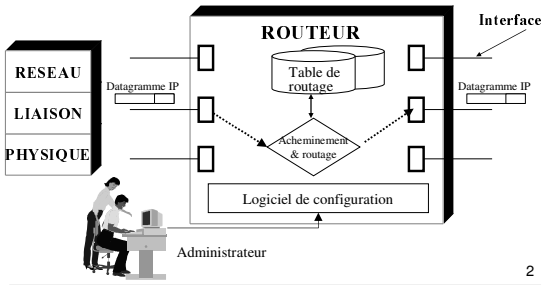
La Qualité de Service dans l'Internet

Sidi Mohammed SENOUCI
Orange Labs

* Ces transparents ont été en grande partie réalisés grâce au support de cours de Nadjib ACHIR (Univ. Paris-Nord) et fortement inspiré des tutoriaux d'Olivier BONAVENTURE (UCL)

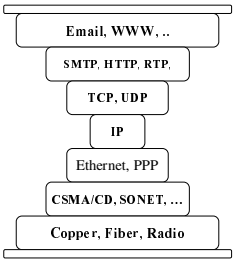
L'Internet Actuellement

- Un routeur IP fait du routage
  - Il construit une table de routage afin d'effectuer l'acheminement des paquets IP



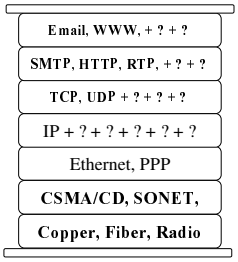
L'Internet Actuellement (2)

- Une très grande panoplie d'applications
- Une grande panoplie de protocoles de communications
- IP goulot d'étranglement
  - **Unification du moyen de communication**
- Une grande panoplie de méthodes d'accès
- Une très grande panoplie de medias d'accès



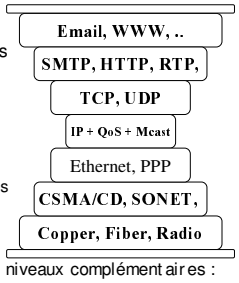
Les changements possibles (1)

- Faire participer le réseau dans les communications
  - Introduction des caches
    - Réduit la surcharge sur le réseau
    - Réseau de distribution de contenus
- Offrir de nouveaux services
  - Communications multimédias
    - Offrir des garanties suivant les contraintes des applications
  - Communication de groupe (Multicast)
    - Routage, fiabilité et QoS



Les changements possibles (2)

- Le besoin de plus de fonctionnalités
  - Autre que le routage
  - ⇨ Faire évoluer notre équipement de routage
- Ajout de la QoS, du Multicast, …
  - QoS : satisfaire les contraintes des applications en terme de débit (ou bande passante) et de délai
- Offrir des garanties se fait à deux niveaux complémentaires :
  - Mécanismes de contrôle de trafic de niveau paquet
    - Comment chaque paquet est-il traité au niveau d'un routeur ?
  - Mécanismes de contrôle de trafic de niveau flux
    - Comment chaque flux est-il traité au niveau d'un réseau ?



PLAN

- Rappel : Fonctionnement de TCP
- Partie 1 : Mécanismes de contrôle de trafic de niveau paquet
- Partie 2 : Mécanisme des contrôle de trafic de niveau flux

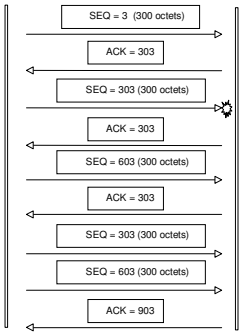
Fonctionnement de TCP

RAPPEL

TCP : Transport Control Protocol

- TCP est le protocole qui permet de fiabiliser une connexion dans l'Internet
  - Mise en place des connexions
    - Three Way Handshake (SYN, SYN-ACK, ACK)
  - Reprise après perte de paquets
    - Grâce à un mécanisme d'acquittement des paquets reçus
- TCP s'engage à produire un service de transport
  - Service de bout en bout
  - Sans erreur
  - Ordonnancement des paquets
  - ⇨ Compense les manques de IP (fiabilité)
- Idée de base de TCP :
  - Réussir à transmettre avec le débit qui lui est disponible dans le réseau
    - Grâce à ses mécanismes d'augmentation/réduction du débit après chaque envoi réussi ou non
    - Si toutes les connexions sont TCP friendly alors on devrait avoir une équité entre les flux pour le partage des ressources réseau

Comment fonctionne TCP ?

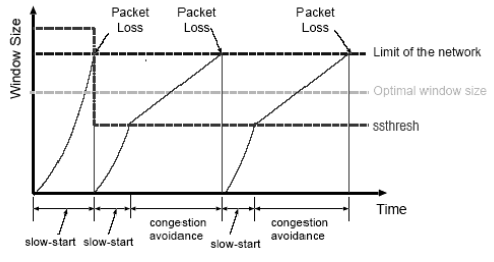


Comment fonctionne TCP (Reno) ?

- Hormis les aspects protocolaires ,TCP repose sur deux mécanismes principaux
  - Slow Start
    - Au début TCP commence par envoyer un seul segment
    - Si ce segment est correctement acquitté alors TCP double son débit d'émission (fenêtre d'émission) sinon il le divise par deux (**comportement agressif**)
    - TCP sort de la phase Slow-Start si sa fenêtre d'émission atteint un seuil prédéfinie
      - Ce seuil est divisé par deux également après la perte d'un paquet
  - Congestion Avoidence
    - TCP continue à envoyer ses segments
    - A chaque acquis positif, il augmente sa fenêtre d'émission d'une petite portion (<< à la taille d'un paquet)
    - S'il ne reçoit pas d'acquis, il renvoi le paquet immédiatement (fast retransmit) puis divise son débit de transmission par 2
      - Après 3 pertes consécutives on repasse par la phase Slow-Start en réduisant de moitié le seuil max de la phase Slow-Start

L'idée de TCP est qu'un flux puisse se réguler de telle sorte qu'il trouve la valeur de débit disponible pour lui dans le réseau

Evolution de la fenêtre de congestion TCP



Congestion control of TCP Tahoe

Mécanismes de contrôle du trafic au niveau paquet

PARTIE 1

Mécanismes de contrôle du trafic au niveau paquet

Plan

- Les différents types de garanties
  - Satisfaire les contraintes des applications
  - Bande passante, Délai
- Le support du service Best Effort
  - Ou, comment pouvons-nous supporter efficacement un service TCP/IP classique sans contraintes ?
- La garantie de la bande passante
  - Bande passante maximum
  - Bande passante minimum
- La garantie du délai

Idée : construire petit à petit un routeur qui offre les garanties que l'on désire mettre en place

Quel type de garanties ?

- Trois types de garanties de "bande passante"
  - **Best Effort** (pas de garantie de BP)
    - Adaptée aux applications classiques (Mail, FTP, ..), pas critiques, applications élastiques
  - Garantie de bande passante **maximum**
    - Une partie de la bande passante est réservée pour le flux
    - Le flux ne peut pas transmettre plus que la BP maximum
    - Adaptée aux applications non adaptatives
    - Exp. : le réseau téléphonique offre des garanties de BP maximum (32Kbs)
  - Garantie de bande passante **minimum**
    - Le flux peut, à n'importe quel moment, utiliser la bande passante minimum qui lui a été allouée
    - Le flux peut utiliser plus de BP si le réseau n'est pas congestionné
    - Adaptée aux applications élastiques critiques et aux applications adaptives de type streaming
      - Flux ayant un besoin minimum pour fonctionner correctement mais pouvant s'adapter pour profiter de plus de bande passante

Quel type de garanties ? (2)

- Garantie de délai et de gigue (variation du délai)
  - Flux Best effort
    - Pas de garantie de délai et de gigue → Internet aujourd'hui …
  - Applications non adaptatives avec contraintes strictes
    - Flux ayant une garantie de BP maximum (téléphonie interactive)
    - Garantie du délai maximum
      - Exemple : application dite de voix interactive (téléphonie, 32Kbs, d<100ms)
    - Garantie de gigue
      - Plus la variation du délai est courte (nulle c'est l'idéale) plus les buffers de playback sont courts et mieux c'est pour l'application
      - Pas de buffer de playback (Réseau téléphonique) c'est l'idéal mais ce n'est pas possible dans l'Internet
  - Applications adaptatives ou élastiques avec contraintes
    - Flux ayant une garantie de BP minimum (streaming)
    - Ici on autorise la perte de paquets (2 types de paquets garantis et non garantis)
      - ⇨ La gigue ici n'a pas vraiment de sens (un paquet perdu à un délai infini)
      - ⇨ On peut offrir une garantie de Délai maximum aux paquets qui arrivent

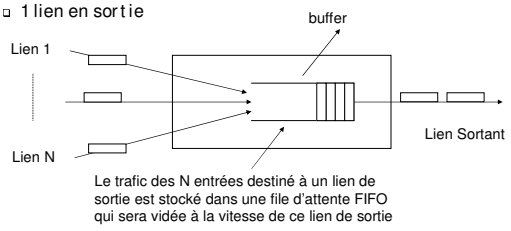
Mécanismes de contrôle du trafic au niveau paquet

Plan

- Les différents types de garanties
  - Bande passante, Délai
- Le support du service Best Effort
  - Le fonctionnement de TCP dépend de celui du routeur
  - Le routeur n'offre aucune garantie de délai/BP
  - Comment peut-il faire de son mieux ?
- La garantie de la bande passante
  - Bande passante maximum
  - Bande passante minimum
- La garantie du délai

Service Best Effort : routeur simple

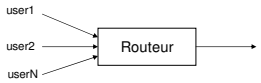
- Le routeur le plus simple possible
  - N liens en entrées
  - 1 lien en sortie



  - C'est la solution la plus simple pour le service Best Effort
  - Que doit-on rajouter comme mécanismes pour obtenir un service best effort optimal (acceptable par tous)

Caractéristiques du service Best Effort

- Quel doit être le but du service Best Effort ?
  - Faire tout son possible pour que tous les utilisateurs puissent obtenir le meilleur service
  - Ce "possible" doit être le même pour tous les utilisateurs du réseau
  - Le réseau doit donc offrir un service équitable à tous ces utilisateurs
    - Définition de l'équité pour un lien partagé
      - N flux arrivant au routeur => partager le débit de sortie par N et offrir à chacun 1/Nème du débit (2 voisins auront le même débit)



    - Pas généralisable dans un réseau complexe tel que l'Internet (optimisation dépendante de la "topologie")

Caractéristiques du service Best Effort (2)

- Comment peut-on définir l'équité dans tout un réseaux ?
  - Vue utilisateur : maximiser la bande passante reçu par les utilisateurs
    - Ne permet pas toujours d'optimiser les ressources du réseau
  - Vue opérateur : maximiser l'utilisation des ressources du réseaux
    - Ne permet pas d'avoir le même facteur de satisfaction entre les utilisateurs
  - Ces deux vues ne sont pas satisfaisantes puisque deux voisins ayant les mêmes conditions n'auront probablement pas la même ressource
  - L'idée est d'essayer d'obtenir le même facteur de satisfaction des utilisateurs tout en maximisant l'utilisation des ressources du réseau ->équité max-min

19

Equité Max-Min

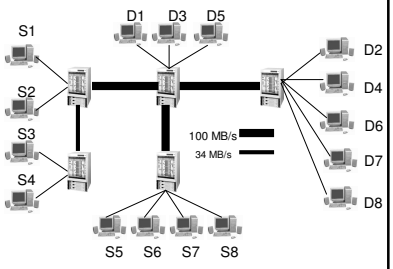
L'allocation de bande passante Max-Min est une allocation de bande passante qui maximise la BP allouée aux utilisateurs recevant le moins de débit

- Propriété de l'équité Max-Min :
  - Dans le cas où un utilisateur est insatisfait par la bande passante qu'il reçoit, la seule manière de le satisfaire serait de diminuer la bande passante reçue par ceux qui ont moins de bande passante que lui
    - ⇨On ne peut donc pas se plaindre de ce genre de distribution =>Satisfait les utilisateurs quine pourront pas se plaindre
    - ⇨Idéale pour les utilisateurs mais pas forcement idéale pour les opérateurs pour maximiser l'utilisation de ces ressources (voir vue opérateur)

20

Exemple : Max-Min

S1 : 45,25 MB/s
S2 : 20,75 MB/s
S3 : 17 MB/s
S4 : 17 MB/s
S5 : 37,75 MB/s
S6 : 20,75 MB/s
S7 : 20,75 MB/s
S8 : 20,75 MB/s



Nous pouvons facilement vérifier que nous ne pouvons pas augmenter la BP d'une communications sans en enlever à une autre ayant la même quantité de BP ou moins

21

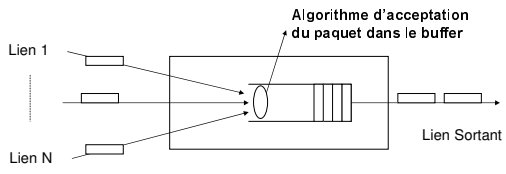
Comment faire ?

- Equité Max-Min dans un réseau TCP/IP
  - Un but idéal, mais difficile a mettre en œuvre en réel
  - Dans un réseau réel, il n y a pas d'oracle central permettant de dire à chaque source la quantité de BP qui lui est attribuée
- L'équité Max-Min est un objectif "idéale" que l'on va essayer d'atteindre dans un réseau Internet TCP/IP
- Son obtention dépend de deux mécanismes :
  - Mécanisme au niveau du routeur pour le traitement des paquets à l'intérieur du réseau
    - Quels sont les paquets à éliminer suivant le niveau de congestion
  - Mécanisme de contrôle de congestion au niveau des terminaux
    - Cas d'applications utilisant TCP (95% du trafic Internet est TCP): utilisation de son mécanisme de contrôle de congestion
      - Diminue son débit lors des congestions pour essayer d'approcher une équité Max-Min dans le réseau
    - Et les applications utilisant UDP (jeux en réseaux) ? → Pas de contrôle de congestion comme dans TCP
      - Si la proportion de flux UDP augmente → Problème!

22

Routeur Simple V1

- Traitement des paquets à l'intérieur du routeur



- A l'arrivé d'un paquet l'algorithme décide si le paquet doit être accepté ou non
  - Permet d'influer sur le comportement de TCP

23

Algorithme d'acceptation des paquets

- Deux questions fondamentales :
  - Quand faut il éliminer des paquets ?
    - Quand le buffer est rempli à 100% (ex. Tail Drop)
      - Simple, facile et implémenté par tous les routeur
    - Quand l'occupation du buffer atteint un certain seuil (ex. RED – Random Earky Detection)
      - Le but est de prévenir TCP de l'imminence d'une congestion
      - Autre option: éliminer lorsque l'occupation grandit trop vite pour éviter les congestions car la réaction de TCP n'est pas immédiate
  - On a décidé de jeter un paquet, la question est de savoir lequel ?
    - Les paquets qui arrivent (ex. Tail Drop)
      - Pas très équitable
    - Des paquets appartenant aux même flux que le paquet qui arrive (réduire le délai pour que TCP détecte la congestion)
    - Des paquets appartenant à d'autres flux
    - Les paquets à la tête du buffer
      - TCP réagira beaucoup plus vite
      - Peut améliorer les performances de TCP

24

Algorithme d'acceptation des paquets (2)

- Objectif de cet algo → Contrôler la quantité de paquets dans les buffer
  - Supporter efficacement le service Best effort
    - Doit offrir une utilisation équitable des buffers
      - Essayer d'approcher une allocation Max-Min
      - La seul manière de contrôler le débit est de contrôler les pertes entre les différentes sources
      - ⇨ Essayer d'avoir des taux de pertes équivalent entre les flux
  - Offrir une protection entre les différents flux
    - Un flux ne peut pas perturber d'autres flux en les empêchant d'avoir des paquets dans le buffer
    - Il faut donc pouvoir accéder au buffer d'un routeur de façon équitable entre les flux
  - Obtenir une bonne utilisation du lien de sortie
    - Pour éviter que TCP ne devienne inefficace car les buffers sont **trop peu** ou **trop fortement** chargés en moyenne

25

Tail Drop

- Le plus simple algorithme pour l'acceptation des paquets
- Principes :
  - Quand un paquet arrive et le buffer est remplis, le paquet est détruit
  - Avantages
    - Trivial à implémenter (dans tous les routeurs)
    - Permet de limiter le nombre de paquets perdus pour les grands buffers
  - Inconvénients
    - Pas de distinction entre les différents flux
      - Exp: un paquet Telnet qui arrive alors qu'il y a pleins de paquets FTP qui sont dans le buffer
    - Pas la meilleur solution pour répondre aux besoins de TCP
      - Si le buffer est plein, il risque de le rester un certain temps … cela risque de provoquer des pertes en rafales pour certaines connexions TCP → On n'arrivera pas à une équité Max-Min

26

Random Early Detection (RED)

- Objectifs de RED :
  1. Faire mieux que Tail Drop en considérant les contraintes de TCP
  2. Doit être facile à implémenter dans un routeur simple du début des années 90 (Pas trop de calculs CPU)
  3. Permet un faible, mais non nulle, taux d'occupation du buffer
     - Un faible taux d'occupation → un faible délai pour les applications interactives
     - Un faible taux d'occupation → permet d'éviter les problèmes liés à TCP et énoncés précédemment
     - Un taux d'occupation différent de zéro → une bonne utilisation du lien de sortie
  4. Essayer d'approximer une élimination équitable des paquets appartenant à différents flux dans le buffer (une table avec toutes les sources et pénaliser celles qui on plus de trafic -> sol non réaliste dans un routeur simple)
  5. Elimine des paquet d'une façon utile pour TCP (friendly)
     - Eviter que des blocs (bursts) de paquets ne soit éliminés, car TCP réagit mal à ce type de pertes en rafale

Comment atteindre ses objectifs?

27

Random Early Detection (RED) -2-

- Principes :
  - Comment détecter la congestion ?
    - Une congestion ici n'implique pas un taux d'occupation du buffer = 100 %
    - Mesure le taux d'occupation moyen du buffer
    - Le buffer est considéré congestionné s'il atteint un taux d'occupation supérieur à un certain seuil
      - La valeur du seuil (pré-configurée) entre 10% - 20% de sa taille
  - Que doit on faire en cas de congestion ?
    - Une élimination probabiliste pour les paquets entrant
      - Force TCP à réduire son débit (Slow Down)
      - La probabilité d'élimination augmente avec le niveau de congestion

28

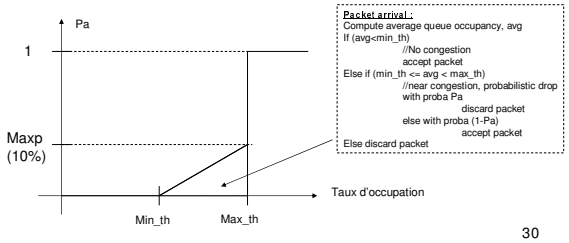
Random Early Detection (RED) -3-

- Pourquoi une élimination probabiliste ?
  - Eviter d'éliminer des (blocs) bursts de paquets appartenant à un même flux
    - TCP n'aime pas cela
    - Comme les pertes sont aléatoires on a très peu de chance d'éliminer plusieurs paquets consécutifs d'un même flux
  - Essayer d'éliminer des paquets pour chaque flux en fonction de l'utilisation du réseau
    - Si un flux possède un débit 3x plus grand qu'un autre, il perdra 3x plus de paquets
      - Taux de pertes approximativement "équitables"
  - Eviter la synchronisation (entre les connexions TCP)
    - Lorsque le degré de congestion est faible on perd pour certaines connexions et pas d'autres
      - Tous les TCP ne diminuent pas et n'augmentent pas leurs débits en même temps

29

Random Early Detection (RED) -4-

- Implémentation de RED
  - Adapté aux routeurs avec un seul buffer
  - On essaye de se débrouiller pour rester entre les deux seuils (Min_th, Max_th)
    - Augmenter linéairement la probabilité de rejet avec le taux d'occupation moyen (quand min_th <= avg <max_th)



30

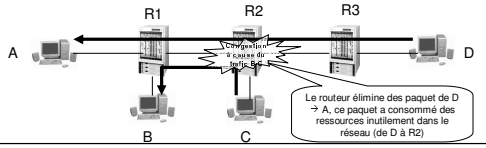
Random Early Detection (RED) -5-

- Devons nous déployer RED dans un réseau ?
- Difficile d'offrir une réponse claire aujourd'hui
  - Une partie de la communauté
    - Une meilleur utilisation du réseau
    - Un faible délai d'attente dans les files
  - L'autre partie de la communauté
    - Comment fixer, dans un réseau opérationnel, les seuils ***min***, ***max***, ***maxp***, ainsi que le temps sur lequel on fait la moyenne?
      - On n'a pas de réponse claire!
    - Un mauvais choix des paramètres de RED offre des performances médiocres
      - Même que parfois Tail Drop devient meilleur

31

TCP Explicit Congestion Notification (ECN) -1-

- Contrôle de congestion dans les réseaux TCP/IP
  - Supposition de base
    - La perte de paquets est la seule façon de détecter une congestion
  - Comportement du routeur
    - Elimine les paquets en cas de congestion
  - Comportement des systèmes terminaux
    - Control de congestion TCP (slow-start, congestion avoidance)
- Regardons les problèmes dues à ces suppositions dans l'exemple suivant
  - Création d'une certaine inefficacité dans le réseau



32

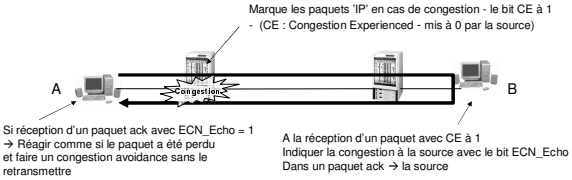
TCP Explicit Congestion Notification -2-

- Comment peut on améliorer le contrôle de congestion de TCP/IP ?
  - Modifier le comportement du routeur
    - En cas de « faible » congestion le routeur doit informer la source
    - En cas de « forte » congestion le routeur doit éliminer des paquets
  - Comment le routeur peut il informer la source ?
    1. Envoie un message explicite à la source
       - ICMP "source quench", rarement utilisé car envoyer un nouveau paquet en cas de congestion n'est pas une bonne idée
    2. Ajouter une notification aux paquets congestionnés
       - Reçu par la destination et retournée vers la source grâce aux paquets TCP-ACK

33

TCP Explicit Congestion Notification -3-

- Idée de base utilisant TCP-ACK

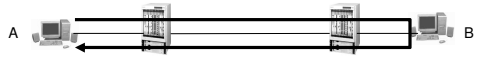


  - Reste 2 problèmes à résoudre ?
    1. Que ce passe t'il si le paquet ECN-ACK est perdu ?
    2. Comment déployer cette solution si 99,99% des sources/ destinations TCP ne supportent pas ECN ?
       - A noter que c'est supporté par les derniers noyaux Linux

34

Pertes des ECN-ACK

- Problème
  - Si un ECN-ACK est perdu, la source ne se rend pas compte de la congestion
    - Dans TCP, si un ACK est perdu, l'ACK suivant récupérera la perte du premier → Comment faire pour le bit ECN_Echo
- Solutions :
  1. Le récepteur doit transmettre N ECN-ACK, mais la source doit réduire qu'une seule fois son débit
  2. Permettre à la source de confirmer la notification au récepteur (flag CWR dans TCP)



Comportement de la source
Confirmation de la notification de la congestion
Mettre à 1 le flag CWR (Cong. Window Reduced) dans le prochain Paquet transmis vers la destination (sorte d'acquittement d'acquittement!)

Comportement de la destination
A la réception de paquet avec CE a 1
Indiquer la congestion à la source avec le bit ECN_Echo
Dans tous les paquets ACK → la source

35

Déploiement d'ECN -1-

- Comment supporter ECN dans les systèmes terminaux ?
  - Cas de TCP
    - TCP doit être modifié pour supporter les nouveaux flags
    - Une négociation peut se faire à l'établissement de la connexion
      - Les deux systèmes terminaux supportent ECN ou non
        - Envoi lors de l'établissement de la connexion d'un paramètre indiquant le support d'ECN
        - Suivant la réponse l'initiateur saura si oui ou non ECN est supporté par le récepteur
  - Cas d'autre protocoles de transport (UDP, RTP/RTCP)
    - Nécessité d'adapter ces protocoles à l'utilisation de ECN

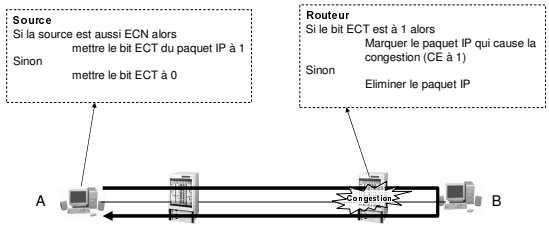
Pour le moment on l'utilise que pour TCP

36

Déploiement d'ECN -2-

- Comment déployer ECN dans les routeurs ?
  - Un routeur doit être capable de distinguer entre
    - Les paquets IP provenant des flux supportant ECN
      - en cas de congestion → une notification est faite (bit CE à 1)
    - Les paquets IP provenant des flux ne supportant pas ECN
      - en cas de congestion → les paquets sont éliminés
  - Solution : Le bit ECT (ECN Capable Transport) dans l'entête IP
    - Mis à 1 par toutes les sources utilisant ECN
  - RMQ : ECN fonctionne bien avec RED mais pas avec Tail Drop
    - On fait fonctionner RED en mode marquage et non en mode élimination ->les sources réagiront par la suite

Déploiement d'ECN -3-



ECN est en cours de standardisation. Utilisé par les stations Linux mais pas tous les routeurs

Mécanismes de contrôle du trafic au niveau paquet

Plan

- Les différents types de garanties
  - Bande passante, Délai
- Le support du service Best Effort
- La garantie de la bande passante
  - Bande passante maximum
    - Comment identifier des flux (applications)
    - Comment fournir cette garantie aux flux identifiés
  - Bande passante minimum
- La garantie du délai

Service Best Effort Vs Service Garantie

- Service Best Effort
  - Peut être réalisé uniquement si tous les paquets IP veulent recevoir exactement le même service
  - Les paquets IP sont traités sans se soucier d'où ils viennent ni où ils vont …
- Service à bande passante garantie
  - Tous les paquets IP ne sont pas égaux
  - Donc, quelque part dans le réseau, les routeurs doivent savoir quel type de garantie est associée à un paquet IP particulier
    - La bande passante est une caractéristique d'un flux de paquets IP
  - Deux problèmes à résoudre
    1. Associer un paquet IP à un flux
    2. Offrir des garanties à certains flux

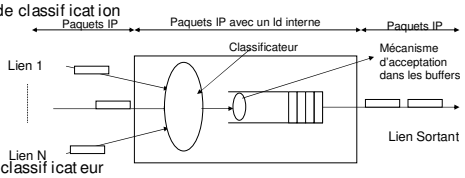
C'est quoi un flux ?

- Définition
  - Un flux est une séquence de paquets avec des caractéristiques communes
    - Les caractéristiques peuvent être basées sur n'importe quel champ du paquet
    - Un flux existe pour une période de temps (instant début, instant de fin)

  - Un flux de niveau N est une séquence de paquets avec des caractéristiques communes de niveau N
    - Niveau 3 (dépendent d'infos de l'entête IP)
    - Niveau 4 (TCP/UDP)
    - Niveau 7 (données applicatives)

Routeur Simple V2

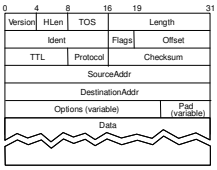
- Règles de classification



- Rôle du classificateur
  - Identifier le flux auquel appartient le paquet qui arrive
    - Cette opération d'identification peut être complexe
  - Une fois le flux identifié, on enregistre cette information localement pour être utilisée après par le routeur
    - La classification doit être faite au maximum 1 fois par routeur
    - Identification interne simple et non complexe (marquage utilisant le champs ToS par exemple) qui sera retirée à la sortie du routeur

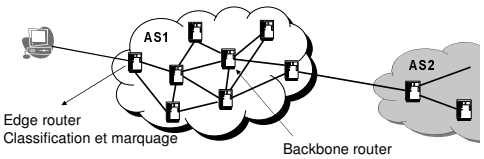
Flux de niveau 3 (1)

- Identification des flux de niveau 3
  - Adresse IP Source / Destination avec ou sans le masque de sous réseau
    - Eg. Id des flux = 132.227.0.0/16 (tous les paquets avec une telle adresse IP source)
  - Tout le trafic IP ayant la même route ou encore le même routeur de sortie (Id des flux = BGP next hop)
    - Requière du classificateur un parcours de la table de routage (en plus des @IP)



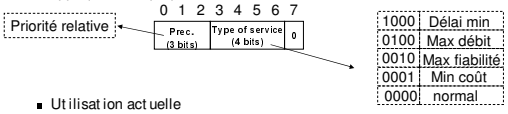
Flux de niveau 3 (2)

- La classification de niveau 3 dans chaque routeur peut être coûteuse
- Solution alternative
  - Effectuer une classification dans les routeurs de bordure (*edge/border routers*)
    - Les *edge* routers sont des routeurs à bas débit
  - Marquer explicitement les paquets classifiés
    - Les routeurs cœurs (*core/backbone routers*) n'ont plus besoin de classifier les paquets



Marquage des paquets IP (1)

- Comment peut-on marquer un paquet IP?
  - Un champ dans l'en-tête IP
    - ToS : Type of Service
    - Définit l'importance relative du paquet et le type de service à appliquer à ce paquet



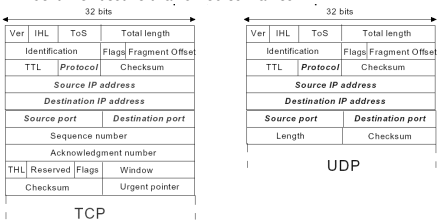
    - Utilisation actuelle
      - La partie « Precedence » est utilisée dans certains réseaux
        - Pour donner une priorité au trafic de routage par rapport au trafic normal
      - La partie « Type of Service » est rarement utilisée
    - Utilisation du ToS pour le marquage des paquets IP par les routeurs de bordure
      - Avantages : facile à réaliser (extension facile des fonctionnalités d'un routeur)
      - Inconvénient : nombre de flux pouvant être marquer (256)
        - Permet d'identifier de gros flux (trafic agrégé)

Marquage des paquets IP (2)

- Comment peut on marquer un paquet IP
  - Couche 2 (ATM ou Frame Relay)
    - Les flux de couche 3 sont mappés sur plusieurs flux de niveau 2
    - Fournir une QoS par la couche 2
  - Rajouter un nouveau champs avant l'entête IP
    - Utilisé par MPLS (en-tête de 32 bits dont 20 bits pouvant servir au marquage)
    - Principe
      - Les routeurs *edge* identifient les flux de niveau 3 et rajoutent un label de 32 bits à l'avant du paquet IP
      - Le backbone utilise ce champ pour le forwarding (swapping) → Commutation de circuits virtuels

Flux de niveau 4

- Indentification à l'aide du quintuplé
  - Adr Source, Adr Destination, Protocole, Port source, Port destination
    - Facile à réaliser par les routeurs
    - Identifier tout le trafic web ou mail ou …



Indentification des applications (1)

- Possibilité d'identifier des applications particulières selon le numéro du port TCP ou UDP
- Exemple de traitements :
  - Ne pas jeter les paquets Telnet (tout flux utilisant le port 23)

Application	Transport	Numéro du port
HTTP	TCP	80, 443
IMAP	TCP/UDP	143, 220
DHCP	UDP	67, 68
SMTP	TCP	25
SSH	TCP	22
Telnet	TCP	23
POP	TCP/UDP	109,110

Indentification des applications (2)

- Le problème en pratique
  - Pas toutes les applications utilisent un numéro de port bien connu
  - Exemples
    - FTP
      - Par défaut 20/21 (contrôle/transfert) mais on peut négocier un autre port
    - Applications multimédia stockées sur un serveur NetShow/RealAudio
      - Une négociation via le protocole RTSP (port 554) permet de choisir la vidéo et/ou l'audio (en streaming) ainsi que les numéros de ports TCP ou UDP à utiliser (de manière dynamique)
    - Pareil pour les appels RPC (Remote Procedure Call) pour faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications
    - Napster, Gnutella, Kazaa, ...
      - Utilisent rarement un numéro de port bien connu

Indentification des applications (3)

- FTP, RealAudio, Netshow, RPC utilisent des protocoles de contrôle annexes pour définir le numéro de port à utiliser
  - ftp-control pour FTP
  - RTSP pour RealAudio et NetShow
  - Portmap pour RPC
- Le routeur peut essayer de capturer ces informations de contrôle pour les utiliser ensuite
- Inconvénients
  - Complexe à mettre en œuvre
  - Consomme beaucoup de temps CPU (regarder le contenu des paquets de contrôle)
  - Marche à bas débit mais pas à très haut débit
  - Ne peut pas être utilisé pour un large nombre de flux

Limitations

- Connaître toutes les applications → **IMPOSSIBLE**
  - Aujourd'hui, on n'arrive pas à identifier tout ce qui traversent l'Internet
    - 10% du trafic TCP n'a pu être identifié lors d'une analyse effectuée en 2000 sur le trafic Internet passant par un Transit ISP
- Le déploiement de tunnels cryptés cache l'entête TCP/UDP aux nœuds intermédiaires → IPSec dans IPV6
  - Nécessité de marquer au niveau 3 avant le cryptage

Flux de niveau 7

- Exemple d'identification de flux au niveau 7
  - Autoriser tous les documents HTML contenant uniquement du texte
  - Réduire la bande passante consommée par les images (dans un document HTML) en les éliminant
    - En regardant le champ « Content Type » des paquets HTTP on peut savoir ce qu'il contient
- Peut être fait à bas débit mais pas à haut débit

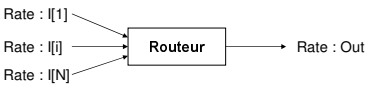
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<BODY BGCOLOR="#ffffff" BACKGROUND="/icons/back.gif">

This is a simple html page.<P>

<IMG SRC="left.gif" ALT="Left">
<IMG SRC="right.gif" ALT="Right">
</BODY>
</HTML>
```


La classification actuellement

- A quel niveau la classification doit se faire
  - Couche 7 (très coûteuse)
    - Nécessite de voir l'entête du paquet et le contenu
    - Ne fonctionne pas sur des routeurs haut débit
    - Jamais dans le Backbone
      - Fait au niveau firewall en général et on les optimise pour certaines applications uniquement (HTTP, Mail, FTP, ..)
  - Couche 3 Vs Couche 4
    - Tous les routeurs peuvent faire une classification de niveau 4 ou 3
      - On ne peut pas différencier entre les deux en terme d'avantages et inconvénients
    - La classification se fera uniquement dans les routeurs de bordure et les routeurs de cœur regarde uniquement le marquage

Offrir une garantie de bande passante

- Comment peut-on offrir une garantie de bande passante ?



- ***Si*** le trafic est fluide (liquide) ***Alors*** la bande passante peut être garantie à condition que :

$$\sum_{j=1}^{j=N} I[j] \leq Out$$

Théorique uniquement

- Avec des paquets, le problème est différent
  - On ne peut pas transmettre l'information dès son arrivée
    - On doit stocker certains paquets à l'intérieur du buffer
    - Nécessité de prendre en compte ce stockage si l'on veut offrir des garanties

☞ 2 contraintes : débit de la ligne de sortie + capacité du buffer du routeur

Offrir une garantie de bande passante

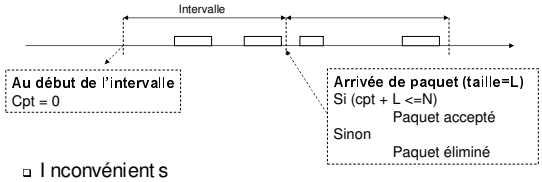
- Problème
  - Dans un réseau de paquet, le trafic est un flux de paquets de tailles variables et donc NON fluide
- Comment faire pour offrir une telle garantie de BP?
  1. S'assurer que le lien sortant n'est pas un goulot d'étranglement $\sum_{j=1}^{j=N} I[j] \leq Out$
     - On doit limiter le débit des flux dans les liens entrants
  2. S'assurer que le buffer du routeur ne soit jamais plein pour absorber les flux en entrée avant de les transmettre vers la ligne de sortie
     - On doit limiter aussi la quantité de buffer utilisée par les flux en entrée
- Il faut donc trouver un mécanisme qui permet de limiter le débit des lignes d'entrées dans tous les routeurs

Limitation du débit d'un flux entrant

- Comment peut-on limiter le débit des flux de paquets de taille variable pour un lien entrant?
  - Définir le débit à garantir à partir du contrat de trafic
    - Quelle unité pour représenter ce débit maximum ?
      - Nombre de paquets par unité de temps
        - Paquet de 40 bytes par seconde vs. Paquet de 1500 bytes par seconde
        - On ne prend pas en compte cette variabilité
      - Nombre de bytes par unité de temps
        - Semble meilleure mais quelle unité de temps (1ms, 1s, 1min, 1h, ..)
        - 1Mb/s = 1Mb/10ms puis rien vs. 1kb/ms => impact différent
  - Sur l'arrivée des paquets
    - Si le débit courant satisfait le contrat → paquet accepté
      Sinon éliminer le paquet pour ne pas gêner les autres

Mesurer le débit d'un flux (1)

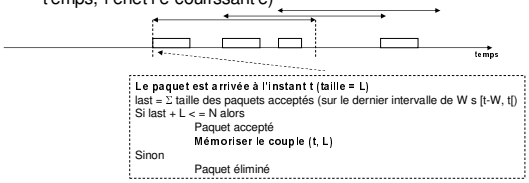
- Solution Simple au niveau du routeur
  - Le temps est divisé → intervalles de taille fixe
  - Le contrat du trafic est fixé comme N octets par intervalle



  - Inconvénients
    - Le début de l'intervalle peut influencer sur les paquets qui doivent être éliminés ainsi que sur leur nombre (déphasage)
    - Peut mener à l'insatisfaction du client qui estime que son trafic suit bien le contrat

Mesurer le débit d'un flux (2)

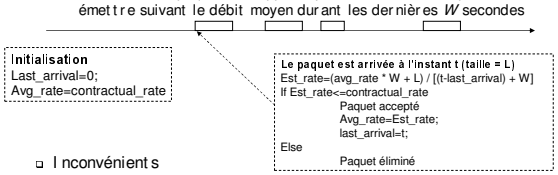
- Amélioration pour éliminer le problème de déphasage
  - Sur n'importe quelle fenêtre de W secondes, vous avez le droit de transmettre N octets (W: intervalle de temps, fenêtre coulissante)



  - Inconvénients
    - Il faut donc mémoriser les tailles des paquets sur les derniers *W* secondes
      - Instant d'arrivée de chaque paquet + garbage collector
    - Pas vraiment implémentée en réalité

Time Sliding Window

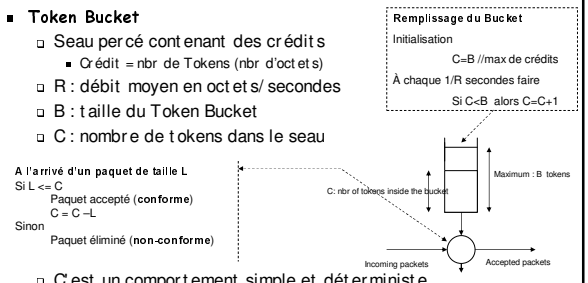
- Approximation de la solution précédente
  - Se souvenir des instants d'arrivée de tous les paquets n'est pas une chose simple
    - Estimation du débit moyen d'arrivée (idée approximative)
    - A l'arrivé d'un paquet, on suppose que le flux est fluide puis émettre suivant le débit moyen durant les dernières *W* secondes



  - Inconvénients
    - Solution approximative qui dépend de l'instant d'initialisation et de la valeur de *W*
  - C'est une solution qui marche en pratique

Token Bucket (1)

- **Token Bucket**
  - Seau percé contenant des crédits
    - Crédit = nbr de Tokens (nbr d'octets)
  - R : débit moyen en octets/ secondes
  - B : taille du Token Bucket
  - C : nombre de tokens dans le seau



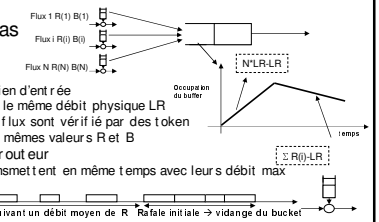
  - C'est un comportement simple et déterministe
    - Indépendant de l'instant d'initialisation
  - Durant une période de T secondes, le Token Bucket accepte au plus $B+(TxR)$ octets du trafic (Borne max)

Token Bucket (2)

- Avantages
  - Peut être utilisé par les fournisseurs d'accès à Internet pour vérifier les contrats de trafic
    - Paquets conformes au contrat
    - Paquets non-conformes au contrat
  - Permet une limitation du débit moyen
  - Permet de limiter le débit maximum du trafic durant une période de temps : B+(TxR)
    - Grâce à cela, il permet de fixer la taille des buffers à l'intérieur des routeurs

Dimensionnement du buffer

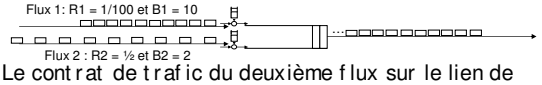
- Comment peut-on s'assurer qu'aucun paquet ne sera éliminé à cause d'un débordement du buffer (buffer overflow)?
- Analyse du pire cas
  - Simplifications
    - Un seul flux par lien d'entrée
    - Tous les liens ont le même débit physique LR
    - On suppose les N flux sont vérifié par des token buckets ayant les mêmes valeurs R et B
  - Le pire cas pour le routeur
    - Tous les flux transmettent en même temps avec leurs débit max



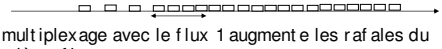
- Offrir des garanties de débit en tenant compte de ce pire cas que l'on sait estimer
  - Ca permet de dimensionner le buffer du 1er routeur mais pas pour les autres routeurs qui le suivent dans le réseau

Impact du multiplexage

- Que ce passe t'il lorsque les flux sont multiplexés à l'intérieur du routeur ?



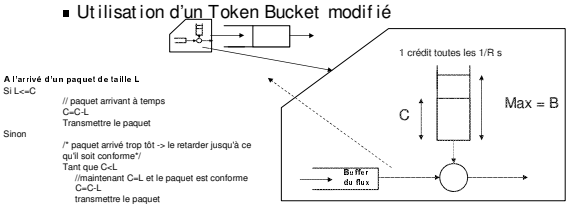
- Le contrat de trafic du deuxième flux sur le lien de sortie



  - Le multiplexage avec le flux 1 augmente les rafales du deuxième flux
- Dans le réseau, le taux de rafale d'un flux augmente à chaque routeur intermédiaire
  - Cela peut poser des problèmes pour le dimensionnement des buffers dans le réseau
  - Pouvoir dimensionner les buffers => pouvoir borner le délai

Le Token Bucket en mode lissage (shaping)

- Problème
  - Comment peut on s'assurer qu'un flux est toujours conforme à son contrat après multiplexage
    - Utilisation d'un Token Bucket modifié



    - Avantage : le trafic sera toujours conforme à la spécification du Token Bucket (Contrat R,B du trafic)

Mécanismes de contrôle du trafic au niveau paquet

Plan

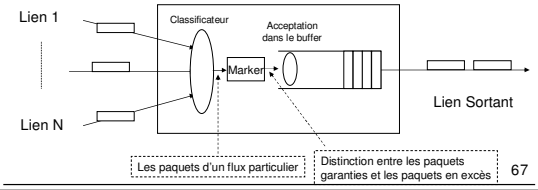
- Les différents types de garanties
  - Bande passante, Délai
- Le support du service Best Effort
- La garantie de la bande passante
  - Bande passante maximum
  - Bande passante minimum
- La garantie du délai

Comment offrir un débit minimal garantie ?

- Problème
  - A l'intérieur de chaque flux, on va avoir deux types de paquets
    - Les paquets IP qui font partie du minimum garantie
      - Ces paquets ne sont pas éliminés (par les routeurs)
    - Les paquets IP qui ne font pas partie du minimum garantie
      - Ces paquets sont traités en Best Effort et peuvent être éliminés si nécessaire
- Principe
  - Identifier chaque type de paquets (garanti ou non garanti)
  - Eliminer de façon préférentielle les paquets qui ne sont pas garanties

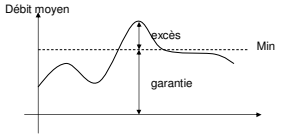
Identification des paquets garanties

- Comment faire ?
  - Mesurer le débit du flux en entrée
  - Identifier les paquets respectant le débit minimum
  - Identifier les paquets dépassant le débit minimum
  - Les paquets peuvent être marqués d'une façon explicite ou interne
- L'acceptation des paquets dans le buffer dépend de :
  - La charge du buffer
  - Est-ce que le paquet est dans le débit à garantir ou non



Comment marquer les paquets ?

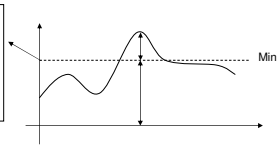
- Problème à résoudre
  - Comment marquer les paquets en fonction du débit moyen du flux entrant ?
  - Deux méthodes pour le marquage des paquets
    - Probabiliste
    - Déterministe



Marquage probabiliste

- Principe
  - Mesurer le taux moyen du flux entrant
    - En utilisant par exemple le *Time Sliding Window*
  - Marquer le paquet en fonction de la mesure

```
Algo de marquage
Si avg_rate ≤ min_rate
        Paquet garantie
Sinon
        P = (avg_rate – min_rate) /avg_rate
        Avec une probabilité P
                Marquer Paquet en excès
        Sinon
                Marquer Paquet garantie
```



Problème : Ne prend pas en considération la taille des paquets (Les paquets de 40 octets sont traités comme les paquets de 1500 octets) → On ne peut dimensionner le buffer correctement (sortie du marquage peut ne pas etre conforme (supérieur) au contrat de trafic BP min.

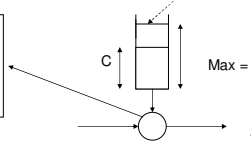
→Marchera bien sur de longues échelles de temps mais risque de ne pas marcher sur de petites échelles

Marquage déterministe

- Principe
  - Modifier le Token Bucket pour marquer les paquets non-conformes au contrat de trafic (au lieu de les éliminer)
    - Dans ce cas, on doit spécifier, en plus de la bande passante minimum, une taille du Token Bucket
  - Evite les aléas probabilistes de la solution précédente
    - Permet de pouvoir dimensionner les buffers dans les routeurs grâce aux spécificités des Token Buckets

```
Remplissage du Bucket
C=B // Initialisation
À chaque 1/R secondes faire
        Si C<B alors C=C+1
```

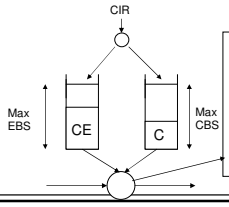
```
A l'arrivé d'un paquet de taille L
Si L <= C
        Paquet garantie
        C = C – L
Sinon
        Paquet en excès
```



Extension du Token Bucket (1)

- Les vendeurs ont implémenté des solutions un peu plus complexes avec 3 niveaux d'importance
- Première implémentation : Single rate three color marker
  - 2 tocken buckets et 3 paramètres
    - Committed Information Rate (CIR)
      - "Débit minimum" garantie (très important)
    - Committed Burst Size (CBS)
      - Rafale garantie (important)
    - Excess Burst Size (EBS)
      - Rafale d'excès (pas important)

```
Bucket filling
Initialisation
C=CBS
CE=EBS
Chaque 1/CIR secondes faire
        Si C<CBS C=C+1
        Sinon   Si CE<EBS
                CE=CE+1
                Sinon rien
```



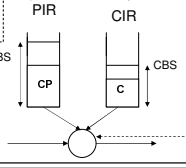

```
A l'arrivé d'un paquet P de taille L
Si L<=C
        paquet est vert (très important)
        C=C-L
Sinon
        Si L<= CE
                paquet jaune (important)
                CE=CE-L
                sinon
                        paquet rouge
```


Extension du Token Bucket (2)

- Seconde implémentation : Two rate three color marker
  - 2 tocken buckets et 4 paramètres
    - Committed Information Rate (CIR): **débit minimum** garantie
    - Committed Burst Size (CBS): rafale garantie
    - Peak Information Rate (PIR): débit de pointe
    - Peak Burst Size (PBS): rafale de pointe

```
Committed Bucket filling
Initialisation
C=CBS
Chaque 1/CIR secondes faire
        Si C<CBS C=C+1
```

```
Peak Bucket filling
Initialisation
CP=PBS
Chaque 1/PIR secondes faire
Si CP<PBS CP = CP +1
```



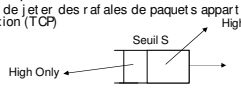

```
A l'arrivé d'un paquet P de taille L
Si L<CP
        paquet est rouge
Sinon Si L<C
                le paquet est jaune
                CP=CP-L
        Sinon
                le paquet est vert
                CP=CP-L
                C=C-L
```

Probabiliste *vs.* Déterministe

- Marquage probabiliste
  - Un marquage approximatif des paquets en fonction du débit moyen du flux
  - Adapté au comportement de TCP
    - Répartir le marquage sur différents paquets TCP → répartir les pertes et éviter les pertes en rafale
  - Pas de preuve mathématique (probabilité)
    - On peut accepter plus comme moins
  - Ca marche bien en moyenne !!!
- Marquage déterministe
  - Validé par des preuves mathématiques
  - Facile à mettre en œuvre
    - Implémentation hardware possible
  - Peut ne pas être la meilleur solution pour TCP
    - On peut jeter des rafales
      - Problème d'équité …
  - C'est bien pour le réseau mais pas forcement pour les flux

Elimination préférentielle des paquets (1)

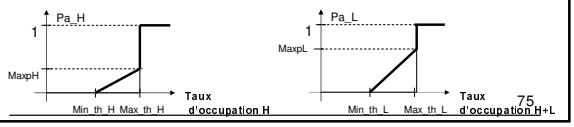
- Problème
  - Deux types de paquets
    - Paquets de **haute** et de **basse** priorité
  - Traiter en premier les paquets de haute priorité
- Solution
  - Eliminer en premier les paquets les moins prioritaires
- Solution 1 : Partial Buffer Sharing
  - Définition d'un seuil S
  - Lorsqu'un paquet arrive
    - Si l'occupation instantanée du buffer est inférieure à un seuil S → accepter le paquet entrant quelque soit sa priorité
      Sinon
      Si c'est un paquet de haute priorité → le laisse passer
      sinon on le jette
  - Simple mais possède les mêmes inconvénients que Tail Drop
    - Risque de jeter des rafales de paquets appartenant à une même connexion (TCP)



Partial Buffer Sharing

Elimination préférentielle des paquets (2)

- Solution 2 : Weighted RED
  - Une extension de RED pour **supporter une élimination préférentielle des paquets** en fonction de leur priorité
  - Exemple avec deux types d'éliminations préférentielle
    - Deux algo RED en parallèle (Initialement appelés RIO)
      - Le premier RED décide de l'acceptation des paquets de haute priorité (**rouge**)
      - Le deuxième RED décide de l'acceptation des paquets les moins prioritaires (**bleu**)
    - Avec comme précaution
      - Seuils : (Min_th_L, Max_th_L) <(Min_th_H, Max_th_H)
      - Probabilité de rejet : MaxpH <MaxpL



Mécanismes de contrôle du trafic au niveau paquet

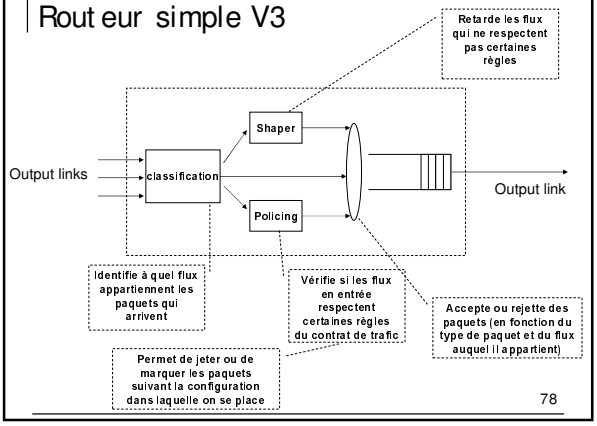
Plan

- Les différents types de garanties
  - Bande passante, Délai
- Le support du service Best Effort
- La garantie de la bande passante
  - Bande passante maximum
  - Bande passante minimum
- La garantie du délai

Vers des réseaux multiservices

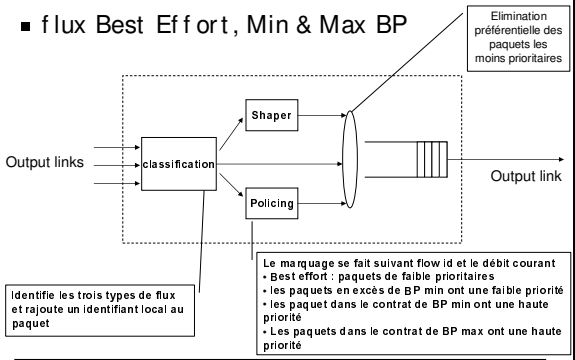
- Problèmes
  - Comment peut on multiplexer différents flux à travers un routeur classique Best Effort
    - Garantir que les paquets garanties ne soit pas perturbés par le trafic Best effort
    - Les paquets Best effort doivent aussi être capables d'utiliser le lien sortant lorsqu'il n y a pas de paquets garanties
- Comment modifier notre routeur V2 pour obtenir cette garantie ?

Routeur simple V3



Multiplexer avec le routeur V3

- flux Best Effort, Min & Max BP



Mécanismes de contrôle du trafic au niveau paquet

Plan

- Les différents types de garanties
  - Bande passante, Délai.
- Le support du service Best Effort.
- La garantie de la bande passante
  - Bande passante maximum
  - Bande passante minimum
- La garantie du délai

80

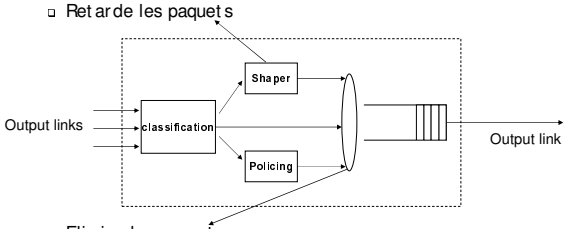
Vers des réseaux multiservices

- Problèmes auquel on a déjà répondu
  - Comment peut on offrir des garantie de BP et comment peut on multiplexer différents flux à travers un routeur classique Best Effort
    - Garantir que les paquets "garanties" ne soient pas perturbés par le flux Best effort
    - Les paquets Best effort doivent aussi être capables d'utiliser le lien sortant lorsqu'il n y a pas de paquets garanties
- Problèmes restants à résoudre
  - Comment peut-on garantir des délais différents ou comment modifier notre routeur pour obtenir cette garantie ?

81

Et la garantie de délai ?

- Les mécanismes supportés par le routeur
  - Retarde les paquets

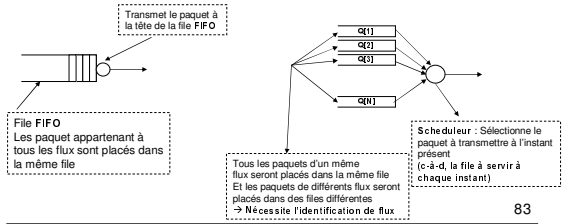


  - Elimine les paquets
- Comment peut-on faire pour s'assurer que les paquets prioritaires (Vol P par exemple, 150ms) soit transmis avant les paquets les moins prioritaires (FTP par exemple) ?

82

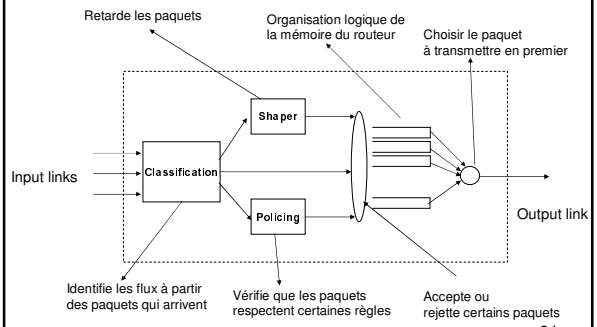
Et la garantie de délai ?

- Solution
  - Rajouter à la différentiation de pertes une différentiation de délais
    - Des paquets doivent être transmis avant les autres
    - Remplacer le buffer FIFO par plusieurs buffers + un scheduleur (ordonnanceur)



83

Routeur offrant de la QoS - bilan



84

Scheduler

- Sa fonction :
  - Sur toutes les files logiques contenant au moins un paquet, sélectionner le paquet qui doit être transmis
- Le scheduleur doit :
  1. Être facile à mettre en œuvre dans le matériel
  2. Être adaptable et donc supporte les services Best Effort et garanties
  3. Offrir une équité pour le trafic best effort
     - Une équité Max-Min serait souhaitable
  4. Offrir une protection entre les flux
     - Un flux ne doit pas monopoliser la bande passante et pénaliser d'autres flux
  5. Offrir des garanties statistiques ou déterministes
     - Bande passante, délai

Algorithmes du scheduler

- Deux types
  - Work-conserving scheduler (scheduleur conservatif)
    - S'il ya au moins un paquet à l'intérieur des files, alors on transmet le paquet
      - Intuitivement, des qu'il y a du travail à faire (des paquets à transmettre) le scheduleur le fait
  - Non-Work-conserving scheduler (scheduleur non-conservatif)
    - Le scheduleur peut choisir de ne pas transmettre un paquet dans le réseau (retarder la transmission)
      - Pour garantir une faible variabilité du délai (gigue)
      - Peut être implémenté en partie avec des Token Bucket
    - Intéressant du point de vue théorique mais pas implémenté en pratique

Ordonnancement des flux best effort

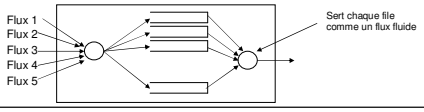
- Objectifs conceptuels
  - Offrir une distribution équitable de la bande passante entre les flux **best effort** actifs dans le routeur pour supporter une équité max-min dans le réseau
    - L'équité ne doit pas dépendre du comportement des mécanismes de contrôle de congestion dans les systèmes terminaux
  - Offrir une protection entre les flux
    - Un flux qui se comporte de façon erratique ne doit pas consommer une grande partie de bande passante (il ne doit consommer que ce qui lui est disponible)
      1. Le scheduleur assure une distribution équitable de la bande passante **en sortie** entre les différentes files
      2. Le mécanisme d'acceptation des paquets doit permettre de contrôler l'équité de l'**arrivée** des paquets à l'intérieur des files du routeur
  - Que ce soit implantable (en hardware) pour le haut débit

PS - Processor Sharing

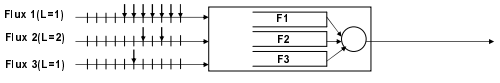
- Processor Sharing (PS)
  - Un scheduleur conservatif idéal
    - Idéal car il suppose que les flux sont fluides (flux de liquide)
      - Chaque file est servie par le scheduleur comme si elle contenait un fluide
      - Scheduleur sera le robinet contrôlant la sortie des différentes files
    - A l'instant t, Queue[i] est servie à un taux

$$rate[i] = \frac{link_{rate}}{N_{activeflows}}$$

      - Un flux est considéré actif si sa file contient au moins un bit en attente
      - Nombre de flux actifs va varier dynamiquement en fonction des I/O

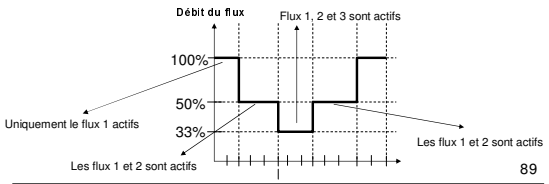


Exemple de Processor Sharing



Flux 3 et Flux 1 ont des paquets de taille unitaire (L=1)

Flux 2 a des paquets de taille double (L=2)

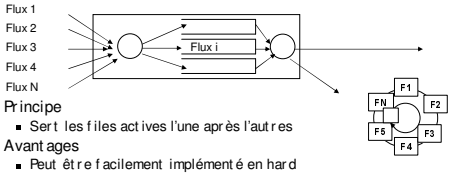


PS - Processor Sharing

- Avantage
  - Si aucun paquet n'est éliminé, un réseau avec un Processor Sharing offre un service à équité max-min
    - L'équité ne dépend pas du mécanisme de contrôle de congestion
      - Donc on ne prend aucune supposition sur les sources de trafic (TCP ou pas TCP)
    - Si les paquets sont éliminés, alors l'élimination doit se faire d'une manière équitable entre les flux
- Inconvénient
  - Bonne solution « mathématiquement », pas implémentée en réalité
    - – Supposition d'un modèle fluide et non paquets
    - + Il existe des approximations implémentables en pratique

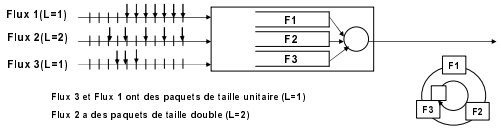
RR - Round Robin

- Round Robin (1ére approximation du PS)



  - Principe
    - Sert les files actives l'une après l'autres
  - Avantages
    - Peut être facilement implémenté en hard
    - Permet une protection du trafic best effort
    - Offre une distribution équitable de la bande passante à condition que les paquets soient de tailles fixes
      - De plus cette équité n'est offerte que pour des durées multiples du cycle de Scheduling → 4 Vs. 10.000 files à servir ☺
  - Inconvénient
    - L'équité n'est pas respectée avec des paquets de tailles variables

Exemple de Round Robin

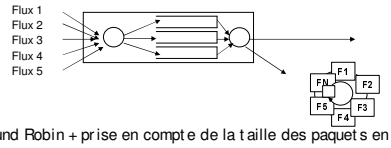


Flux 3 et Flux 1 ont des paquets de taille unitaire (L=1)
Flux 2 a des paquets de taille double (L=2)

T	In F1	In F2	In F3	Q(F1)	Q(F2)	Q(F3)	Scheduled	
0	P10[1]	P20[2]	-	P10	P20	-	F1:P10	
1	P11[1]	-	-	P11	P20	-	F2:P20	
2	P12[1]	P22[2]	-	P11,P12	P22	-	F2:P20	(cont)
3	P13[1]	-	-	P11,P12,P13	P22	-	F1:P11	
4	P14[1]	P24[2]	-	P12,P13,P14	P22,P24	-	F2:P22	
5	P15[1]	-	P35[1]	P12,P13,P14	P24	P35	F2:P22	(cont)
6	P16[1]	P26[2]	P36[1]	P12,P13,P14	P24,P26	P35,P36	F3:P35	
7	-	-	P37[1]	P12,P13,P14	P24,P26	P36,P37	F1:P12	
8	-	P28[2]	-	P13,P14,P15	P24,P26	P36,P37	F2:P24	
9	-	-	-	P13,P14,P15	P26,P28	P36,P37	F2:P24	
10	-	P2A[2]	-	P13,P14,P15	P26,P28	P36,P37	F3:P36	

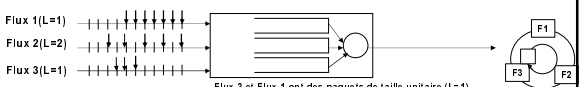
Service équitable en terme de nombre de paquets transmis mais non équitable en terme de nombre d'octets transmis

DRR – Deficit Round Robin



- Idée
  - Round Robin + prise en compte de la taille des paquets en tête de file
- Principe
  - Un compteur *d[i]* associé à chaque file – appelé le déficit de la file indiquant la quantité d'octets que la queue peut transmettre
  - Augmenter *d[i]* par un *quantum* à chaque fois que *queue[i]* est visité sauf si elle est vide
    - If first_packet of queue[i] larger than d[i] {
      packet stays in the queue[i]}
      else {
      packet is transmitted on output link;
      d[i] = d[i] – packet length;
      if queue[i] is empty {d[i] = 0;} }

Exemple de Deficit Round Robin



Flux 3 et Flux 1 ont des paquets de taille unitaire (L=1)
Flux 2 a des paquets de taille double (L=2)

T	Incrémenter	D[1]	D[2]	D[3]	Q(F1)	Q(F2)	Q(F3)	Scheduled
0	F1	0+1-1	0	0	P10	P20	-	F1:P10
1	F2,F1	0+1-1	0+1	0	P11	P20	-	F1:P11
2	F2	0	1+1-2	0	P12	P22	-	F2:P20
3	-	0	0	0	P13	P22	-	F2:P20(cont)
4	F1	0+1-1	0	0	P14	P22,P24	-	F1:P13
5	F2,F3	0	0+1	0+1-1	P14,P15	P22,P24	P35	F3:P35
6	F1	0+1-1	0	0	P14,P15,P16	P22,P24,P26	P36	F1:P14
7	F2	0	1+1-2	0	P15,P16	P22,P24,P26	P36,P37	F2:P22
8	-	0	0	0	P15,P16	P24,P26	P36,P37	F2:P22(cont)
9	F3	0	0	0+1-1	P15,P16	P24,P26	P36,P37	F3:P36
10	F1	0+1-1	0	0	P15,P16	P24,P26	P37	F1:P15
11	F2,F3	0	0+1	0+1-1	P15,P16	P24,P26	P37	F3:P37

Service équitable en terme de nombre d'octets transmis car le quantum est faible (1ø). En pratique quantum = 200 à 500ø à cause de la surcharge de calcul (équitable dans le temps)

Mécanisme d'éliminat ion des paquet s

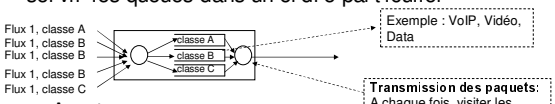
- Service équitable = scheduling équitable + éliminat ion équitable
- Comment peut on éliminer des paquets d'une façon équitable ?
  - Les paquets doivent être éliminés des flux provoquant la congestion
    - Comment peut on identifier ces flux ?
      - C'est ceux qui ont un débit d'entrée > débit de sortie
        - Donc, ceux qui ont le plus de paquets en attente dans la file
  - Si des paquets d'un flux entrent dans une file plus rapidement qu'ils en ressortent alors la file explose
    - Les flux avec les files les plus grandes sont responsables de la congestion
    - Élimination des paquets à partir des files les plus grandes
      - Dépend de la situation, des contraintes d'implem. et de perf.
      - A la fin de la file
        - C'est assez simple à implémenter (Si le paquet est destiné à la file la plus longue Alors l'éliminer au lieu de le mettre dans la file)
      - A la tête de la file
        - C'est parfois moins simple car dépend de comment les files sont implantées
        - Rappelons que cette stratégie peut être avantageuse pour TCP
      - Toute la file (la plus langue)
        - Très facile à implémenter (tête de file = Null)

Ordonnancement des flux garant ies

- Objectifs conceptuels
  - Support efficace des flux avec une garantie de débit minimale et maximale
    - Offre une garantie de débit
    - Offre une garantie de délai
  - Offrir une protection entre les flux
    - Le comportement d'un flux ne doit pas altérer les garanties des autres flux
  - Implémentable à haut débit

PBS : Priority-based Scheduler

- Simple priority scheduling : utilisant les priorités et servir les queues dans un ordre particulier



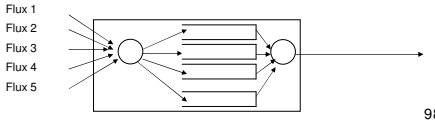
  - Avantages
    - Facile à mettre en œuvre
    - Les paquets avec une haute priorité auront un délai très faible (VoIP)
  - Inconvénients
    - Pas de protection entre les différents flux
    - Les flux de haute priorité peuvent "tuer" les flux de faible priorité
      - Ceci peut être souhaitable pour certains cas de figure
        - Exemple : donner la priorité absolue à la VoIP face aux Data
      - Doit être utilisé avec prudence (limitation du trafic de HP)

Un processor sharing généralisé (1)

- Generalized Processor Sharing (GPS)
  - Ordonnanceur conservatif idéal
    - Un poids ***W[i]*** est associé à chaque file ***queue[i]***
    - Chaque file est servie par le scheduleur comme si elle contenait un flux fluide
    - A l'instant ***t***, ***queue[i]*** est servie avec un taux :

$$rate\_queue[j] = link_{rate} \times \frac{W[j]}{\sum_{i=queues\_actives} W[i]}$$

    - Une file est active si elle contient quelque chose

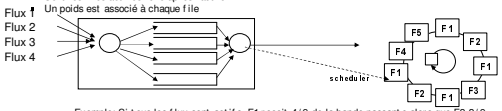


Un processor sharing généralisé (2)

- Avantages
  - Offre une garantie de *bande passante* par flux (à condition d'avoir une file par flux)
    - A travers un scheduleur GPS
    - A travers un réseau de scheduleurs GPS
  - Offre une garantie de *délai* par flux
    - Pour avoir une garantie de délai il faut contraindre le trafic du flux (flux non infini)
    - Si le trafic est contraint par un TB (R,B) alors avec GPS le délai max = B/R
    - La garantie de délai max est valable aussi bien pour un seul GPS que pour un réseau de GPS (délais de propagations négligés)
  - Offre une borne sur l'utilisation du buffer → taille Max = B
  - Offre une protection entre les différents flux
    - Un flux ne peut pas monopoliser les ressources (indépendance totale)
- Inconvénients
  - C'est un scheduleur idéal non implémentable en pratique
  - Pas de garantie sur la variabilité de délai (gigue)

WRR - Weigthed Round Robin

- Weigthed Round Robin (1ére approximation de GPS)
  - Principe
    - Sert les files actives l'une après l'autre
    - Un poids est associé à chaque file



  - Exemple: Si tous les flux sont actifs, F1 reçoit 4/9 de la bande passante alors que F2 2/9, F3, F4 et F5 1/9
- Avantages
  - Facile à mettre en œuvre avec un petit scheduling
    - Complexité dépendante de la taille du cercle (nombre de fractions)
  - Différents poids offrent une différentiation de bandes passantes
  - Protection inter-flux
  - Deficit Round Robin peut être étendu pour supporter les poids (prendre en compte des tailles variables de paquets)
- Inconvénients
  - Un scheduling assez long est nécessaire s'il y a beaucoup de flux avec de petites bandes passantes (Avec 2 files actives parmi 2000 il faudra tout parcourir) → Complexe

WFQ - Weighted Fair Queuing

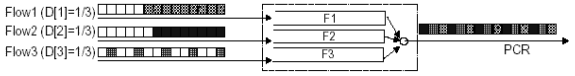
- But
  - Définir une approximation de GPS « implémentable »
  - Une approximation plus proche de GPS que WRR
- Idée
  - **Simuler** le fonctionnement de GPS paquet par paquet
  - Servir les paquets approximativement de la même façon que ce qui est fait dans GPS
- Comment faire ceci ?
  - GPS sert les files sous la forme de fluides (c-à-d octet par octet)
  - WFQ va donc calculer à tout moment à quel temps il faut servir chaque paquet à la tête des files d'attente
    - Ceci se fait en calculant les dates de fin de service de chaque paquet et en les servant en fonction de cela
  - WFQ reste assez complexe à mettre en place en pratique

VC - Virtual Clock (1)

- Approximation de GPS (implémentation plus simple que WFQ)
  - Essayer d'avoir une approximation (implémentation) ayant une complexité raisonnable
  - Idée
    - Associer un *timestamp* à chaque arrivée de paquet
    - Le scheduler sélectionne le paquet ayant le plus petit *timestamp*

    ⇨Toute la complexité de l'algorithme va résider dans le calcul adéquat du *timestamp* pour approximer GPS
  - Première solution
    - *D[i]* : bande passante associée a la file *File[i]*
    - *V[i]* : variable d'état associée à la file *File[i]*
    - Arrivée d'un paquet de taille *P* bytes dans la file *File[i]*
      - *V[i] = V[i] + (P/ D[i])*
      - Associée *V[i]* au paquet comme un timestamp
    - Scheduleur
      - Sélectionne le paquet ayant le plus petit timestamp pour le transmettre

Virtual Clock - exemple 1-

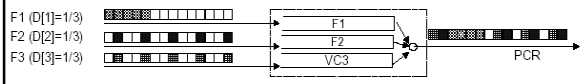


T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	0+3	3	0+3	3	0+3	3	F1
1	3+3	6	3+3	3,6	3	3	F3
2	6+3	6,9	6+3	3,6,9	3	-	F2
3	9+3	6,9,12	9+3	6,9,12	3+3	6	F1
4	12+3	9,12,15	12+3	6,9,12,15	6	6	F3
5	15+3	9,12,15,18	15+3	6,9,12,15,18	6	-	F2
6	18+3	9,12,15,18,21	18+3	9,12,15,18,21	6+3	9	F1
7	21+3	12,15,18,...	21+3	9,12,15,18,21	9	9	F3
8	24+3	12,15,18,...	24+3	9,12,15,18,...	9	-	F2
9	....						

Pour cet exemple VC suit bien la répartition voulue initialement pour chaque file d'attente.

La question est de savoir si VC approxime bien GPS dans tous les cas ? 103

Virtual Clock - exemple 2 -



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	0	-	0+3	3	0+3	3	F2
1	0	-	3	-	3	3	F3
2	0	-	3	-	3	-	-
3	0	-	3+3	6	3+3	6	F2
4	0	-	6	-	6	6	F3
5	0	-	6	-	6	-	-
6	0	-	6+3	9	6+3	9	F2
7	0	-	9	-	9	9	F3
8	0	-	9	-	9	-	-
9	0+3	3	9+3	12	9+3	12	F1
10	3+3	6	12	12	12	12	F1
11	6+3	9	12	12	12	12	F1
12	9+3	12	12+3	12,15	12+3	12,15	F1
13	12+3	15	15	12,15	15	12,15	F2

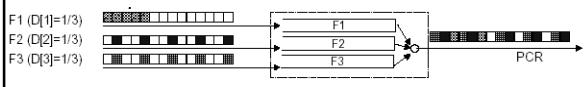
104

Virtual Clock - exemple 2 (suite) -

- Le problème
  - Si F1 reste inactif pendant 1.000.000 d'unités de temps et qu'il commence à envoyer des rafales => il accaparera toute la bande passante pendant un long moment
  - Ce n'est pas une bonne approximation de GPS
  - Ceci est dû au fait que *V[1]* n'évolue pas pendant le temps ou les autres *V[i]* évoluent
- Seconde solution
  - A l'arrivé (instant t) d'un paquet de taille P
    - *V[i] = max (V[i],t) + (P/ D[i])*

105

Virtual Clock - exemple 2 (suite) -

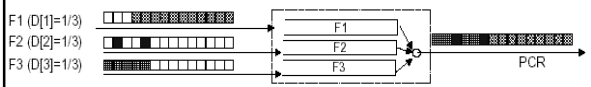


T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	0	-	max(0,0)+3	3	max(0,0)+3	3	F2
1	0	-	3	-	3	3	F3
2	0	-	3	-	3	-	-
3	...						
9	max(0,9)+3	12	max(9,9)+3	12	max(9,9)+3	12	F1
10	max(12,10)+3	15	12	12	12	12	F2
11	max(15,11)+3	15,18	12	-	12	12	F3
12	max(18,12)+3	15,...	max(12,12)+3	15	max(12,12)+3	15	F1
13	max(21,13)+3	18,...	15	15	15	15	F2

Le problème d'effet mémoire (de F[2] et F[3]) est donc résolu!

La question est de savoir si cette solution est satisfaisante dans tous les cas ? 106

Virtual Clock - exemple 3 -



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	max(0,0)+3	3	0	-	0	-	F1
1	max(3,1)+3	6	0	-	0	-	F1
2	max(6,2)+3	9	0	-	0	-	F1
...							
8	max(24,8)+3	27	0	-,	0	-	F1
9	max(27,9)+3	30	max(0,9)+3	12	max(0,9)+3	12	F2
10	max(30,10)+3	30,33	12	-	max(12,10)+3	12,15	F3
11	33	30,33	12	-	max(15,11)+3	15,18	F3
12	33	30,33	max(12,12)+3	15	max(18,12)+3	18,21	F2
13	33	30,33	15	-	max(21,13)+3	18,21,24	F3
14	33	30,33	15	-	max(24,14)+3	21,24,27	F3

Problème inverse à tout à l'heure. L'effet mémoire pénalise F[1]. Ceci n'est pas équitable au sens GPS (garantie de BP minimal). 107

SCFQ / Virtual Spacing

- SCFQ : Self Clocked Fair Queuing
- Approximation de GPS
  - Associe un *timestamp* à chaque paquet qui arrive
  - Le scheduler sélectionne le paquet avec le plus petit *timestamp*
  - Algorithme
    - *D[i]* : bande passante associée à la file *File[i]*
    - *V[i]* : variable d'état associée avec la file *File[i]*
    - *V* : variable d'état associée au scheduleur
      - A chaque instant t, *V* est égale au *timestamp* du paquet qui est en transmission
    - Arrivée d'un paquet de taille *P* bytes dans la file *File[i]*
      - *V[i] = max(V[i],V) + (P/ D[i])*
      - Associée *V[i]* au paquet qui arrive
    - Scheduleur
      - Sélectionne le paquet ayant le plus petit *timestamp* pour le transmettre

108

Garanties

- PBS → pas de protection entre les flux
- WRR → problème d'équité au niveau de la garantie de bande passante

- Scheduleur garantissant une bande passante par flux, et une protection entre les flux
  - GPS
  - WFQ
  - SCFQ
  - Deficit-WRR

- La garantie de délai est complètement indépendante du comportement des flux garanties ou des flux Best effort
  - Un flux ne peut pas monopoliser la bande passante
    - Nécessité d'avoir un bon mécanisme d'acceptation des paquets dans les files (buffers)

Garanties - 2

- Les garanties de délai sont possibles uniquement pour les flux respectant un modèle de trafic - Token Bucket (B,R)
  - Ces garanties sont indépendantes des autres flux
- Délai sur une suite de « n » scheduleurs
  - GPS $\frac{B}{R}$
  - WFQ / PGPS $\frac{B + n \times P_{max}}{R} + \sum_{i=1}^{n} \frac{P_{max}}{C_i}$
  - Virtual Clock $\frac{B + n \times P_{max}}{R} + \sum_{i=1}^{n} \frac{P_{max}}{C_i}$
  - SCFQ $\frac{B + n \times P_{max}}{R} + \sum_{i=1}^{n} \frac{K_i \times P_{max}}{C_i}$

- P_{max} : taille maximale des paquets
- C_i : Débit de sortie du nœud i
- K_i : nombre de flux dans le nœud i
- Token bucket (B,R)

Contrôle de trafic et la QoS dans un réseau IP - mécanismes de contrôle de trafic au niveau paquet