Transport des données et gestion de la congestion dans l'Internet de demain : du contrôle à l'anarchie

Emmanuel Lochin

Habilitation à Diriger des Recherches de
l'Institut National Polytechnique de Toulouse

Spécialité

RESEAUX INFORMATIQUES

présentée par

M. Emmanuel LOCHIN

Transport des données et gestion de la congestion dans l'Internet de demain : du contrôle à l'anarchie

Présentée et soutenue publiquement le 20 octobre 2011 devant le jury composé de

Pr. Isabelle GUÉRIN-LASSOUS	**Rapporteur**
Pr. Guy LEDUC	**Rapporteur**
Pr. Jean-Jacques PANSIOT	**Rapporteur**
Pr. Michel DIAZ	**Examinateur**
Pr. Christophe CHASSOT	**Examinateur**
Pr. Patrick SÉNAC	**Correspondant**

Version revue et corrigée du 4 octobre 2011

Résumé

Le monde du transport est en pleine révolution conceptuelle. Bien qu'ayant assisté au cours de ces dernières années à la naissance de nouveaux contrôles de congestion, dont certains ont été déployés de façon virale, l'universalité d'une solution protocolaire n'existe toujours pas. Plusieurs travaux s'accordent sur le fait que la couche transport a grandement besoin d'évoluer et les nouveaux défis engendrés par l'évolution des réseaux et usages de l'Internet ont pris le pas sur le problème originel de l'effondrement des ressources que devait prévenir le contrôle de congestion de la couche transport. Certains remettent d'ailleurs en question le découpage du modèle OSI et proposent de restructurer complétement la couche transport ; d'autres soulignent l'obsolescence du concept d'effondrement du réseau ou considèrent l'équité comme un critère irréaliste et impraticable.

C'est dans ce contexte que cette habilitation explore plusieurs solutions protocolaires pour la couche transport, principalement pour le monde du *best-effort*, en partant des réseaux à QoS jusqu'aux réseaux anarchiques. Le principal problème de la couche transport n'est pas qu'elle doit évoluer : c'est comment la faire évoluer, c'est-à-dire, sans avoir à convaincre les concepteurs de systèmes d'exploitation. Aussi, nous étudions des approches qui ne rentreraient pas sous la tutelle de ces derniers et qui resteraient, autant que possible, indépendantes d'un déploiement de mécanisme collaboratif dans le cœur du réseau.

Mots-clés

Nouveaux protocoles de transport, évolution de la couche transport, contrôle de la congestion, fiabilité

Abstract

The transport protocol layer needs to evolve. Although we have witnessed these last years of the birth of new congestion controls, where some of them have been rapidely deployed, there still exists no universal transport protocol solution. Several studies agree that the transport layer must be changed and new challenges involved by new networking technologies and usage overtake the original congestion collapse problem. Even the OSI layer is now discussed and some researchers propose to re-slice this model while others emphasize the obsolescence of the congestion collapse problem or consider fairness as a criterion unrealistic and impractical.

In this habilitation thesis, we propose to explore several transport protocol solutions, mainly for the best-effort area, ranging from QoS networks to the anarchical networks. We aim at investigating approaches to change the transport layer without having to convince operating system designers or to deploy core network mechanisms.

Keywords

New transport protocols, transport layer evolution, congestion control, reliability

Table des matières

Index des collaborateurs

Chapitre 1

La couche transport a besoin d'évoluer

Le principal problème est comment faire évoluer la couche transport. Nous savons tous que la couche transport est vieillissante, depuis 1998, elle n'a guère changé. Cependant, s'attaquer à une refonte de la couche transport est synonyme de difficulté car faire du transport, c'est aussi faire de la politique. Il faut tout d'abord convaincre les bonnes personnes et leur donner les bonnes raisons de déployer un nouveau protocole. Lors d'une présentation sur le protocole SCTP au LAAS-CNRS en septembre 2009, le Pr. Paul Amer qui a activement travaillé sur ce protocole le disait de lui-même : sa principale tâche maintenant était de convaincre Microsoft de l'implémenter. On peut donc logiquement être pessimiste à la vue de la longue histoire de l'Internet et des propositions technologiques qui l'accompagne.

Que s'est-il passé depuis 1998 ? Nous ne pouvons que constater l'effort considérable d'adaptation de la couche transport au-dessus des réseaux sans-fil ; au dessus des liens satellites ; en vue d'améliorer le service des applications multimédia ou dans un objectif de garantie de service. L'ossification de cette architecture est évidente et l'Internet se trouve complètement orchestré autour du modèle du sablier (*The Houglass Model*) de Steve Bellovin. Ce modèle illustre bien la forte augmentation des services et protocoles au-dessus et en-dessous des couches réseau et transport (i.e. les bulbes du sablier) comparée au goulot d'étranglement exercé par ces mêmes couches. Cette croissance trop forte a favorisé le caractère incontournable de la pile protocolaire TCP/IP, opérant ainsi à son ossification et la rendant très difficile à évoluer alors que justement, puisque la couche transport actuelle n'implique pas le cœur du réseau, il aurait été logiquement facile d'introduire de nouvelles technologies à ce niveau. A ce jour, il est remarquable que l'architecture de l'Internet actuelle complètement garrottée autour de TCP/IP est loin d'être optimale.

L'universalité d'une solution protocolaire n'existe toujours pas et les nouveaux défis engendrés par les architectures *store and forward* des réseaux DTN ou les performances des réseaux haut débit ont pris le pas sur le problème originel que devait résoudre la couche transport : l'effondrement du réseau (*congestion collapse*). Ce même problème historique est de plus maintenant contesté. De récents travaux démontrent que cet effondrement ne serait

plus possible aujourd'hui et ce, quelque soit le protocole de transport utilisé. Parallèlement à ces visions, le monde de l'Internet s'adapte en proposant des solutions qui prennent en considération les caractéristiques des couches protocolaires basses ou hautes. D'autres remettent au gout du jour l'idée d'une définition plus abstraite de l'interface socket tandis que certaines propositions cherchent à offrir aux développeurs d'applications la possibilité de communication inter-couches *cross-layer* (possibilité encore très peu exploitée à ce jour). La trop grande séparation entre les strates du modèle OSI est devenue un problème qui incite la majeure partie des développeurs d'applications communicantes à opérer à un développement au niveau applicatif de leur propre contrôle de congestion. Ces développements non conventionnels prennent de l'ampleur et tendent maintenant à s'imposer comme nouveau standard architectural. Il est aussi remarquable que les efforts entrepris par l'IETF avec la standardisation de nouveaux protocoles de transport, tels SCTP ou DCCP pour les applications multimedia, ne semble pas inverser cette tendance. Pourtant, ces applications deviennent de plus en plus populaires et bien que l'Internet soit un réseau *best-effort* par nature, nous pouvons noter un engouement pour ces applications temps-réel telles la voix sur IP et la vidéo-conférence.

Les contradictions inhérentes à cette architecture et un goût prononcé pour le design et l'étude des protocoles de transport ont motivé à la fois les recherches présentées dans ce manuscrit et l'exploration de nouvelles pistes. Ces dernières m'ont amenées à la spécification de protocoles et mécanismes permettant le transport de l'information sur des réseaux *best-effort* et contraints. Cette orientation conduit naturellement à se pencher vers la littérature du contrôle de la congestion, de la qualité de service, du *cross-layer*, des refontes architecturales de la couche transport, des nouvelles propositions de gestion de la congestion, de la fiabilité et des applications temps-réel (comme la VoIP), multimedia et de transfert de fichiers. La tâche est complexe mais elle n'est pas insolvable. Sachant qu'il est très difficile de faire évoluer le monde du transport en collaborant directement avec le réseau (ce qui aurait dû être tout à fait légitime), nous explorerons des approches qui ne rentreraient pas sous la tutelle des concepteurs des systèmes d'exploitations et qui resteraient, autant que possible, indépendantes d'un déploiement de mécanisme collaboratif dans le cœur du réseau.

Afin de mieux présenter la logique du chemin parcouru, ce manuscrit fait le lien entre les recherches pour lesquelles j'ai été sollicité ; mes directions personnelles ; Chameleon et Tetrys qui sont deux contributions protocolaires de ce manuscrit ainsi que des propositions de gestion de file d'attente de routeur (AQM) visant à améliorer le service *best-effort* de l'Internet.

Nous démarrons ce mémoire par un rapide état de l'art des évolutions du monde du transport. Avant de présenter les autres études de ce manuscrit qui sont proposées pour les réseaux *best-effort*, nous présenterons un protocole de transport pour réseaux à qualité de service. Au contraire de TCP dont l'objectif est antagoniste à ces réseaux, nous verrons que ce protocole permet de réellement bénéficier des garanties offertes par un réseau à qualité de service. Nous passerons alors au contexte des réseaux *best-effort* et présenterons des

propositions de modification du contrôle de congestion TFRC permettant de mieux prendre en compte les besoins applicatifs. Suite à ces propositions et notamment, à la présentation du protocole de transport Chameleon issue de ces recherches, nous proposons de ré-architecturer la couche transport en décorrélant la détection de la congestion du protocole TCP, l'objectif étant de simplifier l'ajout de futures extensions. Il n'est pas possible de réfléchir à une approche bout-en-bout en s'affranchissant complètement du cœur du réseau. Aussi, le chapitre qui suivra présentera trois contributions déployées dans le cœur du réseau dont une ayant pour objectif d'améliorer l'information retournée par le cœur du réseau à la couche transport. L'avant-dernière contribution de ce mémoire est un mécanisme de fiabilité (Tetrys) qui peut être implémenté au sein d'un protocole de transport ou être utilisé au niveau applicatif. Dans une optique plus prospective, nous passerons alors des réseaux *best-effort* au réseaux anarchiques qui sont caractérisés par une absence totale de contrôle de congestion et nous discuterons comment le mécanisme de fiabilité Tetrys pourrait permettre de déployer ce nouveau concept. Enfin, nous concluons ce mémoire et proposons d'autres perspectives de recherches à ces travaux.

Pour donner au lecteur une meilleure vision des contributions protocolaires de bout-en-bout présentées dans ce mémoire, la figure 1.1 présente une vue globale de ces dernières.

Les idées développées dans chaque chapitre apparaissent par ordre chronologique et l'ensemble du document tente de respecter cette contrainte. En effet, mes différents postes et expériences de recherches ont implicitement influencé les directions choisies. Parsemés au sein du manuscrit se trouvent de courts extraits choisis de discussions provenant principalement de *mailing-lists* IETF. Il est toujours utile de confronter ses idées aux discussions qui y sont énoncées. Je me souviens d'une présentation de Fouad A. Tobagi à IFIP MedHocNet en 2003 lorsque j'étais encore thésard, qui montrait d'une manière très habile que bien des problèmes formulés dans le domaine des réseaux sans-fil avaient déjà été formalisés vingt ans plus tôt par des chercheurs en *electrical engineering*. Cette présentation a été une leçon pour moi car il est très facile, même encore aujourd'hui, de se tromper de voie en cherchant à résoudre des problèmes déjà abordés. Aussi, je considère cette lecture comme un garde-fou permettant de vérifier si le chemin entrepris est toujours le bon. En effet, la pyramide des âges des contributeurs de ces *mailing-lists* possède une ordonnée très haute permettant la confrontation et le débat entre anciennes et nouvelles visions. Sans rentrer dans toutes mes notes, j'ai donc choisi de citer quelques extraits en version originale, qui m'ont conforté ou influencé dans mes directions.

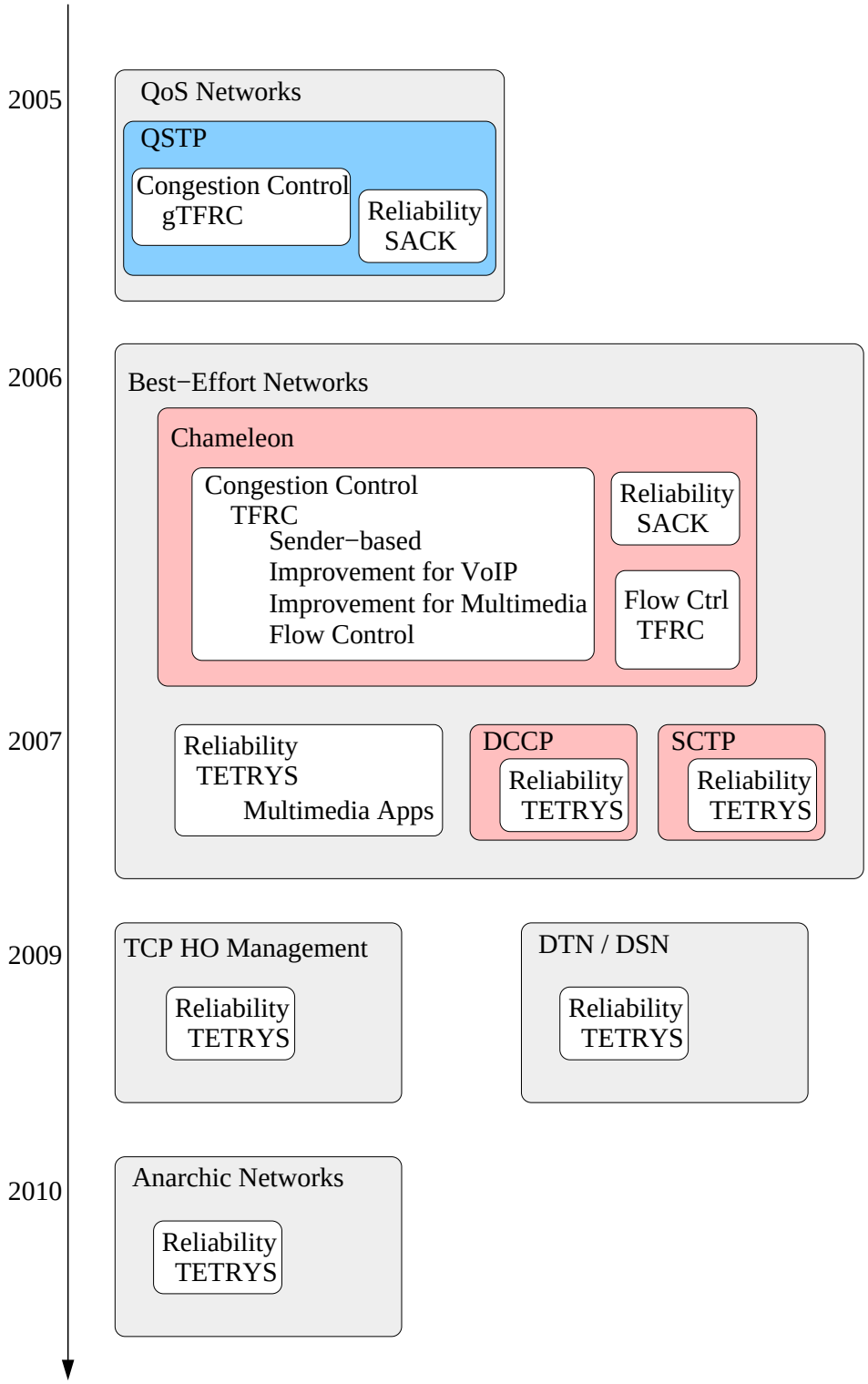


FIGURE 1.1 – Vue d'ensemble des contributions protocolaires de bout-en-bout.

Remerciements

L'ensemble des contributions présentées dans ce manuscrit n'auraient pas abouti sans l'aide de brillants thésards et collaborateurs avec lesquels j'ai eu le privilège de travailler. Je les remercie et leur dois en partie ce manuscrit. Une liste non exhaustive, car correspondante aux contributions présentées dans ce document, est donnée en page 6.

Chapitre 2

Une pression (r)evolutionnaire (un rapide état de l'art)

Le monde du transport accuse une pression évolutionnaire très forte principalement due à la large diversité des propriétés réseaux identifiées aujourd'hui. Parmi les plus actuelles, les réseaux très haut débit soulèvent de nombreux problèmes de performances en termes de convergence vers la capacité maximale tandis que les réseaux tolérants aux délais sont à la source de multiples questions autour du routage et de la communication de bout en bout [1]. Cette tendance est aussi très visible à l'IETF qui lance des réflexions sur la façon dont de nouvelles propositions de contrôle de congestion (voir `[tsvwg]`) devraient être validées ainsi que sur la plus récente liste `[tae]` (Transport Architecture Evolution) qui discute d'idées et de problèmes relatifs à l'évolution au moyen et long terme de l'architecture de la couche transport.

Comme illustration des importantes (r)évolutions proposées, il est à noter : (1) la remise en question du modèle de l'OSI ; (2) la remise en question du contrôle de congestion proposé par Van Jacobson ; (3) la restructuration de la couche transport.

En ce qui concerne le premier point, Bryan Ford dans [2] montre que la couche transport devrait être découpée en trois sous-couches distinctes afin de mieux répondre aux nouvelles caractéristiques réseaux et répondre aux besoins applicatifs. Ce dernier avait notamment proposé dans une précédente contribution [4] un modèle d'architecture protocolaire pouvant fonctionner soit dans l'espace utilisateur soit au niveau noyau. Bien évidemment, ceci renforce la tendance déjà présente au sein d'applications pair à pair, de développer au dessus d'UDP des contrôles de congestion dans l'espace utilisateur. Un très bon exemple du succès du déploiement d'un protocole au niveau applicatif est RTMFP [2] de la société Adobe. Matthew Kaufman and Michael Thornburgh, tous deux Senior Computer Scientists

1. Aujourd'hui il existe des réseaux DTN connectés à l'Internet. Voir également le plaidoyer de Jon Crowcroft sur leur utilité [7]
2. Voir `http://cc.rtmfp.net/`

chez Adobe, ont réussi la prouesse de développer et de déployer de façon complètement transparente, deux versions de protocoles de transport pour les logiciels de la marque pour les systèmes pair à pair utilisant Flash Player 10 [3]. Nous pourrions discuter si ce choix de développement aurait eu lieu si l'offre protocolaire des concepteurs de systèmes d'exploitations était suffisamment en adéquation avec les besoins des développeurs d'applications. Un mécanisme protocolaire non standardisé pourrait se révéler préjudiciable pour l'équilibre de l'Internet. C'est une des inquiétudes de l'IETF qui cherche à éviter que l'Internet se peuple de flots UDP pseudo-contrôlés. Aussi, en proposant le protocole DCCP, l'IETF a tenté de répondre à la demande croissante des développeurs d'applications multimedia qui ne trouvent pas en TCP (à cause de son caractère fiable) ni en UDP (à cause de sa non-adaptitivité à la bande passante disponible) le protocole adéquat. C'est peut-être cette absence qui a conduit au développement de protocoles de congestion adaptatifs dans l'espace utilisateur. Malheureusement, le fossé qui sépare l'IETF de la pile protocolaire d'un système est immense. On peut aujourd'hui dire que l'avènement de la couche IPv6 au sein du système Microsoft résulte de l'influence de Christian Huitema. *Quid* si Eddie Kohler avait travaillé pour Microsoft ?

Certains pensent cependant que ces développements dans l'espace utilisateur ne seraient pas un problème. C'est le cas de Tom Phelan, concepteur d'une librairie DCCP et qui propose d'encapsuler ce protocole dans UDP pour faciliter son utilisation et contrecarrer les choix dominants en termes protocolaires des systèmes d'exploitation non ouverts [4]. Il existe aussi d'autres exemples tels SPDY [5], développé par Google, qui propose un protocol experimental présenté comme une couche session (en-dessous de HTTP et au dessus de la couche présentation) visant à accélérer le trafic web.

Pour le second point, Alexandre Snoeren dans *Decongestion Control* [5], argumente qu'il ne serait pas nécessaire de garder un réseau non-congestionné afin d'obtenir de bonnes performances en termes de débit et d'équité. Selon [5], les protocoles gourmands (*greedy protocol*) ont le potentiel d'atteindre de meilleures performances que TCP, tout en garantissant une protection contre les terminaux opportunistes et illégitimes (au sens du déni de service) et tout en évitant la saturation des files d'attentes des routeurs. Plus récemment, Thomas Bonald dans *Is the "Law of the Jungle" sustainable for the Internet ?* [6] démontre cette conjecture et illustre que l'absence de contrôle de congestion ne résulterait pas en un effondrement de l'Internet. En effet, certains codes à effacement dit "parfait" pourraient très bien remplacer avantageusement un contrôle de congestion. Un an avant cette proposition, le monde du Network Coding s'était invité dans le monde des protocoles de transport [8], [9]. C'est notamment sur cette mouvance que nous avons proposé avec Jérôme Lacan le protocole Tetrys. Il est tout de même à noter que ce concept de décongestion est tout à fait favorable aux opérateurs de télécommunications. Comme l'avait une fois souligné Bob Briscoe sur la mailing-list

3. Voir `http://www.ietf.org/proceedings/10mar/slides/tsvarea-1.pdf`
4. Voir `http://www.ietf.org/id/draft-ietf-dccp-udpencap-00.txt`
5. Voir `http://www.chromium.org/spdy`

[tsvwg] en argumentant son texte sur le mythe de l'allocation des ressources [10], un utilisateur *greedy* occupant plus de ressources que nécessaires n'a jamais été un problème *pour lui*. Dans un monde où la quantité d'informations a un prix et est facturable, tout cela n'est qu'au bénéfice de *son* employeur (*i.e.* British Telecom).

Enfin, le dernier point propose de modifier l'interface socket afin de laisser un libre choix aux applications du protocole de transport à utiliser. Une idée possible est de redéfinir l'API socket telle une classe abstraite [1], voire de mettre en œuvre un méta protocole de négociation lors de l'établissement de la connexion [3]. L'objectif étant avant tout de proposer une alternative plus élégante à celle de l'encapsulation tout UDP.

De nouvelles approches connues sous le nom d'Explicit Router Notification (ERN protocols) proposent une collaboration conjointe entre le réseau et les terminaux afin de déterminer le débit optimal d'émission auquel le protocole de transport peut prétendre afin de maximiser l'utilisation de la bande passante des réseaux très haut débit. Le protocole XCP proposé par Dina Katabi ou Quickstart (RFC 4782) par Sally Floyd font tous deux partie de cette classe de protocole.

Cependant, l'architecture ERN du plan de contrôle est très lourde et mis à part un déploiement dans le cadre de réseau privé comme les grilles de calculs ou bornés par des proxies, nous pourrions penser que ces propositions connaitront un développement et un succès similaires à celles des architectures DiffServ et IntServ (à moins qu'une approche incrémentale ne soit possible). En effet, si l'on regarde par le passé les architectures à QoS telle l'architecture DiffServ, nous pouvons parler d'échec quant à son déploiement dans l'Internet et même d'échec conceptuel en ce qui concerne la classe DiffServ/AF (cette affirmation sera discutée au chapitre 3). Là n'est cependant pas la seule raison : la complexité du déploiement de cette architecture, bien que sans doute utilisée au sein de certains réseaux d'opérateurs, est aussi un facteur qui a pesé dans la balance. Cependant, et au-delà des débats sur la facturation des usagers et de l'Internet à deux vitesses que légitimement réfutent les partisans de la *Net Neutrality*[6], les idées véhiculées par cette architecture restent toujours d'actualité car elles cherchent intrinsèquement à améliorer les performances des protocoles de transport de l'Internet. De façon surprenante, suite au regain d'intérêt des industriels pour les constellations satellites, l'architecture DiffServ semble revenir au premier plan.

6. Voir `http://commerce.senate.gov/pdf/cerf-020706.pdf`

Références sur le chapitre

[1] Michael Welzl, Stefan Jörer, Stein Gjessing, Towards a Protocol-Independent Internet Transport API, FutureNet IV workshop in conjunction with of IEEE ICC 2011, Kyoto, Japan, 5-9 June 2011

[2] Bryan Ford and Janardhan Iyengar, Breaking Up the Transport Logjam, In the Seventh ACM Workshop on SIGCOMM Hot Topics in Networks (HotNets-VII), Calgary, Alberta, Canada October 6-7, 2008

[3] Bryan Ford and Janardhan Iyengar, Efficient Cross-Layer Negotiation, In the Eighth ACM Workshop on SIGCOMM Hot Topics in Networks (HotNets-VIII), New York City, NY, USA, October 22-23, 2009

[4] Bryan Ford, Structured Streams : a New Transport Abstraction, In Proceedings of ACM SIGCOMM 2007

[5] Barath Raghavan and Alex Snoeren, Decongestion Control, ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-V), 2006

[6] Thomas Bonald, Matthieu Feuillet and Alexandre Proutière, Is the "Law of the Jungle" sustainable for the Internet ?, In Proceedings of INFOCOM, 2009

[7] J. Crowcroft, E. Yoneki, P. Hui, T. Henderson, Promoting Tolerance for Delay Tolerant Network Research, ACM Computer Communication Review, 38(5) :6368, October 2008

[8] Jay Kumar Sundararajan and Devavrat Shah and Muriel Médard, ARQ for Network Coding, In Proceedings of IEEE ISIT, 2008

[9] Network Coding Meets TCP, Jay Kumar Sundararajan, Devavrat Shah, Muriel Medard, Michael Mitzenmacher, Joao Barros, In Proceedings of IEEE Infocom, 2009

[10] Bob Briscoe, Flow Rate Fairness : Dismantling a Religion, ACM Computer Communications Review, 37 (2), 2007

Can we deploy end-to-end protocols ?

Extrait intéressant et inter-générationel à propos des approches bout en bout et clean slate, un fil de discussion entre un jeune doctorant de l'université de Cornell : Mahesh Balakrishnan[7] *et Christian Huitema. La réponse de Manesh dépeint exactement l'Internet sur lequel les gens de ma génération ont commencé leur recherche. Il n'y a pas eu de contre argumentation.*

M. Balakrishnan : In fact there seems to be pushback from both ends — we can't deploy end-to-end protocols because major companies own the end-host stacks ; and we can't push mechanisms deep into the network because ISPs and router companies own the network. Arguably the latter source of pushback played a major role in the emergence of the e2e philosophy ; but now we have equally powerful commercial forces on the other side.

So effectively the only practical mode of deployment seems to be the 'almost' end-to-end middlebox — one hop away from the end-host but not quite into the network (and the Maelstrom work I presented day before yesterday at NSDI would be one example).

C. Huitema : In what world are you living, exactly ? What about systems like Skype, or Bit torrent ? They are definitely pushing the envelope of end to end designs, are widely deployed, and are not controlled by major corporations.

M. Balakrishnan : In a world where clean-slate designs are impractical, apparently. I do belong to a generation of researchers who grew up on P2P and overlays — and who accept the presence of IP and TCP as religious imperatives that must be engineered around. If industry realities confine us to particular philosophies of system design we can stop arguing about their relative merits and simply work with whatever is most practically deployable. Depending on your point of view this is either an excellent thing or the end of pure research as we know it. But I have to admit I don't understand how people can talk of designing the next-generation Internet from scratch when the economics of computer systems are so stacked against any kind of ground-up innovation.

Also - if the router/ISP companies pushed us to the endpoints of the network and now the software companies have pushed us up the software stack — where do we go next ?

7. `http://www.cs.cornell.edu/~mahesh/`

Chapitre 3

Un petit détour par la QoS

Revenons aux architectures QoS qui ont occupé une bonne partie de ma thèse et de mes premiers pas en recherches et en particulier le projet EUQoS. Concernant le concept lui-même, bien que les garanties de la classe temps-réel aient largement été éprouvées par de nombreux modèles déterministes permettant d'obtenir des bornes concrètes de délai, le problème soulevé par la classe élastique concernant l'assurance d'un débit moyen de transfert de données n'a jamais vraiment été résolu. Face aux nombreuses propositions et la littérature prolifique dans ce domaine, fort est de constater qu'aucune solution n'a permis de résoudre la garantie de débit de TCP si chère à la classe AF. Dans un article faisant un bilan de cette recherche qui était le principal sujet de ma thèse en 2004, j'en conclus qu'asservir TCP n'était pas le meilleur chemin et que la solution aurait du venir de la définition d'un protocole de transport fiable QoS compatible [1].

Comme si bien souligné dans le paragraphe suivant extrait de «QoS's Downfall : At the bottom, or not at all ! »[10], tandis que Intserv pouvait être considéré comme *too much, too early*, Diffserv était *too little, too late* :

> *Differentiated services was an attempt to scale down the complexity of Integrated services - but chasing an exponential curve with a linear optimization is bound to fail. Diffserv was too little, too late. Among other problems, it never addressed issues of provider cooperation ; many tier-1 ISPs asked «needs it anyway in the current bandwidth glut ? ».*

Une solution arrivera plus (trop [1]) tard lorsque je me trouvais à NICTA, engagé dans le projet EUQoS [2] qui avait pour objectif la réalisation d'un *backbone* européen à QoS opérationnel entre des réseaux scientifiques et industriels et notamment de mettre en place une plateforme d'*e-learning*, de télémedicine et de diffusion multimedia (WebTV).

Encore un n-ième clône de vaste projet européen sur la QoS ? Pas exactement, la grande différence de ce projet était qu'il proposait des applications concrètes et la volonté de fonda-

1. L'adverbe est relatif à l'essoufflement indéniable de l'intérêt de la communauté réseaux pour les architectures à QoS lourdes. A l'exception peut être de certaines branches de l'IETF.

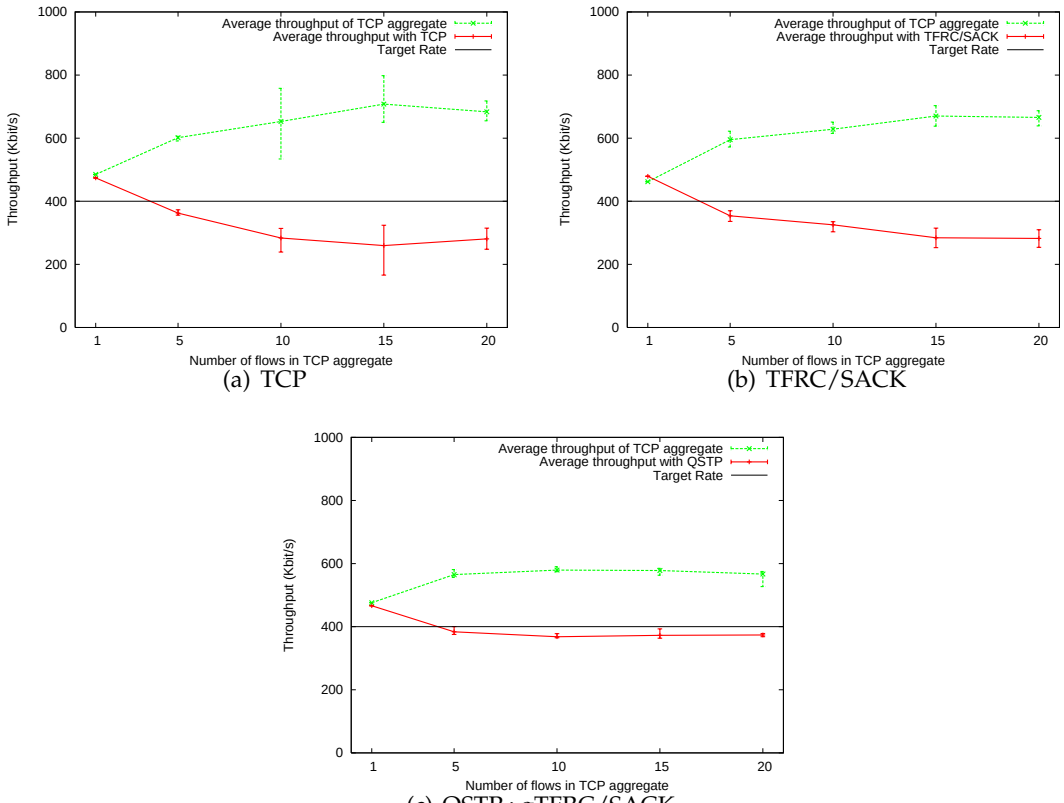
mentalement implémenter et prouver l'utilité industrielle du concept. Plus particulièrement, ce projet s'attelait à étudier la signalisation multidomaine de la QoS qui fût parent pauvre et aurait du être une des premières questions à solutionner selon [10].

Tango Charlie Papa do you copy ?

Durant cette période et toujours dans le cadre du projet EUQoS, nous avons travaillé avec Guillaume Jourjon et Laurent Dairaine sur un contrôle de congestion *QoS-aware* basé sur le protocole TFRC (précédemment RFC 3448 maintenant spécifié comme protocole dans RFC 5348) nommé gTFRC [4]. Le protocole TFRC étant un protocole basé sur un taux d'émission et non pas une fenêtre d'émission comme pour TCP, ce contrôle de congestion permettait enfin une communication entre l'application et la couche transport qui ne renvoyait plus un nombre de paquets par RTT comme unité d'émission. Evidemment, cela ouvrait la porte au design d'un protocole de transport compatible DiffServ. En effet, pourquoi s'entêter à arraisonner TCP sur un réseau avec lequel il ne parlera jamais le même langage et puisque l'architeture DiffServ se cantonnera à des réseaux privés, pourquoi ne pas utiliser un protocole spécifiquement adapté (voir notamment [9] qui fait le même constat), quitte à utiliser des proxies (ou l'approche de multiple contrôle de congestion de Bryan Ford) pour faire le lien entre réseau *best-effort* et réseau à QoS ? C'est ce que nous avons proposé avec gTFRC qui très simplement, permet à la couche transport via une option de l'API socket de différencier la partie du trafic garanti (dite *assured* ou *green marked* : RFC 2475 et 2597) de la partie non-garantie (dite *best-effort* ou *red marked*). Ainsi, la partie assurée n'était plus sous la coupe du contrôle de congestion et s'apparentait à un trafic CBR tandis que la partie non-garantie restait contrôlée et variait suivant la disponibilité de bande passante du réseau.

L'idée, très simple, consiste en l'ajout d'un maximum au calcul du débit retourné par TFRC. Lorsque l'équation de TFRC renvoie un débit d'émission inférieur au débit cible requis par l'application et normalement profilé par un marqueur type *token bucket* à deux couleurs, gTFRC retient alors comme débit minimum d'émission le débit cible.

Afin d'illustrer le résultat obtenu, nous présentons sur la figure 3.1 le débit utile moyen d'un flot en compétition avec un agrégat TCP de taille variable cherchant à atteindre un débit cible de 400Kbit/s. Le calcul du débit correspond à une moyenne effectuée sur dix expérimentations après 150 secondes de test. Nous donnons également sur cette figure les valeurs min et max obtenues. La topologie utilisée est de type papillon et le goulot d'étranglement est de 1 Mbit/s. En ce qui concerne TCP, sans surprise et comme largement montré dans de nombreuses contributions, le flot seul n'arrive pas à obtenir sa partie garantie (figure 3.1(a)). Nous remarquons que la combinaison de TFRC et SACK obtient un résultat du même ordre de grandeur mais que son caractère moins oscillatoire se retrouve dans le faible écart des valeurs min et max (figure 3.1(b)). En revanche, notre proposition (figure 3.1(c)) permet au flot d'obtenir le débit garanti. A noter que la légère différence entre le débit utile et le



(a) TCP (b) TFRC/SACK

(c) QSTP : gTFRC/SACK

FIGURE 3.1 – Débit utile moyen d'un flot en compétition avec un agrégat TCP de taille variable.

débit cible provient de l'encapsulation protocolaire et du coût de traitement du prototype au niveau applicatif, ce protocole ayant été implémenté dans l'espace utilisateur.

QSTP : a QoS-aware Tranport Protocol

Ce contrôle de congestion sera finalisé au sein d'un protocole de transport avec le concours de Sébastien Ardon et Patrick Sénac par l'adjonction d'un mécanisme de fiabilité (SACK) et d'un contrôle de flot tous deux compatibles *rate-based* afin de proposer un protocole de communication opérationnel complet *QoS-aware*. Ce protocole baptisé QSTP (*QoS-aware Tranport Protocol*), bien que tardif sur l'échelle de la recherche en QoS, pouvait être considéré comme, à notre meilleure connaissance, le premier protocole de transport DiffServ compatible. Nous avons défendu le cœur du mécanisme à l'IETF et reçu un retour positif de la part des concepteurs du protocole TFRC qui globalement ne voyaient pas d'objection à l'utilisation et la normalisation d'un tel mécanisme, à condition que ce dernier se limite aux réseaux à QoS. Evidemment, l'accessibilité à un CBR contrôlé pouvait porter préjudice au comportement équitable (dit *TCP-friendly*) de TFRC dans le cadre de l'utilisation d'un tel mécanisme

sur un réseau *best-effort*. La correspondance IETF effectuée en particulier entre Sally Floyd, Mark Allman et James Polk concernant gTFRC, a donné suite à la version 02 du mois d'aout 2006 d'un draft motivant et expliquant le concept [2]. Ces travaux sont présentés dans [3] et cités comme une solution alternative pour le contrôle du trafic élastique dans la RFC6077 [8].

Leçons retenues

Les architectures à QoS ont trouvé une application au sein de certains fournisseurs d'accès Internet [5] et en particulier MPLS au niveau de la couche liaison (grâce notamment au déploiement de MPLS-TE [3]). L'architecture DiffServ ou les mécanismes inhérents à cette architecture semblent être utilisés par quelques opérateurs réseaux (voir la discussion intitulée *QoS usage* et initiée par Henning Schulzrinne sur la mailing list `[Tccc]` [4] ainsi que les présentations de déploiement de QoS chez Google sur [7]). On peut également noter que l'architecture DiffServ est revenue sur le devant de la scène, suite au regain d'intérêt des industriels pour les constellations satellites, comme solution potentielle de gestion du trafic sur ces réseaux. Cependant, le concept de la *Network Neutrality*, fermement défendu par des grands noms de la distribution de contenu de l'Internet, a rendu obsolète cette vision issue du monde des télécommunications qui voulait offrir à l'utilisateur Internet des services garantis. De plus, certains trafics (tel l'IPTV [6]) semblent ne pas avoir besoin de ce type d'architecture, la gestion de la QoS étant réalisée au sein des flots eux-mêmes par l'adjonction de redondance ou dégradation de la qualité en fonction du codage vidéo utilisé. Cette étude a cependant permis d'appréhender une vision protocolaire qui regarde d'abord du côté de l'application avant celui du réseau. Dans le chapitre suivant, nous avons exploité plus en avant cette philosophie.

2. Voir : `draft-lochin-ietf-tsvwg-gtfrc-02.txt`
3. MPLS Traffic Engineering
4. `https://lists.cs.columbia.edu/pipermail/tccc/2010-November/017388.html`

Références sur le chapitre

[1] **Emmanuel Lochin, Pascal Anelli, TCP throughput guarantee in the DiffServ Assured Forwarding service : what about the results ?, Annals of Telecommunications, 2008**

[2] **End-to-End Quality of Service Over Heterogeneous Networks, Editors : Torsten Braun, Michel Diaz, José Enríquez Gabeiras, Thomas Staub, Springer Verlag Book, 2008**

[3] **Guillaume Jourjon, Emmanuel Lochin, Patrick Sénac, Design, Implementation and Evaluation of a QoS-aware Transport Protocol, Elsevier Computer Communications, 31 (9), 2008**

[4] **Emmanuel Lochin, Laurent Dairaine, Guillaume Jourjon, gTFRC, a TCP Friendly QoS-aware Rate Control for Diffserv Assured Service, Springer Telecommunication Systems Journal, 2006**

[5] C. Bastian, T. Klieber, J. Livingood, J. Mills, R. Woundy, Comcast's Protocol-Agnostic Congestion Management System, Internet Engineering Task Force (IETF) RFC6057

[6] Simon Lockhart, Bogons, Inuk Networks Experiences of Delivering IPTV to Student Accommodation in the UK Presentation Date : October 14, 2008, 10 :00 AM - 10 :30 AM `http://nanog.org/meetings/nanog44/abstracts.php?pt=ODY3Jm5hbm9nNDQ=&nm=nanog44`

[7] Enterprise QoS Tim Chung, Google Presentation Date : June 15, 2010, 3 :30 PM - 4 :00 PM `http://nanog.org/meetings/nanog49/abstracts.php?pt=MTU2MyZuYW5vZzQ5&nm=nanog49`

[8] Dimitri Papadimitriou , Michael Welzl, Michael Scharf, Bob Briscoe, Open Research Issues in Internet Congestion Control, RFC 6077, 2011

[9] V. Firoiu, J.Y. Le Boudec, D. Towsley, Z.-L. Zhang and J.-Y. Le Boudec, Advances in Internet Quality of Service, EPFL Technical Report, 2001

[10] Crowcroft, Jon and Hand, Steven and Mortier, Richard and Roscoe, Timothy and Warfield, Andrew, QoS's downfall : at the bottom, or not at all !, ACM SIGCOMM workshop on Revisiting IP QoS, 2003

Userland protocol is a kludge ?

L'extrait qui suit illustre le schisme existant entre le pour et le contre du développement de la couche transport dans l'espace utilisateur. Laissant toute considération en matière de performances de côté, nous pouvons lire deux points de vue divergents sur la prolifération des implémentations protocolaires au niveau de l'espace utilisateur et sur la légitimité de ces développements. Je dirais que les deux arguments se valent, mais je penche plutôt pour le premier, sachant que la negociation de bout en bout n'est pas pour moi un problème insolvable dans l'espace utilisateur et que la fin de ce manuscrit s'oriente vers des propositions au niveau applicatif.

Question : Userland protocol is a kludge ?

Stuart Cheshire : Layering new transport protocols over UDP is not a hack. It's actually the right architecture AND (because of that) it also happens to work though NAT gateways too.

James Woodyatt : I'm not in agreement with that. I still think "layering new transports on UDP" is a kludge. There's a lot of "transportly stuff" that needs to be negotiated on a node-to-node basis, and only sometimes (not always) on a user-endpoint-to-user-endpoint basis, e.g. congestion avoidance, mobility, etc.

Chapitre 4

Le protocole Chameleon

Le protocole Chameleon [2] a été élaboré au NICTA comme souche protocolaire indépendante issue des travaux du projet EUQoS. Il reprend le concept de protocole alliant un mécanisme *rate-based* (TFRC) et un mécanisme de fiabilité (SACK). Ce protocole et certains mécanismes décrits plus bas sont à la base de la thèse de Guillaume Jourjon maintenant chercheur à plein temps au NICTA.

L'idée qui nous intéressait était de spécialiser notre protocole afin de permettre aux développeurs d'applications d'utiliser via l'API socket, un ensemble d'options permettant de mieux répondre aux besoins applicatifs. On peut comparer cet objectif, de multiples configurations possibles d'un protocole de transport, aux architectures protocolaires construisant une couche transport par imbrication de micro-mécanismes. Cette idée, initialement introduite par les concepteurs du xkernel [11], est à la source de nombreuses autres propositions telles UDT [12] (UDP-based Data Transfer qui utilise notamment le protocole *rate-based* RAP qui peut être considéré comme le père de TFRC) et SST [13] (Structured Stream Transport). On peut également noter d'autres contributions, qui ont proposé de définir un langage formel modélisant les caractéristiques des applications, afin de traduire leurs besoins au niveau de la couche transport et ainsi, mettre en œuvre ces micro-mécanismes de manière simple [14].

En 2007, un article de ACM CCR [7] préconisait *don't optimize existing protocols, design optimizable protocols*. C'est la philosophie que nous avons choisi de suivre avec Chameleon.

Pourquoi un protocole *rate-based* fiable ?

L'objectif de combiner un contrôle de congestion *rate-based* tel TFRC avec le mécanisme de fiabilité éprouvé comme SACK, était l'obtention d'une nouvelle classe de protocole *rate-based* fiable, voire partiellement fiable. Nous avons déjà souligné l'utilité d'un protocole de transport qui utilise la même métrique que l'application. L'autre motivation était de bénéficier de la propriété de lissage inhérente à TFRC. En effet, TFRC réagit beaucoup moins violemment que TCP à une perte isolée. Or, c'est à cause du comportement très oscillatoire de TCP sur les réseaux sans-fil, véhiculaires et multihop, qu'intuitivement nous avons cher-

ché une alternative avec un protocole basé sur un taux d'émission. Cette idée a été inspirée des travaux de Kai Chen et Klara Nahrstedt [18] [17]. Deux papiers écrits en 2004 aux titres antagonistes : l'un prônant l'utilité et l'autre les limites de l'utilisation des protocoles *rate-based* dans les réseaux MANET. Dans [18], les auteurs montrent une propriété très utile de TFRC en milieu MANET [1] : sa stabilité lors des phases de changement de débit. Grâce à la définition de plusieurs métriques, les auteurs montrent que TFRC a un comportement très conservatif sur le long et court terme, tout en maintenant un débit lisse et ce, sur une large variété de topologies MANET ayant différents niveaux de trafic perturbateur. Cependant, les performances moyennes obtenues sont parfois inférieures à TCP et les auteurs suspectent un calcul imprécis de l'estimateur du taux de perte. Ce constat est très difficile à démontrer car il suppose (1) une analyse précise de la réaction de TCP face à un ensemble d'événements similaires qui sont difficiles à isoler et (2) d'étudier comment la moyenne effectuée au sein de l'algorithme TFRC peut influencer cette estimation. En ce qui concerne la contribution [17], l'utilité démontrée est similaire à celle que nous avons déjà soulevée en termes de communication à métrique identique. Dans cette contribution, avec une approche similaire aux protocoles ERN (des routeurs MANET participent au calcul du débit d'émission de la source via une signalisation par des acquittements), les routeurs intermédiaires transportent l'information du débit disponible à chaque saut permettant à la source d'inférer sur son débit d'émission efficacement.

Comme souligné tout particulièrement dans [22], étude pionnière en ce qui concerne le comportement de TFRC sur le long terme et qui est à la source de [18], les écarts de performances constatées pour TFRC pourrait aider les ingénieurs réseaux à concevoir de nouveaux protocoles ayant des objectifs similaires à TCP. De plus, et comme déjà souligné plus haut, l'étude [18] permet d'identifier des comportements non désirés et mettre en place des campagnes d'évaluation mieux ciblées.

La composition de notre protocole est donc motivée par tous ces aspects qui nous montrent que :

- il existe des preuves des mauvaises performances de TCP en environnement sans-fil et multihop [19] [20] [16] ;
- il existe des indices suggérant la bonne adéquation des protocoles *rate-based* sur ces réseaux.

L'idée de Chameleon était également de proposer un *framework* générique basé sur un contrôle de congestion de type *rate-based*. L'utilisation de ce protocole, au sein du projet EU-QoS, avait déjà démontré que le mécanisme SACK couplé à TFRC offrait une souplesse en termes de fiabilité permettant à la fois d'obtenir une fiabilité totale ou partielle. Ce protocole, notamment les problématiques d'ingénierie relatives au design d'un contrôle de flot compatible *rate-based* et de l'inclusion du mécanisme SACK dans le système de retour de TFRC, est développé dans [2].

1. Voir notamment la section *III.C Smoothness of rate change* de cet article pour de plus amples détails

La suite des études menées s'est attachée à proposer des améliorations de ce mécanisme afin de le rendre plus performant, voire plus compétitif, que d'autres mécanismes protocolaires. En particulier, bien que DCCP-CCID3 (la version protocolaire de TFRC) a été conçu pour transporter les flots multimedia et trouver une alternative aux protocoles développés au niveau applicatif au dessus d'UDP, nous avons identifié plusieurs problèmes de performances que nous avons tenté de résoudre.

Propositions de modifications de TFRC

Le contrôle de congestion TFRC est bien évidemment le pilier du protocole Chameleon. Aussi, suite au développement du mécanisme de fiabilté similaire au mécanisme SACK et du contrôle de flot compatible avec un protocole *rate-based*, implémentés en validés sous ns-2, nous nous sommes penchés sur les performances en termes de mémoire et de calcul d'un tel protocole dans le cadre d'un prototype écrit en Java. De plus, afin de pouvoir utiliser cette souche protocolaire, nous avons cherché à minimiser et optimiser les temps de calcul et d'occupation mémoire de ce contrôle de congestion.

C'est ce que nous présenterons dans la suite de ce document. Nous parlerons tout d'abord d'optimisations système de l'algorithme TFRC proposant notamment une architecture basée sur l'émetteur : *TFRC-light*. Ensuite nous présenterons plusieurs propositions de modifications de l'algorithme interne TFRC. En particulier et suite à plusieurs mesures de performances, nous nous sommes aperçus que dans le cadre de long-délai, il serait bénéficiable d'adapter dynamiquement la fréquence des messages de retour de TFRC. Avec Roksana Boreli et Golam Sarwar, nous avons évalué grâce à une modification de la souche protocolaire DCCP-CCID3 sous GNU/Linux, les bénéfices du calcul dynamique de cette fréquence et proposé plusieurs modifications à DCCP-CCID4 afin qu'il puisse répondre au mieux aux caractéristiques des applications VoIP. En effet, bien que spécialement conçu pour être utilisé avec ce type d'applications, nos mesures montraient que DCCP-CCID4 (variante de CCID3 pour la VoIP, connue également sous le nom de "variante pour petite taille de paquet TFRC" ou TFRC-SP) obtenait des scores mediocres comparé à ceux obtenus par TCP ! Ce constat a également été effectué dans une étude sur les performances de la VoIP [23]. La logique de ces études et le cœur de ces propositions sont détaillés ci-après.

Sender-based TFRC

Nous savons que TFRC produit un débit plus stable que TCP, ce qui fait de lui un bon candidat pour le transfert multimédia et le *streaming*. Néanmoins, dans le scénario d'une communication client-serveur utilisant TFRC, si les serveurs multimédia sont de puissantes machines en matière de calcul et de débit de sortie, cela n'est pas le cas pour des clients mobiles. En effet, ces clients sont des entités aux ressources limitées qui posent le problème de l'optimisation de ces ressources, en particulier pour des tâches systèmes récurrentes et

des tâches de communication. Dans un souci d'amélioration des performances des systèmes mobiles autonomes, l'allègement de ces processus prend toute son importance.

Un des principaux coûts du mécanisme TFRC est le calcul périodique du temps aller-retour (RTT) et du taux de perte de paquet de la communication. En particulier, la RFC 3448 propose que l'estimation de ce taux de perte soit faite du côté receveur. Ce standard suggère aussi que ce calcul puisse être fait du côté émetteur.

Avec Guillaume Jourjon et Patrick Sénac , nous avons développé cette idée en spécifiant et évaluant une mise en œuvre de TFRC orienté émetteur. Dans cette proposition, le transfert fiable des paquets de contrôle est assuré par l'utilisation d'un mécanisme similaire à SACK. Ce mécanisme est reconnu pour sa robustesse lors de communications dans des canaux à pertes car cette robustesse permet d'éviter la mise en place de mécanismes de contrôle d'erreur trop complexes (voir RFC 2883). De plus, grâce à sa migration sur les serveurs de flux, l'architecture orientée émetteur proposée devient robuste face aux receveurs opportunistes et résoud un problème de sécurité identifié dans la RFC 3448. Ce problème de sécurité est lié au fait que le récepteur renvoie à l'émetteur la valeur du taux de perte de la communication. Or, dans le but de recevoir une meilleure bande passante, un récepteur mal intentionné pourrait sous-évaluer ce taux. Grâce à une architecture orienté émetteur, le serveur n'est plus dépendant de la précision et de la véracité des informations renvoyées par le receveur. D'autres solutions ont été proposées afin de sécuriser TFRC contre ces receveurs opportunistes dans [21] en utilisant RTSP (RFC 2326). Cependant, notre solution requiert moins de modifications et simplifie l'entête des messages et l'algorithme de TFRC. Une autre solution introduisant un TFRC orienté émetteur a été proposée dans DCCP-CCID3 (RFC 4342). Cette solution requiert de la part du receveur l'envoi dans les paquets de contrôle des intervalles d'évènement de pertes. À notre connaissance et au jour de l'écriture de ce manuscrit, cette solution n'a jamais été ni implémentée ni testée. De plus, comparée à notre solution, cette proposition, supposée plus proche du mécanisme original, complexifie le récepteur car il se doit de maintenir une structure permettant de différencier une perte de paquet dun évènement de perte.

Enfin, en faisant l'hypothèse du déploiement de ce protocole, une architecture orientée émetteur permettrait très facilement l'avènement de nouvelles variantes ou évolutions de TFRC. L'illustration en est faite avec le protocole TCP qui est orienté émetteur.

Les détails concernant l'étude qualitative de cette contribution et les métriques utilisées pour démontrer la fiabilité du concept ont été publiés dans [1] et étendus dans [3]. Une optimisation de l'initialisation du *loss-history* a également été proposée dans [4] où nous présentons un algorithme qui divise par un facteur de 1.6 le temps CPU [2] comparé à l'implémentation standard. Le travail de Guillaume Jourjon a également été pris en compte dans la version mise à jour de la nouvelle monture de la RFC 3448bis en section *Sender-based variant* [3]. Plus récemment, une version de TFRC *sender-based* reprenant notre architecture a été intégrée au protocole de communication DTN Saratoga [10].

2. Calculé suivant les caractéristiques du Pentium IV de chez Intel
3. Voir `http://tools.ietf.org/html/draft-ietf-dccp-rfc3448bis-01`

TFRC pour la voix sur IP

Le satellite est un moyen efficace de communication pour les vastes pays que sont les Etats-Unis, le Canada ou l'Australie. En effet, le déploiement d'architectures sans-fil sur de telles contrées peut avoir un coût prohibitif surtout dans les endroits très peu peuplés. C'est aussi le cas en France, dans l'Ariège et la Creuse, où l'Europe finance le déploiement d'antennes satellite pour diminuer la fracture numérique avec les grandes villes.

En Australie, l'Internet par satellite est partagé (voir par exemple, l'offre de l'opérateur Inmarsat qui est de type BGAN : Broadband Global Area Network). En d'autres termes, il n'y a pas d'allocation par flots. Aussi, le partage équitable opéré par l'algorithme AIMD a toute sa place sur ces types de liens.

Lors de mon séjour au NICTA, Roksana Boreli était responsable d'un projet nommé Office in a Box dont l'objectif était de rendre les communications sans fil plus simples et plus rentables. Le but était de fournir un routeur d'accès équipé d'une antenne satellite ou Wimax afin d'offrir des possibilités de communications ubiquitaires aux entreprises ou aux particuliers qui ne peuvent pas être desservis par une infrastructure fixe. L'un des atouts de ce routeur (maintenant commercialisé par la *spin-off* 7-IP [4]) était d'offrir tout un panel de service dont la VoIP via le PBX logiciel Asterisk.

C'est dans ce contexte que nous nous sommes intéressés aux performances des codecs audio au-dessus de DCCP-CCID4. Nous avons donc principalement effectué une campagne de mesures évaluant la version TFRC-SP (*small packet variant* de TFRC) sur une connexion satellite IPSTAR (opérationnelle en Australie et dans de nombreux pays asiatiques) complétée par quelques mesures avec Inmarsat (en grande partie à cause du coût de cet opérateur).

Nous avons observé que les taux de pertes obtenus par tous les codecs testés au-dessus de DCCP-CCID4 étaient plus fort qu'avec UDP. Afin de comprendre la cause de ces mauvaises performances, nous avons considéré les sources de pertes potentielles suivantes :

- les pertes de paquets entre l'application et le protocole de transport résultant de la différence existante entre le taux d'émission sollicité par l'application et celui offert par la couche transport ;
- les pertes du lien, provenant de la congestion ou d'erreurs bit qui impactent sur le taux d'émission calculé par l'émetteur DCCP ;
- les pertes relatives à la gigue puisqu'une application de VoIP peut considérer comme perdu un paquet arrivant trop tard.

Concernant le dernier point, la qualité d'une communication audio est généralement évalué grâce à la métrique du Mean Opinion Score [8]. Dans le calcul du MOS, la perte et le délai sont tous deux facteurs de dégradation de ce coefficient de qualité.

Les résultats du PLR obtenus sur la liaison satellite IPSTAR sont donnés dans le tableau 4.1 et comprennent à la fois les pertes du lien et applicatives (entre l'application et le protocole de transport). On rappelle que ces dernières sont uniquement applicables à DCCP car

4. `http://www.7-ip.com/`

UDP ne temporisent pas l'envoi des données.

Voice Codec & load	Data rate (kbit/s)	CCID4 (%) (%)	UDP (%)
G.711	80	2.01	0.15
G.729	22	1.24	0.1
Speex	average 25	1.84	0.1
Speex/DTX	variable	1.73	0.1
Speex,5 calls	average 96	2	0.15
G.711,12 calls	780	6.32	1.55

TABLE 4.1 – PLR moyen pour differents codecs sur IPSTAR

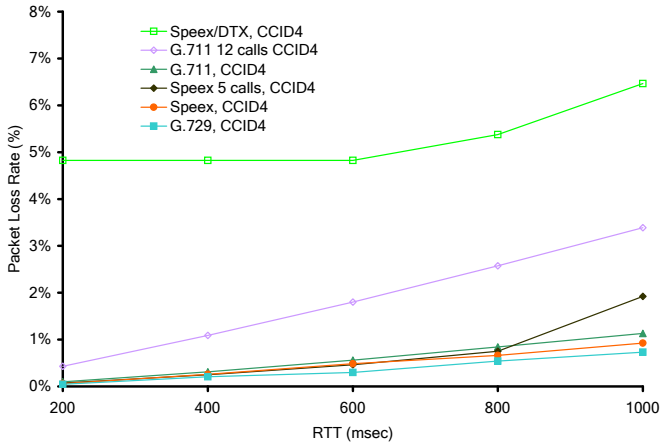


FIGURE 4.1 – PLR pour un RTT croissant avec l'émulateur Netem

Les données expérimentales du tableau indiquent que seul G.711 a obtenu des pertes sur le lien et que toutes les autres pertes ont été entre l'application et la couche transport. Par conséquent, il peut être conclu que la majorité des pertes sont causées par l'incapacité du protocole DCCP-CCID4 à fournir un taux d'émission suffisamment élevé pour la VoIP. En outre, dans le cas où il n'y ait aucune perte, DCCP-CCID4 reste en mode *slowstart* et n'atteind jamais la phase de *congestion avoidance* permettant un partage équitable entre les flots. Pour confirmer que ces pertes de paquets sont principalement en phase de *slowstart*, nous avons également effectué une série d'expériences en utilisant l'émulateur Linux Netem avec des valeurs différentes de RTT. Nous avons émulé le débit IPSTAR sans ajouter de pertes sur le lien. La figure 4.1 donne les valeurs résultantes des pertes entre l'application et le protocole de transport pour différents scénarios expérimentaux.

Suite à ce constat, nous avons proposé plusieurs modifications algorithmiques possibles permettant de diminuer ces pertes en changeant, notamment, la fréquence de génération d'acquittements du protocole et en modifiant la façon de calculer le taux d'émission. Ces

détails protocolaires ont été publiés dans [6].

Afin d'illustrer les bénéfices obtenus par ces modifications, la figure 4.2 présente les résultats obtenus par CCID4-N et CCID4-N100, deux variantes de notre proposition, comparés au protocole UDP et CCID4.

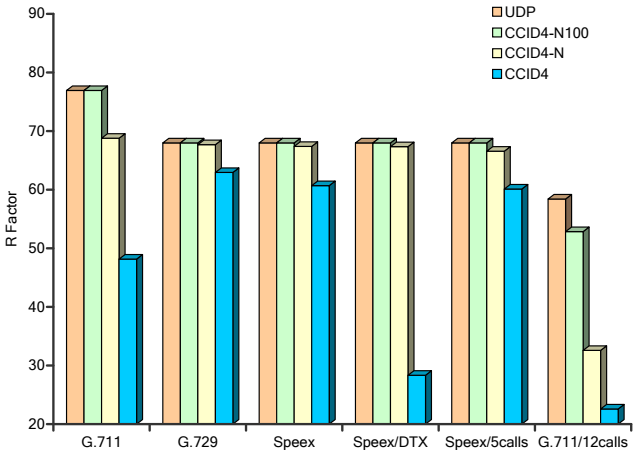


FIGURE 4.2 – Valeur du facteur R utilisé dans le calcul du MOS pour différents codecs au-dessus de IPSTAR

Il est à noter que ces propositions suivent la même philosophie que les récentes modifications de la RFC 5348 qui cherchent à améliorer les performances de DCCP-CCID3 et CCID4 sur les liens long délais. En conséquence, notre algorithme n'a pas été suivi d'une contribution à l'IETF.

Un service de délivrance ordonnée pour TFRC et les applications multimedia

Enfin pour conclure avec ces modifications, la dernière que nous avons proposée et que nous comptons défendre à l'IETF concerne l'implémentation d'un buffer de réordonnancement des paquets au sein du protocole DCCP. Nous savons qu'un paquet arrivant dans le désordre après un certain délai peut être non joué par l'application. Cependant, ce même paquet peut être considéré comme perdu par la couche transport qui réduira alors son taux d'émission. En quelque sorte, il y a une double pénalité qui s'applique et qui est fortement préjudiciable aux performances des applications multimedia lorsque le taux de déséquencement est important.

Nous savons que le déséquencement dans le réseau (*network reordering*) n'est pas un phénomène rare et qu'il se trouve maintenant amplifié à cause des liaisons mobiles mais surtout à cause de celles à fort produit bande passante délai. En particulier dans [9], les auteurs montrent que 40% des liens Internet présents dans leurs échantillons statistiques font effectivement du déséquencement et que 3% à 5% des paquets de l'Internet sont délivrés dans le désordre. À la vue de l'augmentation des vitesses des liens, ce phénomène a de

grandes chances de s'amplifier dans le futur.

De plus, bien qu'étant une conséquence du mécanisme de fiabilité de TCP, le service de délivrance ordonné n'est pas forcément implicite à un service fiable. Certaines applications multimedia utilisent elles-même un *playout buffer* pour temporiser les données qu'elles doivent jouer.

Options used	G.711	G.729
DCCP buffer	4.33	3.96
DCCP ; no buffer, voice buffer	4.33	3.96
DCCP ; no buffer, no voice buffer	3.87	3.27

TABLE 4.2 – MOS pour les codecs audio G.711 et G.729

Options used	MOS 4.2″ screen
DCCP-buffer	4.19
DCCP-no buffer ; video buffer, rate adjustement	2.76
DCCP-no buffer ; no video buffer, no rate adjustment	too low to calculate

TABLE 4.3 – MOS pour le codec video H.264

Puisque DCCP est un protocole foncièrement tourné vers l'application, nous avons proposé de mettre en place un buffer de temporisation prenant en compte dynamiquement le délai toléré par cette dernière. Si l'on considère D_{max}, le délai maximum tolérable par l'application (entre 200ms et 400ms pour l'audio et 100ms pour la vidéo-conférence) et le taux d'émission des données de TFRC (communément noté X) par la couche transport, nous pouvons dimensionner un buffer ayant pour taille $D_{max} * X$ qui permettra alors d'éviter une diminution inutile du débit d'émission du protocole de transport tout en respectant le délai de la contrainte applicative. La démonstration triviale est donnée dans [5].

Nous avons donc implémenté ce type de buffer et évalué les bénéfices pour l'application en calculant le MOS pour deux codecs audio et un codec video. Les tableaux 4.2 et 4.3 présentent les bénéfices en termes de qualité d'expérience de ces codecs sur un réseau ayant un taux de déséquencement de 5% et 100ms de RTT.

Quel avenir pour Chameleon ?

Au même titre que SCTP qui a du mal à se frayer un chemin en tant que protocole de l'Internet, DCCP n'échappe pas à la règle et ne possède à ce jour qu'une implémentation maintenue sous GNU/Linux. La fin du projet japonais WIDE ayant entrainé l'arrêt du développement de la souche Kame pour les systèmes *BSD, ces derniers ne possèdent donc pas une implémentation mûre (ce qui a indirectement un impact sur le déploiement de ce

protocole au sein du système OSX). Fort est de constater qu'un protocole compatible avec un récepteur TCP a beaucoup plus de chances d'être déployé.

Aujourd'hui, les systèmes d'exploitations possèdent une granularité temporelle de l'ordre de la micro-seconde qui permet le prototypage de protocoles de transport dans l'espace utilisateur. Si l'on considère les débits résultants des applications multimedia qui tournent autour de 10Mbit/s pour de la vidéo HD, il devient tout à fait pertinent de proposer des librairies d'implémentations de fonctions transport pour ces applications. Bien qu'à l'encontre de la philosophie de l'IETF, l'implémentation DCCP-TP réalisée dans l'espace utilisateur par Tom Phelan en est un exemple fort et ouvre une piste quant à la possibilité d'utiliser les mécanismes proposés dans ce chapitre. Comme précédemment soulevé au chapitre 2, ces déploiements existent déjà et le goulot du sablier du *hourglass model* présenté au chapitre 1 confirme qu'il a bien glissé de la couche IP à la couche transport. Ceci nous offre un éventail de solutions beaucoup plus large et permet ainsi de contrecarrer l'ossification du monde du transport.

Références sur le chapitre

[1] **Towards sender-based TFRC, Guillaume Jourjon, Emmanuel Lochin, Patrick Sénac, In Proceedings of IEEE ICC, 2007,** ***Best paper award of the Multimedia Communications & Home Services Symposium of ICC 2007***

[2] **Promoting the Use of Reliable Rate Based Transport Protocols : The Chameleon Protocol, Emmanuel Lochin, Guillaume Jourjon, Sebastien Ardon, Patrick Sénac, International Journal of Internet Protocol Technology, Vol. 5, N. 4, 2010**

[3] **Towards a Sender-Based TCP Friendly Rate Control (TFRC) Protocol, Guillaume Jourjon, Emmanuel Lochin, Patrick Sénac, Journal of Internet Engineering, 2009**

[4] **Optimization of TFRC loss history initialization, Guillaume Jourjon, Emmanuel Lochin, Laurent Dairaine, IEEE Communication Letters, 11 (3), 2007**

[5] **Mitigating the Impact of Packet Reordering to Maximize Performance of Multimedia Applications Golam Sarwar and Emmanuel Lochin and Roksana Boreli IEEE ICC 2011**

[6] **Performance of VoIP with DCCP for Satellite Links, Golam Sarwar, Roksana Boreli, Emmanuel Lochin IEEE ICC 2009**

[7] Jiayue He, Jennifer Rexford and Mung Chiang, Don't Optimize Existing Protocols, Design Optimizable Protocols, ACM SIGCOMM Computer Communication Review, 2007

[8] G.1070 : Opinion model for video-telephony applications, ITU-T, 2007

[9] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, James F. Kurose, Donald F. Towsley, Measurement and classification of out-of-sequence packets in a tier-1 IP backbone, IEEE/ACM Transactions on Networking, 15(1) :54-66, 2007

[10] Abu Zafar M. Shahriar, Mohammed Atiquzzaman, Will Ivancic, Lloyd Wood, A Sender-based TFRC for Saratoga : A Rate Control Mechanism for a Space-Friendly Transfer Protocol IEEE Aerospace conference, Big Sky, Montana, March 2011

[11] Norman C. Hutchinson and Larry L. Peterson, The x-kernel : An architecture for implementing network protocols, IEEE Transactions Software Engineering, 17 (1), 1991

[12] Yunhong Gu and Robert L. Grossman, UDT : UDP-based data transfer for high-speed wide area networks, Computer Networks, 51 (7), 2007

[13] Bryan Ford, Structured Streams : a New Transport Abstraction, In Proceedings of ACM SIGCOMM 2007

[14] Ernesto Exposito, Patrick Sénac, Michel Diaz, UML-SDL modelling of the FPTP QoS oriented transport protocol, In Proceedings of International Multimedia Modelling Conference, 2004

[15] A. Leiggener and R. Schmitz and A. Festag and L. Eggert and W. Effelsberg, Analysis of Path Characteristics and Transport Protocol Design in Vehicular Ad Hoc Networks, In Proceedings IEEE VTC, 2006

[16] Sarah Sharafkandi and Naceur Malouch, Simple and Effective End-to-End Approach to Increase TCP Throughput over Ad-hoc Networks, In the 19th International Teletraffic Congress, 2005

[17] Kai Chen and Klara Nahrstedt, The utility of explicit rate-based flow control in mobile ad hoc networks, In Proceedings of the IEEE WCNC, 2004

[18] Kai Chen and Klara Nahrstedt, Limitations of Equation-based Congestion Control in Mobile Ad Hoc Networks, Workshop on Wireless Ad Hoc Networking, 2004

[19] G. Holland and N. Vaidya, Analysis of tcp performance over mobile ad hoc networks, In Proceedings of ACM/IEEE MobiCom, 1999

[20] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, ATP : A reliable transport protocol for ad-hoc networks, In Proceedings of MobiHoc, 2003

[21] M. Georg and S. Gorinsky, Protecting TFRC from a selfish receiver, In Proceedings of Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS), 2005

[22] M. Vojnovic and J. Le Boudec, On the long run behavior of equation-based rate control, In Proceedings of ACM Sigcomm, 2002

[23] An Experimental Evaluation of Voice Quality over the Datagram Congestion Control Protocol, Vlad Balan, Lars Eggert, Saverio Niccolini and Marcus Brunner, In Proceedings of IEEE Infocom, 2007

A second stack ?

Une discussion concernant la pertinence de l'utilisation de contrôle de congestion basée sur le délai sur ICCRG entre John Leslie et Matt Mathis. L'intérêt de cette discussion réside dans l'idée de l'utilisation d'un second congestion contrôle. Il n'y a donc pas universalité de la méthode. C'est ce que nous développons dans le chapitre qui suit.

Matt Mathis : 1) When delay sensing works it generally provides better, more precise, congestion control with less disruption to other applications than loss based congestion control.

John Leslie : I did not take that position, but I won't argue against it...

Matt Mathis : 2) Delay sensing does not work in all environments and must be considered to be an optimization that is conditionally applied in addition to the primary loss based congestion control.

John Leslie : I don't even agree with that ...

Matt Mathis : 3) For the vast majority of Internet users (e.g. home users behind a slow, over buffered access link and perhaps wireless users) #1 applies extremely strongly.

John Leslie : I think Matt means that if we can determine that the bottleneck is between the end-user and his/her ISP, delay is a better metric. I would agree with that.

Matt Mathis : Therefore : all stacks should include a secondary congestion control mechanism that detects when they are causing large queues in the network and regulates their congestion window accordingly.

John Leslie : I couldn't agree with that exact wording, but Matt & I may be close here. Perhaps : "Stacks intending to "play nice" with TCP should implement a delay-based mechanism to avoid reaching a congestion level where packets they send are dropped instead of delivered. Research into such methods is an appropriate field of study of the ICCRG.".

Matt Mathis : It may be an appropriate future work item of the ICCRG and IETF to attempt to standardize such a mechanism.

John Leslie : I think in appropriate <u>now</u> to collect data on how well delay-based reductions (not necessarily multiplicative reductions) in window size signal "gzornenplat" and avoid packets being dropped. ("gzornenplat" is a placeholder for what I actually wanted to talk about : the onset of what I call "congestion", but not so much of it as to require packets to be dropped.)

Chapitre 5

TCP extreme makeover

De nos jours, l'hétérogénéité des environnements rend obsolète l'idée d'un unique contrôle de congestion [7]. Dans l'Internet, le contrôle de congestion est effectué par la couche de transport et donc principalement par TCP. Afin de traiter les nombreux problèmes liés à l'hétérogénéité, une multitude de versions TCP ont été proposées ces dix dernières années (voir les nombreuses propositions disséminées sur les mailing-lists suivantes : `[tcpm]`, `[iccrg]` et `[tsvwg]`). Chacune a une spécificité particulière. Certaines s'attaquent au problème du taux de pertes important des environnements sans fil [8] ou satellite [6] quand d'autres proposent des solutions aux réseaux haut débit [5]. Toutes ces propositions définissent une méthode de détection de la congestion et une fonction de contrôle de la fenêtre d'émission qui leur est propre dans le but d'optimiser une caractéristique réseau spécifique.

Décorreler l'algorithme de détection de la congestion de TCP pour l'adapter à l'Internet d'aujourd'hui

A notre meilleure connaissance, il n'existe pas à ce jour de version TCP universelle capable de se comporter de façon homogène sur différent types de réseaux. La conséquence directe est que le code TCP se complexifie de plus en plus et que les extensions mineures, comme par exemple F-RTO (Forward Retransmission TimeOut-recovery) [9], n'aident pas en termes de clarification du code existant. De plus, certaines de ces extensions ont été développées pour certaines versions de TCP et ne peuvent être utilisées de façon générique. Aussi, la pertinence du modèle OSI, comme déjà souligné en chapitre 2, est de plus en plus remise en question par les auteurs de [10] qui préconisent de découper la couche transport en trois sous-couches distinctes pour mieux répondre aux caractéristiques des nouveaux réseaux et décomplexifier le code source de TCP. L'idée est que factoriser le contrôle de congestion en différentes couches intermédiaires permet d'optimiser ce dernier au lien utilisé (satellite, sans-fil) ; permet un déploiement incrémental de nouveaux contrôles de congestion dans un domaine particulier ; libère l'évolution de contrôle de congestion du joug de la *TCP-*

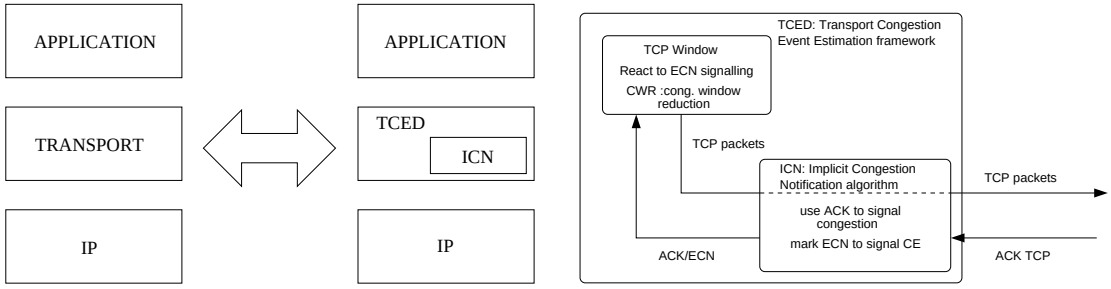
friendliness ; facilite le *multihoming* et les communications multi-chemins.

C'est sur cette idée que nous avions travaillé avec Pascal Anelli sur la dissociation de l'algorithme de détection des pertes de celui du contrôle de la congestion. Ce travail s'est effectué en deux étapes, nous avons tout d'abord travaillé sur un mécanisme de détection nommé ICN (Implicit Congestion Notification) qui sera plus tard intégré dans un framewok du nom de TCED (Transport Congestion Events Detection) comme alternative à l'actuelle couche transport.

Logique et motivation du concept

Quelque soit l'environnement et la source de trafic, une congestion du réseau se traduit toujours par une dégradation du service. Ne serait-il pas alors intéressant, au lieu de voir se multiplier les versions de TCP, de déléguer la détection de congestion ? De façon similaire, c'est aussi la motivation de la notification explicite de congestion (ECN) et des approches à notification explicite de débit (ERN). La détection de la congestion est faite par les routeurs par une gestion active de la file d'attente (AQM) qui ensuite informe le système d'extrémité par un signal binaire. Cependant, nous savons par expérience que le déploiement de nouveaux mécanismes au sein du cœur du réseau n'est pas le choix le plus judicieux. Une solution élégante serait similaire à celle proposée dans [11] où les auteurs proposent d'émuler l'AQM dans le système d'extrémité, ce qui revient à remettre la détection de congestion dans l'extrémité sans déploiement d'AQM au sein du cœur du réseau. Notre idée est donc de modulariser la détection de congestion du protocole de transport tout en maintenant la notification de la congestion par l'extrémité à la source du flot. Ceci présente quelques avantages :

- le protocole de transport en est simplifié. Il continue cependant à contrôler son débit selon une fonction de contrôle qui lui est propre ;
- la détection de congestion peut s'ajuster à l'environnement. En suivant l'idée de la couche de régulation de flot (Flow Regulation Layer [10]), des mesures appropriées peuvent être prises pour éliminer les bruits introduits par l'environnement et mieux s'adapter au lien sous-jacent. La détection de la congestion peut en sortir renforcée et plus fiable ;
- la détection de la congestion devient commune aux versions de TCP voir multi-protocoles ;
- de nouvelles fonctions peuvent être ajoutées comme un contrôle de débit. Certains flots peuvent être notifiés volontairement de fausse congestion. Des fonctions d'administration ou de métrologie peuvent également être réalisées telles que la mesure du taux de congestion pour un flot ou pour une destination.



(a) Nouvelle architecture proposée

(b) Implémentation possible du principe modulaire proposé

FIGURE 5.1 – L'architecture TCED

L'architecture TCED (Transport Congestion Events Detection)

L'architecture TCED s'intercale au niveau de la couche transport comme representé sur la figure 5.1(a).

Au sein de cette couche, nous proposons de modulariser la détection de la congestion et de la désolidariser de la réaction de TCP. Pour donner un exemple d'une implémentation possible (voir figure 5.1(b)), nous avons envisagé un protocole TCP indépendant qui ne réagirait uniquement qu'aux messages ECN provenant du module de détection de la congestion ICN. Ainsi, seul ce module agirait comme un pilote en fonction d'estimations retournées par une ou plusieurs méthodes de détection de congestion (*loss-based* ou *delay-based*).

Principaux résultats

Nous avons validé le principe de détection de congestion à l'extérieur de TCP par le développement d'une méthode de détection des événements de congestion. Cette méthode a été développée pour des sources TCP de type Reno. Avec TCP Reno, un événement de congestion se déduit de la perte d'une donnée. Nous avons donc étudié comment les pertes pouvaient être déduites par l'observation d'un flot TCP en partant de travaux existant sur l'analyse des numéros de séquences des données et des acquittements TCP (voir par exemple l'algorithme LEAST [3] proposé par Mark Allman). Le principe de ces expériences sera repris et étendu avec la notion d'événement de congestion.

La notion d'événement de congestion (CE : *Congestion Event*) indique une fenêtre de données avec un ou plusieurs paquets perdus ou marqués ECN [1] (voir le RFC 3649). En d'autres termes, TCP ne réagit pas à proprement parlé au nombre exact de paquets perdus mais à un événement de congestion qui peut aussi être notifié avec ECN (*Explicit Congestion Notification*) (voir le RFC 3168). De façon générale, un CE constitue une indication (retour négatif)

1. A noter que le terme "événement de perte" (LE : *Loss Event*) s'utilise pareillement et désigne la même notion

pour une source utilisant un contrôle de congestion en boucle de rétro-action (ou dit boucle fermée).

Tout d'abord, nous avons conçu un algorithme qui détecte les congestions comme le fait la version basique de TCP (l'algorithme ICN). TCP doit faire les reprises rapidement et précisément. Or ces deux objectifs sont orthogonaux. Une reprise précise demande du temps pour savoir si l'unité de donnée est vraiment perdue ou simplement retardée. Si les reprises sont lentes ou erronées, la performance de TCP s'en trouve dégradée. TCP fait un compromis entre ces deux objectifs. Dans le cas d'un désequencement (*network re-order*) ou d'une augmentation importante du délai (comme dans le cas des réseaux de mobiles), il y a de grandes chances que TCP détecte à tort une congestion. Nous avons donc proposé deux extensions à notre algorithme qui visent à éliminer les faux événements de congestion. La première extension repose sur un délai de validation. La seconde extension utilise l'option d'estampille temporelle. Cette solution est l'une que TCP peut mettre en œuvre pour se protéger de ce genre d'erreur [4].

Conclusion

L'évaluation de notre proposition a été effectuée sous ns-2 et au travers d'un modèle d'expérimentation, nous avons recherché des motifs variés de congestion en faisant intervenir la charge et le désequencement. Ainsi, nous avons pu établir la faisabilité de notre proposition et la fiabilité apportée dans la détection des événements de congestion. Notamment, nous avons identifié des cas spécifiques relatifs au réordonnancement de paquets dans le réseaux qui n'avaient pas été précédemment identifiés.

L'ensemble des résultats obtenus sont disponibles dans [1]. Le résultat majeur de cette étude est la preuve de la possibilité de décorréler complètement la détection de la congestion de l'algorithme TCP. Du point de vue du génie logiciel, ce type de contribution simplifie considérablement le code et permet une analyse plus fine des performances d'un algorithme de détection de TCP. La simulation sous ns-2 nous a permis d'obtenir une analyse fine des traces *a posteriori* afin de vérifier les bénéfices proposés par notre approche.

Il est à noter qu'une approche similaire est en cours dans le noyau FreeBSD où le CAIA (Centre for Advanced Internet Architectures) de l'université Swinburne de Melbourne propose une modularisation des contrôles de congestion existant au sein du noyau BSD afin d'en mutualiser l'utilisation entre différents protocoles de transport comme par exemple, entre TCP et SCTP [2].

Références sur le chapitre

[1] **Pascal Anelli, Emmanuel Lochin, Fanilo Harivelo and Dino Martin Lopez Pacheco, Transport Congestion Events Detection (TCED) : Towards Decorrelating Congestion Detection from TCP, ACM SAC 2010 Networking Track, 2010**

[2] `http://caia.swin.edu.au/urp/newtcp/tools.html`

[3] Mark Allman, Wesley Eddy, Shawn Ostermann, Estimating Loss Rates With TCP, ACM Performance Evaluation Review, 31(3), December 2003

[4] Michael Welzl, Using the ECN Nonce to detect Spurious Loss Events in TCP, IEEE Globecom, 2008

[5] Geoff Huston, Gigabit TCP, Internet Protocol Journal, 2006

[6] Carlo Caini and Rosario Firrincieli, TCP hybla : a TCP enhancement for heterogeneous networks, International Journal of Satellite Communications and Networking, 2004.

[7] Ao Tang and David Wei and Steven H. Low, Heterogeneous Congestion Control : Efficiency, Fairness and Design, IEEE ICNP, 2006

[8] Ye Tian and Kai Xu and Nirwan Ansari, TCP in Wireless Environments : Problems and Solutions, IEEE Communications Magazine, 2005

[9] P. Sarolahti and M. Kojo, Forward RTO-Recovery (F-RTO) : An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP), RFC 4138 (Experimental), August 2005.

[10] Bryan Ford and Janardhan Iyengar, Breaking Up the Transport Logjam, In the Seventh ACM Workshop on SIGCOMM Hot Topics in Networks (HotNets-VII), 2008

[11] Bhandarkar, Sumitha and Reddy, A. L. Narasimha and Zhang, Yueping and Loguinov, D., Emulating AQM from end hosts, ACM Sigcomm, 2007

Chapitre 6

Agir dans le réseau : Kohonen-RED, ECN* et Favour Queue

Agir dans le réseau est la proposition la plus difficile à mettre en œuvre. Bien que nous avons la certitude que la conception d'un mécanisme qui managerait de façon optimale la capacité réseau tout en offrant une équité intra et inter-flots ne peut être réalisée sans collaboration de ce dernier [14, 16], les propositions impliquant le cœur du réseau ne sont généralement pas déployées.

Ce chapitre présente cependant trois contributions distinctes qui pourraient, pour certaines, être déployées de façon incrémentale.

Introduction

L'étude des files d'attente actives (AQM : Active Queue Management) a occupé la communauté réseaux des années 90 qui avait investi un effort considérable dans ces algorithmes. Le but ciblé était d'offrir aux opérateurs réseaux une méthode simple en vue d'obtenir des délais de traversée rapide et d'atteindre de haut débits. L'idée était de détecter la congestion avant qu'elle ne se produise effectivement dans le réseau et deux méthodes avaient alors été envisagées :

1. soit jeter les paquets, obligeant implicitement les sources TCP à réagir ;
2. soit marquer le bit indicateur de congestion ECN (au lieu de jeter les paquets) afin de solliciter une réaction de la source TCP.

Nous ne reviendrons pas sur les débats qui avaient animé la communauté scientifique sur l'invraisemblance de jeter des paquets pour prévenir la congestion et sur le refus de la politique de rejet inhérente à l'AQM RED [27]. En effet, nombreuses sont les études qui ont montré que RED était fondamentalement difficile, voire impossible à mettre en œuvre [26] et très instable [25] [24].

Il existe cependant un amalgame entre la solution RED/Dropping et la solution RED/ECN qui reste synonyme de mauvaises performances et d'infaisabilité. Pourtant, des études montrent les atouts indéniables des bénéfices apportés par une indication explicite de congestion vers les extrémités. En effet, Sally Floyd en 1994 avait déjà montré que cette indication permet–tait d'augmenter considérablement les performances du système [15]. Dans [22], les auteurs font la même constatation sur le trafic web et encore plus récemment en 2005, Aleksandar Kuzmanovic dans *"The Power of Explicit Congestion Notification"* [28] reconsidère la pertinence du marquage ECN et démontre que les terminaux qui utiliseront ECN dans l'instant, bénéficieront de performances accrues et immédiates et ce, sans avoir besoin d'attendre que la communauté Internet se décide à supporter définitivement ce choix.

Malgré le poids de ces arguments, l'étude suivante [29] précise que ECN n'est utilisé que par 2,1% des hôtes en 2004. Pourtant, bien que ECN soit relativement bien implémenté aussi bien dans les extrémités (dans les systèmes GNU/Linux, MacOSX et Windows) que dans les éléments de cœur du réseau (le système IOS des routeurs Cisco propose une variante de RED/ECN nommée WRED/ECN), cette option reste désactivée par défaut. Concernant les extrémités, c'est assez paradoxal. Alors qu'aujourd'hui des variantes de TCP non standardisées telles CUBIC ou Compound sont proposées par défaut (respectivement au sein de systèmes GNU/Linux et Windows Vista), soulevant de nombreux débats concernant leur équité avec la version TCP Newreno standard, un mécanisme prouvé et standardisé comme ECN ne l'est pas.

Cette absence de déploiement pourrait s'expliquer en deux points :

1. les paramètres de configuration de RED sont très difficiles à déterminer. Même lorsque cela est possible pour des hypothèses données, il restent très peu généralisables ;
2. la tâche ingénieuriale de mise en œuvre d'un tel mécanisme au sein de l'Internet est complexe.

Il est également important de noter que ce pourcentage assez faible s'explique aussi par les pare-feux et autres *middle-boxes* de l'Internet qui réinitialisent généralement (et sans justification concrète) une connexion TCP utilisant le drapeau ECN [28]. Cela ne plaide pas en la faveur de certaines architectures *clean slate* qui proposent de révolutionner l'Internet avec des solutions qui semblent difficilement déployables sur une courte échelle temporelle. De plus, ces architectures peuvent rentrer en conflit avec certains intérêts économiques et légaux (voir notamment l'argumentation des *"Tussle"* de l'Internet [17]).

Dans un contexte plus privatif, il est à noter une proposition protocolaire pour Data Center se basant sur ECN afin de réduire le délai de traversée des files d'attentes de routeur [10]. En effet, bien que les réseaux sur-provisionnés, possédant de grandes capacités de bufferisation, permettent de réduire le taux de pertes, ces derniers ne permettent pas de minimiser le délai. Le tout récent protocole DCTCP [10] proposé par Microsoft utilise donc un marquage particulier basé sur ECN afin de réduire la taille des buffers des routeurs.

Ce chapitre présente trois contributions qui cherchent à agir au sein du réseau de cœur. La première nommée KRED, cherche à résoudre le premier point en vérifiant si l'intuition de Sally Floyd, et notamment la configuration complexe des paramètres de RED peut être solutionnée à l'aide des réseaux neuronaux. La seconde, ECN*, se trouve en marge de ces points et propose d'améliorer la lisibilité par ECN du niveau de la congestion d'un chemin du réseau. Enfin, Favour-Queue, qui est une file d'attente sans état, autonome, sans configuration initiale et facilement déployable, propose une alternative concrète aux performances du trafic et en particulier des flots courts.

Kohonen-RED (KRED)

Lorsque j'étais étudiant en thèse, je fus à la fois intrigué par l'intuition géniale de Sally Floyd pour la file d'attente RED et la littérature prolifique qui s'afférait à démontrer l'inutilité de ce mécanisme. A cette époque, peu d'études ne s'attelaient à chercher la meilleure configuration possible de ce mécanisme. Certaines d'entre-elles ont très vite parlé d'impossibilité.

C'est en discutant avec Bruno Talavera, ingénieur à l'UPMC et spécialisé dans l'étude des réseaux de neurones, que j'ai cherché à trouver une solution en dehors de ces études. Notre but n'était pas de proposer une n-ième variante de RED, mais simplement de trouver l'algorithme adaptatif qui permettrait son fonctionnement optimal (i.e. la détermination correcte du paramètre de rejet). Nous avons donc fait l'hypothèse que RED était bien capable de stabiliser une file d'attente quelque soit le trafic en entrée si sa probabilité de rejet était correcte. Bien évidemment, nous n'étions pas les premiers à utiliser un mécanisme issu de la théorie du contrôle ou de l'intelligence artificielle. Mais notre approche était très différente des contributions existantes. L'objectif était de trouver un problème analogue et résolu qui se transposerait à RED et donc de vérifier si vraiment, RED pouvait être paramètré de façon générique. Après une longue investigation des réseaux neuronaux existants, nous avons trouvé une solution avec une classe particulière de réseaux connus sous le nom de Kohonen ou Self Organizing Map (SOM).

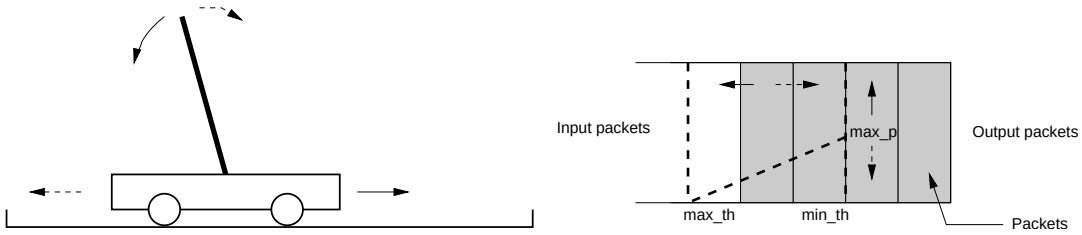
Les réseaux Kohonen font partie d'une classe de réseaux neuronaux capable de résoudre le problème du "balancier" ou dit du "funanbule" [21]. Le système du balancier est un problème de stabilité bien connu des mécaniciens. Il consiste en un balancier sur un axe libre placé sur un charriot mobile. Le charriot se déplace alors suivant une ligne de taille fixée comme schématisé sur la figure 6.1(a). L'objectif est d'appliquer des forces sur le charriot pour que le balancier reste à l'équilibre.

L'idée principale de notre contribution est basée sur l'analogie existante entre ce problème et la stabilisation de la file d'attente RED. En effet, nous pouvons comparer le balancier à l'évolution de la file d'attente RED, qui doit se stabiliser entre deux seuils donnés (min_{th}, max_{th}). Les forces qui s'appliquent correspondent à l'intensité du trafic en entrée tandis que l'action résultante sur le balancier se traduit par la valeur de la probabilité de rejet de RED.

La figure 6.1 illustre cette analogie.

	Balancier	RED
valeur d'entrée #1	position précédente	précédente taille de la file
valeur d'entrée #2	position courante	taille de la file courante
valeur de sortie #1	force à appliquer en Newton	max_p

TABLE 6.1 – Valeurs d'entrée et de sortie du réseau de neurones



(a) Balancier (b) RED adaptatif

FIGURE 6.1 – Analogie existante entre le problème du balancier et l'AQM RED

Nous savons qu'il est nécessaire de changer dynamiquement max_p en fonction du trafic entrant afin de minimiser les oscillations de la file. Aussi, les files RED auto-configurantes telles FRED ou ARED mettent à jour la valeur max_p en fonction du trafic entrant afin de stabiliser la taille de la file d'attente entre les deux seuils min_{th} et max_{th}. Partant de ce constat, de nombreuses solutions utilisant un algorithme AIMD ou basées sur la théorie du contrôle (contrôleur Proportional-Integral) ont été proposées. Tous ces algorithmes ont malheureusement des paramètres de configuration initiaux qui sur-spécialisent le mécanisme (ces mêmes paramètres α et β, qui ont des fonctions différentes mais qui, par exemple, correspondent aux coefficients multiplicateur et diviseur dans le cas de l'AIMD). Ces choix sont donc problématiques dans le cas de changement brutal du niveau du trafic. Nous avions donc un candidat idéal pour la théorie des réseaux neuronaux car l'ensemble des paramètres à prendre en compte est trop grand pour avoir un modèle mathématique générique et exploitable.

Le réseau de neurones que nous utilisons ici est connu sous le nom de Kohonen à cartes auto-organisatrices (SOM : Self Organizing Map) [19]. Il consiste en une ou deux couches de traitement d'information d'entités fonctionnelles appelées neurones. Il est relié à des données d'entrée qui sont vues comme un vecteur et qui fournit, en retour, des données de sortie également sous forme de vecteur. Nous présentons dans le tableau 6.1 les entrées utilisées pour alimenter le réseau de neurones dans les deux cas et le résultat obtenu. Le vecteur d'entrée contient la longueur de la file courante et précédente tandis que le vecteur de sortie retourne à la probabilité max_p à utiliser. Pour continuer l'analogie, nous donnons dans ce tableau les vecteurs utilisés pour le problème d'équilibrage du balancier.

Le détail exact du fonctionnement du réseau Kohonen ainsi que son implémentation ont été initialement détaillés dans [4]. Des résultats avec plusieurs configurations de réseaux sont proposés dans [3]. A titre illustratif, nous présentons quelques résultats comparant KRED et d'autres files d'attentes dans un environnement très dynamique. La topologie est celle dite du papillon et le trafic TCP change toutes les 50 secondes en variant de 25 à 100 flots.

Nous avons implémenté une version de KRED et comparé ses performances avec d'autres AQM telles RED, FRED, ARED, REM et PI. Les configurations de chacune d'entre-elles ont été choisies en fonction des recommandations des auteurs et afin de permettre une comparaison aisée de notre mécanisme, nous avons suivi les paramètres utilisés dans un récent

papier comparant plusieurs de ces AQM entre-elles [18]. Enfin, le code source et les scripts de simulation sont disponibles sur le web [1].

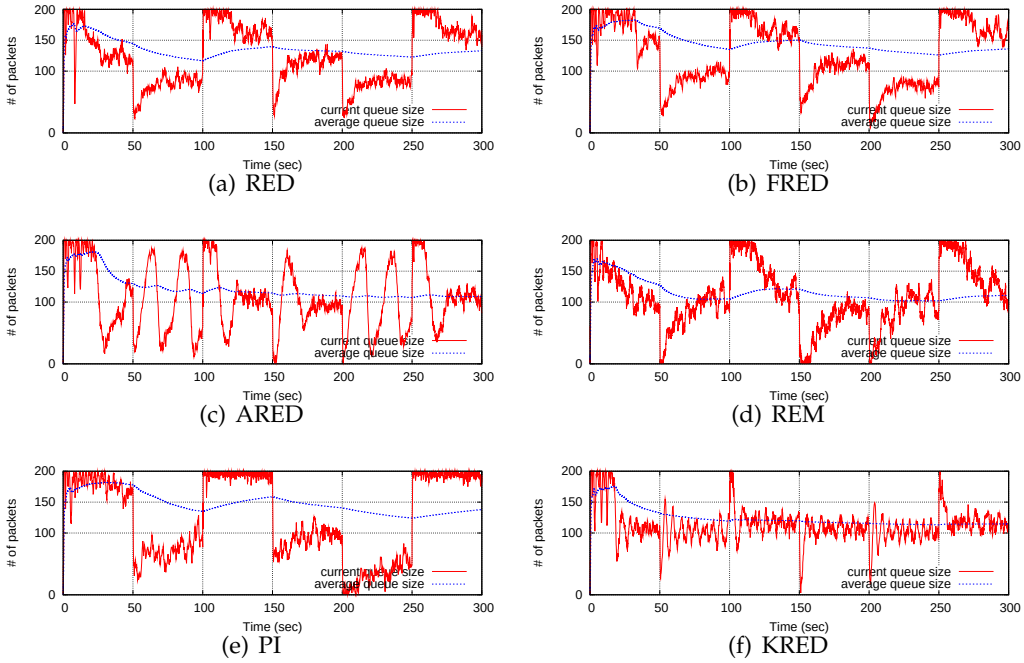


FIGURE 6.2 – Evaluation des performances de KRED comparées à différentes AQM. Le RTT de chaque flot varie entre 104ms et 142ms.

Les résultats présentés figure 6.2 illustrent la forte adaptabilité de KRED à un environnement dynamique. Après un entrainement du réseau neuronal, il existe donc une carte Kohonen capable de stabiliser une file d'attente à un débit de sortie donné. Cette carte peut alors être codée en dur dans un routeur car il est possible de déterminer une carte en fonction du débit effectif de sortie et ce, quelquesoit la variabilité du débit entrant. A noter que notre carte Kohonen représente un tableau d'entiers de 25×25 éléments, ce qui ne va pas à l'encontre d'une implémentation réelle.

Enfin et pour conclure, l'autre résultat important de cette étude est qu'il est tout à fait envisageable d'utiliser une intelligence cognitive au sein des réseaux IP et que de multiples problèmes peuvent également trouver solution dans ce domaine (voir notamment l'argumentation de deux chercheurs du CSAIL-MIT qui a en partie motivé l'aboutissement de cette étude [20]).

1. `http://manu.lochin.net/kred.html`

ECN*

Nous allons passer maintenant à une autre contribution complémentaire visant à déterminer avec exactitude le niveau de congestion d'un réseau. Nous nous intéressons maintenant à la pertinence de l'utilisation du drapeau ECN et cherchons à étendre sa capacité de lecture de l'état du réseau.

Dans l'introduction de ce chapitre, nous avons souligné que plusieurs études ont démontré l'intérêt de l'utilisation du marquage ECN. Cependant, nous pensons qu'il est possible de faire mieux que de retourner un simple signal binaire qui ne reflète pas le niveau réel de la congestion du réseau. Intuitivement, les protocoles CUBIC et Compound TCP pourraient être plus performants que TCP/Newreno avec ECN car la réaction face à un marquage ECN peut être disproportionnée à la vue de la congestion réelle du goulot d'étranglement.

Tout récemment, une solution nommée re-ECN, propose l'utilisation d'un second bit supplémentaire qui permet par l'observation en un point du réseau donné, de différencier la congestion en amont et en aval de ce point d'observation. La portée de cette solution est limitée dans le sens où elle ne permet pas d'identifier exactement le niveau de congestion du goulot d'étranglement du réseau mais de déterminer sa localisation par rapport au point d'observation. Ainsi, cette proposition trouve plutôt son intérêt au niveau des opérateurs Internet (qui leur indique si la congestion provient ou pas de leur propre domaine) tandis que nous cherchons avec notre solution une observation précise de la congestion au niveau des extrémités.

Prolongeant cette idée mais sans tomber sous la coupe d'une signalisation lourde et dans la même lignée que BMCC [14] ou XCP [16], nous avons proposé avec Rémi Diana une méthode métrologique simple et ne nécessitant pas un déploiement global (seuls les routeurs à étudier devront mettre en œuvre ce type de marquage) qui permet d'estimer de façon précise le niveau de congestion des files d'attente d'un réseau de routeurs sans impliquer les routeurs dans un quelconque calcul.

En particulier, nous visons à fournir une solution pratique permettant une mesure exploitable de la congestion afin d'éviter une réaction aveugle, approximative ou excessive de la source TCP. La seule modification porte sur la méthode de marquage qui passe d'un champ binaire à un compteur ECN similaire au champ TTL du paquet IP. En pratique, nous n'avons pas besoin d'étendre les en-têtes IP car le DiffServ Codepoint convient très bien à notre proposition. On pourrait faire valoir, comme dans [14], si une telle modification implique ou non un lourd processus de normalisation IETF. Cependant, nous pensons qu'il serait beaucoup plus complexe et incertain de convaincre un intégrateur d'ajouter une méthode d'estimation complexe dans leurs propres équipements. En outre, cette solution est suffisamment générique pour considérer ce drapeau soit comme un simple indicateur binaire (i.e. la solution reste compatible avec ECN) soit comme un compteur. Enfin, nous signalons qu'un récent groupe IETF dénommé Conex (Congestion exposition), propose "d'exposer" la congestion au niveau de la couche réseau. La principale solution candidate à ce jour est re-

ECN [13] Cette proposition a reçu un très bon accueil de la part des fournisseurs de services Internet et des intégrateurs réseaux qui possèderaient alors un outil (actuellement faisant défaut) afin de mieux gérer et contrôler leur trafic [2]. Si cette solution est adoptée, nous pourrions assister à un déploiement plus vaste de l'ECN, ce qui faciliterait alors le déploiement de notre proposition.

Le bit ECN tel qu'il est défini dans la RFC 3168, est un champ binaire contenu dans l'entête des paquets IP. Ce champ ne peut donc contenir qu'une valeur booléenne indiquant si oui ou non, le paquet ECN-capable a traversé au moins un routeur congestionné. Ainsi, il est impossible de distinguer un paquet marqué une fois d'un paquet marqué plusieurs fois et qui aurait donc traversé plusieurs routeurs congestionnés. Ceci constitue un obstacle si l'on souhaite obtenir une analyse métrologique suffisamment fine de l'état du chemin afin par exemple, de permettre une réaction proportionnée des sources vis-à-vis de cette congestion. En effet, un paquet de données ECN-capable traversant respectivement un lien contenant deux routeurs et marquant à un taux respectif de 1% et 2%, aura une probabilité de marquage de 2.98% (*i.e.* $1-(1-0.01)(1-0.02)$) qui ne reflète pas le niveau de congestion du goulot d'étranglement. Aussi, nous proposons de transformer ce bit ECN en compteur ECN (noté ECN*) qui comptabilisera le nombre de fois qu'un paquet a été marqué [3]. Nous utiliserons cette nouvelle métrique (*i.e.* le nombre de marquage consécutif d'un paquet) pour déterminer le niveau de remplissage du goulot d'étranglement du chemin traversé. Un routeur RED/ECN* voulant marquer un paquet pour notifier de son état de congestion incrémente ce compteur au lieu de simplement mettre sa valeur à "vrai". Cette mise en œuvre reste donc très simple et facilement déployable. Nous pouvons alors pour chaque paquet reçu déterminer le nombre de fois qu'il a été marqué.

Partant de la distribution du nombre de fois qu'un paquet a été marqué ECN*, l'hôte (émetteur ou récepteur) peut construire un histogramme de la distribution des marquages. Ensuite, sachant que chaque routeur implémente la même politique de marquage RED, un calcul probabiliste permet d'obtenir un polynôme dont les racines retourneront la probabilité de marquage de chaque file du réseau. Cet algorithme permet alors, lorsque l'échantillon statistique le permet, d'obtenir une bonne estimation des taux de marquage et donc, du niveau de remplissage de chaque file d'attentes congestionnées du chemin du flot. Sinon, le drapeau ECN peut être interprété par la source comme un simple signal binaire de la congestion, ce qui permet de garder la compatibilité avec l'ECN standard lorsqu'une estimation n'est pas encore possible avec ECN*. Le principe de base de l'algorithme est disponible dans [2] et une amélioration de la méthode de résolution permettant de converger plus rapidement à une solution (i.e. en utilisant un échantillon statistique plus petit) est explicitée dans [1]. Cette dernière étude discute notamment des aspects de convergence, d'application et de

2. Voir la mailing-list IETF `[re-ecn]` pour plus de détails.
3. Nous ne discuterons pas de la mise en œuvre de ce mécanisme qui peut se faire via le champ option d'IP

déploiement d'ECN*. La prochaine étape serait de coupler ECN* avec la couche transport afin de gérer la réaction du protocole de transport face à un niveau de congestion et non pas à un signal de la congestion. Une approche possible est celle développée dans [14].

Afin d'illustrer le principe de base d'ECN*, nous proposons le scénario de simulation ns-2 présenté sur la figure 6.3. Nous utilisons lors de la simulation uniquement des flots TCP. La réaction des sources TCP face au bit ECN a été désactivée. Ainsi, ces derniers ne réagissent pas par une diminution de leur fenêtre d'émission lorsqu'ils reçoivent un acquittement ECN marqué. Nous utilisons juste le champ de marquage ECN que nous avons modifié afin d'utiliser un compteur en lieu et place de la valeur booléenne standard. Tous les routeurs RED/ECN* utilisent les mêmes paramètres : $min_{th} = 50$, $max_{th} = 100$, $max_p = 1$ avec une taille de file d'attente de 100. L'algorithme est appliqué sur deux flots TCP, le flot #1 de SRC1 à RCV1 et le flot #2 de SRC2 à RCV2. La topologie présente volontairement deux routeurs en commun afin de montrer que l'étude précédente n'est pas perturbée par des trafics croisés.

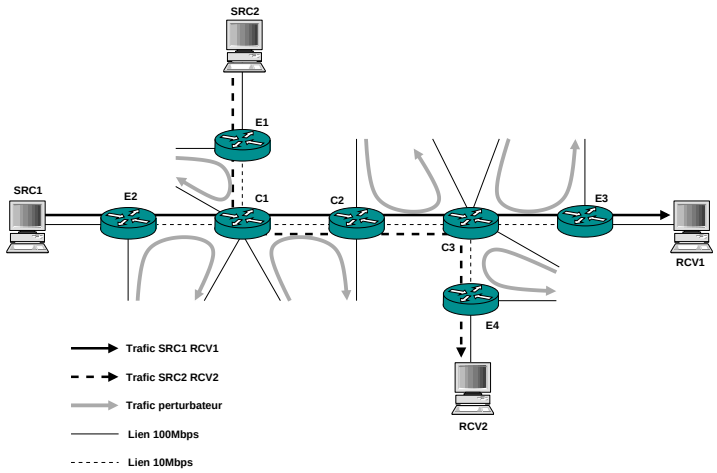
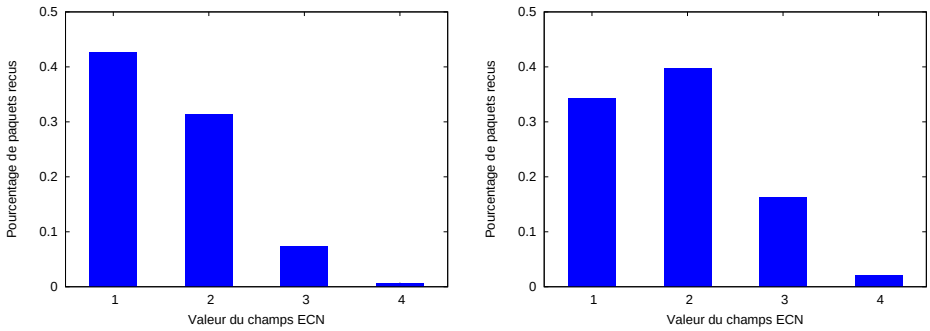


FIGURE 6.3 – Topologie utilisée pour la simulation

L'étude statistique effectuée consiste à construire l'histogramme de la valeur de marquage ECN* des paquets des flots #1 et #2. Ces résultats sont regroupés et présentés en Figure 6.4(a) et Figure 6.4(b).

Nous comparons maintenant les résultats obtenus avec les tailles moyennes mesurées de chacune des files d'attente RED/ECN* durant notre simulation. On peut ainsi en déduire les taux de marquage effectifs de chaque files RED. Ces résultats sont regroupés et présentés dans le tableau 6.2 et correspondent aux valeurs calculées par notre algorithme pour les deux flots #1 et #2. Nous précisons que les valeurs moyennes des files d'attente observées ont des écarts type négligeables. Ces valeurs sont quasiment constantes pendant toute la durée de l'analyse.

Ces résultats correspondent donc aux estimations avec parfois une légère différence due à la taille de l'échantillon statistique. De plus, si l'on fait une corrélation entre les résultats



(a) Résultats pour le flot #1 (b) Résultats pour le flot #2

FIGURE 6.4 – Distribution du marquage des paquets

File d'attente	Taille Moyenne (# pkts)	Taux de marquage Théorique	Taux de marquage estimé	
			Par RCV1	Par RCV2
File1 (lien E2–C1)	55.5	11%	11%	⊘
File2 (lien C1–C2)	60.5	21%	21%	20%
File3 (lien C2–C3)	72	44%	42%	45%
File4 (lien C3–E3)	77.5	55%	55%	⊘
File5 (lien E1–C1)	65.5	32%	⊘	31%
File6 (lien C3–E4)	87	74%	⊘	74 %

TABLE 6.2 – Taille moyenne des files d'attente et taux de marquage théorique correspondant

obtenus analytiquement et ceux par simulation Tab 6.2, nous pouvons noter que l'on obtient deux taux de marquage communs correspondant aux deux routeurs communs traversés par les deux flots étudiés. Ainsi, non seulement ces résultats correspondent aux résultats attendus mais ils soulignent aussi un aspect important : ces mesures ne se perturbent pas l'une de l'autre et sont parfaitement indépendantes. On peut donc réaliser plusieurs mesures en parallèle sur un même réseau. Nous avons également vérifié grâce à cette simulation que la seule hypothèse de stabilité du réseau est suffisante pour obtenir les résultats escomptés. Si l'on pose l'hypothèse d'une relative stabilité des routes et du niveau de congestion des routeurs, ce calcul est valide et nous permet d'obtenir une évaluation correcte de la congestion des routeurs traversés.

Favour Queue

Favour Queue prend un angle d'attaque complètement différent. Étant donné la complexité relative au déploiement d'un mécanisme tel ECN* et la mauvaise popularité de l'AQM RED, avec Eugen Dedu , nous avons cherché à améliorer les performances du trafic *best-effort* en cherchant à accélérer la latence des flots courts. De manière analogue à d'autres travaux du domaine [31, 32, 30], nous avons cherché à concevoir une AQM permettant de servir en premier le trafic TCP court afin d'améliorer le service de transfert des flots dans son ensemble.

Nous savons que le trafic de l'Internet est toujours dominé par le trafic web au dessus de courtes connexions TCP. Comme montré dans [11], parmi 95% de clients TCP de l'Internet, 70% du trafic TCP servit a une taille inférieure à dix paquets. Ceci s'explique par la pratique commune des développeurs web qui, pour accélérer le temps d'affichage, évitent les pages lourdes et multiplient le nombre de connexions TCP. En d'autres termes, l'accès à une page génère plusieurs connexions très courtes qui augmente la vitesse d'affichage du texte comparé aux éléments plus lourds tels des images ou des composants multimedia [4]. De façon évidente et à la vue de l'accroissement du trafic web, nous sommes en droit de penser que ce trafic court perdurera à l'avenir.

Cependant, TCP souffre considérablement de la présence de pertes en rafale, du trafic non-adaptatif ou lorsque sa fenêtre de congestion est petite (c'est-à-dire dans la phase de démarrage lent dite de *slow-start* ou si il opère dans un régime à petite fenêtre). En effet, les pertes en rafale ou les pertes en régime de petite fenêtre peuvent entraîner des délais d'attente de retransmission (RTO) qui déclenchent une phase de *slow-start*. Dans le contexte de flots TCP courts, une retransmission rapide (TCP Fast Retransmit) ne peut pas être déclenchée si il n'y a pas eu suffisamment de paquets en transit. En conséquence, la récupération des pertes se fait essentiellement à l'expiration de l'horloge de retransmission (RTO), ce qui impacte fortement le temps de transfert.

Le principe de base de Favour Queue est très simple et est illustré sur la figure 6.5. Lorsqu'un paquet arrive à l'entrée de la file, si celui-ci n'appartient pas à un trafic existant déjà en attente, ce paquet est alors placé en priorité. Ainsi, le nombre d'état que doit mémoriser le routeur est borné par la capacité de la file d'attente. Ce mécanisme permet de grandement favoriser la période de reprise sur erreur des flots TCP.

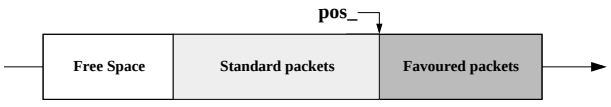


FIGURE 6.5 – Favour Queue

Les avantages immédiats de cette méthode sont que :

4. Voir par exemple : "Best Practices for Speeding Up Your Web Site", Yahoo developer network.

– il n'y a pas de déséquencement des paquets ;
– aucune modification n'est opérée sur les hôtes d'extrémités ;
– il n'y a pas d'état par flots, seul un état pour les flots présents dans la file est nécessaire et donc la complexité correspond à la taille de la file ;
– les premiers paquets d'un flot TCP ne sont pas les seuls à être prioritaires. De façon générale, tous les flots dans un régime de petite fenêtre sont prioritaires ;
– plus il y a de routeurs sur le chemin, meilleur est le temps de transmission ;
– contrairement aux précédentes contributions : notre mécanisme permet d'améliorer la latence générale du trafic sans avoir à mettre en place un quelconque marquage du trafic court, une quelconque classification prédictive ou statistique du trafic entrant.

Afin de montrer clairement l'impact de cette file sur les performances du trafic, nous nous sommes appuyés sur les travaux de [11]. Les flots TCP courts restent principalement en phase de *slow-start* et leur métrique de performance est plus le temps de réponse que le débit utile qui caractérise les flots longs. Nous avons donc choisi, comme dans [11], la métrique de la latence (ou durée du flot) qui correspond au temps écoulé entre le premier paquet envoyé et le dernier paquet reçu.

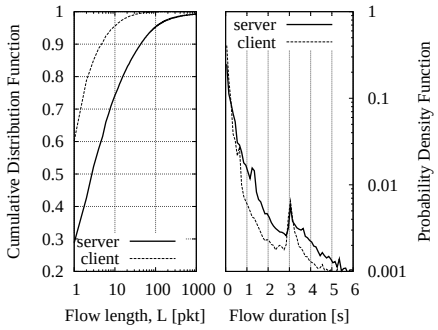


FIGURE 6.6 – Distributions de la taille des flots TCP et de leur latence (avec l'aimable autorisation de [11]).

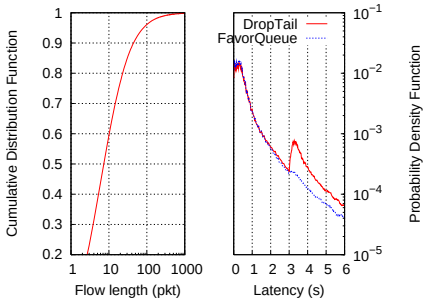


FIGURE 6.7 – Distributions de la taille des flots TCP et de leur latence obtenus avec notre modèle de simulation.

La figure 6.6, fournie par les auteurs de [11], montre clairement que la latence des flots courts est impactée par la perte d'un paquet. Sur cette figure, les auteurs donnent la fonction de répartition de la longueur des flots TCP et la densité de probabilité de leur temps d'exécution. Ces courbes ont été obtenues depuis un ensemble de données de mesures expérimentales d'une journée sur un lien ISP BRAS qui regroupe plus de 30.000 utilisateurs. Nous avons reproduit une expérience similaire avec ns-2 (c'est-à-dire avec une CDF de taille de flots similaires suivant une distribution de Pareto) et avons obtenu une densité de probabilité similaire comme le montre la figure 6.7 pour la courbe obtenue avec une file DropTail. On remarque que ces deux figures (6.6 et 6.7) montrent clairement un pic de latence à $t = 3$ secondes correspondant à la valeur de RTO par défaut et donc à la retransmission de paquets après expiration du RTO. Dans cette expérience, le ratio de la reprise sur RTO est de

56% (contre 70% pour [11]). La seconde courbe de la figure 6.7 montre le résultat obtenu par Favour Queue. Noter que le pic précédemment identifié a disparu. Ce qui explique que les pertes initiales qui impactaient fortement les performances du trafic TCP ont diminué.

Le principe de base de l'algorithme a été initialement énoncé dans [5] et une étude plus exhaustive, conjointement menée avec Pascal Anelli et Rémi Diana a permis de mieux identifier les bénéfices de ce mécanisme. En particulier nos évaluations ont montré que 58% des flots courts ont une latence améliorée et que 80% des flots longs bénéficient également de cette AQM. En phase de non-congestion, Favour Queue n'a aucun effet sur le trafic réseau. Ce mécanisme doit être vu comme une amélioration de DropTail qui vise à améliorer le service *best-effort* en favorisant plutôt qu'en rendant prioritaires certains paquets.

Afin d'évaluer les capacités de notre algorithme, nous avons utilisé le simulateur ns-2. Notre modèle de simulation suit les principales directives admises en vue d'évaluer TCP et ses variantes de façon cohérente [12]. La topologie utilisée est celle dite du papillon où n sources sont connectées à n destinations via un goulot d'étranglement entre deux routeurs. Nous modélisons le trafic du réseau en termes de flots ou sessions. Chaque flot correspond à une requête HTTP au-dessus d'une connexion TCP. La charge du lien (*link load*) est définie par :

$$\rho = \frac{\lambda E[\sigma]}{C}, \tag{6.1}$$

avec C, la capacité du goulot d'étranglement. La demande de trafic, exprimée en débit, est le produit du taux d'arrivée des flots λ avec la taille moyenne d'un flot $E[\sigma]$. Afin de supprimer la synchronisation globale des acquittements TCP et l'effet de phase, une charge de trafic de 10% est générée dans la direction opposée. L'ensemble des caractéristiques de la simulation est fournie dans [6].

Nous n'affirmons pas que ces hypothèses permettent d'obtenir une charge de trafic réaliste. Cependant, nous considérons que notre modèle de simulation est une bonne représentation du trafic des flots courts de l'Internet puisque nous trouvons la même distribution de la latence que la figure 6.6. Enfin, afin d'améliorer la qualité des échantillons statistiques, certaines expérimentations font la moyenne de dix tests consécutifs (c'est le cas de la figure 6.7), ce qui représente un dataset d'environ 17 millions de paquets.

Afin de présenter les performances générales de Favour Queue comparées à DropTail, nous présentons sur les figure 6.8 et 6.9 la moyenne et l'écart type de la latence et des pertes en fonction de la charge du trafic. Nous donnons à la fois les performances obtenues par la version de Favour Queue initiale ainsi qu'un raffinement consistant en l'adjonction d'un mécanisme *push-out* qui permet de retirer un paquet d'un flot "non-prioritaire" lorsque la file d'attente est pleine au profit d'un paquet à favoriser. On constate une amélioration des performances avec Favour Queue qui sont accentuées avec la version *push-out* lorsque la congestion est sévère. Comme montré sur la figure 6.9, la proportion de paquets jetés est globale-

ment la même entre les deux versions de Favour Queue (avec ou sans *push-out*). Cependant, la version *push-out* diminue la probabilité de perte d'une retransmission TCP, ce qui améliore grandement la latence.

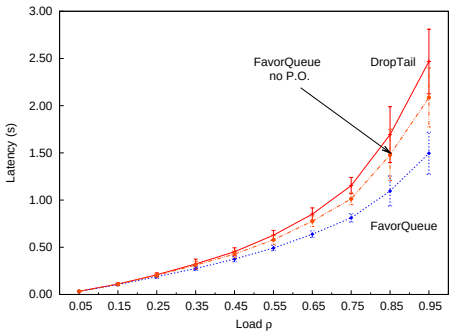


FIGURE 6.8 – Latence en fontion de la charge du trafic

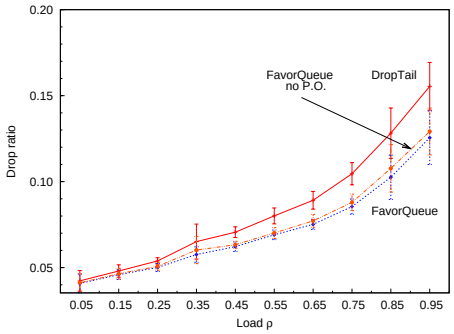


FIGURE 6.9 – Proportion de pertes en fontion de la charge du trafic

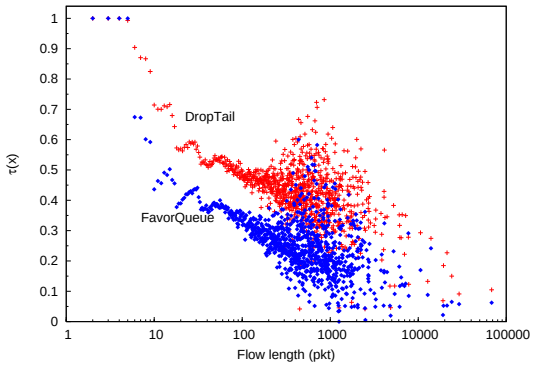


FIGURE 6.10 – RTO recovery ratio according to flow length

Pour montrer l'impact au niveau du flot, nous évaluons le ratio de reprise sur RTO comme suit :

$$\tau(x) = \frac{\sum_{i=1}^{N} RTO_i}{\sum_{i=1}^{N} (RTO_i + FR_i)}, \tag{6.2}$$

avec FR_i, le nombre de Fast Retransmit TCP pour le flot i et RTO_i son nombre de RTO. La figure 6.10 montre une diminution significative du nombre de reprises avec un RTO. En ce qui concerne le nombre de Fast Retransmit pour cette expérimentation, nous avons observé une augmentation de 14% avec FavourQueue. Puisqu'un paquet issu d'une procédure Fast Recovery est placé au début de la fenêtre TCP, FavourQueue évite la perte de cette retransmission. Ainsi, le nombre de reprises avec Fast Retransmit est supérieur avec FavourQueue et la latence observée est donc meilleure puisque les retransmissions sont plus rapides et assurées. A noter que pour un flot d'une taille inférieure à 6 paquets, la reprise est exclusivement effectuée avec un RTO car il n'existe pas assez d'acquittements dupliqués pour générer

un Fast Retransmit. De façon générale, la reprise sur RTO est de 56% pour DropTail contre 38% pour FavourQueue. De plus, la diminution du gain suit l'augmentation de la taille des flots, ce qui signifie que FavourQueue aide la phase d'établissement de connexion TCP.

Conclusion

Nous avons présenté dans cette partie deux contributions de gestion de file d'attente et une solution métrologique permettant de mieux informer les sources du niveau de congestion du réseau. La première contribution, KRED, améliore la stabilité de la file d'attente RED afin de permettre une meilleure signalisation binaire de la congestion. La seconde solution présentée (ECN*), s'attache à offrir aux sources TCP une information quantitative plus que qualitative afin d'affiner leur réaction face à la congestion. ECN* est à différencier des deux AQMs présentées car cette solution cherche avant tout à augmenter le niveau d'information du drapeau ECN. Enfin, la dernière propose une amélioration de DropTail permettant d'accélerer le transfert des données. Bien que ces contributions impliquent le cœur du réseau, elles n'impliquent pas les extrémités et FavourQueue est une solution complétement indépendante qui pourrait grandement améliorer le service *best-effort*. De plus, si l'on regarde la tendance actuelle où les protocoles de transport implémentent de plus en plus des mécanismes basés sur le delai (*delay-based congestion control*), FavourQueue serait une solution idéale ces nouveaux protocoles *delay-based*.

Références sur le chapitre

[1] **Rémi Diana, Emmanuel Lochin ECN verbose mode : a statistical method for network path congestion estimation - extended version, Elsevier Computer Networks**

[2] **Rémi Diana, Emmanuel Lochin ECN verbose mode : a statistical method for network path congestion estimation, IEEE Infocom Work in Progress, 2010**

[3] **Emmanuel Lochin, Bruno Talavera, Managing Internet routers congested links with a Kohonen-RED queue, Elsevier Engineering Applications of Artificial Intelligence, 2011**

[4] **Emmanuel Lochin, Bruno Talavera, Managing network congestion with a Kohonen-based RED queue, IEEE ICC, 2008**

[5] **Eugen Dedu and Emmanuel Lochin, A Study on the Benefit of TCP Packet Prioritisation, The 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP) Weimar, Germany, 2009**

[6] **Pascal Anelli, Emmanuel Lochin, Rémi Diana FavourQueue : a Stateless Active Queue Management to Speed Up Short TCP Flows, Soumis à publication**

[7] Ofcom : Office of Communications, "UK Broadband Speeds 2008 : Research report", January 2009

[8] Toby Moncaster and Louise Krug and Michael Menth and João Taveira Araújo and Steven Blake and Richard Woundy, The Need for Congestion Exposure in the Internet, Internet Engineering Task Force, Internet Draft, `draft-moncaster-conex-problem-00`, March, 2010

[9] Philip Eardley, Congestion Exposure : Were All in This Together, IETF Journal, Volume 5, Issue 3, January 2010

[10] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, Sudipta Sengupta, and M. Sridharan, DCTCP : Efficient Packet Transport for the Commoditized Data Center, ACM SIGCOMM 2010, New Delhi, India, August 2010

[11] Ciullo, D. and Mellia, M. and Meo, M., Two Schemes to Reduce Latency in Short Lived TCP Flows, IEEE Communications Letters, 2009

[12] Andrew L.H. Lachlan et Al., Towards a Common TCP Evaluation Suite, Protocols for Fast, Long Distance Networks (PFLDnet), Mar 2008

[13] Bob Briscoe et Al., Re-ECN : Adding Accountability for Causing Congestion to TCP/IP , IETF Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-06.txt, juillet 2008

[14] I. A. Qazi, L. L. H. Andrew and T. Znati, Congestion Control using Efficient Explicit Feedback, IEEE INFOCOM, 2009

[15] Sally Floyd, TCP and explicit congestion notification, ACM Computer Communcation Review, vol. 24, num. 5, 1994

[16] Dina Katabi, Mark Handley, Charlie Rohrs, Congestion Control for High Bandwidth-Delay Product Networks, ACM SIGCOMM, 2002

[17] David D. Clark and John Wroclawski and Karen R. Sollins and Robert Braden, Tussle in cyberspace : Defining tomorrows Internet, ACM SIGCOMM, 2002

[18] Jong-hwan Kim and Ikjun Yeom, Reducing Queue Oscillation at a Congested Link, IEEE Transactions on Parallel and Distributed Systems, 2008

[19] Teuvo Kohonen, Self-Organizing Maps, Springer Verlag Information Sciences, vol 30, Third Extended Edition, 2001

[20] Robert Beverly and Karen Sollins, The Role of Learning in Network Architecture, Research Abstract of the Computer Science and Artificial Intelligence Laboratory (CSAIL), 2007, `http://publications.csail.mit.edu/abstracts/abstracts07/beverly2/beverly2.html`

[21] A. Makarovic, Machine intelligence 12 : towards an automated logic of human thought, Clarendon Press, New York, NY, USA, 1991

[22] L. Le and J. Aikat and K. Jeffay and F. Smith, The effects of active queue management on web performance, In Proceedings of ACM SIGCOMM, 2003

[23] Bob Briscoe and Arnaud Jacquet and Toby Moncaster and Alan Smith, Re-ECN : Adding Accountability for Causing Congestion to TCP/IP, draft-briscoe-tsvwg-re-ecn-tcp-06.txt, Internet Engineering Task Force, Internet Draft, 2008

[24] M. May and J. Bolot and C. Diot and B. Lyles, Reasons Not to Deploy RED, IEEE IWQoS, 1999

[25] T. Ziegler and S. Fdida and C. Brandauer, Stability Criteria of RED with TCP Traffic, IFIP ATM&IP Working Conference, 2001

[26] Steven H. Low and Fernando Paganini and Jiantao Wang and John C. Doyle, Linear stability of TCP/RED and a scalable control, Computer Networks, 2003

[27] Sally Floyd and Van Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, 1993

[28] Aleksandar Kuzmanovic, The power of explicit congestion notification, ACM SIGCOMM Computer Communication Review, 2005

[29] A. Medina and M. Allman and S. Floyd, Measuring the Evolution of Transport Protocols in the Internet, ACM SIGCOMM Computer Communication Review, 2005

[30] Xuan Chen and John Heidemann, Preferential Treatment for Short Flows to Reduce Web Latency, Elsevier Computer Networks, 2003

[31] Konstantin Avrachenkov and Urtzi Ayesta and Patrick Brown and Eeva Nyberg, Differentiation Between Short and Long TCP Flows : Predictability of the Response Time, IEEE INFOCOM 2004

[32] Idris A. Rai and Ernst W. Biersack and Guillaume Urvoy-Keller, Size-based Scheduling to Improve the Performance of Short TCP Flows, IEEE Network 2005

DTN is an overlay-like network ?

Un autre débat sur la caractérisation des réseaux DTN, ce qui m'intéresse dans ce texte est la nouvelle notion de fiabilité qui y est donnée et notamment ce concept de «délégation des droits de retransmission fiable »que notre proposition protocolaire Tetrys permet de solutionner de bout en bout et contrairement à ce qui est argumenté, «de façon scalable ».

Question : DTN is an overlay-like network ?

L. Wood : Yes : such as the end-to-end principle being a countermeasure against an unreliable underlying network infrastructure

K. Fall : No : The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)

W. Eddy : In mobility architectures, there is sometimes the concept of «delegation of signaling rights »which is basically the concept that some node that is more-capable or more-central within the network can take burden off of either the limited wireless links or the edge nodes themselves, by aggregating and proxying the various bits of signaling involved in mobility management. This allows the architecture to function as it scales up in various terms, which would be impossible otherwise. DTN bundle protocol uses the notion of «delegation of retransmission rights »or «delegation of reliable transmission rights »to a similar end. Though applications could ultimately be forced to be responsible, this doesn't scale, just like though mobiles can do their own signaling, it doesn't scale. Also, removing the need for an ack channel and the need for storage while waiting for acks is a big win. It allows the application on a challenged host with limited resources to send its data and move on with its mission. Erasure coding has some tradeoffs and isn't always a win.

Chapitre 7

Le protocole Tetrys

Les performances et applications des codes correcteurs d'erreurs ont été largement étudiées au sein de la communauté réseaux. Ces codes visent à renforcer la fiabilité du lien ou du chemin de bout de bout en réduisant le taux de bits en erreur et de paquets perdus. Aucune variante n'a vocation à fournir une fiabilité totale. Cependant, grâce à un mécanisme de retransmission hybride, il est possible de retransmettre des paquets suivant le mécanisme ARQ (Automated Retransmission Request). Il en résulte qu'une succession d'erreurs qui ne pouvaient être masquées par le mécanisme de fiabilisation devaient attendre au minimum un RTT pour être corrigées. Evidemment, les applications temps-réel évoluant déjà au-dessus d'un réseau *best-effort* ne peuvent se satisfaire d'une telle contrainte. Les concepts apportés par la théorie des codes permettent aujourd'hui de combler le fossé entre fiabilisation et fiabilité totale en s'abstrayant du concept d'ARQ.

Ce chapitre présente un mécanisme innovant nommé Tetrys, dont l'une des caractéristiques est de pouvoir reconstruire les pertes dans un temps paramétrable et indépendant du RTT. De plus, sa configuration est grandement simplifiée car il ne nécessite que l'estimation du PLR moyen sans hypothèse sur la sporadicité des pertes et la taille des rafales de paquets perdus.

Les applications ciblées par ce mécanisme de fiabilité sont celles nécessitant une fiabilité totale avec contrainte de délai. Il s'avère qu'à taux de redondance égal, des applications telles que la VoIP et la vidéo-conférence sont bien plus performantes lorsque les flux sont protégés par le mécanisme Tetrys que par les mécanismes FEC ou H-ARQ classiques.

Tetrys est né à l'ISAE fin 2007 après une discussion mêlant transport et théorie des codes avec Jérôme Lacan. C'est à ce moment que nous avons trouvé une application au concept présenté ci-après. Ce protocole de fiabilité est à la base de la thèse de Pierre-Ugo Tournoux qui a fortement contribué en finalisant un prototype complet et en démontrant les propriétés de ce protocole.

Introduction

Les codes FEC (Forward Error Correction), ou encore code FEC de niveau application (AL-FEC), ont gagné en popularité ces dernières années car ils permettent de réduire de façon drastique les pertes sur un canal à effacement. Leur principe est le suivant : à un ensemble de k paquets sources, les codes AL-FEC associent $n-k$ paquets de redondance de manière à ce que la récupération d'au moins k paquets parmi les n initiaux permettent la récupération des k paquets originels. En pratique, seul les codes optimaux (aussi appelés MDS) possèdent cette propriété tandis que d'autres type de codes (tels que LDPC, Fountain ou Raptor Codes) nécessitent de recevoir plus que k paquets afin de reconstruire la séquence initiale. Parmi les codes optimaux, certains sont dits de type systématique et envoient les k paquets initiaux en clair (non modifiés) et les $n-k$ paquets de redondance qui contiennent une combinaison des k paquets initiaux. A l'inverse, certains codes transforment les k paquets initiaux en n paquets encodés et aucun d'eux ne contient l'information initiale. Dans le second cas, on ne pourra récupérer que k ou aucun paquet initial alors que les codes systématiques permettent l'utilisation des paquets en clairs correctement reçus et ce, même si plus de $n-k$ pertes ont été observées. On note le taux de redondance le ratio $(n-k)/n$ et le taux d'encodage le rapport k/n. Ainsi, un taux d'encodage proche de 1 signifie une faible quantité de redondance et donc une faible tolérance aux pertes tandis qu'un taux proche de 0 signifie une forte tolérance aux pertes mais également synonyme d'une forte surcharge dans le réseau.

Quel qu'en soit l'utilisation, les avantages des AL-FEC reposent sur deux points principaux. Le premier est le fait que les pertes deviennent anonymes, dans le sens où $n-k$ pertes parmi n peuvent être tolérées avec uniquement $n-k$ paquets de redondance. Le second avantage se fait au niveau du délai de récupération d'une perte par rapport aux schémas ARQ. La perte de x paquets (avec $x < k$) peut être reconstruite après la réception de k paquets parmi les n initiaux. Les débits courants étant suffisamment grands, la durée entre l'instant de la première perte parmi les n paquets et l'instant de réception du k-ème paquet est souvent inférieure comparé au temps requis pour demander à la source une retransmission. Ceci est particulièrement vrai lorsque le RTT est important. Cette seconde propriété rend l'utilisation des AL-FEC particulièrement avantageuse dans le cas d'applications temps réel où par définition, les données deviennent obsolètes si elles ne sont pas reçues avant l'écoulement d'un certain délai.

Positionnement de Tetrys

On constate donc que les propriétés qui rendent les FEC adaptées aux applications temps-réel sont rompues par H-ARQ puisque des retransmissions sont incontournables et donc un délai d'au moins un RTT est nécessaire pour reconstruire des pertes qui n'ont pu être masquées par les FEC. Une caractéristique principale de Tetrys est de s'affranchir des re-

transmissions par l'introduction d'un concept de codage élastique, permettant le maintien des caractéristiques de FEC pour la fiabilité totale, à savoir la reconstruction des pertes en un temps indépendant du RTT. Ce mécanisme autorise un faible délai de reconstruction et peut opérer sur des réseaux bi-directionnels de type *best-effort*, sur liens asymétriques et caractérisés par un taux de perte important. Le terme de fenêtre élastique provient du fait que contrairement aux codages classiques, le nombre k de segments de données encodés dans les paquets de redondance n'est pas fixe et les informations contenues d'une opération de codage à l'autre ne sont pas forcement disjointes. Ces paquets de redondance contiennent une combinaison de toute les informations envoyées par l'émetteur et dont la réception n'a pas encore été accusée (les acquittements sont optionnels mais permettent de diminuer la complexité en temps de calcul). Contrairement aux codes FEC classiques qui utilisent des coefficients particuliers pour assurer que la matrice soit inversible, le caractère élastique de la fenêtre impose que les coefficients soient choisis de manière aléatoire au sein du corps fini $\mathbb{F}_q$. La probabilité d'inversion de la matrice nécessaire au décodage demeure suffisamment importante pour ne pas avoir d'impact sur les performances du mécanisme. Les acquittements permettent à l'émetteur de savoir quels paquets ont été reçus et donc ceux qu'il pourra supprimer de la fenêtre d'encodage. Seul le nombre de paquets de données inclus dans les paquets de redondance joue sur la complexité du mécanisme en termes de temps de calcul. C'est cette propriété qui permet à Tetrys d'être tolérant face aux pertes de ses acquittements ainsi qu'aux réseaux fortement asymétriques.

Présentation du mécanisme

Soit P_x le x-ème paquet de données envoyé et P_1 le premier paquet. La taille en nombre d'octets Sz des paquets de données envoyés étant fixe, les données incluses dans P_x sont celles allant de $Sz * (x-1)$ à $Sz * x$ en supposant que la numérotation des octets commence à 0. $R_{(i..j)}$ dénote le paquet de redondance qui contient une combinaison linéaire de tous les paquets de données P_k avec k allant de i à j. Chaque paquet de redondance est construit comme suit : $R_{(i..j)} = \sum_{k=i}^{j} \alpha_k^{(i,j)}.P_k$ où les coefficients $\alpha_k^{(i,j)}$ appartiennent au corps fini $\mathbb{F}_q$ et la multiplication avec le P_x est faite par partie de même nombre de bit que ceux requis pour coder le corps fini [5]. Nous choisissons d'envoyer un acquittement à une fréquence donnée : F_{SACK}. Cet acquittement peut contenir un bloc SACK tel que défini dans le RFC 2018 et qui contiendra tous les paquets correctement reçus ou décodés. L'ensemble W_{NACK} contient tous les paquets envoyés mais non accusés par le récepteur.

La notation et l'expression du taux de codage définie pour les AL-FEC reste valable à l'exception de $n-k$ qui vaut toujours 1. Ainsi, après l'émission de k paquets de données, un paquet de redondance $R_{W_{NACK}}$ est construit avec tous les paquets envoyés mais non accusés, c'est-à-dire l'ensemble des paquets de W_{NACK}. De son coté, le récepteur effectue l'opération de décodage inverse pour reconstruire les paquets perdus. L'exemple suivant

illustre le fonctionnement du mécanisme et la récupération sur une perte.

Supposons que la source ait envoyé les cinq paquets suivants (P_1, P_2, $R_{(1,2)}$, P_3, $R_{(1..3)}$). Si (P_2,P_3) sont perdus, les paquets P_1, $R_{(1,2)}$ et $R_{(1,3)}$ suffisent à récupérer P_2 et P_3 et le décodage est similaire à celui de FEC classiques. Par contre, si P_1 et P_2 sont perdus, il faut au préalable soustraire P_3 de $R_{(1..3)}$) en effectuant $R'_{(1..3)} = R_{(1..3)} - \alpha_3^{(1,3)}.P_3$. On obtient donc : $(R_{(1,2)}, R'_{(1..3)})^T = G.(P_1, P_2)^T$ où G est la matrice 2×2 :

$$G = \begin{pmatrix} \alpha_1^{(1,2)} & \alpha_2^{(1,2)} \\ \alpha_1^{(1,3)} & \alpha_2^{(1,3)} \end{pmatrix} \tag{7.1}$$

Ainsi, la reconstruction de (P_1, P_2) à partir de $(R_{(1,2)}, R'_{(1..3)})$ n'est possible que si G est inversible car si G^{-1} existe, il en résulte $(P_1, P_2) = G^{-1}.(R_{(1,2)}, R'_{(1..3)})$. La probabilité d'inversion de la matrice dépend du choix initial des coefficients de codage α. Bien que dans le cas de Tetrys, les variations de taille de la fenêtre nous contraignent à utiliser des coefficients aléatoires, cette probabilité demeure suffisamment proche de 1 et ce à partir d'un corps fini de taille 8. La figure 7.1 illustre le fonctionnement de Tetrys avec des acquittements. Le taux de redondance est de $2/3$ avec $n = 3$ et $k = 2$ et correspond donc à un paquet de redondance qui est construit et envoyé tous les 2 paquets de données.

Dans cet exemple, le paquet P_2 est perdu tandis que le paquet de redondance $R_{(1,2)}$ qui est correctement reçu permet de reconstruire P_2. Le paquet accusant la réception du paquet P_1 est perdu, ce qui implique que P_1 continuera a être inclus dans les prochains paquets de redondance sans conséquence sur le devenir du transfert. Les paquets suivant, P_3, P_4 et $R_{(1..4)}$ sont perdus, contrairement au mécanisme H-ARQ, aucun d'entre eux ne nécessite d'être retransmis puisqu'ils seront reconstruits grâce à la réception des paquets allant de P_5 à $R_{(1..8)}$. En supposant que l'application source soit à débit constant et émet un paquet toutes les $10ms$ et que le délai de transit soit symétrique et égal à $100ms$ (soit un RTT de $200ms$), le délai observé entre la génération de P_3 et sa réception effective serait de $160ms$ contre un minimum $320ms$ pour H-ARQ. Comme pour la matrice (7.1) le récepteur peut reconstruire P_3, P_4 en soustrayant dans un premier temps les paquets source de manière à obtenir $(R'_{(1..6)}, R'_{(1..8)})$. L'étape suivante consiste alors en l'inversion puis la multiplication par $(R'_{(1..6)}, R'_{(1..8)})$ de la matrice 2×2. Enfin, cet exemple illustre bien que la voie retour n'est utilisée que pour diminuer la complexité d'encodage au niveau de la source, en effet, à la réception du second acquittement (le premier ayant été perdu), la source recommence son processus de codage à partir du paquet #9, ce qui permet à la fois au récepteur de diminuer la complexité du codage et de s'assurer que les précédents paquets source ont bien été reçus.

Performance et complexité

Il est clair que le point clé de la performance du mécanisme est lié au délai entre l'instant où un paquet de données est perdu et l'instant où le récepteur aura collecté suffisamment

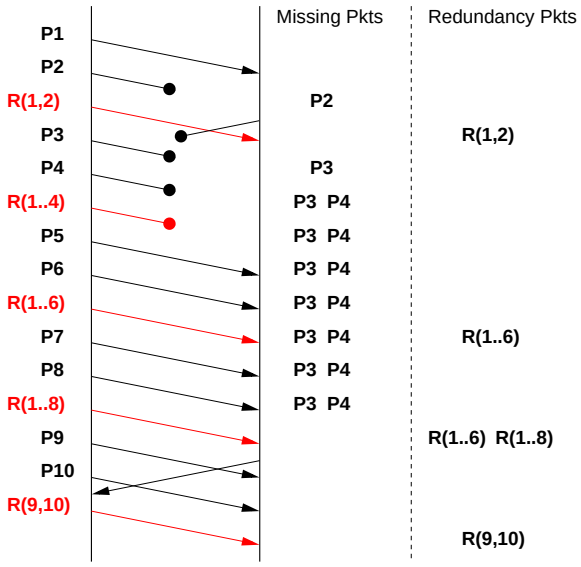


FIGURE 7.1 – Exemple simple de transfert unidirectionnel avec acquittements pour un taux d'encodage de $2/3$

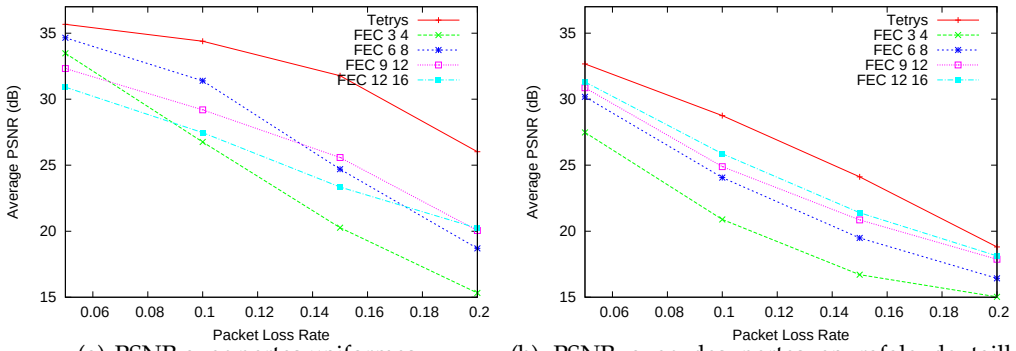
de paquets de redondance pour le reconstruire. Plus formellement, on nomme "temps de récurrence" le délai entre le moment où une perte est détectée et le moment où le nombre de pertes devient inférieur ou égal au nombre de paquets de redondance reçus. Ce délai va être impacté par trois éléments : le taux de codage, le taux de perte et la distribution des pertes (uniformes ou par rafales). Un modèle analytique permet de calculer le délai moyen pour des pertes uniformes et un heuristique basé sur une distribution de Weibull permet de généraliser ce temps à tout type de canal à effacement quelque soit la taille de la rafale pertes. Nous avons également montré que les buffers des émetteurs et récepteurs étaient bornés et évoluaient en fonction de $(RTT + F_{SACK}) * RTT) *$ débit d'emission. L'ensemble de ces travaux est disponible dans [1].

Application à fiabilité partielle

En guise d'exemple d'application temps-réel ne requérant qu'une fiabilité partielle, nous avons choisi l'application de vidéo conférence. Du point de vue de la transmission sur le réseau, la vidéo possède des caractéristiques qui rendent sa protection plus délicate comparé à d'autres trafics comme la VoIP. Tout d'abord, le délai de bout en bout ne doit pas dépasser les 100ms afin de préserver l'interactivité de la conférence.

Nous comparons Tetrys à des codes FEC idéaux avec différentes tailles de blocs et un taux de codage constant, en l'occurrence (k,n)= { (3,4), (6,8), (9,12), (12,16) }. Comme pour la VoIP, nous répétons les mesures avec des pertes uniformément distribuées ainsi qu'avec des pertes en rafale de taille moyenne 2 et 3 suivant un modèle de Gilbert Elliot. Nous avons utilisé la dernière recommandation en date en matière de codec vidéo, à savoir H.264 avec

l'implantation JM 15.1 H.264/AVC [1]. La séquence video utilisée est celle de Foreman au format CIF (352x288) avec 15 trames par secondes dont une trame I et 14 trames P. Les trames B ne sont pas utilisées. Le débit moyen ainsi produit par l'encodeur vidéo est de $384kbps$ et les paquets sont de 500 octets chacun. Nous fixons à $200ms$ le délai maximum tolérable de bout en bout. Tous les paquets arrivant après cette limite sont jetés. La séquence dure 10 secondes que nous répétons 20 fois afin d'obtenir des résultats représentatifs, ce qui fait un total de 3000 trames pour 200 secondes. Cette configuration correspond aux recommandations faites par Stephan Wenger [4]. Pour évaluer la qualité de la vidéo reçue, nous avons utilisé le framework d'évaluation vidéo Evalvid dans lequel la métrique utilisée est le PSNR (Peak Signal to Noise Ratio).



(a) PSNR avec pertes uniformes

(b) PSNR avec des pertes en rafale de taille moyenne 3 (Gilbert Elliot)

FIGURE 7.2 – PSNR obtenu par Tetrys et FEC sous différents types de pertes.

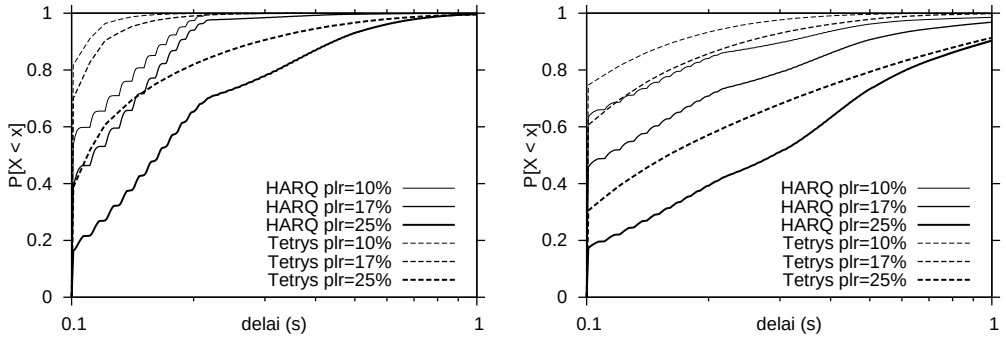
Sans surprise, à taux de codage équivalent, la qualité de la vidéo est meilleure lorsqu'elle est protégée avec Tetrys qu'avec FEC. Dans le cas des pertes uniformément distribuées, la figure 7.2 présente un gain de 7.19dB pour un taux de perte de 15% obtenu par Tetrys comparé au FEC optimal (FEC 6 8). De plus, la chute du PSNR n'est que de 4bB pour Tetrys lorsque le taux de perte passe de 5% à 15%. Il reste ainsi au-dessus de 30dB ce qui correspond à une image de bonne qualité.

Pour des pertes en rafale, la figure 7.2(b) montre la même tendance bien que les gains soient moins significatifs lorsque la taille des rafales augmente (de $2.7dB$ pour des rafales de taille moyenne 3). Cela s'explique notamment par le fait que la correction des pertes en rafale requiert plus de redondance que pour des pertes uniformes. A la vue de leur PSNR, aussi bien FEC que Tetrys sont sous-dimenssionnés.

1. Voir H.264/AVC JM Reference Software, `http://iphome.hhi.de/suehring/tml/`

Application à fiabilité totale

Pour les applications requérant une fiabilité totale, Tetrys ne peut pas être comparé à FEC et c'est donc uniquement les performances d'H-ARQ qui vont être étudiées. Il existe une pléthore d'H-ARQ adaptatifs (noté de type II) qui ajustent les paramètres du codage (taux de codage, taille de bloc, surprotection des retransmissions, ...) au fil de la transmission en fonction des conditions réseaux. Cependant, aucun ne semble faire l'unanimité. Dans notre étude, le taux de pertes ne varie pas avec le temps, il n'y a donc aucun avantage à utiliser des H-ARQ de type II. Par contre, le fait de renforcer la protection des retransmissions peut éviter une augmentation du délai supplémentaire et ce, quel que soit le contexte. Malgré tout, par souci de clarté nous ne comparerons à Tetrys qu'un modèle H-ARQ de type I (i.e. non adaptatif).



(a) Pertes uniformément distribuées (b) Pertes en rafales de taille 3

FIGURE 7.3 – CCDF du délai de livraison des paquets en ordre pour Tetrys et H-ARQ avec un taux de redondance de $1/3$, un RTT de $200ms$ et un débit de 10 paquets par seconde

Le délai est la métrique utilisée pour comparer les performances de ces différents mécanismes. Il correspond au temps entre l'émission du paquet par la source et celui où il est transmis (dans l'ordre) à la couche supérieure au mécanisme de fiabilité du coté du récepteur. Les figures 7.3(a) et 7.3(b) présentent la complémentaire de la fonction de répartition (CCDF) des délais de chaque paquet avec une échelle logarithmique pour l'axe des abscisses. Ce choix de représentation se justifie par le fait que les applications sont différemment impactées par les variations de délais et une simple moyenne n'aurait pas été suffisamment descriptive. Les paramètres de simulation utilisés sont les suivants : un taux de codage de $2/3$; chaque simulation dure 100 secondes correspondant à une réception de 10^5 paquets ; les pertes sont respectivement distribuées de manière uniforme et par rafales de moyenne 3. Pour chacune de ces figures, la distribution des délais est présentée pour H-ARQ et Tetrys et pour des taux de pertes de 10%, 17% et 25%. On constate qu'à délai égal, la probabilité d'être inférieur à ce dernier est toujours significativement supérieure pour Tetrys. Cela reste vrai lorsque l'on compare Tetrys avec 17% de pertes et H-ARQ avec 10% de pertes. A titre d'exemple, avec un taux de perte de 17% et des pertes uniformément distribuées, la probabilité d'obtenir un

délai inférieur à $150ms$ est de 67% pour H-ARQ contre 94% pour Tetrys. Il est à noter que lorsque le taux de perte approche du taux de redondance, le temps de récurrence est tel que le délai rencontré par Tetrys sera supérieur à ceux d'H-ARQ. Cependant, comme pour la VoIP, il est probable que les performances d'H-ARQ ne soient également pas acceptables avec un tel ratio entre le taux de redondance et le taux de perte.

Conclusion

Nous venons de présenter un mécanisme de fiabilité permettant une fiabilité totale dans sa forme générique. Tetrys est résistant aux pertes d'acquittements et permet d'obtenir des délais de récupération des paquets perdus paramétrables et indépendants du RTT. À taux de redondance égal, les performances de Tetrys sont sensiblement meilleures que celles obtenues par FEC pour la fiabilité partielle. Pour la fiabilité totale, au prix d'une charge plus légère pour le réseau (Tetrys ne fait aucune retransmission), Tetrys permet un délai de récupération également inférieur à celui des H-ARQ de type I. Ses caractéristiques en termes de complexité en temps et en espace autorisent son implémentation sans restriction particulière. Le point faible du mécanisme, qui est l'augmentation du temps de récurrence lorsque le taux de perte approche le taux de redondance, peut être contourné par une simple solution d'ingénierie telle que l'adaptation du taux de redondance en fonction du taux de perte observé de manière à ce que le délai reste dans les bornes requises par l'application.

Ce mécanisme a également montré ses bénéfices dans d'autres contextes applicatifs et notamment pour la diffusion de flots type *streaming-like* au sein de réseaux DTN mobile (DTMN) [2] en collaboration avec Jérémie Leguay et dans le contexte de la mobilité pour la fiabilisation efficace des handovers verticaux [3] au sein d'un protocole de management de la mobilité en collaboration avec Henrik Petander. L'ensemble des travaux concernant le cœur du mécanisme est disponible dans [1].

Références sur le chapitre

[1] **On-the-Fly Coding for Time-Constrained Applications, Pierre-Ugo Tournoux, Emmanuel Lochin, Jérôme Lacan, Amine Bouabdallah, Vincent Roca, IEEE Transactions on Multimedia 2011**

[2] **Robust Streaming in Delay Tolerant Networks, Pierre Ugo Tournoux, Emmanuel Lochin, Jérémie Leguay, Jérôme Lacan, IEEE International Conference on Communications (IEEE ICC 2010), Cape Town, South Africa, May 2010**

[3] **A Packet Error Recovery Scheme for Vertical Handovers Mobility Management Protocols, Pierre-Ugo Tournoux, Emmanuel Lochin, Henrik Petander, Jérôme Lacan, ICST MobiQuitous, Sydney, Australia, December 2010**

[4] H.264/AVC over IP, Stephan Wenger, IEEE Transactions on Circuits and Systems for Video Technology, 2003

[5] Effective erasure codes for reliable computer communication protocols, Luigi Rizzo, ACM Computer Communication Review, 1997

Chapitre 8

Réseaux anarchiques (travaux prospectifs)

Nous nous intéressons dans ce chapitre aux réseaux dits "anarchiques" qui sont caractérisés par une absence totale de contrôle de congestion aux extrémités. Contrairement à TCP qui domine les hôtes de l'Internet et qui assure une utilisation efficace et équitable du réseau via une adaptation volontaire et collaborative du débit ; dans les réseaux anarchiques, les sources peuvent émettre à la vitesse maximale de leurs interfaces sans se soucier d'une quelconque équité. Ce nouveau concept a été initialement introduit par Alexandre Snoeren dans [2] où l'auteur est le premier à remettre en question l'utilité du contrôle de congestion de l'Internet. Ce concept a été suivi de contributions théoriques visant à vérifier la véracité et la pertinence de l'idée [3, 4]. Dans ces contributions, les auteurs montrent que la tragédie communément admise de l'effondrement du réseau n'aurait pas lieu d'être dans l'Internet d'aujourd'hui. Cependant, bien qu'en termes d'efficacité de débit les réseaux anarchiques ont un intérêt indéniable, du point de vue de l'application, le débit utile n'est pas optimal et un service fiable n'est pas envisageable. Aussi, les auteurs présupposent l'utilisation d'un code parfait [1] au sein de ces réseaux. Bien évidemment, l'utilisation pragmatique de ce type de réseaux est donc relatif à l'existence purement théorique de ce code.

C'est dans ce contexte qu'avec Pierre-Ugo Tournoux, nous nous sommes posé la question de l'utilité de Tetrys au sein de ces réseaux. Dans sa thèse [1], Pierre-Ugo a complété ses précédents travaux par l'implémentation d'un mécanisme de transport anarchique basé sur Tetrys tout en adressant des points tels que l'efficacité et l'équité obtenue en fonction de différentes topologies réseaux réalistes. De plus, nous avons exploré la possibilité d'utiliser des applications temps réel sur de tels réseaux.

1. Un code est dit parfait s'il ne contient aucune redondance inutile. Dans notre cas, cela signifie que toute la redondance émise est utile et donc que le taux de redondance utilisé est optimal.

Topologies, Congestion Collapse et Contrôle de Congestion

Le phénomène de *congestion collapse* reste dans la mémoire collective du monde des réseaux un synonyme d'absence de contrôle de congestion. Ce phénomène fut initialement observé avec l'utilisation de la première version de TCP qui effectuait uniquement un contrôle de flux via une fenêtre de taille fixe. Dans ce contexte, lorsque la capacité d'un lien était atteinte, les files d'attente se remplissaient et l'accroissement du RTT résultant de l'augmentation de la taille des files pouvait se faire plus rapidement que le rafraichissement du RTO. Ainsi, des paquets en transit pouvaient être considérés comme perdus. De plus, à la détection d'une perte, les paquets suivants étaient considérés comme perdus et retransmis (phénomène connu sous le nom de retransmission abusive ou *spurious RTO*). Ces paquets déjà reçus augmentaient alors la probabilité de perte. Un mécanisme d'espacement croissant des retransmissions permettait néanmoins au réseau de rester stable, bien qu'inefficace (voir RFC 896).

Ce phénomène n'est pas uniquement lié à une absence de contrôle de congestion, la topologie est aussi un facteur à prendre. En effet, certaines topologies (comme la topologie en papillon) sont connues pour générer plus de paquets "morts" que certaines autres. Un paquet est dit "mort" si il a déjà occupé de la bande passante sur des goulots d'étranglement avant d'être jeté par un goulot d'étranglement en aval de ces derniers. Ainsi, la bande passante occupée sur le ou les précédents goulots d'étranglement se fait au détriment des autres flots qui auraient pu en bénéficier. Au sein d'une topologie en papillon, on observe généralement une réduction drastique du débit utile mais il est à noter tout de même que le réseau reste utilisable. Il existe cependant des topologies particulières (cycliques) où la proportion de paquets morts implique que le débit peut tendre vers zéro [5, 6]. On retrouve ces topologies chez certains fournisseurs d'accès à Internet qui architecture leur réseau en anneau (c'est le cas notamment du fournisseur d'accès Free). Cependant, ce problème d'effondrement est également prévenu par un contrôle d'accès efficace.

Pourquoi déployer un réseau anarchique

Quelle qu'ait été l'importance de la topologie ou des retransmissions, le contrôle de congestion TCP Tahoe proposé par Van Jacobson a solutionné le *congestion collapse* assurant ainsi une meilleure robustesse d'Internet, sa pérennité et une relative équité entre les utilisateurs. Néanmoins, ce dernier peut être critiqué sur deux points. D'une part, le trafic obtenu est largement sous-optimal (aussi bien en termes de débit utile qu'en termes d'équité [9]) et d'autre part, cela suppose une volonté collaborative de chaque utilisateur. En effet, un utilisateur malveillant ou mal configuré peut occuper toute la bande passante et rendre ainsi le réseau inutilisable pour les utilisateurs *congestion-controlled*. Cependant, du point de vue de l'Internet, un simple utilisateur n'aurait un impact que sur son réseau local puisque les opérateurs réseaux effectuent généralement un contrôle d'accès [10]. C'est une des raisons

pour laquelle Raghavan et Snoeren remettent en cause dans [2] la pertinence même de l'utilisation d'un contrôle de congestion. Notamment, ils proposent que chaque flot puisse émettre à une vitesse arbitraire ce qui, d'après ce que nous avons vu, devrait mener à un *congestion collapse* de par le problème des retransmissions et des paquets morts résultant de la topologie. Il proposent d'écarter le problème des retransmissions via l'hypothèse de l'existence d'un mécanisme de codage qui, contrairement à TCP, permettrait que chaque paquet reçu à la destination lui soit utile (*i.e.* utilisation d'un code optimal).

Enfin, un dernier point à noter est qu'une attaque par déni de service, distribuée ou pas, ne peut pas aboutir sur un tel réseau.

Comment déployer un réseau anarchique

L'utilisation de file d'attente active (AQM) est considérée comme incontournable dans les réseaux anarchiques. En effet, il faut d'une part prévenir l'apparition excessive de paquets morts et d'autre part, il faut assurer une certaine équité entre les utilisateurs de manière à ce que le débit obtenu par un flot ne soit pas déterminé que par le débit de son lien d'accès au goulot d'étranglement. Les algorithmes de Fair Queuing et leurs variantes permettent d'atteindre ce but. Cependant, leur complexité les rend inutilisables en pratique et, dans le contexte des réseaux anarchiques, les mécanismes à perte équitable leur sont préférés puisque bien moins consommateurs de ressources.

Les travaux de [3] et [2] posent l'hypothèse de l'existence d'un mécanisme de codage optimal ou du moins suffisamment efficace pour concurrencer l'efficacité des réseaux actuels mais ne fournissent pas de tels mécanismes. L'autre point pourtant crucial pour l'adoption d'un tel réseau est le fait qu'ils négligent totalement l'impact sur les applications temps réel et leur prise en compte. Ce sont là deux points majeurs que nous cherchons à adresser. En effet, dans [2] l'allocation des ressources n'est pas étudiée et l'impact de la topologie (via les paquets "morts") sur l'efficacité du réseau a été négligé. De plus, les auteurs de [3] montrent que sur des topologies réalistes, la perte d'efficacité est modérée ($\approx 20\%$) avec des files d'attente DropTail et quasi nulle si elles sont de type Fair Dropping.

Tetrys comme mécanisme de codage optimal

Nous proposons donc d'utiliser Tetrys comme mécanisme de codage optimal permettant la mise en œuvre de ce type de réseau. Pour cela, nous proposons une méthode de configuration dynamique des paramètres d'encodage de Tetrys afin d'illustrer son efficacité en tant que mécanisme de transport pour réseaux anarchiques. Les premiers résultats montrent que grâce à Tetrys, un réseau anarchique est capable de concurrencer les réseaux actuels gérés par des sources à contrôle de congestion. De plus, nos premières évaluations montrent que pour des RTT de l'ordre d'une centaine de millisecondes, la distance à la capacité du réseau

obtenue est du même ordre que celle des contrôles de congestion actuels. Pour des RTT plus longs, la version anarchique remplace avantageusement le principe du contrôle de congestion. Du point de vue de l'équité, il s'avère que contrairement à ce qui était pressenti dans de précédents travaux, la solution proposée (couplée à un mécanisme de rejet équitable des paquets sur les routeurs de type *Fair-Drop*) permet aux réseaux anarchiques d'approcher une meilleure allocation *max-min fair* que TCP [2]. Autre point important, une première campagne de mesures montre que pour certaines topologies, l'ajout de file d'attente à partage équitable n'améliore pas grandement le service et donc qu'un réseau anarchique, suivant une règle topologique précise, peut être déployé avec de simples files d'attente DropTail.

Evaluation sur une topologie représentant un FAI national

Ces travaux sont encore à un stade embryonnaire, cependant nous proposons dans cette partie d'illustrer les performances de Tetrys sur une topologie représentant un FAI (qui correspond aux conditions rencontrées par les flots une fois sortis du réseau local). A noter que cette topologie est connue pour générer de façon importante des paquets morts.

Afin de reproduire le réseau d'un FAI, nous proposons dans la figure 8.1 une topologie constituée d'hôtes, de DSLAM, de routeurs d'accès et de routeurs de cœurs. On observe cinq cycles formés par les routeurs d'accès qui correspondent à une desserte régionale, ainsi qu'un cycle formé par le coeur du réseau. Chacun des cycles locaux est formé de 4 routeurs auxquels est rattaché un DSLAM qui connecte 4 *clients ADSL*. Les *cycles locaux*, que l'on nommera par la suite *pétales* par analogie à l'aspect floral de la figure 8.1, sont rattachés à un seul routeur de cœur qui servira à l'identifier. Dans le cas de topologies réelles, on peut considérer que les pertes sont généralement localisées dans les routeurs d'accès puisque le cœur du réseau est sur-dimensionné et qu'un contrôle d'accès est également effectué par les routeurs de bordures. Cependant, les fournisseurs d'accès Internet ne dévoilent généralement par leur topologie au grand public pour des raisons de sécurité évidentes. Nous ne disposons donc pas d'information sur le rapport entre la bande passante des liens qui connecte les pétales aux routeurs de cœur et les liens du routeur de cœur. Afin de ne pas présenter un cas trop optimiste, nous considérons que la congestion peut se produire sur les routeurs de cœur et nous fixons la bande passante des différents liens en conséquence. Ainsi, chacun des 40 flots subit successivement 25% de pertes lors de l'accès au DSLAM ; 16% des pertes sur le dernier lien de la pétale et 40% dans le cœur.

Pour faciliter l'analyse des résultats, chaque flux doit avoir le même délai et être en concurrence avec le même nombre de flots. Pour se faire, chaque paire source/destination doit traverser trois routeurs de cœur (équivalent à deux liens). De plus, si la source appartient à la partie inférieure de la pétale, la destination appartient à la partie supérieure de la pétale de destination. Ainsi, chaque flot traverse neuf liens et le RTT est donc de $180ms$. Sur chaque

2. Une allocation de ressources est équitable si on ne peut augmenter les ressources allouées à un utilisateur qu'au prix d'une diminution des ressources déjà inférieures d'autres utilisateurs.

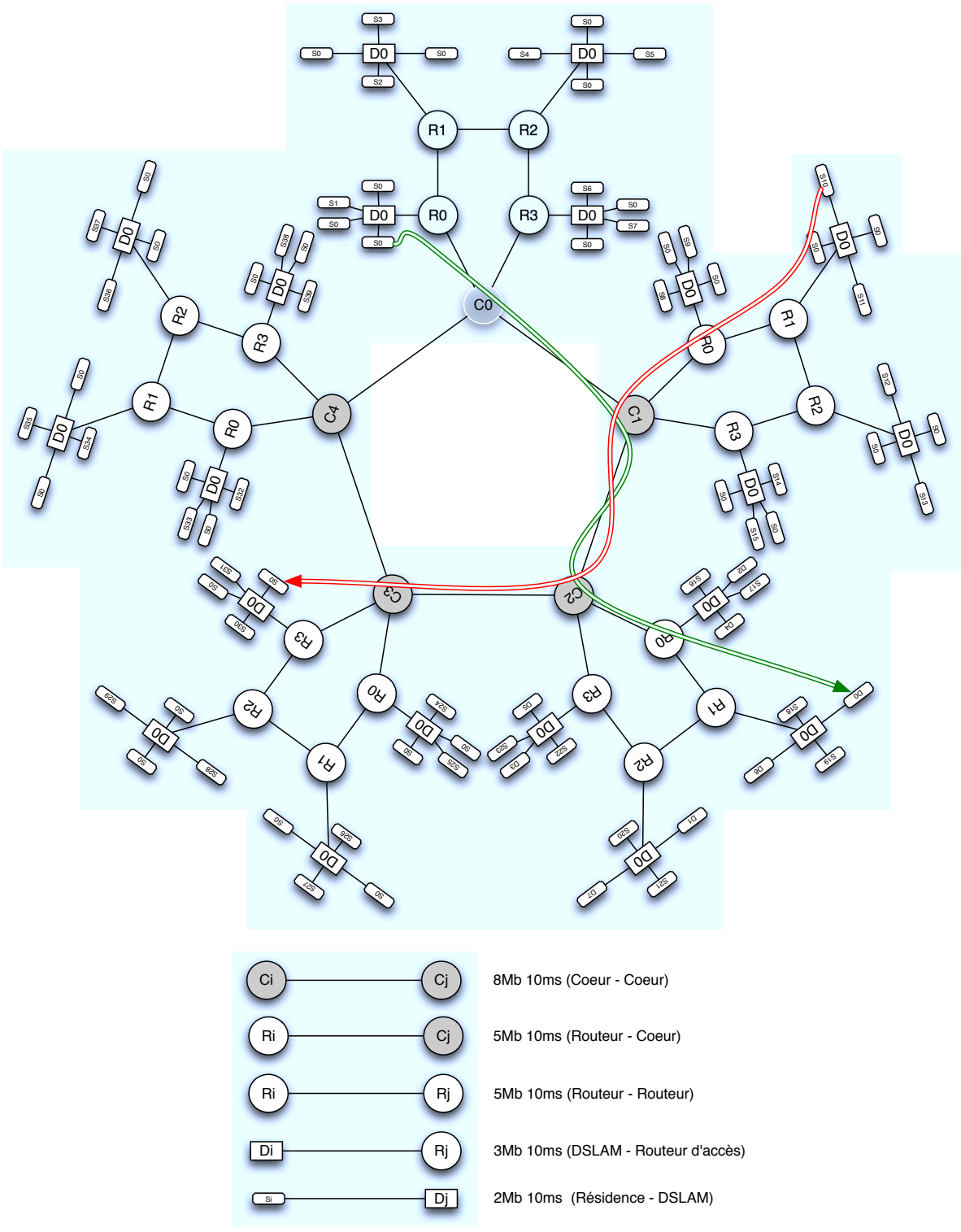


FIGURE 8.1 – Topologie cyclique reproduisant un FAI national (suivant les informations données par l'opérateur Free).

DSLAM, deux utilisateurs émettent un trafic à la vitesse de l'interface tandis que les deux autres sont récepteurs pour du trafic émis par une autre pétale. Leurs débits passent ainsi progressivement de $2Mb/s$ à $0.5Mb/s$.

File d'attente	Efficacité		Equité	
	Tetrys	TCP	Tetrys	TCP
RED	46.1 / 11.6	86.8	0.92	0.98
Fair-Drop	90.0 / 23.5	94.0	0.99	0.99

TABLE 8.1 – Efficacité / ratio entre le nombre de paquets envoyés par Tetrys et le nombre de paquets reçus au total et équité correspondante.

Le tableau 8 montre que les efficacités obtenues par Tetrys sont de 91% et 90%. Il est clair qu'il est nécessaire de déployer des files d'attentes à rejet équitable sur ce type de réseau.

Evaluation des applications à contrainte de temps

File d'attente	Mécanisme	Efficacité	Délai : 95th / moyenne	PLR VoIP
FD(10)	Tetrys	82.6	117/109ms	0%
FD(10)	TCP/UDP	82.4	95/88ms	2.0%
DT (10)	TCP/UDP	82.0	91ms	2.2%
RED(10)	TCP/UDP	73.0	92/86ms	4.4%
FD(20)	Tetrys	88.0	128/120ms	0%
FD(20)	TCP/UDP	86.8	108/94ms	1.7%
DT (20)	TCP/UDP	85.4	111/93ms	2.4%
RED(20)	TCP/UDP	84.2	98/89ms	3.5%
50 FD	Tetrys	84.6	168/155ms	0%
FD(50)	TCP/UDP	91.9	131/106ms	0.3%
DT (50)	TCP/UDP	90.2	154/112ms	2.4%
RED(50)	TCP/UDP	84.2	105/82ms	0.2%
FD(190)	Tetrys	82.2	356/322ms	0%
FD(190)	TCP/UDP	94.8	243/182ms	0.2%
DT (190)	TCP/UDP	94.4	391/277ms	1.4%
RED(190)	TCP/UDP	82.8	105/91ms	0.08%

TABLE 8.2 – Efficacité, délai et taux de perte perçu par la VoIP en fonction du type de file d'attente et de leur taille.

Le dernier point à adresser concerne la tolérance de Tetrys aux pertes résultantes d'un réseau anarchique. Nous étudions le comportement de quatre flots de téléphonie sur Internet (VoIP) combinés à 36 flots de transfert de données. Afin de générer des fluctuations de la bande passante disponible et du taux de pertes, 4 flots démarrent respectivement aux instants $t = \{20, 40, 60 \text{ et } 80s\}$. Dans le cas où les sources cherchent à minimiser leurs débits

d'envoi, les flots de VoIP émettent avec le plus petit taux de redondance possible leur permettant de satisfaire les besoins de l'application, en l'occurrence 95% des paquets doivent être reçus avant $400ms$ si l'on veut obtenir une communication toujours audible. Dans le cas échéant, les flots VoIP émettent au débit de leur interface ($2Mb/s$) et ce, bien que le débit des données utiles ne soit que de $64Kb/s$. L'ensemble des paramètres demeurent ceux de la figure 8.1, chacun des flots pouvant obtenir au minimum $500Kb/s$ (partage équitable sur le goulot d'étranglement avec Fair-Drop). Le RTT est fixé à $180ms$ pour l'ensemble des flots.

Nous comparons les performances obtenues par la version anarchique (Tetrys) et la solution actuelle, à savoir TCP pour les flots FTP et UDP pour la voix sur IP. Afin de ne pas introduire de biais lié à la capacité de la file d'attente, nous répétons les mesures en fixant cette dernière à 10, 20 50 et 190 paquets. Les mesures sont également répétées pour TCP/UDP avec Drop-Tail et RED (en mode *gentle*).

Les métriques que nous utilisons sont d'une part l'efficacité du réseau en termes de paquets utiles reçus par les applications. En d'autres termes, qu'il s'agisse de la VoIP ou de FTP les paquets de redondance reçus ne sont pas comptabilisés. Cela nous permet d'évaluer la perte d'efficacité induite par la combinaison Tetrys et VoIP. D'autre part, nous considérons les conditions perçues par les flux de VoIP, à savoir le délai de bout en bout moyen et le 95^{eme} percentile ainsi que le taux de pertes.

Le premier constat qui peut être fait d'après les résultats du tableau 8.2 est que dans tous les cas de figures testés, les contraintes de l'application ont été satisfaites aussi bien avec la version anarchique que dans les réseaux standards. Il est donc possible de supporter les applications à contrainte de temps telles que la VoIP sur les réseaux anarchiques. Il est aussi important de noter que les choix de configuration utilisés ne sont pas en notre faveur. En effet, les liens étant configurés en $C_i - C_i = 8Mb/s$ et $R_i - C_i = 5Mb/s$, une seule pétale sature déjà largement le cœur du réseau, ce qui est particulièrement défavorable à Jungle, en particulier vue l'aspect cyclique de la topologie. On peut s'attendre à des résultats nettement meilleurs au niveau de l'efficacité, par exemple, en doublant $C_i - C_i = 16Mb/s$.

Critiques et travaux futurs

Nous avons montré à l'aide d'une topologie similaire à celle d'un FAI que les réseaux anarchiques sont dépendants des files d'attente Fair-Drop, y compris dans les routeurs de cœur de réseau. Etant donné qu'un traitement et des états pour chacun des flots n'est pas possible sur les routeurs de cœurs, une alternative doit être trouvée à Fair-Drop sur ces derniers [3]. Une solution serait d'avoir un mécanisme de minimisation du débit semblable à Tetrys mais qui fonctionnerait également avec DropTail. Puisque l'existence même d'un tel mécanisme n'est pas trivial, nous proposons que les files d'attentes à pertes équitables par interfaces d'entrées plutôt que par flot. La complexité de cette dernière est négligeable et son implé-

3. Cependant, nous savons que la puissance des routeurs de cœur peut gérer un grand nombre d'états à la volée.

mentation sur des routeurs à haut débit serait donc possible. Le trafic obtenu avec cette file d'attente n'est pas nécessairement de type *max-min fair*, et ses propriétés sont à étudier pour des topologies de tailles réalistes. Un des avantages serait par exemple le fait que le taux de perte à appliquer peut être dérivé du contrôle d'accès. L'opérateur pourrait adapter librement la répartition du trafic afin de privilégier l'efficacité plutôt que l'équité.

Nous avons montré que les performances de Tetrys surpassent celles de TCP pour des files d'attente de faible capacité mais néanmoins supérieures à 8 paquets. En dessous de cette valeur, l'implémentation de Fair-Drop tend à perdre des paquets par dépassement de buffer plutôt que de manière équitable (*e.g.* se comporte comme un file d'attente de type Drop-Tail). Il faut donc repenser l'implémentation de Fair-Drop de manière à ce que les paquets des différents flots demeurent équitablement jetés malgré des files d'attente de capacité arbitrairement petite.

De même, afin de pallier aux effets de synchronisation des flots, nous avons introduit un temps d'attente aléatoire à l'émission de chaque paquet. En considérant une échelle de temps assez courte, le débit obtenu par un flot lorsqu'il partage un goulot d'étranglement tend à être variable. Le débit préalablement estimé est donc biaisé, ce qui pénalise l'efficacité de Tetrys en particulier pour un faible RTT. De la même manière, les pertes appliquées par Fair-Drop sont uniformes. Les appliquer de façon régulière (à la façon d'un tourniquet) permettrait d'obtenir un débit beaucoup moins variable à court terme et donc de pénaliser moindrement l'estimation du débit et du nombre de paquets manquants.

Bien que l'évaluation des applications temps réel nous permette de dire qu'il est possible de supporter ces dernières dans un réseau anarchique, elle reste sommaire et n'apporte pas d'information sur les conditions rencontrées par les flots et particulièrement sur la distribution des pertes. En effet, plus les pertes sont groupées, plus il faut que Tetrys génère de la redondance afin de les reconstruire dans le délai imparti par l'application et moins les réseaux anarchiques sont efficaces. Il semblerait malgré tout qu'avec Fair-Drop ou RED, les pertes soient uniformément distribuées.

Conclusion

Tetrys a été évalué sur plusieurs topologies (étoile, cyclique, parking, papillon) en fonction de deux métriques que sont l'efficacité et l'équité [4]. Nos premières évaluations montrent que Tetrys obtient des performances égales ou meilleures que TCP en termes d'efficacité et d'équité.

Le déploiement de ce type de réseaux n'apporte pas forcément un gain substantiel pour l'utilisateur et nécessite une refonte importante des routeurs de l'Internet qui doivent implémenter un mécanisme de rejet équitable. Cependant, un déploiement incrémental est en-

4. L'efficacité de Tetrys ou de TCP est définie comme le rapport entre le débit utile aux applications et la capacité du réseau, c'est à dire le débit total lorsque ce dernier est maximisé tandis que l'équité a été calculée suivant l'index de Jain.

visageable et une cohabitation avec TCP est possible. Aussi, cette proposition pourrait être intéressante au sein de grilles de calcul ou chez les opérateurs réseaux. Un des points important de cette recherche sera donc de prouver cette possibilité.

Références sur le chapitre

[1] Pierre-Ugo Tournoux, Un protocole de fiabilité basé sur un code à effacement «on-th-fly », ISAE, 10 decembre 2010

[2] Barath Raghavan and Alex Snoeren, Decongestion Control, ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-V), 2006

[3] Thomas Bonald, Matthieu Feuillet and Alexandre Proutière, Is the "Law of the Jungle" sustainable for the Internet ?, In Proceedings of INFOCOM, 2009

[4] Luis López, Antonio Fernández, Vincent Cholvi, A game theoretic comparison of TCP and digital fountain based protocols, Elsevier Computer Networks, 2007

[5] Jean-Yves Boudec, Rate adaptation Congestion Control and Fairness : A Tutorial, 2000

[6] Sally Floyd and Senior Member and Kevin Fall, Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, 1999

[7] Lawrence G. Roberts, A radical new router, IEEE Spectrum, 2009

[8] Andrew L.H. Lachlan et Al., Towards a Common TCP Evaluation Suite, Protocols for Fast, Long Distance Networks (PFLDnet), Mar 2008

[9] L. Massoulié and J. Roberts, Bandwidth Sharing : Objectives and Algorithms, IEEE/ACM Transactions on Networking, 1999

[10] Rob Sherwood, Bobby Bhattacharjee, Ryan Braud, Misbehaving TCP receivers can cause internet-wide congestion collapse, Proceedings of the 12th ACM conference on Computer and Communications Security, 2005

[11] Ravesh Mahajan and Sally Floyd, RED with preferential dropping (RED-PD), IEEE ICNP, 2001

[12] Rong Pan and Balaji Prabhakar and Lee Breslau and Scott Shenker, Approximate Fair Allocation of Link Bandwidth, IEEE Micro, 2003

Chapitre 9

Conclusion et perspectives de recherche

Ce manuscrit a présenté un ensemble de solutions visant à améliorer le service du transport des données. Mis à part la première contribution qui concernait les réseaux à qualité de service, l'ensemble des mécanismes présentés sont proposés pour les réseaux *best-effort* de l'Internet. Nous avons également présenté quelques propositions situées dans le cœur du réseau dont l'objectif était soit l'amélioration du service *best-effort*, soit l'augmentation de la quantité d'information que pouvait retourner ce dernier aux extrémités. Bien que ces contributions soient éloignées des propositions d'extrémités, elles ont toute légitimité dans ce mémoire car il est bien évidemment difficile de réfléchir à une approche bout-en-bout en s'affranchissant du cœur du réseau.

On peut également noter que bon nombre des contributions présentées se sont attachées à résoudre des problématiques long-délai, qui sont inhérentes aux architectures satellites et relatives à mon implication dans l'équipe satellite du NICTA et mon dernier poste à l'ISAE. Cette problématique reste un problème ouvert. Même si l'Internet a levé des verrous en termes de débit, le délai reste quant à lui une donnée incompressible. Enfin, l'Internet actuel est de plus en plus utilisé comme un réseau qui fournirait des garanties temps-réel et les applications correspondantes à ce critère sont légions sur la toile. C'est aussi une des raisons qui ont motivé les études présentées sur le tranport des données multimedia.

Depuis mon arrivée sur Toulouse, ma recherche est fortement orientée vers les communications aérospatiales et donc : les DSN (Deep Space Networks) ; les DTN (Delay Tolerant Networks) ; et les communications long-délai (e.g. satellite). Je pense que les recherches qui en découleront (comme cela a été le cas pour le protocole Tetrys qui était destiné à fournir une couche de fiabilité à un protocole de transport DTN), sont toutefois applicables à l'Internet *best-effort* qui dans un ordre de grandeur différent, pose aussi des problèmes de délivrance des données dans un temps borné aux applications multimedia. De plus, le mariage entre théorie des codes, network coding et transport semble apporter des solutions de fiabilité

intéressantes qui permettent, entre autres, de mitiger l'impact du temporisateur de retransmission.

En conclusion de ce mémoire, nous présentons ci-dessous les deux perpectives principales des travaux qui ont été présentés :

Poursuite des travaux sur le protocole Tetrys

Nous sommes encore loin d'avoir exploré les possibilités que nous offre le protocole Tetrys et les perspectives de recherche qui en découlent toucheront aussi bien la diffusion multicast fiable que la fiabilisation des protocoles multi-chemins (SCTP, MTCP). Cette classe de protocole offre également une perspective de recherche intéressante qui permettrait l'établissement d'une voie redondante ou l'agrégation d'interfaces afin d'augmenter le débit utile de l'application. A ce jour, nous réfléchissons avec Golam Sarwar et Pierre-Ugo Tournoux au problème de la minimisation des retransmissions abusives au sein des protocoles multi-chemins par l'adjonction d'un codage particulier visant à ce que chaque retransmission (même abusive), soit toujours utile pour la communication en cours. Une autre facette du travail de Golam concerne la gestion de multiples mécanismes de contrôle de congestion et de fiabilité au sein d'un protocole de transport multi-chemins. De son côté, Guillaume Smith, doctorant en co-tutelle avec l'Australie, explore de façon plus générale les capacités de ces codes à fenêtres élastiques et réfléchit à un système de calcul dynamique de la redondance en fonction des retours du réseau. L'objectif étant de déterminer comment calculer dynamiquement le meilleur taux de redondance afin d'approcher la performance d'un code parfait. Enfin, les derniers travaux sur lesquels nous nous penchons concernent le déploiement de réseaux anarchiques qui font l'objet du chapitre 8.

Routage à contrôle de congestion au sein des topologies satellites et détermination de capacité des réseaux DTN

Enfin dans un schéma encore plus prospectif, ma derniere orientation de recherche concerne le routage à contrôle de congestion au sein des topologies satellites et la détermination de capacité des réseaux DTN. Celui-ci reste dans la même lignée que les travaux entrepris jusqu'alors et mélange à la fois des problématiques de routage, de dimensionnement de réseaux DTN et de recherche de *max-flow min-cut* au sein de constellation de satellites. La topologie d'une constellation de satellites joue un rôle prépondérant sur les performances du système. Certaines constellations présentent notamment des liens inter-satellites à connexion intermittentes qui complexifie les tâches de routage et de dissémination de l'information. Ainsi, il est nécessaire de mettre en place au-dessus de ces topologies une signalisation précise, parfois gourmande en ressources, voir de pré-calculer toutes les tables de routage d'une constellation, ce qui pose des problèmes en termes de tolérance aux erreurs. Après un cer-

tain désintérêt, le monde industriel semble se tourner de nouveau vers les constellations. De plus, l'intérêt de l'utilisation d'Internet par satellite s'accroît progressivement [1]. Aussi, le développement de mécanismes de routage permettant la dissémination efficace d'information prenant en compte la variabilité du trafic et la capacité en termes de mémorisation et d'écoulement du trafic est un problème qu'il semble important d'étudier. Cette problématique fait l'objet de la thèse de Rémi Diana co-encadrée avec Telecom Bretagne, le CNES et Thales Alenia Space.

1. Voir : Google goes after 3 billion with super satellite : `http://www.theregister.co.uk/2008/09/09/other_three_billion/`