TCP

David Bromberg

TCP

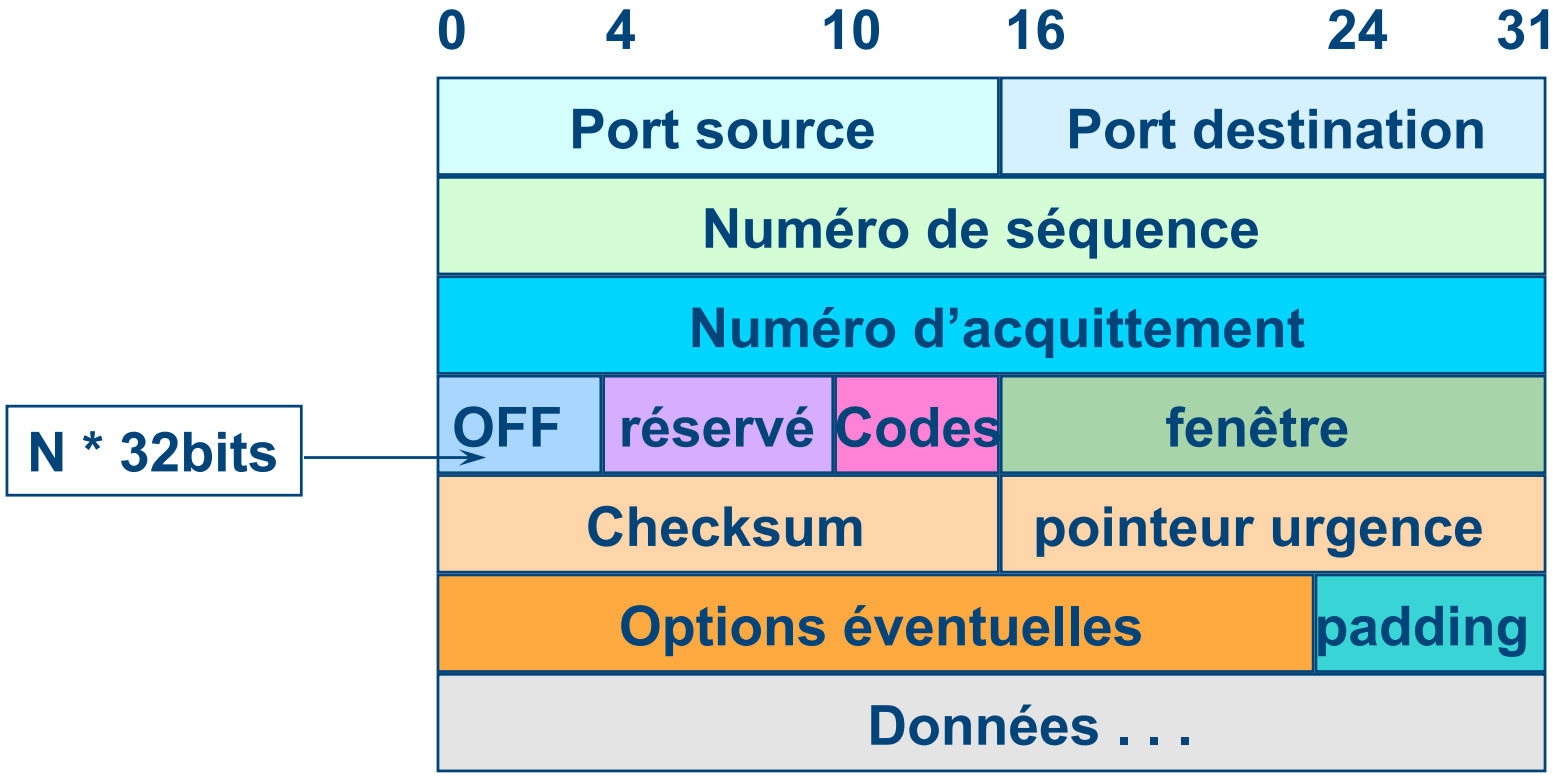
Caractéristiques

RFC 793

- TCP contient un mécanisme pour assurer ***le bon acheminement des données.***
- Le protocole TCP permet l'établissement ***d'un circuit virtuel*** entre les deux points qui échangent de l'information
- TCP a la capacité de ***mémoriser des données.***
- TCP est ***indépendant vis à vis des données transportées***, c'est un flux d'octets non structuré sur lequel il n'agit pas.
- TCP simule une connexion en ***« full duplex »***.

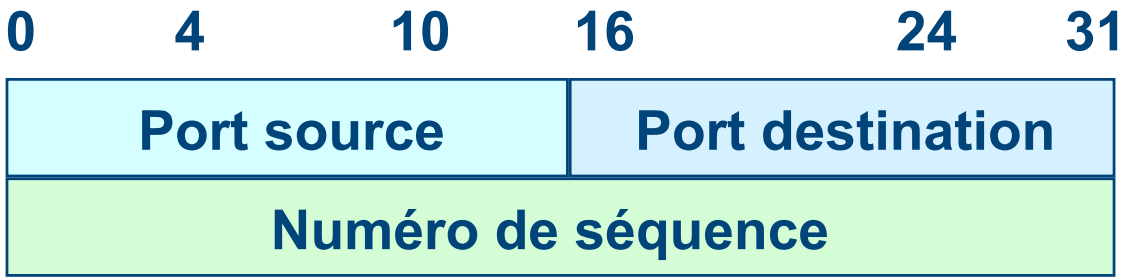
SEGMENT TCP

Le format



TCP

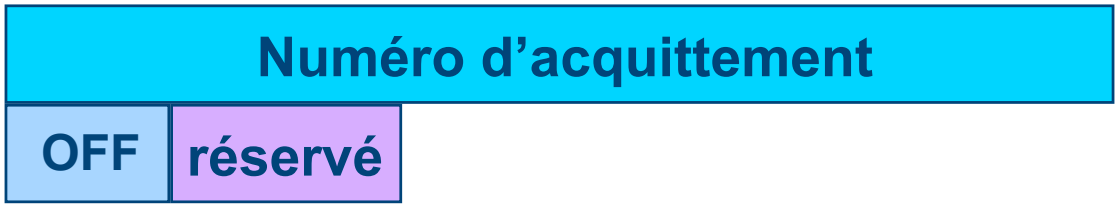
Les champs



- **TCP SOURCE PORT**
  - Numéro de port de l'application locale.
- **TCP DESTINATION PORT**
  - Numéro de port de l'application distante.
- **SEQUENCE NUMBER**
  - Nombre qui identifie la position des données **à transmettre** par rapport au segment original.
  - Au démarrage de chaque connexion, ce champ contient une valeur non nulle et non facilement prévisible, c'est la séquence initiale ou ISN.
  - TCP numérote chaque octet transmis en incrémentant ce nombre 32 bits non signé. Il repasse à 0 après avoir atteint 2^32 - 1 (4 294 967 295).
  - Pour le premier octet des données transmis ce nombre est incrémenté de un, et ainsi de suite...

TCP

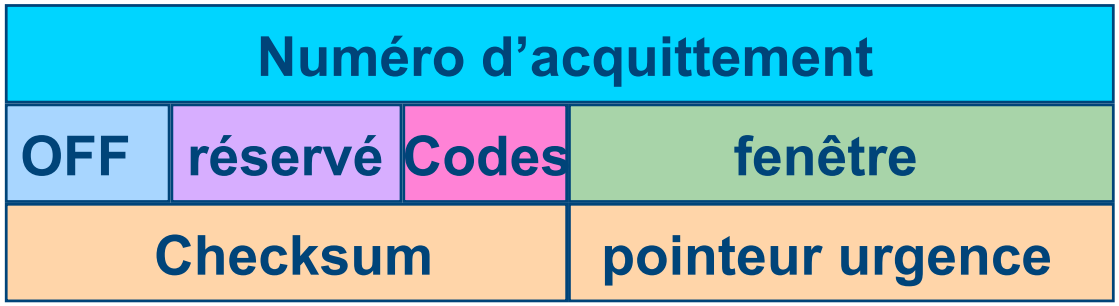
Les champs



- **ACKNOWLEDGEMENT NUMBER**
  - Numéro qui identifie la position du **dernier octet reçu** dans le flux entrant.
  - S'accompagne du drapeau ACK.
- **OFF (OFFSET)**
  - Déplacement qui permet d'atteindre les données quand il y a des options
  - Codé sur 4 bits
  - Nombre de mots de 4 octets qui composent l'en-tête.
  - Le déplacement maximum est donc de 60 octets (24-1 x 4 octets).
  - Dans le cas d'un en-tête sans option, ce champ porte la valeur 5. 10 mots de 4 octets sont donc possibles pour les options.
- **RESERVED**
  - Six bits réservés pour un usage futur !

TCP

Champ code



- **CODE**
- Six bits pour influer sur le comportement de TCP en caractérisant l'usage du segment :
  - **URG :** Le champ ``URGENT POINTER'' doit être exploité.
  - **ACK :** Le champ ``ACNOWLEDGMENT NUMBER'' doit être exploité.
  - **PSH :** Notification de l'émetteur au récepteur,
    - Indique que toutes les données collectées doivent être transmises à l'application sans attendre les éventuelles données qui suivent.
  - **RST :** Re-initialisation de la connexion
  - **SYN :** Le champ ``SEQUENCE NUMBER'' contient la valeur de début de connexion.
  - **FIN :** L'émetteur du segment a fini d'émettre. En fonctionnement normal un seul bit est activé à la fois mais ce n'est pas une obligation.

TCP

Champ Window

fenêtre

- **WINDOW**
  - Le flux TCP est contrôlé de part et d'autre pour les octets compris dans une zone bien délimitée et nommée ``fenêtre''.
  - La taille de celle-ci est définie par un entier non signé de 16 bits, qui en limite donc théoriquement la taille à 65 535 octets
  - Chaque partie annonce ainsi la taille de son buffer de réception. Par construction, l'émetteur n'envoie pas plus de données que le récepteur ne peut en accepter.
  - Cette valeur varie en fonction de la nature du réseau et surtout de la bande passante devinée à l'aide de statistiques sur la valeur du RTT.

TCP

Checksum	pointeur urgence

- **CHECKSUM**
  - Un calcul qui porte sur la totalité du segment, en-tête et données.

- **URGENT POINTER**
  - Ce champ n'est valide que si le drapeau URG est armé.
  - Ce pointeur contient alors un offset à ajouter à la valeur de SEQUENCE NUMBER du segment en cours pour délimiter la zone des données urgentes à transmettre à l'application.

TCP

Options

Options éventuelles

- C'est un paramètrage de TCP. Sa présence est détectée dès lors que l'OFFSET est supérieur à 5.
- Les options utilisées :
  - **MSS**
    - Taille maximale du segment des données applicatives que l'émetteur accepte de recevoir. Au moment de l'établissement d'une connexion (paquet comportant le flag SYN), chaque partie annonce sa taille de MSS. Ce n'est pas une négociation. Pour de l'Ethernet la valeur est 1460 (= *MTU* - 2 x 20).
  - **Timestamp**
    - pour calculer la durée d'un aller et retour (RTT ou ``round trip time'').
  - **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 x 2shift. Par exemple si ``shift'' vaut 1 la taille de la fenêtre est de 131072 octets soit encore 128 ko.
  - **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 ; à cet effet une option ``No Operation'' ou nop, codée sur 1 seul octet, est prévue pour compléter les mots.

TCP

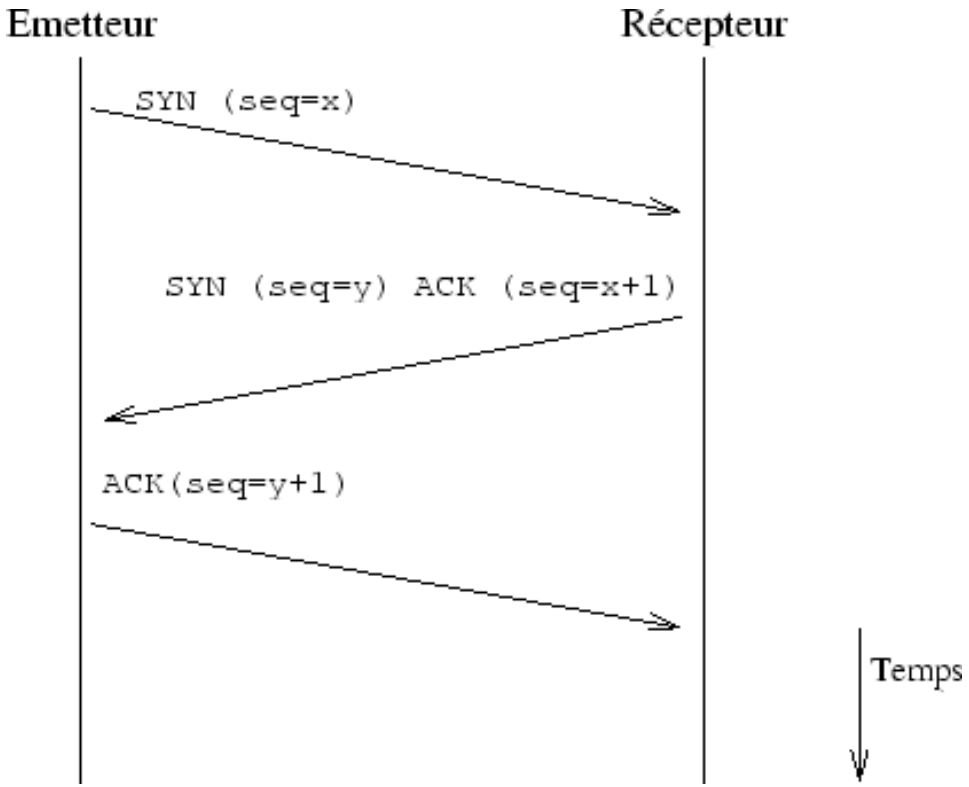
La connexion

- 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))
- 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

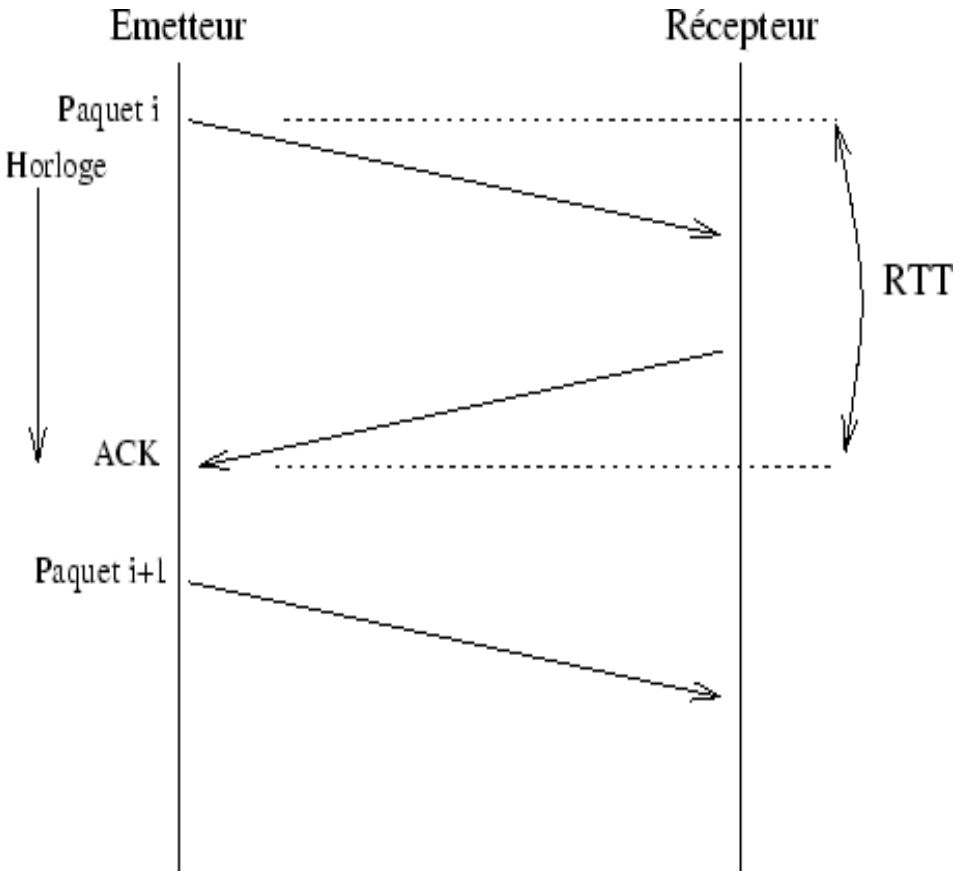
La connexion



- L'établissement d'une connexion TCP s'effectue en trois temps.

TCP

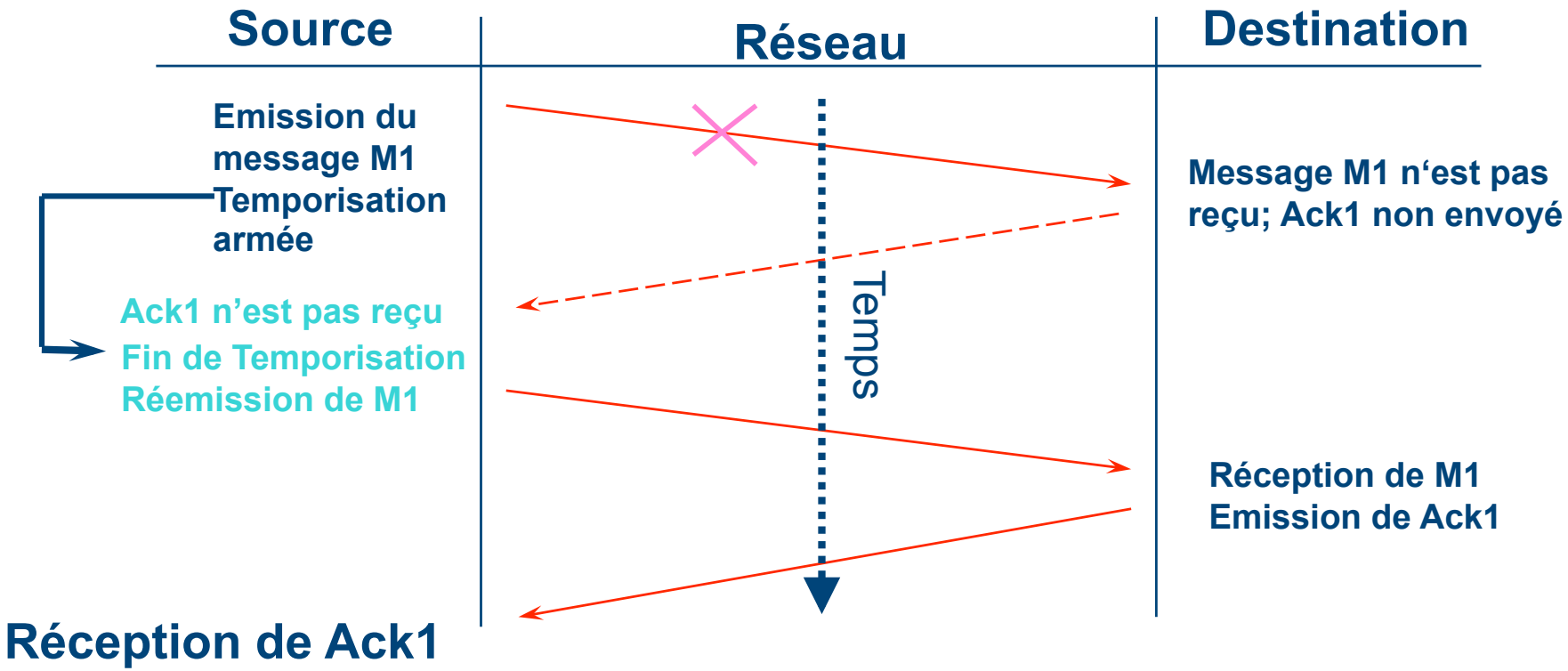
Qu'est ce que l'acquittement ?



- Au départ du *Paquet i* une horloge se déclenche.
- Si cette horloge dépasse une valeur limite avant réception de l'ACK le *Paquet i* est retransmis.
- Le temps maximum d'attente est de 2 x *MSL* (Maximum Segment Lifetime").
- Le temps qui s'écoule entre l'émission d'un paquet et la réception de son acquittement est le RTT, il doit donc être inférieur à 2 x *MSL.*
- L'émetteur conserve la trace du *Paquet i* pour éventuellement le renvoyer.

TCP

Qu'est ce que l'acquittement ?



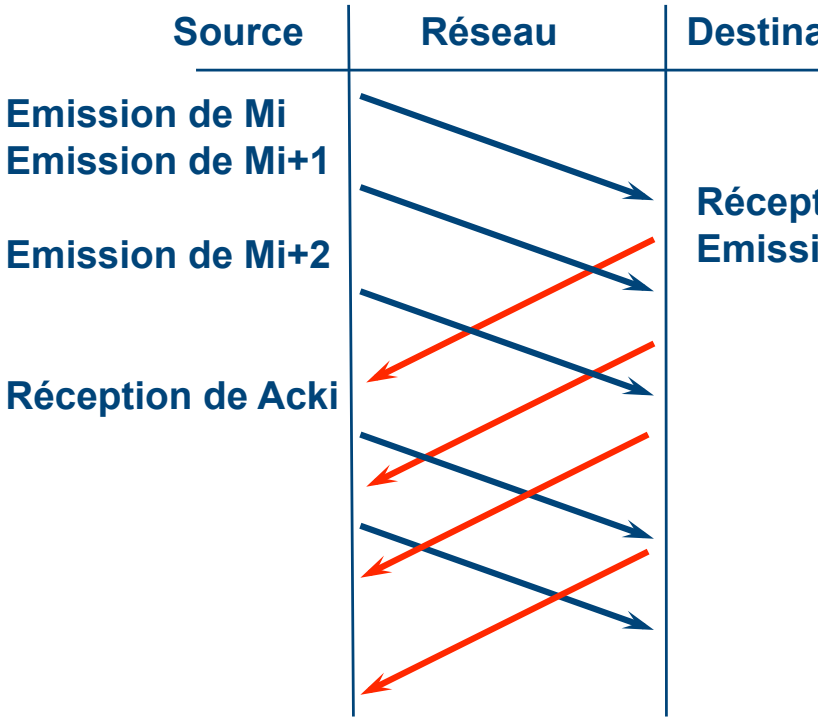
TCP

Limitation de l'acquittement

- Mécanisme est totalement inadapté au transfert de flux de données.
- Sous-emploi la bande passante du réseau.

TCP

Qu'est ce que le fenêtrage?



Fenêtrage de taille 3

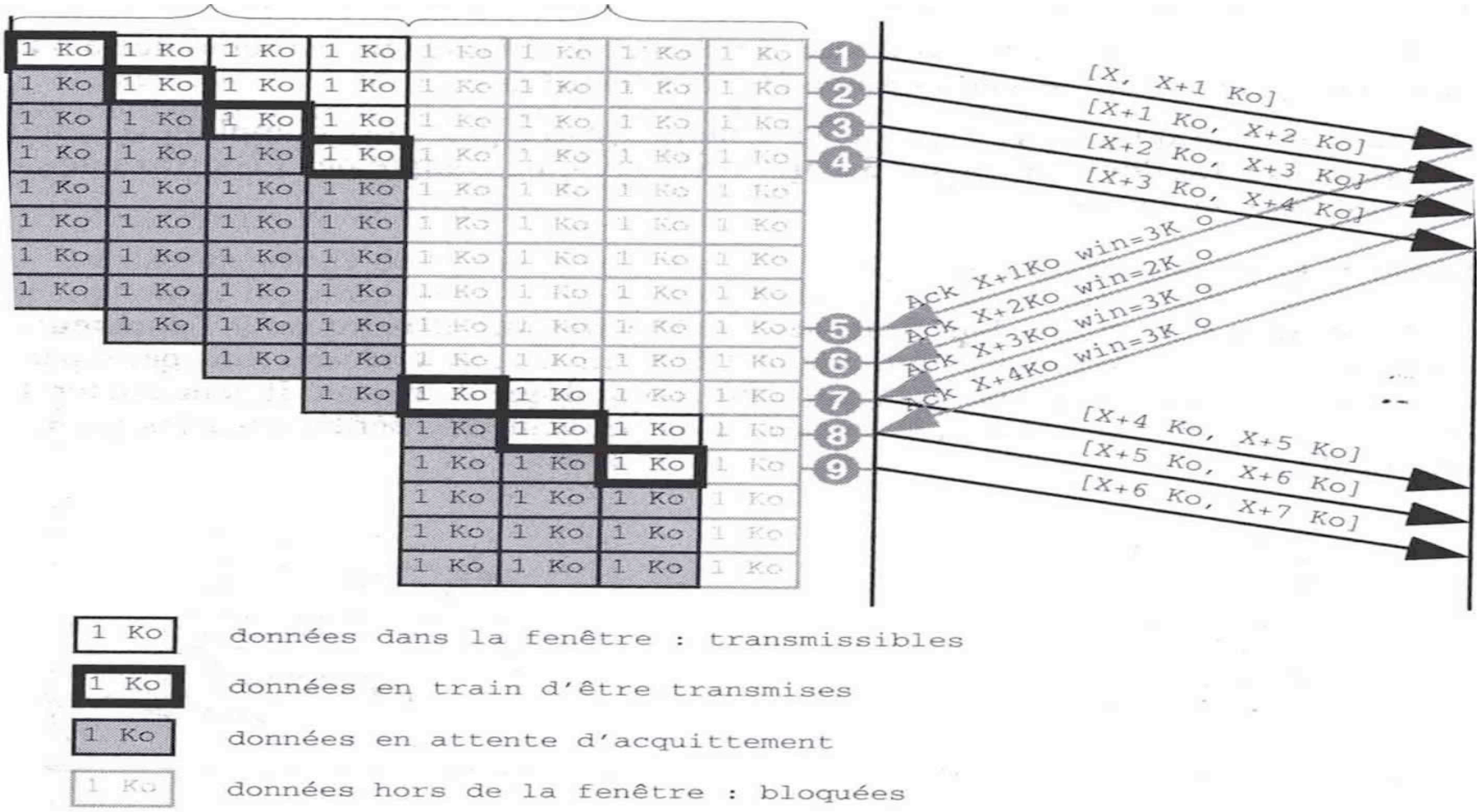
- La technique acquittement simple pénalise les performances puisqu'il faut attendre un acquittement avant d'émettre un nouveau message. Le fenêtrage améliore le rendement des réseaux.
- La technique du fenêtrage : une fenêtre de taille T, permet l'émission d'au plus T messages "*non acquittés*" avant de ne plus pouvoir émettre.

TCP
Qu'est ce que la fenêtre glissante ?

- 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.
- Le mécanisme de fenêtrage mis en oeuvre dans TCP opère au niveau de l'octet et non pas au niveau du segment.

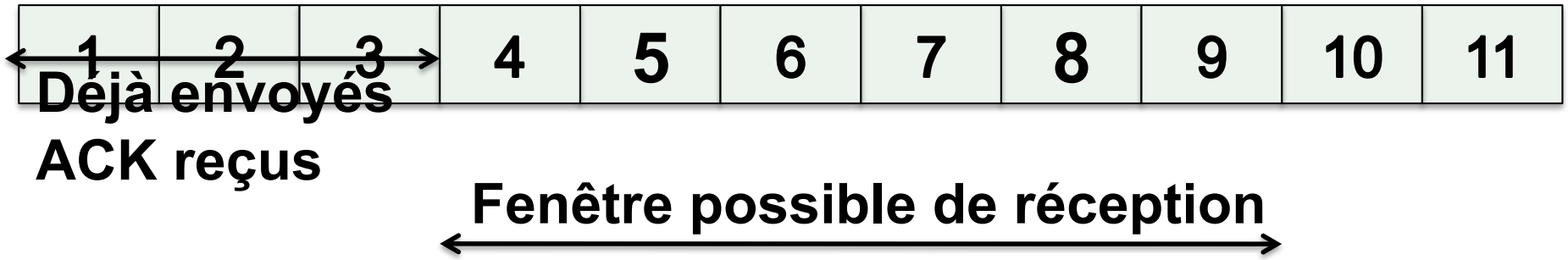
TCP

Avancement de la fenêtre



TCP

Qu'est ce que la fenêtre glissante ?



TCP
Qu'est ce que la fenêtre glissante ?

- Le nombre de paquets à envoyer avant d'attendre le premier acquittement est en fonction de deux paramètres :
  - La largeur de la fenêtre : champ WINDOW de l'en-tête (4096, 8192 ou 16384)
    - Elle change dynamiquement pour deux raisons :
      - L'application change la taille du ``buffer de la socket' qui correspond à la taille de cette fenêtre.
      - Chaque acquittement ACK envoyé est assorti d'une nouvelle valeur de taille de la fenêtre, permettant ainsi à l'émetteur d'ajuster à tout instant le nombre de segment qu'il peut envoyer simultanément.
  - La taille maximale des données TCP , ou MSS vaut 512 octets par défaut.
    - Le datagramme IP a donc une taille égale au MSS augmentée de 40 octets (20 + 20), en l'absence d'option de TCP.
    - Cette option apparait uniquement dans un paquet assorti du drapeau SYN, donc à l'établissement de la connexion.
- Comme de bien entendu cette valeur est fortement dépendante du support physique et plus particulièrement du MTU.

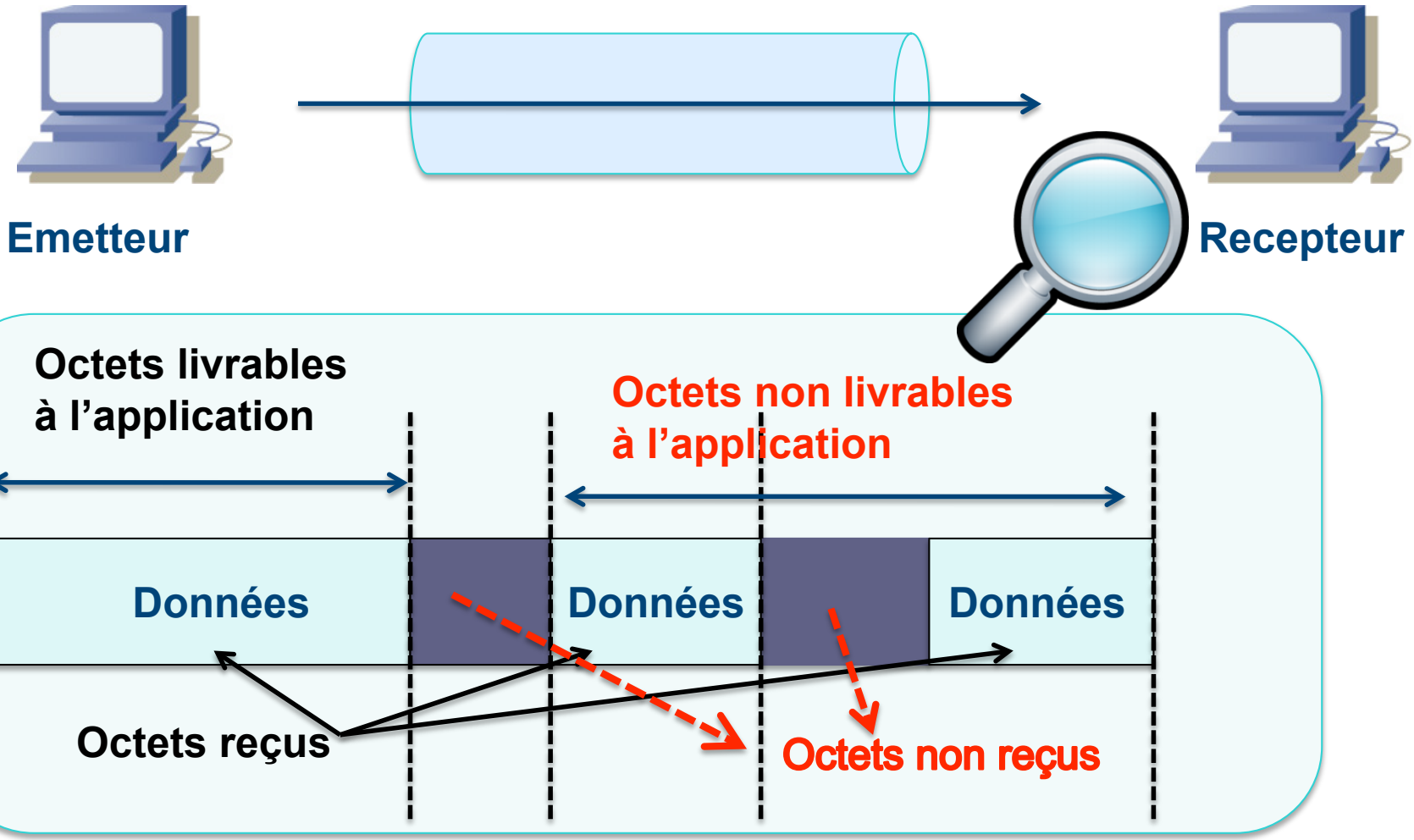
TCP

La fenêtre glissante ?

- Le débit obtenu dépend :
  - De la taille de la fenêtre
  - De la bande passante disponible.
- L'agrandissement de la taille de la fenêtre jusqu'à une limite optimale
  - Autrement les paquets sont perdus
    - ⇔ Paquets envoyés trop rapidement pour être reçus par le destinataire.
  - TCP doit optimiser son rendement d'émission de paquets
    - ⇔ Eviter la réémission.
- Taille limite optimale de la largeur de la fenêtre est fonction :
  - De la bande passante théorique du réseau.
  - Du taux d'occupation instantané du réseau.
    - ⇔ Taux d'occupation dynamique
    - ⇔ TCP doit modifier continuellement les tailles de fenêtre pour en tenir compte.

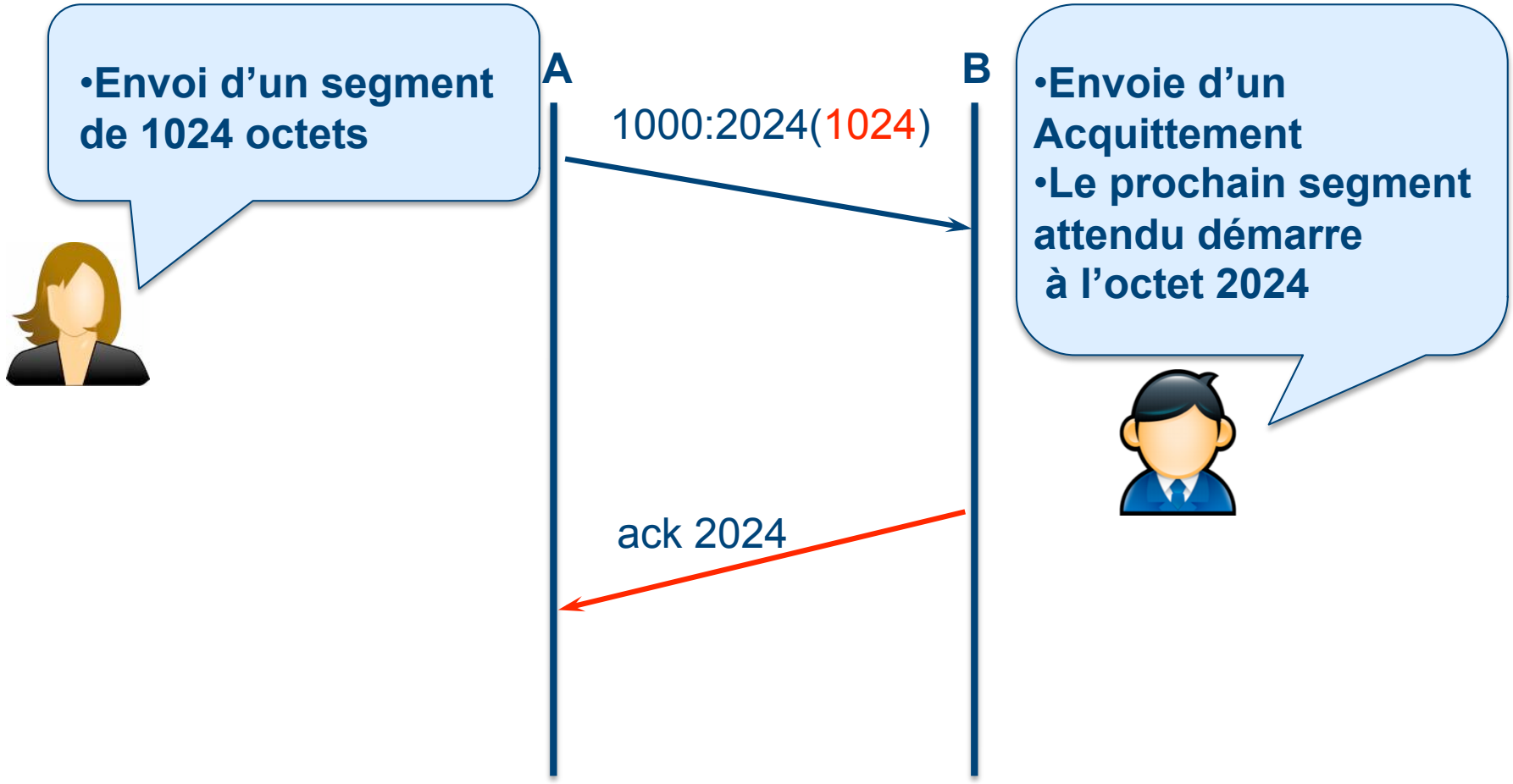
TCP

Buffer de réception



TCP

Transfert de données



TCP

Détection de pertes (1/3)

Comment détecte t-on les pertes de segments ?

- **Deux cas de détection des pertes**
  - Réception d'acquittements dupliqués
  - Non retour d'un acquittement avant un certain délai (expiration d'horloge)

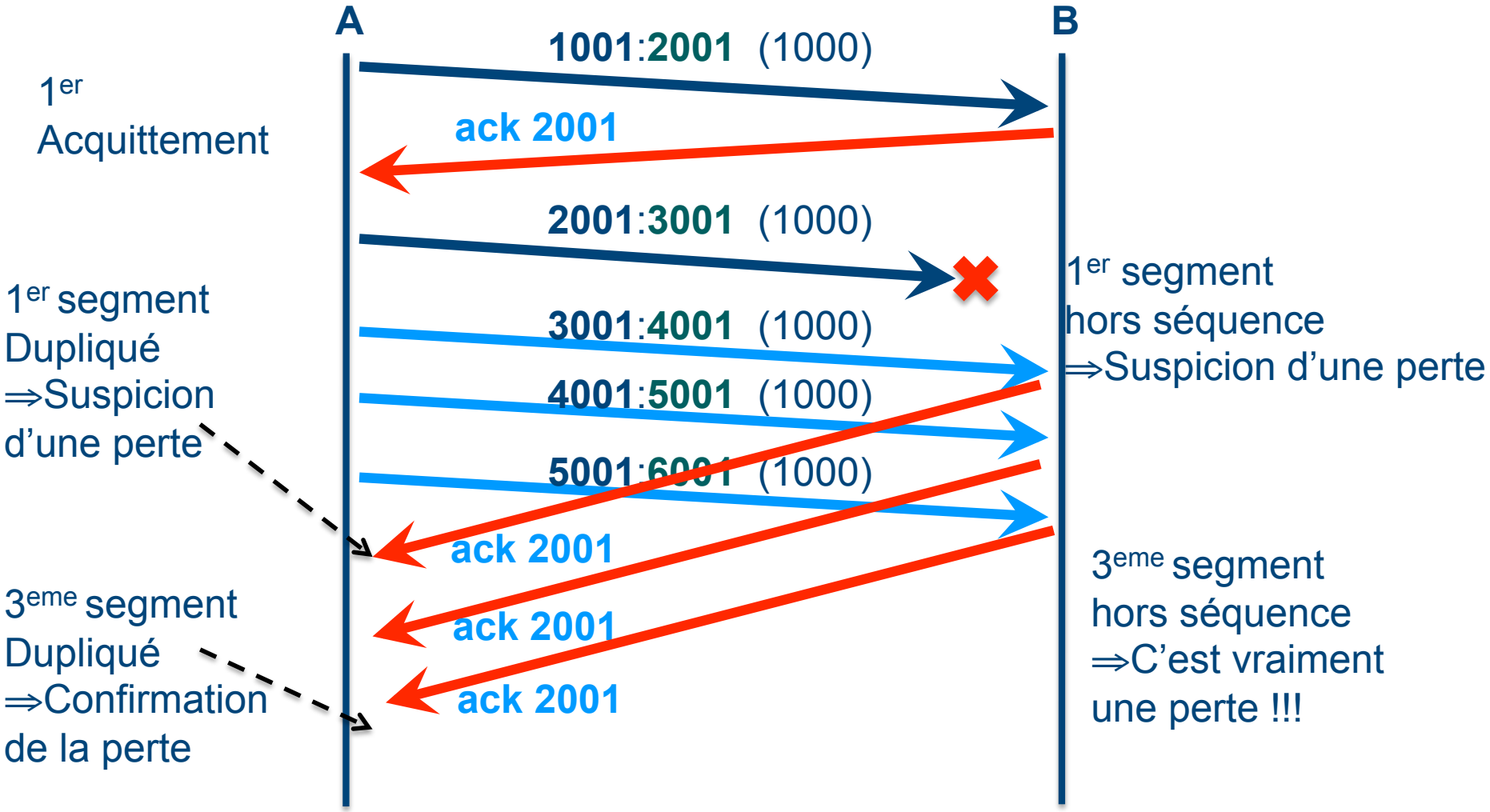
TCP

Acquittements dupliqués

Qu’est ce qu’un acquittement dupliqué ?

TCP

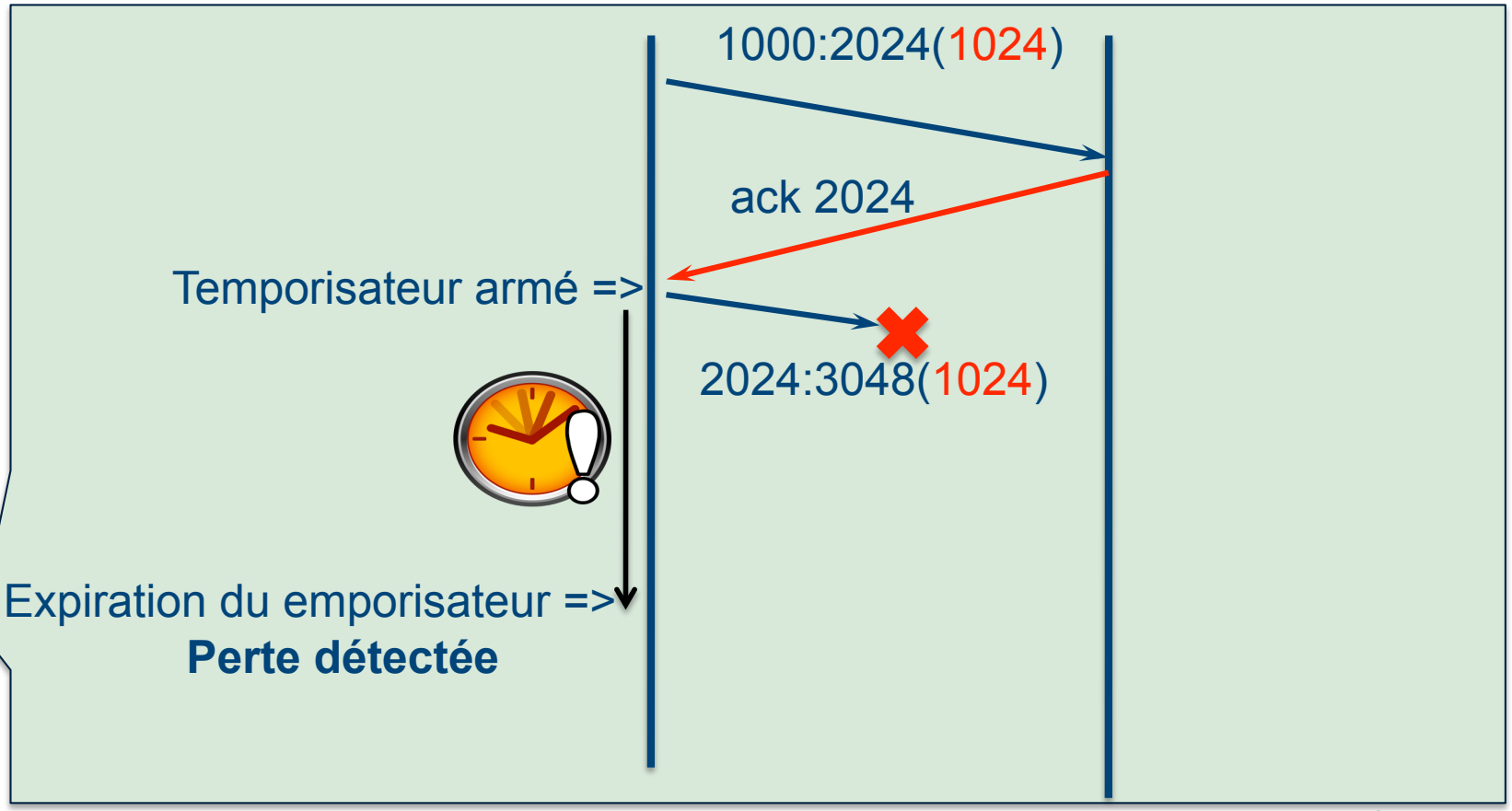
Acquittements dupliqués



TCP

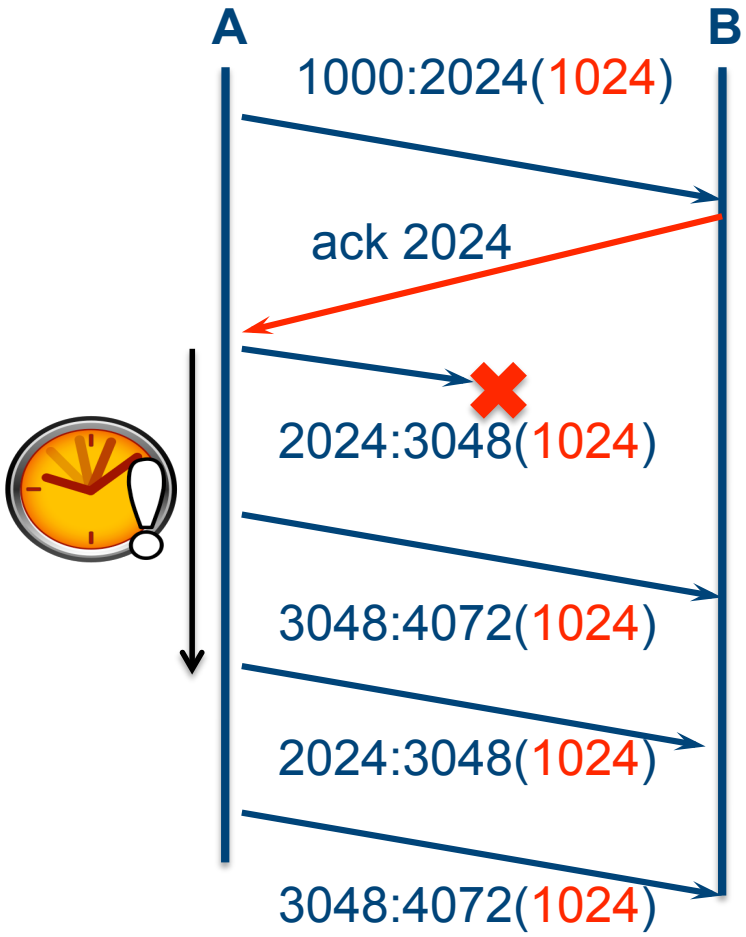
Expiration du délai d'attente

Qu'est ce qu'une expiration d'horloge ?



TCP

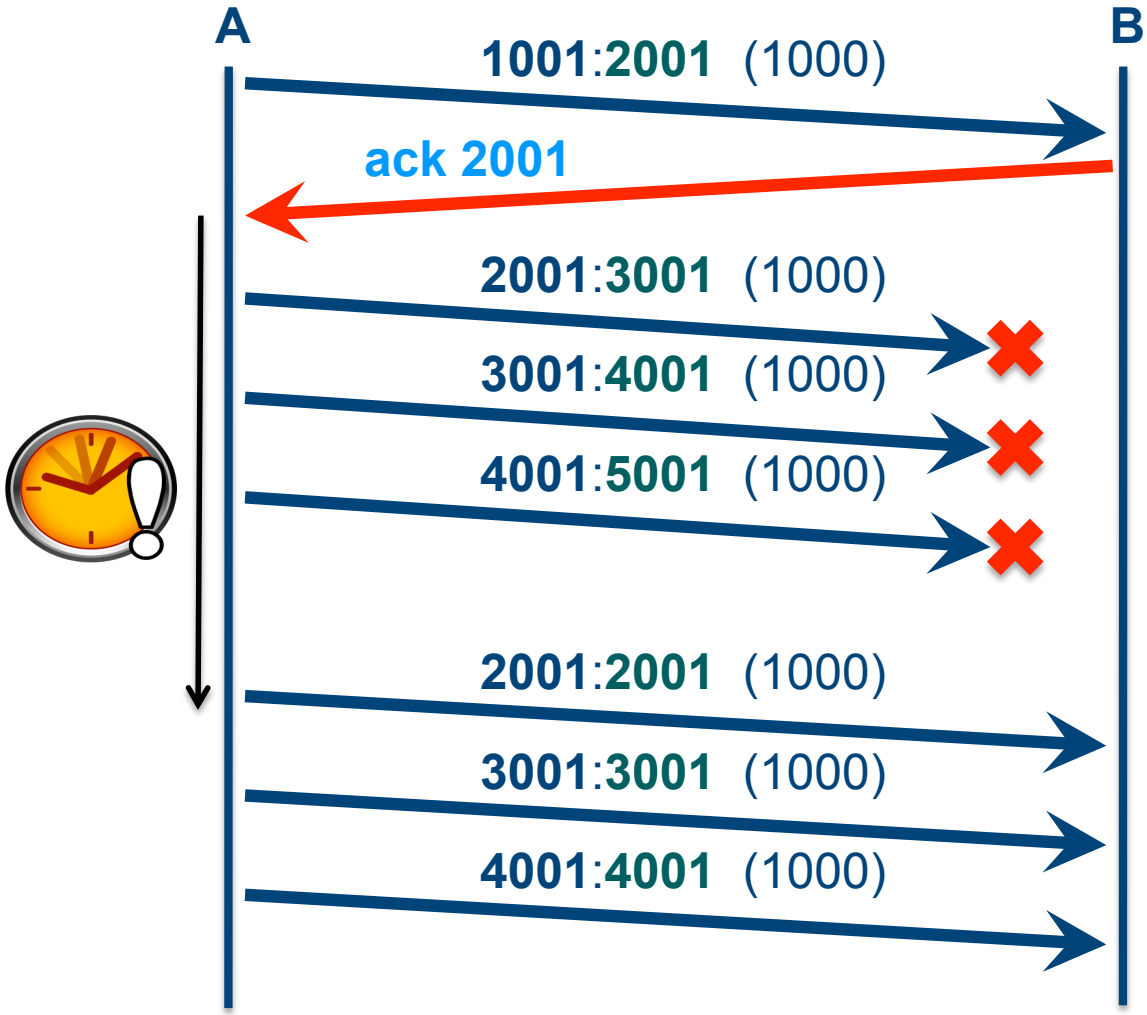
Détection de pertes (2/3)



- **Retransmission go back N**
  - L'émetteur retransmet tous les segments non acquittés
- Pas de réception d'acquittements dupliqués avant que le temporisateur ne soit arrivé à échéance.

TCP

Transmission go back N



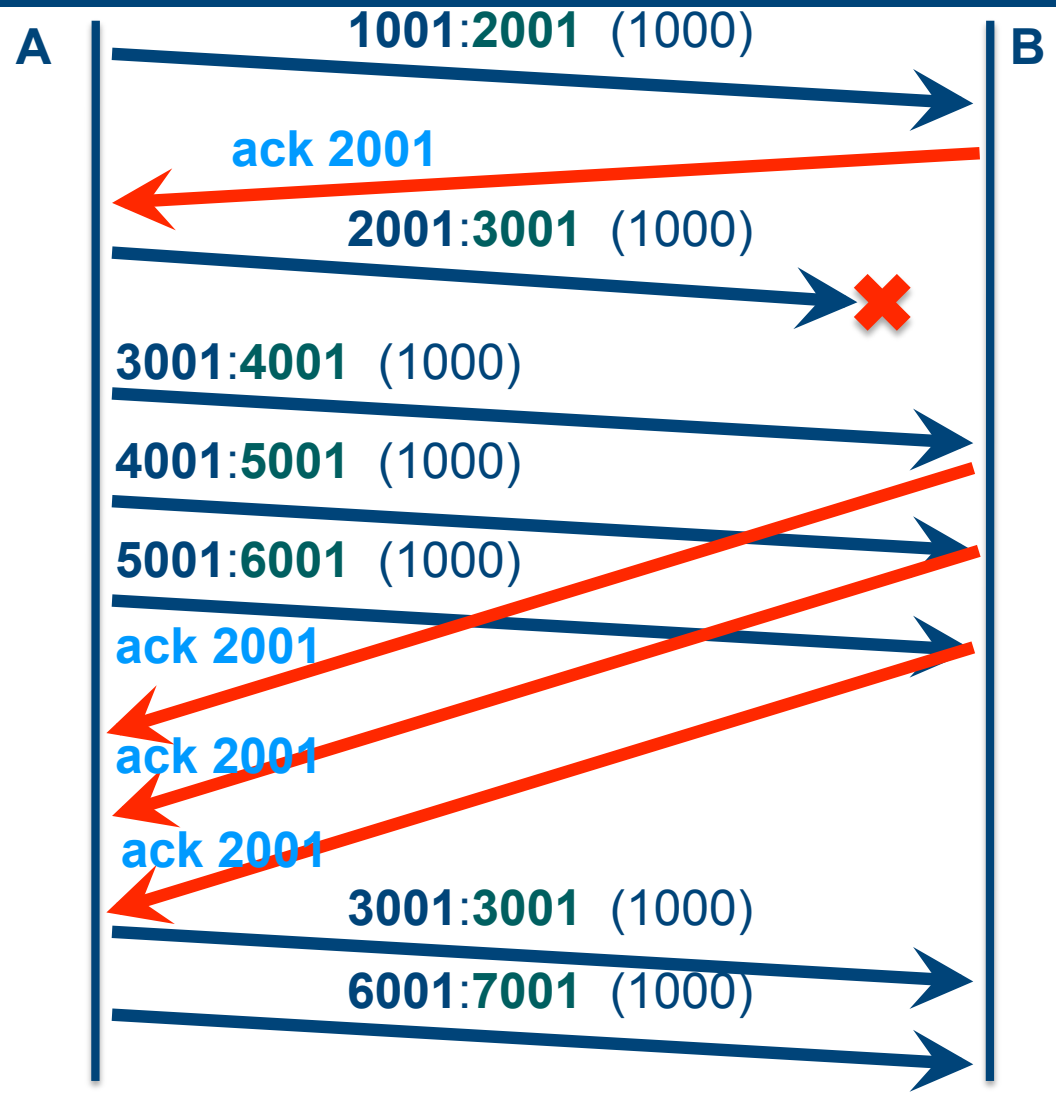
TCP

Détection de pertes (3/3)

- **Retransmission sélective**
  - L'émetteur retransmet uniquement le segment considéré comme perdu

TCP

Retransmission sélective



TCP

Les politiques d'acquittement

- Différentes politiques d'acquittement peuvent être mises en œuvre dans les protocoles de transfert de données tel que TCP

- Acquittement immédiat
- Acquittement retardé d'un délai constant
- Acquittement cumulé (acquittement produit tous les deux segments)

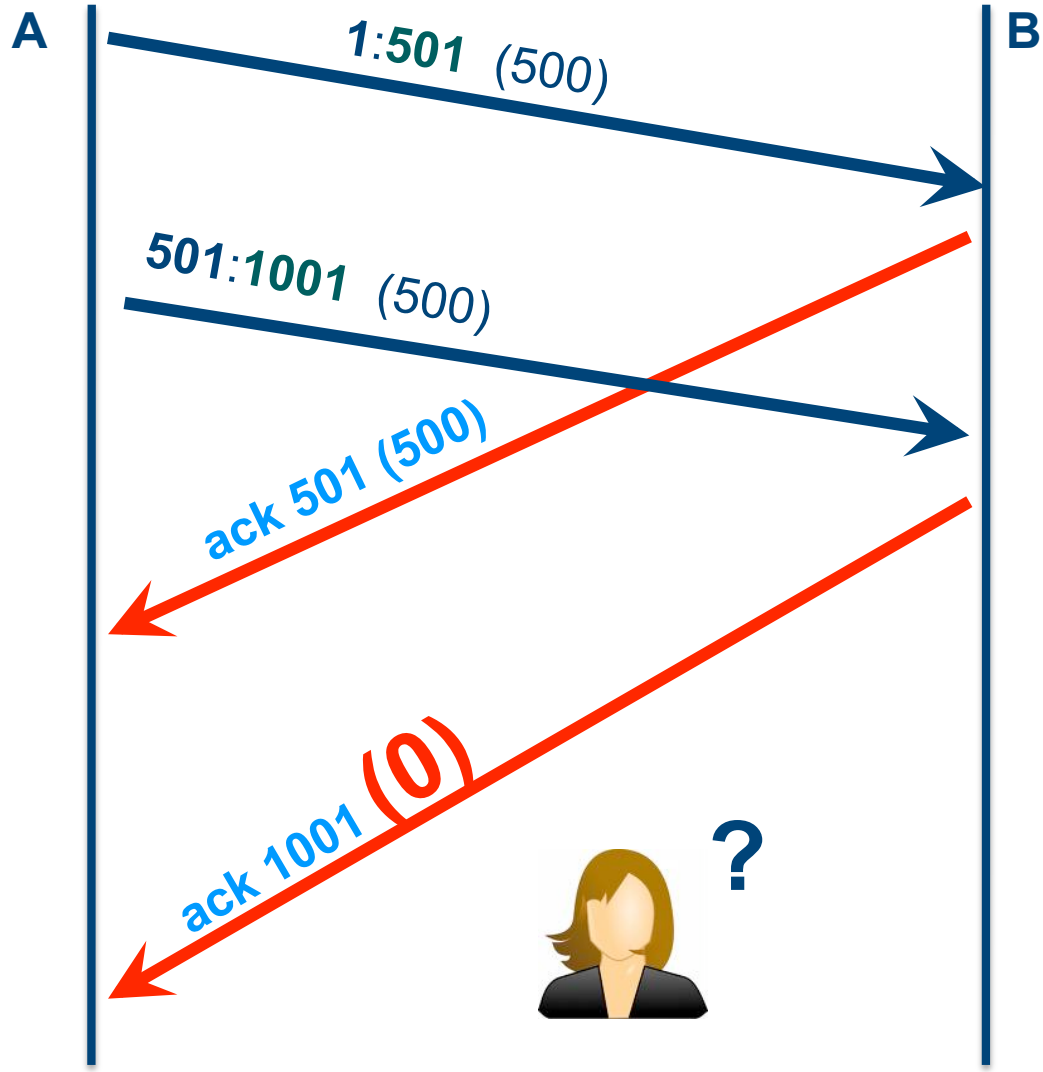
TCP

Les politiques d'acquittement

- **Le choix d'une politique doit s'adapter à l'application utilisant TCP**
  - Permettre une utilisation optimale des ressources du réseau
    - Moins gourmand possible en terme de bande passante
    - Meilleure réactivité aux pertes de segments et de congestion
  - Doit satisfaire les contraintes de l'application
    - Contraintes différentes lorsqu' s'agit
      - D'une application interactive
      - D'un transfert de masse

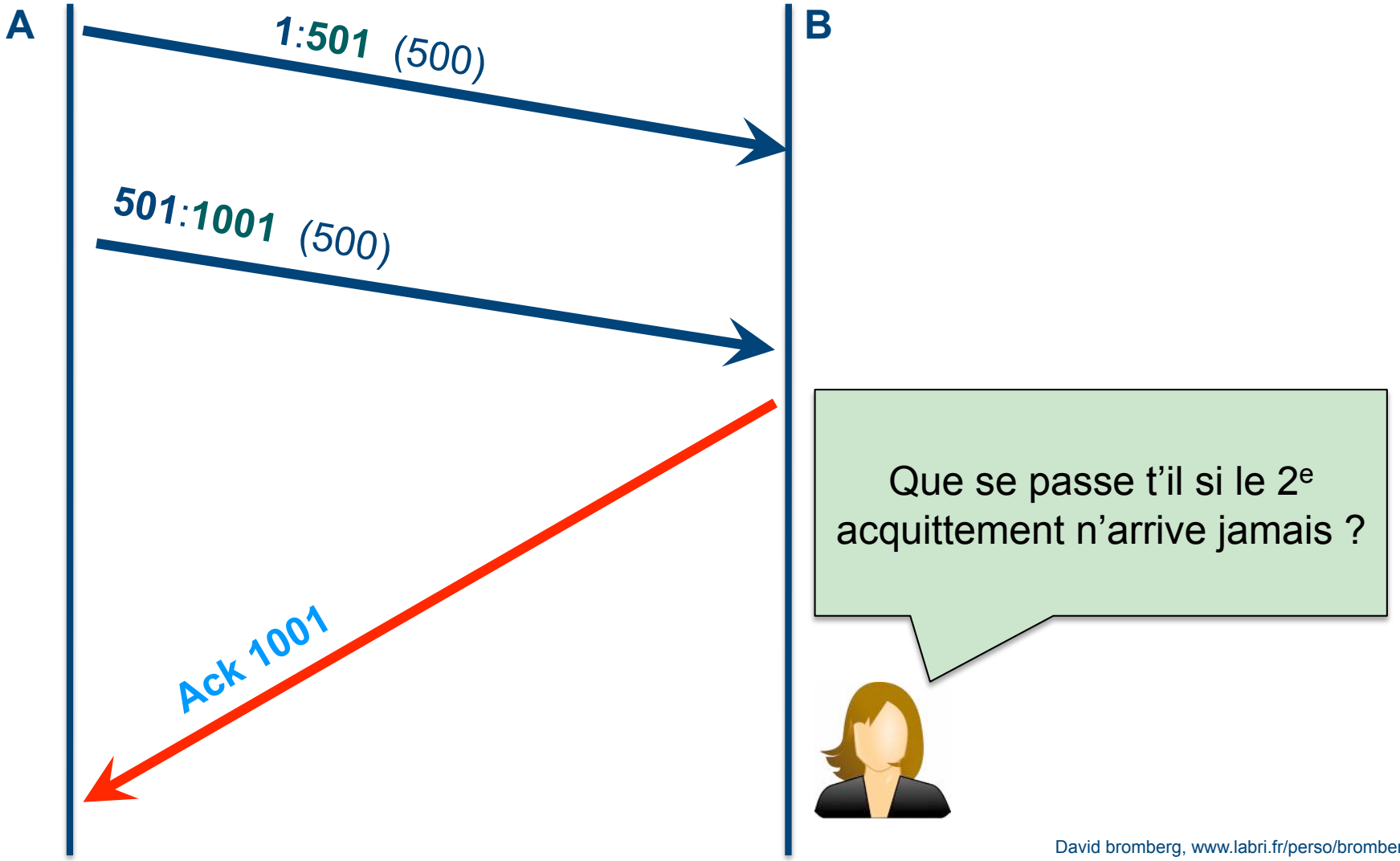
TCP

Acquittements immédiats

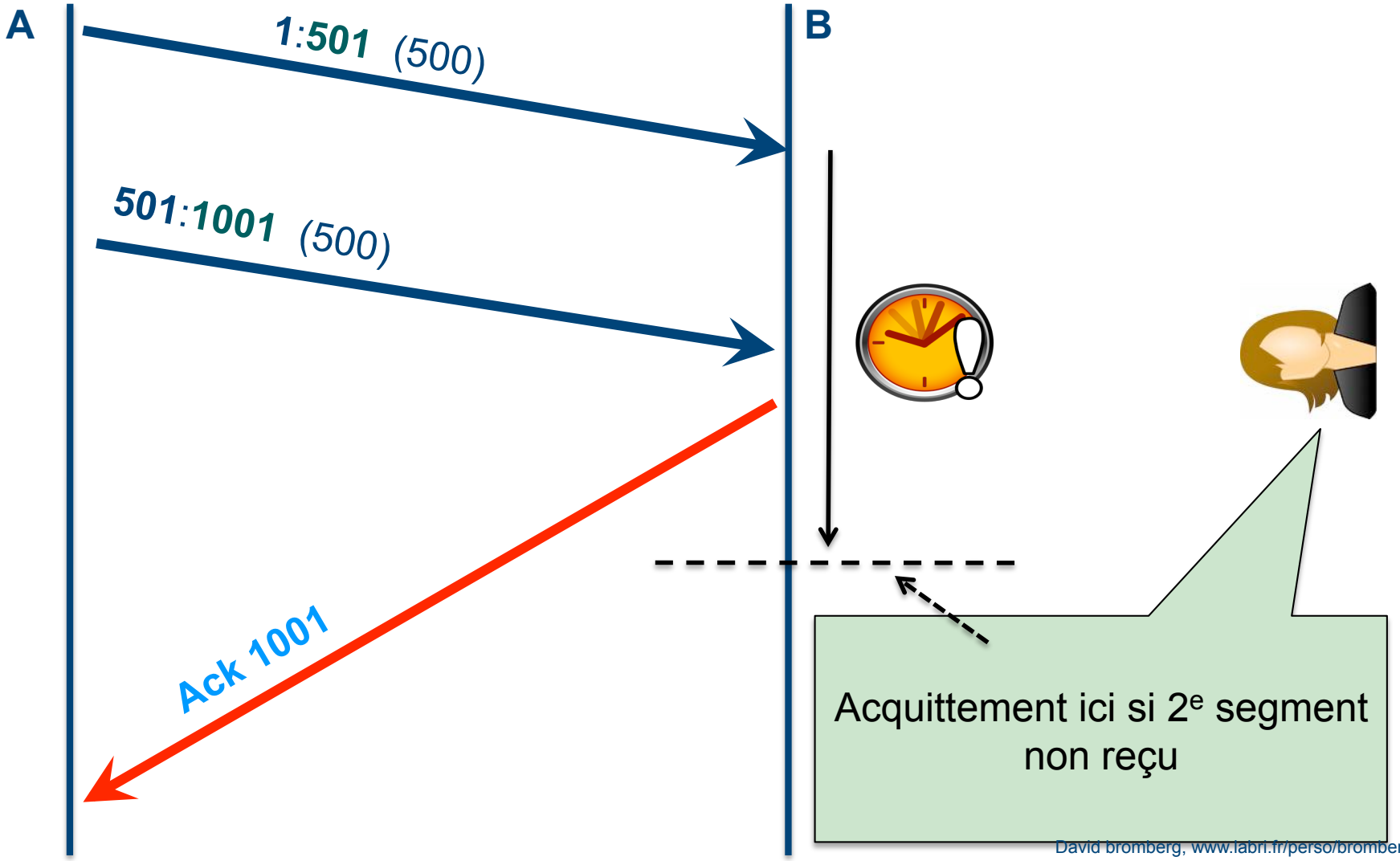


TCP

Acquittements cumulés

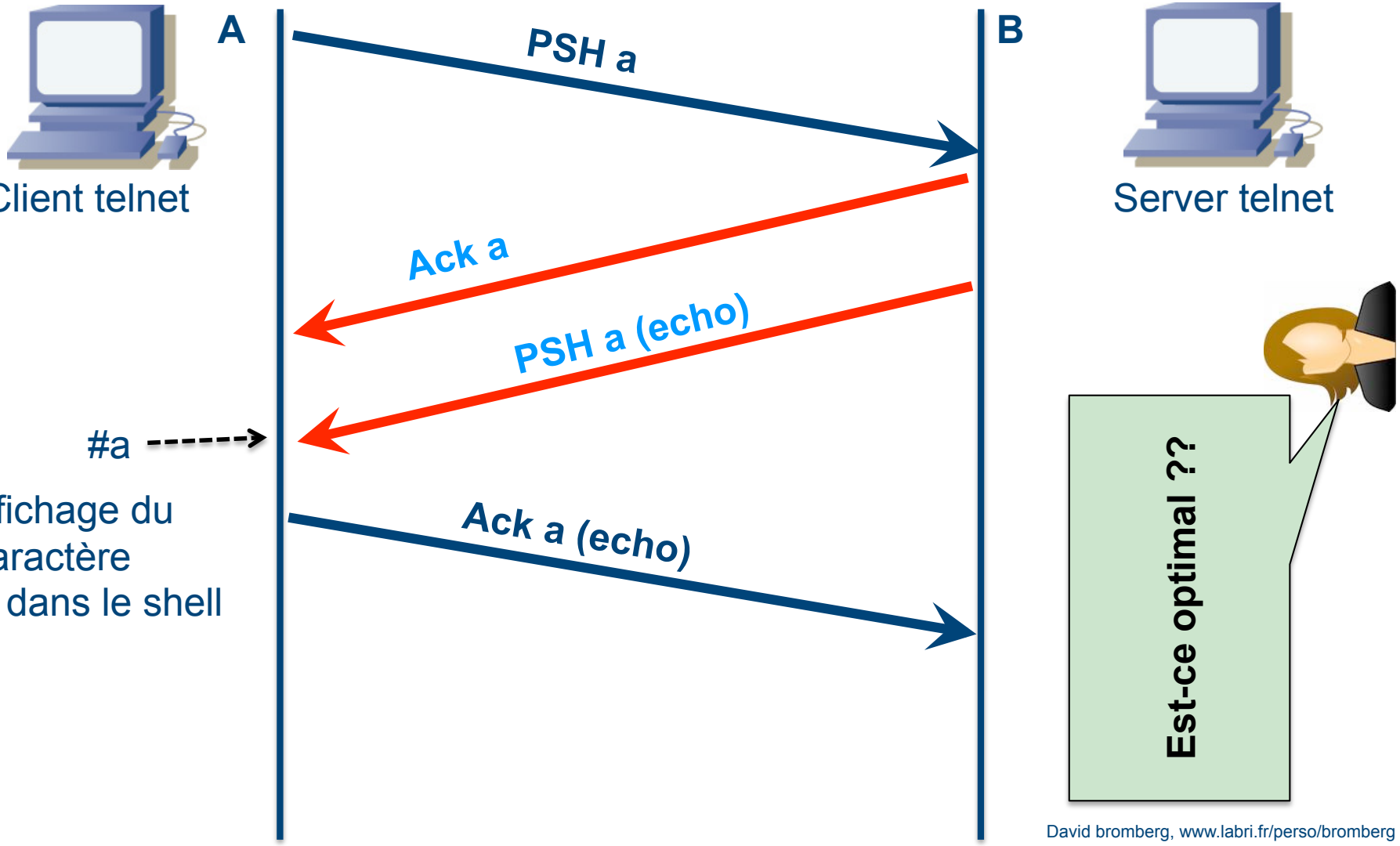


TCP
Acquittements cumulés et retardés



TCP

Exemple d'ack immédiat



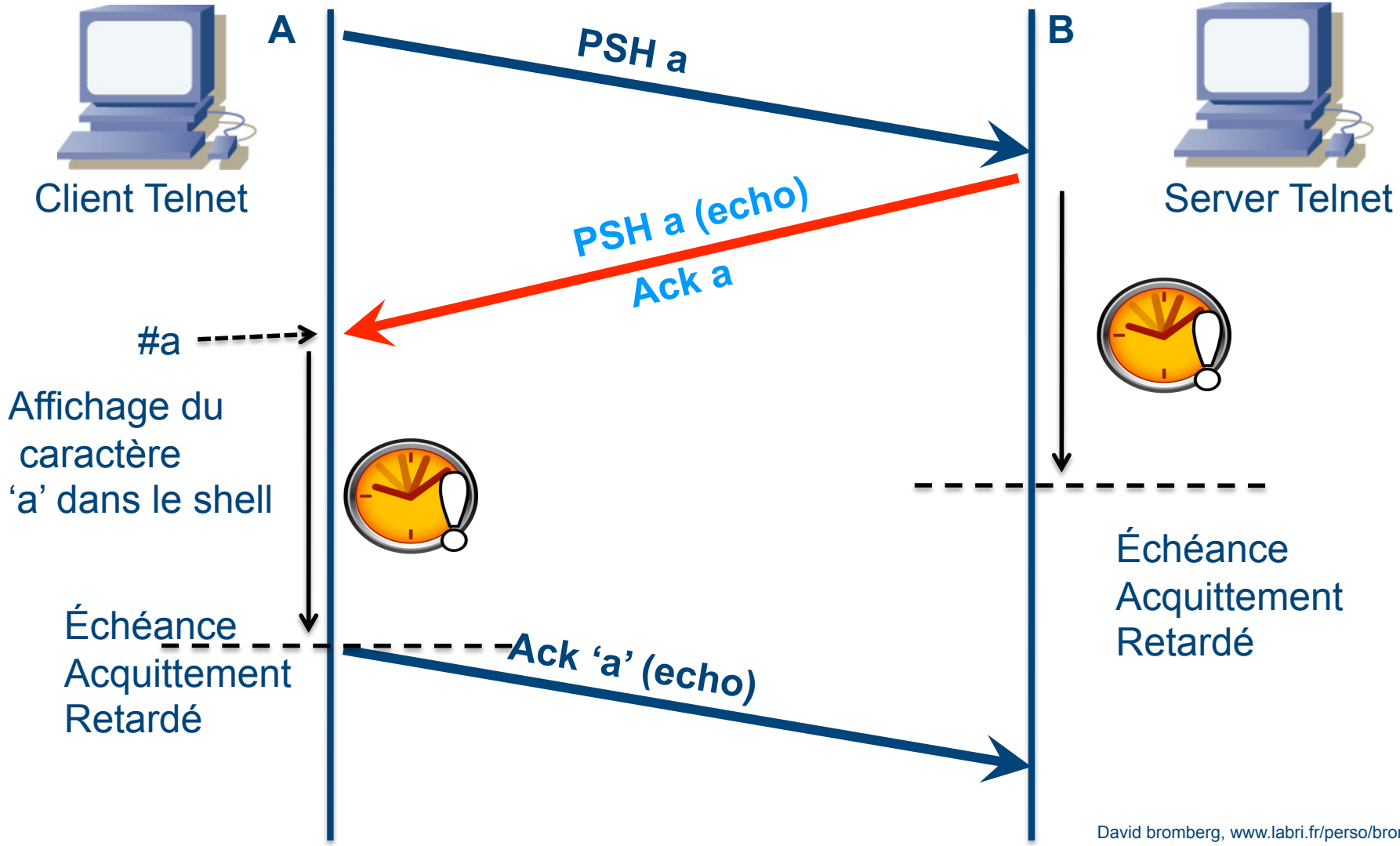
TCP

ACK immédiat, Détails

- A la frappe du caractère ‘a’ par l’utilisateur
  - Le client Telnet transmet un segment TCP et le marque PSH
    - => caractère transmis immédiatement
  - Si l’on compte les entêtes :
    - Transmission de ce segment consomme **41 octets**
- Le serveur Telnet renvoie immédiatement :
  - Un acquittement (Drapeau PSH) => **40 octets**
  - Après quelques millisecondes de traitement
    - L’echo est transmis dans un nouveau segment =>**41 octets**
    - Provoque de la part du client un nouvel acquittement =>**40 octets**
- **Pour un caractère ‘a’**
  - **=> transmission de 4 segments**
  - **=> 162 octets**

TCP

Exemple ACK retardé



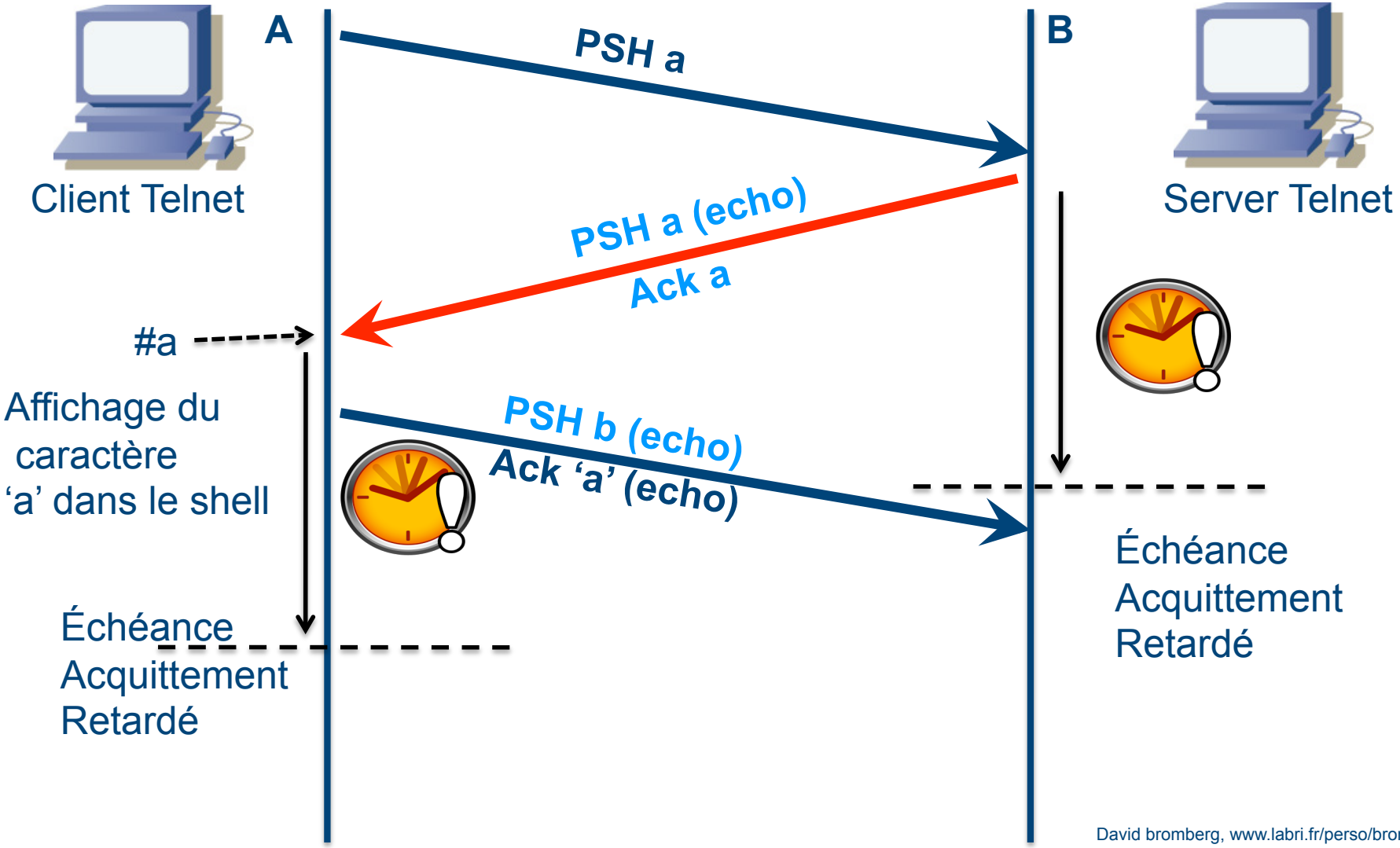
TCP

ACK retardé, Principe

- Chaque entité gère un temporisateur dès qu'il y a des segments à acquitter
  - Le retard d'acquittement est de ~200ms
- Une entité envoie un acquittement quand :
  - Le temporisateur d'ACK a expiré
  - Deux segments ont étés reçus
  - Des données sont à transmettre
    - => Cumul des données avec l'acquittement

TCP

ACK retardé, Réseau local



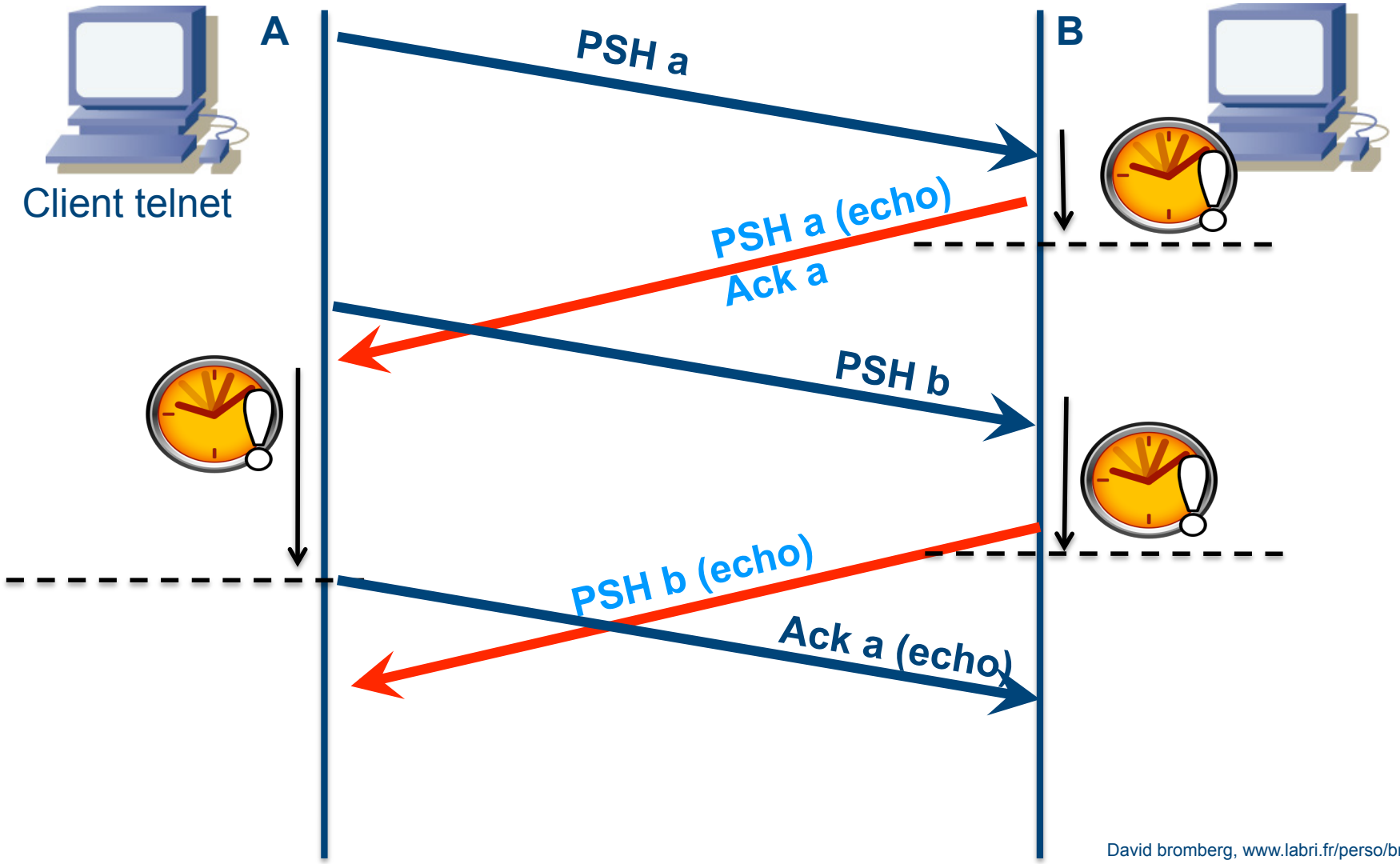
TCP

ACK retardé, Réseau local

- Frappe de deux caractères
  - La frappe du 2e caractère s'effecture pendant la période du retard de l'acquittement de l'echo du caractère 'a'
  - ⇔ C'est la transmission du caractère 'b' qui provoque l'écho du caractère 'a'
  - La frappe d'un caractère = 2 segments transmis
    - => 82 octets transmis
  - **La politique de l'ACK retardé est le meilleur compromis entre occupation de la bande passante et l'interactivité de l'application Telnet**

TCP

ACK retardé, Réseau WAN



TCP

ACK retardé, Réseau WAN

- WAN => Réseau grande distance où les délais d'aller/retour sont longs
  - La Frappe du 2e caractère se produit avant l'acquittement du 1er caractère
  - Le 2e caractère est transmis immédiatement et n'est pas cumulé avec un acquittement
  - **La transmission de chaque caractère provoque l'émission de 3 segments => 122 octets**

TCP
Optimisation

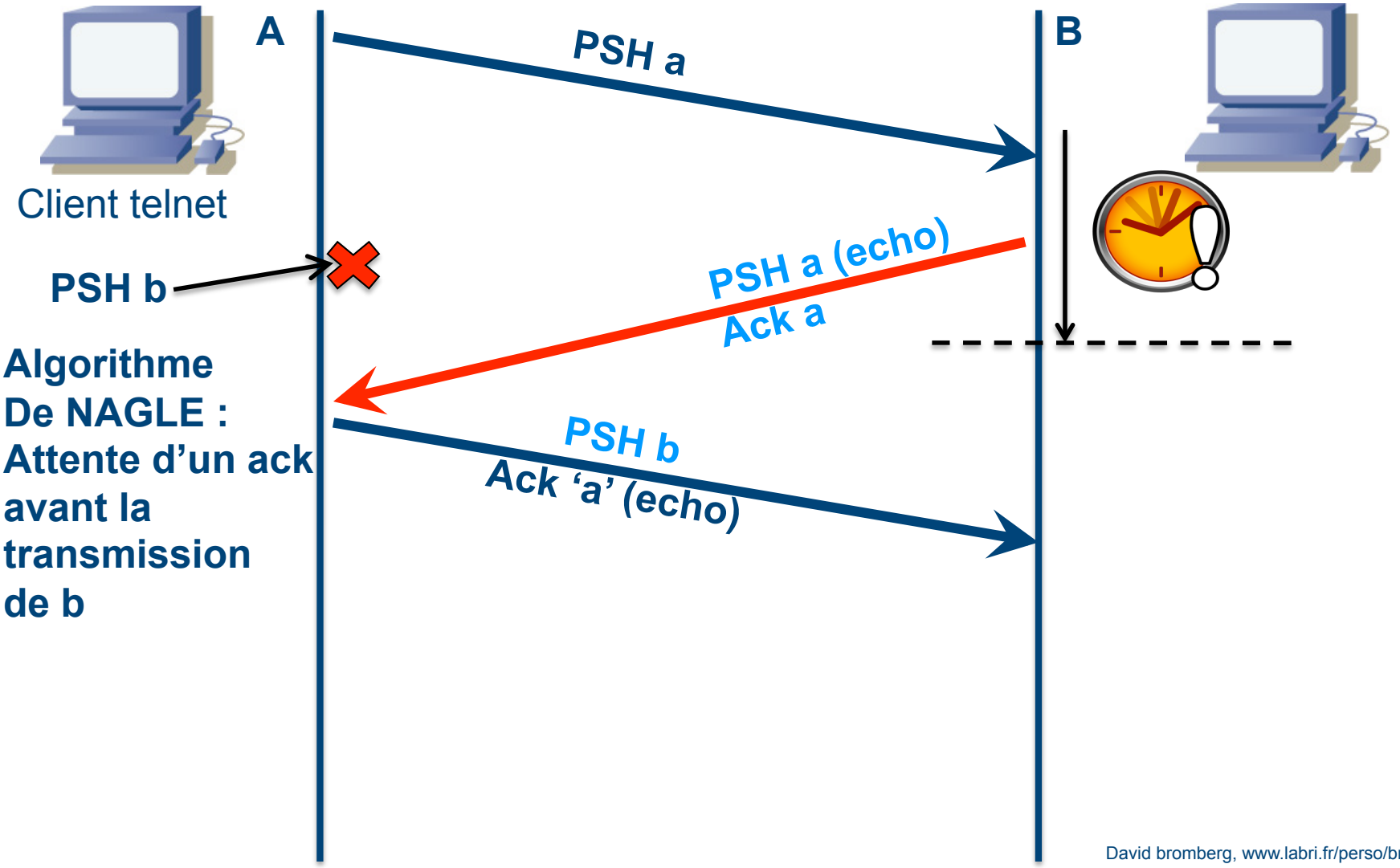
- La prolifération de petits paquets va surcharger les routeurs.
- Les routeurs :
  - Le temps de traitement lié à la consultation des tables de routage

TCP

Rappel

- **Petits paquets**
  - ⇔ **Surcharge des routeurs**
  - ⇔ **Augmentation de la taille des files d'attentes**
  - ⇔ **Augmentation des risques de pertes**
  - ⇔ **Augmentation des retransmissions**
  - ⇔ **Congestion du réseau.**

TCP
Algorithme de Nagle, Réseau WAN



Algorithme de Nagle

RFC 896

- L'algorithme précise que l'on ne peut émettre sur le réseau qu'un seul paquet de petite taille non acquitté.
- Deux cas se présentent donc :
  - Le réseau est lent.
    - TCP accumule dans un même buffer les octets en partance.
    - Dès réception de l'acquittement il y a émission du contenu du buffer en un seul paquet.
  - Le réseau est rapide.
    - Les acquittements arrivent rapidement
- La qualité lent/rapide du réseau est calculée à partir du 'timestamp' envoyé dans les options de TCP et qui est établi dès le premier échange (puis réévaluée statistiquement par la suite).

Contrôle de congestion

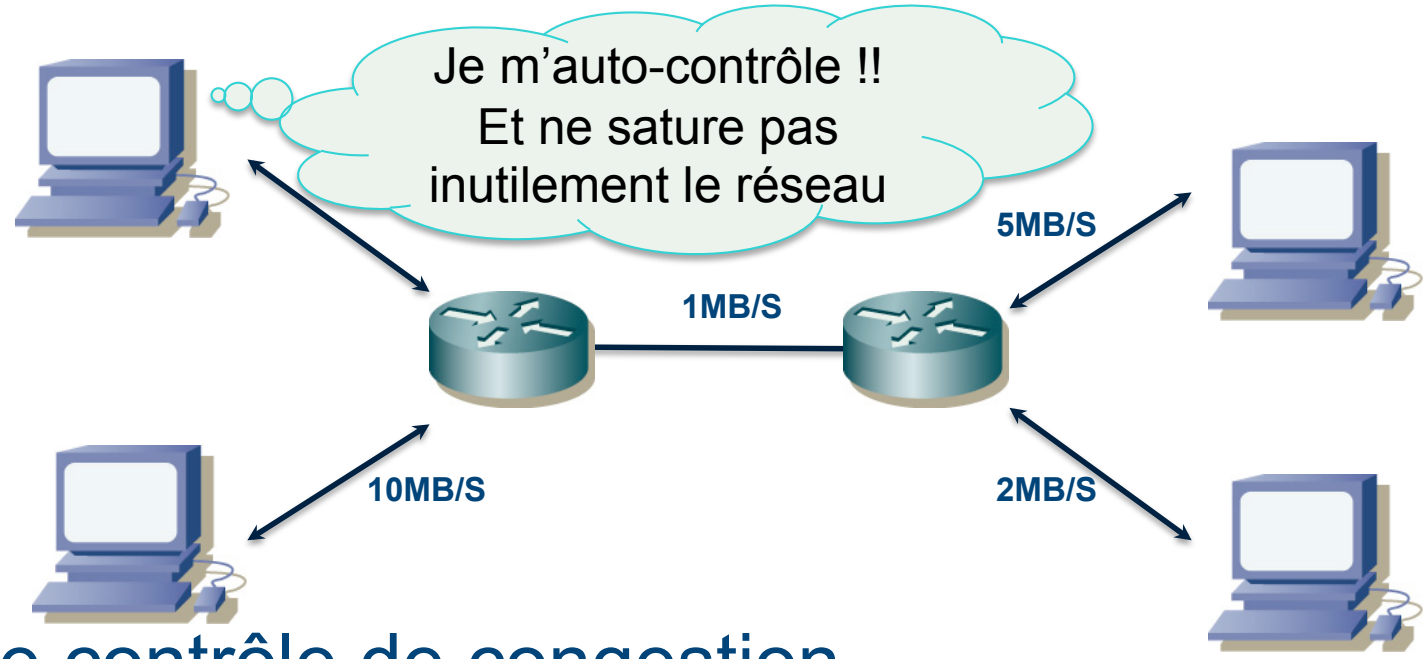
- Evite la transmission de segments sur le réseau alors qu'il est congestionné
- Optimise le taux d'utilisation des ressources du réseau

TCP

Contrôle de congestion

- Un paquet est réémis parce qu'il :
  - Arrive corrompu
  - N'arrive jamais.
    - ⇔ Routeurs congestionnés, les files d'attente saturée.
- ⇔ Une réémission entraîne un blocage de l'avancement de la "fenêtre glissante", pénalisant pour le débit.

Contrôle de congestion



- Le contrôle de congestion
  - Ne requiert aucune signalisation avec le réseau
  - Auto-discipline que la source s'impose à elle même
  - Basé sur les manifestations indirectes de la congestion
    - Perte de segments

Contrôle de congestion

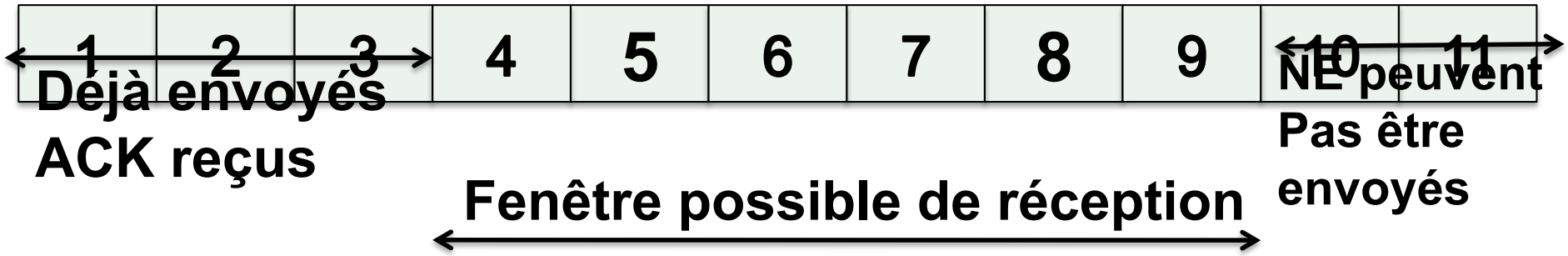
- Une entité source TCP dispose de 3 états de contrôle de congestion
  - Démarrage lent (Slow Start)
  - Evitement de congestion (Congestion Avoidance)
  - Recouvrement rapide (Fast Recovery)
- La détection d'une perte de segments implique un changement d'état de la source

TCP TAHOE et TCP RENO

- **TCP TAHOE**
  - Slow start
  - Congestion avoidance
- **TCP RENO**
  - TAHOE + fast recovery

TCP

Rappel, Fenêtre glissante



Envoyés
ACK non reçus

Peuvent être
Envoyés

TCP

Contrôle de congestion, Principe

- **Au début de la connexion**
  - Chaque entité TCP se met dans l'état **Slow Start**
    - Démarrage prudent
    - Sonde l'état du réseau par l'envoie d'un seul segment
      - Cwnd =1, puis au retour de l'acquitement, cwnd=2, puis 4, etc…
- **Arrivé à un seuil**
  - L'entité TCP infléchit le rythme de croissance de cwnd
    - Passe d'une croissance exponentiel à linéaire
    - => Etat **Congestion Avoidance**
- **Si une perte est détectée**
  - **Avec un timeout**
    - Passage dans l'état **Slow Start**
  - **Avec des acquittements dupliqués**
    - Passage dans l'état **Fast Recovery**

TCP

Contrôle de congestion, Principe

- **Fenêtre de congestion cwnd**,
  - Doit être considérée comme un droit d'émettre **cwnd** segments sur une période d'un RTT
  - Emission par salves de cwnd segments
    - Puis attente du 1er acquittement

TCP

Contrôle de congestion

On souhait optimiser la consommation de la bande passante disponible
Mais aussi limiter la congestion du réseau

- **Dans ce contexte,**
  - On comprend bien qu'il vaut mieux ne pas envoyer la totalité du contenu de la fenêtre dès le début de la connexion aussi rapidement que l'autorise le débit théorique du réseau.

⇔ Asservissement de l'émission des paquets au rythme de la réception de leurs acquittements.

- **Technique du Slow START** :
  - A chaque fois que l'émetteur reçoit un acquittement pour un segment qu'il a émis, il augmente la taille de la fenêtre de congestion

TCP

Phase du slow start

- **Phase de démarrage**
  - La taille de la fenêtre de congestion **cwnd** est initialisée à 1 MSS
- **Suite à**
  - L'envoie d'une salve de **cwnd** segments
  - Au retour du premier acquittement
    - La fenêtre cwnd est incrémentée de 1

TCP

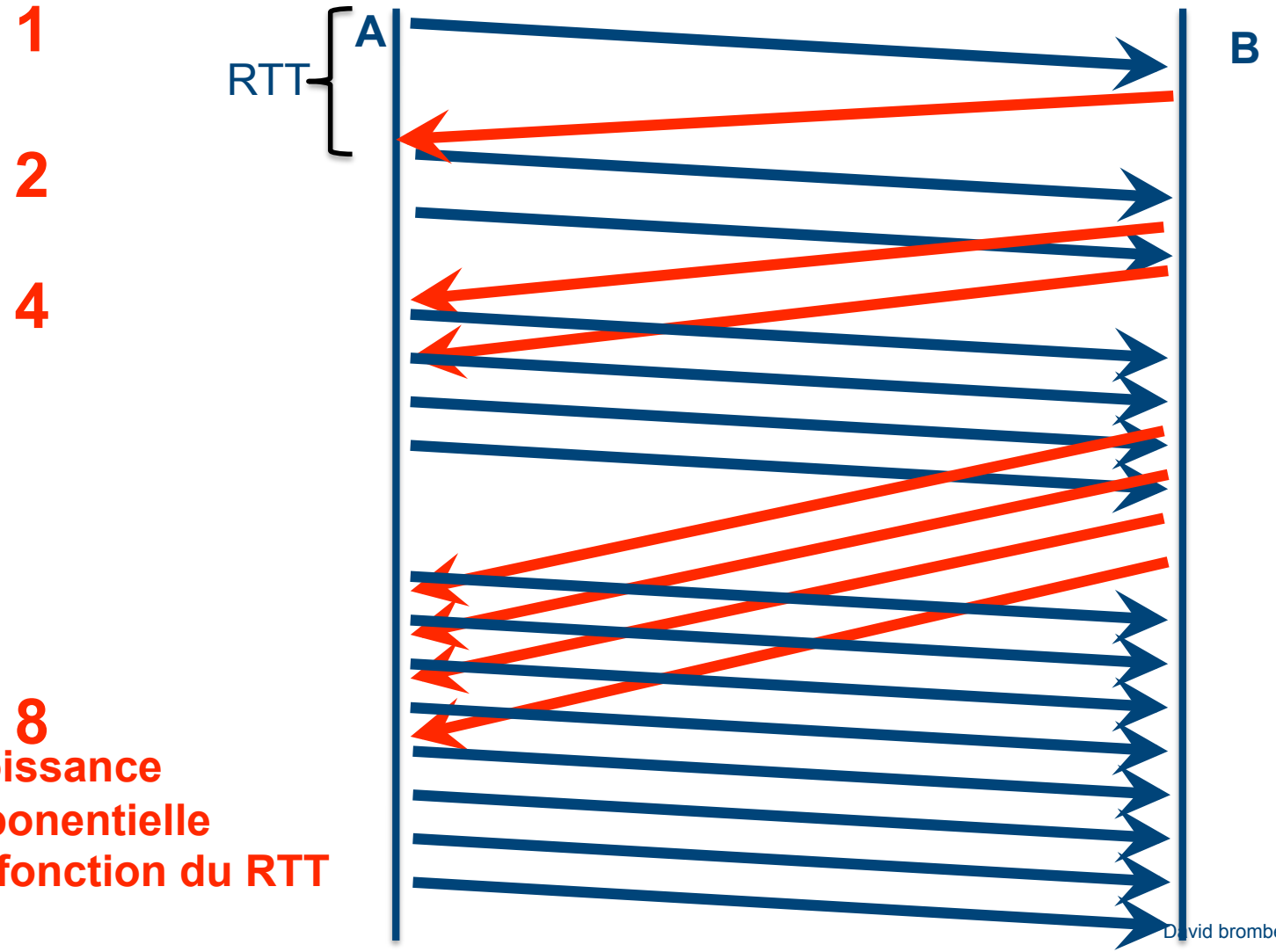
Phase du slow start

- **A la réception de chaque acquittement de la salve :**
  - cwnd = cwnd + 1
  - ⇔ Ajustement de la taille de la fenêtre de congestion
    - En fonction du taux d'arrivée des acquittements du réseaux
      - et des capacités instantanées du réseau

⇔Augmentation exponentielle de la fenêtre

TCP

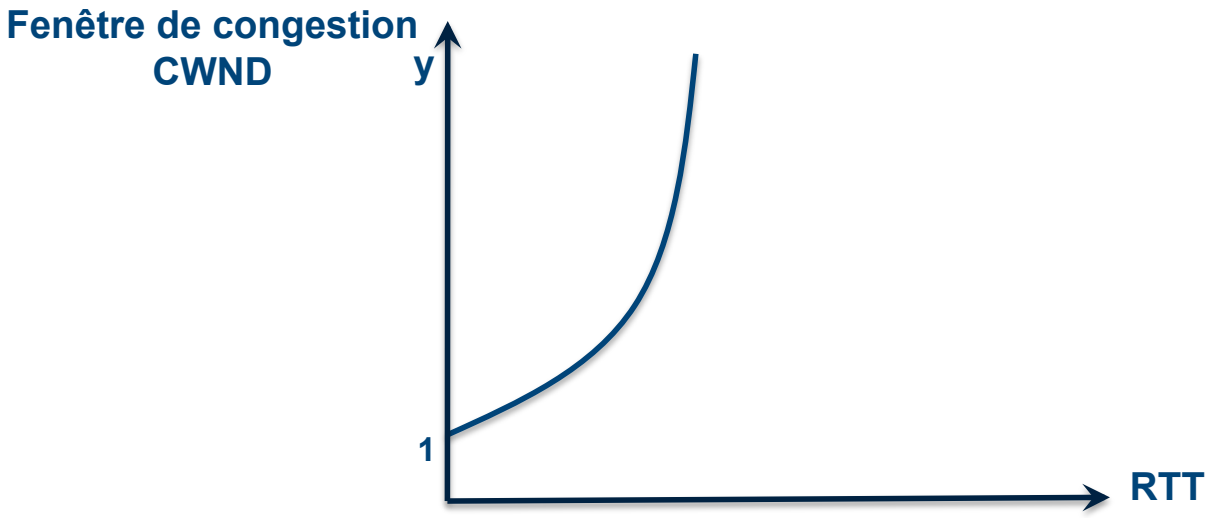
Slow start, principe (1/2)



Croissance
Exponentielle
En fonction du RTT

TCP

Slow start, principe (2/2)



TCP
Congestion Avoidance

- Utilisation d'un seuil **threshold** = 65536 octets
- **Dans la phase Slow Start**
  - A la réception d'un segment de taille MSS
    - Si Cwnd * MSS < threshold : rester dans l'état slow start
    - Sinon
      - Cwn =cwnd + 1/cwnd
      - Passer à l'état **Congestion Avoidance**

TCP
Congestion Avoidance

- **Dans l'état Congestion Avoidance**
  - A la réception d'un acquittement d'un segment de taille MSS
    - Faire cwnd = cwnd + 1/cwnd
    - Rester dans l'état **Congestion Avoidance**

TCP

Congestion Avoidance

En phase Slow Start la fenêtre de congestion augmente exponentiellement

⇔ Risque d'une saturation rapide du réseau

TCP s'impose un infléchissement du rythme d'augmentation de la fenêtre des lors que cwnd arrive à un certain seuil

⇔ Rythme d'augmentation linéaire

TCP

Congestion Avoidance

Cwnd N

N

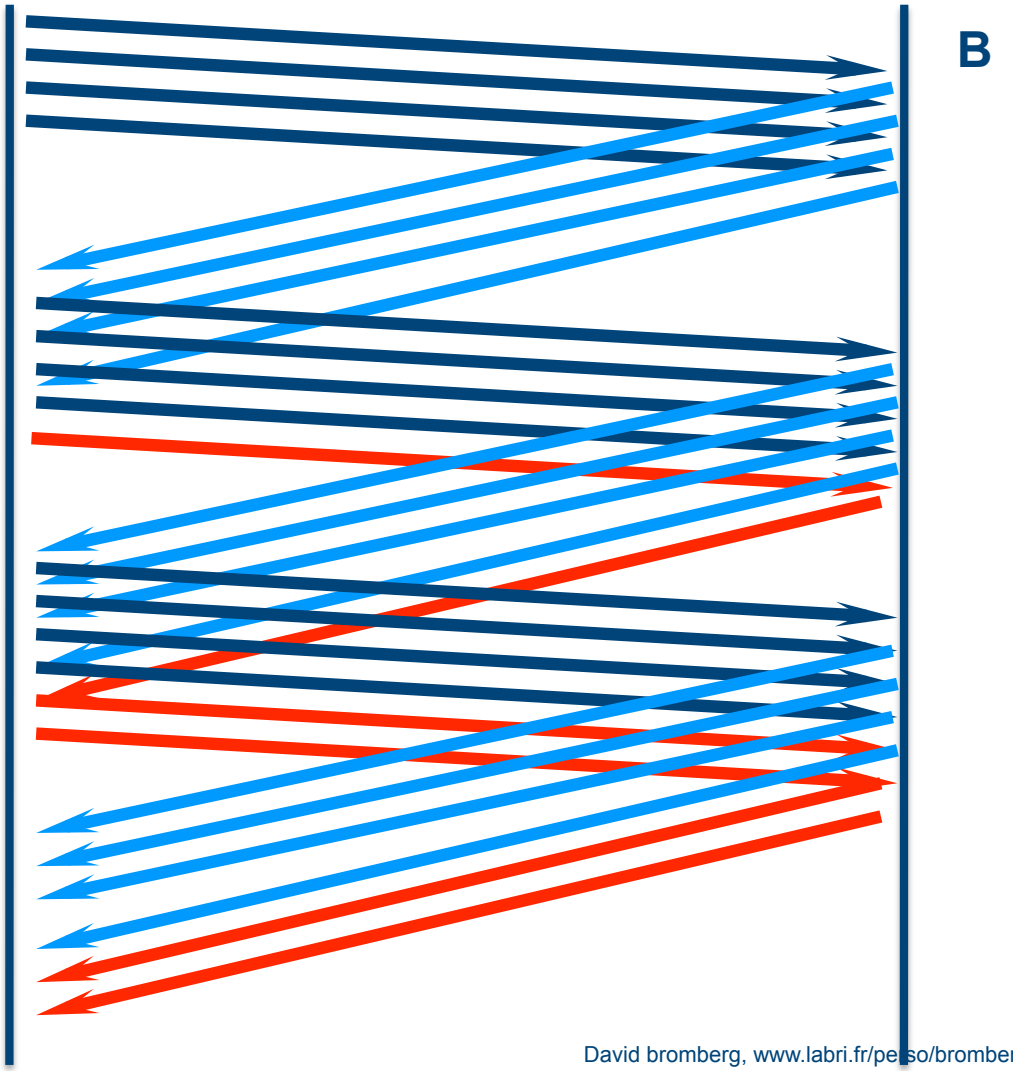
Cwnd = threshold
⇔ Congestion Avoidance

N+1

N+2

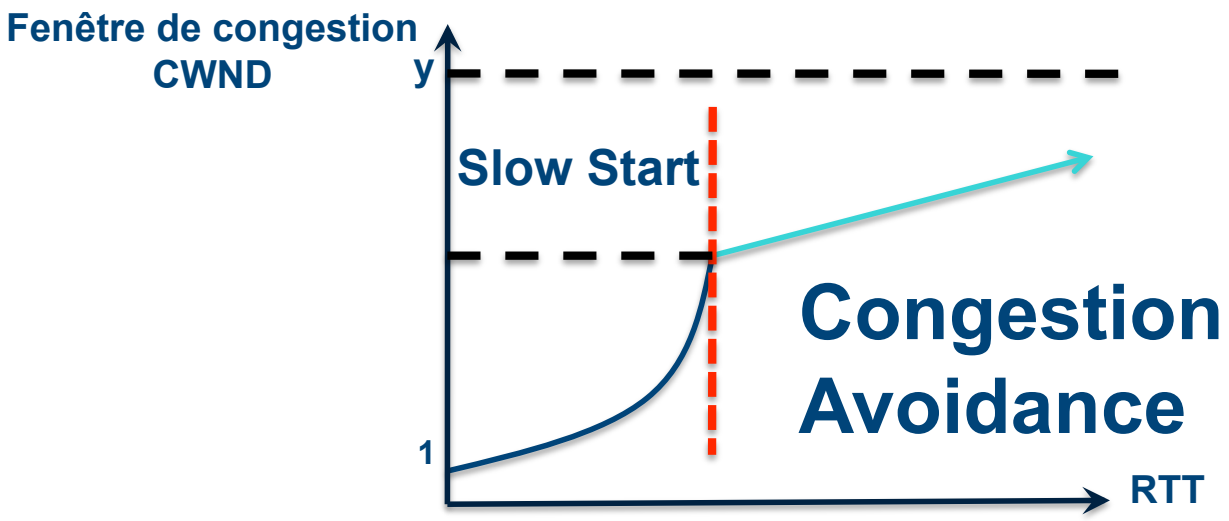
Progression Linéaire

N+3



TCP

Congestion Avoidance



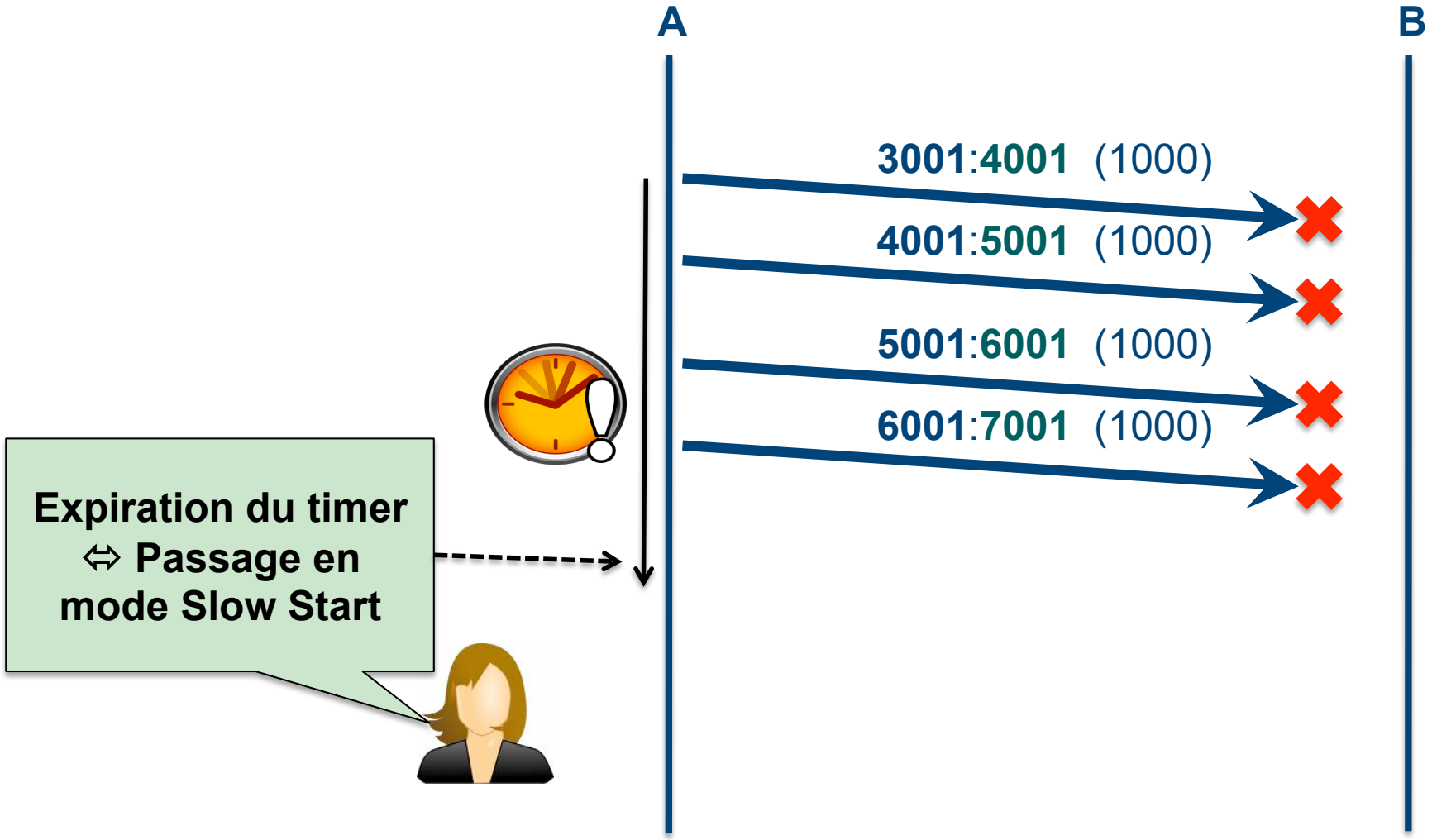
TCP

Détection de la congestion (1/3)

- A l'échéance du temporisateur de retransmission
  - Cwnd =1
  - Threshold = max(cwnd/2, 2*MSS)
  - Passer à l'état

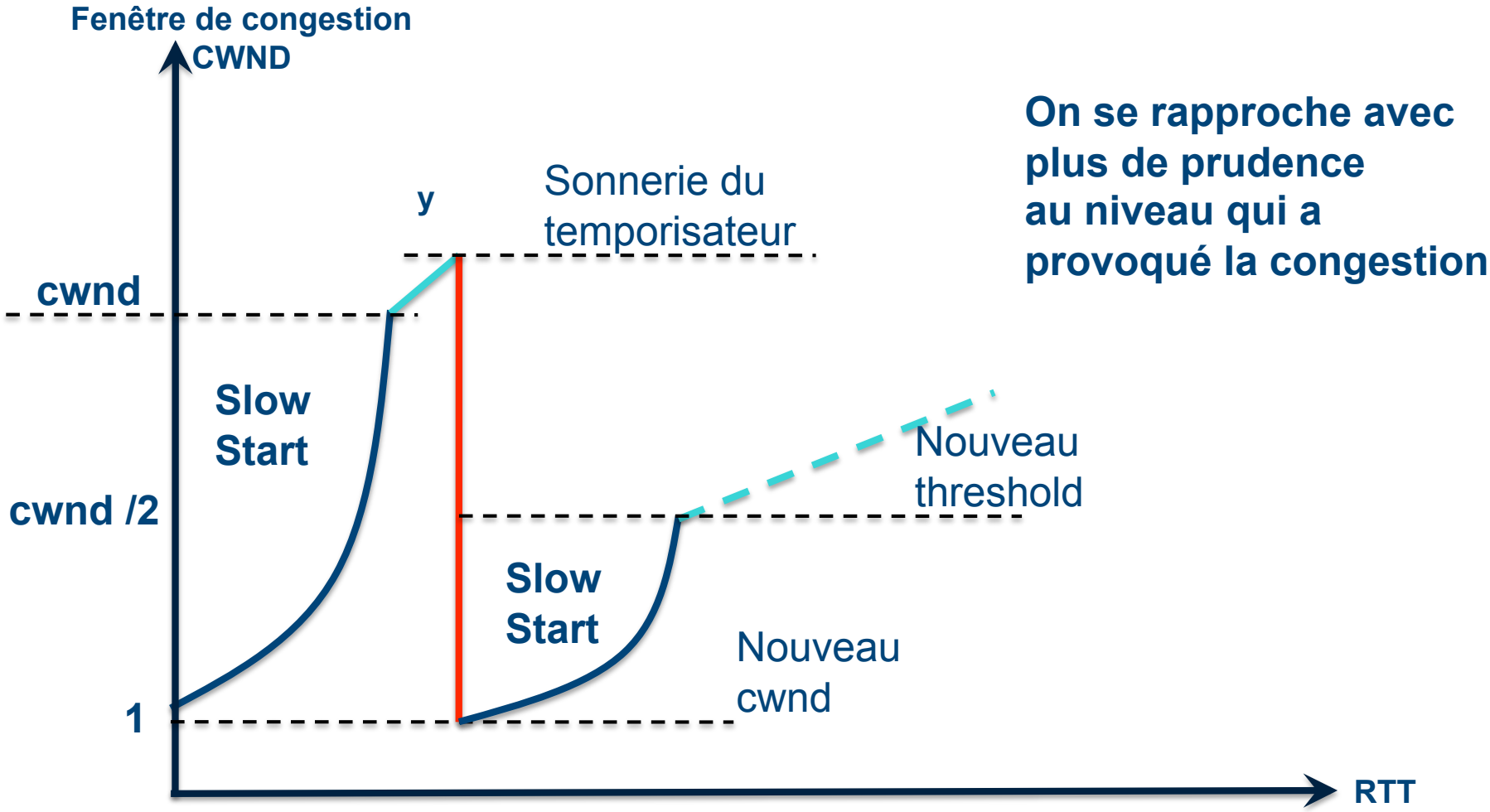
TCP

Retour vers l'état Slow Start



TCP

Retour vers l'état Slow Start



On se rapproche avec plus de prudence au niveau qui a provoqué la congestion

TCP

Détection de la congestion (2/3)

- L'entité a déjà envoyé n+k segments et a reçu un acquitement ack(n) réclamant le segment n

- L'entité reçoit 3 acquittements dupliqués ack(n)
  - Threshold = Max (cwnd/2, 2*MSS)
  - Cwnd = Min (cwnd/2 +3, 65535)
  - Retransmet le segment n
  - Passer à l'état Fast Recovery

TCP

Détection de la congestion (3/3)

- **Si détection via la réception d'acuittements dupliqués**
  - Le réseau est considéré comme ponctuellement ou modérément congestionné
  - On reçoit des acquittements, donc le réseau a encore des ressources

Ok ! Donc peut être qu'un seul segment s'est perdu ☺

TCP

Fast Recovery

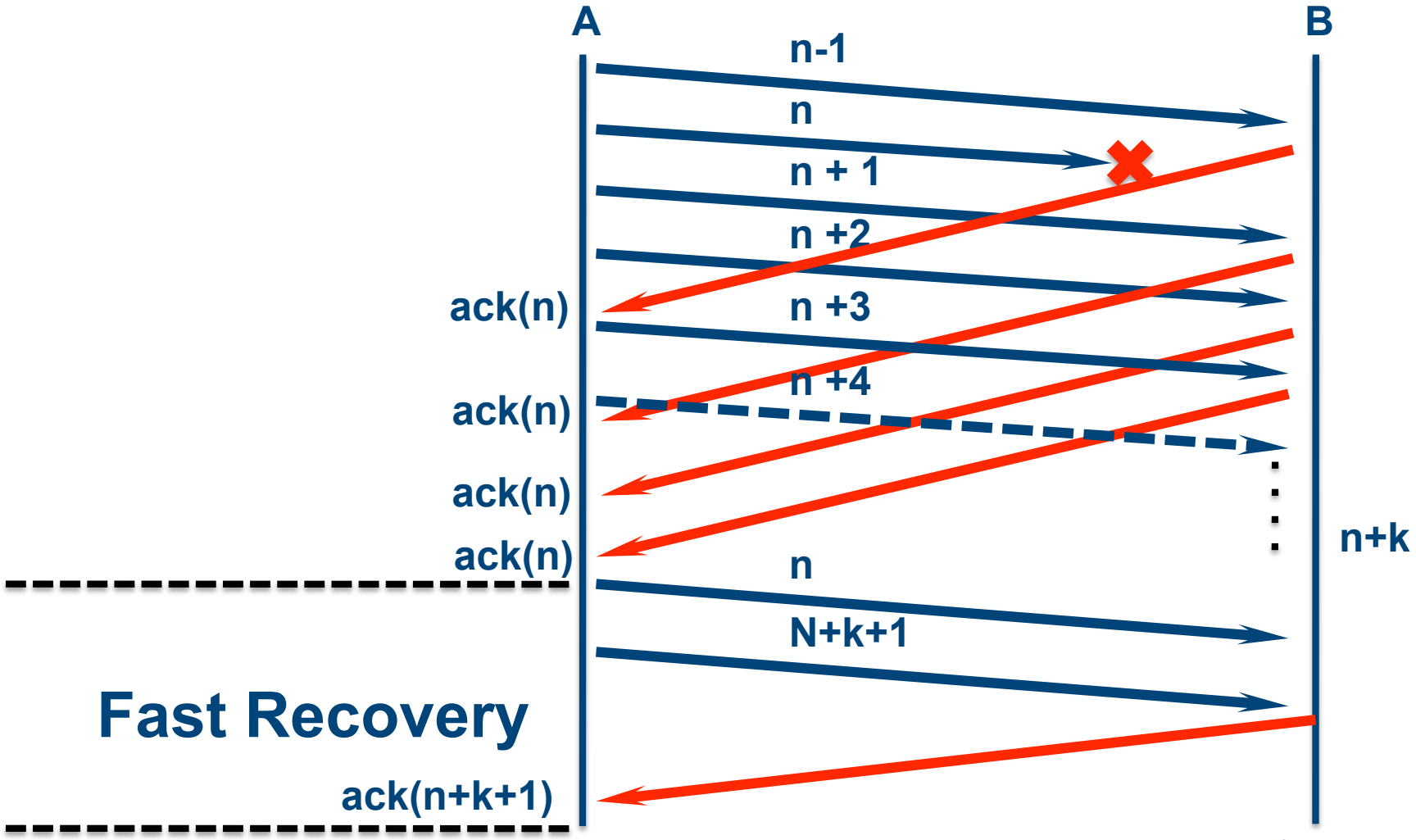
- **La réaction doit donc être proportionnée**
  - Passage dans l'état **Fast Recovery**
  - Chute moins brutale que dans le **Slow Start**
  - **L'état Fast recovery est transitoire**
    - **On y sort dès que le segment perdu n a été acquitté**

TCP
Fast Recovery

- **Comportement à la Slow Start**
  - **Avec un cwnd initial de Min(cwnd/2 + 3, 65535)**
- **Réception d'un acquittement dupliqué ack(n)**
  - Cwnd = cwnd + MSS
  - Continuer l'envoie de segments > n+k
- **Réception de l'acquittement ack > n**
  - Cwnd = threshold/2
    - Mémorisé avant le passage à **Fast Recovery**
  - Passage à l'état **Congestion Avoidance**

TCP

Fast Recovery



TCP

Fast Recovery