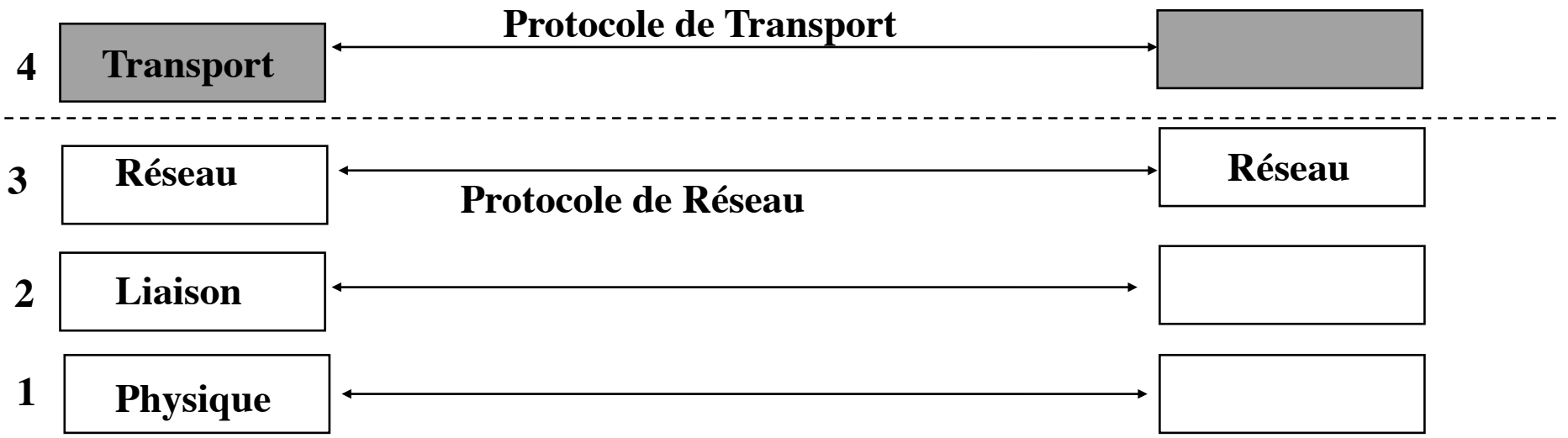
La couche Transport



Services de la couche transport

- Transfert fiable efficace, sûr et économique d'informations de bout en bout
- Dans l'idéal, répondant à des Qualités de Services (QoS) diverses
- Dernière couche avant les "applications"
- Les qualités de services nécessaires aux applications
  - Très variables
  - Exemples:
    - Transfert de fichier: Taux erreur nul, débit n'est pas primordial, le temps de transit non plus
    - Téléphone: Taux d'erreur peut être non nul, débit minimum indispensable, latence maximale (< 0,25 s)

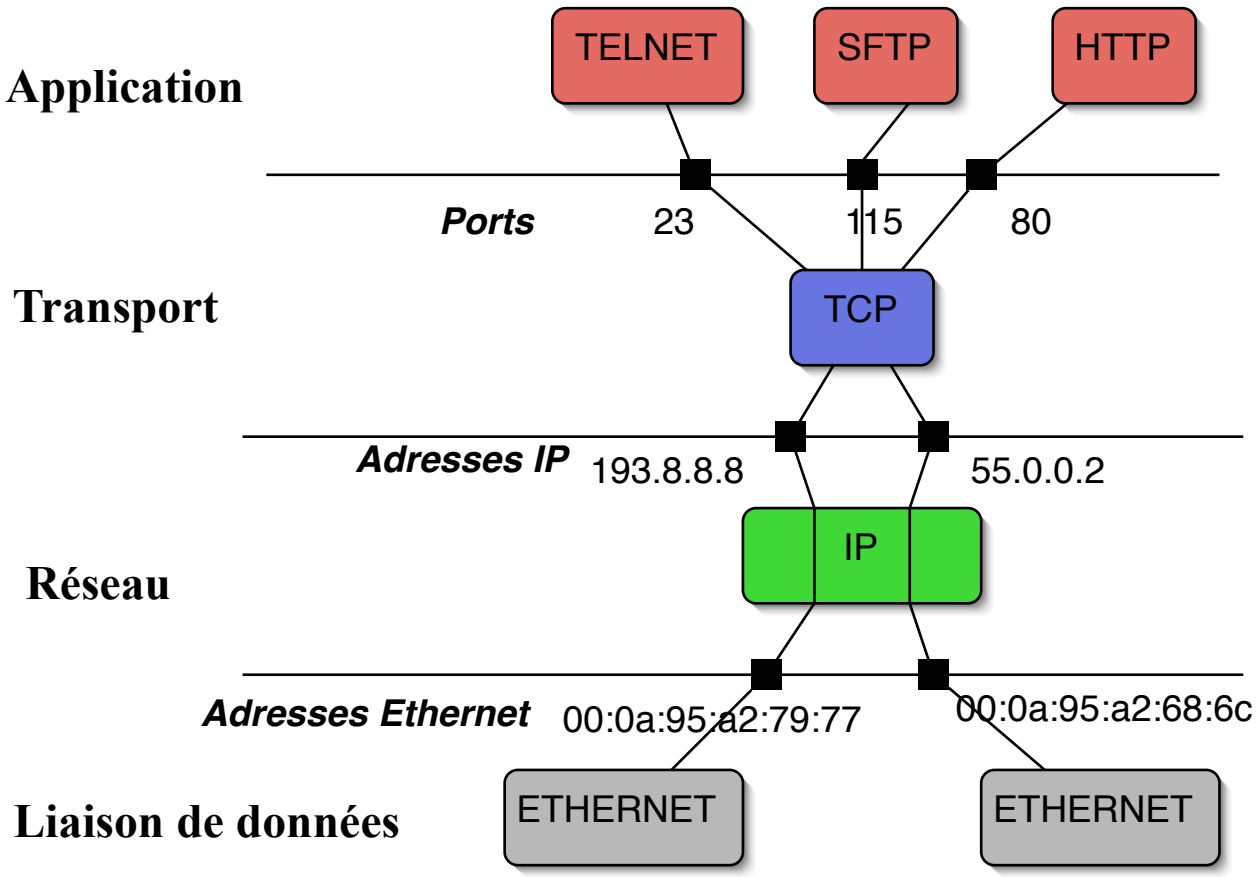
Les services de la couche transport d'Internet

- Rappel: IP sans connexion non fiable, déséquencement possible, délai de transfert très variable
- Protocoles pensés au départ pour le transfert de fichier et l'utilisation de machine à distance
- Deux types de services suivant les besoins de l'application à développer
  - Sans connexion, aucune QoS: User Datagram Protocol (UDP)
  - Avec connexion: Transport Control Protocol (TCP)
    - » Ouverture et fermeture de connexion
    - » Fragmentation et reassemblage
    - » Contrôle de flux et récupération des erreurs
- TCP et UDP se sont imposés naturellement. Les normes OSI ont été définies en parallèle mais elles sont arrivées trop tard
- TCP garantie seulement un taux de perte nul
- Dans les normes OSI, il existe différents niveaux de QoS pouvant être utilisés suivant le réseau sous-jacent

Fonctionnalités des protocoles transport d'Internet

- **Protocoles client/serveur:**
  - » Le serveur se met en attente de demandes
  - » Le client initie le dialogue par une demande
- Interface des "**sockets**": librairies de primitives d'accès aux protocoles transport TCP et UDP
- 1 serveur : 1 **numéro de port fixé**
  - Côté serveur: réservés pour applications standards
    - Fichier /etc/services
    - Exemple: HTTP port 80 en TCP
  - Côté client: alloués dynamiquement

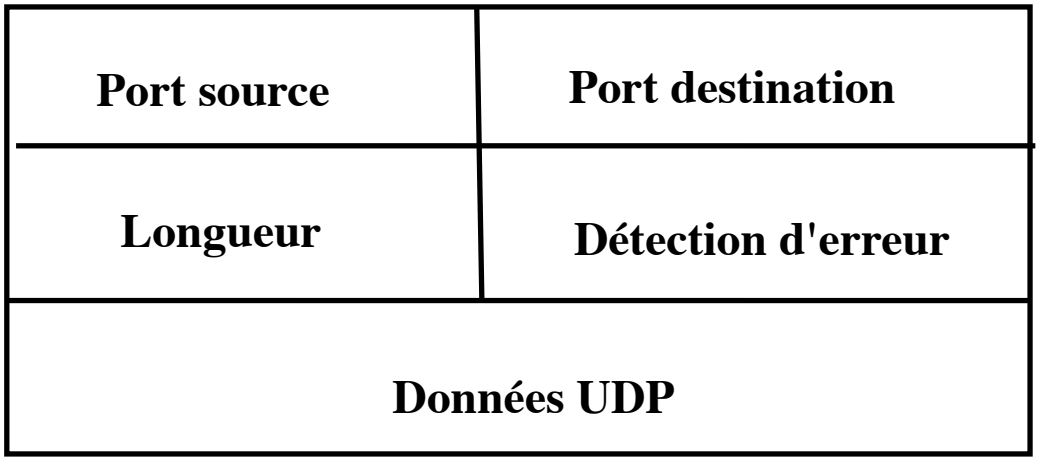
Multiplexage vers les applications



- Connexion définie par les Adresses Internet source et destination et les numéros de ports source et destination
- Informations se trouvant dans les entêtes réseau (IP) et transport (TCP ou UDP)

Les services du protocole UDP

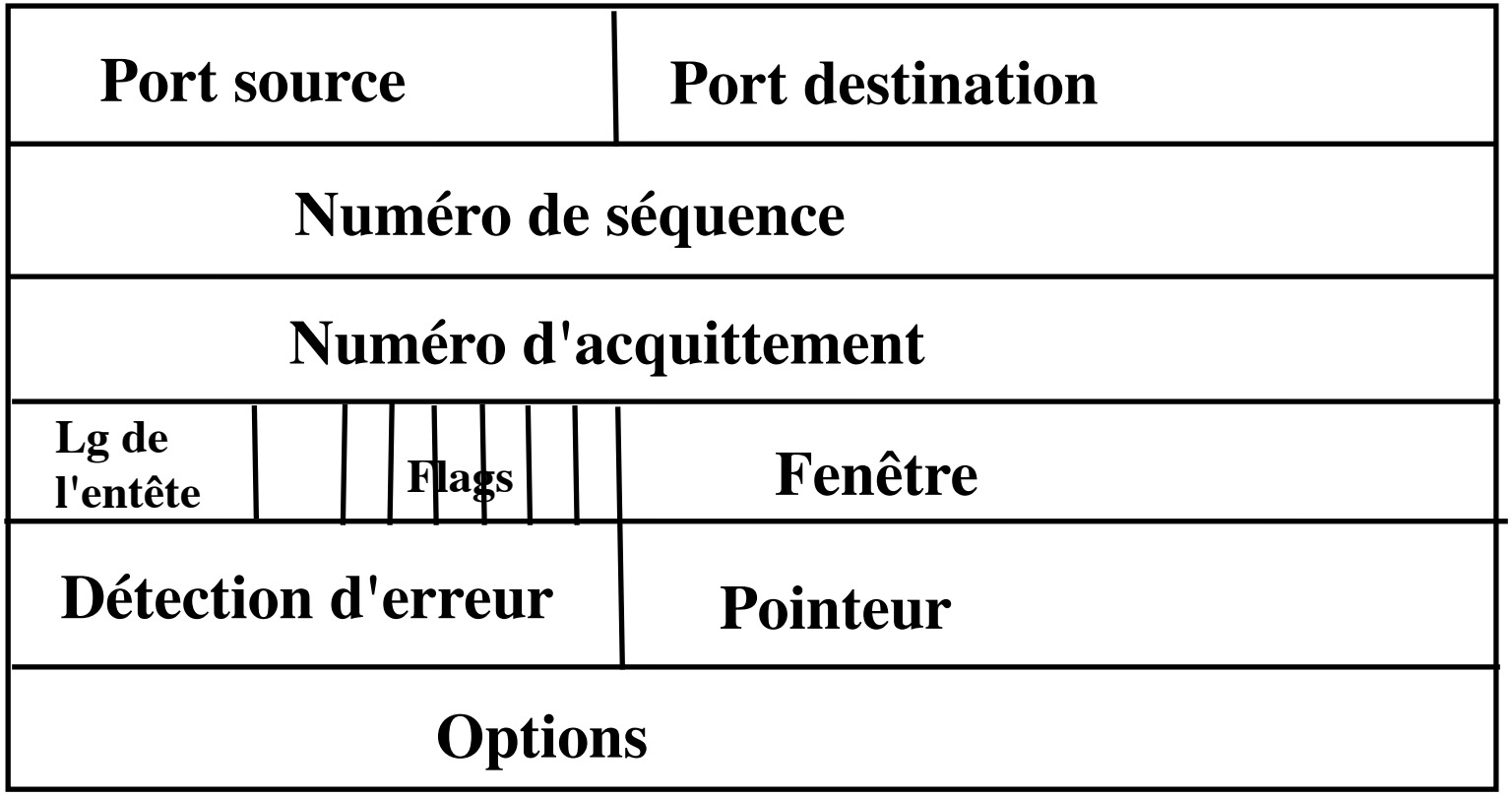
- Mode sans connexion (Datagram)
- Multiplexage vers les applications
- Aucun contrôle de flux et de récupération d'erreur
- **Entête UDP:**
  - Numéros de port (sur 2 octets)
  - Longueur en nombre d'octets (sur 2 octets)
  - Détection d'erreur optionnelle (par "checksum")



Services du protocole TCP

- Service orienté connexion
  - Ouverture et libération de la connexion
- Segmentation et re-assemblage des messages
  - Pas de notion de paquets de donnée à la réception : flux d'octets
- Rétablir l'ordre des paquets
  - Numérotation des octets de données
- Multiplexage vers plusieurs applications (numéro de port)
- Transfert de données exprès
- Contrôle de flux
- Détection des paquets erronés ou perdus
- Récupération des erreurs par re-émission
- Détection d'inactivité

Le format du paquet TCP



- Flags: Urgent, Ack, Psh, Rst, Syn, Fin

Connexion de niveau Transport

- Trois phases de communication :

  1- établissement ou ouverture de la connexion

  - permet de savoir si l'entité distante est prête
  - échange possible des paramètres de connexion

  2- la communication proprement dite (échange des données)

  3- la rupture ou fermeture de la connexion

- Facilite certains services comme le contrôle de flux et la récupération des erreurs

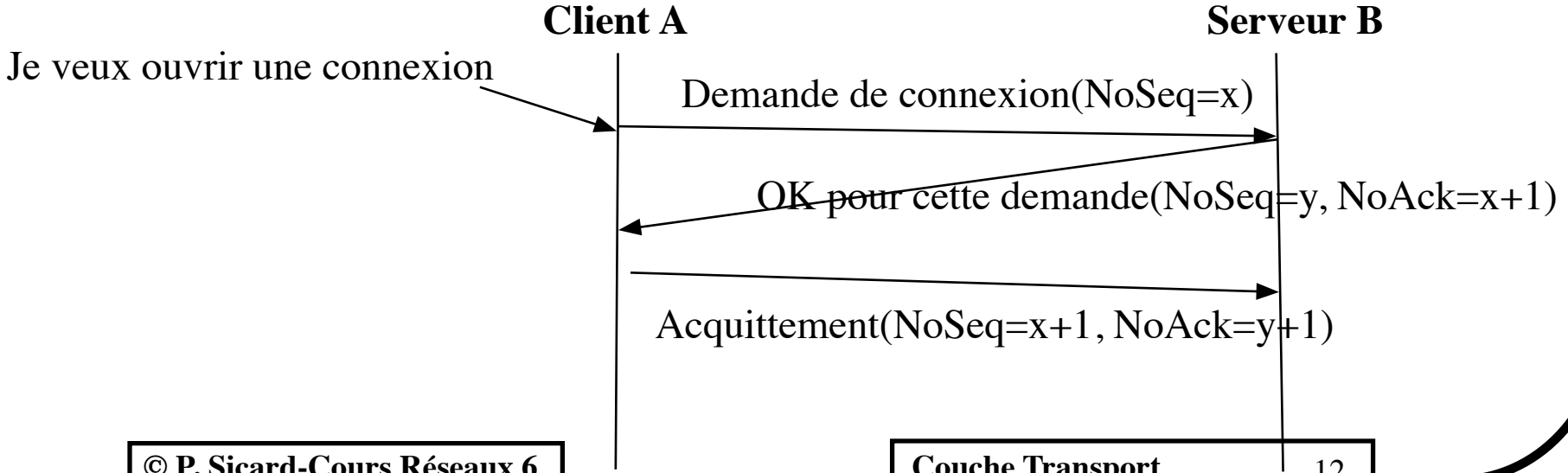
L'établissement d'une connexion

- **Problème :**
  - Etablir une connexion virtuelle entre deux entités
  - Une fois la connexion établie, l'échange des paquets de données est bidirectionnel entre les deux entités
  - Comment définir une connexion sans ambiguïté ?
  - Comment associer une donnée à une connexion ?
- **Rappel**
  - Le niveau transport dans Internet est mis en œuvre au dessus d'une couche réseau non fiable
    - Perte de paquet
    - Retardement de paquets (doublons potentiels)
    - Deséquencement des paquets

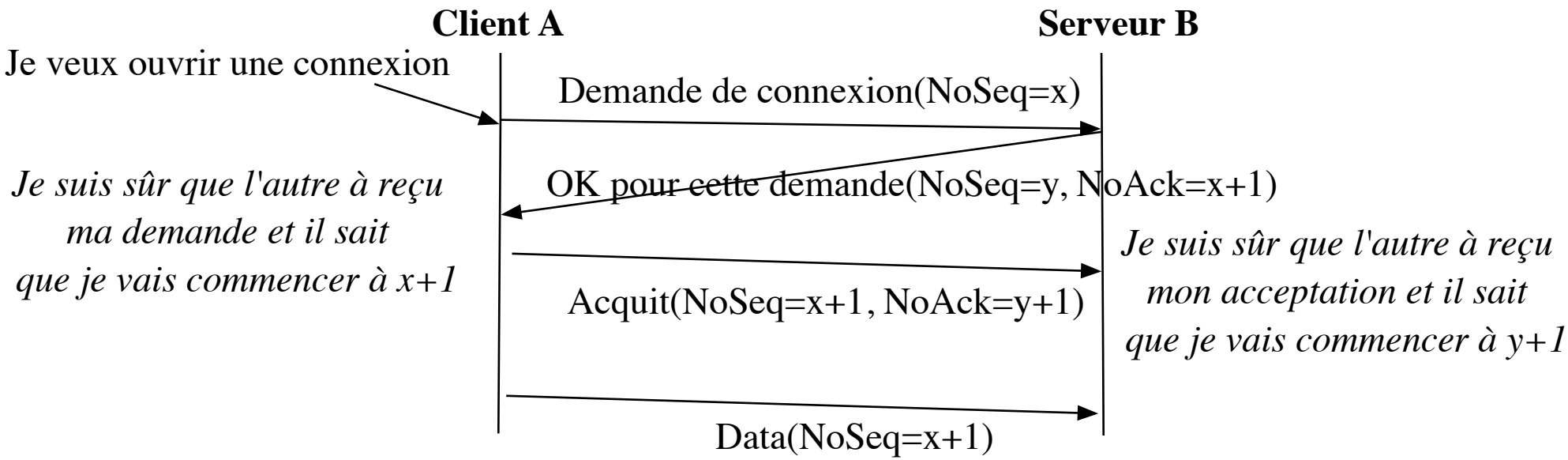
Mécanisme d'établissement de connexion

- Les Numéros de séquences permettent d'identifier les demandes de connexions
- Ils vont aussi servir ensuite à numéroter les octets de données dans le mécanisme de récupération d'erreur
- L'ouverture en 3 étapes est nécessaire afin de garantir au client et au serveur que le premier numéro de séquence utilisé soit connu de l'autre

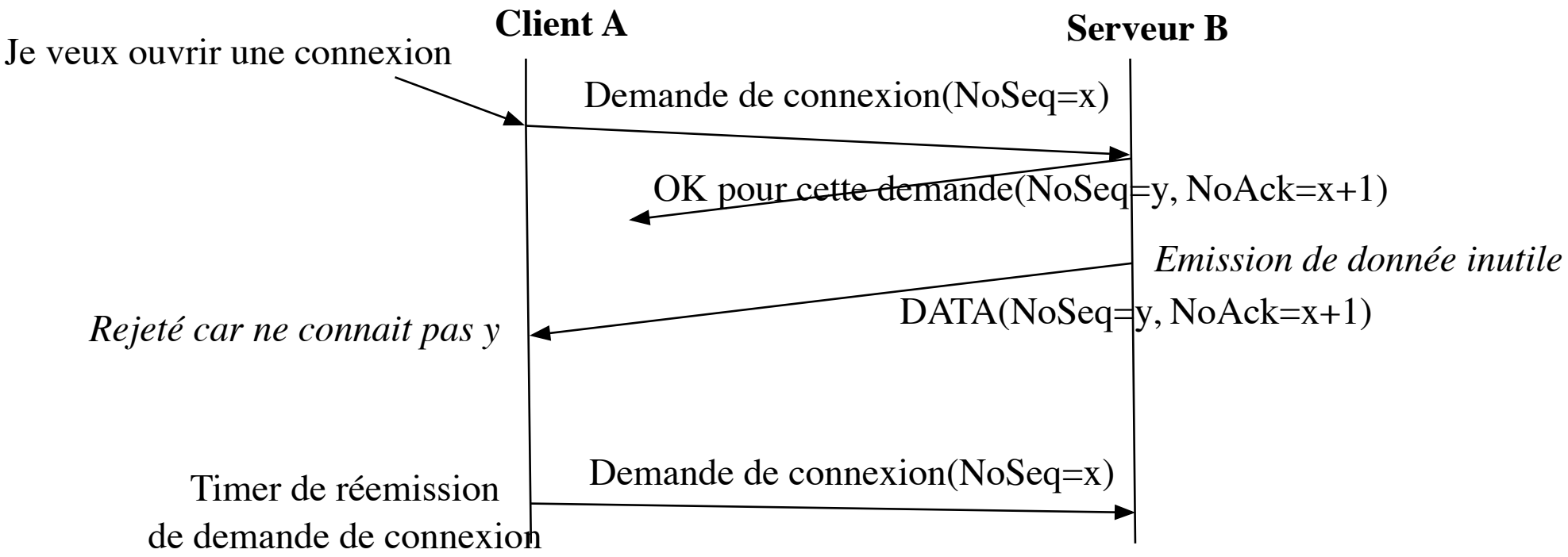
- **Une méthode d'ouverture de connexion fiable (Ray Tomlinson 1975):**
  - Une connexion est identifiée par deux références uniques choisies respectivement par chacune des entités de transport
  - L'identification de la connexion est établie au cours d'un protocole d'ouverture à trois phases :
    - » A envoie une demande de connexion portant un numéro de séquence NoSeq= x
    - » B répond par une acceptation de demande de connexion: (NoSeq= y, NoAck= x+1)
    - » A envoie un acquittement de l'acceptation de la demande de connexion: (NoSeq = x+1, NoAck = y+1)



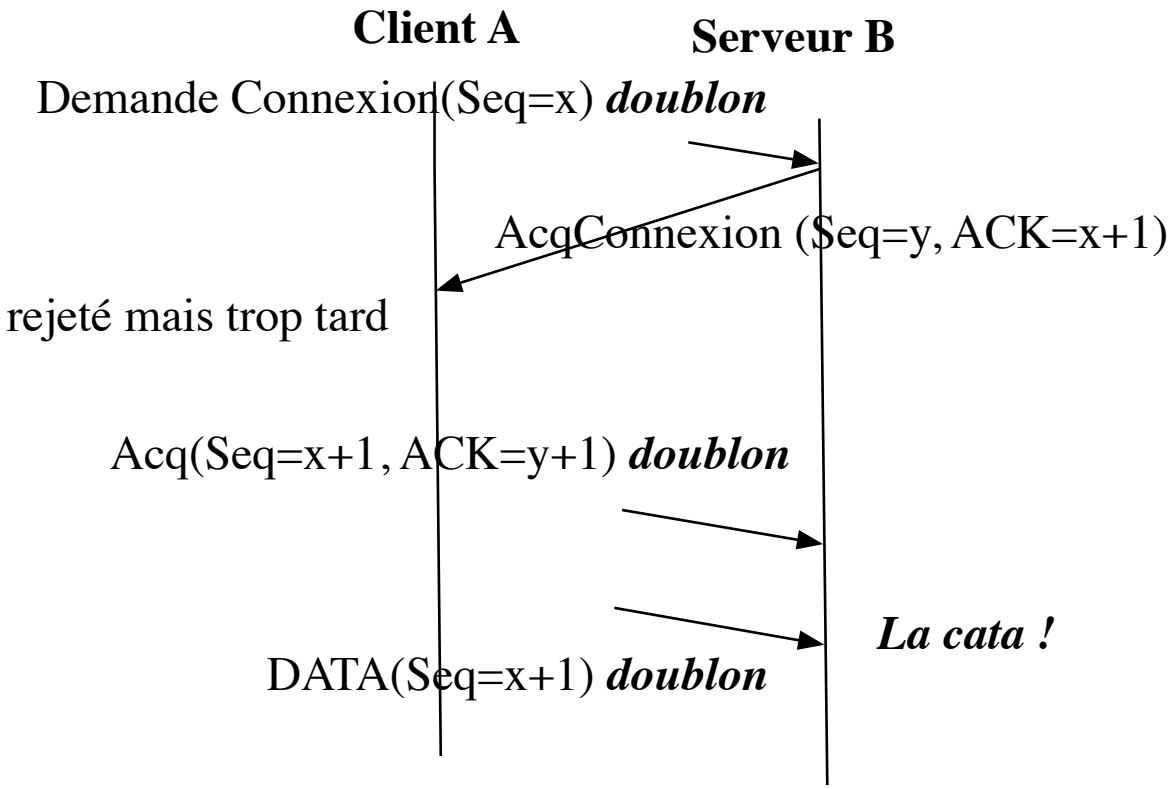
Trois étapes nécessaires



Exemple de problème avec une ouverture à 2 paquets



Exemple de mise en défaut avec vieux doublons



- Pour éviter de confondre des vieux doublons (ouvertures de connexion ou données), il faut ne pas réutiliser des numéros de séquence avant que tous les doublons potentiels portant ces numéros est disparus.

Calcul de numéro de séquence initiaux

Une solution aux vieux doublons :

- Les numéros de séquence initiaux sont calculés à partir d'une horloge système de période 4 microsecondes.
- Le champ numéro de séquence étant sur 4 octets, cela garantit l'unicité d'un identificateur de connexion pendant 4 microseconde * 2 puissance 32 ~ 4 heures
- On est sûr que tous les doublons portant des numéros en cours d'utilisation auront disparu.

Problème de la « zone interdite »

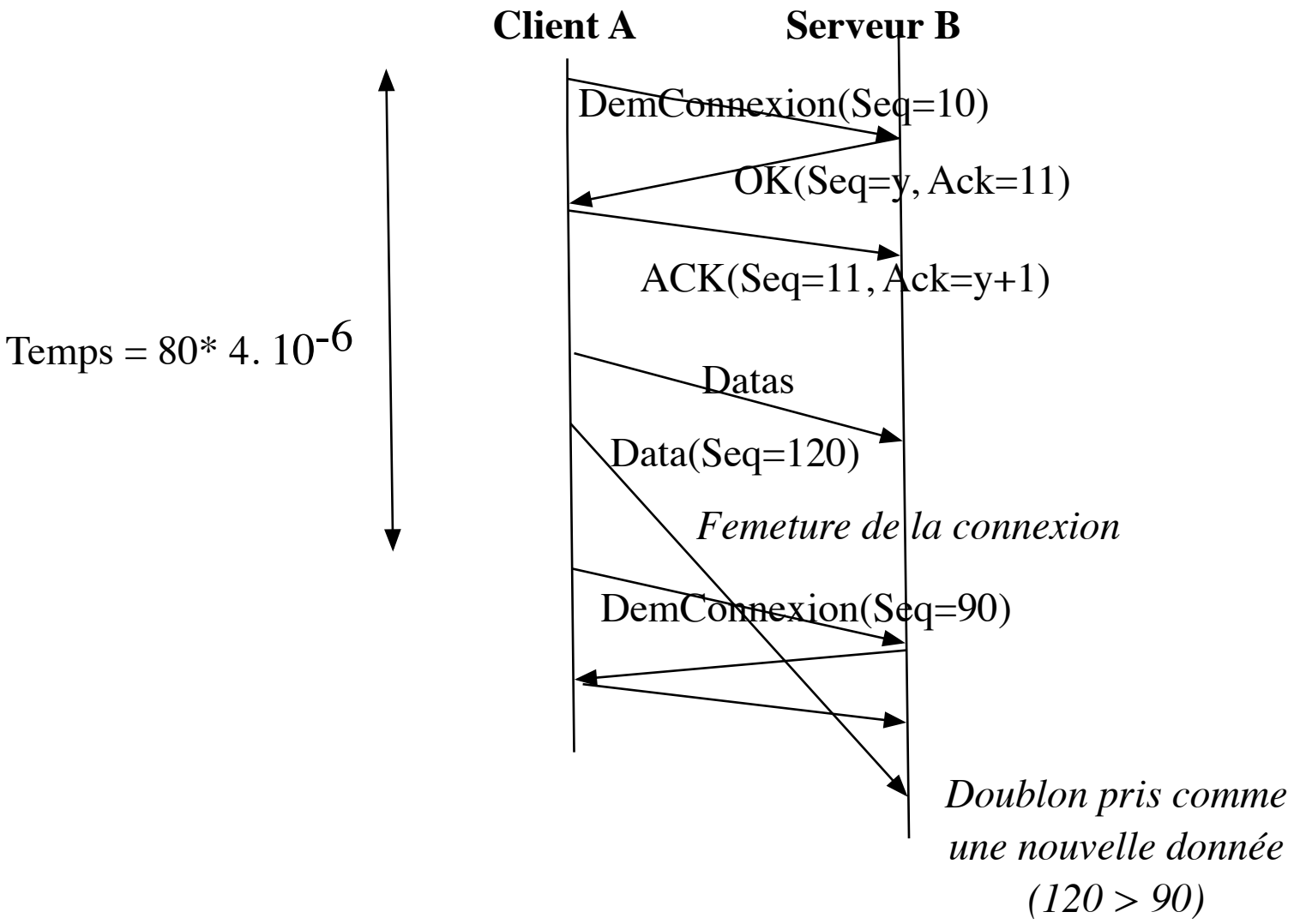
- Les numéros de séquences servent à numéroter aussi les octets de données
- **Problème: l'évolution des numéros de Séquence des données dépend du débit de la connexion**
  - On peut arriver à des cas où un doublon d'une ancienne connexion resurgisse avec un numéro de séquence suffisamment grand pour être pris comme une nouvelle donnée dans une autre connexion.
- **Solution** : on attend la durée de vie maximale d'un paquet estimée sur Internet avant de ré-ouvrir une connexion sur la même paire de ports (~2 minutes)

Zone interdite

- Débit permettant d'arriver dans une zone interdite

  1 octet/4.10 $^{-6}$ sec= 8 bits/4*10^{-6}= 2 Mégabit/s

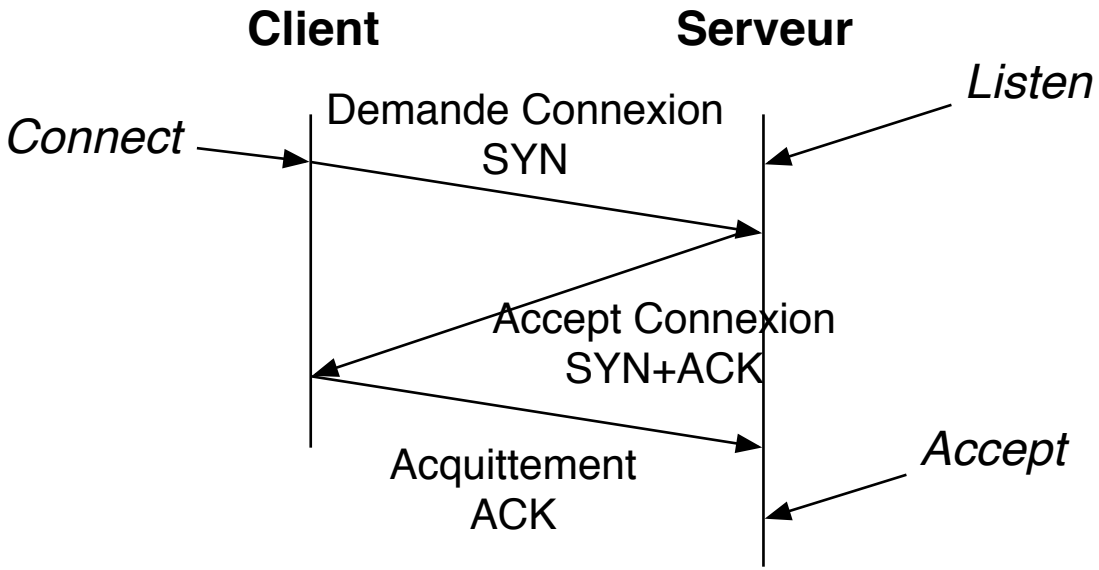
  On néglige le temps d'ouverture et de fermeture de la connexion



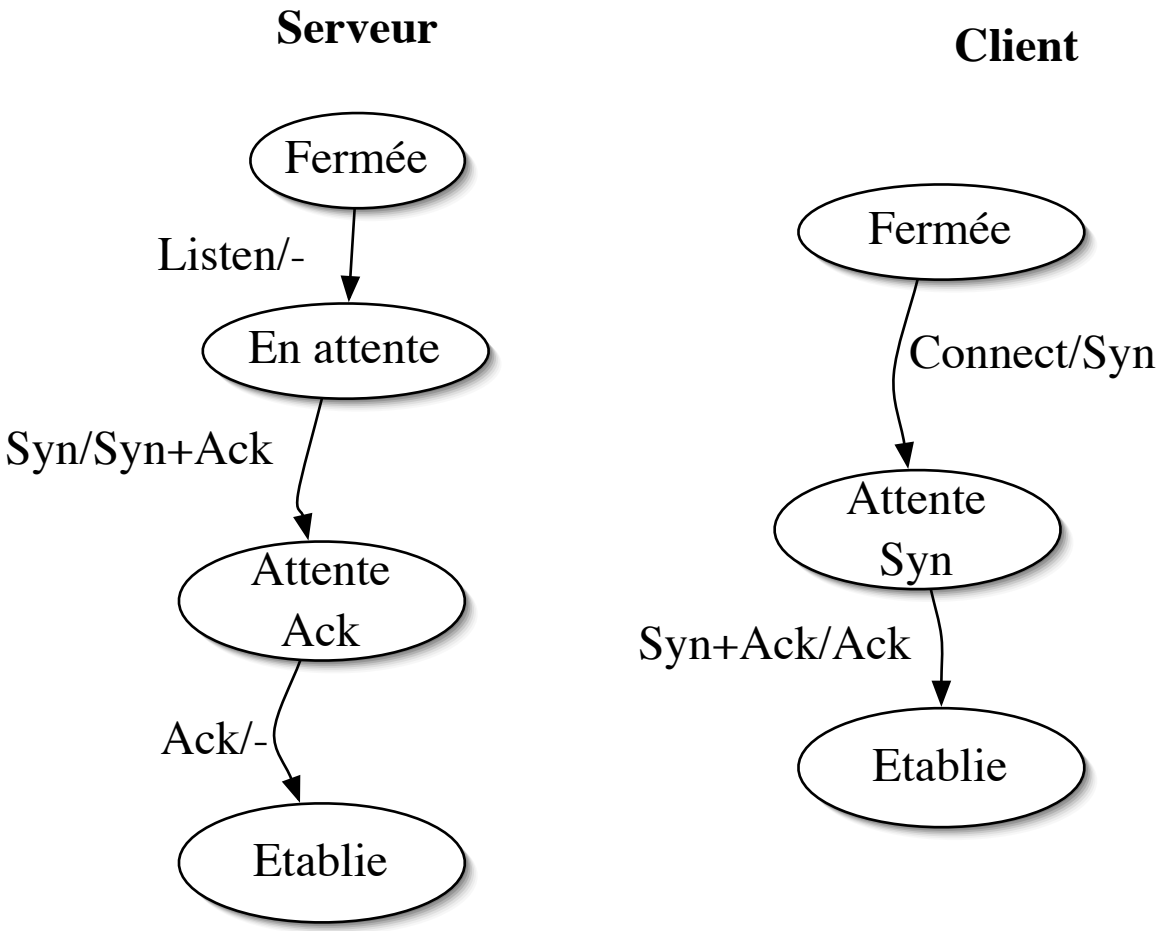
Etablissement d'une connexion TCP

- **Des paquets particuliers pour ouvrir la connexion**
  - Demande de connexion: Flag SYN=1 et ACK =0
  - Acceptation de connexion : Flags SYN= 1 et ACK=1
  - Acquittement: flag ACK
  - En cas d'incohérence (rejet d'une demande…) : flag RST

- **Ouverture**
  - **Protocole à trois phases:**



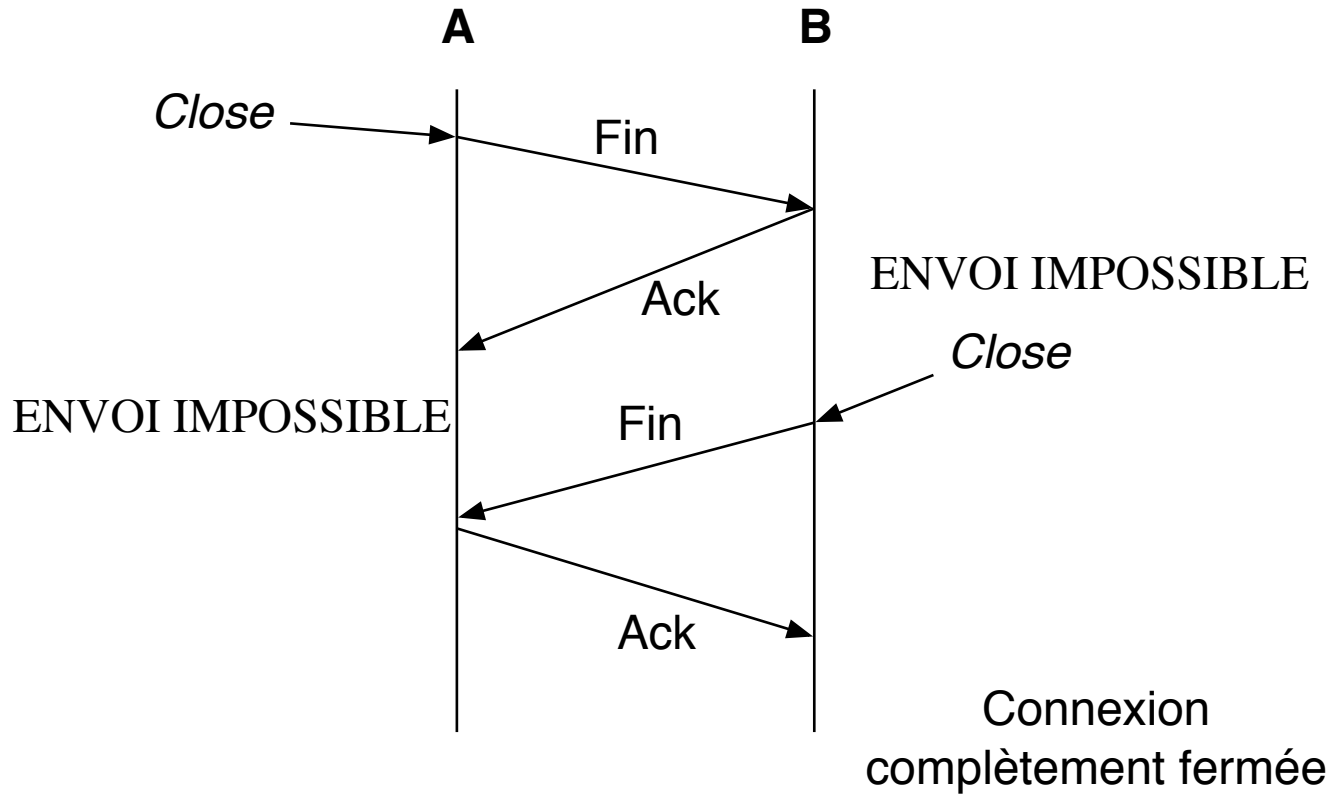
Automates de Mealy de l'ouverture de connexion



Fermeture de connexion

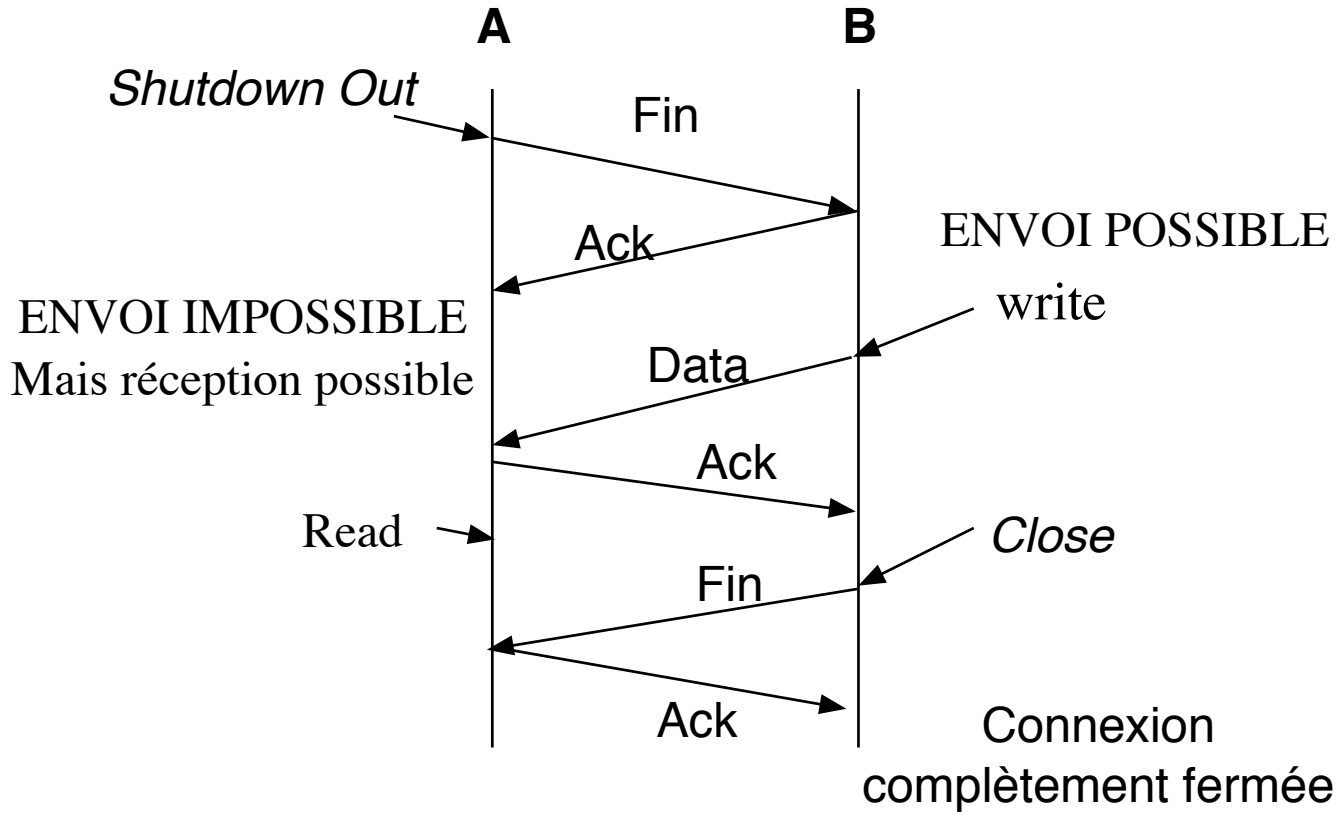
- **1ère solution:**
  - Fermeture brutale, quand on veut fermer la connexion d'un coté , on ne se soucie pas de l'état de l'autre. Comme au téléphone, on raccroche. Mais on peut s'être mis d'accord au niveau applicatif ("au revoir" avant de raccrocher)
  - Primitive *close* des Sockets
- **2ème solution**
  - On veut pouvoir libérer les ressources d'émissions (tampons) d'un côté
  - Solution: la connexion est gérée comme deux demi-connexions unidirectionnelles
  - On peut fermer en émission, et continuer à recevoir, si l'autre n'a pas fini d'émettre
  - **Primitive *shutdown* des sockets**

- Flag FIN pour signifier la demande de fermeture
- L'entité distante acquitte pour qu'il n'y ait pas d'ambiguité sur l'état de la connexion



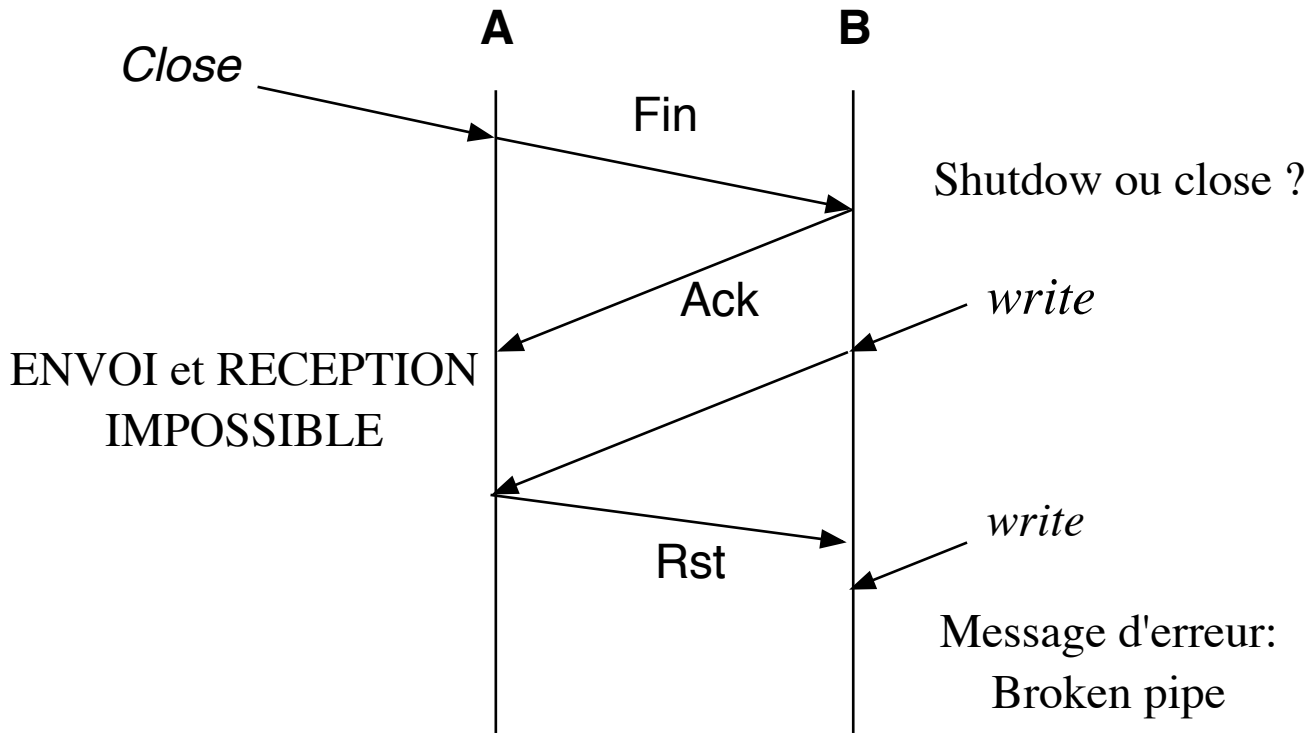
Fermeture de connexion
Le shutdown

- **Shutdown out, in, both**



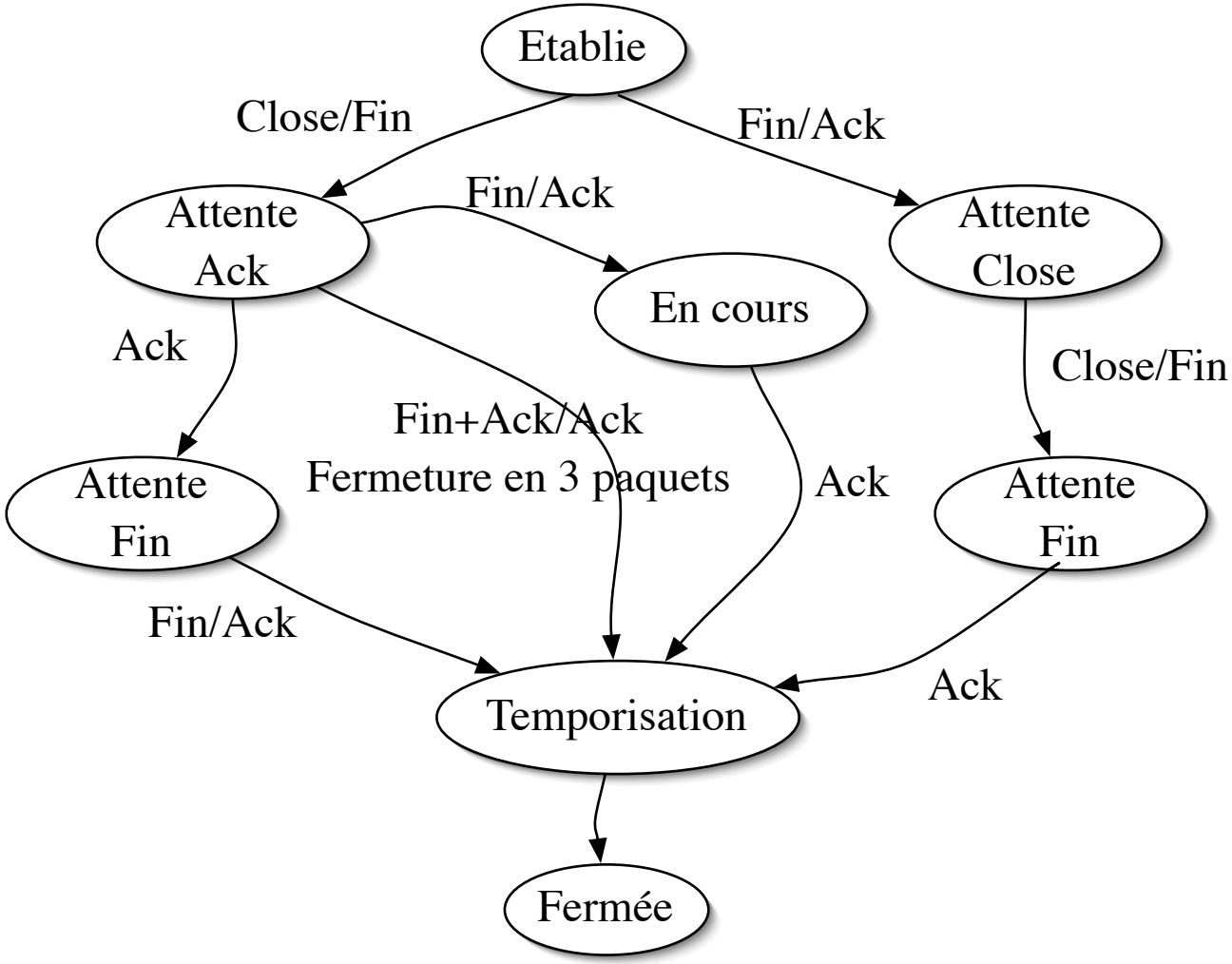
Fermeture de connexion
Un seul flag pour shutdown et close

- **Rien ne permet de différencier le shutdown et le close à la réception de Fin**



Automate de la fermeture

Serveur et client



Principe de la récupération d'erreur dans TCP

» **Fenêtre d'anticipation avec re-émission sélective et acquittements "cumulatifs" (voir chapitre Contrôle d'erreur)**

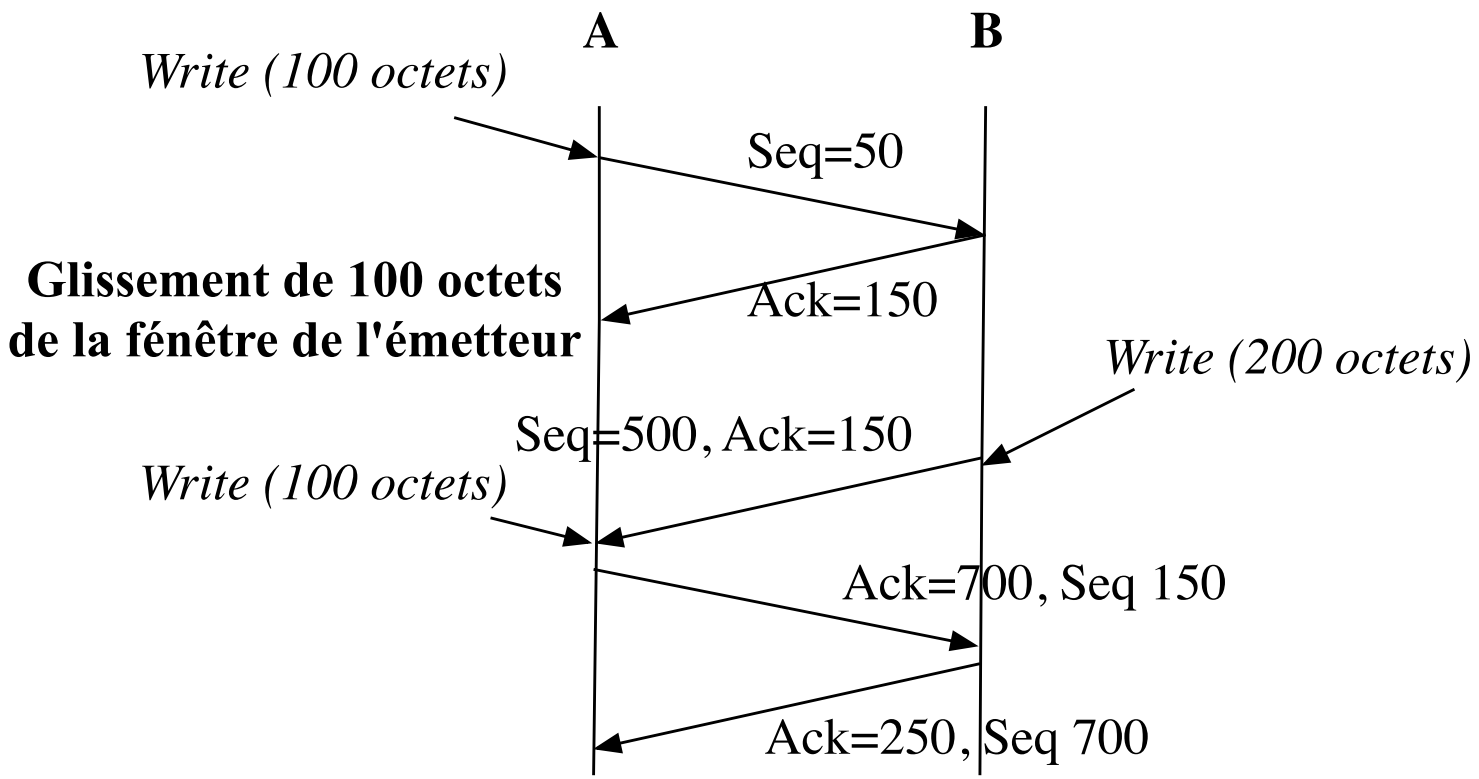
- Un timer par paquet, mémorisation côté émetteur des paquets jusqu'à qu'ils soient acquittés
- Un acquittement signifie "j'ai bien reçu jusqu'à ce numéro"
- Le récepteur mémorise les paquets non reçus en séquence
- Le timer de re-émission est calculé dynamiquement car la latence est très variable
- La taille du tampon nécessaire à la fenêtre d'anticipation de la récupération d'erreur est décidée par le programmeur côté émetteur
- La taille de ce tampon peut donc influer sur l'efficacité du protocole. Blocage fréquent en cas de tampon sous-dimensionné par rapport à la latence du réseau

Récupération d'erreur

- Flux d'octets (contrairement à UDP)
- Une fois la connexion ouverte, elle est symétrique et le transfert des données est bidirectionnel
- Numéro de séquence:
  - Numéro du 1er octet de donnée du message (sert aussi à la segmentation et réassemblage)
- Numéro d'acquittement:
  - Numéro de séquence + 1 du dernier octet bien arrivé (dans l'autre sens) (flag Ack=1)
- Un paquet peut servir à transporter des données et à acquitter un paquet du flux de sens inverse

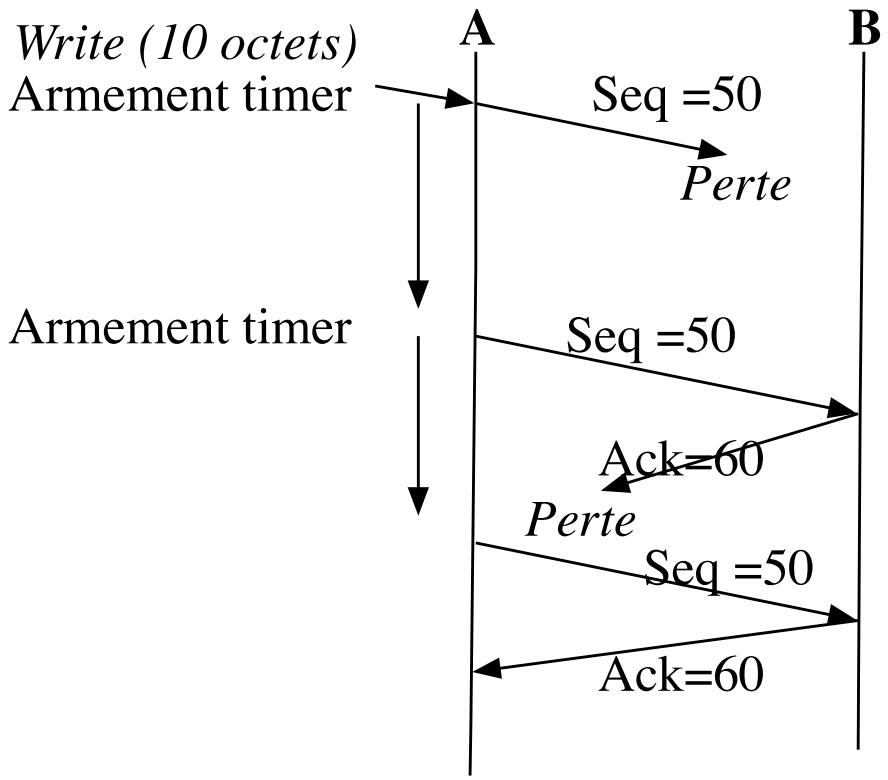
Exemple 1 de récupération d'erreur

- Le deuxième et le dernier paquet sont des acquittements "purs" mais il porte malgré tout un numéro de séquence
- Le quatrième paquet est un paquet de donnée qui sert aussi d'acquittement

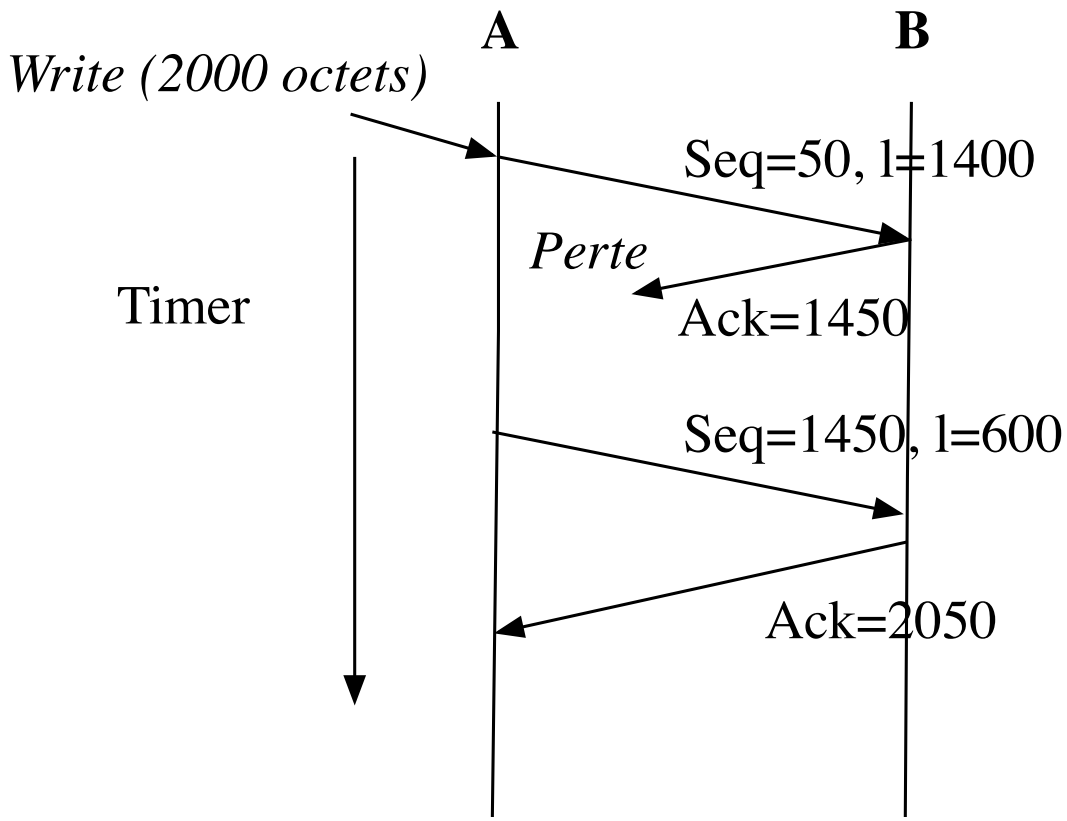


Exemple 2 de récupération d'erreur

- La perte d'un paquet de donnée déclenche la re-émission
- De même la perte d'un acquittement déclenche une re-émission



Exemple 3 de récupération d'erreur



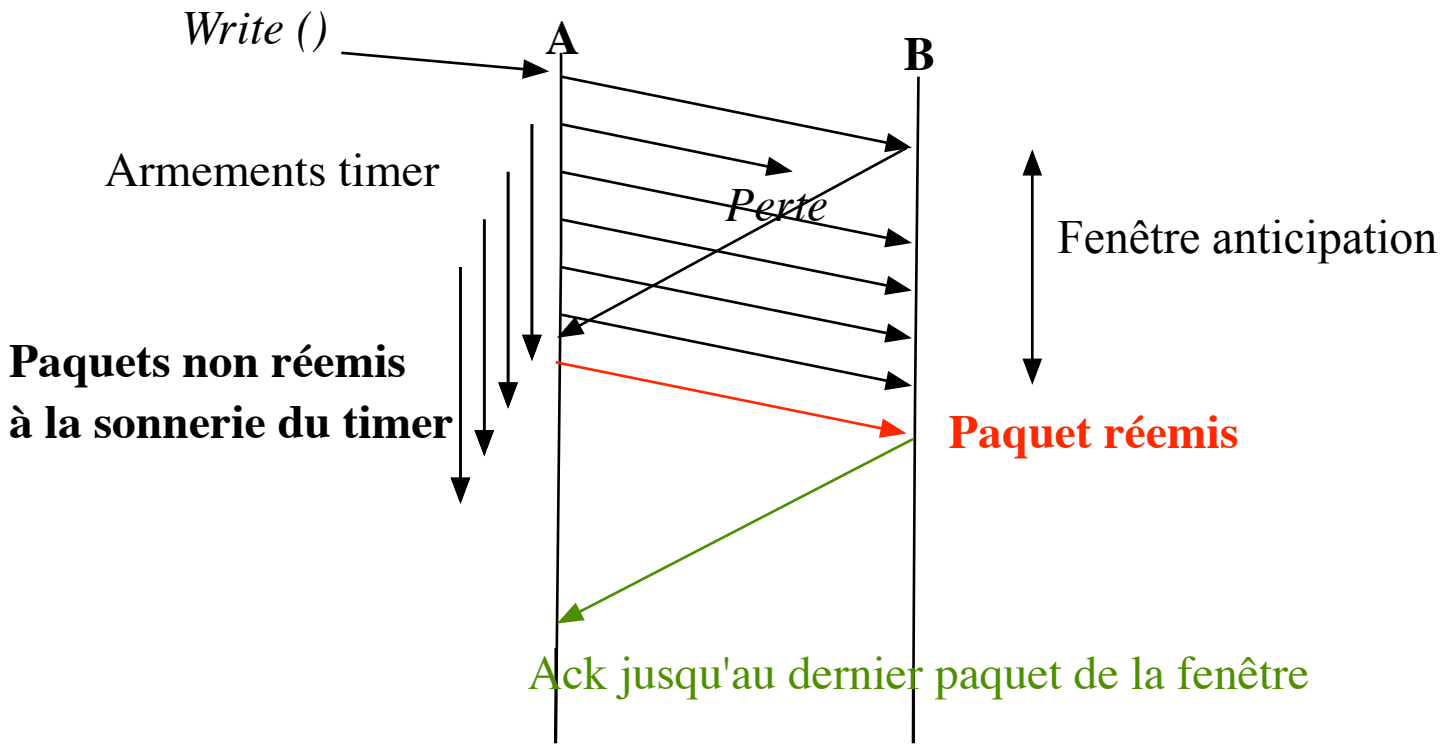
- Le deuxième acquittement acquitte les deux paquets de données précédent, il n'y a pas de re-émission à la sonnerie du timer (intérêt de l'acquittement "cumulatif")

Mécanisme de la récupération d'erreur

- **Côté récepteur:**
  - Si Numéro de séquence attendu alors acquittement jusqu'au dernier reçu
  - Sinon : mémorisation du paquet dans un tampon et acquittement jusqu'au dernier reçu sans « trou »
  - un Acquittement est envoyé à chaque réception d'un paquet (No du dernier reçu en séquence)

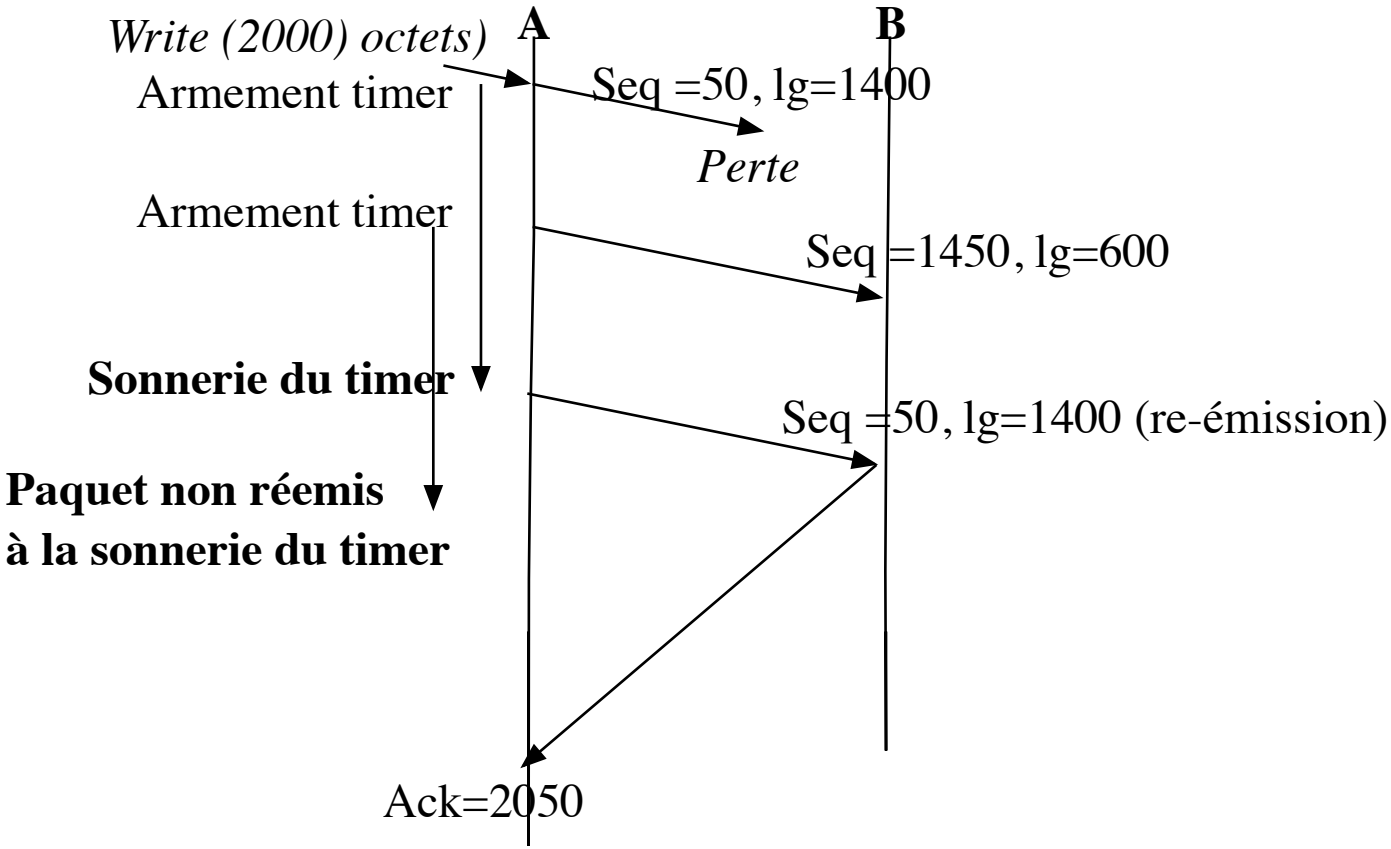
- **Côté émetteur:**
  - Timer associé à un paquet et mémorisation dans un tampon des paquets émis en attente d'acquittement
  - Si un timer sonne : re-émission du paquet de donnée associé
  - A la réception d'un acquittement : arrêt du timer
  - Attention: Un seul paquet en cours de re-émission (pour éviter les re-émissions inutiles)

Limitation des re-émissions



- Pour éviter les re-émissions inutiles dues à une perte dans la fenêtre à anticipation
- Un seul paquet est re-émis à l'expiration d'un timer
- Le dernier acquittement acquitte tous les paquets bien reçus en séquence

Exemple 4 de récupération d'erreur

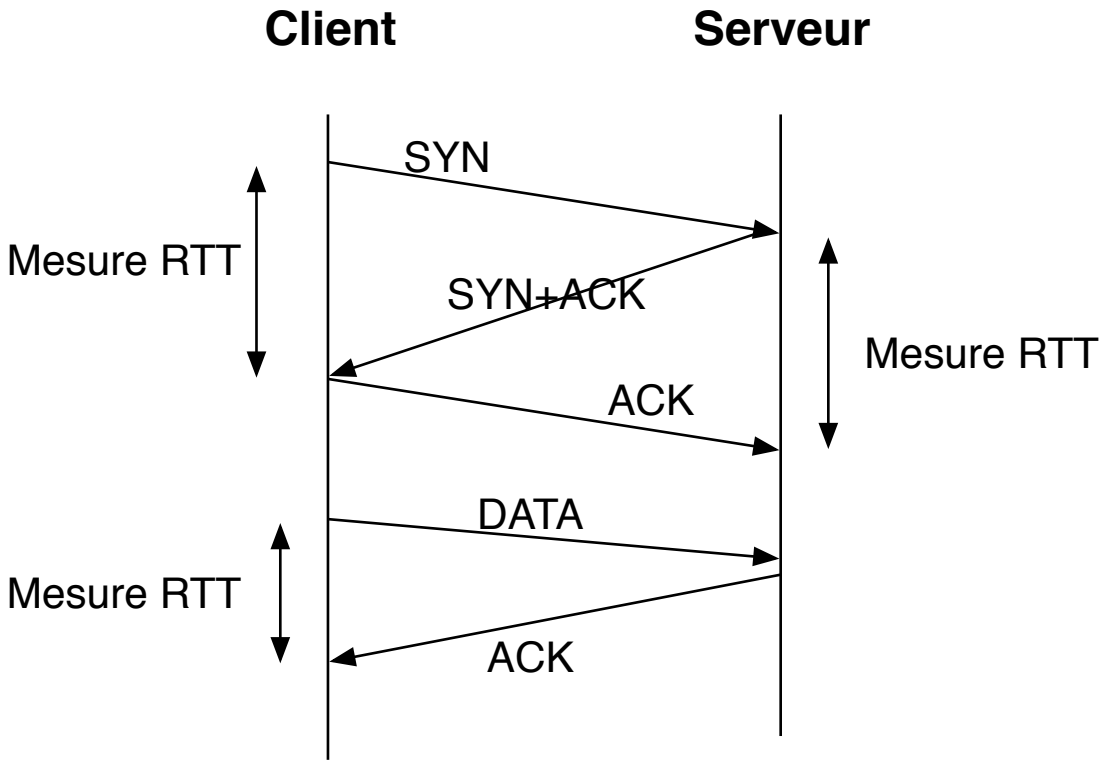


- Le premier paquet est re-émis par expiration du timer
- Le deuxième ne sera pas re-émis tout de suite à l'expiration du timer, on attendra que le premier soit acquitté
- On évite ainsi des re-émissions inutiles (surtout quand la fenêtre est grande)

Calcul du timer de re-émission

- A régler au plus juste en fonction de la Latence du réseau
- Si timer trop court : re-émission inutile
- Si timer trop long : perte de temps en cas de perte
- Latence très variable : le timer est donc calculé dynamiquement en fonction du temps d'aller-retour des paquets de données et ACQ précédents
- Possible dès l'ouverture de la connexion

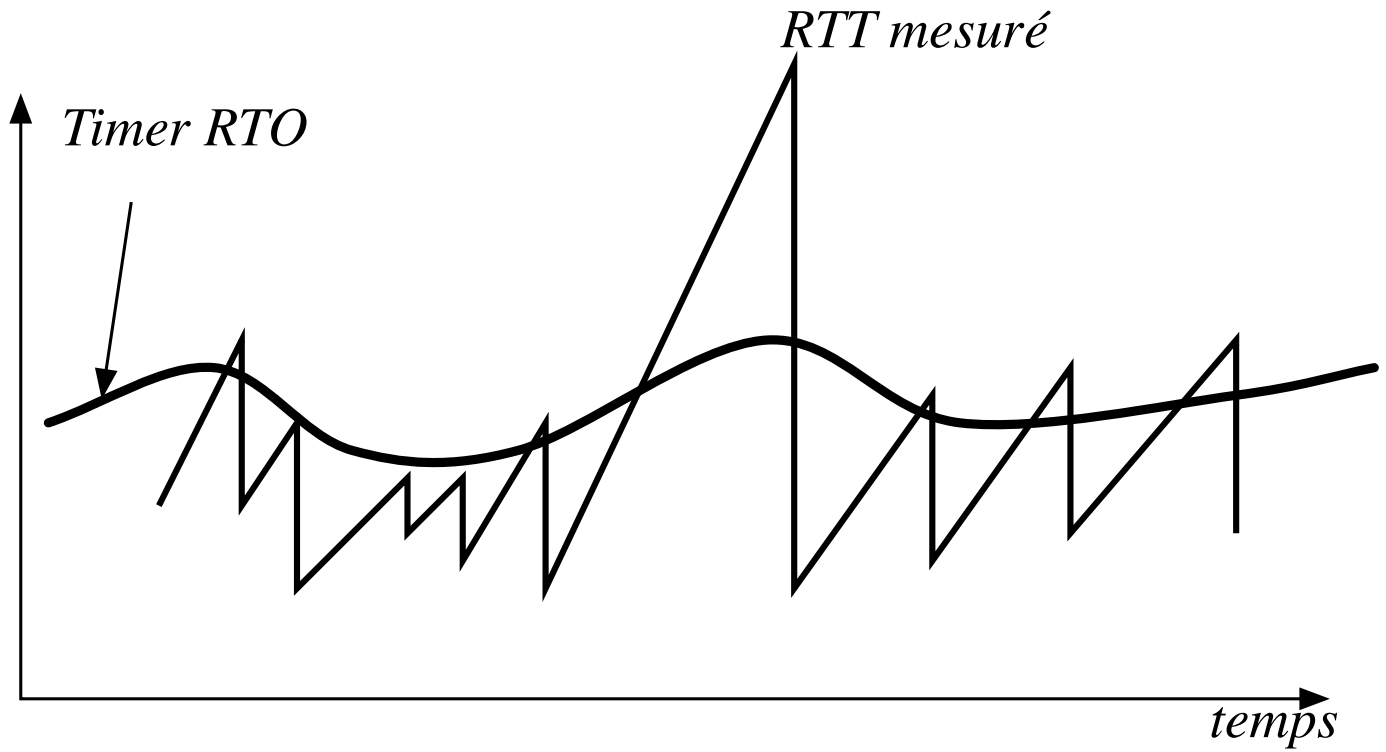
Calcul du timer de re-émission



Fonctions de calcul du timer de re-émission

- A l'arrivée de chaque Acquittement donnant un RTTMesuré:
- Calcul d'une moyenne
  - **Erreur= Moyenne - RTTMesuré**
  - **Moyenne= Moyenne - A * Erreur**
  - A : "importance" des mesures les plus récentes
  - Dans la pratique (recommandé) A= 1/8
- Calcul d'une déviation (variation) des mesures
  - **Dev= Dev + B*(|erreur| - Dev)**
  - B recommandé à 1/4
- Calcul du timer: RTO (Retransmission Time Out)
  - **RTO= Moyenne + 4 * Dev**
  - Permet de ne pas sous-dimensionner le timer en cas de forte déviation

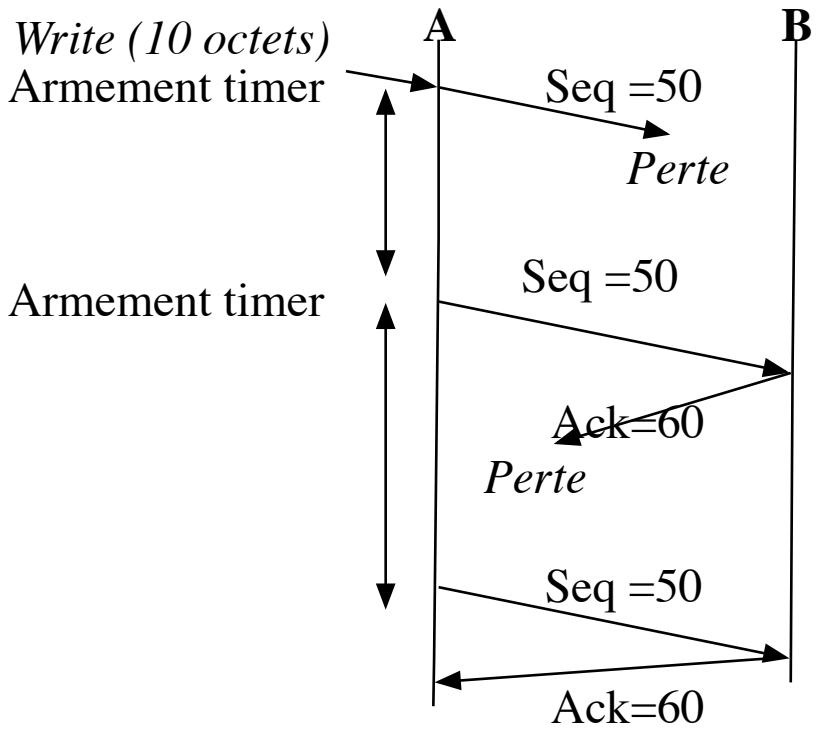
Evolution du Timer de retransmission RTO



Cas de re-émission

» En cas de re-émission la durée du timer est doublée

- En effet les pertes sont principalement dues à des saturations de routeurs et il est important de ne pas augmenter l'embouteillage par des re-émissions successives



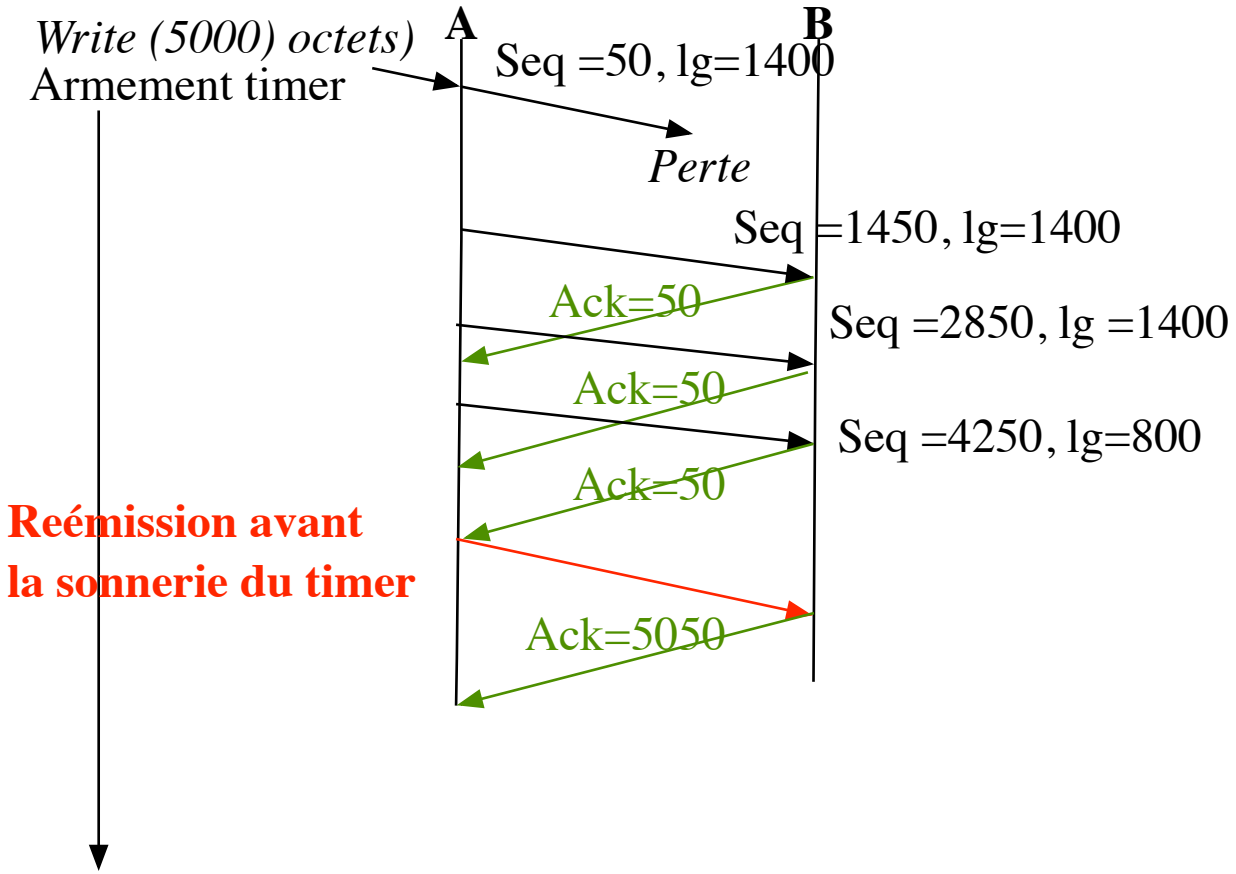
Optimisation dites de la *retransmission rapide*

» Le calcul précédemment vu du timer permet d'éviter des retransmissions inutiles en sur-dimensionnant sa durée par rapport au RTT

» En cas de perte par contre l'efficacité du protocole en est affecté. Le débit applicatif va fortement baisser (beaucoup de « temps perdu »).

Principe de la Retransmission rapide

- Après une perte le récepteur acquitte le dernier octet reçu avant la perte
  - A la réception de 3 ***Acks dupliqués*** (portant le même numéro) indique une perte probable du paquet portant ce numéro de séquence
  - Alors l'émetteur peut faire une re-émission immédiate du paquet de numéro de séquence égal au numéro d'Ack sans attendre la sonnerie du timer

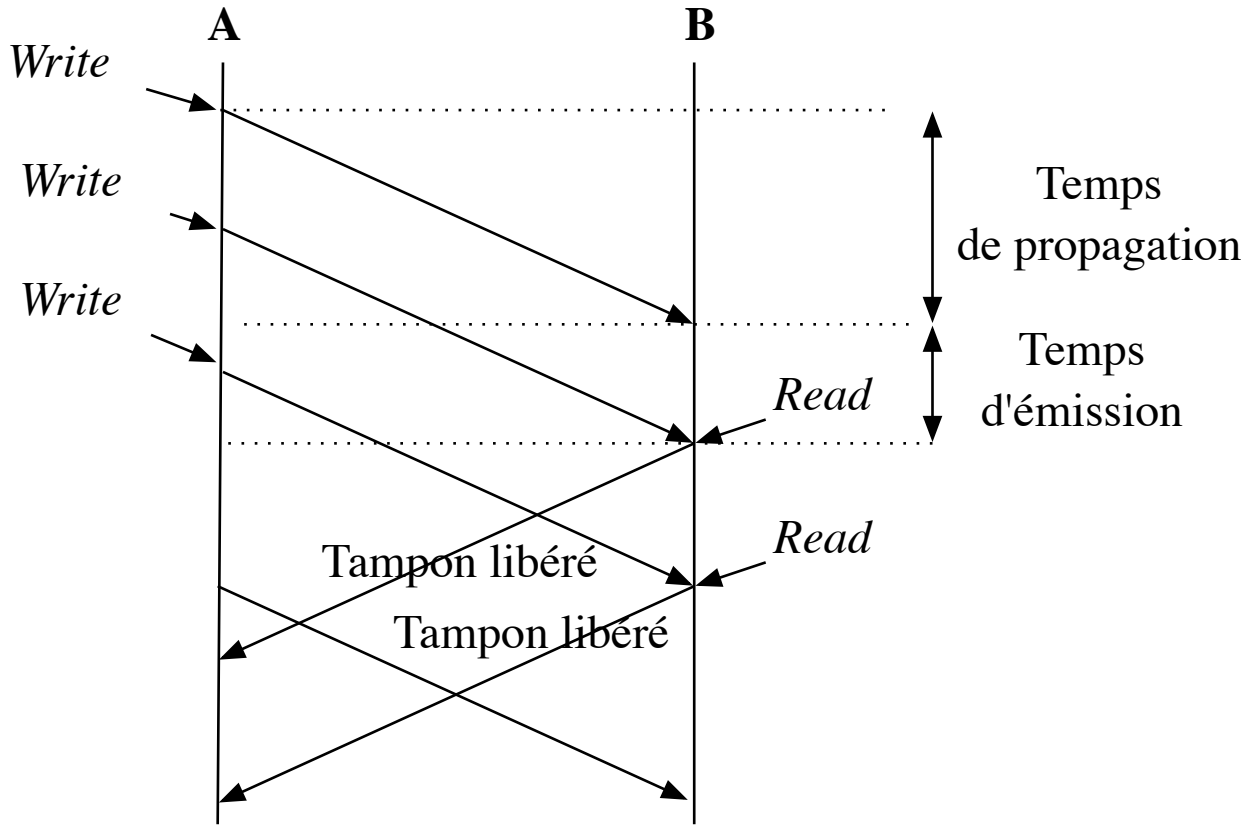


Contrôle de flux

- Problème: limiter le flux de l'émetteur pour ne pas saturer le récepteur
- Un tampon de taille fixe en réception: si le tampon est plein (l'application n'a pas encore récupéré les données) les prochaines données seront perdues
- Hypothèses:
  - La taille du tampon peut varier suivant les connexions (charge de la machine, nombre de connexions ouvertes...)
  - La latence est très variable au niveau transport, il est donc difficile de définir une taille de fenêtre optimale
  - La taille du tampon est fixée par le programmeur

Rappel : Contrôle de flux

- Acquittements permettant de signaler à l'émetteur la libération des buffers



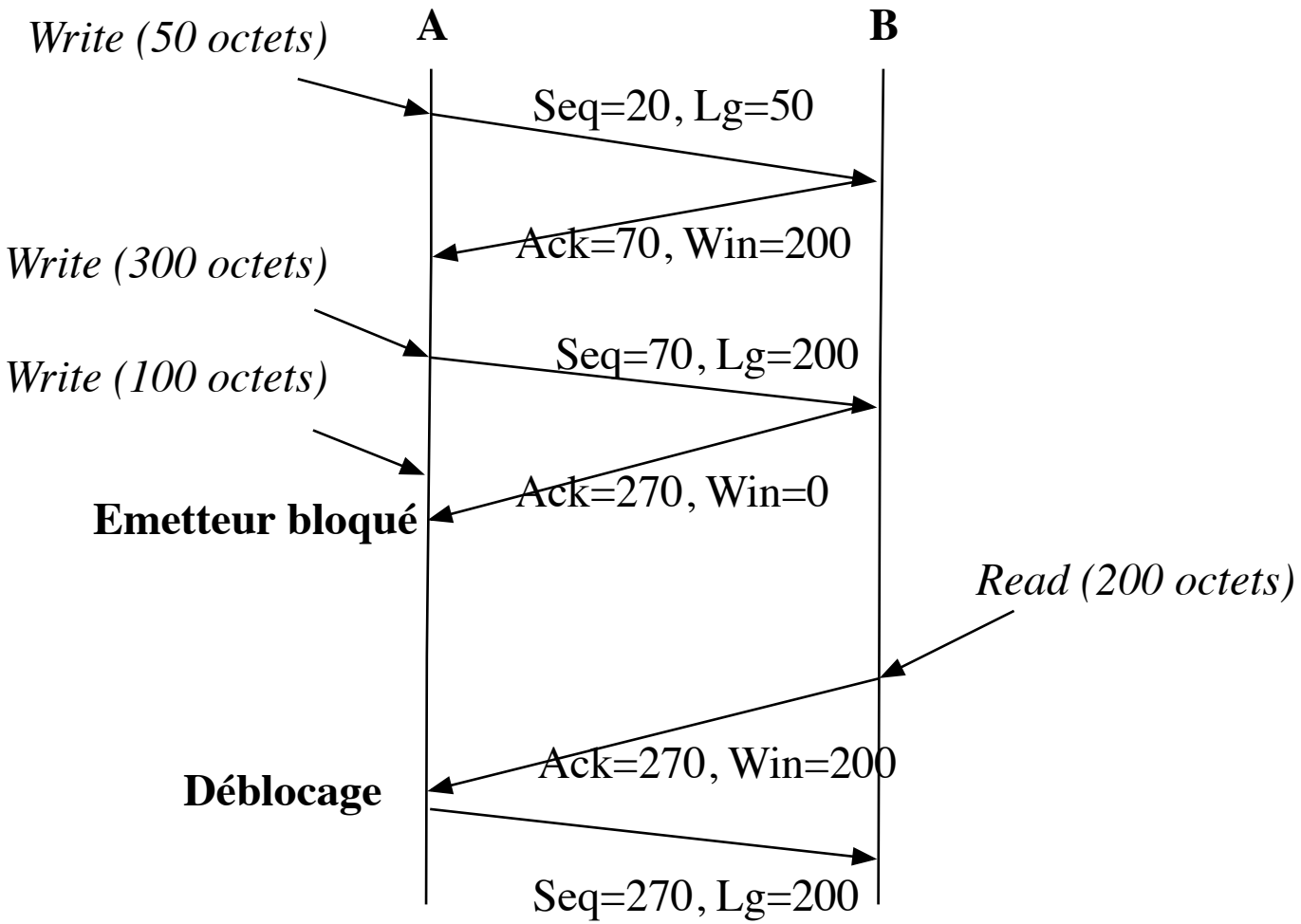
Mécanisme à nombre de crédits

- A la place de l'envoie d'acquittement spécifique au contrôle de flux TCP utilise un mécanisme à "***nombre de crédits***"
- TCP spécifie dans un acquittement de la récupération d'erreur le nombre d'octets libre dans le tampon de réception
- Permet à l'émetteur d'envoyer des paquets tant que le tampon du récepteur n'est pas plein
- Champ Fenêtre (WIN): nombre d'octets libre dans le tampon de réception et donc pouvant être expédiés après le numéro d'acquittement.

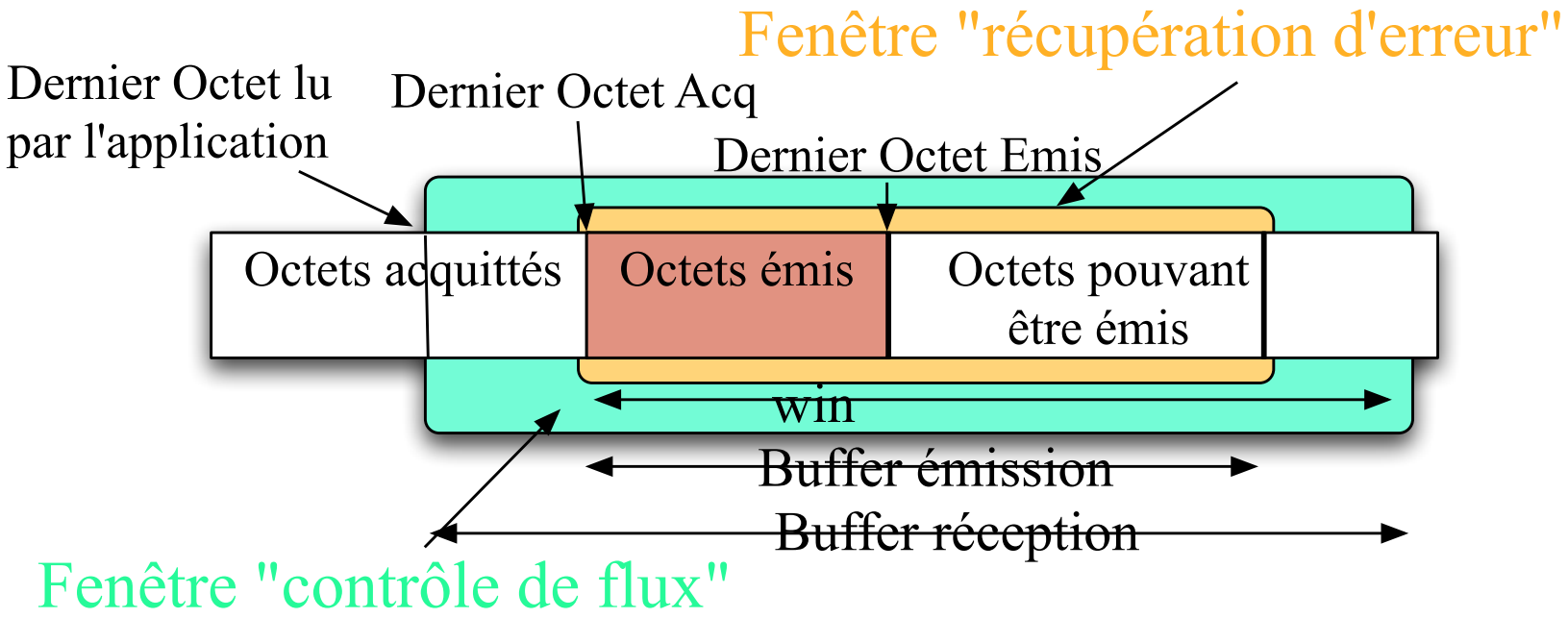
Mécanisme à nombre de crédits

- La taille des buffers est spécifiée dans le champ WIN des paquets de l'ouverture de connexion
- Quand WIN = 0 :
  - Le buffer du récepteur est plein
  - L'émetteur est bloqué jusqu'à la libération du tampon (lecture de l'application côté récepteur)

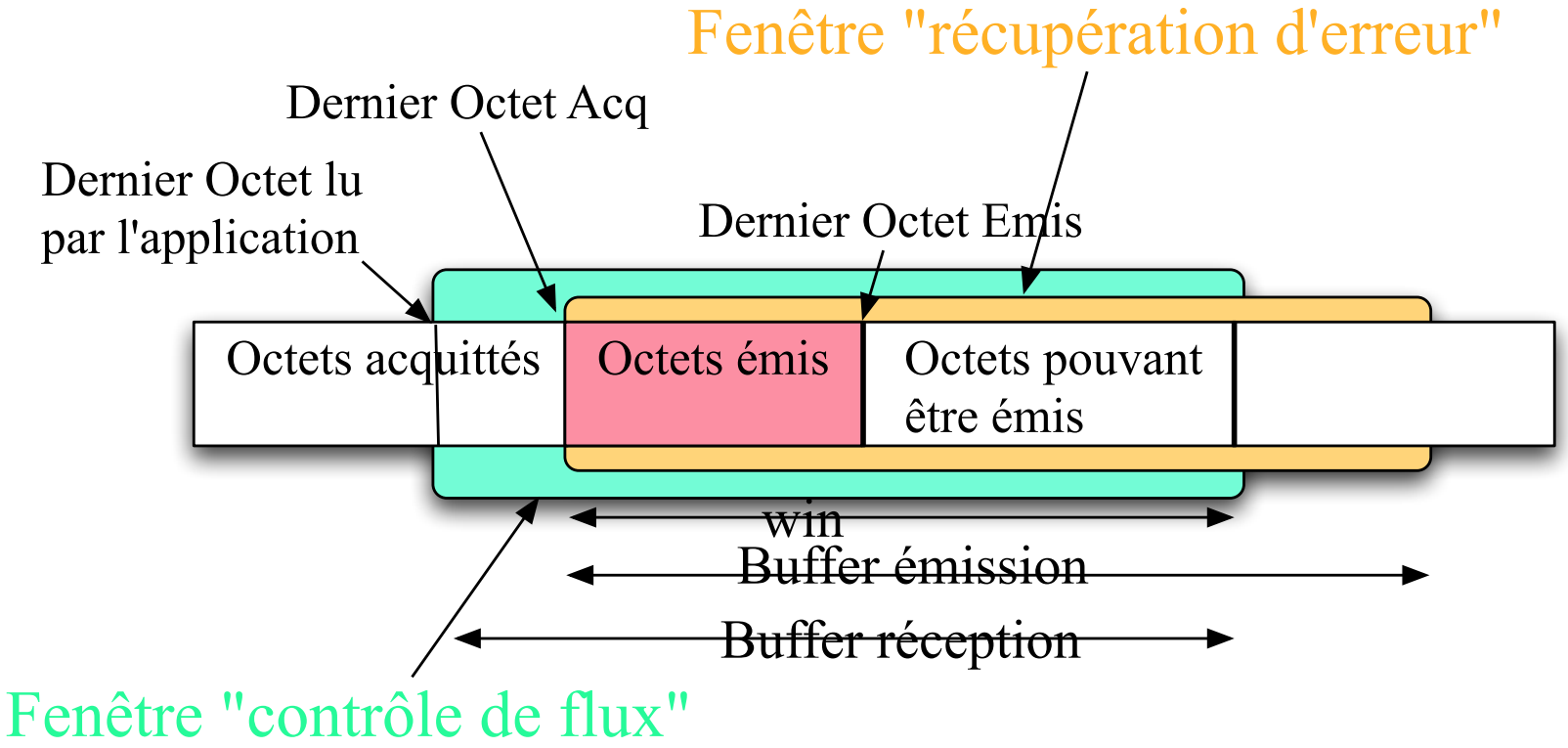
Exemple


A
B
Write (50 octets)
Seq=20, Lg=50
Ack=70, Win=200
Write (300 octets)
Seq=70, Lg=200
Write (100 octets)
Ack=270, Win=0
Emetteur bloqué
Read (200 octets)
Ack=270, Win=200
Déblocage
Seq=270, Lg=200

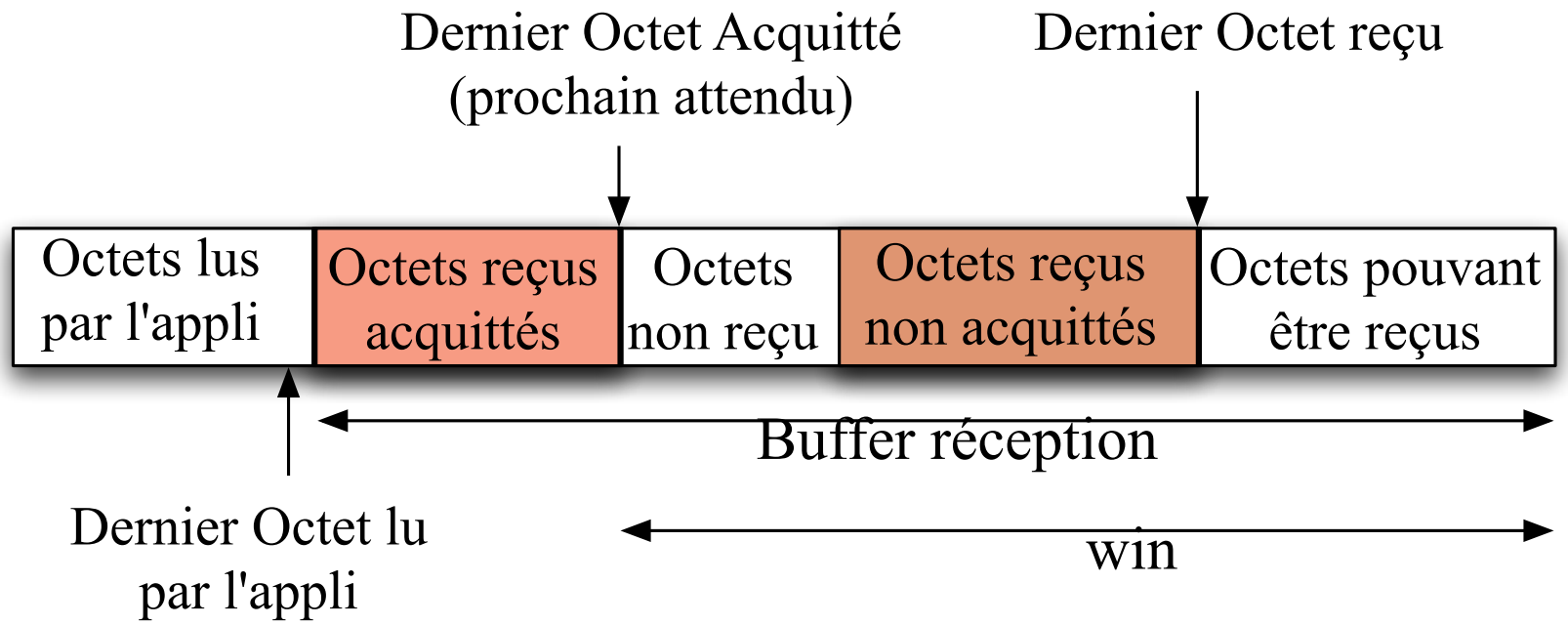
- A tout moment l'émetteur est bloqué en émission par la fenêtre la moins "avancée"
- Cas où la fenêtre du contrôle de flux est plus "avancée" que celle de la récupération d'erreur. L'émetteur va se "bloquer" sur la fenêtre de récupération d'erreur.



- Cas où la fenêtre du contrôle de flux est moins “avancée” que celle de la récupération d’erreur. L’émetteur va se “bloquer” sur la fenêtre du contrôle de flux



Côté récepteur

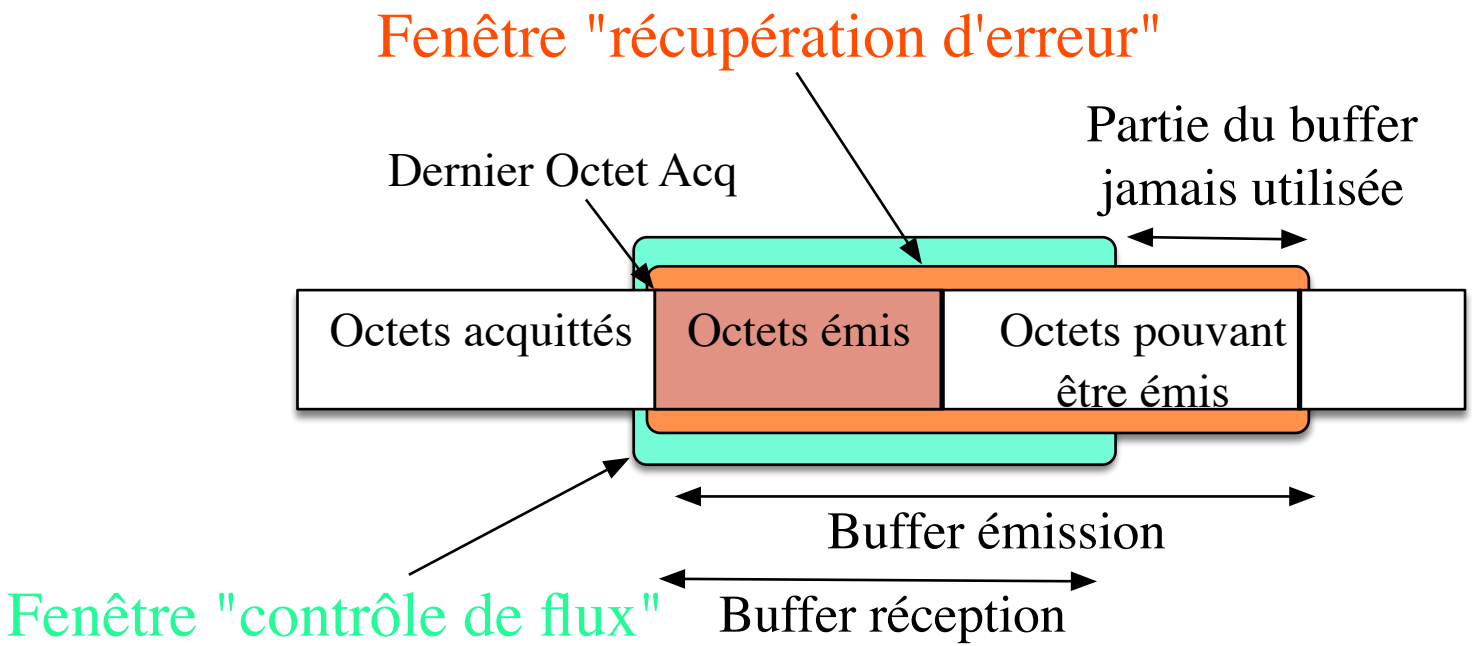


Contrôle de flux: choix de la taille du tampon de réception

- On peut essayer de régler la taille du tampon pour que les données soient émises au débit maximal si le récepteur suit le rythme.
- Dans le cas d'un récepteur suivant le rythme, un tampon trop petit en réception impliquera des blocages fréquents en émission et donc des performances moindres
- Cette taille minimale peut être calculé à partir du RTT : Taille tampon= RTT * débit physique (on néglige les entêtes)
- Ce calcul est difficile car le RTT est très variable. On peut essayer d'estimer un RTT maximal.
- Un calcul similaire peut être fait aussi pour le tampon de l'émetteur nécessaire à la fenêtre à anticipation de la récupération d'erreur.

Choix de la taille des tampons d'émission et de réception

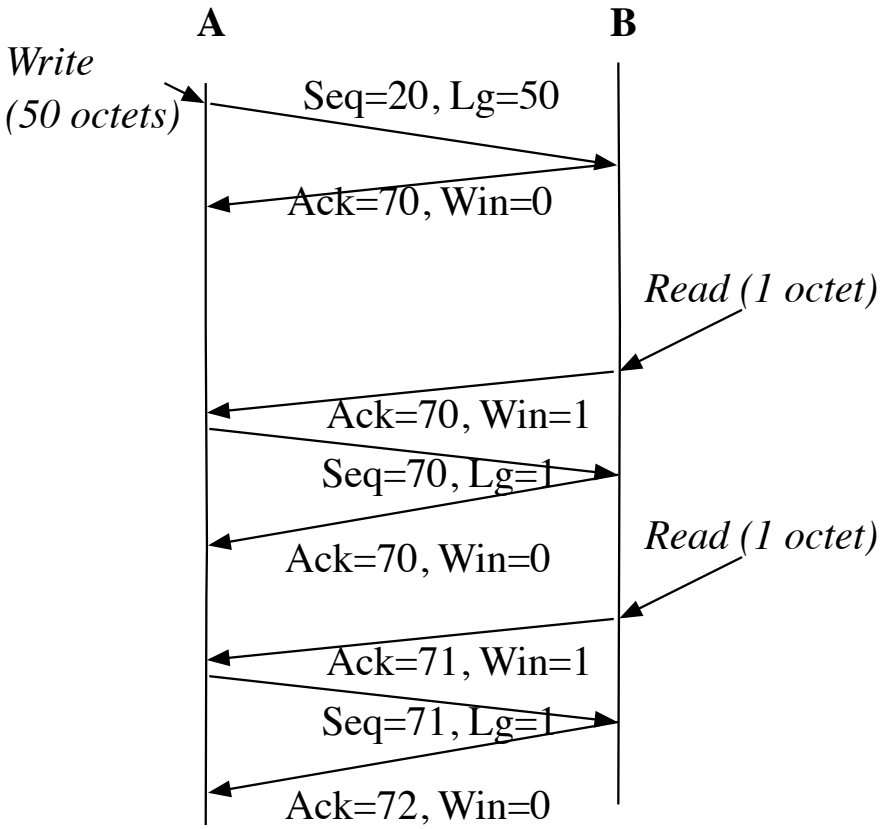
- Il ne sert à rien d'avoir la taille du tampon d'émission (contrôle d'erreur) supérieure à la taille du tampon de réception



- Il peut être intéressant d'avoir un tampon en réception plus grand qu'en émission si le récepteur a parfois des « coups de mou »
- Sous free BSD: Taille Tampon réception= 2 * Taille Tampon d'émission

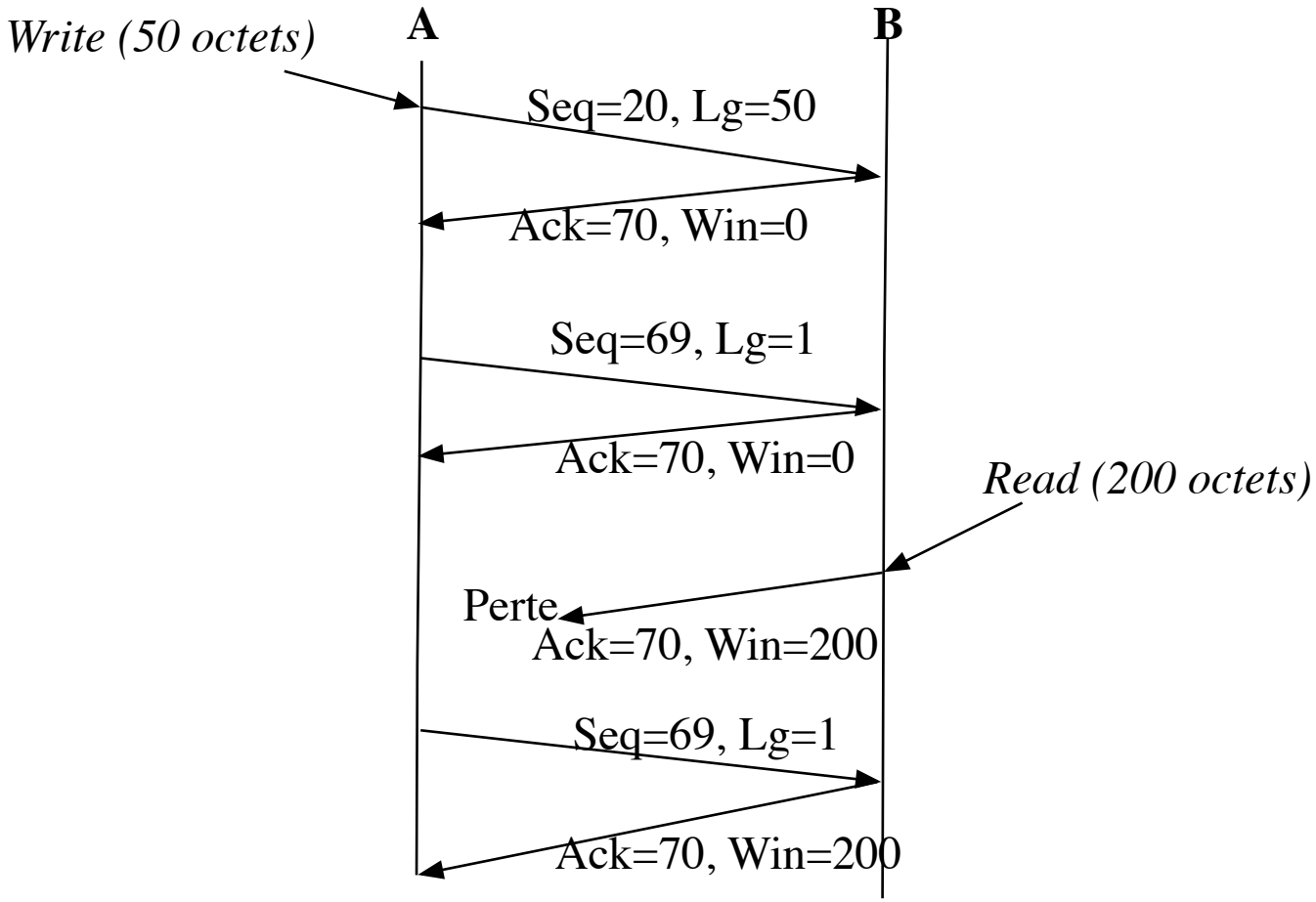
Cas de la fenêtre stupide

- Libération du buffer d'un seul octet
  - Déblocage par envoi d'un paquet avec champ Fen= 1
  - Envoi d 'un paquet de 1 seul octet de donnée
  - Charge du réseau inutile
- Solution:
  - Attendre une libération « importante » du buffer avant de débloquer
  - En pratique: Moitié du buffer ou taille maximale des segments



Cas de blocage

- Après un remplissage du tampon de réception si le paquet de “déblocage” se perd, on arrive à une situation de blocage
- Solution: l’émetteur continue à envoyer le dernier octet de donnée afin de forcer le récepteur à émettre un acquittement



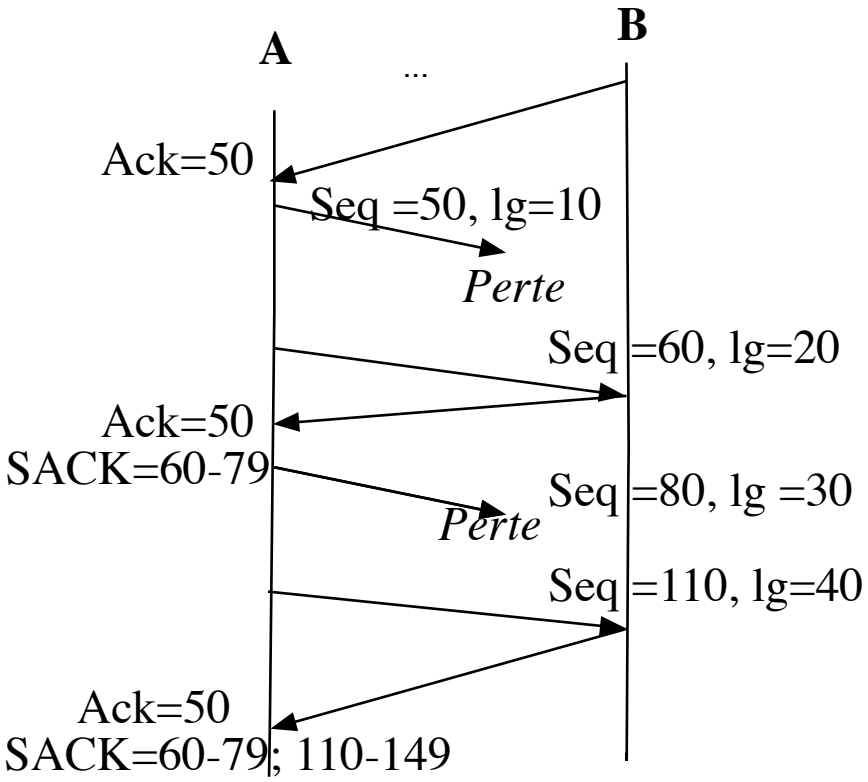
Options à l'établissement d'une connexion TCP

- Des notifications d'options sont possibles à l'ouverture de la connexion
- Elles apparaissent dans l'extension d'entête des 3 paquets d'ouverture de connexion
- Maximum segment size (MSS): permet de spécifier la taille maximale des paquets dans les deux réseaux de bout de connexion
- Time stamp: permet par estampillage temporel de connaître le temps aller-retour (round trip time: RTT) et d'initialiser le timer de reémission (voir récupération d'erreur)
- Valeur du décalage du champ WIN : permet d'augmenter la taille possible du champ WIN. Intéressant pour les réseaux haut-débit ou à temps de propagation important

Options à l'établissement d'une connexion TCP

- *Selective acknowledgment (sack)*: permet de spécifier des acquittements sélectifs
- Dans l'échange de données des acquittements peuvent porter dans l'extension d'entête des intervalles de numéro d'octets bien reçus après le numéro d'acquittement
- Exemple : NoACK= 12, (SACK= 50-60 ; 80-90)
- L'émetteur ne re-émettra que 12 à 49 et 61 à 79
- **Intéressant pour augmenter l'efficacité de TCP en cas de taux de perte non négligeable**

Exemple d'utilisation des acquittements sélectifs



- Le récepteur sait que les octets de 60 à 79 et 110 à 149 ont été reçus
- Seul les octets de 10 à 49 et 80 à 109 sont re-émis
- Sans acquittement sélectif TCP doit attendre la sonnerie des timers des deux paquets perdus