INTERNET – COUCHE TRANSPORT (TCP / UDP)

Réseaux – Fiche 4

I. Notion de ports

- **Port serveur :** Identifie une application sur le serveur (1 à 65535)
- **Port client :** Port sur lequel le client attend la réponse (dynamique/éphémère)

II. Protocole UDP

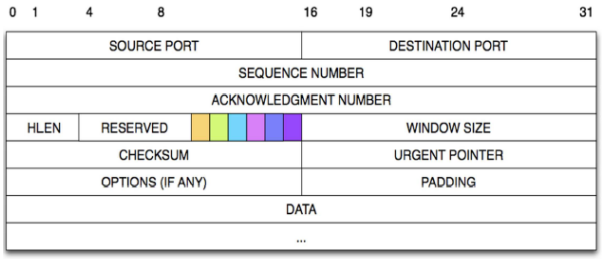
- Sans connexion, sans acquittement, rapide mais non fiable

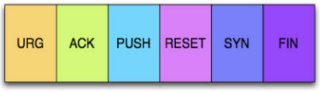
Port source	Port destination
Longueur	Checksum
Données	

- Checksum calculé sur une « pseudo en-tête » (IP source et dest., protocole, long.) et données

III. Protocole TCP

Avec connexion et acquittement



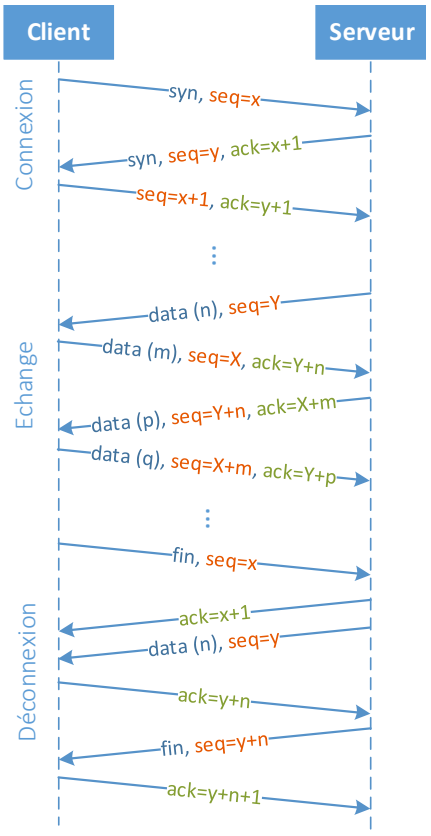


1. Datagramme

- **Seq number :** n° de séquence
- **Ack number :** n° du prochain octet attendu
- **Hlen :** Lg en-tête / 4
- **Options :**
  - URG : Urgent
  - ACK : Acquittement du *ack number*
  - PUSH : remonter directement les données à la couche supérieure
  - RESET : reset connexion
  - SYN : demande de connexion
  - FIN : demande de déconnexion
- **Window size :** Nb octet que l'on peut transmettre
- **Checksum, Urgent pointer, options**

2. Connexion, échange de données, déconnexion

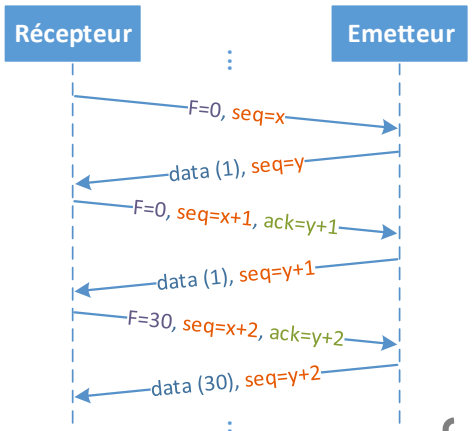
Voir le schéma ci-contre.



3. Contrôle de flux

La taille de fenêtre indique le nombre d'octets que l'on peut encore recevoir actuellement.

Une fenêtre à 0 invite l'émetteur à stopper l'émission. Pour maintenir la connexion, l'émetteur émet régulièrement 1 octet.



4. Bufferisation

On met les données en buffer et on les envoie en blocs, sauf en cas de besoin d'interactivité, auquel cas on utilise l'option PUSH et on envoie les données directement.

IV. Programmation de sockets

Socket : Moyen de communication entre un client (prend l'initiative) et un serveur (attend une demande), via un réseau.

1. Création du socket

`socket(domaine, type, protocole)`	Création du socket
`bind(socket, adresse, lgAdr)`	Spécification de l'adresse (pour le serveur)

2. Envoi du message en UDP

`sendto(socket, message, long, flags, adressDest, lgAdr)`	Envoi de message
`recvfrom(socket, *buffer, lgMax, flags, *adresseSrc, *lgAdr)`	Reception de message

3. Envoi de message en TCP

`connect(socket, adresse, lgAdr)`	Connexion client au serveur
`listen(socket, lgFileAttente)`	Ecoute les connexion serveur
`accept(socket, * adresseSrc, *lgAdr)`	Accepte une connexion serveur
`read(socket, *buffer, lgMax)`	Lit un message
`write(socket, message, long)`	Envoie un message
`close(socket)`	Ferme la connexion

4. Schéma des appels de fonctions