TCP: TRANSPORT CONTROL PROTOCOL

Licence Pro ESSIR

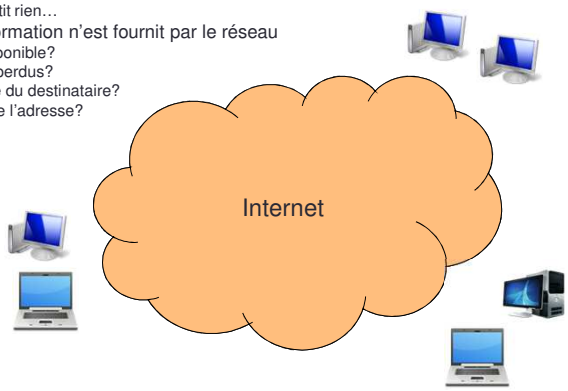
Introduction

Historique

- Réseaux X25
  - Orienté circuit (établissement de circuit pour chaque communication)
  - Fiabilisation assuré par le réseau
- Débit très faible
- Tâches très lourdes
  - Justifié par un taux de pertes important

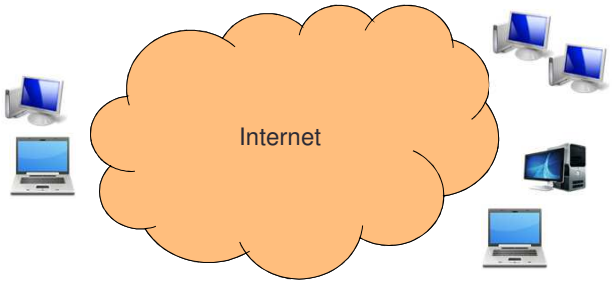
Ce que fait l'Internet

- Services offert par le réseau:
  - Acheminement des paquets
  - Rien n'est garantit (*best effort*)
    - Pas de garantit de « livraison » (fiabilité)
    - Pas de garantit de débit
    - Ne garantit rien…
  - Aucune information n'est fournit par le réseau
    - Débit disponible?
    - Paquets perdus?
    - Existence du destinataire?
    - Validité de l'adresse?
    - Etc.



La complexité déplacée

- L'acheminement est une tâche simple
- Les tâches complexes sont:
  - La fiabilisation
  - Les contrôles des débits des sources
  - L'établissement des circuits/connexions
- Internet:
  - Les tâches complexes sont déplacées en bordure du réseau
  - Sur les PCs! A la charge des systèmes d'exploitations.
  - Simplification à l'extrême des tâches opérées dans le réseau

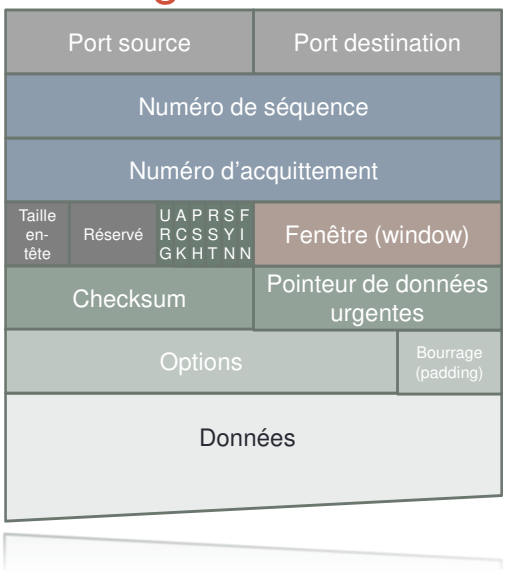


TCP: format des en-têtes / encapsulation / implémentation

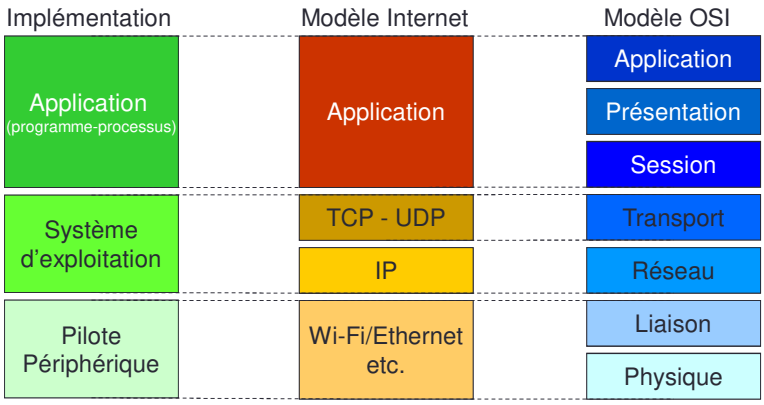
Encapsulation

- L'en-tête TCP et les données sont encapsulés dans un paquet IP.
- L'en-tête TCP fait 20 octets sans les options

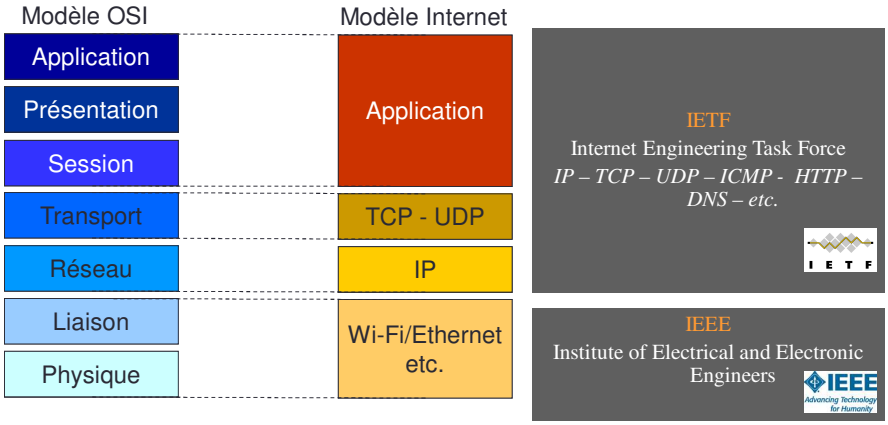
Format des segments



Modèle OSI - Implémentation



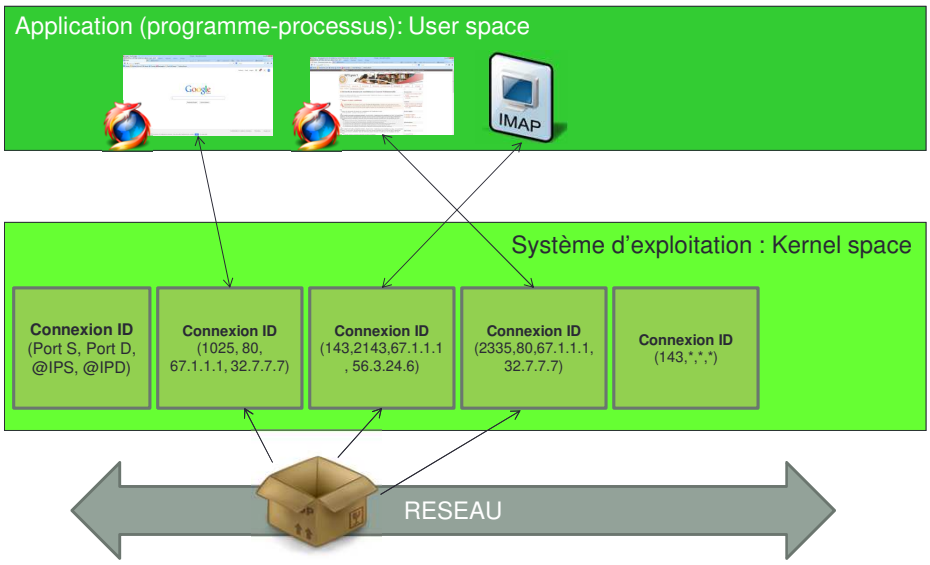
Les instances de normalisation dans l'Internet



Les numéros de ports

- Entiers codés sur 2 octets (<65536)
- Identifie la connexion (client/serveur)
- Du point de vue du client:
  - N° port destination
    - Port < 1024 (sauf exception)
    - Identifie le service
    - Association statique (alloué par l'IANA)
      - Exemple: http=80, telnet=23, DNS=53, ftp=21, imap=143, etc.
  - N° port source
    - Choisit localement par le système d'exploitation
    - Port > 1024
    - Une implémentation possible: compteur incrémenter de 1à chaque connexion.
- Du point de vue du serveur
  - C'est l'inverse:
    - N° port destination: port choisit par le client
    - N° port source: port identifiant le service

Implémentation: aiguillage



Codes en C

Code du serveur

```
if(setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&tr,sizeof(int)) == -1) erreur("Erreur setsockopt() SO_REUSEADDR",2);

if(bind(sockfd, (struct sockaddr *) res->ai_addr, res->ai_addrlen)<0) erreur("Erreur bind()",2);

if(listen(sockfd, SOMAXCONN)<0) erreur("Erreur listen()",2);

size=sizeof(clientAddr);//N'Ã©tait pas nÃ©cessaire en IPv4
while((confd=accept(sockfd, (struct sockaddr *) &clientAddr, (socklen_t *) &size))>0)
{
```

Code du client

```
if((sockfd=socket(res->ai_family, res->ai_socktype, res->ai_protocol))<0) erreur("Erreur socket():",2);

if(connect(sockfd,res->ai_addr,res->ai_addrlen)<0) erreur("Erreur connect()",2);

if(send(sockfd,requete,strlen(requete),0)<0) erreur("Erreur send()",2);
```

Exercice 1: numéro de port

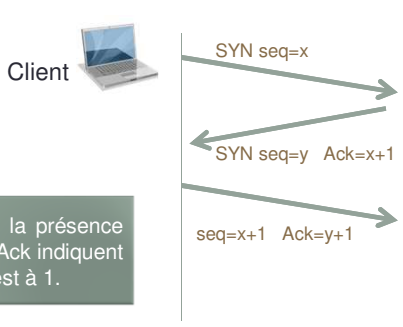
- Un client d'adresse IP 12.1.1.1 a 3 connexions sur le même serveur web d'adresse IP 175.1.1.1.
  - Quels seront les numéros de port source et destination de ces 3 connexions?
  - Au niveau du serveur,si le numéro de port source est déjà utilisé par un autre client, que se passe-t-il?
  - Un client peut-il utiliser le même numéro de port source pour différentes connexions?
  - Donnez les identifiants complets de ces connexions

TCP:

- Ouverture des connexions
- Fiabilisation
- Fermeture des connexions

Ouverture de connexion

- Emission de segments avec le flag SYN à 1
- Permet de vérifier que le serveur
  - existe bien,
  - est allumé,
  - Est prêt à recevoir/envoyer des données sur un port donné (**le port est dit ouvert**)

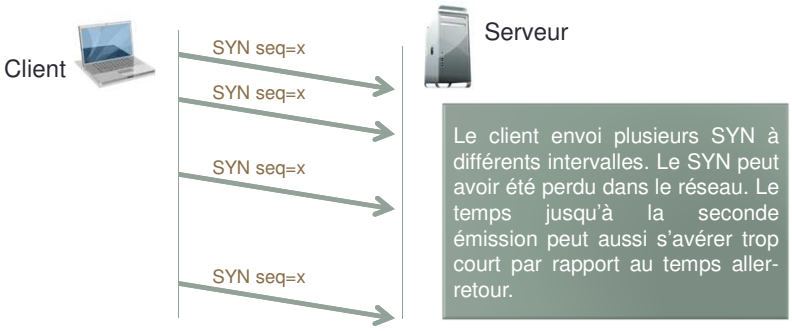


Convention: la présence de SYN ou Ack indiquent que le flag est à 1.

Les numéros de séquences sont fixés localement de part et d'autres. Ils sont échangés lors de l'ouverture de la connexion TCP. Ils serviront à numéroter les segments. Ils ne commencent pas à 0 pour éviter que l'insertion de faux segments soit trop facile.

Ouverture de connexion: le serveur ne répond pas

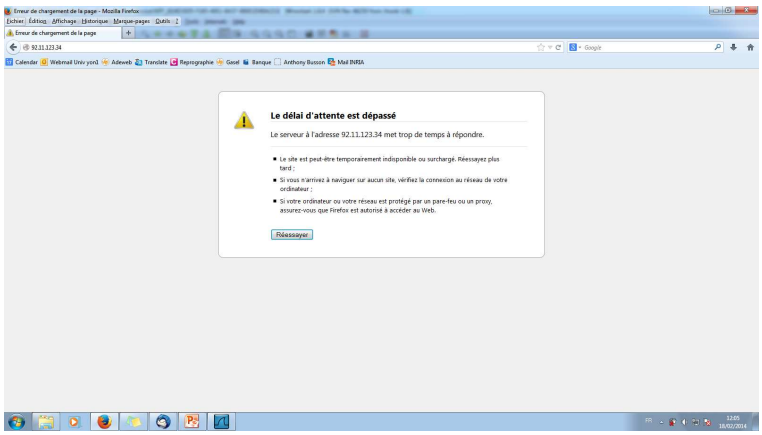
- Si le serveur ne répond pas:
  - L'adresse IP est invalide
  - Les paquets sont filtrés
  - Il n'y a pas de routes
  - Etc.
- Emission de plusieurs SYN à intervalles croissants



Le client envoi plusieurs SYN à différents intervalles. Le SYN peut avoir été perdu dans le réseau. Le temps jusqu'à la seconde émission peut aussi s'avérer trop court par rapport au temps aller-retour.

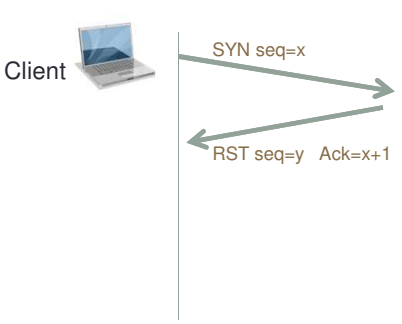
Exemple

- L'adresse dans votre navigateur n'est pas bonne.



Ouverture de connexion: le port est fermé

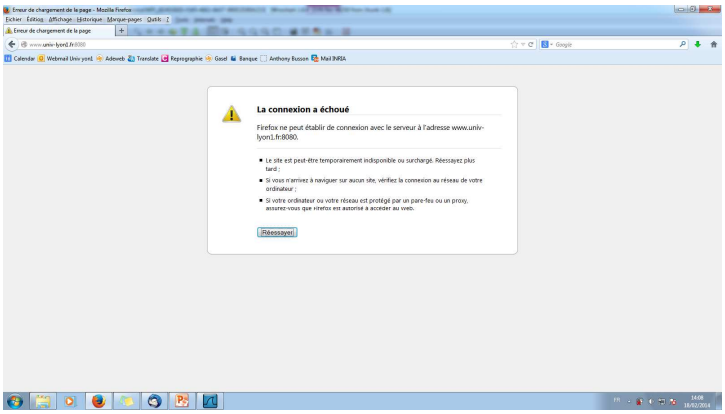
- Si le port est fermé:
  - Emission d'un TCP RST du serveur
  - Annulation de la connexion



Les numéros de séquences sont fixés localement de part et d'autres. Ils sont échangés lors de l'ouverture de la connexion TCP. Ils serviront à numéroter les segments. Ils ne commencent pas à 0 pour éviter que l'insertion de faux segments soit trop facile.

Exemple

- La destination est présente, mais le port est fermé.



TCP: fiabilisation

- But:
  - S'assurer que les données transmises sont toutes bien reçues
    - Sinon les retransmettre
  - Remise en ordre
- Mise en œuvre :
  - chaque segment est numéroté
  - On acquitte les segments reçus
  - Un segment non acquitté est retransmis
- Détails vus plus loin
  - Le système de numérotation
  - Le système d'acquittement
  - La procédure de retransmission

TCP: numérotation des segments

Ce sont les octets qui sont numérotés. Le numéro d'un segment correspond au numéro du premier octet transporté.

Le numéro du premier octet est fixé lors de l'ouverture de connexions pour chaque sens de transmission.

Pour un segment de n octets, les octets seront de numérotés de x (numéro du ségment) à n+x-1. Le segment suivant portera le numéro n+x.

Exemple: 2 segments TCP émis consécutivement.

Seq=1347

Segment de 300 octets (payload) numérotés de 1347 à 1646.

Seq=1647

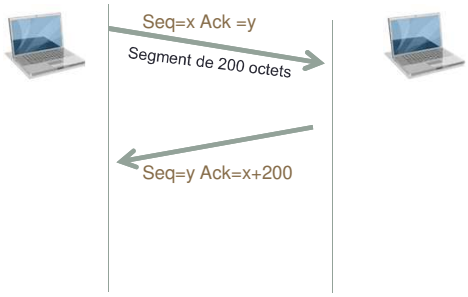
Segment suivant de 210 octets numérotés de 1647 à 1846.

TCP: numérotation des acquittements

Les segments reçus sont acquittés. Lors de la réponse, le flag ACK est mis à 1. Le champ ack correspond au numéro de l'octet attendu en réception.

Un segment de numéro de séquence x contenant n octets sera acquitté avec un ack valant n+x.

Les acquittements peuvent contenir des données.

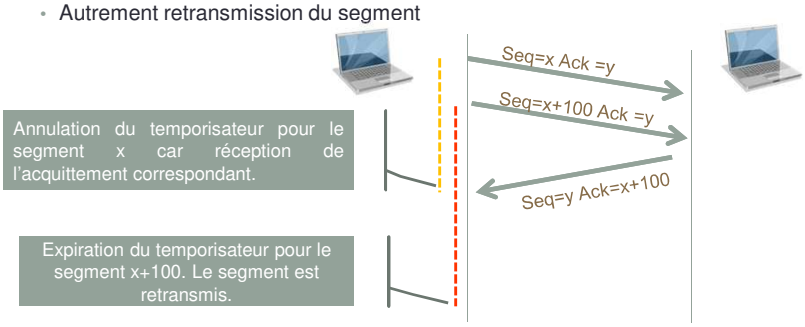


Détection d'une perte

- Quand est ce qu'un segment est considéré perdu
  - Pas de ack (à l'expiration d'un temporisateur)
  - Acquittements dupliqués (transparents suivants)

C'est la source qui détecte les pertes.

- Perte détecté à l'expiration d'un temporisateur
  - Déclenchement d'un temporisateur à chaque émission de segments
  - Il y a un temporisateur différent pour chaque segment
  - Si un ACK est reçu avant expiration, le temporisateur est annulé
  - Autrement retransmission du segment

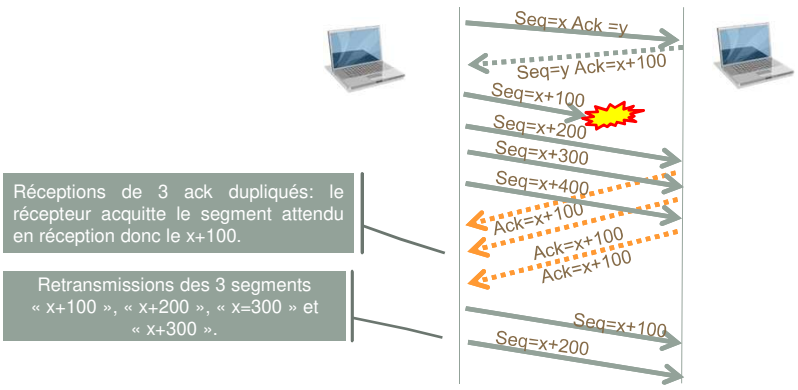


Valeur des timers de détection des pertes

- Temps aller-retour inconnu
  - Dépend de la distance source-destination
  - De l'état de congestion du réseau
  - Varie dans le temps
- Algorithme
  - Mis à jour d'une variable RTT (Round Trip Time)
  - Valeur fixe pour la première transmission
  - Mise à jour (lissage) en fonction des mesures:
    - RTT= β RTT + (1- β) Mesure
  - Le timer de retransmission est RTO=2*RTT (RTO: Retransmission Time Out)

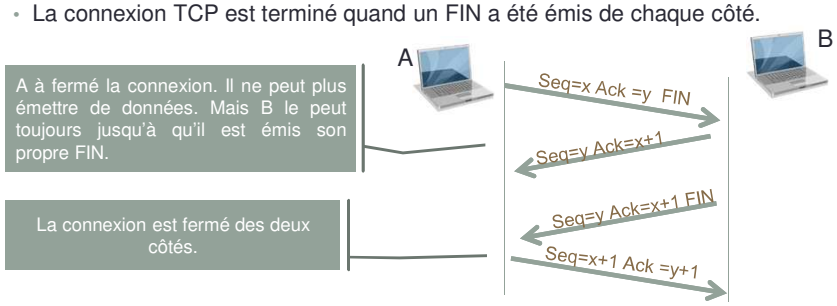
Détection d'une perte: acquittements dupliqués

- Seules les données en séquence sont acquittés.
- Si des données arrivent hors séquences, le segment attendu en réception est acquitté (pour la 2ème, ou nième fois).
- Perte détecté sur acquittement dupliqué
  - Si le même acquittement est reçu 4 fois (3 ack dupliqué) les segments suivants sont renvoyés.



Fermeture de connexion

- La communication étant bidirectionnelle, la fermeture a lieu des deux côtés.
- Emission d'un segment avec la flag FIN pour fermer la connexion dans un sens
- Acquittement du FIN
- La connexion TCP est terminé quand un FIN a été émis de chaque côté.



Note 1: le deuxième segment peut acquitter et fermer la connexion en même temps. Il n'y a alors que 3 segments échangés pour fermer la connexion TCP.

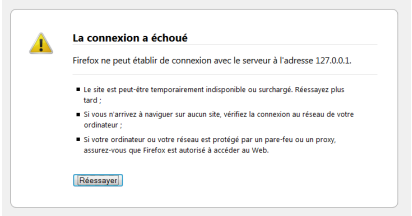
Note 2: le système d'exploitation garde trace de la connexion quelques minutes après la fermeture bidirectionnelle. Cela permet de traiter les paquets qui auraient retardés dans le réseau. Il s'agit de l'état FIN WAIT.

Exercice 2: questions diverses

1. Peut-on avoir plusieurs services (http, ftp, mail, etc.) sur un même serveur physique?
2. En quoi consiste en pratique ces services?
3. Lorsqu'un client se connecte (via TCP) à ce serveur, le système d'exploitation accepte-t-il la connexion? Pourquoi et en fonction de quoi?
4. Une fois la connexion ouverte, comment le système d'exploitation associe-t-il les données au bon processus?
5. A quoi sert d'ouvrir la connexion?

Exercice 3: quel peut-être le problème?

Messages d'erreurs affichés par firefox



Console Linux

```
linux-prompt$ telnet 123.11.3.4.5 21
Error: connexion refused by server
linux-prompt$ telnet 123.11.3.4.5 22
Welcome on 123.11.3.4.5 ssh server
login:…
```

TCP: contrôle du débit (flot / congestion)

TCP: débit des sources

- Le système d'exploitation (au travers de TCP) doit fixer le débit:
  - Rappel: aucune info (même sur la liaison locale parfois : ADSL par ex.)
  - 2 contrôles de débits à effectuer:
    - Contrôle de flot (mémoires du récepteur)
    - Contrôle de congestion (capacité du réseau)

Contrôle de flot

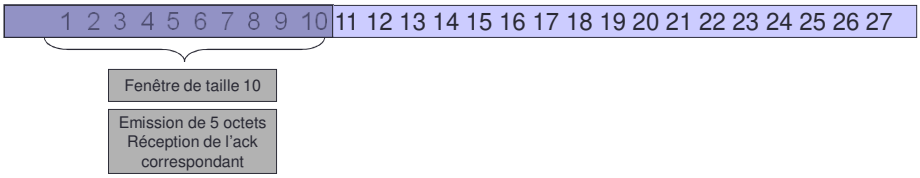
- Solution: Indiqué l'état de la mémoire à chaque émission (TCP).
  - Un champ TCP est dédié: window
  - Le nombre d'octets émis sans ack par une source ne peut dépasser la valeur du champ window
- Fenêtre glissante si on connaît la taille du buffer en réception

Contrôle de congestion

- Solution:
  - La seule information disponible est le nombre de pertes
  - Hypothèses:
    - perte = congestion
    - pas de perte = pas de congestion
  - Augmentation du débit quand il n'y a pas de pertes (vitesse?)
  - Diminution du débit en cas de pertes

Contrôle de congestion (2)

- Utilisation d'une fenêtre de congestion
  - Indique les octets qui peuvent être émis
  - Fenêtre glissante

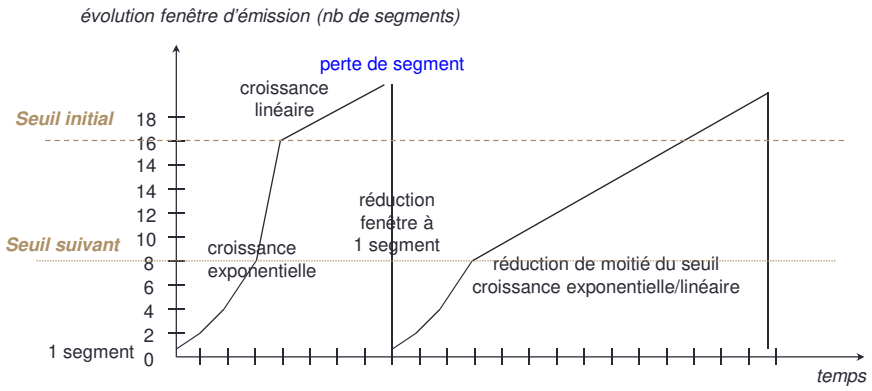


- La fenêtre est déplacé du nombre d'octets acquittés,
- Plus la fenêtre est grande/large plus le débit sera important
- Relation entre la taille de la fenêtre et le délai aller-retour (RTT).

Contrôle de congestion (3)

- La fenêtre a une taille dynamique
- 2 algorithmes:
  - Augmentation du nombre d'octets acquittés (slow-start)
  - Augmentation d'un segment / tailleFenetre
  - Seuil (sur la taille de fenêtre) pour passer de l'un à l'autre
- En cas de pertes:
  - Si perte détecté sur temporisateur
    - La fenêtre retourne à la taille minimale (Max Segment Size)
    - Le seuil est divisé par 2
  - Si perte détecté sur acquittement dupliqué:
    - La fenêtre est divisée par 2
    - Le seuil est divisé par 2

Contrôle de congestion: bilan



Les options

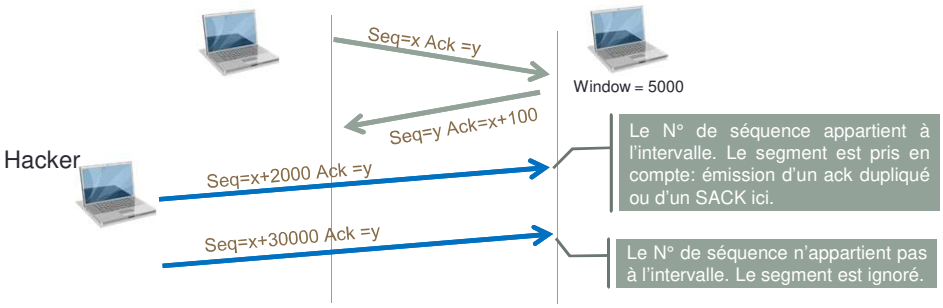
- Maximum Segment Size (MSS)
  - La taille maximale des segments est indiqué lors de l'établissement de connexion.
  - Permet d'adapter la taille des segments.
- Selective acknowledgement (RFC 2018)
  - Permet de ne pas retransmettre tous les segments quand un seul a été perdu (voir acquittement dupliqué).
  - Fonctionnement:
    - Lors d'une perte mais de la réception de segments de séquences supérieures
    - Une option SACK est rajouté avec la liste (*range* ou *blocks* des données reçues).
    - Retransmission des segments perdus uniquement.

TCP: sécurité

Insertion dans une connexion en cours

- Possibilité de s'insérer dans une connexion en cours:
  - Les adresses IP et les numéros de ports doivent correspondre.
  - Les N° de séquence des segments en réception doivent appartenir à l'intervalle:

[N° du dernier segment acquitté ; N° du dernier segment acquitté + window]



Deny of Service

- Deny of Service: utilisé toutes les ressources d'un service/serveur de manière à ce que celles-ci ne soit plus disponible pour les utilisateurs légitimes.
- SYN flood
  - Envoyer plein de SYN sur un serveur avec des adresses IP forgées
  - ½ ouverture de connexions au niveau du serveurs: il a envoyé le SYN ACK et attend la réponse.
  - Les ressources sont réservés pour cette connexion durant plusieurs dizaines de secondes
  - Le nombre de connexions TCP maximum peut être atteint et les ressources indisponibles
- Solutions:
  - Limiter le nombre de connexion par adresse IP ou plage
  - Libération des connexions semi-ouvertes aléatoire (pour faire de la place)
  - SYN cookies

Firewall / Access list

- Les acces list ou firewall avec état gardent traces des connexions TCP établit.
- Avantage
  - Des paquets hors connexions préalablement établit sont filtrer
  - Vérification des N° de séquences / ack / syn / etc.
- Inconvénient
  - Le nombre d'états sur les firewall peut être très importants
  - DoS
- Note: l'implémentation courante sur les ACLs Cisco consiste uniquement à vérifier que le bit ACK est mis.

UDP

UDP: User Datagram Protocol

- N'assure aucun service, sauf
  - Numéro de ports
  - Checksum
- Utilisé
  - Transaction (DNS, etc.)
  - Applications temps réels
  - Application implémentant son propre système de fiabilisation/contrôle de débit.

Exercice 4

- Transmission de 3000 octets de A vers B, puis 1000 octets de B vers A.
- Le MSS est de 1460 octets
- Donnez les valeurs des flags, des numéros de séquences et des numéros d'acquittement.