TCP/IP
Protocole de transport

Jean-Patrick Gelas
Université de Lyon

Sources

- « Réseaux locaux et Internet », 3rd édition, Laurent Toutain, Hermes.
- “Analyse Structurée des réseaux, 2nd édition”, James Kurose, Keith Ross. Pearson Education.
- “Réseaux et Télécoms”, 2nd édition, Claude Servin. Dunod.
- Jacobson, Van (1995), "Congestion Avoidance and Control", ACM SIGCOMM Computer Communication Review 25(1): 157 - 187
- Data Transport Research Group, GGF (Global Grid Forum), “*A survey of Transport Protocols other than Standard TCP*”. Eric He, Pascale Primet, Michael Welzl et al., May 10,2005.
- http://research.csc.ncsu.edu/netsrv/?q=content/bic-and-cubic

Historique

- **1970** : Première publication du « ARPANET Host-Host protocol » C.S. Carr, S. Crocker, V.G. Cerf, "HOST-HOST Communication Protocol in the ARPA Network," in AFIPS Proceedings of SJCC.
- 1974 : Vint Cerf et Bob Kahn publie un article qui spécifie en détail l'architecture d'un programme de contrôle de transmission (TCP). « A protocol for Packet Network Interconnection », IEEE Trans comm.
- 1975 : Satellite links cross two oceans (to Hawaii and UK) as the first TCP tests are run over them by Stanford, BBN, and UCL
- 1978 (mars) : TCP se scinde en TCP et IP.
- **1981** (septembre) : RFC793 - Transmission Control Protocol
- 1982 : DCA and ARPA establish the Transmission Control Protocol (TCP) and Internet Protocol (IP), as the protocol suite, commonly known as TCP/IP, for ARPANET.
  - This leads to one of the first definitions of an "internet" as a connected set of networks, specifically those using TCP/IP, and "Internet" as connected TCP/IP internets.
  - DoD declares TCP/IP suite to be standard for DoD
- 1989 (octobre) : RFC1122 - Requirements for Internet Hosts - Communication Layers

http://www.zakon.org/robert/internet/timeline/

Les algorithmes TCP historique

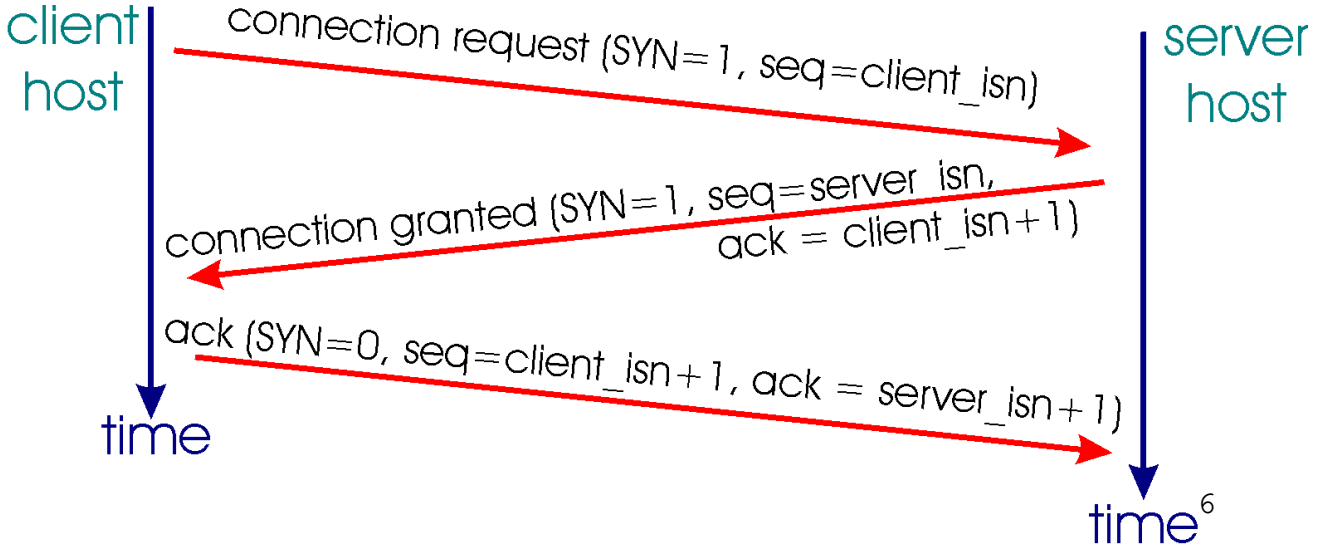
- 1986 – Tahoe : Première implémentation. La plus simple et la moins efficace (cwnd=1 MSS à chaque congestion, *ssthreshold* = *cwndMax* / 2)
- **1990** – Reno : Usage de *Fast recovery. Cwnd=cwnd / 2.* Débit offert un peu plus stable.
- 1994 – Vegas : Prise en compte du temps de réponse du destinataire (RTT). Modifications des trois algorithmes (slowstart, congestion avoidance et Fast retransmit).
- 1999 – New Reno : Modification légère de Reno (reste en mode Fast recovery tant qu'il n'a pas reçu les ACK de tous les paquets perdus).

Introduction

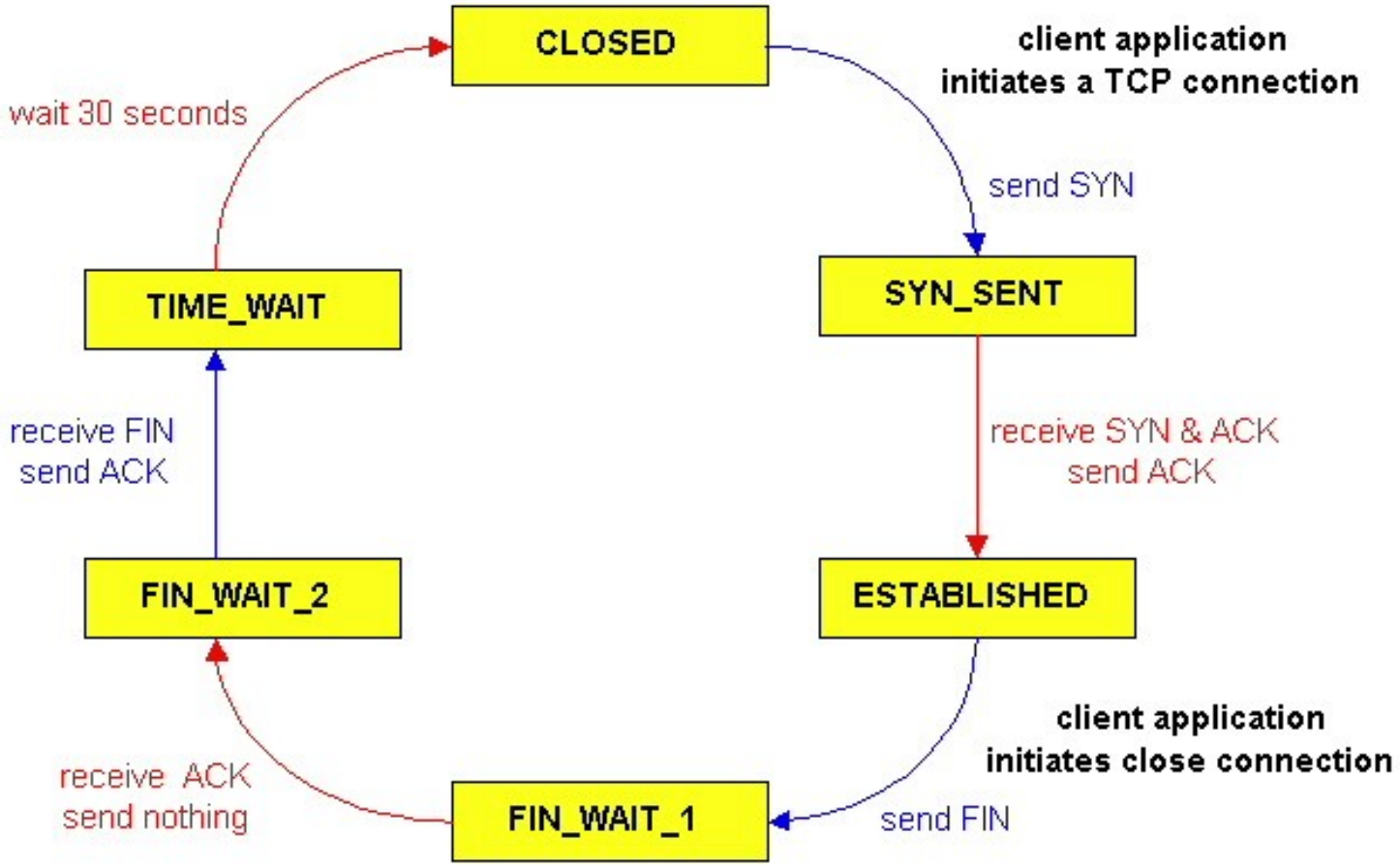
- Le protocole TCP standard (*TCP Reno*) est un protocole de transport fiable qui fonctionne bien dans les réseaux traditionnels.
- Cependant, les expériences et l'analyse de ce protocole montre qu'il n'est pas adapté à toutes les **applications** et à tous les **environnements,** par exemples :
  - les communications interactives ;
  - le transfert haut débit de grosse quantité de données (*bulk data transfer*) ;
  - les réseaux dont le produit *débit* et *temps d'aller/retour* (*throughput* x *RTT (round-trip time)*) sont grand.
  - réseaux sans-fils.

La gestion de connexion TCP
Établissement de connexion en trois étapes

- La procédure d'établissement d'une connexion TCP constitue une partie non négligeable du temps de réponse perçu par les utilisateurs.
- Etape 1 : **segment SYN** (*bit SYN=1*), le client choisi un numéro de séquence initial (seq = *client_isn*).
- Etape 2 : **segment SYNACK**. Le récepteur alloue un tampon et des variables, renvoie un datagramme avec *SYN=1*, seq = *server_isn*, ack = *client_isn+1*
- Etape 3 : Le client alloue un tampon des variables, et accuse réception de cette autorisation de connection.
  SYN=0, seq = cleint_*isn+1, ack = server_isn+1*

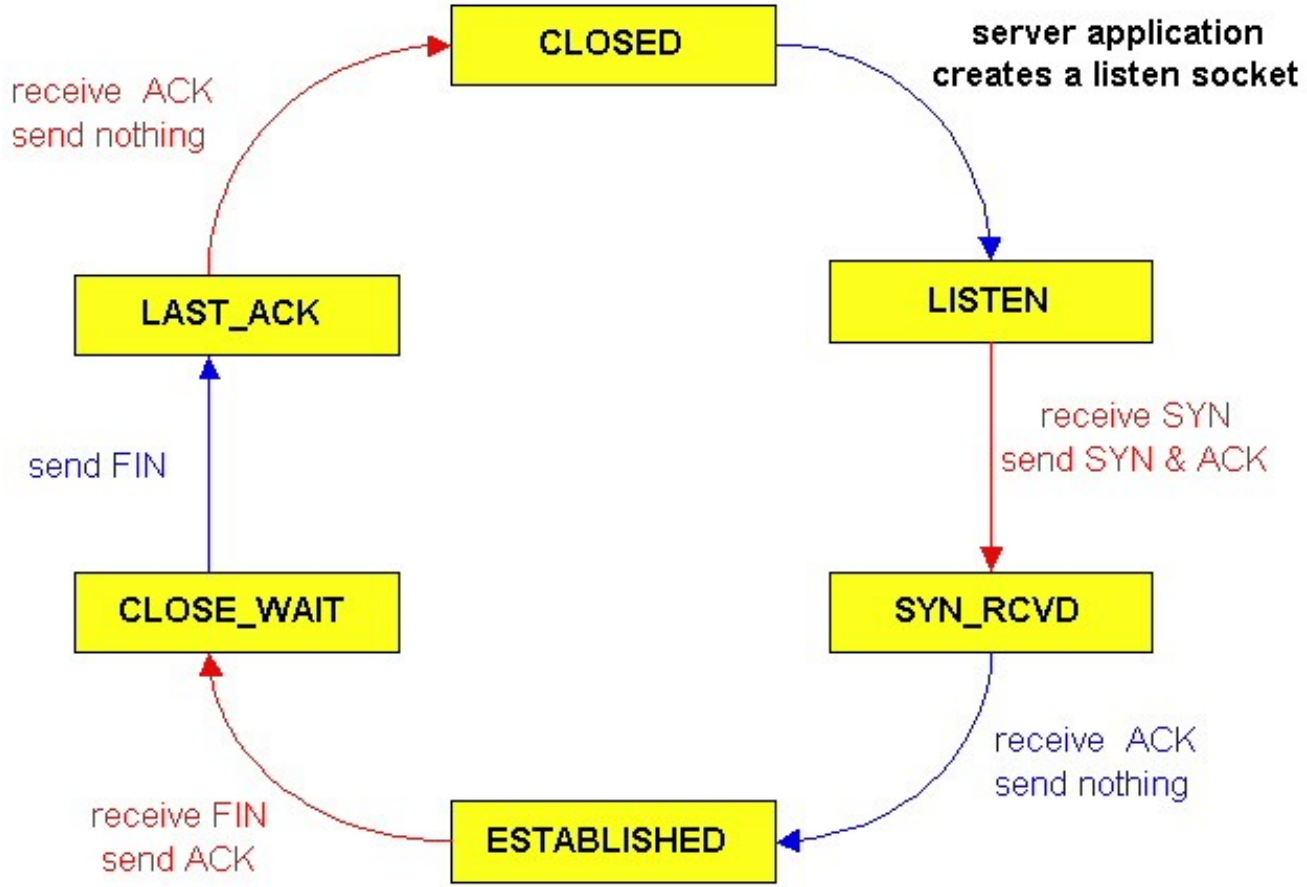


La gestion de connexion TCP



*Etats TCP traversé par le pôle **client** TCP.*

La gestion de connexion TCP



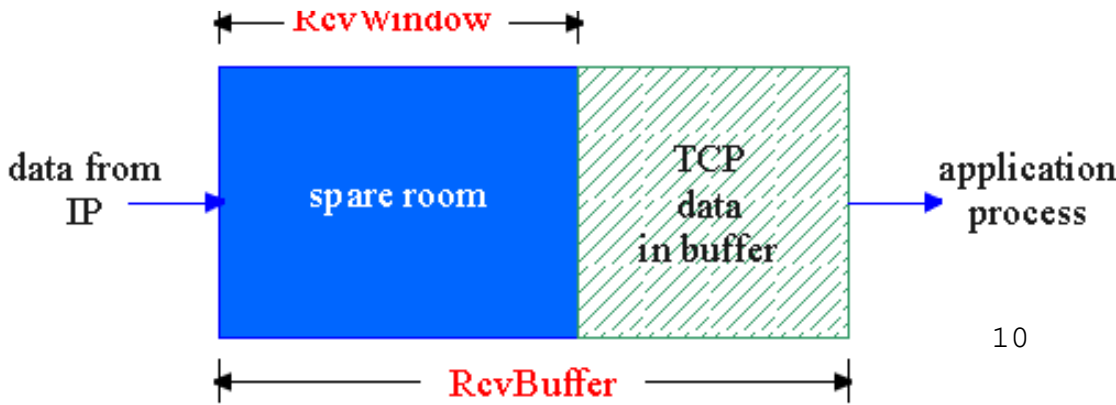
*Etats TCP traversé par le pôle **serveur** TCP.*

Deux composants

- **Contrôle de flux** : Comment faire pour être sur que le récepteur reçoit aussi vite que l'on émet ?
- **Contrôle de congestion** :
  Comment être sur que le réseau délivre les paquets au récepteur ?

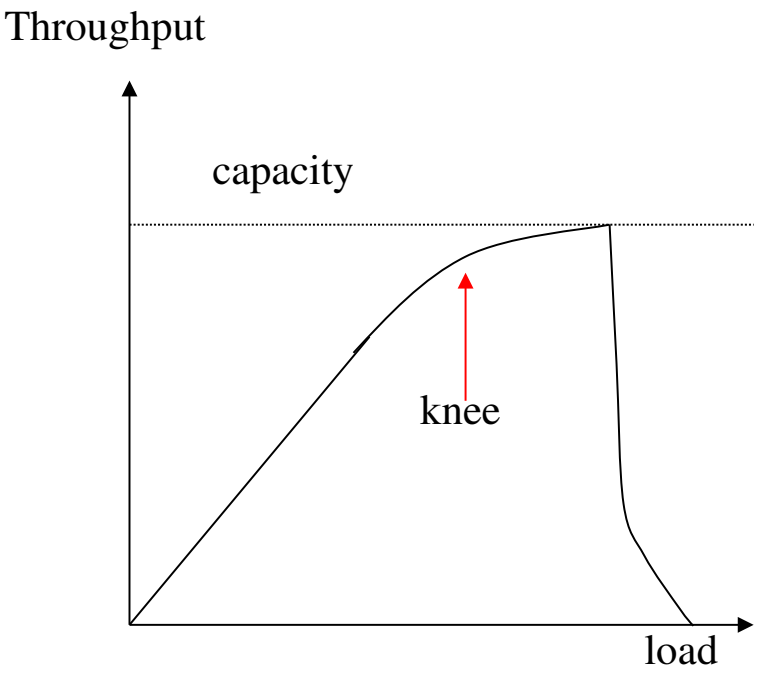
Le contrôle de flux de TCP

- Chaque pôle d'une connexion à un tampon de réception.
- L'application vient chercher ses données dans ce tampon de façon asynchrone.
- Sans service de contrôle de flux un émetteur pourrait rapidement saturer le tampon de réception (*RcvBuffer*) dont la taille est fixe.
- L'expéditeur est informé de la taille variable de la **fenêtre de réception** (*RcvWindow*) du destinataire.
- Le récepteur informe l'expéditeur de l'espace disponible en insérant la variable *RcvWindow* dans la fenêtre de réception de tous les segments qu'il lui envoie.
- Lorsque *RcvWindow*=0 et que le récepteur n'a rien à envoyer, l'émetteur continue à envoyer des segments de 1 octets qui généreront des acquittements porteur des informations d'actualisation sur le tampon de réception.



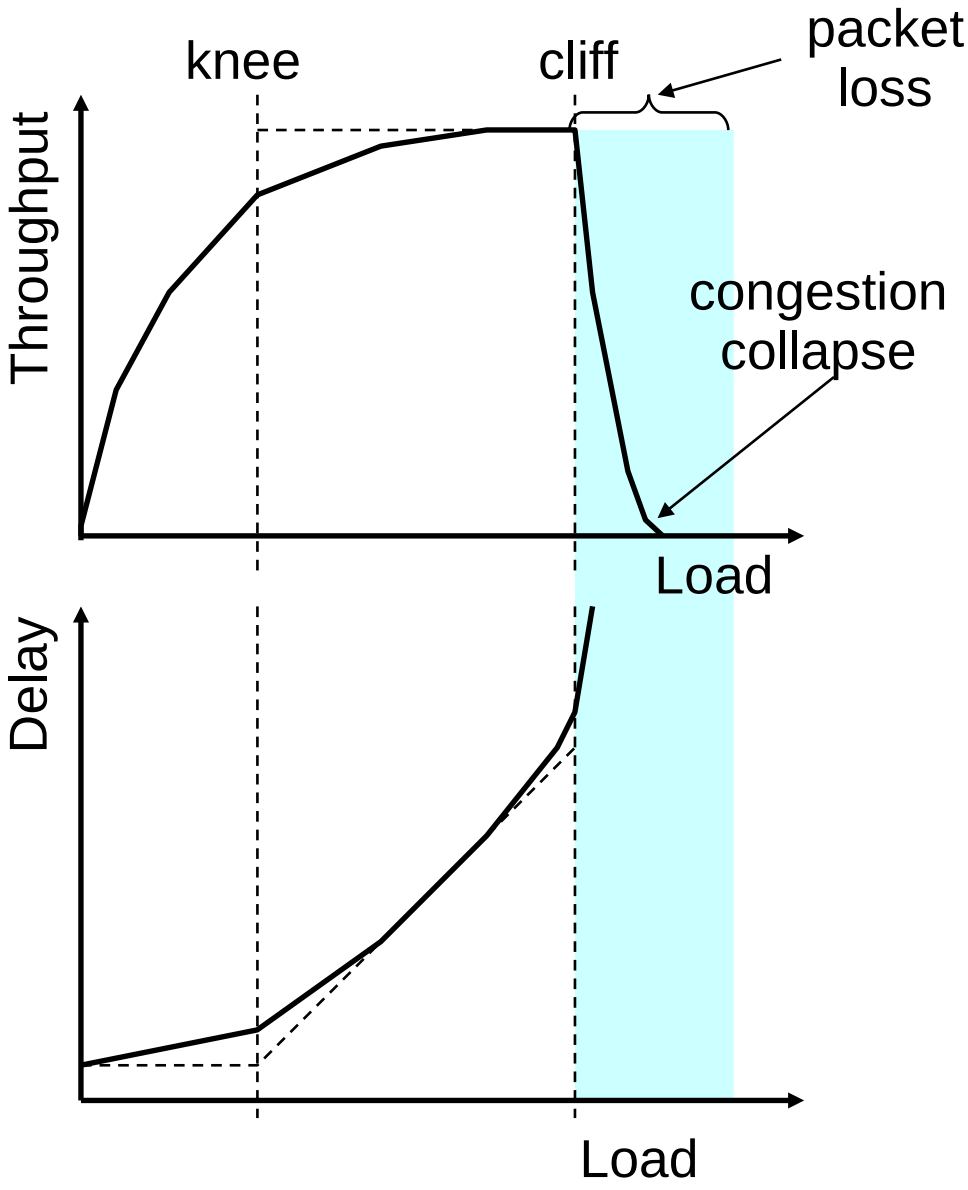
Le contrôle de congestion de TCP

- L'objectif principal du contrôle de congestion est d'éliminer la congestion.
- Lorsque la demande totale du trafic excède la capacité (*throughput*) du lien, cette demande doit être réduite.
- Sinon, *congestion collapse* potentiel... (événement couramment observé en 1986 avant l'ajout du contrôle de congestion)

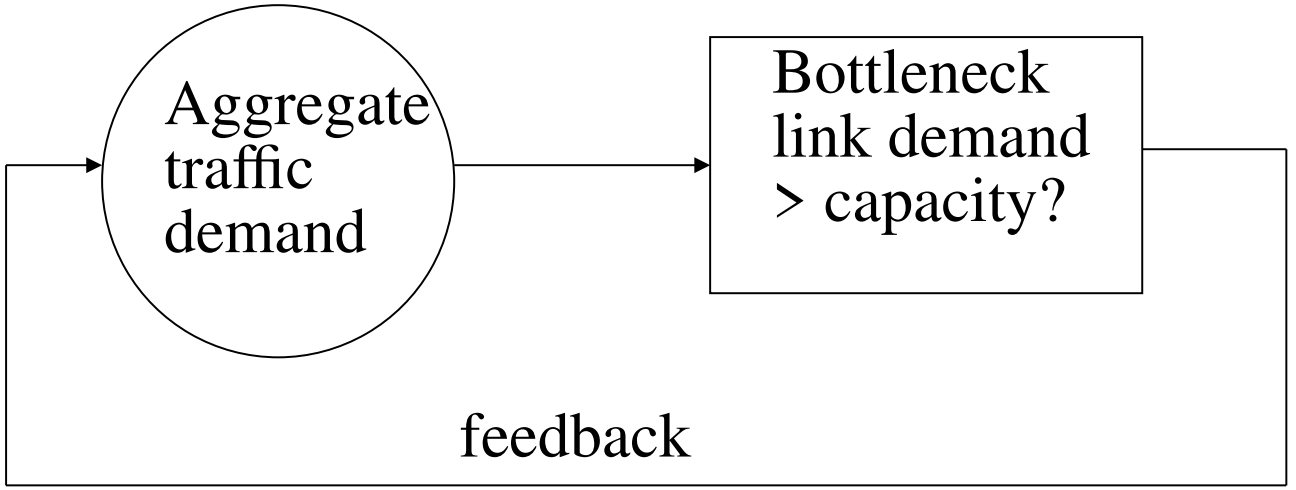


What’s Really Happening?

- **Knee** – point après lequel
  - Le débit augmente doucement
  - Le délai augmente
- **Cliff** – point après lequel
  - Le débit décroit rapidement (congestion collapse)
  - Le délai tend vers l'infini



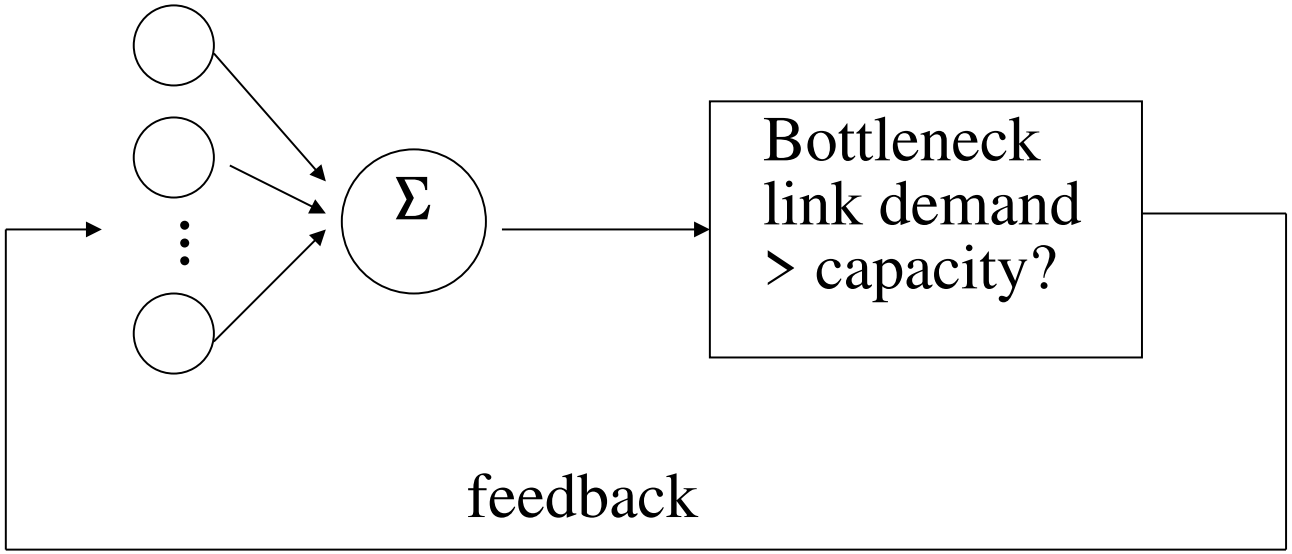
Modèle de base du contrôle de congestion



- Quel est le type de *feedback* ?
- Est-ce stable ?

Contrôleurs multiple

- Dans le cas du contrôle de congestion de réseaux (plusieurs connections partageant le même lien), il y a plusieurs contrôleurs



- Est-ce équitable ? (fairness)

Boucle de rétro-action binaire

- Le cas le plus simple (*binary feedback*) :
  - Le réseau est congestionné ou pas.
- TCP utilise comme indication de congestion binaire la destruction (ou non) de paquets.

- Est-ce adéquat ?

Le contrôle de congestion de TCP

- Puisque IP ne fournit aucune information explicite aux terminaux, TCP a nécessairement recours à une approche de **bout-en-bout** (plutôt qu'à un contrôle de congestion assisté par le réseau).
- La vitesse d'émission est régulée en fonction du niveau de congestion **perçu**.
- Régulation du taux d'envoi :
  - La variable **fenêtre de congestion** (*CongWin* ou *cwnd*) impose une limite au rythme auquel l'expéditeur est autorisé à charger des données sur le réseau.
  - Le volume de donnée non-confirmées ne peut dépasser le minimum de *CongWin* et *RecvWin*.

  $LastByteSent - LastByteAcked = \min\{CongWin, RecvWin\}$
- Le “phénomène de perte” est identifé/perçu soit par :
  - l'expiration du temps imparti (*timeout*), soit par
  - la réception d'ACK dupliqués de la part du destinataire,

et interprété comme un indice de congestion sur le parcours vers le destinataire.

Effets de bords des protocoles à évitement de congestion

- **Wifi** : Les protocoles à évitement de congestion supposent toujours que les pertes sont dues à une congestion. Sur un réseau Wifi cela conduit à un débit faible.
- **Connexions courtes** (*short lived connections*) : Le protocole *slow-start* est inadapté pour les connexions très courte. Il est conseillé d'utiliser une même connexion dans laquelle on effectuera plusieurs transactions courtes.

Rappel sur le contrôle de congestion de TCP :

Départ lent (*slow start*)

- Au début d'une connexion, *CongWin* = 1 MSS, donc
  un taux d'envoi ~= MSS/RTT (ex: 500 bytes / 200 ms = 20 kbit/s).

- Au début, l'émetteur à en fait recours à une accélération exponentielle. Pour chaque ACK reçu **cwnd = cwnd + 1**.

- Autrement dit, cela consiste à doubler la taille de la *CongWin* à chaque RTT.

  1 MSS, 2 MSS, 4 MSS,...

- Cette technique est utilisé jusqu'à :
  - ce qu'il y ait une perte*,* moment auquel *CongWin* est **divisé par 2** ou
  - qu'on atteigne une valeur seuil ***slowstart threshold****.*

*et passe en mode de **progression linéaire** ou évitement de congestion (congestion avoidance).*

Rappel sur le contrôle de congestion de TCP :

AIMD : Accroissement additif et décroissance multiplicative

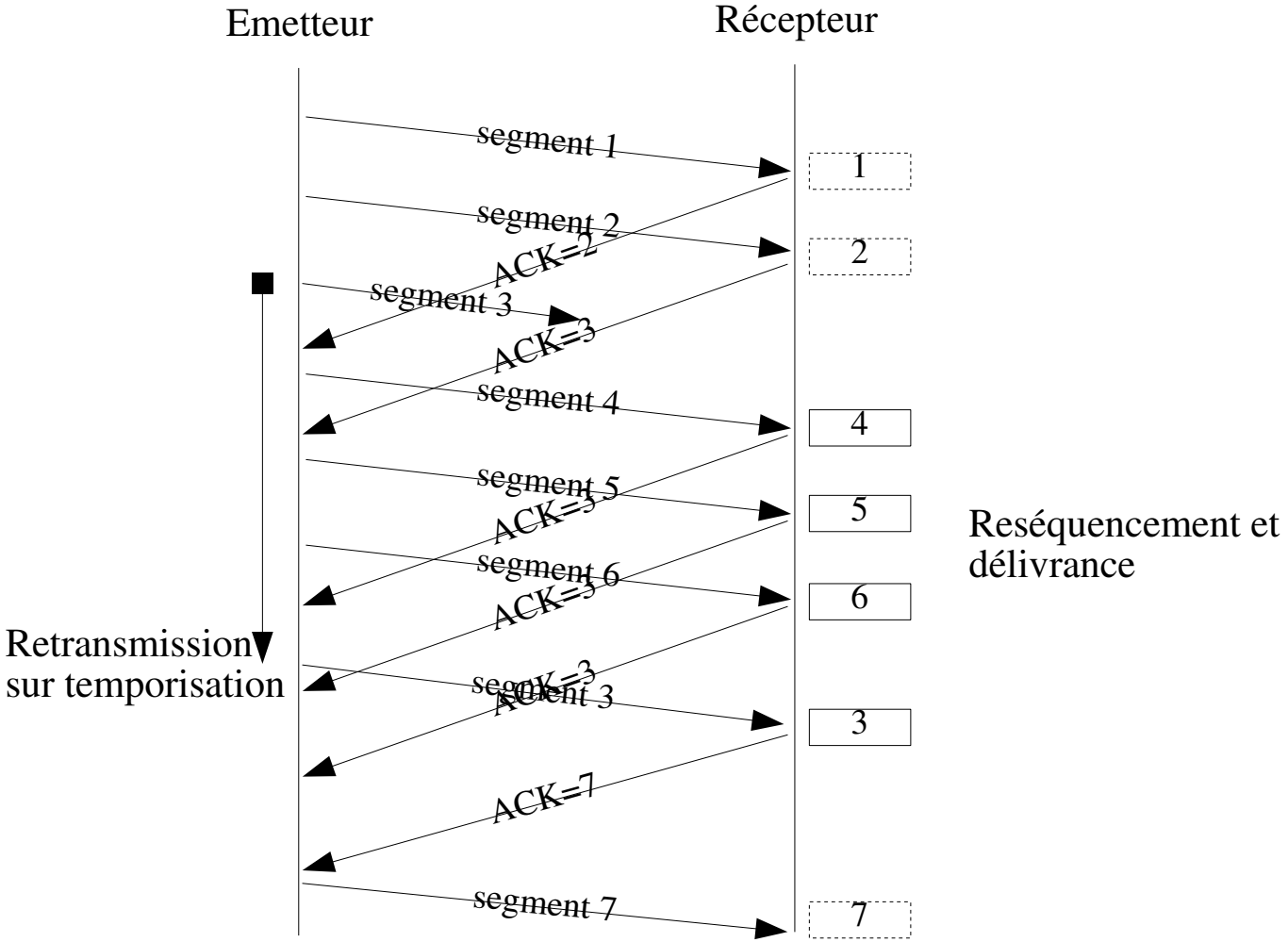
- L'idée de base est de faire en sorte que l'expéditeur réduise son taux d'envoi en diminuant la taille (par 2) de sa fenêtre de congestion dés qu'un phénomène de perte se déclare (ex: 20ko -> 10ko -> 5 ko ... -> 1 MSS (*Maximum Segment Size*)) : **cwnd = cwnd / 2**

- Si le réseau ne présente pas de congestion, une “certaine” valeur de débit doit être disponible pour la connection TCP.
- TCP procède a un agrandissement progressif de sa *CongWin*, “sondant” scrupuleusement toute éventuelle fraction de débit disponible sur le chemin de bout-en-bout : Pour chaque ACK reçu **cwnd = cwnd + 1/cwnd**.
- autrement dit, TCP ajoute environ 1 MSS à chaque temps de trajet aller/retour (RTT) tant qu'aucun phénomène de perte ne se déclare.

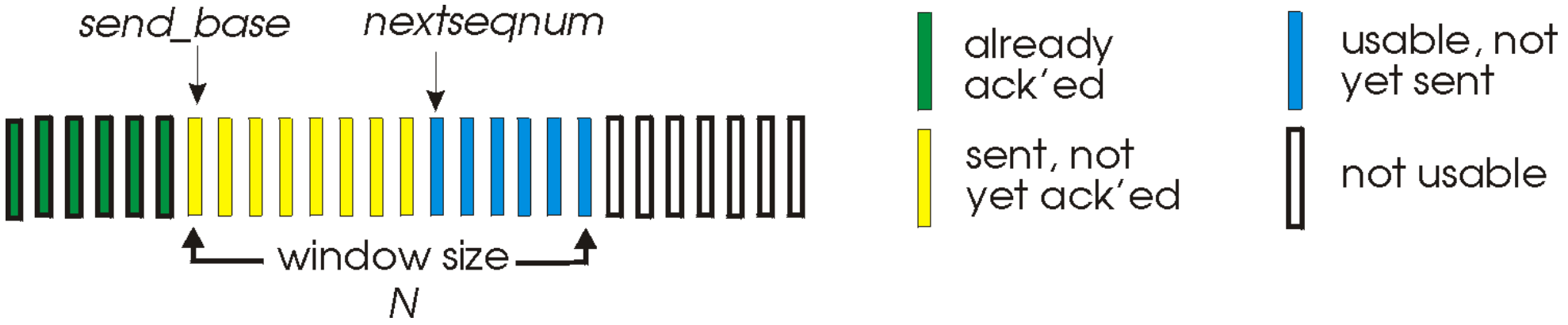
- L'algorithme de congestion de TCP est donc appelé AIMD (*Addititve Increase, Multiplicative Decrease*).
- La phase de croissance linéaire -> phase d'évitement de congestion (*congestion avoidance*).

La reprise sur temporisation

- Même si des données sont correctement reçues, TCP ne les acquitte pas si leur numéro de séquence est supérieur à celui attendu.
- Il renvoit un acquittement contenant le numéro de séquence attendu.

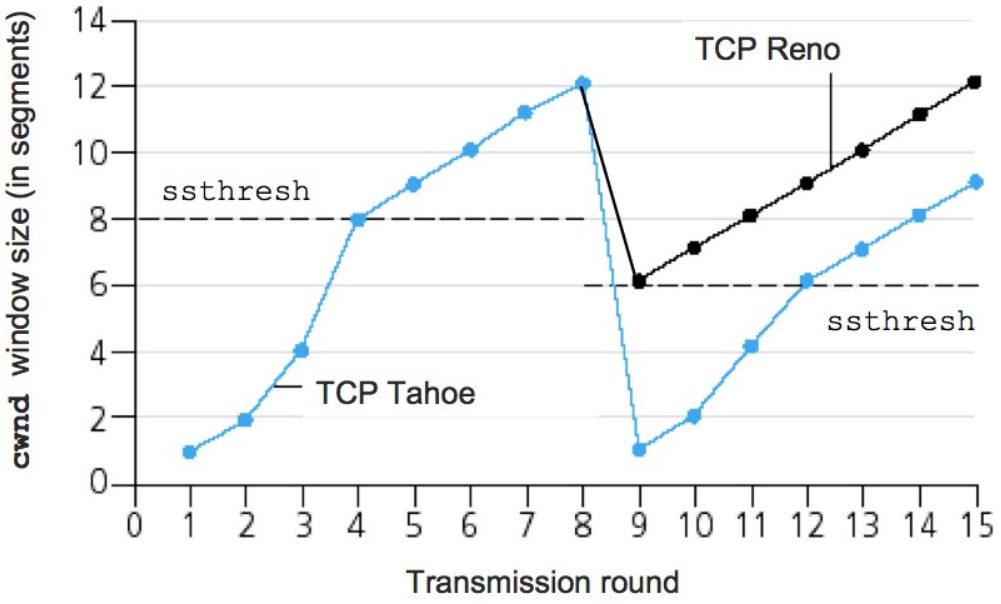


Le contrôle de congestion de TCP
La fenêtre glissante



Réaction au phénomène d'expiration du temps imparti ou du triple ACK

- Le phénomène de congestion de TCP ne réagit pas de la même manière à un phénomène de perte détecté par :
  - l'expiration du temps imparti (*timeout*), ou
  - l'arrivée de 3 ACK identiques
- Pour le cas ***Timeout*** : CongWin=1, agrandissement exponentielle jusqu'à une valeur seuil (*SSTreshold*),puis agrandissement linéaire.
- Pour le cas **Triple ACK** : L'annulation de la phase de départ lent après un triple ACK est appelé récuperation rapide (*fast recovery*).
- *SSThreshold* initial par défaut = 64 ko, en cas de perte *SSTreshold = CongWin / 2*



Fast Retransmit

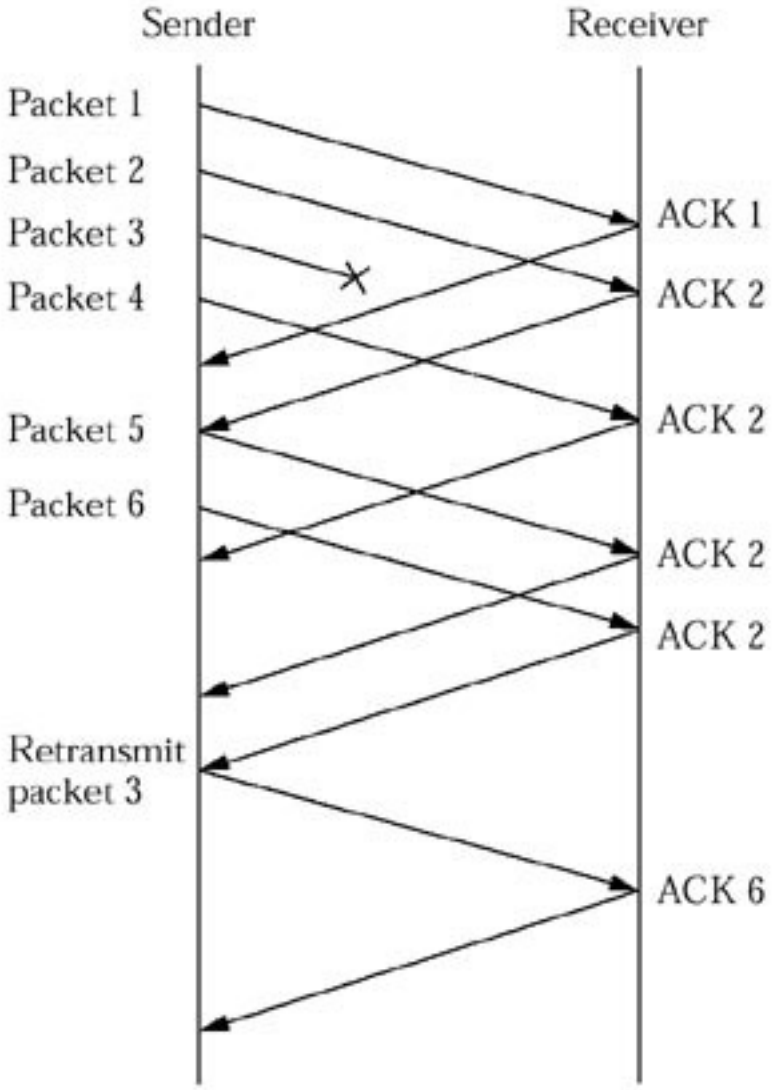
(Retransmission rapide)

- Les version *Reno* et *NewReno* de TCP définissent une amélioration de l'algorithme qui limite le nombre de paquets à retransmettre.
- Lorsqu'un récepteur reçoit un paquet « désordonné », il envoit un ACK dupliqué et met en cache le paquet « désordonné ».
- Lorsque l'émetteur reçoit 3 ou plus de ACK dupliqués, le paquet est considéré comme perdu.
- Un paquet dont la perte est détecté via des acquittements dupliqués, est retransmis immédiatement, sans attendre le *timeout*.

Fast retransmit
(illustration)

Exploite les acquittements dupliqués pour déclencher une retransmission.

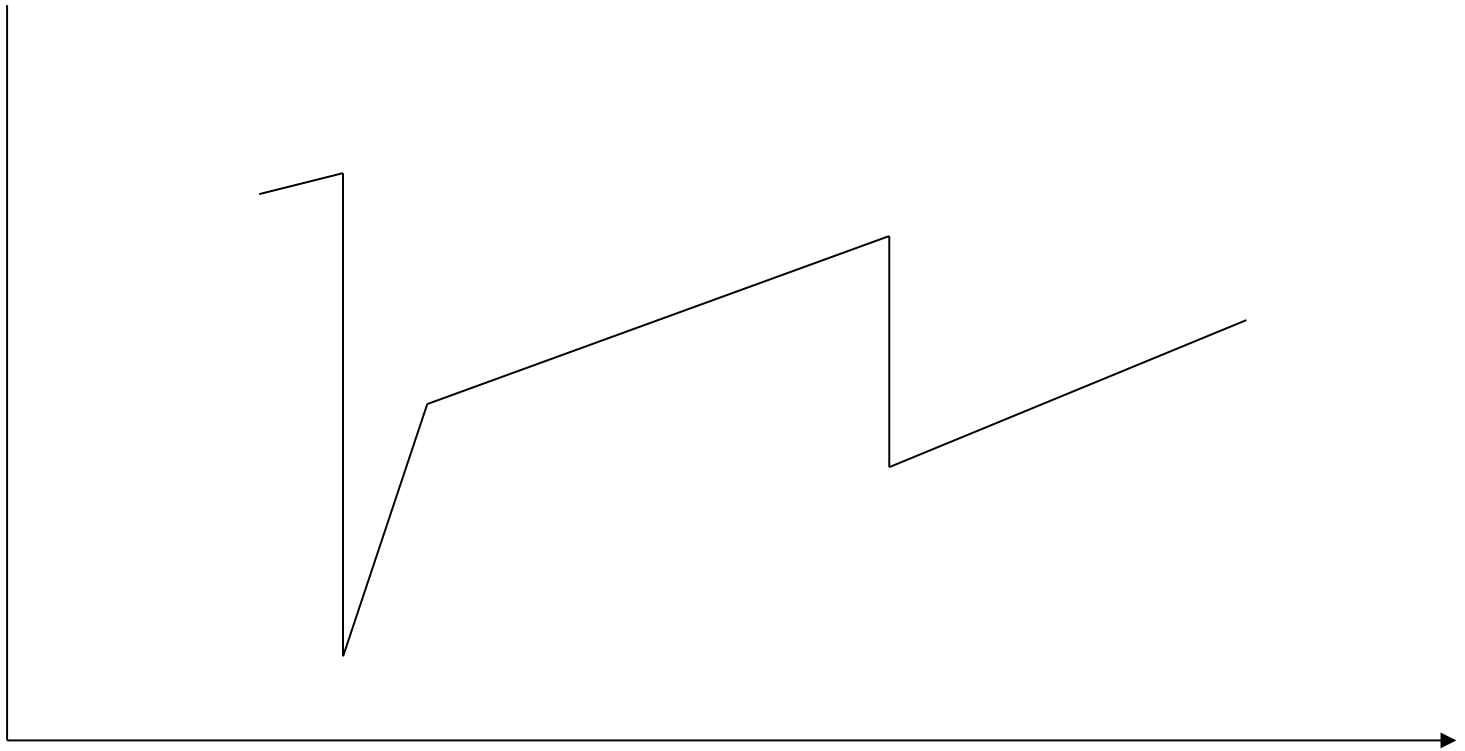
(Permet d'éviter les temps morts (*idle periods*))



Fast recovery

- Technique permettant d'éviter de réduire le débit d'une façon trop brutale.
- Sender enters “fast recover” phase
  - Reduce **ssthresh** by half
  - Temporarily reduce **cwnd** to (**ssthresh** + #**dup_ack**)
  - Retransmit missing packet, assuming receiver caches out-of-order packets
- When receiver receives the retransmitted packet (filling the gap) it acks all packets (incl. out-of-order ones).
- Sender set **cwnd** to **ssthresh** and returns to congestion avoidance phase.
- See RFC2001

Comportement dynamique

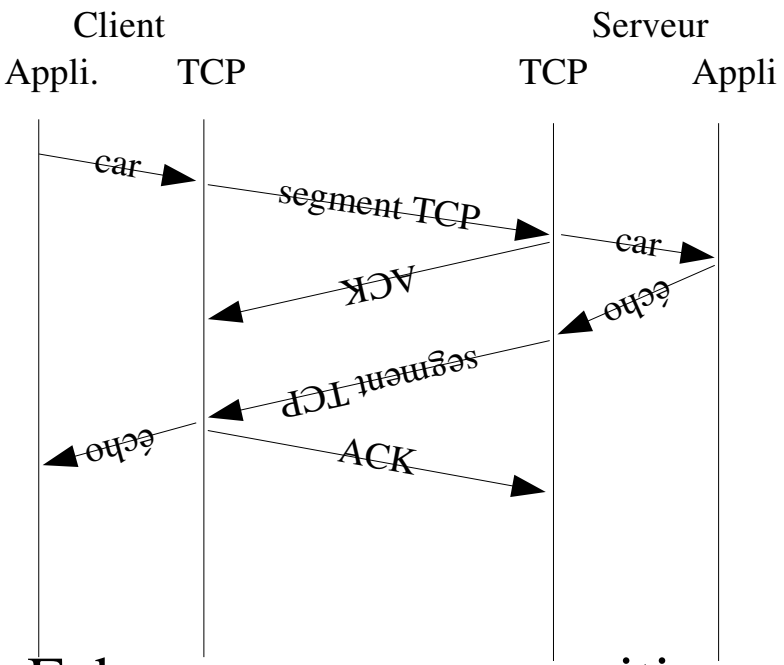


Packet loss by timeout

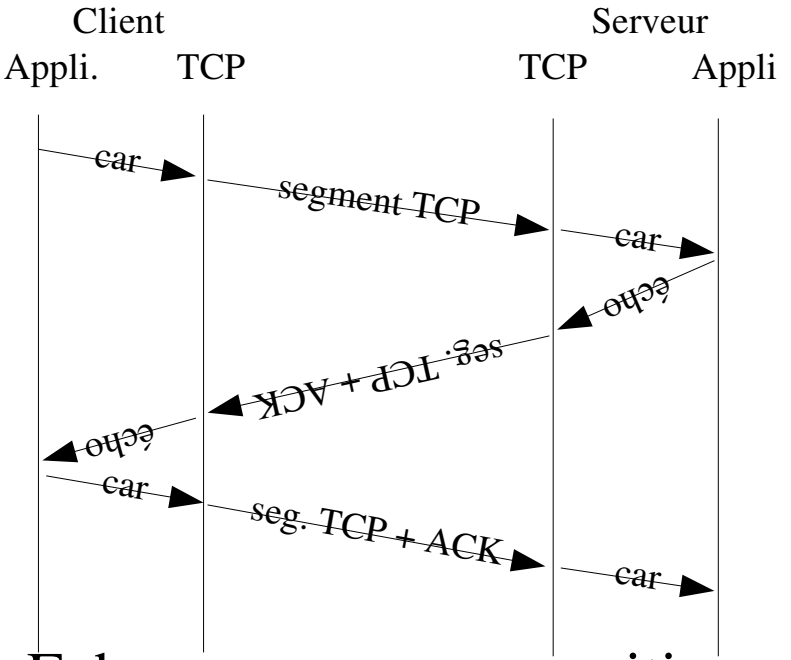
Packet loss by fast retransmit and recovery

La gestion des acquittements

- TCP utilise le mécanisme de l'anticipation et met en oeuvre la technique de la fenêtre glissante.
- Chaque bloc émis doit être acquitté immédiatement.
- Soit une application de type TELNET (illustration de la superposition *(piggybacking))*.



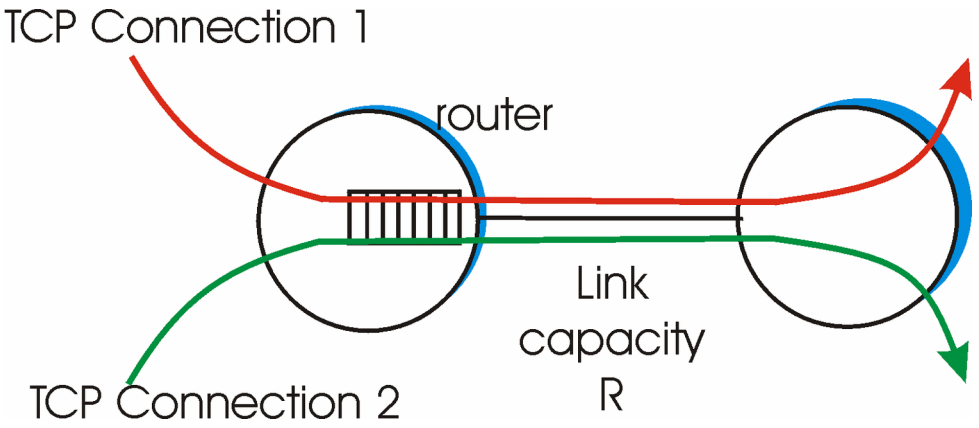
Echange sans superposition

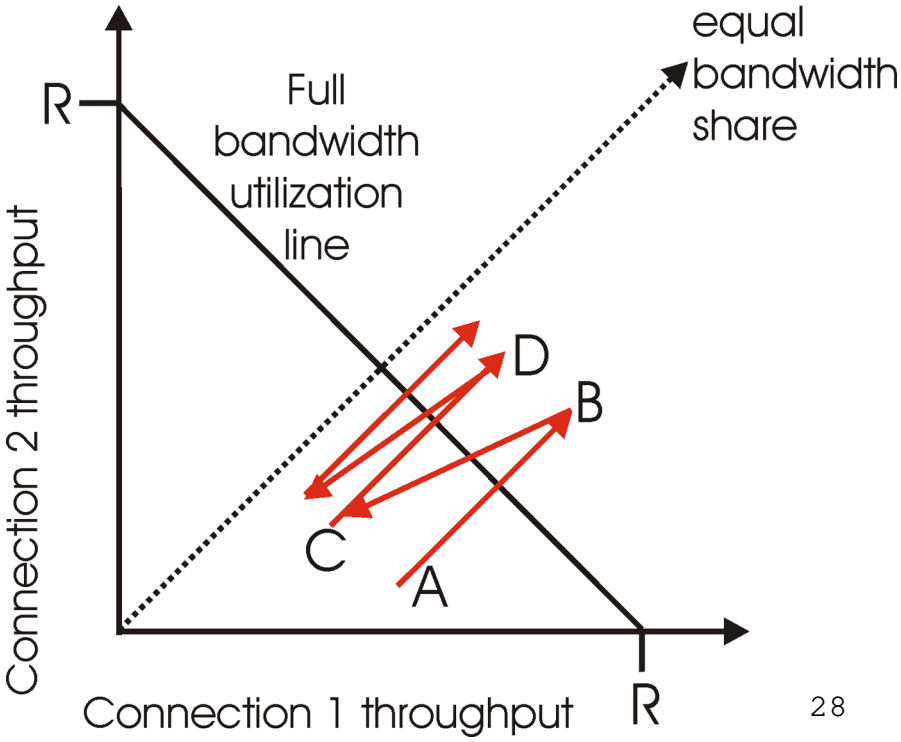


Echange avec superposition

Equité (*fairness*)

- Soit *K* connections TCP empruntant toutes une même liaison “faible” (goulet d'étranglement) d'un débit de *R* bit/s.
- On suppose qu'il n'y a pas de flux UDP.
- Un système de contrôle de congestion est dit équitable si le débit moyen de chaque connexion ~= *R* / *K*
- AIMD est-il équitable ? (connexions pas forcément établi en même temps, fenêtre de taille différente).





Fairness in the real world!

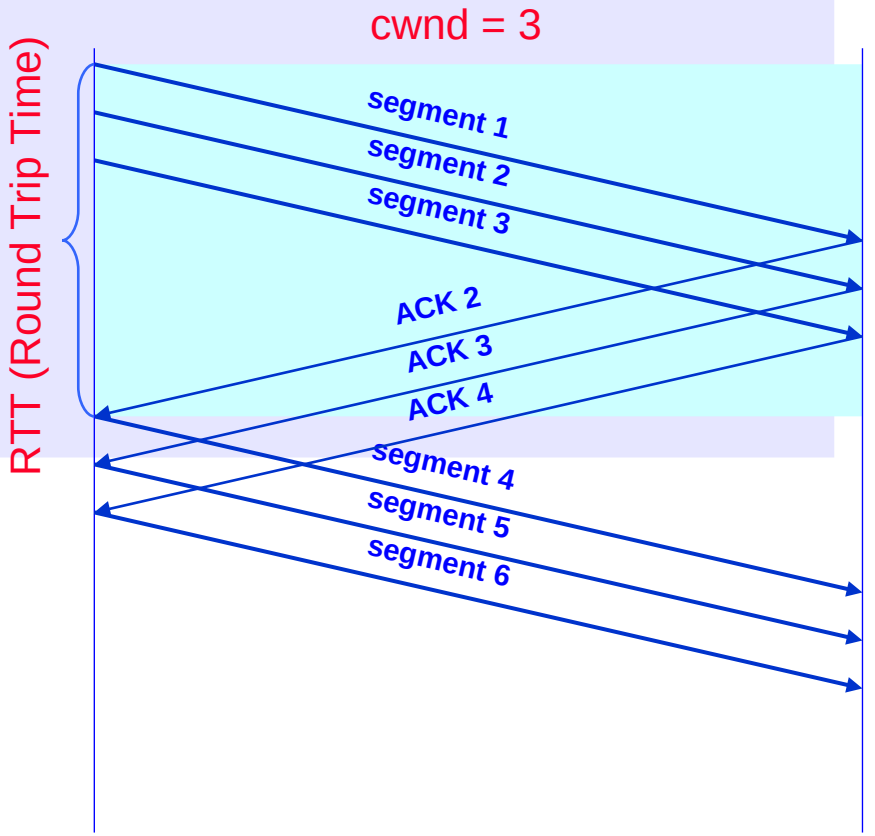
- L'illustration précédente fait beaucoup d'hypothèses simplificatrice...
- En pratique les applications client/serveur se voient souvent attribuer des débits très **inégaux**.
- Dans le cas d'un goulet d'étranglement, les sessions présentant le plus petit RTT sont souvent plus promptes à s'arroger le peu de débit disponible (i.e agrandir leur fenêtre de congestion).

- Les applications multimédia utilisent UDP car elles ne peuvent se permettre de voir leur débit bridé.
- Du point de vue de TCP les **applications multimédia** ne sont **pas équitables** car elles ne **coopèrent pas** avec les autres connections et n'adapte jamais leur débit aux **conditions du réseau**.

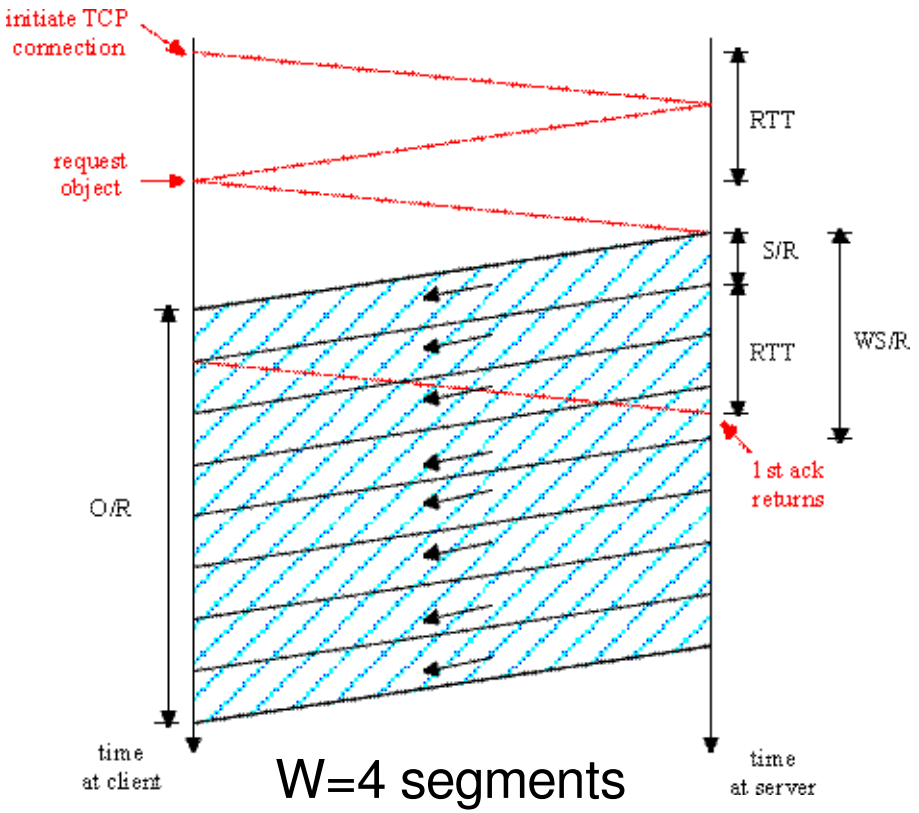
- Il n'y a a priori pas moyen d'empêcher une application (TCP) d'utiliser plusieurs connexions en parallèle (ex: navigateur web) $\rightarrow$ L'application bénéficie alors d'une plus grande proportion de la bande passante.

Taille de fenêtre et débit

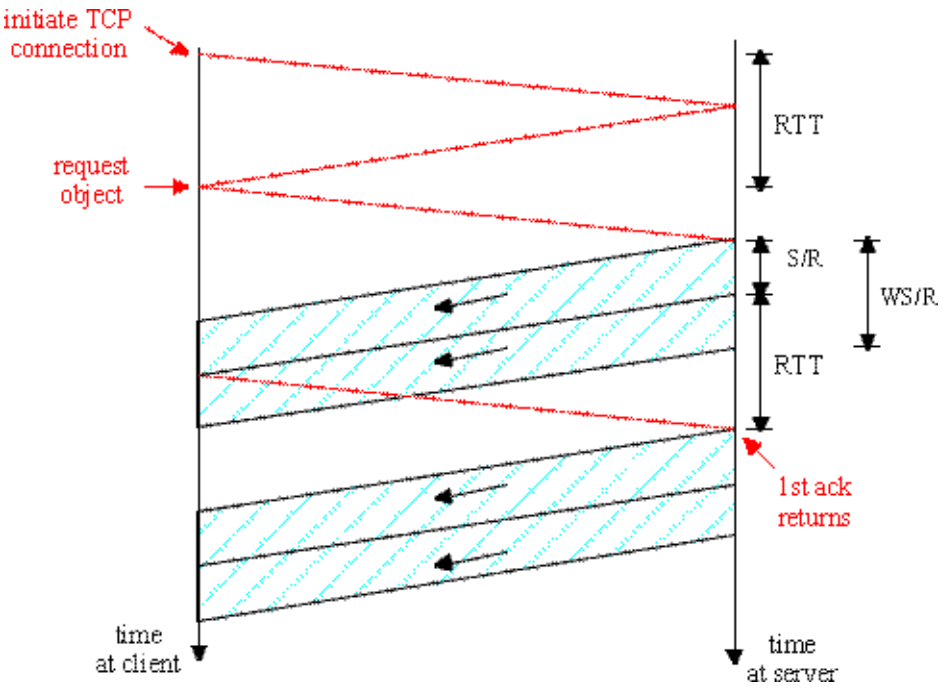
- De la taille de fenêtre de congestion dépend le débit !
- Throughput = cwnd / RTT
- Attention aux séquences...



Fenêtre de congestion statique



W=4 segments



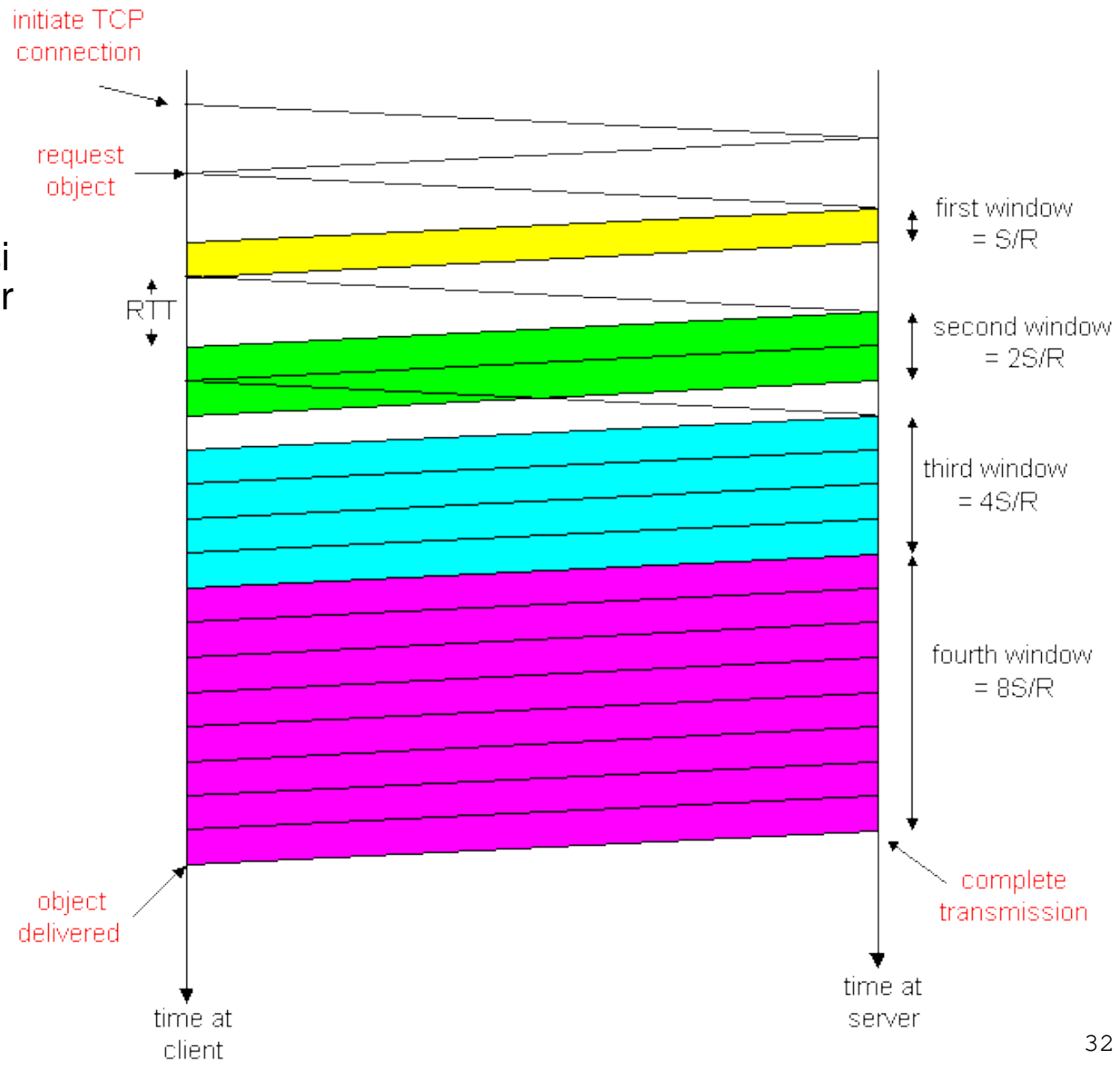
W=2 segments

O bits = Taille de l'objet à transmettre
S bits = Taille de segment maximum (MSS) (ex: 536 octets)
R bit/s = Débit de la liaison

Le **temps de latence** (*latency*) est le laps de temps séparant l'établissement de la connection TCP par un client et la réception de l'objet dans son intégralité.

Fenêtre de congestion dynamique

Le *slow start* n'a pas de répercussion significative, a priori, sur le temps de latence si RTT << O/R (i.e si le temps de trajet aller-retour est très inférieur au temps de transmission de l'objet).



TCP sur différents supports

- TCP utilisé universellement pour le transport fiable de données sur Internet.
- Conçu à la fin des années 70 : époque où la topologie du réseau et les caractéristiques des liens étaient différentes.
- TCP doit suivre les évolutions technologiques :
  - Débits : de 9600 b/s (GSM) au Terabit/s (fibres optiques)
  - Délais de propagation : de quelques microsecondes à une seconde.
  - Taux d'erreur : de quasi nul (fibre) à très élevé sur les liaisons hertziennes (satellite)

Réseaux haut débit

TCP sur différents supports

- Amélioration à apporter pour obtenir de bonnes performances sur des réseaux où le délai de propagation est important :
  - Soit à cause de l'augmentation des distances ;
  - Soit proportionnellement à la vitesse de transmission.
- Le paramètre est donné pas le produit :
  « bande-passante x délai de propagation »
  qui représente la capacité du réseau.
- Les réseaux dont le produit est élevé sont appelés LFN (*Long Fat Network*) -> *elephant*!

Elephant networks
Long Fat Networks (LFN)
TCP sur différents supports

Problèmes liés à ce type de réseau :

- Si la capacité du réseau est supérieure à la valeur maximale de la fenêtre (solution : option de l'en-tête IP multipliant la valeur de la fenêtre).
- La valeur initiale du champ séquence. Deux flux successifs (utilisant les mêmes numéros de port) mélangent des paquets (solution : PAWS *Protect Against Wrapped Sequence* (usage d'estampille temporel)).
- La détection de perte de paquet étant relativement longue (solution : l'option d'acquittement sélectif permet d'améliorer les performances).

Utilisation inefficace du lien...

- Pour une connection TCP (standard) avec des paquets de 1500 octets et un RTT (*Round-Trip Time*) de 100 ms sur un lien a 10 Gbps, il faudrait une fenêtre de congestion moyenne de 83.333 paquets et un taux de perte égal à au plus une congestion tous les 5 milliards de paquets (soit toutes les 1h40).
- Cela est principalement du a l'algorithme d'évitement de congestion (*congestion avoidance*) basé sur le principe “Additive Increase, Multiple Decrease” (AIMD).
- Une connection TCP réduit sa bande passante par **deux** lors de la détection d'une “congestion”, mais prend 1h40 pour utiliser à nouveau toute la bande passante (si aucune perte n'est détectée pendant ce temps la !).

”Apparently, standard TCP does not scale well in high bandwith, large roud-trip networks !”

Réseaux asymétriques

TCP sur différents supports

- Ex: ADSL. Le débit de réception est supérieur au débit d'émission. Globalement compatible avec l'usage d'internet (Ex: consultation de pages web, écoute d'un flux audio,...)
- Cette asymétrie à des conséquences sur le comportement de TCP et peut conduire à une sous-utilisation du sens rapide.
  - Un ACK émis par paquet reçu (risque de perte d'ACK et donc réduction du débit de la source).
- Solutions proposées :
  - Filtrage des ACKs.
  - Contrôle de congestion des ACKs (ACC) basé sur le bit ECN de l'en-tête IP.

Réseaux hertziens

TCP sur différents supports

- Ex: Réseaux satellitaires -> délai de transmission relativement grands, parfois asymétriques, avec un taux d'erreur important.
- Le contrôle de flux de TCP à été conçu pour considérer qu'une erreur provient d'une saturation d'un lien et non d'une erreur de transmission.
- La taille de *fenêtre de congestion* reste petite.
- Les SACKs corrigent un peu le problème (permet la retransmission des données avant que TCP ne passe dans l'état récupération *rapide*).
- L'usage de FEC (*Forwarding Error Correction*) permet de fiabiliser la liaison.
- Sujet de recherche toujours en cours... (rfc2760)

Contexte

- Lors de la conception de TCP (RFC 793, Sept. 1981) certaines hypothèses avaient été faites (exemples) :
  - Les applications qui n'ont pas besoin de livraison fiable et ordonné de paquets, n'ont pas besoin de mécanismes de contrôle de congestion.
  - Lorsqu'un paquet est perdu, TCP fait l'hypothèse que c'est du à une congestion (*i.e.* contention dans une ressource réseau), mais dans un résau *wireless* c'est peut-être du à une mauvaise réception du système récepteur.
- Les performances de TCP dépendent du produit : **bande passante** x **RTT** (*transfer rate x round-trip delay*).
- Lorsque le résultat de ce **produit** est **grand**, cela conduit à une **utilisation inefficace du lien**.

Propositions

- Pour résoudre ces problèmes, deux approches sont proposées :
  - modifier TCP et plus particulièrement l'algorithme AIMD,

    ou
  - utiliser des protocoles totalement nouveaux.

Les fonctions du service de transport et les caractéristiques des protocoles

- Les protocoles de la couche transport fournissent une communication de bout-en-bout entre deux hôtes (ou plus).
  - un **service** de transport propose un ensemble de fonctions pour les couches supérieures,
  - le **protocole** de transport détail comment la couche transport de l'émetteur et du récepteur coopèrent pour fournir ce service.

*Caractéristiques des **services** de transport*

CO_message	vs	**CO_byte**	vs	Connectionless
No loss	vs	Uncontrolled loss	vs	Controlled loss
No duplicate	vs	May be duplicate		
Ordered		vs Unordered		vs Partially ordered
Data-integrity	vs	No Data-integrity	vs	Partial data-integrity
Blocking	vs	Non Blocking		
Multicast	vs	**Unicast**		
Priority	vs	**No priority**		
Security	vs	**No security**		
Status reporting	vs	No status reporting		
Quality of Service	vs	No Quality of Service		

CO: connection oriented; CL: Connection less.

*Caractéristiques des **protocoles** de transport*

Connection oriented (CO)/ Connection less (CL)
Transaction oriented (one request/one response)
Connection oriented features:
- **Signaling in-band**/out of band
- Unidirectional/**bidirectional**
- Connection establishment: implicit/2 way/**3 way handshake**
- Connection termination: **gracefully**/ungracefully

Error control:
- **Error detection**
- Error reporting
- **Error recovery**
- Piggybacking
- Cumulative/Selective acknowledgement
- Retransmission strategy
- **Duplicate detection**
- Forward Error Correction

Flow/Congestion control:
- Flow control techniques: **sliding window**/**rate control**
- Flow control for congestion control: **fairness**, access control

Multiplexing/**Demultiplexing**

Segmentation/**Reassembling**

CO: connection oriented; CL: Connection less.

Critères de comparaison et objectifs (1/2)

- **Performance** : Les performances des nouveaux protocoles devront être comparable à TCP dans un réseau bas débit à petit RTT et bien meilleur que TCP dans les réseaux haut débit et grand RTT.
- **Contrôle de congestion** : mécanisme à inclure puisque le protocole devra être utiliser dans Internet (ou autre réseau publique *best-effort*). Cependant, le contrôle de congestion est-il nécessaire dans un réseau privé où la QoS peut-être garantie ?
- **TCP-"*friendly*"** : Un flux TCP-*friendly* réagit a une notification de congestion. Il n'empêche pas les autres flux de circuler.

Critères de comparaison et objectifs (2/2)

- **Equité intra-protocole** : Traite de l'équité parmi les connections utilisant le même protocole.
- **Déploiement** : besoin de modifier ou reconstruire le noyau de l'OS, ou juste ajouter une bibliothèque de niveau utilisateur qu'une application peut appeler directement.
- **Modèle analytique** : (1) Si il existe il permet de comparer les résultats expérimentaux aux prévisions. (2) Il permet aux développeurs d'identifier systématiquement les facteurs qui influence les perf. global et prédire le bénéfice d'une quelconque amélioration du protocole.
- **Scénario d'usage** : “One size fits all” c'est bien beau mais difficile à concevoir.

État de l'art
(non exhaustif !)

- **Variantes de TCP :**
  - HighSpeed TCP, Scalable TCP, Fast TCP, TCP Weswood+, SACK TCP, BIC and CUBIC, Compound TCP.
- **Protocoles basés sur UDP :**

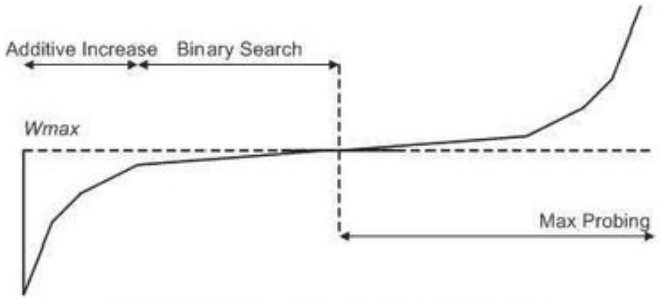
  UDP Lite, Reliable Blast UDP, Tsunami, UDT,...
- **Protocoles exigeant l'aide des routeurs :**

  XCP, CADPC/PTP,...
- **Autres :**

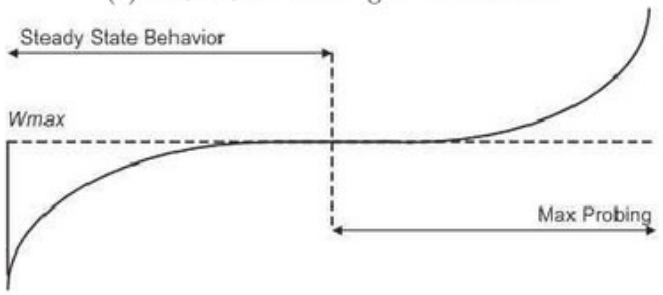
  SCTP, DCCP,...

Variantes de TCP :
BIC and CUBIC

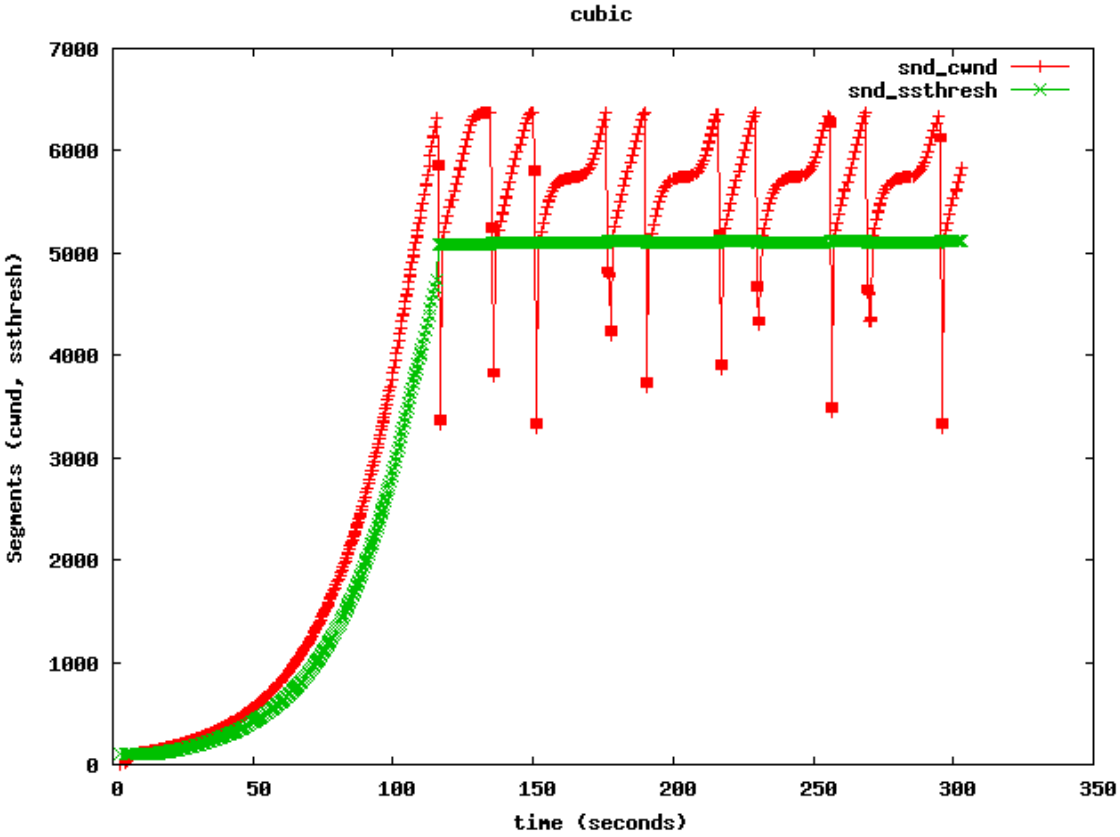
- Injong Ree, North Carolina Univ.
- Binary Search Increase --> target_win = (max_win – min_win) / 2
- Additive Increase --> cwnd = cwnd + (target_win – cwnd) / cwnd
- Slow Start --> cwnd+1, cwnd+2, cwnd+4... cwnd+Smax
- En cas de pertes --> cwnd = cwnd * (1 - Beta)



(a) BIC-TCP window growth function.



(b) CUBIC window growth function.



Variantes de TCP : Compound TCP (CTCP)

- Proposé par Microsoft (à partir de Windows Vista et WS 2008)
- Maintient deux fenêtres de congestion :
  - Une première qui augmente de façon linéaire mais décroît en cas de perte via un certain coefficient.
  - La seconde est liée au temps de réponse du destinataire.
- C'est donc une combinaison de New Reno (perte de paquets) et Vegas (adaptation préventive à la congestion).
- La taille de la fenêtre est la sommes des deux.
  - Si le RTT est petit, la fenêtre basée sur le délai augmente rapidement.
  - Si il y a perte de segments, la fenêtre basée sur la perte de paquet diminuera rapidement.
- Les fenêtres se compensent et permettent de tendre de façon presque constante vers une valeur de fenêtre effective proche de celle estimée.
- Bien adapté aux réseaux LFN.

Vocabulaire utilisé dans les protocoles de contrôle de congestion

- cwnd – congestion window
- RTT – Round trip Time
- RTO – Retransmission Timeout
- MSS – Maximum Segment Size
- DUPACK – ACK dupliqué
- ssthresh – Slow Start Threshold

Conclusion

- Face à la mise en place de plus en plus importante de liens longue distance à très haut débit (débit x RTT important), la mise au point de nouveaux protocoles capable d'exploiter ces liens est indispensable.
- Difficile de créer un protocole capable de répondre aux exigences de toutes les applications existantes et à venir...
- Beaucoup d'activités de recherche dans ce domaine suivi de trés prêt par tous les équipementiers réseaux.