Cours 5 : la couche transport - protocoles TCP et UDP -

Module R4 : Technologie IP

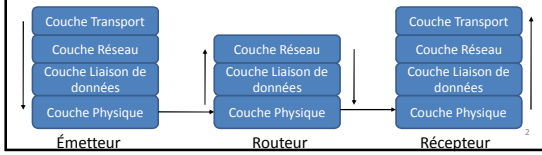
IUT R&T 1ère année

David Mercier

La couche transport

Présentation

- **Rôle** : fournir à l'utilisateur (couche application) un **service de transport** (transfert d'informations d'une machine source à une machine destination) fiable, efficace et économique.
- Propose une liaison en **mode connecté** ou **non connecté**.
- Est la première de **bout en bout** :



Couche transport dans l'Internet

- Internet utilise principalement 2 protocoles pour la couche transport :
  - **un protocole orienté connexion : TCP**
    - Garantit une certaine qualité de service.
  - **un protocole sans connexion : UDP**
    - Simple, temps d'exécution très rapide.

TCP (RFC 793) Transmission Control Protocol

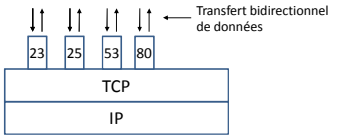
- Protocole orienté connexion.
- TCP offre un service de transport **fiable**.
- Données échangées = flot de bits divisés en octets devant être reçus dans l'ordre où ils sont envoyés.
- Transfert de données TCP :
  1. Phase d'établissement connexion
  2. Phase de transfert
  3. Phase de fermeture

Autres propriétés du service offerts par TCP

- L'application choisit la taille de ses données. TCP découpe les données en paquets de la taille qui lui convient (**segments**).
- Pour rendre le transfert plus performant, l'implantation de TCP attend qu'il y ait suffisamment de données pour remplir un datagramme.
- **Connexions bidirectionnelles simultanées** : 2 flots indépendants de sens contraire, sans interaction apparente.
- TCP garantit la remise d'un flot de données sans perte ni duplication (**fiabilité**).
  - Utilisation de la technique de l'acquittement positif avec retransmission (cf cours R2).

Connexions de plusieurs applications sur une même adresse IP

- TCP autorise l'établissement et le multiplexage de plusieurs connexions sur une même interface réseau.
- Pour cela il utilise la notion de **port**



- **Définition** : « adresse IP : numéro de port » = **« socket »** (ce qu'on peut traduire par « connecteur » ou « prise »)

Exemples de ports bien connus

Well-known ports : 1-1023 (serveur)

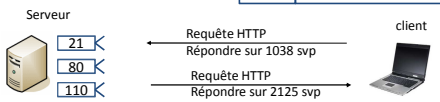
Registered ports : 1024–49151

Port	Protocole	Utilisation
20	ftp-data	Données transfert FTP
21	ftp	Contrôle transfert FTP
22	ssh	Session distante sécurisée
23	telnet	Session distante
25	smtp	Envoie de courriel
53	dns	Gestion nom de domaine
80	http	Web
110	pop-3	Réception de courriel
143	imap	Réception de courriel
179	bgp	Protocole de routage externe (cf cours 2)

Liste complète des ports bien connus et enregistrés sur le site de l'IANA
Voir aussi http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers,
page des énoncés de TP sur iut-gtr2, fichier */etc/services* sous Unix/Linux

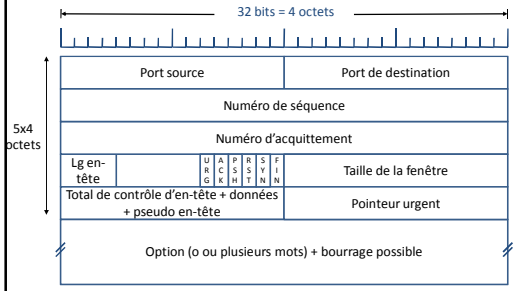
Exemples de connexions client-serveur

- **Rappel** : serveurs doivent envoyer des informations pertinentes aux clients qui en réclament.
- Problème : serveur ne sait pas quand un client va le contacter.
- Solution : serveur est à l'écoute sur un certains ports bien connus :
  - port 80 pour HTTP, le port 110 pour POP3, le port 21 pour FTP…
  - Derrière ces ports tournent en tâche de fond des petits programmes à leurs écoutes : des **daemons**
- Exemple connexion web :



Deux sessions web sont associées à deux numéros de ports différents, sinon les pages se mélangeraient !

Format en-tête segment TCP



Champs TCP

SEQ, ACK et lg en-tête

- **Numéro de séquence** (SEQ) : indique le numéro du premier octet porté par le fragment.
- **Numéro d'acquittement** (ACK) : indique le numéro du premier octet attendu (soit dernier reçu + 1).
- Ces numéros assurent la non perte et la non duplication des données.
- Mécanisme de **fenêtre glissante** : l'émetteur a la possibilité d'émettre plusieurs paquets avant de recevoir le premier ACK.
- **Longueur en-tête** : idem IPv4, exprimée en mots de 32 bits (4 octets)
  - De 5 à 15 = $(1111)_2$ soit 20 octets à 60 octets.

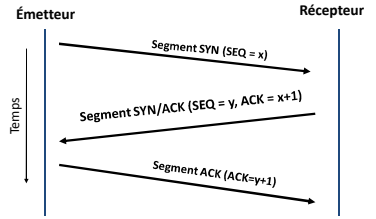
Champs TCP

Les 6 drapeaux

- Bit **URG** : indique que le segment contient des données urgentes (désignées par le champ pointeur d'urgence).
- Bit **ACK** :
  - 1 : Numéro d'acquittement est valide
  - 0 : Numéro d'acquittement ignoré (segment ne contient pas d'ACK)
- Bit **PSH** : signifie poussée (pushed), demande au destinataire de ne pas stocker les données.
- Bit **RST** : réinitialise la connexion (« Houston, we've got a problem »…).
- Bit **SYN** : sert à établir une connexion.
- Bit **FIN** : sert à libérer une connexion.

Établissement d'une connexion TCP

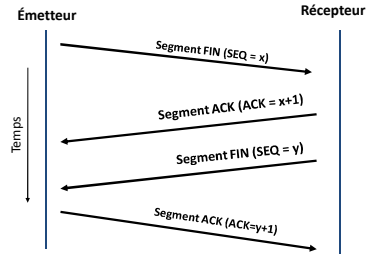
- Processus en 3 temps (three-way handshake)



La taille maximum d'un segment (**MSS**, Maxi Segment Size) fait partie des paramètres de négociation lors de l'établissement de la connexion (dans le champ option, par défaut = 536 o)

Libération d'une connexion TCP

- Processus en 3 temps modifié

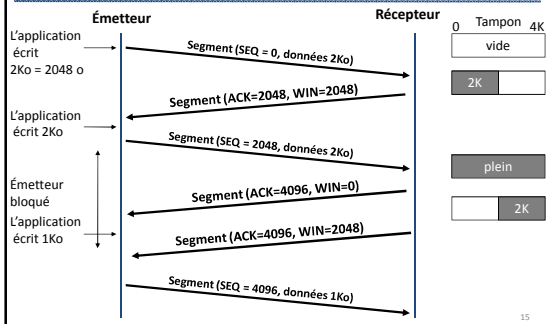


Champs TCP

Taille de fenêtre

- Le **contrôle de flux** dans TCP est réalisé au moyen de fenêtres dynamiques de tailles variables
- Champ **Taille de fenêtre** : indique combien on peut transmettre d'octets après l'octet acquitté.
- Si ce champ est nul : le récepteur ne souhaite plus recevoir de données pour l'instant.

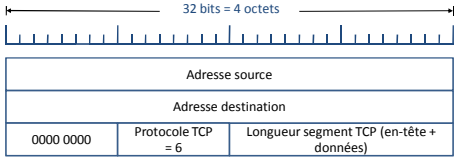
Exemple de gestion de fenêtre sous TCP



Champs TCP

Total de contrôle

- **Total de contrôle** : permet d'assurer une très grande fiabilité.
  - Porte sur en-tête + données + pseudo en-tête défini par :



Champs TCP

Pointeur urgent

- **Pointeur urgent** : indique l'emplacement de données urgentes (un octet particulier).
- Plus précisément :
  - bit URG étant positionné à 1 ;
  - pointeur urgent indique le décalage en octets à partir du numéro de séquence pour indiquer où se trouve les données urgentes

Contrôle de congestion dans TCP

- Pour ce contrôle, chaque émetteur gère deux fenêtres :
  - la fenêtre que le récepteur à accorder (cf champ fenêtre précédent) ;
  - Une **fenêtre de congestion**.
- Chacune de ces fenêtres reflète le nombre d'octets que l'émetteur peut envoyer (le minimum des deux).
- Un **timer de retransmission** est aussi utilisé (il sera vu en TD).

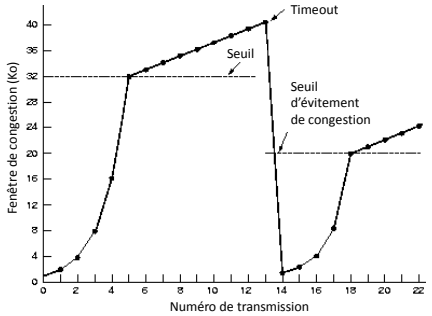
Algorithme de congestion d'Internet

- Principe :
  - À l'établissement de la connexion, l'émetteur initialise la fenêtre de congestion à la taille maximale de segment.
  - Démarrage lent (**slow start**), en fait il n'est pas lent du tout ! (exponentiel) : tant que les segments sont acquittés avant expiration du timer, la taille de la fenêtre de congestion est multipliée par deux.
  - Arrivé à un certain **seuil d'évitement de congestion**, la taille de la fenêtre de congestion croît linéairement .
  - Quand un timer expire :
    - le seuil d'évitement est fixé à la moitié de la taille de la fenêtre de congestion courante.
    - la taille de la fenêtre de congestion est réinitialisée à la taille maximale de segment (MSS), et à nouveau l'algorithme slow start est employé jusqu'au seuil d'évitement.

19

Exemple d'algorithme de congestion de l'Internet

cf Tanenbaum section 6.5.9



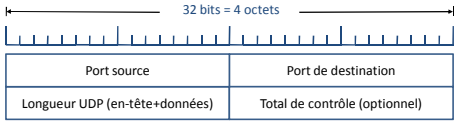
20

UDP (RFC 768) User Datagram Protocol

- Protocole de la couche transport non fiable sans connexion.
- Ne fait pas :
  - Contrôle de flux
  - Contrôle de congestion
  - Contrôle d'erreur / retransmission (couche application doit s'en occuper)
- Fait :
  - Utilise le protocole IP pour acheminer les messages entre machines avec une fonction de multiplexage des différents processus via l'utilisation des ports et c'est tout…

21

Format en-tête segment UDP



En Résumé, UDP :

Utilise simplement la notion de port pour distinguer les différents services sur une machine.
Aucun contrôle (perte, duplication, …)
Protocole très simple ⇒ temps d'exécution très rapide

22

Exercices

- Quels services utilisent plutôt TCP ?
- Quels services utilisent plutôt UDP ?
- Pourquoi le protocole DNS est-il principalement encapsulé dans de l'UDP ?
- UDP et TCP utilisent des numéros de ports pour identifier l'entité de destination lorsqu'ils livrent un message. Pourquoi ces deux protocoles ont-ils générés un nouvel identifiant abstrait (le numéro de port) alors qu'il était possible d'exploiter les identifiants de processus qui existaient déjà à l'époque de la conception de ces deux protocoles ?

23