Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Protocoles & Prog. Réseau

Corrections TD n°4

TCP vs UDP

TCP

1 – Est-ce que la fragmentation et le ré-assemblage des datagrammes IP concernent TCP ?
Est-ce que cela veut dire que TCP n'a pas à se préoccuper de l'ordre d'arrivée des données ?

- ⋆ *TCP tient compte de la taille d'un datagramme IP, donc indirectement de la MTU, pour définir un MSS, «Maximum Segment Size» adapté à la taille de transport de la technologie réseau ;*
- ⋆ *la fragmentation du datagramme IP survient quant un datagramme doit traverser un réseau dont la MTU est inférieure. Cette fragmentation est réalisée, en IPv4, par les routeurs.*
  *Elle ne concerne que le contenu d'un **seul datagramme**, soit un **seul segment**.*
- ⋆ *la fragmentation ne concerne pas TCP qui n'en est même pas informé (il reçoit le contenu du datagramme après défragmentation seulement, c-à-d le segment contenu dedans)*
  ⟹ TCP doit réaliser son propre traitement de réordonnancement et de recomposition du flux d'octets à partir de ses segments.

2 – La connexion TCP :

a. Pourquoi le schéma d'établissement d'une connexion TCP correspond à trois échanges ?
Ce schéma correspond à l'établissement de deux $\frac{1}{2}$ connexions :
- ◇ *une du client ⟶ serveur (pour empêcher le client d'envoyer des données sans autorisation du serveur , et synchroniser les numéros de séquence pour éviter la confusion avec des paquets d'une communication précédente) ;*
- ◇ *l'autre du serveur ⟶ client (pour synchroniser les numéros de séquence et empêcher la confusion avec des paquets d'une communication précédente).*

Pour l'établissement de chaque $\frac{1}{2}$ connexion, il faut un segment de demande (SYN) et un segment d'acceptation (ACK), soit 4 segments.
L'optimisation en 3 segments consiste à combiner le segment du ACK de la première $\frac{1}{2}$ connexion avec le segment de demande de la seconde $\frac{1}{2}$ connexion (SYN/ACK).

b. Pourquoi le protocole TCP structure les échanges de données en segment alors qu'il rend un service de flux d'octets ?
Parce que TCP est construit par dessus un réseau en «mode datagramme», il doit donc découper le flux de données en morceaux ou segment, un par datagramme, et s'assurer
- ◇ *du réordonnancement de ces segments lors de la réception ;*
- ◇ *du contrôle d'erreur résultant de la perte possible de datagramme (réémission et acquittement).*

c. Pourquoi ne pas faire commencer le «ISN», *Initial Sequence Number* à 0 ?
- ◇ *pour des raisons de contrôle d'erreur : imaginons qu'un client effectue de nombreuses connexions successives avec le même serveur :*
  - ⋆ *chacune de ces connexions possède le même $TSAP_{source}$ et $TSAP_{destination}$;*
  - ⋆ *si l'ISN commence toujours à zéro, alors un segment «perdu» qui arriverait finalement au moment d'une nouvelle connexion pourrait être par erreur intégrer dans la nouvelle connexion (si son numéro de séquence appartient à la fenêtre du récepteur) ;*
- ◇ *pour des raisons de sécurité : un attaquant ne doit pas pouvoir prévoir l'ISN qu'il ne peut pas connaître par «sniffing» (il est situé ni dans le réseau source, ni dans celui destination) pour essayer d'intégrer dans une connexion TCP un segment falsifié ou «forgé».*

3 – Les numéros de séquence du protocole TCP font référence au nombre d'octets transmis et non aux numéros des paquets incrémentés de 1 pour chaque paquet envoyé.

a. pourquoi utilise-t-on ce type de notation pour définir les numéros de séquence ?
Parce que la taille d'un segment est variable.

b. on suppose que l'on utilise TCP sur un lien de débit 1Gbps et que l'émetteur n'est jamais bloqué par la fenêtre d'émission (servant à empêcher la congestion).
Combien faut-il de temps pour utiliser l'ensemble complet des numéros de séquence ?
Le numéro de séquence est indiqué sur 32 bits, il représente au plus 2^{32} *octets transmis.*
   - *Une connexion 1Gbps transfert* 10^9 *bits par seconde, il faut donc : temps* $= \frac{2^{32}*8}{10^9} = 34,36s$ *pour épuiser le numéro de séquence si on ne tient pas compte des entêtes présentes pour le transport des données TCP.*
   - *Si on veut tenir compte de la taille des en-têtes :*
     - *trame : 1518 octets (sans tenir compte du préambule et du SFD qui font 8 octets) ;*
     - $2 * 6 + 2 + 4 = 18$ *pour deux adresses MACs, l'Ethertype et le FCS, soient 1500 octets pour le MTU ;*
     - $20 + 20 = 40$ *octets au minimum pour les deux en-têtes (IP et TCP), soit une taille maximale de segment, MSS, de 1460 octets ;*
     - $\Longrightarrow$ *pour chaque segment envoyé on a une occupation réseau de* $8 + 18 + 40 = 66$ *octets.*

     Ce qui donne : $\frac{(\frac{2^{32}}{1460}*66+2^{32})*8}{10^9} = 35,91s$

UDP

4 – On veut utiliser le protocole UDP dans le modèle « Client/Serveur ».

a. Quel est la nature des échanges entre le client et le serveur ?
On échange des datagrammes UDP, c-à-d un datagramme IP contenant un port source et destination ainsi qu'une taille et un cheksum.

b. Comment ces échanges sont-ils identifiés sur le client et le serveur ?
Par le $TSAP_{client}$ *pour le serveur et par le* $TSAP_{serveur}$ *pour le client.*

c. Comment le serveur peut-il être trouvé par le client et comment peut-il répondre au client ? *Il faut que son TSAP soit connu du client : @IP donnée et port standardisé par exemple.*
Lorsqu'il reçoit un datagramme UDP, ce datagramme contient le port source utilisé par le client et le système peut lui donner l'@IP source, il peut ainsi répondre en renvoyant un datagramme UDP en direction du $TSAP_{client}$ *qu'il a appris.*

Dans le cas d'un protocole « Question/Réponse », où une question dans un paquet UDP est envoyée du client vers le serveur, et une réponse est envoyée du serveur vers le client dans un seul paquet UDP :

d. Est-ce que toutes les réponses arrivent dans le même ordre que les questions ont été envoyées ?
Non, car le réseau est en « mode datagramme » ; il peut même y avoir des datagrammes perdus.

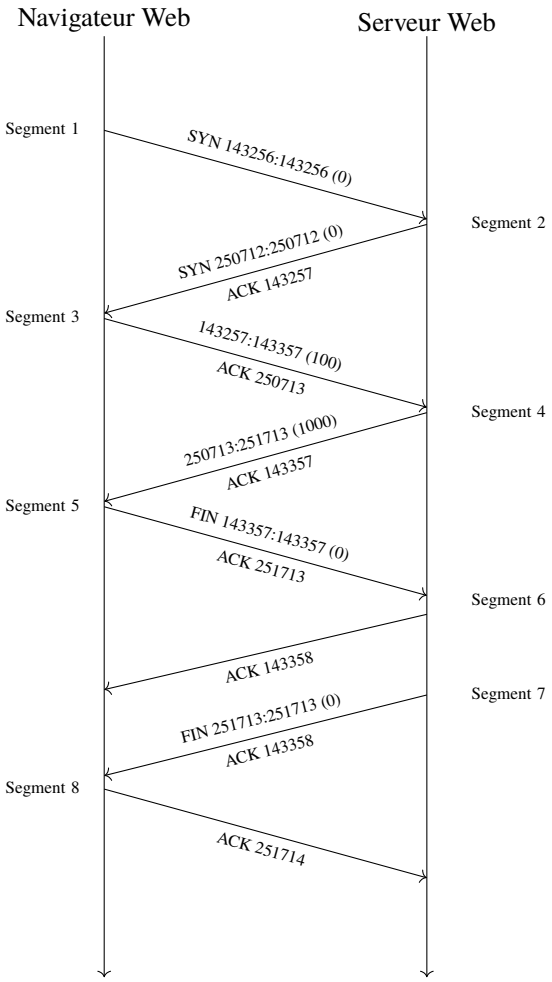
e. Comment ne pas mélanger les réponses ?
   - *on peut reprendre la question dans la réponse ;*
   - *on peut ajouter une « étiquette », c-à-d un numéro géré par le client, dans la question et la reprendre dans la réponse.*

f. Comment tenir compte des questions et réponses perdues ?
Si une réponse ne vient pas, alors le client renvoie sa question (avec éventuellement une étiquette différente) $\Longrightarrow$ *la réponse est une forme d'acquittement de la bonne réception de la question.*

g. Si l'on veut que le serveur gère simultanément plusieurs clients est-ce que cela fonctionne toujours ?
Oui cela fonctionne car chaque étiquette est gérée par le client associé, indépendamment du serveur qui ne fait que les réutiliser dans ces réponses et chaque client est identifié par un TSAP différent.
De plus le serveur ne mémorise aucune des étiquettes, ce qui le protège d'un épuisement de ressource mémoire.

5 – L'échange TCP de la figure suivante correspond au transfert d'une page WEB entre un navigateur Web et un serveur Web.

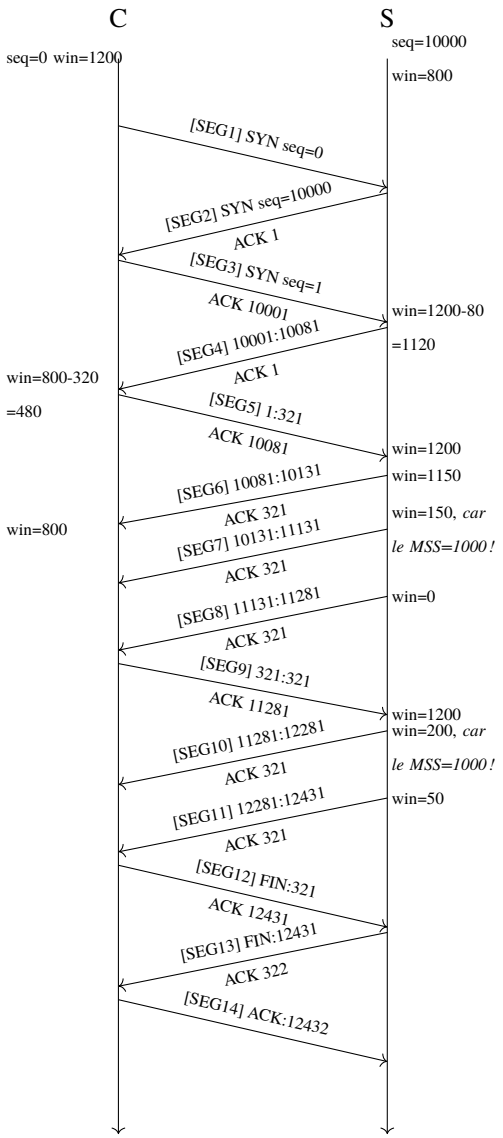
a. Le schéma est complété de la façon suivante :



b. Description des différents segments :
   - ▷ segments 1, 2 et 3 correspondent au « handshake » TCP, c-à-d l'établissement de la connexion TCP.
   - ▷ segment 3 : contient les données de la requêtes Web transmises.
   - ▷ segment 4 : correspond aux données de réponses de la part du serveur Web.
   - ▷ segment 5 : confirme la bonne réception des données envoyées par le serveur et demande la terminaison de la demi-connexion $N \rightarrow S$.
   - ▷ segment 6 : confirme, de la part du serveur, la terminaison de la demi-connexion demandée par le segment 5.
   - ▷ segment 7 : le serveur demande à son tour, la terminaison de la demi-connexion de $S \rightarrow N$.
   - ▷ segment 8 : confirme, de la part du navigateur, la terminaison de la demi-connexion demandée par le segment 7.

c. Les données de la requête ont été envoyées en un seul segment, la taille de la fenêtre appliqué au navigateur par le serveur est supérieure à 100 octets.
   De même, la taille de la fenêtre appliqué au serveur par le navigateur est supérieure à 1000 octets, car les données de la réponse ont été transmises en un seul segment.

6 – Une communication est décrite du point de vue de la couche 5, « application », en 6 étapes :



- segment 1, 2& 3 : « *handshake* » de connexion avec $ISN_{client} = 0$ et $ISN_{serveur} = 10000$
- segment 4 : le serveur envoie 80 octets, la « window size » du client est de 1200, il est autorisé à envoyer $1200 - 80 = 1120$ octets ;
- segment 5 : le client acquitte les données reçues du serveur et 320 octets de données (sa fenêtre d'autorisation d'envoi passe à 480) ;
- segment 6 : le serveur envoie 50 octets, la « window size » du client est de 1200, il est autorisé à envoyer $1200 - 50 = 1150$ octets ;
- segment 7 : le serveur n'a pas reçu d'acquittement de la part du client, il envoie un segment de taille inférieur au MSS (1000 octets) et tenant dans la fenêtre d'autorisation du client (1150 octets) ;
- segment 8 : envoie des 150 octets autorisés par la fenêtre du client ;
- segment 9 : le client acquitte la réception des segments 6, 7 & 8 ;
- segment 10 &11 : le serveur a fait « glisser » sa fenêtre d'émission grâce à l'acquittement du client, il recommence à envoyer deux segments pour envoyer les $2300 - 1150 = 1150$ octets restants ;
- segments 12,13 & 14 : fin simultanée de la connexion TCP.