RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'IBN KHALDOUN - TIARET

Département Informatique

Spécialité : Informatique

Option : Réseaux et Télécommunication

Mémoire de fin d'études en vue de l'obtention du diplôme de Master

Thème

Développement d'algorithmes d'ordonnancement des paquets IP pour un routeur QoS

Rédigé par :

- CHAIB Mostefa
- BENKHELIFA Benbrahim

Dirigé par :

- **Mr.** NASSANE Samir

Année universitaire 2015-2016

Table des matières

Liste de figures

Liste des tableaux

Introduction générale

Au 21ème siècle l'internet est devenue l'infrastructure de communication universelle. Elle devient un outil indispensable dans notre vie et notre travail. Mais le déploiement de nouvelles applications IP temps réel et interactives comme le télé-enseignement, la vidéoconférence, la téléphonie sur IP, la multimédia et autres relèvent de nouveaux défis en exigeant des demandes de ressources très différentes et précises par rapport à celles pour laquelle internet a été conçu.

Le traitement actuel des flux de la même manière par l'internet n'offre qu'un seul niveau de service. Celui-ci ni guère adapté à la téléphonie sur IP qui exige délai et un taux de perte minimale au-delà de lequel il est inutilisable. Ou aux transactions bancaires qui exigent un traitement particulier. Offrir plusieurs niveaux de services est le fondement de la qualité de service QoS.

Dans le but de doter l'internet par une QoS, en 1994, l'IETF a créé plusieurs groupes de travail pour définir une architecture à intégration de services IntServ. Cette technologie a très vite révélé des lacunes. Par conséquent un autre groupe de travail a été créé en 1997 pour définir un modèle de différentiation de services DiffServ. Mais La communauté d'internet ne cesse de proposer d'autre technologies et modèles pour garantir une bonne QoS et créer l'internet multiservices.

Toutes architectures cherchant à offrir les capacités de garantir une QoS dans le réseau doit décrire des fonctions et mécanismes élémentaires et nécessaires à leurs aboutissements. L'identification d'un flux IP, son caractérisation, le mécanisme d'ordonnancement et le mécanisme de gestion des files d'attente.

Notre travail s'inscrit dans le mécanisme d'ordonnancement qui sert à satisfaire relativement la qualité de service réclamé par la majorité des flux transitant dans internet. L'ordonnancement sert essentiellement à contrôler la distribution des ressources entre les flux ou les classes de services. Plusieurs ouvrages traitant le problème d'ordonnancement dans les nœuds du réseau ont été élaborés. Ainsi plusieurs techniques ont été développées, soit pour contrôler le partage des ressources, soit pour isoler les classes de services ou bien pour réduire le temps d'attente dans les files ou encore pour minimiser le taux de perte de certains flux.

Cette mémoire est constituée de trois parties. La première traite la notion de qualité de service et les différentes technologies et modèles développées actuellement. Dans la deuxième partie, les principaux algorithmes d'ordonnancement sont détaillés par leurs techniques et mécanismes de mise en œuvre selon les documents et travaux publiés. La dernière partie est consacrée à la conception et le développement d'une application de simulation des principaux algorithmes d'ordonnancemnt.

Chapitre 01 : Qualité de Service

1. Introduction :

Quand l'internet a vu le jour, le principal but était d'acheminer des données d'une source vers sa destination sans se préoccuper du temps de transit. Ainsi, Les paquets circulent de nœud en nœud y subissant le même traitement quel que soit leur type ou destination. Aujourd'hui la diversification des flux traversant le réseau exige, de plus, une qualité de service en termes de délai, débit ou perte selon leur type, ces dernières contraintes ne sont pas satisfaites par le service BE (Best-Effort) offert nativement par internet.

L'apparition de la voix sur IP, la vidéoconférence le streaming et autres types de flux a orienté les chercheurs et les concepteurs vers deux plans de travail. Ou bien investir dans de nouveaux supports de transmission plus performante en débit : fibre, satellite etc. Ou bien améliorer les architectures actuelles et revoir les protocoles et les disciplines d'acheminement de données et leurs algorithmes. C'est dans le deuxième sens que la qualité de service s'est orientée, car quel que soit la performance des supports et des ressources, d'autres applications viendront probablement les consommer, et l'augmentation du trafic converge vers l'encombrement et la surcharge du réseau et on sera toujours en mesure d'exiger des qualités de réceptions d'information. En résumé le réseau a besoin d'un mécanisme de QoS répondant à leurs besoins, les volumes de trafic étant importants, ce mécanisme doit pouvoir passer à l'échelle.

2. Définition :

Une définition informelle de la qualité de service dans un réseau est : l'attribution des ressources aux flux en fonction de leurs besoins. Ou bien intuitivement le pouvoir d'offrir des garanties de performances aux applications en différentiant leurs besoins.

La qualité de service est l'effet de satisfaire les besoins d'un utilisateur de service (application, hôte,…) [LTO]. Autrement dit : un réseau est qualifié de 'qualité de service' lorsqu'il est capable de répondre aux attentes d'une application à besoins spécifiques [ELO]. La définition donnée dans la recommandation de l'UIT.TE.800, (Union International des Télécommunications) généralise le concept de la qualité de service et la considère comme un « ensemble des caractéristiques d'une entité qui lui permettent de satisfaire aux besoins explicites et aux besoins implicites ».

3. Critères :

Les besoins en qualité de service sont basées sur plusieurs facteurs et paramètres qui doivent être garanties totalement ou partiellement. Les principaux paramètres ou métriques qui influent

sur la performance du réseau et par conséquent sur la qualité de service sont le débit, le délai, la gigue, le taux de perte, la disponibilité etc.… [MAG1]. Ces derniers éléments peuvent être offerts, plus au moins, au sein d'un seul réseau, mais leur assurance de bout en bout reste à considérés. En effet, les composants constituants une connexion de bout en bout influencent sur le rendu final de la qualité de service : La variabilité des équipements terminaux, la combinaison des supports d'accès et de la technologie utilisée (sans fil, câble optique ou métallique, ADSL, …), les technologies des fournisseurs d'accès et systèmes autonomes (commutation, IP, ..), type de trafic et routage contribuent énormément à la rentabilité de la QoS. D'où le besoin de la normalisation des architectures et modèles de QoS.

3.1. Délai :

C'est le temps total nécessaire à la transmission d'un paquet depuis sa source jusqu'à sa destination. Le délai peut concerner une liaison de bout en bout, comme il peut désigner un élément particulier dans le réseau. Un autre terme est souvent employé dans la terminologie des réseaux, c'est le temps nécessaire à un aller-retour, il s'appelle RTT (*Round Trip Time*).

La diversité de l'information qui circule dans le réseau en matière de flux, de nature ou de fréquence influe sur les demandes de qualité de service. La téléphonie par internet, la vidéoconférence et le streaming par exemple requiert des garanties temporelles et sont sensibles au délai, à la gigue et aux taux de perte. Garantir le délai nécessite des mécanismes de traitement et d'acheminement de l'information dans un temps minimal et doivent prendre en considération que le délai englobe le temps de propagation, de transmission et le temps d'attente au sein des nœuds du réseau.

3.2. Taux de perte :

C'est le nombre de paquets n'arrivant pas correctement ou bien du tous à leurs destination par rapport à ceux ayant acheminé leur trajet. Ce phénomène est dû à deux évènements :

- La congestion où les équipements doivent supprimer des paquets moins chanceux, faute de stockage dans les files d'attente
- Le rejet en cas d'erreurs, après calcul des sommes de contrôle non correctes,

3.3. Gigue :

C'est la variation dans le délai du transfert d'un paquet IP. Il caractérise l'instabilité du réseau. Ce paramètre peut être appliqué à une connexion de bout en bout ou à un élément particulier dans le réseau.

3.4. Débit

Les applications qui consomment plus de bande passante ne cessent de s'accroitre en ralentissant ou bloquant d'autres applications moins consommatrices. La capacité d'un réseau en bande passante doit être suffisamment importante pour laisser transiter les différents types d'information sans retard et sans perte. Cependant la bande passante d'un chemin est égale à la bande passante minimale de tous les liens constituant le chemin, donc les propriétés du lien physiques et de configuration influencent grandement sur cette métrique. Des disciplines de contrôle de flot doivent être envisagées dans les routeurs selon le besoin de consommation des applications en débit.

En outre, la garantie de la qualité de service est relativement exposée à autres facteurs qui varient selon les architectures et les modèles envisagés par exemple :

- La garantie de QoS de bout en bout appliqué dans l'architecture IntServ ou appliqué à des domaines particuliers dans l'architecture DiffServ.
- L'application d'une QoS pour tous les flux ou pour des sessions particulières.
- Garantir une QoS unidirectionnelle ou bidirectionnelle.

4. Classification des flots :

Un champ ToS (Type Of Service) de huit bits est défini dans l'entête IP pour spécifier les paramètres de qualité de service voulu [RFC791] : Trois bits pour caractériser la priorité d'un paquet (PRECEDENCE) ; les trois bits suivants sont destinés pour contrôler le retard, le délai et la fiabilité d'une connexion ; les deux derniers sont inutilisés. Le document [RFC1349] redéfini l'octet ToS et consacre les quatre bits, après ceux de la priorité, au contrôle du débit, délai, fiabilité et le coût monétaire et laisse un bit non utilisé.

Mais ce champ ToS n'était pas pleinement exploité par la communauté de l'internet et ces participants. En effet, les services offerts par l'internet étaient suffisants et les besoins des applications en QoS n'ont pas été si massifs qu'aujourd'hui. D'autre part, la recommandation ne spécifié pas le comportement adéquat pour chaque type de service, ainsi chaque routeur ou administrateur fait de son mieux pour gérer les flux portant un ToS significatif ou ils ne s'en préoccupent guère. Mais après l'apparition du document [RFC2474] et la nécessité d'introduire la QoS dans les réseaux, toutes les propositions et les recommandations concernant le champ ToS sont devenues obsolètes, et il est devenu désormais le champ DS, pour (Differentiated Services Field).

En effet, L'internet aujourd'hui s'est transformé progressivement en une infrastructure commerciale. Les applications utilisant internet ont des profils de trafic et des besoins variables.

Ce qui a conduit à différencier certaines nouvelles classes de services que des entreprises sont prêtes à payer pour déployer les applications nécessitantes ces besoins spécifiques. Trois profils de trafic sont distingués :

4.1. Trafic temps réel :

L'utilisation des applications de jeux ou de la visioconférence requiert une bonne fluidité dans les échanges de données et donc un temps de réponse faible tout en tolérant un faible taux de perte. Cela suppose donc, un service réseau avec des bandes passantes et des délais constants, des pertes limitées, des temps de parcours variant faiblement dans le réseau (gigue).Ces applications génèrent du trafic dit « temps réel ».

4.2. Trafic élastique :

Le mail, les transferts de fichiers n'ont pas les mêmes besoins, ils savent s'adapter à un délai un peu plus important. L'indicateur de bon déroulement du transfert serait ici le fait que ce dernier soit terminé correctement et que toutes les données aient été acheminées et soient exploitables. Aucune perte n'est tolérée mais on peut en revanche attendre le fichier un peu plus longtemps. On dit que ce trafic est « élastique » car il est capable de s'adapter aux fluctuations importantes du réseau. Sa seule contrainte étant de ne pas supporter de perte.

4.3. Trafic temps réel adaptatif :

Les applications de streaming permettent de jouer une vidéo ou un fichier audio à la demande alors qu'il se trouve stocké dans le réseau. Pour ces applications, l'utilisateur accepte d'attendre avant que la diffusion commence mais ensuite, le flux doit être joué de façon fluide. Pour cela, l'application qui va jouer la vidéo commence par une phase de stockage des données au fil de leur arrivée, puis, quand elle en a reçu suffisamment, la vidéo commence à être lue. Cela permet d'atténuer les variations de performances du réseau. Cependant une fois que le flux commence à être joué, les exigences de l'utilisateur sont proches du temps réel. Nous retrouvons ici le comportement des deux types de flux précédent : au début c'est élastique mais ensuite plutôt temps réel. On dit que ce trafic est partiellement élastique.

4.4. Protocoles et architectures :

Pour mieux appréhender les différentes classes il faut présenter les architectures de QoS existantes. L'IETF (Internet Engineering Task Force) a proposé plusieurs modèles et architectures répondant au besoin de QoS et qui peuvent satisfaire la majorité des types de trafic cités précédemment :

1- les services intégrés IntServ (Integrated Service Protocol).
2- les services différenciés DiffServ (Differentiated Service Protocol).

3- la commutation de labels multi protocoles MPLS (Multi Protocol Label Switching).
4- l'ingénierie de trafic TE (Traffic engineering) et ses protocoles.

5. Les services intégrés et signalisation RSVP :

Ce modèle proposé par l'IETF dans [IS] est basé sur la réservation de ressources afin de fournir à chaque flux de données une QoS qui est lui spécifique. Pratiquement ce protocole dédie une partie de la bande passante et réserve un chemin pour assurer l'acheminement des messages prioritaires. Pour mettre en œuvre ce mécanisme, les nœuds du réseau doivent établir et exécuter certains protocoles et des disciplines :

- Un protocole de réservation de ressource qui, de façon implicite, signalise le chemin à établir en sollicitant des réservations de bande passante sur chaque routeur traversé du réseau. Le protocole RSVP (Resource Reservation Protocol) est généralement utilisé.
- Le contrôle d'admission permet d'autoriser ou non l'injection d'un nouveau flot dans le réseau, muni de sa QoS, sans perturber les QoS des autres flots existant.
- La classification des paquets admis dans des files d'attente spécifiques, afin d'appliquer sur eux le traitement adéquat via l'ordonnanceur.
- L'ordonnanceur de paquets qui détermine l'ordre de servir les paquets à transmettre sur la ligne de sortie en fonction de la QoS demandée.
- Un régulateur de trafic (*shaper*) permet à une source de respecter une certaine enveloppe de trafic. Celle-ci est définie principalement par son débit de pointe, son débit moyen et la taille de sa plus grande rafale. Un régulateur peut aussi être utilisé à la sortie d'un routeur pour remettre en forme le trafic avant que celui-ci ne transite dans un autre domaine.
- Un contrôleur de trafic (*Policer*) permet de vérifier que l'enveloppe de trafic est respectée à l'entrée d'un domaine.

Ce modèle garantit deux types de services à côté du service offert nativement par l'internet :

- ***GUARANTEED SERVICE (GS) :*** Qui garantit la bande passante et un délai d'acheminement limité.
- ***CONTROLLED LOAD* (CL)** : Pour les services temps réels tolérant une variation du délai de transit. Il est équivalent à un service Best Effort dans un environnement non surchargé.
- ***BEST EFFORT (BE) :*** Qui ne garantit pas une qualité de service.

5.1. Le protocole RSVP :

Le signal RSVP étant constitué par l'information de contrôle de la QoS, celui-ci propose des directives afin de mettre en place la réservation mais ne dit pas comment la mettre en place, ce sont les routeurs du réseau qui prennent en compte la signalisation RSVP.

Des messages périodiques sont transmis afin de maintenir dynamiquement le chemin réservé par les routeurs.

Notons que RSVP oblige le récepteur de faire la demande QoS, Cela est très utile dans le cas d'une transmission multicast. En effet, dans le cas où on aurait prévu que la demande de ressources soit faite par l'émetteur, une QoS identique à tous les récepteurs aurait été mise en œuvre et n'aurait pas été adaptée aux besoins de chacun d'eux. En revanche, la tendance à demander la meilleure QoS par les récepteurs pose problème. L'avantage dans le fait que le récepteur détermine les ressources dont il a besoin permet une facturation différenciée par récepteur.

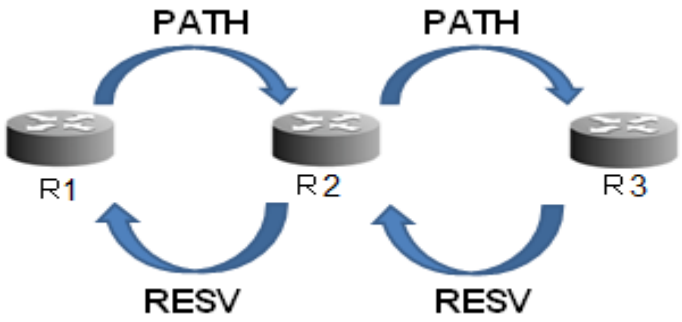


Figure 1: Messages échangés lors de la réservation de ressources dans RSVP

RSVP travaille au-dessus de IP (IPv4 ou IPv6) et occupe la place d'un protocole de transport dans la pile des protocoles mais ne transporte pas de données utilisateurs et peut être encapsulé dans des paquets UDP. Il passe de façon transparente les routeurs non RSVP. Les messages RSVP seront traités en détails dans la section consacrée à la technologie MPLS.

5.2. Limitations :

Le protocole RSVP est obligé de maintenir l'état d'un flot. En effet, dès l'ouverture d'une session, un chemin est établi grâce au contrôleur d'admission, et une file d'attente lui est attribuée dans chaque routeur. Ce chemin doit rester le même, tant que la session est utilisée. L'augmentation du nombre des états à maintenir et le rafraîchissement de ses états, en conséquent, dégrade les performances du réseau.

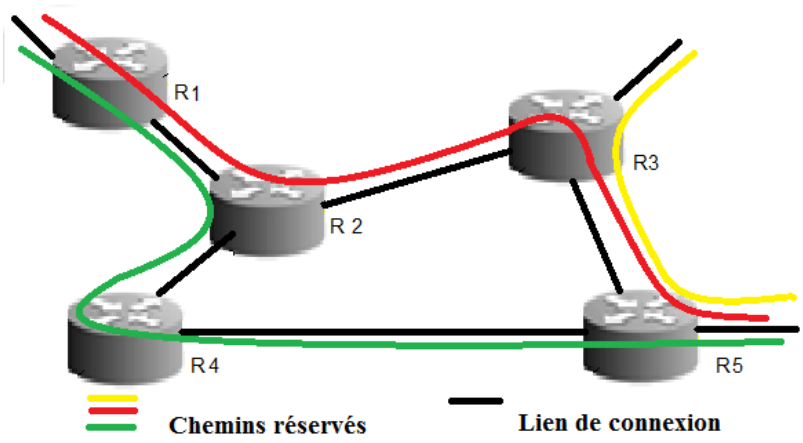


Figure 2 Maintien des états dans IntServ

De plus, chaque routeur doit disposer des premiers quatre mécanismes cités au-dessus (§5), ce qui ne facilite pas l'exploitation du réseau. Il est donc plus adapté pour de petits réseaux (LAN) et pas à l'internet dans son ensemble. Donc IntServ ne permet pas le passage à l'échelle qui est un aspect très important vu l'accroissement des réseaux de grande envergure et l'augmentation perpétuelle du trafic.

6. Les services différenciés :

La meilleure façon de résoudre la majorité de problèmes rencontrés dans IntServ consiste à ne plus traiter les flux de données individuellement, mais de les regrouper en classes qui agrégeront un ensemble de flux demandant le même type de QoS. En d'autres terme préparer les offres QoS et laisser les applications les utiliser ou non selon leur besoins ; c'est le dimensionnement ou la différentiation de service. Ce modèle est standardisé par l'IETF dé 1989 dans les documents [RFC2474] et [RFC2475].

Dans ce modèle, chaque classe est identifiée par une valeur codée dans l'entête IP. La classification doit se faire sur les routeurs de bordure ER (Edge Router) à l'entrée du réseau.

Six bits du champ ToS (Type of Service), rebaptisé DS, sont prévus à cet effet dans l'en-tête standard des paquets IPv4 ; ce qui permet potentiellement d'identifier 64 services différents. Ces six bits constituent le DSCP (DiffServ Code Point). L'en-tête des paquets IPv6 prévoit également 6 bits à cet effet. L'intérêt du marquage est de placer un code indiquant le traitement à fournir au paquet directement dans son en-tête, ce qui n'augmente pas le temps de traitement d'un paquet dans le routeur. C'est donc le paquet qui transporte l'information.

La QoS est garantie de bout en bout, c'est à dire de la source à la destination, ce qui veut dire que le paramétrage des réseaux traversés doit être cohérent. La complexité est concentrée dans

le dimensionnement des offres et le paramétrage, surtout dans les équipements de bordure des réseaux.

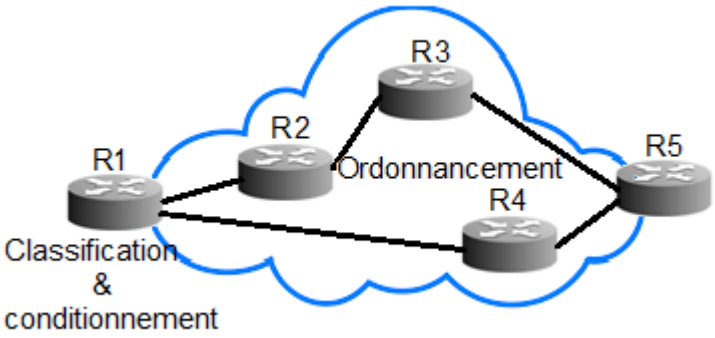


Figure 3 Architecture DiffServ

Ce modèle correspond très bien au modèle commercial actuel. En effet, une entreprise qui demande des services précis de son fournisseur de service Internet négociera avec celui-ci un contrat, portant globalement sur la quantité de trafic qui pourra être échangée dans chaque classe de service disponible chez cet ISP. Il est en effet classique qu'une organisation paie pour un débit global à l'accès, plutôt que pour chaque flux de données. Cela facilite aussi grandement la facturation. Le contrat négocié entre le client et le fournisseur de services (ISP) définit le trafic échangé à l'interface avec celui-ci. Il comprend typiquement une description de l'enveloppe de trafic admise (débit de pointe, débit moyen, plus grande rafale, …). C'est sur cette base que l'ISP vérifiera si le client respecte son contrat et, le cas échéant, détruira ou dégradera les paquets non conformes.

Trois principaux services sont définis dans le modèle DiffServ et cela selon la classification faite précédemment :

- Le service premium qui correspond au trafic temps réel.
- Le service assuré qui correspond au trafic tems réel adaptatif.
- Le simple service d'inter connectivité qui correspond au trafic élastique.

6.1. Mécanismes de gestion de la QoS dans DiffServ:

Le modèle DiffServ définie certains mécanismes dont le traitement se fait soit à l'entrée du réseau par les routeurs de bordures soit au niveau de ceux du domaine :

6.1.1. Classification et conditionnement :

Dans un réseau de QoS les routeurs de bordure ont la principale mission de surveiller et conditionner le trafic entrant afin de limiter la quantité de trafic injecté dans chaque classe de service. Un marquage des paquets admis dans le réseau en résulte s'ils sont conformes (marquage :

Mark), sinon une remise en forme (lissage : *Shap*) est effectuée ou une suppression de ceux dépassant les normes (Rejet : *Drop*).

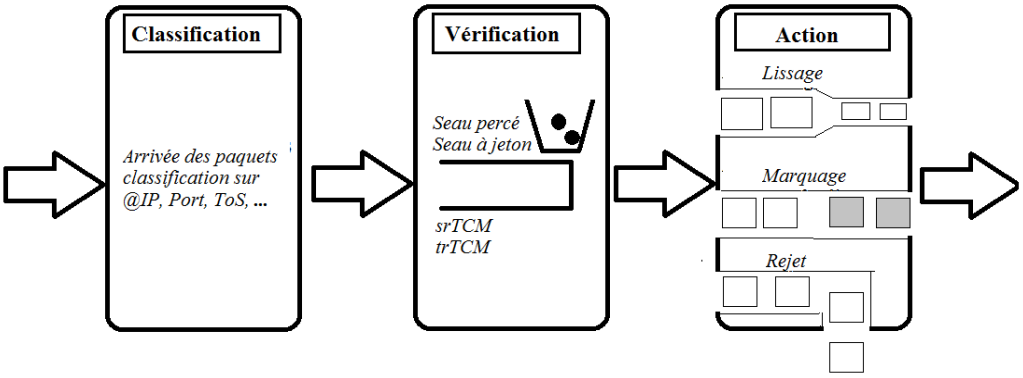


Figure 4: Tâches d'un routeur de bordure

6.1.2. Traffic shaping (mise en forme du trafic) :

Le but est d'éviter les longues rafales de paquets en introduisant un délai entre les émissions de ceux-ci. Les mécanismes les plus simples, devenus de référence, implémentés sont :

- **Leaky Bucket (Seau percé)** :

Consiste à permettre le passage de quantités régulières dans des intervalles réguliers. Ce mécanisme a été introduit par Jonathan S. Turner [JTU]. Pour chaque flot de données, un tampon de taille β limite le nombre de paquets en attente, par l'élimination de l'excès. On imagine que les paquets sont des gouttes d'eau. L'émetteur, au lieu d'envoyer les paquets directement, doit les placer dans un seau. Ce seau est percé d'une manière qui permet à des gouttes de tomber avec une vitesse ρ, exprimée en paquets par seconde. Si la taille des paquets est limitée, on limite ainsi le débit. Il en résulte un flux régulier émis chaque 1/ρ unité de temps. Il agit comme un ralentisseur contre les flux de haut débit et aux rafales.

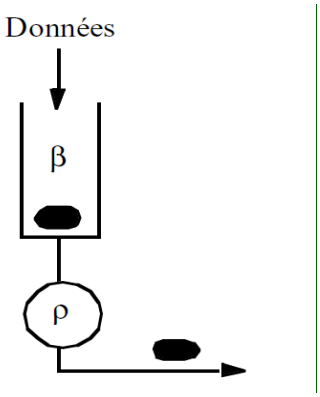


Figure 5: Régulateur de trafic, Seau percé

- **Token Bucket (seau à jeton)** :

C'est une extension du précédent, mais qui laisse les rafales tous en limitant leur durée. Ici ρ est le débit des jetons placés dans le tampon dont la capacité ne dépasse pas ***β.*** Pour transmettre un paquet le dispositif enlève un jeton. Son avantage est qu'il n'élimine pas les paquets mais il laisse la responsabilité au réseau d'en décider.

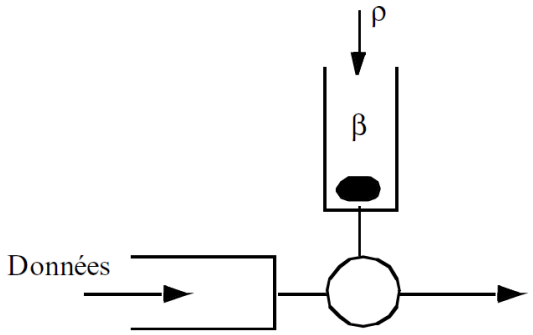


Figure 6: Régulateur de trafic, Seau à jeton

Un seau à jeton pourra distinguer deux niveaux de priorité. Pour trois niveaux on peut utiliser deux seaux à jeton en série.

6.1.3. Traffic policing (contrôle du trafic) :

C'est une vérification de la conformité du trafic à sa description. Plusieurs politiques peuvent être appliqués sur les paquets frauduleux ou non conformes à la spécification qui varient de la suppression jusqu'au traitement par la non garantie du BE (Best Effort). Pour contrôler et distinguer les flux d'une classe de service, la couleur « vert » est attribué à ceux conforme et la couleur « rouge » pour les flux en excès. Le document [RFC2697] propose trois couleurs pour classer les flux transitant en approbation avec son contrat. Dans ce document le contrat est décrit selon trois paramètres SrTCM (Single Rate Three Color Marker) :

- Le débit d'informations garantis CIR (Committed Information Rate) ;
- La taille des rafales garanties CBS (Commited Burst Size) ;
- La taille des rafales en excès EBS (Excess Burst Size) ;

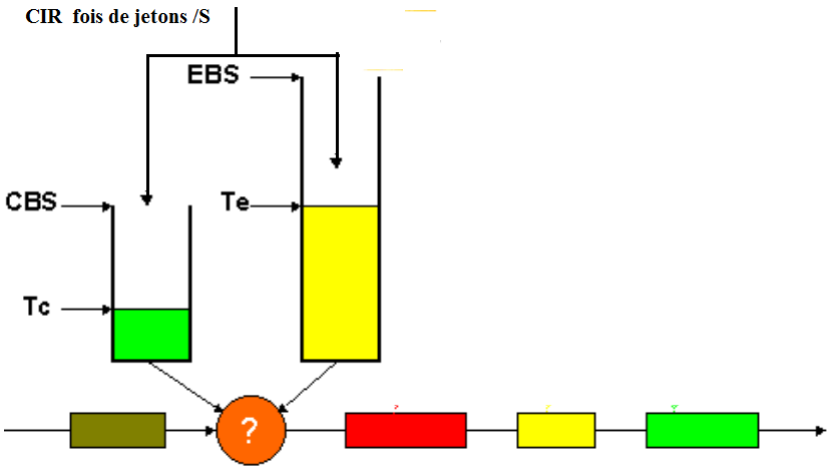


Figure 7 Le Contrôle de flux par SrTCM

Le paquet est « vert » s'il ne dépasse pas le CBS « jaune » s'il dépasse le CBS mais pas l'EBS et « rouge » sinon. Ce mécanisme permet des rafales temporaires, lorsque la ligne était sous-utilisée auparavant. Un autre algorithme est proposé dans la [RFC2698] pour le même marquage mais avec double débit TrTCM (Twoo Rate Three Color Marker) en ajoutant un paramètre du débit d'information en pic PIR (Peak Information Rate).

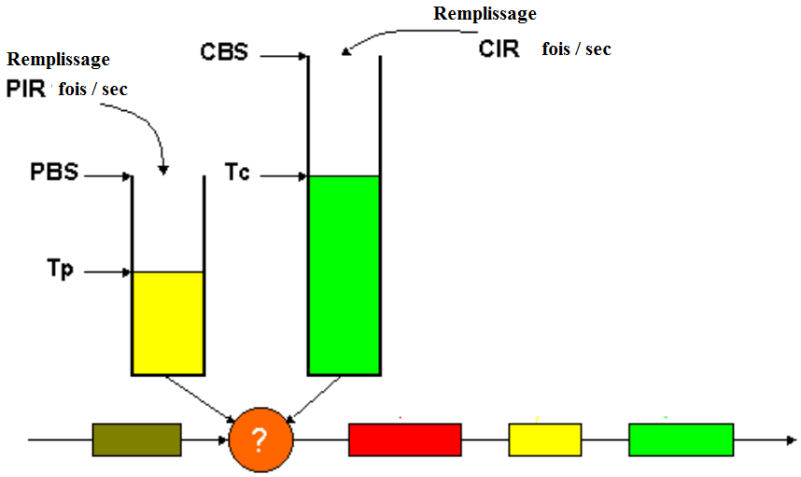


Figure 8 Contrôle de flux par TrTCM

6.1.4. Buffer management (gestion de tampon) :

Surtout appliquée dans les routeurs internes, en cas de congestion, des paquets seront éliminés en fonction des critères de la QoS appliquée. Les algorithmes de la gestion du tampon se basent sur le comportement PHB associé à la classe du trafic. Parmi les algorithmes proposés :

- **Drop-Tail** : c'est le comportement par défaut des routeurs n'utilisant pas de QoS. Il est encore utilisé pour gérer les files d'attente associées aux classes n'exigeant pas de QoS comme BE. Le principe est d'éliminer tous paquet arrivant après la congestion.

- **RED** : Proposé dans [RED], consiste à supprimer les paquets avant même la congestion en fonction d'une probabilité proportionnelle avec la BP utilisée, et cela après un seuil minimum de la taille du buffer. Au-dessus d'un deuxième seuil, les paquets seront tous éliminés. Toutefois, il n'est pas adopté pour la QoS, dans la mesure où plusieurs niveaux de trafic coexistent dans le tampon, mais utilisé plutôt pour éviter la congestion dans TCP qui interagit avec les indicateurs de congestion, et non UDP ou ICMP.
- **WRED** : Proposé par CISCO [WRED], il est basé aussi sur l'estimation de l'occupation de la file d'attente en supprimant les paquets selon leurs probabilités. En revanche, la priorité des classes de service est prise en considération. Il utilise trois seuils d'élimination pour distinguer ainsi trois niveaux de priorités.

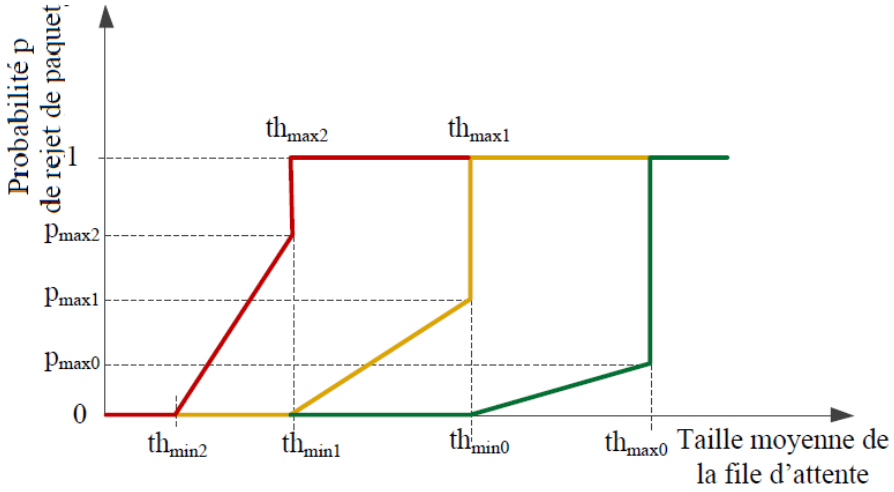


Figure 9 Rejet de paquets par l'algorithme WRED

- **ECN :** La gestion de la mémoire tampon ne se limite pas seulement à l'élimination des paquets. En effet, une perte d'un paquet n'indique pas nécessairement qu'une congestion s'est produite. Pour ce but l'IETF a adopté le protocole ECN dans [RFC3168] qui spécifier un mécanisme de notification de la congestion pour informer les participants d'une probable congestion, au lieu d'éliminer les paquets. Il utilise le mécanisme AQM (Active Queue Management) avec une version modifiée de l'algorithme RED pour signaler au récepteur que les files commencent à être remplies par une marque dans les paquets. Le récepteur informe explicitement la source de ce phénomène pour qu'elle anticipe la congestion en réduisant son débit.

6.1.5. Traffic Scheduling (ordonnancement du trafic) :

C'est le module le plus critique dans les réseaux QoS et en particulier dans l'architecture DiffServ. Même en l'absence de congestion, les paquets subissent des délais d'attente plus ou moins importants que les flux temps réel ne le tolèrent. Le choix de la politique d'ordonnancement et décisif pour garantir la QoS attendue. Les algorithmes qu'on détaillera dans les chapitres qui suivent peuvent être vus de deux façons, une vision IntServ qui manipule chaque flux séparément et une vision DiffServ qui traite chaque classe de service comme un flux unique. Principalement tous les algorithmes découlent de trois types d'ordonnancements :

- A priorité fixe.
- Basé sur le paradigme du temps partagé (GPS)
- Basé sur la trame temporelle.

6.2. Les comportements PHB :

Avant d'aborder la notion de PHB (Per Hop Behavior), examinons les éléments fonctionnels de l'architecture DiffServ. Les éléments de bordure (BR) font la classification des paquets et le conditionnement du trafic cités au-dessus. Les éléments du cœur (CR) consacrent toutes leurs ressources à la commutation, la gestion, l'ordonnancement et l'envoi des paquets. Ces traitements sont établis en se basant sur un certain comportement précis, associé à l'identifiant DSCP de la classe du trafic déjà marqué par les éléments de bordure (BR). Donc un PHB résume le traitement adéquat associé à une classe de service que les CRs doivent exécuter.

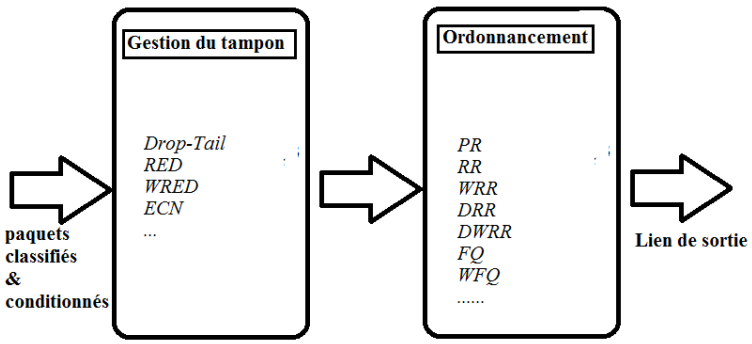


Figure 10 Architecture d'un routeur de cœur CR

sLe modèle DiffServ ne fait qu'exploiter six bits de l'octet désignant le champ ToS de l'entête IP et le baptise DSCP. Cependant, il donne plus de granularité et plus de précision en affectant à chaque classe un comportement spécifique et à l'intérieur de chaque une d'elle des niveaux de perte probabilistes. Le tableau qui suit montre la relation entre les 3 bits de priorité (PRECEDENCE) du champ ToS et leur correspondance dans DSCP :

5	**101**	*Expedited Forwarding (EF)*
4	**100**	*Classe 4 (AF)*
3	**011**	*Classe 3 (AF)*
2	**010**	*Classe 2 (AF)*
1	**001**	*Classe 1 (AF)*
0	**000**	*Best effort (BE)*

Tableau 01 : les champs de ToS dans DSCP

DiffServ distingue trois comportements :

6.2.1. PHB EF (Expedited Forwarding) :

Ce comportement est destiné à traiter le service premium, dont le trafic ne supporte en aucun cas le retard ou la perte des paquets. Deux possibilités sont proposées pour réaliser ce PHB :

1- L'utilisation d'une file d'attente dédiée pour stocker les paquets EF qui vont être servis en premier dès l'arrivée d'un paquet, à condition que cette file ne soit pas remplie.

2- Sur-dimensionner une file d'attente qui sera servie la plus part du temps.

Il faut toutefois restreindre le nombre de paquets de ce type de classe dans le réseau sinon cela augmente le temps d'attente dans la file (délai). Cette tâche est affectée au contrôle d'admission, vu au-dessus, qui décide si un paquet EF peut être accepté dans le réseau sans nuire aux autres paquets EF y transitant.

6.2.2. PHB AF (Assured Forwarding) :

Ce comportement offre une vaste différentiation de service. Il est meilleur que BE et plus faible que EF mais pouvant comporter des délais et des pertes. DiffServ propose quatre catégories AF différentes garantissant ainsi quatre services de QoS. Chaque classe garantis une certaine BP avec la possibilité d'utiliser d'autres BP supplémentaire si elle existe. Chaque catégorie comporte trois niveaux de priorité, en cas de congestion, les paquets des niveaux les plus bas (de haute probabilité) sont éliminés, puis ceux de moyenne priorité.

La détermination du niveau de priorité d'un paquet dans une classe est faite par les mécanismes, décrits au-dessus, pour décider si les paquets d'un flux sont conformes au contrat. Chaque nœud dans le réseau est en mesure de :

- Réduire les longues congestions dans la classe.
- Accepter relativement les rafales à court terme.
- Traiter de la même façon les micros flots et ne pas favoriser les uns sur les autres.

Pour accomplir ces tâches les algorithmes vus au-dessus, comme RED, WRED, ECN … sont mis à l'œuvre dans les routeurs

Probabilité de suppression	*Classe 1*	*Classe 2*	*Classe 3*	*Classe 4*

Basse	*AF11* ***10(001010)***	*AF21* ***18(010010)***	*AF31* ***26(011010)***	*AF41* ***34(100010)***
Moyenne	*AF12* ***12(001100)***	*AF22* ***20(010100)***	*AF32* ***26(011100)***	*AF42* ***36(100100)***
haute	*AF13* ***14(001110)***	*AF23* ***22(010110)***	*AF33* ***30(011110)***	*AF43* ***38(100110)***

Tableau 02 : Les probabilités de perte dans les classes AF

6.2.3. PHB BE (Best Effort) :

Ce comportement a la plus faible priorité, c'est celui du service implémenté nativement dans internet. Il ne distingue pas entre les flots prioritaires de ceux non prioritaires. Autrement dit, il n'offre aucune garantie de service.

6.3. Limitations :

Le service différentié a bien remédié aux problèmes de passage à l'échelle rencontrés dans les services intégrés, mais la différentiation absolue de quelques classes, en les privilégiant par rapport aux classes concurrentes (pour le partage de ressources), cause un problème de famine entre eux. Un autre inconvénient est la nécessité d'une bonne estimation des demandes à venir et un surdimensionnement des ressources. Il faut ensuite paramétrer les équipements du réseau pour mettre en œuvre les garanties liées aux classes de services. La complexité du dimensionnement vient de la difficulté de traduire des propriétés de QoS de bout en bout en paramétrage d'équipements individuels. La configuration de chaque équipement sur un chemin entre une source et une destination doit être cohérente avec les classes mise en œuvre sur ce chemin. Cela devient encore plus difficile si les équipements n'offrent pas tous les mêmes caractéristiques ou les mêmes possibilités matérielles.

Il est peut être intéressant de recopier de [MAG1] une comparaison entre les deux architectures IntServ et DiffServ pour bien éclaircir les enjeux et les objectifs d'un réseau de qualité de service.

Propriété	IntServ	DiffServ
Garantie en QoS	Par flux	Par agrégation
Zone de QoS	De bout en bout	Dans les bordures BRs (domaines DiffServ)
Réservation de ressource	Contrôlé par les applications	Configuré dans les BRs selon des SLA
Gestion des ressources	Distribuée	Centralisé dans un domaine DiffServ
Signalisation	Protocole dédié (RSVP)	Par le champ DSCP dans l'entête IP
Passage à l'échelle	Non recommandé	Dans tous les réseaux
Classes de service	GS, CL, BE	EF, AF, BE

Tableau 03 : Comparaison entre IntServ et DiffServ

7. MPLS

Standardisé par l'IETF dans les années 90 ; MPLS (Multi Protocol Label Switching) a été conçu pour simplifier le routage dans les réseaux IP en évitant la consultation de grandes tables de routage afin d'augmenter la capacité d'acheminement des routeurs. Après la convergence des WAN vers IP et la complexité de l'ingénierie de trafic, ce protocole devient de plus en plus intéressant et peut résoudre d'autres problèmes.

Son principe est d'acheminer les paquets en empruntant des chemins prédéfinis en fonction d'une étiquette (Label), non pas par les entêtes IP. Il dérive donc du mode virtuel comme ATM, néanmoins des paquets sont commutés et non pas des cellules. A l'équivalent de l'ATM qui se base sur l'étiquette VPI/VCI pour commuter, MPLS se base sur des labels dans l'entête IP, ajoutés tôt, à l'entrée d'un domaine MPLS. Donc tout au long du circuit, seul ce label sera examiné et toutes les décisions d'acheminement seront prises uniquement en fonction de celui-ci (Label Switching). Chaque label est alloué à ce qu'on appelle un FEC (Forwarding Equivalence Class) qui représente un ensemble de paquets qui doivent subir le même traitement.

MPLS est particulièrement adapté pour le modèle DiffServ, où on trouve la notion de routeurs de bordure et ceux du cœur ainsi que la notion de groupement de flot.

7.1. Positionnement :

Le but de MPLS est de simplifier et alléger le traitement dans le cœur du réseau en concentrant le traitement dans les bordures. Le M et le P dans MPLS signifient Multi Protocol car il met en œuvre des fonctions d'acheminement de paquets qui sont de la responsabilité des protocoles réseau mais aussi de la commutation qui est du ressort de la couche 2. Par conséquent c'est un protocole se situant entre la couche 2 et la couche 3 du modèle OSI. Il est qualifié parfois de protocole de couche 2.5.

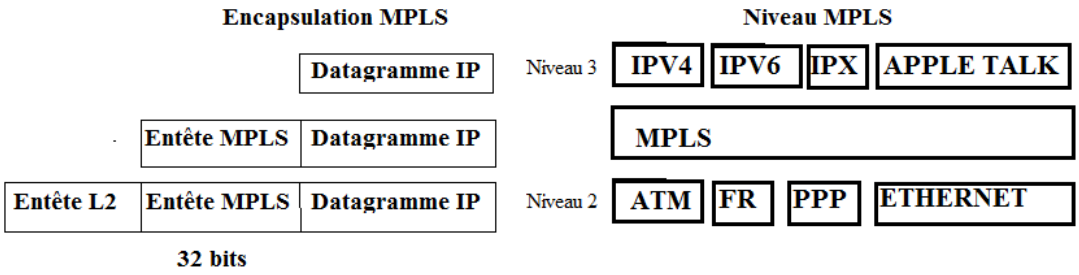


Figure 11 Niveau MPLS dans la pile OSI

Le L est S dans MPLS signifient Label Switching, car l'acheminement des paquets se fait par commutation selon un label transporté dans chaque paquet. Un label est un entier associé à une route, définissant un label et une interface de sortie, dans une table de commutation.

7.2. Architecture :

Les routeurs de bordure du réseau, appelés LER (Label Edge Router) font l'interface avec le monde IP à l'extérieur du réseau, car ils vont devoir ajouter un label aux paquets entrants pour qu'ils puissent être acheminés dans le réseau MPLS et retirer le label aux paquets à la sortie vers le monde extérieur IP.

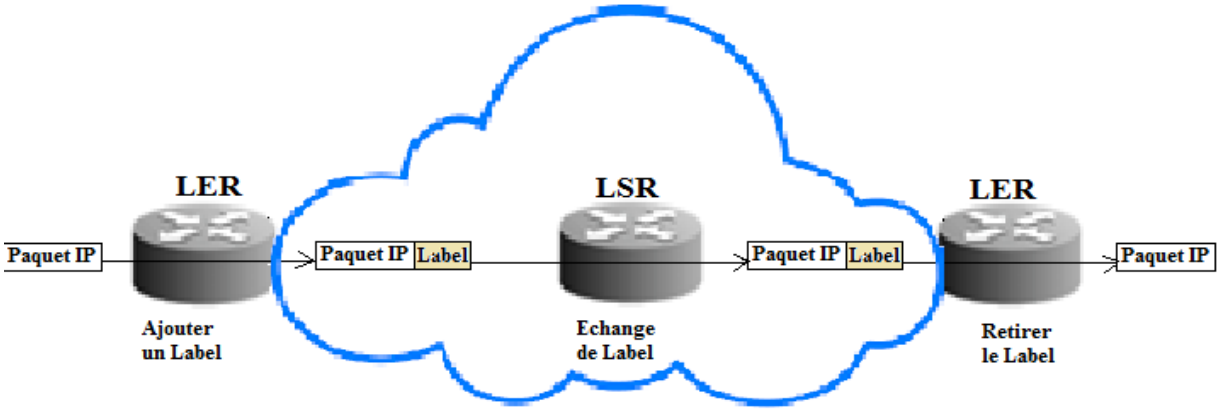


Figure 12: Architecture MPLS

Le réseau MPLS va créer des LSP (Label Switching Path), des chemins à commutation de label, ces chemins définissent une route de bout en bout par la concaténation de labels. A l'intérieur du réseau, des routeurs appelés LSR (Label Switch Router) font la commutation de labels et achemineront les paquets de proche en proche sans tenir compte des adresses IP.

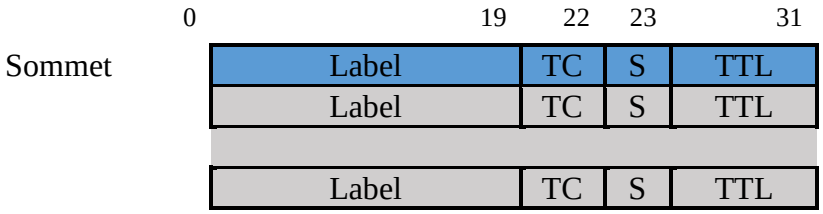
7.3. Classe d'équivalence FEC (Forwarding Equivalence Class)

La notion de FEC est identique à la notion de classe d'équivalence mathématique qui représente tous les éléments d'un ensemble similaires selon une propriété donnée, où les éléments représentent les flux. Les propriétés selon lesquelles les flux sont regroupés en FEC varient énormément avec la possibilité de combiner entre eux, en prenant en compte que tous les flux appartenant à une FEC subissent le même traitement au sein des nœuds internes du réseau MPLS. La détermination des FECs fait appel au savoir-faire de l'administrateur du réseau. Ces FECs faciliteront énormément l'ingénierie de trafic dans sa phase de dimensionnement basant sur des propriétés comme :

- La similarité entre l'entrée et la sortie d'un flux.
- Les flux qui doivent sortir d'un nœud particulier.
- Les flux ayant besoin d'une bande passante particulière.
- Les flux représentant un port d'entrée ou de sortie.
- …

7.4. Entête MPLS :

L'entête MPLS est positionné entre la couche 2 et la couche 3. Il est constitué d'une pile de labels, chaque entrée de la pile est de 32 bits [RFC3031]. L'empilement de labels permet en particulier d'associer plusieurs contrats de service pour un paquet ou bien l'appartenance à un VPN. Une entrée dans la pile MPLS contient :



Label : Valeur de label sur 20 bits.
TC :Traffic Class Field.
S : sur 1bit = 1 si l'entrée est le fond de la pile =0 sinon.
TTL: Sur 8 bits Time to Live.

Figure 13Les champs de l'entête MPLS

- Une valeur de label, représentée par un entier sur 20 bits, servant à identifier le LSP
- Un champ TC sur 3 bits, expérimental contenant l'identifiant d'un PHB correspondant à une classe de service, pour la compatibilité avec DiffServ
- Un champ S sur 1 bit, indiquant si l'entrée est aussi le fond de la pile
- Un champ TTL sur 8 bits, cloné à partir de l'entête IP admis à l'entrée du réseau, pour la détection des boucles

La portée des labels est locale entre deux nœuds, il doit être changé à chaque saut. Lors de l'acheminement d'un paquet seule l'entrée au sommet de la pile est considérée par les routeurs LSR. Ils peuvent ensuite :

- Remplacer la valeur du label au sommet de la pile (*Swap*) avant retransmission
- Dépiler une entrée laissant remontrer l'entrée suivant à l'entête de la pile (*Pop*), si la pile ne contenait qu'une entrée, alors le LSR supprime l'entête MPLS, et le paquet sera traité par un protocole de niveau 3
- Empiler (*Push*), pour insérer la pile de label

7.5. Chemins MPLS

Chaque LSR gère une table de commutation. Pour chaque FEC définie par l'administrateur, il va créer deux associations :

- la première est entre la FEC et le prochain saut. Si l'administrateur décide du routage explicitement, il devra paramétrer cette association. Sinon, MPLS utilise le routage pour

choisir le prochain saut, grâce aux informations stockées dans la table de routage. Ces informations ont été constituées par les protocoles de routage tels qu'OSPF, ISIS, BGP.

- Pour la seconde association, il crée un label et le lie à la FEC. Il va devoir ensuite distribuer le label à son voisin.

Le label a une portée locale ce qui implique que deux LSR voisins doivent décider de la même valeur pour la FEC. C'est le nœud en aval qui décide de la valeur du label et en informe son voisin en amont, quand tous les LSR ont effectué cette opération, le chemin dit LSP sera créé.

7.5.1. Association entre FECs et labels

Les associations sont créées dynamiquement à l'arrivée d'un flux ou bien à statiquement à l'avance.

Dans la première méthode, le LSR attend qu'un flux n'ayant pas encore de label associé à sa FEC se présente pour déclencher le mécanisme d'association. Cela a pour avantage de ne générer aucune association inutile et donc de ne pas utiliser plus de labels que nécessaires. Mais cela implique un temps de traitement avant tout acheminement du paquet dans le réseau MPLS.

Dans la deuxième méthode, l'association est établie dès que possible, avant toute utilisation. Les évènements qui peuvent déclencher cette opération sont par exemple une demande de réservation de chemin RSVP ou un changement dans le routage OSPF. L'avantage est que lors de l'arrivée d'un paquet de la FEC, le label sera déjà connu et le paquet peut être transmis sans attendre. L'inconvénient est que la table de commutation sera bourrées d'entrées qui ne serviront peut être jamais.

7.5.2. Distribution de labels

MPLS utilise les informations fournies par les protocoles de routage tels que BGP ou OSPF afin de connaître le prochain saut pour une FEC donnée. Les labels sont distribués grâce à BGP, RSVP ou par un protocole dédié LDP (Label Distribution Protocol). Lors de l'association des labels aux FECs, l'ordre de cette procédure se fait en deux manières :

- **Mode ordonné**, où les LSRs créent les labels les uns après les autres en commençant par le nœud en fin de chemin.
- **Mode non sollicité**, où chaque LSR en aval décide du label, ce qui minimise le temps, en revanche un LSR peut envoyer des données dès qu'il reçoit le label d'un aval, alors que le label du suivant n'est pas encore attribué.

Les boucles sont inévitables et ils sont traités avec les mécanismes courants, comme le TTL.

7.5.3. Le protocole LDP

La [RFC5036] définit 4 types de messages pour le protocole LDP (Label Distribution Protocol). Il utilisera tout d'abord des messages de découverte des voisins grâce à l'échange de messages Hello via UDP. Puis deux LSR voisins échangeront des messages d'établissement de session via TCP. Enfin ils utiliseront des messages d'annonce pour signaler les labels et les associations avec les FECs.

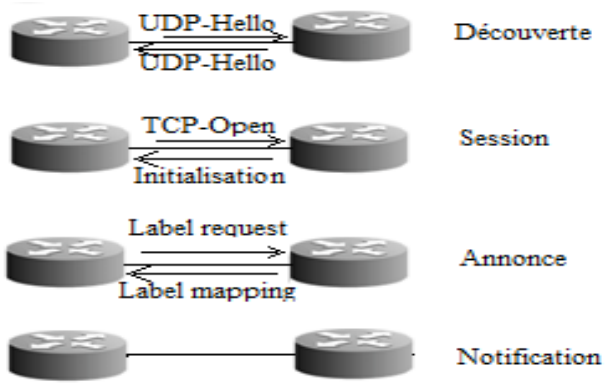


Figure 14 Protocole de Distribution de label LDP

La gestion des erreurs du protocole est assurée grâce à des messages de notifications.

7.6. Limitations :

On peut distribuer les labels grâce à BGP, RSVP ou LDP. Cependant, contrairement à RSVP, BGP et LDP ne peuvent gérer que des flux Best Effort. Pour améliorer les performances d'un réseau en qualité de service, la simple version de MPLS tel qu'elle est insuffisante. Plutôt une autre version adaptée à la prise en compte de métriques plus évoluées, par exemple des métriques de qualité de service. C'est MPLS-TE : MPLS pour l'ingénierie de trafic. En effet, parmi les objectifs de la technologie MPLS est d'augmenter la vitesse de traitement des paquets. Mais avec l'apparition des giga routeurs, la performance est passée au second plan et c'est plutôt l'aspect fonctionnalité qui motive son développement est sa standardisation :

- L'intégration IP/ATM,
- La création de VPN,
- L'ingénierie de trafic permettant de définir des chemins de routage explicites dans les réseaux IP,
- La flexibilité : possibilité d'utiliser plusieurs types de media (ATM, FR, Ethernet, PPP),
- Les Services différenciés (DiffServ),
- Le routage multicast,

8. Ingénierie de trafic& ses protocoles (Routage basé sur QoS) :

La qualité de service est définie par des garanties de performances, ce qui implique que tous les équipements soient paramétrés individuellement mais en cohérence. Ceci est déjà compliqué à l'intérieur d'un réseau et doit être vrai également entre tous les systèmes autonomes pour garantir les performances de bout en bout. Une alternative est l'ingénierie de trafic. Cette méthode ne peut être utilisée que par l'administrateur de réseau qui va optimiser l'utilisation de son réseau pour offrir des garanties au trafic. C'est donc une solution orientée opérateurs.

8.1. Définition

Il n'existe pas de définition formelle mais on peut définir l'ingénierie de trafic comme un moyen d'allouer efficacement les ressources du réseau en fonction de contraintes du trafic de l'utilisateur ou du réseau tout en maximisant l'objectif de l'administrateur. Si on se réfère à la définition donnée à l'IETF dans [RFC3272], il s'agit d'optimiser les performances de réseaux IP opérationnels.

Les modèles précédents IntServ et DiffServ ne peuvent pas éliminer la congestion, ils ne font que favoriser des flux sur d'autres en cas de congestion. Avec l'ingénierie de trafic on va définir le routage. Les flux ne seront plus systématiquement envoyés sur les plus courts chemins mais sur des chemins choisis par l'administrateur.

8.2. Les offres :

L'ingénierie de trafic a pour but d'améliorer le fonctionnement du réseau selon un objectif défini par l'administrateur. Plusieurs possibilités peuvent être envisagées, les exemples qui suivent illustrent quelques-unes offertes par ce modèle :

- Prévoir des chemins de secours pour réagir en cas de pannes.
- Répartir les charges sur plusieurs chemins pour équilibrage.
- Différentier les chemins suivis par des données de nature différente.
- Réduire le coût opérationnel en diminuant la consommation d'énergie des équipements en période de faible utilisation, et changer le routage.

8.3. Cycle de l'ingénierie de trafic :

Ce modèle s'inscrit dans le cycle de maintenance du réseau. La première phase est le dimensionnement et le calcul de la situation optimale qui va être appliquée. Dans la deuxième phase, le choix et l'application de la solution choisie en configurant les équipements y participants. Il faut ensuite mesurer et analyser et voir si la solution satisfait les besoins, c'est le monitorage. La

quatrième phase consiste à planifier des actions correctives, affiner le dimensionnement ou bien optimiser une autre fois en cas d'évolution du réseau.

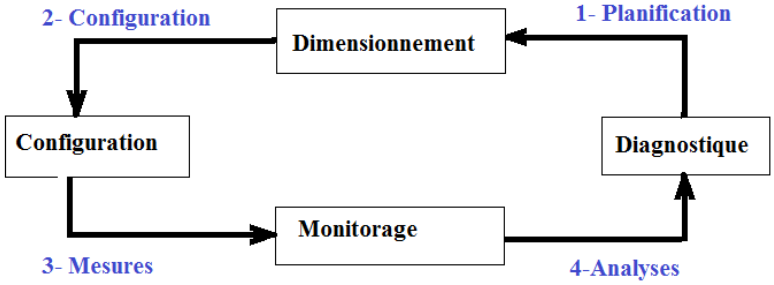


Figure 15 Cycle de maintenance dans TE

8.4. Routage contraint CBR (Constraint Based Routing) :

L'idée de l'ingénierie de trafic nous pousse à repenser l'existant du réseau et à envisager d'autres mécanismes, à part ceux cités au-dessus, qui pourront participer à la garantie de la qualité de service. Un routage comme BGP met en œuvre un algorithme CBR qui minimise le nombre des systèmes autonomes traversés en prenant en compte des contraintes administratives. Pourquoi ne pas faire des contraintes sur les délais, la perte ou sur la bande passante, on fera alors du routage basé sur la qualité de service.

8.5. Protocoles d'ingénierie de trafic :

La différence entre un protocole d'ingénierie de trafic et un protocole classique vient des métriques qu'il permet de manipuler. Les protocoles de routage utilisent une métrique plutôt stable telle que le coût ou la distance pour choisir les chemins. Un protocole d'ingénierie de trafic permet de prendre en compte des métriques beaucoup plus complexes et dynamiques. Par exemple des métriques de qualité de service telles que la bande passante disponible, les délais, les pertes, ou les coûts. Le protocole peut également considérer des métriques de politique, des propriétés sur les liens, s'ils partagent ou non un risque commun de panne.

Le protocole d'ingénierie de trafic permet de diffuser les valeurs de ces métriques dans le réseau et maintient une base de donnée stockant ces valeurs : la TED (Traffic Engineering Database).Pour pouvoir maintenir la TED, les protocoles de routage ont été dérivés dans une version d'ingénierie de trafic. Ainsi les protocoles OSPF et ISIS ont été étendu avec OSPF-TE et ISIS-TE. Pour cela, de nouveaux types d'annonces de liens permettant de transporter les métriques d'ingénierie de trafic.

8.6. Limitations :

Pour mettre en œuvre l'ingénierie de trafic, il faut connaître la topologie du réseau, c'est à dire l'ensemble des nœuds (les équipements) et des liens (les liaisons) du réseau, la connectivité entre ces équipements et la capacité ou les mesures de performance des liens. Mais aussi le routage mis en œuvre actuellement. Cela se traduit par les règles utilisées pour calculer les chemins. Enfin, Il faut avoir une bonne vision du trafic actuellement géré par le réseau. Le nombre de paramètres à configurer se croit avec le nombre de flux admis dans le réseau et avec les objectifs désirés.

9. Conclusion :

Dans ce chapitre on a présenté le concept général de la notion de qualité de service et l'évolution des modèles et architectures proposés par les standards mondiales depuis la fin des années 80. Cela à commencer par RTP/RTCP que l'on pas mentionné ici car il n'assure pas une Qos mais plutôt destiné à mesurer les performances réseau, ensuite les solutions se succèdent commençant par les services intégrés et RSVP, les services différenciés, MPLS et enfin l'ingénierie de trafic. Toute nouvelle architecture est venue couvrir les lacunes de sa précédente solution ou bien l'améliorer.

On a vu comment IntServ réserve des chemins pour favoriser des flux. DiffServ qui agrège des flux dans des classes codées sur DSCP et traitées selon des PHB. MPLS inspirant d'ATM la notion de commutation pour alléger le traitement des routeurs de cœur. Enfin l'ingénierie de trafic et les protocoles TE qui ont ajouté de la granularité sur les mécanismes de traitement de la QoS.

Les besoins attendus de la part des différentes architectures sont relativement atteints, favorisant quelques métriques et paramètres sur d'autres. Donc, de grandes perspectives d'études dans l'avenir pour améliorer les performances et apporter des solutions plus générales et complètes.

Ayant détaillé les mécanismes développés et utilisés au sein des réseaux, l'ordonnancement y toujours mentionné et implémenté dans les nœuds du réseau et influe fortement dans les solutions proposées. Pour cela le prochain chapitre est dédié aux algorithmes d'ordonnancement les plus célèbres et performants.

Chapitre 02 : Etat de l'art

1. Introduction :

La notion d'ordonnancement n'est pas réservée au domaine du réseau, mais elle est liée aux phénomènes concurrentiels où les ressources ne suffisent pas et doivent être négociées. Dans le monde du réseau, une panoplie de modèles et d'algorithmes ont été développés dans le but de trouver la meilleur méthode pour garantir une bonne qualité de service et partager proportionnellement les ressources en Bande passante, temps et espace mémoire. Cependant, vu le nombre croissant des paramètres intervenant dans la QoS, la tendance est menée vers un bon algorithme qui satisfait le maximum de ces paramètres.

Les mécanismes de gestion du tampon, eux seules, ne permettent pas de prévoir et garantir les délais attendus d'un réseau à QoS. Même en absence de congestion les paquets doivent relativement attendre leurs tours d'acheminement vers la sortie d'un routeur dont seul l'ordonnanceur a la décision. Donc la fonction d'ordonnancement est critique et le choix d'un des algorithmes est primordial pour les besoins de la politique de qualité de service attendue.

La qualité de service est basée sur deux paradigmes, la réservation de ressource comme dans IntServ ou le dimensionnement dans DiffServ. La granularité dans les protocoles d'ingénierie de trafic ou MPLS laisse l'administrateur le choix entre les deux. Ainsi la tâche d'ordonnancement dans les réseaux QoS doit traiter les paquets, ou bien comme des flux séparés selon le premier paradigme, ou bien comme des classes à servir selon le deuxième. Par conséquent, ***l'isolation des flots*** est l'un des principales caractéristiques d'un ordonnanceur dans un réseau à QoS.

Les architectures de QoS vues dans le premier chapitre cherchent toujours à alléger la lourdeur des traitements effectués dans les routeurs. L'ordonnanceur doit aussi se caractérisé d'***un faible temps de traitement*** et atteindre les objectifs attendus d'une bonne QoS.

Le passage à l'échelle est aussi une propriété importante lorsque les flux deviennent très nombreux induisant des demandes de QoS variées, et par conséquent beaucoup de files d'attente à gérer. Il est nécessaire donc d'avoir ***une implémentation facile et une complexité acceptable*** d'un algorithme d'ordonnancement.

En résumé un bon algorithme d'ordonnancement doit avoir les propriétés suivantes :

- Isolation des flux afin de les protéger contre ceux ne respectant pas leur contrat ;
- Equité en proportion avec les besoins en QoS ;
- Passage à l'échelle pour supporter un grand nombre de flux et des débits importants ;

2. Ordonnancement :

La fonction de l'ordonnancement dans un réseau est de déterminer l'ordre de servir un paquet afin de l'acheminer vers la sortie, en respectant ses contraintes temporelles.

On peut diviser les algorithmes d'ordonnancement en deux parties, à file unique ou à plusieurs files. Mais, pour les distinguer selon leur mode opératoire et leur principe de fonctionnement, on accepte l'idée qu'ils découlent de trois principales catégories [DGA] :

a. Ordonnancement À priorité fixe ;

Un paquet ne peut pas être transmis tant que ceux ayant une priorité plus élevée sont en attente. Il est mieux adapté pour les flux temps réel néanmoins ce type souffre de la domination des flux prioritaires s'ils ne sont pas limités ou bien régulés et ainsi défavorise les flux moins prioritaires. La discipline SPQ est inscrite dans cette catégorie.

b. Ordonnancement basé sur le paradigme GPS (Generalized Processor Sharing) ;

C'est un mécanisme très idéal proposé dans [Par92] et [Par93]. Son principe de base et de servir les files à tour de rôle par la plus fine unité de données. Les données sont ainsi traitées au niveau des bits !!!, théoriquement en mode continu comme un fluide, par conséquent, cette version n'est pas applicable mais illustre bien la notion de partage équitable. D'autres versions en mode discret plus améliorées sont envisagées, qui traitent les flux paquet par paquet et requièrent l'utilisation d'un temps virtuel qui consiste à calculer le temps de traitement du dernier bit d'un paquet (comme s'il était traité par GPS) pour déterminer son ordre dans la sortie. On trouve dans cette catégorie WFQ, WF2Q, SCFQ, SFQ, etc...

c. Ordonnancement basé sur la notion de trame temporelle ;

Le principe de cette catégorie est de servir les files d'attente non vides à tour de rôle selon des trames, proportionnelles au temps, assignées aux files. Ce type est récemment proposé et plus facile à implémenter comme DRR, DWRR, NDRR, etc....

3. Ordonnancement équitable :

Plusieurs définitions ont été élaborées dans le contexte réseau [Nag87], [Dem89], [Dem90]. La notion d'équité d'un ordonnanceur est liée aux ressources disponibles dans un nœud du réseau et aux besoins des classes de service en QoS des flux y circulant. C'est donc l'allocation des ressources disponibles proportionnellement avec les poids de chaque flux, quel que soit le temps d'observation.

D'une manière plus formelle, on définit les paramètres suivants :

- r_i est le poids du flux i ;
- $W_i(t_1, t_2)$ est le volume du flux i servi entre t_1 et t_2 ;

L'ordonnancement est dit équitable si : $\forall(i,j)$; $\forall(t_1,t_2), \frac{w_i(t_1,t_2)}{r_i} = \frac{w_j(t_1,t_2)}{r_j}$

4. Algorithmes d'ordonnancement :

Dans cette section, les algorithmes les plus célèbres et utilisés sont détaillés avec leurs améliorations et apports dans les réseaux à qualité de service.

4.1. FIFO (First In First Out):

La discipline du premier arrivé premier servi (PAPS) est la plus célèbre est répandue dans tous les domaines. Elle est encore utilisée [YJ02], surtout pour l'ordonnancement au sein des files d'attente d'une même classe de service. Le traitement des paquets se fait sans discrimination entre leurs classes de service. Ce qui le rend plus facile à implémenter. Cependant son utilisation pour ordonnancer le trafic dans les réseaux à QoS risque d'augmenter les délais des flux prioritaires en cas de surcharge potentiel d'un trafic particulier et en diminue ainsi le débit. Dans l'exemple qui suit, deux types de flux se présentent dans un routeur, le premier est élastique arrivant au temps t_1 l'autre non élastique ayant des contraintes temporelles strictes se présente au temps t_2 tel que $t_1 <<< t_2$. Le deuxième flux sera pénalisé dans une file FIFO malgré sa haute priorité en termes de QoS.

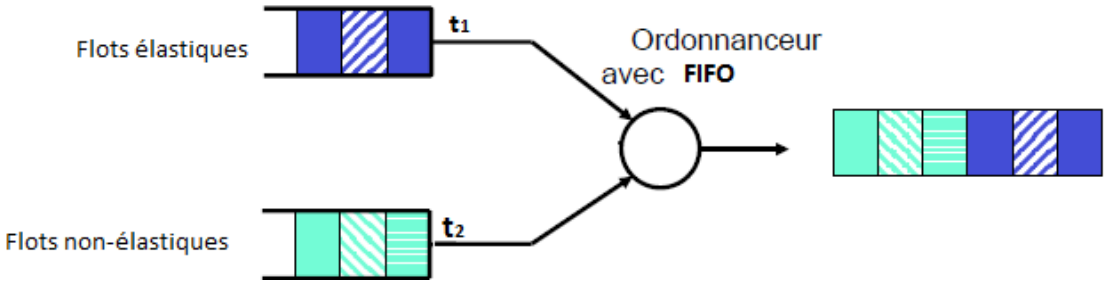


Figure 1 Ordonnancement avec FIFO

4.2. PQ (Priority Queueing) :

Il appartient au premier groupe d'ordonnancement cité au-dessus. Après la classification des paquets à l'entrée, ils sont dirigés vers leurs files d'attente appropriées selon une priorité définie. La file ayant la priorité supérieure sera servie en premier tant qu'elle n'est pas vide ensuite celle de moins de priorité et ainsi de suite.

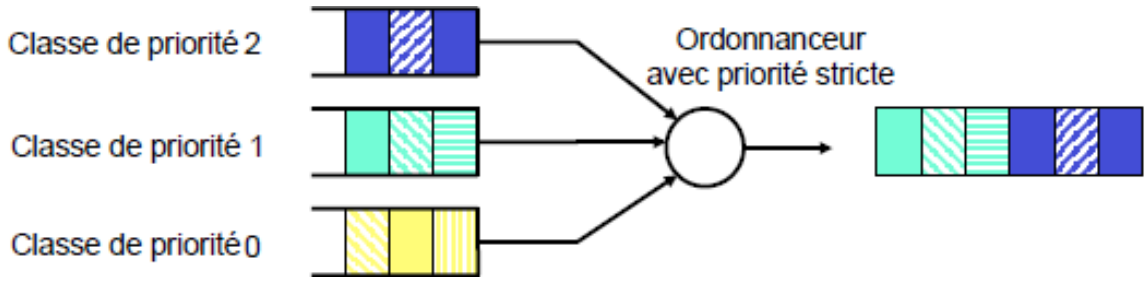


Figure 2 Ordonnancement par PQ

Ce simple mécanisme permet bien de différencier les classes de service selon leurs priorités. Son implémentation n'est pas complexe et ne nécessite que peu de ressources avec un traitement très rapide. Il garantit bien les délais pour les applications critiques. Mais dans ce type d'ordonnancement, le traitement sera consacré totalement à la file prioritaire durant le temps des pointes ou en cas de débit non attendu. Les files moins prioritaires doivent attendre pendant toute la durée de la charge et peuvent subir ainsi le phénomène de la famine.

Une solution consiste à donner un seuil limite de bande passante à ne pas dépasser pour le flux prioritaire. Ainsi les autres files auront toujours leurs taux de service quel que soit la charge du réseau.

4.3. RR (Round Robin) :

C'est une alternative pour remédier les défauts de la discipline FIFO. Les paquets sont servis à tour de rôle paquet par paquet sans se préoccuper de leurs tailles. En conséquent, cette discipline n'est équitable que par le nombre de paquets servis pour chaque file. Les flux ayant des paquets de tailles plus grandes consomment la grande partie de ressources. En revanche, ce mécanisme assure le partage loyal si les paquets des différents flux ont la même taille.

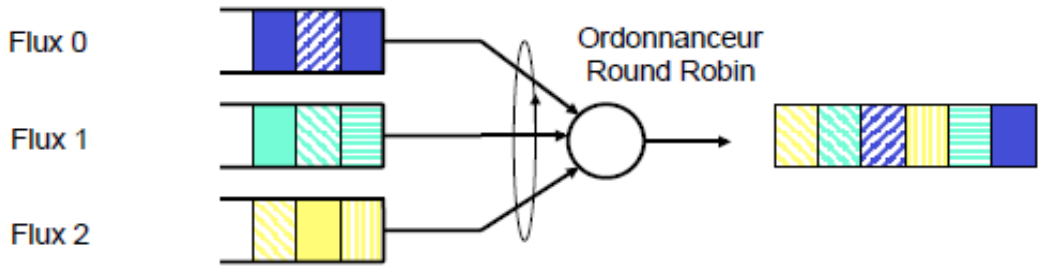


Figure 3 Ordonnancement avec RR

4.4. BR (Bit-by-Bit Round Robin):

C'est un algorithme RR parfait, indépendant de la longueur des paquets. Référencé dans [Dem89], il reste un concept théorique, impossible à implémenter. Comme son nom l'indique, il sert un bit à la fois à chaque visite. Une autre variante plus équitable est envisagée si des poids sont attribués aux files, alors il faut servir le nombre adéquat de bits selon le poids à chaque visite.

4.5. FQ (Fair Queuing) :

Proposé par Nagel en 1985 [Nag87], cette algorithme est basé sur le principe de partage équitable de ressources en bande passante (GPS), cette solution était envisagée dans le but d'éviter qu'une session consomme la majorité de la bande passante en cas de surcharge. Son mécanisme est inspiré de la version (Bit-Per-Bit Round Robin), appelée aussi BR qui sert les files bit par bit et non pas par paquet à chaque visite, de tel sorte que : si la capacité du lien de sorite est C, et N files sont actives, alors chaque session aura le même taux C/N.

La concrétisation du traitement par paquet nécessite la notion du temps virtuel qui devance la terminaison du dernier bit d'un paquet. L'ordre de service du paquet est calculé en fonction de ce temps est la taille du paquet. Pour chaque flot les calculs suivants sont élaborés :

- $A(j)$: le temps d'arrivée du j$_{ème}$ paquet ;
- $L(j)$: la longueur du j$_{ème}$ paquet ;
- $Service(j) = \max(A(j),\ Service(j-1)) + L(j)$;

Dans l'exemple suivant trois files contenant des paquets de taille 1000, 300, 1000 octets. Pour faciliter le calcul supposons que : $A(j) < Service(j-1)$ Soit $Service(0) = 0$ pour toutes les files.

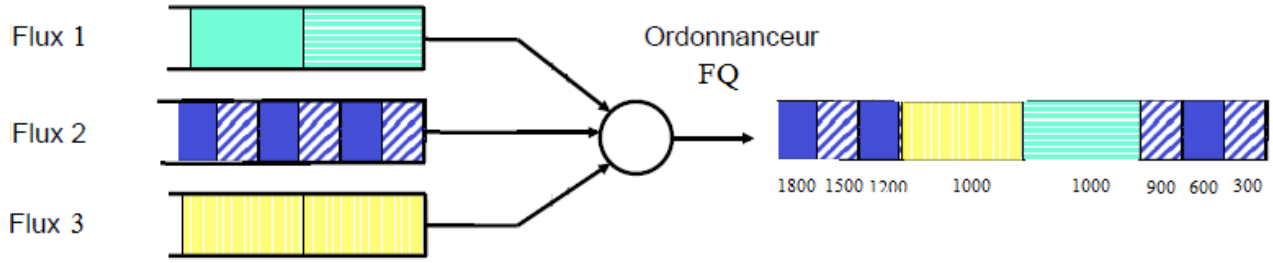


Figure 4 Ordonnancement FQ

En suivant le temps virtuel de chaque paquet l'ordre sera le suivant (Flux, Temps) : (F_2,300), (F_2,600) , (F_2,900), (F_1,1000), (F_3,1000), (F_2,1200), (F_2,1500), (F_2,1800), (F_1,2000), (F_3,2000), …

	t_1	t_2	t_3	t_3	t_4
F_1	1000	2000	…	…	…
F_2	300	600	900	1200	1500
F_3	1000	2000	3000	…	

Tableau 1 Ordre de service FQ

En pratique, l'implémentation de cet algorithme est complexe et nécessite, plus que le temps virtuel, une fonction de hachage pour la classification et la séparation des flux entrants. Il introduit alors plusieurs paramètres pour accomplir cette tâche et pour créer dynamiquement des files pour chaque classe active. Par conséquent, il surcharge le traitement dans les routeurs, en cas de grande activité dans le réseau. Ce sont les raisons pour lesquelles ce modèle ne peut pas se mettre à l'échelle, en plus cette idée ne convient pas à la notion QoS qui distingue des classes de services dans le but de répartir les ressources en proportion de leurs priorités et non pas en égalité.

C'est un paradigme qui reste théorique plus que pratique. En effet, des versions plus simples et plus pratiques sont implémentées, qui prennent en compte la fluctuation du réseau et les besoins croissants et variés de QoS.

4.6. WFQ (Weighted Fair Queuing) :

C'est une amélioration de FQ avec une attente pondérée dans les files, selon le taux de l'interface ou le besoin en QoS. Proposé par Demers, Kaskav, Shenke et Zhang en 1989 [Dem89], [Dem90], il a été conçu pour s'adapter à l'hétérogénéité des flots en termes de taille. C'est bien une version discrète de GPS en traitant des paquets en terme d'un flux de bits, la raison pour laquelle est appelé (Packetized-GPS).

On a mentionné que GPS considère les flux comme des fluides. Cependant, dans un réseau de paquets, une seule connexion est servie à la fois, et un paquet doit être transmis en entier avant qu'un autre le soit. L'idée des auteurs de WFQ est d'approximer GPS pour les réseaux de paquets en se basant sur l'estampillage qui consiste à calculer le temps de service du dernier bit d'un paquet s'il aurait été traité par GPS et c'est le temps avec lequel est défini l'ordre de fin de traitement de ce paquet dans la sortie. En revanche, ce traitement par bit, la prise en compte de la taille des paquets et le nombre variable des files (comme son précurseur FQ) le rendent un peu plus complexe.

L'estampille du $i_{ème}$ paquet du flot j est calculée avec la formule [LTO] :

$$E_j^i = \frac{1}{r_j} L_j^i + E_j^{i-1}$$

Où L_j^i est la taille du $i_{ème}$ paquet du flot j. Le calcul précédent est récurent, et donnera donc des estampilles très faibles pour des flux qui ont été inactifs pendant un moment, par conséquent ils seront accélérés au moment de leur réactivité par des estampilles faibles au détriment des flux légitimes en ressources. Pour corriger cette faiblesse, et tenir compte des autres flux, un temps virtuel *v(t)* vient s'ajouter au calcul :

$$v(t_2) - v(t_1) = \frac{1}{\sum_{r_a} r_i}(t_2 - t_1)$$

Où r_a est le poids de la classe active, et $t_2 - t_1$ est le temps pendant lequel le flot est actif. Par conséquent le calcul de l'estampille avec les nouveaux paramètres devient :

$$E_k^i = \frac{1}{r_k} L_k^i + \max(v(a_k^i), E_k^{i-1})$$

Où a_k^i est le temps d'arrivée du paquet *i* du flot *k*.

Le calcul d'estampille peut être simulé comme si on pouvait découper les paquets en morceaux de même taille, et en servant les files selon leur poids, comme l'illustre l'exemple suivant :

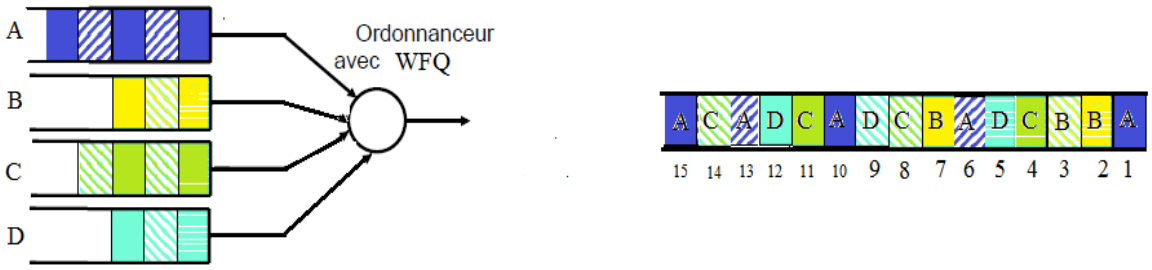


Figure 5 Ordonnancement avec WFQ

Soient quatre paquets (A, B, C, D) de taille (5, 3, 4, 3) arrivant en même temps sur des files de poids (1, 2, 1, 1). Le temps de fin de chaque paquet est (B, 7) (D, 12) (C, 14) enfin (A, 15). Par conséquent les paquets seront servis dans l'ordre **BDCA**.

Le calcul avec les formules donne le même résultat, avec estampille initiale égale à zéro :

Paquet	*Poids r*	*Taille L*	*Estampille*	*Ordre*
A	1	5	E_A = 5/1 + 0 = 5	**4**
B	2	3	E_B = 3/2 + 0 = 1.5	**1**
C	1	4	E_C = 4/1 + 0 = 4	**3**
D	1	3	E_D = 3/1 + 0 = 3	**2**

Tableau 2 Ordre de service WFQ

Notons l'existence d'autres algorithmes qui adoptent le principe GPS, basés sur la notion d'estampille, que nous ne développons pas ici, comme SFQ dans [SFQ] et SCFQ dans [SCFQ].

4.7. CBFQ (Class-Based Fair Queuing):

Cette approche est destinée aux réseaux différentiés car elle se base sur un nombre fini de classes de service. Elle consiste à attribuer un pourcentage de la bande passante aux classes différentiées selon leurs priorités et la garantir quel que soit la charge du réseau [DGA].

4.8. CBWFQ (Class-Based Weighted Fair Queuing):

C'est la combinaison des deux derniers : WFQ et CBFQ. Il introduit la notion du temps virtuel avec la différentiation de services en classes. Le pourcentage de la bande passante allouée à une classe défini le nombre de paquets à servir tandis que le temps virtuel est calculé pour prédire l'ordre des paquets dans la sortie [DGA]. L'objectif principal est de restreindre le nombre de files traitées dans WFQ à un nombre prédéfinis selon les classes envisagées par les besoins en QoS.

4.9. CBQ (Class-Based Queuing):

Basé sur le mécanisme de classification des flots comme PQ, le nombre fixe des classes permet d'attribuer un pourcentage de la BD globale à chaque file d'attente. Le fonctionnement de ce modèle est détaillé par V. Jacobson et S. Floyd dans [CBQ]. Les files sont visitées séquentiellement par le mécanisme de RR selon le poids attribué à chaque classe. Il assure ainsi qu'il n'y aura pas de famine pour les classes à basse priorité. A la sortie, un composant dit ***estimateur*** calcule le débit de chaque classe et marque les files selon le cas : moins, égale ou supérieur au débit qui lui est attribué. Un autre composant appelé ***ordonnanceur d'excès*** redistribue les ressources non consommées sur les classes ayant droit (définies par l'estimateur) [LTO] [DGA].

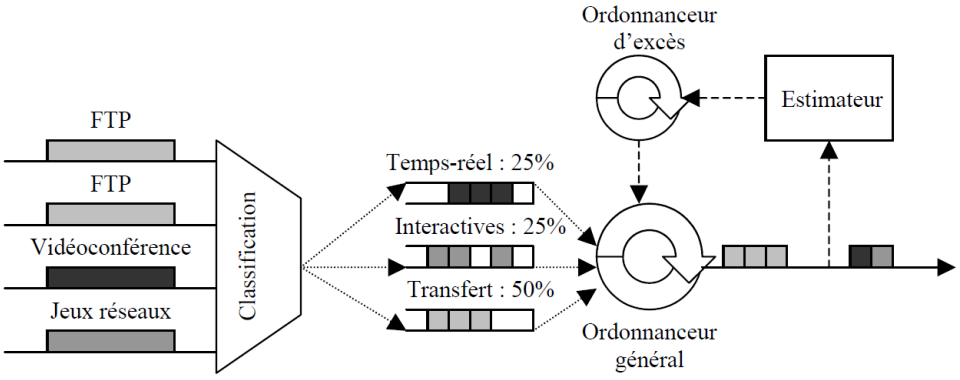


Figure 6 Ordonnancement avec CBQ

Cette algorithmes peut être implémenté directement dans les composants des routeurs en plus il garantit efficacement le partage de la BD et exploite pleinement les ressources. Néanmoins il n'est équitable que si les flux ont la même taille et ne peut pas faire face au nombre très varié des classes qui ne cessent d'augmenter. Il peut aussi conduire vers des délais d'attentes importantes pour des classes ne pouvant pas subir de retard dans leur contrat de QoS.

4.10. WRR (Weighted Round Robin):

Il vient corriger les défauts de PQ et FQ en attribuant à chaque file des poids qui sont normalisés selon la taille moyenne des paquets ou les besoins en bande passante. Les poids seront à nouveau normalisés pour obtenir des entiers correspondant au nombre de paquets servi à chaque visite de la file en mode Round Robin. Il parait au premier abord qu'il est semblable à WFQ, mais la différence réside dans les unités traitées. En effet celui-ci traite les flux en termes de paquet dans sa globalité, par contre WFQ traite les paquets en termes de bits à chaque visite.

Dans l'exemple qui suit trois files de tailles respectivement : 50, 500, 1500 Octets. Les poids au début : 0.5, 0.75, 1. La première normalisation donne : 0.5/5, 0.75/500, 1/1500. La deuxième normalisation donne : 60/6000, 9/6000, 4/6000. Enfin les files sont servis selon les poids 60, 9, 4, qui nous donne en terme de taille : 3000, 4500, 6000 octets.

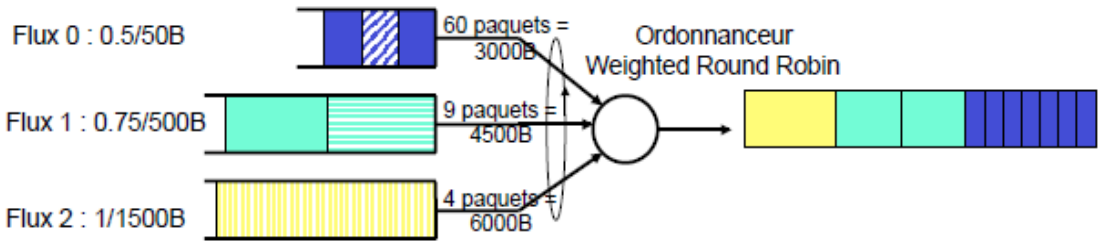


Figure 7 Ordonnancement avec WRR

L'un des difficultés de ce traitement est la connaissance de la vrai taille des paquets qui est parfois difficile pour cela il opère sur la taille moyenne estimée dans chaque file d'attente. En plus il ne corrige pas le problème de délai et ne fait pas distinction entre les flux agrégés dans une même classe. En effet, les paquets dans les différentes files n'ont pas la même taille, et l'estimation de la moyenne peut s'écarter, ce qui perturbe l'équité de partage à long terme. Néanmoins, il est très facile à implémenter et fait bien la différentiation des services.

4.11. DRR (Deficit Round Robin) vs DWRR (Deficit Weighted Round Robin):

Dans le but d'adresser les limites de WRR et garantir un partage équitable en bande passante, M. Shreedhar et G. Varghese proposèrent [DRR] l'ordonnancement par DWRR ou DRR. Ils ajoutèrent autres paramètres qu'un prorata :

- Un poids (prorata pour chaque file) indépendant de la taille des paquets.
- Un Quantum de qualité de service pour chaque file mesuré en octet proportionnel au poids.
- Un compteur de Déficit (DC) indiquant la quantité maximum en octet qui doit être servie quand la file est visitée.

Il peut servir des paquets de tailles variables sans nécessairement les connaitre. Les paquets excédant le DC associé à leur fil sont ralentis jusqu'au prochain tour. Dans le cas contraire, ils peuvent être servis en retranchant leur taille du DC qui a accumulé le dernier reste (valeur précédente) avec le quantum associé à la file. Les figures suivantes illustrent, étape par étape, le mécanisme de fonctionnement de l'algorithme DWRR

Etape 0 : Trois files pondérées de poids : 50%, 25%, 25% reflétant les quantums : 1000, 500, 500 octets. Notons que pour DRR le mécanisme est le même sauf que les poids sont égaux, ainsi que les quantums. Les valeurs des paquets sont mesurées en octets.

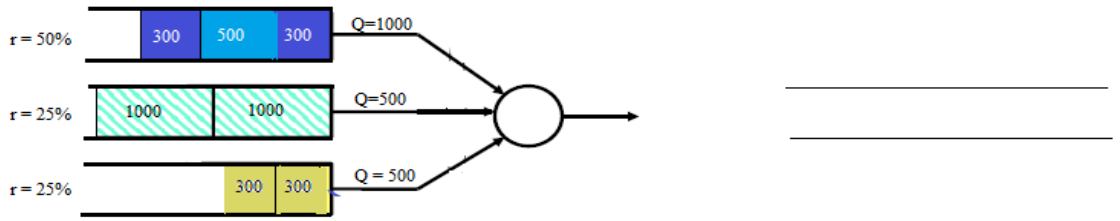


Figure 8 DWRR à l'arrivée des paquets

Etape 1 : La première file du plus grand poids est visitée en premier, dans l'entête (300+500) <1000 donc deux paquets seront servis, il reste DC =1000 – 800 = 200 dans le compteur. Dans l'entête de la deuxième file se trouve un paquet de longueur 1000 > Q=500, rien se passe et le compteur garde sa valeur DC = Q = 500. La troisième file peut être servie avec un paquet car 300 < Q et DC = 200.

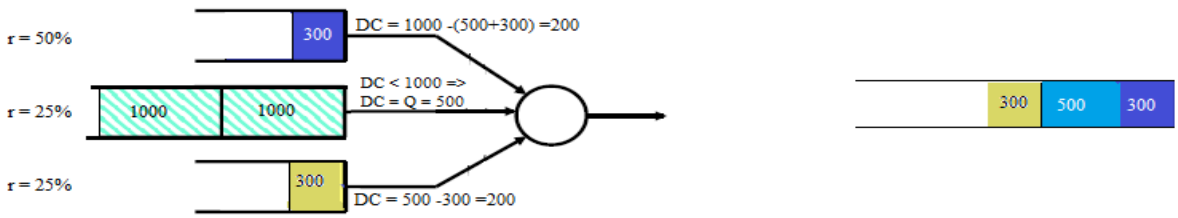


Figure 9 DWRR 1èr tour

Etape 2 : Rappelons que dans le cas où la file n'est pas vide, le compteur ajoute le quantum y associé. Donc, la première file verra un compteur DC = 200 + 1000 = 1200, et comme l'entête contient un paquet de 300 il sera servi. Le compteur de la deuxième file augmente avec 500, et aura comme valeur DC = 1000, dans ce cas un paquet de 1000 sera servi. Dans la troisième file, l'ancienne valeur 200 s'accumulera avec le quantum 500 et donnerons DC = 700, pour servir le paquet restant de 300.

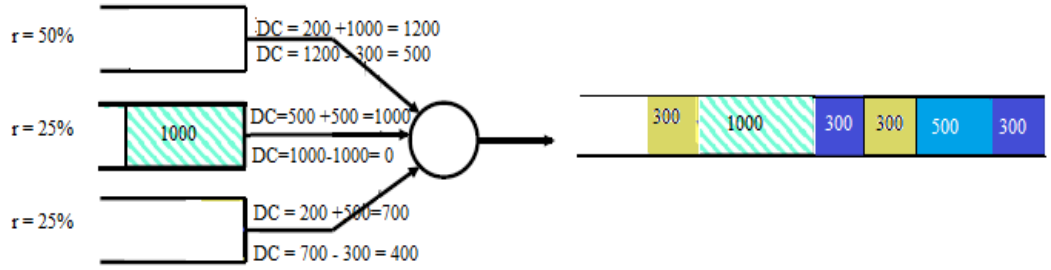


Figure 10 DWRR 2ème tour

Etape 3 : le troisième tour est très intéressant car à la rencontre d'une file vide le DC = 0 ; et son quantum sera partagé avec les autres files, donc le compteur de la deuxième file se verra une accumulation de son quantum plus le quantum de la première file DC = 500 + 500 = 1000. Il pourra ainsi servir le paquet de 1000.

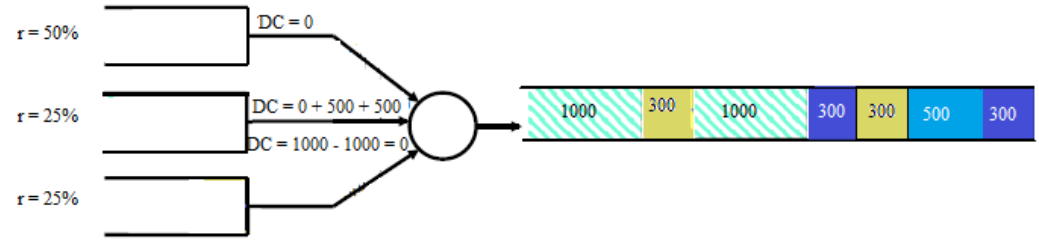


Figure 11 DWRR 3ème tour

L'algorithme suivant illustre le fonctionnement de l'ordonnancement DRR qui affecte à toutes les files le même quantum ce qui lui fait un processus relativement équitable en BD. Pour DWRR l'amélioration réside dans l'ajout d'un poids, selon les besoins en QoS pour chaque classe, entrainant ainsi une différence de quantums entre les files :

Variables et constantes

const integer N // Nb de files

const integer Q[1..N] // Quantum des files

integer DC[1..N] // Compteur de Déficit

queue queue[1..N] // Les files

Boucle d'ordonnancement

```
while (true)
  for i in 1..N
    if not queue[i].empty()
      DC[i]:= DC[i] + Q[i]
      while( not queue[i].empty() and
             DC[i] >=
queue[i].head().size() )
        DC[i]:= DC[i] -
queue[i].head().size()
        send( queue[i].head() )
        queue[i].dequeue()
      end while
      if queue[i].empty()
        DC[i]:= 0
      end if
    end if
  end for
end while
```

D'après [DRR], il est mentionné que le modèle DRR se caractérise :

- Une très grande isolation du flux.
- Un coût d'implémentation assez faible.
- Une assurance d'équité quel que soit la longueur des paquets.
- Des résultats d'équité avec différents types de trafic donnent :

$$\forall(i,j);\ \forall(t_1,t_2),\left|\frac{w_i(t_1,t_2)}{r_i}-\frac{w_j(t_1,t_2)}{r_j}\right|\leq\ 2\,max+Q$$

Avec, *max* : la longueur maximale d'un paquet, *Q :* le quantum associé au flux *i.*

5. Conclusion :

Dans ce chapitre on a fait le parcours des principales types d'ordonnancement, ceux de référence GPS, RR, BR et ceux actuellement implémentés, les anciens comme les récents FIFO, WFQ, WRR, DRR, DWRR… . Autres modèles ne sont pas détaillés ici à cause de leur complexité, comme (Virtual Clock) [VC90], [VC91], SCFQ (Self-Clocked Fair Queuing) [SCFQ], LL-WRR (Low Latency WRR) [LLWRR], et plein d'autres.

La diversité des modèles d'ordonnancement est motivée par la multiplicité des paramètres et métriques intervenant dans la garantie d'une meilleure qualité de service QoS. En effet, chaque

discipline se base sur la garantie de l'un ou quelque uns de ces métriques : débit, gigue, délai (latence), perte …, et ça peut satisfaire quelques classes de service, mais la combinaison de tous ces paramètres dans un algorithme équitable, qui satisfait tous les besoins en QoS, est loin d'être atteinte.

L'autre motivation est de proposer un traitement moins complexe, facile à implémenter et consommant moins de ressources. A cause de ces derniers desseins, les ordonnanceurs fondés sur les flux deviennent de plus en plus complexe, et leur implémentation nécessite plus de mémoire avec la montée des files d'attentes relative à l'augmentation des flux dans le réseau. Par conséquent, ils sont plus adaptés pour les routeurs de périphéries ER (Edge Router). Par contre ceux fondés sur les classes sont plus aisés à implémenter en raison du nombre fixe de ces dernières. Leur rapidité de traitement, et la faible mémoire utilisée les font convenir pour les routeurs de cœur du réseau CR (Core Router).

Parfois il est judicieux de combiner plusieurs mécanismes pour avoir des bons résultats. Réserver une partie des ressources entièrement à un trafic (ensemble de classes fixe) particulier, et laisser l'autre au reste du trafic (ensemble de classes dynamique) ; et appliquer ainsi l'ordonnancement adéquat à chaque trafic. Beaucoup de situations sont envisageables, reste à prouver leur performances et apports en QoS.

Chapitre 03 : Conception et Implémentation

1. Introduction :

Toutes les simulations faites dans le contexte d'ordonnancement cherchent à démontrer l'efficacité de nouveaux algorithmes ou argumenter d'autres. En effet, les mécanismes actuels d'ordonnancement sont déjà implémentés et mis en œuvre dans les nœuds du réseau, par conséquent ils intègrent déjà les outils de simulations actuels comme NS2, GNS3, OPNET et d'autres. Ainsi, la raison pour laquelle on n'a pas recours à ces outils est claire : on ne veut rien y ajouter. Notre objectif est d'étudier les algorithmes déjà existants et fonctionnels.

Donc, pour des fins éducatives et pédagogiques, on va concevoir et implémenter les algorithmes détaillés dans le chapitre précédent par un langage simple Java. La visualisation et l'analyse des résultats seront très faciles à comprendre et à comparer.

Les scénarios que l'on va développer servent à comparer les différents algorithmes étudiés en terme de :

- Performance (délai, gigue, débit, …)
- Compatibilité QoS
- Equité entre les classes de service

2. Outils de développement :

Le paradigme orienté objet est très répandu, beaucoup utilisé, et même apprécié et demandé pour la réalisation des projets de fin d'études. Par conséquent le langage Java est de loin le plus simple et le plus ouvert et célèbre grâce au grand nombre de communautés actives à son développement et son amélioration et par la multiplicité des plugins et APIs existants (Spring, Maven, Hibernat, …) et surtout par la diversité des environnements de développement intégrés open source (Eclipse, NetBeans, …).

2.1. Environnement :

Netbeans est l'environnement ouvert (open source) le plus célèbre car il offre tous les outils et technologies standards pour développer des projets simples ou de grande envergure :

- Analyse de code, complétion et indentation intelligente ;
- Modèles utilisables ;
- Conception rapide des interfaces utilisateurs ;
- Gestion du projet selon plusieurs vues ;

2.2. JavaFX 8 :

Un ensemble d'API$_s$ Java pour la programmation des applications du bureau, c'est une extension à la librairie Swing qui fonctionne sur toutes les plateformes où JVM est installée. Elle contient en addition des composants java habituels :

- des API$_s$ pour les graphes et courbes ;
- des API$_s$ pour la programmation WEB ;
- des API$_s$ pour le multimédias (audio, vidéo,…) ;
- un langage de balisage FXML comme alternative à la conception graphique. Il sert généralement à séparer la partie applicative (contrôle) d'un programme de celle destinée à la conception visuelle de l'interface utilisateur (vue). Ainsi, il réduit le code java de l'application ;
- des API$_s$ pour les animations : transitions et temporisations ;

Dans notre contexte de simulation, JavaFX n'est rien d'autre que le Framework Java intégrant des API$_s$ facilitant les tracés des graphes et courbes concernant les résultats de nos scénarios d'ordonnancements.

2.3. JDK 8 :

L'apport de la version 8 de Java est l'introduction de l'aspect fonctionnel par les expressions lambda. D'autre part, la repensée de la programmation concurrentielle dans cette version est très poussée. Par conséquent, elle nous facilite la mise en œuvre de plusieurs instances de générateurs en même temps et l'exécution de plusieurs threads en parallèle.

3. Conception :

Vu l'aspect concurrentiel et l'aspect visuel dans notre simulation, on est amené à utiliser des patterns de conception, les automates et des schémas symboliques très simples. Ensuite, on donnera les diagrammes de classe de quelques modules.

3.1. Conception d'un système d'attente :

Tout système d'attente est caractérisé par les éléments de base suivants :

- Loi d'arrivée des clients (paquets) ;
- Loi de traitement (service) ;
- Durée d'attente (avec le temps de traitement forment le délai) ;
- La taille de la file (tampon) ;
- Nombre de serveurs (processus) ;

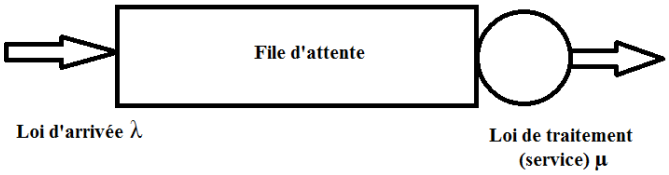


Figure 1 Caractéristiques d'une file d'attente

Les files d'attente apparaissent systématiquement quand la capacité du lien de sortie est inférieure à la somme de tous les débits des flux en entrées transitant vers cette sortie. Cependant, on les trouve aussi même dans des systèmes non congestionnés. En effet, l'arrivée des paquets dans un système n'est pas à intervalle régulier mais plutôt aléatoire et parfois par rafales. Pour cela, même dans un routeur ayant une capacité plus élevée, des files d'attente se forment et des congestions temporaires peuvent surgirent.

Cela dit, on ne peut pas prévoir le temps d'arrivée d'un trafic ou le nombre de paquets qui survient dans un réseau réel. En revanche, il nous faut bien nous disposer d'une source de trafic dans un environnement de simulation. Les travaux et les simulations effectuées se basent sur des modèles de trafic aléatoire [THI] et [HAS], En particulier, le processus de poisson, exponentiel et markovien qui approchent assez bien le comportement réel des trafics transitant dans les réseaux.

Le modèle de poisson simule le nombre de paquets arrivés par unité de temps, tandis que le modèle exponentiel est utilisé soit pour l'approximation du temps d'inter-arrivée des paquets ou bien le temps de leurs services.

3.2. Génération de paquets :

Pour générer une variable aléatoire X qui suit un processus probabiliste, dont la fonction de répartition est F, on adoptera la transformation inverse F^{-1} qui a la même distribution que X. Si $U_{[0,1]}$ est la valeur générée suivant une loi uniforme sur [0,1], on pose $X= F^{-1}(U)$. Pour la loi de poisson de paramètre λ :

$$F(X) = 1 - e^{-\lambda x}, \text{ pour } x > 0 ;$$

La fonction inverse est :

$$X = -\frac{1}{\lambda} \ln(U)$$

Dans [DM] P. Del Moral a démontré que le paramètre λ dans la formule $X = -\frac{1}{\lambda}\log(U)$représente la fréquence d'arrivée des paquets par unité de temps, autrement dit, les paquets arrivent en moyenne toutes les (1/λ) unités de temps. "Pour simuler l'arrivée d'un client

dans une file d'attente, on prend donc tout simplement le logarithme d'un nombre uniforme sur [0,1] fourni par l'ordinateur, et on multiplie le résultat par (-1/ λ) "

On suppose avoir un générateur de trafic aléatoire pour nos tests avec un taux d'arrivée paramétrable pour chaque flux. Toutefois, on prend des modèles de données bien déterminés et connus à l'avance pour les tests, ces modèles sont téléchargés à partir d'une source de données stockée dans un fichier texte, XML, CSV ou autres.

3.3. Classification :

Selon les scénarios simulés et le nombre de files d'attente, on instancie le nombre de générateur y correspondant. Les classes sont marquées par le champ ToS dans l'entête IP standard, qui est utilisé presque par toute les architectures traitant la QoS. Ce marquage est fait selon des critères de débit, destination, nature du flux ou autre. Donc le module d'ordonnancement est supposé être au cœur du réseau et ne traite que des paquets déjà classifiés, marqués, contrôlés et régulés dans un nœud de bordure. Ce dernier est vu comme un module à part séparé intégrant les dits générateurs.

3.4. Configuration interne :

Pour le routeur on suppose qu'il offre un seul service avec un taux de traitement fixe µ. Pour se focaliser entièrement sur les mécanismes d'ordonnancement, on prend, dans un premier temps, une taille illimitée du tampon. Ensuite, en limitant cette taille, on pourra gérer les congestions qui risquent de survenir par les mécanismes déjà détaillés $2.

4. Implémentation :

Les possibilités offertes sont multiple : Threads, Pipes, Sockets, Queues. On a choisis la programmation concurrentielle et multi threading, vu l'aspect temporelle, visuel et la nature de notre simulation :

- Plusieurs générateurs de paquets : aléatoires ou à partir d'un fichier ;
- Arrivées des paquets et service d'ordonnancement en concurrence ;
- Animations et transitions instantanée des paquets ;

Les patrons de conception nous viennent au secours par l'utilisation du patron Observateur où tous les paquets observent le changement de l'état de la file pour modifier leurs comportements. Le patron état (State) prend la responsabilité de réagir à ce changement. Notamment le pattern producteur / consommateur où les producteurs de paquets alimentent les files qui sont partagées avec un seul consommateur qui n'est d'autre que l'ordonnanceur.

4.1. Le pattern Observer :

A l'état initial toutes les files d'attente sont vides, dès que leurs générateurs commencent à produire des paquets (arrivées), elles vont prendre plusieurs états selon le nombre de paquets s'y trouvant. On peut concevoir les états de chaque file par une liste dynamique de paquets.

La file implémente donc l'interface Observable qui notifier tous les paquets y enregistrés comme observateurs. Ces derniers seront informés par l'état instantané de la file et réagissent en conséquence. La notification se fait à chaque arrivée et à chaque fin de service d'un paquet. Mais aussi, pour visualiser l'avancement, la notification sera déclenchée quand un paquet entre en attente, en avance ou en service.

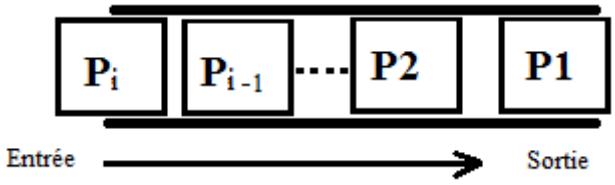


Figure 2 Etat possible d'une file d'attente

4.2. Le pattern State :

Un paquet (Observer) dans l'une des files d'attente est soit en état START, AVANCE, PAUSE, SERVICE ou FIN. La réaction de chaque paquet à la notification de la file (Observable) se fait selon le patron State.

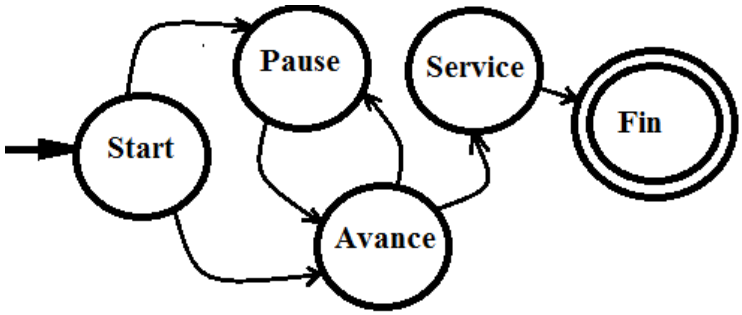


Figure 3 Les états possibles d'un paquet

4.3. Terminologie et relations de base :

λ : taux d'arrivée des paquets

μ : taux de service

η_i : nombre moyen de paquets qui attendent d'être servis

t_i : temps moyen d'attente en file

η_s : nombre moyen de paquets dans le système

t_s : temps moyen dans le système

$1/\mu$: temps moyen de service

$1/\lambda$: temps moyen d'inter arrivée

$\rho = \lambda / \mu$: taux d'utilisation du système (charge, taux de trafic, activité du serveur)

Le taux d'arrivée (λ) et le taux de service (μ) doivent être exprimés par la même unité de mesure.

La condition de stabilité : $\rho = \lambda / \mu < 1$, c.à.d. débit d'entrée < débit du serveur.

4.4. Diagramme de classes :

Vu l'aspect orienté objet du langage Java, UML (Unified Modeling Language) est le mieux adapté pour modéliser notre travail. Un diagramme de classe simple illustre les principales classes nécessaires implémentées :

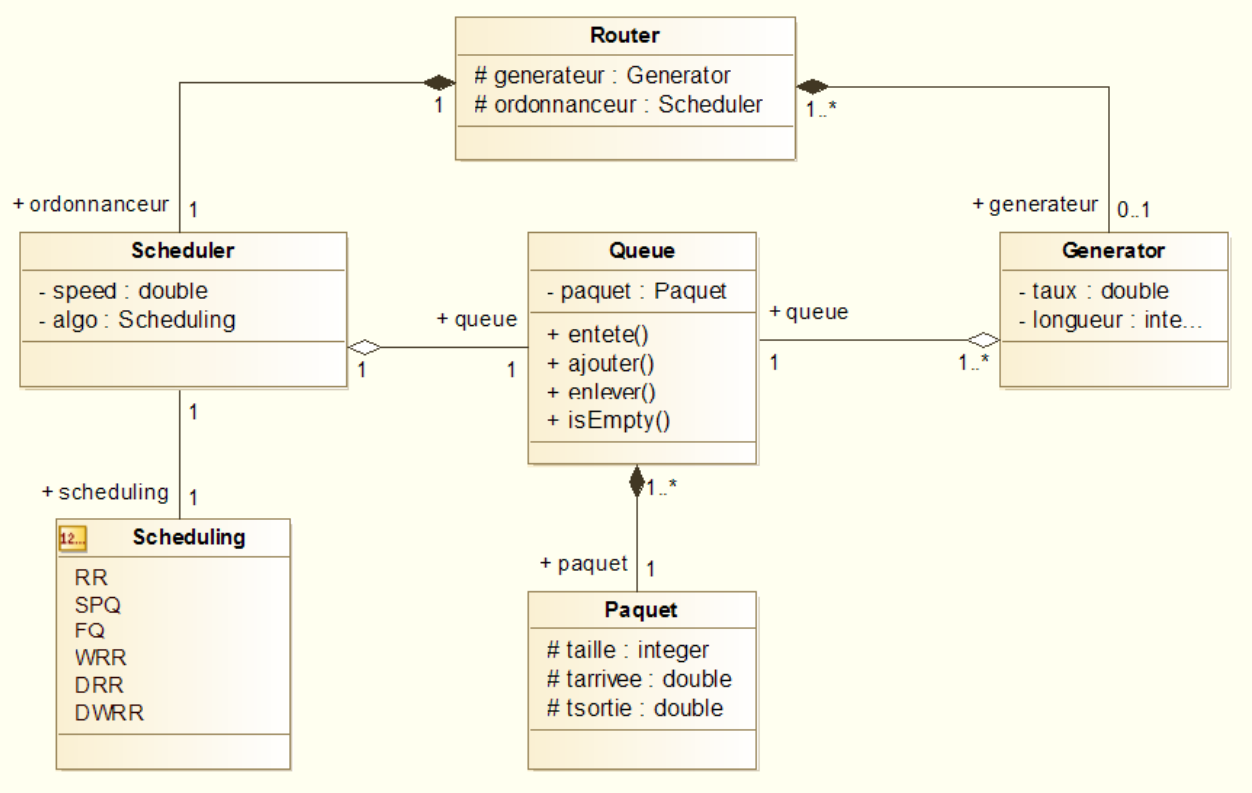


Figure 4 *Diagramme de classes*

4.5. Interface utilisateur :

L'interface utilisateur est composée de trois volets :

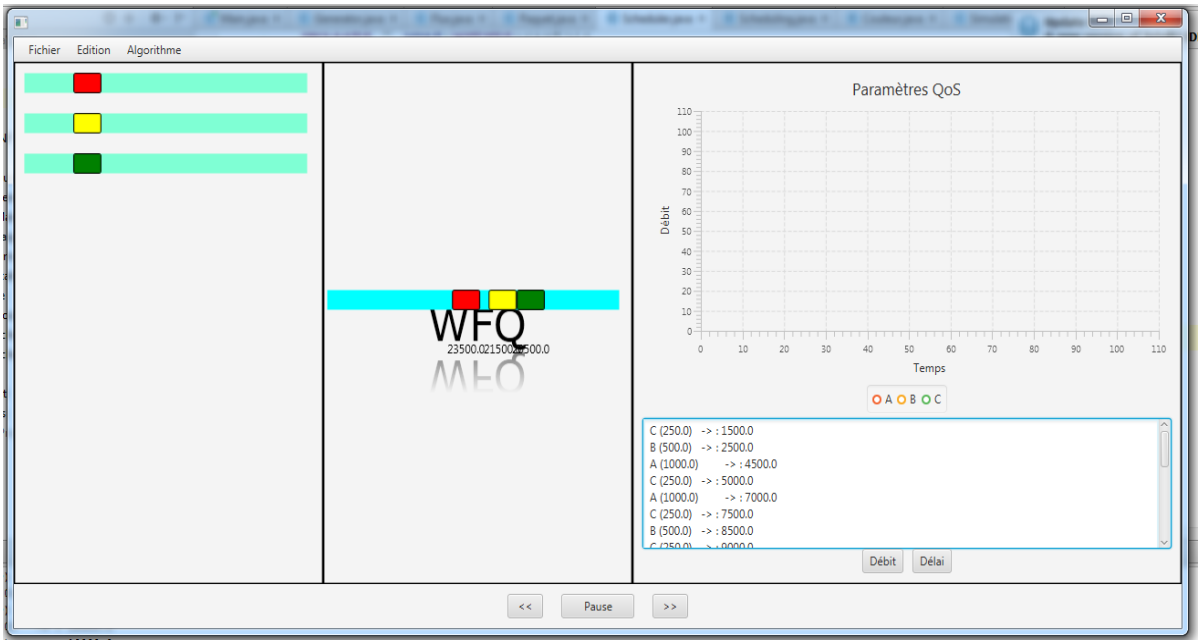


Figure 5 Interface utilisateur

- Le volet à gauche pour les flux entrants ;
- Le centre pour la file de sortie ;
- Le dernier est laissé pour l'affichage des métriques QoS ;

5. Résultats et analyses :

Pour bien analyser et comparer les différents algorithmes cités dans le chapitre précédents, on commence par des tests simples à partir d'un fichier texte. Ces derniers peuvent être validés manuellement afin de comparer les résultats, et par la suite faire des tests beaucoup plus réels par l'utilisation d'un générateur de paquets aléatoire à l'entrée du routeur.

5.1. Scénario 1 :

Le modèle utilisé est un ensemble de trois entrées ; leur apparition dans le fichier défini leur priorité :

Temps (µs)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000
1000 octets	A	A		A	A	A		A	A	A	A
500 octets	B		B	B	B	B	B	B	B	B	B
250 octets	C	C	C	C	C		C	C		C	C

Tableau 1 Données du scénario N:1

Les déficits sont respectivement : 500, 500, 1000.

Les poids sont respectivement : 0.5, 0.5, 1.

Les débits sont mesurés en octets par ms et les délais en ms.

	RR		PQ		DWRR		FQ		WFQ	
A	290.32	31000	391.30	23000	276.92	32500	276.92	32500	277	32500
B	156.25	32000	208.33	24000	166.66	30000	163.93	30500	164	30500
C	69.23	32500	91.84	24500	73.77	30500	76.27	29500	76	29500

Tableau 2 Débits et délais sous diférents algorithmes d'ordonnancement du scénario 01

Ces résultats montre bien les lacunes de quelques algorithmes vues au chapitre précédent. En effet, dans ce scénario, les flux ont chacun des paquets de même taille, celui ayant des paquets de grande taille domine toujours avec le meilleur débit. Car même, en inversant les priorités, le flux "A" de grande taille obtient toujours les bons résultats.

	RR		PQ		DWRR		FQ		WFQ	
B	167	30000	179	28000	167	30000	164	30500	164	30500
C	74	30500	79	28500	74	30500	76	29500	76	29500
A	277	32500	295	30500	277	32500	277	32500	277	32500

Tableau 3 Résultats du scénario 01 modifié

Pour cette raison des mécanismes comme le seau percé, le seau à jeton, SrTCM et TrTCM ont été développés et implémentés dans les nœuds du réseau afin de réguler et contrôler le trafic y circulant. Aussi, le découpage des paquets exigé par certains nœuds dans le réseau participe à maintenir une bonne QoS.

5.2. Scénario 2 :

Le deuxième scénario simule des entrées avec des flux ayant différentes tailles de paquets mais de quantités globales approximativement égaux :

t (µs)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000	12000
A	400	600		300	500		600	100	300		200	500
B		400	500		400	350		100	600	400		300
C	500		700	200		100		800	700		200	600

Tableau 4 Données du scénario N : 2

Les déficits sont respectivement : 400, 600, 1000.

Les poids sont respectivement : 0.4, 0.6, 1.

	RR		FQ		PQ		DWRR		WFQ	
A	185	18900	179	19500	205	17100	169	20700	169	20700
B	165	18500	174	17500	183	16700	174	17500	163	18700
C	184	20700	184	20700	201	18900	193	19700	199	19100

Tableau 5 Débits et délais sous diférents algorithmes d'ordonnancement du scénario 02

Dans ce scénario, on voit bien comment les algorithmes pondérés (DWRR, WFQ) offrent le moyen de différencier les demandes de QoS, et cela avec les poids attribués à chaque classe de service.

6. Conclusion :

Les résultats analysés précédemment sont généraux et ne montrent pas la convergence des algorithmes étudiés à long terme. Les graphes qui suivent montrent qu'il y a une relation entre les trois paramètres de qualité de service : débit, gigue, délai. L'amélioration de deux paramètres est au détriment de l'autre. En plus, les débits se stabilisent mais les délais augmentent et conduiront obligatoirement à des congestions.

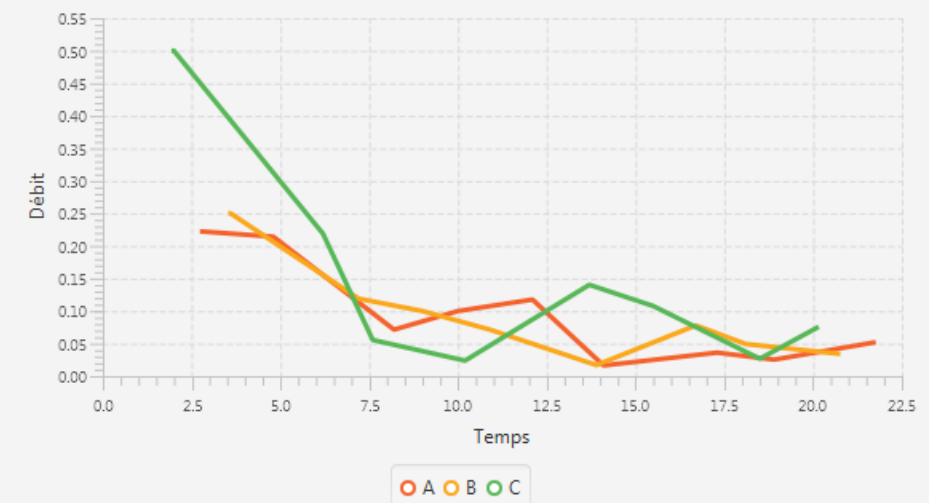


Figure 6 Variation de débit WFQ scénario 02

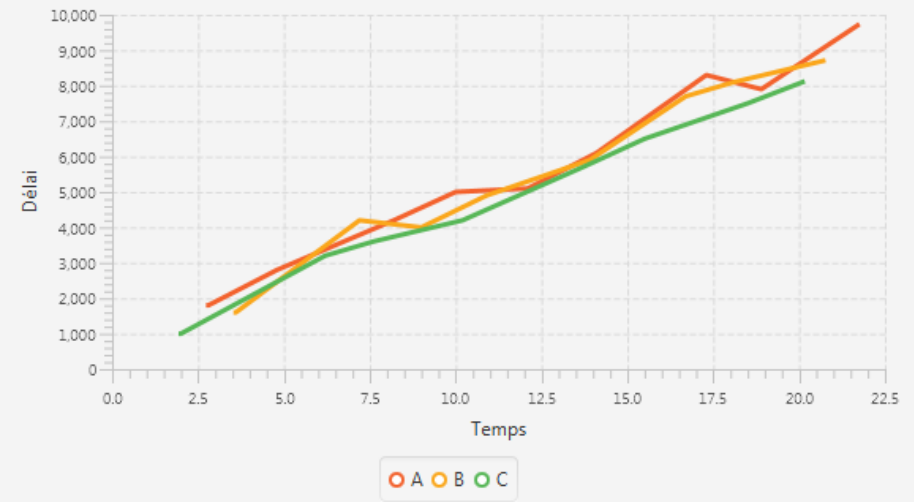


Figure 7 Variation du délai WFQ scénario 02

C'est la raison pour laquelle les nœuds du réseau sont toujours équipés avec des modules de traitement de congestion en parallèle avec l'ordonnanceur.

Le tableau suivant illustre l'ordre de sortie des paquets qui ont servis à élaborer les graphes précédents. Le nom du paquet avec sa taille en octets en première ligne ; dans la deuxième ligne les temps de sortie correspondant.

B(350)	C(100)	A (500)	B(400)	A(300)	C(200)	B(500)	C(700)	A(600)	B(400)	A (400)	C (500)
10900	10200	10000	9000	8200	7600	7200	6200	4800	3600	2800	2000

A(500)	B(300)	C(600)	A(200)	C(200)	B(400)	A(300)	B(600)	C(700)	A(100)	B(100)	C(800)	A(600)
21700	20700	20100	18900	18500	18100	17300	16700	15500	14100	13900	13700	12100

Conclusion générale

Les travaux élaborés par la communauté internet pour offrir des capacités de QoS à internet a permet de construire plusieurs approches : IntServ, DiffServ et l'ingénierie de trafic. Ce qui a fait créer plusieurs domaines de recherches. Parmi ces derniers le mécanisme d'ordonnancement est très actif, malgré l'énorme publication de documents et recherches y consacrés. La technique ou l'algorithme satisfaisant est loin d'être atteint. En effet, un ordonnanceur doit mettre trois paramètres en évidence : le délai d'attente qu'un paquet doit supporter afin d'aboutir à la sortie, le taux de perte qu'un flux peut subir, et enfin le débit qui doit être garanti.

Les trois paramètres cités ne sont pas indépendants, car le maintien de l'un d'entre eux est au détriment des autres. En effet, en offrant une capacité illimitée à la file d'attente pour éviter la perte risque d'avoir un délai très important. En revanche, en limitant la capacité de la file, un taux de perte est inévitable mais contrôlable et les délais seront minimes. Et pour un débit fixe, en réduisant les délais le taux de perte augmente.

Tous les algorithmes cités dans le deuxième chapitre mettent un compromis entre les paramètres mentionnés au-dessus. Aucun algorithme n'est parfait pour satisfaire toutes les demandes de QoS. Néanmoins, chacun est adapté pour certaines situations que d'autres. Peut-être un algorithme dynamique s'adaptant à toutes les situations verra le jour et sera standardisé.

Bibliographie

[LTO] Leila Toumi.*"Algorithmes et mécanismes pour la qualité de service dans des réseaux hétérogènes"*. Réseaux et télécommunications [cs.NI]. Institut National Polytechnique de Grenoble - INPG, 2002

[ELO] Emmanuel Lochin. *"Qualité de Service dans l'Internet : Garantie de Débit TCP dans la Classe AF"*. Réseaux et télécommunications [cs.NI]. Université Pierre et Marie Curie - Paris VI, 2004.

[MAG1] J. Gozdecki, A. Jajszczyk, R. Stankiewicz," *Quality of Service Terminonolgy in IP Network"*, IEEE Communications Magazine Mars 2003.

[IS] R. Braden, D. Clark, S. Shenker, *"Integrated Service in the Internet Architecture: an Overview,"* IETF RFC 1633, Juin 1994.

[RED] S. Floyd, V. Jacobson, *"Random Early Detection Gateways for Congestion Avoidance",* IEEE/ACC Transactions in Networking,Vol. 3. 397-413.Aout 1993,

[WRED] CISCO, "*Congestion Avoidance Overview*", Cisco IOS Quality of Service Solutions Configuration Guide, QC188,

[JTU] Turner, J. S. *"New Directions in Communications"* . IEEE Communications Magazine, vol. 24, pp. 8 – 15. October 1986,

[RFC791] Postel, J., "*Internet Protocol*", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981,

[RFC1349] Almquist, P., "*Type of Service in the Internet Protocol Suite",* RFC 1349, DOI 10.17487/RFC1349, July 1992,

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "*Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*", RFC 2474, DOI 10.17487/RFC2474, December 1998,

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "*An Architecture for Differentiated Services*", RFC 2475, DOI 10.17487/RFC2475, December 1998,

[RFC2697] Heinanen, J. and R. Guerin, "*A Single Rate Three Color Marker*", RFC 2697, DOI 10.17487/RFC2697, September 1999,

[RFC2698] Heinanen, J. and R. Guerin, "*A Two Rate Three Color Marker*", RFC 2698, DOI 10.17487/RFC2698, September 1999,

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "*The Addition of Explicit Congestion Notification (ECN) to IP*", RFC 3168, DOI 10.17487/RFC3168, September 2001,

[RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "*Overview and Principles of Internet Traffic Engineering*", RFC 3272, DOI 10.17487/RFC3272, May 2002,

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "*LDP Specification*", RFC 5036, DOI 10.17487/RFC5036, October 2007,

[LTO] Leila Toumi. *"Algorithmes et mécanismes pour la qualité de service dans des réseaux hétérogènes".* Réseaux et télécommunications [cs.NI]. Institut National Polytechnique de Grenoble - INPG, 2002. Français.

[ELO] Emmanuel Lochin. *"Qualité de Service dans l'Internet : Garantie de Débit TCP dans la Classe AF".* Réseaux et télécommunications [cs.NI]. Université Pierre et Marie Curie - Paris VI, 2004. Français.

[DGA] David Gauchard. *"Simulation hybride des réseaux IP-DiffServ-MPLS multi-services sur environnement d'exécution distribuée"*. Automatique / Robotique. Université Paul Sabatier - Toulouse III, 2003. Français

[Nag87] J. Nagle, *"On Packet Switches with Infinite Storage",* IEEE Transactions on Communications, Volume 35, pp 435-438, 1987.

[Dem89] A. Demers, S. Keshav and S. Shenkar, *"Analysis and Simulation of a Fair Queuing Algorithms,"* Proceedings of SIGCOMM '89, August 1989, pp. 3-12

[Dem90] A. Demers, *"Analysis and Simulation of Fair Queuing Algorithm"*, Internetworking: Research and experience, Vol. 1, 3-26, 1990.

[Zh96] J. C. R. Bennett and H. Zhang, *"WF2Q: Worst-case Fair Weighted Fair Queuing,"* Proceedings of INFOCOM '96, March 1996, pp. 120-128.

[Par92] A.K. Parekh, R.G. Gallager, *"A generalized processor sharing approach to flow control in integrated services networks-the single node case"*, IEEE INFOCOM '92, 1992, pp. 521-530.

[Par93] A.K. Parekh, R.G. Gallager, *"A generalized processor sharing approach to flow control in integrated services networks-the multiple node case"*, IEEE INFOCOM '93, 1993, pp. 915-924

[SFQ] P. Goyal, H. M. Vin and Haichen Cheng, *"Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks,"* in IEEE/ACM Transactions on Networking, vol. 5, no. 5, pp. 690-704, Oct 1997.

[SCFQ] S. J. Golestani, *"A self-clocked fair queuing scheme for broadband applications,"* INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE, Toronto, Ont., 1994, pp. 636-646 vol.2.

[DRR] M. Shreedhar et G. Varghese, *"Efficient Fair Queuing Using Deficit Round Robin,"* IEEE/ACM Transactions On Networking, Vold, 4, No, 3, Juin 1996.

[YJ02] Yuming Jiang, "*Delay bounds for a network of guaranteed rate servers with FIFO aggregation,"* Computer Networks: The International Journal of Computer and Telecommunications Networking, v.40 n.6, p.683-694, 20 December 2002

[LLWRR] Z. Patel, U. Dalal, *"Design and Implementation of Low Latency Weighted Round Robin (Llwrr) Scheduling For High Speed Networks"*, International Journal of Wireless & Mobile Networks (IJWMN) Vol. 6, No. 4, August 2014.

[VC90] L. Zhang, *"VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks"*, ACM 089791-405-8/90/0009/0019, 1990.

[VC91] L. Zhang, *"VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks"*, ACM Transactions on Computer Systems, Vol. 9, No. 2, Pages 101-124, May 1991.

[CBQ] *Sally Floyd and Van Jacobson. "Link-sharing and resource management models for packet networks". IEEE/ACM Trans. Netw.* 3, 4 (August 1995)

[THI] E. Thibault *"Etat de l'Art de la modélisation des Flux Multimédia",* INRIA Sophia Antipolis, Rapport du Rrojet RNRT Esquimaux, 2001

[HAS] H. Hassan, J. Garcia, O. Brun, and D. Gauchard.*"Caractérisation de l'autosimilaritéde trafics web et ftp par un outil de simulation"*. In 7eme rencontres francophones surles aspects algorithmiques des télécommunications (ALGOTEL'2005), Presqu'ile de Giens(France), Mai 2005.

[DM]P. Del Moral,"*Précis de simulation*", Centre INRIA Bordeau Sud-Ouest & Institut mathématique de Bordeau, Université Bordeau I.

Liste des acronymes

ADSL	Asynchr		**LDP**	Label Distribution Protocol
AF	Assured Forwarding		**LER**	Label Edge Router
AQM	Active Queue Management		**LLQ**	Low Latency Queuing
ATM	Asynchronous Transfer Mode		**LSP**	Label Switching Path
BE	Best Effort		**LSR**	Label Switch Router
BP	Bande Passante		**MCR**	Minimum Cell Rate
BR	Border Router		**MPLS**	Multi Protocol Label Switching
Bc	committed Burst		**OSPF**	Open Shortest Path First
Be	excess Burst		**PCR**	Peak Cell Rate
BGP	Border Gateway Protocol		**PHB**	Per-Hop Behavior
CBR	Constraint Based Routing		**PIR**	Peak Information Rate
CBS	Committed Burst Size		**PPP**	Point to Point Protocol
CR	Core Router		**QoS**	Quality of Service
CIR	Committed Information Rate		**RED**	Random Early Detected
CL	Controlled Load		**RFCs**	Request for Comments
DiffServ	Differentiated Services		**RSVP**	Resource Reservation Protocol
DS	Differentiated Services		**RTT**	Round Trip Time
DSCP	DiffServ Code Point		**SLA**	Service Level Agreement
EBS	Excess Burst Size		**SrTCM**	Single Rate Three Color Marker
ECN	Explicit Congestion Notification		**TCP**	Transfer Control Protocol
EF	Expedited Forwarding		**TE**	Traffic Engineering
EGP	Exterior Gateway Protocol		**TED**	Traffic Engineering Database
ER	Edge Router		**ToS**	Type of Service
FEC	Forwarding Equivalence Class		**TrTCM**	Twoo Rate Three Color Marker
FR	Frame Relay		**TTL**	Time To Live
GPS	Généralized Process Sharing		**UIT**	Union International des Télécommunications
GS	Guaranteed Service		**UDP**	User Datagram Protocol
ICMP	Internet Control Message Protocol		**VBR-rt**	Variable Bit Rate, Real-Time
IETF	Internet Engineering Task Force		**VoIP**	Voice over IP
IGP	Interior Gateway Protocol		**VCI**	Vitual Channel Identifier
ISP	Internet Service Provider		**VPI**	Virtual Path Identifier
IntServ	Integrated Services		**VPN**	Virtual Private Network
IP	Internet Protocol		**WAN**	Wide Area Network
IS-IS	Intermediate System-to-Intermediate System		**WRED**	Weighted Randomly Early Detected