LES PROTOCOLES DE TRANSPORT

1. Le protocole TCP

1.1. Caractéristiques

TCP est un protocole de la couche Transport au sens du modèle OSI. Il s'exécute au dessus du protocole IP . TCP est un protocole **orienté connexion** qui garantit que les données sont remises de façon **fiable**.

Les fonctionnalités de TCP sont donc principalement :

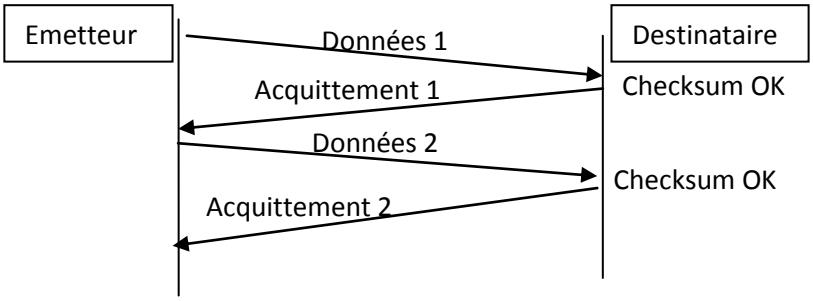
- **Etablissement** d'une **connexion**
- Transmission fiable des données en effectuant un **contrôle des données** et en effectuant une **réémission** pour les données qui n'ont pas pu être transférées correctement.
- **Réordonnancement** des informations transférées. En effet, les informations seront en fait transmises dans des datagrammes IP qui peuvent éventuellement emprunter des chemins différents donc ne pas arriver dans l'ordre d'émission.
- Gérer le **multiplexage**, c'est à dire que plusieurs applications peuvent utiliser simultanément les services du protocole TCP, exemple : un client courrier qui s'exécute en même temps qu'une navigation sur le web et un téléchargement de fichier.
- **Segmentation et séquencement des données:** Lorsque de l'information doit être envoyée d'un émetteur vers un récepteur par le protocole TCP, cette information est découpée en **segments** qui peuvent être de **taille variable**. Mais pour des raisons de fiabilité chaque octet d'un segment va être numéroté avec un **numéro de séquence**, où chaque segment est reconnu par le n° de séquence du premier octet. Les autres numéros (en fait c'est le numéro du dernier qui est intéressant) seront calculés en ajoutant ce numéro au nombre d'octets présents dans la partie "données" du segment. Ce numéro de segment est codé sur 32 bits ce qui permet de numéroter les segments jusqu'à la valeur 2^{32} = 4 294 967 296.

1.2. Mécanisme d'acquittement

La transmission doit être fiable, TCP utilise donc le mécanisme classique d'**acquittement** mais dans une version dite **cumulative**.

1.2.1. Le principe de l'acquittement

Le principe général est assez simple : Lorsqu'un segment est reçu par le destinataire, celui-ci vérifie que les données contenues dans ce segment sont correctes (consultation du checksum) et si c'est le cas envoie un message d'**acquittement positif** (ACK) vers l'expéditeur.



Deux types de problèmes peuvent se produire :

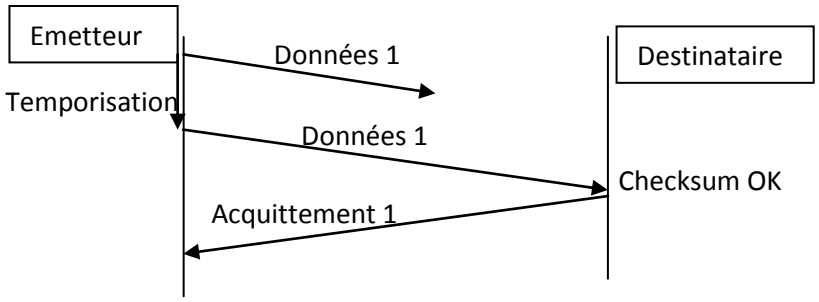
- Les données du segment sont **endommagées**.
- Le segment **n'arrive jamais** à destination.

Pour détecter ce type de problème, chaque fois qu'il envoie un segment l'expéditeur effectue 2 opérations :

- Il stocke dans un buffer une **copie** du segment qu'il vient d'envoyer

- Il arme une **temporisation**

Si au bout d'un certain délai aucun acquittement positif n'a été reçu du destinataire le segment est renvoyé en utilisant la copie présente dans le buffer, si par contre un acquittement est reçu pour ce segment, la copie est supprimée du buffer.

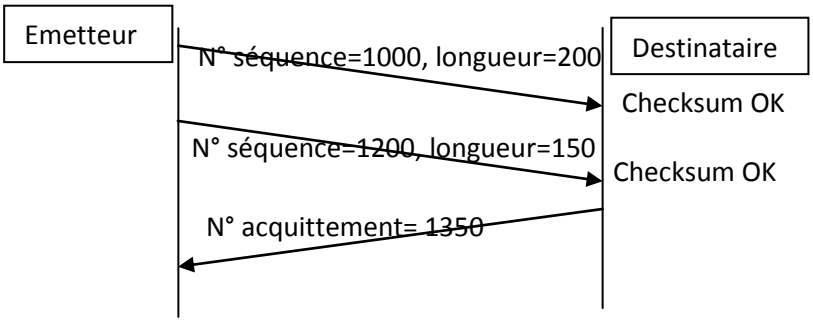


1.2.2. L'acquittement cumulatif

Le protocole TCP utilise le principe de **l'acquittement cumulatif**, c'est à dire que comme les données sont envoyées par segments de taille variable mais comportant le n° de séquence du premier octet du segment, si le contrôle du checksum est satisfaisant, le récepteur déduit à partir du n° de séquence du premier octet et du nombre d'octets reçus le n° de séquence du dernier octet et accuse réception pour cet octet, ce qui implique de façon implicite que tous les octets dont le n° de séquence est inférieur à ce n° de séquence ont été bien reçus.

Il se peut même que pour des raisons d'optimisation, le récepteur attend la réception de plusieurs segments avant d'envoyer un acquittement, ceci a pour but de diminuer le nombre de segments d'acquittements circulant sur le réseau.

Cette technique d'acquittements cumulés a pour principal avantage d'éviter la retransmission de données si un paquet d'acquittement s'est perdu.



1.3. Etablissement d'une connexion

TCP est un **protocole orienté connexion**, cela signifie qu'il va établir et **maintenir** une connexion entre deux machines et **surveiller** l'état de cette connexion pendant toute la durée du transfert.

TCP fonctionne en **full duplex**, c'est à dire que lorsqu'une connexion est établie les données vont pouvoir transiter simultanément dans un sens et dans l'autre.

La demande de connexion peut s'effectuer de 2 manières :

- **passive** (Passive Open), ceci signifie que la machine accepte une connexion entrante. C'est le cas d'un serveur FTP par exemple qui va se mettre en attente de demande d'établissement de connexion de la part d'un client FTP.
- **active** (Active Open) pour demander l'établissement de la connexion.

L'initialisation d'une connexion se fait toujours par ce qui s'appelle une "**Poignée de main à 3 voies**" qui est la traduction littérale de "**Three Way Handshake**", cette initialisation se déroule donc en 3 étapes. Ces 3 étapes ont pour but essentiel de synchroniser les numéros de séquence des 2 machines :

- La machine A envoie un segment de type "ouverture de connexion" avec le n° de séquence X (dans ce segment ne figure aucune donnée)
- La machine B renvoie un segment de type "ouverture de connexion" avec le n° de séquence Y et en acquittant la séquence X envoyée par A
- La machine A renvoie un acquittement à B du segment n° Y

De cette façon chaque machine connaît le n° de séquence de l'autre et l'échange d'information peut débuter.

1.4. Structure des segments TCP

Le segment TCP c'est l'unité de transfert du protocole TCP, il est utilisé indifféremment pour établir les connexions, transférer les données, émettre des acquittements, fermer les connexions.

De façon classique, la structure d'un segment TCP comprend un **entête** de taille variable qui utilise un format en mot de 32bits suivi d'une zone de données.

<table>
<tr><td colspan="4">Port source (16 bits)</td><td colspan="3">Port destination (16 bits)</td></tr>
<tr><td colspan="7">Sequence number (4 octets)</td></tr>
<tr><td colspan="7">Ack number</td></tr>
<tr><td rowspan="2">Data offset
(4bits)</td><td rowspan="2">Réservé
(6 bits)</td><td>U</td><td>A</td><td>P</td><td>R</td><td>S</td><td>F</td><td rowspan="2">Window
(16 bits)</td></tr>
<tr><td>R</td><td>C</td><td>S</td><td>S</td><td>Y</td><td>I</td></tr>
<tr><td></td><td></td><td>G</td><td>K</td><td>H</td><td>T</td><td>N</td><td>N</td><td></td></tr>
<tr><td colspan="2"></td><td colspan="6">(6 bits)</td></tr>
<tr><td colspan="4">Checksum</td><td colspan="3">Pointeur urgent</td></tr>
<tr><td colspan="7">Options</td></tr>
</table>

- **Source Port et Destination Port (2 x 16 bits)** Ces deux champs de 16 bits chacun contiennent les **numéros de port** de la source et de la destination. Certains numéros de ports sont dédiés à un protocole particulier (par exemple le port 80 est dédié à http).
- **Sequence Number (32 bits)** Ce n° sur 32 bits correspond au **numéro de séquence du premier octet** de données de ce segment de données, en effet le protocole TCP numérote chaque octet envoyé. Si le drapeau SYN vaut 1, ce champ définit le numéro de séquence initial (ISN).
- **Acknowledgment Number (32 bits)** Ce champ sert lorsque le segment est un **segment d'acquittement** (le drapeau **ACK** du champ Flags est à 1), il indique le numéro de séquence du prochain octet attendu (c'est à dire le n° de séquence du dernier octet reçu + 1), tous les octets précédents cumulés sont implicitement acquittés.
- **Data Offset (4 bits)** Ce champ donne la **taille en mots de 32 bits de l'entête du segment**. Si le champ Options est vide, cette taille est égale à 5 (entête de 20 octets).
- **Flags (6 bits)** Ce champ comprend 6 drapeaux qui indique le rôle du segment TCP :
  - ✓ ACK : Indique un segment d'acquittement
  - ✓ SYN : Ouverture de la connexion
  - ✓ FIN : Fermeture de la connexion
  - ✓ RST : Réinitialisation de la connexion pour cause d'erreurs non récupérables
  - ✓ PSH : Demande de remise immédiate des données au processus de la couche supérieure
  - ✓ URG : Données urgentes
- **Window (16 bits) Taille de la fenêtre**, c'est à dire le nombre d'octets que le récepteur est en mesure d'accepter à partir du numéro d'acquittement.
- **Checksum (16 bits)** Le Checksum permet de contrôler si le paquet TCP n'a pas été modifié lors de son transport.

- **Urgent Pointer (16 bits)** Donne la position d'une **donnée urgente** en donnant son décalage par rapport au numéro de séquence. Ce champ n'est utilisé que si le drapeau URG est positionné. Les données urgentes devront passer devant la file d'attente du récepteur, c'est par exemple avec ce mécanisme qu'il est possible d'envoyer des commandes d'interruption au programme Telnet.
- **Options (variable)** Utilisé à des fonctions de test.
- **Padding (variable)** Octets de bourrage qui permettent de terminer l'en-tête TCP.

1.5. Numéros de port usuels

N° Port	Mot-clé	Description
21	FTP	File Transfer (Control)
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
80	HTTP	WWW
110	POP3	Post Office Protocol - version 3

2. Le protocole UDP

2.1. Caractéristiques

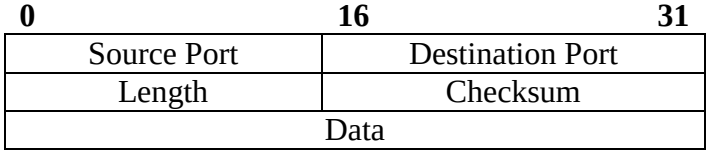
Le protocole UDP est une alternative au protocole TCP. Comme TCP, il intervient au dessus de la couche IP, au niveau **Transport** au sens des couches OSI.

Les caractéristiques du protocole UDP sont les suivantes :

- identifie les processus d'application à l'aide de **numéros de ports UDP** (distincts des numéros de port TCP)
- possède un **contrôle d'erreurs assez rudimentaire**, il est donc destiné aux réseaux fiables
- UDP est un protocole qui n'est **pas orienté connexion**
- peut éventuellement vérifier l'**intégrité** des données transportées
- Les données ne sont **pas séquencées** donc rien ne permet de vérifier que l'ordre d'arrivée des données et le même que celui d'émission. Ceci le destine plutôt aux réseaux locaux où le mode d'acheminement des informations ne risque pas d'inverser l'ordre des données mais également aux applications qui véhiculent des informations de petites tailles qui peuvent tenir en un seul datagramme.
- De par sa structure UDP est **plus rapide** que TCP, mais **moins robuste**

UDP est donc un **protocole orienté commande/réponse.** UDP peut être utile pour les applications qui nécessitent une **diffusion** d'informations car dans ce cas il serait pénalisant d'utiliser un protocole comme TCP orienté connexion qui devrait gérer (ouvrir et fermer) autant de connexion que de nœuds auxquels l'information est destinée.

2.1. Structure du paquet UDP



- **Source Port et Destination Port** : port source et destination
- **Length** : Longueur du paquet UDP
- **Checksum** :champ de contrôle des données.