Habilitation à Diriger des Recherches
de l'Université de la Réunion

Spécialité

RESEAUX INFORMATIQUES

présentée par

M. Pascal ANELLI

Des aléas de la communication :

de la transmission au transport

Présentée et soutenue publiquement le 11 Mai 2012
devant le jury composé de :

Pr. Raouf BOUTABA	**Rapporteur**
Pr. Michel DIAZ	**Rapporteur**
Pr. Olivier FESTOR	**Rapporteur**
Pr. Pascal LORENZ	**Rapporteur**
Pr. Jean DIATTA	**Examinateur**
Pr. Jean-Daniel LAN-SUN-LUK	**Examinateur**

Version revue et corrigée du 4 Avril 2012

Résumé

Une communication au sein d'un réseau peut subir des aléas pouvant se produire à différents niveaux de l'architecture de protocoles. Ce document présente l'évaluation des différentes techniques mises en oeuvre pour protéger les communications. Au niveau physique, nous présentons l'évaluation de performance du système de protection des réseaux optiques. Ce système vise à prendre en compte l'aléa de panne ou de rupture. Dans le cas des systèmes de transmission à diffusion comme les réseaux ad hoc sans fil, un nouvel aléa apparait : celui de l'accès à la transmission. Dans ce domaine, nous proposons un système de contrôle de trafic afin de rendre un service de transfert prévisible efficace. Au niveau du réseau, le trafic introduit le principal aléa appelé couramment congestion. Nous proposons une architecture au niveau des routeurs afin de différencier les services. Dans le cas du trafic issu du protocole de transport TCP, une action au niveau des routeurs reste insuffisante pour qu'il puisse bénéficier d'un service avec un débit prévisible. Nous proposons un conditionnement du flux TCP selon le niveau de congestion mais également une méthode pour déterminer le niveau de congestion dont peut souffrir une connexion TCP. Dans le contexte de l'Internet classique, la congestion n'a pas les mêmes conséquences sur les flots TCP. Une action au niveau des routeurs peut changer la performance des flots courts.

Mots clés :

Communication informatique, survivance, qualité de service, service assuré, détection de congestion, flots courts TCP, contrôle de trafic.

Remerciements

Beaucoup des travaux présentés dans ce mémoire ont été menés avec d'autres collègues. Par crainte d'oublier certains, je n'en citerai aucun. Cependant, je les remercie tous pour avoir partagé leurs expériences et leurs connaissances. Ce sont aussi ces échanges qui font de la recherche une activité intéressante.

Chapitre 1

Introduction

Ce mémoire présente les principaux travaux de recherche que j'ai pu mener en tant qu'enseignant-chercheur au LIP6 (Laboratoire d'Informatique de Paris VI) puis, depuis 2002, au sein du LIM (Laboratoire d'Informatique et de Mathématiques). Il vise à expliquer la démarche de mon activité scientifique en essayant de la mettre en perspective. Les aspects les plus techniques sont décrits dans les articles publiés. Ces articles sont disponibles sur ma page web des publications au format pdf :
`http://personnel.univ-reunion.fr/panelli/3_publications`

Problématique

L'histoire des réseaux nous apprend que le réseau informatique est un fournisseur de service aux applications. La particularité de ce service est qu'il ne doit avoir aucune sémantique applicative. Ce principe est maintenant largement admis et connu sous le terme de "bout en bout" en référence aux implications architecturales qu'il a entraînées. Le service offert par le réseau est un simple service de communication. Autrement dit, c'est un service de transport de blocs de données d'une machine émettrice à une machine destinatrice. Il n'en reste pas moins que le service de communication doit offrir des caractéristiques qui le rendent utilisable par les applications. Tout d'abord, c'est le maintien de la connectivité puis, ensuite, l'accessibilité de la destination. La connectivité dépend de l'infrastructure et du routage. L'accessibilité est un principe qui a été plus long à émerger au niveau de l'Internet. Cette dernière dépend du trafic et est traitée par la thématique du contrôle de trafic. La métaphore du réseau routier est particulièrement bien adaptée pour comprendre des notions existantes également dans un réseau informatique. Si je veux aller de Saint Denis (nord de la Réunion) à Saint Pierre (sud de la Réunion) en voiture, il me faut une infrastructure (les routes) et une signalisation pour m'indiquer le chemin. Mais est-ce suffisant pour que j'atteigne ma destination ? Si je suis seul sur l'ile, il n'y a pas de doute. Mais en réalité, je vais devoir composer avec le trafic qui va me faire varier mon temps de trajet et, au pire, peut me décourager d'y aller. En somme, l'accessibilité est la propriété qui définit l'accès effectif à une destination. Elle revêt un caractère de performance et se mesure de manière générale par le délai.

L'émergence de nouvelles technologies tant au niveau de l'infrastructure que des terminaux a complexifié la connectivité et l'accessibilité. En effet, les types d'applications utilisant le service de communication de l'Internet n'ont cessé d'augmenter. La tendance est au renforcement de la contrainte de délai sur le service de communication. Cela a commencé avec le Web au début des années 1990. La contrainte s'est exprimée alors en terme d'interactivité. Dans les années 2000 avec les applications à média continu dans le temps comme la voix ou la vidéo, le retard est devenu rédhibitoire. La tension sur le service a été énorme, et de nombreuses pistes ont été ouvertes pour répondre à ces

évolutions. Citons par exemple :

- Au niveau du transport, la définition de nouveaux protocoles adaptés aux médias et aux usages des applications.
- Au niveau du réseau, la gestion des ressources et des trafics.

Et tout cela a été effectué dans un contexte de croissance effrénée de la taille du réseau, de l'hétérogénéité grandissante de l'infrastructure.

Thématique d'études

Ma thématique de recherche s'inscrit dans la composante d'accessibilité du service de communication. C'est à dire sur les aléas pouvant perturber la qualité du service de communication. Un aléa se définit d'après le dictionnaire comme "une chance, un hasard favorable ou non". Dans notre contexte, il est interprété comme un hasard défavorable. Pour la petite histoire, il est amusant de regarder le terme anglo-saxon pour indiquer un hasard défavorable. On le traduit par *hazardous* qui donne un sens de incertain, hasardeux voir dangereux. Les aléas traités dans ce mémoire se répartissent sur les couches de la communication (couche 1 à 4 de modèle d'architecture normalisée). Comme le montre la figure 1.1, les contributions présentées par la suite se répartissent sur 3 aléas :

- L'évaluation de performance de la protection mise en oeuvre au niveau des liens optiques. L'objectif de la protection est en quelque sorte de masquer les pannes au composant de routage. C'est ici l'aléa de la rupture de lien ou de la panne qui est traité.
- Le contrôle de l'accès pour les réseaux (ou supports) à diffusion. Le contrôle de l'accès au support est distribué. Lorsque les accès sont partagés équitablement entre les noeuds, comment alors fournir des services prévisibles en qualité dans un tel contexte ? On se situe dans l'aléa d'accès.
- Les services de communication à la couche de transport. Dans ce thème, la question porte sur comment gérer la congestion ou faire que le contrôle de congestion ne soit pas un frein à la performance du service de communication. Ce thème traite de l'aléa du trafic dans les réseaux filaires.

La figure 1.1 montre également la progression thématique dans le temps. A noter, les travaux de développement d'une souche IPv6 effectués dans le cadre du projet DRET sortent du cadre de ce mémoire.

Ma contribution à l'évaluation de performance de la protection fait suite à une demande d'un industriel (Alcatel-CIT) pour développer des commutateurs pour des liens haut débit. La question portait sur les temps de traitement autorisés dans les noeuds pour respecter la contrainte d'une correction de la panne en 50ms. Le second chapitre de ce mémoire traite du domaine de la survivance de réseau.

En partant du déploiement de l'architecture Diffserv sur les réseaux à diffusion de type Ethernet, nous avons développé un système de contrôle d'accès efficace pour réseau ad hoc complet. Le chapitre 3 présente ma contribution pour fournir des services avec une qualité prévisible pour ces réseaux.

En ce qui concerne les services de communication à la couche de transport, mon travail porte sur le plan utilisateur. Plus précisément, cela a consisté à déterminer des traitements applicables aux paquets et aux flots dans le but d'améliorer le niveau de service. Les problématiques traitées sont la fourniture d'un débit minimum pour TCP dans l'architecture Diffserv et l'amélioration de l'interactivité des flots courts dans l'Internet classique. Ces contributions sont décrites dans le chapitre 4 de ce mémoire.

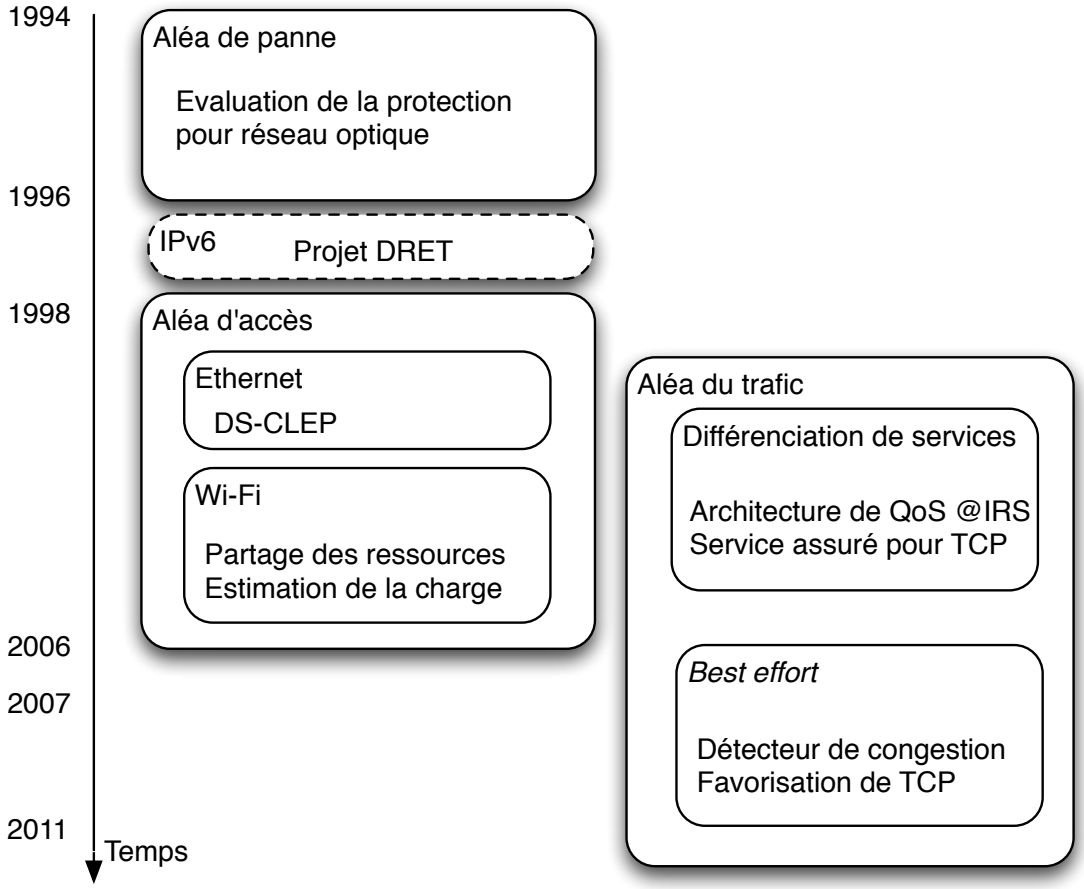


FIGURE 1.1 – Vue synthétique des contributions réparties dans le temps.

Pour chacun des 3 thèmes abordés, je présente les problèmes traités et j'essaie de dégager les grandes lignes de ma contribution. Je termine par un chapitre de synthèse, qui a pour but de proposer une vision et des perspectives de recherche.

Chapitre 2

Evaluation de la protection pour réseau optique

2.1 Présentation du domaine d'étude

Ma première contribution porte sur le thème de la survivance (ou survivabilité) de réseau. La survivance de réseau est aussi appelée protection et rétablissement de réseau. L'objectif est de rendre un réseau robuste à la panne. Cela signifie que le service de communication est maintenu quand bien même des artères ou des noeuds de commutation tombent en panne dans le réseau. La survivance de réseau est devenue un sujet critique au début des années 1990 avec l'usage grandissant de la fibre optique. Les capacités offertes par ces supports sont tels qu'en cas de panne, une bonne partie des utilisateurs se trouve bloqués.

Par exemple dans [1], l'auteur explique que le 8 Mai 1988 à 16 heures au central téléphonique de Springfield (Illinois), une rupture d'alimentation électrique est signalée et qu'un incendie probable toucherait le central de Hinsdale. L'opérateur essaya en vain d'appeler les pompiers de Hinsdale, le feu avait déjà brulé les lignes. Les conséquences du feu impactaient déjà la communauté :

- à l'hôpital tout proche, le personnel n'arrivait plus à se joindre d'un étage à l'autre,
- les commerçants ne pouvaient plus vérifier les cartes de crédit,
- les contrôleurs aériens de l'aéroport international de O'Hare n'étaient plus en mesure d'assurer le trafic aérien,
- les appels du service d'urgence (911) restaient sans réponse.

Le résultat de l'incident causa la perte des services de communication, l'isolement de 35000 abonnés, la destruction de 37000 circuits interurbains, 13500 circuits spéciaux, 118000 circuits longue distance et la panne de 50% des téléphones cellulaires de Chicago. En tout ce fut 500000 abonnés qui passent $3,5$ millions d'appels téléphoniques par jour qui furent touchés par le feu. Le service complet fut rétabli le 5 Juin 1988.

Ce besoin de survivance de réseau a été amplifié ces dernières années avec la convergence des services sur l'infrastructure IP. Bien que le routage IP offre une forme de survivance en reconfigurant les tables de routage, les délais ne sont pas du même ordre de grandeur. Les mécanismes de protection ont des contraintes de rétablissement du service exprimées en ms. Du fait de la structuration en couche des réseaux, la survivance de réseau peut être mise en oeuvre de différentes façons. Le choix résulte d'un compromis entre vitesse de rétablissement et coût des capacités de redondance. Les auteurs dans [2] définissent des critères de classification qui sont :

- le niveau sur lequel opère le rétablissement,

– la méthode de constitution du chemin de réserve,
– l'usage fait des ressources de réserve,
– la portée du rétablissement, à savoir local ou de bout en bout,
– le nombre de domaines que la procédure de rétablissement peut impliquer.

On parle de protection lorsque le chemin de données et son chemin de secours sont établis en même temps. La rétablissement ou restauration est une approche réactive : le chemin de secours est établi après que la panne se soit produite. Dans [2], les auteurs indiquent que les méthodes de recouvrement de pannes ne résolvent pas uniquement un problème de connectivité mais que c'est aussi un facteur de performance entrant dans l'expression de la qualité de service.

La protection la plus simple et la plus ancienne (1986) pour assurer de la survivance au réseau repose sur le système APS (*Automatic Protection Switching*). Ce système est conçu pour les réseaux à commutation de circuits de type SDH/SONET. L'action de protection consiste à provisionner des canaux de secours qui serviront, en cas de panne, à basculer le trafic des canaux primaires. Ces canaux de secours ont la même capacité que les canaux primaires. La topologie de protection est en anneau composé par des liens de deux ou quatre fibres. L'anneau présente la propriété d'être un réseau connexe avec un nombre de liens égal au nombre de noeuds.

2.2 Problématique abordée

Le fonctionnement de l'APS est défini par les standards [3, 4]. Afin que la panne ne soit pas détectable par les couches supérieures et soit perçue comme une erreur de transmission, il faut que la reprise du service s'effectue dans un délai inférieur à 50ms [5]. Le système APS traite deux types de panne :

– la dégradation du signal (SD : *signal degrade*),
– la perte de signal (SF : *signal fail*).

Le recouvrement de la panne s'effectue localement entre deux noeuds SONET par une action combinée du noeud amont et du noeud aval à la panne. Les actions de protection à enclencher au niveau de ces noeuds dépendent de l'impact de la panne. Si le canal de secours est touché, il faut basculer le trafic sur le canal de secours contre-rotatif. Cette situation conduit à emprunter tous les liens de secours pour rejoindre le noeud aval. L'initiateur de cette action de protection est le noeud aval dans la mesure où il détecte la condition de panne. Il envoie une demande de commutation d'anneau (*Ring switching*) afin de recevoir par le noeud amont le trafic du canal primaire sur le canal de secours. Si seul le canal primaire est touché, un simple basculement du trafic sur le canal de secours est suffisant. Cette action de protection se nomme commutation de canal (*Span switching*). Le noeud amont et le noeud aval signalent les actions de protection au moyen de deux octets dans l'en-tête de la trame SDH/SONET.

Dans le cadre du contrat industriel avec ALCATEL-CIT, nous avons évalué la performance du protocole du système APS. Le cahier des charges portait sur la détermination d'un temps de traitement maximum par les noeuds chargés de traiter la panne. Le temps de traitement maximum est contraint par la borne maximum du délai de reprise de service fixée à $50ms$. Ce temps de traitement va servir à l'industriel pour faire les choix technologiques pour la mise en oeuvre du protocole dans les commutateurs SONET/SDH. Les objectifs de cette étude étaient :

– de modéliser le protocole du système APS au moyen d'un modèle décrit dans le simulateur OPNET,

T_{proc}	Temps de traitement d'un noeud d'extrémité (amont et aval)
T_s	Temps de traitement d'un noeud intermédiaire
T_t	Temps de transit dans un noeud intermédiaire ($125\mu s$)
T_p	Temps de propagation du signal sur un lien ($380\mu s$)

TABLE 2.1 – Paramètres temporels du système APS.

- de déterminer le cas de panne le plus contraignant et le temps de traitement maximum admissible,
- de vérifier le temps de recouvrement quand deux pannes se produisent quasi simultanément. Les pannes sont localisées dans divers endroits de l'anneau. Par exemple, ce cas de panne peut se produire dans la situation d'un tremblement de terre où des ruptures d'alimentation électrique ou des ruptures de liens peuvent survenir en plusieurs endroits.

Les paramètres temporels retenus dans le modèle sont indiqués dans le tableau 2.1. La difficulté de cette étude ne provient pas du nombre de paramètres mais de l'analyse du protocole et du parallélisme intrinsèque au système APS. C'est la raison qui nous a motivé à utiliser un outil de simulation.

2.3 Contribution

Notre contribution a porté sur la détermination des temps maximum dans les noeuds. La figure 2.1 montre l'évolution du temps maximum admissible de chaque noeud encadrant la panne, lorsqu'il n'y a qu'une seule panne qui se produit sur un réseau sain. On voit clairement que l'existence d'un cas de panne défavorable n'a de signification uniquement dans un contexte d'anneau donné (i.e. nombre de noeuds). Dans le cas le plus défavorable, la valeur maximum admissible est de $16ms$ et concerne la commande de commutation d'anneau pour dégradation du signal (SD_R). Les autres commandes sont SF_R et SF_S pour respectivement commutation d'anneau et commutation de canal suite à une perte de signal.

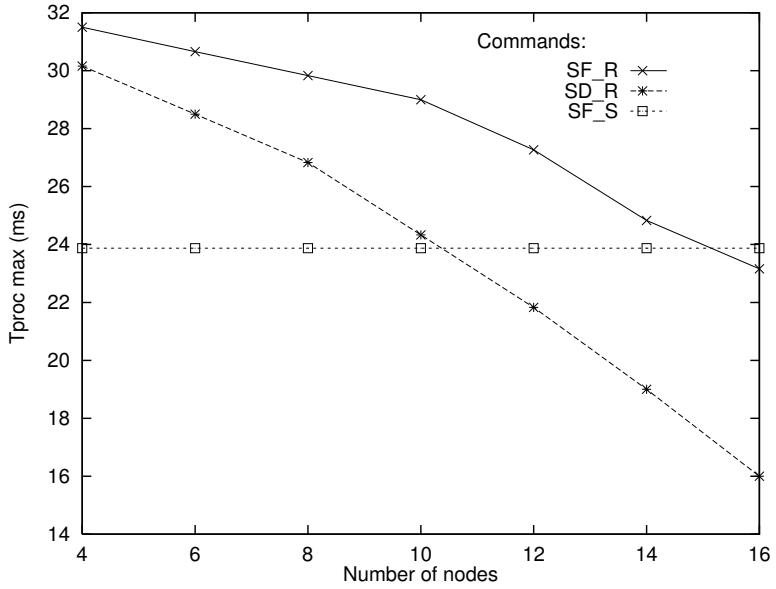


FIGURE 2.1 – Temps de traitement maximum admissible par le noeud amont et aval en fonction de la taille de l'anneau.

Le temps de traitement des noeuds d'extrémités se compose de deux termes : (1) temps de traitement de la signalisation ; (2) temps d'action de protection. Nous montrons dans l'étude que le temps de traitement de la signalisation est critique. Enfin, si le temps de traitement dans les noeuds intermédiaires est nul, le temps de traitement dans les noeuds d'extrémités peut être augmenté de près de $4ms$ en respectant toujours la contrainte de $50ms$.

Lorsque deux pannes sont entrelacées, nous avons mis en évidence que :
- le temps de recouvrement est dans tous les cas inférieur à $100ms$,
- le temps de recouvrement ne suit pas une courbe linéaire lorsque les deux pannes nécessitent une commutation d'anneau pour cause de défaut de signal (SF_R). Cela est dû au parallélisme de traitement déclenché par la deuxième panne.

Notre découverte, la plus surprenante, dans ce genre de panne, porte sur la valeur de traitement dans les noeuds intermédiaires. Dans le cas de la simple panne, il vaut mieux avoir un temps de traitement T_s proche de $0ms$ pour avoir plus de temps dans les noeuds d'extrémités. Avec deux pannes, cette configuration conduit à des temps de recouvrement plus importants. La figure 2.2 montre dans le cas de deux pannes entrelacées l'évolution du temps de recouvrement de la panne. On remarque que le temps de recouvrement le plus important est pour le noeud ayant $T_s = 0$. Dans cette configuration de panne, on peut considérer que le temps de traitement dans les noeuds d'extrémités est l'élément prédominant par rapport au temps pris pour les échanges. Comme avec $T_s = 0$, la valeur du temps de traitement dans les noeuds d'extrémités est plus importante (un maximum de $20ms$), le cumul de ce temps de traitement rend la procédure de recouvrement moins performante. Sachant que T_s n'a pratiquement pas d'influence pour les anneaux comportant peu de noeuds, l'effort de mise en oeuvre ne doit pas porter sur l'optimisation du temps de traitement dans les noeuds intermédiaires.

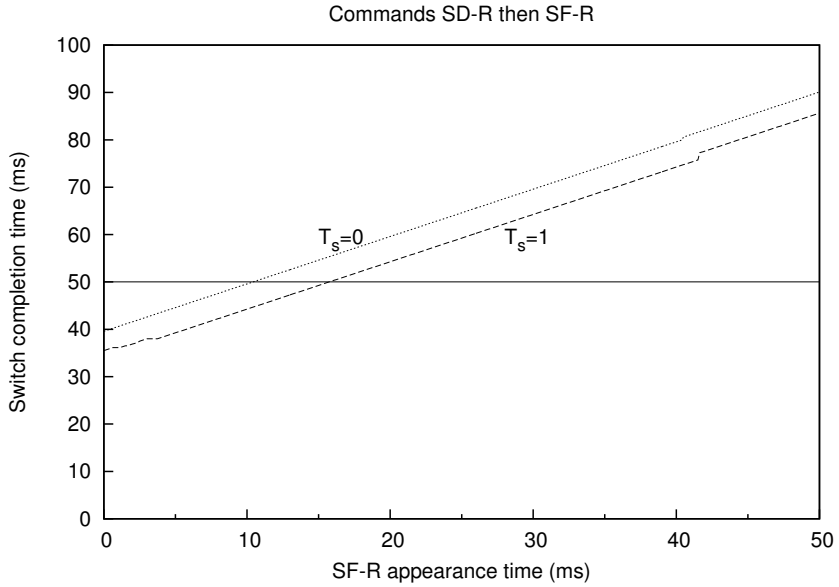


FIGURE 2.2 – 2 pannes entrelacées avec des temps de traitements des noeuds d'extrémités contraints par T_s.

L'étude complète sur l'évaluation de performance du protocole du système APS a été publiée dans *IEEE transactions on Communications* [6].

2.4 Synthèse

Les travaux réalisés ont identifié le cas de panne le plus défavorable ainsi que la limite de temps de traitement admissible pour les noeuds en charge des actions de protection. Nous avons montré que le temps critique porte sur le temps de traitement de la signalisation dans les noeuds d'extrémités. La contrainte sur le temps de traitement dans les noeuds intermédiaires est faible. L'étude de la double panne confirme que la contrainte porte essentiellement sur les temps de traitements dans le noeud amont et le noeud aval. Par conséquent, une mise en oeuvre efficace du protocole APS dans les commutateurs SONET/SDH consiste à développer un contrôleur qui effectue les actions dans les noeuds d'extrémités en $16ms$ maximum avec un temps de traitement de la signalisation ne dépassant pas $8ms$. Les traitements des noeuds intermédiaires ont comme limite maximum $1ms$.

2.5 Conclusion

La protection en un délai de $50ms$ de SONET/SDH est devenue un standard pour les autres techniques visant à rendre un réseau robuste [7]. En effet, la survivance a répondu en premier à une demande des réseaux en mode circuit. C'est maintenant une propriété qui a été intégrée dans les réseaux IP. La demande de survivance des réseaux IP s'est accrue en même temps qu'ils sont devenus une infrastructure critique. Le RFC 6414 issu du groupe de travail Benchmarking Methodology (BMWG) de l'IETF illustre bien cette demande de protection de la couche IP [8]. Ce groupe a formalisé les termes et les mesures de la performance de la protection faite au niveau IP. Ces dernières années ont vu l'émergence de techniques de survivance dans différentes couches : MAC, MPLS, SONET/SDH et optique [9]. Pour la couche MAC, le groupe de travail IEEE 802.17 appelé *Resilient Packet Ring* (RPR) propose une méthode d'accès sur un anneau intégrant une protection à la panne [10]. Ces travaux s'inspirent de la protection mise en oeuvre dans SONET/SDH mais la transmission est optimisée pour le transport des trames Ethernet. Le délai de reprise suite à une action de protection dans un RPR reprend la valeur limite de 50ms [11].
Dans le cadre d'un anneau SONET/SDH, notre contribution a porté sur la répartition des délais de traitement entre les noeuds afin de respecter le délai de 50ms dans le cas d'une panne et de 100ms dans le cas de deux pannes. Une configuration qui respecte la contrainte de 50ms n'est pas forcément la meilleure dans le cas de deux pannes. C'est l'une des conclusions que l'on peut retenir de notre contribution. Dans le cas du réseau RPR, cette question relative à la double panne mérite d'être posée.

Enfin pour clore ce chapitre sur l'aléa de panne, il est intéressant de noter que la survivance de réseau rentre dans le domaine de la résistance [12]. Dans ce cas, la résistance est vue sous l'angle de la propriété de robustesse du réseau. Un autre angle de la résistance est la tolérance aux perturbations. Les perturbations s'expriment par rapport à la connectivité. Par exemple, le système de communication est perturbé de telle manière qu'il lui est difficile de maintenir une connexion stable entre deux utilisateurs voire même qu'il n'existe plus de chemins stables entre eux. Cette caractéristique a conduit à la notion *disruption-tolerant networking* (DTN) et de communication opportuniste que nous reverrons dans le chapitre 5.

Chapitre 3

Contrôle de l'accès pour les réseaux à diffusion

3.1 Présentation du domaine d'étude

Dans un contexte d'un Internet à QoS, un service à qualité prévisible doit être rendu quelle que soit la technologie de transmission sous-jacente. Dans le cas des liens point à point, l'accès au support est géré dans une seule interface réseau. Le provisionnement des services peut se faire en contrôlant l'accès à l'interface réseau. Dans le cas des liens multipoints comme Ethernet, l'accès au support est partagé entre les noeuds ayant une interface réseau connectée au lien. Le provisionnement des services devient problématique du fait de l'accès distribué au support. Cette distribution du contrôle introduit un aléa à l'accès.

Le brevet CLEP (*Controlled Load Ethernet Protocol*) d'Eric Horlait [13] décrit un procédé pour la gestion de la bande passante. Ce procédé fournit un support à l'architecture Intserv pour Ethernet. L'architecture Diffserv a amené une gestion des ressources par classe de trafic et non plus par flot applicatif. A partir du brevet CLEP, j'ai proposé une gestion des ressources adaptée à l'architecture Diffserv. Mon objectif a été double :

- fournir un support pour un service prévisible en débit,
- maximiser le taux d'utilisation des ressources.

Le principe consiste à offrir une différenciation de services par le contrôle du trafic normal (*Best effort*) indépendamment du ou des services de communications sous-jacents. Le contrôle appliqué au niveau IP repose sur l'état du réseau. Ce travail a été étendu avec Fanilo Harivelo au domaine des réseaux ad-hoc complets et a donné lieu à une collaboration avec Bernd Wolfinger. Nous avons étudié à augmenter l'efficacité de l'usage du support par une diminution des messages de signalisation.

3.2 Réseau filaire

Le contrôle de la qualité de service est un problème pour les réseaux Ethernet en mode bus. La méthode d'accès sur ce média partagé repose sur un contrôle d'accès distribué et égalitaire. La raison du problème s'explique par le fait que les architectures de QoS sont postérieures à la méthode d'accès d'Ethernet. Celle-ci ne tolère pas la prioritisation des flots. Cette faiblesse a été corrigée pour les réseaux Ethernet en mode commuté par la spécification IEEE 802.1Q. Le format de trames comporte des bits de priorité [14]. Cependant, le problème reste entier pour le mode en bus. Dans ces conditions les problèmes qui se posent sont :

– Le support est multipoint et la multiplicité des accès au support empêche l'ordonnancement de l'ensemble des paquets émis comme dans le cas d'un accès unique.
– La bande passante du support est instable, elle peut dépendre de la charge offerte. Après avoir atteint un maximum, les performances se dégradent d'une façon presque inversement proportionnelle à la charge dans les conditions de pointe.
– Tous les émetteurs partagent la même ressource. Comme la méthode d'accès est répartie et qu'il n'y a aucun contrôle de trafic, aucune isolation des flots n'existe pour protéger les flots à QoS des flots sans QoS. Il n'est pas possible de garantir la moindre QoS à des flots spécifiques.

3.2.1 Evaluation de CLEP

Les premiers travaux ont été initiés par Eric Horlait. Le principe consistait à empêcher que le support soit saturé. Pour cela les trafics étaient contrôlés au niveau de chaque noeud de telle façon que le taux de transfert ne dépasse pas une valeur maximum. Cette proposition a offert une solution pour stabiliser la bande passante tout en garantissant un débit pour des flots à QoS. Les flots *Best effort* se répartissent la bande passante non réservée. Le contexte d'utilisation était l'architecture Intserv. L'approche de CLEP est différente de la proposition de standard SBM (*Subnetwork Bandwidth Manager*) [15]. Dans ce dernier, le contrôle porte sur les flots à QoS et les flots *Best effort* ont leur débit contrôlé par le contrôle de congestion TCP. Cette méthode est évaluée pas très efficace pour garantir la QoS sur un réseau local [16]. En effet, les flots *Best effort* introduisent des interférences sur le service à QoS.

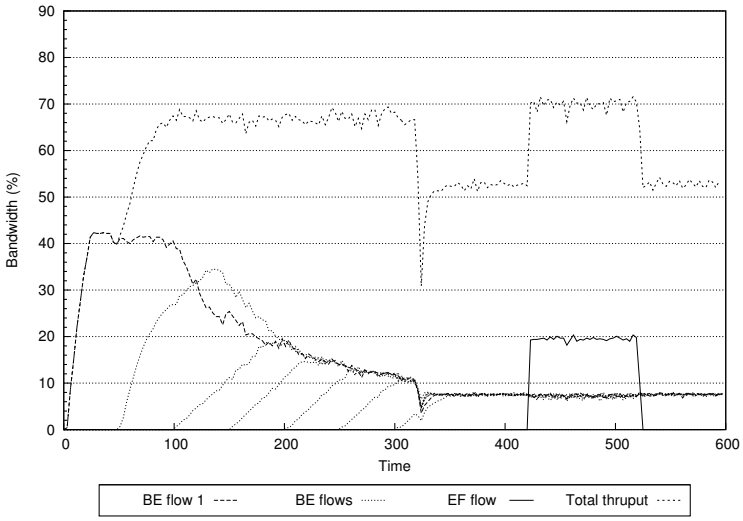


FIGURE 3.1 – Réservation avec la proposition CLEP.

La figure 3.1 a été réalisée par un modèle développé pour le simulateur ns-2. Elle présente le débit écoulé pour un réseau sous CLEP lorsqu'une station pose une réservation à l'instant $t = 320s$ et la chute du débit total écoulé qui s'ensuit car le flot à QoS (noté EF) ne démarre que $100s$ plus tard. Cette figure illustre un des défauts de la gestion de la réservation par flot de CLEP. Les ressources sont réservées à un flot et non partagées. Si le flot à QoS est absent après sa réservation ou qu'il n'utilise pas la totalité des ressources réservées, le taux d'utilisation des ressources chute. Dans le contexte de l'architecture Diffserv, la réservation des ressources se pose par classe de service et non plus par flot. Cette réservation est assimilable à un provisionnement attribué aux noeuds du réseau. C'est à dire que chaque noeud a reçu (sous une forme quelconque) un provisionnement

pour son trafic à QoS. Ce trafic peut être constitué de un ou plusieurs flots. Dans tous les cas, il est peu probable qu'un noeud utilise en permanence la totalité de la provision qu'il a sur le service à QoS. Par conséquent, les ressources provisionnées à la classe de service à QoS mais non utilisées peuvent être partagées pour le trafic *Best effort*. Ce partage est important dans la mesure où il augmente le taux d'utilisation des ressources et donc le débit écoulé. Cette modification a été étudiée dans la proposition DS-CLEP décrite dans le paragraphe suivant.

3.2.2 Proposition DS-CLEP

Le principe de stabilisation de la bande passante est repris de CLEP mais la gestion des ressources et des trafics repose sur un nouvel algorithme. En *Best effort*, l'algorithme calcule le partage des ressources selon une équité Max-min en tenant compte des ressources non utilisées par le service à QoS. Le partage selon l'équité Max-min est effectué lorsque le réseau est saturé c'est à dire lorsque la demande excède ce que peut écouler le réseau. Quelle que soit la répartition des flots entre les noeuds et les services, le réseau écoule à un débit quasi constant tout en préservant la classe à QoS. La figure 3.2 montre bien les avantages de la nouvelle proposition dite DS-CLEP. Le scénario est identique à la figure 3.1.
En premier lieu, la vitesse de convergence pour les flots *Best effort* a été augmentée. Ceci a été obtenu par la prise en compte du débit de la demande afin de pouvoir calculer un partage équitable et donc le débit alloué à chaque noeud. Comme chaque noeud diffuse sa demande en débit moyen pour chaque classe de service, un noeud peut alors calculer son partage équitable et donc le débit du trafic *Best effort* qu'il peut émettre sans risque de saturer le support.
En second lieu, le taux d'utilisation du support reste important, quelle que soit l'activité du flot à QoS. La ressource non utilisée par le flot à QoS est récupérée localement pour le flot *Best effort*.

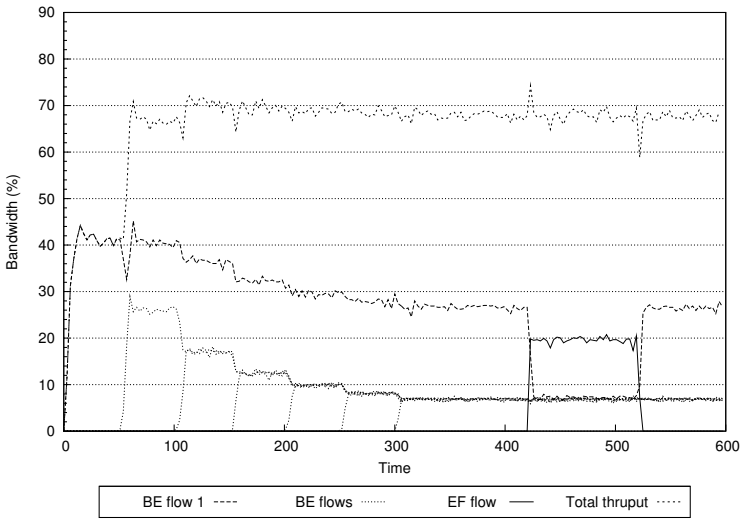


FIGURE 3.2 – Réservation avec la proposition DS-CLEP.

La proposition a montré qu'un contrôle de la bande passante sur un médium partagé pouvait être effectué de manière distribuée en utilisant le principe du partage équitable Max-min. De plus, il est également possible d'attribuer des garanties de débit sans entraîner nécessairement une baisse du taux d'utilisation de la capacité de transmission. Les

évaluations de cette proposition ont été publiées dans 2 conférences [17, 18]. Les extensions et les études complémentaires ont été faites pour les réseaux sans fil. Le paragraphe suivant présente ces travaux.

3.3 Réseau sans fil : Wi-Fi

Comme indiqué dans [19], tant que le support n'est pas saturé, une différenciation de service reste possible. La limitation du trafic *Best effort* offert par chaque nœud permet de libérer des ressources pour les trafics qui posent des contraintes de temps et de débit. DSCLEP effectue justement une régulation des flots de paquets avant leur accès au support. Son application sur un réseau ad hoc complet a été étudié. On qualifie un réseau ad hoc de complet lorsque les stations se trouvent à portée de transmission, les unes des autres.

Les réseaux Wi-Fi sont dotés d'une méthode d'accès qui offre huit catégories de trafic en jouant sur la priorité d'accès au support [20]. La régulation à l'accès est complémentaire de la prioritisation à l'accès [21]. La prioritisation vise à choisir le noeud qui doit accéder au support. La régulation au dessus de la couche de transmission détermine le paquet à transmettre à l'intérieur du noeud.

Afin de prendre en compte la contrainte de bande passante plus faible que pour le support filaire, nous avons cherché à maximiser le taux d'utilisation des ressources tout en garantissant les services à QoS. Les allocations de bande passante au service à QoS s'effectuent à la demande. Pour ce faire, l'état du réseau en ressources consommées est estimé. Deux mises en œuvre ont été étudiées :

- dans la première, une table de l'utilisation des ressources reflète l'état du réseau. La limitation du trafic normal repose alors sur cette table. La maintenance de la table s'effectue par des échanges de messages de signalisation ;
- dans la seconde, l'état du réseau se déduit d'observations faites localement. La signalisation est donc supprimée.

Dans les deux mises en œuvre, la reprise des ressources non utilisées par le service à QoS s'effectue entre tous les noeuds et non plus localement comme dans l'étude du cas filaire. Le partage équitable en est renforcé.

3.3.1 Table des ressources

L'objectif d'un partage équitable global au niveau du service *Best effort* peut avoir des conséquences en terme d'intensité de signalisation. En effet, les noeuds vont être amenés à générer beaucoup plus de messages pour assurer le transfert et la récupération des ressources.

Les auteurs de [22] étudient la performance induite sur les délais lorsque l'allocation des ressources au service à QoS s'effectue par seuil. Un tel système vise à tolérer les faibles variations du flot à QoS et donc à éviter une génération importante de messages de signalisation. Ce système à base de seuil est repris et évalué dans[23, 24]. Avec un système d'allocation à quatre états comme représenté par la figure 3.3, les messages de signalisation sont émis par un noeud aux changements d'état. On note $\hat{\rho}$ l'estimation du débit d'émission dans le service à QoS pour un noeud donné. Les valeurs des quatre états représente le pourcentage de bande passante provisionnée à un noeud pour son trafic à QoS.

La table 3.1 donne les résultats de l'évaluation comparative de la proposition DSCLEP et du système à base de seuil. Le scénario consiste en cinq noeuds qui émettent

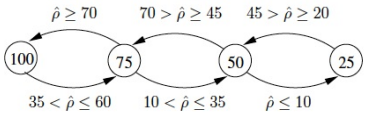


FIGURE 3.3 – Allocation selon 4 états.

vers un sixième noeud. Les quatre premiers émettent un trafic *Best effort* correspondant à 40% de la bande passante du support. A l'instant $t = 628s$, un noeud quitte le réseau. Le cinquième noeud émet un flot à QoS variant de 0 à $320kbit/s$ obtenu à partir de traces d'un film au format MPEG. Le tableau 3.1 montre le résultat du scénario précédemment décrit dans les trois versions de WLAN. L'allocation dynamique par seuil donne un taux d'utilisation proche de la version du WLAN classique avec en plus un service à QoS. En terme de délai, les valeurs sont supérieures à la version DS-CLEP. L'augmentation du délai du service à QoS provient du temps de récupération de la bande passante au service *Best effort*.

Scénario	WLAN	WLAN + DS-Clep	WLAN + seuil
Taux d'utilisation(%)	65.67	42.82	61.50
Transmis(pkt)	197359	136487	188693
Collisions (%)	19	0.4	1.7
Signalisation(%)	0	4.4	3.3
Délai EF max (ms)	1881	35	58
Délai EF moyen (ms)	53	6	9
Délai EF σ (ms)	85	4	6

TABLE 3.1 – Evaluation comparative.

En terme de débit écoulé, la figure 3.4 présente le débit par flot de chaque service et le débit total. Les débits crêtes du flot à QoS sont respectés.

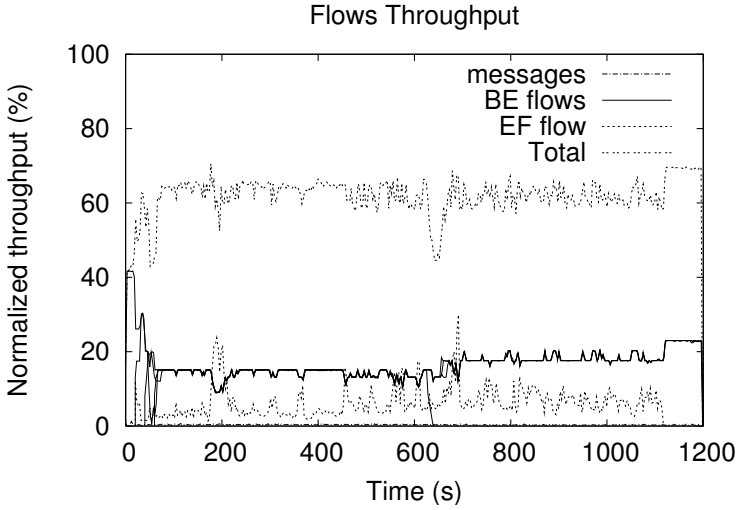


FIGURE 3.4 – Débit écoulé pour un WLAN avec système à seuil.

3.3.2 Par observation de l'accès

La gestion d'une table d'états des ressources par chaque noeud peut être un problème avec la dynamique de connectivité d'un réseau sans fil. Nous proposons d'étudier un système de contrôle sans signalisation. L'idée est de déduire un état binaire du réseau à partir d'un échantillonnage périodique du temps de réponse de l'accès au niveau de la transmission. Ce retour sert à ajuster le débit d'émission *Best effort*. Pour ses propriétés de convergence et d'équité, la fonction de contrôle utilisée est AIMD (*Additive Increase Multiplicative Decrease*). Le débit d'émission augmente par incrément et diminue selon un facteur multiplicatif. La figure 3.5 présente le débit total écoulé et le débit du flot à QoS. Le service à QoS fonctionne. Mais on remarque une plus grande instabilité du débit pour les flots *Best effort* et donc du débit total écoulé. L'absence de synchronisation globale correspond bien aux réseaux sans fil, dans lesquels les noeuds peuvent se connecter et se déconnecter très rapidement du fait de leur mobilité. Cette flexibilité dans la connectivité a un coût en terme d'efficacité. Cette étude a été présentée à une conférence [25].

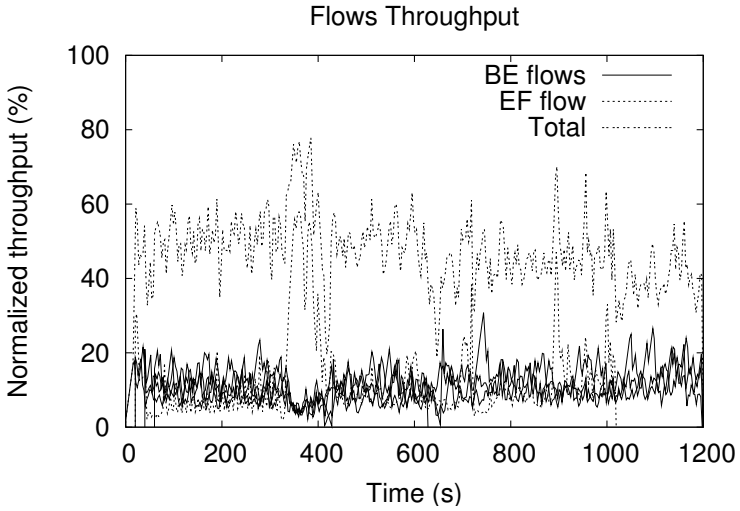


FIGURE 3.5 – Débit écoulé avec un contrôle sans signalisation.

3.4 Synthèse

Nous avons présenté dans cette partie une contribution pour traiter la QoS au niveau des accès sur les supports à diffusion. Le principe consiste à contrôler les demandes de transferts afin de ne pas saturer le support. Ce système de contrôle s'ajoute au dessus de la couche de transmission. La première partie, DS-CLEP améliore la rapidité du partage selon une équité Max-min et offre un service à QoS tout en maintenant un débit écoulé important par le support.

Dans le contexte des réseaux locaux complets sans fil, la seconde partie s'attache à maintenir un taux d'utilisation élevé du support quelle que soit la localisation des flots *Best effort* par rapport aux flots à QoS. Cet objectif est atteint tout en conservant un surdébit de signalisation faible. Nous avons montré également qu'un service à QoS peut être offert sans signalisation. L'état du réseau peut se déduire par l'observation du délai d'accès au support. Ces travaux ont été développés dans le cadre de la thèse de Fanilo Harivelo.

3.5 Conclusion

Bien que les travaux de QoS sur Ethernet aient atteint leurs objectifs, les progrès dans la capacité d'écoulement des réseaux et l'utilisation des commutateurs ont rendu moins pertinent les approches à QoS. La commutation et les débits importants n'étaient pas des caractéristiques des réseaux Wi-Fi. Dans [26], les auteurs montrent que la méthode d'accès IEEE 802.11 rend un service uniquement *Best effort*. De ce fait, la communauté réseau a publié un nombre important d'études et de propositions pour rendre des services à qualité prévisible ou différenciée sur les réseaux sans fil. Les auteurs de [27] classent ces méthodes en trois catégories :

- Différenciation de service MAC. L'accès au support est fonction de la priorité associée à la trame à transmettre. A titre d'exemple, l'extension IEEE 802.11e rentre dans cette classe de méthode. A cause de l'inefficacité de la méthode d'accès, la différenciation de service ne fonctionne pas bien sous des conditions de forte charge de trafic [27].
- Mécanismes de QoS pour la couche d'adaptation au support. La puissance d'émission du signal est modulée en fonction de la qualité du service. La proposition faite par les auteurs de [28] suit ce principe. Dans la mesure où un noeud n'a aucune connaissance des conditions de trafic, des nouveaux flots peuvent apparaître et saturer le service de transmission à QoS.
- Mécanismes de contrôle d'admission et de réservation. Dans [29], un contrôle d'admission est effectué par le routeur local du réseau sans fil puis la ressource est réservée à un flot. Dans cette solution, avant de transmettre, il faut effectuer un échange de signalisation. Le contrôle de trafic est fait au niveau IP et donc est indépendant de la couche MAC.

Notre contribution de QoS sur réseau sans fil se classe dans la dernière catégorie. Elle se différencie des autres par l'absence de relation hiérarchique entre les noeuds. Notre dernière contribution infère les conditions du trafic par des informations locales. Ceci donne au noeud une propriété d'autonomie qui est intéressante pour les réseaux pouvant s'auto-organiser. Un problème reste non résolu, c'est celui de l'allocation de la ressource pour le service à QoS notamment pour éviter qu'un réseau s'auto-organise avec des noeuds dont la somme des ressources allouées dépasse la capacité disponible.

De nos jours, la capacité d'écoulement des réseaux sans fil a beaucoup progressé du fait de l'utilisation de la technique MIMO (*Multiple-Input and Multiple-Output*). Cette technique utilise un mode de transfert multi-canal et permet donc d'augmenter le débit total. La norme IEEE 802.11n publiée en 2009 vise des débits de plusieurs centaines de $Mbit/s$. Actuellement, le comité 802.11 travaille à la norme Multi-Gigabit dont le principe consiste à orienter un faisceau hertzien vers le destinataire [30]. On assiste à une transformation du support hertzien d'un mode partagé en mode dédié. Les commutateurs WI-Fi ne sont plus très loin. Cependant, si la profusion de bande passante a fait disparaître les demandes de services différenciés, il n'est pas sûr que cette capacité soit utilisable par des terminaux dont l'énergie est limitée. En effet lorsque l'on sait que la puissance consommée par la transmission est de plusieurs ordres de magnitude supérieure à celle consommée par le traitement, on peut craindre que les demandes de communication avec QoS fassent leur réapparition.

Chapitre 4

Services de communication à la couche de transport

Ce chapitre traite de l'aléa de congestion qui est l'expression paradoxale de la contention d'accès au support dans les réseaux commutés. Cette contention provient de demandes simultanées de transmission. Elle est intrinsèque du mode de transfert asynchrone des réseaux en mode paquet comme l'Internet. La congestion et plus généralement la dynamique du trafic constitue depuis le milieu des années 1980 un sujet largement étudié. En l'absence de solution universelle, c'est encore de nos jours un *hot topic* de la communauté réseau.
Je présente dans ce chapitre mes contributions pour la fourniture de services de communication prenant en compte la congestion.

4.1 Problématique abordée

Les années 90 ont vu l'émergence de nouvelles contraintes au niveau du service. Ces dernières issues d'applications temps réel ont commencé à adopter les réseaux en mode paquet. Parallèlement à cette évolution des usages, une forte activité en recherche était menée afin de concevoir le réseau du vingt et unième siècle. On parlait alors d'autoroute de l'information. Plus techniquement, le concept était à l'intégration de services et à la technologie ATM. Du coté Internet, le succès allait grandissant avec le Web. Les premières craintes sur des problèmes provenant de sa taille apparaissaient. La contention grandissante sur les ressources rendait le service Internet de plus en plus imprévisible. La congestion était assez marquée pour que le support d'applications temps réels ne soit pas satisfaisant. Influencé par l'ATM, le concept de Qualité de Service (QoS) est apparu pour l'Internet, tout d'abord, au travers de l'architecture à intégration de services (Intserv) sous la forme d'un contrôle strict puis sous la forme d'une gestion de classes de service avec l'architecture à différenciation de services (Diffserv). Le concept de qualité de service a périclité avec le temps pour des raisons de complexité et aussi à cause de la profusion de bande passante offerte par les supports optiques. La congestion est devenue moins marquée et le service *Best effort* est (re)devenu suffisant. Les architectures à qualité de service ont pour principe d'ajouter des contrôles afin que le trafic ne puisse pas saturer les ressources du réseau. La congestion ne peut plus se produire. Le service de communication devient contrôlable et prévisible.

Mes activités dans ce domaine ont évolué dans le temps en fonction de ces changements de contexte et des avancés de mes travaux. C'est en tout quatre types de problèmes qui ont été abordés :

– Dans un premier temps, l'aléa de la congestion a été traité dans le contexte d'une architecture de QoS à différenciation de services. Mon objectif était de développer une architecture de routeur pour obtenir un ensemble de trois classes de service.
– En fait, avec la classe de service élastique, TCP (*Transmission Control Protocol*) n'arrive pas à bénéficier de l'assurance de débit moyen. Le but a été d'étudier un conditionnement des flots TCP permettant de garantir un débit.
– Que ce soit dans le contexte du service élastique ou du service *Best effort*, pour que TCP fonctionne correctement, il faut que la congestion soit détectée. La détection de congestion comporte du bruit tel que le déséquencement. L'objectif a été d'inférer avec une forte probabilité la congestion pour le notifier à TCP.
– Dans le contexte du service *Best effort*, si l'aléa de congestion affecte de préférence les flots pouvant mieux le supporter et épargne les flots les plus sensibles, la performance globale du service est améliorée. C'est avec cette idée que j'ai évalué un mécanisme pour favoriser les flots courts de TCP.

La séquence de ces quatre problématiques structure la suite de ce chapitre.

4.2 Architecture @IRS

Dans le cadre du projet national RNRT @IRS (*Architecture Intégrée de Réseaux et de Services*) et en collaboration avec le LAAS et France Télécom R&D, nous avons étudié dès 1999 une plateforme de réseau pour fournir des classes de service IP. La conception de cette plateforme fait suite aux premières spécifications de l'architecture Diffserv publiées en 1998 par l'IETF. Les objectifs de ce projet consistaient à développer et analyser de nouveaux mécanismes et protocoles (QoS et multicast) pour l'internet dans un environnement de réseaux hétérogènes (réseaux locaux, réseaux filaires et sans fil) en vue de supporter des services de communication pour un large spectre d'applications.

Dans le domaine de la QoS sur réseau filaire, mes contributions ont porté sur :
– la définition de l'architecture @IRS et sur l'étude des éléments fonctionnels pour la gestion de la QoS sur le chemin de données dans un routeur en vue de fournir les services identifiés ;
– la mise en oeuvre de cette architecture sur une plateforme et mesure de performance des services ;
– l'identification des problèmes à l'utilisation d'un service réseau à QoS pour les flots TCP.

4.2.1 Définition de l'architecture @IRS

En premier lieu, les services ont été définis. Cette réflexion sur le nombre de services et leurs caractéristiques a été menée, par le groupe de travail d'Intserv [31]. L'architecture Diffserv [32] s'appuie aussi sur un découpage des services en trois catégories assez similaires à celui proposé par Intserv. Ce découpage en trois classes de service (par défaut, intermédiaire et absolu) repose sur le constat que les applications se divisent en fonction de leurs exigences de QoS schématiquement en deux catégories [33].
– Les applications interactives, qui en plus du débit posent des contraintes sur les délais de transit de chaque paquet et la variabilité de ces délais.
– Les applications dites élastiques, qui consistent typiquement en un transfert d'un ou plusieurs documents de grande taille. Elles sont sensibles uniquement au temps de transfert total du ou des documents. La QoS requise porte sur le débit moyen reçu au cours du transfert, et non sur le délai de transit de chaque paquet.

La sémantique des services développés pour la plateforme d'expérimentation @IRS s'appuie sur cette analyse des besoins applicatifs. Les services proposés sont classés en trois catégories :

- un service sans qualité de service (*Best effort* : BE) ;
- un service à débit assuré (*Assured service : AS*) qui améliore la qualité du service BE. Le service AS est conçu pour les flots élastiques issus des applications adaptatives. Les flots de ces applications ont un débit qui augmente tant qu'il y a des ressources disponibles et diminue quand une congestion apparaît. Le débit de ces flots se décompose en un débit minimum et en un débit opportuniste. Le premier correspond à un débit minimum assuré et invariant et ne doit pas souffrir de la congestion. Le second correspond à la partie variable du débit dans lequel les paquets sont acheminés par le réseau sur le principe du "au mieux" (*Best effort*). Le débit opportuniste doit varier en fonction de l'état des ressources utilisées, d'où son caractère élastique. Il demande à l'utilisateur du service réseau d'adapter son débit opportuniste à la capacité du moment du réseau ;
- un service à garanties fermes (*Guaranteed Service : GS*) pour des applications rares qui posent des contraintes de QoS fortes et qui, de plus, ne supportent pas des variations de la QoS. Ce service est destiné aux flots déterministes et est souvent assimilé à une émulation de ligne louée.

L'architecture Diffserv suppose un provisionnement du domaine au regard des trafics générés. Dans le contexte de notre plateforme, le provisionnement a été fourni par hypothèse d'un sur-provisionnement des ressources. Cette hypothèse est importante car elle nous a fait évacuer le plan de contrôle pour nous laisser nous concentrer sur le plan de données. La préservation de l'intégrité d'un flux au sein de la classe nécessite un contrôle d'accès. La politique de contrôle retenue a été :

- un écartement des paquets GS non conformes au profil de trafic annoncé ;
- un marquage prioritaire à la perte (noté OUT) des paquets AS opportunistes.

Ces traitements sont mis sur l'interface d'entrée du premier routeur rencontré (appelé routeur de bordure du domaine). Enfin, sur l'interface de sortie de tous les routeurs, l'ordonnancement des paquets a été défini de la façon suivante :

- traitement prioritaire des paquets GS (*priority queuing*) ;
- traitement à priorité pondérée (*weighted fair queuing*) entre paquets AS et BE et une politique discriminatoire de gestion de file d'attente en cas de congestion pour les paquets AS.

La mise en oeuvre de ces fonctions au niveau des routeurs est illustrée dans la figure 4.1. L'architecture complète de la plateforme a été présentée dans [34]. La justification des choix pour le service GS a été menée par T. Bonald [35]. Avec T. Ziegler, nous avons évalué la performance offerte lorsque le réseau gère l'ensemble des flots en deux classes de trafic [36]. Ces travaux ont validé les choix de mise en oeuvre de la classe AS et de la classe BE. Pour qu'une classe de service offre un meilleur service qu'une autre classe, il faut que le paramètre de pondération de WFQ soit supérieur à la charge dans la classe. D'où l'importance du plan de contrôle pour un provisionnement correct de la classe. La politique discriminatoire à base de gestion probabiliste des pertes de type RED a été remise en question à cause de l'instabilité et des oscillations introduites par RED [37] [38]. Pour ces raisons, son utilisation dans les routeurs reste problématique [39]. Notre solution repose sur une politique déterministe mise en œuvre par un mécanisme à simple seuil de type PBS (*Partial Buffer Sharing*) [40]. Les paquets opportunistes sont systématiquement rejetés dès que le nombre total de paquets en attente dans la file excède un certain seuil. Le choix d'une politique à base d'expulsion, un moment envisagé, n'a pas montré un

avantage significatif [41].

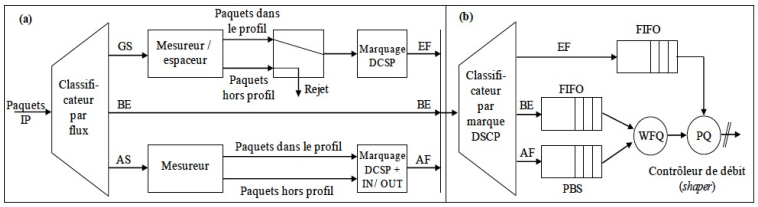


FIGURE 4.1 – Traitements dans les routeurs : (a) à l'entrée du premier routeur rencontré ; (b) à la sortie.

4.2.2 Déploiement de la plateforme

La plateforme @IRS (ou @IRSBone) a été construite sur les bases d'une plateforme ATM (RENATER2), offrant des liaisons de type VP (*Virtual Path*) de bout en bout de type CBR. Un site est connecté au @IRSBone par le biais d'un routeur de bordure. Ces routeurs sont eux-mêmes connectés au routeur de coeur. Les liaisons ATM sont tirées entre routeurs de coeur. Les développements des mécanismes pour les routeurs ont été réalisés au LIP6 pour le système d'exploitation FreeBSD.

Les mesures de performances concernant les applications interactives ont été effectuées en collaboration avec C. Chassot du LAAS. L'objectif visait à évaluer :

- les caractéristiques d'une classe de QoS en présence de trafic sans qualité de service,
- les caractéristiques de QoS lorsque toutes les classes de QoS sont utilisées simultanément et conjointement à du trafic sans qualité de service.

Les résultats ont permis de conclure positivement sur l'adéquation des choix d'architecture aux caractéristiques des services attendus : protection parfaite du flux GS et accroissement acceptable du délai pour le flux AS. Ces résultats sont détaillés dans [42, 43, 44].

Avec Emmanuel Lochin, nous nous sommes intéressés à l'évaluation de la QoS des applications générant un trafic élastique. Notre objectif tenait dans cette question : dans quelle mesure l'assurance de débit peut-elle être contrôlée ? Pour répondre à cette question, nous avons défini un test avec un flot de référence ayant une assurance de débit qui est en concurrence avec d'autres flots de la même classe de service. La somme des assurances données est équivalente au provisionnement du service.

Le flot de référence va être conditionné selon des profils différents au moyen d'un *token bucket* de paramètres (r, b). Le paramètre de débit r prend la valeur de l'assurance. Une sporadicité b importante combinée avec celle inhérente de TCP accroît les rafales de paquets dans la partie du débit minimum du service. Pour éviter ce genre de situation, la taille du seau b est invariable et est équivalente à la taille d'un paquet, soit 1024 octets.

La figure 4.2 représente l'évolution du débit utile normalisé par rapport à R_{AS} (débit alloué à la classe AS) du flot de référence en fonction du nombre de flots AS supplémentaires. Cette figure montre une forte variation du débit du flot de référence lorsque le nombre de flots AS augmente.

Le flot de référence n'est pas isolé des conditions de trafic. L'augmentation du taux r du *token bucket* du flot de référence ne se traduit pas par une augmentation significative du débit utile. Quand n tend vers l'infini, le débit du flot de référence converge quel que soit son assurance. Ce test montre également que le *token bucket* est un très mauvais marqueur pour les flots TCP car il ne prend pas en compte la sporadicité de TCP [45]. Les résultats de cette étude ont été publiés dans [46, 47]

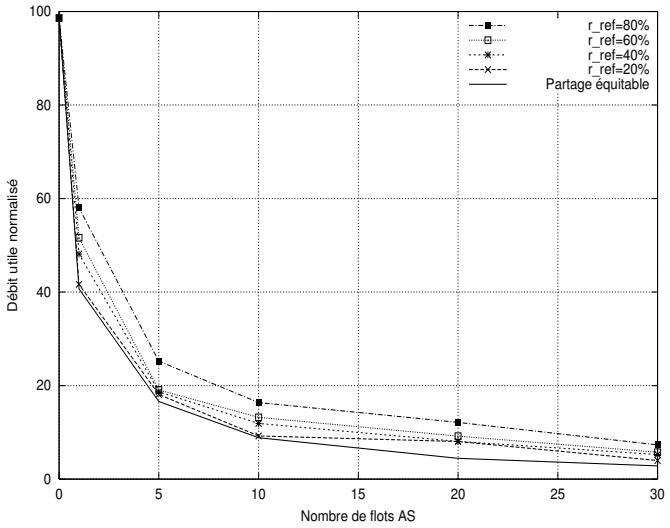


FIGURE 4.2 – Débit utile du flot de référence en fonction du nombre de flots AS.

4.2.3 Bilan

Le bilan de cette activité est plutôt positif : la plateforme fournit bien des services à QoS au niveau IP. D'ailleurs, le projet a eu une suite sur la problématique de plan de contrôle dont nous avons vu l'importance (projet @irs++). Cependant, nos travaux ont démontré que dans l'état, cette architecture n'était pas utilisable par des flots élastiques. Il existe un problème de conditionnement des flots TCP vis à vis de la valeur de l'assurance. C'est à partir de ce constat que nous avons commencé l'étude de l'assurance de débit pour TCP que nous détaillons dans la partie suivante.

4.3 Service assuré pour TCP

Comment TCP peut-il bénéficier d'un service à QoS ? En collaboration avec Emmanuel Lochin, nous avons travaillé pour répondre à cette question. Les mesures faites dans le projet @IRS ont montré que la difficulté ne venait pas des mécanismes internes mis dans les routeurs. Le problème résidait dans le marquage des paquets du flot TCP. La difficulté de donner une garantie de débit à un flot TCP utilisant la classe de service assurée provient de plusieurs caractéristiques intrinsèques à TCP tel que :

- La fonction de contrôle du débit : la perte d'un paquet même hors profil est fortement préjudiciable pour son débit de transfert. Après la perte, la fenêtre de contrôle de congestion a été divisée par deux, son débit d'émission peut devenir inférieur au débit nominal.
- La durée de la boucle de contrôle (RTT : *Round Trip Time*) : la différence de RTT entre les flots influe sur le débit d'émission.

Nous avons vu précédemment qu'une solution reposant sur le *token bucket* est inappropriée. Cela a été démontrée dans [48].

4.3.1 Contribution

Les travaux dans ce domaine cherchent par différentes méthodes à jouer sur la probabilité de perte des paquets hors profil (partie opportuniste du débit). Nous avons orienté nos travaux vers le contrôle de la sporadicité. C'est à dire à jouer plutôt sur les aspects temporels du flot TCP.

Nous avons remarqué que si nous appliquons un lissage au débit assuré à un flot TCP (tous les paquets sont dans le profil), celui-ci obtient un débit constant quel que soit le nombre de flots dans la classe de service. Le flot n'enregistre aucune perte. Cependant, le lissage du trafic ne peut être une solution pour obtenir une différenciation de services car en n'émettant aucun paquet hors profil, le débit de la source est limité à son débit minimum (indiqué par le profil). Elle ne peut donc concourir à obtenir la bande passante en excès si elle existe. Dans ce cas, le service AS peut devenir un service moins performant que BE. Cette expérience amène à penser qu'il est possible d'effectuer une différenciation entre les flots par une priorité spatiale mais que cette différenciation dépend en grande partie du conditionnement.

Le RFC2990 [49] précise que le comportement sporadique de TCP introduit des difficultés pour le réseau et a des répercussions directes sur le débit écoulé. En effet, les rafales risquent de conduire à des pertes par débordement de file d'attente. Notre orientation a été de diminuer le caractère sporadique du flot TCP au moyen d'une remise en forme. Notre proposition porte sur un *penalty shaper*. Le principe n'est plus de favoriser les flots qui n'ont pas atteint leur assurance mais de pénaliser ceux qui sont au dessus de leur assurance. L'idée est de faire libérer les ressources par ces flots opportunistes (flots qui surpassent leur assurance) pour que les flots qui n'ont pas leur assurance puisse l'atteindre. La pénalité sous forme d'un délai se déclenche lorsque des pertes de paquets hors profil apparaissent dans le réseau. La pénalité revient à augmenter la durée de la boucle de contrôle et donc à ralentir le débit d'émission. Ce choix nous a paru plus pertinent pour TCP que d'influer sur la probabilité de perte. On sait que TCP réagit fortement aux pertes alors que les variations de RTT sont mieux acceptées, bien qu'elles peuvent avoir un impact sur le calcul du délai de garde du temporisateur de retransmission.

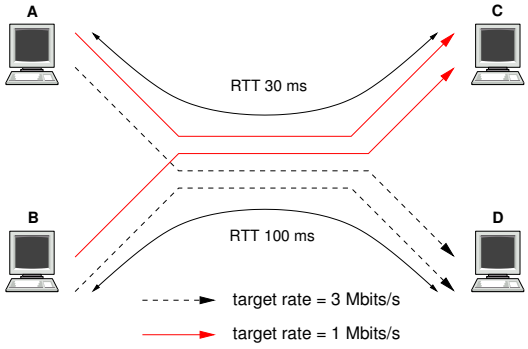
La mise en oeuvre de ce conditionneur demande :

- une interaction avec les routeurs de coeur pour apprendre l'existence de pertes de paquets hors profil,
- une mesure du débit d'émission pour savoir si le flot émis est opportuniste,
- une fonction de contrôle de la pénalité.

L'état de congestion est retourné au conditionneur (localisé du coté émetteur) par un marquage de type ECN (*Explicit Congestion Notification* [50]). Lorsqu'un paquet hors profil est supprimé, tous les paquets en profil présents dans la file reçoivent une marque de congestion. Le récepteur doit réfléchir cette marque sur l'acquittement. Lorsque le conditionneur (du coté émetteur) reçoit cette indication, il applique une pénalité si le débit d'émission est supérieur à la valeur nominale de l'assurance. Dans le cas contraire, la marque est ignorée. Sachant que la stabilité de l'Internet repose sur une fonction de contrôle AIMD (*Additive Increase Multiplicative Decrease*)[51], cette dernière présente une propriété de convergence vers l'équité [52]. Le délai traduisant la pénalité évolue selon la fonction de contrôle AIMD. Le but recherché est d'appliquer une pénalité multiplicative pour les flots les plus opportunistes et donc d'avoir une convergence des débits des flots opportunistes. En l'absence d'indication de congestion, la pénalité diminue selon une pente additive. Cette proposition offre l'avantage de pouvoir s'utiliser sur un groupe de flots TCP car elle ne demande aucune détermination des paramètres de performances de TCP comme la mesure du RTT ou du taux de pertes.

Nous montrons dans le tableau de la figure 4.3(b) les résultats de l'assurance donnée avec et sans notre proposition APS (*AIMD Penalty Shaper*). Pour cette évaluation nous mettons en compétition quatre agrégats avec différents RTT, débits assurés et nombre de flots. L'agrégat est constitué par un ensemble de flots TCP ayant la même source et destination. Le scénario est représenté par la figure 4.3 (a). En se plaçant dans le cas le plus défavorable à savoir garantir un débit assuré élevé aux flots ayant un fort RTT,

le tableau 4.3 (b) nous montre que chacun des agrégats respecte son débit assuré avec l'APS. Ce conditionnement permet bien à TCP d'avoir un partage de la bande passante de manière prévisible et contrôlée et non selon un critère d'équité.



(a)

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

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

(b)

FIGURE 4.3 – Evaluation avec des agrégats TCP différents.

4.3.2 Synthèse

Nous avons proposé un nouveau conditionneur pour le service assuré permettant de garantir un débit de transfert à un agrégat de flots TCP. Cette garantie est indépendante du RTT, du nombre de flots TCP et du débit assuré de l'agrégat. Cette méthode de conditionnement reste simple dans la mesure où elle ne nécessite pas de calculs difficiles à réaliser comme l'estimation du RTT ou l'estimation des pertes TCP. Enfin, à part la valeur de l'assurance, le conditionneur est totalement autoconfigurable. Ces travaux ont fait l'objet de la thèse d'Emmanuel Lochin et ont été publiés [53, 54].

La faiblesse de la solution proposée réside sur le fait d'avoir recours à un système de notification de la congestion de type ECN. En effet, notre conditionnement de trafic TCP prend en compte les conditions de trafic sur la route pour le calcul de la pénalité. Cette critique est une des motivations des travaux sur la détection de congestion que je présente dans le paragraphe suivant.

Plus généralement, la QoS est un concept de bout en bout, tous les éléments entre la source et la destination doivent fournir la QoS ou autrement dit le service rendu sera celui du maillon faible. La structure distribuée et morcelée de l'Internet n'introduit aucune forme de motivation à changer le service. Pour que l'Internet offre des services de communication de qualité, il faudrait au moins une action coordonnée. Et au vue de sa structure et de sa taille, cela parait impossible. D'autant plus que la surabondance de bande passante apparue ces dernières années n'a pas été une motivation à bouger vers les architectures de QoS [55]. Les raisons de l'échec des architectures de QOS sont à chercher de ce coté là.

Plus particulièrement, la question de l'utilisation de TCP comme protocole de transport pour fournir de la QoS est posée [56]. Les auteurs suggèrent même que dans le contexte de réseaux à QoS, de nouvelles formes de contrôle de congestion pourraient être définies. En fait, TCP est retenu comme protocole de transport car il est utilisé par la majorité des applications. Ce choix entraîne des traitements coûteux en aval pour adapter le flot aux contraintes de QoS. La prise en compte de ces contraintes directement au niveau de la source semble plus appropriée. C'est le sens du bilan que nous avons publié dans la revue TSI [57].

4.4 Détecteur de congestion

Notre objectif a été de détecter la congestion en bordure du réseau. Cette action a commencé comme une suite pour notre conditionneur de flot TCP. Cependant, la détection de congestion en bordure offre de nouvelles perspectives. En effet, il devient évident que l'hétérogénéité des environnements rend obsolète l'idée d'un unique contrôle de congestion [58]. Dans l'Internet, le contrôle de congestion est effectué par la couche de transport et donc principalement par TCP. Une multitude de versions TCP ont vu le jour ces dernières années afin d'adapter le contrôle de congestion aux différentes caractéristiques des prochains réseaux ou applications [59]. Certaines versions de TCP s'attaquent au problème de l'important taux de pertes relatifs aux environnements sans fil [60] tandis que d'autres proposent des solutions au problème de l'occupation de la bande passante des réseaux haut débit [61]. Plus généralement, dans [62], les auteurs présentent toutes les déclinaisons de TCP proposées ces dernières années. Ces propositions définissent à la fois une méthode de détection de la congestion et une fonction de contrôle de la fenêtre d'émission en fonction des caractéristiques du réseau sous-jacent. La conclusion à tirer est qu'il n'existe pas de version TCP universelle capable de se comporter de façon performante sur différents types de réseaux.

L'idée que j'ai développée avec Emmanuel Lochin et Fanilo Harivelo est de décorreler l'algorithme de détection de la congestion de TCP pour l'adapter aux caractéristiques de l'Internet. Cette idée est également étudiée dans [63]. Les auteurs préconisent de découper la couche transport en trois sous-couches distinctes pour mieux répondre aux caractéristiques des nouveaux réseaux. Le but recherché est d'optimiser le contrôle de congestion au système de transmission sous-jacent utilisé, d'avoir un déploiement incrémental de nouveaux contrôles de congestion dans un domaine particulier et d'offrir de nouvelles marges de manoeuvre pour l'évolution de contrôle de congestion de TCP.

Dès 2007, nous avions travaillé sur la dissociation de l'algorithme de détection des pertes de celui du contrôle de la congestion de TCP. Nous avons tout d'abord étudié un mécanisme de détection nommé ICN (*Implicit Congestion Notification*). Ce mécanisme pourra se coupler par la suite avec TCP.

4.4.1 Conception

Dès 1994, S. Floyd a proposé d'attribuer la détection de congestion au niveau du réseau avec la proposition ECN (*Explicit Congestion Notification*)[64]. Le but est d'éliminer l'ambiguité du signal de congestion à l'interprétation d'une perte et d'augmenter la performance de TCP en diminuant le taux de pertes. Dans [65], l'auteur considère la pertinence du marquage ECN et démontre que les sources utilisant ECN obtiendront un gain de performances significatif même si le déploiement d'ECN n'est que partiellement supporté. En 2005, le taux d'utilisation de ce mécanisme au sein de l'Internet est de $2,1\%$ [66]. La taille de l'Internet est devenue telle que le déploiement de nouvelles solutions semble de plus en plus compliqué à mettre en oeuvre. Une solution possible à ce problème du déploiement de nouvelles solutions dans l'Internet serait celle proposée dans [67] où les auteurs proposent d'émuler la gestion active de la file d'attente (AQM : *Active Queue Management*) dans le système d'extrémité. Ceci revient en quelque sorte à mettre la détection de congestion dans l'extrémité sans déploiement d'AQM au sein du coeur du réseau.

Notre proposition suit cette idée : découpler la détection de congestion du protocole transport tout en la laissant à l'extrémité du coté de la source. La source est notifiée de la congestion par l'extrémité de manière similaire à la proposition ECN. Cette séparation entre détection de congestion et contrôle de congestion 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. 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 ;
– 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 être réalisées telle que la mesure du taux de congestion pour un flot ou pour une destination ;
– la détection d'une congestion peut être factorisée par un gestionnaire de congestion [68] et ainsi fournir aux nouveaux flots une indication de la capacité disponible. Ces précisions au démarrage de connexion peuvent avoir un impact sur la performance des flots ayant un échange bref [62].

La détection de la congestion n'étant plus du ressort de TCP, nous proposons de désolidariser le contrôle de perte du contrôle de congestion. Nous envisageons la version du protocole TCP dite *TCP robust* qui ne réagirait uniquement qu'aux messages ECN provenant du module de détection de la congestion ICN. Celui-ci serait localisé dans une *middlebox* [69] ou dans le routeur d'accès. Ainsi, ce module notifierait TCP en fonction d'estimations de congestion retournées par une ou plusieurs méthodes de détection.

4.4.2 Evaluation

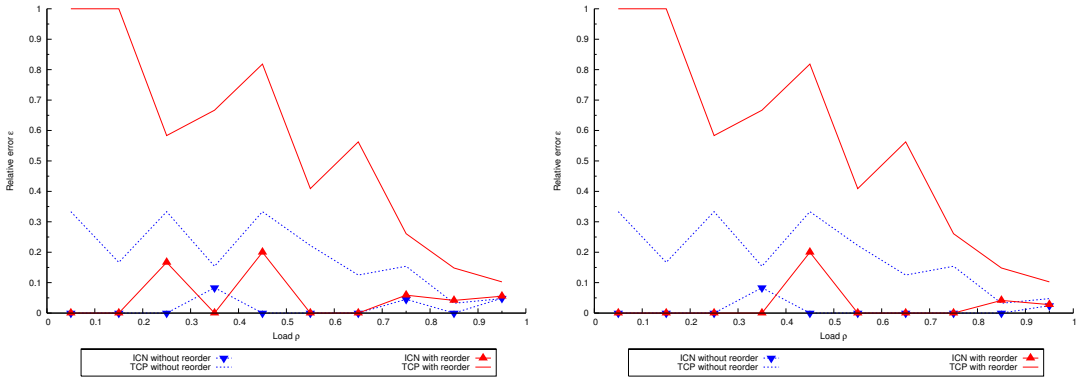
Nous avons validé le principe de détection de congestion à l'extérieur de TCP par le développement d'une méthode de détection des événements de congestion (CE : *Congestion Event*) [1]. La notion d'événement de congestion indique une fenêtre de données avec un ou plusieurs paquets perdus ou marqués ECN [71]. En d'autres termes, TCP ne réagit pas à proprement parlé au nombre exact de paquets perdus ou marqués ECN mais à un événement de congestion. Nos travaux ont repris les travaux d'identification des pertes développés à partir de l'observation des retransmissions [72, 73]. Les observations de pertes seront complétées par l'identification de la taille de la fenêtre. La détection des retransmissions ne suffit pas pour identifier une perte. Il faut pouvoir s'assurer que la retransmission ne s'est pas produite suite à une erreur de TCP. Ce cas doit être pris en compte pour ne pas surestimer les CE dans les routes perturbées par les déséquencements ou les fortes variations de RTT. La difficulté a porté sur ce point précis : l'identification des retransmissions abusives. TCP est lui même confronté à ce dilemme entre reprise rapide et reprise juste. 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. Des extensions à TCP ont été conçues pour l'aider à identifier une congestion en présence de bruits comme de fortes variations de RTT [74], de déséquencements dans le réseau [75] ou de réception de données dupliquées[76].

Nous voulons identifier les retransmissions inutiles en analysant le flot TCP. Nous avons développé deux versions de notre algorithme qui se différencie par la méthode d'élimination de l'ambiguïté de la retransmission. L'ambiguïté de la retransmission est définie comme l'incapacité de l'émetteur TCP de distinguer l'acquittement de la transmission originale de l'acquittement de la retransmission. La première version effectue une identification des retransmissions abusives en ajoutant un délai de validation. A la

1. Le terme événement de perte (LE : *Loss Event*) s'utilise pareillement et désigne la même notion [70]

suite d'une retransmission, le délai de réception de l'acquittement correspondant est relevé et comparé avec le RTT. Si le délai est proche du RTT, la perte est réelle. Autrement, la retransmission est classée comme abusive. La seconde version utilise l'option d'estampille temporelle (*timestamp*) de TCP de manière similaire à l'approche Eifel [75]. L'estampille temporelle sert ici à déterminer quel segment à déclencher l'émission de l'acquittement reçu. Une retransmission abusive est détectée lorsqu'un acquittement est reçu avec l'estampille temporelle de la transmission initiale.

Nous avons évalué les versions de notre algorithme au moyen du simulateur ns-2. Nous avons recherché à concevoir un modèle avec des motifs de congestion variés et en introduisant des déséquencements. Le trafic généré représente des requêtes HTTP à savoir majoritairement des flots TCP à durée de vie courte. Une fois l'état stationnaire atteint un flot de référence est injecté dans le modèle. C'est la trace de ce flot qui est analysée. La figure 4.4 représente le taux d'imprécision d'ICN et de TCP avec ou sans déséquencement du réseau. On notera que l'imprécision diminue par rapport à la charge du fait du nombre croissant de congestion (le taux de déséquencement reste constant). Les séquences spéciales sont en proportion plus importantes à faible charge. C'est la raison qui explique les fortes variations. Les séquences spéciales sont des cas d'erreurs introduits par le déséquencement comme par exemple une vraie perte qui se produit dans un faux événement de congestion.



(a) Composante temporelle (b) Estampille temporelle

FIGURE 4.4 – Imprécision en fonction de la charge de TCP et ICN.

4.4.3 Synthèse

Notre contribution a formalisé et vérifié un algorithme de détection des événements de congestion à partir d'une observation passive des traces d'un flot TCP. Le résultat majeur de cette étude est la preuve de la possibilité de décorréler complètement la détection de la congestion de l'algorithme TCP. De plus nous avons pu établir la fiabilité apportée par notre algorithme dans la détection des événements de congestion. Cet algorithme a pour contrainte de rester simple afin de pouvoir s'exécuter de manière synchrone au flot TCP. Nous l'avons voulu déployable rapidement. Pour cela il ne demande pas de modification du réseau ou une collaboration de la destination (le mécanisme est dit orienté émetteur). L'ensemble des résultats obtenus sont disponibles dans [77, 78].
Ma perspective reste le couplage d'ICN avec notre version de TCP (nommée *TCP robust*). L'élaboration d'un détecteur de congestion multi-protocoles ou multi-critères reste également un axe intéressant.

4.5 Favorisation de TCP

Le modèle de service de communication évolue assez peu dans le réseau. Nous l'avons vu avec les travaux sur la QoS. La croissance de l'Internet est un facteur d'ossification. Cependant, la congestion se situe au niveau des routeurs tandis que les sources de trafic et le contrôle de congestion sont dans les hôtes. L'idée serait alors d'impliquer les routeurs dans la gestion de la congestion pour améliorer le service. Cette idée a été largement étudiée depuis une vingtaine d'années au travers de différentes propositions de gestion de la file d'attente (AQM : *Active Queue Management*). Le but poursuivi est de renforcer l'équité entre les flots. La détermination du paramètrage optimal et robuste de ces mécanismes est une tâche compliquée. Cela serait une des raisons avancées pour expliquer leur faible usage dans l'Internet. Cependant, l'aide du réseau dans le contrôle de congestion est encore de nos jours un défi [79].

Avec Emmanuel Lochin, nous avons travaillé pour diminuer la latence des flots courts. La latence se définit comme le délai nécessaire pour effectuer le transfert d'un fichier de bout en bout. Une définition courante pour qualifier un flot TCP court serait un transfert qui ne peut pas utiliser la procédure de *fast retransmission* pour corriger une perte, du fait du faible nombre de segments échangés [80].

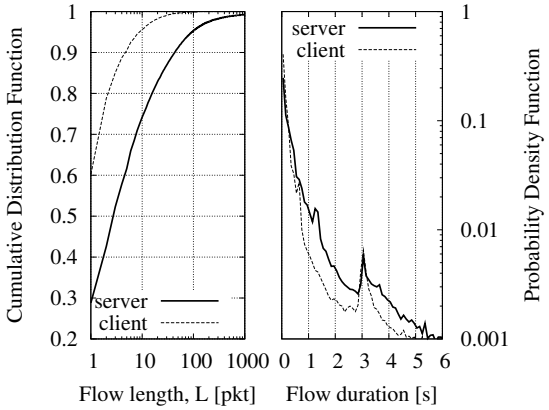


FIGURE 4.5 – Distribution de la latence des flots TCP. Extraite et reproduite avec l'autorisation des auteurs de [81].

4.5.1 Problématique

Des études récentes montrent que le trafic Internet est généré majoritairement par le web et caractérisé par des connexions TCP très courtes [82]. D'après les mesures faites par [81], 95% des flots émis par les clients et 70% des flots serveurs ont une taille inférieure à 10 paquets. Comme le signale [83], la pratique courante du web consiste à écrire des pages avec peu de contenu afin de diminuer les latences de transfert et améliorer l'interactivité de la navigation. En effet, nous avons pu tous constater que le texte s'affiche en premier suivi par des images et se termine par les éléments volumineux. Cette interactivité est obtenu au prix de l'établissement de plusieurs connexions courtes. Pour cette raison, bien que le trafic Internet soit en forte croissance, les flots courts garderont une position dominante dans l'Internet de demain. Il semble donc pertinent de continuer à réfléchir à une stratégie de prise en charge de ce trafic. Surtout lorsque l'on sait que les connexions TCP courtes et longues n'obtiennent pas les mêmes performances. Les flots courts ont leur dynamique d'émission conduite par la procédure de contrôle du *slow-start*. Celle-ci entraîne un problème qui se présente sous deux aspects :

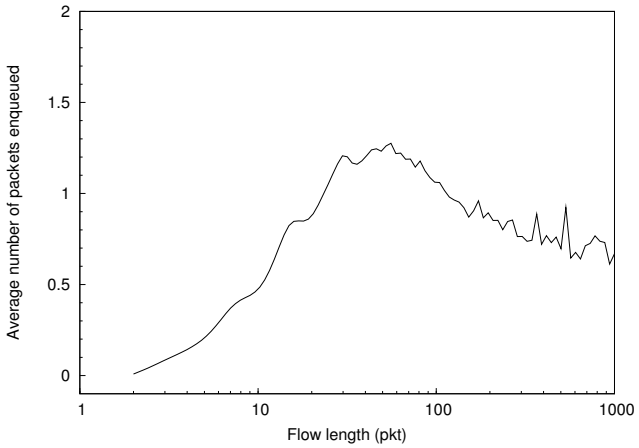


FIGURE 4.6 – Nombre moyen de paquets en attente par taille de flots.

– Les premiers échanges s'effectuent avec une petite taille de fenêtre. Le débit écoulé est faible. Ce problème s'apprécie sous le point du vue de l'équité avec les flots longs[84]. L'augmentation de la taille de fenêtre initiale a été étudiée comme une solution [85].
– La croissance exponentielle de la taille de la fenêtre produite par le *slow-start* accroit la sporadicité du flux TCP. La probabilité de perte en est augmentée. Les pertes en début de connexion sont très coûteuses en terme de délai car elles sont reprises par le temporisateur de retransmission (RTO : *Retransmission Timeout*).

Comme le montre [81], les pertes initiales dégradent fortement la latence d'un flot TCP court. La figure 4.5 présente la fonction de répartition de la taille des flots observés et la fonction de densité de probabilité de la latence (notée *flow duration*) pour ces mêmes flots. Sur la courbe de la latence, on distingue un pic à $t = 3$ secondes qui correspond, à la valeur par défaut du temporisateur de retransmission [86]. Cette courbe montre bien que les connexions TCP courtes souffrent d'une perte initiale reprise par RTO. Les auteurs précisent que le taux de reprise par RTO est de l'ordre de 70%. Ces mesures démontrent que la phase de démarrage n'est pas transitoire pour les flots TCP courts et qu'elle joue un rôle important en terme de performance. Cette phase reste encore un des problèmes ouverts du contrôle de congestion de l'Internet [79]. De cette analyse, nous avons orienté notre proposition pour qu'elle diminue le taux de pertes pour les flots courts.

4.5.2 Proposition : FavourQueue

Nous avons cherché à concevoir une AQM permettant de protéger les flots courts de la perte. L'identification d'un flot court repose sur l'hypothèse que ce type de flot ne produit pas assez de paquets pour occuper la file d'attente. Le principe de FavourQueue est assez simple : lorsqu'un paquet arrive à l'entrée de la file, si celui-ci n'appartient pas à un flot ayant déjà un paquet en attente, ce paquet est alors protégé de la perte. La protection est produite par un mécanisme d'expulsion si la file est pleine et par une transmission prioritaire pour augmenter la protection. En effet, si les paquets non protégés sont émis avant ceux à protéger, la file peut se remplir de paquets à protéger et donc la protection disparait une fois la file pleine. D'ailleurs, la proposition Run2C [87] met en oeuvre une protection par un couplage d'un mécanisme à expulsion avec celui d'un accès prioritaire. La classification est faite avec la même idée que celle de Choke [88]. A savoir que la file d'attente est suffisante pour identifier les flots consommant le moins de ressource. Avec un modèle de simulation, cette hypothèse est vérifiée par la figure 4.6. Celle-ci montre le nombre moyen de paquets du même flot en file d'attente quand un paquet arrive dans la

file d'attente. On constate que pour des flots de taille inférieure à 10 paquets, ce nombre est toujours inférieur à celui donné pour des flots de taille supérieure à 10 paquets. Enfin, précisons que le nombre d'états que doit mémoriser le routeur est borné par la capacité de la file d'attente.

4.5.3 Evaluation

A l'aide de ns-2, nous avons reproduit le trafic web constitué de requêtes HTTP utilisant des connexions TCP sur un modèle avec une topologie dite en papillon. Les échanges sont brefs afin de rester principalement en phase de *slow-start*. Les requêtes HTTP sont réparties sur un ensemble de 10 clients-serveurs où toutes les paires ont un RTT différents. Les caractéristiques des requêtes pour un niveau de charge donné avec DropTail sont enregistrées. La même séquence de requêtes pourra être rejouée avec d'autres AQM. Cet enregistrement nous donne les moyens de comparer les résultats flot par flot.

Comme pour la figure 4.5, la courbe du DropTail de la figure 4.7 présente un pic de latence à $t = 3$ secondes correspondant à la valeur de RTO par défaut. Le ratio de la reprise sur RTO s'élève à 56%. La courbe FavourQueue de la figure 4.7 montre que le pic précédemment identifié a disparu. Les pertes initiales pénalisant les performances du trafic TCP ont diminué. Nos évaluations ont montré que 58% des flots courts ont une latence améliorée et que 80% des flots longs bénéficient également de cet AQM. En phase de non-congestion, FavourQueue n'a aucun effet sur le trafic réseau.

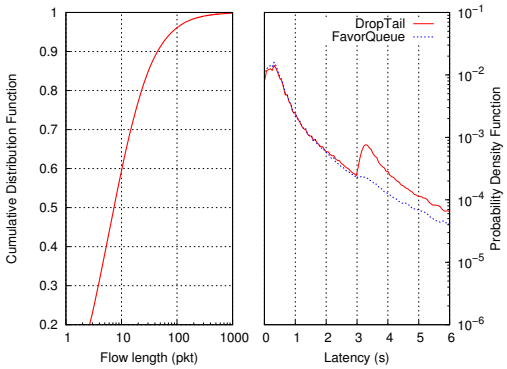
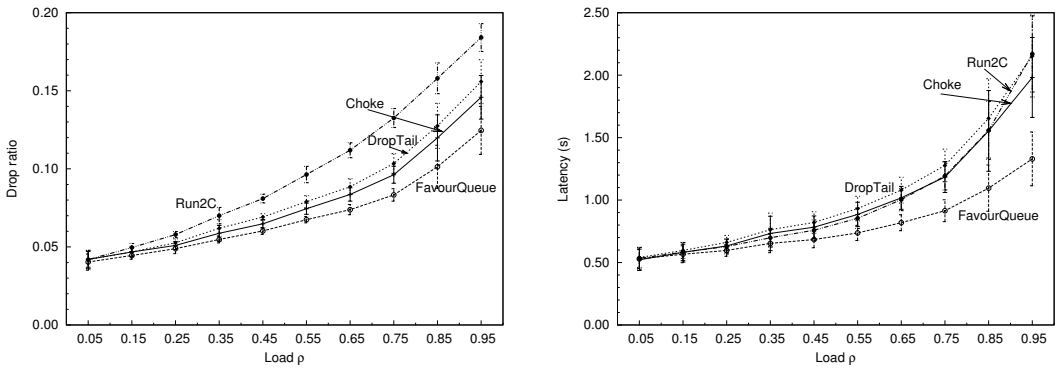


FIGURE 4.7 – Distributions de la taille des flots TCP et de leur latence avec le modèle de simulation.



(a) Taux de pertes (b) Latence

FIGURE 4.8 – Comparatif de FavourQueue avec d'autres gestionnaires de file d'attente.

Nous présentons sur les figures 4.8(a) et 4.8(b) la moyenne et l'écart type du taux de

pertes et de la latence en fonction de la charge du trafic. L'évaluation comparative de FavourQueue est réalisée vis à vis de Choke [88], Run2C[87] et la politique par défaut DropTail. Le taux de pertes le plus important ressort pour Run2C. Ceci ne se traduit pas au niveau de la latence. Les pertes sont surtout enregistrées pour les flots ayant une taille supérieure à 20 paquets. Cette valeur correspond au seuil utilisé par Run2C pour sa classification. Run2C montre que ce n'est pas la quantité qui compte mais la qualité. La perte d'un paquet n'a pas des conséquences équivalentes sur la performance des flots. C'est aussi l'orientation prise par FavourQueue.

Une analyse détaillée de FavorQueue éclaire sur son fonctionnement. Tout d'abord, le taux de pertes est plus faible. En fait, en phase de *slow-start*, il effectue une certaine remise en forme du trafic à savoir que les grumeaux de deux paquets sont brisés. Le premier paquet est protégé et passe en file prioritaire, le second reste un paquet standard mis en fin de file. Un délai est ajouté entre les deux paquets. La sporadicité du *slow-start* en est diminuée et donc la probabilité de perte de paquet également. Ensuite, les paquets isolés ne sont quasiment plus perdus. Ceci participe à fiabiliser les procédures de reprises par retransmission. En fait, le taux de reprise par RTO a diminué à 38% (contre 56% avec DropTail). La combinaison de ces effets donne l'explication du gain en latence que l'on observe. La figure 4.9 illustre l'évolution du gain (par rapport à DropTail) en latence en fonction de la taille du flot. La diminution du gain suit l'augmentation de la taille des flots, ce qui signifie que FavourQueue aide la phase d'établissement et de démarrage de la connexion TCP. Les flots longs tirent partie également du déploiement de FavourQueue. Car de manière évidente, avant d'être un flot long, un flot TCP est d'abord un flot court.

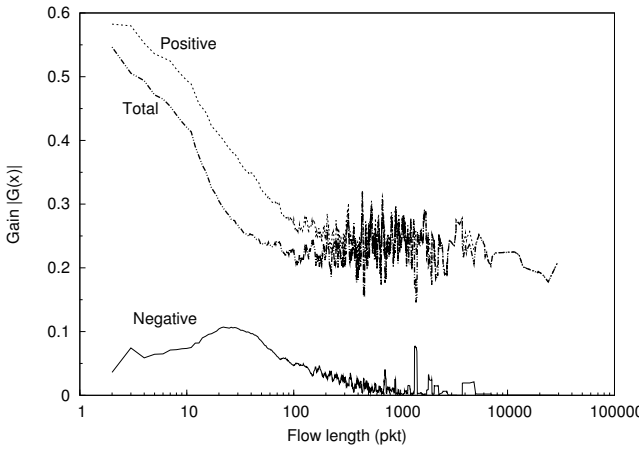


FIGURE 4.9 – Evolution du gain en latence fonction de la taille du flot.

4.5.4 Synthèse

L'analyse détaillée d'une cause de sous performance des flots courts a orienté la conception d'un AQM. Les avantages présentés par FavourQueue sont les suivants :

- le séquencement des paquets dans un flot est conservé ;
- tous les flots ne remplissant pas une file d'attente ont leur paquet protégé ;
- il n'y a pas d'état permanent 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. Le principe de conception de l'Internet de ne pas mettre d'état de la connexion dans le réseau est respecté ;
- le déploiement peut se faire indépendamment et incrémentalement ;

– le fonctionnement de cet AQM est autonome vis à vis du transport : pas d'interaction avec les extrémités ou de modification avec les extrémités ;
– il ne demande pas de marque dans les paquets, de mesure du trafic entrant ou de paramètrage ;
– les phases sensibles de TCP comme l'établissement de connexion et la reprise sur erreur sont protégées.

Ce mécanisme doit être vu comme une amélioration de DropTail qui vise à accélérer le transfert des données rendu par le service *Best effort*. Le principe consiste à favoriser localement certains paquets. Le contrôle de congestion à détection par le délai *delay-based congestion control* est plus performant que celui par la perte *drop-based congestion control*[89]. La difficulté de son déploiement provient du fait qu'il soit moins performant en présence du second. FavourQueue pourrait être une solution pour la coexistence de ces deux formes de contrôle de congestion. Cette évaluation constitue une perspective de cette proposition.

L'étude de FavourQueue a été publiée sous forme d'un rapport [90] et a été soumise pour publication dans une revue. Cette étude montre que la fiabilité est importante en phase d'établissement de connexion. La question que l'on peut se poser est : quel serait le gain si uniquement les segments SYN étaient protégés ? Avec Fanilo Harivelo et Richard Lorion, nous avons évalué un AQM qui décide de la protection sur l'analyse de l'en-tête TCP. Cette étude est publiée dans une conférence [91].

4.6 Conclusion

La principale cause de perturbations de l'accessibilité du service réseau provient de l'aléa de congestion. Dans le contexte de l'architecture à qualité de service, la congestion est contrôlée par des traitements différenciés des flots par le réseau. C'est ce qui a été montré et réalisé dans le projet @IRS. Le problème de l'usage des services à QoS est loin d'être évident. Rien que sur le plan des données, nous avons vu la difficulté d'utiliser TCP. Sa dynamique d'émission est inadaptée à un service à débit assuré. Notre contribution a porté sur un conditionneur agissant sur la composante temporelle d'un ou plusieurs flots TCP en fonction de la congestion. L'idée est de diminuer le débit des flots opportunistes sans pour autant créer des pertes.

L'étape suivante a été de pouvoir détecter les événements de congestion en bordure de réseau sans avoir recours à une signalisation. Nous avons montré que ces événements peuvent être décorrélés de TCP. Ceci ouvre la voie à une version de TCP plus modulaire qui peut prendre en compte l'hétérogénéité du réseau.

Dans le contexte de l'Internet *Best effort*, la performances des connexions TCP courtes sont fortement dégradées par les pertes au début de connexion. Notre proposition porte sur un AQM protégeant les paquets des échanges initiaux. Le second effet de ce mécanisme est de diminuer la sporadicité du *slow-start* et donc de diminuer le risque de congestion de la file d'attente. La favorisation, c'est à dire le traitement appliqué à un paquet en fonction d'une classification effectué à partir d'un contexte local mérite encore d'être approfondi. Car la phase de démarrage d'une connexion est encore de nos jours un défi à relever dans le domaine du contrôle de congestion [79].

Chapitre 5

Synthèse

5.1 Conclusion du travail effectué

Ce mémoire a présenté un ensemble de contributions dont l'objectif est de maintenir ou d'améliorer les services de communication. La démarche adoptée part de l'identification de l'aléa qui affecte le système de communication. Par la suite, une ou plusieurs solutions sont envisagées pour traiter les conséquences ou encore mieux la cause générant l'aléa. Les travaux réalisés traitent de trois aléas.

5.1.1 L'aléa de la panne

Dans ce domaine, la contribution n'a pas porté sur de nouveaux mécanismes pour assurer la survivance du réseau. Notre préoccupation a été d'identifier les traitements critiques pour respecter la contrainte de délai de reprise après panne. Nous avons étudié le fonctionnement du protocole de protection de SONET. Ce protocole a été modélisé par un automate d'états et traduit en un modèle de simulation. Les différents cas de pannes ont été évalués afin de découvrir le cas le plus défavorable. Enfin les cas de pannes simultanées ont été étudiés. Cette configuration est celle qui se produit lors des catastrophes naturelles ou par cause humaine.

5.1.2 L'aléa d'accès

Il concerne les réseaux locaux sur support à diffusion. Nous souhaitons, dans le contexte de la méthode d'accès CSMA/CD, différencier les accès à la transmission. Nous avons proposé un contrôle de trafic au niveau IP qui empêche un effondrement des performances, réserve des ressources pour un service de communication de débit prévisible et effectue un partage des ressources réservées non utilisées selon un critère d'équité Max-min. Cette récupération des ressources offre à la solution un taux d'utilisation des ressources important. Cette proposition a été développée et évaluée à l'aide de l'outil ns-2. Les réseaux évoluent continuellement. Comme le fait justement remarquer Crowcroft [92], un sujet est d'actualité mais trois ans plus tard, il est devenu périmé. L'usage d'Ethernet est passé d'un mode partagé à un mode dédié avec le développement des commutateurs et des réseaux virtuels (VLAN). L'approche proposée n'avait plus la même pertinence pour Ethernet, mais par contre, elle a encore un sens pour les réseaux sans fil. L'avantage de notre proposition est qu'elle ne demande pas de modification de la couche de transmission et évite la saturation du support. Des modifications au niveau de l'allocation de ressources pour le service à qualité prévisible ont été étudiées et évaluées. La détermination de la ressource disponible à partager entre les mobiles d'un réseau sans fil est une opération très complexe voire impossible. Une façon naturelle de traiter la

complexité est l'auto-organisation [59]. Dans un système auto-organisé, les composants atteignent une fonction globale en suivant des règles locales simples. Nous avons proposé d'effectuer le partage des ressources en fonction d'une mesure locale. Les mobiles sont ainsi autonomes.

5.1.3 L'aléa du trafic

Au sein du projet @IRS, nous avons considéré une nouvelle architecture d'Internet à qualité de service de type Diffserv. Nous avons développé et évalué les performances de ces services de communication. Ce travail d'évaluation a été fait à l'aide d'une plateforme d'expérimentation nationale. Le conditionnement des flux en bordure du réseau a montré qu'il était inadapté pour les flots élastiques tels que ceux générés par TCP.

Notre contribution a porté sur un conditionneur qui sanctionne par une augmentation du RTT les flots surpassant leur débit assuré. Cette action vise à faire diminuer le débit d'émission des flots "bien servis" pour que les flots "mal servis" puissent atteindre la qualité attendue. L'avantage comparatif de notre conditionneur porte sur l'absence de mesure à réaliser sur le flot TCP. Il peut même s'utiliser sur un groupe de flots TCP. Cette solution offre un certain caractère d'insensibilité au nombre (*scalability*) de flots pouvant être conditionnés.

Dans [93], les auteurs décrivent les éléments nécessaires pour fournir de la qualité de service sur Internet. Ils notent que les classes de service ne fonctionnent que si elles ne représentent qu'une petite partie du trafic et ses effets ne sont même pas perçus par l'utilisateur final. Une qualité de service explicite est une démarche complexe impliquant beaucoup d'acteurs. Le public n'est pas prêt à payer une technologie si aucun bénéfice n'en est tiré [94]. L'auteur pense qu'il n'y aura jamais de qualité de service explicite dans l'Internet. Il est préférable d'augmenter les capacités que d'ajouter du contrôle. Pour conclure, nous avons fait des contributions intéressantes dans le domaine de la qualité de service mais ce domaine n'est plus pertinent. D'autres solutions de QoS que nous verrons dans le paragraphe 5.2 ont détrôné la qualité de service rendue explicitement.

Dans le contexte de l'Internet classique, le débit écoulé par une connexion TCP dépend du taux d'événements de congestion. La détection de la congestion est un élément important dans la performance d'une connexion TCP. Celle-ci est portée par TCP ou par le réseau. Nous voulons un signal clair de congestion comme le fait le réseau tout en restant en bordure du réseau. L'évolution architecturale proposée porte sur le placement de la fonction de détection de congestion. Nous montrons que la congestion peut être détectée en bordure de réseau et donc notifiée à TCP. L'algorithme développé arrive à distinguer la congestion des événements non issus de la congestion comme les déséquencements.

Certains flots TCP supportent moins bien que d'autres la congestion. C'est notamment le cas des flots courts. Nous proposons FavourQeue, un AQM visant à diminuer le taux de pertes pour ces flots. L'identification d'un flot court est fait en fonction de l'occupation de la file d'attente. Les paquets des flots courts sont protégés de la perte. L'évaluation a été faite à l'aide de l'outil ns-2 sur un modèle de trafic représentant le web. L'amélioration de la performance constatée provient d'une diminution des reprises par RTO et d'une diminution générale du taux de pertes (tous flots confondus). Il est intéressant de noter que la diminution de la sporadicité en phase de *slow-start* est le principal facteur de la diminution du taux de pertes.

Dans de nombreux travaux présentés, l'évaluation repose sur un modèle de simulation décrit pour l'outil ns-2. Au niveau de la méthode, l'usage répété de cet outil m'a conduit au développement d'une interface de modélisation [95]. La simulation donne toujours des résultats. La validité de ces résultats est toujours une question. Lorsque l'on

effectue une étude, les conditions appliquées au modèle changent. Concrètement, le code et les paramètres changent. Ma motivation a été d'avoir un outil pour construire des modèles de la manière la plus fiable possible. Mon idée a été de factoriser le code commun de tous les modèles par l'interface de modélisation. La description d'un modèle s'effectue dorénavant par des paramètres exprimés sous forme d'une paire (type valeur). L'intérêt est d'avoir une bonne fiabilité dans les modèles (pas ou peu de codes) tout en offrant une flexibilité pour adopter un modèle aux différentes études. Cet outil sert aussi pour les activités d'enseignement car il ne demande pas la connaissance de ns-2 ou de Tcl. Il est disponible librement au téléchargement [1].

Au cours de ces années, j'ai abordé des problématiques de service sur les principales classes de réseau, à savoir les réseaux locaux en mode diffusion et les réseaux commutés longues distances. Mon activité de recherche a évolué des réseaux à qualité de service vers l'Internet classique à service *Best effort*.

Du fait de ma localisation et des contraintes liées à l'éloignement des grands centres, j'ai eu le souci d'orienter ma recherche vers des niches à forte valeur régionale. Toute la difficulté réside à ne pas retenir un *cold topic*. Cependant, la Réunion offre des opportunités qu'il nous appartient de saisir. Nous les détaillons dans le paragraphe 5.3.

5.2 Evolution générale de l'Internet

Le futur de l'Internet est déjà présent, tout du moins si on arrive à faire le tri entre les idées à la mode et les idées de fond. Les tendances lourdes sont déjà identifiables. Elles concernent en fait le consommateur de service et le producteur de service. Le réseau n'est qu'un intermédiaire entre ces deux éléments dont l'évolution est guidée par l'objectif de faire rencontrer ces deux éléments. Le consommateur de service en dernier ressort est un utilisateur humain qui interagit avec un terminal au sens large. Le paragraphe 5.2.1 présente l'évolution prise par les terminaux. Tandis que le producteur de service classiquement centré sur le serveur est confronté au facteur d'échelle induit par la croissance continue de la taille de l'Internet. De nombreuses pistes ont été ouvertes pour traiter cette problématique liée à la taille qui se traduit en terme d'accessibilité au service. Les principales pistes sont présentées dans le paragraphe 5.2.2.

5.2.1 Terminaux mobiles

Une tendance forte d'évolution datant d'une quinzaine d'années porte sur les terminaux. En 1996, Christian Huitema avait fait un exposé à l'Ecole d'été de Cauteret. Dès cette époque, il avait indiqué que le futur de l'Internet serait constitué par des terminaux mobiles (des téléphones) et des composants électriques aussi simples qu'une lampe. On ne peut pas dire qu'il a eu tort. Il avait anticipé l'émergence de nouveaux terminaux Internet mobiles couramment appelés *smartphone* et la connectivité de n'importe quel type d'objet pour former de nos jours ce que l'on dénomme l'Internet des objets (*Internet of Things*).

Le développement des capacités de traitement des téléphones n'a cessé d'augmenter pour en faire de véritables petits ordinateurs. Ceci est tellement vrai que les cartes ont été redistribuées entre les constructeurs de téléphones. L'industrie est maintenant tenue par les grands de l'informatique tels que Microsoft, Google, Apple. Ces "super" téléphones (ou *smartphones*) sont remplis de capteurs. Mais ils ont aussi la capacité de communiquer avec des capteurs externes tout en ayant une connectivité Internet. Ils deviennent un

1. http ://personnel.univ-reunion.fr/panelli/2_research/modint/

médiateur entre l'utilisateur et un monde numérique. A titre d'exemple, l'expérimentation BikeNet [96] montre comment un *smartphone* peut être utilisé pour l'enregistrement des paramètres des déplacements en vélo. Les capteurs du vélo comme le compteur de vitesse sont interfacés avec le téléphone. De plus, BikeNet utilise un mode de communication opportuniste. Ce concept est apparu récemment [97]. Il qualifie un mode de communication où l'information passe de mobile en mobile au gré des rencontres. Plus généralement dans [98], les auteurs indiquent que les *smartphones* équipés de leurs capteurs vont révolutionner beaucoup de secteurs de notre économie. Ils ouvrent la porte à un nouveau domaine de recherche que l'on peut appeler la perception par téléphone portable (*mobile phone sensing*). Ce domaine est transversal et s'appuie aussi bien sur la communication sans fil de capteurs, que sur la fouille de données (*data mining*).

La miniaturisation croissante de l'électronique, la faible complexité de IP et le contrôle de l'énergie ont étendu l'idée des terminaux à Internet aux objets mobiles. C'est l'Internet des objets dans lequel les petits systèmes numériques ne sont plus déconnectés du monde virtuel mais sont contrôlés à distance et peuvent remonter des informations à des services Internet [99]. Dans [100], le futur de l'internet est vu au travers de petits objets qui communiquent de manière opportuniste en vue de réaliser des fonctions plus riches en utilisant les ressources de leur environnement. Les auteurs introduisent la notion de traitement opportuniste (*opportunistic computing*). Cependant, la question centrale reste comment atteindre une interopérabilité entre des objets hétérogènes et activer leurs capacités d'adaptation et d'autonomie [101].

Cette évolution des terminaux a conduit à leur multiplication. Leur nombre va produire une masse d'informations sans précédent faisant alors peser une formidable contrainte sur le réseau et sur les serveurs pour respectivement acheminer et traiter ces données.

5.2.2 Architecture des services

Le service rendu à l'aide d'un simple serveur est un modèle qui a vécu. Le défi à relever pour le service a trait aux nombres des utilisateurs et donc à son accessibilité. Cela doit être réalisé en prenant en compte une contrainte environnementale liée à la consommation énergétique. En effet, les préoccupations environnementales ont fait leur apparition ces dernières années dans l'industrie des TIC. La consommation électrique de l'Internet est estimée dans une fourchette de 83 à 143 GW [2] [102]. Sachant que l'électricité est produite par des énergies fossiles à 64% environ, le monde virtuel laisse une empreinte écologique dans le monde réel.

L'évolution de l'architecture des services a été importante. Nous avons assisté à l'émergence de deux phénomènes extrêmes ces dix dernières années.

En premier, la combinaison des systèmes distribués avec les fermes de serveurs a offert une nouvelle élasticité dans la ressource informatique. Le service ne repose plus sur un ou des serveurs dédiés mais sur un ensemble de ressources allouées à la demande. Les auteurs de [103] indiquent que la notion d'informatique en nuage *cloud computing* est constituée par un centre informatique et des logiciels. A titre d'exemple, les Google Apps illustrent bien cette idée d'informatique en nuage. Au contraire de ce que fait Google, un nuage peut aussi se former à l'aide d'un centre informatique dont les composants sont répartis sur l'Internet, c'est la thèse développée dans [104]. Cette proposition nommée NaDa (*Nano Data center*) offre un gain en coût énergétique.

En second, c'est à la notion même de serveur qui a disparu. Le paradigme du pair à pair (P2P) a introduit une nouvelle forme d'interaction entre le client et le service dans

2. soit entre 3.6 à 6.2% de la production électrique de l'humanité

laquelle le client est lui-même le fournisseur de service pour les autres [105]. Ce paradigme revient à constituer un réseau superposé de niveau applicatif dans lequel les règles d'adressage du contenu et de routage sont propres à ce niveau. Le P2P donne une vision de l'Internet dans laquelle le service ne repose que sur les moyens de ces utilisateurs.

Entre des centres informatiques dédiés au service et l'absence de serveur, il existe une vision intermédiaire née pour apporter une solution au problème de la qualité du service. Pour le service de consultation, c'est à dire l'accès à des informations, il existe un problème de qualité pour la restitution des données. Les ressources partagées que sont le réseau et le serveur ont un risque de saturation et donc de dégradation du service. La solution proposée est de séparer le producteur d'information de l'activité de diffusion. Le contenu produit est stocké sur des caches répartis sur l'Internet. L'avantage est double : supprimer le serveur central et raccourcir le chemin entre le contenu et le client. Plus ce chemin est court et moins le trafic a de risque de rencontrer des aléas. A la limite, si le contenu souhaité est disponible sur l'accès Internet du client, l'aléa de trafic a disparu. Cette approche offre une qualité de service implicite. A savoir elle est fournie avec le service et il n'y a pas de tarification pour l'avoir [93]. Cette idée est née dans le milieu des années 90 et à donner naissance à un nouvel intermédiaire : le réseau à distribution de contenu (CDN : *Content Delivery Network*) [106]. De nos jours, la qualité du service de consultation repose sur ce type de réseau auquel les grands producteurs de contenu ont recours.

On voit bien se dessiner une direction de l'Internet vers plus d'abstraction des services par rapport aux aspects matériels. Les contenus, les logiciels ne sont plus sur un serveur, ils deviennent diffus sur le réseau. Van Jacobson l'a bien compris et propose de changer l'abstraction réseau sur laquelle repose IP par une abstraction portant sur le contenu [107]. Cette nouvelle notion est connue sous le terme de *Content-Centric Networking* (CCN). L'ambition est de proposer une nouvelle architecture de réseau. Les auteurs de [108] montrent que cette architecture est efficace énergétiquement dans la distribution de contenu.

Face à ces évolutions des services sur le long terme, à plus court terme, la technologie et les moyens changent également. En premier, le défi du déploiement d'IPv6 est lancé depuis le début de l'année. Il n'y a plus de bloc d'adresses disponible depuis février 2011 [3]. Vu la taille et la structure de l'Internet, le déploiement de nouvelles fonctionnalités dans le réseau est devenu une tâche insurmontable. Cependant, nous le voyons bien, le réseau doit pouvoir évoluer sans perdre la valeur donnée par les services actuels. Comment déployer IPv6, CCN ou d'autres architectures ? Dans [109], les auteurs promeuvent l'usage de la virtualisation comme moyen de déploiement et de test pour les nouvelles architectures. Cette technique est utilisée dans la plateforme Planetlab. Elle repose sur le concept d'un noeud comportant plusieurs machines virtuelles. Les auteurs de [110] expliquent comment ce concept peut servir de nos jours à constituer, dans un environnement commercial, des réseaux virtuels (VN : *virtual network*) pour fournir un certain service. Ceci montre que la virtualisation de réseau n'est pas cantonnée à l'évolution de l'Internet, elle peut devenir un autre moyen de partage des ressources et donc de diminution des coûts [111]. Les auteurs de [112] montrent que la virtualisation a encore quelques défis à relever comme la gestion des ressources, la performance pour son acceptation générale. Il ne reste pas moins que c'est un moyen ouvrant la voie à une évolution de l'Internet.

5.2.3 En conclusion

Se pose la question : allons nous vers plusieurs Internet ? Un Internet des mobiles et des choses, un Internet du contenu et un Internet classique en IPv6. La division en

3. `http://www.potaroo.net/tools/ipv4/`

plusieurs réseaux serait une perte de valeur. En fait, l'architecture de l'Internet doit évoluer pour prendre en compte les propriétés d'autonomie et d'opportunité des réseaux constitués par les nouveaux terminaux. Mais elle doit aussi prendre en compte l'ubiquité pouvant être donnée aux contenus. La virtualisation s'annonce comme un moyen d'un déploiement rapide de ces évolutions sur l'infrastructure existante. Enfin, on notera à quel point l'accessibilité au service est un moteur puissant d'évolution tant au niveau des serveurs que du réseau.

5.3 Mes perspectives de recherches

Mes perspectives de recherches s'inscrivent dans la politique régionale de développement ainsi que dans celle de la recherche de mon établissement. La région Réunion a l'ambition d'autonomie énergétique pour 2025. Cela nécessite des nouveaux gisements de production d'énergie mais également une meilleure efficacité énergétique. Les TIC peuvent concourir à cet objectif en rendant les équipements consommateurs intelligents. Par exemple, des bâtiments intelligents contrôleraient la consommation de l'énergie en fonction de l'anticipation ou de la perception des besoins (des humains).

Mes perspectives doivent aussi prendre en compte les contraintes et les caractéristiques de la Réunion dans son accès à l'Internet. Notre accès a la particularité d'avoir une connectivité unique passant par le câble sous-marin Safe (avec une liaison satellite de secours). Cette caractéristique n'existe pas pour les régions continentales où le maillage est la règle. Au niveau de la population, celle-ci accède à 61% à Internet dont la moitié au moyen d'un accès haut débit. Pour être complet sur les statistiques de la population, on peut noter que celle-ci est densifiée sur une zone réduite du littorale et de mi-hauteur. Enfin la région est dotée d'un réseau fédérateur haut débit (réseau Gazelle).

Plus aujourd'hui qu'hier, le choix des activités de recherches doit prendre en compte les compétences au sein de l'université. Comme le fait si bien remarquer Jennifer Rexford, notre champ d'action en réseau est inter-disciplinaire [113]. Cela veut dire que la résolution de nos problèmes se fait en empruntant d'autres disciplines. Autrement dit l'application d'autres disciplines aux problèmes des réseaux se fait en collaborant avec des spécialistes. L'évolution de l'Internet présentée au paragraphe précédent nous ouvre de nouveaux champs d'études. Les critères pour retenir des problèmes de recherches intéressants sont définis par Craig Partridge [114]. Il y cite l'attention suscitée par les autres chercheurs, l'ouverture de perspectives, la probabilité de réussite.

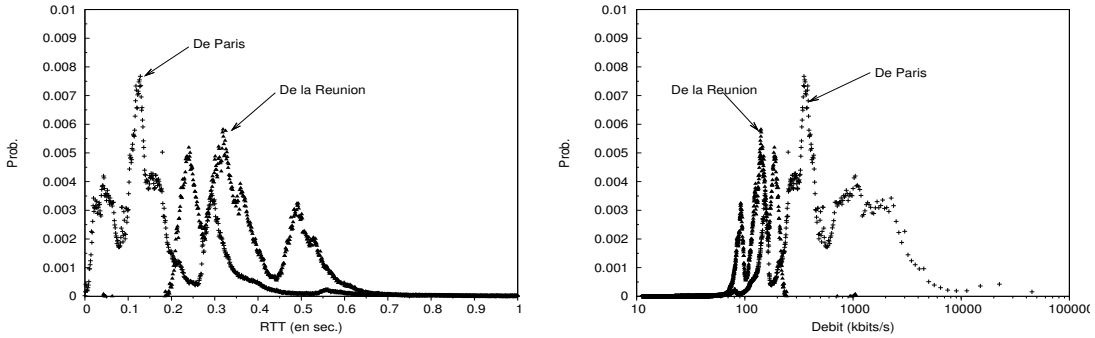
En fait, mes perpectives se répartissent sur deux types de problèmes. Le premier traite de l'accès de la Réunion à Internet. Le second, plus ambitieux et prospectif, porte sur le développement de nouveaux services sur des usages locaux.

5.3.1 Accès à Internet

La figure 5.1(a) présente les résultats des mesures du RTT faites par une campagne de `ping` vers des adresses IP tirées aléatoirement. Par rapport à Paris [4], on remarque une forte répartition après les $200ms$ (dont $150ms$ sont pris par la propagation du signal). Le trafic Internet est majoritairement constitué par des flots TCP. Le débit écoulé par TCP est inversement proportionnel au RTT [115]. A titre d'illustration, la figure 5.1(b) présente la distribution du débit écoulé moyen avec comme hypothèse un taux de pertes de 10% et une taille des segments de 1460 octets. Ces graphiques montrent qu'un hôte à

4. D'après les mesures faites par M. Latapy :
`http://www.complexnetworks.fr/rtt-distribution/`

la Réunion a un débit écoulé plus faible qu'un hôte à Paris. De plus, la connectivité de la Réunion est plutôt pauvre en bande passante [5]. L'amélioration de la qualité de notre accès à Internet ne passe pas uniquement par une augmentation de la bande passante. La figure 5.1(b) montre qu'il y a une limitation protocolaire. Différentes solutions sont à explorer en fonction du type de communication. Une contrainte forte dans le choix des solutions porte sur la progressivité du déploiement.



(a) Probabilité de RTT. (b) Probabilité de débit.

FIGURE 5.1 – Comparaison entre un accès parisien et réunionnais.

Etude métrologique

La première chose à faire pour envisager des solutions est un état des lieux du trafic montant et descendant à la Réunion. Cette étape est importante, elle vise à mettre en lumière les principaux freins aux débits écoulés. Cette étude sur le trafic vise à identifier la corrélation des événements de congestion entre les flots, l'importance des anomalies affectant TCP (pertes initiales, faux événements de congestion), la répartition des flots (en taille, en protocole, en application) et les paramètres de performances (taux de pertes, RTT, débit écoulé). Les méthodes d'analyse de trafic font l'objet d'une importante activité de la communauté. Dans [116], les auteurs présentent les principales techniques pour réaliser une étude de trafic. Cette activité est inter-disciplinaire et peut faire l'objet d'une collaboration locale dans le domaine du *data mining*.

Service consultation de contenu

Si le trafic descendant est principalement généré pour des applications de consultation de contenu, alors le développement d'une solution à base d'un cache permettrait d'augmenter la qualité du service tout en améliorant la qualité des autres services du fait de la diminution du trafic de consultation sur le câble sous-marin. Plus généralement, le problème du placement local du contenu est encore à résoudre. Cette question ne nous est pas propre et se décline en un vrai sujet de recherche [114]. L'idée sous-jacente à toutes solutions est de favoriser les échanges locaux au détriment des échanges lointains. Une fois un contenu lointain sur l'ile, il faut alors le partager pour ne plus utiliser le câble sous-marin. Une solution pourrait s'appuyer sur les équipements d'accès Internet locaux dont je présente ci-dessous le principe.

Le nombre de logements à la Réunion est estimé en 2008 à 280000 (Source INSEE). Les accès résidentiels par DSL peuvent être estimés à 100000. D'un point de vue énergétique,

5. http://www.domtom-adsl.com/news/internet-reunion---du-cache-pour-augmenter-la-bande-pas
html

on peut noter que le passage d'un accès bas débit à un accès haut débit accroit le nombre de consommateurs électriques. En effet, la "box" est alimentée électriquement à son point de branchement. Avec une capacité de 100000 "box", dans quelle mesure un cache local reposant sur les principes de Nada ne pourrait pas être développé ? En effet, les "box" sont connectées localement par un réseau haut débit (le très haut débit est à l'étude par la région). L'ajout d'un dispositif de stockage à une "box" n'aurait qu'une augmentation marginale de la consommation électrique. Cette solution est plus efficace énergétiquement que la construction d'un centre informatique (data center) et présente une propriété d'insensibilité au facteur d'échelle. Les "box" avec leur dispositif de stockage formeraient des noeuds d'un système de cache. Ce système fonctionnerait en interne selon le principe de CCN. La densité de la population de la Réunion est un autre facteur motivant pour une telle solution. Certes elle est plus faible qu'une agglomération, mais elle reste forte pour une région. Ce critère de densité de population est important pour évaluer la distance moyenne entre les noeuds et donc les capacités de transfert.
Cette solution est porteuse de nombreux défis comme l'alimentation du cache en contenu, la gestion des contenus, l'interaction avec l'internet classique des clients.

Service conversationnel

Si le trafic comporte une part importante de communications inter-personnelles issues des applications de conversation ou messagerie, dans cette situation, l'interaction se fait directement avec la destination. Du point de vue de la mise en oeuvre, il faut viser des améliorations de l'existant à déploiement progressif ou partiel. Sachant qu'ici, les aléas de la communication sont plus qu'ailleurs très coûteux en terme de performance, citons notamment :

- les fausses congestions (déduites des fausses pertes). Elles diminuent le débit écoulé à tort.
- les vraies pertes au démarrage d'une connexion. Elles détériorent l'interactivité significativement.
- les pertes dues à la congestion. Elles entraînent des réductions importantes du débit écoulé. Dans la mesure où la fenêtre doit couvrir un produit élevé de délai-bande passante, la réduction de fenêtre peut représenter une grande quantité.

Les réponses sont à trouver principalement au niveau du contrôle de congestion. J'identifie 4 pistes qui apporteraient une amélioration du service par rapport à la situation actuelle.

Piste 1 : détection fiable de la congestion

Pour se prémunir contre les faux événements de congestion, la notification de la congestion (ECN : *Explicit Congestion Notification*) par le réseau a été démontrée comme une bonne solution. Mais le déploiement de cette solution est très long. La notification de congestion par la bordure peut offrir une alternative. La détection de la congestion peut se faire par combinaisons de méthodes. Notre proposition TCED peut encore s'enrichir par d'autres méthodes. Notamment si les analyses de trafic montrent une corrélation entre les événements de congestion de plusieurs flots. L'idée du gestionnaire de congestion [68] pourrait devenir pertinente.

Piste 2 : démarrage de connexion

Le problème de démarrage des connexions est toujours d'actualité [79]. Certains proposent des modifications protocolaires de la phase d'établissement de connexion. Citons [117] où les auteurs suggèrent la solution *TCP Fast Open* consistant à commencer le transfert de données avec les échanges de segment SYN. Plus surprenant, les auteurs de [118] combinent la résolution de noms avec l'établissement de connexion. Ces solutions ne

sont pas progressives, ou tout du moins elles demandent que les deux entités de transport soient compatibles.

Nous avons commencé une étude sur la protection de la phase d'établissement de connexion TCP. Cette première étude a porté sur l'évaluation du gain apporté par la protection des segments SYN. Les auteurs de [119] ont relevé sur les traces analysées un taux de pertes de SYN qui se monte à 0.5%. Les travaux sur la favorisation nous a appris que la diminution de la sporadicité du *slow-start* s'est traduite par une baisse du taux de pertes. Bien que le lissage de flot TCP ait été démontré comme une mauvaise solution [120], les auteurs notent que lors de la phase initiale, la version de TCP avec lissage de flot a atteint de meilleures performances. La piste d'un lissage du flot à la source mérite d'être approfondie. Dans le cas d'une augmentation de la fenêtre initiale à $10MSS$ [85], la question de la pertinence du lissage reste à étudier [121]. Dans le contexte de l'extension *Multipath TCP* [122], une taille dynamique de fenêtre initiale pour les nouveaux sous flots en fonction de la performance des sous-flots actifs pourrait avoir un impact positif sur le taux de pertes. Bien que nous étudions le problème de démarrage de connexion dans le contexte des flots courts de TCP, il n'est pas à douter qu'il touche les autres protocoles de transport.

Piste 3 : Favorisation des flots à faible taux de pertes

Les pertes, on vient de le voir, sont préjudiciables pour les flots TCP courts. En fait, elles le sont aussi pour les flots longs dans la mesure où elles entraînent de manière générale une diminution du débit écoulé. TCP pourrait être plus performant en évitant de charger le canal (*pipe*) au delà de sa capacité. Les versions de TCP ayant un contrôle de congestion *delay-based* visent à anticiper la perte et donc à diminuer le taux. Ce type de version de TCP serait une bonne solution. Malheureusement, les flots avec ce contrôle de congestion souffrent d'une faible performance en présence de flots avec un contrôle de congestion *drop-based*. La favorisation pourrait offrir une perspective intéressante pour faire coexister ces deux types de contrôle de congestion. La favorisation mérite encore d'être approfondie dans cet axe là. Si la congestion se manifeste principalement sur notre liaison Internet, le déploiement d'un AQM sur quelques routeurs peut être suffisant pour améliorer la performance des flots TCP.

Piste 4 : Changement de paradigme du contrôle de congestion

Un autre aspect peut augmenter la performance des flots TCP pour la Réunion, c'est la suppression de la dépendance au RTT. Un contrôle de congestion *rate-based*, tel que proposé par [123] offre cette caractéristique. Dans cette proposition, les auteurs proposent de jouer sur les délais inter-paquets pour déterminer le débit d'émission. Malheureusement, un contrôle de congestion *rate-based* reste difficile à utiliser dans la mesure où il demande le concours du réseau et/ou du destinataire. Dans quelle mesure, une telle approche pourrait s'utiliser avec la version TCP à mesure du débit comme TCP Westwood [124] ?

Enfin des chercheurs ont montré que l'effondrement dû à la congestion (*congestion collapse*) n'existerait plus aujourd'hui dans les réseaux sans contrôle de congestion si le débit des sources est inférieur d'un ou deux ordres de magnitude de la capacité des liens [125]. Ce constat ouvre un nouveau domaine de recherche, à savoir si un Internet sans contrôle de congestion peut se réaliser. Cette idée est intéressante car elle supprime de fait la dépendance au RTT. Cependant elle demande un AQM à pertes équitables (*fair drop*). Dans quelle mesure Choke [126] ou FavourQeue [90] pourrait s'utiliser comme un tel AQM ? Une étude sur l'impact des AQM éclairait sur la faisabilité de ces réseaux dit anarchiques.

5.3.2 Nouveaux services

Ma seconde perspective de recherche plus prospective veut s'inscrire dans la politique de recherche de l'université de la Réunion. L'université a identifié plusieurs axes dans lesquels elle souhaite fédérer ses chercheurs afin de donner une plus grande visibilité à sa recherche. L'un de ces axes porte sur la télésurveillance. Ce thème se veut pluri-disciplinaire afin de faire collaborer les chercheurs. Au niveau du laboratoire et du département d'informatique, nous avons la volonté de développer l'informatique à base de mobiles et *smartphones*. C'est dans ce contexte que je présente ici les premières réflexions menées avec des collègues.

Les demandes locales pour des services ont trait à la gestion énergétique, aux activités de déplacement, à la vie sociale et au tourisme. Ce sont des applications à caractère ubiquitaire et fortement génératrice de données. La vision du projet est de mettre l'humain et ses activités comme principal producteur de données. La structure classique de ces applications est équivalente à celle présentée par BikeNet [96] : une collecte des données par un réseau de capteurs installé sur le vélo qui les communique à un nuage informatique. Le *smartphone* prend dans cette organisation un rôle de médiateur entre les capteurs et le nuage informatique. Les données reçues par le nuage informatique sont analysées et traitées par une application afin de fournir des informations d'aide à la décision ou pour modifier l'environnement.

Mon intuition porte sur la quantité de données. Les applications ubiquitaires vont être un facteur de génération de données. Chaque individu va devenir producteur d'une multitude de données. De plus si le monde réel se remplit d'objets numériques communiquant aussi divers que l'interrupteur électrique, la voiture ou des appareils électroménagers. Tout ceci va faire croître le nombre de terminaux et donc le nombre de producteurs de données. On peut craindre que la quantité de données produite soit phénoménale. Cette idée de faire remonter les données au nuage informatique pose un problème dans ce contexte. En effet, on peut se demander si le réseau va tenir la charge : est ce que cette approche va pouvoir passer le facteur d'échelle d'un point de vue de la communication ? Par conséquent, les projets que je compte développer auront comme ligne directrice de limiter les remontées de données vers le nuage informatique et de favoriser les échanges locaux.

La première application envisagée se place à l'échelle d'un bâtiment, voir d'un campus universitaire équipé d'un ensemble d'objets et capteurs. Ce bâtiment aurait la capacité de suivre en temps réel les différentes données :

- générées par les capteurs (température, luminosité), ou
- communiquées par les *smartphones* des utilisateurs.

Avec ces données, le bâtiment doté d'intelligence aurait les moyens de surveiller son utilisation efficace et d'en extraire des informations afin d'agir sur certains éléments comme la climatisation ou les ouvrants en fonction de la stratégie de la consommation énergétique. Nous souhaitons, dans un premier temps, nous limiter à la collecte des données, à leur communication et dans leur interprétation afin de pouvoir appliquer une gestion de la consommation énergétique des bâtiments. Ces travaux fournissent déjà un cadre applicatif ambitieux et répondent à la volonté actuelle de la maîtrise des énergies.

La seconde application porte sur le déplacement des personnes. A la Réunion, cette activité est articulée autour de la voiture individuelle. La solution actuelle n'est pas durable. Une alternative au tout automobile consisterait à favoriser à l'intermodalité des transports entre le bus, le car ou le co-voiturage. L'information est une composante essentielle pour le choix des modes de déplacement. L'idée que nous souhaiterions déve-

lopper dans ce projet porte sur l'équipement des bus comme agents de surveillance. Ils collecteraient des informations tel que l'état du trafic, l'état de la route ou la météo. Ces informations seraient ensuite échangés avec les autres cars et les arrêts de bus pour être ensuite diffusées auprès des utilisateurs. La question porte sur l'intérêt de la communication opportuniste et de la pertinence des informations échangées.

5.3.3 Conclusions

A l'image des travaux présentés dans ce mémoire, mes perspectives se déclinent à la fois sur les communications distantes et locales. Pour les communications distantes, mon activité "accès Internet" vise à proposer une alternative crédible au toujours plus de bande passante. L'articulation réside dans un traitement approprié du trafic dans les routeurs et dans la couche de transport. Pour les communications locales, ma perspective porte sur le développement de nouveaux services. Cette activité est encore à un stade très prospectif mais elle offre un fort potentiel de fédération au sein de l'établissement. Cette caractéristique est essentielle dans un contexte insulaire qui peut devenir une cause d'isolement.

Bibliographie

[1] J.C. McDonald. Public network integrity – avoiding a crisis in trust. *IEEE Journal on Selected Areas in Communications*, 12(1) :5–12, January 1994.

[2] P. Cholda, A. Mykkeltveit, B.E. Helvik, O. Wittmer, and A. Jajszcyk. A survey of resilience differenciation frameworks in communicationnetworks. *IEEE Communications Surveys & Tutorials*, 9(4), Fourth quarter 2007.

[3] American Standard for Telecommunications. Telecommunications - synchronous optical network (SONET) - automatic protection switching, 1995.

[4] UIT-T. Types and characteristics of SDH network protection architectures, July 1995.

[5] A. Haider and R. Harris. Recovery techniques in next generation networks. *IEEE Communications Surveys & Tutorials*, 9(3), Third quarter 2007.

[6] P. Anelli and M. Soto. Evaluation of the APS protocol for SDH rings reconfiguration. *IEEE Transactions on Communications*, 47(9) :1386–1393, September 1999.

[7] G. Suwala and G. Swallow. SONET/SDH - like resilience for IP networks : A survey of traffic protection mechanisms. *IEEE Network*, March 2004.

[8] S. Poretsky, R. Papneja, J. Karthik, and S. Vapiwala. Benchmarking Terminology for Protection Performance. RFC 6414 (Informational), November 2011.

[9] T. Wang and H. Dai. Network system survivability survey : An evolution approach. In *2nd International Conference on Future Computer and Communication (ICFCC)*. IEEE, May 2010.

[10] IEEE 802.17. Resilient Packet Ring, 2004.

[11] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun. IEEE 802.17 resilient packet ring tutorial, 2003.

[12] J.P.G. Sterbenz, D. Hutchison, E. Çetinkaya, A. Jabbar, J.P. Rohrer, M. Schöller, and P. Smith. Resilience and survivability in communication networks : Strategies, principles, and survey of disciplines. *Elsevier Computer Networks*, 54(8) :1245–1265, June 2010.

[13] Horlait E. Procédé de gestion de bandes allouées dans les réseaux locaux à accès partagés, protocole et filtre de mise en oeuvre. Déposé pour la France, l'Europe et l'Amérique du Nord et le Japon, Juin 1997.

[14] IEEE Standard 802.1. IEEE standards for local and metropolitan area networks : Virtual bridged local area networks. *IEEE 802.1q Std.*, 1998.

[15] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, and M. Speer. SBM (Subnet Bandwidth Manager) : A Protocol for RSVP-based Admission Control over IEEE 802-style networks. RFC 2814 (Proposed Standard), May 2000.

[16] M. Hassan, S. Jha, and A. Baig. Performance evaluation of subnet bandwidth manager for supporting qos over shared leagcy lans. In *International Conference on Telecommunications (ICT)*, pages 496–499, Bucharest, 2001. IEEE.

[17] P. Anelli and G. Le Grand. Differentiated services over shared media. In *International Workshop on Quality of Service (IwQoS'01)*, page 6, Karlsruhe, Germany, June 2001.

[18] G. Le Grand and P. Anelli. Service à débit différencié sur réseau à médium partagé. In *Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel)*, Meze, France, Mai 2002.

[19] M. Heusse, P. Starzetz, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Bandwidth allocation for diffserv based quality of service over 802.11. In *IEEE Global Telecommunications Conference (Globecom)*, San Francisco, California, December 2003.

[20] IEEE Standard 802.11. Wireless lan medium access control (MAC) and physical layer (PHY) specifications : Medium access control (MAC) quality of service (QoS) enhancements. *IEEE 802.11e Std.*, 2005.

[21] G. Le Grand, R. Meraihi, S. Tohmé, and M. Riguidel. Intelligent ad hoc networking to support real time services. In *Vehicular Technology Conference (VTC)*, Orlando, Florida, October 2003. IEEE.

[22] B. Wolfinger, W. J., and G. Le Grand. Improving node behavior in a QoS control environment for local broadcast networks. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, Montreal, Canada, July 2003. SCS.

[23] F. Harivelo, G. Le Grand, P. Anelli, J. Wolf, and B.E. Wolfinger. Expedited forwarding for wifi. In *1st International Symposium on Wireless Communication Systems (ISWCS)*, Port Louis (Mauritius), September 2004. IEEE.

[24] F. Harivelo and P. Anelli. Différenciation de services sur WLAN. In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'05)*, page 8, Bordeaux, France, Mars 2005.

[25] F. Harivelo and P. Anelli. A robust service for delay sensitive applications on a WLAN. In *4th International Conference on Networking (ICN'05)*, pages 84–91, pp.8, La Réunion, France, April 2005. IEEE, IARIA.

[26] Q. Ni, L. Romdhani, and T. Turletti. A survey of QoS enhancements for IEEE 802.11 wireless lan. *Wiley Journal of Wireless Communications and Mobile Computing*, 4(5) :547–566, 2004.

[27] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran. A survey of quality of service in IEEE 802.11 networks. *IEEE Wireless Communications*, 11(4), August 2004.

[28] A. Nyandoro, M. Hassan, and L. Libman. Service differentiation using the capture effect in 802.11 wireless LANs. *IEEE Transactions on Wireless Communications*, pages 2961–2971, 2007.

[29] J.A. Garcia-Macias, F. Rousseau, G. Berger-Sabbatel, L. Toumi, and A. Duda. Quality of service and mobility for the wireless Internet. *Kluwer Wireless Networks*, 9(4) :341–352, July 2003.

[30] Wireless Gigabit Alliance. Defining the future of multi-gigabit wireless communications. White paper, July 2010. http ://wirelessgigabitalliance.org/ ?getfile=1510.

[31] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture : an Overview. RFC 1633 (Informational), June 1994.

[32] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[33] J. Roberts. Quality of service guarantees and charging in multi-service networks. *IEICE Transactions on Communications*, (E81-B(5)) :15, 1998.

[34] C. Chassot, F. Garcia, A. Lozes, P. Anelli, and T. Bonald. Architecture de QoS en environnement IPv6 à services différenciés". In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, Toulouse, Octobre 2000.

[35] T. Bonald and J.W. Roberts. Performance of bandwidth sharing mechanisms for service differentiation in the Internet. In *specialist seminar on IP traffic (ITC)*, Monterey (USA), September 2000.

[36] P. Anelli and T. Ziegler. Evaluation de la différenciation avec WFQ ou WRED. Rapport AIRS, Septembre 1999.

[37] T. Bonald, M. May, and J-C. Bolot. Analytic evaluation of RED performance. In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*, volume 3, pages 1415–1424, Tel-Aviv, Israël, March 2000.

[38] T. Ziegler, S. Fdida, and U. Hofmann. RED+ gateways for identification and discrimination of unfriendly best-effort flows in the Internet. In *IFIP Broadband Communications*, pages 27–38, 1999.

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

[40] I. Cidon, R. Guérin, and A. Khamisy. On protective buffer policies. *IEEE/ACM Transactions on Networking*, 2(3) :240–246, 1994.

[41] K.V. Cardoso, J.F. De Rezende, and N.L.S. Fonseca. On the effectiveness of push-out mechanisms for the discard of TCP packets. 2001.

[42] F. Garcia, C. Chassot, A. Lozes, M. Diaz, P. Anelli, and E. Lochin. Conception, implementation and evaluation of a QoS - based architecture for an IP environment supporting differential services. In *Interactive Distributed Multimedia Systems (IDMS'2001)*, Lancaster, UK., September 2001.

[43] C. Chassot, F. Garcia, G. Auriol, A. Lozes, E. Lochin, and P. Anelli. Performance analysis for an IP differentiated services network. In *IEEE International Conference on Communications (ICC)*, New York, NY, USA, April 2002. IEEE.

[44] F. Garcia, G. Auriol, C. Chassot, A. Lozes, E. Lochin, and P. Anelli. Conception, implementation et mesure des performances d'une architecture de communication à qds garantie dans un domaine IPv6 à services différencés". In *9ième Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'02)*, Montréal, Québec, Canada, Mai 2002.

[45] W. Lin, R. Zheng, and J. C. Hou. How to make assured service more assured. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, page 182, Toronto, Canada, November 1999.

[46] E. Lochin, P. Anelli, and S. Fdida. Méthodes de garantie de débit d'un flot TCP dans un réseau à service assuré. In *10ième Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, Paris (France), Octobre 2003.

[47] E. Lochin, P. Anelli, S. Fdida, F. Garcia, G. Auriol, C. Chassot, and A. Lozes. Evaluation de la différenciation de services dans l'Internet. *Techniques et Sciences Informatiques*, 23(5–6) :675–699, Mai 2004.

[48] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu. On achievable service differentiation with token bucket marking forTCP. UMASS CMPSCI Technical Report 99-72, November 1999.

[49] G. Huston. Next Steps for the IP QoS Architecture. RFC 2990 (Informational), November 2000.

[50] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. Updated by RFCs 4301, 6040.

[51] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, pages 43–56, Stockholm, August 2000. ACM.

[52] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. In *Computer Networks and ISDN Systems*, volume 17, pages 1–14, 1989.

[53] E. Lochin, P. Anelli, and S. Fdida. AIMD penalty shaper to enforce assured service for TCP flows. In *4th International Conference on Networking (ICN)*, pages 275–283, pp.8, La Réunion, France, April 2005. IEEE, IARIA.

[54] E. Lochin, P. Anelli, and S. Fdida. Penalty shaper to enforce assured service for TCP flows. In *Proc. of Networking*, University of Waterloo, Waterloo Ontario Canada, May 2005. IFIP Technical Committee on Communication Systems (TC 6).

[55] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. QoS's downfall : At the bottom, or not at all ! In *Proc. of ACM SIGCOMM*, 2003.

[56] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang. Advances in Internet quality of service, October 2001.

[57] E. Lochin and P. Anelli. TCP throughput guarantee in the diffserv assured forwarding service : What about the results ? *Annals of Telecommunications*, 64(3-4) :215–224(10), October 2008.

[58] A. Tang, D. Wei, S.H. Low, and M. Chiang. Heterogeneous congestion control : Efficiency, fairness and design. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, November 2006.

[59] M. Csernai, A. Gulyas, Z. Heszberrger, S. Molnar, and B. Sonkoly. Congestion control and network management in future Internet. *Scientific Association for Infocommunications Infocommunications Journal*, LXIV(IV), 2009.

[60] Y. Tian, K. Xu, and N. Ansari. TCP in wireless environments : Problems and solutions. *IEEE Communications Magazine*, 43(3), March 2005.

[61] G Huston. Gigabit TCP. *Cisco Internet Protocol Journal*, 9(2), June 2006.

[62] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock. Host-to-host congestion control for TCP. *IEEE Communications Surveys & Tutorials*, 12(3) :304 –342, 2010.

[63] B. Ford and J. Iyengar. Breaking up the transport logjam. In *in Seventh ACM Workshop on Hot Topics in Networks (HotNets-VII)*, Calgary, Alberta, Canada, October 2008.

[64] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5) :10–23, October 1994.

[65] A. Kuzmanovic. The power of explicit congestion notification. In *Proc. of ACM SIGCOMM*, pages 61–72, Philadelphia, August 2005. ACM.

[66] A. Medina, M. Allman, and S. Floyd. Measuring the evolution of transport protocols in the Internet. *ACM Computer Communication Review*, 35(2), April 2005.

[67] S. Bhandarkar, N. Reddy, Y. Zhang, and D. Loguinov. Emulating AQM from end hosts. In *Proc. of ACM SIGCOMM*, 2007.

[68] H. Balakrishnan and S. Seshan. The Congestion Manager. RFC 3124 (Proposed Standard), June 2001.

[69] B. Carpenter and S. Brim. Middleboxes : Taxonomy and Issues. RFC 3234 (Informational), February 2002.

[70] S. Floyd. Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166 (Informational), March 2008.

[71] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[72] P. Benko and A. Veres. A passive method for estimating end-to-end TCP packet loss. In *Proc. of IEEE GLOBECOM*, November 2002.

[73] M. Allman, W. Eddy, and S. Ostermann. Estimating loss rates with TCP. *ACM SIGMETRICS Performance Evaluation Review*, 31(3) :12–24, December 2003.

[74] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata. Forward RTO-Recovery (F-RTO) : An Algorithm for Detecting Spurious Retransmission Timeouts with TCP. RFC 5682 (Proposed Standard), September 2009.

[75] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP. RFC 3522 (Experimental), April 2003.

[76] E. Blanton and M. Allman. Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions. RFC 3708 (Experimental), February 2004.

[77] P. Anelli, F. Harivelo, and E. Lochin. Détection des evènements de congestion de TCP. In *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, page 12, Arcs, France, Mars 2008.

[78] P. Anelli, E. Lochin, F. Harivelo, and D. Lopez. Transport congestion events detection (TCED) : Towards decorrelating congestion detection from TCP. In *25th Symposium On Applied Computing (SAC)*, page 7, Sierre, Switzerland, March 2010. ACM Special Interest Group on Applied Computing.

[79] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe. Open Research Issues in Internet Congestion Control. RFC 6077 (Informational), February 2011.

[80] A. Hafsaoui, D. Collange, and G. Urvoy-Keller. Revisiting the performance of short TCP transfers. In *Proc. of Networking*, Aachen, Germany, May 2009. Springer-Verlag.

[81] D. Ciullo, M. Mellia, and M. Meo. Two schemes to reduce latency in short lived TCP flows. *IEEE Communications Letters*, 13(10), October 2009.

[82] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir. Atlas Internet observatory 2009 annual report. In *47th NANOG*, 2009.

[83] X. Chen and J. Heidemann. Preferential treatment for short flows to reduce web latency. *Computer Networks*, 41(6) :779–794, April 2003.

[84] S. Floyd. Congestion Control Principles. RFC 2914 (Best Current Practice), September 2000.

[85] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, N. Sutin, A. Agarwal, T. Herbert, and A. Jain. An argument for increasing TCP's initial congestion window. *ACM Computer Communication Review*, 40(3), July 2010.

[86] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298.

[87] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows : Predictability of the response time. In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*, Hong Kong, 2004. IEEE.

[88] R. Pan, B. Prabhakar, and K. Psounis. Choke : A stateless aqm scheme for approximating fair bandwidth allocation. In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*. IEEE, 2000.

[89] J. Martin, A. Nilsson, and I. Rhee. Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, 11(3) :356 – 369, June 2003.

[90] P. Anelli, R. Diana, and E. Lochin. Favourqueue : A stateless active queue management to improve TCP traffic performance. *Submitted in Computer Networks*, 2011.

[91] P. Anelli, F. Harivelo, and R. Lorion. TCP syn protection : An evaluation. In *To appear in The Eleventh International Conference on Networks (ICN)*, page 6, La Réunion, France, March 2012. IEEE, IARIA.

[92] J. Crowcroft. Cold topics in networking. *ACM Computer Communication Review*, 38(1) :45–47, January 2008.

[93] A. Meddeb. Internet QoS : Pieces of the puzzle. *IEEE Communications Magazine*, January 2010.

[94] B. Turner. Why there's no Internet QoS and likely never will be, December 2006.

[95] P. Anelli. Interface de modélisation pour ns-2. Guide d'utilisateur, 2008.

[96] S.B. Eisenman, E. Miluzzo, N.D. Lane, R.A. Peterson, G.-S. Ahn, and A.T. Campbell. Bikenet : A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks*, 6(1), December 2009.

[97] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking : Data forwarding in disconnected mobile ad hoc network. *IEEE Communications Magazine*, 44(11) :134–141, 2006.

[98] N. Lane, Z. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A.T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9), September 2010.

[99] F. Mattern and C. Floerkemeier. From the Internet of computers to the Internet of things. In *From Active Data Management to Event-Based Systems and More*, pages 242–259. Springer-Verlag, 2010.

[100] M. Conti, S. Giordano, M. May, and A. Passarella. From opportunistic networks to opportunistic computing. *IEEE Communications Magazine*, September 2010.

[101] D. Bandyopadhyay and J. Sen. Internet of things : Applications and challenges in technology and standardization. *Springer Wireless Personnal Communications*, pages 49–69, 2011.

[102] B. Raghavan and J. Ma. The energy and emergy of the Internet. In *10th ACM Workshop on Hot Topics in Networks (HotNets-X)*, Cambridge, Massachusetts, November 2011.

[103] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4), April 2010.

[104] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez. Greening the Internet with nano data centers. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, 2009.

[105] G. Camarillo and IAB. Peer-to-Peer (P2P) Architecture : Definition, Taxonomies, Examples, and Applicability. RFC 5694 (Informational), November 2009.

[106] C. Deleuze. Content networks. *Cisco Internet Protocol Journal*, 7(2), June 2004.

[107] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, December 2009.

[108] U. Lee, I. Rimac, and V. Hilt. Greening the Internet with content-centric networking. In *Proceedings of the 1st International Conference on Energy-Efficient (e-Energy)*. ACM, 2010.

[109] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. In *Workshop on SIGCOMM Hot Topics in Networks (HotNets)*. ACM, 2005.

[110] P. Papadimitriou, O. Maennel, A. Greenhalgh, A. Feldmann, and L. Mathy. Implementing network virtualization for a future Internet. In *International Teletraffic Congress (ITC)*, 2009.

[111] J. Carapinha, P. Feil, P. Weissmann, S.E. Thorsteinsson, C. Etemoglu, O. Ingborsson, S. Ciftçi, and M. Melo. Network virtualization - opportunities and challenges for operators. In *Fis 2010*, volume 6369 of *Lecture Notes in Computer Science*. Springer-Verlag, 2010.

[112] K. Chowdhury and R. Boutaba. A survey of network virtualization. *Elsevier Computer Networks*, 54 :862–876, 2010.

[113] J. Rexford. The networking philosopher's problem. *ACM Computer Communication Review*, 41(3), July 2011.

[114] C. Partridge. Forty data communications research questions. *ACM Computer Communication Review*, 41(5), October 2011.

[115] M. Mathis, J. Semke, J. Mahdavi, and T.J. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM Computer Communications Review*, 27(3) :67–82, July 1997.

[116] A. Callado, C. Kamienski, G. Szabó, B.P. Gero, Judithkelner, S. Fernandes, and D. Sadok. A survey on Internet traffic identification. *IEEE Communications Surveys & Tutorials*, 11(3) :37–52, Third quarter 2009.

[117] S. Radhakrishnan, Y. Cheng, J. Chu, and A. Jain. TCP fast open. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, December 2011.

[118] Q. Li, W. Zhou, M. Caesar, and B. Godfrey. ASAP : A low-latency transport layer. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, December 2011.

[119] D. Damjanovic, P. Gschwandtner, and M. Welzl. Why is this web page coming up so slow ? investigating the loss of SYN packet (work in progress). In *Proc. of Networking*, Aachen, Germany, May 2009.

[120] A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*, pages 1157 – 1165. IEEE, March 2000.

[121] M. Scharf. Comparison of end-to-end and network-supported fast startup congestion control schemes. *Elsevier Computer Networks*, 55, February 2011.

[122] C. Raiciu, M. Handley, and D. Wischik. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356 (Experimental), October 2011.

[123] V. Konda and J. Kaur. Rapid : Shrinking the congestion-control timescale. In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*, 2009.

[124] C. Marcondes, M.Y. Sanadidi, M. Gerla, and H. Shimonishi. Tcp adaptive westwood combining tcp westwood and adaptive reno : A safe congestion control proposal. In *Proc. of the IEEE International Conference on Communications - ICC*, 2008.

[125] T. Bonald, M. Feuillet, and A. Proutiere. Is the law of the jungle sustainable for the Internet ? In *Proc. of the IEEE International Conference on Computer Communications - INFOCOM*. IEEE, 2009.

[126] K. Psounis, R. Pan, and B. Prabhakar. Approximate fair dropping for variable-length packets. *IEEE Micro*, 21(1), January 2001.

ANNEXES

Sommaire

1. Curriculum Vitae

2. Anelli, P. et Soto, M. ; (1999), IEEE Transactions on Communications, September, 47(9), pp. 1386-1393(8).
   Evaluation of the APS Protocol for SDH Rings Reconfiguration.
   `http://dl.comsoc.org/cocoon/comsoc/servlets/GetPublication?id=141085`

3. Lochin, E. et Anelli, P. ; (2008), Springer Annals of Telecommunications, October, 64(3-4), pp. 215-224(10).
   TCP Throughput Guarantee in the Diffserv Assured Forwarding Service : What About the Results ?
   `http://dx.doi.org/10.1007/s12243-008-0053-2`

4. Anelli, P. ; Lochin, E. ; Harivelo, F. et Lopez, D. ; (2010), SAC 25th Symposium On Applied Computing, Sierre, Switzerland, March, pp. 7.
   Transport Congestion Events Detection (TCED) : Towards Decorrelating Congestion Detection from TCP.
   `http://dl.acm.org/citation.cfm?id=1774226`

5. Anelli, P. ; Lochin, E. et Diana, R. ; (2011), ArXiv Research report, 1103.2303, March and improved version submitted in Computer Networks, pp. 12.
   Favourqueue : A Stateless Active Queue Management to Speed up Short TCP Flows (and Others Too !).
   `http://arxiv.org/abs/1103.2303`

Evaluation of the APS Protocol for SDH Rings Reconfiguration

Pascal ANELLI, Michel SOTO

Abstract— **The SDH (Synchronous Digital Hierarchy) architecture is one of the underlying technologies used by ATM networks. The SDH includes various protection mechanisms. One main design issue is probably the reconfiguration process in case of failure. In case of SDH works on optical fibres with a ring network topology, the Automatic Protection Switching (APS) protocol can be used.**

This paper addresses the problem of maximum allowed recovery time in four fibers ring architecture. We analyze the APS protocol and derive upper bounds for the processing time in each node of the network in order to cope with the maximum reconfiguration time of 50 ms, as specified in the standard.

We finally analyze the behavior of the system in case of two interleaved failures. A worst case analysis is carried out, showing that a 100 ms reconfiguration time can be guaranteed.

Keywords— **Synchronous Digital Hierarchy, B-ISDN, Protocols, Communication System Performance**

I. Introduction

IN the area of B-ISDN, the ATM architecture of protocols is widely studied. This architecture needs a physical layer able to cope with the high throughput involved in the application developments they are supposed to support. The Synchronous Digital Hierarchy (SDH) technology offers technical possibilities to build an infrastructure of a high speed transport network which conveys the broadband services.

Broadband network services rely on high throughput and high reliability. SDH is an ITU-T digital transmission standard [1] that defines common interfaces for vendor compatibilities, digital hierarchy for fibre optic transmission and a frame structure for multiplexing. This standard was initially proposed by ANSI under the name Synchronous Optical Network (SONET)[2]. ITU-T has further refined and generalized the concept to produce the SDH (Synchronous Digital Hierarchy) of which SONET is a subset. An introduction in [3] presents the basic concepts about SDH and its frame formats.

SDH based networks will have many advantages over the digital networks currently in use. They well meet the requirements of the new broadband network services: the operation and maintenance functions provide automatic restoration of services when equipments or links fail like cable cuts for instance. Each level of SDH network disposes of overhead and elements of quality monitoring needed to exchange informations about its operation and maintenance [4].

To achieve the highest network reliability and survivability such networks must be designed according to a ring architecture [5] and uses physical redundancy. The ring topology allows to protect against link or node failures. whereas linear architecture is useful only to protect against link failures. The redundancy in the ring deals with bandwidth as well as network components. Thank to that redundancy, the ring has the ability to be self-healing. Moreover, the ring topology benefits include simplicity, flexibility and potential fast restoration time. The restoration times must be less than 50 ms after detection of the failure according to current network protection switching time requirements at the SDH level [6].

These architectures of self-healing ring (SHR) are divided into two categories: the bidirectional SHR's (B-SHR's) and unidirectional SHR's (U-SHR's)[7]. The type of the ring is defined according to the direction of the traffic flow under normal working conditions. In a B-SHR, the duplex traffic is on the same path and traverses the same set of nodes for both directions of transmission. Conversely, in U-SHR, the duplex traffic travel over opposite path and all the nodes of the ring are involved i.e. transmitting and receiving in normal condition is done on the same fibre. The SHR can be further categorized into path or multiplex section protection depending on which layer the protection switch is made [5]. The path protection switching uses path layer indications and restore individual end-to-end service channel. While multiplex section protection switching uses indications located in the section overhead (SOH) to restore multiplex demand from a failed equipment. The various protection schemes for fibre networks are presented in [8]. In [6], ITU-T defines a protection scheme for the multiplexage section level. The restoration service at this level is achieved by using Automatic Protection Switching (APS) system to perform a loop-back function when a failure occurs. Each node of the network participates in the restoration process.

For the moment, the APS system is well-defined for the Multiplex Section Shared Protection Rings (MS-SPRing) [6]. A MS-SPRing is a bidirectional ring that uses the multiplex section level status and performance parameters to initiate APS. The APS system is based on three different elements: first, the presence of a protection channel which is able to substitute to the channel carrying the traffic under working conditions; second, the capability in the nodes to switch and to select the traffic between protection channel and main channel; third, a protocol that provides self-healing capability to mitigate network component failure and uses the SDH multiplex section layer indications for

This work was supported by Alcatel CIT, Lannion center, France.

Université Pierre et Marie Curie, LIP6, Paris, France. E-mail: Pascal.Anelli@lip6.fr

Université René Descartes, LIP6, Paris, France. E-mail: Michel.Soto@lip6.fr

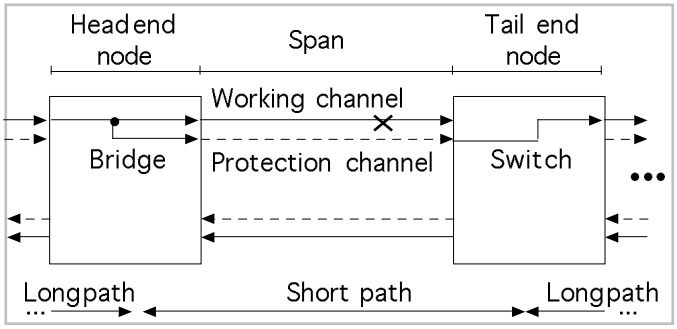


Fig. 1. APS vocabulary

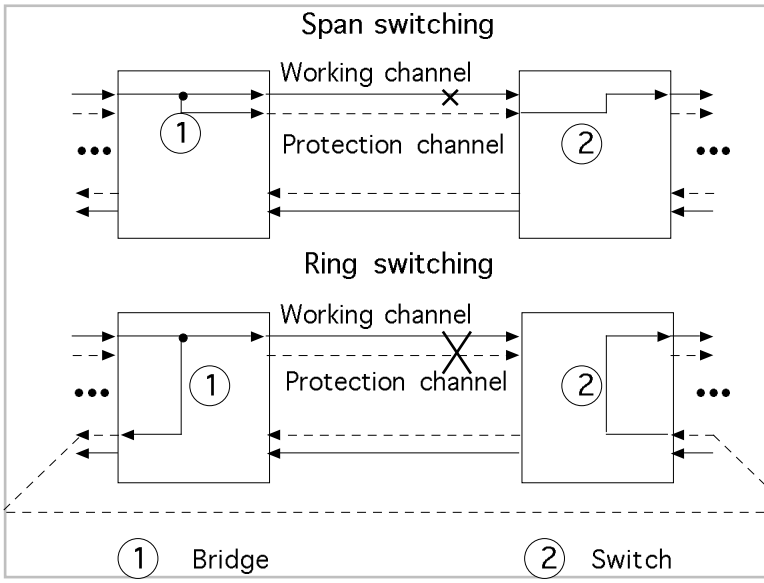


Fig. 2. Protection mechanisms

protection switching.

It is crucial to know exactly the processing time allowed for a node to meet the current network protection switching time requirements (i.e. 50 ms). This consideration is taken into account in the design of the node architecture when implementing the APS system in a node.

The paper is organized as follows: section 2 describes the APS protocol in the SDH environment; section 3 presents the modeling approach and the chosen performance criteria; section 4 gives the modelization results; finally, section 5 presents some design issues for the SDH nodes.

II. The APS protocol

We will now present the APS principles and the vocabulary used in this paper [6]. Figure 1 shows the basic terms of the system. The traffic travels in the working channel bidirectionally on separate fibres through the same ring nodes. The working channel is the channel used for traffic transport under normal conditions, i.e. without any failure. A switch event occurs when a network component failure occurs. The working channel is protected by an alternate channel (the protection channel) which is used to transport the traffic during a switch event. Otherwise, protection channel is used to carry extra traffic which is not protected and could be preempted in case of a switch event. MS-SPRing can be implemented on 2 or 4 fibres. With a two-fibre MS-SPRing, protection channel is provided by reserving a part of the bandwidth on each fibre. In this case, no fibre is dedicated for protection, so the protection channel and the working channel share the same fibre. With the four-fibre MS-SPRing, the working and protection channels are carried over separate fibres. In 2 or 4 fibres case, two adjacent nodes are connected by a set of four channels which is called a span.

A failure on a span is detected and corrected by its adjacent nodes. These nodes are called switching nodes and they use the bridge and switch actions for the protection of the working channel. The bridge action consists in transmitting identical traffic on both the working and protection channels. While the switch action consists in selecting traffic from the protection channel rather than the working channel. When a node detects a failure, it becomes a switching node and more precisely a tail end i.e. it requests that the previous node behaves like a bridge. Conversely, a node that is notified for a failure is the second switching node called head end i.e. it executes a bridge. The requests for switching are transmitted on both the short and long path. The short path is defined as "the path segment on the span for which the request is initiated". This span is always the one to which both the head end and the tail end are connected. The long path is the other path segment which connects the head end and the tail end but including all the other spans. Therefore, other nodes belong to this path segment.

There are two protection mechanisms as shown on figure 2: ring switching and span switching. The latter is only used on a four-fibre MS-SPRing in order to provide an additional degree of protection. A span switching consists in the transmission of the working traffic on the protection channel of the span where the failure occurs. In this case, the protection channel is substituted to the working channel in fault. A restoration using the ring switch is only needed if both the protection and the working channels on the same span are affected by failures. Here, the working traffic is transmitted to the other switching node, through the protection channel of the long path.

When a node decides that a switch is required, it sends the appropriate request in both directions, i.e. the short and long path. Consequently, a node can receive the same request from different directions. Span switching fully relies on the request transmitted on the short path. The long path signaling informs the other nodes that a span switch exists elsewhere in the ring. At the opposite, the ring switching fully relies on the requests transmitted on the long path. The failure of a channel can be any one of two: a hard failure mainly due to a loss of signal, or a soft failure asserted when the observed bit error ratio (BER) exceeds a preselected threshold. In the first case, the failure is called

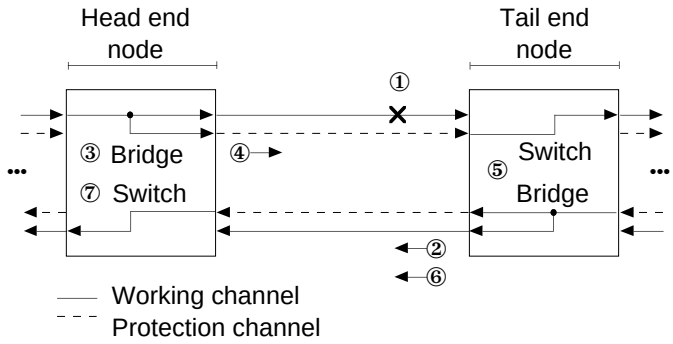


Fig. 3. SDH automatic protection switching principle

a Signal Fail (SF) and in the later case, it is called a Signal Degrade (SD).

A. APS algorithm

The APS protocol uses in-band signaling for protection switching through K_1 and K_2 bytes within the SDH multiplex section overhead. K_1 identifies the switch priority and the destination node while K_2 designates the source node and indicates the path and the status of the switching operation. The basic idea of the protocol is to ensure the working traffic protection by local actions. The two neighbors of the failure are involved in the protocol exchange. The protocol operation follows this general pattern as shown on figure 3:
1. The node which detects a failure (1) (tail end) sends a request (2) to the other adjacent node (head end) of the failed span. The request depends on the type of the failure.
2. The head end, after receiving the request, executes a bridge or a bridge and switch (3). Then it sends a acknowledgment to the tail end (4).
3. On receiving the confirmation, the tail end executes a bridge and switch in one or two steps (5). It terminates by sending its status (6).
4. Finally, the head end finishes by a switch if necessary (7).

The requests to perform protection switching can be initiated either externally or automatically. The external commands are initiated by the operating system or by a craft person. Afterwards, only the automatically initiated protection switching commands are studied because they are based on multiplex section and equipment performance criteria. So, the lockout of channel, the forced switch, the manual switch and the exercise are excluded. Likewise, we do not take into account failures of the protection channel. Only the traffic carried over the working channel is protected and consequently, no mechanism is provided to protect information possibly transferred over the protection channel when no switch even took place. There are four automatically initiated commands. When a hard failure affects the working channel only, the restoration is done in using the span switching method. In this case, the command is Signal-Fail-Span (SF_S). The Signal-Fail-Ring (SF_R) command is used for the same type of failure affecting both the working and protection channels over the same span. In case of soft failure affecting the working channel, the command initiated is the Signal-Degrade-Span (SD_S) and a Signal-Degrade-Ring (SD_R) when the failure affects both channels. When the failure has been cleared, the ring is restored in its initial state (if no other failure condition exists).

Since the protection channels are shared among all spans, contention among the nodes for the protection facility may arise when multiple failures occur. Consequently, the APS protocol defines a priority between the commands. The basic principle is that the priority of signal degrade is less than signal fail, and that span switching has priority over ring switching when facing the same type of failure (e.g. SF or SD). So the priority for the automatically initiated commands is in increasing order : SD_R, SD_S, SF_R, SF_S.

The priority is examined by each node before performing any protection switching activity. The APS protocol contains a preemption mechanism between ring and span switches. According the priorities determined by the standard, five couples of preemption schemes may exist. These couples are: (SD_R - SD_S), (SD_R - SF_R), (SD_R - SF_S), (SD_S - SF_R) and (SF_R - SF_S).

The standard allows the co-existence of switches under some special circumstances. Any span switch can co-exist with any other span switch; so SF_S and SD_S co-exist because these commands are processed locally and independently from each others. The SF_R command co-exist with SF_R command, the ring will recover by segmenting into multiple sub-rings. In this case, only the connections limited at one segment will be recovered. Conversely, when multiple SD_R commands exist over different spans, no action is executed and existing bridges and switches are dropped. If the failures can co-exist, each one is repaired but in case of preemption, only the last failure will be repaired at the end of the switch completion time. If the protection can be considered good for one failure, with two or more failures ring functionnality will be degraded.

III. The model

ITU-T in [6] defines the general network objectives. The maximum number of nodes on a MS-SPRing is sixteen. The end-to-end switch completion time should be within 50 ms after detection of a fault condition when no previous switch request exists for a ring of less than 1 200 km of fibre. The switch completion time is defined as the *interval from the decision to switch to the completion of the bridge and switch operation at a switching node initiating the bridge request.* This means that the detection time is not included in the switch completion time. This latter is completed as soon as the switching nodes complete their operation. So this time represents the amount of time following the failure detection until the effective recovery of the service (i.e., Bridge and Switch executed in each switching node).

During this period, the head and tail end nodes must execute the bridge and switch and thus must exchange the appropriate information. The request sent on the long path takes a time due to the propagation delay plus the processing time spent in each intermediate node. An inter-

mediate node is located between 2 switching nodes on the long path. The propagation delay depends on the length of the ring and on the signal speed through the fibre. For a network of maximum size (i.e. sixteen nodes and 1 200 km of fibre), the propagation delay is constant and depends on the physical properties of the fibre.

The question that arises now is to know the time that the switching and intermediate nodes can spend to process the current command without exceeding the maximum switch completion time, that is 50 ms. Unfortunately, the APS protocol has different behaviors for each command. For example, a node executing a span switch sends the request on the short path, while the long path is used for a ring switch. In general, the request sent on the long path takes more time to be received. Facing this problem, we will try to determine command which is critical with respect to the time it consumes, so that the reconfiguration can take place within the maximum allowed duration.

The processing time in the switching node includes the time for the K_1 and K_2 bytes generation, the K_1 and K_2 bytes software processing and loop back switch control. This paper evaluates the acceptable maximum processing time when a failure occurs, the ring being initially in normal state. When this maximum processing time is determined, we study the switch completion time when a second failure occurs during the processing of the first one. The second failure occurs at $t + \Delta t$, if t is the date of the first failure. The protection switching step is time critical. Consequently, the ring restoration in its initial state when the failure(s) has cleared is not studied. The study relies on real situations and is mainly focused on the worst case recovery time of the SDH ring. The results are obtained by simulation using the OPNET tool [9]. Simulation is the only way we can use, because of the complexity of the protocol, in particular its inherent parallelism.

According to [6], there are three node states: the idle state, the switching state, and the pass-through state. Conceptually, these states apply to a single APS controller which is the part of the node that is responsible of performing protection switching operations. The idle state means that the ring is running under normal conditions without any fault condition detected. The switching state is devoted to a switching node (tail or head end) near the outage span. The pass-through state is reserved to the intermediate nodes; these nodes are located between the switching nodes along the long path.

In the pass-through state, a node transmits on one side exactly what is received from the opposite side. The tail and head nodes go into the switch state and then begin the exchange of control informations using bytes K_1 and K_2 in both directions on the ring. The intermediate nodes only propagate K_1 and K_2 bytes and go into pass-through state as soon as they know the occurrence of the failure. In case of contention between commands, the priorities of these commands can modify the node state, in particular between switch and pass-through states.

For modeling purposes, we choose to divide the APS controller of the node in 2 parts, one for each span. Thus, a node is made of 2 state machines in our simulation model. The general behavior of an APS controller is to look at all incoming informations, then to choose the highest priority input, and to take action based on that choice. In order to model this behavior, each state machine communicates with the other, so that only the one in charge of the request of higher priority is active.

The switch completion time depends on the values of the APS protocol parameters we will now present. The processing time T_{proc}, as already explained, is one of the main parameters. It is the time for a node to move from the idle state to the switch state. The determination of a bound for this parameter is the main goal of this study. The rule on the K_1 and K_2 bytes validation also influences the switch completion time. This rule applies to nodes which are in the idle state and in the switch state. The K-byte validation rule is the following: *before accepting the K-bytes as valid by the node, the value must be received identically in three successive frames.* The time to detect the change of K_1 and K_2 bytes and the time to move from the idle state to the pass through state is represented in an intermediate node by T_s. The time T_s depends on the implementation of the change detection from idle to pass-through in an intermediate node. It is 0 ms in a full hardware processing implementation and 1 ms in a software processing implementation. Once an intermediate node is in the pass-through state, it is supposed to transmit transparently K_1 and K_2 bytes from one span to the other with a pass through delay D_p. This maximum delay is equal to 125 μs. We assume that the average size of a span is 80 km (obtained by dividing the maximum total size of 1 200 km by the maximum number of nodes over the ring, i.e. 16). The propagation delay (T_p) over a span is supposed to be constant and equal to 380 μs. Finally T_{base} represents the transmission time for a SDH frame i.e. one frame (including K_1 and K_2 bytes) every 125 μs.

The span switch commands are initiated on a MS-SPRing with four fibres. Accordingly the network model uses four fibres. The processing of a request received on the short path is not interrupted by the reception of a request with the same priority on the long path. The last request received will be processed by the switching node when the processing of the current request is completed. Circuits are ignored because they do not matter in re-establishing traffic continuity.

The value of T_{proc} is supposed to be the same for each switching nodes. When a node moves from idle state to switch state, T_{proc} is divided in 2 components: T_{p_1} represents K_1 and K_2 bytes software processing time, that is the reaction time when the switching node has detected a failure or has been notified that a failure has occurred, plus the time for K_1 and K_2 bytes generation; T_{p_2} is the bridge and switch time. The bridge and switch time is assumed to be equal to the bridge time $T_{p_{2_a}}$ plus the switch time $T_{p_{2_b}}$. So, we have:

$$T_{proc} = T_{p_1} + T_{p_2}$$

and

$$T_{p_2} = T_{p_{2_a}} + T_{p_{2_b}}$$

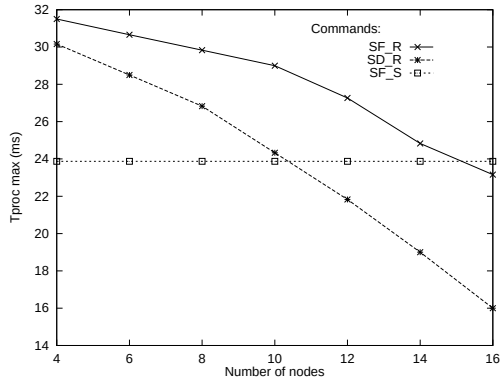


Fig. 4. Maximum admissible processing time for each command depending on network configuration

When a ring request is preempted, the time to undo a bridge and switch is equal to the time to execute the bridge and switch, T_{p_2}.

We begin by the calculation of the maximum processing time, $T_{proc_{max}}$, that each switching node can use without violating the maximum completion time of 50 ms from the failure detection to the recovery. The first approach uses the maximum value of T_s, that is 1 ms (software processing implementation case). Under that hypothesis, the time for a request to go from one switching node to the other by the long path is the bigger possible.

Unidirectional or bidirectional failures may occur on a ring. A unidirectional failure means that only one way of the span is damaged. In this case, the failure is detected by the tail end node and notified to head end node on the short path. The notification of the failure on the short path by the tail end node is a time consuming operation. When a bidirectional failure occurs, the span is damaged on both ways. As a consequence, a bidirectional failure is simultaneously detected by both of the switching nodes and both of them become a tail end node i.e. request a bridge. Thus, a bidirectional failure needs less time to restore traffic than a unidirectional failure since it is not necessary to notify the failure on the short path. Following these remarks, we focus the study only on unidirectional failures, involving the following commands: SF_S as a result of signal failure on the working channel only, SF_R when the signal failure is also on protection channel and SD_R as a result of signal degradation on both working and protection channels. The SD_S request is not studied because it is equivalent to SF_S request sequence. For convenience, in the following, the name of the command is also used as a synonym of the corresponding failure.

IV. Results

The value of $T_{proc_{max}}$ is evaluated in the worst cases of failure that is unidirectional failures and for 4-fibre rings ranging from 4 to 16 nodes. Assuming $T_{p_1} = T_{p_2}$, the values of $T_{proc_{max}}$ are shown on figure 4.

The maximum processing time, $T_{proc_{max}}$ is expressed by:

$$T_{proc_{max}}(i,j) = T^{-1}_{reconfig}(50ms, T_s)(i,j)$$

where $i \in \{SF\_S,\ SF\_R,\ SD\_R\}$, $j \in \{4,\ 5,\ \ldots,\ 16\}$ and $T^{-1}_{reconfig}$ is the inverse function of $T_{reconfig} = f(T_{proc}, T_s)$.

The analysis of figure 4 exhibits a constant $T_{proc_{max}}$ for span switching command, whatever is the number of nodes of the ring. For this command, the traffic protection switching procedure mainly uses the short path. The long path is not used and thus, the number of nodes has no effect.

It is interesting to notice that there is no command giving always the lowest value for $T_{proc_{max}}$, whatever is the number of nodes. When the ring has a small number of node, span switch commands give $T_{proc_{max}}$ values less than those obtained for ring switch commands in the same configuration. This means that a ring switching is faster than a span switching on small rings. This could be surprising because span switch only uses the short path and, thus, the time consumed to exchange commands is very low. The explanation of this phenomenon is given by the span switch procedure itself: the operations involved in a span switch for the switching nodes are purely sequential so that the needed processing steps alternate between the two switching nodes. Conversly, in the ring switch procedure, propagation of the commands and processing steps are done simultaneously. In this case, a command can arrive during the processing of previous one and can be processed immediately after. The parallelism of the the ring switch procedure is favored while the propagation time of the command through the long path is less than the processing time at the switching nodes; this situation occurs with a small number of nodes. When the number of node is greater than 10, SD_R becomes the worst failure case.

Figure 4 also shows that the value of $T_{proc_{max}}$ for the SD_R procedure is always less than the $T_{proc_{max}}$ for the SF_R procedure. The reason is also given by the SD_R procedure which needs one more exchange of command on the long path than the SF_R procedure.

Finally, the SF_R command allows the highest value for $T_{proc_{max}}$ for a number of nodes less than 15. Thus, the worst situation for T_{proc} is not related to the type of command executed. For commands using the long path (typically ring switch), the value of $T_{proc_{max}}$ is related to the number of node in the ring and the ring length. As the number of nodes and the ring length increase, the propagation of K_1 and K_2 bytes through the ring consumes time and thus, the acceptable value of $T_{proc_{max}}$ decreases. This fact is amplified by the software implementation detection of the change from idle to pass-through in an intermediate node. Consequently, $T_{proc_{max}}$ is different for every ring configuration.

From the set of values obtained for $T_{proc_{max}}$, we select the value $T_{proc_{limit}}$ defined as follow:

$$T_{proc_{limit}} = \min_{i,j} T_{proc_{max}}(i,j)$$

where

$$i \in \{SF\_S, SF\_R, SD\_R\} \text{ and } j \in \{4, 5, \ldots, 16\}$$

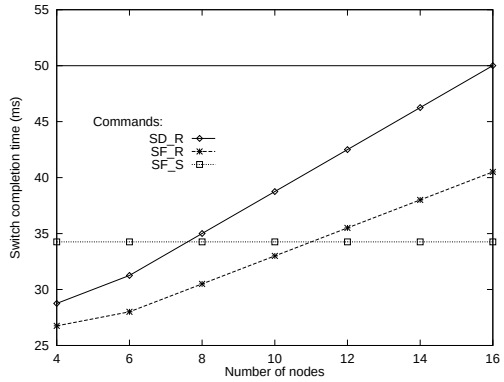


Fig. 5. Switch completion time for different network configurations

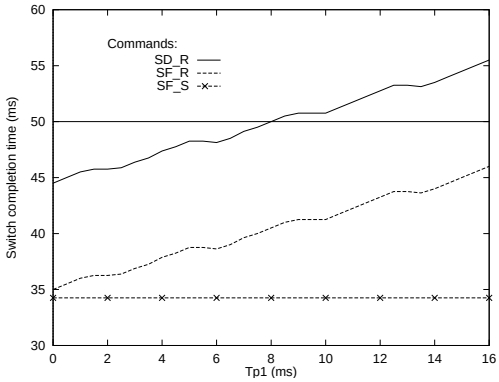


Fig. 6. Time distribution inside T_{proc}

The value of $T_{proc_{limit}}$ is the processing time not to be exceeded by switching nodes to guarantee that the switch completion time is less or equal to 50 ms for any ring ranging from 4 up to 16 nodes and for any command. As shown on figure 6, this value is met with a 16 nodes ring on unidirectional SD_R failure. In this situation, $T_{proc_{limit}}$ is equal to 16 ms. Figure 5 shows the switching completion times for the different failures, when $T_{proc_{limit}}$ is used and T_s is equal to 1 ms.

Figures 4 and 5 show that for span switch commands, the only important parameter affecting the completion time is T_{proc}. In this case, the long path is not used. Further, when requests propagation times are small compared to processing times, ring switch procedures result on smaller reconfiguration times than span switch procedures. This is due, as we mentioned before, to the intrinsic parallelism of ring switch procedure. Finally, the protection procedure following a SF_R is faster than that following a SD_R, whatever is the configuration of the ring. The reason is the extra request through the long path needed by the later one.

For convenience, $T_{p_1} = T_{p_2}$ was assumed. Nevertheless, actual distribution of T_{p_1} and T_{p_2} are strongly implementation dependent. In most of cases, T_{p_1} will be different from T_{p_2}. So, in the following, this assumption is relaxed and the completion time is studied when T_{p_1} ranges from 0 to $T_{proc_{limit}}$. The value of T_{p_2} is given by:

$$T_{p_2} = T_{proc_{limit}} - T_{p_1}$$

The switch completion time for a 16 nodes ring is shown on figure 6. As expected, the completion time of the SF_S command is not sensitive to the respective values of T_{p_1} and T_{p_2}. At the opposite, the completion times of the SD_R and SF_R commands are influenced: when $T_{p_1} < T_{p_2}$, the completion time is reduced. As T_{p_1} is the part of T_{proc} devoted to K_1 and K_2 processing, a small value for T_{p_1} favors the parallelism of the ring switch procedures. In consequence, implementors should focus on optimization of K_1 and K_2 bytes processing time to get efficient implementations of the APS protocol.

APS standard only defines a switching completion time requirement of 50 ms for a single failure on a clean ring (i.e, no extra traffic and no previous bridge request). Under all other conditions, the required completion time may be exceeded but must remain less than 100 ms. The priority scheme allows APS to deal with more than one failure. It is interesting to evaluate how APS behaves in worst case regarding the switch completion time, that is when a second failure occurs during the traffic restoration process due to a first failure.

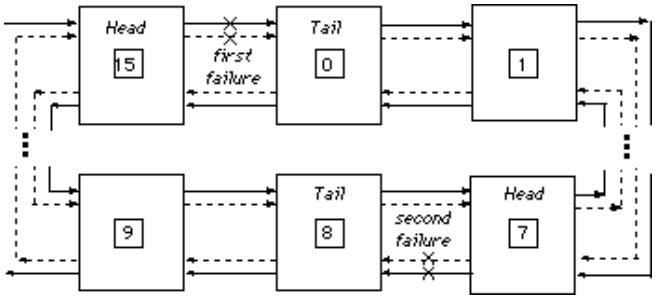


Fig. 7. Network configuration for 2 interleaved failures

In this section, the switch completion time is studied when two interleaved failures occur on four-fiber ring. We selected 3 scenarios of paired failures:

1. SD_R then SF_S, preemption of ring request by a span request,
2. SD_R then SF_R, preemption of a ring request by another ring request,
3. SF_R then SF_R, coexisting ring requests. At the end of the switch completion time, the ring is segmenting into two sub-rings.

We assume the second failure is always located on the farest span from the one where the first failure occurs as shown on Figure 7. The first failure occurs on the span between nodes 15 and 0 while the second failure occurs on the span between nodes 7 and 8.

The delay between the first failure and the second failure ranges from 0 to the completion time of the first failure. Figure 8 shows the results obtained with $T_{proc_{limit}}$, and T_s = 1 ms. For any pair of interleaved failures the completion time stays under 50 ms while the second failure occurs within 16 ms after the first one. In any case, the completion

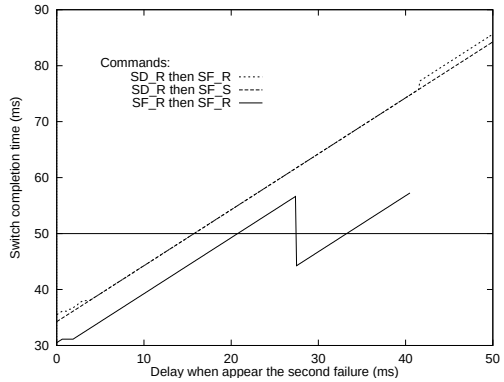


Fig. 8. 2 interleaved failures

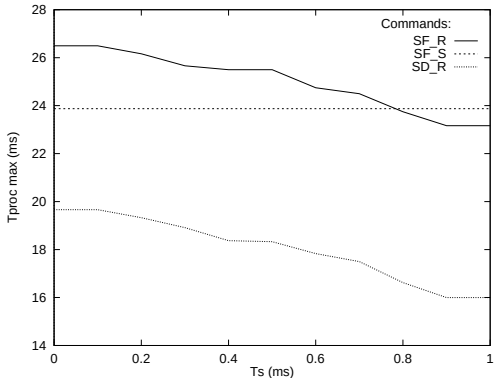


Fig. 9. $T_{proc_{max}}$ as a function of T_S

time is less than 100 ms.

The unusual shape of the curves is explained by the location of the first request when the second failure occurs. If we focus on the curve of the two interleaved SF_R failures, which is the most unsual, we can see the switch completion time growing lineary while the second failure happens in a delay less than 27.5 ms after the first failure.

Until 27.5 ms, the first request can not pass on the span between nodes 7 and 8 before the second failure occurs. The ring reconfiguration process for the first failure is always delayed by the starting of the ring reconfiguration process due to second failure. The thrust of the curve occurs at time 27.5 ms. From this time, the first request always passes the span between nodes 7 and 8 *before* the second failure occurs. In consequence, the ring reconfiguration process for the first failure is no more delayed by the starting of the ring reconfiguration process due to second failure and the two ring reconfiguration processes are performed with a lot of more parallelism. Such a parallelism immediatly decreases the switch completion time. Then, the switch completion time restart growing lineary. In general, the time of the thrust on the curve is directly linked to the location on the ring of the second failure. Actually, on the long path, the closer is the second failure to tail of the first failure, the sooner the thrust of the curve occurs.

As mentioned before, the T_s parameter is related to the implementation of the change detection from idle state to pass-through state in an intermediate node. The value of T_s has a direct impact on the propagation of K_1 and K_2 bytes on the long path and thus on the value of $T_{proc_{max}}$. We have previously set T_s to 1 ms. We now study $T_{proc_{max}}$ when T_s ranges from 0 to 1 ms on a 16 nodes ring.

As shown on figure 9, the admissible value of $T_{proc_{max}}$ decreases as the value of T_s increases. As we can have predicted, SF_S is insensitive to different values of T_s. The reason is that it does not use the long path. On the other hand, for ring switch type procedures, it is to be noticed that the greater is T_s, the smaller is $T_{proc_{max}}$. This is due to the requests exchanged that consume more time to the detriment of the processing time in the switching nodes. To verify this relation between T_s and $T_{proc_{max}}$ we studied the switch completion time as function of T_{proc} for T_s = 0 ms and T_s = 1 ms. The results on figure 10 confirm that, whatever is the value of T_{proc}, the switch completion time is always greater for T_s = 1 ms than for T_s = 0 ms.

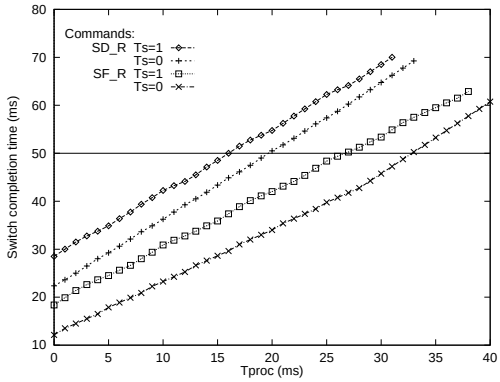


Fig. 10. Switch completion time for ring request with $T_S = 0$ and $T_S = 1$

Nevertheless, these results are not any longer valid with two interleaved failures. Figure 11 shows that the switch completion time is greater for T_s = 0 ms than for T_s = 1 ms. If we consider that T_{proc} is consumed twice (i.e., once for each interleaved failure) in the switch completion time we can make the following approximation to explain this contradictory result:

$$Switch\ completion\ time = 2 * T_{proc} + T_{exchange}$$

where $T_{exchange}$ represents the time consumed for exchanging the needed information between switching nodes. When T_s = 0 ms, the value of T_{proc} will be the highest possible. As T_{proc} is consumed twice, the switch completion time will be more important for T_s = 0 ms than for T_s = 1 ms. This result has consequence for design issues as explained in section 5.

V. Design issues

Implementors of the APS protocol should keep in mind there is no protection switching command that involves the lowest T_{proc} whatever the number of node of a SDH ring. At the opposite, the available T_{proc} is a function of both

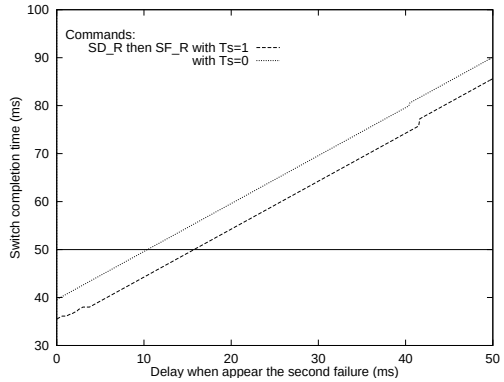


Fig. 11. 2 interleaved failures with 2 different T_S

the switching command and the number of node except for the SF_S command which exhibits a constant completion time in any case. An efficient implementation also involves an optimization of K_1 and K_2 processing (i.e., T_{p_1}) after a detection of an error condition in the switching nodes rather than an optimization of the bridge and switch operation (i.e., T_{p_2}). In this case, the switch completion time is reduced. For instance, if T_{p_1} is equal to T_{p_2}, T_{proc} must be less than 16 ms in order to keep the switch completion under 50 ms. In the intermediate nodes $T_s = 0$ allows the maximum value for T_{proc}. Nevertheless, with two interleaved failures the switch completion is greater when $T_s = 0$ ms. In others words, with two interleaved failures it is better to try to reduce T_{proc} rather than T_s. The switch completion time is more strongly reduced.

A good implementation of APS controller is one which minimizes the processing time in the switching node. So, in case of 2 failures interleaved, the switch completion is reduced when T_{proc} decrease and this whatever the value of T_s.

VI. Conclusions

This study has determined the maximum time processing available in a switching node to meet 50 ms completion time requirement whatever the ring configuration and the type of failure. An important result is that the completion time is not related to a single command. Particularly, when there are less than 8 nodes, ring switch commands are faster than span switch ones. This study has also shown with the maximum processing time available (with our hypothesis, T_{proc} = 16 ms) in a switching node, in any way, the completion time in the case of two interleaved failures meets the requirement of be under 100 ms. Finally the impact of the processing time in the intermediate node has been relativized from this of the switching node.

Acknowledgment

The authors would like to thank Pr. Éric Horlait for his careful reading and his helpful english writing of this paper.

References

[1] ITU-T Recommandation G.707, *Network node interface for the synchronous digital hierarchy (SDH)*, March 1996.

[2] American Standard for Telecommunications, *Telecommunications - Synchronous Optical Network (SONET) - Basic Description including Multiplex Structures, Rates and Formats*, 1995, T1.105-1995.

[3] C.G. Omidyar and A. Aldridge, "Introduction to SDH/SONET," *IEEE Communications Magazine*, vol. 31, no. 9, pp. 30–33, September 1993.

[4] M. Sexton and A. Reid, *Broadband Networking: ATM, SDH, and SONET*, Artech House, second edition, 1997.

[5] J. Hughes, D. Beckett, and J. Meloy, "Protection switching," *Telephone Engineer and Management*, vol. 97, no. 4, pp. 35–39, 1993.

[6] ITU-T Recommandation G.841, *Types and characteristics of SDH network protection architectures*, July 1995.

[7] T.H. Wu and R.C. Lau, "A class of self-healing ring architectures for SONET network applications," *IEEE Transactions on Communications*, vol. 40, no. 11, pp. 1746–1756, November 1992.

[8] T.H. Wu, "Emerging technologies for fiber network survivability," *IEEE Communications Magazine*, vol. 33, no. 2, pp. 58–74, February 1995.

[9] MIL 3 Inc, The INTELSAT Building, 3400 International drive, NW, Washington DC 20008, *OPNET: OPtimised Network Engineering Tool*, m version edition, 1991, M version.

TCP throughput guarantee in the DiffServ Assured Forwarding service: what about the results?

Emmanuel Lochin[1] and Pascal Anelli[2]

[1]Université de Toulouse, ISAE, LAAS-CNRS, France
[2]IREMIA - Université de la Réunion, France

emmanuel.lochin@isae.fr, pascal.anelli@univ-reunion.fr

***Abstract*—Since the proposition of Quality of Service architectures by the IETF, the interaction between TCP and the QoS services has been intensively studied. This paper proposes to look forward to the results obtained in terms of TCP throughput guarantee in the DiffServ Assured Forwarding (DiffServ/AF) service and to present an overview of the different proposals to solve the problem. It has been demonstrated that the standardized IETF DiffServ conditioners such as the token bucket color marker and the time sliding window color maker were not good TCP traffic descriptors. Starting with this point, several propositions have been made and most of them presents new marking schemes in order to replace or improve the traditional token bucket color marker. The main problem is that TCP congestion control is not designed to work with the AF service. Indeed, both mechanisms are antagonists. TCP has the property to share in a fair manner the bottleneck bandwidth between flows while DiffServ network provides a level of service controllable and predictable. In this paper, we build a classification of all the propositions made during these last years and compare them. As a result, we will see that these conditioning schemes can be separated in three sets of action level and that the conditioning at the network edge level is the most accepted one. We conclude that the problem is still unsolved and that TCP, conditioned or not conditioned, remains inappropriate to the DiffServ/AF service.**

***Index Terms*—QoS, End to End guarantee, TCP, DiffServ, Assured Forwarding.**

I. Introduction

The Differentiated Services architecture [1] proposes a scalable mean to deliver IP Quality of Service (QoS) based on handling of traffic aggregates. This architecture adheres to the basic Internet philosophy namely that complexity should be relegated to the network edges while simple functionality should be located in the core network. This architecture advocates packet tagging at the edge and lightweight forwarding in the core. Core devices perform only differentiated aggregate treatment based on the marking set by the edge devices. Edge devices in this architecture are responsible to ensure that user traffic conforms to traffic profiles.

The Assured Forwarding (AF) Per Hop Behavior (PHB) is one of the DiffServ forwarding mechanism [20]. The service called Assured Service (AS) built on top of the AF PHB is designed for elastic traffics and is intended to assure a minimum level of throughput. The minimum assured throughput is given according to a negotiated profile with the client. The throughput increases as long as there are available resources and decreases when congestion occur. Such traffic is generated by adaptive applications.

In the assured service, the throughput of these flows breaks up into two parts. First, a fixed part that corresponds to a minimum assured throughput. In the event of network congestion, the packets of this part are preserved from loss (colored green or marked in-profile). Second, an elastic part which corresponds to an opportunist flow of packets[1] (colored red or marked out-profile). No guarantee is brought to these packets. They are conveyed by the network on the principle of best-effort service (BE). In case of congestion, these packets are first dropped. This opportunistic part of the flow must vary according to the level of resources used, hence its elastic character. In any case, the throughput offered by this service must be better than the BE service. In this architecture, the ultimate goal is to obtain an assured throughput in the absence of per-flow treatment in the network.

The drop precedence sets in the core routers provides a good indication of the congestion level. If the network is far from being congested, the in-profile packets will rarely be dropped and their dropping probability will be neglectable. If the network is going to be congested, almost all of the out-profile packets will be dropped. The dropping mechanism used on the core routers is generally the well-known RIO queue [5]. RIO is the basic active queue management mechanism suitable for the setup of the AF PHB. In order to decide whether to discard out-profile packets, respectively in-profile packets, RIO uses the average size of the total queue formed by in-profile and out-profile, respectively in-profile packets only.

Concerning the edge of the network, edge routers use a conditioner/marker in order to profile the traffic. There isn't hypothesis on the localization of these conditioner/marker. Indeed, they could be set on the client side rather than the edge router. In the first DiffServ network specification, the edge routers used a token bucket marker mechanism in order to characterize the traffic by marking in-profile and out-profile the packets of a flow. This traffic profile consists of a minimum throughput, characterized by two token bucket parameters, namely the token rate r and the size of the bucket b. Thus, the conformity control of an aggregate compared to the profile is

[1]an opportunist traffic is a traffic which occupies more bandwidth than its target rate in a congested network

done by a token bucket as proposed in [20], [21].

It is likely that the assured service was designed for applications relying on the TCP protocol. TCP increments continuously its throughput and as a consequence, the bandwidth occupation by increasing the data transmission rate in function of the acknowledgement packets. If the network drops packets, TCP decreases its transmission rate. Obviously, TCP is not aware of the underlying QoS offered by the network. In the assured service, this TCP feature can involve poor performances. If a user is allowed to send packets exceeding profile requirements, these packets will be classified as out-profile by the edge routers. In case of network congestion, these packets can be dropped. Depending of the number of losses, this dropping can involve a high reduction of the transmission rate at the TCP level. As a consequence, the performance of a TCP flow carried out by the assured service is mainly determined by its out-of-profile packets. Even if the network has sufficient bandwidth for in-profile packets, the losses experienced by out-of-profile packets decrease the overall performance of the TCP flow. Indeed, the TCP congestion control is not aware of the assured traffic. This problem is the motivation of several years of research in order to correctly characterize the TCP flow in a DiffServ environment as proposed in these numerous studies [10], [29], [32], [33], [34], [35], [36]. In this paper, we proposes to detail these numerous proposals and to look at their impact in terms of TCP throughput guarantee over the assured service.

II. Background

A network can be either **over provisioned** or **under provisioned**. Basically, these two cases deal with the excess bandwidth available in the network.

Let $r(i)_{AS}$ be the assured rate allocated to the flow i (*i.e.* in-profile packets throughput), n the number of AS TCP flows in the aggregate at the bottleneck level and C the link capacity. Precisely, this capacity corresponds to a bottleneck link in the network. If a number of i flows cross this link, the total capacity allocated for assured service R_{AS} is:

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

Let C_{AS} be the resource allocated to the assured service. If we have:

$$R_{AS} \leqslant C_{AS} \tag{2}$$

It means an **over provisioned** network. In this case, there is excess bandwidth in the network. If we are in the special case where $R_{AS} = C_{AS}$, this network is called **exactly provisioned**. It means there is enough bandwidth only for the in-profile traffic. In [31], the authors explain some good properties in terms of achieving a differentiation level with such a network.

When:

$$R_{AS} > C_{AS} \tag{3}$$

We are in the context of an **under provisioned** network: there isn't excess bandwidth. This configuration is the worst case for the AS. It means there is available bandwidth for the in-profile traffic only. This service must provide an assurance until the over-subscription case is reached. Afterwards, the service is downgraded since no enough resources are available. As no assurance is provided, this configuration is equivalent to best effort.

In a well-dimensioned network, the inequity (3) should be avoided. When there are losses in the network, it corresponds to the losses of out-profile packets, and not in-profile packets. It means that a light network congestion appears in the network and some out-profile packets must be dropped.

The throughput obtained by a flow depends on the packets dropping policy of the network and how the transport protocol reacts to these losses. TCP reacts to a loss by halving its congestion window and increases this one linearly each time a packet is delivered according to the AIMD principle: *additive increase* and *multiplicative decrease* [22], [15].

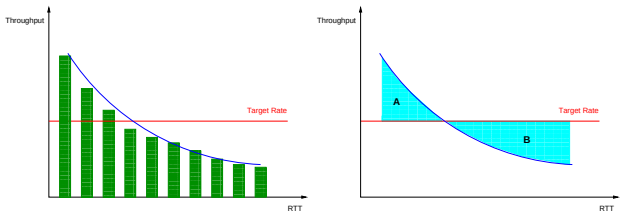
A thorough study of the TCP and UDP behavior in the AF service was undertaken in [39]. The latter showed that when the service has excess bandwidth (compared to the QoS requested), a flow guarantee can be given independently of these five following parameters: the Round Trip Time (RTT), the number of flows, the target rate, the size of the packets, the number of non-reactive flows (such as UDP flows). The distribution of the excess bandwidth between each TCP flow depends on these five parameters. Similar conclusions were presented in [17], [6]. Lastly, Seddigh et Al. [39] defines three criteria concerning equity between TCP and UDP according to the network state. They show that in an over-provisioned network, all TCP and UDP flows can obtain: 1) their target rate; 2) a fair share of the excess bandwidth proportional to their target rate; 3) in an under-provisioned network, all TCP and UDP flows observe a decrease of their throughput. This decrease is proportional to their assured throughput. Another well-known problem is that a large RTT difference between flows influences the desired assured throughput. In the case of identical RTT, each TCP flow in a network shares in a fair manner the available bandwidth. On the other hand, the TCP fair share does not exist if each flow has a different RTT.

In [38], Sahu et Al. demonstrates that:

- the obtained throughput is not proportional to the marked throughput;
- it is not always possible to reach the target rate;
- a flow with a high target rate will never reach its target rate if a flow with a low target rate outperforms its profile;
- in the case describes below, the token bucket marker parameters have no effect on the assured throughput.

Indeed, in the case of an over-provisioned network, when the loss probability of an in-profile packet can be considered has null : $p(i)_{IN} = 0$ and that the loss of an out-profile is not : $p(i)_{OUT} > 0$, if the target rate of a flow verifies the following equation :

$$r(i)_{AS} < \frac{1}{RTT}\sqrt{\frac{3}{2\ p(i)_{OUT}}} \tag{4}$$



(a) TCP flows with different RTT (b) Excess and deficit areas

Fig. 1. TCP throughput as a function of RTT

then the token bucket marker has no effect on the reached throughput [38].

This important result gives a strong limitation to the use of the token bucket marker for TCP conditioning. Indeed, equation (4) shows that a simple token bucket marker is unable to achieve a large range of requested target rate by increasing or decreasing the out-profile marking of a TCP flow. As a result, new marking strategies propose to control the TCP achieved throughput by dynamically choosing a target rate $r(i)_{AS}$ as a function of the loss ratio.

III. Marking strategy big picture

Basically, the principle of the marking strategy is to infer on the TCP throughput of the opportunist flow by controlling the number of losses in their out-profile parts. Following the simple model of the TCP throughput given in [27]:

$$\text{TCP throughput} = \frac{C * MSS}{RTT * \sqrt{p}} \tag{5}$$

With C a constant and p the loss probability and MSS the maximum segment size. In order to increase the loss probability of the opportunist flows, almost all the DiffServ conditioners presented in section V are based on the increase of the out-profile part of theses flows. As a result, the loss probability raises and the TCP throughput decreases as shown in (5). Unfortunately, changing the p value from the equation (5) thanks to a marking strategy is complex. Indeed, it is necessary to evaluate the loss probability of the network and to estimate an RTT for each flow. In order to obtain these keys values, the authors in [8] propose to compute an average loss interval thanks to the method presented in [16] instead of the loss probability and estimate the RTT with a time stamping method. On the other hand, the authors in [19] proposes an RTT-RTO conditioner that is based exclusively on throughput measurement and in addition of RTT estimation, proposes a solution to take into account the TCP timeout in the marking strategy. In order to illustrate the probability marking concept, figure 1 presents the aim of this marking strategy. Figure 1 (a) symbolizes the throughput obtained by ten flows with different RTTs and the same target rate. The smaller the RTT, the higher is the throughput. The flows with a small RTT occupies more bandwidth than necessary as explained by area A in figure 1 (b). Basically, the aim of a marking strategy is to distribute fairly excess bandwidth from area A to area B. In order to summarize, we identify these three important points:

- the TCP throughput is closely related to the packet loss probability, the RTT and the target rate;
- the loss of an out-profile packet is always prejudicial to the TCP assured throughput;
- the loss probability, RTT and RTO estimations are complex to estimate in a passive manner (.i.e in an intermediate node and not at the sender side).

IV. Synthesis of the methods used to obtain a TCP throughput guarantee

In the DiffServ architecture, we can act on three different levels to solve the TCP throughput guarantee problem in the AF service: at the hosts level, at the edge of the network or at the core network inside the Active Queue Management (AQM).

- at the TCP level: solutions suggested raise some deployment problems. First of all, it needs a modification of the TCP code. This is a problem in regard of the numerous diversity and versions of the operating systems and the number of hosts in the Internet. In the context of a DiffServ architecture, the marking is carried out exclusively by the source. In this case, marking is not under the responsibility of the Internet Service Provider (ISP). The checking of the marking by the ISP is not either without raising difficulties of realization. Lastly, this solution is not possible when marking is carried out on the aggregate.
  In [12], an evolution of the TCP congestion control proposes to integrate the marking according to a profile. The solution consists in splitting the congestion into two parts: one for each part of the assured service. The size of each part of the congestion window changes depending on the network state and the observed throughput. Thus, the marking probability is computed from the assured part of the congestion window;
- at the conditioning level: the objective is to copy a marking which is in conformity with the TCP dynamics. Marking is a functionality which should remain under the responsibility of the ISP. Conditioning is an element which is put on cut on the road. It can evolve and move independently of the other components of the DiffServ architecture;
- at the AQM level, new scheduling techniques such JoBS [4] makes it possible to impose flows guarantees in the assured service. These techniques are derived from the proportionality mechanisms introduced by [7]. Another solution would be in the inter-dealing which the AQM with a TCP source can have. The *Explicit Congestion Notification* TCP flag is often used as a complement to control the throughput of a flow in order to limit the packets marked out-profile in the network.

In the following, these three levels of action are compared to the guarantee provided to the flow, the facility of deployment, the scalability. The most tackled solution is on the conditioning level. The prolific literature is an illustration. Nevertheless, we will see that the solution is not obvious and that this level is not inevitably the good one.

V. State of the art of the TCP conditioning

This section gives an extended overview of the concepts used inside the DiffServ conditioners and illustrates the mechanisms chosen to achieve the desired target rate. It will not be interesting to describe all the existing solutions as some of them deal with similar approaches. So, we have selected a set of DiffServ conditioners in order to highlight the concepts used to solve the TCP throughput guaranteed problem and selected some AQM which tries to enforce the service differentiation.

Concerning the TCP marking proposals, they are divided into two families: those which treat the TCP marking with an aggregate profile, and those which treat TCP marking compared to an individual profile. The former aims the equity in addition to the flow guarantee sought by the latter. We will show that most of these solutions are based on the time sliding window algorithm and/or on the two or three token bucket colors marker. Moreover, we will see that the majority of these approaches are based on a weighted probabilistic marking of the excess traffic.

A. Proportional Differentiated Services

This proposal, presented in [7], doesn't deal with a marking strategy as the service differentiation is made at the AQM level. However, since this scheme inspired many proposals in the field of the TCP throughput guaranteed in the DiffServ/AF service, we present in this section the concept introduced in [7]. In this proposal, each packet arriving in the network is marked either in-profile or out-profile according to a token bucket marker. It is on the AQM level, within the router, that the treatment is carried out. Assume two flows with two target rates r_1 and r_2 having an RTT and an identical packets size. On the basis of the TCP equation (5), we have:

$$r_i \leqslant \frac{1.5\sqrt{\frac{1}{3}} * k_i}{RTT * \sqrt{p_i}} \tag{6}$$

With :

- r_i: target rate of flow i;
- k_i: packets size of the i flow;
- p_i: loss probability of the i flow.

following the equation (6), we obtain:

$$\frac{r_1}{r_2} = \sqrt{\frac{p_2}{p_1}} \tag{7}$$

If we compare the number of dropped packets time unit: $d1$ and $d2$, corresponding to the losses throughput, we obtain:

$$\frac{d_1}{d_2} = \frac{r_1 * p_1}{r_2 * p_2} = \sqrt{\frac{p_1}{p_2}} = \frac{r_2}{r_1} \tag{8}$$

It means that the number of dropped packets per time unit must be inversely proportional to the target rate of a flow. This concept of proportionality is the basis of many studies in the TCP throughput guarantee such [4]. It paves the first step on dropping based on the desired target rate and will be derived and enhanced.

B. Qualitative microflows marking [28]

In an other hand, Marco Mellia in [28] study the feasibility of improving the performance of TCP flows in a network with RIO routers by marking packets according to per-flow TCP states at network edges. The key observation is that TCP performance decreases significantly either in the presence of bursty, non adaptive cross-traffic or when it operates in the small window regime, i.e., when the congestion window is small. This is because bursty losses or losses during the small window regime may cause retransmission timeouts (RTOs), which will ultimately result in TCP entering the slowstart phase. The objective of the TCP-aware marking algorithm is then to selectively mark packets in order to reduce the possibility of TCP entering these undesirable states. Marco Mellia exploits the fact that IN packets are delivered with a very high probability. Thus selectively marking packets as IN allows TCP to exit as fast as possible from the undesirable states. In order to take into account these states, Marco Mellia in [28] proposes:

- to mark the first several packets of the flow. This will protect the first packets against loss, and it will allow TCP to safely exit the initial small window regime;
- to mark several packets after an RTO occurs. The purpose of this is to make sure that the retransmitted packet is delivered with high probability, and that TCP sender exits the small window regime which follows the Slow Start phase entered after the RTO event;
- to mark several packets after receiving three duplicate acknowledgements. The present idea is to protect the retransmitted packet in order to allow TCP to come out the Fast Recovery phase without losing other packets.

This marking scheme is qualitative as it can improve the throughput of long lived TCP flows up to 20%, and reduce the completion time of short lived TCP flows by half according to the author. The main disadvantage of this approach is that it needs to know the TCP window size and the slow-start threshold (*ssthresh*). So, it needs to operate to a modification of the TCP stack in order to use this algorithm.

This algorithm improves a target rate but does not give any guarantee about the target rate requested.

C. Marking schemes based on the Time Sliding Window algorithm

Several algorithms of this type were proposed to work with the AS service. The Two Rate Three Color Marker (TRTCM) [21] based on a token bucket estimator algorithm and the Time Sliding Window Three Color Marker (TSW3CM) [11] based on an average throughput estimator: the Time Sliding Window (TSW) [5]. In these markers, two rates are defined: an assured rate called Committed Information Rate (CIR) and a maximum rate: the Peak Information Rate (PIR) used in case of excess bandwidth.

The main difference between these two markers is the way they mark the packets. Even if they take each one in argument the assured rate: $r(i)_{AS}$, at the opposite of the TRTCM, the TSW3CM applies a probabilistic packets marking. Indeed, the TRTCM marks a packet out-profile if this one is not in the

profile defined by the *token bucket* parameters: (r, b). On the other hand, the TSW3CM marks a packet out-profile with a probability as a function of the average rate estimated by the TSW and the PIR and CIR. The TSW3CM gives better results than a simple TRTCM as its marking scheme describes better the TCP traffic. Starting with this point, a lot of others marking strategies proposed to improve this static probability marking scheme.

We define these enhanced marking strategies as adaptive marking. In the following, we present in details enhanced proposals based on the TRTCM and TSW3CM.

VI. Towards an enhanced TCP conditioner: the Adaptive Marking

The adaptive marking proposes to improve the TRTCM or the TSW3CM conditioners by changing dynamically the marking rate. It means that the target rate of the marker evolving in the time as a function of the network conditions and the throughput obtained by a conditioned flow. We give below an overview of three majors adaptive algorithms.

A. Adaptive marking with dynamic target rate

In [41], Yeom and Reddy present a marking scheme for a TCP flow inside an aggregate. This scheme is based on a mathematical TCP model defined in [42]. This model is given in equation (9).

$$b_i = \frac{3}{4}m_i + \frac{3k_i}{4RTT}\sqrt{\frac{2}{p_i}} \tag{9}$$

assume that:

$$b_i = \frac{3}{4}m_i + \epsilon_i \tag{10}$$

With b_i: throughput of the i flow; k_i: its packets size ; p_i: its loss probability and m_i: its initial target rate value which corresponds to the $r(i)_{AS}$ token bucket marker parameter. This equation gives the throughput of the flow as a function of the token bucket marker parameters used. Thanks to the equation (10), Yeom and Reddy propose to act on the marking process as a function of the following states:

1) if $b_i \leqslant \frac{3}{4}m_i + \epsilon_i < r(i)_{AS}$: in this state, the flow observes an oversubscribed network, and some in-profile packets are lost. Thus, the marker reduces m_i so that b_i is maintained to be higher than $\frac{3}{4}m_i$ to avoid wasting resources;
2) if $\frac{3}{4}m_i+\epsilon_i < b_i < r(i)_{AS}$: in this state, the flow does not reach its target. Since the network is not oversubscribed, b_i can be increased by increasing m_i. Thus, the marker increases m_i of that flow if resources are available;
3) if $r(i)_{AS} \leqslant b_i$: in this state, the flow already achieved its target. Thus, the marker reduces m_i to avoid wasting resources.

In [3], Chait et Al. present a similar concept with a dynamic token bucket marker configuration. The constituent components of this design include two-color token bucket edge markers coupled with a two-level AQM controller embedded in the core routers. The interactions between TCP flows and these components are managed by a proportional-integral controller (PI) which is a control loop feedback mechanism widely used in automatic and control systems. The PI controller attempts to adjust the target rate of the TCP flows as a function of the information returned by the network and the current TCP achieved throughput.

These mechanisms, based on a dynamic target rate parameter, do not provide a fair sharing of excess or lack of bandwidth when the network is respectively over-subscribed or under-subscribed. Indeed, the allocation is determined by the dynamics of the TCP congestion control mechanism. In the following sections, we present an extension of this dynamic approach but with a fair sharing of the excess bandwidth.

B. Adaptive marking based on memorisation

This conditioning based on memorisation was proposed in [24]. The principle of marking inherits from the TSW3CM algorithm except that the marking probability is weighted by the use of a variable memory. This variable keeps a history of the average throughput estimated by the TSW algorithm of the TSW3CM marker. It is used for indirectly detecting a variation of the TCP window size or an RTT variation of the conditioned flow. This method improves the fair sharing of the excess bandwidth between the flows whatever their RTT or their target rates.

C. Adaptive marking based on a marking probability

In the previous section, we saw that Yeom et Al. used a mathematical model of a TCP flow in a DiffServ network in order to obtain a theoretical value of the TCP throughput. They use this model to act on the marking rate of the token bucket marker. The advantage of this approach is that it can operate with a conditioning method based on the microflow or the aggregate level. However, this solution is not efficient in all network conditions as the model doesn't take into account all the network and TCP parameters (such as the TCP timeout value, the RTT variation, ...). On the other hand, in [8], Gendy showed that it exists a duality between the marking based on the evaluation of the throughput and the loss probability. As a result, Gendy proposes to find the best marking rate according to a more accurate TCP model proposed in [30]. The scheme has a better feedback of the network state than [41] since it takes into account and evaluates all the parameters of this complex and accurate model. However, this scheme is strongly limited to the accuracy of the passive measurements used to feed this equation.

The equation used is given in [30] (we denote $F()$ this equation, see appendix A for details) and takes into account: p: the packet loss probability; W_{max}: the maximum TCP window size; RTT: the round trip time; RTO: the TCP timeout value; MSS: the maximum segment size; and returns the throughput X with:

$$X = F(p_{OUT}, Wmax, RTT, RTO, MSS) \tag{11}$$

Basically, if we assume that we are in a well-provisioned network (i.e. equation (2) is true). The loss probability of a

packet is corresponding to the loss probability of an out-profile packet: p_{OUT} since the loss probability of an in-profile packet should be near zero: $p_{IN} \simeq 0$. The main difficulty is to reverse this equation in order to obtain p as a function of X:

$$p_{OUT} = F(X, Wmax, RTT, RTO, MSS) \quad (12)$$

by changing X with the target rate of a flow (r_i) we obtain:

$$p_{OUT} = F(r_i, Wmax, RTT, RTO, MSS) \quad (13)$$

The idea is to solve this equation in order to obtain the optimal marking rate $r(i)_{AS}$ of the token bucket parameter.

Although this proposal is certainly the most achieved one, the complexity induces by the passive measurements used to feed the equation, the need of a strong accuracy and the per-flow monitoring involved in the conditionning process are the main barrier to the deployement of such mechanism. In the following section VII, we will see how other proposals have overcome the problem of assessing the network by using the Explicit Congestion Notification (ECN) mechanism [37].

VII. Dealing with ECN feedbacks

Following the difficulty to feed the parameters used to characterize the TCP flows, recent approaches propose to use ECN feedbacks in order to assess the congestion level in the network. The main idea is to use either the number of ECN marked packets or specific information carried inside the feedback packet as a congestion indication level.

A. Proportional Bandwidth Allocation

In [31], Park and Choi analyze the steady state throughput of TCP flows in a differentiated network. They show that current DiffServ networks are biased in favor of those flows that have a smaller target rate, which results in an unfair bandwidth allocation. However, they demonstrate when the network is exactly-provisioned, there is no bias in favor of an aggregate that has a smaller target rate. So, they propose to adjust the target rate of the token bucket markers in order to match the bottleneck capacity as a function of the network congestion level. In other words, the sum of each marking rates should be equal to the bottleneck capacity, this result in having almost in-profile traffic in the network. This approach is original since it deals with two new concepts. First, the amount of ECN marked packet drives the target rate value and second, having a network exactly-provisioned should reduce TCP sources oscillation as the number of dropped packets decreases.

Unfortunately, this solution is strongly linked to the RTT of each flow. In their paper, the authors evaluate their solution with RTT equals or in the same order of magnitude. As a consequence, this solution cannot be generalized to a multi-domain network with a large range of RTT values.

B. AIMD Penalty Shaper

Finally, as opposed to the marking strategy adopted by new conditioners, we have proposed a delay based shaper [26]. This shaper applies a delay penalty to a flow if there are out-profile packets losses in the network and if it outperforms its target rate. The basic idea is that the penalty is a function of the out-profile packet losses. Instead of raising the p value, from equation (5), of the most opportunist flow, the AIMD Penalty Shaper raises a delay penalty to the flow. It results in a growth of the RTT. Mathematically, as shown in (5), increasing RTT value is similar to increasing p value in terms of TCP throughput. In [40], the authors have shown that limiting out-profile packets is a good policy to achieve a target rate. Indeed, by limiting packets dropping we avoid TCP retransmission. This is an efficient solution to optimize the bandwidth usage. Thus, the goal is to reduce out-profile losses by applying a delay penalty to the flows that are the most opportunist in the network. Therefore, when a RIO[2] [5] router in the core network is dropping out-profile packets, it marks the ECN flag [37] of the in-profile packets enqueued in the RIO queue. In a well-dimensioned network, there is no in-profile packet loss. Then, the edge device can be aware that there is a minimum of one flow, or set of flows, which are opportunists in the network. This opportunist traffic is crossing the same path. The edge device evaluates its sending rate thanks to a Time Sliding Window (TSW) algorithm [11]. If its sending rate is higher than its target rate, it considers that its traffic may be opportunist. Then, it applies a penalty to the incoming traffic until the network returns that there are out-profile packets losses. This penalty allows a raise of the RTT and consequently, decrease the TCP throughput. In [25], the authors choose to use an AIMD penalty instead in order to decrease rapidly the throughput [14]. If there is no loss anymore, the penalty decreases linearly and the TCP throughput raises. This principle follows the TCP congestion control. The main advantage of this solution is that the conditioning can be made on flows with similar RTTs (i.e. in the same order of magnitude). Moreover, this solution doesn't depend on the complex problem of RTT estimation necessary to the functioning of the conditioners presented before.

VIII. Discussion

Among the multiple conditioning schemes presented, it resides two main classes. First, the quantitative conditioning class which includes: equation based marking [30], [8], [41] ; memory-based marking [24] ; TSW-based marking [21], [5] and penalty shaper conditioning [26], [25]. Second, the qualitative conditioning class with the marking scheme inspired by [28].

If these conditioning mechanisms work well theoritically or in simulation, the scalabity of most of these proposals is not proofed in particular in case of real world experiments with cross-traffic and this case can strongly decrease the efficiency of many conditioners. Furthermore, even if the

[2]RED with IN and OUT

DiffServ architecture is based on per-flow conditioning, it is obvious that for an ISP, it will be more easier to profile a client emitted TCP aggregate than every single TCP flows of a TCP source. Indeed, the client is typically a source domain as defined in [1] that communicates with another source domain within a DiffServ core network. A good conditioner should provide a service differentiation between two source domains, on a set of TCP flows, based on its marking profile.

In the context of the use of ECN feedbacks, to the best of our knowledge, it exists no analytical study in order to assess the network congestion level as a function of the amount of ECN traffic. A recent study from Bob Briscoe [2] proposes to extend the ECN protocol in order to carry a better and truthful prediction of the congestion of the path. However, there is still no rule allowing to compute in an accurate manner the exact congestion level of a network following the number of RED-ECN traffic marked.

Another widespread idea is to claim that the over-provisioning is the best solution allowing the DiffServ/AF service to work without any kind of improvements. However, this method has a cost and even in case of over-provisioned network, there is no better guarantee to reach and maintain a negotiated target rate. The question remains the same: how to size this provisionning at a low cost? Indeed, there is no exact method to determine efficiently the necessary excess bandwidth size.

In 2001, in an interesting and well-known unpublished technical report [13], Victor Fioriu and Al. wrote that: "*the possibility to use TCP in order to provide differentiated QoS is under question and replacing the TCP congestion mechanism in the context of QoS networks is currently an open research area*". Nowadays, new approaches have proposed to design specific DiffServ transport protocols (such as [9], [18]) able to be aware of the negotiated QoS thanks to a QoS congestion control and cross-layer mechanisms allowing the transport protocol to be fully aware of the target rate negotiated between the application an the network provider. Following these proposals which have demonstrated their complete compliances with DiffServ QoS network architecture such as the EUQoS[3] architecture [23], the question to use TCP as QoS transport protocol in order to provide services guarantees seems outdated.

APPENDIX

A. TCP Model

$$F(p_{OUT}, W_{max}, RTT, RTO, MSS) =$$

$$\begin{cases} MSS\dfrac{\frac{1-p}{p}+W(p)+\frac{Q(p,W(p))}{1-p}}{RTT(\frac{b}{2}W(p)+1)+\frac{Q(p,W(p))F(p)To}{1-p}} & \text{if } W(p) < W_{max} \\ MSS\dfrac{\frac{1-p}{p}+W_{max}+\frac{Q(p,W_{max})}{1-p}}{RTT(\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2)+\frac{Q(p,W_{max})F(p)To}{1-p}} & \text{otherwise} \end{cases} \quad (14)$$

where

$$W(p) = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \frac{2+b}{3b}^2} \quad (15)$$

$$Q(p,w) = min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3})}{1-(1-p)^w}\right) \quad (16)$$

$$F(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \quad (17)$$

[3]http://www.euqos.eu/

b : is the average number of packets acknowledged by an ACK, usually 2.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request For Comments 2475, IETF, December 1998.

[2] Bob Briscoe, Arnaud Jacquet, Alessandro Salvatori, Martin Koyabe, and Toby Moncaster. Re-ECN: Adding accountability for causing congestion to TCP/IP. Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-04.txt, IETF, July 2007.

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

[4] N. Christin, J. Liebeherr, and T. Abdelzaher. A quantitative assured forwarding service. In *Proc. of IEEE INFOCOM*, volume 2, pages 864–873, New York, NY, June 2002.

[5] D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[6] Jos Ferreira de Rezende. Assured service evaluation. In *Proc. of IEEE GLOBECOM*, pages 100–104, Rio de Janeiro, Brasil, December 1999.

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

[8] M.A. El-Gendy and K.G. Shin. Assured Forwarding Fairness Using Equation-Based Packet Marking and Packet Separation. *Computer Networks*, 41(4):435–450, 2002.

[9] Ernesto Exposito and Michel Diaz and Patrick Sénac. Design Principles of a QoS-oriented Transport Protocol. In *IFIP International Conference on Intelligence in Communication Systems*, Bangkok, November 2004.

[10] EuQoS. End-to-end quality of service support over heterogeneous networks. http://www.euqos.org/.

[11] W. Fang, N. Seddigh, and AL. A Time Sliding Window Three Colour Marker. Request For Comments 2859, IETF, June 2000.

[12] W. Feng, Dilip Kandlur, Debanjan Saha, and Kang S. Shin. Adaptive Packet Marking for Providing Differentiated Services in the Internet. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, October 1998.

[13] V. Firoiu, J. Le Boudec, D. Towsley, and Z. Zhang. Advances in internet quality services.

[14] S. Floyd. Congestion control principles. Request For Comments 2914, IETF, September 2000.

[15] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[16] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based Congestion Control for Unicast Applications. In *Proc. of ACM SIGCOMM*, pages 43–56, Stockholm, Sweden, August 2000.

[17] M. Goyal, A. Durresi, R. Jain, and C. Liu. Effect of number of drop precedences in assured forwarding. In *Proc. of IEEE GLOBECOM*, pages 188–193, 1999.

[18] Emmanuel Lochin Guillaume Jourjon and Patrick Senac. Design, implementation and evaluation of a qos-aware transport protocol. *To appear in Computer Communications*, 2008.

[19] Ahsan Habib, Bharat Bhargava, and Sonia Fahmy. A Round Trip Time and Time-out Aware Traffic Conditioner for Differentiated Services Networks. In *Proc. of the IEEE International Conference on Communications - ICC*, New-York, USA, April 2002.

[20] J. Heinanen and R. Guerin. A Single Rate Three Color Marker. Request For Comments 2697, IETF, September 1999.

[21] J. Heinanen and R. Guerin. A two rate three color marker. Request For Comments 2698, IETF, September 1999.

[22] Van Jacobson. Congestion avoidance and control. In *Proc. of IEEE SIGCOMM*, pages 314–329, Stanford, CA, August 1988.

[23] Guillaume Jourjon, Emmanuel Lochin, Laurent Dairaine, Patrick Senac, Tim Moors, and Aruna Seneviratne. Implementation and performance analysis of a QoS-aware TFRC mechanism. In *Proc. of IEEE ICON*, Singapore, September 2006.

[24] K. Kumar, A. Ananda, and L. Jacob. A Memory based Approach for a TCP-Friendly Traffic Conditioner in DiffServ Networks. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, Riverside, California, USA, November 2001.

[25] Emmanuel Lochin, Pascal Anelli, and Serge Fdida. AIMD Penalty Shaper to Enforce Assured Service for TCP Flows. In *Proc. of the International Conference on Networking - ICN*, La Reunion, France, April 2005.

[26] Emmanuel Lochin, Pascal Anelli, and Serge Fdida. Penalty shaper to enforce assured service for TCP flows. In *IFIP Networking*, Waterloo, Canada, May 2005.

[27] M. Mathis, J. Semke, and J. Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), 1997.

[28] Marco Mellia, Ion Stoica, and Hui Zhang. TCP-aware packet marking in networks with diffserv support. *Computer Networks*, 42(1):81–100, May 2003.

[29] G. Verticale P. Giacomazzi, L. Musumeci. Transport of TCP/IP traffic over assured forwarding IP-differentiated services. *IEEE Network*, (5):18–28, September 2003.

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

[31] Eun-Chan Park and Chong-Ho Choi. Proportional Bandwidth Allocation in DiffServ Networks. In *Proc. of IEEE INFOCOM*, Hong Kong, March 2004.

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

[33] Global communication architecture and protocols for new qos services over ipv6 networks. http://www.laas.fr/GCAP/.

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

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

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

[37] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to ip. Request For Comments 3168, IETF, September 2001.

[38] Sambit Sahu, Philippe Nain, Christophe Diot, Victor Firoiu, and Donald F. Towsley. On achievable service differentiation with token bucket marking for TCP. In *Measurement and Modeling of Computer Systems*, pages 23–33, 2000.

[39] N. Seddigh, B. Nandy, and P. Pieda. Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network. In *Proc. of IEEE GLOBECOM*, page 6, Rio De Janeiro, Brazil, December 1999.

[40] Ikjun Yeom and Narasimha Reddy. Realizing throughput guarantees in a differentiated services network. In *Proc. of IEEE International Conference on Multimedia Computing and Systems- ICMCS*, volume 2, pages 372–376, Florence, Italy, June 1999.

[41] Ikjun Yeom and Narasimha Reddy. Adaptive marking for aggregated flows. In *Proc. of IEEE GLOBECOM*, San Antonio, Texas, USA, November 2001.

[42] Ikjun Yeom and Narasimha Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Transactions on Networking*, 9(1):31–46, 2001.

Transport Congestion Events Detection (TCED): Towards Decorrelating Congestion Detection from TCP

Pascal Anelli
Université de la Réunion; LIM;
France
pascal.anelli@univ-reunion.fr

Emmanuel Lochin
CNRS; LAAS;
Université de Toulouse; ISAE;
France
emmanuel.lochin@isae.fr

Fanilo Harivelo
Université de la Réunion; LIM;
France
fanilo.harivelo@univ-reunion.fr

Dino Martin Lopez Pacheco
I3S; Université de Nice;
France
dino.lopez@unice.fr

ABSTRACT

TCP(*Transmission Control Protocol*) uses a loss-based algorithm to estimate whether the network is congested or not. The main difficulty for this algorithm is to distinguish spurious from real network congestion events. Other research studies have proposed to enhance the reliability of this congestion estimation by modifying the internal TCP algorithm. In this paper, we propose an original congestion event algorithm implemented independently of the TCP source code. Basically, we propose a modular architecture to implement a congestion event detection algorithm to cope with the increasing complexity of the TCP code and we use it to understand why some spurious congestion events might not be detected in some complex cases. We show that our proposal is able to increase the reliability of TCP NewReno congestion detection algorithm that might help to the design of detection criterion independent of the TCP code. We find out that solutions based only on RTT (*Round-Trip Time*) estimation are not accurate enough to cover all existing cases. Furthermore, we evaluate our algorithm with and without network reordering where other inaccuracies, not previously identified, occur.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Transport Mechanisms

General Terms

Transport Protocol

Keywords

TCP, Congestion Event, Measurements

SAC'10 March 22-26, 2010, Sierre, Switzerland.

1. INTRODUCTION

TCP has the capability to adapt its sending throughput to the changing bandwidth available following the principle described in [11]. TCP considers a loss of a segment as a congestion in the network. A congestion event (or loss event) corresponds to one or several losses (or, in the context of ECN (*Explicit Congestion Notification*) [19]: at least one ACK (*acknowledgment)* packet with an ECN-echo) occurring in one TCP window during one current RTT period [6].

TCP congestion events play a role in terms of throughput performance as the congestion event involves a multiplicative decrease from the source. It is also important to clearly distinguish the loss ratio from the congestion ratio. Indeed, the number of losses during one RTT is not taken into account to characterize a congestion event as this number only impacts on the recovery time period.

TCP is strongly sensitive to spurious timeout [21] which trigger spurious retransmissions and result in a throughput decrease. Spurious timeout occurs when a non lost packet is retransmitted due to a sudden RTT increase (handover, route fluttering, network reordering, ...) which implies an expiration of the retransmission timer [18] set with a previous, and thus outdated, RTT value. This effect is known to be the root cause of spurious retransmission [20]. Several research work have raised this problem [15, 21, 3, 22].

In this paper, we propose a novel transport layer architecture where the congestion event detection algorithm is realised independently of the TCP code and detail the essential brick of this proposal: the congestion events detection mechanism. We aim to illustrate the feasibility of this concept by demonstrating that we can either obtain similar performances or also improve the accuracy of this detection outside the TCP stack. The main idea is to determine CE (i.e. the congestion detection) which impact on the TCP flow performance by monitoring the TCP flow itself. The principle is to obtain a detection system, at the edge of a network or at the sender-side which analyses the TCP behaviour through the observation of both data packets and acknowledgments paths.

We implement this Implicit Congestion Notification (ICN) algorithm inside a framework that we call TCED (*Transport*

Congestion Events Detection). Thus, TCED can be used either at the border of an autonomous system or conjointly within a TCP stack as a sublayer. ICN allows also to better understand and investigate the problem of congestion events estimation as well as proposing possible solutions to suppress inaccuracies of the current TCP loss-based algorithm. Following more exhaustive measurements, we show that an external congestion event detection is thus possible and present in section 2 the rationale of this design. Furthermore, we have identified that solution only based on RTT estimation are not accurate enough to cover all existing cases (*e.g* in particular retransmissions triggered at the begin of a connection). These results are provided in Section 4, just after the details of the algorithm (Section 3).

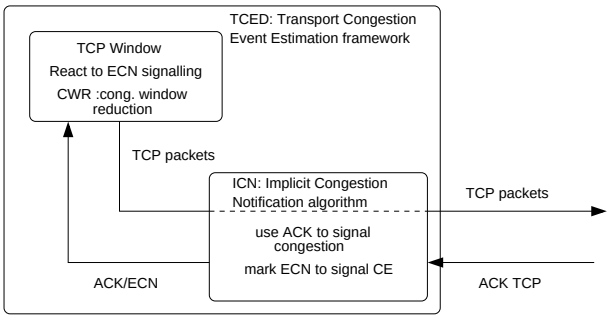


Figure 1: Decorrelating Congestion Detection from the Transport Layer.

2. MOTIVATION TO BUILD A STAND-ALONE TCP CONGESTION EVENTS ALGORITHM

These last years, the number of TCP variants and minor extensions have greatly increase (see `[tcpm]`, `[iccrg]` and `[tsvwg]` IETF mailing lists). Some of them are deeply specialized to specific networks such as wireless LAN [16] or satellites links [5] while others focus on high speed networks [6, 23]. To date, except the historical and generic TCP Newreno/SACK (*Selective ACKnowledgment*) variant, there exists no universal TCP protocol able to perform indifferently over any kind of networks. The direct observable consequence of these many proposals is that **TCP source code is gaining in complexity** and that minor extensions proposed, such as for instance F-RTO (*Forward Retransmission TimeOut-recovery*) [21], do not help in terms of clarification of the source code. Furthermore, some improvements might be linked to a specific TCP version and cannot be deployed in the common TCP source code. Thus, the relevance of the OSI model is under question and in a recent paper [9], the authors argue that the transport layer should be now sliced in three sub-layers to cope with new network characteristics and the inherent complexity of the source code. All these reasons motivate the present study which aims at decorrelating the congestion events detection from the transport layer as presented in figure 1. In a sake of clear software engineering development, the main goal of this architecture is to simplify the task of kernel developers as well as improving TCP performances. Indeed, such architecture greatly facilitates protocol evolution and permits incremental rollout of congestion detection improvements. Finally, this scheme opens the door to another way to react to congestion by enabling ECN emulation at end-host. In this case, ICN emulates ECN marking to imply a congestion window reduction (see figure 1) with the same philosophy than in [4] where the authors enable AQM (*Active Queue Management*) emulation at end-host.

The target is to switch between the classical transport layer with the TCED layer architecture as illustrated figure 2.

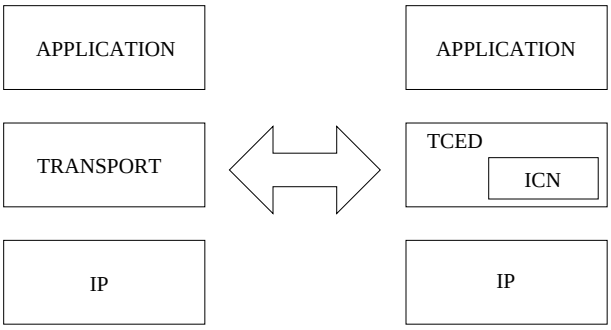


Figure 2: Re-architecture of the transport layer.

3. IMPLICIT CONGESTION NOTIFICATION (ICN) ALGORITHM

This section presents the design of our proposal and details the algorithm.

3.1 Design hypothesis

The design hypothesis is related to the localization of the Congestion Event (CE) estimator (i.e. from the source or from a node immediately connected to the source before the edge router). In this case, the estimator must be on a symmetrical path (ACK and data segments are received by the node) and the resulting RTT estimated is thus similar to the TCP connection. As our estimator operates to a live analysis (and not over past captured traces), only traffic at the sender side must be analyzed.

As TCP is a reliable transport protocol, lost data are obviously retransmitted. Thus, a CE can be identified from the TCP retransmitted packets. However, neither all of the retransmissions do not always indicate a loss, nor all losses do not always signal a CE:

- When the TCP retransmission timer expires, TCP triggers a *Go Back N* recovery procedure which could lead to retransmissions of packets effectively received (*i.e.* spurious retransmissions). Furthermore, TCP can consider losses following network packets reordering or following a significant increase of the RTT (*e.g.* in case of vertical handoff[1]). In this particular case, known as spurious timeout, the ACK get back too late to reset the retransmission time. These false losses identifications strongly impact on the TCP overall performances in terms of achieved throughput;

[1]In a mobile context, a vertical handoff occurs when a mobile node is moving from a low delay network such as wireless LAN to a high delay network such as UMTS/GPRS. Due to the sudden RTT increase, spurious retransmissions might occur.

- Each loss does not have to be taken into account when identifying a CE. Indeed, when multiple losses occur in a single window, only the first loss is needed to identify a CE and other losses inside the same window must be ignored. To realize this, we need to be able to detect the first loss and the size of the sending window.

The ICN algorithm allows to determine which loss affects the TCP flow from the capture of TCP segments.

3.2 Interpreting CE

As previously explained in the introduction, a congestion event is defined as a set of losses occurring on a TCP window which involves a TCP congestion adaptation during an RTT. As a data window is emitted during an RTT, the estimation of the window size allows to identify data transmitted during one RTT given. When there is a retransmission, the bottom of the window is set on the retransmitted segment of data while the top of the window corresponds to the highest sequence number data sent. Then, all retransmitted packets between these two bounds are considered as belonging to the same CE. Indeed, a CE starts when a loss occurs and stops when the top of the data window during this congestion notification is acknowledged.

Detecting only retransmitted packets is not enough to identify a loss. Indeed, we have to be sure that a retransmission is not due to a TCP error. This case must be taken into account to avoid an overestimation of the CE over path where high RTT variations and reordering occur.

In [20] and [2], the authors classify as unnecessary (spurious) retransmission, those acknowledged in a delay lower than $\alpha * RTT_{min}$ (where RTT_{min} is the lowest RTT measurement and $\alpha = \frac{3}{4}$). If the next new ACK arrives in delay lower than $\alpha * RTT_{min}$ after the retransmission, then it means the ACK was already in transit when the retransmission occurred, and the timeout was spurious. This delay is a key value in the detection process. Indeed, when the delay is very small, this can lead to interpret unnecessary retransmissions as losses. On the contrary, when the delay is close to the current RTT, a new ACK could be received and the retransmission would be considered as unnecessary. In this configuration, the number of CE can be underestimated. As above, the principle to identify retransmission is based on a waiting delay T before validation. Obviously, this method introduces an additional delay which can be considered as a trade-off between the reliability and swiftness of TCP in terms of losses detection. In this context, it exists a delay between the network congestion and its detection by ICN. The CE detection is twofold. First, the identification and then, the classification of a retransmission as a lost.

In brief, to obtain an accurate CE estimation we need:

- to accurately estimate the RTT to size the delay to validate a loss;
- to take into account the TCP window size to distinguish the loss triggering a CE from the loss occurring during a CE;
- to identify spurious retransmissions that should not be identified as lost packets;
- to manage multiple data retransmissions. This might append during severe congestions. In this case, multiple CE occur; we denote such situation of re-congestion.

3.3 ICN algorithm

Starting from the observation of the data segments and the ACK, we identify each CE from each TCP connection with a state machine. This state machine (given in Figure 3) identifies the control congestion phase and classifies retransmissions as spurious or not. TCP congestion control reacts following binary notification feedbacks allowing to assess whether the network is congested or not. The state machine used enters in two states as a function of these notification feedbacks:

1. the *normal state* which characterizes a TCP connection without losses. Following Karn [12] algorithm, we can estimate the RTT in this state, knowing that for each emitted segment, the delay is computed until the corresponding ACK is received. As soon as this estimation is done, the process restarts for the next transmitted segment. This RTT estimation is used to size the validation delay (T); In the context where connections start over a severely congested network, the losses of segments prevent to realize an RTT estimation. Then, the initial value is set arbitrarily and is set to the same value than the retransmission timer: 3 seconds [18];

2. the *congestion state* which starts from the loss of the first window data segment. Each time ICN enters in this state, a CE is counted for the TCP connection.

Two others temporary states are added between these both states. These states aim at identifying spurious retransmissions with the help of the validation delay as previously explained. Finally, when the top of the window (denoted *recover*[2] in Figure 3) is acknowledged, ICN enters in the normal state.

In the case where it is supposed that a retransmission of a packet has been lost, and such a packet is retransmitted again, ICN enters into a *re-congestion* state. Once in the re-congestion state, if the validation delay T expires before *recover* is acknowledged, ICN comes to the *congestion* state and the CE counter is increased in one unit.

In the context of using ECN flag, congestion are not deduced only from losses. However, for this kind of connection we need to extend our algorithm to analyze this ECN signalization. Otherwise, ICN will under estimate the CE ratio which affects each TCP connection. A recent study proposes to help congestion events detection using ECN marking [22]. However and to the best of our knowledge, the deployment of ECN inside the Internet remains marginal. Indeed, the authors in [17] show that ECN flag is used only by 2.1% hosts in 2004 and all current systems do not enable ECN by default[3]. As a result, we choose to study this particular case in a future work.

It is important to note that ICN does not manage error control which remains under the responsibility of TCP. If some improvements concerning the retransmission decision are introduced inside TCP, as ICN does not depend on the error control, our proposal remains valid. As a result, ICN is a generic algorithm which does not depend on the TCP version used.

[2]Equivalent to the *recover* variable of TCP NewReno

[3]See Sally Floyd's ECN page for further details `http://www.icir.org/floyd/ecn.html`

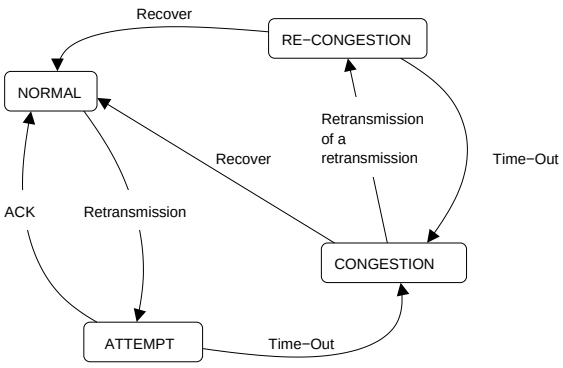


Figure 3: ICN state machine.

4. VALIDATION

We use the ns-2 simulator to estimate the notification error ratio as a function of the real CE. Our simulation model is able to apply different packet drop rates, different levels of statistical multiplexing and to introduce a level of packet reordering. The proposed scenario is motivated by the need to estimate ICN accuracy in the presence of spurious retransmissions and complex pattern packet drops. We aim at understanding the behaviour of ICN qualitatively rather than quantitatively. It means that we are not interested in determining the exact inaccuracy value through a statistical analysis but to determine a global accuracy trend.

The experiments are done over a simple dumbbell topology. A recent paper [13] provides a guideline to make a simulation model to evaluate TCP congestion control extensions. The model used afterwards follows these recommendations. We model the network traffic in terms of flows or sessions. Each flow corresponds to a HTTP request supported by a TCP connection. The *link load* is defined as follows:

$$\rho = \frac{\lambda E[\sigma]}{C}, \tag{1}$$

with C the bottleneck capacity. The traffic demand, expressed in bit rate, is the product of the flow rate arrival λ with the average flow size $E[\sigma]$. A reasonable fit to the heavy-tail distribution of the flow size observed in practice is provided by the Pareto distribution. The shape parameter is set to 1.3 for all simulations in the paper. As all flows are independent, the flow arrivals are modeled by a Poisson process. The load is changed by varying the arrival flow rate. Thus, the congestion level increases as a function of the load.

The load introduced in the network experiences different RTT (ranging from 59 to 250ms). To remove synchronization in TCP feedback and the phase effect, a traffic load of 10% is generated in the opposite direction. Measurements are saved after a "warm-up time" (*i.e* in the steady state) and the simulation duration corresponds to the reference flow duration. The bottleneck link capacity is set to 10$Mbps$. All others links have a capacity of 100$Mbps$. When the transient phase is finished, a long live flow of 400 packets is started, which represents the reference flow on which the ICN algorithm is applied. The minimum RTT experienced by the reference flow is around 100ms.

A packet reordering is applied on the reference flow such that randomly selected packets are randomly delayed. The network reordering is done after the bottleneck to maintain a same reorder rate for each network load. We adopt the delay distribution given in [24]. Delays are chosen from a normal distribution. The mean value is set to 50ms with a standard deviation of 16ms. Thus, the most chosen packets are delayed with values ranging from 0 to 100ms. The reorder rate used is 3.5%.

We make no claim about how realistic our background traffic is. We only want to reduce any simulation anomaly. The main objective of this model is to apply various and exhaustive congestion patterns to the analyzed flow. Figure 4 shows the conditions applied to the reference flow. The drop ratio results from packet drops of the background traffic without the reference flow. The reorder ratio is given from the measurements done on the reference flow.

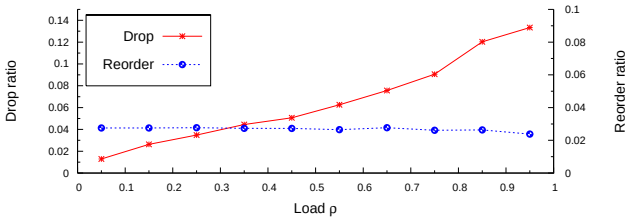


Figure 4: Scenario applied to the reference flow.

The accuracy of the CE detection is given by the inaccuracy parameter denoted ϵ and defined as follows:

$$\epsilon = \frac{N_{spu} + N_{und}}{N_{real} + N_{spu}} \text{ with } \epsilon \in [0, 1]. \tag{2}$$

With N_{spu}: the number of spurious CE identified either by ICN or TCP, N_{und}: the number of undetected real CE not identified by ICN or TCP and N_{real}: the number of effective (real) CE in the network. A real CE is deduced from the packets lost analysis (done *a posteriori*) occurring in the network with the TCP window value when the lost is detected. When $\epsilon = 1$, it means that all detections are spurious and the more ϵ tends to 0, the more real congestions are detected.

We are not interested in determining the exact number of TCP losses as the LEAST algorithm presented in [1]. We aim at detecting when TCP congestion events occur in the network. In this work, we choose to compute a Loss Event Ratio (LER) (also called the congestion event rate [7]) defined as the ratio between the number of congestion event and the number of sent packets.

4.1 Choice of the validation delay

An important value which impacts on the ICN accuracy is the validation delay T. Three possible values are proposed to size T: RTT_{min}, $SRTT$ (the exponential mean of the RTT estimations) and RTT (the last RTT estimation). These values are adjusted according to a fraction called α as previously explained in Section 3.2). The goal is to find out the appropriate α that gives the best results in terms of CE identification.

We test the following α: $0.25, 0.5, 0.75, 1$. In the context of re-congestion, the validation delay will be tested with and without fraction. Fig. 5 gives the average inaccuracy over all possible combinations for a ρ background traffic ranging

from 0.05 to 0.95 with a step of 0.1. All flows use TCP NewReno and the queue management for both routers is DropTail. When α is used, no validation delay is set. The inaccuracy in this case is the same as TCP. The α values are evaluated in growing order for each RTT base values. The dashed line in Fig. 5, represents the case where $\alpha = 1$ and when a re-congestion is detected.

As expected in our scenario, we can see that the network reorder introduces significantly errors in the congestion detection. The strong RTT variations produce spurious retransmissions. In this scenario, RTT_{min} is a good candidate to detect false CE notification. Near 0.25, ICN fails to detect spurious CE. Inversely when the validation delay is near RTT or $SRTT$, ICN fails to detect true CE. The distinction between congestion and re-congestion phases is not consistent. When the delay applied in the re-congestion state depends on $SRTT$ or RTT, the number of undetected CE increases. Congestion also decreases because ACK which correspond to a segment retransmitted are faster than the RTT measurement duration before the packet drop. The best trade-off for this scenario seems to choose an RTT_{min} validation delay. Indeed, this value leads to shorter detection delay while avoiding false retransmission identification at a low rate. In this experiment, normal congestion and re-congestion use the same validation delay. In the following, the validation delay is set to RTT_{min}.

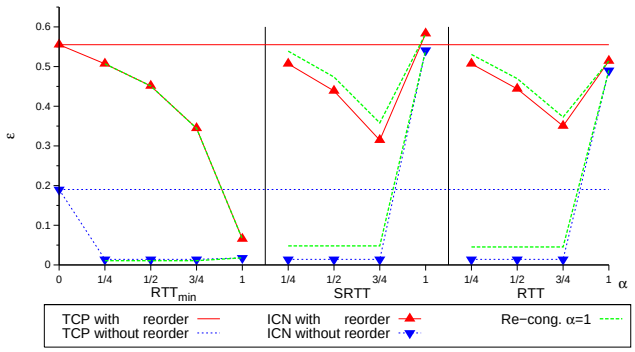


Figure 5: Sizing the validation delay.

4.2 Impact on different loads

Figure 6 shows the inaccuracy parameter as a function of load in the case where ICN uses RTT_{min}. To understand figure 6, we need to firstly look at figure 7. We define SLER, the TCP Spurious LER as the ratio between the number of spurious CE and the total number of sent packets. The SLER is almost constant in scenario with network reorder (because reorder rate does not change in our scenario). The LER increases with the load. Consequently and according to equation (2), the TCP inaccuracy decreases. For example, in case 0.05, no CE occurs and network reordering introduces spurious CE. This leads to an inaccuracy equals to 1.

In absence of network re-order, TCP inaccuracy is mainly due to spurious timeout in the Fast Recovery period. Indeed, with the impatient variant of TCP NewReno, the retransmission timer is only reset after the first partial ACK if a large number of packets are dropped from a data window [8]. The retransmission timer from the TCP sender will ultimately expire and the TCP sender will trigger a Slow-Start whatever the last retransmit packet outcome. The observed variations of inaccuracy in figure 6 come from unusual phenomenons as:

1. Real CE overlapping in a spurious CE. This case triggers a double error: one spurious CE and one undetected CE for TCP. This occurs when the bottom of the window is a re-ordered segment and inside this window a segment is lost. The retransmission of lost segment does not trigger a CE;

2. Packet reorder in the initial Slow-Start. This inserts spurious CE and let a congestion window smaller for TCP. For example, we can observe this for the 0.35 load, the accuracy seems better with reorder rather than without. In fact, ICN does not get the same number of errors but the number of CE in the scenario without network reorder is greater as shown figure 7. Surprisingly, reordering in Slow-Start avoids a burst of drops and a long delay to correct packet drops;

3. Spurious timeout from an ACK loss of a retransmission sent after a timer expiration. This case also triggers a spurious CE.

In a general manner, ICN is more accurate than TCP. Most of errors occur at the beginning of the flow when RTT_{min} are not correctly set. When a loss occurs in the first segments, there is a risk that a CE is not detected.

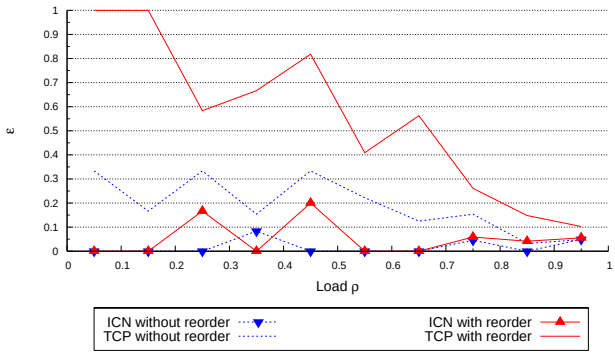


Figure 6: Inaccuracy function of load for TCP NewReno.

We have done measurements with TCP/SACK and reached similar conclusions. As SACK improves TCP retransmission decisions, there are less spurious retransmission which results in a better ICN accuracy.

4.3 ICN with timestamp

In order to distinguish an ACK resulting from a data packet to a retransmission packet, the Eifel algorithm [14] uses the timestamp option described in [10]. This option allows the sender to add timestamp to a packet later returned by the receiver in the corresponding ACK. Thus, the sender is able to compute an RTT by subtracting the current time to this timestamp.

The Eifel algorithm aims to distinguish an ACK which arrives after a retransmit has been sent in response to the original transmit or the retransmit. Although this timestamp scheme is not enabled by default mainly due to the

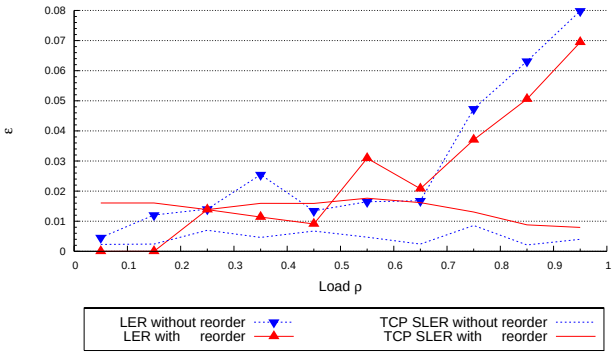


Figure 7: Reference flow of LER and TCP SLER function of the load.

overhead introduced by the option (i.e. 10 bytes in every packets) we choose to verify whether it might help the CE detection. As shown figure 8 and compare to 6, we can see this option improves the ICN detection but some identifications are still missing. For example, when the network reorder introduces a delay greater than twice the RTT (see load 0.45 in Figure 8), ICN cannot detect the spurious retransmission. As a result and whatever the algorithms used, a perfect detection is not realized today. The following section proposes to globally discuss all remaining cases.

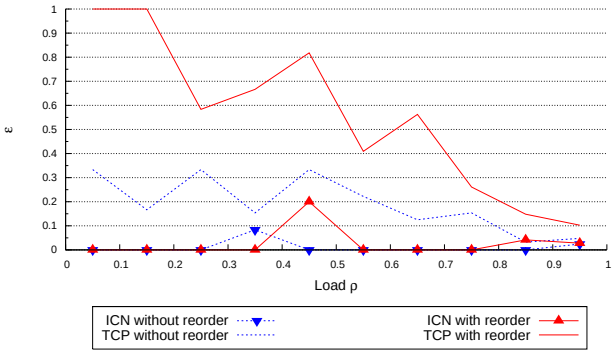


Figure 8: Use of the timestamp option.

4.4 Discussion

The off-line traces analysis shows that all congestions cannot be detected such as the case of spurious retransmission dropped (which corresponds to the following sequence: transmit, retransmit, drop; denoted $(t-r-d)$). Indeed, the drop of a retransmission is already corrected by the initial transmission. From a transport protocol point of view, this sequence is similar to a spurious retransmission $(t-r)$. TCP has nothing to do (the first transmission is well-received). In the context of ICN, the $(t-r-d)$ sequence is an issue as ICN cannot detect a CE from this loss. This is an open problem since the congestion does not appear at the transport level but at the network level and thus, is out of the scope of the present study.

One important result is that a solution only based on the RTT is not really reliable at the beginning of the connection compared to ICN. Indeed, the RTT estimated is not accurate due to the weak number of possible measurements. This limits the use of such solutions to long-lived flows.

Globally and following the scenarios proposed, ICN algorithm gives good results. However, solutions based on a validation delay need an RTT higher or in the same order of magnitude than the reordering delay. If the difference between the RTT and the reordering delay is too large, it becomes difficult to detect spurious retransmissions. In this context, ICN algorithm reaches its limit of use. To get an accurate detection, the events' duration which causes spurious retransmissions must be less than an RTT.

5. CONCLUSION

This paper has proposed an algorithm (ICN) able to estimate congestion events occurring in the network and implemented as a stand-alone component inside a framework (TCED). The purpose of this scheme is to demonstrate that congestion event detection can be realized independently of the TCP code in a sake of better detecting congestion occuring in the network. This algorithm is based on the combined use of two realistics and feasible assumptions which are 1) a delay or a timestamp to validate a loss following retransmission and 2) the acknowledgments path. We have evaluated this algorithm with and without network reordering cases and shown that an external and live congestion events detection is possible. We also emphasized that previous solutions based only on RTT measurements are not able to cover all cases. In particular and following measurements done with the ICN algorithm, we show a lack of differentiation between retransmissions due to reordering of loss and accuracy of the measurements with short TCP flows when using only RTT measurements.

Following, this work and the results obtained so far, we are currently planning the developpement of a kernel implementation of this framework and expect to drive a larger range of measurements which aims at benchmarking our proposal compared to embedded TCP spurious retransmission detection algorithms such as F-RTO and Eifel.

6. ACKNOWLEDGMENTS

We would like to thank Marc Allman from the ICSI Center for Internet Research for providing us the LEAST code. Pierre Ugo Tournoux for the development of the first ICN prototype and Tanguy Perennou for useful comments about this work.

7. REFERENCES

[1] M. Allman, W. Eddy, and S. Ostermann. Estimating loss rates with tcp. *ACM SIGMETRICS Performance Evaluation Review*, 31(3):12–24, December 2003.

[2] Mark Allman and Vern Paxson. On estimating end-to-end network path properties. *Computer Communication Review*, 29(4), October 1999.

[3] S. Bhandarkar, A. L. N. Reddy, M. Allman, and E. Blanton. Improving the Robustness of TCP to Non-Congestion Events. RFC 4653 (Experimental), August 2006.

[4] Sumitha Bhandarkar, Narasimha Reddy, Yueping Zhang, and Dmitri Loguinov. Emulating aqm from end hosts. In *Proc. of ACM SIGCOMM*, 2007.

[5] Carlo Caini and Rosario Firrincieli. TCP hybla: a TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 22, 2004.

[6] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[7] S. Floyd. Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166 (Informational), March 2008.

[8] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard), April 2004.

[9] Bryan Ford and Janardhan Iyengar. Breaking up the transport logjam. In *in Seventh ACM Workshop on Hot Topics in Networks (HotNets-VII)*, Calgary, Alberta, Canada, October 2008.

[10] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[11] Van Jacobson. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, pages 314–329, Stanford, CA, August 1988.

[12] Phil Karn and Craig Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Computer Communications Review*, 17(5):2–7, 1987.

[13] Andrew Lachlan, Marcondes Cesar, and Floyd Sally. Towards a common tcp evaluation suite. In *PFLDnet*, 2008.

[14] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP. RFC 3522 (Experimental), April 2003.

[15] Reiner Ludwig and Randy H. Katz. The eifel algorithm: making TCP robust against spurious retransmissions. *SIGCOMM Comput. Commun. Rev.*, 30(1):30–36, 2000.

[16] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proc. of ACM MOBICOM*, 2001.

[17] A. Medina, M. Allman, and S. Floyd. Measuring the evolution of transport protocols in the internet. *Computer Communication Review*, 35(2), April 2005.

[18] V. Paxson and M. Allman. Computing TCP's Retransmission Timer. RFC 2988 (Proposed Standard), November 2000.

[19] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001.

[20] S. Rewaskar, J. Kaur, and F.D. Smith. A passive state-machine approach for accurate analysis of tcp out-of-sequence segments. *ACM Computer Communications Review*, 36(3):51–64, 2006.

[21] 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.

[22] Michael Welzl. Using the ecn nonce to detect spurious loss events in TCP. In *Proc. of IEEE GLOBECOM*, December 2008.

[23] Lisong Xu, Khaled Harfoush, and Injong Rhee. Binary increase congestion control (bic) for fast long-distance networks. In *Proc. of IEEE INFOCOM*, 2004.

[24] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A reordering-robust TCP with DSACK. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, 2003.

FavourQueue: a Stateless Active Queue Management to Speed Up Short TCP Flows (and others too!)

Pascal Anelli[1], Emmanuel Lochin[2,3] and Remi Diana[2,3]
[1] Université de la Réunion - EA2525 LIM, Sainte Clotilde, France
[2] Université de Toulouse ; ISAE ; Toulouse, France
[3] TéSA/CNES/Thales ; Toulouse ; France

***Index Terms*—Active Queue Management; TCP; Performance Evaluation; Simulation; Flow interaction.**

***Abstract*—This paper presents and analyses the implementation of a novel active queue management (AQM) named FavourQueue that aims to improve delay transfer of short lived TCP flows over a best-effort network. The idea is to dequeue in first packets that do not belong to a flow previously enqueued. The rationale is to mitigate the delay induced by long-lived TCP flows over the pace of short TCP data requests and to prevent dropped packets at the beginning of a connection and during recovery period. Although the main target of this AQM is to accelerate short TCP traffic, we show that FavourQueue does not only improve the performance of short TCP traffic but also improve the performance of all TCP traffic in terms of drop ratio and latency whatever the flow size. In particular, we demonstrate that FavourQueue reduces the loss of a retransmitted packet, decrease the RTO recovery ratio and improves the latency up to 30% compared to DropTail.**

I. Introduction

Internet is still dominated by web traffic running on top of short-lived TCP connections [1]. Indeed, as shown in [2], among 95% of the client TCP traffic and 70% of the server TCP traffic have a size lower than ten packets. This follows a common web design practice that is to keep viewed pages lightweight to improve interactive browsing in terms of response time [3]. In other words, the access to a webpage often triggers several short web traffics that allow to keep the downloaded page small and to speed up the display of the text content compared to other heavier components that might compose it[1] (e.g. pictures, multimedia content, design components). As a matter of fact and following the growth of the web content, we can still expect a large amount of short web traffic in the near future.

TCP performance suffers significantly in the presence of bursty, non-adaptive cross-traffic or when the congestion window is small (i.e. in the slow-start phase or when it operates in the small window regime). Indeed, bursty losses, or losses during the small window regime, may cause Retransmission Timeouts (RTO) which trigger a slow-start phase. In the context of short TCP flows, TCP fast retransmit cannot be triggered if not enough packets are in transit. As a result, the loss recovery is mainly done thanks to the TCP RTO and this strongly impacts the delay. Following this, in this study we seek to improve the performance of this pervasive short TCP traffic without impacting on long-lived TCP flows. We aim to exploit router capabilities to enhance the performance of short TCP flows over a best-effort network, by giving a higher priority to a TCP packet if no other packet belonging to the same flow is already enqueued inside a router queue. The rationale is that isolated losses (for instance losses that occur at the early stage of the connection) have a strong impact on the TCP flow performance than losses inside a large window. Then, we propose an AQM, called FavourQueue, which allows to better protect packet retransmission and short TCP traffic when the network is severely congested.

In order to give to the reader a clear view of the problem we tackle with our proposal, we lean on paper [2]. Figure 1 shows that the flow duration (or latency[2]) of short TCP traffic is strongly impacted by an initial lost packet which is recovered later by an RTO. Indeed, at the early stage of the connection, the number of packets exchanged is to small to allow an accurate RTO estimation. Thus, an RTO is triggered by the default time value which is set to two seconds by default [4]. In this figure, the authors also give the cumulative distribution function of TCP flow length and the probability density function of their completion time from an experimental measurement dataset obtained during one day on a ISP BRAS link which aggregates more than 30,000 users. We have reproduced a similar experiment with ns-2 (i.e. with a similar flow length CDF according to a Pareto distribution) and obtained a similar probability density function of the TCP flows duration as shown in Figure 2 for the DropTail queue curve. Both figures (1 and 2) clearly highlight a latency peak at $t = 3$ seconds which corresponds to this default RTO value [4]. In this experiment scenario, the RTO recovery ratio is equal to 56% (versus 70% in the experiments of [2]). As a matter of fact, these experiments show that the success of the TCP slow-start is a key performance indicator. The second curve in Figure 2, shows the result we obtain by using our proposal called FavourQueue. Clearly, the peak previously emphasized has disappeared. This means the initial losses that strongly

[1]See for instance: "Best Practices for Speeding Up Your Web Site" from Yahoo developer network.

[2]The latency refers to the delay elapsed between the first sent and the last packet received.

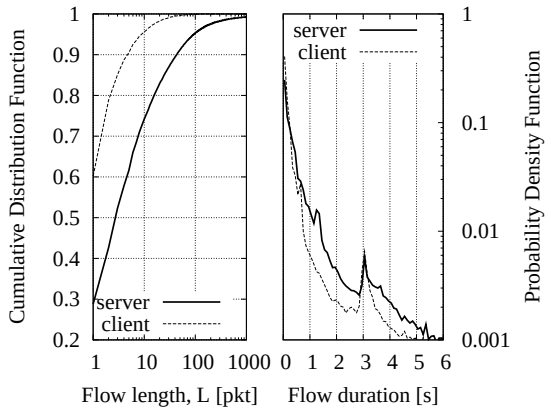


Fig. 1. TCP flow length distribution and latency (by courtesy of the authors of [2]).

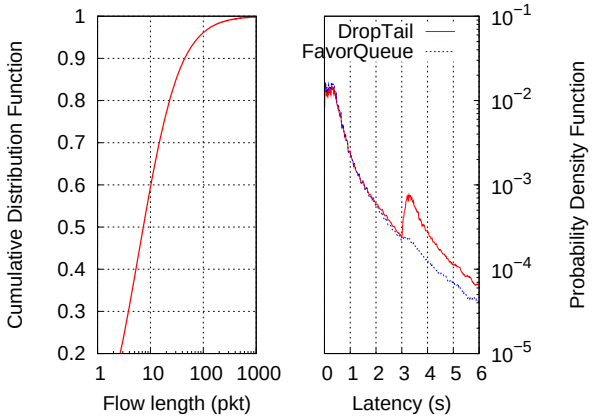


Fig. 2. TCP flow latency distribution from our simulation model.

impacted the TCP traffic performance have decreased.

An important contribution of this work is the demonstration that our scheme, by favouring isolated TCP packets, decreases the latency by decreasing the loss ratio of short TCP flows without impacting long TCP traffic. However, as FavourQueue does not discriminate short from long TCP flows, every flows take advantage of this mechanism when entering either the slow-start or a recovery phase. Our evaluations show that 58% of short TCP flows improve their latency and that 80% of long-lived TCP flows also take advantage of this AQM. For all sizes of flows, on average, the expected gain of the transfer delay is about 30%. This gain results from the decrease of the drop ratio of non opportunistic flows which are those that less occupy the queue. Furthermore, the more the queue is loaded, the more FavourQueue has an impact. Indeed, when there is no congestion, FavourQueue does not have any effect on the traffic. In other words, this proposal is activated only when the network is severely congested.

Finally, FavourQueue does not request any transport protocol modification. Although we talk about giving a priority to certain packet, there is no per-flow state needed inside the FavourQueue router. This mechanism must be seen as an extension of DropTail that greatly enhances TCP sources performance by favouring (more than prioritizing) certain TCP packets. The next Section II describes the design of the proposed scheme. Then, we presents in Section III the experimental methodology used in this paper. Sections IV and V dissects and analyses the performance of FavourQueue. Following these experiments and statistical analysis, we propose a stochastic model of the mechanism Section VI. We also present a related work in Section VII where we position FavourQueue with other propositions and in particular discuss how this AQM completes the action of recent proposals that aim to increase the TCP initial slow-start window. Finally, we propose to discuss the implementation and some security issues in Section VIII and conclude this work Section IX.

II. FavourQueue description

Short TCP flows usually carry short TCP requests such as HTTP requests or interactive SSH or Telnet commands. As a result, their delay performance are mainly driven by:

1) the end-to-end transfer delay. This delay can be reduced if the queueing delay of each router is low;
2) the potential losses at the beginning connection. The first packets lost at the beginning of a TCP connection (i.e. in the slow-start phase) are mainly recovered by the RTO mechanism. Furthermore, as the RTO is initially set to a high value, this greatly decreases the performance of short TCP flows.

The two main metrics on which we can act to minimize the end to end delay and protect from loss the first packets of a TCP connection and are respectively the queuing delay and the drop ratio. Consequently, the idea we develop with FavourQueue is to favor certain packet in order to accelerate the transfer delay by giving a preferential access to transmission and to protect them from drop.

This corresponds to implement a preferential access to transmission when a packet is enqueued and must be favoured (temporal priority) and a drop protection is provided when the queue is full (drop precedence) with push-out scheme that dequeue a standard packet in order to enqueue a favoured packet.

When a packet is enqueued, a check is done on the whole queue to seek another packet from the same flow. If no other packet is found, it becomes a favoured packet. The rationale is to decrease the loss of a retransmitted packet in order to decrease the RTO recovery ratio. The proposed algorithm (given in Algorithm 1) extends the one presented in [5] by adding a drop precedence to non-favoured packets in order to decrease the loss ratio of favoured packets. The selection of a favoured packet is done on a per-flow basis. As a result the complexity is as a function of the size of the queue which corresponds to the maximum number of state that the router must handle. The number of state is scalable considering today's routers capability to manage million of flows simultaneously [6]. However the selection decision is local and temporary as the state only exists when at least one packet is enqueued. This explains why we prefer the term of favouring packet more than prioritizing packet. Furthermore, FavourQueue does not introduced packet re-ordering inside a flow which obviously badly impacts TCP performance [7]. Finally, in the specific case where all the traffic becomes

Algorithm 1 FavourQueue algorithm

```
function enqueue(p)
# A new packet p of flow F is received
if less than 1 packet of F are present in the queue then
    # p is a favoured packet
    if the queue is full then
        if only favoured packets in the queue then
            p is drop
            return
        end if
    else
        # Push out
        the last standard packet is dropped
    end if
    p inserted in position pos_
    pos_ ← pos_ +1
else
    # p is a standard packet
    if the queue is not full then
        p is put at the end of the queue
    else
        p is dropped
    end if
end if
```

favoured, the behaviour of FavourQueue will be identical than DropTail.

III. Experimental methodology

We use ns-2 to evaluate the performance of FavourQueue. Our simulation model allows to apply different levels of load to efficiently compare FavourQueue with DropTail. The evaluations are done over a simple dumbbell topology. The network traffic is modeled in terms of flows where each flow corresponds to a TCP file transfer. We consider an isolated bottleneck link of capacity C in bit per second. The traffic demand, expressed as a bit rate, is the product of the flow arrival rate λ and the average flow size $E[\sigma]$. The load offered to the link is then defined by the following ratio:

$$\rho = \frac{\lambda E[\sigma]}{C}. \quad (1)$$

The load is changed by varying the arrival flow rate [8]. Thus, the congestion level increases as a function of the load. As all flows are independent, the flow arrivals are modeled by a Poisson process. A reasonable fit to the heavy-tail distribution of the flow size observed in practice is provided by the Pareto distribution. The shape parameter is set to 1.3 and the mean size to 30 packets. Left side in Figure 2 gives the flows' size distribution used in the simulation model.

At the TCP flow level, the ns-2 TCP connection establishment phase is enabled and the initial congestion window size is set to two packets. As a result, the TCP SYN packet is taken into account in all dataset. The load introduced in the network consists in several flows with different RTT according to the recommendation given in the "Common TCP evaluation suite" paper [8]. The load is ranging from 0.05 to 0.95 with a step of 0.1. The simulation is bounded to 500 seconds for each given load. To remove both TCP feedback synchronization and phase effect, a traffic load of 10% is generated in the opposite direction. The flows in the transient phase are removed from the analysis. More precisely, only flows starting after the first forty seconds are used in the analysis. The bottleneck link capacity is set to $10Mbps$. All other links have a capacity of $100Mbps$. According to the small buffers rule [9], buffers can be reduced by a factor of ten. The rule of thumb says the buffer size B can be set to $T \times C$ with T the round-trip propagation delay and C the link capacity. We choose $T = 100ms$ as it corresponds to the averaged RTT of the flows in the experiment. The buffer size at the two routers is set to a bandwidth-delay product with a delay of $10ms$. The packet length is fixed to 1500 bytes and the buffer size has a length of 8 packets.

To improve the confidence of these statistical results, each experiment for a given load is done ten times using different sequences of pseudo-random numbers (in the following we talk about *ten replications experiment*). Some figures also average the ten replications, meaning that we aggregate and average all flows from all ten replications and for all load conditions. In this case, we talk about *ten averaged experiment* results which represents a dataset of nearly 17 million of packets. The rationale is to consider these data as a real measurement capture where the load is varying as a function of time (as in [2]) since each load condition has the same duration. In other words, this represents a global network behaviour.

The purpose of these experiments is to weight up the benefits brought by our scheme in the context of TCP best-effort flows. To do this, we first experiment a given scenario with DropTail then, we compare with the results obtained with FavourQueue. We enable FavourQueue only on the uplink (data path) while DropTail always remains on the downlink (ACK path). We only compare all identical terminating flows for both experiments (i.e. DropTail and FavourQueue) in order to assess the performance obtained in terms of service for a same set of flows.

We assume our model follows Internet short TCP flows characteristics as we find the same general distribution latency form than Figure 1 which is as a function of the measurements obtained in Figure 2. This comparison provides a correct validation model in terms of latency. As explained above, Figure 2 corresponds and illustrates a *ten averaged experiment*.

IV. Performance evaluation of TCP flows with FavourQueue

We present in this section global performance obtained by FavourQueue then we deeper analyze its performance and investigate the case of persistent flows. We compare a same set of flows to assess the performance obtained with DropTail and FavourQueue.

A. Overall performance

We are interested in assessing the performance of each TCP flows in terms of latency and goodput. We recall from Section

I that we defined the latency as the time to complete a data download (i.e the transmission time) and the goodput is the average pace of the download. In order to assess the overall performance of FavourQueue compared to DropTail, Figure 3 gives the mean and standard deviation of the latency as a function of the traffic load of FavourQueue. We both study FavourQueue with and without the push-out mechanism in order to distinguish the supplementary gain provided by the drop precedence.

The results are unequivocal. FavourQueue version without push-out as presented in [5] provides a gain when the load increases compared to DropTail (i.e. when the queue has a significant probability of having a non-zero length) while the drop precedence (with push-out) clearly brings out a significant gain in terms of latency. Basically, Figure 4 shows that both queues (with and without push-out) globally drop the same amount of packets. However, the push-out version better protects short TCP flow (and more generally: all flows entering a slow-start phase) as when the queue is congested, it always enqueues a packet from a new flow. As a result, initial lost of packets further decreases. Indeed, as already emphasized in Figure 2 from the introduction, losses do not occur at the beginning of a connection and as a result, the flow is not impacted anymore by the retransmission overhead resulting from an RTO. Thus, our favouring scheme allows to prevent lost packets during the startup phase. As a matter of fact, this is explained by a different distribution of these losses.

Following this, we have computed the resulting normalized goodput for all flows size for all experiments and obtained is 2.4% with DropTail and 3.5% with FavourQueue (i.e. around 1% of difference). This value is not weak as it corresponds to an increase of 45%.

Figure 5 gives the average latency as a function of the flow length. The cumulative distribution function of the flow length is also represented. On average, we observe that FavourQueue obtains a lower latency than DropTail whatever the flow length. This difference is also larger for the short TCP flows which are also numerous (we recall that the distribution of the flows' size follows a Pareto distribution and as a result the number of short TCP flow is higher). This demonstrates that FavourQueue particularly favors the slow-start of every flow and as a matter of fact: short TCP flows. The cloud pattern obtained for a flow size higher than hundred is due to the decrease of the statistical sample (following the Pareto distribution used for the experiment) that result in a greater dispersion of the results obtained. As a result, we cannot drive a consistent latency analysis for sizes higher than hundred.

To complete these results, Figure 6 gives the latency obtained when we increase the size of both queues. We observe that whatever the queue size, FavourQueue always obtains a lower latency. Behind a given queuesize (in Figure 6 at $x = 60$), the increase of the queue does not have an impact on the latency. This enforces the consistency of the solution as Internet routers prevent the use of large queue size.

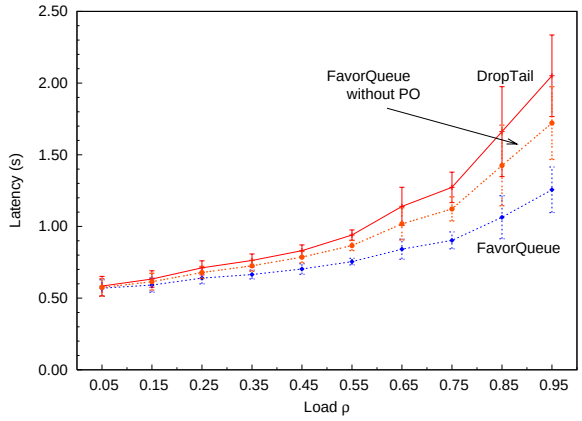


Fig. 3. Overall latency according to traffic load.

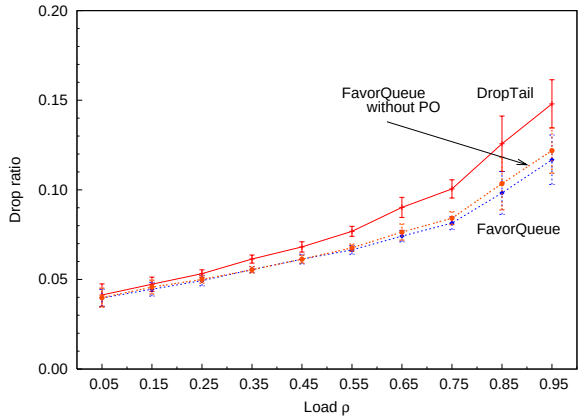


Fig. 4. Overall drop ratio according to traffic load.

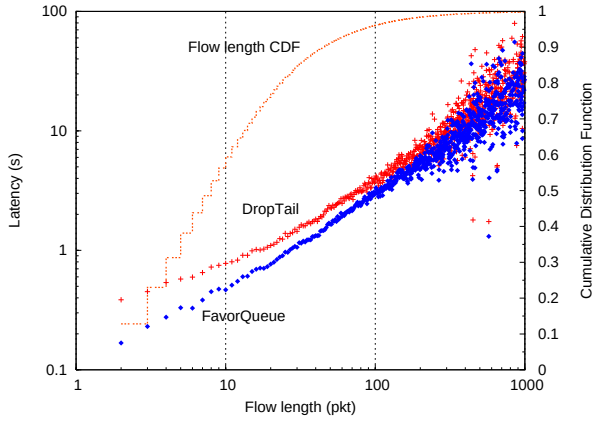


Fig. 5. Flow length CDF and mean latency as a function of the flow length.

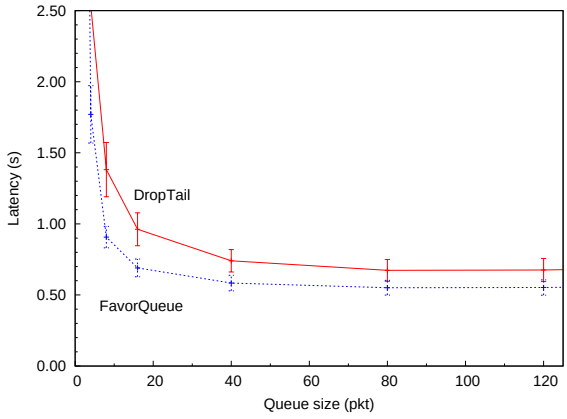


Fig. 6. Overall latency according to queuesize.

B. Performance analysis

To refine our analysis of the latency, we propose to evaluate the difference of latencies per flows for both queues. We denote $\Delta i = Td_i - Tf_i$ with Td and Tf the latency observed respectively by DropTail and FavourQueue for a given flow i. Figure 7 gives the distribution of the latencies difference. This figure illustrates that there is more decrease of the latency for each flow than increase. Furthermore for 16% of flows, there is no impact on the latency i.e. $\Delta = 0$. In other words, 84% of flows observe a change of latency; 55% of flows observe a decrease ($\Delta > 0$) and 10% of flows observe a significant change ($\Delta > 1$ second). However, 30% of the flows observe an increase of their latency ($\Delta < 0$). In summary, FavourQueue has a positive impact on certain flows that are penalised with DropTail.

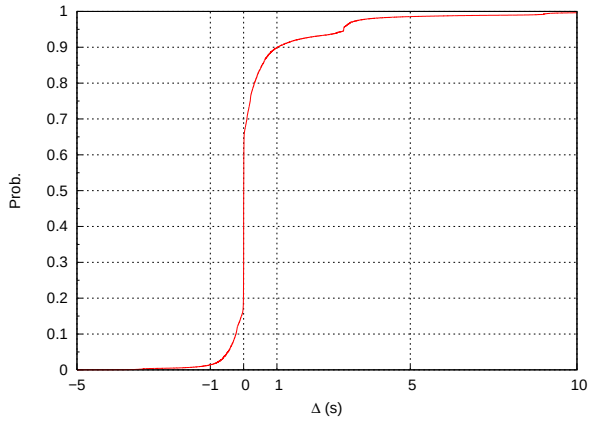


Fig. 7. Cumulative distribution function of latency difference Δ.

In order to assess the flows that gain in terms of latency, Figure 8 gives the probability of latency improvement. For the whole set of short TCP flows, (i.e. with a size lower than 10 packets), the probability to improve the latency reaches 58% while the probability to decrease is 25%. For long TCP flows (i.e. above 100 packets), the probability to improve and to decrease the latency is respectively 80% and 20%. The flows with a size around 30 packets are the ones with the highest probability to be penalised. For long TCP flows, the large variation of the probability indicates a uncertainty which mainly depends on the experimental conditions of the flows. We have to remark that long TCP flows are less present in this experimental model (approximately 2% of the flows have a size higher or equal to 100 packets). As this curve corresponds to a ten averaged experiment, each long TCP flows have experienced various load conditions and this explains these large oscillations.

Medium sized flows are characterized by a predominance of the slow-start phase. During this phase, each flow opportunistically occupies the queue and as a results less packets are favoured due to the growth of the TCP window. The increase of the latency observed for medium sized flows (ranging from 10 to 100) is investigated later in subsection V-A. We will also see in the next subsection IV-C that FavourQueue acts like a shaper for these particular flows.

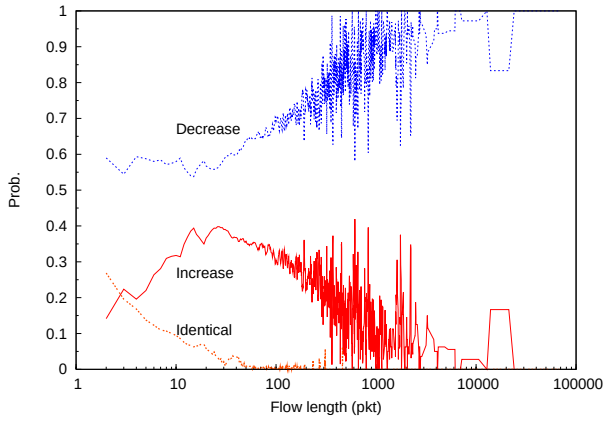


Fig. 8. Probability to change the latency.

To estimate the latency variation, we define $G(x)$ the latency gain for the flows of length x as follows:

$$G(x) = \frac{\sum_{i=1}^{N} \Delta_{x_i}}{\sum_{i=1}^{N} Td_{x_i}}. \quad (2)$$

with N, the number of flows of length x. A positive gain indicates a decrease of the latency with FavourQueue. Figure 9 provides the positive, negative and total gains as a function of the flows size. We observe an important total gain for the short TCP flows. The flows with an average size obtain the highest negative gain and this gain also decreases when the size of the flows increases. Although some short flows observe an increase of their latency, in a general manner, the positive gain is always higher. This preliminary analysis illustrates that FavourQueue improves by 30% on average the best-effort service in terms of latency. The flows that take the biggest advantage of this scheme are the short flows with a gain up to 55%.

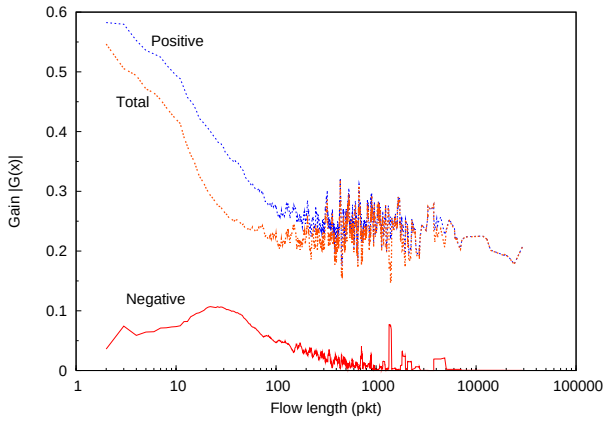


Fig. 9. Average Latency gain per flow length.

Finally and to conclude with this section, we plot in Figure 10 the number of flows in the system under both AQM as a function of time to assess the change in the stability of the network. We observe that FavourQueue considerably reduces

both the average number of flows in the network as well as the variability.

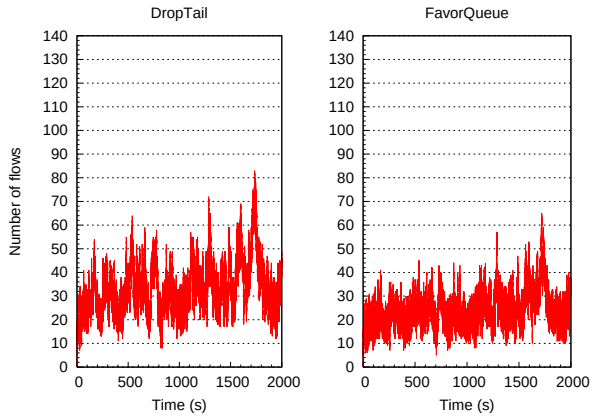


Fig. 10. Number of simultaneous flows in the network.

C. The case of persistent flows

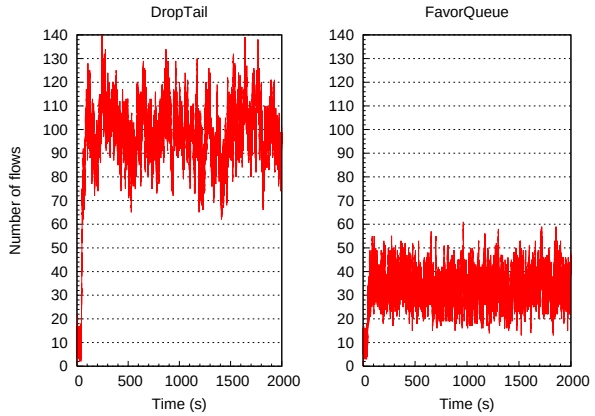


Fig. 11. Number of short flows in the network when persistent flows are actives.

Following [10], we evaluate how the proposed scheme affects persistent flows with randomly arriving short TCP flows. We now change the network conditions with 20% of short TCP flows with exponentially distributed flow sizes with a mean of 6 packets. Fourty seconds later, 50 persistent flows are sent. Figure 11 gives the number of simultaneous short flows in the network. When the 50 persistent flows start, the number of short flows increases and oscillates around 100 with DropTail. By using FavourQueue, the number increases to 30 short flows. The short flows still take advantage of the favour scheme and Figure 12 confirms this point. However we observe in Figure 13 that the persistent flows are not penalized. The mean throughput is nearly the same (1.81% for DropTail versus 1.86% for FavourQueue) and the variance is smaller with FavourQueue. Basically, FavourQueue acts as a shaper by slowing down opportunistic flows while decreasing the drop ratio of non opportunistic flows (those which less occupy the queue).

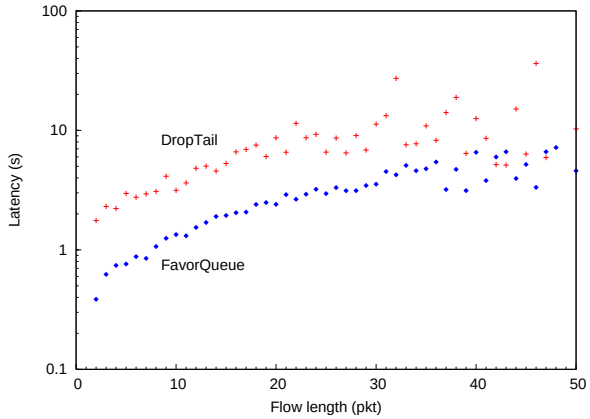


Fig. 12. Mean latency as a function of flow size for short flows in presence of persistent flows.

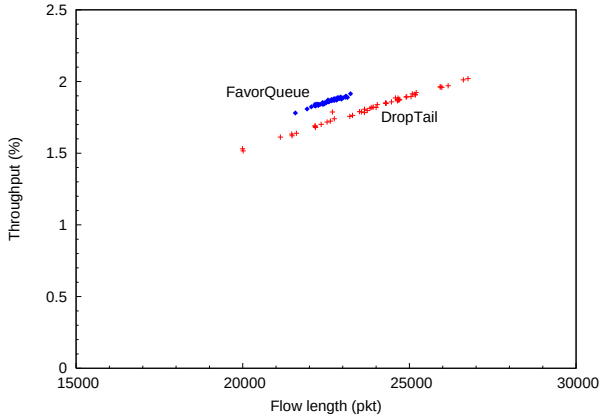


Fig. 13. Mean throughput as a function of flow length for 50 persistent flows.

V. Understanding FavourQueue

The previous section has shown the benefits obtained with FavourQueue in terms of service. In this section, we analyse the reasons of the improvements brought by FavourQueue by looking at the AQM performance. We study the drop ratio and the queueing delay obtained by both queues in order to assess the reasons of the gain obtained by FavourQueue. We recall that for all experiments, FavourQueue is only set on the upstream. The reverse path uses a DropTail queue. In a first part, we look at the impact of the AQM on the network then on the end-host.

A. Impact on the network

Figure 14 shows the evolution of the average queueing delay depending on the size of the flow. This figure corresponds to the 10 averaged replications experiment (as defined Section III). Basically, the results obtained by FavourQueue and DropTail are similar. Indeed, the average queueing delay is 2.8ms for FavourQueue versus 2.9ms for DropTail and both curves similarly behave. We can notice that the queueing delay for the medium sized flows slightly increases with FavourQueue. These flows are characterized by a predominance of the slow-start phase as most of the packets that belong to these flows are emitted during the slow-start. Since during this phase

each flow opportunistically occupies the queue, less packets are favoured due to the growth of the TCP window. As a result, their queuing delay increases. When the size of the flow increases (above hundred packets length), the slow-start is not pervasive anymore and the average queueing delay of each packet of these flows tends to be either higher or lower as suggested by the cloud Figure 14 depending on the number of favoured packets during their congestion avoidance phase.

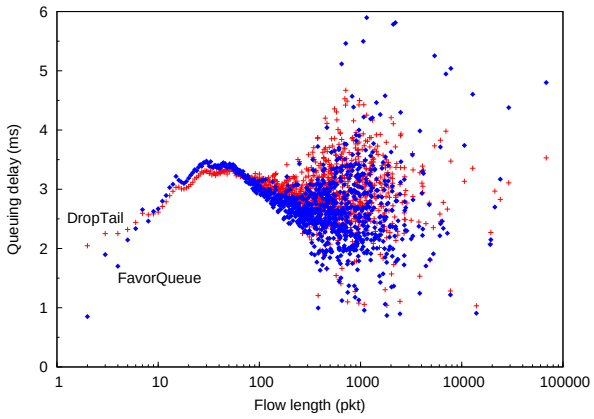


Fig. 14. Average queuing delay according to flow length.

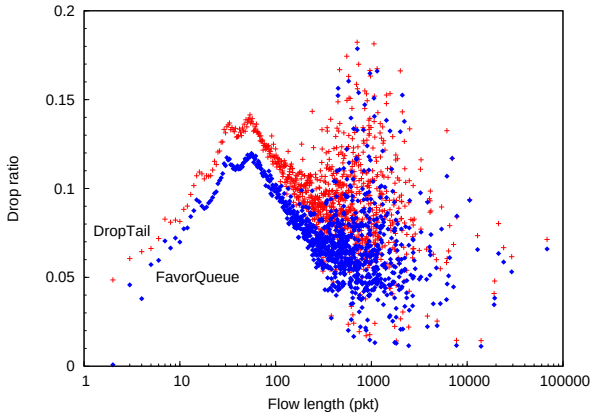


Fig. 15. Average drop ratio according to flow length.

However and as suggests Figure 15, the good performance in terms of latency obtained by FavourQueue previously shown Figure 3 in Section IV are mostly due to a significant decrease of the drop ratio. If we look at the average drop ratio of both queues in Figure 15, still as a function of the flow length, we clearly observe that the number of packets dropped is lower for FavourQueue. Furthermore, the loss ratio for the flow size of 2 packets is about 10^{-3} meaning that the flows of this size obtain a benefit compared to DropTail. The slow-start phase is known to send burst of data [11]. Thus, most of the packets sent during the slow-start phase have a high probability to be not favoured. This explains the increase of the drop ratio according to the flow size until 60 packets. Indeed, in the slow-start phase, packets are sent by burst of two packets. As a result, the first packet is favoured and the second one will be favoured only if the first is already served. Otherwise, the second packet might be therefore delayed. Then, when their respective acknowledgements are back to the source, the next sending will be more spaced. As FavourQueue might decrease the burstiness of the slow-start, we might decrease the packet loss rate and thus improve short TCP flow performance.

If we conjointly consider both figures 14 and 15, we observe that FavourQueue enables a kind of traffic shaping that decreases TCP aggressivity during the slow-start phase which results in a decrease of the number of dropped packets. As the TCP goodput is proportional to $1/(RTT.\sqrt{p})$ [12], the decrease of the drop ratio leads to an increase of the goodput which explains the good performance obtained by FavourQueue in terms of latency.

The loss ratio of SYN segments is on average 1.8% with DropTail. However for a load higher than 0.75, this loss ratio value reaches 2.09% while with FavourQueue, this ratio is 0.06%. Finally on average for all load conditions, this value is 0.04% with FavourQueue. These results demonstrate the positive effect to protect SYN segments from the loss. Obviously, by using FavourQueue in duplex mode (we recall that we have tested FavourQueue only on the upstream), this would further improve the results as SYN/ACK packets would have also been protected.

B. Impact on the end-host performance

The good performance obtained with FavourQueue in terms of latency are linked to the decrease of the losses at the beginning of the flow. In the following, we propose to estimate the benefits of our scheme by estimating the RTO ratio as a function of the network load. We define the RTO ratio $T(\rho)$ for a given load ρ as follows:

$$T(\rho) = \frac{\sum_{i=1}^{N} RTO_i}{\sum_{i=1}^{N}(L_i + R_i)}, \tag{3}$$

with RTO_i, the number of RTO for the i flow; R_i its number of retransmission and L_i its size. The ten replications experiment in Figure 16 presents the evolution of the RTO ratio for FavourQueue and DropTail and shows that the decrease of the loss ratio results in a decrease of the RTO ratio for FavourQueue. This also shows the advantage to use FavourQueue when the network is heavily loaded.

We now evaluate the RTO recovery ratio as a function of the flow length. We define this RTO recovery ratio as follows:

$$\tau(x) = \frac{\sum_{i=1}^{N} RTO_i}{\sum_{i=1}^{N}(RTO_i + FR_i)}, \tag{4}$$

with FR_i, the number of TCP Fast Retransmit for the i flow. In terms of RTO recovery, Figure 17 shows a significant decrease of the number of recovery with an RTO. Concerning the ratio of Fast Retransmit for this experiment, we observe an increase of 14% with FavourQueue. As a fast recovery packet is placed at the beginning of a window, FavourQueue prevents the loss of a retransmission. Then, the number of recovery with Fast Retransmit is higher with FavourQueue and the latency observed is better since the retransmission are faster.

For the flow with a size strictly below six packets, the recovery is exclusively done by a RTO. Indeed, in this case there is not enough duplicate acknowledgement to trigger a Fast Retransmit. For the flow above six packets, we observe

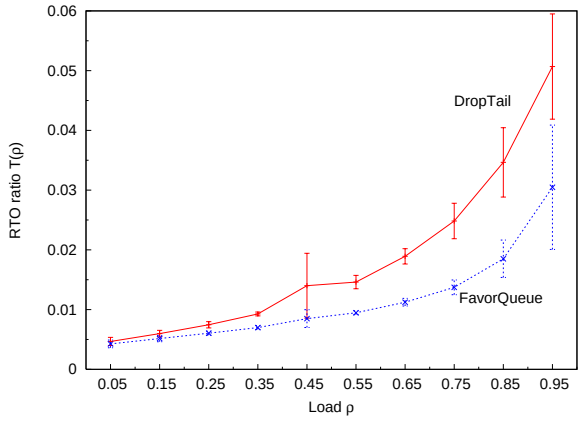


Fig. 16. RTO ratio as a function of the network load.

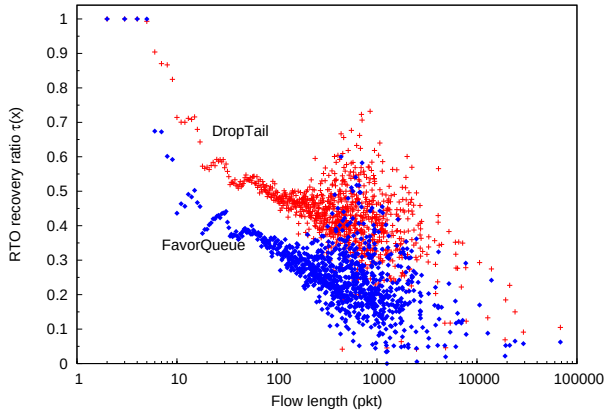


Fig. 17. RTO recovery ratio according to flow length.

a noticeable decrease of the RTO ratio due to the decrease of the packet lost rate on the first packets of the flow. Thus, the number of duplicate acknowledgement is higher, allowing to trigger a Fast Retransmit recovery phase. The trend shows a global decrease of the RTO ratio when the flow length increases. On the overall, the RTO recovery ratio reaches 56% for DropTail and 38% for FavourQueue. The decrease of the gain obtained follows the increase of the flow size. This means that FavourQueue helps the connection establishment phase.

VI. Stochastic model of FavourQueue

We analyze in this part the impacts of the temporal and drop priorities previously defined in Section II and propose a stochastic model of the mechanism.

A. Preliminary statistical analysis

We first estimate the probability to favour a flow as a function of its length by a statistical analysis. We define $P(Favor|S = x)$, the probability to favour a flow of size s, as follows:

$$P(Favor|S = s) = \frac{\sum_{i=1}^{N} fa_i}{\sum_{i=1}^{N} s + R_i}. \quad (5)$$

with fa_i, the number of packets which have been favoured and R_i the number of retransmitted packets of a given i flow.

The number of favoured packets corresponds to the number of packets selected to be favoured at the router queue. Figure 18 gives the results obtained and shows that:

- the flows with a size of two packets are always favoured;
- the middle sized flows that mainly remain in a slow-start phase are less favoured compared to short flows. The ratio reaches 50% meaning that one packet among two is favoured;
- long TCP flows get a favouring ratio around 70%.

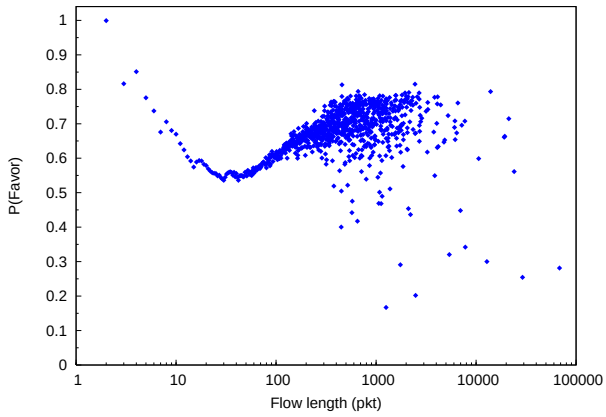


Fig. 18. Probability of packet favouring according to flow length.

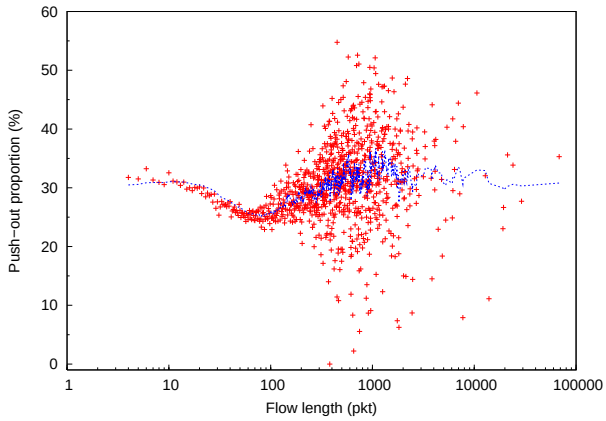


Fig. 19. Push-out proportion of drop as a function of flow length.

We also investigate the ratio of packets dropped resulting from the push-out algorithm as a function of the flow length in order to assess whether some flows are more penalised by push-out. As shown, Figure 19, the mean is about 30% for all flows, meaning that the push-out algorithm does not impact more short than long TCP flows.

We now propose to build a stochastic model of Figure 18 in the following.

B. Stochastic model

We denote S: the random variable of the size of the flow and Z: the Bernoulli random variable which is equal to 0 if no favoured packets are present in the queue and 1 otherwise. We then distinguish three different phases:

- phase #1 : each flows have a size lower than s_1. In this phase, the flows are in slow-start mode. This size is a parameter of the model which depends of the load. ;

- phase #2 : each flows have a size higher than s_1 and lower than s_2. In this phase, flows progressively leave the slow-start mode (corresponding to the bowl between $[10:100]$ in Figure 18). This is the most complex phase to model as all flows are either in the congestion avoidance phase or at the end of their slow-start. s_2 is also a parameter of the model which depends of the load;
- phase #3 : each flows have a size higher than s_2. All flows are in congestion avoidance phase. Note that the statistical sample which represents this cloud is not large enough to correctly model this part (as already pointed out in Section IV-A). However, one other important result given by Figure 18 is that 70% of packets of flows in congestion avoidance mode are favoured. We will use this information to infer the model. This also confirms that the spacing between each packet in the congestion avoidance phase increases the probability of an arriving packet to be favoured.

First phase: We consider a bursty arrival and assume that all packets belonging to the previous RTT have left the queue. Then, the burst size (BS) can take the following values: $BS = 1, 2, 4, 8, 16, 32, ...$ If $Z = 0$, we assume that a maximum of 3 packets can be favoured in a row[3], the packets number that are favoured are $1, 2, 3, 4, 5, 6, 8, 9, 10, 16, 17, 18, ...$ and $1, 2, 4, 8, 16, 32, ...$ if $Z = 1$. Thus, if $Z = 0$, the probability to favour a packet of a flow of size s is:

$$P(Favor|(Z=0,S=s)) = \begin{cases} s, s \leqslant 6 \\ \frac{s-1}{s}, 7 \leqslant s \leqslant 10 \\ \frac{9}{s}, 11 \leqslant s \leqslant 15 \\ \frac{s-6}{s}, 16 \leqslant s \leqslant 18 \\ \frac{12}{s}, 19 \leqslant s \leqslant 31 \\ (...) \end{cases} \quad (6)$$

and with $Z = 1$:

$$P(Favor|(Z=1,S=s)) = \begin{cases} 1, s = 1 \\ \frac{2}{s}, 2 \leqslant s \leqslant 3 \\ \frac{3}{s}, 4 \leqslant s \leqslant 7 \\ \frac{4}{s}, 8 \leqslant s \leqslant 15 \\ \frac{5}{s}, 16 \leqslant s \leqslant 31 \\ (...) \end{cases} \quad (7)$$

The probability to favour a packet of a flow of size s is thus:

$$P(Favor|S=s) = P(Z=0).P(Favor|(Z=0,S=s)) + P(Z=1).P(Favor|(Z=1,S=s)) \quad (8)$$

Once again, $P(Z = 0)$ and $P(Z = 1)$ depends on the load of the experiment and must be given.

Second phase: In this phase, each flow progressively leaves the slow-start phase. First, when a flow finishes its slow-start phase, each following packets have a probability to be favoured of 70% (as shown in in Figure 18). So, we now need to compute an average value of the probabilty to favour a packet for a given flow. We also have to take into account that, for a given size of flow s, only a proportion of these flows have effectively left the slow-start phase. The other ones remain in slow-start and the analysis of their probabilty to favour a packet follows the first phase. To correctly describe this phase, we need to assess which part of flows of size s, $s_1 \leqslant s \leqslant s_2$, has left the slow start phase at packet s_1, $s_1 + 1$, ... s. As a first approximation, we use a uniform distribution between s_1 and s_2. This means that for flows of size s, the proportion of flows which have left the slow-start phase at s_1, $s_1 + 1$, ... $s - 1$ is $\frac{1}{s_2 - s_1}$ and the proportion of flows of size s which have not yet left the slow-start phase is thus $\frac{s_2 - s}{s_2 - s_1}$.

If we denote p_k the proportion of flows of size $s \geqslant s_1$ that have left the slow start-phase at k we have:

$$P(Favor|(S=s,Z=0)) = \sum_{i=0}^{s-s_1-1} p_k.P(Favor|k=s_1+i,Z=0,S=s)$$

and

$$P(Favor|(S=s,Z=1)) = \sum_{i=0}^{s-s_1-1} p_k.P(Favor|k=s_1+i,Z=1,S=s)$$

and as in (8) we obtain:

$$P(Favor|S=s) = P(Z=0).\sum_{i=0}^{s-s_1-1} p_k.P(Favor|k=s_1+i,Z=0,S=s) + P(Z=1).\sum_{i=0}^{s-s_1-1} p_k.P(Favor|k=s_1+i,Z=1,S=s)$$

Third phase: The model of this phase is quite simple. In fact, each packet of a flow which have left the slow-start phase have a probability to be favoured of 70%. We compute the probability for a packet to be favoured by taking into account the time at which a flow has left the slow-start phase and the proportion of flows as in the second phase.

Model fitting: To verify our model, among the ten loads that are averaged in Figure 18, we choose two verify our model for two loads: $\rho = 0.25$ and $\rho = 0.85$. For the first one we have estimated $P(Z = 1) = 0.25$ and $P(Z = 1) = 0.7$ for the second. Figures 20 and 21 show that our model correctly fits both experiments.

This model allows to understand the peaks in Figure 20 when the flow size is lower than hundred packets. These peaks are explained by the modelling of the first phase. Indeed, the traffic during the slow-start is bursty, then, each burst has either one or two packets favoured as a function of Z (i.e. up to three packets are favoured when $Z = 0$ and only one when $Z = 1$ as given by (6) and (7)).

VII. Related work

Several improvements have been proposed in the literature and at the IETF to attempt to solve the problem of short

[3]The rationale is the following, if $Z = 0$ a single packet (such as the SYN packet) is favoured and one RTT later, the burst of two packets (or larger) will be favoured if we consider that the first packet of this burst is directly served.

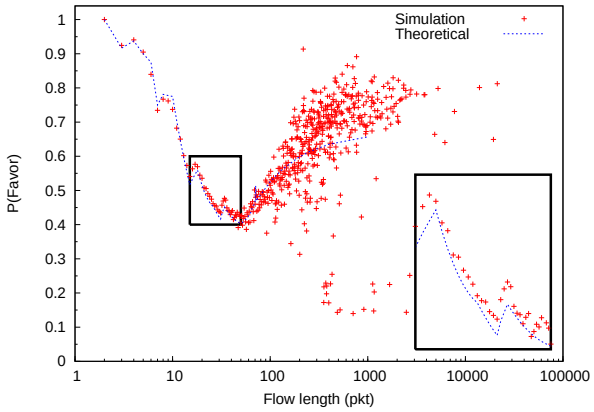


Fig. 20. Model fitting for $\rho = 0.25$ with $P(Z = 1) = 0.25$.

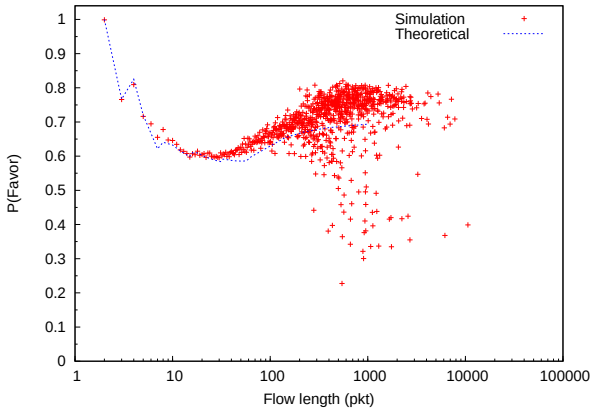


Fig. 21. Model fitting for $\rho = 0.85$ with $P(Z = 1) = 0.7$.

TCP flows performance. Existing solutions can be classified into three different action types: (1) to enable a scheduling algorithm at the router queue level; (2) to give a priority to certain TCP packets or (3) to act at the TCP level in order to decrease the number of RTO or the loss probability. Concerning the two first items, the solution involves the core network while the third one involves modifications at the end-host. In this related work, we first situate FaQ among several core network solutions and then explain how FaQ might complete end-hosts' solutions.

A. Enhancing short TCP flows performance inside the core network

1) The case of short and long TCP flows differentiation: Several studies [13][10][14] have proposed to serve first short TCP traffic to improve the overall system performance. These studies follow one queueing theory result which stands that the overall mean latency is reduced when the shortest job is served first [15]. One of the precursor in the area is [14], where the authors proposed to adapt the Least Attained Service (LAS) [15], which is a scheduling mechanism that favors short jobs without prior knowledge of job sizes, for packets networks. As for FavourQueue, LAS is not only a scheduling discipline but a buffer management mechanism. This mechanism follows FavourQueue principle since the priority given to the packet is done without knowlegde of the size of the flow and that the classification is closely related to the buffer management scheme. However, the next packet serviced under LAS is the one that belongs to the flow that has received the least amount of service. By this definition, LAS will serve packets from a newly arriving flow until that flow has received an amount of service equal to the amount of least service received by a flow in the system before its arrival. Compared to LAS, FavourQueue has no notion of amount of service as we seek to favour short job by accelerating their connection establishement. Thus, there is no configuration and no complex settings.

In [13] and [10], the authors push further the same idea and attempt to differentiate short from long TCP flows according to a scheduling algorithm. The differences between these solutions are based on the number of queues used which are either flow stateless or stateful. Theses solutions uses an AQM which enables a push out algorithm to protect short TCP flow packets from loss. Short TCP flows identification is done inside the router by looking at the TCP sequence number [10]. However and in order to correctly distinguish short from long TCP flows, the authors modify the standard TCP sequence numbering which involves a major modification of the TCP/IP stack. In [13], the authors propose another solution with a per-flow state and deficit round robin (DRR) scheduling to provide fairness guarantee. The main drawback of [14][13] is the need of a per-flow state while [10] requires TCP senders modifications.

2) The case of giving a priority to certain TCP packets: Giving a priority to certain TCP packets is not a novel idea. Several studies have tackled the benefit of this concept to improve the performance of TCP connection. This approach was really popular during the QoS networks research epoch as many queueing disciplines was enabled over IntServ and DiffServ testbed allowing researchers to investigate such priority effects. Basically, the priority can be set intra-flow or inter-flow. Marco Mellia et Al. [16] have proposed to use intra-flow priority in order to protect from loss some key identified packets of a TCP connection in order to increase the TCP throughput of a flow over an AF DiffServ class. In this study, the authors observe that TCP performance suffers significantly in the presence of bursty, non-adaptive cross-traffic or when it operates in the small window regime, *i.e.*, when the congestion window is small. The main argument is that bursty losses, or losses during the small window regime, may cause retransmission timeouts (RTOs) which will result in TCP entering the slow-start phase. As a possible solution, the authors propose qualitative enhancements to protect against loss: the first several packets of the flow in order to allow TCP to safely exit the initial small window regime; several packets after an RTO occurs to make sure that the retransmitted packet is delivered with high probability and that TCP sender exits the small window regime; several packets after receiving three duplicate acknowledgement packets in order to protect the retransmission. This allows to protect against losses the packets that strongly impact on the average TCP throughput. In [3][17], the authors propose a solution on inter-flow priority. The short TCP flow are marked IN. Thus, packets from these flows are marked as a low drop priority. The differentiation in

core routers is applied by an active queue management. When sender has sent a number of packets that exceeds the flow identification threshold, the packet are marked OUT and the drop probability increase. However, these approaches need the support of a DiffServ architecture to perform [18].

B. Acting at the TCP level

The last solution is to act at the TCP level. The first possibility is to improve the behavior of TCP when a packet is dropped during this start up phase (i.e. initial window size, limited transit). The second one is to prevent this drop by decreasing the probability of segments lost. For instance, in [19], the authors propose to apply an ECN mark to SYN/ACK segments in order to avoid to drop them. The main drawback of these solutions is that they require important TCP sender modifications that might involve heavy standardisation process.

We wish to point out that one of the current hot topic currently discussed within the Internet Congestion Control Research Group (ICCRG) deals with the TCP initial window size. In a recent survey, the authors of [20] highlight that the problem of short-lived flows is still not yet fully investigated and that the congestion control schemes developed so far do not really work if the connection lifetime is only one or two RTTs. Clearly, they argue for further investigation on the impact of initial value of the congestion window on the performance of short-lived flows. Some recent studies have also demonstrated that larger initial TCP window helps faster recovery of packet losses and as a result improves the latency in spite of increased packet losses [21], [22]. Several proposals have also proposed solutions to mitigate the impact of the slow start [23], [24], [25].

Although we do not act at the end-host side, we share the common goal to reduce latency during the slow start phase of a short TCP connection. However, we do not target the same objective. Indeed, end-host solutions, that propose to increase the number of packets of the initial window, seek to mitigate the impact of the RTT loop while we seek to favour short TCP traffic when the network is congested. At the early stage of the connection, the number of packets exchanged is low and a short TCP request is both constrained by the RTT loop and the small amount of data exchange. Thus, some studies propose to increase this initial window value [21], [22]; to change the pace at which the slow-start sends data packets by shrinking the timescale at which TCP operates [26]; even to completely suppress the slow-start [24]. Basically, all these proposals attempt to mitigate the impact of the slow-start loop that might be counterproductive over large bandwidth product networks. On the contrary, FavourQueue do not act on the number of data exchanged but prevents losses at the beginning of the connection. As a result, we believe that FavourQueue must not be seen as a competitor of these end-host proposals but as a complementary mechanism. We propose to illustrate this complementarity by looking at the performance obtained with an initial congestion window sets to ten packets. Figure 22 gives the complementary cumulative distribution function of the latency for DropTail and FavourQueue with flows with an initial slow-start set to two or ten packets. We do not have changed the experimental conditions (i.e. the router buffer is still set to eight packets) and this experiment corresponds to a ten averaged experiments (see section III). As explained in [21], if we focus on the results obtained with DropTail for both initial window size, the increase of the initial window improves the latency (with the price of an increase of the loss rate as also denoted in [21]). However, the use of FavourQueue enforces the performance obtained and complement the action of such end-host modifications making FavourQueue a generic solution to improve short TCP traffic whatever the slow-start variant used.

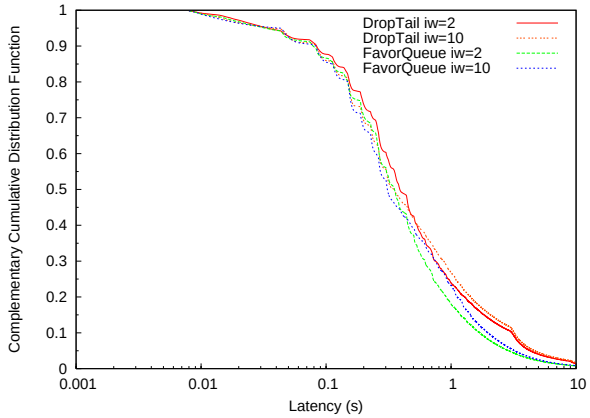


Fig. 22. Comparison of the benefit obtained in terms of latency with an initial TCP window size of ten packets.

VIII. Discussion

A. Security consideration

In the related work presented in Section VII, we present a similar solution to our proposal that gives priority to TCP packets with a SYN flag set. One of the main criticism that raises such kind of proposals usually deal with TCP SYN flood attack where TCP SYN packets may be used by malicious clients to improve this kind of threat [27]. However, this is a false problem as accelerating these packets do not introduce any novel security or stability side-effects as explained in [28]. Today, current kernel enables protection to mitigate such well-known denial of service attack[4] and current Intrusion Detection Systems (IDS) such as SNORT[5] combined with firewall rules allow network providers and companies to stop such attack. Indeed, the core network should not be involved in such end-host security issue that should remain under the reponsability of edge networks and end-hosts. Concerning the reverse path and as raised in [28], provoking web servers or hosts to send SYN/ACK packets to third parties in order to perform a SYN/ACK flood attack would be greatly inefficient. This is because the third parties would immediately drop such packets, since they would know that they did not generate the TCP SYN packets in the first place.

[4]See for instance http://www.symantec.com/connect/articles/hardening-tcpip-stack-syn-attacks

[5]http://www.snort.org/

B. Deployment issue

Although there is no scalability issue anymore inside new Internet routers that can manage millions of per-flow state [6]. FavourQueue does not involve per-flow state management and the number of entries that need to handle a FavourQueue router is as a function of the number of packets that can be enqueued. Furthermore, as the size of a router buffer should be small [9], the number of states that need to be handle is thus bounded.

To sump up, the proposed scheme respects the following constraints:

- easily and quickly deployable; this means that FavourQueue has no tuning parameter and does not require any protocol modification at a transport or a network level;
- independently deployable: installation can be done without any coordination between network operators. Operation must be done without any signaling;
- scalable; no per-flow state is needed.

FavourQueue should be of interest for access networks; entreprise networks or universities where congestion might occur at their output Internet link.

IX. Conclusion

In this paper, we investigate a solution to accelerate short TCP flows. The main advantages of the proposed AQM is that FavourQueue is stateless; does not require any modification inside TCP; can be used over a best effort network; does not request to be completely deployed over an Internet path. Indeed, a partial deployment could only be done over routers from an Internet service provider or over a specific AS.

We drive several simulation scenarios showing that the drop ratio decreases for all flow length, thus decreasing their latency. FavourQueue significantly improves the performance of short TCP traffic in terms of transfer delay. The main reasons are that this mechanism strongly reduces the loss of a retransmitted packet triggered by an RTO and improves the connection establishment delay. Although FavourQueue targets short TCP performance, results show that by protecting retransmitted packets, the latency of the whole traffic and particularly non-opportunistic flows, is improved.

In a future work, we aim at investigating FavourQueue with rate-based transport protocols such as TFRC in order to verify whether we would benefit similar properties and with delay-based TCP protocol variants (such as TCP Vegas and TCP Compound) that should intuitively take large benefit of such AQM. We also expect to enable ECN support in FavourQueue.

Acknowledgements

The authors wish to thank Eugen Dedu for numerous discussions about this work.

References

[1] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir, "Atlas internet observatory 2009 annual report," in 47th NANOG, 2009. [Online]. Available: http://www.nanog.org/meetings/nanog47/presentations/Monday/Labovitz_ObserveReport_N47_Mon.pdf

[2] D. Ciullo, M. Mellia, and M. Meo, "Two schemes to reduce latency in short lived TCP flows," Communications Letters, vol. 13, no. 10, October 2009.

[3] X. Chen and J. Heidemann, "Preferential treatment for short flows to reduce web latency," Computer Networks, vol. 41, no. 6, pp. 779–794, April 2003.

[4] R. Braden, "Requirements for internet hosts," RFC 1122, Internet Engineering Task Force, October 1989.

[5] E. Dedu and E. Lochin, "A study on the benefit of TCP packet prioritisation," in PDP, Weimar, Germany, February 2009.

[6] L. Roberts, "A radical new router," Spectrum, vol. 46, no. 7, July 2009.

[7] C. Arthur, A. Lehane, and D. Harle, "Keeping order: Determining the effect of TCP packet reordering," in ICNS, Athens, Greece, June 2007. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/ICNS.2007.78

[8] L. Andrew, C. Marcondes, and S. Floyd, "Towards a common TCP evaluation suite," in Workshop on Protocols for Fast Long-Distance Networks (PFLDnet), Manchester, March 2008.

[9] Y. Ganjali and N. McKeown, "Update on buffer sizing in internet routers," Computer Communication Review, vol. 36, no. 5, October 2006.

[10] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg, "Differentiation between short and long TCP flows: Predictability of the response time," in INFOCOM. Hong Kong: IEEE, March 2004.

[11] C. Barakat and E. Altman, "Performance of short TCP transfers," in NETWORKING. Paris: Springer-Verlag, May 2000.

[12] M. M., S. J., and M. J., "The macroscopic behavior of the TCP congestion avoidance algorithm," Computer Communications Review, vol. 27, no. 3, 1997.

[13] A. Kantawala and J. Turner, "Queue management for short-lived TCP flows in backbone routers," in GLOBECOM. IEEE, November 2002, pp. 2380–2384.

[14] I. Rai, E. Biersack, and G. Urvoy-Keller, "Size-based scheduling to improve the performance of short TCP flows," Network, vol. 19, no. 1, pp. 12–17, January 2005.

[15] L. Kleinrock, Queueing systems, ser. Wiley Interscience. John Wiley & Sons, 1975, vol. 1.

[16] M. Mellia, I. Stoica, and H. Zhang, "TCP-aware packet marking in networks with diffserv support," Computer Networks, vol. 42, no. 1, pp. 81–100, 2003.

[17] L. Guo and L. Matta, "The war between mice and elephants," in ICNP. Riverside, California: IEEE, November 2001.

[18] S. Blake, D. Blak, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," RFC 2475, Internet Engineering Task Force, December 1998.

[19] A. Kuzmanovic, A. Mondal, S. Floyd, and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) capability to TCP's SYN/ACK packets," RFC 5562, Internet Engineering Task Force, June 2009.

[20] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," Communications Surveys & Tutorials, vol. 12, no. 3, pp. 304 –342, 2010.

[21] N. Dukkipati et al., "An argument for increasing TCP's initial congestion window," Computer Communication Review, vol. 40, no. 3, July 2010.

[22] M. Scharf, "Performance evaluation of fast startup congestion control schemes," in NETWORKING. Aachen, Germany: Springer-Verlag, May 2009. [Online]. Available: http://www.springerlink.com/content/6078217660g68170/

[23] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," RFC 4782 (Experimental), January 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4782.txt

[24] D. Liu, M. Allman, S. Jin, and L. Wang, "Congestion control without a startup phase," in Workshop on Protocols for Fast Long-Distance Networks (PFLDnet), 2007.

[25] M. Scharf and H. Strotbek, "Performance evaluation of quick-start TCP with a Linux kernel implementation," in NETWORKING, 2008.

[26] V. Konda and J. Kaur, "RAPID: Shrinking the congestion-control timescale," in INFOCOM. IEEE, April 2009.

[27] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987 (Informational), Internet Engineering Task Force, August 2007.

[28] A. Kuzmanovic, "The power of explicit congestion notification," in SIGCOMM. Philadelphia: ACM, August 2005, pp. 61–72. [Online]. Available: http://www.cs.northwestern.edu/~akuzma/doc/ecn.pdf