

Réseaux : modèles, protocoles, programmation

Pablo Rauzy

`pablo.rauzy@univ-paris8.fr`

`pablo.rauzy.name/teaching/rmpp`



UFR MITSIC / L2 informatique

Séance 5

La couche transport

La couche transport

- ▶ La couche transport est la quatrième couche du modèle OSI.
- ▶ Elle est en charge des communications de bout en bout entre processus.
- ▶ C'est généralement la plus haute couche où on se préoccupe des erreurs de transmissions.
- ▶ Le service offert par cette couche est donc une abstraction complète des services réseaux fournis par les couches du dessous : au dessus, on transfère simplement des octets garantis sans corruptions ou pertes.

- ▶ Il existe quelques protocoles au niveau transport.
- ▶ Les deux principaux (et les seuls importants) sont incontestablement UDP et TCP.

- ▶ UDP signifie *User Datagram Protocol*.
- ▶ Son rôle est de permettre à moindre coût la transmission de données entre deux processus défini par une adresse IP et un numéro de port.
- ▶ Il ne fournit aucune garantie sur la livraison des datagrammes, ni sur leur ordre d'arrivée.

- ▶ UDP est utilisé essentiellement dans deux cas :
 - pour transmettre des petites quantités de données,
 - pour transmettre rapidement beaucoup de données, dans les cas où perdre un datagramme de temps en temps n'est pas grave.
- ▶ Le premier cas d'utilisation typique est le DNS.
- ▶ Les seconds cas d'utilisation sont le streaming, la VoIP, ou les jeux en ligne.

- Les entêtes d'UDP sont réduits au strict minimum :
- le port source,
 - le port destination,
 - longueur totale du datagramme (entêtes+données),
 - somme de contrôle.

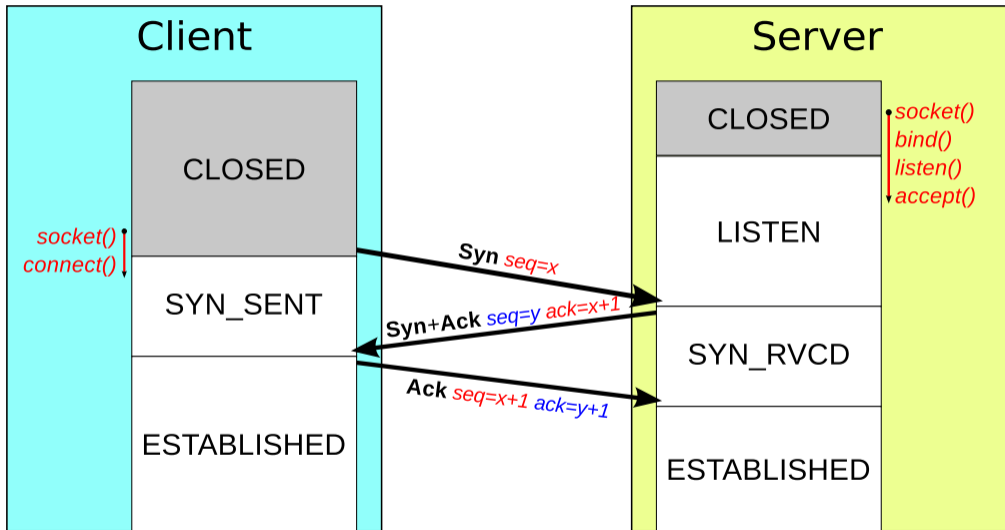
Port Source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données (longueur variable)	

- Elle est calculée non seulement sur les entêtes UDP, mais aussi sur :
- les données,
 - certains des entêtes IP.

+	Bits 0 - 7	8 - 15	16 - 23	24 - 31
-96	Adresse Source			
-64	Adresse Destination			
-32	Zéros	Protocole	Taille UDP	
0	Port Source		Port Destination	
32	Longueur		Somme de contrôle	
64	Données			

- ▶ TCP signifie *Transmission Control Protocol*.
- ▶ C'est un protocole de transport *fiable* en *mode connecté*.
- ▶ Fiable, cela signifie que TCP garanti aux couches d'au dessus que :
 - les *segments* arrivent tous à destination,
 - les segments arrivent dans l'ordre à destination,
 - les données échangées sont intègres.
- ▶ Une *session* TCP se passe en 3 étapes :
 - l'ouverture de la connexion,
 - l'échange de données,
 - la fermeture de la connexion.

- ▶ Dans TCP, la connexion est ouverte par un *handshake* en trois temps entre le client et le serveur.
- ▶ Le but de cet échange est d'établir des numéros de séquence qui permettront de garantir l'ordre et la complétude de l'ensemble des segments transmis.
- ▶ L'ouverture se passe en 3 étapes :
 - Le client envoie un segment SYN au serveur avec un numéro de séquence aléatoire A.
 - Le serveur répond au client avec un segment SYN-ACK.
Il contient un numéro d'acquittement qui vaut A+1.
Il contient aussi un numéro de séquence aléatoire B.
 - Le client répond au serveur avec un segment ACK.
Le numéro de séquence du segment vaut A+1.
Le numéro d'acquittement vaut B+1.



- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.
- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.
- ▶ À chaque transfert de segment :

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.

- ▶ À chaque transfert de segment :
 - l'émetteur envoie :
 - numéro de séquence : initial+nombre d'octets envoyés,
 - numéro d'acquittement : prochain numéro de séquence attendu du destinataire ;

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.

- ▶ À chaque transfert de segment :
 - l'émetteur envoie :
 - numéro de séquence : initial+nombre d'octets envoyés,
 - numéro d'acquittement : prochain numéro de séquence attendu du destinataire ;
 - le destinataire répond avec un segment ACK :
 - numéro de séquence : numéro d'acquittement du segment reçu,
 - numéro d'acquittement : numéro de séquence reçu augmenté du nombre d'octets reçus ;

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.

- ▶ À chaque transfert de segment :
 - l'émetteur envoie :
 - numéro de séquence : initial+nombre d'octets envoyés,
 - numéro d'acquittement : prochain numéro de séquence attendu du destinataire ;
 - le destinataire répond avec un segment ACK :
 - numéro de séquence : numéro d'acquittement du segment reçu,
 - numéro d'acquittement : numéro de séquence reçu augmenté du nombre d'octets reçus ;
 - si au bout d'un temps limite (qui est adapté au fur et à mesure des échanges) l'émetteur ne reçoit pas d'ACK, il renvoie le segment,

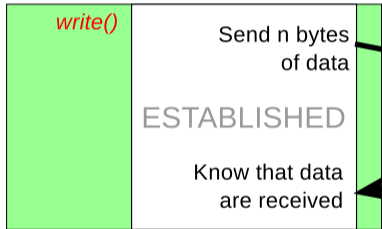
- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.

- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.

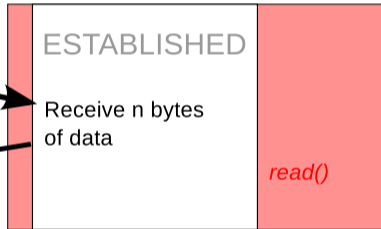
- ▶ À chaque transfert de segment :
 - l'émetteur envoie :
 - numéro de séquence : initial+nombre d'octets envoyés,
 - numéro d'acquittement : prochain numéro de séquence attendu du destinataire ;
 - le destinataire répond avec un segment ACK :
 - numéro de séquence : numéro d'acquittement du segment reçu,
 - numéro d'acquittement : numéro de séquence reçu augmenté du nombre d'octets reçus ;
 - si au bout d'un temps limite (qui est adapté au fur et à mesure des échanges) l'émetteur ne reçoit pas d'ACK, il renvoie le segment,
 - si le destinataire reçoit deux fois le même segment il s'en rend compte grâce au numéro de séquence et ignore le second,

- ▶ Pendant l'échange de données :
 - les sommes de contrôle sont utilisées pour vérifier l'intégrité des segments,
 - les numéros de séquence et acquittement pour vérifier que l'ensemble des données est bien arrivé et dans l'ordre.
- ▶ Comme pour UDP la somme de contrôle est calculée sur les entêtes et les données.
- ▶ À chaque transfert de segment :
 - l'émetteur envoie :
 - numéro de séquence : initial+nombre d'octets envoyés,
 - numéro d'acquittement : prochain numéro de séquence attendu du destinataire ;
 - le destinataire répond avec un segment ACK :
 - numéro de séquence : numéro d'acquittement du segment reçu,
 - numéro d'acquittement : numéro de séquence reçu augmenté du nombre d'octets reçus ;
 - si au bout d'un temps limite (qui est adapté au fur et à mesure des échanges) l'émetteur ne reçoit pas d'ACK, il renvoie le segment,
 - si le destinataire reçoit deux fois le même segment il s'en rend compte grâce au numéro de séquence et ignore le second,
 - dans chaque segment est indiqué une taille de buffer disponible chez son émetteur, qui signale combien il peut recevoir d'octet avant d'envoyer un ACK.

Computer A



Computer B



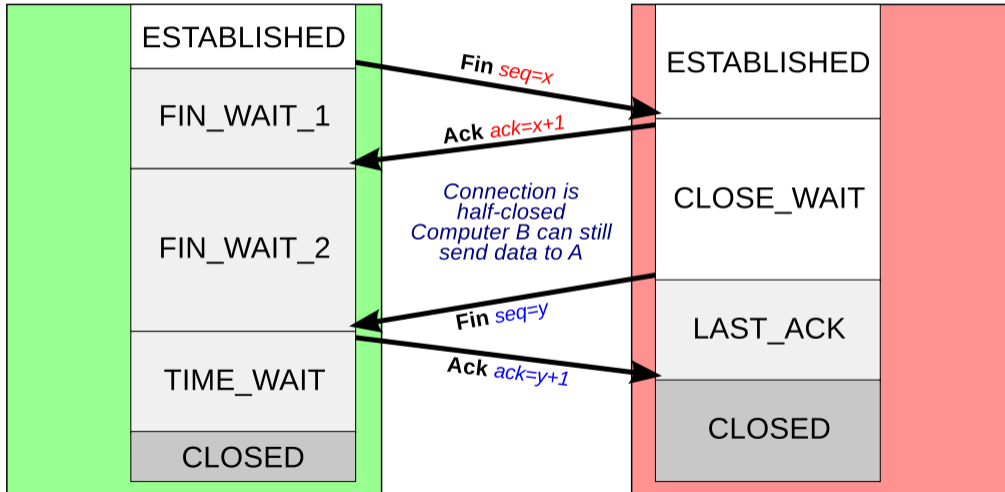
Data *seq=x* *ack=y*

Ack *seq=y* *ack=x+n*

- ▶ La fermeture de connexion se fait indépendamment de chaque côté mais avec confirmation, donc en quatre temps en tout.
- ▶ D'abord un des deux participants A ferme sa connexion vers l'autre B :
 - A envoie un segment FIN avec son numéro de séquence courant X,
 - B répond un segment ACK avec comme numéro d'acquittement X+1.
- ▶ Ensuite, B ferme sa connexion vers A :
 - B envoie un segment FIN avec son numéro de séquence courant Y.
 - A répond un segment ACK avec comme numéro d'acquittement Y+1.

Computer A

Computer B



Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset	Reserved 0 0 0	N S	C W R	E C R	U R C	A S S	P S S	R S Y	F I N	Window Size																					
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																															
...																															

- ▶ Programmons ensemble un petit *echo server*...