THÈSE D'HABILITATION À DIRIGER DES RECHERCHES

présentée par

Guillaume URVOY-KELLER

DE LA QUALITE DE SERVICE

À L'ANALYSE DE TRAFIC

Soutenue le 11 décembre 2006, devant le jury composé de :

Rapporteurs	André-Luc Beylot	Professeur à l'ENSEEIHT
	Andrzej Duda	Professeur à l'ENSIMAG
	Gérard Hébuterne	Professeur à l'INT
Directeur de thèse	Ernst Biersack	Professeur à Eurecom
Examinateurs	Walid Dabbous	Directeur de Recherche INRIA
	Tijani Chahed	Maître de Conférences à l'INT

Table des matières

Chapitre 1

CURRICULUM VITÆ

Date de naissance : 04/10/1971
Nationalité : française
Addresse Actuelle :
934, chemin des âmes du purgatoire, Bât. E
06600, Antibes
e-mail : urvoy@eurecom.fr
Profession : Maître de Conférences, 1ère classe, Institut Eurecom

Thèmes de recherche :
- Mesures dans les réseaux ;
- Réseaux pair-à -pair ;
- Qualité de service dans l'Internet.

1.1 Formation

1999 : Doctorat de l'Université Paris 6

1995 : Ingénieur diplômé de l'Institut National des Télécommunications

1995 : DEA MISI de l'Université de Versailles

1994 : Licence de mathématiques pures de l'Université Paris 6

1989-1992 : Classes préparatoires supérieures et spéciales

1.2 Expérience Professionnelle

Depuis 2002 : Maître de Conférences à l'Institut Eurecom

2000-2002 : Assistant d'Enseignement et de Recherche à l'Institut Eurecom

1999-2000 : ATER (1/2 poste) à l'Université de Versailles

1996-1999 : Thèse de doctorat - bourse Cifre Philips

1995-1996 : Scientifique du Contingent

1.3 Encadrements

Thèses de Doctorat

– Idris A. Rai (co-encadrement avec Ernst Biersack) - "Qualité de services dans les serveurs de bordure" (2001- 2004)
– Matti Siekkinen (co-encadrement avec Ernst Biersack) - "Métrologie des connexions TCP" (2003-2006)
– Diego Ferrero - "Réseaux sans-fil maillés" (2006-)

Post-doctorat

– T. En-Najjary - " Analyse de trafic" (01/2005 - 07/2006)
– M. Izal - "Réplication de contenu pair-à -pair" (01/2003-01/2004)

Stages

– C.-H. La (co-encadrement avec Marc Dacier) - Stage de fin d'étude (IFI-Vietnâm), "Etude et validation de l'application du paradigme des pots de miel aux attaques visant les protocoles de routage" (2006).
– R. Chand - Stage de DEA (UNSA), "Non équité à court terme des flux TCP" (2002).
– C. Schleippmann - Stage de diplôme (Université de Munich) - "Design and Implementation of a TCP Rate Analysis Tool" (2003).
– Encadrant académique pour 5 à 7 stages Ingénieurs Eurecom depuis 2001.

1.4 Participation à des jurys de thèse

– Matti Siekkinen, "Root Cause Analysis of TCP Throughput : Methodology, Techniques, and Applications", UNSA, Sophia-Antipolis, Octobre 2006.
– Mahamadou Issoufou Tiado, "Mécanismes multi-niveaux pour les réseaux sans fil", ENSEEIHT, Toulouse, France, Juillet 2006.
– Anwar Al Hamra, "Approaches for Scalable Content Distribution in the Internet", UNSA, Sophia Antipolis, France, Decembre 2004.
– Idris A. Rai, "QOS Support in Edge Routers", Télécom Paris, Institut Eurecom, Sophia-Antipolis, France, Septembre 2004.

1.5 Charges Collectives

Co-Responsable de la filière "Réseau" à Eurecom. Eurecom offre une formation équivalente aux deux dernières années d'un cycle ingénieur pour des étudiants venant de : Télécom Paris (ENST), l'Ecole Polytechnique Fédérale de Lausanne (EPFL), l'université technologique de Helsinky (HUT), Polytechnique Turin (Politecnico di Torino), Télécom Bretagne (ENSTB), l'Institut National des Télecommunications (INT). Certains étudiants arrivent en septembre pour un cursus 24 mois et d'autres en mars pour un cursus 18 mois.

Ils doivent choisir une filière parmi 7 filières possibles. En tant que co-responsable de la filière "Réseaux", mes attributions sont :
- Organisation du choix des cours (obligatoires, optionnels, menu) ;
- Présentation de la filière (cours, débouchés) aux étudiants en septembre et en mars ;
- Suivi des étudiants dans leur choix de cours et étude des demandes d'équivalence de cours dans le cas d'une redondance avec un cours effectué en école mère ;
- Aide à la recherche de stage de fin d'étude.
- Suivi de plusieurs étudiants durant leur stage.

Membre du groupe de travail "Cursus" à Eurecom depuis 2005 : ce groupe est en charge de l'adaptation du règlement des études pour les différents cursus (18 mois, 24 mois), des règles de validation des examens, des stages, etc. Les réunions sont pluri-annuelles.

Présentation d'Eurecom à l'Institut National des Télécommunications (1 fois par an à partir de 2006)

Comités de programme : QoSIP (2003,2005), IMSA 2006, SECRYPT (2006,2007), IEEE ICC 2007

Reviewer pour :
- Conférences : ACM Sigmetrics 2003, 2004, 2006, IEEE ICNP 2003, 2004, IEEE Infocom 2004, ACSAC 2005, ACNS 2005, ACM SIGCOMM 2005, NGI 2006.
- Revues : Performance Evaluation, IEEE Transaction on Networking, The Computer Journal, IEEE Transactions on Dependable and Secure Computing.

1.6 Projets de Recherche

2003-2006 : Réseau d'excellence RESIST
Objectif : Favoriser la collaboration entre des équipes de recherche au niveau européen dans les domaines de la fiabilité et de la sécurité.
Contributions : Représentant pour Eurecom dans le groupe de travail "Training and Dissemination" chargé de la publicité des travaux effectués dans le cadre de RESIST, de l'organisation des workshops étudiants dans le cadre du projet et de la mise au point d'un curriculum type, au niveau européen, dans les domaines conjoints de la sécurité et de la fiabilité.

2004 - 2006 : Contrat de Recherche Externalisé France Télécom R&D : Gestion de Trafic Internet
Objectif : Etude passive des performances des clients d'un opérateur d'accès.
Contribution : Responsable du projet côté Eurecom. La thèse de doctorat de M. Siekkinen et le stage post-doctorale de T. En-Najjary ont été financé pour partie par ce contrat.

2003-2006 : Réseau d'excellence E-NEXT

Objectif : Favoriser la collaboration entre des équipes de recherche au niveau européen dans les domaines des réseaux et du multimédia.
Contributions :
– Participations aux groupes de travail sur la distribution de contenu et sur les mesures dans les résaux (WG2 et WG4).
– Mise en place d'un cours de niveau master entre l'Université d'Oslo, l'Université de Darmstadt, l'Université de Lancaster et L'Institut Eurecom couvrant les domaines suivant : mesures, pair-à-pair, réseaux ad-hoc, système de base de données pour le traitement des flux continus. L'objectif était de présenter les derniers développement en recherche dans ces domaines. Les cours magistraux (enregistrés à chaque institut et mis à disposition sur un serveur centralisé à Darmstadt) étaient complétés par des projets en équipes internationales.

2001-2004 : Projet RNRT METROPOLIS (METROlogie Pour l'Internet et ses Services)
Partenaires : Eurecom, LIP6, LAAS, FT R&D, Renater
Objectifs : Développer des outils pour mesurer la QoS de l'Internet, effectuer des campagnes de mesures (actives et passives) et construire de nouveaux modèles de trafic.
Contributions : Rédaction de la soumission pour Eurecom et gestion du projet côté Eurecom.

1996-1997 : Projet européen NETPERF (Network Performance)
Partenaires : UIA (Anvers), SUG (Gand), UPC (Barcelone), UST(Stuttgart), SIMULOG, LEP.
Objectifs : Proposer et implémenter des modèles de sources et d'éléments d'interconnexion pour un outil de simulation de réseaux ATM (Modline/QNAP).
Constributions :
– Proposition d'un plan de validation de l'outil ;
– Validation de certains modèles de sources ;
– Editeur de certains livrables.

1.7 Publications

Chapitres de livre

1. "Chapter 5 : Algorithms for scalable content distribution" in "Quality of Future Internet Services, COST Action 263, Final Report", *Lecture Notes in Computer Science*, volume 2856. Editors : M. Smirnov et al., 2003.

Journaux internationaux avec comité de lecture

1. T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, G. Urvoy-Keller "From content distribution networks to content networks - issues and challenges" *Computer Journal*, March 2006.

2. P. Felber, K.W. Ross, E.W. Biersack, L. Garces-Erice, and G. Urvoy-Keller "Structured Peer-to-Peer Networks : Faster, Closer, Smarter" *IEEE Data Engineering Bulletin* 28(1) :55–62 2005.

3. I. Rai, E. W. Biersack, and G. Urvoy-Keller "Size-based Scheduling to improve the Performance of Short TCP Flows" *IEEE Network Magazine* 2005.

4. L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber and G. Urvoy-Keller, "Hierarchical Peer-to-Peer Systems". Parallel Processing Letters (PPL), 13(4) :643–657 Dec 2003

5. G. Urvoy-Keller and G. Hébuterne , "EBBPS : An efficient weight assignment for GPS servers". Annals of Telecommunications 57(11,12) :1226–1243 Nov/Dec 2002

6. G. Urvoy-Keller, G. Hébuterne, and Y. Dallery "Traffic Engineering in a Multipoint-to-point Network". IEEE Journal on Selected Area in Communications 20(4) :834–849 May 2002

7. G. Urvoy-Keller, G. Hébuterne et Y. Dallery, "CAC Procedures for Leaky Bucket-constrained Sources". Performance Evaluation 41(2) :117–132 July 2000

Conférences internationales avec actes et comité de lecture

1. P. Michiardi and G. Urvoy-Keller "Performance Analysis of Cooperative Content Distribution for Wireless Ad hoc Networks" *To Appear , in Proceedings of IEEE/IFIP WONS 2007, January 24-26, 2007, Obergurgl, Austria.*

2. A. Legout, G. Urvoy-Keller, and P. Michiardi "Rarest First and Choke Algorithms Are Enough" *In ACM SIGCOMM/USENIX IMC'2006 October 2006, Rio de Janeiro, Brésil.*

3. G. Urvoy-Keller and P. Michiardi "Impact of Inner Parameters and Overlay Structure on the Performance of BitTorrent" *In Global Internet Symposium, Barcelona, Spain, April 2006*

4. T. En-Najjary and G. Urvoy-Keller "PPrate : A Passive Capacity Estimation Tool". *In E2EMON Vancouver, Canada April 2006.*

5. F. Pouget, G. Urvoy-Keller, and M. Dacier "Time Signatures to detect multi-headed stealthy attack tools" *In 18th Annual FIRST Conference* Baltimore, Maryland, USA June 2006.

6. M. Siekkinen, E. W. Biersack, V. Goebel, T. Plagemann, and G. Urvoy-Keller "InTraBase : Integrated Traffic Analysis Based on a Database Management System" *In Proceedings of IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services* Nice,France, May 2005.

7. Matti Siekkinen, Guillaume Urvoy-Keller, Ernst Biersack, and Taoufik En-Najjary "Root Cause Analysis for Long-Lived TCP Connections" *In Proceedings of CoNEXT*, Toulouse, France, October 2005.

8. G. Urvoy-Keller "On the Stationarity of TCP Bulk Data Transfers" *In Passive and Active Measurements* 2005, Juan Les Pins, France, March 2005.

9. I. A. Rai, G. Urvoy-Keller, M. K. Vernon and E. W. Biersack "Performance models for LAS based scheduling disciplines in a packet switched". *ACM Sigmetrics 2004*, NY,USA, June 2004

10. T. Plagemann, V. Goebel, A. Bergamini, G. Tolu, G. Urvoy-Keller and E W. Biersack "Using Data Stream Management Systems for Traffic Analysis - A Case Study". To appear in *Passive and Active Measurements 2004*, Juan-les-pins, France, April 2004

11. M. Izal, G. Urvoy-Keller, E W. Biersack, P.Felber, A.A. Hamra, L. Garces-Erice "Dissecting BitTorrent : Five Months in a Torrent's Lifetime". In *Passive and Active Measurements 2004*, Juan-les-pins, France, April 2004

12. L. Garces-Erice, P. A. Felber, E. W. Biersack, G. Urvoy-Keller, and K. W. Ross "Data Indexing in Peer-to-Peer DHT Networks" *In Proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS)* Tokyo, Japan 2004

13. L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller "Hierarchical P2P Systems" *In Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)* Klagenfurt, Austria 2003

14. L. Garces-Erice, K. W. Ross, E. W. Biersack, P. A. Felber, and G. Urvoy-Keller "Topology-Centric Look-Up Service" *In Proceedings of COST264/ACM Fifth International Workshop on Networked Group Communications (NGC)* Munich, Germany 2003

15. A. A. Hamra, E. W. Biersack, and G. Urvoy-Keller "Architectural choices for Video-on-Demand Systems" *In Proc. of WCW* Hawthorne, NY USA September 2003

16. I. A. Rai, G. Urvoy-Keller, and E. W. Biersack "Analysis of LAS Scheduling for Job Size Distributions with High Variance" *In ACM Sigmetrics 2003* pages 218–228 June 2003

17. D. Choi, E. W. Biersack, and G. Urvoy-Keller "Cost-optimal Dimensioning of a Large Scale Video on Demand System" *In NGC 2002* October 2002, Boston, USA

18. I. A. Rai, G. Urvoy-Keller, and E. W. Biersack "Size-based Scheduling with Differentiated Services to Improve Response Time of Highly Varying Flow Sizes" *In ITC 15th Specialist Seminar, IP 2002* July 2002, Germany

19. G. Urvoy-Keller and E. W. Biersack "A Multicast Congestion Control Model for Overlay Networks and its Performance" *In NGC 2002* October 2002, Boston, USA

20. G. Urvoy-Keller and E. W. Biersack "DSS : A Deterministic and Scalable QoS Provisioning Scheme" *In QofIS'2001 : 2nd International Workshop on Quality of future Internet Services* pages 310–323 Coimbra, Portugal 2001

21. G. Urvoy-Keller and G. Hébuterne "Guaranteeing Deterministic QoS in a Multipoint-to-point Network" *In ITC'17 : 17th International Teletraffic Congress* pages 943–954 Salvador da Bahia, Brazil december 2001

22. G. Urvoy-Keller, G.Hébuterne et Y.Dallery, "Deterministic End-to-end Delay in an Accumulation Network", *IFIP Networking'2000 - Performance of Communication Networks (PCN)*, mai 2000, Paris, France

23. G. Urvoy-Keller, G. Hébuterne et Y. Dallery, "Delay-constrained VBR Sources in a Network Environment", *ITC'16*, juin 1999, volume 1, pages 687-696, Edimbourg, Grande-Bretagne

24. G. Urvoy-Keller, G. Hébuterne et Y. Dallery, "CAC Procedures for Delay-constrained VBR Sources", *IFIP ATM'98*, juillet 1998, session 4-B, papier n^o 39, Bradford, Grande-Bretagne

Posters

1. D. Ferrero, G. Urvoy-Keller, "A size-based scheduling approach to improve fairness over 802.11 wireless networks", ACM SIGCOMM 2006, Poster Session

Mémoires

1. Mémoire de thèse : "Qualité de Service de Bout en Bout et Algorithmes d'Admission d'Appels", Université Paris 6, 1999

2. Mémoire de DEA (sous la direction de J. Kohlenberg) : "Reséquencement dans les réseaux d'interconnexion multi-chemins". Université de Versailles, 1995

Tutoriels

1. G. Urvoy-Keller "A Tutorial on Network Calculus" *In IFIP IP & ATM July 2000* Ilkley, UK

1.8 Enseignements

Projets de dernier semestre à Eurecom : Encadrements de 3 à 4 projets (binômes) par an de dernier semestre (sept-février) sur des sujets orientés recherche, en collaboration éventuelle avec des entreprises (France Télécom R&D, TMG, etc.).

Année	Niveau	Matière	Type	Volume horaire
2005-2006	Ingénieur	Practical Performance	Cours/TD/TP	21/15/9 h
	"	Technologie des réseaux	Cours/TP	3/18 h
	"	Distributed Course (Eurecom,Oslo, Darmstadt, Lancaster)	Cours/Projets	21/21 h
2004-2005	Ingénieur	Practical Performance	Cours/TD/TP	21/15/9 h
	"	Technologie des réseaux	Cours/TP	3/18 h
	"	Distributed Course[1]	Cours/Projets	21/21 h
2003-2004	Ingénieur	Internet 3	Cours/TP	7,5/3 h
	"	Technologie des réseaux	Cours/TP	3/15 h
	"	Internet 2	TD	2 h
	DEA	Pair-à -pair	Cours	12 h
2002-2003	Ingénieur	Internet Application Layer Protocols	Cours	8 h
	"	Internet 3	Cours/TP	4/3 h
	"	Technologie des réseaux	Cours/TP	3/15 h
	"	Internet 2	TD	6 h
	"	Internet 1	TP	9 h
	"	Protocols & Networks	TP	9 h
	DEA	Pair-à -pair	Cours	4,5 h
2001-2002	Ingénieur	Internet 3	Cours/TP	6/3 h
	"	Technologie des réseaux	Cours/TP	3/15 h
	"	Internet 2	TD	2 h
	"	Internet 1	Cours/TD	16/4 h
	DEA	Forwarding Look-up	Cours	6 h
2000-2001	Ingénieur	Internet 3	Cours/TP	6/3 h
	"	Internet 2	TD	4 h
	"	Internet 1	TD	12 h
	"	Protocols & Networks	Cours/TP	2/12 h
	DEA	Forwarding Lookup	TP	3 h

Enseignements (Antérieurs à Eurecom)

Année	Niveau	Matière	Type	Volume horaire
1999-2000	DEA Licence Ingénieur Ingénieur	Qualité de Service Théorie de la transmission Fondement des réseaux Architecture des réseaux	Cours TD TD TD	3h 30 h 30 h 30 h
1998-1999	IUT Mastère Ingénieur	Introduction aux Réseaux évaluation de performances Gestion de Trafic	TD TP TD/TP	30 h 12h 6/12h
1997-1998	DEUG IUT Mastère	Algorithmique et Pascal Réseaux locaux évaluation de performances	TP TD/TP TP	30 h 21/30 h 20h

1.9 Programme de recherche

Je présente dans cette section les travaux de recherche que j'ai effectués depuis l'année 1996 jusqu'à cette année ainsi que les perspectives liées à ces travaux. Le premier thème, "Qualité de Service Déterministe" est lié directement à mes travaux de thèse de doctorat. Le thème "Politiques de services reposant sur la taille" a été développé dans le cadre de la thèse d'Idris Rai que j'ai co-encadrée avec Ernst Biersack. Dans le thème "Analyse de trafic", on retrouve plusieurs types de travaux/collaborations. Tout d'abord, des résultats obtenus par Matti Siekkinen durant sa thèse de doctorat qui s'achèvera fin 2006. Sont également présentés dans cette section des travaux effectués seul ou en collaboration avec Taoufik En-Najjary, Pietro Michiardi et Arnaud Legout, autour de la distribution de contenus et des mesures passives. Ces thèmes vont rester des sujets actifs de recherche pour moi dans les prochaines années. Enfin, dans une dernière partie, je présente des "Thèmes Futurs Spécifiques" sur des sujets de recherche que je développe depuis peu.

1.9.1 Qualité de Service Déterministe

Ce thème de recherche a constitué le sujet de ma thèse de doctorat (1996-1999) effectuée sous la direction de Gérard Hébuterne (Télécom-INT) et Yves Dallery (école Centrale de Paris). Ces travaux de thèse doivent être replacés dans le contexte de l'époque, à savoir d'un côté un monde télécom qui poussait pour faire aboutir le modèle ATM qui visait à transformer les réseaux téléphoniques filaires en réseaux multiservices et le monde IP qui voulait un support natif de la qualité de service au sein des routeurs de l'Internet. Une particularité intéressante des modèles de qualité de service prônés par l'IETF et l'ATM Forum est qu'ils supposent tous deux des connexions contraintes par des leaky-bucket (ATM) ou token buckets (IP). Ces modèles cherchent à caractériser des connexions au travers d'un comportement à long terme et d'une mesure quantitative de la divergence par rapport à ce comportement. Ils sont similaires au sens où le premier est une version fluide du second

Les problèmes traités dans la thèse sont des problèmes de performance. Une approche traditionnelle des problèmes de performances dans les réseaux est une approche probabiliste. Au contraire, l'approche utilisée dans la thèse est déterministe, reposant sur le Network Calculus. Le Network Calculus est un formalisme qui permet d'obtenir des bornes sur les

délais et les tailles des files d'attente dans des réseaux de communication. Nos contributions dans le domaine sont :

- L'analyse déterministe de disciplines de services offrant différents degrés de multiplexage des connexions dans le cas d'un serveur simple.
- L'étude de la généralisation de ces modèles au cas multi-serveurs. Nous avons montré la difficulté de traiter le cas général d'une topologie de réseau quelconque dans un cadre déterministe, dans la mesure où les trajectoires qui mènent au pire cas (nécessaires pour trouver des bornes déterministes) sont très difficiles à construire.
- L'analyse des réseaux en arbre, dit aussi réseaux d'accumulation, que l'on rencontre dans des cas pratiques tels les réseaux overlay MPLS et qui se prêtent à des études déterministes.
- L'analyse de réseaux fondés sur un *shaping* à l'entrée du réseau à un débit égal à "leur bande passante effective". L'idée est de se ramener au sein du réseau à l'étude de flux à débits constants (bien que les flux en entrée soient à débit variable). Nous avons effectué une comparaison systématique de cette approche avec une approche utulisant des ordonnanceurs GPS (Generalized Processor Sharing) où aucun *shaping* n'est fait à l'entrée du réseau et montré que notre approche consomme moins de ressources dans le réseau pour un même niveau de qualité de service.

1.9.2 Politiques de Services fondées sur la Taille

Au début des années 2000, l'Internet était devenu si grand qu'introduire des mécanismes de qualité de service sur tous les routeurs était devenu une tâche très complexe. Par ailleurs, la course au débit des opérateurs avait généré un cœur de réseau surdimensionné et relégué les goulots d'étranglement sur les "bords" du réseau (par exemple les serveurs de contenu). Notre approche durant la thèse d'Idris Rai (2001-2004) a été de proposer des solutions pour améliorer les performances perçues par l'utilisateur en ce concentrant sur ces goulots d'étranglement et en tirant partie des caractérisations du trafic dérivées de mesures réelles. Ces dernières ont montré que la majorité des flots observés sont interactifs (souris pair-à -pair et Web notamment) mais qu'ils ne représentent qu'une faible fraction de la charge totale du réseau. Nous avons alors eu l'idée de remplacer la discipline de service FIFO par une politique qui tiendrait compte de la taille des flots. La politique que nous avons utilisée est LAS (Least Attained Service) qui est connue depuis les années 1970. Notre argument principal est qu'en utilisant LAS dans des endroits spécifiques où des goulots d'étranglement se forment (notamment sur les bords du réseaux - par exemple, les concentrateurs ADSL) on pourrait accroître l'interactivité perçue par l'utilisateur au prix d'un léger ralentissement des connections longues (typiquement, transfert de gros fichiers). Nos contributions dans ce domaine de recherche sont :

- L'analyse de LAS pour une file M/G/1, où G est une distribution avec une forte variabilité. Nous avons comparé LAS avec un large éventail de politiques : PS, SRPT, les politiques non-préemptives (NPP) telles FIFO, LIFO, etc. Nous avons également étudié LAS dans le cas de surcharge $\rho \geq 1$. De manière générale, nous avons montré que LAS réduit le temps de réponse des clients courts sans pénaliser notablement les clients longs si la distribution des tailles de clients est suffisamment variable.
- L'analyse de performance de LAS dans un réseau paquet et l'étude de l'interaction de

LAS avec le protocole TCP. Nous avons simulé LAS avec le simulateur ns-2 et comparé les performances de LAS avec FIFO dans le cas d'un unique goulot d'étranglement pour des tailles de flot fortement variables. Nos simulations ont montré que, comparé à FIFO, LAS diminue fortement le temps de réponse des flots ainsi que leur taux de pertes.
– L'évaluation de LAS dans les réseaux pathologiques où la bande passante se trouve partagée inéquitablement entre les sources en raison d'une asymétrie en terme de latence des chemins des sources ou bien à cause de protocoles de transport différents ou encore de l'existence de multiples goulots d'étranglement. Les résultats de simulation montrent que LAS est plus efficace que FIFO dans ces cas car il rétablit toujours l'équité entre les sources.
– La conception, la modélisation et l'analyse de politiques de services dérivées de LAS et offrant des services différenciés. L'objectif principal de ces politiques est de protéger les flots longs transportant des données prioritaires. Pour les différentes politiques proposées, nous avons développé des modèles analytiques au niveau flot permettant d'obtenir les temps de réponse conditionnels de ces politiques. Ces modèles ont été validés par simulation.

Nous étudions actuellement les suites de ces travaux au travers d'une part de l'implantation dans un routeur linux de LAS et d'autre part, de la transposition de ces résultats au cas des réseaux sans-fil (voir section 1.9.4).

1.9.3 Analyse de Trafic

L'explosion de l'Internet (en termes de taille et de volume) a rendu nécessaire l'étude *in vivo* des protocoles et applications afin de vérifier que leur comportement est conforme à leurs objectifs. Une caractéristique de l'Internet est que les applications qui l'utilisent changent à un rythme rapide. On pense notamment à l'explosion du trafic Web au début des années 1990 et plus récemment au pair-à -pair, aux flux RSS (Real Simple Syndication) ou du trafic multimédia type Skype. On peut néanmoins noter que TCP (Transport Control Protocol) demeure le protocole de niveau transport dominant avec typiquement 80 à 90% des octets d'un lien donné. Pourtant, TCP a été déployé au début des années 80 avec des hypothèses de trafic qui diffèrent fondamentalement de ce qu'est le trafic aujourd'hui. De plus, de nombreuses versions (Reno, New Reno, SACK) et implantations (valeurs de timers différents, etc.) existent et cohabitent aujourd'hui. Pour toutes ces raisons, TCP reste un sujet de recherche important. Nos contributions dans ce domaine d'étude font l'objet de la section "Niveau transport" ci-après.

Dans le domaine de l'analyse de trafic, je mène également des études sur une application spécifique, BitTorrent. Cette application pair-à -pair vise à une réplication rapide d'un contenu donné sur un grand nombre de pairs. Des études ont montré que cette application est capable à elle-seule de représenter la majorité des octets transportés sur certains liens, notamment aux Etats-Unis et en Angleterre. La réplication de contenu à large échelle est sans aucun doute un enjeu majeur pour l'Internet mais aussi pour les Intranets des entreprises qui ont de plus en plus besoin de mises à jour continuelles de leur parc de machines. Nous détaillons nos contributions sur BitTorrent dans la section "Niveau applicatif" ci-après.

Niveau transport

Déterminer la cause qui limite le débit de connexions TCP capturées sur Internet est essentiel pour comprendre où sont les limitations actuelles du réseau. Savoir si la limitation provient de l'application, des couches TCP de l'émetteur ou du récepteur, ou de containtes liées au réseau (congestion, goulot d'étranglement) est essentiel pour les concepteurs d'applications réseaux ainsi que pour les opérateurs d'accès ou de cœur. La compétition entre ces derniers exige en effet qu'ils aient une vision fine des performances perçues côté client. Ils ne peuvent plus seulement se fonder sur des indicateurs globaux, telle l'utilisation de leur équipements physiques au sein de leur réseau. Nos contributions dans le domaine de l'analyse opérationnelle de TCP sont les suivantes :

– La mise au point d'un algorithme qui permet de quantifier l'impact exact de l'application sur un flux TCP et ce, indépendamment de l'application. Cela signifie que notre méthode peut être appliquée sur un flux applicatif sur lequel on n'a aucun *a priori* (application client/serveur ou pair-à -pair, multimédia ou non). Cette dernière propriété est essentielle car de nouvelles applications apparaissent sans cesse dans l'Internet. Il est aussi possible que le flux TCP soit crypté.
– Une nouvelle méthode reposant sur l'assignation de scores obtenus à partir de différentes séries temporelles issues de l'observation d'un flux TCP pour inférer la cause qui limite le débit. Cette méthode s'est montrée plus précise et plus fiable que les méthodes précédentes.
– Une avancée dans le domaine méthodologique en proposant d'utiliser une approche base de données pour effectuer les analyses de trafic. Cette méthode se révèle efficace pour des traces de taille moyenne (à l'échelle d'un opérateur réseau), de l'ordre de 10 Gigaoctets.

Nous continuons actuellement nos efforts dans le domaine de l'analyse de TCP. Ces efforts sont menés pour partie conjointement avec France Télécom. Par ailleurs, les bords (réseaux d'accès) de l'Internet utilisant de plus en plus des technologies sans-fil, nous projetons d'étudier spécifiquement les performances de TCP dans ce type d'environnement filaire/sans-fil.

Niveau applicatif

Le second problème d'analyse de trafic sur lequel nous avons travaillé est l'étude d'un protocole pair-à -pair particulier, BitTorrent (BT). Nous avons étudié BT dans l'objectif de comprendre si les algorithmes de choix des pairs et des pièces utilisées par l'application ne doivent pas former à terme l'ossature de toute distribution de contenu dans l'Internet. Ce problème est critique pour la mise à jour d'application ou d'antivirus à grande échelle dans l'Internet et aussi dans les entreprises où la gestion de parcs de machines Windows reste un souci majeur des services informatiques. Nos études ont montré que BT est conçu de manière très intelligente et capable de passer à l'échelle. De plus, nous avons montré que lorsque de mauvaises performances étaient observées, elles n'étaient pas imputables directement aux algorithmes de BT. Les perspectives sur BT concernent des études plus théoriques sur la distribution de contenu et dans des environements particuliers tels les réseaux ad-hoc.

1.9.4 Thèmes Futurs Spécifiques

En plus de la poursuite des thèmes précédents, je souhaite développer deux nouveaux thèmes de recherche : la "Détection d'intrusion et attaque sur l'infrastructure de routage", en collaboration avec Marc Dacier (Eurecom) et les "Réseaux sans-fil maillés", dans le cadre de la thèse de Diego Ferrero, dont je suis le directeur de thèse.

Détection d'intrusion et attaque sur l'infrastructure de routage

La détection d'intrusion dans l'Internet est un domaine très actif. L'équipe autour de Marc Dacier à Eurecom s'attache à la caractérisation à large échelle des processus d'attaque de par le monde sur les machines d'utilisateurs privés ou institutionnels. L'approche suivie est le déploiement de plusieurs sondes (actuellement 35) sur les 5 continents qui collectent des traces d'attaque. Ces machines n'offrent aucun service et toute tentative de connexion sur ces machines peut donc être considérée comme potentiellement suspecte. Au delà de l'étude d'un vers ou d'un outil particulier, l'objectif est une connaissance des processus d'attaques (combien d'outils, avec quelle fréquence, est-ce que les processus d'attaque sont homogènes ou non du point de vue géographique ou temporel). Nous souhaitons, parallèlement à ces travaux en cours, mener des études sur les attaques potentielles sur l'infrastructure même de l'Internet, à savoir ses routeurs. Ces machines sont *a priori* très sécurisées, mais le protocole BGP lui-même ne l'est pas. Il souffre de nombreuses limitations car plus qu'un protocole de routage et de découverte de plus-court chemin, c'est un protocole d'implantation de politiques commerciales. Nous souhaitons mener des études par mesures sur le sujet pour déterminer si oui ou non des attaques ont effectivement lieu contre l'infrastructure du réseau. Ce sujet est très sensible car si des attaques ont eu lieu, il est fort probable que les victimes n'en font pas de publicité à cause des enjeux économiques sous-jacents. Par ailleurs, le problème est complexe d'un point de vue technique car il faut savoir où placer des sondes et comment différencier les attaques du bruit naturel de BGP : 10% des préfixes sont instables au sens où des updates BGP sont continuellement envoyés. Notons que ce ne sont pas toujours les mêmes préfixes qui sont instables et ces problèmes ne sont pas des problèmes de connectivité mais des problèmes liés à du *multi-homing ou à des politiques commerciales qui engendrent des oscillations dans les routes. Un acteur majeur* dans le domaine, CISCO, souhaite travailler sur le sujet avec nous. En effet, un des enjeux majeurs dans le futur pour CISCO est la sécurisation de BGP. La connaissance de ce qu'il y a vraiment à sécuriser (le protocole, les routeurs eux-mêmes ou les pratiques d'adminstration) est donc une information cruciale pour CISCO.

Réseaux sans-fil maillés

Les réseaux sans-fil maillés (wireless mesh networks - WMN) étendent la couverture classique des fournisseurs d'accès en utilisant des technologies sans-fil. D'un point de vue économique, déployer des points d'accès reliés de manière filaire à l'Internet est coûteux. Une solution moins onéreuse est de relier par voie hertzienne différents points d'accès et former un WMN. Selon les projections, il faudrait que 20% des points d'accès soient effectivement reliés à l'Internet pour assurer un service satisfaisant. Les WMNs sont une technologie d'avenir avec de gros investissements en perspectives, notamment des collecti-

vités locales. Il reste pourtant de nombreux problèmes à résoudre pour faire fonctionner de manière opérationnelle ces réseaux. Notons que ces problèmes diffèrent de ceux des réseaux ad-hoc qui sont mono-radio, mono-canal et bien souvent mono-applicatif. Au contraire les réseaux WMN sont au minimum multi-radio et multi-applicatif avec un trafic asymétrique (beaucoup de trafic de l'Internet vers les utilisateurs). Dans le cadre de la thèse de doctorat de Diego Ferrero qui a débuté en mars 2006, nous aborderons les problèmes suivants :

– Problèmes architecturaux : comment appliquer des techniques d'équilibrage de charges entre passerelles sans-fil/Internet ?
– Comment supporter une variété d'applications, certaines étant interactives, d'autres des transferts plus longs ?
– Quel est le comportement de TCP dans ce type d'environnement. Comment définir et assurer l'équité entre flux TCP ?

1.10 Suite du Document

Dans le chapitre 2 sont résumés mes travaux de thèse effectués sous la direction conjointe de Gérard Hébuterne, professeur à Télécom-INT et Yves Dallery, professeur à l'école Centrale de Paris. Le thème de cette thèse de doctorat est la garantie déterministe dans les réseaux Internet et ATM.
Dans le chapitre 3, nous présentons les travaux effectués dans le cadre de la thèse de doctorat d'Idris Rai, soutenue en septembre 2004, que j'ai co-encadré avec Ernst Biersack.
Dans le chapitre 4, nous présentons nos travaux sur l'analyse de trafic. Ce chapitre contient deux sous parties. La première présente nos travaux sur le protocole TCP, effectués en majorité dans le cadre de la thèse de doctorat de Matti Siekkinen (soutenance prévue fin 2006). La seconde concerne la distribution de contenu sur l'Internet avec l'exemple emblématique de BitTorrent.

Chapitre 2

Garanties Déterministes de Qualité de Service dans des Réseaux Paquets

2.1 Contexte

Les résultats présentés dans ce chapitre ont été obtenus durant ma thèse de doctorat. Ces travaux datant de la période 1996-1999, il importe de les replacer dans le contexte de l'époque. À la fin des années 90, l'Internet prend une importance croissante. Les grandes entreprises se doivent d'avoir un site web qui ne soit pas seulement un miroir statique de leurs activités, mais propose des contenus régulièrement mis à jour telles des offres d'emploi. Plus généralement, on commence à percevoir dans ces années-là le potentiel du commerce en ligne. On est en fait juste à la veille de l'explosion de "la nouvelle économie". L'industrie des télécommunications, habituée à monopoliser les services de communication à plus fort rendement tels le fax ou les services clientèles, ne voient pas l'Internet d'un bon œil. Ces compagnies veulent faire évoluer leur réseau afin de supporter de nouveaux types de service. Cette situation a conduit au développement de la norme ATM (Asynchronous Transfer Mode) qui visait à faire évoluer les réseaux téléphoniques fondés sur un modèle orienté connexion, en des réseaux multi-services capables de multiplexer efficacement des flux temps réels et non temps-réel (*best-effort*). Ce dernier point sous-entendait qu'une part importante des nouveaux services serait de type temps réel, et notamment la voix sur IP, la vidéo à la demande ou le *streaming vidéo* (télévision). Avec le recul, on peut remarquer que ce ne sont pas ces services qui ont généré des revenus puisque ceux-ci commencent à apparaître depuis 2005 seulement. Et encore, si on prend l'exemple emblématique de Skype[13], le modèle économique utilisé est de ne faire payer que les communications vers des postes fixes (sur le RTC). Au contraire, de nombreux services à forte valeur ajoutée, qui ont émergé ces dernières années, ont été des services non temps réels. Les exemples les plus emblématiques sont le système d'enchères d'ebay et le système de vente de livres amazon (qui évolue vers la vente de biens informatiques). Les revenus publicitaires sur Internet ont engendré un modèle économique spécifique où l'objectif est de capter et de fidéliser une communauté d'utilisateurs aussi grande possible, leur fournir un service gratuit et générer des revenus en faisant payer les annonceurs publicitaires. C'est le cas des moteurs de recherche tels google ou yahoo. La communauté Internet, tout comme la communauté télécom a pensé, à la fin des années 90, que les

efforts devaient être concentrés sur les services temps réel. Cela s'est traduit à l'IETF par la création de deux groupes de travail emblématiques : IntServ (Integrated Services) et DiffServ (Differentiated Services). Les mondes "données" et "télécom" n'étant pas disjoints, de grands noms du monde télécom (Bell labs) se retrouvent derrière les propositions d'IntServ et de DiffServ. L'idée derrière IntServ et DiffServ était de surimposer au service "best-effort" de l'Internet, des capacités de garanties de service.

L'état d'esprit à la fin des années 90 était donc qu'il fallait absolument créer un nouveau réseau ou faire évoluer l'Internet, en façonnant des briques de base pour des services avec des contraintes de qualité de services fortes en termes de délai et ou gigue et ou perte. Replacer dans le contexte technologique de l'époque, à savoir des accès Internet bas débit et des réseaux surdimensionnés, cette vision était justifiée. L'explosion des accès haut débit (ADSL, câble, peut être Wimax sous peu) et la stratégie de surdimensionnement délibéré des réseaux de cœur et d'accès, ont profondemment modifié la donne.

Les travaux de thèse présentés ci-après, se replacent dans ce contexte. Une particularité intéressante des modèles IETF et ATM est qu'ils modélisent les flux applicatifs au travers de modèles de leaky-bucket ou token buckets (sauts percés - ces deux modèles sont similaires, le premier étant une version fluide du second). Ces modèles caractérisent des connexions au travers d'un comportement à long terme et d'une mesure quantitative de la divergence par rapport à ce comportement. En partant de cette modélisation des sources, les questions posées dans la thèse sont les suivantes :

- Peut-on prédire des bornes sur le délai maximum de délivrance des paquets pour des sources décrites par un *leacky bucket* et des serveurs FIFO (First In First Out) ?
- Comment construire une politique d'admission d'appel qui permette de garantir de la qualité de service (en terme de délai maximum) pour des sources contraintes par des *leaky-buckets* et un réseau FIFO ?
- Que peut-on faire si la politique de service n'est plus FIFO mais GPS (Generalized Processor Sharing) ?

Notons que la faisabilité de modéliser des sources à l'aide de modèles de type leacky buckets, bien qu'importante, n'a pas été abordée dans la thèse. Les problèmes traités sont des problèmes de performance. Une approche traditionnelle des problèmes de performances réseaux est une approche probabiliste souvent fondée sur la théorie des files d'attente. Au contraire, l'approche utilisée dans la thèse est de type déterministe. Le principal outil d'analyse déterministe est le Network Calculus [23]. Après les travaux initiaux de Réné Cruz de l'Université de San Diego en Californie (UCSD), le Network Calculus est réellement devenu populaire avec les travaux de Jean-Yves Le Boudec de l'EPFL [21] et de Chen Sheng Chang de l'Université Tsing Hua à Taiwan [27]. Le Network Calculus permet d'obtenir des bornes déterministes sur (notamment) les délais dans des réseaux de communications.

Dans la suite de ce chapitre, nous allons présenter les modèles de qualité de service de l'ATM et et de l'IETF puis le Network Calculus. Bien que la présentation de ces notions soient un prérequis nécessaire à la bonne compréhension des résultats de la thèse, il faut préciser que seuls les concepts de base du Network Calculus ont été utilisés pour déduire les résultats présentés ci-après.

2.1.1 Modèle de Source ATM et IP

Dans l'ATM, IntServ [124, 107, 108] et DiffServ [18, 65, 70], les sources sont décrites par des modèles de *leaky-bucket* qui se modélisent bien à l'aide du Network Calculus. Les descripteurs de trafic en ATM et IntServ/DiffServ sont définis au travers de deux algorithmes : le GCRA (*Generic Cell Rate Algorithm*) [3] et le *Token Bucket* [32] respectivement. Ces deux mécanismes sont déterministes, c'est-à -dire qu'une décision déterministe (accès ou rejet/marquage) des paquets/cellules est prise pour chaque paquet/cellule. Ils sont très similaires, et les études de type fluide (le modèle générique de source est présenté dans la section suivante) permettent d'englober les deux.

2.1.2 Le Network Calculus

Le *Network Calculus* a été introduit par R.L. Cruz [45, 43, 44]. Il a été enrichi et simplifié dans son formalisme par J.-Y. Le Boudec [21, 22] et C.-S. Chang [26]. C'est une méthode d'analyse déterministe. L'objectif principal est l'obtention de bornes déterministes, en terme de délais (locaux et de bout en bout) et de taille de files d'attente.

Nous présentons ci-après les bases utiles pour comprendre les résultats de ce chapitre. Nous avons pris le parti de les présenter en insistant sur leur utilité plutôt que d'adopter un formalisme fort.

Le point de départ du Network Calculus est de capturer le comportement des connexions[1] et des éléments actifs du réseau au travers de fonctions mathématiques. Ces fonctions sont appelées courbes d'arrivée pour les sources et courbes de service pour les éléments actifs. La courbe d'arrivée est une borne supérieure sur la quantité de données émise sur tout intervalle de temps $\tau > 0$ par une source. Un cas important pour cette thèse est celui d'une source conforme à un *leaky bucket*. On note p le débit crête de la source, M la taille maximale de la rafale (la taille du réservoir ou *bucket* correspondant, est $M' = M\frac{p-R}{p}$) et R le débit d'arrivée des jetons. Ce mécanisme correspond au schéma de la figure 2.1. On le note $LB(p, R, M)$.

La courbe d'arrivée (minimale) correspondant à une source contrôlée par un *leaky bucket* est : $\alpha(\tau) = \min(p\tau, R\tau + M\frac{p-R}{p})$. Une trajectoire importante pour une source contrainte par une courbe d'arrivée donnée est la trajectoire vorace. Cette trajectoire est telle que la courbe de débit cumulé suit exactement la courbe d'arrivée minimale. Cette trajectoire joue un rôle important dans la recherche de bornes de bout en bout.

Pour modéliser les serveurs, le Network Calculus utilise une courbe de service qui fournit une borne inférieure sur le service fourni. Dans le cas simple d'un serveur FIFO utilisant une discipline *work-conserving* avec un débit maximum C, une courbe de service offerte à l'agrégat de sources qui traverse ce serveur est $\beta(\tau) = C\tau$.

Une fois les sources et serveurs modélisés, le Network Calculus fournit un ensemble de théorèmes pour combiner les courbes de services entre elles et déduire des bornes de bout en bout sur le délai, ou pour dimensionner un réseau d'éléments pour une contrainte de QoS donnée. Ci-après, nous introduisons les deux théorèmes nécessaires pour la compréhension des résultats de ce chapitre.

[1] Connexions ou sources - nous employons ces deux termes de manière équivalente.

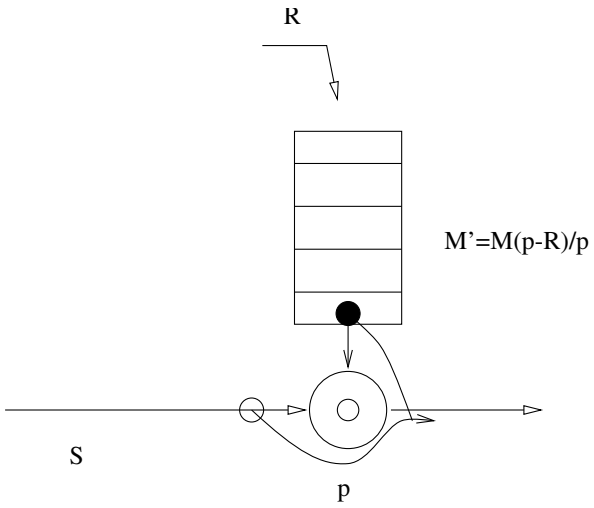


FIG. 2.1 – Source conforme au LB(p,R,M)

Le premier théorème sert à la détermintation de bornes sur le délai tandis que le second sert au dimensionnement d'un élément pour un trafic d'entrée avec des contraintes de QoS données.
Considérons un système $\mathcal{S}$ pouvant être un élément ou un ensemble d'éléments actifs. Soit $R(t)$ et $R^*(t)$ les fonctions débit cumulé en entrée et sortie de $\mathcal{S}$. Le *backlog* à l'instant t est $b(t) = R^*(t) - R(t)$. R. Cruz définit également le *délai virtuel* $d(t)$ à l'instant t : $d(t) = \inf\{T : T \geq 0,\ R^*(t+T) \geq R(t)\}$.

Théorème 1 *Soit un flux contraint par une courbe d'arrivée α et un système offrant une courbe de service β. On a :*

$$b(t) \leq v(\alpha, \beta)$$

$$d(t) \leq h(\alpha, \beta)$$

où, $v(\alpha, \beta)$ et $h(\alpha, \beta)$ sont respectivement les distances horizontales et verticales maximales entre les courbes représentant α et β (voir figure 2.2), soit :

$$v(\alpha, \beta) = \sup_{s \geq 0}(\alpha(s) - \beta(s))$$

$$h(\alpha, \beta) = \sup_{s \geq 0}(\inf\{T : T \geq 0,\ \alpha(s) \leq \beta(s+T)\})$$

Considérons maintenant un lien à débit constant C servant une source ayant une courbe de service α avec une discipline de service *work-conserving* . Soit D le délai maximum que le flux peut subir. Une question fondamentale en ingénierie de trafic est la valeur minimale de la bande passante C_D qui garantit un délai borné D. Pour répondre à cette question, J.-Y. Le Boudec a introduit la notion de "bande passante équivalente déterministe" (*Deterministic Effective Bandwidth*).

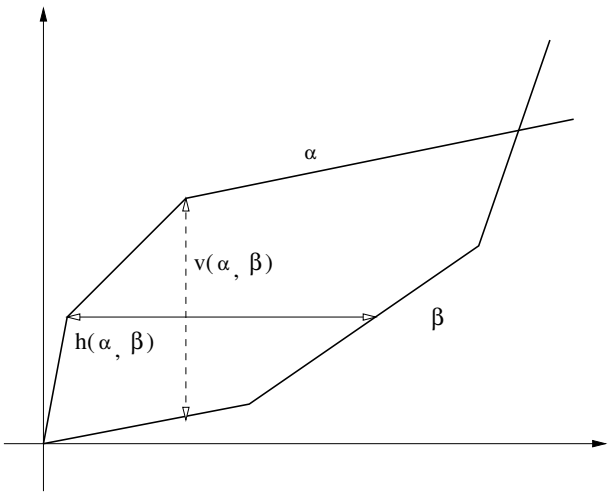


FIG. 2.2 – Bornes supérieures sur le délai et le backlog

Théorème 2 *Un serveur de capacité C garantit un délai D au flux contraint par la courbe d'arrivée α si $C \geq e_D(\alpha)$:*

$$e_d(\alpha) = \sup_{s \geq 0}(\frac{\alpha(s)}{s+d}).$$

2.2 Contributions

Dans cette section, nous présentons de manière synthétique les résultats obtenus durant la thèse de doctorat. L'ordre suivi est chronologique.

2.2.1 Cas d'un serveur simple

Le Network Calculus étant un outil d'analyse nouveau avec des résultats et surtout des démonstrations souvent peu intuitives, la première question que nous nous sommes posée a été de savoir si les bornes obtenues étaient "non triviales". Nous sommes partis du problème suivant. Soit n sources avec des contraintes de QoS et un serveur de capacité C, quelle est la valeur de C minimale qui garantie les contraintes de QoS de toutes les sources pour une politique de service donnée ?
On considère trois politiques de services différentes suivant le degré de connaissance que le serveur a des connexions qu'il sert. Avec la première politique, le serveur ne maintient aucun état sur les connexions et sert les paquets dans leur ordre d'arrivée. C'est la politique FIFO que nous renommons CS pour "Complete Sharing". Dans la seconde politique, le serveur maintient un état par connexion. Le débit global est alors la somme des débits alloués par connexion. Ce cas correspond à un routeur IntServ ou à un commutateur ATM au travers duquel passe des flux CBR et VBR. La dernière politique envisagée est un compromis entre les deux politiques précédentes. On maintient des états, non par connexion, mais par groupes de connexions. Un groupe de connexions est ici défini

comme l'ensemble des connexions ayant des contraintes de QoS communes. L'idée de cette politique est d'offrir un service similaire, dans l'esprit, à DiffServ. On nomme ClassP cette dernière politique.
On adopte les notations suivantes. Le serveur a un débit constant C et les n sources sont décrites chacune via un couple (courbe d'arrivée, délai demandé)=(α_i, d_i), $i \in \{1, n\}$. On suppose qu'on a p classes de service, c'est-à -dire p valeurs possibles pour les délais d_i, $i \in \{1, n\}$: $(\mathcal{D}_1, ..\mathcal{D}_p)$, avec $\mathcal{D}_1 \leq \mathcal{D}_2 \leq ...\mathcal{D}_p$. Le théorème sur le calcul de la bande passante effective permet de déterminer le débit minimum C pour chacune des 3 politiques de service. On obtient :

$$C_{CS} = \sup_{\tau \geq 0} \Big(\frac{\sum_{i=1}^{n} \alpha_i(\tau)}{\tau + \min_{i \in \{1,n\}} d_i} \Big) \tag{2.1}$$

$$C_{CP} = \sum_{i=1}^{n} \Big(\sup_{\tau \geq 0} \big(\frac{\alpha_i(\tau)}{\tau + d_i} \big) \Big) \tag{2.2}$$

$$C_{ClassP} = \sum_{c=1}^{p} \Big(\sup_{\tau \geq 0} \Big(\frac{\sum_{i_c \in I_c} \alpha_i(\tau)}{\tau + \mathcal{D}_c} \Big) \Big) \tag{2.3}$$

Au-delà de la formulation mathématique du problème, nous voulons comprendre le gain de bande passante que l'on peut obtenir en utilisant une politique de service plutôt qu'une autre. Nous avons obtenu deux types de résultats. Des résultats analytiques et des résultats numériques.

Résultats analytiques

Il n'y a pas une politique qui offre toujours systématiquement les meilleures performances, i.e. consomme le moins de bande passante pour garantir les contraintes de QoS. Tout dépend du degré d'homogénéité des connexions et de leurs contraintes de QoS. Nous avons obtenu les résultats suivants :

Règle n°1 : si toutes les sources sont identiques mais demandent des délais différents, il est plus intéressant de les séparer (avec CP) que de les grouper (avec CS).

Règle n°2 : si toutes les sources sont différentes mais demandent le même délai, il vaut mieux les grouper (avec CS) que séparer (avec CP).

Règle n°3 : la politique ClassP demande systématiquement moins de bande passante que la politique CP

Résultats numériques

L'intuition naturelle que l'on peut développer pour les politiques CP, CS et ClassP est que la politique ClassP devrait, dans la majorité des scénarios, être la moins gourmande en bande passante. Pour illustrer cette intuition, nous avons effectué une campagne

numérique où les descripteurs de trafic et de QoS des connexions sont tirés aléatoirement. Les formules analytiques précédentes sont ensuite utilisées pour déterminer le besoin en bande passante de chacune des politiques. Les descripteurs de sources sont tirés en utilisant le tableau 2.1. Les délais sont supposés pouvoir prendre 3 valeurs. Cette hypothèse d'un nombre fini et faible de délais possibles est raisonnable car un opérateur qui offrirait un service de type VBR le déclinerait vraisemblablement en un petit nombre de classes, correspondant chacune à un délai. Les résultats numériques (figure 2.3) montrent que ClassP est en général la meilleure politique lorsque les sources ayant les contraintes les plus strictes en terme de bande passante (axe des x) ne sont pas majoritaires. Notons néanmoins que ces résultats sont des résultats en moyenne et il est possible de trouver des cas où CS est meilleure que ClassP, même si ceux-ci sont rares. On remarque aussi que les gains obtenus peuvent être vraiment significatifs, ce qui montre que les bornes obtenues avec le Network Calculus ne sont pas triviales.

TAB. 2.1 – Paramètres des sources

Débit crête p	*Débit moyenR*	*Sporadicité M*
10	0.1	1
100	1	10
1000	10	100

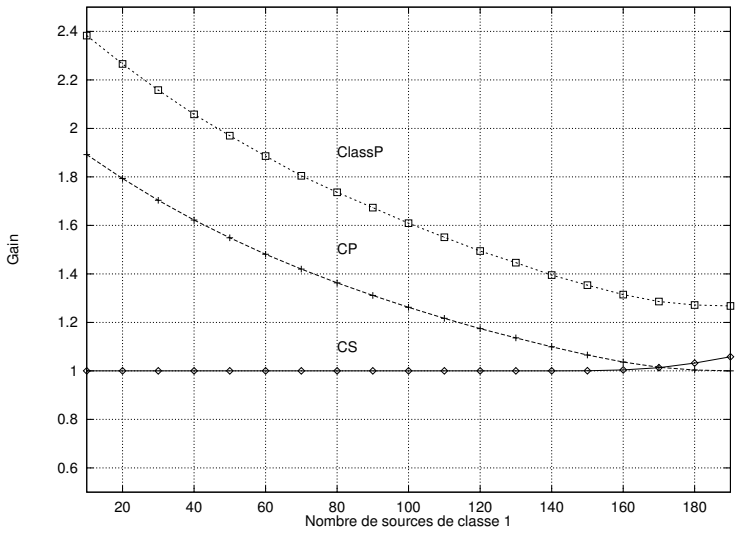


FIG. 2.3 – Gains relatifs de CP, CS et ClassP

2.2.2 Réseaux de serveurs

Le Network Calculus offre un certain nombre de théorèmes qui permettent soit de déduire une courbe d'arrivée pour une source une fois que celle-ci a traversé un serveur, soit de

combiner entre elles les courbes de services d'un ensemble de serveurs pour obtenir une courbe de service globale, appelée courbe de service réseau.
La première question que nous nous sommes posée lors du passage serveur simple/réseau de serveurs est la possibilité d'étendre les schémas CS/CP/ClassP. Il apparaît que l'extension à un cas réseau est difficile pour les raisons suivantes.
CP est la politique qui se prête le plus facilement à une analyse avec le Network Calculus car des ressources sont dédiées à chaque routeur. Cependant la ressource qui est dédiée est la bande passante et non l'espace mémoire. Si on veut dimensionner les tampons mémoire, la dimension du problème est de l'ordre du nombre de sources et du nombre de serveurs. Au-delà de ce problème, la question de l'implantation de CP rejoint celle d'IntServ. Le problème de scalabilité est si important qu'il faut envisager une approche par groupage des sources dans un réseau réel, que ce groupage soit partie (ClassP) ou total (CS).
Pour ClassP, le problème que nous avons rencontré est "l'assignation" d'une source à une classe. En effet, l'assignation doit prendre en compte le délai maximum de bout en bout toléré par une source mais aussi le nombre de serveurs traversés car le délai est une métrique additive. Si par exemple, deux sources ont la même contrainte de délai mais la première traverse un serveur et la seconde dix (le premier étant commun aux deux sources), il n'est pas possible d'assigner une même classe aux deux sources au niveau du premier serveur. La généralisation de ce cas simple montre que le nombre de classes nécessaires croît rapidement avec la taille du réseau.
La politique CS (i.e. FIFO) pose le problème de détermination d'une borne de bout en bout dans le cas général où les sources ne suivent pas le même chemin. Nous n'avons pas trouvé à l'époque une courbe de service pour une source particulière dans un serveur FIFO. Des avancées ont été faites récemment sur cette question [29]. Ce problème est important pour déterminer une bonne borne de bout en bout. Sans cette courbe de service, il est toujours possible de sommer les courbes d'arrivées de toutes les connexions à chaque serveur, puis d'utiliser le thèorème 1. On obtient alors une borne de bout en bout en sommant les bornes obtenues indépendamment sur chaque serveur. Ce faisant, on risque d'obtenir des bornes de mauvaise qualité. Nous montrons plus loin qu'il existe une classe de réseaux pour laquelle cette borne *a priori* naïve est en fait atteignable. Plus généralement, la raison pour laquelle on peut penser que la borne n'est pas de bonne qualité est l'impact des sources parasites. Ce sont des sources qui accompagnent une source donnée pendant un bout de son chemin puis la quittent. Ces sources ont deux effets contradictoires :

1. La capacité d'augmenter le délai local sur les serveurs qu'elles ont en commun avec les sources de bout en bout ;
2. La modification de ce flux de bout en bout après leur départ qui diminue leur capacité ultérieur à créer des files d'attente.

Le premier point signifie que pour obtenir un délai maximum à un serveur, les sources doivent être synchronisées et suivrent leur trajectoire vorace[2]. Si on applique cette règle au premier serveur, la source qui traverse les deux serveurs ne pourra plus suivre sa trajectoire vorace sur le second serveur (et donc y maximiser son délai). C'est le second point ci dessus.

[2]La trajectoire vorace est telle que la courbe de débit cumulé suit exactement la courbe d'arrivée minimale.

A partir de ce constat, on peut suivre deux démarches. On peut tout d'abord tenter une approche exacte en trouvant pour un réseau donné, la trajectoire des sources qui maximise le délai de bout en bout (cette trajectoire n'est pas forcément la même pour tous les chemins). La seconde approche est une approche par recherche de bornes.
Nous avons suivi la première voie pour le cas de deux serveurs en tandem. Nous avons obtenu de manière analytique le délai maximum de bout en bout dans ce cas de réseau. Les détails sont dans [116]. L'enseignement important de [116] est que la détermination du délai maximum pour un réseau de plus de deux nœuds est très difficile d'un point de vue analytique.
Pour des réseaux plus grands, nous avons utilisé la seconde approche. Nous sommes partis de la **borne additive** obtenue par sommation des bornes sur le délai obtenues indépendamment à chaque serveur à l'aide du théorème 1. Cette borne est utilisable dans les réseaux en arbre où tous les trafics convergent vers la racine de l'arbre. Ce type d'architecture a été proposé dans le cas de réseaux d'opérateurs fondés sur MPLS pour gérer le trafic entre les différentes entrées et sorties du réseau. La figure 2.4 illustre ce type d'architecture et montre son avantage par rapport à une architecture maillée qui nécessite beaucoup plus de liens.

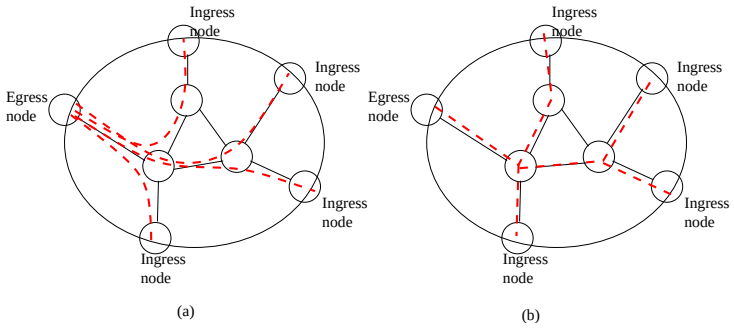


FIG. 2.4 – P2p (a) vs. m2p (b)

Nous présentons ci-après les résultats obtenus pour des réseaux en arbre. Nous résumons ensuite l'approche que nous avons proposée pour le cas de réseaux avec une architecture quelconque.

2.2.3 Topologies en arbres

Notre premier résultat pour l'étude des réseaux en arbre que nous appelons aussi réseaux d'accumulation a été d'identifier une sous-classe de ces réseaux pour lesquels il existe une trajectoire du système qui génère la borne additive. Pour obtenir la condition pour laquelle un réseau d'accumulation est additif, nous partons de la trajectoire vorace du système, c'est-à -dire celle pour laquelle toutes les sources sont voraces et synchrones, i.e. démarrent au même instant. La trajectoire vorace permet de définir de manière univoque et pour chaque serveur, les grandeurs suivantes (voir figure 2.5) :
– $d_{j\ max}$ le délai local maximum au serveur $j \in \{1, n\}$.
– $t_{j\ max}$: l'instant d'arrivée sur le serveur considéré du bit qui va subir $d_{j\ max}$ (voir figure 2.5).

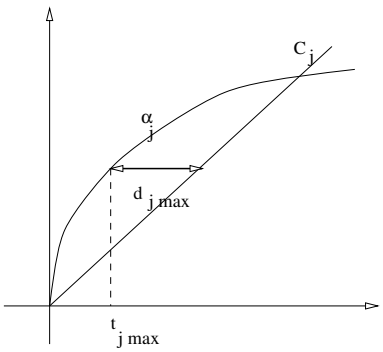


FIG. 2.5 – Paramètres intrinsèques $(t_{j\ max}, d_{j\ max})$

A partir des grandeurs ci-dessus, nous avons établi le résultat suivant :

Théorème 3 *Une condition nécessaire et suffisante pour qu'un réseau d'accumulation soit additif est que pour tout couple* $(i, i+1)$ *de serveurs adjacents,* $t_{imax}+d_{imax} \geq t_{i+1max}$.

La preuve de ce théorème s'obtient en construisant une trajectoire du système qui soit telle que le délai maximum d'un bit (dans un modèle fluide) soit égal à la borne additive. Les détails sont dans le manuscrit de thèse mais l'idée de la démonstration est de moduler l'instant de démarrage des sources sur les différents serveurs. Cela signifie notamment que cette trajectoire pire-cas n'est pas la trajectoire vorace définie ci-dessus.
Nous avons considéré ensuite le cas des réseaux d'accumulation généraux, i.e. qui ne remplissent pas nécessairement le théorème 3. Nous avons considéré deux types de réseaux d'accumulation généraux. Tout d'abord des réseaux "strictement non additifs" dont on est sûr par construction qu'ils vont violer le théorème 3. Nous avons aussi considéré des réseaux dit bien-formés dont le débit des serveurs croît des feuilles vers la racine de l'arbre, une technique raisonnable de dimensionnement dans des cas tels que celui décrit figure 2.4. Pour ces deux types, nous avons montré que la trajectoire pire-cas utilisée dans la démonstration du théorème 3 générait un délai maximum proche de la borne additive. Proche signifie que nos résultats numériques ont montré un ordre de grandeur de 50% au maximum entre les deux quantités pour les réseaux strictement non additifs et de l'ordre de 10% au maximum pour les réseaux bien formés.
Une autre contribution dans le domaine des réseaux d'accumulation a été la détermination et l'étude d'un algorithme d'admission d'appel (distribué ou centralisé) fondé sur la borne additive.

2.2.4 Topologies générales

L'extension des résultats obtenus pour les réseaux d'accumulation pour le cas des réseaux ayant un architecture générale n'est pas possible car la présence des sources parasites rend la borne additive trop grossière. Pour mettre en œuvre une solution de garantie de QoS pour un réseau quelconque, nous avons proposé une méthode basée sur un *shaping* des flux avant à leur entrée dans le réseau (cf. figure 2.6). Le problème de dimensionnement

dans le cœur du réseau est simplifié par le *shaping* car les sources y sont traitées comme des sources à débit constant. Nous proposons que chaque serveur implante une politique de service qui permette de récupérer les ressources inutilisées par les flux demandant de la QoS au profit des flux *best-efforts* (cf. figure 2.7). Il peut s'agir par exemple d'un ordonnanceur GPS avec 2 classes de services (QoS et best-effort). La méthode proposée est simple. L'idée est de faire mieux qu'un dimensionnement naïf au débit crête en réduisant le débit maximum des connexions avant leur entrée dans le réseau. Le débit choisi pour le shaping est la bande passante effective (théorème 2) de la connexion qui est fonction de sa demande de QoS.

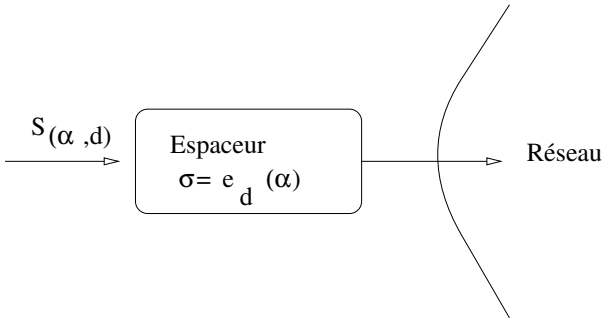


FIG. 2.6 – Utilisation d'un espaceur en entrée du réseau

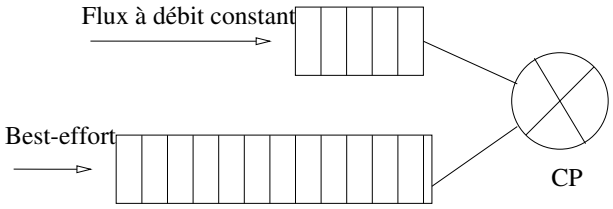


FIG. 2.7 – Politique de service et architecture des serveurs

La validation de notre schéma de garantie de QoS s'est faite par comparaison avec un même réseau GPS de même topologie et sans *shaping* des connexions. Le réseau GPS est *a priori* irréaliste car chaque ordonnanceur doit garder un nombre d'états proportionnel au nombre de connexions actives. Nous ignorons ce problème de scalabilité et nous nous concentrons sur le coût en bande passante en utilisant la seule méthode de dimensionnement des réseaux GPS connue à l'époque : RPPS (Rate Proportional Processor Sharing). RPPS utilise un poids égal au débit moyen des connexions. Nous montrons analytiquement (sous certaines conditions - voir [117] pour les détails) que le besoin global de RPPS en bande passante dans le réseau est supérieur à celui de notre schéma avec shaping en entrée.

2.3 Conclusions

L'objectif de cette thèse était double. Nous voulions étudier des solutions de garanties de service déterministes "scalables" et nous voulions aussi utiliser ce nouvel outil qu'était le Network Calculus. Pour ce qui est du second point, le Network Calculs reste un outil complexe. Si on sort des hypothèses d'utilisation normales des théorèmes, il faut adapter des méthodes souvent peu intuitives. Cela nous a conduit à faire des analyses trajectorielles déterministes plus que du Network Calculus à proprement parlé.
Nos schémas de garanties de service sont sans doute intéressants et applicables à des cas pratiques, que ce soit le schéma pour les réseaux d'accumulation ou le schéma pour les réseaux quelconques. Avec le recul, l'approche de cette thèse (mais aussi dans beaucoup d'autres travaux à l'époque) visant à fournir des garanties déterministes de qualité de service, est sans doute criticable. On peut lui faire les objections suivantes. Tout d'abord, la caractérisation des sources par des paramètres de leaky bucket, bien qu'en apparence simple, est dans la pratique complexe. Cela a conduit des chercheurs comme Jim Roberts à proposer une allocation au débit crête pour les flux temps-réel et à se concentrer sur la qualité de service perçue par les autres flux dits élastiques [17, 82, 88]. Une seconde critique liées aux méthodes déterministes tient dans la surestimation de bande passante qu'elles induisent. Si le Network Calculus permet d'obtenir des bornes non triviales, celles-ci restent souvent des ordres de grandeur au-dessus de celles fournies par les méthodes stochastiques. Notons néanmoins que des efforts existent pour combiner le Network Calculus et les méthodes stochastiques [31].
D'autres éléments ont aussi présider à un certain abandon des garanties déterministes dans l'Internet, notamment l'amélioration des techniques de codage qui demandent des débits de plus en plus faibles et sont de plus en plus tolérantes aux pertes. De plus, et comme signalé dans l'introduction de ce chapitre, le commerce en ligne qui génère des revenus sans cesse croissants, est largement fondé sur la vente de biens (matériel informatique) et de services (vente de billets, réservation de chambre d'hôtel), c'est-à-dire des services non temps-réel. La vidéo à la demande apparaît uniquement au travers des offres "triple-play" où l'utilisateur achète simultanément un accès à Internet, au téléphone fixe et à la télévision, avec parfois en plus de la vidéo à la demande. Mais dans ce cas, ce n'est pas l'Internet qui véhicule le flux vidéo. Il est amené sur des serveurs situés dans les terminaux téléphoniques où finissent les lignes téléphoniques des utilisateurs.
Signalons enfin que l'on voit apparaître d'autres domaines où le Network Calculus pourrait être un outil précieux, comme le milieu aéronautique, qui pour faire baisser ses coûts, essaie d'utiliser les technologies Ethernet dans les avions [60, 56]. Ethernet est une technologie intéressante car elle offre de hauts débits et des coûts faibles, mais elle n'offre aucune garantie de service alors que des temps de réponse bornés sont nécessaires dans le milieu aéronautique.

Chapitre 3

Une Nouvelle Approche pour la Qualité de Service : l'Amélioration du Service Best-Effort

3.1 Introduction

Les résultats de ce chapitre ont été obtenus dans le cadre de la thèse d'Idris Rai que j'ai co-encadrée avec Ernst Biersack. Cette thèse a été soutenue le 15 septembre 2004. Les résultats seront donnés sans démonstration - se reporter à [98] pour les preuves.
L'approche adoptée dans ce travail de thèse s'appuie sur l'ensemble des connaissances acquises au travers des études de trafic effectuées depuis la fin des années 90 . Ces études ont montré que :
– Le trafic véhiculé par l'Internet a une forte variabilité [42] ;
– Les goulots d'étranglement sont en général proches des bords du réseaux, i.e. à quelques sauts au niveau IP [67] car les ISPs de cœur ont des réseaux très peu chargés [55].
Cela a motivé notre choix de travailler sur des solutions qui améliorent le service best-effort au sens des performances perçues par les utilisateurs en nous concentrant sur ces goulots d'étranglement. La solution que nous proposons permet d'avoir des taux de pertes et des temps de transfert la plupart du temps inférieurs à ceux fournis par l'Internet actuel, de maintenir l'équité entre les flots et/ou d'offrir de la différentiation de service.

3.1.1 Caractéristiques du trafic Internet

Les mesures sur l'Internet ont montré que le trafic est constitué de nombreux flots[1] très courts de 10 ou 20 paquets alors que 1% des flots les plus longs transportent 50% des octets. Pour parler de ce phénomène, on parle souvent de souris et d'éléphants. Les souris sont

[1]Le flot est une unité de mesure intermédaire entre le niveau paquet et le niveau connexion. Un flot est défini comme un ensemble de paquets partageant les mêmes attributs (port source, port destination, IP source, IP destination, numéro de protocoles) qui sont émis proches les uns des autres par la source. Typiquement, on ne veut pas qu'il y ait un écart supérieur à 10 secondes entre 2 paquets. L'intuition derrière un flot est que ces paquets vont traverser le réseau en subissant un traitement similaire.

en général des transferts Web alors que la plupart des éléphants sont dus aux applications pair-à -pair [115].
De nombreuses distributions ont été proposées pour modéliser le trafic de l'Internet : Pareto, lognormal, hyper-exponentielle, Weilbull [42], Gaussienne inversée, etc., avec toujours des coefficient de variations (C) plus grands que 1 [7, 42, 53]. Le coefficient de variation est défini comme le rapport entre l'écart type et la moyenne d'une loi. C est une métrique usuelle pour mesurer la variabilité d'une distribution (plus C est grand, plus la variabilité est grande).
De nombreuses études ont tenté de tirer partie de la grande variabilité des distributions de l'Internet pour améliorer ses performances. Par exemple, [63, 41, 40] ont proposé des systèmes de répartition de charge dans des systèmes distribués ; [104, 52] ont proposé des modifications au niveau du routage ou de la commutation ; [62, 38] de nouvelles politiques de services pour le Web. Nous montrons dans la section suivante que si la forte de variabilité du trafic Internet n'est pas prise en compte, les performances en patissent.

3.1.2 Impact de TCP et FIFO sur les performances de petits flots

Le trafic Web, qui constitue une large fraction des flots courts dans l'Internet, est transféré au moyen du protocole de transport TCP. Une étude attentive de TCP souligne les facteurs qui affectent les performances des flots courts : indépendamment de la bande passante du réseau, TCP va de manière préventive, initialiser systématiquement sa fenêtre de congestion à un paquet (notons que ce chiffre dépend en fait du système d'exploitation. Il peut aller jsuqu'à 3 dans certains systèmes Linux) et doubler ensuite pour chaque ACK reçu jusqu'au moment où il détecte une perte (phase de *slow-start*). Ainsi, même si le chemin choisi à suffisamment de ressource, la durée du transfert dépend fortement du RTT (*round trip time* ou temps d'aller-retour) de la connexion. Les flots courts TCP souffrent aussi de problèmes de performances en terme de reprise sur erreur puisqu'ils n'ont souvent pas assez de paquets à émettre pour un *fast-retransmit* [10]. Au contraire, ils doivent faire une reprise sur timer qui prolonge de fait le temps total de transmission et ce d'autant plus que le timer de retransmission est initialisé avec des valeurs en général très grandes (3 secondes). Ainsi, perdre des paquets dans une phase de slow-start pour un flot court est très pénalisant.
FIFO (First In First Out) est la discipline de service utilisée couramment dans l'Internet. La théorie des files d'attente nous enseigne que FIFO favorise les clients longs et pénalise les clients courts car un client arrivant dans une file d'attente doit attendre le service complet de tous les clients devant lui avant d'être servi. Un phénomène similaire est observable dans un réseau de paquets où l'entité atomique est le flot (et non le client - voir section 3.3) [54]. De plus, les files dans les routeurs de l'Internet sont gérées suivant la politique *drop-tail* qui ne différencie pas les flots longs des flots courts. Ainsi, un flot court est défavorisé par l'utilisation de FIFO/*drop-tail* alors qu'il est déjà défavorisé avec TCP. Puisque ces flots courts constituent la majorité des flots de l'Internet, on voit qu'une amélioration de leur service aura un impact direct sur les performances perçues par l'utilisateur.
L'objectif de ce travail a été de trouver une politique de service alternative à FIFO

qui soit utilisable dans les réseaux de paquets et améliore significativement le temps de réponse des flots courts sans trop pénaliser les flots longs. Les politiques de services qui discriminent suivant la taille des clients sont des politiques dites fondées sur la taille.

3.2 LAS et les Politiques de Services basées sur la Taille

Les politiques de service fondées sur la taille des clients ont été étudiées dans les années 1960-1970 pour une possible application au domaine des ordinateurs [76, 33, 36]. L'idée était que ces politiques pouvait aider à favoriser les flux interactifs au détriment des flux *batch* (calculs longs). Deux politiques ont été particulièrement étudiées : SRPT (Shortest Remaining Processing Time) et LAS (Least Attained Service). La première repose sur le temps de service restant à un client tandis que la seconde repose sur le temps de service déjà reçu pour fixer les priorités. SRPT a été proposée pour des cas tels que des serveurs Web [64] où il est possible d'estimer le temps de service résiduel d'un client. LAS présente en revanche l'avantage de pouvoir être utilisé dans les routeurs de l'Internet[2]. Nous nous sommes concentrés sur LAS dans nos études. Bien évidemment, il y a un prix à payer pour utiliser LAS plutôt que SRPT. SRPT minimise le temps moyen de réponse (et donc le nombre moyen de clients actifs de par là loi de Little) parmi toutes les politiques de service possibles car SRPT a une connaissance exacte du temps de service restant. Ce n'est pas le cas de LAS. Il a en revanche été montré que dans le cas où la distribution des temps de service était fortement variable [3], le temps moyen de réponse de LAS est inférieur à celui d'autres politiques telle FIFO. La question qui vient alors naturellement à l'esprit est l'impact de cette diminution du temps moyen sur les grands clients. En se fondant sur la loi de conservation de Kleinrock [75] (valable pour des arrivées poissoniennes), on s'attend à ce que la diminution du temps de service des petits clients induise mécaniquement des temps de service très longs pour les grands clients comparés à d'autres politiques telles Processor Sharing (PS). Il faut tout d'abord noter que la loi de conservation ne s'applique pas à LAS car c'est une politique qui utilise des informations sur les temps de service pour prendre ses décisions. Dans [24], chapitre 3, proposition 3.1., P. Brown démontre même que pour le cas de processus de services de Pareto avec variance infinie, le temps de réponse de tous les clients est inférieur lorsque LAS est utilisé plutôt que PS. Plus généralement, il faut retenir que les temps de réponse moyens et individuels obtenus avec LAS comparés à ceux d'autres politiques comme PS ou FIFO vont dépendre de la distribution ou de certaines de ses caractéristiques comme son coefficient de variation. Notons que les distributions que nous utiliserons dans la suite de ce chapitre sont à moyenne et à variance finis.
Nous donnons maintenant une description plus précise de LAS. En théorie des files d'at-

[2]Même si LAS n'a pas à avoir connaissance de la fin d'un flot ou d'une connexion, il faut néanmoins garder des états par connexions ce qui devient difficile si le nombre de flots simultanés devient grand. Cela étant, si on suppose que les goulots d'étranglement sont en majorité situés en des points proches des bords du réseau, on peut espérer que le nombre de connexions à gérer reste faible

[3]Typiquement, une distribution avec un coefficient de variation $C >> 1$. Voir par exemple la figure 3.3

tente, LAS est une politique de service préemptive multi-niveaux qui donne la priorité au client dans le système qui a reçu le moins de service jusqu'à présent. Dans le cas d'égalité, les clients partagent le serveur en mode processor sharing (PS) [76]. Un client nouvellement arrivé préempte toujours le (ou les) client(s) en service et est servi jusqu'à son départ, une nouvelle arrivée ou jusqu'au moment où il a obtenu un service égal à celui des clients qu'il a préempté. Une implantation de LAS requiert la connaissance de la quantité de service reçue par les clients. LAS est aussi connue dans la littérature sous les noms de *foreground-background* (FB) [34, 76] ou *shortest elapsed time* (SET) first scheduling [33].

3.3 Études niveau paquets et niveau clients

Dans ce travail de thèse, nous évaluons l'utilisation de LAS dans des réseaux paquets pour réduire le temps de transfert des flots courts et réduire le temps de réponse global perçu par l'utilisateur. Dans les réseaux paquets, la métrique à considérer est la performance d'un flot de paquets, qui est une entité différente d'un client dans une file d'attente. Le terme de client est communément utilisé en théorie des files d'attente pour désigner une quantité de travail qui arrive instantanément dans le système. Avec cette définition, il est clair qu'un flot de paquets ne peut être assimilé à un client dans une file d'attente. Au contraire, une source dans un réseau paquet transmet un flot de paquets espacés dans le temps qui se trouvent multiplexés avec les paquets d'autres flots. De plus, lorsque TCP est utilisé, l'envoi des paquets est fonction de l'historique (récent) de la connexion. Pour toutes ses raisons, il est clair que les modèles analytiques de LAS conçus pour les files d'attente ne peuvent pas s'appliquer *a priori* tels quels dans les réseaux de paquets.

Dans le cas des flots, le prochain paquet servi par LAS est celui appartenant au flot qui a émis le moins d'octets jusqu'à présent. Par définition, LAS octroie toute la bande passante à un nouveau flot jusqu'à ce que celui-ci soit préempté ou ait reçu une quantité de service égale à celle des flots qu'il a préemptés. Si plusieurs flots ont reçu une quantité de service équivalente, ils partagent équitablement le serveur (politique *round-robin*).

Nous présentons ci-après un choix de contributions issues du travail de thèse d'Idris. Nous présentons d'abord des résultats analytiques obtenus au niveau client qui permettent de comparer LAS, SPRT et la classe des disciplines NPP (Non Preemptive Policy) qui englobe FIFO. Nous présentons ensuite des résultats obtenus dans des réseaux de paquets, en mettant l'accent sur l'interaction avec TCP. Enfin, nous présenterons des extensions de LAS qui peuvent être utilisées lorsque l'on veut différencier le service donné à une classe de connexions données. Bien que cela puisse être vu comme un retour caché des concepts de QoS décriés précédemment, l'approche est ici plus pragmatique et ne sous-entend pas un déploiement ubiquitaire. Nous nous plaçons dans une approche " à la Jim Roberts" [16, 99, 87, 20] où on définit deux classes : les flux temps réels et les flux élastiques ; et l'on veut protéger les premiers tout en gardant les bonnes propriétés de LAS.

3.4 LAS au niveau client

3.4.1 Temps de réponse

Soit λ le taux d'arrivée des clients. On considère une distribution X avec une densité $f(x)$. l'abréviation c.m.f.v.f signifie dans la suite continue, à moyenne finie et à variance finie. étant donnée une fonction de répartition $F(x) \triangleq \int_0^x f(t)dt$, on dénote par $F^c(x) \triangleq 1-F(x)$ le complémentaire de cette fonction.
On définit $m_n(x)$ comme $m_n(x) \triangleq \int_0^x t^n f(t)dt$. Ainsi $m_1 \triangleq m_1(\infty)$ est la moyenne et $m_2 \triangleq m_2(\infty)$ le second moment de la distribution considérée. La charge des clients de taille inférieur ou égal à x est donnée par $\rho(x) \triangleq \lambda \int_0^x tf(t)dt$, et $\rho \triangleq \rho(\infty)$ est la charge totale du système.
On considère une file M/G/1 dans cette section, où G est une distribution c.m.f.v.f et M signifie des arrivées poissonnienes. Un processus de Poisson fournit une modélisation acceptable du processus d'arrivée des flots dans l'Internet [15]. L'expression du temps de réponse conditionnel des clients de taille x pour la politique de service M/G/1/SRPT [103] est :

$$T(x)_{SRPT} = \frac{\lambda(m_2(x) + x^2F^c(x))}{2(1-\rho(x))^2} + \int_0^x \frac{1}{1-\rho(t)}d(t)$$

Le temps moyen conditionnel de réponse $T(x)_{LAS}$ pour LAS est donné par [102],

$$T(x)_{LAS} = \frac{\lambda(m_2(x) + x^2F^c(x))}{2(1-\rho(x)-\lambda xF^c(x))^2} + \frac{x}{1-\rho(x)-\lambda xF^c(x)} \tag{3.1}$$

Finalement, les formules de $T(x)$ pour des files M/G/1/PS et M/G/1/NPP, sont [36] :

$$T(x)_{PS} = \frac{x}{1-\rho} \tag{3.2}$$

$$T(x)_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + x \tag{3.3}$$

Les politiques NPP incluent FIFO, *last-in first-out* (LIFO), RANDOM, etc. On définit le temps de réponse conditionnel normalisé par $S(x) = \frac{T(x)}{x}$. La définition du temps de réponse normalisé $S(x)$ montre que pour deux politiques A et B, le ratio $\frac{T(x)_A}{T(x)_B} = \frac{T(x)_A/x}{T(x)_B/x} = \frac{S(x)_A}{S(x)_B}$. Le temps de réponse normalisé est une métrique très importante pour comparer des politiques de service entre elles. Il permet de mesurer le degré d'équité dans le traitement de clients de différentes tailles. Par exemple, pour le cas de PS, le temps de réponse normalisé est indépendant de la taille du client, ce qui signifie que cette politique est particulièrement équitable. PS sert donc souvent d'étalon pour mesurer l'équité d'autres politiques et nous l'utiliserons souvent dans ce sens.

3.4.2 Choix de la distribution des tailles des clients

Nous allons analyser LAS pour des distributions de taille de clients c.m.f.v.f. spécifiques. Le choix d'une distribution est motivé par les caractéristiques du trafic Internet où *la plupart des flots sont courts et plus de la moitié de la charge est due à une très faible fraction des flots les plus grands*. Plusieurs distributions ont été utilisées pour modéliser le trafic Internet avec toujours un coefficient de variation plus grand que 1 [7, 42, 53]. Parmi ces distributions, on peut citer : la distribution de Pareto, la distribution de Pareto bornée, la distribution lognormale, la distribution hyper-exponentielle, la distribution Gaussienne inversée et des mélanges de lognormal et Pareto. Soit X la taille des clients. La densité des clients pour une distribution de Pareto est :

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad x \geq k, 0 \leq \alpha \leq 2 \tag{3.4}$$

où k est la taille minimale des clients et α l'exposant de la loi de puissance.
Nous avons utilisé des variantes de la distribution de Pareto, et notamment la distribution de Pareto bornée.
Celle-ci est souvent utilisée car elle peut avoir une grande variance et modélise bien les distributions observées sur l'Internet. De plus la taille maximale bornée des flots permet de rendre compte de la taille maximale des flots observés [125, 38, 12]. Nous la notons $BP(k, P, \alpha)$, où k et P sont les tailles minimales et maximales des clients et α l'exposant de la loi de puissance. La densité d'une loi de Pareto est donnée par :

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/P)^{\alpha}} x^{-\alpha-1}, \qquad k \leq x \leq P,\ 0 \leq \alpha \leq 2 \tag{3.5}$$

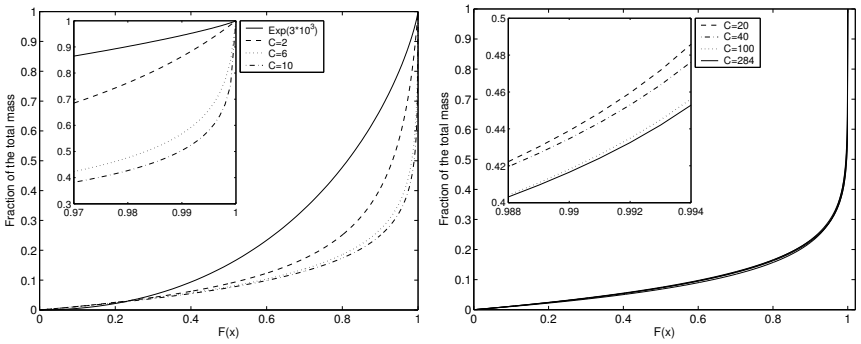
Le n$^{\text{ième}}$ moment m_n pour une loi $BP(k, P, \alpha)$ est donné par :

$$m_n = \frac{\alpha}{(n-\alpha)(P^{\alpha}-k^{\alpha})}(P^n k^{\alpha} - k^n P^{\alpha})$$

Ainsi, l'expression du coefficient de variation d'une loi $BP(k, P, \alpha)$ est donné par $C \triangleq \frac{\sqrt{m_2 - m_1^2}}{m_1}$. Pour obtenir différentes valeurs de C pour une loi $BP(k, P, \alpha)$, on ajuste un ou plusieurs paramètres de la distribution, i.e., k, α, ou P.
Nous considérons également le cas des distributions exponentielles des tailles de clients pour évaluer LAS dans le cas d'une file M/M/1. Pour une loi exponentielle, le coefficient de variation est 1. On dénote la loi exponentielle avec une moyenne de $1/\mu$ par $Exp(1/\mu)$. La densité de la distribution exponentielle est donnée par :

$$f(x)_{Exp} = \mu e^{-\mu x}, \qquad x \geq 0,\ \mu \geq 0$$

La variabilité de la distribution peut être estimée à partir de la fonction de masse pondérée $(M_w(x))$ [39], qui (pour un client de taille x) est définie comme la fraction de la masse constituée par les clients de taille inférieure ou égale à x : $M_w(x) \triangleq \frac{\rho(x)}{\rho}$.
La Figure 3.1 représente la fonction de masse pondérée pour des distributions BP avec différentes valeurs de C et pour la distribution exponentielle, toutes avec une même moyennes de 3×10^3. La figure 3.1(a) montre que pour la distribution exponentielle,



(a) *Exp*(3×10^3) *and BP with* $C = 2, 6, 10$

(b) *BP with* $C = 20, 40, 100, 284$

FIG. 3.1 – *Fonction de masse pondérée pour des lois* $BP(k, P, \alpha)$ *et* $Exp(3 \times 10^3)$.

1% des clients les plus gros constituent environ 5% de la masse totale. Au contraire, pour une loi BP, on observe que 1% des plus gros clients concentrent une masse qui croît avec C de 15% de la masse totale pour $C = 2$ à près de 50% pour $C = 10$. Par ailleurs, la figure 3.1(b) montre que les lois BP avec $C \geq 20$ exhibent toutes une forte variabilité au sens où 50% de la masse totale est constitué par moins de 1% des plus grands clients.

3.4.3 LAS vs. PS et FIFO

Nous avons comparé LAS avec plusieurs autres politiques de services incluant PS, SPRT et FIFO pour une distribution des durées des clients BP ou exponentielle. Nous présentons dans cette section ces résultats dans le cas de FIFO et PS.

Distribution générale

Pour une distribution générale, nous avons établi le théorème 4 et son corollaire 1 qui montrent la relation entre les temps de réponse moyens normalisés de LAS et PS :

Théorème 4 *Pour toute distribution c.m.f.v.f et toute charge* $\rho < 1$,

$$S_{LAS} \leq \frac{2 - \rho}{2(1 - \rho)} S_{PS} \tag{3.6}$$

Corollaire 1 *Pour toute distribution c.m.f.v.f et toute charge* $\rho < 1$,

$$T_{LAS} \leq \frac{2 - \rho}{2(1 - \rho)} T_{PS} \tag{3.7}$$

La figure 3.2 illustre le théorème 3. On peut voir que le ratio entre les temps de réponse normalisés est inférieur à 2 pour une charge $\rho \leq 0.66$ et inférieur à 6 pour $\rho \leq 0.9$. Ce résultat indique que pour des charges modérées, le temps de réponse normalisé de LAS reste proche de celui de PS. On peut donc espérer que la pénalité subie par les clients les

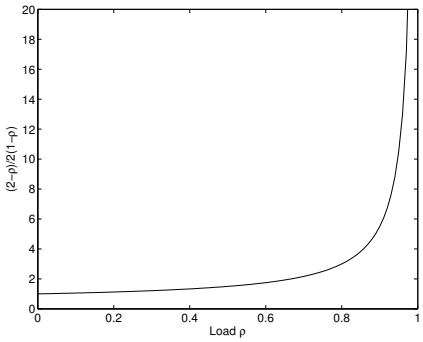


FIG. 3.2 – *Borne supérieure du* $\frac{S_{LAS}}{S_{PS}}$.

plus longs sous LAS ne soit pas trop élevée (ce qui se traduirait par un temps de réponse normalisé moyen élevé). A noter qu'un résultat plus puissant a été prouvé par A. Wieman et al. [122], à savoir les conditions exactes sous lesquelles le temps de réponse moyen de LAS était inférieur à celui de PS. Le paramètre clef est la fonction $\mu(x) = \frac{f(x)}{F(x)}$ qui doit être décroissante.

Nous avons aussi déterminé une borne supérieure qui compare le temps de réponse moyen de LAS avec la famille des lois non préemptives NPP. Cette borne est fonction du coefficient de variation et de la charge.

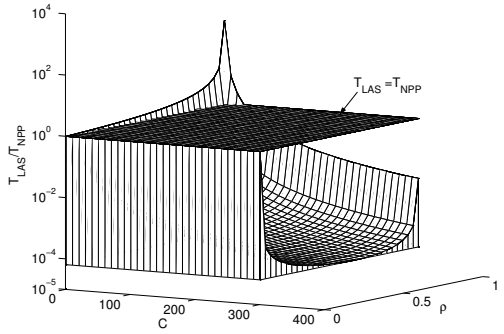
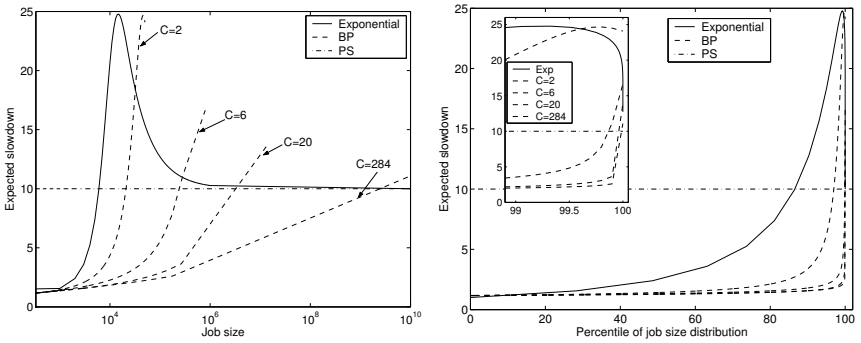


FIG. 3.3 – *Borne supérieure sur* $\frac{T_{LAS}}{T_{NPP}}$ *comme une fonction de la charge* ρ *et du coefficient de variation* C.

La figure 3.3 représente cette borne sur le ratio du temps de réponse de LAS au temps de réponse des politiques NPP pour $C \geq 1$ et $\rho < 1$. On observe que LAS a un temps de réponse moyen plus élevé que les politiques NPP seulement dans le cas où C est proche de 1. Dans ce cas, on observe que le ratio augmente avec la charge. A l'opposé, le temps de réponse moyen de LAS est inférieur à celui des lois NPP pour toute distribution avec un coefficient de distribution supérieur à 1 et pour toute charge $\rho < 1$. De manière générale, on observe que pour une charge ρ donnée, le ratio $\frac{T_{LAS}}{T_{NPP}}$ décroît avec C.

Distributions spécifiques

Nous présentons maintenant des résultats obtenus pour des distributions BP avec différentes valeurs de C et pour la distribution exponentielle. Un client est dit traité inéquitablement si son temps moyen de réponse conditionnel avec une politique de service donnée (par exemple LAS) est inférieur à ce même temps moyen conditionnel avec PS. Nous étudions ici l'inéquité de LAS pour des distributions avec différentes valeurs de C. Nous utilisons la distribution exponentielle ($C = 1$) et des distributions BP avec des valeurs de C de 2, 6, 20, et 284. Toutes les distributions ont la même moyenne de 3×10^3. Pour obtenir des distributions BP avec différentes valeurs de C et une même moyenne, on fait varier α et p.



(a) *Taille de clients*

(b) *Quantile de la distribution des tailles de clients.*

FIG. 3.4 – *Temps moyen de réponse normalisé sous LAS et sous PS pour différentes valeurs de C.*

Les figure 3.4(a) et 3.4(b) représentent les temps moyens de réponse conditionnels en fonction des quantiles de la distribution des tailles de clients. On voit, à partir de ces deux figures, que le pourcentage des flots les plus gros qui sont pénalisés avec LAS (i.e. ont un temps de réponse normalisé supérieur avec LAS qu'avec PS) ainsi que le niveau de la pénalité décroissent lorsque C croît. Pour la distribution BP avec $C \geq 6$, on observe que moins de 0.5% des flots les plus longs souffrent d'une pénalité et que la différence de performance entre $C = 20$ et $C = 284$ est minime.
Dans le cas de la distribution exponentielle, on peut obtenir des résultats plus précis et notamment le théorème suivant :

Théorème 5 *Pour une distribution exponentielle et une charge $\rho < 1$,*

$$\begin{aligned} S_{LAS} &\leq S_{PS} \\ T_{LAS} &= T_{PS} \end{aligned} \tag{3.8}$$

Notons que LAS fonctionne mieux pour des distributions avec des valeurs de C élevées. La distribution exponentielle, elle, a un coefficient de variation égal à 1, mais, malgré cela, le théorème 5 montre que les performances moyennes de LAS restent meilleures que celles de PS.

3.4.4 LAS et SRPT

SRPT est une politique optimale au sens où elle minimise le temps de réponse moyen. Il est donc important de comparer LAS à SRPT pour évaluer de combien elles diffèrent et voir si la différence est fonction de la variabilité de la loi. Nous avons tout d'abord comparé le temps moyen conditionnel de réponse de LAS et SRPT :

Théorème 6 *Soit* $\phi(x) \triangleq \rho(x) + x\lambda F^c(x)$. *Alors, pour toute distribution c.m.f.v.f et pour toute charge* $\rho < 1$,

$$T(x)_{SRPT} \leq T(x)_{LAS} \leq \Big(\frac{1-\rho(x)}{1-\phi(x)}\Big)^2 T(x)_{SRPT} \tag{3.9}$$

Les résultats numériques déduits du théorème précédent montre que LAS offre un temps de réponse similaire à SRPT quand la distribution de la taille des clients est fortement variable. Pour la distribution exponentielle, LAS offre des temps de réponse notablement plus élevés que SRPT.
Nous avons aussi analysé LAS en surcharge et comparé les résultats obtenus avec ceux de SRPT. Nous avons tout d'abord prouvé que LAS peut servir des clients même en cas de surcharge ; puis, nous avons déterminé le temps de réponse conditionnel moyen sous LAS en surcharge :

$$T(x)_{\text{LAS}} = \begin{cases} \frac{\lambda(m_2(x)+x^2(1-F(x)))}{2(1-\rho(x)-\lambda x(1-F(x)))^2} + \frac{x}{1-\rho(x)-\lambda x(1-F(x))} & \text{si } x < x_{\text{LAS}}(\lambda) \\ +\infty & \text{si } x \geq x_{\text{LAS}}(\lambda) \end{cases}$$

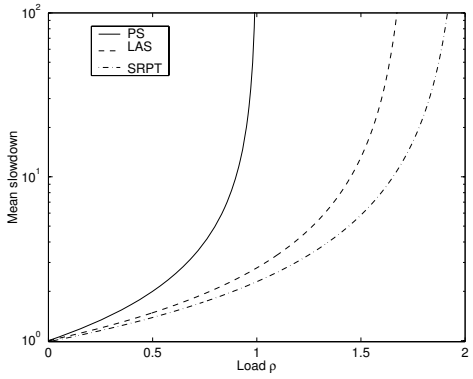


FIG. 3.5 – *Temps moyen de réponse normalize pour le 99ème percentile pour la distribution* $BP(322, 10^{10}, 1.1)$ *en fonction de la charge.*

Pour rappel, le temps de réponse de PS est non borné lorsque $\rho > 1$. Ce n'est pas le cas pour SRPT comme prouvé dans [12]. La figure 3.5 montre qu'au contraire de PS, LAS et SRPT sont stables en surcharge pour certaines tailles de clients. On observe également que LAS devient instable avant SRPT, ce qui était prévisible car SRPT possède des informations précises sur le futur d'un temps de service alors que LAS ne peut que l'estimer à partir du passé.

3.5 LAS dans les réseaux paquet

Note thèse est que LAS pourrait être très utile dans les réseaux paquets, et notamment dans l'Internet où TCP domine. Dans cette section, nous étudions l'interaction de LAS avec TCP pour examiner en détail les performances au niveau flot avec LAS. A l'aide de simulations, nous montrons que LAS, comparé à FIFO, limite les pertes de paquets et réduit significativement les temps de réponse des flots courts au prix d'une pénalité négligeable pour les flots longs. Nous montrons également que le gain en performance avec LAS est dû principalement à la façon dont LAS interagit avec TCP dans la phase de slow-start (SS) en réduisant le RTT et en minimisant les pertes. Nous étudions également les performances des flots TCP et UDP illimités (ou extrêmement longs) en comparant les résultats obtenus avec ceux de FIFO.

3.5.1 Réseaux à goulot d'étranglement simple

Dans les réseaux paquets, la politique LAS interagit bien avec les flots TCP courts car elle insère les premiers paquets d'un flot court en tête de buffer. Le RTT des flots courts se trouve ainsi réduit, ce qui réduit directement le temps de réponse de ces flots. De plus, avec LAS, un paquet qui arrive au serveur est d'abord inséré dans la file[4], puis le paquet qui a la plus basse priorité est éliminé. Cela limite les pertes pour les flots courts. Les flots qui subissent des pertes sont les flots longs *a priori*. Or ceux-ci ayant suffisamment de paquets à transmettre sont en général capables de récupérer leurs pertes à l'aide d'un fast retransmit qui n'affecte que modérément le débit de la connexion par opposition à une expiration de timer. Au contraire, les flots courts ont plus tendance à récupérer leurs pertes par timer, ce qui affecte d'autant leurs performances. La figure 3.6 montre par exemple le taux de perte de paquets moyen en fonction de la taille des flots. On voit clairement que pour cette simulation, les flots de moins de 40 paquets ne subissent aucune perte avec LAS alors qu'ils subissent beaucoup de perte avec FIFO.

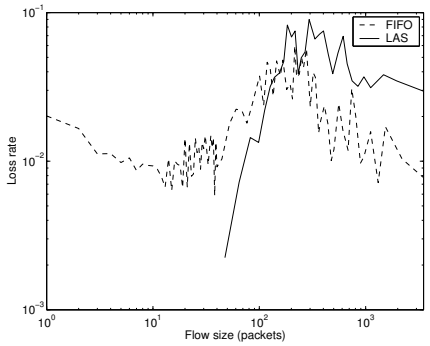


FIG. 3.6 – *Taux de perte de paquet avec une distribution de Pareto des tailles de flots (Web) pour* $\rho = 0.73$

[4]Notons ici que dans un réseau paquet, LAS devient pour nous une discipline de service mais aussi une discipline de gestion des tampons. Le problème de la gestion du tampon dans les modèles analytiques précédents ne se posait pas car nous supposions des tampons infinis.

La figure 3.6 montre ainsi que LAS réduit le taux de perte pour les flots courts et multiplie approximativement par deux le taux de perte des flots longs par rapport à FIFO. Il faut néanmoins garder à l'esprit que les flots longs ne représentent qu'1% des flots du trafic. En résumé, LAS offre un gain de performance très important pour les flots courts dans un réseau paquet.
De manière similaire, le tableau 3.1 montre la moyenne et la variance du débit d'un transfert FTP très long (illimité) sous LAS et sous FIFO pour différentes valeurs de charge. Les résultats de simulation du tableau 3.1 sont obtenus en calculant le débit obtenu par le flot (en paquets/s) pour des fenêtres de 100 secondes sur lesquelles on calcule ensuite la moyenne et la variance de processus. Pour une charge de 0.75, LAS se comporte de manière similaire à FIFO. Ce n'est que pour des charges très élevées, au dessus de 0.90 que les performances du transfert FTP se dégradent notablement par rapport à FIFO. Notons également que la variance augmente de manière très importante pour des charges élevées car avec LAS car il y a des périodes où le flot FTP ne reçoit aucun service (le flot FTP est préempté par les flots plus courts).

	LAS		FIFO	
Charge ρ	moyenne	variance	moyenne	variance
0.75	157.8	1.03	156.30	1.31
0.92	33.93	2.27	97.55	1.01
0.98	4.82	151.35	11.86	43.68

TAB. 3.1 – *Moyenne et Variance du débit (en paquets/s) avec LAS et FIFO.*

3.5.2 Réseaux pathologiques

Bien que TCP vise à un partage équitable des ressources entre flots, de nombreux cas (ou réseaux) pathologiques ont été identifiés pour lesquels ce partage ne se fait de manière convenable. Nous avons étudié les performances de LAS pour ces réseaux TCP pathologiques. Nous avons considéré les cas suivants : les réseaux où les connexions ont des temps de propagation très différents, les réseaux où les protocoles de transport sont hétérogènes (TCP et UDP), et les réseaux avec des goulots d'étranglement multiples. Notons que ces cas pathologiques sont légions dans l'Internet où les connexions ont des RTTs différents et où le trafic UDP est toujours présent, même s'il ne représente en général que quelques pourcents du trafic. Les travaux précédents sur ces problèmes ont conclu en des faiblesses de TCP et ont donc proposé certaines modifications de TCP. Nous avons montré que LAS élimine ces effets indésirables :

– Les flots TCP ayant des RTTs différents obtiennent le même débit moyen avec LAS ;
– Les flots UDP ne peuvent plus "tuer" les flots TCP avec LAS. Au contraire, chaque flot obtient une même fraction du débit disponible.
– Les flots TCP qui traversent plusieurs routeurs successifs où ils sont compétitions avec des flots parasites ne traversant qu'un seul routeur (voir figure 3.7) reçoivent le même débit que ces flots parasites avec LAS.

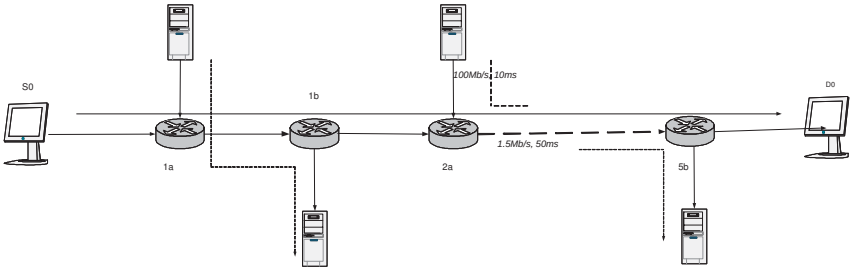


FIG. 3.7 – *Topologie de réseau simulée.*

3.5.3 Modèle analytique de LAS dans des réseaux de paquets

Bien que nous ayions souligné dans la section 3.3 que les modèles de file d'attente s'appliquent au niveau client et non au niveau flot, nous montrons ici que LAS est un cas particulier où l'on peut adapter le modèle dedev niveau client pour en faire un modèle au niveau paquet. Supposons que chaque flot ait au moins un paquet dans la file d'attente du goulot d'étranglement la quasi-totalité du temps. Alors, le temps moyen de service du client de taille x avec LAS exprimé en unité de temps de transfert d'un paquet au niveau du goulot d'étranglement peut se modéliser par le temps de réponse d'un client de taille x dans un file d'attente gérée avec LAS, c'est-à -dire [76] :

$$T(x)_{LAS} = \frac{W_o(x) + x}{(1 - \rho_x)}, \tag{3.10}$$

où $W_o(x)$ est le nombre moyen de paquets en attente avec un numéro de séquence inférieur ou égal à x à un instant aléatoire :

$$W_o(x) = \frac{\lambda \overline{x_x{}^2}}{2(1 - \rho_x)}.$$

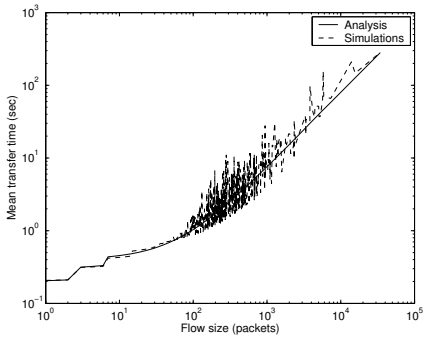


FIG. 3.8 – *Validation du modèle paquet de LAS, avec* $\rho = 0.73$ *et un taux de perte* $= 1.2\%$.

Bien que chaque flot n'ait pas nécessairement un paquet dans la file d'attente tout le temps, comme le demande le modèle ci-dessus pour être exact, un paquet qui arrive un peu plus tard que prévu (dans le modèle de file d'attente) est tout de même capable de

récupérer son retard car sa priorité plus grande le fait passer devant les paquets situés devant lui.
La figure 3.8 représente le temps moyen conditionnel de transfert en fonction de la taille du flot obtenu par le modèle analytique ainsi que par simulation avec des flots TCP en ns-2. La figure 3.8 montre que le modèle paquet de LAS est très précis pour quasiment toutes les tailles de flots.

3.6 Politiques LAS différenciées

Les travaux présentés ci-après ont été effectués dans le cadre d'une collaboration avec Mary Vernon (Wisconsin University).

3.6.1 Introduction

Avec LAS, la priorité des paquets d'un flot décroît avec le nombre de paquets envoyés. Dans cette section nous considérons 3 politiques de services fondées sur LAS capables de fournir un service différencié en fonction de la classe d'un flot. Ces politiques reposent sur LAS car on désire garder les bonnes caractéristiques de LAS (temps de réponse faible pour les petits flots, etc.). Les politiques de service proposées fonctionnent avec deux classes : la classe des flots prioritaires et la classe des flots ordinaires. l'objectif de ces politiques de service différenciées est de réduire le temps moyen de réponse des flots de la classe prioritaire sans trop affecter le temps de réponse des flots ordinaires et c'est pour cela que ces politiques sont déduites de LAS. Ces trois politiques sont dénommées LAS-fixed(k), LAS-linear (k) and LAS-log (k), car la priorité des paquets des flots de la classe prioritaire décroît de façon fixe, linéaire ou logarithmique.
De manière plus formelle, avec les politiques LAS à deux classes, les flots ordinaires ont une priorité à un instant donné égale au nombre de paquets envoyés (en supposant que tous les paquets ont la même taille - on pourrait aussi raisonner au niveau octets, mais la présentation au niveau paquet est plus intuitive). Au contraire, les flots prioritaires voient leur priorité croître suivant la fonction $P(x)$ définie ci-après :

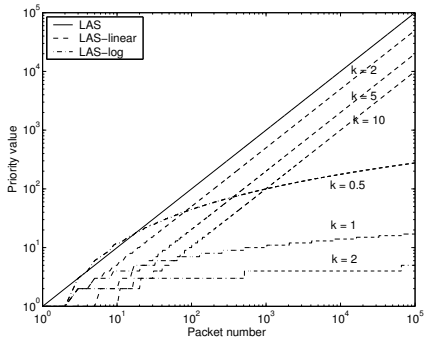


FIG. 3.9 – *Priority value as a function of attained service.*

– *LAS-fixed(k)* : le $x$ième paquet de la classe 1 a une priorité $P(x) = k$, avec $k > 0$ fixe.

– *LAS-linear (k)* : le $x$ième paquet de la classe 1 a une priorité $P(x) = x/k$, avec $k > 0$ fixe.
– *LAS-log (k)* : le $x$ième paquet de la classe 1 a une priorité $P(x) = (\log_2(x))^{\frac{1}{k}}$, avec $k > 0$ fixe.

On peut observer que dans le cas de LAS, $P(x) = x$ pour le paquet numéro x. Pour chaque politique, la fonction $P(x)$ détermine la priorité relative de chaque paquet pour un flot donné. Puisque $P(x) \leq x$ pour LAS-fixed(k) (pour $x \geq k$), LAS-linear (k) and LAS-log (k), on comprend qu'un flot prioritaire et de taille x pour l'une quelconque de ces politiques interfère avec les flots ordinaires de taille inférieure à x. En conséquence, leur temps de réponse doit être inférieur au cas de LAS (sans classe). Dans cette section, nous étudions LAS-fixed(k), LAS-linear (k) and LAS-log (k) pour obtenir de premières intuitions sur les avantages et les inconvénients de ces modèles à deux classes.

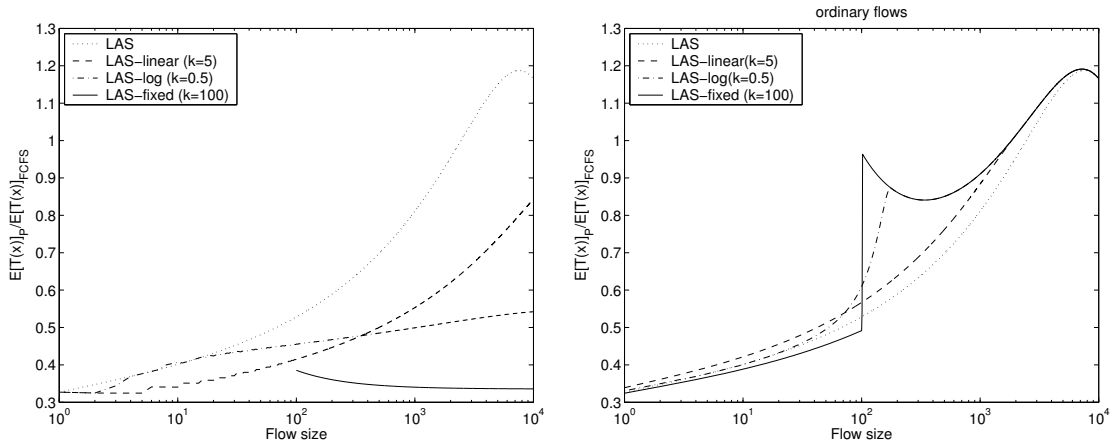
La détermination analytique des temps de réponse conditionnels des politiques LAS à deux classes (LAS-fixed(k), LAS-linear (k) and LAS-log (k)) est présentée dans [97]. Le type d'analyse est similaire à l'étude de LAS au niveau paquet de la section précédente. Pour chacune de ces politiques, nous avons développé un modèle analytique qui estime le temps de réponse conditionnel moyen et nous avons validé chaque modèle analytique avec des simulations ns-2 pour une topologie en étoile (avec un goulot d'étranglement central) et des taux de pertes inférieurs à 2%. De plus, nous avons montré qu'une file d'attente PS est un modèle raisonnable pour la discipline FIFO dans un réseau paquet. Notons néanmoins que cette dernière modélisation est moins précise que nos modélisations des politiques LAS différenciées.

3.6.2 Comparaisons des politiques LAS différenciées

Nous avons comparé les politiques LAS différenciées suivant deux axes :

– Nous avons utilisé les modèles analytiques pour l'analyse des temps de réponse conditionnels ;
– Nous avons étudié la gigue de flux longs prioritaires UDP et le débit des flux prioritaires TCP par simulation.

Les modèles analytiques ont l'avantage que les comparaisons peuvent être effectuées indépendamment d'un choix de distribution. Nous avons pu mesurer l'impact des paramètres des différentes politiques sur les flots prioritaires et ordinaires. Nous avons obtenu des figures telles que la figure 3.10, où les flux prioritaires représente 10% des flux. La partie droite de la figure dépeint les performances pour les flux ordinaires et la partie de gauche celles des flux prioritaires. On remarque tout d'abord qu'il est intéressant de garder LAS pour faire de la différentiation car cela permet de garder des temps de réponse bas pour les flux ordinaires courts. Sans rentrer dans les détails, nous avons montré que LAS-log est souvent un compromis intéressant (au sens de la protection des flux prioritaires et du maintien de temps de réponse bas pour la majorité des flux ordinaires) pour effectuer la différentiation de service.



(a) *Priority flows* (b) *Ordinary flows*

FIG. 3.10 – *Policy Performance Comparisons : Ratio of Mean Flow Transfer Time for Policy P to FCFS ($q = 0.1$, $\rho = 0.7$).*

3.7 Conclusions et Perspectives

Nous avons obtenu plusieurs contributions dans le cadre de notre étude de LAS. La première contribution est l'analyse de LAS pour une file M/G/1, où G est une distribution avec une forte variabilité. Nous avons comparé LAS avec un large éventail de politiques de service : PS, SRPT, les politiques non-préemptives (NPP) telles FIFO, LIFO, etc. Nous avons également étudié LAS dans le cas de surcharge $\rho \geq 1$. De manière générale, nous avons montré que LAS réduit le temps de réponse des clients courts sans pénaliser notablement les clients si la distribution des tailles de client est suffisamment variable.

La seconde contribution de cette étude est l'analyse de LAS dans un réseau paquet et l'étude de l'interaction de LAS avec le protocole TCP. Nous avons simulé LAS avec le simulateur ns-2 et comparé les performances de LAS avec FIFO dans le cas d'un goulot d'étranglement unique pour des tailles de flot fortement variables. Nos simulations ont montré que, comparé à FIFO, LAS diminue fortement le temps de réponse des flots ainsi que leur taux de perte.

La troisième contribution de cette étude est l'évaluation de LAS dans les réseaux pathologiques où la bande passante se trouve partagée inéquitablement entre les sources à cause d'une asymétrie en terme de latence des chemins des sources, à cause de protocoles de transport différents ou à cause de l'existence de multiples goulots d'étranglement. Les résultats de simulation montrent que LAS est plus efficace que FIFO dans ces cas car il rétablit toujours l'équité entre les sources.

La quatrième contribution de cette étude consiste en la conception, la modélisation et l'analyse de politiques de services déduites de LAS et offrant des services différenciés. L'objectif principal des ces politiques est de protéger les flots longs transportant des données prioritaires. Pour les différentes politiques proposées, nous avons développé des modèles analytiques au niveau flot permettant d'obtenir les temps de réponse conditionnels de ces

politiques. Ces modèles ont été validés par simulation.
En ce qui concerne les perspectives de ce travail, nous notons tout d'abord que de nombreux travaux scientifiques s'attèlent à l'étude des disciplines fondées sur la taille des flots [119, 9, 46, 123] et notamment SPRT [58, 59].
Quelques voix discordantes ont mis en doute l'utilisation de politiques de service spécifiques pour résoudre les problèmes d'attente dans les serveurs Web [100] ou le fait d'utiliser la taille du fichier pour approcher le temps de service restant [80]. Ces critiques touchent néanmoins plutôt SPRT que LAS.
En ce qui concerne les perspectives concernant LAS, nous avons identifié deux domaines dans lequel LAS pourrait être utile.
Tout d'abord, celui des accès ADSL chez les particuliers ou petites entreprises. En effet, dans ce genre d'environement où la capacité du lien montant est faible, le partage de la bande passante entre les flux montants longs tels que les flux pair-à -pair et les flux d'accusés de réception ou de données pour les flux interactifs est souvent problématique. C'est d'ailleurs pourquoi les applications pair-à -pair offrent en général la possibilité de limiter la bande passante qu'elles utilisent. Nous avons implanté LAS dans une machine Linux dans le cadre du stage de dernier semestre d'Alessandro Salvartori [101]. LAS permet bien évidemment de priorétiser les flux d'accusés de réception. Néanmoins, nos études ont aussi montré qu'un problème important dans le cas d'un accès ADSL est le bien fondé des hypothèses de trafic et notamment la variabilité de la distribution de la taille des flux. Une première leçon que nous avons tirée d'une campagne d'expérimentations avec notre routeur LAS est que si le lien est très chargé ou si une connexion peut occuper une fraction importante du lien à elle seule, une politique de type LAS-fixed(k) (LAS pour les flots de taille inférieur à k et FIFO pour les flots plus grands) est préférable à une politique LAS simple. Le choix du paramètre k consistue bien évidemment un problème. Nous continuons actuellement nos campagne de mesure pour approfondir ces résultats.
Un second domaine où LAS devrait être utile est celui des réseaux sans fil, que ce soient les réseaux ad-hoc [61], les réseaux à point d'accès ou les réseaux sans fil maillés [6], ces deux derniers cas consituant des extensions sans fil de l'accès à Internet, à l'échelle d'une entreprise ou d'une ville par exemple. Les réseaux sans-fil souffrent d'un double handicap que LAS devrait en partie solutionner. Premièrement, la bande passante disponible est en général plus faible dans ce type d'environement sans fil que dans un réseau filaire. Dans ce contexte, priorétiser les flux courts pour maintenir l'interactivité perçue par l'utilisateur final est important. Deuxièmement, les performances de TCP dans un environement sans-fil sont en général mauvaises. Cela provient notamment du fait que TCP interprête la contention comme un signal de congestion et réduit de manière excessive son débit d'émission. L'étude de ces problèmes sera abordé dans le cadre de la thèse de doctorat de Diego Ferrero qui a débuté en mars 2006, sous ma direction. De premiers résultats encourageants ont été présentés sous forme d'un poster à la conférence ACM SIGCOMM 2006.

Chapitre 4

Analyse de Trafic

4.1 Préambule

Les travaux présentés dans cette section ont été effectués durant la période 2004-2006. Ils sont le résultat de collaborations avec diverses personnes et équipes. Les résultats de la section 4.3 ont été obtenus dans le cadre de la thèse de doctorat de Matti Siekkinen (soutenue le 30 octobre 2006) que j'ai co-encadrée avec Ernst Biersack pour la partie analyse de trafic. Une autre partie de la thèse, consacrée à l'étude d'une solution base de données pour l'analyse de trafic a été co-encadrée par Ernst Biersack et Vera Goebel (Université d'Oslo, Norvège).
Les travaux de la section 4.4.1 ont été effectués pour partie en collaboration avec Mikel Izal (Université de Pampelune, Espagne) lors de son stage post-doctoral à Eurecom en 2003, sous ma direction. Le reste des travaux de la section 4.4 ont été faits en collaboration avec Pietro Michiardi (Institut Eurecom) et Arnaud Legout (INRIA).

4.2 Introduction

L'analyse de trafic est apparue comme un domaine de recherche à part entière à la fin des années 1990. Le point de départ symbolique est sans doute la thèse de Vern Paxson en 1997 [92]. Vern Paxson a créé NIMI (National Internet Measurement Infrastructure [93]), le premier système de mesure à (relativement) grande échelle de l'Internet. Ce réseau "overlay" qui a contenu jusqu'à une petite centaine de machines, principalement localisées dans l'Internet académique-recherche, a permis d'effectuer des mesures actives sur le débit, la latence et les taux de perte des connexions TCP.
Le besoin de mesurer les performances opérationnelles du réseau s'est imposé naturellement alors qu'il apparaissait de manière claire que le déploiement de solutions de garanties de services serait très difficile d'une part et que le réseau arrivait à prendre en charge des applications de masse, tel le Web,, d'autre part. La course au débit entre opérateurs (ISP) pour attirer les clients a aussi contribué à repousser le besoin de solutions de QoS qui se conçoivent surtout lorsque les ressources sont rares [1]. L'analyse de trafic s'est aussi

[1] Même si les ressources sont abondantes, on peut avoir besoin de solutions de QoS, comme dans le cas de services temps réel demandeurs en débit telle la vidéo à la demande. Mais comme nous l'avons

imposée comme la seule manière d'obtenir des réponses à des questions fondamentales ou pratiques telles que celles citées par Sally Floyd sur sa page web en 2002 [2]. Parmi ces questions, on trouve notamment le problème de la localisation des goulots d'étranglement dans l'Internet, la durée des périodes de congestion ou le diamètre de l'Internet.

Mesurer le réseau est devenu une nécessité en premier lieu pour les opérateurs. Leur premier besoin est un besoin de dimensionnement. Il n'existe pas pour l'Internet de règle équivalente à celle de Erlang [81] pour les réseaux téléphoniques commutés où les circuits sont à débit fixe et le critère de performance le plus important est la probabilité de blocage. Le fait même qu'il n'y ait pas de contrôle d'admission dans l'Internet et la grande variabilité [42] du trafic rendent la tâche de dimensionnement très difficile. L'expression "grande variabilité" recouvre des notions telles que la non stationarité [25, 112] ou l'auto-similarité [37] qui "montrent" que le trafic est difficilement prévisible et modélisable. Bien que de nombreux efforts théoriques aient été accomplis dans ce domaine, il nous semble que l'état de l'art pour les ISP se bornent à un sur-dimensionnement des liens. Voir à ce sujet le travail de Roch Guerin sur le sur-dimensionnement pour les grands réseaux [68].

Une autre raison pour laquelle les ISP veulent faire des études de trafic est qu'ils veulent surveiller le trafic de et vers leurs utilisateurs/clients. L'objectif est double. D'une part, la compétition entre ASPs (Access Service Providers) n'est plus limitée au débit nominal d'accès au réseau. Il faut également prendre en compte les performances perçues par les utilisateurs. L'ISP (ou ASP) veut répondre à des questions du type :

- Est-ce que certains de mes utilisateurs subissent de mauvaises performances ?
- Est-ce que le problème provient de mon réseau (en tant qu'ASP) ou d'un autre AS (Autonomous System) ?
- Est-ce que le problème provient d'une mauvaise configuration côté client ou côté serveur (par ex., serveur Web congestionné) ?

D'autre part, le rythme des attaques, dénis de service et apparitions de vers sur l'Internet est devenu tel que les ISP se doivent de surveiller le trafic depuis ou vers leurs utilisateurs [77].

Les besoins des ISPs en terme de mesures rejoignent ceux des utilisateurs qui veulent connaître l'origine des problèmes de performances qu'ils perçoivent. Un exemple extrême est la sensibilité des joueurs sur Internet aux variations de performances réseau [48].

La présente introduction n'a pas pour prétention de présenter le domaine de l'analyse de trafic dans son ensemble, qui couvre bien d'autres problématiques que celles esquissées ci-dessus et notamment : des mesures sur des services ou applications spécifiques tel BGP [121], le DNS [91] ou les applications pair-à -pair [74] (voir aussi la section 4.4 pour une contributions dans ce domaine) ; la localisation géographique [113] ; des mesures dans les réseaux sans-fil [4] ; des mesures en IPv6 [28].

Dans la suite de ce chapitre, nous décrivons nos contributions dans le domaine de l'analyse de trafic, qui vont de l'analyse opérationelle de TCP à la détection d'intrusion en passant par des mesures de BitTorrent [35] et le design et l'évaluation d'outils de mesures actifs ou passifs.

souligné au chapitre 2, d'autres services très rentables sont apparus qui n'ont pas ces besoins et tirent le marché actuellement.

4.3 TCP

4.3.1 État de l'Art

TCP est l'une des rares constantes de l'Internet. En effet, même si le type d'applications que les internautes utilisent évoluent rapidement (hier le web, aujourd'hui le pair-à -pair ou les commnunications multimédias type microsoft messenger ou skype), TCP reste la couche dominante sous la couche applicative et transporte en général 80 à 90% des octets d'un lien donné. Pour s'en convaincre, on peut ouvrir au hasard une des traces mises à jour quotidiennement sur l'archive de trace du projet MAWI au Japon [1] et consulter le ratio TCP/UDP de cette trace. TCP est le protocole de niveau transport essentiel à l'Internet puisque c'est grâce à lui que l'Internet ne s'effondre pas sur lui-même. Cela étant, et même si les algorithmes de base de TCP (slow-start, congestion avoidance, Nagle's, fast retransmit, fast recovery, SACK) sont bien connus, leurs performances opérationnelles restent largement inconnues. Tout d'abord, il existe de nombreuses versions de TCP, Reno, NewReno, SACK et pour chaque version (voir [51] pour une étude comparative des performances de ces variantes), de nombreuses implantations qui différent dans les choix de certains paramètres. Voir [89, 83] pour des techniques de test d'implantation et de conformité aux différentes versions du protocole TCP. Ces deux études, faites à quelques années d'écart permettent aussi de mesurer les changements dans les implantations. Le constat général est que TCP New Reno s'est affirmé entre 2001 et 2004 comme la version majoritaire et l'utilisation de l'option d'accusé de reception sélectif (SACK) s'est généralisée. Notons néanmoins que les tests précédents sont effectués sur des serveurs web car ils peuvent être intérrogés et génèrent un échange de données suffisamment long pour inférer la version et les options utilisées par la couche TCP du serveur.

Plus généralement, les performances de TCP demeurrent difficiles à apprécier à l'échelle de l'Internet. La question qu'il faut résoudre peut se formuler ainsi : supposons un ensemble de traces passives collectées sur différents liens, quelles sont les causes qui expliquent les performances observées ? Pour pouvoir répondre à cette question, il faut tout d'abord être capable de reconstituer tout ou partie de la machine à états de TCP, ou du moins certains paramètres clefs internes ou externes, notamment le RTT [106, 120, 5] ou la fenêtre de congestion [114]. La recherche elle-même des causes dans des traces passives a reçu moins d'attention. Le travail principal est [126]. Ce travail est le premier à avoir abordé le problème de manière systématique. La première contribution de cet article a été de proposer une taxonomie des causes possibles de limitation du débit d'une connexion. Les auteurs ont identifié 4 causes possibles principales : la taille du tampon côté émetteur ou récepteur, l'application au dessus de TCP, un goulot d'étranglement partagé ou non, la faible durée de l'échange (nous avons vu au chapitre 3 que la majorité des connexions sont très courtes dans l'Internet). Leur étude reposait sur l'identification de groupes de paquets (*flights*) émis simultanément par la source et qui permettait d'une part d'inférer le RTT de la connexion, et d'autre part de marquer les paquets comme étant en *slow-start*, congestion avoidance ou pertes. Le talon d'achille de leur méthode réside dans l'idenfication des groupes de paquets qui reste un sujet de recherche ouvert [105].

Une autre question que l'on peut se poser est celle de l'impact du comportement observé

de TCP sur les performances. TCP est en effet supposé "potentiellement agressif" au sens où il peut générer de longs trains (*flights* ou *bursts*) de paquets qui peuvent charger les files d'attente et créer des pertes. Plusieurs études se sont concentrées sur cette caractéristique de TCP [19, 105, 72] avec pour premier objectif d'identifier les bursts sur une trace passive. TCP est un des sujets de recherche les plus actifs depuis sa création au début des années 1980. De nombreuses évolutions sont régulièrement proposées pour améliorer les performances du protocole dans des environnements particuliers tels les chemins avec un produit bande passante délai élevé [78, 30], des liens satellitaires [49, 66], des environnements sans-fil [61] ou des liens asymétriques [11]. La modélisation de TCP et la mise au point de techniques de prédiction du débit effectif est également un sujet actif de recherche [90, 86, 8]. Nous n'avons pas eu pour but ici de faire un état de l'art complet de TCP, mais de présenter un tableau suffisamment précis dans lequel s'insère nos contributions :

- Une approche méthodologique pour l'analyse de trafic TCP utilisant des bases de données. En effet, il nous est apparu très tôt dans nos premiers pas en analyse de trafic que le problème n'était pas tant de collecter des traces que de savoir où était telle trace et ce qu'elle contenait.
- Une technique de filtrage et de quantification des effets dus à l'application au-dessus de TCP. Cette première étape est nécessaire avant d'aller plus loin dans l'analyse des causes de limitation de débit d'une connexion observée passivement.
- De nouvelles techniques d'analyse de causes de limitation de débit avec une approche fondée sur l'analyse de multiples séries temporelles déduites du flux d'arrivée des paquets d'une connexion.

4.3.2 Méthodologie : une approche base de données pour l'analyse de trafic

La démarche "classique" en analyse de trafic est de (i) collecter des données et (ii) utiliser des scripts en perl ou dans un autre langage optimisé pour la manipulation de fichiers et de chaînes de caractères et (iii) générer un papier de recherche. C'est typiquement la méthode que nous avons utilisée dans nos études de BitTorrent, section 4.4. L'avantage de cette méthode est sa rapidité de mise en œuvre, si tant est que l'on maîtrise un langage de script approprié. Les inconvénients sont néanmoins nombreux avec ce type de méthode :

- le code de recherche est, par définition, peu commenté et quand la personne ayant écrit le script quitte l'équipe, les programmes sont difficiles à transférer à quelqu'un d'autre ;
- plus fondamentalement, les langages de scripts n'ont pas de notion de sémantique associée à des valeurs. Une estampille temporelle ou une adresse IP sont toutes deux vues comme des nombres sans dimension. La conséquence directe est que les erreurs sont plus difficilement détectables car on peut facilement mélanger des pommes et des oranges, surtout lorsque les conventions de nommage font qu'une adresse IP s'appelle $1 et une estampille $2. Le corollaire est que les résultats intermédiaires d'un programme ne sont en général pas réutilisables et il faut à chaque fois repartir des données brutes pour générer des résultats.
- Les débits des réseaux et des applications augmentant régulièrement, il faut énormément optimiser un script pour éviter qu'il implose lorsqu'il traite des fichiers de données de

plusieurs gigaoctets. Par exemple tcptrace [2] qui génère des résumés au niveau connexion ou des graphiques à partir de traces paquets obtenus avec tcpdump [3] explose pour un fichier tcpdump de plus de un gigaoctet.

En plus des arguments ci-dessus, obtenir des traces devient de plus en plus facile, car il existe de nombreuses sources de traces publiques (et collecter des données TCP avec un client BitTorrent est chose aisée), on se retrouve rapidement face à un problème d'indexation des traces elles-mêmes.

Pour résoudre ces problèmes, Matti Siekkinen a développé une approche base de données. L'idée initiale était simplement d'indexer nos nombreuses traces. Mais après discussion avec une spécialiste des bases de données (Vera Goebel), il est apparu que ces dernières étaient une solution idéale pour l'analyse de trafic car :

– Une fois chargées dans la base de données, les données acquièrent une sémantique.
– La majorité des opérations effectuées sur des données en analyse de trafic sont des actions de filtrages et groupages de données qui sont des opérations naturellement prises en charge et optimisées dans les systèmes de bases de données.
– Les bases de données sont optimisées pour gérer de très grands volumes de données et fournissent des techniques d'indexation et de clustering (regroupement physique des données sur un disque ou une paritition) qui accélèrent l'accès aux données.
– Les bases de données permettent également de gérer l'ensemble des fonctions créées par les utilisateurs.

Dans [109], nous avons présenté notre approche base de données pour l'étude de traces de paquets de connexions TCP. Un sous-ensemble des tables et des relations de notre base de données, appelée Intrabase, est représenté dans la figure 4.1.

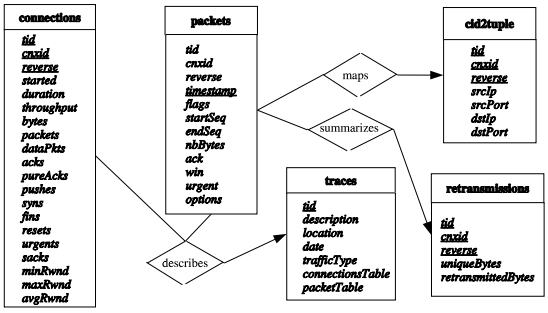


FIG. 4.1 – Tables et relation dans Intrabase

Nous avons évalué les performances de Intrabase en terme d'espace disque et de temps de chargement pour deux traces occupant la même taille disque, mais comportant un nombre différent de connexions. La première trace est une trace de trafic observée sur un lien donné alors que la seconde est une trace de trafic BitTorrent uniquement et comporte de ce fait beaucoup moins de connexions. On voit dans la figure 4.2 que le temps de traitement augmente relativement linéairement (propriété de scalabilité) avec la taille du fichier, même si le nombre de connexions augmente la pente de la droite. En ce qui

[2] http ://jarok.cs.ohiou.edu/software/tcptrace/
[3] http ://www.tcpdump.org/

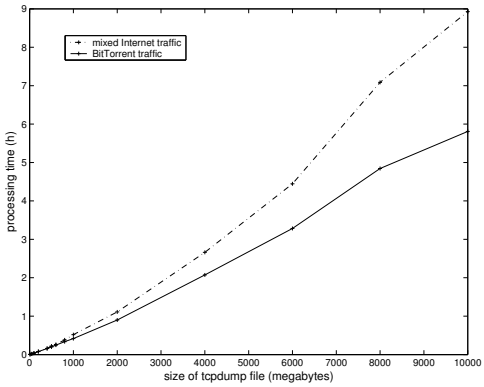


FIG. 4.2 – Temps de traitement vs. taille du fichier.

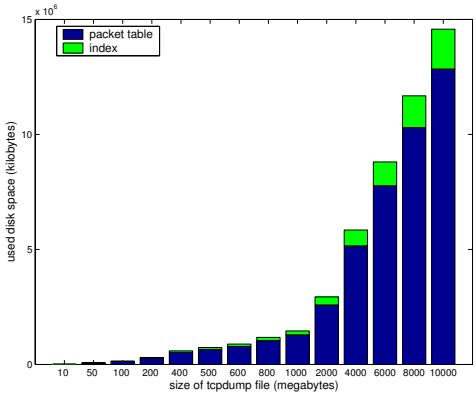


FIG. 4.3 – Espace disque utilisé pour le trafic BitTorrent

concerne l'espace disque utilisé, on s'aperçoit en regardant la figure 4.3 que l'indexation utilisée ne représente au final qu'un surplus modéré (moins de 10%) d'espace. Notons que c'est le prix nécessaire à payer pour optimiser les temps d'accès aux données.

Au final, notre expérience pratique avec les bases de données au quotidien montre que c'est un outil utile quand la masse de données que l'on a à traiter augmente. Néanmoins, si cet outil est bien adapté aux besoins de la recherche, il ne l'est pas tel quel pour un ISP qui va générer plusieurs dizaines de Gigaoctets de données tous les jours. Dans ce cas, il faut soit suivre une approche "data streaming" dans laquelle on génère à la volée des résumés des données [94] ou alors n'utiliser Intrabase que pour des traitements avancés sur des "petites" quantités de données (de l'ordre de 10 gigaoctets maximum). Ces chiffres sont le reflet de notre expérience avec France Télécom R & D Sophia-Antipolis qui utilise Intrabase pour certaines analyses de données collectées quotidiennement sur une plaque ADSL de 4000 utilisateurs.

4.3.3 Méthode générique de filtrage des effets applicatifs

Dans leurs travaux pionniers, Zhang et al. [126] ont observé que 50% des connexions TCP observées (sur des liens offrant différents niveaux d'agrégation - depuis les bords jusqu'au cœur de l'Internet) avaient leur débit limité principalement par l'application au-dessus de TCP. Leur méthode d'identification de cette limitation par l'application était de rechercher un train de paquets terminé par un paquet de taille inférieur au MSS (*Maximum Segment Size*) et séparé du paquet suivant par un temps significativement supérieur au RTT de la connexion. Leur méthode était qualitative puisqu'à partir du moment où le motif précédent était repéré, la connexion était déclarée "limitée par l'application". Notons qu'une connexion peut se voir attribuer plusieurs limitations, par exemple une limitation par l'application et une limitation par le tampon côté récepteur.
La méthode que nous présentons dans cette section se veut au contraire quantitative : on veut pouvoir quantifier la quantité de données transmises dont le débit est limité par l'application. Nous avons développé (voir [110] pour les détails) un algorithme qui sépare systématiquement une connexion en des parties limitées par l'application et des parties non limitées par l'application (mais limitée soit par le réseau, soit par l'algorithme de TCP ou ses variables d'environnement). Notre point de départ est la détermination de l'ensemble des localisations où TCP et l'application peuvent interagir. Pour cela, on se réfère à la figure 4.4.

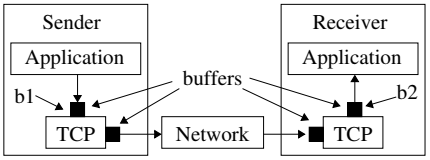


FIG. 4.4 – Interaction entre TCP et l'application.

On voit sur la figure 4.4 que TCP et l'application interagissent principalement au niveau des buffers b1 et b2. Du point de vue de TCP, l'application ralentit le flux des données dans le réseau si le buffer b1 est vide alors que TCP pourrait transmettre (la fenêtre de congestion n'est pas entièrement consommée) ou le buffer b2 est plein ce qui va faire passer la fenêtre du récepteur à zéro (*advertised window*). Le second cas est anecdotique alors que le premier est le cas courant qui nous sert à définir une limitation par l'application du débit d'une connexion TCP.
Si maintenant on regarde de plus près comment une application peut effectivement ralentir son débit d'émission vers la couche TCP, on s'aperçoit qu'il y a plusieurs cas possibles. Tout d'abord, l'application peut ne pas avoir de données à émettre pendant un certain temps. Le premier exemple qui vient à l'esprit est le cas telnet et des applications interactives en général, telles les applications de *chat* [47]. Dans ce cas, plus que l'application elle-même, c'est l'utilisateur au-dessus de l'application qui induit la limitation. Une classification naturelle des applications est de les diviser entre applications interactives et applications de transfert continu (bulk transfer applications). Si cette typologie est pertinente lorsque l'on veut effectuer une classification de haut niveau [85], elle n'est pas suffisante dans notre cas. En effet, les applications de transfert continu peuvent utiliser

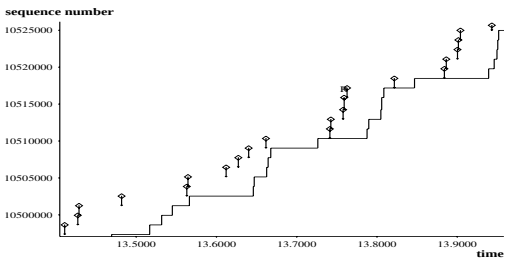


FIG. 4.5 – Connexion eDonkey.

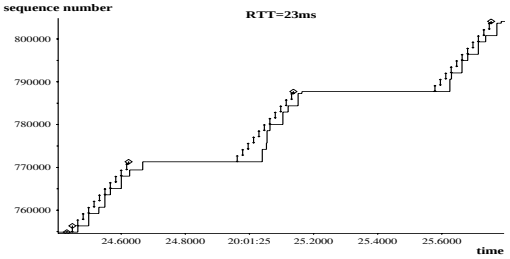


FIG. 4.6 – Connexion BitTorrent.

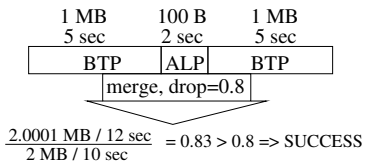


FIG. 4.7 – Groupage réussi.

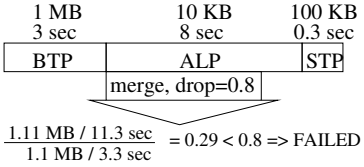


FIG. 4.8 – Groupage impossible.

différentes techniques pour contrôler leur débit qui ne se traduisent pas de la même façon du point de vue de TCP. Une première façon de limiter le débit est d'émettre des paquets de taille inférieure à un MSS avec un push flag (bit de contrôle de TCP qui permet à l'application de "forcer" TCP à émettre les données sans en attendre plus). C'est ce que peut faire un client edonkey comme illustré dans la figure 4.5. Une autre façon de faire est d'envoyer des petits trains de paquets (de taille un MSS) séparés par des périodes de silence. C'est ce que peut faire un client BitTorrent, voir figure 4.6.

Plus généralement, on se rend compte qu'une application peut interagir avec TCP de multiples manières et à de multiples échelles de temps (inférieure ou supérieur au RTT). Notre algorithme d'analyse de connexion fonctionne en deux phases. Dans une première phase, on cherche à faire une séparation des periodes dites ALP (Application limited periods) où l'application domine, des périodes BTP (Bull Transport Period) où c'est un autre facteur qui domine. La première condition pour avoir une BTP est d'avoir au moins 130 paquets. Cette valeur est choisie de manière à être (relativement) sûr que TCP ait quitté la phase de Slow-Start. La détermination des frontières entre ALP et BTP se fait en tenant compte de la densité de *Push Flags* et du temps d'inter-arrivée des paquets par rapport au RTT estimé de la connexion [4]. Les détails sont dans [110], actuellement en soumission.

Une fois les ALP et BTP extraitent, notre algorithme entre dans une seconde phase durant laquelle il va chercher à former des BTP plus grandes en agrégeant des ALP à des BTP. L'idée est que si on a de longues BTP séparées par de petites ALP, il peut être intéressant de former une macro-BTP qui inclut le tout. Un paramètre appelé drop permet de contrôler la décroissance en débit obtenue lorsque l'on fait l'agrégation. Les figures 4.7 et 4.8 illustrent la méthode.

Afin de montrer l'intérêt de l'algorithme et les difficultés que certaines analyses pourraient

[4] Nous implantons plusieurs méthodes différentes issues de [120, 71]

rencontrer si la séparation entre les ALPs et les BTPs n'est pas faite correctement, nous avon appliqué notre algorithme à une grande variété de traces de différentes applications : eDonkey, BitTorrent, SSH, Gnutella, FTP, etc. À titre d'illustration (plus de détails et d'analyses sont dans [110]), on montre figure 4.9 le débit des connexions communes dans les 50 BTP ayant le débit le plus élevé et les 50 connexions ayant le débit le plus élevé pour du trafic BitTorrent. On s'aperçoit avec la figure 4.9 que seules 12 connexions sont communes aux deux ensembles et, de plus, les débits ne sont pas ordonnés de la même manière.

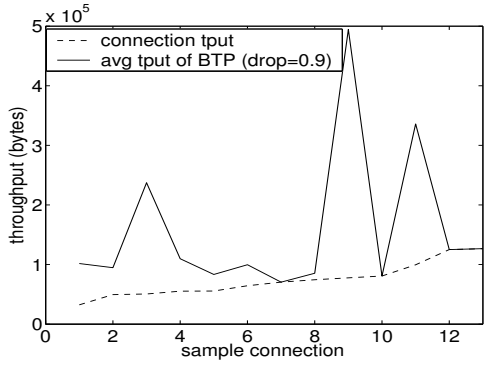


FIG. 4.9 – Débits des connexions communes au 50 BTP ayant le plus gros débit et les 50 connexions ayant le plus gros débit pour le trafic BitTorrent ($drop = 0.9$) .

Une seconde illustration est le ratio des temps d'aller-retour moyens des ALP et des BTP pour une même connexion dans le cas d'une estimation passive du RTT telle que proposée dans [120]. La méthode présentée dans [120] est illustrée figure 4.10. On voit que le lorsque le délai d3 entre l'envoie de deux paquets par l'application devient grand, l'estimation du RTT moyen pour les ALPs augmente. Notons que les délais durant les BTPs peuvent être également grands si l'application charge une file d'attente par exemple. On retrouve ces deux effets dans la figure 4.11 où l'on peut voir que la distribution des ratios varie pour certaines applications sur une grande échelle de valeur. Par exemple, pour eDonkey, on a 10% des valeurs inférieures à 0,1 et 18% des valeurs supérieures à 10.

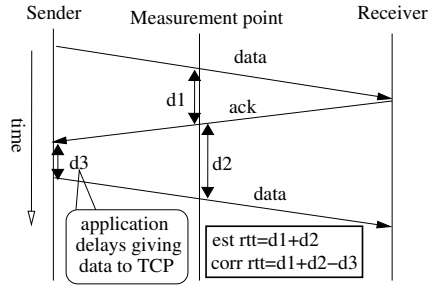


FIG. 4.10 – Estimation passive du RTT.

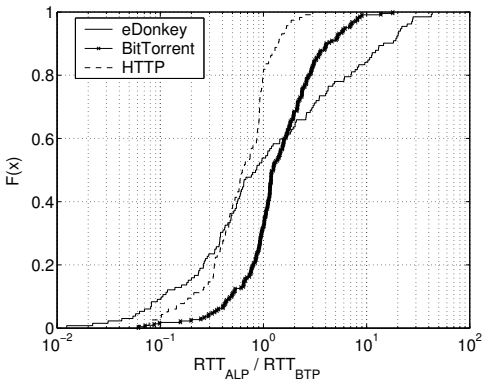


FIG. 4.11 – CDFs du ratio $\frac{\overline{RTT}_{ALP}}{\overline{RTT}_{BTP}}$.

4.3.4 Détermination du facteur limitant d'une connexion

Dans cette section, nous résumons les travaux présentés dans [111]. Notre objectif est d'aller au-delà des résultats de Zhang et al. [126] et d'avoir une méthode plus fiable et qui soit de plus quantitative. Comme nous l'avons dit dans la section 4.3.2, l'algorithme utilisé dans [126] repose sur la notion de trains de paquets (flights) entre lesquels on espère régulièrement observer des durées proches de paramètres clefs de la connexion, tel le débit de l'émetteur ou le RTT de la connexion. Des chercheurs ont déjà critiqué la faiblesse de cette approche en montrant qu'il est difficile d'observer ces fameux trains de paquets dans des mesures passives [106]. Pour illustrer le problème, nous montrons sur les figures 4.12 et 4.13 des histogrammes des temps d'inter-arrivées des paquets pour des connexions limitées par la fenêtre de contrôle de flux (advertised window). La figure 4.12 (resp. la figure 4.13) correspond au cas où on utilise pas (resp. on utilise) la méthode de *delayed ack*. Dans les deux cas, on observe bien des trains de paquets, mais c'est seulement pour le premier cas que l'on peut relier les durées inter-trains au RTT de la connexion. S'il est facile d'observer des trains, il est donc souvent difficile d'inférer le RTT de la connexion à partir de ces trains, d'autant plus que dans un cas réel, du bruit s'ajoute aux précédents histogrammes en raison des interactions entre les connexions dans les routeurs (dans notre cas, il n'y avait aucune connexion parasite).

Notre approche dans [111] consiste à calculer un certain nombre de scores associés à des séries temporelles calculées à partir d'une trace au niveau paquet d'une connexion. Au contraire de [126], nous supposons que la trace est nécessairement bidirectionnelle. Nous n'avons pas étudié le cas des traces unidectionnelles (accusés de réception ou paquets seulement) car il nous semble déjà difficile d'avoir des estimations fiables de certains paramètres tel le RTT en observant les deux parties d'une connexion. En revanche, comme [126], nous ne supposons pas que la trace ait été capturée proche de l'émetteur ou du récepteur d'une connexion.

Les séries temporelles que nous calculons sont :

– La série temporelle des temps d'inter-arrivées des accusés de réception purs. À partir de cette série temporelle, nous estimons le débit d'accès du chemin entre l'émetteur et le récepteur TCP (voir [50]) pour les détails).

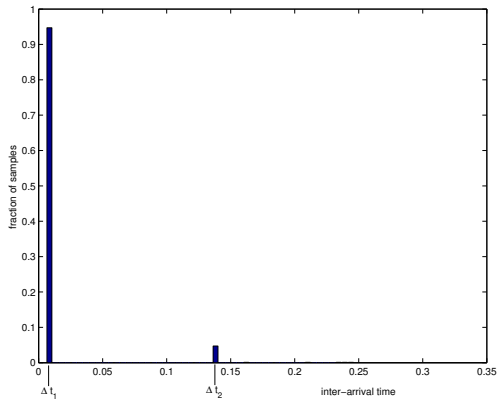


FIG. 4.12 – Sans Delayed ACKs

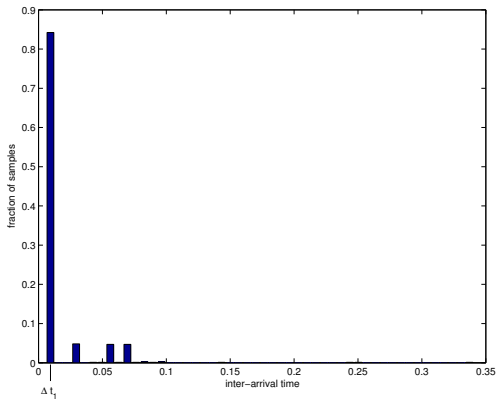


FIG. 4.13 – Avec Delayed ACKs

- La série temporelle de la taille de fenêtre de contrôle de flux (à partir du champ advertised window dans le flux d'accusé de réception).
- La série temporelle du taux de retransmission par intervalle d'une seconde.
- La série temporelle du nombre d'octets émis mais non encore aquittés. Cette série temporelle, croisée avec celle de la taille de la fenêtre de contrôle de flux permet de voir si le débit est limité par cette fenêtre de contrôle de flux (receiver window limitation).
- La série temporelle du débit par intervalle de une seconde $tput(t)$. À l'aide de la capacité estimée C, nous calculons la série temporelle de la dispersion $1 - \frac{tput(t)}{C}$. Intuitivement, si la dispersion est proche de 0, la connexion ne partage pas le goulot d'étrangement du chemin (si par exemple c'est une connexion modem et que c'est le debit de 56 kbit/s du modem qui limite le débit) et s'il est proche de un, le goulot d'étranglement est partagé.

Puisque l'on suppose que les traces peuvent être collectées entre l'émetteur et le récepteur, il faut tenir compte du fait que l'on est entre ces deux points pour le calcul des séries temporelles. Cela nous oblige par exemple, à translater la série temporelle du nombre d'octets émis qui est observée au point de mesure, d'une valeur qui soit égale à la fraction du RTT entre l'émetteur et le point d'observation ($\frac{d1-d3}{2}$ sur la figure 4.14). Cela nécessite un bon estimateur du RTT de la connexion (nous en utilisons plusieurs). Nous avons validé notre approche à l'aide d'expérimentations où l'on collecte le flux de données côté récepteur et où on reconstitue ce flux côté émetteur. On compare alors la série estimée avec la série "exacte" obtenue à l'aide de l'outil Web100. Cet outil est un *patch* du noyau linux qui permet d'interroger en temps-réel les variables d'états d'une connexion TCP. Un exemple de résultats obtenus est présenté dans la figure 4.15.

Nous avons appliqué dans [111] notre approche aux BTP (voir section précédente) d'une trace BitTorrent de plusieurs Gigaoctets. Nous présentons ci-dessous quelques figures illustrant le type de résultat que cette analyse des scores peut donner. Tout d'abord, dans la figure 4.16, on croise le taux de retransmission de la connexion avec le score de dispersion. La tendance que l'on observe est conforme à l'intuition : si une connexion partage un goulot d'étranglement avec d'autres, son taux de perte risque d'être plus élevé que si elle ne le partage pas (score de dispersion proche de 0).

Dans la figure 4.17(a), on a représenté le taux de retransmission face au score de limitation

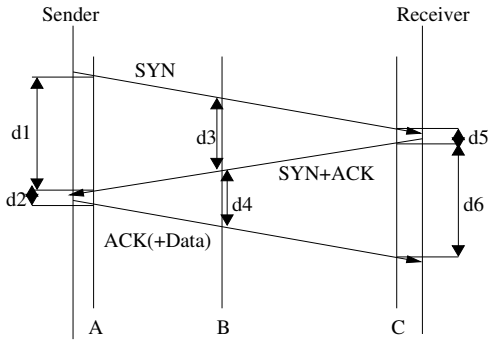


FIG. 4.14 – Determination du point de mesure

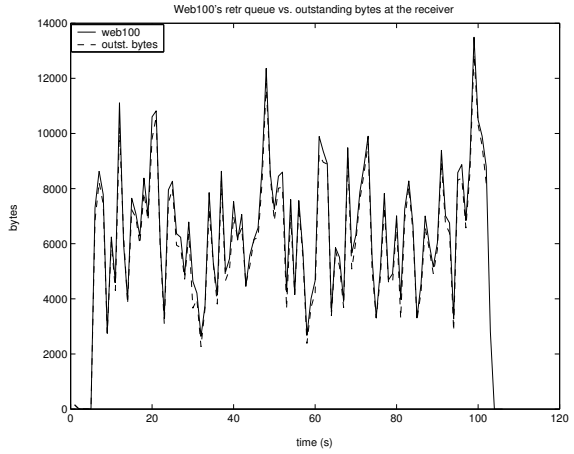


FIG. 4.15 – Validation de la technique d'inférence de la série temporelle des octets non acquittés côté emetteur à partir de mesures côté récepteur.

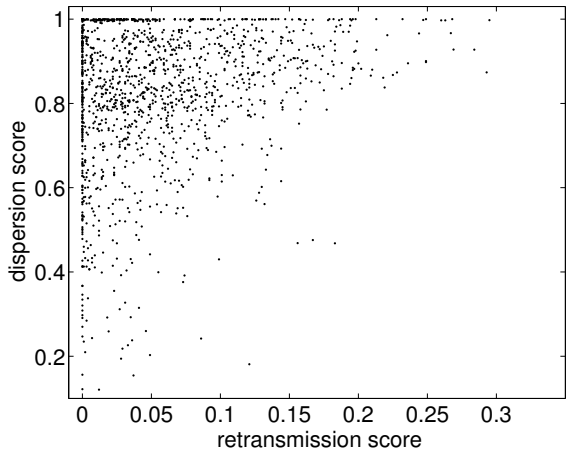
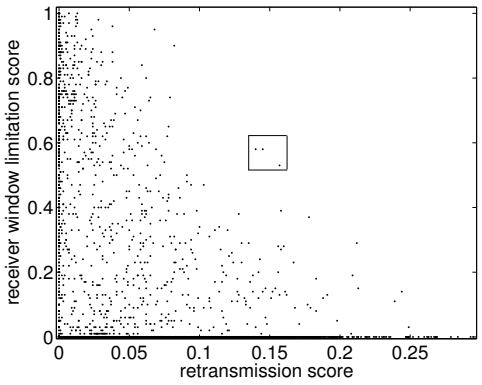
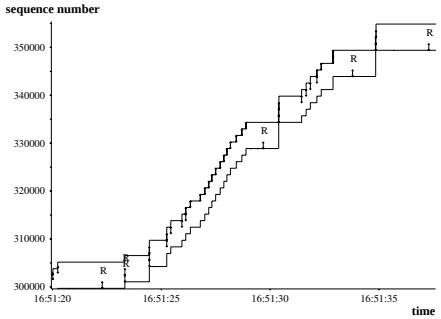


FIG. 4.16 – Dispersion vs. Retransmission.

par la fenêtre de contrôle de flux. Intuitivement, si on est limité par cette dernière, cela veut dire que notre débit fluctue peu autour de la valeur de l'adversited window (et que le taux de perte est faible). C'est ce que l'on observe sur la figure 4.17(a). On peut aussi pousser plus loin l'analyse et voir qu'il existe des points (identifiés par un rectangle dans la figure 4.17(a)) pour lesquels on a des valeurs intermédiaires pour les deux scores. Une étude attentive de ces connexions (l'une d'entre elles est illustrée figure 4.17(b)) montre qu'en fait les deux causes existent pour ces connexions mais à des periodes de temps différentes. Par exemple, dans le cas de la figure 4.17(b), le diagramme numéro de séquence TCP en fonction du temps montre que l'émetteur est souvent proche de la fenêtre de controle de flux (ligne supérieure) mais que régulièrement aussi il y a des pertes. Plus généralement ce type de résultat souligne la difficulté de ce type d'analyse et le fait que plusieurs causes sont possibles pour une même connexion, même si elles ne sont pas simultanément subies.



(a) Score de Retransmission vs. Score de limitation par l'adverised window



(b) Zoom sur une connexion.

4.3.5 Stationnarité des connexions TCP

Nous présentons maintenant des travaux plus théoriques sur la stationnarité de TCP. La motivation de ce travail est que les flux TCP longs deviennent de plus en plus courants dans l'Internet. C'est le cas par exemple des transferts pair-à -pair (partage de fichiers avec, par exemple, edonkey ou réplication de fichiers avec, par exemple, BitTorrent) ou des communications multimédias de type Skype[5]. Bien que l'Internet offre un simple service best-effort, le sudimensionnement des liens de cœur du réseau et la généralisation des accès à haut débit pour les particuliers laissent espérer que l'on puisse observer des débits élevés pendant des périodes longues. Pour quantifier ces intuitions, nous avons développé un outil d'analyse des séries temporelles du débit de connexions TCP. Avec cet outil, on peut espérer quantifier :

– La distribution des durées pendant lesquelles les connexions TCP sont stables ;

[5]Skype privilégie les transferts UDP mais devant la généralition des pare-feux, beaucoup de transferts sont effectués en TCP.

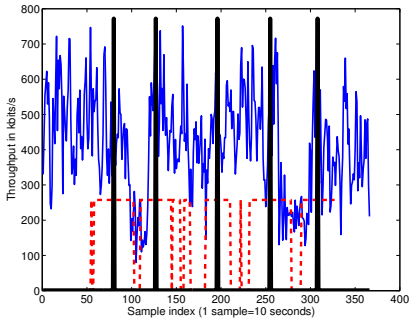


FIG. 4.17 – Série temporelle initiale (ligne continue), série binaire (ligne pointillée) et points de changements (barres)

– La valeur des sauts en moyenne ou en variance d'une période stable à une autre.

Utiliser la notion de stationnarité mathématique pour attaquer les questions ci-dessus permet de définir un cadre rigoureux d'étude. Mais il faut remarquer que les travaux précédents [127] dans le domaine ont souligné qu'un analyste voudrait sans doute faire la différence entre stationnarité mathématique et stationnarité opérationnelle, définie comme la période pendant laquelle le débit ne bouge pas de plus de $x\%$. Le contre exemple utilisé dans [127] est qu'un processus mathématique stationnaire pouvait avoir une variance aussi élevée que l'on voulait par construction et que du point de vue de l'analyste réseau, ce processus ne serait pas "stationnaire". Cela étant, l'exemple précédent nous semble un peu exagéré car les données de base sur lesquelles nous travaillons sont des séries temporelles de débit de connexions TCP qui ont peu de chance d'être à la fois stationnaire d'un point de vue mathématique et à haute variance.

Pour découper une série temporelle de débit d'une connexion en des périodes stationnaires, nous avons développé un algorithme de détection de changement [14] fondé sur le test d'ajustement de Kolmogorov-Smirnov (KS) qui permet de comparer deux échantillons statistiques et de rejeter, si besoin est, l'hypothèse suivant laquelle ils sont issus d'une même distribution. Le test de KS est un test non-paramétrique, c'est-à -dire sans hypothèse sur le modèle *a priori* de ce qui est observé. Il nous fallait en effet un modèle non paramétrique car nous n'avions pas d'idée *a priori* sur les distributions que nous pourrions observer. Pour pouvoir appliquer le test de KS sur des séries temporelles, il faut avoir des échantillons indépendants. Pour ce faire, nous avons travaillé à une échelle de temps de 10 secondes. Il a fallu aussi définir des règles de décision pour détecter un changement véritable. L'idée est de tester des sous-séries temporelles (d'une cinquantaine d'échantillons) contiguës, d'appliquer le test de KS, puis de se décaler d'un échantillon. Cela permet de transformer notre série initiale en série binaire (0= les échantillons sont équivalents, 1= ils sont manifestement différents). Ensuite, nous avons défini un critère empirique suivant lequel il fallait que la série binaire ait 15 valeurs consécutives à '1' pour déclarer qu'il y avait effectivement un changement. Un exemple d'application de l'algorithme est donné figure 4.17 où les barres verticales sont les points de changement, la ligne continue la série initiale et la ligne en pointillés les résultats des tests (i.e. la série binaire des résultats de tests mais agrandie pour être lisible).

Pour appliquer et valider notre algorithme, nous avons travaillé sur des traces BitTorrent qui génèrent naturellement de longs transferts. Un transfert BitTorrent peut être vu comme un transfert FTP avec des périodes actives qui alternent avec des périodes inactives. Nous avons isolé les périodes et appliqué notre algorithme sur ces périodes. En terme de validation nous avons comparé les sauts de moyennes et de variances entre périodes stationnaires avec les sauts de moyenne et de variance à l'intérieur des périodes stationnaires lorsque celles-ci étaient (artificiellement) coupées en deux. Les figures 4.18 et 4.19 montrent les boxplots pour la moyenne et la variance respectivement. Les boxplots pour les sauts entre périodes jugées stationnaires étant plus larges, nous obtenons une bonne validation de notre algorithme.
On peut ensuite regarder des paramètres plus opérationnels comme la durée des périodes stationnaires par rapport à celles des périodes actives initiales, voir figure 4.20. La moyenne est de 16 min pour les premières et 73 pour les secondes. Ces résultats montrent que les transmission longues sur l'Internet sont effectivement viables car 15 min est déjà une valeur relativement longue. En ce qui concerne les débits on observe sur la figure 4.21 que les débits des périodes stationnaires sont un peu plus élevés. Loin d'être une erreur, ce résultat montre que notre algorithme fonctionne bien car, par exemple, si on a une période de tansfert longue à un débit relativement faible avec au milieu une sous-période à débit élevé, l'algorithme va générer 3 valeurs, deux faibles et une élevée, alors qu'avec les périodes initiales, il y a un seul échantillon de valeur faible.

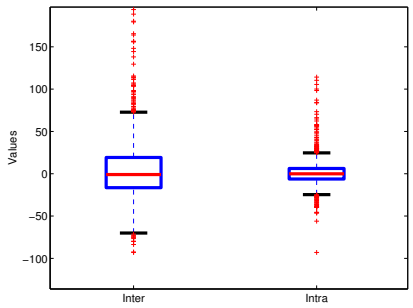


FIG. 4.18 – Boxplot des sauts de moyennes entre périodes stationnaires (gauche) et des sauts de moyennes intra périodes stationnaires (droite)

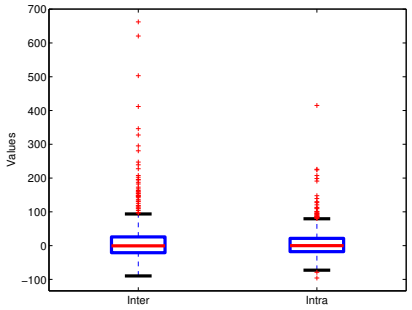


FIG. 4.19 – Boxplot des sauts d'écart type entre périodes stationnaires (gauche) et des sauts d'écart type intra périodes stationnaires (droite)

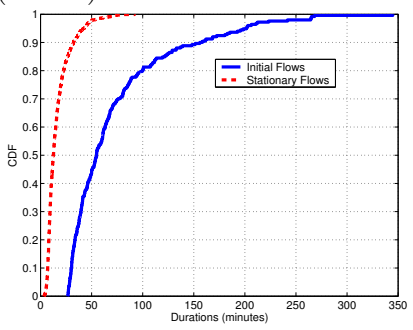


FIG. 4.20 – Fonctions de répartition des durées

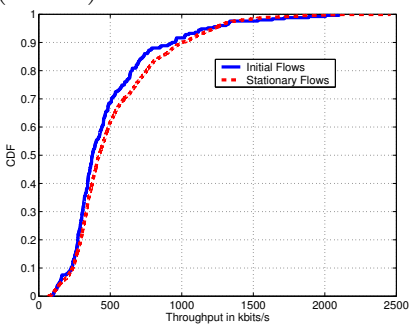


FIG. 4.21 – Fonction de répartion des débits

4.4 Mesures de niveau applicatif : BitTorrent

BitTorrent [35] est une application pair-à -pair de réplication de contenu. On utilise le terme de réplication de fichiers par opposition au partage de fichiers. L'objectif premier dans les applications de partage de fichiers est la localisation du contenu alors que dans le cas de la réplication de fichiers, l'accent est mis sur la vitesse de réplication d'un contenu unique entre plusieurs pairs. BitTorrent (BT) est à notre connaissance le seul protocole de réplication de contenu qui soit utilisé à large échelle.

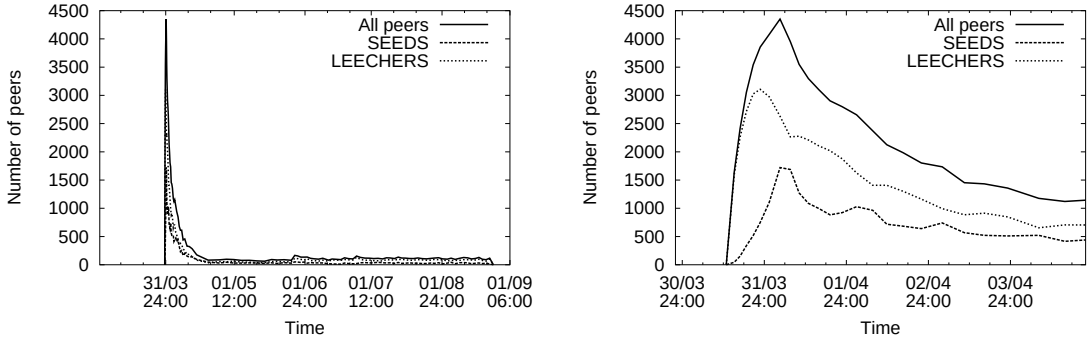
Le principe de fonctionnement de BT est le suivant. Le point d'entrée d'une session BT (aussi appelée torrent) est en général un serveur Web qui fournit au client BT un fichier ".torrent" qui contient entre autres l'adresse IP d'un tracker et les codes d'intégrité (SHASH1) des différentes pièces qui forment le contenu. Le client BT va alors contacter le tracker qui va lui fournir une cinquantaine d'adresses IP de clients qui participent déjà au torrent. Le rôle du tracker se limite à collecter des statistiques sur le torrent et servir de point de rendez-vous pour tous les clients BT du torrent. Un client BT maintient en permanence une liste de clients, son *peer set*, avec qui il coopère, c'est-à -dire échange des bouts (pièces) du fichier. Plus précisement, un pair émet des pièces avec au plus 4 clients à un instant donné. L'algorithme *choke* est utilisé pour choisir ces 4 clients. Ce sont les 4 clients qui envoient avec le débit le plus élevé vers le client considéré. Afin de permettre aux clients nouvellement arrivés d'obtenir des premières pièces : un client est régulièrement choisi au hasard et ajouté à la liste des 4 clients contrôlés par l'algorithme *choke*. Une fois le client choisi, il faut choisir la pièce qui sera envoyée. Le principe retenu est de demander en priorité les pièces qui sont les moins répliquées dans le *peer set*. L'idée est que cette optimisation locale (à un *peer set*) va mener globalement un torrent à avoir une bonne entropie globale (au niveau du torrent) des pièces. Nous définirons plus avant la notion d'entropie dans la section 4.4.2. Dans la terminolgie de BT, une *seed* est un client qui a fini de répliquer le contenu et qui reste dans le torrent pour servir les clients qui n'ont pas fini et sont appellés des *leechers*.

4.4.1 Le torrent Redhat

Dans [69], nous avons étudié un torrent particulier de très grande taille, le torrent de la distribution Redhat 9 de Linux. Le fichier à répliquer faisait 1.77 Gigaoctets. Notons que l'utilisation de BT pour ce genre de transferts est courante car elle est nettement plus efficace (et moins coûteuse) que l'utilisation de serveurs miroirs répartis dans le monde entier. Nous avons travaillé sur deux sources de données :

– Le log du tracker du torrent sur une durée de 5 mois depuis le début du torrent.
– Le log au niveau applicatif d'un client BT que nous avons fait participer au torrent, approximativement 3 mois après le début du torrent.

Concentrons nous d'abord sur le fichier log du tracker. Les informations contenues dans ce fichier sont les rapports que les clients envoient, approximativement toutes les 30 minutes, au tracker avec le nombre d'octets qu'ils ont émis et reçus. Sur les 5 mois, nous avons observé un total de 180000 clients avec une grosse concentration de 50000 clients (flash crowd) durant la première semaine, comme le montrent les figures 4.22(a) et 4.22(b).



(a) Trace complète (b) Zoom sur les 5 premiers jours

FIG. 4.22 – Nombre de clients actifs à un instant donné

Les *seed*s jouent un rôle essentiel dans BT et nous nous sommes tout d'abord demandés si l'efficacité supposée de BT n'était pas simplement due à la présence des *seeds*. Dans la figure 4.23, nous présentons le débit cumulé d'octets émis par les *seeds* et les *leechers* sur les 5 mois. Il apparaît que si les *seeds* fournissent la majorité des données, les *leechers* fournissent tout de même un volume significatif de données.

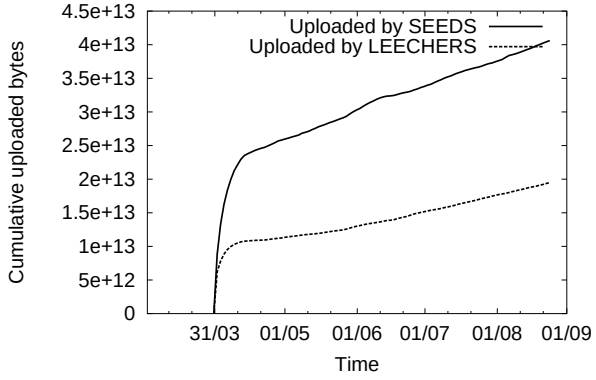


FIG. 4.23 – Volumes uploaded by *seed*s and *leecher*s

Nous nous sommes ensuite intéressés au débit moyen perçu par un utilisateur. Il apparaît que ce débit est très élevé, de l'ordre de 500 kbits/s. Cette valeur est même plus élevée lors du flash-crowd des 5 premiers jours. Ces chiffres sont impressionants car les débits agrégés instantannés atteignent des valeurs de 800Mbits/s. À titre de comparaison, il faudrait 80 serveurs FTP transmettant à 10 Mbits/s pour obtenir des débits comparables.
Si les débits sont bons, le nombre de clients qui réussissent à obtenir une copie complète est faible, de l'ordre de 19% seulement. Néanmoins, une analyse plus fine des 81% de clients qui ne réussissent pas à finir montre que ceux-ci ne restent que peu de temps dans

le torrent (60% des clients restent moins de 1000 secondes). Au total il ne représentent que 20% des octets échangés.
En résumé, cette analyse de BT que nous avons effectuée était la première étude à large échelle de ce protocole et a apporté des premiers éléments objectifs sur les bonnes performances de cette application.

4.4.2 Évaluation des Algorithmes de choix des pièces et des pairs

Dans [79], nous avons effectué une étude avancée des algorithmes de sélection des pièces (rarest first) et des pairs (*choke*) dans BT à l'aide de mesures. Ces algorithmes forment le cœur de BT. La question à laquelle nous avons cherché à répondre est de savoir s'ils étaient efficaces ou s'il fallait les améliorer ou les remplacer. Plusieurs études ont, en effet, soutenu que ces algorithmes étaient souvent sous-optimaux et devaient être remplacés. Dans [57], les auteurs proposent de remplacer ces algorithmes par une technique dite de codage réseau, censée garantir une meilleure entropie des pièces dans le torrent. En ce qui concerne l'algorithme *choke*, plusieurs études ont souligné sa possible inéquité [96, 73]. Plutôt que de simplement classer les pairs en fonction du débit qu'ils offrent et envoyer à ceux qui offrent les meilleurs débits, ces études proposent que les quantités de données échangées soient égalisées, afin d'obtenir une équité au niveau bit.
Notre étude repose sur une collection de traces de niveau applicatif obtenues avec un client BT instrumentalisé. Ce dernier nous permet de suivre tous les échanges de pièces et de connaître les changements d'états des différents pairs dans le peer set de notre pair instrumentalisé. Nous avons travaillé sur des torrents de toute taille pour des contenus de toute sorte, comme montré dans le tableau 4.1.
En étudiant une grande variété qualitative de torrents, nous espérons que nos conclusions dépassent le simple cadre anecdotique et caractérisent le protocole BT de manière générale.

Algorithme Rarest First

L'algorithme rarest first cherche à favoriser les échanges de pièces entre pairs de manière à ce que deux pairs aient toujours des pièces à s'échanger. Nous parlons alors d'entropie idéale. Comme nous n'avons pas accès à tous les pairs d'un torrent, mais seulement à notre pair instrumentalisé et par son intermédiaire, à la connaissance des pièces que chaque pair possède à un instant donné[6], nous avons caractérisé l'entropie du torrent du point de vue de notre client en calculant deux ratios :
– le ratio $\frac{a}{b}$ du temps cumulé a pendant lequel notre pair instrumentalisé est intéressé[7] par un autre pair de son *peer set* sur le temps cumulé b pendant lequel ce pair est dans le *peer set* du pair instrumentalisé pendant que ce dernier est un *leecher* (n'a pas encore la totalité du contenu à répliquer).

[6]BT spécifie que chaque pair doit annoncer à son *peer set* toutes les pièces qu'ils reçoit - cela sert à déterminer les pièces les plus rares dans le *peer set*.

[7]un pair intéressé par un autre pair signifie que le premier est intéressé dans une pièce au moins que le second possède et qu'il ne possède pas.

TAB. 4.1 – Caractéristiques des torrents.

Torrent	**# of *seed*s**	**# of *leecher*s**	**Size (MB)**
1	50	18	600
2	1	40	800
3	1	2	580
4	115	19	430
5	160	5	6
6	102	342	200
7	9	30	350
8	1	29	350
9	12612	7052	140
10	462	180	2600
11	1	130	820
12	30	230	820
13	0	66	700
14	3	612	1413
15	3697	7341	349
16	1	50	1419
17	11641	5418	350
18	11975	4151	350
19	514	1703	349
20	20	126	184

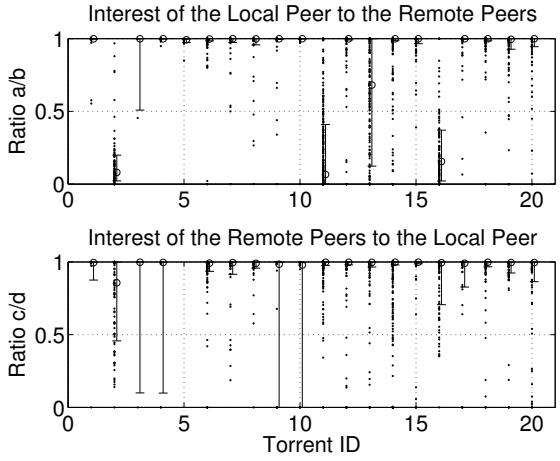


FIG. 4.24 – Entropie vue du client instrumentalisé

– le ratio $\frac{c}{b}$ où c est le temps pendant lequel un pair distant est intéressé par le pair instrumentalisé.

Dans le cas d'une entropie idéale, les ratios précédents doivent être égaux à un pour tous les pairs du *peer set*. La figure 4.24 représente les deux ratios pour les 20 torrents que nous avons étudié. Pour chaque torrent, on représente tous les ratios et on ajoute un marquage pour le 20 ième, 80 ième et 50 ième percentile de la distribution. On s'aperçoit que pour la majorité des torrents, les 20 ième percentiles sont proches de un, ce qui veut dire que l'entropie est proche de l'idéal.

Nous avons étudié plus avant les torrents pour lesquels l'entropie était "mauvaise" (les valeurs des ratios étaient plus proches de 0 que de 1). Les détails de l'analyse sont dans [79]. La conclusion est que les torrents pour lesquels l'entropie n'est pas bonne sont dans un mode transitoire caractérisé par le fait que certaines pièces ne sont détenues que par une seule *seed* qui forme alors un goulot d'étranglement. Au contraire, les pièces qui possèdent quelques copies sont répliquées à une vitesse bien supérieure. À la limite, on se retrouve dans un cas où les pièces qui ont plus d'une copie sont en fait entièrement répliquées dans le torrent et les pairs n'ont plus d'intérêt que pour la *seed* et non les uns pour les autres.

Algorithme *choke*

L'algorithme *choke* a un comportement différent en mode *leecher* et en mode *seed*. En mode *leecher*, un pair classe les débits qu'il perçoit des autres pairs et envoie des données aux 4 premiers, sans chercher à équilibrer les quantités de données envoyées. D'aucuns ont critiqué cette politique et veulent définir une politique où les quantités de données échangées seraient quasi identiques. Nous montrons dans [79] que cette définition n'est pas adaptée au cas des échanges pair-à -pair sur l'Internet où les débits des pairs sont asymétriques. De plus, nous montrons que l'algorithme actuel résulte en un bon équilibre des échanges de données. La figure 4.25 illustre ce point en montrant que les pairs qui reçoivent le plus de notre pair instrumentalisé sont ceux de qui on reçoit le plus (avec des valeurs de quantité de données similaires).

En mode *seed*, il n'y a des échanges que dans un sens. Il faut donc un critère différent.

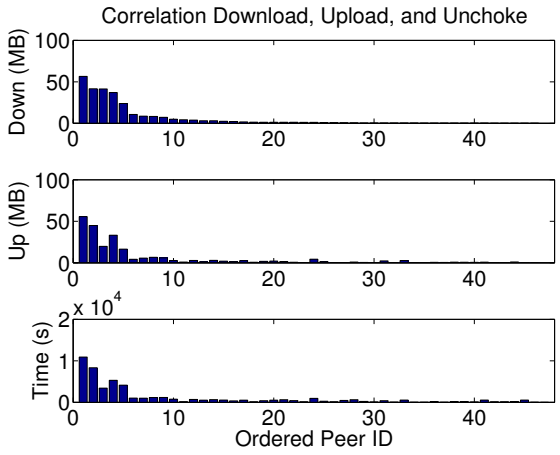


FIG. 4.25 – Quantité de données reçues et envoyées (les pairs sont ordonnés toujours dans le même ordre, celui lié aux envois)

Deux versions de l'algorithme existent. Dans l'ancienne version[8], une *seed* envoyait au 4 clients les plus rapides. Cela avait tendance à favoriser les *free-riders*, i.e les pairs qui ne font que recevoir des données et ne veulent pas "jouer le jeu". En effet, si un *free-rider* a un très bon débit de réception, il va monopoliser la *seed* au détriment du reste du torrent car il ne renvoie rien aux autres pairs du torrent. Nous avons montré dans nos expérimentations que la nouvelle version de l'algorithme en mode *seed* évitait cet écueil en équilibrant le service offert aux différents clients.

4.5 Conclusions et Perspectives

4.5.1 Conclusions

Nous avons présenté dans ce chapitre nos contributions dans le domaine de l'analyse de trafic. Nous nous sommes concentrés sur deux problèmes.
Le premier est l'analyse passive de connexions TCP. Déterminer ce qui limite le débit de connexions TCP capturées en différents points de l'Internet est essentiel pour comprendre où sont les limitations actuelles. Savoir si la limitation provient de l'application, des couches TCP de l'émetteur ou du récepteur ou de contraintes liées au réseau (congestion, goulot d'étranglement) est essentiel pour les concepteurs d'applications reseaux et les opérateurs d'accès ou de cœur. La compétition entre ces derniers exige en effet qu'ils aient une vision fine des performances du client. Ils ne peuvent plus seulement se fonder sur des indicateurs globaux, telle l'utilisation de leurs équipements physiques au sein de leur réseau. Notre première contribution est un algorithme qui permet de quantifier l'impact exact de l'application sur un flux TCP et ce, indépendamment de l'application. Cela signifie que notre méthode peut être appliquée sur un flux applicatif sur lequel on a aucun

[8]Du client BitTorrent écrit en python et maintenu par B. Cohen, l'inventeur de BT. Les autres clients, Azureus, BitComet, etc, ont tendance à répliquer les modifications apportées par B.Cohen dans son client

a priori (application client/serveur ou pair-à -pair, multimédia ou non). Cette dernière propriété est essentielle car de nouvelles applications apparaissent régulièrement sur Internet. Il est aussi possible que le flux TCP soit encrypté. Notre seconde contribution est une nouvelle méthode fondée sur l'attribution de scores obtenus à partir de différentes séries temporelles issues de l'observation d'un flux TCP pour inférer la cause qui limite le débit. Cette méthode se montre plus précise et fiable que les méthodes précédentes. Nous avons également fait des avancées dans le domaine méthodologique en proposant d'utiliser une base de données pour effectuer des analyses sur des grandes traces (plusieurs Gigaoctets). Il reste des efforts à faire pour sélectionner des agrégats de connexions sur des critères de performance (par exemple, plus de 1% de taux de perte) et/ou géographiques (chemin en terme d'AS équivalent).
Le second problème d'analyse de trafic auquel nous nous sommes intéressés est l'étude d'un protocole pair-à -pair particulier, BitTorrent. Cette application est responsable, selon la société Sandvine, de la majorité des octets observés sur certains liens aux Étatsd-Unis et en Angleterre (http ://www.sandvine.com/news/article_detail.asp ?art_id=595). Notre première étude de BT [69] a consisté à observer globalement la capacité de passage à l'échelle de l'application au travers de l'étude d'un gros torrent (180000 clients cumulés) sur une période de 5 mois. Cette étude a effectivement montré que BT était capable de tirer partie de l'ensemble des ressources (liens montant des pairs) pour offrir un débit de réception élevé par pair, même dans la période initiale où peu de pairs avaient une copie complète du fichier.
Dans un second temps [79], nous avons étudié BT dans l'objectif de comprendre si ses algorithmes de choix des pairs et des pièces ne devraient pas former à terme l'ossature de toute distribution de contenu dans l'Internet. Ce problème est critique pour la mise à jour d'applications ou d'antivirus à large échelle dans l'Internet et aussi dans les entreprises où la gestion de parcs de machine Windows reste un souci majeur des services informatiques. Notre étude a abouti aux conclusions suivantes. Tout d'abord, l'algorithme de choix des pairs de BT permet d'équilibrer au global les volumes reçus et transmis. De plus, l'algorithme de choix des pièces permet une bonne répartition des pièces par pair dans le torrent, garantissant ainsi que chaque pair trouve toujours un autre pair avec lequel il peut collaborer. Enfin, nous avons montré que les contre-performances que l'on peut parfois observer n'étaient pas directement imputables aux algorithmes de BT.

4.5.2 Perspectives

En terme de perspectives pour l'analyse de trafic, en plus de la continuation des travaux décrits dans ce chapitre (et notamment en coopération avec FT R& D), nous souhaitons travailler sur deux nouveaux domaines. Le premier est celui de la conception et de l'utilisation d'outils de mesure de capacité (valeur maximale) et de bande passante disponible (résidu utilisable) dans l'Internet. Nous avons déja débuté de premiers travaux dans ce domaine. Dans [50], nous proposons une méthode d'estimation passive de capacité à partir de traces TCP. Jusqu'à présent, l'état de l'art dans ce domaine se "limitait" à des méthodes actives qui ne se prêtent donc pas à des études à large échelle. Dans un second travail, actuellement en soumission, nous avons collecté plusieurs semaines de traces pour différents outils actifs de mesure de bande passante disponible pour tenter de répondre à

deux problèmes : (i) comment comparer ces outils entre eux et (ii) quelles informations peut on effectivement tirer de ce type d'observation en terme de cycles (diurne/nocturne, semaine/week-end), événements particuliers, etc.
Les perspectives sur la distribution de contenu concernent des études plus théoriques sur la distribution de contenu, car nos travaux à base d'analyse de trafic ont montré que le protocole possédait des propriétés uniques. Nous avons d'ores et déjà publié une première contribution dans ce domaine [118] qui concerne une étude par simulation du protocole. Nous avons mis l'accent sur certains paramètres clefs et notamment le nombre de clients connus (*peer set*) par un client donné et le nombre de clients directement contactés par un client donné. Nous avons relié ces paramètres à la structure du réseau de connexions établies entre les pairs d'un réseau BT (overalay) pour expliquer les différences de performance observées. Nous travaillons actuellement sur des extensions de ces travaux à des cas plus complexes en terme de processus d'arrivée des pairs et de leur hétérogénéité en terme de capacités d'émission/réception. Nous avons également commencé à travailler sur une adaptation de BT dans un environement sans fil ad-hoc [84]. Nous avons notamment montré que pour maximiser la réutilisation spatiale dans le réseau ad-hoc (le nombre de communications simultanées), il fallait modifier la façon dont BT construit le réseau overlay en cherchant à minimiser la distance physique entre voisins.
Le dernier axe de recheche dans le domaine de l'analyse de trafic que nous voulons développer est celui de l'analyse de données dans un contexte de détection d'intrusion. Nous travaillons actuellement, avec l'équipe de Marc Dacier, sur l'analyse des données de la plateforme leurecom.com (http ://www.leurrecom.org/) formé de pots de miels (*honeypot*[9]) distribués au travers de 25 pays et qui collecte du trafic malicieux. Nous avons déja fait un travail en commun sur la recherche de similitude temporelle de processus d'attaques [95]. Nous avons notamment montré dans ce travail que des processus d'attaque apparemment sans lien (qui ne visent pas les mêmes services et ne proviennent apparemment pas des mêmes machines) exhibent de si fortes corrélations temporelles que celles-ci ne peuvent pas être dues seulement au hasard. Notre hypothèse est qu'il existe un petit nombre d'outils extrêmement polymorphes et donc potentiellement très dangereux dans la nature. Nous continuons actuellement nos recherches sur l'analyse à large échelle des attaques et la recherche des causalités sous-jacentes (par exemple, est-ce qu'un pays précis peut être considéré comme responsable de la tendance mondiale observée pour un outil d'attaque donné ?). Nous avons également entrepris une étude préliminaire, dans le cadre du stage de fin d'étude de Chi Anh La (IFI- Hanoi) sur les attaques ciblant non les machines des utilisateurs ou les serveurs, mais l'infrastructure elle même, à savoir les routeurs BGP. Les premiers résultats obtenus, à l'aide des données de leurre.com et de données d'un téléscope (Class B "situé" en Israël), soulignent notamment la difficulté de distinguer un processus d'attaque d'une mauvaise confirguration ou du bruit ambiant naturel créé par BGP.

[9]Un pot de miel est une machine passive qui ne génère aucun trafic et qui n'offre aucun service. Toute tentative de contacter cette machine est donc potentiellement suspecte

Bibliographie

[1] "http ://tracer.csl.sony.co.jp/mawi/".

[2] "http ://www.icir.org/floyd/questions.html".

[3] "Traffic Control and Congestion Avoidance in B-ISDN", Technical Report, ITU Recommandation I.371, 1996.

[4] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network", In *SIGCOMM '04 : Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 34, pp. 121–132, New York, NY, USA, October 2004, ACM Press.

[5] J. Aikat et al., "Variability in TCP Round-trip Times", In *Internet Measurement Conference 2003*, October 2003.

[6] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks : a survey", *Computer Networks*, 47(4) :445–487, 2005.

[7] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW", In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, pp. 92–103, 1996.

[8] M. Arlitt, B. Krishnamurthy, and J. C. Mogul, "Predicting Short-Transfer Latency from TCP Arcana : A Trace-based Validation", In *Internet Measurement Conference*, 2005.

[9] K. Avrachenkov, U. Ayesta, P. Brown, and R. Nunez-Queija, "Discriminatory processor sharing revisited", In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pp. 784–795 vol. 2, 2005.

[10] H. Balakrishnan, V. Padmanabhan, S. Seshan, S. M., and R. Katz, "TCP Behavior of a Busy Internet Server : Analysis and Improvements", In *Proc. Infocom 98*.

[11] H. Balakrishnan and V. Padmanaghan, "TCP Performance Implications of Network Asymmetry", draft-ietf-pilc-asym-00.txt, Internet Draft, September 1999.

[12] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling : Investigating Unfairness", In *Sigmetrics 2001 / Performance 2001*, pp. 279–290, June 2001.

[13] S. Baset and H. Schulzrinne, "An Analysis of the Skype P2P Internet Telephony Protocol", CUCS-039-04, Department of Computer Science, Columbia University, 2004.

[14] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, Prentice-Hall, Inc. ISBN 0-13-126780-9, 1993.

[15] S. Ben Fredj, T. Bonald, A. Proutiére, G. Réginié, and J. W. Roberts, "Statistical Bandwidth sharing : a Study of Congestion at Flow Level", In *Proc. ACM SIGCOMM*, August 2001.

[16] N. Benameur, B. Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. W. Roberts, "Integrated Admission Control for Streaming and Elastic Traffic", In *QofIS*, Coimbra, Portugal, September 2001.

[17] N. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. W. y. . Roberts, "Integrated Admission Control for Streaming and Elastic Traffic".

[18] S. Blake, D. Black, M. Carlson, E. Davies, and W. W. Z. Wang, "An Architecture for Differentiated Services", *IETF RFC 2475*, 1998.

[19] E. Blanton and M. Allman, "On the Impact of Bursting on TCP Performance", In *Proceedings of Passive and Active Measurements (PAM)*, 2005.

[20] T. Bonald and J. W. Roberts, "Performance Modeling of Elastic Traffic in Overload", In *SIGMETRICS*, Cambridge, MA, USA, June 2001.

[21] J.-Y. L. Boudec, "Network Calculus Made Easy", EPFL/DI 96/218, Networking and Communication Lab, EPFL, Lausanne, Switzerland, December 1996.

[22] J.-Y. L. Boudec, "An Application of Network Calculus to Guaranteed Service Networks", *IEEE Transactions on Information Theory*, 44(3), May 1998.

[23] J.-Y. L. Boudec and P. Thiran, *Network Calculus*, Springer Verlag LNCS 2050, June 2001.

[24] P. Brown, *Partage des ressources dans les réseaux TCP/IP*, Ph.D. Thesis, Université de Nice - Sophia-Antipolis, 2005.

[25] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the nonstationarity of Internet traffic", In *ACM SIGMETRICS*, pp. 102–112, ACM Press, 2001.

[26] C.-S. Chang, "Matrix Extensions of the Filtering Theory for Deterministic Traffic Regulation and Service Guarantees", *IEEE Journal on Selected Areas in Communications*, 1998.

[27] C.-S. Chang, *Performance Guarantees in Communication Networks*, Springer-Verlag, 2000.

[28] K. Cho, M. Luckie, and B. Huffaker, "Identifying IPv6 network problems in the dual-stack world", In *NetT '04 : Proceedings of the ACM SIGCOMM workshop on Network troubleshooting*, pp. 283–288, New York, NY, USA, 2004, ACM Press.

[29] V. Cholvi, J. Echague, and J.-Y. Le Boudec, "Worst Case Burstiness Increase due to FIFO Multiplexing", In *Performance 2002*, 2002.

[30] Y. Chuh, J. Kim, Y. Song, and D. Park, "F-TCP : light-weight TCP for file transfer in high bandwidth-delay product networks", In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 1, pp. 502–508 Vol. 1, 2005.

[31] F. Ciucu, A. Burchard, and J. Liebeherr, "A network service curve approach for the stochastic analysis of networks", In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 279–290, New York, NY, USA, 2005, ACM Press.

[32] D. D. Clark, S. Shenker, and L. Zhang, "Supporting Real-time Applications in an Integrated Services Packet Network : Architecture and Mechanism", In *Proceedings of the ACM SIGCOMM Conference*, 1992.

[33] E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall Inc., Englewood Cliffs, 1980.

[34] E. G. Coffman and L. Kleinrock, "Feedback Queueing Models for Time-Shared Systems", *Journal of ACM (JACM)*, 15(4) :549–576, October 1968.

[35] B. Cohen, "Incentives Build Robustness in BitTorrent", In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

[36] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley Publishing Company, 1967.

[37] A. B. M. E. Crovella, "Self-Similarity in World Wide Web Traffic : Evidence and Possible Causes", *IEEE Transactions on Networking*, 5(6) :835–846, December 1997.

[38] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection Scheduling in Web Servers", In *USENIX Symposium on Internet Technologies and Systems (USITS '99)*, pp. 243–254, Boulder, Colorodo, October 1999.

[39] M. E. Crovella, "Performance Evaluation with Heavy Tailed Distributions", In *Job Scheduling Strategies for Parallel Processing (JSSPP)*, pp. 1–10, 2001.

[40] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System", In *ACM SIGMETRICS (poster paper)*, July 1998.

[41] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task Assignment in a Distributed System : Improving Performance by Unbalancing Load", In *ACM SIGMETRICS (poster paper)*, July 1998.

[42] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic : evidence and possible causes", In *SIGMETRICS '96 : Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 160–169, New York, NY, USA, 1996, ACM Press.

[43] R. Cruz, "A calculus for Network Delay, Part I : Network Elements in Isolation", *IEEE Transactions On Information Theory*, 37(1) :114–131, January 1991.

[44] R. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks", *IEEE Journal on Selected Areas in Communications*, 13(6) :1048–1056, August 1995.

[45] R. Cruz, "A calculus for Network Delay, Part I : Network Elements in Isolation", *IEEE Transactions On Information Theory*, 37(1) :114–131, January 1991.

[46] S. Deb, A. Ganesh, and P. Key, "Resource allocation between persistent and transient flows", *IEEE/ACM Trans. Netw.*, 13(2) :302–315, April 2005.

[47] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of Internet chat systems", In *IMC '03 : Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 51–64, New York, NY, USA, 2003, ACM Press.

[48] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of factors affecting players' performance and perception in multiplayer games", In *NetGames '05 : Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pp. 1–7, New York, NY, USA, 2005, ACM Press.

[49] R. C. Durst, G. J. Miller, and E. J. Travis, "TCP extensions for space communications", In *MobiCom '96 : Proceedings of the 2nd annual international conference on Mobile computing and networking*, pp. 15–26, New York, NY, USA, 1996, ACM Press.

[50] T. En-Najjary and G. Urvoy-Keller, "PPrate : A Passive Capacity Estimation Tool", In *E2EMON*, Vancouver, Canada, April 2006.

[51] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communication Review*, 26(3) :5–21, June 1996.

[52] A. Feldmann, J. Rexford, and R. Caceres, "Efficient Policies for Carrying Web Traffic Over Flow-Switched Networks", *ACM/IEEE Transactions on Networking*, 6(6) :673–685, December 1998.

[53] M. J. Fischer, D. Gross, D. M. B. Masi, and J. F. Shortle, "Analyzing the Waiting Time Process in Internet Queueing Systems With the Transform Approximation Method", *The Telecommunications Review*, pp. 21–32, 2001.

[54] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, 1(4) :397–413, August 1993.

[55] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level Traffic Measurement from the Sprint IP Backbone", *IEEE Network Magazine*, November 2003.

[56] J. P. Georges, E. Rondeau, and T. Divoux, "Evaluation of switched Ethernet in an industrial context by using the Network Calculus", In *Factory Communication Systems, 2002. 4th IEEE International Workshop on*, pp. 19–26, 2002.

[57] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution", In *Proc. Infocom 2005*, April 2005.

[58] M. Gong and C. Williamson, "Quantifying the properties of SRPT scheduling", In *Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium on*, pp. 126–135, 2003.

[59] M. Gong and C. Williamson, "Simulation evaluation of hybrid SRPT scheduling policies", In *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on*, pp. 355–363, 2004.

[60] J. Grieu, *Analyse et évaluation de techniques de commutation Ethernet pour l interconnexion des systèmes avioniques*, Ph.D. Thesis, Institut National Polytechnique de Toulouse, September 2004.

[61] A. A. Hanbali, E. Altman, and P. Nain, "A survey of TCP over ad hoc networks", *Communications Surveys & Tutorials, IEEE*, pp. 22–36, 2005.

[62] M. Harchol-Balter, B. Bansal, and M. Agrawal, "Implementation of SRPT Scheduling in Web Servers", In *IPDS 2001*, pp. 1–24, 2001.

[63] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing", *ACM Transactions on Computer Systems*, 15(3) :253–285, 1997.

[64] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based Scheduling to Improve Web Performance", *ACM Transactions on Computer Systems*, 21(2) :207–233, May 2003.

[65] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group", *IETF RFC 2597*, 1999.

[66] T. R. Henderson and R. H. Katz, "Transport protocols for Internet-compatible satellite networks", *Selected Areas in Communications, IEEE Journal on*, 17(2) :326–344, 1999.

[67] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks : algorithms, measurements, and implications", In *ACM SIGCOMM '04*, pp. 41–54, New York, NY, USA, 2004, ACM Press.

[68] Y. Huang and R. Guerin, "Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger ?", pp. 225–235, 2005.

[69] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice, "Dissecting BitTorrent : Five Months in a Torrent's Lifetime", In *Passive and Active Measurements 2004*, April 2004.

[70] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB", *IETF RFC 2598*, 1999.

[71] M. Jain, R. S. Prasad, and C. Dovrolis, "Socket Buffer Auto-Sizing for Maximum TCP Throughput", In *subm. to INCP 2003*, October 2003.

[72] H. Jiang and C. Dovrolis, "Why is the internet traffic bursty in short time scales ?", In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, volume 33, pp. 241–252, New York, NY, USA, June 2005, ACM Press.

[73] S. Jun and M. Ahamad, "Incentives in BitTorrent Induce Free Riding", In *3rd Workshop on Economics of P2P Systems (P2P Econ)*, August 2005.

[74] T. Karagiannis, A. Broido, M. Faloutsos, and K. C. Claffy, "Transport layer identification of P2P traffic.", In *Internet Measurement Conference*, pp. 121–134, 2004.

[75] L. Kleinrock, *Queuing Systems, Volume I : Theory*, Wiley, 1975.

[76] L. Kleinrock, *Queuing Systems, Volume II : Computer Applications*, Wiley, New York, 1976.

[77] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows.", In *Internet Measurement Conference*, pp. 201–206, 2004.

[78] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss", *Networking, IEEE/ACM Transactions on*, 5(3) :336–350, 1997.

[79] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough", In *ACM SIGCOMM/USENIX IMC'2006*, October 2006.

[80] D. Lu, P. Dinda, Y. Qiao, and H. Sheng, "Effects and implications of file size/service time correlation on Web server scheduling policies", In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, pp. 258–267, 2005.

[81] Masson, editor, *Ecoulement du trafic dans les autocommutateurs*, Gérard Hébuterne, 1985.

[82] L. Massoulié and J. Roberts, "Bandwidth sharing and admission control for elastic traffic", *Telecommunication Systems*, 15(1-2) :185–201, 2000.

[83] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", *SIGCOMM Comput. Commun. Rev.*, April 2005.

[84] P. Michiardi and G. Urvoy-Keller, "Performance Analysis of Cooperative Content Distribution forWireless Ad hoc Networks", In *To Appear , in Proceedings of IEEE/IFIP WONS 2007*, January 2007.

[85] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques", In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, volume 33, pp. 50–60, New York, NY, USA, June 2005, ACM Press.

[86] Neal Cardwell, Stefan Savage and T. Anderson, "Modeling TCP Latency", In *Proc. INFOCOM 2000*, April 2000.

[87] S. Oueslati-Boulahia, S. Ben Fredj, and J. W. Roberts, "Measurement-based admission control for elastic traffic", In *17th International Teletraffic Congress*, 2001.

[88] S. Oueslati-Boulahia and J. W. Roberts, "Impact of "Trunk Reservation" on Elastic Flow Routing", In *NETWORKING*, pp. 823–834, 2000.

[89] J. Padhye and S. Floyd, "On Inferring TCP Behavior", In *Proc. Sigcomm 2001*, August 2001.

[90] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput : A simple Model and its Empricial Validation", In *Proceedings of the ACM SIGCOMM Conference*, Vancouver, British Columbia, Canada, September 1998.

[91] J. Pang, J. Hendricks, A. Akella, R. D. Prisco, B. M. Maggs, and S. Seshan, "Availability, usage, and deployment characteristics of the domain name system.", In *Internet Measurement Conference*, pp. 1–14, 2004.

[92] V. Paxson, *Measurements and analysis of End-to-End Internet Dynamics*, Ph.D. Thesis, University of California, Berkely, April 1997.

[93] V. Paxson, A. Adams, and M. Mathis, "Experiences with NIMI", In *Passive And Active Measurement Workshop*, January 2000.

[94] T. Plagemann, V. Goebel, A. Bergamini, G. Tolu, G. Urvoy-Keller, and E. W. Biersack, "Using Data Stream Management Systems for Traffic Analysis - A Case Study", In *Passive and Active Measurements 2004*, April 2004.

[95] F. Pouget, G. Urvoy-Keller, and M. Dacier, "Time Signatures to detect multi-headed stealthy attack tools", In *18th Annual FIRST Conference*, Baltimore, Maryland, USA, June 2006.

[96] D. Qiu and S. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Perr-to-Peer networks", In *Proc. SIGCOMM 2004*, August 2004.

[97] I. A. Rai, G. Urvoy-Keller, M. Vernon, and E. W. Biersack, "Performance models for LAS-based scheduling disciplines in a packet switched network", In *ACM SIGMETRICS-Performance*, pp. –, June 2004.

[98] I. Rai, *QOS Support in Edge Routers*, Ph.D. Thesis, Télécom Paris, Institut Eurecom, Sophia-Antipolis, France, September 2004.

[99] J. W. Roberts and S. O. Boulahia, "Quality of Service by Flow Aware Networking", In *Philosophical Transactions of The Royal Society of London*, August 2000.

[100] Y. Ruan and V. S. Pai, "The origins of network server latency & the myth of connection scheduling", In *SIGMETRICS 2004/PERFORMANCE 2004 : Proceedings of the joint international conference on Measurement and modeling of computer systems*, volume 32, pp. 424–425, New York, NY, USA, June 2004, ACM Press.

[101] A. Salvatori, "LAS scheduler implementation and performance analysis", February 2005.

[102] L. E. Schrage, "The Queue M/G/1 with Feedback to Lower Priority Queues", *Management Science*, 13(7) :466–474, 1967.

[103] L. E. Schrage and L. W. Miller, "The Queue M/G/1 with the Shortest Processing Remaining Time Discipline", *Operations Research*, 14 :670–684, 1966.

[104] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive Routing of Long-lived IP Flows", In *Proc. ACM SIGCOMM*, pp. 215–226, September 1999.

[105] S. Shakkottai, N. Brownlee, , and kc claffy, "A Study of Burstinesss in TCP Flows", In *Proc. Passive & Active Measurement : PAM-2005*, April 2005.

[106] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, and kc claffy, "The RTT Distribution of TCP Flows in the Internet and its Impact on TCPbased Flow Control", , Cooperative Association for Internet Data Analysis (CAIDA), University of Illinois, 2004.

[107] S. Shenker, C. Partridge, and R. Guérin, "Specification of Guaranteed Quality of Service", *IETF RFC 2212*, 1997.

[108] S. Shenker and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", *IETF RFC 2215*, 1997.

[109] M. Siekkinen, E. W. Biersack, V. Goebel, T. Plagemann, and G. Urvoy-Keller, "IntraBase : Integrated Traffic Analysis Based on a Database Management System", In *Proceedings of IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, May 2005.

[110] M. Siekkinen, G. Urvoy-Keller, and E. W. Biersack, "On the Interaction Between Internet Applications and TCP", Technical Report, Institut Eurecom, October 2006.

[111] M. Siekkinen, G. Urvoy-Keller, E. Biersack, and T. En-Najjary, "Root Cause Analysis for Long-Lived TCP Connections", In *Proceedings of CoNEXT*, October 2005.

[112] D. X. Sun, J. Cao, W. S. Cleveland, and D. Lin, "On the Nonstationarity of Internet Traffic", In *Proc. ACM Sigmetrics*, pp. 102–112, June 2001.

[113] L. Tang and M. Crovella, "Virtual landmarks for the internet.", In *Internet Measurement Comference*, pp. 143–152, 2003.

[114] D. Towsley, S. Jaiswal, G. Iannaccone, C. Diot, and J. Kurose, "Inferring TCP Connections Characteristics from Passive Measurements", In *Proc. Infocom 2004*, March 2004.

[115] K. Tutschku, "A Measurement-based Traffic Profile of the eDonkey Filesharing Service", In *5th Passive and Active Measurement Workshop (PAM2004)*, April 2004.

[116] G. Urvoy, G. Hébuterne, and Y. Dallery, "Delay-constrained VBR Sources in a Network Environment", In *ITC-16 : 16th International Teletraffic Congress*, pp. 687–696, Edinburgh, Scotland, June 1999.

[117] G. Urvoy-Keller and E. W. Biersack, "DSS : A Deterministic and Scalable QoS Provisioning Scheme", In *QofIS'2001 : 2nd International Workshop on Quality of future Internet Services*, pp. 310–323, Coimbra, Portugal, 2001.

[118] G. Urvoy-Keller and P. Michiardi, "Impact of Inner Parameters and Overlay Structure on the Performance of BitTorrent", In *9th IEEE Global Internet Symposium 2006 in conjunction with IEEE Infocom 2006*, Barcelona, Spain, April 2006.

[119] G. van Kessel, R. Nunez-Queija, and S. Borst, "Differentiated bandwidth sharing with disparate flow sizes", volume 4, pp. 2425–2435 vol. 4, 2005.

[120] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times", In *Proceedings of Passive and Active Measurements(PAM)*, 2005.

[121] F. Wang and L. Gao, "On inferring and characterizing internet routing policies.", In *Internet Measurement Comference*, pp. 15–26, 2003.

[122] A. Wierman, N. Bansal, and M. Harchol-balter, "A note on comparing response times in the M/GI/1/FB and M/GI/1/PS queues", *Operations Research Letters*, 32(1) :73–76, January 2004.

[123] A. Wierman, M. Harchol-Balter, and T. Osogami, "Nearly insensitive bounds on SMART scheduling", In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, volume 33, pp. 205–216, New York, NY, USA, June 2005, ACM Press.

[124] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", *IETF RFC 2211*, 1997.

[125] S. Yang and G. Veciana, "Size-based Adaptive bandwidth Allocation : Optimizing the Average QoS for Elastic Flows", In *INFOCOM*, 2002.

[126] Y. Zhang, L. Breslay, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates", In *ACM Sigcomm 2002 Conference*, Pittsburgh, PA, USA, August 2002.

[127] Y. Zhang, V. Paxson, and S. Shenker, "The Stationarity of Internet Path Properties : Routing, Loss, and Throughput", , ACIRI, May 2000.

Chapitre 5

Sélection d'articles

1. I. Rai, E. W. Biersack, and G. Urvoy-Keller "Size-based Scheduling to improve the Performance of Short TCP Flows" *IEEE Network Magazine* 2005.

2. Matti Siekkinen, Guillaume Urvoy-Keller, Ernst Biersack, and Taoufik En-Najjary "Root Cause Analysis for Long-Lived TCP Connections" *In ACM CoNEXT*, Toulouse, France, October 2005.

3. G. Urvoy-Keller "On the Stationarity of TCP Bulk Data Transfers" *In Passive and Active Measurements* 2005, Juan Les Pins, France, March 2005.

4. I. A. Rai, G. Urvoy-Keller, and E. W. Biersack "Analysis of LAS Scheduling for Job Size Distributions with High Variance" *In ACM Sigmetrics 2003* pages 218–228 June 2003

5. G. Urvoy-Keller, G. Hébuterne, and Y. Dallery "Traffic Engineering in a Multipoint-to-point Network". *IEEE Journal on Selected Area in Communications* 20(4) :834–849 May 2002

Size-Based Scheduling to Improve the Performance of Short TCP Flows

Idris A. Rai, Ernst W. Biersack, and Guillaume Urvoy-Keller, Institut Eurecom

Abstract

The Internet today carries different types of traffic that have different service requirements. A large fraction of the traffic is either Web traffic requiring low response time or peer-to-peer traffic requiring high throughput. Meeting both performance requirements in a network where routers use droptail or RED for buffer management and FIFO as the service policy is an elusive goal. It is therefore worthwhile to investigate alternative scheduling and buffer management policies for bottleneck links. We propose to use the Least Attained Service (LAS) policy to improve the response time of Web traffic. Under LAS, the next packet to be served is the one belonging to the flow that has received the least amount of service. When the buffer is full, the packet dropped belongs to the flow that has received the most service. We show that under LAS, as compared to FIFO with droptail, the transmission time and loss rate for short TCP flows are significantly reduced, with only a negligible increase in transmission time for the largest flows. The improvement seen by short TCP flows under LAS is mainly due to the way LAS interacts with the TCP protocol in the slow start phase, which results in shorter round-trip times and zero loss rates for short flows.

Offering service guarantees to existing and emerging applications in the Internet has been a big challenge to Internet designers. Most of these applications either are interactive (e.g., Web applications), have delay constraints (e.g., multimedia streaming applications), or require high throughput (e.g., file transfer). A number of mechanisms were proposed in the past to enable the Internet to support applications with varying service requirements. Examples of these mechanisms include *admission control*, *active queue management*, and *scheduling*.

Admission control is meant to prevent network overload so as to ensure service for the applications already admitted [1]. Admission control is deployed at the network edges where a new flow is either admitted when network resources are available or rejected otherwise. A flow is defined as a group of packets with a common set of attributes such as (source address, destination address, source port, destination port). Although it seems plausible to use admission control in the Internet, it has not been deployed so far.

Active queue management has been proposed to support end-to-end congestion control in the Internet. The aim of active queue management is to anticipate and signal congestion before it actually occurs. One of the most important active queue management approaches is random early discard (RED) [2]. RED allows for dropping packets before the buffer overflows and was proposed to achieve various goals such as ensuring fairness among sources with different round-trip times, reducing average transfer delays, punishing unresponsive flows, and ensuring better interaction with TCP congestion control. A flow is declared unresponsive if its source does not respond to network congestion. However, in practice RED fails to offer most of these features. For example, RED cannot protect TCP flows against unresponsive flows. Similarly, for Web flows, first-in first-out (FIFO) with RED was shown to yield similar flow transfer times as FIFO with droptail.

One of the most important mechanisms to provide service guarantees is scheduling. Scheduling determines the order in which the packets from different flows are served. *Packet* scheduling in routers has been an active area of research in the last two decades, and most of the attention has focused on processor sharing (PS) types of scheduling algorithms [3]. Processor sharing has many interesting properties. It ensures a max-min fair allocation of bandwidth among competing flows, provides protection against unresponsive flows, and allows end-to-end delay bounds to be established for individual flows [4]. Despite all these properties, PS scheduling has not been widely deployed in the Internet where routers still implement FIFO scheduling.

The above mechanisms are building blocks out of which quality of service (QoS) architectures can be assembled. Internet QoS allows either guaranteeing the required service for all applications or providing better service to certain applications, for instance, by satisfying their minimum service requirements. The performance guarantees in QoS architectures are expressed in terms of delay, jitter, throughput, and loss. Examples of QoS architectures include integrated services (IntServ), differentiated services (DiffServ), and *traffic engineering* [5]. Unfortunately, like their basic building blocks, the QoS architectures have also not been widely deployed in the Internet, mainly due to lack of scalability and backward compatibility. As a consequence, the Internet today still offers only *best effort* service, with TCP the most widely used transport protocol for end-to-end data transfer.

The Variability of Internet Traffic

Studies have shown that Internet traffic exhibits *high variability*: Most TCP flows are short, while more than 50 percent of the bytes are carried by less than 5 percent of the largest flows

0890-8044/05/$20.00 © 2005 IEEE

[6, 7]. Short flows are mainly Web data transfers triggered by user interactions. As the use of the Web remains popular, Web traffic will continue not only to make up a significant fraction of Internet traffic but also to play a major role in the variability of the overall traffic distribution. The largest flows seen in the Internet originate from peer-to-peer file sharing services that have recently become very popular. The variability of Internet traffic can negatively affect the performance experienced by end users. The existing Internet uses TCP as a transport protocol and FIFO as a scheduling discipline in routers. As will be seen later, both penalize short flows.

The goal of this work is to study an alternative link scheduling policy to FIFO scheduling with droptail in packet networks in order to improve overall user perceived performance by favoring short flows without penalizing the performance of long flows too much. The class of *size-aware scheduling* policies that give service priority to packets from short flows are an ideal candidate to achieve these goals.

Size-Aware Scheduling Policies

Background

Size-aware scheduling policies have been extensively studied in the area of operating systems, where they are used to decide which job to service next. Examples of size-aware scheduling policies are Shortest Job First (SJF), Shortest Remaining Processing Time (SRPT), and Least Attained Service (LAS) [8]. These policies have been known for several decades, but have not been proposed so far for packet networks. The reason is that when analyzed under the $M/M/1$ queue, where service times do not exhibit high variability, these policies are known to favor short jobs but also to penalize the largest jobs a lot. The high variability of flow sizes observed in the Internet prompted researchers to revisit size-aware policies and study their performances for workloads exhibiting high variability.

SJF is a non-preemptive scheduling policy that gives service to the shortest job. A newly arriving shortest job must wait for the job in service to complete before it can receive service. SRPT is a preemptive version of SJF, which favors short jobs by giving service to the job that has the shortest remaining processing time. In order to know the remaining processing time, though, one needs to know the total service required by the job (the size of the job). Fair sojourn protocol (FSP) [9] is a very recent scheduling discipline that combines some elements of PS and SRPT. FSP services next the job that would complete service first among all jobs when the service discipline is PS. FSP has been shown [9] to ensure that the response time is never higher than that achieved by PS. However, FSP requires knowledge of job sizes, as does SRPT.

Suitable policies for link scheduling in the Internet must not require any knowledge about flow sizes. LAS is a size-aware scheduling policy that does not need to know the flow size. In this article we study the interaction of LAS with TCP congestion control and use simulation to show the benefits of LAS in packet networks.

LAS Scheduling

LAS is a preemptive scheduling policy that favors short jobs without prior knowledge of job sizes. To this end, LAS gives service to the job in the system that has *received the least amount of service*. In the event of ties, the set of jobs having received the least service shares the processor in PS mode. A newly arriving job always preempts the job currently in service and retains the processor until it departs, the next arrival occurs, or it has obtained an amount of service equal to that received by the job preempted on arrival, whichever occurs first. In the past, LAS has been believed to heavily penalize large jobs. However, it has recently been shown that the mean response time of LAS highly depends on job size distribution [10]. LAS is also known as Foreground-Background (FB) [8] or Shortest Elapsed Time (SET) first scheduling.

In operating systems scheduling, the term *job* is commonly used to denote a piece of workload that arrives to the system all *at once*. Given this definition, a *flow* in a packet network cannot be considered a job since a flow does not arrive at the link at once. Instead, the source transmits a flow as a sequence of packets, possibly spaced out in time, that are in turn statistically multiplexed with packets from other flows. Also, when TCP is used as transport protocol, due to the closed-loop nature of TCP, the treatment given to an individual packet may affect the temporal behavior when transmitting the remaining packets in the flow. For these reasons, we are not sure if the analytical results on LAS that have been derived for jobs can be directly be applied to flows. Therefore, we use simulation to investigate the performance of LAS for flows.

In the context of flows, the next packet serviced under LAS is the one that belongs to the flow that has received the *least amount* of service. By this definition, LAS will serve packets from a newly arriving flow until that flow has received an amount of service equal to the amount of least service received by a flow in the system before its arrival. If two or more flows have received an equal amount of service, they share the system resources fairly. In packet networks, LAS is not only a scheduling discipline as in operating systems but also comprises a buffer management mechanism: when a packet arrives to a queue that is full, LAS first inserts the arriving packet at its appropriate position in the queue and then drops the packet at *the end of the queue*. Therefore, LAS gives buffer space priority to short flows as the packet discard mechanism under LAS biases against flows that have utilized the network resources the most, which differs from droptail or RED buffer management.

Simulation results in this article show that LAS significantly reduces the mean transmission time and loss rate of short TCP flows from that of a FIFO scheduler with droptail, with a small increase in mean response time of large flows. The results also show that under moderate load values a long-lived ftp flow under LAS is not locked out when competing with short TCP flows. However, the performance of the long-lived flow under LAS deteriorates severely when competing at high load against short flows. For this reason, we propose another class-based LAS policy that is able to guarantee service to long-lived TCP flows if needed.

The Impact of TCP Congestion Control on Short TCP Flows

TCP Congestion Control

The main task of TCP congestion control is to adjust the sending rate of the source in accordance with the state of the network. For this purpose, TCP limits the amount of outstanding data. The congestion window (`cwnd`) represents the maximum amount of data a sender can have sent and for which no acknowledgment was yet received. In particular, when the source starts sending data, TCP conservatively initializes `cwnd` to one packet and then doubles `cwnd` for every window worth of ACKs received until congestion is detected. This behavior is referred to as slow start (SS). Network congestion is identified by packet losses. When loss is detected, `cwnd` is reduced to react to congestion and reduce the risk of losing more packets. The *congestion avoidance* algorithm is then used to slowly increase `cwnd` in a linear manner by one packet for every window worth of packets acknowledged. This is called the congestion avoidance (CA) phase.

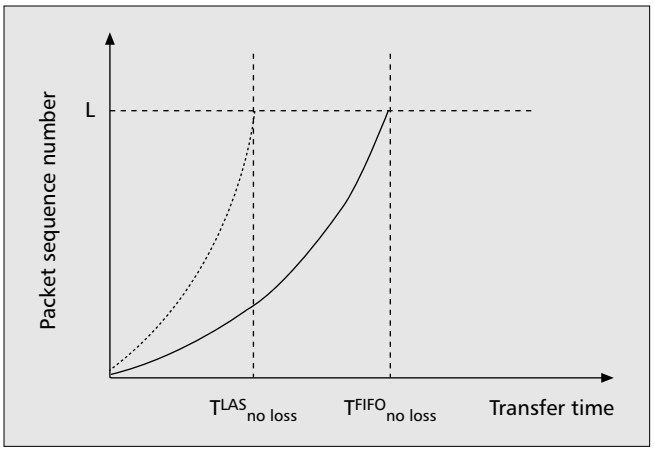


Figure 1. *Slow start behavior for short TCP flows with no packet loss under both polices. Dotted line: LAS; solid line: FIFO.*

TCP uses two error recovery mechanisms. The first is timeout-based and relies on the retransmission timeout (RTO) to decide when to retransmit. At the beginning, the RTO is initialized to a conservatively chosen value initial timeout (ITO), which is often set to a value as high as 3 s and later updated using the measured round-trip time (RTT) samples. When a source does not receive during an RTO period the ACK for a packet, the timer expires. The source then first retransmits the packet, goes into SS, and sets `cwnd` to one packet. The second error recovery mechanism in TCP is called *fast retransmit*: a packet i is considered lost if the sender receives four times in a row the ACK for packet $i - 1$. The sender then immediately retransmits packet i, sets `cwnd` to half its previous value, and goes into CA. The details of the subsequent behavior of TCP depend on the version of TCP.

The Impact of LAS on Short TCP Flows

A closer look at the behavior of TCP congestion control reveals a number of factors that negatively affect the performance of short flows. Short flows complete their data transfer during the SS phase. However, even if the network is completely uncongested, the duration of a transfer during SS is dominated by the RTT of a connection, which is in general much longer than the time to actually transfer the packets of short flows. A short flow that has to send $n = 2^{k+1}$ packets and does not experience any loss of packets or ACKs will need $R(n) = k \times RTT$, where RTT is defined as $RTT = 2(T_{e2e} + T_Q)$, and T_{e2e} denotes the end-to-end propagation and transmission delay and T_Q the queuing delay. Under LAS, the first packets in a flow will have service priority and see no or little queuing delay T_Q, which reduces the transfer time $R(n)$ for short flows from that in FIFO.

Figures 1–3 compare the qualitative behavior of TCP under LAS and under FIFO with droptail for different scenarios. Figure 1 plots the sequence number of received packets vs. transfer time. It illustrates the impact of LAS and FIFO with droptail on the transfer time $R(n)$ where there is no loss of packets. We see that a short flow of L packets transmits its packets faster under LAS than under FIFO. The difference is mainly due to a lower RTT under LAS than under FIFO. As a result, the transmission time of the same flow under LAS ($T^{LAS}_{no\ loss}$) is lower than the transmission time under FIFO ($T^{FIFO}_{no\ loss}$).

Short TCP flows also have poor loss recovery performance since they often do not have enough packets in transit to trigger the fast retransmit mechanism. Instead, they must rely on the retransmission timeout to detect losses, which can significantly prolong the transmission time of a short TCP flow; TCP uses its own packets to sample the RTTs and estimate the appropriate RTO value. When a lost packet is among the first packets of a flow, which is very likely for short flows, TCP uses the ITO as the timeout value to detect the loss. Thus, losing packets in the SS phase can significantly prolong the transmission time of a short flow.

The packet discard mechanism under LAS significantly reduces packet losses during the initial SS phase for all flows, regardless of their sizes. Instead, LAS mainly drops packets from large flows that have already transmitted a certain amount of data (have low service priority). Hence, under LAS there is hardly a need for packet error recovery to rely on the RTO to detect packet loss. This is an important feature of LAS for short flows that FIFO scheduling cannot offer. Figure 2 shows the sequence number vs. time plot for a short flow of size L packets that does not experience losses under LAS but loses packet $(L_l + 1)$ under FIFO. The short flow under FIFO retransmits the packet after time RTO^{FIFO}, which prolongs the transmission time T^{FIFO}_{loss} of the flow.

Consider also the case when a short flow under both policies loses packet $(L_l + 1)$. Figure 3 shows that the response time under LAS is still lower than that under FIFO. It is important to note here that RTO^{FIFO} is bigger than RTO^{LAS}. This is because the computations of RTO in TCP at any time depend directly on previous RTT sample values. These values are smaller under LAS than under FIFO due to the absence of queuing delay for the packets transmitted during SS under LAS.

In practice, packet losses are likely to occur sooner under FIFO with droptail than under LAS. In this case, FIFO is very likely not to have enough RTT samples to compute the RTO, and TCP must use the high ITO value to detect the losses. This again lengthens the transmission time of short flows under FIFO.

To summarize, we see that the two main factors that allow LAS to improve the performance of short TCP flows are:

- The reduction of RTT during the SS phase
- The reduction of losses seen by short flows

Note that the arguments presented in this section apply not only to short flows but also to long TCP flows during their initial SS phase.

Related Work

Closely related to our work are network-based approaches that improve the performance of short flows. A number of two-queue-based policies have been proposed [11–14] that give service priority to short flows. These policies use a variable threshold value, `th`, to differentiate between short and long flows: the first `th` packets of a flow are enqueued in the first queue, and the remaining ones in the second queue. Each queue is served in FIFO order, and packets from the second queue are only served if the first queue is empty. Two-queue-based disciplines reduce the mean transfer times of

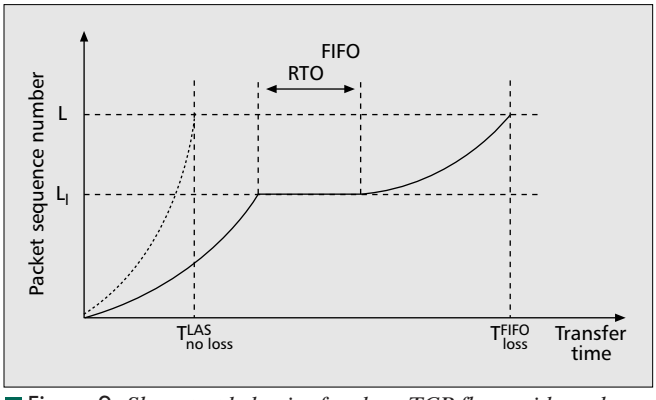


Figure 2. *Slow start behavior for short TCP flows with packet loss under FIFO. Dotted line: LAS; solid line: FIFO.*

short flows from those with FIFO scheduling. For example, simulation results show that the proposed short flows differentiating (SFD) policy reduces the mean transmission time of short flows by about 30 percent [12]. The work in [13] shows a reduction of mean transfer time by about 80 percent for medium-sized flows between 50–200 packets from simple FIFO with RED. A recent scheme called RuN2C [14] proposes a two-queue policy similar to the policy proposed in [12]. However, LAS offers lower transfer times to short flows than RuN2C because packets of short flows under RuN2C are serviced using FIFO, and they must always wait behind any packets they meet in the first queue upon their arrival. In addition, a practical difficulty with all threshold-based schemes is the proper tuning of `th` to capture all short flows.

Williamson and Wu [15] proposed a Context Aware Transport/Network Internet Protocol (CATNIP) for Web documents. This work is motivated mainly by the fact that loss of the first packets of a TCP flow has a much more negative impact on the transmission time of a flow than loss of other packets. CATNIP uses application layer information, the Web document size, to provide explicit context information to the TCP and IP protocols; using CATNIP, IP routers identify and mark packets of a flow differently to avoid the most important packets getting dropped. Experiments with CATNIP in routers show that it can reduce the mean Web document retrieval time by 20–80 percent. In contrast to CATNIP, LAS does not require knowledge of flow sizes.

Simulation Results

To evaluate LAS, we simulate it with the ns-2 network simulator for a case where a pool of clients request Web objects from a pool of servers. The simulated Web traffic profile is obtained by setting the distributions of intersession, interpage, and interobject time (all in seconds), session size (in Web pages), page size (in objects), and flow size (in packets) as seen in Table 1. The parameters used in the table result in a total load for Web traffic of $\rho = 0.7$. Exp(μ) denotes exponential distribution with mean μ. We consider a Pareto *object size* distribution denoted as $P(a, \alpha, \bar{x})$, where a is the minimum possible object size, α is the shape parameter, and $\bar{x}$ is the mean object size. The Pareto distribution is used since it exhibits high variability and models flow size distributions well as empirically observed in the Internet. Note that the transfer of each object will result in a new flow.

Pages/session	Objs/page	Intersession	Interpage	Interobj	Obj size
Exp(60)	*Exp(3)*	*Exp(8)*	*Exp(5)*	*Exp(0.5)*	*P(1,1.2,12)*

Table 1. *Web traffic profile.*

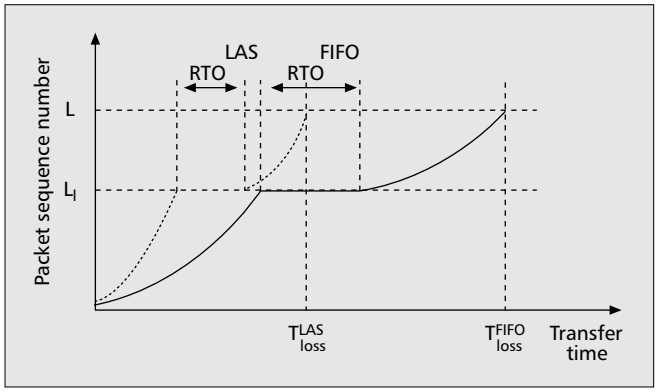


Figure 3. *Slow start behavior for short TCP flows with packet loss under both policies. Dotted line: LAS; solid line: FIFO.*

In this section we use simulations to show the benefits of LAS in terms of reducing the mean transfer times and loss rates for short flows from those of FIFO scheduling with droptail.

Mean Transmission Time

The transmission time of a flow is the time interval starting when the first packet of a flow leaves a server and ending when the last packet of the flow is received by the corresponding client. Figures 4a and 4b show the mean transmission time for LAS and FIFO as a function of flow size and percentile of object size, respectively. The pth percentile is flow size x_p such that $F_x(x_p) = p/100$, where $F_x(x)$ is the cumulative distribution

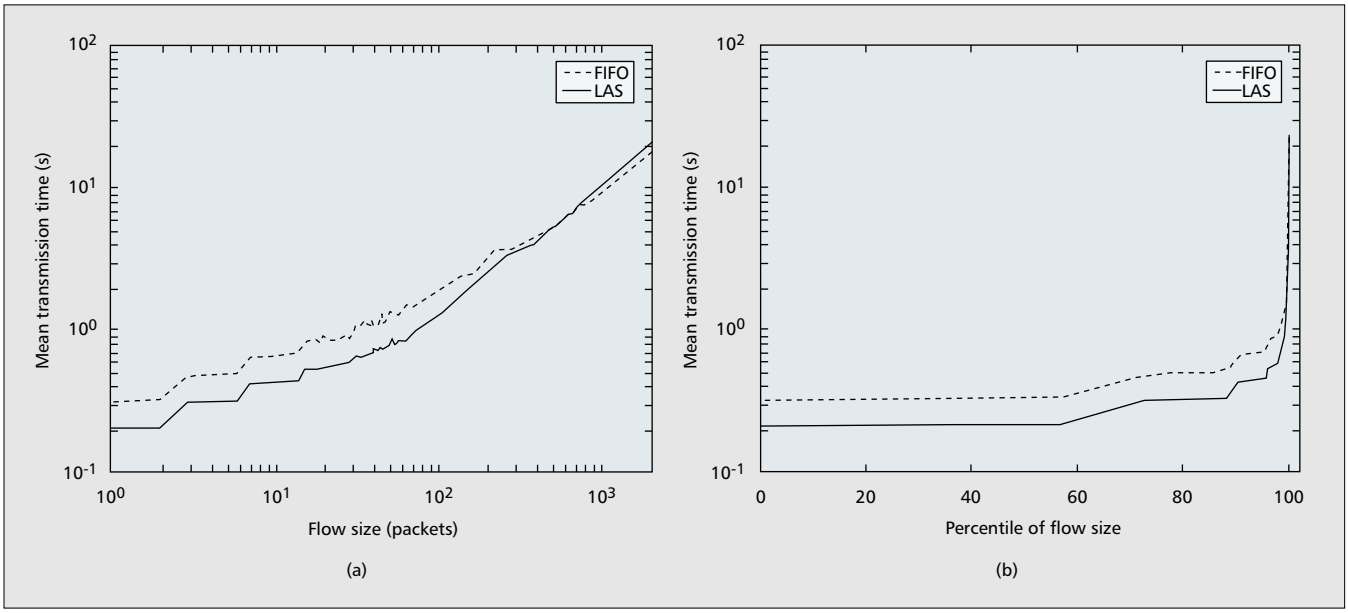


Figure 4. *Mean transmission time of Pareto Web flow sizes, load $\rho = 0.7$: a) flow sizes; b) percentiles.*

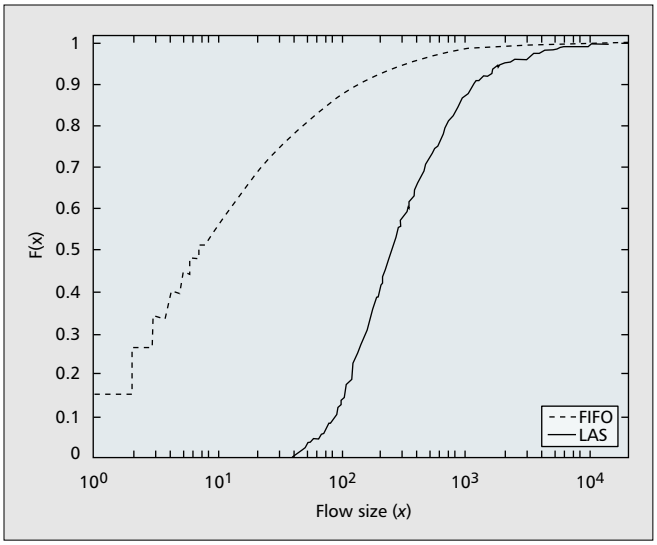


Figure 5. *CDF of flows that experience loss for Pareto distributed Web flow sizes, load* ρ = *0.7.*

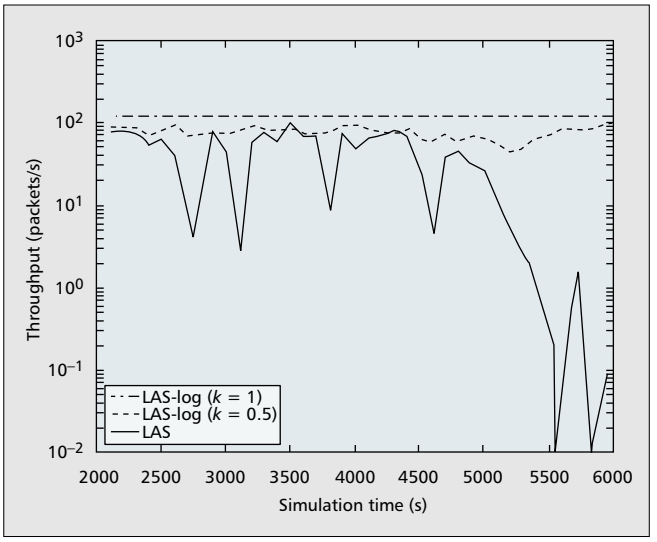


Figure 6. *Throughput of the long-lived ftp flow under LAS and LAS-log* (k)*:* k = *0.5 and* k = *1.*

function of x. We see that LAS significantly reduces the mean transmission time with a negligible penalty to large flows. More than 99 percent of flows benefit from LAS.

We also observe that the mean transmission time of large flows under LAS are only slightly higher than under FIFO. When experiencing losses, large flows are likely to have large congestion windows that permit the use of fast retransmission to recover the losses, which helps reduce the transfer time of large flows from that of recovery via timeout. We also investigated the performance of LAS at other load values and observed that short flows also benefit under LAS at lower load values like ρ = 0.5. However, the performance of short flows under LAS compared to FIFO improves with increasing load values.

Loss Rate

The overall performance of TCP depends to a certain degree on how efficiently it can detect and reduce network congestion. In general, short TCP flows are not capable of reacting to congestion. When a short flow with very few packets experiences loss(es), the congestion window is too small for reducing its size to have any impact on reducing congestion. On the other hand, network congestion is alleviated when a source with a large congestion window detects loss. In this case, reducing the congestion window by halving the value of `cwnd` reduces overall network load and therefore congestion. Hence, reducing the congestion windows of a few flows with large congestion windows can be sufficient to react to network congestion.

LAS speeds up the SS phase of all flows (including the largest) and enables sources to reach the CA phase faster. This improves the throughput performance of TCP as TCP flows under LAS, especially large flows, are likely to detect packet losses during the CA phase. In routers with FIFO scheduling, short flows are as likely as large flows to see their packets dropped, which deteriorates the throughput performance of short flows without alleviating congestion. Under LAS, when the queue is full, LAS first inserts an incoming packet at its position in the queue, and then drops the packet at the tail of the queue. It turns out that LAS very much protects flows from losing packets during the initial SS phase and most likely drops packets from large flows. Apparently, speeding up the SS phase and avoiding packet losses in this phase make LAS a very effective policy for short flows.

Figure 5 presents simulation results that illustrate the positive impact of LAS on the packet losses seen by short flows. The simulation was performed with the parameters of Table 1 for Pareto distributed flow sizes. We observe that no short flow of less than 40 packets experiences packet loss under LAS, whereas short flows up to 40 packets make up about 80 percent of the flows that experience packet losses under FIFO. In general, it is essentially the short flows that benefit from LAS in terms of significantly reduced packet losses. The overall loss rate under LAS is about 3.3 percent, whereas under FIFO with droptail it is about 3.9 percent. The number of flows that experience one or more packet losses under FIFO is 12,212, almost twice the number of flows that experience loss under LAS, which is 6763 out of about 47,800 flows.

A Single Long-Lived Flow Competing with Web Background Traffic

We have seen that the mean transmission times of the largest Web flows under LAS and FIFO are quite similar. Similarly, we observed that the loss rates of the largest Web flows under LAS and FIFO do not differ too much. In this section we investigate the performance of long-lived TCP flows that are larger than the largest Web flows. One expects that a long-lived TCP flow, when competing against short TCP flows, must starve at some point as LAS favors short flows. However, simulation results show that this is true only at high load close to overload. If the Web traffic makes up for a high load, a long-lived TCP flow under LAS will experience starvation. This is clearly a drawback of LAS that can heavily affect mission-critical long-lived flows under high network load. To protect these flows from starvation, we propose a modified LAS policy called **LAS-log(*k*)** that provides service differentiation for flows that require improved service. To illustrate this point, we consider a two-class policy with ordinary and priority flows. Assume a flow that has received x amount of service. If this flow belongs to the ordinary class, its service priority $P(x)$ under LAS-log(k) is x. If the flow belongs to the priority class, its service priority $P^H(x)$ is $(log_2(x))^{1/k}$, with $k > 0$. The packets of flows from the two classes are serviced just as under simple LAS policy based on their service priority. If $(log_2(x))^{1/k} < x$, (which is the case, e.g., for $k = 0.5$ and $x > 4$) a priority flow will receive priority service over an ordinary flow that has received the same amount of service.

We studied the performance of a long-lived ftp flow under LAS and LAS-log(k) using a setup where the long-lived flow competes against Web traffic across a bottleneck link. In LAS-log, the ftp flow will be in the priority class and all the Web flows in the ordinary class. All access links have a capacity of 10 Mb/s

except the access link for the ftp source, which has a capacity of 1 Mb/s. Otherwise, when we study the performance under LAS-log, the fact that the ftp flow is in the priority class will allow this flow to hog all the bottleneck link bandwidth, which is not realistic.

Figure 6 compares the throughput of the long-lived ftp flow under plain LAS and LAS-log with $k = 0.5$ and $k = 1$ as a function of simulation time when the load due to Web traffic is about $\rho = 0.7$. We first note that the throughput under plain LAS is highly variable and decreases to zero after a long simulation time. In contrast, the LAS-log discipline improves the throughput of the flow. For $k = 0.5$, the ftp flow achieves an almost constant throughput of around 90 packets/s. For $k = 1$, the performance of the ftp flow reaches about 120 packets/s and does not deteriorate in time. Thus, we conclude that LAS-log improves the service of long-lived priority flows to the extent that these flows are not affected by ordinary flows.

Implementation Issues

LAS for link scheduling is non-preemptive, and the packet to be served next will be taken from the head of the queue. To implement LAS link scheduling, we need a *priority assignment unit* whose functions include computing the service priority of each arriving packet and inserting the packet to its appropriate position in the priority queue. The priority assignment unit needs to track for each flow f the amount of bytes S_f served so far. S_f is used to compute the service priority of a packet of flow f, which is needed to determine where to insert this packet into the priority queue. Recall that the lower the value of the service priority, the higher the priority, and the closer to the head of the queue the packet is inserted. S_f is initialized with 0. When a packet of size P for flow f arrives, the priority assignment unit executes the following operations:

- Use S_f to compute the service priority and insert the packet into the priority queue.
- Update S_f to $S_f := S_f + P$.

There are a number of methods to implement a priority queue for high-speed packet networks. The work in [16] presents a hardware implementation called *pipelined priority queue architecture* of a priority queue that scales well with respect to the number of priority levels and queue size. In particular, the implementation can support data rates up to 10 Gb/s and over 4 billion priority levels.

Keeping per-flow state is known to be a difficult task in the core of the Internet due to the presence of a large number of simultaneous active flows. We do not envision deploying LAS in the *core* of the Internet. This is also not necessary, since the core of the Internet is not congested. Congestion in the Internet typically occurs at the edges [17], where the number of flows simultaneously active is moderate. Thus, deploying LAS at the edges should be sufficient to reap the benefits of LAS in terms of reducing the response time of short TCP flows.

Conclusion

LAS scheduling for jobs has been known for decades, but it was never proposed before for packet networks. This article argues that LAS could be deployed in packet networks at the bottleneck links. Using a realistic traffic scenario that captures the high variation of flow sizes observed in the Internet, we see that LAS deployed in the bottleneck link:

- Significantly reduces the mean transmission time and loss rate of short TCP flows
- Does not starve long-lived TCP flows except at load values close to overload

The performance degradation under LAS seen by long TCP flows can be avoided if we use a modified version called LAS-log.

Other scheduling policies such as PS need to be deployed in every router along the path from source to destination. On the other hand, incremental deployment of LAS at a bottleneck router is effective as short flows that pass through the bottleneck will see their response times reduced. As LAS is deployed in more bottleneck routers, more short flows will benefit.

There exist various proposals on how to improve the performance of short TCP flows by changing the mechanisms deployed in a router. However, our proposal has distinctive advantages: LAS is conceptually very simple as there is no parameter that needs to be tuned. The priority mechanism of LAS improves the transmission time of short flows much more than schemes that propose to use two separate FIFO queues for short and long flows.

Acknowledgments

We would like to thank the reviewers and the Editor-in-Chief for their constructive comments.

References

[1] J. Roberts, "Internet Traffic, QoS, and Pricing," *Proc. IEEE*, vol. 92, no. 9, Aug. 2004, pp. 1389–99.
[2] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Net.*, vol. 1, no. 4, Aug. 1993, pp. 397–413.
[3] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks," *IEEE Proc.*, Oct. 1995.
[4] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks-The Single Node Case," *Proc. IEEE INFOCOM*, 1992, pp. 914–24.
[5] X. Xiao and L. Ni, "Internet QoS: A Big Picture," *IEEE Network*, Mar./Apr. 1999, pp. 8–18.
[6] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. Net.*, vol. 3, June 1995, pp. 226–44.
[7] K. Claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," *Proc. INET*, July 1998.
[8] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*, Wiley, 1976.
[9] E. J. Friedman and S. G. Henderson, "Fairness and Efficiency in Web Servers," *Proc. ACM SIGMETRICS*, June 2003, pp. 229–37.
[10] I. A. Rai, G. Urvoy-Keller, and E. W. Biersack, "Analysis of LAS Scheduling for Job Size Distributions with High Variance," *Proc. ACM SIGMETRICS*, June 2003, pp. 218–228.
[11] L. Guo and I. Matta, "The War between Mice and Elephants," *Proc. 9th IEEE Int'l. Conf. Net. Protocols*, Riverside, CA, 2001.
[12] X. Chen and J. Heidemann, "Preferential Treatment for Short Flows to Reduce Web Latency," *Computer Networks: Int'l. J. Comp. and Telecommun. Net.*, vol. 41, no. 6, 2003, pp. 779–94.
[13] L. Guo and I. Matta, "Differentiated Control of Web Traffic: A Numerical Analysis," *SPIE ITCOM 2002: Scalability and Traffic Control in IP Networks*, Aug. 2002.
[14] K. Avrachenkov *et al.*, "Differentiation between Short and Long TCP Flows: Predictability of the Response Time," *Proc. IEEE INFOCOM*, Mar. 2004.
[15] C. Williamson and Q. Wu, "A Case for Context-Aware TCP/IP," *Perf. Eval.. Rev..*, vol. 29, no. 4, Mar. 2002, pp. 11–23.
[16] R. Bhagwan and B. Lin, "Fast and Scalable Priority Queue Architecture for High-Speed Network Switches," *Proc. IEEE INFOCOM*, 2000, pp. 538–47.
[17] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based Inference of Internet Link Lossiness," *Proc. IEEE INFOCOM*, 2003.

Biographies

ERNST BIERSACK [M'88, ACM'84] (erbi@eurecom.fr) received his M.S. and Ph.D. degrees in computer science from Technische Universitat Munchen, Germany. Since March 1992 he has been a professor of telecommunications at Institut Eurecom, Sophia Antipolis, France. For his work on synchronization in video servers he received (together with W. Geyer) the Outstanding Paper Award of the IEEE Conference on Multimedia Computing and Systems in 1996. For his work on reliable multicast he received (together with J. Nonnenmacher and D. Towsley) the 1999 W. R. Bennet Price of the IEEE for the best original paper published in 1998 in *ACM/IEEE Transactions on Networking*.

IDRIS RAI (i.rai@comp.lancs.ac.uk) has an M. S. degree from Bilkent University, Turkey, and did his Ph.D. at Institut Eurecom. He is now with the University of Lancaster, United Kingdom.

GUILLAUME URVOY-KELLER (urvoy@eurecom.fr) received an Engineering degree from INT, France, in 1995 and a Ph.D. in computer science from University Paris 6, France, in 1999. Since 2000, he has been an assistant professor at Institut Eurecom, Sophia Antipolis, France. His research interests cover traffic analysis, QoS, and peer-to-peer networks.

Root Cause Analysis for Long-Lived TCP Connections

M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, T. En-Najjary
Institut Eurécom
BP 193, 06904 Sophia-Antipolis Cedex, France
Email: {siekkine,urvoy,erbi,ennajjar}@eurecom.fr

ABSTRACT

While the applications using the Internet have changed over time, TCP is still the dominating transport protocol that carries over 90% of the total traffic. Throughput is the key performance metric for long TCP connections. The achieved throughput results from the aggregate effects of the network path, the parameters of the TCP end points, and the application on top of TCP. Finding out which of these factors is limiting the throughput of a TCP connection – referred to as TCP root cause analysis – is important for end users that want to understand the origins of their problems, ISPs that need to troubleshoot their network, and application designers that need to know how to interpret the performance of the application. In this paper, we revisit TCP root cause analysis by first demonstrating the weaknesses of a previously proposed flight-based approach. We next discuss in detail the different possible limitations and highlight the need to account for the application behavior during the analysis process. The main contribution of this paper is a new approach based on the analysis of time series extracted from packet traces. These time series allow for a quantitative assessment of the different causes with respect to the resulting throughput. We demonstrate the interest of our approach on a large BitTorrent dataset.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Network**]: Network Operations—*Network monitoring*; C.2.5 [**Computer Communication Network**]: Local and Wide-Area Networks—*Internet*

General Terms

Algorithms, Measurement, Performance

Keywords

Internet, TCP root cause, throughput, time series.

1. INTRODUCTION

Motivation: During recent years, the Internet traffic has experienced a massive growth as the number of users skyrocketed. This applies also to the amount of traffic per user, as the capacities of access links have increased by several order of magnitudes. New Internet applications, such as peer-to-peer (P2P), have emerged and the relative importance of HTTP and FTP has decreased. On the other hand, TCP still remains the dominating transport protocol that conveys the vast majority of the traffic – typically more than 90% of the bytes. As a consequence, the behavior and performance of TCP in the Internet is a major concern. The heterogeneity of today's applications running on top of TCP and the diversity of the environments they are used in, imply that a meaningful analysis can only be done during their operation in the Internet.

Throughput is the most important performance measure for long TCP connections. The achieved throughput represents the aggregate effects of the network path, the end points, and the application. Our research tries to find out which of them are responsible for limiting the throughput of a given TCP connection at a given time instant. Knowledge about these root causes can be used by Internet Service Providers (ISP) for quality of service evaluation and troubleshooting. Traffic modeling is another area that would benefit from this knowledge, leading to more accurate workload models of TCP traffic. It is equally important for Internet application designers to know when the limiting factor is the network or the TCP end points and when it is the application.

Approach: We adopt an approach that requires as input bidirectional packet header traces captured at a measurement point along the path from source to destination and produces as output quantitative information about the limitation causes of TCP's throughput per connection. We assign a score to each of these limitations and track the evolution of limitation scores with time. For this purpose, we base our approach on a set of time series generated from the (TCP and IP) packet headers. We focus on long transfers where TCP slow start no longer dominates the throughput achieved.

As stated above, we distinguish three main classes of root causes: (i) limitations due to the application, (ii) limitations due to the TCP end-hosts and (iii) limitations due to the network. We apply a divide and conquer approach to infer the limitation causes. First, the periods where the throughput is determined by the application are isolated. The remaining trace data consists of so called *bulk transfer* periods. We then apply a set of tests to derive the most likely cause (or causes) that explains the performance of each bulk transfer period. Whenever possible, the algorithms are validated using live measurements in the Internet that are compared against the results given by Web100 [10]. We also used NIST Net [4] to create specific conditions for a given transfer.

Zhang et al. [18] performed pioneering research work into the origins of Internet TCP throughput limitation causes. They defined taxonomy of rate limitations (application, congestion, bandwidth,

CoNEXT'05, October 24–27, 2005, Toulouse, France.

sender/receiver window, opportunity and transport limitations) that we build on and introduced the TCP Rate Limitation Tool (T-RAT). T-RAT turned out to suffer from a number of limitations. First, to identify rate limitation, T-RAT needs to identify so called "flights" of packets. These flights often cannot be identified, as we will see in section 3, which undermines the main premise of T-RAT. Second, T-RAT breaks long connections into "flows" of at most 256 consecutive packets. In contrast, we perform true connection level analysis.

Challenges: The problem is very challenging for several reasons. Operating at connection level complicates the analysis because with long connections, we have a higher probability to observe several limitation causes over different periods of time. For instance, some Internet applications such as BitTorrent [8] or HTTP1.1 operate by switching between active transfer periods and passive keep-alive periods. Our first challenge is to detect those different periods and analyze them separately.

The great number of parameters that influence the behavior of a TCP connection is also a major issue: round-trip time (RTT), receiver advertised window, link capacities, available bandwidth, delayed acknowledgment, and TCP version to name a few.

Contributions: The contribution of this paper is threefold: First, in section 2, we discuss the problem of inferring causes for TCP's transmission rate limitation by elaborating more on the limitation concepts themselves with respect to the approach taken by T-RAT. Second, in section 3, we demonstrate through simulations that an important characteristic of a TCP transfer, the so called flights, have in many cases a very different form than the one assumed in T-RAT. Third and most importantly, we provide a set of algorithms to infer causes that limit the throughput of a given TCP transfer (sections 4 and 5), and apply the algorithms to an example set of real Internet traffic (section 6).

2. CAUSES FOR RATE LIMITATION

In this section, we discuss the different rate limitation causes that we want to infer. While the classification is inspired by T-RAT, we extend the scope of their work, and exemplify the difficulties of identifying certain causes or assessing the impact of others. We present the causes in a top down manner, starting from the application level down to the network level.

2.1 Application

The application operating on top of TCP can be the cause for the throughput achieved. In this case, TCP is not able to fully utilize the transport or network layer resources because the application does not produce data fast enough. There exist two scenarios where the application is the limitation cause.

In the first scenario, the application is producing small amounts of data at a relatively constant rate for the TCP layer. This results in small bursts of packets, in the extreme case a single packet of size less than the maximum segment size of the connection. Typical examples are live streaming applications such as Skype [1] that transfer data over TCP at a constant rate of 32 Kbit/s. Also, applications that use permanent TCP connections and send keep-alive packets during inactive periods, fall in this category (BitTorrent exhibits this behavior during choke periods -see [8]).

In the second scenario the application is producing data in bursts separated from each other by idle periods. An example of such behavior is web browsing with persistent HTTP connections.The user clicks on a link to load a web page, causing a transfer period, reads the page, causing an idle period, and clicks on another link, causing another transfer period.

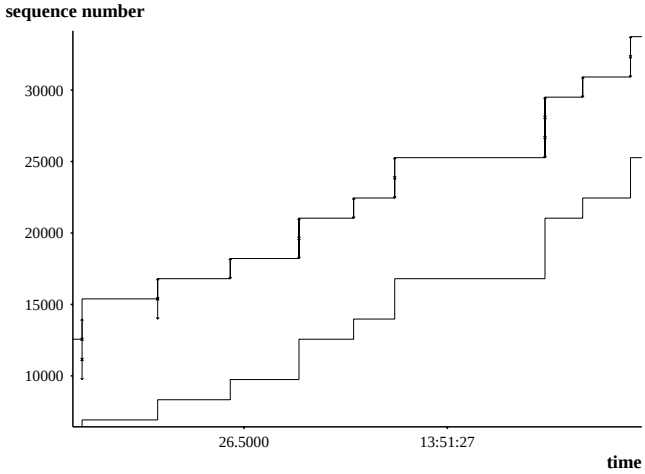


Figure 1: A piece of a receiver window limited connection.

2.2 TCP End-Point Limitations

The achieved throughput of TCP can be limited by the size of the buffers allocated at the two end-points of a connection. The receiver buffer (between the TCP layer and the application layer) constrains the maximum number of outstanding bytes the other end is allowed at any given time instant. On the other hand, the sender buffer (between the TCP layer and the MAC layer) constrains the maximum number of bytes in the retransmit queue. Consequently, the size of the sender buffer also constrains the amount of unacknowledged data that can be outstanding at any time. Following the convention of T-RAT, we call the first limitation *receiver window* limitation and the second one *sender window* limitation. If the transmission rate of a connection is limited by a window size (either sender or receiver window limitation), the sliding window of TCP will be consistently smaller than the bandwidth delay product of the path. Figure 1 shows a time vs. sequence diagram of an receiver window limited connection. The staircase-like lines indicate the left (lower) and right (upper) limit of the sliding window and the vertical arrows represent data segments that were sent. Since most of the time, the lines for the data segments transmitted coincide with line tracking the upper limit of the sliding window, the sender is receiver window limited.

Sender and receiver window limitations result in the same observable behavior. As we will see in section 5, identifying a receiver window limitation is possible using the advertised window information carried by TCP packets. On the other side, identifying a sender window limitation is a much more complex task, that we do not address in this work. In [18], the authors use the notion of *flight size* to infer a sender-window limitation. However, we will see in section 3 that identification of flights is, most of the time, impossible. We expect that for most transfers in the Internet, the sender buffer to be at least the size of the receiver window. Indeed, in most Unix implementations of TCP, the minimum size for the sender buffer is 64 Kbytes, which is equal to the maximum receiver window size when the window scale option (RFC 1303) is not used. When the window scale option is used, a correct implementation of a TCP stack should resize the sender buffer when receiving the window scale factor of the other side. However, a re-

cent study [12] has observed that 97% of the hosts that support the window scale option used a window scale factor of 0, meaning that the maximum receiver window was at most 64 Kbytes.

There is an additional type of limitation at the transport layer that is referred to as opportunity limitation in T-RAT. This limitation occurs for short connections carrying so few bytes that the connection never leaves the slow start phase. Since it is the slow start behavior of TCP that limits the rate of the TCP transfer we do not classify these connections as application limited. As will become clear later, we concentrate on analyzing long TCP connections in which case opportunity limitation will not be an issue.

2.3 Network Limitation

A third category of limitation causes for the throughput seen by a TCP connection are due to the network. We focus on the case where one or more bottlenecks on the path limit the throughput of the connection (see [7] for a study on the location and lifetime of bottlenecks in the Internet). While other network factors, such as link failures or routing loops [16], might impact a TCP connection, we do not consider them in the present work as we can reasonably expect their frequency to be negligible as compared to the occurrence of bottlenecks.

For the following, we need to borrow a few definitions from [5]. We define first general metrics independent of the transport protocol:

capacity C_i of link i:
the maximum possible IP layer transfer rate at that link

end-to-end capacity C in a path:
$C = min_{1,\ldots,H} C_i$, where H is the number of hops in the path

the average available bandwidth A_i of a link i:
$A_i = (1 - u_i)C_i$, where u_i is the average utilization of that link in the given time interval

the average available end-to-end bandwidth A in a path:
$A = min_{1,\ldots,H} A_i$, where H is the number of hops in the path

We also define two TCP specific metrics:

bulk transfer capacity (BTC_i) of link i:
the maximum throughput obtainable by a single TCP connection at that link

bulk transfer capacity (BTC) of a path:
$BTC = min_{1,\ldots,H} BTC_i$, where H is the number of hops in the path

As in [5], we call the link i with the capacity $C_i = C$ the *narrow link* of the path and the link j with the the average available bandwidth $A_j = A$ the *tight link* of the path. Furthermore, we define link k as the *bottleneck link* if it has a bulk transfer capacity $BTC_k = BTC$. Note that while at a given time instant, there is a single bottleneck for a given connection, the location of the bottleneck as well as the bulk transfer capacity at the bottleneck can change over time. Please note that the bottleneck link is not (necessarily) the same as the tight link. Also the available bandwidth differs from the bulk transfer capacity of a path: The classical example is the case of a link of capacity C used at 100% by a single TCP connection. The available bandwidth is zero on the link while the bulk transfer capacity should be $\frac{C}{2}$.

If the bottleneck link explains the throughput limitation observed for a connection, we declare this connection as *network limited*.

Packet losses are natural indicators of a bottleneck and we will use the packet loss rate as a measure of the impact of the network on a connection. However, note that the packet loss rate by itself does not fully characterize the impact of congestion on the throughput of a given connection. Especially, two connections with different RTT values will not see their throughput affected in the same way even if they experience a similar loss rate, as can be seen directly from the TCP throughput formula [11]: $Tput = \frac{MSS}{RTT}\frac{C}{\sqrt{p}}$, where MSS is the maximum segment size of the connection, C is a constant and p is the loss event probability (which is related to the loss rate, while not similar, as it indicates the frequency of loss periods where one or more packets are lost).

For the above reasons, we will use two metrics to infer if a connection is network limited: (i) the retransmission rate of the connection and (ii) the dispersion ratio (see section 5.2) that can be used to detect if the bottleneck link is shared or not. Figure 2 shows a time vs. sequence diagram of a connection whose throughput is limited by a non-shared bottleneck link. The regular spacing between sent data segments and similarly of the acknowledgments received (tracked by the right limit of the sliding window) is easy to observe.

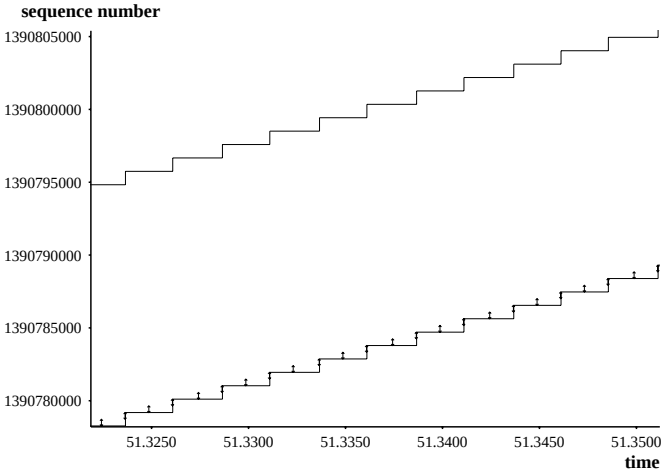
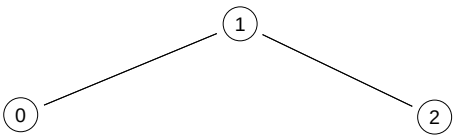


Figure 2: A piece of a bandwidth limited connection where packets are regularly spaced due to the bottleneck link.

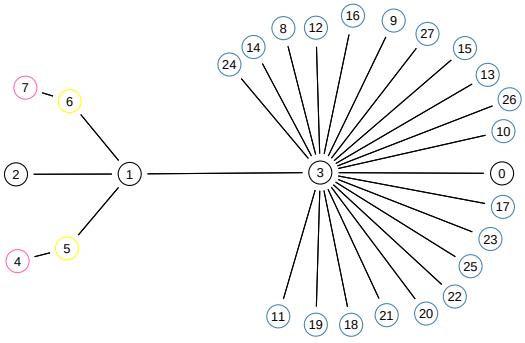
3. ON THE FLIGHT NATURE OF TCP

It is commonly assumed that TCP transfers packets in flights, i.e. in groups of packets that are sent back to back within a group. This is justified by the window based flow and congestion control mechanisms used in TCP. Flights are a very important notion for T-RAT as it needs to relate the flights to the different phases of TCP, namely slow-start, congestion avoidance, and loss recovery. However, in [14], the authors search for flights in Internet traffic traces and arrive to the conclusion that flights can be rarely identified, which means that a tool such as T-RAT is unable to function properly. In the present work, we investigate the notion of flights through simulations, and come to a similar conclusion: while it is possible to observe groups of packets it is difficult to relate them to the well-known phases of TCP.

We simulated TCP connections limited by a specific cause using `ns-2` and varied different parameters (RTT, receiver advertised window, TCP version, etc.) affecting the behavior of the connection. Our objective was to study the similarity of the signatures of connections limited by the same cause but having different parameter values. By signature, we mean the distribution of packet interarrival times (IATs). For example, in the case of a receiver window limited connection one would expect to observe a bimodal distri-



(a) The Basic Configuration.



(b) The Configuration With Cross Traffic

Figure 3: The Configurations for the Simulation.

bution of the IATs, with the principal mode at $\Delta t_1 = \frac{S}{C}$ and the secondary mode at $\Delta t_2 = RTT - \frac{(W-1)*S}{C}$, where S is the packet size (typically can be assumed to be equal to MSS), C the capacity of the narrow link, and W is the receiver advertised window. The principal mode corresponds to the time it takes to transmit a single packet on the narrow link on the path. As all the packets of a single window should be sent back to back in a single flight, their IATs correspond to this value. The position of the second mode corresponds to the time interval between observing the last packet of the previous flight and the first packet of the next flight. Moreover, the ratio of the heights of these peaks should be close to a factor of $W-1$ because for each window worth of packets one observes $W-1$ times an IAT of Δt_1 and one time an IAT of Δt_2. In the following, we show a few examples to demonstrate that this type of simple reasoning rarely holds.

We start with a simple topology with one client (node 0), server (node 2), and one intermediate router (node 1) shown in Figure 3(a). A two-minute long FTP transfer was set up on top of a TCP connection established from node 2 to node 0. Figures 4 and 5 show histograms of IATs of packets where the connection is limited by the receiver advertised window of 20 packets. In Figure 4, delayed acknowledgments were not used by the TCP receiver while in Figure 5 delayed acknowledgments were used. As expected, in Figure 4 we observe the two modes at $\Delta t_1 = 5.1ms$ and $\Delta t_2 = 83.7ms$ and the ratio of their heights is approximately 20. However, if the TCP receiver is delaying acknowledgments the situation becomes more complex. We can still observe the principal mode Δt_1 in Figure 5 but instead of a single secondary mode we observe several additional modes. Due to the delayed acknowledgment timer at the receiver, the set of W packets sent is divided into several smaller sets of packets sent back-to-back. The number of these groups of packets depends on the ratio of the RTT to the delayed acknowledgment timer value but also on W. We can already conclude from this first experiment that relating flights to one of the phases of TCP is a difficult task.

We next consider a more realistic scenario (Figure 3(b)) with cross-traffic using the web client-server class of ns-2 at node 1. Tuning the parameters of the clients, we simulated different load values. Figure 6 shows an example evolution of the probability density function (pdf)[1] of the inter-arrival times of packets when increasing the offered load of the cross-traffic. In these simulations the delayed acknowledgments mechanism is used as this is the most common case. The loss rate for the ftp connection experiencing cross-traffic was zero for all cases of offered load.

The main observation from these plots is that even with small amounts of cross-traffic the structure of the pdf (and consequently of the groups of packets) is much more complex than in the first simple scenario. In general, cross-traffic adds to the queuing delay, which lowers the modes (i.e. creates much more different group sizes). This means that the flight sizes become more complex to identify, which makes it difficult to track the size of the congestion window. These simulations further confirm that it is often impossible to rely on the flight sizes to identify the state of the TCP connection. Therefore, T-RAT [2] [18] will be unable to work properly in many cases.

4. TIME SERIES-BASED APPROACH

Our approach to infer the TCP root causes of a given connection is to generate a number of time series using the packet trace of the connection. We then use these time series to compute scores that characterize the impact of the different causes. As the location of the measurement point on the path impacts the way the time series are generated, we devote the next subsection to this issue. We then briefly present all the time series used in our tests, along with some tests made to validate some heuristics used to derive these time series.

4.1 Measurement Point Location

Most of the time series that we use, are very easy to compute if the packet trace was captured at the sender side. However, we do not want to limit ourselves to this specific case as, for instance, we may use publicly available traces collected by third parties and for which we do not have exact information about the measurement configuration.

The first problem then is to infer the location of the measurement point. Let us consider the case depicted in figure 7, where the measurement point A is close to the sender while points B and C are not. We can determine the measurement point with respect to the connection initiator by measuring and comparing the delay between the SYN and SYN+ACK packets, and the delay between SYN+ACK and ACK packets, referred to as $d1$ and $d2$, respectively, for the measurement point A. We conclude that A is close to the connection initiator if $\frac{d2}{d1} < 0.01$.

More generally, we need to compute the RTT samples for each connection. In the case that our measurement point is close to the sender, we compute the RTT for each acknowledged data packet as the time interval between the timestamps of the last transmission od a data packet and the first acknowledgment for this data packet[3]. In

[1]We compute the pdf estimates using a kernel density estimation technique [15] with Gaussian kernel function.

[2]No publicly available version of T-RAT has been released so far.

[3]Here we must take into account that packets may be sent multiple times, in the case of losses, and similarly acknowledged multiples times, in the case of lost or piggybacked acknowledgments.

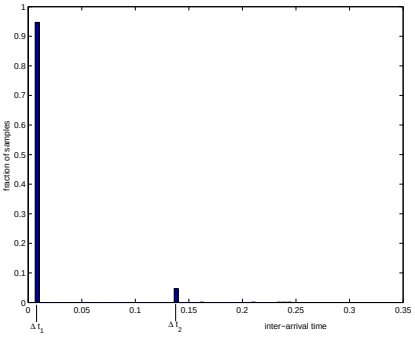


Figure 4: Without Delayed ACKs

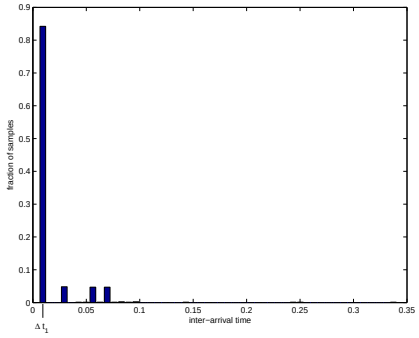


Figure 5: With Delayed ACKs

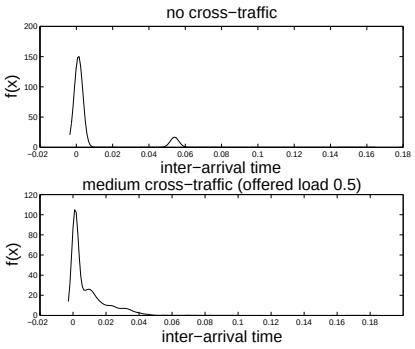


Figure 6: Evolution of the pdf of the inter-arrival times of packets from a receiver window limited connection without and with cross traffic.

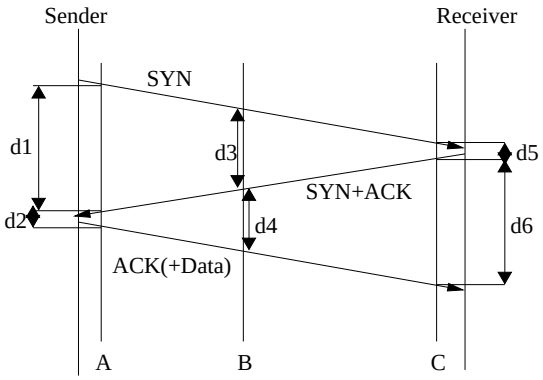


Figure 7: Determining the measurement position from the three-way handshake of TCP

the case that our measurement point is not close to the sender and TCP timestamps are available, we implement the method described in [17].

4.2 Time Series

In this section, we list all the time series that we use in our tests to find the root causes for the throughput seen by a TCP connection.

Fraction of Pushed Packets: A pushed TCP packet is sent with a PUSH flag. RFC-793 says: "The sending user indicates in each SEND call whether the data in that call (and any preceding calls) should be immediately pushed through to the receiving user by the setting of the PUSH flag.". Pushed packet thus indicates that the application on top of TCP has for the moment no more data to send. We compute for each direction of a TCP connection the time series of the fraction of pushed packets observed over all consecutive non-overlapping time intervals of fixed duration. To compute those fractions, we only consider packets carrying data and discard pure acknowledgments. If no packets have been seen during a given time window the value is set to -1.

Inter-arrival Times of Acknowledgments: We compute the inter-arrival times of acknowledgments separately for each direction of a connection. The ACKs included in the computation are either acknowledging one or two data packets of size MSS or duplicate acknowledgments. Furthermore, we cancel the effect of delayed ACKs by dividing by two the inter-arrival time of ACKs that acknowledge two data packets.

Retransmission Rate: We compute for each direction of a TCP connection the time series of the retransmission rate as the fraction of retransmitted bytes per all (data) bytes transmitted in consecutive time intervals of 1 second. A packet is considered to be a retransmission if (i) the packet carries an end-sequence number lower than or equal to any previously observed one; and (ii) the packet has an IPID value [2] [3] higher than any previously observed values.

Note that we cannot rely on observing retransmitted packets twice and counting them since the packets may be lost before the measurement point especially if the measurement point is far from the sender. With the help of the IPID we remove false positive retransmissions caused by reordering of packets by the network that can occur if the measurement point is far from the sender.

Receiver Advertised Window: We compute the time series for receiver advertised window, which consists of time-weighted averaged values over a given time interval: Each time a packet is received from the other end, the receiver window indication in the packets will be considered as the actual receiver window value until either the end of the time window occurs or the reception of a new packet. This technique is valid if the measurement point is located at the sender side. However, if the measurement point is away from the sender, we virtually shift in time the observed timestamp values by the time delay between the sender and the observation point. For example, in Figure 7, when the measurement point is at C, we would shift in time the timestamp values of packets sent by the receiver by $+\frac{d6}{2}$, which is the estimated time at which this packet should arrive at the sender. Note that $d6$ will be estimated using the technique borrowed from [17] as indicated in section 4.1

Outstanding Bytes: Another value of interest is the amount of data bytes sent and not yet acknowledged at a given time instant. Since the computation is done by inspecting both directions of the traffic, we need to take into account the location of the measurement point.

If the measurement point is close to the sender, we produce the time series by calculating the difference between the highest data packet sequence number and the highest acknowledgment sequence number seen for each packet and then averaging these values over a time window in the same way that we do for the receiver advertised window values.

If the measurement point is away from the sender, we do the computation by shifting in time the timestamp values of arriving packets. For example, in Figure 7, we would shift the timestamp values of data packets arriving from the sender at C by $-\frac{d6}{2}$ and of acknowledgments arriving from the receiver at C by $+\frac{d6}{2}$.

The above algorithms (at the sender and at the receiver side) are heuristics that we tested with real transfers on the Internet. We analyzed `scp` transfers from a Web100 enabled machine to another machine and ran `tcpdump` at the sending or receiving machine. Web100 is a kernel patch that allows to access the actual internal variables of the active TCP connections of a host. It provides the exact values for the sending TCP's retransmission queue size, which corresponds to our definition of outstanding bytes. We did `scp` transfers and compared the values obtained from Web100 and our algorithms: from Institut Eurecom to University of Oslo in Norway and from Eurecom to University of Navarra in Spain. As the transfer to Spain proved to be over a lossy path (with approx. 6% of retransmitted bytes) and the one to Oslo not, we were able to capture two different environments.

Figures 8(a) and 8(b) show the comparison in the case where the measurement point is at the sender and receiver side, respectively. Unfortunately, we were unable to dump traffic in the machine located in Oslo and present only the results from the transfer to Spain in the receiver side case. In Figure, 8(a) the two curves for the transfer to Spain are plotted on top of each other, indicating a nearly perfect agreement. In the case of the transfer to Oslo there is a difference of approximately one MSS on average (note that the window scale option was negociated between the two parties in this experiment). The reason for this discrepancy is not clear and may be due to timestamp inaccuracies caused by the high throughput of this transfer. Figure 8(b) for the case where the measurement point is at the receiver side also shows a good agreement between the estimated and actual values.

From the above experiments, we conclude that, in most case, we can expect to observe a maximum a discrepancy between the actual and estimated values that remains below one MSS, a good enough precision for the tests based on these time series (see section 5.2.1).

5. IDENTIFYING AND ANALYZING BULK TRANSFER PERIODS

In this section, we show how we use the time series introduced in section 4 to separate bulk transfer periods from application limited periods. We also introduce the different tests for TCP end host and network limitations.

5.1 Identifying Bulk Transfer Periods

The first operation we perform on a connection is to separate periods limited by the application from other periods. We identify the active phases of a connection where TCP consistently transfers and call them bulk transfer periods.

We identify bulk transfer periods using the time series of fractions of pushed packets using time window of 0.5 seconds. A smaller value for the duration of a time window would risk to interpret the idle time due to the sender waiting for new acknowledgments after having sent a congestion window full of packets, as an indication of application limitation. The algorithm used to separate bulk transfer periods from application limited periods varies between two states: active and inactive. We define the starting state to be inactive. The algorithm switches to the active state (start of a new bulk transfer period) if the fraction of pushed packets is consistently below a value p for Δt_1 consecutive time periods. The algorithm switches back to the inactive state (end of the current bulk transfer period, and start of a new application limited period) if the fraction of pushed packets observed has been consistently above p or if no traffic was sent for Δt_2 consecutive time periods.

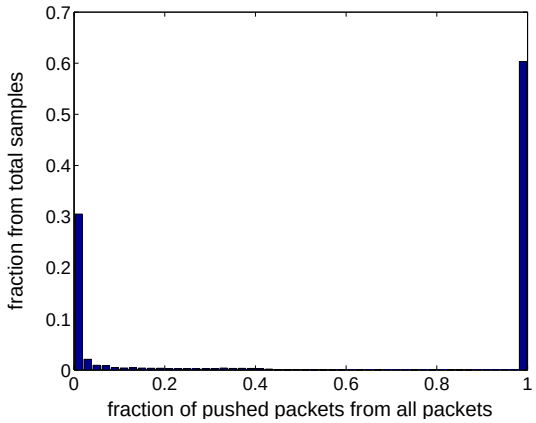


Figure 9: Histogram of the time series values of the fractions of pushed packets for all connections of a 10GB BitTorrent packet trace.

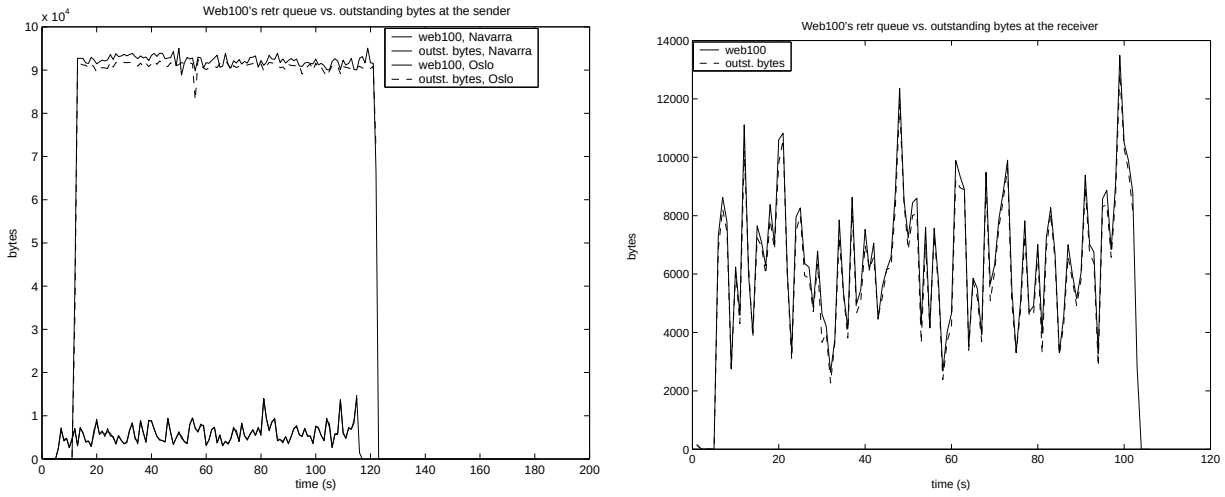
The algorithm is able to recognize both types of application limited periods discussed in Section 2 since we consider both the idle time and the ratio of pushed packets. However, the thresholds p, Δt_1, and Δt_2 need to be tuned according to the type of input traffic and the focus of the analysis. In section 6, we present results for a 10Gbytes BitTorrent trace and we chose p as follows: We extracted the time series of the fractions of pushed packets during one-second time intervals for all those connections of the trace that had more than 10 data packets (we do not process these small connections in any case, see Section 6). We then computed a histogram of all these values, (Figure 9). Based on this histogram we set p to 0.7 but clearly choosing any value from $[0.5, 0.95]$ would practically give the same result. We set $\Delta t_1 = 5$ seconds and $\Delta t_2 = 10$ seconds, which implies that we discard any transfer periods shorter than five seconds.

5.2 Bulk Transfer Period Analysis

After separating bulk transfer periods from application limited periods, we apply the tests for TCP end-point and and the networking limitations on the identified bulk transfer period. These limitation tests are not exclusive. Each of the tests yields a score between 0 and 1 that quantifies the level of the limitation, which is a major improvement over T-RAT [18] that was only providing qualitative results, i.e. a binary answer for each test.

5.2.1 Receiver Window Limitation

We use two time series to test for receiver window limitation: the outstanding bytes time series and the receiver advertised window time series. The difference of the values of these two time series indicates how close the TCP sender's congestion window is to the limit set by the receiver window. Specifically, for each pair of values in the two time series, we compute their difference and generate a binary variable with value one if this difference is less than $n * MSS$ and zero otherwise (in section 6 we discuss the impact of the n value). The receiver limitation score is the average value of the resulting binary time series for the analyzed bulk transfer period.



(a) Measurement point is at the sender.

(b) Measurement point is at the receiver.

Figure 8: Validation of the outstanding bytes algorithms.

5.2.2 Network Limitation

We use two metrics to infer whether the network limits the throughput of a connection: (i) the retransmission score and (ii) the dispersion score.

Retransmission score: The retransmission score for a bulk transfer period is computed as the ratio of the amount of data retransmitted divided by the total amount of data transmitted during this period. Note that since TCP may perform unnecessary retransmissions, retransmission score does not exactly correspond to the loss rate. However, we can expect these quantities to be close to one another in general and especially if the version of TCP uses SACK.

Dispersion score: The objective of the dispersion score is to assess the impact of the bottleneck on the throughput of a connection. We introduce this factor starting from the simple case of a non-shared bottleneck in the network, next moving to the case where there is a shared bottleneck which is the narrow link of the path up to the general case where the bottleneck link is not the narrow link of the path. The dispersion score is computed from the times series of the inter-arrival times of acknowledgments and the average throughput $tput$ of the bulk transfer period under consideration.

Let us first consider the case where the network limitation consists of a non shared bottleneck on the path. The bottleneck is evidently the narrow link of the path. Let r be the capacity of the narrow link. The histogram of the inter-arrival times of acknowledgments (computed as explained in Section 4.2) should exhibit a mode located at $\frac{MSS}{r}$ that contains most of the mass. Since the bulk transfer period is network limited and the narrow link is not shared, the ratio of $tput$ to r should be approximately equal to 1, i.e. $\frac{tput}{r} \sim 1$. We define the *dispersion score* as $1 - \frac{tput}{r}$.

Consider now the more complex case where the bottleneck link is still the narrow link of the bulk transfer period but it is now shared [4]. The histogram of the inter-arrival times of acknowledgment should still exhibit a mode located at $\frac{MSS}{r}$. However, since the bottleneck is now shared, this mode will contain a smaller fraction of the total mass of the histogram. Also, the ratio $\frac{tput}{r}$ represents the share the connection obtains at the narrow link during this bulk transfer period.

Consider now the more general case where the bottleneck is not the narrow link. The mode at $\frac{MSS}{r}$ in the histogram of the inter-arrival times of acknowledgments should still persist, though less pronounced, and it is thus still possible to identify the capacity of the narrow link. (We refer the reader to [9] for a related work that takes advantage of the distribution of the inter-arrivals of packets to identify link capacities.) In this case, $\frac{tput}{r}$ does not represent any more the share that the connection obtains at the bottleneck. However, the dispersion score can still be seen as an indicator of the distortion (or dispersion) introduced by the network.

We validate the method to infer the narrow link capacity by setting up an artificial narrow link with NIST Net at Institut Eurecom, the receiving end of the path, and transferring a large file with scp to Eurecom from University in Oslo (UiO), University of Navarra (UN), and Helsinki University of Technology (HUT). We performed two experiments with three parallel transfers for two different narrow link capacities (2 Mbit/s, 5 Mbit/s) in order to observe the effect of cross traffic, and three experiments with a single transfer each to observe the impact for dispersion score. The results in table 1 show that the accuracy of the estimated r value in these tests is good regardless of the retransmission score on the path. We also observe that the low dispersion scores correctly reveal that there is no sharing at the bottleneck.

6. EXPERIMENTAL RESULTS

6.1 Dataset

We applied our algorithms to a 10 Gbytes `tcpdump` packet trace

[4] In practice, detecting this case is not an easy task and would require the use of tools like `Pathneck` [7] that detects bottlenecks in combination with a tool that measures the capacity of a path [5]. This study is out of the scope of the present work

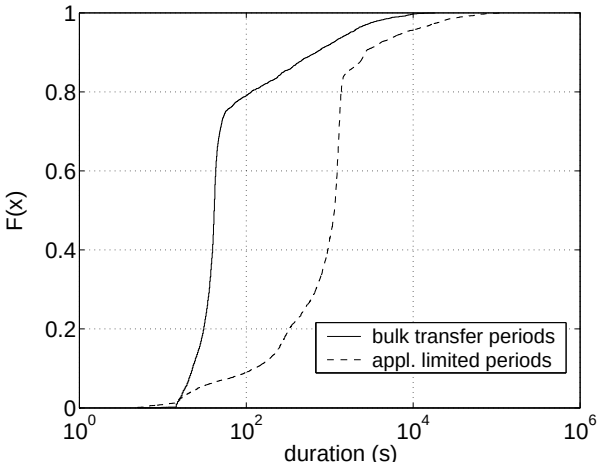


Figure 10: Cdfs of durations of the periods

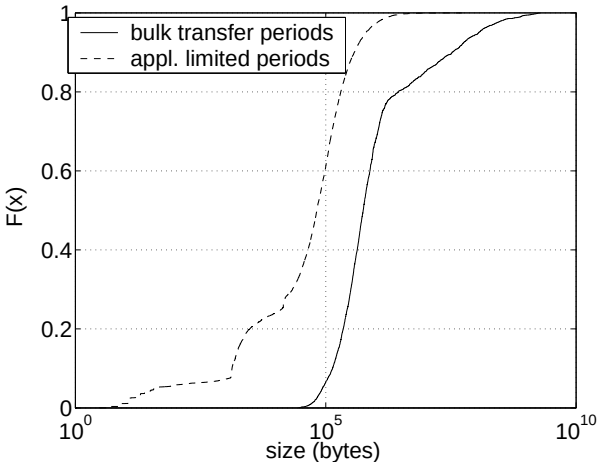


Figure 11: Cdfs of the size of the periods in bytes

Table 1: Validation results from inferring the capacity of the narrow link.

src	set r	r estimate	$1 - \frac{tput}{r}$	retr score	RTT
UN	2.0 Mbit/s	2.0 Mbit/s	0.89	0.175	56 ms
UiO	2.0 Mbit/s	2.0 Mbit/s	0.43	0.027	71 ms
HUT	2.0 Mbit/s	2.0 Mbit/s	0.69	0.037	71 ms
UN	5.0 Mbit/s	5.1 Mbit/s	0.92	0.089	56 ms
UiO	5.0 Mbit/s	5.1 Mbit/s	0.35	0.008	72 ms
HUT	5.0 Mbit/s	5.1 Mbit/s	0.78	0.013	71 ms
UN	2.0 Mbit/s	2.0 Mbit/s	0.21	0.055	57 ms
UiO	2.0 Mbit/s	2.0 Mbit/s	0.10	0.014	70 ms
HUT	2.0 Mbit/s	2.0 Mbit/s	0.03	0.005	71 ms

of BitTorrent traffic captured at the University of Navarra, Spain. The machine at Navarra was involved in a single torrent and the traffic was recorded once the machine had obtained a full copy of the file and thus only acting as a server (seed in the BitTorrent terminology). Hence, all the traffic was captured at the sender side. The trace contains nearly $60,000$ connections with a total amount of 102 million packets.

6.2 Separating the Wheat From the Chaff

Out of the $60,000$ initial connections, we first filtered out the $57,118$ connections with less than 10 data packets. We then applied our algorithm to isolate the bulk transfer periods and the applications limited periods to the remaining 2882 connections. We discarded the connections that consisted of a single application limited period (our algorithm discards transfer periods of less than 5 seconds, refer to Section 5.1). We ended up with *only* 686 connections (to 413 hosts) consisting of 3295 bulk transfers and $10,365$ application limited periods.

Figures 10 and 11 show the cumulative probability density functions (cdf) of the durations and sizes in bytes of both types of periods. We observe that even though BitTorrent is sending only small protocol messages (e.g. to request a block or a piece, or to keep connection alive) during the periods when it is non active or only downloading data, i.e. the application limited periods, the duration of those periods is so large as compared to the bulk transfer periods (figure 10) that eventually the total amount of bytes of some of the application limited periods can be non negligible as compared to the amount carried by some of the bulk transfer periods (figure 11). Indeed, a closer look revealed that the ones transferring up to 2.5 Mbytes are several hour-long connections sending small packets with push flags at a very low rate ($< 5Kbit/s$).

For the bulk transfer periods the coefficient of correlation between the throughput and the size is 0.65 and between the throughput and the duration is 0.52. Such strong correlations are the consequence of the BitTorrent protocol that favors fast transfers between peers. Hence, the faster the transfer, the more likely it is to last, and thus to be large.

6.3 Receiver Window Limitation

Figure 12 shows a cdf of the receiver window limitation score for different values of n (see section 5.2.1). Clearly, the choice of n is not critical as the shape of the curve remains practically the same for $n \in \{1, 2, 3\}$. We use in the following analysis $n = 2$. We observe that approximately 65% of the transfers are never limited by the receiver window and 17% are limited half of their life time. Only a small fraction of the transfers are limited more than 90% of the time by the receiver window. In Figure 13, the receiver window limited score is plotted against the mean value of the receiver advertised window size. The three most common advertised window values are distinguishable from Figure 13 as horizontal stripes: 8, 16, and 64 Kbytes, an observation that agrees with [12]. The coefficient of correlation between the limitation score and average advertised window size is -0.37 which indicates that it is more probable to be receiver window limited when the average advertised window value is smaller. However, we note that there is a significant amount of transfers with a high limitation score and an average advertised window of 64 Kbytes, the largest usable value without window scaling. This observation suggests that perhaps, in some cases, a higher throughput could be obtained by using the window scaling option, though it is not certain and depends on the amount of available bandwidth on the path. Indeed, using a larger window value might equally lead to congestion and lower throughput [13].

We found that all of the 413 client hosts used window scaling or supported its usage in our dataset. Nevertheless, approximately 93% of the hosts did not scale their own advertised window, while 6% used a value of 2 and 1% a value of 7. This agrees with the results in [12] where 97% of the hosts that support window scaling set the value to zero. On the other hand, only 26.6% of the hosts in

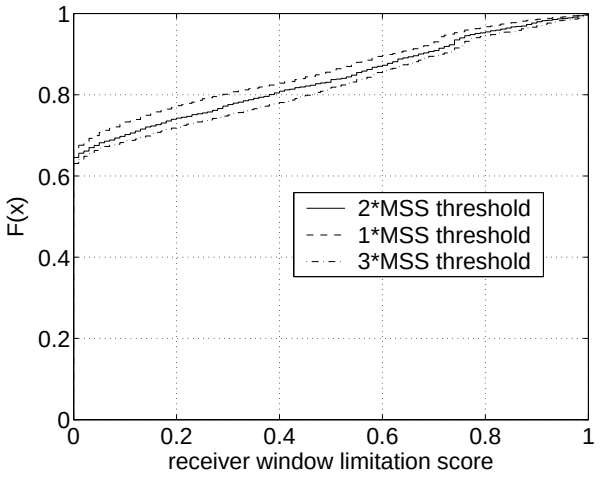


Figure 12: Cdf of receiver window limitation score

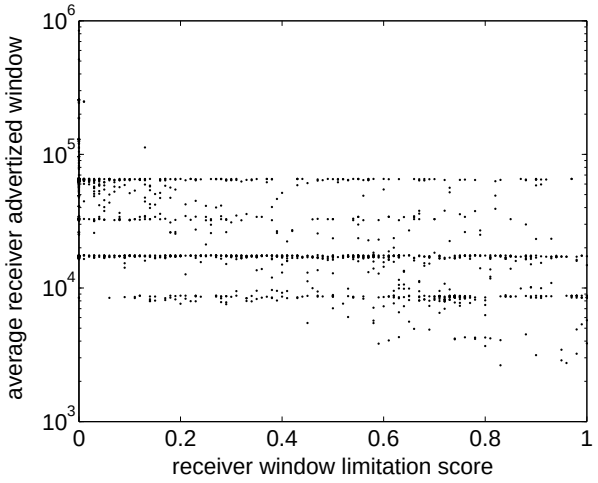


Figure 13: Receiver window limitation score vs. mean advertised window size

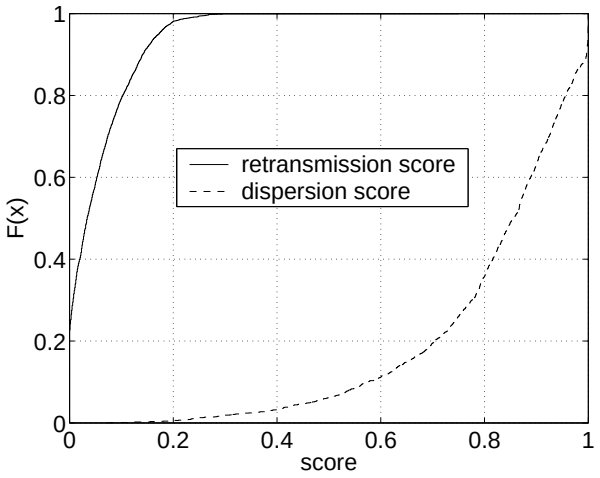


Figure 14: Cdfs of network limitation scores

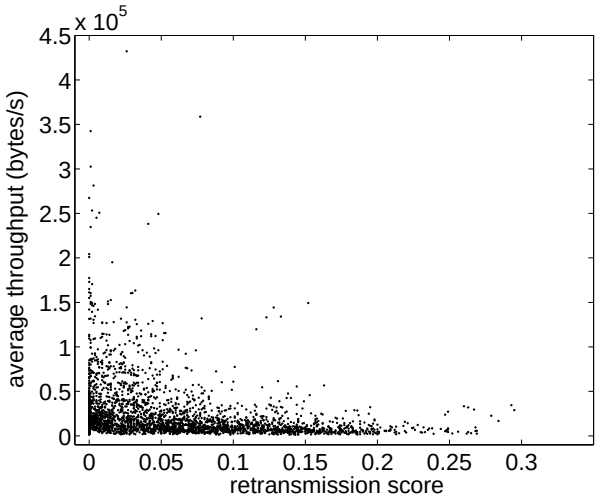


Figure 15: Retransmission score vs. throughput

their dataset support window scaling, as compared to 100% in our case.

6.4 Network Limitation

We observed surprisingly elevated levels of network limitations in our dataset as can be seen from Figure 14. In 20% of the transfer periods, at least 10% of the bytes were retransmitted. When we plot the retransmission score against achieved throughput in Figure 15, we observe that the higher the retransmission score the lower the throughput as stated by the TCP throughput formula [11]. The coefficient of correlation between the retransmission score and the throughput is -0.21.

The cdf of the estimated capacities of the narrow link is presented in Figure 16. The most dominant capacity is around 2 Mbit/s with 16% of the values (highlighted with a box). The values around 1 Gbit/s are erroneously inferred since the capacity of the access link of our measurement host was less than 1Gbit/s and may be due to ack compression. Out of the 1791 narrow links for which we were able to infer a capacity we identified only 10 potential non-shared bottlenecks, i.e. cases where the dispersion score was smaller than 0.2. As the retransmission score was high throughout the dataset and the inferred narrow link capacities fairly modest (in more than 70% of the cases below 2.5 Mbit/s), a possible explanation for high dispersion scores (see Figure 14) could be a very congested high capacity link close to our measurement host. Figure 17, which plots the retransmission score against the dispersion score for bulk transfer periods with a receiver window limitation score lower than 0.5, reveals that there is a connection between these two scores. The coefficient of correlation between them was 0.25. More precise interpretations of this phenomenon would require more information about the cross-traffic, available bandwidth, and bottleneck locations on the path and is left as future work.

We looked more closely at some of the bulk transfer periods with zero retransmission score and a high dispersion score. We found 30 to 60 seconds-long bulk transfers where TCP is in congestion avoidance and experiences very long RTTs that prevents it from growing its congestion window large enough to reach the limit set by the receiver window or available bandwidth before the end of the transfer. The connection depicted in Figure 18 had an initial RTT of $65ms$ while during the transfer period the RTT grew up to

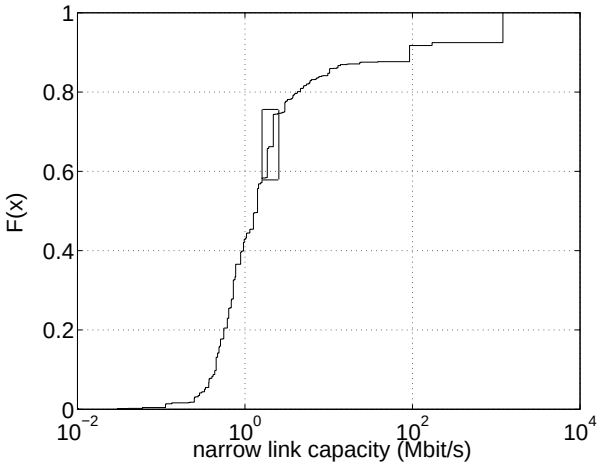


Figure 16: Cdf of inferred narrow link capacities. The box highlights the 2Mbit/s transfers.

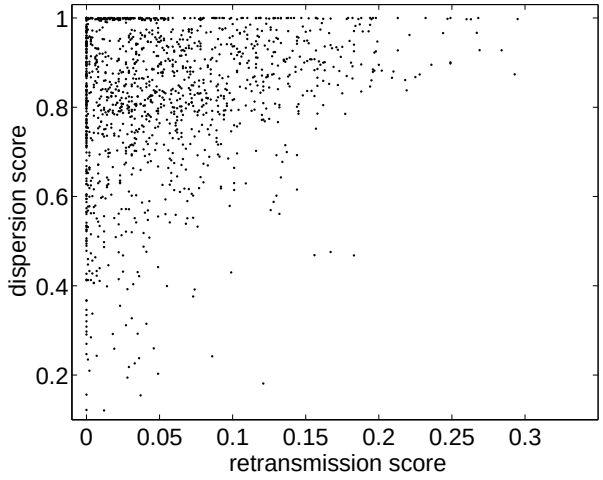


Figure 17: Retransmission score vs. dispersion score

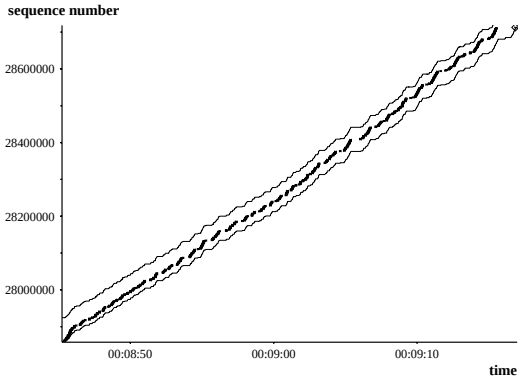


Figure 18: A complete bulk transfer period with a high dispersion score and no retransmissions.

$1.3s$. This suggests that somewhere along the path large queuing delays were introduced causing a network limitation that could not be detected by only observing retransmissions.

6.5 Exclusiveness of the Limitation Causes

We want to shed more light on the dynamics of bulk transfer periods that experience both network and receiver window limitation. Figure 19(a) reveals that even though the trend is, as expected, that a high score in one test excludes a high score in the other test, there are still quite a few transfer periods with significant scores for both limitations. We had a closer look at the three such transfer periods highlighted with a box in Figure 19(a). Two of them exhibited occasional retransmissions alternating with periods where the senders were receiver window limited as visible in Figure 19(b). Retransmissions are marked with vertical arrows with an R on top. Non-retransmitted data segments often hit the upper limit of the sliding window. In contrast, the third transfer period was receiver window limited until just before the end of the transfer where it retransmitted a large number of packets.

7. CONCLUSIONS AND OUTLOOK

In this paper, we have revisited the issue of the root cause analysis of TCP connections introduced in [18]. We have first demonstrated the weakness of the flight-based approach adopted in [18]. We have provided a thorough discussion on the different limitation causes, i.e. the application, the TCP end point parameters and the network, emphasizing the need to account for the impact of the application on the observed traffic. We then came up with a new analysis method based on various time series extracted from the headers of a packet trace. Our technique is robust and allows to precisely assess the impact of each limitation. A score representing the limitation level between 0 and 1 is provided for each of the causes (as opposed to T-RAT [18] that provided only a binary answer, yes or no, to each test). A first application of the tool on a large BitTorrent dataset has demonstrated the interest of the technique.

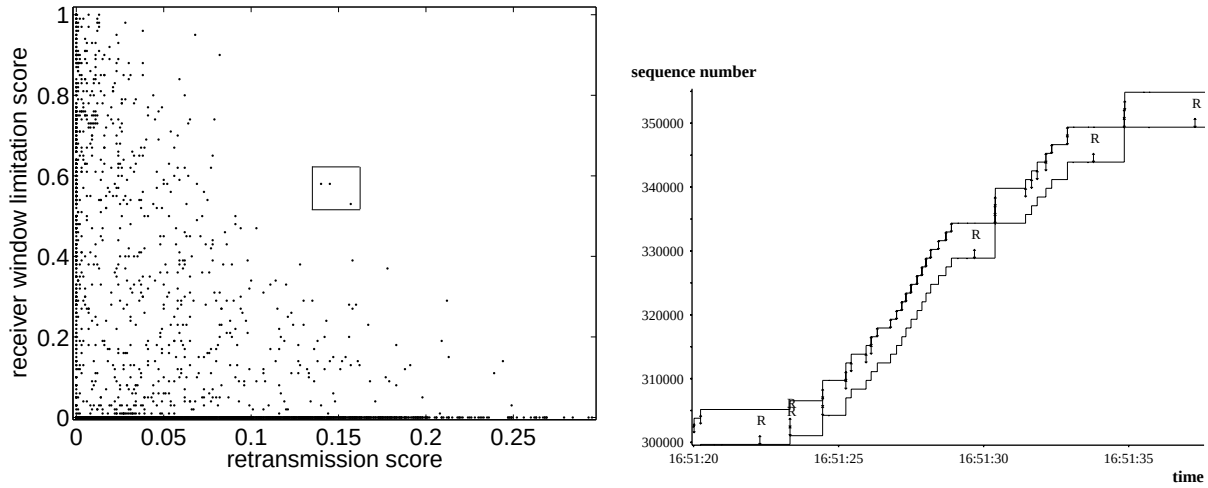
Future work includes applying our tool to publicly available traces that contain data of various applications. As explained in Section 5.1, there are currently several parameters that need to be tuned according to the application type. We are working on a reimplementation of the algorithm for bulk transfer identification that requires neither a time window nor the thresholds and could therefore work regardless of the time scale of the transfer and the type of the application. We want to analyze other applications because we believe that the ”bulk transfer applications” (P2P file transfers, FTP, scp, etc.) do not form a homogeneous class of applications but can generate many different traffic patterns due to a number of factors, e.g. application-level mechanisms, compression, and encryption, which all have an impact on how data is delivered to TCP and subsequently transferred. Another challenge in analyzing publicly available traces is that RTT estimation is still difficult in case the measurement point is not at the sender and TCP timestamps are not available [6]. We are currently investigating whether other methods for RTT estimation are suitable in this case.

We would also like to analyze the evolution of the TCP root causes of bulk transfer periods within a connection, and eventually, study in more depth the temporal dynamics of the causes and their interaction within a bulk transfer period.

Acknowledgments

This work has been partly supported by the European Union under the E-NEXT project FP6-506869 and by France Telecom, project CRE-46126878.

The authors would like to thank Mikel Izal from University of



(a) Retransmission score vs. receiver window limitation score

(b) Close up of a receiver window and network limited transfer period.

Figure 19: Network and receiver window limitation

Navarra, Spain for providing the BitTorrent dataset used in our experiments.

8. REFERENCES

[1] S. Baset and H. Schulzrinne, "An Analysis of the Skype P2P Internet Telephony Protocol", CUCS-039-04, Department of Computer Science, Columbia University, 2004.

[2] J. Bellardo and S. Savage, "Measuring packet reordering", In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 97–105, New York, NY, USA, 2002, ACM Press.

[3] S. M. Bellovin, "A technique for counting natted hosts", In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 267–272, New York, NY, USA, 2002, ACM Press.

[4] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool", *Comput. Commun. Rev.*, 33(3):111–126, 2003.

[5] K. Claffy, R. S. Prasad, M. Murray, and C. Dovrolis, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools", *IEEE Network*, 17(6):27–35, November 2003.

[6] M. Dyrna, "Network Tomography Tools", M.S. Thesis, TU Muenchen/Eurecom, September 2005.

[7] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: algorithms, measurements, and implications", In *Proceedings of ACM SIGCOMM 2004 Conference*, pp. 41–54, New York, NY, USA, 2004, ACM Press.

[8] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", In *Passive and Active Measurements 2004*, April 2004.

[9] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss, "MultiQ: Automated Detection of Multiple Bottleneck Capacities Along a Path", In *Proceedings of Internet Measurement Conference (IMC '04)*, pp. 245–250, October 2004.

[10] M. Mathis, J. Heffner, and R. Reddy, "Web100: extended TCP instrumentation for research, education and diagnosis", *Comput. Commun. Rev.*, 33(3):69–79, 2003.

[11] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", *Comput. Commun. Rev.*, 27(3):67–82, July 1997.

[12] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", *Comput. Commun. Rev.*, 35(2):37–52, April 2005.

[13] R. S. Prasad, M. Jain, and C. Dovrolis, "Socket Buffer Auto-Sizing for High-Performance Data Transfers", *Journal of Grid Computing*, 1(4):361–376, December 2003.

[14] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, and kc claffy, "The RTT Distribution of TCP Flows in the Internet and its Impact on TCPbased Flow Control", , Cooperative Association for Internet Data Analysis (CAIDA), University of Illinois, 2004.

[15] B. Silverman, *Density Estimation for Statistics and Data Analysis*, CRC Press, 1986, ISBN 0412246201.

[16] R. Teixeira and J. Rexford, "A measurement framework for pin-pointing routing changes", In *NetT '04: Proceedings of the ACM SIGCOMM workshop on Network troubleshooting*, pp. 313–318, New York, NY, USA, 2004, ACM Press.

[17] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times", In *Proceedings of Passive and Active Measurements(PAM)*, 2005.

[18] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates", In *Proceedings of ACM SIGCOMM 2002 Conference*, Pittsburgh, PA, USA, August 2002.

On the Stationarity of TCP Bulk Data Transfers

G. Urvoy-Keller

Institut Eurecom, 2229, route des crêtes, 06904 Sophia-Antipolis, France
urvoy@eurecom.fr

Abstract. While the Internet offers a single best-effort service, we remark that (i) core backbones are in general over provisioned, (ii) end users have increasingly faster access and (iii) CDN and p2p solutions can mitigate network variations. As a consequence, the Internet is to some extent already mature enough for the deployment of multimedia applications and applications that require long and fast transfers, e.g. software or OS updates. In this paper, we devise a tool to investigate the stationarity of long TCP transfers over the Internet, based on the Kolomogorov-Smirnov goodness of fit test. We use BitTorrent to obtain a set of long bulk transfers and test our tool. Experimental results show that our tool correctly identify noticeable changes in the throughput of connections. We also focus on receiver window limited connections to try to relate the stationarity observed by our tool to typical connection behaviors.

1 Introduction

The current Internet offers a single best-effort service to all applications. As a consequence, losses and delay variations are managed by end-hosts. Applications in the Internet can be classified into two classes: elastic applications, e.g. web or e-mail, that can tolerate throughputs and delays variations; and real time applications, that are delay sensitive (e.g. voice over IP) or throughput sensitive (e.g. video-on-demand). With respect to the above classification, a common belief is that the current Internet with its single best-effort service requires additional functionality (e.g. DiffServ, MPLS) to enable mass deployment of real-time applications. Still, a number of facts contradict, or at least attenuate, this belief: (i) recent traffic analysis studies have deemed the Internet backbone ready to provide real-time services [15]; (ii) the fraction of residential users with high speed access, e.g. ADSL or cable, increases rapidly; (iii) network-aware coding schemes, e.g. mpeg4-fgs [7], combined with new methods of transmission like peer-to-peer (p2p) techniques, e.g. Splitstream [5], have paved the way toward the deployment of real-time applications over the Internet.
The above statements have lead us to investigate the variability of the service provided by the Internet from an end connection point of view. As TCP is carrying most of the bytes in the Internet [10], our approach is to concentrate on long lived TCP connections. Bulk data transfers represent a significant portion of the current Internet traffic load, especially with p2p applications [2]. By analyzing bulk data transfers, we expect to better understand the actual interaction

between TCP and the Internet. This is important for future applications [1] and also for CDN providers that rely on migrating traffic on the "best path" from central to surrogate servers [8]. CDN providers generally rely on bandwidth estimation tools, either proprietary or public tools [12] to perform path selection. However, the jury is still out on the stationarity horizon provided by such tools, i.e. how long will the estimation provided by the tool remain valid or at least reasonable. In the present work, we propose and evaluate a tool that should help solving these issue. The rest of this paper is organized as follows. In Section 2, we review the related work. In Section 3, we present our dataset. In Section 4, we present our tool to extract stationarity periods in a given connection. In Section 5, we discuss results obtained on our dataset. Conclusions and future work directions are presented in Section 6.

2 Related Work

Mathematically speaking, a stochastic process $X(t)$ is stationary if its statistical properties (marginal distribution, correlation structure) remain constant over time.
Paxson et al. [17] have studied the stationarity of the throughput of short TCP connections (transfers of 1Mbytes) between NIMI hosts. The major difference between this work and the present work is that we consider long bulk data transfer (several tens of minutes) and our dataset is (obviously) more recent with hosts with varying access capacity, whereas NIMI machines consistently had good Internet connectivity. Other studies [4, 13] have concentrated on the non stationarity observed on high speed link with a high number of aggregated flows. They studied the time scales at which non stationarity appears and the causes behind it. Also, recently, the processing of data streams has emerged as an active domain in the database research community. The objective is to use database techniques to process on-line stream at high speed (e.g. Internet traffic on a high speed link). In the data stream context, detection of changes is a crucial task [14, 3].

3 Dataset

Our objective is to devise a tool to assess the stationarity of TCP bulk data transfers. To check the effectiveness of the tool, we need to gather samples, i.e. long TCP transfers, from a wide set of hosts in the Internet. A simple way to attract traffic from a variety of destinations around the world is to use a p2p application. As we are interested in long data transfers, we used BitTorrent, a popular file replication application [11]. A BitTorrent session consists in the

[1] Our focus in the present work is on throughput, which is an important QoS metrics for some multimedia applications, e.g. VoD, but arguably not all multimedia applications, a typical counter-example being VoIP.

replication of a single large file on a set of peers. BitTorrent uses specific algorithms to enforce cooperations among peers. The data transfer phase is based on the swarming technique where the file to be replicated is broken into chunks (typical chunk size is 256 kbytes) that peers exchange with one another. The BitTorrent terminology distinguishes between peers involved in a session that have not yet completed the transfer of the file, which are called *leechers* and peers that have already completed the transfer, which are called *seeds*. Seeds remain in the session to serve leechers. Connections between peers are permanent TCP connections. Due to the BitTorrent algorithms [11], a typical connection between two hosts is a sequence of on periods (data transfers) and off periods (where only keep-alive messages are transfered). Figure 1, where y axis values are one second throughputs samples, depicts a typical one way connection of approximately 14 hours with clear on and off phases .

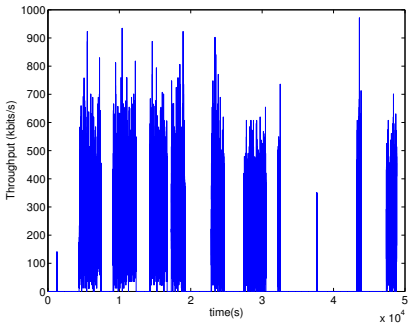


Fig. 1. A typical (one-way) BitTorrent connection

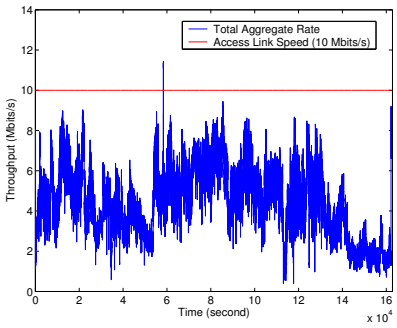


Fig. 2. Aggregate rate of the BitTorrent application during the experiment

The dataset we have collected consists of connections to about 200 peers that were downloading (part of) the file (latest Linux Mandrake release) from a seed located at Eurecom. More precisely, a tcpdump trace of 10 Gbytes was generated during a measurement period of about 44 hours. While the 200 connections are all rooted at Eurecom, the 10 Mbits/s access link of Eurecom should not constitute a shared bottleneck for two reasons. First, with BitTorrent, a client (leecher or seed) does not send to all its peers simultaneously but only to 4 of them, for sake of efficiency. Second, the total aggregate throughput remains in general far below the 10 Mbits/s as shown in figure 2 while the average traffic generally observed on this link (to be added to the traffic generated by our BitTorrent client to obtain the total offered load for the link) exhibits an average rate around 1 Mbits/s with a peak rate below 2 Mbits/s.
To illustrate the diversity of these 200 peers, we have used the maxmind service (http://www.maxmind.com/) to assess the origin country of the peers. In table 1, we ranked countries based on the peers that originate from each of them. Unsurprisingly, we observe a lot of US peers (similar observation was made in [11] for a

similar torrent, i.e. Linux Redhat 9.0) while the other peers are distributed over a wide range of 27 countries (see http://encyclopedia.thefreedictionary.com/ISO%203166-1 for the meaning of the abbreviations used in table 1).

Our objective is to study long bulk data transfers in the Internet. To obtain

Country	# peers	Country	# peers	Country	# peers	Country	# peers
US	87	NL	4	BR	2	YU	1
UK	24	DE	3	LT	2	BE	1
CA	14	AU	3	CN	1	AT	1
FR	12	PE	3	NO	1	ES	1
IT	8	AE	3	SI	1	CH	1
SE	8	CL	2	TW	1		
PL	7	PT	2	CZ	1		

Table 1. Origin countries of the 200 peers

meaningful samples, we extracted the on periods from the 200 connections, resulting in a total of 399 flows. The algorithm used to identify off-periods is to detect periods of at least 15 seconds where less than 15 kbytes of data are sent, as BitTorrent clients exchange keep-alive messages at a low rate (typically less than 1000 bytes per second) during periods where no data transfer is performed. We further restricted ourselves to the 184 flows whose duration is higher than 1600 seconds ($\sim$ 26.6 minutes), for reasons that will be detailed in section 4. We call flow or initial flow an on-period and stationary flow a part of a flow that is deemed stationary. For each flow, we generate a time series that represents the throughput for each 1 second time interval. The average individual throughput of these 184 flows is quite high, 444 kbits/s. Overall, these flows correspond to the transfer of about 50 Gbytes of data over a cumulated period of about 224 hours (the flows of duration less than 1600 seconds represent about 14 Gbytes of data). Due to its size, we cannot claim that our dataset is representative of the bulk transfers in the Internet. It is however sufficiently large to demonstrate the effectiveness of our tool. It also shows that BitTorrent is a very effective application to collect long TCP transfers from a variety of hosts in terms of geographical location and access link speed (even if it is unlikely to observe clients behind modem lines, as downloading large file behind a modem line is unrealistic).

4 Stationarity Analysis Tool

4.1 Kolmogorov-Smirnov (K-S) test

Given two i.i.d samples $X_1(t)_{t\in\{1,...n\}}$ and $X_2(t)_{t\in\{1,...n\}}$, the Kolmogorov-Smirnov test enables us to determine whether the two samples are drawn from the same distributions or not. The test is based on calculating the empirical cumulative distribution functions of both samples and evaluating the absolute maximum

difference $D_{\max}$ between these two functions. The limit distribution of $D_{\max}$ under the null hypothesis (X_1 and X_2 drawn from the same distribution) is known and thus $D_{\max}$ is the statistics the test is built upon. In the sequel of this paper, we used the matlab implementation of the K-S test with 95% confidence levels.

4.2 K-S test for change point detection

Our objective is to detect stationary regions in time series, or equivalently to detect change points (i.e. border points between stationary regions). We used the K-S test to achieve this goal. Previous work as already used the K-S test to detect changes [9, 3], though not in the context of traffic analysis.
The basic idea behind our tool is to use two back-to-back windows of size w sliding along the time series samples and applying the K-S test at each shift of the windows. If we assume a time series of size n, then application of the K-S test leads to a new binary time series of size $n - 2w$, with value zero whenever the null hypothesis could not be rejected and one otherwise. The next step is to devise a criterion to decide if a '1' in the binary time series corresponds to a false alarm or not. Indeed, it is possible to show that even if all samples originate from the same underlying distribution, the K-S test (or any other goodness of fit test [16]) can lead to spurious '1' values. The criterion we use to deem detection of a change point is that at least $w_{\min} \approx \frac{w}{2}$ consecutive ones must be observed in the binary time series. $w_{\min}$ controls the sensitivity of the algorithm. The intuition behind setting $w_{\min}$ to a value close to $\frac{w}{2}$ is that we expect the K-S test to almost consistently output '1' from the moment when the right-size window contains about 25% of points from the "new" distribution (the distribution after the change point) up to the moment when the left-size window contains about 25% of points from the "old" distribution. In practice, a visual inspection of some samples revealed that using such values for $w_{\min}$ allows to correctly detect obvious changes in the time series. Figure 3 presents an example on one of our TCP flows time series (aggregated at a 10 seconds time scale - see next section for details) along with the scaled binary time series output by the tool and the change points (vertical bars). This example illustrates the ability of the test to isolate stationary regions. Note also that the output of the binary time series that represents the output of the K-S test for each window position (dash line in figure 3) exhibits a noticeable consistency. This is encouraging as oscillations in the output of the test would mean that great care should be taken in the design of the change point criterion. As this is apparently not the case, we can expect our simple criterion ($w_{\min}$ consecutive '1' values to detect a change) to be effective.

4.3 K-S test in the presence of correlation

We want to apply the K-S change point tool described in the previous section to detect changes in the throughput time series described in section 3. However, we have to pay attention that, due to the close loop nature of TCP, consecutive

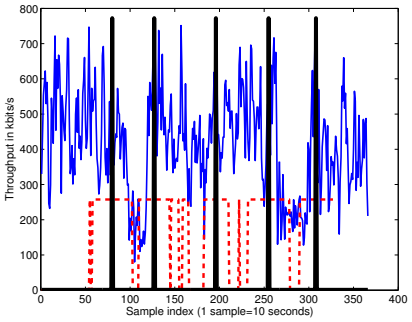


Fig. 3. Initial time series (thin line), binary time series (dash line) and change points (thick bars)

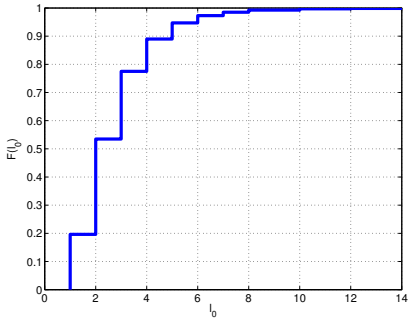


Fig. 4. Cumulative distri. function of l_0

one-second throughputs samples are correlated[2]. If all samples are drawn from the same underlying distribution, a simple heuristic to build an uncorrelated time series out of a correlated time series is to (i) compute the auto-correlation function of the initial time series, (ii) choose a lag l_0 at which correlation is close enough to zero and (iii) aggregate the initial time series over time intervals of size l_0. Specifically, let $X(t)_{t\in\{1,\ldots n\}}$ be the initial time series. Its auto-correlation function is $AC(f) = \frac{\sum_{i=1}^{n-f} \bar{X}(i+f)\bar{X}(i)}{n\sigma^2_{\bar{X}}}$, where $\bar{X}(t) \triangleq X(t) - E[X]$ and $\sigma^2_{\bar{X}}$ is the variance of $\bar{X}$. $AC(f)$ measures the amount of correlation between samples located at positions t and $t+f$. If ever the time series is i.i.d., then $|AC(f)|$ should be upper bounded by $\frac{2}{\sqrt{n}}$ for $f > 1$ [6]. For a correlated time series, we can choose l_0 such that $\forall f > l_0, |AC(f)| \leq \frac{2}{\sqrt{n}}$. We then generate the aggregate time series $Y(t)_{t\in\{1,\ldots\lceil\frac{n}{l_0}\rceil\}}$ where $Y(t) = \frac{\sum_{u=t\times l_0+1}^{(t+1)\times l_0} X(u)}{l_0}$. This method is however not applicable to our TCP time series as changes in the network conditions prevent us from assuming the same underlying distributions over the whole duration of a flow.

To overcome this difficulty and be able to use the K-S test, we aggregate each time series at a fixed value of $l_0 = 10$. This means that we average the initial time series over intervals of 10 seconds. As the average throughput of the flows is 444 kbits/s, an average flow will send more than 400 packets (of size 1500 bytes) in a 10 second time interval, which is reasonably large enough for a TCP connection to have lost memory of its past history (e.g. to have fully recovered from a loss). To assess the level of correlation that persists in the time series after aggregation at the 10 second time scale, we have computed, for each stationary interval obtained with our tool, the autocorrelation function of the process in this interval. We then derive the lag l_0 after which the autocorrelation function

[2] While correlation and independence are not the same, we expect that removing correlation will be sufficient in our context to obtain some almost independent samples.

remains (for 95% of the cases) in the interval $\left[-\frac{2}{\sqrt{n}}, \frac{2}{\sqrt{n}}\right]$. Figure 4 represents the cumulative distribution function of l_0. We notice that about 95% of the l_0 values are below 5, which indicates that the "remaining" correlation is of short term kind only.

Based on the result of figure 4, one could however still argue that we should continue further the aggregation of the time series for which the correlation is apparently too large, say for $l_0 \geq 3$. Note however that the choice of the time scale at which one works directly impacts the separation ability of the K-S test. Indeed, as we use windows of w samples, a window corresponds to a time interval of $10 \times w$ seconds, and we won't be able to observe stationary periods of less than $10 \times w$ seconds. For example, the results presented in section 5 are obtained with $w = 40$, which means that we won't be able to observe stationary periods of less than 400 seconds (~ 6.7 minutes). Thus, there exists a trade-off between the correlation of the TCP throughput time series that calls for aggregating over large time intervals and the separation ability of the test that calls for having as much small windows as possible.

A second reason why we have chosen to aggregate at a fixed 10 second time scale value is that we expect our tool to be robust in the presence of short term correlation. We investigate this claim in the next section, on synthetic data, where we can tune the amount of correlation. While by no means exhaustive, this method allows us to obtain insights on the behavior of K-S test in the presence of correlation.

4.4 Test of the robustness of the tool with synthetic data

We consider a first-order auto-regressive process X with $X(t) = aX(t-1) + Z(t), \forall t\{1, \ldots n\}$ where Z is a purely random process with a fixed distribution. We choose two distributions for Z (leading to Z_1 and Z_2) to generate two samples $X_1(t)$ and $X_2(t)$. We then form the compound vector $[X_1(t)X_2(t)]$ and apply the K-S change point test. We can vary the a parameter to tune the amount of correlation and test how the K-S change point test behaves. Specifically, we consider $a \in \{0.2, 0.5, 0.9\}$ as these values roughly correspond to l_0 values (as defined in the previous section) equal respectively to 2, 5 and 20. With respect to the results presented in figure 4, we expect the K-S test to behave properly for $a \leq 0.5$ (i.e. $l_0 \leq 5$). In table 2, we present results obtained when $Z_1(t)$ and $Z_2(t)$ are derived from normal distributions with respective means and variances $(0.4, 0.3)$ and $(1.5, 1.5)$ where a given sample $Z_1(t)$ (resp. $Z_2(t)$) is obtained by averaging 10 independent samples drawn from the normal distribution with parameters $(0.4, 0.3)$ (resp. $(1.5, 1.5)$). The main idea behind this averaging phase is to smooth $X_1(t)$ and $X_2(t)$ in a similar fashion that the throughput samples are smoothed at a 10 second time scale in the case of our BitTorrent dataset. As the transition between $X_1(t)$ and $X_2(t)$ is sharp thanks to the difference in mean between Z_1 and Z_2, we expect that the change point tool will correctly detect it. Now, depending on the correlation structure, it might happen that more change points are detected. This is reflected by the results presented in table 2, where for

different values of a, w and $w_{\min}$, we compute over 1000 independent trajectories, the average number of detections made by the algorithm (without false alarm, we should obtain 1) and the percentage of cases for which a change is detected in the interval $[450, 550]$ that corresponds to the border between $X_1(t)$ and $X_2(t)$ in the compound vector $[X_1(t)X_2(t)]$, each vector having a size of 500 samples. When the latter metric falls below 100%, it indicates that the correlation is such that our tool does not ncessarily notice the border between $X_1(t)$ and $X_2(t)$ any more. From table 2, we note that such a situation occurs only for $a = 0.9$. Also, when the amount of correlation increases, the average number of points detected increases dramatically, as the correlation structure of the process triggers false alarms as illustrated by the trajectory depicted in figure 5. For a given w value, increasing the treshold $w_{\min}$ helps reducing the rate of false. Note that while the results obtained here on synthetic data seem to be better for a criterion $w_{\min} = 40$, we used $w_{\min} = 15$ on our dataset as it was giving visually better results. A possible reason is the small variance of the throughput time series as compared to the corresponding mean for our dataset. More generally, we note that tuning w and $w_{\min}$ is necessary to tailor the tool to the specific needs of a user or an application.

a	w	$w_{\min}$	% of cases with one detection in $[450, 550]$	Average number of detections
0.2	40	15	100	2.4
0.2	40	40	100	1
0.2	80	15	100	2.4
0.2	80	40	100	1.2
0.5	40	15	100	5.6
0.5	40	40	100	1.1
0.5	80	15	100	4.5
0.5	80	40	100	1.9
0.9	40	15	100	14.6
0.9	40	40	99	6.5
0.9	80	15	89.9	8
0.9	80	40	90.7	5.6

Table 2. Change point detection tool performance in the presence of correlation

4.5 Empirical validation on real data

For the results obtained in this section and the rest of the paper, we used $w = 40$ as special care must be taken when using the K-S test for smaller values [16]. Also, we consider $w_{\min} = 15$ as it visually gives satisfying results on our dataset. In addition, to obtain meaningful results, we restrict the application of the tool to time series with at $4 \times w$ samples (the tool will thus output at least $2 \times w$

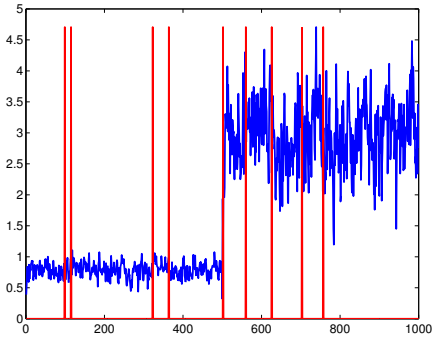


Fig. 5. Sample trajectory of $[X_1 X_2]$ with the detected change points (vertical bars) for $w = 40$ and $w_{\min} = 15$

results), i.e. to flows that last at least 1600 seconds.
Our change point analysis tool can be easily validated with synthetic data. However, we need to further check whether the results obtained on real traces are reasonable or not. We thus applied our tool on our 184 flows to obtain 818 stationary flows. To assess the relevance of the approach, we proceeded as follows: for any two neighboring stationary flows from the same flow, we compute their means μ_1 and μ_2 and their standard deviations σ_1 and σ_2. We then compute the "jump in mean" $\Delta_\mu = \frac{\mu_2 - \mu_1}{\mu_1} \times 100$ and "jump in standard deviation" $\Delta_\sigma = \frac{\sigma_2 - \sigma_1}{\sigma_1} \times 100$. We then break each stationary flows into two sub-flows of equal size and compute their means μ_1^i and μ_2^i and standard deviations σ_1^i and σ_2^i $(i = 1, 2)$. We can then define jumps in means and standard deviations between two sub-flows of a given stationary flow. The latter jumps are called intra jumps while the jumps between stationary flows are called inter jumps. The idea behind these definitions is to demonstrate that the distributions of intra jumps are more concentrated around their mean value than the distributions of inter jumps. To compare those distributions, we used boxplot representations. A boxplot of a distribution is a box where the upper line corresponds to the 75 percentile $\hat{p}_{0.75}$ of the distribution, the lower line to the 25 percentile $\hat{p}_{0.25}$ and the central line to the median. In addition, the $\hat{p}_{0.25} - 1.5 \times IQR$ and $\hat{p}_{0.75} + 1.5 \times IQR$ values ($IQR = \hat{p}_{0.75} - \hat{p}_{0.25}$ is the inter quantile range, which captures the variability of the sample) are also graphed while the samples falling outside these limits are marked with a cross. A boxplot allows to quickly compare two distributions and to assess the symmetry and the dispersion of a given distribution. In figure 6, we plotted the boxplots for the inter jump in mean (left side) and intra jump in mean (right side). From these representations, we immediately see that the intra jump distribution is thinner than the inter jumps distribution which complies with our initial intuition. Note also that the means of the inter and intra jump distributions are close to zero as the Δ_μ definition can result in positive or negative values and it is quite reasonable that overall, we observe as much positive

as negative jumps. Figure 7 depicts the boxplots for the inter and intra jumps in standard deviations. The results are somehow similar to the ones for jumps in mean although less pronounced and more skewed toward large positive values.

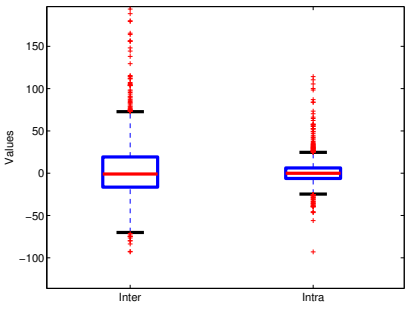


Fig. 6. Boxplot representation of the inter jump in mean (left side) and intra jump in mean (right side)

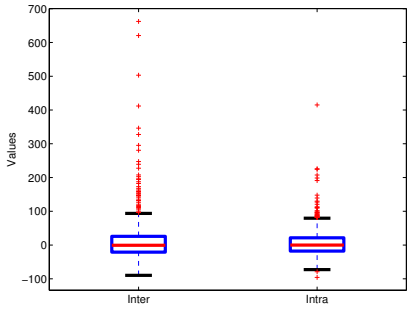


Fig. 7. Boxplot representation of the inter jump in standard deviation(left side) and intra jump in standard deviation (right side)

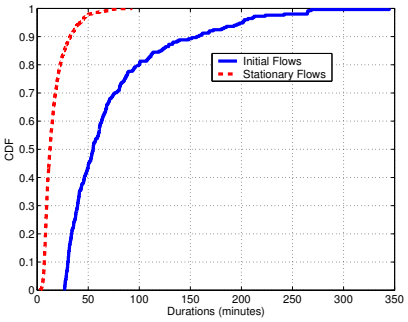


Fig. 8. CDFs of flows and stationary flows durations

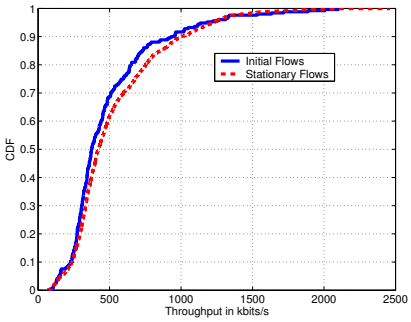


Fig. 9. CDFs of flows and stationary flows throughputs

5 Results on the BitTorrent Dataset

5.1 Stationary periods characterization

As stated in the previous section, the K-S change point tool has extracted 818 stationary flows out of the 184 initial flows. This means that, on average, a flow is cut into 4.45 stationary flows. Figure 8 represents the cumulative distribution functions (cdf) of the duration of stationary and initial flows. Stationary flows have an average duration of 16.4 minutes while initial flows have an average duration of 73 minutes.
Figure 9 represents the cumulative distribution functions of throughputs of the

stationary and initial flows. Overall, stationary flows tend to exhibit larger throughputs than initial flows. Indeed, the mean throughput of stationary flows is 493.5 kbits/s as compared to 444 kbit/s for the initial ones. This discrepancy is an indication that the K-S change point test is working properly as it extracts from the initial flows stationary periods where the throughputs significantly differ from the mean throughput of the flow. The cdfs differ at the end because whenever the K-S test exhibits a small period (relative to the flow it is extracted from) with high throughput, it will become one sample for the cdf of stationary flows, whereas it might have little impact for the corresponding sample for the cdf of the initial flows (if the high throughput part only corresponds to a small fraction of the initial flow).
Using our tool, we can also investigate transitions between consecutive stationary periods. The left boxplot of figure 6 allows us to look globally at transitions between stationary periods. From this figure, we can observe that most of the changes result in jumps of the mean value that are less than 20% in absolute values. This is encouraging for applications that can tolerate such changes in their observed throughput since they can expect to experience quite long stable periods, typically several tens of minutes (at least in the context of our dataset). However, a lot of values fall outside the plus or minus $1.5 \times IQR$ interval, meaning that some transitions are clearly more sharp than others.

5.2 The case of receiver window limited connections

In a effort to relate the stationarity observed by our tool to the intrinsic characteristics of the connections, we considered the case of receiver window limited flows. A receiver window limited flow is a flow whose throughput is limited by the advertised window of the receiver. The motivation behind this study is that as receiver window limited flows are mostly constrained by some end hosts characteristics (the advertised window of the receiver), they should exhibit longer stationary periods than other flows. Indeed, the intuition is that those other flows have to compete "more" for resources along their path with side traffic, which should affect their throughput, leading to change points.
We first have to devise a test that flags receiver window limited flows. We proceed as follows. For each flow, we generate two time series with a granularity of 10 seconds. The first time series, $Adv(t)$ represents the advertised window of the receiver while the second one, $Out(t)$ accounts for the difference between the maximum unacknowledged byte and the maximum acknowledged byte. The second time series provides an estimate of the number of outstanding bytes on the path at a given time instant. The $Out(t)$ time series is accurate except during loss periods. Note that the computation of $Out(t)$ is possible since our dataset was collected at the sender side, as the Eurecom peer in the BitTorrent session was acting as a seed during the measurement period. A flow is then flagged receiver window limited if the following condition holds:

$$\frac{\sum_{t=1}^{N} 1_{Adv(t)-3\times MSS \leq Out(t) \leq Adv(t)}}{N} \geq 0.8$$

where N is the size of the two time series and MSS is the maximum segment size of the path. The above criterion simply states that 80% of the time, the estimated number of outstanding packets must lie between the advertised window minus three MSS and the advertised window. By choosing a treshold of 80%, we expect to be conservative.
Application of the test on our dataset leads us to flag about 13.7% of the flows as receiver window limited. The next issue is to choose the non window limited flows. We adopt the following criterion:

$$\frac{\sum_{t=1}^{N} 1_{Out(t) \leq Adv(t) - 3 \times MSS}}{N} \geq 0.9$$

Applying the above criterion, we obtained about 14.4% of non receiver window limited flows. A straightforward comparison of the durations of the stationary flows extracted from the flows of the two families (receiver window limited and non receiver limited) is misleading as the duration of their respective connections is different. We thus use two other metrics. First, we compute the number of stationary flows into which a flow is cut in each family. We obtain that the receiver window limited flows are on average cut into 3.5 stationary flows while non receiver window limited flows are cut into 4.5 stationary flows. The second metric we consider is the relative size, in percentage, of the stationary flows with respect to the size of flow they are extracted from for the two familied. Figure 10 represents the cumulative distribution functions of the percentages for the two families. From this figure, we observe that receiver window limited stationary flows are relatively larger than non receiver window limited ones in most cases. Also, in figure 11, we plot the cumulative distributions of the throughput of the stationary flows for both families. We conclude from figure 11 that receiver window limited stationary flows exhibit significantly smaller throughputs values than non receiver window limited ones. This might mean that receiver limited flows correspond to paths with larger RTT than non receiver window limited ones, as this would prevent these flows from achieving high throughput values. This last point as well as our definition of window limited flows (we only considered around 28% of the flows of our dataset to obtain those results) would clearly deserve more investigation.

6 Conclusion and Outlook

Internet Traffic analysis becomes a crucial activity, e.g. for ISPs to do troubleshooting or for content providers and researchers that are willing to devise new multimedia services in the Internet. Once information on some path has been collected, its needs to be analyzed. The first step is to divide traces into somewhat homogeneous period and to flag anomalies. In this paper, we concentrate on the analysis of the service perceived by long TCP connections in the Internet. We have developed a change point analysis tool that extracts stationary periods within connections. We follow a non parametric approach and based our tool on the Kolmogorov-Smirnov goodness of fit test. We validated our change

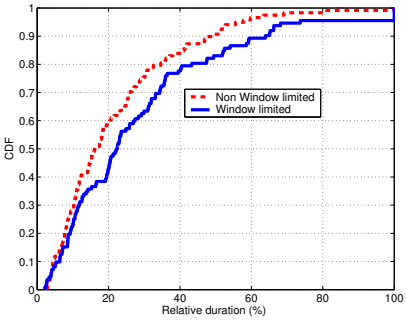


Fig. 10. Histogram of relative size of rec. window and non rec. window limited stationary flows

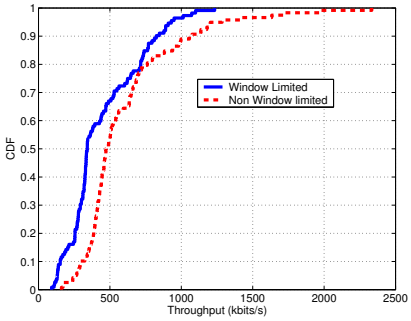


Fig. 11. Histogram of rec. window and non rec. window limited stationary flows throughputs

point tool in various ways on synthetic and operational datasets. Overall, the tool manages to correctly flag change points as long as little correlation persists at the time scale at which it is applied. We worked at the 10 second time scale, which is a reasonable time scale for some multimedia applications such as VoD. We also focused on receiver window limited connections to relate the stationarity observed by our tool to typical connection behaviors.
As future work, we intent to pursue in this direction by correlating the stationarity periods with some other network events like RTT variations or loss rates. We would also like to study the extent to which our tool could be used in real time and to investigate how it could be tailored to the need of some specific applications. It is also necessary to compare our tool with some other change point techniques [1].

Acknowledgment

The author is extremely grateful to the anonymous reviewers for their valuable comments and to M. Siekkinen for the trace collection and time series extraction.

References

1. M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, Prentice-Hall, Inc. ISBN 0-13-126780-9, 1993.
2. N. Ben Azzouna, F. Clerot, C. Fricker, and F. Guillemin, "Modeling ADSL traffic on an IP backbone link", *Annals of Telecommunications*, December 2004.
3. S. Ben-David, J. Gehrke, and D. Kifer, "Detecting changes in data streams", In *Proceedings of the 30th International Conference on Very Large Databases*, 2004.
4. J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the nonstationarity of Internet traffic", In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 102–112, ACM Press, 2001.

5. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", In *Proceedings of SOSP'03*, New York, USA, October 2003.
6. C. Chatfield, *The analysis of time series - An introduction*, Chapman & Hall, London, UK, 1996.
7. P. De Cuetos, P. Guillotel, K. Ross, and D. Thoreau, "Implementation of Adaptive Streaming of Stored MPEG-4 FGS Video over TCP", In *International Conference on Multimedia and Expo (ICME02)*, August 2002.
8. J. Dilley, B. Maggs, J. Parikh, H. Prokop, and R. Sitaraman, and B. Weihl, "Globally distributed content delivery", *Internet Computing, IEEE*, pp. 50–58, Sept.-Oct 2002.
9. H. Eghbali, "K-S Test for Detecting Changes from Landsat Imagery Data", *IEEE Trans Syst., Man & Cybernetics*, 9(1):17–23, January 1979.
10. M. Fomenkov, K. Keys, D. Moore, and k claffy, "Longitudinal study of Internet traffic from 1998-2003", Cooperative Association for Internet Data Analysis - CAIDA, 2003.
11. M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", In *Passive and Active Measurements 2004*, April 2004.
12. M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", *IEEE/ACM Transactions on Networking*, 11(4):537–549, 2003.
13. T. Karagiannis and et al., "A Nonstationary Poisson View of Internet Traffic", In *Proc. Infocom 2004*, March 2004.
14. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications", In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 234–247, ACM Press, 2003.
15. A. Markopoulou, F. Tobagi, and M. J. Karam, "Assessing the quality of voice communications over Internet backbones", *IEEE/ACM Transactions on Networking*, 11:747–760, October 2003.
16. S. Siegel and N. J. Castellan, *Nonparametric statistics for the Behavioral Sciences*, McGraw-Hill, 1988.
17. Y. Zhang, V. Paxson, and S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput", ACIRI, May 2000.

Analysis of LAS Scheduling for Job Size Distributions with High Variance*

Idris A. Rai, Guillaume Urvoy-Keller, Ernst W. Biersack
Institut Eurecom
2229, route des Crêtes
06904 Sophia-Antipolis, France
{rai,urvoy,erbi}@eurecom.fr

ABSTRACT

Recent studies of Internet traffic have shown that flow size distributions often exhibit a high variability property in the sense that most of the flows are short and more than half of the total load is constituted by a small percentage of the largest flows. In the light of this observation, it is interesting to revisit scheduling policies that are known to favor small jobs in order to quantify the benefit for small and the penalty for large jobs. Among all scheduling policies that do not require knowledge of job size, the *least attained service* (LAS) scheduling policy is known to favor small jobs the most. We investigate the M/G/1/LAS queue for both, load $\rho < 1$ and $\rho \geq 1$. Our analysis shows that for job size distributions with a high variability property, LAS favors short jobs with a negligible penalty to the few largest jobs, and that LAS achieves a mean response time over all jobs that is close to the mean response time achieved by SRPT.

Finally, we implement LAS in the ns-2 network simulator to study its performance benefits for TCP flows. When LAS is used to schedule packets over the bottleneck link, more than 99% of the shortest flows experience smaller mean response times under LAS than under FIFO and only the largest jobs observe a negligible increase in response time. The benefit of using LAS as compared to FIFO is most pronounced at high load.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance attributes.

General Terms

Performance, Experimentation.

*Institut Eurécom's research is partially supported by its industrial members: Bouygues Télécom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom , Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

SIGMETRICS'03, June 10–14, 2003, San Diego, California, USA.

Keywords

Size-based scheduling, least attained service, high variability property, Web objects response time.

1. INTRODUCTION

Internet traffic studies revealed that Internet traffic consists of many short flows and that a tiny fraction of the largest flows constitutes more than half of the total load [6, 21]. It is interesting to study the performance of scheduling policies in the light of these findings to see if one can improve the performance of short jobs without penalizing too much the long jobs. We say that a job size distribution exhibits the *high variability property* if less than 1% of the largest jobs account for more than half of the load. This property has also been referred to as *heavy-tail property* [7, 2], but it is not restricted to heavy-tail distributions [12]. In this paper, we analyze the impact of the high variability property of job size distributions on the performance of the LAS scheduling policy.

Response time and *slowdown* are commonly used as performance metrics to analyze scheduling policies. Response time is the overall time a job spends in the system. Slowdown is the normalized response, i.e., the ratio of the response time of a job to the size of that job. In particular, we use the **conditional mean response time** and the **conditional mean slowdown**, which (for a job of size x) are defined as $E[T(x)] \triangleq E[(T|X = x)]$ and $E[S(x)] \triangleq E[(S|X = x)]$ respectively. We also use the **mean response time** and the **mean slowdown** defined as $E[T] \triangleq \int_0^\infty E[T(x)]f(x)dx$ and $E[S] \triangleq \int_0^\infty E[S(x)]f(x)dx$ respectively. Slowdown is a useful metric to analyze fairness [2]. Processor Sharing (PS) is known to be a fair policy since it offers the same slowdown to all jobs.

The *shortest remaining processing time* (SRPT) scheduling policy is proven [20, 25] to be the optimal policy for minimizing the mean response time. It has been known for a long time that for negative exponentially distributed job sizes, SRPT severely penalizes large jobs. However, Bansal and Harchol-Balter [2] have recently shown that the performance of SRPT and the penalty experienced by the largest jobs very much depend on the characteristics of the job size distribution. In particular, for some load values and for job size distributions with the high variability property, SPRT does not increase at all the conditional mean response time of long jobs as compared to PS, i.e., all jobs have a lower conditional mean slowdown under SRPT than under PS. Nevertheless, SRPT has the drawback that it needs to know the size of the jobs. While the job size is known, for instance, in a Web server with static Web pages [14], it is generally not known in environments such as in Internet routers or Web servers with dynamic pages.

The *least attained service* (LAS) scheduling policy is a multi-level scheduling policy that favors small jobs without requiring knowledge about the job sizes. LAS is a preemptive scheduling policy that gives service to the job in the system that has received the least service. In the event of ties, the set of jobs having received the least service shares the processor in a processor-sharing mode. A newly arriving job always preempts the job (or jobs) currently in service and retains the processor until it departs, or until the next arrival appears, or until it has obtained an amount of service equal to that received by the job(s) preempted on arrival, whichever occurs first. An implementation of LAS needs to know the amount of service it has delivered to each job, which can be easily obtained from the server. LAS is also known in the literature under the names of *foreground-background* (FB) [17] or *shortest elapsed time* (SET) first scheduling [4].

The expression for the conditional mean response time $E[T(x)]$ for an M/G/1 queue served using LAS has been originally derived in [24], and were re-derived in [28, 4, 18] as well. However, the expression for $E[T(x)]$ is complex and difficult to evaluate numerically, and therefore very little has been done to evaluate the performance of LAS with respect to the variability of the job size distribution. The variability of a job size distribution can be expressed in terms of its **coefficient of variation** (C), which is defined as *the ratio of the standard deviation to the mean of the distribution*. For a given mean, the higher the variance, the higher the C value and so the higher the variability of the distribution. Coffman ([4], pp. 188-187), gives an intuitive result that compares the mean response times for LAS and PS: The mean response time of LAS is lower than the mean response time of PS for job size distributions with a C greater than 1, and it is higher for job size distributions with a C less than 1. Kleinrock ([17], pp. 196-197) compares the mean waiting times of LAS and FIFO scheduling for an M/M/1 queue and observes that LAS offers lower waiting times to short jobs while large jobs see significantly higher waiting times. The analysis of $M/G/1/LAS$ queue at steady state was done by Schassberger [22, 23]. However, Schassberger did not evaluate the impact of the variance of the service time distribution. In a recent work, Balter et al. [15] show that for an M/G/1 queue, the conditional mean slowdown of all work-conserving scheduling policies, which includes LAS, asymptotically converges to the conditional mean slowdown of PS, i.e. $\lim_{x\to\infty} E[S(x)]_{LAS} = 1/(1-\rho)$.

This paper investigates the performance of LAS considering the variability of job size distributions. We first compare LAS to PS to analyze its unfairness. We derive an upper bound on the unfairness of LAS for a general job size distribution (Theorem 1). In particular, for job size distributions that exhibit the high variability property, we observe that more than 99% of the jobs see their conditional mean slowdown significantly reduced under LAS and less than 1% of the largest jobs experience a negligible increase of their conditional mean slowdown under LAS. We also derive an upper bound that compares the mean response times of LAS and nonpreemptive policies (NPP) (Lemma 2).

While both, LAS and SRPT, favor short jobs, it is not known how the performance of LAS compares to the optimal policy SRPT. We provide mathematical expressions that compare these two policies and show that for job size distributions with the high variability property, both policies offer very similar conditional mean response times to all jobs (Theorem 4).

We also consider the overload case, i.e. $\rho \geq 1$, and prove the existence of a job size $x_{LAS}(\lambda)$ such that all jobs strictly smaller in size than $x_{LAS}(\lambda)$ will have a finite response time under LAS (Theorem 5). We then present a closed form expression for the conditional mean response time of LAS under overload (Theorem 6). We evaluate this expression by comparing the performance of LAS with PS and SRPT for $\rho \geq 1$. PS is known to exhibit an infinite mean response time under overload as opposed to LAS, which offers finite mean response time for job sizes less than $x_{LAS}(\lambda)$. The results are intriguing; for the job size distribution with the high variability property considered in this paper, at overload of $\rho = 2$, jobs of size up to the 99th percentile have a finite conditional mean response time under LAS.

The analysis conducted for M/G/1/LAS uses the notion of a job, which is defined as the entity that requests service from system *at once*. In packet networks we are interested in the performance of flows. However, the packets of a flow are spaced in time and are statistically multiplexed with packets from other flows. A flow therefore does not arrive at once in the system. To evaluate the performance of LAS in a packet network, we implement LAS in the ns-2 network simulator [16] and analyze it for Web-traffic with Pareto-distributed flow sizes. The simulation results for different load values show that more than 99% of the short flows generated during the simulation benefit from LAS as compared to FIFO and a negligible percentage of the largest flows experience a higher response time under LAS than under FIFO. Also, the difference in the mean response time between LAS and FIFO increases with increasing load.

The paper is organized as follows: In the next section we discuss the mathematical background necessary for the analysis carried out in the rest of the paper. We analytically compare LAS to PS, to nonpreemptive (NPP) policies, and to SRPT in Section 3. In Section 4, we investigate the performance of LAS under overload. In Section 5 we study the performance of LAS in a bottleneck router and compare the response time of flows under LAS and FIFO. We conclude the paper in Section 6.

2. MATHEMATICAL BACKGROUND

In this section we first present mathematical expressions of the conditional mean response time and the conditional mean slowdown for different scheduling policies. Then we discuss the reasons for the choice of the particular job size distributions used in this paper.

2.1 Expressions for the Performance Metrics

Let the average job arrival rate be λ. Assume a job size distribution X with a probability mass function $f(x)$. The abbreviation c.f.m.f.v is used to denote continuous, finite mean, and finite variance. Given the cumulative distribution function as $F(x) \triangleq \int_0^x f(t)dt$, we denote the survivor function of X as $F^c(x) \triangleq 1 - F(x)$. We define $m_n(x)$ as $m_n(x) \triangleq \int_0^x t^n f(t)dt$. Thus $m_1 \triangleq m_1(\infty)$ is the mean and $m_2 \triangleq m_2(\infty)$ is the second moment of the job size distribution. The load of jobs with size less than or equal to x is given as $\rho(x) \triangleq \lambda \int_0^x tf(t)dt$, and $\rho \triangleq \rho(\infty)$ is the total load in the system.

In most cases, this paper assumes an M/G/1 queue, where G is a c.f.m.f.v distribution. The expression of $E[T(x)]$ for M/G/1/SRPT [26] can be decomposed into the conditional mean waiting time $E[W(x)]$ (the time between the instant when a job arrives at the system until it receives the service for the first time) and the conditional mean residence time $E[R(x)]$ (the time a job takes to complete service once the server has started serving it). Hence,

$$E[T(x)]_{SRPT} = E[W(x)]_{SRPT} + E[W(x)]_{SRPT}$$

$$E[W(x)]_{SRPT} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x))^2} \quad (1)$$

$$E[R(x)]_{SRPT} = \int_0^x \frac{1}{1-\rho(t)} d(t) \quad (2)$$

Similarly, we decompose the expression of the $E[T(x)]$ for $M/G/1/LAS$ into two terms. We denote the first term as $E[\tilde{W}(x)]_{LAS}$ and the second term as $E[\tilde{R}(x)]_{LAS}$. Note that these terms do not denote the conditional mean waiting and residence times for LAS, rather they are introduced to ease the analysis in Section 3.3 where we compare LAS to SRPT. The formulas for $E[T(x)]_{LAS}$ are given in [24],

$$E[T(x)]_{LAS} = E[\tilde{W}(x)]_{LAS} + E[\tilde{R}(x)]_{LAS}$$

$$E[\tilde{W}(x)]_{LAS} = \frac{\lambda(m_2(x) + x^2 F^c(x))}{2(1-\rho(x) - \lambda x F^c(x))^2} \quad (3)$$

$$E[\tilde{R}(x)]_{LAS} = \frac{x}{1-\rho(x) - \lambda x F^c(x)} \quad (4)$$

Finally, the formulas of $E[T(x)]$ for the M/G/1/PS and M/G/1/NPP [5] (where NPP stands for any non-preemptive policy) are:

$$E[T(x)]_{PS} = \frac{x}{1-\rho} \quad (5)$$

$$E[T(x)]_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + x \quad (6)$$

The definition of $E[S(x)]$ reveals that for two scheduling policies A and B, the ratio $\frac{E[T(x)]_A}{E[T(x)]_B} = \frac{E[T(x)]_A/x}{E[T(x)]_B/x} = \frac{E[S(x)]_A}{E[S(x)]_B}$.

2.2 The Choice of the Job Size Distribution

We will analyze LAS by using a general c.f.m.f.v job size distribution and some specific job size distributions. The choice of specific job size distributions is motivated by the characteristics of the Internet traffic where *most of the flows are short and more than half of the load is due to a tiny fraction of the largest flows*. Different distributions have been shown to model the empirical Internet traffic given their coefficient of variation is larger than 1 [1, 6, 10]. These distributions include Pareto, bounded Pareto, lognormal distributions, hyper-exponential, Weibull, inverse Gaussian, and hybrid of lognormal and Pareto distributions. The bounded Pareto (BP) distribution is commonly used in analysis because it can take a high variance and thus it can exhibit the high variability property as observed in the Internet traffic and also because the maximum job size can be set to mimic the largest Internet flow size [29, 7, 2]. In this paper, we also use the bounded Pareto job size distribution for the same reasons.

We denote the bounded Pareto distribution by $BP(k, p, \alpha)$, where k and p are the minimum and the maximum job sizes and α is the exponent of the power law. We will also consider exponentially distributed job sizes to see the performance difference when LAS is analyzed under M/M/1 queue model. Let X be a job size distribution, the probability density functions of the bounded Pareto ($f(x)_{BP}$) and the exponential $f(x)_{Exp}$ job sizes are given as:

$$f(x)_{BP} = \frac{\alpha k^\alpha}{1-(k/p)^\alpha} x^{-\alpha-1}, \; k \le x \le p, 0 \le \alpha \le 2 \quad (7)$$

$$f(x)_{Exp} = \mu e^{-\mu x}, \qquad x \ge 0, \mu \ge 0$$

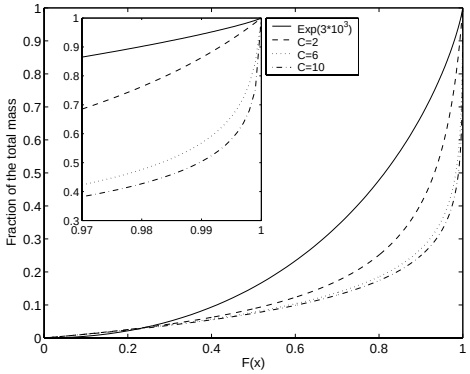
We also denote the exponential distribution with mean of $1/\mu$ by $Exp(1/\mu)$. The cumulative distribution function $F(x)$ and the n-th moment m_n of the $BP(k, p, \alpha)$ distribution are:

$$F(x) = \frac{1}{1-(k/p)^\alpha}[1-(k/x)^\alpha], \; k \le x \le p, 0 \le \alpha \le 2$$

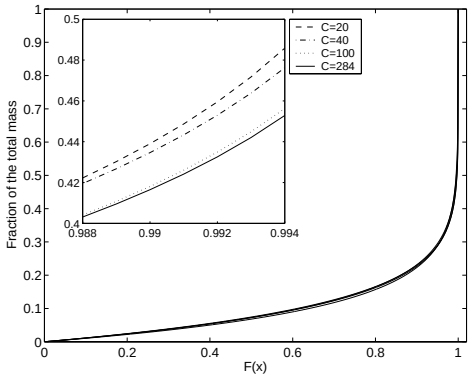
$$m_n = \frac{\alpha}{(n-\alpha)(p^\alpha - k^\alpha)}(p^n k^\alpha - k^n p^\alpha)$$

Hence, the expression for coefficient of variation is $C \triangleq \frac{\sqrt{m_2 - m_1^2}}{m_1}$. To obtain different values of C for $BP(k, p, \alpha)$, one needs to change one or more of the parameters of the BP distribution, i.e., k, α, and p. The C value of exponential distribution is always 1.

The variability of a distribution can be determined by using the *mass-weighted function* ($M_w(x)$) [8], which (for a job of size x) is defined as the fraction of the total load constituted by jobs of size less than or equal to x or $M_w(x) \triangleq \frac{\rho(x)}{\rho}$. We plot $M_w(x)$ as a function of $F(x)$ to see the fraction of total mass constituted by jobs of size less than or equal to x. The mass-weighted function will help us to see if a job size distribution exhibits the high variability property.



(a) $Exp(3 * 10^3)$ *and BP with* $C = 2, 6, 10$



(b) *BP with* $C = 20, 40, 100, 284$

Figure 1: ***Mass-weighted functions for*** $BP(k, p, \alpha)$ ***and*** $Exp(3 * 10^3)$ ***job size distributions.***

Figure 1 shows the mass-weighted functions for the BP job size distribution with different C values and the exponential job size distribution, each with the same mean value of $3 * 10^3$. Figure 1(a) shows that for the exponential distribution, the largest 1% of

the jobs constitute only about 5% of the total load, and that for BP the variability increases with increasing C value. We observe that the load constituted by the largest 1% of the jobs increases with increasing C from 15% of the total load for $C = 2$ to close to 50% for $C = 10$. On the other hand, Figure 1(b) shows that all BP distributions with $C \geq 20$ exhibit the high variability property since 50% of the total load is constituted by less than 1% of the largest jobs.

In this paper, we shall use the BP job size distribution $BP(332, 10^{10}, 1.1)$ with $C = 284$ as a typical example of the BP distribution that exhibits the high variability property. This distribution was also used in [2] to analyze the unfairness of SRPT. BP distributions with similar parameters are also used in [29, 7]. In addition to the BP with $C = 284$, we also use other BP distributions with $C < 284$ and the exponential distribution $Exp(3 * 10^3)$ to see the impact of degree of variability on the performance of LAS.

3. ANALYTICAL RESULTS

Previously, no analysis of M/G/1/LAS has been carried out for different values of C. The analysis for M/G/1/LAS is tedious as it involves nested integrals of complicated functions. Here, we provide elements to compare M/G/1/LAS to M/G/1/PS, and M/G/1/SRPT systems.

3.1 Comparison for a general job size distribution

Processor sharing is a well known fair scheduling policy that at any load $\rho < 1$ gives the same conditional mean slowdown to all jobs, i.e., $E[S(x)]_{PS} = \frac{1}{(1-\rho)}\ \forall x$. In this section, we assume a general c.f.m.f.v job size distribution to investigate the unfairness of the LAS policy by comparing its slowdown to the slowdown of PS. Nonpreemptive (NPP) policies include FIFO scheduling, which is commonly implemented in todays routers. We also compare LAS with NPP policies to investigate the performance benefits offered under LAS in terms of mean response time with varying C and ρ values.

3.1.1 Comparison between LAS and PS

We define the function ϕ as $\phi(x) \triangleq \lambda \int_0^x t f(t)dt + \lambda x F^c(x)$. If we integrate $\int_0^x t f(t)dt$ by parts in the definition of ϕ we obtain $\phi(x) = \lambda \int_0^x F^c(t)dt$. The following result holds for the conditional mean response time of a job of size x:

Theorem 1. *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$E[T(x)]_{LAS} \leq (1-\rho)\frac{2-\phi(x)}{2(1-\phi(x))^2}E[T(x)]_{PS} \quad (8)$$

$$E[S(x)]_{LAS} \leq (1-\rho)\frac{2-\phi(x)}{2(1-\phi(x))^2}E[S(x)]_{PS} \quad (9)$$

PROOF.

$$E[T(x)]_{LAS} = \frac{\lambda \int_0^x t^2 f(t)dt + \lambda x^2 F^c(x))}{2(1-\phi(x))^2} + \frac{x}{1-\phi(x)}$$

$$\leq \frac{\lambda x \int_0^x t f(t)dt + \lambda x^2 F^c(x))}{2(1-\phi(x))^2} + \frac{x}{1-\phi(x)} \quad \text{since } \int_0^x t^2 f(t)dt \leq x\int_0^x t f(t)dt$$

$$= E[T(x)]_{PS}\frac{1-\rho}{1-\phi(x)}\Big(\frac{\lambda \int_0^x t f(t)dt + \lambda x F^c(x)}{2(1-\phi(x))} + 1\Big) \quad \text{since } E[T(x)]_{PS} = \frac{x}{1-\rho}$$

$$= E[T(x)]_{PS}\frac{1-\rho}{1-\phi(x)}\Big(\frac{\phi(x)}{2(1-\phi(x))} + 1\Big)$$

By definition of $\phi(x)$

$$= (1-\rho)\frac{2-\phi(x)}{2(1-\phi(x))^2}E[T(x)]_{PS} \quad (10)$$

Equation (9) follows directly by dividing both sides of Equation (10) by x. □

We use Theorem 1 to elaborate on the comparison between the mean slowdown of LAS and PS. We first introduce a lemma that we need in the proof of Theorem 2.

Lemma 1. *For any load $\rho < 1$,*

$$\frac{2-\phi(x)}{2(1-\phi(x))^2} \leq \frac{2-\rho}{2(1-\rho)^2} \quad (11)$$

PROOF. Function $\phi(x)$ is an increasing function for $x \in [0, \infty]$ since $\frac{d\phi(x)}{dx} = \lambda F^c(x) \geq 0$. Also, function $\frac{2-u}{2(1-u)^2}$ is an increasing function for $u \in [0, 1]$. Hence, $\frac{2-\phi(x)}{2(1-\phi(x))^2}$ is an increasing function of x and upper bounded by $\frac{2-\rho}{2(1-\rho)^2}$ since $\phi(x) \in [0, \rho] \subset [0, 1]$. □

Theorem 2. *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$E[S]_{LAS} \leq \frac{2-\rho}{2(1-\rho)}E[S]_{PS} \quad (12)$$

PROOF.

$$E[S]_{LAS} = \int_0^{+\infty} \frac{E[T(x)]_{LAS}}{x} f(x)dx$$

$$\leq \int_0^{+\infty} \frac{2-\phi(x)}{2(1-\phi(x))^2} f(x)dx \quad \text{Theorem 1} \quad (13)$$

$$\leq \frac{2-\rho}{2(1-\rho)^2}\int_0^{+\infty} f(x)dx \quad \text{Lemma 1}$$

$$= \frac{2-\rho}{2(1-\rho)^2}$$

$$= \frac{2-\rho}{2(1-\rho)}E[S]_{PS} \quad \text{Since } E[S]_{PS} = \frac{1}{1-\rho} \quad (14)$$

□

Corollary 1. *For all c.f.m.f.v job size distributions and load $\rho < 1$,*

$$E[T]_{LAS} \leq \frac{2-\rho}{2(1-\rho)}E[T]_{PS} \quad (15)$$

PROOF. The proof is similar to the proof of Theorem 2. □

We illustrate the result of Theorem 2 in Figure 2. We can see that the ratio between the mean slowdown of LAS and PS is bounded by a value that is less than 2 for $\rho \leq 0.66$ and less than 6 for $\rho \leq 0.9$. This result clearly supports the fact that using LAS instead of PS results in a better slowdown for small jobs without affecting too much larger jobs since the average slowdown over small and large jobs under PS and LAS remains fairly close for most system loads. The result also indicates that for low and medium load the mean slowdown of LAS remains close to the mean slowdown for PS. Note that the real ratio between the average slowdown of LAS and PS is below the value of $\frac{2-\rho}{2(1-\rho)}$ used in Figure 2 due to the rather crude majorization of $\phi(x)$ applied in Lemma 1.

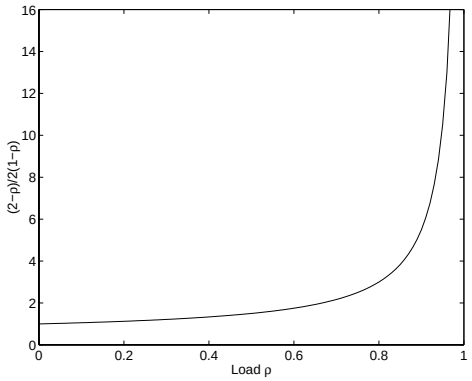


Figure 2: ***Upper bound on the mean slowdown ratio*** $\frac{E[S]_{LAS}}{E[S]_{PS}}$**.**

3.1.2 Comparison between LAS and Nonpreemptive policies (NPP)

We now compare the performance of LAS to NPP policies in terms of mean response time for job size distributions with varying C and for different values of load $\rho < 1$. Examples of nonpreemptive policies include FIFO, LIFO, and RANDOM. We derive an upper bound of the mean response time that is a function of C and ρ values. The bound shows that the mean response time offered by LAS improves in increasing C values. From Equations (5) and (6), we get the expressions of mean response times for PS and NPP policies as:

$$E[T]_{PS} = \frac{m_1}{(1-\rho)} \quad (16)$$

$$E[T]_{NPP} = \frac{\lambda m_2}{2(1-\rho)} + m_1 \quad (17)$$

Equation (17) can also be expressed as:

$$\begin{aligned} E[T]_{NPP} &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} + m_1 - \frac{m_1}{(1-\rho)} \\ &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} - \frac{\rho m_1}{(1-\rho)} \\ &= \frac{m_1}{(1-\rho)} + \frac{\lambda m_2}{2(1-\rho)} - \frac{\lambda m_1^2}{(1-\rho)} \\ &\quad \text{from } \rho = \lambda m_1 \\ &= \frac{m_1}{(1-\rho)} + \frac{\rho[C^2-1]}{2(1-\rho)} m_1, \\ &\quad \text{from } m_1^2 C^2 = m_2 - m_1^2 \text{ and } \lambda = \rho/m_1 \\ &= E[T]_{PS} + \frac{\rho[C^2-1]}{2(1-\rho)} m_1, \end{aligned} \quad (18)$$

Lemma 2. *For a c.f.m.f.v. job size distribution with mean m_1 and coefficient of variation C, the mean response time under LAS at load ρ ($E[T]_{LAS}$) is upper bounded as:*

$$E[T]_{LAS} \leq \frac{(2-\rho)}{2(1-\rho)} E[T]_{NPP} - \frac{\rho(2-\rho)[C^2-1]}{4(1-\rho)^2} m_1 \quad (19)$$

PROOF. The proof follows directly when substituting $E[T]_{PS}$ obtained from Equation (18) in Corollary 1. □

The bound on the mean response time of LAS (Equation (19)) is a function of C and load ρ, since $E[T]_{NPP}$ itself is a function of C and load ρ (see Equation 18) for a given value of m_1. This bound is interesting since it enables us to compare the performance of LAS relative to that of any nonpreemptive policy for job size distributions with a large range of C values. Figure 3 shows the upper

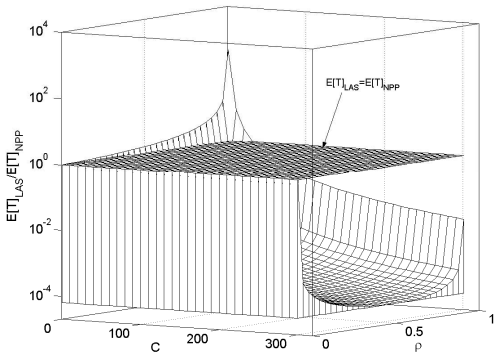


Figure 3: ***Upper bound on the mean response time ratio*** $\frac{E[T]_{LAS}}{E[T]_{NPP}}$ ***as a function of load*** ρ ***and coefficient of variation*** C**.**

bound on the ratio of the mean response time of LAS to the mean response time of a nonpreemptive policy as a function of $C \geq 1$ and load $\rho < 1$. Observe that LAS has a higher mean response time than nonpreemptive policies only for distributions with C values close to 1. In this case, we also observe that the mean response time ratio increases to large values with increasing load. On the other hand, the mean response time of LAS is lower than that of nonpreemptive policies for distributions with higher C values at all load values $\rho < 1$. Generally, for a given load ρ, the ratio $\frac{E[T]_{LAS}}{E[T]_{NPP}}$ decreases with increasing C.

In summary, we see that LAS achieves lower mean response time than any NPP policy, such as the FIFO policy, for job size distribution with a high variance. This is to be expected because in contrast to LAS, the service of a job under FIFO is not interrupted until the job leaves the system and so large jobs are favored over small jobs.

3.2 Distribution dependent comparison

In this section, we compare LAS to PS for job size distributions with varying C values. A comparison of LAS with PS helps to analyze the degree of unfairness (penalty) seen by the largest jobs under LAS. We start with general results for exponentially distributed job sizes. We show that for the case of the exponential distribution, the average slowdown of jobs under LAS is always better than the average slowdown of the jobs under PS.

Theorem 3. *For an exponential job size distribution and load $\rho < 1$,*

$$E[S]_{LAS} \leq E[S]_{PS} \quad (20)$$

PROOF. Following the same reasoning as in Theorem 2, one obtains:

$$E[S]_{LAS} = \int_0^{+\infty} \frac{E[T(x)]_{LAS}}{x} f(x)dx$$

Using Equation (13) in Equation (21), we get,

$$E[S]_{LAS} \leq \int_0^{+\infty} \frac{x(2-\phi(x))}{2(1-\phi(x))^2} \frac{f(x)}{x} dx \quad (21)$$

For exponential job sizes, we have $\phi(x) = \int_0^x \lambda F^c(t)dt = \rho(1 - e^{-\mu x}) = \rho F(x)$ and $f(x) = \mu e^{-\mu x}$. Using these facts and plug-

ging $h(\phi(x)) \triangleq \frac{2-\phi(x)}{2(1-\phi(x))^2}$ in Equation (21) we obtain:

$$E[S]_{LAS} \leq \frac{1}{\rho}\int_0^{+\infty} h(\rho F(t))\rho f(t)dt$$

Replacing $\rho F(t)$ by u in the above equation we get:

$$\begin{aligned} E[S]_{LAS} &\leq \frac{1}{\rho}\int_0^{+\rho} h(u)du \\ &= \frac{1}{2\rho}\Big(-\ln(1-\rho)+\frac{\rho}{1-\rho}\Big) \\ &= \frac{1}{2\rho}\Big(-(1-\rho)\ln(1-\rho)+\rho\Big)E[S]_{PS} \quad (22) \\ &\quad \text{since } E[S]_{PS} = \frac{1}{1-\rho} \end{aligned}$$

A straightforward study of the derivative of ψ defined as $\psi(\rho) \triangleq \frac{(\rho-(-1-\rho)\ln(1-\rho))}{2\rho}$ indicates that ψ is a decreasing function and $\psi(\rho) \sim_{\rho\to 0} \frac{2\rho-\rho^2}{2\rho} \to 1$. Thus, $\forall \rho < 1, \psi(\rho) < 1$. □

This result shows that even for the case of exponentially distributed job sizes, where the high variability property does not hold, the mean slowdown is at most $\frac{1}{(1-\rho)}$. We could not derive bound similar to the one of theorem 3 for the BP distribution. However, we expect LAS to perform better in terms of the mean slowdown for job size distributions with $C > 1$ than for exponential job size distribution.

Next, we investigate the unfairness of LAS for job size distributions with varying C values. We use the exponential distribution (with $C = 1$) and the BP distributions with C values 2, 6, 20, and 284 all with the same mean value of $3 * 10^3$. To obtain BP job size distributions with different C values and the same mean we varied the values of α and p.

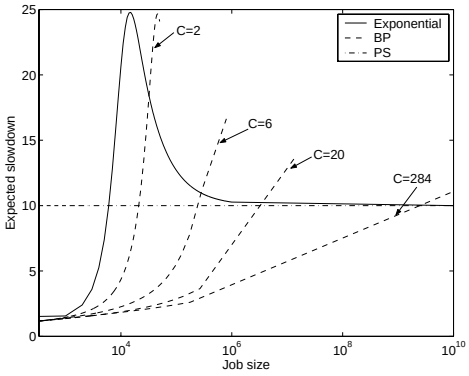
Figure 4(a) and 4(b) show the conditional mean slowdown as a function of job size and percentile respectively. We see from both figures that the percentage of the largest jobs that experience a penalty under LAS (i.e., have higher mean slowdown under LAS than under PS) and the degree of penalty decreases in increasing C value. For the BP distribution with $C \geq 6$, less than 0.5% of the largest jobs suffer a small penalty and the performance difference between $C = 20$ and $C = 284$ is minor. We observe that as the C value increases the penalty experienced by the largest jobs and the percentage of these largest jobs decrease. In the remainder of this paper we will only consider $Exp(3 * 10^3)$ and the $BP(332, 10^{10}, 1.1)$ distribution when we numerically analyze LAS.

3.3 Quantitative comparison between LAS and SRPT

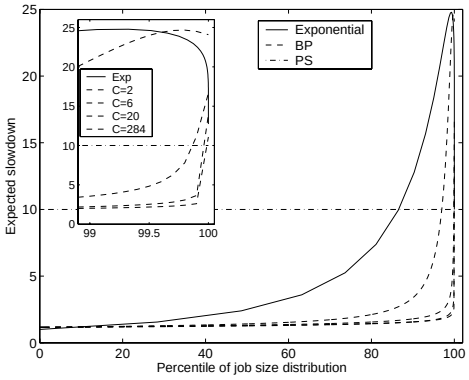
It is stated in [2] that for any job size distribution and at any load $\rho < 1$ every job x has higher mean response time ($E[T(x)]$) under LAS than under SRPT. However, it was not elaborated how much higher the mean slowdown of a job under LAS can be as compared to SRPT. We now compare LAS and SRPT quantitatively and show that at load $\rho < 1$, the conditional mean response time of a job under LAS is quite close to the mean response time under SRPT.

Theorem 4. *Let $\phi(x) \triangleq \rho(x) + x\lambda F^c(x)$, then for all c.f.m.f.v job size distributions and at load $\rho < 1$,*

$$E[T(x)]_{SRPT} \leq E[T(x)]_{LAS} \leq \Big(\frac{1-\rho(x)}{1-\phi(x)}\Big)^2 E[T(x)]_{SRPT} \quad (23)$$



(a) *Job size*



(b) *Percentile of job size distribution*

Figure 4: ***Expected slowdown under LAS and PS for different job size distributions and different C values.***

PROOF. Since $\rho(x) \leq \phi(x) \leq \rho$, we obtain directly from Equations (1) and (2) and Equations (3) and (4): $E[\tilde{R}(x)]_{LAS} \geq E[R(x)]_{SRPT}$ and $E[\tilde{W}(x)]_{LAS} \geq E[W(x)]_{SRPT}$. Hence the left-hand side inequality. We begin the proof of the right-hand side inequality by studying $E[\tilde{R}(x)]_{LAS} - E[R(x)]_{SRPT}$:

$$\begin{aligned} E[\tilde{R}(x)]_{LAS} - E[R(x)]_{SRPT} &= \frac{x}{1-\phi(x)} - \int_0^x \frac{dt}{1-\rho(t)} \\ &= \int_0^x \frac{dt}{1-\phi(x)} - \int_0^x \frac{dt}{1-\rho(t)} \\ &= \int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)}\frac{1}{1-\rho(t)}dt \end{aligned}$$

Applying the Chebyshev integral inequality [19] to $f(t) = \frac{\phi(x)-\rho(t)}{1-\phi(x)}$, which is a decreasing function of t and $g(t) = \frac{1}{1-\rho(t)}$, which is an increasing function of t, we get $E[\tilde{R}(x)]_{LAS} - E[R(x)]_{SRPT}$

$$\leq \frac{1}{x}\int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)}dt \underbrace{\int_0^x \frac{dt}{1-\rho(t)}}_{E[R(x)]_{SRPT}} \quad (24)$$

Besides:

$$\begin{aligned}
\frac{1}{x}\int_0^x \frac{\phi(x)-\rho(t)}{1-\phi(x)}dt &= \Big([t(\phi(x)-\rho(t)]_0^x + \int_0^x \lambda t^2 f(t)dt\Big) \\
&= \frac{1}{x(1-\phi(x))}(x(\lambda x F^c(x)) \\
&\quad +\lambda m_2(x)) \\
&= \frac{\lambda}{x(1-\phi(x))}(x^2F^c(x)+m_2(x)) \\
&\leq \frac{\lambda}{x(1-\phi(x))}(x^2F^c(x) \\
&\quad +x\int_0^x tf(t)dt) \\
&\quad \text{since} \int_0^x t^2f(t)dt \leq x\int_0^x tf(t)dt \\
&= \frac{x\phi(x)}{x(1-\phi(x))} \\
&= \frac{\phi(x)}{1-\phi(x)} \qquad (25)
\end{aligned}$$

Using Equation (25) in Equation (24), one obtains:

$$\begin{aligned}
E[\tilde{R}(x)]_{LAS} &\leq \Big(1+\frac{\phi(x)}{(1-\phi(x))}\Big)E[R(x)]_{SRPT} \\
&= \frac{1}{1-\phi(x)}E[R(x)]_{SRPT} \qquad (26)
\end{aligned}$$

Consider now $E[\tilde{W}(x)]_{LAS}$ and $E[W(x)]_{SRPT}$, we have:

$$E[\tilde{W}(x)]_{LAS} = \frac{(1-\rho(x))^2}{(1-\phi(x))^2}E[W(x)]_{SRPT} \qquad (27)$$

Adding Equations (26) and (27), we obtain:

$$\begin{aligned}
E[T(x)]_{LAS} &\leq \max\Big(\frac{(1-\rho(x))^2}{(1-\phi(x))^2}, \frac{1}{1-\phi(x)}\Big)E[T(x)]_{SRPT} \\
&= \frac{1}{1-\phi(x)}\max\Big(\frac{(1-\rho(x))^2}{1-\phi(x)}, 1\Big)E[T(x)]_{SRPT}
\end{aligned}$$

Since $(1-\rho(x))^2 = 1-2\rho(x)+\rho(x)^2 = 1-\rho(x)+\underbrace{\rho(x)^2-\rho(x)}_{=\rho(x)(1-\rho(x))\geq 0}$ and $1-\phi(x) = 1-\rho(x)-\underbrace{xF^c(x)}_{\geq 0}$, then $\frac{(1-\rho(x))^2}{1-\phi(x)} \geq 1$ and the right-hand side of the Theorem follows directly from Equation (28). □

Figure 5 illustrates the result of Theorem 4 for the BP with $C = 284$ and the exponential distribution $Exp(3*10^3)$ as a function of percentiles of job sizes to obtain results for a job size distribution that exhibits the high variability property and for a job size distribution that does not exhibit the high variability property. We observe from Figure 5 that the ratio between $E[T(x)]_{LAS}$ and $E[T(x)]_{SRPT}$ highly depends on the variability of the job size distribution. We see that for the bounded Pareto distribution, the ratio $\frac{E[T(x)]_{LAS}}{E[T(x)]_{SRPT}}$ is about 1.25 for all jobs even under very high load, which shows that the conditional mean response times $E[T(x)]$ of jobs under LAS are quite close to ones under SRPT. For BP distributions with lower values of $284 > C \geq 20$, we observed that the value of $\frac{E[T(x)]_{LAS}}{E[T(x)]_{SRPT}}$ is quite close to the value for BP with $C = 284$ for all load values.

Jobs whose size is exponentially distributed experience under LAS a slightly higher higher conditional mean response time

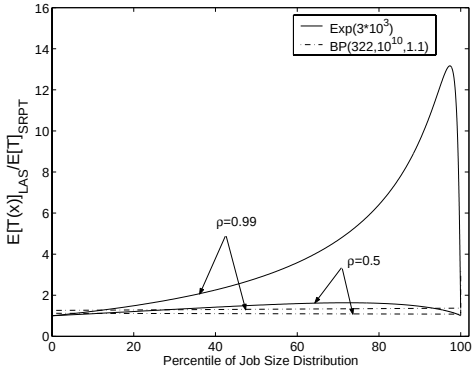


Figure 5: $\frac{E[T(x)]_{LAS}}{E[T(x)]_{SRPT}}$ ***as a function of percentile of job size distribution.***

$E[T(x)]_{LAS}$ than under SRPT for medium load $\rho = 0.5$. However, for very high load $\rho = 0.99$, the difference is much more pronounced and the ratio $\frac{E[T(x)]_{LAS}}{E[T(x)]_{SRPT}}$ approaches 13 for some large jobs. Note also that as $x \to 100th$ percentile job size, the ratio $\frac{E[T(x)]_{LAS}}{E[T(x)]_{SRPT}} \to 1$ because $\lim_{x\to\infty}\phi(x) = \rho(x)$.

We conclude that for job size distributions with the high variability property, LAS scheduling is an attractive policy to be used, for instance, in Web-servers handling requests for both, static and *dynamic* pages. As opposed to SRPT , LAS can handle requests for dynamic pages since it does not need to know the job size and yet offers a response time performance very similar to SRPT.

4. LAS UNDER OVERLOAD

In real systems, it may happen that jobs arrive to the system at a higher rate than the rate at which they are serviced. This situation is referred to as overload condition where $\rho \geq 1$. To the best of our knowledge, no analysis of LAS under overload has been conducted. Here, we show that at load $\rho \geq 1$ LAS is stable for all small jobs up to a certain job size and we derive the formulas of the conditional mean response time for LAS under overload. SRPT was also proven to be stable under overload in [2] for a range of short jobs. Hence in this section, we also compare LAS to SRPT under overload.

Theorem 5. *Let $\theta(x) \triangleq m_1(x) + xF^c(x)$, λ be the job mean arrival rate, and $f(t)$ be a service time distribution with mean m_1. Then, for any load $\rho = \lambda m_1 \geq 1$, every job size $x < x_{LAS}(\lambda)$ with $x_{LAS}(\lambda) \triangleq \max\{x \mid \theta(x) \leq \frac{1}{\lambda}\}$ has a finite conditional mean response time under LAS, whereas every job of size $x \geq x_{LAS}(\lambda)$ experiences an infinite response time.*

PROOF. The load offered to the server when LAS system is under overload is $\rho = \lambda m_1 \geq 1$. However, the effective load $\rho_{\text{effective}}$ that corresponds to the work serviced by the server is equal to 1. Let $\theta_{\text{effective}} \triangleq \frac{\rho_{\text{effective}}}{\lambda} = \frac{1}{\lambda}$. Then, $\theta_{\text{effective}}$ is the expected service offered to the set of jobs that access the server under overload. This set depends on the policy. With LAS, every newly arriving job in the system gets an immediate access to the server. On the average a job receives a service of $\theta_{\text{effective}}$. But, since some jobs require less than $\theta_{\text{effective}}$, the jobs of size strictly smaller than $x_{\text{LAS}}(\lambda)$ may receive more service than $\theta_{\text{effective}}$, where $x_{\text{LAS}}(\lambda)$ defined as:

$$x_{\text{LAS}}(\lambda) \triangleq \max\{x \mid \theta(x) \leq \frac{1}{\lambda}\} \qquad (28)$$

Hence, for LAS under overload, one obtains $\rho_{\text{effective}} = \lambda\theta(x_{\text{LAS}}) = 1$. □

Corollary 2. *For SRPT under overload, every job of size $x < x_{SRPT}(\lambda)$ with $x_{SRPT}(\lambda) \triangleq \max\{x \mid m_1(x) \leq \frac{1}{\lambda}\}$ has a finite conditional mean response time, whereas every job of size $x \geq x_{SRPT}(\lambda)$ experiences an infinite response time*

PROOF. The reasoning for SRPT is different from the one for LAS. Note that the service of a job under SRPT is not affected by the arrival of jobs with larger sizes. Hence, a set of jobs with size less than or equal to x contributes a system load of $\rho(x) = \lambda \int_0^x tf(t)dt$. Thus under overload, only jobs with size strictly less than $x_{\text{SRPT}}(\lambda)$ receive service, where $x_{\text{SRPT}}(\lambda) = \max\{x \mid \rho(x) \leq \rho_{\text{effective}} = 1\}$, which is equivalent to $x_{\text{SRPT}}(\lambda) = \max\{x \mid m_1(x) \leq \frac{1}{\lambda}\}$. The jobs with size greater than or equal to $x_{\text{SRPT}}(\lambda)$ can not access the server since the jobs with size less than $x_{\text{SRPT}}(\lambda)$ always preempt them to maintain $\rho_{\text{effective}} = 1$. □

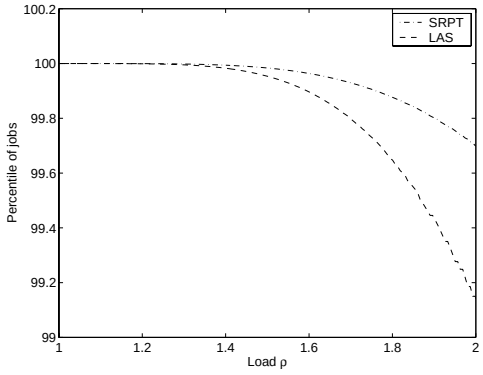


Figure 6: ***Percentile of the largest job sizes that can be serviced under overload for $BP(332, 10^{10}, 1.1)$ job size distribution.***

Note that $x_{\text{SRPT}}(\lambda) \geq x_{\text{LAS}}(\lambda)$ since with SRPT under overload, a job accesses the server only if its service can be entirely completed, which may not the case with LAS.

We observe in Figure 6 that for $BP(332, 10^{10}, 1.1)$, the maximum job size that SRPT can service under overload is always larger than or equal to the maximum job size that LAS can service at a given load. It is worth mentioning that for the overload values considered, the maximum job that is serviced by both policies is above the 99th percentile of the job size distribution. This result is to be expected since for the job size distribution considered, more than 99% of the jobs are small jobs and the fraction of load that these small jobs contribute to the system load is less than 50% of the total load.

We now compute the formulas for the conditional mean response time of the jobs under overload. For the SRPT policy, the formulas in case of underload and case of overload are different [2]. This may be explained by the facts that all the jobs of size $x < x_{\text{SRPT}}(\lambda)$ with $\rho(x_{\text{SRPT}}) \triangleq \int_0^{x_{\text{SRPT}}(\lambda)} tf(t)dt = 1$ have a finite response time, and all the jobs of size $x \geq x_{\text{SRPT}}(\lambda)$ receive no service at all. Therefore, under overload, the SRPT system works as if there are no jobs of size $x \geq x_{\text{SRPT}}(\lambda)$. Hence, the moment[2] $m_2(x) + x^2F^c(x)$ that appears in the formulas of $E[T(x)]_{SRPT}$ is computed only for the jobs of size $x < x_{\text{SRPT}}(\lambda)$, whereas in underload, $E[T(x)]_{SRPT}$ is computed considering all job sizes.

[2] $m_2(x) + x^2F^c(x)$ and $m_1(x) + xF^c(x)$ are the moments of a truncated distribution $f_x(y)$ (with $f_x(y) = f(y)$ if $y < x$, $f_x(y) = F^c(x)$ if $y = x$, and $f_x(y) = 0$ if $y > x$) that account for the contribution of all jobs of size y to the response time of the job of size x (see ([17], pp. 173).

For LAS under overload, all jobs can receive up to $x_{\overline{\text{LAS}}}(\lambda) \triangleq \max\{x \mid \theta(x) < \frac{1}{\lambda}\}$. Thus, if the initial service requirement of a job is larger than $x_{\overline{\text{LAS}}}(\lambda)$, it receives only $x_{\overline{\text{LAS}}}(\lambda)$). Therefore, the moments[2] $m_2(x) + x^2F^c(x)$ and $m_1(x) + xF^c(x)$ that appear in the formulas for $E[T(x)]_{LAS}$ must be computed considering all job sizes. As a consequence, the formulas for the conditional mean response time in overload and underload are identical:

Theorem 6. *The conditional mean response time of a job size x for LAS under overload is:,*

$$E[T(x)]_{LAS} = \begin{cases} \frac{\lambda(m_2(x)+x^2(1-F(x)))}{2(1-\rho(x)-\lambda x(1-F(x)))^2} + \frac{x}{1-\rho(x)-\lambda x(1-F(x))} & \text{if } x < x_{LAS}(\lambda) \\ +\infty & \text{if } x \geq x_{LAS}(\lambda) \end{cases}$$

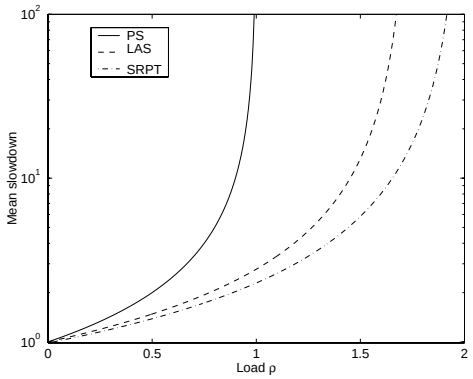


Figure 7: ***Mean slowdown for the 99th percentile job for $BP(322, 10^{10}, 1.1)$ as a function of load.***

Figure 7 shows the mean slowdown of the 99th percentile job of the $BP(332, 10^{10}, 1.1)$ job size distribution under PS, LAS, and SRPT under overload. At load $\rho = 1.4$, the mean slowdown of the job under PS is infinity, whereas it is 10 and 4 under LAS and under SRPT respectively. It is obvious that LAS becomes unstable at lower load values than does SRPT. However, we see that LAS is quite close to the optimal policy SRPT in terms of the mean slowdown of a job even under overload. As seen in the Figure, LAS is unstable under overload for the job size considered. In general, it is well known that the mean response times of all jobs under PS $E[T(x)]_{PS}$ is undefined for load $\rho \geq 1$.

5. EVALUATION OF LAS IN A PACKET NETWORK

We analyzed the M/G/1/LAS queue using a notion of a job. In a packet switched network the entity that arrives at the router at once is a packet. However, we are not interested in the performance (response time and slowdown) of a packet but the performance of a *flow*. The results obtained for jobs cannot be directly applied to flows since a job is an entity that arrives to the system at once, whereas a flow consists of a sequence of packets that are spaced in time and are statistically multiplexed with packets from other flows. To investigate the performance of LAS in packet networks we implement LAS using the ns-2 simulator. We compare the mean response time of flows with varying sizes under LAS to their mean response time under FIFO scheduling with drop-tail buffer management policy, which drops newly arriving packets that arrive when the buffer is full.

LAS is implemented such that the packet that will be served (transmitted) next belongs to the flow that has received the least amount of service. If two or more flows have received an equal amount of service, they share the system resources fairly. The implementation of LAS involves maintaining a single priority queue and inserting each incoming packet at its appropriate position in that queue. The less service a flow has received so far, the closer to the head of the queue its arriving packets will be inserted. When a packet arrives and the queue is full, LAS first "inserts" the newly arriving packet at its appropriate position in the queue and then drops the packet that is at the end of the queue. Hence, LAS guarantees service priority to short flows even during congestion and will drop packets from the largest flows that currently have a packet in the queue.

The idea of giving preferential treatment to short flows has been considered by other researchers [3, 13], who propose DiffServ like models where *edge* routers mark packets as belonging to short or long flows and *core* routers utilize the marking to give preferential treatment to short flows. The first proposal is referred to as PS-w [13]. In PS-w, an edge router maintains a counter for every active flow and assigns a high priority to an incoming packet if the counter value is less than the predefined threshold otherwise a packet is assigned a low priority. In the core routers, low priority packets are serviced only if no high priority packets are backlogged. During congestion, low priority packets are more likely to be dropped than high priority ones. The simulation results in [13] show that with appropriate tuning of the drop rate of high priority and low priority packets, PS-w can reduce the response time of medium-sized flows that have 50-200 packets by up to 80% compared to *random early detection* (RED) [11] queue management policy without significantly penalizing the large flows. The response times of flows of sizes less than 20 packets are similar for PS-w and RED.

In [3], a short flow differentiating (SFD) algorithm is proposed and implemented as DiffServ policy in the ns-2 simulator. Similarly to PS-w, edge routers in SFD mark packets as IN or OUT based on the amount of data a flow has already sent. In the core routers, SFD applies the RED buffer management discipline to handle IN and OUT packets by maintaining a single FIFO queue with two virtual RED queues or two separate RED queues for IN and OUT packets with preferential dropping to OUT packets. For the case of physical RED queues, strict priority packet scheduler is used. The simulation results [3] show that SFD reduces the response time of about 90% of the shortest flows on average by about 30% as compared to RED and only 1% of the largest flows see higher response time under SFD.

Both studies show that it is feasible to implement LAS-like policies in the core routers and achieve scalability by identifying (marking) flows only at the edge of the network. In the next section, we present simulation results that compare LAS to FIFO. The results show that LAS offers much higher performance improvement for short flows, in terms of reducing their mean reponse times, than PS-w or SFD.

5.1 Simulation Results

We simulate LAS using the network topology shown in Figure 8 where C1-C5 are clients initiating a series of Web sessions, each retrieving some Web pages from a server randomly chosen from a pool of S1-S5 as proposed in the ns-2 Web model in [9]. Each Web page contains a certain number of objects. Each time an object is requested, a new TCP connection is established. We refer to the packets that belong to one TCP connection as a flow. We set the Web parameters as shown in Table 1. These values are based on the findings of [27] and summarized by [3]. The density function of the Pareto distribution is obtained from Equation (7) when $p \to \infty$. Here, we use the Pareto distribution with mean of 12, $\alpha = 1.2$, and $k = 1$. The coefficient of variation of this Pareto distribution is infinite, since $\alpha < 2$. We use fixed size data packets of 1400 bytes in simulations. To obtain different load conditions we adjust the number of sessions and the session interarrival times as shown in Table 1.

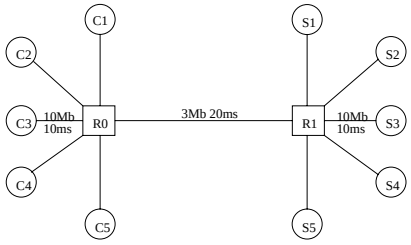


Figure 8: ***Simulated network topology.***

We study three load conditions: load $\rho = 0.55, 0.73$, and 0.97. We run each simulation for 36000 seconds and start collecting performance statistics after an initial period of 6000 seconds (to avoid transient effects). The results are obtained by first averaging the response times for all objects oa a given size, then further averaging the mean response times of objects of sizes greater than 30 with linearly increasing bins starting with a bin of size 1.

The simulation results are shown in Figures 9 and 10. For all three load values, at least 99.5% of the shortest Web flows benefit from LAS as compared to FIFO. We see that for all loads, the mean response time of short flows is smaller under LAS than under FIFO. For heavy load $\rho = 0.97$ there is a difference of a factor of 40 for 99% of the shortest flows while 0.001% of the largest flows have higher mean response time under LAS than under FIFO by a factor of about 3. These results agree with the general numerical and analytical results observed previously. For lower load values, the difference in mean response time between LAS and FIFO for short objects decreases and the mean response time for the largest objects is similar under both policies.

The reduction in response time for short flows under LAS is due to two factors. (i) The packets of short flows are always inserted close to the head of the queue, which reduces their waiting time as compared to FIFO. (ii) Under high load, there will be congestion resulting in buffer overflow: For FIFO with drop-tail, both, short and long flows will experience loss. For LAS, however, it is only the long flows that suffer loss. These two effects explain why there is such a large difference in response time for short flows between LAS and FIFO. For LAS at high load, the response time for very long flows is higher than under FIFO since under LAS, as long flows who have received a certain amount of service will get service only when there is no packet in the queue from a shorter flow.

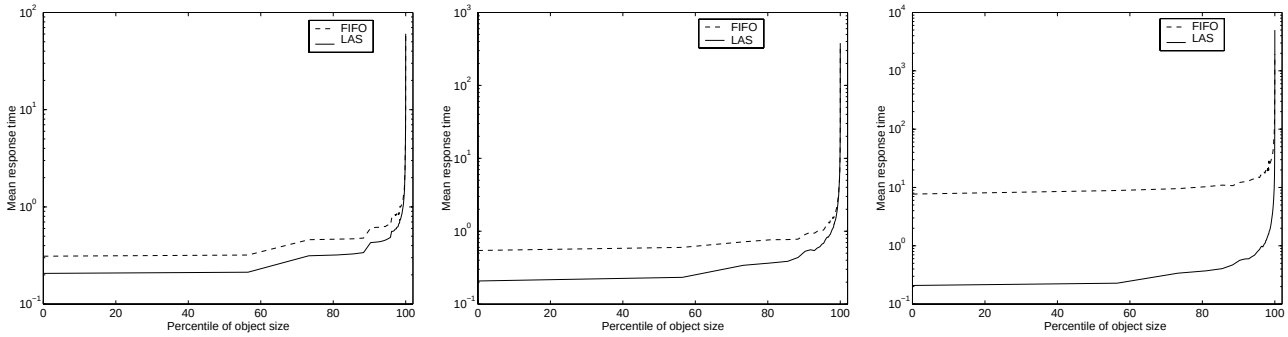
	Loss Rate		$\frac{E[T(x)]_{FIFO}}{E[T(x)]_{LAS}}$
Load ρ	FIFO	LAS	for the 99% shortest flows
0.55	1.2%	0.9%	1.5
0.74	4.2%	3.66%	2.75
0.97	20.7%	4.9%	40

Table 2: ***Loss and response time performance for different loads.***

In summary, we can conclude that LAS in a packet network improves the response time for short flows without penalizing much the long flows. Another advantage of LAS over FIFO is that it reduces the loss rate for a given load and assures that short flows will

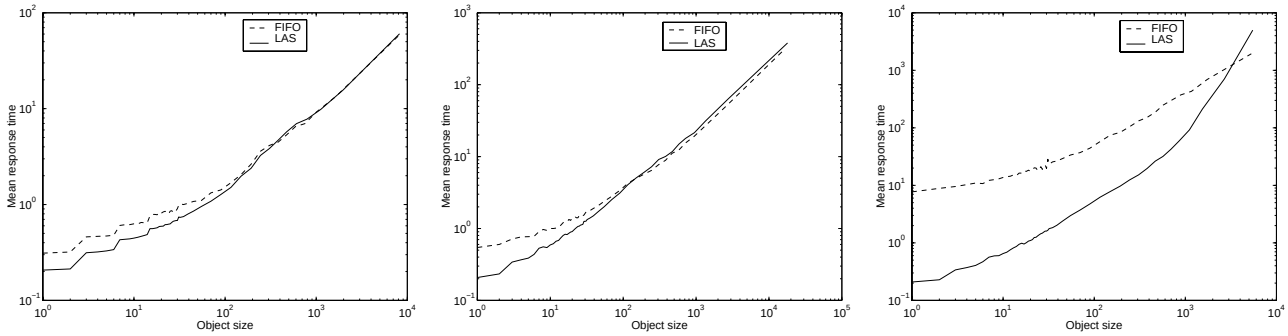
Web model elements	Element attributes	Distribution	Parameters
Web session	Number of sessions	Constant	$\{3400, 3600, 4000\}$
	Interval between sessions (sec)	Exponential	mean: $\{20, 15, 10\}$
	Number of web pages per session	Exponential	mean:100
Web page	Interval between pages (sec)	Exponential	mean: 10
	Number of web objects per page	Exponential	mean: 3
Web object	Interval between objects (sec)	Exponential	mean : 0.001
	Object size (packets)	Pareto	mean : 12, shape: 1.2

Table 1: ***Web Traffic Attributes and Corresponding Distributions.***



(a) Load $\rho = 0.55$ (b) Load $\rho = 0.73$ (c) Load $\rho = 0.97$

Figure 9: ***Mean response time as a function of percentile of object size (in packets).***



(a) Load $\rho = 0.55$ (b) Load $\rho = 0.73$ (c) Load $\rho = 0.97$

Figure 10: ***Mean response time as a function of object size (in packets).***

see no or very little loss. The difference in loss rate between LAS and FIFO increases with increasing load as seen in Table 2. The results obtained indicate that only the *very few largest flows* suffer a penalty under LAS, while all the other flows see their response time reduced. For this reason, we expect that even under HTTP/1.1 with persistent connections, the size of most of these connections in terms of the number of packets will see their response time reduced under LAS. Table 1 shows that the mean size of a Web page is 36 packets. Hence, even if all data on the page is sent in one connection under HTTP/1.1, our simulation results indicate that on average the connection still benefits from LAS as compared with FIFO. Therefore, only those persistent connections that contain a request for a very large object will see an increase in their response time.

6. CONCLUSION

We analyzed the $M/G/1/LAS$ queuing model to evaluate the performance of the LAS for job size distributions with different degrees of variability. We showed through analysis and numerical evaluation that the variability of a job size distribution is important in determining the performance of LAS in terms of response time and slowdown. In particular, we saw that the percentage of jobs that have higher slowdowns under LAS than under PS is smaller for job size distributions with the high variability property than for job size distributions with values of C close to 1. For the case of the exponential job size distribution, we proved that the mean slowdown for LAS is always less than or equal to the mean slowdown for PS. Even for the case of general job size distribution, we

showed that for moderate load values, the mean slowdown of LAS remains fairly close to the mean slowdown of PS.

The camparison of LAS to nonpreemptive policies (NPP) reveals that the benefits of LAS over NPP in terms of low mean response time improves with increasing variance of job size distribution. Similarly, we compared LAS to the optimal policy SRPT and demonstrated that $E[T(x)]_{LAS}$ is closer to the $E[T(x)]_{SRPT}$ for job size distributions with high variability property.

We also proved that LAS is stable under overload for a subset of small jobs and obtained the expression for the conditional mean response time $E[T(x)]$ for LAS under overload. The ability of LAS scheduling to schedule jobs under overload does not apply for PS and FIFO scheduling.

We used simulation to evaluate the benefit of LAS over FIFO for scheduling the transmission of packets over a bottleneck link and saw that most flows experienced a reduction of their response time that can be very significant under high load. These results illustrate the potential benefit of flow size aware scheduling policies such as LAS in routers. However, more work needs to be done to come up with a complete architecture for flow size aware scheduling in the Internet that specifies the functionalities required in the edge routers and the core routers.

Acknowledgement

We would like to thank Ken Sevcik and the anonymous Sigmetrics reviewers of this paper for their constructive comments.

7. REFERENCES

[1] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW", In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, pp. 92–103, 1996.

[2] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness", In *Sigmetrics 2001 / Performance 2001*, pp. 279–290, June 2001.

[3] X. Chen and J. Heidemann, "Preferential Treatment for Short Flows to Reduce Web Latency", *To appear in Computer Networks*, 2003.

[4] E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall Inc., 1973.

[5] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley Publishing Company, 1967.

[6] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, pp. 835–846, December 1997.

[7] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection Scheduling in Web Servers", In *USENIX Symposium on Internet Technologies and Systems (USITS '99)*, pp. 243–254, Boulder, Colorodo, October 1999.

[8] M. E. Crovella, "Performance Evaluation with Heavy Tailed Distributions", In *Job Scheduling Strategies for Parallel Processing 2001 (JSSPP)*, pp. 1–10, 2001.

[9] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control", In *Proc. of the ACM/SIGCOMM'99*, August 1999.

[10] M. J. Fischer, D. Gross, D. M. B. Masi, and J. F. Shortle, "Analyzing the Waiting Time Process in Internet Queueing Systems With the Transform Approximation Method", *The Telecommunications Review*, pp. 21–32, 2001.

[11] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[12] W. Gong, Y. Liu, V. Misra, and D. Towsley, "On the Tails of Web File Size Distributions", In *Proc. of 39-th Allerton Conference on Communication, Control, and Computing*, October 2001.

[13] L. Guo and I. Matta, "Differentiated Control of Web Traffic: A Numerical Analysis", In *SPIE ITCOM'2002: Scalability and Traffic Control in IP Networks*, August 2002.

[14] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based Scheduling to Improve Web Performance", *ACM Transaction on Computer Systems*, (May), 2003.

[15] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic Convergence of Scheduling Policies with respect to Slowdown", *Performance Evaluation*, 49:241–256, September 2002.

[16] http://www.isi.edu/nsnam/ns/, "The Network Simulator ns2",

[17] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*, Wiley, New York, 1976.

[18] H. Krayl, E. J. Neuhold, and C. Unger, *Grundlagen der Betriebssysteme*, Walter de Gruyter, Berlin, New York, 1975.

[19] D. S. Mitrinovic, *Analytic Inequalities*, Springer-Verlag, 1970.

[20] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. E. Gehrke, "Online Scheduling to Minimize Average Stretch", In *40th Annual Symposium on Foundation of Computer Science*, pp. 433–442, 1999.

[21] V. Paxon and S. Floyd, "Wide-Area Traffic: The failure of Poisson Modelling", *IEEE/ACM Transactions on Networking*, 3:226–244, June 1995.

[22] R. Schassberger, "A Detailed Steady State Analysis of the M/G/1 Queue under various Time Sharing Disciplines", In P. J. C. In O.J. Boxma G. Iazeolla, editor, *Computer Performance and Reliability*, pp. 431–442, North Holland, 1987.

[23] R. Schassberger, "The Steady State Distribution of Spent Service Times Present in the M/G/1 Foreground-Background Processor-sharing Queue", *Journal of Applied Probability*, 25(7):194–203, 1988.

[24] L. E. Schrage, "The queue M/G/1 with feedback to lower priority queues", *Management Science*, 13(7):466–474, 1967.

[25] L. E. Schrage, "A Proof of the Optimality of the Shortest Remaining Service Time Discipline", *Operations Research*, 16:670–690, 1968.

[26] L. E. Schrage and L. W. Miller, "The queue M/G/1 with the shortest processing remaining time discipline", *Operations Research*, 14:670–684, 1966.

[27] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP protocol headers can tell us about the web", In *Proc. ACM SIGMETRICS*, pp. 245–256, June 2001.

[28] R. W. Wolff, "Time Sharing with Priorities", *SIAM Journal of Applied Mathematics*, 19(3):566–574, 1970.

[29] S. Yang and G. Veciana, "Size-based Adaptive bandwidth Allocation: Optimizing the Average QoS for Elastic Flows", In *Infocom 2002*, 2002.

Traffic Engineering in a Multipoint-to-Point Network

Guillaume Urvoy-Keller, Gérard Hébuterne, and Yves Dallery

***Abstract*—The need to guarantee quality-of-service (QoS) to multimedia applications leads to a tight integration between the routing and forwarding functions in the Internet. multiprotocol label switching tries to provide a global solution for this integration. In this context, multipoint-to-point (m2p) networks appear as a key architecture since they provide a cheaper way to connect edge nodes than point-to-point connections. M2p networks have been mainly studied for their load balancing ability. In this paper, we go a step further: we propose and evaluate a traffic management scheme that provides deterministic QoS guarantees for multimedia sources in an m2p network. We first derive an accurate upper bound on the end-to-end delay in an m2p architecture based on the concept of additivity. Broadly speaking, an m2p network is additive if the maximum end-to-end delay is equal to the sum of local maximum delays. We then introduce two admission control algorithms for additive networks: a centralized algorithm and a distributed algorithm and discuss their complexity and their scalability.**

***Index Terms*—Admission control, deterministic bounds, multiprotocol label switching, multipoint-to-point networks, quality-of-service.**

I. Introduction

PROVISIONING OF quality-of-service (QoS) in high-speed networks has received much attention in the last decade. The asynchronous transfer mode (ATM) community advocated for a connection oriented solution while the Internet community advocated for a connectionless solution. Today, there is a trend to combine these solutions since the backplane of many core routers is an ATM switch fabric. ATM switches provide an high-speed and low cost per port solution for the Internet. However, they are not universally used. Multiprotocol label switching (MPLS) [1], has been developed to offer a universal forwarding layer to the Internet. MPLS may interoperate adequately with ATM [2] or frame relay [3] or provide an ad-hoc forwarding service.

An Internet service provider (ISP) may use MPLS to establish a set of routes between its ingress nodes and its egress nodes. If point-to-point (p2p) routes are used and there are n edges, then $O(n^2)$ routes are required to connect n nodes. Another possibility to cover the network is to use multipoint-to-point (m2p) connections rooted at the egress nodes. With m2p connections, only $O(n)$ routes are required. The use of m2p label switch paths (LSPs) allows to merge several p2p LSPs: m2p LSP ease traffic management since they reduce the amount of states (corresponding to LSPs) at each node, i.e., not only at edge nodes but also at interior nodes (see Fig. 1). In this paper, we propose and evaluate a traffic management scheme for an m2p network that guarantees a deterministic QoS to variable bit-rate sources. Sources are assumed to be leaky bucket constrained with a maximal end-to-end delay requirement. We assume a fluid model that closely approximates the behavior of a packet network with a small packet size compared with the service rates of the servers. The fluid model enables us to concentrate on the central issues. The proposed scheme is based on the first-in first-out (FIFO) scheduling policy, because of its scalability. It is quite likely that more complex policies such as PGPS [4], [5] will be used in the future. However, these policies will not run at a connection level but rather at a class level (to ensure scalability) and a given class will see the m2p network as a FIFO network (with a time-varying service rate). Thus, a first problem to solve is the case of a FIFO m2p network. To our best knowledge, this problem has not been treated previously.

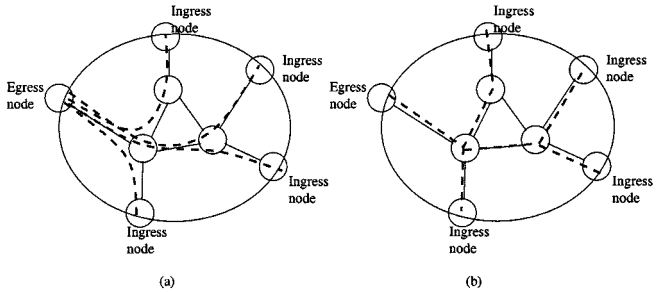


Fig. 1. (a) P2p strategy. (b) M2p strategy.

The remainder of the paper is organized as follows. In Section II, we review the related work in the fields of m2p architectures and end-to-end bounds in FIFO networks. In Section III, we recall the main results of the Network Calculus [6]–[8] that we use to obtain a bound on the end-to-end delay. In Section IV, we emphasize the difficulty to directly derive an end-to-end delay bound for a FIFO network from the concept of service curve introduced in the network calculus. In Section V, we study the maximum end-to-end delay in the case of an m2p network with two servers. The analysis demonstrates that a bounding approach is required for the case of larger networks. We also introduce the concept of *additivity*, which is central to our analysis in the case of larger networks carried in Section VI. In Section VII, we propose and evaluate two admission control algorithms based on our end-to-end delay bound. In Section VIII, we conclude and provide some insights for future work.

Manuscript received February 21, 2001; revised November 23, 2001.

G. Urvoy-Keller is with the Institut Eurecom 06904 Sophia-Antipolis, France (e-mail: urvoy@eurecom.fr).

G. Hébuterne is with the Institut National des Télécommunications, 91011 Evry cedex, France (e-mail: hebutern@hugo.int-evry.fr).

Y. Dallery is with the Ecole Centrale de Paris, 92 295 Chatenay Malabry cedex, France (e-mail: dallery@pl.ecp.fr).

Publisher Item Identifier S 0733-8716(02)03062-7.

II. Related Work

A first step toward the provision of QoS service is the ability to balance load in the network. In the traditional Internet, this is achieved with metric-based routing. Network administrators adjust link metrics to balance the traffic. However, this ad-hoc solution is not satisfying in the context of QoS provisioning and there exists a need to explicitly control the routes of the flows in the network. Such a control may be achieved with MPLS. Such a solution has been investigated in [9] in the context of Internet protocol (IP) over ATM and in [10] in the context of IP over MPLS. In [10], Saito *et al.* propose a traffic engineering scheme for Internet backbones that tries to provide an optimal load balancing for reliability. The proposed scheme uses multiple m2p LSPs between each ingress/egress pair to achieve load balancing and reliability. Traffic demands are expressed as service rates. Sources are, thus, implicitly assumed to be constant bit rate sources.

A first step toward the design of our traffic management scheme is the derivation of an accurate bound on the end-to-end delay for an m2p network. Determining an end-to-end delay bound in a network based on the FIFO scheduling discipline is a challenge since the stability of a FIFO network with a general architecture has not been established yet. Tassiulas *et al.* [11] proved that the ring architecture is stable under any work-conserving scheduling discipline (and thus under the FIFO scheduling discipline). The result is interesting since the ring architecture is often considered as a "worstcase" architecture due to the high dependency it induces among sessions. However, this result has not yet been extended yet to the case of a general architecture. Chlamtac *et al.* [12] have focused on FIFO networks with peak rate constrained sources. The authors show that if the peak rate of each source in the network satisfies a constraint related to the number of sources that the source meets on its route, then the network is stable and bounds on end-to-end delays and backlogs exist. The result applies to FIFO networks with a general architecture, but it is restricted to the case of constant bit rate sources. In the present work, we concentrate on a specific architecture, the m2p architecture, however, with variable bit rate sources.

III. Network Calculus

The network calculus [6]–[8], [13], [14] is an analytical method to derive deterministic bounds on end-to-end delays and backlogs. The network calculus has been developed both for continuous time [13] and discrete time [8]. We use here the continuous version that is better suited for a fluid-flow analysis. We present, in the following, the basic concepts of the network calculus that will be used in the rest of this paper.

A. Sources and Network Elements Modeling

1) General Sources: Consider a source $\mathbb{S}$. Let S be a given trajectory of $\mathbb{S}$ and $\Gamma_{\mathbb{S}}$ be the set of all possible trajectories for $\mathbb{S}$.

Definition 1: The cumulative rate function $A_S(t)$ of S is defined as the cumulative amount of bits issued by S in the interval $[0, t]$ (the cumulative rate function of a given trajectory fully characterizes this trajectory).

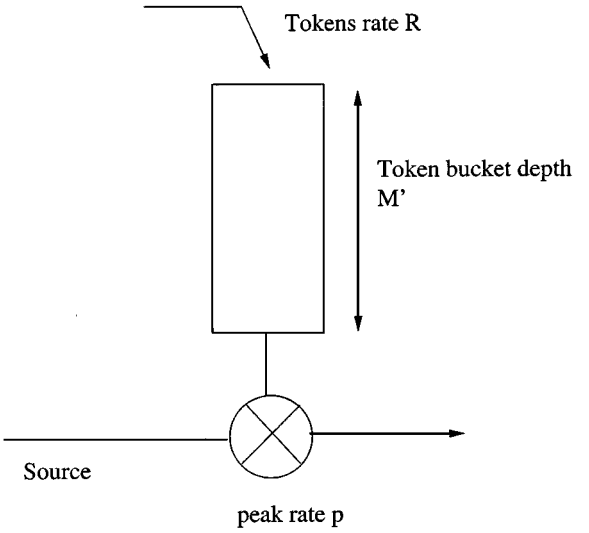


Fig. 2. Leaky bucket controller.

Definition 2: A function α is an arrival curve for $\mathbb{S}$ if

$$A_S(t+\tau) - A_S(t) \leq \alpha(\tau), \forall S \in \Gamma_{\mathbb{S}}, \forall \tau \geq 0, \forall t \geq 0.$$

An arrival curve for $\mathbb{S}$ provides an upper bound on the number of bits that $\mathbb{S}$ can send on any time interval.

Definition 3: We define $\Xi_{\mathbb{S}}$ as the set of arrival curves associated to $\mathbb{S}$

$$\Xi_{\mathbb{S}} = \{\alpha | \forall S \in \Gamma_{\mathbb{S}}, \forall (t,\tau), A_S(t+\tau) - A_S(t) \leq \alpha(\tau)\}.$$

Theorem 1: (See [13] for proof) $\Xi_{\mathbb{S}}$ as a minimum element α^*, called the minimum arrival curve and defined as follows:

$$\alpha^*(\tau) = \max_{S \in \Gamma_{\mathbb{S}}} \max_t (A_S(t+\tau) - A_S(t)), \forall \tau \geq 0.$$

In the remaining of this paper and for sake of simplicity, the term "source" may be used to refer to a trajectory.

2) Source Model: In the remaining of this paper, we consider sources that are leaky bucket constrained with an additional constraint on their peak rate. A traffic descriptor for a given source S has three parameters (p, R, M) (we note $S \sim (p, R, M)$) that are, respectively, the peak rate, the mean rate, and the maximum burst size of S. Such a source is able to traverse the leaky bucket controller depicted in Fig. 2 without experiencing any loss. The size of the token bucket is $M' = M(p/(p-R)) > M$ since the peak rate of the source is finite. Let $\Omega(p, R, M)$ be the set of sources S such as $S \sim (p, R, M)$. The greedy source (trajectory), associated to $\Omega(p, R, M)$, plays an important role in worst case analyses and is defined as follows.

Definition 4: For a given set $\Omega(p, R, M)$, $G_{\Omega(p,R,M)}$ (or simply G) is the source that consumes tokens as soon as possible. With a token bucket initially full at time $t = 0$, the greedy source G emits at its peak rate during $[0, M/p]$ and then emits at its mean rate R, i.e.,

$$A_G(\tau) = \min\left(p \cdot \tau, R \cdot \tau + M\frac{p-R}{p}\right).$$

The following results hold for the greedy source (proof is left to reader):

- $\alpha_G^*(\tau) = A_G(\tau), \forall \tau \geq 0$ (the minimum arrival curve of the greedy source is its cumulative rate function);
- $\forall S \in \Omega(p, R, M) \forall \tau \geq 0, \alpha_S^*(\tau) \leq \alpha_G^*(\tau)$ (the minimum arrival curve of the greedy source is the minimum arrival curve of all the sources of $\Omega(p, R, M)$).

The minimum arrival curve of a multiplex of leaky bucket constrained sources is the sum of the minimum arrival curves of the sources of this multiplex (see [14]). The resulting source has a concave piecewise linear arrival curve. We make use of the two notions (source constrained by a single leaky bucket or by a set of leaky buckets) in the remaining of this paper.

The source model presented above encompasses the case of an IP source declared with a TSPEC and the case of a variable bit rate (VBR) ATM source. An ATM VBR source is constrained by a pair of generic cell rate algorithms (GCRAs) algorithms with parameters (T, τ) and $(T', \tau' + \tau)$. Let δ be the cell size in bits. A minimal arrival curve α for a VBR source is (see [15])

$$\alpha(t) = \min\left(p \cdot t + b_p,\ R \cdot t + b_M\right)$$

where $p = \delta/T$ is the peak rate of the source in bit/s, $R = \delta/T'$ is the sustainable rate in bit/s, $b_p = p \cdot \tau + \delta$ corresponds to the cell jitter (in bits) and $b_M = \tau' R$ corresponds to the maximum burst size of the source (in bits).

Similarly, an IP source described with a TSpec has a minimal arrival curve $\alpha(t) = \min(M + p \cdot t, b + r \cdot t)$ where M is the maximum size of a packet of the source, p its peak rate, r its sustainable rate and b its bucket depth.

3) Network Elements: Within the network calculus framework, a network element is characterized by its service curve that intuitively represents a lower bound on the service it provides. Several definitions of a service curve exist. We use the extended service curve defined by Le Boudec [13].

Definition 5: A network element offers an extended service curve β to a given source $\mathbb{S}$ if

$$\forall S \in \Gamma_{\mathbb{S}}, \forall t \geq 0, \exists t_0 \leq t : A_{S_{\text{out}}}(t) - A_S(t_0) \geq \beta(t - t_0)$$

where $A_{S_{\text{out}}}$ is the cumulative rate function of S seen at the output of the network element.

Example: A service curve β for a work-conserving server with a service rate C is $\beta(\tau) = C \cdot \tau$. The proof is straightforward: t_0 is equal to the beginning of the busy period that t belongs to, or $t_0 = t$ if there is no backlog at time t.

B. Advanced Results

1) Bounds on Delays and Backlogs: Consider a system, seen as a black box. Let $R(t)$ (with respect to $R^*(t)$) be the cumulative rate function seen at the input (with respect to output) of the system. The backlog at time t is $b(t) = R^*(t) - R(t)$. Cruz [8] has introduced the virtual delay $d(t)$ defined as follows:

$$d(t) = \inf\left\{T : T \geq 0,\ R^*(t+T) \geq R(t)\right\}.$$

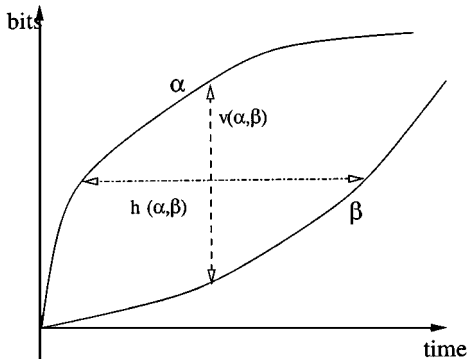


Fig. 3. Upper bounds on backlogs and delays.

Theorem 2: Consider a source with an arrival curve α traversing a system that offers a service curve β. Then

$$b(t) \leq v(\alpha, \beta) \text{ and } d(t) \leq h(\alpha, \beta)$$

where $v(\alpha, \beta)$ and $h(\alpha, \beta)$ represent, respectively, the maximum vertical and horizontal distances between α and β (see Fig. 3)

$$v(\alpha, \beta) = \sup_{s \geq 0}(\alpha(s) - \beta(s))$$
$$h(\alpha, \beta) = \sup_{s \geq 0}\left(\inf\left\{T : T \geq 0,\ \alpha(s) \leq \beta(s+T)\right\}\right).$$

2) Output Characterization:

Theorem 3 : (See [13] for proof.) An arrival curve α^{out} for a source seen at the output of a system that offers a service curve β is

$$\alpha^{\text{out}}(\tau) = \sup_{v \geq 0}(\alpha(\tau + v) - \beta(v)), \forall \tau \geq 0$$

where α is the arrival curve of the source seen at the input of the system.

3) Network Service Curve: A straightforward way to obtain end-to-end delay bounds is to apply Theorems 2 and 3 at each stage in the network and sum the obtained local bounds. It is, however, possible to obtain a tighter result with the network service curve paradigm.

Theorem 4: (See [13] for proof.) Consider a source S traversing p network elements. Each network element is characterized by an extended service curve $(\beta_j)_{j \in \{1,\ldots,p\}}$. S may see these p network elements as a single network element characterized by a network service curve β that is the convolution of $(\beta_j)_{j \in \{1,\ldots,p\}}$

$$\beta(t) = \inf_{t_1 + \cdots + t_p = t}\left(\beta_1(t_1) + \cdots + \beta_n(t_n)\right).$$

The strength of Theorem 4 is that the obtained end-to-end delay bound is smaller than the one obtained through summation of local delay bounds (using Theorems 2 and 3).

IV. End-to-End Delays: A Service Curve Approach

To define a complete traffic management scheme for m2p networks, we first need to derive an accurate bound on the end-to-end delay. As seen in the previous section, the network calculus provides a way to derive end-to-end bounds using network service curves. In the present section, we investigate this approach.

A. Service Curve Offered by a FIFO Server

Consider two sources, S_1 and S_2 (with respective arrival curves α_1 and α_2) and a server that implements the FIFO scheduling policy with a service rate C. Let λ_C be the function such that: $\forall t \geq 0,\ \lambda_C(t) = C \cdot t$ and R_i (with respect to R_i^*) be the cumulative rate function of S_i at the input (with respect to output) of the server. For a given time t, let s_0 be the last time instant with no backlog in the server ($s_0 \leq t$). Thus, $R_1^*(s_0) = R_1(s_0)$ and $R_2^*(s_0) = R_2(s_0)$. Since the scheduling policy is work conserving, this yields

$$R_1^*(t) - R_1^*(s_0) + R_2^*(t) - R_2^*(s_0) = C \cdot (t - s_0). \quad (1)$$

Causality implies that $R_2^*(t) \leq R_2(t)$. Thus

$$R_2^*(t) - R_2^*(s_0) \leq R_2(t) - R_2(s_0).$$

Since S_2 is constrained by α_2 and the server adds a constraint on the peak rate of the output source, we obtain

$$R_2^*(t) - R_2^*(s_0) \leq \min\left(C \cdot (t - s_0), \alpha_2(t - s_0)\right). \quad (2)$$

From (1) and (2), we obtain

$$R_1^*(t) - R_1(s_0) \geq C \cdot (t - s_0) - \min\left(C \cdot (t - s_0), \alpha_2(t - s_0)\right).$$

Let us define $(x)^+$ as $\max(0, x)$. Then, $\beta_1 = (\lambda_C - \alpha_2)^+$ is a service curve for S_1, since $C \cdot t - \min(C \cdot t, \alpha_2(t)) = (\lambda_C(t) - \alpha_2(t))^+$.

B. Discussion

The service curve β_1 is conservative. Indeed, if S_2 were preemptive over S_1, the obtained service curve would be the same since, in this case, S_1 receives only the extra capacity unused by S_2. Besides, assume that S_1 and S_2 transit in a second server where they mix with a third source S_3. To derive a service curve for S_1 in the second server, we need an arrival curve for S_2 at the input of the second server. This arrival curve may be obtained by applying Theorem 3 to the arrival curve of S_2 at the input of server 1 and its service curve at server 1. However, the arrival curve for S_2 at the second server is also pessimistic since the service curve for S_2 at server 1 is pessimistic. Thus, the conservative aspect of the result increases with the size of the network. This approach leads inevitably to pessimistic results. For instance, consider a single server and assume S_1 and S_2 have the same traffic descriptor, namely (p, R, M). The following relation exists between the delay bound D_{SC} obtained with the service curve approach and the maximum delay $D_{\max}$: $D_{\max} = ((C - R)/C)D_{SC}$. Thus, when $R \to C/2$ (stability requires that $C > 2R$), $D_{SC} \to 2D_{\max}$.

The weaknesses of this service curve approach demonstrates the necessity of a new approach the problem. Note, however, that a better (more accurate) service curve than β_1 for a FIFO server may exist. The determination of such a service curve remains an open issue.

V. End-to-End Delay in a Tandem Network

In this section, we study the end-to-end delay in a network with two servers in sequence, called a tandem network (except in the first part where the results hold for p servers in sequence) and stress the complexity of an exact analysis.

A. Single Source/p Servers in Sequence

Let S be a source traversing p servers in sequence. The service rate of server j is C_j. We assume that S is constrained by a concave piecewise linear arrival curve α (i.e., S is constrained by a set of leaky buckets). We also assume that $\forall (i, j) \in \{1, \ldots, p\}^2, i \leq j, C_i \geq C_j$, without any loss of generality, since if C_j is greater than C_i the traffic outgoing of server i experiences no delay in j since its peak rate is lower than the service rate of j.

1) Worstcase Analysis: The analysis addresses two dual problems: computation of the maximum end-to-end delay and computation of the buffer requirement at each server. Note that the latter is equivalent to compute the local maximum delay at each server in the case of FIFO scheduling policy.

a) End-to-end Delays:

Lemma 1: Consider a source S traversing p FIFO servers with respective service rates $(C_j)_{j \in 1, \ldots, p}$. The end-to-end delay of a bit of S is the same as if the network were restricted to a single server with a service rate $\min_{j \in \{1, \ldots, p\}} (C_j)$.

Proof: As noted previously, we can assume that the servers rates are decreasing. Let us also assume that server j is backlogged during $[0, T_j]$. This means that during $[0, T_j]$, the output process of server j has a constant rate C_j. Since $C_j \geq C_{j+1}$, server $j + 1$ is also backlogged during $[0, T_{j+1}]$ with $T_{j+1} \geq T_j$. As a consequence, any backlog period of a given server j is included in a backlog period of any server k with $j \geq k \geq p$.

Now consider a bit that experiences some delay in the network. Let j be the first server where it experiences delay. It will also experience some delay at server $j + 1, \ldots, p$. The backlog period at server p has begun at a certain time in the past that we choose to be time zero. The bit has entered the network at time $t \geq 0$. Since the backlog periods are included into each other, the bit that enters at time t, has to wait for all the bits sent in $[0, t]$ to be served by server p. Thus, its end-to-end delay is the same as if the network would only comprise server p. □

Theorem 5: The maximum end-to-end delay of a source constrained by a concave piecewise linear arrival curve α traversing p FIFO servers in sequence is achieved when the source is greedy.

Proof: The maximum end-to-end delay in the network is the same as if the network would only comprise the slowest server (Lemma 1). For the case of a single node, the maximum delay is achieved when the source is greedy since the cumulative rate function of the greedy source is equal to the arrival curve of the source (Theorem 2). This proves the result. □

b) Buffer Requirements: Let us now compute the minimum buffer capacity required at each server to ensure a zero loss rate. We first establish a relation between $S \in \Gamma_{\mathbb{S}}$ and G at the output of the first server. Let $S_{\text{in},i}$ and $S_{\text{out},i}$ be the input and output sources at server i for trajectory $S(A_s = A_{s_{\text{in},1}})$. The following result holds.

Lemma 2: $\forall t \geq 0$, $A_{G_{\text{out},1}}(t) = \min\left(A_{G_{\text{in},1}}(t), C_1 \cdot t\right) = \min\left(\alpha(t), C_1 \cdot t\right)$.

The proof is straightforward (see Fig. 4). From Lemma 2, we can deduce that $A_{G_{\text{out},1}}$ is a concave piecewise linear function. Moreover, $A_{G_{\text{out},1}}$ is an arrival curve for the outgoing traffic of server 1 for any source $S \in \Gamma_{\mathbb{S}}$.

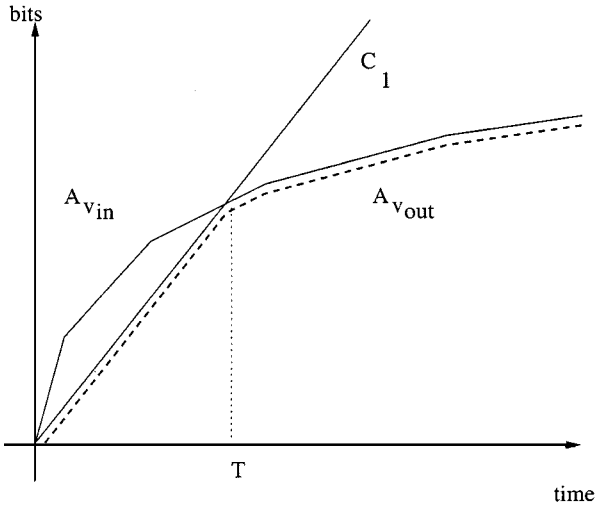


Fig. 4. Output cumulative rate function for a greedy source.

Lemma 3: $\forall S \in \Gamma_{\mathbb{S}}, \alpha^*_{S_{\text{out},1}} \leq A_{G_{\text{out},1}}$.

Proof: To prove that $A_{G_{\text{out},1}}$ is an arrival curve for any trajectory S, we use the definition of an arrival curve: an upper bound on the amount of traffic emitted on any time interval. Suppose that there exists a trajectory S, a time instant t and a time interval τ such that: $x = A_{S_{\text{out},1}}(t+\tau) - A_{S_{\text{out},1}}(t) > A_{G_{\text{out},1}}(\tau)$. Then, necessarily, $\tau > T$ where T (see Fig. 4) is the maximum time where the server is backlogged (and thus, its effective output rate is C_1). For $S_{\text{out},1}$ to produce x during τ, $S_{\text{in},1} = S$ must have at least produced x during a time interval of at most τ in the past, since the scheduling policy is work-conserving

$$\exists t' \leq t, A_{S_{\text{in},1}}(t'+\tau) - A_{S_{\text{in},1}}(t') \geq x > A_{G_{\text{out},1}}(\tau).$$

From Lemma 1 and $\tau > T$, we have

$$A_{G_{\text{out},1}}(\tau) = A_{G_{\text{in},1}}(\tau) = \alpha(\tau).$$

Combining the last two equations, we obtain

$$\exists t' \leq t, A_{S_{\text{in},1}}(t'+\tau) - A_{S_{\text{in},1}}(t') > \alpha(\tau).$$

We, thus, have a contradiction since S is constrained by α. □

A recursive application of Lemma 2 indicates that the worstcase source for each server is generated by the greedy source. Thus, the greedy source yields the maximum backlogs.

Theorem 6: The maximum backlog at each server for a source S with a concave piecewise linear arrival curve α traversing p servers in sequence, is obtained when the source is greedy (see Fig. 5).

Note that the arrival curve obtained from Lemma 2 is better (smaller) than the arrival curve that could be obtained with Theorem 3. However, the result holds only for FIFO servers whereas Theorem 3 holds for any scheduling discipline.

B. M2P Tandem Networks

Consider a tandem m2p network, i.e., a tandem network where sources may enter at node 1 or 2 but exit at node 2 only. Sources that enter the network at node i may be aggregated since they have the same route in the network. Let S_i be the resulting source at node i. S_i is constrained by n_i leaky buckets, where n_i is the number of sources entering at node i. Let α_i be the minimum arrival curve of S_i. Unlike the single source case, buffer requirements and end-to-end delay bounds must be estimated separately.

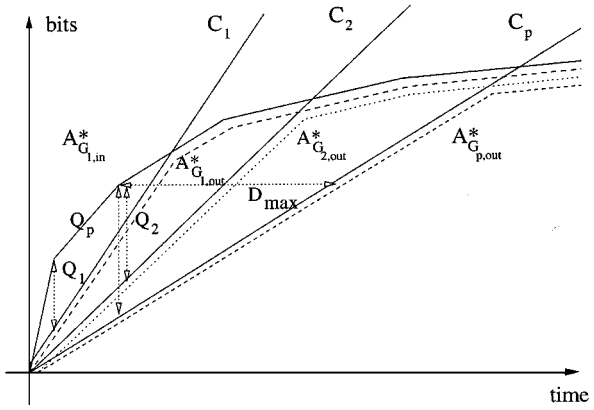


Fig. 5. End-to-end delay (D) and buffer requirements (Q_1, Q_2, Q_3).

1) Buffer Requirements: The results of this part are obtained for m2p with p servers in sequence. Let Q_j be the minimum buffer requirement at server j that guarantees a zero loss rate. Using Theorem 2, we obtain an upper bound for Q_j with the minimum arrival curve of the input flow at server j. This yields the minimum buffer requirement, provided that one can prove that there exists a trajectory of the system such that the input flow at server j has a cumulative arrival curve equal to this minimum arrival curve.

A consequence of Lemma 2 is that the minimum arrival curve ($\alpha^*_{S_{\text{out}}}$) at the output of a FIFO server for an aggregation of leaky bucket constrained sources is maximum when the sources are greedy and strictly synchronous (i.e., they start emitting at the same time instant). Moreover, this minimum arrival curve is a concave piecewise linear function. Thus, S_{out}, the source seen at the output of the server, is multileaky bucket constrained. If this source is to be mixed with a second (leaky bucket constrained) source S_2 and injected in a second server, the maximum backlog is achieved when S_{out} and S_2 are greedy and synchronous. This result can be extended to m2p networks of any size.

Theorem 7: For a given m2p network with leaky bucket constrained sources, the maximum backlog Q_j at server j is achieved when all the sources are greedy and strictly synchronous, i.e., when the sources start emitting at the same time instant.

2) End-to-End Delay: For the single server case, the maximum backlog corresponds to the maximum delay. We prove here that for the case of a tandem m2p networks, achievement of the local maximum delays does not necessarily results in the maximum end-to-end delay. For the single server case, two conditions must be met to obtain the maximum delay: greediness and a strict synchronization (they start emitting at the same time instant) of the sources. For the case of a tandem m2p network, we prove that greediness property is still mandatory (this is intuitively logical since "being greedy" means generating traffic at the maximum possible rate during a maximum period of time, a basic condition to create some backlog), while the synchronization between sources is no more a strict one.

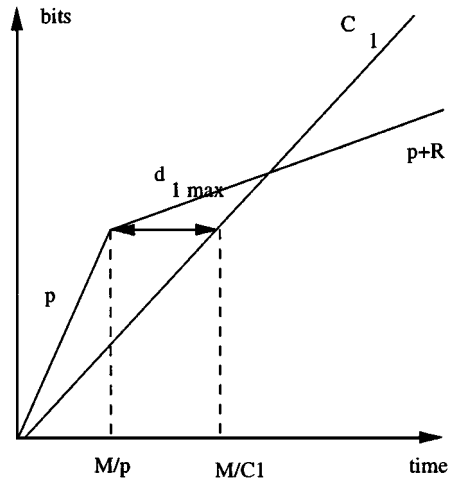


Fig. 6. Maximum backlog generation (server 1).

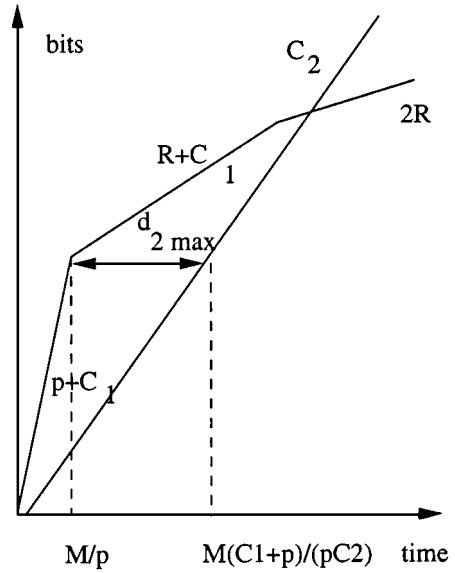


Fig. 7. Maximum backlog generation (server 2).

a) Synchronization: A simple counter-example. Consider a system with two servers with respective service rates C_1 and $C_2\,(C_1 < C_2)$ and two sources S_1 (entering at server 1) and S_2 (entering at server 2). We assume that S_1 and S_2 have the same leaky bucket parameters (p, R, M). We also assume that $0 \leq R \leq C_2 - C_1$. According to the results obtained in the previous section, the maximum backlogs are obtained when S_1 and S_2 are strictly synchronous. This corresponds to the trajectories depicted in Figs. 6 and 7. Let $d_{i\,\max}(i \in 1, 2)$ be the maximum delay at server i (achieved when the sources are greedy and synchronous) and $d_{\max_{S_1,S_2}}$ the maximum end-to-end delay for a given trajectory of S_1 and S_2. Since $C_2 \geq C_1$ and since the 2 sources have the same traffic descriptor, the bit that experiences $d_{1\,\max}$ does not experience $d_{2\,\max}$. Thus, the maximum end-to-end delay is strictly less than the sum of the local maximum delays, i.e., $d_{\max_{S_1,S_2}} \leq d_{1\,\max} + d_{2\,\max}$. Also, since, $R \leq C_2 - C_1$, the maximum backlogs are experienced by the bits that enter the two servers at time $t = M/p$.

Let us now delay the beginning of emission of S_2: S_1 starts at time zero and S_2 starts at time $t = (M/C_1) - (M/p)$, S_1 and S_2 still being greedy. The bit that experiences $d_{1\,\max}$ enters the network at time $t_0 = M/p$. It exits the first server at time $t_0 + d_{1\,\max} = (M/p) + (M/C_1)$. During $[(M/C_1) - (M/p), (M/p) + (M/C_1)]$, S_1 has produced $C_1 \cdot M/p$. Thus, the maximum local delay $d_{2\,\max}$ is achieved at server 2 at time $t_1 = (M/p) + (M/C_1)$, which is the time instant when the bit that experienced $d_{1\,\max}$ at server 1 reaches server 2. Thus, this bit experiences an end-to-end delay equal to $d_{1\,\max} + d_{2\,\max}$, which is strictly greater than in the strictly synchronous case. This example clearly emphasizes the impact of the synchronization among the sources on the end-to-end delay.

b) Worst Case Conditions: We now investigate the conditions yielding the maximum end-to-end delay. The bit that experiences the maximum end-to-end delay is chosen as the reference bit. There are two cases:

1) the reference bit experiences delay in servers 1 and 2;
2) the reference bit experiences delay in server 2 only.

Note that the case "delay in server 1 only" is not possible since if the reference bit experiences some delay in the first server, this means that the server is in a backlog period. Thus, the output process of server 1 has a rate C_1 during a certain time interval. If we mix this flow with S_2 emitting at its peak rate (we assume $p_2 + C_1 \geq C_2$, otherwise server 2 would be transparent to the flow), this resulting flow would create some backlog at server 2 and, thus, the reference bit would necessarily also experience some delay at server 2.

The case "delay in server 2 only" is easy to solve since the problem transforms into determining the maximum delay in a single server with two leaky bucket constrained sources: S_1, with a peak rate equal to C_1 and S_2. We can, thus, apply Theorem 5: the maximum delay is achieved when the two sources are greedy and synchronous.

The case "delay in the two servers" is far more complex as we now see.

c) Delay Equations: We adopt the following notations.

1) θ_1 and θ_2 are the epochs of beginning of the backlog periods where the reference bit enters server 1 and 2, respectively. We set $\theta_1 = 0$. θ_2 might be positive or negative.
2) $d_1(t)$ (with respect to $d_2(t + d_1(t))$) is the delay experienced by the bit entering server 1 at time t (with respect to $t + d_1(t)$). Note that $t + d_1(t) \geq \theta_2$ since we focus on the delay of bits of S_1 experiencing some delay in the two servers.
3) $D(t)$ is the end to end delay of the bit that enters server 1 at time t: $D(t) = d_1(t) + d_2(t + d_1(t))$.

The following equations hold:

$$d_1(t) = \frac{A_{S_{1,\text{in}}}(t) - C_1 t}{C_1}$$

$$d_2\left(t + d_1(t)\right) = \frac{A'_{S_{1,\text{out}}}(t + d_1(t)) + A_{S_{2,\text{in}}}(t - \theta_2 + d_1(t))}{C_2} - (t - \theta_2 + d_1(t))$$

where $A'_{S_{1\text{out}}}$ is the amount of bits generated by S_1, which have already reached server 2 when the bit emitted at time t arrives at server 2. Since $D(t) = d_1(t) + d_2(t + d_1(t))$, we obtain

$$D(t) = \frac{A'_{S_{1,\text{out}}}(t + d_1(t)) + A_{S_{2,\text{in}}}(t + d_1(t) - \theta_2)}{C_2} - (t - \theta_2). \quad (3)$$

If $\theta_2 \geq 0$, $C_2 \geq C_1$, since during $[0, \theta_2]$ server 1 is backlogged and, thus, emits at a constant rate C_1 and no backlog appears at server 2. As a consequence: $A'_{S_{1,\text{out}}}(t + d_1(t)) = A_{S_{1,\text{in}}}(t) - C_1\theta_2$.

If $\theta_2 \leq 0$, S_1 may emit some traffic during $[\theta_2, 0]$ as long as its instantaneous emission rate is smaller than C_1. Let $a^-_{S_1,\text{in}}(\theta_2)$ be the amount of traffic sent by S_1 during $[\theta_2, 0]$. Note that S_1 can emit at least at a constant rate R (the sum of the mean rates of the sources composing S_1). Indeed, a source constrained by a leaky bucket always receives tokens at a rate R (the arrival rate of tokens in its token pool) and, thus, can emit bits at this constant rate without affecting its ability to send later at a rate greater than R. As a consequence: $A'_{S_{1,\text{out}}}(t + d_1(t)) = A_{S_{1,\text{in}}}(t) + a^-_{S_{1,\text{in}}}(\theta_2)$.

Equation (3) may, thus, be rewritten as follows:

$$D(t) = \begin{cases} \dfrac{A_{S_{1,\text{in}}}(t) + a^-_{S_{1\text{in}}}(\theta_2) + A_{S_{2,\text{in}}}\left(\frac{A_{S_{1,\text{in}}}(t)}{C_1} - \theta_2\right)}{C_2} \\ \quad - (t - \theta_2) \quad \text{if } \theta_2 \leq 0 \end{cases} \quad (4)$$

$$= \begin{cases} \dfrac{A_{S_{1,\text{in}}}(t) - C_1\theta_2 + A_{S_{2,\text{in}}}\left(\frac{A_{S_{1,\text{in}}}(t)}{C_1} - \theta_2\right)}{C_2} \\ \quad - (t - \theta_2) \quad \text{if } \theta_2 \geq 0. \end{cases} \quad (5)$$

d) Greediness: Consider the case $\theta_2 \geq 0$. The end-to-end delay D is a function of the cumulative rate functions $A_{s_{1,\text{in}}}$ and $A_{s_{2,\text{in}}}$. These cumulative rate functions are upper bounded by the corresponding arrival curves: $\forall i \in \{1, 2\}, \forall t \geq 0, A_{S_{i,\text{in}}}(t) \leq \alpha_i(t)$. Since D is an increasing function of $(A_{S_{i,\text{in}}})_{i\in\{1,2\}}$, D is maximized when the sources are greedy. We can conclude that, when $\theta_2 \geq 0$, the maximum end-to-end delay is achieved for sources that are greedy starting at a certain time.

When $\theta_2 \leq 0$, we have that

$$A_{S_{1,\text{in}}}(t) + a^-_{S_{1\text{in}}}(\theta_2) \leq \alpha_1(t - \theta_2) \quad (6)$$

since S_1 is constrained by α_1 and the considered time interval has a duration $t - \theta_2$. The right-hand side and left-hand side of (6) are not necessarily equal since during $[\theta_2, 0]$, S_1 must have an instantaneous emission rate less or equal than C_1. We make use of the following lemma to prove that (6) is an equality.

Lemma 4: The maximum end-to-end delay is achieved at a time instant t such that after t, $S_{1,\text{in}}$ is not able to emit at a rate greater than C_1.

Proof: Let us prove the result by contradiction. Suppose that the maximum end-to-end delay is achieved for a bit sent at time t and suppose that, after time t, S_1 is still able to send at a rate greater than C_1, say during $[t, t + \delta]$. The end-to-end delay at time t is given by (3)

$$D(t, \theta_2) = \frac{A_{S_{1,\text{out}}}(t + d_1(t)) + A_{S_{2,\text{in}}}(t - \theta_2 + d_1(t))}{C_2} - (t - \theta_2).$$

Let us now delay the beginning of emission of S_2 by an offset δ and compute the delay at time $t + \delta$ (with the assumption that S_1 emits at rate C_1 during $[t, t + \delta]$). Equation (3) gives

$$\begin{aligned} D(t + \delta, \theta_2 + \delta) = &\frac{A_{S_{1,\text{out}}}(t + \delta + d_1(t + \delta))}{C_2} \\ &+ \frac{A_{S_{2,\text{in}}}((t + \delta)}{C_2} \\ &+ \frac{(\theta_2 + \delta) + d_1(t + \delta))}{C_2} \\ &- ((t + \delta) - (\theta_2 + \delta)). \end{aligned}$$

Since $d_1(t + \delta) = d_1(t) + \delta$ (during $[t, t + \delta]$, the backlog at server 1 is increased by $C_1 \cdot \delta$), we obtain

$$D(t + \delta, \theta_2 + \delta) > D(t, \theta_2).$$

The result is thus proved by contradiction. □

A consequence of Lemma 4 is that (6) may be an equality for the instant of interest (where the maximum delay is achieved): it is possible to send $\alpha_1(t)$ during $[0, t]$ and, $\alpha_1(t - \theta_2) - \alpha_1(t)$ during $[\theta_2, 0]$, since during this period of time, the emission rate of S_1 is less than C_1.

As a consequence, D is an increasing function of $A_{S_{1,\text{in}}}(t) + a^-_{S_{1\text{in}}}(\theta_2)$ (and also of $(A_{S_{i\text{in}}})_{i\in\{1,2\}}$) in the case when $\theta_2 \leq 0$. Thus, the end-to-end delay is maximized when the sources are greedy starting at a certain time.

To summarize, the greediness of the sources is mandatory in any case (backlog at server 2 only or in the two servers with $\theta_2 \geq 0$ and $\theta_2 \leq 0$).

Lemma 5: For any tandem m2p network, the maximum end-to-end delay is achieved when the sources are greedy with different starting times.

Equations (5) and (7) can be rewritten using Lemma 5

$$D(t) = \begin{cases} \dfrac{\alpha_1(t - \theta_2) + \alpha_2\left(\frac{\alpha_1(t)}{C_1} - \theta_2\right)}{C_2} \\ \quad - (t - \theta_2) \quad \text{if } \theta_2 \leq 0 \end{cases} \quad (7)$$

$$= \begin{cases} \dfrac{\alpha_1(t) - C_1 \cdot \theta_2 + \alpha_2\left(\frac{\alpha_1(t)}{C_1} - \theta_2\right)}{C_2} \\ \quad - (t - \theta_2) \quad \text{if } \theta_2 \geq 0. \end{cases} \quad (8)$$

e) Delay Function Study: We consider sources with piecewise linear concave arrival curves. D, as defined in (8), is thus a concave function since concavity is preserved by summation and composition. It has a bell-shaped curve, which starts from zero at time $t = 0$ and goes back to zero at time $t = T$, where T is the duration of the network backlog. There is, thus, only one local maximum. For the remaining of this section, we assume that $\theta_2 \geq 0$. A similar study (though not obvious) could be carried out for $\theta_2 \leq 0$.

f) Synchronization: We want to study the influence of θ_2. To do so, we consider the derivative function $(dD/(d\theta_2))(t)$. Equation (8) (f' stands for the derivative of f) gives

$$\frac{dD}{d\theta_2}(t) = \frac{(C_2 - R) - \alpha_2'\left(\frac{\alpha_1(t)}{C_1} - \theta_2\right)}{C_2} \quad \text{if } \theta_2 \geq 0 \qquad (9)$$

D is maximized for $\theta_2 = \theta_{2\max}$ and $t = t_{\max}$. We study, for t set to $t_{\max}$, the influence of θ_2. Since $\theta_2 \geq 0$, $C_2 \geq C_1$. Also, since the bit that experiences the maximum end-to-end delay experiences some delay in the two servers, S_2 must start emitting no later than at the arrival time of the reference bit in the second server, i.e., at time $\theta_{2\max} = (\alpha_1(t_{\max}))/C_1$. $D(t)$ has, thus, one maximum in $[0, (\alpha_1(t_{\max}))/C_1]$. The derivative function can be interpreted as follows: it is all benefit to trigger S_2 sooner than a given θ_2 (say at time $\theta_2 - \delta$) if the value of α_2' is greater than $C_2 - C_1$ After $(\alpha_1(t_{\max})/C_1) - \theta_2$ (arrival time of the reference bit at server 2, which is backlogged from time θ_2), that is if the amount of work done by server 2 in $[\theta_2 - \delta, \theta_2]$, i.e., $C_2 \cdot \delta$, is less than what S_2 and S_1 can produce during the interval $[\theta_{2\max} - \delta, \theta_{2\max}]$, i.e., $C_1 \cdot \delta + \alpha_2'((\alpha_1(t_{\max})/C_1) - \theta_2)\delta$.

g) Conclusion: We usually do not know the conditions leading to the maximum end-to-end delay (whether the reference bit experiences some delay in the two servers or in the second server only). For the two server cases, it is possible to derive the value of θ_2 (see [16]). However, the analysis does not scale easily to larger networks. A bounding approach is, thus, necessary for larger networks. A first step toward this objective is the introduction of the concept of *additivity*.

h) Delay Additivity: We say that the delay in a tandem m2p network is additive if the maximum end-to-end delay $D_{\max}$ is equal to the sum of the local maximum delays $d_{i\max}^G$. Note that it is not the case in general. Indeed, if $d_{i\max}$ is the maximum delay at server i for the trajectory leading to the maximum end-to-end delay, we have: $\forall i$, $d_{i\max} \leq d_{i\max}^G$ and, thus, $D_{\max} \leq \sum_i d_{i\max}^G$.

i) Additivity Conditions: Let us first remark that the only chance for the end-to-end delay to be equal to the sum of the maximum local delays is that the bit that experiences the maximum delay in server 1 also experiences the maximum delay in server 2.

We adopt the following conventions.

1) $t = 0$ is the time instant corresponding to the beginning of the activity period at the first server.
2) $t_{1\max}$ is the arrival time of the reference bit that experiences the maximum delay in the first server ($t_{1\max} = \max(t|(d\alpha_{S_1}(t)/dt) > C_1)$).
3) $t_{1\max} + d_1(t_{1\max}) = t_{1\max} + d_{1\max}$ is the arrival time of the bit at the second server ($d_{1\max}$ is the maximum delay at server 1. $d_{1\max} = (\alpha_1(t_{1\max})/C_1) - t_{1\max}$).

Let A_2 be the cumulative rate function of S_2 and θ_2 its instant of beginning of emission. Since the maximum delay at server 2 must be achieved at time $t_{1\max} + d_{1\max}$, the following conditions must hold:

$$\frac{d\left(\alpha_{1\text{out}} + A_2\right)(t)}{dt} > C_2 \text{ for } t \in [\theta_2, t_{1\max} + d_{1\max}] \qquad (10)$$

$$\frac{d\left(\alpha_{1\text{out}} + A_2\right)(t)}{dt} < C_2 \text{ for } t > t_{1\max} + d_{1\max}. \qquad (11)$$

In the interval $[0, t_{1\max} + d_{1\max}]$, the output rate of server 1 is C_1 (backlog period). Thus, (10) and (11) become

$$\frac{dA_2(t)}{dt} > C_2 - C_1 \text{ for } t \in [\theta_2, t_{1\max} + d_{1\max}] \qquad (12)$$

$$\frac{dA_2(t)}{dt} < C_2 - C_1 \text{ for } t > t_{1\max} + d_{1\max}. \qquad (13)$$

For the previous conditions to hold, a necessary condition is $C_2 \geq C_1$. We also know that a necessary condition to generate $d_{2\max}$ is that S_2 is greedy. Let $t_{2\max} = \max(t|(d\alpha_2(t)/dt) > C_2 - C_1)$. A necessary and sufficient condition for the bit arriving at time $t_{1\max} + d_{1\max}$ (and experienced $d_{1\max}$) to experience $d_{2\max}$ is that $t_{2\max} \leq t_{1\max} + d_{1\max}$ (since then $\theta_2 = t_{1\max} + d_{1\max} - t_{2\max}$). We obtain the following theorem.

Theorem 8: In a tandem m2p network, sources can be synchronized so as to generate an end-to-end delay equal to the sum of the local maximum delays if and only if

$$\begin{cases} C_2 \leq C_1 \text{ and } t_{1\max} + \frac{\alpha_1(t_{1\max})}{C_1} \geq t_{2\max}, \text{ with:} \\ t_{1\max} = \max\left(t|\frac{d\alpha_1(t)}{dt} > C_1\right) \\ t_{2\max} = \max\left(t|\frac{d\alpha_2(t)}{dt} > (C_2 - C_1)\right). \end{cases} \qquad (14)$$

Lower Bound: Let us now assume that the conditions of Theorem 8 are not fulfilled, i.e., $t_{1\max} + (\alpha_1(t_1\max)/C_1) \geq t_{2\max}$ where

$$t_{1\max} = \max\left(t|\frac{\alpha_1(t)}{dt} > C_1\right) \qquad (15)$$

$$t_{2\max} = \max\left(t|\frac{(\alpha_{1\text{out}} + \alpha_2)(t)}{dt} > C_2\right). \qquad (16)$$

Note that the definitions of $t_{2\max}$ given in (14) and (16) are equivalent since in (14), C_1 is the rate of the greedy source seen at the output of server 1. The key idea is to build a trajectory of S_1 where the burst leading to $d_{1\max}$ is delayed in such a way that the reference bit (experiencing $d_{1\max}$) exits server 1 at time $t_{2\max}$ (we still assume that S_2 is greedy during $[0, t_{2\max}]$). This trajectory is defined as follows:

1) S_1 is silent during $[t_{2\max} - d_{1\max}, t_{2\max}]$;
2) S_1 is greedy during $[t_{2\max} - d_{1\max} - t_{1\max}, t_{2\max} - d_{1\max}]$.

We still have to define the trajectory of S_1 during $[0, t_{2\max} - d_{1\max} - t_{1\max}]$. We assume that it is maximal, i.e., that S_1 produces as much traffic as possible, under the constraint that it remains able to generate a burst leading to $d_{1\max}$ for $t \geq t_{2\max} - d_{1\max} - t_{1\max}$. S_1 is, thus, able to generate as many bits as the greedy source during $[0, t_{2\max} - d_{2\max}]$, i.e., $\alpha_{1\text{out}}(t_{2\max} - d_{1\max})$. If the sources were greedy and synchronous, the second server would receive $\alpha_{1\text{out}}(t_{2\max})$ in $[0, t_{2\max}]$. Thus, one "looses" (as compared with the strictly synchronous case) the difference Q between these two quantities, that is $Q = \alpha_{1\text{out}}(t_{2\max}) - \alpha_{1\text{out}}(t_{2\max} - d_{1\max})$. Thus, the bit of S_1 that experiences $d_{1\max}$ in the first server

experiences $d_{2\max} - (Q/C_2)$ in the second server. The end to end delay of this bit is, thus

$$D = d_{1\max} + d_{2\max} - \frac{\alpha_{1\text{out}}(t_{2\max}) - \alpha_{1\text{out}}(t_{2\max} - d_{1\max})}{C_2}. \tag{17}$$

Since $\alpha_{1\text{out}}$ is given by Lemma 1, we can easily compute D. We have, thus, obtained a lower bound for the end-to-end delay since D corresponds to a trajectory of the system. If D is close to the sum of the local maximum delays, this would prove that the sum of maximum local delays provides a good approximation of the end-to-end delay. We further investigate this approach in the next section to obtain a bound on the end-to-end delay for m2p networks of arbitrary sizes.

VI. General Multipoint-to-Point Networks

In the previous section, we proved that a tandem m2p network is additive if and only if $t_{2\max} \leq t_{1\max} + d_{1\max}$. We also characterized the corresponding additive trajectory: S_1 and S_2 greedy respectively from times $\theta_1 = 0$ and $\theta_2 = t_{1\max} + d_{1\max} - t_{2\max}$. In the following, we call additive bound, the sum of the local maximum delays along the route of an m2p network. We generalize the approach of the previous section to the case of p servers in sequence. First, note that any m2p network with p servers in sequence can be partitioned in a set of subnetworks for which the following property either holds or not.

Property 1: For any two adjacent servers j and $j+1$, we have: $t_{(j+1)\max} \leq t_{j\max} + d_{j\max}$.

A. Additive Networks

Consider an m2p network with p servers in sequence for which Property 1 holds. Let $(\theta_j)_{j\in\{1,\ldots,p\}}$ be defined as follows:

1) $\theta_1 = 0$;
2) $\theta_{j+1} = \theta_j + \left(t_{j\max} + d_{j\max} - t_{(j+1)\max}\right)$, $j \in \{1,\ldots,p-1\}$.

If S_j is greedy, starting from time $t = \theta_j$, (note that $\theta_{j+1} \geq \theta_j$), the bit experiencing $d_{1\max}$ at server 1 experiences $d_{j\max}$ at server j for all $j \in \{1,\ldots,p\}$. The end-to-end delay of this bit is thus: $D_{\max} = \sum_{j=1}^{p} d_{j\max}$. An m2p network for which Property 1 holds is thus additive. Besides, since the only way for a bit to experience $\sum_{j=1}^{p} d_{j\max}$ is to experience $d_{j\max}$ at server j ($j \in \{1,\ldots,p\}$), it follows that a network that does not fulfill Property 1 is not additive. This means that Property 1 is a necessary and sufficient conditions for m2p networks with p servers in sequence to be additive.

B. Nonadditive Networks

In this section, we generalize the lower bound approach initiated in the tandem network case. We then use this lower bound to test the accuracy of the additive bound in the case of nonadditive networks. A straightforward generalization would hide the difficulty of the construction of the trajectory. Therefore, we first present the three-server case.

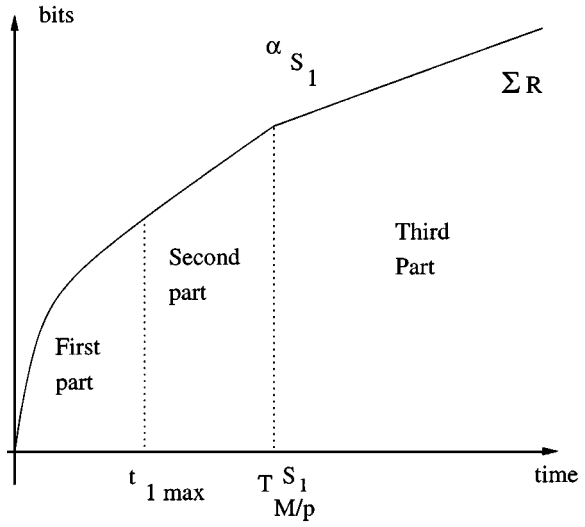


Fig. 8. S_1 initial trajectory.

1) Three-Server Case:

a) Lower Bound: First, consider a two-stage network. If it is nonadditive, this means intuitively that the burst leading to $d_{1\max}$ is not sufficient to obtain $d_{2\max}$ at server 2 (considering the greedy trajectory of the system), since when all the bits from this burst have reached server 2, the local delay on this server is less than $d_{2\max}$. The idea behind the lower bound approach is to delay the burst at server 2 so as to synchronize local maximum delays. Obviously, the delay at the second server will necessarily be less than the delay in the greedy synchronous case.

Consider now an m2p network with three servers and three sources $(S_i)_{i\in\{1,\ldots3\}}$ (S_i entering at node i). The trajectories of the sources are chosen so as to maximize the amount of bits in buffer j when the reference bit (the one experiencing $d_{1\max}$ at server 1) arrives.

b) Trajectory of Sources: Consider the greedy trajectory of S_1 (see Fig. 8). It can be divided into three parts. The first part corresponds to the part of the trajectory necessary to achieve the local maximum delay $d_{1\max}$. The second part corresponds to the time interval necessary for the last bucket of the sources composing S_1 to empty. In the last part, all the sources composing S_1 emit at their mean rate.

Now consider the trajectory of S_1 given in Fig. 9. S_1 is a multiplex of n_1 sources. The traffic descriptor of source k is (p_k, R_k, M_k) ($k \in \{1,\ldots n\}$). For the greedy trajectory of the system, the source with index k emits at its peak rate p_k during $[0, (M_k/p_k)]$ and then emits at its mean rate R_k. Let us define

$$T^{S_1}_{(M/p)} = \max_{k\in\{1,\ldots,n_{S_1}\}} \left(\frac{M_k}{p_k}\right).$$

$T^{S_1}_{(M/p)}$ corresponds to the beginning of the third part defined in Fig. 8. The modified trajectory is built by changing the beginning of emission of the sources composing S_1 as follows:

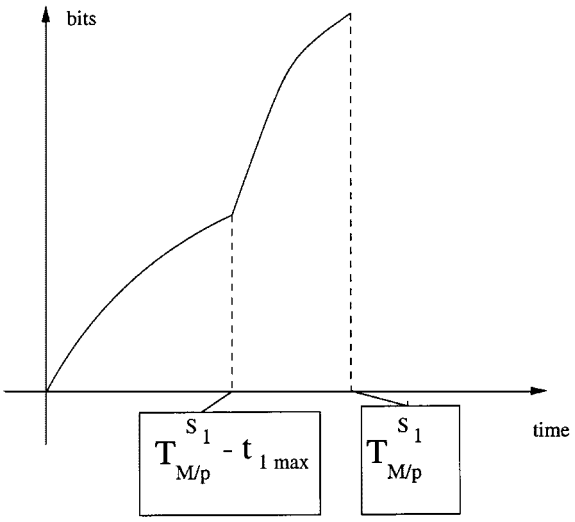


Fig. 9. S_1 modified trajectory.

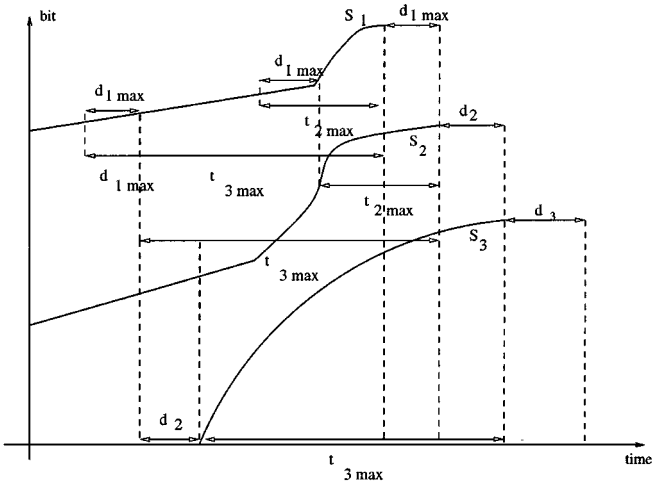


Fig. 10. Synchronization of sources.

1) if $(M_k/p_k) \leq t_{1\max}$ then S_k:
   a) emits at its mean rate during $\left[0, T^{S_1}_{(M/p)} - t_{1\max}\right]$,
   b) becomes greedy for $t \geq T^{S_1}_{(M/p)} - t_{1\max}$ (this is possible since its bucket is still full at this time).
2) if $(M_k/p_k) \geq t_{1\max}$, then S_k:
   a) emits at its mean rate during $\left[0, T^{S_1}_{(M/p)} - (M_k/p_k)\right]$,
   b) becomes greedy for $t \geq T^{S_1}_{(M/p)} - M_k/p_k$.

The modified trajectory has two parts (see Fig. 9).

1) A first part where some sources emit at their peak rate whereas others emit at their mean rates. This part corresponds to the second part of the initial greedy trajectory with a slight modification: if a source emits at its peak rate during τ_1 and then at its mean rate during τ_2 in the initial trajectory, then, in the modified trajectory, it first emits at its mean rate during τ_2 and then at its peak rate during τ_1. Due to this inversion between τ_1 and τ_2, we term this part the "inverted part" of the modified trajectory.
2) A second part, strictly equivalent to the first one in the initial trajectory.

Note that, as with the initial trajectory, the last bucket empties at time $t = T^{S_1}_{(M/p)}$. A modified trajectory is built for S_2 and S_3 using the same method. We now set the synchronization parameters.

c) Synchronization of Sources: With the modified trajectory described above for S_1, the last bit of the burst (reference bit) experiences a delay $d_{1\max}$ at the first server. S_2 is triggered so that the end of its burst corresponds to the arrival of the reference bit. This bit will then experience $d_2 \leq d_{2\max}$ in the second server. Since, *a priori*, $T^{S_1}_{(M/p)} \neq T^{S_2}_{(M/p)}$, the previous synchronization method leads one of the sources to start emitting before the other. Assume that S_2 starts emitting before S_1. To maximize the number of bits backlogged at server 2 at the time where the reference bit arrives, it is possible to modify the trajectory of S_1 such that it emits at its mean rate before the beginning of the modified trajectory, in an interval of length $T^{S_2}_{(M/p)} - T^{S_1}_{(M/p)}$. This trajectory of S_1 is valid with respect to its leaky bucket constraint. The same method is applied to synchronize S_3, as shown in Fig. 10.

TABLE I
SOURCES DESCRIPTORS

Peak rate p	*Mean rate M*	*Burstiness M*
10	0.1	10
100	1	100
1000	10	1000

d) Result for Delay: The lower bound on the maximum end-to-end delay is obtained as the end-to-end delay of the reference bit in the modified trajectory. Since all the sources are leaky bucket constrained, the initial and modified trajectories correspond to piecewise linear curves. Computation of the intrinsic parameters as well as the delay of the reference bit is, thus, straightforward from the algorithmic point of view.

2) Numerical Results: We want to estimate the accuracy of the additive bound in a nonadditive m2p network by using the lower bound presented above. Accuracy means here the relative difference between the additive bound and this lower bound. We consider m2p networks with $p = \{4, 5, 8, 10\}$ servers in sequence. For each server, we draw the number of sources entering at this stage in a uniform fashion in the set $\{1, \ldots, 5\}$. Characteristics of the sources are also randomly chosen from Table I using a uniform law. We have to set the capacities of the servers. A necessary condition for a network to be additive is that the rates of the servers increases (from the leaves to the root). Conversely, if capacities decrease, the network is nonadditive (sufficient but not necessary). We set the service rate of all servers to be equal to the sum of the mean rates of all the sources times $\gamma = 1.01$ (γ is used to ensure stability). This sum represents the minimum capacity of the last server. Doing so, the most important part of the end-to-end delay is concentrated on the last server of the network. To obtain significant results, we calculate the relative range, defined as the difference between the lower bound and the additive bound divided by the additive

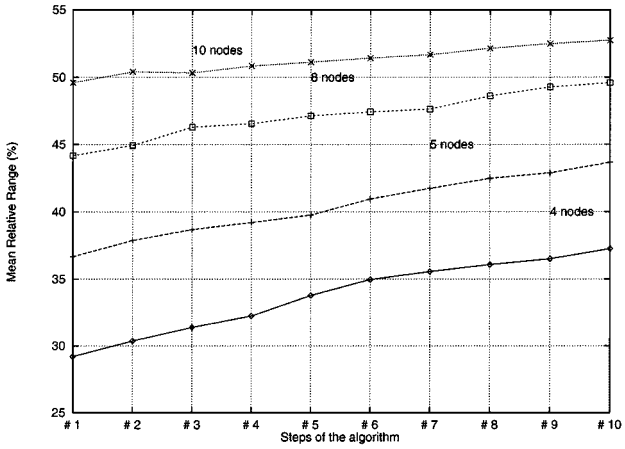


Fig. 11. Accuracy of the additive bound.

bound, for this initial system, i.e., a particular random generation of the sources descriptors and capacities of servers. We then change the input server of some of the sources. The following algorithm is applied:

Step 1: the initial network is built.
Step 2–9: each source is "moved" from node j to node $j-1$ with probability 0.1.

Applying this algorithm, the m2p network heuristically "worsens" and, thus, the relative range should increase.

The results, presented in Fig. 11, are obtained for 10 000 successive random generations of networks. The x axis is indexed following the steps of the algorithm. For each step of the algorithm and for each network size, we compute the mean relative range.

a) Discussion: For nonadditive m2p networks, we have proposed an upper bound on the end-to-end delay, the additive bound and a heuristically obtained lower bound. The maximum, exact, end-to-end delay over all possible trajectories of the system lies between these two bounds and gives full meaning for considering the relative range as a performance parameter. The obtained results confirm the good accuracy of the additive bound. The mean relative ranges remain reasonable even for large size of networks. The maximum error, not presented here is no more than 67%. It thus remains within the same order of magnitude, which clearly indicates that the additive bound is a valid approximation of the end-to-end delay.

C. Well-Formed M2P Networks

Our expectation is that the additive bound always represents an accurate upper bound on the maximum end-to-end delay for m2p networks. Proving such a statement requires an exhaustive study, which is not possible. We restrict our study to a specific class of m2p networks, that we term well-formed m2p networks. A well-formed m2p network is an m2p network where the following rule applies: capacities of the servers increase from the leaves to the root of the tree. We extend here the previous results to the case of well-formed m2p networks with p servers in sequence, using the same lower bound as in the nonadditive case. Indeed, the method used to build the trajectory leading to the lower bound is based only on the set of intrinsic parameters $(t_{j\max}, d_{j\max})$. It does not rely on any assumption concerning the additivity of the network. It may thus be applied to the case of well-formed m2p networks.

TABLE II
AVERAGE RELATIVE RANGES (IN%) WITH NETWORKS OF DIFFERENT SIZES

	Size=3	*Size=5*	*Size=10*	*Size=15*	*Size=20*
$\alpha = 1.1$	0.72	1.29	2.03	2.89	3.87
$\alpha = 1.5$	1.97	3.36	5.67	8.42	11.23
$\alpha = 2.0$	1.96	3.13	5.06	7.64	10.18

1) Results: The method used to generate a well-formed network is the following.

1) For each server, the number of sources (between 1 and 5) entering at this node and their characteristics are drawn from Table I using uniforms laws.
2) The service rate of server j is then set to the sum of the mean rates of the sources served by this server times a coefficient α. α can take one of the three following values $\{1.1, 1.5, 2.0\}$, which, for each set of sources, leads to three different networks.

We present in Table II the numerical results obtained for networks of various sizes (from 3 to 20 servers). For each network size, 10 000 networks are drawn. The performance parameter computed for each network is the relative range between the lower bound and the additive bound.

2) Discussion: The results in Table II strongly confirm our claim: the additive bound represents an accurate approximation of the end-to-end delay. These results are also interesting since the way well-formed m2p networks are built here is close to a real dimensioning process. Indeed, α^{-1} is the rate of the server divided by the sum of the average rates of the sources it serves. It thus represents the average activity rate of the servers and tuning activity rates at a given value is a common way to dimension networks. Compared with the results obtained in the previous section, the relative ranges obtained here are significantly smaller: for instance, for a network with ten servers, the relative range was close to 50% whereas, here, it is close to 5%. This is due to the method used to set the server rates in each case. In the case of strictly nonadditive m2p networks, all the servers had the same capacity, which lead to a strictly nonadditive network, whereas here, the rates increase from one server to another, which is a necessary (though not sufficient) condition to obtain an additive network.

VII. ADMISSION CONTROL ALGORITHM

In this section, we derive an admission control algorithm based on the additive bound presented in the previous section. We propose two versions of the algorithm, a centralized and a distributed version.

A. Centralized Algorithm

Consider first a single FIFO server and n leaky bucket constrained sources. The maximum delay is obtained when all the sources are greedy and strictly synchronous. An arrival curve of the aggregated source is the sum of the arrival curves of all

the sources. Since summation is a commutative operation, the maximum delay does not depend on the order in which sources are introduced in the network. Thus, from the admission control algorithm point of view, the answer to the admission request of a new source in a single-server network with n established sessions is equivalent to the answer to the request of the $n+1$ sources simultaneously.

Consider now an m2p network. With an admission control algorithm based on the additive bound, the admission of a new session requires to compute the local maximum delay at each server along the path of the session up to the root server. Since the root server belongs to the path of all sources, admitting a new source, with n sessions already set-up, is equivalent to admit these $n+1$ sources simultaneously. We make use of this property to simplify the presentation of the centralized algorithm. The problem to solve is the following: "Given n sources with specific QoS requirements, is it possible to admit these n sources simultaneously?" The algorithm has two phases: 1) computation of the additive bounds along each path of the network; and 2) checking the nonviolation of the QoS constraint of each session.

1) Computing the Additive Bound: So far, the computation of the additive bound as been presented only in the case of m2p networks with servers in sequence. Generalization to a tree m2 networks relies on the following observation: the flow seen at the output of a given subtree of a given m2p network is multileaky bucket constrained (see Lemma 2 and 3). As a consequence, 1) maximum local delays are obtained when all the sources are greedy and synchronous; and 2) computation of these delays can be made starting from the leave servers and moving to the root server.

2) Checking the QoS Constraint: Once the maximum local delays are obtained, we can compute the additive bound along each path of the network. We assume that the centralized algorithm has a complete knowledge of the network topology and of the input server of each source (which is equivalent to know its path in the network). We must thus compare, for each session, the required end-to-end delay and the additive bound along its path to accept or reject the new session request.

B. Distributed Algorithm

When executed, with n sources already accepted, the admission control algorithm must process the request from a new source. Let $(S_i)_{i\in\{1,\ldots,n\}}$ be these already accepted sources, $(D_i)_{i\in\{1,\ldots,n\}}$ their delay requirements and $(D_i^{\text{eff}})_{i\in\{1,\ldots,n\}}$, the effective delays, i.e., the additive bounds along the path of the sources.

Since $(S_i)_{i\in\{1,\ldots,n\}}$ have already been accepted, $D_i \leq D_i^{\text{eff}} (\forall i \in \{1,\ldots,n\})$. The quantities $\delta_i = D_i^{\text{eff}} - D_i (i \in \{1,\ldots,n\})$ represent safety margins for the sources.

The admission of a new source S_{n+1} requires to recompute local delays for all servers along the path of S_{n+1}, but not on all the servers in the network. As explained before, this operation can be made sequentially starting from the input server of S_{n+1} and moving down to the root server. To limit the amount of computation, each server stores the arrival curve of the flow seen at each of its inputs. Let I_i be the set of indices of the servers along the path of S_i and $(\Delta_j)_{j\in I_{n+1}}$ the variations of the local maximum delays induced by S_{n+1} at the servers of I_{n+1}. The admission algorithm must check whether the admission of the new source violates the QoS requirements of the other sources, which can be expressed through the following system of equations:

$$\delta_i \geq \sum_{k\in I_i\cap I_{n+1}} \Delta_k, \forall i \in \{1,\ldots,n\}. \tag{18}$$

For each source S_i, $i \in \{1,\ldots,n\}$, $I_i \cap I_{n+1}$ is the set of servers where S_i and S_{n+1} meet. This set is never empty: it contains at least the root server. The problem with the distributed algorithm is that checking (18) can be made only at the root server, since this is only at this server that all the local delay variations $(\Delta_j)_{j\in I_{n+1}}$ are known. The final admission decision is thus made at this last step. Therefore, we have two options.

1) If we want to limit the amount of messages exchanged between servers, safety margins should be stored in the root server only. Then, to check (18), the root server needs to have a complete knowledge of the network topology and of the path of each session. This is possible only for small networks and a small number of sources. Another drawback of this method is that QoS violations are detected at the last possible moment. For instance, if at the input server of S_{n+1}, the delay variation Δ is greater than the safety margin δ of a source served by this server, the admission algorithm could have been aborted at this step of the procedure.
2) On the opposite, if we impose that a server only has a local vision of the network, then, once a source is accepted, the following operations must be performed:
   a) the safety margin δ must be distributed among all the servers of the path of this source;
   b) if a safety margin is modified because of a new source has been accepted, it must be transmitted to the next server since the two sources now share the same path and, thus, the new source will change this safety margin on every server until the root server is reached;
   c) if a source is accepted, one must ensure that all safety margins are correctly updated.

We now present an algorithm that does not rely on the assumption that the root server has a complete knowledge of the network topology and of the routes of the sources. We describe the data structures used at each server and provide a skeleton of the two phases of the algorithm: in the first phase, local delay variations are computed and QoS violations are checked. In the second phase, called the termination phase, the decision to admit or reject the new source is made.

1) Data Structures: Each server stores a table with, for each source that it serves, an identifier and its safety margin. Each server must also store the arrival curve of the input flow at each of its interfaces (when a new source arrives, only one arrival curve is modified). An arrival curve is stored as a list of points since with leaky bucket constrained sources and FIFO servers, arrival curves are piecewise concave linear functions. The maximum local delay $d_{\max}$ must also be stored.

2) First Step: Admitting a New Source: This step of the algorithm is initiated by the input server of the new source and is propagated sequentially until the root server is reached. Let D be the sum of the maximum local delays between the input server of the new source and the current server. D becomes equal to the additive bound when the current server is the root server.

Algorithm at server $j (j \in I_{n+1})$
1. Upon receipt of a new session request: computation of the new arrival curve of the input flow.
2. Safety margins are (temporarily) updated using the list of modified safety margins received from the previous server. They become effective only if the request is accepted.
3. Computation of the new value of the maximum local delay $d_{\max}$, which gives the variation Δ_j and the new value of D, i.e., $D + d_{\max}$.
4. If $[(\min_{i|j\in I_i} \delta_i \geq \Delta_j)$ and $(D \leq D_{n+1})]$ then: (the source is locally accepted)
 (a) if (server $j ==$ root server) then:
 i. A *Confirmation of Acceptance Message* is sent to the previous server in the path of S_{n+1} with the new values of the safety margins of the sources arriving at this interface.
 ii. *Updating Messages* are sent on the other interfaces with the final value of the safety margins (which are known only at this stage).
 (b) else:
 i. Storage of the new (temporary) value of the safety margins: $\forall i, j \in I_i, \delta_i = \delta_i - \Delta_j$
 ii. Transmission to the next server in I_{n+1} of:
 A. the set $(\delta_i)_{j\in I_i}$;
 B. the new value of D;
 C. the arrival curve at the output.
5. Else: a *Rejection Message* is sent to the previous server in I_{n+1}.

3) Second Step: Updates: This phase is initiated by the root server or by the server where a QoS violation is detected. There are three cases:

1. Receipt of a *Confirmation of Acceptance Message*. Only servers from I_{n+1} may receive this message. The server must:
 (a) update the safety margins with the received values (including S_{n+1});
 (b) forward this message to the previous server in I_{n+1};
 (c) send *Updating Messages* on the other ingress links with the corresponding safety margins values.
2. Receipt of a *Rejection Message*. Only servers from I_{n+1} may receive this message. They have to:
 (a) release the temporary structures (safety margin values, arrival curve and maximum local delay);
 (b) forward this message to the previous server in I_{n+1}.
3. Receipt of an *Updating Message*. Only servers that do not belong to I_{n+1} may receive this message. The server must:
 (a) update its current safety margins with the received values;
 (b) forward this message with the corresponding safety margins on each of its ingress links.

4) Session Termination: The admission control algorithm is also executed at each session termination. The local delays for the servers of the path of this session must be updated as well as the safety margins of the sources. The procedures involved are similar to the ones used for acceptance.

5) Discussion: To ensure the correctness of the algorithm, two admission procedures cannot be made simultaneously. They must be sequentialized. If the two admission procedures are initiated on two disjoint paths in the tree network, the root server will have to choose which source is treated first. This will obviously have an impact on the other source since the source that is treated first is more likely to be accepted than the second one. Note, however, that the admission procedure for the second source does not have to be reinitiated. For the case where the two admission procedures are initiated on the same path, the first one that reaches the first server that the two sources share, gains priority over the other one. This means that the second procedure is delayed until the decision for the first source is made. Note, finally, that the algorithm converges as long as no message is lost. A reliable communication channel, such as a permanent TCP connection, may be used to ensure that no messages are lost between adjacent servers.

C. Example

We further illustrate the distributed admission control algorithm presented above for the case of the m2p network of Fig. 12. For this figure as well as Figs. 13 and 14, we adopt the following conventions.

1) Each source is constrained by a single leaky bucket.
2) S_j is the source entering the network at server j, D_j its required end-to-end delay and δ_j its safety margin.
3) For each server, $d_{j\max}$ is the current maximum local delay at a given step of the algorithm and $d'_{j\max}$ the new value of the maximum local delay. Δ_j is the maximum local delay variation, i.e., $\Delta_j = d'_{j\max} - d_{j\max}$.
4) Procedures used at each server are represented with squares linked to servers in Figs. 13 and 14. Expressions like "$D = d'_1 \leq D_1$?" correspond to tests performed by the server. We suppose that all the tests succeed, which allows to study the acceptance of a new source.
5) Arrows between squares correspond to data exchanges.

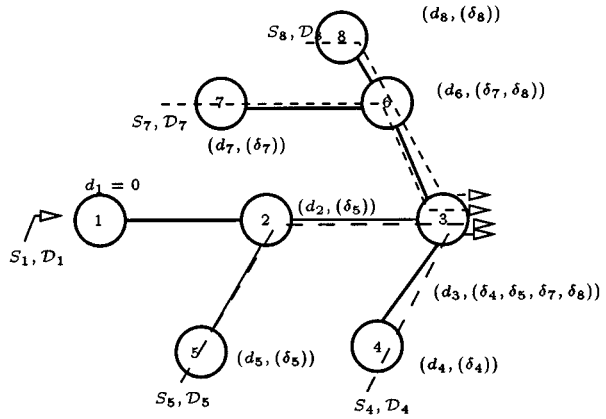


Fig. 12. m2p network with eight servers.

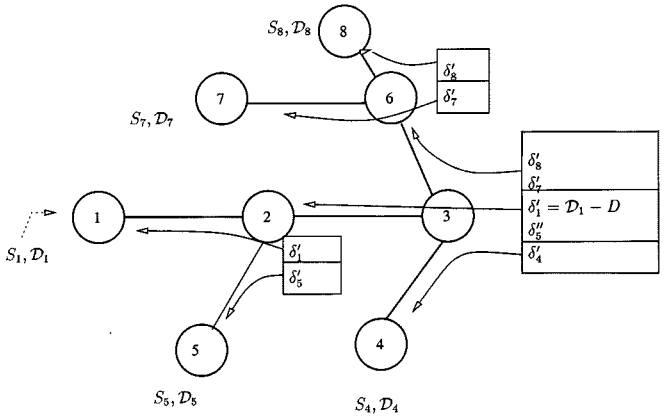


Fig. 14. Second step: updating phase.

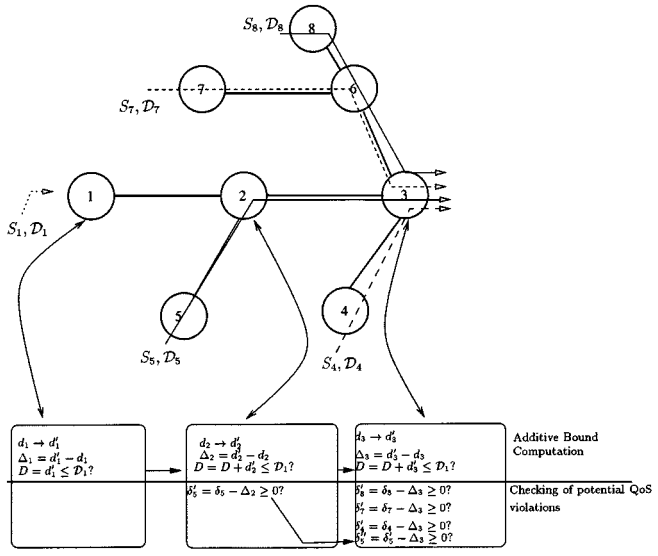


Fig. 13. First step: bound determination.

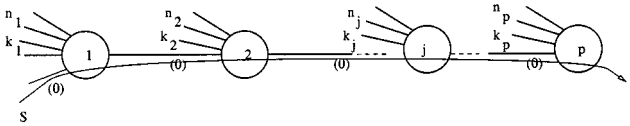


Fig. 15. Reference configuration for estimating the complexity.

6) D represents the value of the additive bound along the considered path at the consider server. It becomes equal to the additive bound at the root server.

We assume that sources S_4, S_5, S_7, and S_8 are already set up and consider the admission of S_1. There are two steps in the algorithm. The first one (Fig. 13) corresponds to the computation of the additive bound. It begins at server 1 and moves down to server 3. At each server, the algorithm tries to detect any QoS violation for the already established sessions as well as for the new source. The second step (once S_1 is accepted—Fig. 14) corresponds to the updates of the safety margins of all sources.

D. Complexity

To study the complexity of the admission control algorithm presented above, we evaluate its storage requirement and the amount of data to transfer. We study successively the two main phases of the algorithm, namely "Computation of the bound" and "Updates".

Let us consider a source S that traverses p servers (see Fig. 15). We adopt the following conventions.

1) The index of the interface where S enters at each server is denoted as (0).
2) For each server j, we define a pair (n_j, k_j), where n_j is the number of sources entering at server j by an interface other than (0) and k_j is the number of servers belonging to the paths of these sources (each server is counted only once).
3) $N = \sum_{j=1}^{p} n_j$ is the total number of active sources in the network and $K = \sum_{j=1}^{p} k_j$ is the total number of active servers.

1) Phase 1: Bound Computation:

b) Storage Requirement Q: Each server stores the arrival curve for the incoming flow and the safety margins of each source that it serves. An arrival curve is stored as a list of points. Considering a single server with n Input sources, the number of points to store is upper bounded by $n + 1$ since each greedy source adds one point corresponding to the time where its emission rate decreases from its peak rate to its mean rate and the server adds one point corresponding to the time instant where it clears the backlog. Applied to server j of Fig. 15, we obtain that the total number of points of the arrival curve for the incoming flow is upper bounded by the sum of the number of sources crossing this server and the number of servers that have already treated these sources. For each point, the total amount of data to store is constant (λ). The storage capacity required to store the arrival curve of the input flow is:

1) at server 1: $\lambda((n_1 + p_1 + \underbrace{1}_{S}) + \underbrace{1}_{\text{server}})$;
2) at server 2: $\lambda((n_1 + 1 + p_1 + 1) + (n_2 + p_2))$;
3) …;
4) at server p, $\lambda\left(\sum_{j=1}^{p}(n_j + k_j + 1) + 1\right)$.

The total amount Q_1 of memory required to store the arrival curves can be upper bounded by p times the amount of data required at server p. We obtain

$$Q_1 \leq p\lambda \left(\sum_{j=1}^{p} (n_j + k_j + 1) + 1 \right) \leq \lambda p(N + K + p + 1). \tag{19}$$

Thus

$$Q_1 = O(p(N + K + p)). \tag{20}$$

As for safety margins, each server stores the safety margins of all the sources that is serves. Let γ be the size of the memory used to store a safety margin. Then, server 1 has to allocate a memory of size $(n_1 + 1)\gamma$, server 2 has to allocate $(n_1 + 1 + n_2)\gamma$ and server p, $\left(\sum_{j=1}^{p} n_j + 1\right)\gamma$. The total amount Q_2 of memory required to store the safety margins may be upper bounded by p times the amount of memory required at server p. We obtain

$$Q_2 \leq p\gamma \left(\sum_{j=1}^{p} n_j + 1 \right) \leq \gamma p(N + 1). \tag{21}$$

Thus

$$Q_2 = O(pN). \tag{22}$$

Eventually, we obtain

$$Q = Q_1 + Q_2 = O(p(N + K + p)). \tag{23}$$

c) Amount of Transmitted Data X_1: Each server provides its neighbors with its output arrival curve and its set of safety margins. As a consequence, the amount X_1 of data to transfer is of the same order of magnitude as Q.

2) Phase 2: Updates: In the second phase of the admission control algorithm (when a new session is accepted), the root server provides each server with the new values of the safety margins of the sources that it serves. There are N safety margins and a given server must at most transmit all these safety margins. Since there are K servers, the total amount of data to transfer X_2 in the second phase is such that

$$X_2 = O(KN). \tag{24}$$

E. Discussion

Q, X_1 and X_2 depend on the number of sources, the number of servers and the length of the path of S. To provide orders of magnitude, helping at discussing complexity issues, we use the following assumptions:

1) the length of the path is equal to the mean length of a path in a binary tree network, i.e., $p \sim \log_2 K$;
2) the network is dimensioned so that the number of servers is proportional to the number of sources, i.e., $K \sim O(N)$.

With the two above assumptions, (23) and (24) become $Q = O(N \log_2 N)$ and $X_1 + X_2 = O(N^2)$. Thus, when the number of sources is increased by a factor of two, the amount of data to be transferred increases by a factor of four. This nonlinearity indicates that the admission algorithm may not scale well. A detailed analysis of the algorithm indicates that while the first phase of the algorithm (bound computation) is computationally intensive, the lack of scalability is mainly due to the second phase (updates). Indeed, this second phase results in a flooding of the network so as to ensure the exactness of the admission algorithm. However, we should keep in mind that the concentration of the traffic at the edge servers results from their role as interfaces between backbones of different ISPs. It is not due to the m2p architecture. The m2p architecture is used to reduce the cost in terms of number of connections (or LSPs) required to cover the network ($O(n)$ rather than $O(n^2)$).

We are now at the point where we can provide some guidelines for the design of an operational admission control algorithm for our traffic management scheme.

- If the backbone has a moderate size or, more precisely, if there is a moderate number of edge routers, then a centralized solution is a good option. A dedicated server, connected to all the edge routers via an m2p LSP, acts as an admission control server, the so-called bandwidth broker in the DiffServ terminology [17], [18]. The bandwidth broker must have a complete knowledge of the paths of each source (and thus the topology of the networks) with their characteristics and their safety margins. Note that the bandwidth broker only interacts with edge routers, not with interior routers (but it must keep track of the changes in the topology, which can be done through the routing protocol for instance).
- If the backbone is large, a distributed algorithm should be used. However, in this case, some additional means must be used to guarantee the scalability of the traffic management scheme. Note that in the distributed case, not only edge routers but also all interior routers are engaged in the admission control procedure. A way to ensure scalability could be to minimize the frequency of execution of the admission control algorithm. This may be achieved with an adequate grouping of sessions at the ingress servers. For instance, the set of sources issued by an other ISP with the same QoS constraint, can be grouped. This could be done by the bandwidth broker of a given domain or by the clients of an ISP who would rent VBR trunks.

VIII. Conclusion and Outlook

Traffic engineering is getting more and more important with the emergence of applications with QoS constraints and potentially a highly varying emission rate. Current routing algorithms do not take QoS constraints into account while ISPs need to have more control over the routes followed by packets in their network. A mixed solution that combines routing and forwarding, as proposed by MPLS, is very appealing. In the context of MPLS, the m2p architecture is a key architecture. In this paper, we have discussed the fundamental problem of designing a complete traffic management scheme for multimedia applications and for m2p networks. The first problem is to obtain an accurate upper bound on the end-to-end delay in an m2p archi-

tecture. A bounding approach is required as demonstrated in the study of a tandem m2p network. Therefore, we introduce the concept of additivity. A path in an m2p network is additive if its maximum end-to-end delay is equal to the sum of the local maximum delays. We show that there exists a whole class of m2p networks that are additive. For the most intricate case of nonadditive networks, we show that the additive bound represents an accurate approximation of the maximum end-to-end delay.

We next propose two admission control algorithms based on the additive bound. The first algorithm is a centralized one, the second algorithm is a distributed one. We discuss the key aspects of the two algorithms and especially their complexity and their scalability. This enables us to provide some guidelines concerning the design of a complete traffic management scheme. The choice of a distributed or centralized version depends heavily on the number of routers (edge routers and interior routers) in the backbone. All in all, it seems that a centralized version with an admission control server acting as the so-called bandwidth broker in DiffServ, is an appealing solution.

Future work should concentrate on practical experiments to compare the centralized and distributed versions. Another important research issue is the use of multiple m2p LSPs among a pair of ingress/egress routers. This method allows to achieve reliability and load balancing [10]. Our traffic management scheme could be extended to the multiple m2p LSP case with each m2p LSP representing a given class of service.

REFERENCES

[1] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF, RFC 3031, Jan. 2001.

[2] B. Davie, J. Lawrence, and K. McCloghrie, "MPLS using LDP and ATM vc switching," IETF, RFC 3035, Jan. 2001.

[3] A. Conta, P. Doolan, and A. Malis, " Use of label switching on frame relay networks specification," IETF, RFC 3034, Jan. 2001.

[4] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Networking*, vol. 1, pp. 344–357, June 1993.

[5] ——, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, pp. 137–150, Apr. 1994.

[6] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Trans. Inform. Theory*, vol. 37, pp. 114–131, Jan. 1991.

[7] ——, "A calculus for network delay, part II: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37, pp. 132–141, Jan. 1991.

[8] ——, "Quality-of-service guarantees in virtual circuit switched networks," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1048–1056, Aug. 1995.

[9] H. Ahmed, R. Callon, A. Malis, and J. Moy, "IP switching for scalable IP services," *Proc. IEEE*, vol. 85, pp. 7984–7997, Dec. 1997.

[10] H. Saito, Y. Miyao, and M. Yoshida, "Traffic engineering using multiple multipoint-to-point LSPs," presented at the IEEE Infocom, Tel Aviv, Israel, Mar. 2000.

[11] L. Tassiulas and L. Georgiadis, "Any work-conserving policy stabilizes the ring with spatial reuse," presented at the IEEE Infocom, Toronto, Canada, June 1994.

[12] I. Chlamtac, A. Faragó, and H. Zhang, "A deterministic approach to the end-to-end analysis of packet flows in connection-oriented networks," *IEEE Trans. Networking*, vol. 6, pp. 422–431, Aug. 1998.

[13] J.-Y. Le Boudec and P. Thiran, *Network Calculus*. New York: Springer-Verlag, 2001, vol. LNCS 2050.

[14] C.-S. Chang, *Performance Guarantees in Communication Networks*. New York: Springer-Verlag, 2000.

[15] J.-Y. Le Boudec, "An application of network calculus to guaranteed service networks," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1087–1096, May 1998.

[16] G. Urvoy, G. Hébuterne, and Y. Dallery, "Delay-Constrained VBR Sources in a Network Environment," Institut National des Télécommunications, Tech. Rep. 98-10-04, 1998.

[17] S. Blake, D. Black, M. Carlson, E. Davies, and W. W. Z. Wang, "An Architecture for Differentiated Services," IETF, RFC 2475, 1998.

[18] X. Xiao and L. N. Ni, "Internet QoS: A big picture," *IEEE Network*, pp. 8–18, Mar./Apr. 1999.

Guillaume Urvoy-Keller received the Engineer Degree from the Institut National des Télécommunications, France, in 1995 and the Ph.D. degree in Computer Science from the University of Paris VI, France, in 1999.

From 1999–2000, he was an Assistant Professor at the University of Versailles, France. He is currently an Assistant Professeur at Institut Eurecom, France. His interests are in the QoS provisioning and traffic engineering for the Internet.

Gérard Hébuterne received the "Doctorat" (Ph.D.) and an "Habilitation à diriger les Recherches".

From 1973 to 1994, he was with CNET (France Telecom research lab), France. He first specialized in traffic studies in SPC switches and then participated actively in performance studies for broadband systems (ATM, FR). He is with the Institut National des Télécommunications,France, since July 1994, where he leads the Networks Department. He has specialized in traffic studies and especially overload control in telecommunications systems and broadband networks. His present work focuses on the QoS aspects in multiservices broadband networks.

Yves Dallery received the Ph.D. degree and the degree of "Habilitationà Diriger des Recherches" from the Institut National Polytechnique de Grenoble (INPG), France, in 1984 and 1989, respectively.

He is currently Professor of Manufacturing and Logistics at Ecole Centrale de Paris, France. Before that, he was Directeur de Recherche at the Centre National de la Recherche Scientifique (CNRS), France. From 1984 to 1985, he was a Postdoctoral Fellow at Harvard University (M.I.T.), Cambridge, MA. From 1991 to 1992, he was a Visiting Scientist at Massachusetts Institute of Technology, Cambridge, MA, and in 1992–1993, he was an Associate Professor of Manufacturing Engineering at Boston University, Boston, MA. His research interests are in operations management, supply chain management and stochastic models.