

LES PROTOCOLES TCP ET UDP

1. LES NUMEROS DE PORT

TCP et UDP sont des protocoles de la couche Transport (niveau 4) du modèle OSI .

A la réception d'un datagramme, une machine est identifiée de manière unique par la couche IP à l'aide de l'adresse Internet, la couche IP délivre ensuite le datagramme reçu à TCP ou à UDP en fonction de la valeur du champ Protocole contenu dans l'en-tête IP.

La couche TCP comme la couche UDP doit délivrer le datagramme à l'application destinataire, TCP et UDP doivent alors reconnaître l'application destinataire. Cette distinction est assurée par un numéro appelé *numéro de port*, à chaque application correspond un numéro de port (quelquefois deux).



Le numéro de port est un entier positif sur 16 bits, le rfc 1700 donne la liste des numéros de port attribués aux applications connues. Les numéros 0 à 1023 sont réservés aux applications standards, les numéros supérieurs à 1024 sont disponibles pour n'importe quelle autre application.

| Application | N° de port | Protocole | Description |
|-------------|------------|-----------|---|
| echo | 7 | TCP | Le serveur retourne les données envoyées par le client |
| discard | 9 | TCP | Le serveur rejette les données envoyées par le client |
| ftp | 21 | TCP | Utilisé par ftp pour transmettre les ordres |
| ftp | 20 | TCP | Utilisé par ftp pour transmettre les données |
| telnet | 23 | TCP | Terminal virtuel de réseau |
| smtp | 25 | TCP | Le courrier électronique |
| http | 80 | TCP | Protocole utilisé par les serveurs Web et les navigateurs |
| echo | 7 | UDP | Le serveur retourne les données envoyées par le client |
| discard | 9 | UDP | Le serveur rejette les données envoyées par le client |
| tftp | 69 | UDP | Transfert de fichiers (Trivial FTP) |
| talk | 517 | UDP | Dialogue entre usagers distants |

2. LE SEGMENT TCP (Transport Control Protocol)

TCP procure un service transport fiable : on dit que TCP offre un service de transfert de données orienté connexion ou bien encore que TCP travaille en mode connecté.

Lorsque 2 applications veulent échanger des données, le service connecté apporté par TCP est caractérisé dans ses grandes lignes par les principales fonctionnalités suivantes :

- Etablissement préalable d'une connexion entre les 2 applications qui veulent échanger des données.
- Identification de chaque paquet transmis à l'aide d'un système de numérotation.
- Acquiescement par le destinataire de chaque paquet reçu.
- Libération de la connexion à la demande des applications.

Le paquet de données délivré par TCP à la couche IP est appelé un *segment*.

Segment et en-tête TCP

| | | | | | | | | |
|------------------------|---------|---|----|-----------------------|----|----|----|----------------------|
| 0 | 7 | 8 | 15 | 16 | 23 | 24 | 31 | |
| Port Source | | | | Port Destination | | | | |
| Numéro de séquence | | | | | | | | |
| Numéro d'acquiescement | | | | | | | | |
| L (sur 4 bits) | Réserve | U | A | P | R | S | F | Taille de la fenêtre |
| Checksum | | | | Offset message urgent | | | | |
| Options | | | | | | | | |
| Données | | | | | | | | |

L'en-tête TCP a une longueur de 20 octets sans les options.

Port Source : Permet d'identifier de manière unique l'application qui a émis ce paquet.

Port Destination : Permet d'identifier de manière unique l'application destinataire de ce paquet.

Numéro de séquence : Ce champ identifie le 1^{er} octet du champ Données du paquet transmis. Chaque octet est numéroté, le 1^{er} numéro est égal à ISN+1, ISN (Initial Sequence Number) est un nombre déterminé en interne par chaque émetteur de données. Le 1^{er} octet du 1^{er} paquet transmis porte le numéro ISN+1.

Numéro d'acquiescement : Lorsque le récepteur acquiesce l'émetteur (flag Ack positionné), il place dans ce champ le prochain numéro de séquence qu'il s'attend à recevoir.

L (Longueur de l'en-tête) : Donne la longueur de l'en-tête en mots de 32 bits; la taille normale de l'en-tête est de 20 octets quand il n'y a pas d'option.

Les bits de contrôle : Ils permettent de définir la validité de certains champs et la fonction du paquet.

- U (URGent) : Le champ Offset message urgent est valide, ce champ offset indique le rang du 1^{er} octet du message urgent dans le paquet transmis.
- A (ou ACK comme ACKnowledge) : Acquiescement, indique que le champ Numéro d'acquiescement est valide.
- P (ou PSH comme PuSH) : La couche TCP doit délivrer ce segment à la couche Application le plus rapidement possible.
- R (ou RST comme ReSeT) : Fermeture de la connexion à cause d'une erreur irrécupérable.
- S (ou SYN comme SYNchronisation) : Indique l'ouverture d'une connexion, le champ numéro de séquence est valide.
- F (ou FIN comme FINish) : Indique une fermeture normale de la connexion, l'émetteur de ce paquet signale ainsi qu'il n'a plus de données à émettre.

Taille de la fenêtre : Ce champ permet un contrôle automatique de flux, un récepteur indique ainsi à un émetteur le nombre maximal d'octets qu'il peut accepter.

Checksum : Cette somme de contrôle est calculée comme il est indiqué plus bas.

Offset message urgent : Champ valide que si le bit URG est positionné à 1.

Options : Le champ Options le plus courant se rencontre lors de l'établissement de la connexion entre 2 entités. Chaque entité indique alors à l'autre la taille maximale en octets d'un paquet de données qu'elle peut recevoir. Cette taille maximale de la partie Données d'un segment TCP est appelée MSS (Maximum Segment Size).

La segmentation

Lors d'une émission, TCP reçoit le message à émettre de la couche Application. Il doit segmenter ce message si sa taille en octets est supérieure au MSS du récepteur.

Par exemple, si la couche Application transmet un message 1200 octets et que le MSS du récepteur est de 500 octets, cela impose la segmentation des données par la couche TCP en 3 paquets : le 1^{er} et le 2nd auront une longueur de 500 octets, le 3^{ème} une longueur de 200 octets.

TCP ajoute son entête à chaque paquet, il délivre successivement à IP 3 segments de longueur 520, 520 et 220 octets.

La couche TCP du récepteur reçoit successivement les 3 segments, elle s'occupe de réassembler les 3 paquets de données dans le bon ordre afin de reconstituer le message initial qu'elle délivre à l'application concernée.

Le checksum

Le checksum est calculé sur tout le segment TCP et sur un pseudo en-tête dont le format est le suivant :

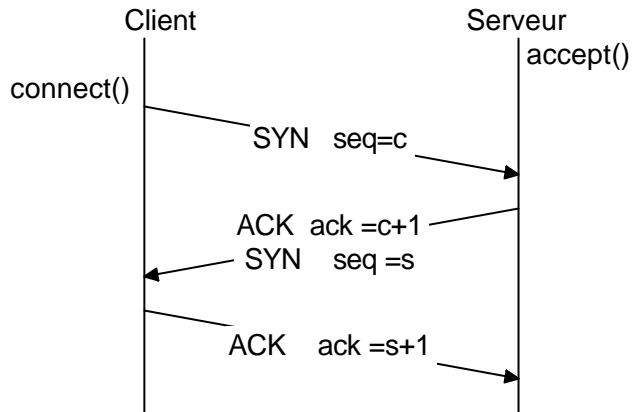
| | | | |
|------------------------------------|------------------------|--------------------------------|----|
| 0 | 15 | 16 | 31 |
| Adresse Internet de la source | | | |
| Adresse Internet de la destination | | | |
| 00 | Protocole destinataire | Longueur totale du segment TCP | |

Le checksum est obtenu en effectuant la somme des tous les mots du pseudo en-tête et du segment entier TCP puis en prenant le complément à 1 du résultat.

3. ETABLISSEMENT D'UNE CONNEXION

TCP est un protocole connecté, il établit une connexion avant que s'effectue le transfert des données. On indique sur les diagrammes temporels les fonctions de la bibliothèque TCP/IP que les applications Serveur et Client utilisent.

- Un serveur se place en attente d'une demande de connexion, il exécute la fonction `accept()`.
- Un client demande à se connecter en exécutant la fonction `connect()`. Le client doit parfaitement identifier l'application serveur par son adresse IP ainsi que le numéro port qui lui est affecté.



Il en résulte un échange de segments entre les couches TCP du client et du serveur.

1. La couche TCP du client demande l'ouverture d'une connexion en positionnant le bit SYN à 1, il place aussi dans le champ N° de séquence seq une valeur c égale à ISN, ce N° de séquence initial correspond à la valeur d'un compteur 32 bits (valeur au moment de la demande de connexion, le compteur est incrémenté de 1 toutes les 4 ms).
2. La couche TCP du serveur répond positivement à cette demande de connexion en positionnant aussi le bit SYN à 1. Il donne lui aussi son N° de séquence seq égal à s. Il positionne le bit ACK à 1 indiquant ainsi que le champ N° d'acquittement est valide, ce champ ack contient le N° de séquence c du client augmenté de 1. On dit qu'un segment SYN consomme un N° de séquence (alors qu'il ne transporte pas de données).
3. La couche TCP du client acquitte le segment SYN du serveur, il positionne le bit ACK à 1 indiquant ainsi que le champ N° d'acquittement est valide, ce champ ack contient le N° de séquence s du serveur augmenté de 1.

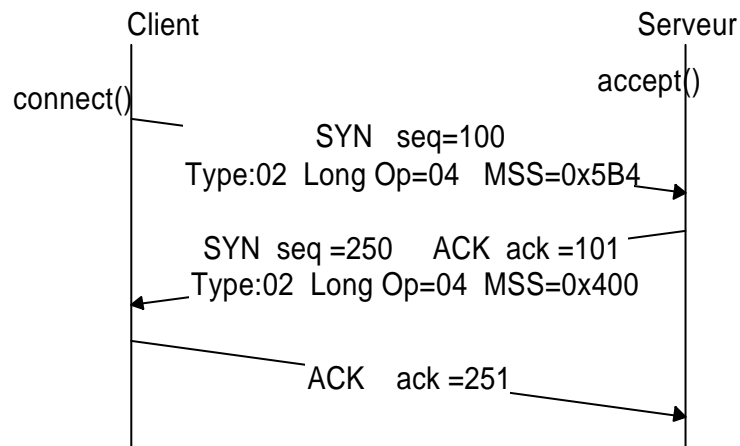
La taille maximale d'un segment ou MSS

Cette valeur est transmise dans le champ Options entre les 2 couches TCP au moment de la connexion.

| | | |
|----------|--------------------|-----|
| Type = 2 | Longueur options=4 | MSS |
|----------|--------------------|-----|

Le champ Options est constitué d'un octet Type, d'un octet Longueur du champ Options et d'un mot 16 bits qui est le MSS.

L'exemple suivant montre un client qui a une valeur de MSS égale à 0x5B4 (1460) et un serveur dont le MSS vaut 0x400 (1024).



On constate que le MSS du client est optimisé pour Ethernet : 1460 de MSS + 20 octets d'en-tête TCP + 20 octets d'en-tête IP donne 1500 octets qui le nombre maximal de données dans une trame Ethernet.

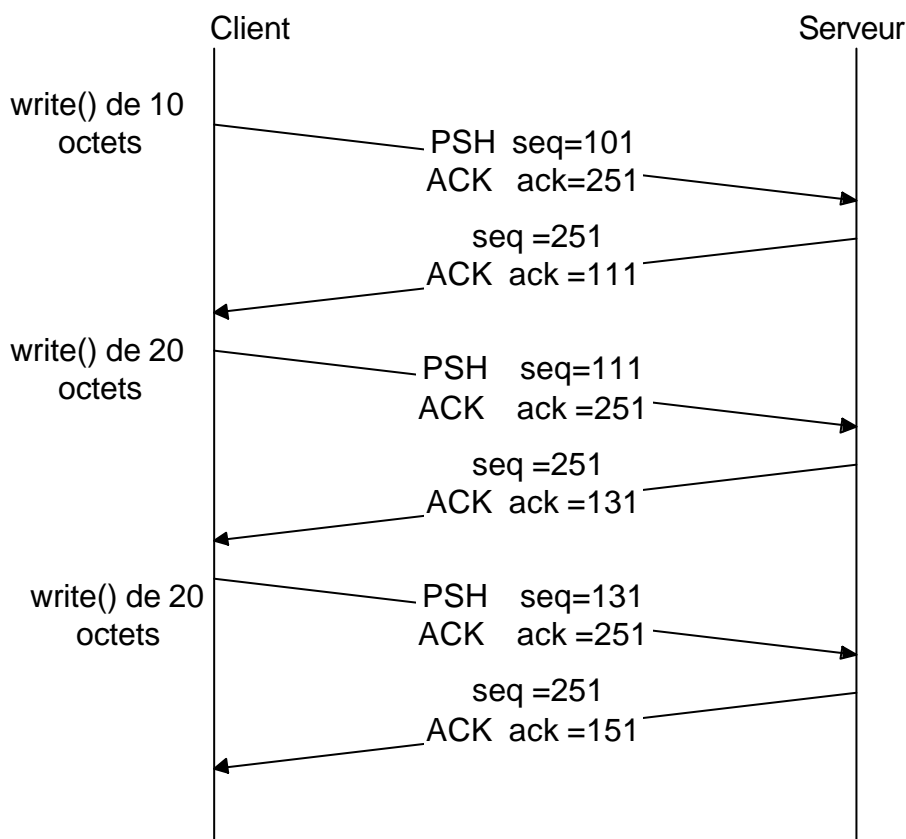
4. TRANSFERT DE DONNEES

Le code d'une application peut employer :

- La fonction write() pour fournir des données à la couche TCP (émission).
- La fonction read() pour lire les données que TCP tient à sa disposition (réception).

On part de l'exemple précédent où le N° de séquence attendu du client est 101 et celui du serveur 251.

Exemple : Le client transmet des données



- L'application client effectue un 1^{er} write() de 10 octets, la couche TCP du client transmet un segment contenant ces 10 octets. Le flag PSH indique à la couche TCP du serveur qu'elle peut délivrer ces 10 octets à l'application car il n'y a pas de segmentation.
- La couche TCP du serveur lit ces 10 octets, elle les stocke en attendant que le client les lise. Elle acquitte le client en lui transmettant un segment dans lequel le bit ACK est positionné indiquant un acquittement et que le champ N° d'acquittement est valide. Ce champ contient le N° de séquence du 1^{er} octet du prochain segment que TCP du serveur s'attend à recevoir : 101+10=111.
- L'application client effectue un 2^{ème} write() de 20 octets, la couche TCP du client transmet un segment contenant ces 20 octets. Le flag PSH indique à la couche TCP du serveur qu'elle peut délivrer ces 20 octets à l'application car il n'y a pas de segmentation.

- La couche TCP du serveur lit ces 20 octets, elle les stocke en attendant que le client les lit. Elle acquitte le client en lui transmettant un segment dans lequel le bit ACK est positionné indiquant un acquittement et que le champ N° d'acquittement est valide. Ce champ contient le N° de séquence du 1^{er} octet du prochain segment que TCP du serveur s'attend à recevoir : $111+20=131$.
-

L'indicateur PSH signale à la couche TCP réceptrice qu'elle peut transmettre les données du paquet à l'application. Cela signifie concrètement que si l'application effectue un `read()`, elle ne bloquera pas et retournera aussitôt avec des données.

Acquittements retardés

En réalité TCP n'acquitte pas instantanément le segment reçu, il attend un certain délai pour tenter d'associer dans un même segment l'acquittement du paquet reçu et l'envoi de données. La valeur maximale de ce retard dépend des diverses implémentations, sa valeur courante est fixée à 200ms.

L'utilisation de ce mécanisme diminue le trafic sur le réseau. Il faut alors que l'application réceptrice soit en attente de lecture de données (avec `read()` par exemple) et que l'exploitation de ces données entraîne une réaction immédiate avec l'envoi d'une réponse (avec `write()` par exemple).

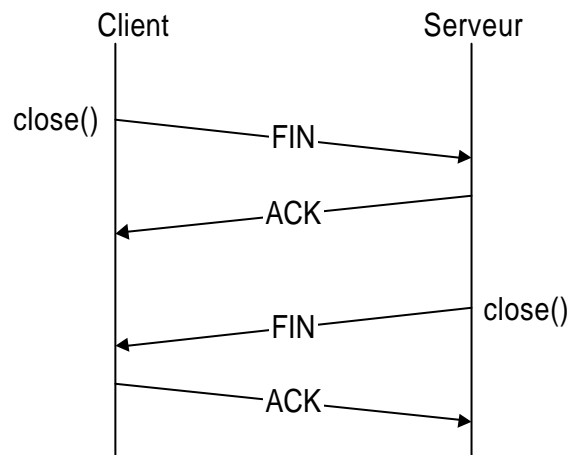
5. FERMETURE DE LA CONNEXION

La fermeture de la connexion est normalement laissée à l'initiative du client.

L'application client exécute `close()`, la couche TCP du serveur acquitte automatiquement le client mais il faut que l'application serveur détecte alors la fermeture du client.

Cette application serveur est généralement en attente de données du client en exécutant `read()`.

La fonction `read()` retourne alors 0, TCP du serveur signalant ainsi à l'application la fermeture de la connexion.

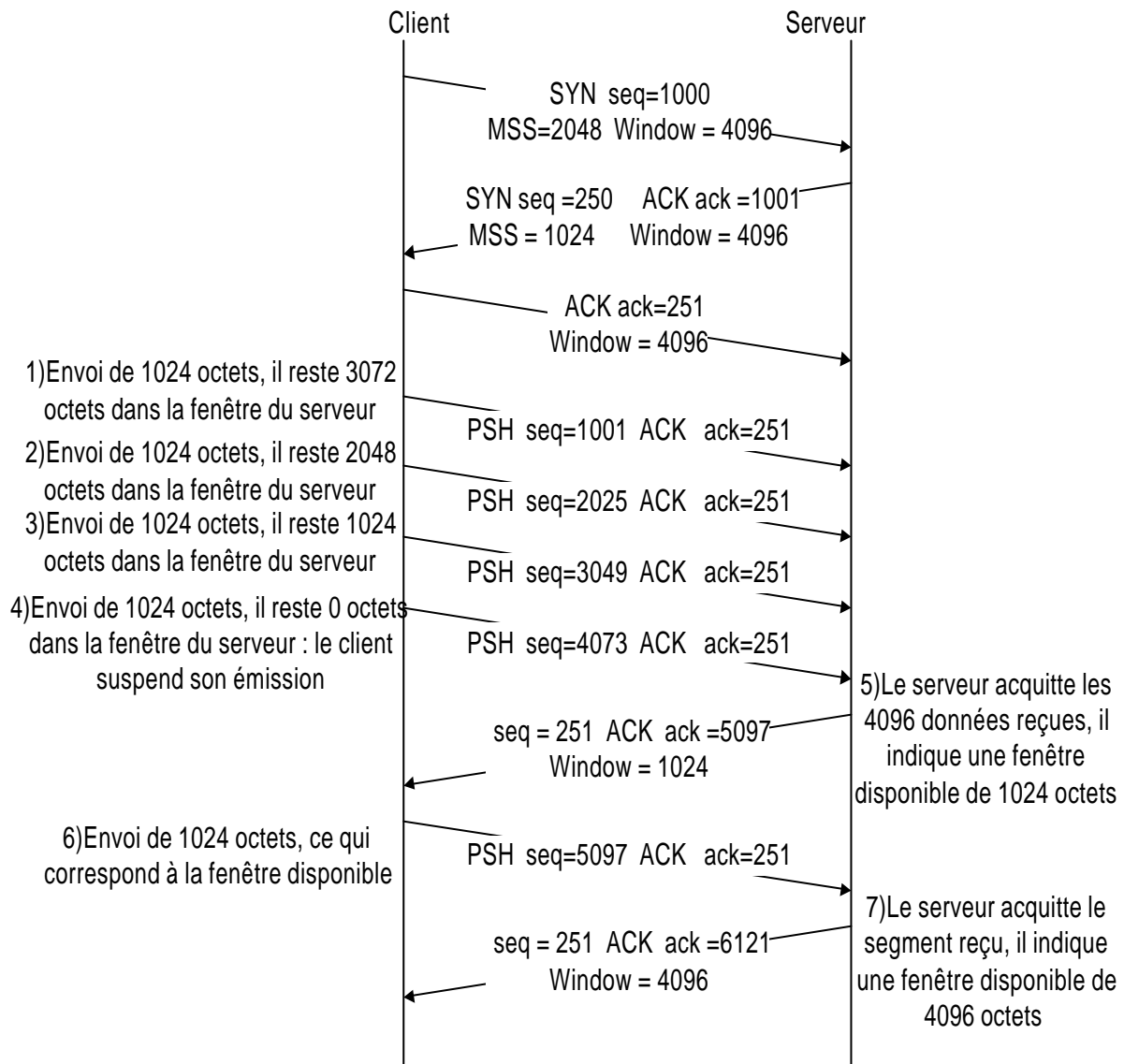


6. FLUX DE DONNEES EN MASSE - LA FENETRE GLISSANTE

Le dialogue "Envoi d'un paquet - Attente de l'acquittement" peut s'avérer très lent dans le cas d'un réseau WAN. TCP autorise alors un émetteur d'envoyer un paquet sans attendre l'acquittement du paquet précédent émis. Mais, dans ces conditions, il est évident que le récepteur peut se trouver engorgé par trop de données reçues. Pour éviter cette situation, TCP a prévu de réguler l'émission des paquets de l'émetteur par un contrôle de flux appelé protocole de fenêtre glissante.

Le champ Window de l'en-tête TCP indique au destinataire le nombre d'octets maximal que l'émetteur peut recevoir.

Le diagramme des temps suivant illustre ce mécanisme de la gestion de la fenêtre.



Le client transmet 4 segments consécutifs 1, 2, 3 et 4. Il remplit la fenêtre du serveur. Le serveur acquitte en 5 tous les segments reçus et précise que sa nouvelle fenêtre est de 1024 octets. Le client émet alors en 6 un segment de 1024 octets. Le serveur acquitte ce segment en 7 et signale que sa fenêtre est revenue à la taille maximale.

7. DIAGRAMME ETATS TRANSITIONS DE TCP

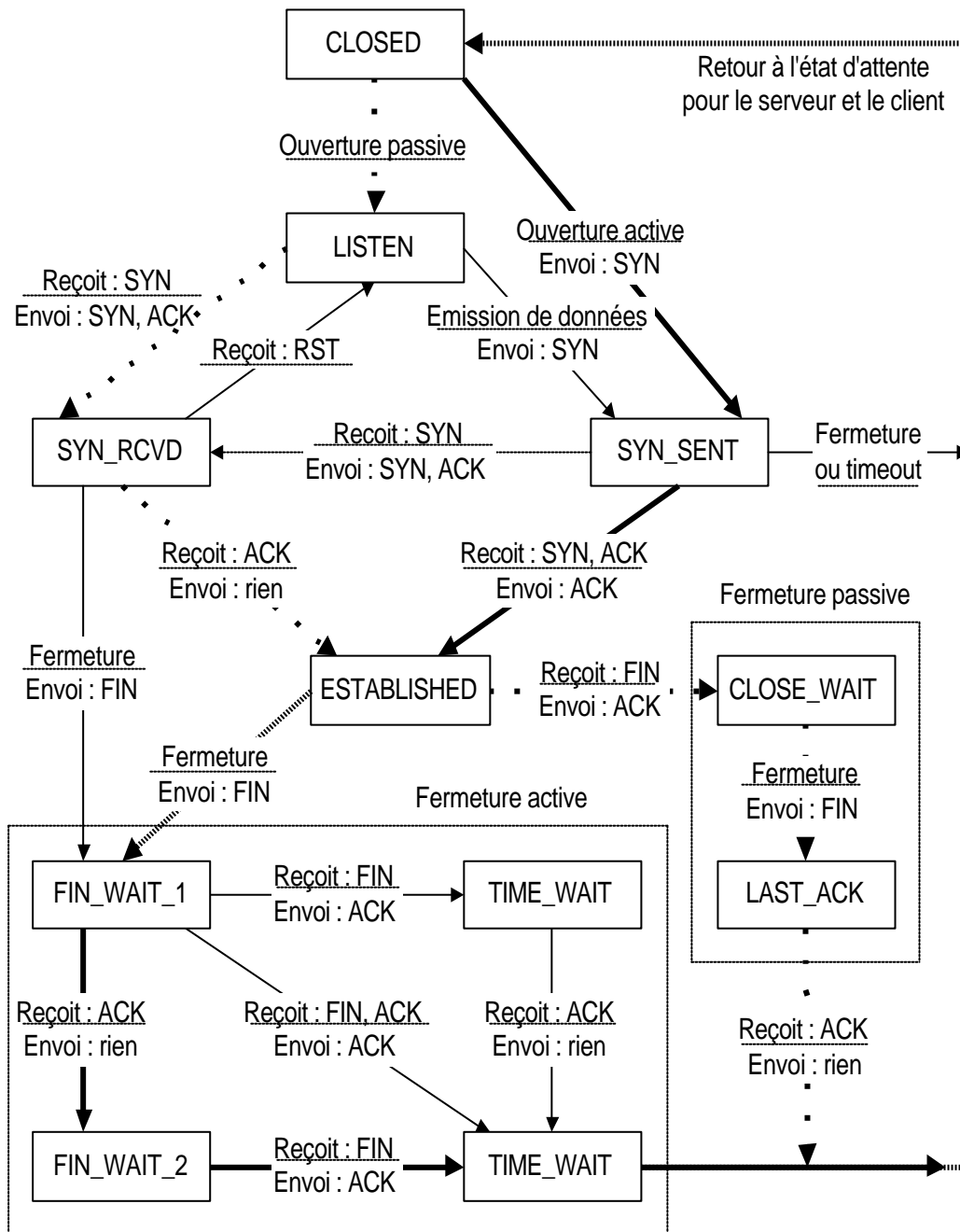
Chaque état est symbolisé par un rectangle, les flèches indiquent les transitions entre les états.

- ➔ En trait plein épais : Transitions normales suivies par le client
- - - -➔ En trait pointillé épais : Transitions normales suivies par le serveur

Chaque transition est représentée par une barre verticale avec des noms de part et d'autre, le nom au-dessus de cette barre indique l'événement survenu, le(s) nom(s) en dessous

désigne(nt) les actions résultantes engagées par TCP en réponse à l'occurrence de l'événement.

Les événements suivants sont provoqués par l'application : Ouverture active, ouverture passive, fermeture, émission de données.



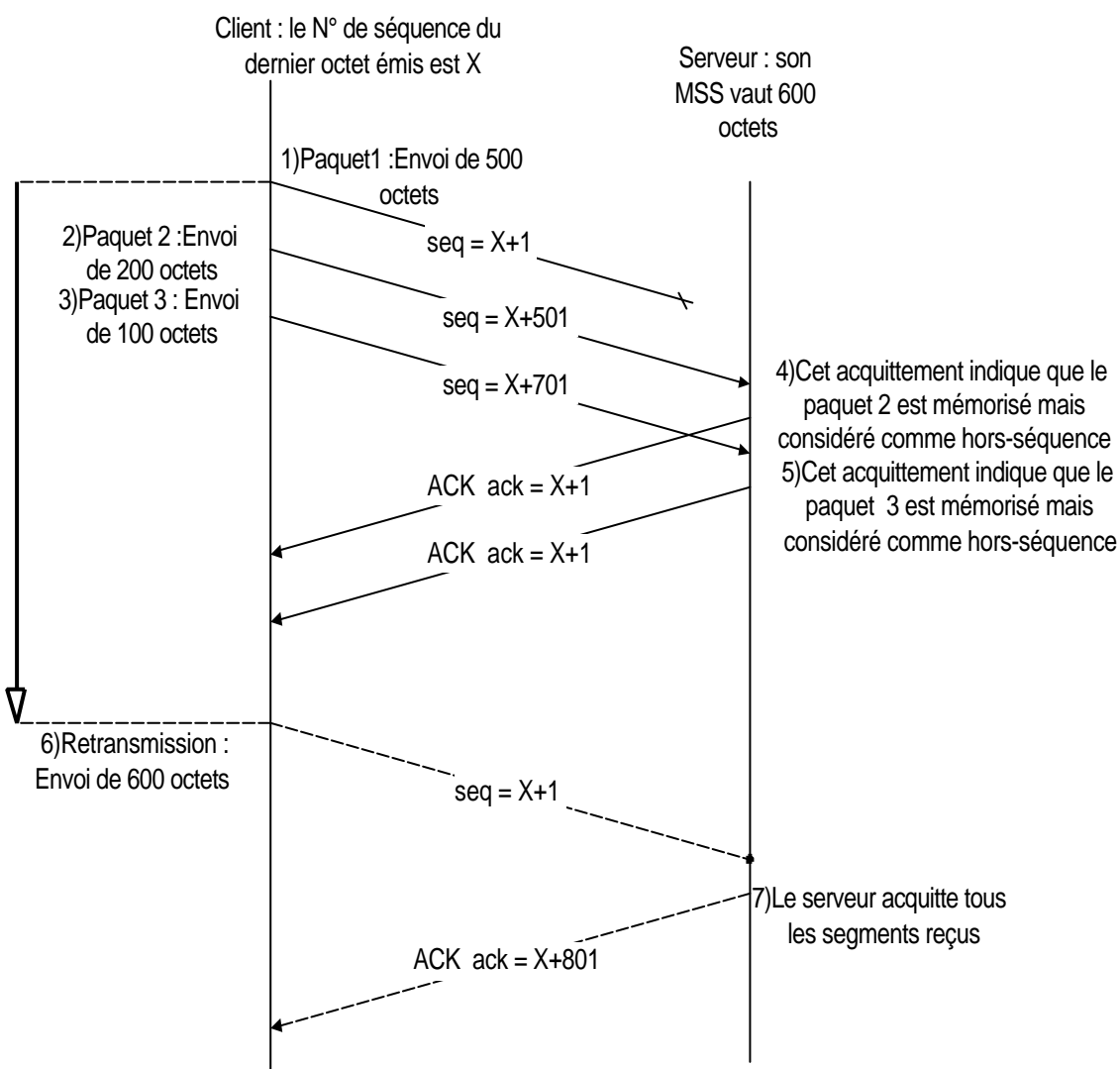
8. TIMEOUT ET RETRANSMISSION DE TCP

Le mécanisme des acquittements mis en place par TCP assure un transport fiable. Il se peut que des segments de données ou des acquittements émis se perdent, TCP gère un compteur de temps qui est initialisé à chaque segment émis, il retransmet le paquet si le timeout se produit alors que les données n'ont pas été acquittées.

Le diagramme des temps suivant présente un exemple de la retransmission par TCP d'un segment non acquitté.

Une temporisation est déclenchée lors de l'émission de chaque paquet.

- 1) Le 1^{er} paquet est émis, le temporisateur est déclenché. L'exemple présenté suppose que le 1^{er} paquet est perdu.
- 2) et 3) L'émetteur anticipe l'acquittement de ce 1^{er} paquet en émettant successivement deux autres paquets.
- 4) et 5) L'émetteur reçoit deux acquittements mais il ne peut pas savoir pas à quel paquet ils correspondent. Le serveur précise comme N° d'acquittement le dernier qu'il avait donné dans sa dernière trame d'acquittement (non représenté sur la figure).
- 6) Le délai mesuré par le temporisateur est écoulé, le client sait que la 1^{er} paquet n'a pas été acquitté, il retransmet alors un paquet qui contient MSS données à partir de celle qui n'a pas été acquittée, c'est à dire les données X+1 à X+601.
- 7) Le serveur acquitte tous les paquets reçus.



La difficulté pour TCP est ici d'évaluer convenablement la valeur de la temporisation. Il ne faut pas une valeur trop petite car il y aurait des retransmissions inutiles; et si la valeur est trop grande, l'attente entre 2 retransmissions ralentira de trop le fonctionnement et sous exploitera les capacités du réseau.

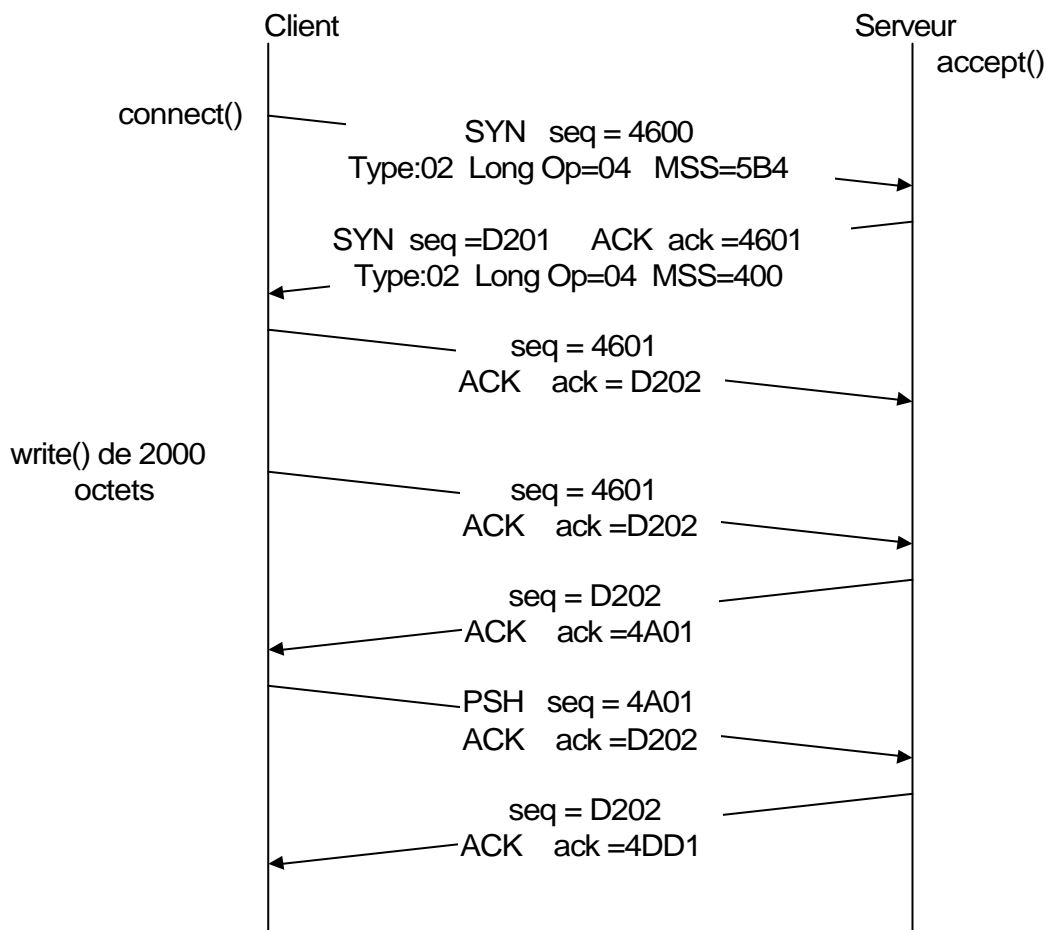
La valeur de la temporisation est calculée à partir du temps d'aller et retour (RTT : Round Trip Time), le RTT est réévalué à la réception de chaque acquittement.

9. SEGMENTATION

Un des rôles de TCP est de découper un message trop long afin d'envoyer des segments dont la taille soit égale au MSS du récepteur.

L'exemple suivante présente une application client qui délivre 2000 octets à sa couche TCP. Le MSS du serveur est de 1024 (0x400), le client émet un 1^{er} paquet de 1024 octets de données, puis un 2^{ème} de 976 (0x3D0) données.

- Le 1^{er} paquet de 1024 octets de données est émis par TCP du client avec le bit d'état PSH qui n'est pas positionné.
- La couche TCP du serveur acquitte ce paquet, le champ ack vaut 4A01=4601+400, elle détecte que PSH est à 0 et stocke ce paquet (il n'est pas transmis à l'application).
- Le 2^{ème} paquet de 976 octets de données est émis par TCP du client avec le bit d'état PSH positionné.
- La couche TCP du serveur acquitte ce paquet, le champ ack vaut 4DD1=4A01+3D0, elle détecte que PSH est à 1, elle reconstitue alors le message avec le précédent paquet puis le transmet à l'application dès qu'elle le demande.



10. LE PROTOCOLE UDP (User Datagram Protocol)

UDP est un protocole simple orienté datagramme qui permet d'envoyer des données d'une application à une autre. UDP n'offre aucune garantie de fiabilité de la communication.

L'en-tête UDP a une longueur de 8 octets.

Datagramme et en-tête UDP

| | | | | |
|-------------|-----|------------------|-------|----|
| 0 | 7 8 | 15 16 | 23 24 | 31 |
| Port Source | | Port Destination | | |
| Longueur | | Checksum | | |
| Données | | | | |

Port Source : Permet d'identifier de manière unique l'application qui a émis ce paquet.

Port Destination : Permet d'identifier de manière unique l'application destinataire de ce paquet.

Longueur : Indique la longueur totale en octets du datagramme (en-tête + données).

Checksum : Ce mot de contrôle se calcule comme pour les paquets TCP.

11. LES SOCKETS

Une application sur une machine donnée est totalement identifiée par :

- L'adresse Internet de la machine.
- Le port affecté à l'application.

L'association <Adresse Internet - N° de port> est appelée une *socket*.

Certains éditeurs de logiciel et quelques auteurs intègrent le protocole utilisé dans la définition d'une socket. Cette précision s'avère nécessaire car deux applications sur une même machine peuvent être associées à un même numéro de port mais l'une travaillant avec TCP et l'autre avec UDP.

La paire de sockets <Adresse Internet Client - N° de port Client - Protocole> <Adresse Internet Serveur - N° de port Serveur - Protocole> identifie de manière unique la "connexion" entre 2 applications.