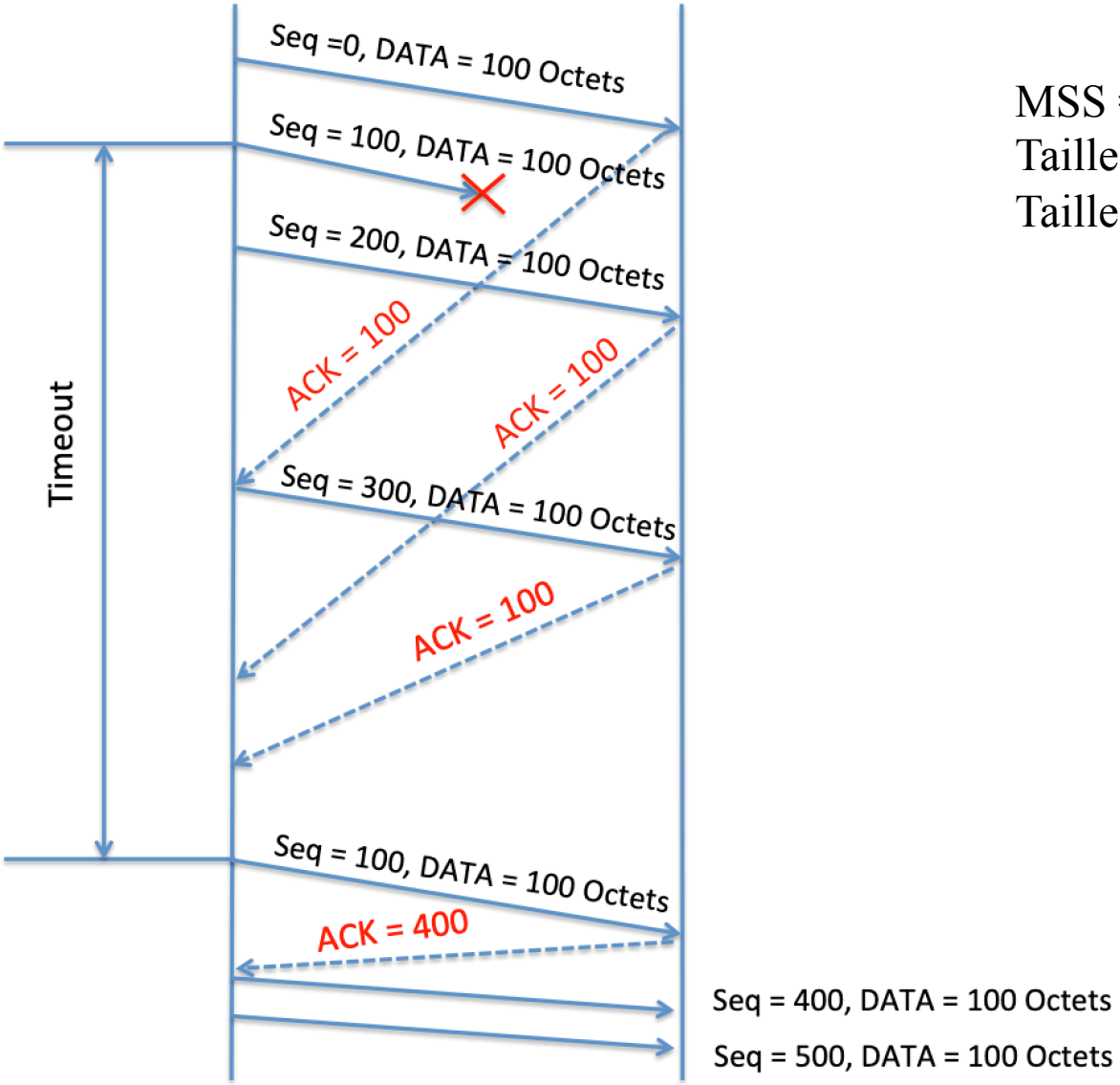
Scénario : timeout prématuré

Autres exemples d'échange TCP (cas idéal)



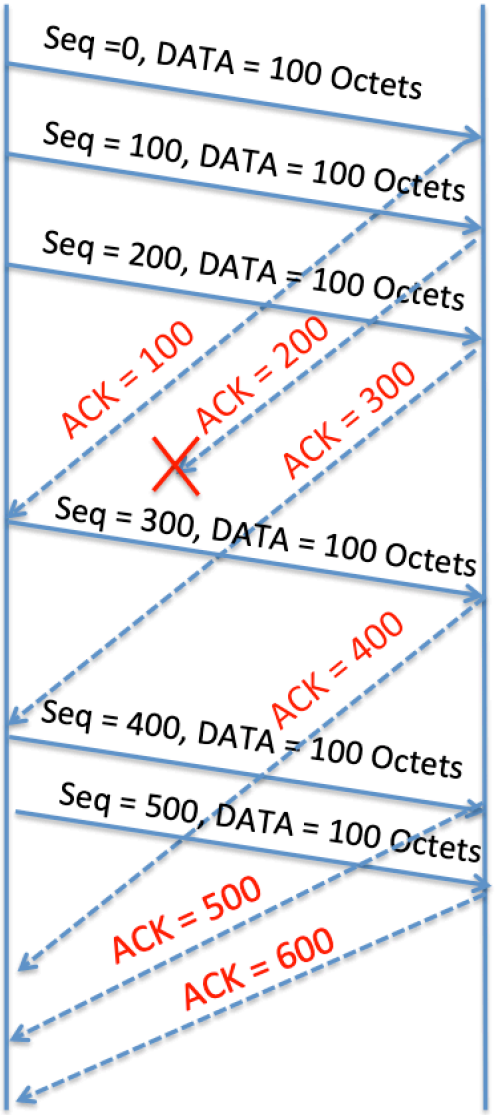
MSS = 100 Octets
Taille fichier = 600 Octets
Taille de fenêtre d'émission = 3

Autres exemples d'échange TCP (perte d'un segment)



MSS = 100 Octets
Taille fichier = 600 Octets
Taille de fenêtre d'émission = 3

Autres exemples d'échange TCP (perte d'un ACK)



MSS = 100 Octets
Taille fichier = 600 Octets
Taille de fenêtre d'émission = 3

Retransmission rapide de TCP

(TCP Fast Retransmit)

- Le timeout est souvent relativement long:
  - long délai avant la retransmission d'un paquet perdu

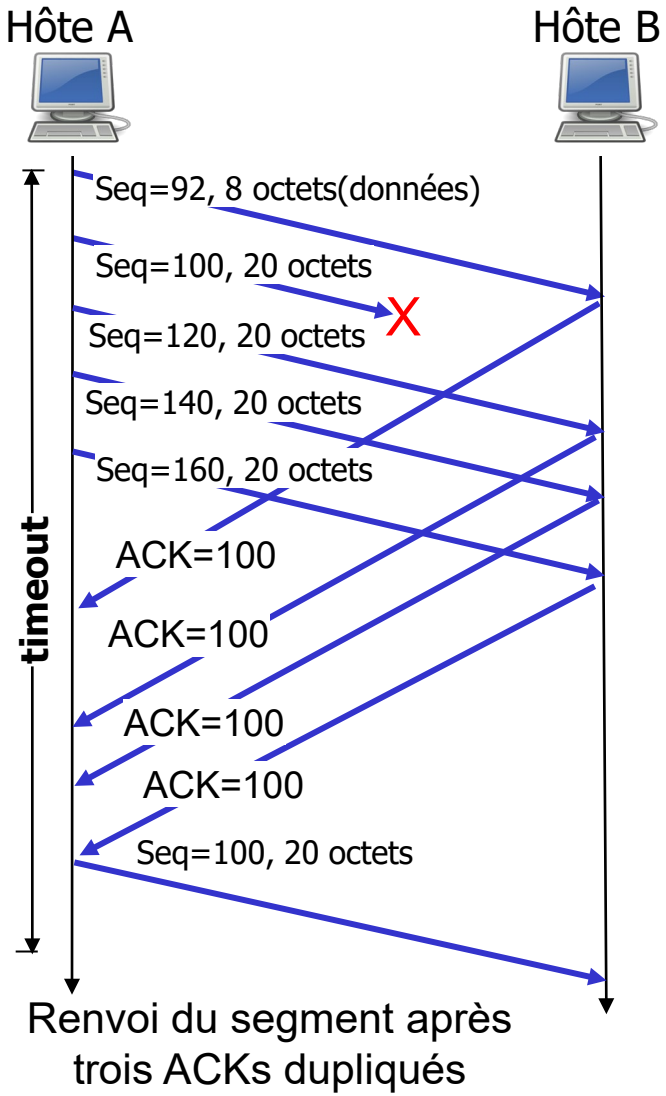
Q : Comment détecter plus rapidement les éventuelles pertes?

R : On peut détecter les segments perdus via les ACKs dupliqués.

- L'émetteur envoie souvent plusieurs segments consécutifs
- À l'arrivée d'un segment qui n'est pas dans l'ordre, le récepteur envoie un ack dupliqué (indiquant le # de séquence du prochain octet attendu (ACK))
- si un segment est perdu, il y aura plusieurs ACKs dupliqués.

Retransmission rapide de TCP

(TCP fast retransmit)



TCP fast retransmit

Si l'émetteur reçoit 3 ACKs dupliqués pour les mêmes données, il renvoie le segment non acquitté ayant le plus petit numéro de séquence

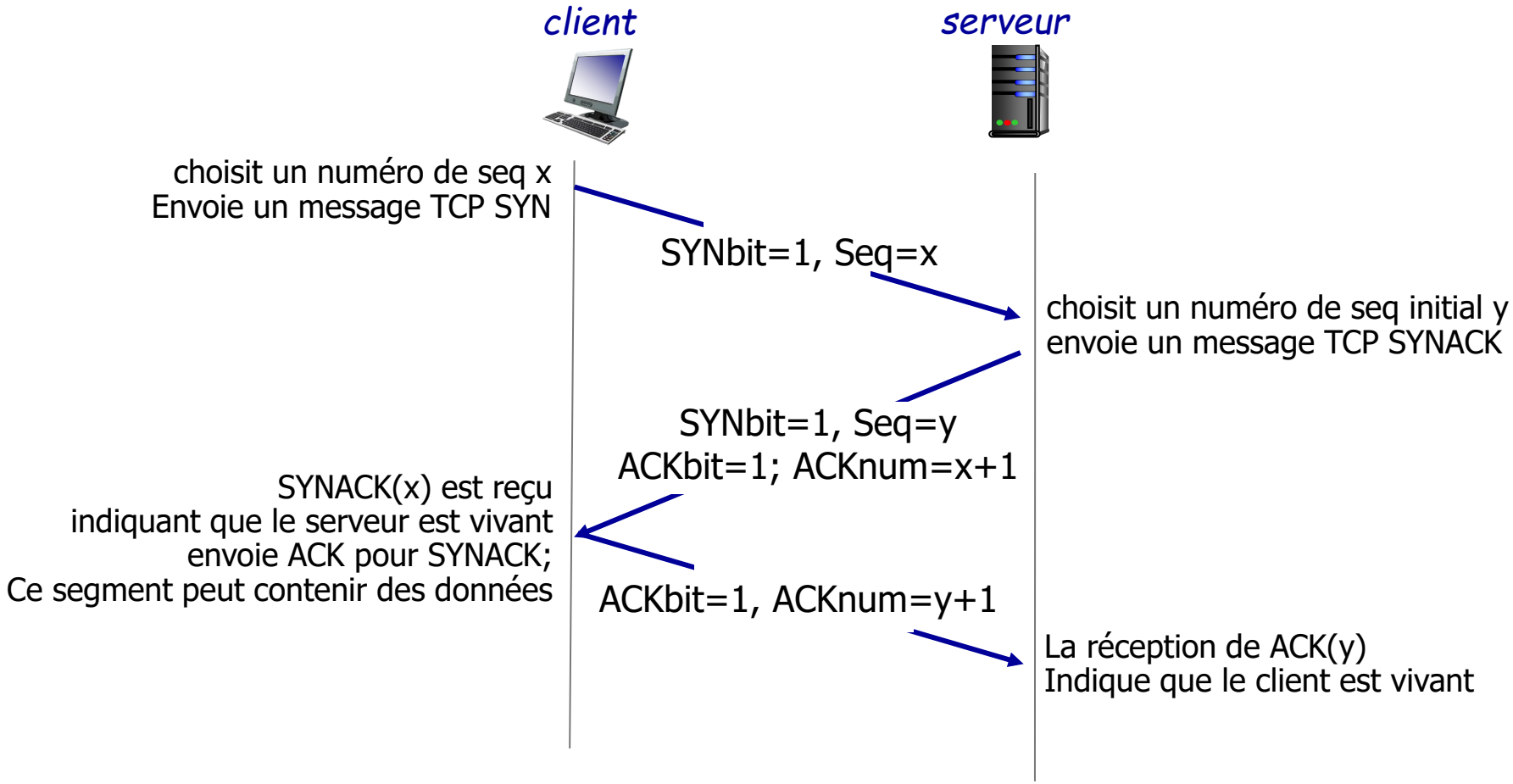
Probablement, ce segment a été perdu donc ce n'est pas la peine d'attendre l'expiration du timeout

Chapitre 3 : la couche transport

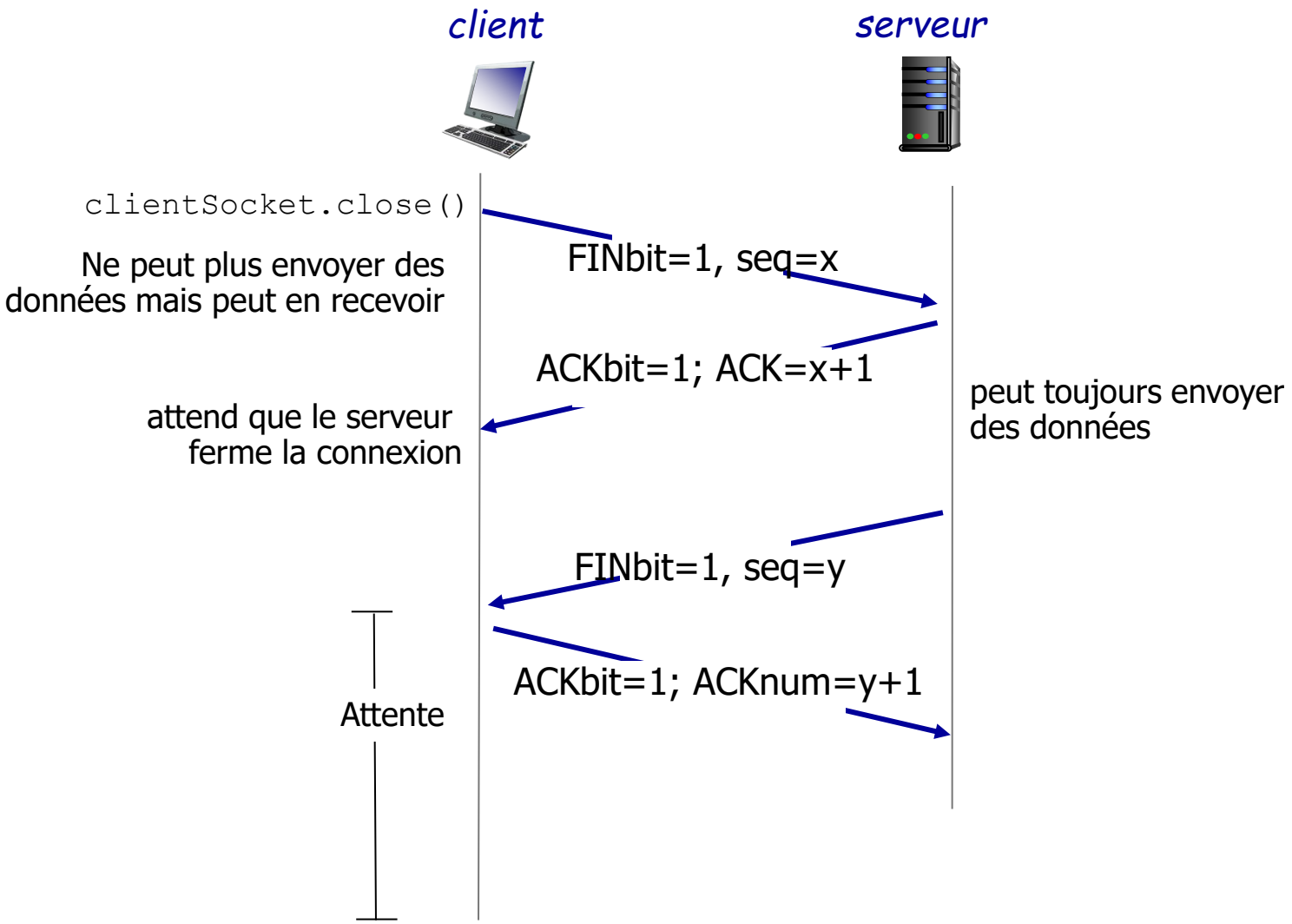
1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

Ouverture d'une connexion TCP

TCP 3-way handshake



Fermeture d'une connexion TCP



Chapitre 3 : la couche transport

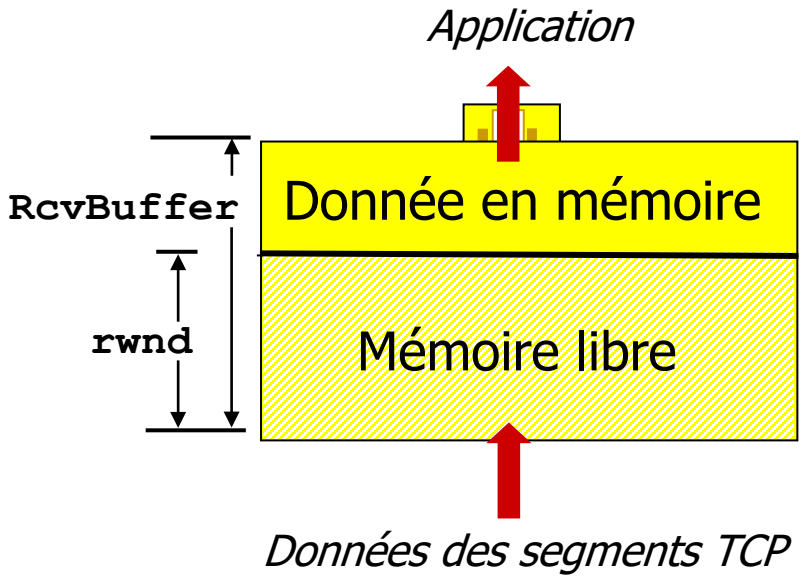
1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

Contrôle de flux TCP

Contrôle de flux

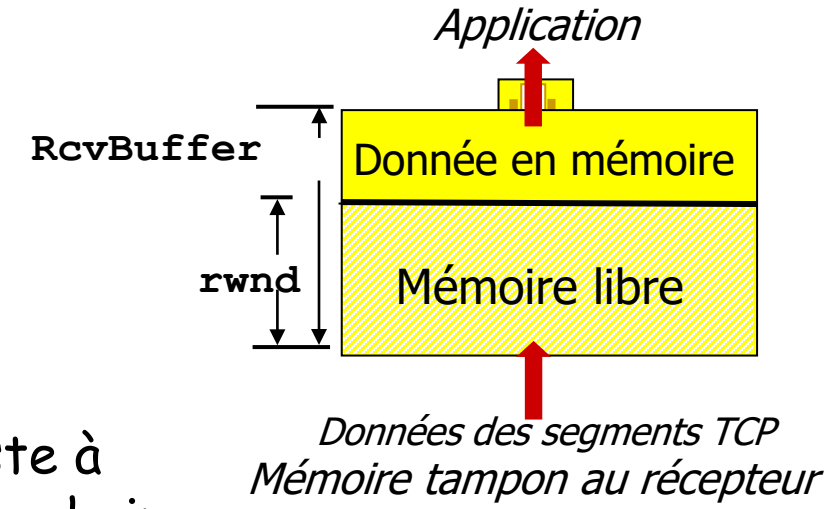
Adapter le débit de transmission de l'émetteur à la capacité du récepteur.
➔ éviter que la mémoire tampon du récepteur déborde

- Contrôle de flux dans TCP
  - Le récepteur annonce l'espace libre en incluant une valeur **`rwnd`** dans les segments
  - L'émetteur limite les données non acquittées à **`rwnd`**



Mémoire tampon au récepteur

Contrôle de flux TCP



Un hôte A échange des données avec un hôte B.

a) Au moment où l'hôte B signale à l'hôte A un *rwnd* de 200, il reste à l'hôte A 500 octets à envoyer. Quelle est la taille maximale du prochain segment que l'hôte A peut envoyer à l'hôte B?
Réponse:

b) Après le segment précédent, l'hôte B signale à l'hôte A un *rwnd* de 0.
Quelle est la taille maximale du prochain segment que l'hôte A peut envoyer à l'hôte B?
Réponse:

c) Après quelques minutes, l'hôte B signale à l'hôte A un *rwnd* de 400.
Quelle est la taille maximale du prochain segment de l'hôte A vers l'hôte B?
Réponse:

Chapitre 3 : la couche transport

1. Les services de la couche transport
2. Multiplexage et démultiplexage
3. Transport sans connexion: UDP
5. Transport orienté connexion: TCP
   - Structure d'un segment TCP
   - Fiabilité de transmission dans TCP
   - Gestion d'une connexion TCP
   - Contrôle de flux dans TCP
   - Contrôle de congestion dans TCP

Congestion des réseaux

- La quantité de paquets transmise dans le réseau dépassent sa capacité
- Les files d'attente dans les routeurs deviennent surchargées
  - Perte de paquets
  - Longs délais (dus aux longues attentes dans les routeurs)

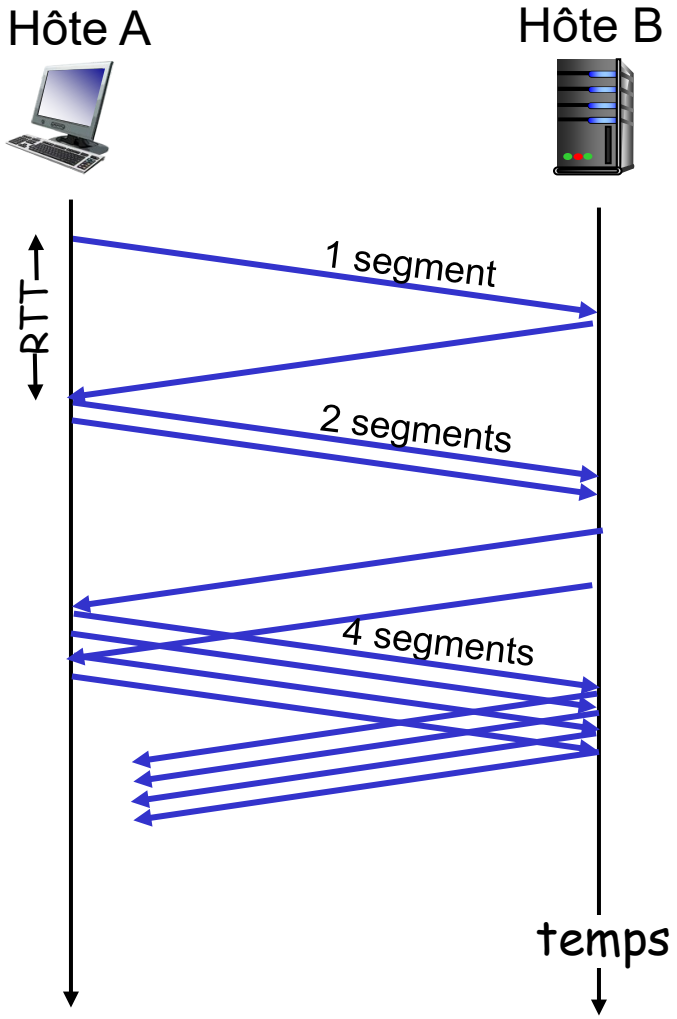
Contrôle de congestion pour TCP

- Contrôle de congestion
  - Contrôler la quantité de données envoyée dans le réseau dans le but de laisser diminuer la congestion ou d'éviter de l'empirer!
  - C'est différent du contrôle de flux !
- Difficulté : Aucune information de la part du réseau
  - La congestion peut être détectée par les hôtes en considérant les pertes et les délais
- Contrôle de congestion pour TCP :
  - Tester graduellement le réseau: *Slow start* + *Congestion avoidance*
  - Réagir aux pertes

TCP Slow Start

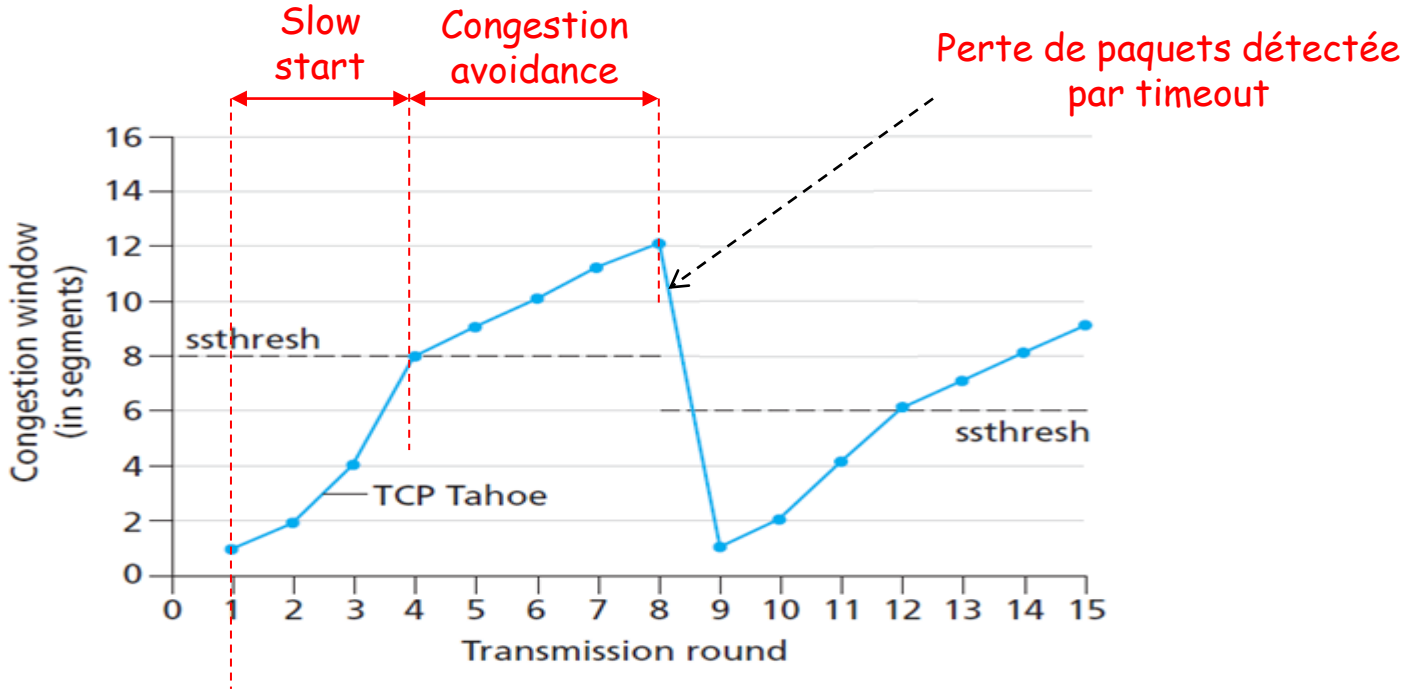
- Au début d'une connexion, le débit est augmenté exponentiellement jusqu'à la première perte:
  - Initialement cwnd= 1 MSS
  - cwnd double à chaque RTT
  - Équivalent à incrémenter **`cwnd pour`** chaque ACK reçu

- Le débit initial est faible mais augmente exponentiellement



Fenêtre de congestion (*congestion window - cwnd*) :
C'est le nombre de segments qu'on peut envoyer sans attendre l'acquittement

TCP : Congestion Avoidance



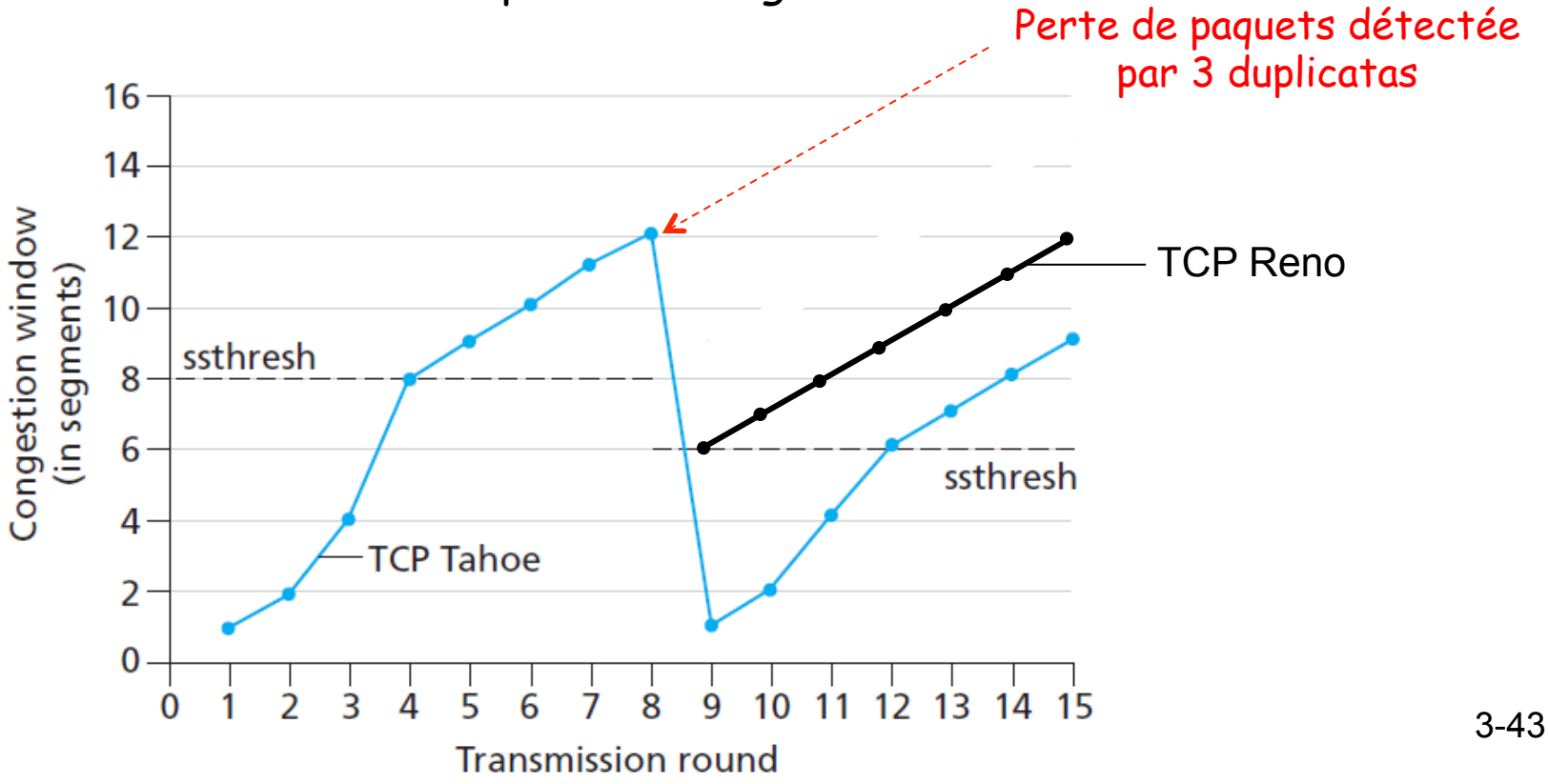
- Le débit augmente linéairement: cwnd est incrémenté de 1 à chaque RTT.

Q: quand est ce qu'on passe d'une augmentation exponentielle à une augmentation linéaire (*congestion avoidance*)?

R: quand **cwnd** atteint le seuil ssthresh

TCP : Perte de paquets (2 versions)

- TCP Tahoe : si une perte est détectée ➔ retour au *slow start*
- TCP Reno : la réaction dépend de la façon avec laquelle la perte a été détectée
- timeout indique un scénario de congestion alarmant➔retour au *slow start*
- 3 ACKs dupliqués indique que le réseau est capable de livrer quelques segments ➔ On réduit le débit et on passe au *congestion avoidance*



Résumé: contrôle de congestion TCP

- Quand **cwnd** est inférieur au seuil, l'émetteur est en *slow start*, la fenêtre augmente exponentiellement (x2 chaque RTT).
- Quand **cwnd** atteint le seuil (**ssthresh**), l'émetteur passe en phase *congestion avoidance*, la fenêtre augmente linéairement (+1 chaque RTT).
- Si une perte est détectée par timeout : le seuil est fixé à **cwnd/2** et **cwnd** est remis à 1 MSS. On revient à slow-start.
- Si une perte est détectée par 3 ACK dupliqués :
  - TCP Tahoe : de même que pour les pertes détectées par timeout
  - TCP Reno: le seuil est fixé à **(cwnd/2)** et **cwnd** prend la valeur du seuil. L'émetteur passe en phase *congestion avoidance* et la fenêtre augmente linéairement.

Questions?