

Garanties de performance pour les flots IP dans l'architecture Flow-Aware Networking

Jordan Augé

▶ **To cite this version:**

Jordan Augé. Garanties de performance pour les flots IP dans l'architecture Flow-Aware Networking. Réseaux et télécommunications [cs.NI]. Télécom ParisTech, 2014. Français. NNT : 2014ENST0072 . tel-01354843

HAL Id: tel-01354843
https://pastel.hal.science/tel-01354843

Submitted on 19 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0072

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Jordan AUGÉ

le jeudi 27 Novembre 2014

Garanties de performance pour les flots IP dans l'architecture Flow-Aware Networking

Directeur de thèse : **Daniel KOFMAN**
Co-encadrement de la thèse : **James ROBERTS**

Jury

Mme. Isabelle GUERIN LASSOUS, Professeur, Université Lyon 1/LIP, Lyon	Rapporteur
M. Steve UHLIG, Queen Mary University of London, UK	Rapporteur
M. Tijani CHAHED, Professeur, Telecom SudParis, Evry	Examinateur
M. Fabien MATHIEU, Chercheur, Alcatel-Lucent Bell Labs France	Examinateur
M. James ROBERTS, Chercheur senior, IRT System-X	Directeur de thèse (industriel)
M. Daniel KOFMAN, Professeur, Télécom ParisTech	Directeur de thèse (académique)

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech

46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Garanties de performance pour les flots IP dans l'architecture Flow-Aware Networking

Résumé :
La thèse s'intéresse à la réalisation d'une architecture de Qualité de Service en rupture avec les approches classiques, permettant d'offrir des garanties de bout en bout pour le trafic. L'approche Flow-Aware Networking considère le trafic au niveau des flux applicatifs, pour lesquels des modèles de trafic simples mais robustes conduisent à une relation fondamentale entre ressources offertes par le réseau, la demande générée, et la performance obtenue. En particulier, l'architecture de routeur Cross-Protect propose la combinaison d'un ordonnancement fair queueing et d'un contrôle d'admission afin d'assurer de manière implicite la performance des flots streaming et élastique, sans nécessiter ni marquage ni procotole de signalisation. Dans un tel contexte, nous considérons le dimensionnement des buffers au sein des routeurs, l'introduction d'un ordonnancement de type fair queueing dans le réseau et son impact sur la performance des protocoles TCP, ainsi que la réalisation d'un algorithme de contrôle d'admission approprié. Pour terminer, une déclinaison de cette architecture pour le réseau d'accès est proposée.

Mots clés : Qualité de Service ; Performance ; Flots IP ; Dimensionnement de buffer ; Ordonnancement équitable ; TCP ; Différenciation de service implicite ; Contrôle d'admission ; Réseaux de cœur ; Réseaux d'accès

Flow-level performance guarantees for IP traffic in the Flow-Aware Networking architecture

Abstract :
The thesis deals with the realization of a Quality of Service architecture that breaks with traditional approaches, and allows end-to-end performance guarantees for the traffic. The Flow-Aware Networking approach considers the traffic at the flow level, for which simple but robust traffic models lead to a fundamental relationship between the resources offered by the network, the demand and the obtained performance. In particular, the Cross-Protect router architecture proposes the combination of a fair queueing scheduler, and an admission control so as to implicitly ensure the performance of both streaming and elastic flows, without the need for any marking nor signalization protocol. In such a context, we consider the sizing of router buffers, the introduction of fair queueing scheduling inside the network and its impact on the performance of TCP protocols, as well as the realization of a suitable admission control algorithm. Finally, a declination of this architecture for the access network is proposed.

Keywords : Quality of service ; Performance ; IP flows ; Buffer sizing ; Fair queueing ; TCP ; Implicit service differenciation ; Admission control ; Core networks ; Access networks

Table des matières

1 Introduction **9**
- 1.1 Introduction . . . 10
  - 1.1.1 Architecture d'un réseau IP . . . 10
  - 1.1.2 Routage des flux de données . . . 11
  - 1.1.3 Dimensionnement de réseau . . . 11
  - 1.1.4 Enjeux des mécanismes de QoS pour un opérateur . . . 11
- 1.2 Contenu et contributions de la thèse . . . 12
  - 1.2.1 Présentation . . . 12
  - 1.2.2 Plan de la thèse . . . 12
  - 1.2.3 Résumé des contributions et publications associées . . . 14

2 Garanties de service pour les flots IP : un état de l'art **17**
- 2.1 Le modèle *Best-Effort* actuel et ses limites . . . 18
  - 2.1.1 Réseau *Best Effort* et ordonnancement FIFO/DropTail . . . 18
  - 2.1.2 Protocoles de transport . . . 18
  - 2.1.3 Comportement de TCP . . . 18
  - 2.1.4 Limites de TCP . . . 19
  - 2.1.5 Vers un déploiement d'architectures de qualité de service ? . . . 19
- 2.2 Garanties de service pour les flots IP . . . 19
  - 2.2.1 Granularité du trafic IP . . . 20
  - 2.2.2 2 classes d'applications : S et E . . . 20
  - 2.2.3 Régimes opérationnels des liens . . . 20
- 2.3 Techniques d'amélioration de la QoS . . . 21
  - 2.3.1 Allocation de ressources et contrôle de congestion . . . 22
  - 2.3.2 Ordonnancement . . . 23
  - 2.3.3 Contrôle de la surcharge . . . 24
  - 2.3.4 Différenciation et classes de trafic . . . 24
- 2.4 Analyse des principales architectures de qualité de service . . . 25
  - 2.4.1 Présentation des principales architectures . . . 25
  - 2.4.2 Discussion et positionnement de FAN . . . 28

3 Flow-Aware Networking (FAN) **31**
- 3.1 Introduction . . . 32
  - 3.1.1 Ressources – Demande – Performance . . . 32
  - 3.1.2 Modèles de trafic . . . 32
- 3.2 Modélisation du trafic streaming . . . 33
  - 3.2.1 Multiplexage statistique "sans buffer" : cas d'un lien isolé . . . 33
  - 3.2.2 Garanties de performance de bout en bout . . . 34
- 3.3 Modélisation du trafic élastique . . . 34
  - 3.3.1 Modèle de sessions Poisson . . . 34
  - 3.3.2 Modèle processeur partagé : cas d'un lien isolé . . . 34
  - 3.3.3 Extension à un réseau de données . . . 37
- 3.4 Intégration des flots *streaming* et élastiques . . . 38
- 3.5 L'architecture de différentiation implicite de service Cross-Protect . . . 39
  - 3.5.1 PFQ : un algorithme de FQ extensible . . . 39
  - 3.5.2 Indicateurs de congestion . . . 40
  - 3.5.3 Contrôle d'admission basé sur des mesures . . . 41
- 3.6 Vers la réalisation d'un routeur Cross-Protect . . . 42
  - 3.6.1 Évaluation de l'architecture en simulation . . . 42

3.6.2 Évaluation de l'architecture par expérimentation . . . 42

4 Dimensionnement des buffers pour les routeurs IP de cœur de réseau 45
4.1 Introduction . . . 46
4.2 Dimensionnement des buffers et TCP . . . 47
4.2.1 Dimensionnement pour un flot TCP . . . 47
4.2.2 Synchronisation des flots TCP : extension à N flots . . . 47
4.3 État de l'art sur le dimensionnement de buffers . . . 48
4.3.1 Dimensionnement proportionnel au nombre de flots . . . 48
4.3.2 Vers une réduction drastique de la taille des buffers . . . 48
4.3.3 Vers une distinction flots *bottlenecked*/*non-bottlenecked* . . . 49
4.3.4 Cas où les flots sont tous *non-bottlenecked* . . . 49
4.4 Impact de la taille du buffer sur la performance des flots . . . 49
4.4.1 Régimes opérationnels . . . 49
4.4.2 Performance des flots *streaming* . . . 50
4.4.3 Performance des flots élastiques . . . 50
4.5 Dimensionnement des buffers dans le régime transparent . . . 50
4.5.1 Arrivées localement Poisson . . . 51
4.5.2 Calcul de la taille requise du buffer . . . 51
4.6 Dimensionnement des buffers pour le régime élastique . . . 52
4.6.1 Comportement des flots *bottlenecked* . . . 52
4.6.2 Flots *bottlenecked* au débit crête non limité . . . 55
4.6.3 Trafic élastique avec des flots à débit crête limité . . . 58
4.7 Conclusions . . . 59

5 Partage de bande passante étendu : nouveaux protocoles TCP & *fair queueing* 61
5.1 Introduction . . . 62
5.2 État de l'art : Nouveaux protocoles TCP et évaluation de leur performance . . . 63
5.2.1 Faiblesses de TCP Reno pour les transferts à haut débit . . . 63
5.2.2 Nouveaux protocoles TCP . . . 64
5.3 Motivations pour l'intégration de fair queueing . . . 65
5.3.1 Environnement de simulation . . . 65
5.3.2 Performance des flots *non-bottlenecked* . . . 66
5.3.3 Performance des flots *bottlenecked* . . . 70
5.4 Comportement de TCP dans un environnement équitable . . . 74
5.4.1 Motivations . . . 74
5.4.2 Environnement de simulation . . . 74
5.4.3 Modèle d'étude . . . 75
5.4.4 Résultats . . . 75
5.5 Discussion . . . 84
5.5.1 Enjeux . . . 84
5.5.2 Indicateurs de congestion . . . 84
5.5.3 Apports du *fair queueing* . . . 84
5.6 Conclusion . . . 85

6 Un algorithme de MBAC pour Cross-Protect 87
6.1 Introduction . . . 89
6.2 État de l'art des approches de MBAC . . . 89
6.2.1 Measured sum . . . 89
6.2.2 Borne de Hoeffding . . . 90
6.2.3 Calcul d'enveloppes de trafic . . . 90
6.2.4 Approche basée sur la théorie de la décision . . . 90
6.2.5 Théorie de la bande passante équivalente . . . 90
6.2.6 Stratégies de *back off* . . . 91
6.2.7 Décomposition selon différentes échelles de temps . . . 91
6.2.8 Discussion . . . 93
6.3 Implémentation et évaluation de l'algorithme de Grossglauser et Tse . . . 93
6.4 Un algorithme de MBAC pour Cross-Protect . . . 94
6.4.1 Critères d'admission . . . 94
6.4.2 Moyenne et variance du *priority load* . . . 94
6.4.3 Approximation basée sur une hypothèse Poissonnienne . . . 95

6.4.4 Limite du nombre d'arrivées par intervalle . . . 96
6.4.5 Instabilité du *priority load* mesuré . . . 96
6.4.6 Influence de la taille du slot sur les mesures . . . 97
6.5 Évaluation des algorithmes . . . 98
6.5.1 Environnement de simulation . . . 98
6.5.2 Critères de performance . . . 98
6.5.3 Utilisation et probabilité de débordement . . . 99
6.5.4 Prédictibilité du MBAC XP . . . 99
6.5.5 Performance avec un mélange de débits crête . . . 100
6.5.6 Impact de la gigue . . . 101
6.5.7 Contrôle d'admission en régime non-stationnaire : situations de *flashcrowd* . . . 102
6.6 Conclusions . . . 103
6.A Estimation de la variance sur un intervalle de taille quelconque . . . 105

7 Self-Protect : une approche orientée flot et centrée sur l'utilisateur pour la QoS des liens d'accès 107
7.1 Introduction . . . 109
7.2 Mécanismes actuels de garantie de service pour le réseau d'accès . . . 109
7.2.1 Gestion du trafic montant . . . 109
7.2.2 Gestion du trafic descendant . . . 110
7.2.3 Discussion . . . 110
7.3 Vers une architecture de QoS pour le réseau d'accès . . . 110
7.3.1 Classes de trafic et ordonnancement . . . 111
7.3.2 Différenciation . . . 111
7.3.3 Gestion de la surcharge . . . 112
7.4 L'architecture Self-Protect . . . 112
7.4.1 Fonctionnement de Self-Protect . . . 112
7.4.2 Gestion des tables de flots . . . 113
7.4.3 Contrôleur et protocole de signalisation . . . 113
7.5 Réalisation et évaluation de l'architecture Self-Protect . . . 115
7.5.1 Prototype . . . 115
7.5.2 Évaluations . . . 117
7.6 Conclusion . . . 119

9 Conclusion 121

Chapitre 1

Introduction

Résumé

Ce chapitre introductif présente le contexte dans lequel s'inscrit notre étude, qui propose de développer et d'étudier un ensemble de mécanismes réseaux permettant d'assurer des garanties de services (QoS [1]) de bout en bout pour les flots de données au sein du réseau Internet.

Nous passerons en revue les principales architectures de qualité de service qui ont été proposées dans la littérature, avant de nous intéresser plus en détail à une proposition d'architecture de routeur en rupture avec les approches classiques. Il s'agit de l'architecture *Flow-Aware Networking* (FAN), qui propose de considérer le trafic au niveau des flux de données, et pour laquelle nous avons développé un prototype de routeur, évalué à la fois dans le simulateur ns-2, et sur une plateforme expérimentale (*testbed*) basée sur une implémentation dans le noyau Linux.

L'approche FAN se fonde sur un ensemble de modèles de trafic simples mais robustes, permettant de comprendre qualitativement la relation fondamentale qu'il existe entre les ressources du réseau, la demande en trafic et la performance obtenue pour les flots. Nous nous attacherons particulièrement à considérer un modèle de trafic plus réaliste que ceux utilisés généralement pour des études similaires.

Ce chapitre présente ainsi les bases nécessaires aux contributions de cette thèse, qui s'appuient fortement sur ces considérations de trafic. Ces contributions toucheront à différents aspects liés à la réalisation et au déploiement d'un tel routeur, et à l'évaluation de la performance ainsi obtenue :

- quelle taille de buffer permet une performance satisfaisante pour les flots de données, tout en restant suffisamment modeste pour être réalisable ?
- quel est l'impact de l'introduction d'un ordonnancement équitable dans le réseau, et quelles évolutions permet-il au niveau des protocoles de transport ?
- enfin, quels mécanismes de contrôle peut-on intégrer dans le réseau, afin de le protéger en cas de situations exceptionnelles de surcharge ?

1. Dans ce manuscrit, nous référerons la qualité de service par le sigle anglais plus connu de QoS (*Quality of Service*).

1.1 Introduction

Le réseau Internet tel que nous le connaissons aujourd'hui permet l'échange d'informations à l'échelle mondiale, et offre une multitude de services. La dernière décennie à connu une explosion des débits et une multiplication des applications : des exemples notables sont les solutions de pair-à-pair, ou encore les flux de téléphonie, de télévision ou de vidéo, résultant de la convergence des réseaux historiques d'opérateurs [2].

La transmission des données au travers du réseau repose sur le protocole IP (*Internet Protocol*), qui a pour but d'assurer un adressage unique de l'ensemble des terminaux connectés au réseau. Le flux de données est encapsulé dans un ensemble de paquets de taille variable, pour lesquels aucun chemin n'est préétabli. Ceux-ci sont acheminés au mieux par le réseau sans aucune distinction sur leur contenu : le réseau IP est dit *Best Effort*. Il ne contient aucun mécanisme permettant le transport de plusieurs applications simultanément, ou le maintien d'une connexion fiable de bout en bout. Ce sont respectivement les surcouches protocolaires de type UDP et TCP qui en sont responsables. C'est également à TCP que sont aujourd'hui confiées les tâches de partager et d'exploiter équitablement la bande passante, ou d'éviter les situations de congestion.

Un des principes fondateurs de l'Internet est de garder un réseau très simple, et d'ajouter de l'intelligence en bordure de réseau, au travers de protocoles bâtis sur IP. Il est reconnu que c'est cette simplicité qui a valu son succès au protocole (de par les faibles coûts qu'il engendre) et qui a permis sa polyvalence et l'émergence de nombreuses applications.

On peut y opposer les réseaux bâtis sur le protocole ATM [158], qui ont été conçus pour intégrer nativement différents types de trafic, incluant voix, vidéos et données. Ceux-ci ont été globalement supplantés par IP, mais subsistent dans certaines parties du réseau (comme l'accès ADSL). On en retrouvera de nombreux mécanismes dans les solutions de QoS proposées pour IP (voir Chapitre 2).

1.1.1 Architecture d'un réseau IP

Au niveau physique, le réseau est constitué d'un ensemble d'équipements interconnectés, commutateurs (ou *switches*) et routeurs, sous le contrôle de différentes entités indépendantes appelées Systèmes Autonomes (ou AS, *Autonomous Systems*) : ce sont généralement des fournisseurs d'accès, de contenu, des universités, des entreprises, etc.

Dans un réseau d'opérateur, on peut distinguer trois composantes structurellement différentes, en termes de technologies, de débits des liens ou encore de nombre d'utilisateurs simultanées. Leur étude est généralement spécifique et met en jeu des modèles théoriques distincts.

le réseau d'accès est la partie en bordure de réseau reliant le client à l'opérateur. C'est le cas du réseau ADSL par exemple, qui relie le modem/la passerelle domestique à un DSLAM, ou des réseaux sans fil et mobiles tels que WiFi, WiMAX, UMTS, etc. Sa capacité est aujourd'hui plutôt limitée (surtout dans le sens montant pour un réseau ADSL) et il est souvent un goulot d'étranglement pour le trafic (les connexions peuvent facilement atteindre le débit d'accès), mais la venue de la fibre optique pourrait déplacer la congestion plus en amont dans le réseau.

le réseau de collecte centralise le trafic de plusieurs DSLAM (prenons l'exemple de l'ADSL) et le connecte au réseau de l'opérateur proprement dit par l'intermédiaire du BRAS (*Broadband Remote Access Server*), ou Équipement d'Accès au Service (EAS). Typiquement le réseau de collecte est en ATM ou Ethernet Gigabit, et peut potentiellement être un point de saturation en fonction des nouveaux services consommateurs de bande passante comme la vidéo à la demande par exemple.

le réseau de cœur est la partie centrale qui offre la connectivité entre les utilisateurs et les différents services. La capacité des liens est généralement grande comparée à celle des réseaux d'accès (plusieurs dizaines de gigabits par seconde), puisque les liens de cœur transportent le trafic d'un très grand nombre d'utilisateurs.

D'autres contextes tels que les *datacenters* possèdent des caractéristiques particulières en terme de débits, de latences et de trafic.

Dans cette thèse, nous nous intéressons principalement à la gestion du trafic dans un réseau de cœur, mais les mécanismes sont directement applicable au contexte des *datacenters*. Le Chapitre 7 considère une déclinaison de ces mécanismes pour le réseau d'accès.

2. Cette convergence à notamment permis une réduction des coûts par rapport au maintien de réseaux séparés, comme le Réseau Téléphonique Commuté Public (RTC), ou en anglais PSTN (Public Switched Telephone Network)

1.1.2 Routage des flux de données

Afin d'assurer une connectivité globale, les différents acteurs établissent des contrats d'interconnection et définissent des politiques de routage, qui décident de la manière dont sont routés les flux de données entre eux. En suivant Gao [59], on peut les catégoriser selon trois grands types :

client-fournisseur : un réseau achète la connectivité Internet à un autre réseau (on parle souvent de hiérarchie *tier-1*, *tier-2*,etc.) ;

peering : deux réseaux décident d'échanger du trafic sans coût pour les deux parties ;

sibling : un accord plus complexe ou les deux réseaux offrent chacun la connectivité à l'autre (ce peut être le cas lorsque deux AS historiques sont regroupés sous la même entité administrative).

Les décisions de routages sont réalisées grâce au protocole BGP[3]. Ce protocole ne régit que les échanges entre les différents AS et en aucun cas l'acheminement des flux à l'intérieur d'un AS. Chacun est libre d'utiliser le protocole de son choix, les plus courants étant IS-IS[4], OSPF[5] ou RIPv2[6].

1.1.3 Dimensionnement de réseau

La première étape qu'un opérateur doit considérer afin d'offrir des garanties de service au trafic qu'il transporte est de dimensionner correctement son réseau, afin de faire face à la demande de ses utilisateurs. Connaître cette demande nécessite des mesures au sein du réseau, afin de connaitre la matrice de trafic entre les différents points du réseau. Il est enfin possible de l'extrapoler afin de faire des prévisions à plus ou moins long terme et de planifier les investissements à effectuer (ajout de liens, de bande passante, etc.).

On considère généralement la demande en trafic en heure de pointe, et le dimensionnement est effectué de manière robuste à une panne (afin d'absorber la surcharge en trafic générée par un tel cas). Comme nous le verrons dans la suite de cette thèse, le volume de trafic seul ne suffit généralement à établir la performance des différents types de flux, et il convient d'avoir des modèles simples, mais efficaces et robustes de dimensionnement.

L'ajout de mécanismes de QoS constitue une garantie supplémentaire, et permet de gérer des situations exceptionnelles : cas de liens coûteux à dupliquer, de pannes multiples ou autre événement imprévu. Ils ne se substituent en aucun cas à un dimensionnement correct du réseau.

1.1.4 Enjeux des mécanismes de QoS pour un opérateur

Il est crucial pour un opérateur d'assurer la qualité de service des flux applicatifs qu'il transporte, que ce soit pour ses propres utilisateurs, ou pour les client auxquels il assure le transit (risque de pénalités).

A l'heure actuelle, peu de mécanismes de QoS sont utilisés dans les réseaux d'opérateurs pour l'Internet public, sauf d'éventuels mécanismes de filtrage, etc. (La situation est différente cepdendant pour les réseaux privés d'entreprise.) La raison essentielle est qu'il semble plus simple et moins onéreux de surdimensionner les liens, et qu'aucun mécanisme n'offre à l'heure actuelle de garanties raisonnables. et e est toutefois remise en cause avec la montée des débits d'accès (fibre optique), et du volume de trafic généré par chaque utilisateur (pair-à-pair, vidéo HD par exemple), qui pourraient rapidement saturer les réseaux d'accès ou de collecte. De plus, certains liens sont aujourd'hui d'ores et déjà saturés (*i.e.*, certains liens de *peering* ou transatlantiques), et le surdimensionnement n'est ni envisageable ni durable.

On peut ainsi s'attendre à un regain d'intérêt pour une solution qui sera efficace et simple à mettre en œuvre et configurer. Il conviendra aussi d'assurer la compatibilité avec d'éventuelles régulations liées à la neutralité des réseaux. Les avantages de la solution que nous préconisons sont nombreux :

- garantir un partage efficace et équitable de la bande passante disponible, et se prémunir contre des flux agressifs ou malicieux qui pourraient causer des pertes à d'autres ;
- gérer des surcharges exceptionnelles, où la demande en trafic est supérieure à la capacité des liens ;
- obtenir une performance prévisible, idéalement indépendamment du profil des applications véhiculées par le réseau ;
- éventuellement améliorer l'utilité globale du réseau, grâce à ces mécanismes de différenciation par exemple.

3. Border Gateway Protocol
4. Intermediate System to Intermediate System
5. Open Shortest Path First
6. Routing Information Protocol

1.2 Contenu et contributions de la thèse

1.2.1 Présentation

L'architecture *Flow Aware Networking* (FAN) propose une gestion du trafic IP en rupture avec les architectures classiques de qualité de service, inappropriées pour comprendre et garantir la performance des applications. FAN propose de considérer le trafic au niveau des flots utilisateurs, en permettant aux équipements dits *flow-aware* de les reconnaitre. La modélisation au niveau flot permet l'utilisation de descripteurs de trafic simples, et de modèles de trafic robustes qui permettent d'établir la relation entre la capacité du réseau, la demande utilisateur et la performance obtenue. Ils servent de base à un dimensionnement correct du réseau, ainsi qu'une compréhension du trafic qui sera nécessaire aux analyses que nous ferons dans la suite de ce document.

Plus précisément, nous considérons une implémentation de FAN adaptée pour le cœur de réseau, Cross-Protect, qui combine un ordonnancement par flot *fair queueing* – qui réalise un partage équitable de la bande passante – à un contrôle d'admission des flots – qui permet de contrôler la charge offerte au lien. Un tel système possède la propriété intéressante de différencier implicitement entre différentes classes de flots qui possèdent des contraintes de qualité de service différentes. On parlera respectivement de flots *streaming* et élastiques, qui seront introduits plus précisément dans le chapitre suivant.

Cette thèse s'appuie sur l'architecture FAN et les modèles de trafic associés, et a pour objectif de mieux comprendre la performance d'une telle approche. Notamment, nous considérons plusieurs aspects liés à la réalisation et l'évaluation d'un prototype, et son interaction avec un trafic réaliste.

- le dimensionnement des buffers au sein des équipements, ainsi que leur impact sur la performance ;
- l'introduction de nouveaux protocoles TCP et leur comportement, les apports d'un ordonnancement *fair queueing* sur la performance des flots *streaming* et élastiques, ainsi que l'impact de l'introduction de tels mécanismes dans le réseau et notamment le dimensionnement des buffers ;
- la réalisation d'un contrôle d'admission efficace dans un contexte de différenciation implicite.

Comme nous l'avons précisé plus haut, la portée des résultats obtenus est plus générale que le cœur de réseau.

1.2.2 Plan de la thèse

Les premiers chapitres de cette thèse présentent un état de l'art des architectures de garanties de service pour les réseaux IP, ainsi qu'une description détaillée de l'architecture *Flow-Aware Networking* dont l'évaluation fait l'objet de cette thèse. Nous y décrivons notamment les réalisations logicielles qui nous ont permis son évaluation :

Chapitre 2 : Qualité de service dans les réseaux IP Ce chapitre souligne les insuffisances de l'état actuel des réseaux et de la stratégie couramment utilisée de surdimensionnement. Après un tour d'horizon des problématiques et des besoins, nous décrivons un ensemble de caractéristiques qui nous semblent nécessaires à une architecture de garantie de service, visant à assurer la performance des deux principaux types de flots, *streaming* et élastiques. Nous passons ensuite en revue un ensemble de techniques proposées pour gérer ou améliorer la performance des flux : il s'agit d'une part d'étendre les capacités des protocoles de transport de type TCP, et de l'autre d'ajouter des fonctionnalités dans le réseau pour complémenter voire prendre la main sur ces derniers. Nous faisons finalement un état de l'art des architectures existantes, ce qui nous permet de justifier l'intérêt d'une approche telle que FAN et de la positionner.

Chapitre 3 : Flow-Aware Networking (FAN) Ce chapitre présente la proposition FAN qui propose des mécanismes robustes de gestion du trafic au niveau flot, ainsi qu'une réalisation proposée pour un routeur de cœur de réseau : Cross-Protect. Contrairement aux architectures classiques, FAN permet d'établir la relation qui existe entre les ressources fournies par le réseau (capacité des liens, buffers), la demande utilisateur (caractérisée ici par la charge moyenne et le débit crête des flots) et la performance obtenue (des indicateurs statistiques).

Les modèles utilisés pour étudier la performance des flots *streaming* et *élastique*, ainsi que leur intégration sont à la base des réflexions présentées dans les chapitres suivants. Ils nous permettent notamment de considérer des scénarios de trafic réaliste (mélange de flots de différents débit crête, à charge donnée, dont certains sont goulottés sur le lien en cours).

Nous avons réalisé deux implémentations logicielles nous permettant par la suite l'évaluation de la proposition Cross-Protect ainsi que de nos contributions :

par simulation : plusieurs modules pour le simulateur ns-2, incluant les algorithmes de *fair queueing* et de contrôle d'admission, certains permettant la génération de flots de caractéristiques variées selon différents profils d'arrivées, ainsi qu'un ensemble de scripts permettant des simulations exhaustives et la représentation graphique des résultats.

par émulation : une implémentation dans un contexte GNU/Linux du routeur Cross-Protect – dont les mécanismes d'ordonnancement et de contrôle d'admission dans l'espace noyau pour des raisons de performance (après un prototypage dans l'espace utilisateur) –, ainsi que la réalisation d'une plateforme complète d'expérimentation : scénarios avec la génération de trafic réel et artificiel, mesure et représentation graphique, ayant permis la démonstration du concept et de sa faisabilité technique.

Chapitre 4 : Dimensionnement des buffers pour les routeurs de cœur de réseau Nous décrivons maintenant les contributions faites à l'architecture Cross-Protect pour le cœur de réseau, que nous avons présentée dans le chapitre précédent.

Dans ce chapitre, nous étudions l'impact du dimensionnement des buffers[7] au sein des routeurs IP sur la performance des flots. Ces buffers ont un double rôle aujourd'hui : d'une part il permettent d'absorber les rafales de paquets (à une petit échelle de temps) telles que celles provoquées par des arrivées simultanées ; d'autre part, il permettent d'assurer la performance des flux de transferts de données (basés sur TCP), dont le débit instantané peut varier sur une plus grande échelle de temps.

Nous nous limitons à des flots TCP standard[8], et à un ordonnancement FIFO (*First In First Out*) sur les liens. Les modèles fluides introduits dans FAN reposent sur une approximation fluide de TCP et ne prennent pas en compte les interactions au niveau paquet dues au protocole de transport et aux buffers (supposés de taille infinie).

Cette problématique est d'autant plus importante que les grandes tailles de buffers utilisées jusqu'alors sont fortement remises en question – à la fois pour des raisons de performance et par contrainte technique –, et plusieurs propositions suggèrent une réduction de leur taille plus ou moins drastique. Nous réconcilions ces différents résultats au travers d'un modèle réaliste de trafic, et montrons que la performance des flux TCP est fortement affectée par de faibles tailles de buffer, et notamment en présence d'un trafic de fond concurrent. Si dans un régime de lien transparent, de petits buffers suffisent en effet, ce n'est plus le cas lorsque les flots TCP se partagent effectivement la bande passante, et saturent le lien jusqu'à provoquer des pertes de paquets.

Dans un tel contexte, nous caractérisons le débit moyen obtenu par un flot, et donc son temps moyen de complétion au travers d'un modèle simple prenant en compte la performance réalisée par les flux TCP. Il apparait que la performance est entièrement conditionnée par le débit réalisé par un flot. En l'absence de modèle nous permettant d'établir une telle valeur, nous recourons à un ensemble de simulations. Il semble toutefois possible de réduire sensiblement la taille des buffers à condition de faire un léger compromis sur la performance. Il convient toutefois de ne pas dépasser une taille minimale en dessous de laquelle la performance des fortement dégradée. Nous caractérisons de manière empirique ce régime critique, qui permet de comprendre la performance des flots avec de petits buffers ; une taille voisine de 100 paquets semble assurer une performance convenable indépendamment de la capacité du lien.

Chapitre 5 : Partage de bande passante étendu : nouveaux protocoles TCP & *fair queueing* Ce chapitre suit naturellement le précédent, mais concerne un partage des ressources étendu au travers de l'utilisation de nouveaux protocoles de transport, et de la présence de mécanismes tels que le *fair queueing* dans le réseau. Nous montrons que le partage de bande passante ne peut alors plus être supposé optimal, et que l'utilisation de protocoles efficaces nécessite l'emploi de *fair queueing*.

Ce dernier permet de restaurer l'équité inter- et intraprotocolaire, et de permettre une utilisation maximale du lien. Sa capacité de différenciation permet de protéger les flots *streaming* des délais (dûs à des tailles de buffer importantes), et des pertes subies aux instants de saturation du buffer.

Dans un tel contexte, l'utilisation de protocoles TCP avancés n'est plus limitée par le besoin d'être équitable, et il est possible de concevoir de nouveaux protocoles plus efficaces pour exploiter les ressources disponibles (buffer et bande passante). Nous étudions notamment la performance d'un certain nombre de propositions actuelles avec des tailles de buffers réduites, avec la même approche que précédemment.

Il est intéressant de noter que FQ permet de simplifier le provisionnement des réseaux en ne se préoccupant plus de caractéristiques détaillées comme les protocoles de transport, et d'obtenir une per-

7. La traduction française littérale serait "tampons", mais pour des raisons de lisibilité, nous avons préféré garder le terme de *buffers* dans la suite du document.

8. les versions dénommées Reno/NewReno

formance prévisible pour les flots en fonction de critères tels que leur charge et leur débit. Il appartient aux utilisateurs d'exploiter indépendamment et au mieux la capacité qui leur est allouée par l'utilisation de protocoles efficaces dans l'environnement considéré. Il est ainsi probable qu'un environnement optique ou de *datacenter* soit limité à de très petites tailles de buffer, et il conviendra d'établir la relation donnant la performance des flots dans un tel contexte.

Chapitre 6 : Un algorithme de MBAC pour Cross-Protect Le terme MBAC (*Measurement-Based Admission Control*) désigne un contrôle d'admission basé sur des mesures.

Les deux chapitres précédents considèrent des liens dans un régime transparent ou élastique, mais ne prennent pas en compte des cas de surcharge où la demande est supérieure à la capacité des liens. Il convient d'éviter de telles situations où la performance est fortement dégradée par des mécanismes appropriés, puisque ni TCP ni FQ ne permettent de pallier à de telles situations.

L'architecture Cross-Protect intègre un mécanisme de contrôle d'admission des nouveaux flots, qui s'appuie sur des estimateurs fournis par l'ordonnanceur *fair queueing*. Il n'est cependant pas adapté au cas réaliste où la plupart des flots *streaming* et élastique ont un débit limité. Un tel algorithme doit se satisfaire d'informations limitées au sujet des flots, à la différence de la plupart des travaux existants, et doit accomplir un double objectif : maintenir un débit suffisant pour les flots élastique tout en assurant une latence faible pour un flot possédant un débit crête inférieur à un certain seuil.

Nous proposons un algorithme et évaluons sa performance dans plusieurs scénarios de régimes stationnaires (incluant des flots gigués) et dans un cas d'arrivées massives dans un petit intervalle de temps (phénomène de *flash crowd*). Il s'avère offrir un compromis satisfaisant entre la performance des flots et l'utilisation du lien.

Chapitre 7 : Self-Protect Ce dernier chapitre présente une déclinaison possible de l'architecture FAN pour le réseau d'accès : *Self-Protect*.

Si le besoin de mécanismes de qualité de service dans le cœur de réseau divise les avis, il est cependant reconnu comme nécessaire sur les liens d'accès. En particulier, les liens ADSL (notamment dans le sens montant) ou sans fil représentent un goulot d'étranglement pour le trafic. Les flux d'opérateurs sont généralement gérés par un mécanisme de priorités (flux de contrôle, téléphonie ou TV/VoD par exemple), mais le reste du trafic internet fait partie d'une même classe *Best Effort* et n'est pas différencié.

Self-Protect propose une implémentation de mécanismes *flow-aware* de gestion du trafic sur les équipements à même de les contrôler. Il s'agit de la passerelle domestique et du DSLAM, respectivement pour le trafic montant et descendant d'une connexion ADSL. Cette solution propose un ensemble de mécanismes d'ordonnancement et de gestion du trafic au niveau flot qui seront implémentés sur les différents équipements, afin de permettre à l'utilisateur de contrôler son trafic au sein de la bande passante qui lui est allouée. La détection du profil d'un flot sera faite par la passerelle domestique, avec éventuellement un support de l'utilisateur ou de ses applications, et transmis au DSLAM par signalisation. Le contexte d'un réseau d'accès où le nombre de flots est limité, et où l'utilisateur a la connaissance de son trafic permet en effet d'effectuer des opérations de traitement plus complexes, et d'optimiser les ressources réseau qui sont très coûteuses.

Une plateforme expérimentale a été réalisée dans l'environnement GNU/Linux afin de démontrer la faisabilité et l'efficacité de la solution. Une telle solution permet en outre la mesure d'informations qui peuvent être utilisées pour aider l'utilisateur à mieux comprendre la performance de sa connexion, ou fournir un support à d'autres applications de monitoring.

1.2.3 Résumé des contributions et publications associées

Le tableau suivant résume les principaux sujets abordés dans les différents chapitres de la thèse, les méthodes d'évaluation employées (modélisation, simulation, émulation), et donne des pointeurs vers les publications relatives à chaque chapitre.

Chapitre	Méthodologie	Mots-clés	Publications
Chapitre 2	État de l'art	Réseaux IP ; flots *streaming* et élastique ; Architectures de QoS	
Chapitre 3	implémentation ; testbed ; simulation ; émulation ; expérimentations	Flow-Aware Networking ; Cross-protect ; modèles de trafic	
Chapitre 4	modélisation ; simulations	Dimensionnement des buffer ; performance des flots élastiques	[P3, P4]
Chapitre 5	simulations	*fair queueing* ; nouveaux protocoles TCP ; performance des flots ;	[P1, P2]
Chapitre 6	modélisation ; algorithmes ; simulations	MBAC ; différenciation	[P5]
Chapitre 7	implémentation ; testbed ; expérimentations		

Chapitre 2

Garanties de service pour les flots IP : un état de l'art

Résumé

Ce chapitre passe en revue un ensemble de mécanismes de gestion du trafic, ainsi que les principales architectures de garantie de service proposées dans la littérature. Nous posons les bases nécessaire à l'analyse des différentes propositions au vu de notre compréhension du trafic, de sa performance et du challenge que représente l'introduction de nouveaux mécanismes dans un réseau opérationnel. Une discussion des avantages et des faiblesses de chaque solution nous permet de positionner et de justifier l'approche alternative *Flow-Aware Networking* (FAN), qui sera détaillée dans le prochain chapitre.

Contributions :

- Proposition d'une méthodologie d'évaluation des solutions de QoS par rapport aux besoins des différentes types de flots ;
- État de l'art des principales architectures de QoS.

Sommaire

2.1 Le modèle *Best-Effort* actuel et ses limites 18
2.2 Garanties de service pour les flots IP 19
2.3 Techniques d'amélioration de la QoS 21
2.4 Analyse des principales architectures de qualité de service 25

2.1 Le modèle *Best-Effort* actuel et ses limites

2.1.1 Réseau *Best Effort* et ordonnancement FIFO/DropTail

Le principe fondateur de l'Internet est d'offrir un réseau, fondé sur la brique IP, qui offre une connectivité de bout en bout sans aucune garantie de service [45]. On parle de réseau *Best Effort*. Les routeurs effectuent un routage simple des datagrammes IP en utilisant des files d'attente FIFO (*First In First Out*), dans lesquelles un paquet entrant est rejeté lorsque la file est pleine (mécanisme *DropTail*).

Il appartient ainsi aux couches supérieures, situées en bordure du réseau, d'assurer les fonctionnalités manquantes. C'est ainsi que le protocole de transport TCP a été proposé afin de réaliser un transfert fiable et équitable des données. On peut également citer des applications pair-à-pair qui font transiter le trafic au sein d'un overlay géré au niveau applicatif.

Cette architecture simple a permis l'émergence d'une multitude d'applications et a permis au réseau d'évoluer jusqu'à celui que nous connaissons aujourd'hui. Elle est toutefois remise en cause notamment pour son insuffisance à offrir un traitement avancé du trafic, ainsi que les garanties de service plus strictes requises par les nouvelles utilisations du réseau. Ces constatations ont motivé l'évolution des protocoles de transport, ainsi que l'introduction dans le réseau de mécanismes permettant un contrôle plus ou moins avancé du trafic.

2.1.2 Protocoles de transport

Les protocoles de transport utilisés majoritairement sur le réseau sont respectivement TCP (il en existe plusieurs variantes) et dans une moindre mesure UDP.

TCP (*Transmission Control Protocol*) permet d'assurer des transferts fiables, en mode connecté, entre deux points et est ainsi généralement utilisé pour supporter les flots élastiques. TCP à notamment pour objectif le partage équitable des ressources réseau disponibles ; il existe pour cela un grand nombre d'algorithmes dont nous présenterons les principaux dans la suite.

Dans une moindre mesure, on trouve aussi UDP (*User Datagram Protocol*) qui permet l'envoi non-fiable de données entre deux entités, et fonctionne en mode non-connecté. UDP est généralement associé avec les flots *streaming* et n'implémente pas de mécanisme de contrôle. Notons que les flots *streaming* peuvent également être transportés par TCP, et que l'on trouve également des protocoles adaptatifs basés sur UDP. Lorsque des flots UDP non-adaptatifs sont en compétition avec des flots TCP, ces derniers ont tendance à adapter leur débit et être défavorisés. Bien qu'ils subissent également des pertes, les flots UDP parviennent généralement à émettre à leur débit intrinsèque.

2.1.3 Comportement de TCP

La prédominance du protocole TCP explique pourquoi il se retrouve au cœur de nombreuses études, et que son fonctionnement puisse guider le dimensionnement de certaines ressources du réseau, voire être l'objet de mécanismes de gestion dédiés. Bien que nous privilégions des approches agnostiques aux procoles utilisés, il est nécessaire de présenter le fonctionnement de ce protocole afin de mieux comprendre ensuite la performance réalisée par les flots élastiques.

TCP a pour but d'établir, de manière décentralisée, une allocation efficace et équitable de la bande passante disponible sur les liens traversés. Le contrôle de congestion qu'il réalise détecte les événements de congestion dans le réseau et adapte en fonction le débit d'envoi des paquets. Il a été proposé dans l'article séminal de Van Jacobson [75] afin de faire face à la saturation du réseau (*congestion collapse*). Cette version originale de TCP (nommée TCP Tahoe), ainsi que les améliorations de l'algorithme de contrôle de congestion qui ont suivi (Reno et NewReno) s'appuient sur la détection des pertes de paquets. Elles se produisent lorsque les connexions TCP provoquent la saturation du buffer d'un des routeurs. Une alternative utilisée par TCP Vegas [37] est d'utiliser une estimation du délai subi par les paquets lors de la traversée des différentes files d'attente. Dans la suite, sauf mention contraire, nous réfèrerons implicitement à la version TCP NewReno.

Chaque paquet envoyé par TCP doit être acquitté par le récepteur. Le protocole maintient une fenêtre de congestion (*cwnd*), qui représente le nombre de paquets non encore acquittés, et qui permet de réguler le flux d'envoi. Le contrôle de congestion d'une connexion active (lorsque TCP est dans un mode dit *congestion avoidance*) repose sur l'algorithme AIMD (Additive Increase Multiplicative Decrease) qui gère l'évolution de *cwnd* : sa valeur est augmentée d'un pour chaque *cwnd* paquets acquittés, et diminuée de moitié lorsqu'une congestion est détectée. TCP possède d'autres modes de fonctionnement : *slow-start* intervient au début d'une connexion, afin de rapidement approcher le débit équitable ; *fast recovery* et *fast-retransmit* ont été proposées dans TCP Reno et NewReno afin de mieux

gérer les événements de congestion. Padhye *et al.* [120] ont établi la relation entre le débit atteint par une connexion TCP en fonction du taux de perte subit.

2.1.4 Limites de TCP

L'allocation équitable réalisée par TCP dépend de la coopération des sources [109]. Elle est ainsi vulnérable à celles n'implémentant pas TCP, parfois dans un but malicieux [1].

L'algorithme AIMD, sur lequel repose l'équité, souffre également de plusieurs imperfections. Il est par exemple inefficace sur des liens possédant un produit délai x bande passante élevé, et nécessite des buffers suffisamment dimensionnés (cette problématique sera abordée plus en détails dans le Chapitre 4). Plusieurs propositions de protocoles "haut débit" permettent à plusieurs flots longs en compétition d'obtenir de bonnes performances, avec un certain degré d'équité. Mais comme nous le verrons dans le Chapitre 5, elles sont souvent plus agressives et ne permettent pas une coexistence équitable avec les connexions TCP Reno, ce qui est une propriété recherchée (on parle de protocoles *TCP friendly*).

Le contrôle de *cwnd* en fonction des pertes subies par les connexions est également la cause d'autres imperfections de TCP. D'une part, il ne s'agit que d'une vision limitée des capacités du réseau, uniquement aux instants de congestion (action corrective). D'autre part, une perte de paquet ne signale pas nécessairement une congestion. Les réseaux optiques possèdent par exemple un taux d'erreur qui génèrera des pertes, et la réduction de *cwnd* et donc de débit ainsi causée ne sera compensée que tardivement, à cause de la lenteur de AIMD sur un lien à fort débit. Enfin, lors des événements de saturation, de nombreuses connexions TCP réduisent leur *cwnd* en même temps, ce qui entraîne une moins bonne utilisation des ressources du lien.

Nous remarquons également que les débits d'accès des utilisateurs sont aujourd'hui encore relativement faibles par rapport aux capacités de cœur de réseau. Ainsi le goulot d'étranglement se situe généralement dans le réseau d'accès, et les mécanismes de TCP n'interviennent pas dans le cœur de réseau qui reste transparent, sauf pour quelques flots bien particuliers. Toutefois, l'augmentation des débits d'accès avec l'arrivée de la fibre optique pourrait déplacer ce problème plus en amont.

2.1.5 Vers un déploiement d'architectures de qualité de service ?

La présence de nouvelles applications du réseau, comme la téléphonie ou la vidéo, requiert d'importantes garanties de service pour lesquelles le modèle Best Effort actuel, reposant majoritairement sur TCP, semble insuffisant. Un certain nombre de propositions mettant en œuvre de tels mécanismes ont été proposées afin d'apporter des garanties de service dans le réseau. Pourtant, les opérateurs sont toujours hésitants à introduire des mécanismes de gestion de trafic plus avancés. Le manque de preuves concernant l'efficacité de telles solutions, auquel s'ajoute leur complexité de mise-en-œuvre en est souvent la cause. Au lieu de cela, les opérateurs appliquent un surdimensionnement systématique des ressources afin de garantir une performance raisonnable lorsque le réseau n'est pas fortement chargé. Un dimensionnement qui assure une charge des liens inférieure à 40% est par exemple appliqué, pour faire face à une éventuelle augmentation de la charge en cas de panne et re-routage du trafic sur un autre lien.

Une telle gestion requiert des opérateurs des investissements réguliers coûteux, et semble être de plus en plus remise en cause avec la montée des applications gourmandes en ressources, comme la vidéo ou le pair-à-pair. On voit ainsi de nombreux réseaux déployer des mécanismes complexes d'inspection de trafic et de bridage ad-hoc. L'augmentation des capacités dans les réseaux d'accès (fibre optique, etc.) pourrait également être une motivation suffisante pour reconsidérer le déploiement de mécanismes de gestion de trafic. D'autant que certains mécanismes peuvent, comme nous le verrons, apporter une performance prévisible, présentant un avantage compétitif pour un opérateur et lui simplifiant sa tâche de dimensionnement.

2.2 Garanties de service pour les flots IP

Nous commençons par présenter un certain nombre de notions générales qui nous semblent utiles pour la compréhension du trafic et l'évaluation de sa performance, pour ensuite établir un ensemble de critères nous permettant de comparer les architectures existantes de garantie de service.

1. Par exemple, en ne réduisant par son débit d'envoi lors d'une congestion, une connexion peut bénéficier d'un débit plus élevé que les autres connexions partageant le lien qui réduisent leur débit.

2.2.1 Granularité du trafic IP

La modélisation du trafic peut être réalisée à différents niveaux de granularité, représentés sur la figure 2.1.

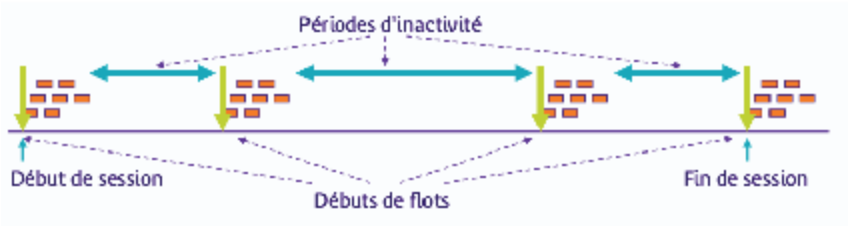


FIGURE 2.1 – Représentation des différentes échelles de trafic.

Le *paquet* IP est l'unité élémentaire traitée par la couche réseau. L'échelle de temps correspondante est de l'ordre de la microseconde ou de la milliseconde en fonction du débit des liens considérés. L'étude du trafic à cette échelle est rendu complexe par sa propriété d'autosimilarité [98]. Elle résulte de la distribution de la taille des flots, qui possède une variance infinie, ainsi que des rafales de paquets induites par TCP [124].

Le *flot* consiste en un ensemble de paquets groupés dans le temps, appartenant à une même application. Il correspond typiquement au transfert d'un objet numérique sur le réseau, dont la performance sera ressentie par l'utilisateur (par exemple son temps de transfert). En pratique, l'identification des flots dans le réseau repose sur le fait que les paquets ont des valeurs communes pour certains champs de leur en-tête IP et TCP/UDP[2], et sont groupés dans le temps.

Les flots se présentent au réseau en *sessions*. Il s'agit d'une succession de flots entrecoupés de temps de réflexion (*think times*) de l'ordre de quelques secondes, et reflétant l'activité d'un utilisateur (session de navigation, e-commerce, etc.). Les sessions sont en pratique plus difficiles à identifier que les flots.

2.2.2 2 classes d'applications : S et E

En dépit de la diversité des applications transportées par le réseau, il est possible d'en distinguer deux groupes ayant des propriétés et des besoins de garanties de qualité de service différents [140, 131].

flots élastiques : ils sont induits par le transfert de documents numériques (pages web, fichiers, ...), et utilisent généralement la couche de transport TCP. Ils représentent la majeure proportion du trafic Internet. Leur dénomination vient du fait que leur débit s'adapte aux ressources disponibles dans le réseau. Le maintien d'un débit moyen à long terme est suffisant pour garantir un temps de réponse satisfaisant pour l'utilisateur.

flots streaming : ils sont produits par les applications audio et vidéo, et sont typiquement transportés sur UDP. Ils sont caractérisés par leur durée intrinsèque (le signal à transmettre). Certaines applications sont également caractérisées par leur débit, alors que d'autres sont adaptatives en fonction du niveau de congestion rencontré, qui détermine la qualité reçue. Contrairement aux flots élastiques qui ne requièrent qu'une garantie de service à long terme, leur qualité de service est ici affectée par le débit instantané reçu. Ils peuvent également nécessiter, au niveau paquet, de faibles délais ainsi qu'un taux de pertes négligeable.

2.2.3 Régimes opérationnels des liens

Afin de comprendre la performance réalisée par les flux, il est utile de distinguer trois régimes de partage de bande passante, représentés en figure 2.2. Dans le dessin du haut, les flots sont représentés par des boites de hauteur égale à leur débit exogène, et dont la position horizontale est déterminée par le temps de début et la durée du flot. La position verticale de la boîte sur le lien est choisie aléatoirement. Au dessous, on représente l'évolution dans le temps du débit global des flots en cours.

2. Ce sont le protocole de transport et l'adresse IP accompagnés soit des ports TCP/UDP pour IPv4, soit du *flow label* pour IPv6

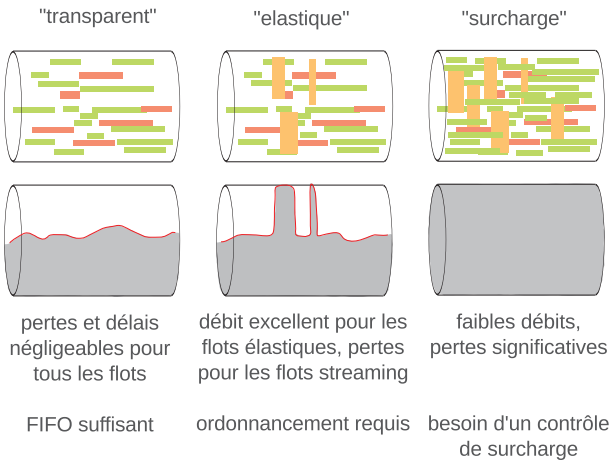


FIGURE 2.2 – Trois régimes d'opération du lien : "transparent", "élastique" et "surcharge"

Le cas de gauche transcrit un régime "transparent" dans lequel tous les flots ont un débit limité et la somme de leurs débits reste inférieure à la capacité du lien (avec une forte probabilité). La limite en débit peut être imposée par un lien d'accès, ou encore par l'allocation faite sur un autre lien. Les pertes de paquets ainsi que les délais sont alors très faibles. C'est le régime dans lequel se trouvent la plupart des liens du réseau Internet actuel, en raison du surprovisionnement effectué par les opérateurs et des débits d'accès relativement faibles de la plupart des utilisateurs.

Le régime "élastique" représenté au milieu de la figure se produit lorsque des flots au débit potentiellement non limité s'ajoutent au trafic généré par les flots goulottés. Une telle situation se produit même lorsque la charge globale est inférieure à la capacité du lien. Ce sera typiquement le cas d'un lien d'accès. Il est également possible que des flots limités en débit se combinent pour saturer les ressources du lien : par exemple dans un réseau de collecte, où les flots sont limités par le débit d'accès de l'utilisateur, voire dans le cœur de réseau avec l'arrivée des technologies à très haut débit comme la fibre optique. Lors des instants de congestion, les buffers des liens se remplissent et débordent, causant des délais et des pertes impactant notamment les flots *streaming*. Les délais peuvent être significatifs si la taille des buffers est importante. En plus d'un dimensionnement adéquat des ressources, il semble nécessaire d'utiliser un mécanisme d'ordonnancement ou de gestion de la file d'attente afin de préserver la QoS des flots dans ce régime (eg. différenciation). Notons que les liens de peering entre opérateurs réseaux, ou que les liens d'interconnexion d'un *datacenter* sont typiquement en régime élastique.

Le cas de droite correspond à un régime de "surcharge" où l'offre de trafic (produit du taux d'arrivée des flots et de la taille moyenne d'un flot) dépasse la capacité disponible sur le lien. Dans ces conditions, le nombre de flots augmente très rapidement et leur débit tend alors vers zéro. Les algorithmes de contrôle de congestion tels que ceux de TCP sont incapables de gérer ce genre de congestion. Il en résulte une QoS dégradée pour tous les utilisateurs, uniquement stabilisée par l'abandon des connexions. Des mécanismes de gestion de cette surcharge tels qu'un contrôle d'admission sont nécessaires afin d'éviter une telle configuration.

2.3 Techniques d'amélioration de la QoS

Les insuffisances présentées plus haut de la situation actuelle ont donné lieu à un ensemble de propositions. Celles-ci reposent sur l'extension des mécanismes en bordure de réseau, comme le développement de protocoles TCP plus adaptés, et/ou l'intégration au réseau de mécanismes avancés de gestion du trafic.

Le développement de nouveaux protocoles de transport n'étant pas l'objet de notre étude, nous n'entrerons pas dans trop de détails, au delà de la description faite à l'occasion du Chapitre 5. Il est toutefois difficile, comme le souligne la RFC 2309 [35], de contrôler dynamiquement le trafic depuis la bordure de réseau, sans indication supplémentaire que les pertes de paquets. Ce document recommande ainsi l'introduction de mécanismes d'ordonnancement et de contrôle des files d'attente dans le réseau (AQM, introduits ci-dessous).

Nous passons en revue ici de telles contributions, avant de nous intéresser à des mécanismes plus avancés tels que le contrôle de la surcharge ou la différenciation de trafic. Leur analyse nous permettra de mieux comprendre la performance des architectures plus complètes qui seront étudiées à la fin de ce chapitre.

2.3.1 Allocation de ressources et contrôle de congestion

Gestion des files d'attente

L'ajout de mécanismes de gestion des files d'attente au sein des routeurs (*Active Queue Management*, AQM) représente un complément au contrôle de congestion de bout en bout [53]. Ces mécanismes permettent de gérer la taille de la file d'attente en éliminant des paquets si nécessaire. Leur vision locale du trafic permet d'informer de manière plus ou moins complexe les protocoles de transport de l'état de congestion du lien, et de remédier aux problèmes dûs à la politique *DropTail*. On retrouve ici les éléments de la terminologie ATM : *congestion-indication* et *explicit-rate* [132].

Exemple d'AQM : prévention de la congestion

RED [57], l'AQM le plus connu, rejette les paquets de manière probabiliste en fonction de la longueur de la file d'attente, afin de forcer TCP à adapter son débit. La variante *Weighted RED* (WRED) permet de définir plusieurs classes de trafic en fixant le seuil de rejet. La probabilité de rejet ne dépend pas de l'activité des flots et souffre ainsi de problèmes d'équité. De plus, RED ne permet pas d'éliminer les pertes de paquets.

Feng *et al.* [51] suggèrent que la taille de la file d'attente ne permet pas d'estimer correctement le niveau de congestion, puisque le trafic n'est pas de type Poisson [126, 98]. Leur proposition, BLUE, se base sur les taux d'inactivité du lien et de pertes de paquets. L'idée de découpler la mesure de congestion de la taille de la file d'attente est également présente dans REM [97], PI [72, 71] et AVQ [144].

Malgré les propriétés d'ajustement automatique des algorithmes plus récents tels que BLUE, ARED [58] ou PI, la plupart des AQM sont complexes à paramétrer et n'offrent pas une performance prévisible.

Marquage de paquets

Afin d'éviter le rejet d'un paquet, qui gaspille les ressources du réseau, Floyd [56, 130] propose d'ajouter un ou plusieurs bits ECN (*Explicit Congestion Notification*) dans les entêtes TCP/IP des paquets, qui indiquent un état de congestion imminent. Cette modification préventive nécessite une standardisation afin d'obtenir la coopération des routeurs intermédiaires, notamment entre plusieurs domaines administratifs, ainsi que des protocoles de transport à chaque bout de la connexion.

Flots non-coopératifs

La plupart de ces mécanismes reposent sur une collaboration des utilisateurs, et ne prennent en compte ni les protocoles autres que TCP, ni les flots malveillants. La difficulté principale est d'éviter de maintenir un état pour chaque flot en cours, ce qui poserait des problèmes de passage à l'échelle pour des liens de cœur de réseau.

RED with penalty box [106] détermine les flots non-coopératifs à partir d'un historique des derniers événements de perte, et leur impose une limite en débit (CBQ [54]). Flow RED [101] se base sur la place occupée par chaque flot dans le buffer. Stabilized RED [117] estime le nombre de connexions actives. SFB [51] maintient une probabilité de rejet pour les flots de manière compacte grâce à l'utilisation des filtres de Bloom, sur le même principe que BLUE (taux de perte de paquets et utilisation du lien).

CHOKe [121] utilise le contenu du buffer en y choisissant un paquet tiré au hasard ; si ce paquet appartient au même flot que le paquet entrant, les deux sont rejetés. L'algorithme AFD [122] maintient une structure de données annexe afin d'estimer le débit des flots qui émettent au dessus du débit équitable, et d'adapter la probabilité de rejet en conséquence.

Pour les algorithmes utilisant le contenu de la file d'attente, la question de leur sensibilité à la taille de cette dernière reste à déterminer. Le paramétrage reste problématique dans tous les cas et consiste souvent en des règles empiriques.

Solutions mixtes transport/AQM

Plutôt que de signaler une congestion, XCP [81] propose aux sources une évolution incrémentale de *cwnd* sur plusieurs RTT[3], qui remplace ainsi la brique AIMD de TCP. L'évolution lente de XCP, ou des protocoles basés sur AIMD, n'offre pas une performance satisfaisante pour les flots courts, qui représentent la majorité du trafic. RCP [47] propose d'envoyer explicitement aux sources des flots un débit à réaliser afin d'émuler un comportement *Processor Sharing* (PS). Ce débit est un débit équitable calculé périodiquement par le routeur à partir du débit des flots et de la taille de la file d'attente. Un

3. *Round Trip Time*, une mesure du délai aller-retour entre deux hôtes

inconvénient sérieux de propositions telles que XCP et RCP est de requérir la coopération de l'ensemble des routeurs du réseau.

2.3.2 Ordonnancement

Differents types d'ordonnancement

L'ordonnancement au sein des routeurs permet également de réguler les flux de paquets en choisissant lequel sera envoyé prochainement. L'allocation est effectuée par le routeur, ce qui permet de gérer plus simplement les flots non-réactifs, et ainsi de ne pas dépendre du protocole de transport, de sa rapidité à s'adapter, voire de sa coopération. Il existe de nombreuses alternatives au schéma FIFO, qui consistent généralement au maintien de plusieurs files d'attentes (virtuelles ou non). Cette classification peut se faire par flot, ou par classes.

L'émission d'un paquet des différentes files d'attente se fait soit en priorité – on parle de *Priority Queueing* (PQ) – ou en fonction de mécanismes tels que RR (*Round Robin*) [160], WRR (*Weighted Round Robin*) [157] ou DRR (*Deficit Round Robin*) [155], qui réalisent des mécanismes de tourniquets entre les files d'attente afin qu'aucune ne monopolise les ressources du lien. L'ordonnancement idéal est souvent considéré être de type *fair queueing* (FQ) [156], qui consiste à attribuer le même débit à chaque flot, et il en existe de nombreuses implémentations.

Fair Queueing

FQ réalise un partage *max-min* de la bande passante et présente de nombreux avantages d'isolation et d'équité entre les flots. Il permet notamment de laisser au protocole de transport la liberté d'exploiter au mieux les ressources qui lui sont allouées sans gêne pour les autres flots.

Une vision plus détaillée des avantages présentés par l'introduction de FQ sera présentée dans le Chapitre 5, où nous étudierons les interactions entre les protocoles et FQ. Ses propriétés de différentiation sont également exploitées, et notamment dans la proposition de contrôle d'admission pour l'architecture FAN que nous présentons dans le Chapitre 3. Enfin, Keshav [87] montre comment il est possible d'exploiter dans le protocole de transport l'information implicitement fournie par l'ordonnanceur qui émet les flots au débit équitable.

FQ n'est souvent pas considéré comme une alternative réalisable du fait qu'il nécessite de maintenir une file d'attente par flot. Plusieurs propositions permettent d'obtenir un algorithme approximatif [107, 145] mais elles restent encore à étudier plus profondément. Nous verrons dans le Chapitre 3 une proposition d'un algorithme réalisable et extensible de FQ, appelée PFQ (*Priority Fair Queueing*), qui a été faite préalablement au début de cette thèse, dans le cadre de l'architecture *Flow-Aware Networking*.

Equité des débits

L'équité en débit n'est pas nécessairement un objectif absolu. Elle est d'ailleurs remise en cause dans [40], au profit d'une égalité de la congestion causée dans le réseau. Annoncée comme plus juste pour un utilisateur, elle permet notamment d'éviter le recours à plusieurs flots parallèles afin d'augmenter la quantité de bande passante reçue. Une telle architecture est présentée plus loin dans la Section 2.4.1.

SRPT [69] (*Shortest Remaining Processing Time*) est un ordonnancement préemptif qui donne la priorité aux documents les plus courts : le serveur est supposé connaître le volume de données restant à être transféré de chaque document, et dédie la totalité de sa capacité à servir celui représentant le minimum. La performance des flots longs est en effet relativement indépendante de celle réalisée par les flots courts, il est ainsi intéressant de leur donner priorité : [69, 10]. Cette discipline de service est connue pour améliorer le temps de service des documents, notamment lorsque leur taille possède une distribution à queue lourde (eg. [132]). Si cet ordonnancement semble améliorer globalement la performance, sa mise en œuvre reste cependant complexe, et il conviendrait de déterminer plus précisent son comportement au sein du réseau.

D'autres propositions existent, que nous n'avons pas prétention à énumérer. Nous verrons cependant dans le prochain chapitre qu'une telle équité des débits est fortement recommendée et nous permet le développement de modèles de performance simples et robustes.

2.3.3 Contrôle de la surcharge

Impact de la surcharge

Le contrôle de congestion effectué par TCP (ou par un mécanisme au niveau paquet au sein d'un routeur) n'empêche toutefois pas le réseau d'entrer en situation de surcharge, lorsque la demande en débit dépasse la capacité. Le débit individuel des flots décroit vers zéro, ce qui entraine une accumulation des connexions et rend le système instable : le temps moyen de transfert des flots tend vers l'infini. Notons qu'il s'agit d'une instabilité au niveau flot, puisque les tailles finies des buffers causent des pertes qui stabilisent le système au niveau paquet.

En réalité, ce phénomène est mitigé par les *timeouts* subis par les connexions TCP, ainsi que par un phénomène d'impatience de la part des utilisateurs ou applications. Les conséquences d'une telle surcharge, due aux retransmissions et aux transferts abandonnés, sont présentées dans [137, 34].

Contrôle d'admission

S'il est parfois contesté dans le cœur de réseau, le besoin d'un contrôle d'admission [133, 104] est généralement reconnu, notamment pour protéger la performance des flots *streaming*. Shenker et al [140] l'illustrent par exemple au travers de fonctions d'utilité : il existe un débit minimal en-dessous duquel le flot n'est pas utile. Roberts et al. [137, 14, 18, 20] le recommandent également pour les flots élastiques (surtout en cas de tarification à l'usage [135]). Un tel contrôle d'admission peut être réalisé de différentes manières, reposant sur une signalisation du trafic, ou sur des mesures effectuées localement (on parle de contrôle d'admission implicite).

La notion de contrôle d'admission est souvent mal perçue, notamment dans le cœur de réseau, du fait qu'il est difficile de justifier à un utilisateur du rejet d'un flot. Il faut bien préciser que de tels mécanismes supposent un dimensionnement satisfaisant au préalable par l'opérateur, qui analyse le trafic passé et tente de prévoir les évolutions possibles dans le futur, les cas de panne, etc. Les mécanismes de QoS n'ont pas pour ambition de s'y substituer, mais plutôt de gérer des situations exceptionnelles de surcharge, comme une panne suivie d'un re-routage de trafic, où la performance est généralement dégradée de manière significative (phénomènes de *flash crowd*).

Autres mécanismes

Le contrôle d'admission est transparent en ce qu'il peut faire partie intégrante des autres mécanismes de gestion de la surcharge : multi-chemins et routage adaptatif [119], ou encore équilibrage de charge [27].

Il est parfois possible que le lien devienne surchargé malgré l'utilisation d'un contrôle d'admission (suite à des erreurs de mesure, un changement de profil des flots acceptés, etc.). Nous en verrons un exemple dans le Chapitre 6 lors de situations de *flash crowd*. Il convient alors de considérer des mécanismes de préemption de flot, qui consistent à interrompre un certain nombre de flots en cours.

2.3.4 Différenciation et classes de trafic

Une autre approche consiste à grouper les flots en différentes classes (CBQ, *Class-Based Queueing*) [154] et leur offrir un traitement différencié à des fins de tarifications, ou encore pour augmenter l'utilité du réseau. Comme nous le verrons dans le Chapitre 7, il peut être nécessaire de distinguer plusieurs classes de service dans le réseau d'accès ou de collecte, afin de protéger les flux de voix ou les flux TV à fort débit par exemple [25].

Dans le réseau de cœur, il est possible d'exploiter les différentes contraintes de QoS entre les flots *streaming* (faibles délais et pertes) et élastiques (débit moyen minimal) en servant les premiers en priorité, sans affecter les derniers. Dans un tel contexte, Bonald et Massoulié [28] exhibent des configurations de réseaux instables même lorsque le système n'est pas saturé ; un mécanisme de contrôle de la surcharge est alors d'autant plus nécessaire.

La classification du trafic permet d'offrir différents niveaux de transparence pour chaque type de flot (en termes de délais ou de taux de pertes pour les flots *streaming*, ou de bande passante pour les flots élastiques). Toutefois, Benameur *et al.* [20] suggèrent que ce type de différenciation n'est pas efficace, puisque la QoS effectivement ressentie par les flots est soit excellente, soit très mauvaise et il est difficile de définir plusieurs niveaux intermédiaires. La différence est visible en surcharge uniquement, et n'est pas manifeste si les flots ont un débit d'accès limité. Roberts *et al* [131] montrent que l'attribution de poids aux flots élastiques (modèle *Discriminate Processor Sharing*) est peu intéressante et rend leur performnce sensible à la distribution de leur taille. Il en est de même pour les flots *streaming*, et il devient difficile de prédire la performance obtenue.

Comme nous l'avons vu précédemment, l'utilisation d'une discipline de service SRPT peut être intéressante. Elle revient à différencier les flots en fonction de leur taille. Bansal *et al.* [11] montrent que cet ordonnancement est plus efficace que *Processor Sharing*. Le risque de famine des flots longs nécessite cependant un contrôle de la charge des autres flots, même si cet effet est minimisé pour des distributions de tailles de flots à queue lourde.

L'usage d'un contrôle d'admission permet en outre de garantir une bonne qualité de service pour l'ensemble des flots acceptés, et ainsi de reporter l'impact de la congestion sur un petit nombre de flots rejetés. On réalise ainsi une différenciation par la disponibilité du réseau [17] (qui peut être également déclinée en plusieurs classes).

De nombreuses méthodes existent pour identifier les flots à des fins de différenciation. La plus connue est certainement la méthode DPI (*Deep Packet Inspective*) qui inspecte le contenu de chaque paquet jusqu'au niveau applicatif éventuellement. Elle est généralement utilisée afin d'identifier une application. Il est également possible de distinguer le trafic en fonction de sa source ou de sa destination, ou encore de reposer sur une signalisation et la vérification a posteriori de sa conformité.

De tels procédés sont généralement soustraits à une tarification différenciée, et contraires à la neutralité des réseaux. Des approches telles que SRPT (sur la taille des flots), ou la différenciation faite par le *fair queueing* (sur leur débit, voir Chapitres 3 et 5) sont au contraire compatibles avec cette neutralité.

2.4 Analyse des principales architectures de qualité de service

Devant la nécessité d'offrir des garanties de service au trafic, et face aux insuffisances d'un réseau *Best-Effort*, un certain nombre d'architectures ont été proposées combinant un ensemble de mécanismes présentés dans la section précédente. Nous avons volontairement limité notre présentation à celles que nous jugeons les plus importantes, ou qui présentent des caractéristiques intéressantes afin de positionner l'approche Flow-Aware Networking (FAN) qui est l'objet de cette thèse. Ce sont : (1) IntServ, (2) DiffServ, (3) Flow-Aware Diffserv, (4) DPS/SCORE, (5) Flow-State Aware et variantes, (6) Congestion Exposure, et enfin (7) Flow-Aware Networking.

Nous présentons d'abord succinctement les points importants de chacune de ces architectures, avant de procéder à leur analyse. Pour cela, nous nous basons sur notre compréhension du trafic et de ses besoins en termes de performance tels que présentés plus haut dans ce chapitre. Nous procédons à une analyse systématique des fonctionnalités supportées afin de fournir des garanties pour les différents types de flots, ainsi que des mécanismes mis-en-œuvre pour leur réalisation.

Il est intéressant de remarquer qu'un grand nombre de ces propositions, comme FAN, considèrent le trafic, ou du moins la performance, au niveau d'un flot [4]. Joung [79] et Wojcik [164] font également une analyse de ces approches, mais ils se limitent à un petit nombre d'aspects tels que la définition des flots ou des classes de service, le contrôle d'admission, la gestion des files et la signalisation. Nous renvoyons le lecteur à ces références pour plus de détails.

2.4.1 Présentation des principales architectures

Intserv

IntServ [165] est l'une des premières architectures proposées afin de fournir des garanties de service pour les flux IP. Elle repose sur le protocole RSVP [36], qui permet pour chacun de réserver des ressources réseau (bande passante et buffer) au sein des routeurs successifs. Trois principales classes de trafic sont définies et la plus prioritaire permet d'assurer un débit (déterministe) pour les flots *streaming*. Sans le spécifier, IntServ recommande un contrôle d'admission pour les classes les plus prioritaires, et nécessite l'introduction d'un ordonnancement approprié (WFQ/PQ). Un mécanisme de préemption est également prévu.

Intserv nécessite le support de l'ensemble des routeurs de bout en bout. Le besoin d'établir une signalisation pour chaque flot rend difficile le passage à l'échelle hors d'un réseau d'accès, notamment lorsqu'un grand nombre de flots est en compétition. De plus, la caractérisation du trafic lors de la réservation, ainsi que la vérification de sa conformité restent problématiques.

DiffServ

DiffServ [67] est proposé comme une alternative plus extensible à IntServ, où le contrôle individuel des flots (par exemple par RSVP) n'est effectué qu'en bordure de réseau. Le reste du temps, chaque flot

4. la définition d'un flot est parfois plus floue, ou considère quelques champs supplémentaires de l'en-tête IP, sans toutefois affecter les remarques faites ici

est associé à un agrégat au travers d'une marque faite dans l'en-tête IP du paquet [5]. Il suffit alors à un routeur d'inspecter la marque apposée pour classifier le paquet et lui offrir un traitement différencié. Trois classes principales sont prévues (éventuellement subdivisées), correspondant à celles de IntServ.

DiffServ convient dans une certaine mesure pour un réseau d'opérateur qui souhaite proposer à ses clients des services à valeur ajoutée (Triple Play [6] par exemple), puisqu'à la fois les serveurs et les clients sont situés dans le même domaine.

L'évaluation de la performance offerte aux flux *streaming* en fonction des ressources réservées à la classe prioritaire est cependant difficile, et ce notamment dans un contexte inter-domaine où chaque opérateur peut implémenter et configurer DiffServ différemment. En outre, le manque d'un contrôle individuel des flots au sein du réseau fait qu'il est difficile d'offrir des garanties aux flux sauf à appliquer un facteur de surdimensionnement. La performance requiert une bonne connaissance de la matrice de trafic entre les différents points du réseau, et est ainsi sensible aux changements de routage.

Nous verrons dans le chapitre suivant que la distinction d'un grand nombre de classes de trafic n'est pas utile puisque la performance est soit très bonne si le lien n'est pas en surcharge, soit très mauvaise si le lien est en surcharge : l'ensemble des flots des classes les moins prioritaires subissent alors la congestion.

Flow-Aware DiffServ et autres extensions

Une faiblesse supplémentaire de l'architecture DiffServ est que le taux de service de chaque classe n'est pas proportionnel au nombre de flots en cours. Il est ainsi possible que des classes prioritaires peu chargées offrent un meilleur service que les classes plus prioritaires. Li *et al.* [99] abordent ce problème en introduisant des estimateurs permettant d'adapter dynamiquement le taux de service au nombre de flots en cours dans chaque classe.

Un certain nombre d'autres propositions étendent DiffServ afin de mieux gérer des situations de surcharge, et ainsi moins reposer sur un surdimensionnement des liens.

Pre-Congestion Notification

Un exemple est PCN (*Pre-Congestion Notification*), qui repose sur l'ajout de marques de congestion au sein du domaine Diffserv, afin que les nœuds de bordure puissent décider du contrôle des flots. Ce mécanisme s'applique aux classes les plus prioritaires de DiffServ et permet la réalisation d'un contrôle d'admission et d'un processus de terminaison de flots.

Un tel déploiement, *Controlled Load* [41], propose d'appliquer deux marques différentes en fonction de la taille atteinte par la file d'attente. Les paquets reçoivent une marque de pré-congestion lorsqu'un seuil d'admission est dépassé (par exemple par un mécanisme de type ECN sur une file virtuelle de capacité réduite). Ces signaux sont agrégés afin de décider de l'admission des nouveaux flots. Lorsqu'un second seuil est dépassé, les paquets reçoivent une marque de congestion, qui permettra de décider de la préemption éventuelle de certains flots. Les auteurs de [44] introduisent une alternative n'utilisant qu'un seul type de marquage pour définir les seuils d'admission et de préemption de flots.

Ces mécanismes nécessitent de standardiser à la fois les conditions de marquage et de congestion. Les mécanismes de marquage reposent généralement sur des files virtuelles avec RED, ou sur des seaux à jetons (*token bucket*) [161], dont la performance est sensible à des caractéristiques détaillées sur le trafic. Il est de plus difficile de prédire la performance réalisée en fonction des seuils de marquage, d'admission ou de préemption.

Stateless CORE & Dynamic Packet State

L'approche SCORE (*Stateless CORE*) propose un compromis entre une architecture où chaque routeur doit garder un état par flot (comme IntServ), et une architecture sans état (comme DiffServ). Les routeurs de bordure reconnaissent les flots et effectuent les traitements les plus complexes. L'information nécessaire est alors encodée au sein des paquets afin de permettre aux routeurs de cœur de prendre les décisions appropriées (*Dynamic Packet State*). Nous présentons deux réalisations de cette architecture : CSFQ et CJVC.

CSFQ (*Core Stateless Fair Queueing*) [145] consiste a émuler un réseau *fair queueing* en insérant des estimations du débit des flots dans les en-têtes de paquets, afin d'adresser les problèmes d'extensibilité de DRR tout en offrant une performance similaire. La tâche des routeurs de cœur consiste à évaluer le débit équitable sur le lien, et rejeter les paquets entrant de manière probabiliste en fonction du ratio entre le débit du flot encodé dans le paquet, et ce débit équitable.

5. réutilisation du champ ToS, dénommé DSCP dans DiffServ
6. Ce terme dénote les offres fournissant à la fois des services Internet, de TV et de téléphonie

CJVC (*Core Jitter Virtual Clock*) [82] est une autre adaptation des mêmes mécanismes afin d'offrir des garanties de service par flot (délai et bande passante) en implémentant un mécanisme de contrôle d'admission. Il s'agit de l'adaptation dans l'architecture SCORE de *Jitter Virtual Clock* qui émule le fonctionnement de IntServ.

Au delà des modifications nécessaires dans les en-têtes de paquets, il n'est pas clair comment ces différents algorithmes peuvent être combinés pour fournir une architecture réalisant à la fois CSFQ et CJVC par exemple. La robustesse d'une telle architecture dépend fortement du routeur de bordure qui effectue l'encodage des paquets. Il est proposé que ce soient les utilisateurs eux-mêmes qui marquent les paquets : une vérification statistique peut alors être utilisée par les routeurs de bordure, avec isolation des hôtes malveillants, mais cela reste un mécanisme complexe à réaliser.

Flow Routing

Les sociétés Caspian et Anagran ont commercialisé des routeurs travaillant au niveau flot. La contribution essentielle est une technologie permettant de maintenir en mémoire l'état de l'ensemble des flots en transit (rendue possible d'après eux par la réduction du coût des mémoires). Cela leur permet de réaliser un routage par flot, où la décision de routage est prise uniquement pour le premier paquet. Un tel comportement serait moins gourmand en temps processeur que l'approche classique qui inspecte chaque paquet.

Ces solutions prouvent la faisabilité de maintenir une table de flots au sein du routeur, et ont permis la proposition de l'architecture Flow-State Aware, présentée ci-après.

Flow-State Aware

L'architecture Flow-State Aware (FSA) préconise un ensemble de classes de trafic qu'un réseau devrait supporter, et a fait l'objet d'une recommandation UIT [1].

Si la plupart restent classiques, on remarquera la classe *Conditionally Dedicated Bandwidth* (CDBW). CDBW propose une variante moins stricte de contrôle d'admission qui permet de concentrer les pertes de paquets en cas de congestion sur un petit ensemble de flots [112], par exemple les flots dernièrement admis. FSA réalise cette fonctionnalité à l'aide d'un indicateur de priorité à deux états (*discard first* et *discard last*), qui détermine un ensemble de flots qui subira la congestion en premier. On peut les voir comme une relaxation de la notion de rejet ou d'admission.

FSA offre ainsi certaines classes de trafic pour lesquelles un flot peut démarrer immédiatement, sans attendre de décision du réseau, en étant admis conditionnellement. Il pourra ensuite évoluer vers un statut plus prioritaire. Ce mécanisme ressemble au *cell loss priority tagging* dans ATM [143]. Cette approche permet également de ne pas nécessiter de réservation de ressources pour les classes autres que *Guaranteed Rate* et ainsi d'obtenir une utilisation plus efficace des ressources du lien.

FSA souligne l'importance de gérer la qualité de service du trafic au niveau des flots individuels, mais recommande l'utilisation de mécanismes DiffServ pour le cœur de réseau, à des fins de passage à l'échelle : l'état d'un agrégat[7] contient alors la somme des caractéristiques individuelles des flots.

Alors que la spécification n'impose aucun type de signalisation, on peut considérer l'usage de la solution Flow State présenté ci-dessus, qui repose sur une signalisation intra-bande légère [113]. Cela permet une réalisation plus extensible que RSVP par exemple, mais nécessite en contrepartie un traitement matériel de la structure de QoS annoncé dans l'en-tête du paquet, afin de tenir les haut débits.

La plupart des classes de trafic reposent également sur des profils de flots, et la recommandation n'indique pas comment vérifier leur conformité, ce qui reste encore un problème complexe. Le problème de l'inter-domaine persiste également.

Congestion Exposure

Congestion Exposure est une architecture en cours de standardisation à l'IETF, qui consiste à rendre visible pour le réseau la congestion engendrée par les flots. Elle est illustrée par la proposition *re-feedback* qui force la source à annoncer au sein de ses paquets le taux de congestion qu'elle pense causer au réseau, par exemple au travers du bit ECN [159] des en-têtes TCP et IP (on parle de *re-ECN*).

Les routeurs détectent le niveau de congestion, par l'utilisation de mécanismes de contrôle de la file d'attente, et positionnent des marques ECN dans les paquets. Les récepteurs répètent cette marque dans le champ IP, qui est alors vue comme une dette pour l'émetteur qui doit alors puiser dans son crédit pour marquer également les paquets qu'il envoie. Les routeurs de bordure proches de l'émetteur

7. un ensemble de flots groupés dans la même classe, au sens DiffServ

peuvent alors rejeter les paquets dont la balance de crédit est en déficit par rapport aux marques de congestion reçues.

Dans cette architecture, aucune notion de flot n'est maintenue dans le réseau, et le contrôle est entièrement effectué de bout en bout. Elle représente une alternative à l'équité des débits qui est généralement considérée comme un objectif en soi de la plupart des solutions de qualité de service, remise en cause par Briscoe [39].

On en trouve les prémisses dans le concept d'Internet autogéré proposé par Kelly [83], qui montre que l'ajout d'une telle marque de congestion permet aux utilisateurs en bout de réseau de gérer de façon optimale l'allocation des ressources, et notamment de garantir de faibles délais grâce à de petits buffers. Le paiement d'une faible taxe proportionnelle au taux de congestion engendré, vu comme incitatif, a en fait été mal perçu par la communauté, notamment parce qu'il pouvait être un encouragement à sous-dimensionner volontairement les ressources du réseau.

2.4.2 Discussion et positionnement de FAN

L'architecture FAN sera présentée en détail dans le chapitre suivant. Dans cette section, nous signalons les caractéristiques de chaque proposition, et nous en servons pour positionner l'architecture FAN.

Granularité des propositions

La reconnaissance des flots est poussée plus ou moins loin dans le réseau en fonction des propositions. *Congestion Exposure* est similaire à l'approche *Best Effort* actuelle et considère le trafic uniquement au niveau paquet ; il appartient aux hôtes en bordure de réseau de gérer leur performance.

DiffServ ne matérialise les flots qu'en bordure de réseau ("domaine DiffServ"), et traite uniquement des agrégats dans le cœur en fonction d'une marque simple apposée dans l'en-tête de paquet. C'est également le cas de l'approche DPS/SCORE, qui maintient un état par flot (plus complexe) sur les nœuds de bordure uniquement, et transporte l'information nécessaire pour le cœur de réseau dans l'en-tête du paquet. De telles approches ont été conçues suites aux problèmes d'extensibilité de l'architecture IntServ, dans laquelle il convenait de signaler et de garder en mémoire l'état de chaque connexion.

Les approches Caspian/Anagran, FSA, FAN quand à elles identifient les flots sur chaque équipement. Les réalisations faites par Caspian et Anagran sont la preuve de la possibilité technique de réaliser un état par flot, qui apporte des avantages en terme de routage et de monitoring.

IntServ repose sur une signalisation lourde basée sur RSVP, FSA sur une signalisation intra-bande légère, alors que FAN se base uniquement sur des mesures implicites du trafic.

De plus, seul le contrôle d'admission de cette dernière requiert un état pour tous les flots, et il est possible de relaxer cette contrainte ; l'algorithme de *fair queueing* ne nécessite de retenir que les flots actifs, qui sont en nombre beaucoup plus limité, indépendamment de la capacité du lien.

Classes de service et différenciation

Congestion Exposure qui repose sur un réseau *Best Effort* ne fait aucune distinction de classe. Au contraire, la plupart des autres architectures reposent sur une distinction relativement complexe d'un certain nombre de classes (trois pour IntServ, trois éventuellement subdivisées pour Diffserv, cinq pour FSA) qui reçoivent soit une priorité fixe, soit une part fixe ou ajustable de bande passante. DPS/SCORE propose plutôt un mécanisme permettant d'émuler la présence de certaines classes de service. Ces classes sont généralement utiles en cas de congestion, les moins prioritaires étant dégradées.

Pour le cœur de réseau, FAN ne fait aucune distinction de classe, mais permet la différenciation implicite des flots *streaming* et élastique, qui possèdent des critères différents de qualité de service. Les flots *streaming* sont servis en priorité et les autres utilisent la bande passante laissée disponible. En régime nominal, la distinction de plusieurs classes de service n'est pas utile. En surcharge, il semble ainsi préférable d'éliminer l'excédent de charge par un contrôle d'admission. De ce fait, la différenciation ne se fait pas par un lien plus ou moins transparent, mais par un taux d'accessibilité au lien. La congestion ne dégrade pas tous les flots mais touche uniquement ceux qui sont rejetés. Il est possible de définir plusieurs classes de service possédant des taux d'accessibilité différents.

Équité, isolation et flots non-réactifs

Aucune architecture n'assure l'équité de débit entre les flots, à l'exception de FAN qui intègre un ordonnancement *fair queueing*, et DPS/SCORE qui peut l'émuler[8]. Cette équité n'est d'ailleurs

8. Il n'est pas précisé s'il est possible d'émuler plusieurs mécanismes simultanément

pas forcément recherchée pour *Congestion Exposure*. Faute d'un tel ordonnancement, la gestion des flots non-réactifs est soit inexistante, soit nécessite des mécanismes complexes de traitement. Dans *Congestion Exposure*, cette gestion repose sur un mécanisme de "facturation" de la congestion générée à l'utilisateur.

Seules les architectures qui implémentent un contrôle d'admission peuvent garantir l'isolation des flots (IntServ, DiffServ+PCN, FSA et FAN). La performance des flots ne peut être réellement protégée que dans les architectures travaillant à la granularité du flot comme IntServ, FSA et FAN. Les autres reposent sur la coopération des utilisateurs en bout de réseau. Nous soulignons que le traitement des flots *streaming* est réalisé implicitement par FAN.

Contrôle de la surcharge

Si un contrôle de la surcharge est prévu, son implémentation n'est que conseillée et rarement explicitée : IntServ, DiffServ+PCN, FSA. On trouve cependant plusieurs propositions de contrôle d'admission et de préemption basées sur RSVP ou PCN.

FAN intègre un mécanisme de contrôle d'admission, local, implicite et basé sur des mesures. Son amélioration fait l'objet du Chapitre 6. Comme nous l'avons évoqué précédemment, il peut faire l'objet de mécanismes avancés de re-routage, et complété par des mécanismes de routage multi-chemin ou d'équilibrage de charge.

Discussion

Parmi les architectures que nous venons de présenter, peu implémentent l'ensemble des mécanismes nécessaires à garantir la performance des flots *streaming* et élastiques. Elles ne reposent généralement pas sur un modèle simple et robuste permettant de connaitre la performance réalisée par ces flots, et nécessaire au respect des contrats de services avec les utilisateurs.

Comme nous allons le voir dans le chapitre suivant, FAN repose sur des descripteurs simples du trafic et des flots (la charge moyenne et le débit crête). La performance est dictée par des modèles efficaces et robustes aux évolutions du trafic, qui permettent une performance prévisible, et fournissent une ligne directrice pour le dimensionnement des réseaux et de leurs ressources.

Enfin, parce qu'elle ne repose sur aucun marquage ni signalisation, FAN permet une implémentation simple, neutre du point de vue des utilisateurs et des applications, ne nécessitant aucune standardisation. Son déploiement peut être progressif, ne pose aucun obstacle à assurer la performance des flots dans l'inter-domaine. Elle offre des composants que nous jugeons nécessaires aux évolutions futures des réseaux, des protocoles et de leurs utilisations.

Chapitre 3

Flow-Aware Networking (FAN)

L'assurance de garanties de service pour le trafic requiert la compréhension de la relation qui existe entre la capacité du réseau, la demande, et la performance obtenue. Cette relation peut-être établie en considérant une approche du trafic au niveau flot, qui hérite des études du télétrafic pour le réseau téléphonique. L'introduction de mécanismes "Flow-Aware" que nous présentons dans ce chapitre semble une alternative désirable aux architectures classiques et complexes de qualité de service, et suffisante malgré sa simplicité afin de satisfaire les besoins des applications véhiculées par le réseau.

La proposition *Flow-Aware Networking* (FAN) est réalisée pour le cœur de réseau au travers de l'architecture de routeur Cross-Protect. Elle consiste en l'association d'un ordonnancement équitable par flot (*fair queueing*), et d'un contrôle d'admission implicite par flot. Cette combinaison permet d'assurer la performance à la fois des applications temps-réel et des transferts de données, sans nécessiter la distinction de classes de service, ni la propagation de spécifications de trafic par signalisation. L'introduction de Cross-Protect est transparente pour les utilisateurs notamment parce qu'elle conserve l'interface *Best-Effort* du réseau.

Après la présentation de haut niveau faite dans le chapitre précédent, nous entrons ici plus en détail dans le fonctionnement de *Cross-Protect*, et nous décrivons les modèles de trafic sous-jacents. Ces derniers permettent la compréhension de la performance des réseaux intégrant différent types de flux applicatifs. Les résultats pour un lien isolé sont d'abord introduits, puis étendus à un contexte de réseau. Les conditions nécessaires à leur intégration sont ensuite explicitées. Les résultats présentés dans ce chapitre montrent comment l'architecture s'intègre au sein d'une démarche saine d'opération d'un réseau, fondée sur l'instrumentation, le dimensionnement et la gestion en temps réel des ressources.

Les travaux de cette thèse – qui s'appuient sur les hypothèses et les modèles de trafic introduits ici – s'inscrivent dans la continuité des travaux entrepris autour de FAN, qui ont pour objectif la conception d'une architecture de routeur efficace, robuste et correctement dimensionnée. A la fin de ce chapitre, nous présentons l'environnement de simulation et la plateforme d'expérimentation GNU/Linux que nous avons réalisés afin de supporter nos résultats, et de tester et démontrer les mécanismes proposés.

Sommaire

3.1 Introduction **32**
3.2 Modélisation du trafic streaming **33**
3.3 Modélisation du trafic élastique **34**
3.4 Intégration des flots *streaming* et élastiques **38**
3.5 L'architecture de différentiation implicite de service Cross-Protect **39**
3.6 Vers la réalisation d'un routeur Cross-Protect **42**

Contributions :

- Extension de l'algorithme PFQ (*Priority Fair Queueing*) pour les besoins des travaux réalisés lors de cette thèse (ajout de nouveaux indicateurs de congestion).
- Réalisation d'un ensemble de modules pour le simulateur de réseau NS-2, parmi lesquels les mécanismes *Cross-Protect* (PFQ + Contrôle d'admission), les modèles de trafic utilisés, etc.
- Dépôt logiciel : *Implémentation sur noyau Linux des mécanismes Cross-Protect V1.0*
- Réalisation d'une plateforme de test Cross-Protect dans un environnement GNU/Linux ayant permis la démonstration de la faisabilité de l'architecture et ses avantages, intégrant des outils de génération de trafic, ainsi que d'analyse et de visualisation des résultats.

Démonstrations :

- Démonstration interne France Télécom
- Démonstration lors d'un séminaire France Télécom sur le trafic (25/03/2005)

3.1 Introduction

3.1.1 Ressources – Demande – Performance

Flow Aware Networking (FAN) [118] propose une approche de la QoS en rupture avec les architectures classiques présentées dans le chapitre précédent. FAN propose de caractériser et de contrôler le trafic au niveau des flots utilisateurs ; on en trouve les prémisses dans les travaux de Roberts *et al.* ([132] et suivants), qui insistent sur la nécessité de caractériser la relation liant les ressources du réseau, la demande utilisateur, et la performance obtenue.

ressources : il s'agit principalement de la capacité disponible sur les différents liens, et du partage réalisé entre les flots. On parle d'allocation de ressources. Les buffers au sein des routeurs IP sont également une ressource importante qu'il convient de considérer, comme nous le verrons dans le Chapitre 4.

demande : la demande en trafic correspond au trafic généré par les utilisateurs qui consomment les ressources du réseau. On peut la caractériser par le type et les caractéristiques essentielles des différents flots qui la constitue. De nombreuses analysent considèrent un nombre fixe de flots permanents ; Roberts et Massoulié [132] ont montré que le trafic Internet est constitué d'une superposition dynamique de flots de taille finie, rythmée par les arrivées de nouveaux flots, et les départs de flots dont le transfert se termine. La demande est généralement caractérisée lors de la période de pointe, en vue du dimensionnement des ressources. Nous verrons plus loin que la connaissance d'un faible nombre de critères comme la charge globale (le "volume" de trafic) et le débit crête des flots peut être suffisante afin d'en déduire la performance des flots.

performance : la performance est une mesure de la qualité, des garanties de service fournies au trafic. La nature aléatoire de ce dernier nous mène à considérer des garanties probabilistes, correspondant aux besoins des différents types de flots : on s'intéressera à l'espérance du débit (ou au temps moyen de réponse) pour les flots élastiques, ainsi qu'à la probabilité de perte ou au délai des paquets *streaming*.

La présence d'un mécanisme de contrôle d'admission permet l'assurance de garanties de services (minimum) à chacun des flots admis. Il est alors censé de s'intéresser à des métriques telles que le taux de blocage des flots.

3.1.2 Modèles de trafic

Les sections qui suivent présentent un ensemble de modèles de trafic permettant l'étude de la performance des flots *streaming* et élastiques. Ces modèles de trafic forment la base d'une *théorie du trafic internet*. Les modèles les plus simples permettent une compréhension qualitative de la performance réalisée par les flots, ainsi que de l'impact de l'introduction de nouveaux mécanismes. Des versions plus raffinées, que nous référencerons uniquement, permettront par contre de meilleurs prédictions. Les chapitres suivants s'appuieront sur de tels résultats.

Une caractéristique essentielle de ces modèles est leur propriété d'insensibilité : la performance obtenue ne dépend pas de caractéristiques détaillées du trafic (par exemple la connaissance de la distribution de la longueur des flots). On trouvera ainsi de nombreuses analogies avec les résultats établis pour le réseau téléphonique (télétrafic), dont la formule d'Erlang offre une très bonne illustration.

Cette formule, proposée par Erlang en 1917, permet de calculer la probabilité de blocage $B(\cdot)$ d'un appel en fonction du nombre de circuits n et de la demande moyenne E générée par les appels (taux d'arrivée des appels x temps moyen de communication) :

$$B(E,n) = \frac{\frac{E^n}{n!}}{\sum_{i=0}^{n} \frac{E^i}{i!}}$$

Elle nécessite seulement que les arrivées des appels suivent un processus de Poisson, et possède la propriété d'être insensible à la distribution de la longueur des appels. C'est la raison pour laquelle elle peut toujours être utilisée pour le dimensionnement des réseaux téléphoniques actuels, malgré l'évolution des usages et de leurs implications sur le trafic généré.

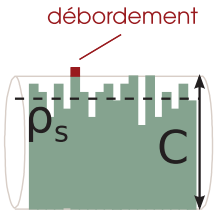
3.2 Modélisation du trafic streaming

Lors de leur génération par les applications audio et vidéo, les flots *streaming* possèdent des profils de débits divers : on trouve des flots à débit constant (CBR, *Constant Bit Rate*) ou variable (VBR, *Variable Bit Rate*). Les principales applications possèdent des débits limités ($<$ 64kb/s pour la téléphonie, 4 à 16Mb/s pour les flux vidéos, etc.) relativement faibles par rapport aux capacités des liens considérés pour le cœur de réseau. Les garanties de QoS pour ces flots s'expriment au niveau paquet, par de faibles délais et un taux de pertes négligeable.

Plusieurs modèles proposent de contrôler la charge moyenne du trafic afin qu'elle ne dépasse pas la capacité du lien. La gigue acquise par les flots est problématique : les multiplexages successifs au sein des files d'attente des routeurs créent des rafales de paquets qu'il est difficile de caractériser. Ces modèles ne sont pas adaptés pour offrir des garanties de service, sauf à considérer des situations très conservatrices avec des tailles de buffer surdimensionnées. Nous considérons ce problème de dimensionnement de buffer dans le Chapitre 4.

Nous nous intéressons ici à une proposition alternative n'utilisant pas de buffer, sauf pour absorber les arrivées simultanées de paquets ou les excédents transitoires de trafic.

3.2.1 Multiplexage statistique "sans buffer" : cas d'un lien isolé



Le modèle de multiplexage dit *sans buffer*, *Rate Envelope Multiplexing* [138, 19], propose l'utilisation d'un contrôle d'admission afin de conserver la transparence du lien. En d'autres termes, la probabilité que la charge du trafic *streaming* ρ_s dépasse la capacité du lien C doit rester négligeable : $P[\rho_s > C] \leq \epsilon$, comme illustré sur la figure ci-contre. Les délais de paquets restent alors faibles, et la probabilité de perte l est donnée approximativement par : $l = E[(\rho_s - C)^+]/E[\rho_s]$. Elle dépend uniquement de la distribution stationnaire du débit d'entrée et n'est pas sensible à d'autres caractéristiques détaillées du processus d'arrivée de paquets (telles que les corrélations lors des rafales).

Le taux d'utilisation atteint est plus faible que si l'on se permettait d'utiliser le buffer. Il reste toutefois très satisfaisant dès lors que l'agrégat de trafic présente une variabilité limitée, lorsque les flots ont des débits faibles par rapport à la capacité du lien (par exemple moins de 1% de C). C'est en pratique le cas sur un réseau comme l'Internet, et la capacité restante peut avantageusement être utilisée pour le transfert des flots élastiques.

Contrôle d'admission pour les flots *streaming*

Les algorithmes de contrôle d'admission basés sur des mesures sont particulièrement adaptés à une telle réalisation, puisqu'ils peuvent se contenter de descripteurs de trafic minimums. L'algorithme présenté dans Gibbens *et al.* [62] requiert par exemple uniquement la connaissance du débit crête du flot, associée à une estimation de la charge moyenne du lien. Un très grand nombre d'algorithmes ont été proposés, et nous passons en revue les plus adaptés à notre contexte dans le Chapitre 6, qui considère un réseau intégrant les flots *streaming* et élastiques.

Dimensionnement

Considérons des flots de débit r et de durée moyenne σ_s arrivant sur le lien selon un processus de Poisson d'intensité λ_s. La demande moyenne en trafic est $\rho_s = \lambda_s \sigma_s r$ (en b/s). En supposant un

contrôle d'admission limitant le nombre de flots, la probabilité de blocage p_b dans ce modèle est alors approximée par la formule d'Erlang, où ρ_s/r erlangs de trafic sont offerts à C/r circuits :

$$p_b = B(\rho_s/r, C/r) \qquad \text{avec} \qquad B(E,m) = \frac{E^m/m!}{\sum_{i=0}^{m} E^i/i!} \tag{3.1}$$

Il est alors possible de déterminer la capacité C nécessaire pour un blocage minimal. Des généralisations de ces modèles existent pour des débits d'accès différents, ou des flots à débit variable, voir par exemple [131].

3.2.2 Garanties de performance de bout en bout

Bonald *et al.* [33] montrent que lorsque les flots *streaming* ont un débit crête limité et qu'ils sont traités en priorité non-préemptive, il est possible de donner des bornes statistiques sur les délais et la gigue, qui dépendent alors uniquement de la charge de ces flots. Notamment, ils n'acquièrent jamais une gigue suffisante pour avoir une performance moins bonne que des arrivées de paquet de taille MTU suivant un processus de Poisson (conjecturé). Il est ainsi possible en limitant la charge des flots par un contrôle d'admission sur chaque lien, de garantir une performance convenable de bout en bout.

3.3 Modélisation du trafic élastique

Les modèles de partage statistique de bande passante (*Statistical Bandwidth Sharing*) permettent de mieux comprendre comment les flots TCP se partagent les ressources disponibles, ainsi que l'impact des caractéristiques du trafic sur la performance obtenue. Dans cette section, nous suivons l'approche de Roberts [134] qui propose une vue d'ensemble de ces modèles et de leur application à l'étude des réseaux de données. Certains résultats permettent d'expliquer le comportement des connexions TCP au niveau paquet. Padhye [120] a par exemple montré que le débit moyen d'une connexion TCP est lié au taux de pertes subi. Cependant, l'abstraction au niveau flot présente l'avantage de simplifier l'analyse, notamment au vu des propriétés d'autosimilarité du trafic, et de s'abstraire d'une version précise de TCP.

3.3.1 Modèle de sessions Poisson

Alors qu'il est reconnu que les arrivées de sessions suivent un processus de Poisson [125], ce n'est généralement pas le cas des arrivées de flots. Bonald *et al.* [32, 15] ont montré que l'on peut se ramener à un tel contexte pour l'étude de la performance du trafic. Ils considèrent pour cela un modèle assez général de sessions, constituées d'une alternance de flots et de périodes d'inactivités, de distributions générales, et où les arrivées de flots peuvent éventuellement être corrélées. Il est par exemple possible de considérer des distributions à queue lourde pour le nombre de flots par session, ce qui contribue à générer un trafic autosimilaire [98, 124].

3.3.2 Modèle processeur partagé : cas d'un lien isolé

Le modèle à processeur partagé, noté PS pour *Processor Sharing*, permet de modéliser le partage des ressources réalisé par TCP au niveau flot. Il suppose un partage instantané et équitable. En réalité, le partage réalisé par TCP n'est pas instantané (lenteur de l'algorithme AIMD), et est biaisé par le démarrage en mode *slow start* et des différences de RTT entre les connexions. Ce modèle illustre cependant bien l'impact du nombre de flots en cours sur la performance.

Nous verrons dans le Chapitre 4 comment adapter ce modèle au cas où la performance de TCP est affectée par la présence de petits buffers. Le Chapitre 5 montrera dans quelle mesure l'introduction du *fair queueing* dans le réseau permet de relaxer ces hypothèses.

Propriété d'insensibilité

Considérons un lien de capacité C partagé par un ensemble de flots. La taille et la constitution de cet ensemble varient dans le temps au fil des arrivées et départs des divers flots. Lorsque les flots sont générés selon le modèle de sessions Poisson, on peut montrer que les mesures de performance telles que le débit moyen des flots sont insensibles à des caractéristiques détaillées du trafic comme la distribution de la taille des flots, ou le nombre de flots par session. Les caractéristiques essentielles du

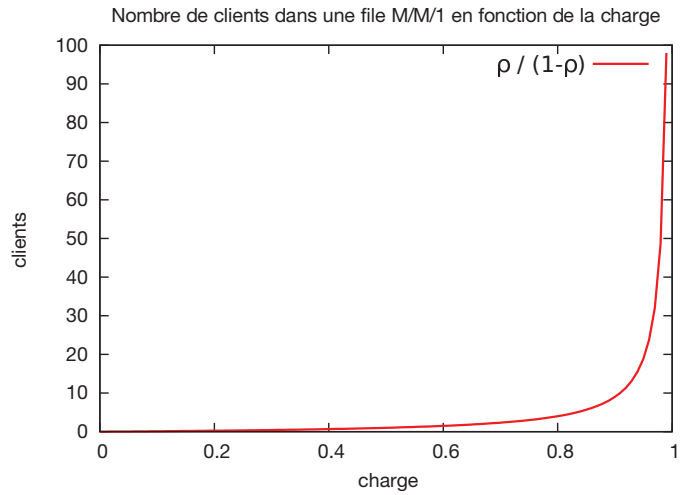


FIGURE 3.1 – Nombre moyen de flots en cours dans une file M/M/1, en fonction de la charge ρ.

trafic sont la **charge du lien** ρ, égale au taux d'arrivée des flots x taille moyenne des flots / capacité du lien, et le **débit crête des flots**, qui est le débit maximum qu'un flot peut atteindre indépendamment du lien considéré. Une telle propriété d'insensibilité n'est pas garantie dans le cas FIFO.

Composition du trafic

Lorsque tous les flots peuvent individuellement atteindre la capacité du lien, le nombre de flots en cours dans le modèle fluide PS idéal correspond à un processus de naissance et de mort. Il possède une distribution géométrique de moyenne $\rho/(1-\rho)$. représentée sur la figure 3.1.

Malgré la simplicité du modèle, c'est une bonne indication que le nombre de flots en cours à n'importe quel instant est généralement plutôt faible (par exemple moins de 20 avec une probabilité de .99 pour une charge de 80%). Or, en pratique, le nombre de flots observé sur un lien de cœur de réseau est nettement plus élevé (jusqu'à plusieurs dizaines de milliers sur un lien gigabit), ce qui induit que **la plupart ne sont pas *bottlenecked***. En effet, de très nombreux flots sont contraints par leur débit d'accès, par un lien saturé ailleurs dans le réseau, ou encore parce qu'ils ne parviennent pas à quitter le mode *slow start* et donc n'atteignent pas leur débit crête.

Le nombre de flots *bottlenecked* est très grand seulement lorsque la charge du lien est proche de 1. Une confirmation empirique du nombre limité de flots *bottlenecked*, indépendamment de la capacité du lien, est présentée dans Kortebi *et al.* [94], où les auteurs analysent plusieurs traces de trafic. Ce résultat renforce les premières observations de Suter *et al.* [146] qui montrent que la population des flots est très dynamique dans le buffer.

Débit des flots

Nous illustrons la performance du partage statistique de bande passante par la mesure de débit des flots γ = taille moyenne des flots / durée moyenne des flots. La mesure γ peut également être interprétée comme l'espérance du débit instantané d'un flot [26]. Nous considérons un modèle PS généralisé où les flots partagent un taux de service dépendant du nombre de flots en cours. Nous notons $\phi(i)C$ le taux de service du lien lorsque i flots sont en cours, et $\Phi(i) = \prod_{j=1}^{i} \phi(j)$. Nous avons alors :

$$\gamma = \frac{\sum \rho^i / \Phi(i)}{\sum i\rho^{i-1} / \Phi(i)}. \tag{3.2}$$

La figure 3.2 représente γ en fonction de ρ lorsque les flots n'ont aucune contrainte de débit crête ($\phi(i) = 1$), ainsi que lorsqu'ils possèdent un débit crête limité $p = 0.1C$ ($\phi(i) = \min(ip, 1)$). **Lorsque les flots ont un débit crête limité p, le lien est transparent pour la performance du débit jusqu'à de fortes charges (proches de $1 - p/C$).**

Contrôle d'admission pour les flots élastiques

Lorsque la charge du lien est trop proche de 1, le nombre de flots en cours augmente rapidement et les débits tendent vers 0 (figure 3.2). Le comportement de la file d'attente est instable et le temps de

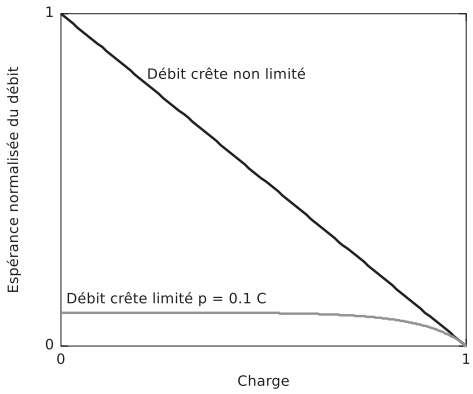


FIGURE 3.2 – Débit normalisé des flots γ en fonction de la charge ρ lorsque les flots n'ont aucune limite de débit, et lorsqu'ils possèdent un débit limité (ici égal à $.1C$).

réponse des flots n'est plus borné. On notera qu'au niveau paquet, avec des buffers de taille finie, le lien est toujours stable en raison des pertes de paquets.

Ben Fredj *et. al* [15] remarquent que l'accumulation des flots est plus lente lorsque la distribution de leur taille possède une queue lourde, les flots de petite taille parvenant à terminer, au détriment des flots plus longs. Les auteurs modélisent également l'impatience des utilisateurs (des applications) qui stabilisent le système en interrompant les connexions lorsque le temps de réponse devient trop important. Cette argumentation diffère de celle reposant sur les modèles classiques d'utilité pour les flots élastiques où celle-ci est strictement concave, même lorsque le débit tend vers 0 [140].

Les abandons de flots entraînent un gaspillage des ressources du réseau, qu'il convient de prévenir au plus tôt. Le même problème de congestion se produit dans les architectures Diffserv qui protègent uniquement la performance des classes les plus prioritaires, en reportant la congestion sur la classe *Best effort* [20]. Ben Fredj *et al.* [14] proposent un contrôle d'admission pour les flots élastiques en limitant le nombre de flots en cours se partageant la bande passante.

Lorsque les flots n'ont pas de limite de débit et que le partage est équitable, une proposition simple revient à limiter les flots à un nombre maximal m, afin que leur débit moyen soit d'au moins C/m. Le système peut être représenté par une file M/G/1/m/PS. Lorsque $\rho < 1$, la probabilité de blocage p_b est négligeable, et pour $\rho > 1$:

$$p_b(\rho, m) = \frac{\rho^m(1-\rho)}{1-\rho^{m+1}} \xrightarrow[m\to\infty]{} 1 - 1/\rho$$

Le choix du seuil d'admission n'est pas critique pour la performance dès que $m > 50$ [14]. Une valeur de m plus importante n'a pas d'impact sur la probabilité de blocage des flots, mais entraîne un gaspillage des ressources. Il est donc bénéfique de choisir une valeur la plus petite possible, d'où le compromis $m = 100$ choisi dans [91].

Avec des flots de débit limité r, on peut choisir $m = C/r$. La probabilité de blocage devient alors : $p_b(\rho, m) \approx B(\rho m, m)$, où $B(.)$ est la formule d'Erlang présentée en (3.1). Le choix de $m = 100$ fait précédemment convient également dans ce cas.

Dimensionnement

Ces formules permettent le dimensionnement d'un lien en vue d'offrir un blocage négligeable aux flots à charge nominale. La présence de flots à débit limité ne permet pas de limiter directement le nombre de flots en cours, mais il convient plutôt de considérer le débit qu'aurait un nouveau flot non goulotté sur le lien. Nous verrons comment ce contrôle d'admission est mis en pratique dans Cross-Protect dans la Section 3.5.3.

L'ajout d'un contrôle d'admission permet également de conserver l'insensibilité du modèle. Des extensions de ces résultats existent pour des débits d'accès différents, où dans le cas d'un partage non-équitable (modèle processeur partagé discriminatoire, ou DPS, *Discriminatory Processor Sharing*) ; l'insensibilité est alors approximative.

3.3.3 Extension à un réseau de données

Considérons un réseau formé d'un ensemble $\mathcal{L}$ de liens de capacité $C_l, l \in \mathcal{L}$. K classes sont définies, telles que les flots de classe k suivent la route $r_k \subset \mathcal{L}$ et ont un débit d'accès limité a_k. Nous considérons les allocations donnant un débit équitable aux flots de la même classe.

Allocation de ressources

Considérons temporairement des flots permanents de K classes, dont le nombre est représenté par le vecteur $X = (n_1, \cdots, n_K)$. Le vecteur des débits à long terme de chaque classe est $\Phi = (\phi_1, \cdots, \phi_K)$, tel que le débit de chaque flot de classe k est ϕ_k / n_k.

La région des débits réalisable $\mathcal{R}$ représente l'ensemble des vecteurs Φ possibles ; c'est un polytope convexe [1] défini par les contraintes posées par chaque lien [2]. Une allocation est efficace si Φ se situe sur le bord de $\mathcal{R}$. Elle est Pareto-efficace (et donc efficace) si elle consomme toutes les ressources [3].

Les allocations usuelles sont basées sur une fonction concave d'utilité $U(.)$, et se ramènent au problème d'optimisation suivant :

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{k=1}^{K} n_k U\left(\frac{\phi_k}{n_k}\right) \\ \text{s.c.} \quad & \left(\sum_{k \backslash l \in r_k} \phi_k \leq C_i\right) (\forall l \in \mathcal{L}) \\ & \phi_k / n_k \leq a_k, \forall k \in [\![1, K]\!] \end{aligned}$$

La première contrainte correspond aux capacités des liens, et la seconde aux débits d'accès.

On retrouve les allocations classiques : *max throughput* ($U(\cdot) = \cdot$), *proportional fairness* [84] ($U(\cdot) = log(\cdot)$), *minimum potential delay* [136] ($U(\cdot) = 1/\cdot$), et *max-min* [22] qui est un cas limite d'une mesure d'utilité [110]. Les auteurs de [110] regroupent ces allocations, dites *α-fair*, sous une formulation mathématique commune. Lan *et al.* [96] généralisent plusieurs notions d'équité (dont les allocations *α-fair* et l'indice de Jain) et donnent un aperçu du compromis réalisé par les allocations *α-fair* entre équité et performance.

Ce problème d'optimisation se prête notamment bien à la réalisation distribuée d'algorithmes de contrôle de congestion de type TCP. Sauf retour explicite du réseau, la saturation des liens est typiquement déterminée de manière heuristique, en se basant par exemple sur les pertes de paquet pour les versions classiques de TCP.

Stabilité

La région de stabilité S est définie par l'ensemble des vecteurs $\rho = (\rho_1, \cdots, \rho_K)$ tels que le réseau est stable, c'est-à-dire que les flots terminent en un temps fini, i.e. $X(t)$ est borné. Lorsque R est convexe, alors S est exactement $\mathring{R}$, la frontière de R. Il suffit ainsi que la charge de tous les liens soit inférieure à leur capacité [28], ce qui peut-être assuré par un contrôle d'admission. Les allocations basées sur l'utilité (ainsi que l'allocation balancée que nous allons introduire) maximisent la région de stabilité, et sont ainsi une bonne approche pour les réseaux filaires.

Allocation balancée

Comme nous l'avons vu, le nombre de flots en cours est un processus aléatoire. On note $X(t)$ le vecteur représentant le nombre de flots de chaque classe, ν_i le taux d'arrivée des flots sur le nœud i, $p_{i,j}$ la probabilité de routage de i à j et μ_i le taux de service moyen de la file i. Les flots de classe k arrivent selon un processus de Poisson (sans perte de généralité dans le cadre d'arrivées de sessions Poisson) d'intensité λ_k et de tailles exponentielles i.i.d de moyenne σ_k. La charge correspondante est $\rho_k = \lambda_k \sigma_k$ bits/s.

Dans un tel contexte dynamique, Massoulié et Roberts [132] précisent qu'à l'échelle de temps des flots, qui est celle perçue par un utilisateur, la performance dépend autant du processus du nombre de flots en cours $X(t)$ que de l'équité de l'allocation. L'utilité d'un flot est plus justement mesurée par

1. Cela est valable pour un réseau filaire, ce n'est généralement pas le cas, par exemple pour un réseau sans fil avec interférences [103]

2. On a un polytope convexe variable dans le temps si l'on considère des mécanismes de priorité, des chemins multiples (*multipath*), ou encore des pannes de lien

3. Ces deux notions sont équivalentes si la bordure de $\mathcal{R}$ ne contient aucun segment parallèle à l'axe d'une classes de flots.

des métriques telles que leur temps de complétion. Il suffit que l'allocation soit suffisamment équitable pour que la performance des flots soit satisfaisante [28].

Le problème majeur des allocations *α-fair* est qu'elles sont sensibles à des caractéristiques détaillées du trafic et ne permettent pas une modélisation efficace de la performance des flots. Bonald et Proutière [30] introduisent la notion d'allocation balancée (BF, pour *balanced fairness*), qui est l'unique allocation optimale et insensible possible dans un réseau. Sa formulation repose sur les réseaux de Whittle, qui sont une extension des réseaux de Jackson où le taux de service dépend de l'état. Ce modèle est une extension directe du modèle PS à un réseau de files d'attente. Les auteurs ont montré que l'insensibilité de la distribution stationnaire du nombre de flots [4] correspondait à la réversibilité partielle du processus Markovien $X(t)$. Cela se transcrit dans le réseau par l'équilibrage des débits par une fonction positive $\Phi(x) : \phi_k(x) = \Phi(x - e_k)/\Phi(x)$, pour $k = 1, \cdots, K$.

L'allocation balancée correspond au choix d'une telle fonction respectant les contraintes de capacité et maximisant l'utilisation des ressources. Elle est définie de manière récursive avec $\Phi(0) = 1$ et

$$\Phi(x) = \max\left(\max_{l=1,\ldots,L}\left(\frac{1}{C_l}\sum_{i:l\in r_i}\Phi(x-e_i)\right), \max_{k:x_k>0}\left(\frac{1}{a_k x_k}\Phi(x-e_k)\right)\right)$$

avec $\Phi(x) = 0$ si $x \notin \mathbb{N}^N$.

Bornes stochastiques de performance

Malgré sa propriété intéressante d'insensibilité pour un dimensionnement robuste des liens, l'allocation balancée reste complexe à évaluer dans un contexte pratique, puisqu'elle dépend de la demande sur toutes les routes et des capacités des liens. Bonald et Proutière [31] proposent des bornes stochastiques sur le temps de réponse des flots, permettant d'évaluer de manière conservatrice la performance de l'allocation à partir d'informations locales aux liens uniquement. Les auteurs suggèrent un comportement similaire pour les autres allocations équitables de type *max-min* ou *proportional fair*. Kortebi *et al.* [94] donnent notamment un exemple d'application de l'allocation balancée, afin d'estimer la performance de *max-min*.

L'architecture *Flow-Aware Networking* réalise *max-min* grâce à l'emploi de *fair queueing* [132], ce qui semble être un choix acceptable et dont on peut approximer la performance à l'échelle du réseau grâce à l'allocation balancée.

3.4 Intégration des flots *streaming* et élastiques

Les besoins très différents de performance entre les flots *streaming* et élastiques permettent à [15, 91] de proposer un ordonnancement approprié qui bénéficie de leur différenciation : les flots *streaming* sont servis en priorité, tandis que flots élastiques utilisent la capacité laissée disponible. Le lien est ainsi transparent pour le trafic *streaming*, du moment que leur charge reste contrôlée. Nous nous intéressons ici à la performance réalisée par les flots lors d'une telle intégration.

Si [88] montre que la stabilité d'un réseau avec des flots *streaming* adaptatifs n'est pas affectée, ce n'est généralement pas le cas avec des flots non-adaptatifs. Les articles [116], [46] et [89] évaluent la performance d'un réseau où les flots *streaming* non-adaptatifs sont traités en priorité. Delcoigne *et al.* [46] donnent notamment des explications qualitatives sur la performance réalisée : la performance des flots élastiques peut être affectée par un phénomène d'instabilité locale, quand bien même la charge globale reste inférieure à la capacité du lien. Un contrôle d'admission approprié pour les flots *streaming* permet de garantir la performance dans un tel contexte, ainsi que de fournir des bornes de performances insensibles pour le temps de réponse d'un flot. Les bornes sont très étroites dans des conditions réalistes de trafic.

La relation entre ressources, demande et performance, introduite dans la section 3.1.1 est représentée pour FAN en figure 3.3. Etant données les capacités des liens, et une caractérisation simple de la demande en trafic (charge et débit crête pour les différents types de flots), l'architecture FAN apporte des garanties de service pour les flots *streaming* (délais, taux de pertes et gigue négligeables), et élastique (débit moyen minimum garanti).

L'architecture Cross-Protect que nous allons présenter réalise un contrôle d'admission implicite dans un tel contexte. L'article original [91] considère des situations où la proportion du trafic *streaming* n'est pas trop importante. Nous nous intéressons à un contexte plus réaliste où la plupart des flots sont limités en débit dans le Chapitre 6.

4. le fait qu'elle ne dépende pas de la distribution de la durée des flots

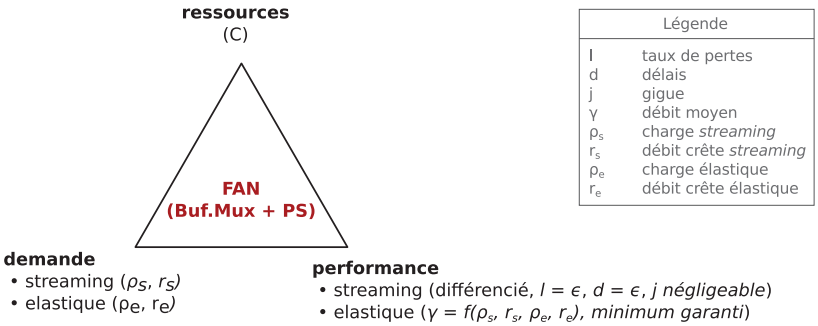


FIGURE 3.3 – Relation fondamentale entre ressources, demande et performance réalisée par l'architecture FAN

.

3.5 L'architecture de différentiation implicite de service Cross-Protect

Nous considérons une réalisation de FAN pour le cœur de réseau, Cross-Protect, qui permet de garantir la performance des flots *streaming* et élastiques. Cross-Protect repose sur la combinaison d'un d'ordonnancement équitable (*fair queueing*) et d'un contrôle d'admission implicite au niveau flot, basé sur des mesures (MBAC). Cette architecture ne nécessite pas de mécanisme complexe de marquage ou de signalisation, et permet ainsi au réseau de conserver son interface *Best Effort*, tout en permettant les évolutions nécessaires au sein du réseau : routage adaptatif, optimisations de TCP, etc. Une déclinaison de ces mécanismes pour le réseau d'accès sera considérée dans le Chapitre 7.

Le nom Cross-Protect vient de la complémentarité des deux mécanismes mis en œuvre. Le contrôle d'admission utilise des mesures de congestion fournies par l'ordonnanceur afin de décider de l'acceptation ou du rejet d'un nouveau flot ; en contrôlant la charge des flots *streaming* et élastique, il permet de conserver leur performance et d'assurer l'extensibilité de l'ordonnanceur (le nombre de flots à maintenir à tout instant est borné).

3.5.1 PFQ : un algorithme de FQ extensible

Le comportement naïf d'un algorithme d'ordonnancement équitable (*fair queueing*, ou FQ) consiste à diviser l'espace du buffer en plusieurs files d'attente (virtuelles), qui reçoivent chacune les paquets d'un seul flot. Il en résulte que FQ est souvent considéré comme non-extensible sur des liens de forte capacité où le nombre de flots en cours est très élevé.

Principe de l'algorithme PFQ

L'algorithme *Priority Fair Queueing* (PFQ) [91] est une extension de l'algorithme Start-Time Fair Queuing (SFQ), qui maintient un temps virtuel et attribue une estampille à chaque paquet déterminant son ordre d'émission. PFQ distingue implicitement les flots selon que leur débit instantané est inférieur ou supérieur au débit équitable. Celui-ci fluctue et est défini à tout instant en fonction du nombre et des caractéristiques des flots en compétition. Il faut noter que le débit de chaque flot n'est pas calculé, mais au lieu de cela PFQ se fonde sur l'évolution de la file d'attente.

L'algorithme est détaillé dans l'article original [91] et illustré en figure 3.4. Les flots dont le débit est supérieur au débit équitable voient leurs paquets s'accumuler dans la file d'attente du routeur jusqu'aux instants de pertes de paquets. L'algorithme d'ordonnancement est responsable du partage de la bande passante laissée disponible entre ces flots. PFQ intègre une gestion de la file d'attente qui oriente les pertes sur les flots de plus fort débit qui ont accumulé le plus grand nombre de paquets : cela protège les flots *streaming* à débit limité et permet au mécanisme de contrôle de congestion de TCP de réagir. Il est suffisant pour un algorithme d'ordonnancement de maintenir une file pour ces flots uniquement, qualifiés de flots actifs. Les flots de débit inférieur ont à tout moment au plus un paquet dans les files d'attente du routeur. Ils ne participent pas activement aux mécanismes de partage de bande passante et peuvent être "ignorés" par l'algorithme. Sans affecter la performance des autres flots, ils peuvent être servis en priorité, tant que leur charge reste limitée. PFQ maintient une file mixte, dite file PIFO (*Push In First Out*), dont le début représente la partie prioritaire et où tous les paquets ont la même estampille, et dont la fin reçoit les autres paquets, ordonnés en fonction de leur estampille.

Nous avons montré précédemment que le nombre de flots goulottés sur un lien est généralement faible jusqu'à de fortes charges. Ce nombre est plus important pour des flots à débit limité, mais [94]

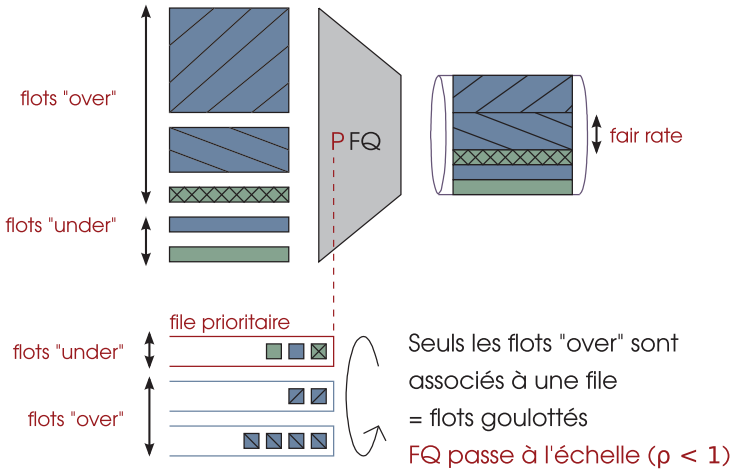


FIGURE 3.4 – Illustration du fonctionnement de l'algorithme PFQ

montre à partir d'un modèle confirmé par des traces de trafic que ce nombre est limité à quelques centaines jusqu'à une forte charge indépendamment de la capacité du lien.

Il existe d'autres déclinaisons de l'algorithme PFQ pour Cross-Protect , comme PDRR [92] (*Priority Deficit Round Robin*), qui se base sur DRR (*Deficit Round Robin*) [155] et présente l'avantage d'avoir une complexité en O(1), et d'être plus adaptée à la réalisation au sein d'un routeur commercial incluant déjà des mécanismes tels que DRR. Si elle est similaire en théorie, cette réalisation se révèle un peu moins équitable en pratique en raison de la dynamique des arrivées et départs de flots.

Le fair rate instantané : une caractéristique de la file d'attente

Lorsque tous les flots sont *non-bottlenecked*, les arrivées et départs de paquets définissent des périodes d'activité (*busy period*) et de silence (*idle period*) de la file d'attente, qui permettent de déduire directement le débit équitable instantané, ou *fair rate* instantané (fr_i).

Lorsqu'un ou plusieurs flots sont *bottlenecked*, on définit alors un *busy cycle* comme l'intervalle pendant lequel l'ensemble des flots *non-bottlenecked* sont traités, ainsi qu'un quantum (par ex. 1 MTU) au plus de chaque flot *bottlenecked*. Lorsqu'un paquet d'un flot arrive alors qu'il a déjà été servi dans le même cycle, il est inséré dans la file non prioritaire (*backlogged*) pour être traité dans les cycles suivants.

L'algorithme a ainsi besoin de maintenir une liste des flots actifs pendant chaque cycle ou *busy period* afin de garder en mémoire les flots précédemment émis. Cette liste est vidée lorsque le lien devient inactif, ou du moins purgée des flots terminés lors d'un changement de cycle, qui se manifeste par le changement de l'estampille du paquet en tête de file.

3.5.2 Indicateurs de congestion

Le fonctionnement naturel de l'algorithme PFQ permet la mesure de plusieurs indicateurs de congestion, qui sont utilisés par le contrôle d'admission. Le *fair rate* et le *priority load* sont présents dans la proposition initiale de PFQ.

La charge prioritaire, ou *priority load* (PL), est une mesure lissée exponentiellement du volume de trafic $B(t)$ arrivant dans la file prioritaire pendant un intervalle de temps τ.

$$pl = \Delta B / \tau$$

$$PL = (1 - \alpha) pl + \alpha PL$$

où $\Delta B = B(t + \tau) - B(t)$ et α un paramètre de lissage ($0 < \alpha < 1$).

Le débit équitable, ou *fair rate* (FR), est également une moyenne à long terme de la quantité suivante :

$$fr = MAX(\Delta vt, S.C) / \tau$$

$$FR = (1 - \beta) fr + \beta FR$$

où vt est le temps virtuel de la file, c'est-à-dire l'estampille temporelle du paquet en tête de file, S la durée pendant laquelle la file est vide sur l'intervalle τ, et $\Delta vt = vt(t + \tau) - vt(t)$. FR représente le débit moyen d'un flot fictif qui aurait constamment des paquets à émettre.

Nouveaux indicateurs de congestion

Dans le contexte d'un lien sur lequel les flots sont majoritairement élastiques, des estimations grossières suffisent, notamment pour PL, et les valeurs de τ sont dans les deux cas choisies égales à 100ms. Nous verrons dans le Chapitre 6 le besoin de définir un algorithme plus avancé pour les cas fréquents où le lien transporte majoritairement des flots à débit limité (cas d'un lien ADSL par exemple).

L'adaptation de l'algorithme afin de fournir une mesure de fr_i a été faite pour les besoins de l'algorithme présenté dans le Chapitre 6. Nous ajouterons également une mesure de la variance du PL (en plus de la valeur moyenne), et nous analyserons l'importance d'un choix correct de τ.

3.5.3 Contrôle d'admission basé sur des mesures

Motivations

L'intégration d'un contrôle d'admission est une solution efficace et éprouvée permettant d'agir proactivement afin de garantir la performance des flots en cours sur un lien [139, 24], en rejetant l'excédent de trafic. Il doit être transparent dans des conditions normales de charge.

Contrôle d'Admission Basé sur des Mesures

Les algorithmes de contrôle d'admission basés sur des paramètres de trafic, PBAC (*Parameter Based Admission Control*), offrent des garanties déterministes aux flots en fonction de descripteurs de trafic. Outre le fait que ces derniers sont très complexes à définir en raison du comportement du trafic IP au niveau paquet, ces algorithmes s'avèrent peu efficaces. Ils conduisent à une faible utilisation du lien, et nécessitent des mécanismes de signalisation et de contrôle de conformité du trafic.

Le choix pour Cross-Protect d'un contrôle d'admission basé sur des mesures (MBAC, *Measurement Based Admission Control*) provient de sa simplicité et de son efficacité. Il permet d'offrir aux applications des garanties statistiques (suffisantes) à partir de descripteurs simples (tels que le débit crête des flots) sans recourir à de tels mécanismes. Il permet également une adaptation au trafic et à ses changements.

Un algorithme de MBAC doit être simple, extensible et robuste (notamment aux erreurs de mesure). La complexité réside principalement dans le maintien d'une liste des flots protégés (identification au vol, purge après un certain temps d'inactivité, etc.), puisque le nombre de flots en cours sur un lien peut être élevé. Il est cependant possible d'utiliser des technologies matérielles telles que celles présentées dans la section 2.4.1 du Chapitre 2, ou d'inscrire les flots dans la table de manière probabiliste (ce qui élimine un grand nombre de flots très courts sans trop affecter la performance du lien).

MBAC et différenciation implicite de service

Les deux modèles présentés précédemment pour gérer les flots *streaming* et élastiques reposent sur un moyen de limiter le nombre de flots en cours :

- pour les flots *streaming*, il s'agit de garder le lien transparent afin de préserver les conditions de multiplexage sans buffer ;
- pour les flots élastiques, cela permet d'assurer un débit minimal pour les flots en cours et de garantir le temps de complétion. Un grand nombre de flots en cours cause un grand nombre de pertes et de retransmissions, et peut conduire à l'abandon de connexions (impatience utilisateur ou applicative), gaspillant inutilement des ressources.

Les mesures fournies par l'algorithme PFQ sont adaptées au contrôle de la performance des deux classes de trafic :

- PL permet de contrôler la charge des flots *streaming* (envoyés dans la file prioritaire) avec la performance du multiplexage sans buffer ;
- FR est une mesure du débit à long terme des flots élastiques, ordonnancés selon une discipline FQ.

Le fonctionnement du contrôle d'admission et sa complémentarité avec l'ordonnanceur PFQ sont illustrés en figure 3.5.

Seuils d'admission

L'hypothèse générale est que **les flots *streaming* possèdent un débit limité $r < p$ typiquement faible par rapport à la capacité du lien**, et nécessairement inférieur au seuil sur le *fair rate*. Le contrôle d'admission permet de garantir que les flots de débit inférieur ou égal à p sont toujours traités en priorité, tandis que PFQ réagit à la surcharge en différenciant les flots de débits les plus élevés.

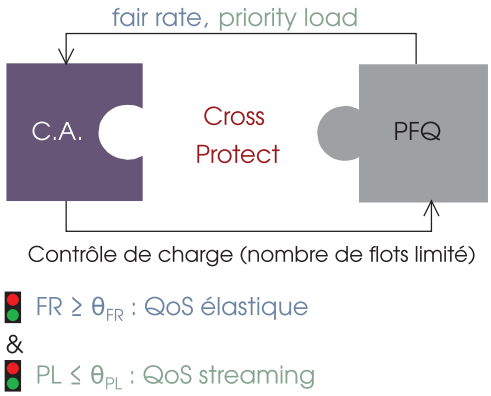


FIGURE 3.5 – Complémentarité du Contrôle d'Admission (C.A.) et de l'ordonnanceur PFQ au sein de Cross-Protect, et mention des critères d'admission.

Nous avons vu précédemment que le seuil sur le FR est peu sensible, et une valeur de 1% est satisfaisante pour la performance des flots en cours (voir Section 3.3.2).

Le seuil sur *PL* n'est pas critique du moment que le trafic est majoritairement élastique sur le lien ; c'est pourquoi la proposition initiale [91] fixe cette valeur à 70% de la capacité du lien. Dans le Chapitre 6, nous reconsidérerons cet aspect lorsque les flots sont majoritairement à débit limité, comme sur un lien de collecte ADSL par exemple.

3.6 Vers la réalisation d'un routeur Cross-Protect

3.6.1 Évaluation de l'architecture en simulation

L'architecture Cross-Protect a initialement été évaluée à l'aide du simulateur NS-2 [105], pour lequel plusieurs modules ont été proposés, afin d'avoir un cadre d'évaluation complet, notamment utilisé dans les chapitres suivants :

- algorithmes de contrôle d'admission, PFQ et de leur combinaison ;
- outils de génération de trafic respectant les modèles que nous avons présentés ;
- modules et outils permettant le suivi des flots ainsi que le calcul de diverses métriques de performance ;
- un module permettant la simulation d'un grand nombre de flots à débit d'accès limité ;
- un module permettant le rejeu de traces au niveau paquet et flot, avec conservation des caractéristiques telles que le débit crête ;
- un module reproduisant le comportement de flots TCP avec retentatives.

3.6.2 Évaluation de l'architecture par expérimentation

En plus d'un module pour le simulateur NS-2, les mécanismes Cross-Protect ont été réalisés au sein de l'architecture de contrôle de trafic du noyau Linux (`iptables` et `traffic control`)[5], afin de construire une plate-forme d'expérimentation et d'évaluation de la solution. Les objectifs d'une telle réalisation sont multiples :

- convaincre de la faisabilité de l'architecture, et notamment du traitement des flots, à haut débit ;
- démontrer le fonctionnement de Cross-Protect dans des conditions réelles,
- analyser l'évolution des différentes structures de données (notamment les tables de flots) ;
- cerner les contraintes d'implémentation de l'architecture, notamment en identifiant les différences entre routeur purement logiciel et routeur spécialisé ;
- mieux comprendre les contraintes posées par le trafic (identification des flots, encapsulation, chiffrement, etc.) et les applications le générant (téléphonie IP, *streaming* vidéo).
- enfin, évaluer différentes propositions pour les briques constituant l'architecture.

Cette réalisation a été limitée aux flots TCP, UDP et ICMP de IPv4, qui sont reconnus par le quintuplet classique (adresses et ports source et destination + protocole, ou, dans le cas d'ICMP, adresses source et destination + type et code ICMP + protocole), et cohérents dans le temps (*timeout*). Une possible utilisation du *flow label* d'IPv6 n'a pas été considérée pour l'évaluation.

5. voir `http://netfilter.org/`

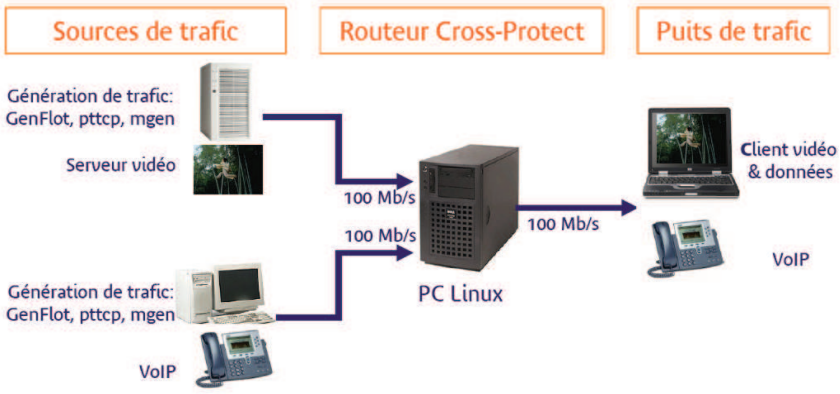


FIGURE 3.6 – Plateforme de démonstration Cross-Protect

Plate-forme d'expérimentation

Cette réalisation a été intégrée au sein d'une plateforme plus complète d'expérimentation [8], illustrée en figure 3.6. Elle est constituée d'un routeur central constituant un lien par lequel transite le trafic entre plusieurs sources et puits de trafic : téléphones IP, serveurs de fichiers ou de vidéo *streaming*, applications de génération de trafic et de visualisation (certaines étant communes avec le simulateur).

Évaluation

Cette évaluation expérimentale de l'architecture Cross-Protect au travers de la plateforme GNU/Linux a pu donner lieu à deux démonstrations. Les évaluations par expérimentation qui ont été effectuées et démontrées sont incluses dans la thèse d'Abdesselem Kortebi et par conséquent non répétées ici. Le document est disponible en ligne [6].

6. `http://jordan.auge.free.fr/files/these_kortebi.pdf`

Chapitre 4

Dimensionnement des buffers pour les routeurs IP de cœur de réseau

Le contrôle de congestion dans l'Internet repose principalement sur les mécanismes implémentés par TCP, auxquels s'ajoute une limite au débit des flots imposée par les débits d'accès. La performance des flots, et notamment du contrôle effectué par TCP, nécessite un dimensionnement correct des buffers au sein des routeurs de cœur de réseau, qui sont gérés en FIFO. La règle empirique de dimensionnement traditionnellement utilisée – dite du *Bandwidth Delay Product* (BDP) – montre ses limites avec l'accroissement constant des capacités des liens, et les contraintes techniques qui limitent la taille des buffers. D'abord mis en évidence par l'article de Appenzeller *et al.* [7], le problème du dimensionnement a depuis suscité une attention croissante.

Dans ce chapitre, nous examinons le problème au vu de notre compréhension des caractéristiques du trafic, et de la performance du partage de bande passante statistique. Nous montrons au moyen d'un modèle analytique simple couplé aux résultats de simulations ns2 que, tandis qu'un buffer de taille équivalente au BDP n'est pas forcément nécessaire, la proposition récente de les réduire à quelques dizaines de paquets est certainement trop drastique. La taille nécessaire pour le buffer dépend de manière significative du débit crête exogène des flots multiplexés.

Sommaire

4.1 Introduction **46**
4.2 Dimensionnement des buffers et TCP **47**
4.3 État de l'art sur le dimensionnement de buffers **48**
4.4 Impact de la taille du buffer sur la performance des flots **49**
4.5 Dimensionnement des buffers dans le régime transparent **50**
4.6 Dimensionnement des buffers pour le régime élastique **52**
4.7 Conclusions **59**

Publications et présentations :

Ce chapitre à fait l'objet des publications suivantes :

- Jordan Augé, James Roberts, **Buffer Sizing for Elastic Traffic**, *NGI2006, 2nd Conference on Next Generation Internet Design and Engineering, València, April 3-5 2006*
- Jordan Augé, James Roberts, **A Statistical Bandwidth Sharing Perspective on Buffer Sizing**, *ITC'20 – Ottawa, Canada – June 17-21, 2007*

4.1 Introduction

Le dimensionnement des buffers au sein des routeurs a reçu beaucoup d'intérêt récemment [7, 129, 163, 128, 49, 3, 2], avec notamment la remise en cause de la règle empirique dite du *Bandwidth Delay Product* initialement proposée par Villamizer *et al.* [149]. La réalisation de grand buffers est un défi technique pour des liens ayant des débits de plusieurs dizaines de gigabits par seconde. C'est un but pour l'instant inaccessible pour des routeurs purement optiques qui constitueront vraisemblablement la prochaine génération d'équipements. La pertinence de cette règle empirique est remise en cause au vu de l'évolution des capacités des liens de cœur de réseau et du volume de trafic transporté. Il serait par exemple moins coûteux d'accepter de sacrifier un petit pourcentage de l'utilisation, quitte à prévoir un supplément de bande passante.

Qualité de service

Lors des périodes de congestion, le lien est utilisé à 100% et des paquets s'accumulent dans le buffer. Comme nous l'illustrerons dans le chapitre suivant, sans différenciation, la présence de buffers importants est une source de délais et de gigue pour les flots *streaming*, aggravée par les émissions en rafales des flots TCP. Réduire la taille des buffers permettrait de mitiger ces problèmes. Les conséquences d'une telle réduction sur la performance du trafic élastique restent un sujet d'étude, auquel nous consacrons une grande partie de ce chapitre. Néanmoins, un dimensionnement correct de ces buffers n'éliminera pas les pertes de paquets subies par les flots lors des instants de saturation du buffer causés par des rafales.

Avrachenkov *et al.* [9] présentent une analyse des performances de TCP en fonction de la taille du buffer. Une analyse par point fixe à partir d'un modèle assez complexe de TCP conclut sur l'inefficacité de buffers soit trop petits, soit trop grands. Elle est confirmée par des simulations ns-2. Si l'on se réfère à l'explication donnée précédemment pour justifier la règle du *Bandwidth Delay Product*, un buffer trop petit ne permet pas à TCP d'utiliser la totalité de la bande passante disponible (pendant les périodes où il n'émet pas de paquets), et il causera un fort taux de pertes étant souvent saturé. Réciproquement un buffer surdimensionné sera rarement vide et allongera les délais subis par les paquets, d'autant que la majorité des flots qui sont contraints par leur débit d'accès n'en profiteront pas.

La règle du *Bandwidth Delay Product* nécessite de connaître le RTT moyen des flots sur le lien, qui n'est pas facile à caractérise. Les effets d'un sous-dimensionnement étant considérés plus graves (utilisation du lien, équité entre les flots) que ceux d'un surdimensionnement, il est fréquent de prendre une valeur moyenne pour le RTT de 200 à 250ms (de l'ordre du temps de propagation transatlantique).

Conséquences sur la réalisation de routeurs IP

On trouve aujourd'hui des liens OC768 correspondant à un débit de l'ordre de 40Gb/s. En reprenant l'exemple cité dans [7], un lien de 10Gb/s et de RTT moyen 250ms recevra par la règle du BDP une taille de buffer de 2.5Gb. La présence de tels buffers dans les routeurs pose des problèmes technique de réalisation (vu la technologie actuelle des mémoires), d'encombrement et de chaleur. De plus l'évolution constante des capacités des liens remet en cause la pérennité d'une telle approche, et montre l'intérêt d'opérer avec de plus petits buffers. En outre, l'émergence de routeurs tout-optique, pour lesquels on ne sait que réaliser des buffers de quelques dizaines de paquets, requiert de comprendre la performance du trafic en présence de telles tailles de buffers.

A taille de buffer donnée, la performance peut être caractérisée par les délais et taux de pertes de paquets, ou par le débit réalisé par les flots. Elle dépend significativement des hypothèses sur les caractéristiques du trafic. Les articles cités précédemment fondent leur raisonnement sur une grande diversité d'hypothèses de base, ce qui explique leurs conclusions conflictuelles. Notre but dans le présent chapitre est d'identifier les caractéristiques essentielles du modèle au niveau flot, et d'évaluer le compromis entre la taille du buffer et la performance obtenue sous des hypothèses réalistes de trafic.

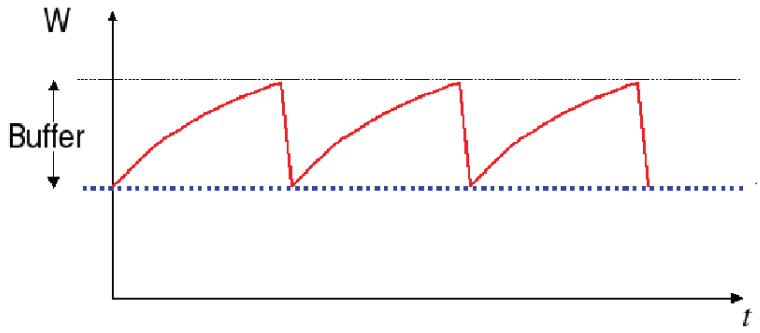


FIGURE 4.1 – Évolution de la taille de la fenêtre de congestion TCP sur un lien avec buffer.

4.2 Dimensionnement des buffers et TCP

Le trafic observé sur un lien de cœur de réseau est majoritairement composé de flots TCP. Il n'est donc pas étonnant que son comportement ait été au centre de nombreuses études et qu'il ait à lui seul dicté le dimensionnement des buffers et entraîné la règle dite du *Bandwidth Delay Product* (BDP). Nous verrons pourtant dans la suite de ce chapitre qu'un tel modèle n'est pas forcément réaliste et qu'il convient de mieux caractériser le trafic et les interactions entre les différents flots afin de pouvoir comprendre la performance réalisée.

4.2.1 Dimensionnement pour un flot TCP

La règle empirique du BDP a été proposée par Villamizar et Song [149] et suggère une taille de buffer calculée en fonction du produit entre le débit du lien multiplié par le délai moyen. L'objectif est d'assurer une utilisation totale du lien par une connexion TCP suffisamment longue [1].

La figure 4.1 [2] représente l'évolution typique de la taille de la fenêtre de congestion d'une connexion TCP seule sur un lien, dans le mode *congestion avoidance*. On y reconnaît le motif caractéristique de TCP en dent de scie dû à l'algorithme AIMD. Les instants de décroissance correspondent à la détection par TCP d'une congestion (perte d'un paquet, reconnue par la réception de trois acquittements identiques). La taille de la fenêtre de congestion *cwnd* est alors divisée par deux, et aucun nouveau paquet n'est émis jusqu'à ce qu'un nombre suffisant d'acquittements soient reçus et que le nombre de paquets en transit soit à nouveau inférieur à *cwnd*. La règle du BDP vise à dimensionner le buffer de telle façon qu'il emmagasine suffisamment de paquets à transmettre pour compenser la période pendant laquelle la connexion n'émettra pas.

Puisque *cwnd* régule l'envoi des paquets pendant chaque RTT, le débit d'une connexion TCP est $cwnd/RTT$. Notons $cwnd_p = cwnd$ lors de la détection d'une perte ; TCP n'émet plus de paquets le temps de recevoir $cwnd_p/2$ acquittements, en raison de la réduction de *cwnd* de moitié, qui arrivent au débit du lien C. La taille du buffer B doit donc être suffisante pour permettre l'émission de paquets au débit C durant cette période : $B \geq cwnd_p/2$ paquets. Le cas idéal correspond à un buffer vide lorsque la source émet à nouveau au débit du lien : $C = cwnd_p/2RTT$. Nous avons $cwnd_p = 2C \cdot RTT$ et ainsi $B = C \cdot RTT$.

4.2.2 Synchronisation des flots TCP : extension à N flots

Le raisonnement que nous venons de présenter est valable lorsqu'un seul long flot TCP occupe la totalité de la bande passante disponible sur le lien. Il se maintient toutefois dans le cas de plusieurs flots TCP simultanés en considérant un phénomène de synchronisation entre les connexions. Sur un lien FIFO/DropTail, les flots ont une tendance à tous subir des pertes au même instant de congestion, ce qui entraine une évolution similaire des fenêtres de congestion au cours du temps. Il suffit alors de considérer la somme de l'ensemble des fenêtres de congestion, qui connait également une évolution en dent de scie, pour justifier l'application du *Bandwidth Delay Product*.

Ce phénomène est généralement exhibé dans des simulations avec des connexions permanentes, mais est moins connu avec du trafic aléatoire (arrivées Poisson de session par exemple). Il est remis en cause dans [7] dans le cas d'un lien de cœur de réseau où de très nombreuses connexions sont en cours et se partagent la capacité disponible (un tel cas est irréaliste sous nos hypothèses de trafic). L'étude

1. c'est-à-dire pour que la connexion sorte du mode *slow start* de TCP en faveur du mode *congestion avoidance*. Les flots courts voient en effet leur débit conditionné principalement par l'évolution de la fenêtre de congestion pendant *slow start*.
2. Cette figure est extraite de la présentation de [7] à la conférence Sigcomm'04.

est approfondie dans [129] afin d'établir le lien entre la synchronisation des flots et le problème de dimensionnement des buffers.

Dans la suite de ce chapitre, nous étudions la performance d'un environnement FIFO/DropTail, mais les résultats obtenus restent compatibles avec l'utilisation d'un ordonnancement *fair queueing*. Ils seront généralement différents avec l'utilisation d'autres mécanismes de gestion du trafic tels que RED, etc. Le chapitre suivant considérera l'impact de l'introduction de nouveaux protocoles TCP, et d'un ordonnancement équitable.

4.3 État de l'art sur le dimensionnement de buffers

La problématique du dimensionnement des buffers à suscité un intérêt croissant suite aux travaux de Appenzeller *et al.* [7]. Nous présentons ici les résultats les plus significatifs, qui peuvent sembler contradictoires à première vue. Nous verrons plus loin qu'ils proviennent en fait d'hypothèses différentes sur le trafic.

De nombreux articles envisagent des approches adaptatives de dimensionnement des buffers, se basant sur l'observation de la complexité et de la variabilité du trafic. Nous nous intéressons à l'allocation physique de mémoire au sein d'un routeur, et non à sa gestion. L'algorithme PFQ, que nous avons présenté dans le chapitre précédent, remplit ce rôle d'allocation et permet une isolation entre les différents flots.

4.3.1 Dimensionnement proportionnel au nombre de flots

La proposition de Morris [111] s'appuie sur la constatation suivante : si une connexion TCP a une taille de fenêtre de congestion inférieure à quelques segments, elle subit de fortes pertes et souffre de fréquents *timeouts* de retransmission qui interrompent le fonctionnement en *congestion avoidance*. Un tel régime dégradé peut être évité si l'on permet à chaque connexion d'émettre quelques segments dans le buffer, et l'auteur recommande ainsi un dimensionnement proportionnel au nombre de flots.

Un buffer suffisamment dimensionné est crucial pour des raisons de performance et d'équité entre les flots. Par exemple, il est possible de limiter le taux de pertes à 2% avec un buffer 6 fois proportionnel au nombre de flots, et à 1% pour un facteur 9.

L'article ne précise toutefois pas quels flots doivent être considérés, et nous pouvons supposer qu'il s'agit de ceux partageant effectivement la bande passante sur le lien, que nous qualifions de *bottlenecked*.

4.3.2 Vers une réduction drastique de la taille des buffers

Appenzeller *et al.* [7] mettent en évidence la difficulté de réaliser des buffers dimensionnés selon le BDP, au vu de l'évolution des débits des liens. Ils questionnent d'autant son utilité sur des liens de cœur de réseau où un très grand nombre de flots sont en compétition.

Ils montrent qu'au delà d'une centaine de flots en compétition, les évolutions des différentes fenêtres de congestion *cwnd* des flots TCP sont désynchronisées, et qu'une taille de buffer proportionnelle à la racine carrée du nombre de flots suffit. Ils remarquent l'intérêt d'estimer l'équité réalisée par les connexions pour des tailles de buffer intermédiaires.

Le phénomène peut se comprendre intuitivement en considérant le comportement en "dents de scie" de l'évolution de *cwnd* : la taille requise par un dimensionnement selon le BDP correspond à la diminution de *cwnd* lors d'une perte. Lorsque les connexions sont nombreuses et désynchronisées, la superposition des différentes fenêtres *cwnd* permet une atténuation du phénomène, qui suggère des besoins moins importants en buffer pour garantir la performance des flots. En supposant que les flots sont identiques, la somme des tailles des fenêtres tend vers une distribution normale (théorème central limite), de laquelle se déduit la proposition de dimensionnement. Appenzeller *et al.* [7] utilise un tel modèle afin d'estimer la probabilité de sous-utilisation du lien.

Les auteurs questionnent également la validité du dimensionnement lorsque les flots sont courts et leur débit conditionné par le *slow start* de TCP. Le modèle utilisé est une file M/G/1, ils montrent alors la dépendance du dimensionnement à la charge et à la longueur des rafales.

L'étude d'un modèle mixte mélangeant flots longs et flots courts est complexe. Une hypothèse avancée et confirmée au travers de simulations par les auteurs est qu'il est suffisant de ne considérer que les longs flots pour le dimensionnement d'un routeur de cœur de réseau. D'autres simulations montrent que les performances avec des flots courts sont meilleurs avec de petits buffers. Il serait intéressant de vérifier ces résultats plus formellement.

Dans la suite, nous qualifierons de telles tailles de buffer de "moyennes".

Quelques remarques sur les modèles de trafic

Il a été remarqué, cependant, par Raina et Wischik [129] ainsi que Raina *et al.* [128] que la synchronisation des flots dépend de la taille du buffer et que, pour un très grand nombre de flots, le dimensionnement proposé dans [7] est toujours trop important. Ils suggèrent que la capacité du buffer doit être réduite à quelques dizaines de paquets. Aucune instabilité n'a été observée dans [7] parce que les auteurs ont seulement effectué des simulations avec quelques centaines de flots alors que le phénomène se manifeste pour quelques milliers. Dans le chapitre 3, nous avons expliqué pourquoi il n'est pas raisonnable de supposer que tant de flots sont en fait *bottlenecked* sur le lien, ce qui nous permet de remettre en cause la validité de cet argument favorable à de petits buffers.

Une publication ultérieure [162] précise que des buffers de 20 paquets suffisent si les paquets émis par les flots sont espacés suffisamment régulièrement (*pacing*). Sinon, il convient plutôt de considérer une taille de l'ordre de 20 rafales. L'accent est mis sur l'instabilité des tailles intermédiaires de buffers.

Nous qualifierons de telles tailles de buffer de "petites" dans la suite du chapitre.

4.3.3 Vers une distinction flots *bottlenecked/non-bottlenecked*

Dhamdhere et al. [3] reconnaissent l'importance de distinguer les flots *bottlenecked* des flots *non-bottlenecked*. Ils suggèrent qu'il est nécessaire d'obtenir de faibles taux de pertes tout en maintenant une utilisation élevée des liens. En conséquence, ils recommandent d'avoir un buffer relativement grand qui est proportionnel au nombre de flots *bottlenecked*. Les auteurs poussent plus en avant leur analyse dans [2], notamment en introduisant des modèles de trafic dynamiques au niveau flot, *open* et *closed loop* qui correspondent à notre notion de partage statistique de bande passante.

Cependant le modèle dans [3] est annoncé valide lorsque les flots *bottlenecked* constituent plus de 80% de la charge du lien et la performance est évaluée dans [2] dans un cas de très forte charge (où environ 200 flots *bottlenecked* sont en compétition). Une nouvelle fois nous remettons en cause la pertinence de ces hypothèses sur le trafic en vue d'évaluer les besoins en buffer.

4.3.4 Cas où les flots sont tous *non-bottlenecked*

Le modèle proposé par Enachescu *et al.* [49, 50] évalue la taille de buffer nécessaire lorsque les capacités des réseaux d'accès et de cœur sont d'ordres de grandeur différents : les paquets se retrouvent alors naturellement espacés lors de leur transmission. Il est également possible de recourir à des piles TCP modifiées qui espacent les paquets au lieu de les envoyer par rafales (*pacing*).

Lorsque les paquets sont espacés au débit moyen déterminé par la taille de la fenêtre plutôt qu'émis en rafales, [50] suggère que la taille du buffer doit être proportionnelle au logarithme de la fenêtre de congestion maximale de TCP.

Cette hypothèse est licite pour de nombreux liens et correspond à ce que nous appelons le régime transparent (Ch.2, Sec.2.2.3) : tous les flots sont *non-bottlenecked*. Nous remarquons que les analyses dans [129, 128] se basent sur un modèle fluide et supposent ainsi implicitement que les paquets n'arrivent pas par rafales.

4.4 Impact de la taille du buffer sur la performance des flots

4.4.1 Régimes opérationnels

Si l'utilisation globale du lien est préservée pour des buffers de taille moyenne, ou en grande partie pour de petites tailles de buffers, nous ne savons rien de la performance individuelle de chaque flot : un débit satisfaisant et équitable pour les flots élastiques, ainsi que des délais et taux de pertes négligeables pour les flots *streaming*.

En fonction de la charge offerte et des débits des flots considérés, le lien se retrouvera dans l'un des trois régimes de bande passante, que nous avons introduits dans le Chapitre 2 :

transparent : La somme des débits des flots est inférieure à la capacité du lien, et il est possible de dimensionner le buffer afin de ne gérer que les arrivées simultanées de paquets (multiplexage des flots sans buffer). En assurant des délais et taux de pertes négligeables à l'ensemble du trafic, on garantit également la performance des flots élastiques. L'ensemble du trafic conserve son débit exogène à la traversée du lien.

élastique : La taille nécessaire des buffers doit être évaluée pour un mélange de flots *non-bottlenecked* – pour lesquels les paquets sont espacés au débit du flot – et de flots *bottlenecked* – qui généralement émettent leurs paquets par rafales – correspondant à un mélange caractéristique d'une charge réaliste sur un lien de cœur de réseau. Il est notamment important de tenir compte des émissions par rafales des paquets appartenant aux connexions TCP.

surcharge : il s'agit d'une situation anormale devant être gérée, par exemple, par un contrôle d'admission ; nous ne nous intéresserons pas à ce régime pour le dimensionnement des buffers.

4.4.2 Performance des flots *streaming*

Nous supposons que la charge des flots *streaming* reste inférieure à la capacité du lien. Cette hypothèse est réaliste dans des conditions normales pour le réseau, et garantie par l'utilisation d'un mécanisme de contrôle de la surcharge, tel qu'un contrôle d'admission.

Lorsque le lien n'est pas saturé, en régime transparent, le trafic ne s'accumule pas dans le buffer et les flots concurrents n'ont qu'un impact négligeable sur la performance, pourvu que le buffer soit suffisamment dimensionné pour absorber les arrivées simultanées.

Lorsque des flots possèdent un débit crête suffisamment élevé et/ou sont en nombre suffisant pour saturer le lien (régime élastique), le buffer se remplit et la performance des flots *streaming* se retrouve dégradée :

- ils subissent des pertes aux instants où le buffer est plein (ces instants sont plus ou moins longs en fonction de la durée des rafales TCP).
- ils subissent des délais importants, ainsi que de la gigue produite par les oscillations du buffer (voir [129]).

Nous présentons des simulations de telles situations dans le Chapitre 5, et montrons l'intérêt d'introduire un ordonnancement de type *fair queueing* pour y remédier : les flots se retrouvent isolés, et leur performance est alors indépendante de la taille du buffer.

4.4.3 Performance des flots élastiques

Les modèles qui considèrent un très grand nombre de flots permanents, comme par exemple [7], ne correspondent pas à ce que l'on peut observer sur un lien de cœur de réseau à un niveau de charge nominal. Nous considérons un modèle de trafic réaliste tel que décrit dans le chapitre précédent.

La majorité des flots ne sont pas *bottlenecked*. Ces flots émettent des paquets de temps à autre ; ils transitent dans le buffer lorsque ce dernier est saturé, et ne s'y accumulent généralement pas si le lien n'est pas surchargé. La participation de ces flots à l'allocation de bande passante faite par TCP est négligeable. Ces flots conservent leur débit exogène, à l'exception des instants où ils subissent des pertes.

Ces sont les flots de plus fort débit (leur nombre dépend de la charge du lien) – qui accumulent un grand nombre de paquets dans le buffer et causent des pertes – qui réalisent un partage des ressources. Une première approximation est de considérer qu'ils utilisent la capacité laissée disponible par les autres flots. Cette hypothèse est pratique, et donne de bons résultats dans la mesure ou les instants de pertes restent des événement rares. Elle devient réaliste avec l'utilisation de *fair queueing*. Il est alors possible de considérer que les flots *non-bottlenecked* constituent un trafic de fond de charge donnée [3].

Une analyse de l'équité du partage entre les flots est faite dans le Chapitre 5. Nous verrons que l'utilisation de *fair queueing* permet un multiplexage efficace des flots, et garantit leur isolation et l'équité de leurs débits.

Dans la suite de l'évaluation, nous acceptons comme dans [7] de sacrifier une fraction du taux d'utilisation du lien si ce choix permet une diminution importante de la taille des buffers : une telle décision peut être économiquement rentable et permettre la conception de routeurs de capacité plus importante. Nous allons pour les régimes transparent et élastique examiner si la règle du BDP reste de rigueur ou non.

4.5 Dimensionnement des buffers dans le régime transparent

Un régime transparent à été défini plus tôt tel que la somme des débits crête des flots reste inférieure à la capacité du lien avec une forte probabilité. Le buffer doit être dimensionné de telle sorte à absorber des arrivées de paquets simultanées, provenant de flots indépendants. Nous supposons que le débit

3. On peut supposer que la charge induite par les flots *streaming* n'est pas affectée par les pertes qu'ils subissent (modèle *open loop*), et que les pertes sont faibles et distribuées sur un grand nombre de flots pour les flots élastiques de faible débits, d'où une influence négligeable sur la charge.

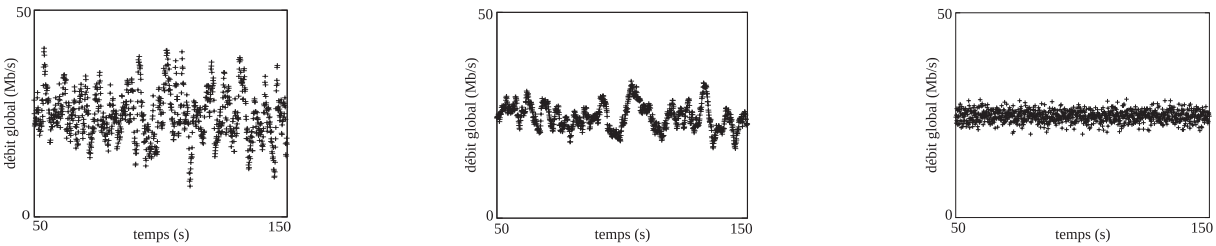


FIGURE 4.2 – Processus d'arrivée des paquets : débit des paquets arrivant dans des intervalles successifs de 100ms pour des flots de débit crête $p = 200$Kb/s (à gauche), $p = 50$Kb/s (au milieu) et $p = 0$ (à droite).

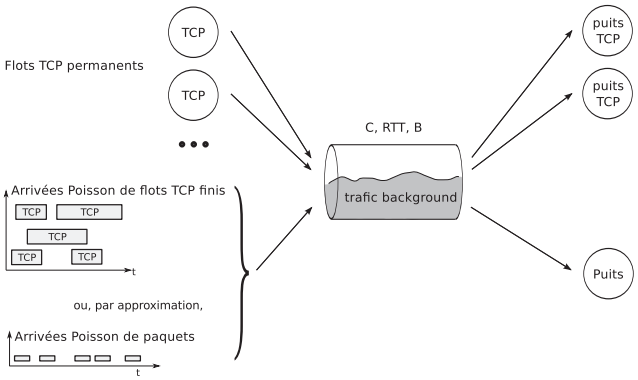


FIGURE 4.3 – Environnement de simulation : sauf précision supplémentaire, les simulations utilisent les paramètres suivants : $C = 50$ Mb/s, $B = 20$ paquets, $RTT = 100$ ms, $\rho_b = 0.5C$, ordonnancement FIFO, paquets de 1000 octets.

crête des flots est bien défini à l'échelle de temps de l'émission des paquets comme, par exemple, lorsqu'il est limité par un lien d'accès en amont.

4.5.1 Arrivées localement Poisson

La figure 4.2 décrit le débit global d'une superposition de flots TCP Reno ayant un débit crête limité, en utilisant l'environnement de simulation présenté en figure 4.3. Sans trafic *bottlenecked*, les flots arrivent selon un processus de Poisson et leur taille est tirée aléatoirement depuis une distribution exponentielle de moyenne 100 paquets. Les figures représentent le débit moyen dans des intervalles successifs de 100ms. Le débit est montré pour deux débits crête, $p = 200$Kb/s et $p = 50$Kb/s, ainsi que celui mesuré pour pour des arrivées de paquets selon un processus de Poisson ($p = 0$).

Un processus d'arrivée de paquets Poisson n'est visiblement pas une bonne approximation à moins que le débit crête des flots ne soit relativement faible par rapport au débit du lien. Cependant, dans un petit intervalle de temps (par exemple dans chaque intervalle de 100ms), le processus d'arrivée des paquets, en tant que superposition d'un grand nombre de processus périodiques, est approximativement Poisson. La figure présente une réalisation de ce processus de Poisson modulé. Nous notons son intensité Λ_t.

Une telle supposition permet d'approximer l'occupation du buffer localement par celle d'une file M/G/1. Si l'on simplifie d'avantage en supposant des tailles de paquets exponentielles, la probabilité de perte d'un paquet étant donné un buffer B est approchée par la formule $(\Lambda_t/C)^B$.

4.5.2 Calcul de la taille requise du buffer

Un approche pour le dimensionnement est de calculer un taux de perte moyen en conditionnant sur la distribution $F(\lambda)$ de Λ_t, avec pour objectif que $\int(\lambda/C)^B dF(\lambda) < \epsilon$. C'est un choix raisonnable quand les variations de débits sont rapides de telle sorte que ϵ est une bonne mesure de la performance d'un flot choisi arbitrairement. Il s'avère que, pour un débit crête inférieur à $.1C$ et $\epsilon > .001$, la taille requise en buffer est la même que celle qui aurait été nécessaire pour un processus d'arrivée Poisson. En d'autres termes, la formule de la M/M/1 $\rho^B < \epsilon$ demeure un critère de dimensionnement utile. Par

exemple, un buffer de 20 paquets limite la charge admissible à $\rho = .79$ pour $\epsilon = .01$ ou $\rho = .7$ pour $\epsilon = .001$.

Wishik [162] propose une autre justification pour cette hypothèse Poissonnienne, qui peut sembler contradictoire avec la présence généralement acceptée de dépendance à long terme au sein du trafic Internet. D'après les auteurs, ce phénomène ne se manifeste vraiment qu'à de plus grandes échelles de temps, alors qu'ici on s'intéresse à une granularité plus faible. En effet, de petits buffers seront rapidement saturés, et les variations de la file d'attente seront très rapides, comparativement à la charge du lien (plusieurs RTT). Le taux de perte sera ainsi très proche de celle d'une file connaissant des arrivées Poisson.

4.6 Dimensionnement des buffers pour le régime élastique

Il n'est pas possible dans le cas général de garantir que le débit crête des flots est limité sur un lien. Même les réseaux d'accès ADSL transportent des tunnels agrégeant d'autres connexions. Il se peut alors que certains flots saturent (temporairement) le lien et voient leurs paquets stockés dans le buffer. Le taux de pertes de paquets est alors fortement dépendant de la taille de ce buffer. Nous considérons une charge de lien inférieure à 1 (régime stable).

Il est alors nécessaire de comprendre l'impact de la taille des buffers sur la performance du régime élastique – c'est-à-dire lorsqu'un ou plusieurs flots *bottlenecked* se combinent avec la charge *background* pour saturer momentanément le lien pendant des périodes qui sont longues comparées à l'échelle de temps d'émission des paquets. Le comportement de chaque type de flot est différent, et leur performance est mal connue dans le cas général.

Les flots *non-bottlenecked* n'atteignent pas le débit équitable, notamment à cause des contraintes en débit qu'ils subissent, mais aussi à cause de pertes dues à la saturation du buffer. Si ces pertes ne se produisent que de temps en temps (aux instants de saturation) et n'affectent qu'un petit nombre de flots, elles sont toutefois néfastes pour ces flots qui ont déjà un débit restreint. Elles seront d'autant plus importantes que les protocoles de transport mis en œuvre pour le transfert des flots à fort débit seront agressifs. Nous envisagerons ce cas dans le prochain chapitre.

En ce qui concerne les flots *bottlenecked*, nous allons voir que la présence de trafic concurrent (*background*) n'est pas sans conséquence sur le débit réalisé. Nous commençons par analyser quelques résultats simples de simulation, avant de proposer une étude plus exhaustive basée sur un modèle PS. Cette dernière nous permettra de proposer des recommandations pour un dimensionnement raisonnable des buffers.

4.6.1 Comportement des flots *bottlenecked*

Environnement de simulation

Afin de simplifier l'analyse et la discussion, nous supposons une stricte dichotomie entre ces deux classes de flots. Les flots *non-bottlenecked* représentent une charge *background* (ρ_b) qui est par hypothèse fixe (régime quasi-stationnaire), le reste de la bande passante est partagé par les flots *bottlenecked*. Les flots *non-bottlenecked* sont en grand nombre et possèdent chacun au maximum un paquet dans le buffer. Si leur taux de perte n'est pas trop élevé, la charge offerte au lien ne sera pas trop impactée. La présence de *fair queueing*, que nous considérons dans le chapitre suivant, permettra de rendre cette hypothèse plus robuste.

Le scénario de simulation s'appuie sur une topologie dite *dumbbell* (Fig. 4.3) avec un lien central de 50Mb/s et un RTT égal à 100ms. Les flots *non-bottlenecked* sont des flots TCP de taille finie et de débit égal à 1Mb/s, arrivant selon un processus de Poisson. Cependant, afin de faciliter l'évaluation d'un grand nombre de configurations, nous avons remplacé le processus au niveau flot (qui résulte en un trafic *background* de débit variable) par un processus d'arrivée de paquets Poisson de même intensité. Les résultats présentés ici ne sont pas affectés par cette simplification.

Dans les simulations qui suivent, le trafic *background* occupe la moitié de la capacité disponible ($\rho_b = 0.5$). Nous avons simulé 1, 2 et 4 flots TCP NewReno permanents, chargés de représenter différents états du lien en régime quasi-stationnaire. En effet, sur une période grande par rapport à l'échelle de temps des variations, le nombre de flots peut être considéré constant, et TCP est supposé atteindre rapidement un régime stationnaire. La performance des flots est analysée lorsque l'on fait varier la taille du buffer, entre 20 paquets (la valeur recommandée dans [129]) et un dimensionnement selon le *Bandwidth Delay Product*. Ces quelques simulations nous permettent déjà de tirer un certain nombre de remarques sur la performance des flots.

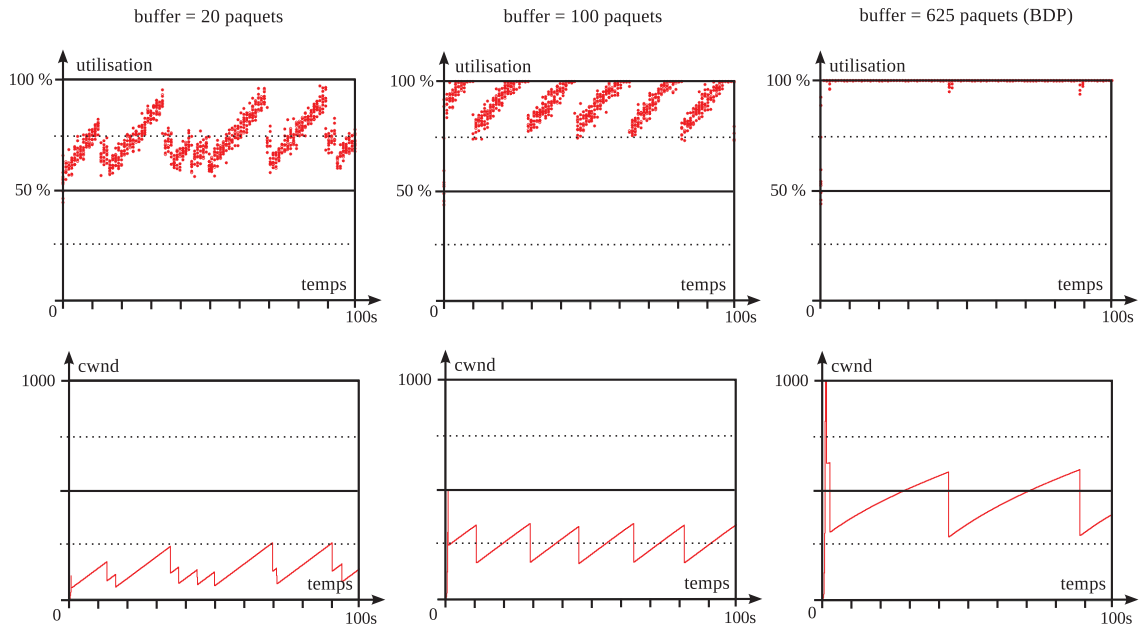


FIGURE 4.4 – De gauche à droite, taux d'utilisation et taille de la fenêtre de congestion (*cwnd* en fonction du temps, pour différentes tailles de buffer : 20, 100 and 625 paquets (BDP).

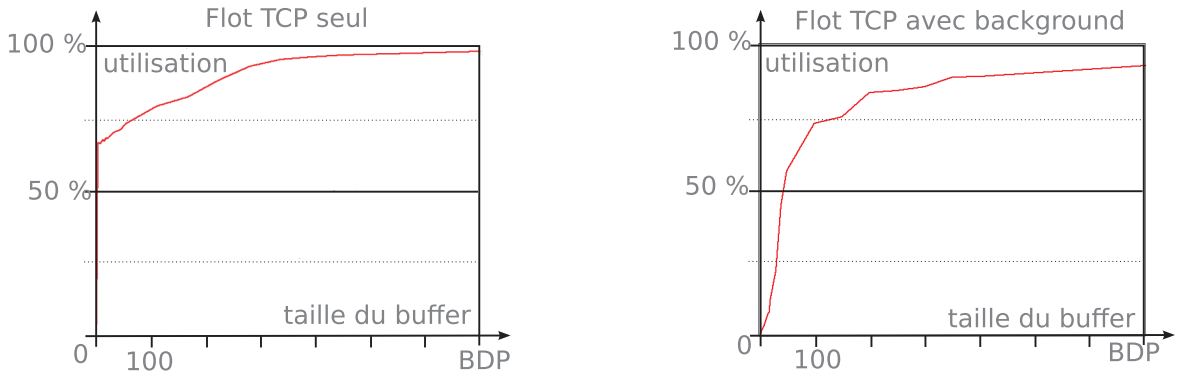


FIGURE 4.5 – Comparaison de l'utilisation de la bande passante disponible par un flot TCP, avec ou sans trafic concurrent

Impact du trafic *background* sur le comportement d'un flot TCP

La figure 4.4 illustre l'impact de la taille du buffer B sur l'évolution de la fenêtre de congestion *cwnd* (graphe du bas), ainsi que l'utilisation du lien moyennée sur un RTT (graphe du haut) pour un seul flot *bottlenecked*. Nous présentons des résultats pour trois tailles de buffers : petite (20 paquets, [129]), grande (625 paquets, correspondant au *Bandwidth Delay Product*), ainsi qu'une taille intermédiaire de 100 paquets.

Les résultats pour $B = 20$ montrent clairement que ce choix est inadapté au modèle de trafic considéré. Si l'on s'attend à ce que le flot n'utilise pas toute la capacité laissée disponible, l'importance de la dégradation de performance peut surprendre. Un flot TCP seul sur un lien vide disposant d'un buffer de 1 paquet devrait acquérir un débit voisin de 75% de la capacité disponible sur le lien (avec un raisonnement identique à celui permettant d'établir le BDP). Et nous nous attendons à ce que cette valeur dépasse les 80% pour un buffer de 20 paquets. Nous avons vérifié expérimentalement ces valeurs, en simulant un flot TCP seul sur un lien à 25 Mb/s, ainsi qu'un autre flot sur un lien à 50Mb/s où 50% de la charge est constitué de trafic *background*. Les résultats sont représentés sur la figure 4.5. On remarque que la présence de trafic *background* réduit considérablement le débit réalisé jusqu'à seulement 40% environ de la bande passante résiduelle.

D'autres simulations avec une charge *background* différente montrent que le taux d'utilisation de la

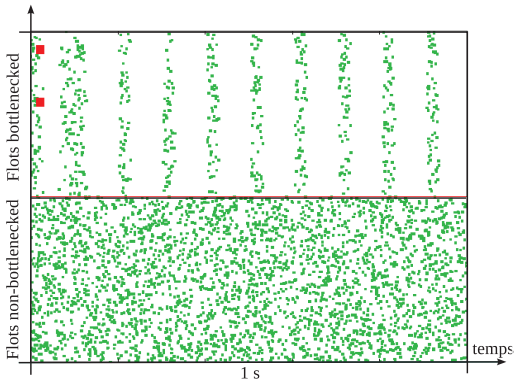


FIGURE 4.6 – Représentation des arrivées de paquets TCP NewReno sur un lien avec un buffer de 20 paquets. Chaque point est un paquet ; sa position verticale est aléatoire ; les gros carrées sur la gauche représentent des paquets perdus.

capacité résiduelle décroît avec l'augmentation de cette charge.

La raison est que le trafic *background* en compétition se combine avec les rafales des flots TCP *bottlenecked* pour saturer momentanément le lien. Nous illustrons ce phénomène dans le cas d'un seul flot *non-bottlenecked* en figure 4.6. Chaque point dans la moitié inférieure représente un paquet émis par un flot *background*, tandis que ceux de la moitié supérieure appartiennent au flot *bottlenecked*. La position sur l'axe des abscisses indique l'instant d'émission et celle sur l'axe des ordonnées est choisie aléatoirement. L'apparition de bandes reflète le mécanisme d'émission de la fenêtre TCP, qui malgré l'auto-régulation par le mécanisme d'acquittements, tend à émettre les paquets en rafales. Ces bursts sont de deux natures : des "micro-*bursts*" dus aux paquets émis dos à dos lorsque la fenêtre de congestion croît, qui possèdent souvent un débit supérieur au débit équitable (et d'autant plus fréquents en *slow start*) [6] ; ainsi que des "rafales sub-RTT", dont le débit moyen respecte le débit équitable, mais qui sont émises pendant un intervalle plus court que le RTT [77].

Tant que la fenêtre TCP reste petite comparée au *Bandwidth Delay Product* résiduel $C(1-\rho_b)\times$ RTT, ces rafales sont émises à partir d'instants séparés par un RTT. L'auto-ajustement de TCP fait que la somme des débits des rafales et du débit du processus *background* est très proche de la capacité du lien, voire supérieure. L'occupation du buffer a ainsi tendance à croître dans ces conditions de forte charge pendant que la rafale est en cours d'émission, puis à décroître quand le lien est à nouveau uniquement en présence d'arrivées du processus *background*.

En l'absence de perte, TCP augmente *cwnd* de 1 paquet par RTT, prolongeant ainsi la durée de la prochaine période de surcharge. A moment donné, les arrivées combinées des flots *background* et *bottlenecked* se combinent pour saturer le buffer et causer la perte d'un paquet. Pour un buffer de 20 paquets, cet événement se produit assez souvent, même pour de très faibles valeurs de *cwnd*. Dans la figure, la perte de paquet se produit à la fin de la première rafale que nous avons représentée. Elle est détectée un RTT plus tard ce qui mène à la diminution de moitié de la taille de la fenêtre courante. Sans la présence de trafic *background*, la perte ne se serait produite que lorsque *cwnd* aurait excédé le *Bandwidth Delay Product*.

Ces premiers résultats montrent que le protocole TCP majoritairement utilisé de nos jours n'est pas adapté à la présence de petits buffers dans le réseau. Le flot sort de sa période *slow start* prématurément, et la phase *congestion avoidance* sera fortement pénalisée par les pertes de paquets. L'inefficacité du protocole sera accentuée par la lenteur de la croissance de la fenêtre de congestion dans l'algorithme AIMD après une perte. Le choix d'un grand buffer nous permet ici d'assurer une utilisation totale du lien.

Une taille intermédiaire de 100 paquets apparaît comme un compromis raisonnable, à la fois au vu du débit réalisé mais aussi parce qu'elle permet d'assurer de plus faibles délais au trafic *streaming* potentiellement multiplexé avec les autres flots *background*.

Multiplexage de plusieurs flots *bottlenecked*

La figure 4.7 illustre la performance obtenue lorsque le nombre de flots *bottlenecked* multiplexés augmente. La taille du buffer est toujours de 20 paquets. On observe que même en présence de seulement deux flots, il y a très peu de pertes synchronisées et l'utilisation du lien s'améliore avec le nombre de flots. L'évolution de *cwnd* pour chaque flot montre que la bande passante est partagée approximative-

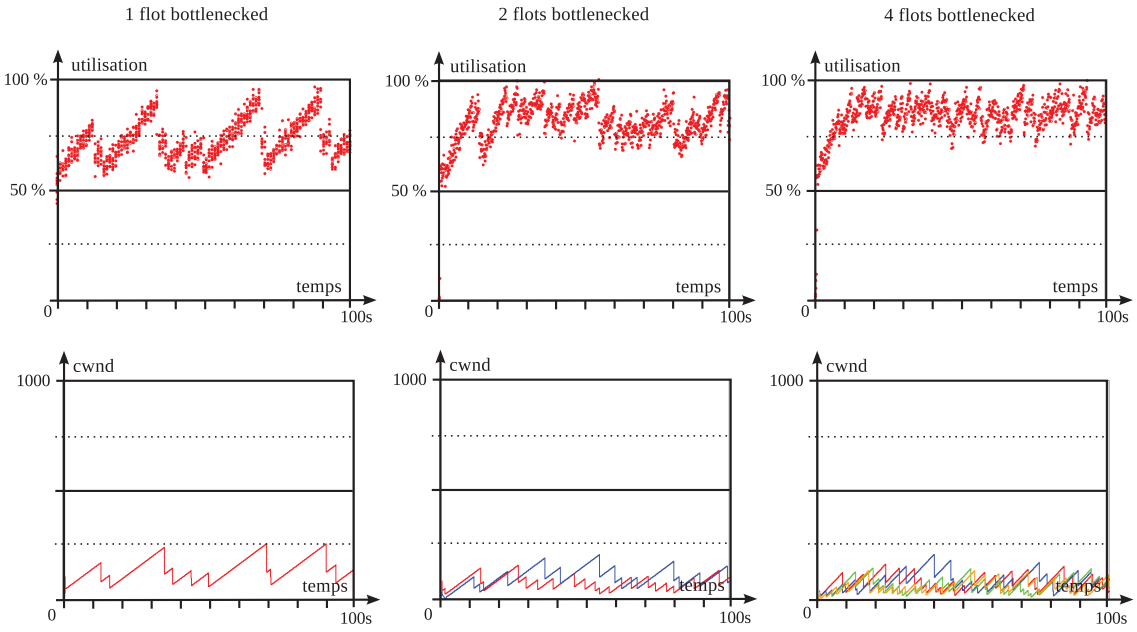


FIGURE 4.7 – De gauche à droite, utilisation et taille de fenêtre *cwnd* en fonction du temps pour 1, 2 et 4 flots *bottlenecked* avec un buffer de 20 paquets. Tous les flots utilisent TCP NewReno et l'ordonnancement est FIFO.

ment équitablement.

Le modèle de trafic considéré prévoit un nombre de flots en compétition réduit à quelques unités, le cas d'un seul flot étant à la fois le plus probable et le pire cas. A la lumière de ces résultats, une taille de buffer réduite à 20 paquets n'est pas justifiée, et des tailles plus importantes – comme suggérées par le scénario B=100 – doivent être considérées.

La suite de cette section approfondit l'étude des phénomènes causés par des petits buffers, en étendant notamment le modèle de trafic à des cas plus réalistes. Nous tentons d'une part de comprendre l'impact de buffers réduits (qui peuvent être une contrainte technique, par exemple pour des buffers optiques), et d'autre part de proposer un dimensionnement des buffers permettant de préserver la performance des flots élastiques. Une taille plus importante permettra d'absorber les fluctuations causées par le processus aléatoire d'arrivée des paquets *background*, permettant ainsi une évolution satisfaisante de la fenêtre de congestion des flots TCP.

4.6.2 Flots *bottlenecked* au débit crête non limité

Méthode

Afin d'évaluer le débit des flots, nous procédons comme suit. Pour une capacité de lien, une taille de buffer et une charge *background* donnée, nous simulons successivement un nombre de flots TCP *bottlenecked* permanents. Pour chaque nombre i (entre 1 et 500, ce qui permet une estimation suffisamment précise), nous évaluons le débit global réalisé $\phi(i)$ exprimé en tant que fraction de la capacité résiduelle $C(1-\rho_b)$. Nous dérivons alors l'espérance du débit des flots γ par la formule 3.2 présentée dans le Chapitre 3, Section 3.3.2. Cela correspond à une analyse quasi-stationnaire qui nous permet d'ignorer des phénomènes tels que limitations de débit dues au *slow start* et les iniquités temporaires entre les flots. Il s'agit de trouver la probabilité stationnaire du système représenté en figure 4.8, et de calculer l'espérance du nombre de flots.

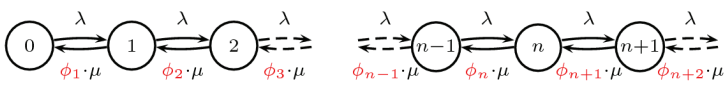


FIGURE 4.8 – File d'attente représentant le nombre de flots en cours, pour un régime élastique régi par TCP

Les figures 4.9, 4.10, 4.11 représentent les valeurs de $\phi(i)$ et γ en fonction de la charge du lien ρ

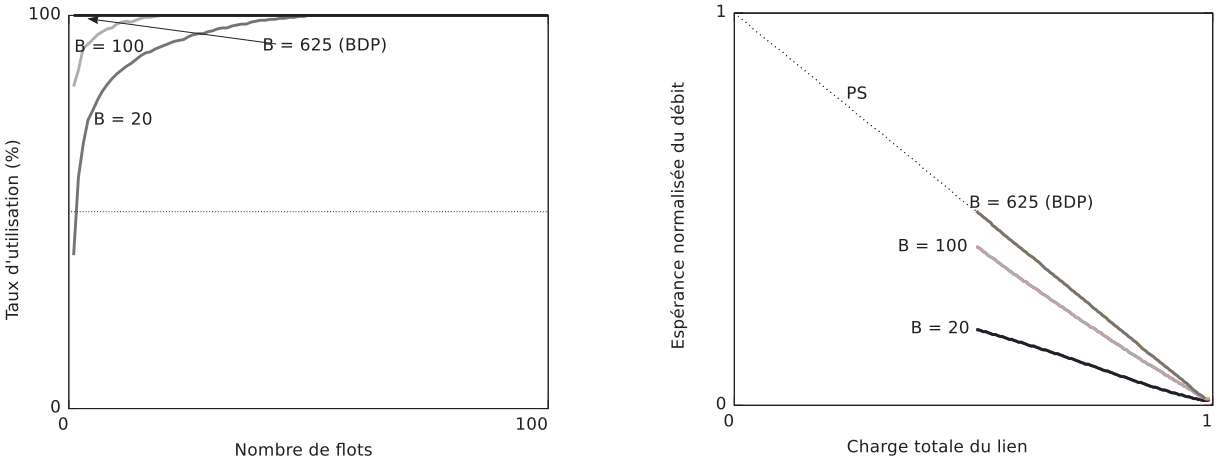


FIGURE 4.9 – Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ, **pour différentes tailles de buffer**

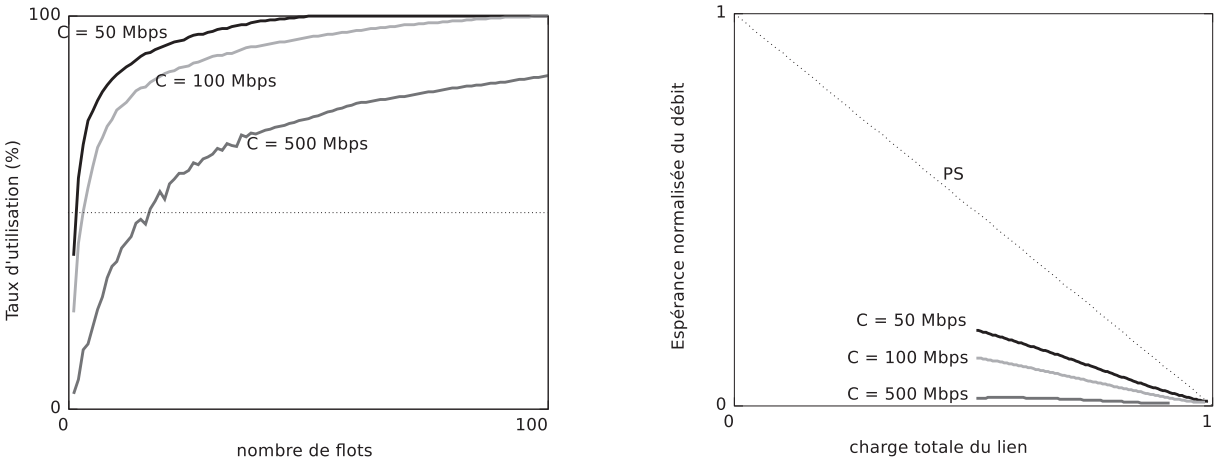


FIGURE 4.10 – Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ, **pour différentes capacités de lien**

pour un ensemble de configurations. Il convient de noter que γ est seulement défini pour des charges supérieures à la charge *background* ρ_b et que sa valeur pour cette charge est déterminée par $\phi(1)$, débit résiduel utilisé par un seul flot *bottlenecked*.

Exprimons le débit moyen γ d'un flot dans le cas d'un partage équitable [26] :

$$\gamma = \frac{\rho}{E[X]}$$

où $E[X]$ représente l'espérance du nombre de flots en cours. Nous cherchons à déterminer la valeur de γ lorsque la charge tend vers 0. Numérateur et dénominateur tendant vers 0, nous pouvons appliquer le théorème de l'Hôpital :

$$E[X](\rho) = \pi_0(\rho) \cdot \sum_{i=1}^{\infty} \frac{i\rho^i}{\prod_{j=1}^{i} \phi_j}$$

$$\frac{dE[X](\rho)}{d\rho} = \pi_0 \cdot \sum_{n=1}^{\infty} \frac{i^2 \cdot \rho^{i-1}}{\prod_{j=1}^{i} \phi_j}$$

qui tend vers $\frac{1}{\phi_1}$ en 0 ($\pi_0(0) = 1$). γ tend donc vers ϕ_1 pour une charge nulle. C'est en effet le débit qu'aurait un flot arrivant dans un système vide.

Nous avons expliqué plus tôt les raisons causant une dégradation de la valeur de $\phi(1)$, qui détermine également la forme de la courbe (elle est approximativement linéaire dans le cas d'un partage équitable entre les flots). Elle décroît de sa valeur maximale atteinte en $\rho = \rho_b$ vers 0 pour $\rho = 1$.

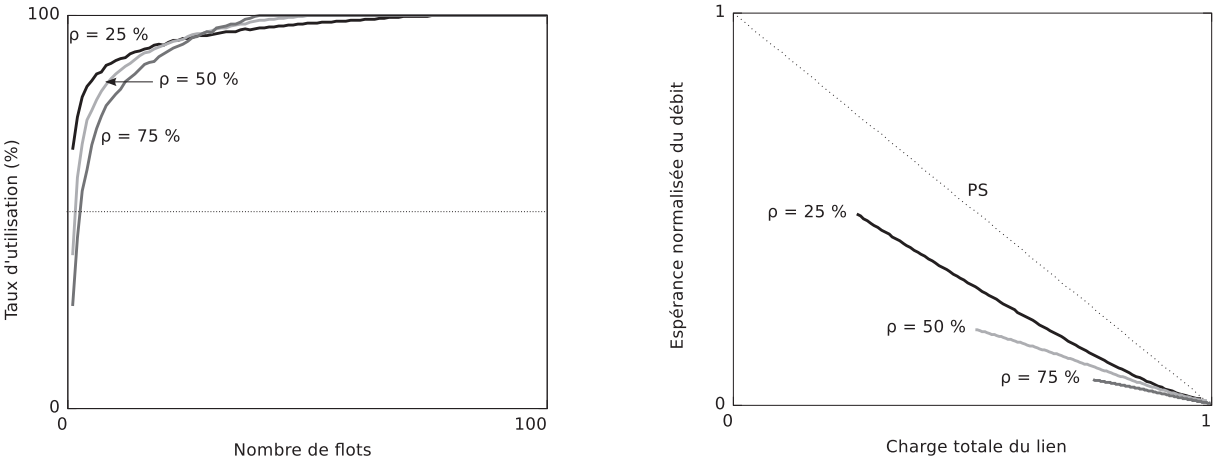


FIGURE 4.11 – Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ, **pour différentes charges de trafic *background***

Les résultats montrent qu'il y a une chute significative du débit avec de petits buffers (fig. 4.9) et que cette perte est encore plus marquée quand la capacité du lien augmente (fig. 4.10). Plus la charge *background* est importante, plus il est difficile pour TCP d'utiliser la bande passante résiduelle (fig. 4.11).

Débit moyen d'un flot *non-bottlenecked* en fonction de la taille du buffer

Nous avons vu que la connaissance de $\phi(1)$ détermine la performance du système pour des conditions données de capacité de lien, de taille du buffer et de charge (charge *background* + charge due aux flots TCP). Si peu d'articles évaluent la performance d'une connexion TCP dans les conditions que nous avons présentées, [80] propose toutefois un modèle analytique de TCP Reno dans un tel contexte, lorsque le trafic *background* suit un processus D-BMAP (*Discrete Batch Markov Arrival Process*). La complexité de ce modèle ne permet pas d'estimer facilement l'impact de la taille du buffer sur la performance de TCP. De plus, nous ne possédons pas de tels résultats pour les autres protocoles que nous étudierons dans le chapitre suivant ; c'est pourquoi nous nous contentons de simulations dans le reste de cette section.

La figure 4.12 représente le produit $\phi(1)C(1-\rho_b)\texttt{RTT}$ comme une fonction de la taille du buffer. Cette grandeur représente le taux d'utilisation de la bande passante résiduelle renormalisé par le BDP de cette même capacité résiduelle. Trois configurations sont représentées ; la capacité du lien et le RTT varient (C=50Mb/s et RTT=100ms ; C=100Mb/s et RTT=50ms ; C=100Mb/s et RTT=100ms) ; la charge du trafic *background* reste la même (50%). Dans les deux premiers cas, la valeur du BDP est identique (2.5Mb) ; dans le troisième elle est le double (5Mb). On constate que la courbe d'utilisation croît d'abord fortement avec l'augmentation de la taille du buffer, puis subit un point d'inflexion pour converger vers une utilisation complète de la capacité résiduelle (une ligne horizontale sur la figure). Dans cette représentation, on note que les flots qui subissent le même BDP ont le même comportement.

Si l'on représente des configurations similaires en faisant varier la charge *background*, les portions du graphe correspondant aux petites tailles de buffers se superposent à charge fixée. Dans un souci de clarté, elles ne sont pas représentées sur la figure ; seule l'est la tendance qu'elles évoquent (en pointillés). Ainsi, pour de faibles tailles de buffers, cela suggère une dépendance à la charge *background* uniquement.

Nous pouvons ainsi distinguer deux zones, caractérisées par la dépendance de la performance à la taille du buffer :

- Si le *Bandwidth Delay Product* résiduel est suffisamment grand et que le buffer est petit, alors le processus décrivant la valeur de *cwnd* lorsque la perte se produit ne dépend que de ρ_b. Cette charge détermine la taille moyenne de la fenêtre de congestion, et ainsi le débit du flot.
- Pour une taille de buffer plus importante, *cwnd* peut croître jusqu'à éventuellement atteindre une valeur suffisante pour une utilisation complète de la bande passante disponible.

Vers une proposition de dimensionnement...

L'allure de $\phi(1)$ en fonction de B suggère que le buffer devrait être dimensionné afin d'éviter au moins la forte dégradation initiale du débit, due à l'interaction de TCP avec le trafic *background*

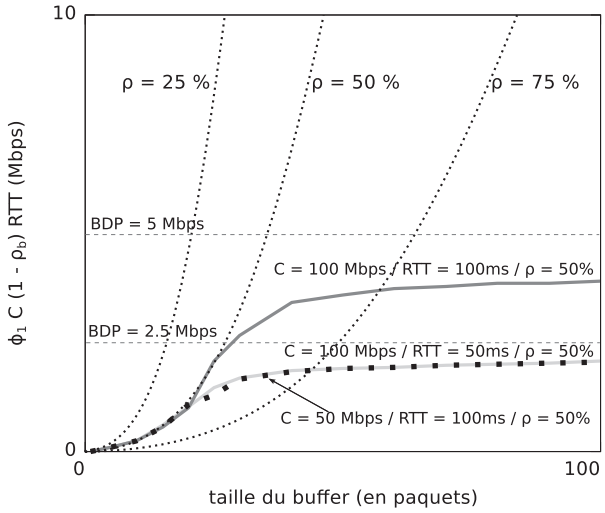


FIGURE 4.12 – Utilisation de la bande passante disponible faite par 1 flot TCP *bottlenecked*, pour différentes valeurs de la capacité C, du RTT des flots, et de la taille du buffer B (représentée en abscisse). Les courbes de *Bandwidth Delay Product* (BDP) équivalents se retrouvent superposées. Elles sont comparées à l'utilisation atteinte pour un dimensionnement suffisamment important, selon la règle empirique du BDP (lignes horizontales en pointillés). Enfin, il n'est pas possible de représenter toutes les combinaisons explorées, mais les tendances des courbes pour de petites tailles de buffer sont représentées en pointillés pour différents niveaux de charge (ρ), et suggèrent un dimensionnement en la racine carrée du BDP.

concurrent. Il est pas nécessaire cependant d'atteindre une utilisation de 100%, et une taille bien plus faible que le *Bandwidth Delay Product* semble suffire. Une possibilité est de choisir une valeur de B à la jonction des deux zones définies précédemment.

L'inspection de la courbe caractéristique des petites tailles de buffer suggère une dépendance de l'ordre de B^2. En d'autres termes, la taille requise pour B serait alors grossièrement proportionnelle à la racine carrée de la bande passante résiduelle. Cette constatation mérité d'être approfondie, mais il s'agit néanmoins d'une indication précieuse pour le dimensionnement.

Nous comprenons également pourquoi une taille de buffer intermédiaire, que nous avons évoqué plus tôt, est satisfaisante. Elle représente un bon compromis pour la performance des flots, tant que la charge *background* n'est pas trop importante (et ca sera notamment le cas sur des liens opérationnels qui sont généralement peu chargés), même pour de grandes valeurs du BDP. Par exemple, l'allure de la courbe pour une charge de 50% suggère que 100 paquets seront suffisants même si le BDP devient très élevé.

4.6.3 Trafic élastique avec des flots à débit crête limité

Notre hypothèse de flots *bottlenecked* de débit crête illimité peut être remise en cause sachant que le débit des liens d'accès n'est souvent qu'une fraction de celui des liens du cœur de réseau. Il s'ensuit un espacement naturel des paquets qui nous pousse à considérer l'impact de flots *bottlenecked* mais de débits limités [4]. Ce cas fait notamment la transition avec le régime transparent, puisque le lien ne devient saturé que lorsque plusieurs flots se combinent entre eux, ce qui se produit à forte charge. Il s'agit des cas que nous avons exclu précédemment, le débit crête définissant la charge au delà de laquelle le lien entre en régime élastique.

Afin d'illustrer l'impact de flots à débit crête limité p, nous supposons que de tels flots partagent un lien avec un trafic *background* Poisson, comme précédemment. La figure 4.13 représente le débit $\phi(i)$, en fonction du nombre de flots *bottlenecked* i, et γ/C, en fonction de la charge du lien pour différentes configurations.

Les résultats montrent que $\phi(i)$ croît linéairement tant que le débit global des flots *bottlenecked* reste relativement inférieur à la capacité résiduelle, vu que chaque flot réalise son débit crête et que le lien opère en régime transparent. Quand la charge agrégée atteint toutefois un niveau où les flots *bottlenecked* commencent à perdre des paquets, l'inefficacité des petits buffers est à nouveau visible.

4. En fait ces flots ne sont *bottlenecked* que si leur nombre est assez grand.



FIGURE 4.13 – Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ, pour des flots *non-bottlenecked* de débit limité à 2Mb/s

Par exemple pour $p = 2$ Mb/s dans la figure 4.13, le débit $\phi(i)$ chute lorsqu'il y a plus de 11 flots, et n'augmente à nouveau vers 100% de la capacité que lorsque ce nombre devient bien plus important.

Cependant la perte d'efficacité avec de petits buffers est dans ce cas moins significative au niveau du débit des flots comme le montre le comportement de γ en fonction de la charge du lien. La perte de débit est seulement visible pour de fortes charges où elle accentue la dégradation qui se produit dans le modèle PS idéal (vu ici lorsque B=625 paquets). Plus le débit crête des flots sera faible, plus le lien opèrera en régime transparent jusqu'à de fortes charges.

4.7 Conclusions

La relation entre la taille du buffer et la performance réalisée dépend clairement des hypothèses sur les caractéristiques du trafic. La plus significative est le mélange des débits crête exogènes des flots, c'est-à-dire les débits qu'ils atteindraient si le lien considéré était de capacité infinie. La charge du lien (taux d'arrivée des flots $\times$ taille moyenne des flots / capacité du lien) détermine alors quels flots, s'il y en a, parmi ceux de plus forts débits sont *bottlenecked*, les autres représentant pour eux une charge *background*. Nous distinguons trois principaux régimes de partage statistique de bande passante :

- lorsque tous les débits crête sont relativement peu élevés et que la charge n'est pas trop proche de 1, la somme des débits crête reste inférieure à la capacité du lien avec une forte probabilité ; nous nommons cela régime transparent ; un simple modèle de files d'attente M/M/1 peut être utilisé pour évaluer la relation entre la taille du buffer et la probabilité de perte des paquets ; un petit buffer est alors approprié ; par exemple, un buffer de 20 paquets déborde avec une probabilité de 0.01 à une charge proche de 80% ;
- lorsque quelques flots peuvent individuellement saturer la bande passante résiduelle non utilisée par la charge *background* (flots de faible débit crête), le partage de bande passante est réalisé par le contrôle de congestion de bout en bout (TCP) ; nous appelons cela le régime élastique ; avec la pratique actuelle des protocoles d'envoyer les paquets aussitôt un acquittement reçu, un petit buffer tend à déborder trop rapidement pour permettre à la fenêtre de congestion de TCP de croître complètement, ce qui peut mener à une utilisation très faible des ressources ; la taille des buffers nécessaire dans ce régime augmente avec le *Bandwidth Delay Product* résiduel ; une analyse empirique suggère que la taille du buffer soit proportionnelle à la racine carrée du *Bandwidth Delay Product* résiduel ;
- lorsque les flots de plus forts débits crête doivent se combiner (nous entendons par là plusieurs flots en parallèle) pour saturer la bande passante résiduelle, nous avons un régime intermédiaire plus général transparent/élastique ; lorsque le débit crête des flots (*potentiellement*) *bottlenecked* est une fraction relativement faible de la bande passante résiduelle (par exemple 1/10), et que la charge globale n'est pas trop proche de 1, le lien est rarement saturé et un petit buffer dimensionné comme pour le régime transparent demeure adéquat.

Il semble ainsi possible d'envisager un cœur de réseau entièrement optique, impliquant de petits buffers, si l'on peut continuer à assurer la transparence du lien par une disparité entre les débits d'accès et la capacité du lien de cœur de réseau [50]. Toutefois, dès lors qu'il sera possible pour des flots d'avoir un débit crête non négligeable par rapport à la capacité du lien, la performance se verra dégradée (plus ou moins en fonction encore du débit de ces flots). Le pire cas correspond à un flot pouvant saturer à lui seul la capacité résiduelle.

Toutefois cette étude s'intéresse uniquement à la performance de l'agrégat de flots, et ne prends pas en compte leur performance individuelle, notamment au niveau du partage réalisé. Elle ne tient pas non plus compte des évolutions possibles et probables des protocoles de transport, ni des mécanismes de qualité de service qu'il est possible de mettre en œuvre afin peut-être d'obtenir une meilleure performance. C'est ce que nous nous proposons de considérer dans le prochain chapitre.

Chapitre 5

Partage de bande passante étendu : nouveaux protocoles TCP & *fair queueing*

Résumé :

Dans ce chapitre, nous revisitons la relation fondamentale qu'il existe entre les ressources du lien, la demande en trafic et la performance obtenue. Nous considérons l'introduction de nouveaux protocoles TCP plus efficaces, conjointement à l'utilisation d'un ordonnancement *fair queueing* (PFQ, muni de la politique LQD).

PFQ/LQD permet un partage équitable de la capacité du lien (C) et du buffer (B), l'introduction de nouveaux mécanismes TCP plus efficaces (et moins dépendant des ressources disponibles), ainsi que la différenciation implicite des flots *streaming* et élastiques. La combinaison PFQ/LQD et MBAC permet de garantir leur performance.

Sommaire

5.1 Introduction **62**
5.2 État de l'art : Nouveaux protocoles TCP et évaluation de leur performance **63**
5.3 Motivations pour l'intégration de fair queueing **65**
5.4 Comportement de TCP dans un environnement équitable **74**
5.5 Discussion **84**
5.6 Conclusion **85**

Contributions :

- Illustration de la dégradation de performance des flots streaming et élastique sans ordonnancement, et des apports du *fair queueing* ;
- Évaluation des versions les plus courantes de TCP à la lumière de notre compréhension du trafic, dans un environnement équitable ;
- Propositions pour la conception d'algorithmes TCP plus efficaces.

Publications et présentations :

Ce chapitre a fait l'objet des présentations suivantes en *workshop* :

- James Roberts, Jordan Augé, **Fair Queueing and Congestion Control**, *Workshop on Congestion Control, Hamilton Institute, September 2005.*
- Jordan Augé, James Roberts, **Performance of TCP over a Fair Queueing Link**, *Proceedings of Workshop on QoS and Traffic Control, EuroNGI, Paris, December 2005.*

Il intègre également du contenu de la publication citée dans le chapitre précédent :

- Jordan Augé, James Roberts, **Buffer Sizing for Elastic Traffic**, *NGI2006, 2nd Conference on Next Generation Internet Design and Engineering, València, April 3-5 2006*

5.1 Introduction

Les modèles de trafic tels que ceux présentés dans le Chapitre 3 se basent sur une version fluide, idéalisée du trafic, où les mécanismes au niveau paquet n'interviennent pas. En pratique, TCP ne réalise pas un partage équitable, instantané, et efficace des ressources. Comme il a été évoqué plus tôt, les versions classiques de TCP – Reno/NewReno – sont reconnues comme insuffisantes pour exploiter convenablement la bande passante disponible sur des liens à fort "Bandwidth-Delay product", notamment au vu de la lenteur de l'algorithme AIMD. Nous avons vu de plus que la présence de petits buffers ne permet pas à l'algorithme une utilisation efficace des ressources. Les interactions au niveau paquet au sein d'un petit buffer, exacerbées par l'envoi en rafales causé par le contrôle de congestion de bout en bout, entraînent des pertes de paquets "naturelles", non liées à de la congestion, qui obligent TCP à adapter son débit bien en dessous du débit équitable. Dans des conditions réalistes de trafic, il est ainsi difficile de réduire drastiquement la taille des buffers dès lors que l'on souhaite assurer la performance des flots TCP Reno, sur un lien FIFO.

Les évolutions que connaissent les réseaux aujourd'hui, avec l'arrivée des réseaux optiques à très haut débit, ou le fort intérêt suscité par les *datacenters*, sont autant de facteurs motivant l'introduction de nouveaux protocoles ou mécanismes de gestion du trafic plus adaptés. De nombreux travaux montrent aujourd'hui l'intérêt de reconsidérer l'introduction d'ordonnancement (*scheduling*) ou de politiques de gestion des files d'attente (*AQM*), afin d'améliorer la performance des flots (FQ-Codel [70], pFabric [5], etc.).

Une alternative radicale aux solutions FIFO ou avec AQM est d'introduire un ordonnancement équitable, ou *fair queueing* (FQ), entre les flots. FQ permet un découplage entre le contrôle de débit et l'ordonnancement des flots. Nous utilisons l'implémentation PFQ (que nous avons réalisé dans ns-2), avec une politique rejetant les paquets de la file la plus longue (LQD, *Longuest Queue Drop*). Suter *et al.* montrent qu'un tel mécanisme est largement plus efficace que RED pour la gestion du trafic. Nous verrons qu'il permet aux flots *non-bottlenecked* de ne subir aucune perte. Leurs paquets subissent notamment de faibles délais [1], ce qui est une propriété désirable pour les flots *streaming* qui y sont inclus.

Un autre avantage bien connu du *fair queueing* est la possibilité d'introduire et d'expérimenter de nouvelles versions de TCP sans nuire aux utilisateurs de la version standard, puisque l'équité dans le réseau est assurée indépendamment du comportement de l'utilisateur. Le *fair queueing* peut être considéré comme infaisable si le nombre de flots à contrôler est trop important, et s'il croît avec la capacité du lien. En réalité, le nombre de flots à ordonnancer à tout instant est relativement faible, indépendamment de la capacité du lien (voir Chapitre 3, et [95, 93]).

1. Les délais sont d'autant plus réduits avec l'utilisation d'une file prioritaire décrite dans [91, 92].

L'impact sur la performance ne sera bien entendu perceptible que lorsque le lien sera saturé, ce que nous avons qualifié précédemment de régime élastique. Ainsi dans ce chapitre, nous explorons sous une nouvelle perspective la relation introduite dans le Chapitre 2, entre ressources, demande et performance. Nous réexaminons les arguments relatifs à l'introduction de tels protocoles au vu de notre compréhension du trafic, et en considérant l'utilisation complémentaire d'un mécanisme de *fair queueing*.

Nous commençons dans la section 5.2.2 par présenter un ensemble d'évolutions proposées pour TCP, pour ensuite motiver l'introduction dans la section 5.3 de *fair queueing* (FQ), qui assure la performance des flots *streaming* et élastiques, et permet l'introduction de nouveaux protocoles TCP plus efficaces. En considérant désormais un environnement équitable, nous analysons dans la section 5.4 un ensemble représentatif de versions de TCP au vu de notre compréhension du trafic, afin de vérifier dans quelle mesure les résultats établis dans le chapitre précédent sont confirmés (impact de petits buffers en présence de charge *background*). Finalement, la section 5.5 discute les résultats obtenus et un ensemble de pistes. Malgré une performance supérieure, nous constatons que les versions existantes de TCP peuvent encore être améliorées afin de mieux accepter les pertes aléatoires causées par les petits buffers et mieux exploiter la bande passante disponible. Il est possible d'exploiter les bénéfices apportées par le *fair queueing* afin de concevoir de nouveaux protocoles, plus agressifs, tout en gardant un taux de perte de paquets limités. Un complément prometteur, bien que de mise en œuvre complexe, semble être l'utilisation de techniques d'espacement des paquets (*pacing*),

5.2 État de l'art : Nouveaux protocoles TCP et évaluation de leur performance

5.2.1 Faiblesses de TCP Reno pour les transferts à haut débit

TCP a été introduit à une époque où les débits des liens étaient encore relativement faibles. Ce protocole a dû s'adapter aux différentes évolutions que le réseau a connu, et assurer un transfert fiable même sur les capacités actuelles des réseaux. Cependant, on lui reconnaît aujourd'hui certaines faiblesses sur les liens à forts délais et/ou à haut débit.

Le premier problème vient notamment de l'algorithme AIMD, qui régit l'évolution de la taille de la fenêtre de congestion w ainsi :

$$w \leftarrow w + \frac{1}{w}$$

à chaque acquittement reçu en mode congestion avoidance, et

$$w \leftarrow \frac{w}{2}$$

en cas de perte de paquet.

Lors d'une congestion ponctuelle, l'algorithme AIMD entraîne une réduction relativement brutale (*multiplicative decrease*) de la taille de la fenêtre de congestion (elle est divisée par deux), et celle-ci ne retrouvera sa taille d'origine que longtemps après à cause de la faible croissance (*additive increase*). La réalisation de très hauts débits avec TCP Reno nécessite un taux de perte infime, qui peut être difficile en pratique. Ainsi, les pertes subies par un flot TCP entraîneront un débit moyen de transfert bien en deçà de la limite équitable permise par le lien. Cette sensibilité peut être diminuée en ne considérant plus seulement une notion binaire telle qu'une perte de paquets, mais en y adjoignant d'autres estimateurs (TCP Vegas utilise par exemple une estimation du nombre de paquets attente dans le buffer). L'évolution de la fenêtre de congestion est modulée par l'état de congestion du réseau.

5.2.2 Nouveaux protocoles TCP

	Protocole	Caractéristiques
Loss	Reno	Incrément additif, décrément multiplicatif de *cwnd* (AIMD)
	HSTCP [52]	AIMD classique pour de faibles valeur de *cwnd*, version plus agressive au delà
	Scalable [85]	Incréments et décréments multiplicatifs (MIMD)
	Westwood+	Incrément additif, et décrément adaptatifs en fonction d'une estimation de bande passante basées sur le flux d'acquittements avant une perte
	BIC	Une combinaison de fonctions logarithmiques et exponentielles pour la croissance de *cwnd*
	CUBIC	Extension de BIC avec une fonction cubique
Delay	H-TCP	AIMD adaptatif
	Vegas	Incrément et décrément additifs (AIAD)
	Fast [78]	Incrément/décrément basés sur une équation faisant intervenir le délai principalement, ainsi que le signal de perte
Hybrid	Compound	Somme de deux valeurs de *cwnd*, de type Reno et Vegas
	Illinois	AIMD dont les paramètres d'incrément/décrement sont fonction du délai
	Veno	Combinaison Reno/Vegas afin de détecter les pertes non dues à la congestion
	Yeah	Alternance entre un protocole agressif (eg. Scalable TCP) et un contrôle de congestion à la Reno, en fonction d'une estimation de délai.

TABLE 5.1 – Résumé des versions de TCP considérées dans ce chapitre

Il existe un très grand nombre de protocoles TCP. Il ne s'agit pas ici d'en fait un inventaire exhaustif, mais d'en sélectionner un petit nombre représentatif, et couvrant les différentes approches. Nous avons retenu les protocoles présents dans le noyau Linux, disponibles également dans le simulateur NS avec le package TCP Linux. Ils sont résumés dans le tableau 5.1, en fonction de leur signal de congestion principal (pertes, délais ou mixte), et accompagnés d'un bref descriptif de leurs différences.

Nous n'avons pas considéré dans ce chapitre des protocoles nécessitant un marquage ou une signalisation explicite des débits, comme XCP [81], qui est difficilement réalisable dans l'interdomaine et nécessite des modifications importantes au sein des routeurs, ou encore DCTCP [4], qui est proposé dans un contexte de *datacenter* uniquement, et qui n'était pas disponible au moment de ces travaux.

Performance

Il n'est pas surprenant que l'évaluation de la performance des différents algorithmes TCP ait reçu un intérêt considérable de la communauté, et l'on recense un nombre incalculable de publications réalisant des évaluations analytiques, par simulation ou expérimentales de la performance d'une ou plusieurs propositions d'algorithme TCP.

Une des premières études de l'impact de la taille du buffer sur la performance de HSTCP est par exemple proposée par Barman *et al.* [12]. Le débit réalisé atteint respectivement 90% et 98% de la capacité du lien pour des tailles de buffer égales à 10% et 20% du *Bandwith Delay Product*. Pour une trop petite valeur du buffer toutefois, le débit observé est bas. Typiquement, le fonctionnement des protocoles est dégradé en présence de petits buffers, ce qui est consistent avec nos observations. On pourra se référer à [100] pour une évaluation expérimentale systématique d'un grand nombre de variantes TCP. Les métriques généralement considérées sont les suivantes :

équité : plusieurs notions et métriques d'équité ont été définies dans la littérature. L'étude est complexe dès lors que l'on considère plusieurs protocoles car il convient de considérer l'équité intraprotocolaire (entre protocoles similaires), ainsi que l'équité interprotocolaire (en fonction des différentes combinaisons possibles). Le comportement d'un réseau intégrant de nombreuses versions hétérogènes de TCP est peu maitrisé, et à de fortes chances de le rester.

En pratique, les systèmes d'exploitation Linux et Windows offrent déjà des versions différentes, respectivement CUBIC et Compound. Nous ne nous intéresserons pas à cette métrique, puisque nous confierons l'allocation de ressources à l'ordonnanceur FQ. Le rôle de TCP sera alors de maximiser son efficacité et l'utilisation de la bande passante disponible. A notre connaissance, aucune évaluation des protocoles n'a été effectuée dans un tel environnement.

vitesse de convergence : elle représente le temps mis par le protocole à s'adapter aux arrivées et départs de flots pour rejoindre le débit équitable. Lors d'un départ de flot, chaque flot recevra équitablement une partie du débit correspondant qui, s'il n'ajuste pas son débit suffisamment rapidement, sera soit perdue, soit utilisée par un autre flot. Lors de l'arrivée d'un nouveau flot, et lorsqu'il atteindra son débit équitable, cette lenteur se manifestera par des pertes de paquets. En présence de *fair queueing*, une convergence plus lente entraînera soit des pertes de paquets, soit un débit temporairement inférieur à celui alloué.

efficacité : il s'agit du débit moyen atteint par les flots. C'est la métrique que nous allons plus particulièrement développer dans ce chapitre, puisqu'elle reflète les critères de performance que nous considérons pour les flots élastiques (un débit moyen suffisant à garantir un temps de complétion satisfaisant).

débit utile : le débit utile est lié à la notion précédente et correspond au ratio entre le débit d'envoi, et le débit effectif après pertes. La présence de *fair queueing* permettrait en théorie à une source d'émettre au débit maximum, mais cela causerait un gaspillage inutile des ressources lorsque les paquets sont rejetés en masse à l'entrée d'un lien saturé. L'utilisation de codes correcteurs d'erreur comme alternative au mécanisme d'acquittement serait une autre motivation pour conserver un taux de pertes acceptable.

stabilité des buffers : à ne pas confondre avec la stabilité au niveau flot (demande < capacité), ou au niveau paquet (avec un buffer fini, le lien sera stabilisé par les pertes de paquets). Il s'agit de minimiser les oscillations dans le buffer, responsables de gigue pour les paquets, et qui sont liées avec la synchronisation des flots TCP. Généralement, une diminution de la taille des buffers améliorera cette métrique (Voir les travaux de Raina et Wishik [129] introduits dans le Chapitre 4 Section 4.3.2).

Peu d'études considèrent un modèle de trafic avec des arrivées dynamiques de flots finis. La plupart considèrent des flots permanents, et en très grand nombre, ce qui ne correspond pas à un taux de charge réaliste pour un lien opérationnel. Nous avons remarqué dans le Chapitre 4 que la présence de trafic concurrent pouvait fortement dégrader la performance d'un flot TCP. Un phénomène similaire à été observé plus tard dans [150, 151] et [148], mais n'a pas donné lieu à une évaluation des protocoles dans un tel contexte. Une évaluation intéressante de la performance d'un flot TCP en présence d'une charge *background* est faite dans [80], mais le modèle est complexe et ne peut être réutilisé pour notre étude (spécifique à Reno).

5.3 Motivations pour l'intégration de fair queueing

Dans cette section, nous effectuons quelques simulations permettant d'illustrer les bénéfices apportés par l'utilisation d'un ordonnancement *fair queueing*. Ce même environnement de simulation sera utilisé par la suite pour l'évaluation des différents protocoles TCP.

5.3.1 Environnement de simulation

Topologie et modèle de trafic

Nous utilisons un environnement de simulation similaire à celui du chapitre précédent, permettant de réaliser un simple lien goulotté au travers d'une topologie *dumbbell*. Le lien central possède une capacité C (de 50, 100 ou 500Mb/s), et le RTT des flots y est de 100ms. Les simulations mettent en jeu des tailles de buffers B variées, notamment inspirées du chapitre précédent : $B = 20$, 100 ou 625 paquets (BDP). Deux disciplines de services sont considérées : FIFO/DropTail et PFQ/LQD.

Notre modèle de trafic repose sur une hypothèse de quasi-stationnarité qui nous permet de considérer deux ensembles de flots : un trafic *background* de charge ρ_b qui est constitué de l'agrégat des flots *non-bottlenecked* (la majorité des flots), ainsi qu'un ensemble de N flots *bottlenecked*, que l'on peut alors assimiler à des flots permanents.

Flots *bottlenecked* (contraints localement)

Comme nous l'avons vu dans le Chapitre 3, leur nombre est généralement faible avec $Pr[N \geq i] \sim \rho^i$, où ρ désigne la charge relative des flots goulottés. Pour illustrer la performance des flots *bottlenecked* et *non-bottlenecked*, nous limiterons nos simulations aux quelques cas représentatifs tels que $N \in [1,2,4]$. Considérons par exemple un lien avec une charge *background* $\rho_b = E[flowarrivalrate] \times E[flowsize]/C = 50\%$, pour une charge globale de 0.6, on a $Pr[N \geq 5] \approx \rho^5 = 0.2^5 \approx 0.0003$.

Flots *non-bottlenecked* (non-contraints localement)

En régime transparent, nous avons vu dans le chapitre précédent qu'il est possible de garantir la performance des flots (*non-bottlenecked*) avec un dimensionnement correct du buffer. Des petites tailles suffisent tant que la charge n'est pas trop proche de 100%.

Ici, nous considérons le cas réaliste où de tels flots sont acheminés en présence de flots élastiques à plus fort débit, utilisant éventuellement des versions plus agressives de TCP, et saturant le lien (nous

avions nommé ce régime "transparent-élastique"). Ces flots *non-bottlenecked* ne réalisent pas leur débit équitable (*slow start*, débit d'accès limité ou congestion sur un autre lien). Ils ne participent pas activement à l'allocation des ressources réalisée par TCP, et sont éventuellement victimes des connexions à plus fort débit.

Les simulations présentées dans cette section correspondent à une charge *background* $\rho_b = 50\%$, générée par des arrivées Poissonniennes de flots TCP de taille exponentielle de moyenne 200ko et de débit crête 1Mb/s.

5.3.2 Performance des flots *non-bottlenecked*

Impact d'un flot TCP sur les flots *non-bottlenecked*

Dans les figures 5.1 et 5.2, nous représentons les résultats de simulation obtenus avec un flot *bottlenecked*, respectivement TCP Reno et HSTCP, pour B=20, 100 et 625 paquets (de gauche à droite), sur un lien FIFO. De haut en bas, nous avons : l'évolution de *cwnd* pour le flot TCP, l'évolution de la file d'attente, le taux de pertes de paquets pour chaque classe, et une représentation des pertes de chaque flot (le rectangle sur l'axe des abscisses représente les dates de début et de fin du flot, la position verticale est aléatoire, la hauteur représente le débit du flot, l'aire la taille transmise, et enfin le dégradé de couleur le taux de pertes subi (gris : pas de perte, rouge : faible taux de perte, noir : taux élevé).

L'évolution de *cwnd* dans les différentes configurations est caractéristique de l'algorithme AIMD au cœur du contrôle de congestion réalisé par TCP. Ce dernier est responsable de trois phénomènes qui affectent la performance réalisée par les flots *non-bottlenecked* :

- des **pertes de paquets**. Les figures 5.1 et 5.2 montrent les pertes pour chaque classe de flots, causées par la saturation du buffer lorsque la fenêtre de congestion devient trop importante. Ces instants de saturations sont d'autant plus fréquents que le protocole est agressif.
- des **délais** : un paquet sera retardé d'autant plus que la taille de la file d'attente sera importante à son arrivée (FIFO). Un buffer de 625 paquets (le BDP) représente par exemple une latence d'environ 100ms sur un lien à 50Mb/s (en fait, 1 RTT maximum tel que considéré pour le dimensionnement).
- de la **gigue** : elle est causée par les oscillations de la file d'attente.

Ces phénomènes sont aggravés lors des périodes de *slow-start* de TCP (croissance exponentielle de *cwnd*), ou avec l'utilisation de protocoles plus agressifs comme HSTCP.

Les taux de pertes subis par le flot permanent (*bottlenecked*) et par les flots dynamiques (*non-bottlenecked*) sont représentés respectivement en trait rouge plein, et en bleu pointillés. Ce taux peut être relativement élevé, de 10 à 30% pour TCP Reno en mode *congestion avoidance*, jusqu'à plus de 70% lors des périodes de *slow start*. Les figures montrent la répartition de ces pertes : même si elles sont distribuées sur un grand nombre de flots, elles affecteront néanmoins plus ou moins fortement leur performance en fonction des codecs utilisés (dans la figure du bas, la couleur rouge symbolise un faible taux de pertes, et noir un fort taux de pertes).

Les flots *non-bottlenecked* contiennent en partie des flots élastiques, pour lequels la considération du taux de pertes seul n'est pas suffisamment représentatif de la performance. Il s'agit de flots n'atteignant déjà pas le débit équitable, potentiellement interrompus pendant l'établissement de la connexion, la période de *slow-start*, etc. La conséquence pour ces flots sera un allongement de leur temps de transfert. Nous représentons en figure 5.3 la distribution du nombre de flots en cours, qui permet d'illustrer la performance réalisée. La durée moyenne d'un flot τ est associée au nombre moyen de flots N et à leur taux d'arrivée λ par la loi de Little : $\tau = N/\lambda$. L'allongement du temps de transfert d'un flot est reflété par le plus grand nombre de flots en cours. Les figures semblent indiquer que la performance est principalement dégradée par le délai subi par les flots (buffer), et par les pertes dans une moindre mesure (le protocole HSTCP étant plus agressif).

Petites tailles de buffers

L'impact du protocole semble atténué par la présence d'un petit buffer, certainement parce qu'il limite l'évolution de *cwnd*. Cela va dans le sens des propositions d'un réseau *Best Effort* avec de petits buffers, où le contrôle est effectué en bordure. On voit toutefois que des flots de fort débit crête peuvent dégrader la performance, notamment lors du *slow start*, et d'autant plus fortement qu'ils seront agressifs.

Tailles de buffers intermédiaires

Nous avons recommandé une taille de buffer B de l'ordre de 100 paquets comme un compromis raisonnable pour assurer la performance des flots élastiques (*bottlenecked* et *non-bottlenecked*). La perfor-

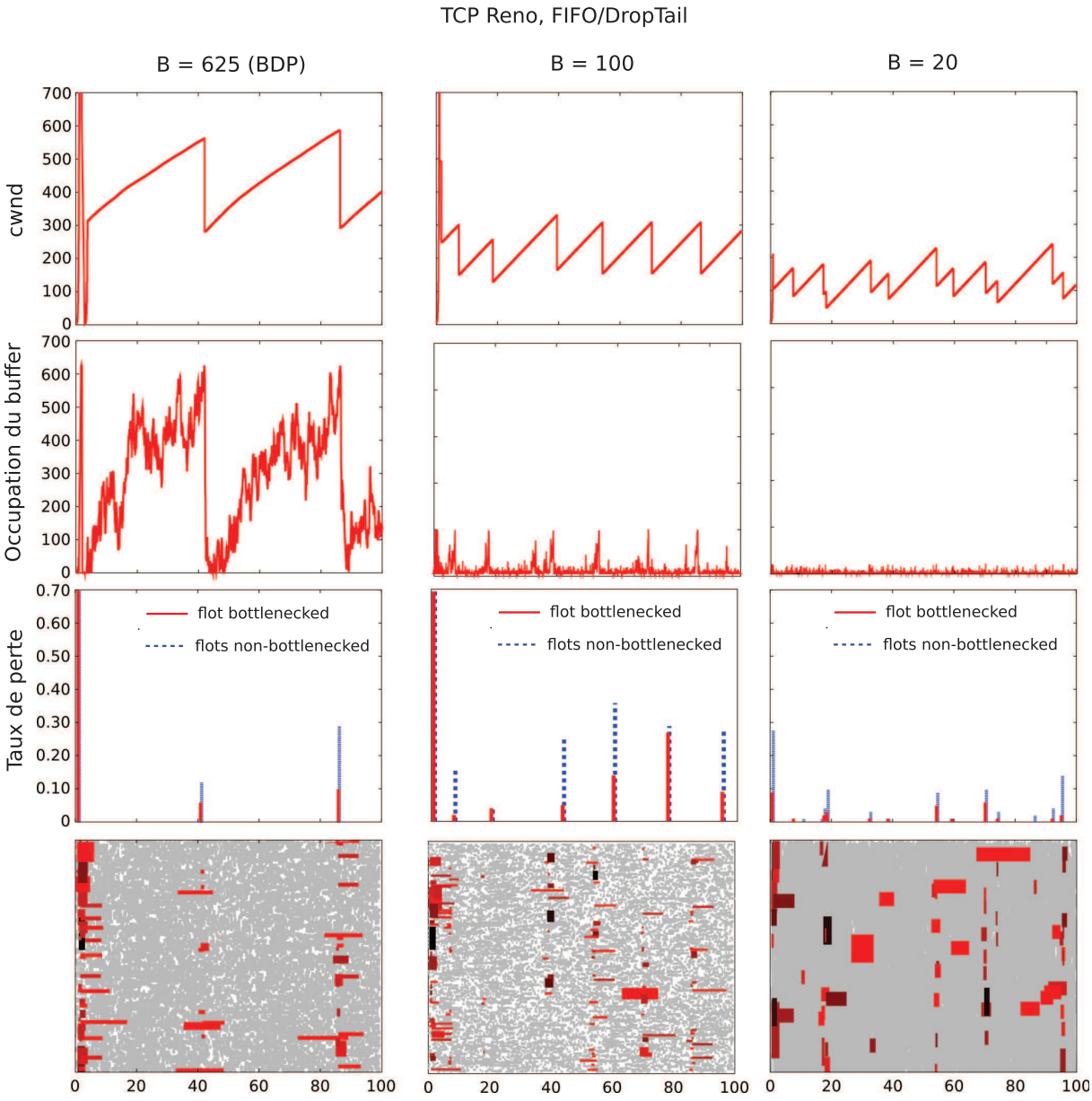


FIGURE 5.1 – Scénario impliquant un flot TCP Reno en compétition avec une charge *background* de 50%, sur un lien FIFO/DropTail, pour différentes tailles de buffer. De haut en bas : *cwnd* du flot TCP, occupation de la file d'attente, taux de pertes subit par les deux classes de flots, et représentation des pertes distribuées sur les flots *non-bottlenecked*

.

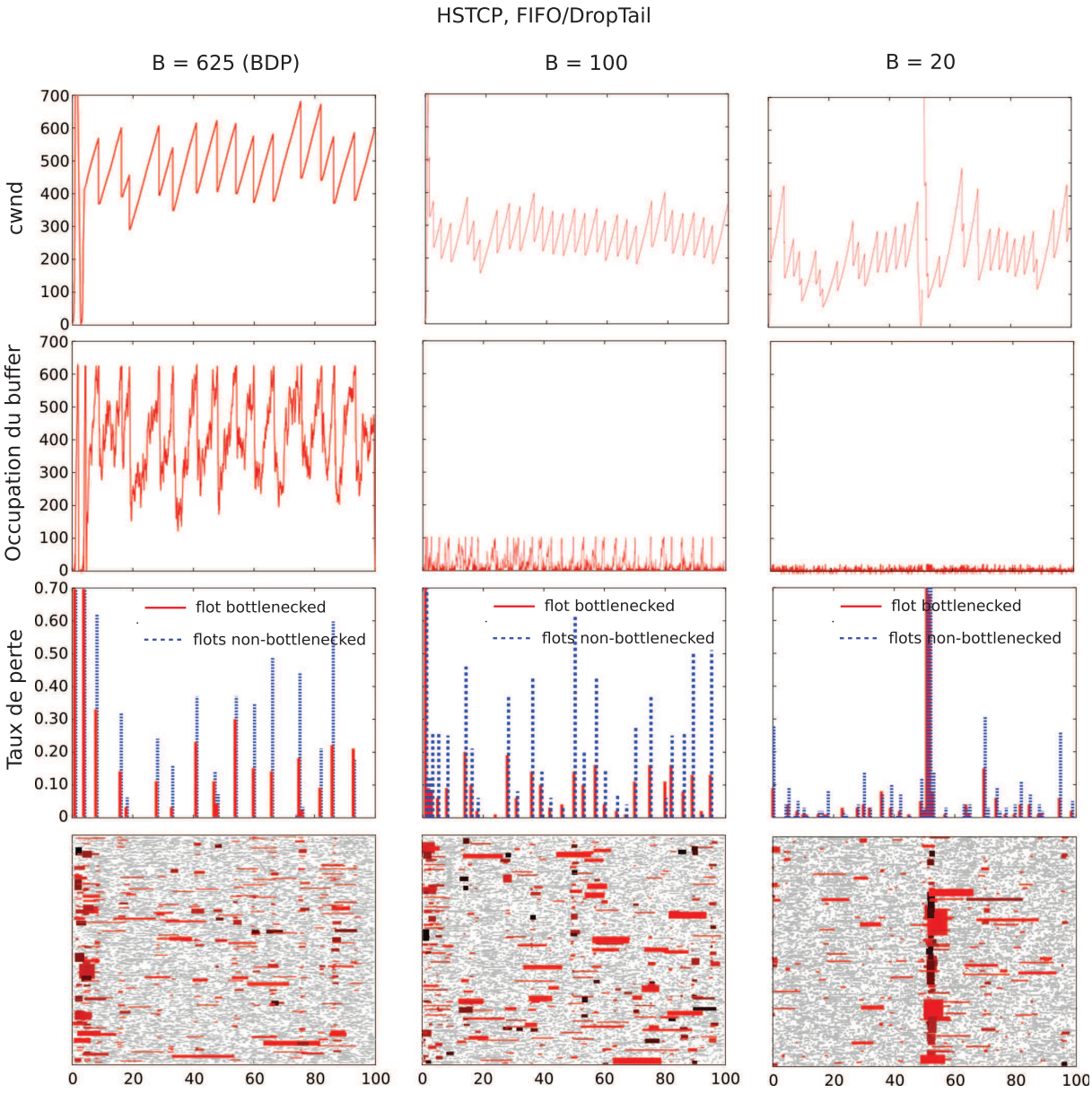


FIGURE 5.2 – Scénario impliquant un flot HSTCP en compétition avec une charge *background* de 50%, sur un lien FIFO/DropTail, pour différentes tailles de buffer. De haut en bas : *cwnd* du flot TCP, occupation de la file d'attente, taux de pertes subit par les deux classes de flots, et représentation des pertes distribuées sur les flots *non-bottlenecked*

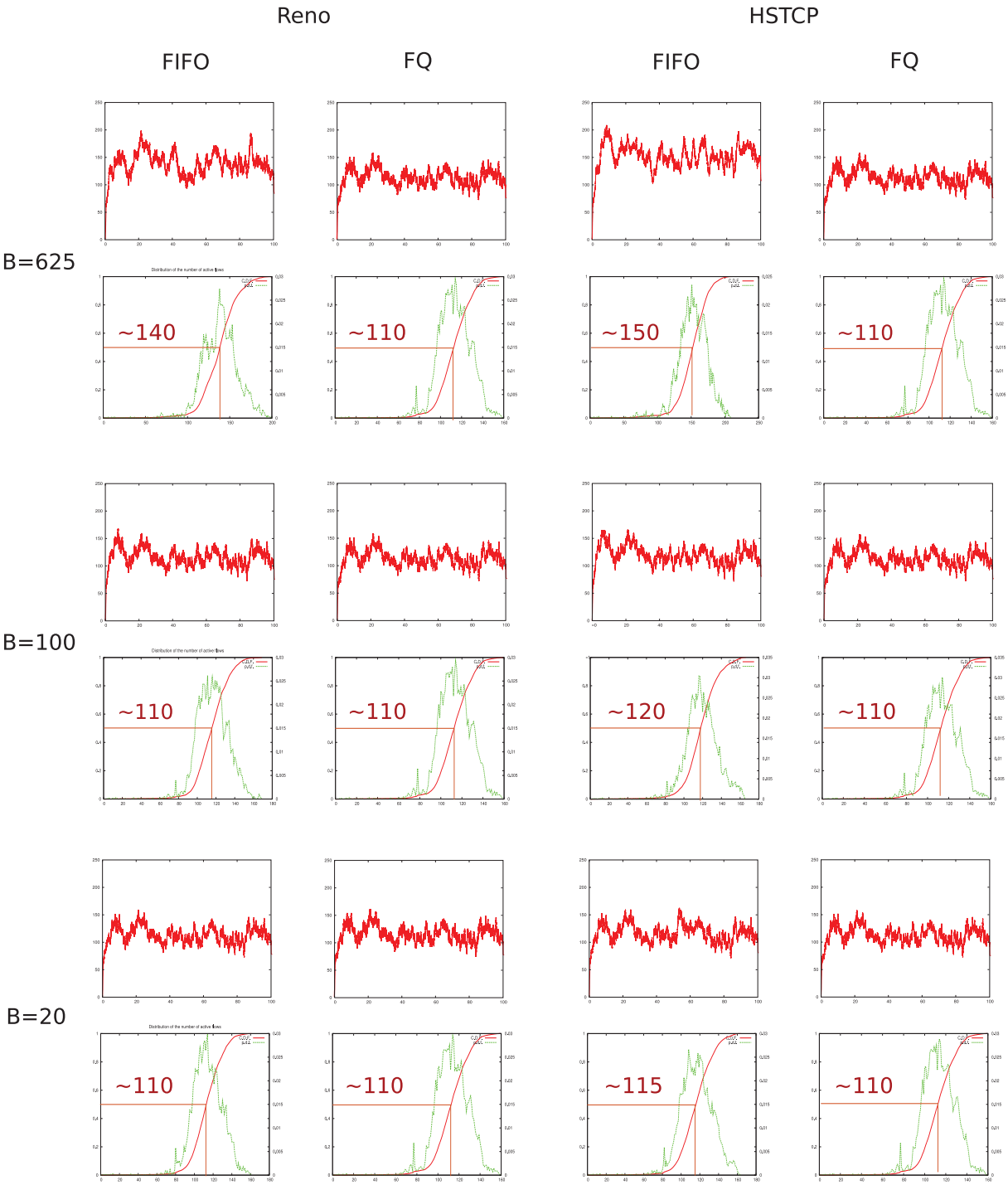


FIGURE 5.3 – Nombre de flots en cours (figure du haut), et densité/distribution/mediane (figure du bas), pour différents scénarios impliquant un flot TCP Reno ou HSTCP en compétition avec une charge *background* de 50%, sur un lien FIFO et FQ, pour différentes tailles de buffers.

mance réalisée par les flots *non-bottlenecked* est similaire au cas $B = 20$ avec l'utilisation de NewReno. Elle reste également acceptable pour HSTCP, malgré la présence de plus nombreux événements de saturation qui causent des pertes importantes.

Utilisation de *fair queueing*

Dans l'ensemble des scénarios FIFO considérés, la performance des flots est dégradée, c'est pourquoi nous recommandons l'utilisation de *fair queueing*. LQD élimine les pertes pour les flots *non-bottlenecked* (et permet de garantir un débit moyen équitable pour les autres), tandis que le rejet d'un paquet en tête de file (FrontDrop) permet une meilleure réactivité des flots TCP [147].

Les résultats utilisant FQ correspondant aux figures 5.1 et 5.2 ne sont pas représentés ici. Ils montrent que son utilisation permet de protéger les flots *non-bottlenecked* des pertes, et leur traitement en priorité assure des délais négligeables et une gigue bornée (conjecture de gigue négligeable), pourvu que la charge prioritaire reste contrôlée. Dans le cas de flots à débit crête limité, mais qui auraient accumulé de la gigue dans une autre partie du réseau, cette dernière sera absorbée puisque ces flots seront alors servis au débit équitable (voir Chapitre 6). Bonald *et al* [25] remarquent que l'ordonnancement joue un rôle plus important que le traitement en priorité. La figure 5.3 montre que FQ rétablit la performance optimale en ce qui concerne la distribution du nombre de flots en cours (ces flots sont alors uniquement limités par le *slow-start*).

Nous remarquons que la réalisation de FQ sur le lien permet ainsi le développement de nouvelles méthodes d'estimations fiables de la bande passante disponible, telles que *packet pair* [87], ainsi que de protéger l'ensemble des connexions établis d'un éventuel comportement agressif lors de cette phase de découverte. L'introduction de nouveaux mécanismes permettant une performance plus satisfaisante que les méthodes actuelles de *slow start* reste une piste ouverte de recherche.

5.3.3 Performance des flots *bottlenecked*

Le but de cette section est d'illustrer l'impact de l'introduction de nouveaux protocoles TCP. Nous présentons un certain nombre de résultats de simulation nous permettant d'établir un ensemble d'observations. Les figures présentées sont toujours sous forme de paire avec en haut l'évolution de *cwnd* (la taille de la fenêtre de congestion) pour chaque flot goulotté, et en bas l'utilisation du lien au fil du temps. Nous considérons généralement trois tailles de buffer : 20, 100 et 625 paquets, ce dernier correspondant à un dimensionnement selon la règle empirique dite du *Bandwidth Delay Product*. Dans l'ensemble de ces scénarios, la charge de trafic *background* offerte est de 50%.

Nous limitons dans cette section les résultats présentés au protocole HSTCP, que nous comparons avec TCP Reno. Les résultats sont qualitativement similaires avec l'utilisation des autres variantes de TCP présentées plus haut. Il conviendra de se référer par exemple à Li *et al.* [100] pour une telle étude. Notons toutefois dans nos scénarios la présence de trafic *background* qui affecte par exemple le débit réalisé par un flot en présence de petits buffers, ou permet l'ajout d'un certain aléa qui parfois aide à une convergence plus rapide.

Scénarios avec 1 flot *bottlenecked*

Nous commençons par un scénario identique aux conditions du chapitre précédent, où un seul flot TCP Reno est en compétition avec un trafic *background* de charge 50%, et faisons varier la taille du buffer.

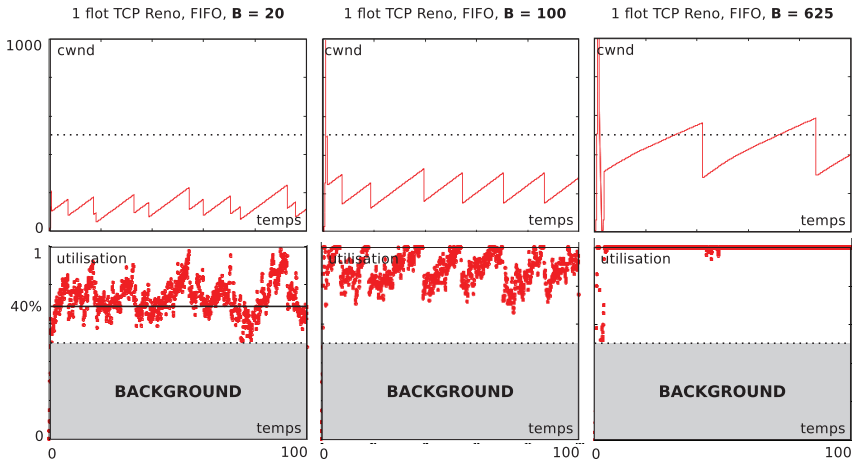


FIGURE 5.4 – Évolution de *cwnd* et utilisation de la bande passante disponible faite par une connexion TCP Reno en compétition avec une charge *background* de 50%, pour des tailles de buffer de 20, 100 et 625 paquets.

Les résultats présentés en figure 5.4 nous permettent de rappeler l'observation faite dans le chapitre précédent, à propos de TCP Reno :

Observation 1. *De très petits buffers de l'ordre de 20 paquets entrainent une perte importante de débit pour TCP Reno. Le flot n'utilise que 40% de la capacité disponible.*

Nous comparons la performance obtenue pour une taille de buffer de 100 paquets, entre TCP Reno et HSTCP (figure 5.5 à gauche).

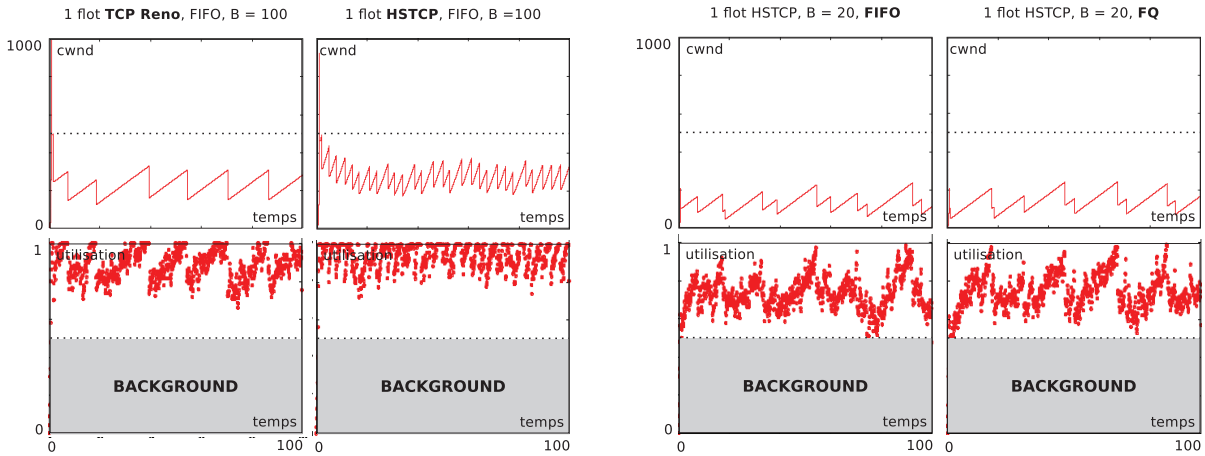


FIGURE 5.5 – Évolution de *cwnd* et utilisation de la bande passante disponible faite par une connexion TCP Reno et une connexion HSTCP en compétition avec une charge *background* de 50%, dans des scénarios FIFO et FQ impliquant plusieurs tailles de buffer.

Observation 2. *HSTCP augmente l'utilisation du lien, au détriment de pertes plus importantes pour les flots* background *(lors des instants de saturation du buffer).*

Nous considérons maintenant le comportement d'un flot HSTCP avec un buffer de 100 paquets, avec un lien FIFO et FQ (figure 5.5 à droite) :

Observation 3. *L'utilisation de fair queueing permet de protéger les flots* background *des pertes, avec un très faible impact sur le flot goulotté (qui semble même positif).*

Scénarios avec 2 flots *non-bottlenecked*

Nous considérons désormais des scénarios avec 2 flots TCP.

Équité intra-protocolaire : Nous comparons en figure 5.6, les comportements respectifs de deux flots TCP Reno (à gauche) et deux flots HSTCP (à droite), pour des tailles de buffer de 20 et 625 paquets, sur un lien FIFO.

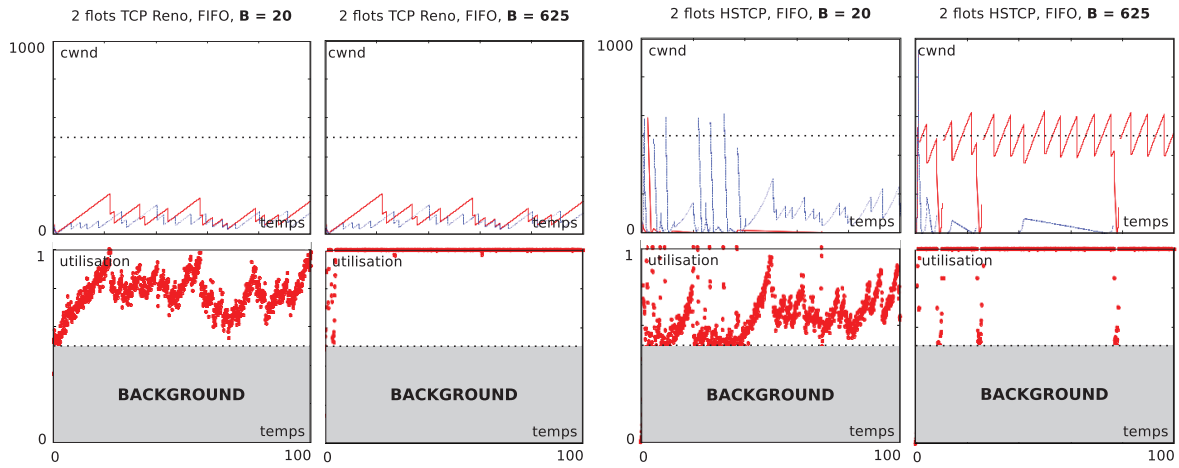


FIGURE 5.6 – Évolution de *cwnd* et utilisation de la bande passante disponible faite par deux connextions TCP Reno (à gauche) et deux connexions HSTCP (à droite), en compétition avec une charge *background* de 50%. Le lien est FIFO et les tailles de buffer de 20 et 625 paquets.

Observation 4. *TCP Reno offre une équité approximative.*

Observation 5. *HSTCP est très inéquitable même dans un contexte homogènes où tous les flots utilisent HSTCP.*

Equité inter-protocolaire : Dans la figure 5.7, nous représentons maintenant la performance réalisée par un flot TCP Reno en compétition avec un flot HSTCP, pour des tailles de buffer de 20 et 625 paquets sur un lien FIFO (les deux figures de gauche), et FQ (les deux figures de droite).

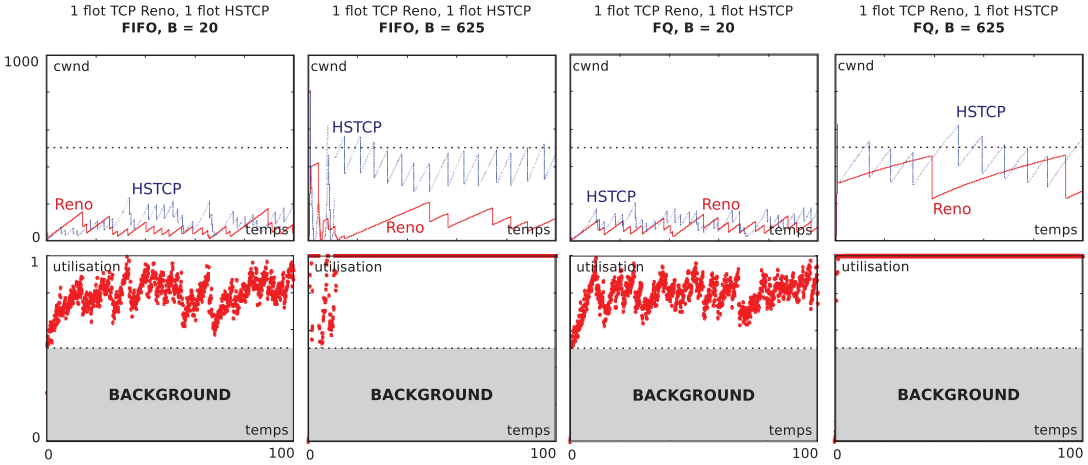


FIGURE 5.7 – Évolution de *cwnd* et utilisation de la bande passante disponible faite par un flot TCP Reno en compétition avec un flot HSTCP, en compétition avec une charge *background* de 50%. Le lien est respectivement FIFO (les deux figures de gauche) et FQ (les deux figures de droite), et les tailles de buffer considérées sont de 20 et 625 paquets.

Observation 6. *HSTCP est très inéquitable envers TCP Reno.*

Observation 7. *L'ajout de* fair queueing *est très efficace pour améliorer l'équité, bien que HSTCP soit plus agressif et obtiennent plus de débit. Il revient au protocole de transport d'utiliser au mieux la capacité qui lui est offerte. Ici AIMD ne récupère que très lentement son débit après une perte.*

Scénarios avec 4 flots goulottés

Nous comparons maintenant, en figure 5.8, les comportements respectifs de 1 et 4 flots TCP Reno, pour des tailles de buffer de 20 paquets (les deux figures de gauche), et 625 paquets (les deux figures de droite).

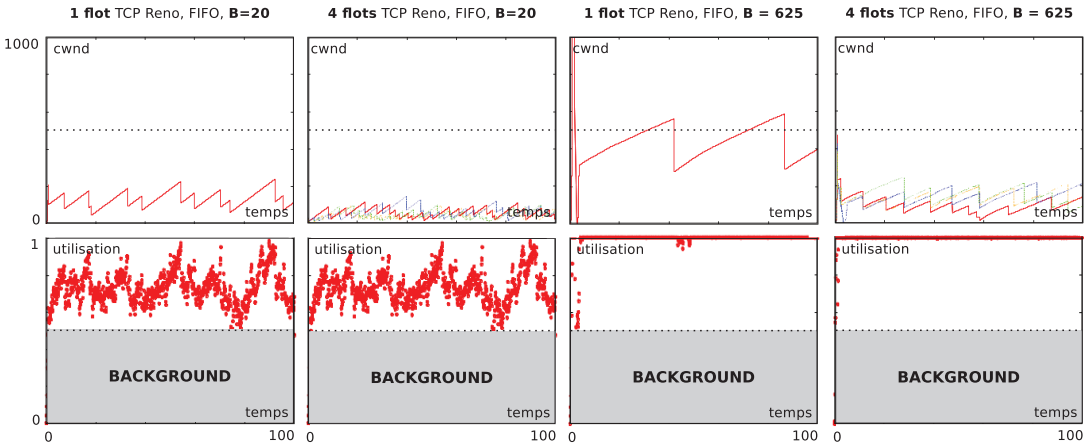


FIGURE 5.8 – Évolution de *cwnd* et utilisation de la bande passante disponible faite par 1 et 4 flots TCP Reno en compétition avec une charge *background* de 50%. Le lien est FIFO, et les tailles de buffer considérées sont de 20 paquets (les deux figures de gauche) et 625 paquets (les deux figures de droite).

Observation 8. *Pour TCP Reno, l'équité est généralement approximative avec de petits ou de grands buffers. Elle est meilleure lorsque le nombre de flots concurrents augmente.*

Nous effectuons la même simulation en figure 5.9 avec 1 flot (figure de gauche) et 4 flots HSTCP (figure du milieu), pour une taille de buffer de 20 paquets, et un lien FIFO.

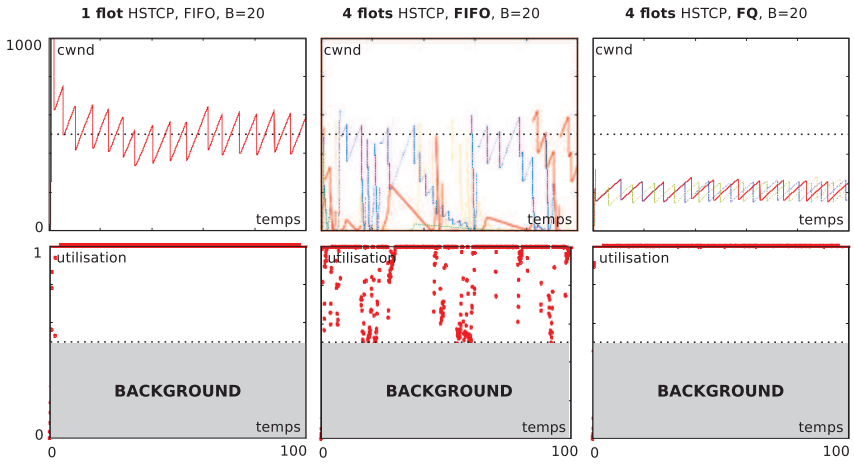


FIGURE 5.9 – Évolution de *cwnd* et utilisation de la bande passante disponible par 1 flot (à gauche) et 4 flots HSTCP (au milieu) sur un lien FIFO, et par 4 flots HSTCP sur un lien FQ (à droite). Les flots sont en compétition avec une charge *background* de 50%, et la taille de buffer considérée est de 20 paquets.

Observation 9. *En présence de 4 flots HSTCP, on observe un comportement assez chaotique des flots, où chaque flot domine temporairement l'un après l'autre. Aussitôt qu'un flot possède une grande taille de fenêtre cwnd, il tend à être plus agressif et présence son avantage relatif jusqu'à ce qu'il soit éventuellement détrôné par un autre. Ce genre de comportement à été observé précédemment par les flots un flot après l'autre. Le manque d'équité ainsi que la perte de débit ont également été observés par Li* et al. *[100].*

Enfin, la partie la plus à droite de la figure 5.9 présente un scénario impliquant 4 flots HSTCP, toujours pour un buffer de 20 paquets, mais fois-ci en présence de FQ.

Observation 10. *L'introduction de* fair queueing *rétablit l'équité et permet de préserver le débit des flots et une utilisation optimale du lien. Il atténue considérablement l'impact de l'arrivée de nouveaux flots sur ceux déjà en place, pour lesquels le comportement agressif du* slow start *conduirait à des événements de perte importants.*

Observation 11. *Deux avantages supplémentaires du* fair queueing *sont des événements de pertes désynchronisés en raison de l'isolation entre les flots, et le fait que les émissions de paquets se retrouvent espacées au débit équitable courant. Ce dernier effet signifie que les acquittement sont également lissés, ce qui conduit à moins de rafales dans le prochain cycle d'émissions de paquets. Ces deux phénomènes semblent améliorer la performance du partage de bande passante, notamment pour de petites tailles de buffers.*

5.4 Comportement de TCP dans un environnement équitable

5.4.1 Motivations

Les résultats préliminaires que nous venons d'établir mettent en évidence les problèmes liés à la présence de flots TCP à fort débit. D'une part, le manque d'isolation entre les flots entraîne des perturbations pour les flots streaming, qui devraient être protégés (et qui ont déjà un faible débit). D'autre part, contrairement aux hypothèses souvent faites, les différentes versions de TCP ne permettent pas un partage équitable, efficace et instantané de la capacité disponible.

TCP

Une partie de ces imperfections provient de la multiplicité des rôles que doit assurer le protocole. En plus de garantir une connexion fiable entre les hôtes (grâce aux mécanisme d'acquittement et de retransmission), TCP a pour objectif (au travers de l'algorithme AIMD pour sa version classique) la réalisation :

- d'une utilisation optimale de la bande passante disponible sur le lien (nous avons vu précédemment que TCP s'appuie sur la présence de buffers suffisamment dimensionnés) ;
- de l'allocation équitable entre les différentes connexions de cette capacité disponible (et de l'espace buffer).

Cette dernière tâche repose généralement sur la détection par TCP de la congestion d'un lien, par des heuristiques basées sur les pertes de paquets ou les délais constatés. Elle nécessite de plus la coopération des utilisateurs, et ne tient pas compte de protocoles non-réactifs tels qu'UDP. En particulier, une propriété souvent recherchée dans la conception de nouveaux protocoles – et potentiellement limitante – est la rétro-compatibilité avec TCP Reno (TCP *friendliness*) qui garantit qu'un tel protocole en compétition avec un flux TCP Reno sera le plus équitable possible envers lui. HSTCP par exemple possède deux régimes opérationnels, l'un équivalent à TCP Reno, l'autre plus agressif. Les résultats présentés dans le Chapitre 3, d'insensibilité notamment, reposent sur un partage équitable des ressources.

Fair Queueing

La discipline FIFO ne permet pas d'éviter les pertes de paquets dues à la saturation du buffer. Nous avons vu dans le Chapitre 3 que la présence de *fair queueing* au sein du réseau permettrait de mieux répondre à ces problèmes, sans nécessiter de mécanisme complexe de contrôle de trafic, et qu'un tel ordonnanceur est réaliste dans un contexte de cœur de réseau. Les simulations précédentes illustrent les bénéfices d'une telle proposition. Les flots sont isolés entre eux, ce qui assure notamment l'absence de perte et des délais négligeables pour les flots *streaming*. De plus, des flux de nature très hétérogènes peuvent coexister et se partager équitablement la bande passante.

Nous avons également vu qu'un tel mécanisme est fondamental afin de dimensionner et d'opérer un réseau de manière robuste et prévisible, en suivant des modèles simples de trafic. Il permet de rendre réalistes les hypothèses d'équité entre les flots et de convergence rapide vers l'état d'équilibre. Dans la suite, nous allons explorer la faculté des différents algorithmes à exploiter efficacement la bande passante disponible. Notre objectif ici est de démontrer que l'introduction de *fair queueing* permet le développement de nouveaux protocoles TCP à la fois simples et efficaces, sans nuire à l'existant.

5.4.2 Environnement de simulation

Nous considérons l'environnement de simulation introduit dans la section précédente, avec les modifications suivantes :

Approximation Poisson pour les flots *non-bottlenecked*

Comme dans le chapitre précédent, section 4.5.1, nous considérons des arrivées Poisson de paquets générant une charge *background* $\rho_b \in [0,1]$. Ce trafic modélise l'agrégat de flots à débit limité p. Tant que le ratio C/p n'est pas trop élevé, le lien voit un paquet de chaque flot de temps en temps et tous seront traités en priorité par l'algorithme de *fair queueing* PFQ.

Flots TCP

En plus des algorithmes TCP présentés en début de ce chapitre, nous considèrerons des généralisations des algorithmes AIMD et MIMD que nous avons développées pour lesquels il nous sera possible de faire varier les coefficients d'incrément et de décrément.

Bien que les différents algorithmes évalués soient paramétrés pour être *TCP-friendly*, leur diversité nous permet de balayer une large gamme de mécanismes de contrôle de congestion (voir Section 5.2.2). Notre évaluation porte notamment sur l'évolution de la performance du protocole en fonction de la capacité C du lien et du RTT des flots au travers de leur produit $C \times RTT$ (ou *Bandwidth Delay Product*), la taille B du buffer, ainsi que la charge ρ_b des flots *non-bottlenecked* concurrents.

Nous ne considérons pas des flots de RTT différents dans cette section ; leur instant de démarrage sera tiré aléatoirement afin d'éviter des phénomènes artificiels de synchronisation. La présence de *fair queueing* suffira ensuite à prévenir de tels cas.

5.4.3 Modèle d'étude

Débit instantané des flots

Nous avons introduit le paramètre γ, représentant l'espérance du débit instantané, dans le Chapitre 3, Section 3.3.2. Sa formulation requérait toutefois l'équité des débits des flots, qui n'était qu'approximative pour TCP Reno. Cette hypothèse sera ici assurée par la présence de *fair queueing*, pourvu que les protocoles soient suffisamment efficaces pour exploiter la bande passante qui leur est allouée.

Nous rappelons ici l'expression de γ :

$$\gamma = \frac{\rho}{E[X](\rho)} = \frac{\sum \rho^i / \Phi(i)}{\sum i \rho^{i-1} / \Phi(i)}. \tag{5.1}$$

avec $\Phi(i) = \prod_1^j \phi(j)$, et $\phi(i)C$ le taux de service du lien lorsque i flots sont en cours.

γ est une fonction rationnelle n'admettant pas 0 pour pôle, elle est donc développable en série entière en 0 telle que : $\sum_{k=0}^{\infty} \beta_k \rho^k$. On a en particulier : $\gamma = \phi_1 + (1 - 2\phi_1/\phi_2)\rho + o(\rho^2)$.

Il est possible d'en déduire le temps moyen de complétion d'un flot, $R(s)$:

$$R(s) = \frac{s}{\gamma(\rho)}$$

.

Dans les simulations mettant en œuvre TCP Reno, la courbe gamma était approximativement linéaire avec pour ordonnée à l'origine $\phi(1)$. Ce n'est généralement pas le cas. L'expression de γ n'est pas surprenante dans la mesure où à faible charge, la probabilité d'avoir 1 ou 2 flots est largement dominante.

Remarques

Nous ferons, comme dans le chapitre précédent, l'hypothèse simplificatrice qui consiste à négliger le temps de convergence des protocoles vers un état équitable. Ce modèle nous permet toutefois d'obtenir une estimation qualitative de la performance réalisée par les différents protocoles TCP, plus juste que certains modèles considérant uniquement des flots permanents et sans charge *background*.

Nous remarquons qu'à faible charge, on a très peu de flots et le changement de débit important dû à une arrivée ou un départ est balancé par un taux d'arrivée faible, et un taux de départ réduit (faible efficacité). A forte charge, les taux de départ et d'arrivée sont importants, mais le grand nombre de flots en présence induit de faibles variations de débit. Il sera intéressant de voir dans quelle mesure ces phénomènes ont un impact sur la performance effectivement réalisée.

5.4.4 Résultats

Dans cette discussion, nous allons présenter les principales observations faites à partir de ces simulations.

Performance des différentes versions de TCP avec de petits buffers

La figure 5.10 représente les valeurs de ϕ et γ pour notre scénario de référence ($B = 20$ paquets, et $\rho = 0.5$). Comme dans le chapitre précédent, la performance théorique du modèle PS est représentée en noir.

Pour l'utilisation de la capacité disponible $\phi(.)$, nous avons représenté l'axe des abscisses – le nombre de flots en cours – selon une échelle logarithmique. Un nombre faible de flots correspond en effet aux cas les plus probables pour des charges pas trop élevées. Nous nous limitons uniquement à une valeur indicative de la performance, et n'avons pas représenté les intervalles de confiance sur la figure à des fins de clarté.

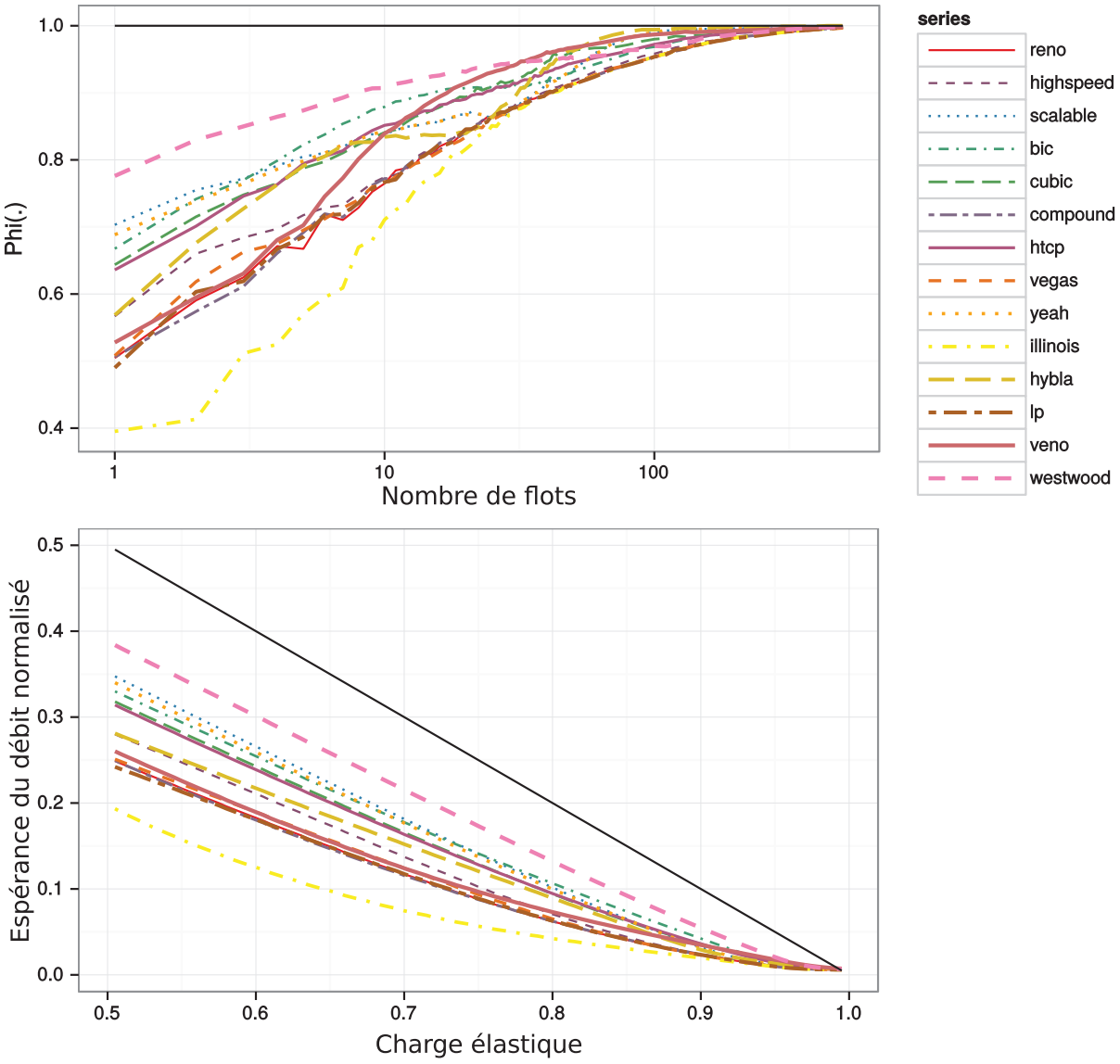


FIGURE 5.10 – Performance des différentes versions de TCP, $B = 20$, $\rho = 0.5$. La valeur théorique pour le modèle PS est représentée en noir.

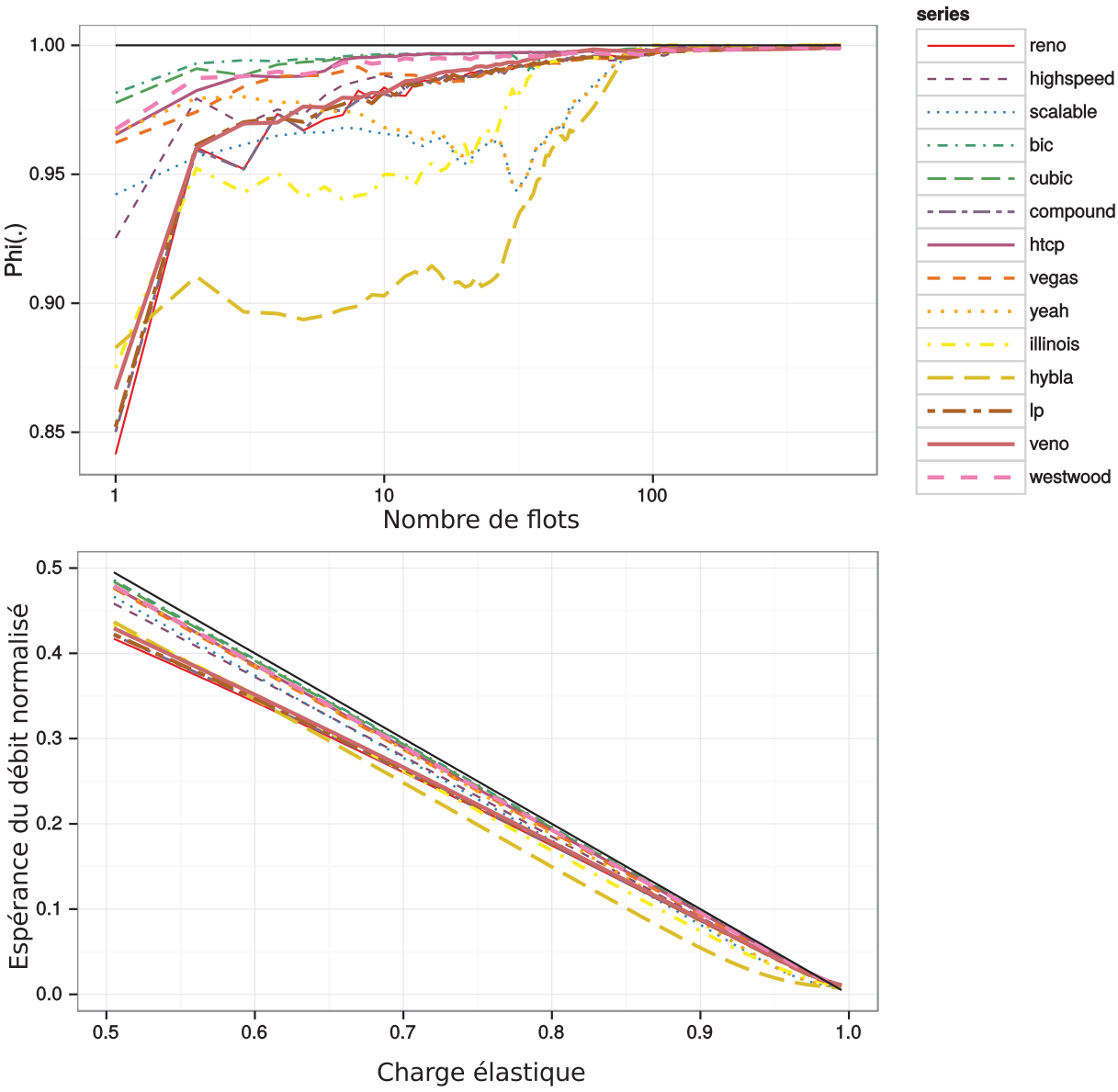


FIGURE 5.11 – Performance des différentes versions de TCP, $B = 100$, $\rho = 0.5$. La valeur théorique pour le modèle PS est représentée en noir.

Nous remarquons que l'utilisation varie du simple au double, avec $\phi(1) \sim 0.4$ pour TCP Hybla, et $\phi(1) \sim 0.8$ pour TCP Westwood ; et qu'elle croît dans tous les cas lentement, en fonction du nombre de flots en présence.

La plupart des protocoles permettent d'améliorer la performance obtenue par TCP Reno. Celle-ci est d'ailleurs légèrement supérieure à celle obtenue dans le chapitre précédent, sans l'utilisation de *fair queueing*. Ceci peut s'expliquer par un lissage des rafales de paquets par l'ordonnanceur, ce qui permet à la fenêtre de congestion de croître plus longuement avant de causer des pertes de paquets.

L'utilisation d'une taille plus importante de buffer ($B = 100$ paquets) est représentée en figure 5.11. Cette fois-ci, l'utilisation dépasse les 85%, voire 95% pour la plupart des protocoles. Nous remarquons que ce sont cette fois-ci les protocoles BIC et CUBIC qui obtiennent une meilleure performance, et qu'ils semblent ainsi plus sensibles à de petits buffers. Avec un choix correct de protocoles et une taille de buffers de 100 paquets, on obtient ainsi une performance satisfaisante (elle était déjà correcte pour TCP Reno).

Alors que ce n'était pas le cas pour $B = 20$ paquets, γ décroit ici de manière aproximativement linéaire à la charge.

Le cas d'un buffer dimensionné au *Bandwith Delay Product* n'est pas représenté ici mais il permet une performance optimale pour l'ensemble des protocoles.

Nous avons enfin représenté, en figures 5.12 et 5.13, la relation entre entre l'espérance du débit normalisé pour 1 flot (en abscisse) et le taux de pertes subit (en ordonnée), pour des tailles de buffer

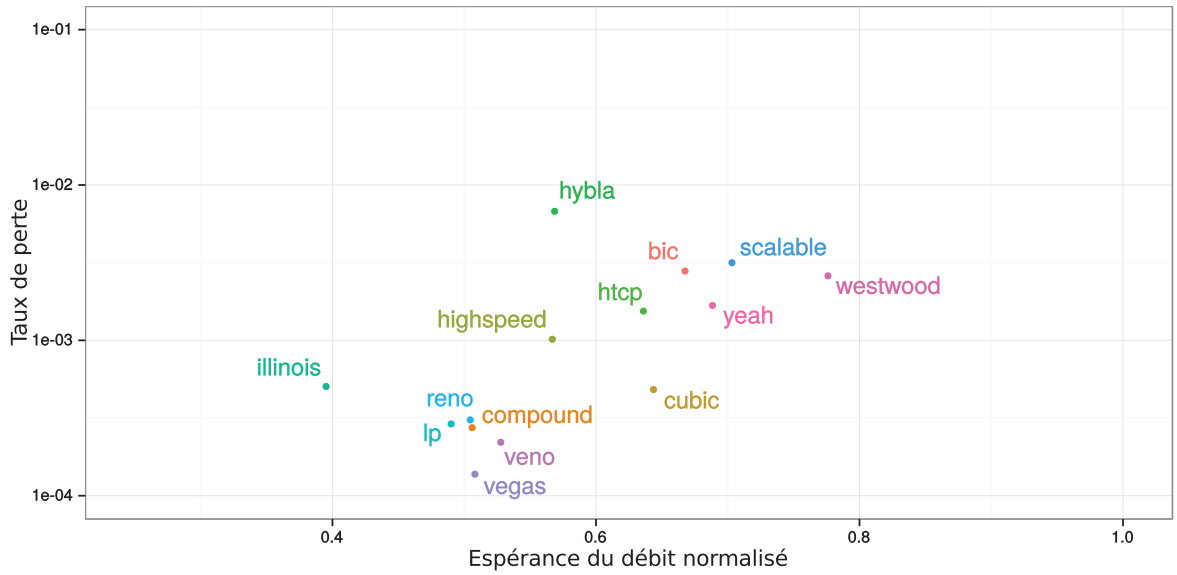


FIGURE 5.12 – Performance réalisée par les différents protocoles TCP pour une taille de buffer $B = 20$ paquets : compromis entre l'espérance du débit normalisée, pour 1 flot (en abscisse) et le taux de pertes subit (en ordonnée).

respectives de 20 et 100 paquets. Le fonctionnement interne de chaque protocole aboutit à un couple débit/taux de perte dépendant de l'environnement considéré, certains étant plus adaptés que d'autres à de petits buffer. Cette figure illustre en outre le compromis nécessaire entre le comportement du protocole – son agressivité/opportunisme pour émettre des paquets – et le débit réalisé. Nous reviendrons plus tard sur ce compromis, sachant qu'un taux de perte cible minimum semble inévitable dans un tel contexte.

Impact de la charge *background*

Comme attendu, on constate une forte sensibilité à la charge *background* de l'ensemble des protocoles pour $B = 20$ paquets. L'ordre entre les protocoles reste sensiblement le même pour $\rho_b = 0.25$, $\rho_b = 0.50$ et $\rho_b = 0.75$.

Avec une taille $B = 100$ paquets, on obtient une performance acceptable pour tous les protocoles. L'ordre est majoritairement conservé, mais on constate que les protocoles comme BIC, CUBIC, H-TCP et Scalable TCP arrivent cette fois-ci en tête en même temps que Westwood.

La performance de Westwood avec de faibles buffers n'est pas surprenante puisque ce protocole est une adaptation de Reno qui estime la bande passante disponible à partir du flux des ACKs, et ajuste la diminution de *cwnd* en fonction. Ce mécanisme lui permet d'être moins sensibles aux pertes aléatoires liées aux petits buffers, qui n'ont aucun rapport avec la congestion. Nous verrons dans la suite l'importance du paramètre de réduction de la fenêtre.

Sensibilité des protocoles au dimensionnement

Nous comparons la performance de Reno avec celle obtenue pour Westwood, qui est l'algorithme offrant le meilleur compromis pour différentes configurations de charge *background* et de buffer. La figure 5.14 présente le débit atteint par une connexion TCP Reno (à gauche) et Westwood (à droite), en fonction de ρ_b (en abscisse) et B (en ordonnée).

Nous considérons d'abord le cas $C = 50$Mb/s. Pour une taille $B = 20$ paquets, Westwood offre une performance acceptable jusqu'à des charges d'environ 50%, alors que le débit d'un flot TCP Reno est fortement dégradé. Les besoins en buffer du protocole sont ainsi beaucoup plus réduits, de par sa résistances aux pertes aléatoires. Un buffer d'une centaine de paquets convient, pour tous les seuils de charge *background*.

A plus haute capacité ($C = 100$Mb/s), Westwood réalise toujours une bien meilleure performance que Reno. La performance, moyenne pour $B = 20$, devient rapidement bonne au-delà.

Étant basé sur TCP Reno, Westwood souffre également de la lenteur de la croissance additive de *cwnd*, qui n'est pas trop visible dans ces simulations avec un seul flot stationnaire. Ceci explique

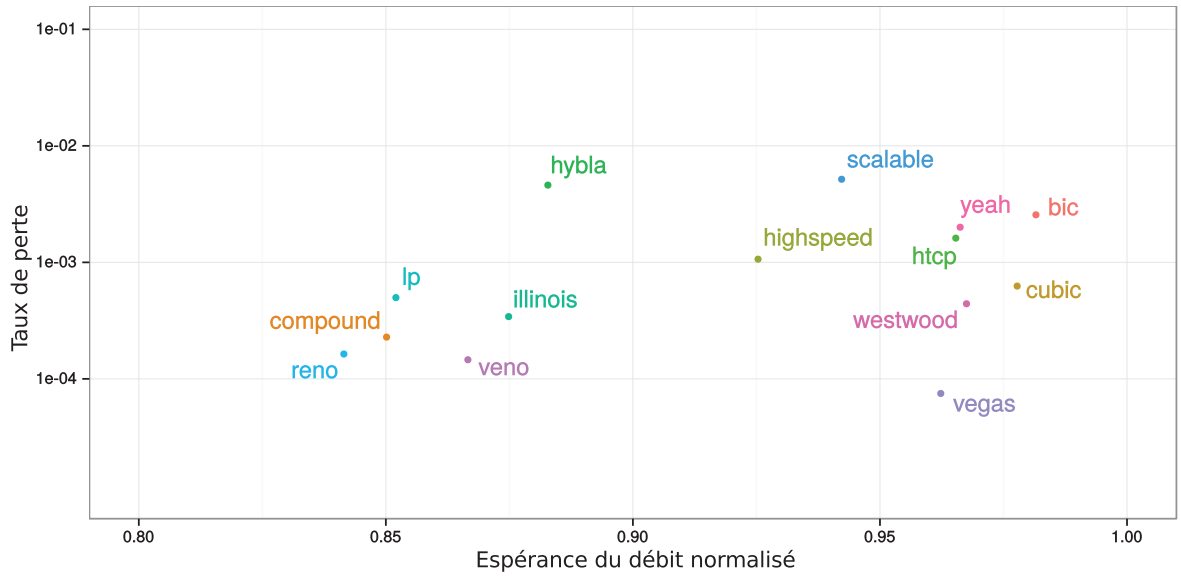
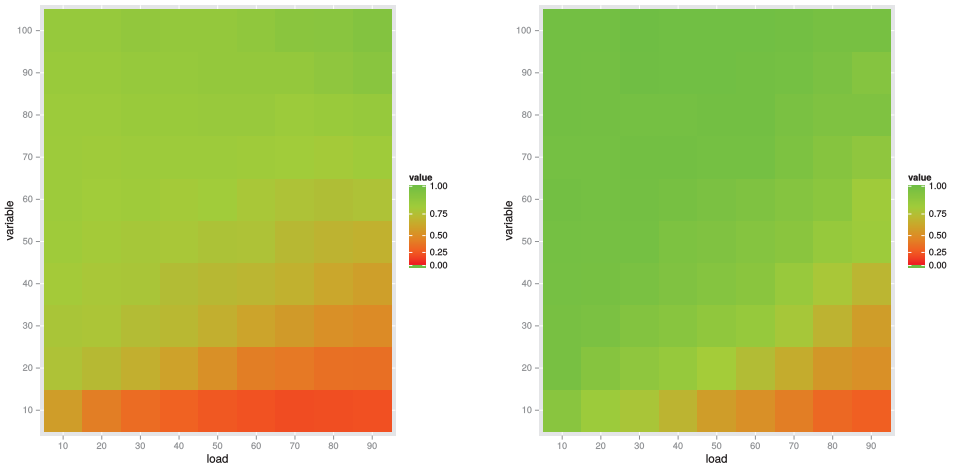
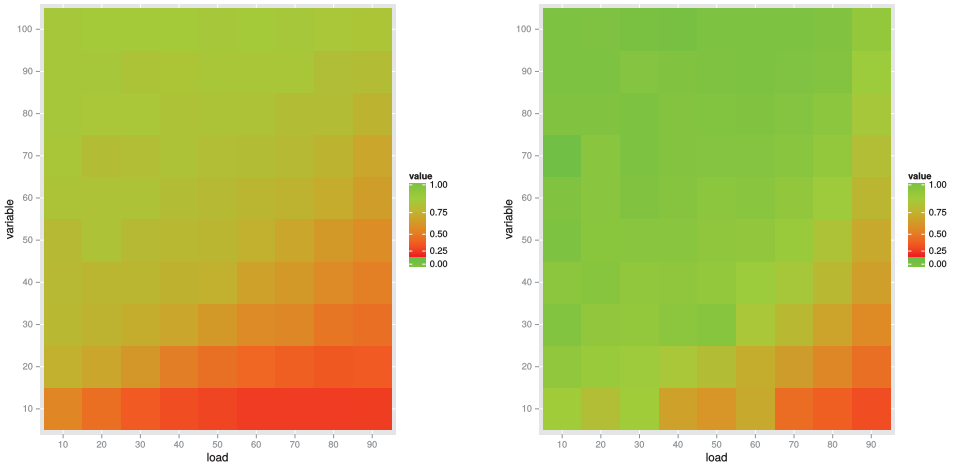


FIGURE 5.13 – Performance réalisée par les différents protocoles TCP pour une taille de buffer $B = 100$ paquets : compromis entre l'espérance du débit normalisé, pour 1 flot (en abscisse) et le taux de pertes subit (en ordonnée).



(a) TCP Reno (b) TCP Westwood

FIGURE 5.14 – Performance de TCP pour différentes valeurs de B et ρ_b, pour $C = 50$Mb/s



(a) TCP Reno (b) TCP Westwood

FIGURE 5.15 – Performance de TCP pour différentes valeurs de B et ρ_b, pour $C = 100\text{Mb/s}$

probablement la dégradation lors de l'augmentation de la capacité.

Versions plus agressives AIMD et MIMD

Les nouvelles versions de TCP sont généralement conçues avec le double objectif de respecter l'équité avec TCP Reno, et avec d'autres connexions utilisant le même protocole : le protocole doit permettre à un nouvel entrant de récupérer de la bande passante, tandis que ce dernier ne doit pas dégrader la performance des connexions en cours. Cela limite nécessairement l'efficacité du protocole, qui ne peut être suffisamment agressif dans les phases *slow start* et *congestion avoidance*. Ces limites n'ont plus de raison d'être si le réseau réalise une allocation équitable.

Nous avons évalué 2 versions d'un protocole TCP plus agressif, basées respectivement sur des algorithmes AIMD – inspiré de TCP Reno – et MIMD – inspiré de Scalable TCP, pour lesquels il nous est possible de faire varier les paramètres de croissance (α) et de décroissance (β) de la fenêtre. Les figures 5.18 et 5.19 représentent respectivement le taux d'utilisation et le taux de pertes de paquets obtenus pour différentes configuration de α et β, le débit cible étant de 25Mb/s.

AIMD : Nous remarquons que les performances peuvent être sensiblement améliorées avec une plus grande valeur de α et une plus faible valeur de β. Nous passons par exemple d'environ 12Mb/s avec les valeurs par défaut de Reno ($\alpha = 1$, $\beta = 2$) à plus de 19Mb/s pour $\alpha = 5$ et $\beta = 10$.

C'est le paramètre de décroissance qui semble jouer le rôle le plus important : il permet à l'algorithme d'être **moins sensible aux pertes** de paquets, qui sont "normales" pour de petites tailles de buffers. L'amélioration de la performance se fait au prix d'une augmentation du taux de pertes, qui reste cependant raisonnable pour les configurations que nous avons évaluées. On peut cependant s'attendre à ce qu'un tel algorithme soit dépendant de la capacité du lien (*additive increase*).

Les valeurs trop importantes de α semblent causer un taux important de pertes, et ne permettent pas à la connexion d'utiliser plus de bande passante. Ces pertes sont probablement dues à l'émission de **rafales** de paquets d'autant plus importantes que le coefficient α est élevé.

MIMD : L'impact de MIMD est plus important : on retrouve un débit de 17Mb/s pour les valeurs par défaut de Scalable TCP ($\alpha = 50$ et $\beta = 3$), qui peut aller jusqu'à 19Mb/s pour des protocoles plus agressifs. On remarque cependant que le taux de pertes croît rapidement avec des valeurs plus agressives de α et β, pour dépasser le taux généralement accepté de 1%. On retrouve le même phénomène de sensibilité aux rafales que décrit précédemment.

Il convient de remarquer que l'augmentation de l'agressivité du protocole, en diminuant la réduction de la fenêtre lors de la détection d'une perte, entraînera de plus forts taux de pertes lorsqu'un nouveau flot entrera en compétition.

La conception d'un protocole adapté à de petites tailles de buffer ($B = 20$) semble cependant un objectif réaliste. Le besoin d'un protocole agressif dans un tel contexte implique un mécanisme de lissage des envois de paquets, que nous considérons ci-après.

Pour de petites valeurs de buffers, il semble difficile de faire beaucoup mieux avec MIMD que Scalable TCP, sauf à accroître au delà de 1% le taux de pertes accepté. Nous soulignons toutefois que les auteurs recommendent l'utilisation de *pacing* avec un algorithme MIMD, pour éviter la formation de rafales.

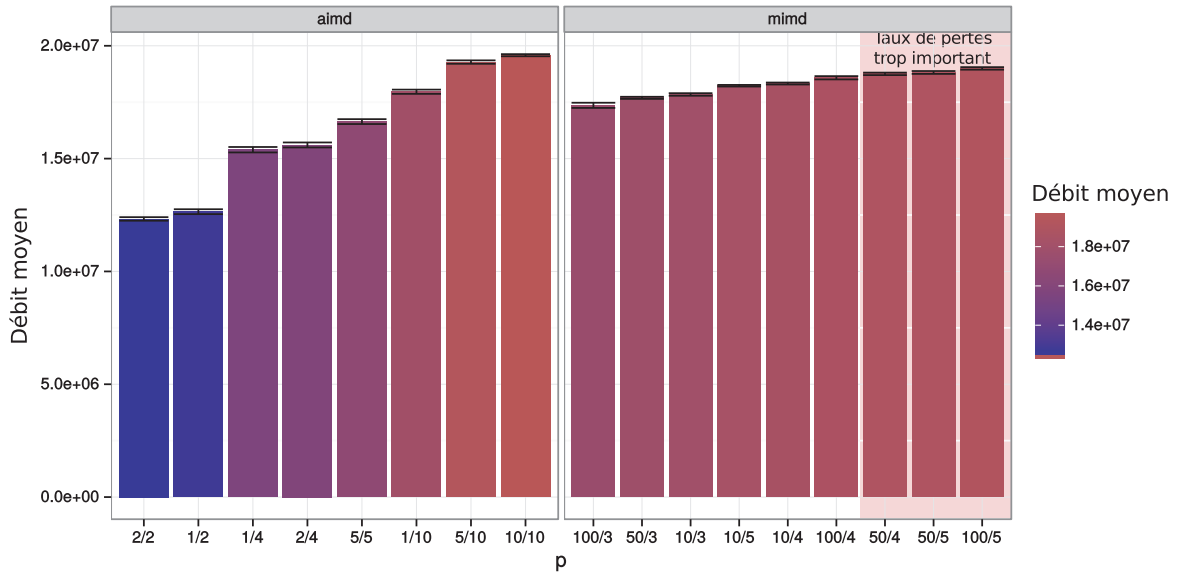


FIGURE 5.16 – Débit réalisé par différents paramétrages des algorithmes AIMD (sur la gauche) et MIMD (sur la droite) pour une taille de buffer $B = 20$ paquets : les coefficients d'incrément (α) et de décrément (β) sont représentés sous la forme α/β.

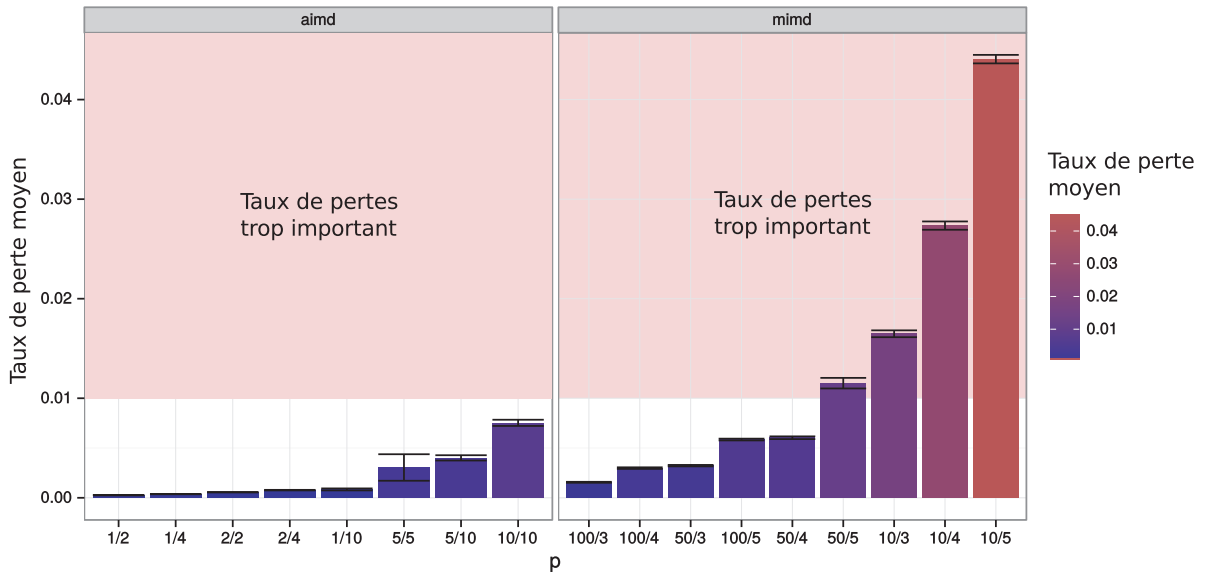


FIGURE 5.17 – Taux de pertes réalisé par différents paramétrages des algorithmes AIMD (sur la gauche) et MIMD (sur la droite) pour une taille de buffer $B = 20$ paquets : les coefficients d'incrément (α) et de décrément (β) sont représentés sous la forme α/β.

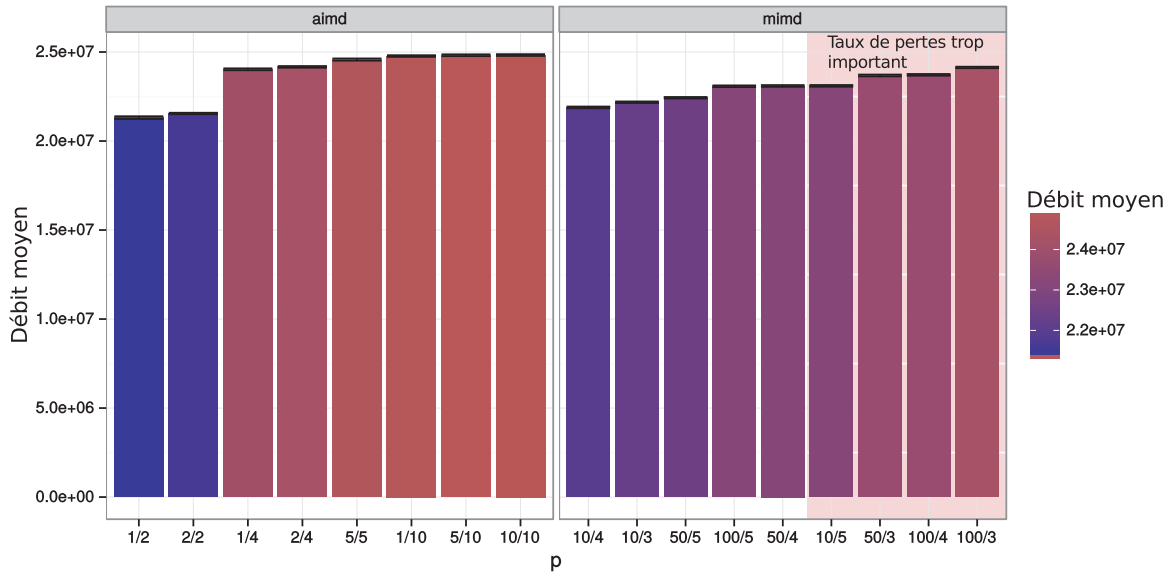


FIGURE 5.18 – Débit réalisé par différents paramétrages des algorithmes AIMD (sur la gauche) et MIMD (sur la droite) pour une taille de buffer $B = 100$ paquets : les coefficients d'incrément (α) et de décrément (β) sont représentés sous la forme α/β.

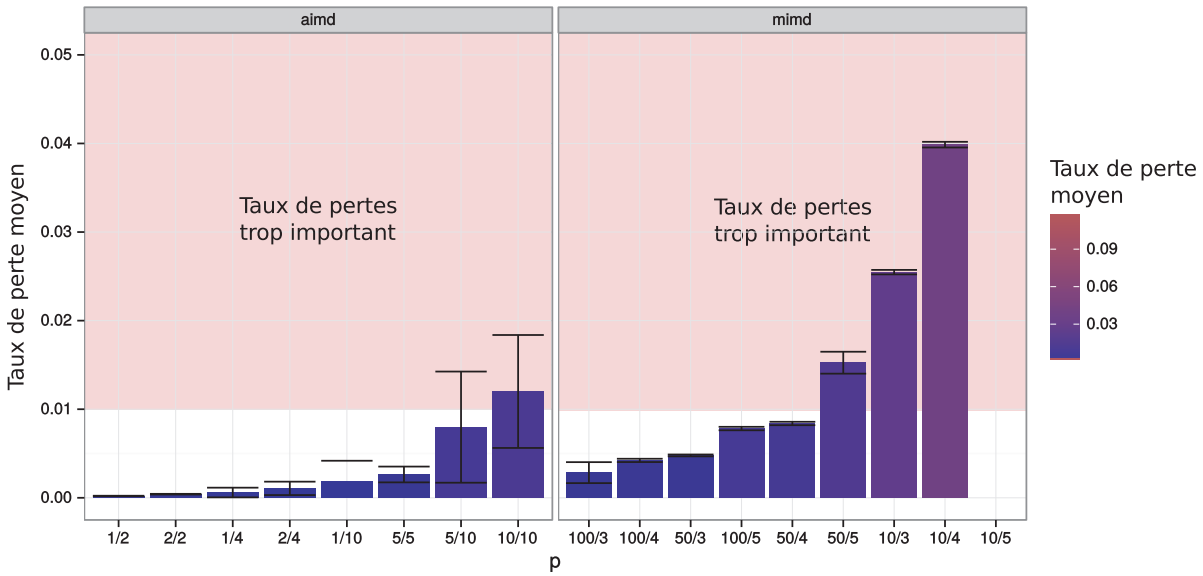


FIGURE 5.19 – Taux de pertes réalisé par différents paramétrages des algorithmes AIMD (sur la gauche) et MIMD (sur la droite) pour une taille de buffer $B = 100$ paquets : les coefficients d'incrément (α) et de décrément (β) sont représentés sous la forme α/β.

L'algorithme TCP Westwood ressort de nos simulations comme le plus adapté aux différentes conditions envisagées, notamment avec des capacités élevées et de petites tailles de buffers. Son équité avec TCP [64] en fait un protocole de choix pour diverses configurations de réseaux, réalisant ou non un ordonnancement *fair queueing*.

Espacement des paquets (*pacing*)

Enachescu *et al.* [50] proposent que les flots qui n'ont pas un débit crête limité par leur lien d'accès utilisent du *pacing*. Un tel mécanisme atténuerait en effet la perte de débit pour de petites tailles de buffers, qui se produisent lorsque le contenu d'une fenêtre de congestion est émis en rafale au début de chaque intervalle de RTT.

Bien que recommandé, aucune implémentation efficace de *pacing* n'a été disponible jusqu'à très récemment [48]. Les implémentations précédentes utilisaient généralement soit des *timers* coûteux pour chaque paquet, soit des spécificités de la couche Ethernet (datagramme PAUSE), perdant ainsi l'accès aux flots individuels. La réalisation faite dans [48] utilise un algorithme proche d'un ordonnanceur *fair queueing*. Elle offre à la couche applicative une limitation optionnelle des débits des flots utilisant un seul *timer*, mais malheureusement n'est pas intégrée avec les versions de TCP.

D'autres résultats discutent de l'intérêt du *pacing* [152], qui peut entraîner une synchronisation des flots, voire une certaine iniquité due à un nombre accru de pertes indépendantes. Nous pensons que l'utilisation de *fair queueing* permet d'éviter de telles situations, et d'exploiter les avantages du *pacing*.

Faute d'implémentation disponible (à la fois en simulation ou dans le noyau Linux), nous nous contentons d'un simple scénario illustratif montrant le gain en débit que peut obtenir un flot TCP *paced*. Il serait intéressant d'étendre plus en détail ces résultats avec les algorithmes AIMD et MIMD plus agressifs introduits précédemment, ainsi que différentes variantes de *pacing*.

Les figures 5.20 illustrent l'évolution de *cwnd* pour un seul flot *bottlenecked* avec[2] ou sans *pacing*, avec $B = 20$ et $B = 100$. Les résultats confirment que le *pacing* améliore sensiblement les performances obtenues pour de petites tailles de buffers puisque le débit obtenu est en gros deux fois plus élevé. La différence est négligeable pour une taille de buffer plus grande, $B = 100$.

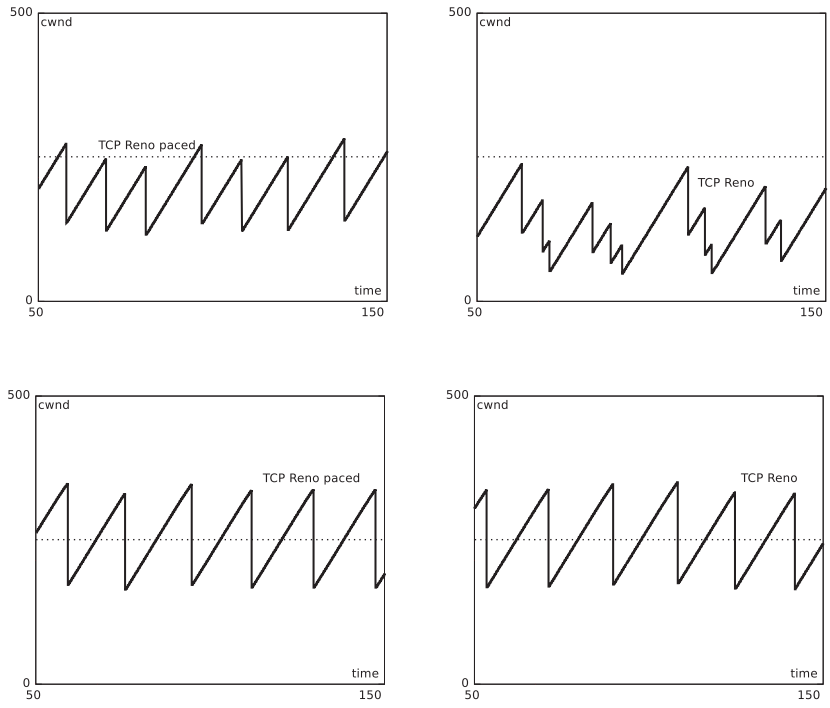


FIGURE 5.20 – Évolution de `cwnd` pour un flot Reno avec *pacing* (à gauche) et un flot Reno sans *pacing* (à droite), pour des tailles de buffers $B = 20$ paquets (en haut) et $B = 100$ paquets (en bas), pour $C = 50$ Mb/s, $\rho_b = 0.5$

L'utilisation de *fair queueing* au sein des routeurs permet également de réguler les arrivées de paquets au sein des flots. Nous en verrons un exemple dans le Chapitre 6, Section 6.5.6. Il en résulte

2. Nous avons utilisé le code TCP ns2 mis à disposition par D.X. Wei : *A TCP pacing implementation for NS2*, disponible à l'adresse `http://www.cs.caltech.edu/~weixl/technical/ns2pacing/index.html`, avec l'option *traditional pacing*.

généralement une probabilité de débordement plus faible, ce qui permet au flot de maintenir une fenêtre de congestion plus importante et ainsi un débit plus élevé.

5.5 Discussion

5.5.1 Enjeux

L'introduction de *fair queueing* présente ainsi des enjeux importants pour un opérateur, afin de pouvoir garantir la performance des applications de ses utilisateurs. Il ne requiert pour cela pas de configuration complexe au sein des routeurs, et permet une évolution saine des protocoles en bordure de réseau.

Son importance nous semble d'autant plus cruciale avec l'arrivée prochaine de réseaux tout-optique, où il n'est techniquement pas possible de réaliser des buffers de plus de quelques dizaines de paquets. Avec la montée des débits, la solution TCP Reno/FIFO/DropTail présente des insuffisances qu'il serait possible de surmonter au moins partiellement grâce à des protocoles plus efficaces, et un meilleur contrôle de l'ordonnancement des flots. Nous avons présenté une méthode mixte mélant un modèle simple à des résultats de simulation permettant d'avoir une estimation qualitative de la performance obtenue par différents protocoles.

Un autre domaine ou l'intégration de *fair queueing* semble décisive est celui des *datacenters*. Ses capacités de différenciation implicite et d'isolation permettront de protéger efficacement les flots court et de leur offrir un temps de réponse satisfaisant, tout en partageant équitablement les ressources pour les autres. Les problèmes d'équité intra-protocolaire sont d'autant plus importants que le besoin de performance motive l'introduction de nouvelles versions de TCP a des degrés de maturité différents (DCTCP, etc.). De plus, l'émergence des *datacenters* partagés, où des machines (virtuelles) sont loués à différents clients, nous pousse à considérer l'interaction de versions TCP hétérogènes comme un cas très probable.

5.5.2 Indicateurs de congestion

La plupart des mécanismes TCP que nous avons considérés utilisent les délais ou les pertes de paquets comme indicateurs de congestion afin d'adapter leur fenêtre de congestion. Avec de petits buffers, les délais d'attente sont faibles et difficiles à estimer, et les pertes de paquets ne sont pas représentatives d'une congestion.

Il convient sans doute de considérer d'autres signaux, prenant en compte non plus les événements, mais plutôt leur distribution ou bien leur moyenne. Par exemple des statistiques sur une certaine échelle de temps du taux de pertes de paquets ou du débit peuvent offrir une bonne indication de l'état du réseau. FQ permet de supprimer l'excédent de trafic par rapport au *fair rate*.

Il serait intéressant de voir dans quelle mesure il est possible de définir un protocole de contrôle plus adapté à de petites tailles de buffers, et permettant un meilleur compromis entre le taux de pertes réalisé et le débit atteint. Idéalement, l'objectif d'un tel algorithme serait d'émettre un paquet au sein de chaque *busy period* de la file d'attente (dont l'évolution est aléatoire), afin d'être servi au débit équitable. Cela nécessite un algorithme suffisamment opportuniste pour profiter de la capacité non occupée par les autres flots et de l'espace disponible dans le buffer, mais sans toutefois risquer de perdre trop de paquets. Ce protocole exploiterait avantageusement le compromis entre son agressivité, et le taux de perte subit.

Un protocole ayant un taux de pertes de paquets cible de 1% par exemple ne nous semble pas irréaliste, afin de pouvoir estimer convenablement cette valeur. Il convient de garder ce taux tout de même relativement restreint afin de ne pas consommer des ressources inutilement sur les liens, même si le débit gaspillé fait partie de l'allocation équitable d'un utilisateur (phénomène dit de *dead packet*). Au choix du protocole, ces pertes pourront être compensées par un mécanisme de retransmission, comme dans TCP, ou par l'utilisation de codes correcteur d'erreur.

5.5.3 Apports du *fair queueing*

La présence de *fair queueing* dans le réseau permettrait l'utilisation de mécanismes de contrôle de congestion alternatifs, comme la proposition *packet pair* de Keshav [86], qui permet de démarrer plus rapidement les connexions (typiquement en mode *slow start*) : le débit équitable est estimé au bout de quelques paquets. Cela peut nous permettre de réduire sensiblement le temps de réponse des flots courts. Une méthode encore plus simple et radicale serait de commencer au débit maximum, pour ensuite réguler le débit au moyen d'un algorithme AIMD/MIMD suffisamment efficace par exemple, ou

tout autre proposition satisfaisante. Les quelques simulations que nous avons effectuées nous laissent présager qu'il s'agit d'une première piste intéressante, en attendant le développement de mécanismes plus sophistiqués.

Il est licite de penser que le traitement équitable apporté par le *fair queuing* entre les flots permettra des estimateurs moins bruités et plus robustes.

Enfin, la comparaison d'une telle approche avec celles plus explicites telles que XCP [81] ou p-Fabric [5] nous semble intéressante à explorer. Il est possible que la complexité que ces solutions ajoutent ne soit pas fondamentalement nécessaire.

5.6 Conclusion

Les insuffisances de TCP Reno pour des liens à fort *Bandwith Delay Product*, et les contraintes fortes qu'il impose sur le dimensionnement des buffers nous conduisent à considérer l'introduction de nouveaux protocoles, plus adaptés à ces nouvelles conditions. Bien que conçus avec l'objectif d'être équitables avec TCP Reno, nous avons mis en évidence des problèmes d'équité inter- et intraprotocolaires, ainsi que les dégradations causées sur les flots *non-bottlenecked* en compétition, incluant des flots *streaming* (pertes, délais, gigue).

L'introduction de *fair queueing*, est proposée comme une alternative plus robustes aux différentes solutions d'AQM, permettant l'isolation des flots. Cette isolation permet d'une part de protéger les flux *streaming* à débit limité, leur assurant de faibles délais (grâce à la différenciation) et un taux de pertes négligeable, et d'autre part d'assurer l'équité entre les flux TCP en compétition pour la bande passante sur le lien. Nous rappelons que dans des conditions réalistes de trafic, le *fair queueing* est à la fois faisable et extensible puisque le nombre de flots à ordonnancer à chaque instant reste de l'ordre de quelques centaines indépendamment de la taille du lien. FQ impose une équité max-min, qui est nécessaire afin de pouvoir offrir des garanties prévisibles et robustes pour le trafic, sans nécessiter la collaboration des différentes sources de données.

Dans un contexte ou une allocation équitable est imposée par le réseau, il n'est plus nécessaire que les algorithmes TCP se restreignent à être équitables avec TCP Reno. Une évaluation de la performance des principales propositions TCP, notamment celles conçues pour des liens à fort *Bandwidth Delay Product* montrent qu'elles sont généralement insuffisantes pour exploiter la bande passante disponible en présence de petits buffers causant des pertes aléatoires (vraisemblable dans un contexte de *datacenters* ou de réseaux optiques, ...).

Grâce à la présence de *fair queueing* dans le réseau, il est possible aux sources de trafic d'utiliser un algorithme très simple suffisamment agressif basé sur AIMD ou MIMD pour obtenir une performance satisfaisante. L'étude suggère qu'un bon compromis entre utilisation et taux de pertes pourrait être atteint par l'utilisation d'un protocole TCP avec un incrément multiplicatif de la fenêtre de congestion comme dans Scalable TCP, utilisant du *pacing*, et un décrément basé sur une estimation de la bande passante disponible comme dans Westwood. La conception d'un algorithme optimal exploitant l'information implicitement transmise par le *fair queueing*, si tant est qu'il est nécessaire, reste un problème ouvert.

Enfin, dans un réseau optique, la réalisation d'un algorithme de *fair queueing* tel que nous l'avons proposé doit être reconsidérée, car il est techniquement difficile de réordonnancer les paquets. Il serait alors intéressant de considérer des solutions alternatives qui rejetteraient les paquets entrants en fonction de l'estimation du débit équitable, à la manière de la proposition *Approximate Fair Dropping* (AFD) [122]. Une piste d'amélioration serait de paramétrer la discipline de service au vu de notre compréhension du trafic, afin d'améliorer les propositions de règles empiriques proposées par les auteurs.

Chapitre 6

Un algorithme de MBAC pour Cross-Protect

Dans ce chapitre, nous nous penchons sur la problématique bien connue du contrôle d'admission et, en particulier, la réalisation d'un contrôle d'admission basé sur des mesures (MBAC, *Measurement Based Admission Control*). Malgré une recherche abondante sur de nombreuses années, il est raisonnable de dire qu'aucun algorithme satisfaisant n'a été proposé pour l'admission des flots à débit variable, même dans le cas favorable d'un multiplexage sans buffer ("bufferless multiplexing"). Le sujet est clairement passé de mode mais le contrôle d'admission demeure un composant essentiel au contrôle de la qualité de service et de nombreuses questions restent posées. Nous considérons ici la conception de l'algorithme de MBAC dans le contexte particulier de Cross-Protect, qui est nécessairement basé sur les informations limitées sur le trafic offertes par l'ordonnanceur.

Sommaire

6.1 Introduction 89
6.2 État de l'art des approches de MBAC 89
6.3 Implémentation et évaluation de l'algorithme de Grossglauser et Tse 93
6.4 Un algorithme de MBAC pour Cross-Protect 94
6.5 Évaluation des algorithmes 98
6.6 Conclusions 103
6.A Estimation de la variance sur un intervalle de taille quelconque 105

Contributions :

- Évaluation de l'applicabilité des algorithmes connus de MBAC au contexte de Cross-Protect ;
- Adaptation de l'algorithme de contrôle d'admission de Cross-Protect à un contexte plus réaliste de trafic et proposition de deux variantes ;
- Évaluation dans de nombreux scénarios de flots hétérogènes, gigués, ainsi qu'en régime non stationnaire de surcharge ;
- Étude de l'impact de l'intervalle d'échantillonnage des estimateurs.

L'implémentation des algorithmes développés et utilisés dans cette section ont été intégrés dans le module ns-2 Cross-Protect.

Publications et présentations :

Ce chapitre à fait l'objet de la publication suivante :

- Jordan Augé, Sara Oueslati, James Roberts, **Measurement-based admission control for flow-aware implicit service differentiation**, *23rd International Teletraffic Congress (ITC 2011), San Francisco (CA), Sep 6-8, 2011*.

En outre, un ensemble de résultats non présentés dans cette thèse ont fait l'objet des rapports techniques internes suivants :

- Sara Oueslati, Jordan Augé, Philippe Olivier, James Roberts, Raymond Cicutto, Jean-Louis Simon, **Premières analyses de la trace en surcharge IDF-La Réunion**, Rapport technique, Décembre 2005.
- Philippe Olivier, Jordan Augé, Sara Oueslati, **Étude préliminaire des caractéristiques du trafic IP sur un lien en surcharge**, Rapport Technique, Février 2006.

Ils décrivent la capture, l'analyse et le rejeu par simulation avec un contrôle d'admission d'une trace de trafic capturée sur un lien opérationnel en situation de surcharge, à la suite d'une panne de lien et d'un reroutage de son trafic.

6.1 Introduction

Le contrôle d'admission au sein de Cross-Protect, qui est un Contrôle d'Admission Basé sur des Mesures (MBAC, *Measurement-Based Admission Control*), joue le rôle essentiel de maintenir le *fair rate* suffisamment haut – afin de continuer à traiter les flots *streaming* en priorité et d'assurer un débit suffisant aux flots élastiques – et de conserver la charge de la file prioritaire en dessous d'un certain seuil. Il est soumis à la connaissance limitée du trafic dont nous disposons dans l'ordonnanceur. Il s'agit essentiellement du volume de paquets émis dans la file prioritaire lors des intervalles de temps successifs, ainsi qu'un estimateur du *fair rate* courant. Nous n'avons aucune connaissance *a priori* des caractéristiques du trafic et détectons seulement la terminaison des flots par l'absence de nouveaux paquets pendant un intervalle de temps donné (*timeout*).

Quand une proportion significative du trafic est élastique, et que les flots peuvent atteindre le *fair rate*, il est relativement facile de protéger la performance des flots en cours, simplement en rejetant les nouveaux flots lorsque le *fair rate* se retrouve en dessous d'un certain seuil (généralement de l'ordre de 1% de la capacité du lien [19] comme nous l'avons vu dans le Chapitre 3, Section 3.5.3, et rarement dépassé sauf en cas de surcharge importante). Nous considérons ici le cas plus problématique mais très probable où la grande majorité des flots possède un débit limité (par exemple à cause de leurs débits d'accès ADSL) et par conséquent se retrouve, dans des conditions de charge normale, traitée en priorité par l'ordonnanceur de Cross-Protect.

Bien que la connaissance des caractéristiques du trafic soit limitée, le contexte que nous considérons est très favorable à un multiplexage statistique efficace. Par hypothèse, le débit du lien est bien supérieur au débit crête maximum des flots devant recevoir un service prioritaire. Ce dernier est nécessairement inférieur au seuil sur le *fair rate* aux alentours de 1%, et même bien inférieur encore la plupart du temps (par exemple, en considérant des flots vidéo à 4Mb/s qui se partagent un lien de cœur de réseau OC196 à 10Gb/s). Il est ainsi possible de maintenir une forte utilisation des ressources tout en garantissant une performance excellente pour chaque flot.

Dans la suite de chapitre, nous discutons le choix d'un critère d'admission approprié pour préserver la qualité des flots *streaming* dont le débit crête est inférieur à un seuil dénoté $p < FR$. Nous proposons un algorithme simple et illustrons ses performances aux travers de nombreuses simulations, en régime de surcharge et dans des conditions de *flash crowds*.

6.2 État de l'art des approches de MBAC

De nombreux algorithmes de MBAC ont été présentés et étudiés au fil des années. Cependant, il n'en existe qu'un petit nombre qui se satisfait des hypothèses minimales sur le trafic qui conviennent dans notre contexte d'étude. Nous limitons cette section à ces contributions uniquement.

6.2.1 Measured sum

Jamin et al. [76] proposent un algorithme simple de MBAC appelé *Measured Sum* (MS). Un nouveau flot est admis si la somme de son débit nominal r et du débit estimé de l'agrégat des flots en cours $\widehat{v}$ est inférieure à un seuil d'utilisation u de la capacité du lien C, comme illustré par l'équation 6.1.

$$\widehat{v} + r \leq uC \tag{6.1}$$

La bande passante résiduelle constitue une marge de sécurité qui permet d'absorber les fluctuations du trafic. Le débit agrégé des flots est mesuré à l'aide d'un estimateur sur une fenêtre glissante de T slots de durée S ; Casetti *et al.* [43] proposent une version de l'algorithme basée sur un ajustement adaptatif de la longueur de cette fenêtre. $\widehat{v}$ est mis à jour à la fin de chaque slot, ainsi qu'à l'arrivée d'un nouveau flot, où est il augmenté du débit crête du flot :

$$\widehat{v} = \begin{cases} MAX(\widehat{v}^S), & \text{sur une fenêtre de T slots} \\ \widehat{v}^S, & \text{si } \widehat{v}^S > \widehat{v}, \text{ où } \widehat{v}^S \text{ est le} \\ & \text{débit mesuré sur une période S} \\ \widehat{v} + r & \text{à l'admission d'un nouveau flot} \end{cases}$$

Cet algorithme présente l'avantage d'être simple malgré la sensibilité aux paramètres S et T, et il ne requiert que la connaissance du débit crête des flots. La performance réalisée (taux de pertes) est cependant peu prévisible (au mieux la performance du pire cas, qui suppose un débit crête maximal afin de calculer le taux d'utilisation maximal cible).

6.2.2 Borne de Hoeffding

Le contrôle d'admission introduit par S. Floyd [55] calcule la bande passante équivalente d'un ensemble de flots (une estimation de la bande passante agrégée) en utilisant la borne de Hoeffding (un résultat de la théorie des probabilités qui donne une borne supérieure à la probabilité que la somme de variables aléatoires dépasse un certain seuil).

Cette bande passante équivalente dépend du choix d'une probabilité de perte cible et s'exprime ainsi :

$$\hat{C_H}(\hat{\mu}, \{p_i\}_{1 \leq i \leq n}, \epsilon) = \hat{\mu} + \sqrt{\frac{\ln 1/\epsilon \sum_{i=1}^{n} (p_i)^2}{2}}$$

$\hat{\mu}$ est un lissage exponentiel du débit d'entrée agrégé, ϵ est la probabilité de perte cible, p_i le débit crête du flot i, et n le nombre de flots en cours.

Un nouveau flot est admis si la somme de son débit crête et de la mesure de la bande passante équivalente est inférieure à la capacité du lien :

$$\hat{C_H}(\hat{\mu}, \{p_i\}_{1 \leq i \leq n}, \epsilon) + p_{n+1} \leq C$$

Comme pour l'algorithme MS, l'estimation du débit entrant agrégé $\hat{\mu}$ est augmentée à chaque addition du débit crête du nouveau flot.

L'algorithme a l'avantage d'intégrer explicitement la probabilité de perte cible dans les conditions d'admission. La borne de Hoeffding n'est cependant pas une borne forte, ce qui conduit à une performance conservatrice de l'algorithme, c'est-à-dire une utilisation faible. La méthode suppose également la connaissance du débit crête individuel de chaque flot, et encore plus difficile, le nombre de flots en cours. De plus les calculs de bande passante équivalente sont basés sur des flots à débit constant, ce qui est loin d'être le cas en réalité.

6.2.3 Calcul d'enveloppes de trafic

La méthode de Qiu et Knightly [127] utilise des mesures de l'enveloppe maximale d'un agrégat de trafic. La condition d'admission est calculée à l'aide de la moyenne et de la variance de ces enveloppes, ainsi que d'une probabilité de perte cible ; un nouveau flot de débit crête p est admis si :

$$\overline{R}_k + p + \alpha\sigma_k \leq C, \forall k = 1, 2, .., T$$

où $\overline{R}_k$ est l'estimation du débit crête entrant agrégé, mesuré pour différentes échelles de temps – k variant de 1 à T slots dans la fenêtre courante (de T slots) –, et σ_k est la variance de ces mêmes mesures sur les M dernières fenêtres. $\alpha = Q^{-1}(p_q)$, avec p_q la probabilité de perte cible et $Q(.)$ la fonction de distribution complémentaire d'une variable aléatoire Gaussienne $N(0,1)$.

Un niveau de confiance est également calculé afin de pallier aux incertitudes des mesures. L'enveloppe est utilisée afin de capturer la variabilité du trafic à différentes échelles de temps. Toutefois, ils utilisent un modèle avec buffer, qui le rend dépendant de caractéristiques détaillées du trafic. Le processus de mesure est de plus relativement complexe à cause des nombreuses échelles de temps nécessaires.

6.2.4 Approche basée sur la théorie de la décision

Gibbens et al. [62] proposent une approche basée sur la théorie de la décision, où un nouveau flot est admis si la charge agrégée courante est inférieure à un seuil approprié. Parmi les méthodes proposées, la plus simple ne prend en compte que le débit crête d'un flot.

Cela correspond exactement à nos besoins et les résultats confirment qu'un multiplexage efficace est possible dans un tel cadre d'étude. Malheureusement, le calcul du seuil nécessite des hypothèses sur le niveau de charge offerte, ainsi que sur la *burstiness* des flots (le degré de rafales de paquets).

6.2.5 Théorie de la bande passante équivalente

Gibbens et Kelly [61] présentent une famille d'algorithmes basées sur des tangentes à une courbe de bande passante équivalente, calculée à partir de la borne de Chernoff (qui améliore la borne de Hoeffding).

Ici, différents choix de tangentes impliquent la connaissance de différentes caractéristiques du trafic. Les flots sont supposés appartenir à un nombre donné de classes. L'approche la plus simple nécessite la connaissance de la charge globale instantanée (comme dans [62]) ainsi que le nombre de flots de chaque classe accompagné de leur débit crête. Cette méthode s'apparente à l'utilisation de la borne de Hoeffding proposée par Floyd [55].

La méthode de la tangente au débit crête admet un nouveau flot si :

$$np(1-e^{-sp})+e^{-sp}\widehat{v}\leq C$$

où n est le nombre de flots admis, p le débit crête, s le paramètre spatial de la borne de Chernoff, $\widehat{v}$ l'estimation de la charge globale et C la capacité du lien.

La complexité de ce genre d'algorithmes (basés sur des bornes de Chernoff) est trop importante pour qu'ils puissent être considérés pour la prise de décisions d'admissions en temps réel. De plus, les alternatives identifiées dans [61] requièrent toutes des informations supplémentaires qui ne sont pas disponibles dans Cross-Protect.

6.2.6 Stratégies de *back off*

Les méthodes précédentes, proposées par Gibbens *et al.* [62], Gibbens et Kelly [61] et Floyd [55] proposent toutes d'utiliser une stratégie de *back off* : lorsqu'un flot est bloqué, aucun autre flot n'est accepté tant que l'un des flots en cours ne termine pas. Cela évite le problème d'un grand nombre d'admissions dans des conditions de forte charge, résultant d'une seule mesure faible. Malheureusement un tel procédé n'est pas possible avec Cross-Protect puisque les terminaisons de flots ne sont pas explicitement connues.

6.2.7 Décomposition selon différentes échelles de temps

La notion d'échelle de temps critique

L'algorithme proposé par Grossglauser et Tse [65, 66] se base sur l'idée de décomposer les variations du débit agrégé selon deux échelles de temps : des fluctuations à court et à long termes. La durée critique séparant les deux échelles de temps est :

$$\tilde{T}_h := T_h/\sqrt{n}$$

où T_h est la durée moyenne des flots, n est l'estimation du nombre maximal de flots en cours et qui peut être écrit : $n=\sqrt{\frac{C}{\mu}}$, où μ est le débit moyen des flots. D'après les auteurs, les fluctuations du débit agrégé qui se produisent à une échelle de temps plus grande que $\tilde{T}_h$ sont automatiquement compensées par les arrivées et les départs de flots. Par conséquent, il suffit de réserver de la bande passante uniquement pour absorber les fluctuations à court terme, qui sont dues aux variations de débit intrinsèques aux flots en cours.

Une condition d'admission est déduite afin de satisfaire une probabilité de perte cible, sous les hypothèses du multiplexage sans buffer.

Leur première proposition suppose que les caractéristiques individuelles des flots, leur débit moyen et leur variance, sont mesurées. Un nouveau flot est admis si la condition suivante est satisfaite :

$$C-(N_t+1)\widehat{\mu}_t>\alpha_q\widehat{\sigma}_t^H\sqrt{N_t+1} \tag{6.2}$$

où C est la capacité du lien, N_t le nombre de flots dans le système au temps t, $\widehat{\mu}_t$ le débit moyen mesuré des flots, $\alpha_q=Q^{-1}(\epsilon)$, où ϵ est la probabilité de perte cible, $Q(.)$ la cdf complémentaire d'une variable aléatoire Gaussienne $N(0,1)$, et $\widehat{\sigma}_t^H$ l'écart type estimé du débit par flot.

Si S_t est le débit entrant agrégé au temps t, alors $\hat{\mu}_t$ est donné par :

$$\hat{\mu}_t=\int_0^\infty \frac{S_{t-\tau}}{N_{t-\tau}}g_\tau d\tau$$

où g_t est un filtre passe-bas, de fréquence de coupure $1/\tilde{T}_h$; et $\widehat{\sigma}_t^H$, l'estimation de la variance des hautes fréquences, est donnée par :

$$\hat{\sigma}_t^H=\left[\int_0^\infty\left[\frac{S_{t-\tau}^H}{N_{t-\tau}}-\int_0^\infty\frac{S_{t-u}^H}{N_{t-u}}h_u du\right]^2 h_\tau d\tau\right]^{1/2}$$

Nous ne pouvons pas utiliser cette méthode notamment parce que le nombre de flots en cours N_t n'est pas connu dans Cross-Protect.

Le deuxième article fournit une méthode qui est plus proche de nos hypothèses, lorsque N_t n'est pas connu, puisqu'elle utilise des mesures du débit agrégé en entrée, ainsi que le débit crête des flots admis. Leur méthode se base sur une approximation Gaussienne du débit agrégé. Un nouveau flot est admis si :

$$C - A_t^\lambda - p > \alpha_q \widehat{\sigma}_t^{AH} \tag{6.3}$$

où A_t^λ est la mesure agrégée du débit, p est le débit crête des flots et $\widehat{\sigma}_t^{AH}$ l'écart type du débit agrégé.

Pour obtenir cette formule , les variables A_L^t et A_H^t sont définies et correspondent à l'application de filtrages, respectivement passe-bas et passe-haut, aux fluctuations du débit entrant agrégé S_t :

$$\begin{aligned} A_t^L &= \int_0^\infty S_{t-\tau} g_\tau d\tau \\ A_t^H &= S_t - A_t^L \end{aligned}$$

où g_t est un filtre passe-bas.

Une autre variable $\hat{\sigma}_t^{AH}$ est introduite et correspond à l'estimation de la variance de la composante haute agrégée A_t^H :

$$\hat{\sigma}_t^{AH} = \left[\int_0^\infty \left[A_{t-\tau}^H - \int_0^\infty A_{t-u}^H h_u du \right]^2 h_\tau d\tau \right]^{1/2}$$

où h_t est également un filtre passe-bas.

L'estimateur passe-bas corrigé A_t^λ est dérivé ainsi :

$$A_t^\lambda = A_t^L + \lambda_t * g_t$$

le facteur de correction λ_t étant :

$$\lambda_t = \sum_i r\delta(t - t_i)$$

où t_i est la date d'admission du flot i, r son débit crête et δ la distribution de Dirac. Autrement dit, lorsqu'un flot est admis, son débit crête est ajouté à A_t^λ. Cela permet d'éviter des situations de surcharges momentanées à forte charge (leur hypothèse de trafic) dues aux admissions suivant un estimateur A_t^λ faible pendant une période t. Les instants d'arrivées de flots peuvent n'être connus que partiellement dans Cross-Protect puisque le contrôle d'admission est réalisé de manière distribuée sur plusieurs *line cards*.

Si l'on suppose que le débit agrégé en entrée ne change qu'en des instants discrets, alors A_t^λ peut être exprimé ainsi :

$$A_{t_i}^\lambda = \phi_i A_{t_{i-1}}^\lambda + (1 - \phi_i) S_{t_{i-1}} + r.\mathbf{1}_{\{\text{admission de flot à } t_i\}}$$

où t_i est l'instant où la bande passante agrégée S_t change ou bien qu'un nouveau flot est admis et

$$\phi_i = \exp\left(\frac{t_{i-1} - t_i}{\tilde{T}_h} \right)$$

.

La méthode proposée par Grossglauser et Tse [66, 65] est la plus proche de nos hypothèses sur le trafic. Elle possède notamment l'avantage, par rapport par exemple à [66], de ne réserver de la bande passante que pour les fluctuations du trafic à court terme, puisque celles à long terme peuvent être compensées par les arrivées et les départs de flots. Elle permet ainsi de parvenir à une meilleure utilisation du lien à condition de savoir déterminer l'échelle de temps critique $\tilde{T}_h$. Son calcul nécessite la connaissance de la durée moyenne ainsi que du débit moyen des flots, information qui n'est généralement pas disponible. Toutefois il est possible de contourner cet obstacle en mesurant par exemple $\tilde{T}_h$ en fonction des données passées. Nous pouvons alors profiter des nombreux avantages de cette méthode.

Knightly *et al.* utilisent une estimation classique non-biaisée, basée sur les données mesurées sur les M dernières fenêtres temporelles afin de prédire le trafic, tandis que Grossglauser et Tse utilisent un filtrage passe-bas à la fréquence de coupure $\tilde{T}_h^{-1}$ pour estimer $\hat{\mu}$, et à $T_s = k\tilde{T}_h^{-1}$ pour $\hat{\sigma}$. Cela permet une plus forte réactivité aux changement dans le profil de trafic et apparaît un choix plus raisonnable dans la mesure où les données récentes ont plus d'impact sur l'estimation du débit entrant que les plus vieilles.

6.2.8 Discussion

Malgré de grandes différences entre les algorithmes de MBAC, il s'avèrent qu'ils semblent tous conduire au même compromis entre le taux d'utilisation du lien et la performance perçue au niveau flot. Ceci est illustré dans [38] et [21] où la performance est mesurée en termes de taux de pertes de paquets. Les algorithmes diffèrent au niveau de la prévisibilité de leur performance : le choix de la valeur des paramètres pour la condition d'admission permettant d'obtenir un taux de performance cible. En fait Breslau et al. [38] montrent que tous les algorithmes qu'ils ont testés sont très peu efficaces à prédire la performance qui dépend de manière significative de caractéristiques du trafic qui ne sont pas explicitement prises en compte. Des raisons pour lesquelles il est extrêmement difficile de contrôler des indicateurs de performances tels que le taux de pertes de paquets au moyen d'algorithmes de MBAC sont évoqués par Bean [13]. Dans nos travaux, nous espérons ainsi aboutir à des algorithmes raisonnablement efficaces et c'est la limite de notre ambition.

6.3 Implémentation et évaluation de l'algorithme de Grossglauser et Tse

Une version discrétisée de l'algorithme de Grossglauser et Tse a été réalisée dans le simulateur `ns-2`. Nous la notons (`GT`). Les résultats obtenus avec cet algorithme nous serviront de référence lors de l'évaluation réalisée dans la section 6.5.

A la fin du (n+1)-ième intervalle, les nouvelles valeurs de la moyenne et de la déviation standard sont calculées ainsi :

$$\begin{aligned} A^L_{n+1} &= \alpha A^L_n + (1-\alpha) S_{n+1} \\ A^H_{n+1} &= S_{n+1} - A^L_{n+1} \\ D_{n+1} &= \beta D_n + (1-\beta)(A^H_{n+1})^2 \\ E_{n+1} &= \beta E_n + (1-\beta) A^H_{n+1} \\ \sigma_{n+1} &= (D_{n+1} - E^2_{n+1})^{1/2} \\ A^\lambda_{n+1} &= A^L_{n+1} \end{aligned}$$

A tout moment dans le (n+2)-ième intervalle, un flot peut être admis si :

$$A^\lambda_{n+1} + r < c - \alpha_q \sigma_{n+1}$$

Après une admission,

$$A^\lambda_{n+1} += r$$

Les paramètres de lissage des filtres passe-haut et passe-bas sont donnés par :

$$\begin{aligned} \alpha &= \exp\left(-\frac{\tau}{\tilde{T}_h}\right) \\ \beta &= \exp\left(-\frac{\tau}{T_s}\right) \end{aligned}$$

où T_s est environ 5 à 10 fois plus long que $\tilde{T}_h$.

Il convient de noter que λ_n est calculé à la fin du n-ième intervalle et est augmenté du débit crête des flots dans le $(n+1)$-ième comme suit :

$$A^\lambda_n = A^L_n + \lambda_n$$

La condition d'admission reste inchangée après l'admission d'un flot dans le $(n+1)$-ième :

$$A^\lambda_n += r$$

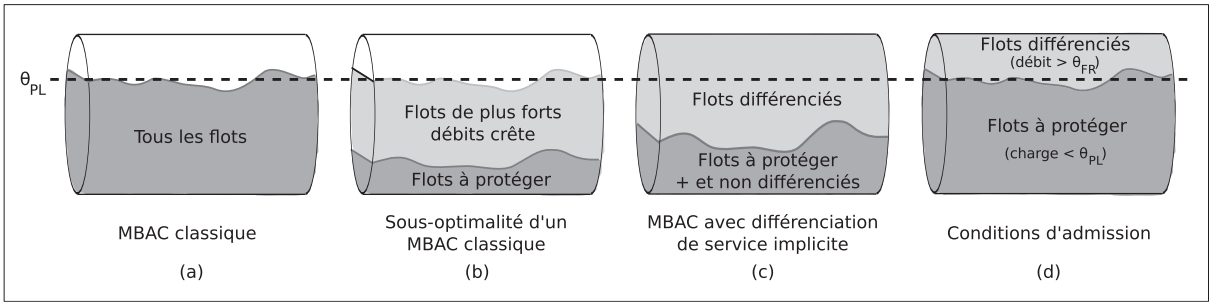


FIGURE 6.1 – Illustration de l'algorithme de MBAC avec divers régimes d'utilisation

6.4 Un algorithme de MBAC pour Cross-Protect

6.4.1 Critères d'admission

L'admission des flots dans Cross-Protect est déterminée par la valeur de deux mesures de congestion, le *fair rate* FR et le *priority load* PL [91]. Ces valeurs sont calculées par lissage exponentiel des valeurs mesurées sur des intervalles successifs de durée τ, et l'admissibilité dans l'intervalle t est déterminée par les valeurs calculées à $t-1$.

FR correspond au débit qu'un flot aurait en sortie s'il avait constamment des paquets à envoyer, et est estimé à partir des données disponibles dans l'ordonnanceur. Il est moyenné sur un intervalle en rapport avec l'échelle de temps des arrivées et des départs de flots. PL est la charge moyenne du trafic qui arrive dans la file prioritaire pendant chaque intervalle de temps. Les paquets envoyés dans la file prioritaire sont ceux qui n'appartiennent à aucun flot goulotté.

Quand une grande partie du trafic provient de flots élastiques qui ne sont nulle part limités en débit, le critère d'admission le plus significatif est FR. Le *priority load* reste bas de telle sorte que les délais et taux de perte des flots *streaming* de débit crête globalement inférieur au FR sont négligeables [91, 92]. Le contrôle d'admission est plutôt simple dans ce cas puisque les flots élastiques sont naturellement tolérants à une imprécision sur l'estimation du *fair rate*. La charge des flots *streaming* dans l'architecture Cross-Protect est ainsi initialement limitée à un seuil voisin de 70 ou 80%.

En pratique cependant, la grande majorité des flots élastiques possède un débit crête limité et, lorsqu'il est inférieur au seuil sur le *fair rate*, ceux-ci sont traités dans la file prioritaire conjointement aux flots *streaming*. Il s'agit par exemple de scénarios ADSL où les flots sont limités par leur débit d'accès. FR n'est alors plus critique tandis que le *priority load* mesuré PL inclue notamment des flots de débit supérieur à p que nous ne souhaitons pas protéger.

La difficulté pour le contrôle d'admission consiste à pouvoir différencier les flots en fonction de leur débit crête, sans avoir à les mesurer explicitement. Nous supposons que les flots de débit crête inférieur ou égal à p subiront des pertes et des délais négligeables si le trafic envoyé dans la file prioritaire de Cross-Protect reste inférieur à la capacité C du lien avec une probabilité d'au moins $1-\epsilon$. La valeur ϵ peut-être calibrée afin d'assurer des pertes et délais suffisamment bas.

6.4.2 Moyenne et variance du *priority load*

Notons $b(t)$ la mesure en bit/s de la charge envoyée dans la file prioritaire dans le slot t. Le *priority load* $B(t)$ est la moyenne glissante :

$$B_t = (1-\beta) \times B_{t-1} + \beta \times b_t. \tag{6.4}$$

où le paramètre $\beta = 1 - \tau/\tilde{T}_h$ et $\tilde{T}_h$ est l'échelle de temps critique définie dans [66].

Nous suivons l'approche des auteurs et proposons d'estimer la variance comme suit :

$$\widehat{\sigma}_t^2 = D_t - E_t^2 \text{ où} \tag{6.5}$$
$$D_t = (1-\gamma)D_{t-1} + \gamma(b_t - B_t)^2, \tag{6.6}$$
$$E_t = (1-\gamma)E_{t-1} + \gamma(b_t - B_t). \tag{6.7}$$

En suivant [66], le paramètre de lissage γ est choisi égal à $1-(1-\beta)/10$, et des mesures de $\tilde{T}_h$ peuvent être produites périodiquement (déduites lors des *timeouts*). En fait, l'estimation de cette dernière valeur reste problématique puisque la distribution de la durée des flots n'est pas exponentielle comme considérée dans [66] mais à queue lourde. Heureusement, il apparait de nos résultats non

présentés ici, ainsi que dans des travaux précédents, que le choix des paramètres β et γ n'est pas extrêmement critique.

L'utilisation de la condition d'admission (6.3) sur ces estimateurs est trop conservatrice pour les flots de débit crête supérieur à p. Nous supposons que de tels flots sont capables d'ajuster leur débit au *fair rate* en cas de surcharge (ou le devraient), et ne nécessitent pas d'être protégés. Par exemple, dans la figure 6.1b), les mesures du *fair rate* et du *priority load* sont les mêmes que dans la figure 6.1a), mais nous pourrions accepter plus de flots sans violer nos contraintes de délai pour les flots protégés. L'ajustement du seuil d'admission en tenant compte de la variance de l'ensemble des flots chercherait à maintenir la charge en dessous du seuil θ_{PL} et empêcherait le système d'entrer dans l'état favorable représenté dans la figure 6.1c). Dans cet état, les flots de débit supérieur à p deviennent *backlogged* et ne contribuent plus au *priority load*. Le dessin de la figure 6.1d) présente une situation idéale ou le contrôle d'admission différencie parfaitement les flots de débit crête inférieur et supérieur à p, et la composition du trafic est telle qu'elle permet au lien d'être saturé.

Nous voyons ici en quoi le contrôle d'admission de Cross-Protect diffère des algorithmes classiques qui opèrent en régime transparent. Ici, la saturation du lien est possible et même recommandée afin que les flots que nous ne souhaitons pas protéger soient différenciés.

6.4.3 Approximation basée sur une hypothèse Poissonnienne

Supposons que le nombre de flots inélastiques de débit crête inférieur ou égal à p en cours dans un intervalle donné a une distribution de Poisson. Ce serait le cas en l'absence de blocage, et dans le contexte du modèle de sessions Poisson (voir Chapitre 3). Si les paquets sont de taille constante maximale L, le nombre de paquets qui arrivent dans un intervalle plus petit que L/p possède également une distribution de Poisson [1]. Cela suggère qu'il est possible d'estimer la variance du débit dans un intervalle à partir de sa moyenne, ce qui permet de proposer le MBAC très simple qui suit, dénoté Poisson :

- Nous choisissons un intervalle de durée $\tau = L/p$.
- Étant donné la charge prioritaire mesurée B_t bits/s, $m_t = B_t\tau/L$ est une estimation du nombre de paquets, et nous déduisons l'estimation de la variance $\widehat{\sigma}_t^2 = m_t L^2/\tau^2 = B_t p$.
- Cette estimation peut alors être appliquée à la condition d'admission 6.3.

Performance du multiplexage statistique

En ignorant le blocage des flots et le fait que certains flots émettent des paquets de taille inférieure à L, l'approximation Poisson conduit à des décisions d'admission conservatrices. Cependant, p représentant un faible pourcentage de la capacité C, l'approximation permet une forte utilisation des ressources du lien. Le tableau 6.1 donne le seuil sur la charge du lien correspondant à des valeurs particulières de C/p et ϵ. Le contrôle d'admission serait appliqué dans l'intervalle t lorsque B_{t-1} dépasse ce seuil.

C/p	100	100	1000	1000
ϵ	0.001	0.01	0.001	0.01
α_q	3.09	2.33	3.09	2.33
Seuil	0.73	0.79	0.91	0.93

TABLE 6.1 – Seuils d'admission

Notons que si les flots de débit crête supérieur à p sont inclus dans l'estimation de la charge prioritaire B_t, la variance estimée par 6.5 sera plus grande que l'estimation Poisson $B_t p$. Le MBAC dérivé de [66] sera trop conservateur, préservant des états tels qu'en figure 6.1b). Nous espérons l'estimateur Poisson mieux capable de permettre des transitions vers des états tels qu'en figure 6.1c et d).

Le seul cas problématique pour la différenciation des flots est lorsque l'on a un mélange de flots qui ont en partie un débit inférieur à p et en partie supérieur. Toutefois, la contribution de ces flots en raison du carré de leur débit crête assurera une différenciation dans la majorité des cas.

Supposons que le débit crête des flots est $p = 100$kb/s, sur un lien de capacité $C = 10$Mb/s. Nous normalisons ces valeurs afin d'obtenir une capacité unitaire ($p = 10^{-2}$). Afin d'assurer une probabilité de perte $\epsilon = 10^{-2}$, nous avons vu que la charge du lien devait être limitée à une valeur de $\theta = 0.79$ telle que : $\theta + \alpha_q\sigma = 1$ avec $\sigma = \sqrt{\theta p}$.

1. Un flot de débit r contribue de manière indépendante 1 paquet avec la probabilité r/p.

Supposons maintenant qu'avec les mêmes paramètres, à la même charge, le trafic soit un mélange de 2 classes de flots, respectivement $P_1 = 50$ et $P_2 = 200$kb/s. Chacune contribue à la moitié de la charge générée.

Le calcul de la variance du trafic donne alors : $V = \frac{\theta}{2P_1} P_1^2 + \frac{\theta}{2P_2} P_2^2$, soit $\sigma' > \sigma$ et ainsi une probabilité de débordement supérieure qui se traduira par une différenciation des classes de flots avec une forte probabilité.

Intégration de la variance

Si le trafic global contient de nombreux flots de débit crête inférieur à p, l'approximation Poissonnienne peut être trop conservatrice. C'est pourquoi nous proposons un algorithme plus raffiné, dénoté (MinVar) pour *Minimum de Variance*, où l'estimation de la variance est le minimum de $B_t p$ et de celle calculée par (6.5). Les algorithmes (Poisson) (MinVar) sont évalués pour différents scénarios de test dans la section 6.5 ci-dessous.

6.4.4 Limite du nombre d'arrivées par intervalle

Le contrôle d'admission est particulièrement utile lors d'événements exceptionnels quand la demande dépasse considérablement la capacité du lien. Cela se produit en particulier lorsque une panne induit un reroutage du trafic sur le lien considéré. Le trafic va croître jusqu'à atteindre une nouvelle charge stationnaire. Le MBAC doit être capable de rejeter l'excédent de trafic dans ce régime afin de préserver les objectifs de performance. Cependant, l'impact majeur des pannes apparaît pour les flots en cours sur le lien coupé puisqu'ils apparaissent tous soudainement comme des nouveaux flots sur les liens de secours. Un lien sur ce chemin ne sera généralement pas congestionné avant la panne et donc disposé à accepter de nouveaux flots. Si toutefois tous les "nouveaux" flots qui arrivent dans les quelques intervalles de temps suivant la panne sont acceptés, le lien deviendra immédiatement fortement congestionné.

Pour prévenir cette situation, nous limitons le nombre de flots acceptés dans un intervalle de temps, comme dans [66]. Étant donnés les estimateurs A_t et $\widehat{\sigma}_t^2$, nous acceptons un maximum de n nouveaux flots où n est le plus petit entier tel que $(n+1)p + A_t + \alpha_q \widehat{\sigma}_t > C$. Il se peut que cela ne suffise pas vu le temps nécessaire avant que le trafic issu des nouveaux flots soit pris en compte dans l'estimation de la charge A_t. Il convient également de remarquer que, puisque les durées des flots ont généralement une distribution à queue lourde (*heavy-tailed*), l'espérance de la durée *résiduelle* des flots reroutés sera supérieure à la moyenne. L'impact des mauvaises décisions d'admission est ainsi plus sévère pour ces flots que pour ceux réellement nouveaux.

La stratégie de *back off* proposée dans [62] où, lorsqu'un flot est bloqué, aucun autre flot n'est accepté jusqu'à ce que l'un des flots en cours se termine, n'est pas applicable dans notre système puisque les terminaisons de flots ne sont pas explicites. La solution envisagée est d'interrompre la protection d'un nombre suffisant de flots en cours pour prévenir la congestion en remarquant que l'interruption des flots est dans tous les cas nécessaire quand la combinaison du trafic sur le lien en panne et le lien de secours dépasse la capacité disponible.

6.4.5 Instabilité du *priority load* mesuré

Le *priority load* peut varier abruptement lorsqu'un ensemble de flots de même débit P deviennent *backlogged* et qu'ils ne sont plus pris en compte dans sa mesure. Lorsque P est voisin du seuil p, il est possible qu'une situation anormale se produise où B_t est inférieur au seuil critique et la moyenne (à long terme) du *fair rate* reste au-dessus du seuil minimum, malgré le fait que des flots de débit p deviennent *backlogged*. Le lien sera alors ouvert à de nouvelles admissions. Ce phénomène sera d'autant plus marqué que le nombre de classes de débit est faible, puisqu'un grand nombre de flots sera rapidement sujet au basculement.

Afin de limiter l'impact de ce phénomène, nous imposons une valeur pour le *priority load* égale à $C\tau$ dans tous les intervalles ou un flot de débit crête p pourrait être *backlogged*. Cette dernière condition peut-être facilement déduite des paramètres de l'algorithme PFQ présenté dans le Chapitre 3. Il s'agit d'une mesure instantanée du *fair rate*, tandis que FR en est une moyenne sur le long terme.

Il n'est pas possible d'anticiper ces situations puisque dans Cross-Protect le *fair rate* instantané, qui cause le basculement, n'est connu qu'au moment où ce dernier se produit. Le contrôle exercé sur le *priority load* PL permet de limiter l'occurrence de tels instants, il demeure essentiel car l'indicateur sur le *fair rate* instantané seul ne serait pas stable (indication binaire).

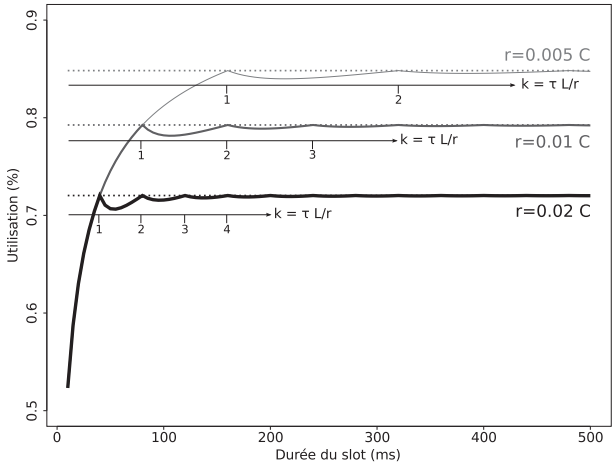


FIGURE 6.2 – Valeur du *priority load* mesuré en fonction de la taille de l'intervalle, pour différentes valeurs de p. La valeur réelle est indiquée par les lignes en pointillés.

Cette modification se montre également utile lorsque les flots ont un débit crête nominal p mais subissent de la gigue ; ces derniers se retrouvent momentanément *backlogged* quand l'intervalle entre deux arrivées de paquets est trop faible (voir la section 6.5.6).

6.4.6 Influence de la taille du slot sur les mesures

Le choix de l'intervalle de discrétisation sera généralement guidé soit par des contraintes techniques[2], soit avec l'objectif de mesurer un débit crête donné pour les flots. Nous considérons des intervalles de longueur kL/p pour $k \geq 1$. De plus petites valeurs de k ne permettent pas de prendre en compte la faible variance des flots de débit inférieur à p. D'autre part, de trop grandes valeurs de k ont tendance à rendre le MBAC moins réactif à des changements soudains tels que ceux considérés dans les scénarios de *flash-crowd*. Nous remarquons que les grandes valeurs de k n'apportent qu'une amélioration mineure de la performance. C'est pourquoi une valeur de quelques unités semble un bon compromis. Nous évaluerons l'impact de k dans la section 6.5.

Valeurs de k et estimation de la variance

Afin de comprendre l'impact du choix de cet intervalle sur l'estimation de la variance, nous considérons un flot émettant des paquets de taille L à débit constant sur un intervalle τ quelconque. Les calculs détaillés dans un annexe à ce chapitre donnent :

$$V_t = \frac{B_t}{p}\left[\frac{\Delta_t^2 L^2}{\tau^2} + (\frac{\tau p}{L} - \Delta_t)(1 + 2\Delta_t)\frac{L^2}{\tau^2}\right] \tag{6.8}$$

avec $\Delta_t = \lfloor \frac{\tau p}{L} \rfloor$

Puisque la variance est monotone en fonction de la charge, une itération jusqu'au point fixe permet de déterminer le seuil optimal de *priority load* pour ϵ donné. La figure 6.2 présente ces valeurs, pour $\epsilon = 10^{-2}$, et $p = 50$, 100 et 200kb/s.

Pour $k = 1$, la variance est celle du cas Poisson, qui permet d'estimer convenablement la variance et donc d'obtenir une utilisation optimale. La performance est sensiblement la même pour $k > 1$, et reste la même pour les valeurs entières de k. Par contre, pour $k < 1$, la variance est surestimée et coïncide avec la variance d'un flot de débit plus important. C'est-à-dire que si $\tau = L/s$ avec $r < s < p$, alors la performance atteinte sera celle correspondant au débit s et non pas r. La moyenne quant à elle n'est pas affectée. Par exemple, pour $p = 100$kb/s, une valeur de $k = 4$ permettra de mesurer correctement la variance de flots ayant un débit aussi faible que $r = 25$kb/s, et donc d'obtenir la performance correspondante. En ajustant le seuil d'admission à une valeur plus optimale, on s'attend à ce que le MBAC (MinVar) assure une meilleure différenciation des flots lorsque k est suffisamment grand.

2. C'est le cas par exemple du noyau Linux qui selon les versions présente une résolution minimale par défaut de 1 ou 10ms

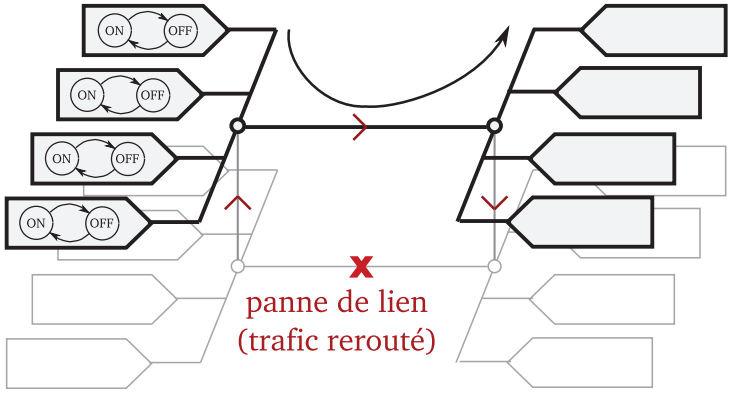


FIGURE 6.3 – Topologie utilisée pour la simulation : la partie représentée en trait gras est utilisée pour les simulations en régime stationnaire ; l'ensemble de la topologie sera utilisée pour simuler un scénario de *flash crowd* dans la section 6.5.7

.

6.5 Évaluation des algorithmes

Nous évaluons les propositions de MBAC (Poisson) et (MinVar) à l'aide d'un grand nombre de simulations, et nous comparons les résultats avec l'algorithme (GT) que nous avons présenté plus haut.
.

6.5.1 Environnement de simulation

La topologie pour la simulation est la topologie classique *dumbbell* avec un lien central de capacité $C = 10\text{Mb/s}$. Le choix délibéré d'une faible valeur de C permet d'obtenir des temps de simulation suffisamment courts, puisque la performance ne dépend que du ratio C/p (confirmé par simulations).

Le trafic est composé de flots UDP dont la durée est exponentiellement distribuée de moyenne $T_h = 60\text{s}$, et qui arrivent selon un processus de Poisson. Les flots génèrent leurs paquets selon un processus *on-off* assez général, de débit crête constant pendant la période *on* (de 50 à 300 kb/s) ; chaque période a une durée exponentielle de moyenne 500ms (sauf indication contraire). La taille des paquets est constante et fixée à 1000 octets. Les simulations sont lancées pendant 2000s plusieurs fois (25), et nous ne conservons que le régime stationnaire (en éliminant les 200 premières secondes). La probabilité de débordement cible ϵ est fixée à .01. Nous simulons une charge stationnaire égale à 100, 120 et 140% de la capacité du lien. Sauf indiqué, le taux d'arrivée des flots est tel que la limite d'admission par slot n'opère pas. L'intervalle $\tau = kL/p$ est choisi avec $k = 1$ ou une faible valeur indiquée. Nous notons qu'une valeur de p de l'ordre du seuil sur le *fair rate*, généralement choisi égal à 1% de C, correspondra à un pire cas pour la performance (voir tableau 6.1).

6.5.2 Critères de performance

La mesure de la performance des flots *streaming* et élastique nécessite de sélectionner des métriques qui reflètent la différentiation effectuée par notre ordonnanceur. La probabilité de débordement (notée *ov* pour *overflow*) représente la proportion d'intervalles où la charge prioritaire instantanée est supérieure à C. Elle est représentative de la performance des flots *streaming*, à condition que ceux-ci soient correctement envoyés dans la file prioritaire, ce qui est mesuré par la probabilité de *backlog* (*bk*, mesurée pour chaque classe de débit).

Nous mesurons également la probabilité de blocage (*bl*), qui représente la proportion de flots qui ne sont pas admis, ainsi que le taux de pertes de paquets (*lo* pour *losses*), la proportion de paquets qui sont perdus à cause de débordements du buffer. La combinaison des deux métriques montre comment l'excédent de trafic est éliminé pendant les instants de congestion. Un MBAC efficace devrait permettre une valeur de *bl* aussi faible que possible tout en garantissant la qualité de service des deux classes de flots. Nous détectons également le nombre de fois que chaque critère d'admission est responsable du blocage (bl_{PL} et bl_{FR}). Enfin, nous mesurons l'utilisation du lien (*ut*) qui, si elle dépend de la composition du trafic, nous donne cependant une indication de comment les ressources du lien sont exploitées.

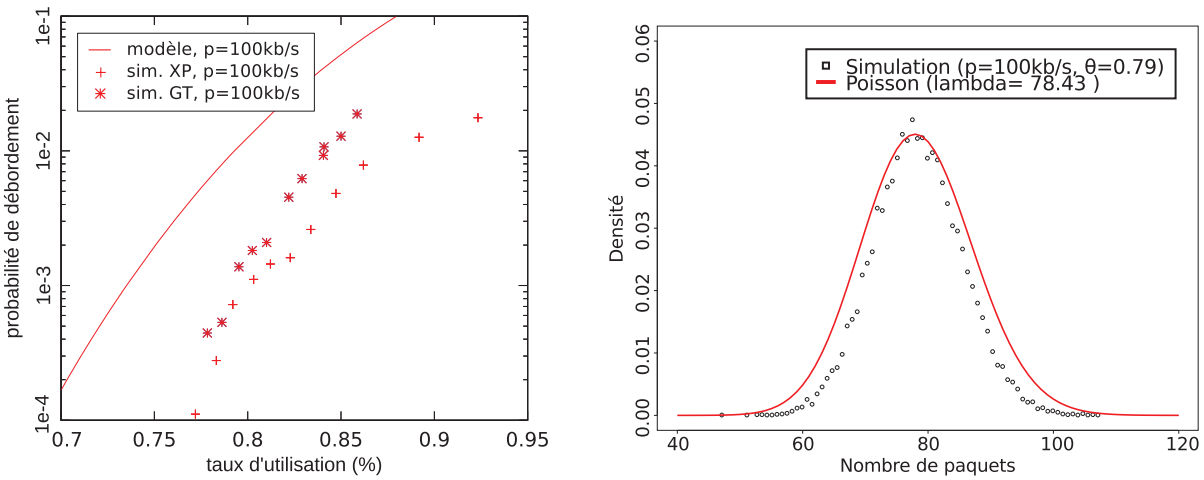


FIGURE 6.4 – A gauche : utilisation en fonction de la probabilité de débordement pour les MBAC XP et GT, comparés un cas où la charge par intervalle suit une distribution de Poisson, avec $p = 100\text{kb/s}$ – A droite : densité de charge reçue par intervalle pour $\rho = 1.20$, soumise au contrôle d'admission, comparé à une distribution de Poisson ajustée, pour $p = 100\text{kb/s}$.

6.5.3 Utilisation et probabilité de débordement

La relation entre l'utilisation et la probabilité de débordement représente la performance optimale qui pourrait être atteinte par l'algorithme dans des conditions stationnaires. Elle est obtenue pour chaque MBAC en faisant varier son critère d'admission. Pour (GT), nous faisons varier ϵ et notons l'utilisation et la probabilité de débordement réalisées. Pour les algorithmes de Cross-Protect (Poisson) et (MinVar), notés (XP) dans la figure, nous changeons simplement le seuil d'admission sur la charge B_t, sans utiliser ni la moyenne ni la variance. La figure 6.4 (à gauche) présente les résultats obtenus lorsque tous les flots ont un débit crête de 100kb/s. S'y trouve également la même relation pour une charge théorique suivant une distribution de Poisson.

Sans surprise au vu des résultats présentés dans [38], la performance obtenue est globalement la même pour les deux MBACs. Une différence subtile se produit à forte charge. Au fur et à mesure que le lien devient saturé, les flots dans Cross-Protect se retrouvent momentanément *backlogged* et ne contribuent plus au *priority load*, ce qui permet d'autres admissions et ainsi une utilisation plus élevée. De telles fortes valeurs du seuil correspondent à des situations où les flots de débit p deviennent *backlogged*, et où la performance n'est plus correctement mesurée par ov. Les seuils adaptatifs d'admission visent à prévenir de tels cas.

La figure suggère que la charge sur chaque intervalle est moins variable que Poisson, puisque l'utilisation atteinte est plus forte à probabilité de débordement égale. La figure 6.4 (à droite) montre l'histogramme empirique du nombre de paquets qui arrivent dans un intervalle pour des flots de débit crête 100kb/s sous le MBAC (Poisson). L'ajustement à une distribution de Poisson donne une nouvelle courbe proche de la mesure empirique, suggérant que notre approximation, certes conservatrice, reste très raisonnable (le trafic réel est moins variable). D'autres résultats non présentés ici montrent le même comportement pour d'autres valeurs de débit crête.

L'utilisation atteinte empiriquement ici pour un taux de débordement donné constitue une référence utile par la suite pour comparer la performance des deux variantes de notre algorithme. Le seuil d'admission optimal étant celui qui garantit l'utilisation la plus forte, tout en conservant une proportion négligeable de paquets traités en *backlog*.

6.5.4 Prédictibilité du MBAC XP

Bien que les algorithmes de MBAC possèdent généralement la même frontière de performance, ils diffèrent par la simplicité ou non de leur paramétrage, et par leur capacité à fournir une performance prévisible pour un jeu de paramètres donnés. Nous évaluons cette capacité pour les algorithmes (Poisson) et (MinVar) pour des flots de débit crête homogène r. Le débit protégé p peut être égal, supérieur ou inférieur à r. Des résultats avec un intervalle de confiance de 95% sont présentés dans le tableau 6.2 pour un intervalle $\tau = L/p$.

%	p	r	ov	ut	bl	bk	lo
Poisson	50	50	3.98e-4 ±0.92e-4	84.08 ±0.02	31.37 ±0.38	2.82e-3 ±0.52e-3	0.00 ±0.00
	100	100	3.11e-4 ±0.64e-4	78.44 ±0.06	35.20 ±0.61	8.14e-3 ±1.80e-3	0.00 ±0.00
	100	50	3.05e-3 ±0.20e-3	78.79 ±0.02	35.80 ±0.29	1.85e-5 ±2.19e-5	0.00 ±0.00
	100	300	2.71e-4 ±0.86e-4	97.63 ±0.96	3.04 ±1.49	76.27 ±3.32	15.60 ±1.14
MinVar	50	50	5.42e-3 ±0.31e-3	88.28 ±0.05	27.99 ±0.33	6.30e-2 ±0.44e-2	0.00 ±0.00
	100	100	3.24e-3 ±0.15e-3	83.59 ±0.08	30.98 ±0.59	1.32e-1 ±0.10e-1	0.00 ±0.00
	100	50	7.69e-3 ±0.30e-3	81.25 ±0.06	33.68 ±0.40	6.54e-4 ±5.62e-4	0.00 ±0.00
	100	300	2.13e-4 ±0.56e-4	98.33 ±0.69	1.84 ±0.98	78.91 ±2.56	16.21 ±1.08

TABLE 6.2 – Performance du MBAC pour Cross-Protect pour des flots de même débit crête

Flots de débit crête connu

Les résultats pour le cas $r = p$ sont présentés dans le tableau 6.2, pour $p = 50$Kb/s et $p = 100$kb/s. La probabilité de débordement reste d'un ou deux ordres de grandeur plus faible que notre objectif, ce qui conduit à une utilisation moins importante qu'il n'aurait été possible (déduite de la figure 6.4). Cependant, la perte en utilisation n'est pas trop importante et les contraintes de qualité de service sont respectées. Cette tendance conservatrice est un caractéristique commune à tous les MBAC. Les résultats pour le MBAC (MinVar) montrent que l'utilisation de la mesure de variance améliore la performance puisque l'agrégat de trafic est moins variant que Poisson (en raison du blocage notamment). La performance est encore meilleure pour $k = 2$.

Flots de débit crête plus faible

Généralement, la valeur de p sera suffisamment élevée pour protéger tous les flots *streaming*, et beaucoup auront un débit crête bien plus faible. Les lignes de la table 6.2 pour $p = 100$ et $r = 50$ montrent l'impact de supposer un débit crête plus élevé qu'il ne l'est. Avec le MBAC (Poisson), le seuil dépend uniquement de p de telle sorte que la performance réalisée avec $r = 50$kb/s est grossièrement similaire à $r = 100$kb/s. Cela illustre le coût d'utiliser le MBAC simple (Poisson), qui reste faible cependant avec une telle taille de lien (10Mb/s). (Minvar) est plus efficace dans cette situation lorsque l'intervalle d'échantillonnage est égal à kL/p et $k \geq 2$ (résultats non présentés ici) et, en fait, il donne la même performance que le MBAC (GT) (l'utilisation progresse de 81.25 $\pm$ 0.06 % à 88.02 $\pm$ 0.06 %).

Flots de débit crête plus important

Le dernier cas où $r > p$ illustre l'avantage majeur du MBAC (XP). Quand le critère d'admission est fixé à $p = 100$, les flots de débit crête $r = 300$ deviennent *backlogged* et ne contribuent plus que très faiblement au *priority load* (seulement les débuts de bursts). Le lien est complètement utilisé et ces flots perdent une forte proportion de leur paquets. La différenciation ne dépendant que du débit p, il est normal que l'on n'observe aucune différence entre les deux algorithmes. Enfin, dans les cas présentés, le *fair rate* à long terme ne passe jamais au dessous du seuil fixé à $C/100$ (ce qui correspond à une forte charge), d'où l'absence de blocage.

6.5.5 Performance avec un mélange de débits crête

Nous considérons maintenant deux classes de flots, de débits crête en dessous et au dessus de p, et une charge totale $1 \leq \rho_1 + \rho_2 \leq 1.40$. La figure 6.5 représente la différenciation réalisée dans un ensemble de simulations où $r_1 = 50$kb/s et $r_2 = 300$kb/s, respectivement avec (Poisson) (à gauche) et (MinVar) avec $k = 1$ (au milieu) et $k = 2$ (à droite). ρ_1 et ρ_2 sont représentées sur les axes x et y, et les nuances de gris représentent des niveaux de charge constante. Le symbole indique le niveau de différenciation : ✓ (totale), $\sim$ (bonne, se produit avec un certain délai), $\approx$ (différenciation épisodique), $\times$ (pas de différenciation).

Avec (Poisson), la différenciation se produit pour $\rho = 1.20$ et $\rho = 1.40$ dès que la charge de la classe de faible débit crête n'est pas trop importante. (MinVar) améliore la différenciation tout en maintenant les critères de QoS, notamment quand k est fixé de manière adéquate (ici $k = 2$). (MinVar) avec $k = 2$ parvient à différencier le trafic dans l'ensemble des configuration avec $\rho = 1.20$ et $\rho = 1.40$.

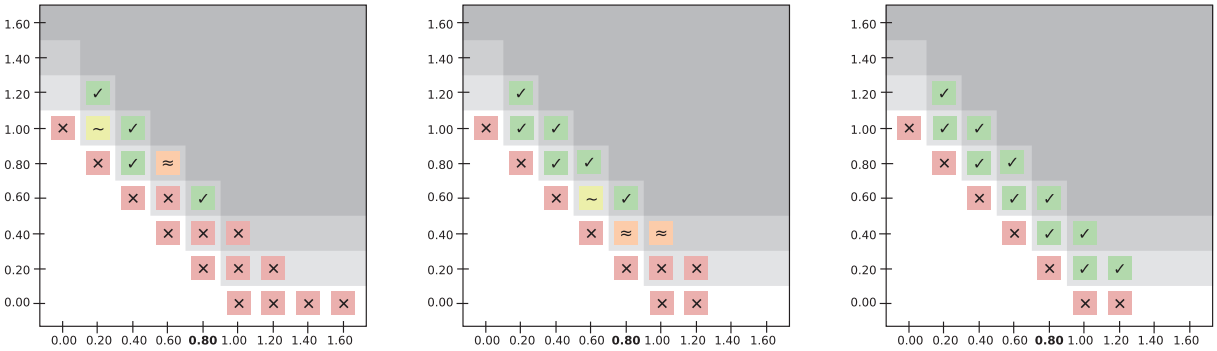


FIGURE 6.5 – Différenciation réalisée par (Poisson) avec $k = 1$ (à gauche), et (MinVar) avec $k = 1$ (au milieu) et $k = 2$ (à droite). Dans chaque figure, le débit crête protégé est $p = 100$kb/s, et la charge des flots de classe 1 ($r_1 = 50$kb/s) et 2 ($r_2 = 300$kb/s) est représentée respectivement en abscisse et en ordonnée.

En regardant de plus près les métriques de performance, nous voyons que la différenciation est variable selon les paramètres de trafic. La discrimination est reflétée dans les différentes valeurs du taux de blocage bl et du taux de pertes lo. Il est nécessaire dans tous les cas d'éliminer au moins l'excédent de charge ($bl + lo > (\rho - 1)/\rho$). Dans les cas sans discrimination, cela est réalisé uniquement par blocage, et l'utilisation reste relativement peu élevée. Dans les autres, les flots de plus fort débit perdent des paquets, ce qui réduit la proportion bloquée (qui est la même pour les deux classes) et l'utilisation est proche de 100%. Les flots différenciés sont servis en fonction de la bande passante laissée disponible par les flots prioritaires, et leur qualité de service est assurée par la borne inférieure sur le *fair rate* à long terme.

Le début de la simulation correspond à un régime transitoire où la charge prioritaire inclue l'ensemble du trafic entrant. Lorsque le lien commence à être saturé, les flots de plus fort débit crête deviennent *backlogged* et cessent de contribuer à cet estimateur. Cette situation est due au fait que le MBAC se base sur une estimation de la variance donnée par l'approximation Poissonnienne. Une fois la différenciation effectuée, PL décroit jusqu'à une valeur qui permet de nouvelles arrivées. Ce processus continue jusqu'à ce que le MBAC n'accepte plus aucun flot (lorsque le *fair rate* instantané deviendrait plus faible que p).

Parfois, la composition du trafic est telle que le MBAC ne permet pas que le lien devienne saturé. Cela se produit généralement quand l'estimation de la variance est basse à cause à la fois de débits crête relativement faibles, et d'une faible demande de la part des flots de fort débit. Les cas où p est petit comparé au seuil sur FR seront favorables à une différenciation. Dans tous les cas, nous voyons que les flots de débit crête inférieur à p sont protégés, puisqu'ils subissent des taux de débordement et de *backlog* négligeables, et aucune perte.

6.5.6 Impact de la gigue

En pratique, les flots de débit nominal r sont sujets à des délais variables et acquièrent de la gigue lorsqu'ils sont multiplexés dans les files d'attente successives des routeurs qu'ils traversent. Il est important de comprendre comment ce phénomène peut affecter la performance du MBAC. Une hypothèse de pire cas, selon la conjecture dite de "gigue négligeable" [33], est que les flots envoient des paquets selon un processus de Poisson de débit r pendant leurs périodes d'activité. **Cela sera vraisemblablement pire que la gigue réellement acquise par les flots**, notamment dans un réseau équipé de routeurs Cross-Protect où le *fair queueing* tend à restaurer l'espacement original des flots gigués. L'évaluation du MBAC soumise à un trafic Poissonnien au niveau paquet n'en reste pas moins intéressante en soi.

Nous avons ré-évalué la performance des deux algorithmes dans les scénarios précédents lorsque les flots ont acquis de la gigue. Pour des raisons de place, nous ne détaillons ici que nos principales observations.

Lorsque le débit des flots $r < p$, la gigue n'est pas assez importante pour avoir un impact sur la performance. La différence se produit lorsque r est proche de p. Les paquets de flots gigués sont temporairement retardés par l'ordonnanceur lorsque leur débit instantané est plus grand que p. Cela permet au MBAC de saturer le lien lorsque les caractéristiques du trafic le permettent. Les nouveaux flots sont bloqués par la condition sur FR, qui est un signe de forte charge. Alors que (GT) n'opère que sur l'ensemble du trafic, Cross-Protect peut mettre en attente les paquets qui possèdent un fort débit instantané, admettre de nouveaux flots et ainsi diminuer la probabilité de blocage. De cette façon, il

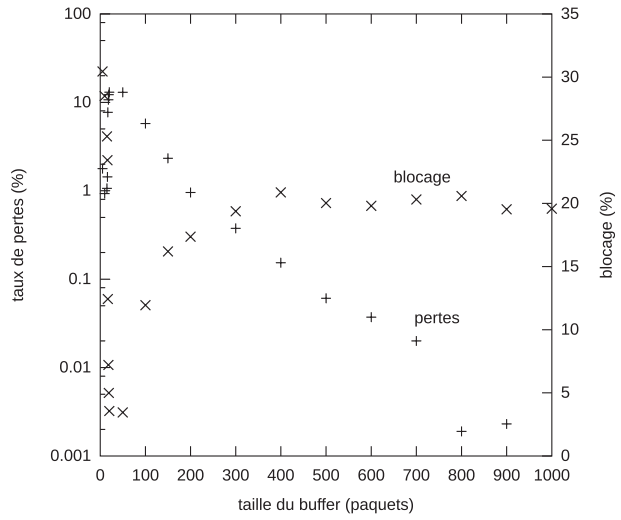


FIGURE 6.6 – Illustration du compromis taux de pertes de paquets / taux de blocage des flots en fonction de la taille du buffer pour $r = p = 100$kb/s

réduit la gigue, ce qui est un avantage supplémentaire de la différenciation. Il convient toutefois de s'assurer que le buffer est suffisamment dimensionné pour accueillir les paquets sans provoquer de perte.

La figure 6.6 présente à titre illustratif ce compromis pertes/blocage pour différentes valeurs de B, dans le cas $r = p = \theta_{FR} = 100$kb/s qui est le plus défavorable ici.

6.5.7 Contrôle d'admission en régime non-stationnaire : situations de *flashcrowd*

Nous considérons maintenant les cas non-stationnaires introduits dans la Section 6.4.4, en limitant notre étude à un scénario simple où tous les flots sont UDP[3]. Les flots ont les mêmes propriétés que dans le cas homogène précédent (arrivées selon un processus *on/off*, distribution exponentielle de leur taille). La topologie simulée est illustrée en figure 6.3 et consiste en deux topologies parallèles similaires aux simulations précédentes. La charge initiale sur chacune est de 60%. La durée de simulation est de 1000s ; le second lien central subit une panne à $t = 500$s, ce qui cause le reroutage de tout le trafic sur le premier lien qui devient goulotté.

Nous fixons $p = 100kb/s$ et considérons des flots de débit crête $r_a = 100$kb/s (un pire cas pour la performance). La figure 6.7 montre l'évolution des estimateurs FR (à gauche) et PL (à droite), accompagnés de leurs valeurs instantanées. Après la panne, les flots de débit r_a deviennent *backlogged* le temps de quelques secondes jusqu'à ce que le lien retrouve le régime stationnaire tel qu'obtenu dans la section 6.5.4. Le *fair rate* instantané devient inférieur à p, ce qui indique que trop de flots ont été admis. Pendant l'événement de *flash-crowd*, le blocage des flots est dû uniquement à la limite du nombre d'admissions introduite dans la section 6.4.4. L'ajout de la valeur du débit protégé p à la charge mesurée B_t permet de corriger l'estimateur qui est ainsi mis à jour rapidement malgré le coefficient de lissage. Il serait inutile sinon puisque l'échelle de temps caractéristique calculée n'est plus représentative du taux d'arrivées et de départ des flots. Nos simulations montrent que la performance est bien plus affectée si nous désactivons cette condition supplémentaire.

Afin de comprendre l'impact du débit crête sur la performance des flots *streaming*, nous considérons également le cas $r_b = 50$kb/s. La figure 6.8 trace la probabilité de *backlog* avec des flots de débit r_a (à gauche) and r_b (à droite), avec les deux algorithmes et dans un ensemble de configurations. Pendant quelques secondes après le reroutage, une grande partie des flots *streaming* est *backlogged*. Avec r_b, ce taux est bien plus faible, ce qui est encourageant puisque comme nous l'avons déjà précisé, la valeur de p sera généralement bien plus élevée que le débit effectif des flots à protéger.

Bien qu'encourageants, ces résultats préliminaires illustrent clairement la difficulté de contrôler le trafic au travers d'un mécanisme de contrôle d'admission, dans des cas tels que celui considéré. La condition limitant le nombre de flots acceptés dans un intervalle de temps peut s'avérer trop conservatrice lorsque les flots ont un débit crête inférieur à p ou, comme observé dans des traces réelles, s'il y a de nombreux flots consistant en un seul paquet. Le blocage des nouvelles arrivées après avoir détecté une congestion comme nous le faisons ici peut être insuffisant pour des régimes non-stationnaires.

3. Les flots TCP reroutés pourraient se retrouver en mode *slow start*, et accroître leur débit progressivement après avoir été acceptés, ce qui combiné avec leur contrôle adaptatif, rend la performance moins compréhensible.

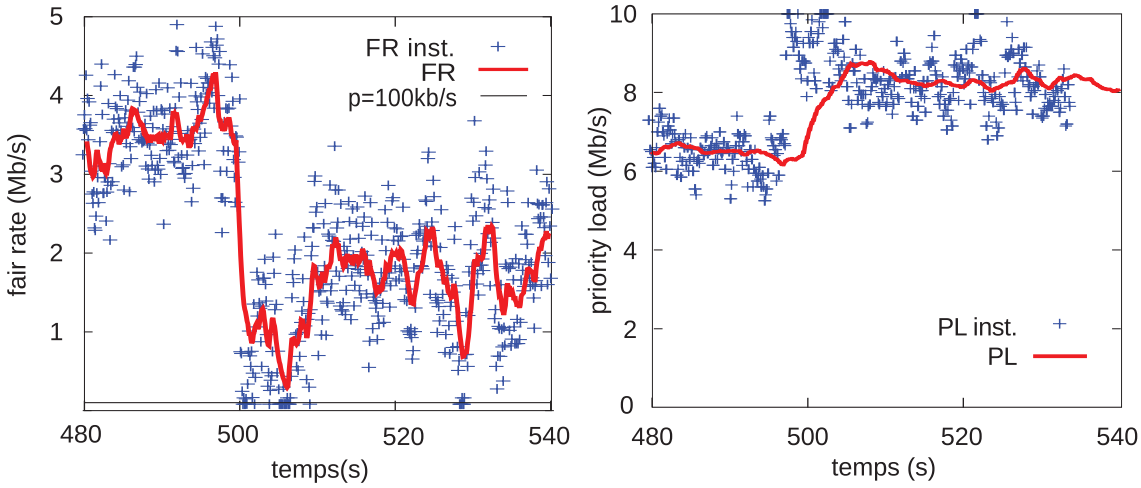


FIGURE 6.7 – Évolution du *fair rate* (à gauche) et du *priority load* (à droite) lors d'un événement de *flashcrowd* avec MinVar, $k = 2$, $p = 100\text{kb/s}$ et $r = 100\text{kb/s}$

Cette remarque est d'autant plus justifiée si l'on considère des scénarios plus réalistes avec des flots TCP et des distributions à queue lourde. Des travaux futurs peuvent considérer l'ajustement des facteurs de lissage en fonction du changement de la variance qui est caractéristique de telles situations. Il est également possible de tenter d'utiliser des algorithmes de détection de changement (issus de la théorie du signal). La condition est que ces solutions restent relativement simples à implémenter sur des liens à haut débit. Il faut cependant garder à l'esprit que des mesures plus radicales seront sans doute nécessaires pour empêcher une dégradation globale de la performance, comme l'interruption de flots déjà établis.

6.6 Conclusions

La proposition FAN basée sur les mécanismes Cross-Protect permet des garanties de performance pour les flots *streaming* et élastiques sans la complication de devoir marquer les paquets pour une différenciation de service. Il est toutefois nécessaire de proposer et calibrer des algorithmes de contrôle d'admission adaptés. Des travaux précédents suggèrent qu'un algorithme simple fondé sur une estimation du *fair rate* est suffisant lorsque la plupart des flots constituant le trafic sont élastiques et goulottés sur le lien considéré. Cependant, dans un réseau de cœur, la majorité des flots est contrainte ailleurs, par exemple par leurs débits d'accès plus faible que celui du lien. Le trafic est alors composé d'un ensemble de flots avec des débits crête limités. Certains ont un débit inférieur au *débit protégé* p (et ils doivent être servis en priorité), tandis que pour d'autres sont de débit supérieur à p. Le challenge est de proposer un MBAC qui permette à ces derniers de devenir *backlogged* (ils sont supposés à débit adaptatif), tout en continuant de servir les autres en priorité. Cela est d'autant plus délicat que l'algorithme ne peut se baser sur aucune connaissance des caractéristiques des flots.

Dans ce chapitre, nous avons proposé un MBAC simple basé sur des mesures de la moyenne et de la variance de la charge offerte à la file prioritaire de Cross-Protect. Il diffère de celui proposé par Grossglauser et Tse [66] en ce que la variance n'est utilisée que si elle est inférieure à la variance estimée lorsque l'on suppose que tous les flots ont le débit cible p. Nos simulations montrent que notre proposition permet la différenciation que nous recherchons dès lors que les débits des flots ne sont pas trop proches les uns des autres.

Les évaluations dans un scénario de *flash-crowd* sont moins encourageantes. Il est difficile pour de nombreuses raisons de trouver un compromis acceptable entre un trop grand nombre de flots acceptés – qui entraîne une période importante où la performance est dégradée – et un fort conservatisme qui se traduit par le rejet de plus de flots que nécessaire. Une étude plus approfondie d'un tel scénario est nécessaire vu son importance pratique. Nos résultats suggèrent qu'un contrôle d'admission simple n'est peut-être pas suffisant. Il est possible que l'interruption d'un sous ensemble des flots en cours soit également nécessaire afin de retrouver un niveau de charge acceptable.

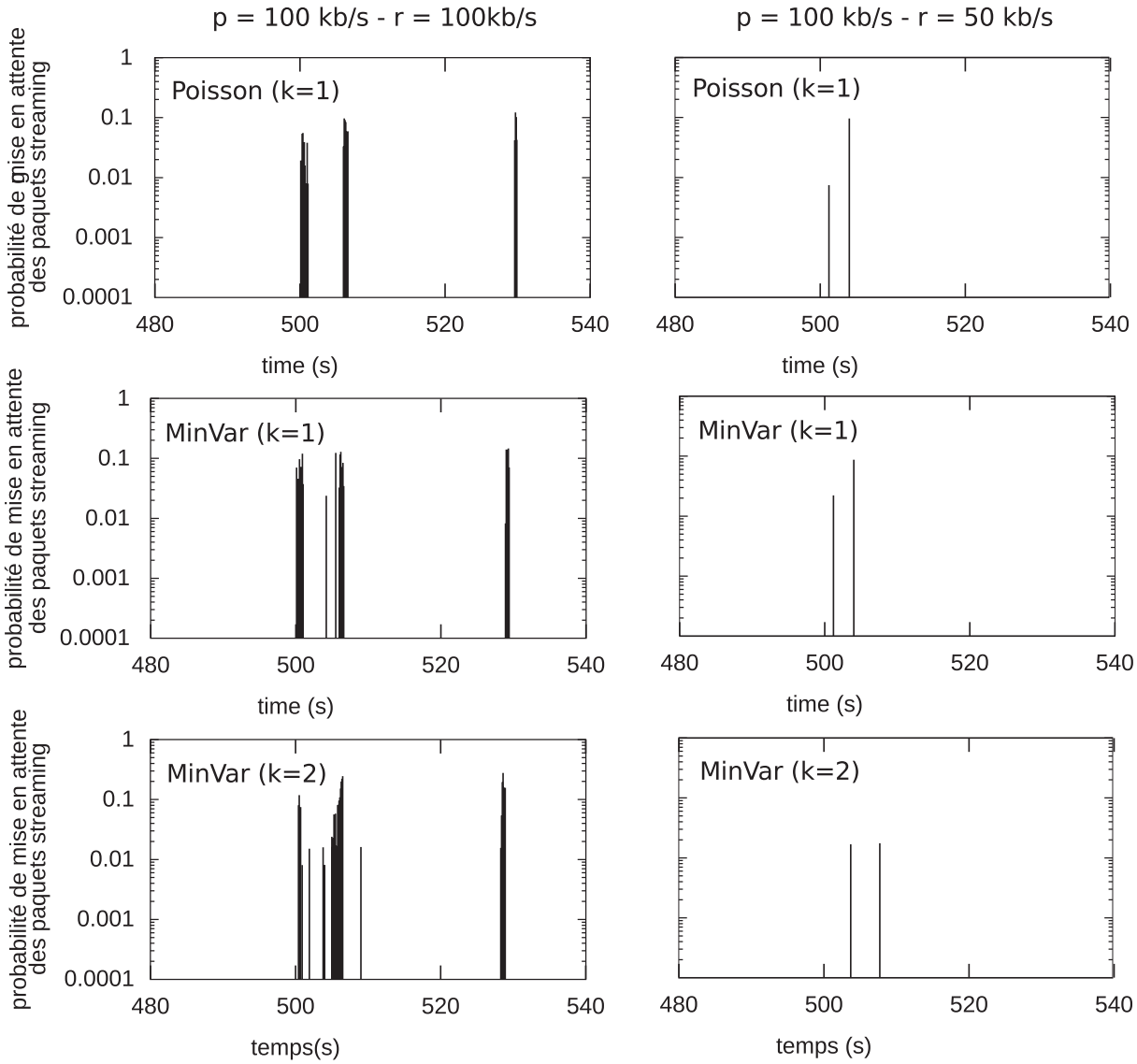


FIGURE 6.8 – Probabilité de mettre des paquets *streaming* en file d'attente dans deux scénarios : $p = 100$kb/s, $r = 100$kb/s (à gauche) et $p = 100$kb/s, $r = 50$kb/s (à droite), avec Poisson, $k = 1$ (en haut), MinVar, $k = 1$ (au milieu), and $k = 2$ (en bas).

6.A Estimation de la variance sur un intervalle de taille quelconque

Afin d'obtenir le résultat souhaité, nous devons d'abord établir quelle est la variance d'une somme d'un nombre aléatoire de variables aléatoires.

$$\mathrm{Var}\left[\sum_{i=1}^{N_t} M^{(i)}\right] = \mathrm{Exp}[S^2] - \mathrm{Exp}[S]^2, \text{ avec } S = \sum_{i=1} N_t M^{(i)}$$

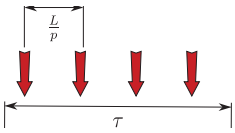
$$\mathrm{Exp}[S] = \mathrm{Exp}[N] \cdot \mathrm{Exp}[M]$$

$$\begin{aligned}
\mathrm{Exp}[S^2] &= \mathrm{Exp}\left[\sum_{i=1}^{N_t} M_i^2 + \sum_{i=1, i\neq j}^{N_t}\sum_{j=i}^{N_t} M_i M_j\right] \\
&= \mathrm{Exp}[N]\,\mathrm{Exp}[M_i^2] + \mathrm{Exp}\left[N(N-1)\,\mathrm{Exp}[M]^2\right] \\
&= \mathrm{Exp}[N]\,\mathrm{Exp}[M^2] + \left[\mathrm{Exp}[N^2] - \mathrm{Exp}[N]\right]\mathrm{Exp}[M]^2
\end{aligned} \tag{6.9}$$

D'où :

$$\begin{aligned}
\mathrm{Var}[S] &= \mathrm{Exp}[N]\,\mathrm{Exp}[M^2] + \mathrm{Exp}[N^2]\,\mathrm{Exp}[M]^2 - \mathrm{Exp}[N]\,\mathrm{Exp}[M]^2 - \mathrm{Exp}[N]^2\,\mathrm{Exp}[M]^2 \\
&= \mathrm{Exp}[N]\,\mathrm{Var}[M] + \mathrm{Var}[N]\,\mathrm{Exp}[M]^2
\end{aligned} \tag{6.10}$$

Nous déterminons maintenant la variance mesurée sur un intervalle de taille τ quelconque, en supposant toujours que les flots ont un débit crête constant p pendant leur période d'activité :



Puisque nous avons un processus d'arrivée Poissonnien, le nombre de flots actifs dans l'intervalle t, N_t, est également Poissonnien, au taux λ. Nous appliquons également une hypothèse de séparation des échelles de temps afin de pouvoir considérer que le nombre de flots actifs dans un intervalle est constant.

Le nombre de paquets reçus pour un flot actif pendant l'intervalle t est $M_t = D_t + R_t$ où $D_t = \lfloor \frac{\tau p}{L} \rfloor$ et R_t tel que $\Pr\{R_t = 0\} = \frac{\tau p}{L} - D$ et $\Pr\{R_t = 1\} = 1 - \frac{\tau p}{L} + D$.

$$\begin{aligned}
A_t &= \mathrm{Exp}[\sum_{i=1}^{N_t} M_t^{(i)}]\frac{L}{\tau} \\
&= \mathrm{Exp}[N_t] \cdot \mathrm{Exp}[M_t]\frac{L}{\tau} \\
&= \lambda\left[(D_t+1)(\frac{\tau p}{L} - D_t) + D_t(1 - \frac{\tau p}{L} + D_t)\right]\frac{L}{\tau}) \\
&= \lambda\left[\frac{\tau p}{L} - D_t\right]\frac{L}{\tau} \\
&= \lambda p
\end{aligned} \tag{6.11}$$

$$V_t = \left[\sum_{i=1}^{N_t} M_t^{(i)}\right]\frac{L^2}{\tau^2} \tag{6.12}$$

Nous avons alors :

$$\mathrm{Exp}[M_t] = D_t + \frac{\tau p}{L} - D_t = D_t + \gamma \tag{6.13}$$

$$\mathrm{Var}[M_t] = \mathrm{Var}[R_t] = \left(\frac{\tau p}{L} - D_t\right)\left(1 - \frac{\tau p}{L} + D_t\right) = \gamma(1-\gamma) \tag{6.14}$$

En insérant 6.13 et 6.14 dans 6.10, nous obtenons :

$$
\begin{aligned}
V_t &= \lambda \left[\gamma(1-\gamma) + (D_t + \gamma)^2 \right] \frac{L^2}{\tau^2} \\
&= \lambda \left[\frac{D^2 L^2}{\tau^2} + \gamma(1 + 2D_t) \frac{L^2}{\tau^2} \right] \\
&= \frac{A_t}{p} \left[\frac{D^2 L^2}{\tau^2} + (\frac{\tau p}{L} - D_t)(1 + 2D_t) \frac{L^2}{\tau^2} \right] \qquad (6.15)
\end{aligned}
$$

Quand $p < \frac{L}{\tau}$, $D_t = 0$ et V_t se simplifie en :

$$
\begin{aligned}
V_t &= \lambda \frac{\tau p}{L} \cdot \frac{L^2}{\tau^2} \\
&= \lambda \frac{PL}{\tau} \\
&= A_t \cdot \frac{L}{\tau} \qquad (6.16)
\end{aligned}
$$

Si $\tau \leq \frac{L}{p}$ alors la mesure de variance reste égale à la mesure obtenue dans le cas simpliste où $\tau = \frac{L}{p}$ que nous avons vu précédemment.

Chapitre 7

Self-Protect : une approche orientée flot et centrée sur l'utilisateur pour la QoS des liens d'accès

Dans les chapitres précédents, nous avons considéré des environnements de réseau caractérisés par un grand nombre d'utilisateurs, et une bande passante supérieure de plusieurs ordres de grandeur au débit des flots. Nous y avons proposé et étudié la proposition Cross-Protect, dont les mécanismes d'ordonnancement équitable par flot et de contrôle d'admission pourraient être également appliqués au contexte des *datacenters*.

Le réseau d'accès cependant ne possède pas les mêmes caractéristiques, et nous verrons qu'une solution seulement basée sur du *fair queueing* n'est pas suffisante pour garantir la bonne performance des flots. Il est pourtant nécessaire si l'on souhaite protéger les flots utilisateur de bout en bout de considérer la performance du réseau d'accès, qui est généralement un goulot d'étranglement pour le trafic.

Dans ce chapitre, nous envisageons une déclinaison de l'architecture *Flow-Aware Networking* pour le réseau d'accès, que nous nommons Self-Protect, permettant la gestion du trafic au niveau flot, dans les sens montant et descendant. L'utilisation d'un ordonnanceur de paquets approprié nous permet une réponse aux problèmes récurrents de *bufferbloat*, tout en considérant la performance des flots élastiques requérant de larges buffers. Cette solution présente l'avantage de reposer sur la collaboration de l'utilisateur, et reste donc une approche neutre vis à vis des réseaux

Après avoir proposé un ensemble de mécanismes de gestion des flots, nous décrivons la proposition Self-Protect, pour enfin illustrer ses avantages dans un petit nombre de cas d'utilisation représentatifs pour un utilisateur du réseau d'accès.

Sommaire

7.1 Introduction 109
7.2 Mécanismes actuels de garantie de service pour le réseau d'accès 109
7.3 Vers une architecture de QoS pour le réseau d'accès 110
7.4 L'architecture Self-Protect 112
7.5 Réalisation et évaluation de l'architecture Self-Protect 115
7.6 Conclusion 119

Contributions :

- Proposition d'une architecture de QoS au niveau flot adaptée au réseau d'accès.
- Méthode de synchronisation de table de flots, permettant la gestion du trafic descendant
- Réalisation d'un contrôleur permettant la gestion décentralisée d'une table de flots
- Réalisation d'un prototype et démonstrations internes.

Publications et présentations :

Les résultats de ce chapitre n'ont pas été publiés. La réalisation du prototype a été faite conjointement avec Elie Roux, qui a effectué son stage de fin d'études au sein de l'équipe, et pour qui j'ai assuré l'encadrement technique.

7.1 Introduction

Alors que les questions de Qualité de service dans le réseau de cœur sont débattues et divisent les chercheurs, il est généralement reconnu que le réseau d'accès présente un goulot d'étranglement pour le trafic. Récemment, le problème de *bufferbloat* [60] a été exhibé et a entraîné un regain d'intérêt de la communauté pour définir des mécanismes de contrôle efficaces pour le réseau d'accès (notamment les AQM CoDel, FQ-Codel et PIE). Mais le problème est complexe et n'a toujours pas reçu de réponse satisfaisante.

L'introduction de *fair queueing* tel que proposée dans Cross-Protect convient pour le cœur de réseau avec des liens de forte capacité. Elle n'est pas suffisante pour un lien d'accès, où un utilisateur peut souhaiter exprimer des politiques de gestion du trafic complexes, et où un flot *streaming* à protéger peut avoir un débit occupant une grande partie de la capacité d'accès.

Dans ce chapitre, nous considérons un mécanisme de gestion du trafic pour les ressources près du client, comme le dernier kilomètre d'un réseau ADSL, ou la connexion sans-fil à un point d'accès. Dans le sens montant, depuis le réseau domestique jusqu'au premier nœud du réseau, il est relativement naturel que l'utilisateur puisse déterminer les priorités qu'il souhaite donner aux flots concurrents qu'il génère, via les "boxes". Pourtant, dans le réseau actuel, c'est plutôt l'opérateur qui impose ses propres priorités. D'autant plus dans le sens descendant, où l'opérateur donne priorité au trafic à valeur ajoutée provenant de son propre réseau (VoIP, flux TV, etc.), tandis que les autres flots restent traités en *Best-Effort*.

Notre approche propose la réalisation dans le réseau des mécanismes permettant à l'utilisateur de déterminer l'ordonnancement et le contrôle appliqué à ses flots dans le sens descendant, de la même manière que dans le sens montant. Le profil des flots est déterminé par l'équipement domestique de l'utilisateur, qui peut analyser le ou les premiers paquets et les associer à une politique locale. Il peut alors être utilisé localement et envoyé à l'équipement en amont à l'aide d'un protocole de signalisation.

Au delà de permettre une gestion plus appropriée du trafic utilisateur, cette architecture qui réalise les mécanismes FAN dans le réseau d'accès permet de collecter des informations importantes sur le réseau du côté client, permettant le diagnostic des pannes ou de certains problèmes de connexion ou de performance. Un prototype GNU/Linux a été réalisé afin de montrer les avantages et la faisabilité d'une telle architecture.

7.2 Mécanismes actuels de garantie de service pour le réseau d'accès

L'architecture des réseaux d'accès diffère fortement d'un réseau à l'autre en fonction du contexte ou des technologies utilisées : réseau filaire ADSL, réseau mobile UMTS ou LTE, accès VPN d'entreprise, etc. On constate généralement soit l'application de mécanismes de type DiffServ, soit l'absence de gestion du trafic. Nous présentons plus en détail le cas d'un accès ADSL, pour lequel nous illustrons notre proposition. Nous remarquons toutefois son applicabilité aux autres types de réseaux d'accès.

7.2.1 Gestion du trafic montant

La configuration la plus classique dans un accès ADSL est la présence d'une passerelle à la frontière entre le réseau d'accès et le réseau domestique. Il s'agit généralement d'un système d'exploitation GNU/Linux, pour lequel sont disponibles la plupart des mécanismes classiques présentés dans le Chapitre 2, section 2.3[1].

Les premiers cas d'utilisation de mécanismes de QoS, dans les cas favorables d'équipements ouverts à l'utilisateur, remontent à 2002 avec des scripts de type *Wondershaper* [74], ayant pour objectif :

- le maintien d'une faible latence pour le trafic interactif (eg. SSH),
- la navigation à une vitesse raisonnable pour l'*upload* et le *download*,
- l'assurance que les transferts dans les sens montant et descendant ne se gênent pas les uns les autres.

De telles solutions sont très complexes et mettent en œuvre des mécanismes de partage de bande passante, des limites de débit (*shaping*), etc. Avec l'avènement des offre de *triple-play*, la présence de *box* opérateur s'est généralisée. Ces équipements ne disposaient généralement d'aucun mécanisme de QoS et ont longtemps géré le trafic en FIFO. Une exception est la Freebox en France, qui propose nativement une file prioritaire classifiant le trafic en fonction du champ IP ToS (*Type of Service*), et applique l'ordonnancement *Stochastic Fair Queueing* (SFQ) pour les files contenant du trafic élastique. Plus récemment, l'utilisation de l'ordonnanceur *Shortest Queue First* (SQF) est proposé pour la Livebox d'Orange [23,

1. voir par exemple le tutoriel de référence [102] pour une description détaillée de ces mécanismes

29, 16]. Il possède l'avantage de permettre un traitement du trafic au niveau des flots, ainsi qu'une différenciation implicite du trafic *streaming* basé sur le caractère *bursty* des flots. L'utilisation de SQF permet de se passer d'un marquage explicite généralement effectué par les applications, qui a tendance à ne plus trop être utilisé.

Aujourd'hui, l'identification du problème de *bufferbloat* [60] a relancé l'intérêt de la communauté pour proposer de nouvelles techniques de gestion du trafic pour le réseau d'accès, comme PIE [123] ou CoDel [115] qui permettent de contrôler les files d'attente afin d'y limiter le temps de séjour des paquets. Une combinaison de SFQ et CoDel (FQ-CoDel) a également été proposée par Dumazet [70], permettant en plus de donner priorité aux nouveaux flots afin de favoriser le trafic interactif, un mécanisme déjà présent dans la solution Cross-Protect (Chapitre 3). Enfin, la discipline *Fair Queue Packet Scheduler* [48], permet d'offrir un service de lissage du trafic à la source (*pacing*), qui peut-être utile pour éviter les rafales au sein des flots élastiques.

7.2.2 Gestion du trafic descendant

Dans le sens descendant ADSL, la plupart du trafic Internet est géré en *Best Effort*, alors que des mécanismes de gestion de trafic protègent généralement les flux de l'opérateur, comme la téléphonie, la TV ou la VoD. Il s'agit par exemple des circuits virtuels de l'architecture ATM, ou d'une classe DiffServ pour les réseaux IP. Il en est de même pour les réseaux mobiles, où la technologie UMTS définit par exemple 4 niveaux de priorités pour protéger le trafic le plus sensible : conversationnel, *streaming*, interactif, et *background*.

Afin de classifier les flots dont l'origine n'est pas connue, il est également possible de recourir à des techniques d'inspection plus ou moins sophistiquées (dites DPI, *Deep Packet Inspection*), qui peuvent inspecter jusqu'au contenu applicatif des paquets ou des flots. De telles solutions sont généralement utilisées pour identifier et appliquer des traitements aux flux applicatifs (dans des réseaux VPN d'entreprise par exemple), voire pour bloquer ou limiter certains types de trafic (pair-à-pair, applications non incluses dans un forfait mobile, etc.). Ces méthodes sont en conflit avec le principe de neutralité des réseaux, et sont actuellement sujettes à de nombreux débats.

7.2.3 Discussion

Comme cela a déjà été évoqué dans le Chapitre 2, même ces nouveaux mécanismes restent complexes à paramétrer et reposent pour certains sur une coopération des utilisateurs au moyen de TCP. Leur capacité de différenciation reste limitée, et ils n'offrent pas de performance prévisible et garantie, ni de protection contre une surcharge au niveau flot.

Il est également possible que l'utilisateur souhaite formuler des besoins plus précis, comme donner plus de débit à certains flots, ou en traiter d'autres en basse priorité. L'interaction entre les protocoles TCP et les AQM est mal connue et peut causer des résultats inattendus [63]. C'est pourquoi nous recommandons l'utilisation d'un ordonnancement par flot, guidé explicitement par l'utilisateur. Ceci nous permettra en outre une solution respectant le principe de neutralité des réseaux.

L'utilisation de *fair queueing* ou de FQ-CoDel seul n'est pas satisfaisante pour le réseau d'accès, où le profil de trafic d'un utilisateur peut être très complexe. Par exemple, une vidéo HD peut avoir un débit crête qui est une part significative du débit d'accès ; un utilisateur n'acceptera pas de dégrader la qualité de cette vidéo parce qu'elle sera en compétition avec des téléchargement et que son débit excèdera le débit équitable.

Enfin, nous proposons comme précédemment de réaliser une différenciation des flots *streaming*, afin de leur garantir un réseau transparent et leur assurer de faibles délais au sein des buffers, réglant ainsi le problème de *bufferbloat*. L'espace dans les buffers sera mis à profit pour garantir un débit satisfaisant pour les connexions TCP (voir Chapitre 4).

7.3 Vers une architecture de QoS pour le réseau d'accès

Dans cette section, nous discutons la réalisation d'un mécanisme de garantie de QoS pour le réseau d'accès, à la lumière des résultats présentés dans les chapitres précédents. La gestion du trafic concerne à la fois les sens montant et descendant, afin de traiter respectivement les flots envoyés et reçus par un utilisateur. Bien qu'elle n'offre pas de réponse complète et définitive au problème, l'architecture que nous présentons ici offre un compromis raisonnable entre performance et simplicité de mise en œuvre.

7.3.1 Classes de trafic et ordonnancement

Comme pour le cœur de réseau, nous supposons qu'il est possible de gérer le trafic utilisateur au moyen d'un faible nombre de classes de trafic, afin de permettre la réalisation de la solution sur un équipement gérant un grand nombre d'utilisateurs comme un DSLAM (nous supposons que l'allocation d'une certaine capacité à un utilisateur à été effectuée, indépendamment de la manière dont cette capacité sera partagée entre les flots) :

trafic *streaming* protégé : le trafic *streaming* sera traité en priorité selon les critères du multiplexage "sans buffer" présenté dans le Chapitre 4. Il conviendra pour cela de contrôler la charge de cette file d'attente. Lorsque cette classe de trafic constitue une part significative du trafic, Bonald *et al.* [25] suggèrent qu'il peut être nécessaire de distinguer entre les flots VoIP et les autres flots *streaming* moins sensibles à la latence (TV, etc.).

trafic élastique : ce sont les flots élastiques correspondant à du trafic de navigation (page web, email, etc) et nécessitant des garanties au niveau du temps de réponse. Nous recommandons ici l'utilisation de *fair queueing* entre les flots, tout en reconnaissant qu'il existe certainement des cas où l'on voudrait raffiner la gestion du trafic.

L'extension de la proposition avec des mécanismes permettant une différenciation entre flots élastiques serait une piste intéressante pour l'extension de la proposition. Nous n'avons pas non plus considéré l'introduction de mécanismes de *pacing*.

trafic à faible priorité : cette classe contiendra les flux élastiques non prioritaires, pouvant par exemple être interrompus et reprendre plus tard, et ne manifestant pas de phénomène d'impatience. C'est pas exemple le cas des gros téléchargement ou des transferts de fichiers en peer-to-peer.

Nous recommandons l'utilisation de *fair queueing* pour cette classe également, bien que sa performance ne nous intéresse pas. Le trafic de faible priorité sera géré en *Best Effort* lorsque de la bande passante sera laissée disponible par les autres classes. Il serait également possible de réserver une fraction de la bande passante pour cette classe.

trafic bloqué : comme son nom l'indique, elle contiendra les flux que l'on ne souhaite pas voir acheminés sur le lien.

Il conviendra d'associer des estimateurs de performance aux différentes classes de trafic pour lesquelle on veut offrir des garanties de performance, à l'image du *priority load* et du *fair rate* introduits dans le Chapitre 3 pour les flots *streaming* et élastiques.

7.3.2 Différenciation

Différenciation implicite streaming-élastique

La discipline *Shortest Queue First* (SQF) a été proposée pour gérer les flots au niveau des liens d'accès [23, 29]. Elle exploite le fait que les packets des flots *streaming* sont typiquement régulés par un codec, plutôt que par un algorithme de contrôle de congestion, et ce quel que soit le protocole de transport utilisé. Ils génèrent peu d'accumulation au sein des files d'attente, ce qui peut-être utilisé pour les différencier des flots élastiques. SQF nécessite que la charge des flots *streaming* ne soit pas trop importante, et que le buffer soit convenablement dimensionné. Ce mécanisme suppose de plus que les flots élastiques n'implémentent pas de mécanisme de *pacing*.

Différenciation implicite élastique-élastique

La discipline SRPT (*Shortest Remaining Processor Time*) [90] est connue pour être optimale en ce qui concerne le temps moyen de réponse des flots ; elle nécessite cependant une connaissance préalable des tailles de flots, ce qui n'est généralement pas le cas pour les réseaux. Plusieurs propositions ont été faites afin d'exploiter les avantages d'une différenciation basée sur la taille des flots, comme LAS (*Least Attained Service*) [69], qui est une bonne approximation de SRPT, ou encore RuN2C [10]. L'intêret d'une telle différenciation est également discuté par Bonald *et al* [25], notamment en cas de congestion, où les auteurs suggèrent que la solution est une alternative au contrôle d'admission qu'il serait intéressant de considérer et d'étudier. Ce même article suggère que les flots *streaming* peuvent souffrir d'une telle différenciation, et doivent être traités séparément, comme nous le faisons ici.

Différenciation explicite

L'avantage de la différenciation implicite est de ne pas dépendre des protocoles/contenus/chiffrements/etc., mais elle n'est pas toujours possible ou suffisante. Le nombre limité de flots utilisateur permet cependant l'utilisation de mécanismes avancés de différenciation explicite pilotés par l'utilisateur. Cela permet l'identification du trafic dans le sens montant (typiquement contrôlé par l'utilisateur), mais également dans le sens descendant comme nous le verrons dans la section 7.4.3.

7.3.3 Gestion de la surcharge

La garantie de faibles délais et taux de pertes pour la classe prioritaire, ainsi que d'un débit minimum pour la classe élastique requiert l'utilisation de mécanismes de contrôle de charge, similaires à ceux proposés dans l'architecture Cross-Protect. Il est désirable en effet de limiter l'accès aux files protégées lorsqu'un nouveau flot dégraderait la performance globale du système, et/ou de pouvoir en informer l'utilisateur. Il est alors possible de limiter la dégradation de performance à un sous-ensemble de flots bien délimité, qu'il est facile de communiquer à l'utilisateur.

Estimateurs de surcharge et contrôle d'admission

Afin de préserver les flots *streaming*, il est pratique de se reposer sur un algorithme de contrôle d'admission garantissant les conditions de multiplexage sans buffer (voir Chapitre 6, section 3.2.1), tel que celui proposé par Grossglauser et Tse [65, 66] et détaillé dans la section 6.2.7 du Chapitre 6 (mesure de la moyenne et de la variance de la charge prioritaire). La charge critique dépendra en effet de la composition du trafic entrant et de ses caractéristiques, notamment la combinaison des débits crête.

L'utilisation d'un ordonnancement *fair queueing* pour la classe élastique nous permet d'estimer le débit équitable sur le lien, ou *fair rate* (Chapitre 3, section 3.5.2) : il s'agit du débit qu'aurait un flot en l'absence de contrainte extérieure. Il est possible de détecter une surcharge lorsque ce débit équitable devient trop faible (il reste toutefois à définir un seuil efficace comme pour le cœur de réseau).

Le rejet d'un flot est difficilement envisageable sur le lien d'accès de l'utilisateur, sauf s'il est possible de l'utiliser comme un signal de reroutage dans le cas d'un *multi-homing*. Il est par contre possible d'admettre le flot dans la classe *Best-Effort*, avec information à l'utilisateur, en attendant une admission ultérieure. Cette solution permet d'admettre le flot tout en protégeant ceux déjà établis, un peu à la manière de la solution *Conditionally Dedicated Bandwidth* présentée dans le Chapitre 2 en section 2.4.1.

Préemption de flots

Dans notre contexte, un erreur d'admission, ou tout simplement le changement de profil d'un flot peut remettre en cause les décisions d'admission faites précédemment. Il convient alors de considérer la préemption de certains flots. Encore une fois, le choix des flots à interrompre est une tâche complexe qui peut se reposer sur des heuristiques (comme le rejet des derniers flots admis) ou l'interaction avec l'utilisateur.

7.4 L'architecture Self-Protect

L'architecture Self-Protect propose la gestion au niveau flot du trafic montant et descendant sur le lien d'accès d'un utilisateur. Elle repose sur le maintien d'une table des flots au sein des équipements, et d'un découplage entre les décisions de QoS faites pour les flots (par un contrôleur), et leur application. La gestion du trafic dans le sens descendant est rendue possible par la décentralisation du contrôleur vers le client (par exemple dans la passerelle domestique), qui communique le profil des flots à l'équipement en amont au travers d'un protocole de signalisation (voir Figure 7.1).

La proposition nécessite la définition a priori d'une architecture de gestion du trafic telle que faite dans la section précédente. Par souci de généralité, nous considérerons ici qu'un certain nombre de classes de service ont été définies, et qu'il est possible d'associer un profil de trafic à chaque flot afin qu'il soit affecté à l'une de ces classes. Ce profil peut correspondre simplement à un identifiant de priorité, mais être également plus complexe avec par exemple des garanties minimales et/ou maximales de bande passante.

7.4.1 Fonctionnement de Self-Protect

Supposons qu'un nouveau flot arrive depuis le réseau, afin d'illustrer le fonctionnement de notre proposition. Il est inconnu sur le DSLAM qui applique un traitement par défaut. Lorsqu'il atteint la

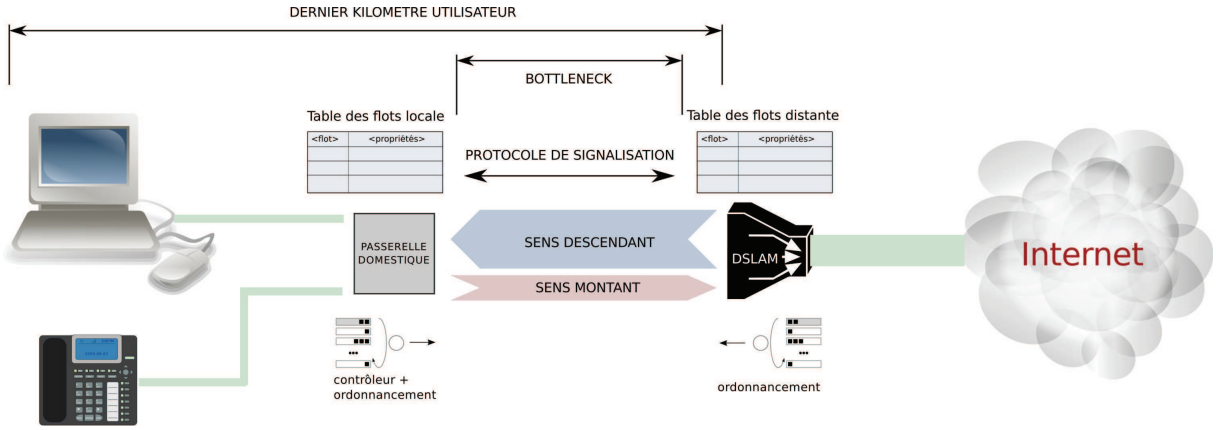


FIGURE 7.1 – Positionnement de l'architecture Self-Protect dans le réseau d'accès

passerelle domestique, il est reconnu comme nouveau flot, une nouvelle entrée est crée dans la table des flots, et un profil de QoS lui est affecté en fonction de la politique locale. L'identifiant de flot et son profil sont transmis au DSLAM au moyen d'un protocole de signalisation, qui les ajoute à sa table de flots. Une fois le flot reconnu, les paquets suivants seront ordonnancés de la même façon dans les deux directions[2]. Parfois, le premier paquet n'est pas suffisant pour déterminer la nature réelle du trafic, et il est tout à fait possible d'envoyer ou renvoyer la signalisation plus tard.

Le seul besoin essentiel de la proposition est de maintenir une table de flots dans les équipements en bordure de réseau. Elle peut ainsi s'appliquer à plusieurs types de réseaux d'accès filaires, comme xDSL ou PON – elle serait alors respectivement dans le DSLAM ou dans le switch/routeur –, ou sans-fil – elle serait alors localisée dans le RNC pour le réseau UMTS, ou dans le point d'accès d'un hotspot WiFi par exemple. Par souci de simplicité, nous nous restreindrons ici à l'étude d'un réseau d'accès ADSL.

7.4.2 Gestion des tables de flots

Une table des flots permet d'associer à chacun un profil de trafic (par exemple l'identifiant d'une file d'attente). Les deux fonctions minimales suivantes doivent être appliquées par les équipements.

traiter les paquets recus : Pour chaque paquet entrant, l'équipement cherchera une correspondance dans la table des flots, et marquera le paquet en fonction du profil de QoS, afin de l'affecter à la bonne file d'attente. Un traitement par défaut sera appliqué pour les paquets pour lesquels aucun flot n'est trouvé. Ce sont typiquement les premiers paquets d'un flot, ou ceux pour lesquels on ne souhaite pas associer de profil de trafic (flots courts de 1 paquet par exemple). Par souci de performance, il sera possible d'affecter un certain nombre de profils par défaut, qui ne nécessiteront pas de signalisation. Par exemple pour traiter en priorité les requêtes DNS, qui dépassent rarement 1 paquet.

gérer les flots inactifs : La date de dernière activité du flot sera mémorisée afin de pouvoir purger périodiquement la table des flots terminés. Il sera possible d'affecter un *timeout* plus faible sur le DSLAM afin de limiter la taille de la table des flots, quitte à ce que l'entrée d'un flot qui redevient actif soit rétablie au travers du protocole de signalisation.

Le DSLAM devra en plus obéir à la signalisation reçue par le contrôleur.

7.4.3 Contrôleur et protocole de signalisation

La reconnaissance des flots, l'attribution d'un profil de trafic, ainsi que les décisions d'admission sont entièrement réalisées par un contrôleur, décentralisé sur les équipements client qui sont seuls responsables du contrôle de leurs flots. Cette solution permet de distribuer la charge d'identification des flots et d'attribution d'une classe de service sur l'équipement utilisateur, en l'occurrence la passerelle domestique pour l'ADSL. Le logiciel tournant sur le DSLAM est réduit à son strict minimum : une interface permettant la manipulation d'une partie restreinte de la table des flots utilisée pour la gestion du trafic.

2. En fait il sera possible de donner des profils différents en fonction du sens, par exemple pour donner priorité au flux d'acquittement des connexions TCP

Reconnaissance des flots

Les premiers paquets de chaque flot sont redirigés vers un contrôleur dont la responsabilité est de leur affecter un profil de trafic. La reconnaissance des flots est faite de manière classique (sur le quintuplet IP et ports source et destination + protocole pour IPv4, sur les IP + le *flow label* pour IPv6).

La reconnaissance du profil des flots peut-être faite de manière implicite comme présenté plus haut, ou de manière explicite par un ensemble de règles, sous le contrôle de l'utilisateur. Ces règles pourront être choisies sur mesure par chaque utilisateur, par exemple à partir d'un dépôt contenant les applications les plus populaires, ainsi qu'un ensemble de profils-type d'utilisateur (par exemple un joueur voudra que les flots associés aux jeux-vidéos soient traités en priorité afin d'avoir une latence minimale).

La table de flots du côté utilisateur pourra également contenir un ensemble de statistiques utile pour le contrôleur afin de procéder à l'identification des flots (comme sa taille en octets, le nombre de paquets ou encore sa durée). L'utilisateur pourra bien sûr à tout moment ajouter de nouvelles règles ou modifier manuellement le profil d'un flot. Il sera également possible d'offrir un mode d'apprentissage comme pour les *firewalls* qui permettra la construction d'un jeu de règles personnalisé pour l'utilisateur. Enfin, pour le trafic montant, il sera toujours possible de s'appuyer sur le champ ToS (*type of service* de l'en-tête IP, qui contient des préférences de QoS demandées par les applications supportant cette fonctionnalité.

Gestion de la surcharge

Les décisions prises par le contrôleur se font à partir de l'inspection des paquets entrants, ainsi que des statistiques collectées par la table des flots. Il sera nécessaire, notamment pour les décisions d'admission, d'obtenir des métriques indiquant le niveau de congestion des différentes files d'attente, à la fois dans les sens montant et descendant.

Ces informations pourront être envoyées périodiquement par le DSLAM (à des fins de statistiques), voire uniquement en cas de congestion. La meilleure solution reste encore à déterminer. Il conviendra à cet effet de donner une priorité absolue au trafic de signalisation afin de permettre le fonctionnement du système. Le contrôleur disposera de la classe "trafic bloqué" afin de bloquer l'admission d'un nouveau flot, ou de préempter un flot en cours.

Faisabilité

Tables de flots et signalisation

La plupart des briques nécessaires à la réalisation de la proposition Self-Protect existent depuis longtemps.

Par exemple, les équipements tournant sous le système d'exploitation Linux possèdent déjà une implémentation de table de flots au travers du système *netfilter* [114], qui permet un suivi de connexion pour la réalisation de *firewalls*. Le système est utilisé sur de nombreuses passerelles domestiques, et même sur des équipements réseau [153].

Plus récemment cependant, le protocole OpenFlow [108] a été proposé par McKeown *et al.* et a gagné suffisamment de reconnaissance de la communauté et de l'industrie pour être reconnu comme un standard et implémenté dans de nombreux équipements. OpenFlow repose sur des concepts similaires à Self-Protect en ce qu'il permet de piloter une table des flots distante en découplant le contrôleur de la gestion de la table des flots. Initialement prévu pour le routage, les dernières versions d'OpenFlow proposent un support très basique de la QoS, permettant l'affectation d'un flot à une file préexistante. Les extensions QoS d'OpenFlow pourraient s'inspirer des protocoles classiques tels que RSVP [165] ou uPNP pour décrire le profil des flots, et être éventuellement étendues pour permettre le rapatriement des statistiques sur les files d'attente nécessaire pour l'application d'un contrôle de surcharge.

Des implémentations libres de contrôleurs existent, comme NOX [68], et pourraient être étendues pour le support de nos besoins de QoS. Enfin, l'isolation entre les utilisateurs pourrait être assurée par des solutions de virtualisation de l'espace de flots, comme FlowVisor [141].

Sécurité et charge du DSLAM

La gestion d'une table de flots à des fins de QoS est typiquement moins sensible que des problèmes similaires de pare-feu. La décentralisation du contrôleur permet de déplacer sur l'équipement client les opérations coûteuses d'identification de flots et de traitement, ainsi que celles pouvant poser des problèmes de sécurité.

Reste le problème de la charge induite sur le DSLAM par la signalisation proportionnelle au nombre de flots. Il est possible d'une part de limiter ce trafic par utilisateur, et d'autre part de mettre en place des traitements par défaut permettant d'éviter une signalisation pour une grande partie des flots courts qui représentent la majorité des flots (par exemple les requêtes DNS).

Valeur ajoutée

La présence de tables de flots au sein des équipements n'est pas limitée au fonctionnement de la proposition. Elles sont déjà utilisées aujourd'hui pour l'implémentation de *firewalls*, voire par certains constructeurs pour faire du routage au niveau flot (voir Chapitre 2, section 2.4.1).

Les données recueillies sur la performance des flots permettent à un utilisateur de mieux comprendre la performance de son trafic, et de déterminer par exemple si un problème de congestion est dû à son réseau domestique, à une congestion sur le lien d'accès ou encore plus en amont dans le réseau. La mise en commun de ces données permettrait d'étendre les fonctionnalités de monitoring collaboratif qui sont actuellement développées, ou encore de faire de la détection d'attaque par DDOS (*Distributed Denial of Service*, déni de service distribué)).

Ordonnancements et stardardisation

Le plus difficile reste l'introduction d'ordonnanceurs appropriés au sein des équipements réseaux. La communauté pousse cependant pour l'adoption d'interfaces ouvertes [142], et les évolutions du côté des organismes de standardisation montrent l'intérêt des constructeurs à fournir des mécanismes de gestion du trafic sur le dernier kilomètre.

Parmi les nombreuses propositions de standardisation, nous pouvons retenir d'une part le *Broadband Forum* (anciennement *ADSL forum*) [42] et la *Home Gateway Initiative* (HGI) [73] qui proposent respectivement la standardisation des aspects réseaux du réseau d'accès, et des spécifications techniques de la passerelle domestique. Dans les deux cas, les objectifs de QoS visent à fournir une faible latence pour certains flots, et de bénéficier du multiplexage statistique.

Les deux propositions reposent fortement sur les mécanismes DiffServ pour attribuer les flots à plusieurs classes de service, et de leur associer soit une priorité, soit un poids pour un algorithme WRR, en fonction de leurs besoins. Il est prévu que la passerelle domestique et le DSLAM puisse effectuer une classification des flots dans les deux sens à partir d'un ensemble de règles stockés sur un serveur tiers, et interrogé au travers d'un protocole de signalisation approprié. Il est également prévu, mais non spécifié, que l'équipement utilisateur puisse demander un service différencié aux équipements en amont. La plupart de ces recommandations concernent le BRAS puisque les DLAMs opèrent encore souvent sur la couche 2 uniquement, mais l'évolution des réseaux d'accès pourrait déplacer le traitement vers le DSLAM. La spécification insiste enfin sur le besoin d'effectuer une protection contre la surcharge, par exemple à l'aide d'un contrôle d'admission.

7.5 Réalisation et évaluation de l'architecture Self-Protect

7.5.1 Prototype

Afin d'évaluer la faisabilité et l'efficacité de la solution, nous avons réalisé un prototype de Self-Protect au sein d'une petite plateforme expérimentale recréant les conditions d'un réseau d'accès. L'objectif de notre prototype était de se baser majoritairement sur des briques logicielles existantes en effectuant un minimum de modifications, afin de prouver qu'une telle solution est d'ores et déjà réalisable dans un réseau de production.

Architecture

La plateforme est composée de trois machines standard sous GNU/Linux, faisant respectivement office de client ADSL, de passerelle domestique et de DSLAM. Elle est illustrée sur la Figure 7.2. D'autres machines permettent la création de trafic artificiel et la diffusion de vidéos. Toutes les machines sont connectées avec une interface à 100MB/s, et le débit ADSL est simulé en forçant l'auto-négociation des cartes réseaux à 10Mb/s pour le sens descendant, et en utilisant un limiteur de débit logiciel (HTB) configuré à 1Mb/s pour le sens montant. Une passerelle domestique se comporte soit en routeur, soit comme un noeud transparent. Nous avons retenu ce dernier choix pour des raisons de simplicité (pas de NAT, d'adresses privées, etc.), ce qui ne change en rien les évaluations.

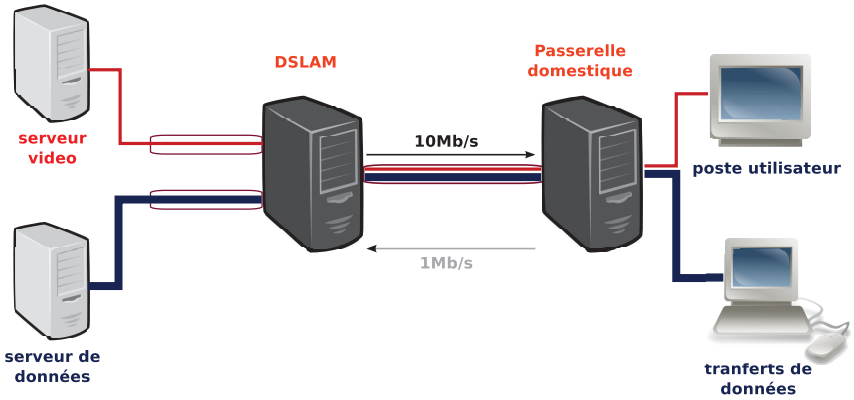


FIGURE 7.2 – Prototype Self-Protect

Classes de flots et ordonnancement

Le framework *netfilter* offre une interface permettant de contrôler le type d'ordonnancement souhaité pour les différents équipements, appelée *tc* (pour *traffic control*). Notre prototype utilise un mécanisme simple de priorités (*pfifo_fast*). Un ordonnancement *fair queueing* est ajouté pour les files de faibles et moyennes priorités afin d'assurer un partage équitable des ressources (nous avons utilisé l'algorithme *sfq*, qui est également disponible en standard).

Il suffit ainsi d'attribuer une marque (un entier) aux différents flots, et d'attribuer les paquets ainsi marqués à la file correspondante, par des règles *iptables*, de la même manière que l'on configurerait un pare-feu, au travers d'une interface spécifique (table *mangle*). Il est enfin possible d'assurer une bande passante minimale à la file la moins prioritaire grâce à un algorithme Weighted Fair Queueing (WFQ).

Contrôleur et caractérisation des flots

La passerelle domestique, le contrôleur, est responsable d'effectuer la reconnaissance des flots et leur signalisation. Les règles de caractérisation des flots pourraient être faites au niveau de l'interface *iptables* du framework *netfilter*, mais nous avons choisi de les faire au sein d'un démon tournant sur la machine pour plus de flexibilité.

L'intégration de la reconnaissance des protocoles au niveau 7 se fait au travers de l'outil *iptables*, qui supporte un grand nombre de protocoles reconnus. La démarche est d'installer les règles nécessaires permettant d'identifier les protocoles utilisés dans les filtres définis par l'utilisateur. Une marque sera alors associée au paquet, spécifiant le protocole identifié, et qu'il sera ensuite possible de réutiliser lors de l'application des règles de filtrage.

Les règles sont basées sur un ensemble de métriques maintenues par le démon en fonction du contenu ou de la taille des paquets, de leurs délais, ou de statistiques sur les flots (taille, durée, etc.). Nous utilisons également les possibilités de suivi de connexion offertes par le système. Enfin, un démon est installé sur la machine client afin de surveiller les applications lancées, et de leur associer des propriétés supplémentaires : utilisateur, nom de l'application, etc.

Tables de flots

Notre implémentation des tables de flots réutilise le système de suivi de connexions (*conntrack*) du framework *netfilter*, qui est intégré dans le noyau Linux, et utilisé dans le cadre de pare-feu ou de NAT. Le flot considéré est associé à un socket, ce qui correspond exactement à la notion de quintuplet classique. Nous avons uniquement redéfini la notion de *timeout* à quelques secondes (les spécifications usuelles sont plus longues). Il est possible d'associer une marque (*connmark*) à un flot, qui sera appliquée ensuite à chaque paquet. Le trafic de signalisation n'est pas traqué afin d'éviter une réaction infinie (grâce à la cible particulière NOTRACK).

Le système *conntrack* peut être piloté depuis n'importe quelle application grâce à la bibliothèque *libnfconntrack*, qui disponible pour de nombreux langages.

Deux démons (écrits en Python) tournent sur le DSLAM et la passerelle domestique, afin de gérer la table des flots. Le démon de la passerelle est chargé d'analyser les paquets afin d'associer le profil de trafic à chaque flot, de mettre à jour la table locale et d'envoyer un message au démon tournant sur le DSLAM. Ce dernier se contente de peupler sa table en fonction des messages reçus.

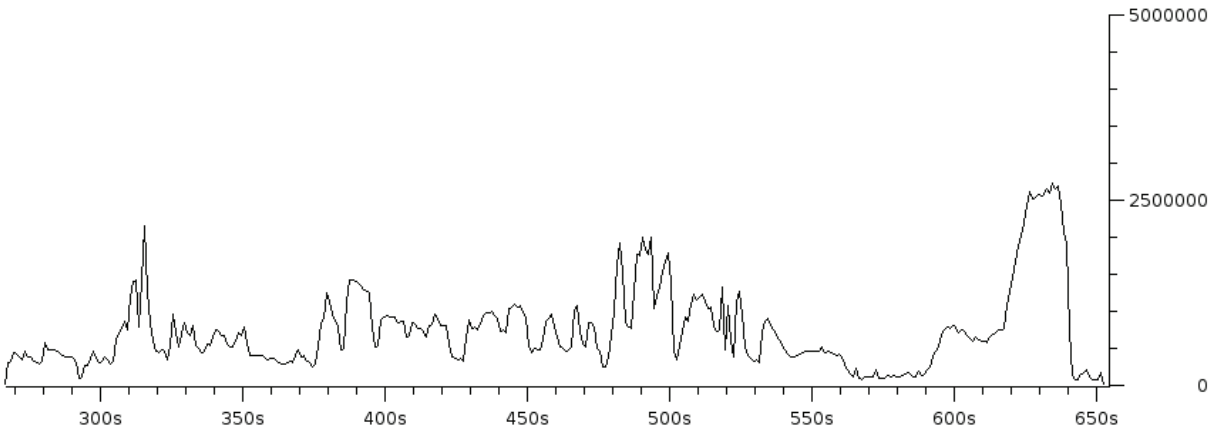


FIGURE 7.3 – Débit instantané de la vidéo en fonction du temps

Signalisation

A des fins de simplicité, nous avons implémenté notre protocole de signalisation sur XMLRPC. Chaque message contient un identifiant de l'action à faire (ajout/suppression de flot, modification de la marque, etc.), un descripteur de flot ainsi que les paramètres requis. En pratique, il conviendrait de réutiliser un protocole standard afin d'assurer l'interopérabilité des équipements.

Interface utilisateur

L'interface du prototype offre une vision des différentes classes de trafic, avec la liste des flots concernés, ainsi que leurs caractéristiques et statistiques. L'utilisateur peut promouvoir ou rétrograder un flot, le bloquer ou encore supprimer son entrée des tables de flots. L'interface permet la création et la gestion des différentes règles. Des graphes présentant la charge des différentes files d'attente sont également disponibles.

7.5.2 Évaluations

Cette section présente un ensemble de résultats expérimentaux qui ont été établis lors de démonstrations internes. Elle est volontairement courte puisque l'évaluation de la plateforme se ramène à celle des mécanismes d'ordonnancement associés (tels qu'une file prioritaire, etc.). Nous mettons en avant les choix justifiant les mécanismes introduits dans la Section 7.3, et le comparons aux solutions FIFO et FQ.

Différenciation *streaming*/élastique

Nous considérons un scénario classique où un flux vidéo à haut débit est en compétition avec un petit nombre de connexions TCP possédant également un fort débit. Nous avons choisi le film *Elephant Dream*, un court métrage d'animation libre [3], en version moyenne définition (1024x576) ; sa taille est de 446Mo. Nous avons capturé les paquets de la vidéo diffusée en UDP par le logiciel VLC, et analysé leur débit instantané avec Wireshark. Il est représenté en figure 7.3. On constate que la vidéo présente des instants avec un fort débit crête. Les connexions TCP concurrentes sont réalisées soit avec le logiciel `iperf`, soit en téléchargeant par FTP un gros fichier. Ces connexions ne sont pas limités en débit et se partagent la bande passante disponible.

Avec l'ordonnancement FIFO, la présence d'une connexion concurrente dégrade la vidéo notamment aux instants ou son débit crête est le plus élevé. En présence de 3 connexions parallèles, la vidéo n'est plus regardable.

L'utilisation de *fair queueing* permet de préserver la vidéo lorsqu'il n'y a qu'une connexion TCP (le débit équitable d'environ 5Mb/s est globalement supérieur au débit de la vidéo). Par contre, avec 4 connexions, ce débit équitable (d'environ 2Mb/s), n'est plus suffisant pour protéger la vidéo et l'on retrouve des dégradations aux instants ou son débit est plus élevé.

La priorité stricte offerte par Self-Protect permet la protection de la vidéo en toute circonstance.

3. disponible sur `http://www.elephantsdream.org`

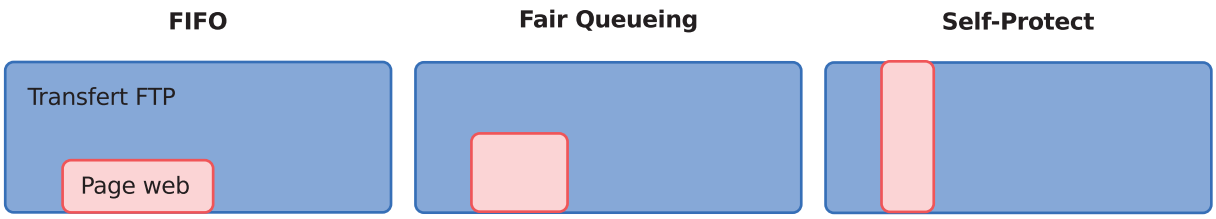


FIGURE 7.4 – Illustration du débit réalisé par le transfert web interactif en compétition avec des gros transferts de données dans les cas FIFO (à gauche), (*fair queueing*) au centre, et avec une priorité stricte (à droite).

Différenciation élastique/élastique

Afin de montrer l'intérêt de différencier les applications élastiques entre elles, nous considérons un scénario impliquant à la fois une session utilisateur de navigation Web, et des téléchargements de données à fort débit comme précédemment. Nous utilisons ici 4 connexions.

La session Web est simulée par le téléchargement d'une page web contenant de nombreuses images (une image découpée en plusieurs petits morceaux). Elle correspond typiquement à des flots dont l'interactivité ou la réactivité importent pour l'utilisateur.

Avec FIFO, l'affichage de la page est plutôt long. La présence de *fair queueing* améliore légèrement la performance, mais la différence est notable avec l'utilisation de deux files prioritaires afin de distinguer ces deux types d'applications élastiques.

L'utilisation de *fair queueing* ou de files prioritaires n'impacte pas le temps de transfert des fichiers concurrents, qui se produit sur une échelle de temps plus importante, comme illustré en figure 7.4. Les blocs représentent le volume de trafic à écouler pour chaque classe de flots. L'axe des abscisses représente le temps, et on représente en ordonnée le partage de la bande passante à chaque instant. Dans le cas FIFO (à gauche), le flux Web parvient à obtenir une part plus ou moins équitable de la bande passante, en fonction des mécanismes TCP sous-jacents. FQ (au centre), assure un partage équitable, et permet au flux Web d'obtenir la moitié des ressources. Enfin, le service en priorité effectué par Self-Protect (à droite), lui permet d'être servi immédiatement, donnant un temps de réponse maximum pour les flux importants pour l'utilisateur. On peut remarquer qu'un tel traitement n'affecte par le temps de transfert FTP, du moment que la charge des flux Web n'est pas trop importante.

Intérêt du *fair queueing* au sein d'une classe contenant des flux élastiques

Afin de mettre en évidence les apports du *fair queueing* au sein des classes les moins prioritaires recevant des flux élastiques, nous considérons les deux scénarios suivants :

connexions de RTT différents : Nous considérons deux sources de trafic avec un RTT différent. Pour cela, nous émulons un délai de propagation plus important pour l'une des deux sources à l'aide de la discipline de service `netem` disponible dans `tc`.

utilisation de protocoles TCP différents : Cette fois-ci, les RTT sont équivalents, mais l'une des deux connexions utilise un algorithme TCP avancé, comme BIC ou CUBIC, tandis que l'autre utilise la version standard TCP Reno.

Avec une discipline FIFO, la connexion ayant le RTT le plus faible ou l'algorithme TCP le plus agressif obtient un débit plus élevé. Une équité approximative est restaurée par l'utilisation de *fair queueing*.

Compétition sens montant et descendant

Ce scénario met en évidence le besoin de mettre en priorité les paquets d'acquittement (ACK) des connexions TCP qui sont dans la file la moins prioritaire. Pour cela nous générons une connexion TCP simple entre un serveur et la HGW que l'on met en priorité 3, et une connexion entre la HGW et le serveur que l'on met en priorité 2.

Avec une discipline FIFO sur la passerelle domestique, la connexion montante sature la bande passante et les ACK de la connexion descendante sont considérablement ralentis. La connexion descendante n'occupe pas toute la bande passante et obtient un débit très faible. La présence de files prioritaires bloque les ACK montants, ce qui stoppe la connexion descendante. Si on met les ACK de la connexion descendante en priorité 2 au lieu de 3, mais sans fair queuing sur la priorité 2, la file de priorité 2 contiendra les ACK de la connexion descendante et la connexion montante, on se ramène

donc au premier cas et on observe le même débit. Si enfin on met une discipline de fair queuing dans la file 2, la connexion descendante prend son débit maximum, sans pénaliser la connexion montante. Ce scénario illustre la nécessité de prioriser les ACK des connexions de priorité 3.

Surcharge

Pour terminer, nous illustrons l'importance de détecter des situations de surcharge afin de préserver la performance des applications, notamment en ce qui concerne les flux *streaming*. Nous considérons un scénario ou beaucoup trop de flux sont affectés à la classe la plus prioritaire (servie en FIFO).

Dans une telle situation, il n'est plus possible de garantir la performance des flots, et il convient de terminer ou déclasser certains flots. L'identification de la surcharge n'est pas faite par le prototype, mais il serait intéressant de détecter des situations prolongées où le trafic prioritaire excède une certaine limite, afin d'alerter l'utilisateur ou de prendre les mesures nécessaires automatiquement.

7.6 Conclusion

Dans les chapitres précédents, nous avons motivé l'introduction d'une architecture de gestion de la qualité de service au niveau flot, afin d'obtenir des garanties robustes de performance pour le trafic IP. Les modèles utilisés nous permettent le dimensionnement du réseau de cœur afin de transporter le trafic de manière transparente, et gérer les éventuelles situations de congestion et de surcharge.

Dans la plupart des déploiements, il est reconnu que le réseau d'accès est le principal goulot d'étranglement pour le trafic. Une solution telle que Cross-Protect n'est pas envisageable pour le réseau d'accès, qui possède des caractéristiques différentes en termes de capacité, de nombre d'utilisateurs, et de trafic. Nous avons proposé ici une réalisation alternative de l'architecture *Flow-Aware Networking*, dite Self-Protect, plus appropriée pour ce contexte.

Self-Protect repose sur la gestion du trafic au niveau flot, à la fois dans les sens montant et descendant, grâce à un ordonnancement approprié. Cet ordonnancement permet notamment de garantir la performance des différents flots *streaming* et élastique, envoyés et reçus par l'utilisateur (et résout les problèmes récurrents de *bufferbloat*). Les flots sont reconnus et analysés par un contrôleur situé sur l'équipement utilisateur (la passerelle domestique pour un réseau ADSL), afin d'être traités dans le sens montant. Pour le sens descendant, l'équipement (par exemple le DSLAM) maintient uniquement une table de flots, qui est gérée de manière distante par le contrôleur, au travers d'un protocole de signalisation. Il en résulte une solution distribuée, permettant sa réalisation dans un réseau d'opérateur, et satisfaisant les contraintes de neutralité des réseaux puisque les traitements effectués sont à l'initiative de l'utilisateur.

Nous avons proposé un prototype simple basé sur un petit nombre de classes de trafic, et démontré son efficacité dans des scénarios simples. Bien que suffisant pour de nombreux cas d'utilisation, il est possible d'étendre ce prototype afin de supporter des profils de trafic plus complexes, afin de satisfaire des besoins plus stricts pour des utilisateurs professionnels par exemple. Une réalisation exploitant les nouveaux mécanismes de *Software Defined Networking* (notamment *OpenFlow*) est à envisager…

Chapitre 9

Conclusion

Bien que longtemps délaissées par la communauté, on constate aujourd'hui un regain d'intérêt pour les problématiques de performance et de qualité de service dans le réseau. Les évolutions des technologies et du marché imposent de plus en plus aux opérateurs de fournir un réseau efficace, tout en limitant les investissements. Les solutions choisies doivent être adaptées aux nouvelles technologies (fibre optique, 5G, etc.), aux environnements porteurs de valeur tels que les réseaux d'entreprise ou les *datacenters*, et être compatibles avec les nouvelles problématiques de neutralité des réseaux ou les contraintes telles que *bufferbloat*.

Aspects essentiels de QoS et Flow-Aware Networking

Notre objectif tout au long de ce document a été de présenter et d'évaluer différents aspects d'une architecture simple et efficace de qualité de service, en rupture avec les approches actuelles. Le constat est que le réseau actuel FIFO/DropTail, sur lequel coexistent un petit nombre de versions de TCP, est depuis longtemps jugé insuffisant. La complexité de mise en œuvre des architectures classiques de QoS a été prohibitive au point que les opérateurs appliquent généralement des règles empiriques de surdimensionnement, sauf dans des cas particuliers de réseaux d'entreprise par exemple, qui sont mieux contrôlés.

L'architecture que nous considérons essentiellement dans cette thèse, nommée *Flow-Aware Networking* (FAN), propose de considérer le trafic au niveau des flux de données, et possède des bases théoriques solides, permettant d'établir la relation qu'il existe entre capacité, demande et performance. L'identification de plusieurs régimes de partage de bande passante (transparent, élastique et surcharge), nous permet de justifier l'introduction de mécanismes d'ordonnancement équitable (*fair queueing*), et de contrôle de surcharge (en l'occurrence, un contrôle d'admission).

L'approche au niveau flot possède ceci d'intéressant qu'elle permet d'exprimer la performance du trafic de manière plus juste et à une échelle de temps correspondant au ressenti des utilisateurs. Elle met en œuvre des résultats forts de la théorie des files d'attente (tels que l'insensibilité), qui nous permettent la caractérisation du trafic au travers de mesures simples telles que la charge moyenne et le débit crête des flots, et de caractériser de même la performance. Ces résultats sont en fait dépendants des conditions garanties par les mécanismes d'équité et de contrôle de surcharge, qui ne sont qu'approximatifs sinon dans un réseau FIFO/DropTail/TCP.

FAN permet une exploitation saine d'un réseau de télécommunications : elle fournit d'une part des modèles simples et efficaces permettant son dimensionnement ; d'autre part, ses réalisations Cross-Protect et Self-Protect offrent les mécanismes fondamentaux permettant de l'opérer de manière robuste. Ces deux aspects sont essentiels afin de fournir des garanties de performance pour le trafic IP : des délais et taux de pertes négligeables pour le trafic *streaming*, et un débit moyen suffisant pour assurer un transfert efficace pour les données élastiques.

Les différentes études réalisés s'inscrivent dans la proposition FAN, et empruntent ses modèles de trafic et conclusions permettant de guider les choix effectués, les métriques de performance retenues ainsi que la définition d'un modèle de trafic réaliste.

Dimensionnement de buffers

Les buffers sont une ressource importante qui doit être considérés dans l'étude de la performance réalisée de bout en bout par les flots. Leur dimensionnement est généralement guidé par une règle empirique remontant aux origines de TCP, qui a été récemment remise en cause par la communauté

pour des raisons techniques et pour son impact sur la qualité de service. Plusieurs propositions de réduction drastique de la taille de ces buffers ont été faites, parfois contradictoires.

Cette taille de buffer n'est malheureusement par prise en compte par les modèles fluides au niveau flot qui supposent une taille infinie. L'analyses des différents régimes de partage de bande passante, ainsi que quelques modèles simples nous permettent de réconcilier les résultats des différentes propositions de la littérature, et de suggérer un dimensionnement approprié en fonction du modèle de trafic considéré. S'il est possible de proposer de petites tailles de buffer lorsque le lien est peu chargé ou que les débits crête des flots sont relativement faibles, ce n'est plus le cas dès lors que des flots de débit élevé se partagent la bande passante du lien (régime élastique).

Les interactions au niveau paquet jouent alors un rôle important, et perturbent le contrôle de bout en bout effectué par les mécanismes TCP, basés sur une estimation imparfaite et délayée de l'état de congestion. C'est typiquement le cas dans un réseau de *datacenter*, où les flots ont un débit élevé comparativement à la capacité des liens, ou encore un réseau d'opérateur avec des débits d'accès élevés ou transportant des tunnels d'aggrégation de trafic..

Nous remarquons qu'une bonne estimation de la performance peut-être obtenue en considérant le comportement d'un seul flot TCP en compétition avec une charge donnée de trafic. Il est difficile d'obtenir une formulation simple de la dépendance du débit des flots à la taille du buffer, mais les résultats suggèrent une dépendance en racine carrée au *Bandwidth Delay Product* classique, avec deux régimes distincts de fonctionnement. Nous proposons un dimensionnement empirique de la taille des buffers, qui suggère que les propositions de réductions les plus drastiques ne sont alors pas acceptables.

Equité

En plus de besoins importants en buffer, les versions standard de TCP ont depuis longtemps été identifiées comme insuffisantes dans des environnement de *datacenter*, ou à fort *Bandwidth Delay Product*. On trouve ainsi de nombreuses propositions dans la littérature, qui ont souvent pour objectif de se comporter équitablement avec la version classique. Ce n'est souvent pas le cas en pratique, l'impact sur le trafic *streaming* est non négligeable, et le comportement du protocole dans un contexte hétérogène est difficile à prévoir. C'est le cas dans un réseau d'opérateur classique, ou d'un datacenter partagé.

En plus de permettre l'opération d'un réseau avec une performance prévisible, la mise en application d'un ordonnancement équitable (*fair queueing*, FQ) permet de rétablir l'équité éventuellement perdue entre les flots élastiques, et de protégér le trafic *streaming* grâce à ses propriétés de différenciation implicite. L'évaluation des débits et taux de perte réalisés par les versions les plus courantes de TCP montre leurs limitations dans des environnements à forte capacité ou avec de petites tailles de buffer par exemple. En présence de FQ, de simples améliorations permettant de rendre les protocoles existants suffisamment agressifs pour mieux exploiter les resources disponibles. La conception d'un algorithme plus optimal exploitant l'information implicite fournie par FQ reste un problème ouvert.

Contrôle de surcharge

L'offre de garanties de performance nécessite un contrôle de la charge offerte au réseau. Les mécanismes de TCP qui réalisent l'allocation ne permettent pas d'éviter les situations de surcharge. Il en résulte une accumulation d'un grand nombre de flots, dont le débit individuel tend vers zéro, et seulement limité par un phénomène d'impatience (au niveau applicatif ou utilisateur).

Nous nous intéressons à la réalisation d'un contrôle d'admission local préventif, dont l'objectif est de rejeter les flots dont l'admission dégraderait la performance globale du lien. Un tel contrôle d'admission est conçu pour intervenir en cas de situations de surcharge exceptionnelles, et ne se substitue pas à un dimensionnement correct des ressources du réseau. Il permet néanmoins d'être moins conservateur lors du dimensionnement (cas de pannes multiples rares par exemple), et de servir de base à des mécanismes de trafic engineering (multipath, etc.).

L'architecture Cross-Protect prévoit deux conditions d'admission, pour prévenir à la dégradation de performance des flots *streaming* et élastique. L'objectif est d'assurer que le trafic *streaming* rencontre un lien transparent, par la différenciation faite par FQ et le maintien des conditions dites du *Bufferless Multiplexing*, et qu'un débit minimal suffisant soit assuré pour les flux élastiques. Il se base respectivement sur les mesures de *priority load* et *fair rate* faites dans l'ordonnanceur. Si le contrôle sur le *fair rate* n'est pas critique, l'autre est en revanche plus délicat. Nous étendons ici les propositions faites dans l'article original de Cross-Protect afin de proposer un algorithme efficace de contrôle des flux *streaming* dans un cas de différenciation implicite, et dans le contexte réaliste où la majorité des flots ont un débit limité.

Nous montrons la pertinence des hypothèses effectuées et étudions le paramétrage de ces algorithmes, notamment l'intervalle d'échantillonnage des estimateurs. Les capacités de protection et de

différenciation de la proposition sont évaluées dans un grand nombre de scénarios, en présence de flots gigués et dans des situations non-stationnaires de *flashcrowd*.

QoS en bordure de réseau

Nous avons jusque la détaillé une solution adaptée pour un contexte de cœur de réseau ou de *datacenter*. Une composante essentielle est l'application de mécanismes de *fair queueing* garantissant l'équité entre les flots dans le réseau. Une telle proposition n'est pas envisageable dans le cas d'un réseau d'accès, en raison de la complexité des profils de trafic des utilisateurs, et du débit crête relativement élevé des flots par rapport à la capacité des liens.

Afin de complémenter la solution Cross-Protect, et de considérer l'offre de service de bout en bout pour les flots IP, nous proposons une déclinaison de l'architecture FAN pour l'accès. La première contribution est la considération d'un ordonnancement approprié, répondant aux besoins des différents types de flots et de leurs garanties de service. La seconde est une architecture, dénommée Self-Protect, qui offre à l'utilisateur la possibilité de contrôler son trafic au niveau flot, dans les sens montants et descendants.

Elle consiste en une solution décentralisée permettant l'identification des flots et de leur profil de QoS sur l'équipement utilisateur, et leur communication à l'équipement réseau en amont par un protocole de signalisation. Reposant sur la collaboration avec l'utilisateur, elle permet notamment de satisfaire les contraintes de neutralité des réseaux.

Simulation et prototypes expérimentaux

En complément des modèles de performance, nous nous sommes attachés à évaluer les résultats et les architectures proposés à la fois dans un contexte de simulation, et sur une plateforme expérimentale permettant de les démontrer visuellement et dans des conditions plus réalistes. La phase expérimentale nous a notamment permis l'évaluation de la faisabilité et de la scalabilité de la solution dans des systèmes réels.

Nouveaux challenges pour la QoS

L'architecture FAN semble ainsi une solution prometteuse pour la réalisation, le dimensionnement et l'opération des réseaux de télécommunications. Elle est notamment plus simple à mettre en œuvre que les solutions classiques, puisqu'elle repose sur une caractérisation minimale du trafic, ne nécessite pas de coordination entre différents opérateurs de réseau, et permet à partir d'un contrôle local du trafic, d'assurer des garanties de bout en bout. Elle peut en outre être déployée dans le réseau incrémentalement, et dès maintenant, puisque ses mécanismes sont à la fois faisable et extensibles.

Une fonctionnalité essentielle est l'utilisation de *fair queueing*, et de ses capacités d'isolation et de différenciation implicite. Nous avons particulièrement insisté sur le contexte d'un réseau d'opérateur Internet dans cette thèse, mais les mécanismes et les bénéfices associés sont directement applicables dans des environnements différents.

Nous avons vu qu'une telle solution permet de résoudre de manière élégante les problèmes de bufferbloat que l'on trouve dans de nombreux réseaux d'accès, sans nécessiter de mécanismes plus complexe d'AQM, etc. Dans un *datacenter*, l'utilisation de *fair queueing* pourrait se faire conjointement à l'utilisation de protocoles TCP adaptés, et permettre à la fois des débits importants pour les transferts de données, et des temps de propagation faibles pour les requêtes. L'adaptation des mécanismes de FQ à des réseaux optiques sera à considérer également, puisque des ordonnancements complexes n'y sont pas possibles.

Bibliographie

[1] In : *Traffic Control and Congestion Control in IP-based networks* (ITU-T: standard Y.1221, 2010).

[2] A.Dhamdhere et C. Dovrolis. « Open Issues in Router Buffer Sizing ». In : *ACM SIGCOMM CCR (editorial section)* (January 2006).

[3] A.Dhamdhere, C. Dovrolis et H. Jiang. « Buffer Sizing for Congested Internet Links ». In : *Proc. of INFOCOM'05* (March 2005).

[4] Mohammad Alizadeh et al. « Data center tcp (dctcp) ». In : *ACM SIGCOMM CCR* 41.4 (2011), p. 63–74.

[5] Mohammad Alizadeh et al. « pFabric: Minimal Near-optimal Datacenter Transport ». In : *Proc. of SIGCOMM'13*. Hong Kong, China, 2013, p. 435–446.

[6] Mark Allman et Ethan Blanton. « Notes on Burst Mitigation for Transport Protocols ». In : *ACM SIGCOMM CCR* 35.2 (2005), p. 60.

[7] Guido Appenzeller, Isaac Keslassy et Nick McKeown. « Sizing Router Buffers ». In : *Proceeding of ACM SIGCOMM '04* (2004).

[P5] J. Augé, S. Oueslati et J. Roberts. « Measurement-Based Admission Control for Flow-Aware Implicit Service Differentiation ». In : *23rd International Teletraffic Congress (ITC 2011)*. San Francisco, USA, 6 sept. 2011.

[P4] J. Augé et J. Roberts. « A Statistical Bandwidth Sharing Perspective on Buffer Sizing. » In : *International Teletraffic Congress*. Sous la dir. de Lorne Mason, Tadeusz Drwiega et James Yan. T. 4516. Lecture Notes in Computer Science. Springer, 10 sept. 2007, p. 410–421. isbn : 978-3-540-72989-1. url : `http://dblp.uni-trier.de/db/conf/teletraffic/itc2007.html#AugeR07`.

[P3] J. Augé et J. Roberts. « Buffer sizing for elastic traffic ». In : *Next Generation Internet Design and Engineering, 2006. NGI'06. 2006 2nd Conference on*. IEEE. 2006, 8–pp.

[P2] J. Augé et J. Roberts. « Performance of TCP over a link using Fair Queueing ». In : *EuroNGI, Proceedings of Workshop on QoS and Traffic Control* (déc. 2005).

[8] Jordan Augé et al. *Maquette de démonstration Cross-Protect*. Rapp. tech. 2005.

[9] KE Avrachenkov et al. « The effect of router buffer size on the TCP performance ». In : *In Proc. of the LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*. 2001.

[10] K. Avrachenkov et al. « Differentiation between short and long TCP flows: Predictability of the response time ». In : *IEEE INFOCOM*. T. 2. 2004, p. 762–773.

[11] N. Bansal et M. Harchol-Balter. « Analysis of SRPT scheduling: Investigating unfairness ». In : *Proc. of SIGMETRICS'01*. 2001, p. 290.

[12] Dhiman Barman, Georgios Smaragdakis et Ibrahim Matta. « The effect of router buffer size on highspeed TCP performance ». In : *Proc. of GLOBECOM'04*. T. 3. 2004, p. 1617–1621.

[13] N.G. Bean. *Estimation and Control in ATM Network*. Rapp. tech. 1994.

[14] *Measurement-based Admission Control for Elastic Traffic*. Salvador, Brazil, déc. 2001.

[15] Slim Ben Fredj et al. « Statistical bandwidth sharing: a study of congestion at flow level ». In : *Proc. of SIGCOMM'01* (2001).

[16] Nabil Benameur, Fabrice Guillemin et Luca Muscariello. « Latency Reduction in Home Access Gateways with Shortest Queue First ». In : *Proc. ISOC Workshop on Reducing Internet Latency*. London, UK, 2013.

[17] Nabil Benameur et al. « Integrated admission control for streaming and elastic traffic ». In : *Quality of Future Internet Services*. Coimbra, Portugal, sept. 2001, p. 69–81.

[18] Nabil Benameur et al. « Quality of service and flow level admission control in the Internet* 1 ». In : *Computer Networks* 40.1 (2002), p. 57–71.

[19] Nabil Benameur et al. « Quality of service and flow-aware admission control in the Internet ». In : *Computer Networks* 40 (2002), p. 57–71.

[20] Nabil Benameur et al. « Selective service protection in overload; differentiated services or per-flow admission control? » In : *Proc. of Networks'04*. Vienna, Austria, juin 2004, p. 217.

[21] H. van den Berg et M.Mandjes. « Measurement-based admission control: algorithms, evaluation, and research perspectives ». In : *COST 257* (1999).

[22] D.P. Bertsekas et R. Gallager. *Data networks*. 1987.

[23] T Bonald et L Muscariello. « Shortest queue first: implicit service differentiation through per-flow scheduling ». In : *Demo at IEEE LCN* (2009).

[24] T Bonald, S Oueslati-Boulahia et J Roberts. « IP traffic and QoS control: the need for a flow-aware architecture ». In : *World Telecommunications Congress*. 2002.

[25] T. Bonald et J. Roberts. « Scheduling network traffic ». In : *ACM SIGMETRICS Performance Evaluation Review* 34.4 (2007), p. 35.

[26] Thomas Bonald. « Throughput performance in networks with linear capacity constraints ». In : *Proc. of CISS'06* (20006).

[27] Thomas Bonald, Matthieu Jonckheere et Alexandre Proutiere. « Insensitive load balancing ». In : *Proc. of SIGMETRICS'04*. 2004, p. 367–377.

[28] Thomas Bonald et Laurent Massoulié. « Impact of fairness on Internet performance ». In : *Proc. of SIGMETRICS'01*. 2001, p. 91.

[29] Thomas Bonald, Luca Muscariello et Norberto Ostallo. « Self-prioritization of audio and video traffic ». In : *Proc. of ICC'11*. 2011, p. 1–6.

[30] Thomas Bonald et Alexandre Proutiere. « Insensitive bandwidth sharing in data networks ». In : *Queueing systems* 44.1 (2003), p. 69–100.

[31] Thomas Bonald et Alexandre Proutiere. « On performance bounds for balanced fairness ». In : *Performance Evaluation* 55 (2004), p. 25–50.

[32] Thomas Bonald et Alexandre Proutière. « Insensitivity in processor-sharing networks ». In : *Proc. of Performance* (2002).

[33] Thomas Bonald, Alexandre Proutière et James Roberts. « Statistical performance guarantees for streaming flows using expedited forwarding ». In : *Proc. of INFOCOM'01*. T. 2. 2001, p. 1104–1112.

[34] Thomas Bonald et James W Roberts. « Congestion at flow level and the impact of user behaviour ». In : *Computer Networks* 42.4 (2003), p. 521–536.

[35] B. Braden et al. « Recommendations on Queue Management and Congestion Avoidance in the Internet ». In : *RFC 2309* (avr. 1998).

[36] R. Braden et al. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. RFC 2205. Internet Engineering Task Force, sept. 1997. url : `http://www.rfc-editor.org/rfc/rfc2205.txt`.

[37] L.S. Brakmo. « TCP Vegas: End to end congestion avoidance on a global Internet ». In : *IEEE Journal on selected Areas in Communications* 13.8 (1995), p. 1465.

[38] L. Breslau, S. Jamin et S. Shenker. « Comments on the Performance of Measurement-Based Admission Control Algorithms ». In : Tel Aviv, Israel, mar. 2000.

[39] Bob Briscoe. « Flow Rate Fairness: Dismantling a Religion ». In : *ACM SIGCOMM CCR* 37.2 (avr. 2007), p. 63–74.

[40] Bob Briscoe. « Flow rate fairness: Dismantling a religion ». In : *ACM SIGCOMM CCR* 37.2 (2007), p. 63–74.

[41] B. Briscoe et al. « Pre-congestion notification marking ». In : *IETF Internet draft* draft-briscoe-tsvwg-cl-phb-03 (2006).

[42] *Broadband Forum*. http://www.broadband-forum.org/.

[43] Claudio Casetti, Jim Kurose et Don Towsley. « An adaptive algorithm for measurement-based admission control in integrated services packet networks ». In : *Proc. of the Int. Workshop on Protocols for High-Speed Networks* (1996).

[44] A. Charny et al. « Pre-congestion notification using single marking for admission and preemption ». In : *draft-charny-pcn-single-marking-00 (work in progress)* (2006).

[45] David Clark. « The design philosophy of the DARPA Internet protocols ». In : *ACM SIGCOMM CCR*. T. 18. 4. 1988, p. 106–114.

[46] F. Delcoigne, A. Proutiere et G. Régnié. « Modeling integration of streaming and data traffic ». In : *Performance Evaluation* 55 (2004), p. 185–209.

[47] N. Dukkipati et al. « Processor sharing flows in the internet ». In : *Proc. of IWQoS'05* (2005), p. 271–285.

[48] Eric Dumazet. *Fair Queue packet scheduler*. 2013.

[49] M. Enachescu et al. « Part III: routers with very small buffers ». In : *ACM SIGCOMM CCR, v.35 n.3* (July 2005).

[50] M. Enachescu et al. « Routers with very small buffers ». In : *Proc. of INFOCOM'06* ().

[51] W. Feng et al. « The BLUE active queue management algorithms ». In : *IEEE/ACM Transactions on Networking* 10.4 (2002), p. 528.

[52] S. Floyd. « HighSpeed TCP for Large Congestion Windows ». In : *RFC 3649, Experimental, December 2003* ().

[53] S. Floyd et K. Fall. *Router mechanisms to support end-to-end congestion control*. 1997.

[54] S. Floyd et V. Jacobson. « Link-sharing and Resource Management Models for Packet Networks ». In : *IEEE/ACM Trans, on Networking* 3.4 (août 1995), p. 365–386.

[55] Sally Floyd. « Comments on measurement-based admissions control for controlled-load services ». In : (1996).

[56] Sally Floyd. « TCP and explicit congestion notification ». In : *ACM SIGCOMM CCR* 24.5 (1994), p. 8–23.

[57] Sally Floyd et Van Jacobson. *Random Early Detection Gateways for Congestion Avoidance. 1 (4): 397–413*. 1993.

[58] S. Floyd, R. Gummadi, S. Shenker et al. « Adaptive RED: An algorithm for increasing the robustness of RED's active queue management ». In : *Preprint, available at http://www. icir. org/floyd/papers. html* (2001).

[59] Lixin Gao. « On inferring autonomous system relationships in the Internet ». In : *IEEE/ACM Transactions on Networking (Transactions on Networking)* 9.6 (2001), p. 733–745.

[60] Jim Gettys et Kathleen Nichols. « Bufferbloat: Dark buffers in the internet ». In : *Queue* 9.11 (2011), p. 40.

[61] Richard J Gibbens et Frank P Kelly. « Measurement-based connection admission control ». In : *15th International Teletraffic Congress*. T. 2. 1997, p. 879–888.

[62] Richard J. Gibbens, Frank P. Kelly et Peter B. Key. « A decision-theoretic approach to call admission control in ATM networks ». In : *JSAC* 13.6 (1995), p. 1101–1114.

[63] Y Gong et al. « Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control ». In : *Computer Networks* (2014).

[64] Luigi Alfredo Grieco et Saverio Mascolo. « TCP Westwood and easy red to improve fairness in high-speed networks ». In : *Protocols for High Speed Networks*. Springer. 2002, p. 130–146.

[65] Matthias Grossglauser et David NC Tse. « A framework for robust measurement-based admission control ». In : *IEEE/ACM Transactions on Networking* 7.3 (1999), p. 293–309.

[66] Matthias Grossglauser et David NC Tse. « A time-scale decomposition approach to measurement-based admission control ». In : *IEEE/ACM Transactions on Networking* 11.4 (2003), p. 550–563.

[67] D. Grossman. *New Terminology and Clarifications for Diffserv*. RFC 3260. Internet Engineering Task Force, avr. 2002. url : `http://www.rfc-editor.org/rfc/rfc3260.txt`.

[68] Natasha Gude et al. « NOX: towards an operating system for networks ». In : *ACM SIGCOMM CCR* 38.3 (2008), p. 105–110.

[69] M. Harchol-Balter et al. « Size-based scheduling to improve web performance ». In : *ACM Transactions on Computer Systems (TOCS)* 21.2 (2003), p. 207–233.

[70] T. Hoeiland-Joergensen et al. *FlowQueue-Codel*. 2014.

[71] Christopher V Hollot et al. « Analysis and design of controllers for AQM routers supporting TCP flows ». In : *IEEE Transactions on Automatic Control* 47.6 (2002), p. 945–959.

[72] C.V. Hollot et al. « On designing improved controllers for AQM routers supporting TCP flows ». In : *INFOCOM'01*. T. 3. PI - Proportional integral. 2001, p. 1726–1734.

[73] *Home Gateway Initiative*. http://www.homegatewayinitiative.org/.

[74] Bert Hubert. *The Wonder Shaper*. http://lartc.org/wondershaper/.

[75] V. Jacobson et RT Braden. « Congestion control and avoidance ». In : *Proc. ACM SigComm*. T. 88. 1988, p. 314–329.

[76] Sugih Jamin, Scott J Shenker et Peter B Danzig. « Comparison of measurement-based admission control algorithms for controlled-load service ». In : *Prof. of INFOCOM'97*. T. 3. 1997, p. 973–980.

[77] Hao Jiang et Constantinos Dovrolis. « Why is the internet traffic bursty in short time scales? » In : *ACM SIGMETRICS Performance Evaluation Review*. T. 33. 1. 2005, p. 241–252.

[78] C. Jin, D.X. Wei et S.H. Low. « Fast TCP: Motivation, architecture, algorithms, performance ». In : *Proc. of INFOCOM-06* 14.6 (2006), p. 1259.

[79] Jinoo Joung, Jongtae Song et Soon Seok Lee. « Flow-based QoS management architectures for the next generation network ». In : *ETRI journal* 30.2 (2008), p. 238.

[80] A.E. Kamal. « A Discrete-Time Model of TCP Reno with Background Traffic Interference ». In : *mascots*. Published by the IEEE Computer Society. 2002, p. 0445.

[81] Dina Katabi, Mark Handley et Charlie Rohrs. « Congestion control for high bandwidth-delay product networks ». In : *ACM SIGCOMM CCR*. T. 32. 4. 2002, p. 89–102.

[82] Jasleen Kaur et Harrick M Vin. « Core-stateless guaranteed rate scheduling algorithms ». In : *Proc. of INFOCOM'01*. T. 3. 2001, p. 1484–1492.

[83] F.P. Kelly. « Models for a self-managed Internet ». In : *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* (2000), p. 2335–2348.

[84] F.P. Kelly, A.K. Maulloo et D.K.H. Tan. « Rate control for communication networks: shadow prices, proportional fairness and stability ». In : *Journal of the Operational Research society* 49.3 (1998), p. 237–252.

[85] Tom Kelly. « Scalable TCP: Improving performance in highspeed wide area networks ». In : *ACM SIGCOMM CCR* 33.2 (2003), p. 83–91.

[86] S. Keshav. « Congestion Control in Computer Networks ». In : *PhD Thesis, published as UC Berkeley TR-654* (1991).

[87] Srinivasan Keshav. « Packet-pair flow control ». In : *IEEE/ACM Transactions on Networking* (1995).

[88] P. Key et al. « A network flow model for mixtures of file transfers and streaming traffic ». In : *Proc. of ITC'18*. 2003.

[89] P. Key et al. « Fair Internet traffic integration: network flow models and analysis ». In : *Annals of Telecommunications* 59.11 (2004), p. 1338–1352.

[90] Leonard Kleinrock. « Queueing systems, volume II: Computer applications ». In : (1976).

[91] A. Kortebi, S. Oueslati et J. Roberts. « Cross-protect: implicit service differentiation and admission control ». In : *Proc. of HPSR'04* (2004).

[92] A. Kortebi, S. Oueslati et J. Roberts. « Implicit service differentiation using Deficit Round Robin ». In : *Proc. of ITC 19* (2005).

[93] A. Kortebi et al. « Evaluating the Number of Active Flows in a Schedular Realizing Fair Statistical Bandwidth Sharing ». In : *Proc. of SIGMETRICS'05* (juin 2005).

[94] A. Kortebi et al. « Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing ». In : *ACM SIGMETRICS Performance Evaluation Review* 33.1 (2005), p. 228.

[95] A. Kortebi et al. « On the scalability of fair queueing ». In : *Proc. of HotNets-III* (November 2004).

[96] Tian Lan et al. « An Axiomatic Theory of Fairness in Network Resource Allocation ». In : mar. 2010, p. 1–9.

[97] DE Lapsley et SH Low. « Random exponential marking for internet congestion control ». In : *Proc. of IEEE Globecom*. 1999, p. 66–74.

[98] W.E. Leland et al. « On the self-similar nature of Ethernet traffic (extended version) ». In : *IEEE/ACM Transactions on Networking (ToN)* 2.1 (1994), p. 15.

[99] J-S Li et C-S Mao. « Providing flow-based proportional differentiated services in class-based DiffServ routers ». In : *IEE Proc.-Communications* 151.1 (2004), p. 82–88.

[100] Y-T. Li, D. Leith et R.N. Shorten. « Experimental Evaluation of TCP Protocols for High-Speed Networks ». In : *Hamilton Institute, NUI Maynooth* (2005).

[101] D. Lin et R. Morris. « Dynamics of Early Random Detection ». In : *ACM SIGCOMM*. T. 97. 1997.

[102] *Linux Advanced Routing & Traffic Control*. http://www.lartc.org/.

[103] J. Liu et al. « Flow-level stability of data networks with non-convex and time-varying rate regions ». In : *Proc. of SIGMETRICS'07*. 2007, p. 250.

[104] Laurent Massoulié et J Roberts. « Arguments in favour of admission control for TCP flows ». In : *In: Proc. ITC 16: Teletraffic engineering in a competitive*. 1999.

[105] S. Mccanne, S. Floyd et K. Fall. *ns2 (network simulator 2)*. http://www-nrg.ee.lbl.gov/ns/. url : `http://www-nrg.ee.lbl.gov/ns`.

[106] Steven McCanne, Sally Floyd et Kevin Fall. « The LBNL network simulator ». In : *Software on-line: http://www. isi. edu/nsnam* (1997).

[107] Paul E McKenney. « Stochastic fairness queueing ». In : *INFOCOM'90*. 1990, p. 733–740.

[108] Nick McKeown et al. « OpenFlow: enabling innovation in campus networks ». In : *ACM SIGCOMM CCR* 38.2 (2008), p. 69–74.

[109] A. Medina, M. Allman et S. Floyd. « Measuring the evolution of transport protocols in the Internet ». In : *ACM SIGCOMM CCR* 35.2 (2005), p. 52.

[110] J. Mo et J. Walrand. « Fair end-to-end window-based congestion control ». In : *IEEE/ACM Transactions on Networking (Transactions on Networking)* 8.5 (2000), p. 556–567.

[111] R. Morris. « Scalable TCP Congestion Control ». In : *Proc. of INFOCOM'00*. T. 3. Mar. 2000, p. 1176–1183.

[112] D. Mustill et J. Adams. « Conditionally Dedicated Bandwidth Transfer Capability ». In : *proposed text for 100 Y.1221, ITU-T Q. 16/12* (oct. 2005).

[113] D. Mustill et J. Adams. « IP QoS Signaling Concepts ». In : *ITU-T Q. 16/12 document* (mai 2005).

[114] *Netfilter*. http://netfilter.org/.

[115] Kathleen Nichols et Van Jacobson. « Controlling queue delay ». In : *Communications of the ACM* 55.7 (2012), p. 42–50.

[116] S. Nunez-Queija, H. vd Berg et M. Mandjes. « Strategies for integration of elastic and stream traffic: a performance evaluation ». In : *Proc. of ITC'16*.

[117] T.J. Ott, TV Lakshman et L.H. Wong. « Sred: stabilized red ». In : *INFOCOM'99*. T. 3. 1999, p. 1346–1355.

[118] S. Oueslati et J. Roberts. « A new direction for quality of service: Flow aware networking ». In : *Proc. of NGI'05* (2005).

[119] S. Oueslati, J. Roberts et F.T. R&D. « Comparing flow-aware and flow-oblivious adaptive routing ». In : *Proc. of ISS'06*. 2006, p. 655–660.

[120] Jitendra Padhye et al. « Modeling TCP throughput: A simple model and its empirical validation ». In : *ACM SIGCOMM CCR*. T. 28. 4. 1998, p. 303–314.

[121] R. Pan, B. Prabhakar et K. Psounis. « CHOKe-A stateless queue management scheme for approximating fair bandwidth allocation ». In : *Proc. of INFOCOM'00*. 2000, p. 942–951.

[122] Rong Pan et al. « Approximate fairness through differential dropping ». In : *ACM SIGCOMM CCR* 33.2 (2003), p. 23–39.

[123] Rong Pan et al. « Pie: A lightweight control scheme to address the bufferbloat problem ». In : *Proc. of HPSR'13*. 2013, p. 148–155.

[124] K. Park, G. Kim et M. Crovella. « On the relationship between file sizes, transport protocols, and self-similar network tra c ». In : *Computer* (1996).

[125] V. Paxson et S. Floyd. « Difficulties in simulating the internet ». In : *IEEE/ACM Transactions on Networking* 9.4 (2001), p. 392–403.

[126] V. Paxson et S. Floyd. « Wide area traffic: the failure of Poisson modeling ». In : *IEEE/ACM Transactions on Networking (TON)* 3.3 (1995), p. 244.

[127] Jingyu Qiu et Edward W Knightly. « Measurement-based admission control with aggregate traffic envelopes ». In : *IEEE/ACM Transactions on Networking (TON)* 9.2 (2001), p. 199–210.

[128] G. Raina, D. Towsley et D. Wischik. « Part II: control theory for buffer sizing ». In : *ACM SIGCOMM CCR, v.35 n.3* (July 2005).

[129] G. Raina et D. Wischik. « Buffer sizes for large multiplexers: TCP queueing theory and instability analysis ». In : *Proc. of NGI'05, Rome* (April 2005).

[130] K. Ramakrishnan, S. Floyd et D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard). Internet Engineering Task Force, sept. 2001. url : `http://www.ietf.org/rfc/rfc3168.txt`.

[131] J. Roberts. « Realizing quality of service guarantees in multiservice networks ». In : *Proc. of IFIP Seminar PMCCN*. T. 97. 1998.

[P1] J. Roberts et J. Augé. « Fair Queueing and Congestion Control ». In : *Workshop on Congestion Control* (sept. 2005).

[132] J. Roberts et L. Massoulié. « Bandwidth sharing and admission control for elastic traffic ». In : Yokohama, oct. 1998.

[133] Jim W. Roberts et Laurent Massoulié. « Bandwidth sharing and admission control for elastic traffic ». In : *ITC Specialist Seminar*. 1998, p. 185–201. url : `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.9812%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf`.

[134] J.W. Roberts. « A survey on statistical bandwidth sharing ». In : *Computer Networks* 45.3 (2004), p. 319–332.

[135] JW Roberts. « Quality of service guarantees and charging in multiservice networks, to appear in IEICE Trans ». In : *Communications* (1998).

[136] JW Roberts et L. Massoulie. « Bandwidth sharing: objectives and algorithms ». In : *INFOCOM'99*. 1999.

[137] JW Roberts et L. Massoulié. « Arguments in favour of admission control for TCP flows ». In : *Proc. of ITC 16th*. 1999.

[138] JW Roberts, U. Mocci et J. Virtamo. « Methods for the performance evaluation and design of broadband multiservice networks ». In : *Published by the Commission of the European Communities, Information Technologies and Sciences, COST* 242 (1996).

[139] JW Roberts et S. Oueslati-Boulahia. « Quality of service by flow-aware networking ». In : *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* (2000), p. 2197–2207.

[140] S. Scott. « Fundamental design issues for the future internet ». In : *JSAC* 13.7 (1995), p. 1176–1188.

[141] Rob Sherwood et al. « Flowvisor: A network virtualization layer ». In : *OpenFlow Switch Consortium, Tech. Rep* (2009).

[142] Anirudh Sivaraman et al. « No silver bullet: extending SDN to the data plane ». In : *Proc. of Hotnets'13*. 2013, p. 19.

[143] AJ Smith et al. « Use of the Cell Loss Priority Tagging Mechanism in ATM Switches ». In : *Proc ICIE* 91 (1991).

[144] K. Srisankar et R. Srikant. « Analysis and design of an adaptive virtual queue(AVQ) algorithm for active queue management ». In : *Applications, Technologies, Architectures, and Protocols for Computer Communication*. 2001.

[145] Ion Stoica, Scott Shenker et Hui Zhang. « Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks ». In : *IEEE/ACM Transactions on Networking (TON)* 11.1 (2003), p. 33–46.

[146] B. Suter et al. « Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queuing ». In : *IEEE Journal in Selected Areas in Communications* (août 1999).

[147] B. Suter et al. « Design considerations for supporting TCP with per-flow queuing ». In : *Proc. of INFOCOMM '98* (Apr. 1998).

[148] Gajendra Hari Prakash Theagarajan, Sivakumar Ravichandran et Vijay Sivaraman. « An experimental study of router buffer sizing for mixed TCP and real-time traffic ». In : *Proc. of ICON'06*. T. 1. 2006, p. 1–6.

[149] C. Villamizar et C. Song. « High performance TCP in ANSNET ». In : *ACM SIGCOMM CCR, 24(5):45-60* (oct. 1994).

[150] Arun Vishwanath et Vijay Sivaraman. « Routers with very small buffers: Anomalous loss performance for mixed real-time and TCP traffic ». In : *Proc. of IWQoS'08*. 2008, p. 80–89.

[151] Arun Vishwanath, Vijay Sivaraman et George N Rouskas. « Anomalous loss performance for mixed real-time and TCP traffic in routers with very small buffers ». In : *IEEE/ACM Transactions on Networking* 19.4 (2011), p. 933–946.

[152] D Wei et al. « TCP pacing revisited ». In : *Proc. of IEEE INFOCOM'06*. 2006.

[153] Wikipedia. *Carrier Grade Linux — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-February-2014]. 2013.

[154] Wikipedia. *Class-based queueing — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2014]. 2013.

[155] Wikipedia. *Deficit round robin — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2014]. 2014.

[156] Wikipedia. *Fair queuing — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2014]. 2013.

[157] Wikipedia. *Weighted round robin — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2014]. 2013.

[158] Wikipédia. *Asynchronous Transfer Mode — Wikipédia, l'encyclopédie libre*. [En ligne; Page disponible le 29-janvier-2014]. 2013. url : %5Curl%7Bhttp://fr.wikipedia.org/w/index.php?title=Asynchronous_Transfer_Mode&oldid=97359358%7D.

[159] Wikipédia. *Explicit Congestion Notification — Wikipédia, l'encyclopédie libre*. [En ligne; Page disponible le 30-janvier-2014]. 2013. url : %5Curl%7Bhttp://fr.wikipedia.org/w/index.php?title=Explicit_Congestion_Notification&oldid=90350071%7D.

[160] Wikipédia. *Round-robin (informatique) — Wikipédia, l'encyclopédie libre*. [En ligne; Page disponible le 29-janvier-2014]. 2013. url : %5Curl%7Bhttp://fr.wikipedia.org/w/index.php?title=Round-robin_(informatique)&oldid=98884396%7D.

[161] Wikipédia. *Seau à jetons — Wikipédia, l'encyclopédie libre*. [En ligne; Page disponible le 30-janvier-2014]. 2013. url : %5Curl%7Bhttp://fr.wikipedia.org/w/index.php?title=Seau_%C3%83%C2%A0_jetons&oldid=91725914%7D.

[162] D. Wischik. « Buffer Requirements for High-Speed Routers ». In : *Proc. of ECOC'05* 5 (2005), p. 23–26.

[163] D. Wischik et N. McKeown. « Part I: buffer sizes for core router ». In : *ACM SIGCOMM CCR, v.35 n.3* (July 2005).

[164] Robert Wójcik et Andrzej Jajszczyk. « Flow oriented approaches to QoS assurance ». In : *ACM Computing Surveys (CSUR)* 44.1 (2012), p. 5.

[165] J. Wroclawski. *The Use of RSVP with IETF Integrated Services*. RFC 2210. Internet Engineering Task Force, sept. 1997. url : http://www.rfc-editor.org/rfc/rfc2210.txt.