Couche Transport – Sommaire

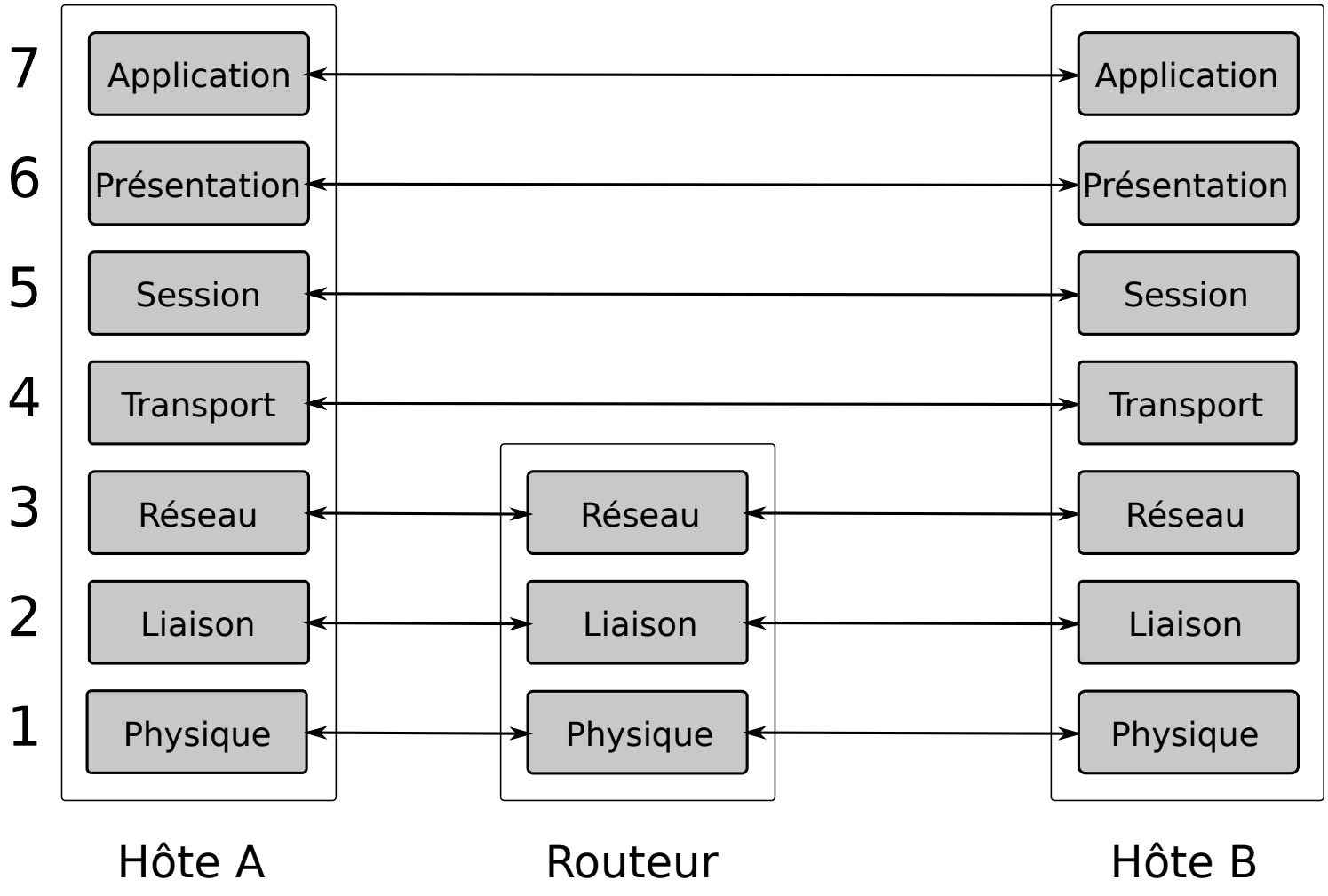
But

- La couche réseau offre une communication de machine à machine.
- La couche transport permet :
  - une communication d'application à application,
  - un transfert fiable :
    - sans corruption : checksum
    - sans pertes : FOO BAR $\nrightarrow$ BAR
    - dans l'ordre : FOO BAR $\nrightarrow$ BAR FOO
    - sans duplication : FOO BAR $\nrightarrow$ FOO FOO BAR
  - des services avec ou sans connexion,
  - d'offrir différentes qualités de service (QoS) pour le mode connecté,
  - une communication de bout en bout.

Connecté/Non Connecté

- Mode connecté : établissement d'une connexion avant transmission puis transmission de données et libération de la connexion (téléphone)
- Mode non connecté : transmission de données (courrier postal)

Transport

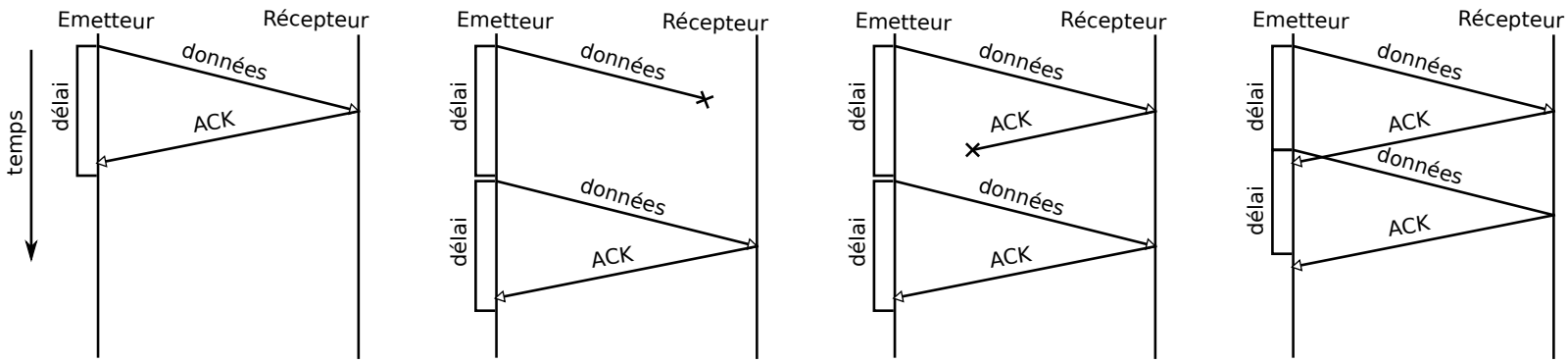

7
Application
6
Présentation
5
Session
4
Transport
3
Réseau
2
Liaison
1
Physique
Réseau
Liaison
Physique
Application
Présentation
Session
Transport
Réseau
Liaison
Physique
Hôte A
Routeur
Hôte B

Problèmes à résoudre

- Comment assurer la fiabilité du transport i.e. que toutes les données envoyées sont bien reçues ?
- Comment identifier les applications (pb d'adressage) ?
- Comment gérer plusieurs communications sur une même machine i.e. en utilisant une unique IP/MAC (pb de multiplexage) ?
- Quels mécanismes pour s'assurer l'établissement/la libération d'une connexion ?
- Comment éviter qu'une station soit surchargée et ne puisse plus répondre correctement (contrôle de flux) ?
- Comment éviter qu'un réseau soit encombré (contrôle de la congestion) ?

Transport fiable

- Principe général : *Automatic Repeat reQuest* (ARQ) : 1) on lance un **timer** à chaque envoi de données et on répète l'envoi tant que l'on n'a pas reçu un **acquittement** dans le temps imparti (i.e. si timeout)
- Mode "Basic" : on émet et on attend (*Stop-and-wait ARQ*)



- nécessité du *numéro de séquence* $\rightarrow$ identification des données

Transport fiable

- Problème : délai entre deux paquets égal au double du temps de transmission
- Solution : envoyer plusieurs paquets successivement sans attendre d'acquittement
- Avec plusieurs déclinaisons :
  - *Go-Back-N ARQ* : retransmission complète à partir de la détection de perte
  - *Selective Repeat ARQ* : possibilité d'acquitter/de redemander des données sélectivement

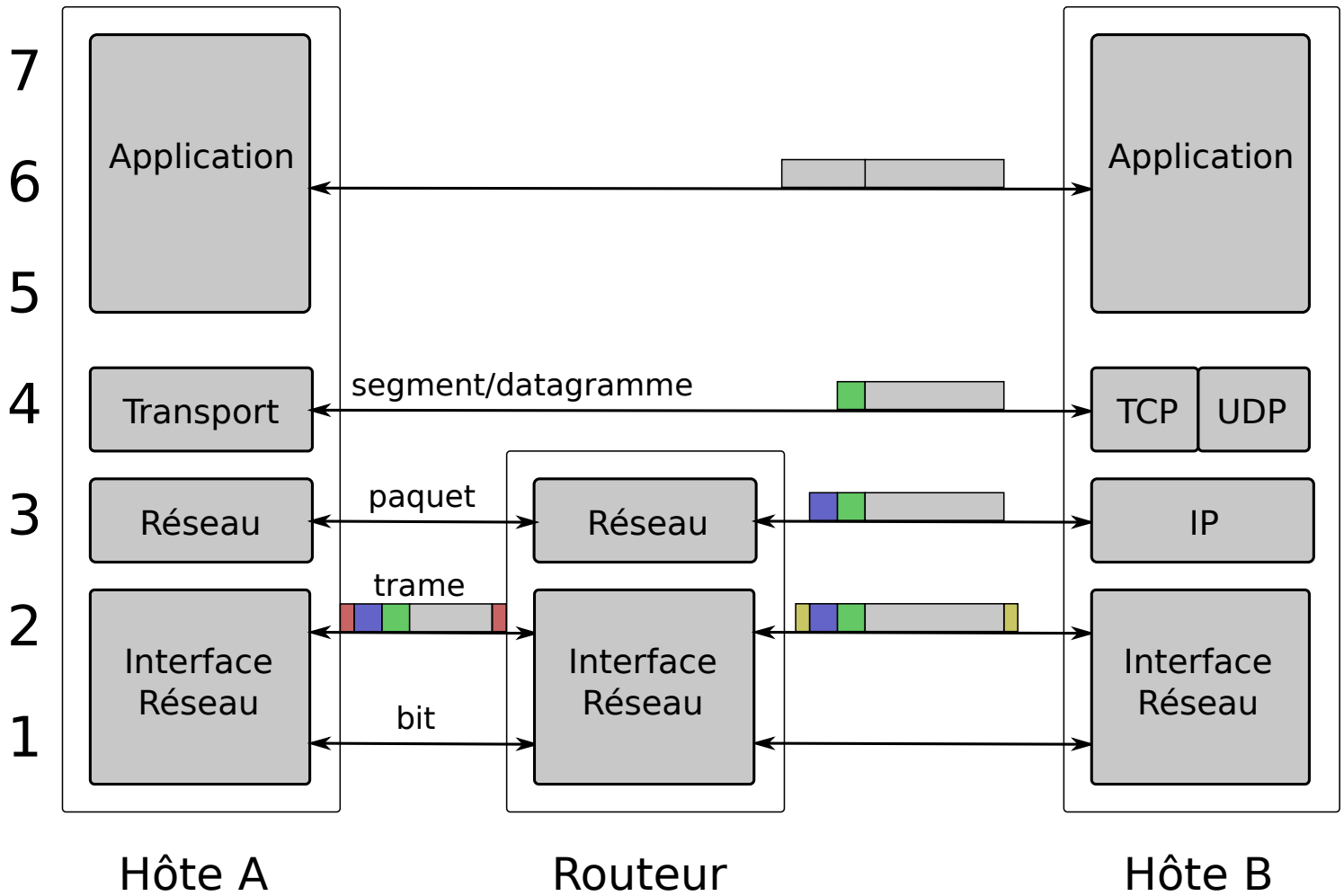
Flux vs Congestion

- contrôle de **congestion** :
  - problème global du **réseau** : éviter trop de trafic dans un sous-réseau
  - cas typique : réseau lent, nombreux terminaux transmettant de gros fichiers
- contrôle de **flux** :
  - problème d'une **station** : éviter qu'un émetteur rapide ne sature un récepteur lent
  - cas typique : réseau rapide, un terminal rapide transmet à un terminal lent

Transport dans TCP/IP

- deux protocoles :
  1. UDP *User Datagram Protocol* protocole en mode sans connexion, non fiable
  2. TCP *Transmission Control Protocol* protocole orienté connexion, fiable

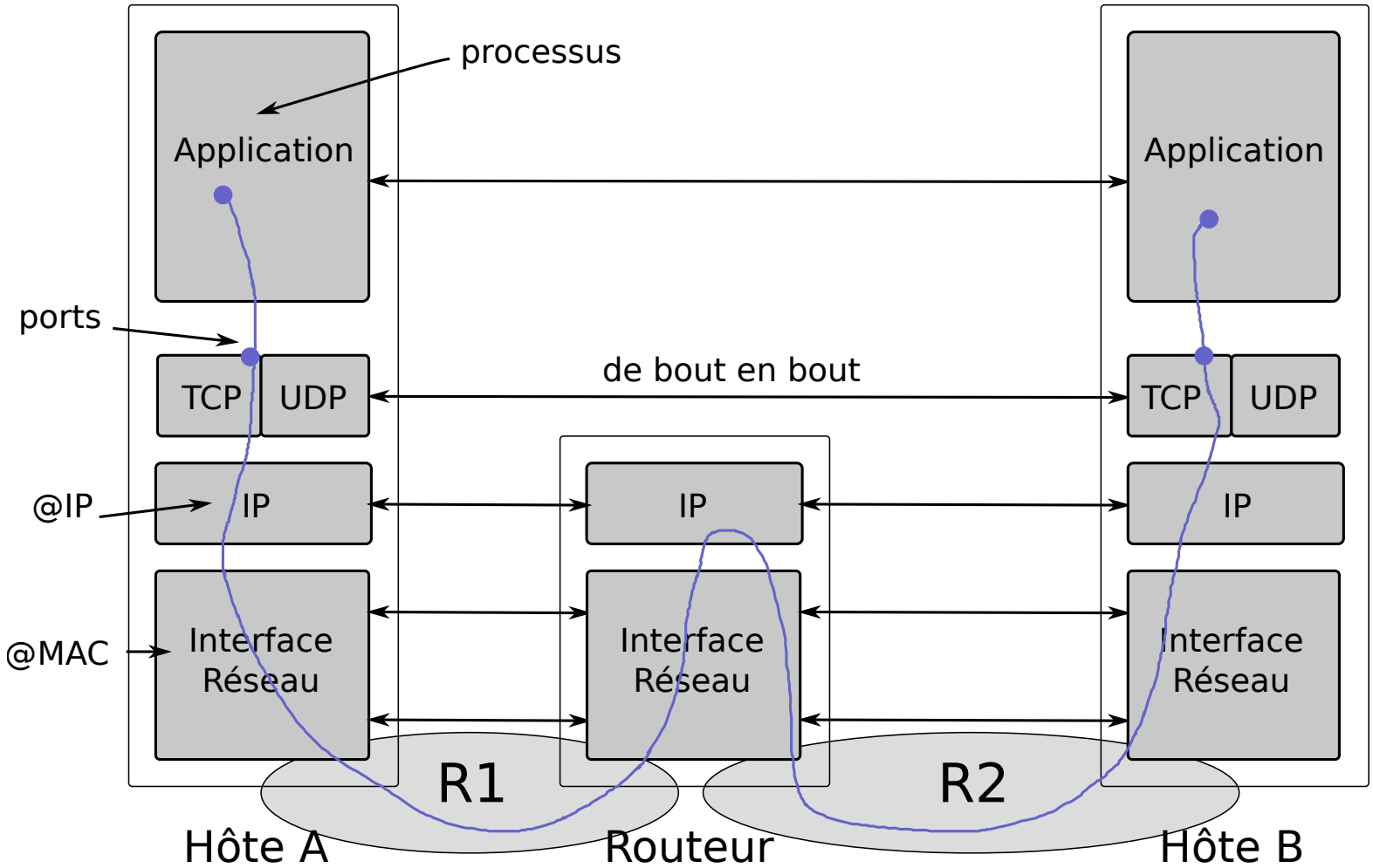
Encapsulation



Adressage : identification des services

- machine $\longleftrightarrow$ IP
- sur une machine : processus, application $\longleftrightarrow$?
- concept de **port**
- IP + port = **socket** (unique sur le réseau)
- l'envoi d'un message nécessite port source/port destinataire $\rightarrow$ **multiplexage** par l'expéditeur/demultiplexage par le destinataire (pour la station, la liaison sous IP est unique, les num de ports permettent de distinguer les sessions)
- interface permettant d'utiliser les ports $\rightarrow$ API Socket (C, java, ...)

Bout en bout


processus
Application
Application
ports
TCP
UDP
de bout en bout
TCP
UDP
@IP
IP
IP
IP
@MAC
Interface Réseau
Interface Réseau
Interface Réseau
R1
R2
Hôte A
Routeur
Hôte B

Addressage : port (RFC 6335)

- Trois catégories :
  1. **system** port aka *well-known* :
     - attribués par l'IANA
     - de 0 à 1023
     - utilisable par des utilisateurs privilégiés (root)
  2. **user** port aka *registred* :
     - listés par l'IANA
     - de 1024 à 49 151
  3. **dynamic/private** ports :
     - alloués dynamiquement
     - de 49 152 à 65 535

Listés initialement dans la RFC 1700, puis dans une base en ligne d'après la RFC 3232, la dernière version consultable ici RFC 6335[1]

[1] `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml(.txt)`

Ports standards

cat /etc/services

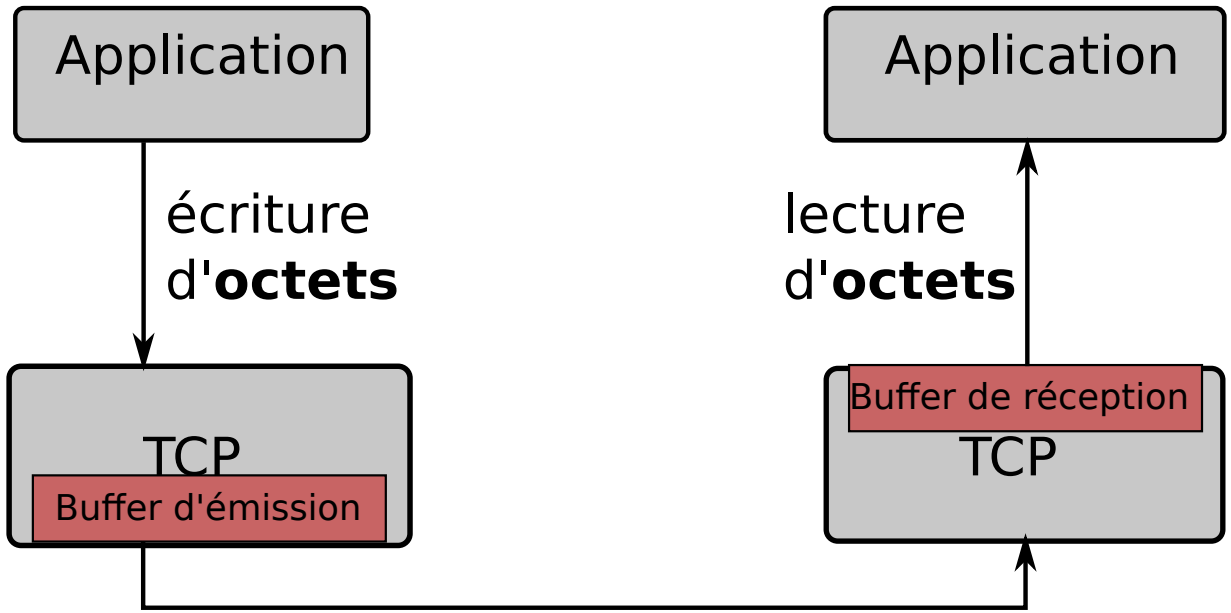
7	echo	
13	daytime	Daytime (RFC 867)
20	ftp-data	File Transfer [Default Data]
21	ftp	File Transfer [Control]
25	smtp	Simple Mail transfert Protocol (envoi de mail)
53	domain	Domain Name Service (résolution de noms)
80	http	www
110	pop3	Post Office Protocol v3 (réception de mail)

Transmission Control Protocol (RFC793)

TCP fournit un service de transfert de données fiable :

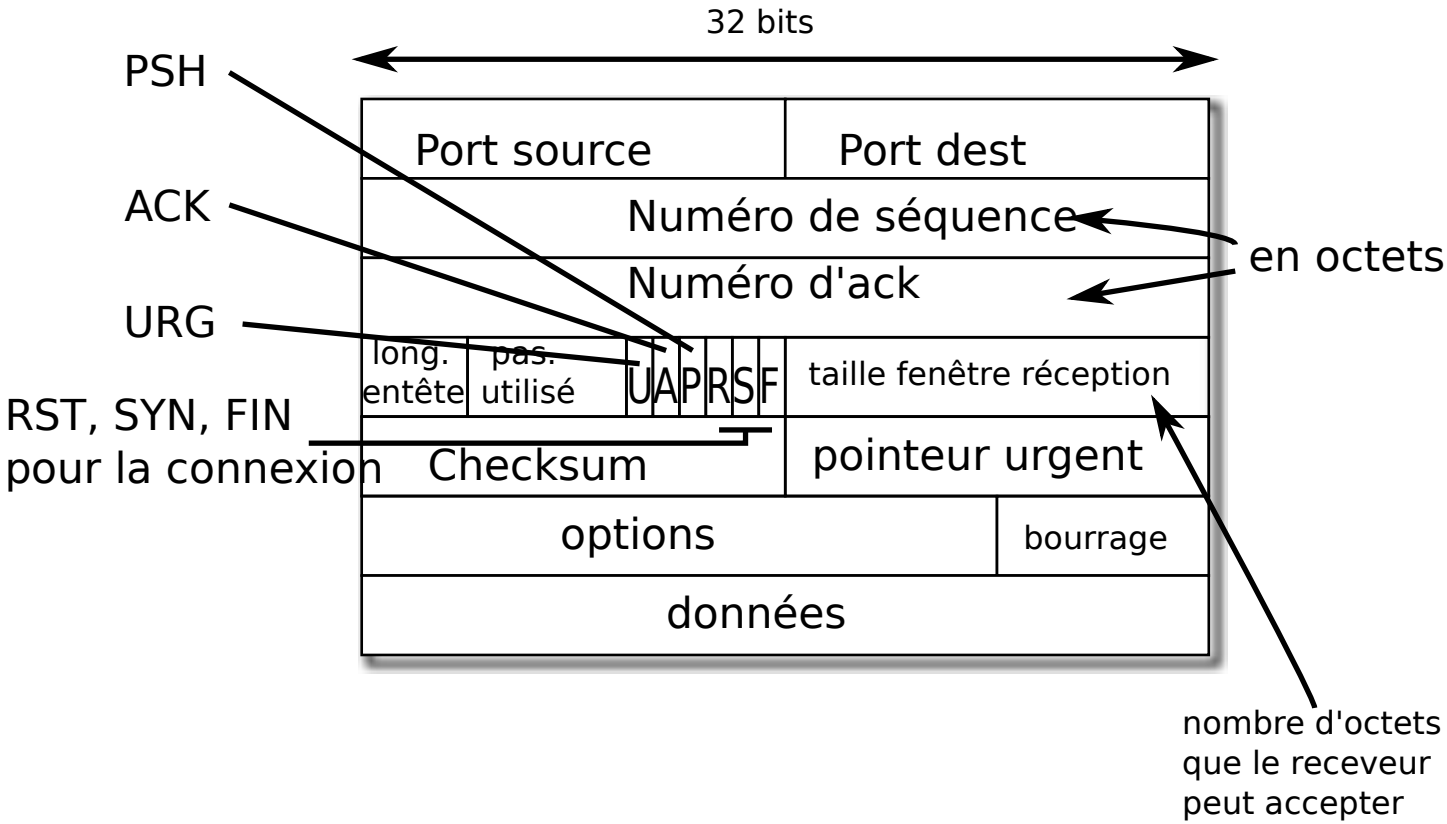
- orienté connexion (connexion / transfert de données / déconnexion)
- communication point à point (*sockets*)
- contrôle des erreurs + acquittements + retransmission
- communication full-duplex
- fournit aux applications un flot d'octets :
  - non structuré
  - sans perte
  - sans duplication
  - dans le bon ordre
  - bufferisé pour améliorer la performance
- contrôle de flux : problème de bout en bout, entre les deux extrémités
- contrôle de congestion : problème de saturation du réseaux
- usage : applications critiques

Flot de données



- Pas de transmission individuelle d'octet
- émission : bufferisation des octets, regroupés en segments envoyés
- réception : bufferisation des segments, lecture par l'application d'octets.

Segment TCP


32 bits
PSH
ACK
URG
RST, SYN, FIN
pour la connexion
Port source
Port dest
Numéro de séquence
Numéro d'ack
long. entête
pas. utilisé
U A P R S F
taille fenêtre réception
Checksum
pointeur urgent
options
bourrage
données
en octets
nombre d'octets
que le receveur
peut accepter

Segment TCP

- Port source (16 bits)/Port destination (16 bits)
- Numéro de séquence (32 bits) : numéro de séquence du premier octet de données envoyé
- Accusé de réception (32 bits) : numéro de séquence du prochain octet à acquitter
- Entête (4 bits) : longueur de l'entête
- Flags (6 bits) : SYN début de communication, FIN fin d'envoi de l'émetteur, RST réinitialisation de la connexion, URG message urgent, ACK acquittement, PSH traitement forcé sans mise en mémoire tampon,
- Fenêtre (16 bits) : utilisé pour le contrôle de flux, nombre d'octet que l'émetteur peut envoyer sans acquittement ou nombre d'octet que l'on peut recevoir (buffer de réception)
- Checksum (16 bits)
- Pointeur de données urgentes (16 bits) : indique l'offset du caractère urgent dans les données si le flag URG est positionné

Segment TCP

- Options :
  - MSS : Maximum segment size (souvent 1460 = 1500 - 40)
  - SACK : ré-émission sélective ; évite la retransmission globale après un segment erroné
  - Window scale : facteur d'échelle pour la taille de fenêtre utilisée (décalage à gauche de la valeur de fenêtre)
  - Time stamp : horodatage pour estimer le temps d'un aller-retour

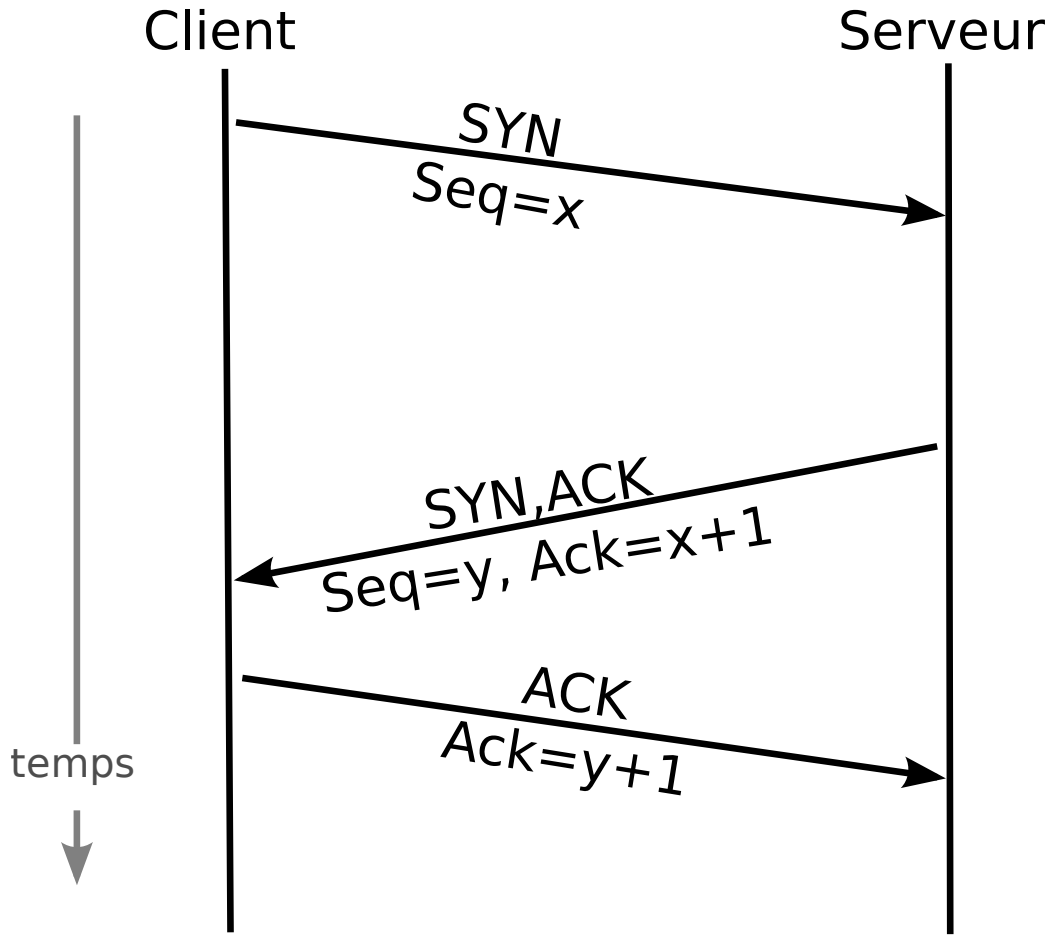
Numéro de séquence et numéro d'acquittement

Une station A

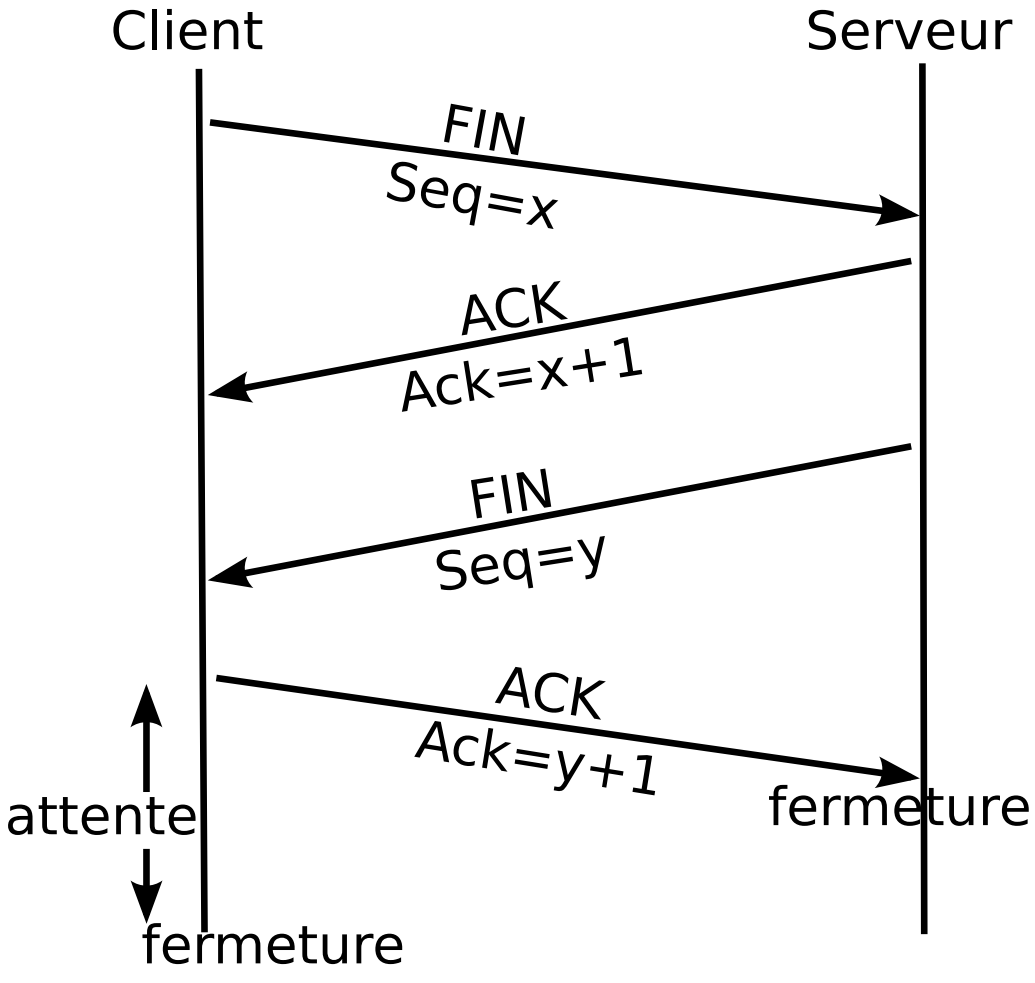
- génère aléatoirement un numéro de séquence à l'initialisation de la session TCP
- accuse réception d'un segment en envoyant le numéro de sequence correspondant au prochain segment attendu
  Celui s'obtient en ajoutant au numéro de séquence reçu le nombre d'octets utiles (de payload) reçus, ou bien 1 si le flag SYN, RST ou FIN est présent

Une station B contactée par A applique le même algo mais avec un SEQ initiale totalement dissocié de A

Ouverture connexion TCP



Fermeture connexion TCP



Transfert fiable

- corruption : données et entête sont protégées par le checksum
- non respect de l'ordre et duplication $\rightarrow$ numéro de séquence
- pertes $\rightarrow$ temporisateur (émetteur) + acquittements $\rightarrow$ retransmissions

Piggybacking

- idée envoyer l'acquittement avec les données
- trois règles :
  1. émission données + acquittement $\rightarrow$ fait dans un (seul) segment
  2. émission acquitement $\rightarrow$ segment spécial envoyé
  3. émission données $\rightarrow$ envoie données + duplication du dernier acquittement

Acquittements et TCP

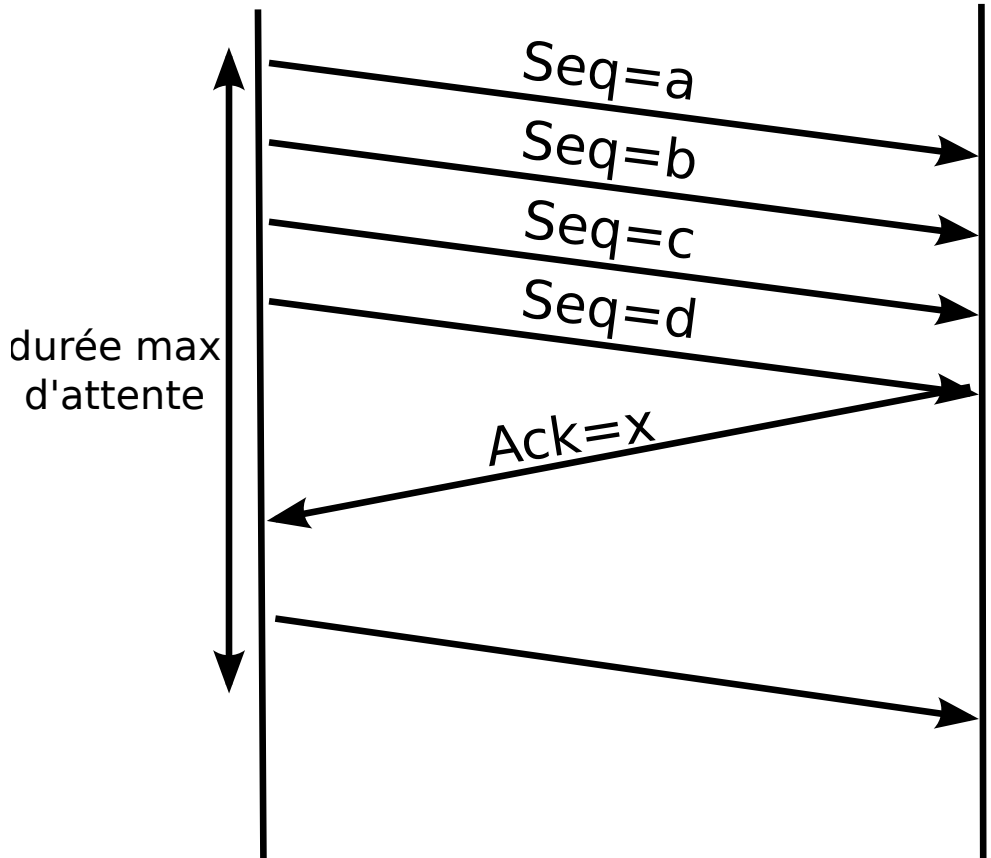
- TCP acquitte les octet reçus et non les segments
- la valeur du temporisateur (Timer), le temps de propagation aller-retour (*Round-Trip delay Time* ou RTT) est calculée dynamiquement $\Rightarrow$ adaptation aux différents réseaux.
- la politique de réémission est le GO-BACK-N, reprise depuis le dernier **octet** spécifié.
- le nombre de segments émis avant acquittement repose sur l'utilisation du mécanisme de fenêtre glissante.

Temporisation

- RTT *Round-Trip delay Time* :
  temps aller-retour segment
  obtenu par : horodatage à émission, recopié dans l'acquittement $\rightarrow$ RTT échantillon
- RTT moyen :
  moy(RTT) = (1-α) $\times$ moy(RTT) + α $\times$ RTT échantillon
  avec valeur typique de α : 0.125
- variance :
  var(RTT) = (1-β) $\times$ var(RTT) + β $\times$ |RTT échantillon-moy(RTT)|
  valeur typique de β : 0.250
- délai d'attente
  Délai = moy(RTT) + sécurité
  valeur typique de sécurité : 4 $\times$ var(RTT)

Fenêtre glissante

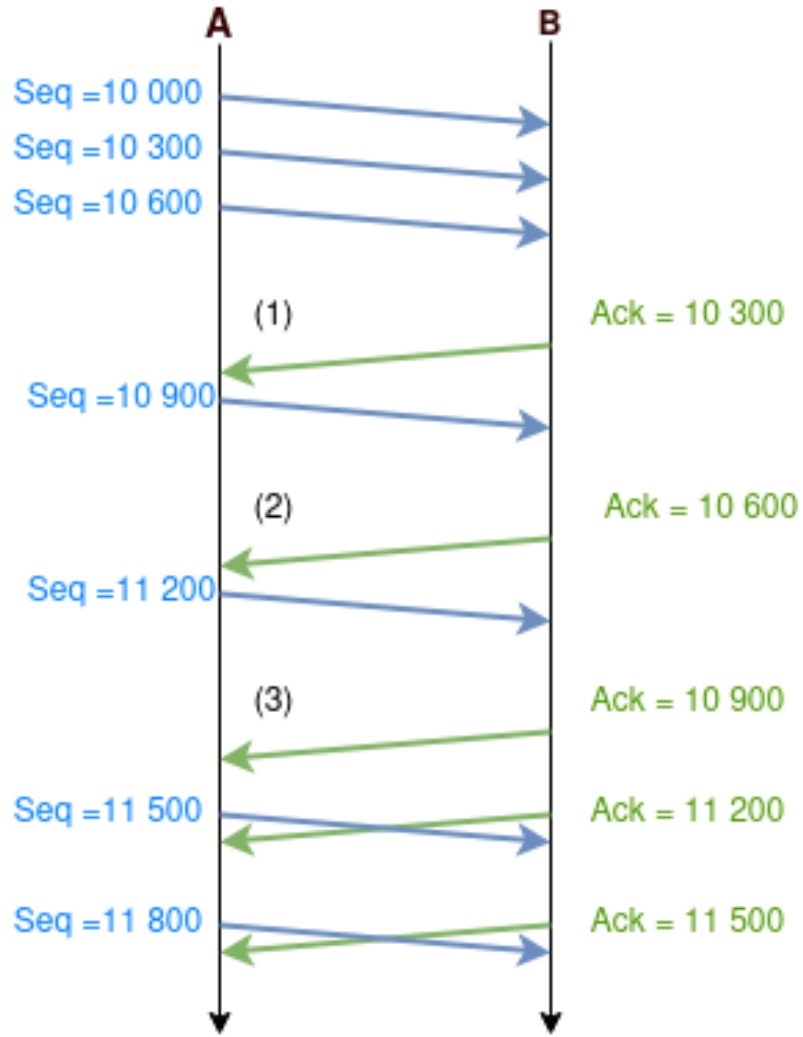
- temps de transit dans le réseau long $\rightarrow$ Stop-and-wait pénalisant (i.e. d'attendre un acquittement pour chaque segment)
- émission de plusieurs segments sans attente d'acquittement mais jusqu'à atteindre la capacité que le destinataire a communiquée



Fenêtre glissante

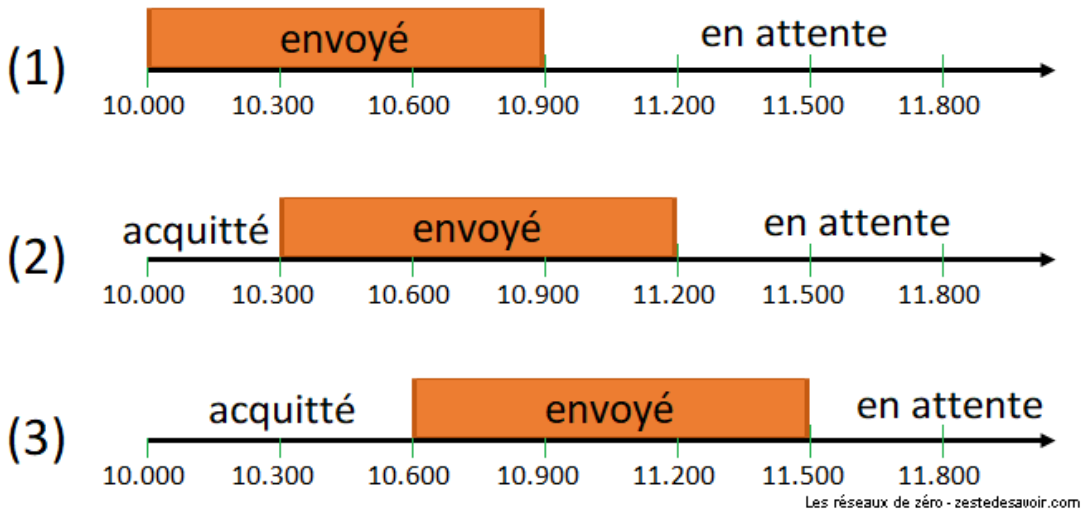
Ci-dessous la connexion TCP est établie, B a informé à A que sa fenêtre (i.e. mémoire tampon) est de 900. On suppose une MSS des segments de 300.

- Après 3 envois, A atteint la fenêtre de B et se met à attendre un acquittement
- B n'acquitte que 300, A envoie en conséquence
- la cadence d'envoie est ralentie



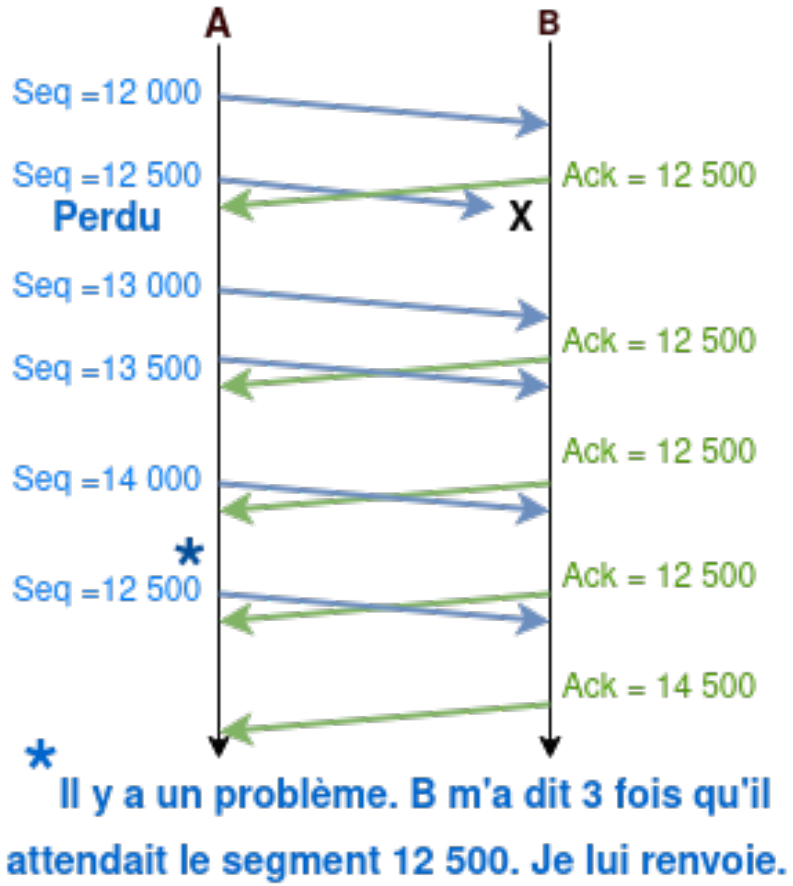
Fenêtre glissante

On parle de **fenêtre glissante** (sliding window) pour désigner la zone qui se déplace le long du flux de données à envoyer



Contrôle de congestion

- l'objectif est de trouver la bande bassante disponible i.e. le nombre de segments que l'on peut envoyer sans attendre d'acquitement (on parle de fenêtre de congestion)



Modes slow start et congestion avoidance

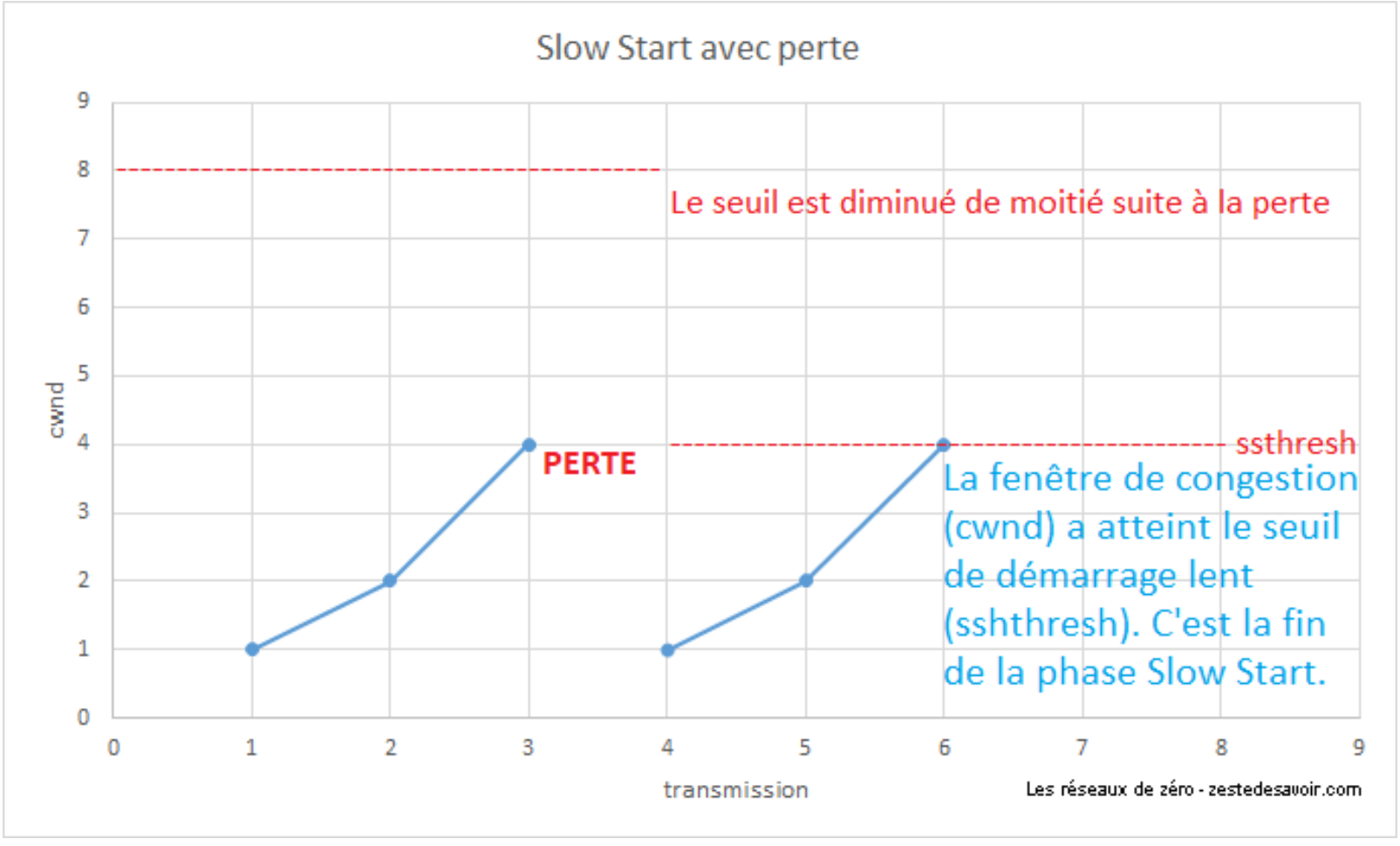
Comment définir la taille de la fenêtre de congestion ?

Principle

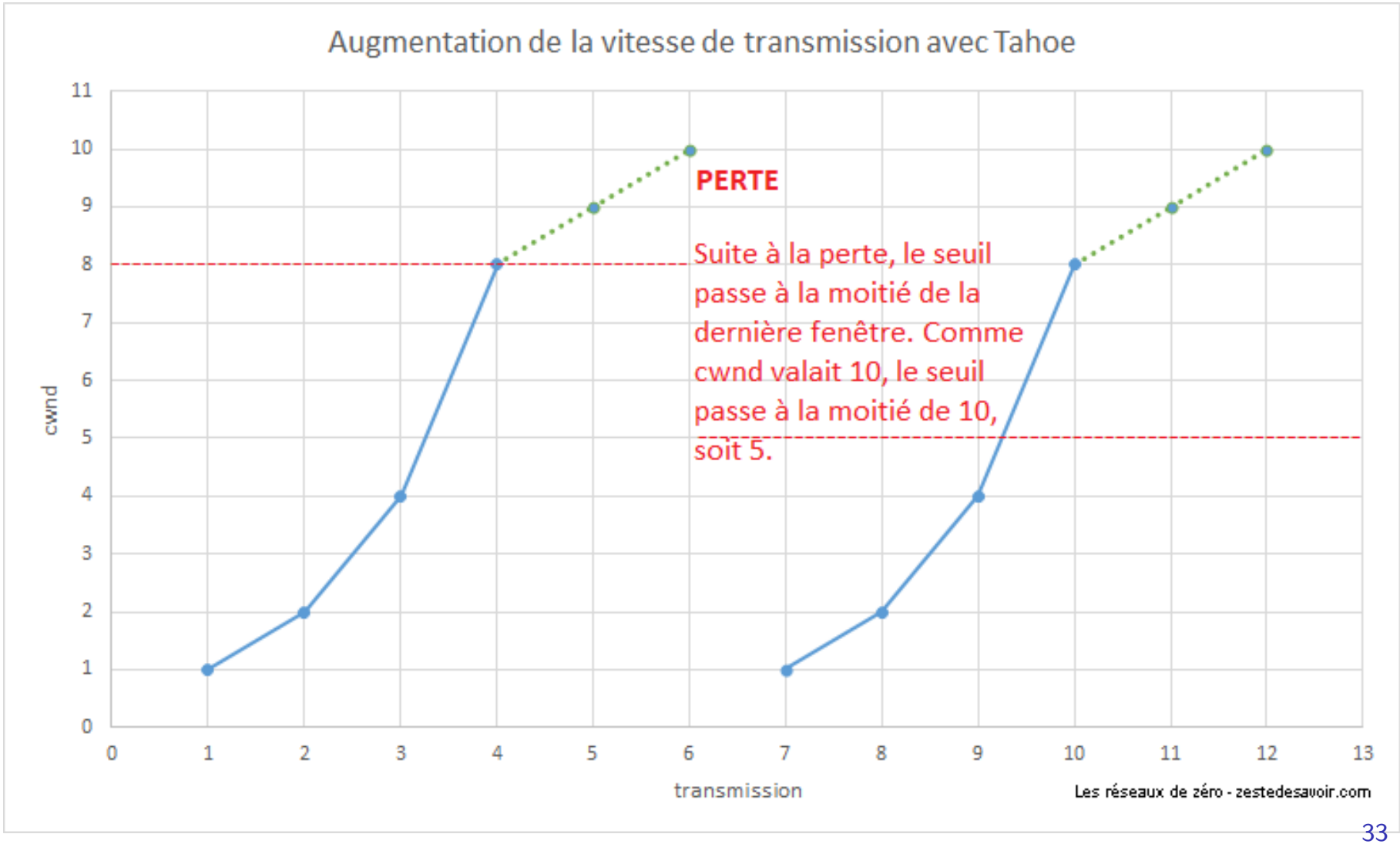
- débute en mode **slow start** i.e. augmentation exponentielle de la **taille de la fenêtre de congestion** (*congestion window size aka cwnd*)
- jusqu'à atteindre un **seuil de démarrage lent** (*slow start threshold aka ssthresh*) à partir duquel passe en mode **congestion avoidance** i.e. l'augmentation est linéaire (i.e. plus prudente)

- initiée généralement à 1 MSS, augmentation par un facteur 2 à chaque acquittement puis par addition de 1 après le seuil
- si problème constaté (perte d'un segment) alors ssthresh $=$ cwnd$/2$; cwnd $= 1$; mode $=$ slow start

Algorithme tahoe - slow-start



Algorithme tahoe - congestion-avoidance



fast retransmission et mode fast recovery

fast retransmit

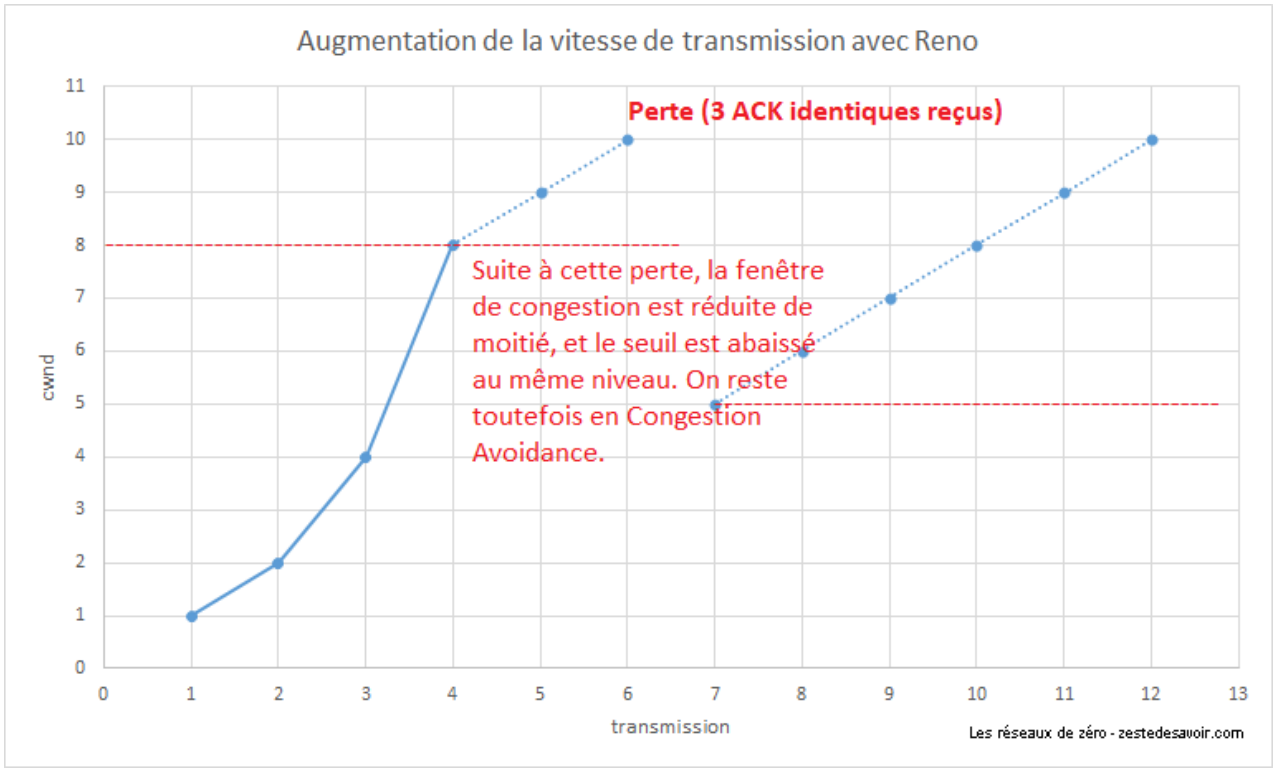
- Un émetteur utilise un timer pour détecter la perte de segments.
- Lorsque celui-ci reçoit des acquittements dupliqués, il déduit la perte d'un segment et le renvoie sans attendre la fin du timer

Si 3 ACKs dupliqués reçus alors

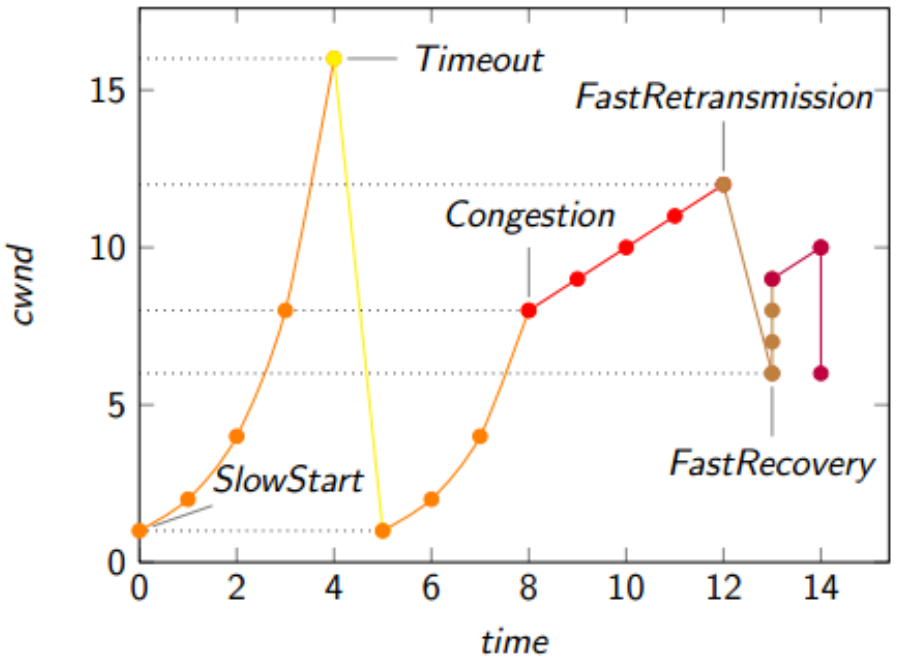
Tahoe fast retransmit et ssthresh = cwnd/2 ; cwnd = 1 ; mode = slow start

Reno fast retransmit et ssthresh = cwnd = cwnd/2 ; mode = **fast recovery**

reno - fast retransmission



Contrôle de congestion TCP



Idée : TCP ne doit pas empirer la situation du réseau
basé sur les algorithmes suivants :

- Slow Start (croissance exponentielle $\rightarrow$ perte)
- Congestion avoidance (croissance additive, décroissance multiplicative)
- Fast Retransmission/Fast recovery

User Datagram Protocol (RFC 768)

UDP fournit un service non fiable :

- possibilité de perte FOO BAR $\rightarrow$ BAR
- possibilité de duplication FOO BAR $\rightarrow$ FOO FOO BAR
- possibilité de désordre FOO BAR $\rightarrow$ BAR FOO
- mais préservation des limites (orienté message) FOO BAR $\nrightarrow$ FO OBAR
- Si besoin, c'est à l'application de gèrer ordre, unicité et non perte

UDP

- simple
- mode non connecté
- orienté message
- transport non fiable
- cout réduit par rapport à TCP (8 octets / 20 octets)
- envoi direct des informations
- multicast possible
- pas de garantie de séquencement
- contrôle des erreurs
- usage : applications multimédias, DNS

Datagramme UDP