Protocoles de transport UDP et TCP

Mohsen Souissi

ENSTA

Réseaux informatiques
et protocoles de communication

B1-1

06/10/2003

Sources & Remerciements

Merci à François Playe (CentralWeb) d'avoir:

- mis une présentation TCP/IP en ligne sur son site Web : http://www.centralweb.fr/download/
- autorisé les internautes à l'utiliser librement*

* A des fins non commerciales uniquement

Couche transport de la suite TCP/IP

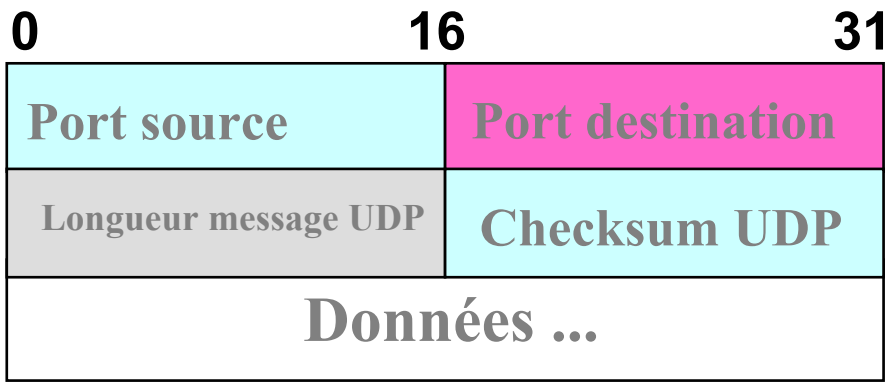
- Transport des messages entre processus applicatifs
- Identification du service : *les ports*
  - Les adresses IP désignent les machines communicantes
  - Un processus qui souhaite communiquer avec un autre processus doit savoir l'adresser
  - L'adressage de ce processus est effectué selon un concept abstrait indépendant du système d'exploitation des machines :
    - les processus sont créés et détruits dynamiquement sur les machines,
    - il faut identifier les destinations selon les services offerts, sans connaître les processus qui les mettent en oeuvre,
- L'émission d'un message se fait sur la base d'un port source et d'un port destination
- Les processus disposent d'une interface système leur permettant de spécifier un port ou d'y accéder (socket, TLI, ...).
- Deux modes de transport :
  - Mode non connecté : utilise le protocole UDP (User Datagram Protocol)
  - Mode connecté : utilise le protocole TCP (Transmission Control Protocol)

Notions de multiplexage / démultiplexage

- Une application qui offre un service doit avoir un numéro de port fixe connu par les clients potentiels
- Une application qui souhaite émettre un message doit connaître le port destination de l'application (serveur) distante
- Le numéro de port source est attribué par la machine locale
- Multiplexage
  - La couche transport peut gérer plusieurs messages simultanés en émission
- Démultiplexage
  - Lorsque la couche transport reçoit un message, elle vérifie que le port destination fait partie des ports actifs (associés à une application) à cet instant et le délivre à l'application associée à ce port
  - Si ce n'est pas le cas, elle émet un message ICMP *port unreachable* (qui sera acheminé vers la source), et détruit le message

UDP : User Datagram Protocol

- RFC 768 (STD 6) : mode de transport non connecté
- La livraison des messages non garantie : protocole non fiable
  - L'ordonnancement n'est pas garanti
- Les messages UDP sont également appelés des datagrammes UDP
- Ils contiennent deux parties : un en-tête UDP et les données UDP

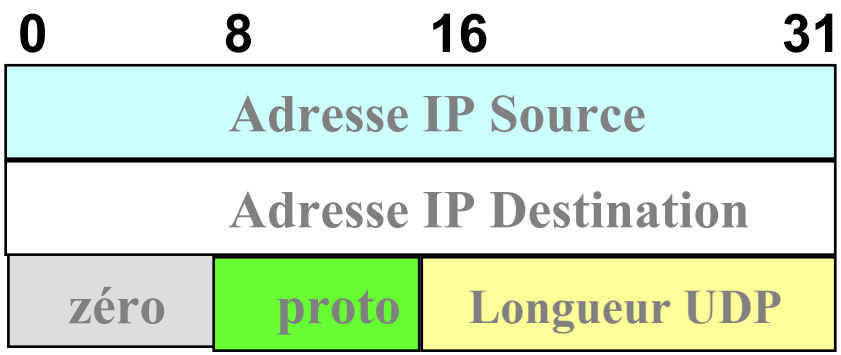


- **Port source** : facultatif (égal à zéro si non utilisé).
- **Port destination** : utilisé par UDP pour démultiplexer les datagrammes
- **Longueur du message** : exprimée en octets (8 au minimum)
- **Champ de contrôle** (**checksum**) : optionnel (0 si non utilisé)

UDP : pseudo en-tête

- Lorsqu'il est utilisé, le champ checksum couvre plus d'informations que celles contenues dans le datagramme UDP

- Calculé avec un pseudo-en-tête non transmis dans le datagramme



Format du pseudo en-tête

Le champ PROTO indique l'identificateur de protocole pour IP (17= UDP)

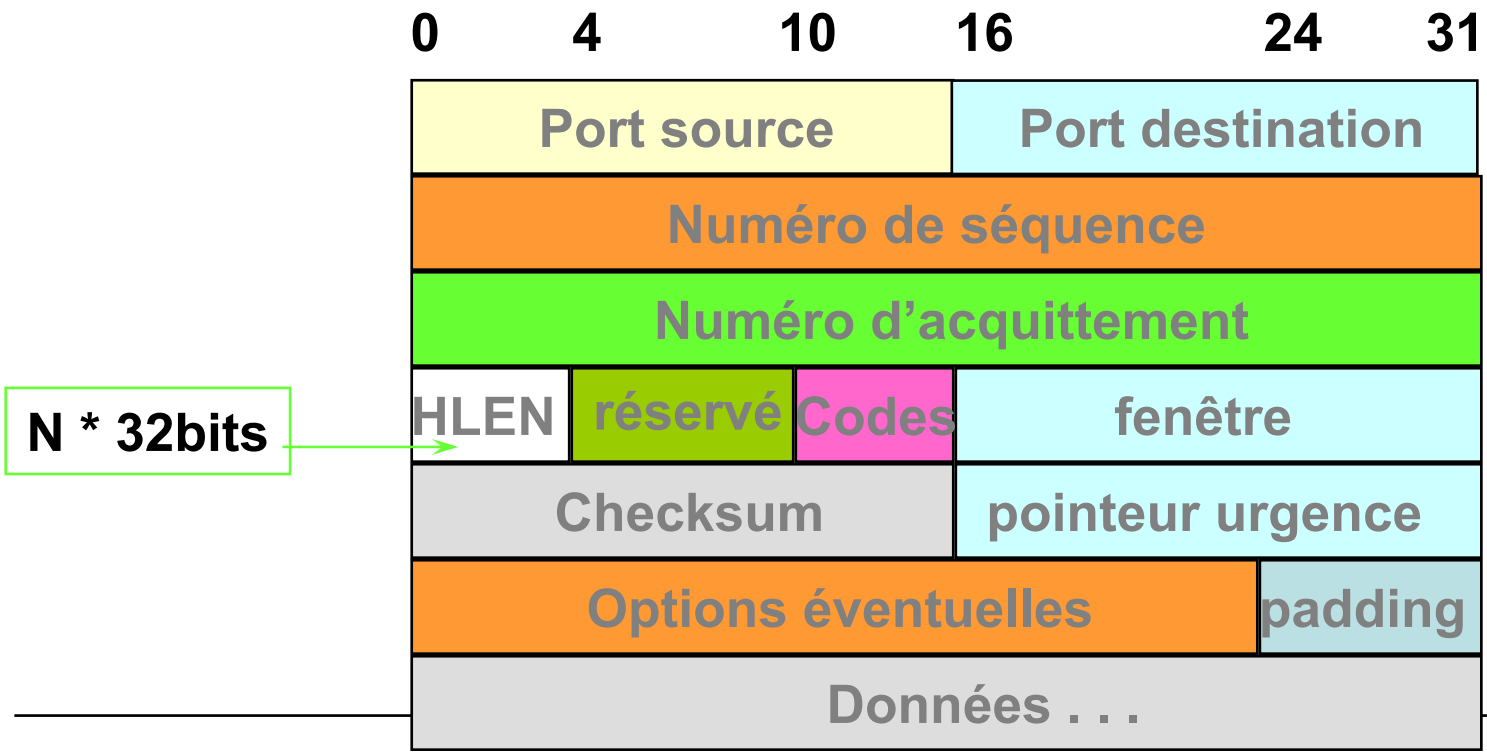
Le champ LONGUEUR UDP spécifie la longueur du datagramme UDP sans le pseudo-en-tête.

TCP : Transmission Control Protocol

- RFC 793 (STD 7) : Mode de transport connecté
- Transport fiable de la suite TCP/IP
- Garantie de non perte de messages ainsi que de l'ordonnancement
- Segmentation, contrôle de flux
  - Les données transmises à TCP constituent un flot d'octets de longueur variable
  - TCP divise ce flot de données en segments en utilisant un mécanisme de fenêtrage
  - Un segment est émis dans un datagramme IP

TCP : segments

- Segment : unité de transfert du protocole TCP
- Échangés pour :
  - ouvrir/fermer les connexions
  - transférer les données
  - émettre des acquittements



TCP : format du segment

- Numéro de séquence : le numéro de séquence du premier octet (NSP) de ce segment. Généralement à la suite d'octets O_1, O_2, ..., O_n (données du message) est associée la suite d'octets NSP, NSP+1, ..., NSP+n.

  Il existe deux exceptions à cette règle :

  - lorsque le bit SYN (voir CODE BITS) est positionné, le NSP représente cette donnée de contrôle et par conséquent la suite NSP, NSP+1, NSP+2, ..., NSP+n+1, correspond à la suite de données SYN, O_1, O_2, ..., O_n.

  - lorsque le bit FIN (voir CODE BITS) est positionné, le NSP+n représente cette donnée de contrôle et par conséquent la suite NSP, NSP+1, NSP+2, ..., NSP+n, associe la suite de données O_1, O_2, ..., O_n, FIN.

- Numéro d'acquittement : le prochain numéro de séquence NS attendu par l'émetteur de cet acquittement. Acquitte implicitement les octets NS-1, NS-2, etc

TCP : format du segment (2)

- Fenêtre: la quantité de données que l'émetteur de ce segment est capable de recevoir; ceci est mentionné dans chaque segment (données ou acquittement).

- **CODE BITS** : indique la nature du segment :

  - **SYN** : utilisé à l'initialisation de la connexion pour indiquer où la numérotation séquentielle commence. SYN occupe lui-même un numéro de séquence bien que ne figurant pas dans le champ de données. Le Numéro de séquence inscrit dans le datagramme (correspondant à SYN) est alors un *Initial Sequence Number* (ISN) produit par un générateur garantissant l'unicité de l'ISN sur le réseau (indispensable pour identifier les duplications).

  - **FIN** : utilisé lors de la libération de la connexion;
  - **URG** : le pointeur de données urgentes est valide (exemple : interrupt en remote login), les données sont émises sans délai, les données reçues sont remises sans délai.

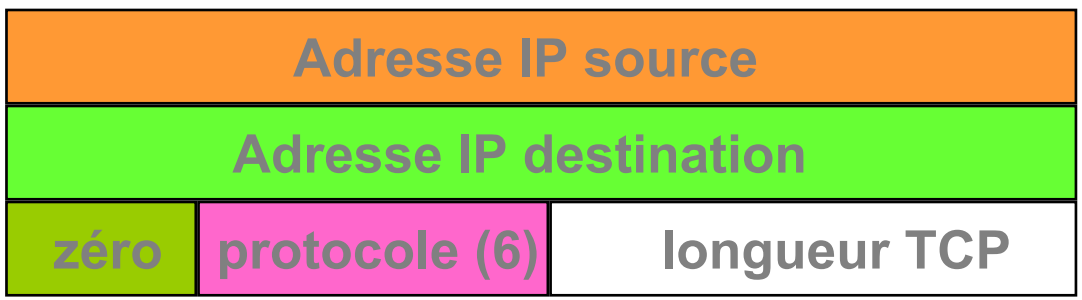
  - **ACK** : signifie que le segment contient un acquittement

TCP : format du segment (3)

- **PSH** : fonction push. Normalement, en émission, TCP reçoit les données depuis l'applicatif , les transforme en segments à sa guise puis transfère les segments sur le réseau; un récepteur TCP décodant le bit PSH, transmet à l'application réceptrice, les données correspondantes sans attendre plus de données de l'émetteur. Exemple : émulation terminal, pour envoyer chaque caractère entré au clavier

- **RST** : utilisé par une extrémité pour indiquer à l'autre extrémité qu'elle doit réinitialiser la connexion. Ceci est utilisé lorsque les extrémités sont désynchronisées

- CHECKSUM : calcul du champ de contrôle : utilise un pseudo-en-tête et s'applique à la totalité du segment obtenu (PROTO =6) :



TCP : format du segment (4)

OPTIONS

- L'option la plus utilisée est le **MSS** (*Maximum Segment Size*)
  - Permet d'annoncer la taille maximale des segments échangés. Cette option n'est présente que dans les segments d'initialisation de connexion (avec bit SYN).
  - TCP calcule une taille maximale de segment de manière à ce que le datagramme IP résultant corresponde au MTU (Maximum Transmit Unit) du réseau. La recommandation par défaut est de 536 octets.
  - La taille optimale du segment correspond au cas où le datagramme IP n'est pas fragmenté

TCP : la connexion

- Une connexion de type circuit virtuel est établie avant que les données ne soient échangées : appel + négociation + transfert
- 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))

- Une extrémité de connexion peut être partagée par plusieurs autres extrémités de connexions (multi-instanciation)

- La mise en oeuvre de la connexion se fait en deux étapes :
  - une application (extrémité) effectue une ouverture passive en indiquant qu'elle accepte des connexions entrantes,
  - une autre application (extrémité) effectue une ouverture active pour demander l'établissement de la connexion

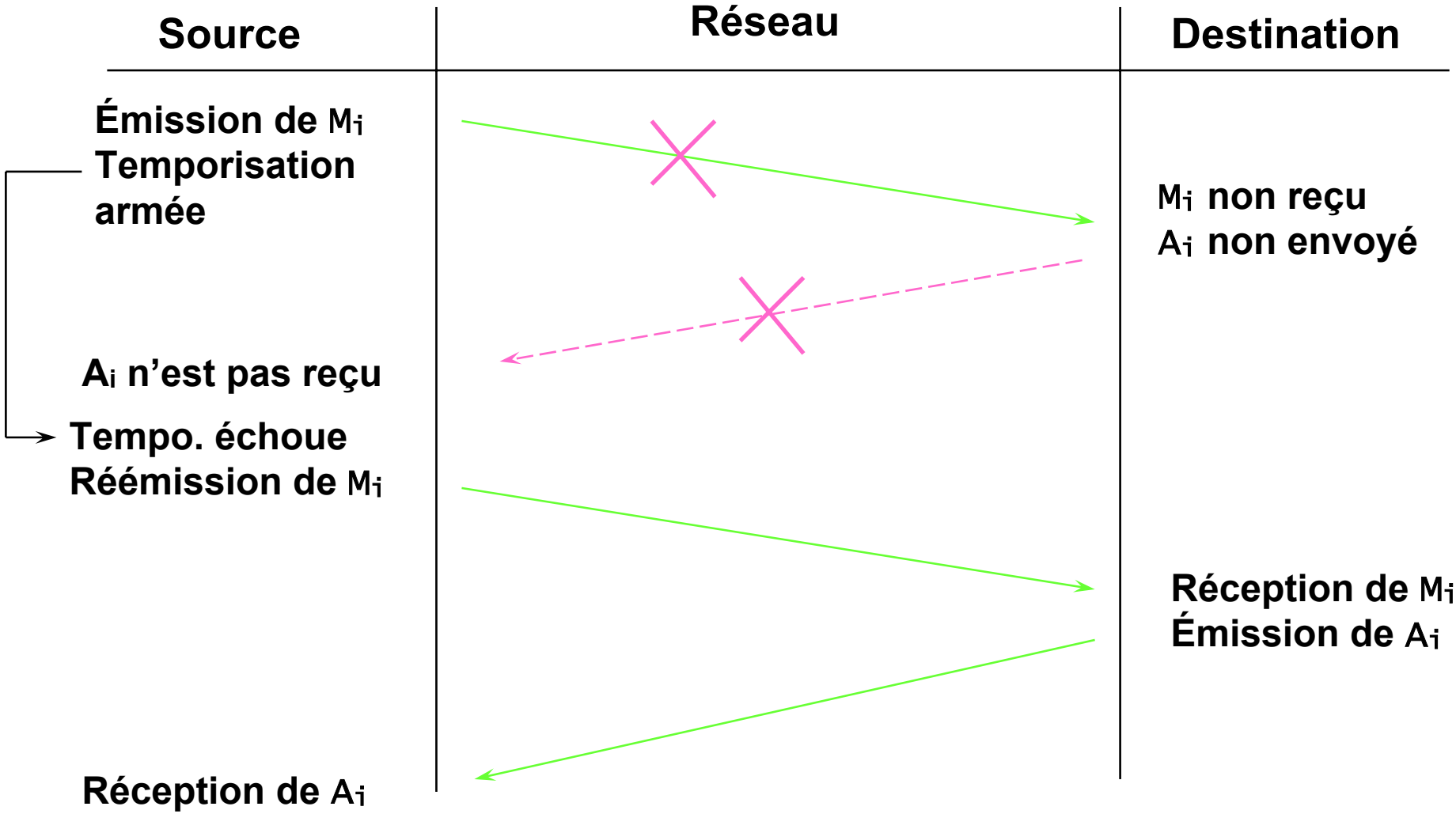
TCP : acquittements

- Contrairement à UDP, TCP garantit l'arrivée des messages, c'est-à-dire qu'en cas de perte, les deux extrémités sont prévenues

- Ce concept repose sur les techniques d'acquittement de message : lorsqu'une source S émet un message M_i vers une destination D, S attend un acquittement A_i de D avant d'émettre le message suivant M_{i+1}

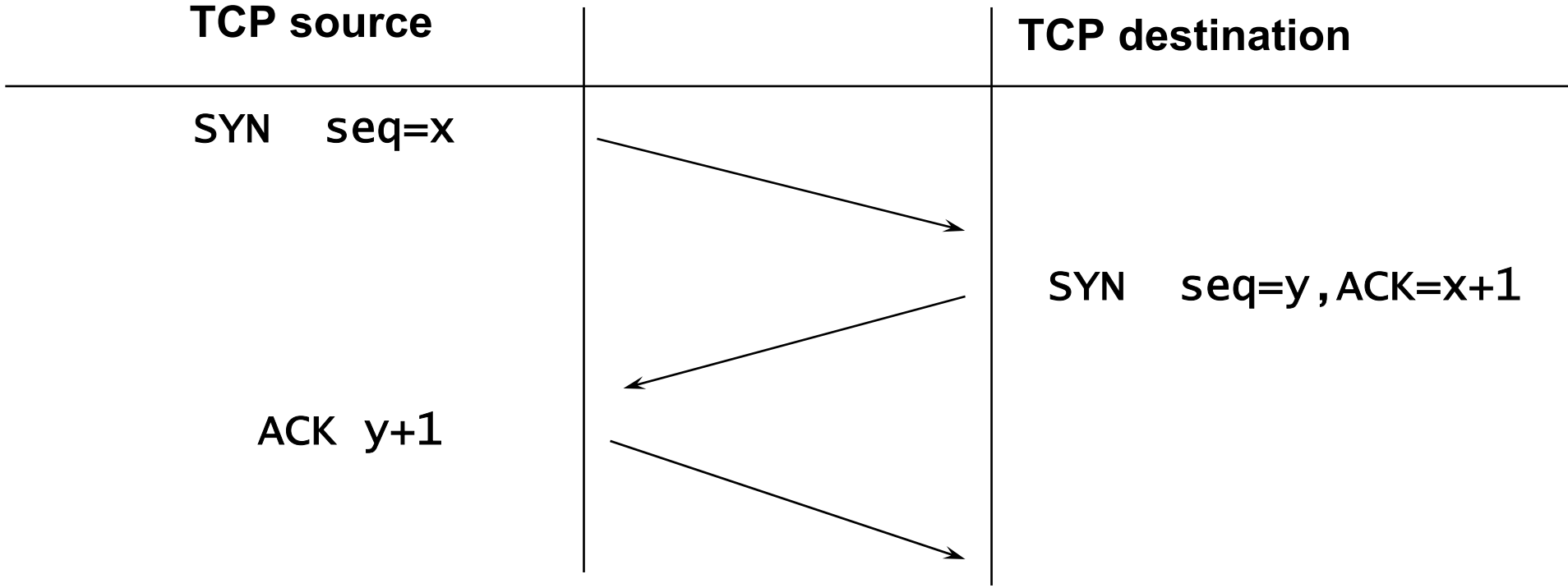
- Si l'acquittement A_i ne parvient pas à S, S considère au bout d'un certain temps que le message est perdu et réémet M_i

TCP : acquittements (2)



TCP : établissement de la connexion

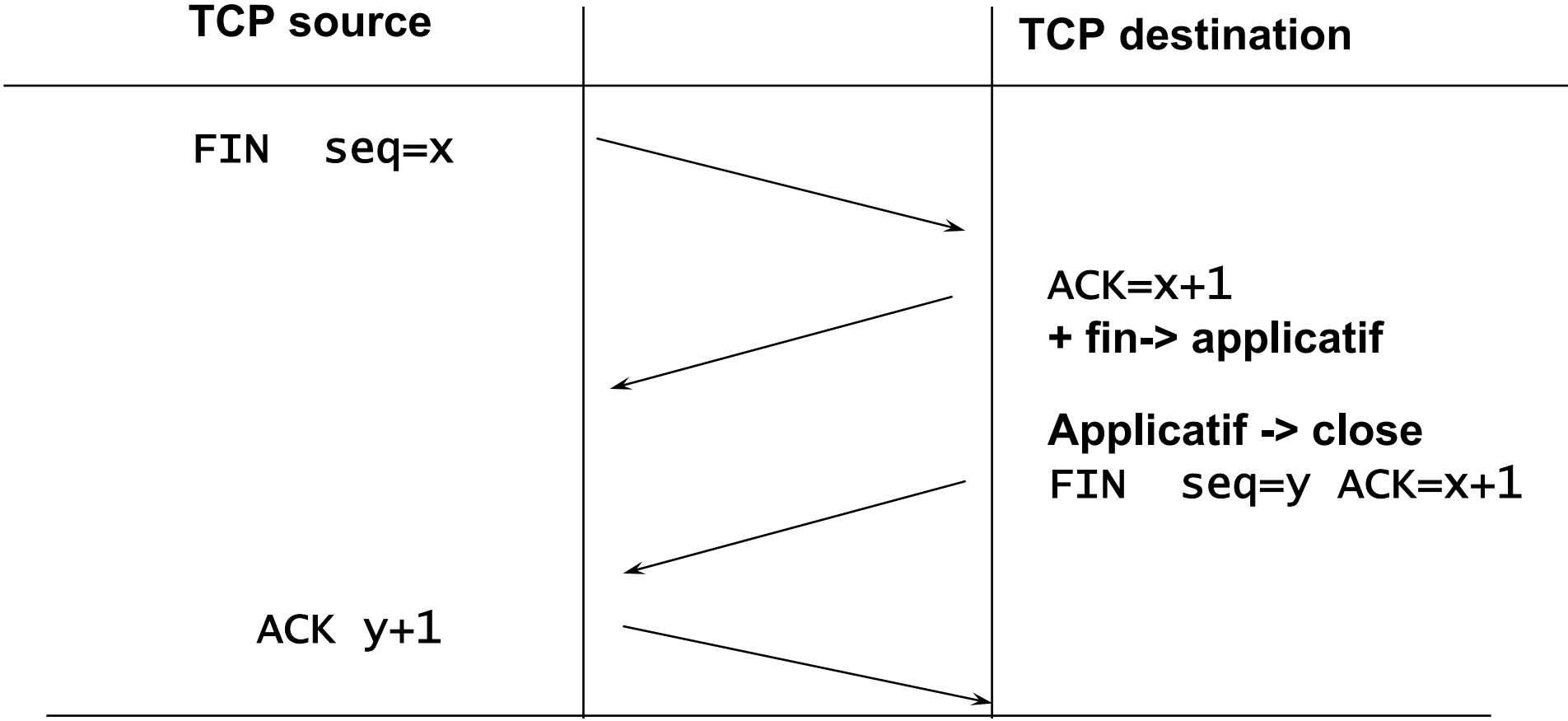
Une connexion TCP est établie en trois temps (« ***three-way handshake*** ») de manière à assurer la synchronisation nécessaire entre les extrémités :



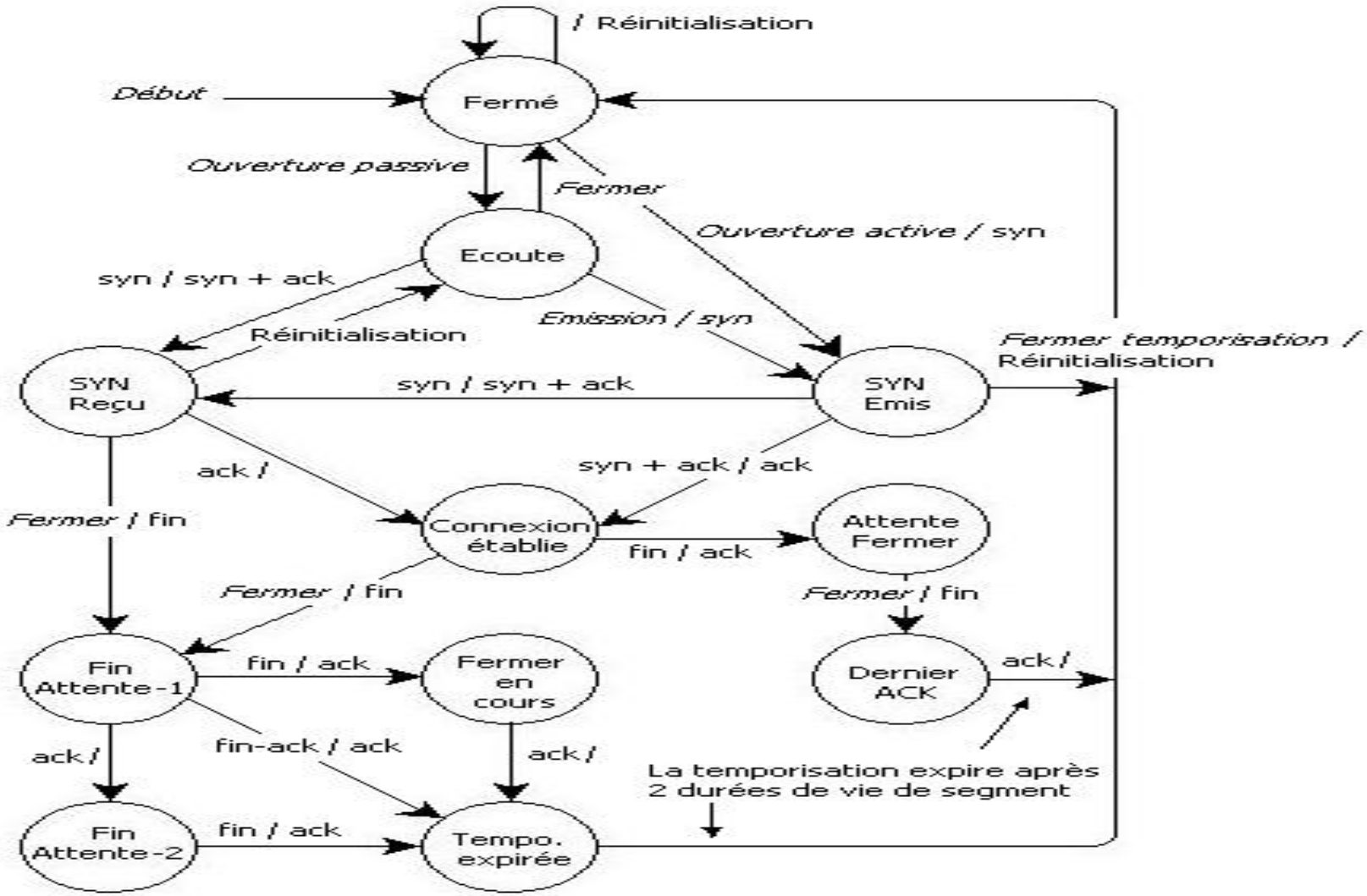
Ce schéma fonctionne lorsque les deux extrémités effectuent une demande d'établissement simultanément. TCP ignore toute demande de connexion, si cette connexion est déjà établie.

TCP : déconnexion

- Une connexion TCP est libérée en 4 temps



Automate TCP

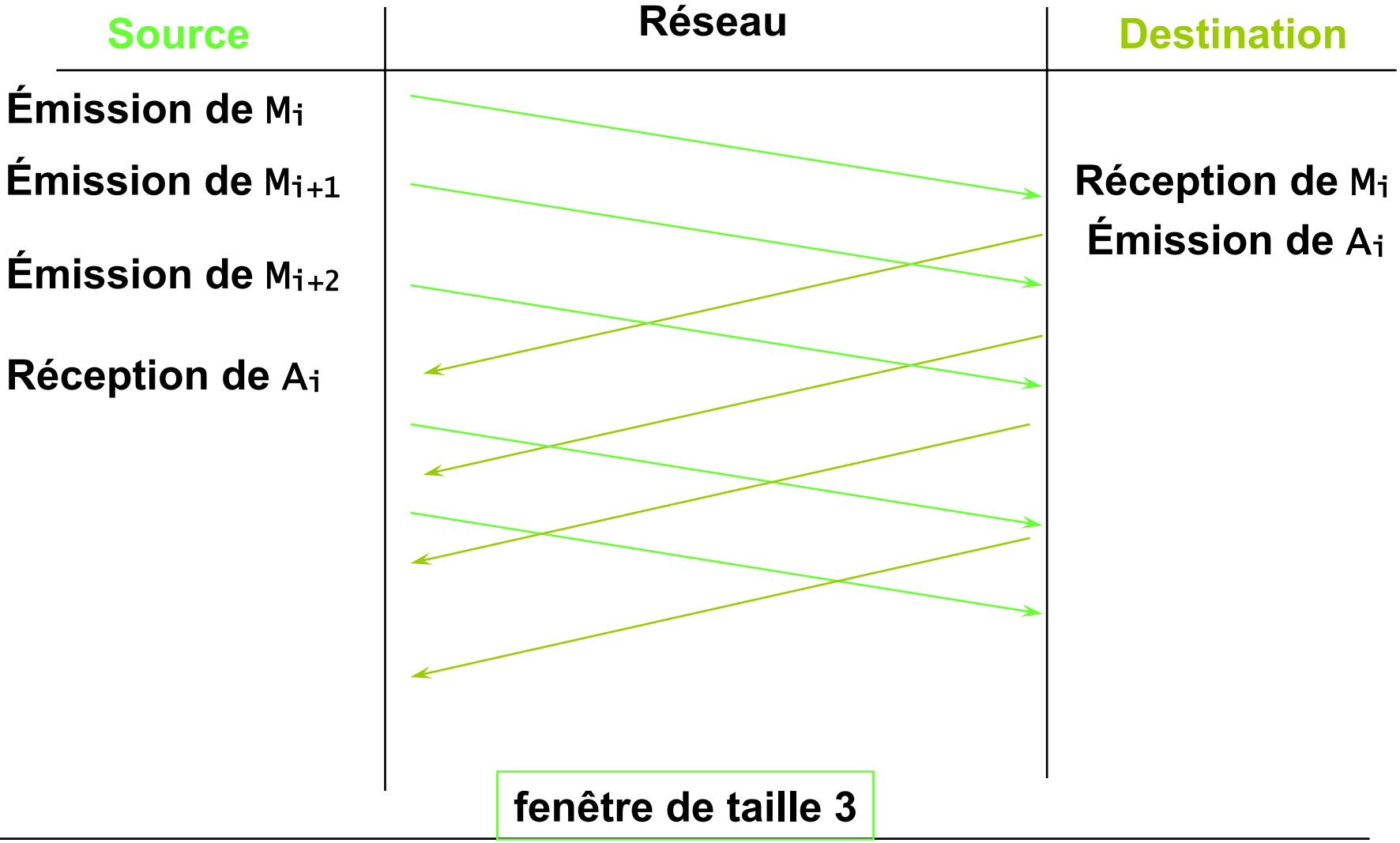


TCP : fenêtrage

- La technique d'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 : fenêtrage (2)



TCP : technique de fenêtrage

- Fenêtre 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; il repose sur :
  - la numérotation séquentielle des octets de données,
  - la gestion de trois pointeurs par fenêtre :

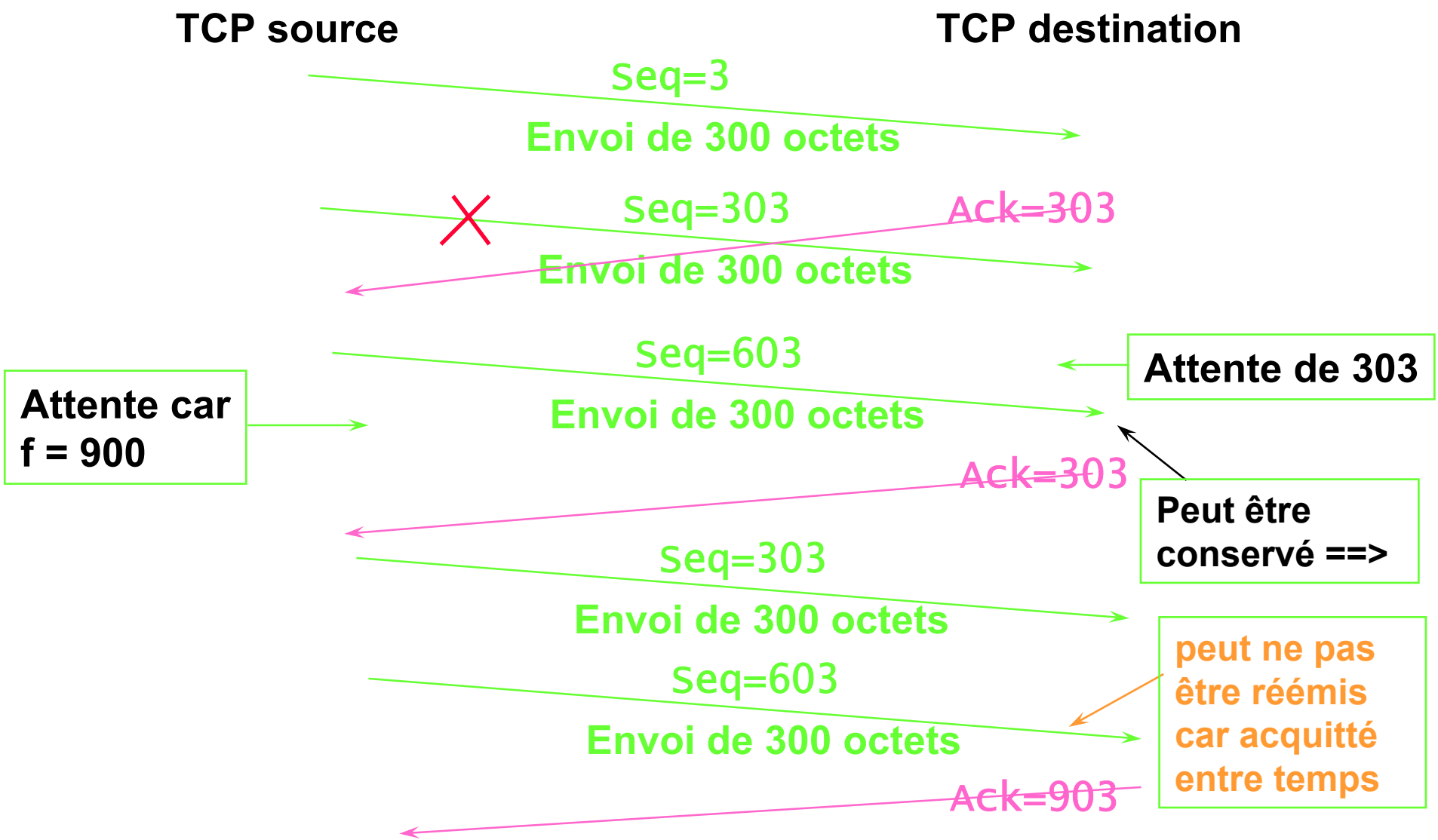
Acquittements TCP : détails

- Le mécanisme d'acquittement de TCP est cumulatif :
  - Il indique le numéro de séquence du prochain octet attendu : tous les octets précédents cumulés sont implicitement acquittés
  - Si un segment a un numéro de séquence supérieur au numéro de séquence attendu (bien que dans la fenêtre), le segment est conservé mais l'acquittement référence toujours le numéro de séquence attendu

- Pour tout segment émis, TCP s'attend à recevoir un acquittement
  - Si le segment n'est pas acquitté, le segment est considéré comme perdu et TCP le retransmet.
  - Or un réseau d'interconnexion offre des temps de transit variables nécessitant le réglage des temporisations ;
  - TCP gère des temporisations variables pour chaque connexion en utilisant un algorithme de retransmission adaptative

- Exemple : Fenêtre=900, Segment=300

Acquittements TCP : détails



TCP : gestion des temporisations (1)

- Algorithme de retransmission adaptative
  - enregistre la date d'émission d'un segment,
  - enregistre la date de réception de l'acquittement correspondant,
  - calcule l'échantillon de temps de boucle A/R écoulé,
  - détermine le temps de boucle moyen RTT (Round Trip Time) :

$$RTT = (a * anc\_RTT) + ((1-a) * NOU\_RTT))$$

avec $0 \leq a < 1$

a proche de 1 : RTT insensible aux variations brèves,

a proche de 0 : RTT très sensible aux variations rapides,

  - calcule la valeur du temporisateur en fonction de RTT.
  - les premières implémentations de TCP ont choisi un coefficient constant B pour déterminer cette valeur :
    Temporisation = B * RTT avec B >1 (généralement B=2).

TCP : gestion des temporisations (2)

- Algorithme de Karn
  - Appliqué pour affiner la mesure du RTT : (tient compte des retransmissions)
  - Repose sur les constatations suivantes :
    - en cas de retransmission d'un segment, l'émetteur ne peut savoir si l'acquittement s'adresse au segment initial ou retransmis (ambiguïté des acquittements), =>l'échantillon RTT ne peut donc être calculé correctement,
  - ➔ TCP ne doit pas mettre à jour le RTT pour les segments retransmis
  - Combine les retransmissions avec l'augmentation des temporisations associées (*timer backoff*):
    - une valeur initiale de temporisation est calculée
    - si une retransmission est effectuée, la temporisation est augmentée (généralement le double de la précédente, jusqu'à une valeur plafond)
  - Cet algorithme fonctionne bien même avec des réseaux qui perdent des paquets.

TCP : limitation du trafic

- Retard de l'acquittement
  - Réunir des acquittements pour améliorer le débit utile (exemple : terminal virtuel)
  - Ne doit pas être trop important (pour ne pas faire croire à l'émetteur qu'il y a une perte)
  - Ne concerne que les données reçues en séquence
- Algorithme de Nagle
  - Appliqué aux réseaux grande distance où
    - Les petits paquets chargent les routeurs et le réseau est plus lent
    - Cercle vicieux: pertes → retransmission → pertes → …
  - Une connexion TCP peut avoir uniquement un segment de petite taille non acquitté : aucun segment supplémentaire de petite taille ne peut être émis avant la réception de l'acquittement
  - L'émission des données est différée et regroupée dans des segments de taille plus importante de taille
  - L'algorithme doit être désactivé pour certains trafics comme X-Window

TCP : la congestion (1)

- Gestion de la congestion
  - TCP gère le contrôle de flux de bout en bout mais également les problèmes de congestion liés à l'interconnexion.
  - La congestion correspond à la saturation de noeud(s) dans le réseau provoquant des délais d'acheminement de datagrammes jusqu'à leur pertes éventuelles.
  - Les extrémités ignorent tout de la congestion sauf les délais. Habituellement, les protocoles retransmettent les segments ce qui aggrave encore la situation.
  - Dans la technologie TCP/IP, TCP participe à la gestion de la congestion en diminuant le débit lorsque les délais s'allongent :

TCP : la congestion (2)

- TCP maintient une ***fenêtre de congestion***
- TCP applique la fenêtre d'émission suivante :
  - fenêtre_autorisée = min (fenêtre_récepteur, fenêtre_congestion)

- Dans une situation de non congestion :
  - fenêtre_récepteur = fenêtre_congestion

- En cas de congestion, TCP applique une diminution dichotomique :
  - à chaque segment perdu, la fenêtre de congestion est divisée par 2 (minimum 1 segment)
  - la temporisation de retransmission est augmentée exponentiellement.

TCP : congestion / « slow start »

- Si la congestion disparaît, TCP définit une fenêtre de congestion égale à 1 segment et l'incrémente de 1 à chaque acquittement reçu

  - ❖ démarrage lent et progressif (***slow start***)

Fenêtre_congestion = 1	émission du 1er segment, attente acquittement, réception acquittement,
Fenêtre_congestion = 2	émission des 2 segments, attente des acquittements, réception des 2 acquittements,
Fenêtre_congestion = 4	émission des 4 segments, …

Log2 (N) itérations pour envoyer N segments. Lorsque la fenêtre atteint une fois et demie sa taille initiale, l'incrément est limité à 1 pour tous les segments acquittés de la fenêtre (croissance linéaire)

TCP : quelques ports standards

#port	Mot-clé	Description
20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
22	SSH	Secure Shell
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
53	DOMAIN	Domain Name System (DNS)
80	HTTP	WWW
110	POP3	Post Office Protocol – V3
111	SUNRPC	SUN Remote Procedure Call
119	NNTP	USENET News Transfer Protocol
123	NTP	Network Time Protocol
...		

Références

- UDP : RFC 768
- TCP : RFC 793
- TCP Extensions for High Performance : RFC 1323
- TCP/IP Illustrated, Volumes 1 & 2, *W. Richard Stevens*
- Réseaux locaux et Internet, 2ème édition, *Laurent Toutain*
- TCP/IP : Architecture, protocoles, applications, *Douglas Comer*
- `http://www.centralweb.fr/download/`