Chapitre 06 : Contrôle de trafic et QoS dans les réseaux IP

Partie I

« Introduction aux besoins des applications”

Contrôle de trafic et QoS dans des réseaux IP « Sommaire »

- Applications
  - Classes d'application
  - Conditions d'applications
- mécanismes de contrôle de trafic au niveau Paquet
- Services Normalisés
- Contrôle de trafic d'Intradomain
- Contrôle de trafic d'Interdomain
- Études de cas

Taxonomie des applications

- Pour construire un réseau « supportant le contrôle de trafic », il faut qu'on ait une bonne idée de fonctionnellement des applications et de quelles sont leurs besoins
- En terme de contrôle de trafic et de la Qualité de Service, on peut établir une taxonomie assez simple des applications.
  - On peut les classer en deux classes principales (avec lesquelles on couvre la plus part des applications):
    1. Applications élastiques ou opportunistes
    2. Applications streaming

1. Applications élastiques ou opportunistes

- Applications **élastiques** ou opportunistes :
  - Si les ressources sont disponibles, les applications élastiques essayeront de les utiliser « les consommer »
  - Si les ressources ne sont pas temporairement disponibles, elles attendront « ou vont réduire leurs besoins » sans être sévèrement affectées
  - Exemples : WWW email, ftp (transfert de fichiers non critiques) , news...

2. Applications streaming

- Applications streaming
  - Une quantité minimum de ressources est exigée pour que l'application puisse fonctionner
  - Si la quantité minimum est disponible, l'application fonctionne bien
  - Si la quantité minimum non disponible, l'application ne fonctionnera pas
    - ⇔ du type Go/NoGo : soit tout fonctionnera bien, ou bien non
    - ⇔ En grande partie, pour ces applications qu'on mettra le contrôle de trafic
  - Exemple : téléphonie sur IP, Vidéo sur IP

Applications opportunistes « élastiques » Quelques Sous-classes

1. Applications interactives fiables (⇔ les deux termes sont importants):
   - **Fiables :** ⇔ on peut pas trop créer des taux de perte
   - **Interactives** ⇔ il y un utilisateur qui est en train de manipuler l'application, donc il faut que le temps de réponse soit le plus court possible pour permettre à l'utilisateur de manipuler correctement l'application
   - telnet, accès aux bases de données, …
2. applications Requête/Réponse :
   - Client/Serveur, NFS, RPC, l'informatique répartie...
   - Deux ordinateurs en train de dialoguer entre eux
   - Ce qui est important est que le **temps de réponse au niveau du réseau** soit **le plus faible possible** pour que les interactions se fassent le plus rapidement possible
3. Applications « batch » :
   - ftp, protection à distance, news, ..etc.
   - Besoin de **transférer un gros volume d'informations**
   - Le temps de transfert est moins important que d'être capable de **transférer correctement les informations**

Applications Streaming : Quelques Sous-classes

1. Applications multimédia conversationnelles
   - Conversation directe entre deux être humains ⇔ des contraintes en terme de délais
   - Exemple : voix ou vidéo sur IP
2. Applications multimédia interactifs
   - Un être humain qui dialogue avec une machine, mais les besoins en terme de temps de réponse ne seront pas les mêmes que lorsqu'on a deux être humains qui dialoguent entre eux
   - Exemple : simulations distribuées, jeux de réseau
3. Applications multimédia non interactives
   - flux continu d'information multimédia
   - Exemple : Enseignement à distance, diffusion audio/video en broadcast, news à la demande...

Applications et pile de protocoles –1-

- Un problème important à prendre en compte pour ces applications c'est de savoir comment ces applications vont fonctionner à l'intérieur du réseau ⇔ Le type de protocole de transport utilisé par ces applications.
- Deux protocoles de transport sont utilisés par les applications en fonction de leurs besoins :
  - **UDP**
  - **TCP**
- Ces applications vont s'appuyer sur l'un ou l'autre en fonction de leurs besoins

Applications et pile de protocoles –2-

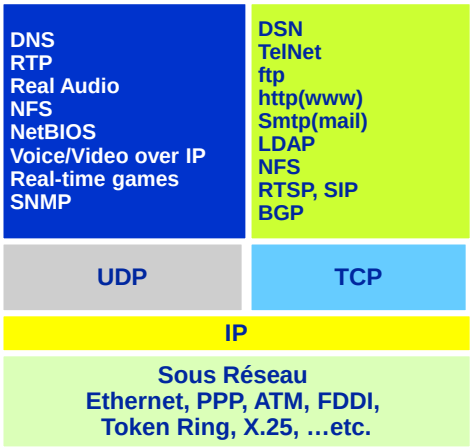
- Quel protocole de transport pour quelle application? Principe de base:

1. UDP convient à des:

- ✓ applications **Requête/Réponse** (principalement dans un environnement LAN, NFS)
- ✓ applications **multimédia** (LAN et WAN) (interactivité)
- ✓ Possibilité de faire de **multicast**

2. TCP est utilisé pour des transferts fiables :

- ✓ Transferts de fichier
- ✓ Émulation terminale
- ✓ applications **Requête/Réponse** principalement dans un environnement WAN **(profiter des mécanisme de contrôle TCP)**
- ✓ **Unicast** seulement

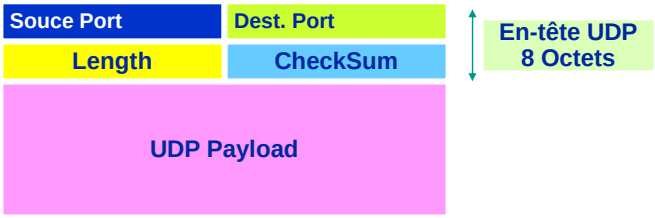


UDP –1-

- Objectifs : Fournir un service de transfert de paquets non fiable, sans connexion au dessus d'un service non fiable fourni par la couche IP « couche réseaux »
- Mécanismes :
  - format simple de paquet avec un checksum facultatif (pas de correction d'erreur)
  - Les applications sont identifiées par (le quadruplet):
    1. Adresse IP de la source
    2. Adresse IP de la destination
    3. Les numéros de port de la source et de la destination
  - N'inclut pas des mécanismes pour s'assurer :
    1. D'Une délivrance fiable des paquets
    2. Des erreurs sur les bits d'une trame peuvent être détectées mais non corrigées
    3. Délivrance respectant le séquencement des paquets (pas de garantie de séquencement)

UDP-2-

- **Au départ, UDP a été défini dans [RFC768] en 1980 et n'a pas été modifié depuis**
  - **Il s'appuie sur l'utilisation d'une en-tête de huit octets contenant les informations suivantes** :
    1. nombre port de source de 16 bits
    2. nombre port de destination de 16 bits
    3. longueur de TPDU-UDP : 16 bits
    4. Checksum TPDU-UDP :16 bits
- **La longueur maximum d'un paquet UDP correspond à la longueur maximum d'un paquet d'IP, à savoir 64KBytes**.
  - Même si UDP supporte des paquets de grande taille, la plus part des applications évitent d'envoyer des paquets de UDP plus grands que le **MTU** (Maximum Transmission Unit) d'IP



TCP-1-

- Objectifs : Fournir un service fiable, orienté connexion pour transférer un flux au dessus du service non fiable de la couche IP « couche réseau »
- Mécanismes :
  1. Un format de paquet protégé par un checksum
  2. établissement et libération de la connexion
  3. Une connexion TCP est identifiée par les quatre informations
     1. Adresse IP de la source
     2. Adresse IP de la destination
     3. Le numéro de port de la source
     4. Le numéro de port de destination
  4. Mécanisme permettant un Transfert fiable de l'information. Pour réaliser ça, TCP s'appuie sur :
     A. Le contrôle de flux avec acquittement entre la Source et la Destination.
     B. Détection et retransmissions des paquets perdus
  5. Contrôle de congestion (qui permet à TCP de s'adapter au débit qui est disponible à l'intérieur du réseau) « concerne le réseau »

TCP –2-

- Points importants au sujet de TCP :

  1. Le TCP essaie toujours d'envoyer de gros paquets (TCP est optimisé pour envoyer à chaque fois des paquets de # 1500 octets avec les entêtes IP, TCP)
  2. Le débit maximum est limitée par taille des fenêtres de l'émetteur et du receveur (⇔ le min entre la taille des deux fenêtres)
     - Le débit max disponible ≅ (taille de la fenêtre) / RTT

       ⇔Si les Sources et Destination sont loin dans le réseau => débit plus faible

       ⇔Pour des transferts à Haut Débit, il faut utiliser des fenêtres de taille importante sur les serveurs et les clients

TCP –3-

3. Dans TCP, il y plusieurs Mécanismes de retransmission des paquets en cas de pertes (s'appuient sur des temporisateurs) :
   1. Mécanisme qui s'appuie sur un temporisateur « Expiration de temporisateur de retransmission »
      - Quand TCP envoie un paquet, il va armer un temporisateur. Si le temporisateur expire avant réception de l'acquittement , TCP retransmet le paquet (en supposant qu'il était perdu)
      - Ce mécanisme fonctionne toujours, mais donne de très mauvaises performances
      - Donc, pour avoir de bonnes performances avec TCP, Il faut s'arranger pour éviter que TCP ne doive retransmettre des paquets à l'expiration du temporisateur de retransmission. ⇔ Forcer TCP à utiliser son autre de mécanisme retransmission qui est `fast retransmit`

TCP –4-

2. Retransmission rapide « fast retransmit »

- Principe : Fonctionnement sur base de retransmission sélective des paquets perdus
  a. L'Émetteur envoie des paquets
  b. Si un seul paquet a été perdu, le Receveur va envoyer des acquittement qui pointent vers le même paquet.
  c. Au niveau de la source, lorsqu'on a reçu trois acquittements (dupliqués) qui pointent vers le même paquet, on va retransmettre le paquet perdu
- Grâce à ça, on peut éviter des baisses de performances au niveau de TCP
- Pour profiter de mécanisme de Fast Retransmit, il faudra s'arranger au niveau de réseau pour perdre des paquets individuels. ⇔ Il faut éviter de perdre des blocs de paquets d'une même connexion TCP

TCP -5-

- **Au niveau de Contrôle de congestion dans TCP's, il y a un certain nombre d'éléments à prendre en compte :**

1.le contrôle de congestion s'appuie sur une **fenêtre de congestion** au niveau de l'émetteur qui interagit avec le **contrôle de flux (basé sur les fenêtres)** et qui permet à la source de s'adapter au débit auquel, la destination est capable de recevoir,

2.comment ? => Trois mécanismes de base

- La transmission des paquets de la source **est régulée par l'envoie des acquits** qui reviennent de la destination. permet à la destination d'indiquer à la source à quel moment il faut envoyer de nouveaux paquets (⇔ auto régulation de la source à la destination). **C'est le premier mécanisme de contrôle de congestion dans TCP**
- **Les autres mécanismes** : **Slow Start** et **Congestion avoidance** : Permettent à la source d'adapter **sa fenêtre de congestion** en fonction de la **disponibilité du réseau** : l'idée est qu'on va restreindre la quantité d'information que la source peut envoyer en attente d'acquittement en fonction **des ressources qui sont disponibles dans le réseau** (S'il y a beaucoup de ressources dispo dans le réseau, on aura une fenêtre de congestion qui est élevée, si non on aura une fenêtre de congestion qui est faible)

TCP -6-

- Sur base de ces mécanismes, on peut constater que le comportement de TCP en cas de congestion n'est pas toujours parfait :

1. A cause de l'existence d'une interaction entre la source et la destination sur base des acquittements, les sources et les destinations qui se trouvent **loin dans le réseau** (⇔ qui ont un long RTT) vont être pénalisées au niveau de contrôle de congestion par rapport aux sources qui ont un RTT plus court
2. La détection de la congestion dans TCP se fait uniquement lorsque la congestion s'est produit. Le mécanisme qui indique la congestion dans TCP ce sont les pertes de paquets ⇔ TCP va régulièrement transmettre trop d'informations dans le réseau pour provoquer une congestion et pour savoir la limite de débit qu'il peut utiliser
3. Le comportement de TCP est « par des expérimentations » nettement meilleur lorsqu'on perd des paquets individuels plutôt que lorsqu'on va perdre des groupes de paquets

   ⇔ on aura tendance à optimiser le réseau pour essayer de perdre des paquets isolés dans le réseau)

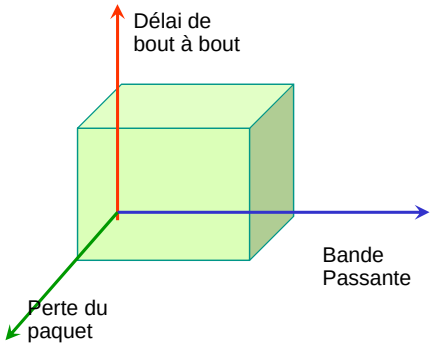
Caractéristiques du trafic d'Internet – comportement des application sur @-

- **Autre chose à prendre en compte sont les caractéristiques du trafic sur Internet**
- **Types de trafic sur le Backbone internent**
  - **Étude sur le backbon de MCI's en avril 1998 (à 55 Mbits/s)**
  - **trafic: protocoles de transport**
    - TCP: 90% -95%, UDP: 5-10% du trafic

1. **Applications utilisant TCP**
   - **WWW : 75% du trafic de TCP**
     - principalement des connexion TCP de courtes durées
     - taille moyenne d'une des paquet : 10-20 paquets de TCP (taille moyenne d'une page HTML ou une image GIF # 10KOctets)
   - **News : 7 % du trafic de TCP**
     - Principalement des connexion TCP de longue durée (100-2000 paquets)
   - **ftp: 4% du trafic de TCP**
     - Principalement des connexion TCP de longue durée (100-1000 paquets de TCP)
   - **email: 3% du trafic de TCP**
     - Principalement des connexion TCP de courte durée (10-20 paquets de TCP)
2. **Applications utilisant UDP: DNS**
   - Streaming (RealAudio, RealVideo), les jeux, VoIP (⇔ faibles à l'époque)

Besoins des applications –1-

- **Dans un monde idéal**
  - Délai de bout à bout : 0
  - Perte du paquet : 0%
  - Débit : infinie



- **Dans un monde réel**
  - Optimisez pour les délais de bout à bout, pour le débit ou perte, **mais pas pour toutes**!

Besoins d'application –2-

- **Pour se faire une idée intéressante des besoins des applications, il est important de revenir sur ces classifications et essayer d'identifier leurs besoins en terme de délai, en terme de taux de pertes et en terme de débit**

- **Applications fiables interactives :**

- **Telnet, accès aux bases de données, WWW, transactions, Xwindows...** : Un utilisateur attend des informations d'un serveur (⇔ interaction avec un être humain). Pour que l'interaction se fasse bien,
  - **il faut minimiser les Délais de bout à bout : (minimiser le délai < 100 ms pour que l'appli fonctionne bien)**
  - **Les Débits exigés sont , en général, faibles** (on peut jouer sur le débit via le taux de compression)
  - Au niveau des taux de pertes : **Les perte de paquet seront récupérées, en principe, par TCP**
    - ✓ Si on a des pertes de temps en temps : ce n'est pas critique
    - ✓ Mais des pertes élevées augmenteront le délai de bout à bout au niveau applicatif

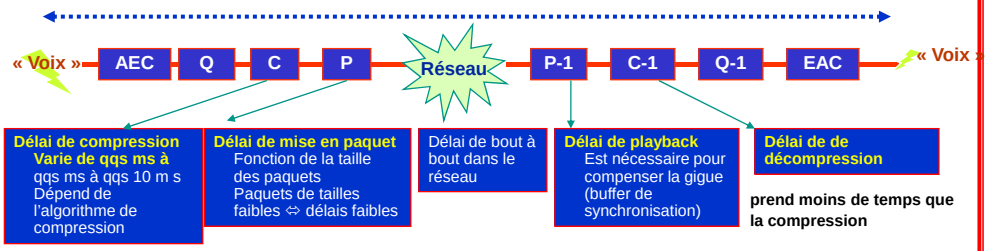
Besoins d'application –3-

- Applications multimédia interactifs :
- **Voix ou vidéo sur IP :** Deux utilisateurs discutent via le réseau. Pour que l'interaction se fasse bien,
  - il faut minimiser les **Délais de bout à bout** (< 200 ms pour l'audio entre les deux utilisateurs) => il faut prendre en compte pas seulement le délai au niveau du réseau mais aussi dans les autres composantes (téléphone source, …etc.)

    ⇔ *un paquet en retard est équivalent à un paquet perdu*
  - **Le débit exigée peut être importante** (dépend de l'application) (le cas de la vidéo / à l'audio)
  - L'application peut continuer à fonctionner si le **taux de perte des paquets** est **faible (< 20%)**

Délai de bout à bout dans une application multimédia

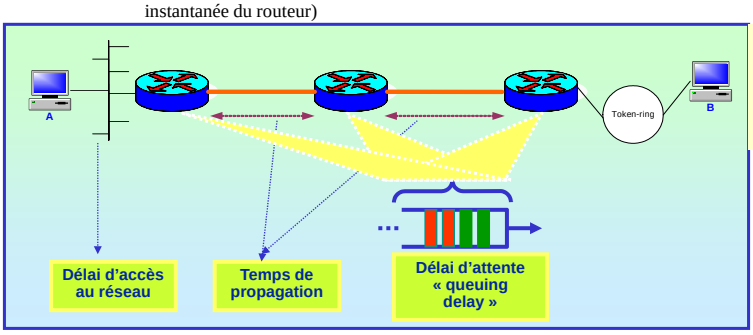
- Besoins pour la voix interactive (⇔ prendre en compte tous les délais)
  - Délai de bout à bout (bouche-à-oreille) ne devrait pas dépasser 100 millisecondes
    - Délai de bout à bout se compose des délais suivants :



- Standards pour l'échantillonnage et la compression de la voix
  - ✓ G. 711 : 64 Kbps, délai de compression : 0.75 msec
  - ✓ G.726 : 32 kbps, délai de compression : 1 msec
  - ✓ G. 728 : 16 kbps, délai de compression : 3-5 msec
  - ✓ G. 729 : 8 kbps, délai de compression : 10 msec
  - ✓ G. 723.1 : 6.3 kbps, délai de compression : 30 msec

Délai de bout à bout des paquets

- Les composants du délai de bout à bout dans le réseau :
  1. **Délai d'accès au réseau [variable]**
     - Dépend de la **technologie** du réseau et de **sa charge** : token ring d'e.g.CSMA/CD...
  2. **Délai lié au temps de propagation sur les liens [fixé]**
     - le signal électrique a besoin de 5µsec approximativement pour traverser **1** kilomètre
  3. **Délai lié au temps de transmission [fixé]**
     - un paquet de **P** octets a besoin de **(P*8 / B)** secondes pour être envoyées sur un lien de **débit de B bits/seconde** (⇔ dimensionnement des lignes et en budget des lignes)
  4. **Délai d'attente « queuing delai » à l'intérieur des routeurs intermédiaires [variable]**
     - change avec l'occupation des buffers de chaque routeur (⇔ dépend de la charge du réseau et de la charge instantanée du routeur)



1. Sur le designe du réseau (débit des lignes)
2. Sur les délais dans les buffers
3. Sur les réseaux d'accès

Besoins d'application –4-

- Contrôle du réseau : OSPF, RIP SNMP, BGP...
  - Les besoins en terme de débit sont relativement faibles
  - Délai devrait être faible (préférable)
  - Perte de paquet devrait être évitée : la perte de paquet peut augmenter le temps de convergence des protocoles de routage ou affecter les trappes SNMP
- Applications Requête / Réponse : Applications C/S, NFS...
  - Les besoins en terme de débit peuvent être importantes
  - Les délai devraient être faibles pour les transactions courtes spécialement lorsque l'utilisateur attend une réponse
  - la perte de paquet peut être récupérée par les protocoles des couches supérieurs

Besoins d'application –5-

- Applications « batch » : ftp, Archivage à distance « remote backUp », transactions HTTP longues,…
  - Un utilisateur attend la réalisation d'un travail mais n'attend pas derrière sa machine
  - Délais de bout à bout des paquets peuvent être importants
  - Besoins de débit peuvent être importants
  - Le taux de pertes des paquets peuvent n'a pas besoin d'être faible
    - Les pertes de paquet seront récupérées par TCP
- Applications non interactives de multimédia : Enseignement à distance, Diffusion audio/vidéo en « broadcast »...
  - Un utilisateur reçoit un flux continu d'information
  - Délais de bout à bout n'a pas besoin d'être faible (dépend du type d'applications)
  - Besoins en terme de débit peuvent être importants
  - La perte des paquets peut être tolérées
    - mais un taux de perte important des paquets peut complètement tuer l'application

Applications : Résumé - Bilan

- Deux classes des applications
  1. **Applications élastiques « opportunistes »**
     - Utilisent TCP
     - applications dominantes dans l'Internet d'aujourd'hui
  2. **Applications du type streaming**
     - Utilisent RTP/UDP
     - Utilisées de plus en plus sur Internet
- Besoins des applications, on a des besoins
  - En terme de délais (et jeter « gigue »)
  - En terme de débit
  - Et en terme de Taux de perte