Qualité de Service dans l'Internet : Garantie de Débit TCP dans la Classe AF

Emmanuel Lochin

Thèse de Doctorat de l'université Paris VI
Pierre et Marie Curie

Spécialité

Systèmes Informatiques

présentée par

M. Emmanuel Lochin

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Qualité de service dans l'Internet : Garantie de débit TCP dans la classe AF

Présentée et soutenue publiquement le 10/12/04 devant le jury composé de

Pr. Michel Diaz	**Rapporteur**
M. Fabrice Guillemin	**Rapporteur**
M. Olivier Bonaventure	**Examinateur**
M. Patrick Cocquet	**Examinateur**
M. Pascal Anelli	**Co-encadrant de thèse**
Pr. Serge Fdida	**Directeur de thèse**

Thèse de Doctorat de l'université Paris VI
Pierre et Marie Curie

Spécialité

Systèmes Informatiques

présentée par

M. Emmanuel Lochin

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Qualité de service dans l'Internet : Garantie de débit TCP dans la classe AF

Présentée et soutenue publiquement le 10/12/04 devant le jury composé de

Pr. Michel Diaz	**Rapporteur**
M. Fabrice Guillemin	**Rapporteur**
M. Olivier Bonaventure	**Examinateur**
M. Patrick Cocquet	**Examinateur**
M. Pascal Anelli	**Co-encadrant de thèse**
Pr. Serge Fdida	**Directeur de thèse**

Il serait présomptueux de penser que ce que l'on sait soi-même n'est pas accessible à la majorité des autres hommes.

Konrad Lorentz 1905-1989
(Les huit péchés capitaux de notre civilisation, p.8, Flammarion)

Remerciements

J'ai eu la chance de faire cette thèse dans un cadre motivé et sympathique, tout en exerçant mon travail d'ingénieur au sein de l'équipe Réseaux et Performances du LIP6.

Je tiens à remercier Serge Fdida de m'avoir proposé une thèse et de m'avoir fait confiance tout au long de ces années. Je dois dire qu'en tant qu'ingénieur ou qu'en tant que doctorant, je n'ai jamais eu à me soucier des problèmes concernant les moyens ou les budgets. Cela facilite grandement les choses quant à la réalisation des travaux demandés. Je tiens également à remercier Pascal Anelli pour la qualité de son encadrement. J'ai toujours eu réponse à mes questions, soutien et encouragements. Je souhaite à tout doctorant de recevoir un encadrement de cette qualité et d'être aussi complice que j'ai pu l'être avec mes encadrants de thèse. Bref, Pascal et Serge merci.

Je tiens à remercier tous les membres de ce jury : M. Michel Diaz, directeur de recherche du CNRS au LAAS à Toulouse, et M. Fabrice Guillemin de France Télécom R & D Lannion, qui ont accepté d'être rapporteurs de ma thèse, pour l'intérêt qu'ils ont porté à ce travail. M. Olivier Bonaventure de l'Université catholique de Louvain et M. Patrick Cocquet, président de l'IPv6 Task Force France et président de la société 6WIND, qui ont eu l'amabilité de bien vouloir participer à ce jury.

Je remercie également les membres de l'équipe réseau et performances avec qui j'ai apprécié travailler pendant ces années, et plus particulièrement : toute l'équipe système : Françoise, Konstantin, ma page de Manuel préférée : Manu Bouyer et Bruno Talavera qui, par son sens aiguisé de la critique, m'a permis de faire évoluer mon travail. Les membres de l'équipe, notamment Anne avec qui j'ai particulièrement apprecié de travailler et avec qui j'espère bien continuer. Michel, toi, thésard, si tu te sens désespéré, va relativiser le tout dans le bureau de Michel. Sans oublier Kim, Bruno, Kavé, Timur, Eric (pardon, Monsieur le Directeur), Brigitte, Bénédicte et Olivier. Je remercie également Naceur des discussions éclairées que nous avons eues ensemble et qui reconnaîtra sûrement quelques courbes du chapitre 7. Les doctorants passés et futurs et notamment Prométhée pour les pauses café. Je tiens aussi à remercier Clémentine et Véronique qui se sont occupées de toutes les démarches administratives et qui, par leur efficacité, contribuent grandement au bon fonctionnement du laboratoire.

Enfin j'aimerais remercier mes parents et ma famille de m'avoir laissé mener ma barque comme je le souhaitais et Karine, qui a relu et corrigé cette thèse et m'a soutenu tout au long de ce travail.

Résumé

Les travaux présentés dans cette thèse concernent la qualité de service dans les réseaux à commutation de paquets de grande dimension et plus spécifiquement la garantie de débit pour les flots TCP. L'architecture à différenciation de services, définie par l'IETF, dite DiffServ s'articule autour de trois services : le service garanti, le service assuré et le service par défaut dit « au mieux ». Le service assuré a été élaboré pour servir les flots à caractère élastique comme les flots TCP. Cependant, des problèmes subsistent en ce qui concerne cette garantie de débit pour ce type de flots dans ce service. Ces dernières années, un effort important a été fait pour améliorer les mécanismes d'ordonnancement, de classification et de rejet au sein des routeurs afin qu'ils répondent au mieux à la spécification du service assuré. Malheureusement, leurs choix de conception et la complexité de leur mise en œuvre est très souvent un frein à leur déploiement dans un Internet réel. Dans cette thèse, nous proposons une solution originale de conditionnement des flots TCP permettant de garantir un débit à la fois pour des flots individuels et des groupes de flots. Cette solution fonctionne quelquesoit le facteur d'echelle.

Mots-Clés :

Qualité de Service, différenciation de services, facteur d'échelle, agrégation de flots, service assuré, TCP.

Abstract

This work focuses on quality of service in large scale packet switching networks and, more specifically, throughput guarantee for TCP flows. The differentiated architecture, defined by the IETF, also called DiffServ, provides three services : the guaranteed service, the assured service and the best effort service. The assured service was conceived for serving elastic flows such as TCP flows. Nevertheless some problems persist when it comes to assuring a TCP throughput. These past years, a consequent effort has been made in order to improve scheduling, classification and dropping mechanisms in routers so that they match the assured service specification as best as possible. Unfortunately, design choices and complex implementation are frequent limitations to their deployment in the real Internet. In this thesis, we propose an original conditioning approach for TCP flows allowing for a guarantee both for single and aggregate TCP flows. This solution is scalable.

Key Words:

Quality of Service, differentiated services, scalability, aggregation, assured service, TCP

Table des matières

A Karine
et
à mes parents.

Chapitre 1

Introduction

1.1 EVOLUTION DE L'INTERNET

En quittant le monde académique et militaire pour se développer dans la société, les contraintes posées sur le service de transfert offert par l'Internet ont changé. En effet, l'Internet commercial s'accommode mal du service « au mieux » qui a prévalu depuis ses débuts. Les progrès des technologies numériques ont fait émerger des applications qui posent de nouvelles contraintes au service de communication. La Qualité de Service (*QoS*) est au cœur de ces nouvelles demandes. La QoS s'exprime principalement au travers des paramètres de bande passante, de latence, de variation de la latence également appelée gigue et de taux de perte. Au niveau de la couche d'interconnexion IP, la QoS désigne les traitements appliqués aux flots de paquets afin qu'ils obtiennent un niveau de service en adéquation avec la demande applicative. Le choix d'un service de communication pertinent pour l'application pose le problème de la sémantique des services offerts par un réseau au sens large. Ceux-ci doivent être en nombre limité et doivent servir un large éventail d'applications actuelles ou futures. La multiplicité des services est un élément de complexité dans la réalisation, la gestion et la lisibilité des services.

Cette réflexion sur le nombre de services et leurs caractéristiques a été menée, il y a déjà quelques années, par le groupe de travail Integrated Services (IntServ). La solution retenue [BCS94] s'appuie sur trois services que l'on peut schématiser de la manière suivante :

- le service de l'Internet classique de communication dit « au mieux » ;
- un service d'un Internet sans congestion, surdimensionné, c'est-à-dire peu de pertes de paquets (dues aux erreurs d'intégrité uniquement) et une latence proche d'un réseau peu chargé. La définition du service reste relativement peu précise ;
- un service pour des utilisateurs exigeants sur les délais.

L'architecture DiffServ, proposée ensuite par [NJZ99], s'appuie aussi sur un découpage des services en trois catégories assez similaires à celui proposé par IntServ. Ce découpage en trois classes de services (par défaut, intermédiaire et absolu) reste encore pertinent de nos

jours. Il repose sur le constat que les applications se divisent schématiquement, en fonction de leurs exigences de QoS, en deux catégories [Rob98] :

- Les applications **interactives**, comme par exemple les applications distribuées, les applications de type « Contrôle/Commande » qui effectuent un contrôle distant de systèmes mécaniques, les applications de téléphonie ou de vidéo conférence. Plus généralement, cela concerne toutes les applications qui en plus du débit posent des contraintes sur les délais de transit de chaque paquet et sur la variabilité de ces délais ;
- Les applications dites **élastiques**, qui consistent typiquement en un transfert de fichiers (comme les transferts effectués par le protocole *ftp*[1]), sont sensibles uniquement au temps de transfert total du ou des fichiers. La QoS requise porte alors sur le débit moyen reçu au cours du transfert, et non sur le délai de transit de chaque paquet.

Cette division des applications en deux catégories identifie les garanties de QoS exigées d'un réseau en termes de **délai de transit** pour un débit intrinsèque donné, ou en termes de **débit utile** (*goodput*) indépendamment du délai de transit des paquets ou du débit instantané.

Les services proposés sont de trois catégories :

1. un service « au mieux » (*Best Effort : BE*) traditionnel qui a l'avantage d'être simple à mettre en œuvre et économique ;
2. un service à débit assuré (*Assured Service : AS*) qui améliore la qualité du service BE et destiné à mieux traiter une large gamme d'applications. Le service AS est conçu pour les flots à caractère élastique. Ces flots ont un débit qui augmente tant qu'il y a des ressources disponibles et diminue quand une congestion apparaît. Le débit de ces flots se décompose en deux parties :
   - **un débit minimum** assuré et invariant. En cas de congestion dans le réseau, les paquets de cette partie étant marqués comme inadéquats à la perte, ils ne doivent pas souffrir de la congestion ;
   - **un débit opportuniste**, correspondant à un débit de paquets opportunistes. Il constitue la partie variable du débit dite « élastique ». Aucune garantie n'est apportée à ces paquets. Ceux-ci sont acheminés par le réseau sur le principe dit « au mieux ». En cas de congestion, ce sont ces paquets qui seront éliminés en premier. Le débit opportuniste doit varier en fonction de l'état des ressources utilisées, d'où son caractère d'élasticité. Il demande à l'utilisateur du service réseau d'adapter son débit opportuniste à la capacité du moment du réseau. En tout état de cause, le débit offert par ce service doit être meilleur que le service BE.
3. un service à garanties fermes (*Guaranteed Service : GS*) pour des applications qui posent des contraintes de QoS fortes et ne supportant pas de variations de la QoS. Ce service est destiné aux flots déterministes et temps réels. Il est conçu pour des flots continus ou réguliers et est souvent assimilé à une émulation de ligne louée.

La faisabilité de ces services et les problèmes de la QoS de manière générale sont étudiés dans de nombreux projets ; citons en particulier l'activité TF-TANT [TFT] du projet européen GEANT [GEA], le projet QBone [THD+99] et les projets européens TEQUILA [TEQ], CADENUS [CAD] et AQUILA [AQU], contribuant tous à la proposition d'architectures orientées DiffServ, dont le cadre général est défini dans [BBC+98].

[1] File Transfert Protocol

1.1.1 Contributions de cette thèse

Les travaux présentés dans cette thèse s'inscrivent dans la continuité du projet @IRS[2] et portent sur la conception, l'implémentation et la mesure des performances d'une architecture de communication garantissant une QoS de bout en bout.

Les apports de cette thèse sont :

- le développement d'une architecture de traitement différencié des paquets pour Internet, en conformité avec l'architecture DiffServ classique. Une mise en œuvre de cette architecture ainsi qu'une évaluation de ses performances sera présentée ;
- faisant suite aux mesures réalisées tout d'abord avec le protocole UDP [GAC+02], nous évaluerons les performances de cette architecture avec le protocole TCP au sein du service assuré (AS). Ces mesures mettront en évidence deux points importants : le premier est que la caractérisation et le conditionnement de flots TCP par des mécanismes DiffServ standards comme ceux définis dans [BBC+98] ne sont pas suffisants pour obtenir une garantie de débit moyen ; le second est que ces mécanismes de caractérisation de trafic, comme le *token bucket marker*, sont de très mauvais descripteurs de trafic TCP ;
- après une étude exhaustive des propositions existantes [NPE00, EGS02, KAJ01, FRK00, HBF02] permettant d'apporter une réponse à ce problème de garantie de débit, une classification de ces méthodes sera effectuée. Cette classification permettra d'identifier les différents niveaux d'actions possibles et leur faisabilité. Au travers de l'étude de ces nouvelles méthodes de conditionnement, nous mettrons en évidence que la complexité des mesures nécessaires au fonctionnement de ces nouveaux algorithmes est un frein pour passer du cadre de la simulation au cadre d'une utilisation réelle. Nous verrons également que le conditionnement pose un gros problème de passage à l'échelle car la granularité choisie par ces techniques est le microflot. Enfin, nous verrons que le point commun majeur de toutes ces méthodes proposées est d'effectuer un marquage à la perte plus agressif de la partie opportuniste des flots. Or, [YR99] a montré que l'augmentation du nombre de paquets marqués prioritaires à la perte n'est pas forcément la meilleure solution. En effet, ce marquage peut provoquer de fortes oscillations du débit instantané d'un flot TCP pouvant être préjudiciable pour son débit moyen ;
- partant de ce constat et des mesures effectuées, nous proposerons une solution en opposition à ces mécanismes. Tout d'abord, nous n'opèrerons pas sur le marquage d'un flot TCP. Nous utiliserons un mécanisme de lissage de trafic original, prenant en compte la quantité de trafic opportuniste véhiculée au niveau du goulot d'étranglement du réseau, sans hypothèse sur sa localisation. Le conditionnement du trafic s'effectuera sur une granulité plus large permettant un meilleur passage à l'échelle et sera donc plus réaliste pour une utilisation dans le cadre d'un réseau réel. Plusieurs mesures illustreront l'efficacité de cette solution et sa capacité à garantir un débit à un flot ou un ensemble de flots TCP au travers de la classe assurée.

[2] @IRS (Architecture Intégrée de Réseaux et Services - décembre 1998 - avril 2001) est un projet français financé par le Réseau National de la Recherche en Télécommunications (RNRT).

1.1.2 Plan du document

La thèse est structurée en 7 chapitres de la façon suivante : le chapitre 2 définit le cadre de cette thèse en donnant une présentation de la qualité de service dans l'Internet. Les mécanismes utilisés pour sa mise en œuvre seront étudiés au chapitre 3. Les principes de l'architecture DiffServ ainsi qu'une implémentation et sa mise en oeuvre seront exposés au chapitre 4. Des mesures sur cette implémentation seront données au chapitre 5. L'état de l'art des techniques de garantie de débit d'un flot TCP au sein d'une classe AF sera présenté au chapitre 6. Les objectifs de cette étude feront suite. Les solutions envisagées, leur développement et leurs résultats seront exposés au chapitre 7. En conclusion, nous donnerons la perspective de travaux futurs.

Chapitre 2

La qualité de service dans l'Internet

2.1 INTRODUCTION

Le développement considérable de l'Internet et l'arrivée sur le marché d'offres de connexions à des débits supérieurs à 1Mbits/s ont ouvert la voie au développement de nombreuses applications multimédias. Ces applications possèdent des contraintes qui ne se satisferont pas du service dit « au mieux » (*best-effort*) fourni nativement par l'Internet. Même si certaines de ces applications sont conçues pour fonctionner avec ce service, elles ont besoin d'un minimum pour travailler correctement. D'autres applications, au contraire, possèdent de fortes contraintes et ne peuvent s'adapter à ce service.

Un réseau est dit à « qualité de service » lorsqu'il est capable de répondre aux attentes d'une application à besoins spécifiques. Il doit être capable d'offrir un service prévisible relativement constant et d'être capable de répondre à des paramètres de performance comme une garantie de débit ou une garantie de délai de bout en bout.

Deux architectures, basées sur des concepts totalement différents, ont été proposées par l'IETF[1] afin de fournir la qualité de service. Nous détaillerons dans ce chapitre les principes de ces deux architectures.

2.2 LE MODÈLE INTSERV / RSVP

Historiquement, la première architecture à qualité de service qui fût proposée pour l'Internet remonte à l'époque où l'idée d'intégration de service était étudiée avec les réseaux ATM. L'architecture à services intégrés appelée IntServ s'inspire de cette idée. Cette approche, proposée par l'IETF, est définie dans ses grandes lignes dès 1994 dans le RFC 1633

[1] Internet Engineering Task Force

[BCS94] dont la partie modèle de service est très largement inspirée de [CSZ92]. Cette architecture ne posséda pas le succès escompté à cause du manque de ce que les anglais appellent « *scalability* » (littéralement le passage à l'échelle). Cette technique de réservation de bout en bout céda la place à une autre approche dite à différenciation de services. Néanmoins, l'architecture IntServ a été remise au goût du jour par la percée des réseaux sans-fil. Cette approche étant considérée parfois comme la seule alternative possible concernant la réservation de ressources sur ces nouveaux réseaux à diffusion ou comme seule possibilité permettant la négociation de contrat de QoS entre réseaux filaires et sans-fil [RKV+01].

Le but du groupe de travail IntServ de l'IETF était la transformation de l'Internet actuel en un réseau à intégration de services. L'architecture IntServ s'organise autour du concept de flots de données correspondant à un ensemble de paquets résultant d'une application utilisatrice et ayant un besoin d'une certaine QoS. Afin de satisfaire la QoS requise, IntServ propose d'effectuer une réservation des ressources nécessaires à l'établissement de celle-ci via le protocole de réservation de ressources nommé RSVP. RSVP étant constitué par l'information de contrôle de la QoS, celui-ci propose des directives afin de mettre en place la réservation mais ne dit pas comment la mettre en place, ce domaine étant réservé aux routeurs du réseau qui prennent en compte la signalisation effectuée par le protocole RSVP. Pour se faire, les routeurs disposent de quatre fonctions de contrôle du trafic qui sont :

1. le protocole de réservation de ressources. Il sollicite, sur le chemin prédéfini par le protocole de routage, des réservations de ressources (bande passante et tampon mémoire) sur chaque routeur traversé du réseau ;
2. le contrôle d'admission. Il autorise ou refuse l'arrivée d'un nouveau flot muni de sa QoS sans perturber les QoS des autres flots existant ;
3. le classificateur de paquets. Il identifie les paquets des flots devant recevoir un service spécifique ;
4. l'ordonnanceur de paquets. Il détermine l'ordre de service des paquets.

2.2.1 RSVP

L'IETF a conçu le protocole RSVP pour la signalisation des besoins utilisateurs au sein d'un réseau Internet. Il peut être utilisé pour assurer la qualité de service et gérer les ressources de transport du réseau pour les sessions point à point autrement appelée session *unicast* et point à multipoint : *multicast*. RSVP est un système de contrôle et de signalisation qui donne la possibilité de réserver la bande passante nécessaire au bon fonctionnement d'une application. C'est un besoin qui touche principalement les flux multimédias, plus sensibles aux aléas de l'acheminement que les flux de données pures du fait de leurs contraintes temporelles.

Une session RSVP est identifiée par l'adresse du destinataire du flot de données (il s'agit d'une adresse IP *unicast* ou *multicast*), le numéro de port de destination et le protocole utilisé (TCP ou UDP). RSVP fonctionne avec trois types d'éléments : l'expéditeur, le réseau et le destinataire. L'expéditeur indique au réseau la caractérisation du flux qu'il va envoyer. Le réseau enregistre les demandes, les traite, notifie au destinataire que des messages sont en route. Le destinataire informe le réseau de ce qu'il est capable de recevoir et la fin de

la réception. C'est le destinataire du flot de données qui fait la demande de réservation en s'adressant à un processus RSVP local. La requête formulée, RSVP la transporte de routeur en routeur dans le sens inverse de celui que vont emprunter les données concernées. RSVP va maintenir un chemin dit à état mou (*soft state*) car temporaire. Ce dernier doit être rafraîchi régulièrement pour ne pas disparaître. RSVP ne traite pas le routage et à ce titre, il peut être aussi associé avec le protocole IP Multicast, mono ou multi-émetteurs, lorsque des échanges ont lieu au sein d'un groupe d'ordinateurs selon un monde analogue à celui utilisé dans le cas de la télévision ou de la radio. [Whi97] donne un très bon tutorial sur ce protocole.

2.3 LE MODÈLE DIFFSERV

En réaction aux limites et aux difficultés de déploiement du modèle IntServ (notamment en ce qui concerne son passage à l'échelle), un nouveau groupe de travail de l'IETF, *The Differentiated Services Working Group* ou *DiffServ* [BBC+98], a été chargé d'étudier une nouvelle approche, appelée la « différenciation de services ».

Au contraire du modèle IntServ qui traite indépendamment chaque flot, le modèle DiffServ propose de séparer le trafic par classes. Nous avons donc affaire à une granularité moins fine mais qui résiste plus au facteur d'échelle. En effet, la granularité par flot implique la réaction en chaîne suivante : plus il y a d'utilisateurs, plus il y a de flots dans le réseau et plus il y a d'états à maintenir au sein des routeurs. La conséquence directe est une augmentation de la charge des routeurs qui deviennent alors de moins en moins performants.

Cette architecture repose sur le principe de l'agrégation de flots de paquets en classes de services et de la gestion des ressources par classe plutôt que par flot individuel [BBC+98]. Un flot individuel consiste en une séquence de paquets issus d'une même source de trafic. Celui-ci peut être tout aussi bien un site ou une application. Ces flots sont regroupés en fonction du service requis pour former des agrégats de flots. Le nombre d'agrégats de flots dépend des classes de services offertes. L'agrégat de flots forme un macroflot. Par opposition, les flots individuels sont appelés des microflots.

2.3.1 Organisation d'un réseau DiffServ

L'architecture DiffServ distingue la frontière de l'intérieur d'un domaine d'administration. Un domaine se définit comme une portion contiguë de l'Internet contrôlée par une même autorité administrative. La frontière d'un domaine d'administration est marquée par un routeur de bordure. Ce routeur joue un rôle supplémentaire de celui situé au cœur du domaine : celui du conditionnement du trafic. Cette architecture s'inscrit dans le même paradigme que l'Internet qui est : « de reléguer la complexité dans les extrémités du réseau et de laisser le cœur du réseau aussi simple que possible ». Cette architecture conduit à procéder à un simple ordonnancement des agrégats de flots au cœur du réseau et au contrôle de flots individuels en bordure.

Le groupe DiffServ propose donc d'abandonner le traitement du trafic sous forme de flots pour le caractériser sous forme de classes de trafic[2]. Chaque classe est identifiée par une valeur codée dans l'en-tête IP. Cette classification doit se faire sur les routeurs de bordures (*edge router*) à l'entrée du réseau. Un exemple de domaine DiffServ est représenté sur la figure 2.1.

L'architecture des services différenciés proposée dans [BBC+98] contient deux types d'éléments fonctionnels :

1. Les **éléments de bordure** (*edge functions*) : ils sont responsables de la **classification des paquets et du conditionnement du trafic**. En bordure du domaine, c'est-à-dire à l'arrivée du premier élément actif capable de traiter le champ DS (*DS-capable*), les paquets arrivant ont dans leur champ TOS (*Type Of Service* pour IPv4) ou TCO (*Traffic Class Octet* pour IPv6), une certaine valeur DS. La marque d'un paquet identifie la classe de trafic auquel il appartient. Après son marquage, le paquet est dit conditionné ;
2. Les **éléments du cœur** (*core functions*) : ils sont responsables de l'**envoi** uniquement. Quand un paquet, marqué de son champ DS, arrive sur un routeur *DS-capable*, celui-ci est envoyé au prochain nœud selon le PHB (*Per Hop Behaviour*) associé à la classe de trafic. Le PHB influence la façon dont les tampons mémoires et la bande passante sont partagés parmi les différentes classes de trafic. Une chose importante dans l'architecture DiffServ est que les PHB routeurs se basent uniquement sur le marquage de paquets, c'est-à-dire la classe de trafic auquel le paquet appartient ; en aucun cas ils ne traiteront différemment des paquets de sources différentes.

Le principal avantage de cette architecture est qu'il n'y a plus nécessité de maintenir un état des sources et des destinations dans les routeurs de cœur, d'où un meilleur passage à l'échelle.

2.3.2 Classification et conditionnement du trafic

Le routeur de bordure a en charge la surveillance et le conditionnement du trafic entrant. Ces tâches sont complexes et mettent en jeu une grande variété de contextes. Elles servent à limiter la quantité de trafic injecté par chaque utilisateur dans sa classe de service. Elles sont essentielles et limitent l'apparition de la congestion dans la classe de service. Les contrôles sur le trafic entrant s'appliquent au niveau du flot utilisateur. La granularité du flot utilisateur et les paramètres du conditionnement de trafic sont décrits dans un profil (TCA : *Traffic Conditioning Agreement*). Le résultat du conditionnement se traduit concrètement par un marquage des paquets admis dans le domaine, par la suppression des paquets excédentaires, ou par la remise en forme du flot (assurer un espacement temporel entre les paquets).

La classification s'effectue suivant une ou plusieurs valeurs contenues dans l'en-tête IP (exemple : adresse source - destination, port source - destination, protocol ID, ...). Celle-ci faite, elle dirige le paquet vers la fonction de marquage appropriée. Une fois les paquets

[2][BBC+98] utilise le terme de *behaviour aggregate (BA)*

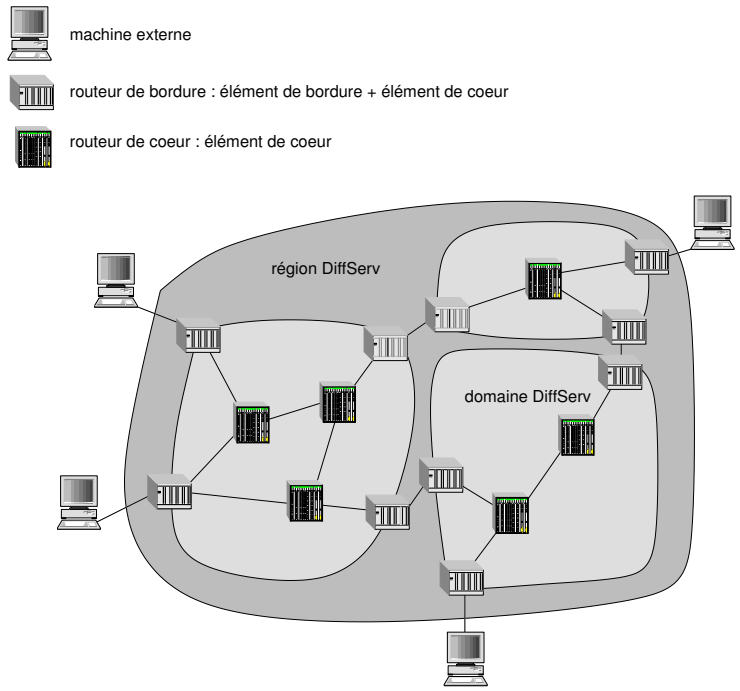


Fig. 2.1 Domaine DiffServ

marqués, ils sont envoyés à leur destination puis à chaque routeur *DS-capable*, ils reçoivent le service associé à leur classe.

Le classificateur est paramétré par les fonctions du plan de contrôle. Les tables de marquage des paquets sont configurées en fonction d'une table d'adresses sources donnée qui sera alors communiquée au routeur de bordure.

En plus de cette classification/marquage, un mécanisme de contrôle du trafic est défini par le groupe de travail DiffServ. Ce contrôle du trafic (*traffic profile*) a pour objet la prise en compte du taux de sortie des paquets afin de ne pas dépasser un seuil maximum de paquets à envoyer sur le réseau. Ainsi, un mécanisme de mesure du trafic permet de savoir si le flot de paquets sortant correspond au profil de trafic négocié. Si ce flot dépasse un certain seuil, certains paquets seront marqués comme moins prioritaires et seront automatiquement jetés en cas de congestion dans le réseau comme l'illustre la figure 2.2.

2.3.3 Les PHB

Le traitement des paquets par les routeurs du domaine est défini par le comportement de relayage (PHB : *Per-Hop Behavior*). La sélection du PHB est fonction de la marque contenue dans l'en-tête du paquet. En plus du comportement standard actuel dit DE (*Default*) utilisé par le service BE, deux PHB sont disponibles EF (*Expedited Forwarding*) [DCB+02] et AF (*Assured Forwarding*) [HBWW99]. Le PHB EF permet de réaliser un service de transfert à forte contrainte temporelle tandis que le PHB AF assure à certains paquets une protection contre la perte en cas de congestion.

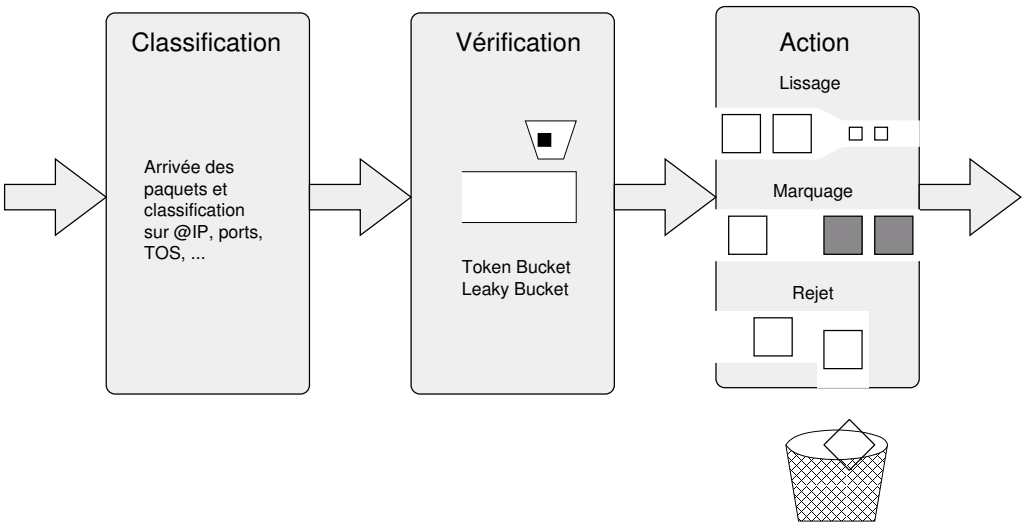


Fig. 2.2 Classification, marquage et conditionnement du trafic au niveau du *edge router*

Expedited forwarding

Le PHB EF permet de mettre en œuvre une classe de service surdimensionnée, au même titre qu'un lien privé virtuel, permettant d'obtenir des performances très supérieures à celle du classique réseau *best-effort*. Deux classes de traitements sont proposées : une classe à très faible délai et sans perte et une classe *best-effort* classique. Ce PHB doit offrir un service de bout en bout avec une garantie de bande passante à taux de perte, délai et gigue faible. Il a été tout d'abord défini d'une manière relativement intuitive dans [J+99]. Une étude analytique a montré que le service de bout en bout attendu était impossible à mettre en œuvre dans certains cas particuliers [BBC+01]. Sa définition a alors été remaniée de manière plus complexe et formelle dans [DCB+02]. Une bonne façon d'illustrer ce PHB est la mise en œuvre que l'on peut faire de ce service avec un ordonnanceur comme le *priority queuing* (voir section 3.3.2). La file EF est strictement prioritaire sur la file BE (*best-effort*). Dans [Fer00], une implémentation de ce type est comparée avec un autre ordonnanceur de paquets. Il est montré qu'en contrôlant la quantité de trafic prioritaire à laquelle les usagers sont abonnés, il est possible d'assurer que le trafic EF arrive en petite quantité sur les routeurs et qu'il ne sature pas le débit de sortie. Ainsi, la file BE reçoit, malgré sa priorité inférieure, un service satisfaisant.

Assured Forwarding

Il s'agit en fait non d'un PHB mais d'une famille de PHBs (*PHB group*). Quatre classes de « traitement assuré » sont définies, chacune comprenant trois niveaux de priorité spatiale (*drop precedence*) [HBWW99]. Dans un nœud donné, le niveau d'assurance de traitement d'un paquet dépend de la bande passante allouée à la classe, de la charge actuelle de la classe et de la priorité spatiale du paquet.

Chaque nœud doit mettre en œuvre des mécanismes visant à :

1. réduire la congestion de long terme dans la classe ;
2. accepter les rafales (congestion à court terme) ;
3. traiter identiquement les microflots ayant des débits moyens identiques (i.e. ne pas défavoriser un microflot *bursty*).

Ceci implique la mise en œuvre d'un algorithme de gestion active de la file, du type RED (*Random Early Detection*) [FJ93].

Le service fourni par chaque classe peut être modulé par le réglage des paramètres de cet algorithme en conjonction avec ceux du conditionnement de trafic opéré aux nœuds de bordure. Des « marqueurs à trois couleurs » sont proposés [HG99a, HG99b] pour diviser le trafic de chaque classe selon les trois priorités.

2.4 CONCLUSION DU CHAPITRE

Nous avons présenté dans ce chapitre deux architectures très différentes. Le modèle DiffServ se distingue du modèle IntServ de part son meilleur passage à l'échelle (*scalability*). En effet, en séparant le trafic en un nombre réduit de classes et en repoussant la complexité (classification et conditionnement) aux extrémités du réseau, ce modèle atteint l'objectif de passage à l'échelle pour le plan de données. Par contre, par rapport à IntServ il est clair que l'on perd soit en flexibilité soit en fermeté des garanties. De plus, à moins de se contenter d'une forte sous-utilisation du réseau (i.e. conditionner le trafic de manière très conservatrice), le problème du plan de contrôle reste entier. Dans le cas de trafic multicast, les choses se compliquent encore et aucune approche n'est proposée sur la façon de dimensionner le réseau et de conditionner ce type de trafic. Clairement, le groupe de travail se trouve confronté sur ce point à des problèmes très difficiles à résoudre relevant du domaine de la recherche.

Chapitre 3

Mécanismes utilisés pour le traitement des services différenciés

3.1 INTRODUCTION

La gestion des files d'attente est nécessaire pour équilibrer le trafic [BCC+98]. Elle évite la monopolisation d'une file d'attente par un seul flot. Une simple gestion de files n'évite pas qu'elles soient pleines pour des périodes longues, alors que pour avoir de faibles délais il est souhaitable que les files ne soient pas trop chargées. En effet, de petites files d'attente réduisent les délais de transmission. Les objectifs de la mémorisation du réseau sont d'absorber les pointes de trafic, de gérer les contentions d'accès (par exemple, lorsque deux paquets demandent à utiliser une ligne en même temps, les demandes sont alors sérialisées), de réguler les trafics issus de lignes à vitesse différente (par exemple, d'une ligne à 100Mbits/s vers une ligne à 10Mbits/s). En revanche, l'objectif du réseau à qualité de service n'est pas le même que celui du réseau dit « *best effort* ». Pour ces raisons, des mécanismes supplémentaires complètent la simple gestion de files d'attentes de sortie. Ces mécanismes permettent de diminuer le nombre de datagrammes éliminés et le délai de bout en bout ainsi que d'éviter le remplissage permanent des files d'attente et cela tout en gardant une bonne utilisation du réseau.

Le [BCC+98] définit deux types d'algorithmes de contrôle de congestion : la gestion des files et l'ordonnancement. Le premier gère la taille de la file en éliminant des paquets si nécessaire tandis que le second détermine le prochain paquet à envoyer sur le lien. Ces deux algorithmes sont utilisés pour gérer le temps d'accès au support et par conséquent, le débit d'un flot de données.

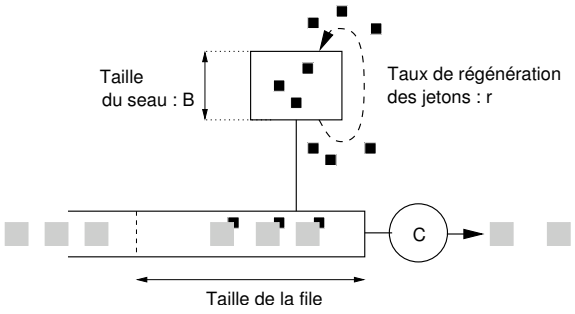


Fig. 3.1 *Token Bucket*

3.2 CARACTÉRISER UN FLOT

Le but de la caractérisation du trafic est de vérifier que la demande ne soit pas supérieure à la ressource disponible afin de ne pas saturer le réseau. Cette caractérisation va être ensuite utilisée par la politique de trafic qui se chargera d'autoriser ou de refuser l'envoi des paquets à l'intérieur du réseau. Dans un réseau à qualité de service, il est nécessaire de contrôler le trafic entrant afin d'assurer à chacun des flots la qualité demandée. Parmi les caractéristiques qui vont influer de façon significative sur le comportement du réseau, l'intensité et la sporadicité du trafic sont de première importance. Ce contrôle de trafic vise à limiter le volume et l'intensité du trafic émis par une source. Pour se faire, trois critères importants vont entrer en jeu :

1. le débit moyen : r ;
2. le débit crête : C ;
3. la taille maximum de la rafale : elle limite la quantité de données émise au débit crête : B.

Pour se faire, on va utiliser un mécanisme de caractérisation de trafic comme le seau à jetons (*token bucket*) afin de contrôler le débit moyen et la taille des rafales. Afin d'obtenir le débit crête, les implémentations actuelles sérialisent un second seau à jetons. Ainsi, on contrôlera le volume de trafic entrant dans le réseau et le débit avec lequel il est transmis. [Kes97] distingue le seau à jetons, utilisé pour caractériser un trafic au régulateur à seau percé (*leaky bucket regulator*) qui lui est un régulateur de trafic constitué d'un seau à jetons et de mémoires tampons.

Le principe du seau à jetons est le suivant : le trafic ne traverse pas directement le seau mais est transmis sur la base de jetons présents dans le seau. Ce mécanisme permet à un trafic en rafale d'être transmis tant qu'il y a des jetons dans la file d'attente, ceux-ci ayant pu être accumulés en situation de réseau peu chargé. Un jeton correspond à un nombre de bits donnés. Comme nous le montre la figure 3.1, un *token bucket* consiste en un seau de profondeur B (correspondant au nombre de jetons qu'il peut stocker), constamment rempli par des jetons ayant pour taux d'arrivée r (correspondant au débit moyen). Chaque jeton arrivant laisse sortir un paquet de données entrant de la file d'attente. Ce paquet est alors effacé du seau. Pour comprendre le résultat obtenu prenons l'exemple suivant : soit un paquet à envoyer de taille b avec $b < B$, si le seau contenant les jetons est :

1. plein : le paquet est envoyé et b jetons sont retirés du seau ;
2. vide : le paquet doit attendre que le seau se remplisse à nouveau de b jetons pour être envoyé ;
3. partiellement plein : il y a B jetons dans le seau. Si $b \leqslant B$, le paquet est envoyé immédiatement sinon, il doit attendre la régénération de $b - B$ jetons avant d'être envoyé.

On parle alors de flots (r, B) régulés.

3.3 LES ORDONNANCEURS

Dans le précédent chapitre, nous avons vu que le traitement différencié des paquets dans le modèle DiffServ correspond à la classification de flots en classes de service et à l'introduction de priorités à l'intérieur de ces flots. Cette partie va s'attacher aux algorithmes d'ordonnancement. Ils sont utilisés afin de contrôler la distribution de ressources entre les différentes classes de service. Plusieurs techniques ont été développées pour contrôler le partage de ressources, pour isoler des classes de service ou pour réduire le temps d'attente des paquets dans les files. Il en existe deux approches : l'approche mono ou multi files. Nous allons présenter dans cette partie les généralités de ces deux approches.

3.3.1 Les mécanismes à file unique, le *Fair Queuing*

Ces mécanismes se basent sur la notion d'estampilles temporelles qui déterminent le point d'insertion d'un paquet dans une file unique. Cette estampille est calculée à partir du poids attribué à la classe de service, de la longueur du paquet et de l'interaction avec les autres classes actives sur le lien.

Weighted Fair Queuing (WFQ)

Le modèle idéal du partage de débit entre flots/sessions est GPS : *Generalised Processor Sharing*, dans lequel on suppose pouvoir partager le débit disponible de façon fluide et continue. [PG92] définit ce modèle théorique de multiplexage de flots/sessions de la façon suivante : un serveur GPS conserve le travail (*work conserving*) et opère à un débit fixe C. Un serveur dit "conservateur du travail" signifie qu'il doit être occupé si il y a des paquets en attente dans le système. Il est caractérisé par des nombres réels positifs $\phi_1, \phi_2, ...\phi_n$. Soit $S_i(\tau, t)$ la quantité de service reçue par la session i dans l'intervalle de temps $[\tau, t]$. Le serveur GPS est défini tel que :

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geqslant \frac{\phi_i}{\phi_j} \text{ avec } , j = 1, 2, ...N \tag{3.1}$$

pour toutes sessions i possédant du trafic en attente sur l'intervalle $[\tau, t]$.
En sommant :

$$S_i(\tau, t)\ \phi_j \geq \phi_i\ S_j(\tau, t) \tag{3.2}$$

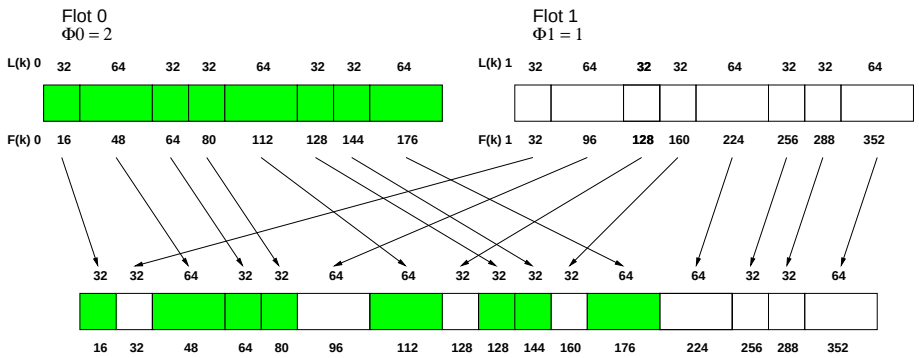


Fig. 3.2 Ordonnancement par *Weighted Fair Queuing*

on obtient pour toutes les sessions j :

$$S_i(\tau,t) \sum \phi_j \geq \phi_i \sum S_j(\tau,t) \tag{3.3}$$

$$S_i(\tau,t) \geq \frac{\phi_i}{\sum \phi_j} C\ (t-\tau) \tag{3.4}$$

Ainsi, une part de la bande passante est garantie à la session i :

$$g_i = \frac{\phi_i}{\sum \phi_j} C \tag{3.5}$$

Si l'on assure que $\sum \phi_j \leqslant 1$, par exemple grâce à un contrôle d'admission, cette garantie s'exprime alors en termes de bande passante absolue. Chaque classe dans ce modèle dispose d'un débit minimum garanti, et les classes excédant leur débit minimum garanti se partagent le débit excédentaire équitablement.

[PG92] proposent une émulation de GPS pour des réseaux réels, qui ne peuvent transmettre qu'un paquet à la fois. Cette émulation porte le nom de PGPS pour *packet by packet* GPS. L'idée est de servir les paquets en les triant sur leur date de fin de transmission sous GPS, c'est à dire que les paquets seront émis dans l'ordre où ils finissent (et non pas commencent) leur transmission avec un serveur GPS. Il va donc falloir trier les paquets avec une estampille temporelle correspondant à la date de leur fin de transmission en GPS.

WFQ [BZ96b], SCFQ [Gol94] ou encore SFQ [GVC96] sont des mécanismes qui se basent sur cette notion d'estampille temporelle. Cette estampille détermine le point d'insertion d'un paquet dans une file unique. Elle est calculée à partir du poids attribué à la classe de service, de la longueur du paquet et de l'interaction avec les autres classes actives sur le lien à partir de la formule suivante :

$$F_i^k = \frac{1}{\phi_i} L_i^k + F_i^{k-1} \text{ avec } F_0^i \text{ initialisé à zéro} \tag{3.6}$$

avec F_i^k l'estampille du $k^{ème}$ paquet du flot i, ϕ_i le poids attribué au flot et L_i^k la longueur du paquet. Les estampilles F_i^{k-1} et F_i^k correspondent respectivement aux dates auxquelles le paquet commence et termine son service. Le poids d'une classe, ϕ_i, détermine le pourcentage de bande passante que la classe se voit attribuer. On dit que les poids sont normalisés

si pour tout i on a $\sum \phi_i = 1$. La figure 3.2 donne une illustration de ce calcul pour deux flots numérotés 0 et 1 de poids respectifs ϕ_0 et ϕ_1 et d'estampilles temporelles respectives F^0 et F^1 . Le poids attribué à F^0 est le double de celui de F^1 et les paquets ont une taille identique : $L^0 = L^1$. Malgré la taille identique des paquets, le flot 1 a des estampilles deux fois plus grandes que le flot 0. L'état de la file représentée en bas de la figure montre que les paquets du flot 0 sont effectivement servis en premier. A long terme, le flot 0 doit observer un débit deux fois plus important que le flot 1.

La formule (3.6) présente deux défauts. Tout d'abord, elle ne prend pas en compte le temps d'arrivée des paquets. Si un flux reste inactif pendant que d'autres évoluent, la valeur des estampilles F_i^k est désynchronisée. Lors de son réveil, le flot se verra attribuer des estampilles de faible valeur et obtiendra une quantité de ressources ne correspondant plus à son poids ϕ_i. Deuxièmement, la formule ne prend pas en compte la présence d'autres flots, or cette information est indispensable pour calculer la quantité exacte de ressources qu'un flot peut obtenir. Des techniques plus avancées introduisent la notion de temps virtuel pour caractériser l'accélération du service quand certains flots ne sont pas actifs à un instant donné. La fonction de temps virtuel permet de définir le temps que les paquets restent dans une file d'attente. Elle définit l'accélération du service quand certaines classes de service sont absentes. Le temps virtuel, $v(\tau)$, est calculé de la façon suivante :

$$v(\tau') - v(\tau) = \frac{1}{\sum_{\phi\ actifs} \phi_i}(\tau' - \tau) \tag{3.7}$$

avec τ' etτ représentant des intervalles de temps où l'ensemble de flots actifs reste constant. Posons que le $k^{ème}$ paquet de la session i arrive au temps a_i^k et a pour taille L_i^k. Avec le serveur GPS, on a :

$$F_i^k = \frac{1}{\phi_i} L_i^k + max(F_i^{k-1}, v(a_i^k)) \tag{3.8}$$

PGPS et WFQ [BZ96b] sont le même algorithme donnant une approximation du GPS. Les auteurs du WFQ le concevaient comme une émulation d'un tourniquet bit à bit, aussi la notion de temps virtuel est remplacée par le numéro du cycle de service (à chaque cycle un bit est servi pour chaque session active). L'avantage de la discipline WFQ est qu'elle permet d'obtenir un degré d'équité. Il est possible d'obtenir une bande passante par connexion ainsi que des bornes de délais. Les rafales sont lissées et il n'y a pas besoin de faire un contrôle de trafic en amont [Kes97]. En revanche le calcul des estampilles est complexe, il est nécessaire d'avoir un état par connexion et il faut classer les paquets avant de les servir. De plus, de petites bornes de délais demandent une large réservation de bande passante.

WF^{2}Q

Une version améliorée de WFQ ajoute une condition pour éviter à certains flux de transmettre en rafale : dans la discipline WF^{2}Q [BZ96a] les paquets ne sont éligibles à la transmission que s'ils auraient déjà commencé leur transmission dans le modèle GPS. Elle garde les avantages de la discipline WFQ mais possède des bornes absolues de délai plus

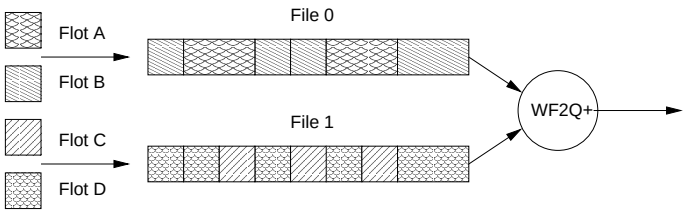


Fig. 3.3 *WF^2Q+ à files multiples*

petites qui la rend plus équitable que WFQ. Son principal désavantage est l'emploi d'un régulateur pour traiter les paquets inéligibles.

WF^2Q+

WF^2Q+ est un raffinement de WF^2Q. L'aspect clé est l'utilisation d'un système d'horloge virtuelle qui garantit toutes les propriétés de WF^2Q tout en simplifiant considérablement les calculs, les ramenant à $O(n)$ avec n : nombre de files. Il est décrit dans [BZ97].

Conclusion

La discipline WFQ est fréquemment implémentée ; on la trouve par exemple dans de nombreux commutateurs ATM comme ordonnanceur pour les conduits ABR. Elle pose toutefois deux problèmes : le temps de calcul, qui varie en $O(log(n))$ où n est le nombre de classes actives dans le lien [SV96] et le fait qu'elle peut introduire des rafales en « prenant de l'avance » pour certains flux sur l'idéal GPS. Enfin, l'insertion ordonnée des paquets dans une file unique est une opération coûteuse.

Afin de s'affranchir de la complexité de l'insertion ordonnée des paquets dans une file unique, les implémentations actuelles proposent de déporter le problème en utilisant la classification sur plusieurs files. Les nouvelles implémentations de WFQ, tout en gardant le principe d'estampille temporelle, utilisent plusieurs files d'attentes pour classer un ensemble de trafic particulier, ce qui augmente son efficacité. De plus, à cause des simplifications de calcul apportées par WF^2Q+, l'on préfère son utilisation à celui de WFQ. C'est le cas notamment de l'implémentation *dummynet* de Luigi Rizzo [Riz97]. Nous verrons en chapitre 4 que nous utiliserons une discipline WF^2Q+ à files multiples pour le traitement de la classe AF et DE (*Default*) dans notre implémentation d'architecture DiffServ. La figure 3.3 illustre le principe d'utilisation de la discipline WF^2Q+ à files multiples. Sur cette figure, les flots sont classifiés dans deux files distinctes : la file 0 et la file 1. La classification peut être faite par exemple sur deux adresses IP source différentes. L'estampillonnage temporel est alors réalisé sur ces deux flots. Il n'y a donc plus besoin d'un état par connexions entrantes car celles-ci se retrouvent agrégées sous forme de deux flots distincts.

3.3.2 Les mécanismes à files multiples, l'algorithme *Round Robin*

Le principe de l'algorithme *Round Robin* est très adapté à l'isolement de flux grâce au placement des paquets en plusieurs files d'attente. La figure 3.4 montre l'architecture de

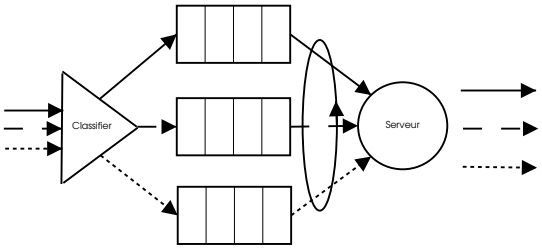


FIG. 3.4 Round Robin

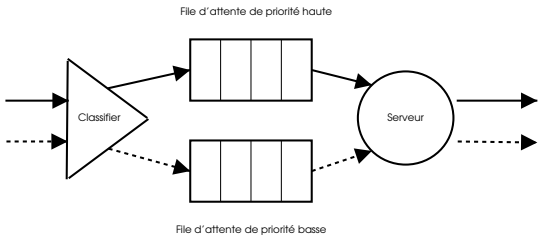


FIG. 3.5 Priority Queuing

base de l'algorithme *Round Robin*. La granularité est au niveau du paquet. L'ordonnanceur parcourt en séquence les files et prend le premier élément. Si une file est vide, il passe à la file suivante. L'équité offerte par ce mécanisme dépend de la taille moyenne des paquets dans chaque file : une file contenant des paquets deux fois plus gros que les autres obtient deux fois plus de bande passante. Il est donc nécessaire de modifier cet algorithme pour limiter le nombre d'octets que l'ordonnanceur peut retirer de chaque file afin d'assurer l'équité.

Priority Queuing

Avec ce type d'ordonnancement, les paquets arrivant sur le lien de sortie sont classifiés en une, deux, voire plusieurs classes sur la file de sortie. La classe d'un paquet dépend alors d'un marquage explicite se trouvant dans l'en-tête même de celui-ci. Par exemple, en prenant en compte le champ TOS d'IPv4, ou alors en prenant en compte les autres données présentes nativement dans l'en-tête comme l'adresse source/destination, le port source/destination, ou tout autre critère. Comme le montre la figure 3.5, chaque classe a sa propre file d'attente. Le serveur choisira d'abord un paquet se trouvant dans la file d'attente haute priorité si celle-ci est non vide, avant ceux se trouvant dans la file basse priorité. La granularité de classification se basant sur l'en-tête même du paquet est assez flexible. En revanche, ce système implique une dégradation des performances. Il peut y avoir un problème lorsque le trafic haute priorité est très important ; en effet, il peut y avoir rejet des paquets du trafic normal à cause de la taille de la file d'attente basse priorité. On parle alors de famine. [FC00] présente une étude comparative intéressante entre la discipline WFQ et PQ pour la classe EF.

Class Based Queuing

C'est une variation de WFQ qui utilise également un tourniquet sur plusieurs files mais un classificateur en amont de la batterie de file va s'intéresser à classifier chaque paquet en fonction de sa classe. Il n'y a donc plus de classification sur le flot. Grâce au *round robin* en sortie des files, on évite qu'une seule classe de trafic ne monopolise toutes les ressources.

Dans [FJ95], Sally Floyd et Van Jacobson proposent une architecture basée sur le partage de liens (*link sharing*) . Ce découpage de la bande passante peut être faite en prenant en compte :

– La famille de protocoles utilisés sur le lien,

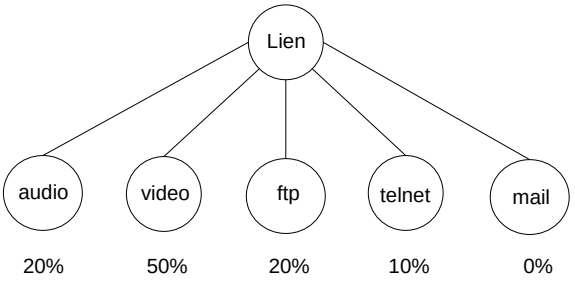


FIG. 3.6 Partage de lien entre plusieurs classes de services

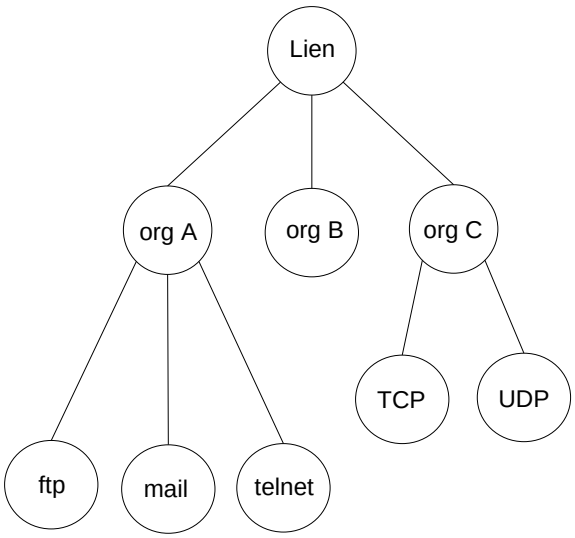


FIG. 3.7 Partage hiérarchisé d'un lien

– Les types de trafics applicatifs (telnet, ftp, mail, ...),
– Les différentes organisations partageant le lien.

Le but du *link sharing* est la classification de ces différents types de trafic afin d'opérer à un partage de la bande passante entre eux comme le montre la figure 3.6. Chaque classe, avec une demande correspondante à ses besoins, doit être en mesure de recevoir approximativement sa bande passante allouée durant un certain intervalle de temps lorsque le réseau est congestionné. Sur la figure 3.6, le *link-sharing* ne donne aucune garantie de bande passante pour le trafic mail en cas de congestion. Le partage peut également être hiérarchisé, par exemple, entre diverses organisations comme le montre la figure 3.7. Une part de la bande passante est donc attribuée à chaque niveau. La gestion des files met en œuvre les différentes variantes de gestion des listes de processus : priorité, temps partagé. En fait, pour les cas extrêmes, il est utile de pouvoir considérer qu'un certain type de trafic est éliminable. Les mécanismes de partage dans [FJ95] ne tentent pas de fournir un contrôle de congestion au niveau des feuilles de l'arbre (correspondant à une classe de trafic). Ces mécanismes étant à implémenter par l'ordonnanceur général à l'entrée du réseau. [WGC95] donne de plus amples explications sur l'implémentation de ces techniques.

Une autre approche consiste à ralentir les flux plutôt qu'à gérer les priorités ou la pénurie le cas échéant. Le ralentissement se commande à l'aide de messages ICMP de ralentissement émis depuis un routeur intermédiaire vers l'émetteur initial du message. Ces messages ne remontent cependant pas à l'application et ne sont donc pas forcément suivis d'effet, d'où l'idée du lisseur de trafic (*trafic shaper*) qui diminue la taille de la fenêtre d'anticipation dans l'en-tête TCP de façon à réduire le débit de certains flux. Le classement des trafics se fait selon différents critères : champ TOS, adresses IP source et/ou destination, numéros de ports UDP ou TCP qui permettent d'identifier les flux. On distingue les applications à trafic temps réel ou les applications du type messagerie au trafic moins urgent. Ces modes de gestion sont plus ou moins extensibles à des réseaux importants. Le trafic est géré par

le champ TOS pour les datagrammes qui circulent de façon indépendante. Selon [BCC+98] la notification et la prise en compte de la régulation par IP sont à recommander.

Clark-Shenker-Zhang

[CSZ92] propose une architecture pour les réseaux de type ISPN[1] supportant deux types distincts de services temps réel :

1. Les services garantis : c'est la forme traditionnelle des services temps réels bornés par des contraintes de délai correspondant au pire des cas ;
2. Les services prédictifs : ils utilisent des mesures de performance du réseau par le biais de calcul de bornes de délai d'acheminement de paquets.

Les concepts décrits dans l'article s'intéressent tout particulièrement aux services prédictifs ainsi qu'aux paramètres d'installation passés entre le réseau et la source. Notamment, l'interface de service doit inclure :

1. Une caractérisation de la source ;
2. Une description de la QoS que le réseau est en mesure d'offrir.

La dernière pièce de cette architecture est l'ordonnanceur de paquets utilisé.

L'idée principale de [CSZ92] est de créer des flots WFQ pour chaque service garanti et d'allouer le reste de la bande passante à un *flow-0*. Le *flow-0* comprend le trafic de services prédictifs et le trafic *best effort*. Ce flot est géré par un ordonnanceur de priorité attribuant la bande passante haute priorité au trafic prédictif, le reste étant partagé par le trafic *best effort*.

[CSZ92] utilise le mécanisme du *token bucket* afin de caractériser son trafic (voir la section 3.2). Une source de trafic est conforme à un *token bucket (r,b)* si il y a toujours assez de jetons dans le seau chaque fois qu'un paquet est généré. L'architecture [CSZ92] suppose également que les flots soient passés par un mécanisme de contrôle d'admission à l'entrée du réseau.

3.3.3 Prévention de la congestion : mécanismes de gestion de file d'attente

Par rapport aux mécanismes d'ordonnacement qui décident du paquet à transmettre, le rôle des mécanismes de gestion de file d'attente est de déterminer les paquets à éliminer au sein d'une même classe en cas de congestion. Le rejet d'un paquet est soit dû à la congestion ou à des techniques de prévention de la congestion. Il existe des propositions [Jai89, WC91] dans lesquelles le protocole de transmission aux extrémités contrôle la vitesse d'émission afin de réduire la taille des files dans les routeurs. Ces solutions sont difficiles à mettre en œuvre et leur efficacité est relative. Selon [FJ93], c'est au sein du routeur qu'est le meilleur endroit pour contrôler l'évolution des files d'attente. Nous allons décrire dans la partie suivante des mécanismes de gestion de file d'attente qui pourront être utilisés pour

[1] Integrated Services Packet Network

mettre en œuvre une politique d'élimination sélective nécessaire au comportement assuré du modèle DiffServ.

Drop Tail : DT

La gestion de file utilisée par défaut au sein des routeurs est : *Drop Tail* Les paquets sont mis dans la file de sortie et servis dans l'ordre avec lequel ils ont été reçus par le ou les interfaces d'entrée. C'est aussi la discipline la moins coûteuse en termes de traitement par le routeur. Cette technique est suffisante dans un réseau à forte capacité car nous pouvons considérer que les files restant presque toujours vides, les délais sont alors faibles, voire insignifiants. Par contre, dans le cas d'une rafale, la file d'attente peut se retrouver en débordement et les paquets arrivés après la rafale peuvent être jetés. Dans ce cas, les paquets jetés le sont de manière indifférenciée, sans prise en compte du type de trafic auquel ils correspondent. En utilisant des stratégies de mise en file d'attente différenciée, on peut permettre à certains types de trafic d'être privilégiés en détruisant certains paquets plutôt que d'autres.

Random Early Detection : RED

Ce mécanisme actif de gestion de file d'attente (AQM[2]) a pour objectif d'anticiper la congestion avant que cette dernière ne se produise. Pour se faire, il va :

- soit jeter aléatoirement des paquets par mesure préventive afin que TCP réduise sa transmission ;
- soit utiliser le *flag* ECN[3] [RFB01] pour indiquer à TCP que les paquets ont traversé un noeud en voie de congestion.

Le seuil à partir duquel on commence à jeter les paquets, ainsi que le taux de remplissage de la file, doivent être configurés par l'administrateur. Dans un routeur, cela reviendrait à mesurer le taux d'occupation de la mémoire.

Sa fonction d'élimination est basée sur la taille moyenne de la file, ce qui autorise des fluctuations instantanées nécessaires pour le traitement du trafic très variable. Dans sa définition [FJ93], RED n'impose aucune formule pour le calcul de l'occupation moyenne. Par contre, dans les exemples, un filtre passe bas est utilisé comme le montre l'algorithme de la figure 3.8.

Avec q : l'occupation instantanée de la file et Wq le poids attribué à chaque nouvelle mesure. Il faut remarquer que le code prend en compte les périodes pendant lesquelles la file est inactive, supposant que m paquets de taille minimale auraient pu être transmis pendant ce temps-là.

Toujours basé sur la moyenne (avg), RED utilise une fonction aléatoire croissante pour décider si un paquet doit être éliminé. Grâce à cette caractéristique, les éliminations successives sont évitées. Le rejet de plusieurs paquets appartenant à une même rafale peut forcer un flux TCP à rentrer dans la phase de *slow start*. Le rejet aléatoire est une solution

[2]Active Queue Management
[3]Explicit Congestion Notification

```
Initialisation:
avg = 0
A l'arrivée de chaque paquet :
   SI la file n'est pas vide
    avg = (1 - Wq)avg + Wq q
   SINON
    m = temps pendant lequel la file était vide
                  m
    avg = (1 - Wq)  avg
```

Fig. 3.8 Calcul de l'occupation moyenne de la file dans RED

équitable puisque la probabilité pour qu'un flux subisse une perte est directement proportionnelle au pourcentage d'occupation de ce flux dans la file. Dans RED, l'élimination aléatoire se réalise uniquement lorsque la taille moyenne de la file se trouve entre deux seuils *thmin* et *thmax* comme le montre l'algorithme de la figure 3.9

```
SI thmin <= avg <= thmax
   count = count + 1
   Pb = Pmax(avg - thmin)/(thmax - thmin)
   Pa = Pb / (1 - count * Pb)
   avec probabilité Pa :
        éliminer le paquet
        count = 0

SI thmax <= avg
   éliminer le paquet
   count = 0
```

Fig. 3.9 Calcul de l'occupation moyenne de la file dans RED

La variable *count* compte les paquets acceptés dans la file depuis la dernière élimination. Cette variable réduit considérablement le risque d'éliminer deux paquets consécutifs. Dans le code, Pa représente la probabilité de rejet du paquet. Elle augmente linéairement de 0 à $Pmax$ pendant que la moyenne varie de *thmin* à *thmax* comme le montre la figure 3.10. A partir de *thmax*, tous les paquets sont éliminés. Si les sources réduisent leur débit en conséquence, la moyenne revient rapidement à un niveau acceptable.

Le mode de fonctionnement de RED a été utilisé dans la conception d'autres méthodes visant à améliorer la performance de TCP, en particulier le drapeau ECN. Avec cette méthode une source TCP est informée de la congestion grâce à un drapeau dans l'en-tête IP. Dans ce cas, RED n'élimine pas les paquets, il se contente d'activer le drapeau pour indiquer que le débit doit être réduit. Le récepteur du paquet TCP reflète la valeur de ce drapeau dans les acquittements pour que la source réagisse à cette demande. L'ECN est un mécanisme orthogonal à DiffServ. [RFB01] détaille son utilisation.

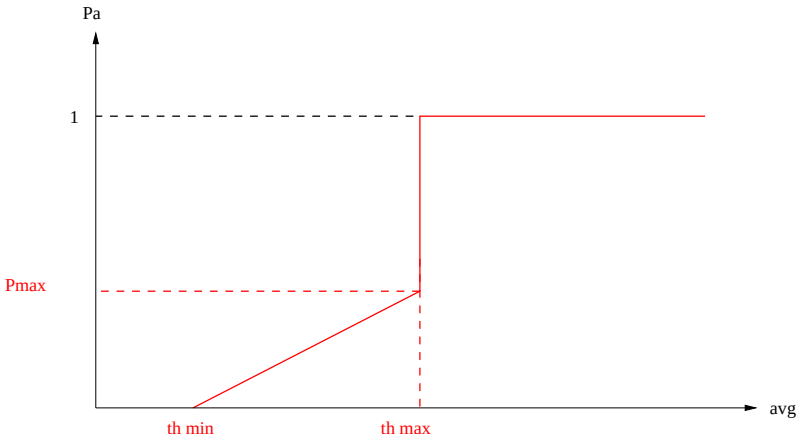


Fig. 3.10 *RED*

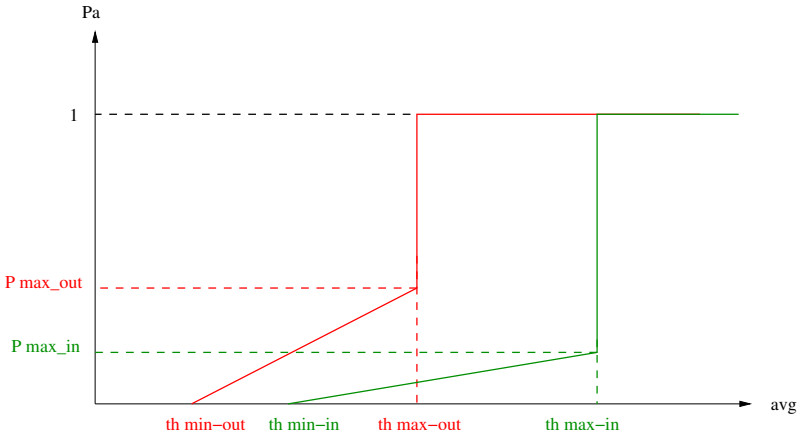


Fig. 3.11 *RIO*

[BCC+98] recommande l'installation sur les routeurs du mécanisme RED et la définition de mécanisme de gestion des flux non régulés. Le couplage de l'algorithme RED à un mécanisme de gestion de priorité du type de celui défini dans DiffServ pour le choix des datagrammes à éliminer est évoqué.

Random Early Detection IN OUT : RIO

RIO est l'un des premiers algorithmes à avoir été proposé pour effectuer une différenciation de services. Dans [CF98], il est décrit toute une architecture qui a servi de référence aux travaux de l'IETF. A l'origine, RIO était conçu pour faire de la différenciation entre deux niveaux de priorité à la perte (IN et OUT). Actuellement, le modèle a été étendu pour supporter plusieurs niveaux. Dans RIO, la différenciation se base en partie sur le calcul de l'occupation moyenne de la file. A la différence de WRED, l'algorithme calcule une moyenne par priorité à la perte (*drop precedence*). La moyenne avg_n est actualisée à chaque fois qu'un paquet de priorité à la perte égale ou inférieure à n arrive dans la file. Pour le cas de DiffServ, avg_0 reflète la moyenne des paquets de basse priorité à la perte présents

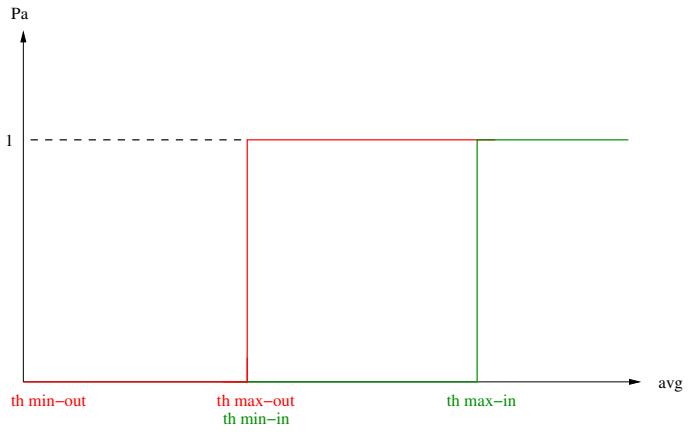


Fig. 3.12 *PBS*

dans la file ; avg_1, la somme des paquets de priorité à la perte basse et moyenne ; enfin, avg_2, comme dans RED, prend en considération tous les paquets qui attendent dans la file. Ceci permet d'isoler la probabilité de rejet pour chaque niveau. Pour les paquets de basse priorité à la perte, seul un excès de paquets de la même priorité à la perte justifie une élimination. Pour les paquets de haute priorité à la perte, l'excès de paquets de priorité à la perte moins forte suffit pour qu'ils se soient rejetés. [CF98] proposent aussi d'utiliser des paramètres RED ($pmax$, $thmin$ et $thmax$) différents pour chaque niveau. La figure 3.11 illustre la mise en place de ces seuils. Des valeurs similaires à celles qui seraient utilisées dans WRED pourraient être utilisées. Le calcul différencié des moyennes ajouté au paramétrage de RED font de RIO un algorithme très performant pour l'élimination basée sur la priorité à la perte.

Weighted Random Early Detection : WRED

Même technique que la précédente mais qui permet de déterminer quel trafic on jette grâce à la prise en compte du champ IP Precedence par les routeurs et cela afin d'éviter la monopolisation des ressources par certaines classes de trafic. Cette pondération gérée via ce champ permet au réseau de demander aux flux de trafic moins prioritaires de s'adapter au profit du trafic plus prioritaire. L'algorithme WRED est expliqué dans [Sys01].

Partial Buffer Sharing : PBS

La file PBS [PF91], dont le principe de seuil est illustré figure 3.12 est un cas particulier de configuration de la file RIO. Des études ont montré que la politique de gestion de l'attente RED introduit de l'instabilité et des oscillations [BMB00, ZFH99]. Cette solution repose sur une politique déterministe mise en œuvre par un mécanisme à simple seuil. Les paquets opportunistes sont systématiquement rejetés dès que le nombre total de paquets en attente dans la file excède un certain seuil.

Push Out : PO

A l'opposé des mécanismes à seuil, le mécanisme *Push Out* [HG90] éjecte les paquets non prioritaires qu'une fois la file d'attente pleine. Dans ce cas, les nouveaux paquets non prioritaires sont immédiatement rejetés tandis que les nouveaux paquets prioritaires prennent la place des paquets non prioritaires déjà présents dans la file. Les paquets prioritaires sont alors rejetés lorsque la capacité maximale de la file d'attente est atteinte et qu'aucun autre paquet non prioritaire est présent dans la file.

3.4 CONCLUSION DU CHAPITRE

Nous avons décrit dans ce chapitre les mécanismes réalisant des actions nécessaires au support de la qualité de service dans le réseau. Tout d'abord, l'identification et la caractérisation d'un flot va permettre de déterminer la conformité de ce dernier vis-à-vis de son contrat de service. Puis, la mise en place des mécanismes de gestion de file d'attente et d'ordonnancement va tenter de satisfaire les besoins de qualité de service requise par le flot. La mise en place, la configuration ainsi que les interactions de ces mécanismes dans le réseau définissent un « modèle » de qualité de service. Dans le modèle IntServ, la configuration de ces mécanismes est faite dynamiquement à partir des contraintes de qualité de service que les flots vont signaler. Encore un problème de résistance au facteur d'échelle. Au contraire, le modèle DiffServ définit une configuration statique de ces mécanismes faisant suite à une identification plus générale des besoins en qualité de service. C'est à partir de ce modèle que nous allons proposer l'architecture développée dans le chapitre suivant.

Chapitre 4

Mise en œuvre d'une architecture DiffServ : Projet AIRS

4.1 INTRODUCTION

Nous allons décrire dans ce chapitre la mise en œuvre d'un domaine DiffServ réalisée au sein du projet @irs[1]. Nous détaillerons plus particulièrement la structure interne d'un routeur de cœur et d'un routeur de bordure gérant la qualité de service. Tout d'abord, un développement basé sur la pile IPv6 Musica IP a été effectué afin d'inclure les mécanismes de qualité de service choisis pour le projet. Enfin, une interface utilisatrice, permettant de piloter ces modules, a été développée. La syntaxe des commandes est inspirée du code ADServ [Med00] développé pour l'interface ALTQ [Cho99].

4.2 PARTICULARITÉS MATÉRIELLES ET LOGICIELLES DE LA PLATEFORME

Le premier point marquant du projet est l'utilisation du protocole IPv6 en lieu et place de la version IPv4, ceci afin de bénéficier des nombreuses évolutions associées : capacité d'adressage, de routage, de configuration automatique, d'identification de flots, ... Afin de privilégier une portabilité maximum des développements effectués, une souche IPv6 commune aux deux systèmes utilisés (FreeBSD et Windows NT) a été choisie. Il s'agit de Musica IP développée par la société 6WIND. La totalité des travaux réalisés reposant sur cette souche, nous la détaillerons plus précisément, et nous verrons en quoi cette dernière a influé la mise en œuvre de notre architecture. Au niveau de la plateforme matérielle mise en place au LIP6, on trouve la totalité des machines impliquées dans les différentes expérimentations :

– commutateur ATM relié à Renater 2 ;

[1] `http://www.lip6.fr/airs`

– routeur de cœur sous FreeBSD 3.5 Musica ;
– routeur de bordure sous FreeBSD 3.5 Musica ;
– machine FreeBSD 4.x génératrice de trafic BE (DNS, FTP, Web, ...) ;
– machine de test FreeBSD 3.5 Musica ;
– machine Windows pour applications utilisatrices de QoS.

4.3 MUSICA IP ET LA STRUCTURE D'ACCUEIL

4.3.1 Musica IP

Musica IP est une pile protocolaire implémentant IP et ses protocoles associés (ICMP, TCP, UDP, ...) en versions 4 et 6. La version Windows NT de Musica se présente sous la forme classique d'un protocole réseau, intégrable et activable par les « liaisons » adéquates. Certaines précautions sont à prévoir quant au partage d'adresses IPv4 entre le protocole TCP/IP fourni par Microsoft et Musica. La version FreeBSD de Musica se présente sous la forme d'un ensemble de fichiers objet venant remplacer et s'ajouter au produit de la compilation d'un noyau. Ainsi, contrairement à la version Windows NT, on ne trouve pas dans cette version de difficultés liées à la coexistence de deux implémentations différentes des mêmes protocoles. La structure d'accueil Musica étant conçue pour la version FreeBSD nous ne nous intéresserons à présent qu'à cette dernière.

4.3.2 La structure d'accueil

Les objectifs de Musica pour ses créateurs sont la portabilité, démontrée par l'unification des versions FreeBSD et Windows NT, et la possibilité d'introduire des « modules » altérant les datagrammes en entrée ou sortie de la couche IP. Ces modules sont intégrés à une « structure d'accueil » qui assure l'interface avec Musica, et encapsule le code spécifique dans un module noyau chargeable dynamiquement (*KLM : kernel loadable module* en terminologie BSD). Un schéma des machines équipées de la structure est donnée en figure 4.1. On dispose de bibliothèques additionnelles pour réaliser des programmes utilisateur s'interfaçant avec le module ; c'est de cette façon que les utilitaires de configuration peuvent fixer les paramètres de qualité de service et obtenir des statistiques.

4.3.3 Trajet suivi par un datagramme

Les datagrammes IP parviennent au module QoS par l'intermédiaire de la structure d'accueil. La figure 4.2 nous montre le trajet emprunté par ces datagrammes. Certains points intéressants sont à noter :

– les datagrammes IPv4 et IPv6 sont « mélangés », ce qui impose d'examiner leur entête pour connaître la version ;
– il n'y a pas de notion d'instance d'un module ; ainsi il intercepte automatiquement l'ensemble des datagrammes entrant ou sortant de la couche IP, quelle que soit l'interface réseau concernée ;

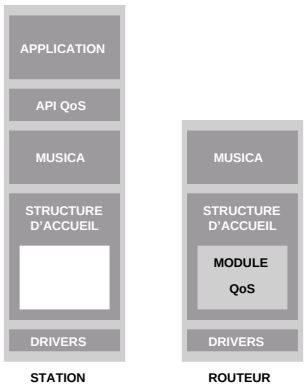


Fig. 4.1 Structure fonctionnelle d'une machine @IRS

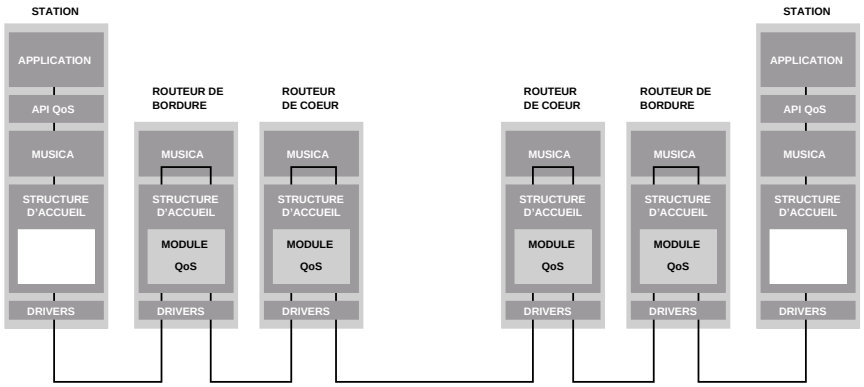


Fig. 4.2 Chemin de données entre 2 stations dans l'architecture @IRS

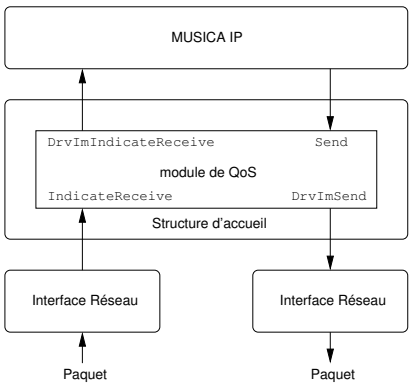


Fig. 4.3 Détail du module de QoS

– les pilotes d'interface réseau ont généralement une file d'attente pour les trames en attente d'émission ; la structure d'accueil ne permet pas de savoir si cette file d'attente peut encore accepter des trames, ce qui introduit donc un comportement de perte indifférenciée en cas de surcharge de l'interface. Nous verrons plus loin les conséquences de ce comportement et la façon dont un autre système de QoS : ALTQ [Cho99] l'a résolu.

Le module traitera les datagrammes dès leur réception suite à un appel aux fonctions d'entrée suivantes : `Send` ou `IndicateReceive`. Il peut alors les modifier, les supprimer, les conserver, voire les laisser à l'identique avant de les faire sortir du module grâce à un appel aux fonctions de sortie suivantes : `DrvImSend` ou `DrvImIndicateReceive`. Comme montré sur la figure 4.3, les fonctions `IndicateReceive` et `DrvImSend` sont utilisées pour communiquer avec l'interface réseau tandis que `DrvImIndicateReceive` et `Send` sont utilisées pour communiquer avec la pile Musica IP.

4.3.4 Interfaces

Le module n'a pas connaissance de la liste des interfaces réseau configurées sur la machine, mais peut savoir pour chaque datagramme qu'il est amené à traiter quelle est son interface source ou destination (selon qu'il est montant ou descendant). Ainsi, pour définir le comportement de qualité de service spécifique à chaque interface, nous devons obtenir son nom dans le module et identifier les paquets concernés par une comparaison textuelle.

4.4 STRUCTURE D'UN ROUTEUR GÉRANT LA QOS

La structure fonctionnelle d'un routeur IP peut se représenter selon la figure 4.4.

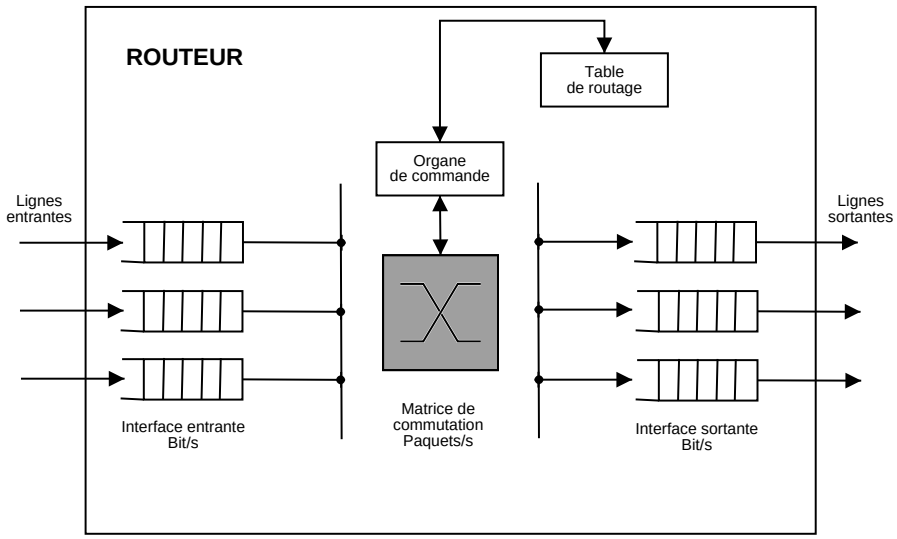


Fig. 4.4 Structure fonctionnelle d'un routeur

Les fonctions de QoS de niveau IP résident dans les interfaces d'entrée et de sortie du routeur. Les fonctions mises en œuvre dépendent du rôle du routeur dans l'architecture (c.à.d. de bordure ou de cœur). Un routeur de cœur comportera uniquement les fonctions liées au relayage, localisées dans l'interface de sortie. Ces fonctions visent à allouer efficacement les ressources que sont la bande passante et la mémoire de la file d'attente en fonction du comportement de relayage propre à chaque classe de paquets. Un routeur bien dimensionné suppose que la capacité de commutation de la matrice est supérieure à la somme des débits entrants. Dans ces conditions, la file d'attente des interfaces d'entrée ne peut être saturée. La congestion se manifeste donc uniquement sur la file d'attente de l'interface de sortie. Il est donc de la responsabilité des fonctions de relayage de protéger les flots à QoS de la congestion afin de maintenir un niveau de service stable. En plus des fonctions de relayage qui sont identiques aux fonctions d'un routeur de cœur, le routeur de bordure conditionne le trafic entrant dans le domaine. Ces dernières fonctions sont placées dans l'interface d'entrée. L'interface d'entrée en question est celle qui reçoit le trafic de l'extérieur du domaine. Cette interface peut être qualifiée d'interface amont.

La démarche prise dans la définition de l'architecture de la plateforme d'expérimentation va consister à identifier un jeu de mécanismes cohérents afin de fournir simplement des services conformes à la spécification. Les paragraphes suivants décrivent les mécanismes de gestion de la QoS pour spécialiser le rôle des routeurs selon leur place dans le domaine.

4.4.1 Elément de bordure

L'interface réseau amont se trouve exclusivement dans le routeur de bordure en entrée du domaine. Elle a en charge le filtrage du trafic entrant. Les éléments fonctionnels de l'interface amont sont représentés par la figure 4.5. Les règles et les paramètres du conditionnement du trafic d'un utilisateur sont décrits dans le TCA. On y trouvera les règles de filtrage pour l'identification de l'utilisateur du service, le profil du trafic et les actions pour le trafic soumis en excès. La classification multicritère (*Multi-field : MF*) repose sur les règles de filtrage. Dans le cas d'IPv6, ce travail d'identification est grandement facilité par le champ *flow label* présent dans l'en-tête des datagrammes. En effet, un flot applicatif est identifié de manière unique par la paire (*source address*, *flow label*).

Concernant les flots GS, le profil de trafic est constitué uniquement d'un débit crête. La vérification du respect du TCA consiste alors simplement en un métreur-espaceur, qui n'est autre qu'une file d'attente de taille finie, dont la vitesse du serveur correspond à ce débit crête. Les paquets acceptés dans la file sont marqués EF, les paquets non conformes sont rejetés. La file d'attente permet d'absorber des rafales de paquets en retardant ces paquets afin de respecter le débit crête ; il faut cependant que la taille de cette file d'attente ne soit pas trop grande pour que le délai d'attente maximal dans cette file soit acceptable.

Concernant les flots AS, le profil d'un flot est défini par un débit minimum et par une sporadicité. La sporadicité sert à ajouter une tolérance au débit (en termes de variation) et est représentée par la taille de la rafale maximale autorisée. Selon cette définition, la sporadicité est exprimée par une quantité de données. Le contrôle de conformité du flot par rapport au profil se fait par un *token bucket* (voir section 3.2). Les paquets sont marqués AF et une priorité spatiale leur est également attribuée en fonction de leur conformité au

profil. Les paquets détectés conformes reçoivent une priorité à la perte faible (marque IN) et les paquets non conformes ont une indication de forte priorité à la perte (marque OUT). Ces derniers deviennent des paquets opportunistes. La priorité à la perte est utilisée quand le paquet passe par des routeurs saturés. En l'absence de congestion, cette priorité ne sert pas.

Concernant les flots BE, aucun contrôle particulier n'est effectué : tous les paquets sont marqués DE (*Default*). Le métreur n'a qu'un rôle d'information pour l'administration de réseau.

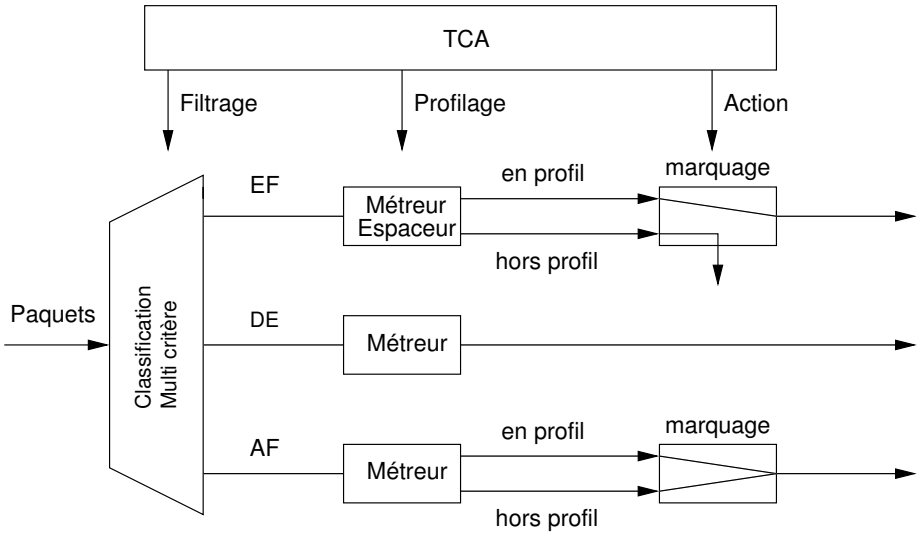


Fig. 4.5 Éléments fonctionnels d'une interface d'entrée d'un routeur de bordure

Marquage des paquets

Comme mentionné ci-dessus, tous les paquets entrants non rejetés seront marqués au niveau de l'en-tête IPv6 dans le champ DS[2].

Le document [NBBB98] présente le format du champ DS. Seuls 6 bits nommés DSCP *(Differentiated Services Codepoint)* servent au marquage, puis à la sélection du PHB (dans les routeurs de cœur). Par souci de compatibilité avec la précédence IP, le DSCP contient une priorité relative codée sur les 3 premiers bits. Ces bits ont un rôle de *Class Selector Codepoints* et identifient la rapidité de relayage relative d'un PHB par rapport à un autre. Le DSCP a le format indiqué par la figure 4.6.

Les valeurs de DSCP retenues pour les PHB utilisés dans @IRS sont présentées dans le tableau 4.1. Elles sont conformes à [HBWW99].

[2]Differentiated Services Field (DS Field)

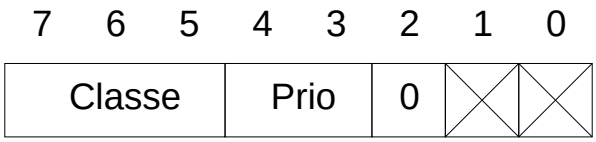


Fig. 4.6 Champ DSCP

PHB	Classe de trafic	Priorité spatiale
EF	101	110
AF	001	conforme : 010 non conforme : 100

Tab. 4.1 Valeurs en binaire du champ DSCP

4.4.2 Element de cœur

L'interface de sortie d'un routeur @IRS comprend l'ensemble des mécanismes de relayage. Ces mécanismes ont en charge l'ordonnancement des paquets et la gestion des pertes en cas de congestion. La figure 4.7 décrit la solution retenue et développée. A l'entrée de l'interface, les paquets sont classés dans l'une des trois files d'attente. Cette classification est effectuée en fonction de la marque attribuée par le conditionnement lors de leur entrée dans le domaine. Une telle classification est dite BA (*Behavior Aggregate*), ou autrement dit selon la classe de service. La notion de flot individuel n'existe plus.

La politique d'ordonnancement s'appuie sur 2 ordonnanceurs mis en étage afin de pouvoir gérer les trois PHB de l'architecture. Le PHB EF de la classe GS est obtenu par une priorité stricte (non préemptive) notée PQ (*Priority Queuing*), ce mécanisme assure un traitement rapide des paquets marqués EF. Le délai de paquétisation (retard induit par le fonctionnement en paquet par rapport à un modèle fluide) est de $Lmax/C$ où $Lmax$ est la taille maximum d'un paquet EF et C la capacité du lien. Un ordonnancement unique de type WFQ (*Weighted Fair Queuing*) introduirait un délai de paquétisation de $Lmax/R$ où R est la bande passante allouée au PHB AF (avec $R < C$). La motivation de PQ se trouve dans la faiblesse du délai de paquétisation ajoutée par rapport à WFQ. Le PHB EF est destiné à un service exigeant sur les délais. [Fer00] montre que le mécanisme du PQ donne le relayage le plus rapide pour le PHB EF. Il protège le trafic de la classe GS du trafic des classes AS et BE, à l'exception du cas où il faut qu'un paquet de la classe GS attende la fin du service en cours d'un paquet d'une autre classe.

Ce délai dû à la non préemption des paquets en cours de transmission est connu sous le nom de délai de sérialisation. L'utilisation d'un PQ doit se compléter d'un contrôle d'admission afin d'éviter que le service GS entraîne une famine sur les autres services et plus généralement pour éviter que la charge dans le service GS soit importante et qu'il se transforme en fin de compte en un service BE bis. La fonction de contrôle d'admission se situe dans le plan de contrôle qui n'est pas dans les objectifs du projet. Le reste de la bande passante non utilisée par GS est partagé équitablement entre les services AS et DE. Ce partage est effectué par un WFQ selon l'algorithme WF^2Q [BZ96a].

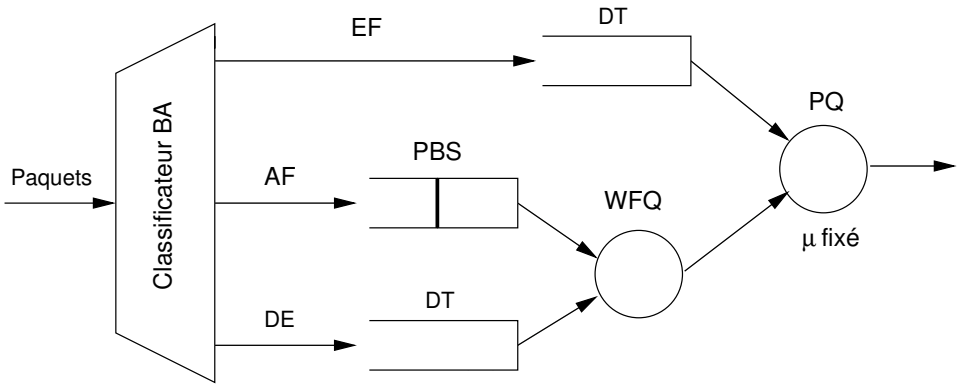


Fig. 4.7 Éléments fonctionnels d'une interface de sortie

Le taux de service μ de la discipline de service PQ représenté sur la figure 4.7 sert à régler le débit d'émission. Le débit d'émission de l'interface est ainsi configurable et indépendant du débit du support. Cette fonction trouve son utilité dans les expérimentations pour faire apparaître un goulot d'étranglement par rapport à la charge de trafic offerte. L'autre utilité consiste, dans le cas de support ATM, à adapter le débit d'émission au débit du lien logique qui est soit un VC (*Virtual Channel*), soit un VP (*Virtual Path*). Le taux de service μ est celui du lien logique.

Des pertes de paquets apparaissent lorsque la file d'attente arrive à saturation. La politique de perte par défaut notée DT (*DropTail*) consiste à éliminer le paquet trouvant une file pleine à son arrivée. Cette politique est appliquée aux PHB DE et EF, bien que dans le cas EF cette situation ne doit pas exister. Concernant le PHB AF, une différenciation est faite entre paquets conformes et paquets non conformes. La discrimination de la perte selon une politique probabiliste telle que RED (*Random Early Discard*) est difficile à mettre en oeuvre principalement à cause du grand nombre de paramètres à régler. De plus, des études ont montré que la politique de gestion de l'attente RED introduit de l'instabilité et des oscillations [BMB00] [ZFH99]. Pour ces raisons, son utilisation dans les routeurs reste problématique [MBDL99]. Notre solution repose sur une politique déterministe mise en oeuvre par un mécanisme à simple seuil de type PBS (*Partial Buffer Sharing*). Les paquets opportunistes sont systématiquement rejetés dès que le nombre total de paquets en attente dans la file excède un certain seuil.

Sortie de file découplée et limitation de débit

La mise en file et la sortie de file sont deux opérations découplées. Les paquets sont mis en file lors de leur entrée dans le module QoS et sortent au rythme défini pour l'interface. Il n'existe aucune relation entre ces deux rythmes.

Nous avons vu que Musica gère l'envoi des paquets depuis un module de la structure d'accueil par le biais d'un appel à `DrvImSend` ; cette fonction achève le traitement du paquet et le transmet au protocole de la couche liaison correspondant à l'interface. Une caractéristique partagée par presque tous les pilotes de niveau liaison est l'existence d'une

file d'attente en émission[3]. Il est important de noter que la demande d'émission d'un paquet n'est pas liée à la capacité d'acceptation des couches réseau sous-jacentes.

Le bon fonctionnement de l'ordonnancement QoS nécessite d'être certain de la bonne émission des paquets ; de plus, son utilité ne se révèle que lorsque l'interface de sortie est saturée (en effet un réseau peu chargé n'a pas vraiment besoin de QoS). C'est précisément dans ce cas que la file d'attente de l'interface réintroduit des pertes imprévisibles. Il faut donc trouver le moyen de ne pas envoyer plus de paquets que ce que l'interface peut effectivement émettre.

La meilleure solution serait d'être « conscient » de la disponibilité de l'interface, et donc de pouvoir envoyer un paquet exactement quand il le faut. Cette solution va totalement à l'encontre de la logique de fonctionnement de la structure d'accueil Musica. Il peut être intéressant de noter que la principale implémentation existante de cette solution sur FreeBSD, à savoir ALTQ, a nécessité l'adaptation de tous les drivers réseau supportés pour fonctionner, ce qui suppose un travail titanesque.

L'autre solution consiste à envoyer des paquets à un débit artificiel et inférieur au débit de l'interface. D'où l'emploi d'un limiteur de débit crête, qui par un appel de fonction périodique, transmet des paquets à concurrence du débit souhaité. Celui-ci est symbolisé par le taux μ sur la figure 4.7.

4.4.3 Gestion et influence du temps sur le module QoS

Le temps apparaît en de nombreux points dans la sémantique de la qualité de service. Le contrôle d'accès et la limitation de débit sont les deux principaux exemples de mesures relatives au temps. La latence est aussi une mesure fonction du temps. C'est à ce titre que l'ensemble du parcours des paquets se trouve soumis à certaines contraintes temporelles.

Descripteur de trafic et mesure du temps

On trouve des descripteurs de trafic pour le marquage et la remise en forme des flots dans le cadre du routeur de bordure. Une mesure fine et précise du temps est nécessaire pour qu'ils puissent fonctionner correctement. Dans le contexte noyau du module QoS, nous utilisons la fonction `microtime`. Cette fonction étant suffisamment précise pour les besoins de l'experimentation.

Remise en forme au niveau IP

Les mécanismes de remise en forme du trafic, que ce soit sur des flots GS individuels dans les routeurs de bordure ou sur l'ensemble du trafic sortant vers une interface pour un routeur de cœur, nécessitent d'introduire des délais entre les paquets. Pour se faire, la technique choisie est l'utilisation du mécanisme de *timeout* système, qui permet de voir une fonction choisie appelée après un certain nombre de *ticks* système.

[3] c'est en particulier le cas pour tous les pilotes Ethernet et ATM de FreeBSD, et donc toutes les interfaces utilisées sur le réseau @IRS

Ces *ticks* sont définis au sein du noyau ; par défaut fixés à 100 Hz, leur granularité est insuffisante. Un bon fonctionnement des remises en forme (c'est-à-dire transmettre des rafales aussi courtes que possible) nécessite une fréquence plus élevée. Pour la plateforme @IRS, nous avons constaté un comportement satisfaisant à partir de 1 kHz, qui s'avère être aussi la fréquence recommandée pour le bon fonctionnement des disciplines d'ordonnancements de ALTQ [Cho99].

La structure d'accueil de Musica réalise une part importante de ses traitements dans un contexte d'interruption système ; dans ces circonstances, les rappels *timeout* sont retardés. Cette difficulté peut être ressentie lorsque le trafic entrant est très important (par exemple en saturant deux liaisons entrantes *fast ethernet*) ; la latence en sortie devient alors plus importante, et des pertes de paquets se font sentir. Cette difficulté doit en fait plutôt être vue comme une conséquence de la performance médiocre de l'assemblage structure d'accueil plus module, tournant sur une plateforme d'une puissance limitée pour les standards actuels.

Remise en forme au niveau ATM

Les interfaces ATM, qui sont utilisées pour les liaisons WAN de la plateforme @IRS, n'effectuent pas un lissage de trafic par VC mais effectuent un lissage sur le débit global de l'interface. Les VC ont un profil CBR[4] de 1 ou 2 Mbit/s. Le contrôle d'accès est effectué par Renater qui détruit les cellules détectées hors profil. Il est donc essentiel de ne pas émettre de rafales de trafic ATM.

Les commutateurs ATM peuvent réaliser un lissage de trafic au débit accepté. Toutefois, ce lissage ne s'applique qu'à de brèves rafales de cellules car il s'effectue avec la mémoire du commutateur. Il faut donc procéder à un pré-lissage du trafic IP afin d'éviter de congestionner l'interface ATM. C'est dans ce but que le limiteur de débit crête de l'élément de cœur a été envisagé.

En pratique, les premiers essais montrent que le lissage ATM au sein des commutateurs est suffisant pour permettre le transport de très courtes rafales IP ; considérant que le trafic est remis en forme selon un espaceur cadencé à 1 kHz (comme indiqué ci-dessus).

4.5 CONFIGURATION DES ROUTEURS

4.5.1 Organisation du réseau et interfaces à configurer

Nous avons vu que le comportement de bordure concerne le trafic provenant d'un réseau périphérique et que le comportement de cœur est présent en sortie de toute interface reliant deux routeurs du domaine d'administration DiffServ.

Les éléments de bordure et de cœur se configurent séparément pour chaque interface concernée. De la même façon que l'implémentation ALTQ, l'élément de bordure traite le trafic entrant et l'élément de cœur le trafic sortant. Comme on peut le voir sur la figure 4.8, un

[4]Constant Bit Rate

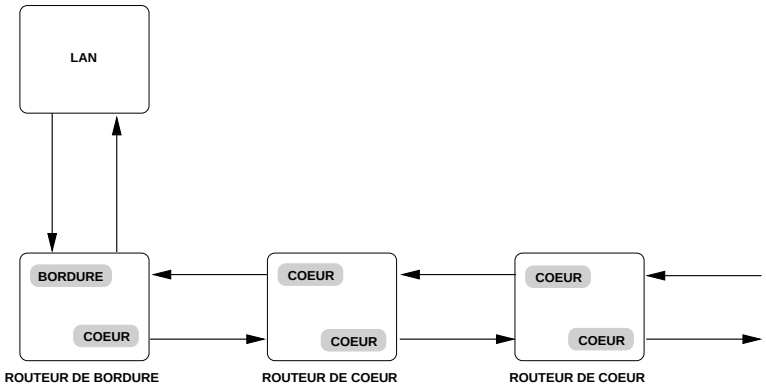


Fig. 4.8 Comportement des interfaces de routeurs

routeur de bordure est constitué d'un élément de bordure et d'un élément de cœur, tandis que le routeur de cœur est constitué de deux éléments de cœur.

4.5.2 Configuration du comportement de bordure

Le programme `edged` est utilisé pour configurer le comportement de bordure d'une machine FreeBSD. En fonction du paramétrage donné par un fichier de configuration, le routeur sera capable d'identifier les paquets appartenant à un flot spécifique, de les conditionner et de les marquer. Hormis les paquets appartenant au flot de type DE, tous les autres paquets seront conditionnés par un *token bucket*. Pour les flots de type AF (Assured Forwarding), le *token bucket* sert uniquement à déterminer si le paquet traité est conforme ou non conforme au profil. Pour les flots de type EF (Expedited Forwarding), le *token bucket* sert à effectuer la régulation du trafic entrant afin qu'ils soient dans le profil. Chacun des flots paramétrés peut se voir affecter un des trois traitements suivants :

- **Drop** : les paquets non conformes au profil sont supprimés, ceux qui sont conformes sont marqués comme paquets DE et délivrés immédiatement ;
- **Mark** : les paquets conformes au profil sont marqués AF prioritaires, ceux non conformes sont marqués AF non prioritaires. Tous sont délivrés immédiatement. Pour être conforme au profil, il faut qu'il y ait autant de jetons dans le *token bucket* que d'octets à envoyer pour le paquet traité ;
- **Shape** : les paquets sont stockés dans une file d'attente spécifique au flot. Dès que le nombre de jetons est suffisant dans le seau, on libère un paquet. Si la file d'attente est pleine, les paquets excédentaires sont supprimés.

Tout paquet n'entrant dans aucun des flots configurés est automatiquement marqué en tant que paquet DE et remis à l'élément de cœur.

Ligne de commande pour exécuter `edged`

Usage :

```
edged [-f fichier_config] [-s] [-o] [-h] nom_interface
```

Description des options :

`-h` affiche le présent mode d'emploi

`-s` affiche les statistiques courantes de l'interface

`-o` OFF, désactive le routage sur interface

`-f fichier_config` active les paramétres lus dans `fichier_config`

`nom_interface` nom de l'interface réseau utilisé

Description du fichier de configuration `edged`

En préalable aux différentes commandes acceptant des paramètres exprimant une vitesse ou une taille, le programme peut accepter différentes notations :

- en bits 8000b soit 8000 bits
- en octets 1000 soit 1000 octets
- en K bits 8Kb soit 8000 bits
- en K octets 1K soit 1024 octets
- en M bits 8Mb soit 8000000 bits
- en M octets 1M soit 1024K

Il existe une seule commande globale pour l'interface, il s'agit de l'indication de la vitesse maximum, exprimée par une ligne :

maxspeed <vitesse>

Tous les flots décrits font ensuite l'objet de 4 lignes distinctes. La première ligne consiste à attribuer un nom au flot, via un ordre :

newtc <identifiant>

La commande suivante va permettre d'effectuer la classification des paquets de ce flot en fonction des adresses source, destination et des numéros de flot mini et maxi autorisés. Les adresses IP peuvent être spécifiées soit numériquement, soit avec le nom du système. Un caractère $*$ pour une adresse signifiera toutes les adresses possibles.

<identifiant> **filter** <source><destination><flot mini><flot maxi>

La méthode appliquée aux paquets du flot (shape/mark/drop), ainsi que les paramètres du *token bucket* associé seront exprimés avec la commande :

<identifiant> **method** <shape|mark|drop><TB taux><TB profondeur>

Enfin, la classe attribuée aux paquets de ce flot sera exprimée avec un ordre :

<identifiant> **class** <N° classe>

Un exemple de fichier de configuration d'un routeur de bordure est disponible en annexe A.

4.5.3 Configuration du comportement de cœur

Ligne de commande pour exécuter `bboned`

Le comportement de cœur se configure par l'utilitaire `bboned` fourni avec la distribution QoS. Sa syntaxe est :

`bboned <interface> on <fichier de configuration>`

Active le comportement de routeur de cœur sur l'interface choisie (les noms d'interface sont les mêmes que pour, par exemple, `ifconfig`). Le fichier de configuration est décrit ci-dessous.

`bboned <interface> off`

Désactive le comportement de routeur de cœur sur l'interface choisie. Si des paquets sont encore en file d'attente, ils sont éliminés.

`bboned <interface> stat`

Fournit un certain nombre d'informations sur les statistiques instantanées et cumulées quant au comportement de routage de cœur d'une interface ; on y trouvera par exemple l'état des files d'attente, le trafic écoulé depuis l'activation, le trafic éliminé, ...

Fichier de configuration

Le fichier est composé de lignes spécifiant les valeurs à utiliser pour les différentes classes de trafic.

rate <débit en Ko/s>

Cette ligne définit le débit maximum en sortie ; le décompte du volume correspond aux PDU IP, et n'inclut donc pas le débit supplémentaire dû aux trames Ethernet, ou AAL et ATM.

pktsize <longueur typique de paquet en octets>

Cette ligne définit la longueur « standard » d'un paquet, pour l'estimation de la décroissance exponentielle de la longueur de file lors des périodes d'inactivité.

GS <nb max de paquets><longueur max en octets>

Cette ligne définit les caractéristiques de la classe GS : nombre maximum de paquets en file et taille cumulée maximum de ces paquets.

BE <nb max de paquets><longueur max en octets><poids WF^2Q>

La file est définie comme GS ; on trouve aussi la part de bande passante garantie au service BE.

AS <nb max de paquets><longueur max en octets><poids WF^2Q><dscp in><dscp out><threshold PBS><poids instantané>

On peut définir plusieurs classes AS, qui distinguent le trafic in/out si les champs DSCP in et out ont des valeurs différentes ; dans ce cas, on utilise le threshold PBS (en octets)

comme limite de taille de file au-delà de laquelle seuls les paquets in sont acceptés. Le poids instantané est l'importance de la longueur instantanée de la file dans les mises à jour de la longueur exponentielle. De premières expériences n'ayant pas montré d'avantage significatif à l'usage d'une moyenne exponentielle, une valeur de 1 est recommandée.

Un exemple de fichier de configuration d'un routeur de cœur est disponible en annexe A.

4.6 CONCLUSION DU CHAPITRE

Ce chapitre a présenté les spécificités techniques de la réalisation d'un routeur à qualité de service issu de la recherche industrielle en vue de la mise en œuvre d'une architecture DiffServ. Les contraintes techniques, les choix de files d'attente et d'ordonnancement ont été exposés. Cette plateforme a été utilisée pour des mesures que nous allons décrire dans le chapitre suivant. Nous verrons plus loin dans cette thèse que ces mesures ont donné lieu à de multiples études.

Chapitre 5

Mesure sur la plateforme @IRS

5.1 INTRODUCTION

Les premières mesures effectuées avec la plateforme ont eu pour but de valider les classes de services les unes par rapport aux autres. Celles-ci ont été effectuées en collaboration avec le LAAS[1]. Pour se faire, le pire des cas est retenu. Un flot UDP occupant 100% des ressources disponibles de la classe de service AS (resp. GS) est émis parallèlement à un flot BE dont la charge va en s'accroissant jusqu'à saturation complète de l'ensemble des ressources du réseau. Un flot dans cette étude représente un agrégat de flots, à savoir la charge totale dans la classe de service. Les résultats de ces mesures ont permis de conclure que les QoS AS et GS étaient conformes à celles attendues, à savoir une influence nulle du trafic BE sur la QoS GS (délai minimum, moyen, maximum, fiabilité et débit moyen inchangés), et relativement faible sur la QoS AS (délai minimum et moyen, fiabilité et débit moyen inchangés, le délai maximum étant variable) [GCL+01].

5.2 METHODE D'EVALUATION DES SERVICES

5.2.1 Conditions d'études

Pour un utilisateur, la perception d'un service de communication avec une qualité de service se fait au niveau de son flot de paquets. L'évaluation présentée consiste à étudier la qualité de service de bout en bout sur la base d'un flot applicatif par des mesures expérimentales. L'isolation des flots aux conditions de trafic est étudiée. Une certaine indépendance doit exister entre les microflots d'un même agrégat. De même, la qualité de service doit être insensible à la charge ou à la composition de la classe de service (ou agrégat). L'étude présentée par la suite vise à vérifier ce postulat pour les deux types de trafic de l'Internet

[1] Laboratoire d'Analyse et d'Architecture des Systèmes de Toulouse

et vise à démontrer qu'une qualité de service peut être fourni au niveau d'un microflot sans avoir recours à une gestion par microflot comme dans l'architecture IntServ. Dans l'architecture proposée, le traitement au niveau d'un microflot s'effectue uniquement à l'entrée du réseau dans le routeur de bordure. Les expérimentations portent sur l'étude de : (1) l'influence de la composition de l'agrégat exprimée en nombre de flots sur la QoS d'un flot individuel, (2) la QoS reçue par un flot élastique lorsqu'il utilise le service à débit assuré.

Ces évaluations sont faites dans le pire des cas. Le réseau est à chaque fois saturé par du trafic BE (le réseau est donc en état de congestion). La mesure de l'influence dans le premier cas d'étude se fera par le délai de transit de bout en bout et le taux de perte. Plus précisément, les valeurs minimales, moyennes et maximales du délai de transit sont déterminées pour des sessions expérimentales d'environ 300 secondes chacune. La fonction de répartition du délai de transit des paquets est également calculée. Le taux de perte correspond au rapport du nombre de paquets non reçus sur le nombre de paquets émis pour une session complète. La dernière évaluation utilisera des sources de trafic TCP et retiendra le débit utile comme paramètre de QoS.

5.2.2 La plateforme de tests

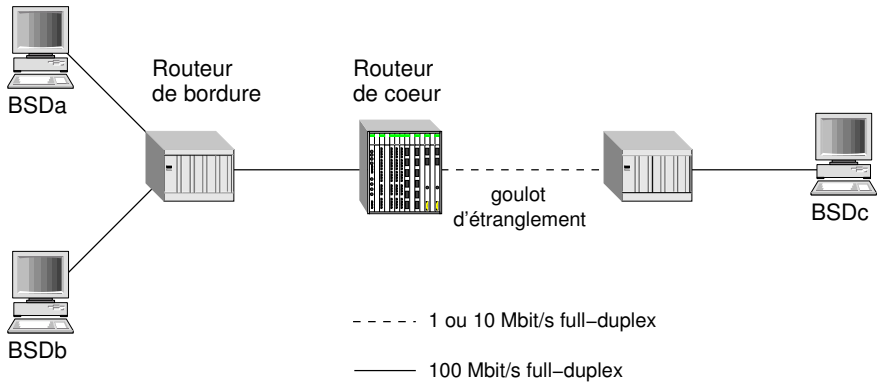


Fig. 5.1 Représentation fonctionnelle de la plateforme de tests

Comme le montre la figure 5.1, la plateforme de tests est composée de trois noeuds. Le routeur utilise un noyau Unix FreeBSB et comporte les mécanismes d'ordonnancement décrits dans la première partie. Les deux hôtes FreeBSD (BSD_A et BSD_B) sont aussi des machines Unix qui émettent un trafic marqué, correspondant aux services GS ou AS. Elles effectuent également les fonctions de conditionnement. Le rôle du puits de trafic qui a en charge certaines mesures est joué par l'hôte (BSD_C). L'hôte noté *BE* génère le trafic du service « au mieux ». Afin de saturer le réseau, l'hôte *BE* émet à un débit équivalent au débit de l'interface de sortie du routeur.

Les tests ont nécessité deux générateurs de trafic. Pour les flots non élastiques, les flots sont générés au moyen d'un logiciel de génération de trafic développé en interne, nommé

DEBIT6[2], permettant d'envoyer un trafic UDP dont le profil est déterminé par un *leaky bucket*. Pour les flots élastiques, le générateur de trafic retenu est BENCH [Roc98] que nous avons porté en IPv6 afin d'effectuer les mesures et modifié pour qu'il puisse marquer le *flow label* de chaque paquet. BENCH offre la possibilité de générer plusieurs connexions simultanées depuis un même émetteur. Cette capacité a été utilisée par l'hôte BSD_B.

Le choix de cette plateforme peut paraître simple au premier abord, mais elle caractérise en fait n'importe quel routeur dans un domaine. Que ce soit un routeur de bordure ou un routeur de cœur, au niveau microscopique, le comportement de relayage de chacun est identique. Au niveau macroscopique, la QoS résultante dépendra du routeur qui accusera les conditions de trafic les plus mauvaises en entrée par rapport à la capacité d'écoulement de son support en sortie. Pour le service AS, les routeurs en aval recevront donc un flot déjà lissé. Construire une plateforme de tests à un seul routeur repose sur l'hypothèse que le flot subira les perturbations les plus significatives au niveau de ce routeur. Une telle configuration peut être désignée par le terme de goulot d'étranglement. Formellement, un lien est défini comme goulot d'étranglement pour un flot si le flot possède le débit plus important par rapport aux autres flots de ce lien et que le lien est saturé. L'objectif de cette plateforme consiste à mesurer les écarts de QoS les plus significatifs. Par exemple, la perte de temps au niveau d'un paquet ou la perte d'un paquet ne se corrige pas sur la suite de la route. Les détériorations d'un flot de paquets ne peuvent qu'empirer ou au mieux rester identiques. Donc une plateforme de tests composée d'un seul routeur pour former un goulot d'étranglement est suffisante pour effectuer l'évaluation des services proposés.

Les hypothèses de configuration de la plateforme sont :
- le débit maximal C_{GS} de trafic GS admis est fixé à 20% de la capacité du goulot d'étranglement. Il reste donc au minimum 80 % de la capacité de la bande passante du goulot d'étranglement pour les autres services ;
- la classe AS est dimensionnée pour avoir une garantie de 50 % de la bande passante non utilisée par le service GS. La pondération du WFQ utilisée pour AS est équivalente à celle de BE à savoir 0.5. Le service AS a donc une garantie de 40 % de la bande passante du goulot d'étranglement. Notons C_{AS}, cette garantie de débit ;
- aucun biais n'est introduit entre les services par la taille des paquets. Tous les paquets ont une taille de 1024 octets ;
- les délais de propagation entre émetteurs et récepteurs sont équivalents ;
- les hôtes sont synchronisés par NTP (*Network Time Protocol*), limitant l'incertitude sur les mesures du délai à +/- 5 ms ;
- la valeur du seuil de PBS est fixée à 32ko. Cette valeur représente la moitié de la fenêtre d'anticipation utilisée par défaut par TCP.

En ce qui concerne le débit du goulot d'étranglement, il est intéressant d'utiliser un faible débit lorsque des délais sont mesurés. Le temps de paquétisation et de sérialisation des paquets dépendent du débit du goulot d'étranglement, les écarts de délai seront plus importants avec un faible débit. De plus, les imprécisions dans les mesures (à cause de NTP) deviennent négligeables. Et les temps de traitement du routeur dépendant du matériel sont minorés par rapport aux termes de délai liés à la communication. En ce qui concerne les

[2]DEBIT6 est disponible sur `http://rp.lip6.fr/airs`

mesures concernant le débit, afin d'augmenter la taille de l'agrégat, celui-ci a été multiplié par dix.

5.3 MESURES ET ANALYSES DES SERVICES

L'évaluation de la QoS au niveau du flot individuel se fera par la mise en oeuvre de trois scénarios d'études. Les deux premiers se placeront dans la perspective de l'utilisation des services par des flots issus des applications interactifs. Le dernier concernera les flots issus des applications élastiques.

5.3.1 Etude en isolation des services

Ce scénario a pour but d'évaluer l'influence du nombre de flots AS (resp. GS) sur la QoS AS (resp. GS) quand le réseau est saturé par du trafic BE. Aucun flot GS (resp. AS) n'est généré dans le réseau. Cette évaluation est faite à charge constante. L'étude du service s'effectue selon trois cas de figure comme indiqués par le tableau 5.1 :
- 1 flot à QoS est généré par l'hôte BSD_A avec un débit moyen équivalent à 50% de la capacité du service ;
- 2 flots à QoS sont générés par les hôtes BSD_A et BSD_B avec un débit moyen équivalent à 23 et 27% de la capacité du service ;
- 4 flot à QoS sont générés par les hôtes BSD_A et BSD_B avec un débit moyen équivalent à 11, 12, 13 et 14% de la capacité du service.

Parallèlement à ces flots, un flot BE (pour le premier cas) et deux flots BE (pour les deux autres) sont générés par la machine BE. La somme de leur débit est toujours équivalente à la capacité du goulot d'étranglement. Le débit du goulot d'étranglement est fixé à 1 Mbits/s.

Service AS	C_{AS}	Service GS	C_{GS}
AS (BSD_A)	50	GS (BSD_A)	50
AS_A (BSD_A)	23	GS_A (BSD_A)	23
AS_B (BSD_B)	27	GS_B (BSD_B)	27
AS_{A1} (BSD_A)	11	GS_{A1} (BSD_A)	11
AS_{A2} (BSD_A)	14	GS_{A2} (BSD_A)	14
AS_{B1} (BSD_B)	13	GS_{B1} (BSD_B)	13
AS_{B2} (BSD_B)	12	GS_{B2} (BSD_B)	12

Tab. 5.1 Différents cas de la génération de trafic pour chaque service

service GS

La figure 5.2 montre l'évolution du délai de transit en fonction du nombre de flots de l'agrégat. D'après le tableau 5.2, la progression est de 8 ms en valeur moyenne et qui est également la valeur de variation pour 100% des paquets (figure 5.2). Cette valeur reste inférieure à l'écart maximum dans le cas d'un flot GS unique (14 ms) qui permet d'indiquer

que le nombre de flots dans l'agrégat a une influence relativement faible. Avec un délai de sérialisation de 10 ms (la capacité du support au niveau paquet vaut 100Koctets/s et tous les paquets ont une taille de 1 Koctets), l'interprétation de l'écart maximal devient simple. L'écart de 20 ms (cas du flot GS_{B2}) signifie qu'il existe au pire une sporadicité de 2 paquets dans l'agrégat. L'étude de la relation entre la sporadicité de l'agrégat et la charge est développée dans [Fer00]. Celle-ci indique que la charge est le terme prédominant dans la QoS de GS. Une méthode de calcul de la borne maximale du délai est développée dans [BBC+01]. Le délai dépend du nombre de noeuds de la route et de la charge de trafic.

Délai(ms)	GS	GS_A	GS_B	GS_{A1}	GS_{A2}	GS_{B1}	GS_{B2}
min	19	16	16	18	18	18	18
moyen	25	26	26	32	31	33	31
max	33	33	35	33	34	37	38
% de perte	0	0	0	0	0	0	0

Tab. 5.2 Évaluation du service GS en fonction de sa composition

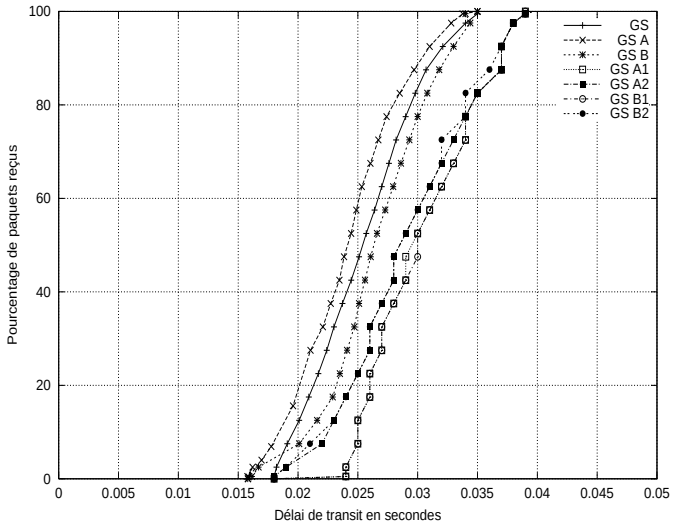


Fig. 5.2 Répartition du délai de transit du service GS

service AS

La courbe du flot de référence de la figure 5.3 représente le service AS dans le meilleur des cas, quand le réseau ne comporte aucun autre trafic. On peut cependant remarquer que 10% des paquets (pour les flots AS_A, AS_{B2}, AS_{A1}, AS_{A2}, AS_{B1}, AS_{B2}) ont un délai de transit nettement plus important que celui observé pour le flot AS seul. Ceci est intrinsèque au fonctionnement paquet et à l'asynchronisme. Lorsqu'il n'y a qu'une source, les paquets vont être générés en séquences. Dès qu'il y a plusieurs sources, des contentions sur l'interface de sortie du routeur peuvent se produire. Plus la charge sera importante et plus la probabilité

Délai(ms)	AS	AS_A	AS_B	AS_{A1}	AS_{A2}	AS_{B1}	AS_{B2}
min	25	18	18	20	18	18	20
moyen	38	38	37	42	42	40	41
max	49	59	63	86	75	65	77
% de perte	0	0	0	0	0	0	0

Tab. 5.3 Évaluation du service AS en fonction de sa composition

de contention sera forte. Ce phénomène est amplifié par WFQ qui fait une remise en forme quand le débit instantané de l'agrégat est supérieur à son débit alloué. Une remise en forme ajoute un délai d'attente pour les paquets concernés. Cette dispersion n'apparaît pas dans les mesures du service GS, justement car la charge est plus faible et que PQ n'effectue aucun contrôle de débit. Enfin, le tableau 5.3 montre que l'influence du nombre de flots sur un flot est faible. En effet, il indique une variation inférieure à 5 ms sur la valeur moyenne du délai de transit et un taux de perte constant à la valeur nulle. Ce résultat est conforté par la figure 5.3 qui met en évidence que le délai de transit est quasi inchangé pour 90% des paquets.

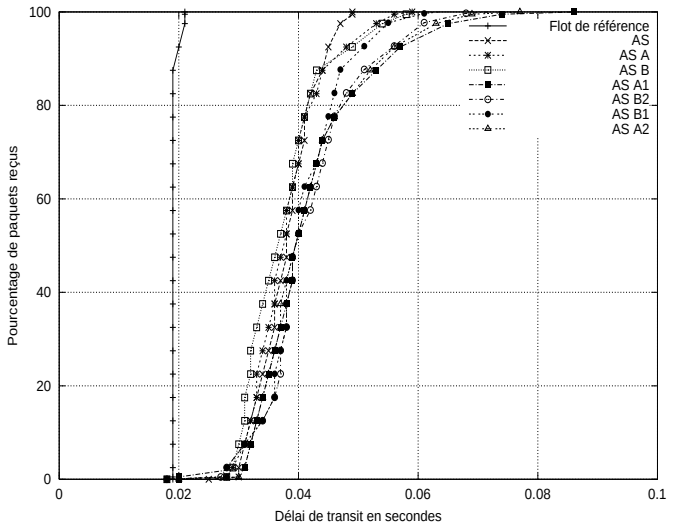


Fig. 5.3 Répartition du délai de transit du service AS

Au terme de cette étude, il faut remarquer que le service AS souffre d'un délai de transit supérieur à celui du service GS dans les mêmes conditions de trafic. Ceci valide les choix de mise en oeuvre des services dans l'interface de sortie.

5.3.2 Étude des services en présence mutuelle

Cette étude a pour but d'évaluer l'influence sur leur QoS du nombre de flots AS et GS générés conjointement quand le réseau est saturé par du trafic BE. Pour ces mesures (voir le tableau 5.4), les flots AS et GS sont émis depuis les hôtes BSD_A et BSD_B à destination de l'hôte BSD_C. Deux cas de figure sont considérés :

- 1 flot AS est généré par l'hôte BSD_A avec un débit correspondant à 100% de C_{AS} ; parallèlement, 1 flot GS est généré par l'hôte BSD_B avec un débit équivalent à C_{GS} ;
- 2 flots AS sont générés par les hôtes BSD_A et BSD_B avec un débit moyen correspondant chacun à 50% de C_{AS} ; parallèlement, 2 flots GS sont générés par les hôtes BSD_A et BSD_B ayant chacun un débit équivalent à 50% de C_{GS}.

Un flot BE est toujours généré à un débit équivalent à celui de la capacité du goulot d'étranglement soit de 1 Mbits/s.

	C_{GS} ou C_{AS}
AS (BSD_A)	100
GS (BSD_B)	100
AS_A (BSD_A)	50
AS_B (BSD_B)	50
GS_A (BSD_A)	50
GS_B (BSD_B)	50

Tab. 5.4 Les 2 cas étudiés de la génération de trafic

L'influence du nombre de flots GS (resp. AS) sur la QoS AS (resp. GS) est presque nulle. En effet, le tableau 5.5 indique une variation inférieure à 6 ms pour AS et 2 ms pour GS sur la valeur moyenne du délai de transit. Ce résultat est illustré par la figure 5.4. Là encore, le taux de perte est inchangé (nul). Notons que les délais de transit sont quasiment les mêmes que ceux observés dans l'étude des services en isolation. Enfin, la figure 5.4 montre bien des délais plus faibles pour le service GS par rapport à celui de AS. La différence provient du terme de paquétisation comme expliqué dans la partie 4.4.2. La différence entre les ordonnanceurs WFQ et PQ s'atténue quand la taille des paquets diminue [Fer00]. L'étude des services en présence mutuelle valide donc le choix des mécanismes mis en oeuvre dans l'interface de sortie.

Délai(ms)	AS	GS	AS_A	AS_B	GS_A	GS_B
min	19	18	17	17	17	18
moyen	42	26	47	48	27	28
max	61	42	78	88	41	40
% de perte	0	0	0	0	0	0

Tab. 5.5 Répartition du délai de transit

5.3.3 Étude du service AS pour les flots élastiques

Le service AS vise à donner une garantie de QoS à un flot en termes de débit et cela quelquesoit la composition de l'agrégat. L'objectif de cette étude consiste à évaluer l'adéquation du service AS pour les flots élastiques. Dorénavant, la source de trafic doit pouvoir changer son débit en fonction de l'état de charge de la route. Des paquets avec des priorités spatiales différentes vont constituer le flot applicatif. A la différence des deux précédents tests, on va s'intéresser au débit du flot. Les mesures sont effectuées en utilisant TCP. Les flots de

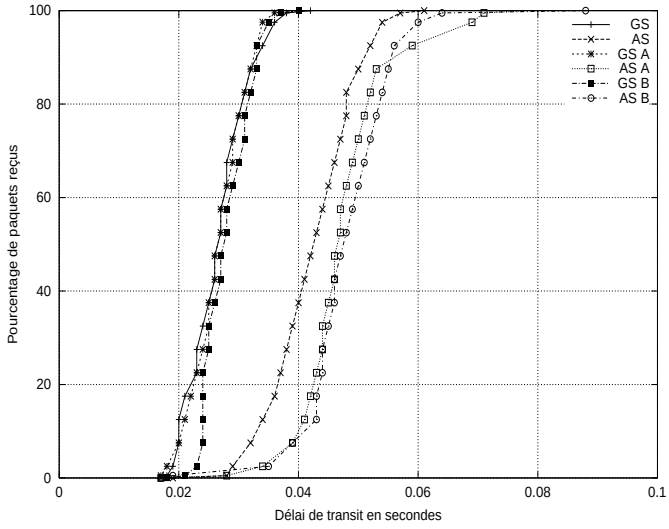


Fig. 5.4 Évaluation des services en présence mutuelle

données reposant sur le protocole de transport TCP présentent un caractère élastique de par le mécanisme de contrôle de congestion de TCP. Le débit est défini pour TCP comme la quantité de bits reçus par le récepteur (à l'exclusion des retransmissions) pendant la durée d'un transfert. Dans le cas présent, ceci correspond au débit utile (*goodput*).

La méthode retenue s'appuie sur un flot AS nommé « flot de référence » généré depuis l'hôte BSD_A. Son comportement est mesuré selon le nombre de flots supplémentaires générés par l'hôte BSD_B composant l'agrégat. Les tests se déroulent de la façon suivante :

- un flot BE en UDP est émis en permanence depuis l'hôte BE (voir figure 5.1) à un débit équivalent au goulot d'étranglement afin de maintenir l'interface de sortie du routeur dans un état de congestion ;
- chaque flot AS est émis en TCP et l'ensemble des flots de l'agrégat commence tous en même temps pour une durée de 120 secondes. Un flot représente le transfert d'un fichier (*bulk data transfer*) ;
- l'ordonnanceur WFQ attribue 50% de bande passante à chacun des deux flots : AS et BE ;
- chaque seconde, le générateur de trafic donne une évaluation du débit utile du flot de référence. En fin de test, celui-ci fait la moyenne des résultats obtenus et retourne en plus la valeur minimale et maximale. Cette mesure est faite par l'hôte source BSD_A.

Le débit du goulot d'étranglement est fixé à 10 Mbits/s. La taille de la file d'attente PBS (voir paragraphe 4.4.2) est fixée à $64Ko$ correspondant à la taille maximum par défaut de la fenêtre TCP et pour seuil, la moitié : $32Ko$ [ZFB01].

Lorsque des évaluations sont faites avec TCP, il faut prendre garde à ce que la voie de retour des acquittements ne soit pas saturée [PTCD01]. Les mesures faites ici ne concernent que l'augmentation du nombre de flots dans un agrégat composé uniquement de segments de données. La voie retour prise par les acquittements est isolée de la voie aller de par la configuration des expérimentations et de l'utilisation de liens *full-duplex* de la plateforme de tests.

Le dimensionnement du service AS (correpondant à la ressource allouée au service AS noté C_{AS}) par rapport au trafic AS (noté R_{AS}) joue un grand rôle dans l'assurance de débit offert à TCP. Soit n, le nombre de flots de l'agrégat dans le service et $r(i)_{AS}$, le paramètre d'un *token bucket marker* d'un flot i, correspondant au débit assuré du flot i (i.e. le débit des paquets en profil). La capacité totale allouée pour le service assuré R_{AS} correspond à :

$$R_{AS} = \sum_{i=1}^{n} r(i)_{AS} \tag{5.1}$$

De manière évidente, l'assurance de débit du service AS n'est valable que si le service est convenablement dimensionné. A savoir, tant que la condition suivante est vraie :

$$R_{AS} < C_{AS} \tag{5.2}$$

Lorsque le service comporte de la bande passante en excès (par rapport à la demande), une assurance de débit peut être donnée indépendamment des cinq paramètres qui peuvent affecter le débit de TCP (RTT(*Round Trip Time*), nombre de flots, débit voulu, taille des paquets, flots non-réactifs) [SNP99, GDJL99, dR99]. La distribution de la bande passante en excès aux flots TCP dépend cependant de ces cinq paramètres. La présente étude se situe dans un contexte différent ; le service est dimensionné convenablement et ne possède pas de bande passante en excès. Ceci représente le cas limite avant la saturation du service AS.

Dans le cas le plus favorable, un flot de référence en-profil est agrégé avec plusieurs flots hors-profil. Dans ces conditions, les paquets du flot de référence sont tous marqués IN et ne sont concurrencés que par des paquets marqués OUT. Le résultat est illustré par le tableau 5.6 et par la figure 5.5. La figure 5.5 représente l'évolution du débit utile normalisé par rapport à la ressource allouée au service AS (C_{AS}) du flot de référence en fonction du nombre de flots AS hors profil. Le tableau 5.6 indique qu'à partir de dix flots hors-profil, la file PBS est pleine et accuse quelques pertes de paquets opportunistes. Le flot de référence reste quant à lui constant au niveau du délai et de son débit utile quelquesoit le nombre de flots hors-profil arrivant au routeur. Ce test démontre qu'un contrôle de la bande passante peut être obtenu par une priorité spatiale [CF98]. Ce test définit la borne supérieure de la QoS AS.

Nb. de flots hors-profil	0	1	2	3	5	10	20	30	50	100
Débit utile Mbits/s	4.930	4.824	4.823	4.728	4.718	4.697	4.703	4.650	4.598	4.583
RTT en ms	1.662	1.698	1.699	1.733	1.736	1.744	1.742	1.762	1.782	1.787
Perte de paquets IN	0	0	0	0	0	0	0	0	0	0
Perte de paquets OUT	0	0	0	0	0	0.27%	2.36%	4.27%	7.58%	13.4%

Tab. 5.6 Débit utile du flot de référence en-profil en fonction du nombre de flots hors-profil

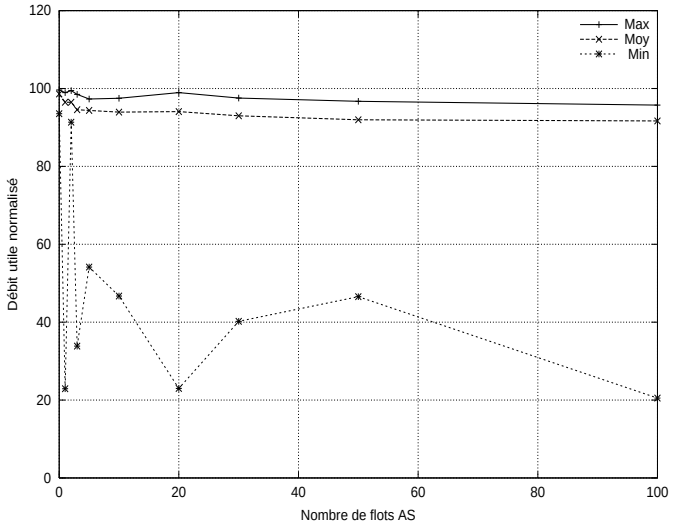


Fig. 5.5 Débit utile normalisé du flot de référence en fonction du nombre de flots hors-profil

Dans le cas courant, le flot de réference est conditionné selon un profil défini par les paramètres (r, B) d'un *token bucket marker* (voir section 3.2). Dans [dR99], il est noté que la taille du seau a une influence sur l'assurance de débit pour les flots qui ont une assurance de débit importante dans le cas d'un service surdimensionné. Le surdimensionnement est défini comme :

$$R_{AS} < 0.4\ C_{AS} \tag{5.3}$$

Cependant, l'augmentation de la taille du seau combinée avec la sporadicité inhérente de TCP accroît les rafales de paquets IN. Dans le cas du dimensionnement utilisé dans la plateforme de test, des congestions conjoncturelles peuvent se produire et conduire à des pertes de paquet IN. Pour éviter ce genre de situation, la taille du seau B est invariable et est équivalente à la taille d'un paquet. L'assurance de débit du flot de référence sera indiquée par le paramètre r. Quatre cas de figure sont retenus pour $C_{AS} = 5Mbits/s$; le débit assuré du flot de référence $r(ref)_{AS}$ varie de 1 à 4 Mbits/s avec un pas de 1 Mbits/s. D'après l'équation (5.2), le débit assuré d'un flot i de l'agrégat est donné par :

$$r(i)_{AS} = \frac{C_{AS} - r(ref)_{AS}}{n-1} \tag{5.4}$$

La figure 5.6 représente l'évolution du débit utile normalisé par rapport à C_{AS} du flot de référence en fonction du nombre de flots AS supplémentaires. Cette figure montre une forte variation du débit du flot de référence lorsque que le nombre de flots AS augmente. Le flot de référence n'est pas isolé des conditions de trafic. De plus, le taux de perte des paquets hors-profil du flot de référence augmente quand le nombre de flots AS augmente. Enfin, l'augmentation du taux $r(i)_{AS}$ du *token bucket marker* du flot de référence ne se traduit pas par une augmentation significative du débit utile pour ce dernier. Quand n tend vers l'infini, le débit du flot de référence converge quelquesoit son profil. Enfin, très logiquement, un flot avec un débit minimum faible atteint plus facilement son objectif qu'un flot avec une forte assurance de débit. L'explication provient de TCP qui réagit par un facteur multiplicatif à la perte et par un facteur additif aux transmissions réussies. Il en résulte un temps différent

pour retrouver la taille de la fenêtre de contrôle de congestion permettant d'émettre au débit minimum [YR99]. Ce test montre également que le *token bucket marker* est un très mauvais marqueur pour les flots TCP car il ne prend pas en compte la sporadicité de TCP [LZH99]. Des rafales de paquets hors-profil sont émis et peuvent entraîner des pertes en séquence. Il est connu que TCP a des problèmes de performance lorsque la connexion souffre de pertes en rafale [FF95].

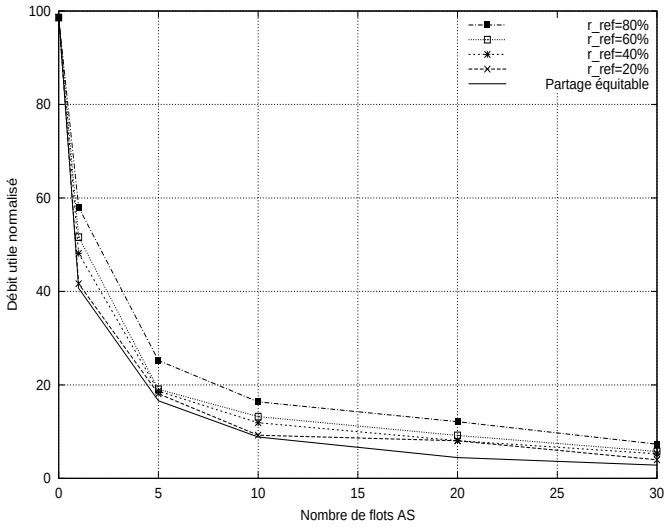


Fig. 5.6 Débit utile normalisé du flot de référence en fonction du nombre de flots AS

A noter que ces résultats sont indépendants de la gestion de la file d'attente du routeur. En effet, il a été observé par des simulations étendues sur les agrégats de flots TCP que la gestion de la file d'attente du routeur a une faible influence sur le débit utile et le taux de pertes [QZK01]. Cependant, une gestion probabiliste de la file d'attente tend à rendre le partage de la bande passante plus équitable. Il n'en reste pas moins une diminution linéaire du débit de transfert de TCP au fur et à mesure que le nombre de connexions augmente.

Les principales conclusions de cette expérience sont qu'il n'est pas possible d'effectuer une différenciation de service entre les flots TCP par un marquage selon un simple *token bucket marker*. Cette analyse rejoint celle faite par plusieurs études dont celle de [SND+00]. Enfin, la perte d'un paquet pour TCP, qu'il soit dans le profil ou hors profil, est préjudiciable au débit utile du flot. TCP réagit à la perte sans distinction de la priorité associée au paquet.

Les enseignements tirés au terme de l'évaluation du service AS pour les flots élastiques amènent à penser qu'il est possible d'effectuer une différenciation entre les flots par une priorité spatiale mais que cette différenciation dépend en grande partie du conditionnement.

5.4 CONCLUSIONS DU CHAPITRE

Les travaux présentés dans ce chapitre traitent du problème de la QoS dans l'Internet et portent sur la conception, l'implémentation et la mesure des performances d'une architec-

ture de communication à QoS garantie, supportant des services différenciés au niveau IP et une QoS par flot de bout en bout. Plusieurs conclusions peuvent être tirées ; elles étendent celles données dans [GCL+01], qui étaient les suivantes : (1) une architecture à services différenciés au niveau IP peut être facilement déployée dans un environnement de type VPN[3] tel que celui de @IRS ; (2) une différenciation de services existe entre les services GS et AS selon le paramètre du délai. Cette conclusion est appuyée par l'étude de faisabilité du déploiement des mécanismes de QoS dans un routeur [GLN+99]. Cette étude montre que la QoS peut être déployée avec un coût minimal sur les performances des routeurs.

Les mesures présentées permettent d'apporter les conclusions supplémentaires suivantes. Pour des flots (AS ou GS) respectant leur profil de trafic : (1) l'influence du nombre de flots GS sur la QoS GS est faible ; (2) les services sont isolés les uns des autres ; et (3) un flot élastique utilisant le service AS n'a pas d'assurance de débit. La cause principale en revient au marquage et que le marquage selon un *token bucket marker* est inadapté à la caractérisation d'un flot TCP.

Ces résulats offrent plusieurs perspectives d'études. La première consiste à évaluer (au moyen du simulateur *ns*) l'influence des paramètres de niveau IP (taille des files des routeurs, poids des WFQ, etc.) sur la QoS ; en effet, la valeur exacte des paramètres mesurés en dépend en grande partie. La seconde perspective est de formaliser les sémantiques de garantie associées aux paramètres de QoS, de façon à établir un lien entre la relative imprécision du service AS et la marge d'erreur qu'une application peut tolérer pour certains de ses flots. La troisième est le développement d'un mécanisme de marquage adapté à TCP afin de contrôler la différenciation de débit entre les flots quelquesoit la composition de l'agrégat. Enfin, la quatrième perspective est de développer un mécanisme (activé par l'API) permettant au programmeur d'une application de faire abstraction du choix des services de niveaux IP et Transport lors de l'accès au système de communication. Plus prospective, la dernière perspective est d'étendre ces travaux à un contexte multidomaine.

Dans le cadre de cette thèse, nous nous sommes plus particulièrement intéressés au développement d'un mécanisme de marquage adapté à TCP afin de contrôler la différenciation de débit entre les flots quelquesoit la composition de l'agrégat. Nous verrons dans le chapitre suivant que les propositions dans ce domaine sont nombreuses mais que peu d'entre elles peuvent passer du cadre de la simulation au cadre d'un réseau réel.

[3] *Virtual Private Network*

Chapitre 6

Garantie de débit TCP pour la classe AF : problématique et état de l'art.

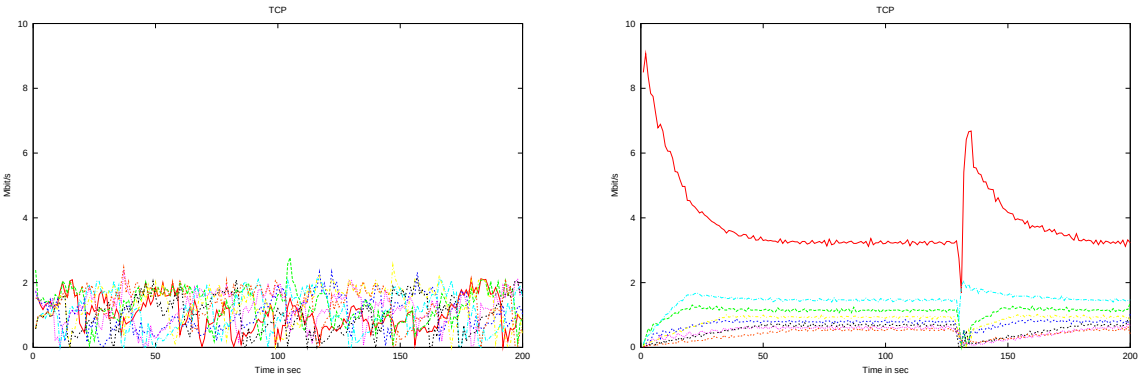
6.1 INTRODUCTION

Les propositions de marquage pour TCP se divisent en deux familles : celles qui traitent du marquage d'un flot TCP avec un profil sur un agrégat, et celles qui traitent du marquage du flot TCP par rapport à un profil individuel. La première vise la propriété d'équité en plus de l'assurance de débit recherchée par la seconde. Ce chapitre présente un état de l'art des différentes solutions mises en œuvre afin d'obtenir une garantie de débit dans la classe AF d'un flot TCP. Nous verrons que ces solutions se basent sur une dérivation de l'algorithme à fenêtre glissante (*time sliding window*) et du marqueur à deux ou trois couleurs (*token bucket marker* noté TBM) et que la majorité de ces approches se base sur un marquage probabiliste des paquets hors profil.

6.2 PRÉSENTATION DU PROBLÈME

La réalisation du débit d'un flot est fonction de la politique de rejet des paquets du réseau et de la façon dont réagit le protocole de transport à ces pertes. TCP réagit à une perte de paquet en diminuant de moitié sa fenêtre de congestion et augmente celle-ci linéairement chaque fois qu'un paquet est délivré suivant le principe AIMD : *additive increase* et *mutiplicative decrease* [Jac88, FF99].

Une étude approfondie du comportement des flots TCP et UDP dans la classe AF a été menée dans [SNP99]. Cette dernière a montré que lorsque le service comporte de la bande passante en excès (par rapport à la demande), une assurance de débit peut être donnée



(a) Flots TCP à RTT identiques

(b) Flots TCP à RTT differents

Fig. 6.1 Comportement de débit TCP observé par des flux à RTT identiques (a) et différents (b)

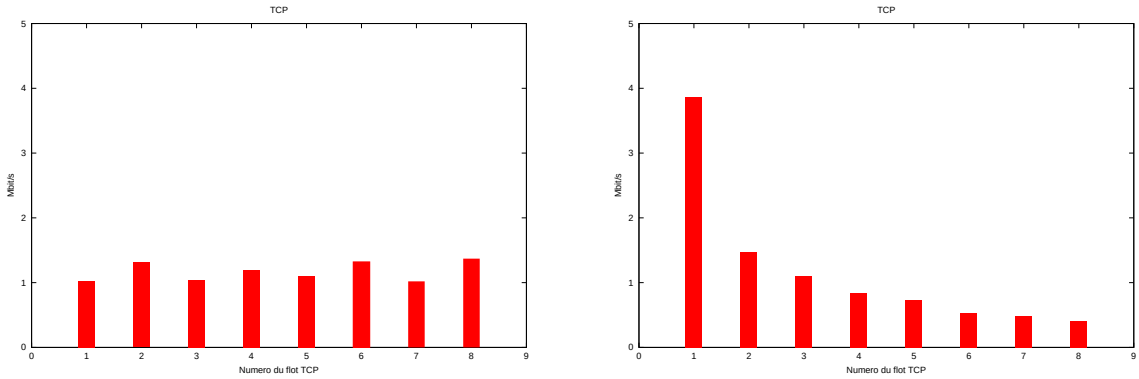
indépendamment des cinq paramètres suivants : RTT[1], nombre de flots, débit voulu, taille des paquets, flots non réactifs. La distribution de la bande passante en excès aux flots TCP dépend de ces cinq paramètres. Des conclusions similaires ont été présentées dans [GDJL99, dR99]. Enfin, [SNP99] définit trois critères concernant l'équité entre TCP et UDP en fonction de l'état du réseau. Dans un réseau surdimensionné, tous les flots TCP et UDP peuvent obtenir : (1) leur débit assuré (ou débit cible (*target rate*) noté TR) ; (2) un partage équitable de la bande passante en excès proportionnel à leur débit assuré. Dans un réseau sous-dimensionné, tous les flots TCP et UDP observent une dégradation de leur débit proportionnelle à leur débit assuré.

Lors des tests que nous avons effectués sur plateforme réelle, nous avons remarqué que sur un flot TCP long, une perte d'un paquet hors profil était fortement préjudiciable pour son débit de transfert. En partant des hypothèses que : (1) le réseau est convenablement dimensionné ; c'est-à-dire que la somme des paquets en profil transmis au réseau ne dépasse pas sa capacité et (2) qu'un paquet en profil a une probabilité proche de 1 d'être délivré alors nous pouvons considérer que la perte accusée par le flot est bien celle d'un paquet marqué hors profil.
Après une perte de paquet hors profil, la source TCP entame une phase de *fast recovery* (sa fenêtre de congestion a été divisée par deux) qui a pour conséquence de diminuer son débit d'émission. Au niveau du goulot d'étranglement, les autres flots TCP présents vont occuper la place libérée. Il devient alors très difficile pour le flot qui a subi une perte de reprendre son débit nominal au sein du routeur congestionné.

Autre problème connu, la différence de RTT entre les flots influe sur le débit assuré désiré. Dans le cas de RTT identiques, les figures 6.1(a) 6.2(a) nous montrent que chaque flot se partage équitablement la bande passante disponible. En revanche, à RTT différent comme le montre les figures 6.1(b) 6.2(b), le partage équitable TCP n'existe plus.

Dans [SND+00] il est montré que :
– le débit obtenu n'est pas proportionnel au débit de marquage ;

[1] *Round Trip Time*



(a) Flots TCP à RTT identiques (b) Flots TCP à RTT differents

FIG. 6.2 Total des débits TCP observé par des flux à RTT identiques (a) et différents (b)

- il n'est pas toujours possible d'atteindre le débit assuré ;
- un flot avec un débit assuré élevé n'atteindra jamais son but si un flot avec un faible débit assuré dépasse son profil ;
- il existe une échelle de valeurs du débit assuré pour laquelle les paramètres du *token bucket marker* n'ont aucune influence.

Concernant ce dernier point, dans le cas d'un réseau avec bande passante en excès (réseau surdimensionné), lorsque la probabilité de perte d'un paquet en profil est nulle : $p(i)_{IN} = 0$ et que la perte d'un paquet hors profil ne l'est pas : $p(i)_{OUT} > 0$ si le débit assuré du flot vérifie l'inégalité suivante :

$$r(i)_{AS} < \frac{1}{RTT}\sqrt{\frac{3}{2\ p(i)_{OUT}}} \tag{6.1}$$

alors [SND+00] montre que le *token bucket marker* n'a aucune influence sur le débit atteint.

6.3 COMMENT SOLUTIONNER CES PROBLÈMES

Un réseau peut se trouver dans deux états distincts. En état de sous-dimensionnement (*under provisionning / over subscribed*) ou en état de surdimensionnement (*over provisionning / under subscribed*). Dans l'état de surdimensionnement, le réseau possède de la bande passante en excès qui va influer sur les débits des flots TCP.

Soit ϕ_{AS} le poids normalisé provisionnant le service AS. n le nombre de flots TCP de classe AS au niveau du goulot d'étranglement et C la capacité du lien. Plus exactement, cette capacité est représentée par le goulot d'étranglement du réseau. En partant de l'équation (5.2) donnée au chapitre 5, on a :

$$R_{AS} < C_{AS} \quad \text{avec} \quad C_{AS} = \phi_{AS} * C \tag{6.2}$$

La bande passante en excès e vaut :

$$e = C - R_{AS} - min(R_{BE}, \phi_{BE} * C) \tag{6.3}$$

Avec R_{BE}, la bande passante utilisée par les flots BE et ϕ_{BE} le poids normalisé provisionnant le service BE. Nous sommes dans le cas d'un réseau **surdimensionné**. Donc, un flot ayant une valeur de marquage de ces paquets en profil à $r(i)_{AS}$ devrait obtenir un débit théorique b_i de [YR01a] :

$$b_i = r(i)_{AS} + \frac{e}{n} \tag{6.4}$$

avec n le nombre de flots TCP traversant le lien. Cette équation considère que la bande passante en excès est équitablement distribuée entre les flots. Cela est vrai si et seulement si les RTT de chaque flot sont identiques. En effet, comme nous le montrent les figures 6.1 et 6.2, si un routeur est traversé par des flots TCP ayant des RTT différents, la bande passante n'est plus équitablement distribuée. De plus, cette évaluation n'est bien évidemment valide que dans le cas où le lien ne brasse que des flots TCP et devient inutilisable dans le cas d'un brassage avec des flots UDP.

L'état de saturation correspond à :

$$R_{AS} \geqslant C_{AS} \tag{6.5}$$

Dans ce cas, il n'y a pas de bande passante en excès. Nous sommes alors dans le cas d'un réseau **sous-dimensionné**. Lors des tests, il a été remarqué que plus il y a de flots TCP dans l'agrégat, plus le contrat de trafic est difficile à maintenir. Et plus le débit assuré est proche du partage équitable de TCP, plus il est facile à un flot de le maintenir.

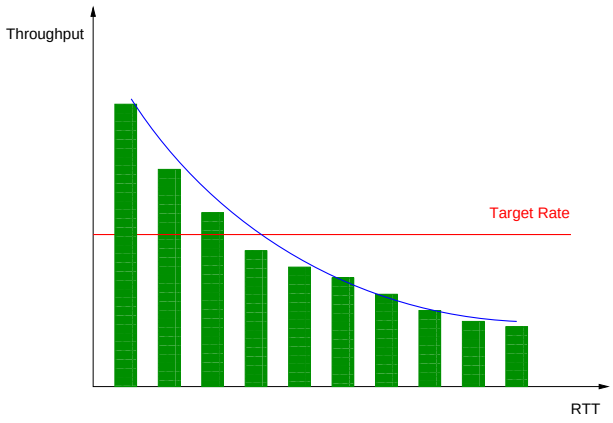
Dans un réseau correctement dimensionné, l'inégalité de l'équation (6.2) doit être respectée. Lorsqu'un réseau accuse des pertes, ces dernières doivent correspondre à des pertes de paquets hors profil et non à des pertes de paquets en profil. Cela signifie qu'il y a une légère congestion dans le réseau et que quelques paquets hors profil doivent être détruits. [FF99] a développé un modèle simplifié de calcul du débit d'un flot TCP illustré par l'équation (6.6) :

$$\text{Débit TCP} = \frac{Cte * MSS}{RTT * \sqrt{p}} \tag{6.6}$$

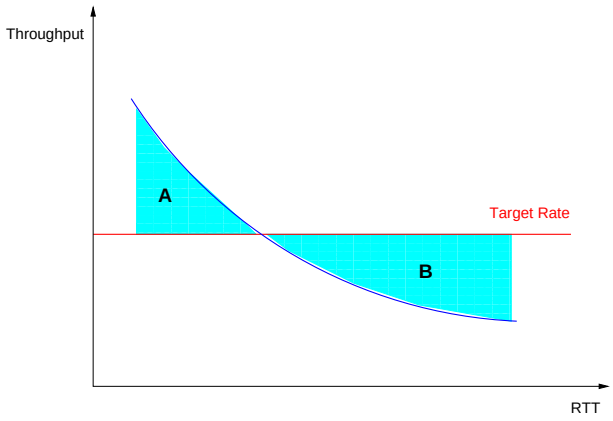
Avec MSS : taille maximum d'un segment TCP, p : la probabilité de perte et RTT : temps d'aller-retour (*round trip time*) d'un flot.

Les conditionneurs que nous allons présenter plus loin proposent d'augmenter la partie hors profil des flots qui ont un comportement opportuniste[2] dans le réseau. En partant de l'équation (6.6), il est facile de constater qu'en augmentant la probabilité de perte p d'un flot, son débit va diminuer. Donc, en augmentant la partie hors profil des flots les plus opportunistes, leur probabilité de perte augmente et leur débit TCP diminue. Les flots opportunistes obtiennent une probabilité de rejet supérieure au trafic non opportuniste, ce qui rend l'occupation du réseau plus équitable. Le problème est que changer la valeur p de l'équation (6.6) est une stratégie de marquage complexe. En effet, il est nécessaire d'évaluer

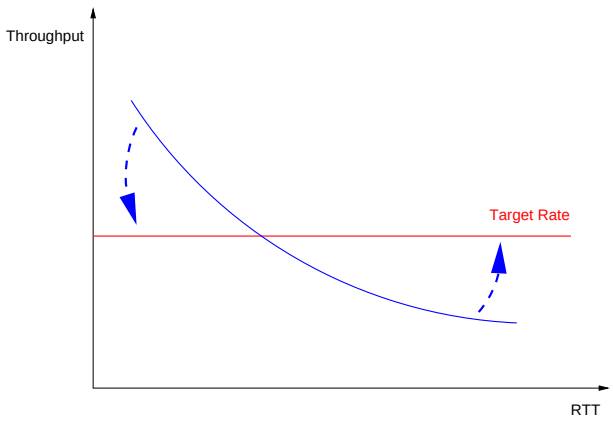
[2]On qualifiera d'opportuniste un flot TCP qui occupe plus de bande passante que son débit assuré dans un réseau congestionné.



(a) Flots TCP à RTT differents



(b) Aires d'excès et de déficit



(c) Solution au problème

Fig. 6.3 Débit de flots TCP en fonction de leur RTT.

la probabilité de perte et d'estimer le RTT propre à chaque flot. [EGS02] propose de calculer un intervalle moyen des pertes grâce à la méthode présentée dans [FHPW00] et d'estimer le RTT par une méthode d'estampillonnage des paquets. [HBF02] propose un conditionneur RTT-RTO basé exclusivement sur une mesure du débit. En plus de l'estimation du RTT, il propose une solution prenant en compte le *TCP timeout* (RTO) dans la stratégie de marquage.

Afin de mieux comprendre le fonctionnement de ces stratégies de marquage, la figure 6.3 illustre le but recherché. La figure 6.3 (a) symbolise le débit obtenu par dix flots TCP avec differents RTT. Plus le RTT est petit, plus le débit obtenu est grand. Récemment, [PC04] a démontré analytiquement et expérimentalement que le trafic hors profil des flots TCP n'était pas distribué proportionnellement à leur débit assuré mais équitablement entre tous les flots TCP. Les flots avec un petit RTT occupent donc plus de bande passante que nécessaire en regard de leur débit assuré. Ce fait est illustré par l'aire A de la figure 6.3 (b). Ces stratégies de marquage tentent d'opérer à une distribution équitable de la bande passante en excès représentée par l'aire A vers la bande passante en déficit représentée par l'aire B. Le but étant de trouver une courbe de solution qui se rapproche le plus possible du débit assuré comme expliqué sur la figure 6.3 (c).

6.4 SYNTHÈSE DES MÉTHODES UTILISÉES POUR GARANTIR LE DÉBIT TCP

Dans l'architecture d'un réseau DiffServ, il existe trois niveaux d'actions pour résoudre le problème de l'assurance du débit TCP dans la classe AF : au niveau des hôtes, à l'entrée du domaine et à l'intérieur du domaine.

- au niveau de TCP : les solutions proposées posent certains problèmes dans la mise en œuvre. Tout d'abord, il demande que le code de TCP soit modifié. Ceci pose un réel problème pour le déploiement d'une telle solution aux vues (1) de la diversité des systèmes d'exploitation et de leurs versions, (2) du nombre d'hôtes de l'Internet. D'un point de vue de l'architecture DiffServ, le marquage est effectué par la source exclusivement. Dans ce cas, le marquage n'est plus sous la responsabilité du prestataire de réseau. La vérification du marquage du client par le prestataire n'est pas non plus sans poser des difficultés de réalisation. Enfin, dans les situations où le marquage est effectué au niveau d'un agrégat, cette solution n'est pas envisageable. [FKSS97] présente une évolution de TCP pour intégrer un marquage en fonction d'un profil en tenant compte de l'état du réseau. Il est proposé un contrôle de congestion à deux fenêtres : une pour chaque partie du service AS ;
- au niveau du conditionnement, l'objectif est de calquer un marquage qui soit conforme à la dynamique des flots TCP. Le marquage est une fonction qui peut rester sous la responsabilité du prestataire de réseau. Le conditionnement est un élément qui se met en coupure sur la route. Il peut évoluer indépendamment des autres éléments de l'architecture DiffServ ;
- au niveau de l'AQM[3], de nouvelles techniques d'ordonnancement telles que JoBS [CLA02] permettent d'imposer des garanties pour les flots de la classe assurée. Ces techniques sont

[3] *Active Queue Management*

dérivées des mécanismes de proportionnalité introduits par [DR00]. Une autre solution serait dans l'inter-agissement que peut avoir l'AQM avec la source TCP. Le drapeau *Efficient Congestion Notification* de TCP pourrait être utilisé afin de contrôler le débit de la source en vue de limiter les paquets marqués hors profil entrant dans le réseau.

Les trois niveaux d'action s'évaluent par rapport à la qualité de la garantie de débit fourni, le déploiement, le passage à l'échelle. Le meilleur compromis sur ces critères se retrouve au niveau du conditionnement. La littérature très prolifique sur le conditionnement en est une illustration. Le paragraphe suivant présente la typologie des conditionneurs du service assuré.

6.5 CONDITIONNEMENT DES FLOTS TCP : ÉTUDE DE L'EXISTANT

Les propositions de marquage pour TCP se divisent en deux familles :
- celles qui traitent du marquage d'un flot TCP avec un profil sur un agrégat. Le but recherché étant l'équité entre les agrégats ;
- et celles qui traitent du marquage du flot TCP par rapport à un profil individuel. Dans ce cas, en plus de l'équité, on recherche une assurance de débit.

6.5.1 Rapport de proportionnalité débit/perte

Cette idée a été introduite par [DR00]. Elle inspira beaucoup de propositions dans le domaine de la garantie de débit TCP pour le service assuré. Chaque paquet arrivant dans le réseau est marqué soit en profil soit hors profil en fonction d'un *token bucket marker*. C'est au niveau de l'ordonnanceur, au sein du routeur, que le traitement s'effectue. Soit deux flots avec deux débits assurés différents r_1 et r_2 ayant un RTT et une taille de paquet identique. En partant de l'équation de modélisation d'un flot TCP donné dans [FF99] :

$$r_i \leq \frac{1.5\sqrt{\frac{1}{3}} * k_i}{RTT * \sqrt{p_i}} \tag{6.7}$$

Avec :
- r_i : le débit assuré du flot i ;
- k_i : taille des paquets du flot i ;
- p_i : probabilité de perte de paquets du flot i.

suivant l'équation (6.7) on a :

$$\frac{r_1}{r_2} = \sqrt{\frac{p_2}{p_1}} \tag{6.8}$$

Si l'on compare le nombre de paquets détruits par unité de temps (correspondant au débit des pertes) noté $d1$ et $d2$, on obtient :

$$\frac{d_1}{d_2} = \frac{r_1 * p_1}{r_2 * p_2} = \sqrt{\frac{p_1}{p_2}} = \frac{r_2}{r_1} \tag{6.9}$$

Ce qui nous indique que le nombre de paquets détruits par unité de temps doit être inversement proportionnel au débit recherché d'un flot. Le principal inconvénient de cette technique est qu'il est nécessaire d'être prêt du niveau de service pour qu'elle soit efficace.

6.5.2 Marquage qualitatif des microflots [MSZ03]

La perte d'un paquet TCP entraîne une reprise TCP. Lorsqu'un flot TCP est conditionné par un *token bucket marker*, il peut durant cette reprise avoir plusieurs paquets marqués hors profil. Ces paquets hors profil, prioritaires à la perte, peuvent se retrouver inacceptables dans la file d'attente RIO d'un routeur. Or, une perte de ce type de paquets, durant cette phase de transmission, a une répercussion importante sur le débit de transfert du flot. Pour cela, [MSZ03] propose de prendre en compte le marquage :
- des premiers paquets TCP émis afin d'initier correctement la fenêtre d'émission TCP ;
- des paquets après l'arrivée d'un *timeout* TCP, pour être sûr que le paquet retransmis le sera avec une forte probabilité ;
- des paquets émis après réception de trois acquittements dupliqués, pour être également sûr que le paquet retransmis le sera avec une forte probabilité et permettre à TCP de sortir de la phase dite *fast recovery*.

Cette technique est dite qualitative car elle cherche à améliorer le débit d'un flot TCP au contraire des techniques quantitatives qui cherchent à contrôler le débit du flot. Le principal problème de cette approche est qu'il faut connaître la taille de la fenêtre TCP et le seuil du *slow-start* noté dans les implémentations courantes : *ssthresh*. Ainsi, il est nécessaire de modifier la pile TCP pour arriver à mettre en œuvre cet algorithme.

6.5.3 Marquage quantitatif basé sur l'algorithme du *Time Sliding Window*

Plusieurs algorithmes de ce type ont été proposés pour travailler avec le service AS. Le *Two Rate Three Color Marker (TRTCM)* dans [HG99b] basé sur un *token bucket* métreur et le *Time Sliding Window Three Color Marker (TSW3CM)* dans [FSA00] basé sur un estimateur de débit moyen : le *Time Sliding Window* (TSW) [CF98]. Dans ces marqueurs, deux débits sont définis : un débit assuré appelé *Commited Information Rate (CIR)* et un débit maximal, le *Peak Information Rate (PIR)*, utilisé lors d'un surplus de bande passante. La principale différence existant entre ces deux marqueurs est la façon dont le marquage est effectué. Même si ils prennent chacun en argument le débit assuré : $r(i)_{AS}$, le TSW3CM, contrairement au TRTCM, applique un marquage probabiliste des paquets. En effet, le TRTCM marque un paquet hors profil si celui-ci est non conforme au profil défini par un *token bucket* : (r, B) (voir section 3.2). Par contre, le TSW3CM marque un paquet hors profil avec une probabilité fonction du débit moyen estimé par le TSW et les variables PIR et CIR. [Med01] a procédé à une étude comparative de ces deux techniques. Il considère qu'une bonne différenciation est observée si elle se rapproche du partage de ressources offert par les algorithmes de *Fair Queueing* (voir section 3.3.1). Dans ce cas, la différenciation est 100% équitable. A partir du débit assuré $r(i)_{AS}$ pour chaque source i et de la capacité du lien C, une différenciation de ce type est observée si la part de ressources

allouées à chaque source $R(i)$ satisfait l'équation suivante :

$$R(i) = C\frac{r(i)_{AS}}{\sum_i r(i)_{AS}} \tag{6.10}$$

Il montre que les deux marqueurs sont tous les deux :
- équitables entre flux TCP avec le même RTT : proches du 100% équitable selon l'équation (6.10) ;
- non équitables entre flux TCP avec des RTT différents ;
- non équitables pour le partage de ressources à l'intérieur des agrégats.

La principale différence entre ces deux marqueurs est qu'au contraire du TRTCM, pour le partage différencié entre agrégats TCP, le TSW3CM est proche du 100% équitable.

6.5.4 Marquage adaptatif

Les marqueurs utilisés pour garantir un débit assuré sont :
- soit basés sur une modification dynamique du débit assuré utilisé par le marqueur : [YR01a, CHM+02] ;
- soit basés sur la probabilité de marquage : [EGS02].

Les sections suivantes se proposent de présenter ces deux approches.

Marquage adaptatif à débit assuré dynamique

Dans [YR01a], il est décrit un marquage équitable d'un flot TCP dans un agrégat en utilisant un modèle de TCP qui repose sur le RTT et la taille du paquet. En partant du modèle mathématique du comportement d'un flot TCP défini dans [YR01b] :

$$b_i = \frac{3}{4}m_i + \frac{3k_i}{4RTT}\sqrt{\frac{2}{p_i}} \tag{6.11}$$

on pose :

$$b_i = \frac{3}{4}m_i + \epsilon_i \tag{6.12}$$

Avec b_i : le débit courant du flot i ; k_i : la taille de ses paquets ; p_i : sa probabilité de perte et m_i : la valeur de marquage du flot i correspondante à la valeur $r(i)_{AS}$ donnée au *token bucket marker*. Cette valeur est dynamique. Cette équation renvoie le débit d'un flot en fonction du paramètre de marquage du *token bucket marker*. Partant de l'équation (6.12), [YR01a] identifie les cas suivants ($r(i)_{AS}$ correspondant toujours au débit assuré souhaité par le flot i) :

1. si $b_i \leqslant \frac{3}{4}m_i + \epsilon_i < r(i)_{AS}$: nous sommes dans un réseau chargé et donc on accuse des pertes en profil. Il est alors très difficile d'atteindre $r(i)_{AS}$;
2. si $\frac{3}{4}m_i + \epsilon_i < b_i < r(i)_{AS}$: le flot n'atteint pas son but. La solution consiste alors à augmenter le taux de marquage m_i ;
3. si $r(i)_{AS} \leqslant b_i$: le flot atteint son but. Pour éviter de prendre des ressources pour rien, on réduit le taux de marquage m_i.

[CHM+02] propose également un *token bucket marker* à configuration dynamique pour le marquage d'un flot TCP selon un profil par flot. C'est un système similaire qui utilise aussi un *token bucket marker* à paramètre $r(i)_{AS}$ dynamique. Néanmoins, la méthode de calcul utilisée est fondamentalement différente et beaucoup plus complexe. En effet, elle se base sur des équations, issues de la physique des systèmes instables, utilisées notamment en mécanique. [CHM+02] montre que TCP peut être assimilé à un système instable et se propose d'utiliser les équations de ce domaine pour déterminer la valeur optimale de marquage.

Marquage adaptatif basé sur la mémorisation

Ce conditionnement basé sur la mémorisation a été proposé par [KAJ01]. Le principe du marquage est celui utilisé par le TSW3CM, sauf que la probabilité de marquage est pondérée par l'utilisation d'une variable mémoire. Cette variable garde en mémoire l'historique du débit moyen évalué par le TSW du marqueur TSW3CM. Elle est alors utilisée pour indirectement détecter une variation de la taille de la fenêtre d'émission TCP ou du RTT du flot conditionné. Cette méthode améliore le partage équitable de la bande passante en excès entre les flots quelquesoit leurs RTT ou leurs débits assurés respectifs.

Marquage adaptatif basé sur la probabilité de marquage

Dans la section précédente, nous avons vu que [YR01a] utilise une équation de comportement d'un flot TCP dans un réseau DiffServ afin de lui permettre d'obtenir une évaluation du débit qu'un flot pourrait obtenir. En fonction de cette évaluation, il propose de jouer sur le taux de marquage du *token bucket marker*. L'avantage de cette approche est qu'il peut opérer à un conditionnement au niveau du microflot ou de l'agrégat. Néanmoins, il ne peut avoir une information exhaustive de l'état du réseau. Sa solution possède en conséquence des limites assez fortes. Avec une autre approche, [EGS02] a montré qu'il existe une dualité entre l'évaluation du marquage basé sur le débit et l'évaluation du marquage basé sur les probabilités, comme le TSW3CM (Cf. [EGS02] figure 2 de l'article). Il se propose donc de déterminer le meilleur taux de marquage en fonction du résultat renvoyé par une équation de modélisation de TCP. Il possède un meilleur retour de l'état du réseau que [YR01a] dans le sens où il fait une évaluation de la probabilité de perte d'un paquet grâce à la méthode de [FHPW00].

L'équation de modélisation utilisée est celle définie dans [PFTK98] (voir en annexe B le détail du modèle), cette équation prend en argument :
- p : la probabilité de perte d'un paquet ;
- Wmax : taille maximale de la fenêtre TCP ;
- RTT : le *round trip time* ;
- RTO : la valeur de *timeout* TCP ;
- MSS : la taille maximum d'un segment TCP ;

et renvoie le débit D sachant que :

$$D = F(p_{OUT}, Wmax, RTT, RTO, MSS) \tag{6.13}$$

En supposant que nous sommes dans le cas d'un réseau bien dimensionné et donc que :

$$R_{AS} < C_{AS} \tag{6.14}$$

La probabilité de perte d'un paquet correspond à la probabilité de perte d'un paquet hors profil noté p_{OUT} puisque la probabilité de perte d'un paquet en profil doit être proche de zéro : $p_{IN} \simeq 0$. La difficulté réside donc dans le fait d'inverser cette équation pour qu'elle retourne p . Grâce à une méthode d'analyse numérique complexe, il obtient :

$$p_{OUT} = F(D, Wmax, RTT, RTO, MSS) \tag{6.15}$$

en substituant avec la valeur du CIR l'équation devient :

$$p_{OUT} = F(CIR, Wmax, RTT, RTO, MSS) \tag{6.16}$$

CIR correspondant au débit désiré par un flot. Le marquage étant effectué par un TSW3CM, il ne lui reste plus qu'à faire varier la probabilité de marquage des paquets P_{OUT} pour que celle-ci permette d'atteindre le CIR désiré. Le résultat obtenu se trouve sur la figure 6.4 dont les valeurs représentées ont été déterminées par simulation analytique.

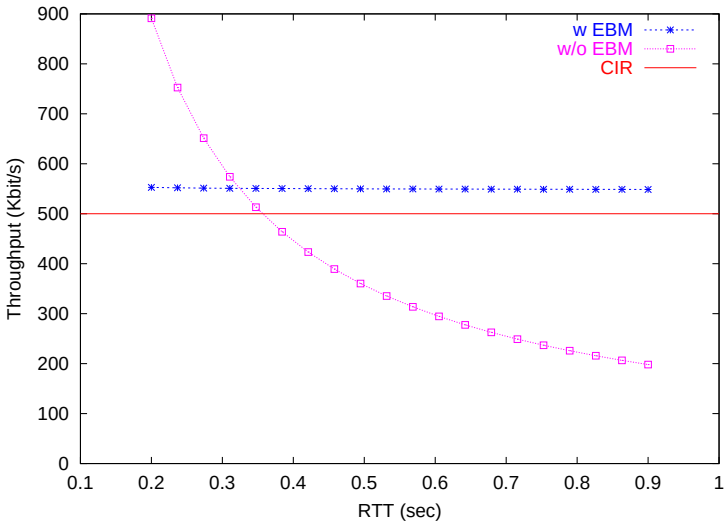


Fig. 6.4 *EBM*

6.6 CONCLUSION DU CHAPITRE

Parmi les multiples méthodes de marquage présentées dans ce chapitre, il en réside deux grandes familles :

- les marquages quantitatifs :
  - marquage basé sur des équations TCP [PFTK98, EGS02, YR01a] ;
  - marquage basé sur la mémorisation [KAJ01] ;

  - marquage inspiré de TSW [HG99b, CF98].
- les marquages qualitatifs :
  - marquage inspiré de [MSZ03].

Le conditionnement de flots TCP pour la classe AF doit s'effectuer en prenant en compte les points suivants :

- le débit TCP est fortement lié aux paramètres suivants : la probabilité de perte d'un paquet, son RTT et son débit assuré (*target rate*) ;
- la perte d'un paquet hors profil est préjudiciable pour le débit moyen ;
- le calcul de probabilité de perte et l'estimation du RTT et du RTO sont complexes.

Chapitre 7

Propositions de conditionneurs TCP pour la classe AF.

7.1 INTRODUCTION

Plutôt que d'élaborer une nouvelle stratégie de marquage qui sera principalement basée sur le marquage en excès des paquets hors profil, nous proposons dans ce chapitre l'utilisation de lisseurs de trafic. Ces lisseurs de trafic conditionnent le flot ou l'agrégat de flots pour le rendre conforme à son profil de trafic. En aucun cas ils ne se substitueront à la méthode de marquage choisie en aval. Deux types de lisseurs sont étudiés : (1) un lisseur sans interaction directe avec le réseau : le Dynamic Shaper et (2) un lisseur avec interaction directe : le Penalty Shaper. Enfin, nous proposerons un raffinement du Penalty Shaper nommé AIMD Penalty Shaper.

7.1.1 Quelles solutions ?

L'idée de l'utilisation d'un lisseur de trafic dynamique est venu à la suite des mesures effectuées au chapitre 5. Ce sont les résultats de la figure 7.1 qui ont décidé de la direction exploratoire. Cette figure reprend l'expérience présentée sur la figure 5.6[1] du chapitre 5 mais une remise en forme du flot de référence est effectuée selon son profil (tous les paquets se trouvent alors dans le profil). Le flot de référence obtient un débit constant quelquesoit le nombre de flots dans l'agrégat. Ceci démontre que la perte d'un paquet hors profil est préjudiciable au débit assuré et que la politique de marquage a donc un rôle prédominant dans la différenciation de services. Cependant, le lissage du trafic ne peut être une solution pour obtenir une différenciation de services car en émettant aucun paquet hors profil, le débit de la source est limité à son débit minimum (indiqué par le profil). Elle ne peut donc

[1] Rappel : dans ce test, le flot de référence et tous les autres flots perturbateurs étaient conditionnés par un TBM. Le changement effectué ici est que le flot de référence est remis en forme et totalement marqué en profil.

concourir à obtenir la bande passante en excès si elle existe. Dans ce cas, le service AS peut devenir un service moins performant que BE. Les contraintes pour un conditionneur dynamique sont :

- préparer le flot au marquage afin que son débit sur le long terme soit supérieur ou égal à $r(i)_{AS}$. Le nombre de paquets marqués en profil doit être proche de 100% ;
- isoler les flots TCP les uns des autres afin d'empêcher qu'un flot TCP ne consomme les ressources d'un autre flot TCP qui n'a pas atteint son débit assuré ;
- préserver un débit minimum au flot TCP tout en ne le privant pas d'atteindre un débit maximum comme s'il utilisait la classe de trafic BE.

Les hypothèses de fonctionnement pour ce type de conditionneur sont :

- la perte d'un paquet en profil est exceptionnelle et correspond à un mauvais dimensionnement du réseau ;
- la perte d'un paquet hors profil reste normale et indique une congestion (un excès conjoncturel de trafic opportuniste).

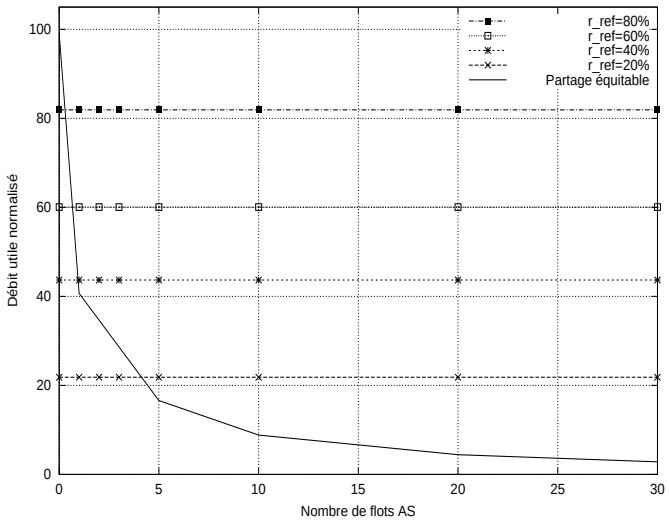


Fig. 7.1 Débit utile normalisé du flot de référence lissé en fonction du nombre de flots AS

7.1.2 Motivation

Les techniques présentées au chapitre 6 se fondent, pour la majeure partie, sur un marquage probabiliste des paquets avant leur entrée dans le réseau DiffServ. Nous avions vu que dans le cas ou nous avions très peu, voire pas de bande passante en excès, la perte d'un paquet hors profil était fortement préjudiciable pour le débit moyen du flot. Dans [NPE00], il est proposé un conditionneur intelligent pour les agrégats de trafic TCP dans le service AF. Il est montré que l'équité de TCP des deux agrégats, en fonction de leur RTT de test, n'est pas respectée. Le conditionneur proposé résout au mieux ce problème. Chacun des deux agrégats, avec ou sans conditionneur, atteint son débit assuré dans les mesures présentées car il y a 60% de bande passante en excès dans les exemples de tests. Notre

intérêt, dans l'architecture DiffServ inspirée du projet @IRS, n'est pas en premier lieu d'assurer une équité au niveau des agrégats TCP, mais de pouvoir garantir un débit de transfert. Les solutions de marquage probabilistique de paquet hors profil dérivées du *Time Sliding Window Color Marker* ont de bons résultats lorsque la bande passante en excès dans le réseau DiffServ est suffisante mais pas quand celle-ci vient cruellement à manquer. La principale conséquence du manque de bande passante est que la probabilité de perte d'un paquet hors profil est proche de 1 et donc qu'un marquage probabiliste n'aura aucune influence dans ce genre de situation. D'où l'idée d'un lisseur adaptatif afin d'obliger TCP à ne pas devenir trop gourmand vis-à-vis des ressources libres. Ainsi, les ressources disponibles se retrouvent partagées avec les flux ayant des difficultés à atteindre leur débit assuré.

7.1.3 A propos du conditionnement de trafic

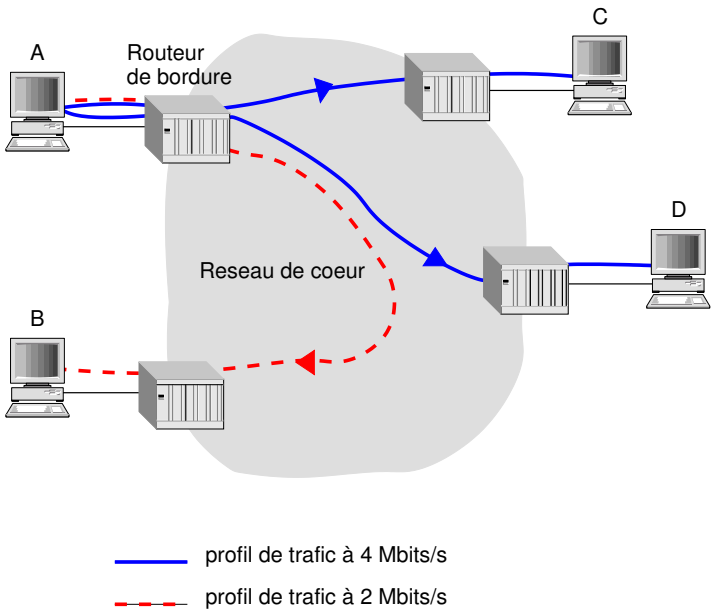


Fig. 7.2 Exemple de conditionnement de trafic

Les stratégies de marquage utilisées dans le conditionnement des flots à l'entrée d'un domaine DiffServ influencent la probabilité de perte des paquets du flot et par conséquent, le comportement de ce dernier à l'intérieur du réseau. De plus, il est nécessaire d'opérer un conditionnement qui puisse facilement passer à l'échelle afin de donner aux FAI[2] une solution facilement exploitable. Nombreux sont les conditionneurs présentés au chapitre 6 qui, à cause de la complexité de leur conditionnement, ne sortiront jamais du cadre de la simulation. Dans tout ce qui suivra, nous avons choisi de faire le conditionnement de trafic de la façon suivante : chaque client émettant un ou plusieurs flots vers une ou plusieurs destinations aura un profil de trafic par destination comme l'illustre la figure 7.2.

[2]Fournisseur d'Accès Internet

Sur cette figure, le client A oblige le routeur de bordure à mettre en œuvre trois conditionneurs de trafic distincts : deux conditionneurs de contrat $4Mbits/s$ et un conditionneur de contrat $2Mbits/s$. Afin de ne pas faire de confusion avec le principe d'agrégation défini dans [BBC+98], dans le reste de l'étude, nous appellerons agrégats TCP client, les agrégats TCP émis d'une source vers une destination précise. Ces agrégats pourront comporter de un à plusieurs flots TCP client. Le principal avantage de cette solution est que nous conditionnons un agrégat avec des flots à RTT de même ordre de grandeur. Par conséquent, nous nous affranchissons du problème complexe de l'évaluation du RTT d'un flot nécessaire au fonctionnement des conditionneurs présentés au chapitre 6. La complexité provient du fait que pour mesurer un RTT correctement, il faut effectuer cette mesure près de la source. Sinon, l'estimation du RTT peut accuser une trop grande marge d'erreur [JD02]. Plusieurs des algorithmes présentés au chapitre 6 mesurent un RTT en partant de l'hypothèse que la voie retour est identique à la voie aller, ce qui n'est pas forcément le cas. Dans l'Internet, cette asymétrie peut être dûe à différentes raisons [Pax97]. En revanche, ce conditionnement n'apporte aucune contribution quant à l'influence de la composition des agrégats. Nous verrons que le nombre de microflots à l'intérieur de ces agrégats client a une importance capitale sur le débit assuré. [SNP99] a montré que des agrégats à nombre de microflots différents et à même débit assuré obtiennent un débit moyen qui est fonction du nombre de microflots contenus dans l'agrégat. Notre solution de lisseur de trafic couplé à un conditionneur/marqueur tel que le *token bucket marker* ou le *time sliding window marker* qui sont les conditionneurs les plus répandus dans les mises en œuvres actuelles, est facilement déployable et résiste au changement du facteur d'échelle.

7.2 LA PLATEFORME DE TESTS

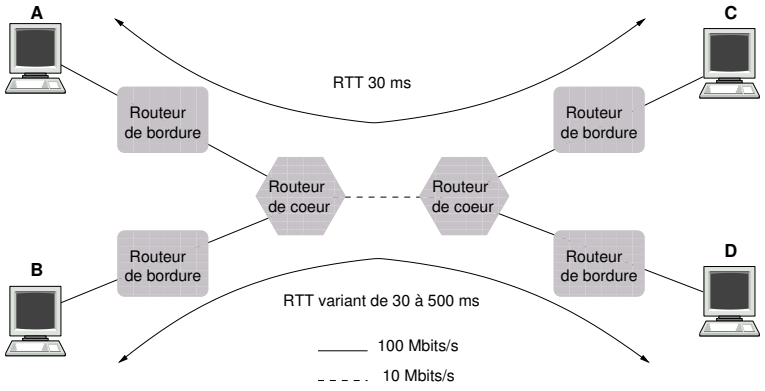


Fig. 7.3 Experimental testbed

Avant de présenter les conditionneurs de trafic développés, cette section introduit la plateforme de test que nous avons utilisée pour leur validation. Cette plateforme permet de tester nos propositions dans un cadre d'utilisation réelle sur des machines équipées d'une pile IP largement répandue et faisant office de référence : la souche Kame[3]. Comme le

[3]Pour de plus amples informations voir le site web du projet : `http://www.kame.net`

montre la figure 7.3, nous utiliserons pour nos tests la topologie dite *dumbbell*. Cette topologie étant largement répandue (voir notamment les expérimentations des articles suivants : [MSZ03, EGS02, NPE00, WMB00, YR99]), elle permettra une comparaison aisée avec les solutions existantes. Le banc de test est composé de machines tournant sous FreeBSD configurées de la manière suivante :

- routeur de bordure ; un *token bucket marker* (TBM) couplé avec une de nos propositions ;
- routeur de cœur ; une file RIO développée dans ALTQ [Cho99]. Concernant notre proposition de lisseur avec interaction directe avec le réseau, la fonctionnalité de marquage du drapeau ECN[4] [RFB01] a été ajoutée. Cette fonctionnalité, qui sera détaillée plus loin, permet de reporter les pertes de paquets hors profil.

La banc de test comprend deux paires d'hôtes (émetteur - récepteur) pour la production de flots TCP utilisés pour les mesures. La limitation à $10Mbits/s$ du lien entre les routeurs de cœur est obtenue grâce à l'utilisation d'une carte réseau ne supportant que du transfert $10Mbits/s$ *full-duplex*. La voie retour prise par les acquittements est isolée de la voie aller de par la configuration des expérimentations et de l'utilisation de liens *full-duplex* de la plateforme de tests. La justification du choix de cette plateforme a été présentée en 5.2.2.

Les principaux paramètres et hypothèses de ces tests sont :
- nous utilisons `Iperf 1.7.0`[5] comme générateur de trafic ;
- la génération de trafic est faite de A vers C (A, C) et de B vers D (B, D) pendant 120 secondes ;
- chaque flot est transmis en TCP ;
- les paquets ont une taille de 1024 octets ;
- `Iperf` utilise un fenêtre maximale TCP : $Wmax = 64Ko$;
- chaque ensemble de flots entre deux hôtes est conditionné par un TBM couplé, ou non couplé avec les mécanismes présentés (Penalty Shaper ou Dynamic Shaper) ;
- le paramètre B du TBM vaut 1 paquet. Ce choix fait suite à l'étude menée dans [SND+00] qui montre que l'utilisation d'une grande valeur pour le paramètre B d'un TBM ne fait qu'augmenter les rafales du trafic en profil ;
- nous utilisons la configuration RIO suivante :
  $(min_{out}, max_{out}, p_{out}, min_{in}, max_{in}, p_{in}) = (1, 63, 0.1, 64, 128, 0.02)$, la taille de la file correspondant à : $2 * Wmax$;
- après 120 secondes, `Iperf` donne un débit moyen obtenu par le flot entre les hôtes (A, C) et (B, D) ;
- les résultats présentés donnent une moyenne de cinq mesures consécutives.

Autre paramètre important à prendre en compte de la capacité allouée au service AS. Lors de l'étalonnage de la plateforme, le débit TCP maximum mesuré obtenu est de $9.44Mbits/s$. En l'étalonnant à l'aide de `Iperf`, un flot UDP obtient un débit à la réception de $9.57Mbits/s$. Pour tous les tests ci-dessous, nous considérons que le débit maximum possible est de $10Mbits/s$, ce qui correspond au débit effectif[6] de la plateforme.

[4] Explicit Congestion Notification

[5] `http ://dast.nlanr.net/Projects/Iperf/`

[6] Le débit effectif se définit comme le nombre maximum de bits qui peuvent être transférés en une seconde sur le câble. Cette valeur est liée aux caractéristiques physiques du médium.

7.3 LE *DYNAMIC SHAPER* (DS)

Ce lisseur de trafic dynamique possède des caractéristiques similaires à celles présentées dans [BdC00]. L'objectif principal de ce lisseur est de perturber aussi peu que possible le principe AIMD[7] du contrôle de congestion TCP. Le Dynamic Shaper diminue la nature inhérente en rafale de TCP. Il est à noter que le Dynamic Shaper est lié au principe de contrôle de congestion TCP qu'il utilise pour calculer le taux de lissage (*shaping rate*) du lisseur de trafic. Un simple *token bucket marker* (TBM) est incapable d'identifier une rafale TCP et donc il risque de marquer le trafic plus hors profil que nécessaire. Afin d'éviter les rafales de paquets hors profil, chaque agrégat TCP est lissé dans le but de changer la sporadicité du trafic TCP. Comme l'illustre la figure 7.5, le Dynamic Shaper permet de changer la sporadicité du trafic par une réorganisation temporelle de celui-ci. L'idée de base est de déterminer le débit du lissage en fonction de la variation du débit de TCP. En effet, le débit TCP dépend de sa perception de l'état du réseau. Une augmentation du débit signifie une absence de congestion dans le réseau. Le débit de lissage peut augmenter. A l'inverse, une diminution du débit indique une surcharge et le débit de lissage doit diminuer. A chaque arrivée de paquet, plutôt que de calculer le taux de lissage en fonction de l'occupation de la file comme dans [BdC00], nous mesurons le débit TCP grâce à un algorithme de *Time Sliding Window* (TSW) [FSA00]. La mesure d'un niveau de remplissage d'une file d'attente donne la différence entre deux débits : celui d'arrivée et de sortie. La mesure du débit TCP ne prend en compte que le débit d'arrivée (et donc indépendante du débit de sortie) et permet de prendre en compte le délai qui sépare les segments de données. Au début de l'algorithme, le paramètre r du Dynamic Shaper est mis à $r(i)_{AS}$, ce qui correspond au débit assuré de l'agrégat TCP i. Si le débit mesuré $r(i)_{measured}$ est inférieur à $r(i)_{AS}$, le Dynamic Shaper garde une valeur de lissage égale à : $r(i)_{AS}$, sinon, celle-ci est égale à $r(i)_{measured}$. Le Dynamic Shaper recalcule le taux de lissage à chaque transmission de x paquets. Cette valeur correspond à la période d'action du lisseur de trafic. Après plusieurs essais de calibration, nous avons pris dans nos tests $x = 50$. Cela signifie qu'à chaque fois que sont émis 50 paquets, le Dynamic Shaper lisse à un taux égal à $r(i)_{measured}$ ou à $r(i)_{AS}$. Nous aurions pu choisir une période d'action égale au RTT mais pour cela, une évaluation de celui-ci devenait nécessaire. Comme il est impossible de déterminer l'état du réseau, le RTT varie constamment. C'est la raison pour laquelle nous avons décidé d'utiliser une période d'action n'ayant aucune relation avec le RTT (voir section 7.1.3). Plus grande est la période d'action, plus faible est le lissage. Le Dynamic Shaper permet d'obtenir un gain de trafic en profil. Comme il l'est montré sur la table 7.1, un simple flot TCP conditionné par un TBM avec un Dynamic Shaper obtient un gain de trafic en profil de 7% par rapport à l'utilisation du TBM seul. Ce flot a été mesuré sur la plateforme de la figure 7.3 et avait pour débit assuré $r(i)_{AS} = 5Mbits/s$.

7.3.1 Evaluation des performances du *Dynamic Shaper*

Cette section présente les résultats obtenus sur une plateforme réelle avec le Dynamic Shaper. Nous évaluons les performances du Dynamic Shaper lorsque les agrégats ont une composition différente ou identique et des RTT égaux ou différents.

[7] Additive Increase Multiplicative Decrease

	IN	IN dropped	OUT	OUT dropped
TBM	51%	0	49%	0.91%
TBM+DS	58%	0	42%	0.63%

Tab. 7.1 TBM (Token Bucket Marker) seul contre TBM avec Dynamic Shaper

```
Initialization
  count = 0
  AVG_INTERVAL = 1

Upon each packet arrival
Bytes_in_win = R_measured *  AVG_INTERVAL ;
New_bytes    = Bytes_in_win + Packet_size;
R_measured   = New_bytes/( interval_sec + AVG_INTERVAL);

IF ( R_measured < R_AS )
r = R_AS
ELSE
r = R_measured
ENDIF

count++

IF ( count == 50 )
   shaping rate = r
   count = 0
ENDIF
```

Fig. 7.4 Algorithme du Dynamic Shaper

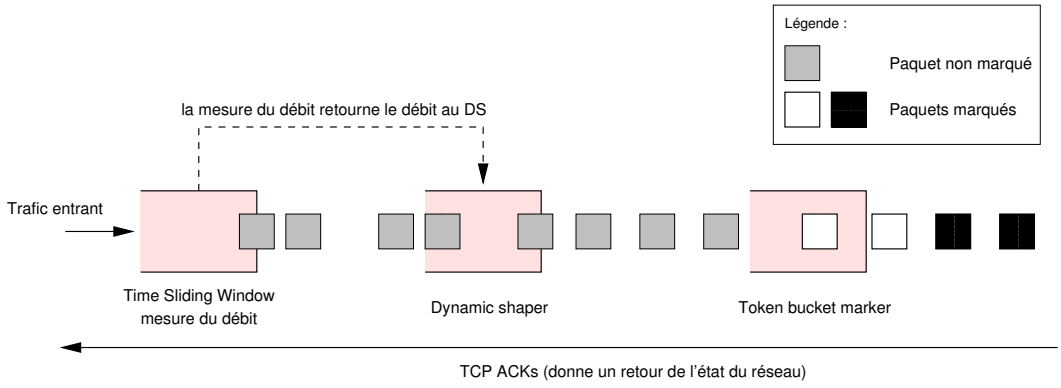


Fig. 7.5 Architecture du Dynamic Shaper

Impact de l'agressivité des agrégats dans un réseau surdimensionné

La caractéristique principale de cette solution est : quelquesoit le nombre de flots dans l'agrégat, le Dynamic Shaper permet d'obtenir un meilleur débit moyen qu'avec un simple

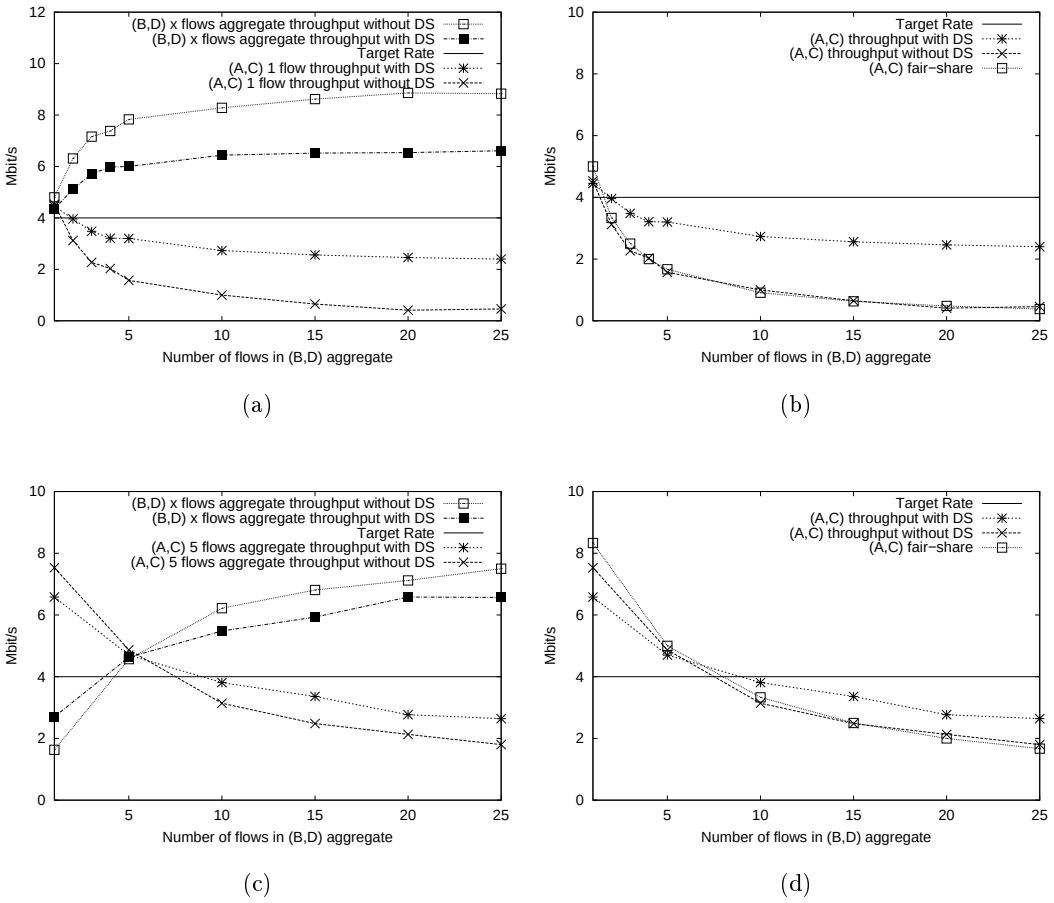


Fig. 7.6 Débit des agrégats TCP en fonction de l'agressivité des agrégats

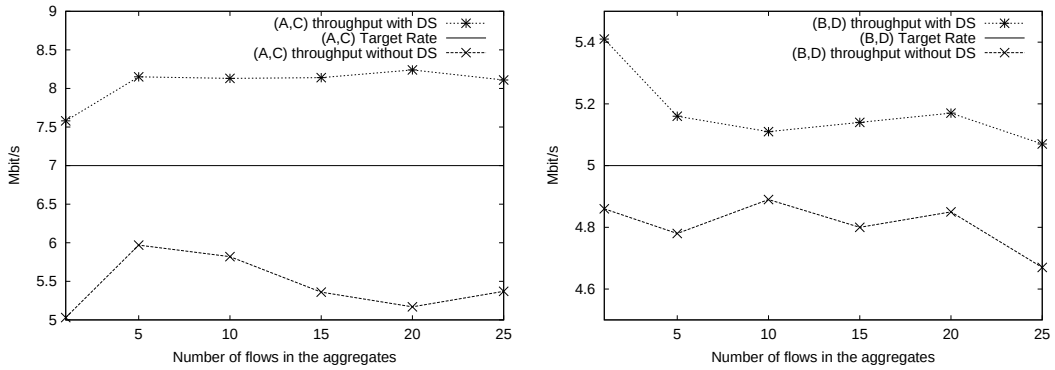
TBM. Ce résultat est présenté sur la figure 7.6. Lorsque deux agrégats, avec un nombre de microflots différents, sont dans un réseau, celui qui possède le plus grand nombre de microflots surpasse celui qui possède le plus petit nombre. Ce problème d'agressivité[8] entre les agrégats, causé par le multiplexage TCP, a été soulevé par [SNP99]. Dans ce test, deux agrégats sont en compétition et chacun a un RTT de $30ms$. L'agrégat (A, C) a un 1 seul microflot et l'agrégat (B, D) a un nombre variable de microflots allant de 1 à 25. Le débit assuré des deux agrégats est : $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. La capacité totale allouée au service assuré est : $R_{AS} = 8Mbits/s$. La ressource allouée au service assuré est : $C_{AS} = 10Mbits/s$. Cette ressource correspond à la capacité du goulot d'étranglement. Etant donné que nous avons $2Mbits/s$ de bande passante en excès, nous sommes dans le cas d'un réseau surdimensionné. La figure 7.6 (a) nous donne le débit obtenu par les deux agrégats. Le Dynamic Shaper est capable d'atteindre un débit plus proportionnel au débit assuré grâce à l'ajout du Dynamic Shaper qu'avec l'utilisation d'un simple TBM seul. Même si le débit assuré de l'agrégat (A, C) n'est pas atteint, le Dynamic Shaper permet de minimiser l'effet de l'agressivité entre agrégats. Pour des raisons de lisibilité, nous avons

[8] On définira comme agressif, l'agrégat composé par le nombre le plus grand de microflots.

dessiné le débit obtenu par l'agrégat (A, C) à côté de la courbe du partage équitable (*fair share*). La figure 7.6 (b) montre que le débit obtenu avec le TBM reste près du partage équitable et que nous obtenons un meilleur débit, plus proche du débit désiré, avec le Dynamic Shaper.

Les mesures des figures 7.6 (c) et (d) présentent le même scénario sauf que l'agrégat (A, C) a un nombre fixé de 5 microflots. Lorsque (B, D) a moins de 5 microflots, (A, C) est le plus agressif des agrégats et respectivement lorsque (B, D) a plus de 5 microflots, (A, C) est le moins agressif des agrégats. Ces figures nous donne un résultat complémentaire apportant les mêmes conclusions que les tests présentés en figure 7.6 (a) et (b). Il est remarquable que l'écart avec le débit assuré est réduit du fait de la réduction de l'écart de l'agressivité.

Impact de la validité de l'assurance, en fonction de la taille de l'agrégat, dans un réseau surdimensionné



(a) (b)

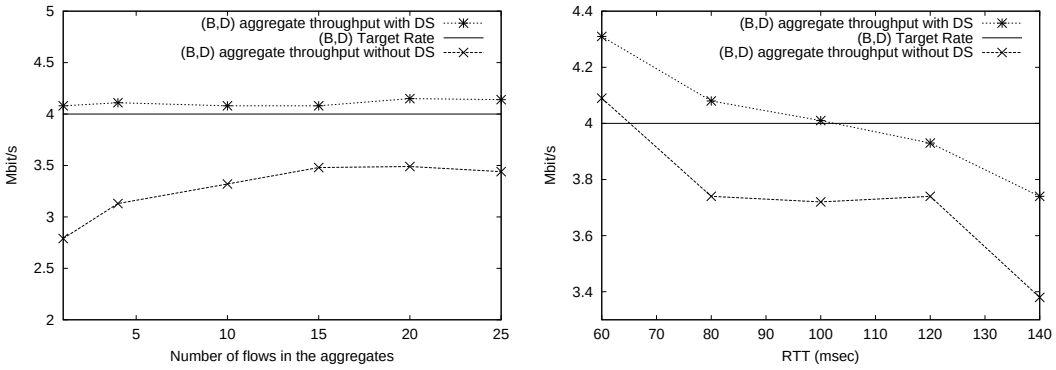
Fig. 7.7 Débit de l'agrégat TCP avec différents débits assurés

Dans cette partie, deux agrégats sont en compétition et le RTT de chaque est égal à $30ms$. Chaque agrégat a le même nombre de microflots. Nous faisons varier le nombre de microflots de 1 à 25. Le tableau 7.2 présente les résultats obtenus pour deux scénarios. Dans le premier, (A, C) et (B, D) ont respectivement un débit assuré de $r(A, C)_{AS} = 3Mbits/s$ et $r(B, D)_{AS} = 5Mbits/s$. Dans le second, ils ont respectivement un débit assuré de $r(A, C)_{AS} = 7Mbits/s$ et $r(B, D)_{AS} = 1Mbits/s$. Ainsi, les deux scénarios illustrent le cas où les agrégats ont des débits assurés proches ou distants dans le cas d'un réseau surdimensionné. Nous avons tracé sur la figure 7.7 (a) le débit obtenu par l'agrégat (A, C) lorsqu'il souhaite un débit assuré de $r(A, C)_{AS} = 7Mbits/s$ (cela correspond à la colonne 5 du tableau 7.2). La figure 7.7 (b) donne le débit obtenu par l'agrégat (B, D) lorsqu'il désire un débit assuré de $r(B, D)_{AS} = 5Mbits/s$ (cela correspond à la colonne 4 du tableau 7.2). Chaque agrégat atteint son débit assuré même si le nombre de microflots qui le compose augmente.

Test	# flots dans l'agrégat (A, C) versus (B, D)	A vers C RTT=30ms	B vers D RTT=30ms	A vers C RTT=30ms	B vers D RTT=30ms
TBM only	1 vs 1	4.53/3	4.86/5	5.03/7	4.36/1
TBM+DS	1 vs 1	3.50/3	5.41/5	7.58/7	1.04/1
TBM only	5 vs 5	4.63/3	4.78/5	5.97/7	3.83/1
TBM+DS	5 vs 5	4.04/3	5.16/5	8.15/7	1.11/1
TBM only	10 vs 10	4.43/3	4.89/5	5.82/7	3.76/1
TBM+DS	10 vs 10	4.08/3	5.11/5	8.13/7	1.02/1
TBM only	15 vs 15	4.86/3	4.80/5	5.36/7	4.23/1
TBM+DS	15 vs 15	3.84/3	5.14/5	8.14/7	1.10/1
TBM only	20 vs 20	4.47/3	4.85/5	5.17/7	3.86/1
TBM+DS	20 vs 20	3.95/3	5.17/5	8.24/7	1.01/1
TBM only	25 vs 25	4.30/3	4.67/5	5.37/7	4.37/1
TBM+DS	25 vs 25	3.89/3	5.07/5	8.11/7	1.00/1

Tab. 7.2 Réseau surdimensionné (légende : débit de transfert en $Mbits/s$ / débit assuré en $Mbits/s$)

Impact du RTT dans un réseau surdimensionné



(a) (b)

Fig. 7.8 Débit TCP en fonction du RTT

Même si l'écart du nombre de flots entre les agrégats est important et même si ils ont une grande différence de RTT, le Dynamic Shaper est capable d'atteindre le débit assuré sollicité par un agrégat. Les figures 7.8 (a) et (b) illustrent ce cas. La figure 7.8 (a) nous donne le débit obtenu par l'agrégat (B, D), avec un $RTT = 100ms$, en compétition avec un agrégat (A, C), avec un $RTT = 30ms$, en fonction du nombre de microflots. Le débit assuré pour (A, C) et (B, D) est $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. Nous avons choisi un $RTT = 100ms$ afin de démontrer que le RTT n'influence pas le Dynamic Shaper. La figure 7.9 présente une évaluation analytique du débit TCP : (a) en fonction de la probabilité de perte et (b) en fonction du RTT pour une probabilité de perte nulle. Ces courbes proviennent des équations développées dans [PFTK98] présentées en annexe B.

Nous pouvons voir que lorsqu'il n'y a aucune perte dans le réseau, pour un $RTT = 100ms$, un flot TCP est capable d'atteindre un débit théorique de $5.1Mbits/s$. Comme le montre la figure 7.8 (a), un simple TBM ne peut pas obtenir un débit théorique faisable de $4Mbits/s$. Grâce au Dynamic Shaper, l'agrégat atteint son débit assuré et l'augmentation du nombre de flots dans les agrégats n'influence pas le débit assuré voulu. Enfin, la figure 7.8 (b) présente le débit obtenu par un agrégat (B, D) de 10 microflots en compétition avec un agrégat (A, C) de 10 microflots également. Pour l'agrégat (A, C), le RTT est égal à $30ms$ et pour l'agrégat (B, D), nous augmentons graduellement son RTT de $60ms$ à $140ms$. Il apparaît que l'agrégat (B, D) atteint son débit assuré lorsque celui-ci reste faisable (c'est-à-dire lorsque $r(B, D)_{AS} > Wmax/RTT$). Lorsque $r(B, D)_{AS} < Wmax/RTT$ (c'est-à-dire : $RTT \geqslant 100ms$), le Dynamic Shaper obtient un meilleur débit qu'avec un TBM. A titre d'information, nous avons fait ce test avec un nombre de flots allant de 1 contre 1 jusqu'à 25 contre 25 et obtenu des résultats similaires. Nous ne présentons ici que la mesure de 10 contre 10 flots.

Impact du nombre de flots dans un réseau sous-dimensionné

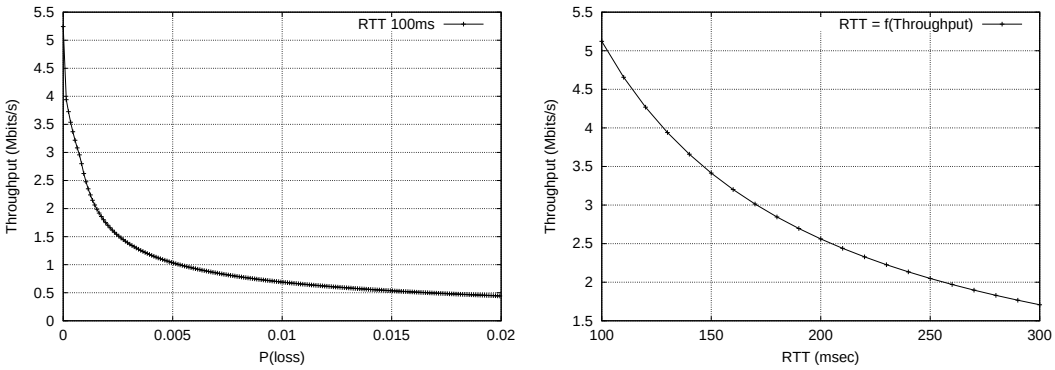
Test	# flots dans l'agrégat (A, C) versus (B, D)	A vers C RTT=30ms	B vers D RTT=30ms	A vers C RTT=30ms	B vers D RTT=30ms
TBM only	1 vs 1	4.82/8	4.34/4	4.83/10	4.52/2
TBM+DS	1 vs 1	5.35/8	3.98/4	6.60/10	2.60/2
TBM only	5 vs 5	5.02/8	4.34/4	5.25/10	4.20/2
TBM+DS	5 vs 5	5.14/8	4.25/4	6.47/10	2.95/2
TBM only	10 vs 10	5.20/8	4.24/4	5.31/10	4.09/2
TBM+DS	10 vs 10	5.09/8	4.09/4	6.34/10	3.02/2

Tab. 7.3 Réseau sous-dimensionné (légende : débit de transfert en $Mbits/s$ / débit assuré en $Mbits/s$)

Sur les deux scénarios du tableau 7.3, la capacité totale allouée au service assuré est $R_{AS} = 12Mbits/s$. Il n'y a aucune bande passante en excès. Nous sommes donc dans le cas d'un réseau sous-dimensionné. Les mesures du tableau 7.3 montrent que l'utilisation du Dynamic Shaper permet d'obtenir un débit TCP plus proportionnel au débit assuré qu'avec le TBM seul. Nous ne nous attarderons pas sur ce cas parce qu'il correspond à un mauvais dimensionnement du réseau et donc qu'il sort du cadre des hypothèses que nous avions définies en section 6.3.

7.3.2 Conclusion pour le Dynamic Shaper

La première conclusion que nous pouvons tirer des mesures faites avec le Dynamic Shaper est que nous sommes capables d'obtenir un débit garanti si les agrégats en compétition ont le même nombre de microflots. Cela reste vrai quelquesoit la différence existante entre leur RTT et leur débit assuré. Lorsque nous travaillons avec des agrégats à composition de microflots différente (et dans le cas d'un réseau sous-dimensionné), le Dynamic Shaper est capable d'obtenir un débit plus proportionnel au débit assuré qu'un simple *token bucket marker*. Il est à noter que la composition des agrégats peut influencer le débit désiré et



(a) en fonction de la probabilité de perte : $P(loss)$

(b) en fonction du RTT avec $P(loss) = 0$

Fig. 7.9 Débit TCP théorique

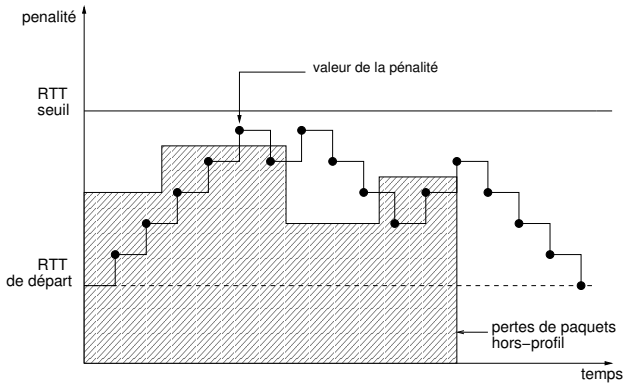
compenser le désavantage d'un long RTT ou d'un débit assuré difficile à obtenir. La solution proposée est facilement déployable et peut être employée avec les conditionneurs les plus répandus tels que le *token bucket marker* ou le *time sliding window marker*. De plus, la capacité allouée au service assuré est proche de la capacité du goulot d'étranglement. Les tests ont été faits avec une capacité allouée théorique de 80%, correspondant à un sur dimensionnement de 20% du réseau. Plus exactement, en ramenant à la capacité exacte du réseau, la bande passante en excès est de 18.9% (voir la configuration de la plateforme de tests en section 7.2). Ce chiffre est la moitié de la valeur type donnée par [dR99] pour caractériser un réseau surdimensionné[9].

Malgré ces résultats encourageants, ce lisseur de trafic ne répond pas encore totalement à nos attentes. En effet, la principale critique que l'on peut lui faire est qu'il est encore trop sensible au nombre de microflots constituant l'agrégat. De plus, notre période d'action, définie en section 7.3, est difficile à déterminer et à configurer. L'idée du Penalty Shaper présenté dans la section suivante tente de donner une réponse élégante à ces deux inconvénients majeurs.

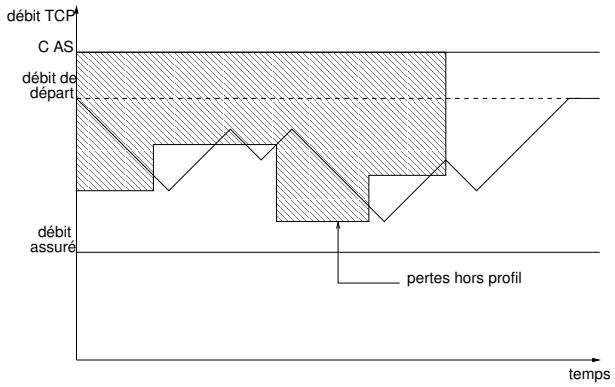
7.4 LE *PENALTY SHAPER* (PS)

Nous présentons ici un lisseur de trafic à pénalité appelée : Penalty Shaper. Ce lisseur de trafic va appliquer une pénalité de délai à un flot conditionné s'il y a des pertes de paquets hors profil dans le réseau et si le flot conditionné surpasse son débit assuré. L'idée de base est que la pénalité appliquée est fonction des pertes de paquets hors profil dans le réseau.

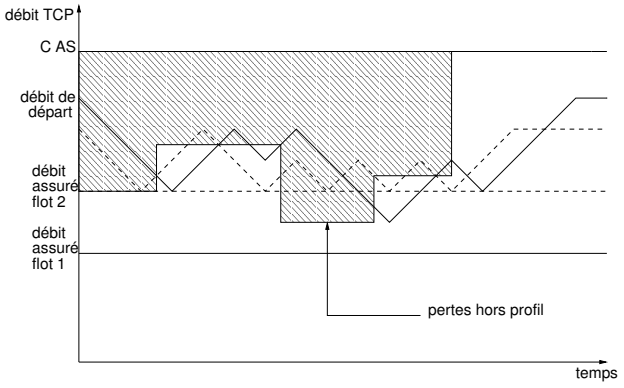
[9] [dR99] considère qu'un réseau est surdimensionné lorsqu'il possède au moins 40% de bande passante en excès.



(a)



(b)



(c)

Fig. 7.10 Résultats théoriques obtenus avec le Penalty Shaper (PS)

7.4.1 Conception du *Penalty Shaper*

Plutôt que d'augmenter la valeur p (correspondante à la probabilité de perte) de l'équation (6.6) du flot le plus opportuniste dans le réseau, le Penalty Shaper va artificiellement augmenter le délai du flot. Cela a pour conséquence directe l'augmentation du RTT du flot. Mathématiquement, comme le montre l'équation (6.6), augmenter la valeur RTT est similaire à augmenter la valeur p en termes de débit moyen TCP. Nous avons choisi d'opérer sur l'augmentation de la valeur RTT plutôt que sur l'augmentation de la valeur p pour les raisons suivantes :

- l'augmentation de la probabilité de perte p augmente la perte de paquets TCP et provoque de fortes oscillations au niveau du débit instantané du flot ;
- cette technique nous affranchit des mesures complexes induites par la modification de la valeur p. En effet, pour modifier la probabilité de perte d'un flot grâce à une stratégie de marquage, il est nécessaire de faire une estimation de la probabilité courante des pertes de paquets dans le réseau. Ce calcul peut être réalisé par la fonction d'estimation d'intervalle de pertes exposée dans [FHPW00]. C'est d'ailleurs la méthode utilisée par [EGS02] pour sa méthodologie de marquage. Le problème est que ce calcul doit être appliqué à chaque microflot de l'agrégat, ce qui la rend relativement complexe à mettre en œuvre ;
- enfin, l'augmentation du délai est une technique moins brutale que l'augmentation de la probabilité de perte. En effet, en augmentant la valeur RTT de l'équation (6.6) nous diminuons la valeur p. Tout en rendant le flot moins opportuniste, nous lui diminuons sa probabilité de perte.

[YR99] a déjà montré que limiter les paquets hors profil dans un réseau était une bonne politique pour atteindre le débit assuré et également une bonne solution pour éviter les oscillations de débit TCP. Avec le Penalty Shaper, nous allons réduire les pertes de paquets hors profil en appliquant une pénalité de délai aux flots les plus opportunistes dans le réseau. Ainsi, lorsqu'un routeur de cœur géré par une discipline RIO[10] [CF98] jette des paquets hors profil, il marquera le drapeau ECN[11] des paquets en profil présent dans la file RIO. Dans un réseau correctement dimensionné, il n'y a pas (ou presque pas) de perte de paquet en profil. Ainsi, un routeur de bordure va être « conscient », à la réception de ces paquets marqués, qu'au moins un agrégat TCP conditionné est opportuniste. Ce flot ou cet ensemble de flots emprunte le même chemin réseau. Le routeur de bordure évalue alors le débit d'émission de ces flots conditionnés grâce à un algorithme de *Time Sliding Window* (TSW) [FSA00].

Si le débit d'émission est supérieur au débit assuré, le routeur de bordure considère que le flot peut être opportuniste dans le réseau et va alors appliquer une pénalité de délai à celui-ci. Cette pénalité sera conservée, voire augmentée tant que le réseau retournera comme information des pertes de paquets hors profil. Cette pénalité va alors permettre une augmentation du RTT du flot et par voie de conséquence, une baisse de son débit

[10] RED with IN and OUT

[11] Nous utilisons ici le drapeau ECN comme signalisation de la congestion au sein d'un routeur RIO. Les hôtes des extrémités doivent être compatibles ECN (ECN-*capable*). Les implémentations actuelles font la distinction entre l'utilisation de la signalisation ECN sans action sur le débit TCP (*passive* ECN-*capable*) et avec action sur le débit TCP (*active* ECN-*capable*). Dans notre cas, nous utilisons le drapeau ECN sans action sur le débit de l'émetteur.

d'émission. La figure 7.10 illustre le comportement, en termes de débit TCP, obtenu avec cet algorithme. L'aire grise de la figure symbolise les pertes hors profil dans le réseau. La courbe de la figure 7.10 (a) donne un exemple de calcul de pénalité en fonction des pertes dans le réseau. Lorsqu'il y a des pertes de paquets hors profil dans le réseau, la pénalité augmente. La ligne de seuil RTT donne la limite théorique à partir de laquelle le débit TCP mesuré est plus petit que le débit assuré (c'est-à-dire lorsque débit assuré $> (Wmax/RTT)$). Si la courbe du débit TCP dépasse ce seuil théorique, l'algorithme diminue la pénalité. La figure 7.10 (b) nous donne le résultat théorique obtenu pour le débit TCP. La valeur C_{AS} des figures 7.10 (b) et 7.10 (c) correspondant à la ressource allouée au service assuré. Si il y a des pertes dans le réseau, une pénalité est appliquée et par conséquent, la courbe du débit TCP diminue. Au contraire, si il n'y a pas de pertes, la pénalité diminue et le débit TCP augmente. Enfin, la figure 7.10 (c) montre le résultat que nous devrions obtenir en termes de débit TCP pour deux flots TCP avec deux débits assurés différents. Pour finir, l'algorithme présenté en figure 7.11 explique comment la pénalité est appliquée.

```
FOR each observation period T
 TSW gives an evaluation of the
 throughput : throughput_measured

 IF throughput_measured < target_rate
    OR there are no out-profile losses
 THEN
     reduce the penalty delay of x
     current_penalty = current_penalty - x
 ELSE
     raise the penalty delay of x
     current_penalty = current_penalty + x
 ENDIF
ENDFOR
```

Fig. 7.11 Algorithme du Penalty Shaper

7.4.2 Architecture

Les routeurs de cœur ont la charge du rejet préférentiel des paquets. Ce rejet donne une bonne indication de l'état de congestion du réseau. Si le réseau est loin d'être dans un état de congestion, les paquets en profil sont rarement rejetés et leur probabilité de rejet est insignifiante. Au contraire, si le réseau est congestionné, presque tous les paquets hors profil sont détruits.

Routeur dans le réseau de cœur

Le mécanisme de rejet utilisé dans les routeurs de cœur est la file RIO [CF98]. RIO est le mécanisme actif de base de gestion de file d'attente approprié pour la mise en œuvre d'un PHB AF (voir la partie 3.3.3 pour de plus amples explications). Les routeurs de

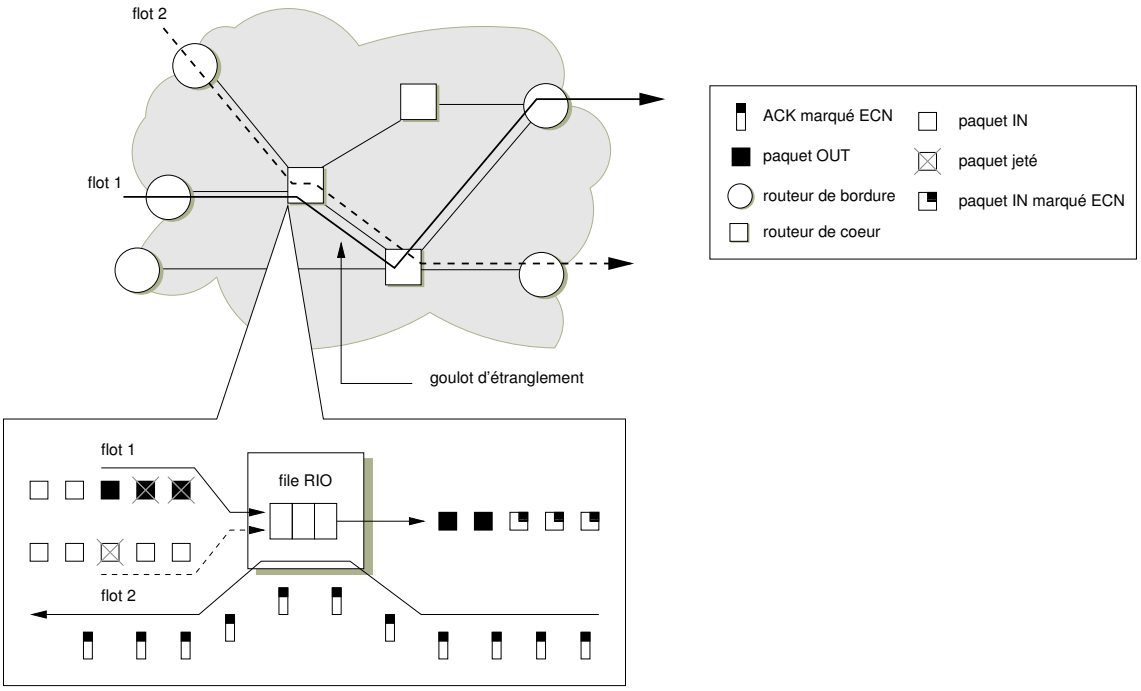


Fig. 7.12 Routeur dans le réseau de cœur

cœur ne maintiennent pas d'état par flot. Dans notre architecture, les routeurs de cœur sont en charge du marquage du drapeau ECN des paquets en profil. La figure 7.12 illustre ce mécanismes. Sur cette figure, deux flots sont émis depuis deux sources distinctes et congestionnent le premier routeur de cœur sur le chemin. Si nous nous trouvons en état de congestion, tous les paquets en profil dans la file d'attente du routeur de cœur ont alors leur drapeau ECN positionné. Ainsi, le routeur de bordure, au retour des acquittements marqués ECN, sera informé qu'un flot, ou qu'un ensemble de flot, conditionné par un de ses conditionneurs, peut être opportuniste dans le réseau.

Router de bordure en entrée du réseau (*ingress edge router*)

Le routeur de bordure utilise un conditionneur/marqueur afin de contrôler et marquer le trafic. Aucune hypothèse n'est faite sur la localisation de ce conditionneur. En effet, le conditionneur/marqueur pourrait être placé du côté du client plutôt que du côté du routeur de bordure. Le Penalty Shaper est juste utilisé pour renforcer la garantie d'obtenir le débit assuré du flot et non pas pour marquer le trafic. Dans nos tests, le routeur de bordure effectue le marquage en profil ou hors profil des paquets à l'aide d'un *token bucket marker* (TBM). Ce mécanisme a déjà été décrit en section 3.2. La conformité du trafic est vérifiée de la même manière que proposée dans [HG99a]. Chaque agrégat TCP client émis pour une destination donnée est conditionné par un seul TBM. Il y a un seul Penalty Shaper couplé avec un seul TBM comme expliqué plus haut en section 7.1.3. La pénalité est appliquée en fonction des informations données par les paquets d'acquittements TCP du trafic en profil marqués ou non marqués ECN. La figure 7.13 illustre le principe fonctionnel de ce mécanisme.

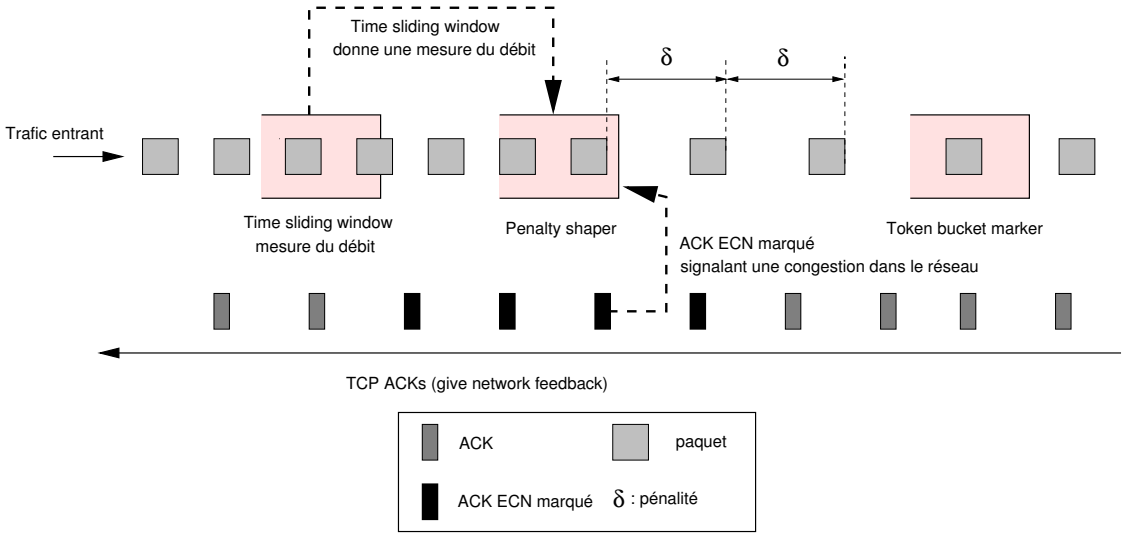
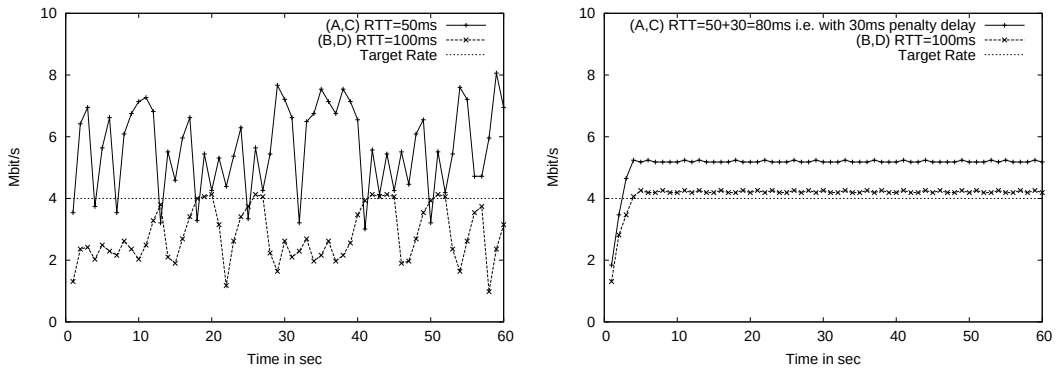


Fig. 7.13 Mécanisme de l'*ingress edge router*

7.4.3 Validation du principe

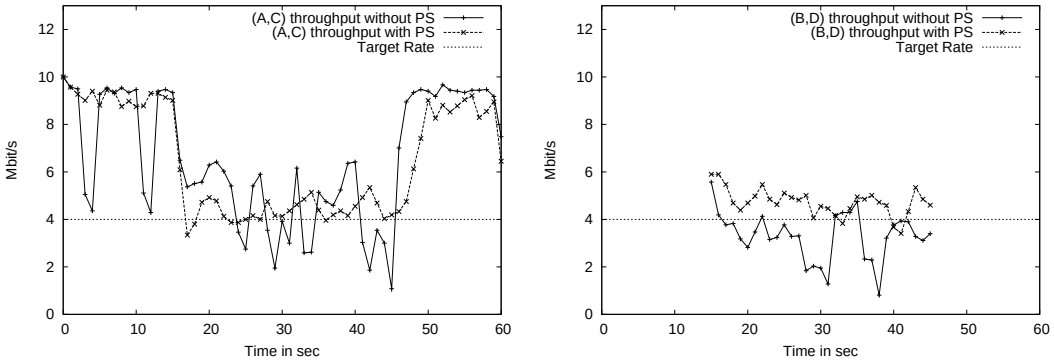


(a) (b)

Fig. 7.14 Débit TCP du côté récepteur sans(a) et avec (b) une pénalité

La figure 7.14 montre deux flots TCP qui traversent la plateforme présentée sur la figure 7.3. Chaque flot est conditionné par un *token bucket marker*. Le débit assuré est de $4Mbits/s$ et est théoriquement faisable. La figure 7.14 (a) donne le débit instantané des deux flots. Le flot (A, C) est le plus agressif car son $RTT = 50ms$. De plus, le flot (B, D) n'atteint pas son débit assuré et le débit instantané des deux flots décrit de larges oscillations. Dans ce cas, le réseau accuse plusieurs pertes hors profil. La figure 7.14 (b) nous présente le même

test mais le flot (A, C) est « pénalisé » par un délai de $30ms$ supplémentaire sur son RTT[12]. Dans ce cas, il n'y a plus de perte hors profil dans le réseau et nous pouvons voir qu'il n'y a plus d'oscillation également. Chaque flot atteint son débit assuré et le flot (A, C) possède toujours la meilleure occupation de bande passante. Les figures 7.14 illustrent que lorsque nous n'avons pas de perte de paquets hors profil dans un réseau, les flots TCP observent un équilibre et tous les flots TCP atteignent leur débit assuré si l'inégalité de l'équation (6.2) est satisfaite.



(a) (b)

Fig. 7.15 Débit TCP avec ou sans Penalty Shaper pour l'agrégat (a) (A,C) et (b) (B,D)

Les figures 7.15 (a) respectivement (b) donnent les débits TCP obtenus pour les agrégats (A, C) respectivement (B, D). Chaque agrégat est composé de 10 microflots et a un débit assuré de $4Mbits/s$. L'agrégat (A, C) a un $RTT = 30ms$ et l'agrégat (B, D) un $RTT = 100ms$. Nous utilisons, sur le routeur de bordure, le Penalty Shaper couplé avec un TBM. La valeur de la pénalité est de $10ms$ et la période d'observation est de $1sec$. A $t = 15sec$, l'agrégat (B, D) est émis durant $30sec$. Grâce au mécanisme du Penalty Shaper, nous pouvons voir sur ces figures que le débit instantané de TCP est moins fluctuant et proche du débit assuré. Le débit moyen TCP sans le Penalty Shaper est de $5.64Mbits/s$ pour l'agrégat (A, C) et de $2.82Mbits/s$ pour l'agrégat (B, D). Avec le Penalty Shaper, le débit moyen TCP est de $5.10Mbits/s$ pour l'agrégat (A, C) et de $4.13Mbits$ pour l'agrégat (B, D). Le débit assuré est atteint avec l'utilisation du Penalty Shaper.

7.4.4 Evaluation des performances du *Penalty Shaper*

Cette section présente les résultats obtenus sur la plateforme avec le Penalty Shaper. Nous évaluons les performances de ce lisseur de trafic lorsque les agrégats des clients TCP ont un nombre différent ou identique de microflots et un RTT identique ou différent. Dans tous les tests, la pénalité de délai est égale à $10ms$ et la période d'observation est égale à $1sec$.

[12]Nous n'utilisons pas encore le Penalty Shaper sur cet exemple, nous augmentons tout simplement le délai de notre flot.

Cela signifie que chaque seconde, l'algorithme donne une estimation du débit et augmente ou diminue la pénalité de $10ms$.

Impact de l'agressivité des agrégats dans un réseau surdimensionné

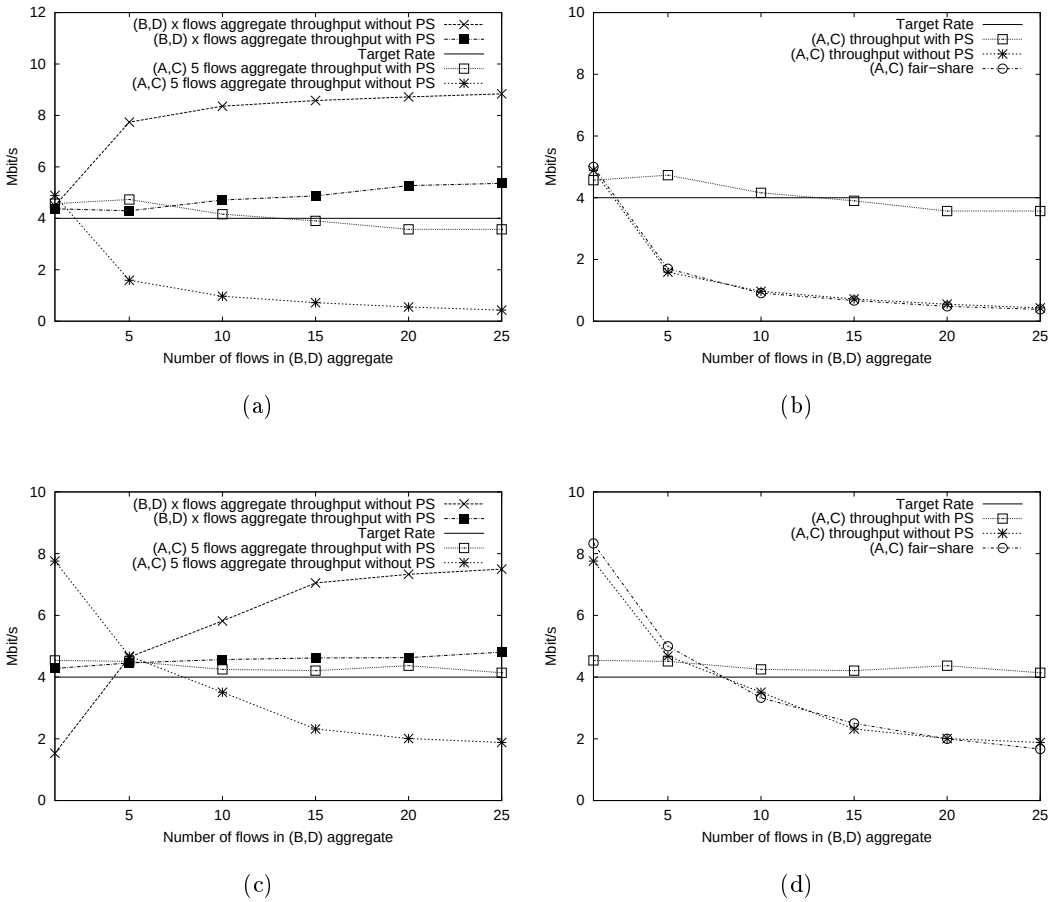


Fig. 7.16 Débit des agrégats TCP en fonction de l'agressivité des agrégats

L'atout principal de cette solution est : même si il y a un nombre différent de microflots entre les agrégats, le Penalty Shaper est capable d'atteindre le débit assuré. Les résultats sont présentés sur la figure 7.16. Dans ce test, deux agrégats sont en compétition et le RTT de chacun d'entre eux est de $30ms$. L'agrégat (A, C) a un nombre fixe de cinq microflots et l'agrégat (B, D) a un nombre de microflots variant de 1 à 25. Le débit assuré de ces deux agrégats est de $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. La capacité totale allouée pour le service assuré est $R_{AS} = 8Mbits/s$. Lorsque l'agrégat (B, D) a moins respectivement plus de cinq microflots, (A, C) est le plus respectivement moins agressif des agrégats. La ressource allouée au service assuré correspond à la capacité du goulot d'étranglement : $C_{AS} = 10Mbits/s$. Nous sommes dans le cas d'un réseau surdimensionné car nous avons

$2Mbits/s$ de bande passante en excès. La figure 7.16 (a) nous donne le débit obtenu par les deux agrégats. Grâce au Penalty Shaper, l'agrégat est capable d'atteindre son débit assuré. Pour des raisons de lisibilité, nous avons tracé sur la figure 7.16 (b) le débit obtenu par l'agrégat (A, C) à côté de la courbe du partage équitable. La figure 7.16 (b) montre qu'avec le TBM, le débit reste proche d'un partage équitable tandis qu'avec le Penalty Shaper, le débit assuré est atteint ou du moins, en est très approché.

Impact de la validité de l'assurance, en fonction de la taille de l'agrégat, dans un réseau surdimensionné

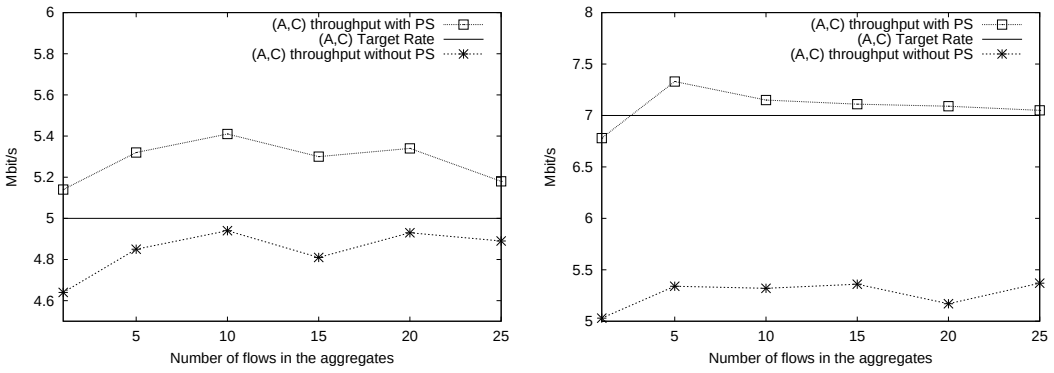
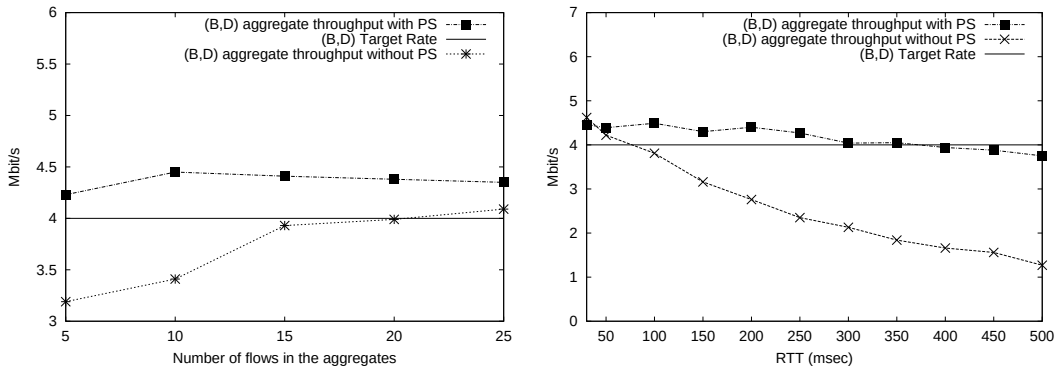


Fig. 7.17 Débit de l'agrégat TCP avec différents débits assurés

Dans cette partie, deux agrégats sont en compétition et leur RTT est égal à $30ms$. Ils ont chacun le même nombre de microflots et ce nombre varie de 1 à 25. La capacité totale allouée pour le service assuré est $R_{AS} = 8Mbits/s$ et la ressource allouée pour le service assuré est $C_{AS} = 10Mbits/s$. Le tableau 7.4 présente les résultats obtenus avec deux scénarios. Dans le premier, (A, C) et (B, D) ont respectivement un débit assuré de $r(A, C)_{AS} = 3Mbits/s$ et $r(B, D)_{AS} = 5Mbits/s$. Dans le second, ils ont respectivement un débit assuré de $r(A, C)_{AS} = 7Mbits/s$ et $r(B, D)_{AS} = 1Mbits/s$. Ainsi, les deux scénarios illustrent le cas où les agrégats ont des débits assurés proches ou distants dans le cas d'un réseau surdimensionné. Nous avons tracé sur la figure 7.17 (a) le débit obtenu par l'agrégat (A, C) lorsqu'il souhaite un débit assuré de $r(A, C)_{AS} = 7Mbits/s$ (cela correspond à la colonne 3 du tableau 7.4). La figure 7.17 (b) donne le débit obtenu par l'agrégat (B, D) lorsqu'il désire un débit assuré de $r(B, D)_{AS} = 5Mbits/s$ (cela correspond à la colonne 5 du tableau 7.4). Chaque agrégat atteint son débit assuré même si le nombre de microflots le composant augmente.

Test	# flots dans l'agrégat (A, C) versus (B, D)	A vers C RTT=30ms	B vers D RTT=30ms	A vers C RTT=30ms	B vers D RTT=30ms
TBM only	1 vs 1	4.64/5	4.29/3	5.03/7	4.36/1
TBM+PS	1 vs 1	5.14/5	3.61/3	6.78/7	1.61/1
TBM only	5 vs 5	4.85/5	4.56/3	5.34/7	4.05/1
TBM+PS	5 vs 5	5.32/5	3.39/3	7.33/7	1.70/1
TBM only	10 vs 10	4.95/5	4.38/3	5.32/7	4.26/1
TBM+PS	10 vs 10	5.41/5	3.47/3	7.15/7	1.64/1
TBM only	15 vs 15	4.81/5	4.48/3	5.36/7	4.23/1
TBM+PS	15 vs 15	5.30/5	3.57/3	7.11/7	1.94/1
TBM only	20 vs 20	4.93/5	4.41/3	5.17/7	3.86/1
TBM+PS	20 vs 20	5.34/5	3.41/3	7.09/7	1.92/1
TBM only	25 vs 25	4.89/5	4.38/3	5.37/7	4.37/1
TBM+PS	25 vs 25	5.18/5	3.69/3	7.05/7	1.92/1

Tab. 7.4 Réseau surdimensionné (légende : débit de transfert en $Mbits/s$ / débit assuré en $Mbits/s$)



(a) (b)

Fig. 7.18 Débit TCP en fonction RTT

Impact du RTT dans un réseau surdimensionné

Même si l'écart du nombre de flots entre les agrégats est important et même si ils ont une grande différence de RTT, le Penalty Shaper doit être capable d'atteindre le débit assuré par agrégat. Les figures 7.18 (a) et (b) illustrent ce cas. La figure 7.18 (a) montre le débit obtenu par l'agrégat (B, D) avec un $RTT = 100ms$ en compétition avec un agrégat (A, C) avec un $RTT = 30ms$ en fonction du nombre de microflots. Nous nous intéresserons aux résultats de l'agrégat (B, D) uniquement car dans notre scénario, l'agrégat (A, C) obtient toujours son débit assuré avec ou sans Penalty Shaper. Le débit assuré pour (A, C) et (B, D) est $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. Grâce au Penalty Shaper, l'agrégat (B, D) atteint son débit assuré et l'augmentation du nombre de microflots dans l'agrégat n'a aucune influence sur le débit assuré désiré. Enfin, la figure 7.18 (b) montre le débit d'un agrégat (B, D) de 10 microflots en compétition avec un autre agrégat (A, C) de 10 microflots également. Le

Test	# flots dans l'agrégat (A, C) versus (B, D)	A vers C RTT=30ms	B vers D RTT=30ms	A vers C RTT=30ms	B vers D RTT=30ms
TBM only	1 vs 1	4.46/8	4.50/4	4.94/10	4.31/2
TBM+PS	1 vs 1	4.69/8	4.33/4	6.40/10	2.31/2
TBM only	5 vs 5	5.02/8	4.34/4	5.25/10	4.05/2
TBM+PS	5 vs 5	5.08/8	4.21/4	7.01/10	2.09/2
TBM only	10 vs 10	5.08/8	4.25/4	5.24/10	4.06/2
TBM+PS	10 vs 10	5.38/8	3.98/4	7.20/10	2.07/2
TBM only	15 vs 15	5.02/8	4.31/4	5.27/10	4.08/2
TBM+PS	15 vs 15	5.53/8	3.74/4	7.32/10	2.05/2
TBM only	20 vs 20	5.05/8	4.36/4	5.43/10	3.84/2
TBM+PS	20 vs 20	5.51/8	3.73/4	7.29/10	2.09/2
TBM only	25 vs 25	4.90/8	4.35/4	5.29/10	3.95/2
TBM+PS	25 vs 25	5.49/8	3.79/4	7.34/10	2.04/2

Tab. 7.5 Réseau sous-dimensionné (légende : débit de transfert en $Mbits/s$ / débit assuré en $Mbits/s$)

RTT de l'agrégat (A, C) est de $30ms$ et varie de $30ms$ à $500ms$ pour l'agrégat (B, D). Il apparaît que l'agrégat (B, D) atteint son débit assuré lorsque celui-ci reste faisable (c'est-à-dire lorsque $r(B, D)_{AS} > Wmax/RTT$). A titre d'information, nous avons fait ce test avec un nombre de flots allant de 5 contre 5 jusqu'à 25 contre 25 et obtenu des résultats similaires. Nous ne présentons ici que la mesure de 10 contre 10 flots.

Impact du nombre de flots dans un réseau sous-dimensionné

Sur les deux scénarios du tableau 7.5, la capacité totale allouée au service assuré est $R_{AS} = 12Mbits/s$. Il n'y a aucune bande passante en excès. Nous sommes donc dans le cas d'un réseau sous-dimensionné. Le réseau accuse par conséquent plusieurs pertes de paquets en profil. Les mesures du tableau 7.5 montrent que l'utilisation du Penalty Shaper permet d'obtenir un débit TCP plus proportionnel au débit assuré qu'avec le TBM seul. Nous ne nous attarderons pas sur ce cas parce que : (1) l'algorithme du Penalty Shaper part de l'hypothèse que le réseau est correctement dimensionné et que la perte d'un paquet en profil est proche de zéro ; (2) comme dans le cadre du Dynamic Shaper, nous considérons que ce cas correspond à un mauvais dimensionnement du réseau.

7.4.5 Conclusion pour le Penalty Shaper

Contrairement au Dynamic Shaper, le Penalty Shaper résout le problème d'agressivité entre les agrégats TCP. Quelquesoit la composition de l'agrégat, son RTT ou son débit assuré, le Penalty Shaper permet d'obtenir le débit assuré. Reste que la pénalité appliquée est encore définie de façon arbitraire et qu'une progression linéaire n'est pas forcément compatible avec la dynamique de TCP. En vue d'améliorer le calcul de la pénalité, nous proposons d'ajouter au PS un algorithme AIMD[13] pour mieux calculer cette pénalité afin qu'il travaille avec la même dynamique que TCP.

[13] Additive Increase Multiplicative Decrease

i	pénalité en ms	détail du calcul
1	10	
2	30	10+2*10
4	70	30+4*10
4	50	70-2*10
1	45	50-0.5*10
1	40	45-0.5*10
1	50	40+1*10
2	40	50-1*10
1	50	40+1*10
2	40	50-1*10
1	50	40+1*10
2	40	50-1*10

Tab. 7.6 Exemple de calcul de pénalité avec l'APS pour des pertes jusqu'à $45ms$

7.5 AIMD *PENALTY SHAPER* (APS)

Le principe du lisseur de trafic APS est en tout point identique au PS, sauf pour le calcul de la pénalité. En effet, la pénalité n'est plus linéaire mais suit un algorithme de type AIMD. L'algorithme utilisé pour le calcul est donné en figure 7.19. Ce choix permet d'être en conformité avec le contrôle de congestion de TCP. Les algorithmes de type AIMD surpassent les algorithmes linéaires en termes d'efficacité [FHPW00]. Afin de mieux illustrer le fonctionnement de l'algorithme, le tableau 7.6 donne un exemple de calcul de pénalité en partant des hypothèses suivantes : (1) le flot n'accuse aucune perte lorsque sa pénalité est supérieure à $45ms$; (2) le flot est toujours au-dessus de son débit désiré. Enfin, la figure 7.20 donne la représentation graphique du tableau.

```
K = 10ms
i = 1
FOR each observation period T
 TSW gives an evaluation of the throughput : throughput_measured
 IF throughput_measured < target_rate OR there are no out-profile losses
 THEN reduce the penalty delay
      current_penalty = current_penalty - ((i/2) * K)
      i = 1
 ELSE
     raise the penalty delay
     current_penalty = current_penalty + (i * K)
     i = i * 2;
 ENDIF
ENDFOR
```

Fig. 7.19 L'algorithme APS

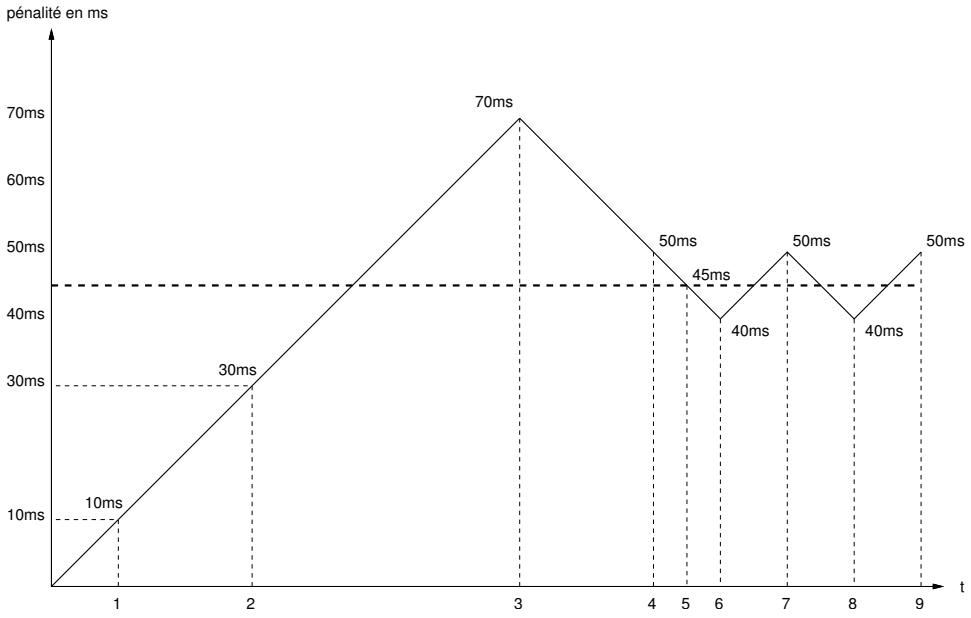


Fig. 7.20 Courbe de la pénalité obtenue avec l'algorithme APS

7.5.1 Evaluation des performances de l'AIMD *Penalty Shaper*

Cette section présente les résultats obtenus avec le lisseur de trafic APS. Les hypothèses de départ, les conditions de génération de trafic et les scénarios sont en tous points similaires à ceux présentés en section 7.4.4 pour le Penalty Shaper.

Impact de l'agressivité des agrégats dans un réseau surdimensionné

Les résultats obtenus sont similaires à ceux obtenus avec le PS. Les scénarios présentés sur les figures 7.21 (a) (b) (c) (d) sont identiques à ceux réalisés dans le cadre du PS ou du DS. Nous avons complété ces mesures par l'ajout de tests mettant en concurrence un flot dans (A, C) contre un nombre variable de flots dans (B, D) avec différents débits assurés. Les résultats de ces tests avec les valeurs minimum et maximum des mesures sont représentés sur les figures 7.22 (a) (b) (c) (d).

Impact du RTT dans un réseau surdimensionné

Dans ce scénario, le débit assuré pour (A, C) et (B, D) est $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. La figure 7.23(a) nous montre le débit obtenu par deux agrégats (A, C) et (B, D) de 10 microflots. L'agrégat (A, C) a un RTT de $30ms$ et le RTT de l'agrégat (B, D) varie de $30ms$ à $500ms$. Contrairement au PS dont le résultat de ce test se trouve figure 7.23 (b), l'APS obtient une meilleure résistance au facteur du RTT. La figure 7.23 (a) montre que l'APS conserve le débit souhaité au-delà des $400ms$. Afin de compléter cette étude, nous avons mesuré sur la figure 7.23 (b) l'influence de la charge de l'agrégat en fonction du RTT. Sur cette figure, l'agrégat (A, C) a un RTT de $30ms$ et un nombre de microflots

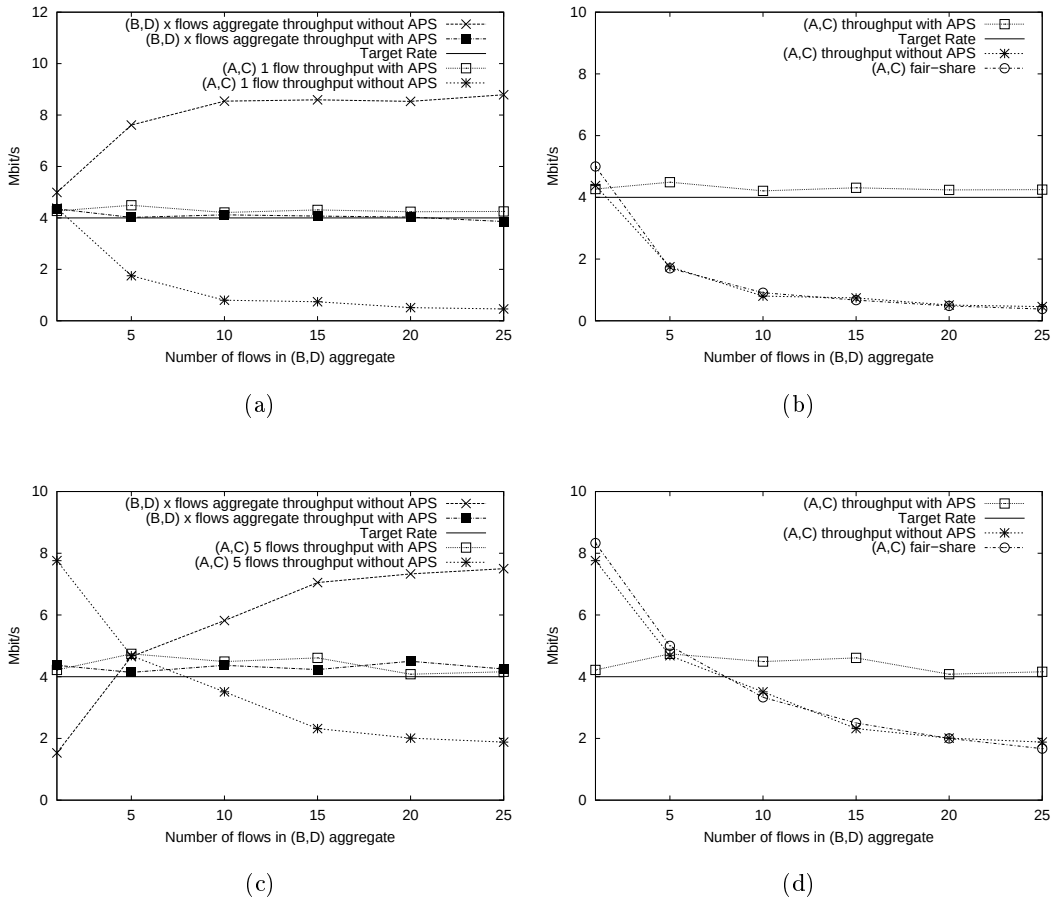


Fig. 7.21 Débit des agrégats TCP en fonction de l'agressivité des agrégats

variant de 10 à 25 (représentés sur l'axe des y à droite de la figure). Pour l'agrégat (B, D), son RTT est fixé à $100ms$ et son nombre de microflots est fixé à 5. Dans ces conditions, l'agrégat (A, C), quelquesoit son nombre de microflots, surpasse toujours son débit assuré. En revanche, l'agrégat (B, D) se trouve dans les pires conditions pour le réaliser. La figure 7.23 (b) nous donne le débit moyen instantané du flot (B, D) avec et sans l'utilisation de l'APS en fonction du nombre de microflots dans l'agrégat (A, C). Nous pouvons remarquer qu'avec l'APS, (B, D) réalise son débit assuré alors que cela lui est impossible avec le TBM seul.

Comparaison de l'efficacité de l'APS avec le PS et le DS

Ce test cherche à comparer l'efficacité de l'APS par rapport aux autres conditionneurs présentés. Dans ce test, deux agrégats de dix microflots sont émis de deux sources distinctes. Pour le flot (A, C) : $r(A, C)_{AS} = 3Mbits/s$, RTT=$30ms$; pour (B, D) : $r(B, D)_{AS} = 6Mbits/s$, RTT=$100ms$. La capacité totale allouée pour le service assuré est $R_{AS} = 9Mbits/s$. Il y a donc très peu de bande passante en excès (environ $0.5Mbits$). Nous

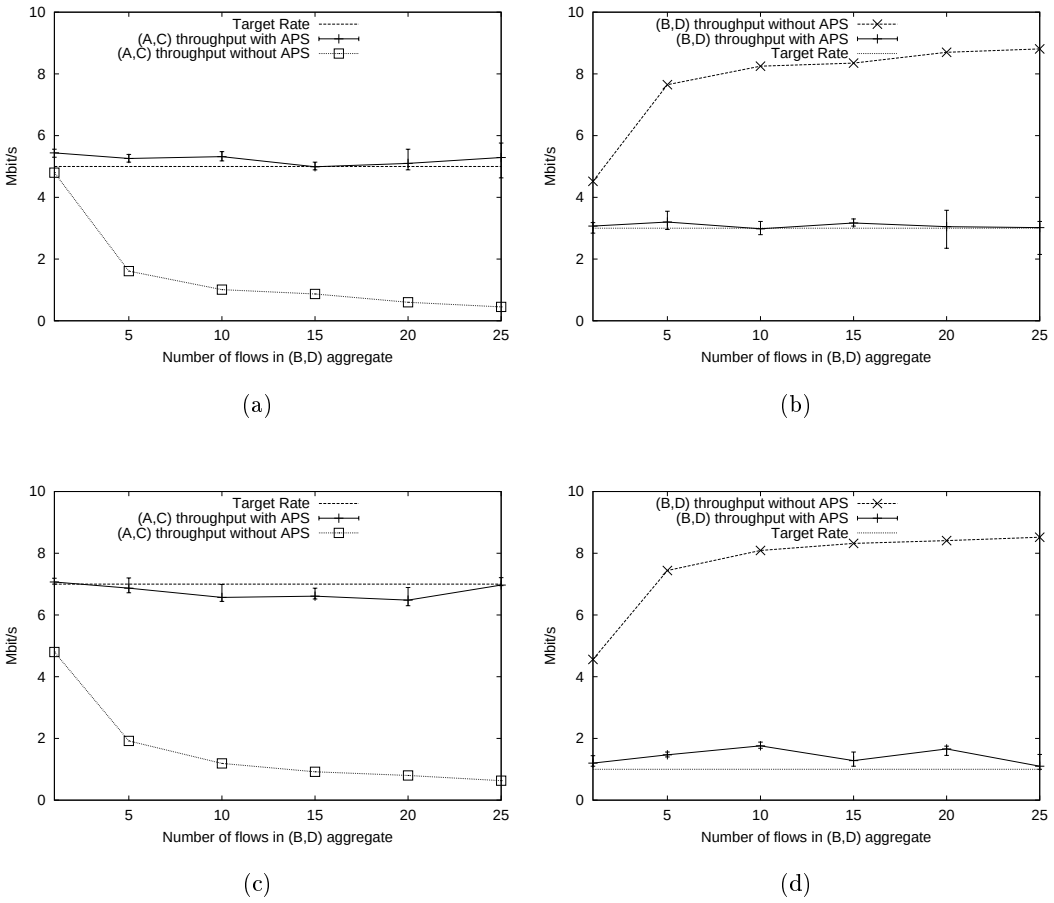
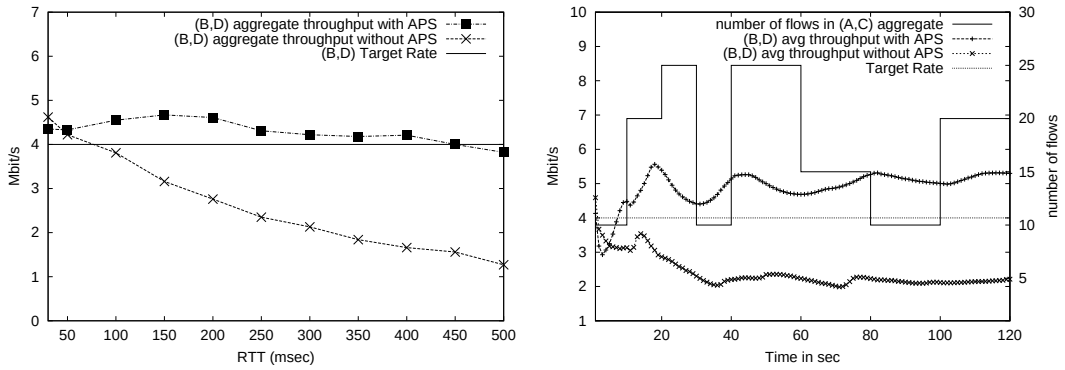


Fig. 7.22 Débit des agrégats TCP en fonction de l'agressivité des agrégats avec différents débits assurés

sommes proches de la condition extrême et (B, D) se trouve dans le pire des cas pour réaliser son débit assuré (long RTT et haut débit assuré). La figure 7.24 compare les résultats obtenus : il est donné la valeur min et max des mesures effectuées, les points centraux reliés par une ligne représentant la valeur moyenne. Sans aucun mécanisme de QoS, le flot (A, C) surpasse son débit assuré au détriment de (B, D). Il est remarquable que grâce à la prise en compte de la perte de paquets hors profil dans le réseau et à l'application d'une pénalité de type AIMD suivant la dynamique TCP, l'APS se trouve au plus proche du débit assuré et permet à chacun des deux agrégats d'obtenir la part de bande passante requise.

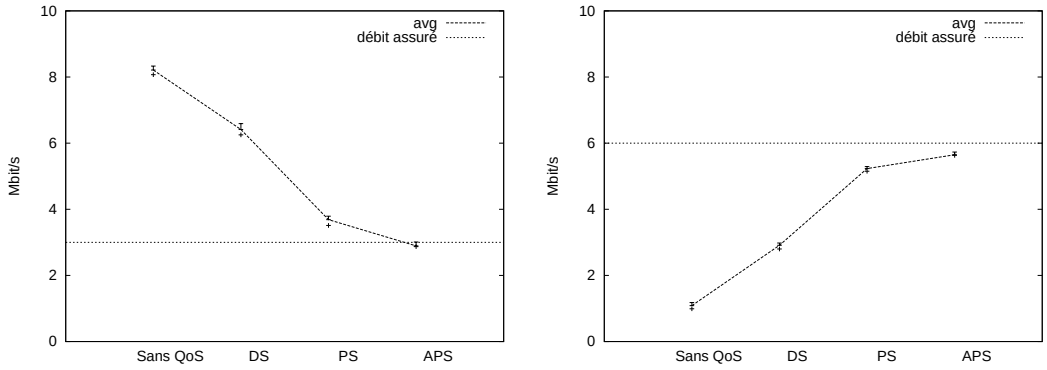
Test général

Dans ce dernier test, nous mettons en compétition 4 agrégats avec différents RTT, débits assurés et nombre de flots. Le scénario est donné par la figure 7.25 (a). Nous nous plaçons dans le pire des cas en essayant de garantir un débit assuré plus grand aux flots ayant un fort RTT que ceux ayant un faible RTT. Le tableau de résultats sur la 7.25(b) nous montre que chacun des agrégats respecte son débit assuré avec l'APS.



(a) (b)

Fig. 7.23 Débit TCP en fonction du RTT

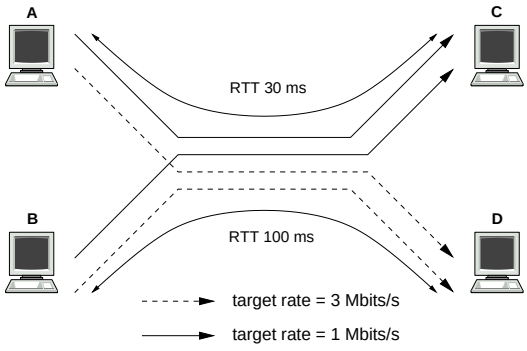


(a) Agrégat (A, C) (b) Agrégat (B, D)

Fig. 7.24 Test de comparaison entre TBM, DS, PS et APS

7.5.2 Conclusion pour APS

L'APS propose une amélioration de l'application de la pénalité. De plus, cette pénalité, même si elle est donnée au départ, est automatiquement calculée en fonction de l'état du réseau et relâchée si les ressources du réseau deviennent à nouveau suffisantes. La mise en œuvre de cet algorithme reste simple et a pu être effectuée sur des machines tournant sous FreeBSD. Les coefficients utilisés dans l'algorithme AIMD ne sont peut-être pas optimaux mais ils fournissent de bonnes performances de qualité. D'autres coefficients sont à tester afin d'obtenir le meilleur résultat possible.



(a)

agregate's name	# flows	without APS	with APS
AC	5	2.12 / 1	1.42 / 1
AD	5	2.54 / 3	3.24 / 3
BC	5	2.04 / 1	1.04 / 1
BD	5	2.64 / 3	3.04 / 3
AC	10	2.19 / 1	1.24 / 1
AD	10	2.45 / 3	3.15 / 3
BC	10	2.27 / 1	1.01 / 1
BD	10	2.45 / 3	3.13 / 3

(caption : goodput in Mbits/s / target rate in Mbits/s)

(b)

Fig. 7.25 Test avec plusieurs agrégats différents

7.6 CONCLUSION DU CHAPITRE

Dans ce chapitre, nous avons proposé un nouveau conditionneur pour le service assuré permettant de garantir un débit de transfert à un agrégat de flots TCP. Cette garantie est indépendante du RTT, du nombre de microflots et du débit assuré de l'agrégat. De plus, ce conditionneur résout le problème d'inéquité existant entre agrégats à nombre de microflots différents (ce que nous avons défini comme l'agressivité entre les agrégats). Une méthode de conditionnement simple et facile à mettre en œuvre a également été présentée. Cette méthode est étroitement liée au conditionneur proposé car elle lui permet de s'affranchir de calculs difficiles à réaliser comme l'estimation du RTT ou l'estimation des pertes TCP. Enfin, au contraire du DS et du PS, l'APS est totalement autoconfigurable.

Chapitre 8

Conclusion

Ce chapitre apporte des conclusions sur ce travail de thèse et donne les perspectives des travaux.

8.1 RÉSUMÉ

L'objectif de cette thèse est de proposer un mécanisme de conditionnement adapté aux flots à caractères élastiques pour la classe assurée de l'architecture DiffServ. Une étude du modèle DiffServ et des mécanismes utilisés afin de mettre en œuvre une telle architecture a été effectuée aux chapitres 2 et 3. Une réalisation et des mesures d'une architecture DiffServ ont été présentées aux chapitres 4 et 5. Après avoir souligné l'incapacité du conditionnement proposé dans cette architecture à donner des garanties de débits aux flots TCP, nous avons fait une étude exhaustive des mécanismes permettant d'apporter une solution concrète à ce problème au chapitre 6. L'étude des différentes propositions et leur classification ont mis en relief leur complexité et pour certaines d'entre elles, leur difficulté à passer à l'échelle. Les récentes propositions de conditionnement de trafic ne cherchent plus à résoudre directement le problème de la garantie de débit d'un flot TCP. De récentes études s'attachent à rendre la classe assurée plus équitable [PC04]. Etant donné que le comportement des routeurs de cœur doit rester le plus simple possible (notamment pour des critères de performance), nous avons axé nos efforts sur la définition d'un nouveau mécanisme de conditionnement à l'entrée d'un réseau DiffServ. Dans le souci de s'affranchir des méthodes complexes de calcul nécessaires à la mise en œuvre des propositions de marquage adaptatif, nous avons développé un lisseur de trafic baptisé AIMD Penalty Shaper. La philosophie du choix de ce mécanisme de lissage de trafic et une évaluation sur une plateforme réelle de son fonctionnement sont données au chapitre 7. L'originalité consiste à utiliser le flux TCP lui-même comme indicateur de marquage des paquets émis. Les résultats montrent qu'avec un

conditionnement simple à mettre en œuvre[1], nous pouvons garantir un débit aux agrégats TCP. Cette garantie reste vraie quelquesoit le nombre d'agrégats en compétition au niveau du goulot d'étranglement du réseau, le nombre de microflots à l'intérieur de ces agrégats, leur débit assuré et leur RTT.

8.2 PERSPECTIVES

Il n'existait pas encore de solution facilement déployable pour garantir un débit TCP efficacement. Nous avons réussi avec l'APS à développer un mécanisme opérationnel satisfaisant l'objectif de garantie de débit TCP fixé. Nous pouvons considérer maintenant que les perspectives directement liées à l'APS sortent du cadre de la recherche et concernent l'ingénierie informatique. Néanmoins, ce mémoire ouvre les pistes suivantes :

Validation expérimentale

L'évaluation faite au chapitre 7 de l'APS n'est qu'une première étape quant à la validation du mécanisme. Des mesures plus ambitieuses seront prochainement effectuées dans un contexte réel[2]. Si le mécanisme répond aux critères de qualité de service définis, nous aurons alors la possibilité de réaliser un développement industriel de ce lisseur de trafic.

Plan de contrôle

Dans ce mémoire, nous nous sommes uniquement intéressés au plan des données. Néanmoins, afin de compléter ce mécanisme de gestion de cette qualité de service, il est nécessaire d'introduire un système de définition de la QoS au niveau applicatif. Plusieurs standards et modèles formels destinés à définir un cadre global de la QoS ont été définis. Notamment, il existe un langage de spécification de la QoS basé sur XML et nommé XQoS (XML-based Quality of Service)[NW01] intégrant à la fois les points de vue de l'utilisateur et du fournisseur du service de communication. Les spécifications des sessions et de types de médias XQoS ont pour but de décrire les besoins à satisfaire par le système de communication. Les spécifications des ressources et des services quant à elles ont pour but de fournir les informations nécessaires afin de découvrir, transférer, déployer et configurer les mécanismes de QoS appropriés. Une perspective intéressante serait d'apporter les éléments pour coupler la définition de la QoS offerte par l'APS à un tel système de gestion.

Réseaux filaires et réseaux sans fil

Les travaux de cette thèse s'inscrivent dans un contexte de réseaux filaires et c'est pour ce type de réseaux qu'APS a été conçu. Il existe à l'IETF un groupe de travail sur les réseaux

[1] En effet, le conditionneur se met en coupure dans le réseau sans entraîner de modifications dans les autres équipements.

[2] Une campagne d'évaluation est prévue durant la deuxième quinzaine de novembre 2004 à l'Université de la Réunion. Cette université possède un goulot d'étranglement *de facto* en sortie vers l'Europe. Nous utiliserons notre mécanisme pour évaluer sa résistance dans un contexte d'utilisation réelle.

ad-hoc appelé MANET[3]. En ce qui concerne les architectures de différenciation de service dans les réseaux sans fil, tout reste encore à faire. La performance rencontrée au sein de ces réseaux est chaotique. Il existe des travaux de réservation de ressources mais il semble pertinent de partir sur la base de services de qualité faible comme le service LBE[4]. Or, ce type de service n'est pas offert par l'APS. Néanmoins, les mécanismes présentés au chapitre 2 peuvent être utilisés, par exemple, pour limiter le trafic dans un réseau sans fil. A ce titre, nous avions effectué des mesures concernant l'économie d'énergie des hôtes mobiles d'un réseau ad-hoc [LFMF03] en vue d'utiliser un mécanisme de limitation du débit pour gérer la puissance d'émission des stations. Ce travail est en cours de réalisation.

[3]Mobile Ad-hoc Networks

[4]Less Than Best-Effort service : `http://www.cnaf.infn.it/~ferrari/tfngn/`

Annexe A

Exemple de fichier de configuration du routeur de cœur

```
#limitation de debit en Ko/s
rate 100
#taille de paquet moyenne
pktsize 750
#classe GS
GS 3000 3000
#classe BE : 50 Ko max en file d'attente, 50% du débit
BE 50000 50000 50
#classe AS : 30 Ko max, threshold PBS à 15 Ko,
#DSCP 40 et 48 (in et out resp.), 50% du débit
AS 30000 30000 50 40 48 15000 1.0
```

Cette configuration une fois stockée dans un fichier nommé, par exemple : `bboned.conf`, est activée pour l'interface `tx0` par la commande suivante :

```
bboned tx0 on bboned.conf
```

Exemple de fichier de configuration du routeur de bordure

Les lignes suivantes correspondent à l'impression d'un fichier de configuration ainsi qu'à l'affichage résultant lors de l'exécution du programme `edged`.

```
maxspeed 10Mb
#----------------------------------------
newtc monflot1
monflot1 filter janus typhon 0 1000
monflot1 method shape 1Mb 3K
monflot1 class 5
#----------------------------------------
newtc ESSAI_MARQUAGE
ESSAI_MARQUAGE filter * * 0 1000000
ESSAI_MARQUAGE method mark 2Mb 5K
ESSAI_MARQUAGE class 1
```

Les lignes ci-dessous correspondent à l'affichage résultant de l'exécution de `edged` avec ce fichier :

```
Récupération configuration des flux OK.

INTERFACE=de0, Vitesse maxi=1250000 (oct/s), Nombre de flux configures=2
REGLE: monflot1
adresse source: 3ffe:0304:0105:003d:0280:c8ff:fe46:293a
adresse destin: 3ffe:0304:0105:003d:0280:c8ff:fe46:31df
flot mini : 0
flot maxi : 1000
Classe : 5
Methode SHAPE TKr 125000 octets/s, TKb 3000 octets
------------------------------------------------------------
REGLE: ESSAI_MARQUAGE
adresse source: 0000:0000:0000:0000:0000:0000:0000:0000
adresse destin: 0000:0000:0000:0000:0000:0000:0000:0000
flot mini : 0
flot maxi : 1000000
Classe : 1
Methode MARK TKr 250000 octets/s, TKb 5000 octets
------------------------------------------------------------
Attachement de l'interface de0 en cours...
```

Annexe B

Modèle de TCP

Ce modele a été présenté par [PFTK98].

$$TR(p, RTT, To, W_{max}) = \begin{cases} MSS\frac{\frac{1-p}{p}+W(p)+\frac{Q(p,W(p))}{1-p}}{RTT(\frac{b}{2}W(p)+1)+\frac{Q(p,W(p))F(p)To}{1-p}} & \text{if } W(p) < W_{max} \\ MSS\frac{\frac{1-p}{p}+W_{max}+\frac{Q(p,W_{max})}{1-p}}{RTT(\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2)+\frac{Q(p,W_{max})F(p)To}{1-p}} & \text{otherwise} \end{cases} \quad (1)$$

avec

$$W(p) = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \frac{2+b}{3b}^2} \quad (2)$$

$$Q(p,w) = min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3})}{1-(1-p)^w}\right) \quad (3)$$

$$F(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \quad (4)$$

En posant $To = 5 * RTT$ suivant [FB00] on obtient :

$$RTT(p, TR, W_{max}) = \begin{cases} MSS\frac{\frac{1-p}{p}+W(p)+\frac{Q(p,W(p))}{1-p}}{TR\left((\frac{b}{2}W(p)+1)+5\frac{Q(p,W(p))F(p)}{1-p}\right)} & \text{if } W(p) < W_{max} \\ MSS\frac{\frac{1-p}{p}+W_{max}+\frac{Q(p,W_{max})}{1-p}}{TR\left((\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2)+5\frac{Q(p,W_{max})F(p)}{1-p}\right)} & \text{otherwise} \end{cases} \quad (5)$$

Publications

[1] AIMD Penalty Shaper to Enforce Assured Service for TCP Flows, Emmanuel Lochin, Pascal Anelli, Serge Fdida, *4th IEEE International Conference on Networking (ICN'2005) - La Réunion, France - April, 2005.*

[2] Penalty Shaper to Enforce Assured Service for TCP Flows, Emmanuel Lochin, Pascal Anelli, Serge Fdida, *Soumis à Networking 2005.*

[3] A Simple TCP Flows Conditioning for Assured Services, Emmanuel Lochin, Pascal Anelli, Serge Fdida, *Soumis à ICC 2005.*

[4] Evaluation de la différenciation de services dans l'Internet, Emmanuel Lochin, Pascal Anelli, Serge Fdida, Fabien Garcia, Guillaume Auriol, Christophe Chassot, André Lozes, *Revue des Technique et Science Informatiques, TSI numéro spécial "Réseaux", éditions Hermes, Mai 2004.*

[5] Methodes de garantie de débit d'un flot TCP dans un réseau à service assuré, Lochin Emmanuel, Anelli Pascal and Fdida Serge, *10ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2003), Paris (France).*

[6] Energy consumption models for ad-hoc mobile terminals, Lochin Emmanuel, Fladenmuller Anne, Moulin Jean-Yves and Fdida Serge, *The 2nd Mediterranean Workshop on Ad-Hoc Networks - Med-Hoc Net 2003 - IFIP-TC6-WG6.8 - Mahdia, Tunisia.*

[7] Performance Analysis for an IP Differentiated Services Network, Chassot Christophe, Garcia Fabien, Auriol Guillaume, Lozes André, Lochin Emmanuel and Anelli Pascal, *ICC 2002 – New York - May, 2002.*

[8] Conception, implémentation et mesure des performances d'une architecture de communication à QdS garantie dans un domaine IPv6 à services différenciés, Garcia Fabien, Auriol Guillaume, Chassot Christophe, Lozes André, Lochin Emmanuel and Anelli Pascal, *9ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2002), 27-30 Mai, Montréal (Canada).*

[9] IPv6, théorie et pratique Cizault Gisèle, *O'Reilly Publishers, troisième édition, Mars 2002.*

[10] Conception, implementation and evaluation of a QoS - based architecture for an IP environment supporting differential services, Garcia Fabien, Chassot Christophe, Lozes André, Diaz Michel, Anelli Pascal and Lochin Emmanuel, *IDMS, England - June, 2001*

[11] QoS sous Linux, Lochin Emmanuel, *Rapport technique LIP6 - Paris - Septembre, 2000.*

Bibliographie

[AQU] Adaptive resource control for qos using an ip-based layered architecture. http://www-st.inf.tu-dresden.de/aquila/.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang et W. Weiss. *An Architecture for Differentiated Services.* Request For Comments 2475, IETF, décembre 1998.

[BBC+01] Jon C. R. Bennett, Kent Benson, Anna Charny, William F. Courtney et Jean-Yves Le Boudec. Delay jitter bounds and packet scale rate guarantee for expedited forwarding. *In : Proc. of IEEE INFOCOM*, pp. 1502–1509.

[BCC+98] Bob Braden, David D. Clark, Jon Crowcroft, Bruce Davie, Steve Deering, Deborah Estrin, Sally Floyd, Van Jacobson, Greg Minshall, Craig Partridge, Larry Peterson, K. K. Ramakrishnan, Scott Shenker, John Wroclawski et Lixia Zhang. *Recommendations on Queue Management and Congestion Avoidance in the Internet.* Request For Comments 2309, IETF, avril 1998.

[BCS94] Robert Braden, David D. Clark et Scott Shenker. *Integrated Services in the Internet Architecture: an Overview.* Request For Comments 1633, IETF, juin 1994.

[BdC00] O. Bonaventure et S. de Cnodder. *A Rate Adaptive Shaper for Differentiated Services.* Request For Comments 2963, IETF, octobre 2000.

[BMB00] Thomas Bonald, Martin May et Jean-Chrysostome Bolot. Analytic evaluation of RED performance. *In : Proc. of IEEE INFOCOM*, pp. 1415–1424. Tel-Aviv, Israël, mars 2000.

[BZ96a] Jon C. R. Bennett et Hui Zhang. WF2Q: Worst-case fair weighted fair queueing. *In : Proc. of IEEE INFOCOM*, pp. 120–128.

[BZ96b] Jon C.R. Bennett et Hui Zhang. Hierarchical packet fair queueing algorithms. *In : Proc. of ACM SIGCOMM*, pp. 143–156. Aussi dans *IEEE/ACM Transactions on Networking* oct 97. v5n5.

[BZ97] Jon C. R. Bennett et Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, vol. 5, n5, 1997, pp. 675–689.

[CAD] Creation and deployment of end-user services in premium ip networks. http://www.cadenus.org/.

[CF98] D. Clark et W. Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking*, vol. 6, n4, août 1998, pp. 362–373.

[CHM+02] Y. Chait, C. Hollot, V. Misra, D. Towsley et H. Zhang. Providing throughput differentiation for TCP flows using adaptive two color marking and multi-level aqm. *In : Proc. of IEEE INFOCOM*. New York, juin 2002.

[Cho99] Kenjiro Cho. Managing traffic with ALTQ. *Proceedings of USENIX Annual Technical Conference: FREENIX Track*, juin 1999, pp. 121–128.

[CLA02] N. Christin, J. Liebeherr et T. Abdelzaher. A quantitative assured forwarding service. *In : Proc. of IEEE INFOCOM*, pp. 864–873. New York, NY, juin 2002.

[CSZ92] David D. Clark, Scott Shenker et Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *In : Proc. of ACM SIGCOMM*, pp. 14–26.

[DCB+02] B. Davie, A. Charny, J.C.R. Bennett, K. Benson et al. *An Expedited Forwarding PHB (Per-Hop Behavior)*. Request For Comments 3246, IETF, mars 2002.

[dR99] José Ferreira de Rezende. Assured service evaluation. *In : Proc. of IEEE GLOBECOM*, pp. 100–104. Rio de Janeiro, Brasil, décembre 1999.

[DR00] C. Dovrolis et P. Ramanathan. Proportional differentiated services, part ii: Loss rate differentiation and packet dropping. *In : Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS*. Pittsburgh, PA, juin 2000.

[EGS02] M.A. El-Gendy et K.G. Shin. Assured forwarding fairness using equation-based packet marking and packet separation. *Computer Networks*, vol. 41, n 4, 2002, pp. 435–450.

[FB00] Victor Firoiu et Marty Borden. A study of active queue management for congestion control. *In : Proc. of IEEE INFOCOM*, pp. 1435–1444.

[FC00] T. Ferrari et P. Chimento. A measurement-based analysis of expedited forwarding phb mechanisms, juin 2000.

[Fer00] Tiziana Ferrari. End-to-end performance analysis with traffic aggregation. *Computer Networks*, vol. 34, n6, décembre 2000, pp. 905–914.

[FF95] K. Fall et S. Floyd. Comparison of tahoe, reno and SACK TCP. *ACM Computer Communications Review*, vol. 26, n3, juillet 1995.

[FF99] Sally Floyd et Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, vol. 7, n4, 1999, pp. 458–472.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye et Jorg Widmer. Equation-based congestion control for unicast applications. *In : Proc. of ACM SIGCOMM*, pp. 43–56. Stockholm, Sweden, août 2000.

[FJ93] Sally Floyd et Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, vol. 1, n4, août 1993, pp. 397–413.

[FJ95] Sally Floyd et Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, vol. 3, n4, août 1995, pp. 365–386.

[FKSS97] W. Feng, Dilip Kandlur, Debanjan Saha et Kang S. Shin. *Adaptive Packet Marking for Providing Differentiated Services in the Internet.* Rapport technique CSE-TR-347-97, IBM, octobre 1997.

[FRK00] Azeem Feroz, Amit Rao et Shivkumar Kalyanaraman. A TCP-friendly traffic marker for IP differentiated services. *In : Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS.*

[FSA00] W. Fang, N. Seddigh et AL. *A Time Sliding Window Three Colour Marker.* Request For Comments 2859, IETF, juin 2000.

[GAC+02] Fabien Garcia, Guillaume Auriol, Christophe Chassot, Andre Lozes, Emmanuel Lochin et Pascal Anelli. Conception, implémentation et mesure des performances d'une architecture de communication à qds garantie dans un domaine ipv6 à services différenciés. *In : 9ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2002)*, pp. 381–394. Montréal (Canada), mai 2002.

[GCL+01] Fabien Garcia, Christophe Chassot, André Lozes, Michel Diaz, Pascal Anelli et Emmanuel Lochin. Conception, implementation and evaluation of a qos - based architecture for an ip environment supporting differential services. *In : IDMS*, pp. 86–98.

[GDJL99] M. Goyal, A. Durresi, R. Jain et C. Liu. Effect of number of drop precedences in assured forwarding. *In : Proc. of IEEE GLOBECOM*, pp. 188–193.

[GEA] GÉant : The pan-european gigabit research network. http://www.dante.net/geant/.

[GLN+99] Roch Guerin, L. Li, Stephen J. Nadas, P. Pan, Vinod G. et J. Peris. The cost of qos support in edge devices: An experimental study. *In : Proc. of IEEE INFOCOM*, pp. 873–882.

[Gol94] S. Jamaloddin Golestani. A self-clocked fair queueing scheme for broadband applications. *In : Proc. of IEEE INFOCOM*, pp. 636–646.

[GVC96] Pawan Goyal, Harrick M. Vin et Haichen Cheng. Start-time Fair Queueing: A scheduling algorithm for integrated services packet switching networks. *In :*

Proc. of ACM SIGCOMM, pp. 157–168. Aussi dans *IEEE/ACM Transactions on Networking* oct 97 v5n5.

[HBF02] Ahsan Habib, Bharat Bhargava et Sonia Fahmy. A round trip time and time-out aware traffic conditioner for differentiated services networks. *In : Proc. of the IEEE International Conference on Communications - ICC*. New-York, USA, avril 2002.

[HBWW99] Juha Heinanen, Fred Baker, Walter Weiss et John Wroclawski. *Assured Forwarding PHB Group*. Request For Comments 2597, IETF, juin 1999.

[HG90] G. Hebuterne et A. Gravey. A space priority queueing mechanism for multiplexing atm channels. *Computer Networks and ISDN Systems*, décembre 1990, pp. 37–43.

[HG99a] J. Heinanen et R. Guerin. *A Single Rate Three Color Marker*. Request For Comments 2697, IETF, septembre 1999.

[HG99b] J. Heinanen et R. Guerin. *A Two Rate Three Color Marker*. Request For Comments 2698, IETF, septembre 1999.

[J+99] V. Jacobson et al. *An Expedited Forwarding PHB*. RFC 2598, IETF, juin 1999.

[Jac88] Van Jacobson. Congestion avoidance and control. *In : Proc. of ACM SIGCOMM*, pp. 314–329. Stanford, CA, août 1988.

[Jai89] R. Jain. A delay based approach for congestion avoidance in interconnected heterogeneous computer networks. *Proc. of ACM SIGCOMM*, 1989, pp. 56–71.

[JD02] H. Jiang et C. Dovrolis. Passive estimation of tcp round-trip times. *ACM SIGCOMM Computer Communication Review*, vol. 32, juillet 2002, pp. 75–88.

[KAJ01] K. Kumar, A. Ananda et L. Jacob. A memory based approach for a TCP-friendly traffic conditioner in diffserv networks. *In : Proc. of the IEEE International Conference on Network Protocols - ICNP*. Riverside, California, USA, novembre 2001.

[Kes97] Srinivasan Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley Publishing Company, 1997.

[LFMF03] Emmanuel Lochin, Anne Fladenmuller, Jean-Yves Moulin et Serge Fdida. Energy consumption models for ad-hoc mobile terminals. *In : The 2nd Mediterranean Workshop on Ad-Hoc Networks - Med-Hoc Net 2003 - IFIP-TC6-WG6.8*. Mahdia, Tunisia, juin 2003.

[LZH99] Wei Lin, Rong Zheng et Jennifer C. Hou. How to make assured service more assured. *In : Proc. of the IEEE International Conference on Network Protocols - ICNP*, p. 182. Toronto, Canada, novembre 1999.

[MBDL99] M. May, J. Bolot, C. Diot et B. Lyles. Reasons not to deploy RED. *In : Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS*, pp. 260–262. London, June 1999.

[Med00] Octavio Medina. Adserv: Altq-based diffserv, 2000. http://www.rennes.enst-bretagne.fr/~medina/ds-imp/.

[Med01] Octavio Medina. *Etude Des Algorithmes D'attribution De Priorités Dans Un Internet à Différenciation De Services.* These de doctorat, ENST Bretagne, Rennes, France, mars 2001.

[MSZ03] Marco Mellia, Ion Stoica et Hui Zhang. TCP-aware packet marking in networks with diffserv support. *Computer Networks*, vol. 42, n 1, mai 2003, pp. 81–100.

[NBBB98] K. Nichols, S. Blake, F. Baker et D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* Request For Comments 2474, IETF, décembre 1998.

[NJZ99] K. Nichols, V. Jacobson et L. Zhang. *A Two-bit Differentiated Services Architecture for the Internet.* Request For Comments 2638, IETF, juillet 1999.

[NPE00] B. Nandy, P.Pieda et J. Ethridge. Intelligent traffic conditioners for assured forwarding based differentiated services networks. *In : IFIP High Performance Networking.* Paris, France, juin 2000.

[NW01] G. Nahrstedt et Y. Wichadakul. *An XMLbased Quality of Service Enabling Language for the Web.* Rapport technique, Department of Computer Science University of Illinois at Urbana-Champaign, Urbana, avril 2001.

[Pax97] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, vol. 5, n5, 1997, pp. 601–615.

[PC04] Eun-Chan Park et Chong-Ho Choi. Proportional bandwidth allocation in diffserv networks. *In : Proc. of IEEE INFOCOM.* Hong Kong, mars 2004.

[PF91] D. W. Petr et V. S. Frost. Nested threshold cell discarding for atm overload control optimization under cell loss constraints. *In : Proc. of IEEE INFOCOM*, pp. 1403–1412.

[PFTK98] Jitedra Padhye, Victor Firoiu, Don Towsley et Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. *In : Proc. of ACM SIGCOMM*, pp. 303–314. Vancouver, CA, 1998.

[PG92] A. K. Parekh et R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks - the single node case. *In : Proc. of IEEE INFOCOM*, pp. 914–924.

[PTCD01] Konstantina Papagiannaki, Patrick Thiran, Jon Crowcroft et Christophe Diot. Preferential treatment of acknowledgment packets in a differentiated services network. *In : Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS*, pp. 187–201.

[QZK01] L. Qiu, Y. Zhang et S. Keshav. Understanding the performance of many TCP flows. *Computer Networks*, vol. 37, n3-4, novembre 2001, pp. 277–306.

[RFB01] K. Ramakrishnan, S. Floyd et D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP.* Request for comments, IETF, septembre 2001.

[Riz97] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communications Review*, vol. 27, n1, janvier 1997.

[RKV+01] Tomas Robles, Arndt Kadelka, Hector Velayos, Antti Lappetelainen, Andreas Kassler, Hui Li, Davide Mandato, Jussi Ojala et Bernhard Wagmann. Qos support for an all-ip system beyond 3g. *IEEE Communications Magazine*, vol. 39, n8, août 2001, pp. 64–72.

[Rob98] J. Roberts. Quality of service guarantees and charging in multi-service networks. *IEICE Transactions on Communications*, vol. 81B, n5, mai 1998.

[Roc98] Vincent Roca. Bench: a network performance evaluation environment, 1998. http://www.inrialpes.fr/planete/people/roca/bench/bench.html.

[SND+00] Sambit Sahu, Philippe Nain, Christophe Diot, Victor Firoiu et Donald F. Towsley. On achievable service differentiation with token bucket marking for TCP. *In : Measurement and Modeling of Computer Systems*, pp. 23–33.

[SNP99] N. Seddigh, B. Nandy et P. Pieda. Bandwidth assurance issues for TCP flows in a differentiated services network. *In : Proc. of IEEE GLOBECOM*, p. 6. Rio De Janeiro, Brazil, décembre 1999.

[SV96] M. Shreedhar et George Varghese. Efficient fair queueing using deficit round-robin. *IEEE/ACM Transactions on Networking*, vol. 4, n 3, juin 1996, pp. 375–385.

[Sys01] Cisco Systems. Release notes for cisco 7000 family for cisco ios release 12.1 e, 2001.

[TEQ] Traffic engineering for quality of service in the internet, at large scale. http://www.ist-tequila.org/.

[TFT] Tf-tant: Differentiated services testing. http://www.dante.net/quantum/qtp/.

[THD+99] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, R. Narayan et F. Reichmeyer. Internet2 qbone: Building a testbed for differentiated services. *IEEE Network*, vol. 13, n5, septembre 1999.

[WC91] Z. Wang et J. Crowcroft. A new congestion control scheme: Slow start and search. *ACM Computer Communications Review*, vol. 21, n 1, janvier 1991, pp. 32–43.

[WGC95] I. Wakeman, A. Ghosh et J. Crowcroft. Implementing real time packet forwarding policies using streams. *In : In Proceedings of USENIX Technical Conference*, pp. 71–82. New Orleans, Louisiana, janvier 1995.

[Whi97] P. White. Rsvp and integrated services in the internet: A tutorial. *IEEE Communications Magazine*, mai 1997, pp. 100–106.

[WMB00] F. Wang, P. Mohapatra et D. Bushmitch. A random early demotion and promotion marker for assured services. *IEEE Joural on Selected Areas in Communications*, vol. 18, n12, décembre 2000.

[YR99] Ikjun Yeom et Narasimha Reddy. Realizing throughput guarantees in a differentiated services network. *In : Proc. of IEEE International Conference on Multimedia Computing and Systems- ICMCS*, pp. 372–376. Florence, Italy, juin 1999.

[YR01a] Ikjun Yeom et Narasimha Reddy. Adaptive marking for aggregated flows. *In : Proc. of IEEE GLOBECOM*. San Antonio, Texas, USA, novembre 2001.

[YR01b] Ikjun Yeom et Narasimha Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Transactions on Networking*, vol. 9, n1, 2001, pp. 31–46.

[ZFB01] T. Ziegler, S. Fdida et C. Brandauer. Stability criteria of RED with TCP traffic. *In : IFIP ATM&IP Working Conference*. Budapest, juin 2001.

[ZFH99] Thomas Ziegler, Serge Fdida et Ulrich Hofmann. RED+ gateways for identification and discrimination of unfriendly best-effort flows in the internet. *In : IFIP Broadband Communications*, pp. 27–38.

Liste des acronymes

AF	*Assured Forwarding*
@IRS	Architecture Intégrée de Réseaux et Services
ALTQ	*ALTernative Queuing*
API	*Application Programming Interface*
PS	*AIMD Penalty Shaper*
AS	*Assured Service*
ATM	*Asynchronous Transfer Mode*
BE	*Best Effort*
BSD	*Berkeley Software Developpement*
CBQ	*Class Based Queuing*
CBR	*Constant Bit Rate*
CNRS	Centre National de la Recherche Scientifique
CSZ	*Clark Shenker Zhang*
DS	*Dynamic Shaper*
DSCP	*DiffServ CodePoint*
DIS	*Distributed Interactive Simulation*
ECN	*Explicit Congestion Notification*
EF	*Expedited Forwarding*
FIFO	*First In First Out*
GNU	*GNU's Not Unix*
GPS	*Generalized Processor Sharing*
GS	*Guaranteed Service*
IETF	*Internet Enginneering Task Force*
IP	*Internet Protocol*
ISDN	*Integrated Services Digital Network*
LAAS	Laboratoire d'Analyse et d'Architecture des Systèmes
LAN	*Local Area Network*
LIP6	Laboratoire d'Informatique de Paris VI

MTU	*Maximum Transmission Unit*
OSI	*Open Systems Interconnection*
PBS	*Partial Buffer Sharing*
PDU	*Protocol Data Unit*
PGPS	*Packet-by-packet Generalized Processor Sharing*
PHB	*Per Hop Behavior*
PQ	*Priority Queuing*
PS	*Penalty Shaper*
QoS	*Quality of Service*
RED	*Random Early Detection*
RFC	*Request For Comment*
RIO	*RED with IN and OUT*
RR	*Round Robin*
RSVP	*Resource ReserVation setup Protocol*
RTT	*Round Trip Time*
SFQ	*Stochastic Fair Queuing*
TBF	*Token Bucket File*
TBM	*Token Bucket Marker*
TCA	*Traffic Conditioning Agreement*
TCP	*Transmission Control Protocol*
TOS	*Type Of Service*
TRTCM	*Two Rate Three Color Marker*
TSW	*Time Sliding Window*
TSW3CM	*Time Sliding Window Three Color Marker*
TTL	*Time To Live*
UDP	*User Datagram Protocol*
WAN	*Wide Area Network*
WFQ	*Weighted Fair Queuing*
WF^2Q	*Worst-case-Fair Weighted Fair Queuing*
WRED	*Weighted Random Early Detection*
YAVOI	*Yet Another Victim Of Infocom*
YAAAAA	*Yet Another Annoying And Ambiguous Acronym*

Liste des figures

Liste des tableaux

Ce document a été rédigé et composé à l'aide de logiciels libres sur le système d'exploitation *NetBSD*[5]. L'éditeur de texte utilisé est *gvim* et le système de formattage est LaTeX (classe de document *book* modifiée). Les schémas ont été réalisés avec le logiciel de dessin vectoriel *xfig* ou *Dia* puis convertis au format *postscript encapsulé*. Les courbes ont été tracées à l'aide du logiciel *gnuplot*.
Tous ces logiciels utilisent des formats de données publics qui garantissent la pérennité et la convertibilité des documents, au contraire des formats secrets *propriétaires* qui rendent l'utilisateur dépendant d'un logiciel et d'un éditeur spécifique.

[5] `http://www.netbsd.org/`

Qualité de Service dans l'Internet :
Garantie de Débit TCP dans la Classe AF

Résumé Les travaux présentés dans cette thèse concernent la qualité de service dans les réseaux à commutation de paquets de grande dimension et plus spécifiquement la garantie de débit pour les flots TCP. L'architecture à différenciation de services, définie par l'IETF, dite DiffServ s'articule autour de trois services : le service garanti, le service assuré et le service par défaut dit « au mieux ». Le service assuré a été élaboré pour servir les flots à caractère élastique comme les flots TCP. Cependant, des problèmes subsistent en ce qui concerne cette garantie de débit pour ce type de flots dans ce service. Ces dernières années, un effort important a été fait pour améliorer les mécanismes d'ordonnancement, de classification et de rejet au sein des routeurs afin qu'ils répondent au mieux à la spécification du service assuré. Malheureusement, leurs choix de conception et la complexité de leur mise en œuvre est très souvent un frein à leur déploiement dans un Internet réel. Dans cette thèse, nous proposons une solution originale de conditionnement des flots TCP permettant de garantir un débit à la fois pour des flots individuels et des groupes de flots. Cette solution fonctionne quelquesoit le facteur d'echelle.

Mots clés Qualité de Service, différenciation de services, facteur d'échelle, agrégation de flots, service assuré, TCP.

Quality of Service in the Internet :
Guaranteed TCP Throughput in the AF Class

Abstract This work focuses on quality of service in large scale packet switching networks and, more specifically, throughput guarantee for TCP flows. The differentiated architecture, defined by the IETF, also called DiffServ, provides three services : the guaranteed service, the assured service and the best effort service. The assured service was conceived for serving elastic flows such as TCP flows. Nevertheless some problems persist when it comes to assuring a TCP throughput. These past years, a consequent effort has been made in order to improve scheduling, classification and dropping mechanisms in routers so that they match the assured service specification as best as possible. Unfortunately, design choices and complex implementation are frequent limitations to their deployment in the real Internet. In this thesis, we propose an original conditioning approach for TCP flows allowing for a guarantee both for single and aggregate TCP flows. This solution is scalable.

Keywords Quality of Service, differentiated services, scalability, aggregation, assured service, TCP