

Table des matières

1	Introduction	5
1.1	La place de la QoS dans les réseaux IP	6
1.1.1	Architecture du réseau Internet	6
1.1.2	Routage des flux de données	6
1.1.3	Protocoles réseaux et QoS	7
1.1.4	Réseaux de cœur, de collecte, d'accès	7
1.1.5	Enjeux et challenges pour un opérateur	8
1.2	L'état de la QoS dans le réseau Internet actuel	8
1.2.1	Un réseau <i>best effort</i> surdimensionné	8
1.2.2	Comportement et performance de TCP	9
1.3	Contenu et contributions de la thèse	10
2	Challenges pour fournir une Qualité de Service dans les réseaux IP	11
2.1	Introduction	12
2.2	Offrir des garanties de service au trafic IP	12
2.2.1	Granularité du trafic IP	12
2.2.2	2 classes d'applications : S et E	13
2.2.3	Challenges - Compromis	13
2.3	Mécanismes de QoS dans les réseaux actuels	14
2.3.1	Le rôle de TCP et UDP dans la gestion du trafic	14
2.3.2	Comportement et performance de TCP avec des liens FIFO DropTail	14
2.3.3	Problèmes de TCP	15
2.4	Régimes opérationnels des liens	15
2.5	Techniques d'amélioration de la QoS	17
2.5.1	Alternatives au contrôle de congestion de bout en bout	17
2.5.2	Contrôle de la surcharge	19
2.5.3	Équité et isolation entre les flots	20
2.5.4	Limitation de la charge des liens par contrôle d'admission	20
2.5.5	Différenciation	21
2.6	Architectures de Qualité de Service	22
2.6.1	Intserv/Diffserv	22
2.6.2	Core Stateless Fair Queueing	23
2.6.3	Contrôle de flot dans Diffserv	23
2.6.4	Conditionally Dedicated Bandwidth	24
2.6.5	Flow State Aware	24
2.6.6	Congestion Exposure	25
2.6.7	Flow-Aware Networking	26
2.6.8	Quasi-Stateful Interv	26
2.6.9	Stateful Diffserv	26
2.6.10	Flow-aggregate-based services (FAbS)	26

2.6.11	Constraint Based Routing	26
2.6.12	IPX, operator managed internet	26
2.7	Discussion	26
3	Flow-Aware Networking (FAN)	29
3.1	Introduction	30
3.2	Modélisation du trafic streaming	31
3.2.1	Cas d'un lien isolé : multiplexage sans buffer	31
3.2.2	Contrôle d'admission pour les flots streaming	31
3.2.3	Garanties de performance de bout en bout	32
3.3	Modélisation du trafic élastique	32
3.3.1	Modèle de sessions Poisson	32
3.3.2	Cas d'un lien isolé : modèle à processeur partagé	33
3.3.3	Contrôle d'admission pour les flots élastiques	34
3.3.4	Extension à un réseau de données	35
3.4	Intégration des flots <i>streaming</i> et élastiques	37
3.5	L'architecture de différentiation implicite de service Cross-Protect	38
3.5.1	PFQ : un algorithme de FQ extensible	38
3.5.2	Indicateurs de congestion	39
3.5.3	Measurement-Based Admission Control	40
3.6	Vers la réalisation d'un routeur Cross-Protect	41
3.6.1	Evaluation de l'architecture en simulation	41
3.6.2	Evaluation de l'architecture par expérimentation	41
4	Dimensionnement des buffers pour les routeurs IP de cœur de réseau	43
4.1	Introduction	44
4.2	Dimensionnement des buffers et TCP	46
4.2.1	Dimensionnement pour 1 flot TCP	46
4.2.2	Synchronisation des flots TCP : extension à N flots	46
4.3	Etat de l'art sur le dimensionnement de buffers	47
4.3.1	Dimensionnement proportionnel au nombre de flots	47
4.3.2	Vers une réduction drastique de la taille des buffers	47
4.3.3	Vers une distinction flots <i>bottlenecked</i> / <i>non-bottlenecked</i>	48
4.3.4	Cas où les flots sont tous <i>non-bottlenecked</i>	49
4.4	Impact de la taille du buffer sur la performance des flots	49
4.4.1	Importance du modèle de trafic pour évaluer la performance des flots	49
4.4.2	Performance des flots <i>streaming</i>	50
4.4.3	Performance des flots élastiques	50
4.5	Dimensionnement des buffers dans le régime transparent	51
4.5.1	Arrivées localement Poisson	51
4.5.2	Calcul de la taille requise du buffer	52
4.6	Dimensionnement des buffers pour le régime élastique	53
4.6.1	Comportement des flots <i>bottlenecked</i>	53
4.6.2	Flots <i>bottlenecked</i> au débit crête non limité	57
4.6.3	Trafic <i>bottlenecked</i> avec des flots à débit crête limité	61
4.7	Conclusions	62
5	Partage de bande passante étendu : nouveaux protocoles TCP, fair queueing	65
5.1	Introduction	66
5.2	TODO	66
5.2.1	Paced TCP	67
5.3	Introduction de nouveaux protocoles TCP	68
5.3.1	Faiblesses de TCP Reno pour les transferts à haut débit	68

TABLE DES MATIÈRES

5.3.2	Impact de l'utilisation des nouveaux protocoles TCP sur la taille des buffers	68
5.4	Enhanced bandwidth sharing	70
5.4.1	New high speed protocols	70
5.4.2	Using fair queuing	71
5.5	Conclusions	72
6	Un algorithme de MBAC pour Cross-Protect	73
6.1	Introduction	75
6.2	Etat de l'art des approches de MBAC	75
6.2.1	Measured sum	76
6.2.2	Borne de Hoeffding	76
6.2.3	Calcul d'enveloppes de trafic	77
6.2.4	Approche basée sur la théorie de la décision	77
6.2.5	Théorie de la bande passante équivalente	77
6.2.6	Stratégies de <i>back off</i>	78
6.2.7	Décomposition selon différentes échelles de temps	78
6.3	Implémentation et évaluation de l'algorithme de Grossglauser et Tse	80
6.4	Un algorithme de MBAC pour Cross-Protect	82
6.4.1	Moyenne et variance du <i>priority load</i>	82
6.4.2	Approximation basée sur une hypothèse Poissonnienne	82
6.4.3	Performance du multiplexage statistique	83
6.4.4	Intégration de la variance pour améliorer l'algorithme	83
6.4.5	Cas des flots de débit crête supérieur	83
6.4.6	Influence de la taille du slot sur les mesures	84
6.4.7	Instabilité du <i>priority load</i> mesuré	85
6.4.8	Limite du nombre d'arrivées par intervalle	85
6.5	Evaluation des algorithmes	86
6.5.1	Modèle de trafic	86
6.5.2	Environnement de simulation	86
6.5.3	Critères de performance	87
6.5.4	Utilisation et probabilité de débordement	88
6.5.5	Prédictibilité du MBAC XP	89
6.5.6	Performance avec un mélange de débits crête	91
6.5.7	Impact de la gigue	96
6.5.8	Contrôle d'admission en régime non-stationnaire	102
6.6	Conclusions	105
6.7	Additional results	106
6.7.1	Influence des paramètres α et β	106
6.7.2	Influence du taux d'activité des connexions	107
6.7.3	Influence de la durée des connexions	108
7	Analyse d'une trace de trafic	109
7.1	Topologie et paramètres réseau	109
7.2	Traitement des traces	109
7.2.1	Traitement du trafic élastique	109
7.2.2	Traitement du trafic streaming	110
7.3	Motivations	110
7.4	Composition du trafic	110
7.5	Caractérisation des flots	110
7.6	Partage de bande passante	111
7.7	Impatience	111

7.8	Etude par ACP	111
7.9	Rejeu du controle d'admission	111
7.10	Rejeu d'une trace au niveau paquet	111
7.11	Rejeu d'une trace au niveau flot	111
8	Self-Protect	113
8.1	Introduction sur la qualité de service dans le réseau d'accès	113
8.1.1	Un goulot d'étranglement actuellement dans le réseau d'accès	113
8.1.2	Quels moyens pour garantir la QoS ?	113
8.1.3	Le débat sur la neutralité de l'Internet	114
8.1.4	Self-Protect dans l'évolution de l'architecture des réseaux	114
8.2	Etat de l'art des solutions de QoS existantes	114
8.2.1	Lien montant	114
8.2.2	Lien descendant	115
8.3	Présentation de l'architecture Self-Protect	116
8.3.1	Présentation	116
8.3.2	Configuration dans lesquelles Self-Protect peut être appliqué	116
8.3.3	Fonctionnement global	116
8.3.4	Ordonnancement des flots	116
8.3.5	Interaction de l'utilisateur	116
8.3.6	116
8.3.7	Signalisation	116
8.3.8	Politique d'attribution des priorités	116
8.3.9	Self-Protect sur le poste client	116
8.3.10	Equivalent Self-Protect avec une solution de DPI	116
8.4	Réalisation d'un prototype de démonstration	116
8.4.1	Composants matériels et logiciels	116
8.4.2	Tables de flots	117
8.4.3	Ordonnancement des flots	118
8.4.4	Détermination des priorités	119
8.4.5	Vision utilisateur de la table des flots	120
8.4.6	Cas des tunnels, socks, VPN, et trafic chiffré	121
8.4.7	Statistiques	121
8.4.8	Arbitrage et contrôle d'admission en cas de surcharge	121
8.4.9	Regroupement des flots en macro-flots	121
8.4.10	Extensions possibles du prototype	122
8.5	Résultats des expérimentations	123
8.5.1	Mise en évidence du besoin de QoS	123
8.5.2	Insuffisances du Fair Queueing seul dans le réseau d'accès	123
8.6	Conclusion	123
8.7	Trucs en vrac à ne pas mettre dans le rapport	123
9	Conclusion	125

Chapitre 1

Introduction

Résumé

Cette thèse est proposée dans le contexte d'une vision de la qualité de service (QoS) dans les réseaux IP au niveau du flot de données, dite *Flow-Aware Networking*. Elle s'inscrit notamment au sein d'une étude approfondie de sa réalisation au sein d'un routeur appelé *Cross-Protect*, par une considération du trafic en rupture avec les approches classiques. Dans ce chapitre, nous présenterons d'abord le contexte dans lequel cette proposition a été faite, ainsi que les différentes caractéristiques des réseaux sur lesquelles vont reposer nos études, et qui ont motivé cette approche en rupture. Un résumé des différents chapitre de la thèse est ensuite proposé, accompagné des contributions de chacun à l'état de l'art.

1.1 La place de la QoS dans les réseaux IP

augmenter la valeur des ressources du réseau protéger les utilisateurs de la congestion partielle complémentaire du trafic engineering dynamique (multipath, etc) ne se substitue par a un dimensionnement correct une brique supplémentaire du réseau qui apporte des informations notamment pour le dimensionnement mais aussi pour le routage dyn/ load balancing / protocoles e2e valeur ajoutée
plusieurs visions de la QoS

1.1.1 Architecture du réseau Internet

Le réseau Internet que nous connaissons aujourd'hui est omniprésent et hétérogène ; il permet à des utilisateurs de profils variés de s'échanger des flux de données qui couvrent aujourd'hui un très vaste champ d'applications en constante évolution, dont les plus courantes sont aujourd'hui la voix-sur-IP (VoIP, *Voice over IP*), des canaux de télévisions, des flux web, mails ou P2P pour n'en citer que quelques unes. On parle d'un *réseau multiservice*. Il représente une réduction des coûts par rapport au maintien de réseaux séparés, comme "historiquement" le Réseau Téléphonique Commuté Public (RTC)¹.

Au niveau physique, le réseau est constitué d'un ensemble d'équipement réseaux 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. Cette thèse s'inscrit dans un contexte relativement général ; certaines notions seront plus spécifiques à un réseau d'opérateur (contrats de services, tarification, accès au réseau, etc.).

1.1.2 Routage des flux de données

Les accords économiques entre ces différents acteurs définissent des politiques de routage, c'est-à-dire la manière dont sont gérés les échanges de trafic entre eux, indépendamment de l'architecture physique sous-jacente.

Ces accords sont catégorisés selon 3 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 où les deux réseaux offrent chacun la connectivité à l'autre (ce peut être le cas lorsque deux AS historiques sont désormais regroupés sous la même entité administrative).

Ces accords sont dynamiques et permettent de classer plusieurs possibilités de routage en fonction de leur viabilité économique. Ils sont réalisés par l'intermédiaire du protocole BGP². Ce protocole ne régit que les échanges entre les différents AS et en aucun cas l'acheminement des flux en interne. Chacun est libre d'utiliser le protocole de son choix, les plus courants étant IS-IS³, OSPF⁴ ou RIPv2⁵.

Ces accords jouent un rôle important par rapport au dimensionnement d'un réseau et à la performance qu'il offre en ce qu'un fournisseur doit assurer la connectivité négociée à son client. Les liens de peering qui quant à eux reposent sur un libre échange du trafic entre deux entités, sont en pratique plus souvent congestionnés.

1. en anglais, PSTN pour *Public Switched Telephone Network*

2. Border Gateway Protocol

3. Intermediate System to Intermediate System

4. Open Shortest Path First

5. Routing Information Protocol

1.1.3 Protocoles réseaux et QoS

Le protocole IP (*Internet Protocol*) – et notamment sa version IPv4 – est aujourd’hui la brique essentielle du réseau. Il permet l’adressage des différentes machines et équipements au travers de leur adresse IP, et ainsi l’établissement de communications. Par conception, le réseau IP est dit *Best Effort*, c’est-à-dire qu’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 équitablement la bande passante, où d’éviter les situations de congestion.

On peut ainsi opposer le réseau *best effort* IP au Réseau Numérique à Intégration de Services (RNIS)⁶ basé sur le protocole ATM⁷. Il s’agit d’une évolution du RTC assez complexe destinée à supporter les différentes applications constituant le réseau, qui a été supplantée par IP pour sa simplicité et son ouverture aux utilisateurs du réseau. ATM subsiste encore dans certaines portions du réseau (accès notamment), et on ne sera pas étonné d’en retrouver certains mécanismes dans les propositions de garanties de service pour IP.

Enfin, à l’inverse du réseau que nous connaissons actuellement, la performance du réseau téléphonique est bien maîtrisée, basée sur des modèles robustes de performance. Nous verrons dans le Chapitre 3 qu’il est possible d’appliquer des principes similaires pour le trafic IP.

1.1.4 Réseaux de cœur, de collecte, d’accès

Nous distinguerons trois composantes structurelles dans un réseau classique d’opérateur, de caractéristiques différentes : nombre d’utilisateurs, capacité des liens et débits des flots, technologie utilisée, etc. La modélisation en sera donc souvent spécifique.

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, GRPS, etc. Sa capacité est aujourd’hui plutôt limitée (surtout dans le sens descendant 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 le problème 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 à proprement dit par l’intermédiaire du BRAS. 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 que nous avons précédemment décrite comme réseau Internet, 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.

Dans ce manuscrit, notre discussion sera principalement focalisée sur les infrastructures de cœur de réseau, à l’exception du Chapitre ?? où l’on traitera des réseaux d’accès.

6. en anglais, ISDN pour *Integrated Services Digital Network*

7. Asynchronous Transfer Mode

1.1.5 Enjeux et challenges pour un opérateur

L'enjeu de la QoS pour un opérateur de réseau apparaît dès lors que le coût de l'investissement pour le surdimensionnement devient trop important. Pourtant même dans un contexte actuel, les mécanismes de QoS permettrait de ralentir ces coûts tout en continuant d'offrir un service satisfaisant aux clients, et ce sans nécessiter d'outils de filtrage. Potentiellement, cette QoS permettrait même de mieux tirer parti des ressources du réseau en fonction des applications, qui sont de plus en plus diversifiées et exigeantes en ressources (temps réel, vidéoconférence, téléphonie, télévision, données, etc.). Elle ne dispense pas bien sûr d'un dimensionnement correct du réseau. Elle semble indissociable d'une tarification faite par un opérateur [?], que ce soit pour la facturation de services, ou simplement à des fins de retour sur investissement.

Le problème de la neutralité des réseaux se pose également au vu des discussions actuelles sur le sujet. Enfin, une solution se doit d'être simple et peu coûteuse, à la fois pour sa réalisation dans les équipements et pour sa maintenance, et elle idéalement et doit permettre d'assurer une qualité de service inter-domaine au travers de plusieurs AS.

Nous verrons pourquoi nous pensons que la proposition *Flow-Aware Networking* que nous présentons dans le chapitre suivant est adaptée à garantir la performance du trafic. Elle est réalisable, robuste, permet de conserver l'interface *best effort* au réseau, et respecte les principes de la neutralité des réseaux[28]. Ne nécessitant aucune standardisation, il est possible de l'introduire progressivement dans le réseau, en commençant par les connexions les plus sujettes à la congestion comme par exemple les liens de *peering*.

1.2 L'état de la QoS dans le réseau Internet actuel

1.2.1 Un réseau *best effort* surdimensionné

Aujourd'hui, les réseaux ne possèdent généralement pas de mécanismes de garantie de QoS, ce qui fait d'Internet un réseau dit *best effort*. Dans les exceptions, nous avons par exemple les réseaux d'entreprises, où des solutions de type Diffserv – que nous allons présenter – sont utilisées.

Des mesures effectuées dans le réseau permettent d'estimer les volumes de trafic échangés afin de définir les capacités des liens et un routage des flux approprié. Le réseau supporte une certaine tolérance aux pannes d'équipements ou de lien en permettant au trafic d'être routé par un autre chemin. Les garanties de services aux utilisateurs ou aux réseaux clients sont définies par un accord (SLA, *Service Level Agreement*) qui définit les attentes de service. Dans un tel contexte, ces garanties sont apportées par un surdimensionnement des ressources (par exemple, un lien ne sera pas utilisé à plus de 40%).

Cette technique est coûteuse en investissements, notamment au vu des difficultés causées par les débits des échanges de fichiers en peer-to-peer, et semble d'autant plus menacée par l'augmentation des débits d'accès par les technologies que la fibre optique. Il semblerait au vu des efforts de filtrage que nous connaissons aujourd'hui de la part des opérateurs que cette solution a aujourd'hui atteint ses limites, et qu'il convient peut-être d'envisager d'autres solutions telles que l'introduction de mécanismes de qualité de service (QoS, *Quality of Service*) jusque dans le cœur de réseau.

La majorité du trafic est dû au protocole TCP, sur qui repose une allocation équitable des ressources réseau et le contrôle de congestion. L'équité provient du fait que la grande majorité des flots implémentent TCP et ne sont pas malveillants [61]. Notons que les débits d'accès sont aujourd'hui relativement faibles comparés au cœur de réseau, et le goulot d'étranglement pour les flots se situe ainsi généralement dans le réseau d'accès. Les mécanismes de TCP n'interviennent généralement pas dans le cœur de réseau, et

1.2 L'état de la QoS dans le réseau Internet actuel

cela correspond à certains avis qui considèrent que la QoS n'y est pas nécessaire et que le surdimensionnement suffit.

Cependant, l'augmentation des débits d'accès avec l'arrivée de la fibre optique pourrait déplacer ce problème plus en avant, au moins dans le réseau de collecte. Il est également possible que des flots utilisateurs, n'utilisant peut-être pas les mécanismes de TCP, soient capable d'avoir un débit (ou des débits combinés) approchant la capacité du cœur de réseau. C'est dans cette vision que s'inscrivent les travaux évoqués dans cette thèse.

1.2.2 Comportement et performance de TCP

Le protocole TCP, utilisé pour les transferts fiables de données, représente depuis de nombreuses années la majeure partie du trafic Internet. Cela explique la position centrale qu'il a dans de nombreuses études de performance.

TCP a pour but d'établir, de manière décentralisée, une allocation efficace et équitable de la bande passante disponible sur le réseau *best effort*. 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 [41] afin de faire face à la saturation du réseau (*congestion collapse*). Cette version de TCP (nommée Tahoe) et améliorations suivantes (Reno et NewReno) s'appuient sur les pertes de paquets qui se produisent lorsque les connexions TCP saturent le buffer d'un des routeurs. TCP Vegas [22] diffère en ce qu'il utilise le délai subi par les connexions à la traversée des files d'attente.

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 (mode *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 possèdent 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 connexions. Nous n'entrerons pas ici dans les détails d'implémentation de TCP.

L'algorithme AIMD, qui permet un partage équitable des ressources, souffre néanmoins de plusieurs imperfections. Il est par exemple inefficace sur les 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 ??). Plusieurs propositions de protocoles "haut débit" permettent à plusieurs long flots 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 équitables avec les connexions TCP Reno, ce qui est une propriété souvent recherchée (on parle de protocoles *TCP friendly*).

L'allocation équitable réalisée par TCP dépend de la coopération des sources, et est ainsi vulnérables à celles n'implémentant pas TCP⁸, dont certaines dans un but malicieux⁹.

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).

8. Notons que certaines applications *streaming* sont désormais capable d'adapter leur débit également, grâce à des protocoles tels que RTP.

9. 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.

D'autre part, lors des événements de saturation d'un buffer, de nombreuses connexions TCP réduisent leur *cwnd* en même temps, ce qui entraîne une moins bonne utilisation des ressources du lien.

flots streaming

Aujourd'hui, ces flots représentent une faible proportion du trafic, et bénéficient du traitement *best effort*¹⁰.

1.3 Contenu et contributions de la thèse

Chapitre 1 : Qualité de service dans les réseaux IP

Ce chapitre introductif présente le contexte et le contenu de la thèse.

Une des contributions de la thèse est de la situer par rapport aux propositions existantes, et de justifier les motivations des études dont elle fait partie.

Chapitre 2 : 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. Les modèles utilisés sont à la base des réflexions présentées dans les chapitres suivants, dont le but est de raffiner et d'améliorer la proposition.

Chapitre 3 : Dimensionnement des buffers pour les routeurs de cœur de réseau

Chapitre 4 :

Chapitre 5 : Un algorithme de MBAC pour Cross-Protect

Chapitre 6 : Analyse d'une trace de trafic

Métrologie, Analyse trace réelle, rejeu trace, évaluation algorithme MBAC ?

Chapitre 7 : Self-Protect

Déclinaison pour l'accès de l'architecture FAN Net Neutralité Prototype de démonstration

10. Cette faible présence s'explique notamment par le manque de support du réseau, qui a mené à des solutions alternatives pour les solutions non-temps réel : streaming HTTP, téléchargement progressif avec mise en tampon, etc. Notons que l'on a également des flux *streaming* à débit adaptatif.

Chapitre 2

Challenges pour fournir une Qualité de Service dans les réseaux IP

Résumé

Cette thèse est proposée dans le contexte d'une vision de la qualité de service (QoS) dans les réseaux IP au niveau du flot de données, dite *Flow-Aware Networking*. Elle s'inscrit notamment au sein d'une étude approfondie de sa réalisation au sein d'un routeur appelé *Cross-Protect*, par une considération du trafic en rupture avec les approches classiques. Dans ce chapitre, nous présenterons d'abord le contexte dans lequel cette proposition a été faite, ainsi que les différentes caractéristiques des réseaux sur lesquelles vont reposer nos études, et qui ont motivé cette approche en rupture. Un résumé des différents chapitre de la thèse est ensuite proposé, accompagné des contributions de chacun à l'état de l'art.

La performance des flots composant le trafic internet est intimement liée à la capacité du réseau et à la demande/charge générée par les utilisateurs. Elle dépend bien entendu de la nature des flots elle-même, mais également des mécanismes interagissant avec ces flots : protocole de transport, ordonnancement, gestion des files d'attente, contrôle d'admission, etc.

Contents

2.1	Introduction	12
2.2	Offrir des garanties de service au trafic IP	12
2.3	Mécanismes de QoS dans les réseaux actuels	14
2.4	Régimes opérationnels des liens	15
2.5	Techniques d'amélioration de la QoS	17
2.6	Architectures de Qualité de Service	22
2.7	Discussion	26

Contributions :

- Etat de l'art des éléments de QoS et challenges
- Comparaison des architectures de QoS

2.1 Introduction

Introduction ou l'on présente ce qu'on va faire

QoS résulte de l'ensemble des interactions entre protocoles de transport, ordonnancement et gestion de la file d'attente et plus globalement des décisions prises par le réseau

initialement intelligence à l'edge Il est ainsi de la responsabilité des couches supérieures d'assurer le bon fonctionnement des applications, comme par exemple le protocole de transport TCP qui assure un transport fiable et équitable des données, ou encore certaines applications overlay (distribution de contenu peer-to-peer par exemple). On réfère souvent au principe de bout-en-bout, selon lequel la complexité du traitement du trafic doit être effectué en bordure de réseau [?]. Il a permis l'émergence et le fonctionnement d'applications diverses jusqu'à aujourd'hui, et a permis de faire faces aux nombreuses évolutions de l'Internet et du trafic qu'il transporte. protocoles TCP, applications overlay (P2P, etc.)

face aux insuffisances, améliorations + ajout de fonctionnalités au réseau Motivations pour l'ajout de fonctionnalités au réseau. Débat sur le besoin de QoS bw abondant and cheap (wdm etc.) but applications to use it available on vander routers - some demand on veut aujourd'hui nouveaux besoins des applications peut-être accéléré par combat ISP / content besoins d'investissements majeurs / valeur dans le réseau Possible contexte net neutrality...

Survol des éléments de QoS, (inspiration de ATM) Pas de garantie de service native dans IP, contrairement à ATM. On notera la similarité des nombreuses propositions à réimplémenter les mécanismes d'ATM sur IP.

architectures de QoS = assemblage de briques, comparaison et évaluation

2.2 Offrir des garanties de service au trafic IP

Help see what's missing in current architectures, and how different solutions address those issues. Help compare different proposals

= notre vision du trafic = comment évaluer les propositions

Challenges for architectures : real scalable interdomain NN etc. our table

2.2.1 Granularité du trafic IP

Modélisation au niveau flot est particulièrement développée dans le chapitre suivant.

La qualité de service des applications est ainsi réalisée à plusieurs niveaux. Il est pratique de considérer une séparation des différentes échelles de temps, représentées sur la figure 2.1.

2.2 Offrir des garanties de service au trafic IP

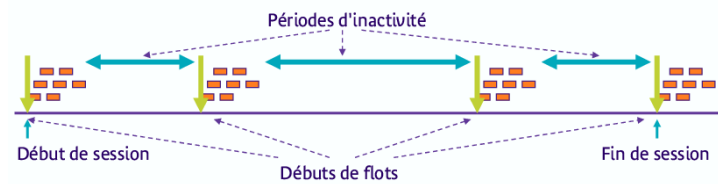


FIGURE 2.1: Représentation des différentes échelles de trafic.

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 [57]. Elle résulte de la distribution de la taille des flots, qui possède une variance infinie, ainsi que du caractère *bursty* du protocole majoritaire TCP[69].

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/UDP¹, 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

Les applications constituant la majeure partie du trafic IP peuvent être classées en deux groupes ayant des propriétés et des besoins de garanties de QoS différents [81, 75].

Les flots élastiques 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.

Les flots *streaming* sont ceux produits par les applications audio et vidéo, et sont typiquement transportés sur UDP. Ils sont caractérisés par leur durée et leur débit intrinsèques. Contrairement aux flots élastiques qui ne requièrent qu'une garantie de service à long terme, leur qualité de service s'exprime au niveau paquet, nécessitant des délais et taux de pertes négligeables.

Il existe également des flots *streaming* à débit adaptatif, qui adaptent leur débit aux ressources disponibles, à la manière des flots élastiques. Il convient toutefois de garantir un débit suffisant à garantir une bonne qualité pour l'utilisateur.

2.2.3 Challenges - Compromis

Prédictabilité : robustesse, insensibilité Garanties : SLA ! services valeur ajoutée

Opex : configuration, description simple du trafic, dimensionnement+quelle mesures
Capex

Mise en oeuvre : réalisabilité, déploiement incr., paramétrage, extensibilité (etats), simplicité (calculs haute vitesse, communication), etats (mémoire), Interdomaine (trust)

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

Legal : transparent and neutral network

2.3 Mécanismes de QoS dans les réseaux actuels

Sauf exception, les réseaux aujourd'hui n'implémentent aucun mécanisme de garantie de service. Ces exceptions sont généralement l'emploi de mécanismes de type Diffserv (que nous allons présenter) pour des réseaux d'entreprise, ou l'ajout ad-hoc d'application de bridage (par exemple envers le peer-to-peer) que l'on peut trouver dans certains réseaux d'accès.

Le réseau qui repose essentiellement sur le transport réalisé par la brique IP est dit *best effort*. Une des raisons probable est la complexité de mise en oeuvre des différentes solutions proposées. Reluctance à introduire des mécanismes dans le coeur, contraire à ce qui a garanti la robustesse et la polyvalence du réseau.

C'est grâce à un surdimensionnement des ressources qu'il permet aux différentes applications de coexister et d'obtenir des performances raisonnables (ex cas de panne, charge à 40%). Semble atteindre ses limites. Applications telles que le P2P, TOR, etc ont tendance à utiliser la BP. Dimensionnement des ressources ne devrait pas être sensible à l'augmentation de ce trafic.

2.3.1 Le rôle de TCP et UDP dans la gestion du trafic

il faudrait avoir présente les différents types d'application avant.

UDP transporte généralement les flux streaming : leur débit ne s'adapte généralement pas aux pertes rencontrées sur le lien. Ce peut être bien pour les congestions temporaires dues à TCP par ex. Certains flux adaptatifs permettent de réagir, désirable en cas de congestion sévère.

La majorité du trafic est dû au protocole TCP, sur qui repose une allocation équitable des ressources réseau et le contrôle de congestion. L'équité provient du fait que la grande majorité des flots implémentent TCP et ne sont pas malveillants [61]. Notons que les débits d'accès sont aujourd'hui relativement faibles comparés au cœur de réseau, et le goulot d'étranglement pour les flots se situe ainsi généralement dans le réseau d'accès. Les mécanismes de TCP n'interviennent généralement pas dans le cœur de réseau, et cela correspond à certains avis qui considèrent que la QoS n'y est pas nécessaire et que le surdimensionnement suffit.

Sa position centrale dans le trafic explique notamment pourquoi il est souvent le centre d'études sur la performance du trafic. Cependant, l'augmentation des débits d'accès avec l'arrivée de la fibre optique pourrait déplacer ce problème plus en amont, au moins dans le réseau de collecte. Il est également possible que des flots utilisateurs, n'utilisant peut-être pas les mécanismes de TCP, soient capables d'avoir un débit (ou des débits combinés) approchant la capacité du cœur de réseau. C'est dans cette vision que s'inscrivent les travaux évoqués dans cette thèse.

2.3.2 Comportement et performance de TCP avec des liens FIFO Drop-Tail

TCP a pour but d'établir, de manière décentralisée, une allocation efficace et équitable de la bande passante disponible sur le réseau *best effort*. 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 [41] afin de faire face à la saturation du réseau (*congestion collapse*). Cette version de TCP (nommée Tahoe) et améliorations suivantes (Reno et NewReno) s'appuient sur les pertes de paquets

2.4 Régimes opérationnels des liens

qui se produisent lorsque les connexions TCP saturent le buffer d'un des routeurs. TCP Vegas [22] diffère en ce qu'il utilise le délai subi par les connexions à la traversée des files d'attente.

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 (mode *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 possèdent 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 connexions. Nous n'entrerons pas ici dans les détails d'implémentation de TCP.

L'algorithme AIMD, qui permet un partage équitable des ressources, souffre néanmoins de plusieurs imperfections. Il est par exemple inefficace sur les 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 ??). Plusieurs propositions de protocoles "haut débit" permettent à plusieurs long flots 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 équitables avec les connexions TCP Reno, ce qui est une propriété souvent recherchée (on parle de protocoles *TCP friendly*).

L'allocation équitable réalisée par TCP dépend de la coopération des sources, et est ainsi vulnérables à celles n'implémentant pas TCP², dont certaines dans un but malicieux³.

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, lors des événements de saturation d'un buffer, de nombreuses connexions TCP réduisent leur *cwnd* en même temps, ce qui entraîne une moins bonne utilisation des ressources du lien.

2.3.3 Problèmes de TCP

Le nombre considérable de propositions visant à
Nombreux pb de TCP lenteur aimd (pertes) rtt bias synchro saturate buffer gros
buffers : bdp overload : timeouts : lost transfer / user impatience no fairness at packet
scale : bursts cooperation :tcp friendliness
fast recovery / fast retransmit
hints inside tcp evolution short connections wireless connections : losses != congestion
feedback fairness performance aspect to be improved
ameliorations de tcp pour chacun de ces points : buffer size introduction de mécanismes dans le réseau

2.4 Régimes opérationnels des liens

Il est pratique de distinguer trois régimes de partage de bande passante, qui sont représentés en figure 2.2. Dans le dessin du haut, les flots sont représentés par des boîtes

2. Notons que certaines applications *streaming* sont désormais capable d'adapter leur débit également, grâce à des protocoles tels que RTP.

3. 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.

de hauteur égale à leur débit exogène moyen, 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.

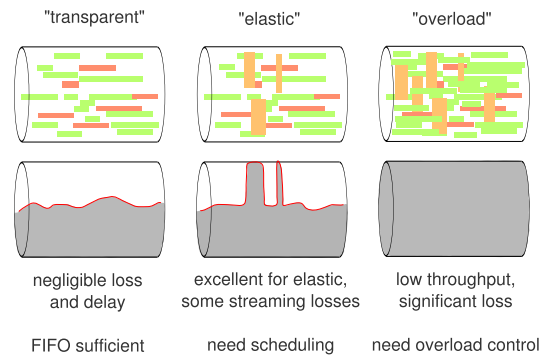


FIGURE 2.2: Trois régimes d'opération du lien : "transparent", "élastique" et "surcharge"

Le cas de gauche transcrit un régime "transparent" dans lequel tous les flots ont un débit limité et la somme de leurs débits reste toujours 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 le débit alloué 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 sur-provisionnement effectué par les opérateurs et des débits d'accès relativement faibles de la plupart des utilisateurs.

Le régime "élastique" 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 tampons des liens se remplissent et débordent, causant des délais et des pertes impactant notamment les flots *streaming*. Les délais peuvent notamment être significatifs si la taille des tampons 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.

Le cas de droite correspond à un régime de "surcharge" où l'offre de trafic dépasse la capacité disponible sur le lien. Dans ces conditions, le nombre de flots augmente très rapidement et leur débit tend vers zéro. Les algorithmes de contrôle de congestion tels qu'implémentés dans 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 semblent ainsi nécessaires afin d'éviter une telle configuration.

2.5 Techniques d'amélioration de la QoS

2.5.1 Alternatives au contrôle de congestion de bout en bout

Cette section ne peut – et ne prétend pas – présenter exhaustivement les principales propositions afin d'améliorer l'état de la QoS dans le réseau. Le choix des solutions présentées nous permet cependant de mettre en lumière les différentes idées mises en œuvre, ainsi que les briques disponibles pour réaliser des garanties de services à l'échelle du réseau. Nous soulignons notamment les problèmes soulevés par ces propositions (réalisabilité, configuration, robustesse, garanties, etc.) qui nous permet de mieux illustrer la difficulté à répondre à l'ensemble des besoins évoqués précédemment.

Evolution des protocoles de transport

Current status

Reasons for change > asymmetric path > other losses » slow AIMD

Window control : Delay, HighSpeed TCP versions

Packet sending rate : pacing

TCP Friendliness

Background transfer : TFRC LEDBAT

packet loss not sufficient network conditions : delays

La RFC2309 [?] souligne la difficulté à contrôler dynamiquement le trafic depuis la bordure de réseau sans indication supplémentaire que les pertes de paquets. Elle présente deux classes d'algorithmes pouvant être réalisés sur les routeurs afin d'effectuer un contrôle de congestion plus efficace.

Gestion des files d'attente

L'ajout de mécanismes de gestion des files d'attente au sein des routeurs (dits AQM pour *Active Queue Management*) représente une alternative au contrôle de congestion de bout en bout [35]. Ils 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 dus à la politique *DropTail*. On retrouve ici les éléments de la terminologie ATM : *congestion-indication* et *explicit-rate*.

Estimation de la congestion

Ainsi, RED [37] 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 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 entre flots.

RED ne permet pas d'éliminer les pertes de paquets. [32] suggère 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 [70, 57]. 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 [?], PI [40] et AVQ [82].

Malgré les propriétés d'ajustement automatique des algorithmes plus récents tels que BLUE, ARED [36] ou PI, la plupart des AQM sont complexes à paramétrer et n'offrent pas une performance prédictible.

Marquage de paquets

Afin d'éviter le rejet d'un paquet, qui gaspille les ressources du réseau, [34?] 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 pose de problèmes de mise à l'échelle pour des liens de cœur de réseau comprenant plusieurs millions de flots.

[35, 59, 64]

RED with penalty box [?] détermine les flots non-coopératifs à partir d'un historique des derniers événements de perte, et leur impose une limite en débit (CBQ [38]). Flow RED [59] se base sur la place occupée par chaque flot dans le buffer. Stabilized RED [64] estime le nombre de connexions actives. SFB [32] 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 [68] 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 [?] 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.

SFB big buffer, needs enhancements SFB uses rate limit

Solutions mixtes transport / AQM

Plutôt que de signaler une congestion, XCP [?] propose aux sources une évolution incrémentale de *cwnd* sur plusieurs RTT, 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 [29] propose d'envoyer explicitement aux flots un débit à réaliser afin d'émuler un comportement PS. Ce débit est un débit équitable calculé périodiquement à partir du débit des flots et de la taille de la file d'attente. Un problème supplémentaire de XCP et RCP est qu'ils requièrent la coopération de l'ensemble des routeurs du réseau.

Quid en cas de flashcrowd ?

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 (*First In First Out*), qui consistent généralement au maintien de plusieurs files d'attentes (virtuelles ou non). Cette classification peut se faire par flot, ou par classes.

2.5 Techniques d'amélioration de la QoS

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*) [?], WRR (*Weighted Round Robin*) [?] ou DRR (*Deficit Round Robin*), 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) – consistant à attribuer le même débit à chaque flot, et il en existe de nombreuses implémentations. 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 alloués 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 différenciation sont quant à elles exploitées dans l'architecture FAN que nous présentons dans le Chapitre 3.

FQ n'est souvent pas considéré comme une alternative réalisable du fait qu'il nécessite de maintenir une file d'attente par flot en cours, de l'ordre de plusieurs millions sur un lien de cœur de réseau. Plusieurs propositions permettent d'obtenir un algorithme approximatif [? ?] mais elles restent encore à étudier.

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*.

Une autre approche classique consiste à regrouper les flots en différentes classes (CBQ (*Class-Based Queueing*) [?]). Cette méthode repose généralement sur un marquage effectué dans un élément distant du réseau après reconnaissance du flot en question par divers mécanismes de caractérisation et de classification.

Caractérisation de trafic

- related to NN How can we characterize traffic
- Simple protocol based on port number less reliable encryption/tunneling/disguised traffic DPI based on users (ToS)
- traffic marks / signalization standardization (routers) trust within a single admin domain
- besoin de vérifier conformance (case of reservation) policing/shaping/marking/signalization
- Leaky bucket/virtual queue pb pour caractériser les flots

2.5.2 Contrôle de la surcharge

Le contrôle de congestion effectué par TCP 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 décroît vers zéro, ce qui entraîne une accumulation des connexions et rend le système instable : le temps moyen de transfert des flots tend vers l'infini. Il convient de noter que le système est toujours stable au niveau paquet puisque les tailles finies de buffer entraînent des pertes de l'excédent de trafic.

Conséquences de la surcharge ineffective traffic and congestion collapse [78] retransmissions and incomplete transfers

Cadm + Preemption = sur quelle partie du trafic concentrer la surcharge

Contrôle d'admission

Besoin d'un contrôle d'admission mis en évidence par Roberts et Schenker. Ok pour les flots streaming mais Roberts le propose aussi pour les flots élastiques.

Ce phénomène est cependant mitigé par les *timeouts* subis par les connexions TCP lorsque le nombre de paquets, ainsi que par un phénomène d'impatience qui entraîne l'abandon des connexions.

contrôle d'admission : généralement reconnu pour flots streaming, peu pour les flots élastique, utilité etc. implicite vs explicite : besoin caractérisation+policing reservation et signalisation

need for admission control [78] even for elastic traffic no utility below a given rate != [81] also see [21] esp. if charging : [74]

future in comparison with differentiated services [11] could better utilization and better performance for best-effort flows reduction in availability minimal and targeted to the least demanding users [1, 10]. more useful traffic with AC, different service availability for price discrimination

large table but feasible [?]

Contrôle d'admission et différenciation

Contrôle nécessaire pour les deux classes de trafic Aucune classe plus importante que l'autre pour l'utilisateur

Notons que le taux de blocage des flots permet une différenciation sur l'accessibilité au lien de plusieurs classes d'utilisateurs lors de condition de surcharge [?]. Il semble être une alternative plus efficace aux différentes solutions tentant d'introduire une différenciation (comme le taux de perte par exemple) au sein de chaque classe de trafic, *streaming* et/ou élastique, qui ont des contraintes différentes.

On pourra se référer à [?], [?] pour plus de précisions en ce qui concerne les mécanismes du contrôle d'admission, ainsi qu'à [?] en ce qui concerne l'impact de la congestion envers l'utilisateur.

2.5.3 Équité et isolation entre les flots

Le régime "élastique" au milieu de la figure se produit lorsque des flots *bottlenecked* s'ajoutent au trafic dit *background* généré par les flots *non-bottlenecked*, même lorsque la charge globale offerte reste inférieure à la capacité du lien. Ces flots *bottlenecked* sont générés par des utilisateurs possédant un débit d'accès particulièrement haut, ce qui est potentiellement possible désormais avec la montée en débit des offres commerciales. Lorsqu'un ou plusieurs flots *bottlenecked* sont en cours, nous pouvons nous attendre à de la congestion au niveau paquet qui conduit à des délais et un certain taux de perte de paquets. Les délais peuvent devenir significatifs si de grands buffers sont utilisés, en particulier pour les applications *streaming* qui y sont sensibles. En plus d'un dimensionnement adéquat, il est nécessaire d'utiliser un mécanisme d'ordonnancement afin de préserver la qualité de service des flots dans ce régime.

2.5.4 Limitation de la charge des liens par contrôle d'admission

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 reroutage de trafic, où la performance est significativement dégradée en général (phénomènes de *flash crowd*). Il est de plus possible de le rendre transparent en considérant des solutions de reroutage des flots, ou par l'emploi de *multipath*.

Nous allons voir pour chaque type de flot, les raisons qui nous poussent à l'introduction d'un contrôle d'admission, ainsi que les mesures de la probabilité de blocage dans

2.5 Techniques d'amélioration de la QoS

chaque cas qui doivent typiquement être considérées par l'opérateur pour le dimensionnement de son réseau.

Préemption de flots en cours

emergency

Path selection

En complement du controle d'admission!!

Load balancing [14] Multipath and trunk reservation [66] admission control fait partie du routage adaptatif, si un chemin bloqué, essaie u nautre randomized load balancing ?

2.5.5 Différenciation

Il s'agit de traiter différemment plusieurs classes de trafic, ce qui peut-être fait dans un but de tarification, ou afin d'augmenter l'utilité des ressources du réseau. Cette différenciation peut se faire au niveau de la transparence de l'accès aux ressources du réseau – en termes de délais ou de taux de pertes de paquets, ou de bande passante allouée à un flot –, par l'utilisation de différentes stratégies d'ordonnancement. [?] suggère que ce type de différenciation n'est pas efficace, puisque la QoS effectivement ressentie par les flots est soit excellente, soit très mauvaise et qu'il est ainsi difficile de définir plusieurs niveaux intermédiaires. De plus, elle répartit les dégradations sur l'ensemble des flots. Il semble plus intéressant d'effectuer cette différenciation au niveau de la disponibilité des ressources au travers d'un contrôle d'admission sélectif, en assurant potentiellement différentes probabilités de blocage. Dans cette situation, seule une faible partie des flots est affectée, tandis que tous les flots acceptés voient un réseau transparent.

Outre ces considérations qui visent principalement une tarification différenciée, il est intéressant de distinguer plusieurs classes de trafic avec l'objectif d'augmenter l'utilité des ressources pour l'utilisateur tout en respectant les contraintes plus ou moins fortes de QoS de chacune. Il est par exemple possible d'exploiter les besoins intrinsèquement différents entre les flots streaming et élastiques, en servant les premiers en priorité – leur garantissant ainsi de faibles délais et taux de pertes – en laissant les flots élastiques exploiter la bande passante restant, ce qui suffit à leur garantir un débit minimum à long terme, et donc un temps de complétion satisfaisant. Bonald et Massoulié [15] exhibent des configurations de réseaux non-équitables (par exemple lorsque certains flots reçoivent une priorité supérieure) qui sont instables même lorsque le système n'est pas saturé. Le contrôle de cette stabilité est ainsi désirable pour de telles configuration dans un contexte dynamique. Autre intérêt du contrôle d'admission.

Le respect de la neutralité des réseaux impose de baser la différenciation sur des critères neutres. Il ne peut ainsi pas s'agir de marquer explicitement un type d'application, ou une destination précise. Par contre, la différenciation implicite effectuée par les algorithmes de fair queueing en fonction du débit des flots, qui leur est une caractéristique intrinsèque de bout en bout, le permet.

Différenciation au sein d'une même classe de flots

Il est plus difficile d'argumenter pour de la différenciation au sein d'une même classe de flots.

La différenciation des flots *streaming* peut se faire en fonction des délais (*Priority Queueing*, *Weighted Fair Queueing*) ou des taux de pertes (*Class-Based Queueing*, *Spatial Queue Priority*), mais il est plus simple dans le cœur de réseau de garantir une bonne QoS pour tous les flots en priorité, notamment si leur charge est relativement faibles. D'autant plus

section à intégrer

que généralement, la performance est soit bonne (jusqu'à une certaine charge), soit mauvaise. Les modèles sous-jacent ne sont pas insensibles et la performance dans le contexte d'une différenciation de trafic est difficilement prédictible.

Comme nous le verrons dans le Chapitre ??, 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 des flux TV à fort débit par exemple [20].

La différenciation des flots élastiques peut reposer sur l'attribution de capacités limitées aux différentes classes (*Class-Based Queueing* ou de poids différents (*Weighted Fair Queueing*, RTT, etc.). La différenciation se produit en surcharge, et n'est pas manifeste si les flows ont un débit d'accès limité. [75] montre qu'une différenciation pour les flots élastiques en attribuant des poids (modèle *Discriminate Processor Sharing*) est peu intéressante et rend la performance des flots sensible à la distribution de la taille des flots.

Puisque la congestion touche l'ensemble des flots *best effort*, quels que soient leurs poids, il est préférable d'introduire un contrôle d'admission.

La performance des longs flots par contre est relativement indépendante du débit réalisé par les flots plus courts. Il est donc intéressant d'envisager de donner la priorité à ces derniers [39, 7]. [8] montre que la discipline correspondante SRPT⁴ est relativement équitable comparée à *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.

2.6 Architectures de Qualité de Service

Combinaison de mécanismes de gestion du trafic, visant à proposer une alternative au réseau Best Effort actuel. Elle résulte souvent de l'assemblage de briques élémentaires telles que nous venons de présenter, avec l'objectif de fournir une solution complète et robuste de qualité de service pour un réseau.

Nous décrivons dans cette section les différentes visions de la QoS et tentons de les analyser et les comparer au vu des critères que nous avons présenté dans les sections précédentes.

[?] : SOTA + advanced diffserv mechanisms

2.6.1 Intserv/Diffserv

Intserv

Les groupes de travail de l'IETF ont défini dès le début des années 1990, deux modèles qui reposent sur la notion de classe de service (CoS) à qualité de service différenciée. Alors que *IntServ* ([?]) introduit un mécanisme de réservation par flot, *DiffServ* ([?]) envisage au contraire une réservation au niveau d'aggrégats de flots.

Nous présentons rapidement ces deux modèles de service en soulignant leurs limitations, ce qui permettra de mieux comprendre les motivations qui ont mené au développement du mécanisme de contrôle d'admission implicite au niveau flot.

Intserv proposée une protection des flots basées sur une réservation de ressources basée sur un descripteur de trafic, suite à une signalisation effectuée par l'utilisateur. Difficulté de caractériser le trafic, esp VBR

Intserv : réservation + signalisation flow-based stateful all or nothing edge reserves to the nw flots individuels non scalable bandwidth and buffer reservations

Controlled Load : BE approx Guaranteed rate : resa+alloc by signalization of predefined rsrc complexity of signalization

4. Shortest Remaining Processor Time

2.6 Architectures de Qualité de Service

Diffserv

L'architecture Diffserv propose la différenciation explicite de classes et sous-classes de flots, en fonction d'un marquage réalisé en bordure de réseau (à la Intserv). Les routeurs de cœur de réseau (le "domaine Diffserv") peuvent ainsi traiter les paquets en inspectant uniquement la marque apposée dans leur en-tête. La RFC définit trois principales classes de service permettant de servir prioritairement les flots *streaming*, puis ceux dont on veut assurer le service et enfin le trafic *best effort*.

complexité du marquage + besoin de standardisation quelle performance pour chaque classe. peu de différence en charge normale classes les moins prioritaires pénalisées entièrement en cas de surcharge problèmes en interdomaine

Enfin, en l'absence d'un mécanisme de contrôle de flot, la garantie de la performance des flots repose sur le surdimensionnement des liens, d'autant plus que le contrôle en bordure de réseau reste approximatif faute d'une connaissance très précise de la matrice de trafic.

Group flows into several traffic classes Marking at the edge Guarantees for S : under certain conditions on load/ network topo

Advanced diffserv : see paper

2.6.2 Core Stateless Fair Queueing

Emulate intserv, packets are carrying perflow state

Same principle : FADE [?] (*Fair Allocation Derivative Estimation*) : feedback of fair share information to edge nodes LBFA [?] (*Link Based Fair Aggregation*) : flow aggregation by input link (?)

Core-Stateless Fair Queueing [15] (CSFQ) is a highly scalable approach for enforcing fairness between flows without keeping any state in the core of the network. The approach relies on per-flow accounting and marking at the edge of the network in conjunction with a probabilistic dropping mechanism in the core of the network. The idea behind CSFQ is to estimate the rate of the flow at the ingress of the network or network cloud and to attach an estimate of the flow's sending rate to every packet that the flow sends. Given this label, intermediate routers at congested links in the network calculate a dropping probability which is derived from an estimate of a fair share of the bottleneck link capacity and the rate of the flow as identified in the label. While CSFQ provides an elegant and efficient solution to providing fairness, it relies on the use of additional information that is carried in every packet of the flow. Thus, the scheme trades off overhead in the packet header at every network link for resource management overhead at the bottleneck router. In addition, it requires that both intermediate routers and edge devices adhere to the same labeling and dropping algorithm. A misconfigured or poorly implemented edge device can significantly impact the fairness of the scheme. SFB, on the other hand, does not rely on coordination between intermediate routers and edge markers and can perform well without placing additional overhead in packet headers.

2.6.3 Contrôle de flot dans Diffserv

Plusieurs propositions suggèrent l'ajout de mécanismes de contrôle de flot à Diffserv pour gérer les situations de surcharge, et ainsi moins reposer sur un surdimensionnement des liens[24].

Le principe 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. [25] propose d'appliquer deux marques différentes en fonction de la taille atteinte par la file d'attente. Les paquets reçoivent une marque 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 ensuite de décider de la préemption éventuelle de certains flots. [26] est une alternative n'utilisant qu'un seul type de marquage pour définir les seuils d'admission de de préemption de flots.

Ces mécanismes nécessitent de standardiser à la fois les conditions de marquage ainsi que les informations de congestion elle-mêmes.

Les mécanismes de marquage reposent généralement sur des files virtuelles avec RED, ou sur des seaux à jetons, 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.

2.6.4 Conditionally Dedicated Bandwidth

ITU-T recommendation Y.1221 amendments specify the five following IP transfer capabilities (TC) of NGNs [20] : dedicated bandwidth (DBW) TC, conditionally dedicated BW (CDBW) TC, statistical BW TC, delay-sensitive statistical BW TC, and best-effort TC.

ITU-T Recommendation Y.1221, Traffic Control and Congestion Control in IP-Based Networks, Mar. 2002.

DBW = diffserv premium CDBW = focus losses on a small set of flows + congestion notification signals

Conditionally Dedicated Bandwidth (CDBW)[?] est une proposition alternative de gestion du trafic, basée sur un mécanisme de signalisation intra-bande léger. Parmi les possibilités offertes, le réseau permet des garanties de service pour les applications *streaming* de débit limité, et il peut informer la couche réseau du débit disponible sur le lien en vue d'une meilleure utilisation par les protocoles de transport. CDBW s'inspire fortement des mécanismes en mode circuit qui offrent un taux de perte négligeable aux connexions.

[?] en propose une implémentation nécessitant le maintien d'un état simple par flot. Notamment, une priorité à deux états est associée à chaque flot, qui détermine le sous-ensemble qui subira les pertes de paquets en cas de surcharge (cela peut être fait aléatoirement ou non, par exemple afin de protéger des appels d'urgence, etc). La priorité d'un flot peut augmenter sous certaines conditions, et, éventuellement, certains flots peuvent être déclassés lors d'une forte congestion. Ces priorités peuvent être interprétées comme des notions moins fortes d'acceptation ou de blocage d'une connexion.

La plupart des classes de trafic n'établissent pas de réservation de ressources, ce qui semble nécessaire pour les flots à débit variable, et permet une meilleure utilisation des ressources. Le maintien d'un état par flot et la signalisation peuvent être limitant pour une implémentation réelle sur des liens à forts débits, notamment pour traiter l'en-tête de QoS, mais ils restent néanmoins plus extensibles que dans le contexte Intserv/RSVP. Par contre, la surveillance du débit maximal des flots peut être complexe à réaliser efficacement.

2.6.5 Flow State Aware

Service contexts and flow specifications

Parameters for flow specifications currently defined are the following : flow identity, requested rate, preference priority, packet discard priority, and service contexts. There are four types of service contexts : maximum rate service (MRS), guaranteed rate service (GRS), available rate service (ARS), and variable rate service (VRS). GRS and MRS flows are similar to flows with guaranteed service in IntServ. flows are similar to flows with guaranteed service in IntServ. MRS differs from GRS in that it has the immediate transfer option, which allows a network to accept without admission control. ARS is similar to ATM available bit rate. Flows with ARS service context may modify their data rate, either

by the application request or the network demand. VRS is a combination of MRS and ARS, such that it is guaranteed a minimum data rate but can ask for more bandwidth on demand later. This rather complex differentiation of flows affects further flow characterizations. Flow identity may be defined by IP 5-tuples and DSCP. It may also be defined by the MPLS label. Therefore, the term “flow” in FSA may indicate either IP 5-tuple flows or aggregates of them. For flows aggregates, aggregation end points are able to create aggregate flow identification and notify the next FSA signaling nodes about the aggregate flow identity. Requested rate (RR) is, for MRS and GRS, the mean data rate the flow requests at which it should be served throughout the flow lifetime. For ARS and VRS, RR is the initial mean data rate; ARS may modify this rate later. For VRS, the initial RR is the minimum rate at which the flow should be served. Later, the flow may increase the mean data rate (with the same RR name). Preference priority is the priority for admission decision. It can be used for packet discard decision in some cases. Packet discard priority (known as “flow state” before the January 2007 ITU-T SG13 meeting) can have two different values, namely “discard first” and “discard last.” It is used for packet discard decision upon congestion.

C. In-Band Signaling FSA emphasizes the use of in-band signaling, although it is not a single mandatory signaling method. DiffServ code point (DSCP) has been suggested as a way to recognize in-band signaling packets. Both signaling packets and plain data packets for FSA should be recognizable, although the exact method has not been defined yet. Proxies may signal instead of end-systems. Active flow identification is also possible. Aggregate signaling can be performed at the edge of a network. There are several types of signaling packets, including request, response, confirm, renegotiate, and close. Several indication flags have been defined, including ignore indication for signal aggregation, changing direction indication, and QoS approval indication. An FSA router sets this indication if it has approved a request from a flow. A router may clear this indication to inform the edge that the request has not been approved.

D. Dynamic Congestion Notification and Data Rate Enforcement FSA requires the network to be able to notify the customer premise equipment of the congestion status or the network rate. The network rate is the desired data rate, which is less than the RR. In this regards, the FSA is similar in concept to ATM ABR rate control. The ABR rate control scheme is somewhat similar to SCORE, in the sense that (flow or network) information is carried by packets. The RM cell imposes RTT delayed response to network congestion.

2.6.6 Congestion Exposure

being standardized at IETF, an extension of the ECN solution in order to make congestion appear to the network and not to the end host only. the point is that flow rate fairness is not a desirable property [23], instead congestion fairness is

rate, volume, application type are not good to decide to drop traffic Triche en établissant plusieurs flots par exemple per-user congestion

re-ECN is a candidate solution. Router detects congestion, TCP ECN mark receiver repeats this mark in the IP field : debt sender need to use some credit by putting a mark in emitted packets discard is the balance of passing marks in debt

congestion based fairness ne signifie pas payer la congestion, sinon encouragement a sous dimensionner les réseaux. base de tarification qui encourage ISP a investir Refeed-back sticks to the Internet principle that the computers on the edge of the network detect and manage congestion. But it enables the middle of the network to punish them for providing misinformation. end hosts to optimize their internet usage prémisses dans le concept d'Internet auto-géré proposé par Kelly [45]. encourager les utilisateurs a mieux équilibrer leurs utilisations, utiliser des protocoles moins agressifs / plus réactifs a la congestion comme LEBDAT[?].

different from congestion charging proposed by Kelly weighted sharing to maximize value users get from their throughput Whois is receiver? How is credit managed? How is debt evaluated? Requires coopération between ISPs

2.6.7 Flow-Aware Networking

2.6.8 Quasi-Stateful Interv

2.6.9 Stateful Diffserv

2.6.10 Flow-aggregate-based services (FABs)

FABs [?]

We proposed a new QoS architecture called FABs. The FABs architecture introduces two novel concepts in QoS management. First, with IDFA, signaling on the data or control planes is exchanged so that the correlation among flow aggregates at contiguous networks is maintained as much as possible. Secondly, EIAC eliminates inefficiency that results from discarding packets in the middle of the path of a flow by congestion notification to the edge nodes. In addition to these novel concepts, FABs incorporates the best combination of building blocks for different congestion situations. Detailed algorithms for the building blocks will be further developed and proposed as standards in NGN architecture.




2.6.11 Constraint Based Routing

Traffic engineering Evolves from QoS routing
distribution of link state information complex computations except for hopcount and bw
routing table size complexity more resource consumption instability

2.6.12 IPX, operator managed internet


























IMS. etc no sound engineering class based diffserv, nor per flow guarantees make sense as traffic control.

2.7 Discussion

Le tableau suivant résume pour chaque architecture si elle répond aux différents besoins que nous avons présentés plus haut :  : OK,  : moyennement,  : pas du tout.

IntServ : per-flow weighted fair queuing and per-flow admission control DiffServ : per-class (a huge flow aggregate that lasts for a single hop) scheduling, per-flow admission control (or per-class rate limiting), and traffic engineering when collocated with MPLS (DiffServ over MPLS) FAN : per-flow fair queuing, implicit admission control without any signaling, and traffic engineering FSA : per-flow or per-aggregate weighted fair queuing + selective per-flow discard upon congestion, and admission control + selective per-flow discard upon congestion

2.7 Discussion

	BE	I	D	D_{fc}	BT	SMN	CP	FAN
Streaming guarantees			 ^b					
Elastic guarantees			 ^a					
Fairness								
Grain								
Scalability								
Control								
Congestion								
Overload								
Preemption								 ^z
Interdomain							?	

^a Best effort traffic will not be protected

^b Provided the overall charge of streaming flows is limited

^z Future work

Chapitre 3

Flow-Aware Networking (FAN)

L'assurance de garanties de service pour le trafic requiert la compréhension de la relation qu'il 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 de applications véhiculées par le réseau.

La proposition 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.

Ce chapitre complète le précédent, et montre plus en détail comment l'architecture FAN intègre les besoins que nous y avons exprimé, et ce de manière robuste grâce aux modèles de performance sous-jacents. Les travaux de cette thèse – qui se fondent sur les hypothèses et les modèles de trafic introduits ici – s'inscrivent dans la continuité des travaux autour de FAN avec l'objectif de converger vers une proposition de routeur efficace, robuste et bien dimensionnée. Il se termine par une description de l'environnement de simulation et la plateforme d'expérimentation GNU/Linux que nous avons réalisé à des fins de démonstration.

Contents

3.1	Introduction	30
3.2	Modélisation du trafic streaming	31
3.3	Modélisation du trafic élastique	32
3.4	Intégration des flots <i>streaming</i> et élastiques	37
3.5	L'architecture de différenciation implicite de service Cross-Protect . . .	38
3.6	Vers la réalisation d'un routeur Cross-Protect	41

Contributions :

- Extension de l'algorithme PFQ 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 un lien Cross-Protect (PFQ + Contrôle d'admission), les modèles de trafic utilisés, etc.
- Implémentation des mécanismes Cross-Protect dans le noyau Linux (dépôt logiciel effectué)
- 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

Démonstrations :

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

3.1 Introduction

L'architecture *Flow Aware Networking* (FAN) [65] propose une nouvelle approche de la QoS avec pour objectifs des garanties de performance pour les flots *streaming* et élastiques. FAN ne requiert pas de mécanisme complexe de marquage ou de signalisation, et permet de conserver l'interface *best effort* qu'ont les utilisateurs avec le réseau.

La gestion du trafic s'effectue au niveau des flots utilisateurs. Contrairement aux hypothèses de nombreuses études qui considèrent un nombre fixe de flots permanents, Roberts et Massoulié [77] 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. On caractérise généralement lors de la période de pointe, en vue du dimensionnement des ressources. Dans un tel contexte, ce sont des garanties statistiques qui semblent le plus appropriées pour mesurer la performance : nous nous intéresserons notamment au taux moyen de blocage des flots.

La modélisation de la performance dans FAN s'appuie sur des modèles simples permettant de comprendre la performance réalisée par chaque catégorie de flot. L'insensibilité de ces modèles permet un dimensionnement robuste des liens en fonction de caractéristiques générales du trafic telles que la charge, quelle que soit par exemple la distribution de la taille des flots.

Ces modèles sont à la base d'une théorie du trafic internet, à l'instar de la formule d'Erlang dont la proposition en 1917 a été le fondement du télétrafic dans le contexte du réseau téléphonique. Cette formule qui 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 :

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

Elle nécessite seulement que les arrivées des appels suivent une distribution 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 les changements qui s'en suivent au niveau du trafic.

Ces modèles permettent d'une part une explication qualitative de la performance actuelle des réseaux, mais fournissent également une base théorique pour évaluer l'in-

3.2 Modélisation du trafic streaming

introduction de nouveaux mécanismes. C'est le cas l'architecture de routeur Cross-Protect que nous introduisons. Cross-Protect réalise FAN par l'association d'un contrôle d'admission implicite basé sur des mesures et d'un ordonnancement équitable des flots (*fair queueing*). Notons qu'une déclinaison de ces mécanismes pour le réseau d'accès sera considérée dans le Chapitre ??.

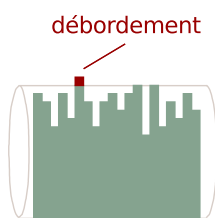
3.2 Modélisation du trafic streaming

3.2.1 Cas d'un lien isolé : multiplexage sans buffer

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 ($< 64\text{kb/s}$ 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. Nous envisageons un dimensionnement adéquat du buffer dans le Chapitre ??.

La principale difficulté d'analyse provient de la gigue¹ acquise par les flots lors de la traversée du réseau. Les multiplexages successifs dans les files d'attente des routeurs traversés créent des rafales de paquets qu'il est complexe de caractériser. Généralement, les modèles qui proposent de contrôler la charge moyenne du trafic ρ_s afin qu'elle ne dépasse pas la capacité C du lien : $E[\rho_s] \leq C$, ne sont pas adaptés à offrir des garanties de service (sauf pour des cas très conservateurs où en allouant une taille de buffer relativement importante).

Multiplexage statistique sans buffer



Il est toutefois possible de garantir une QoS au niveau paquet grâce à un contrôle de la charge au niveau flot dans les hypothèses du modèle de multiplexage statistique sans buffer, dit *Rate Envelope Multiplexing* [80]. Le buffer de taille minimale permet d'absorber les arrivées simultanées de paquets ou les excédents locaux, sous la condition que $P[\rho_s > C] \leq \epsilon$.

Le taux de pertes sur le lien est alors approximativement : $l = E[(\rho_s - C)^+] / E[\rho_s]$. La performance obtenue est insensible aux caractéristiques détaillées du trafic, telles que la durée des flots ou des rafales. Elle dépend uniquement de la charge moyenne offerte et du débit crête des flots.

Le taux d'utilisation du lien 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 1% de C ou moins). Comme nous allons le voir, la bande passante disponible peut avantageusement être utilisée pour le transfert de flots élastiques.

3.2.2 Contrôle d'admission pour les flots streaming

L'ajout d'un contrôle d'admission pour les flots *streaming* permet de garantir les conditions du multiplexage statistique sans buffer.

1. variation du délai

Probabilité de blocage

Considérons des flots de débit crête 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). La probabilité de blocage p_b dans ce modèle est alors donnée directement par la formule d'Erlang où ρ_s/r erlangs de trafic sont offert à C/r circuits :

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

3.2.3 Garanties de performance de bout en bout

Bonald *et al.* [19] 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, les flots n'acquièrent jamais une gigue suffisante pour avoir une performance moins bonne que des arrivées de Poisson de taille MTU.

Il est ainsi possible en limitant la charge des flots par un contrôle d'admission sur chaque lien, de garantir leur performance de bout en bout. Le seuil d'admission n'est pas critique si la proportion des flots *streaming* n'est pas trop importante, ce qui peut être le cas dans un réseau de cœur intégré. Nous nous intéressons au cas où la majorité des flots ont un débit crête limité dans le Chapitre ??, avec la proposition d'un algorithme de contrôle d'admission adapté.

Des généralisations de ces modèles existent pour des débits d'accès différents, ou des flots à débit variable.

3.3 Modélisation du trafic élastique

Les modèles de partage statistique de bande passante 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 [76] 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 [67] a par exemple montré que le débit moyen d'une connexion TCP est lié au taux de pertes subi. Cependant, l'abstraction du niveau paquet présente l'avantage de simplifier l'analyse, notamment au vu des propriétés d'autosimilarité du trafic, ainsi que de s'abstraire d'une version précise de TCP.

3.3.1 Modèle de sessions Poisson

Bonald *et al.* [9, 16] ont proposé un modèle assez général de sessions, permettant de représenter la plupart des caractéristiques du trafic Internet. Les sessions sont constituées d'une alternance de flots et de périodes d'inactivités, de distributions générales, avec des arrivées de flots pouvant 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 [57, 69]. Il est alors suffisant que les sessions arrivent selon un processus de Poisson, ce qui est reconnu comme l'un des rares invariants du trafic Internet [71], pour que la performance du réseau soit équivalente à un ensemble de files équivalentes où les arrivées de flots sont Poisson. La preuve est basée sur la construction

3.3 Modélisation du trafic élastique

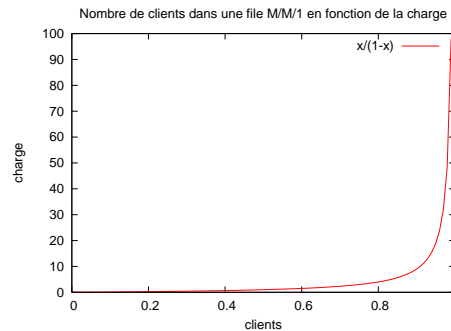


FIGURE 3.1: Nombre moyen de flots en cours dans une file M/M/1, en fonction de la charge ρ .

d'un réseau de Whittle adapté, qui admet un réseau PS équivalent où le comportement de chaque file est indépendant (on parle de réseaux Markoviens à forme produit).

3.3.2 Cas d'un lien isolé : modèle à processeur partagé

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 ?? 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.

Propriétés 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 performances 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 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 .

ref?

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 ??.

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

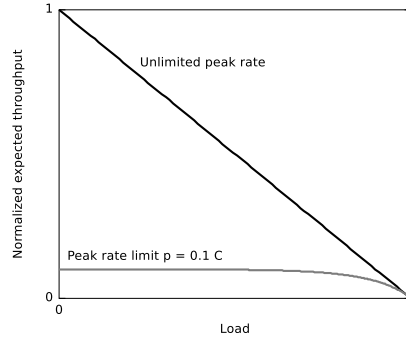


FIGURE 3.2: Débit normalisé des flots γ en fonction de la charge ρ lorsque les flots n'ont aucune limite de débit, et lorsqu'ils possèdent un débit limité (ici égal à $.1C$).

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 (ce nombre est proportionnel à $1/\gamma$). Une confirmation empirique du nombre limité de flots *bottlenecked*, indépendamment de la capacité du lien, est présentée dans [51], où les auteurs analysent plusieurs traces de trafic. Ce résultat renforce les premières observations de [83] 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. Dans le cas d'un partage équitable, γ peut également être interprétée comme l'espérance du débit instantané d'un flot à tout instant [13]. 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 :

$$\gamma = \frac{\sum \rho^i / \Phi(i)}{\sum i \rho^{i-1} / \Phi(i)}. \quad (3.1)$$

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 (proche de $(1 - p/C)$).**

3.3.3 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. Le comportement de la file d'attente est instable et implique que le temps de réponse des flots n'est plus borné. Il est intéressant de remarquer qu'au niveau paquet, le lien est toujours stable en raison des tailles limitées des files d'attente qui causent des pertes de paquets.

3.3 Modélisation du trafic élastique

Ben Fredj *et al* remarquent que l'accumulation des flots est plus lente lorsque la distribution des tailles de flots est à queue lourde. Les flots de petite tailles parvenant à terminer, au détriment d'une discrimination contre les flots plus longs. Ils modélisent également l'impatience des utilisateurs/applications qui stabilise le système en interrompant les connexions lorsque le temps de réponse devient trop important. Cette argumentation diffère de celle reposant sur les modèles d'utilité classique pour les flots élastiques où celle-ci est strictement concave, même lorsque le débit tend vers 0 [81].

Les abandons de flots entraînent un gaspillage des ressources du réseau, qu'il convient de prévenir au plus tôt. [1] propose un contrôle d'admission pour les flots élastiques en limitant le nombre de flots en cours se partageant la bande passante. 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* [11].

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 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 $\rho > 1$:

$$p_b(\rho, m) = \rho^m(1 - \rho)/(1 - \rho^{m+1}) \xrightarrow{m \rightarrow \infty} 1 - 1/\rho$$

Le choix du seuil d'admission n'est pas critique pour la performance dès que $m > 50$ [1]. Une valeur de m plus importante n'a pas d'impact sur la probabilité de blocage des flots, mais entraîne un gaspillage des ressources plus long le temps de détecter la surcharge. Il est donc bénéfique de choisir une valeur la plus petite possible, d'où le compromis $m = 100$ choisi dans [54].

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

Ces formules permettent le dimensionnement d'un lien en vue d'offrir un blocage négligeable aux flots. 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 ??.

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 différentes, où dans le cas d'un partage non-équitable (modèle processeur partagé discriminatoire (DPS, *Discriminatory Processor Sharing*); l'insensibilité est alors approximative.

3.3.4 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 $X = (n_1, \dots, n_K)$. Le vecteur des débits à long terme de chaque classe est $\Phi = (\phi_1, \dots, \phi_K)$, 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 vecteur Φ possibles ; c'est un polytope convexe défini par les contraintes posées par chaque lien ^{2 3}.

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 ⁴.

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

$$\begin{aligned} & \underset{x}{\text{maximize}} && \sum_{k=1}^K n_k U\left(\frac{\phi_k}{x_k}\right) \\ & \text{s.c.} && \left(\sum_{k \in r_l} \phi_k \leq C \right) (\forall l \in \mathcal{L}) \\ & && \phi_k / n_k \leq a_k, \forall k \in \llbracket 1, K \rrbracket \end{aligned}$$

On retrouve par exemple : *max throughput* ($U(\cdot) = \cdot$), *proportional fairness* [46] ($U(\cdot) = \ln(\cdot)$), *minimum potential delay* [79] ($U(\cdot) = 1/\cdot$), et *max-min* [12] qui est un cas limite d'une mesure d'utilité [62]. Les auteurs de [62] regroupent ces allocations, dites α -fair, sous une formulation mathématique commune. [56] généralise plusieurs notions d'équité (dont les allocations α -fair et l'indice de Jain) et donne un aperçu des différents compromis entre équité et performance réalisés par les allocations α -fair.

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 standards de TCP.

Stabilité

La région de stabilité S est définie par l'ensemble des vecteurs $\rho = (\rho_1, \dots, \rho_K)$ 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} . Il suffit ainsi que la charge de tous les liens soit inférieure à leur capacité [15], ce qui peut-être assuré par un contrôle d'admission. Les allocations basées sur l'utilité (ainsi que l'allocation balancée) 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 noeud 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 [77] 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 des métriques telles que leur temps de complétion. Il

2. Cela est valable pour un réseau filaire, ce n'est généralement pas le cas, par exemple pour un réseau sans fil avec interférences [60]

3. 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

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

3.4 Intégration des flots *streaming* et élastiques

suffit que l'allocation soit suffisamment équitable pour que la performance des flots soit satisfaisante [15].

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 [17] 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 flots⁵ 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 $k = 1, \dots, K$.

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

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

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

Bornes stochastiques de performance

Malgré sa propriété intéressant d'insensibilité pour un dimensionnement robuste des liens, l'allocation balancée reste toutefois complexe à évaluer dans un contexte pratique, puisqu'elle dépend de la demande sur toutes les routes et des capacités des liens. [18] propose des bornes stochastiques sur le temps de réponse des flots, permettant d'évaluer de manière conservative 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 or *proportional fair*. [51] donne notamment un exemple de l'utilisation de *balanced fairness* afin d'approcher la performance de *max-min*.

L'architecture *Flow-Aware Networking* réalise *max-min* grâce à l'emploi de *fair queueing* [77], ce qui semble être un choix acceptable et dont on peut approximer la performance à l'échelle du réseau grâce à BF.

3.4 Intégration des flots *streaming* et élastiques

Le chapitre précédent a traité de l'intérêt de servir les flots *streaming* en priorité, et de laisser les flots élastiques utiliser la capacité disponible, grâce à un mécanisme d'ordonnancement approprié [9]. 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 [49] 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. [63], [27] et [50] évaluent la performance d'un réseau où les flots *streaming* non-adaptatifs sont traités en priorité. [27] donne 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

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

dans un tel contexte, ainsi que de fournir des bornes de performances insensibles pour le temps de réponse d'un flot, très étroites dans des conditions réalistes de trafic. Les auteurs montrent également que la présence de plusieurs liens goulottés peut conduire à une forte perte de capacité dans le réseau, qui reste toutefois faible lorsque les flots ont un débit limité comme c'est souvent le cas en pratique.

Enfin, [20] considère le contexte d'un ordonnancement équitable de type *fair queueing*. Les auteurs discutent de l'intérêt de donner la priorité aux paquets, suggérant que la valeur ajoutée est négligeable sur les liens à fort débit.

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

Cross-Protect réalise l'architecture FAN avec la combinaison d'un mécanisme de contrôle d'admission au niveau flot et un algorithme d'ordonnancement de paquets (*Priority Fair Queueing*). Cette association permet d'offrir des garanties de performances aux flots *streaming* et élastique sans nécessiter un marquage explicite des paquets.

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 ; et en contrôlant la charge des flots *streaming* et élastique, il permet de conserver la performance des flots et d'assurer l'extensibilité de l'ordonnanceur.

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, 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.

Principes de l'algorithme PFQ

L'algorithme *Priority Fair Queueing* (PFQ) est une extension de l'algorithme *Start-Time Fair Queueing* (SFQ). 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.

Les flots dont le débit est supérieur au débit équitable voient leurs paquets s'accumuler dans la file d'attente du routeur jusqu'au dépassement des pertes de paquets. L'algorithme d'ordonnancement est responsable de partager la bande passante laissée disponible entre ces flots. PFQ intègre une gestion de la file d'attente qui oriente les pertes sur les flots qui ont accumulé le plus grand nombre de paquets, afin de déclencher les mécanismes de contrôle de congestion de TCP. 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 du moment 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.

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

Nous avons montré précédemment que le nombre de flots goulottés sur un lien est généralement faible jusqu'à de fortes charges. Typiquement, moins de 20 flots à débit non limité sont en compétition avec une probabilité de .99 pour une charge de 80%. Ce nombre est plus important pour des flots à débit limité, mais [51] montre à partir d'un modèle confirmé par des traces de trafic réelles 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* PFQ, comme PDRR [55] (*Priority Deficit Round Robin*), qui se base sur DRR (*Deficit Round Robin*) et présente l'avantage d'avoir une complexité en $O(1)$. Si elle est similaire en théorie, cette réalisation se révèle 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 fr_i .

Lorsque qu'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 que MTU octets 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 qui sont terminés lors d'un changement de cycle, qui se manifeste par le changement de l'empreinte du paquet en tête de file.

3.5.2 Indicateurs de congestion

Le fonctionnement naturel de la file d'attente PIFO 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.

$$pl = \Delta B / \tau PL = (1 - \alpha)pl + \alpha PL$$

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

$$fr = MAX(\Delta vt, SxC) / \tau PL = (1 - \beta)fr + \beta FR$$

où vt est le temps virtuel de la file, c'est-à-dire l'empreinte temporelle du paquet en tête de file et S la durée pendant laquelle la file est vide sur l'intervalle τ . 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 ?? le besoin de définir un algorithme plus avancé pour les cas fréquents où le lien véhicule 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 ?. 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 Measurement-Based Admission Control

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 [? ?], en rejetant l'excédent de trafic. Il doit être transparent dans des conditions normales de charge.

Contrôle d'Admission Basée 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 que ces derniers sont très complexes à définir en raison du comportement multifractale du trafic IP, ces algorithmes conduisent à une faible utilisation du lien. En plus des besoins de signalisation, *policing* et de la difficulté

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 recours à un protocole de signalisation ni besoin de vérifier la conformance du trafic. Il permet notamment de s'adapter 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 en le maintien d'une liste des flots protégés (identification au vol, mécanisme de purge). Nous aborderons rapidement ces considérations dans la section ??, et nous proposerons un algorithme plus évolué de contrôle d'admission dans le chapitre ?.

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.

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.

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 fixe cette valeur à 70% de la capacité du lien. Dans le chapitre ??, nous reconsidérerons cet aspect lorsque les flots sont majoritairement à débit limité, comme sur un lien de collecte ADSL par exemple.

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.

mettre plus en évidence

3.6 Vers la réalisation d'un routeur Cross-Protect

3.6.1 Evaluation de l'architecture en simulation

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

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

3.6.2 Evaluation 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 du trafic du noyau Linux (iptables [?] et `traffic control` (TC) [?]), afin de construire une plate-forme d'expérimentation et d'évaluation de la solution. Les objectifs d'une telle réalisation sont multiples :

- démontrer le fonctionnement de Cross-Protect dans des conditions réelles,
- analyser l'évolution des différentes structure de données (notamment les tables de flots) ;
- convaincre de la faisabilité de l'architecture, et notamment du traitement des flots, à haut débit ;
- 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 éventuellement 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.

Plate-forme d'expérimentation

Utilisation et extension d'outils de génération de trafic (`gen_flot`), de visualisation (communs entre le simulateur et plate-forme).

Evaluation

Cette évaluation expérimentale de l'architecture Cross-Protect au travers de la plate-forme GNU/Linux a pu donner lieu à deux démonstration, présentées dans l'introduction de ce chapitre.

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, repose considérablement sur 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.* [5], 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 au résultat de simulations ns2 que, tandis qu'un tampon 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 tampon dépend de manière significative du débit crête exogène des flots multiplexés.

Contents

4.1	Introduction	44
4.2	Dimensionnement des buffers et TCP	46
4.3	Etat de l'art sur le dimensionnement de buffers	47
4.4	Impact de la taille du buffer sur la performance des flots	49
4.5	Dimensionnement des buffers dans le régime transparent	51
4.6	Dimensionnement des buffers pour le régime élastique	53
4.7	Conclusions	62

Contributions :

- Unifier les hypothèses
- Importance d'un mix réaliste de trafic (d'ailleurs traces...)
- Unification flots longs et flots courts
- Débit crete et charge des flots important !
- Comprendre pourquoi petits buffers suffisent tant que access « cœur
- Potentiellement un dimensionnement pour un scénario avec forts débits crete

Publications et présentations :

Ce chapitre a 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, June 17-21, 2007 - Ottawa, Canada*

4.1 Introduction

Le dimensionnement des buffers au sein des routeurs a reçu beaucoup d'intérêt récemment [5, 73, 87, 72, 30, 3, 2], et notamment la règle empirique dite du *Bandwidth Delay Product* initialement proposée par [85]. La réalisation de si grands buffers est vue comme un challenge pour l'augmentation de la capacité des liens à 40Gb/s, voire même impossible dans le cadre des routeurs purement optiques qui devraient apparaître bientôt. L'évolution des capacités des liens de cœur de réseau et de leur trafic remettrait en cause la pertinence de cette règle empirique, notamment au vu des coûts de revient comparés de la bande passante et de la mémoire.

Qualité de service

Lors des périodes de congestion, le lien est utilisé à 100% et des paquets s'accumulent dans le buffer. Sans discrimination du trafic, les flots *streaming* subissent – outre les pertes dues au débordement du buffer – des délais très importants causés par les paquets en attente, et aggravés par le comportement *bursty* des connexions TCP. Il est tentant de vouloir réduire les buffers afin de minimiser cet impact sur les flots *streaming*, mais les conséquences d'une telle réduction sur la performance du trafic élastique restent un sujet d'étude, auquel nous consacrons la plupart de ce chapitre.

[6] présente 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, qu'il n'est pourtant pas facile de caractériser, étant donné la complexité et la variabilité du trafic. 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

4.1 Introduction

de prendre une valeur moyenne pour le RTT de 200 à 250ms (de l'ordre du temps de propagation transatlantique).

Architecture des 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 [5], il s'avère que pour un lien à 10Gb/s dont le RTT moyen est de 250ms, la taille requise pour les buffers est de 2.5Gb. La présence de tels buffers dans les routeurs pose des problèmes 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, où l'on ne sait que réaliser des buffers de quelques dizaines de paquets, montre l'intérêt de comprendre la performance du trafic en présence de telles tailles de buffers.

Pour une 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 notamment pourquoi ils parviennent à des 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.

Performance des flots

Nous avons vu que le trafic Internet est constitué d'une superposition dynamique de flots de taille finie. Une caractéristique importante des flots partageant un lien donné est leur débit crête. Il d'agit du débit que les flots obtiendraient si le lien était de capacité infinie. Certains flots avec un fort débit crête seront goulottés (*bottlenecked*) par le lien considéré et partageront la bande passante disponible au travers de mécanismes de contrôle de congestion de bout en bout. Cependant, la grande majorité des flots ne sont pas *bottlenecked* parce que leur débit crête, déterminé par exemple par un lien d'accès à faible débit, est beaucoup plus petit que le débit équitable. Le nombre de flots *bottlenecked* n'est pas une caractéristique de trafic exogène mais résulte plutôt du processus de partage statistique de bande passante, et peut être caractérisé en tant que fonction de la charge globale sur le lien.

Dans ce chapitre, nous évaluons des modèles simples de partage statistique de bande passante et identifions deux principaux régimes opérationnels importants pour le dimensionnement des buffers. On parlera de régime transparent lorsque le lien ne pourra devenir saturé qu'avec une probabilité négligeable. Lorsque par contre certains flots possèdent un débit crête suffisant pour saturer le lien, on parlera de régime élastique : de très nombreux flots *bottlenecked* constitueront une charge dite *background*, tandis que quelques flots *non-bottlenecked* se partageront la capacité laissée disponible. Ce modèle sera raffiné en considérant les cas où un flot seul ne peut saturer le lien, mais que plusieurs flots présents simultanément peuvent se combiner afin de le saturer. Tandis que de petits buffers suffisent dans le régime transparent, il semble nécessaire de considérer la capacité des liens pour le dimensionnement des buffers en régime élastique. Nous commençons par une revue de la littérature existant à ce sujet.

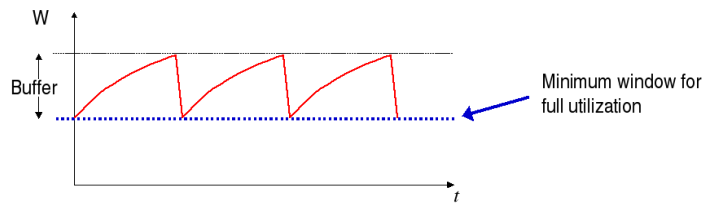


FIGURE 4.1: Evolution de la taille de la fenêtre de congestion TCP sur un lien avec buffer.

4.2 Dimensionnement des buffers et TCP

Le trafic observé sur un lien de cœur de réseau est majoritairement composé de flots TCP (jusqu'à 90%). 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; d'où l'émergence de la règle dite du *Bandwidth Delay Product* (BDP) que nous allons présenter. 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 1 flot TCP

La règle empirique du BDP a été proposée par Villamizar et Song [85] 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 connection TCP suffisamment longue¹.

La figure 4.1² représente l'évolution typique de la taille de la fenêtre de congestion d'une connection 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 connection n'émettra pas.

Puisque $cwnd$ régle l'envoi des paquets pendant chaque RTT, le débit d'une connection 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 donc être suffisante pour permettre l'émission de paquets au débit C durant cette période : $B \geq cwnd_p/2$ paquets. Le cas idéal est que le buffer est complètement vide lorsque la source émet à nouveau au débit du lien : $C = cwnd_p/2RTT$. On a ainsi $cwnd_p = 2C \cdot RTT$ et retrouve donc $B = C \cdot RTT$.

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

1. c'est-à-dire pour que la connection 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 [5] à la conférence Sigcomm'04.

4.3 Etat de l'art sur le dimensionnement de buffers

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 entraîne 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 connaît également une évolution en dent de scie, pour justifier l'application du *Bandwidth Delay Product*.

Ce phénomène est généralement exhibé avec des connexions permanentes, mais est moins connu avec du trafic aléatoire (arrivées Poisson de session par exemple). Il est remis en cause dans [5] dans le cas d'un lien de cœur de réseau où de très nombreuses connexions sont en cours simultanément, ce que nous verrons semble irréaliste. L'étude est approfondie dans [73] afin d'établir le lien entre la synchronisation des flots et le problème de dimensionnement des buffers.

Si dans la suite de ce chapitre nous allons uniquement nous consacrer à l'étude d'un environnement FIFO/DropTail, il convient de garder à l'esprit que les résultats sont fortement conditionnés par la discipline de service.

4.3 Etat de l'art sur le dimensionnement de buffers

La problématique du dimensionnement des buffers, si elle a pu être considérée plus tôt, a suscité un vif intérêt suite aux travaux de Appenzeller *et al.* [5]. 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 ([?] par exemple). 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 et permet aux flots streaming d'être isolés des flots élastiques.

4.3.1 Dimensionnement proportionnel au nombre de flots

La proposition de Morris [?] s'appuie sur la constatation suivante : si une connexion TCP à 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*.

Il est possible d'éviter ce régime de fonctionnement en assurant que chaque connexion puisse avoir quelques segments dans le buffer, d'où une taille suggérée proportionnelle 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 buffer 9 fois proportionnel.

Toutefois, l'article ne précise pas quels flots considérer, et nous pouvons supposer qu'il s'agit de ceux qui partagent effectivement la bande passante, et que nous qualifions de *bottlenecked*.

4.3.2 Vers une réduction drastique de la taille des buffers

Appenzeller *et al.* [5] 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. L'équité entre les connexions sera intéressante à considérer dans ce cas de tailles de buffers 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* à un comportement lissé, 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. [5] utilise un tel modèle afin d'estimer la probabilité de sous-utilisation du lien.

Ils 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, et les auteurs montrent la dépendance du dimensionnement à la charge et à la longueur des bursts.

L'étude d'un modèle mixte mélangeant flots longs et flots courts est complexe. Une hypothèse avancée et confirmée au travers de simulations est qu'il est suffisant de ne considérer uniquement 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 cette hypothèse plus formellement.

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

Quelques remarques sur les modèles de trafic

Il a été remarqué, cependant, par Raina et Wischik [73] ainsi que Raina et al. [72] que la synchronisation des flots dépend de la taille du buffer et que, pour un très grand nombre de flots, le dimensionnement proposé dans [5] 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 [5] 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 [86] précise que des buffers de 20 paquets suffisent si les flots sont *paced*, mais que dans le cas contraire il faut plutôt prévoir une taille de 20 bursts. 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*. Il 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 *close loop* qui correspondent à notre notion de partage statistique de bande passante.

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

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. [30, 31] 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. Nous reviendrons sur ces versions dites *paced* de TCP dans le prochain chapitre.

Lorsque les paquets sont espacés au débit moyen déterminé par la taille de la fenêtre plutôt qu'émis en rafales, [31] suggère que la taille du buffer doit être proportionnelle au logarithme de la fenêtre de congestion TCP maximale.

Cette hypothèse est licite pour de nombreux liens et correspond à ce que nous appelons le régime transparent (Ch.2, Sec.2.4) : tous les flots sont *non-bottlenecked*. Nous remarquons que les analyses dans [73, 72] 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 Importance du modèle de trafic pour évaluer la performance des flots

Si l'utilisation globale du lien est préservée pour des buffers de 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.

Il apparaît nécessaire de considérer un modèle de trafic réaliste tel que présenté dans le chapitre 3. 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 qui y est décrit :

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 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 induite par les flots *streaming* reste inférieure à la capacité du lien.

Lorsque le lien n'est pas saturé, les flots *streaming* étant seuls ou en compétition avec des flots élastiques tels que leurs débits crêtes ne saturent pas le lien, leur taux de pertes est négligeable (aux instants de débordement du buffer) pourvu que le buffer soit suffisamment dimensionné, mais ils subissent des délais dus aux paquets présents dans le buffer.

Lorsque des flots parviennent à saturer le lien, les flots *streaming* connaissent des pertes lorsqu'ils arrivent à des instants où le buffer est saturé (notamment pendant les rafales d'arrivées de paquets de flots à fort débit), et les paquets qui ne sont pas perdus ont un délai dépendant de la taille du buffer à cet instant. Les oscillations du buffer (voir [73]) augmentent également la gigue.

Nous remarquons que l'utilisation de l'ordonnancement PFQ permet de résoudre ces problèmes de délais et de gigue pour les flots *streaming* (supposés avoir un débit toujours inférieur au fair rate) en les isolant des flots saturant le lien (traitement en priorité), et donc de rendre leur performance indépendante de la taille du buffer. Les seuls flots élastiques traités en même temps sont ceux de plus faible débit et qui ont une influence mineure sur leur performances.

4.4.3 Performance des flots élastiques

Considérons d'abord les flots qui se verraient *non-bottlenecked* pour une allocation max-min sur le lien. Ces flots là émettent des paquets de temps en temps et ne se retrouvent que temporairement *backlogged* lorsque le buffer est saturé. Globalement, ils ne participent que faiblement au partage de bande passante fait par TCP et conservent leur débit exogène malgré quelques pertes.

Ce sont principalement les flots de plus fort débit (et l'ensemble des flots à considérer dépend de la charge du lien) qui se partagent effectivement la bande passante laissée disponible par les autres flots. Ces derniers peuvent être considérés approximativement comme un trafic de fond de charge donnée³. Ce sont ces derniers flots qui se partagent principalement l'espace du buffer et qui donc utilisent effectivement les mécanismes de contrôle de congestion et de partage de bande passante de TCP. Les modèles qui considèrent un très grand nombre de flots permanents, comme par exemple [5], ne correspondent pas à ce que l'on peut observer sur un lien de cœur de réseau à un niveau de charge nominal.

Une analyse de l'équité du partage entre les flots est faite dans le chapitre 5. Encore une fois, l'utilisation de l'ordonnancement PFQ permet l'isolation entre les flots, et notamment entre les flots qui saturent le lien et les autres ; nous verrons également dans le chapitre suivant que PFQ permet un multiplexage efficace des flots et garantit leur équité.

Dans la suite de l'évaluation, nous acceptons comme dans [5] 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 capacités plus importantes. Nous allons pour les régimes transparent et élastique examiner si la règle du BDP reste de rigueur ou non, comme annoncé précédemment.

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 également faible pour flots élastiques de plus faible débit, et qu'elles sont distribuées sur un petit nombre de flots et ont une influence négligeable sur la charge.

4.5 Dimensionnement des buffers dans le régime transparent

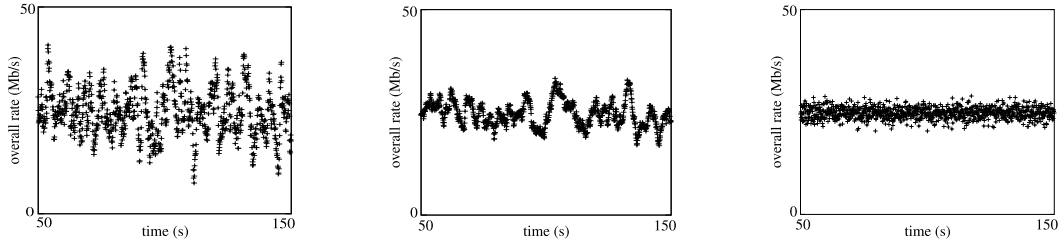


FIGURE 4.2: Processus d'arrivée des paquets : débit des paquets arrivant dans des intervalles successifs de 100ms pour des flots de débit crête $p = 200\text{Kb/s}$, $p = 50\text{Kb/s}$ and $p = 0$.

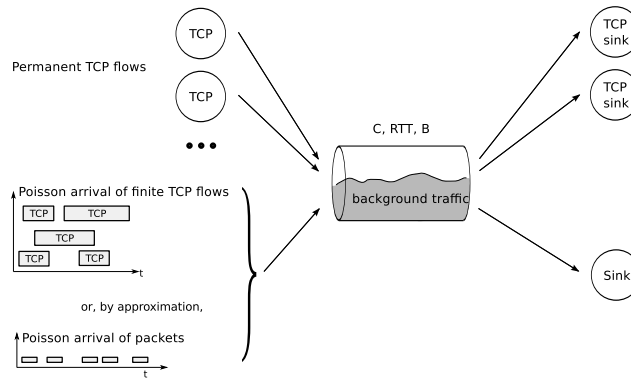


FIGURE 4.3: Environnement de simulation : sauf précision supplémentaire, les simulations utilisent les paramètres suivants : $C = 50\text{ Mb/s}$, $B = 20$ paquets, $RTT = 100\text{ ms}$, $\rho_b = 0.5C$, ordonnancement FIFO, paquets de 1000 octets.

4.5 Dimensionnement des buffers dans le régime transparent

Le régime transparent est caractérisé par le fait que la somme des débits crête des flots est, avec une forte probabilité, inférieure à la capacité du lien. Le buffer est dimensionné afin d'éviter un taux de pertes trop important dû à des arrivées simultanées de paquets provenant de flots indépendants. Nous supposons que le débit 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, $p = 200\text{Kb/s}$ and $p = 50\text{Kb/s}$, ainsi que celui mesuré pour un processus de Poisson.

Un processus d'arrivées 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

	$\epsilon = 0.01$		$\epsilon = 0.001$	
	$\rho = 0.5$	$\rho = 0.8$	$\rho = 0.5$	$\rho = 0.8$
$p = 0.1$	7	?	11	?
$p = .01$	7	21	10	32
$p = .001$	7	21	10	31
$p = 0$ (M/M/1)	7	21	10	31
$p = 0$ (M/M/1/B)	7	21	10	31

TABLE 4.1: Buffer sizes in transparent regime

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 encore 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

- L'approche la plus simple que nous avons écarté précédemment est souvent utilisée. Elle consiste à supposer que le processus d'arrivée global est poissonnien, et de dimensionner B de telle façon que $\rho^B < \epsilon$ ($\rho = E[\Lambda_t/C]$).
- Une meilleure approche pour le dimensionnement des buffers est alors de calculer un taux de pertes moyen en conditionnant sur la distribution $F(\lambda)$ de Λ_t et en demandant que $\int (\lambda/C)^B dF(\lambda) < \epsilon$. Ceci est raisonnable lorsque les variations de débit sont si rapides que ϵ est une bonne mesure de la performance, quel que soit le flot considéré.
- Enfin, il est également possible de faire en sorte que le taux de pertes reste inférieur à un certain seuil, calculé pour une valeur maximale Λ_{\max} atteinte par Λ_t telle que $(\Lambda_{\max}/C)^B < \epsilon$. Afin de s'assurer que ce débit n'est jamais dépassé (presque sûrement), des prévisions de trafic peuvent permettre de déterminer la capacité disponible sur le lien, ou mieux, de mettre en œuvre des mécanismes de contrôle d'admission.

Le tableau 4.1 présente les tailles de buffer nécessaires pour un taux de pertes fixé, en utilisant la deuxième approche (c'est-à-dire en moyennant sur les variations du débit). Nous supposons une distribution Gaussienne de Λ . Lorsque tous les flots ont le même débit crête, le ratio de la variance sur la moyenne est égal à ce débit crête.

Il s'avère que, pour un débit crête inférieur à $.1C$ et pour des valeurs de $\epsilon > .001$, la taille nécessaire du buffer est la même que celle qui aurait été nécessaire pour le processus d'arrivées Poisson. En d'autres termes, la formule de la file M/M/1 $\rho^B < \epsilon$ est une bonne indication de dimensionnement des buffers. Par exemple, un buffer de 20 paquets limite la charge admissible à $\rho = .79$ pour $\epsilon = .01$ ou $\rho = .7$ pour $\epsilon = .001$.

[86] donne également une justification de cette hypothèse Poissonnienne, qui peut paraître contradictoire avec les études montrant une dépendance à long terme au sein du trafic Internet. Elle est licite puisqu'ici nous considérons de petites échelles de temps – la dépendance à long terme n'apparaît que si l'on considère de plus grandes échelles de

temps – et d’autant plus que de petits buffers seront souvent saturés. Les variations de la file d’attente seront très rapides, comparativement à la charge du lien (plusieurs RTT pour cette dernière), et la distribution du taux de pertes 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 aggrégeant d’autres connections. Il se peut alors que certains flots saturer (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, lorsque la charge du lien n’excède pas 1 (auquel cas on se trouve en régime instable).

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 *non-bottlenecked* se combinent avec la charge *background* pour saturer momentanément le lien pendant des périodes qui sont longues comparées à l’échelle 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 *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. Ces pertes 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, et nous verrons en quoi le *fair queueing* permet d’assurer une protection de ces flots.

En ce qui concerne les flots *non-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 ; ce qui nous permettra d’obtenir quelques pistes 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 *bottlenecked* représenteront une charge dite *background* qui est par hypothèse fixe (régime quasi-stationnaire), le reste de la bande passante sera partagé par les flots *non-bottlenecked*. Les flots *bottlenecked* sont en grand nombre et possèdent chacun au maximum un seul paquet dans le buffer. Si leur taux de perte n’est pas trop élevé, ces dernières ne devraient pas impacter la charge offerte au lien. Dans le chapitre suivant, la présence de *fair queueing* sur le lien rendra cette hypothèse plus robuste.

Le scénario de simulation s’appuie sur une topologie *dumbbell* (Fig. 4.4) 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 que nous discutons plus loin ne semblent pas affectés par cette simplification.

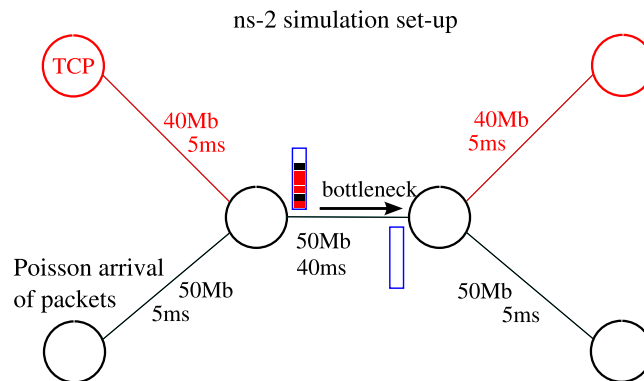


FIGURE 4.4: Simulation scenario

Dans les simulations qui suivent, le trafic *background* occupe la moitié de la capacité disponible. Nous avons simulé 1, 2 et 4 flots TCP NewReno permanents, chargés de représenter différents états du lien en régime quasi-stationnaire (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 [73]) 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.

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

La figure 4.5 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 : 20 paquets ([73]), 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 (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 4.6. 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 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* avec la figure 4.7. Chaque point dans la moitié inférieure représente un paquet émis par un flot *background*, tandis que ceux de la moitié supérieure appartiennent au flot *non-bottlenecked*. La position sur l'axe des abscisses in-

4.6 Dimensionnement des buffers pour le régime élastique

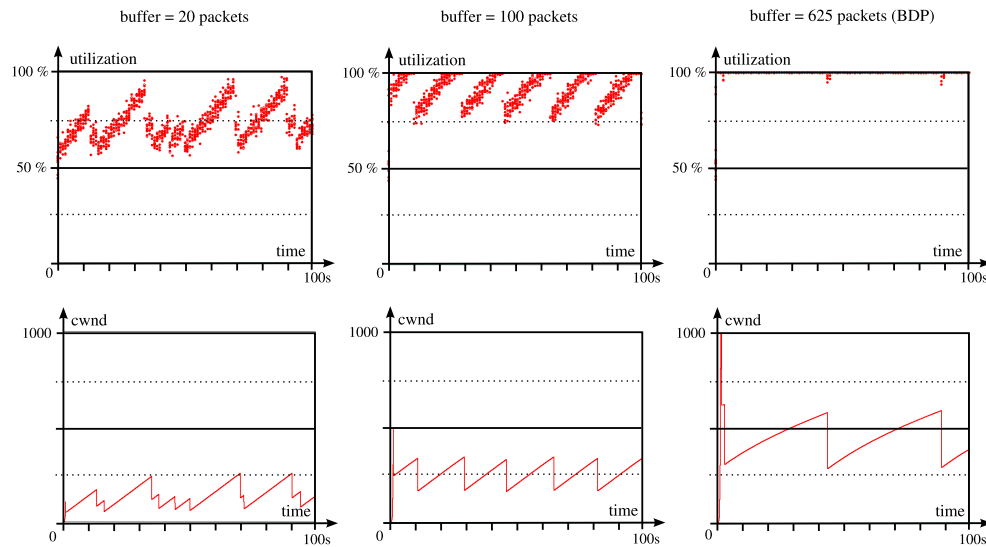


FIGURE 4.5: De gauche à droite, taux d'utilisation et taille de la fenêtre de congestion (*cwnd* en fonction du temps, pour différentes tailles de buffer : 20, 100 and 625 paquets (BDP).

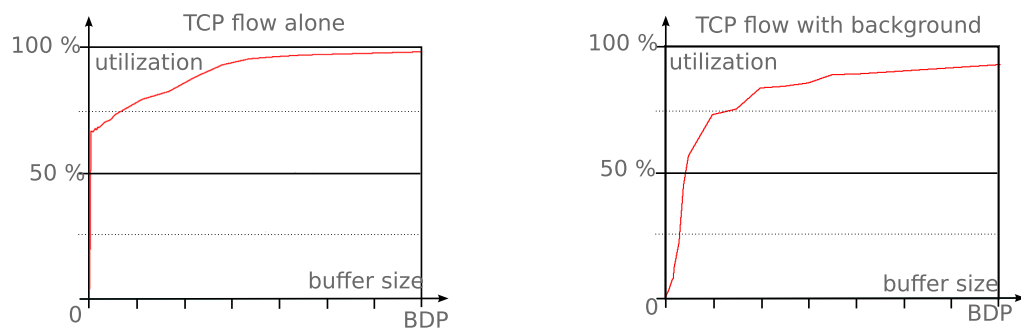


FIGURE 4.6: Comparaison de l'utilisation de la bande passante disponible par un flot TCP, avec ou sans trafic concurrent

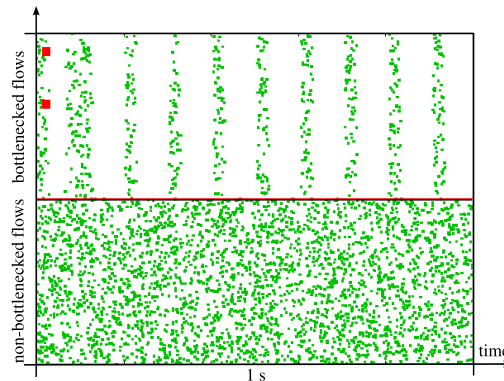


FIGURE 4.7: Représentation des arrivées de paquets TCP NewReno sur un lien avec un buffer de 20 paquets. Chaque point est un paquet ; sa position verticale est aléatoire ; les carrés sur la gauche représentent des paquets perdus.

dique 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*) [4] ; ainsi que des "rafales sub-RTT", dont le débit respecte le débit équitable, mais qui sont émises pendant intervalle plus court que le RTT [42].

Tant que la fenêtre TCP reste petite comparée au *Bandwidth Delay Product* résiduel $C(1 - \rho_b) \times \text{RTT}$, 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 (à cause des rafales). 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 *non-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 représenté. Elle est détectée un RTT plus tard ce qui mène à la diminution de moitié de la taille de 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, couplées à la lenteur de la croissance de la fenêtre de congestion dans l'algorithme AIMD. Nous savons que TCP n'est pas adapté aux cas de fort *Bandwidth Delay Product*, mais le grand buffer que nous avons choisi ici (625 paquets) permet 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*.

4.6 Dimensionnement des buffers pour le régime élastique

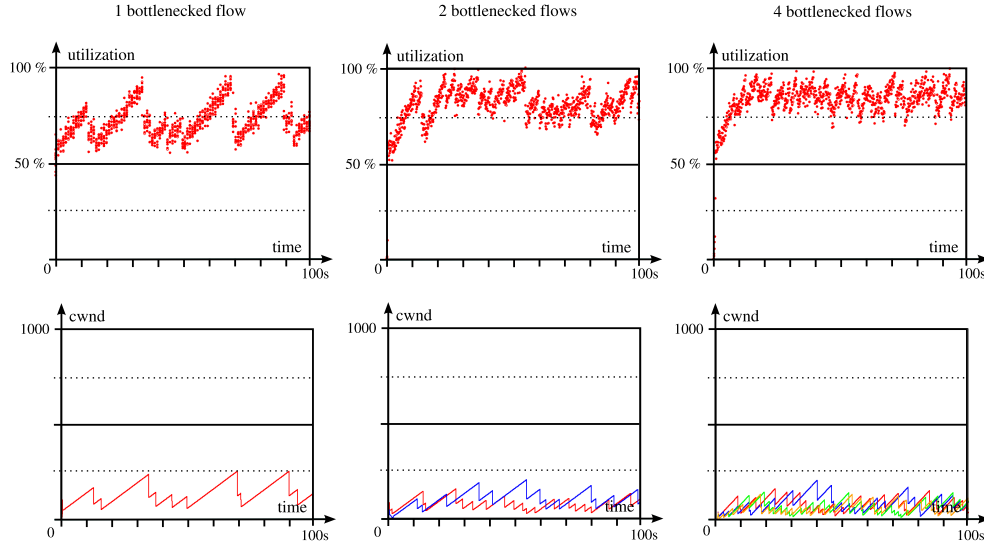


FIGURE 4.8: From left to right, utilization and cwnd size as a function of time for 1, 2 and 4 bottlenecked flows with a 20 packet buffer. All flows use TCP New Reno and the scheduling discipline is FIFO.

Multiplexage de plusieurs flots *bottlenecked*

La figure 4.8 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 approximativement 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. À 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. Il s'agit d'une part de comprendre l'impact de buffers réduits (qui peuvent être une contrainte technique, par exemple pour des buffers optiques), ainsi que 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 la performance du débit des flots, nous procédons comme suit. Pour une capacité de lien, une taille de buffer et une charge *background* donnés, nous simulons successivement un nombre de flots TCP *bottlenecked* permanents. Pour chaque nombre i (entre 1 et 500, ce qui est suffisant), 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

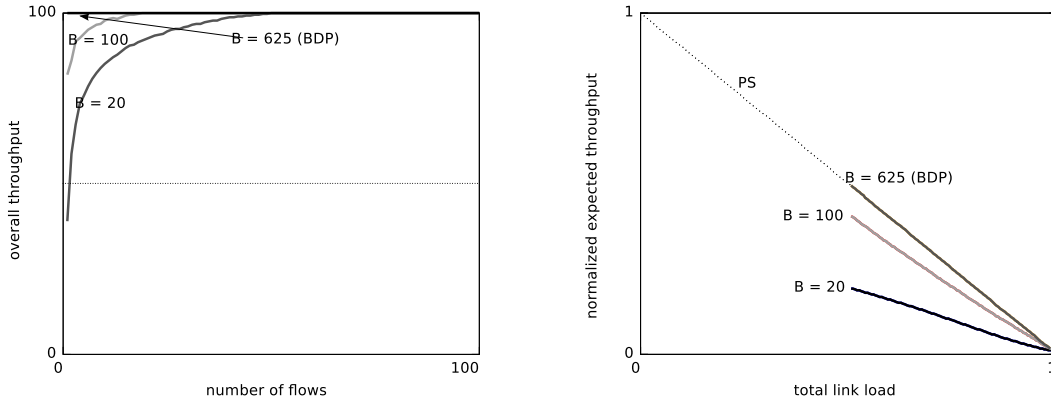


FIGURE 4.10: Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours; et espérance du débit γ d'un flot en fonction de la charge ρ , pour différentes tailles de buffer

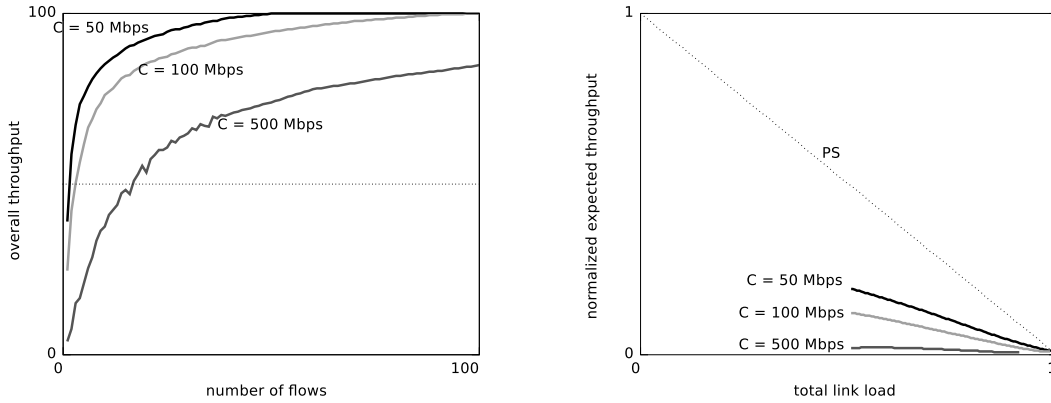


FIGURE 4.11: Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours; et espérance du débit γ d'un flot en fonction de la charge ρ , pour différentes capacités de lien

débit des flots γ par la formule (3.1 présentée au chapitre précédent). Ceci correspond à une analyse quasi-stationnaire qui nous permet d'ignorer des phénomènes tels que limitations de débit dues au *slow start* et les inéquités temporaires entre les flots. Cela revient trouver la probabilité stationnaire du système représenté en figure 4.9, afin de calculer l'espérance du nombre de flots.

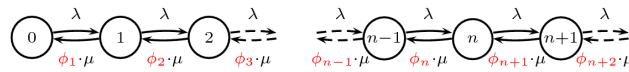


FIGURE 4.9: File d'attente représentant le nombre de flots en cours, pour un régime élastique régi par TCP

Les figures 4.10, 4.11, 4.12 représentent les valeurs de $\phi(i)$ et γ en fonction de la charge du lien ρ 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 *non-bottlenecked*.

4.6 Dimensionnement des buffers pour le régime élastique

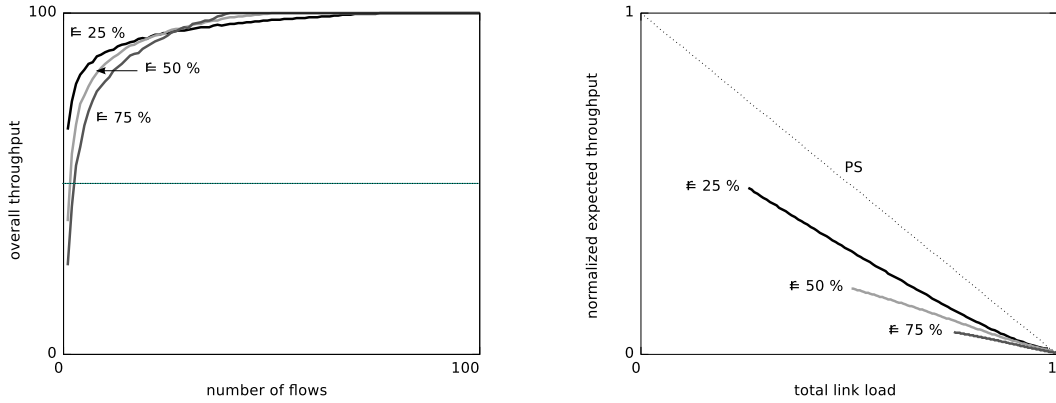


FIGURE 4.12: Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ , pour différentes charges de trafic *abckground*

Exprimons le débit moyen γ d'un flot dans le cas d'un partage équitable [13] :

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

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.

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 $\rho = 1$.

Les résultats montrent qu'il y a une chute significative du débit avec de petits buffers (fig. 4.10) et que cette perte est encore plus marquée quand la capacité du lien augmente (fig. 4.11). Plus la charge *background* est importante, plus il est difficile pour TCP d'utiliser la bande passante résiduelle (fig. 4.12).

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* ainsi que celle due aux flots TCP). Si peu d'articles évaluent la performance d'une connexion TCP dans les conditions que nous avons présentées, [44] propose toutefois un modèle analytique de TCP Reno dans un tel contexte, lorsque le trafic *background*

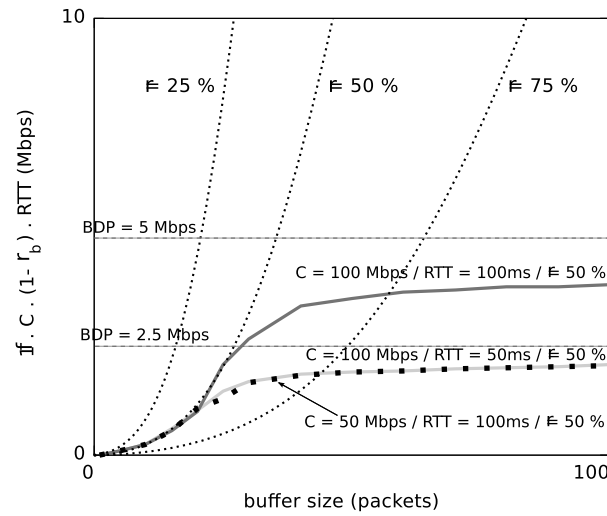


FIGURE 4.13: Throughput performance as function of buffer size for one bottlenecked flow

possède suit un processus de Markov par groupe à temps discret⁴. 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 4.13 représente le produit $\phi(1)C(1 - \rho_b)RTT$ 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 ($C=50\text{Mb/s}$ et $RTT=100\text{ms}$; $C=100\text{Mb/s}$ et $RTT=50\text{ms}$; $C=100\text{Mb/s}$ et $RTT=100\text{ms}$) ; 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

4. plus connu sous le nom de D-BMAP, pour *Discrete Batch Markov Arrival Process*

4.6 Dimensionnement des buffers pour le régime élastique

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* 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 caractéristiques 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 *bottlenecked* 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 *non-bottlenecked* mais de débit limité. 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 4.15 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 ainsi 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. Par exemple pour $p = 2$ Mb/s dans la figure 4.15, le débit $\phi(i)$ chute lorsqu'il y a plus de 11 flots, et ne réaugmente 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 est faible, plus le lien opérera en régime transparent jusqu'à de fortes charges.

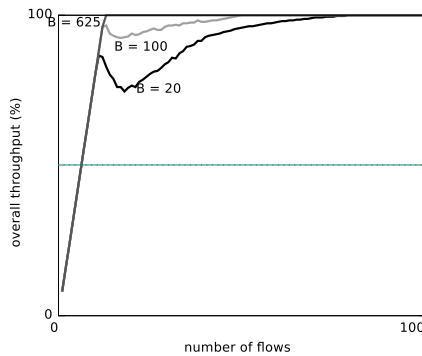


FIGURE 4.14: Overall throughput $\phi(i)$ with i flows as fraction of residual bandwidth and expected flow throughput γ as function of load ρ , for peak rate limited bottlenecked flows (2Mb/s)

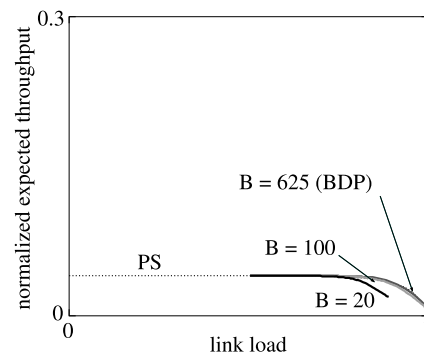


FIGURE 4.15: Taux d'utilisation de la capacité résiduelle $\phi(i)$ atteint pour chaque nombre i de flots en cours ; et espérance du débit γ d'un flot en fonction de la charge ρ , pour des flots *non-bottlenecked* de débit limité à 2Mb/s

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 exogène 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 fort débit 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

4.7 Conclusions

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 [31]. 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 étant lorsqu'un flot peut saurer à lui seul la capacité résiduelle.

Toutefois cette étude s'intéresse uniquement à la performance de l'aggrégat 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 d'étudier dans le prochain chapitre.

Chapitre 5

Partage de bande passante étendu : nouveaux protocoles TCP, fair queueing

Contents

5.1	Introduction	66
5.2	TODO	66
5.3	Introduction de nouveaux protocoles TCP	68
5.4	Enhanced bandwidth sharing	70
5.5	Conclusions	72

Contributions :

- ...

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*.

5.1 Introduction

5.2 TODO

Themes de ce chapitre : Fairness, nouvelles propositions TCP, fair queueing

Les études précédentes avec gamma supposent que les flots partagent équitablement la bande passante. Ca peut etre forcé par le fair queueing. On pourrait alors étudier comment les algorithmes se comportent dans un environnement fair.

On peut d'abord voir ce qui se passe dans un environnement non fair.

Comportement de ces nouveaux protocoles avec des petits buffers. Est-ce que le FQ aide?

Peut on faire le y sans fair queueing, pour montrer 2 classes unfair?

Misc :

AIMD / MIMD synchronization abolir slow start pacing

Chiu Jain phase plot $A+B = C$ == efficient over its overload convergence to equilibrium

8.4. RTT and Fairness

TCP, and congestion controls such as SCTP [RFC2960] that inherit from it, converge on a rate that is inversely proportional to round trip time (RTT). This is due to TCP's original design goal of avoiding adding more than one segment to the data in flight each RTT.

Congestion controls certainly have to take RTT delay in the feedback loop into account to ensure stability. Nonetheless, It is perfectly possible to design a robust congestion control that responds more slowly to changes on longer paths, but still converges to the same rate as it would with a shorter RTT. FAST TCP [FAST] is an example of such a congestion control. Siris's weighted window-based congestion controller [WindowPropFair] also has dynamics that are sensitive to RTT, while converging on a bit-rate that is independent of RTT.

RTT is not in itself a factor that affects fairness. In fact, once a sender is accountable for the congestion it causes, it will be in its own interests to be more cautious on longer RTT paths, as it has proportionately more data in flight so it risks causing more congestion before it can react.

Broadly the extra risk of causing congestion with larger RTTs is usually sufficient to encourage behaviour that leads to stability. However, this gross generalisation needs to be couched in assumptions and constraints that are beyond the scope of this memo (and beyond my ability to keep up with the literature).

Flow startup Alternatives to slow start = big challenge!

5.2.1 Paced TCP

TODO : A mettre dans le prochain chapitre, il faudra aussi un peu annoncer les résultats que l'on obtiens vu que leur établissement suit la meme démarche

[31] propose 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. Les figures 5.1 illustrent l'évolution de *cwnd* pour un seul flot *bottlenecked* avec¹ 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, $B = 100$.

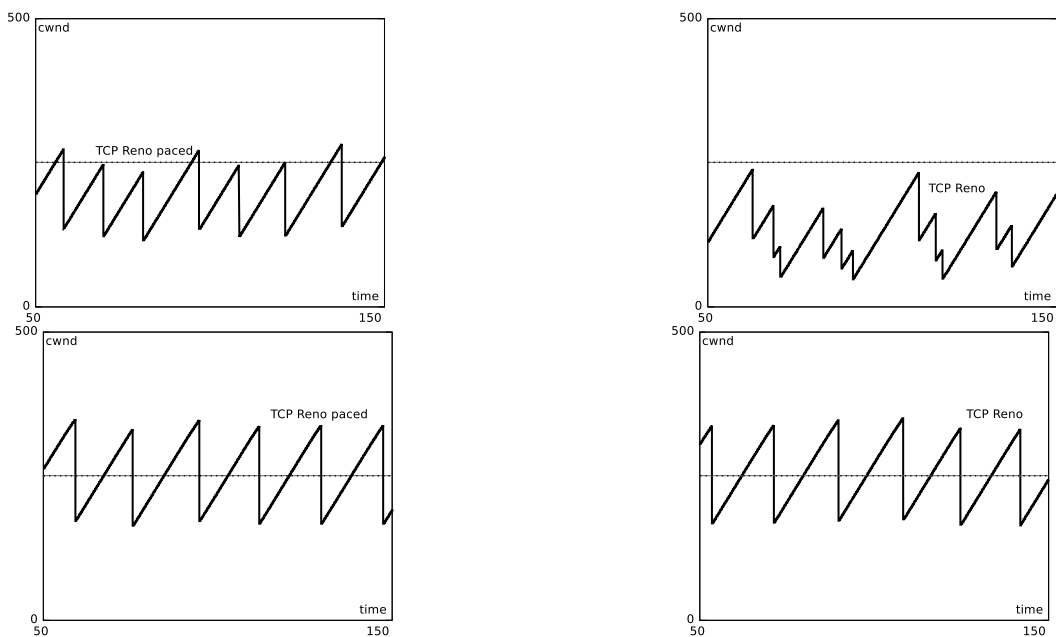


FIGURE 5.1: Evolution of *cwnd* for one paced Reno flow (left) and one Reno flow (right), with $B = 20$ packets (top) and $B = 100$ packets (bottom), for $C = 50$ Mbps, $\rho_b = 0.5$

pacing <http://www.cs.washington.edu/homes/tom/pubs/pacing.html>

Impossibilité de connaître le vrai BDP... d'où l'intérêt de la différenciation.

Une alternative plus radicale aux solutions FIFO ou avec AQM est d'introduction de *fair queueing* entre les flots. L'un des avantages bien connus du *fair queueing* est qu'il est possible 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* possède l'avantage supplémentaire d'assurer de faibles délais pour les flots streaming possédant un débit relativement bas.

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, comme le suggère [?]. Nous prétendons dans la suite de ce chapitre que le nombre de flots à ordonnancer est en fait

1. 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*.

relativement faible et indépendant de la capacité du lien, comme démontré dans [53, 52].

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 [48]. Il reste à étudier les besoins en buffer de tels protocoles.

Nouveaux protocoles TCP et fairness... dur...

Toutefois il apparaît nécessaire de diminuer la taille des buffers pour les raisons que nous avons évoquées. Cette démarche peut aller de pair avec d'autres avancées : d'une part, remédier aux inefficacités de TCP Reno pour les transferts à haut débit, et d'autre part ne plus dimensionner les buffers en fonction d'un protocole, mais plutôt adapter le protocole à utiliser aux mieux les ressources offertes par le réseau.

Des études visant à concevoir des versions plus évoluées ont été proposées dans la littérature. On en comprend l'intérêt avec l'émergence de nouvelles applications du réseau nécessitant de gros transferts de données telles que les grilles.

TODO : Evoquer déjà le *fair queueing* + présenter notre double objectif : quel buffer est idéal, quelle performance pour des petits buffers de l'ordre de 20 paquets, même si on en a déjà un peu parlé + ajouter quelques commentaires sur la QoS des flots non bnk.

5.3 Introduction de nouveaux protocoles TCP

5.3.1 Faiblesses de TCP Reno pour les transferts à haut débit

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

Le premier problème vient notamment de l'algorithme AIMD :

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

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 pertes infimes, qui n'est pas réalisable en pratique. Ainsi, les pertes réellement subies par un flot TCP entraîneront un débit moyen de transfert bien en deçà de la limite équitable permise par le lien. Cette instabilité peut être diminuée en ne considérant plus seulement une notion binaire telle qu'une perte de paquets, mais également en y adjoignant une estimation du nombre de paquets en attente dans les files (TCP Vegas) par exemple. L'évolution de la fenêtre de congestion est modulée par l'état de congestion du réseau.

5.3.2 Impact de l'utilisation des nouveaux protocoles TCP sur la taille des buffers

Parmi les nombreuses propositions que l'on peut trouver dans la littérature, nous retiendrons ici trois d'entre elles : Scalable TCP [47], High Speed Tcp [?] [?] et Fast

5.3 Introduction de nouveaux protocoles TCP

TCP [43]. Après un bref exposé de leurs caractéristiques, nous présentons des études des performances de ces protocoles relativement à la taille des buffers.

Scalable TCP [47] présente un changement relativement simple au contrôle de congestion classique de TCP, en en modifiant les paramètres de l'algorithme AIMD afin de le rendre plus agressif.

$$w \leftarrow w + a$$

$$w \leftarrow w - b \cdot w$$

$$a = 0.01, b = 0.125$$

Le risque est que sur un lien chargé, le taux de pertes de paquets soit plus élevé, ce qui entraînera une baisse du débit efficace.

Scalable TCP étant très similaire à TCP Reno (simplement une modification des constantes de croissance et de décroissance de la fenêtre), les calculs sont similaires donnent :

$$B = \frac{b}{1-b} \cdot C \cdot RTT$$

On remarque qu'avec les valeurs de TCP Reno ($b = \frac{1}{2}$), on retrouve le *Bandwith Delay Product*. En ce qui concerne les valeurs proposées pour l'implémentation de Scalable TCP, $b = 0.125$, on obtient une valeur conseillée pour le buffer de l'ordre de 15% du *Bandwith Delay Product*. Il n'est pas surprenant de trouver une valeur plus faible. La réduction de fenêtre après une perte est très petite comparativement à TCP Reno, une taille de buffer très réduite suffit donc à maintenir le débit de la connexion.

High Speed TCP [?] [?] propose de changer le comportement du contrôle de congestion au delà d'un seuil de taille de fenêtre. L'analyse de High Speed TCP est plus complexe puisqu'elle prévoit une évolution de la taille de la fenêtre de congestion relativement à sa valeur courante.

$$w \leftarrow w + a(w)$$

$$w \leftarrow w - b(w) \cdot w$$

Une étude de l'impact sur la taille du buffer est proposée par [?]. Il 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, le débit observé est bas.

Fast TCP [43] à une approche similaire à TCP Vegas. Il s'appuie en plus sur le délai des paquets dans la file d'attente, qui est une mesure plus extensible (scalable), plus facile à estimer et plus fiable que le taux de pertes. Fast TCP met à jour périodiquement sa fenêtre de congestion par un calcul unique :

$$w \leftarrow \min \left\{ 2w, (1 - \gamma)w + \gamma \left(\frac{baseRTT}{RTT} w + \alpha(w, qdelay) \right) \right\}$$

où $\gamma \in (0, 1]$, $baseRTT$ est le RTT minimum observé, et α pour simplifier est pris égal à une constante.

L'algorithme assume que le RTT minimum observé est le délai de propagation de bout en bout. L'algorithme utilise les mesures du RTT afin d'estimer le délai dans les files d'attente des paquets. Cette hypothèse peut être remise en cause à la lueur des travaux de [?], mais reste valide dans le cadre d'un réseau backbone non surchargé. L'évaluation de Fast TCP, et une comparaison de ses performances avec les autres protocoles est effectuée dans [?]. L'ensemble des protocoles subissent des performances dégradées pour des tailles de buffer trop petites. Chaque flot maintenant

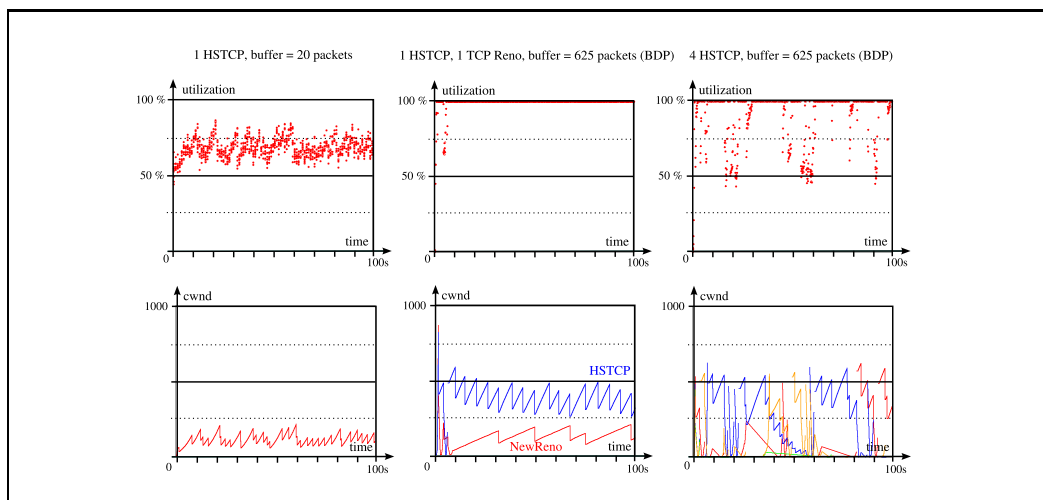


FIGURE 5.2: From left to right, utilization and cwnd size as a function of time for : - 1 HSTCP flow with a 20 buffer packet, - 1 HSTCP flow and 1 TCP New Reno flow with a 625 packet buffer (bandwidth delay product), - 4 HSTCP flows with a 625 packet buffer. The scheduling discipline is FIFO.

α paquet dans la file, le buffer doit être de taille suffisante $n \cdot \alpha$ afin que n flots atteignent une position d'équilibre et se stabilisent. Il convient ainsi d'ajuster ce paramètre dynamiquement en fonction du nombre de flots. Toutefois le protocole ne nécessite pas une taille de buffer au-delà de cette valeur : dès lors qu'un équilibre est atteint, l'occupation du buffer est quasi constante et le taux de pertes nul. La grandeur du RTT ne semble pas intervenir. Enfin, la dégradation des performances n'est pas estimée dans le cadre d'un sous dimensionnement du buffer.

Un frein à l'introduction de ces protocoles est qu'ils ne sont pas *TCP friendly*, c'est-à-dire équitables lorsqu'ils sont mis en compétition avec des connexions TCP Reno standard. Ce problème disparaît si l'on considère un réseau implémentant des algorithmes de *fair queueing*. Il est alors possible de mesurer l'impact d'une politique de gestion de la file d'attente couplées à un ordonnancement de type *fair queueing* (comme RED, CHOCe ou encore AFD).

5.4 Enhanced bandwidth sharing

In considering buffer requirements in the future high speed network, it is appropriate to consider other evolutions that can affect the performance of congestion control. In this section we consider the impact of new TCP versions and the possible use of per-flow fair scheduling.

5.4.1 New high speed protocols

We confine the present evaluation to the use of HSTCP as defined in RFC 3649 [33]. The performance of other protocols will be considered in future work.

Figure 5.2 shows results for three configurations where bottlenecked connections use HSTCP instead of New Reno. The left hand plots demonstrate that a single HSTCP connection uses bandwidth more efficiently than Reno when the buffer size is considerably less than the bandwidth delay product. This is an encouraging result illustrating that the evolution of TCP can improve the efficiency of small buffers.

5.4 Enhanced bandwidth sharing

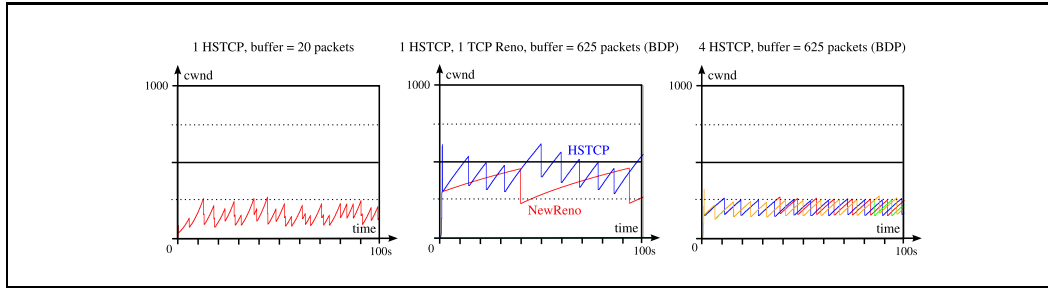


FIGURE 5.3: From left to right, utilization and cwnd size as a function of time for : - 1 TCP New Reno flow with a 20 buffer packet, - 1 HSTCP flow and 1 TCP New Reno flow with a 625 packet buffer (bandwidth delay product), - 4 HSTCP flows with a 625 packet buffer. The scheduling discipline is Fair Queueing.

In the middle, an HSTCP flow shares the 25Mbps residual bandwidth with a Reno flow using a large buffer. The depicted behaviour of the respective flow windows *cwnd* is typical : the Reno connection is unable to compete fairly with the more aggressive HSTCP connection.

The last plots on the right relate to 4 bottlenecked HSTCP flows. We observe quite chaotic behaviour of the flow *cwnds* as one flow after the other gains a temporary ascendancy. As soon as one flow has a large value of *cwnd*, it tends to act more aggressively and preserves its relative advantage until it is eventually superseded by one of the other flows. This kind of behaviour has previously been observed by Li et al. [58].

5.4.2 Using fair queuing

We have implemented per-flow fair queuing in ns2 with a drop from longest queue discard policy. The latter has been shown to largely outperform RED as an active queue management mechanism [84]. In the present case the drop policy is such that non-bottlenecked flows remain loss free. Their packets also have very low latency², a highly desirable feature for any streaming flows included in the background traffic.

Figure 5.3 illustrates the evolution of utilization and *cwnd* for the three cases previously illustrated under FIFO in Figure 5.2. The plot on the left corresponds to a single TCP Reno connection sharing the link with background traffic using a 20 packet buffer. Comparison with the left hand plot of Figure 5.2 shows that fair queuing barely alleviates the loss in performance due to a very small buffer (it does, however, avoid loss for the non-bottlenecked flows).

The middle plot should be compared to the middle plot of Figure 5.2. It is clear that fair queuing considerably reduces the bias in favour of HSTCP. The HSTCP flow does, however, gain slightly more throughput than the Reno flow due to its more aggressive behaviour.

Finally the rightmost plot confirms that fair queuing effectively counters the chaotic behaviour illustrated in the corresponding plot of Figure 5.2. Fair queuing also considerably attenuates the impact on existing flows of newly arriving flows whose aggressive slow-start behaviour can otherwise lead to quite severe loss events.

Two supplementary benefits of fair queuing are desynchronized loss events due to flow isolation and the fact that packet emissions are paced at the current fair rate. The latter effect means acknowledgements are paced leading to less bursty input in the following window cycle. Both phenomena appear to improve bandwidth sharing performance somewhat, notably when the buffer size is small.

2. This can be reduced still further using the priority mechanisms described in [? 55].

5.5 Conclusions

Several authors have recently pointed to the need to reduce Internet buffer sizes by several orders of magnitude. This is necessary for technological reasons but is also claimed to have a positive impact on performance. In the present paper we have contested the relevance of the traffic model assumed in the published evaluations where the link is shared by a very large number of permanent bottlenecked TCP flows.

In fact, it is necessary to account for the random nature of traffic at flow level. At normal link loads, the large number of flows that are observed to share link bandwidth are necessarily limited in rate by constraints that are exogenous to the considered link. The number of bottlenecked flows, whose rate is determined by the considered link through the action of the congestion control algorithms, is typically very small (i.e., 0, 1 or 2, ... flows) with high probability.

We have conducted a preliminary performance evaluation by means of simulation. The results show that the previously recommended reduction in buffer capacity to 20 packets is rather too drastic. For the considered configuration, a buffer of 100 packets seems a more appropriate choice allowing reasonable throughput while maintaining low packet latency. The buffer must be sized to accommodate near coincident packet arrivals from independent non-bottlenecked flows (this requirement can be evaluated using a simple queuing model) while preserving sufficient free space to allow a single flow to efficiently utilize the residual link capacity. For a link of arbitrary capacity, a buffer size of one or two hundred packets appears as a reasonable choice. We are currently developing a mathematical model in order to more precisely determine required buffer size as a function of bottlenecked and non-bottlenecked flow loads.

Use of new high speed TCP versions can significantly reduce the required buffer size. A smaller buffer is sufficient to maintain high throughput when the window additive increase rate is higher and the multiplicative decrease rate is smaller than legacy TCP, as in the proposed HSTCP congestion avoidance algorithm. It is clear, however, that the fairness and convergence properties of these new protocols are less than perfect.

Bandwidth sharing would be further enhanced by the implementation of per-flow fair queuing in router queues. Fair queuing enables the coexistence of legacy and high speed TCP versions by avoiding the above mentioned fairness and convergence problems. Fair queuing also ensures low packet latency for non-bottlenecked flows, including any streaming flows within the traffic mix. Finally, we recall that under the realistic traffic conditions discussed here, fair queuing is both scalable and feasible since the number of flows with packets to be scheduled at any instant is only measured in hundreds for any link capacity.

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.

Contents

6.1	Introduction	75
6.2	Etat de l'art des approches de MBAC	75
6.3	Implémentation et évaluation de l'algorithme de Grossglauser et Tse	80
6.4	Un algorithme de MBAC pour Cross-Protect	82
6.5	Evaluation des algorithmes	86
6.6	Conclusions	105
6.7	Additional results	106

Contributions :

-

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 a 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*.

6.1 Introduction

L'amélioration du contrôle d'admission au sein de Cross-Protect est soumise 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*).

Le contrôle d'admission réalisé par Cross-Protect obéit à un double objectif. Le premier est de maintenir le débit équitable suffisamment élevé afin de préserver l'efficacité du trafic élastique, et ainsi assurer que les flots *streaming* se retrouvent toujours dans la catégorie "under". Le second est de préserver la qualité de service de ces flots *streaming*, en leur assurant de faibles délais et un taux de perte quasi inexistant.

Quand une part 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 (typiquement égal à 1% de la capacité du lien [?]). 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 [? 55]. 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 la majorité du trafic.

Nous supposons que les flots *streaming* que nous souhaitons protéger ont un débit inférieur à p , typiquement inférieur au seuil imposé au *fair rate*. Prenons l'exemple de flots vidéos de 4Mb/s sur un lien OC48 de cœur de réseau à 2.5Gb/s. Dans une telle situation, la capacité du lien C sera très largement supérieure à p ($C > 100p$ en considérant généralement le seuil sur le *fair rate* de l'ordre de 1% de C), ce qui crée un contexte favorable à un multiplexage statistique efficace. Une telle modification permet une meilleure utilisation des ressources et un taux de blocage des flots plus faible, puisque la charge prioritaire est délestée de flots qui ne nécessitent aucune protection et qui peuvent se contenter d'un traitement par l'ordonnanceur *fair queueing*.

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 . Nous proposons un algorithme simple et illustrons ses performances aux travers de simulations dans des situations de surcharge ou de *flash crowds*.

6.2 Etat de l'art des approches de MBAC

Seule une faible quantité d'informations au sujet du trafic est disponible dans l'architecture FAN, typiquement des mesures agrégées du trafic entrant, ainsi qu'une spécification du débit crête maximum des flots à protéger. Si les propositions de MBAC de type multiplexage sans buffer sont nombreuses dans la littérature, aucune n'est cependant applicable à Cross-Protect. Nous résumons ici les propositions qui s'en approchent le plus.

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 per-

formance perçue au niveau flot. Ceci est illustré dans [?] et [?] où la performance est mesurée en terme de taux de perte de paquets. Les algorithmes diffèrent au niveau de la prédictibilité 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. [?] 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 [?]. Nous espérons seulement aboutir à des algorithmes raisonnablement efficaces et c'est là que se situe la limite de notre ambition.

6.2.1 Measured sum

Jamin et al. [?] 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 \hat{v} est inférieure à un seuil d'utilisation u de la capacité du lien C , comme illustré par l'équation 6.1.

$$\hat{v} + r \leq uC \quad (6.1)$$

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 Kurose [?] proposent une version de l'algorithme basée sur un ajustement adaptatif de la longueur de cette fenêtre. \hat{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 :

$$\hat{v} = \begin{cases} \text{MAX}(\hat{v}^S), & \text{of past } T \text{ measurement window} \\ \hat{v}^S, & \text{if } \hat{v}^S > \hat{v}, \text{ where } \hat{v}^S \text{ is the} \\ & \text{measured rate over a period } S \\ \hat{v} + r & \text{when addmiting a new flow} \end{cases}$$

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 prédictible (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

L'inégalité de Hoeffding est un résultat de la théorie des probabilités qui donne une borne supérieure à la probabilité d'une variable aléatoire de dévier de la valeur de son espérance.

Le contrôle d'admission introduit par S. Floyd [?] calcule la bande passante équivalente d'un ensemble de flots (une estimation de la bande passante agrégée) en utilisant la borne de Hoeffding. Cette bande passante équivalente dépend du choix d'une probabilité de perte cible et s'exprime ainsi :

$$\hat{C}_H = \hat{\mu} + \sqrt{\frac{\ln 1/\epsilon \sum_{i=1}^n (p_i)^2}{2}}$$

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

6.2 Etat de l'art des approches de MBAC

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 conservative de l'algorithme, c'est-à-dire une utilisation faible. La méthode suppose également la connaissance du débit crête individuel de chaque flots, 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 [?] utilise des mesures des enveloppes d'agrégat de trafic maximales. 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 :

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

où \bar{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 $Q(\cdot)$ la fonction de distribution complémentaire d'une variable aléatoire Gaussienne $N(0,1)$.

Un niveau de confiance est également calculer 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. Et le processus de mesure est relativement complexe à cause des nombreuses échelles de temps nécessaires.

6.2.4 Approche basée sur la théorie de la décision

Gibbens et al. [?] 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 méthode proposées, la plus simple ne prend en compte que le débit crête d'un flot.

Cela correspond exactement à nos besoin 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.

6.2.5 Théorie de la bande passante équivalente

Gibbens et Kelly [?] présentent une famille d'algorithmes basées sur des tangentes à une courbe de bande passante équivalente, calculée à partir de borne de Chernoff.

Les méthodes de Chernoff consistent à borner une variable aléatoire X , qui représente une séquence de variables aléatoires en étudiant la variable aléatoire \exp^{tX} plutôt que la variable X elle-même.

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 [?]) ainsi que le nombre de flots de chaque classe accompagné de leur débit crête. Cette méthode dérive de l'utilisation de la borne de Hoeffding proposée par Floyd [?].

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

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

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

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

6.2.6 Stratégies de *back off*

Les méthodes précédentes, proposées par Gibbens *et al.* [?], Gibbens et Kelly [?] et Floyd [?] 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 [? ?] se base sur l'idée de décomposer les variations du débit agrégé selon deux échelles de temps : des fluctuations à court et à long terme. 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 départs des 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 débits intrinsèques aux flots en cours.

Une condition d'admission est dérivée 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 sa variance, sont mesurées. Un nouveau flot est admis si la condition suivante est satisfaite :

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

6.2 Etat de l'art des approches de MBAC

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

$$C - A_t^\lambda - p > \alpha_q \hat{\sigma}_t^{AH} \quad (6.3)$$

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

Pour obtenir cette formule, les variables A_t^L et A_t^H sont définies et correspondent l'application d'un filtrage 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-haut.

Une autre variable $\hat{\sigma}_t^{AH}$ est introduite et correspond à l'estimation de la variance de la composante haute fréquence 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 sont 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) due 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 instant discrets, alors A_t^λ peut être exprimé ainsi :

$$A_{t_i}^\lambda = \phi_i A_{t_{i-1}}^\lambda + (1 - \phi_i) S_{t_{i-1}} + r \cdot \mathbf{1}_{\{\text{flow admission at } t_i\}}$$

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 [??] est la plus proche de nos hypothèses sur le trafic, et au vu informations limitées que nous avons à son sujet. Elle possède notamment l'avantage, par rapport par exemple à celle de Knightly *et al.* [?], 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 considérer profiter des nombreux avantages de cette méthode.

Knightly *et al.* utilise 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 changements 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.

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

Une version discrétisée de l'algorithme de Grossglauser et Tse est présentée en annexe ??, afin d'être implémentée et évaluée. Les résultats obtenus avec cet algorithme nous serviront de référence lors de l'évaluation réalisée dans la section 6.5.

6.4 Un algorithme de MBAC pour Cross-Protect

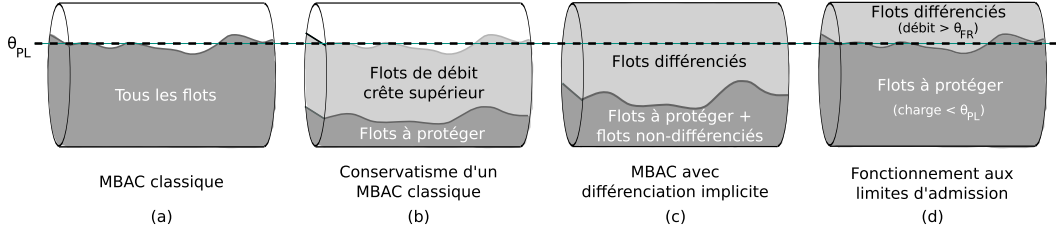


FIGURE 6.1: Illustration de l'algorithme de MBAC avec divers régimes d'utilisation

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

$$\begin{aligned} A_{n+1}^L &= \alpha A_n^L + (1 - \alpha) S_{n+1} \\ A_{n+1}^H &= S_{n+1} - A_{n+1}^L \\ D_{n+1} &= \beta D_n + (1 - \beta) (A_{n+1}^H)^2 \\ E_{n+1} &= \beta E_n + (1 - \beta) A_{n+1}^H \\ \sigma_{n+1} &= (D_{n+1} - E_{n+1}^2)^{1/2} \\ A_{n+1}^\lambda &= A_{n+1}^L \end{aligned}$$

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

$$A_{n+1}^\lambda + r < c - \alpha_q \sigma_{n+1}$$

Après une admission,

$$A_{n+1}^\lambda + = r$$

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 :

$$A_n^\lambda = A_n^L + \lambda_n$$

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

$$A_n^\lambda + = r$$

6.4 Un algorithme de MBAC pour Cross-Protect

6.4.1 Moyenne et variance du *priority load*

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

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

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

Pour appliquer l'approche de [?], nous estimons la variance de la manière décrite dans la section précédente :

$$\hat{\sigma}_t^2 = D_t - E_t^2 \text{ where} \quad (6.5)$$

$$D_t = (1 - \gamma)D_{t-1} + \gamma(b_t - B_t)^2, \quad (6.6)$$

$$E_t = (1 - \gamma)E_{t-1} + \gamma(b_t - B_t). \quad (6.7)$$

En suivant [?], le paramètre de lissage γ est choisi égal à $(1 - \beta)/10$. En fait, l'estimation de \tilde{T}_h reste problématique puisque la distribution de la durée des flots n'est pas exponentielle comme considérée dans [?] mais *heavy-tailed*. Des résultats préliminaires semblent heureusement montrer que le choix des paramètres β et γ n'est pas extrêmement critique.

L'utilisation de la condition d'admission (??) sur ces estimateurs est trop conservatrice pour les flots de débit crête supérieur à p . Nous supposons que de tels flots sont capables (devraient) ajuster leur débit au *fair rate* en cas de surcharge. 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 empêche 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 où le contrôle d'admission différencie idéalement les flots de débit crête inférieurs et supérieurs à p .

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.

6.4.2 Approximation basée sur une hypothèse Poissonnienne

Afin de remédier à ce problème, nous tentons ici de définir un seuil sur le *priority load* qui dépendrait uniquement de sa moyenne. Supposons que nous sommes dans le cas idéal décrit précédemment, et que l'ensemble des flots non-différenciés à un débit crête inférieur ou égal à p , et qu'il possède à tout instant (sur chaque slot) une distribution de Poisson.

C'est le cas en l'absence de blocage pour le modèle de sessions Poisson décrit dans le chapitre précédent dans la section ??). Si les paquets ont une taille constante maximale L , alors la distribution du nombre de paquets arrivant dans un intervalle de temps inférieur à L/p possèdent également une distribution de Poisson. Un flot de débit r contribue de manière indépendante 1 paquet avec la probabilité r/p .

Cette hypothèse est clairement un pire cas puisque nous ignorons à la fois les flots de débit crête inférieur à p , et l'impact du lissage effectué par le contrôle d'admission sur le processus des débits.

6.4 Un algorithme de MBAC pour Cross-Protect

Nous pouvons ainsi estimer la variance du débit sur chaque slot à partir de la moyenne. Choisissons un intervalle de longueur $\tau = L/p$. Étant donnée la mesure du *priority load* B_t bits/s, soit $m_t = A_t\tau/L$ une estimation du nombre de paquets, et nous déduisons l'estimation de la variance $\hat{\sigma}_t^2 = m_t L^2 / \tau^2 = B_t p$. Cette estimation peut alors être appliquée à la condition d'admission (??). Nous dénommons l'algorithme ainsi créé MBAC (Poisson).

6.4.3 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 toutefois 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	0.01	0.001	0.01	0.001
α_q	2.33	3.09	2.33	3.09
Threshold	0.79	0.73	0.93	0.91

TABLE 6.1: Admission thresholds

6.4.4 Intégration de la variance pour améliorer l'algorithme

D'autre part, si l'agrégat de trafic possède de nombreux flots de débit inférieur à p , ce qui sera le cas en pratique, alors l'approximation Poisson peut être trop conservatrice. C'est pourquoi nous proposons une extension au MBAC où l'estimation de la variance est le minimum de $B_t p$ et de celle calculée par (6.6). Les algorithmes (Poisson) et Minimum de Variance (MinVar) sont évalués pour différents scénarios de test dans la section 6.5 ci-dessous.

6.4.5 Cas des flots de débit crête supérieur

Si l'on considère maintenant des flots de débit crête supérieur à p , alors la variance estimée par (6.6) sera supérieure à celle de l'estimation Poisson $B_t p$, donc bornée à la différence du MBAC dérivé de [?], et nous nous attendons à ce que l'estimation Poisson permette plus facilement la transition vers un état tel que Fig. 6.1c).

Le seul cas apparemment problématique pour la différenciation des flots est quand 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 $p = 100\text{kb/s}$, sur un lien de capacité $C = 10\text{Mb/s}$. Nous normalisons ces valeurs afin d'obtenir une charge 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 $\sigma = \sqrt{\theta p}$.

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 $P_2 = 200\text{kb/s}$. Chacune contribue à la moitié de la charge générée.

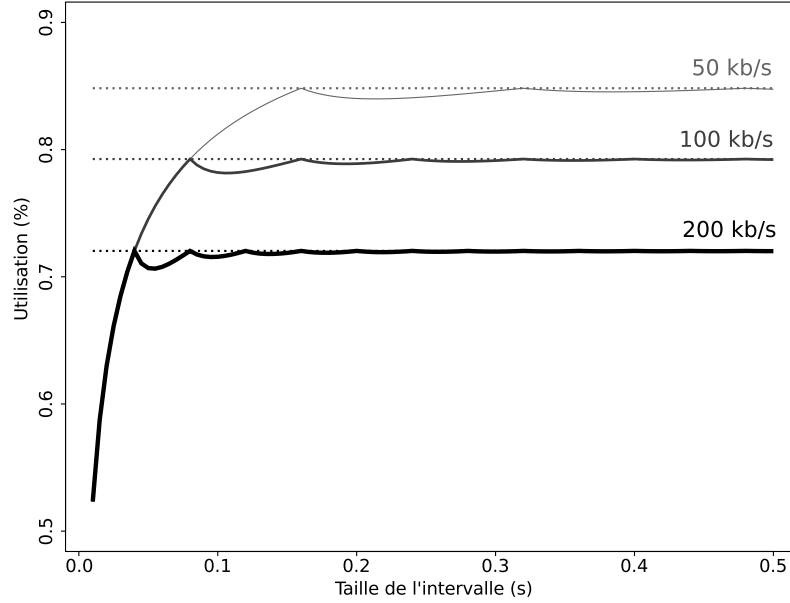


FIGURE 6.2: Valeur du *priority load* mesuré en fonction de la taille de l'intervalle, pour différentes valeurs de p . La valeur réelle est indiquée par les lignes en pointillés.

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.

6.4.6 Influence de la taille du slot sur les mesures

Pour de grandes valeurs de p , la taille de l'intervalle de mesure peut devenir très faible et poser un problème d'implémentation si par exemple le système présente une certaine granularité temporelle¹. Nous relaxons ainsi l'hypothèse sur la taille du slot τ .

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 l'annexe ?? donnent :

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

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

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 200 kb/s.

Nous introduisons le paramètre k tel que $\tau = kL/p$. Pour $k = 1$, la variance est celle du cas Poisson, qui permet d'estimer convenablement la variance et donc d'obtenir une utilisation optimal. 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.

1. C'est le cas notamment du noyau Linux qui selon les versions présente une résolution minimale par défaut de 1 ou 10ms selon les versions. Nous discutons plus en détail de cet aspect dans l'annexe ??.

6.4 Un algorithme de MBAC pour Cross-Protect

Il est donc important de choisir convenablement la valeur de k afin que la mesure de la variance soit convenable, et donc que l'algorithme (MinVar) soit le plus efficace possible. Par exemple, pour $p = 100\text{kb/s}$, une valeur de $k = 4$ permettra de "détecter" convenablement des flots de débits $r \geq 25\text{kb/s}$, et donc d'obtenir le cas échéant la performance correspondante. En ajustant successivement le seuil d'admission à une valeur plus optimale, on s'attend à ce que le MBAC (MinVar) assure une meilleure différenciation des flots.

A noter qu'une trop grande valeur de k peut introduire un manque de réactivité de l'algorithme, notamment dans les régimes de surcharge non stationnaires, tout en n'apportant qu'une amélioration mineure de la performance. Une valeur de quelques unités semble être un bon compromis. Nous évaluerons l'impact de k dans les sections 6.5 et 6.5.8.

6.4.7 Instabilité du *priority load* mesuré

Le *priority load* peut varier abruptement lorsque l'ensemble des flots de 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, ne permettant plus la protection des flots avec B_t inférieur au seuil critique et une mesure lissée du *fair rate* suffisamment grande permettant de nouvelles admissions. Ce phénomène est d'autant plus marqué que le nombre de classes de flots sera faible (comme dans nos simulations dans la section 6.5) puisqu'un grand nombre de flots seront rapidement sujets 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 où un flot de débit crête p pourrait être *backlogged*. Cette dernière condition a été ajoutée à l'algorithme initial présenté au chapitre 3, section ?? (notion de *fair rate* instantané).

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 se produit ce basculement justement. Le contrôle exercé sur le *priority load* permet justement de limiter les instants où cela se produit, et demeure essentiel car l'indicateur sur le *fair rate* instantané seul ne serait pas stable (indication binaire).

Cette modification se montre également utile lorsque les flots qui ont un débit crête nominal mais qui sont en pratique soumis à une gigue deviennent momentanément *backlogged* quand l'intervalle inter-arrivées des paquets devient trop faible (voir la section 6.5.7).

6.4.8 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. Cela se produit en particulier lorsque une panne quelquepart dans le réseau 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 [?]. Étant donnés les estimateurs A_t et $\hat{\sigma}_t^2$, nous acceptons un maximum de n nouveaux flots où $(n+1)p + A_t + \alpha_q \hat{\sigma}_t^2 > C$.

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 *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 [?] 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.

6.5 Evaluation des algorithmes

Nous évaluons les MBAC proposés au travers de simulations ns-2 dans divers scénarios de trafic.

6.5.1 Modèle de trafic

Dans la suite de ce chapitre, nous considérons que le trafic est composé de flots *on-off* supposés arriver selon un processus de Poisson. Un flot émet des paquets de taille constante L dans chaque période *on* à son débit crête P . L'adoption de ce modèle nous permet de simplifier l'évaluation de la capacité de l'algorithme de MBAC à distinguer les flots de débit supérieur à un débit cible p donné.

Supposer que les paquets sont de taille constante est une hypothèse de pire cas si la taille L correspond au MTU². De plus petits paquets avec un débit équivalent produisent en effet un processus d'arrivée plus lisse et de plus faibles délais et taux de pertes.

Le débit moyen est la valeur mesurée A_t .

6.5.2 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 trafic est composé de flots UDP dont la durée est exponentiellement distribuée de moyenne $T_h = 60\text{s}$. Ils arrivent selon un processus de Poisson et génèrent des paquets selon un processus *on-off*, de débit crête constant pendant la période *on* (50, 100 ou 300 kb/s dans nos simulations). Nous qualifierons de classe l'ensemble des flots de même débit crête; la classe prioritaire (resp. non-prioritaire) sera l'ensemble des classes de flots de débit crête inférieur ou égal (resp. supérieur) à p . La durée des période *on* et *off* est distribuée exponentiellement de moyenne 500ms, sauf indication contraire. La taille des paquets est constante fixée à 1000 octets. La durée des simulations est de 2000s, dont les 200 premières sont ignorées pour le calcul des statistiques.

La capacité de 10Mb/s peut sembler dérisoire au vu des débits réels des liens de cœur de réseau, mais elle n'influe pas les résultats de performance que nous observerons puisqu'ils dépendent uniquement du rapport C/p . Ce choix nous permet d'effectuer le grand nombre de simulations que nous présentons plus rapidement.

Nous considérons ici un débit p de l'ordre du seuil sur le *fair rate*, généralement choisi égal à 1% de C , qui correspond à un pire cas (voir tableau 6.1). Sur des liens réels à très

2. Maximum Transmission Unit

6.5 Evaluation des algorithmes

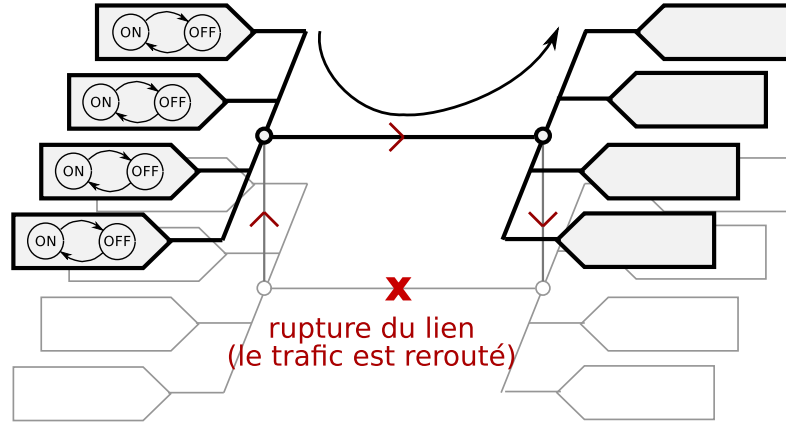


FIGURE 6.3: Topologie utilisée pour la simulation : la partie représentée en trait gras est utilisée pour les simulations en régime stationnaire ; l'ensemble de la topologie sera utilisée pour simuler un scénario de *flash crowd* dans la section 6.5.8

fort débit, les débits des flots que nous souhaitons protéger ne seront pas nécessairement plus grands, c'est pourquoi le ratio C/p sera généralement plus élevé. Le multiplexage statistique réalisé sera plus efficace et permettra une meilleure performance.

Les valeurs de α et β sont fixées selon les recommandations de [?]. La probabilité de débordement cible ϵ est de .01. Nous générons une charge stationnaire le plus souvent égale à 120% de la capacité du lien. L'intervalle d'échantillonnage est $\tau = kL/p$ avec $k = 1$ ou un faible entier (spécifié). Enfin, sauf mention contraire, la composition du trafic est telle que la limite par intervalle de temps n'opère pas.

6.5.3 Critères de performance

La performance du MBAC est mesurée par les critères suivants (exprimés en pourcentages (%)) dans le reste du chapitre) :

- la probabilité de débordement (ov), calculée comme le ratio d'intervalles pour lesquels la charge prioritaire sature le lien, sur le nombre total d'intervalles considérés,

$$P(b_t > C) = \frac{1}{N} \sum_{t=1}^N \mathbf{1}_{b_t > C} < \epsilon$$

où b_t représente le *priority load* instantané dans l'intervalle t et N est le nombre d'intervalles considérés. Cet indicateur reflète le bon maintien du régime transparent pour les flots protégés ;

- le taux d'utilisation du lien (ut), le ratio entre la charge écoulee et la capacité du lien, sur toute la longueur de la simulation. Il n'est vraiment représentatif de la performance réalisée qu'en régime transparent, puisqu'il est fortement dépendant de la composition du trafic sinon ;
- la probabilité de blocage (bl), qui est la proportion de flots qui sont rejetés par le contrôle d'admission. Elle est la même pour tous les flots quel que soit leur débit, et représente efficacement la qualité de service ressentie par un utilisateur ;
- la probabilité de *backlog* (bk), mesurée comme le taux de paquets prioritaires qui ne sont pas transmis via la file prioritaire, mais mis en attente temporairement dans le buffer ;
- le taux de pertes de paquets (lo), dûs aux débordements du buffer.

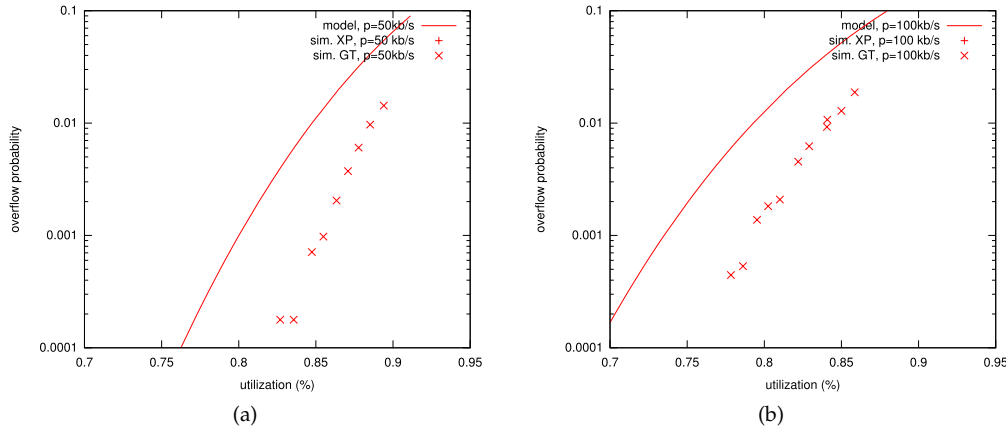


FIGURE 6.4: Utilisation en fonction de la probabilité de débordement pour les MBAC XP et GT, comparés un cas où la charge par intervalle suit une distribution de Poisson, avec $p = 50\text{kb/s}$ (a) et $p = 100\text{kb/s}$ (b)

Nous fournissons également deux indicateurs, bl_{PL} et bl_{FR} , qui donnent la proportion de flots qui ne sont pas admis, en raison des conditions sur FR et PL. On a $bl = bl_{PL} + bl_{FR}$.

6.5.4 Utilisation et probabilité de débordement

La relation entre l'utilisation et la probabilité de débordement est dérivée pour les MBAC considérés en faisant varier le critère d'admission. Pour l'algorithme dérivé de [?], dénoté GT, nous faisons varier ϵ et notons l'utilisation et le taux de débordement observés. Pour l'algorithme de Cross-Protect, dénoté XP, nous faisons simplement varier le seuil d'admission sur la charge réalisée A_i . Les figures 6.4(a) 6.4(b) présentent les résultats obtenus pour des flots de débit crête $r = 50$ et $r = 100\text{kb/s}$, comparés à un modèle idéal Poisson.

Sans surprise au vu des résultats présentés dans [?], la relation est globalement la même pour les deux MBACs. La différence apparaît lorsque le seuil est supérieur au maximum acceptable, certains flots se retrouvent *backlogged* et leurs paquets ne contribuent plus au calcul du *priority load* qui décroît, ouvrant potentiellement la voie à de nouvelles admissions. Avec un seuil d'admission trop élevé, dégradant la performance des flots *streaming*, la performance n'est plus mesurée de manière satisfaisante par la probabilité de débordement. L'algorithme de contrôle d'admission a pour objectif de déterminer ce seuil le mieux possible, tandis que l'impact de tels débordements est limité par la modification que nous y avons apporté dans la section 6.4.7.

A taux de débordement donné, l'utilisation atteinte empiriquement, qui est plus élevée que dans le cas Poisson, révèle une variance du trafic admis plus faible que Poisson.

Les figures 6.5(a) (resp. 6.5(b)) présentent l'histogramme empirique du nombre de paquets qui arrivent dans un intervalle pour les flots de débit crête 50kb/s (resp. 100kb/s), sous le MBAC XP Poisson. L'ajustement à une distribution de Poisson donne une nouvelle courbe proche de la mesure empirique, suggérant que notre approximation, certes conservative, reste très raisonnable.

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 de paquets traités en *backlog* négligeables.

6.5 Evaluation des algorithmes

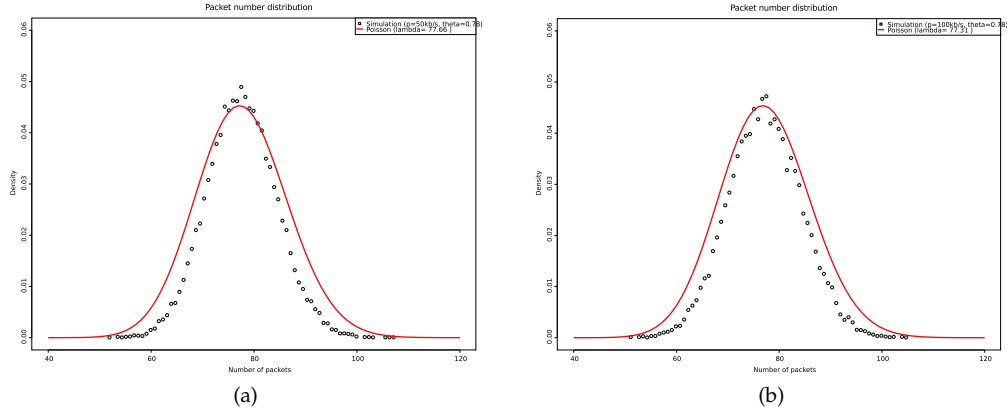


FIGURE 6.5: Densité de charge reçue par intervalle pour $\rho = 1.20$, soumise au contrôle d'admission, comparé à une distribution de Poisson ajustée, pour $p = 50\text{kb/s}$ (a) et $p = 100\text{kb/s}$ (b)

6.5.5 Prédicibilité du MBAC XP

La différence majeure qu'il peut exister entre les algorithmes est leur capacité à prédire la performance obtenue, pour un jeu de paramètres le plus réduit possible. Nous évaluons les deux MBAC XP précédents, Poisson et MinVar pour des flots de débit crête homogènes. Le débit protégé p est inférieur, égal ou supérieur au débit crête des flots r . Les résultats sont présentés dans le tableau 6.2 pour un intervalle $\tau = L/p$.

Flots de débit crête connu

Les résultats pour le cas $r = p$ sont présentés dans le tableau 6.2, pour $p = 50\text{Kb/s}$ et $p = 100\text{kb/s}$. 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 (on peut la déduire 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 une caractéristique commune à tous les MBAC. [?] discute de la difficulté à contrôler la qualité de service en fonction des mesures. Celle-ci n'est pas ressentie par l'utilisateur linéairement au taux de perte : un débordement ponctuel a des conséquences lourdes et nécessite une longue période sans dépassement par la suite. Le conservatisme des algorithmes peut se comprendre comme la difficulté d'opérer un contrôle à la limite.

Les résultats pour le MBAC MinVar montrent que l'utilisation de la mesure de variance améliore la performance puisque l'agrégat de trafic est en fait moins variant que Poisson, en raison du blocage notamment.

Flots de débit crête plus faible

Le tableau 6.2 présente une situation où le débit réel des flots est $r = 50\text{kb/s}$ alors que $p = 100\text{kb/s}$. Le seuil d'admission de l'algorithme MBAC Poisson est déterminé uniquement par la valeur de p , alors qu'il aurait pu être supérieur si l'on avait considéré le débit r . La valeur de p est typiquement choisie assez élevée pour protéger l'ensemble du trafic *streaming*, et implique un coût d'autant plus important sur les performances. Cependant, même dans notre cas défavorable où $p = 0.01C$, la perte reste raisonnable et peut être compensée par le choix de l'algorithme MinVar, pourvu que le paramètre k soit suffisamment élevé. Par exemple, avec $k = 2$, l'algorithme MinVar dans ce cas offre une

	p	r	ov	ut	bl	bk	lo
Poisson	50	50	2.78e-02	83.99	30.79	7.22e-04	0.00
	100	100	3.89e-02	78.52	34.13	1.04e-02	0.00
	100	50	2.61e-01	78.68	34.46	0.00	0.00
	100	300	1.11e-02	99.73	0.00	8.24e+01	14.30
MinVar	50	50	5.56e-01	88.02	26.98	5.96e-02	0.00
	100	100	3.06e-01	83.64	31.30	1.94e-01	0.00
	100	50	7.44e-01	81.16	32.93	7.46e-04	0.00
	100	300	1.11e-02	99.73	0.00	8.24e+01	14.30

(a) k=1

	p	r	ov	ut	bl	bk	lo
Poisson	50	50	0.00	83.99	30.40	7.22e-04	0.00
	100	100	0.00	78.45	34.46	6.25e-03	0.00
	100	50	0.00	78.66	34.55	0.00	0.00
	100	300	0.00	99.73	0.00	8.24e+01	14.30
MinVar	50	50	2.11e-01	90.18	25.12	1.80e-01	0.00
	100	100	1.61e-01	85.02	30.21	3.54e-01	0.00
	100	50	4.83e-01	87.90	27.29	5.38e-02	0.00
	100	300	0.00	99.71	0.00	8.23e+01	14.31

(b) k=2

	p	r	ov	ut	bl	bk	lo
Poisson	50	50	0.00	83.89	29.78	1.45e-03	0.00
	100	100	0.00	78.57	34.81	9.39e-03	0.00
	100	50	0.00	78.67	34.41	0.00	0.00
	100	300	0.00	99.73	0.00	8.24e+01	14.30
MinVar	50	50	1.00e-01	91.46	24.17	4.38e-01	0.00
	100	100	5.00e-02	85.95	29.00	5.39e-01	0.00
	100	50	4.83e-01	88.33	26.53	5.89e-02	0.00
	100	300	0.00	99.72	0.00	8.25e+01	14.34

(c) k=3

TABLE 6.2: Performance du MBAC Cross-Protect avec des flots de même débit crête

6.5 Evaluation des algorithmes

performance similaire au MBAC GT, ce qui confirme nos prévisions, et $k = 3$ ne serait nécessaire que pour mesurer correctement des flots de débits plus faible que r .

L'imprécision induite dans le contrôle par le choix d'un intervalle très grand semble être confirmée par la hausse du taux de paquets *backlogged*, même si elle reste négligeable (toujours moins de 0.6% pour $k = 3$). Le problème pourrait exister pour de très fortes valeurs de k . Cette dernière valeur doit ainsi être un compromis entre la performance souhaitée et la robustesse du contrôle, d'autant qu'à partir d'un certain seuil, le gain devient minime.

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* (début 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 $C/100$, d'où l'absence de blocage.

6.5.6 Performance avec un mélange de débits crête

Considérons maintenant le cas d'un mélange de deux débits crêtes différents, de part et d'autre du seuil de différenciation p . Nous évaluons la différenciation réalisée dans un ensemble de scénarios faisant varier différents paramètres, afin de mieux comprendre l'impact de chacun sur la performance du système. La plupart des résultats découlent des remarques faites pour le cas homogène.

Nous allons analyser la différenciation réalisée par les algorithmes Poisson et MinVar, de paramètre $p = 100\text{kb/s}$, avec un débit crête r_1 pour la première classe de flots (avec une charge indiquée en abscisse), et r_2 pour la seconde (charge en ordonnée). La charge totale du système est obtenue en additionnant les deux contributions, et les zones de surcharges sont représentées en niveaux de gris sur la figure. Le fond blanc caractérise les zones de charge normale où la différenciation n'opère pas comme attendu (les résultats ne sont en général pas reportés dans le tableau). Un symbole \checkmark signifie qu'il y a différenciation, \sim qu'il y a différenciation partielle (différenciation par périodes par exemple), \approx que la différenciation est difficile (oscillation entre un régime de blocage et un régime saturé par exemple), \times qu'il n'y a pas différenciation.

La figure 6.6(a) correspond à l'algorithme Poisson, avec $r_1 = 100\text{kb/s}$ et $r_2 = 300\text{kb/s}$. La différenciation se produit (sauf exception) pour les charges $\rho = 1.20$ et $\rho = 1.40$ pourvu que la contribution relative des flots de faible débit crête ne soit pas trop importante. Le critère important est la variance du trafic au moment où la charge moyenne avoisine le seuil d'admission défini en fonction de p (ici 0.79 pour $p = 100\text{kb/s}$) ; si elle est trop faible (forte proportion de r_1), alors l'algorithme (conservatif) ne fera aucune différence entre les flots et commencera à bloquer les nouvelles arrivées.

Avec un débit $r_2 = 200\text{kb/s}$, aucune différenciation n'est observée : la variance de l'agrégat de trafic ne compense pas suffisamment le conservatisme du MBAC, même avec l'algorithme MinVar. Pour les valeurs supérieures à 300kb/s , la différenciation est d'autant plus marquée que le débit est important. Nous ne présentons pas ces résultats ici.

L'utilisation de l'algorithme MinVar dans 6.6(b) permet une meilleure estimation de la variance de l'agrégat des flots non différenciés, et donc d'obtenir un seuil d'admission plus propice à la différenciation des flots. Nous avons vu que l'efficacité de MinVar

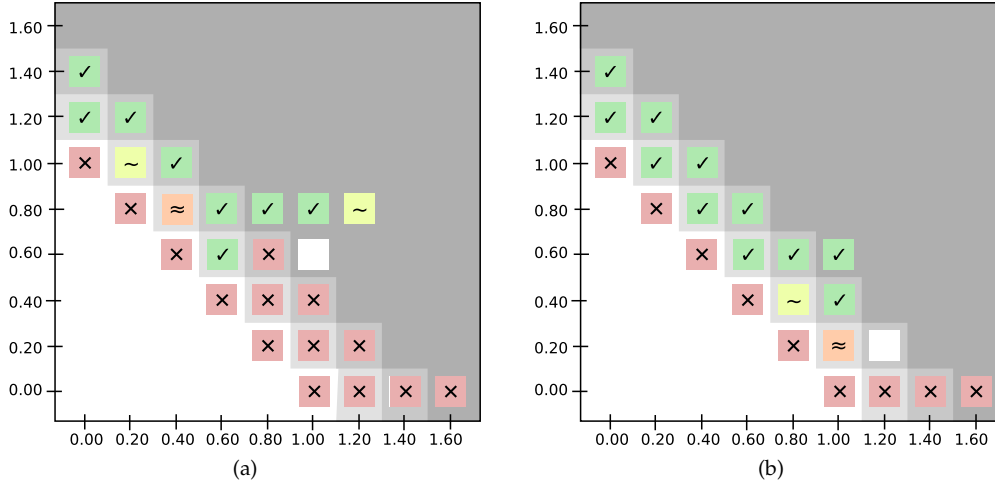


FIGURE 6.6: Différenciation réalisée par les variantes Poisson (a) et MinVar (b) du MBAC XP, avec $p = 100\text{kb/s}$, $r_1 = 100\text{kb/s}$ et $r_2 = 300\text{kb/s}$, pour différentes contributions à la charge de chaque classe de flots

est d'autant plus marquée que le paramètre k est adapté aux débits en présence, c'est pourquoi dans la suite nous le considérons.

Les figures 6.7(a) et 6.7(b) correspondent à l'algorithme Poisson lorsque $r_1 = 50\text{kb/s}$ et $r_2 = 300\text{kb/s}$ et pour des valeurs respectives de k égales à 1 et 2. Globalement, la différenciation est moins bonne que dans les cas précédents. Le seuil d'admission de l'algorithme Poisson étant défini à partir de p et non de r_1 , la raison étant que pour une charge donnée, la variance du trafic est plus faible que précédemment.

Dans le cas $k = 2$, le résultat est d'ailleurs moins bon. Pourquoi ?

De même que précédemment, l'utilisation de MinVar dans la figure 6.7(c) permet d'améliorer les performances, notamment lorsque $k = 2$ dans la figure 6.7(d), où les flots sont parfaitement différenciés dans tous les cas illustrés.

Nous détaillons le résultat d'une partie de ces simulations avec l'algorithme MinVar, avec $\rho = 1.20$, $p = 100\text{kb/s}$ et $r_2 = 300\text{kb/s}$. Les tableaux 6.3a et 6.3b correspondent à $r_1 = 50\text{kb/s}$, 6.4a et 6.4b à $r_1 = 100\text{kb/s}$, avec respectivement $k = 1$ et $k = 2$.

Le tableau montre que la différenciation est variable en fonction des paramètres de trafic. La discrimination est reflétée dans les différentes valeurs des taux de blocage bl et de pertes de paquets lo . Il est nécessaire dans tous les cas d'éliminer les 20% de charge en trop ($(bl + lo > 16.7\%)$). Dans les cas sans discrimination, cela est réalisé uniquement par le blocage, et l'utilisation demeure relativement basse. Dans les autres cas, les flots de débits crête les plus importants perdent des paquets ce qui réduit la proportion de flots bloqués (qui est la même pour les deux classes) et l'utilisation est proche de 100%. Ils sont servis en fonction de la capacité laissée disponible par les flots prioritaires, leur qualité de service étant assurée par le maintien d'un *fair rate* minimal à long terme.

Ainsi, les flots de plus faible débit crête sont protégés (*backlog* et débordement négligeables) et ne subissent aucune perte. Lorsque ces flots sont en majorité, la variance globale du trafic ne suffit pas à saturer le lien et entraîner la différenciation. C'est pourquoi le blocage est alors principalement dû au PL.

Lorsque la différenciation se produit, on observe un taux de paquets traités en *backlog* inférieur à 85%. Il s'agit soit de périodes où temporairement la différenciation n'a pas lieu, soit de paquets initiant une rafale *on* dans notre modèle de trafic. Ces derniers sont

6.5 Evaluation des algorithmes

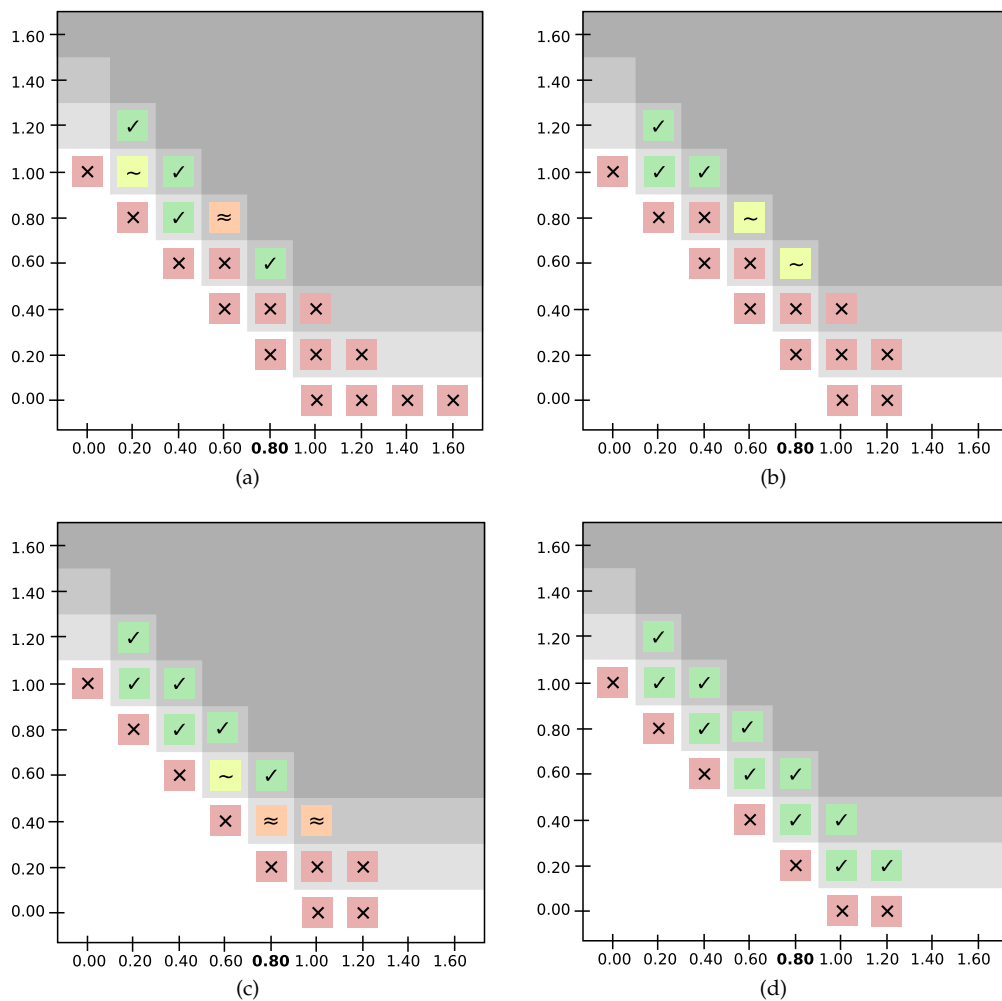


FIGURE 6.7: Différenciation réalisée par les variantes Poisson – (a) pour $k = 1$ et (b) pour $k = 2$ – et MinVar – (c) pour $k = 1$ et (d) pour $k = 2$ – du MBAC XP, avec $p = 100\text{kb/s}$, $r_1 = 50\text{kb/s}$ et $r_2 = 300\text{kb/s}$, pour différentes contributions à la charge de chaque classe de flots

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	7.61e-01	81.15	33.36	0.00	0.00	0.00	0.00	33.36	0.00
1.00	0.20	4.11e-01	83.11	29.99	0.00	11.1	0.00	0.00	29.99	0.00
0.80	0.40	1.33e-01	86.58	25.84	0.00	31.0	0.00	6.45	25.83	0.01
0.60	0.60	2.22e-02	97.43	3.64	0.00	80.5	0.00	26.83	3.08	0.55
0.40	0.80	1.11e-02	99.87	0.00	0.00	89.2	0.00	22.44	0.00	0.00
0.20	1.00	1.67e-02	99.61	0.00	0.00	83.9	0.00	17.12	0.00	0.00
0.00	1.20	2.22e-02	99.33	0.08	0.00	78.4	0.00	12.33	0.08	0.00

(a)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	5.50e-01	87.89	28.03	5.38e-02	0.00	0.00	0.00	28.03	0.00
1.00	0.20	7.78e-02	97.36	17.19	7.05e-02	77.8	0.00	11.28	14.90	2.29
0.80	0.40	0.00	99.89	6.48	4.02e-03	90.3	0.00	29.40	2.15	4.32
0.60	0.60	0.00	99.92	0.70	0.00	90.7	0.00	29.16	0.00	0.70
0.40	0.80	0.00	99.87	0.00	0.00	89.2	0.00	22.44	0.00	0.00
0.20	1.00	0.00	99.61	0.00	0.00	83.9	0.00	17.12	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	78.5	0.00	12.40	0.00	0.00

(b)

 TABLE 6.3: Performance de MinVar pour des flots hétérogènes avec $r_1 = 50\text{kb/s}$, $r_2 = 300\text{kb/s}$ et $p = 100\text{kb/s}$, pour $k = 1$ (a) et $k = 2$ (b)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	3.39e-01	83.52	30.73	1.44e-01	0.00	0.00	0.00	30.73	0.00
1.00	0.20	9.44e-02	89.51	25.52	3.21e-01	34.0	0.00	1.68	25.50	0.02
0.80	0.40	5.56e-03	93.31	17.50	9.26e-01	59.2	0.00	13.47	17.39	0.11
0.60	0.60	5.56e-03	95.88	6.11	5.50e-01	72.9	0.00	21.41	6.02	0.08
0.40	0.80	0.00	92.86	9.37	4.05e-03	60.5	0.00	14.13	9.37	0.00
0.20	1.00	2.78e-02	96.91	2.61	0.00	75.2	0.00	15.31	2.61	0.00
0.00	1.20	2.22e-02	99.33	0.08	0.00	78.4	0.00	12.33	0.08	0.00

(a)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	1.78e-01	84.85	29.62	2.49e-01	0.00	0.00	0.00	29.62	0.00
1.00	0.20	0.00	92.53	23.25	1.06	47.8	0.00	3.73	23.04	0.22
0.80	0.40	0.00	97.97	10.55	1.91	78.9	0.00	20.54	10.01	0.54
0.60	0.60	0.00	96.63	5.79	6.80e-01	74.6	0.00	22.47	5.63	0.17
0.40	0.80	0.00	87.66	17.10	7.12e-03	40.1	0.00	9.32	17.10	0.00
0.20	1.00	0.00	99.70	0.00	0.00	84.6	0.00	16.93	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	78.5	0.00	12.40	0.00	0.00

(b)

 TABLE 6.4: Performance de MinVar pour des flots hétérogènes avec $r_1 = 100\text{kb/s}$, $r_2 = 300\text{kb/s}$ et $p = 100\text{kb/s}$, pour $k = 1$ (a) et $k = 2$ (b)

6.5 Evaluation des algorithmes

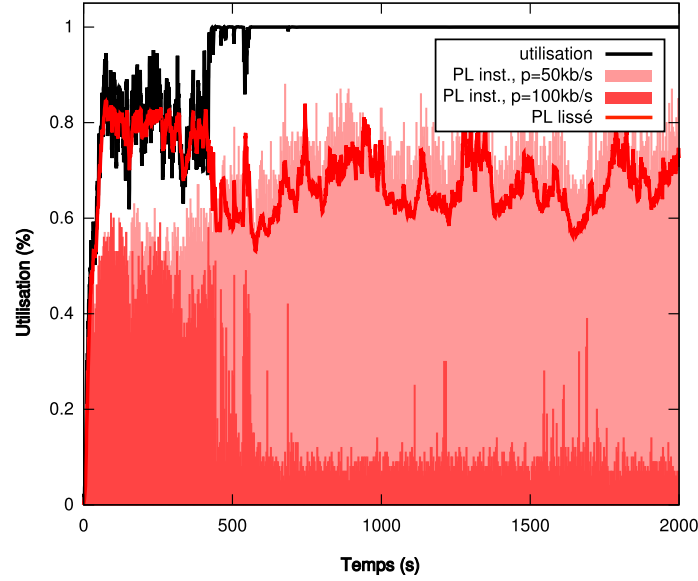


FIGURE 6.8: Illustration de la différenciation implicite : la figure représente l'évolution de différents paramètres de performance pour un scénario hétérogène avec $r_1 = 50\text{kb/s}$, $r_2 = 200\text{kb/s}$ et $p = 100\text{kb/s}$, où chaque classe contribue pour 50% de la charge globale $\rho = 1.20$.

à juste titre traités comme prioritaire puisque le débit instantané du flot à cet instant est très faible. Une variation de la durée relative de chaque bursts (variation du rapport débit moyen à débit crête égal) changerait quantitativement ces résultats, mais le comportement serait identique.

Ce rapport peut être mis en relation avec l'adaptation de la fenêtre d'émission (*cwnd*) d'un flot TCP tandis que son débit crête reste relativement fixe (débit d'accès). La taille du buffer aurait alors une influence sur le taux de pertes de paquets, et ainsi le débit réalisé.

Différenciation implicite

Nous illustrons le mécanisme de différenciation implicite au travers du cas représenté en gras dans le tableau 6.4. La figure 6.8 montre l'utilisation du lien, le *priority load* global lissé, ainsi que la contribution de chaque classe à cette charge dans chaque intervalle de temps.

Les premières 120s correspondent à un régime transitoire où le *priority load* inclue tout le trafic entrant. Tandis que le lien commence à saturer, les flots de plus fort débit crête voient leurs paquets espacés par l'ordonnanceur, deviennent *backlogged* et cessent de contribuer au *priority load*. Il convient de noter que le MBAC permet au lien d'atteindre ce régime puisque l'estimation de la variance est donnée par l'approximation Poisson, ce qui conduit à un débordement qui se traduit en *backlog*. Les deux indicateurs de congestion étant satisfaisants une fois la différenciation effectuée, il est donc à nouveau possible d'accepter des flots.

L'origine du blocage des flots

Quand les flots de fort débit crête deviennent *backlogged*, leurs paquets ne contribuent plus au *priority load*. Dans certains exemples des tableaux 6.3 et 6.4, ces flots

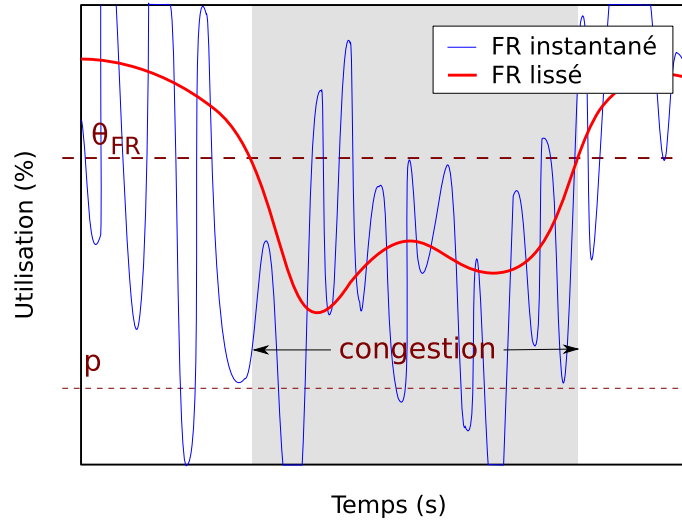


FIGURE 6.9: Illustration de l'évolution du *fair rate* par rapport au seuil θ_{FR} et de son impact sur les conditions d'admission.

différenciés sont alors parfois bloqués par la condition sur le *fair rate*. La figure 6.9 illustre comment le *fair rate* instantané varie typiquement rapidement d'un intervalle de temps à l'autre alors que l'estimation lissée fluctue autour du seuil d'admission θ_{FR} . Dans cette situation, les flots seraient bloqués pendant la période du milieu.

6.5.7 Impact de la gigue

Modèle de trafic

En pratique les flots de débit crête nominal p voient leurs paquets subir des délais variables et acquérir de la gigue au fur et à mesure qu'ils sont multiplexés dans les files d'attente des routeurs successifs. L'étude de cette variance du débit des flots est très difficile, mais il est important de comprendre comment elle affecte la performance de notre MBAC. Une hypothèse de pire cas, selon la conjecture dite de gigue négligeable [?], est que les flots émettent des paquets selon un processus de Poisson de débit p pendant les périodes *on*³. En pratique, la gigue ainsi réalisée sera clairement un pire cas, notamment dans le cadre d'un réseau équipé de routeurs Cross-Protect où le *fair queueing* tend à restaurer l'espacement original des paquets des flots gigués. En outre, l'évaluation du MBAC soumis à un trafic Poisson au niveau paquet reste intéressante à part entière.

Impact de la gigue pour des flots homogènes

La figure 6.10 représente le débit des flots dans un petit intervalle de temps et le compare au débit cible p et au *fair rate* instantané. C'est le débit auquel les paquets d'un flot *backlogged* sont servis durant un intervalle de temps. La figure illustre comment le fait que les arrivées de paquets sont Poisson et non pas déterministes influe sur la possibilité que les flots deviennent *backlogged*.

Les résultats de simulation correspondant à des scénarios identiques à la section 6.5.5 sont présentés dans le tableau 6.5, pour $p = 100\text{kb/s}$ et une taille de buffer $B = 100$ ou

3. Nous avons développé un module ns-2 de génération de trafic disponible sur <http://jordan.auge.free.fr/misc>

6.5 Evaluation des algorithmes

	p	r	ov	ut	bl	bk	lo	bl_{PL}	bl_{FR}
Poisson	50	50	1.67e-02	97.67	18.41	2.31e+01	0.50	0.14	18.27
	100	100	0.00	99.70	11.95	5.41e+01	5.76	0.00	11.95
	100	50	6.67e-02	82.52	31.35	4.67	0.00	31.35	0.00
	100	300	0.00	99.74	0.00	8.07e+01	15.35	0.00	0.00
MinVar	50	50	1.67e-02	97.67	18.41	2.31e+01	0.50	0.14	18.27
	100	100	0.00	99.80	12.24	5.49e+01	5.80	0.00	12.24
	100	50	2.50e-01	87.78	26.83	6.93	0.00	26.83	0.00
	100	300	0.00	99.74	0.00	8.07e+01	15.35	0.00	0.00

(a) $k=1$, $B=100$

	p	r	ov	ut	bl	bk	lo	bl_{PL}	bl_{FR}
Poisson	50	50	1.11e-02	97.13	19.20	2.27e+01	0.00	0.31	18.89
	100	100	0.00	96.66	19.58	4.10e+01	0.00	2.20	17.38
	100	50	6.67e-02	82.52	31.35	4.67	0.00	31.35	0.00
	100	300	0.00	100.00	0.00	9.49e+01	15.10	0.00	0.00
MinVar	50	50	1.11e-02	97.13	19.20	2.27e+01	0.00	0.31	18.89
	100	100	5.56e-03	96.47	20.27	4.09e+01	0.00	2.03	18.23
	100	50	2.50e-01	87.78	26.83	6.93	0.00	26.83	0.00
	100	300	0.00	100.00	0.00	9.49e+01	15.10	0.00	0.00

(b) $k=1$, $B=1000$

	p	r	ov	ut	bl	bk	lo	bl_{PL}	bl_{FR}
Poisson	50	50	0.00	97.59	18.43	2.28e+01	0.48	0.41	18.02
	100	100	0.00	99.72	11.74	5.43e+01	5.79	0.00	11.74
	100	50	0.00	82.63	31.24	4.72	0.00	31.24	0.00
	100	300	0.00	99.74	0.00	8.07e+01	15.35	0.00	0.00
MinVar	50	50	0.00	97.59	18.43	2.28e+01	0.48	0.41	18.02
	100	100	0.00	99.74	12.65	5.48e+01	5.89	0.00	12.65
	100	50	1.11e-02	92.30	23.18	1.18e+01	0.02	22.67	0.51
	100	300	0.00	99.74	0.00	8.07e+01	15.35	0.00	0.00

(c) $k=2$, $B=100$

	p	r	ov	ut	bl	bk	lo	bl_{PL}	bl_{FR}
Poisson	50	50	0.00	96.91	19.45	2.27e+01	0.00	0.19	19.26
	100	100	0.00	97.06	19.30	4.26e+01	0.00	0.80	18.50
	100	50	0.00	82.63	31.24	4.72	0.00	31.24	0.00
	100	300	0.00	100.00	0.00	9.49e+01	15.10	0.00	0.00
MinVar	50	50	0.00	96.91	19.45	2.27e+01	0.00	0.19	19.26
	100	100	0.00	97.11	19.73	4.36e+01	0.00	0.79	18.95
	100	50	0.00	92.16	23.27	1.21e+01	0.00	21.43	1.84
	100	300	0.00	100.00	0.00	9.49e+01	15.10	0.00	0.00

(d) $k=2$, $B=1000$

TABLE 6.5: Performance du MBAC Cross-Protect pour des flots gigués dans un cas homogène où tous les flots ont le même débit crête.

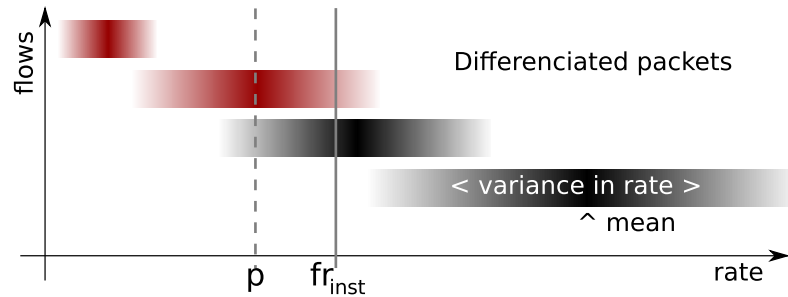


FIGURE 6.10: Illustration de l'impact de la gigue sur le débit instantané des flots et conséquences pour la différenciation implicite.

$B = 1000$ paquets. On peut distinguer deux comportements différents de l'algorithme, en fonction du débit crête des flots r .

Lorsque la valeur de r est faible devant p (cas $r = 50$ et $p = 100\text{kb/s}$), les fluctuations du débit restent suffisamment peu importantes pour que les flots restent protégés (le débit reste généralement inférieur au *fair rate*). D'autant plus que la condition sur PL assure que le *fair rate* instantané reste supérieur à p presque sûrement.

Sinon, les fluctuations du débit crête entraînent un fort taux de paquets *backlogged*; le *priority load* reste faible tandis que l'utilisation avoisine les 100%. La forte variance du trafic rend l'apport de l'algorithme MinVar qui offre sensiblement des résultats identiques à Poisson.

Une charge importe de flots de la classe 1 implique une compétition entre un grand nombre de flots, et le partage de la capacité du lien rend la condition sur le *fair rate* bloquante. Le *fair rate instantané* est alors en moyenne égal à θ_{FR} . En fonction de la valeur de $p \leq \theta_{FR}$, on observe un taux de paquets *backlogged* plus ou moins important, ce qui reflète que les flots ne sont plus protégés momentanément. Toutefois le débit effectivement protégé reste proche de p , ce qui souligne l'imprécision du contrôle d'admission lorsqu'il devient dépendant du *fair rate* uniquement.

Intuitivement, on peut comprendre les débordements comme les moments où un grand nombre de flots se trouve avoir un débit instantané très élevé, alors que le *fair rate* instantané est bas.

On a ici un genre de pire cas, alors que notre algorithme contrôle plus un comportement "moyen" (?)

Une telle imprécision de l'algorithme ne permet ainsi pas d'isoler complètement les flots les moins gigués de ceux présentant des fluctuations de débit plus importantes lorsque ces derniers contribuent à une forte charge du lien. Elle disparaît lorsque p est suffisamment inférieur à θ_{FR} , sinon il convient d'avoir un buffer suffisamment dimensionné afin d'éviter des pertes de paquets.

Ne peut-on pas alors s'attendre à de mêmes conséquences pour des flots VBR? On n'explique pas assez la différence VBR/gigue poisson...

Dimensionnement du buffer

Les résultats de simulations identiques avec une taille de buffer $B = 1000$ paquets sont présentés dans les tableaux 6.5(d) et 6.5???. Ils illustrent qu'il existe un compromis sur la taille du buffer en ce qui concerne l'excédent de trafic, qui est éliminé soit par des pertes de paquets, soit par un blocage des flots. Par exemple, pour $r = p = 100\text{kb/s}$, on passe de 5% de pertes et 12% de flots bloqués avec $B = 100$, à un taux de blocage de 18% avec

6.5 Evaluation des algorithmes

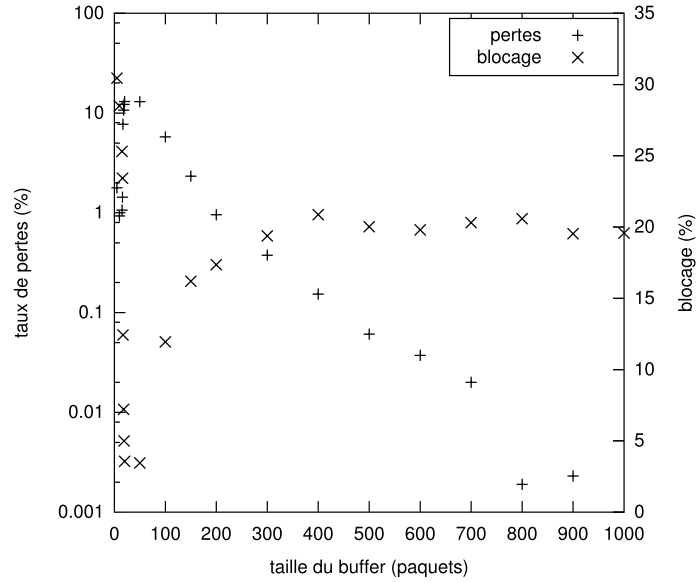


FIGURE 6.11: Illustration du compromis taux de pertes de paquets / taux de blocage des flots en fonction de la taille du buffer pour $r = p = 100\text{kb/s}$

aucune perte pour $B = 1000$.

Il faut bien comprendre que les pertes sont considérées différemment selon qu'elles concernent des paquets prioritaires (cela peut arriver dans des cas extrêmes) ou non-prioritaires. La valeur du *priority load* est basée sur le volume de trafic prioritaire *entrant*, et ne sera pas diminuée par une perte. A l'inverse, les pertes sont considérées comme part intégrante du fonctionnement des protocoles constituant le trafic *backlogged*, et les mesures sur *fair rate* concernent le débit effectif *sortant*.

Le dimensionnement adéquat du buffer, que nous n'approfondiront pas dans ce contexte, et ainsi indispensable à l'élimination de la surcharge des flots normalement prioritaires mais qui se retrouvent *backlogged*. L'absence de pertes pour un buffer suffisamment grand est d'ailleurs une preuve du bon fonctionnement de l'ordonnanceur qui espace les paquets.

La figure 6.11 nous présente à titre illustratif ce compromis pertes/blocage pour différentes valeurs de B , dans le cas $r = p = \theta_{FR} = 100\text{kb/s}$ qui est le plus défavorable ici.

Le cas $r = 300\text{kb/s}$ confirme qu'une plus grande taille de buffer ne nuit pas à la différenciation des flots de fort débit crête, qui ne sont pas bloqués mais continuent voir leurs paquets éliminés en premier lorsque le buffer devient saturé.

Impact de l'algorithme sur la gigue des flots

Nous avons vu précédemment que l'imprécision du contrôle d'admission peut parfois introduire une légère gigue pour les flots CBR de débit voisin de p , lorsqu'ils sont momentanément servis à un débit inférieur à p . Rappelons toutefois que nous considérons un pire cas où l'ensemble des flots présentent une gigue Poisson, ce qui est peu susceptible de se produire dans un scénario réel.

Les flots gigués, par hypothèse de débit moyen $r \leq p$, voient quand à eux leurs paquets espacés par l'ordonnanceur lorsque le lien est saturé, et ils sont alors servis à un débit équitable $r_{eq} \geq p$ (modulo ces imprécisions), lorsque leur débit instantané est supérieur à $_{eq}$. En mettant ces paquets en attente dans le buffer, l'ordonnanceur procède donc à une suppression au moins partielle de la gigue. C'est un avantage supplémentaire de la

τ (p équiv.)	r	ov	ut	bl
160 ms (50)	50	1.15e-02	84.24	30.11
80 ms (100)	100	1.50e-02	79.00	33.06
80 ms (100)	50	1.40e-02	79.53	33.79
80 ms (100)	300	1.83e-02	77.99	37.31

 FIGURE 6.12: Performance du MBAC GT avec des flots gigués ($B = 100$)

différenciation implicite réalisée par Cross-Protect.

Comparaison à l'algorithme de Grossglauser et Tse

Nous simulons le même scénario avec un lien FIFO muni de l'algorithme (GT) (voir tableau 6.12). Pour les trois cas où $r \leq p$, on retrouve une utilisation sensiblement égale, avec la même limite posée par l'intervalle sur la mesure de la variance. La gigue n'affecte pas le fonctionnement normal de l'algorithme qui limite la charge dans tous les cas, et ne profite donc pas du fait que les flots ont un débit crête supérieur à p . Il s'en suit un fort taux de blocage. On n'a pas ici de notion de *fair rate* pour deviner le traitement apporté aux flots CBR, puisque tous les flots sont traités à l'identique. On conserve toutefois la conjecture de gigue négligeable, qui assure une performance satisfaisante pour l'ensemble des flots.

Conséquences sur la différenciation implicite des flots

Les tableaux 6.6 et 6.7 présentent les résultats de scénarios similaires à ceux de la sous-section 6.5.6 lorsque les flots initialement CBR présentent de la gigue. La taille du buffer est à nouveau de 100 paquets. Son impact est une conséquence directe des observations effectuées précédemment ; la figure 6.10 illustrant notamment l'impact des différents débits crête des flots.

Lorsque le trafic est entièrement constitué de flots de la classe 1, on retrouve le cas homogène précédent, notamment où les flots de débit $r_1 = 50\text{kb/s}$ continuent d'être protégés puisque le débit instantané reste généralement inférieur au *fair rate*. Ce n'est plus le cas dès lors que la proportion de flots r_2 n'est plus nulle. L'admission des flots est majoritairement régulée par le seuil sur le *priority load*, le *fair rate* ne devenant bloquant que dans le cas homogène avec $r_1 = p = 100\text{kb/s}$ ainsi que rarement dans certaines situations où de nombreux flots se retrouvent *backlogged* : forte contribution de la charge prioritaire, ainsi que des flots élastiques se partageant la capacité laissée disponible.

L'efficacité de la différenciation se transcrit comme précédemment au niveau du taux de pertes de paquets de chaque classe de trafic. Il est très faible pour les flots de la classe 1 (dû à la taille du buffer) et l'on constate qu'il diminue fortement avec le débit crête des flots, ainsi qu'avec la contribution de la classe prioritaire à la charge globale. Ceci est un autre effet bénéfique de la différenciation implicite, sans laquelle la dépendance serait proportionnelle à la charge globale.

Garder les simulations avec $k = 2$ vu qu'on ne les utilise pas ?

6.5 Evaluation des algorithmes

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	2.61e-01	87.83	27.25	7.03	0.00	0.00	0.00	27.25	0.00
1.00	0.20	4.44e-02	94.80	18.88	1.33e+01	7.28e+01	0.02	13.82	17.33	1.55
0.80	0.40	0.00	99.82	5.10	2.02e+01	9.03e+01	0.05	32.45	0.97	4.13
0.60	0.60	0.00	99.89	0.45	1.72e+01	8.95e+01	0.02	29.58	0.00	0.45
0.40	0.80	0.00	99.79	0.00	1.36e+01	8.72e+01	0.01	22.68	0.00	0.00
0.20	1.00	0.00	99.66	0.00	1.15e+01	8.23e+01	0.00	17.57	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	7.76e+01	13.03	0.00	0.00	0.00

(a)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	0.00	92.03	23.97	1.17e+01	0.00	0.03	0.00	23.02	0.95
1.00	0.20	0.00	96.34	16.73	1.59e+01	7.81e+01	0.04	18.60	11.75	4.98
0.80	0.40	0.00	99.89	4.92	2.06e+01	9.08e+01	0.05	33.58	0.12	4.79
0.60	0.60	0.00	99.89	0.45	1.72e+01	8.96e+01	0.02	29.66	0.00	0.45
0.40	0.80	0.00	99.79	0.00	1.36e+01	8.72e+01	0.01	22.68	0.00	0.00
0.20	1.00	0.00	99.66	0.00	1.15e+01	8.23e+01	0.00	17.57	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	7.76e+01	0.00	13.03	0.00	0.00

(b)

TABLE 6.6: Performance de MinVar pour des flots gigués hétérogènes avec $r_1 = 50\text{kb/s}$, $r_2 = 300\text{kb/s}$ et $p = 100\text{kb/s}$, pour $k = 1$ (a) et $k = 2$ (b)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	0.00	99.61	12.44	5.46e+01	0.00	5.97	0.00	0.03	12.42
1.00	0.20	0.00	99.88	6.45	5.24e+01	9.12e+01	4.90	43.59	0.00	6.45
0.80	0.40	0.00	99.94	0.92	4.74e+01	9.05e+01	3.03	38.72	0.00	0.92
0.60	0.60	0.00	99.73	0.00	3.81e+01	8.75e+01	0.79	27.97	0.00	0.00
0.40	0.80	0.00	99.43	0.00	3.02e+01	8.35e+01	0.27	20.05	0.00	0.00
0.20	1.00	0.00	99.67	0.00	2.79e+01	8.31e+01	0.13	18.23	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	7.76e+01	0.00	13.03	0.00	0.00

(a)

ρ_1	ρ_2	ov	ut	bl	bk_1	bk_2	lo_1	lo_2	bl_{PL}	bl_{FR}
1.20	0.00	0.00	99.63	12.31	5.38e+01	0.00	5.66	0.00	0.00	12.31
1.00	0.20	0.00	99.87	5.96	5.25e+01	9.13e+01	4.87	44.05	0.00	5.96
0.80	0.40	0.00	99.94	0.90	4.75e+01	9.06e+01	3.11	38.97	0.00	0.90
0.60	0.60	0.00	99.73	0.00	3.80e+01	8.75e+01	0.79	27.97	0.00	0.00
0.40	0.80	0.00	99.43	0.00	3.02e+01	8.35e+01	0.27	20.05	0.00	0.00
0.20	1.00	0.00	99.67	0.00	2.79e+01	8.31e+01	0.13	18.23	0.00	0.00
0.00	1.20	0.00	99.34	0.00	0.00	7.76e+01	0.00	13.03	0.00	0.00

(b)

TABLE 6.7: Performance de MinVar pour des flots gigués hétérogènes avec $r_1 = 100\text{kb/s}$, $r_2 = 300\text{kb/s}$ et $p = 100\text{kb/s}$, pour $k = 1$ (a) et $k = 2$ (b)

6.5.8 Contrôle d'admission en régime non-stationnaire

Flash crowds

Comme il l'a été évoqué précédemment dans la sous-section 6.4.8, le contrôle d'admission s'avère particulièrement utile dans des situations non-stationnaires où le lien connaît subitement un très fort taux d'arrivées de flots. On parle alors de *flash crowd*. Ces cas se produisent par exemple lorsqu'une panne se produit sur un lien, causant le reroutage des flots en cours vers un autre chemin. Ces flots en cours apparaissent comme une série de nouveaux flots qui doivent être gérés par le contrôle d'admission présent sur les liens de remplacement.

Le comportement du trafic peut devenir particulièrement complexe lorsque par exemple les flots TCP en cours subissent des pertes successives de paquets, et passent en mode *slow start* suite à un *timeout* avant de recommencer à envoyer des paquets au travers de la nouvelle route. Un MBAC peut être amené à accepter un nombre excessif de flots qui paraissent avoir un débit faible. Ces flots augmentent alors progressivement leur débit, conduisant éventuellement à une congestion que l'on souhaiterait éviter.

La plupart des algorithmes de MBAC sont vulnérables à ce genre de scénarios, car l'analyse de tels phénomènes est très délicate. Il conviendrait par exemple de corriger de telles "erreurs" lors de l'admission par la possibilité de préempter certaines connexions afin de protéger l'ensemble du trafic. Nous nous contenterons dans cette dernière section d'illustrer la performance de notre algorithme dans de tels scénarios, en nous en remettant à des travaux ultérieurs.

Scénario de simulation

Nous simulons un scénario très simple décrit par la figure 6.3. Nous avons deux topologies *dumbbell* identiques initialement isolées sur lesquelles nous générons une charge de 60% avec des flots *on-off* de caractéristiques identiques à la section précédente, et de débit crête $r = p = 100\text{kb/s}$.

La durée de la simulation est de 1000s; le lien central du second *dumbbell* est cassé à 500s ce qui cause le reroutage du trafic sur le lien central du premier *dumbbell*.

L'intervalle d'échantillonnage du MBAC est fixé à $\tau = L/p$.

Résultats préliminaires avec le MBAC XP

La figure ?? représente les indicateurs de performance mesurés FR et PL pendant cet événement de *flash crowd* (à la fois leur valeur instantanée ainsi que l'estimateur lissé). La figure 6.13c montre les valeurs de la probabilité de *backlog* dans des intervalles successifs.

Les résultats démontrent que l'algorithme échoue à préserver la performance dans ce cas. Trop de flots sont admis après le basculement puisque la mesure courante de la charge reset en dessous du seuil d'admission à cause du lissage et de la persistance de la période précédente de faible trafic. Cela induit une période de surcharge qui dure quelques dizaines de secondes avant que le lien ne recouvre l'état normal en régime stationnaire considéré dans la section 6.5.5 ci-dessus.

Une mesure préventive supplémentaire, comme proposée dans [?], est d'ajouter la valeur du débit crête cible p à la charge mesurée $b(t)$ pour chaque nouveau flot admis. De cette manière le trafic nouveau est immédiatement pris en compte dans l'estimateur $A(t)$ de la charge mesurée. Cependant, cela s'avère être trop conservatif quand les flots ont un débit crête bien inférieur à p ou, comme on l'observe dans des traces réelles, qu'il y a de nombreux flots d'un seul paquet.

Ces résultats préliminaires illustrent clairement la difficulté de contrôler le trafic au travers d'un contrôle d'admission dans le scénario de panne considéré. Nous prévoyons

6.5 Evaluation des algorithmes

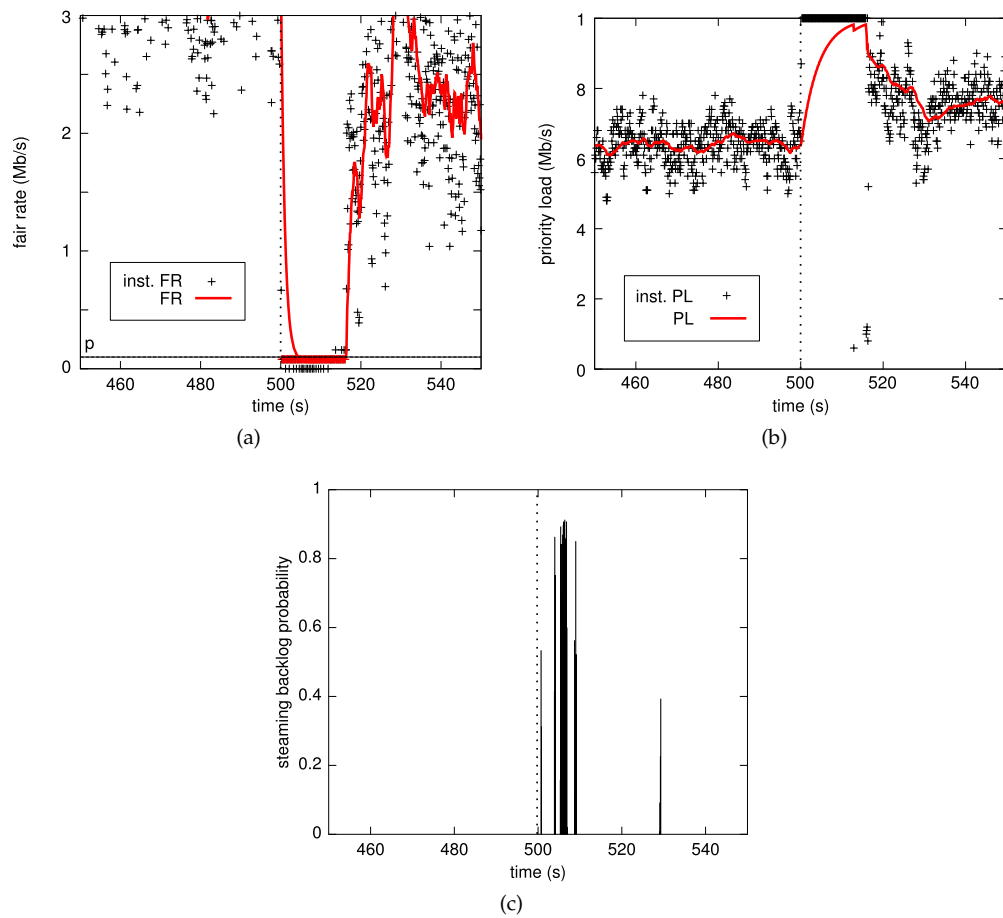


FIGURE 6.13: Evolution du *fair rate* (a), du *priority load* (b) et de la probabilité de *backlog* des paquets streaming (c) lors d'un scénario de *flash crowd*

de continuer cette évaluation dans nos prochains travaux, en intégrant l'impact significatif des flots en *slow start* comme discuté précédemment, ainsi que des hypothèses de trafic plus réalistes incluant une distribution *heavy-tail* de la durée des flots. Il est vraisemblable que des mesures plus radicales sont nécessaires pour empêcher une dégradation globale de la performance des flots, incluant une interruption préemptive de certains flots en cours.

6.6 Conclusions

Flow-Aware Networking, basé sur les mécanismes Cross-Protect, permet d'établir des garanties de performance pour le trafic *streaming* et élastique, sans nul besoin de la complexité d'un marquage des paquets afin d'effectuer une discrimination des classes de service. Il reste cependant à concevoir et calibrer l'algorithme de MBAC approprié. Des travaux précédent suggèrent qu'un simple algorithme fondé sur une estimation du *fair rate* est suffisant lorsqu'une proportion significative du trafic est composée de flots élastiques *bottlenecked* sur le lien en question. Cependant, dans un réseau de cœur la majorité des flots est *bottlenecked* ailleurs, notamment par les liens d'accès des utilisateurs dont le débit est généralement bien plus faible que la capacité des liens.

Le trafic est alors généralement composé d'un mélange de flots à débit crête limité. Certains possèdent un débit inférieur au débit cible p (pour lequel les flots sont censés être traités en priorité) tandis que d'autres l'ont supérieur à p . La difficulté est de concevoir un MBAC qui permet à ces derniers de devenir *backlogged* (ils sont supposés capables d'ajuster leur débit) tout en préservant le traitement prioritaire pour les précédents. La tâche est rendue d'autant plus délicate que le MBAC ne peut recourir à aucune information sur les caractéristiques des flots.

Dans ce chapitre, nous avons proposé un MBAC simple basé sur une mesure de la moyenne et de la variance de la charge offerte à la file prioritaire de Cross-Protect. Cela diffère de l'algorithme proposé par Grossglauser et Tse [?] en ce que la variance est utilisée seulement si elle est inférieure à une estimation de la variance fondée sur l'hypothèse que tous les flots ont le débit p . Nos résultats de simulations montrent que cet algorithme parvient à effectuer la discrimination requise lorsque les débit crête des flots ne sont pas trop similaires.

Les évaluations effectuées en scénario de *flash crowd* sont moins encourageantes. De nombreux facteurs combinés rendent difficile le fait de trouver un compromis satisfaisant entre l'acceptation de trop de flots, qui mène à une période significative où la performance est dégradée, ainsi que d'être trop conservatif et de rejeter bien plus de flots que strictement nécessaire. Une analyse plus poussée de ce scénario est nécessaire étant donnée son importance pratique. Nos résultats préliminaires suggèrent qu'un simple contrôle d'admission peut ne pas être suffisant. Il est vraisemblable que l'interruption d'un sous-ensemble des flots en cours sera en plus nécessaire afin de restaurer un niveau de charge acceptable.

6.7 Additional results

6.7.1 Influence des paramètres α et β

Afin d'évaluer l'influence des paramètres α et β , nous effectuons diverses simulations avec des flots homogènes tels que $r = p = 100\text{kb/s}$, à la fois en régime stationnaire – nous mesurons les indicateurs de performance usuels – et en situations de *flash crowds* – nous nous intéressons au nombre d'admissions autorisés au vu de l'estimation du trafic actuel. Cette dernière valeur N est :

$$N = \frac{C - \alpha_q \sigma - A_t}{p}$$

Lorsque cette valeur se retrouve négative, cela indique que l'on a admis trop de flots précédemment.

- En régime stationnaire, peu de différences pour différentes valeurs de α , pour l'ensemble des indicateurs de performance. Tableau disponible.
- En régime non-stationnaire, α a plus d'importance, et il semblerait qu'une valeur moins conservative soit nécessaire (0.9 au lieu de 0.98) pour assurer la performance des flots lors du basculement. Dû aux caractéristiques des flots reroutés ?

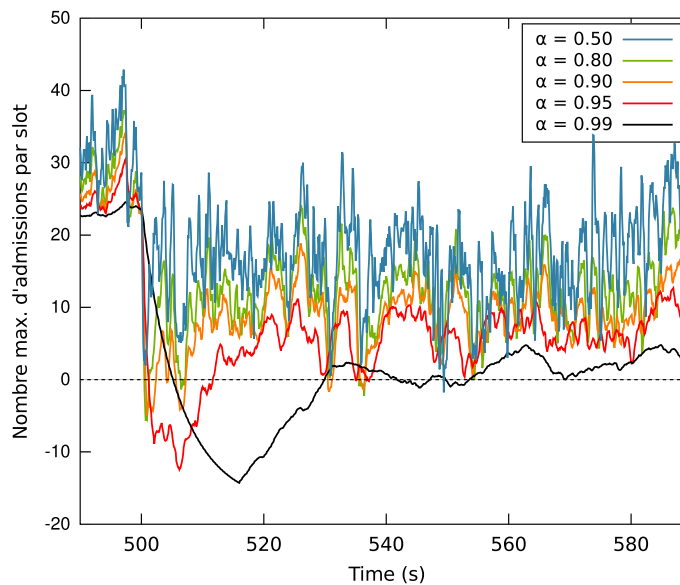


FIGURE 6.14: todo

- En régime stationnaire, la valeur de β n'est pas importante.
- En régime non-stationnaire, lorsque la valeur de α est correctement choisie (0.9 ici), la valeur de β ne change pas énormément les résultats.

6.7 Additional results

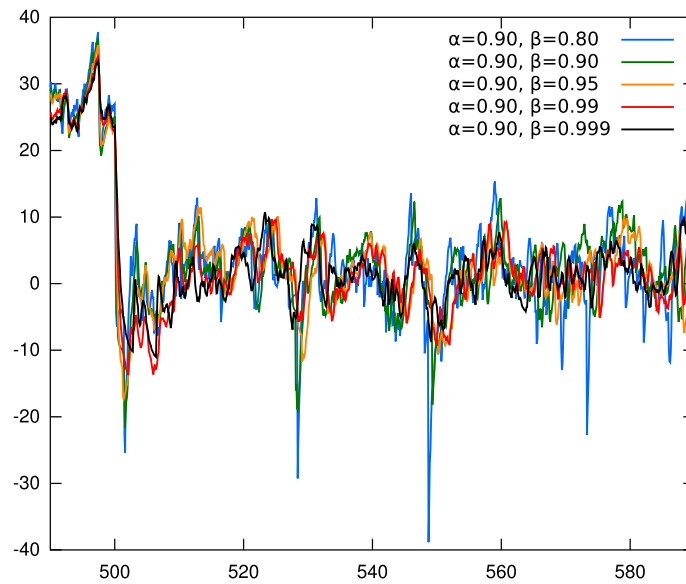


FIGURE 6.15: todo

- Par contre, lorsque α est égal à 0.98, la valeur de β devient beaucoup plus significative, et il semble bien qu'une valeur de β d'un ordre de grandeur plus important que α soit le meilleur choix.

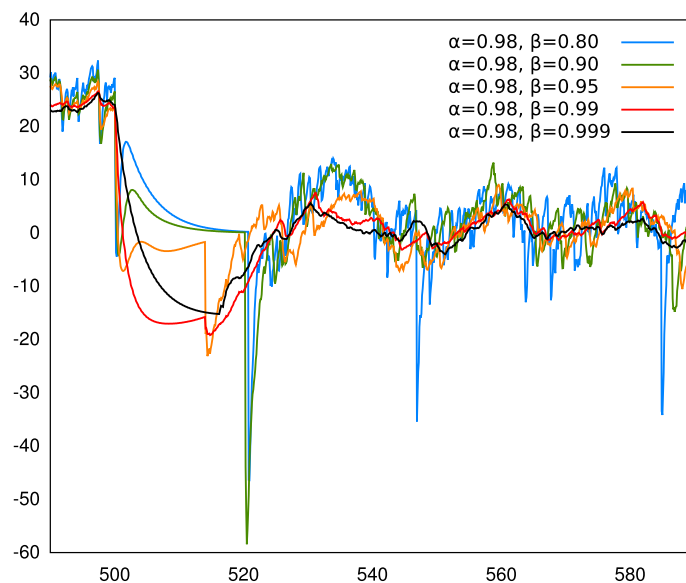


FIGURE 6.16: todo

6.7.2 Influence du taux d'activité des connexions

- Nous avons utilisé des connexions *on-off* où chaque période *on* et *off* a une durée exponentielle de moyenne 500ms. Faire varier la proportion relative de chaque période change le débit moyen – et non pas le débit crête – des flots, et ainsi le nombre moyen de flots en cours.

- Le premier paquet d'un flot qui arrive dans l'ordonnanceur après une période *off* suffisamment longue (fonction de p , ainsi que du *backlog* qu'a acquis le flot) sera traité en priorité puisque le flot aura à ce moment-là un débit crête faible. On comprend ainsi comment faire varier la durée de ces périodes pourra avoir une (légère) influence sur le nombre de paquets de flots *bottlenecked* qui passent dans la file prioritaire.
- Le fait d'avoir un nombre moyen de flots, et par conséquent un taux d'arrivée de flots supérieur, aura un impact en régime non-stationnaire où l'algorithme est le plus susceptible de commettre des erreurs d'admission (trop de flots admis).
- Correspondance avec une connexion TCP dont la fenêtre de congestion augmente : son débit crête max ne change pas (paquets dos-à-dos), mais le débit moyen varie (W/rtt) + *self-clocking*)
- cf simulations sur trace réelle dans le chapitre suivant pour de vraies connexions TCP

6.7.3 Influence de la durée des connexions

- Expo(60s)
- Quid d'une distribution hyperexponentielle ? pire ?

Chapitre 7

Analyse d'une trace de trafic

Buts : quel est l'intérêt d'un contrôle d'admission, quelle sont les différences dans un trafic en surcharge

7.1 Topologie et paramètres réseau

Deux sondes ont été placées sur deux interfaces POS reliant un routeur à Aubervilliers et un routeur à la Réunion. Le débit de chaque interface est de 155Mb/s. Le trafic capturé correspond au sens descendant sur chacune de ces interfaces (vers La Réunion).

La défaillance de l'une des interfaces a permis de capturer le 04/08/2005 une trace de trafic en régime de surcharge suite au reroutage du trafic de l'interface défaillante sur la seconde interface, qui fait donc figure d'interface de secours dans le scénario étudié.

Une deuxième série de captures a été effectuée le 07/09/2005 afin de servir de trace de référence.

L'un des objectifs de ces captures est de rejouer les traces capturées en considérant deux scénarios. Un premier scénario best-effort correspond au scénario réel, et un deuxième scénario où les mécanismes Cross-Protect sont activés (contrôle d'admission et PFQ). Les simulations considèrent dans la mesure du possible les mêmes paramètres opérationnels. Le délai de propagation de NC à NC est de 45ms, la taille des buffers de 1500 paquets. Afin de rejouer les traces capturées, il est nécessaire de les traiter, en distinguant le trafic élastique qui est rejoué au niveau flot, et le trafic streaming qui est rejoué au niveau paquet.

7.2 Traitement des traces

La sonde génère un seul fichier pour les deux interfaces au format ERF par block de 524ko. Un système de sauvegarde tournant permet de sauvegarder l'équivalent en trafic de 250Go. Voici la liste des informations disponibles dans ce fichier.

7.2.1 Traitement du trafic élastique

Par trafic élastique on entend l'ensemble des paquets TCP. Il s'agit de reconstituer à partir de ces paquets les flots TCP. Le flot est constitué du quintuplet classique, et la temporisation associé à chaque flot est de 20 secondes. Pour chaque flot ainsi défini, nous calculons :

- l'identifiant du flot (numéro de séquence croissant avec l'ordre d'arrivée, en tenant compte des flots UDP),

- le temps de début du flot (estampille de son premier paquet),
- la taille du flot (correspond à la somme de la taille des paquets (payload + entetes TCP/IP/niveau2),
- le nombre de paquets (hors paquets SYN FIN RST, ne tient pas compte non plus d'éventuelles retransmissions). Les paquets résultants d'une fragmentations comptent pour un paquet à part entière.

Rejeu de la trace des flots élastiques

Les flots élastiques sont rejoués dans le simulateur ns en fonction de leur ordre d'arrivée. Pour leur rejeu, nous avons été amenés à effectuer les approximations suivantes, qui du reste n'ont pas d'impact sur les conclusions essentielles de l'étude.

- Association flot élastique - connexion TCP : pour ns, chaque flot TCP est une connexion TCP qui débute donc par l'envoi d'un paquet SYN (or vu la méthode d'identification des flots, un flot élastique pourrait correspondre à une partie d'une connexion si ses paquets sont espacés de plus de 20 secondes). Pas de paquet FIN ou autre pour clore la connexion.
- Taille des paquets TCP : dans ns, la taille des paquets d'une connexion TCP est constante. La taille des paquets d'un flot est une valeur moyenne obtenue en divisant la taille d'un flot, telle que définie dans le paragraphe précédent (incluant les en-têtes TCP/IP/Ethernet), par le nombre de paquets du flot. De cette valeur sont retranchés 40 octets correspondant à l'en-tête IP qui est automatiquement rajouté par ns.
- Retransmissions : l'occurrence d'une retransmission peut s'expliquer de deux manières différentes. Soit le paquet correspondant a été perdu en aval du lien observé, ce qui indique qu'il n'est pas le seul goulot d'étranglement, auquel cas les paquets retransmis devraient normalement être rejoués, car ces retransmissions pourraient se produire avec le contrôle d'admission activé au niveau du lien observé. Un délai d'attente important au niveau du lien observé peut aussi déclencher une retransmission. L'activation du contrôle d'admission dans ce deuxième cas évite la retransmission du paquet. Comme indiqué plus haut, nous avons préféré ne pas rejouer les paquets retransmis.

7.2.2 Traitement du trafic streaming

Le trafic streaming est tout autre trafic que TCP. Il est essentiellement constitué de paquets UDP, ICMP, paquets de contrôle, etc. Pour chaque paquet sont précisés l'instant d'observation du paquet (estampille)¹, la taille du paquet en octets (en-tête de niveau 2 inclus) et l'identifiant du flot auquel il appartient. Quelques exemples correspondant aux 5 premiers paquets non TCP sont présentés dans le tableau ??.

7.3 Motivations

7.4 Composition du trafic

7.5 Caractérisation des flots

Processus d'arrivée Longueur des flots

1. Instant de transmission du paquet sur la ligne, et non instant d'arrivée du paquet à la file d'attente, comme supposé

7.6 Partage de bande passante

7.7 Impatience

7.8 Etude par ACP

7.9 Rejeu du controle d'admission

Buts : expérimenter les algorithmes sur du trafic plus réel

7.10 Rejeu d'une trace au niveau paquet

Dans quel but ? Quelles approximations

7.11 Rejeu d'une trace au niveau flot

Mêmes questions, les outils Flots TCP, débits crête, retransmissions ? Quels outils dans ns ?

Chapitre 8

Self-Protect

8.1 Introduction sur la qualité de service dans le réseau d'accès

8.1.1 Un goulot d'étranglement actuellement dans le réseau d'accès

Garantir la qualité de service des flots à l'accès est actuellement problématique. Le dernier kilomètre est souvent une paire de cuivre dont les débits sont fixés par les normes ADSL, et il est difficile de surdimensionner ce lien. Certaines améliorations normalisées (ADSL2(+), voire propriétaires (Broadcom = compression des en-têtes ATM, utilisé dans le réseau Free) ReADSL permettent d'avoir une meilleure portée voire un meilleur débit, mais le gain reste mineur. Bien sûr de nouvelles technologies comme le VDSL existent mais ne sont pas déployées par manque de maturité ou d'utilité vu leur faible portée. La FTTH, dont l'arrivée prévoit de bouleverser l'utilisation du réseau permettra certainement de supprimer le goulot d'étranglement à l'accès mais l'investissement est lourd et FTTH ne sera pas déployé partout.

Cependant, contrairement au réseau de cœur, le nombre de flots en cours et/ou les débits sont généralement beaucoup plus faibles, ce qui permet de réaliser des mécanismes de QoS plus évoluée que dans le cœur de réseau (marquage, signalisation, analyse niveau 7).

8.1.2 Quels moyens pour garantir la QoS ?

L'augmentation des débits a permis un changement de l'usage du réseau (notamment l'apparition des offres triple-play), et le rôle de l'opérateur s'est vu évoluer vers celui de fournisseur de contenu. L'opérateur possédant le réseau possède un atout non négligeable qui est de pouvoir contrôler son propre trafic temps réel, et de pouvoir le marquer en confiance et ainsi le reconnaître et le protéger. D'autant plus que le lien d'accès (comme celui de collecte) est propice à se retrouver saturé, et qu'une différenciation de traitement des flots sera alors intéressante.

Self-Protect se propose comme une solution alternative visant à laisser le contrôle à l'utilisateur de la qualité de service de ses flots. Par utilisateur, on n'entend pas un contrôle systématiquement manuel, mais plutôt régulé par une politique configurable intégrée dans la passerelle domestique, et préemptible à tout moment.

8.1.3 Le débat sur la neutralité de l'Internet

Cette proposition est d'autant plus intéressante qu'un vaste débat a lieu actuellement concernant la neutralité de l'Internet.

- Mon article [ici](#)
- Compléter avec des remarques par rapport à celui de Sara (p2p etc)

8.1.4 Self-Protect dans l'évolution de l'architecture des réseaux

Si l'intégration des mécanismes Self-Protect dans le réseau d'accès nécessite quelques modifications au sein des équipements de réseau tels que les DSLAMs – il semble dur de convaincre les constructeurs –, elle n'en est pas moins réalisable dans le contexte à venir. L'architecture des réseaux est aujourd'hui presque complètement dictée (et surtout dans le cœur de réseau) par les grands constructeurs (Cisco, Juniper, etc.) qui s'appuient sur des solutions propriétaires tant sur les plans de contrôle que de données. Une démarche de modularisation et standardisation des éléments matériels et logiciels a pu démarrer, visant à s'affranchir de la dépendance aux constructeurs.

Advanced TCA est un gros effort de spécifications techniques publiées par le comité PICMG (PCI Industrial Computer Manufacturers Group) définissant les caractéristiques mécaniques, électriques, informatiques d'une architecture de sous-systèmes électroniques (cartes, châssis, etc.). Ce standard est soutenu par plus d'une centaine de sociétés, et est actuellement mis en œuvre dans des nombreux équipements, notamment dans les réseaux d'accès (certains DSLAMs ou nœuds reposant sur l'architecture 3GPP IMS par exemple). Il permet ainsi l'interconnexion dans le réseau voire au sein d'un même châssis des composants matériels et logiciels provenant de différentes sources.

Ces avancées matérielles ont également leurs équivalents dans le logiciel libre, tant sur les noyaux des systèmes d'exploitation (Carrier Grade Linux) qu'au niveau des éléments du plan de données (Click) ou que celui des composants de routage (XORP, FIRE, Quagga, etc.).

Si ces composants interopérables, ouverts et modulaires ne sont pas encore aussi performants que leurs équivalents propriétaires, ils sont d'ores-et-déjà une solution à considérer sérieusement pour les réseaux d'accès et de collecte, tant par leur coût plus réduit que les possibilités d'extensibilité qu'ils offrent. Leur déploiement serait notamment très favorable à l'intégration des mécanismes Self-Protect.

8.2 Etat de l'art des solutions de QoS existantes

8.2.1 Lien montant

Le lien montant est aujourd'hui le seul pour lequel le trafic de l'utilisateur est parfois ordonnancé. Le goulot d'étranglement étant situé entre la passerelle domestique et le DSLAM, il est nécessaire de contrôler les files d'attente directement sur la passerelle, donc que celle-ci le permette. Il reste cependant toujours possible de restreindre manuellement le débit montant sur un PC client, mais c'est à la fois coûteux et inefficace dans le cas d'un lien partagé.

Une des solutions adoptée consiste à faire confiance au marquage du champ ToS des paquets IP effectué par les applications, afin de répartir le trafic en 3 classes possédant une priorité stricte l'une sur l'autre. Au sein de chaque file, un ordonnancement de type *fair queueing* est réalisé par la discipline SFQ (*Stochastic Fair Queueing*) afin de garantir une équité approximative entre les flots.

8.2.2 Lien descendant

Pour des raisons historiques, le réseau d'accès est généralement bâti sur une architecture ATM. Les opérateurs utilisent ainsi les VC ATM afin de différencier les flots qu'ils possèdent au sein d'un VP utilisateur.

Free propose par exemple 4 VC ATM entre ses DSLAMs et la Freebox, sans réservation de bande passante par canal. Au niveau du DSLAM, il y a des files d'attente par niveau, classées par ordre de priorité (stricte) : RTP (téléphone), Multicast (vidéo), Management (MGCP téléphone, html de la télé, etc.) et enfin le trafic du PC client. Le RTP (canal audio d'une conversation téléphonique passe devant la télé car il est très sensible à la gigue malgré son faible débit (64kb/s)

Au delà de l'intérêt au niveau de la gestion de la qualité de service, ces VC permettent une isolation du trafic qui permettra d'en rediriger très simplement une partie au niveau matériel vers des composants adaptés à son traitement (par exemple les flux télévisuels seront dirigés directement vers un chipset de décodage sans besoin de faire intervenir le système d'exploitation présent sur la passerelle.

De nombreuses solutions commerciales existent afin de permettre une différenciation du trafic descendant, principalement à destination des offres professionnelles. Citons par exemple des appliances telles que Ellacoya ou Packeteer, réalisant des opérations très poussées (analyse de niveau 7, analyse stateful, etc.) et qui requièrent une forte configuration adaptée au profil de l'entreprise.

Très peu de propositions cependant au niveau de la protection du trafic internet de l'utilisateur. Quelques projets libres offrent un moyen de compenser ce manque en utilisant le comportement du protocole TCP. L'idée est par exemple de rejeter certains paquets sur le lien descendant, ou de retarder les acquittements montants afin de simuler un lien de capacité plus faible ; en maintenant ainsi une charge peu élevée, les files d'attente au niveau du DSLAM se retrouvent pratiquement vides ce qui assure une qualité acceptable pour les flux sensibles. Le prix à payer est une sous utilisation des ressources, avec un mécanisme complexe et peu viable car dépendant fortement des protocoles mis en œuvre.

La récente augmentation liée au trafic peer-to-peer souvent jugé non prioritaire a mené l'apparition de solution DPI (*Deep Packet Inspection*), afin de reconnaître les flots au travers d'une analyse poussée des paquets (jusqu'au niveau applicatif), voire en fonction du comportement statistiques des flots (taille des flots, etc.). Couplés à Diffserv par exemple, les mécanismes DPI permettent d'associer un PHB à un flot en fonction de règles prédéfinies.

Si cette approche peut paraître contestable au vu de certaines applications temps-réel circulant en peer-to-peer, ou parce que ce trafic est le choix de l'utilisateur, elle est proposée comme une solution permettant de retarder les investissements visant à augmenter la capacité des liens afin de garantir leur transparence (assurer la qualité des flots en maintenant un niveau de charge faible). Il est cependant possible d'offrir de meilleures solutions visant à protéger uniquement les flots pour lequel cela est nécessaire, car les mécanismes DPI, qui ne sont déjà pas complètement efficaces, poussent à une fuite en avant avec la généralisation de nouveaux protocoles décentralisée, utilisant des ports dynamiques, du chiffrement, de la stéganographie, etc.

Citons enfin quelques essais à la limite de la légalité de la part d'opérateurs tels que Comcast de s'immiscer au sein du trafic de l'utilisateur en interrompant volontairement les communications TCP indésirables en forgeant des paquets RST (à la manière du grand firewall de Chine).

8.3 Présentation de l'architecture Self-Protect

8.3.1 Présentation

8.3.2 Configuration dans lesquelles Self-Protect peut être appliqué

8.3.3 Fonctionnement global

8.3.4 Ordonnancement des flots

Priorités

Partage équitable

Flots élastiques

Ordonnancement

8.3.5 Interaction de l'utilisateur

8.3.6

Both the DSLAM and the

8.3.7 Signalisation

8.3.8 Politique d'attribution des priorités

8.3.9 Self-Protect sur le poste client

8.3.10 Equivalent Self-Protect avec une solution de DPI

8.4 Réalisation d'un prototype de démonstration

En fonction des besoins de l'utilisateur, la passerelle domestique peut être configurée de deux manières :

Le mode transparent correspond à une passerelle configurée en bridge ne faisant que relayer les paquets (après traitement éventuel) vers la machine de l'utilisateur possédant l'adresse IP publique (ou plusieurs machines dans le cas de plusieurs IP).

Le mode routeur permet d'avoir un réseau privé de plusieurs machines. Il est cependant parfois délaissé en raison du manque de configuration possible sur la passerelle (confiance dans le firmware, mises à jour, possibilités de configurer le firewall et la QoS, etc.).

Afin de simplifier le prototype, nous nous contentons de réaliser le premier mode de fonctionnement. L'ajout des fonctionnalités relatives au mode routeur sera l'objet de travaux ultérieurs (gestion du NAT, etc.)

8.4.1 Composants matériels et logiciels

Emulation d'un réseau d'accès asymétrique ADSL

Tous les éléments du réseau d'accès émulé sont des PC sous GNU/Linux possédant plusieurs interfaces réseau bridgées, et interconnectés par les liens Ethernet à 100Mb/s. Les limitations en débits émulant le lien asymétrique ADSL sont réalisées différemment

8.4 Réalisation d'un prototype de démonstration

en fonction du débit souhaité. Dans le sens descendant, il est possible de forcer l'autonégociation des cartes réseau à 10Mb/s. Sur le sens montant, le trafic est policé grâce à la discipline de service HTB (Hierarchical Token Bucket) du framework *traffic control*, placée en sortie de la file d'attente de l'interface réseau.

Choix des composants logiciels

Le système GNU/Linux dispose directement de l'ensemble des composants nécessaires à la réalisation d'une maquette de l'architecture Self-Protect. Il s'agit notamment des frameworks Netfilter et Traffic Control, qui contrôlent respectivement le trajet des paquets au sein de la couche réseau du système d'exploitation et leur ordonnancement au niveau des files d'attente.

Notre choix s'est porté sur le contrôle des ressources par des démons fonctionnant dans l'espace utilisateur plutôt que dans le noyau afin d'aboutir plus rapidement à un prototype fonctionnel et portable, vu que les temps de latences ne sont pas ici critiques et que les API mises à notre disposition permettent d'interagir convenablement avec les ressources du noyau. Cette communication aurait de toute façon du avoir lieu puisque nous envisageons de permettre à l'utilisateur de visualiser et paramétrer la performance actuelle des flots dans le système (par une interface web dans notre prototype). Nous expliquerons toutefois dans la suite comment pourrait se faire l'intégration de ces mécanismes au sein du noyau.

8.4.2 Tables de flots

La maintenance d'une table des flots sera une chose aisée puisqu'elle existe déjà en vue de la réalisation du suivi de connexions. Cette opération est nécessaire en effet afin d'obtenir un firewall plus robuste, dépendant de l'état des connexions. Chaque connexion correspond notamment à un couple (origine, réponse) caractérisant les sockets à chaque bout. Cette notion correspond exactement à notre notion de flot, c'est-à-dire le quintuplet des adresses IP et des ports source et destination adjoint du protocole de niveau 4 (généralement TCP ou UDP).

Le système se charge de la création de ces entrées, ainsi que de leur destruction après une certaine durée d'inactivité. Cette dernière dépend de plusieurs facteurs, tels que le protocole ou son état (similaire aux états de TCP). Elle est modifiable au travers de l'interface `/proc`. Dans le noyau, les différentes valeurs sont apportées par des extensions modulaires pour les protocoles de niveau 3 et 4, qu'il serait possible de ne pas prendre en compte, afin de n'avoir qu'une valeur générique à modifier. Nous les changerons toutes sans toucher au noyau.

La manipulation des flots se fera par l'intermédiaire d'une table présente dans les démons en espace utilisateur, qui seront chargés de la synchroniser avec la table noyau. Une priorité sera affectée aux flots par l'attribution d'une marque interne à l'entrée `conntrack`. Il sera de plus possible d'ajouter des attributs à une entrée `conntrack` de l'espace utilisateur sans avoir à modifier la structure au niveau du noyau.

Au sein de la passerelle domestique

Le rôle du démon est de :

- Maintenir la table synchronisée avec le noyau
- Détecter les nouveaux flots et leur attribuer un profil (voir plus loin)
- Signaler le profil au DSLAM
- Maintenir une interaction avec l'utilisateur

Au sein du DSLAM

Le DSLAM ne réalise aucune analyse des flots qu'il véhicule. Le rôle du démon sera uniquement de recevoir la signalisation de la part de la passerelle domestique, et de remonter après vérification le profil au niveau du noyau (ou de détruire le(s) flot(s) de la table).

A des fins de cohérence, le timeout sera supposé égal sur le DSLAM et sur la passerelle domestique.

Spécifications du protocole de signalisation

- signaler un flot
- acquittements
- assurer la transmission
- utiliser des protocoles standards

Afin que la signalisation n'interfère pas avec la vision qu'a l'utilisateur de son trafic, nous l'exclurons du système de suivi de connexions. Cela est possible en marquant explicitement (NO_TRACK) les paquets en question lors de leur arrivée sur la machine (table RAW, qui est la première rencontrée).

Nous avons choisi d'utiliser le protocole XMLRPC, basé sur HTTP, et permettant l'appel de fonction distantes. La connexion fiable est alors assurée par TCP.

8.4.3 Ordonnancement des flots

Description

Contrairement à un lien de cœur de réseau opéré généralement dans un régime transparent, il est souhaité que le lien d'accès puisse être saturé au vu des capacités limitées. Cependant, la distinction d'un grand nombre de niveaux de différenciation sera également inutile, car elle n'entraîne pas de différence significative dans des conditions normales d'utilisation, mais seulement en surcharge. Dans ce dernier cas, nous nous contentons de protéger les flots plus prioritaires.

Dans cette première version du prototype, nous nous proposons d'évaluer un ordonnancement simple du trafic basé sur des files strictement prioritaires :

La priorité haute convient aux flots streaming ayant de fortes contraintes d'interactivité.

Ces flux sont typiquement ceux produits par des services de VoIP, visioconférence ou jeux en réseau. La contrainte la plus forte est celle du délai dont la valeur et la gigue doivent être les plus faibles possibles. La charge de cet agrégat de flots sera généralement plus faible que la capacité du lien d'accès.

La priorité moyenne convient aux flots streaming non-interactifs, typiquement des services de VoD ou de TV, qui peuvent bénéficier d'un buffer chez l'utilisateur. On notera d'ailleurs la quasi-disparition de vrais flots streaming de la part des fournisseurs de contenus, au profit du simple téléchargement des fichiers, depuis la montée des débits. On pourra mettre dans cette classe les flots élastiques que l'utilisateur voudra prioriser (la eb par exemple).

La priorité faible convient au reste des flots, c'est-à-dire la plupart des flots élastiques (transferts FTP, P2P, email, etc.), qui seront traités en *Best Effort*.

Cette répartition est celle assurée par défaut par le mécanisme Self-Protect, et pourra être préemptée par l'utilisateur. Il est cependant nécessaire de s'assurer de certaines conditions de viabilité de la qualité de service quant aux choix de l'utilisateur : la mise en priorité haute de tout le trafic par exemple rend le mécanisme inefficace. Une fonction

8.4 Réalisation d'un prototype de démonstration

d'arbitrage entre plusieurs utilisateurs au niveau de la passerelle domestique reste également à définir.

En raison du caractère *bursty* des flots élastiques, nous ajouterons un traitement *fair queueing* aux files de priorité moyenne et faible, afin d'assurer une équité entre les flots. Nous utiliserons l'algorithme SFQ puisqu'il est disponible d'origine dans le noyau Linux et que ses performances sont suffisantes dans notre cas.

Une différenciation implicite en deux classes streaming et élastique, comme nous le faisons pour Cross-Protect, n'est pas suffisante ici puisque des flots streaming tels que la visioconférence peuvent avoir des débits crête très élevés comparés à la capacité du lien.

Réalisation de l'ordonnancement

Outre les trois files prioritaires correspondant aux trois niveaux de priorité présentés, nous en ajouterons une supplémentaire, plus prioritaire, qui recevra le trafic de signalisation, dont le volume est supposé négligeable.

- Configuration de TC, TBF/prio/SFQ

Par convention, les marques 1, 2 et 3 seront les 3 niveaux de priorité, 1 étant la plus haute ; 5 correspondra à une connexion bannie (dont les paquets seront droppés, fonctionnalité à rapprocher d'un firewall) ; la marque 6 sera attribuée aux paquets de flots traqués mais non identifiés, et qui recevront une priorité par défaut. Ainsi, les seuls paquets restant, dont les flots ne sont pas traqués c'est-à-dire la signalisation, posséderont la marque 0.

Il sera possible d'envisager de réserver une certaine proportion de bande passante pour les nouveaux flots non encore identifiés.

L'étape d'ordonnancement a lieu lorsque le paquet est mis en attente dans les files de l'interface de sortie, c'est-à-dire après sa traversée dans le framework netfilter. Afin d'attribuer une file à un paquet, il est possible de réaliser un filtre qui se basera sur la marque de l'entrée conntrack associée au paquet (un champ dans le *socket buffer*). Nous avons choisi de nous conformer aux usages en marquant en interne les paquets entrants avec la marque de l'entrée conntrack associée (grâce à la table MANGLE), et d'utiliser alors un filtre existant dans tc pour l'attribution de la file prioritaire.

Classement des flots élastique

Il est possible de classer les flots élastiques à la manière de l'algorithme LAS (*Least Attained Service*), en utilisant les fonctionnalités de comptage du trafic de chaque flot (entrée conntrack). Cela consiste à prioriser les flots courts par rapport aux flots longs. Succinctement ce mécanisme peut être expliqué grâce au schéma de la figure ???. Sur ce schéma, un flot est représenté par un rectangle donc la hauteur représente le débit, et la longueur la durée : la priorisation d'un flot court par rapport à un flot long permet d'avoir un temps de transfert moins important pour le flot court, dans l'impacter le flot long. Il convient alors d'estimer correctement le seuil de différenciation. Cette proposition est d'autant plus valable que les flots TCP se comportent comme du trafic streaming (flot non-*bursty* à faible débit) pour des petites tailles de la fenêtre de congestion.

Dans le cadre de Self-Protect, une solution serait d'associer par défaut la priorité moyenne aux flots courts, et la priorité la plus faible aux flots longs.

8.4.4 Détermination des priorités

cf rapport Elie

8.4.5 Vision utilisateur de la table des flots

De manière générale, l'utilisateur doit avoir une vision de la table des flots et notamment la priorité attribuée à chacun d'entre eux. Vu le grand nombre de flots qui seront présents dans le système, et afin de permettre un meilleur contrôle de l'attribution des priorités, le système procédera à une identification relativement poussée des flots.

Nous définissons pour cela des filtres, qui seront appliqués généralement au premier paquet de chaque flot afin de l'identifier. L'apport ultérieur d'informations supplémentaires, que ce soit (comme nous le verrons plus tard) par l'application elle-même ou par le client, entraînera une nouvelle analyse du flot (certainement sur un sous ensemble de filtres), afin d'affiner la reconnaissance.

Chaque filtre sera caractérisé par :

- le quintuplet classique pour les paquets IPv4 (addresses IP et ports source et destination, accompagnées du protocole de niveau 4) qui correspond exactement à la notion de socket pour le système. Il est possible de spécifier à la fois une adresse ip numérique (au format xxx.xxx.xxx.xxx) voire un nom résolu décrit par un certain motif, exprimé au moyen d'une expression régulière. Il conviendra d'adapter cette partie pour gérer l'adressage IPv6 ;
- le volume de trafic transmis par le flot, qui permet la différenciation des flots à la manière de l'algorithme LAS présenté ci-dessus ;
- le champ ToS indiqué par les paquets dans le sens montant (supposés de confiance), qui permet de donner une information supplémentaire sur la nature des flots ;
- le protocole de niveau 7, qui peut être obtenu par une analyse applicative des premiers paquets reçus de chaque flot, et qui permet l'identification de trafic non chiffré sur un port inconnu, exploitant un port connu qui ne lui est pas explicitement attribué, ou encore lorsque plusieurs types de trafic peuvent être véhiculés par un même protocole ;
- le nom de l'application associé à la socket (généralement unique pour des sockets INET, sinon le premier), ainsi que l'utilisateur la possédant, lorsqu'ils auront été signalés par le PC client. Notons que la signalisation envoie les noms plutôt que prendre la décision de signaler un changement de priorité afin de pouvoir centraliser l'ensemble des décisions dans la liste de filtres ;
- éventuellement il sera possible de réagir à un certain contenu de niveau 7 (par exemple pour l'analyse des en-têtes de certains protocoles), dont le fonctionnement est similaire. La différence ici est qu'une même connexion peut véhiculer plusieurs échanges (par exemple HTTP v1.1). Ainsi, d'une part les seuls premiers paquets ne suffiront pas (on veut identifier le champ Content-Type de tous les échanges HTTP) et d'autre part la probabilité de trouver la chaîne recherchée fragmentée sur plusieurs paquets sera plus importante (de même que si la taille des paquets est relativement petite). Cette fonctionnalité n'est pas implémentée dans un premier temps.

Chacun de ces filtres permet d'associer un flot à une *meta-application*, c'est-à-dire un type d'usage pouvant correspondre à plusieurs filtres. Nous citerons pour exemple la meta-application "Navigation Internet" qui peut correspondre aux filtres sur les numéros de ports 80 ou 443. C'est au travers de la méta-application qu'un flot se verra associé une description et une priorité initiale.

Il est possible de laisser un ou plusieurs champ nul afin de ne pas le prendre en compte lors de la comparaison. Pour cette raison et parce que certains filtres seront plus précis que d'autres, l'ordre utilisé pour la comparaison de ces filtres avec un flot affectera la décision finale d'association avec une meta-application.

D'ailleurs à des fins de simplicité, nous définissons dans tous les cas une méta-application "Flots non identifiés" qui permettra d'associer une marque par défaut aux flots qui n'ont été reconnus par aucun des filtres, et dont le filtre unique nul devra être comparé en

8.4 Réalisation d'un prototype de démonstration

dernier lieu.

La relation d'ordre entre les filtres n'étant pas totale, nous devons mettre en place une heuristique permettant de les classer, afin d'éviter que l'utilisateur ne doive le faire (trop grand nombre), ou leur affecter des poids déterminant leur ordre comme c'est souvent le cas.

- Décrire l'heuristique

Dans certains cas, une information complète sur le flot ne sera pas disponible : la résolution des adresses IP qui ne sont pas présentes dans un cache se fera de manière asynchrone ; les analyses applicatives des paquets peuvent n'identifier le protocole qu'après un certain nombre de paquets ; etc. Dans ce cas, l'analyse sera effectuée à nouveau lorsque de l'information supplémentaire sera disponible. Il sera alors possible de ne parcourir que les filtres possédant un tel champ non nul. L'utilisateur ayant à tout moment la possibilité de changer la priorité d'un flot, il conviendra de ne plus la modifier alors même si de nouvelles informations nous parviennent. Nous maintenons donc un état par flot permettant de savoir si la priorité a été affectée, et si oui par le programme ou pas l'utilisateur.

Le fonctionnement de la table *conntrack* ne permet pas de distinguer dans tous les cas le sens dans lequel la connexion a été établie (en cas de purge par exemple). Le flot inscrit dans le sens *original* sera le premier affecté, et celui dans *reply* la version inversée. Un flag sera mis lorsque la connexion aura été vue dans les deux sens. Il est toutefois possible de raffiner l'analyse lorsque l'on connaît les adresses IP utilisées dans le réseau domestique (adresses privées en mode routeur, adresse publique unique sinon), et en la mettant en rapport avec le port associé (dynamique dans le cas d'un client, fixé et connu dans le cas d'un serveur). Cette fonctionnalité n'est pas implémentée dans la version actuelle.

Dans notre architecture, la marque sera attribuée à une entrée *conntrack* qui regroupera les flots bidirectionnels. Il ne sera donc généralement pas possible de donner des marques différentes à chaque sens de la connexion. Une exception cependant pour les paquets ACK.

- Cas de wondershaper à détailler
- Quelle priorité leur donner ?
- Comment refléter cela dans l'interface ?

8.4.6 Cas des tunnels, socks, VPN, et trafic chiffré

- Position du socks (hors du réseau domestique ? en local ?)
- cf cas des entreprises
- Cas youtube pour filtre sur l'url
- Mettre en place un proxy applicatif transparent ?
- Quid du nat également ?

8.4.7 Statistiques

8.4.8 Arbitrage et contrôle d'admission en cas de surcharge

8.4.9 Regroupement des flots en macro-flots

- Norme sur la segmentation du champ flow-id
- Attribution d'un flowid aux flots en fonction de leur identification
- Utilisation dans un algorithme de fair queueing ? quid de la modification du hash ?

Filtrage applicatif

Filtrage applicatif dans le framework netfilter

Il est possible de réaliser du filtrage de niveau 7 dans le framework netfilter au moyen du module `l7filter`, disponible sous la forme d'un patch noyau. Le contenu des premiers paquets est alors analysé à l'aide d'expressions régulières afin de détecter le protocole sous-jacent. D'autres projets permettent de réaliser ce filtrage (IPP2P, HiPPiE notamment), nous ne les considérons pas dans la première implémentation de notre prototype.

Spécificités du module `l7filter`

L'intégration de la reconnaissance des protocoles au niveau 7 se fait au travers de l'outil `iptables`. Puisque nous souhaitons effectuer l'attribution des priorités en fonction des filtres définis dans notre démon, la démarche est d'installer au travers de l'outil `iptables` 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.

Deux marques sont utilisées pour identifier les paquets qui appartiennent à des flots dont le protocole applicatif n'a pas encore été reconnu (*unset*), ou dont la reconnaissance a été interrompue (*unknown*). L'identification s'arrête en effet dès que 10 paquets ou 2ko de données sont dépassés. Il est possible de modifier ces valeurs dans `/proc/net/layer7_numpackets`, ou grâce à la commande `modprobe xt_layer7 maxdatalen=N`, avec N en ko. L'ancien nom du module était `ipt_layer7`. Enfin une limite en dur de 64ko est présente dans le fichier source du module.

Lors de la modification des expressions régulières de reconnaissance des protocoles, il est nécessaire de supprimer toutes les règles modifiées et de les réinsérer puisqu'elles ne sont lues que lors de leur passage à `iptables`.

Filtrage étendu grâce au client Self-Protect

8.4.10 Extensions possibles du prototype

Commentaires sur notre proto

`libnetfilter_conntrack` ruby pkoï ne pas réutiliser `pyctd` / `conntrackd` algorithme de synchro de la table avec le noyau... raisons techniques persistance de la configuration configuration

Mode apprentissage

Log des flots inconnus, ou dont la priorité a été modifiée pour demander à l'utilisateur s'il veut forger une règle...

Considération des aspects sécurité

- Vulnérabilités introduites par une signalisation : équipement tel un DPI qui fait le relai ?
- Minimum de traitement au niveau du DSLAM
- Protection de la signalisation : empêcher l'utilisateur de définir des règles nocives ?
- On modifie dynamiquement les règles `iptables`, on utilise le champ `mark`... quid ?
- Authentification : envoi de trames par une connexion wifi : par besoin d'ack, spoof d'ip etc...

8.5 Résultats des expérimentations

- Flood de signalisation

Notons que la possibilité d'associer une priorité à un flot en fonction de l'application qui possède la socket correspondante n'est pas critique en termes de sécurité. La situation est différente de celle d'un firewall applicatif qui nécessite de garantir que le programme chargé en mémoire (qui s'exécute) correspond bien à celui stocké sur disque auquel on a donné l'autorisation. Il est en effet possible sous certaines conditions d'injecter du code dans l'espace mémoire d'un autre processus afin de bénéficier de ses autorisations. Ici la seule implication est la priorité donnée au trafic.

8.5 Résultats des expérimentations

8.5.1 Mise en évidence du besoin de QoS

8.5.2 Insuffisances du Fair Queueing seul dans le réseau d'accès

8.6 Conclusion

Coupler l'utilisation au firewall... aspects de sécurité...

8.7 Trucs en vracs à ne pas mettre dans le rapport

Chapitre 9

Conclusion

contributions
perspectives

Bibliographie

- [1] *Measurement-based Admission Control for Elastic Traffic*, Salvador, Brazil, December 2001. Elsevier.
- [2] A.Dhamdhere and C. Dovrolis. Open issues in router buffer sizing. *ACM SIGCOMM Computer Communications Review (editorial section)*, January 2006.
- [3] A.Dhamdhere, C. Dovrolis, and H. Jiang. Buffer sizing for congested internet links. *Proceedings of IEEE INFOCOM, Miami FL*, March 2005.
- [4] M. Allman and E. Blanton. Notes on burst mitigation for transport protocols. *ACM SIGCOMM Computer Communication Review*, 35(2) :60, 2005.
- [5] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. *Proceeding of ACM SIGCOMM '04, Portland, Oregon*.
- [6] K. Avrachenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The effect of router buffer size on the tcp performance. In *In Proceedings of the LONIIS Workshop on Telecommunication Networks and Teletraffic Theory*. Citeseer, 2001.
- [7] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows : Predictability of the response time. In *IEEE INFOCOM*, volume 2, pages 762–773. Citeseer, 2004.
- [8] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling : Investigating unfairness. In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, page 290. ACM, 2001.
- [9] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J. Roberts. Statistical bandwidth sharing : a study of congestion at flow level. *SIGCOMM 2001, San Diego, CA, USA*.
- [10] N. Benameur, S. Ben Fredj, S. Oueslati-Boulahia, and J. Roberts. Quality of service and flow level admission control in the Internet* 1. *Computer Networks*, 40(1) :57–71, 2002.
- [11] N. Benameur, A. Kortebi, S. Oueslati, and J. Roberts. Selective service protection in overload ; differentiated services or per-flow admission control ? In *Networks 2004 : 11th International Telecommunications Network Strategy and Planning Symposium : June 13-16, 2004, Vienna, Austria*, page 217. Margret Schneider, 2004.
- [12] D. Bertsekas and R. Gallager. *Data networks*. Prentice-hall Englewood Cliffs, NJ, 1987. - 1992 in google scholar !
- [13] T. Bonald. Throughput performance in networks with linear capacity constraints. *Proceedings of CISS 2006*.

- [14] T. Bonald, M. Jonckheere, and A. Proutiere. Insensitive load balancing. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 367–377. ACM, 2004.
- [15] T. Bonald and L. Massoulié. Impact of fairness on Internet performance. In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, page 91. ACM, 2001.
- [16] T. Bonald and A. Proutière. Insensitivity in processor-sharing networks. *Proceedings of Performance*, 2002.
- [17] T. Bonald and A. Proutiere. Insensitive bandwidth sharing in data networks. *Queueing systems*, 44(1) :69–100, 2003.
- [18] T. Bonald and A. Proutiere. On performance bounds for balanced fairness. *Performance Evaluation*, 55 :25–50, 2004.
- [19] T. Bonald, A. Proutière, and J. Roberts. Statistical performance guarantees for streaming flows using expedited forwarding. In *IEEE INFOCOM*, volume 2, pages 1104–1112. Citeseer, 2001.
- [20] T. Bonald and J. Roberts. Scheduling network traffic. *ACM SIGMETRICS Performance Evaluation Review*, 34(4) :35, 2007.
- [21] J. Boyer, F. Guillemin, P. Robert, and B. Zwart. Heavy tailed M/G/1-PS queues with impatience and admission control in packet networks. In *IEEE INFOCOM*, volume 1, pages 186–195. Citeseer, 2003.
- [22] L. Brakmo. TCP Vegas : End to end congestion avoidance on a global Internet. *IEEE Journal on selected Areas in communications*, 13(8) :1465, 1995.
- [23] B. Briscoe. Flow rate fairness : Dismantling a religion. *ACM SIGCOMM Computer Communication Review*, 37(2) :63–74, Apr. 2007.
- [24] B. Briscoe, P. Eardley, D. Songhurst, F. Le Faucheur, A. Charny, V. Liatsos, J. Babiarz, K. Chan, S. Dudley, G. Karagiannis, et al. An edge-to-edge deployment model for pre-congestion notification : Admission control over a DiffServ region. 2006.
- [25] B. Briscoe, P. Eardley, D. Songhurst, F. Le Faucheur, A. Charny, V. Liatsos, J. Babiarz, K. Chan, S. Dudley, G. Karagiannis, et al. Pre-congestion notification marking. *IETF Internet draft*, (draft-briscoe-tsvwg-cl-phb-03), 2006.
- [26] A. Charny, F. Faucheur, V. Liatsos, and J. Zhang. Pre-congestion notification using single marking for admission and pre-emption. *draft-charny-pcn-single-marking-00 (work in progress)*, 2006.
- [27] F. Delcoigne, A. Proutiere, and G. Régnié. Modeling integration of streaming and data traffic. *Performance Evaluation*, 55 :185–209, 2004.
- [28] J. Domzal, R. Wojcik, and A. Jajszczyk. Qos-aware net neutrality. pages 147 –152, aug. 2009.
- [29] N. Dukkupati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor sharing flows in the internet. *Quality of Service-IWQoS 2005*, pages 271–285, 2005.
- [30] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part iii : routers with very small buffers. *ACM SIGCOMM Computer Communication Review*, v.35 n.3.

BIBLIOGRAPHIE

- [31] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Routers with very small buffers. *Proceedings of the IEEE INFOCOM'06, Barcelona, Spain*.
- [32] W. Feng, K. Shin, D. Kandlur, and D. Saha. The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking (TON)*, 10(4) :528, 2002.
- [33] S. Floyd. Highspeed tcp for large congestion windows. *RFC 3649, Experimental, December 2003*.
- [34] S. Floyd. Explicit Congestion Notification. In *ACM Computer Communication Review*, volume 24, pages 10–23, October 1994.
- [35] S. Floyd and K. Fall. Router mechanisms to support end-to-end congestion control, 1997.
- [36] S. Floyd, R. Gummadi, S. Shenker, et al. Adaptive RED : An algorithm for increasing the robustness of RED's active queue management. *Preprint, available at <http://www.icir.org/floyd/papers.html>*, 2001.
- [37] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. 1 (4) : 397–413, 1993.
- [38] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Trans, on Networking*, 3(4) :365–386, August 1995.
- [39] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems (TOCS)*, 21(2) :207–233, 2003.
- [40] C. Holot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1726–1734. IEEE, 2001. PI - Proportional integral.
- [41] V. Jacobson and R. Braden. Congestion control and avoidance. In *Proc. ACM SigComm*, volume 88, pages 314–329, 1988.
- [42] H. Jiang and C. Dovrolis. Why is the internet traffic bursty in short time scales ?
- [43] C. Jin, D. Wei, and S. Low. Fast tcp : Motivation, architecture, algorithms, performance. *Proceedings of IEEE Infocom, Hong Kong*, 14(6) :1259, 2006.
- [44] A. Kamal. A discrete-time model of tcp reno with background traffic interference. In *mascons*, page 0445. Published by the IEEE Computer Society, 2002.
- [45] F. Kelly. Models for a self-managed Internet. *Philosophical Transactions : Mathematical, Physical and Engineering Sciences*, pages 2335–2348, 2000.
- [46] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks : shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3) :237–252, 1998.
- [47] T. Kelly. Scalable tcp : Improving performance in highspeed wide area networks.
- [48] S. Keshav. Congestion control in computer networks. *PhD Thesis, published as UC Berkeley TR-654*.
- [49] P. Key, L. Massoulié, A. Bain, and F. Kelly. A network flow model for mixtures of file transfers and streaming traffic. In *Proceedings of ITC*, volume 18, 2003.

- [50] P. Key, L. Massoulié, A. Bain, and F. Kelly. Fair Internet traffic integration : network flow models and analysis. *Annals of Telecommunications*, 59(11) :1338–1352, 2004. notes.
- [51] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. *ACM SIGMETRICS Performance Evaluation Review*, 33(1) :228, 2005.
- [52] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. *Sigmetrics'05, Banff, Canada*, June 2005.
- [53] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. On the scalability of fair queueing. *ACM HotNets-III, San Diego, USA*, November 2004.
- [54] A. Kortebi, S. Oueslati, and J. Roberts. Cross-protect : implicit service differentiation and admission control. *IEEE HPSR 2004, Phoenix, USA*.
- [55] A. Kortebi, S. Oueslati, and J. Roberts. Implicit service differentiation using deficit round robin. *Proceedings of ITC 19, Beijing*, 2005.
- [56] T. Lan, D. Kao, M. Chiang, and A. Sabharwal. An axiomatic theory of fairness in network resource allocation. pages 1 –9, mar. 2010.
- [57] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking (ToN)*, 2(1) :15, 1994.
- [58] Y.-T. Li, D. Leith, and R. Shorten. Experimental evaluation of tcp protocols for high-speed networks. *Hamilton Institute, NUI Maynooth*, 2005.
- [59] D. Lin and R. Morris. Dynamics of Early Random Detection. In *ACM SIGCOMM*, volume 97, 1997.
- [60] J. Liu, A. Proutière, Y. Yi, M. Chiang, and H. Poor. Flow-level stability of data networks with non-convex and time-varying rate regions. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, page 250. ACM, 2007.
- [61] A. Medina, M. Allman, and S. Floyd. Measuring the evolution of transport protocols in the Internet. *ACM SIGCOMM Computer Communication Review*, 35(2) :52, 2005.
- [62] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking (ToN)*, 8(5) :556–567, 2000.
- [63] S. Nunez-Queija, H. vd Berg, and M. Mandjes. Strategies for integration of elastic and stream traffic : a performance evaluation. In *Proc. of the 16th International Teletraffic Congress (ITC 16)*.
- [64] T. Ott, T. Lakshman, and L. Wong. Sred : stabilized red. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1346–1355. IEEE, 1999.
- [65] S. Oueslati and J. Roberts. A new direction for quality of service : Flow aware networking. *NGI 2005, Rome*.
- [66] S. Oueslati, J. Roberts, and F. R&D. Comparing flow-aware and flow-oblivious adaptive routing. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 655–660, 2006.

BIBLIOGRAPHIE

- [67] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput : A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, page 314. ACM, 1998.
- [68] R. Pan, B. Prabhakar, and K. Psounis. CHOKe-A stateless queue management scheme for approximating fair bandwidth allocation. In *Proceedings of IEEE Infocom*, pages 942–951, 2000.
- [69] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. *Computer*, 1996.
- [70] V. Paxson and S. Floyd. Wide area traffic : the failure of Poisson modeling. *IEEE/ACM Transactions on Networking (TON)*, 3(3) :244, 1995.
- [71] V. Paxson and S. Floyd. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4) :392–403, 2001.
- [72] G. Raina, D. Towsley, and D. Wischik. Part ii : control theory for buffer sizing. *ACM SIGCOMM Computer Communication Review*, v.35 n.3, July 2005.
- [73] G. Raina and D. Wischik. Buffer sizes for large multiplexers : Tcp queueing theory and instability analysis. *NGI'05, Rome*, April 2005.
- [74] J. Roberts. Quality of service guarantees and charging in multiservice networks, to appear in *IEICE Trans. Communications*, 1998.
- [75] J. Roberts. Realizing quality of service guarantees in multiservice networks. In *Proceedings of IFIP Seminar PMCCN*, volume 97, 1998.
- [76] J. Roberts. A survey on statistical bandwidth sharing. *Computer Networks*, 45(3) :319–332, 2004.
- [77] J. Roberts and L. Massoulié. Bandwidth sharing and admission control for elastic traffic. *Yokohama*, October 1998.
- [78] J. Roberts and L. Massoulié. Arguments in favour of admission control for TCP flows. In *Proceedings of the 1999 16th ITC*, 1999.
- [79] J. Roberts and L. Massoulié. Bandwidth sharing : objectives and algorithms. In *INFOCOM'99*, 1999.
- [80] J. Roberts, U. Mocci, and J. Virtamo. Methods for the performance evaluation and design of broadband multiservice networks. *Published by the Commission of the European Communities, Information Technologies and Sciences, COST*, 242, 1996.
- [81] S. Scott. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communications*, 13(7) :1176–1188, 1995.
- [82] K. Srisankar and 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*. Association for Computing Machinery, Inc, One Astor Plaza, 1515 Broadway, New York, NY, 10036-5701, USA,, 2001.
- [83] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury. Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queuing. *IEEE Journal in Selected Areas in Communications*, Aug. 1999.

- [84] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury. Design considerations for supporting tcp with per-flow queuing. *Proc. of INFOCOMM '98*, Apr. 1998.
- [85] C. Villamizar and C. Song. High performance tcp in ansnet. *ACM Computer Communications Review*, 24(5) :45-60, October 1994.
- [86] D. Wischik. Buffer requirements for high-speed routers. *Optical Communication, 2005. ECOC 2005. 31st European Conference on*, 5 :23– 26, 2005.
- [87] D. Wischik and N. McKeown. Part i : buffer sizes for core router. *ACM SIGCOMM Computer Communication Review*, v.35 n.3, July 2005.

List of Acronyms

BIBLIOGRAPHIE

Glossaire

RTT!