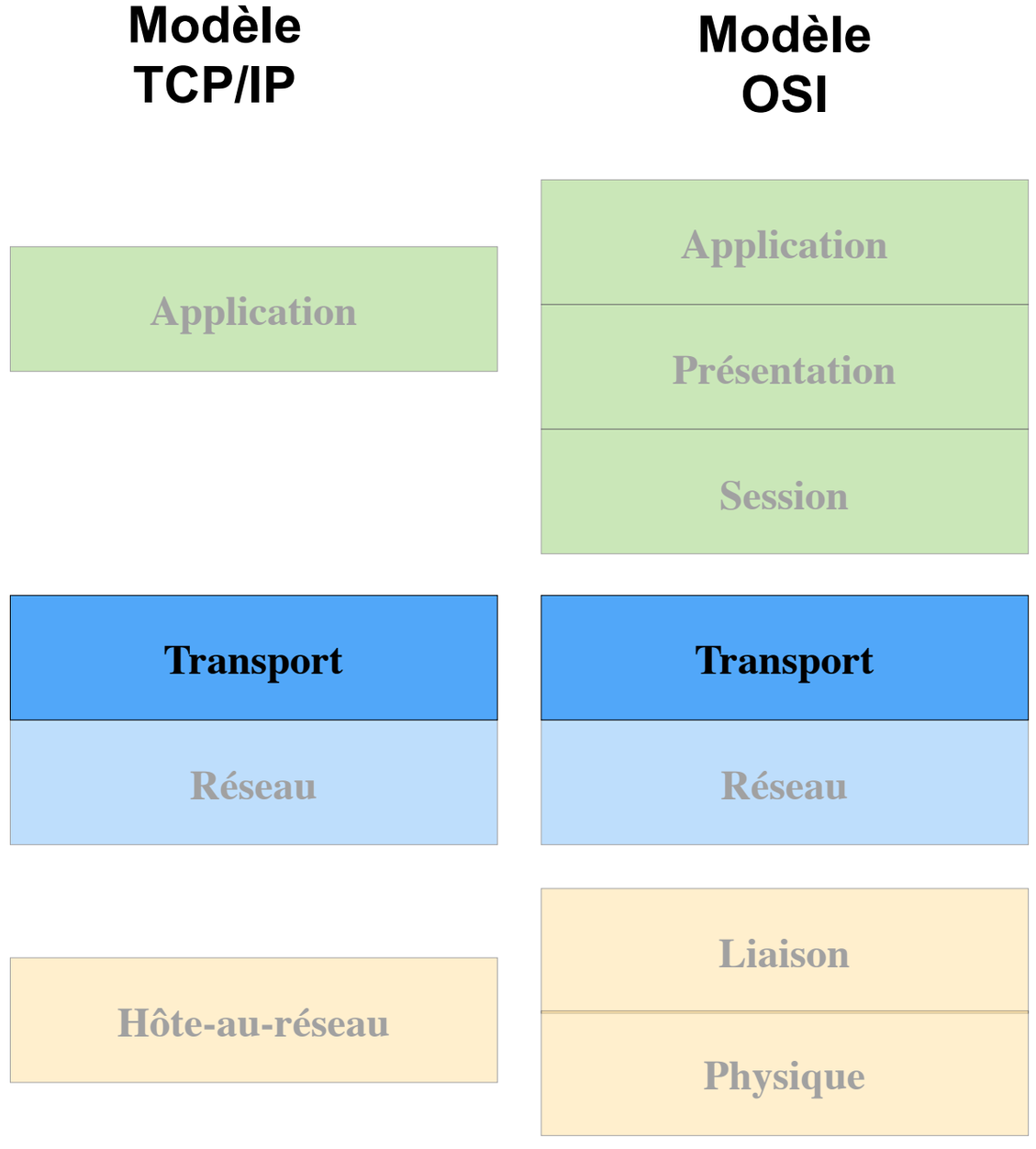
Institut Mines-Télécom

La couche Transport

Claude Chaudet

Introduction

- **La couche transport gère les propriétés de la communication de bout-en bout**
  - Multiplexage entre applications
  - Corrélation des paquets entre eux pour former un flux de données
  - Fiabilité de bout-en-bout
  - Contrôle du débit des flux pour éviter les congestions



Mutliplexage : numéros de port

Pourquoi des numéros de ports ?

- **Une machine est identifiée par une adresse IP**

- **Comment distinguer les applications sur la même machine ?**
  - Numéro entre 0 et 65535

  - Spécifié dans l'en-tête de niveau transport
    - ⟹ Totalement invisible dans le réseau

Comment se passe une connexion ?

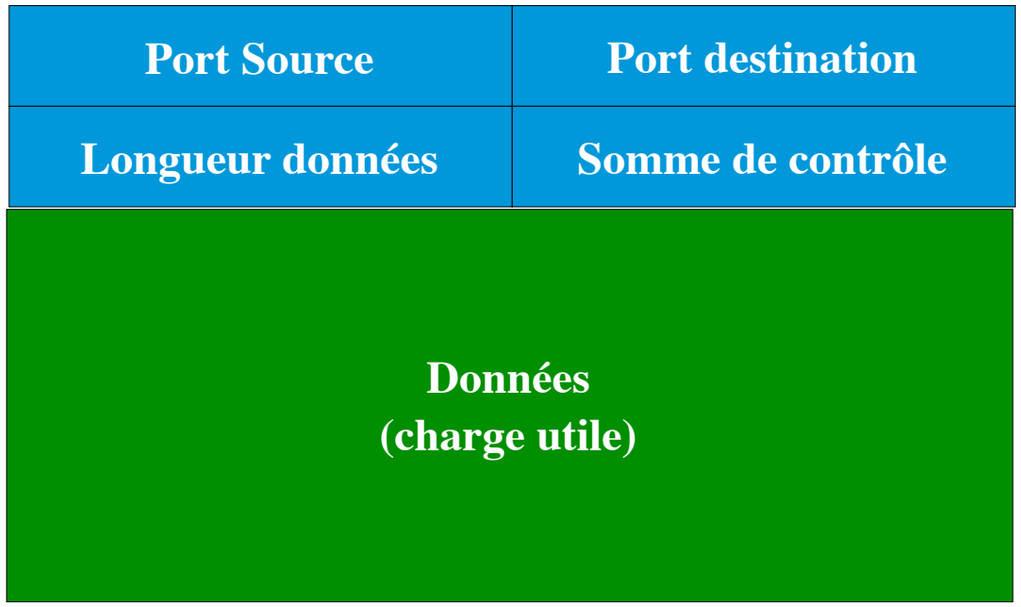
- **Le client envoie une requête au serveur sur un numéro de port précis**
  - Connu à l'avance (explicitement ou par type de service)
  - Aucune obligation d'utiliser un numéro donné pour un service donné, mais des usages

- **Il spécifie dans sa requête un numéro de port sur lequel lui faire parvenir les réponses**
  - Souvent tiré au hasard dans un intervalle

Intervalle	Type	Usage	Exemples
1 — 1023	*Well-known*	Services et protocoles standards	Web (HTTP) : 80 e-mail (SMTP) : 25 session (SSH) : 22 DNS : 53
1024 — 49151	Enregistrés	Applications déclarées à l'IANA	Bittorrent : 6881 Proxy HTTP : 8080 Quake server : 26000
49152 — 65535	Aucune réservation possible	Réponses, applications spécifiques	

En-tête UDP

- **UDP ajoute aux données de l'application un en-tête, minimaliste :**
  - Ports source (16 bits)
  - Port destination (16 bits)
  - Longueur des données (16 bits)



  - Somme de contrôle (16 bits ; RFC 768)
    - Permet de filtrer les paquets erronés au niveau du récepteur
    - Calculé sur l'en-tête + les données
    - Nécessaire car on ne présuppose pas de vérification d'intégrité de la part des couches inférieures

Vocabulaire(s) de la couche transport

- **UDP fournit un service minimal : numéros de port et contrôle d'intégrité**
  - Toute donnée en erreur est perdue pour l'application
  - Les données sont envoyées au rythme défini par l'application source

- **Ce mode de communication s'appelle mode datagramme**
  - L'ensemble formé par les données de l'application et l'en-tête UDP se nomme un datagramme
  - Plutôt destiné aux applications multimédia (tolérance aux pertes)

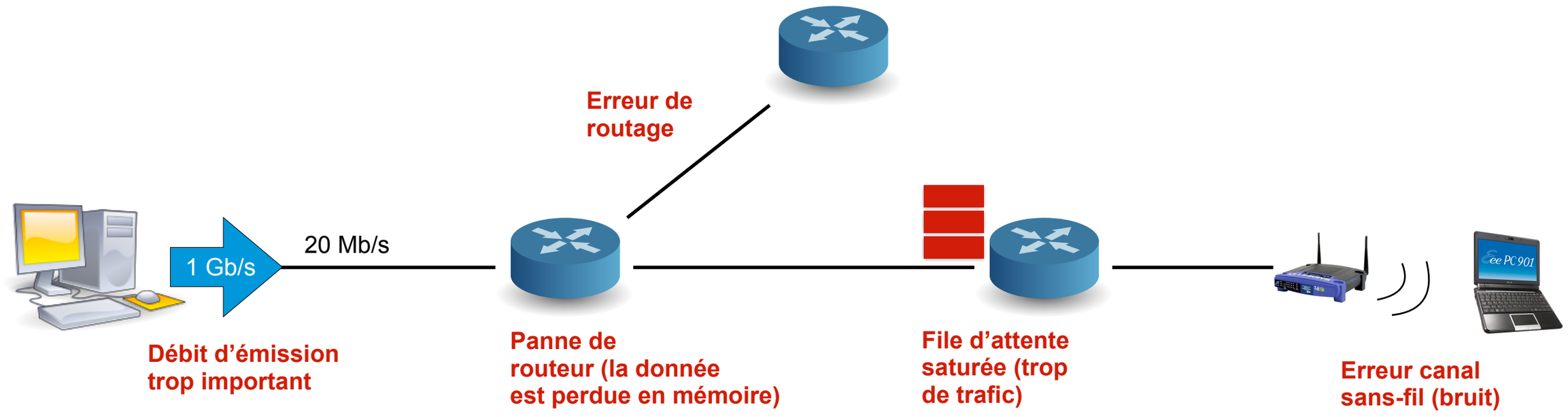
- **TCP ajoute à UDP des fonctions avancées pour les transmissions de données :**
  - Fiabilité (retransmissions en cas d'erreur)
  - Contrôle de flux (débit d'émission)

- **On parle alors de mode connecté**
  - L'ensemble formé par les données de l'application et l'en-tête TCP se nomme un segment

La fiabilité dans TCP

Pourquoi une gestion de la fiabilité ?

- **Des pertes peuvent survenir dans le réseau**
  - Beaucoup de causes possibles

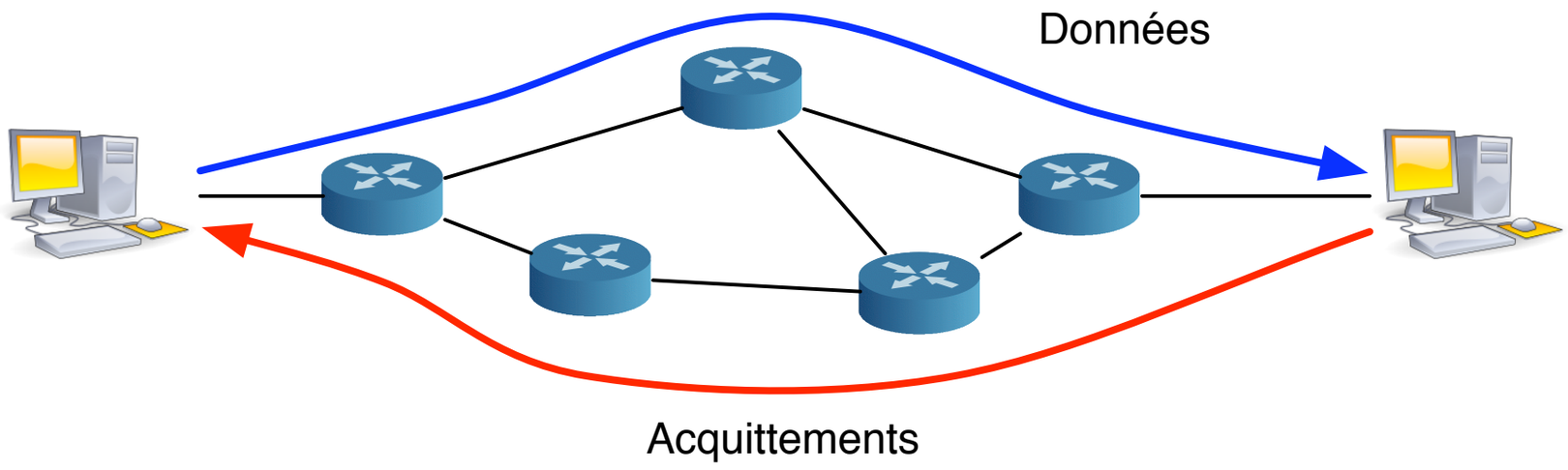


- **Beaucoup d'applications ne peuvent tolérer aucune perte**
  - TCP fournit une gestion des pertes pour que les applications n'aient pas à le faire elles-mêmes

Comment s'assurer de la bonne réception d'un segment ?

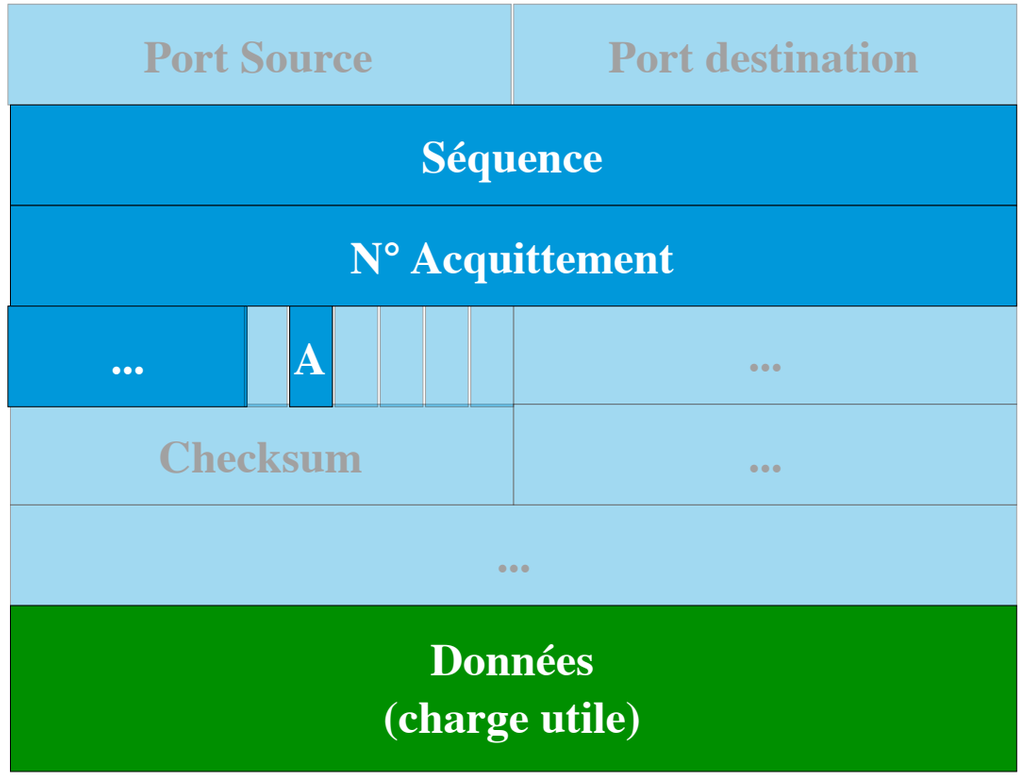
- **Seule la destination pourra être sûre que les informations sont bien arrivées**
  - Acquittement explicite de chaque segment
  - Numérotation des segments pour détecter les pertes
    - En pratique : on numérote les octets, pas les messages

- **Envoi d'un acquittement de la destination à la source**
  - Transmis par le même réseau, en utilisant les mêmes mécanismes
  - Ils peuvent suivre des chemins différents, être perdus, retardés, etc.



Acquittements : en pratique

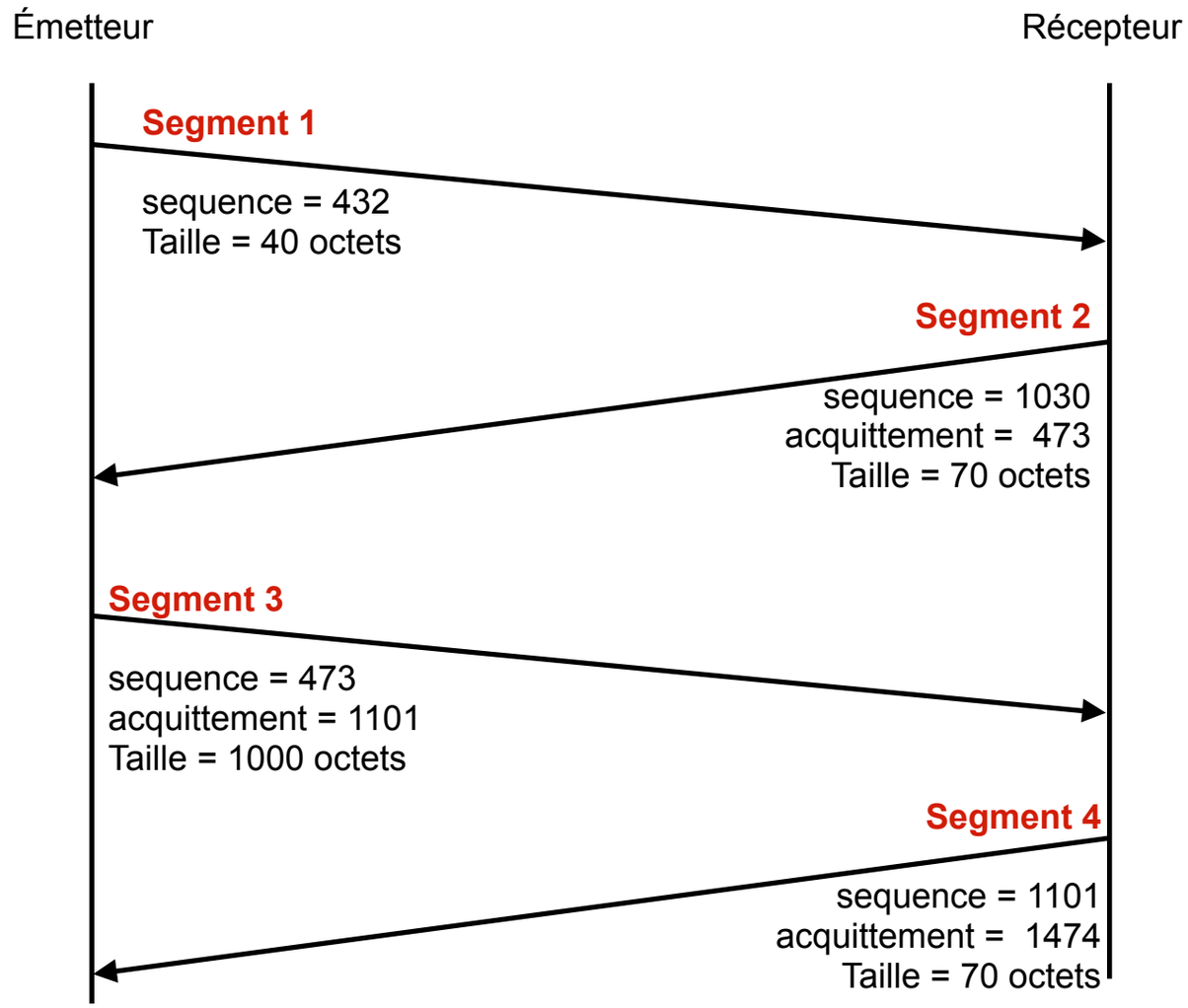
- **Numéro de séquence pour identifier les portions de message**
  - Numéro sur 32 bits
  - Unité : index du premier octet du segment
  - Valeur initiale : aléatoire pour gérer certains cas d'erreurs ambigus



- **Ce numéro est reporté dans les acquittements qui sont des segments sans forcément de charge utile associée**
  - Le numéro d'acquittement est l'index du prochain octet que l'on s'attend à recevoir
  - Permet de combiner transmission effective et acquittement dans le cas de communications bidirectionnelles
  - Les numéros de séquence ne sont pas identiques dans les deux directions
  - Un bit indique si l'on acquitte un segment ou non (A)

Acquittements : exemple

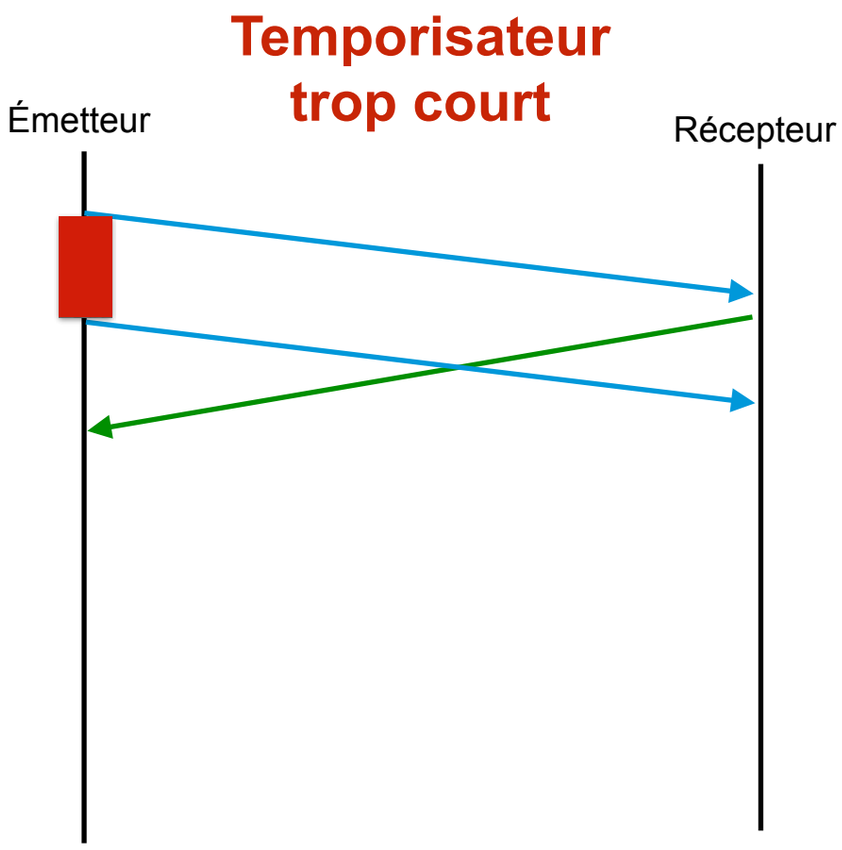
- **Exemple d'une communication bidirectionnelle**
  - L'émetteur commence avec un numéro de séquence égal à 432
  - Le récepteur démarre à 1030
  - Si on dépasse 232, on repart à 0

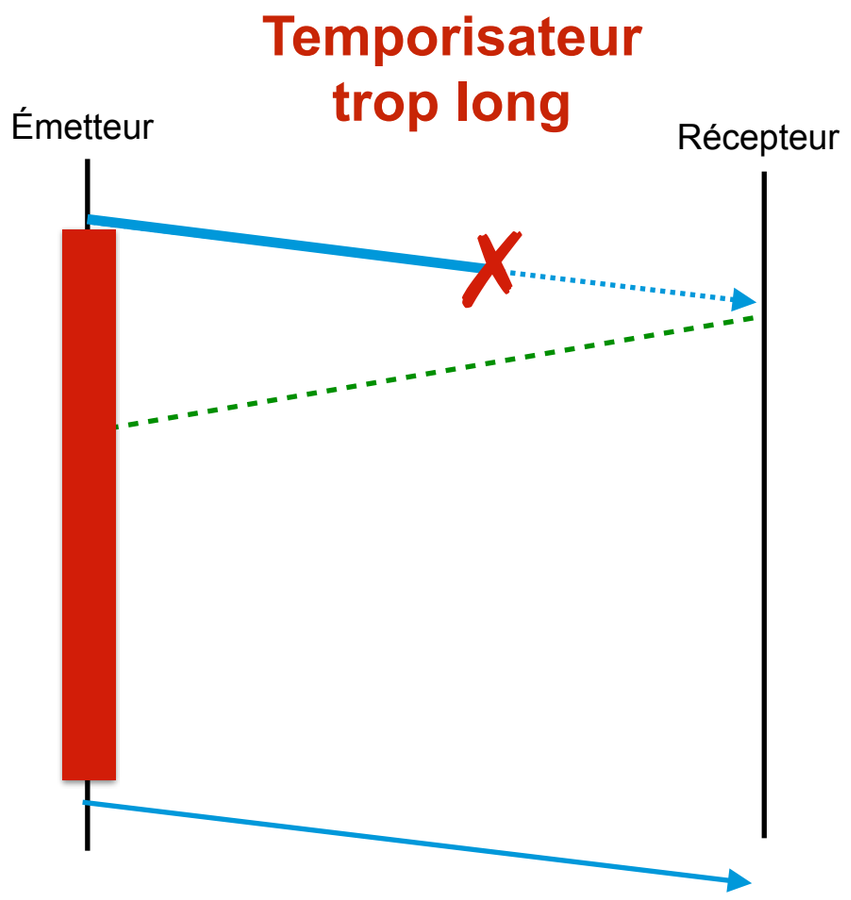


Détection des pertes

- **L'émetteur ne recevant pas d'acquittement conclura à une perte**
  - Retransmission du segment incriminé
  - Il ne peut pas distinguer la perte du segment de celle de l'acquittement

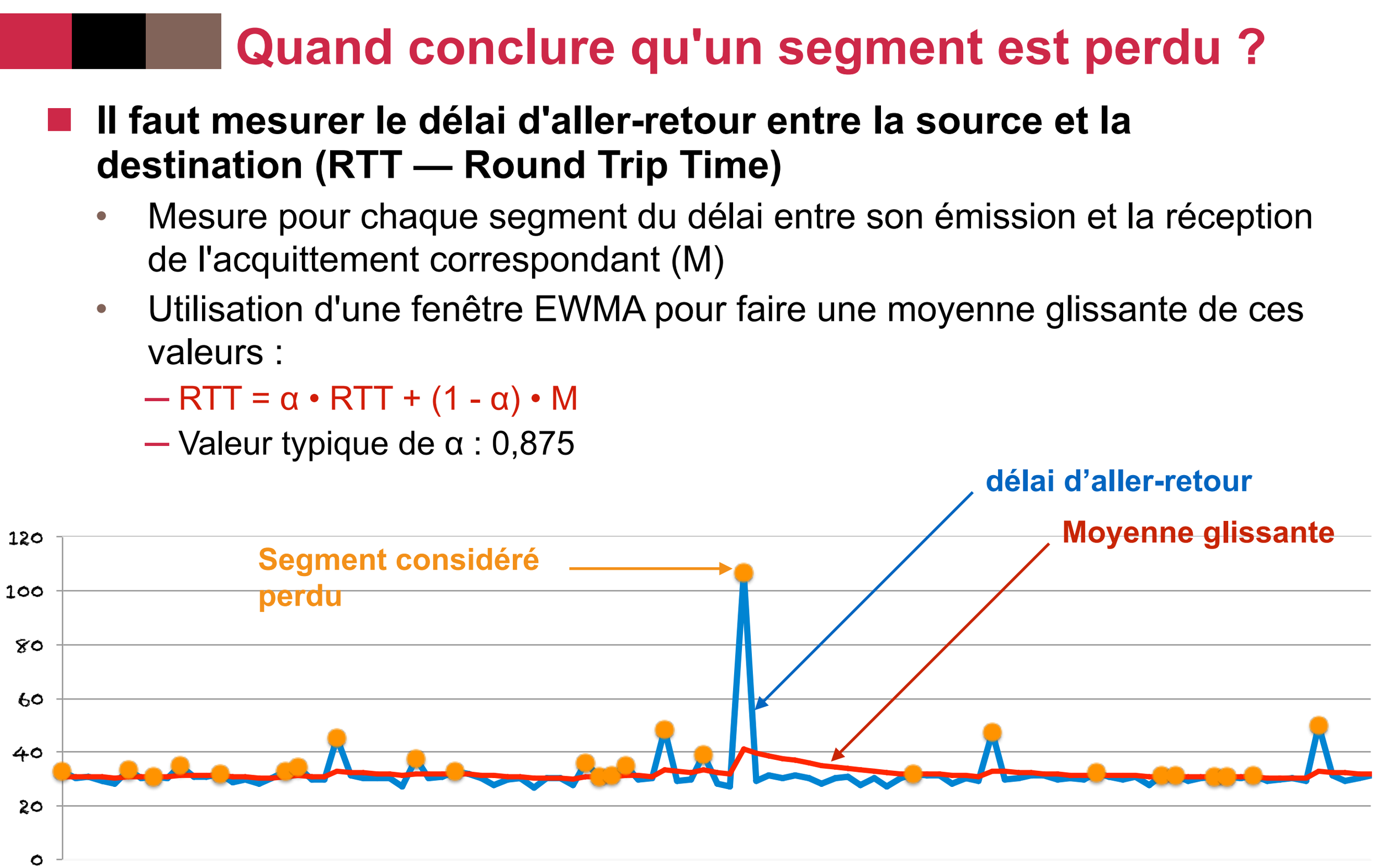
- **Le temporisateur définissant le moment d'une retransmission doit être bien réglé**





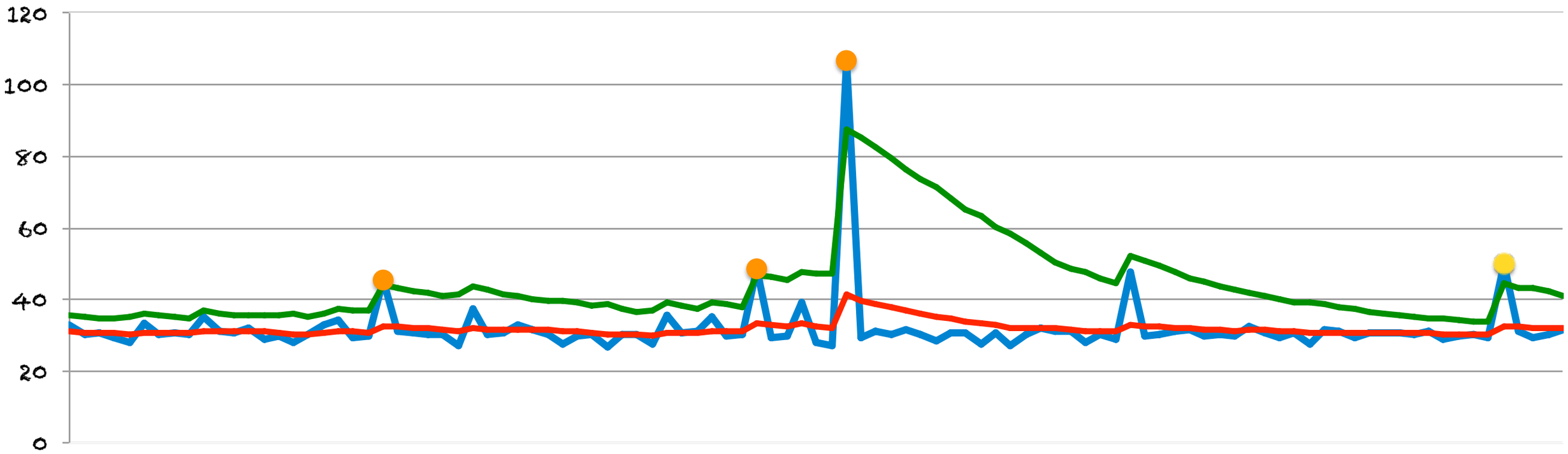
Quand conclure qu'un segment est perdu ?

- **Il faut mesurer le délai d'aller-retour entre la source et la destination (RTT — Round Trip Time)**
  - Mesure pour chaque segment du délai entre son émission et la réception de l'acquittement correspondant (M)
  - Utilisation d'une fenêtre EWMA pour faire une moyenne glissante de ces valeurs :
    - RTT = α • RTT + (1 - α) • M
    - Valeur typique de α : 0,875



Ajout de la dispersion (écart-type)

- **Régler le temporisateur sur la moyenne conduit à retransmettre trop de segments**
  - On mesure la dispersion entre la valeur espérée et la valeur réelle
    - $D = \alpha \cdot D + (1 - \alpha) \cdot | RTT - M |$
  - Et on ajoute alors, empiriquement, 4 fois cette valeur
    - $Temporisateur = RTT + 4 \cdot D$



TCP : Notion de connexion

Notion de connexion TCP

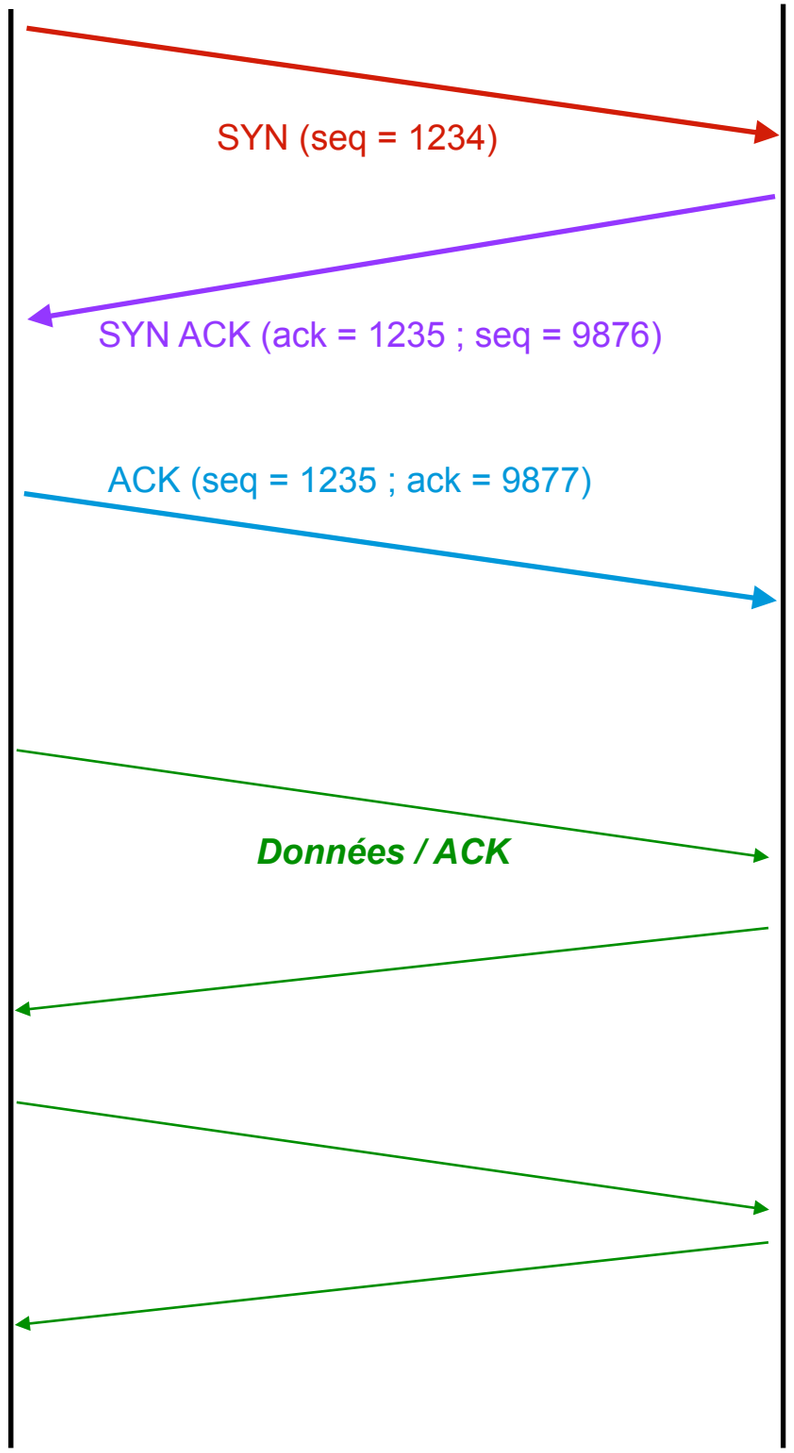
- **Un échange TCP est plus qu'une suite de messages**
  - Ordre des messages préservé
  - Possède un début et une fin

- **Le protocole spécifie donc un échange de messages afin de marquer le début et la fin d'une communication**
  - Initialisation de la connexion : échange de messages pour se mettre d'accord sur les numéros de séquence

  - Terminaison de la connexion : transmission des segments non acquittées avant de finir effectivement la communication

Ouverture d'une connexion

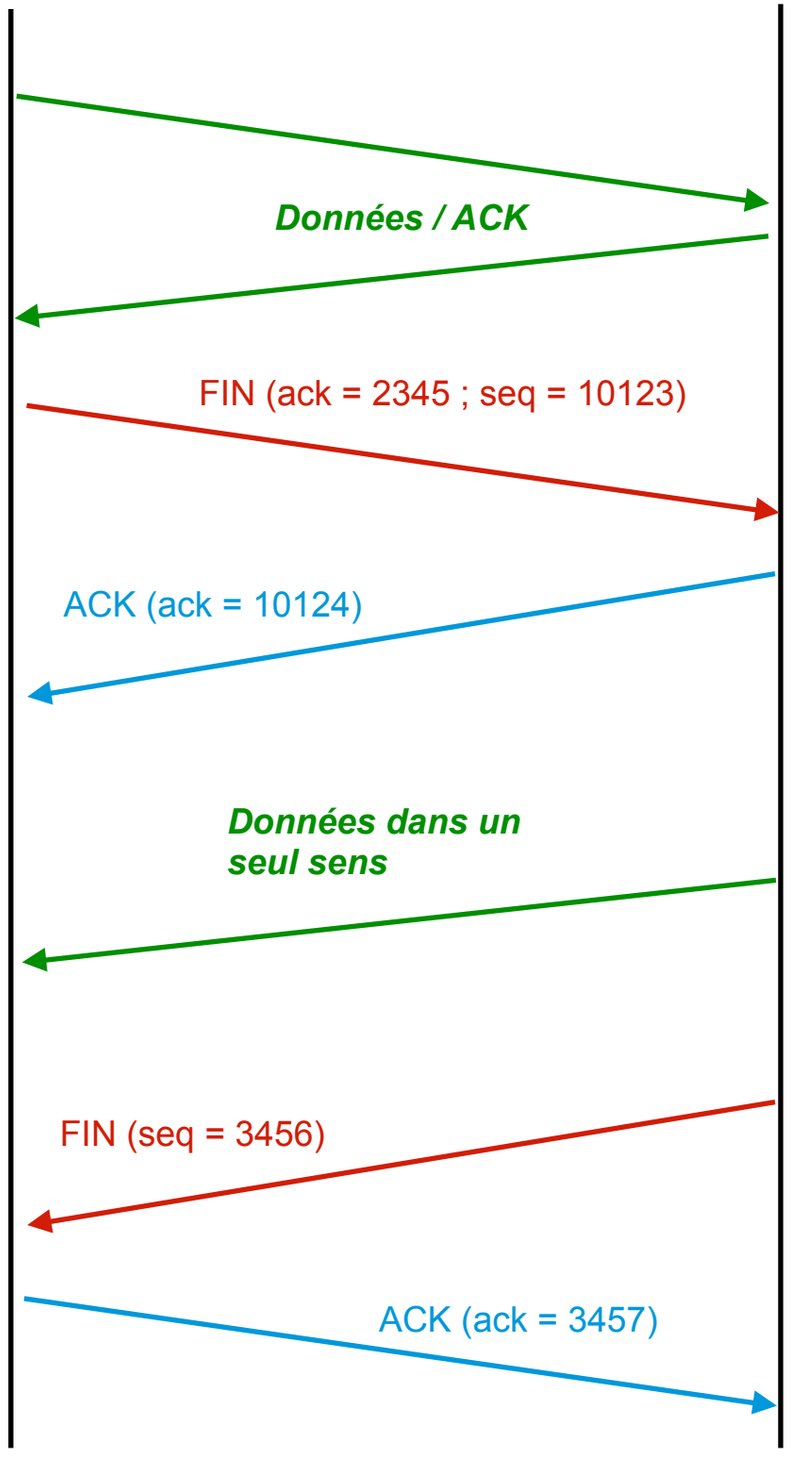
- **L'ouverture d'une connexion s'effectue en 3 messages**
  - SYN : envoi du numéro de séquence initial de l'émetteur
  - SYN ACK : acceptation de la connexion et envoi du numéro de séquence du récepteur
  - ACK : acquittement du la bonne réception du numéro de séquence



Fin d'une connexion

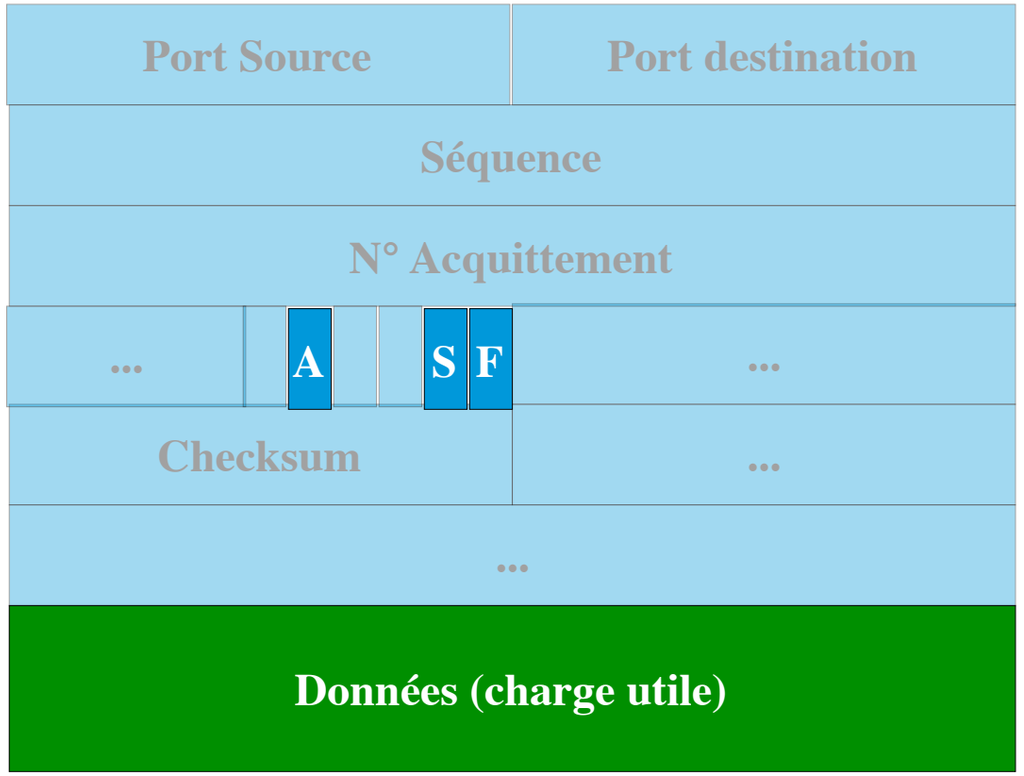
- **La terminaison d'une connexion s'effectue au moyen de deux messages :**
  - FIN : indication que l'on n'a plus de données à transmettre
  - ACK : acquittement de la fin de la communication et de la bonne réception de l'intégralité des données

- **Cet échange a lieu dans chacune des directions, fermant la connexion dans les deux sens.**
  - On pourrait a priori garder la connexion de données ouverte dans un sens unique



En pratique

- **Les messages précédents ne sont pas des messages particuliers mais sont des en-têtes TCP envoyés sans charge utile pour la plupart et dans lesquels on a positionné des drapeaux.**
  - Similaire au mécanisme d'acquittements



- **Drapeaux :**
  - ACK
  - SYN
  - FIN
  - ...

TCP : Contrôle de congestion

Définition et motivations

- **Le contrôle de congestion consiste utiliser le maximum des ressources disponibles, tout en limitant le débit d'émission des flux :**
  - Lorsque le récepteur ne peut traiter / stocker les données à un débit élevé.
  - Lorsque le réseau ne peut acheminer les données au débit d'envoi.

- **Pourquoi ?**
  - L'émission de segments inutiles surcharge le réseau et introduit des dé-séquencements.
  - Les pertes sont souvent dues à de la congestion (surcharge d'un équipement du réseau), donc réduire le débit d'émission peut aider à résoudre le problème

- **Comment ?**
  - L'émetteur doit adapter le débit d'émission en fonction de sa perception de l'état du réseau et du récepteur

Fenêtres

- **Pour effectuer le contrôle de congestion, on dispose de deux fenêtres (exprimées en nombre d'octets):**
  - La fenêtre de congestion, représentant la quantité de données que l'on s'autorise à envoyer avant de recevoir un acquittement
    - Fonction de l'état du réseau et du produit bande passante x délai du chemin de bout-en-bout
  - Une fenêtre de réception, maintenue par le récepteur
    - Indique l'espace libre au niveau du tampon de réception

- **L'émetteur combine ces deux valeurs (min) pour déterminer le bon débit d'émission**
  - Augmentation du débit lorsque les ressources le permettent
  - Réaction rapide à l'apparition de congestion afin de ne pas aggraver le problème
  - Objectif d'équité entre les flux : tout le monde doit participer à la résolution des congestions.

Notification de l'émetteur

- **L'émetteur n'a qu'une seule possibilité pour connaître l'état du réseau et du récepteur : les acquittements de segments**

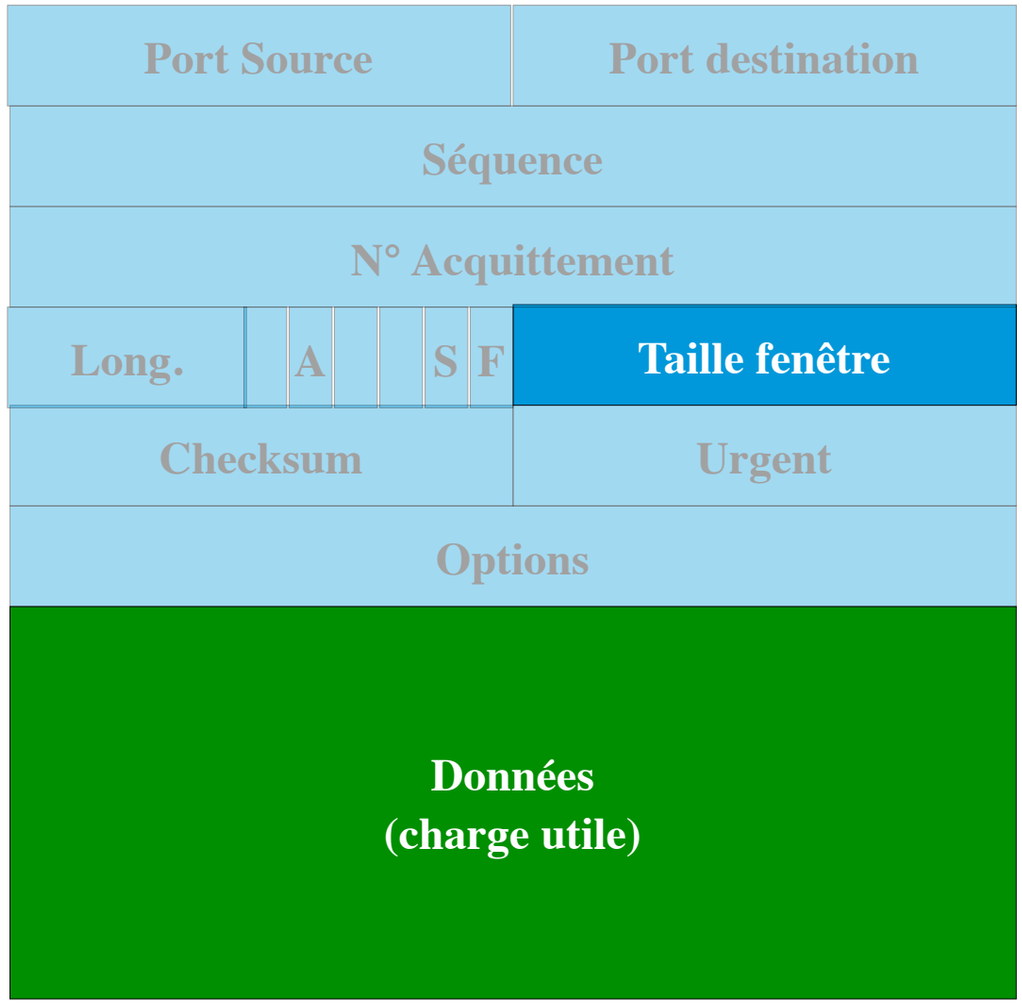
- **L'absence de réception d'un acquittement (dans un délai raisonnable) indique que le segment correspondant a sans doute été perdu**
  - Possibilités alternatives : délai excessif, perte de l'acquittement

- **Afin de distinguer les pertes dues au réseau des pertes dues au récepteur, ce dernier inclut dans les acquittements la valeur de sa fenêtre de réception**
  - Nombre d'octets que le récepteur accepte après ceux acquittés

En-tête TCP complet

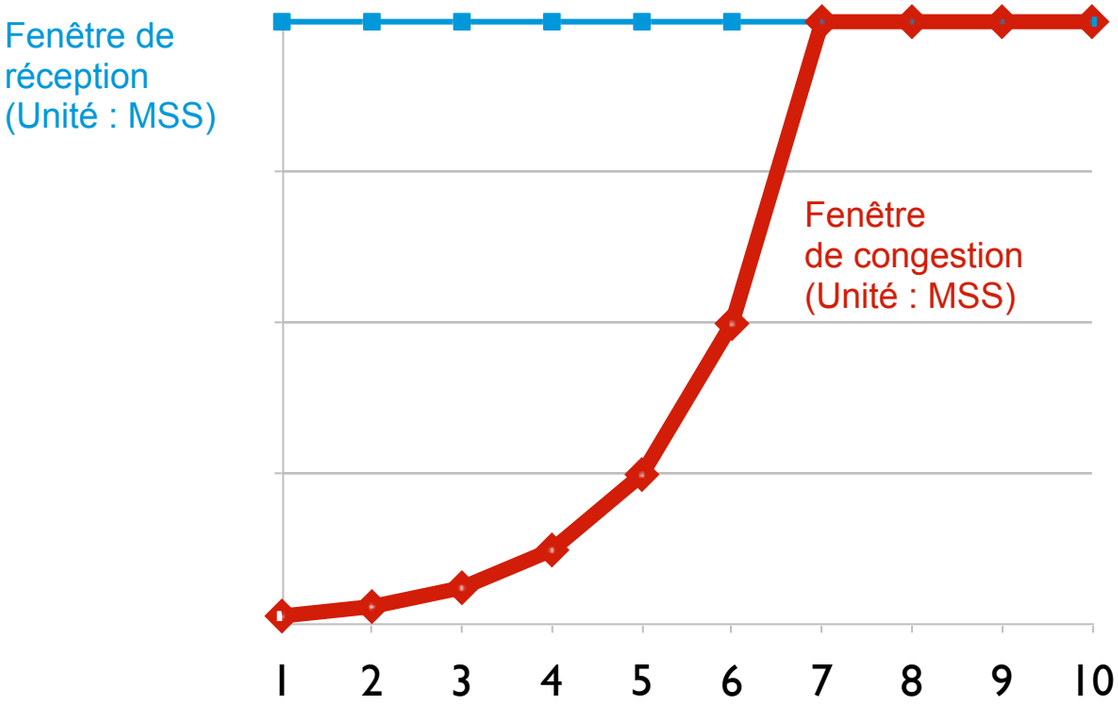
- **On ajoute à l'en-tête TCP (qui constitue les acquittements) un champ taille de fenêtre**

- **Quelques autres champs**
  - Options
  - Longueur de l'en-tête
  - Notion de données urgentes inclues dans le message



Mécanismes : Slow Start

- **Départ précautionneux**
  - S'autorise au départ à envoyer une taille maximale de segment (MSS) et on attend l'acquittement
- **Augmentation exponentielle de la taille de la fenêtre d'anticipation**
  - À la réception de chaque acquittement, on ajoute 1 à la taille de la fenêtre
  - Doublement à chaque RTT
  - On arrête l'augmentation lorsque des acquittements ne sont pas reçus ou lorsque la taille de la fenêtre de réception est atteinte



Note : Le MSS est négocié au moment de la connexion (SYN / SYN ACK) par les deux extrémités.

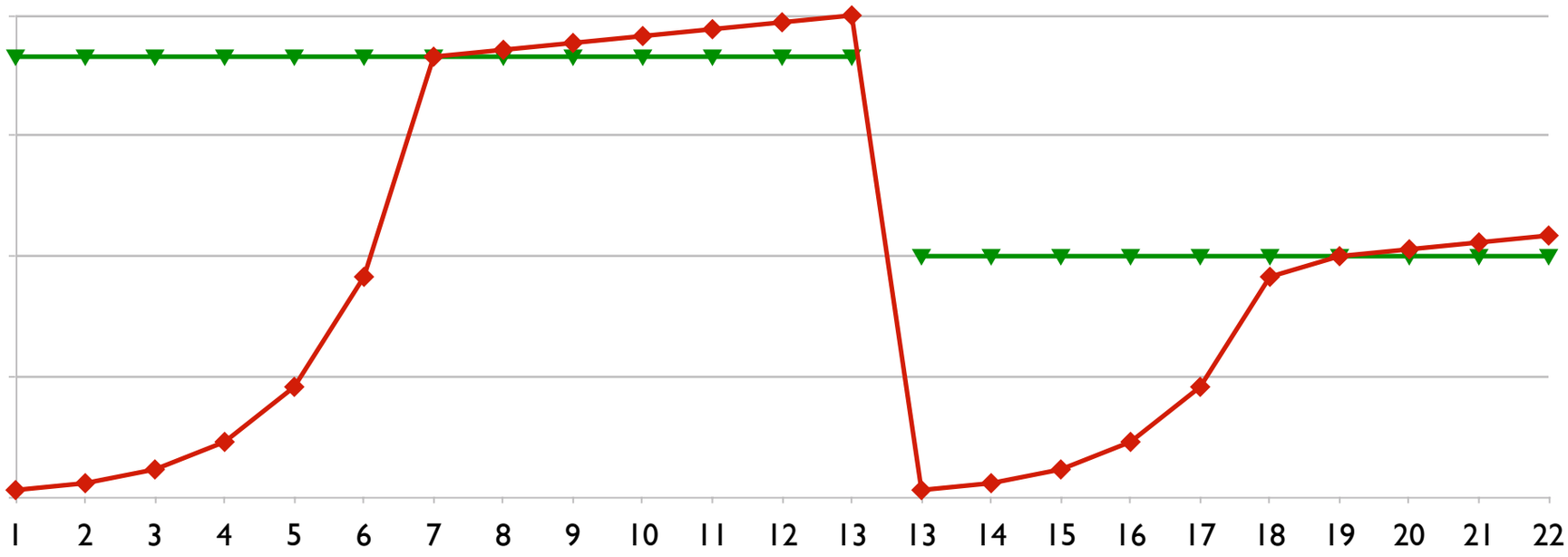
La valeur typique est 1460 octets (MTU d'Ethernet - 40 octets pour les en-têtes IP)

Si non spécifié, valeur par défaut = 536 octets

Mécanismes : Congestion Avoidance

- **Il existe un seuil au delà duquel l'émetteur devient plus prudent**
  - Valeur initiale du seuil : 64 ko
  - Une fois que la fenêtre a atteint cette valeur, augmentation de la taille de la fenêtre d'une taille de segment (MSS) à chaque RTT.

- **Lors de la perte d'un segment (non-réception d'un acquittement)**
  - Réinitialisation de la fenêtre à 1 MSS
  - Le seuil d'évitement de congestion prend alors la valeur courante de la fenêtre divisée par 2.



Mécansimes : Fast retransmit et fast recovery

- **Fast Retransmit**
  - Le temporisateur pour conclure à la perte d'un segment peut parfois être raccourci.
  - Un récepteur acquitte toujours le dernier octet reçu dans l'ordre, en cas de perte, il y aura duplication d'acquittements
  - L'émetteur, recevant 3 acquittements dupliqués (4 en tout), sait qu'il doit retransmettre directement le segment incriminé

- **Fast recovery**
  - Après une phase fast retransmit, on repart en mode collision avoidance (croissance linéaire), sans passer par la phase de slow start (croissance exponentielle). Dans ce cas, on est en présence d'une congestion modérée puisque les segments suivants ont été reçus.
  - En cas d'expiration d'un timer, on repart en phase de slow start (indique une congestion plus importante)
  - Autorise un débit élevé en présence de congestion modérée

Versions de TCP (Flavours)

- **De nombreuses versions de TCP ont été spécifiées**

Version	Année	*Slow Start*	*Congestion Avoidance*	*Fast Retr / Recov.*	*Autres*
Tahoe	1988	oui	oui	partiel	
Reno	1990	oui	oui	oui	
New Reno		oui	oui	oui	Phase de Fast recovery améliorée
Vegas	1994	oui	oui	oui	Utilisation d'un historique pour réduire la fenêtre avant perte
SACK	1996	oui	oui	oui	Acquittements sélectifs (pertes isolées) — Option TCP
BIC		oui	oui	oui	Dédié pour les réseaux à haute latence
CUBIC		oui	oui	oui	Dédié pour les réseaux à haute latence ; plus agressif que BIC
Westwood		oui	oui	oui	New Reno adapté au sans-fil (large produit bande passante x délai ; erreurs de ctransmission)

Protocoles de transport sur IP

- **Plusieurs protocoles de transport existent dans le monde TCP/IP**

- **UDP (User Datagram Protocol)**
  - minimaliste, ne fournit pratiquement que des numéros de port
- **TCP (Transmission Control Protocol)**
  - Service connecté : fiabilité, notion de sessions, ordonnancement des segments
  - Contrôle de congestion

- **DCCP (Datagram Congestion Control Protocol)**
  - Contrôle de congestion sur UDP
- **SCTP (Stream Control Transmission Protocol)**
  - Protocole versatile et configurable

Conclusion

- **Deux protocoles de transport principaux existent**
  - UDP : minimaliste fournit simplement des numéros de port
  - TCP : fiabilité (acquittement), notion de connexion (ouverture, fermeture de session), contrôle de congestion (contrôle de la fenêtre d'émission)

- **Lorsqu'une congestion survient, TCP adapte son débit, UDP subit des pertes**
  - Co-existence a priori non-équitable vis-à-vis du débit de TCP

- **Aujourd'hui une majorité des flux dans Internet sont des flux TCP**

- **D'autres améliorations existent (ECN, ...) et d'autres protocoles sont proposés (DCCP, SCTP, ...)**