CORRIGÉ

Faculté des Sciences
Aix*Marseille Université

Applications Réseaux : TD 3

Fonctionnement de TCP

L3 Informatique – 2019/2020

1. Lorsque TCP envoie un [SYN, SeqNum = x] ou un [FIN, SeqNum = x], le segment ACK correspondant porte AckNum = x+1. En d'autres termes, les segments SYN et FIN consomment une unité de l'espace de numérotation. Est-ce vraiment nécessaire ?

6 Lorsque TCP envoie un [SYN, SeqNum = x] ou un [FIN, SeqNum = x], le segment ACK correspondant porte AckNum = x+1. En d'autres termes, les segments SYN et FIN consomment une unité de l'espace de numérotation.

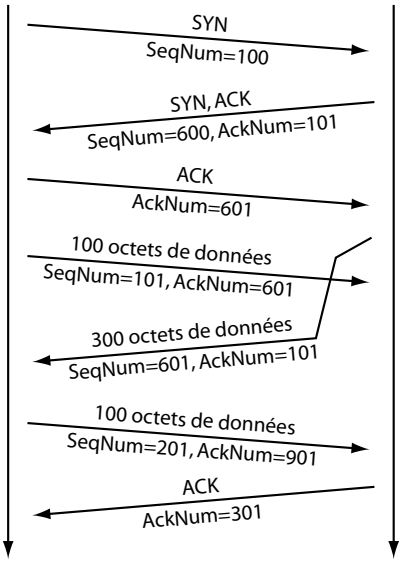
Il est nécessaire d'incrémenter le AckNum d'un **segment FIN**, de façon à ce que l'émetteur du FIN puisse déterminer si son FIN a bien été reçu, et non pas seulement les données précédentes.

Pour un **segment SYN**, tout ACK pour des données postérieures au SYN incrémente AckNum : un tel ACK va donc acquitter implicitement le SYN (puisque des données ne peuvent pas être acquittées tant que la connexion n'est pas établie). Par conséquent, le fait d'avoir [ACK, AckNum = x+1] au lieu de [ACK, AckNum = x] relève plus d'une convention dans un souci d'homogénéité que d'une nécessité protocolaire.

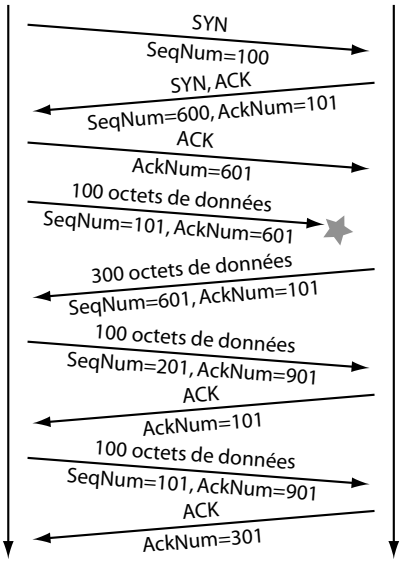
2. On considère une connexion TCP entre deux applications distantes A et B. A doit envoyer à B deux segments de 100 octets de données chacun et B doit envoyer à A un segment de 300 octets de données. Représenter le chronogramme d'un échange entre A et B, sachant que A est à l'origine de l'établissement et que la référence initiale de A est égale à 100 et celle de B à 600. Que se passe-t-il si le premier segment de données de A se perd ?

7 On considère une connexion TCP entre deux applications distantes A et B. A doit envoyer à B deux segments de 100 octets de données chacun et B doit envoyer à A un segment de 300 octets de données. On sait que A est à l'origine de l'établissement et que la référence initiale de A est égale à 100 et celle de B à 600.

Voici le chronogramme d'un échange entre les deux hôtes :



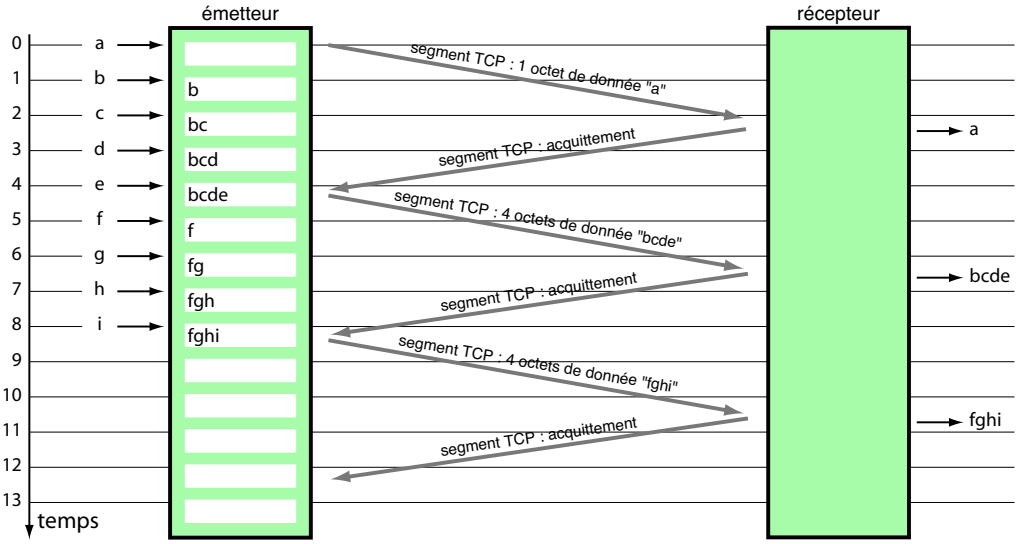
Voici ce qui peut se passer si le premier segment de données de A se perd :



3. Supposons que les caractères "abcdefghi" soient envoyés, à raison d'une lettre par seconde, sur une connexion TCP dont le RTT est de 4,1 s.
   — Rappelez ce qu'est l'algorithme de Nagle.
   — Représentez le chronogramme des émissions de segments.
   — Qu'observe l'utilisateur si les caractères sont tapés sur une connexion Telnet full-duplex ?

9 Supposons que les caractères « abcdefghi » soient envoyées, à raison d'une lettre par seconde, sur une connexion TCP dont le RTT est de 4,1 s.

1. L'algorithme de Nagle permet une amélioration de l'efficacité proposée pour les applications produisant des données octet par octet telles que sont les applications interactives. Il est intégrée dans la plupart des implémentations TCP et impose à l'émetteur de stocker les octets de données passées par l'application source (même s'il y a des push) avant de constituer un segment, tant qu'il n'a pas MSS (Maximum Segment Size) octets ou tant qu'il n'a pas reçu le ACK du segment précédent.
2. Voici le chronogramme des émissions de segments :



3. L'écho des caractères ne se fait que toutes les 4 secondes, et par bloc de 4 caractères. Un tel comportement est courant sur les connexions Telnet, même pour celles ayant des RTT plus modestes.

4. Pourquoi un émetteur attend-il la réception de 3 ACK dupliqués (4 ACK identiques sans réception intermédiaire d'aucun autre segment) avant de retransmettre le segment qu'acquittent ces 4 ACK et ce sans attendre l'expiration du temporisateur de retransmission associé ? La détection d'une perte devrait provoquer le passage immédiat de l'émetteur dans l'état Slow Start. Pourquoi la réception de plus de trois acquittements dupliqués ne provoque-t-elle pas ce passage ?

11 Un récepteur émet immédiatement un acquittement (dupliqué) lorsqu'un segment déséquencé est reçu. Cet ack sert à informer l'émetteur de la réception du paquet déséquencé et du numéro de séquence attendu. Sur réception d'un ack dupliqué, l'émetteur n'est toutefois pas capable de déterminer de manière sûre si cette duplication est due à la perte d'un paquet, à un déséquencement ou à une duplication dans le réseau.

Si l'émetteur reçoit plus de trois acquittements, il est extrêmement probable que la cause est une perte de paquet et la retransmission devient nécessaire. Dans les autres cas, l'émetteur considère qu'un simple déséquencement est à l'origine de l'ack dupliqué. La réception d'au moins trois acks dupliqués indique qu'un seul paquet s'est perdu. L'émetteur considère une perte due à une erreur sur les données, il retransmet le segment manquant. Il est donc inutile d'appliquer l'algorithme de contrôle de congestion par une diminution brutale de la taille de la fenêtre de l'émetteur, d'autant plus qu'un certain nombre de paquets sont encore en transit sur le réseau. Ces deux principes correspondent aux algorithmes de *retransmission rapide* et de *récupération rapide*. À la réception du troisième ack dupliqué, l'émetteur part du principe qu'un (seul) paquet s'est perdu. Il applique alors le *Fast Recovery* (et non le *Slow Start* comme dans le cas d'une expiration du timer de retransmission) qui régit la transmission de nouvelles données jusqu'à ce qu'un ack non dupliqué arrive. Il y a deux raisons à ce choix :

- le récepteur ne peut générer un acquittement dupliqué à la réception d'un segment uniquement si ce dernier a quitté le réseau et se trouve dans les tampons de réception du récepteur (et ne consomme donc pas de ressources réseau).
- il y a préservation du *ack clocking* (principe de conservation des paquets) ce qui permet à l'émetteur de continuer à transmettre de nouveaux segments.

5. Supposons que la fenêtre de congestion de TCP soit égale à 18Ko (MSS=1Ko) et qu'un temporisateur expire. Quelle sera la taille maximale de la fenêtre de congestion quand quatre transmissions en rafale auront été acquittées normalement ?

13 Supposons que la fenêtre de congestion de TCP soit égale à 18 Ko (MSS = 1 Ko) et qu'un temporisateur expire. Pour que la taille soit maximale, il faut que les rafales successives remplissent la fenêtre de congestion, sinon la taille de la fenêtre augmentera plus lentement que par doublements successifs, ce qui ne permettra pas d'atteindre la taille maximale.

Après l'expiration, la fenêtre de congestion vaut la taille de un MSS (1 Ko) et le seuil de congestion vaut la moitié (9 MSS) de la taille de la fenêtre de congestion avant le crash.

temps	reçu	taille de la fenêtre	émis
0	–	1	1 MSS
1 RTT	1 ACK	$1+1=2$	2 MSS
2 RTT	2 ACK	$2+2=4$	4 MSS
3 RTT	4 ACK	$4+4=8$	8 MSS
4 RTT	8 ACK	$\min(8+8,9)$= 9	9 MSS

La taille maximale de la fenêtre après 4 rafales de segments vaut 9 Ko.