Algorithmes et mécanismes pour la qualité de service dans des réseaux hétérogènes

Leila Toumi

Institut National Polytechnique de Grenoble

N° attribué par la bibliothèque

|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : **Informatique, Systèmes et Communications**

préparée au Laboratoire Logiciels, Systèmes et Réseaux

dans le cadre de l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

présentée et soutenue publiquement par

Leila TOUMI

Le 20 décembre 2002

ALGORITHMES ET MÉCANISMES POUR LA QUALITÉ DE SERVICE DANS DES RÉSEAUX HÉTÉROGÈNES

Directeur de Thèse :

M. Andrzej Duda

Jury :

Mme Brigitte PLATEAU	Président
Mme Pascale VICAT-BLANC PRIMET	Rapporteur
M. Ahmed SERHROUCHNI	Rapporteur
M. Andrzej DUDA	Directeur de thèse

« Notre vie vaut ce qu'elle nous a coûté d'efforts »

François Mauriac

A mes parents,

A mon frère,

Qu'ils trouvent ici le fruit de leur patience
et du soutien permanent et quotidien
qu'ils m'ont prodigué pour affronter
tous les moments difficiles

Remerciements

Cette thèse, intitulée « Algorithmes et Mécanismes pour la Qualité de Service dans des Réseaux Hétérogènes » a été soutenue le 20 décembre 2002 à l'Institut National Polytechnique de Grenoble devant le jury composé de Mme Brigitte Plateau, Mme Pascale Primet, Mr Ahmed Serhrouchni et M. Andrzej Duda.

Je tiens à remercier tous les membres de ce jury :

- Mme Brigitte Plateau, Professeur à l'INPG, qui m'a fait l'honneur de bien vouloir présider ce jury de thèse ;
- Mme Pascale Primet, Maître de Conférences à l'Ecole Normale Supérieure de Lyon, ainsi que Mr Ahmed Serhrouchni, Maître de Conférences à l'Ecole Nationale Supérieure des Télécommunications de Paris, qui ont accepté d'être les rapporteurs de ma thèse et pour l'intérêt qu'ils ont porté à ce travail ;
- M. Andrzej Duda, Professeur à l'Institut National Polytechnique de Grenoble, qui a été mon directeur de thèse et qui m'a accueilli chaleureusement, soutenu et encouragé durant toutes ces années de travail, et qui m'a inculqué tout ce que je n'aurai pas pu apprendre sans sa présence.

Un grand merci à mon collègue de bureau Dominique Decouchant, Chercheur au CNRS, qui a eu la tendre gentillesse de m'avoir agréablement et paternellement épaulé durant cette dernière année qui a été la plus difficile, mais aussi la plus souriante grâce à sa présence et sa bonne humeur permanentes.

Mes sincères remerciements s'adressent à Franck Rousseau, Maître de Conférences à l'ENSIMAG, pour m'avoir apporté tout son soutien et ses encouragements, notamment lors des moments les plus difficiles.

Je tiens à remercier vivement Miroslav Klun, chercheur au sein de la société « Verizon Communication » (Boston-USA), pour son abnégation et ses capacités à résoudre les multiples problèmes sous *ns-2*, pour tous ses précieux conseils et ses multiples entretiens téléphoniques qui m'ont été très enrichissants.

Mes sincères remerciements sont adressés à toute l'équipe drakkar, au sein de laquelle j'ai été agréablement accueillie pour effectuer ce travail.

Enfin, je ne pourrai oublier les membres du laboratoire qui m'ont apporté un grand confort de travail dans la joie et la bonne humeur : tout d'abord, les Professeurs Jean-Pierre Giraudin qui m'a apporté de précieux conseils et Dominique Rieu pour ses conseils maternels et son écoute permanente. Christiane Plumeré pour son amabilité à pouvoir résoudre les petits incidents techniques qui arrivaient aux mauvais moments , sa sympathie humaine et son sourire naturel ; Martine Pernice, Liliane Di Giacomo, Solange Roche, pour le soutien qu'elles m'ont apporté, leur bonne humeur quotidienne et leur précieuse aide administrative ; les étudiants Mhamed Saidane et Walid Khayati pour les longues nuits passées dans le laboratoire. Samia Menif, Ibtissem Hassine, Tony, Stéphane Lo-Presti et tous ceux qui m'ont longuement soutenu et encouragé lors des moments difficiles et avec qui j'ai partagé les pires et les meilleurs moments de la thèse.

Résumé

Les réseaux actuels sont assujettis à certaines critiques, notamment lorsque le trafic de données qui les traverse se diversifie (Internet par exemple). La notion de qualité de service (QoS) devient alors un concept important si l'on désire transporter l'information avec un maximum de fiabilité. Les plus importantes métriques de QoS sont le débit, le délai (ainsi que la gigue) et le taux de pertes. Par ailleurs, la multiplicité des applications conduit à une hétérogénéité des données, ce qui amène à une classification en deux principales catégories : les flots élastiques qui représentent les applications qui requièrent un débit soutenu ; d'autre part, les applications non-élastiques qui sont exigeantes en délai. Pour garantir la qualité de service au niveau de ces applications, des études ont été menées pour effectuer de la différenciation de service et par conséquent attribuer un degré de priorité à chaque paquet appartenant à une classe de flot. De nos jours, plusieurs algorithmes d'ordonnancement sont mis en œuvre pour réaliser ce processus, et sont classés en deux catégories : tout d'abord, les ordonnanceurs différenciés tels que ABE ; ensuite, les ordonnanceurs à différenciation proportionnelle qui permettent de partager équitablement les ressources entre les applications, selon leurs besoins (WTP, par exemple). Toutefois, la proportionnalité ne s'est jusque-là effectuée que selon un seul critère de qualité, c'est-à-dire soit en terme de débit, de délai ou de taux de perte. Le travail de cette thèse consiste à cet effet, à remédier aux insuffisances présentées par les mécanismes de différenciation proportionnelle basée sur un seul paramètre de qualité. Il a donc été question de proposer un nouvel algorithme d'ordonnancement à différenciation proportionnelle, fondée sur la fonction de puissance mettant en jeu le délai et le débit : ordonnancement par PSP (Power as a Scheduling Parameter). Ainsi, un partage de ressources délai-débit s'effectuera équitablement entre les applications élastiques et non-élastiques de manière à ce qu'aucune classe ne puisse subir de phénomène de famine. L'implémentation et les tests de l'ordonnanceur ont été réalisés au moyen de l'outil de simulation *ns-2*. Les résultats offrant une validation probante de l'algorithme, des perspectives de recherches à base de cette nouvelle méthode sont à l'esprit.

Mots-clés : Qualité de service, flots élastiques et non-élastiques, garanties de délai et de débit, ordonnancement par PSP, différenciation de service, différenciation proportionnelle, hétérogénéité

Table des matières

Table des figures

Liste des abréviations

ABE	**A**lternative **B**est-**E**ffort
AF	**A**ssured **F**orwarding
BE	**B**est-**E**ffort
CBQ	**C**lass **B**ased **Q**ueuing
CL	**C**ontrolled-**L**oad
CSCP	**C**lass **S**elector **C**ode **P**oints
DiffServ	**D**ifferentiated **S**ervices
DSCP	**D**iff**S**erv **C**ode **P**oint
ECN	**E**arly **C**ongestion **N**otification
EDF	**E**arliest **D**eadline **F**irst
EDS	**E**quivalent **D**ifferentiated **S**ervices
EF	**E**xpedited **F**orwarding
FIFO	**F**irst **I**n **F**irst **O**ut
FQ	**F**air **Q**ueuing
GPS	**G**eneralized **P**rocessor **S**haring
GS	**G**uaranteed **S**ervice
HDP	**H**ybrid **P**roportional **D**elay
HFSC	**H**ierarchical **F**air **S**hare **C**urve
IETF	The **I**nternet **E**ngineering for **T**ask **F**orce
IntServ	**Int**egrated **Serv**ices
IP	**I**nternet **P**rotocol
MDP	**M**ean-**D**elay **P**roportional
OPWA	**O**ne **P**ass **W**ith **A**dvertising
PHB	**P**er-**H**op-**B**ehavior
PLR	**P**roportional **L**oss **R**ate
PQ	**P**riority **Q**ueuing
PSP	**P**ower as a **S**cheduling **P**arameter
QoS	**Q**uality of **S**ervice
RED	**R**andom **E**arly **D**etection
RIO	**R**ED with **I**n and **O**ut
RSVP	**R**esource re**S**er**V**ation setup **P**rotocol
RTCP	**R**eal-time **T**ransfer **C**ontrol **P**rotocol
RTP	**R**eal **T**ime **P**rotocol
SCFQ	**S**elf-**C**locked **F**air **Q**ueuing
Sdp	**S**cheduler **D**ifferenciation **P**arameter
SLA	**S**ervice-**L**evel **A**greement
TCP	**T**ransfer **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
UML	**U**nified **M**odeling **L**anguage
VC	**V**irtual **C**lock
WF²Q	**W**orst-case **F**air Weighted **F**air **Q**ueuing
WFQ	**W**eighted **F**air **Q**ueuing
WRED	**W**eighted **RED**
WTP	**W**aiting **T**ime **P**riority
YESSIR	**YE**t another **S**ender **S**ession **I**nternet **R**eservations

INTRODUCTION GÉNÉRALE

1. Introduction générale

Problématique

L'objet du travail qui a été présenté dans cette thèse provient directement d'une constatation quant à l'utilisation courante des réseaux de communication. Internet est un réseau à commutation de paquets qui fournit un service d'acheminement des paquets « au mieux », plus connu sous l'appellation « Best-Effort ». Par cela, nous entendons que le réseau n'est pas capable d'assurer une garantie de service aux paquets, soit en délai, soit en débit, soit en pertes. Tous les types de trafic qui traversent le réseau sont par conséquent traités de la même manière. Pour les applications qui circulaient aux débuts de l'Internet, ces limitations n'étaient pas incommodantes en raison de l'insensibilité aux variations temporelles des applications (le courrier électronique, ou le transfert de fichiers par exemple) ; d'autre part, la charge des réseaux était bornée ce qui laissait suffisamment de bande-passante disponible pour le trafic en circulation. Mais aujourd'hui, la naissance de nouvelles applications a fait d'Internet un réseau à usage plus hétérogène en terme de type de données et par conséquent plus délicat face à la garantie de service attendue par les utilisateurs. Ainsi, la vidéo et l'audio sont venus se substituer aux autres applications. Cependant, ces dernières ont des caractéristiques, sur le réseau, différentes de celles des applications insensibles au délai. De ce fait, il en découle deux principales catégories de trafic qui traversent couramment les réseaux :

- les applications que nous qualifions d'élastiques, caractérisées par une insensibilité à la métrique « temps » (telles que les *e-mails*); ce type de trafic devient alors plus « gourmand » en terme de débit ;
- les applications moins sensibles en débit mais exigeantes en délai, que nous qualifions de non-élastiques ; la vidéo-conférence et la téléphonie sur Internet en sont les exemples les plus répandus

Ces deux métriques sont, entres autres, à la base d'un besoin de garantie minimum à percevoir. La notion de Qualité de Service (ou *QoS*) intervient à cet effet, puisque pour satisfaire les besoins de ces applications, il est nécessaire de pouvoir leur fournir une qualité de service qui leur soit adaptée.

Depuis l'apparition de cette revendication, d'importants travaux sont menés dans le domaine de la qualité de service pour proposer des solutions remédiant aux insuffisances alors constatées. En particulier, les groupes de l'IETF (*The Internet Engineering for Task Force*) se sont penchés sur deux idées. Tout d'abord, un premier groupe, IntServ (*Integrated Services*) qui a proposé une architecture à base de réservation de ressources pour chaque flot. Cette

dernière s'effectue au moyen du protocole RSVP (*Resource Reservation Setup Protocol*) pour gérer cette réservation au niveau de chaque flot, et ainsi fournir de manière individuelle la ressource requise par l'application demandeuse de qualité de service. Cependant, cette solution s'est avérée insuffisante pour des réseaux à large dimensionnement. IntServ n'est pas propice à des réseaux de grande envergure.

C'est pourquoi un second groupe de travail a abordé le problème en proposant une garantie se service basée sur la différenciation de service. Le groupe DiffServ (*Differentiated Services*) fonde son concept sur une gestion de trafic par classe, sur des méthodes de conditionnement du trafic à l'entrée du réseau et sur le marquage de celui-ci en fonction de son appartenance à une classe, pour un traitement spécifique. Trois principaux services de DiffServ sont mis en œuvre : le service « Best-Effort » pour les paquets à traiter « au-mieux » comme le fait Internet actuellement ; le service « Assured Forwarding » pour des classes à priorité plus grande que le « Best-Effort » ; enfin le service « Expedited Forwarding » pour les classes les plus prioritaires.

Parallèlement à ces travaux, les routeurs ont de même pour fonction de pouvoir distribuer les ressources du réseau aux applications selon des politiques d'ordonnancement. Les algorithmes sont conçus pour un traitement selon une échelle microscopique des données, à savoir un traitement au niveau des paquets et non des flots. L'ordonnancement est en effet le processus qui permet d'acheminer les paquets vers la sortie des routeurs selon une gestion déterminée. Les multiples travaux de recherches effectués sur cette notion ont conduit à un nombre important de techniques d'ordonnancement, chacune se basant sur une métrique pour appliquer l'algorithme de traitement des paquets. Nous distinguons à cet effet les mécanismes qui fondent leur traitement en fonction de l'aspect temporel des paquets, notamment le temps d'attente des paquets au sein d'une file d'attente. D'autres algorithmes sont conçus pour acheminer les paquets en fonction du débit qui leur est offert. L'ordonnancement adopte une forme de différenciation par classe de paquet et par métrique. Cependant, ces techniques ne sont pas parfaitement performantes pour tous les types de scénarii existants et dans la plupart des cas, certaines applications seront désavantagées par rapport à d'autres. Ceci a conduit à l'introduction d'une nouvelle notion dans le monde de l'ordonnancement : la proportionnalité.

La proportionnalité constitue, par définition, à attribuer à deux ou plusieurs entités une part proportionnelle de ressources. Du point de vue ordonnancement proportionnel, il s'agit par conséquent d'allouer à chaque classe de flots qui se présente une part équitable d'une métrique de qualité de service. Les études menées jusque-là dans cette direction ont traité les aspects suivants : différenciation proportionnelle de délai (ordonnancement WTP – *Waiting Time Priority*, par exemple), différenciation proportionnelle de pertes (ordonnancement PLR – *Proportional Loss Rate*). Par ces différenciations, seules les classes exigeantes en délai seront avantagées : les applications multimédia seraient donc toujours plus prioritaires que les applications élastiques. Si nous considérons ce mécanisme dans la réalité, vu la quantité d'applications temps-réel qui circule, nous observerons un faible pourcentage d'applications élastiques sur les réseaux. Or cette dernière situation n'est pas envisageable car il est nécessaire de laisser circuler ce type de trafic, de la même manière que le trafic non-élastique. C'est sur cette constatation que nous avons orienté l'axe de nos recherches. Ainsi, nous avons choisi de répondre aux besoins des deux catégories de trafic, en leur offrant de manière proportionnelle la réponse à leurs requêtes : besoin en débit pour les applications élastiques ; besoin en délai pour les applications temps-réel.

Le présent travail est donc construit à partir de ces derniers éléments, à savoir les exigences temporelles, en débit, et sur la différenciation proportionnelle. Il s'agit donc de proposer un prototype d'ordonnanceur capable d'acheminer les paquets appartenant à des classes de trafic différentes, tout en garantissant leurs besoins respectifs (en débit ou délai, selon le cas) pour une bonne qualité de service. D'autre part, tenir compte du phénomène de famine qui ne doit pas figurer lors des traitements, dans le but d'offrir à chaque classe son besoin sans pour autant discriminer l'une ou l'autre classe. Le procédé de notre approche découle d'une fonction de puissance définie dans [86] mettant en évidence le rapport des métriques « délai » et « débit ».

Plan du document

Le corps de cette thèse est organisé comme suit :

Le premier chapitre est dédié à l'évocation de la problématique du sujet de manière à comprendre les problèmes existants et le but visé par cette thèse.

Nous consacrons, dans un second temps, un chapitre à la qualité de service en définissant tout d'abord la notion en tant que telle, puis en présentant les paramètres qui garantissent la qualité de service dans un réseau. Nous exposons, de même, les différentes techniques développées par les communautés de recherches, à savoir les modèles à intégration de service *IntServ*, puis les modèles à différenciation de service *DiffServ*. Pour chacun de ces modèles, nous détaillons les architectures et les différents services offerts qui permettent d'assurer la qualité de service du trafic sur le réseau.

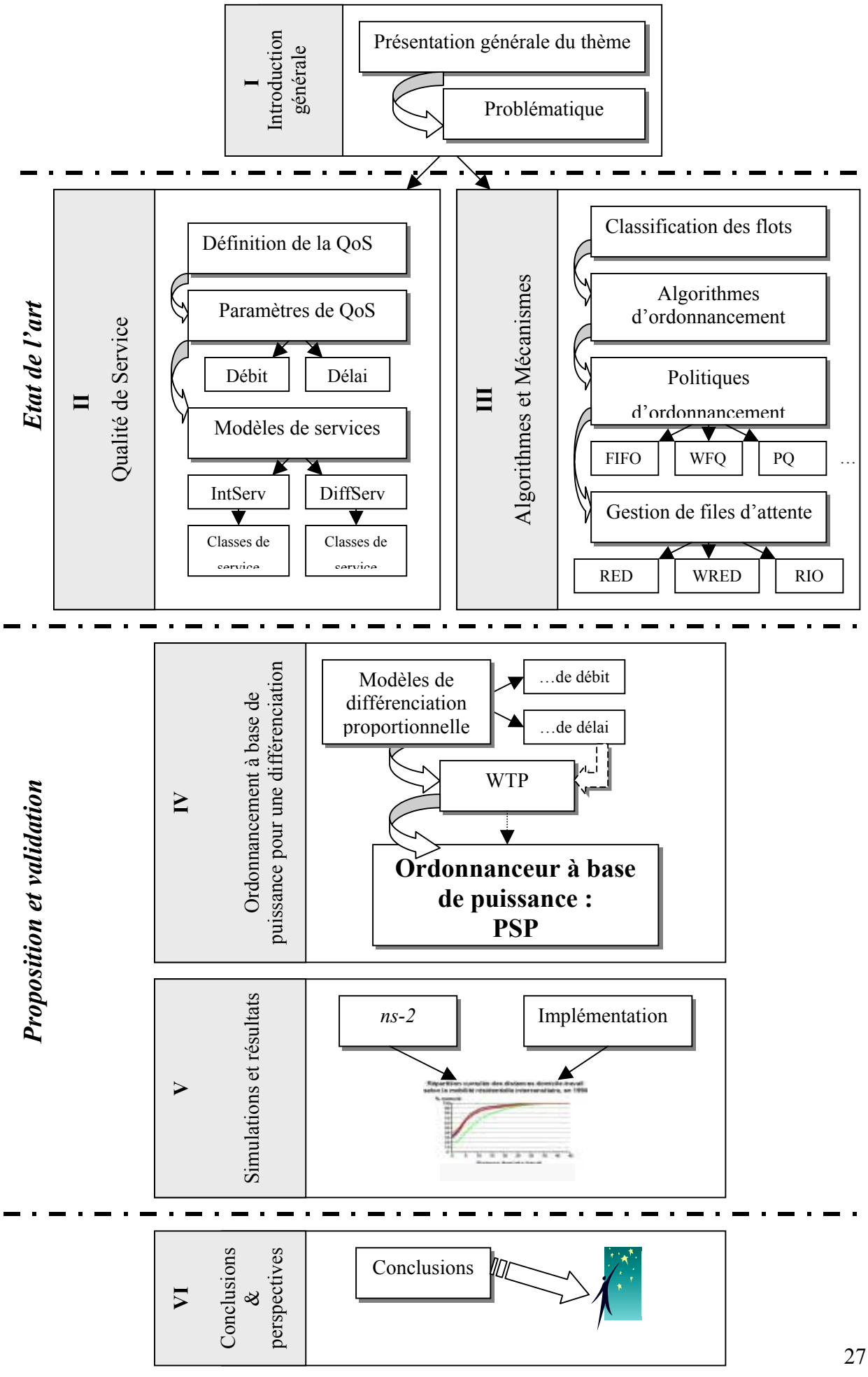
Dans le troisième chapitre, nous abordons la définition et la classification des flots circulant sur les réseaux. Nous définirons ainsi l'hétérogénéité, puisque notre travail s'est basé sur l'hétérogénéité des applications qui traversent tous les jours nos réseaux. Nous présentons par la suite les algorithmes d'ordonnancement dans leur contexte général, puis les mécanismes d'ordonnancement existants, qui se basent sur la métrique du débit ou du délai pour acheminer les paquets.

La quatrième partie de ce document est dédiée à la présentation de notre travail. Nous définissons pour cela le modèle de différenciation proportionnelle qui est la base de notre axe de recherches. En effet, tandis que nous visons d'obtenir une équité de partage des ressources, nous devons introduire la notion de proportionnalité dans la théorie de notre approche. Nous décrivons ensuite l'algorithme d'ordonnancement que nous avons développé, le « *Power as a Scheduling Parameter scheduler* ». Enfin, nous fournissons une topologie de réseau adoptant ce mécanisme, en mettant en valeur les divers modules qui interagissent avec l'algorithme.

La dernière partie est consacrée à l'implémentation de l'algorithme ainsi qu'aux diverses simulations réalisées pour tester et valider ce nouveau mécanisme d'ordonnancement. Pour ce faire, nous avons brièvement présenté la plate-forme de test, en l'occurrence l'outil de simulation ns-2, puis avons exposé les différentes mesures en débit, délai (métriques dont nous tenons compte pour la réalisation de notre ordonnanceur) et puissance (résultant du rapport des deux caractéristiques précédentes). Pour chaque scénario et chaque graphique, nous avons montré et mis en valeur les résultats de notre approche par rapport à un mécanisme à différenciation proportionnelle, *Waiting Time Priority*, que nous avons choisi comme référence de comparaison.

Nous clôturons le document par une conclusion émanant des résultats que nous aurons montrés, puis nous proposons des perspectives de recherches vers d'autres axes, en se basant sur le prototype que nous avons développé.

Nous proposons de résumer l'organisation de cette thèse dans le schéma ci-contre.


I
Introduction générale
Présentation générale du thème
Problématique
Etat de l'art
II
Qualité de Service
Définition de la QoS
Paramètres de QoS
Débit
Délai
Modèles de services
IntServ
DiffServ
Classes de service
Classes de service
III
Algorithmes et Mécanismes
Classification des flots
Algorithmes d'ordonnancement
Politiques d'ordonnancement
FIFO
WFQ
PQ
…
Gestion de files d'attente
RED
WRED
RIO
Proposition et validation
IV
Ordonnancement à base de puissance pour une différenciation
Modèles de différenciation proportionnelle
…de débit
…de délai
WTP
Ordonnanceur à base de puissance : PSP
V
Simulations et résultats
ns-2
Implémentation
VI
Conclusions & perspectives
Conclusions

QUALITÉ DE SERVICE (QoS)

2. Qualité de service (QoS)

2.1 Introduction

Aux débuts de l'Internet, la préoccupation principale était de pouvoir acheminer des paquets d'une source vers une destination, indépendamment de leur temps de transit. Les trafics qui circulaient sur les réseaux n'étaient pas encore autant diversifiés pour rencontrer les problèmes que nous percevons aujourd'hui : encombrement du réseau, ralentissement des téléchargements, mauvaise qualité de l'information en cas de surcharge du réseau, etc… De nos jours, les flux traversent les réseaux du monde selon une échelle de diversification et une échelle temporelle aléatoires. Nous sommes donc de plus en plus exigeants en qualité de réception de l'information : la qualité de service intervient en conséquence, pour obtenir une satisfaction de réception. Cette dernière peut se traduire sous la forme de l'intégralité ou du temps de réception.
Le présent chapitre est consacré à la présentation de la qualité de service déployée dans l'Internet. Nous commençons par donner des définitions de la qualité de service, à la suite de quoi nous étudions les différents trafics qui traversent les réseaux pour obtenir une classification des flots. Nous exposons dans un troisième temps les mécanismes étudiés par les groupes de travail s'intéressant au domaine pour la gestion de la qualité de service. Puis nous concluons le chapitre par un récapitulatif et une étude critique des mécanismes proposés pour offrir la qualité de service dans les réseaux.

2.2 Définition de la Qualité de service

Selon la recommandation E.800 du CCITT, la qualité de service (*QoS pour Quality of Service*) correspond à « l'effet général de la performance d'un service qui détermine le degré de satisfaction d'un utilisateur du service ». Cette définition n'est que subjective et reflète la perception de la qualité de service observée par un utilisateur.
Plus techniquement, nous proposons une seconde définition de la qualité de service : la qualité de service constitue, pour un élément du réseau (une application, un hôte ou même un routeur), la capacité d'obtenir un certain niveau d'assurance de telle sorte que la fluidité du trafic et/ou les services requis soient au mieux satisfaits.
Enfin, une troisième définition consisterait à dire que la qualité de service correspond à tous les mécanismes d'un réseau qui permettent de partager équitablement et selon les besoins requis des applications, toutes les ressources offertes, de manière à offrir, autant que possible, à chaque utilisateur la qualité dont il a besoin. Généralement, cette qualité est axée sur le débit, le délai et la perte des paquets : la téléphonie par Internet a pour but de pouvoir converser en temps réel (*facteur du délai*) sans entre-coupures engendrées par des délais

supplémentaires; télécharger une application volumineuse ne demande pas plus que de disposer d'une assez large bande passante pour récupérer le fichier le plus vite possible (*facteur du débit*) ; les deux applications sont demandeuses (fermement ou plus souplement) en matière de réception de l'intégralité des paquets (*facteur de pertes*).

Pour un maximum de fiabilité, la qualité de service requiert la coopération de toutes les couches actives du réseau ainsi que celle de chaque élément du réseau, de bout en bout (figure 2.1). Des politiques de gestion différentes sont adoptées pour garantir de la qualité de service, selon que l'on se place au niveau des couches du modèle OSI, ou au niveau matériel du réseau (QoS gérée entre les hôtes et les routeurs ou entre les routeurs eux-mêmes).

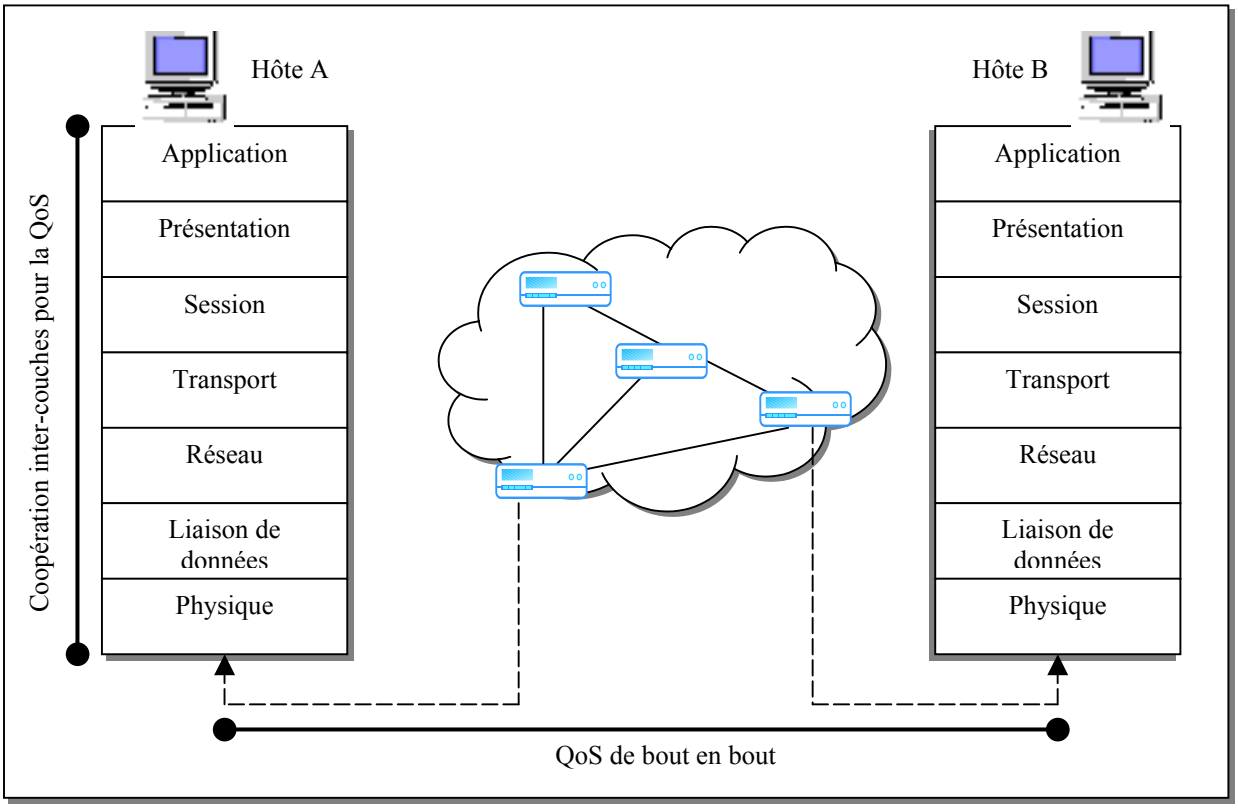


Figure 2.1 : Perception de la QoS dans les réseaux

Nous pouvons considérer la qualité de service comme un aspect tridimensionnel basé sur trois composantes : une composante « étendue », un modèle de contrôle et une garantie de transmission :

- par la composante « étendue » nous définissons les limites de services de qualité de service : par exemple, nous pouvons citer la réservation de ressources d'un flot par le protocole RSVP (*Resource Reservation Protocol*). La réservation s'effectuera entre les hôtes pour délivrer un niveau spécifié de qualité de service (nous présenterons ce protocole dans la section 2.4);
- le modèle de contrôle décrit la granularité, la durée et l'emplacement du contrôle des requêtes de qualité de service. Il est alors nécessaire de disposer d'un ensemble flexible de politiques, de pouvoir éviter ou empêcher des failles de qualité de service, etc. Nous pouvons citer à titre d'exemple la technique du contrôle d'admission des flots à l'intérieur d'un réseau, les mécanismes de gestion de files d'attente, etc. [90]

- la garantie de transmission est accentuée par la 'mesurabilité' qui consiste à pouvoir disposer de moyens permettant le contrôle des performances du réseau. La performance d'un réseau est évaluée selon le débit et délai de transmission, la largeur et la disponibilité de la bande passante offerte, le taux de pertes des paquets, etc.…

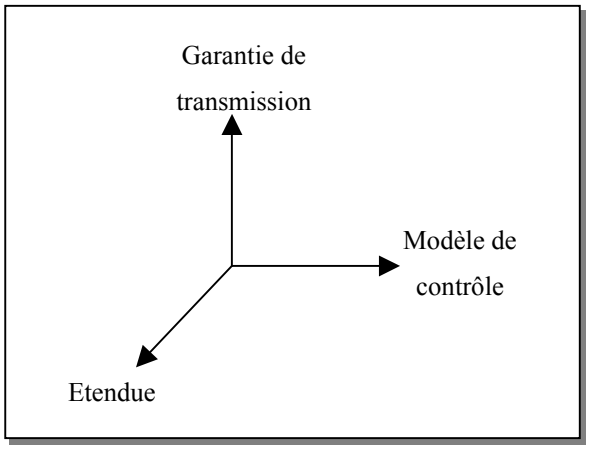


Figure 2.2 : Aspect tridimensionnel de la QoS

2.3 Paramètres de garantie de la qualité de service

La notion de qualité de service est, comme nous l'avons précédemment explicité, un aspect multidimensionnel basé sur des critères plus ou moins complexes à pouvoir garantir. Les principaux aspects connus de la qualité de service sont le délai, la gigue, le débit, la bande passante et la disponibilité (souvent exprimée en termes de taux d'erreurs).

2.3.1 Garanties de délai

L'information qui circule à l'intérieur d'un réseau est hétérogène, tant sur l'aspect de son flux, de sa nature ou de sa fréquence. En effet, les utilisateurs du réseau manipulent aussi bien des applications de transfert de fichiers que des applications multimédia. Contrairement à une opération simple du type de transfert de fichier, le domaine du multimédia requiert beaucoup plus de garanties en matière de qualité de service temporelle. Plus particulièrement, ces dernières applications sont sensibles au délai et à la gigue (variation du délai), mais aussi aux pertes d'information. Ainsi, la téléphonie par Internet, la vidéo-conférence, le multimédia interactif, etc… requièrent de strictes garanties en délai, en gigue et en taux de pertes. Citons à titre d'exemple le cas des jeux interactifs multimédia : les paquets de ces applications, qui subiront un délai de transit significatif ne seront plus correctement utilisés et détérioreront l'efficacité et la synchronisation de l'application. La perte des paquets aura un impact plus accentué sur la qualité du jeu puisque le son et la vidéo seront particulièrement dégradés.

Le terme « délai » englobe en réalité trois aspects temporels différents :

- le délai de propagation, déterminé par la distance physique qui sépare la source de la destination ;
- le délai de transmission dépendant de la taille des flots. Ce paramètre est aussi étroitement lié à l'utilisation du réseau et au partage de la bande passante disponible ;

- enfin, le délai d'attente et de traitement des paquets à l'intérieur des files d'attente, déterminé par la charge du réseau, ainsi que les politiques de traitement de l'information dans les routeurs pour obtenir une fluidité maximale de l'écoulement de l'information.

Garantir le délai implique la nécessité de mettre en œuvre des mécanismes permettant de gérer au mieux l'acheminement de l'information vers la destination en un temps minimal, tenant compte des trois natures de délais précédemment cités. Ainsi, pour minimiser le délai d'écoulement des flots de données, il est nécessaire que ces derniers qui transitent sur le réseau passent un temps négligeable, voire nul, au sein des routeurs. La configuration de ces derniers requiert donc une mise en œuvre de disciplines de services efficaces et adaptées aux besoins des applications pour leur assurer les garanties nécessaires en délais mais aussi en débit.
La gigue, résultant du paramètre « délai », correspond à la variation du délai d'acheminement de bout en bout. Des délais relativement importants éventuellement substitués par les traitements lents des routeurs nuisent automatiquement à la qualité de service par ce paramètre : des variations de délais apparaîtront et affecteront la qualité demandée.
Le taux moyen d'erreurs sur une liaison définit la disponibilité d'un réseau. L'efficacité d'un réseau dépend donc des erreurs qui surgissent sur les liaisons. Des taux d'erreurs minimes, voire nuls caractérisent un certain rendement et paramètrent une bonne qualité de service en matière de disponibilité du réseau. On associe souvent le taux d'erreurs au paramètre temporel, les erreurs affectant directement le transfert des flots, et retardant/bloquant ainsi leur arrivée à destination.
Les délais et les pertes sont les deux facteurs les plus connus qui nuisent aux garanties temporelles et qui engendrent l'amoindrissement des possibilités d'une application, voire rendent celle-ci totalement inefficace et inopérante.

2.3.2 Garanties de débit

Comme nous l'avons indiqué précédemment, les applications actuelles consomment de plus en plus de bande passante (figure 2.3), ce qui ralentit ou bloque le déroulement d'autres applications. De même, une utilisation massive du réseau (plusieurs flots provenant de plusieurs utilisateurs traversant le réseau au même instant) entraîne des conséquences de ralentissement de traversée des flots. La notion de bande passante d'un réseau intervient à ce niveau : un minimum de bande passante est requis pour assurer des garanties de qualité de service point à point, demandées à intervalles différents [11]. La capacité d'un réseau doit être suffisamment importante pour pouvoir laisser passer de l'information sans pour autant qu'il y ait de retard d'acheminement, ni de distorsion des flux d'origine en matière de pertes de paquets. C'est pourquoi nous portons davantage notre attention sur le débit de transfert sur le réseau. Ceci nous conduit à traiter les flots à l'intérieur d'un réseau en fonction du débit que chaque application cliente envisage de consommer. Pour cela, des mécanismes sont implémentés dans les routeurs pour contrôler le trafic et le lisser. Les techniques utilisées pour pratiquer le contrôle et le lissage du trafic seront mentionnées dans ce chapitre, et développées dans un prochain chapitre.

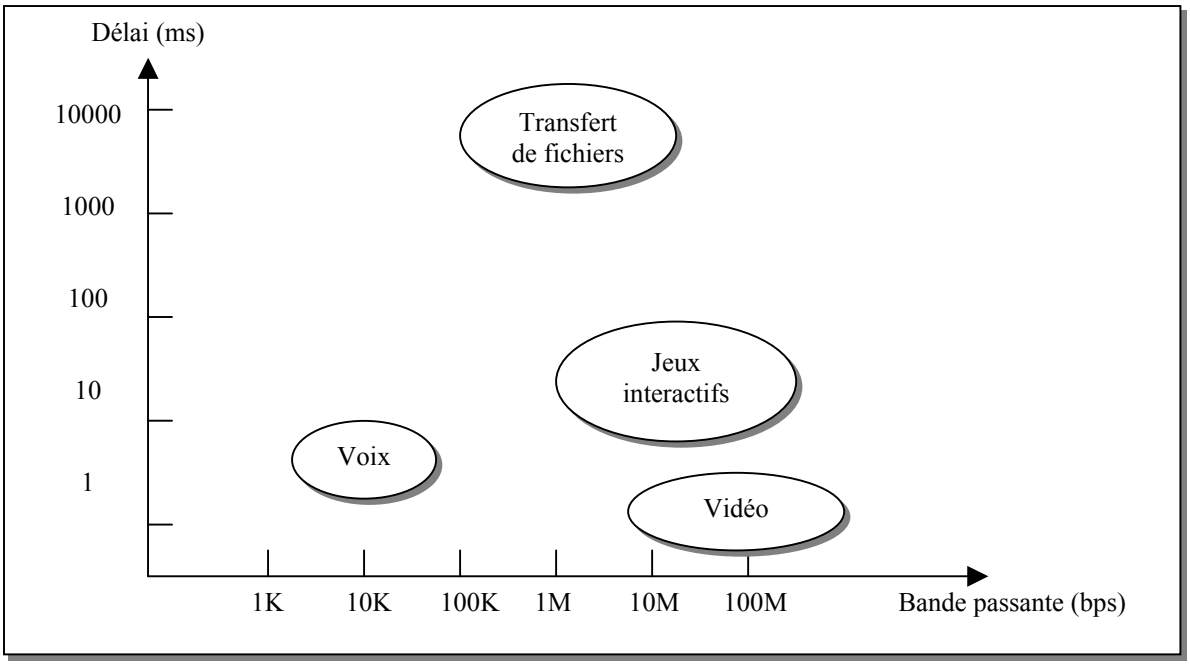


Figure 2.3 : Besoins en délai et bande passante des applications

2.4 Classification des flots

Lorsque nous désirons garantir de la qualité de service , il est nécessaire de pouvoir caractériser le trafic à l'intérieur des réseaux. La classification doit s'effectuer en fonction de la sensibilité des applications vis-à-vis de la qualité de service. Pour cela, [95] propose une approche où il s'agit d'attribuer une fonction d'utilité à chaque application. Il définit tout d'abord un vecteur contenant les mesures, telles que le délai, le débit, le taux de pertes, qui représentent le service livré à une application (ou à un utilisateur). La fonction d'utilité assure par la suite le tracé du service offert selon les performances de l'application. Cette fonction indique que les performances d'une application dépendent du service fourni. Ainsi, une augmentation (respectivement une régression) de la valeur de la fonction d'utilité exprime un accroissement (respectivement une dégradation) des performances de l'application.
La classification des applications s'effectue selon la nature des trafics eux-mêmes : en effet, les applications multimédia ont des caractéristiques et des besoins différents de ceux des simples applications d'échanges et de transfert de données. On distingue donc deux grandes catégories de flots et nous les classerons selon leur élasticité aux paramètres de qualité de service [91]. La première famille de flots adapte plutôt la génération de ses données en fonction des conditions du réseau : on les définira comme étant des flots élastiques, adaptables aux disponibilités du réseau. Ces applications qui peuvent supporter des variations de délais, peuvent être traités selon le principe du « best-effort ». Le second type d'applications génère les données indépendamment de l'état du réseau. En aucun cas les flots de ce type ne pratiquent le principe d'élasticité aux conditions du réseau. Ces applications multimédia sont classées dans la catégorie des applications temps-réel (*real-time applications*), non-élastiques. Cette famille d'applications se divise en deux catégories : les flots peuvent être fermes en garanties de délais et de pertes , auquel cas on parlera d'applications « hard real-time » ; ils peuvent, au contraire, être quelque peu tolérants, et on parlera donc de flots temps-réel adaptatifs [35].

2.4.1 Flots élastiques

Les flots élastiques sont connus pour leur tolérance aux délais et à leurs variations. Les applications qui génèrent du trafic élastique peuvent donc adapter le taux de génération de leurs paquets en fonction des conditions du réseau. Les applications telles que les transferts de fichiers, le courrier électronique, la connexion à un terminal distant, etc… sont classées dans la catégorie des flots élastiques. Cependant, malgré leur tolérance aux délais, certaines applications élastiques requièrent parfois de faibles délais d'acheminement pour pouvoir fonctionner correctement. Par exemple, les e-mails permettent une tolérance à des délais relativement importants, l'essentiel de ces applications étant qu'elles arrivent à destination, quelque soit le temps mis pour atteindre celle-ci. Inversement, des applications telles que « telnet » permettant de se connecter à un ordinateur distant, sont tolérantes aux délais, mais beaucoup plus exigeantes que l'exemple précédent puisqu'au bout de quelques secondes, si la connexion ne peut pas être établie, elle sera interrompue. Ce type d'applications est donc tolérant aux délais, mais jusqu'à une certaine limite ; cette dernière faisant défaut, ces applications d'échanges de données ne s'effectueront pas correctement. La garantie d'un minimum de bande-passante pour les applications élastiques favorise l'obtention d'une certaine garantie en matière de délai.
Par ailleurs, les transmissions de trafic de données doivent être au maximum fiables. Pour cela, les pertes ne doivent pas être tolérées au niveau de la couche application. Un protocole du type TCP est alors nécessaire car il est capable de retransmettre les paquets égarés [85]. TCP est le protocole capable d'adapter le taux de transfert des données à la bande-passante qui est disponible, même en cas de congestion du réseau. La disponibilité d'un minimum de bande-passante favorise ainsi le fonctionnement correct d'applications de cette catégorie.

Selon [95], la fonction d'utilité des flots élastiques est représentée par la figure ci-dessous. On remarque bien l'utilité de disposer d'un minimum de bande-passante pour garantir une certaine efficacité d'utilisation des applications élastiques. A mesure que la bande-passante s'accroît, l'utilité de l'application augmente, d'où le facteur d'élasticité.

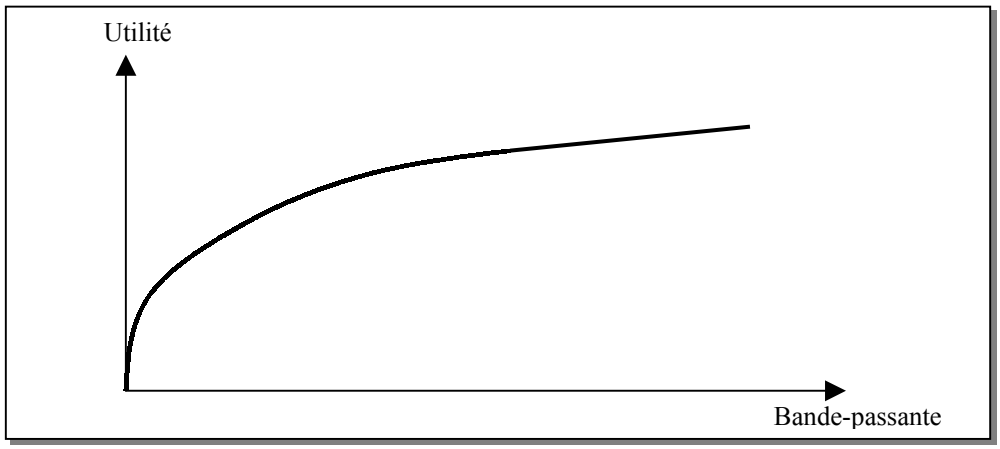


Figure 2.4 : Utilité d'une application élastique en fonction de la bande-passante

Les besoins de performance pour des applications élastiques sont donc plutôt orientés vers les débits (nécessité de bande-passante) et l'intolérance aux pertes. Par ailleurs, selon les applications, les délais sont plus ou moins importants pour garantir la qualité de service de ces flux.

2.4.2 Flots non-élastiques

Les flots non-élastiques sont caractérisés par la non adaptation de leurs trafics vis-à-vis des conditions du réseau sur lequel ils circulent. Pour ces applications, chaque paquet qui est émis doit arriver à destination avec un délai infinitésimal : c'est pourquoi on appelle aussi ces applications, des applications en temps-réel. La vidéo-conférence, la téléphonie sur Internet mais aussi les jeux en réseaux sont les exemples les plus répandus d'applications temps-réel. Ces dernières peuvent être plus ou moins exigeantes, selon leur capacité à supporter les intolérances. En effet, la téléphonie est beaucoup plus stricte en matière de délai et de pertes de paquets que la vidéo-conférence sur Internet. Il est plus difficile de comprendre une conversation téléphonique ayant subi des pertes d'information qu'une vidéo-conférence entrecoupée. C'est pourquoi au sein de cette famille de flots non-élastiques, nous distinguons deux catégories d'applications : les applications exigeantes (*hard real-time applications*) et les applications plus tolérantes (*adaptative real-time applications*).

2.4.2.1 Les applications fermes en temps-réel

Les besoins de ce type d'applications concernent les délais et la variation de ces délais (la gigue). Ces applications attendent des paquets qui constituent leurs flots, d'arriver en un temps minimal. Au-delà de la limite de temps qui est accordée pour l'arrivée des paquets, les performances de l'application se détériorent. Ces applications sont aussi fermes en matière de pertes de paquets : ce dernier facteur est à l'origine de la mauvaise qualité de déroulement de l'application.

La figure ci-dessous montre bien le caractère de fermeté de ces applications. On remarque parfaitement la transition brute qui existe entre la faible utilité (pour une bande-passante allant de l'état « faible » à l'état « seuil minimal ») et l'utilité maximale (pour une bande-passante allant de « seuil minimal » à « seuil maximal »).

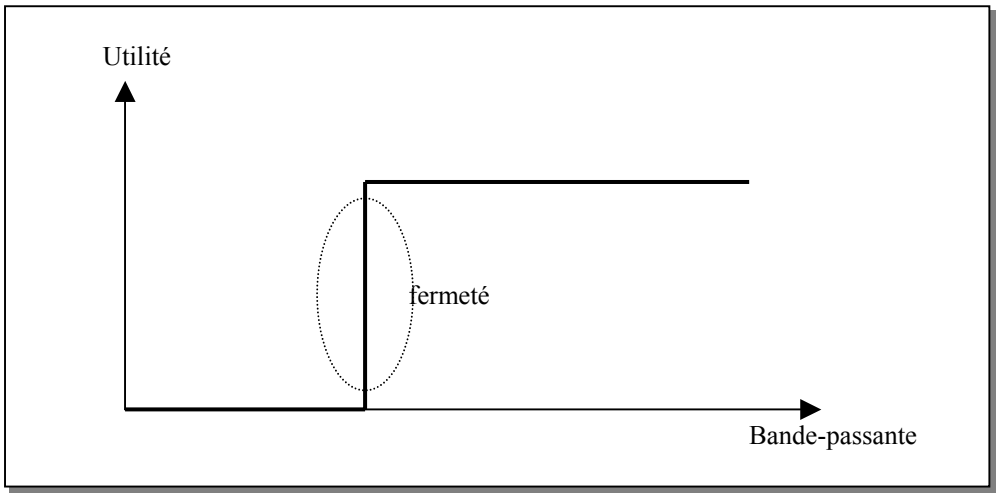


Figure 2.5 : Utilité d'une application temps-réel ferme en fonction de la bande-passante

2.4.2.2 Les applications temps-réel adaptatives

Les applications adaptatives de type temps-réel ne sont pas aussi restrictives en besoin que les applications fermes (*hard*). On peut aussi les classer en deux sous-catégories :

- Les applications temps-réel qui s'adaptent au délai (*delay-adaptative applications*) : ces flots sont quelque peu tolérants vis-à-vis de faibles délais et de pertes de paquets. La génération des flux est indépendante de l'état du réseau ; il est donc nécessaire que ces applications aient des besoins intrinsèques en bande-passante. Au-dessous de ces besoins, les performances sont remarquablement dégradées.

- Les applications temps-réel qui adaptent leur taux de transmission en fonction de l'état du réseau (*rate-adaptive applications*) : lorsqu'une congestion se produit à l'intérieur du réseau, la bande-passante dédiée diminue et les délais restent faibles. La performance de ces applications n'est donc pas liée aux délais (ceux-ci étant suffisamment faibles), mais plutôt au partage de la bande-passante. L'ajustement du taux de transmission peut s'effectuer selon deux cas de figure. En effet, dans un premier cas, la transmission peut être régulée à la source de manière explicite, en réponse aux conditions du réseau. La connaissance de l'état du réseau doit, par conséquent, pouvoir être gérée. La régulation du taux de transmission peut, d'autre part, être effectuée implicitement, en choisissant un niveau et une priorité d'élimination de certains paquets selon des politiques déterminées de telle sorte que ni le réseau, ni l'information, ne soient brutalement affectés. Dans ce cas, on permet donc une faible tolérance de pertes de paquets.

Les applications temps-réel adaptatives ont des besoins de bande-passante intrinsèques pour pouvoir fonctionner correctement. Les performances sous l'utilisation d'une large bande-passante sont indiscutables : les qualités de l'information reçue sont donc celles attendues. De faibles bande-passantes impliquent une nette dégradation de la qualité.
Pour les applications de cette catégorie, la fonction d'utilité donne l'allure présentée à la figure 2.6. L'état transitoire ne s'effectue pas brusquement comme il était le cas pour les applications fermes. Dans ce cas, la transition se fait progressivement, ce qui montre le caractère adaptatif pour une certaine étendue de bande-passante.

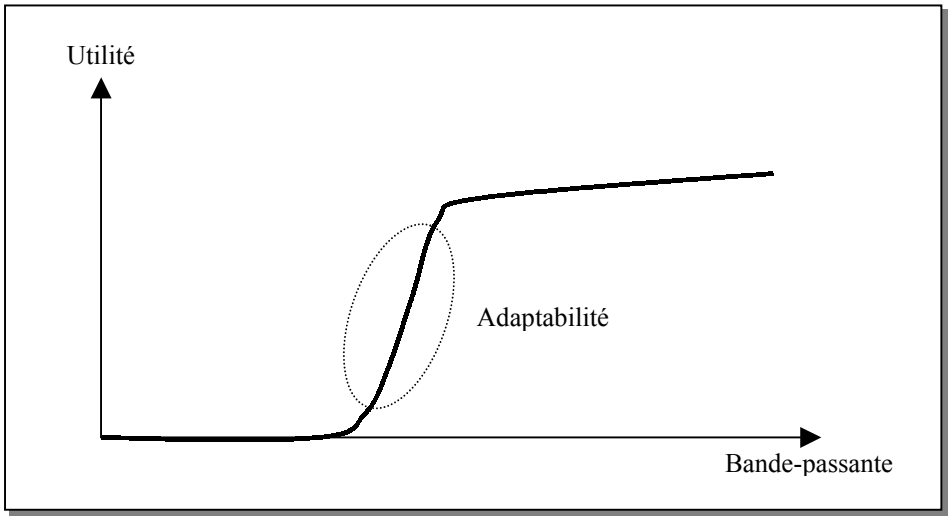


Figure 2.6 : Utilité d'une application temps-réel adaptative en fonction de la bande-passante

2.5 Routeurs et qualité de service

Tous les réseaux sont structurés selon un ensemble de terminaux, de routeurs et de liaisons de transmission. Les terminaux ne devraient pas posséder d'impact négatif sur la qualité de

service des données transmises. Cependant, les liaisons et les routeurs sont entièrement impliqués dans la qualité avec laquelle un flux peut être transmis. En effet, les liens (qu'ils soient filaires ou radios), possèdent des caractéristiques de délai, de débit et de disponibilité variables. De même, les routeurs ont un impact significatif sur ces caractéristiques. En effet, les fonctions d'un routeur consistent à contrôler l'intégrité d'un paquet reçu sur leur interface d'entrée, puis déterminer l'interface de sortie correspondante en fonction de la destination souhaitée, et enfin stocker le paquet dans la file d'attente appropriée. Lorsque le fonctionnement se dégrade (surcharge du réseau, par exemple), les files d'attente se remplissent, ce qui introduit un délai supplémentaire dans la transmission des paquets. Selon la stratégie adoptée par la file d'attente, on peut assister à une élimination des paquets en sus par le routeur qui s'efforce de réagir de la sorte pour éviter certains cas de figures (fortes congestions qui peuvent rendre le réseau inexploitable, les pertes de paquets des flots qui circulent, etc…). Or, toutes ces actions ont un impact sur les paramètres de qualité de service. C'est pourquoi des politiques de contrôle et de gestion du trafic sont implémentées au cœur des routeurs. Par ces mécanismes, il est question de limiter et/ou de prévenir le phénomène de congestion que nous développerons ultérieurement.

Jusque-là, nous avons présenté les aspects de base de la qualité de service. Dans ce qui suit, nous nous intéressons à la manière d'implémenter cela à l'intérieur d'un réseau pour bénéficier au mieux d'une qualité avoisinant celle que l'on demande. Le paragraphe suivant expose les modèles soulevés par les groupes de recherche de l'Internet qui se penchent sur l'aspect qualité de service.

Au départ, l'utilisation d'Internet était essentiellement fondée sur le modèle du « best-effort » qui consiste à traiter tous les paquets de données selon une politique identique et pour lequel le réseau tente de délivrer au mieux les données (d'où l'appellation « best-effort »). Or, ce modèle est rapidement devenu insuffisant pour les données multimédia (flux de type temps-réel) qui requièrent des garanties de délais et une bande passante bien explicite.
Des groupes de recherche de l'IETF (*Internet Engineering Task Force*) se sont penchés sur l'étude de politiques de services visant à classer les flux de données selon les besoins. Tout d'abord, il est apparu le groupe des services intégrés, Integrated Services *(IntServ*), puis, plus récemment, un second groupe pour le développement de services différenciés : Differentiated Services (*DiffServ*).

2.6 Le modèle à intégration de services : IntServ

2.6.1 Définition

Le groupe de travail « Integrated Services » a été développé en 1994 lorsque le multimédia est apparu et est devenu un domaine à part entière de l'Internet. Conçu à la même époque qu'ATM, le système avait pour objectif d'offrir un service semblable sur la couche réseau d'Internet. Le but était donc de pouvoir « améliorer » le réseau Internet et d'en faire un réseau à intégration de services. Il a fallu en effet pouvoir différencier l'utilisation de la bande-passante disponible entre les flots multimédia et les flots de données [54]. Les besoins requis par chacun de ces types de flots n'étant pas les mêmes, le groupe de l'IntServ a défini un ensemble de classes de services qui, implémentées au sein des routeurs, devraient allouer aux flots une certaine qualité de service, à chaque traversée des routeurs, pour les acheminer jusqu'à destination avec cette QoS [100].

Le principe du modèle IntServ repose sur deux fondements [14] :

- tout d'abord, le réseau doit être contrôlé et soumis aux mécanismes de contrôle d'admission des flux ;
- la nécessité de disposer de mécanismes de réservation de ressources pour obtenir différents services. Pour cela, l'émetteur envoie une requête de réservation de bande passante qui doit être acceptée par l'ensemble des équipements qui seront traversés par les flux. Le protocole utilisé est le RSVP (*Ressource reSerVation Protocol*) [15,5,9,73,16,101] (nous le détaillerons ultérieurement).

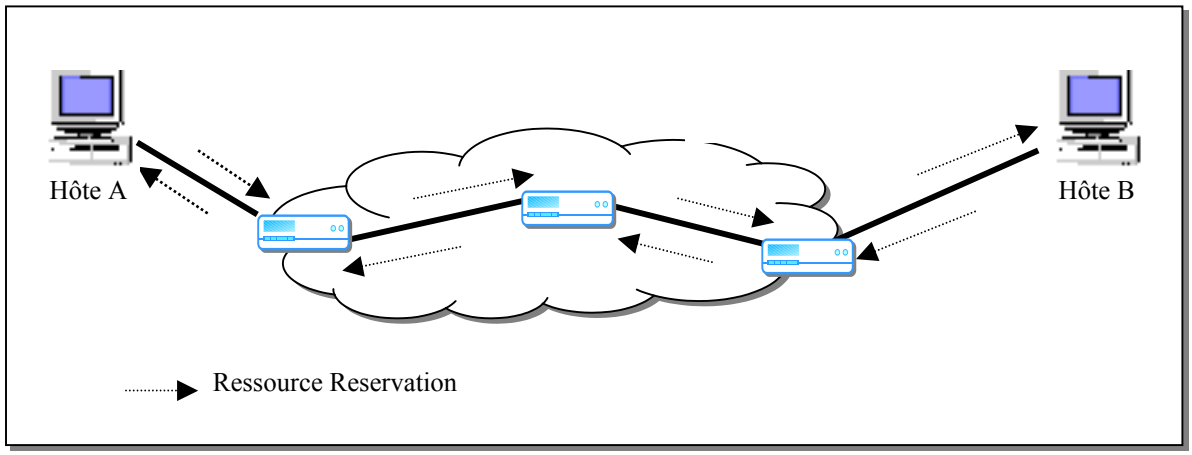
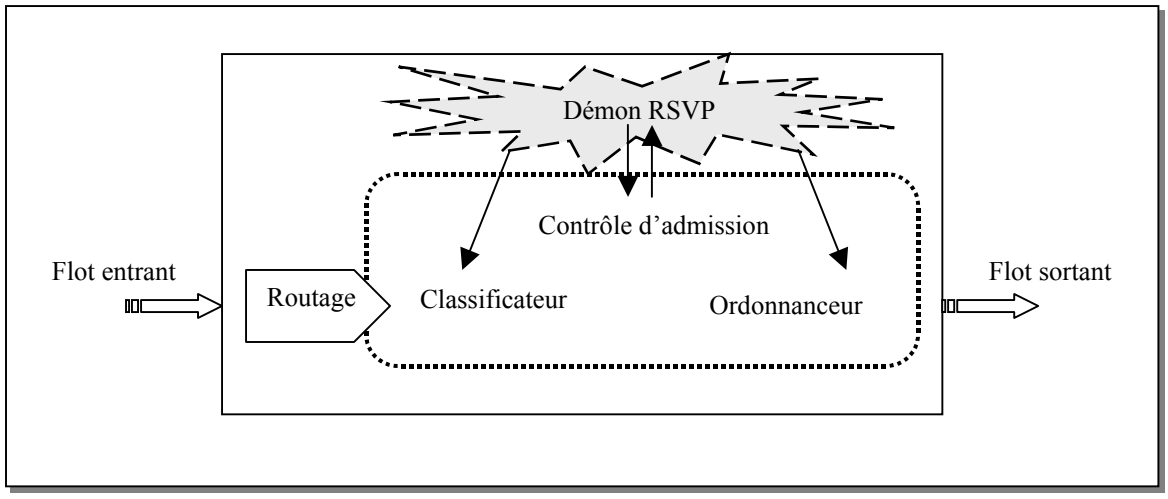


Figure 2.7 : Principe général du modèle à intégration de service

Les réseaux à intégration de services sont donc constitués de routeurs qui assurent les fonctionnalités de contrôle d'admission de flux et de réservation de ressources. Leur architecture est présentée dans la figure 2.8 :



Chaque routeur IntServ est ainsi constitué des éléments suivants :

- le classificateur (*classifier*) accueille les paquets en provenance du module de routage et détermine leur appartenance ainsi que leur type. Par exemple, le classificateur va devoir détecter si les paquets ont besoin ou non d'une réservation, s'ils sont ou non porteurs d'une réservation, etc…

- l'ordonnanceur (*scheduler*) reçoit les paquets du module précédent et gère leur retransmission en utilisant des files d'attente. Tous les paquets qui seront classés par le classificateur et appartenant à une même classe seront traités de la même manière par l'ordonnanceur. Chaque file d'attente implémente des algorithmes de gestion des paquets pour permettre un partage des ressources en fonction des besoins demandés par les utilisateurs. Les algorithmes peuvent par exemple utiliser le principe de partage équitable, le principe de gestion par classe, le principe de priorité, etc… Nous développerons les techniques utilisées par ces algorithmes dans le prochain chapitre ;
- le contrôleur d'admission (*admission controller*) vérifie s'il est capable de garantir la qualité de service requise par un flot et s'il y a suffisamment de ressources disponibles au moment de l'établissement d'une réservation ;
- un démon du protocole RSVP est en permanence en communication avec les différents composants du routeur. A partir de la requête formulée par une application, il fournit les informations au contrôle de trafic de chacun des routeurs qui appartiennent au flot et récupère la réponse du module de contrôle d'admission.

2.6.2 Caractérisation des flots

La spécification des flots permet d'expliciter la qualité de service requise par chacun des flots ainsi que les caractéristiques du trafic généré par ceux-ci.
Les besoins de qualité de service spécifiés par une application sont définis selon les paramètres suivants :

- la priorité : priorité qui sera attribuée au flot ;
- le débit : débit requis par l'application ;
- le délai : besoin en délai de bout en bout ;
- la perte : taux de perte autorisé par l'application.

Pour pouvoir utiliser les services définis pour les réseaux à intégration de services, le groupe IntServ a dû définir et déterminer une propriété pour caractériser les trafics d'un tel réseau. [94] décrit des paramètres de spécification de trafic contenus dans une variable de spécification TSpec (Traffic Specification). La caractérisation s'effectue selon le modèle représenté par la figure ci-dessous : le token bucket (« seau à jetons »).

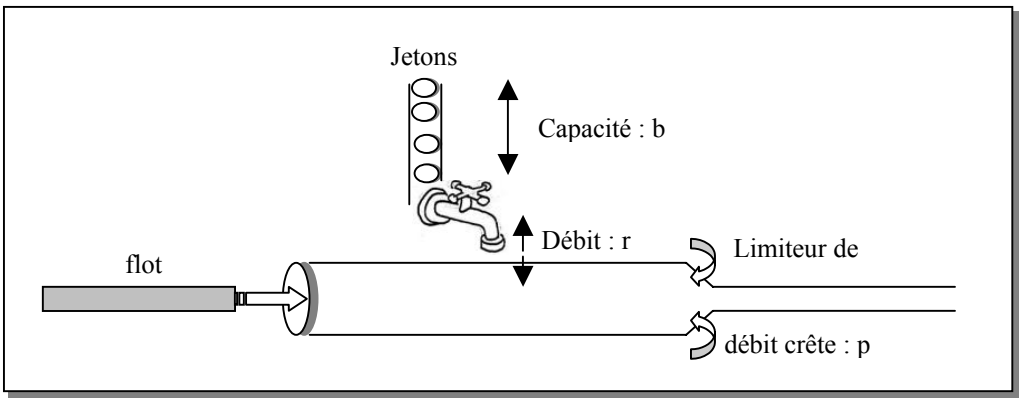


Figure 2.9 : Principe du Token Bucket

Le token bucket regroupe trois paramètres parmi cinq du TSpec. Tout d'abord, par sa capacité (paramètre b) et par le débit autorisé (paramètre r), il permet de contrôler le débit moyen du flot. Le paramètre p est utilisé pour limiter le débit crête. Les deux autres paramètres de

TSpec qui interviennent dans la spécification des flots sont « la taille maximale » du paquet contenu dans le flot (notée M), et « la plus petite unité de traitement» (notée m). Ces deux paramètres ne caractérisent les flots que par rapport à l'implémentation des mécanismes de qualité de service. Ainsi, on ne pourra garantir une certaine qualité de service qu'aux paquets du flot n'excédant pas la taille maximale définie dans TSpec. Le paramètre « m » indique que tous les paquets dont la taille lui sera inférieure, seront tout de même traités comme un paquet ayant cette taille « m ».

Paramètre	Description
b	Capacité du « seau »
r	Débit moyen délivré par le « seau »
p	Débit crête
M	Taille maximale du paquet
m	Taille minimale de « traitement »

Tableau 2.1 : Paramètres de spécification d'un flot

2.6.3 Classes de service

Le groupe de travail IntServ a rendu l'Internet un réseau à intégration de services en distinguant deux classes de services supplémentaires par rapport au service traditionnel du « best-effort ». Ces nouveaux services sont la classe à charge contrôlée, mieux connue sous le nom « controlled load service », où les performances reçues sont celles d'un réseau peu chargé ; et la classe de service garanti, ou encore « guaranteed service », où l'application qui demande le service possède l'assurance que les performances du réseau vont rester celles dont elle a besoin.

2.6.3.1 Le service best-effort

Le service dit « au-mieux » (*Best-Effort, ou BE*) ne garantit aucun critère de qualité de service : ni le délai de transmission, ni l'absence de pertes de paquets, ni l'absence de gigue ne sont considérés pour acheminer les flots de diverses natures. Ce service n'est évidemment pas approprié pour les flux multimédia qui transportent des informations à délivrer en temps réel. Il peut toutefois servir pour le transport de données. La messagerie électronique serait l'application la moins sensible à tous ces critères et supporterait donc sans trop de contraintes, le service du best-effort.

2.6.3.2 Le service « controlled-load »

Le service de charge contrôlée (*Controlled-Load, ou CL*) effectue une différenciation entre les trafics et leur attribue des codes de priorité en fonction de la sensibilité des applications [102] Bien évidemment, ce service est plus adéquat que le *best-effort* pour les applications multimédia plutôt adaptatives (décrites dans le chapitre 3) très sensibles à la congestion dans le réseau. Il offre un service proche de celui présenté par le best-effort lorsque celui-ci se

trouve particulièrement dans des conditions de réseaux non congestionnés. La garantie est donc fournie pour le débit. Mais contrairement au best-effort, le service de charge contrôlée ne détériore pas la qualité du flot lorsque le réseau est surchargé. En effet, les applications qui demandent ce type de service doivent tenir informé le réseau du trafic qui va le traverser, de manière à obtenir une meilleure exploitation du service et du réseau lui-même. Néanmoins, le réseau ne promet pas de garanties temporelles.

La variable de spécification de trafic (TSpec) est nécessaire pour identifier la conformité des paquets par rapport au service. La formule suivante nous montre les besoins auxquels doivent se plier les flots pour bénéficier du service « controlled load ».

$$Durée\_burst \leq r.M + b$$

La durée d'un burst correspond à la durée nécessaire pour transmettre la taille maximale d'un burst au débit demandé. Si sa durée vérifie l'équation ci-dessus, alors le flot est pris en charge par le service à charge contrôlée ; sinon, le trafic est traité comme du best-effort, ou peut être éliminé.

2.6.3.3 Le service garanti

Dans [9], les auteurs définissent le service garanti (*Guaranteed Service, ou GS*) comme étant « un service qui doit assurer de fermes garanties, de telle sorte que le délai de bout en bout mesuré sur les paquets d'un flot n'excède pas une certaine limite ». La classe de service garanti permet de même une garantie de bande-passante. Le service garanti livre les données (applications) en fonction des paramètres négociés et de la classe de service demandée. Le service est défini selon deux paramètres :

- TSpec qui caractérise le trafic pour lequel nous demandons le service garanti ; nous avons précédemment défini les paramètres de TSpec ;
- RSpec qui permet de spécifier les ressources à réserver. RSpec est défini au moyen de deux éléments :
    - R (Requested Rate), qui spécifie un taux de service désiré (exprimé en octets/s) pour lequel le trafic sera envoyé. Dans [93] l'auteur démontre que pour réduire le temps d'attente au niveau de la file, on doit avoir R > r (r étant le paramètre de TSpec identifiant le débit moyen du token bucket)
    - S (Slack term) qui définit la différence entre le délai attendu et le délai obtenu. Cette valeur, exprimée en micro-secondes, permet au réseau d'ajuster le taux alloué pour obtenir les délais requis. Elle peut être utilisée par une entité du réseau pour réduire la réservation locale de ressources d'un flot. Dans certains cas, la présence d'une valeur dans ce champ peut augmenter la probabilité d'effectuer avec succès une réservation de bout en bout. [25]

La classe de service garanti permet ainsi d'apporter aux applications un contrôle considérable du point de vue délai. Le délai d'une application se subdivisant en plusieurs sous-délais, seul le délai d'attente est déterminé par le service garanti , grâce au paramètre R de RSpec (les autres délais sont plutôt liés aux propriétés physiques du réseau). De ce fait, une application peut précisément estimer à l'avance quel délai de mise en attente le service garanti peut offrir. Par conséquent, si le délai d'une application est supérieur à celui attendu, celle-ci peut modifier le token-bucket du trafic ainsi que le taux de données pour obtenir un plus faible délai.

Pour les performances qu'il offre, ce service est particulièrement adapté aux applications temps-réel non adaptatives ayant des exigences de qualité de service très strictes (chapitre 3).

2.6.4 Le protocole RSVP

Le protocole de réservation de ressources (RSVP) est, de nos jours, le seul protocole utilisé par les réseaux intégrant IntServ. Il a été développé en 1993 [104] puis amélioré pour atteindre les objectifs de flexibilité et de capacité à gérer des sessions entre un nombre important d'utilisateurs [15]. RSVP identifie une session par les éléments suivants : adresse de destination, le type de protocole utilisé par la couche transport et le numéro de port de la destination.

RSVP_session = <dest_address, protocol_transport-layer, dest_port_number>

Fonctionnant au niveau de la quatrième couche du modèle OSI, la couche transport, RSVP est caractérisé par les éléments suivants :

- l'hétérogénéité du protocole : des récepteurs qui reçoivent un même flot peuvent recevoir une qualité de service différente ;
- le protocole est orienté récepteur dans le sens où c'est le récepteur qui choisit la qualité de service à fournir au flot ;
- RSVP est dynamique puisque les réservations peuvent être renégociées à tout instant (cas de changement de routage, par exemple) ;
- les états de RSVP sont « relâchés » (soft-state) : l'état qui est maintenu dans les routeurs au passage des messages doit être constamment rafraîchi par réémission des messages. Cette caractéristique permet à de nouveaux participants de se joindre à une session de façon dynamique.

Par ailleurs, le protocole de réservation de ressources est basé sur l'utilisation de deux types de messages unidirectionnels :

- le message PATH qui permet d'établir un état dans chaque nœud du réseau qu'il traverse et qui trace le chemin à suivre par le second type de message;
- le message RESV émis des récepteurs vers les émetteurs et qui consiste à réaliser la réservation proprement dite.

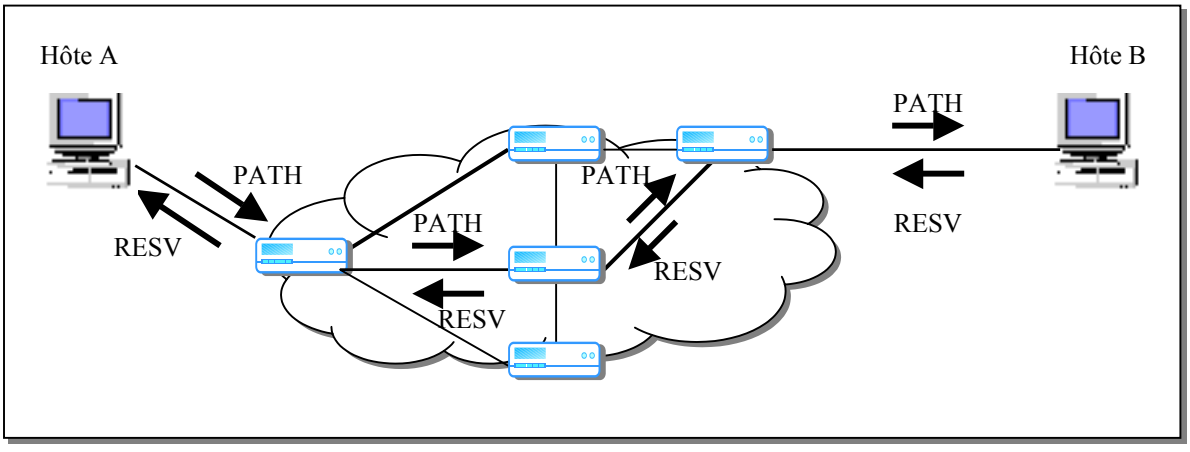


Figure 2.10 : Messages PATH et RESV pour la réservation de ressources

La figure 2.10 montre l'aspect unidirectionnel des messages PATH et RESV et illustre globalement l'algorithme selon lequel fonctionne le protocole RSVP :

1. Une application qui désire obtenir une certaine qualité de service va effectuer une demande de réservation de ressources. Pour cela, elle envoie des messages PATH qui contiennent la spécification du flot (paramètre TSpec) qui seront acheminés jusqu'au(x) destinataire(s).
2. En traversant chaque routeur du circuit, les messages PATH tracent une route créant un « path-state » contenant l'adresse des routeurs précédents. De cette manière, un chemin inverse est créé, et sera exploité au retour par le message RESV.
3. Chaque routeur ajoute des informations au message PATH. Ces dernières sont contenues dans les objets ADSPEC (Advertising Specification) [101] et révèlent les qualités de service que le routeur est capable d'attribuer. Ce mécanisme est défini dans [15] par OPWA (One Pass With Advertising) puisqu'il s'agit à chaque passage de routeur, de collecter des informations supplémentaires nécessaire à l'attribution de la qualité de service.
4. Le routeur IntServ fait appel au processus de routage pour déterminer le lien de sortie sur lequel sera envoyé le message pour atteindre le nœud suivant. Dès lors qu'un message PATH est reçu, on connaît les caractéristiques du flot ainsi que le chemin qui a été emprunté.
5. Pour obtenir la qualité de service requise, l'application envoie en, retour, un message RESV qui atteindra l'émetteur en parcourant le même chemin emprunté par les messages PATH, et ce, grâce aux « path states ». Le message RESV contient le TSpec caractérisant le trafic pour lequel le récepteur veut réserver des ressources, et le RSpec spécifiant les ressources à réserver.
6. Par le contrôle d'admission (figure 2.8), chaque nœud vérifie s'il y a suffisamment de ressources disponibles pour attribuer le service demandé.
7. Un retour positif du contrôleur d'admission crée un état « Resv State » qui contient des informations relatives à la réservation à réaliser, et le message RESV est envoyé vers le nœud suivant.

L'algorithme de RSVP semble être bien complexe puisque chaque routeur doit subir plusieurs opérations ; or les réseaux étant constitués d'un nombre important de nœuds, il est difficile de concevoir l'utilisation de RSVP au sein de réseaux de grande envergure, et à forte charge. La difficulté est notamment due aux communications générées par le protocole entre les équipements, mais aussi au traitement qui est effectué par flots plutôt que par agrégats. Par conséquent, il est incontestable de penser qu'un tel modèle est difficile à déployer à grande échelle (problème de « passage à l'échelle » du protocole), et son utilité reste donc restreinte à des niveaux de réseaux locaux de faible étendue.

Les recherches menées sur le passage à l'échelle se sont tout de même poursuivies et ont abouti à une amélioration de RSVP, donnant naissance à RSVP+ [42] et à RSPV-Switching [28], mais ces nouvelles versions ne sont pas exploitées, puisque les améliorations apportées restent toujours quelque peu limitées.

Cependant, RSVP n'est pas nécessairement le seul protocole que peut utiliser IntServ pour la réservation de ressources, mais il est le seul utilisé par le groupe de l'IETF. Un autre protocole a été proposé dans YESSIR [84]. Fonctionnant au-dessus de RTCP, il accomplit la fonction de réservation de ressources, orienté émetteur [92] , mais simplifie le processus de

réservation et allège la surcharge des routeurs. Néanmoins YESSIR héritant des fonctions de RSVP, effectue des réservation par flot et reste tout de même limité puisqu'il ne traite pas la réservation par agrégats, mais fournit uniquement des réservations de ressources pour des utilisateurs finaux (end users).

2.7 Le modèle à différenciation de services : DiffServ

2.7.1 Principe du service différencié

La difficulté de déployer IntServ sur Internet tient dans sa considération de la qualité de service au niveau du micro-flux. La gestion de la réservation de ressources à ce niveau demande une immense capacité de traitement. Ainsi, plutôt que d'utiliser la technique de réservation par session, un second groupe de l'IETF s'est orienté vers un autre modèle d'implémentation de qualité de service, que l'on peut utiliser pour des réseaux importants en envergure, mais aussi en charge : le modèle à différenciation de services. L'intérêt d'un tel modèle est de pouvoir s'occuper du problème d'approvisionnement en qualité de service à travers une allocation de services basée sur un contrat établi entre un fournisseur de services et un client. Pour le groupe de travail DiffServ, un micro-flux de paquet IP perd son identité propre et circule sur Internet en tant que membre d'une classe de flux. L'approche de DiffServ permet donc à des fournisseurs d'offrir différents niveaux de services à certaines classes de flots de trafic rassemblés. Ainsi, il devient question de supporter un schéma de classification en attribuant des priorités à des agrégats de trafic [69]. De ce fait, les paquets sont classés grâce à un mécanisme de marquage d'octets dans l'en-tête des paquets IP, et les services qui sont octroyés par les routeurs à ces paquets dépendent des classes alors définies [80]. Ce marquage est effectué au niveau de l'étiquette de l'en-tête du paquet : le DSCP (DiffServ Code Point) [79] qui se situe dans le champ DS de l'en-tête IP réservé à DiffServ. La figure 2.11 présente l'emplacement des champs DS et DSCP dans les en-têtes IPv4 et IPv6. Grâce à ce marquage, et à l'approche différente de DiffServ par rapport à IntServ, les routeurs DiffServ gardent une certaine souplesse d'utilisation du point de vue acheminement par rapport à ceux utilisés dans l'IntServ.

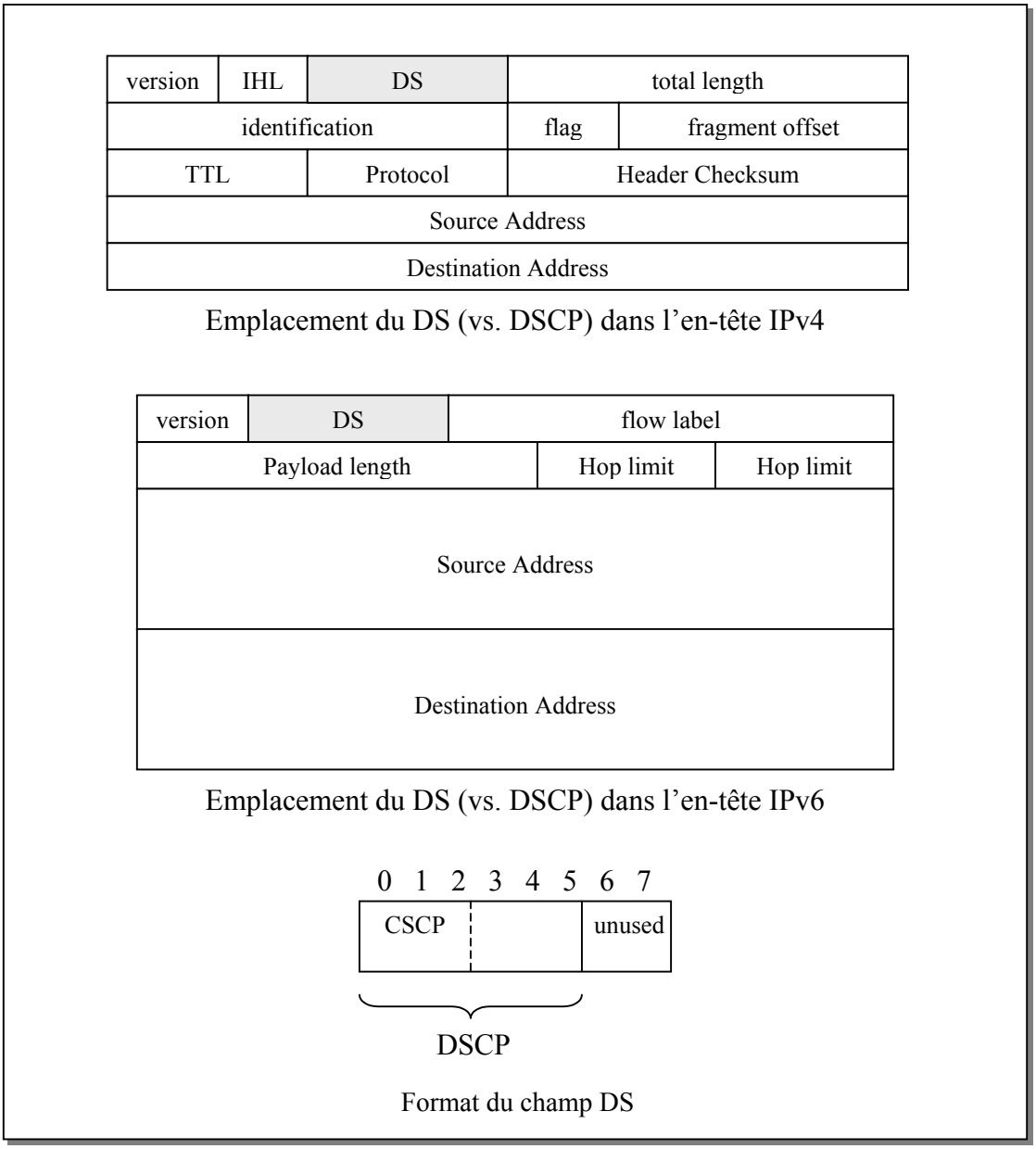


Figure 2.11 : Positionnement du champ DSCP dans les paquets IP

2.7.2 Architecture des routeurs DiffServ

Contrairement à l'architecture IntServ où les routeurs interagissent au moyen de réservations de ressources afin d'assurer des garanties de bout en bout, un réseau DiffServ est vu comme une interconnexion de routeurs obéissant chacun à une politique déterminée dans l'ordonnancement des paquets, mais agissant indépendamment les uns des autres. De ce fait, l'architecture adoptée par DiffServ est fondée sur deux principales catégories de routeurs : les routeurs de bordure (edge routers) et les routeurs de cœur (ou internes) (core routers), comme l'illustre la figure 2.12.

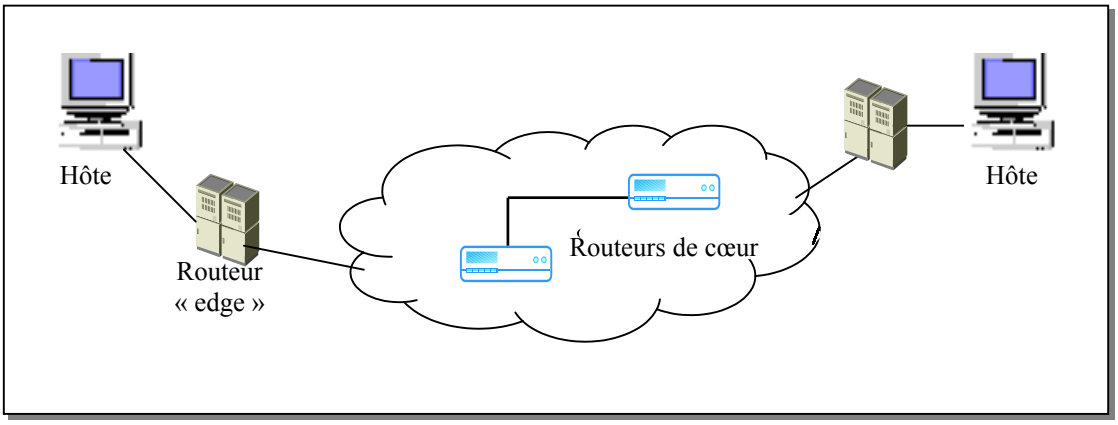


Figure 2.12 : Aspect général d'un réseau DiffServ

2.7.2.1 Les routeurs de bordure : les « edge routers »

Les routeurs « edge » sont responsables de la classification des paquets et du conditionnement du trafic. L'opération de classification est opérée à l'entrée du réseau, zone où la différenciation de service est mise en œuvre, le domaine DS, pour assurer le traitement ciblé : celui de pouvoir différencier les différents flots qui arrivent dans le réseau. Ces routeurs sont caractérisés par leur gestion des états par flots. Après classification, les paquets subissent une opération de vérification (module « meter ») qui consiste à déterminer le niveau de conformité pour chacun des paquets d'un même flot. Ces niveaux de conformité varient en fonction du conditionnement requis par le service. On peut définir par exemple deux niveaux « in » et « out » spécifiant si les paquets sont conformes ou non conformes avec le contrat établi. Cette dernière différenciation est néanmoins utilisée dans l'algorithme d'ordonnancement [22] que nous détaillerons dans le chapitre 3. L'étape qui suit la vérification du niveau de conformité est de deux types : si les paquets sont conformes, alors ils sont envoyés pour être étiquetés (nous présenterons cette technique ultérieurement). Dans le cas contraire, alors il y a trois manières de traiter les paquets non conformes : mise en forme (shape), marquage (mark) ou élimination (drop) :

- l'opération de mise en forme (shape) a pour but de rendre conforme les paquets qui ne le sont pas, et ce, par simple retardement de son acheminement. Après un certain temps de retardement, les paquets deviendront conformes et seront injectés vers le module d'étiquetage.
- le processus d'élimination (drop) est nécessaire pour assurer un fonctionnement fiable du routeur : tous les paquets qui ne sont pas conformes seront forcés à être éliminés. Les flots pour lesquels il serait préférable d'utiliser cette méthode sont les flots interactifs. En effet, si on choisit d'appliquer la mise en forme pour ce type de flots, les paquets subiront un retard significatif et rendront l'interactivité de l'application impossible. C'est pourquoi il leur est préférable d'être éliminés.
- enfin, le mécanisme de marquage (mark) a pour effet d'attribuer une priorité aux paquets en fonction du résultat fourni par l'opération de vérification. Par conséquent, en plus du premier niveau de différenciation, une seconde classification est effectuée par l'opération de marquage (nous verrons son utilisation au niveau des classes de service de DiffServ).

Avant d'être envoyés vers l'intérieur du réseau, les paquets subissent une dernière opération : l'étiquetage effectif du champ DSCP. La valeur attribuée au champ correspond au résultat de toutes les opérations précédentes. Celle-ci peut être modifiée au sein d'autres routeurs de bordure, selon le nouveau conditionnement qui va être appliqué au flot après sa traversée dans les équipements. La marque qu'un paquet reçoit identifie la classe de trafic à laquelle il appartient. Les paquets ainsi marqués sont alors envoyés dans le réseau avec cette mise en forme.

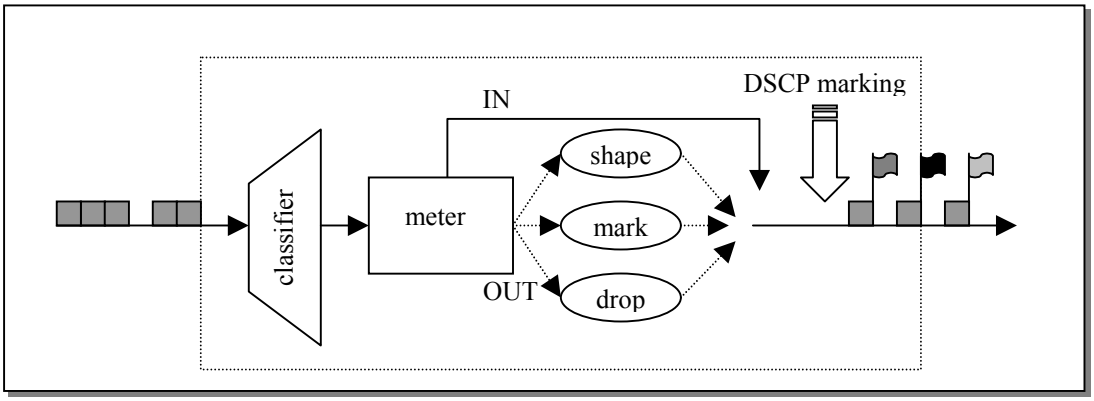


Figure 2.13 : Principe de fonctionnement d'un routeur « edge »

2.7.2.2 Les routeurs de coeur : les « core routers »

Les routeurs internes se chargent de la gestion des états par classe et traitent les paquets en fonction de la classe codée dans l'en-tête en les traitant en accord avec le comportement local correspondant : le Per-Hop-Behavior (PHB). Le service perçu de bout en bout pour un PHB dépend de la politique de d'approvisionnement du réseau pour chaque PHB et du conditionnement de trafic effectué au niveau des routeurs de bordure. Les PHB se basent uniquement sur le marquage de paquet, pour permettre aux routeurs d'identifier la classe de trafic à laquelle le paquet appartient ; en aucun cas ils ne traiteront différemment des paquets de sources différentes.

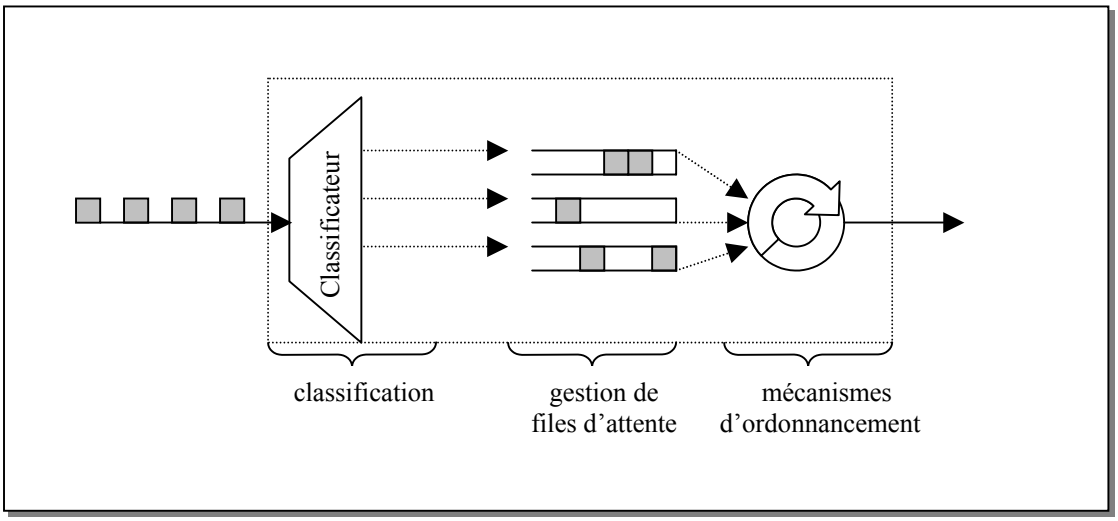


Figure 2.14 : Architecture d'un routeur interne

Comme le montre la figure 2.14, le classificateur envoie les paquets vers différentes files d'attente, selon l'identificateur placé dans le champ DS, préalablement marqué par les routeurs d'entrée du réseau (les routeurs « edge »). Seule une partie du champ permet d'indiquer la file d'attente appropriée : il s'agit du champ CSCP (Class Selector Code Points) (cf. figure 2.11) qui, constitué de 3 bits, permet de coder la classe de service correspondant à la catégorie du paquet. Ainsi, chaque file d'attente caractérise une classe de service et ne recevra que les paquets qui sont conformes à ce service. Au niveau des files d'attente, des mécanismes de gestion sont implémentés pour obtenir un maximum d'équité de traitement lors de congestions. Les mécanismes d'ordonnancement, dont nous définirons les techniques dans le chapitre 3, permettent de traiter les paquets en fonction de leur besoin en débit, délai, etc…

Un troisième type de routeur est mis en jeu dans des réseaux utilisant DiffServ. Il s'agit des routeurs de bordure (border router) placés en sortie des domaines DS pour relier deux réseaux DS. Leur fonction consiste à remarquer les paquets et de les remettre en forme pour les acheminer vers le sous-réseau suivant DS à travers les routeurs de bordure.

La figure 2.15 présente l'architecture d'un réseau à différenciation de services, et montre les différents types de routeurs utilisés pour bénéficier de ce type de services. Nous distinguons deux sous-réseaux DS utilisant chacun les principes de DiffServ. Ces deux sous-réseaux sont reliés entre eux par des routeurs de bordure ; les routeurs de cœur peuvent être reliés entre eux, ou à des routeurs de bordure ou encore à des routeurs « edge ». Enfin, les hôtes sont directement rattachés aux routeurs « edge ». Entre ces derniers et les domaines DS, des contrats de service appelé SLA (Service-Level Agreement) sont mis en œuvre pour spécifier les classes de service supportées et la quantité de trafic autorisée pour chaque classe [103]. Ces informations sont nécessaires à connaître si un client désire obtenir des services différenciés.

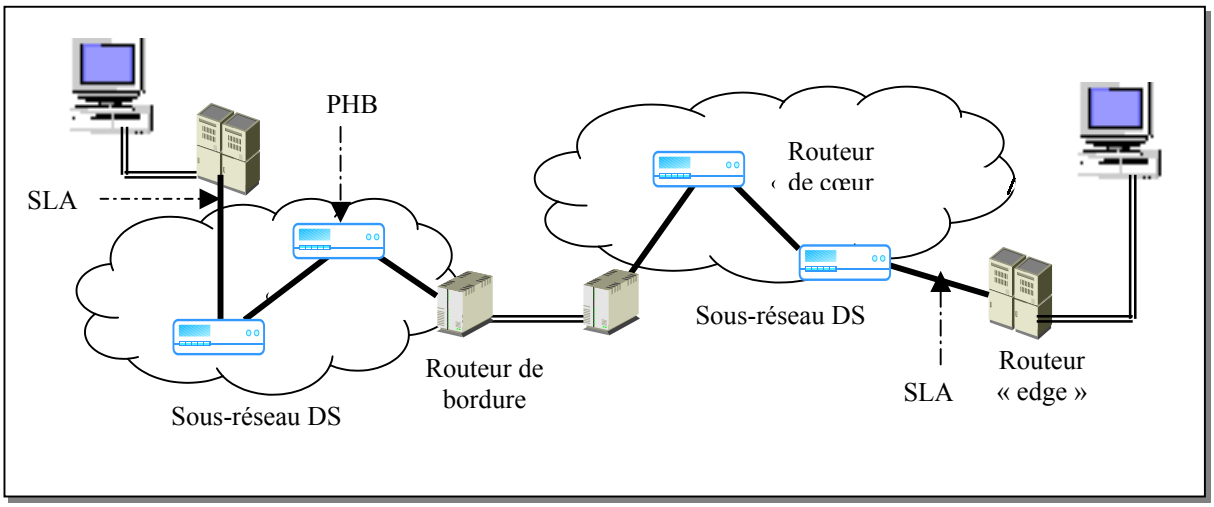


Figure 2.15 : Eléments constitutifs d'un réseau DiffServ

2.7.3 Classes de services de DiffServ

Dans cette approche, on dispose de trois niveaux de priorité : le service « Best-Effort », le service « Assured Forwarding », et le service « Expedited Forwarding ».

2.7.3.1 Le service « Best-Effort »

Le service Best-Effort (BE), que nous avons déjà décrit précédemment, correspond à la priorité la plus faible. C'est le service actuellement utilisé dans l'Internet et qui ne distingue pas les flots prioritaires des flots moins prioritaires.

2.7.3.2 Le service « Assured Forwarding »

Le comportement Assured Forwarding (AF) est le fruit de plusieurs propositions [66,23,80] qui centraient la différenciation des paquets sur des algorithmes de discrimination des paquets à l'intérieur d'une même file d'attente. La description du service AF dans [58] montre qu'il est plutôt dédié à des services généraux. Il englobe quatre classes de traitement et un second niveau de différenciation pour attribuer les priorités : la précédence. Cette dernière met en évidence l'importance d'un paquet et par conséquent la dégradation que peut engendrer sa perte sur la qualité de l'information. La notion de précédence peut être utilisée au sein des routeurs pour perfectionner l'opération de rejet des paquets en cas de surcharge et de congestion du réseau. Le service AF se trouve donc constitué de quatre classes de service définissant chacun trois niveaux de priorité. De ce fait, le service est composé de douze PHB interdépendants (6 bits) pour les douze PHB définis.

PHB_{ij}	Classe 1	Classe 2	Classe 3	Classe 4
Précédence 1	001 010	010 010	011 010	100 010
Précédence 2	001 100	010 100	011 100	100 100
Précédence 3	001 110	010 110	011 110	100 110

Tableau 2.2 : Codage des DSCP correspondants à AF

[58] impose la contrainte suivante : au sein d'une même classe, les paquets qui appartiennent à la précédence « x » sont prioritaires devant ceux ayant une précédence « y », et ce, pour $x < y$. Ainsi, pour la classe 1, les paquets de précédence 2 seront prioritaires par rapport aux paquets de précédence 3.

2.7.3.3 Le service « Expedited Forwarding »

Le service Expedited Forwarding (EF) [65] est un dérivé du service « premium » [80] conçu pour servir des applications demandant de faibles pertes, un délai et une gigue très faibles et une garantie de bande-passante. Le principe de base est de pouvoir garantir une taille des files d'attente dans les routeurs la plus restreinte possible. Pour assurer qu'un routeur puisse avoir un comportement (PHB) fonctionnant en EF, il doit assurer que le débit d'émission du trafic soit toujours supérieur ou égal à un certain seuil. Ce dernier peut être connu et configurable, auquel cas les mécanismes de gestion externes peuvent faire respecter la relation entre le débit d'entrée et le débit de sortie au niveau de tous les nœuds du domaine. Dans [65], les auteurs proposent, suite à une étude comparative de technique d'implémentation du PHB, l'utilisation d'une file prioritaire pour les flux typés EF. Le délai observé pour les flux à l'intérieur des files est fortement réduit, mais un problème de famine des autres classes pourrait se manifester si une mauvaise gestion est opérée dans le système. C'est pourquoi il est nécessaire d'intégrer des méthodes permettant de lisser le trafic EF. Le service EF est un service à forte priorité, délivrant des garanties en matière de bande passante, et permettant des taux de perte,

de délai et de gigue faibles [3]. On peut apparenter ce service à celui du service garanti du groupe IntServ.

101 110

Format du DSCP attribué à EF

Nous résumons dans le tableau ci-dessous les différents services selon leur priorité, puis dans la figure 2.16, nous donnons des exemples d'applications courantes en leur associant les classes de services de l'architecture DiffServ :

Service	Priorité
Best Effort	Faible
Assured Forwarding	Moyenne
Expedited Forwarding	Forte

Tableau 2.3 : Récapitulatif des priorités de services DiffServ

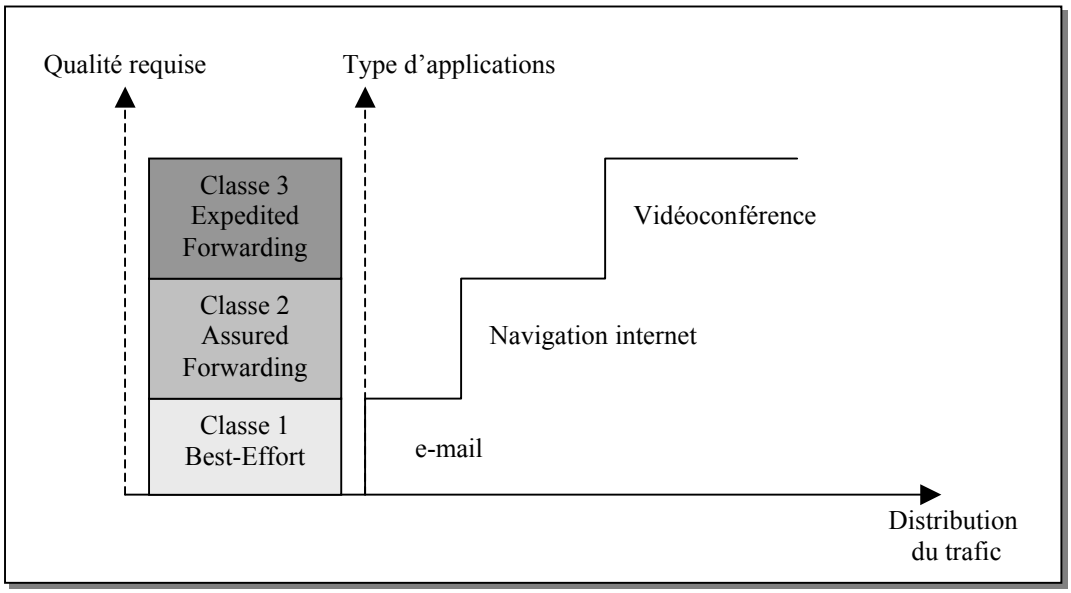


Figure 2.16 : Qualités requises pour des applications sous DiffServ

2.8 Conclusions

Nous avons présenté dans cette section les différentes classes d'applications couramment utilisées sur le réseau Internet. Chaque application est caractérisée par des besoins plus ou moins exigeants en terme de délai, débit et/ou gigue, selon sa nature. Ces critères ont permis aux membres de l'IETF d'effectuer une classification de priorité de l'acheminement de ces flots sur le réseau, tout d'abord lors de la définition des réseaux à intégration de service (IntServ), puis dans un second temps, après des recherches plus avancées, lors de la définition des réseaux à différenciation de service (DiffServ) [74,75]. Nous proposons de récapituler

toutes les notions préalablement exposées dans le tableau ci-dessous. Pour chaque type d'application, nous attribuons un niveau de débit, délai et gigue correspondants aux exigences de celle-ci. Puis nous exposons le type de service qu'il est conseillé (fortement) d'attribuer à chaque application, selon que l'on se place dans un réseau IntServ ou DiffServ.

Applications	Débit	Délai	Gigue	Service IntServ	Service DiffServ
Elastiques	++	=	=	Best Effort	Best Effort
Temps réel adaptatives	+++	+	+	Controlled Load	AF
Temps réel fermes	+++	+++	++	Guaranteed	EF

Tableau 2.4 : Besoins requis pour les différentes applications

La gestion de la qualité de service par IntServ et DiffServ présente tout de même quelques limites.
Tout d'abord, IntServ emploi un mécanisme de réservation de ressources, qui doit être implémentée au niveau de chaque routeur, ce qui conduit à une lourdeur de traitement. D'autre part, IntServ applique ses algorithmes à une échelle macroscopique des données : la gestion de QoS est par conséquent appliquée par flot. De ces limites découlent des critiques en raison de son incapacité à gérer la QoS pour des réseaux largement déployés : le problème de passage à l'échelle est un aspect important puisque l'envergure des réseaux ne cesse d'accroître, et le trafic qui les traverse est en perpétuelle augmentation.
L'approche DiffServ consiste, quant à elle, à remédier aux problèmes de passage à l'échelle et de complexité de gestion. Néanmoins, cette politique présente l'inconvénient de garantir une différenciation absolue, c'est-à-dire que plus une classe est grande, plus elle sera privilégiée pour le partage des ressources par rapport aux autres classes concurrentes. Ces mécanismes engendrent une discrimination de répartition et par conséquent peut provoquer le phénomène de famine entre les classes.

ALGORITHMES ET MÉCANISMES D'ORDONNANCEMENT

3. Algorithmes et mécanismes d'ordonnancement

3.1 Introduction

La littérature a souvent tendance à confondre les notions d'ordonnancement et celles de gestion de files d'attente. Or, ces deux concepts, bien qu'étroitement liés, sont néanmoins différents. En effet, les mécanismes d'ordonnancement ont pour fonction de répartir les ressources d'un même réseau entre plusieurs flots qui le traversent. La gestion de files d'attente, quant à elle, se préoccupe de la gestion d'acheminement, de retardement ou d'élimination des paquets appartenant à une même classe de flot. Dans ce chapitre, nous présentons tout d'abord le principe général de l'ordonnancement, puis nous spécifions les mécanismes existants les plus réputés. Nous exposons de même les politiques de gestion de files d'attente pour mettre en évidence la différence qui existe entre les termes « ordonnancement » et « gestion de files ».

3.2 Algorithmes d'ordonnancement

La question attribuée à l'ordonnancement concerne le contrôle de distribution des ressources entre les différentes classes de service. Pour cela, il est possible d'isoler des classes par rapport à d'autres, ou encore de réduire le temps d'attente des paquets à l'intérieur des files. Dans tous les cas, le principe repose sur une utilisation adéquate des files d'attente [68]. Les principales approches se basent sur des algorithmes utilisant une unique file d'attente, ou sur une utilisation multiple de ces files [7].

3.2.1 Algorithmes à files d'attente unique

Le principe de base utilisé pour l'ordonnancement au sein d'une seule file d'attente concerne une notion temporelle plus connue dans la littérature sous le nom d'estampille temporelle. Cette dernière a pour but de pouvoir déterminer le point d'insertion d'un paquet à l'intérieur de la file d'attente.
Pour pouvoir la définir, l'estampille nécessite la connaissance de trois facteurs :

- le poids attribué à la classe de service concernée : la notion de poids détermine le pourcentage de bande passante que la classe se voit attribuer.
- la longueur du paquet de cette classe de service ;
- l'interaction avec les autres classes actives concurrentes présentes sur le lien.

Nous présentons un exemple d'utilisation de l'estampille pour mieux comprendre le principe de fonctionnement des files qui exploitent cette datation.
Soient deux flots de données F1 et F2, constitués de paquets de longueur variable (dans notre cas, nous simplifions l'hypothèse d'avoir 5 et 10 octets). Nous noterons L^i_j la longueur du $i^{ème}$ paquet du flot j, et r_j le poids attribué au flot j. L'estampille d'un paquet i appartenant au flot j est dénotée E^i_j.
Nous représentons dans la figure 3.1 l'illustration de l'exemple que nous proposons.

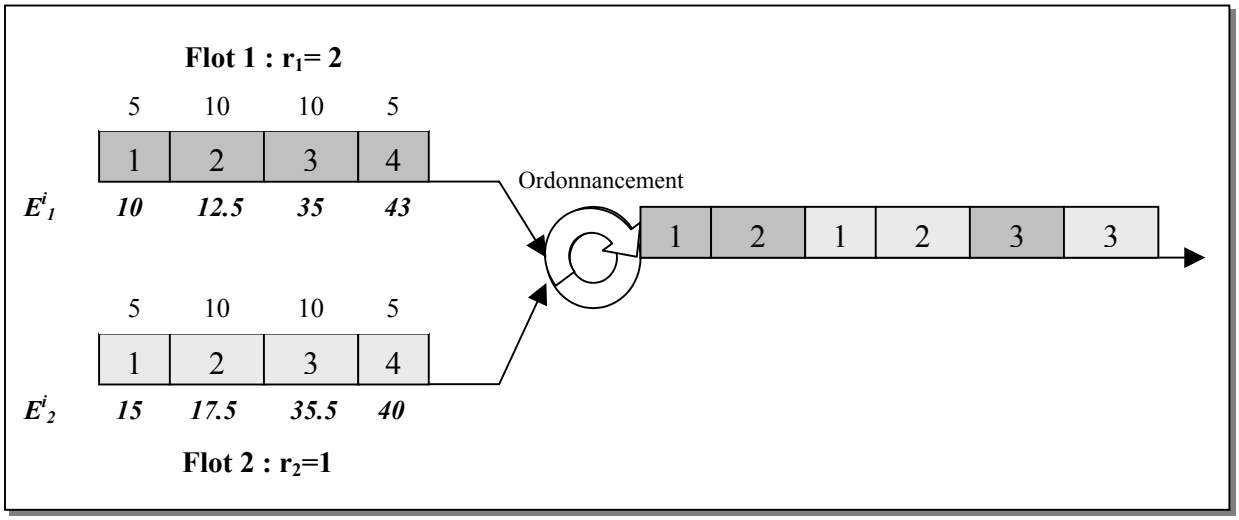


Figure 3.1 : Mécanismes d'ordonnancement par estampillage

Les estampilles sont calculées selon la formule :

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

En appliquant cette équation à notre exemple, avec l'hypothèse de la taille des paquets que nous avons figée, $L^i_1 = L^i_2$, on obtient des estampilles du flot 2 deux fois supérieures à celles du flot 1. La conséquence qui en découle est que les paquets du flot 1 sont transmis avant ceux du flot 2, et ainsi, le débit obtenu pour le flot 2 sera le double de celui réservé au flot 1.

La technique d'ordonnancement qui utilise directement l'estampillage est le *Weighted Fair Queueing* qui permet de partager équitablement les ressources entre les flots, selon leur arrivée dans la file d'attente. Nous donnerons plus de détails concernant ce modèle d'ordonnancement dans la partie du document réservée à la présentation de tous les procédés d'ordonnancement des files d'attente.

Le processus d'estampillage présentant quelques faiblesses de fonctionnement, la définition d'un temps virtuel est venue s'ajouter à l'estampillage des paquets. En effet, le calcul des estampilles ne permet pas de tenir compte de l'inactivité des flots : ainsi, un flot qui reste inactif pendant une période se verra attribuer, au moment de son activité, des faibles valeurs d'estampilles. Il se verra alors allouer des ressources indépendamment du poids du flot préalablement établi, ne correspondant pas aux ressources demandées. Par ailleurs, l'estampillage ne tient pas compte de la présence d'autres flots, information fondamentale pour permettre le partage des ressources. La notion de temps virtuel vient en aide à ces insuffisances pour signaler l'accélération d'un service lors de l'inactivité d'un ou plusieurs flots et permet de définir le temps que les paquets vont passer dans la file d'attente.

Pour définir ce temps virtuel, considérons les instants t_1 et t_2, tels que t_2-t_1 représente l'intervalle pendant lequel on dispose de l'activité des flots.
Le temps virtuel, noté v(t), est alors défini par la formule suivante :

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

où r_a indique le poids des classes actives.

En intégrant cette nouvelle notion de temps virtuel, la formule de calcul de l'estampille peut prendre la tournure suivante :

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

où a^i_k indique le temps d'arrivée du paquet i du flot k.

Nous présenterons ultérieurement les mécanismes d'ordonnancement qui adoptent l'utilisation du temps virtuel pour décider de la priorité des flux à l'intérieur des réseaux.

3.2.2 Algorithmes à files d'attente multiples

Intuitivement, nous pouvons penser que l'utilisation d'une simple file d'attente pour le traitement de multiples flots serait une opération coûteuse en délai de traitement, d'une part, et en matière d'interdépendance des flots les uns vis-à-vis des autres, d'autre part.
Une technique mettant en œuvre plusieurs files d'attente serait un bon moyen pour parer à ces insuffisances.
La première proposition développée par Nagle [76,77] a consisté à allouer et maintenir au niveau de chaque nœud, après classification, différentes files pour chaque catégorie de trafic. Le traitement qui s'effectue pour chacune des files est le suivant : les files sont séquentiellement parcourues une à une (*Round-Robin*) et chaque premier paquet qui se présente à la sortie d'une file est transmis.

La figure 3.2 représente le principe général de l'algorithme de base à file d'attente multiples.

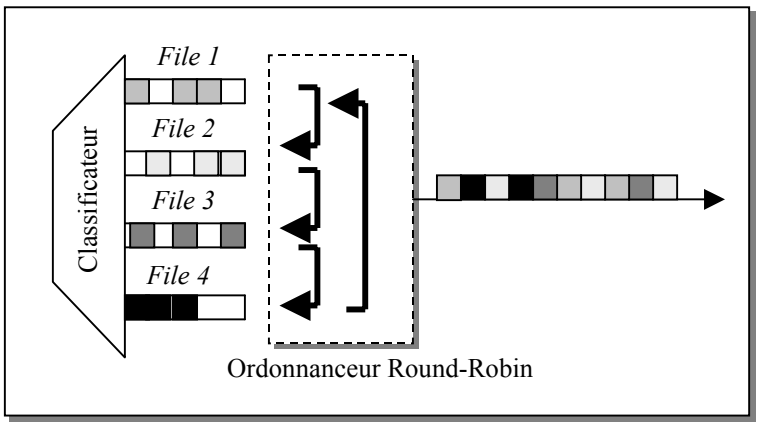


Figure 3.2 : Ordonnancement à files multiples

Décrivons à partir de l'exemple ci-dessus le fonctionnement de l'ordonnancement round-robin. Après classification des flux vers quatre files d'attente, l'ordonnanceur va examiner la première file. A sa tête, on remarque qu'il ne se présente aucun paquet. L'algorithme incrémente alors son indexation pour examiner la seconde file : un paquet est présent et est donc envoyé sur la sortie du lien. L'ordonnanceur passe à la file suivante et effectue le même traitement : un paquet se trouvant à la tête de la file, va être transmis sur le lien. Enfin, la file 4 est examinée, mais aucune information n'est encore présente, l'ordonnanceur va boucler son indexation vers la première file d'attente.

L'algorithme *round-robin* ne discrimine aucune file et essaye d'effectuer par conséquent le partage de bande passante le plus équitablement possible. L'équité ne peut cependant être parfaite que si la taille des paquets est identique pour tous les flux se présentant au niveau des files. Or, cette hypothèse n'est que très rarement vérifiée, et par conséquent, la notion de partage équitable parfaite ne peut être satisfaite dans ce cas de figure : en effet, les files contenant des paquets deux fois plus volumineux que ses concurrentes se verront attribuer, raisonnablement, deux fois plus de bande passante. Par ailleurs, nous avons remarqué que cet ordonnancement parcourant séquentiellement les files une à une, peut rencontrer une file sans paquet en attente et devra, de par son algorithme, passer à l'exploration de la file suivante. Par conséquent, cette méthodologie peut être néfaste vis-à-vis de certains paquets. Supposons à cet effet que le check-up de la file *i* ait été effectué quelques micro-secondes avant qu'un paquet de cette même file arrive. L'algorithme n'ayant trouvé à cet instant aucun paquet à transmettre, va passer à la file suivante et ne reviendra sur cette file *i* qu'après avoir parcouru du nouveau les (*n-i*) files précédant la $i^{ème}$ file : le paquet arrivé peu de temps après la première vérification de sa file devra ainsi attendre un certain temps avant de pouvoir être traité. On peut donc juger de la performance de ce type d'algorithme en notant sa dépendance par rapport à l'arrivée des paquets.

Les algorithmes à files multiples ont été largement adoptés par différents auteurs. Tout comme les mécanismes d'ordonnancement à file unique, nous détaillerons tous les moyens utilisant les files multiples, au cours des prochaines sections, ce qui nous permettra de mieux comprendre chacune des techniques, d'une part, et de mettre en valeur les atouts et les faiblesses que chaque méthode expose.

3.3 Politiques d'ordonnancement

Les réseaux véhiculent simultanément un nombre important de données hétérogènes. Les distances qui séparent les stations émettrices des stations réceptrices sont importantes et induisent des délais de transmission ; la traversée des différents éléments du réseaux engendrent, de même, des délais ; les capacités des liens sont limitées. Ainsi, lors de la circulation de plusieurs flots, les caractéristiques du réseau engendrent des aspects négatifs sur la transmission de ces informations. Ces derniers peuvent se traduire par des délais d'acheminement importants, par une incapacité d'obtenir une bande-passante suffisante ou encore par la perte de paquets. Selon la nature de l'application, ces effets sont nuisibles à la qualité de service requise. En effet, dans des conditions de surcharge du réseau, les applications temps-réel qui ne peuvent pas disposer de ressources suffisantes ne pourront pas offrir la qualité de service attendue. Pour remédier à ces problèmes de partage de ressources, des mécanismes d'ordonnancement sont implémentés dans les routeurs, ce qui permet d'administrer la circulation des applications sur le réseau en attribuant des parts de ressources, plus ou moins équitables selon l'algorithme utilisé. Nous verrons au cours de cette section

comment sont implémentées les politiques d'ordonnancement et nous verrons quelles sont celles qui assurent le partage de la bande-passante, et celles qui sont plutôt orientées délai.

3.3.1 FIFO

Souvent assimilé à la politique *DropTail* de gestion des files d'attente, l'ordonnancement de type FIFO (*First In First Out*) est l'ancêtre et le plus facile d'implémentation de toutes les politiques présentes. FIFO ne requière pas la présence de plusieurs files d'attente car son principe repose sur l'algorithme suivant : les paquets sont envoyés vers la sortie dans le même ordre de leur réception.
FIFO est très peu complexe mais présente néanmoins des inconvénients qui le rendent peu efficace en présence de situations inadaptées. Ainsi, il est fortement recommandé de ne pas utiliser ce mécanisme d'ordonnancement pour des réseaux DiffServ qui demandent une classification des flots pour garantir des qualités de service adéquates aux demandes des utilisateurs. FIFO ne peut en aucun cas assurer cette fonction de distinction des flots puisque, d'une part, l'algorithme ne repose pas sur l'utilisation de multiples files, et d'autre part, l'algorithme, en acheminant les paquets selon leur ordre d'arrivée, ne distingue en aucun cas la classe des paquets (best-effort ou prioritaires). Enfin, les réseaux qui adoptent un ordonnancement de ce type remarquent une forte consommation de la bande passante, pouvant être due à l'envoi massif de données non nécessairement de type temps-réel, par exemple, et pénalisant ainsi les éventuels flots qui exigent leur priorité dans le réseau. C'est pourquoi ce type de mécanisme est notamment recommandé pour des réseaux à forte bande-passante entraînant de faibles délais et présentant une rareté de congestion.

La figure 3.3 illustre un modèle de réseau mettant en œuvre deux machines qui utilisent des applications différentes. Le premier client utilise des applications élastiques, par exemple un transfert de fichiers (FTP); le second participe, par exemple, à une séance de vidéoconférence. La première machine commence ses requêtes à l'instant t1, tandis que la deuxième utilise son application à l'instant t2, tel que t1<<t2. Les paquets envoyés par l'application FTP se présentent à l'entrée de la file d'attente avant les paquets de vidéoconférence et occupent fortement la bande-passante disponible. L'ordonnancement étant du type FIFO, les paquets qui se présentent en premier seront ceux qui seront délivrés en priorité. Ce cas de figure montre bien que les paquets ayant une forte contrainte temporelle (flots non-élastiques) sont retardés par la présence des autres paquets qui ne demandent pas nécessairement une contrainte de délai. Les flots élastiques se trouvent par conséquent discriminés par rapport à la qualité de service qui devrait leur être attribuée.

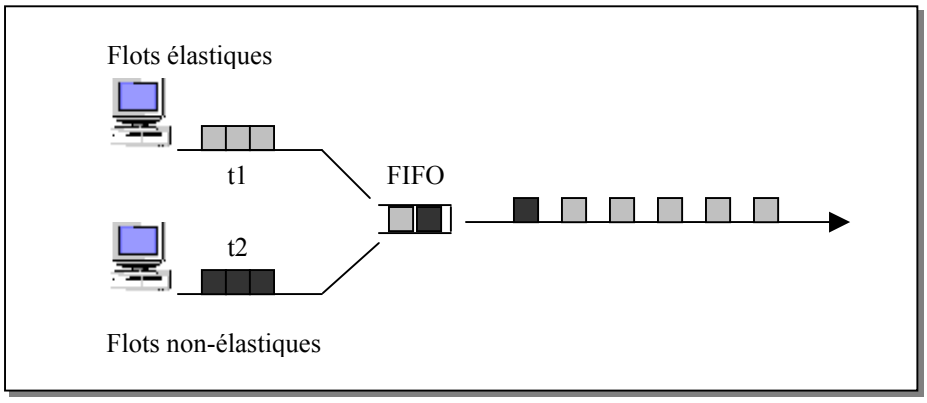


Figure 3.3 : Scénario utilisant l'ordonnancement FIFO

3.3.2 FIFO+

Proposée par [22], la politique FIFO+ a pour but de garder une simplicité d'implémentation de type FIFO, basée sur l'utilisation d'une file unique, mais en ajoutant des éléments de garantie de service, en particulier du point de vue délai. [26] montre à cet effet, à travers une étude analytique et simulée, l'efficacité d'un tel ordonnancement en matière de délai par rapport à un algorithme d'équité, le *Weighted Fair Queuing (WFQ),* que nous développerons par la suite.

L'algorithme de FIFO+ est décrit ci-après :

❶ Au niveau de chaque nœud, on mesure pour chaque classe, le délai moyen des paquets ;
❷ Puis pour chaque paquet, on calcule la différence entre le délai mesuré et la moyenne mesurée par classe ;
❸ On rajoute à l'en-tête du paquet, un champ contenant la valeur de cette différence ; cette opération permet ainsi à chaque nœud d'estimer l'instant d'arrivée d'un paquet dans le cas où il aurait eu un service moyen ;
❹ Le nœud insère le paquet dans la file en fonction de son temps effectif et son temps estimé. Ainsi, les paquets qui arrivent plus tôt que la moyenne sont placés en fin de file, tandis que ceux qui arrivent plus tardivement sont placés en tête de la file.

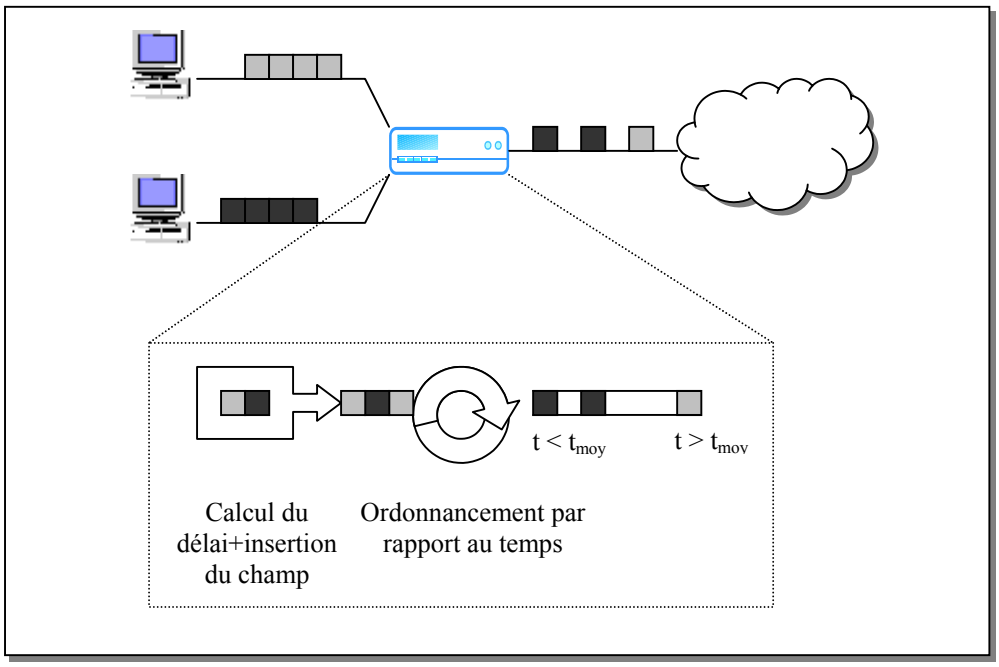


Figure 3.4 : Mécanisme d'ordonnancement FIFO+

Malgré son efficacité en terme de délai par rapport à FIFO et WFQ, le procédé FIFO+ reste une méthode à principe multiplexant. Cette caractéristique peut dans certains cas de figure, du fait de la dépendance des flots, violer les contrats de performances de délai [26].

3.3.3 Fair Scheduling : FQ

Proposé à la fin des années 80, l'ordonnancement *Fair Queuing* a pour principe de permettre à chaque flot d'un réseau de pouvoir accéder aux ressources de ce dernier de manière équitable. Le principe se base tout d'abord sur le classement des flots selon leur caractéristiques, puis sur l'utilisation de plusieurs files d'attente dédiées à ces flots. L'ordonnancement est effectué sur toutes les files contenant des paquets, de manière séquentielle, selon le mécanisme *Round-Robin* décrit antérieurement.
L'intérêt d'un tel algorithme est que la séparation des flots vers différentes files permet d'avoir des traitements indépendants. Ainsi, si un flot désire obtenir plus de ressources que la part qui lui est attribuée, seul ce flot sera affecté sans engendrer d'impact négatif sur les performances des autres files d'attente.
Néanmoins, cette politique présente l'inconvénient de partage non parfaitement équitable. En effet, l'ordonnancement *Fair Queuing* ne peut appliquer exactement l'équité que dans le cas où tous les paquets ont la même taille. Or ce scénario dispose d'une très faible probabilité de se produire réellement ; c'est pourquoi l'équité n'est pas utilisée par l'algorithme dans son sens le plus absolu [53]. La compétitivité de flots temps-réel et adaptatifs sur un réseau qui implémente *Fair Queuing* ne pourra en aucun cas garantir le délai ni le débit qui sont les principaux besoins (respectifs) de ces flots.
Cependant, même si cet ordonnancement n'est plus utilisé dans son état brut, il est réutilisé et raffiné pour la conception d'autres algorithmes, tels que le *Weighted Fair Queuing* ou encore le *Self-Clocked Fair Queuing*, que nous détaillons dans la section suivante.

3.3.4 Weighted FQ : WFQ

Dérivé de l'ordonnancement *Fair Queuing*, le *Weighted Fair Queuing* a été implémenté de manière à déterminer le nombre de paquets qu'il est nécessaire d'ordonnancer à chaque cycle, au niveau de chaque file. Cette quantité est définie par un « poids », d'où la notion de « *weighted* ». Cette notion a été proposée dans [30] pour pouvoir allouer différentes parts de bande-passante aux flots, selon leur besoin. WFQ est implémenté de manière à pouvoir supporter différentes tailles de paquets et ainsi assurer un traitement malgré l'hétérogénéité des flots.

Dans sa manière de servir les files d'attente, WFQ se base sur le principe employé par l'algorithme *Generalized Processor Sharing* (GPS) [52] dont le comportement se résume ainsi : les paquets qui sont placés en tête de file sont servis selon le poids qui leur est attribué, et le traitement des files s'effectue cycliquement et séquentiellement.

WFQ utilise cet algorithme pour réaliser l'ordonnancement, et rajoute un processus de calcul temporel pour chaque paquet (nous avons détaillé le calcul d'estampillage dans le paragraphe 3.2.1). Cette valeur permet ainsi de déterminer l'ordre dans lequel les paquets seront servis. Les paquets qui auront la plus petite estampille seront ceux qui seront acheminés en priorité.
Nous proposons de schématiser par la figure 3.5 le processus d'ordonnancement adopté par WFQ. En observant l'exemple que nous présentons, nous pouvons remarquer l'effet du poids attribué aux files par rapport à la desserte des paquets, malgré le traitement séquentiel des files. En effet, la première file dispose d'un poids de 50%, supérieur aux autres, ce qui permet de pouvoir obtenir une transmission de plus d'un paquet issu d'une même file (les deux premiers paquets de la file n°1).

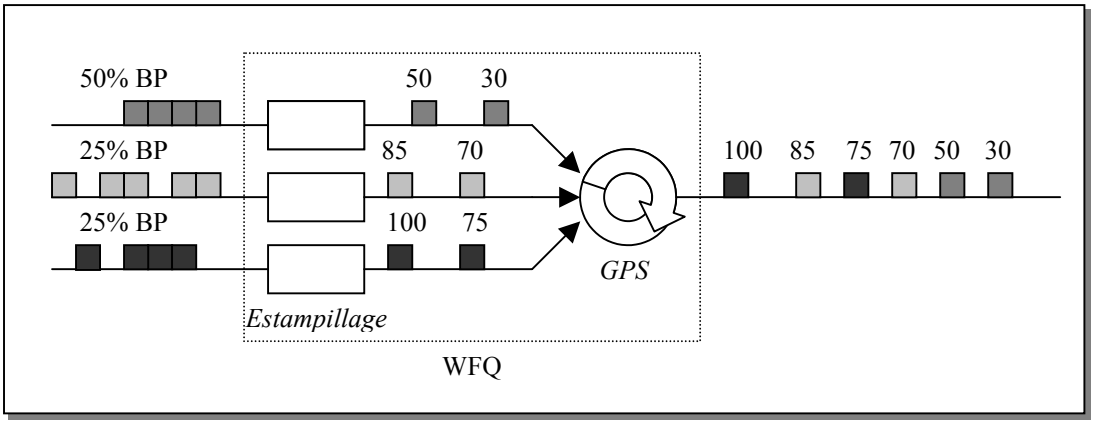


Figure 3.5 : L'ordonnancement par Weighted Fair Queuing

WFQ est un algorithme efficace en terme de partage de débit, et peut aussi assurer de bonnes garanties de délai si les routeurs qui implémentent cette politique disposent d'un mécanisme de lissage du trafic [2]. Mais de par la complexité de l'algorithme, ces paramètres peuvent se dégrader, notamment du point de vue délai, si le nombre de classes de service s'accroît notablement.

3.3.5 Worst-case Fair Weighted Fair Queuing :WF^{2}Q

Présenté dans [8], l'algorithme *Worst-case Fair Weighted Fair Queuing* est le modèle le plus proche d'implémentation de l'algorithme GPS qui reste tout de même à un niveau purement théorique. Comme nous n'avons pas développé l'algorithme de GPS, nous proposons de présenter un exemple de comportement d'ordonnancement sous GPS que nous comparons avec l'ordonnancement WF^{2}Q.

Pour cela, nous illustrons dans la figure 3.9 trois principaux éléments :

- tout d'abord, nous représentons l'arrivée des paquets de cinq flots qui se partagent la bande-passante du lien de sortie : le premier flot se réserve 50% des ressources tandis que les autres disposent de 10%. Chaque paquet est représenté par le binôme *i,j* où *i* représente le numéro du paquet et *j* représente le numéro du flot ;
- nous représentons ensuite la distribution des paquets et par conséquent leur temps de départ dans le cas de l'utilisation de l'algorithme (théorique) GPS ;
- enfin, nous schématisons le temps de départ des paquets en utilisant la politique d'ordonnancement WF^{2}Q.

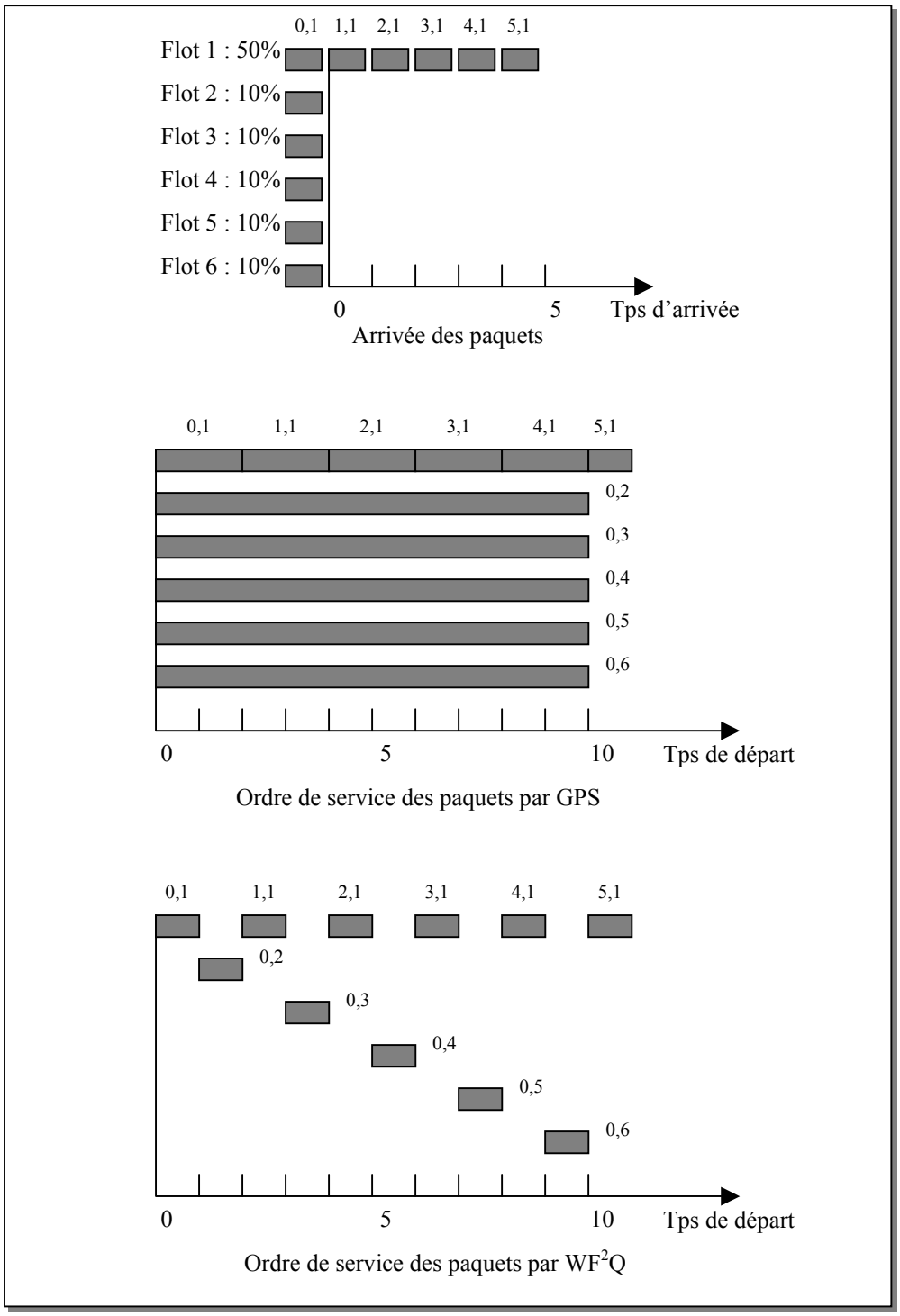


Figure 3.6 : Distribution des paquets pour les politiques GPS et WF^2Q

Le schéma présente bien la distribution efficace des paquets dans WF^2Q, sans discrimination de service de paquets par rapport à d'autres, et, dans cet exemple précis avec peu de flots, les garanties de délais peuvent être respectées. Mais la qualité de service des applications peut se dégrader si plusieurs flots sont concurrents, notamment en terme de délais.

3.3.6 Virtual Clock : VC

L'algorithme d'ordonnancement à horloge virtuelle est un mécanisme dérivé du WFQ. Développé par [105], il vise à réduire la complexité de WFQ. Son principe se fonde sur l'utilisation d'une horloge virtuelle attribuée à chaque flot qui permet de donner une indication en fonction de la quantité de données qui a été écoulée vers la sortie de la file.
On associe les paramètres suivants au mécanisme d'ordonnancement :

- D(i) le débit réservé associé à la file Q*i* ;
- V(i) une variable d'état associée à la file Q*i* ;
- P(i) la longueur d'un paquet de la file Q*i* ;

Pour chaque paquet qui arrive dans la file, on calcule sa nouvelle variable V(i), telle que :

$$V(i)=V(i)+\frac{P(i)}{D(i)}$$

L'ordonnanceur va ordonner les paquets selon l'ordre croissant des valeurs de cette estampille.
Pour comprendre le mécanisme de l'algorithme, nous proposons de présenter un exemple simple. Soit la figure suivante :

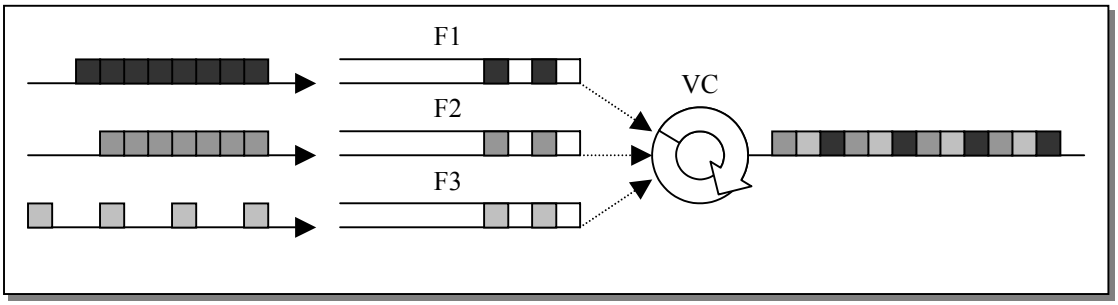


Figure 3.7 : Exemple d'ordonnancement par Virtual Clock

Pour simplifier, nous supposerons que les paquets sont tous de taille unitaire, et que les flots ont chacun à disposition D(i)=1/3 de la bande-passante disponible.
Dans cet exemple, nous avons choisi une arrivée des flots en même temps, c'est-à-dire que nous ne considérons pas le cas où l'un des flots est inactif durant les premières périodes.

L'ordonnancement des paquets s'effectue selon les valeurs calculées présentées dans le tableau 3.1.

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

Tableau 3.1 : Service des paquets sous Virtual Clock

Le cas de figure que nous avons présenté est un scénario idéal, c'est-à-dire qu'il prouve l'efficacité en terme de partage équitable des ressources du réseau : chaque file obtient effectivement le tiers de la bande-passante qui lui est attribuée. Néanmoins, cette situation est difficilement envisageable dans la réalité puisque l'arrivée des flots se fait de manière totalement aléatoire et il est très peu probable d'avoir des arrivées en même temps. Par conséquent, nous affrontons un cas de figure différent qui met en doute l'efficacité de l'algorithme que nous avons présenté. En effet, si l'on suppose une inactivité d'un flot durant une période donnée, les autres flots actifs pourront bénéficier de la part qui leur est attribuée, tandis qu'un pourcentage de bande-passante sera inexploité ; l'inefficacité de *Virtual Clock* apparaît au moment où ce flot, jusque-là inactif, entre en activité. En effectuant les calculs comme dans le tableau précédent, il est facile de montrer que le flot nouvellement actif bénéficiera à lui seul, durant une période T, de l'écoulement de ses paquets uniquement, tandis que les paquets des autres flots seront en attente. Cette situation se traduit en réalité par une désynchronisation des horloges virtuelles (avance ou retard) entre elles, phénomène qui induit un partage inéquitable.

Une variante de l'ordonnanceur *Virtual Clock* a été développée dans [29] pour des garanties de délais au niveau de réseaux de transit, c'est-à-dire pour des réseaux à faible étendue. Les résultats obtenus sont convaincants tout d'abord pour l'aspect temporel, puis pour des réseaux à faible envergure ; mais le partage équitable du débit n'est pas mis en valeur.
L'ordonnancement par *Virtual Clock* n'est donc pas un mécanisme favorable à une garantie en terme de débit si on se limite à l'algorithme en tant que tel. Une modification de ce dernier a donné naissance à une nouvelle politique qui permet de remédier aux inconvénients de VC. C'est ce que nous proposons de présenter dans le paragraphe suivant.

3.3.7 Self-Clocked Fair Queuing : SCFQ

L'algorithme d'ordonnancement SCFQ [51] est une solution qui va dans l'idée de la politique de VC. Son implémentation utilise les mêmes paramètres que ceux définis dans VC, auxquels on rajoute une variable d'état global V de l'ordonnanceur pour pouvoir réaligner les variables décalées. Cette variable d'état est associée à l'ordonnanceur telle qu'à chaque instant t, V est égal à l'estampille du paquet qui est en train d'être transmis sur la ligne de sortie. Le paramètre V représente l'horloge interne de l'ordonnanceur et indique la quantité de travail fournie par celui-ci.

L'estampillage pour cet ordonnancement est calculé, pour chaque paquet, comme suit :

$$V(i)=Max(V(i),V)+\frac{P}{D(i)}$$

Cette valeur va être associée au paquet et l'ordonnancement va s'effectuer selon les valeurs croissantes des paquets.
Cette politique s'avère d'autant plus efficace qu'elle ne présente plus le phénomène de décalage des horloges et assure donc un meilleur partage en terme de débit.

3.3.8 Priority Queuing : PQ

L'ordonnancement par priorité, PQ, utilise un processus de classification pour ordonner les paquets provenant des flux vers différentes files, qualifiées par une priorité. On distingue ainsi quatre niveaux de priorité. La file de plus haute priorité contiendra les paquets jugés par le classificateur comme étant les plus prioritaires, c'est-à-dire les paquets « gourmands » qui demandent une disponibilité de ressources suffisantes. Ces paquets appartiendront par exemple à des flots de type « hard real-time ». Les applications de type temps-réel seront classées vers les files de priorité moyenne ; les applications adaptatives seront plutôt orientées vers la priorité normale ou faible. A l'intérieur de ces files à priorité, le traitement est effectué selon l'ordonnancement FIFO. C'est à la sortie que se présente l'ordonnancement par priorité. La figure 3.8 illustre l'aspect utilisé par l'ordonnancement PQ.

A la tête de toutes les files l'ordonnanceur vérifie la présence des paquets à l'intérieur de chaque file et sert en priorité celles qui disposent de paquets prioritaires. Ainsi, les classes de plus forte priorité pourront disposer de la totalité de la bande-passante disponible, tandis que les autres classes ne pourront obtenir que ce qui leur sera resté après consommation par les autres classes. L'inconvénient d'un tel mécanisme est la forte pénalité d'obtention de ressources des paquets à faible priorité : si un important trafic de haute priorité circule sur le réseau, tandis que certaines applications de priorité moindre apparaissent de temps à autre, seul le premier sera correctement servi, tandis que les autres seront faiblement traités, voire fortement discriminés. Pour remédier à ce problème et éviter d'attribuer aux classes prioritaires beaucoup plus de ressources que les autres classes, il est nécessaire de placer un contrôleur d'admission pour pouvoir conditionner correctement le trafic et éviter d'obtenir uniquement du trafic à forte priorité. Mais il reste tout de même difficile de faire cohabiter sur un même réseau à ordonnancement PQ des flots « Best-Effort » et des flots temps-réel pour garantir une certaine équité du partage [19]. En effet, il est plus facile de pouvoir déterminer les ressources à allouer pour une application de type « voix sur IP » (VoIP) en connaissant à l'avance la taille des paquets, le volume et le comportement du trafic, que de pouvoir les déterminer pour une application de vidéo interactive, pour laquelle les paramètres subissent d'importantes variations. Les limites à poser pour servir les files prioritaires (taille de la file, bande-passante maximale à fournir, etc..) sont donc très difficiles à définir dans ces conditions, ce qui rend l'algorithme de *Priority Queuing* délicat à adopter pour des concurrences de flots hétérogènes.

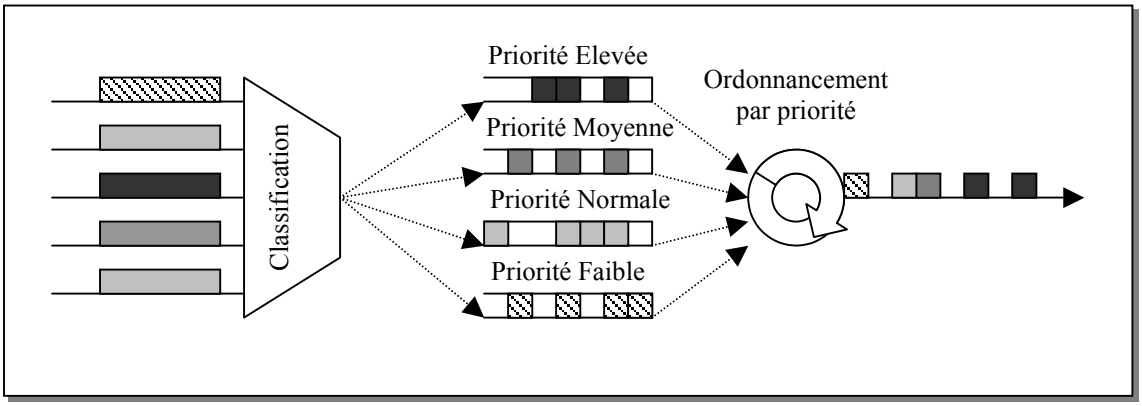


Figure 3.8 : Ordonnancement par PQ

3.3.9 Class Based Queuing : CBQ

D'un principe avoisinant la politique *Priority Queuing*, l'algorithme *Class Based Queuing* (encore appelé *Weighted Round Robin*) tente de remédier aux insuffisances des algorithmes *Fair Queuing* et *Priority Queuing*. D'une part, CBQ est destiné à traiter des flots avec différents besoins de bande-passante – ainsi, CBQ attribuera à chaque file un pourcentage de ressources à fournir ; d'autre part, CBQ assure qu'il n'y ait pas de famines pour les files de basse priorité, l'algorithme agissant séquentiellement sur toutes les files présentes.
L'ordonnancement CBQ utilise, tout comme PQ, un mécanisme de classification des flots (par exemple, les applications temps-réel, les applications interactives et les applications de type transfert de fichiers). A chaque classe est attribuée une file d'attente à laquelle est alloué un pourcentage de bande-passante à attribuer. A la sortie des files « par classe », un ordonnanceur général permet de partager la bande-passante du lien de sortie entre les différentes classes, selon le poids qui leur a été attribué, en utilisant le principe séquentiel de *Round-Robin*. L'estimateur évalue la bande-passante pour chaque classe et effectue un marquage selon le débit d'émission de chaque file : si la file émet à un débit moindre que la bande-passante qui lui est réservée, elle est marquée « *underlimit* » ; pour un débit d'émission égale à la part allouée, la file est marquée « *at limit* » ; enfin, elle sera identifiée comme « *overlimit* » si le débit de sortie de la file est supérieur au pourcentage défini. Un ordonnanceur « d'excès » est placé à la sortie de l'estimateur pour pouvoir distribuer un éventuel excès de bande-passante entre les classes. Ce mécanisme permet d'indiquer la (ou les) classe(s) qui est (sont) autorisée(s) à se partager l'excès de ressources, c'est-à-dire celles qui n'ont pas été consommées.

Nous proposons une illustration du mécanisme pour mieux comprendre le principe utilisé par l'ordonnancement CBQ (figure 3.9).

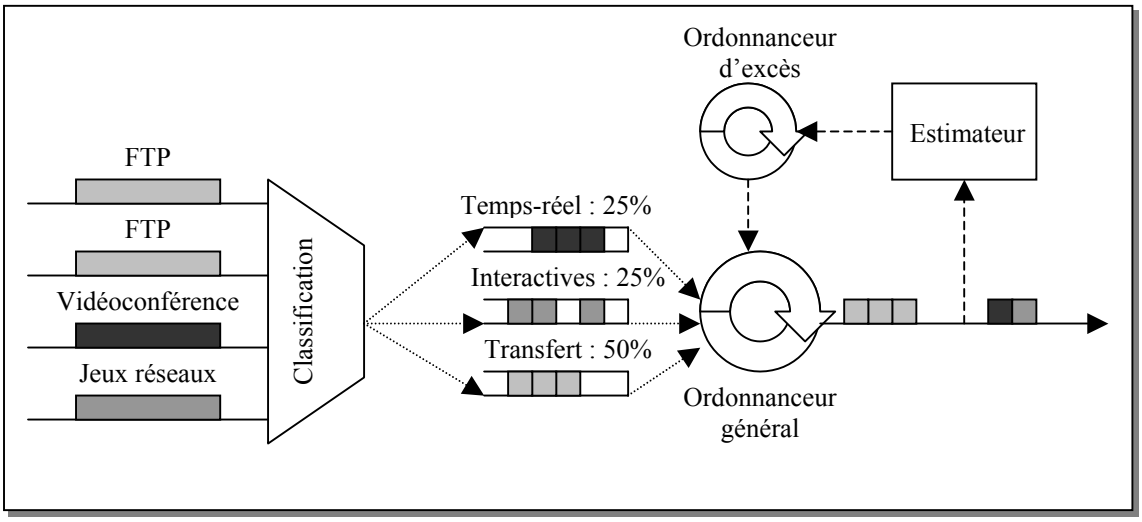


Figure 3.9 : Mécanisme utilisé par l'ordonnancement Class-Based Queuing

L'algorithme CBQ, bien que remédiant à certains problèmes signalés par d'autres politiques tels le partage plus équitable ou la redistribution de ressources inutilisées, reste tout de même peu intéressant pour une variété hétérogène de flots. La raison majeure est que les paquets ne sont pas tous identiques en taille et le taux d'arrivée est variable selon les applications : les transferts de fichiers ont des débits variables, tandis que les flots temps-réel émettent selon un débit plutôt constant, et la taille des paquets de ces applications est bien souvent différente. La garantie de bande-passante peut être respectée, mais encore une fois, comme certains ordonnancements que nous avons détaillés, la politique utilisée par *Class-Based Queuing* peut souvent engendrer des délais importants pour les applications, ce qui pénalise les flots à forte garanties de délai et la qualité de service pour ces classes ne sera pas respectée.

3.3.10 Alternative Best-Effort : ABE

L'*Alternative Best-Effort* a été proposé et décrit pour la première fois, par [61,62].
L'idée principale consistait à pouvoir fournir aux applications interactives un meilleur service que le *Best-Effort* utilisé actuellement. Le but de cet ordonnancement consiste à pouvoir assurer un meilleur acheminement des flots applicatifs et adaptatifs en leur garantissant un faible délai (par faible délai, nous entendons le délai de mise en file d'attente, celui-ci étant le plus important des délais à étudier pour les performances d'un réseau).
ABE se base sur le marquage des paquets par leur coloration. Deux types de paquets sont identifiés [64]:

- les paquets marqués en vert sont dédiés à recevoir un faible délai de traitement, mais subissent des pertes plus ou moins importantes selon la charge du réseau; ces paquets sont associés aux applications interactives ou temps réel de type audio, par exemple ;
- les paquets marqués en bleu subissent le même sort que les paquets qui circulent sur un réseau *Best-Effort*. Ne disposant d'aucune garantie en terme de délai, ils sont caractérisés pour obtenir un débit maximal. On associe donc ces paquets aux applications de type transfert.

Pour garantir la fonction d'ordonnancement, ABE utilise, à la suite du marquage des paquets, la technique de contrôle d'admission des paquets, suivie de la politique EDF (*Earliest Deadline First*)[63]. La figure 3.10 illustre le module d'ABE.

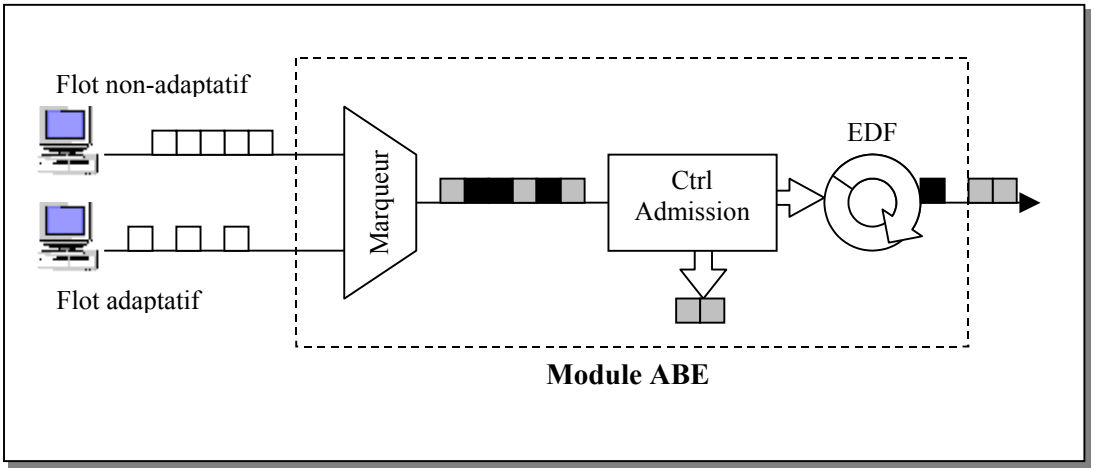


Figure 3.10 : Procédé de l'Alternative Best-Effort

Le processus se base sur le principe utilisé par RED : les paquets marqués en bleu seront éliminés selon la probabilité *p*, tandis que les paquets verts seront éliminés selon une probabilité αp. Ces derniers subiront enfin un autre contrôle pour pouvoir leur assurer un délai minimum. L'algorithme d'admission des paquets est détaillé dans [63].
Le mécanisme de EDF appliqué à ce niveau fonctionne comme suit : chaque paquet est estampillé selon sa couleur. En effet, pour les paquets verts, on choisira de le marquer selon son temps d'arrivée t, tandis que les paquets bleus seront marqués selon leur temps d'arrivée, auquel on rajoute une constante. Cette opération de différenciation d'estampillage permet ainsi de distinguer les paquets à faible délai des autres. De cette manière, les paquets verts (demandant un faible délai) seront prioritaires par rapport aux autres, mais l'algorithme agit de telle sorte qu'il n'y ait pas une discrimination totale vis-à-vis des paquets bleus.

3.3.11 Hierarchical Fair Share Curve : HFSC

La difficulté rencontrée par tous les algorithmes que nous avons définis est le respect de la qualité de service contractée par les applications mises en circulation, lorsqu'elles sont de natures différentes. Nous avons remarqué que toutes les techniques d'ordonnancement assuraient plutôt une garantie en terme de débits, mais moins en ce qui concerne les délais et de ce fait, pour chaque mécanisme, les flots à forte garanties de délais étaient plutôt pénalisés.
[97] propose un nouvel algorithme qui vise à remédier à ce problème de discrimination. L'idée de cette politique, le *Hierarchical Fair Share Curve*, consiste à découpler les aspects débits et délais de manière à spécifier distinctement les deux paramètres et ainsi offrir les garanties requises par chaque application, qu'elle soit de type adaptatif ou non.

Le principe de l'ordonnancement HFSC repose sur l'utilisation d'un modèle de courbes de services , l'une étant définie comme étant concave et l'autre, convexe [37]. Dans la définition de ce modèle, trois paramètres sont essentiels pour caractériser le trafic :

- u_i^{max} : plus grande unité pour laquelle l'application requiert les garanties de délai : cette valeur peut représenter par exemple, la taille d'un paquet d'une application audio, la taille d'une image pour une application vidéo ;
- d_i^{max} : délai garanti pour le plus grand élément du trafic ;
- r_i : moyenne du taux du trafic *i*.

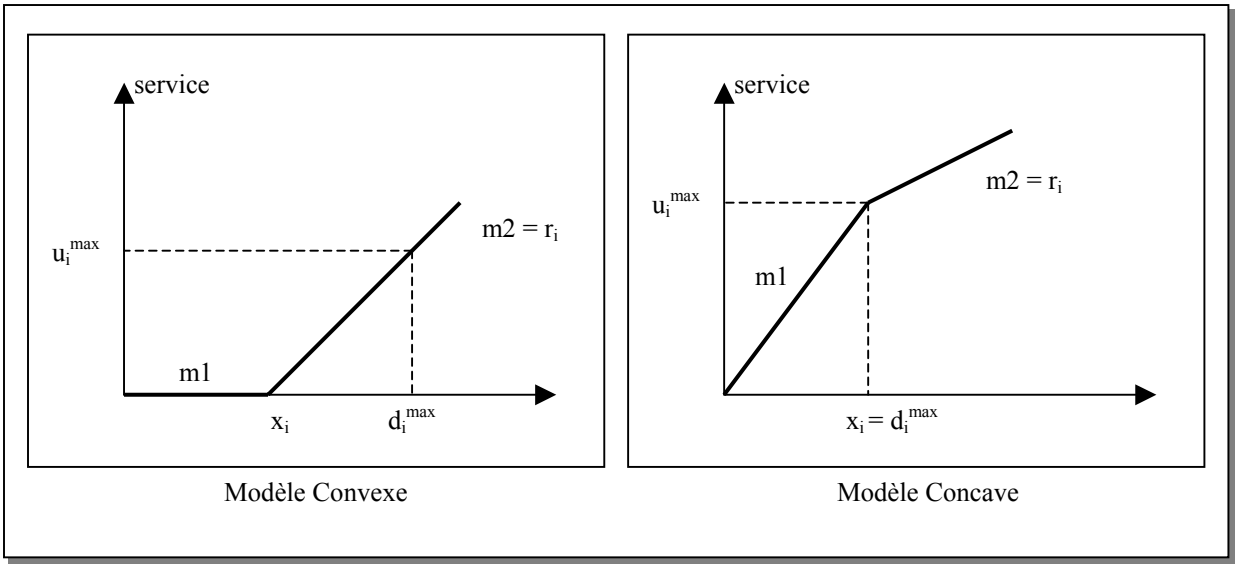


Figure 3.11 : Modèles des courbes de services de HFSC

Les courbes exploitées par l'ordonnancement HFSC sont caractérisées par deux éléments linéaires, m1 et m2, et x_i qui représente le point d'intersection entre m1 et m2. Dans le cas d'un modèle concave, on a $\frac{u_i^{max}}{d_i^{max}} \geq ri$ (ou encore m2>m1) , ce qui se traduit par une allocation de faibles délais. Dans le cas contraire, la courbe est convexe, se traduisant par un premier segment m1 toujours nul.

C'est par ce principe qu'il est possible de découpler le facteur débit de celui du délai et ainsi offrir aux applications au mieux les garanties qu'elles demandent.

3.4 Mécanismes de gestion de file d'attente

Contrairement aux ordonnanceurs, dont la fonction est de distribuer les ressources du réseau entre différentes classes de service, les mécanismes de gestion des files d'attente s'occupent de l'opération d'élimination des paquets appartenant à une même classe, en cas de congestion dans une file. Nous consacrons cette section à la description des différents dispositifs mis en œuvre pour gérer les files d'attente.

3.4.1 La politique DropTail

Le principe de la politique *DropTail* est très élémentaire et est souvent utilisée pour sa simplicité d'implémentation dans les routeurs. Cette technique permet d'écouler en priorité les paquets qui arrivent en premier dans la file d'attente. Si cette dernière se trouve saturée, c'est-à-dire que sa taille maximale est atteinte, les nouveaux paquets qui arrivent sont automatiquement éliminés. L'état de saturation est, trivialement, dû au phénomène suivant : le débit d'arrivée des paquets dans la file est supérieur au débit de sortie. La figure ci-dessous schématise le principe de fonctionnement de cette politique.

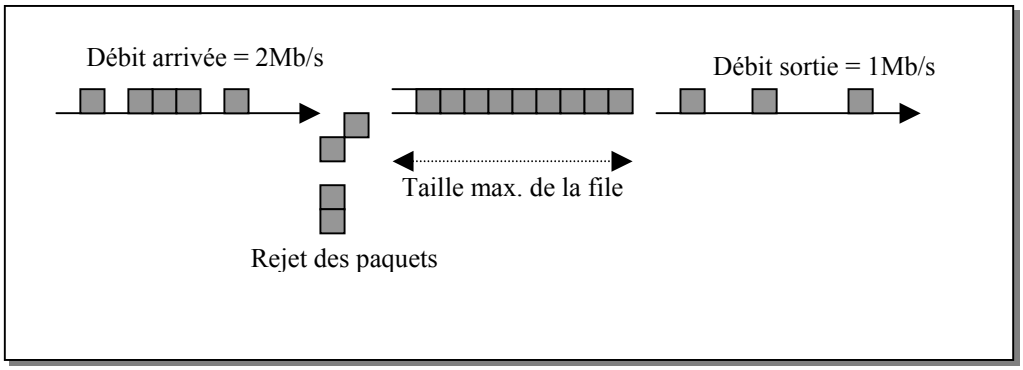


Figure 3.12 : Gestion de file d'attente par DropTail

Malgré sa simplicité d'implémentation, cette méthode présente des faiblesses nuisibles à une bonne utilisation du réseau, en particulier l'équité du partage de ce dernier : en effet, une source qui émet à un débit supérieur à une autre source se verra pénalisée par l'élimination de ses paquets. Par ailleurs, à cause d'importantes éliminations, des sources de type TCP ajustent leur débit en fonction de la charge du réseau, ce qui fait osciller la charge du réseau entre deux valeurs limites (maximale et minimale) et crée par conséquent un déséquilibre. Enfin, le remplissage permanent des files d'attente entraîne une augmentation du délai des paquets, pouvant être fortement néfaste à une application de type temps-réel, par exemple. Notons aussi que cette politique ne permet en aucun cas de distinguer la priorité des trafics, et par conséquent, tous les paquets, qu'ils soient en besoin de priorité ou non, sont traités de la même manière.

3.4.2 La politique RED

La politique *Random Early Detection* [41] met en jeu le paramètre "*threshold*", un intervalle de seuil dans lequel s'effectue l'élimination des paquets, et est basé sur la taille moyenne (*average*) de la file d'attente. L'algorithme développé permet de réduire le taux d'occupation des files et essaye de rester équitable entre les sources sur le partage des pertes. La figure 3.16 illustre le principe de fonctionnement de l'algorithme.

RED fonctionne selon le principe suivant : la probabilité d'élimination d'un paquet est basée sur l'intervalle de seuils définis [$threshold_{min}$; $threshold_{max}$]. Lorsque la taille de la file dépasse le seuil minimum, l'algorithme RED met en place sa politique d'élimination des paquets. La probabilité d'élimination s'accroît linéairement à mesure que la taille de la file augmente. Au-delà du seuil maximum autorisé, RED élimine systématiquement tous les nouveaux paquets entrants.

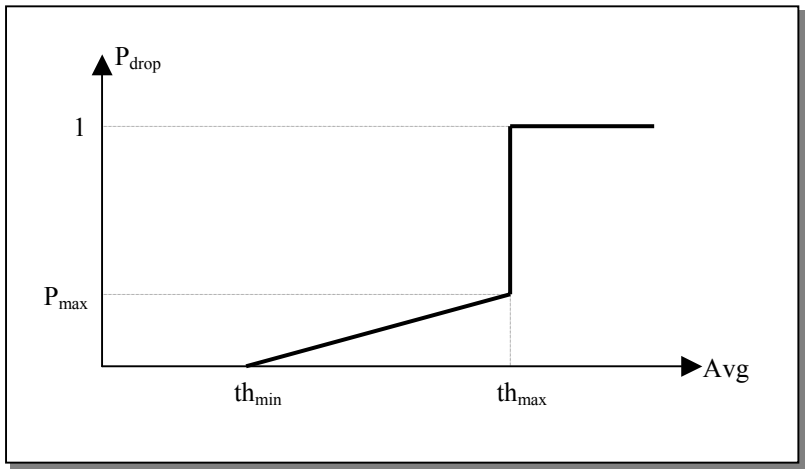


Figure 3.13 : Mécanisme de fonctionnement de RED

Cette politique de gestion de files d'attente est notamment utilisée pour les flots de type TCP en guise de mécanisme de contrôle de congestion : en effet, la méthode ECN (*Early Congestion Notification*) [57,88] utilise le même principe adopté par RED pour indiquer à une source TCP d'adapter son débit d'émission en fonction de la charge du réseau.
L'aspect négatif de RED est qu'il s'agit d'un mécanisme qui ne permet pas de gérer la priorité des paquets, ce qui pourrait largement affecter des flots à haute priorité demandant une importante qualité de service.

3.4.3 La politique RIO

Le mécanisme de RIO (*RED with In and Out*) se base sur le calcul de la longueur moyenne de la file d'attente, mais, par rapport à RED, adopte le principe de différenciation de services [24]. Le calcul de la moyenne est donc basé sur la précédence des paquets. RIO a été développé pour fonctionner dans des réseaux DiffServ avec des flots classés « In » et « Out », soit les flots qui appartiennent à une certaine classe de service, marqués « In » et ceux qui n'y appartiennent pas (marqués « Out »). L'avantage d'un tel mécanisme est qu'il permet d'anticiper et d'éviter les congestions plutôt que d'attendre que la congestion se produise pour pouvoir la contrôler par la suite.
La figure 3.14 présente le mode de fonctionnement du mécanisme de RIO.

Le principe de RIO est qu'il attribue des seuils et des poids de traitement selon l'appartenance des flots à une classe de service bien précise et effectue son traitement d'élimination en fonction de cette appartenance. En effet, nous pouvons remarquer sur la figure deux classes : la classe « In » est considérée comme étant la plus prioritaire ; la classe « Out » bénéficie de la priorité la plus faible. Les paquets qui appartiennent à la classe la moins prioritaire ont une probabilité d'élimination plus important que les paquets marqués « In » et l'élimination est effective à partir d'un certain seuil de taille moyenne de la file relativement faible par rapport aux paquets concurrents classés « In ». Ceci permet d'attribuer la priorité de passage aux flots qui demandent une forte qualité de service.

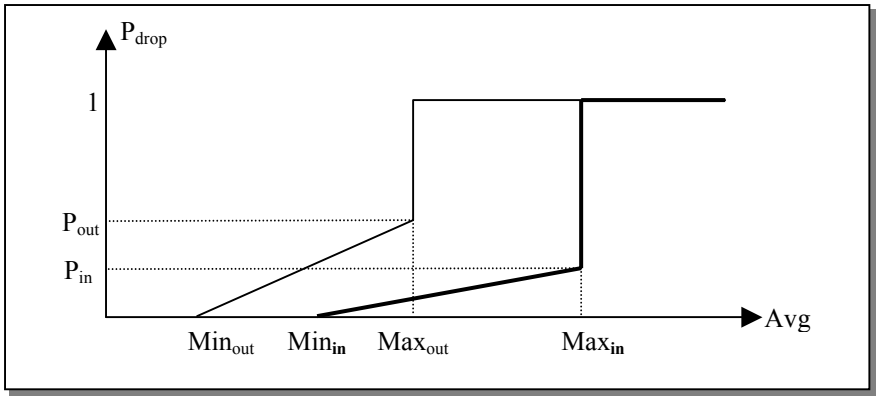


Figure 3.14 : Fonctions de probabilité d'élimination de RIO

3.4.4 La politique WRED

Le principe de la politique de *Weighted RED* se base notamment sur la politique de RED, mais résout le problème de priorité en assurant le traitement différencié. WRED est, en quelque sorte, une extension du modèle RIO puisqu'il permet de traiter la différenciation de service sur plus de deux niveaux. Par ailleurs, WRED peut fonctionner dans des réseaux IntServ utilisant RSVP, ou DiffServ. Avec RSVP, WRED choisit les paquets à éliminer parmi ceux qui appartiennent à des flots non RSVP. Dans les réseaux DiffServ, WRED fonctionne en utilisant le champ de priorité des paquets (*IP Precedence*) pour décider de l'élimination des paquets.

La figure 3.15 illustre le fonctionnement de WRED.

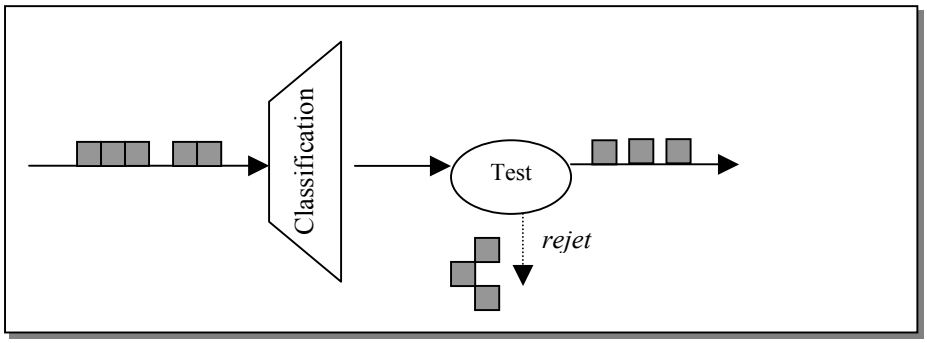


Figure 3.15 : Principe de WRED

A l'entrée de la file d'attente s'effectue une classification des paquets permettant de déterminer la provenance des paquets (IntServ, DiffServ). Le module de test permet à WRED d'éliminer les paquets en fonction de la taille moyenne de la file, et, selon le type du paquet, de la priorité de ce dernier. La taille moyenne de la file est calculée en fonction de la taille moyenne précédente et de la taille courante de la file d'attente.

Pour chaque paquet entrant, l'algorithme suivant est appliqué :

❶ Calcul de la taille moyenne de la file selon la formule :

$$avg = \left(old\_avg \times \left(1 - \frac{1}{2^n} \right) \right) + \left(current\_queue\_size \times \frac{1}{2^n} \right)$$

où n est un facteur de poids, configurable par l'utilisateur [21] ;

❷ Si la taille moyenne calculée est inférieure au seuil autorisé, le paquet est mis en file d'attente ;

❸ Si la taille moyenne de la file est comprise entre le minimum et le maximum du seuil autorisé, le paquet est soit éliminé, soit mis en file (tout est fonction de la probabilité d'élimination (figure 3.19)) ;

❹ Enfin, si la taille moyenne de la file est supérieure au seuil maximum, le paquet est automatiquement éliminé.

La figure 3.16 présente un scénario de WRED agissant sur trois niveaux de priorité : les paquets de faible précédence se voient rapidement éliminés, dès que la moyenne de la taille de la file atteint un niveau relativement bas ; la probabilité d'élimination des paquets de plus haute priorité est moindre, et s'effectue au moment où la moyenne de la taille de la file est plus importante.

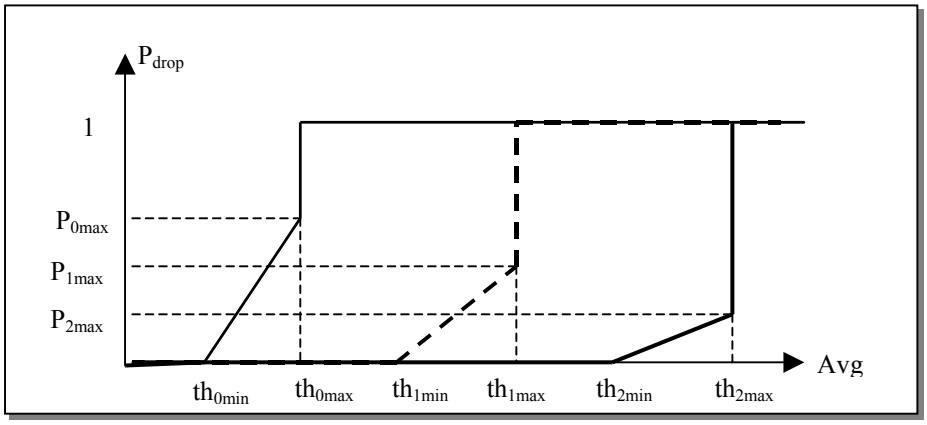


Figure 3.16 : Principe de fonctionnement de WRED

3.5 Conclusions

De nombreuses politiques d'ordonnancement ont été étudiées pour tenter d'offrir une qualité de service adaptée aux besoins des exigences des applications. Nous distinguons pour cela trois grandes catégories d'ordonnancement :

- les politiques qui offrent une garantie sur un paramètre de qualité de service : soit le délai, soit le débit ; les ordonnanceurs basés sur la priorité (PQ par exemple) sont ainsi étudiés pour des applications temps-réel mais ne sont pas conformes à une utilisation réelle à cause du phénomène de famine causé par une arrivée heuristique des paquets ;
- les algorithmes d'ordonnancement qui tentent de fournir une différenciation équitable sur le partage des ressources entre les flots concurrents ; le problème de tels procédés est un

partage non parfaitement équitable si des conditions se présentent sur le réseau (tailles différentes des paquets pour le cas de l'ordonnancement FQ) ;

- les mécanismes tels que ABE qui constituent une approche intéressante pour les flots temps-réel en terme de débit et de délai mais qui engendrent des forts taux de pertes, ce qui en fait des algorithmes non judicieux.

Pour remédier aux faiblesses causées par les algorithmes basés sur une telle différenciation (équitable ou relative à un paramètre), nous proposons dans le chapitre suivant une approche différente d'ordonnancement, basée sur la proportionnalité de partage des ressources.

ORDONNANCEMENT À BASE DE PUISSANCE POUR UNE DIFFÉRENCIATION PROPORTIONNELLE

4. Ordonnancement à base de puissance pour une différenciation proportionnelle

4.1 Introduction

Nous avons évoqué au cours du chapitre précédent, les différents mécanismes d'ordonnancement qui ont été développés dans le but d'améliorer l'acheminement du trafic sur les réseaux. Parmi tous ceux-ci, nous n'avons pas pu spécifier de véritable mécanisme qui puisse traiter l'équité entre le débit et le délai requis par les flots qui circulent de manière aléatoire et hétérogène. Les algorithmes implémentés traitent l'une ou l'autre caractéristique, mais jamais les deux en même temps. Cependant, la problématique de notre travail touche particulièrement à cette équité débit-délai qui doit être respectée entre les classes pour offrir aux différents types d'applications les besoins qu'ils requièrent, selon leur nature : plus de délai et moindre débit pour les applications multimédia interactives ; plus de débit et un délai moindre pour les applications utilisant le protocole TCP. Nous proposons dans ce chapitre de définir tout d'abord ce que nous entendons par différencier un service, et plus particulièrement la différenciation proportionnelle puisque nous visons d'attribuer les besoins de qualité de service de manière équitable. Pour illustrer le principe, nous définissons un ordonnanceur basé sur un modèle de différenciation particulier, *Waiting Time Priority*, établi sur le délai d'attente des paquets à l'intérieur des files d'attente. Nous décrivons par la suite l'approche que nous avons décidé d'adopter pour effectuer une différenciation reposant sur les paramètres de délai et de débit : la puissance d'un réseau s'exprime selon le ratio délai/débit et il est notable que ce paramètre met en jeu de manière rationnelle les deux métriques que nous jugeons utiles pour notre travail. C'est pourquoi nous avons orienté notre axe de recherche vers cette métrique, en appliquant la différenciation de service. Nous clôturons le chapitre par la modélisation de notre approche ainsi que par l'architecture générale des réseaux que nous avons testés.

4.2 Modèle de différenciation proportionnelle

Par définition, un modèle de service de différenciation proportionnelle est un procédé, décrit par un algorithme, qui permet de fournir de la distinction de service. La différenciation, appliquée au niveau IP, garantit qu'une classe de trafic donnée obtienne une performance proportionnelle à celle qu'obtient une autre classe, selon un coefficient déterminé. Cette différenciation a pour objectif d'atteindre deux aspects [31]:

- *la prédictibilité* pour laquelle la différenciation doit être ferme et consistante, indépendamment des classes de trafic qui transitent par le réseau ;
- *la contrôlabilité* qui correspond à la capacité d'ajuster la différence du critère qualité entre les classes selon les variations dynamiques des caractéristiques des réseaux (la charge notamment).

D'une manière générale, l'objectif d'un modèle de différenciation consiste à contrôler certaines métriques de performance préalablement choisies. Ces dernières peuvent représenter le délai, le débit ou le taux de pertes. La proportionnalité est alors définie comme suit :

$$\frac{m_i}{m_j}=\frac{c_i}{c_j},$$

m_i étant le paramètre de QoS choisi pour une classe i de trafic; c_i étant le paramètre de différenciation de qualité qui lui correspond. L'idée principale d'un tel modèle est de pouvoir distribuer également les ressources du réseau entre les différentes classes de trafic qui transitent, et ce, quelles que soient les fluctuations aléatoires du réseau (les plus importantes étant les dynamiques de la charge et par conséquent du délai).

Dans [31], les auteurs estiment que le meilleur moyen d'obtenir des performances de qualité de service des réseaux est de se baser sur les métriques que peuvent donner les files d'attente. Plus précisément, ils proposent d'observer les mesures sur les paquets, essentiellement le temps d'attente à l'intérieur de ces files ainsi que le taux de pertes.
Le modèle de différenciation proportionnelle est défini de la manière suivante : pour une période de mesure τ, et à un intervalle $(t, t+\tau)$, correspond une métrique de performance d'une classe i, que l'on notera $q_i(t, t+\tau)$.
La différenciation impose la condition d'avoir, entre deux classes de trafic i et j, une équité exprimée sous la forme : $\frac{q_i(t,t+\tau)}{q_j(t,t+\tau)}=\frac{c_i}{c_j}$, et telle que les paramètres de différenciation respectent la condition : $c_1<c_2<\ldots< c_n$. Ainsi, en appliquant un algorithme de différenciation, il est possible de contrôler la qualité de service attribuée à chaque classe et de préserver par conséquent l'équité de distribution des ressources, selon les besoins préalablement établis : aucune classe de trafic ne pourra donc être soumise à un phénomène de famine.

4.2.1 Différenciation proportionnelle de délai

Le modèle de différenciation préalablement défini est applicable dans un contexte temporel. En effet, au niveau des files d'attente, un certain délai d'attente est appliqué à tous les paquets. En considérant, sur un intervalle $(t,t+\tau)$, une moyenne de temps d'attente des paquets appartenant à une classe i, notée $d_i(t,t+\tau)$, il est possible, en substituant $q_i(t, t+\tau)$ par $1/d_i(t,t+\tau)$ et en appliquant le modèle général de différenciation, d'obtenir une différenciation sur le délai. Ainsi, nous aboutissons à l'expression suivante : $\frac{d_i(t,t+\tau)}{d_j(t,t+\tau)}=\frac{\delta_i}{\delta_j}$, les δ_i étant les paramètres de différenciation temporels et respectent donc l'ordre : $\delta_1>\delta_2>\ldots.>\delta_n$.

Une différenciation de ce type permet de fournir à toutes les classes de trafic concurrentes de se partager équitablement les délais et ainsi chaque flot bénéficiera d'une ration en terme de qualité temporelle, relative à sa demande. C'est sur cette théorie que différents mécanismes d'ordonnancement temporel ont été élaborés et implémentés. Nous citerons à cet effet, les algorithmes MDP (*Mean-Delay Proportional*) [78], HDP (*Hybrid Proportional Delay*) [34] et WTP (*Waiting Time Priority*) [31,32,34].

4.2.2 Différenciation proportionnelle de pertes

Vue sous l'angle d'une autre métrique, la différenciation proportionnelle peut s'appliquer à un second critère de qualité de service : les pertes. En notant $l_i(t,t+\tau)$ le taux de perte des paquets d'une classe i sur l'intervalle $(t,t+\tau)$, et en substituant, dans la formule générale de différenciation proportionnelle, $q_i(t, t+\tau)$ par $1/l_i(t,t+\tau)$, nous parvenons à la différenciation de pertes définie par : $\frac{l_i(t,t+\tau)}{l_j(t,t+\tau)}=\frac{\delta_i}{\delta_j}$, les paramètres de différenciation de perte suivant l'ordre $\delta_1>\delta_2>\ldots.>\delta_n$.

L'aspect de différenciation proportionnelle de pertes est traité et mis en œuvre dans [33,10], au moyen d'un nouveau mécanisme : le PLR (*Proportional Loss Rate*). Nous classerons cet algorithme dans la catégorie des processus de gestion de files d'attente, car l'algorithme n'assure par un véritable ordonnancement mais est plutôt orienté vers la fonction de décision d'élimination d'un paquet, telle la gestion par l'algorithme RED.

4.3 Ordonnancement par WTP

Parmi les moyens de différenciation proportionnelle, il existe une approche de différenciation de service selon le délai. Préalablement définie par [67] sous la forme de priorités à dépendance temporelle, elle a inspiré [32] pour définir un ordonnanceur de paquets à équité temporelle : WTP (*Waiting Time Priority*). Le principe général du processus se base sur le temps d'attente des paquets à l'intérieur de la file d'attente. La priorité attribuée à un paquet est proportionnelle au temps qu'il met dans la file d'attente pour être servi.

La figure 4.1 illustre le principe de différenciation temporelle mise en œuvre par WTP. Tout d'abord, l'ordonnanceur utilise une file d'attente par classe. A l'entrée de celui-ci, un temporisateur associe à chaque paquet le temps de son entrée dans l'ordonnanceur. L'estampillage permet ainsi de déterminer la durée passée par le paquet dans l'ordonnanceur. D'autre part, on attribue à chaque file *i* (*i.e* à chaque classe *i*), un coefficient s_i. A la sortie du mécanisme d'ordonnancement, WTP écoule les paquets qui possèdent la plus forte priorité. La priorité est calculée de la manière suivante : le routeur calcule le temps passé par chaque paquet qui se présente à la sortie de l'ordonnanceur, soit ω_i, et le normalise en le multipliant par le coefficient de la file d'attente auquel il appartient, soit s_i :

$$\varpi_i(t)=\omega_{.i}(t).s_i$$

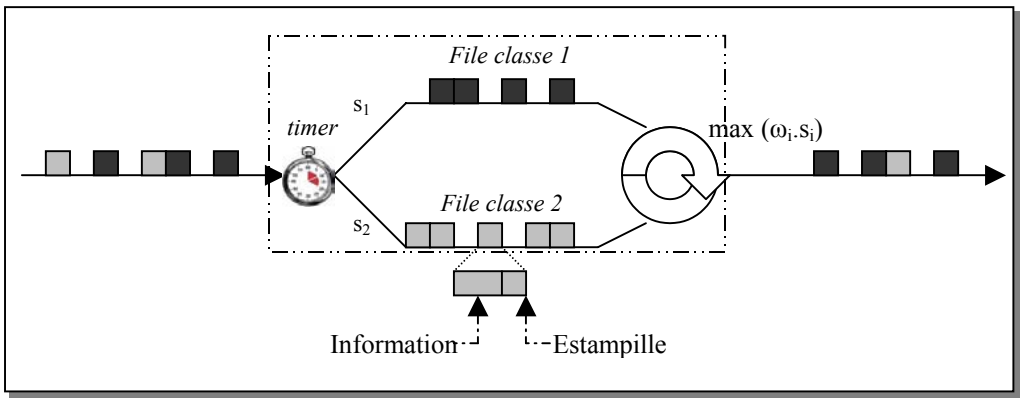


Figure 4.1 : Mécanisme de différenciation décrit par WTP

L'algorithme utilisé par WTP favorise la desserte des paquets qui, d'une part ont reçu un temps d'attente élevé, et, simultanément, d'autre part, qui appartiennent à la file dont le coefficient est le plus important. En d'autres termes, la classe la plus prioritaire est celle dont le coefficient s_i est le plus élevé.

La technique employée par WTP a pour inconvénient de ne pouvoir garantir des résultats satisfaisants qu'à condition que la charge du réseau soit importante (au-delà de 70% de charge). C'est pourquoi dans [36], il a été question de modifier l'algorithme de WTP de manière à ce qu'il soit aussi fonctionnel pour des réseaux à charge moins importante, mais restant toujours dans un contexte de charge modérée. L'ordonnanceur ainsi conçu, AWTP (*Adaptive Waiting Time Priority*), permet non seulement de tenir compte de la moyenne du temps d'attente des paquets à l'intérieur de la file, mais aussi de la charge mesurée à l'intérieur du routeur. De ce fait, la différenciation se fait non plus uniquement lorsque le réseau atteint une charge importante, mais s'effectue lorsqu'il atteint un niveau d'encombrement raisonnable (à partir de 50%).

Malgré leur efficacité, ces deux ordonnanceurs sont conçus pour ne traiter la différenciation qu'au niveau temporel. Cependant, les flots transportés par un réseau sont hétérogènes. Nous les avons classés, dans le chapitre 3, selon les deux métriques qui font d'eux des flots élastiques ou non élastiques. Néanmoins, si le réseau ne dispose pas de processus qui permet de distribuer efficacement la qualité de service, les applications ne bénéficieront certainement pas des ressources dont elles ont besoin. Une dégradation de la qualité apparaîtra nécessairement pour au moins un type de flot : dégradation pour les flots élastiques si un minimum de débit ne leur est pas attribué et/ou partagé ; dégradation pour les flots non élastiques si le délai qu'ils demandent ne peut pas leur être offert. Ces deux affirmations nous ont poussé à mener notre recherche selon l'objectif suivant : comment peut-on agir pour éviter la dégradation des flots hétérogènes sur un réseau ? En d'autres termes, quel processus est-il nécessaire d'utiliser pour distribuer équitablement les ressources d'un réseau en terme de délai et de débit et ainsi obtenir une satisfaction des clients qui utilisent les diverses applications ? Nous répondons à ces questions dans le paragraphe suivant.

4.4 Un ordonnanceur équitable à base de puissance : PSP

Les diverses approches proposées en terme d'ordonnancement différencié ont le plus souvent mis en jeu les paramètres de délai et de pertes pour assurer de la qualité de service. Dans [70], une différenciation est effectuée sur le délai, troisième critère pour garantir une certaine qualité de service. Pour le délai et les pertes, nous avons préalablement distingué les différentes méthodologies pour exercer une différenciation proportionnelle sur ces paramètres. La proportionnalité est appliquée soit sur chacune des métriques, indépendamment l'une de l'autre, soit simultanément sur les deux [43,44,98]. Le modèle d'ordonnancement EDS (*Equivalent Differentiated Services*) [45] offre, par exemple, une différenciation proportionnelle selon les deux paramètres : le taux de pertes et le délai. Il se base les mécanismes WTP et PLR.
Cependant, nous n'avons jamais fait allusion à une différenciation proportionnelle en terme de délai et débit simultanément. C'est pourquoi, et afin de remédier aux insuffisances de proportionnalité apportées par les ordonnanceurs actuels, nous avons opté pour un mécanisme qui puisse satisfaire l'équité entre le débit et le délai.

L'algorithme que nous proposons permettra ainsi de classer et d'attribuer les priorités des flots en fonction de leurs besoins, en attribuant une part de priorité proportionnelle aux

requêtes. Ainsi, la différenciation s'effectuera selon les deux métriques de performance de qualité de service que sont le débit et le délai.
En abordant la proposition de basculer entre le débit et le délai, nous avons basé notre étude sur la fonction de puissance qui a été définie dans [86].

De manière générale, la puissance est calculée selon la formule :

$$P=\frac{\gamma}{\tau}$$

où γ désigne le débit et τ représente le délai.

4.4.1 Approche de la fonction de puissance

La puissance permet de déterminer l'intervalle et plus précisément le point d'apogée des performances d'un réseau.

Pour mieux comprendre la notion de puissance, considérons un modèle théorique de réseau ayant une charge λ telle que $0 \leq \lambda < 1$. Les métriques qui caractérisent un tel réseau sont définies par :

- $\tau(\lambda)$: le temps moyen de réponse d'un paquet qui correspond au temps moyen passé par ce paquet dans le système ;
- $\gamma(\lambda)$: le débit des paquets, soit le taux auquel arrivent les paquets dans le système.

Un système stable, sans pertes, est défini pour une charge inférieure à 100% de la capacité du réseau. Dans de telles conditions, nous pouvons représenter les courbes idéales de débit (figure 4.2) et de délai (figure 4.3).

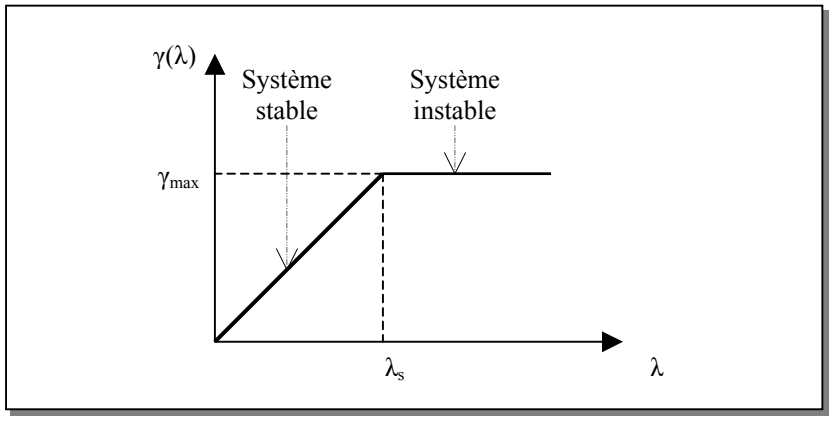


Figure 4.2 : Courbe de débit dans des conditions idéales d'un réseau

Pour un système stable, c'est-à-dire un système pour lequel nous avons la condition de charge $\lambda \leq \lambda_s$ (λ_s représente donc la charge maximale pour obtenir la stabilité) le débit croît proportionnellement à la charge du réseau jusqu'à atteindre le débit maximal γ_{max}. Au-delà de la charge maximale, le système arrive à saturation et devient ainsi instable.

Les caractéristiques en termes de délai pour un système stable sont représentées par la courbe ci-dessous :

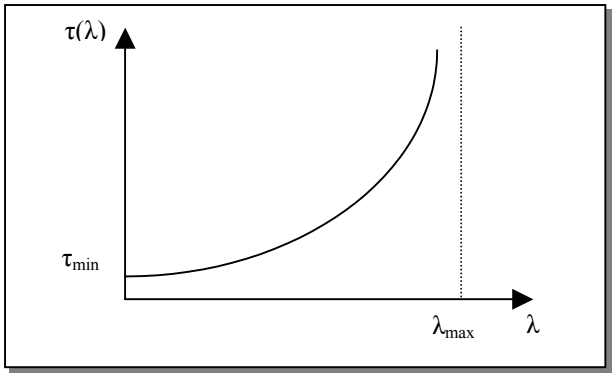


Figure 4.3 : Délai d'un système en fonction de sa charge

La figure 4.3 met en évidence l'évolution du temps moyen d'attente des paquets dans un réseau, selon sa charge. A mesure que le réseau se charge, le délai augmente considérablement, atteignant des valeurs très grandes pour une charge maximale λ_{max} .

Des deux figures précédentes découle l'observation suivante : plus la charge du réseau augmente, plus le débit augmente, mais le délai croît aussi de manière plus prononcée. Cependant, le trafic qui circule étant de nature hétérogène, les besoins requis en terme de débit et de délai sont aussi différents. Les applications temps-réel nécessitent, par exemple, de faibles délais pour écouler l'information sans dégradation qualitative. Un compromis doit alors être considéré pour garantir les caractéristiques requises pour chaque application. Celui-ci peut-être décidé en observant simultanément le débit et le délai ; la fonction qui offre cette possibilité est la fonction de puissance puisque, telle que nous l'avons décrite, elle met en jeu ces deux paramètres en mettant en évidence le rapport débit/délai.

Sur la courbe du délai, il est possible de déterminer le point correspondant à la puissance maximale du système. Celui-ci est identifié, à un niveau de charge λ_k, au moyen de la tangente à $\tau(\lambda)$ qui passe par l'origine (figure 4.4).

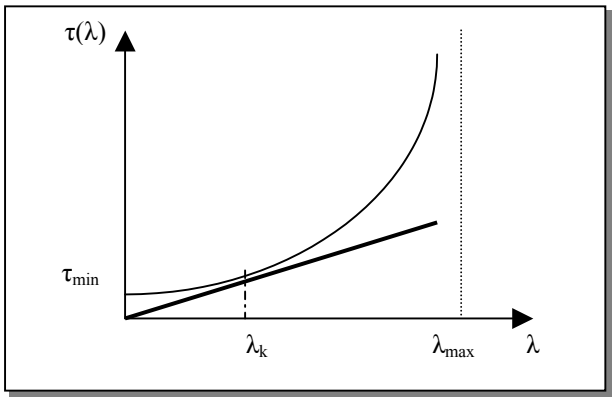


Figure 4.4 : Détermination du point maximum de puissance

La figure ci-dessous montre une autre manière de représentation de la puissance. En deçà de la charge λ_k , la puissance d'un réseau augmente jusqu'à atteindre le niveau maximum P_{max}, qui correspond à l'efficacité maximale du système ; au-delà de ce point, la puissance décroît pour atteindre des valeurs quasi-nulles pour une charge maximale.

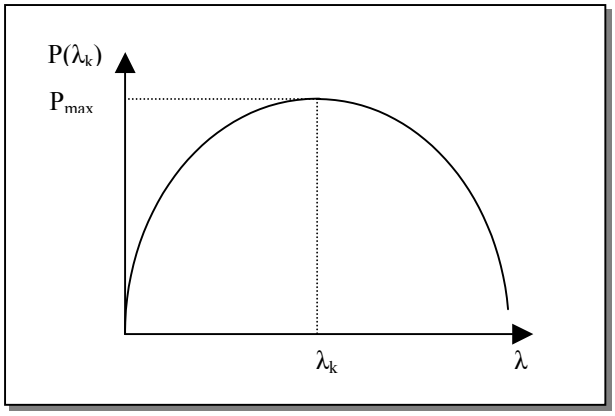


Figure 4.5 : Aspect théorique de la courbe de puissance

En rappelant l'expression de la puissance, $P(\lambda)=\frac{\gamma(\lambda)}{\tau(\lambda)}$, nous noterons que la valeur de cette fonction ne peut excéder la valeur unitaire : $0 \leq P(\lambda) \leq 1$. En effet, le facteur débit ne peut excéder l'unité, le maximum de bande-passante pouvant être atteint étant de 100%.

4.4.2 Modélisation de l'ordonnanceur PSP

L'ordonnanceur que nous proposons doit son appellation au paramètre principalement utilisé pour effectuer la différenciation. En effet, PSP (*Power as a Scheduling Parameter*) se base sur la valeur de la puissance mesurée pour une classe donnée, à l'instant *t*. La priorité d'ordonnancement des paquets est calculée selon la formule suivante :

$$Prio_i(t)=\frac{1}{P_c(t)}.sdp_c$$

Le facteur *sdp* (*Scheduler Differenciation Parameter*) constitue un second paramètre de différenciation pour l'ordonnanceur. Ainsi, à chaque classe de paquets est associé un *sdp* dont la valeur dépend de l'importance de la classe : plus une classe est prioritaire, plus la valeur du *sdp* est importante. Ce paramètre permettra à l'ordonnanceur de calculer de manière dynamique la priorité des paquets à acheminer.
Le second facteur qui permet d'évaluer la priorité des paquets est le paramètre de puissance $P_c(t)$. Pour la classe *c*, à l'instant *t*, la puissance est calculée comme suit :

$$P_c(t)=\frac{\gamma_c(t)}{\tau_c(t)}$$

Pour mieux comprendre notre approche de calcul des deux paramètres $\gamma_c(t)$ et $\tau_c(t)$, nous illustrons le processus d'arrivées/départs des paquets par la figure 4.6.

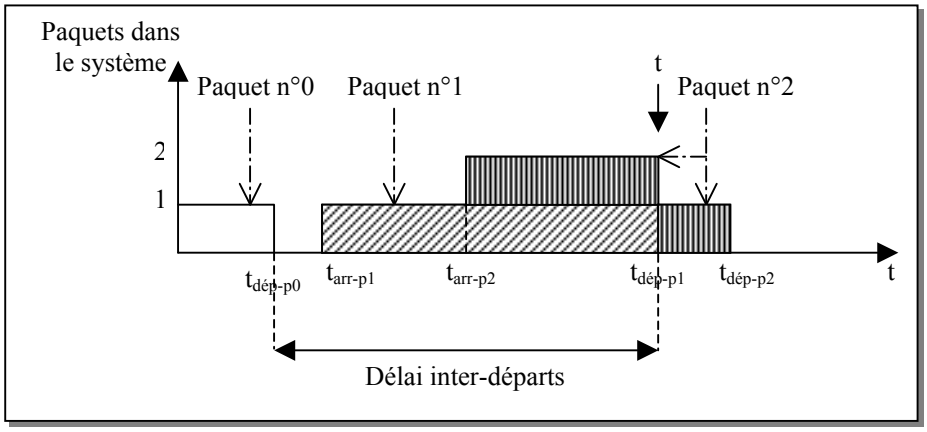


Figure 4.6 : Diagramme d'arrivées et de départs de deux paquets

Chaque paquet qui arrive dans le réseau doit être estampillé à l'arrivée et au départ pour permettre à l'ordonnanceur de calculer les priorités de chacun d'entre eux et pouvoir ainsi les acheminer selon la valeur de ces dernières.

Le débit d'une classe de paquets $\gamma_c(t)$ est déterminé au moyen de la taille du paquet qui se trouve en service à l'instant t et du délai inter-départs, c'est-à-dire les délais qui séparent les départs de deux paquets consécutifs. La formule ci-après permet de calculer ce débit :

$$\gamma_c(t) = \frac{\text{Taille du paquet en service}}{\text{Délai inter-départs}}$$

Pour illustrer cette formule plus pratiquement, nous nous référons à la figure 4.6, ce qui donne l'expression suivante, à l'instant *t* :

$$\gamma_c(t) = \frac{\text{Taille paquet\_1}}{t - t_{dép-p0}}$$

où :

- *Taille paquet_1* représente la taille du paquet n°1, c'est-à-dire le paquet qui se trouve en service à l'instant t ;
- *t* est l'instant auquel nous avons convenu de calculer le débit ;
- $t_{dép-p0}$ représente le temps de départ du paquet précédent le départ du paquet n°1.

Pour calculer le délai d'un paquet *i* appartenant à une classe *c*, nous devons nous fier à l'équation généralisée suivante :

$$\tau_c = T_{dép\_p\_c} - T_{arr\_p\_s} + T_{serv\_proc\_p}$$

où :

- $T_{dép\_p\_c}$ indique le temps de départ du paquet en cours ;
- $T_{arr\_p\_s}$ indique le temps d'arrivée du paquet suivant ;
- $T_{serv\_proc\_p}$ indique le temps de service du prochain paquet.

Pour comprendre plus facilement le processus, nous nous référons à la figure 4.6 et obtenons donc l'équation suivante :

$$\tau_c = t - t_{arr-p2} + \frac{\text{Taille paquet\_2}}{C}$$

avec :

- *t* est l'instant auquel nous avons convenu de calculer le délai ;
- *tarr-p2* désigne le temps d'arrivée du paquet 2 dans la file ;
- *Taille paquet_2* indique la taille du second paquet, c'est-à-dire celui qui sera mis en service ;
- *C* représente la capacité du lien.
- le rapport $\frac{\text{Taille paquet\_2}}{C}$ représente le temps de service du paquet qui sera traité à l'instant t+1.

En rappelant que la fonction de puissance est calculée en fontion du ratio débit/délai, nous pouvons illustrer par la figure 4.7 la théorie sur laquelle est basée la différenciation de service par PSP.

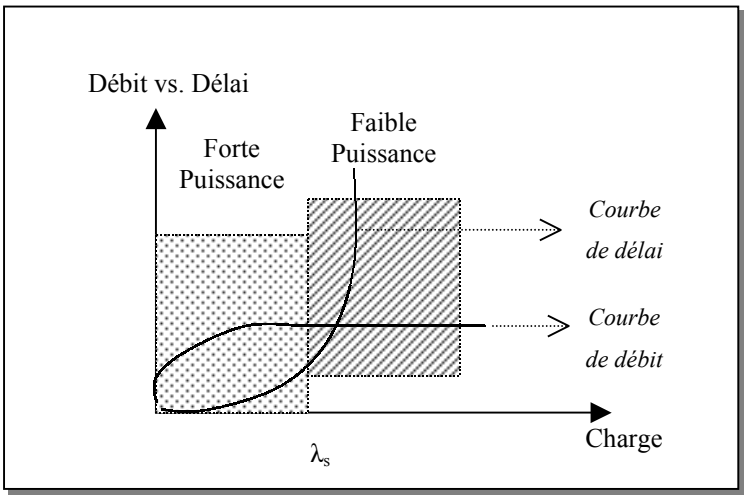


Figure 4.7 : Principe de distinction du paramètre de puissance

En fonction de la charge du réseau, nous avons représenté sur un même graphique les courbes de débit (asymptotique) et de délai (exponentielle) d'une classe quelconque de trafic. Nous avons de même défini deux zones de puissance : la première pour laquelle nous considérons une puissance élevée ; une seconde qui définit l'espace de plus faible puissance. Ainsi, la puissance étant un rapport de débit par rapport au délai, nous considérons qu'à partir d'un certain taux de charge du réseau, λ_s, la puissance calculée est considérée comme importante ; au-delà ce cette charge, la puissance est considérée comme faible. En-deçà de cette charge, la courbe du débit est plus importante que celle du délai. Sur cette portion, à un instant donné, la fonction de puissance donne une valeur supérieure ou égale à l'unité. Au-delà de λ_s, la courbe du délai croît exponentiellement tandis que celle du débit tend vers une stabilisation : la puissance correspondant à un instant donné sur cet intervalle est ainsi supérieure à l'unité. A partir de cette distinction, nous avons basé notre approche sur la théorie suivante : les classes de trafic qui ont une puissance supérieure à l'unité sont les applications qui sont gourmandes en métrique temporelle ; celles qui ont une puissance inférieure sont les applications qui sont plutôt exigeants en terme de débit.

A partir de cette théorie, en appliquant l'algorithme pour la différenciation sur deux classes de données, il est possible d'attribuer la priorité de service d'un paquet en fonction de ses besoins qualitatifs.

L'ordonnanceur que nous avons conçu attribue la priorité à un paquet appartenant à l'une des files en fonction de la puissance calculée à un instant t donné. Le mécanisme agit de la manière suivante : les paquets étant au préalable classés dans des files spécifiques en fonction de la catégorie du flot, l'ordonnanceur PSP applique l'algorithme de calcul de la puissance pour chaque paquet qui se situe à la sortie de la file. La priorité est attribuée à un paquet selon la valeur de la métrique de puissance.

Considérons pour ce faire, deux files d'attente :

- une première réservée à l'accueil des paquets de type UDP, et donc aux paquets qui sont gourmands en consommation de délai (applications interactives) ;
- une seconde destinée à recevoir les paquets de type TCP, c'est-à-dire les applications élastiques qui nécessitent des besoins en terme de débit.

L'ordonnanceur, placé à la sortie des files d'attente, exécute l'algorithme PSP pour chaque paquet qui attend d'être servi. Soient P_{UDP} et P_{TCP} les valeurs des puissances récupérées, respectivement sur le prochain paquet de la file UDP et de la file TCP.

P_{UDP} et P_{TCP} sont définies telles que :

$$P_{UDP}=\frac{\gamma_{UDP}}{\tau_{UDP}} \quad \text{et} \quad P_{TCP}=\frac{\gamma_{TCP}}{\tau_{TCP}}$$

Une comparaison est opérée sur ces deux métriques, aboutissant à une décision d'attribution de priorité, de manière équitable. La différenciation proportionnelle se traduit par les clauses suivantes :

- si $P_{UDP}=\alpha\ P_{TCP}$, avec $\alpha>1$, alors la priorité est léguée au paquet qui appartient au flot UDP, c'est-à-dire celui qui demande des exigences temporelles pour assurer la garantie de son application ;
- sinon, l'ordonnanceur achemine vers la sortie le paquet qui appartient aux applications de type TCP, gourmandes en bande-passante.

En appliquant une telle pratique, nous tentons d'éviter au maximum le phénomène de famine qui, dans l'Internet actuel, se pose, rendant ainsi difficile la garantie de service de certaines applications.

4.5 Topologie des réseaux à ordonnancement PSP

Dans cette section, nous proposons de décrire la topologie générale des types de réseaux que nous avons testés, de manière à exposer chaque constituant et en détailler la fonctionnalité. Nous avons qualifié la topologie de « générale » car, dans le chapitre suivant, nous remarquerons que l'architecture présentée dans cette partie ne représente que le squelette des réseaux que nous simulons. La figure 4.8 en illustre un exemple.

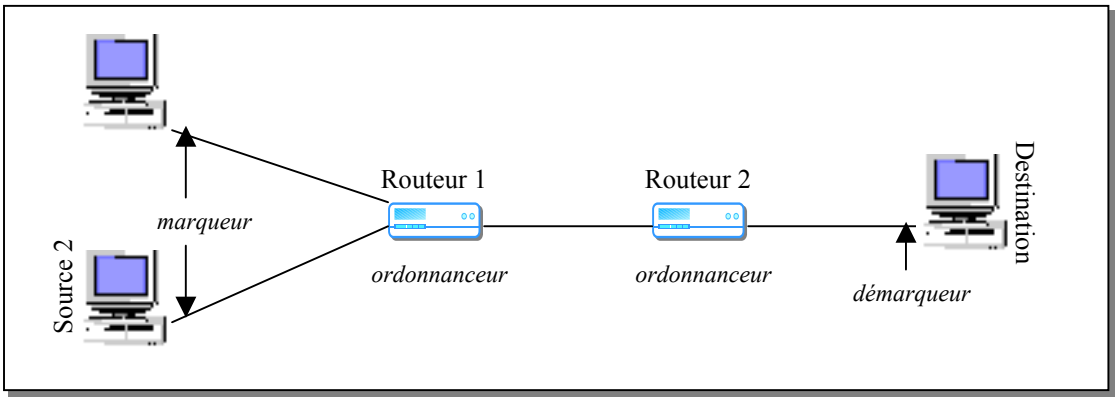


Figure 4.8 : Topologie d'un réseau expérimental

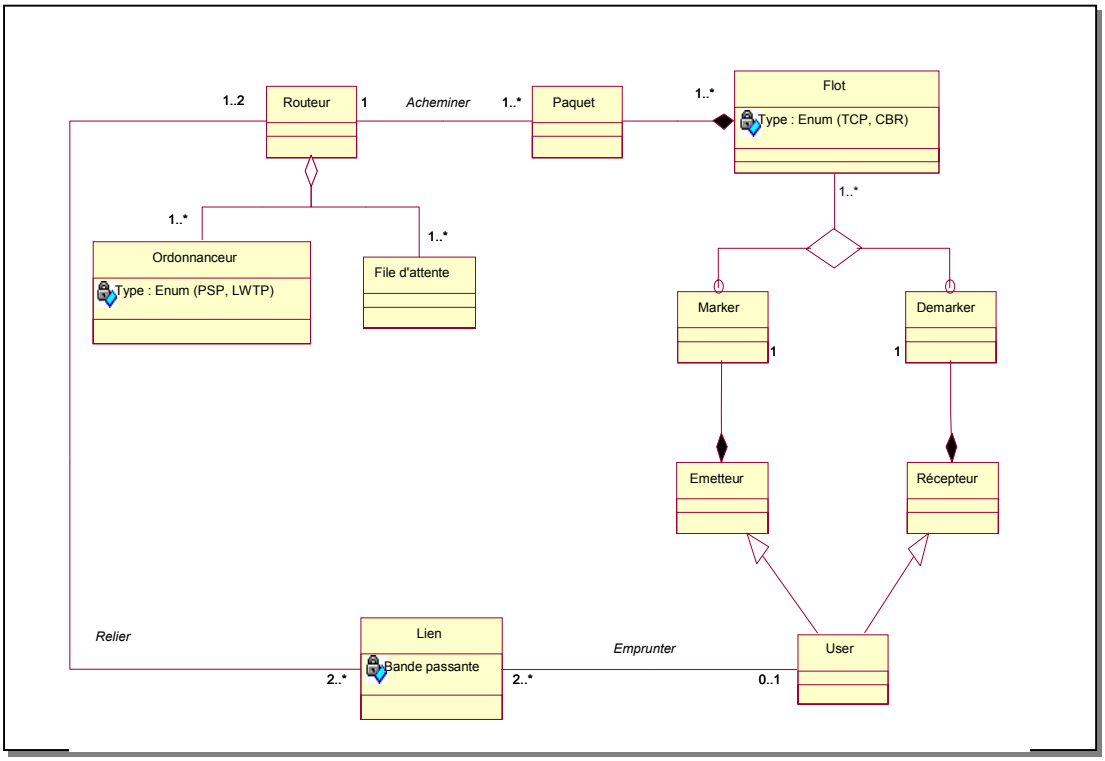


Figure 4.9 : Modélisation de la topologie du réseau

La figure 4.9 expose la topologie du réseau sous la forme d'une modélisation par UML [12]. UML (*the Unified Modeling Language for object-oriented development*) est un langage de modélisation fondé sur la modélisation orientée objets. Nous avons choisi d'adopter ce formalisme pour mettre en évidence les différentes caractéristiques des réseaux simulés que nous présentons dans le chapitre suivant. D'autre part, cette modélisation nous permet d'avoir une vue globale de ce que nous voulons représenter. Nous invitons le lecteur à se référer aux annexes pour mieux comprendre les principes de base et les notations utilisées (classes, associations, etc..).

Dans le contexte de notre recherche, nous avons adopté les faits suivants :

- un utilisateur, qu'il soit émetteur ou récepteur, dispose d'un élément essentiel pour identifier les paquets, et que nous décrirons par la suite : le marqueur pour l'émetteur et le démarqueur pour le récepteur ;
- les flots de données acheminés utilisent le protocole TCP ou UDP ;
- les routeurs disposent du moyen d'ordonnancement PSP.

4.5.1 Marqueur et démarqueur

Positionné à la sortie de la file d'attente de l'utilisateur, le marqueur a pour fonction, comme son nom l'indique, de marquer les paquets selon leur nature. Un paquet sera ainsi reconnu comme appartenant à une classe spécifique et sera acheminé vers la file d'attente qui lui est destinée. Le marquage s'effectue au niveau de l'en-tête de chaque paquet. D'autre part, le marqueur assure la fonction d'estampillage des paquets (figure 4.10). Ainsi, ces derniers disposeront de l'un des paramètres clé pour favoriser le calcul du délai de bout en bout à l'extrémité réceptrice du réseau.

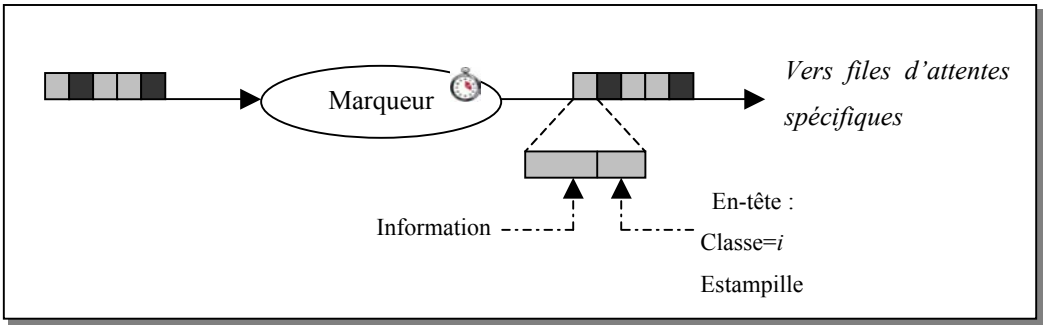


Figure 4.10 : Mécanisme du marqueur

Le démarqueur, placé en aval du réseau, traite les paquets en provenance du dernier routeur. Ces derniers, possédant au niveau de leur en-tête le marquage de la classe à laquelle ils appartiennent, le démarqueur distribue les paquets aux destinataires concernés par la réception de leurs paquets. La figure 4.11 schématise brièvement ce module.

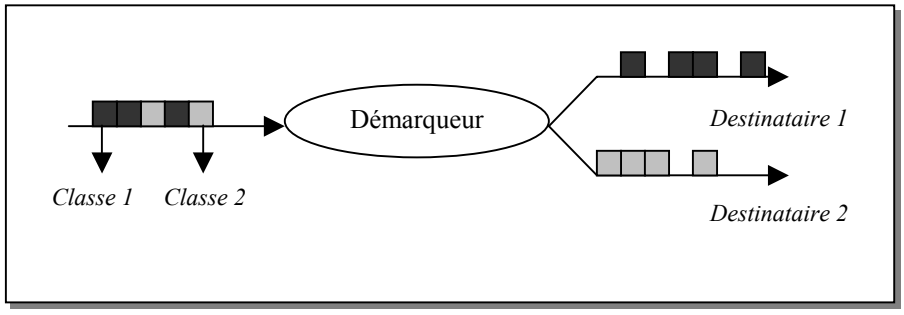


Figure 4.11 : Principe de fonctionnement du démarqueur

4.5.2 Flots de données

La nature des données que nous utilisons est de deux types :

- un trafic représentatif d'applications élastiques, basé sur le protocole TCP dont nous décrirons les caractéristiques exactes dans la section dédiée à l'implémentation des algorithmes ;
- une seconde catégorie représentative d'applications non-élastiques. Ces dernières sont connues pour leur fonctionnement au-dessus des protocoles UDP ou RTP. Pour nos simulations, nous avons orienté notre choix vers le premier protocole.

4.5.3 L'ordonnanceur

Situé au niveau du module de routage, le principe de fonctionnement de l'ordonnanceur PSP est représenté par la figure 4.12.

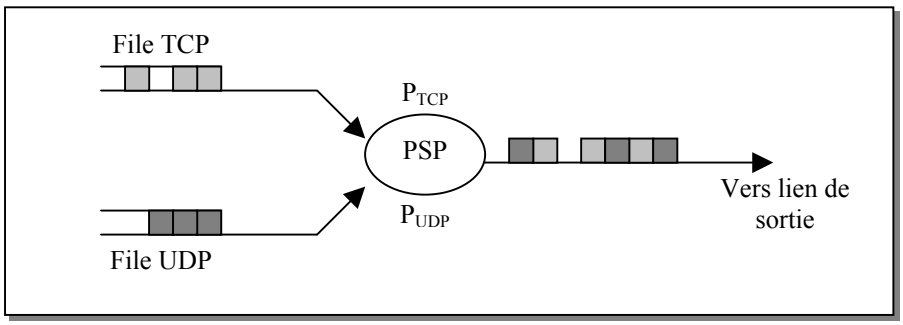


Figure 4.12 : Mécanisme d'ordonnancement PSP

La distribution des paquets à sa sortie est réalisée selon l'algorithme PSP. Naturellement, nous rappelons que l'ordonnancement est exécuté de manière équilibrée, c'est-à-dire qu'en aucun cas un flot ne doit subir d'inégalité de traitement. Les flots bénéficieront d'une part équitable de ressources en débit ou en délai selon leur nature.

4.6 Conclusions

Nous avons présenté dans ce chapitre la modélisation d'un nouvel ordonnanceur à base de puissance dont le but consiste à assurer une distribution équitable des paramètres de QoS qui sont le débit et le délai. Ce partage s'effectue en faveur de deux types de trafic : un trafic basé sur TCP pour représenter des applications élastiques, mais qui demandent un débit minimum ; d'autre part, un trafic basé sur UDP pour modéliser des applications temps-réel et qui demandent des garanties en terme de délai. Ce prototype nous amène à son implémentation. Cette dernière, ainsi que la validation des résultats sont présentées dans le chapitre suivant.

SIMULATIONS & RESULTATS

5. Simulations et résultats

5.1 Introduction

Les réseaux de télécommunications, et en particulier les réseaux informatiques connaissent une expansion sans précédent. La modélisation et la mise en place d'un nouveau prototype nécessite sa validation par l'un des processus suivants : la simulation au moyen d'outils spécifiques, ou l'expérimentation sur une plate-forme réelle.
Ne disposant pas d'une véritable plate-forme de tests, nous avons opté pour la solution de simulation. Il existe actuellement différents outils de simulation de réseaux. Nous citerons par exemple les simulateurs les plus utilisés par les chercheurs universitaires, tels que QNAP [87] ou *ns-2* (Network Simulator) [81] que nous pouvons obtenir gratuitement, ou encore, dans la catégorie « logiciels payants », le simulateur OPNET [82]. A la suite des compromis dont nous disposons (coût du logiciel, performances en terme de traitement des réseaux et files d'attentes), nous avons orienté notre choix vers l'adoption du simulateur ns-2.

5.2 Le simulateur *ns-2*

Network Simulator est l'un des outils de simulations des plus populaires au sein de la communauté scientifique. Développé par le département des techniques informatiques à l'université de Berkeley en Californie, *ns-2* offre un moteur de simulation de réseaux pour permettre à l'utilisateur de décrire et simuler des communications entre nœuds des réseaux IP. Le simulateur fonctionne avec des scripts écrits en Tcl/Tk ou en OTcl ; une utilisation du C++ est toutefois conseillée pour les simulations délicates.
A travers ce langage, il est possible de modéliser tout type de réseau et de décrire les conditions de simulation : la topologie du réseau (LAN, sans-fil, etc…), les caractéristiques des liens physiques, le type de trafic qui circule, les routeurs et les mécanismes d'ordonnancement à appliquer, les protocoles utilisés, les communications qui ont lieu, etc...
Ce dernier a pour essentiel point fort de pouvoir intégrer de nouvelles fonctionnalités et mettre à jour sa bibliothèque : chaque année, une nouvelle version de *ns-2* apparaît.

La simulation avec *ns-2* passe en général par trois phases :

- Définition de la topologie du réseau : dans cette partie, on définit les nœuds et les connexions. On peut définir sur chaque lien, le délai, la bande passante, le fait qu'il soit simplexe ou duplexe et le type file d'attente se trouvant à son extrémité.
- Spécification du scénario de la simulation : dans le cadre de cette étape, l'utilisateur spécifie les différents agents de la communication qui vont agir pendant la simulation (tel ou tel nœud envoie ses données, routeurs actifs (respectivement inactifs). Il spécifie aussi

la succession des différentes opérations (à l'instant t_1, envoi des données ; à l'instant t_2 arrêt d'émission). Il spécifie enfin les différents comportements que prend le réseau vis-à-vis de tel ou tel événement.

- Exploitation des résultats : cette dernière phase consiste en un recueil des statistiques de la simulation. Ces dernières peuvent être exploitées directement par *ns-2* ou par l'un des outils qui l'accompagnent (outil de tracé graphique : *xgraph*, outils d'animation de la simulation : *nam*) ou bien elles seront archivées pour une utilisation ultérieure au moyen d'autres outils de traitements statistiques.

ns-2 est organisé sous forme de classes : il offre une large bibliothèque de modules écrite dans les langages OTcl et C++, caractérisant chaque élément de fonctionnement du réseau. On trouve, par exemple, les classes *Node* pour définir un nœud, *Link* pour le lien physique, *DropTail* pour la politique d'ordonnancement FIFO,...etc. Ces classes disposent de méthodes qui permettent de personnaliser les propriétés des objets. L'utilisateur peut toujours modifier ces modules et les adopter à ses propres besoins. Il peut de même en intégrer de nouveaux, ce qui enrichit chaque jour la liste déjà présente. Ceci est valable pour tous les objets et les méthodes de *ns-2*.

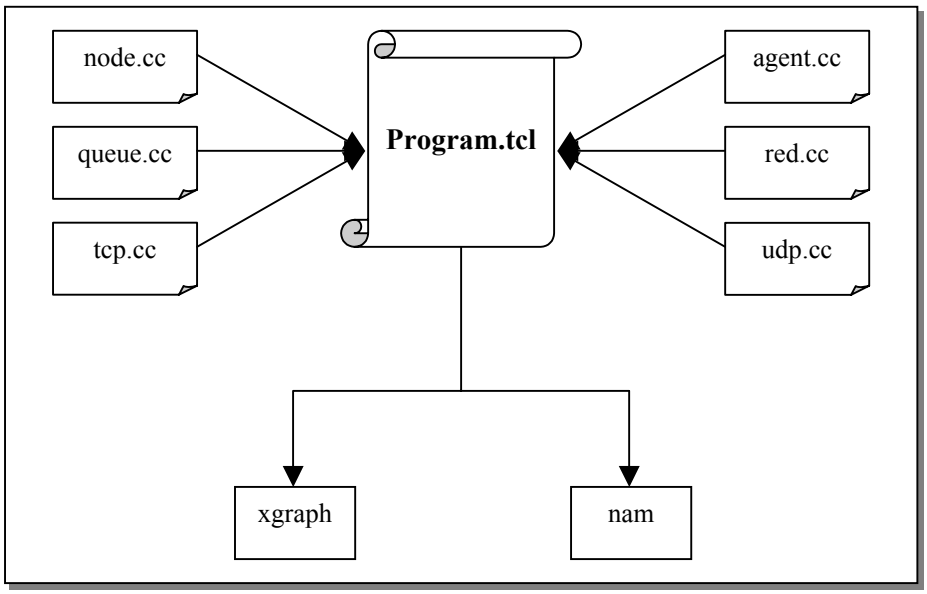


Figure 5.1 : Représentation modulaire de l'outil de simulation ns-2

5.3 Implémentation de l'algorithme

Dans cette section, nous présentons l'architecture du réseau sur lequel nous avons effectué les simulations, puis nous détaillons l'algorithme de chaque module utilisé pour la réalisation des tests.

5.3.1 Topologie du réseau simulé

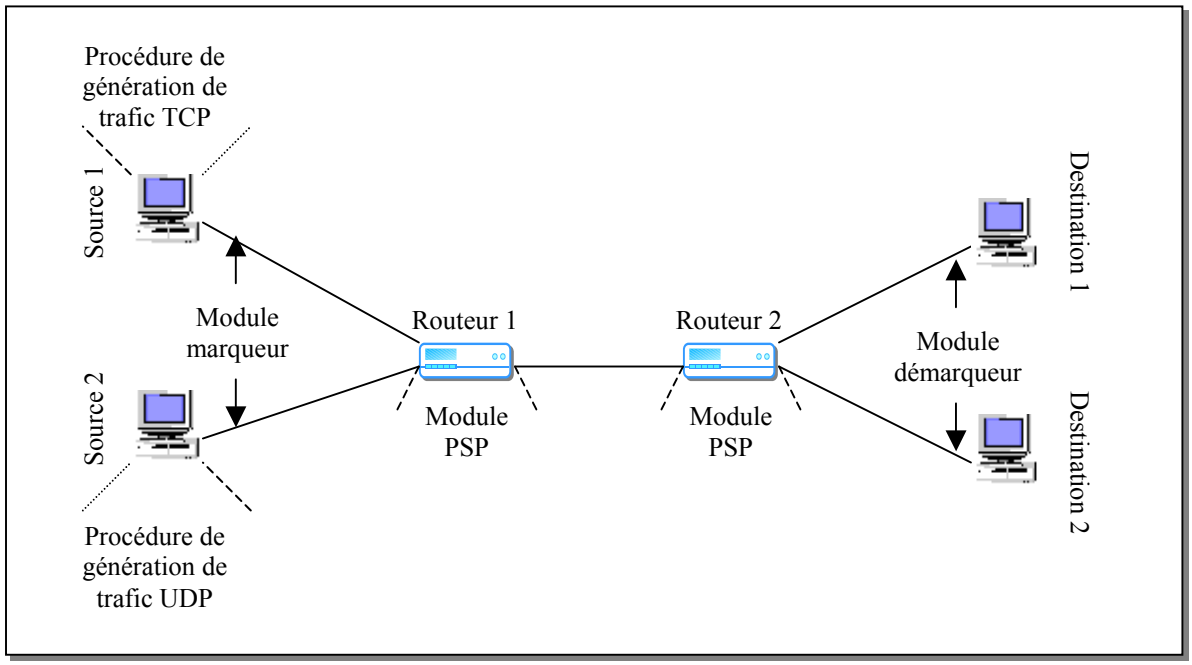


Figure 5.2 : Architecture du réseau simulé

5.3.2 Génération du trafic

Pour nos simulations, nous avons choisi d'étudier le cas de deux sources émettant chacune un type de trafic spécifique. Ainsi, nous avons convenu d'attribuer à la source n°1 un type de trafic basé sur le protocole TCP transportant des applications non temps-réel et à la source n°2 un trafic basé sur le protocole UDP. A chacune de ces sources est attribué un récepteur spécifique : le récepteur n°1 est dédié à recevoir uniquement le trafic provenant de la source n°1 ; le récepteur n°2 est destiné à recevoir l'information envoyée par la source n°2.

5.3.2.1 Trafic TCP

Le module de génération du trafic TCP écrit en Tcl/Tk permet de transporter des données au-dessus du protocole TCP. L'application que nous avons choisie d'implémenter est du type transfert de fichier, soit FTP.
La taille des paquets est variable selon les cas de figures que nous avons simulés : nous indiquerons ultérieurement, dans la section des interprétations des résultats, les différentes tailles de paquets que nous avons utilisées. Par ailleurs, pour tenter de rapprocher au mieux les conditions de simulations à la réalité qui se passe sur les réseaux, nous avons proposé de générer du trafic de manière aléatoire, c'est-à-dire avec des temps d'inter-arrivées de flots (et non de paquets) aléatoire selon une loi exponentielle. Il n'existe pas de temps d'inter-arrivées aléatoire au niveau des paquets dans le cas d'un tel trafic car l'ensemble est régi par les lois de TCP et par conséquent les paquets sont acheminés selon les règles de fenêtrage de TCP. Les paramètres qui caractérisent ce module sont la taille des paquets, le temps de départ d'envoi du premier paquet et le temps d'arrêt d'envoi des paquets d'un même flot. Pour les simulations, nous avons choisi pour chaque scénario, un envoi de 400 paquets par flot.

5.3.2.2 Trafic UDP

Au-dessus du protocole UDP, nous avons choisi d'acheminer un trafic en temps-réel, pas nécessairement ferme en garanties de délai. Pour cela, nous avons opté pour le choix d'un trafic semblable à une application audio sur Internet. La modélisation d'un tel trafic se base sur la génération de rafales de paquets successifs, suivis par un temps de latence. Dans *ns-2*, un tel trafic est construit à partir du module *Exponential On-Off* qui offre à ce propos les caractéristiques attendues. Une source de ce type possède quatre paramètres que nous avons initialisés comme suit :

- « PacketSize » est la taille constante des paquets générés par flot. Selon les scénarii adoptés, nous avons fait varier la taille des paquets.
- « burst_time » est le temps moyen pendant lequel le générateur est "on" et envoie les rafales de paquets. Nous l'avons fixé à 500ms.
- « idle_time » est le temps moyen pendant lequel le générateur est "off", c'est-à-dire le temps pendant lequel la source arrête d'émettre les paquets appartenant à un même flot. Il est fixé à 100ms.
- « rate » est le débit auquel sont envoyés les paquets pendant les périodes "on" , soit 2Mbps.

La génération du trafic UDP est généré de manière aléatoire, exponentiellement, de manière à avoir un scénario de simulation proche de la réalité, c'est-à-dire que plusieurs utilisateurs émettent des requêtes d'applications temps-réel à des intervalles aléatoires.

5.3.3 Marqueurs et démarqueurs

Les marqueurs et démarqueurs sont des éléments clés de notre architecture. En effet, c'est grâce à ceux-ci que s'effectuent la classification des paquets, le calcul du délai de bout en bout, et la distribution des paquets vers les destinations adéquates.

5.3.3.1 Marqueur

Deux fonctions sont définies dans les marqueurs :

- tout d'abord, la fonction de marquage des paquets pour différencier les paquets appartenant aux différents flots. Pour nos simulations, nous avons convenu d'effectuer un marquage déterministe, c'est-à-dire un marquage dont nous avons établi les valeurs comme suit : nous avons choisi de marquer à « 1 » les paquets des flots TCP et à « 2 » les paquets appartenant aux flots UDP ;
- ensuite, la fonction d'estampillage pour favoriser le calcul du délai de bout en bout, assuré par le démarqueur. Chaque paquet d'un flot qui s'apprête à quitter la file de sortie de la source est marqué au niveau de son en-tête par le temps à cet instant.

5.3.3.2 Démarqueur

Placé en aval du réseau, le démarqueur achemine tout d'abord les paquets vers leur destination. D'autre part, il est conçu pour donner les mesures statistiques relatives au débit, délai et puissance de chaque classe, après traitement des paquets par l'ordonnanceur.

5.3.4 L'ordonnanceur PSP

Le mécanisme d'ordonnancement PSP est intégré au niveau de deux opérations :

- une première relative à la mise dans la file d'attente : Enqueue ;

- une seconde opération relative à la desserte du paquet, c'est-à-dire à la sortie de la file d'attente : `Dequeue`.

Dans la suite, nous décrivons les algorithmes de chaque opération. Pour cela, nous disposons des hypothèses suivantes :

- le type `temps` est un enregistrement qui contient les données temporelles (horloge) ;
- le type `Tpaquet` qui encapsule les données (`donnees`) contenues dans un paquet :

```
Type Tpaquet = enregistrement

            taille : entier
            timestamp : temps
            n : entier
            buffer : donnees

          finenregistrement
```

- une liste représente une file d'attente de paquets FIFO;
- le type `Tclasse` défini par :

```
Type Tclasse = tableau de [1..Nbr_classes] de listes
```

- on dispose d'une fonction `header_list (l:Tliste)` qui permet de renvoyer le premier paquet d'une liste ;
- la procédure `enqueue_list (p:Tpaquet ; l:Tliste)` permet de rajouter un paquet dans une liste ;
- la fonction `dequeue_list (l:Tliste)` permet d'extraire le premier paquet d'une liste.

`Enqueue` est responsable de l'estampillage du paquet au moment de l'entrée dans la file d'attente. Cet estampillage est utilisé pour permettre le calcul du temps d'attente du paquet à l'intérieur de la file.

La procédure `Enqueue` est décrite par l'algorithme suivant :

```
Procedure enqueue (p:Tpaquet , classe:Tclasse)
      t_actuel :temps
      Debut
            p.timestamp← t_actuel
            enqueue list (p,classe[p.n])
```

`Dequeue` est responsable de la décision de service des paquets. C'est à ce niveau que toute la fonctionnalité de l'ordonnanceur est centralisée. La fonction est décrite comme suit :

```
Fonction dequeue (classe:Tclasse , SDP:tableau [1.. Nbr_classes]
d'entiers) : Tpaquet

      p : Tpaquet
      i : entier
      max_priorite : entier
      classe_max_priorite : entier
      prio : entier
      C : entier
      Nbr_classes : entier
      t_actuel : temps
      t_n-1 : tableau [1..Nbr_classes] d'entiers
      l_n-1 : tableau [1..Nbr_classes] d'entiers
      d_n-1 : tableau [1..Nbr_classes] d'entiers



      Debut
            max_priorite ← -1
            Pour i de 1 à Nbr_classes faire
                  delai = t_actuel – p.timestamp
                  p ← header_list(classe[i])
                  prio←SDP[i]*((delai+l_n-1+(p.taille)/C))*
            (t_actuel-t_n-1[i]))/l_n-1[i]
```

`prio` est la valeur de la priorité calculée à partir de la formule de base de l'ordonnanceur PSP, décrite dans le chapitre 4.

5.4 Courbes et Interprétations

Dans cette section, nous exposons divers scénarios d'expériences et présentons les résultats par paramètre de qualité de service visé. Autrement dit, nous rappelons que la problématique de notre recherche est de pouvoir satisfaire une équité de priorité entre des applications à fortes garanties de délai et des applications à fortes garanties de débit ; l'équilibre étant fourni par l'ordonnancement PSP qui met en jeu ces deux métriques. Nous exposons donc nos résultats de la manière suivante : pour chaque paramètre (débit et délai), nous superposons les courbes de chaque classe pour mettre en valeur la différenciation obtenue.

Les hypothèses que nous avons adoptées pour nos expériences sont les suivantes :

- tout d'abord, pour toutes les mesures, nous avons fixé le facteur de différenciation spd_{c1} à 1 et sdp_{c2} à 2 ;
- les files d'attente ont une très grande capacité, ce qui nous ramène à simuler un réseau sans pertes ;
- tous les liens ont une capacité de 25Mbps et un délai nul (pour faciliter les calculs) ;

- chaque expérience a duré 100 secondes, de manière à avoir un scénario relativement proche de la réalité, tout en restant dans l'esprit de simulation.

5.4.1 Scénario n°1

Pour cette première série de simulations, nous avons fixé la taille des paquets à 1024 octets pour chaque type de trafic. Le nombre de flots émis est faible. Les résultats présentés par la figure 5.3 montrent la différenciation de délai entre les classes d'applications. Jusqu'à une charge de réseau d'environ 60%, la différenciation reste faible puis elle apparaît correctement au-delà de cette charge. La courbe de la classe 1 augmente de manière significative par rapport à celle de la seconde classe autour de 90% de charge.

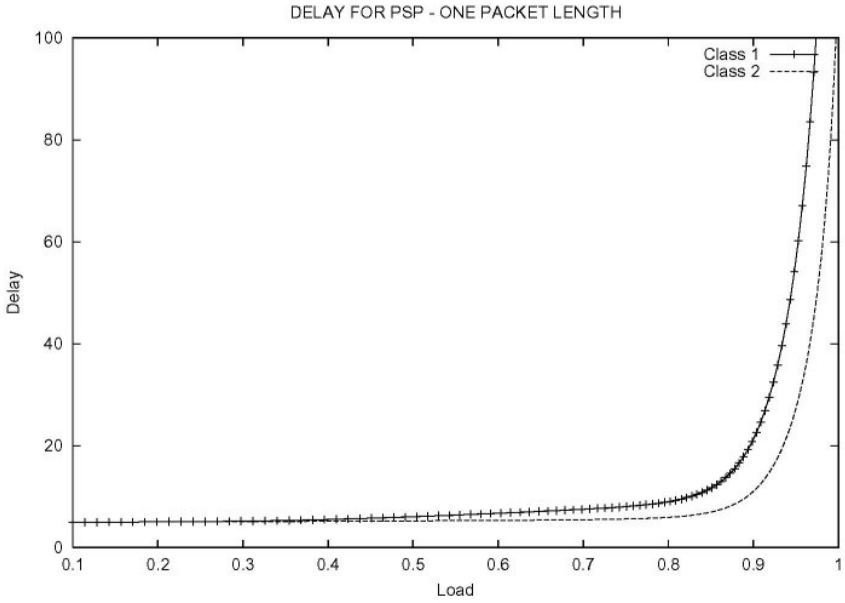


Figure 5.3 : Courbes de délai sous PSP : scénario n°1

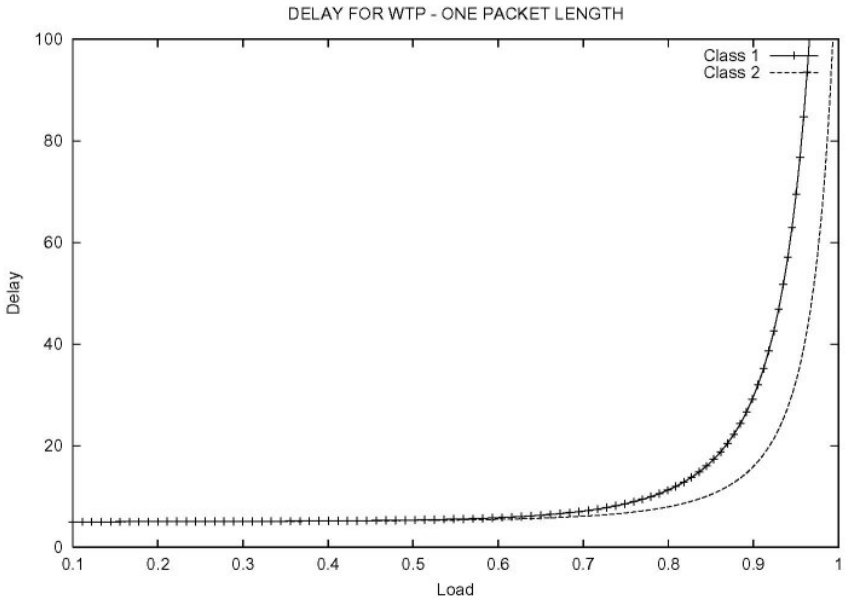


Figure 5.4 : Courbes de délai sous WTP : scénario n°1

La figure 5.5 expose le débit sur les paquets des deux classes. A l'inverse des courbes de délai, la croissance des débits ne se fait pas de manière exponentielle. Toutefois, la différenciation se fait au même niveau de charge du réseau que pour le délai. Par ailleurs, c'est autour de 80% de charge du réseau que la différenciation atteint son maximum et reste quasiment constante jusqu'à la valeur maximale de la charge du réseau. Nous pouvons noter la différence entre les deux classes : les mesures de débit pour la première classe restent toujours au-dessus de celles de la seconde classe.

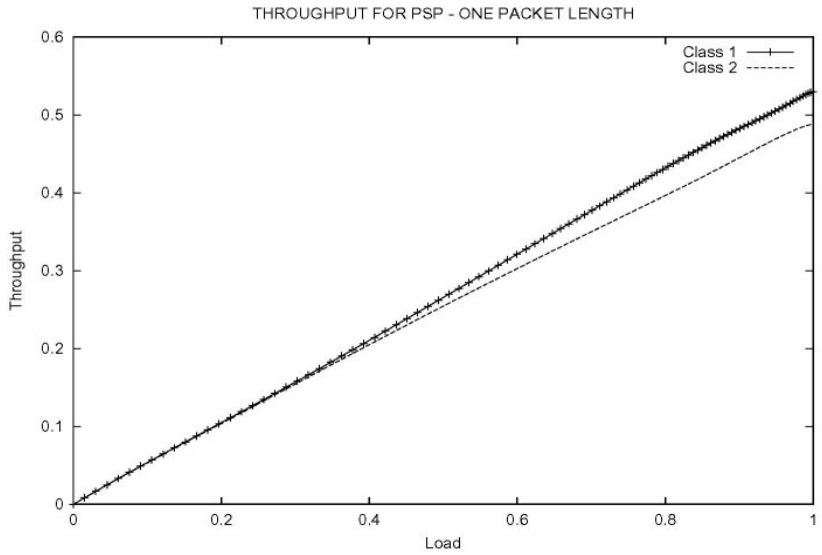


Figure 5.5 : Courbes de débit sous PSP : scénario n°1

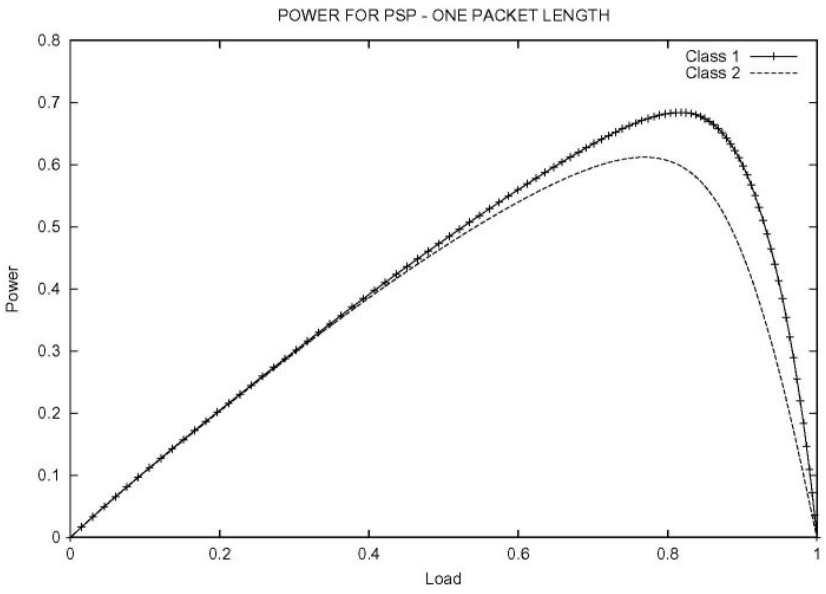


Figure 5.6 : Courbes de puissance sous PSP : scénario n°1

La figure 5.6 présente enfin les courbes de puissance de chaque classe pour donner une représentation du résultat du rapport débit/délai.
De ces figures, nous pouvons déduire les conclusions suivantes :

- en premier lieu, au niveau de la différenciation : le seuil de différenciation est raisonnable. En effet, comparée à la courbe de délai utilisant le mécanisme

d'ordonnancement WTP (figure 5.4), nous remarquons que la différenciation s'effectue à un niveau de charge de réseau inférieure à celle présentée par l'illustration 5.4. WTP a justement pour inconvénient de n'agir correctement qu'à un niveau de charge relativement important [36].

- dans un second temps nous pouvons noter l'équilibre entre le débit et le délai des classes. En effet, nous remarquons que la classe qui présente un délai plus élevée que sa concurrente présente un débit plus important que cette même dernière, et inversement. Par conséquent, nous pouvons remarquer cette équité entre le débit et le délai des classes de trafic, favorisé par l'ordonnancement PSP.

5.4.2 Scénario n°2

La seconde série de simulations s'est effectuée pour un faible nombre de flots dont les paquets sont fixés à 1024 et 2048 octets. Ainsi, un trafic TCP ou UDP peut avoir un ou plusieurs flots avec des paquets à 1024 octets ou à 2048 octets. Nous étudions alors l'effet de la taille des paquets sur l'algorithme PSP et l'effet sur la différenciation des classes de trafic.

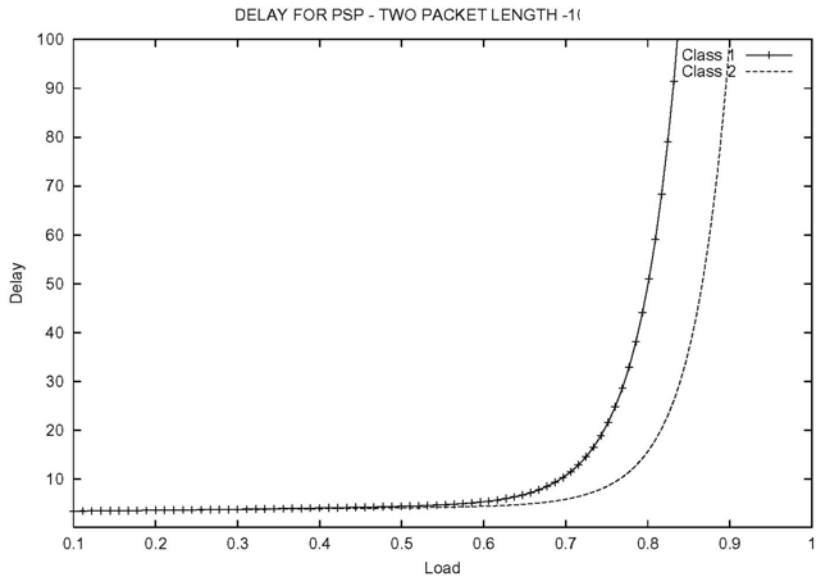


Figure 5.7 : Courbes de délai sous PSP : scénario n°2

Nous pouvons remarquer dans un premier temps l'instant de différenciation : peu avant 70% de la charge du réseau, la différenciation s'effectue sur les classes TCP et UDP. En utilisant le mécanisme WTP, la différenciation s'effectue plutôt au-delà de 75% de charge. D'autre part, nous pouvons constater un augmentation plus rapide des courbes de délai, relativement aux courbes de la figure 5.3, utilisant une seule taille de paquets pour tous les flots. Ce phénomène est notamment dû à l'algorithme PSP qui met en jeu la taille des paquets pour le calcul de la priorité. Cet événement n'est pas constaté pour la figure 5.8 qui montre les courbes de délai des deux classes simulées dans les mêmes conditions mais utilisant l'algorithme d'ordonnancement WTP. Ceci résulte du fait que WTP n'utilise en aucun cas la taille des paquets. Enfin, l'écart de différenciation est plus important lorsque la taille des paquets est diversifiée, notamment si nous utilisons des tailles assez importantes. Toutefois, pour la figure 5.7, nous notons toujours la supériorité de la courbe de la classe 1 par rapport à celle de la classe 2, fait déjà constaté au niveau de la figure 5.3, et visé pour la validation de notre prototype.

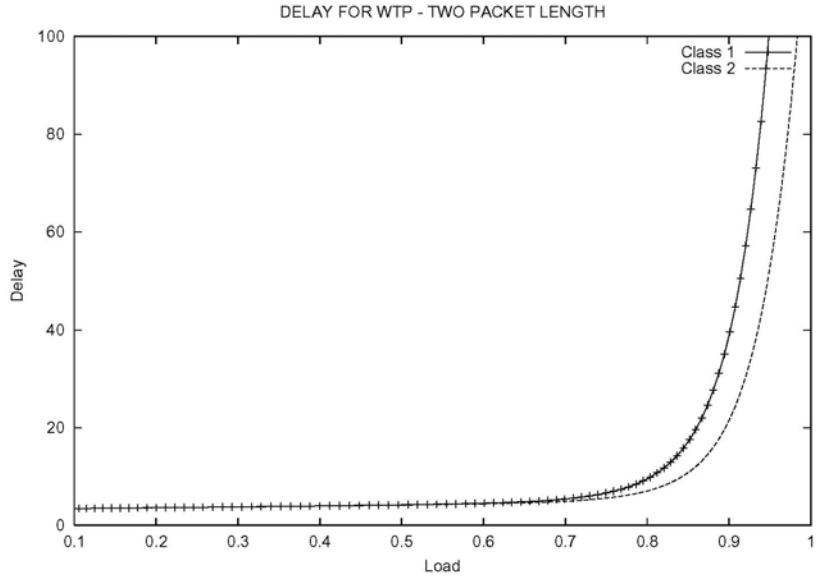


Figure 5.8 : Courbes de délai sous WTP : scénario n°2

La figure 5.9 illustre le débit par classe de trafic, en utilisant le mécanisme d'ordonnancement PSP. La différenciation est visible pour une charge du réseau se situant autour de 40%. Entre 40% et 55%, la différenciation est faible puis elle s'accroît à partir de 60%. Par rapport à la figure 5.5 qui expose les mêmes courbes pour une seule taille de paquets, la différenciation de débit est légèrement différente. Ceci s'explique par la faible présence de flots, malgré la différence de taille des paquets.

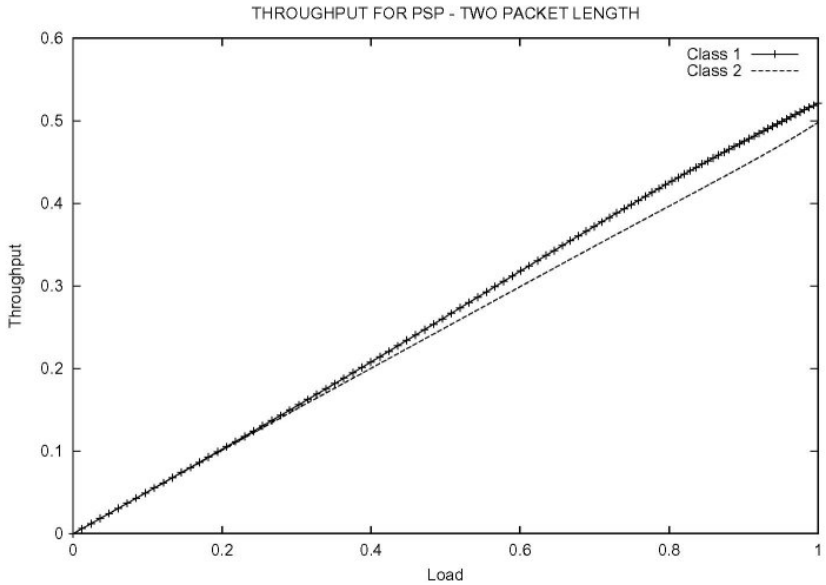


Figure 5.9 : Courbes de débit sous PSP : scénario n°2

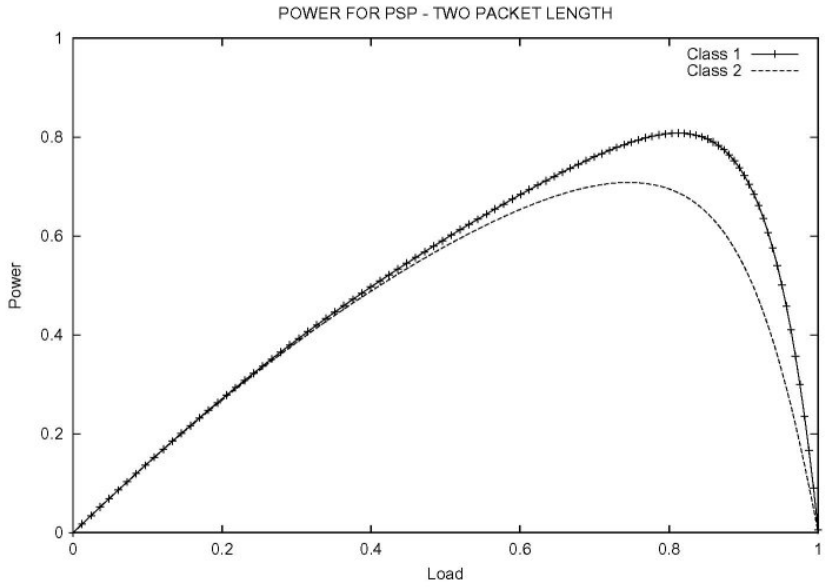


Figure 5.10 : Courbes de puissance sous PSP : scénario n°2

5.4.3 Scénario n°3

La troisième série d'expériences a consisté à mettre en jeu trois tailles de paquets pour un nombre aussi restreint de flots TCP et UDP. Nous avons choisi de rajouter une taille de paquets inférieure à celles précédemment mises en jeu. La taille rajoutée est donc 512 octets.

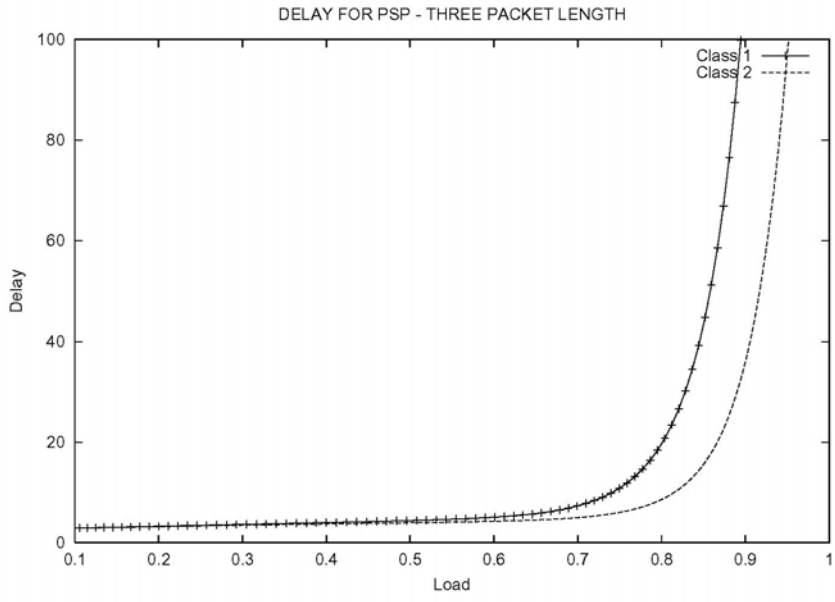


Figure 5.11 : Courbes de délai sous PSP : scénario n°3

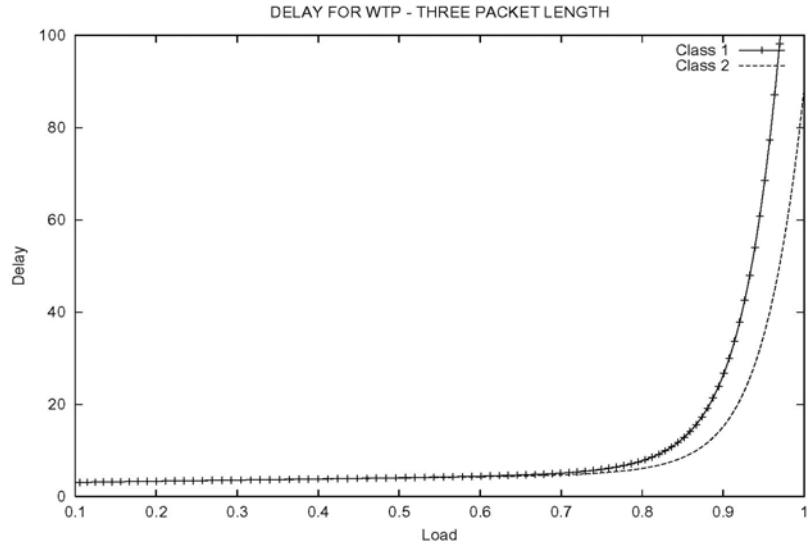


Figure 5.12 : Courbes de délai sous WTP : scénario n°3

Les figures 5.11 et 5.12 présentent les résultats des délais pour chaque classe, tout d'abord en utilisant l'algorithme PSP, puis WTP. Celles-ci montrent en premier lieu la différenciation visée du point de vue délai, à savoir la supériorité des délais de la classe 1 par rapport à celles de la classe 2 dont les flots sont moins demandeurs en délai mais plus fermes en débit. D'autre part, nous remarquons toujours l'efficacité de PSP devant WTP quant au seuil de différenciation. Enfin, l'écart de différenciation reste approximativement le même que celui de la seconde série de simulations puisque la taille de paquet rajoutée n'est pas très importante par rapport aux tailles déjà existantes, d'une part ; d'autre part, le tirage des tailles se faisant aléatoirement, il est probable que peu de flots ayant des paquets à 250 octets aient été envoyés.

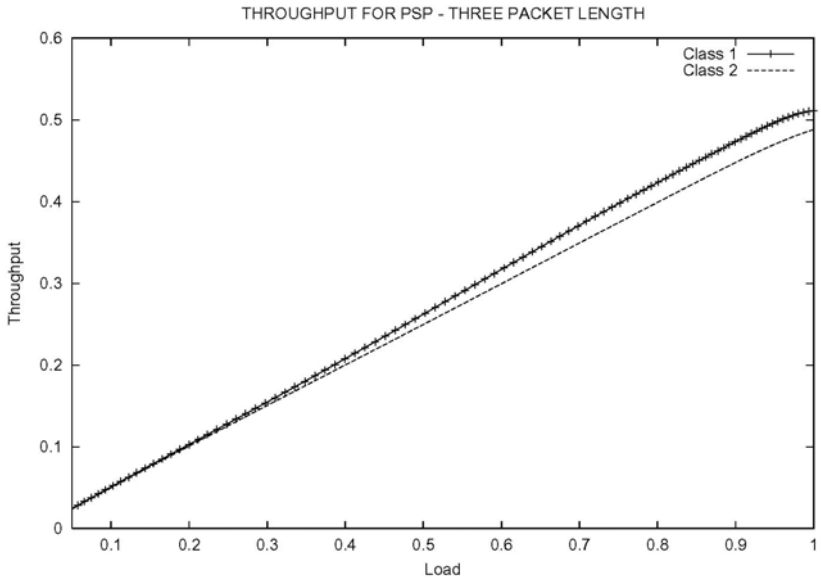


Figure 5.13 : Courbes de débit sous PSP : scénario n°3

Le débit des deux classes, illustré par la figure 5.13, montre une faible différenciation entre 40% et 55% de charge du réseau ; au-delà de cette dernière valeur, la différenciation est plus convaincante. Le réseau n'atteignant pas la saturation, les débits n'adoptent pas de forme

linéaire constante. Par contre, nous pouvons remarquer une légère stabilité autour de 100% de charge du réseau. Cette allure aurait été précoce si la taille des files d'attente était dimensionnée selon une réalité existentielle.

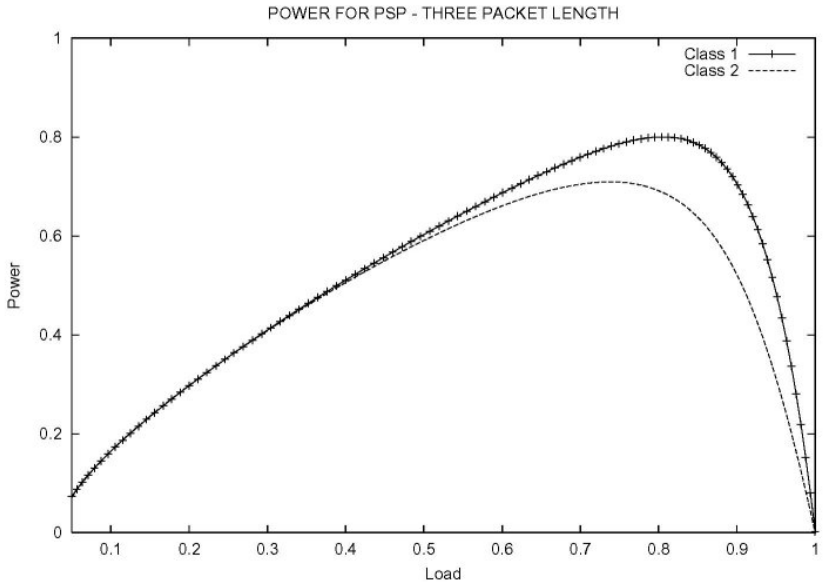


Figure 5.14 : Courbes de puissance sous PSP : scénario n°3

5.4.4 Scénario n°4

Nous avons choisi, pour ce cas de figure, de garder les mêmes tailles de paquets que celles utilisées pour le troisième scénario, mais nous avons augmenté le nombre de flots TCP et UDP. Le temps d'arrivée des flots se fait de manière exponentielle et aléatoire dans le temps, avec des intervalles d'inter arrivées relativement petits pour augmenter le nombre de flots sur le réseau.

Les figures 5.15 et 5.16 représentent les mesures de délai moyen des paquets de la classe 1 et de la classe 2, respectivement en utilisant le mécanisme d'ordonnancement PSP, puis WTP.
Une première constatation est visible pour les résultats sous PSP : la différenciation de délai est remarquable dès une faible charge du réseau (à partir de 10%), et reste constamment faible jusqu'à 65% de charge où la différenciation atteint une plus grande valeur. Sous WTP, la différenciation ne s'effectue que tardivement, pour environ 70% de charge.
D'autre part, nous remarquons un écart de différenciation moindre que les scénarii précédents. Ce phénomène est dû au nombre plus important de flots qui traversent le réseau et est mis en évidence, par ailleurs, par l'augmentation plus rapide des délais.

Cette série d'expériences nous montre dans un premier temps, l'efficacité de l'ordonnancement PSP face à WTP pour la différenciation du point de vue délai. Nous pouvons constater qu'au moyen d'un ordonnanceur PSP, il n'est pas nécessaire d'atteindre une charge importante sur le réseau pour bénéficier de la différenciation.

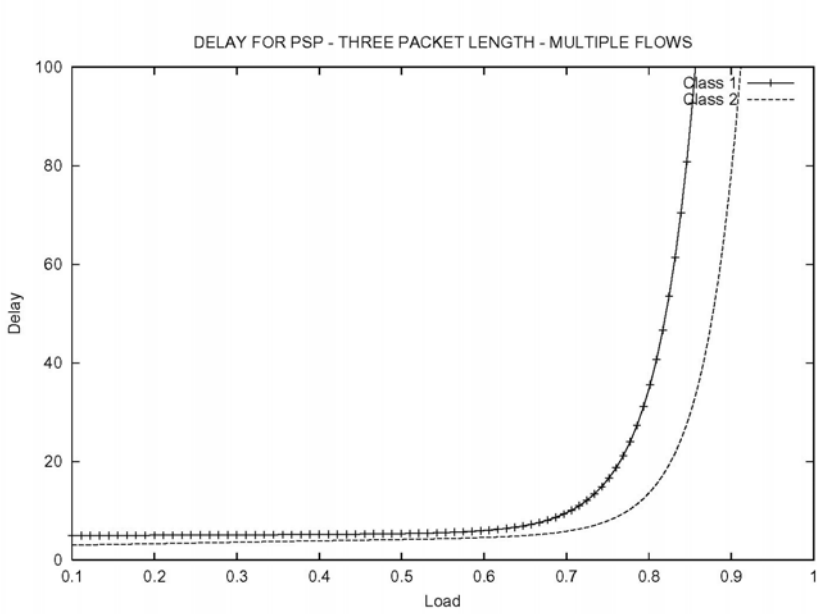


Figure 5.15 : Courbes de délai sous PSP : scénario n°4

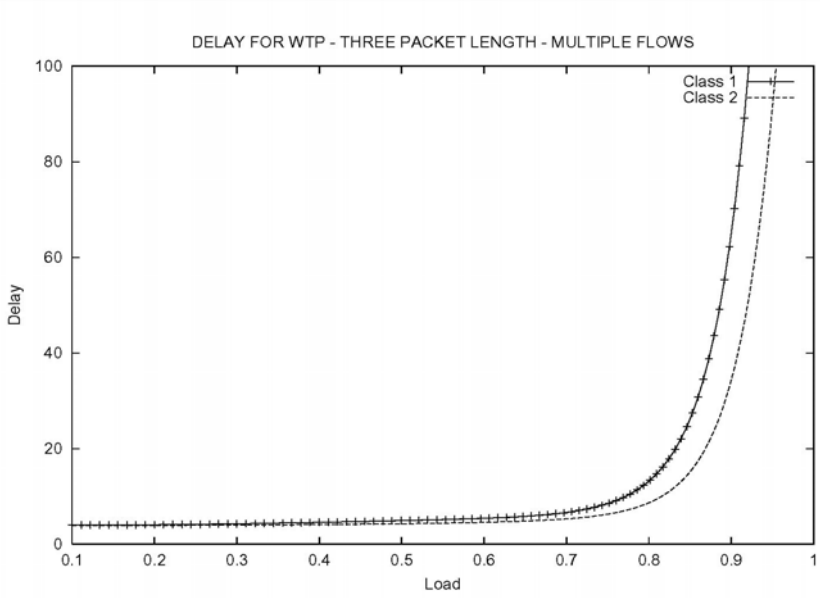


Figure 5.16 :Courbes de délai sous WTP : scénario n°4

La figure 5.17 expose les différents débits des deux classes de trafic TCP et UDP. La différenciation est nette sur la figure et c'est effectivement autour de 10% de charge qu'elle s'accroît et que l'écart de différenciation augmente jusqu'à environ 90% pour garder un écart constant. Par ailleurs, la première classe montre un débit plus important que la première, ce qui valide l'idée relative aux flots élastiques qui utilisent une bande passante plus importante que les flots non-élastiques.
Le résultat visé est nettement plus perceptible pour cette série d'expériences, puisque nous disposons d'un nombre important d'applications sur le réseau, durant un intervalle de temps significatif, comparable à un intervalle d'utilisation des réseaux dans la réalité. Une application à fort débit dispose de moins de délai (temps d'attente relativement important) et inversement.

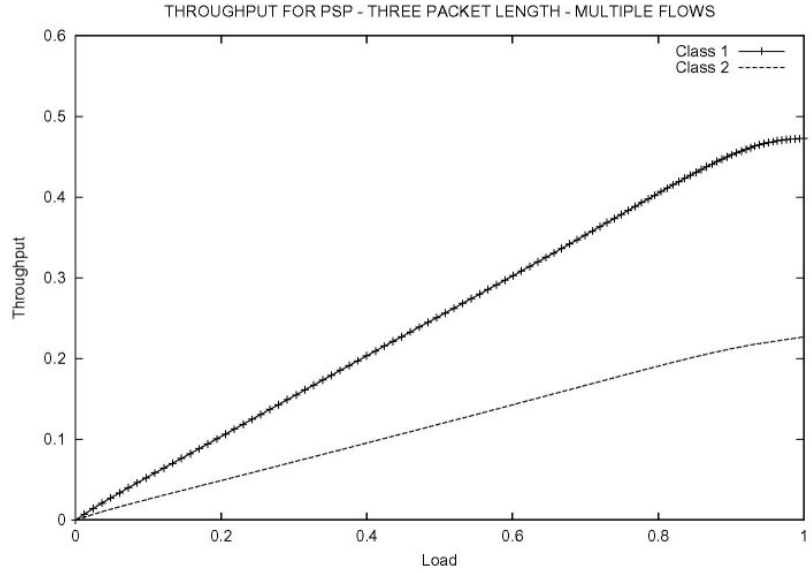


Figure 5.17 : Courbes de débit sous PSP : scénario n°4

La figure 5.18 est l'illustration de la puissance par classe de trafic. Nous observons la nette différence d'écart entre les deux courbes, particulièrement pour la classe n°1 qui voit sa puissance quasiment divisée par 2.

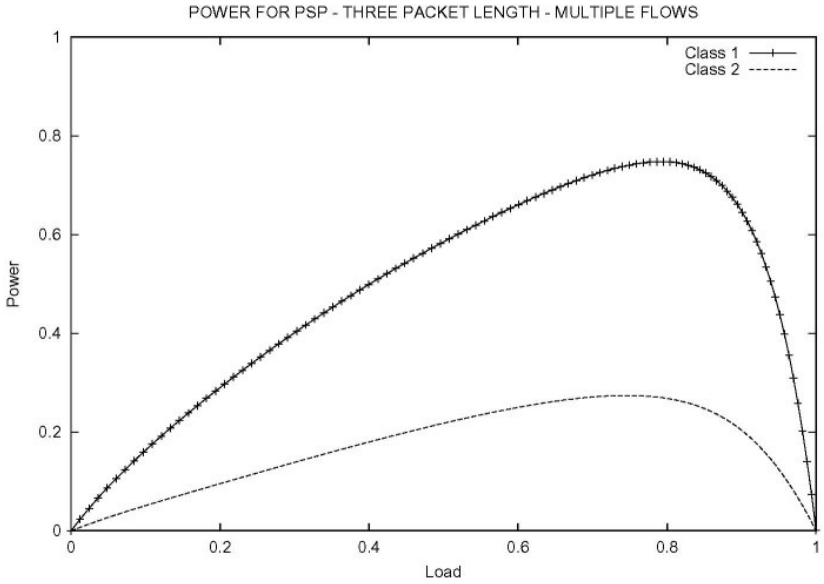


Figure 5.18 : Courbes de puissance sous PSP : scénario n°4

5.4.5 Scénario n°5

Pour ce scénario, nous avons opté pour un choix de figures assez réaliste : par cela, nous entendons que les flots qui circulent soit générés de manière non déterministe avec des inter arrivées relativement courtes, comme pour les expériences précédentes ; que le nombre d'applications qui circule soit important ; et enfin, que les tailles des paquets soient fortement diversifiées. Nous avons par conséquent attribué les différentes tailles suivantes : 50, 100, 210, 512, 1024, 2048, 4000 octets, et les flots adoptent les tailles de leurs paquets de manière absolument aléatoire.

Comme pour les précédentes expériences, nous commençons par présenter les courbes de délais des paquets appartenant aux flots TCP et UDP, tout d'abord en utilisant l'ordonnanceur PSP, puis en adoptant l'algorithme WTP. Les figures 5.19 et 5.20 sont les illustrations de ces résultats.

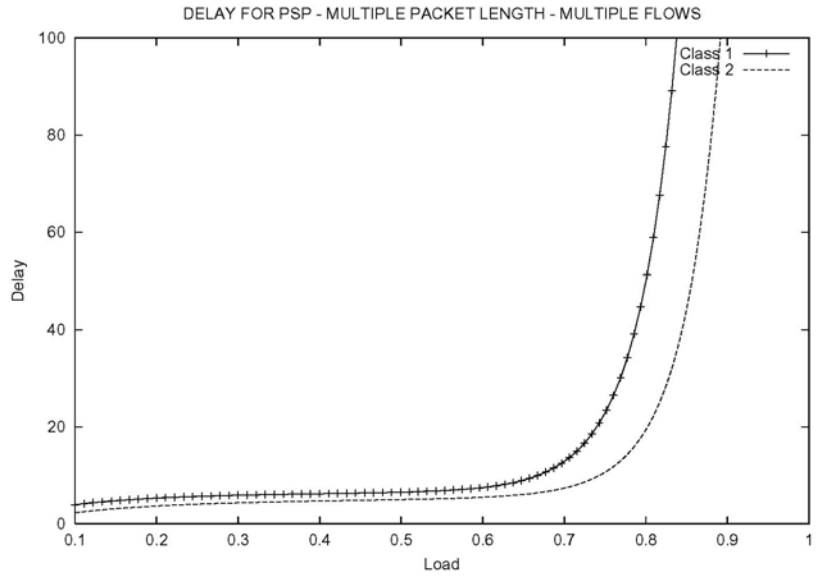


Figure 5.19 : Courbes de délai sous PSP : scénario n°5

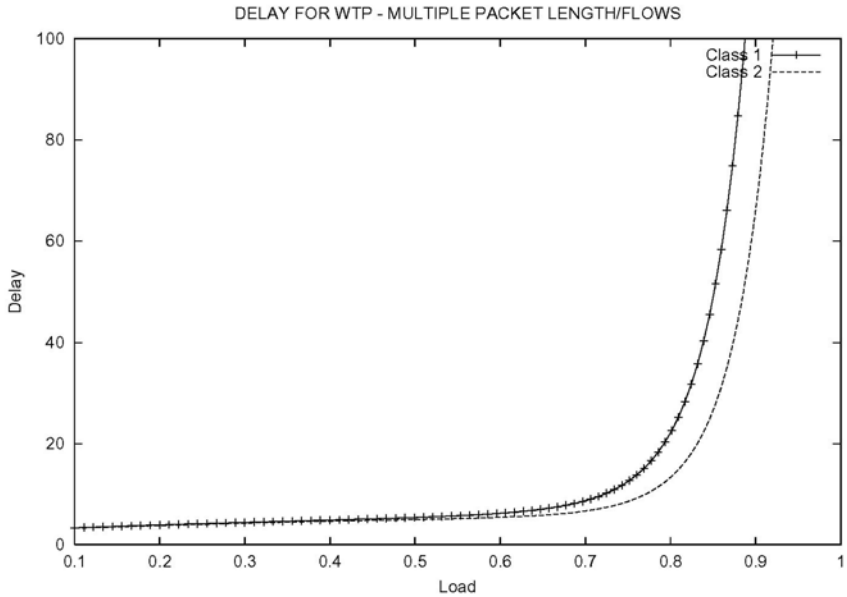


Figure 5.20 : Courbes de délai sous WTP : scénario n°5

La première remarque concerne le niveau de charge pour lequel la différenciation commence à s'effectuer : tout comme le scénario précédent, nous observons une différenciation précoce, c'est-à-dire pour une charge d'environ 10%. Certes, celle-ci est faible, mais elle est apparente. A partir de 55% de charge, elle commence à prendre de l'effet et s'accroît jusqu'à environ 80% pour garder un écart quasi-constant. Ces observations ne sont pas validées pour l'utilisation de l'algorithme WTP, puisque la figure 5.20 montre une faible différenciation à

partir de 65% de charge ; d'autre part, l'écart de différenciation est plus significatif pour notre algorithme que pour WTP, ce qui montre son efficacité.

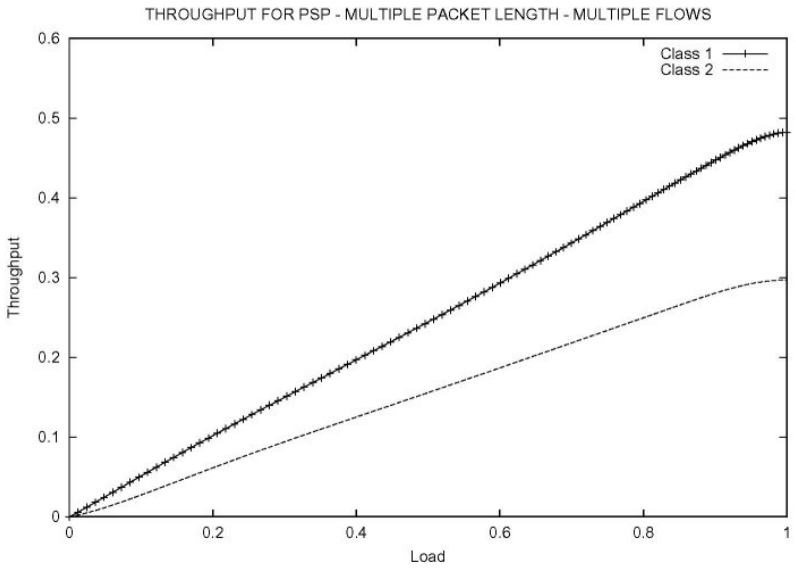


Figure 5.21 : Courbes de débit sous PSP : scénario n°5

La figure 5.21 montre la différenciation des classes du point de vue débit. Nous observons le net écart qui se produit à mesure que la charge du réseau augmente. Le débit de la première classe reste toujours supérieur à celui de la seconde classe. Ce comportement confirme l'efficacité d'un tel procédé à base de PSP. Cette différenciation peut aussi être observée sous l'angle de la puissance par classe, montrée par la figure 5.22.

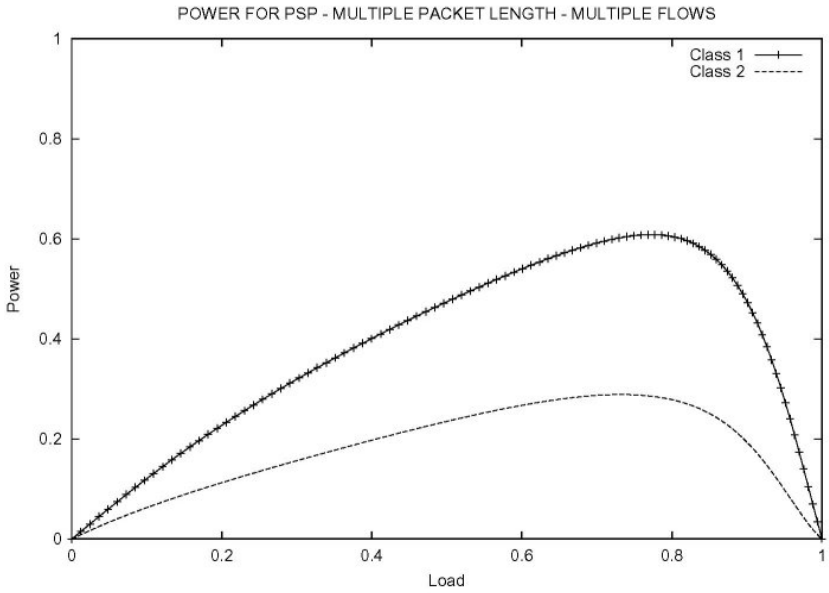


Figure 5.22 : Courbes de puissance sous PSP : scénario n°5

5.4.6 Scénario n°6

La série de simulations que nous présentons dans cette section met en jeu un nombre de flots borné, TCP et UDP, pour lesquels la taille des paquets est respectivement fixée à 1500 et 512

octets. Les paquets des flots élastiques sont ainsi supérieurs à ceux des flots non-élastiques. La figure 5.23 met en évidence le délai mesuré pour chaque classe, sous l'action de l'ordonnanceur PSP. Tout d'abord, une très faible différenciation entre la classe 1 représentant la classe élastique et la classe 2 est remarquable autour de 40% de charge. Cette différenciation s'accroît à mesure que le réseau se trouve chargé et celle-ci est nettement visible dès 55% de charge.
Ces observations ne sont pas mises en évidence pour la figure 5.24 qui présente, sous WTP, les délais pour chaque classe. Par conséquent, nous pouvons déduire que la différenciation sur le délai est plus précoce en utilisant l'ordonnancement PSP qu'en utilisant la politique WTP.

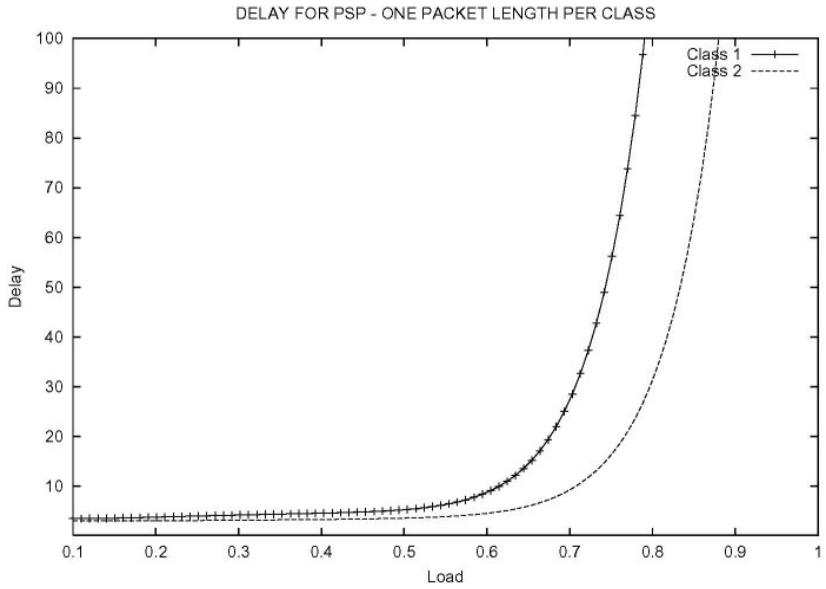


Figure 5.23 : Courbes de délai sous PSP : scénario n°6

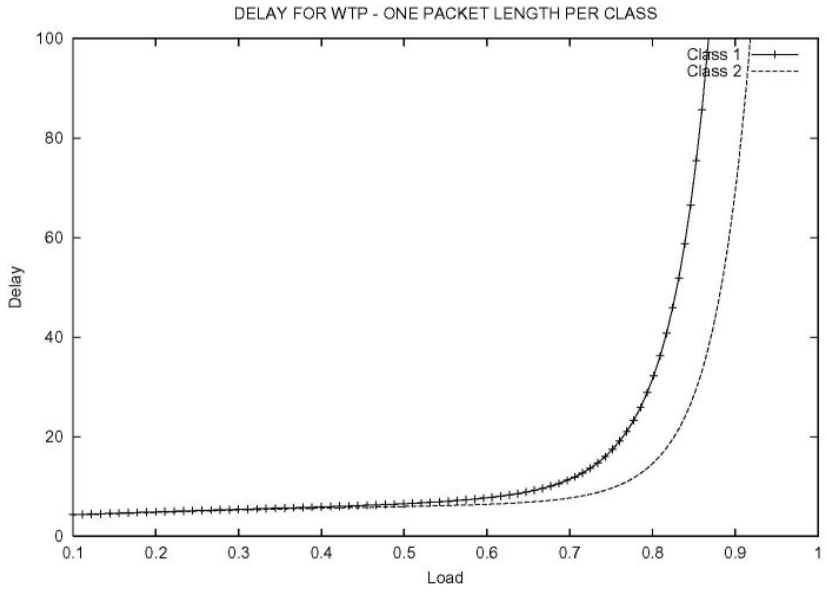


Figure 5.24 : Courbes de délai sous WTP : scénario n°6

La figure 5.25 présente le débit des paquets de chaque classe lorsque le réseau est soumis à la politique d'ordonnancement PSP. Lorsque la taille des paquets pour les flots est déterministe, nous remarquons une nette différence de débit entre les deux classes. D'autre part, nous

notons la supériorité du débit de la classe 1 par rapport à celle de la seconde classe. Ceci traduit et confirme en effet les besoins respectifs des deux classes en terme de débit : la classe temps-réel qui dispose (et qui a besoin) d'un débit moindre que celui requis par la classe élastique.

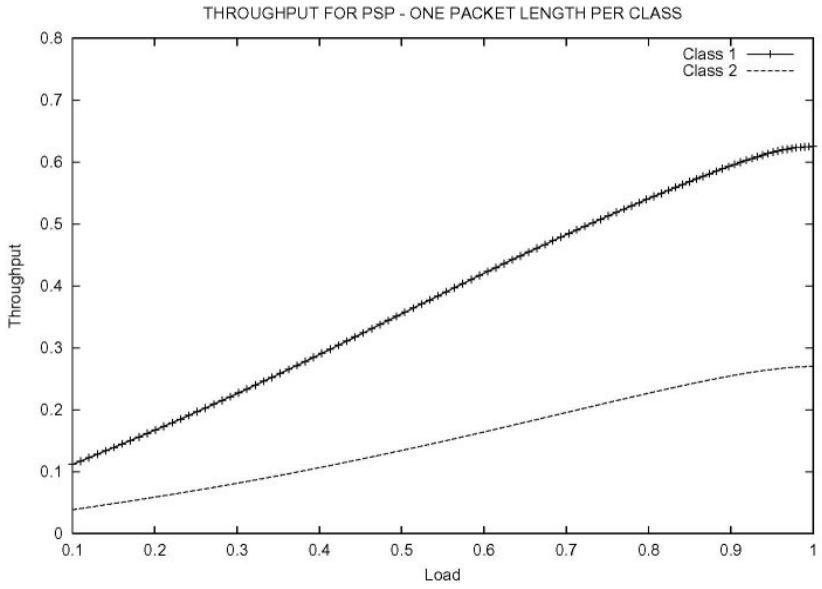


Figure 5.25 : Courbes de débit sous PSP : scénario n°6

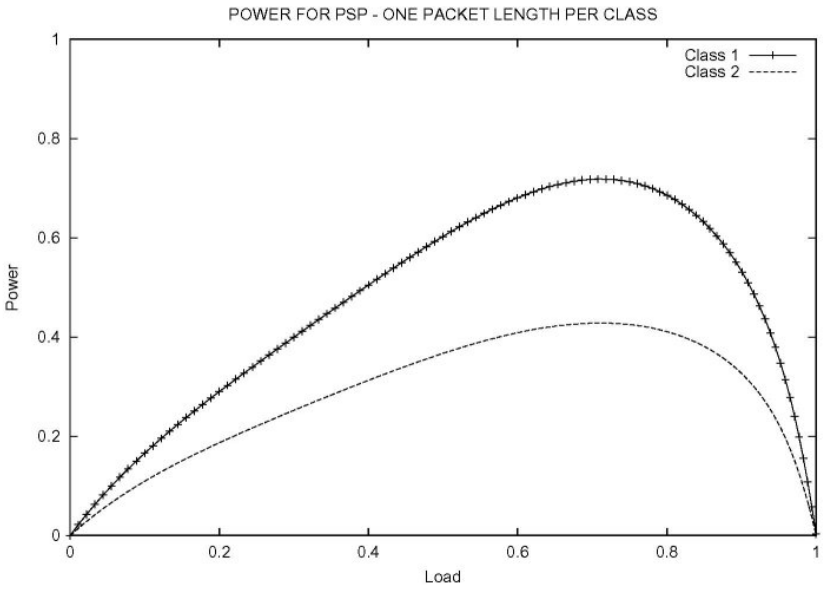


Figure 5.26 : Courbes de puissance sous PSP : scénario n°6

5.4.7 Scénario n°7

Les simulations réalisées pour ce scénario met en jeu un nombre important de flots TCP dont les paquets sont fixés à 1500 octets, en compétition avec des flots UDP dont la taille des paquets est égale à 512 octets. Les flots sont générés de manière aléatoire.
La figure 5.27 expose les résultats obtenus pour les délais mesurés pour les classes TCP et UDP en utilisant l'ordonnancement PSP.

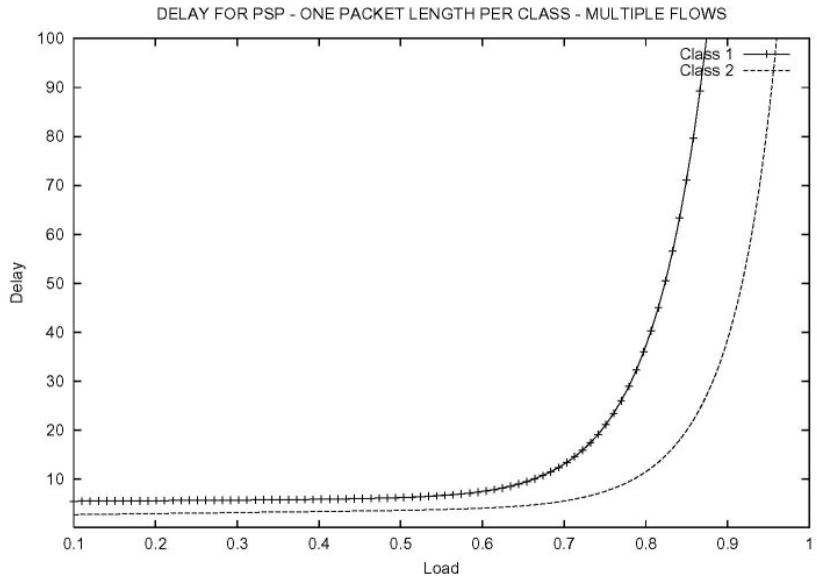


Figure 5.27 : Courbes de délai sous PSP : scénario n°7

Les observations notées sont les suivantes :

- dans un premier temps, nous pouvons remarquer une différenciation notable pour un réseau faiblement chargé ; ce phénomène est dû au nombre important de flots concurrents.
- par ailleurs, si nous comparons cette figure avec la figure 5.19, nous pouvons observer une différence entre les écarts de différenciation. La différenciation est plus nette pour la figure 5.27 du fait que les paquets ne sont pas distribués dans ce dernier cas de figure pour les flots de manière heuristique. Par conséquent, nous pouvons conclure que la différenciation est d'autant plus performante que la taille des paquets est fixée de manière déterministe.

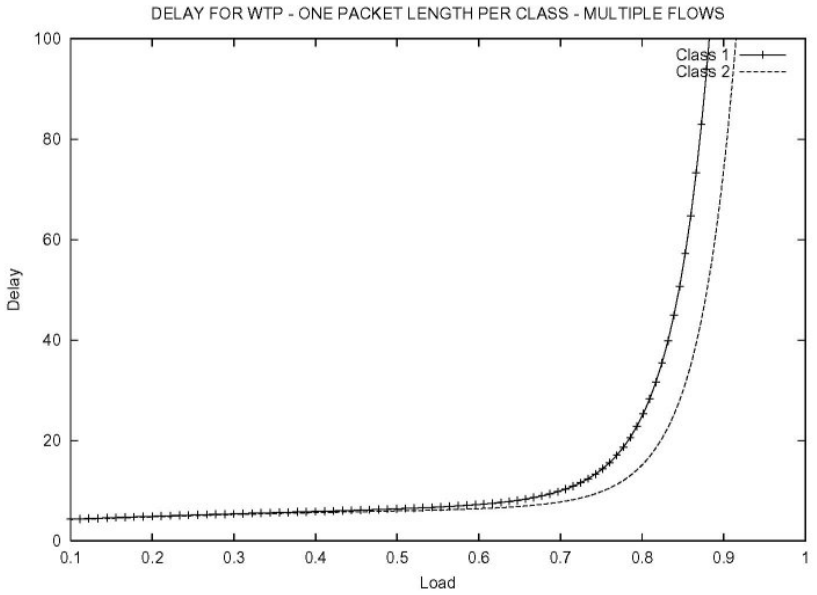


Figure 5.28 : Courbes de délai sous WTP : scénario n°7

La figure 5.28 montre la différenciation du délai sous WTP. Il est à noter visiblement que cette figure est tout à fait comparable avec la figure 5.20 pour laquelle les paquets des classes de trafic sont distribués aléatoirement pour chaque flot. Nous déduisons de cette comparaison que WTP ne tient en aucun cas compte de la taille des paquets et par conséquent n'effectue pas de différenciation comme le montre l'algorithme PSP.

Les courbes présentées par la figure 5.29 exposent le débit de chaque classe de trafic. La différenciation est remarquable, mais du fait du nombre important de flots qui circulent, la différenciation est quelque peu moindre par rapport à la figure 5.25. De même, les débits représentés présentent des valeurs plus faibles que la figure 5.25, toujours du fait de la présence d'un nombre important de flots sur le réseau.

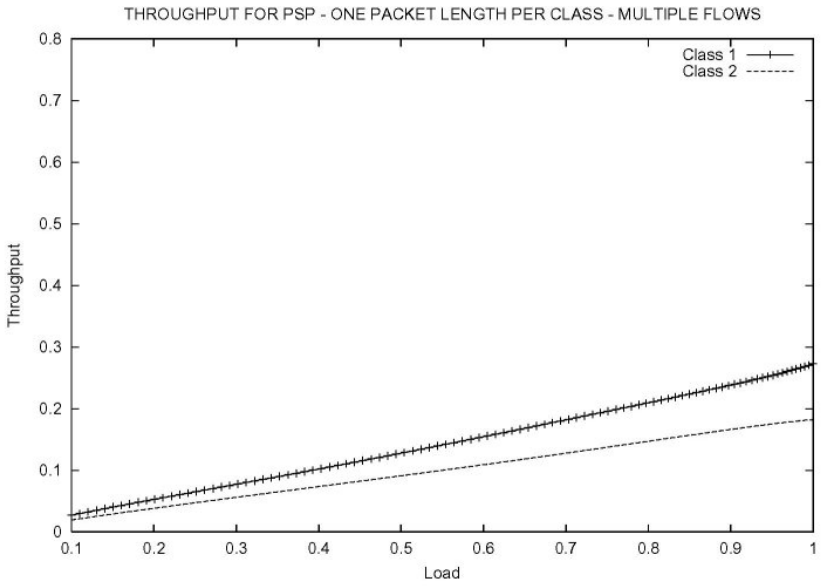


Figure 5.29 : Courbes de débit sous PSP : scénario n°7

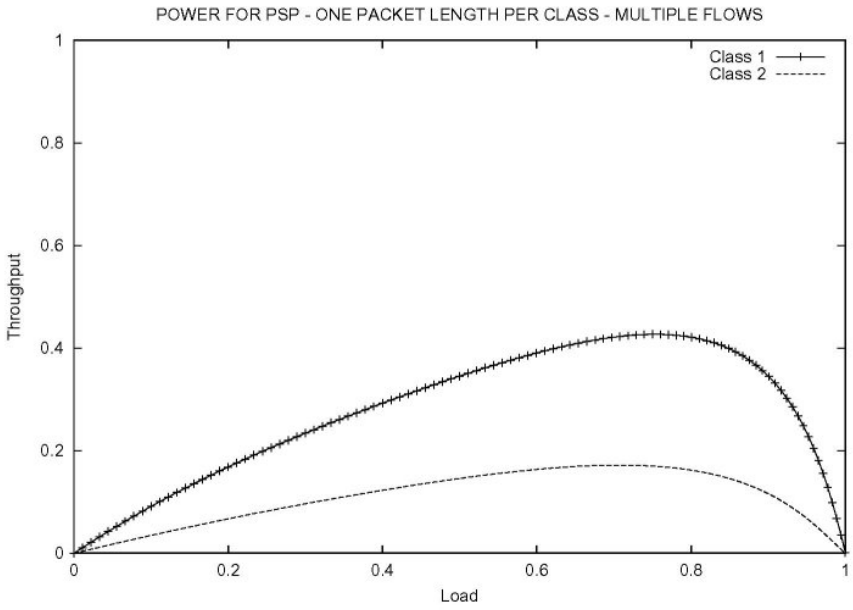


Figure 5.30 : Courbes de puissance sous PSP : scénario n°7

La puissance des classes présentée par la figure 5.30 montre parfaitement le rapport entre le débit et le délai des classes de trafic. Les puissances sont quasi divisées par deux, toujours du fait du nombre de flots concurrents. Par rapport à la figure 5.22, nous pouvons distinguer une légère différence entre les puissances respectives mesurées. Ce résultat découle de la distribution de la taille des paquets pour les flots (déterministe pour le cas de figure actuel ; heuristique pour le cas de la figure 5.22)

5.4.8 Scénario n°8

Le dernier scénario met en compétition plusieurs flots dont la taille des paquets est respectivement fixée à 1500 octets pour les flots de type TCP et 64 octets pour les flots de type UDP. Nous avons augmenté l'écart entre les tailles des paquets pour observer l'effet sur la différenciation proportionnelle.

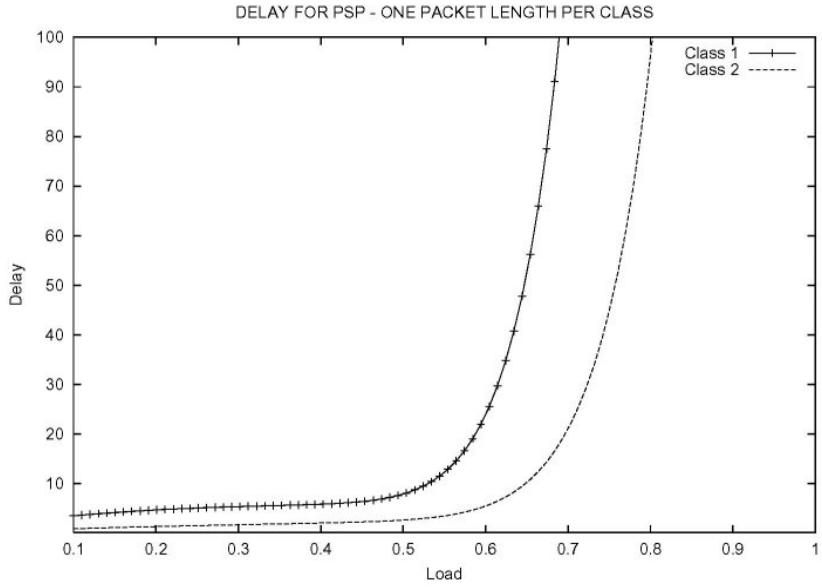


Figure 5.31 : Courbes de délai sous PSP : scénario n°8

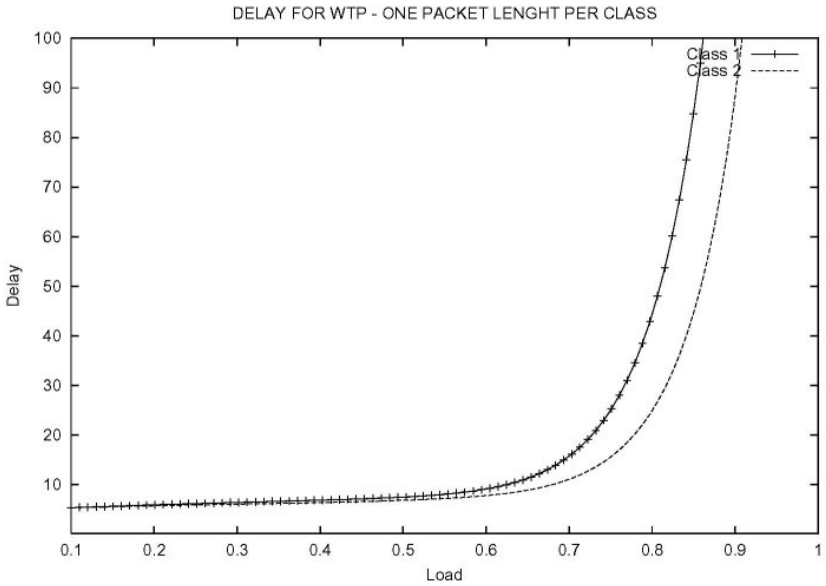


Figure 5.32 : Courbes de délai sous WTP : scénario n°8

Les figures 5.31 et 5.32 montrent respectivement les résultats temporels sous PSP et sous WTP pour le cas de figure exposé dans ce scénario.
Nous notons la différenciation remarquable pour un faible taux de charge du réseau dans le cas où l'ordonnancement utilisé est PSP, ce qui n'apparaît pas pour l'ordonnancement WTP. D'autre part, en comparant la figure 5.31 avec la figure 5.27 pour laquelle la différence entre les tailles des paquets est moindre, nous remarquons une meilleure différenciation pour le cas de figure du scénario n°8. Par conséquent, nous pouvons conclure que plus la taille des paquets est faible, plus la différenciation est importante.

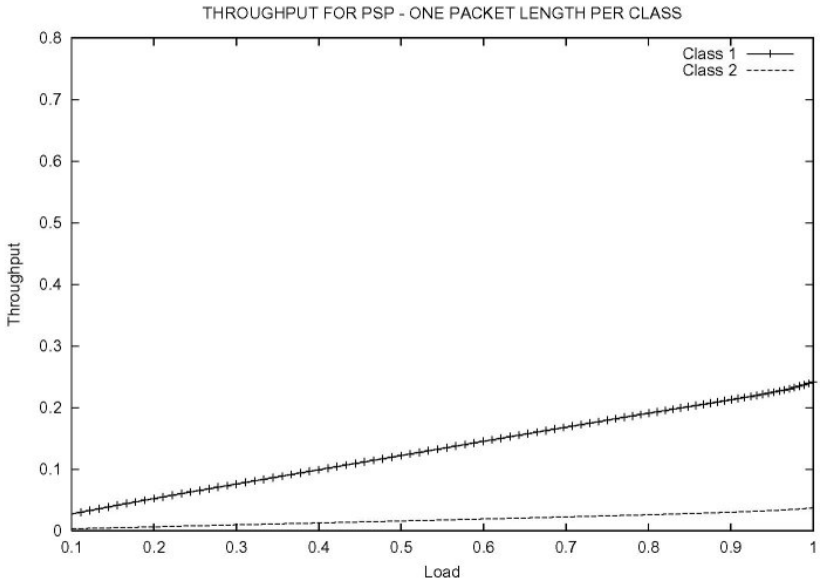


Figure 5.33 : Courbes de débit sous PSP : scénario n°8

La figure 5.33 met en évidence la différenciation du point de vue débit. Les courbes montrent la faible consommation de la bande-passante car les paquets appartenant à cette classe sont de petite taille. En comparant cette figure avec la figure 5.29, nous pouvons constater le phénomène suivant : la classe pour laquelle la taille des paquets a été modifiée a vu son débit nettement inférieur à celui de la figure 5.29.
Nous pouvons ainsi juger de l'impact de la taille des paquets sur l'obtention du partage des ressources d'un réseau en terme de débit.

La figure 5.34 expose la conséquence directe des figures précédentes (5.31 et 5.33) : la puissance des classes est nettement inférieure à celle présentée dans la figure 5.30, toujours du fait de l'écart qui se pose entre la taille des paquets des flots TCP et UDP.

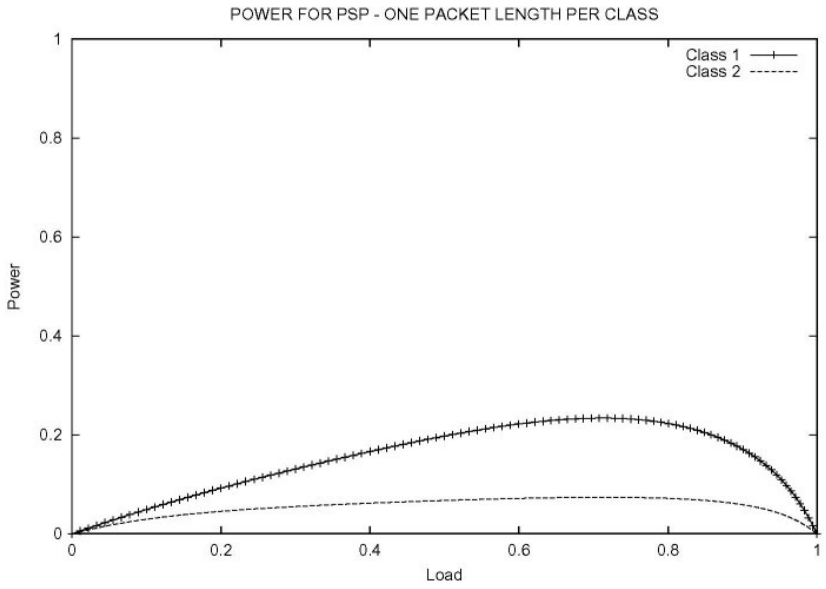


Figure 5.34 : Courbes de puissance sous PSP : scénario n°8

5.5 Conclusions

Nous avons présenté au cours de ce chapitre, l'implémentation de notre mécanisme basé sur la puissance et assurant l'équité entre les métriques « délai » et « débit » des applications qui, d'une part sont gourmandes en débit et d'autre part celles qui le sont vis-à-vis du délai. Nous avons par la suite exposé plusieurs cas de figures pour réaliser des simulations au moyen de l'outil *ns-2* et avons présenté, pour chaque scénario, les résultats des différentes mesures sous la forme de graphes : délais, débits et puissances pour chaque classe de trafic. Nous avons de même comparé notre algorithme avec le mécanisme WTP, présenté dans le chapitre précédent, en confrontant les graphiques des deux mécanismes.

Les conclusions essentielles à tirer sont les suivantes :

- comparé à WTP, le mécanisme que nous avons mis en œuvre offre une meilleure différenciation aussi bien du point de vue écart de différenciation que du point de vue seuil de différenciation (relatif à la charge du réseau) ;
- PSP assure la différenciation aussi bien pour des réseaux très chargés que pour des réseaux faiblement chargés ;
- PSP tient compte de la taille des paquets des flots, contrairement à WTP, ce qui rend son efficacité plus fiable ;
- PSP assure une différenciation relative si la taille des paquets est fixée de manière déterministe. Par conséquent, il est possible d'obtenir une garantie de qualité de service pour des flots hétérogènes si nous déterminons la taille de leurs paquets. Pour des applications multimédia, les paquets devraient être relativement courts, ce qui aboutirait à obtenir des courts délais et un faible débit, ce qui correspond aux caractéristiques requises par les flots non-élastiques ; pour les paquets appartenant à des flots élastiques, les paquets devraient être plus longs, ce qui mènerait à l'obtention d'un débit plus important et un délai plus grand, paramètres requis pour ce type de flots.

CONCLUSIONS & PERSPECTIVES

6. Conclusions et perspectives

Conclusion générale

L'objectif de cette thèse a été l'étude de la qualité de service (QoS) et des techniques d'ordonnancement des paquets à l'intérieur de réseaux hétérogènes.
Dans une première partie, nous avons identifié les paramètres qui définissent la qualité de service d'un réseau, et sur lesquels il faut agir pour garantir une qualité de service au-dessus d'un réseau. Les métriques qui nous permettent de définir la qualité de service sont le débit, le délai et par conséquent la gigue, ainsi que le taux de pertes des données. Ces éléments sont en effet essentiels pour juger de la qualité de transmission des informations et par conséquent, sont primordiaux pour définir la qualité de service. Par ailleurs, nous avons évoqué les mécanismes de qualité de service à un niveau global, en présentant les architectures proposées par les groupes de travail IntServ et DiffServ. Ces architectures sont basées sur deux approches différentes, mais ont pour point commun de tenter d'offrir à chaque application le maximum de ressources pour leur assurer une crédibilité correcte de transmission. La première théorie se base sur le principe de contrôle d'admission des flots ainsi que de la réservation de ressources au moyen du protocole RSVP. Cette approche définit trois classes de service : le service « best-effort » pour permettre d'acheminer les données sans aucune garantie ; le service à charge contrôlée pour offrir une certaine garantie de débit ; enfin le service garanti qui permet d'attribuer de meilleures garanties temporelles et en débit que les précédents services. L'approche DiffServ permet, quant à elle, d'effectuer une différenciation de services en se basant sur les micro-flux : les paquets sont ainsi marqués au niveau de leur en-tête pour leur attribuer des degrés de priorités selon les applications auxquelles ils appartiennent et leurs besoins en débit, délai et/ou pertes. Dans cette architecture, le service « best effort » est destiné aux applications à faible priorité ; le service « assured forwarding » dédié aux applications qui demandent une priorité moyenne, telles la navigation sur le Web ; enfin, le service à forte priorité, « expedited forwarding » associé aux applications temps-réel.

Dans la seconde partie de ce document, nous avons défini les réseaux hétérogènes comme étant des réseaux transportant des informations de diverses natures. Cette hétérogénéité découle des multiples applications qui circulent sur l'Internet actuel, à savoir des simples transferts de données (applications FTP par exemple), des applications multimédia telles que la transmission de voix, vidéo, applications interactives, etc… D'une manière plus précise, nous avons réalisé une classification de ces données hétérogènes en deux grandes catégories en exposant les critères spécifiques par lesquels ils se distinguent : tout d'abord, les flots élastiques qui doivent leur appellation à leur capacité de s'adapter à un environnement à variations temporelles. Cette catégorie englobe les applications du type transfert de fichiers,

fonctionnant au-dessus du protocole TCP et sont aussi connus pour leur ferme besoin en débit. La seconde famille des flots est qualifiée de non-élastique ; contrairement à la catégorie précédente, les applications qui font partie de cette famille sont sujets à de forts besoins de garanties en terme de délai : nous y avons classé les applications temps-réel qui sont elles-mêmes divisées en deux classes : d'une part, les applications fermes telles que la téléphonie ou voix sur IP ; d'autre part, les applications plus souples, telles que la vidéoconférence.
Nous nous sommes de même intéressés à la manière dont tous ces flots sont traités à l'intérieur des routeurs. Nous avons ainsi présenté, dans un premier temps, les algorithmes d'ordonnancement sous un aspect général, en distinguant les mécanismes conçus pour les files d'attente simples, et ceux adaptés pour les files d'attente multiples. Puis nous avons longuement discuté des politiques d'ordonnancement en faisant un état de l'art de l'existant. Nous avons pu distinguer à la suite de cette étude deux catégories d'ordonnancement : celle qui s'intéresse à un ordonnancement selon le délai et qui se trouve avantageux pour les applications non-élastiques à forte contrainte temporelle ; la seconde famille des ordonnanceurs vise la deuxième catégorie des applications, en favorisant leur fluidité selon le paramètre du débit et en leur attribuant par conséquent un minimum de leur besoin concernant cette métrique pour les écouler avec un minimum de rigueur.

La troisième partie de la thèse a visé la problématique que nous avons soulevée et a consisté à présenter l'approche de notre proposition. Il a été tout d'abord question d'aborder le problème de l'ordonnancement à différenciation proportionnelle, aspect primordial de notre axe de recherche. Nous avons en effet pu noter au cours du chapitre dédié à l'ordonnancement, que les mécanismes adoptés n'ont en aucun cas traité une équité proportionnelle de partage de ressources entre les applications hétérogènes. L'ordonnancement n'a en effet été réalisé qu'en tenant compte d'un seul paramètre de qualité de service, soit le délai, soit le débit, soit le taux de pertes. Mais nous n'avons jamais pu traiter un partage équitable proportionnel basé sur les deux principales caractéristiques que nous avons dégagées à la suite de la classification des flots : nous entendons par cela le débit et le délai, traités simultanément. L'approche que nous avons proposée est fondée sur cette idée de différenciation équitable en fonction du débit et du délai. Pour ce faire, nous nous sommes référés à une métrique d'évaluation des performances des réseaux, qui met en jeu aussi bien le débit que le délai, et ce, de manière simultanée. Ce paramètre, exprimé sous forme de rapport débit/délai n'est autre que la puissance et permet de déterminer l'apogée des performances d'un réseau. Notre ordonnanceur, a été conçu pour assurer une différenciation proportionnelle entre, d'une part, les applications élastiques, et d'autre part, les applications non-élastiques qui se partagent le réseau de manière concurrente. De ce fait, les applications se verront attribuer des priorités d'écoulement de leur trafic de manière tout à fait proportionnelle selon le débit et le délai requis par chaque classe de flot. L'implémentation et les tests ont été réalisés au moyen du simulateur ns-2 sur différentes topologies de réseaux.
Les résultats, présentés et discutés au sein de la dernière partie montrent nettement la différenciation attendue. Ainsi, nous avons pu démontrer qu'en présence d'un certain nombre d'applications sur un même canal, il est possible d'éviter le phénomène de famine des ressources de manière à satisfaire une certaine qualité de service à chaque application. Les distributions de priorités se faisant selon les requis en débit et délai, chaque flot bénéficiera d'une part relative à ses besoins.

Perspectives

Cette thèse a été réalisée dans le cadre d'une étude de la qualité de service pour des réseaux hétérogènes. Nous avons limité notre définition d'hétérogénéité à la diversité des flots. Ainsi, les réseaux hétérogènes représentent les réseaux sur lesquels circulent des données mixtes.

Cependant, le contexte d'hétérogénéité peut être associé à un environnement physique des réseaux. Dans ce cas, nous pouvons considérer une architecture de réseaux hétérogènes entre un ensemble de réseaux fixes en communication avec des réseaux sans-fil [47]. En particulier, ayant participé au projet RNRT-AIRS [1] pour l'étude de la gestion de la qualité de service dans les réseaux sans-fils basés sur la technologie 802.11 [48,49,50], nous pensons qu'il est intéressant d'exporter notre approche d'ordonnancement tout d'abord sur ces réseaux sans-fils, puis d'étendre les expérimentations sur les réseaux hétérogènes observés sous un angle physique. Les études pourraient s'effectuer dans un premier temps à l'aide d'un simulateur, puis réaliser des mesures réelles de manière à comparer les catégories de résultats.
Enfin, toutes nos simulations ayant été réalisées pour des files d'attente à capacité importante, nous n'avons pas tenu compte du critère de pertes. Par conséquent, nous pouvons proposer de réaliser toutes les manipulations, tant celles décrites dans le document que celles que nous avons proposées antérieurement, en considérant le taux de pertes et l'impact que cela peut apporter au niveau différenciation proportionnelle appliquée par le nouvel ordonnanceur.

BIBLIOGRAPHIE

Bibliographie

[1] Page web du projet “Qualité de service sur réseaux sans-fil”, Disponible sur http://drakkar.imag.fr/AIRS.html

[2] P. Anelli et T. Ziegler, « Evaluation de la différenciation avec WFQ ou WRED », Publication du projet AIRS, 7 Octobre 1999, Paris.

[3] Anurag, “Study of Expedited Forwarding in Differentiated Service and its Performance Characteristics using CBQ Implementation”

[4] G. Armitage, “Quality of Service in IP networks”, Macmillan Technical Publishing, pp.183-195, April 2000

[5] F. Baker, J. Krawczyk, A. Sastry, “RSVP Management Information Base using SMIv2”, RFC 2206, September 1997

[6] M. Baldi, “Real-time Services over Packet Switching Networks”, Thèse de doctorat - Polytechnique de Turin – 1998

[7] B. Baynat “ Théorie des files d’attente – Des chaînes de Markov aux réseaux à forme produit », Edition Hermès, Juin 2000

[8] C.R. Bennett, H. Zhang, “WF2Q: Worst-case Fair Weighted Fair Queuing”, In Proceedings of IEEE Infocom'96, San Francisco, CA, March 1996, pp. 120-128

[9] L. Berger, T. O'Malley, “RSVP Extensions for IPSEC Data Flows” , RFC 2207 , September 1997

[10] U. Bodin, A. Jonsson, and O. Schelon. “On Creating Proportional Loss Differentiation : Predictability and Performance”, In Proceedings of IWQoS 2001, pp. 372-386, Karlsruhe, Germany, June 2001

[11] T. Bonald and J. W. Roberts, “Performance of bandwith Sharing Mechanisms for Service Differenciation in the Internet”, ITC Specialist Seminar, Monterey, CA, USA, 18-20 September, 2000

[12] G. Booch, J. Rumbaugh et I. Jacobson, “Le guide de l’utilisateur UML”, Editions Eyrolles, 1999.

[13] C. Boutremans, J.Y. Le Boudec, “Adaptive Delay Aware Error Control for Internet Telephony”, Proceedings of 2nd IP-Telephony Workshop, pp.81-92, Columbia University, New York, April 2001

[14] R.Braden, D.Clark, S.Shenker, “Integrated Services in the Internet Architecture: an Overview” , RFC 1633, June 1994

[15] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification”, RFC 2205, September 1997

[16] R. Braden, L. Zhang, “Resource ReSerVation Protocol (RSVP) -- Version 1 Message Processing Rules”, RFC 2209, September 1997

[17] C. Chassagne, “De l’utilité des mesures dans les réseaux et dans l’Internet”, Rapport, Septembre 1997, disponible sur
http://www.urec.fr/metrologie/metrologie.html

[18] C. Chassagne. “Qualité de service dans l’Internet”. Rapport, Août 1998, disponible sur http://www.urec.fr/metrologie/article-qos.html

[19] R. Chipalkatti, J.F. Kurose, and D. Towsley, “Scheduling Policies for Real-Time and Non-Real-Time Traffic in a Statistical Multiplexer”, Proceedings of IEEE Infocom'89, pp. 774-783, Ottawa Canada

[20] N. Christin, J. Liebeherr, T. F. Abdelzaher, “A Quantitative Assured Forwarding Service”, Proceedings of IEEE Infocom 2002

[21] “Weighted Random Early Detection on the Cisco 12000 Series Router” , Technical Support, September 1999

[22] D. Clark, S. Shenker, L.Zhang, “Supporting Real-Time Applications in an Integrated Services Packet Network : Architecture and Mechanisms”, ACM SIGCOMM’92, pp 14-26, August 1992

[23] D.Clark, W.Fang, “Explicit Allocation of Best Effort Packet Delivery Service”, ACM Transactions on Networking, August 1998

[24] D. D. Clark, W. Feng, “Explicit allocation of best-effort traffic”, IEEE/ACM Transactions on Networking, vol. 6, no. 4, August 1998.

[25] J. Crowcroft, M.Handley, I.Wakeman, “Internetworking Multimedia”, 290 pages, 1st edition, November 1999

[26] R. Cruz, “Service Burstiness and Dynamic Burstiness Measures : A Framework”, Journal of High Speed Networks, vol.1, no.2, 1992, pp. 105-127

[27] L. A. Dasilva, "Pricing for QoS-Enabled Networks : A Survey", IEEE Communications Surveys, Second Quarter 2000

[28] C.Deleuze, "Qualité de Service dans l'Internet : Problèmes liés au haut débit et au facteur d'échelle", Thèse de Doctorat, Université Paris VI, Janvier 2000

[29] C. Deleuze and S. Fdida, « Stateless Virtual Clock : Un ordonnanceur de paquets pour les garanties de délai dans les réseaux de transit », CFIP'2000, Toulouse, France, 17-20 Octobre 2000

[30] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", ACM Computer Communication Review, 1989, pp. 3-12

[31] C. Dovrolis and P. Ramanathan, "A case for Relative Differentiated Services and the Proportional Differentiation Model", IEEE Network, September/October 1999

[32] C. Dovrolis, D. Stiliadis and P. Ramanathan, "Proportional Differentiation and Packet Scheduling", Proceedings of ACM SIGCOMM, September 1999

[33] C. Dovrolis and P. Ramanathan, "Proportional Differentiated Services, Part II : Loss Rate Differentiation and Packet Dropping", IEEE/IFIP International Workshop on Quality of Service (IWQoS), pp. 52-61, Pittsburgh, June 2000

[34] C. Dovrolis, D. Stiliadis and P. Ramanathan, "Proportional Differentiated Services : Delay Differentiation and Packet Scheduling", Proceedings of ACM SIGCOMM, September 1999

[35] H. J. Einsiedler and al., "Differentiated Services – Network Configuration and Management, Service Models and Architectures", EURESCOM Technical Information, pp.1-28, January 2001

[36] L. Essafi, G. Bloch, A. Andres, "An Adaptive Waiting Time Priority Scheduler for the Proportional Differentiation Model", High-performance Computing Symposium, 22-26 April 2001

[37] T.S. Eugene Ng, D.C. Stephens, I. Stoica, H. Zhang, "Supporting Best-Effort Traffic with Fair Service Curve", GLOBECOM'99, Rio de Janerio, Brazil, December 1999

[38] P. Ferguson, G. Huston, J. Wiley, "Quality of Service : Delivering QoS in the Internet and in Corporate Networks" , 1998

[39] P.Ferguson, G.Huston, "Quality of Service in the Internet : Fact, Fiction or Compromise ? ", Proceedings INET'98, pp. 21-24, Geneva, Switzerland - July 1998

[40] V. Firoiu, X. Zhang, "Best-Effort Differentiated Services : Trade-off Service Differentiation for Elastic Applications", IEEE ICT'2001 Conference, Bucharest, Romania, 4-7 June 2001

[41] S. Floyd , and V. Jacobson, , "Random Early Detection Gateways for Congestion Avoidance", vol.1, no.4, August 1993, pp. 397-413

[42] S.Gai, D.Dutt, N.Elfassy, Y.Bernet, "RSVP+ : An Extension to RSVP", Internet Draft , June 1999. draft-sgai-rsvp-plus-00.txt

[43] B. Gaidioz, "Création et mise en oeuvre d'un nouveau service DiffServ : le « Fair Forwarding » ", DEA d'Informatique Fondamentale de l'ENS-Lyon, Juin 2000

[44] B. Gaidioz et P. Primet, " Propositions pour une différenciation de services équitable dans l'Internet", De Nouvelles Architectures pour les Communications (DNAC), Paris, France, November 2001

[45] B. Gaidioz and P. Primet. "EDS: A New Scalable Service Differentiation Architecture for Internet". In Proceedings of International Symposium on Computer Communication (ISCC) 2002, pages 777-782, Taormina, Italy, July 2002.

[46] B. Gaidioz, P. Primet, and B. Tourancheau. "Differentiated Fairness Model and Implementation". In Proceedings of IEEE High Performance Switching and Routing (HPSR) 2001, pp. 260-264, Dallas, USA, May 2001

[47] J.A Garcia-Macias and L. Toumi, "Wireless Local Access to the Mobile Internet", chapitre à paraître en Mars 2003 dans Handbook of Wireless Internet, Edité par M. Illyas et B. Furht, CRC Press Florida

[48] J.A. García-Macías, F. Rousseau, G. Berger-Sabbatel, L. Toumi, and Andrzej Duda, "Mobility Management for Providing QoS in Local Area Wireless Networks", Proceedings of DAIS'2001, Third IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems, Kraków, Poland, 2001

[49] J.A. García-Macías, F. Rousseau, G. Berger-Sabbatel, L. Toumi, and Andrzej Duda, "Quality of Service and Mobility for the Wireless Internet", Proceedings of First ACM Wireless Mobile Internet Workshop, Rome, Italy, 2001

[50] J.A. García-Macías, F. Rousseau, G. Berger-Sabbatel, L. Toumi, et Andrzej Duda, "Différenciation des services sur les réseaux sans fil 802.11", Colloque Francophone sur l'Ingénierie des Protocoles, 27-30 Mai 2002, Montréal, Canada

[51] S.J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications", Proceedings of IEEE INFOCOM'94, Toronto, Italy, June 1994, pp.636-646

[52] A.K. Parekh, R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", IEEE/ACM Transactions on Networking, vol. 1, no. 3, June 1993, pp.344—357

[53] A.G. Greenberg, N. Madras, "How Fair is Fair Queuing", Journal of ACM, vol.39, no.3, July 1992, pp. 568-598

[54] R. Guérin and V. Persi, "Quality-of-Service in Packet Networks: Basic Mechanisms and Directions", Computer Networks, Vol. 31, N. 3, pp. 169-189, February 1999

[55] H. Gundersen, F. Trydal, "QoS for real-time IP traffic", Graduate Thesis, Agder University College, Norway, May 2001

[56] L. Gzara, "Les patterns pour l'ingénierie des Systèmes d'Information Produit", Doctorat de l'INPG, spécialité Génie Industriel, Décembre 2000.

[57] J. Hadi Salim and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, July 2000

[58] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "RFC 2597 - Assured Forwarding PHB Group" , June 1999

[59] E. Horlait, N. Rouhana, "Qualité de Service dans l'Architecture TCP/IP", Congrès DNAC De Nouvelles Architectures pour les Communications, 1-3 Décembre 1999, Paris, France

[60] C.Huitema, "Le routage dans l'Internet", Editions Eyrolles, 1994

[61] P. Hurley and all., "The ABE Service" , Internet Draft, draft-hurley-alternative-best-effort-01.txt, November 2000

[62] P.Hurley, J.Y. Le Boudec, P. Thiran "The Alternative Best-Effort Service", SSC Technical Report SSC/1999/036, Institute for Computer Communication and Applications, Lausanne, September 1999

[63] P.Hurley, J.Y. Le Boudec, "A Proposal for an Asymmetric Best-Effort Service", Proceedings of IEEE/IFIP IWQoS `99, London, England., May 1999, pp. 132-134

[64] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara. "ABE : Providing low delay service within best effort. IEEE Networks, 15(3) : 60-69, May 2001

[65] V. Jacobson, K. Nichols, K. Poduri, "RFC 2598 - An Expedited Forwarding PHB" , June 1999

[66] K. Kilkki, J.Ruutu, "Simple Integrated Media Access – an Internet Service Based on Priorities", Proceedings of the 6th International Conference on Telecommunication Systems Modeling and Analysis - TN, USA- March 5-8, 1998

[67] L. Kleinrock, "A Delay Dependent Queue Discipline", Journal of ACM, vol.14, no.2, 1967, pp.242-261

[68] L. Kleinrock, "Queuing Systems, volume II", John Wiley and Sons, 1976

[69] F. Li, N. Seddigh, B. Nandy, and D. Matute. "An Empirical Study of Today's Internet Traffic for Differentiated Services IP QoS", The Fifth IEEE Symposium on Computers and Communication (ISCC 2000), Antibes, France, July 2000

[70] R. R.-F. Liao and A.T. Campbell, "Dynamic Core Provisioning for Quantitative Differentiated Service", In Proceedings of IWQoS 2001, pp. 404-418, Karlsruhe, Germany, June 2001

[71] Q. Ma, "Quality-of-Service Routing in Integrated Sevices Networks", PhD Thesis, Carnegie Mellon University, USA, January 1998

[72] R. Makkar and all., "Empirical Study of Buffer Management Scheme for Diffserv Assured Forwarding PHB", The Ninth International Conference on Computer Communication and Networks (IC3N'2000), Las Vegas, October 2000

[73] A. Mankin, and all, "Resource ReSerVation Protocol (RSVP) -- Version 1 Applicability Statement --Some Guidelines on Deployment", RFC 2208, September 1997

[74] O. N. Medina Carvajal, " Etude des Algorithmes d'Attribution de Priorités dans un Internet à Différenciation de Services », Thèse de Doctorat – Université de Rennes 1, Mars 2001

[75] O. N. Medina, J-M. Bonnin, L. Toutain, " Service DiffServ pour les flux audio et vidéo", Colloque Francophone pour l'Ingénierie des Protocoles (CFIP'2000), Toulouse, France, 17-20 Octobre 2000

[76] J. Nagle, "On Packet Switches with Infinite Storage", RFC 970, December 1985

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

[78] T. Nandagopal, N. Venkitaraman, R. Sivakumar and V. Barghavan, "Delay Differentiation and Adaptation in Core Stateless Networks", In Proceedings of IEEE INFOCOM 2000, pp. 421-430, Tel-Aviv, Israel, April 2000

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

[80] K. Nichols, V. Jacobson, L. Zhang, "RFC 2638 - A Two-bit Differentiated Services Architecture for the Internet" , July 1999

[81] disponible sur http://www.isi.edu/nsnam/ns

[82]disponible sur http://www.opnet.com

[83] disponible sur http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/~checkout~/otcl-tclcl/otcl/doc/tutorial.html

[84] P. Pan, H. Schulzrinne, "YESSIR : A Simple Reservation Mechanism for the Internet", Computer Communication Review ACM SIGCOMM , Vol.29, no2, April 1999

[85] J. Postel, "Transmission Control Protocol", RFC 793, September 1981

[86] C. Kolias and L. Kleinrock, "The Power Function as a Performance and Comparison Measure for ATM Switches", Globecom'98, pp. 381-386,Sydney, Australia, November 1998

[87] http://www-rst.int-evry.fr/~hebutern/IT21/Simu.html

[88] K. Ramakrishnan, S. Floyd, D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP”, RFC 3168, September 2001

[89] F. Risso, “Quality of Service on Packet Switched Networks”, Thèse de doctorat - Polytechnique de Turin – January 2000

[90] J. W. Roberts, “ Qualité de service, tarification et contrôle d’admission dans l’Internet », CENT/DAC, RHDM, Brest, France, Septembre 1999

[91] J. W. Roberts, “Engineering for Quality of Sevice”, France Télécom – CNET, Issy les moulineaux, France, July 1998

[92] H. Schulzrinne, “Resource Control and Reservation”, Advanced Internet Services Course, October 2001
Available at : *http://www.cs.columbia.edu/~hgs/teaching/ais/slides/rsvp.pdf*

[93] S. Shenker, C. Partridge, R. Guerin, “Specification of Guaranteed Quality of Service”, RFC 2212, September 1997

[94] S. Shenker, J. Wroclawski, “General Characterization Parameters for Integrated Service Network Elements”, RFC 2215, September 1997

[95] S. Shenker, “Fundamental Design Issues for the Future Internet”, IEEE Journal of Selected Areas in Communication, vol. 13, no. 7, September 1995, pp. 1176-1188

[96] E. Siegel, “Designing Quality of Service, Solution for the Enterprise”, Wiley Editions, 1999

[97] I. Stoica, H. Zhang, T.S. Eugene Ng, “A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Service”, Proceedings of SIGCOMM'97, Cannes, France, 1997, pp. 249-262

[98] A. Striegel and G. Manimaran, “Differentiated Services : Packet Scheduling with Delay and Loss Differentiation”, Computer Communications, vol.25, no.1, pp.21-31, Jan. 2002

[99] L.Toutain, “Internet à Intégration de services”, Colloque GRES 97, Brest, France, Septembre 1997

[100] P. White and J. Crowcroft. “Integrated services in the Internet : the next stage in Internet : state of the art”, Proceedings of the IEE, Vol.85, no.12, pp.1934-1946, December 1997

[101] J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, RFC 2210, September 1997

[102] J. Wroclawski, “Specification of the Controlled-Load Network Element Service”, RFC 2211, September 1997

[103] X. Xiao and L.Ni, “Internet QoS : The Big Picture”, IEEE Network, vol.13, no.2, pp.1-13

[104] L.Zhang, and all, “RSVP : A New Resource reservation Protocol”, IEEE Network Magazine, pp. 8-18, September 1993

[105] L.Zhang, “Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks”, Proceedings of SIGCOMM '90, pp. 19-29

ANNEXES

Annexes

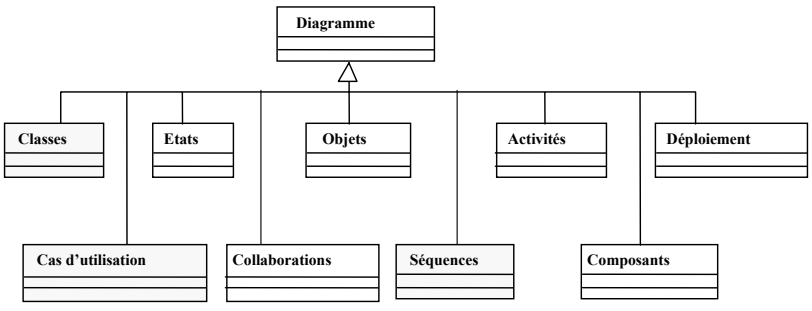
Modélisation de données – UML

UML s'appuie sur des concepts (des choses), des relations et des diagrammes [56].

Les ***concepts*** manipulés dans UML sont de quatre natures. On distingue des concepts *structurels* (les classes, les interfaces, les collaborations, etc.), *comportementaux* (les interactions, les états d'objets), *annotationnels* (les notes) et *de groupement* (les package, les sous-systèmes, etc.)

Les ***relations*** permettent de lier les concepts. UML offre quatre types de relations : les associations, les généralisations, les dépendances et les réalisations.

Les ***diagrammes*** constituent des représentations graphiques d'ensemble de concepts. Ils sont définis par des graphes connexes dont les sommets sont des concepts et les arcs des relations inter-concepts. Les diagrammes permettent de représenter les systèmes sous différentes perspectives et différents niveaux d'abstractions. UML offre neuf types de diagrammes offrant des visions statiques ou dynamiques des systèmes.



Exemple de diagramme

A.1 Classe et Objet

Le concept de base fondamental utilisé dans les diagrammes de classes est bien entendu celui de classe. Une classe décrit un ensemble d'objets de même structure définie par un ensemble

d'attributs et de relations et de même comportement défini par un ensemble d'opérations. Les classes identifient les abstractions du domaine du problème.

Une classe UML comprend trois compartiments qui permettent respectivement de représenter le nom de la classe, ses attributs et ses opérations. Si un diagramme de classes est trop important (trop de classes et de relations) ou si seule une vision globale est nécessaire, il est possible de donner une vision moins complète des classes, par exemple en ne faisant apparaître que les noms des attributs et/ou des opérations, voire en ne donnant que le premier compartiment (nom de la classe). Ce premier compartiment peut être doté d'un stéréotype permettant de " typer " les classes.

A.2 Association et Lien

Les associations constituent les relations fondamentales des diagrammes de classes. Elles permettent d'associer un objet d'une classe à un ou plusieurs objets d'une autre classe. Les liens sont des instances (éléments) des associations au même titre que les objets sont des instances des classes.

Les rôles dans une association décrivent comment une classe voit l'autre classe au travers de l'association. Cette notion est particulièrement importante dans deux cas. Le premier cas concerne les associations multiples entre deux classes. Le deuxième cas pour lequel la notion de rôle est fondamentale est celui des associations récursives. Une association récursive relie une classe à elle-même.

Dans une association, chaque rôle porte une multiplicité qui indique combien (au minimum et au maximum) d'objets de la classe jouant le rôle peuvent être liés à un objet de la classe associée. Les multiplicités usuellement utilisées sont les suivantes :

1	Un et un seul
0..1	Zéro ou un
M..N	De M à N
*	De Zéro à plusieurs
0..*	De Zéro à plusieurs
1..*	De un à plusieurs

Valeurs usuelles des multiplicité

A.3 Agrégation et Composition

UML offre deux types d'association particulièrement intéressantes pour la gestion des données techniques : l'agrégation et la composition. L'agrégation est une association dans laquelle une classe joue un rôle prédominant par rapport à l'autre. Il peut d'ailleurs s'agir de la même classe dans le cas des associations récursives. Une association d'agrégation est notée par un losange blanc du côté de l'agrégat. L'agrégation est utilisée lorsque l'on souhaite indiquer qu'un des objets doit être manipulé comme " un tout.

La composition est un cas particulier de l'agrégation qui désigne une dépendance quasi complète du composant vis à vis du composé : le composant peut naître hors de son composé mais n'est pas partageable. Il s'ensuit que la multiplicité du côté de l'agrégat ne peut prendre que les valeurs 0 ou 1.

A.4 Héritage

La relation d'héritage permet d'établir des classifications de concepts. Dans un premier temps on peut l'assimiler à une inclusion ensembliste : X est une" sorte de " Y qui est elle-même une sorte de Z. La relation d'héritage permet d'organiser les classes par niveau d'abstraction.