Le protocole TCP

Selma - Département Informatique - IUT Fontainebleau/Sénart

Position dans la pile protocolaire.

TCP, Transmission Control Protocol est un protocole réseau au niveau applicatif, c'est à dire au plus près de l'application (processus). **RFC 793**

Pour rechercher un document RFC en particulier ⇒ *RFC Search*

L'unité de données (PDU) transportée au niveau TCP s'appelle un segment ou datagramme TCP.

Caractéristiques.

La transmission des octets de l'application ne commence que lorsqu'une connexion aura été établie.TCP est donc un protocole en *mode connecté.*

La connexion établie est bidirectionnelle (*full duplex*).

La connexion est établie après négociation de paramètres entre les deux points. Elle se termine par accord entre les deux extrémités de la connexion.

Pendant la transmission, un contrôle permanent du flux des octets est assuré. Les deux extrémités de la communication savent à tout moment si on doit attendre des octets ou pas, lesquels, comment les réclamer, etc.

Dans le noyau.

Dans le contexte des réseaux, un point qui communique s'appelle socket. Pour simplifier, on peut le voir comme une paire (adr_ip, num_port).

Au niveau processus, communiquer via le réseau est réalisé en écrivant et en lisant dans un descripteur de socket obtenu via l'appel système :

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

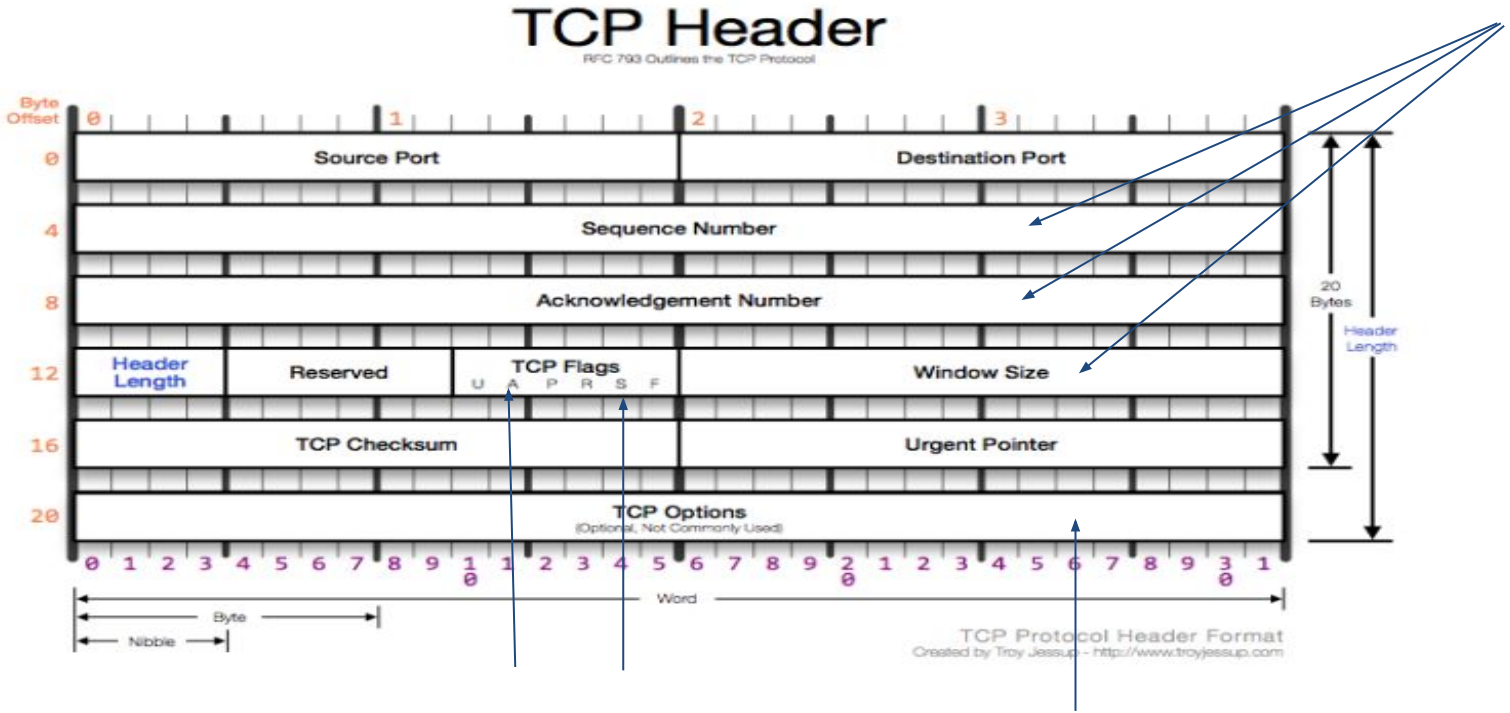
La connexion établie est complètement spécifiée par la paire de sockets aux extrémités.

Pour chaque connexion établie, le noyau de la machine maintient un ensemble de structures de données appelé TCB, *Transmission Control Block*.

Dans le TCB, des pointeurs référencent un tampon, SENDBUF, vers les octets utilisateur (ceux émis par l'application locale) à envoyer, et un autre, RCVBUF, vers les octets utilisateur reçus de l'application distante.

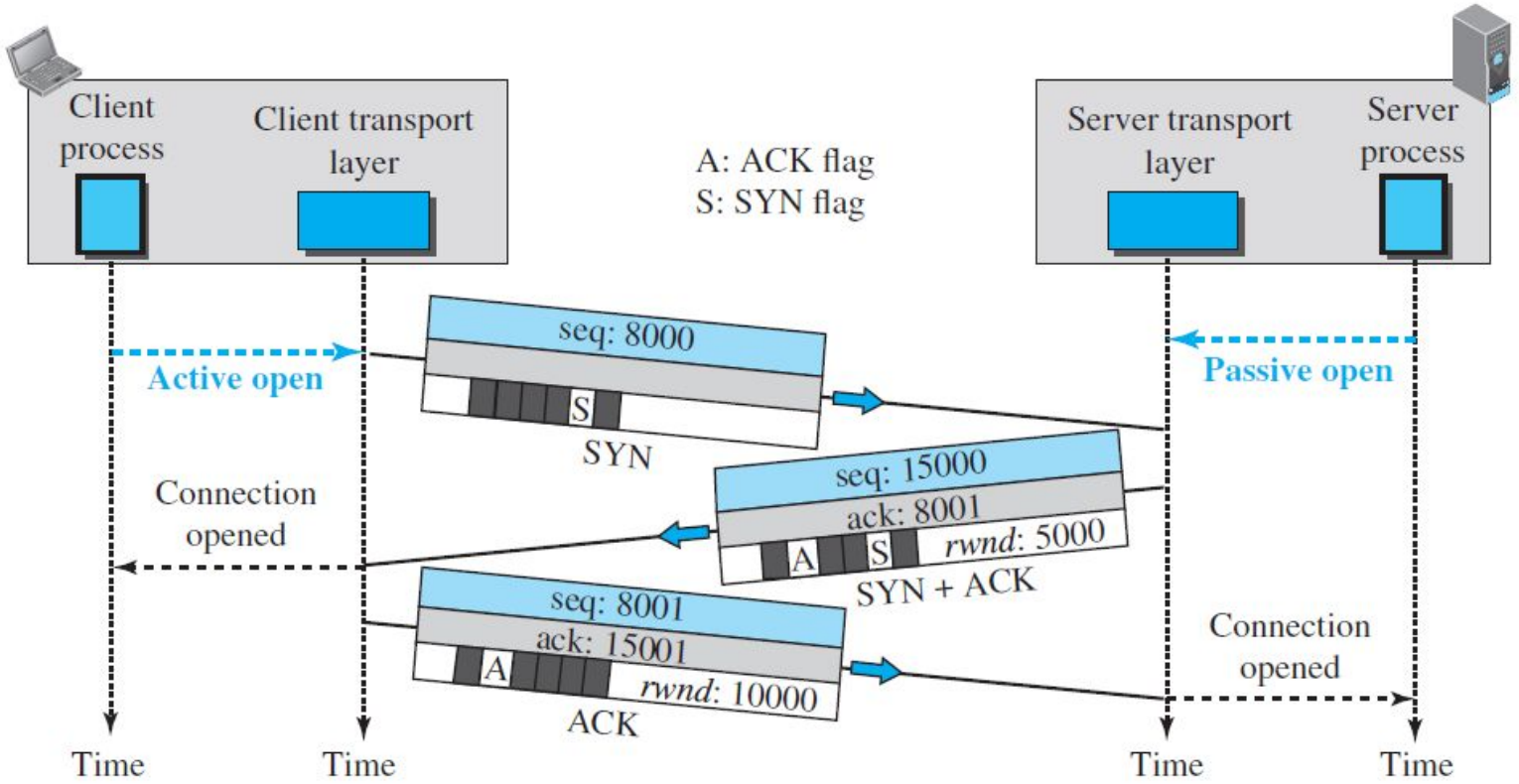
Lorsqu'une application TCP émet des octets (appel système *send* dans le socket approprié), ces octets sont placés dans SENDBUF en vue de leur empaquetage dans des segments. Les octets issus d'une application TCP sont numérotés consécutivement à partir de 0. Le numéro du premier octet dans une séquence d'octets portée par un segment est appelé le *nombre de séquence* du segment, *sequence number*.

Les champs TCP intervenant lors de l'établissement d'une connexion TCP.



Etablissement d'une connexion TCP.

L'extrémité qui initie la demande de connexion est le *client*. L'autre extrémité est le *serveur*.



https://networkengineering.stackexchange.com/questions/63325/sequence-numbers-of-segments-in-three-way-handshakes-of-tcp-connection-establish

Client → Serveur.

Un générateur de nombres initiaux de séquences est utilisé par TCP pour sélectionner un nouveau nombre 32 bits, appelé ISN, *Initial Sequence Number*. Dans le schéma précédent ISN est X=8000.

Un segment part en direction du serveur avec *seq number* = ISN et le flag TCP S (Synchronize) levé.

C'est une demande de synchronisation sur les nombres de séquences.

Serveur → Client

Un générateur de nombres initiaux de séquences est utilisé par TCP pour sélectionner un nouveau nombre 32 bits, ISN. Dans le schéma précédent ISN est Y=15000.

Un segment part en direction du client avec *seq number* = ISN = 15000 =Y, le flag S levé, le flag A (Acknowledgment) levé et le champ *ack number* avec la valeur X+1 : "j'attends l'octet de données numéro X+1".

Client → Serveur.

Un segment part en direction du serveur avec *seq number*=X+1, le flag S à 0, le flag A (Acknowledgment) levé, et le champ *ack number* avec la valeur Y+1 :" j'attends l'octet de données numéro Y+1".

- CONNEXION ÉTABLIE -

Les octets de données vont commencer à voyager en partant du nombre de séquence X+1 côté client, et Y+1 côté serveur.

Autres paramètres communiqués pendant la procédure Handshake.

⇒ MSS option.

Dans le champ option de l'en-tête TCP, chacun communique à l'autre, pendant l'établissement de la connexion, la valeur de MSS, *Maximum Segment Size*. C'est la taille maximale en octets des données empaquetées dans un segment que chacun est capable de traiter en une seule fois à la réception.

La valeur de MSS prend en compte principalement la valeur de la MTU, *Maximum Transmission Unit,* du réseau sur lequel on reçoit le paquet empaquetant le segment.

⇒ RWND variable value.

Receive Window Size est la taille en octets des données reçues que le récepteur accepte d'accumuler sans envoyer d'accusés de réception les concernant. La valeur de cette variable, RCV.WND, est mise à jour dans le TCB du récepteur en fonction de ce qui est constaté comme différences de vitesses de fonctionnement entre l'émetteur et le récepteur.

La valeur de RCV.WND est communiquée par le récepteur dans le champ *Window*.

La fenêtre TCP.

Window: 16 bits (TCP header)
The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

TCP permet au récepteur de gérer le montant d'octets envoyé par l'émetteur en communiquant à l'émetteur un intervalle de nombres de séquences acceptés au-delà du dernier segment qu'il a acquitté.

Cette information est positionnée par le récepteur dans le champ Window des segments qu'il envoie en réponse. Elle est pertinente pour l'émetteur dans chaque segment qu'il reçoit avec un ack number supérieur au dernier ack number reçu.

Si le nombre d'octets envoyés par l'émetteur et non encore acquittés atteint la dernière valeur de la fenêtre notifiée par le récepteur, l'émetteur doit cesser d'émettre jusqu'à la réception d'une nouvelle autorisation.

Cette autorisation aura lieu lorsque le récepteur aura accusé réception de tout ou une partie des données qui attendent d'être acquittées (*outstanding sender data*) et que la nouvelle fenêtre annoncée est "suffisante".

Quand réémettre un segment ?

RFC 5681

Dans le TCB des pointeurs référencent le segment courant (le prochain qui va partir) et une queue de retransmission.

Lorsqu'un segment S de données est envoyé, une copie en est placée dans la queue de retransmission et un minuteur (timer) est démarré avec le temps RTO : *Retransmission Time Out*. Ce temps est calculé en fonction de certains paramètres dont le temps RTT : *Round Trip Time* (temps estimé d'aller-retour).

Si le segment qui acquitte S arrive avant expiration de RTO, alors la copie de S est supprimée de la queue de retransmission.

Si RTO expire et le segment qui l'acquitte n'est toujours pas arrivé, alors le segment S est retransmis, et RTO est multiplié par 2.

Des retransmissions successives du même segment font donc multiplier RTO par 2, à chaque nouvelle retransmission. On abandonne lorsque RTO atteint CTO : *Connection Time Out.*

Si on reçoit un segment avec le même nombre d'acquittement N qu'un segment déjà reçu (un duplicate ack), alors on est en train de nous avertir qu'on constate des trous dans les nombres de séquences des segments reçus.

⇒ Dans ce cas, pour éviter de congestionner le réseau avec des retransmissions, on attend le premier des deux évènements suivants avant de retransmettre le segment de nombre de séquence N : (1) RTO expire - (2) on reçoit encore deux segments avec le duplicate *ack number* N.

Terminaison d'une connexion TCP.

L'utilisateur demande à TCP de fermer la connexion (appel système close sur le socket approprié) :TCP A.

Un segment FIN (Finish) est construit par TCP A avec le flag de contrôle F levé, et est placé dans la queue des segments en partance.

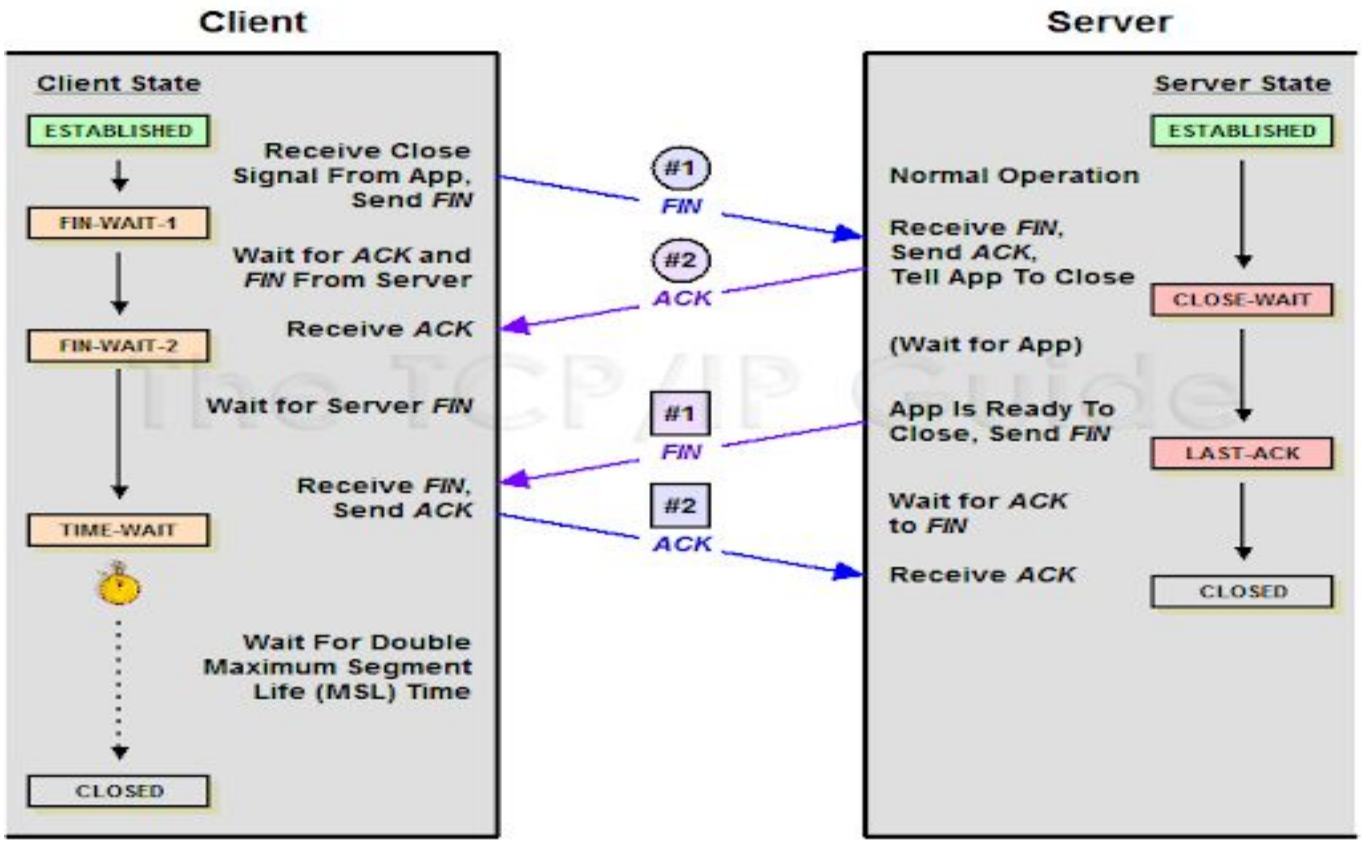
Aucun nouveau segment à envoyer n'est accepté par TCP A. La connexion TCP côté TCP A entre dans l'état FIN-WAIT-1. Pendant cet état, les réceptions de segments peuvent continuer.

Tous les segments envoyés précédemment par TCP A ainsi que le segment FIN sont retransmis si nécessaire jusqu'à ce qu'ils soient tous acquittés par l'autre TCP : TCP B.

Lorsque TCP B aura acquitté le segment FIN et envoyé son propre segment FIN, TCP A pourra acquitter ce segment FIN. La connexion TCP côté TCP A entre alors dans l'état TIME-WAIT où on attend un temps égal à 2MSL avant que la connexion côté TCP A ne passe à l'état CLOSED. Le TCB correspondant est alors supprimé.

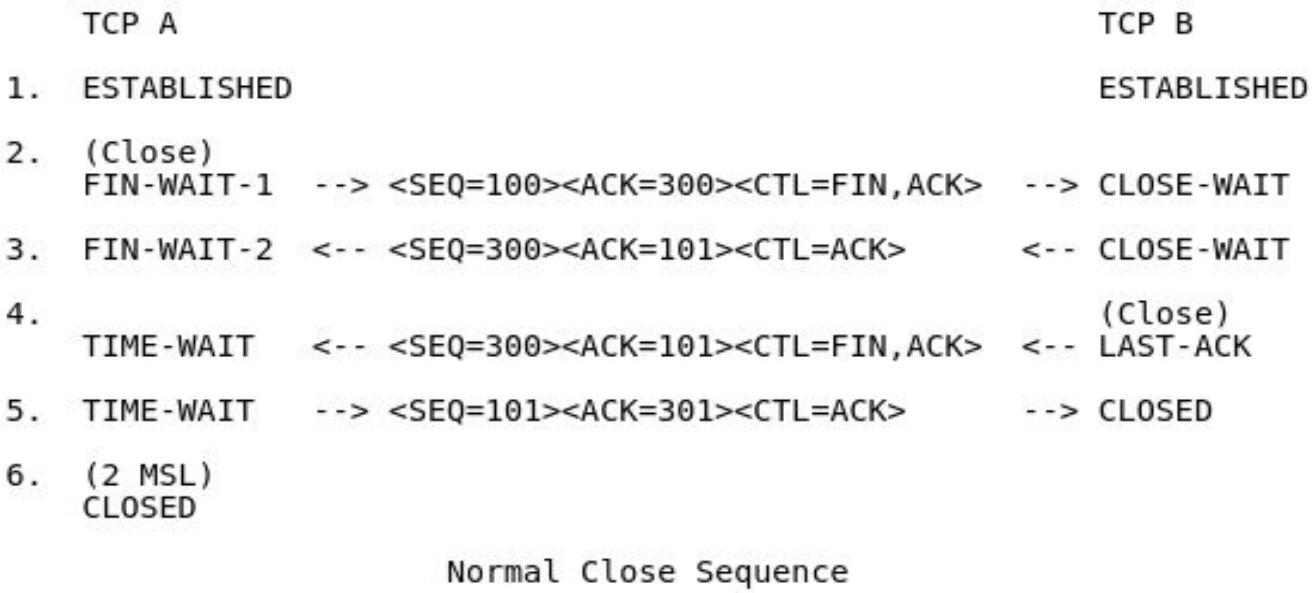
MSL : *Maximum Segment Life* est utilisé pour un certain nombre de raisons. Ici, l'état TIME-WAIT dure 2MSL pour être sûr que ACK de TCP A est arrivé à TCP B et le retransmettre le cas échéant.

Lorsque TCP B reçoit FIN, il l'acquittera mais ne pourra envoyer son propre FIN que lorsque son application aura fermé la connexion.



http://www.tcpipguide.com/free/t_TCPConnectionTermination-2.htm

Extrait de RFC 793



Normal Close Sequence