UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

École Doctorale STIC
Sciences et Technologies de l'Information et de la Communication
Institut EURECOM

THÈSE DE DOCTORAT

de

l'UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

présentée par

Arnaud Legout

pour obtenir le titre de
DOCTEUR ès SCIENCES
de
l'UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS

Spécialité **RÉSEAUX INFORMATIQUES**

Sujet de la thèse :

Contrôle de congestion multipoint pour les réseaux *best effort*

Rapporteur	Dr Ken Chen	Professeur, Université paris 13
	Dr Jim Kurose	Professeur, University of Massachusetts

Soutenue le 24 octobre 2000 à 15h00 devant le jury composé de :

Rapporteur	Dr Ken Chen	Professeur, Université paris 13
Examinateur	Dr Ernst W. Biersack	Professeur, Institut Eurécom
	Dr Walid Dabbous	Directeur de recherche, INRIA
	Dr James Roberts	Chef de département R&D, France Telecom

À mes parents.

Remerciements

Lorsque l'on veut faire une thèse, il faut avant tout chercher un bon directeur de thèse. Dans cette quête, je suis allé demander conseil à Jean Bolot qui m'a orienté vers Ernst Biersack. Ne connaissant que très peu Ernst à cette époque, les paroles de Jean furent déterminantes dans mon choix et je l'en remercie vivement. En arrivant chez Ernst, j'avais de nombreuses attentes qui furent largement dépassées.

Ernst a toujours su m'orienter dans une bonne direction, à commencer par le bureau de Jörg Nonnemacher. J'ai partagé pendant un an le bureau avec Jörg et durant cette période, il m'a constamment aidé, stimulé et fait partager ses nombreuses idées. Jörg a également été celui qui crut possible notre papier INFOCOM'99 un mois avant la date limite de soumission alors que l'ont avaient aucun résultat. Ce papier aurait été impossible sans l'aide de Jörg ; j'ai appris à cette occasion que ce n'est pas en s'enfermant dans un bureau que les bonnes idées viennent, mais que lorsque la bonne idée est là, il faut se fixer des objectifs et ne plus compter son temps pour les atteindre. Pour toutes ces raisons, je remercie Jörg.

Ernst m'a orienté très rapidement vers le contrôle de congestion multipoint en me demandant d'étudier les problèmes du protocole RLM. Il m'a conseillé et soutenu dans mon travail, mais il m'a toujours laissé la plus grande liberté quant'à mes choix ; il m'a appris à écrire des papiers scientifiques et à faire des présentations : il m'a appris à devenir un chercheur ! De plus, ce qui est fondamental pour un doctorant, surtout en période de doutes, il a apporté à mon travail une caution scientifique de grande valeur. Je remercie donc Ernst pour tout ce qu'il m'a appris.

Je remercie les rapporteurs de ma thèse Ken Chen et Jim Kurose qui ont pris le temps de lire ma thèse et de me donner de nombreux commentaires ainsi que Jim Roberts et Walid Dabbous d'avoir fait parti de mon jury.

Je tiens également à remercier plusieurs personnes qui ont facilité mon travail : David Tremouilhac et Didier Loisel m'ont toujours offert un support technique irréprochable ; tout le personnel de l'Institut Eurecom et en particulier Agnes et plus tard Olivia ont facilité mes taches administratives ; Evelyne Biersack a eu le courage de lire ma thèse et de faire de nombreuses corrections.

L'environnement cosmopolite de l'Institut Eurecom a été très enrichissant, je remercie tous les doctorants avec qui j'ai eu le plaisir de passer ces trois années : Morsy, Matthias, Sergio, Neda, Pablo, Jakes, Pierre, Alain, Mamdouh, etc. Je tiens en particulier à remercier Jamel avec qui j'ai partagé un bureau pendant plus d'une année et avec qui j'ai passé de très bon moments.

Pour finir je tiens à remercier Cecile pour son amour ainsi que mes parents qui m'ont soutenu, financé et qui ont cru en moi durant toutes mes études.

Résumé

Une des clefs de l'amélioration de la qualité de service pour les réseaux *best effort* est le contrôle de congestion. Dans cette thèse, on a étudié le problème du contrôle de congestion pour la transmission multipoint dans les réseaux *best effort*. Cette thèse présente quatre contributions majeures. On a commencé par étudier deux protocoles de contrôle de congestion multipoints RLM et RLC. On a identifié des comportements pathologiques pour chaque protocole. Ceux-ci sont extrêmement difficiles à corriger dans le contexte actuel de l'internet, c'est-à-dire en respectant le paradigme *TCP-friendly*. On a alors réfléchi au problème du contrôle de congestion dans le contexte plus général des réseaux *best effort*. Ceci nous a conduit à redéfinir la notion de congestion, définir les propriétés requises par un protocole de contrôle de congestion idéal et définir un nouveau paradigme pour la conception de protocoles de contrôle de congestion presque idéaux. On a introduit à cet effet le paradigme *Fair Scheduler* (FS). L'approche que l'on a utilisée pour définir ce nouveau paradigme est purement formelle. Pour valider cette approche théorique, on a conçu grâce au paradigme FS un nouveau protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur : PLM, qui est capable de suivre les évolutions de la bande passante disponible sans aucune perte induite, même dans un environnement autosimilaire et multifractal. PLM surpasse RLM et RLC et valide le paradigme FS. Comme ce paradigme permet de concevoir des protocoles de contrôle de congestion multipoints et point à point, on a défini une nouvelle politique d'allocation de la bande passante entre flux multipoints et flux point à point. Cette politique, appelée $LogRD$, permet d'améliorer considérablement la satisfaction des utilisateurs multipoints sans nuire aux utilisateurs point à point.

Abstract

An efficient way to improve quality of service for best effort networks is through congestion control. We present in this thesis a study of multicast congestion control for best effort networks. This thesis shows four major contributions. We first exhibit some pathological behaviors for the multicast congestion control protocols RLM and RLC. As these pathological behaviors are extremely hard to fix in the context of the current Internet (i.e. with the TCP-friendly paradigm), we thought about the problem of congestion control in the more general case of best effort networks. We give a new definition of congestion, we define the properties required by an ideal congestion control protocol, and we define a paradigm, the fair scheduler (FS) paradigm, for the design of nearly ideal end to end congestion control protocols. We define this paradigm in a formal way. To validate this paradigm in a pragmatic way, we design with the FS paradigm a new multicast congestion control protocol: PLM. This protocol converges fast to the available bandwidth and tracks this available bandwidth without loss induced even in a self similar and multifractal environment. PLM outperforms RLM and RLC and validates the FS paradigm claims. As the FS paradigm allows to devise multicast and unicast congestion control protocols, we define a new bandwidth allocation policy for unicast and multicast flows. This policy called $LogRD$ allows to increase the multicast receiver satisfaction without significantly decreasing the unicast receiver satisfaction.

Table des matières

Table des figures

Chapitre 1

Introduction

Au début des années 1990, l'internet a connu, avec l'avènement du *Word Wide Web*, une révolution dans son mode d'utilisation. Il est devenu le support de services multimédias pour le grand public. Or, l'internet n'était préparé ni à supporter un service multimédia, ni à connecter le grand public. On verra au § 1.1 que les pionniers de l'ARPANET (le précurseur de l'internet) avaient fait des choix architecturaux permettant le déploiement de nouveaux services et l'interconnexion d'un grand nombre d'ordinateurs. Cependant, même les plus optimistes de l'époque prédirent une croissance de l'internet qui, aujourd'hui, fait sourire, tant elle est inférieure de plusieurs ordres de grandeur à la réalité.

Le grand public attend aujourd'hui de l'internet une certaine qualité de service. Cependant, l'internet est un réseau *best effort* qui, par définition, n'offre pas de qualité de service. En fait, la qualité de service est obtenue par des protocoles aux systèmes terminaux. Par conséquent, un des moyens les plus efficaces d'améliorer la qualité de service est d'améliorer les protocoles aux systèmes terminaux ; en particulier, les protocoles de contrôle de congestion. D'autre part, le concept de transmission multipoint fut introduit pour permettre à l'internet d'offrir de nouveaux services.

On va étudier dans cette thèse le problème du contrôle de congestion pour les réseaux *best effort*, mais en axant notre étude sur la transmission multipoint. Dans la suite, en s'appuyant sur leur fondement historique, on va définir les termes « *best effort* », « contrôle de congestion » et « transmission multipoint ». Cette perspective historique va nous permettre de motiver le sujet de cette thèse. En effet, c'est en comprenant le rôle qu'ont joué l'architecture *best effort* et le contrôle de congestion dans le succès de l'internet que l'on comprendra pourquoi il est primordial pour la pérennité de l'internet d'étudier le contrôle de congestion pour les réseaux *best effort*. Et c'est en expliquant pourquoi la transmission multipoint permet de déployer de nouveaux services, et pourquoi le problème de contrôle de congestion multipoint est si ardu, que l'on pourra juger combien il est important d'orienter notre étude vers la transmission multipoint.

1.1 Le concept de réseau *best effort*

On pourrait définir de nombreux fondements de l'internet actuel : l'architecture décentralisée, la commutation de paquets, l'interconnexion par le protocole IP, le concept de réseaux *best effort*, l'argument *end-to-end*, etc. L'idée directrice qui a guidé les pionniers de l'ARPANET (le précurseur de l'internet, créé en 1969) était de fournir un réseau qui puisse interconnecter les ordinateurs du monde entier [38]. Tous les fondements de l'internet actuel et, en particulier le concept de réseaux *best effort*, découlent de cette idée directrice. Si l'on veut un réseau qui puisse fonctionner en interconnectant un grand nombre de machines très hétérogènes, tant au niveau du matériel que des applications, il doit être simple. Introduire un mécanisme spécifique à une application dans le réseau peut s'avérer néfaste pour une autre application. Le concept de *best effort* signifie que l'on a un réseau qui va s'occuper de transmettre un paquet de données d'un point à un autre du réseau sans aucune garantie de : fiabilité, débit, délai, gigue, etc. ; en résumé, sans aucune garantie de qualité de service. L'argument *end-to-end* [78] complète le concept de *best effort* en disant que le réseau ne doit pas essayer d'offrir un service qu'il ne peut pas entièrement garantir, c'est-à-dire sans le support des systèmes terminaux, sauf si le service partiel offert par le réseau est utile pour toutes les applications ; on dit que c'est un mécanisme d'utilité globale (*broad utility*). En résumé, l'argument *end-to-end* dit simplement qu'il vaut mieux repousser les mécanismes qui donnent de la qualité de service vers les systèmes terminaux. Cependant, la notion de mécanisme d'utilité globale – notion qui autorise à ajouter un mécanisme dans le réseau – est sujette à interprétation. Il est extrêmement difficile de prédire si un mécanisme d'utilité globale à un moment donné ne sera pas néfaste pour une application qui n'apparaîtra que plus tard.

La clairvoyance des pionniers de l'ARPANET qui décidèrent de garder le réseau *best effort* permit l'avènement du *Word Wide Web*. En effet, Leonard Kleinrock écrit en 1974 [52] : « le domaine des réseaux d'ordinateurs est certainement arrivé à maturité, les applications ont été clairement identifiées et la technologie existe pour satisfaire les besoins des applications... » Que se serait-il passé si, croyant les applications clairement identifiées, ils avaient rajoutés dans le réseau des mécanismes pour améliorer ces applications qui étaient le courrier électronique (*e-mail*), le transfert de fichiers et, plus tard, les forums de discussion (*news*) ? Ces mécanismes auraient sans doute engendré des délais sans conséquence pour de telles applications asynchrones, mais rédhibitoires pour une application qui apparaîtra vingt ans plus tard et qui révolutionnera notre mode de communication : le *Word Wide Web*.

Le concept de réseaux *best effort* est donc une nécessité pour assurer la pérennité de l'internet. Cependant, certaines applications nécessitant des garanties de très faible délai – comme les simulations militaires ou les jeux distribués – ou des garanties de haut débit – par exemple la télévision haute définition – auront certainement besoin de mécanismes spécifiques ; inté-

grer de telles applications dans un Internet *best effort* est toujours un sujet d'actives recherches. Même s'il apparaît des réseaux spécialisés pour des applications spécifiques qui ont de fortes contraintes de qualité de service, l'histoire du *best effort* a montré que ce service se justifiera toujours par son faible coût, sa facilité de maintenance et surtout son extrême flexibilité qui se traduit par le très large spectre d'applications autorisées.

1.2 Le contrôle de congestion

Un réseau *best effort* est un réseau qui n'offre aucune garantie de qualité de service aux systèmes terminaux (voir § 1.1). Le support des protocoles aux systèmes terminaux est donc indispensable pour offrir de la qualité de service. Ces protocoles donnent certaines fonctionnalités qui sont directement assimilées à de la qualité de service, comme la fiabilité ou l'ordonnancement des paquets. D'autres fonctionnalités, comme le contrôle de congestion, ont un rôle qui, bien que fondamental au bon fonctionnement du réseau, n'est pas directement assimilé à de la qualité de service. Le problème du contrôle de congestion dans les réseaux d'ordinateurs est né avec les réseaux d'ordinateurs[1]. En 1974, Leonard Kleinrock [52], à la demande de J. Walter Bond l'éditeur de *ACM SIGCOMM Computer Communication Review*, donna son avis sur les domaines qui nécessitaient des investigations urgentes dans les réseaux. Kleinrock cita le problème du contrôle de flux (*flow control*) comme un des problèmes les plus sérieux.

On note que Kleinrock parle de contrôle de flux et non de contrôle de congestion (*congestion control*). On va expliquer cette distinction par la suite. Un mécanisme de contrôle de flux est un mécanisme qui limite l'entrée des paquets dans le réseau pour une raison ou pour une autre [52]. La manière la plus efficace de contrôler un flux est de le contrôler aux extrémités du réseau (soit directement au niveau de la paire source/récepteur, soit au niveau des points d'accès du réseau avec des mécanismes de *shaping* et de *policing*). On note que derrière l'idée de contrôler le flux aux extrémités du réseau, il y a l'idée de l'argument *end-to-end*. À l'époque de l'ARPANET, les goulots d'étranglement venaient des machines aux extrémités du réseau et non du réseau lui-même. Le principal problème pour une source était de ne pas faire déborder la file de réception du récepteur. Si le nombre de paquets reçus, par le récepteur, est plus grand que la taille de la file de réception, il y a des pertes. Dans certains cas pathologiques, on pouvait arriver à des débits très faibles. Le contrôle de flux, en limitant l'entrée des paquets dans le réseau pour éviter le débordement des files de réception, permit de résoudre le problème.

On introduisit la notion de *receiver advertized window* dans TCP [11] pour faire du contrôle de flux ; cette *receiver advertized window* correspond au nombre maximum de bytes que la file de réception du récepteur peut contenir. Grâce à la *receiver advertized window*, TCP limite le nombre de paquets dans le réseau au nombre maximum de paquets que la file de réception

1. Dans toute la suite de cette thèse, le terme *réseau* signifiera toujours *réseau d'ordinateurs*.

du récepteur peut contenir ; il ne peut, par conséquent, jamais y avoir de pertes au niveau de la file de réception du récepteur. Vinton Cerf et al. dans la RFC 675 [11] de décembre 1974, qui est la première description du protocole TCP, indiquent que le but du contrôle de flux est d'éviter la saturation des systèmes terminaux. Dans la RFC 793 [69] de 1981, qui est la dernière spécification du protocole TCP, Jon Postel identifie le contrôle de flux comme une opération de base de TCP et définit le contrôle de flux de TCP comme un mécanisme qui empêche la source d'envoyer plus de paquets que ce que le récepteur peut accepter, par exemple, en fonction de l'espace disponible dans sa file de réception.

Il faut attendre 1984 [56] pour que soit identifiée la nécessité d'un mécanisme de contrôle de congestion dans ce que l'on appelait les réseaux IP/TCP. En effet, John Nagle observa sur le réseau de la *Ford Aerospace and Communications Corporation* une très forte dégradation des performances qu'il appela *congestion collapse*. Le problème se produisait lorsque le réseau était très chargé ; une brusque augmentation de la charge pouvait conduire à une augmentation du RTT (*Round Trip Time*) plus rapide que l'estimateur du RTT de TCP. Par conséquent, TCP retransmettait des paquets qui étaient déjà dans le réseau. Le plus surprenant est que ce phénomène conduisait à un état stable où chaque paquet était transmis plusieurs fois et, par conséquent, où le débit utile – débit effectivement observé par l'application – était très faible. D'autre part, Nagle expliqua que ce phénomène n'était pas encore observé dans l'ARPANET à cause de la grande provision de bande passante de ce réseau, mais qu'un *congestion collapse* était inévitable si un mécanisme de contrôle de congestion n'était pas utilisé dans l'ARPANET. Nagle introduisit la notion de congestion comme un phénomène interne au réseau qui ne pouvait apparaître que dans un réseau suffisamment chargé (phénomène rare dans l'ARPANET jusqu'en 1986). C'est à partir de ce moment que l'on put faire une réelle distinction entre contrôle de flux et contrôle de congestion. Le contrôle de flux était destiné à éviter le débordement des files de réception des récepteurs, le contrôle de congestion était maintenant destiné à éviter la congestion dans le réseau, phénomène dû au remplissage excessif des files dans le réseau.

Octobre 1986, le premier de ce qui deviendra une série de *congestion collapse* se produit. Durant cette période, le débit entre le LBL et UC Berkeley chute de 32 Kbit/s à 40 bit/s. La prévision de Nagle, deux ans plus tôt, s'était vérifiée. Pour résoudre ce problème, Van Jacobson et Michael J. Karels proposèrent, en 1988, sept nouveaux algorithmes [39] à introduire dans TCP. TCP avait un mécanisme de contrôle de flux, il eut en 1988 des mécanismes de contrôle de congestion. Ces mécanismes, qui ont donné des fonctionnalités de contrôle de congestion à TCP, ont permis de préserver l'internet d'un nouveau *congestion collapse* jusqu'à aujourd'hui. À partir de cette époque, le contrôle de congestion devint un élément fondamental des réseaux *best effort* ; sans contrôle de congestion, le réseau est inutilisable. La RFC 2581 [1] spécifie les mécanismes actuels de contrôle de congestion de TCP.

Jusqu'ici, on n'a pas donné de définition précise de la notion de congestion. Une définition

générale sera donnée dans la suite de cette thèse (voir § 3.2.2). La définition de congestion utilisée par TCP est liée à la notion de perte ; il y a congestion pour TCP dès qu'il y a perte. Cette définition de la notion de congestion est, cependant, restrictive et dangereuse. Restrictive, parce qu'elle suppose que seule une perte peut être un signal de congestion : en fait, une perte n'est que le signal d'une congestion qui a commencé bien avant ; dangereuse, parce qu'elle considère les pertes comme nécessaires, c'est le signal de congestion, et parce que des phénomènes autres que la congestion peuvent produire des pertes : par exemple, des erreurs de transmission sur des liens radios. La notion de congestion telle que définie par TCP est donc imparfaite. De plus, comme la notion de contrôle de congestion est essentielle dans les réseaux *best effort*, il nous a semblé nécessaire d'étudier le problème du contrôle de congestion dans ce type de réseaux.

1.3 La transmission multipoint

Les premières pierres de la transmission multipoint pour l'internet ont été posées par Stephen Deering, en 1988 [18], qui a proposé plusieurs algorithmes de routage multipoint. Le principe de la transmission multipoint est le suivant : l'algorithme de routage multipoint établit un arbre entre la source et les récepteurs. La source envoie des paquets aux récepteurs à travers cet arbre. Le gain de la transmission multipoint vient du fait que, contrairement à la transmission point à point où la source doit envoyer autant de copies d'un paquet qu'il y a de récepteurs, une source multipoint envoie une seule copie du paquet et c'est le réseau qui copiera le paquet à chaque fourche de l'arbre multipoint. Ce mode de transmission implique qu'il n'y aura qu'une seule copie de chaque paquet qui passera sur chaque branche de l'arbre multipoint .

Le véritable déploiement de la transmission multipoint a commencé avec les débuts du réseau Mbone [24, 53] (*Multicast Backbone*) en 1992. Les principales applications sur le Mbone étaient – et sont encore – la vidéo, l'audio et le tableau partagé (*whiteboard*). À la différence du tableau partagé qui requiert fiabilité et cohérence temporelle, la vidéo tolère naturellement, mais dans une certaine mesure, les pertes et la congestion. L'audio tolère également les pertes lorsqu'elles sont éparses – c'est-à-dire lorsqu'elles peuvent être corrigées soit par un mécanisme de contrôle d'erreur par anticipation utilisant de la redondance de type FEC (*Foward Error Correction*), soit par des mécanismes prédictifs au niveau du récepteur – et la congestion lorsqu'il est possible d'absorber la gigue par une mémoire tampon de réception bien dimensionnée. Dans ce contexte, « tolérer » signifie sans perte rédhibitoire de satisfaction pour les utilisateurs. La communauté des utilisateurs du Mbone est restreinte et « civilisée ». Si quelqu'un utilise beaucoup de bande passante (par exemple, pour un flux vidéo de bonne qualité) à un moment où il y a peu de sessions multipoints sur le Mbone, il va naturellement diminuer le débit de son flux si le nombre de sessions augmente, pour éviter que son flux ne pénalise les autres sessions.

Cependant, la transmission multipoint est une idée beaucoup trop ambitieuse pour être confi-

née au Mbone. Il est, par conséquent, naturel d'étudier comment fiabiliser la transmission multipoint et comment faire du contrôle de congestion pour diverses applications multipoints dans un réseau *best effort* de type Internet. La fiabilité en multipoint est plus complexe qu'en point à point pour deux raisons qui apparaissent principalement avec les grands groupes. Premièrement, la question de l'envoi des acquittements (*feedback*) est beaucoup plus complexe en multipoint qu'en point à point : lorsqu'un grand nombre de récepteurs envoie du *feedback* à la source pour signaler, par exemple, une perte commune, la source peut s'écrouler sous le trop grand nombre de messages (*feedback implosion*). Plusieurs solutions ont été proposées à ce problème [8, 59, 66, 32, 84]. Deuxièmement, la question des retransmissions est également plus complexe en multipoint qu'en point à point . Là encore, plusieurs solutions ont été proposées [59, 66]. En résumé, la question de la fiabilité en multipoint a été beaucoup étudiée et de nombreuses solutions élégantes et performantes ont été proposées.

Le contrôle de congestion multipoint est beaucoup plus complexe que le contrôle de congestion point à point car en multipoint on a une source mais plusieurs récepteurs. On peut considérer un mécanisme de contrôle de congestion point à point comme un mécanisme distribué qui doit optimiser l'utilisation des ressources du réseau. Pour faire du contrôle de congestion point à point, on doit non seulement tenir compte de la source et du récepteur, mais aussi de tous les autres flux exogènes à la connexion. En effet, il faut maximiser, par exemple, le débit de sa propre connexion sans pénaliser les autres connexions. Lorsque l'on fait du contrôle de congestion multipoint, on doit optimiser l'utilisation des ressources du réseau avec la contrainte supplémentaire, par rapport à la transmission point à point, qu'il y a une corrélation entre les débits que reçoivent les récepteurs d'un même groupe multipoint puisqu'ils appartiennent au même arbre de distribution. Il n'y a pas, contrairement à ce qu'offre TCP pour la transmission point à point, une solution générale pour le contrôle de congestion multipoint, mais plusieurs solutions spécifiques qui seront détaillées dans le chapitre 2.

La transmission multipoint permet donc un gain considérable de bande passante et, par conséquent, le déploiement de nouveaux services dans les réseaux *best effort* comme la diffusion de contenu audio et vidéo de bonne qualité. Même si le contrôle de congestion multipoint est très complexe, il est cependant nécessaire au déploiement d'applications multipoints. On va, dans cette thèse, étudier le problème du contrôle de congestion dans les réseaux *best effort* en s'orientant plus particulièrement vers la transmission multipoint.

1.4 Organisation de la thèse

Cette thèse est organisée de la manière suivante. Dans le chapitre 2 on va donner l'état de l'art du contrôle de congestion pour la transmission multipoint. Dans le chapitre 3 on résumera les contributions de cette thèse avant de conclure au chapitre 4. On trouvera en annexe quatre

chapitres en anglais qui correspondent aux quatre contributions de cette thèse résumées au chapitre 3. Il est conseillé de commencer par lire le chapitre 3 pour avoir une vision globale de la thèse, mais de lire les annexes pour connaître les détails sur une partie précise. Le chapitre 3 comprend quatre parties, chaque partie correspondant à un chapitre placé en annexe. Dans la première partie, on étudie les comportements pathologiques de deux protocoles de contrôle de congestion multipoints RLM et RLC (annexe A). Dans la deuxième partie, on étudie de manière formelle le problème du contrôle de congestion et on introduit la notion de paradigme FS (annexe B). Dans la troisième partie, on introduit PLM, un nouveau protocole de contrôle de congestion multipoint basé sur le paradigme FS et qui surpasse tous les autres protocoles de contrôle de congestion multipoints (annexe C). Dans la quatrième et dernière partie, on étudie des mécanismes d'allocation de la bande passante entre flux multipoints et flux point à point dépendant du nombre de récepteurs (annexe D).

Chapitre 2

État de l'art

Le contrôle de congestion pour la transmission multipoint est le sujet d'actives recherches depuis quelques années. Contrairement à la transmission point à point où un seul protocole de contrôle de congestion peut satisfaire la grande majorité des utilisateurs, la transmission multipoint nécessite plusieurs types de protocoles de contrôle de congestion en fonction du type d'applications utilisées. On va classer ces protocoles de contrôle de congestion multipoints en fonction du type d'architectures utilisées (orientées source ou récepteur) et du type de comportements choisis (*TCP-friendly* ou non *TCP-friendly*), chaque architecture et chaque comportement ayant des avantages et des inconvénients que l'on va détailler dans la suite.

2.1 Architecture du protocole

2.1.1 L'architecture orientée source

Ce type d'architectures est utilisé pour les protocoles de contrôle de congestion point à point et en particulier pour TCP. Dans une architecture orientée source, la responsabilité de l'adaptation du débit de la session aux conditions de congestion du réseau est laissée à la source. Tous les récepteurs de la session observent le même débit, celui de la source. Étant donné que, en général, la source doit s'adapter au récepteur le plus lent, les récepteurs disposant d'une plus grande bande passante seront pénalisés. Même dans le cas où la source envoie les données à un débit supérieur à celui du récepteur le plus lent, le débit unique de la session, inhérent à l'architecture orientée source, ne pourra pas satisfaire tous les utilisateurs en cas d'une grande hétérogénéité de la bande passante disponible pour chaque récepteur. Par conséquent, l'architecture orientée source s'adapte mal aux groupes hétérogènes et doit être réservée aux groupes homogènes. On note, cependant, que même dans le cas de groupes homogènes, l'architecture orientée source présente de nombreuses difficultés. La corrélation des pertes entre les récepteurs [89] rend la découverte du taux de pertes de la session multipoint complexe ; la découverte du RTT est difficile

avec une architecture orientée source, mais également avec une architecture orientée récepteur. On discutera de quelques problèmes liés à la découverte du RTT à la fin de ce paragraphe. Le principal intérêt de cette architecture est qu'elle semble, de prime abord, plus simple à mettre en œuvre que l'architecture orientée récepteur. En effet, ce type d'architectures est bien connu pour le problème du contrôle de congestion point à point et il peut sembler facile de l'étendre à la transmission multipoint. Deux types de mécanismes doivent être considérés pour déterminer le débit de la source dans le cas d'une l'architecture orientée source : les mécanismes orientés fenêtre (*window-based*) et les mécanismes orientés débit (*rate-based*).

Golestani et al. [34] ont étudiés comment étendre ces types de mécanismes à la transmission multipoint. Ils ont montré que pour obtenir une équité de type TCP avec un mécanisme orienté débit la connaissance explicite du RTT est nécessaire alors que ce n'est pas le cas avec un mécanisme orienté fenêtre. De plus, ils ont montré que lorsque l'on applique un mécanisme orienté fenêtre à la transmission multipoint, il est sous optimal de considérer la même fenêtre pour tous les récepteurs. Pour résoudre ce problème, ils ont proposé de maintenir une fenêtre par récepteur. On va, dans la suite, détailler ce qu'est un mécanisme orienté fenêtre et ce qu'est un mécanisme orienté débit.

Un mécanisme orienté fenêtre correspond au type de mécanismes utilisé par TCP. Les récepteurs acquittent chaque paquet et à chaque fois qu'un paquet a été acquitté par tous les récepteurs, la fenêtre d'émission est ouverte. L'inconvénient de ce mécanisme est que les récepteurs doivent acquitter chaque paquet (ACK based). Pour éviter une implosion de la source, le mécanisme de *feedback* doit utiliser une structure hiérarchique pour agréger les acquittements. Le principal avantage d'un mécanisme orienté fenêtre est qu'il permet de facilement imiter le comportement de TCP et, par conséquent, d'être *TCP-friendly* (voir § 2.2.1). Étant donné que les protocoles de fiabilité multipoints utilisent souvent une structure hiérarchique, un protocole de contrôle de congestion multipoint peut être couplé à un tel protocole. Le protocole de transport multipoint fiable RMTP [66] utilise une structure hiérarchique qui permet d'agréger les acquittements (ACK) en utilisant des récepteurs désignés (DR) chargés de collecter les acquittements pour la zone dont ils sont responsables. Chaque DR envoie des acquittements à la source en fonction des acquittements reçus des récepteurs de sa zone. RMTP utilise un mécanisme de contrôle de congestion, exploitant cette structure, basé sur un mécanisme orienté fenêtre. D'autres protocoles sont hybrides ACK/NACK où les ACK sont, en général, toujours responsables de l'ouverture de la fenêtre, mais où les NACK peuvent avoir des rôles divers. MTCP [73] est un protocole hybride ACK/NACK qui utilise une structure en arbre pour agréger le *feedback* des récepteurs, indépendamment de tout protocole de fiabilité. Étant donné que les nœuds de l'arbre sont des récepteurs de la session appelés *sender's agent* (SA), MTCP n'a pas besoin d'un support du réseau pour agréger le *feedback*. Le protocole pgmcc [75] est également hybride ACK/NACK. Les ACK permettent d'ouvrir la fenêtre et de détecter les pertes

après 3 ACK dupliqués ou après un certain délai sans ACK, alors que les NACK permettent de choisir le récepteur responsable d'envoyer les ACK : le *acker*.

Un mécanisme orienté débit autorise la source à envoyer un flux continu de données et c'est le *feedback* des récepteurs, généralement des acquittements négatifs (NACK), qui permet de savoir quand augmenter ou diminuer le débit de la source. Un mécanisme orienté débit est plus facile à mettre en œuvre que son homologue orienté fenêtre. En effet, un mécanisme orienté fenêtre a besoin d'un mécanisme d'agrégation des ACK qui est complexe à mettre en place. Par contre, un mécanisme orienté débit peut conduire à une implosion de la source, en cas de congestion, due aux NACK émis par les récepteurs. Pour résoudre ce problème, des mécanismes de suppression des NACK sont utilisés [8, 60]. Cependant, en diminuant la fréquence du *feedback* on risque d'avoir une vue inconsistante de l'état de congestion du réseau ; on a ici un compromis qui n'est pas facile à trouver. DeLucia et al. [19] ont introduit un protocole orienté débit hybride ACK/NACK. Ils appellent les ACK des *Congestion Clear* (CC) et les NACK des *Congestion Indication* (CI). Les CC sont utilisés pour augmenter le débit de la source alors que les CI sont utilisés pour diminuer le débit de la source et pour élire les récepteurs (*representatives*) responsables d'envoyer les CC à la source.

Que ce soit dans le cas d'un mécanisme orienté fenêtre ou d'un mécanisme orienté débit l'évaluation du RTT est souvent nécessaire, mais complexe à faire. La connaissance du RTT est fondamentale si le protocole veut être *TCP-friendly* (voir § 2.2.1). Golestani et al. [34] ont montré qu'il fallait la connaissance explicite du RTT pour obtenir une équité de type TCP avec un mécanisme orienté débit, mais que la connaissance explicite du RTT n'était pas nécessaire pour obtenir une équité de type TCP avec un mécanisme orienté fenêtre, parce que le RTT est implicitement contenu dans la boucle de *feedback*, comme pour TCP. En effet, TCP n'a pas besoin de la connaissance explicite du RTT pour faire du contrôle de congestion, il en a besoin pour la fiabilité et éviter les retransmissions inutiles. Cependant, on a vu qu'en multipoint la boucle de *feedback* était rompue. Les mécanismes d'agrégation des ACK engendrent des délais supplémentaires dans la boucle de *feedback*. Par conséquent, même pour un protocole orienté fenêtre on a besoin de l'estimation explicite du RTT. Dans MTCP, Rhee et al. [73] introduisent la notion de *Relative Time Delay* (RTD) qui est utilisé à la place du RTT. Le cas de pgmcc est différent puisque le source élit un récepteur qui sera chargé de lui envoyer les ACK. Donc la source pgmcc n'a pas besoin de la connaissance explicite du RTT pour avoir un comportement de sa fenêtre d'émission qui soit compatible avec TCP. Cependant, pour être compatible avec TCP, pgmcc doit toujours choisir comme *acker* le récepteur le plus lent. Or, pour connaître ce récepteur, il faut avoir une estimation du RTT et du taux de pertes de tous les récepteurs, estimations obtenues grâce aux acquittements négatifs (NACK) envoyés périodiquement par les récepteurs à la source. Le récepteur le plus lent est choisi en comparant les récepteurs avec une fonction en $\frac{1}{\mathrm{RTT}\cdot\sqrt{\mathrm{loss}}}$. Cependant, les mécanismes de suppression des NACK rendent

l'estimation du RTT approximative.

2.1.2 L'architecture orientée récepteur

L'architecture orientée récepteur implique que c'est aux récepteurs de décider s'il faut augmenter ou diminuer le débit. Cette architecture a été rendue possible grâce au support des protocoles de routage multipoints [18, 16, 17]. La source envoie les données en les découpant en couches cumulatives et en envoyant chaque couche dans un groupe multipoint différent. La principale propriété d'un découpage en couches cumulatives est qu'à chaque fois que l'on ajoute une couche, on augmente le débit. Chaque récepteur recevra le même contenu, mais à des vitesses différentes en fonction du nombre de groupes multipoints – on utilise également le terme couche à la place de groupe multipoint – auxquels il est abonné. Dans le § 3.1.1 on introduira l'architecture orientée récepteur et la notion de couches cumulatives dans le contexte des protocoles RLM [55] et RLC [87]. Les récepteurs utilisent un mécanisme de découverte de la bande passante pour connaître l'état de congestion de réseau et ils s'abonnent ou se désabonnent à des couches en fonction de cet état. L'avantage de cette architecture est que, contrairement à l'architecture orientée source, chaque récepteur peut utiliser la bande passante qu'il existe sur le chemin entre la source et lui. Cependant, cette architecture requiert un codage à la source pour obtenir les couches cumulatives et la granularité des couches ne permet pas d'exactement utiliser toute la bande passante entre la source et chaque récepteur. De plus, les abonnements et désabonnements aux couches génèrent de la signalisation au niveau du protocole de routage multipoint. Cette architecture est parfaitement adaptée à la diffusion de contenus multimédias à un large groupe hétérogène d'utilisateurs, mais peut également être utilisée pour la distribution de données [86]. Peu de protocoles utilisent cette architecture, principalement RLM [55] et RLC [87]. Linda Wu et al. [88] ont introduit un nouveau protocole de contrôle de congestion multipoint orienté récepteur et basé sur l'utilisation de couches fines (ThinStreams) qui permet de découpler le contrôle de congestion du codage des données multimédias. Turletti et al. [85] ont introduit une version de RLM compatible avec TCP (on donnera quelques détails sur ce protocole un peu plus loin). Rubenstein et al. [77] ont discuté de l'impact sur l'équité d'une architecture orientée récepteur couplée avec l'envoi des données en couches cumulatives. Ils ont montré que cette architecture permettait d'obtenir plusieurs types d'équité et en particulier l'équité max-min [5].

Sisalem et al. [80] ont introduit MLDA, un protocole hybride orienté source/orienté récepteur. Ce protocole se comporte comme un protocole orienté récepteur classique, mais, périodiquement, la source collecte des informations sur la bande passante que voient les récepteurs et ajuste la distribution des couches en fonction de ces informations.

2.2 Comportement du protocole

2.2.1 Le comportement *TCP-friendly*

Le comportement *TCP-friendly* implique que le débit de la session doit être conforme à ce qu'utiliserait un flux TCP dans les mêmes conditions. Plusieurs approximations du débit de TCP ont été introduites [54, 64] ; cependant, l'équation introduite par Padhy [64] est la seule qui fournisse toujours une bonne approximation du débit d'un flux TCP même pour les forts taux de pertes. Le débit d'un flux TCP est toujours fonction du RTT (*Round Trip Time*) et du taux de pertes en $\frac{1}{\text{RTT}\cdot\sqrt{\text{loss}}}$. Par conséquent, la principale contrainte lorsque l'on veut être *TCP-friendly* est de connaître le RTT et le taux de pertes. La notion de RTT dans une session multipoint est mal définie. En effet, le RTT entre la source et chaque récepteur peut être différent.

Pour être *TCP-friendly*, dans le cas d'une architecture orientée source, il faut s'adapter au récepteur le plus lent. Ce dernier est choisi d'après une fonction en $\frac{1}{\text{RTT}\cdot\sqrt{\text{loss}}}$. Il suffit, donc, de connaître le RTT et le taux de pertes entre la source et ce récepteur. Cependant, on a vu au § 2.1.1 que l'estimation du RTT et du taux de pertes n'étaient pas facile. Les protocoles pgmcc [75] et MTCP [73] sont des protocoles orientés source avec un comportement *TCP-friendly*. MLDA [80] est également un protocole *TCP-friendly* qui est hybride orienté source/orienté récepteur.

Dans le cas d'une architecture orientée récepteur, chaque récepteur peut adapter son débit à l'état de congestion du réseau. Par conséquent, chaque récepteur devra connaître le RTT entre la source et lui. Un des avantages de l'architecture orientée récepteur est qu'elle ne nécessite pas de *feedback* entre les récepteurs et la source. Cet avantage devient un inconvénient lorsque l'on veut que le protocole soit *TCP-friendly* ; en effet, lorsqu'il n'existe pas une boucle complète de *feedback*, c'est-à-dire lorsque la source envoie un paquet au récepteur et que le récepteur, à la réception du paquet, renvoie un paquet à la source, il est impossible de déterminer le RTT. Dans ce cas, on doit ajouter un mécanisme spécifique pour obtenir ce RTT. Dans le cas de liens symétriques, le OTT (*One Trip Time*), qui représente le temps que met un paquet pour aller de la source au récepteur, peut donner une bonne approximation du RTT en prenant $\text{RTT} = 2 \cdot \text{OTT}$. Turletti et al. [85] ont introduit une version *TCP-friendly* du protocole RLM. Ils expliquent que le plus difficile pour rendre RLM *TCP-friendly* est d'avoir un bonne estimation du RTT. Pour cela, ils proposent trois solutions et discutent les mérites respectifs de ces solutions.

2.2.2 Le comportement non *TCP-friendly*

Si le débit d'une session n'est pas une fonction en $\frac{1}{\text{RTT}\cdot\sqrt{\text{loss}}}$ et, plus généralement, si le débit ne suit pas les équations données en [54, 64], cette session n'a pas un comportement *TCP-*

friendly. Un protocole qui n'est pas *TCP-friendly* est difficile à déployer dans l'internet parce qu'un tel protocole peut énormément pénaliser les flux TCP. Cependant, certains protocoles essaient de suivre un comportement de type TCP sans pour autant être *TCP-friendly*. C'est le cas, notamment, de RLC [87] qui est un protocole *TCP-like* mais pas *TCP-friendly*. RLC est *TCP-like* parce que le débit entre la source et un récepteur donné diminue de manière exponentielle en cas de pertes sur le chemin entre la source et ce récepteur, « à la manière de TCP » ; cependant, il ne peut pas être TCP-friendly parce qu'il est indépendant du RTT.

Le principal avantage d'un protocole non *TCP-friendly* est qu'il devrait être plus facile à concevoir et plus efficace. En effet, sans la contrainte d'avoir un débit en $\frac{1}{\mathrm{RTT}\cdot\sqrt{\mathrm{loss}}}$ le protocole peut être plus efficace parce qu'il peut être beaucoup plus agressif que TCP. Cependant, en pratique, le problème est beaucoup plus complexe. Le comportement *TCP-friendly* règle le débit de la session, mais garantit, également, l'équité et la stabilité du protocole. Par conséquent, en suivant une seule équation on garantit trois propriétés fondamentales pour un protocole de contrôle de congestion. Lorsque le protocole n'est pas *TCP-friendly*, on doit trouver de nouveaux mécanismes pour garantir ces propriétés. Le protocole RLM [55] est un exemple des problèmes qui se posent avec les protocoles non *TCP-friendly*. Ce protocole n'est ni *TCP-friendly*, ni *TCP-like* et on va voir au § 3.1 qu'il n'est ni stable, ni équitable, ni très efficace. Une des contributions majeures de cette thèse est de décrire un cadre formel pour la conception de nouveaux protocoles de contrôle de congestion qui ne soient pas *TCP-friendly* mais qui soient beaucoup plus efficace qu'un protocole *TCP-friendly* (voir § 3.2). Une autre contribution majeure est d'avoir conçu, dans ce cadre formel, un nouveau protocole de contrôle de congestion multipoint orienté récepteur qui surpasse largement tous les autres protocoles de contrôle de congestion multipoints orientés récepteur (voir § 3.3).

2.3 Conclusion

On a vu qu'il existait une grande variété de protocoles de contrôle de congestion multipoints, chaque type de protocoles ayant des avantages et des inconvénients. Le tableau 2.1 récapitule les propriétés de quelques protocoles de contrôle de congestion multipoints.

On peut finalement noter que certains travaux ne portant pas sur les protocoles de contrôle de congestion multipoints, mais portant sur le problème du contrôle de congestion en général ont inspiré notre travail. Lefelhocz et al. [46] ont discuté de la nécessité d'avoir un nouveau paradigme pour le contrôle de congestion. Ils proposèrent quatre mécanismes nécessaires au contrôle de congestion : ordonnancement, gestion du débordement des files d'attente, *feedback* et mécanisme d'adaptation aux systèmes terminaux. Cependant, leur étude reste informelle et ne présente pas de solutions pour la conception de nouveaux protocoles de contrôle de congestion. Shenker [79] applique la théorie des jeux à l'étude du contrôle de congestion. Il montre que l'on

	RLM	RLC	MLDA	pgmcc	MTCP
orienté source			+	+	+
orienté récepteur	+	+	+		
TCP-friendly				+	+
TCP like		+			

TAB. 2.1 – *Quelques protocoles de contrôle de congestion multipoints et leurs principales caractéristiques.*

peut obtenir des propriétés intéressantes pour un protocole de contrôle de congestion avec des utilisateurs égoïstes et non collaborant si l'on a une fonction d'allocation de la bande passante qui soit équitable (*fair share allocation function*). Cette étude est restée beaucoup trop abstraite pour s'appliquer à un problème concret. La thèse de Keshav [44] nous a fourni une solide base de travail. Keshav a introduit l'utilisation de *Fair Queueing* (FQ) pour le contrôle de congestion point à point. Il a également introduit la technique de l'envoi des paquets par paire appliquée au contrôle de congestion point à point. Cependant, alors que Keshav présentait une solution pour un protocole de contrôle de congestion point à point, nous allons, dans la suite, étudier le problème du contrôle de congestion d'un point de vue général et ensuite appliquer cette étude à la conception d'un nouveau protocole de contrôle de congestion multipoint. On peut considérer une partie de cette thèse, en particulier les § 3.2 et § 3.3, comme une généralisation du travail de Keshav. D'autres auteurs ont étudié le problème du contrôle de congestion, mais d'un point de vue très éloigné du notre. Kelly [43, 42] a étudié l'impact de la facturation d'un service (*pricing*) sur l'équité et la stabilité du réseau. Balakrishnan et al. ont introduit la notion *Congestion Manager* (CM) qui est responsable de fournir aux applications les informations nécessaires à leur adaptation aux conditions de congestion du réseau.

Chapitre 3

Contributions de la thèse

Ce chapitre est divisé en quatre parties, chaque partie étant le résumé d'un chapitre placé en annexe. On va insister, ici, sur l'articulation logique entre chaque partie ; articulation nécessaire pour former une thèse cohérente. La première partie présente une étude des comportements pathologiques de deux protocoles de contrôle de congestion multipoints à couches cumulatives et orientés récepteur : RLM [55] et RLC [87]. Il est cependant extrêmement difficile de corriger les comportements pathologiques de ces protocoles dans le contexte actuel de l'internet. On a alors réfléchi au problème du contrôle de congestion dans le contexte plus général des réseaux *best effort*. Ceci nous a conduit à redéfinir la notion de congestion, puis définir les propriétés requises par un protocole de contrôle de congestion idéal et enfin définir un nouveau paradigme pour la conception de protocoles de contrôle de congestion presque idéaux. On a introduit à cet effet le paradigme *Fair Scheduler* ou paradigme FS. L'approche que l'on a utilisée pour définir ce nouveau paradigme est purement formelle. Pour valider cette approche théorique du contrôle de congestion, on a conçu, grâce au paradigme FS, un nouveau protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur : PLM. Ce protocole surclasse RLM et RLC. Comme le paradigme FS permet de concevoir des protocoles de contrôle de congestion multipoints et point à point, on s'est posé la question suivante : « Comment allouer la bande passante entre un flux multipoint qui possède un million de récepteurs et un flux point à point avec un seul récepteur ? » On a donné une réponse rigoureuse et originale en introduisant une nouvelle politique d'allocation de la bande passante qui tient compte du nombre de récepteurs. De plus, cette politique s'intègre parfaitement dans la discipline *Fair Scheduler*, mécanisme de base du paradigme FS.

3.1 Comportements pathologiques de RLM et RLC

3.1.1 Introduction

Le problème du contrôle de congestion pour la transmission multipoint est ardu. Pour avoir une idée précise des problèmes liés au contrôle de congestion multipoint, nous allons étudier, dans cette partie, deux protocoles de contrôle de congestion multipoints populaires RLM [55] et RLC [87]. L'étude de ces protocoles ne prétend pas couvrir de manière exhaustive les problèmes liés au contrôle de congestion multipoint, mais elle va permettre d'identifier quelques problèmes fondamentaux qui vont orienter notre étude.

RLM et RLC sont deux protocoles de contrôle de congestion multipoints à couches cumulatives et orientés récepteur. Encoder et découper des données multimédias en n couches cumulatives $L_1, \cdots, L_n$ signifie que chaque sous ensemble $\{L_1, \cdots, L_i\}_{i \leq n}$ a le même contenu mais avec une qualité qui augmente avec i. Ce genre de codage est parfaitement adapté au contenu audio et vidéo. Une fois que l'on a une organisation en couches cumulatives des données multimédias, il est aisé d'envoyer chaque couche dans un groupe multipoint différent. Dans la suite, on utilisera indifféremment la terminologie *groupe multipoint* et *couche* pour désigner un groupe multipoint qui transporte une seule couche. Ce type de découpage et d'émission du contenu multimédia est très efficace lorsqu'il est utilisé avec un protocole de contrôle de congestion multipoint orienté récepteur. Pour un tel protocole, la source a un rôle passif. Elle envoie simplement chaque couche dans un groupe multipoint différent. Le récepteur s'abonne ou se désabonne aux couches en se basant sur sa connaissance de la bande passante disponible pour le flux qu'il reçoit. Cette connaissance lui est fournie par un mécanisme de découverte de la bande passante disponible. C'est ce mécanisme qui détermine les propriétés du protocole.

Un protocole de contrôle de congestion multipoint utilisant des couches cumulatives et orienté récepteur est actuellement la solution la mieux adaptée à la distribution de contenu multimédia à un groupe hétérogène de récepteurs. Steven McCanne et al. ont été les premiers à introduire un protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur, RLM [55]. Le comportement de RLM est déterminé par une machine à états finis dont les transitions sont déclenchées par des expirations de *timers* ou par la détection de pertes. Pour être robuste à l'augmentation du nombre de récepteurs, le mécanisme du *shared learning* a été ajouté. On détaillera au § 3.1.2.1 les différents mécanismes de RLM. McCanne et al. évaluèrent RLM pour des scénarios simples. Ils trouvèrent qu'il n'y a pas d'équité entre les sessions RLM. Bajaj et al. [2] explorèrent les avantages respectifs des pertes de paquets uniformes et des pertes de paquets avec priorités au niveau des files d'attente du réseau dans le contexte de la transmission de vidéo en couches. Ils trouvèrent que le comportement de RLM était satisfaisant excepté dans certains cas extrêmes de trafic émis en rafale. Gopalakrishnan et al. [35] étudièrent le comportement de RLM pour des couches VBR (*Variable Bit Rate*). Ils trouvèrent une grande

instabilité de RLM, une faible utilisation de la bande passante ainsi qu'un manque d'équité.

Vicisano introduisit une version *TCP-like* de RLM appelée RLC [87]. Elle est basé sur la génération périodique, par la source, de rafales de paquets qui sont utilisées pour la découverte de la bande passante, et sur des points de synchronisation utilisés par les récepteurs pour savoir quand ajouter une couche. On dit que RLC est *TCP-like* (par opposition à *TCP-friendly*) parce que la distribution du débit des couches est exponentielle. Lorsqu'un récepteur quitte une couche suite à de la congestion, il y a une diminution exponentielle du débit à la manière de TCP (*TCP-like*). Par contre, RLC étant indépendant du RTT, il ne peut être *TCP-friendly*. Vicisano et al. ont trouvé que RLC pouvait être non équitable avec TCP pour des grandes tailles de paquets. On n'a pas connaissance d'autres études sur RLC.

On voit que, d'après les études précédentes, RLM et RLC semblent se comporter raisonnablement bien excepté dans certains cas particuliers. On va cependant montrer que même dans des scénarios simples, ces deux protocoles ont des comportements pathologiques fondamentaux. Les problèmes rencontrés sont pathologiques parce qu'ils diminuent fortement les performances des protocoles ; ils sont fondamentaux parce qu'ils sont inhérents aux protocoles et ne peuvent pas être corrigés par un simple ajustement de paramètres. On note que la notion de comportement pathologique est liée à un environnement de type Internet. En effet, dans certains environnements simplifiés (bande passante garantie, pas d'interaction avec d'autres protocoles, etc.) RLM et RLC pourraient fonctionner correctement. Cependant, la finalité est d'avoir un protocole qui permette le déploiement d'un service multipoint dans l'internet.

3.1.2 Les comportements pathologiques de RLM

RLM (*Receiver-driven Layered Multicast*) [55] a été introduit par Steven McCanne et al. en 1996. RLM est un protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur pour la dissémination de contenu vidéo à un groupe hétérogène de récepteurs.

3.1.2.1 Rappels sur RLM

Une source RLM encode en couches cumulatives le flux vidéo et envoie chaque couche dans un groupe multipoint différent. En fait, toute la « machinerie » du protocole est au niveau du récepteur. Celui-ci s'abonne ou se désabonne à des groupes multipoints en fonction de la bande passante disponible ou de la congestion du réseau. Ainsi, chaque récepteur peut s'adapter à l'état de congestion du réseau sur le chemin entre la source et lui. Le comportement d'un récepteur RLM est déterminé par une machine à états finis dont les transitions sont déclenchées par des expirations de *timers* ou par la détection de pertes.

Dire qu'un récepteur fait un *join-experiment* signifie qu'il ajoute expérimentalement une couche et regarde si cette couche produit de la congestion. Si elle produit de la congestion, il

n'y a pas assez de bande passante pour recevoir cette couche. Alors le récepteur quitte cette couche, on dit que le *join-experiment* a échoué. Si elle ne produit pas de congestion, il y a assez de bande passante pour recevoir cette couche et le récepteur conserve son abonnement à cette couche, on dit que le *join-experiment* a réussi. Un récepteur RLM maintient deux *timers* : un *join-timer* T_j et un *detection-timer* T_d. Le *join-timer* définit la fréquence des *join-experiments*, le *detection-timer* est une estimation du temps jugé nécessaire pour décider si un *join-experiment* a réussi. Le mécanisme de découverte de la bande passante est le suivant : un récepteur fait un *join-experiment* tous les T_j unités de temps et décide que le *join-experiment* a réussi s'il n'a pas observé de pertes durant un intervalle de temps T_d après le début du *join-experiment*. S'il observe des pertes durant cet intervalle de temps, il juge que le *join-experiment* a échoué et augmente le *join-timer* correspondant à la couche qui n'a pas pu être ajoutée. Si le récepteur observe des pertes en dehors d'un *join-experiment*, il va entrer dans un état *hysteresis* qui est destiné à absorber les périodes transitoires de congestion. Après une période T_d dans cet état, le récepteur mesure le taux de pertes et quitte une couche si le taux de pertes est supérieur à un seuil de 25%. Un récepteur ne peut cependant quitter qu'une seule couche par période T_d.

Ce mécanisme de découverte de la bande passante ne fonctionne pas correctement lorsque l'on augmente le nombre de récepteurs. Pour résoudre ce problème McCanne et al. ont introduit le *shared learning* : lorsqu'un récepteur fait un *join-experiment*, il en notifie le groupe entier en envoyant un message indiquant la couche ajoutée expérimentalement. Tous les récepteurs après avoir reçu ce message annuleront leurs *join-experiments* aux couches supérieures à celle annoncée. L'idée est d'éviter une congestion due à un *join-experiment* à une couche supérieure qui serait mal interprétée par un *join-experiment* à une couche inférieure qui se déroule au même moment. Tous les récepteurs observant de la congestion durant un *join-experiment* annoncé vont déduire que ce *join-experiment* a échoué et vont augmenter leur T_j pour la couche correspondante.

3.1.2.2 Comportements pathologiques de RLM

On ne fait, dans ce paragraphe, que résumer nos résultats : tous les détails sur les comportements pathologiques de RLM peuvent être trouvés en annexe A.3. On a trouvé cinq mécanismes qui conduisent à des comportements pathologiques :

– la valeur minimale du *join-timer* définit une borne inférieure à la vitesse de convergence de RLM. En effet, un récepteur ne peut ajouter qu'une seule couche tous les T_j. Cependant, cette valeur minimale du *join-timer* est le résultat d'un compromis entre vitesse de convergence et congestion due aux *join-experiments*. Il est par conséquent très difficile d'en trouver une valeur optimale.

– le grand seuil de pertes (fixé par McCanne à 25%) peut conduire à un taux de pertes très élevé. En effet, un taux de pertes persistant de 24% ne sera pas suffisant pour qu'un récepteur quitte une couche. D'autre part, ce grand seuil rend RLM très agressif avec TCP. On observe ce comportement agressif lorsque RLM est déjà abonné à plusieurs couches ; dans ce cas, les flux TCP sont incapables de produire suffisamment de congestion pour que RLM quitte des couches et libère ainsi de la bande passante. Cependant, le seuil de pertes est un compromis entre comportement réactif et conservateur en cas de pertes. Il faut souligner que RLM a été conçu pour la vidéo, application peu sensible aux pertes mais très sensible aux fréquents changements de qualité caractérisés par des oscillations des abonnements aux couches. Par conséquent, RLM doit être conservateur en cas de pertes.

– le mécanisme du *shared learning* conduit à une synchronisation des récepteurs. En effet, un récepteur qui fait un *join-experiment* empêche un *join-experiment* des autres récepteurs à une couche plus élevée. Par conséquent, l'abonnement aux couches des récepteurs se fait par palier. Cependant, le *shared learning* est un composant principal de RLM et le modifier reviendrait à refondre entièrement RLM.

– le mécanisme de *join-experiment* rend RLM très conservateur avec TCP. En effet, un récepteur RLM ne peut ajouter une couche que s'il ne voit pas de pertes durant toute la durée du *join-experiment*, c'est-à-dire pendant une période T_d. Or, lorsque RLM partage un goulot d'étranglement avec TCP, à cause des pertes périodiques engendrées par TCP à la fin de chaque cycle, RLM ne peut jamais ajouter de couche. Le mécanisme de *join-experiment* est un composant principal de RLM et il est très difficile de modifier ce mécanisme sans entièrement modifier RLM.

– RLM est très conservateur en cas de pertes. En effet, un récepteur ne peut quitter qu'une couche par période de T_d secondes. Le résultat est un très fort taux de pertes transitoires en cas de forte congestion ; par exemple, lorsqu'il faut quitter deux couches afin de s'adapter à la bande passante disponible, il faudra au minimum $2 * T_d$ secondes pour quitter ces deux couches.

On a donc identifié plusieurs comportements pathologiques de RLM dus à des mécanismes propres à RLM. On a également vu que ces mécanismes étaient très difficiles à modifier pour éviter les comportements pathologiques. On peut noter cependant qu'un mécanisme efficace de découverte de la bande passante pourrait résoudre tous les problèmes, sauf celui de synchronisation des récepteurs dû au *shared learning*.

3.1.3 Les comportements pathologiques de RLC

RLC [87] a été introduit par Lorenzo Vicisano et al. en 1998. RLC est un protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur. Cependant, à la différence de RLM, RLC a été proposé pour l'audio, la vidéo et le transfert de fichiers.

3.1.3.1 Rappels sur RLC

Une source RLC encode en couches cumulatives les données et envoie chaque couche dans un groupe multipoint différent. Le débit de chaque couche est distribué de manière exponentielle. Le mécanisme de découverte de la bande passante de RLC est basé sur la génération périodique, par la source, de rafales de paquets qui sont utilisées pour la découverte de la bande passante au niveau de chaque récepteur. La source double son débit sur une période de temps courte et fixe. Cette période d'augmentation du débit est immédiatement suivie d'une période silencieuse, de telle sorte que, en moyenne, le débit est constant. Le but de ces rafales de paquets périodiques est de simuler l'ajout d'une couche pour une courte période de temps. Si la file d'attente du goulot d'étranglement déborde avec cette rafale, il n'y a pas assez de bande passante disponible pour ajouter cette nouvelle couche ; dans le cas contraire, le récepteur peut ajouter une nouvelle couche. L'avantage mis en avant par les auteurs de RLC de ce mécanisme de découverte de la bande passante basé sur des rafales périodiques, par rapport à un mécanisme de découverte de la bande passante basé sur des *join-experiments* comme pour RLM est que les rafales produisent moins de congestion dans le réseau que les *join-experiments*. En effet, comme la durée d'une rafale est courte et de taille fixe, la congestion induite par cette rafale, s'il n'y a pas assez de bande passante pour ajouter une nouvelle couche, sera courte et de taille fixe ; à l'inverse, la congestion induite par un *join-experiment* dépend du temps que le récepteur va mettre à découvrir qu'il y a congestion et du temps qu'il va mettre à quitter cette couche expérimentale. On va voir, au § 3.1.3.2, que le mécanisme de découverte de la bande passante basé sur des rafales périodiques ne fonctionne pas.

RLC possède également un mécanisme de synchronisation des abonnements des récepteurs aux couches basé sur des points de synchronisation – un bit spécial dans un paquet de données. Il y a sur chaque couche des points de synchronisation espacés proportionnellement à la bande passante de la couche. Ces points sont placés à la fin d'une rafale et un récepteur ne peut ajouter une couche que lorsqu'il reçoit un point de synchronisation. Ils permettent de synchroniser les abonnements aux couches et d'éviter ainsi des sous utilisations de la bande passante ou des divergences de comportement dues à des récepteurs abonnés à des couches différentes mais qui partagent le même goulot d'étranglement.

Le mécanisme de découverte de la bande passante au niveau d'un récepteur est le suivant :

– un récepteur ajoute une couche lorsqu'il reçoit un point de synchronisation et qu'il n'a

pas détecté de pertes durant la rafale précédent ce point de synchronisation ;

– un récepteur quitte une couche dés qu'il détecte une perte. Cependant, un récepteur ne peut quitter plus d'une couche par *deaf period*. Une *deaf period* est une période de taille fixe qui sert à éviter les désabonnements de couches en cascade. Elle ne peut pas être ajustée dynamiquement durant la session.

Dés qu'un récepteur détecte une perte, il quitte une couche, avec la contrainte d'une couche maximum par *deaf period* ; comme les couches sont distribuées de manière exponentielle, le récepteur va diminuer son débit de manière multiplicative en cas de pertes, à la manière de TCP. C'est de là que vient la dénomination *TCP-like* de RLC. Par contre, RLC étant indépendant du RTT il ne peut pas être *TCP-friendly*.

3.1.3.2 Comportements pathologiques de RLC

On va résumer dans ce paragraphe nos résultats ; les détails peuvent être trouvés en annexe A.4. On a trouvé trois mécanismes qui conduisent à des comportements pathologiques de RLC :

– le mécanisme de découverte de la bande passante, basé sur la génération de rafales périodiques de paquets, ne fonctionne pas. En effet, pour fonctionner, ce mécanisme devrait simuler l'ajout d'une couche pendant une période suffisamment longue pour qu'il y ait des pertes au niveau du goulot d'étranglement, s'il n'y avait pas assez de bande passante pour ajouter cette nouvelle couche. Or les rafales sont périodiques et de taille fixe. En pratique, ces rafales ne font jamais déborder le goulot d'étranglement. Par conséquent, les récepteurs ne découvrent pas la bonne bande passante disponible et ajoutent une couche alors qu'il n'y a pas assez de bande passante. Cette couche va créer de la congestion dans le réseau et le récepteur va quitter cette couche. Cependant, comme les rafales sont périodiques, ce phénomène va se reproduire continuellement. En outre, il est très difficile d'améliorer ce mécanisme de découverte de la bande passante. En effet, le seul moyen serait d'avoir un mécanisme qui permette de découvrir la durée nécessaire des rafales pour faire déborder le goulot d'étranglement – en supposant que la source puisse effectivement modifier la durée des rafales –, ce qui reviendrait à avoir un mécanisme de découverte de la bande passante supplémentaire.

– la distribution des points de synchronisation dans RLC peut sérieusement ralentir la vitesse de convergence des récepteurs. En effet, les points de synchronisation à la couche $i+1$ sont un sous ensemble des points de synchronisation à la couche i. Par conséquent, périodiquement de 2 jusqu'à n (s'il y a n couches) points de synchronisation seront synchronisés ; c'est-à-dire que l'abonnement à 2 jusqu'à n couches (en fonction du nombre

de points de synchronisation synchronisés) sera possible au même moment. Cependant, s'il y a i points de synchronisation synchronisés et qu'il y a assez de bande passante pour $i-1$ couches mais pas pour i, alors la rafale de la couche i va produire des pertes. Tous les récepteurs en aval du même goulot d'étranglement vont déduire qu'ils ne peuvent pas ajouter une couche supérieure, quelle que soit cette couche. Ce problème est difficile à corriger car il implique de modifier la distribution des points de synchronisation. Même en décalant légèrement les points de synchronisation, le problème persiste puisqu'un récepteur ne peut ajouter une couche que s'il n'a pas détecté de congestion depuis la dernière rafale précédent ce point de synchronisation.

– le comportement *TCP-like* de RLC est dû, comme on l'a vu, à la distribution exponentielle des couches. Cependant, comme RLC est indépendant du RTT, il obtient une très faible fraction de la bande passante disponible lorsqu'il partage le goulot d'étranglement avec des connexions TCP ayant un petit RTT. Là encore, le problème est difficile à corriger. Une solution serait d'avoir une estimation du RTT. Mais premièrement, la notion de RTT est mal définie pour la transmission multipoint ; deuxièmement, étant donné que RLC est orienté récepteur (la source ne reçoit pas de *feedback* des récepteurs), il est impossible pour la source (ou le récepteur) d'avoir une estimation du RTT.

On a identifié pour RLC plusieurs comportements pathologiques. Comme pour RLM, un mécanisme efficace de découverte de la bande passante pourrait résoudre tous les problèmes – sauf celui des points de synchronisation.

3.1.4 Conclusion

L'étude des comportements pathologiques de RLM et RLC nous a permis de mettre en évidence plusieurs résultats fondamentaux. Il était couramment admis que RLM et RLC souffraient de quelques faiblesses. Cependant, ces protocoles étaient supposés pouvoir, au moins temporairement, offrir un service de contrôle de congestion pour la transmission multipoint. On a montré qu'en fait ces deux protocoles souffraient de problèmes fondamentaux qui rendaient leur déploiement irréaliste. On a vu, de plus, que le problème majeur commun aux deux protocoles était un mécanisme de découverte de la bande passante qui ne remplit pas sa tâche. Cependant, on ne peut pas corriger les comportements pathologiques des mécanismes de découverte de la bande passante par un simple ajustement de paramètres. De plus, dans le contexte actuel de l'internet, il est difficile d'améliorer significativement ces mécanismes de découverte de la bande passante. Plutôt que de se cantonner à essayer d'améliorer de manière empirique ces mécanismes de découverte de la bande passante, on a décidé de s'interroger sur la raison profonde de la difficulté de créer des protocoles de contrôle de congestion dans le contexte actuel de l'internet et, en particulier, sur la possibilité d'ajouter des mécanismes dans l'internet –

sans être en violation avec ses concepts de base – pour faciliter la conception de protocoles de contrôle de congestion et améliorer leur performance.

L'étude du contrôle de congestion prend ses racines dans la définition même de congestion. Comment définir un protocole de contrôle de congestion, c'est-à-dire un protocole destiné à éviter la congestion, si l'on n'a pas de définition précise de la notion de congestion ? La définition couramment admise pour la congestion est le débordement d'une file d'attente. Cependant, cette définition nous semble peu satisfaisante. Comment définir un protocole de contrôle de congestion si l'on ne sait pas quelles propriétés il doit avoir ? Parmi les propriétés couramment admises comme souhaitables on trouve l'équité et l'efficacité. Cependant, comment définir ces propriétés ? Par exemple, quand un protocole est-il efficace ? Ces propriétés sont-elles suffisantes ? Peut-on définir des règles qui permettent de concevoir des protocoles de contrôle de congestion efficaces ? C'est à toutes ces questions que l'on va répondre dans la suite de cette thèse.

3.2 Le paradigme *Fair Scheduler*

3.2.1 Introduction

On définit un paradigme pour le contrôle de congestion comme un modèle pour concevoir des protocoles de contrôle de congestion qui ont un même ensemble de propriétés. Tous les protocoles conçus avec le même paradigme seront compatibles au sens des propriétés communes garanties par le paradigme. En fait, un paradigme est un ensemble de contraintes à appliquer lors de la conception du protocole de contrôle de congestion. Cependant, cette notion de paradigme n'a jamais été clairement définie pour le contrôle de congestion dans l'internet. Le paradigme actuel, mais implicitement défini, est le paradigme *TCP-friendly* qui impose aux protocoles de suivre l'équation :

$$T = \frac{C * MTU}{RTT * \sqrt{loss}} \tag{3.1}$$

où T est le débit moyen de la connexion, C est une constante, MTU est la taille des paquets envoyés, RTT est le *Round Trip Time* et *loss* est le taux de pertes de la connexion. Il s'agit donc d'un paradigme qui impose à tous les utilisateurs [1] de collaborer, c'est-à-dire d'avoir une session avec un débit conforme à l'équation 3.1. Padhye et al. [64] ont introduit une meilleure approximation du débit de TCP pour les forts taux de pertes. Cependant, l'équation 3.1 est une bonne approximation de leur équation pour les faibles taux de pertes.

1. Le terme utilisateur doit être pris dans le sens général. Un utilisateur peut être, en fait, tout ce qui contrôle le système terminal : le protocole de contrôle de congestion, l'humain qui communique et qui peut modifier le protocole de contrôle de congestion s'il décide de ne pas collaborer, etc.

Le paradigme *TCP-friendly* a deux implications fondamentales sur la conception de nouveaux protocoles de contrôle de congestion. Premièrement, pour qu'un utilisateur puisse adapter le débit de sa connexion à l'équation 3.1, il faut qu'il connaisse le RTT et le taux de pertes de cette connexion. Cependant, il peut être difficile d'obtenir ces informations et dans certains cas ces informations sont mal définies, par exemple pour la transmission multipoint. De plus, l'obligation de diminuer le débit de la connexion en $\frac{1}{\sqrt{loss}}$ est mal adaptée aux applications qui tolèrent les pertes, comme les applications audios et vidéos. Une application multipoint orientée source devra adapter le débit de toute la session en fonction du récepteur qui perçoit le plus fort taux de pertes pour se conformer à l'équation 3.1, ce qui a pour conséquence de fortement pénaliser les autres membres de la session.

Deuxièmement, le paradigme *TCP-friendly* fait l'hypothèse que tous les utilisateurs collaborent, au sens de l'équation 3.1. Or, cette hypothèse ne peut plus être faite. De nouvelles applications, qui sont déployées dans l'internet, ne respectent pas cette équation. En effet, ces nouvelles applications – le plus souvent des applications de diffusion de contenu audio et vidéo – améliorent la satisfaction des utilisateurs en ne respectant pas le paradigme *TCP-friendly*. Cependant, la multiplication des sessions qui ne sont pas conformes au paradigme *TCP-friendly* risque de mettre en péril les sessions qui, elles, le respectent. À terme, c'est la stabilité de l'internet qui pourrait être compromise. Ce qui empêche, entre autres choses, un nouveau *congestion collapse* c'est que, d'une part, la majorité des utilisateurs est connectée à l'internet avec une liaison bas débit, typiquement un modem à 56 Kbit/s ; d'autre part, le cœur de l'internet possède des liaisons très haut débit, de l'ordre du gigabit jusqu'au térabit. Il est vrai que l'évolution de la technologie fibre optique permet d'obtenir des débits considérables, mais il est vrai que plus on a de bande passante, plus on trouve d'applications gourmandes en bande passante pour la saturer. En tout état de cause, on ne croit pas à une situation où l'internet offrira tellement de bande passante qu'il n'y aura plus de problèmes de congestion.

Le paradigme *TCP-friendly* est le résultat d'un processus entièrement empirique. Il est apparu après le protocole TCP pour permettre aux nouveaux protocoles de contrôle de congestion d'être compatibles avec ce dernier. On aurait pu choisir, pour la suite de cette thèse, deux orientations diamétralement opposées : soit adopter une approche consensuelle et aller dans le sens du paradigme *TCP-friendly* ; soit prendre du recul par rapport à celui-ci et chercher un nouveau paradigme plus efficace. C'est cette deuxième orientation, beaucoup plus ambitieuse, mais beaucoup plus risquée que l'on a choisie. Dans les paragraphes suivants, on va donner une nouvelle définition de la notion de congestion, donner les propriétés d'un protocole de contrôle de congestion idéal et définir un nouveau paradigme pour la conception de protocoles de contrôle de congestion presque idéaux.

3.2.2 Définition de la notion de congestion

Dire qu'un protocole de contrôle de congestion est fait pour éviter la congestion peut sembler trivial, c'est cependant fondamental puisque cela montre la relation qu'il existe entre le protocole de contrôle de congestion et le sens que l'on donne à la notion de congestion. Cette notion est reliée à la notion de débordement de files d'attente dans le paradigme *TCP-friendly*. Cependant, cette définition n'est pas satisfaisante puisqu'elle ne prend pas en compte la satisfaction de l'utilisateur. Il peut être objecté que le plus important est d'éviter des pertes pour garantir que le réseau soit bien utilisé ; on répondra qu'il ne faut pas oublier qu'un réseau n'est pas une fin en soi, mais qu'au contraire le but est de satisfaire les utilisateurs du réseau.

La notion de congestion doit donc être reliée à la notion de satisfaction des utilisateurs, mais doit également être reliée aux performances du réseau. En effet, si la congestion était uniquement fonction de la satisfaction des utilisateurs, on pourrait voir des phénomènes de jalousie créer de la congestion. Par exemple, il y aurait congestion si un utilisateur A apprenait qu'un utilisateur B avait un meilleur service que lui et que A n'était plus satisfait avec son propre service uniquement par jalousie. On ne veut pas que ce type de phénomène entre en jeux dans la notion de congestion. Notre définition de la notion de congestion est la suivante :

Définition 1 (Notion de congestion) *Un réseau est dit congestionné selon un utilisateur i si la satisfaction de i décroît à cause d'une modification de la performance (bande passante, délai, gigue, etc.) de sa connexion.*

C'est cette définition de la congestion que l'on va considérer dans toute la suite de notre thèse. Un protocole de contrôle de congestion devra donc chercher à maximiser la satisfaction des utilisateurs. On compare en annexe B.2.1 notre définition de la notion de congestion et celle donnée par Keshav [44]. On va définir, dans la suite, les propriétés d'un protocole de contrôle de congestion idéal au sens de la définition de congestion que l'on vient de donner.

3.2.3 Propriétés d'un protocole de contrôle de congestion idéal

On a besoin d'introduire deux termes pour la suite :

– un utilisateur « égoïste » est un utilisateur qui ne cherche qu'à augmenter sa propre satisfaction ;

– un utilisateur « collaborant » est un utilisateur qui tient compte des autres utilisateurs ;

en particulier, un utilisateur peut être égoïste et collaborant si sa satisfaction dépend des autres utilisateurs. On va se servir dans ce paragraphe de terminologies empruntées à la micro-

économique et à la théorie des jeux. Voici deux définitions.

Définition 2 (Équilibre de Nash) *Un réseau a atteint un équilibre de Nash si, chaque utilisateur agissant égoïstement, personne ne peut plus accroître sa propre satisfaction.*

Définition 3 (Optimum de Pareto) *Une allocation de bande passante A dans un réseau est un optimum de Pareto s'il n'existe pas une autre allocation B telle que :*

- *tous les utilisateurs aient, avec B, une satisfaction supérieure ou égale à celle obtenue avec A ;*
- *il existe au moins un utilisateur qui ait, avec B, une satisfaction strictement supérieure à celle obtenue avec A.*

On a identifié un ensemble de six propriétés que doit avoir un protocole de contrôle de congestion idéal. Même si les critères utilisés pour les propriétés sont pertinents par rapport à la définition de congestion que l'on a donnée, ils peuvent toujours être sujets à discussion. De plus, la terminologie « protocole de contrôle de congestion idéal » peut également être discutée, mais doit être ramenée au contexte du paradigme *TCP-friendly*. Un protocole de contrôle de congestion conçu avec ce paradigme aura des propriétés très inférieures à celles de notre protocole idéal. Les six propriétés d'un protocole de contrôle de congestion idéal sont les suivantes :

stabilité : comme tous les utilisateurs agissent égoïstement, on veut qu'ils convergent vers un équilibre de Nash. Une fois cet équilibre atteint, personne ne peut augmenter sa propre satisfaction ; par conséquent, cet équilibre est un équilibre pertinent pour le contrôle de congestion. Étant donné que plusieurs équilibres de Nash peuvent conduire à des oscillations entre ces équilibres, l'existence et l'unicité d'un équilibre de Nash sont les conditions de stabilité.

efficacité : lorsque l'allocation de la bande passante est un optimum de Pareto, personne ne peut augmenter sa satisfaction sans diminuer la satisfaction de quelqu'un d'autre. Cette notion d'optimum est donc pertinente pour l'efficacité d'un protocole de contrôle de congestion. De plus, la vitesse de convergence vers cet optimum est également importante. Une convergence rapide vers une allocation de la bande passante qui est un optimum de Pareto est la condition d'efficacité.

équité : il n'existe pas de consensus sur la notion d'équité. On a choisi comme notion d'équité l'équité max-min [5]. Si l'on considère des utilisateurs dont l'utilité est une fonction linéaire de la bande passante reçue, l'allocation de la bande passante qui est max-min équitable est également un optimum de Pareto. Par conséquent, la notion d'équité max-min

définit une borne supérieure pour l'allocation de la bande passante. Si tous les utilisateurs sont avides, ils auront juste la bande passante autorisée par l'allocation max-min. Par contre, si les utilisateurs collaborent, ils pourront atteindre d'autres types d'équité comme *proportional fairness* [43].

robustesse aux attaques : étant donné que l'on n'a aucune restriction sur les utilisateurs – utilisateurs égoïstes avec aucune restriction sur la fonction d'utilité –, on peut avoir des utilisateurs très agressifs. Il ne faut pas que de tels utilisateurs affectent les autres, c'est-à-dire qu'ils ne doivent pas significativement modifier la satisfaction des autres utilisateurs.

robustesse aux facteurs d'échelle : l'internet évolue très rapidement au niveau de la bande passante disponible mais aussi au niveau du nombre d'utilisateurs. Un protocole de contrôle de congestion doit fonctionner aussi bien sur des liens à 28.8 Kbit/s que sur des liens à 155 Mbit/s. Il doit également conserver toutes ses propriétés (stabilité, efficacité, etc.) quel que soit le nombre d'utilisateurs.

faisabilité : cette propriété contient toutes les contraintes techniques. On s'est restreint aux réseaux *best effort* de type Internet. Mais, l'internet connecte une grande variété de machines utilisant une grande variété de logiciels. Un protocole de contrôle de congestion doit fonctionner sur cette grande variété de machines et de logiciels. De plus, le protocole de contrôle de congestion doit rester suffisamment simple afin d'être programmé efficacement. Pour être accepté comme un standard international, un protocole de contrôle de congestion doit être intensivement testé, la simplicité du protocole facilitera cette phase.

Ces propriétés couvrent tous les aspects d'un protocole de contrôle de congestion, de l'aspect théorique de la stabilité à l'aspect pratique de la faisabilité. Cependant, la question qui se pose désormais est : comment concevoir un tel protocole ? On va maintenant répondre à cette question au paragraphe suivant.

3.2.4 Un nouveau paradigme

On veut définir un nouveau paradigme pour la conception de protocoles de contrôle de congestion idéaux et *end-to-end* pour les réseaux *best effort* ; il doit, par conséquent, respecter les fondements des réseaux *best effort* et, en particulier, l'argument *end-to-end* [78]. On veut également que ce paradigme nous permette de concevoir des protocoles de contrôle de congestion proches d'un protocole de contrôle de congestion idéal.

On a vu que le paradigme *TCP-friendly* était très loin de permettre de concevoir des protocoles de contrôle de congestion idéaux. Le problème vient de l'équation 3.1 qui doit garantir à la fois équité, efficacité et stabilité. Pour obtenir ces trois propriétés – qui ne sont pas idéales

dans le cas du paradigme *TCP-friendly* – avec un seul mécanisme aux systèmes terminaux, on doit faire des compromis sur les trois propriétés. Notre idée est de s'aider du support du réseau pour décentraliser la gestion de ces propriétés. Le support réseau peut aller d'un simple mécanisme de gestion de la mémoire tampon des files d'attente (*buffer management*) jusqu'aux réseaux actifs. On a choisi de considérer comme support du réseau un mécanisme d'ordonnancement de type GPS [65]. Cependant, l'ordonnancement GPS est basé sur un modèle fluide ; on a donc besoin d'une approximation discrète de ce modèle. Une bonne approximation du modèle fluide est la politique WF^2Q [3]. Une qualité fondamentale d'un support du réseau basé sur une politique d'ordonnancement GPS est qu'il s'agit d'un support d'utilité globale. En effet, il ne s'agit pas d'un mécanisme spécifique à un protocole de contrôle de congestion, mais bien d'un mécanisme qui améliore la performance globale du réseau (voir annexe B.3.1). Par conséquent, un support du réseau basé sur une politique d'ordonnancement de type GPS est compatible avec l'argument *end-to-end* [71]. Le support réseau que l'on va considérer dans la suite est basé sur la notion de discipline *Fair Scheduler* (FS).

Définition 4 (Fair Scheduler) *On définit une discipline Fair Scheduler (FS) comme étant une approximation discrète d'un ordonnancement fluide par flux de type GPS avec une politique de pertes de paquets à la file la plus longue (longest queue drop buffer management).*

Un paradigme est un ensemble de contraintes à appliquer lors de la conception de nouveaux protocoles de contrôle de congestion. Pour des raisons didactiques, on fait une distinction entre les contraintes en relation avec le réseau et les contraintes en relation avec les utilisateurs. Pour ne pas dérouter le lecteur habitué aux abréviations anglaises, et pour faciliter la lecture des annexes, on conserve les abréviations anglaises NP (*Network Part*) pour la partie réseau et ESP (*End System Part*) pour la partie utilisateur. Afin de concevoir un protocole de contrôle de congestion en fonction du paradigme FS, on doit considérer les contraintes suivantes :

– pour la partie réseau (NP) du paradigme, on a besoin d'un réseau *Fair Scheduler* (FS), c'est-à-dire un réseau où tous les routeurs utilisent une discipline FS ;

– pour la partie systèmes terminaux (ESP) du paradigme, des utilisateurs égoïstes et qui ne collaborent pas sont suffisants. On note que la partie ESP est une condition suffisante mais pas nécessaire ; en particulier, on peut avoir collaboration entre les utilisateurs si cela augmente leur satisfaction.

La contrainte sur les systèmes terminaux est très faible : cela laisse une grande latitude lors de la conception de nouveaux protocoles de contrôle de congestion. Or, on peut légitimement se demander si un protocole de contrôle de congestion conçu selon les contraintes du paradigme FS aura plus de bonnes propriétés, c'est-à-dire les propriétés d'un protocole de contrôle de congestion idéal, que s'il était conçu selon le paradigme *TCP-friendly*. Pour répondre à cette question,

on va voir quelles propriétés d'un protocole de contrôle de congestion idéal sont vérifiées avec le paradigme FS :

stabilité : avec les contraintes de la partie réseau et de la partie systèmes terminaux, l'existence et l'unicité d'un équilibre de Nash est garantie [79]. Par conséquent, un protocole de contrôle de congestion conçu avec le paradigme FS sera stable.

efficacité : avec les contraintes de la partie réseau et de la partie systèmes terminaux, même un algorithme d'optimisation simple convergera rapidement vers un équilibre de Nash. Cependant, cet équilibre de Nash ne sera pas un optimum de Pareto en général, il ne le sera que si tous les utilisateurs ont la même fonction d'utilité ou s'il y a collaboration entre tous les utilisateurs [79]. En résumé, un protocole de contrôle de congestion conçu avec le paradigme FS ne sera pas idéalement efficace dans tous les cas.

équité : la contrainte de la partie réseau garantit une équité max-min en moyenne. Par conséquent, un protocole de contrôle de congestion conçu avec le paradigme FS sera équitable [36].

robustesse aux attaques : la contrainte de la partie réseau garantie la robustesse d'un protocole de contrôle de congestion conçu avec le paradigme FS [20].

robustesse aux facteurs d'échelle : étant donné que la contrainte sur les systèmes terminaux est très faible, on a une grande flexibilité pour concevoir des protocoles de contrôle de congestion évolutifs.

faisabilité : un *Fair Scheduler* de type HPFQ [4] a été inclus dans des routeurs gigabits. Par conséquent, l'application de la contrainte sur le réseau est possible techniquement. De plus, même un algorithme simple donnera un protocole efficace. Un protocole simple sera plus facile à concevoir et à tester.

Le paradigme FS ne permet pas de concevoir des protocoles de contrôle de congestion avec une efficacité idéale dans tous les cas. Cependant, l'efficacité garantie par le paradigme FS est toujours très supérieure à celle garantie par le paradigme *TCP-friendly*. En effet, la contrainte sur le réseau garantie de pouvoir faire un compromis efficace entre bande passante, délai et perte [65].

D'autre part, étant donné que l'on n'a fait aucune hypothèse sur le mode de transmission utilisé, le paradigme FS s'applique aussi bien à la conception de protocoles de contrôle de congestion point à point qu'à la conception de protocoles de contrôle de congestion multipoints. La conception des protocoles de contrôle de congestion multipoints est grandement facilitée par le paradigme FS ; par exemple, on n'a plus besoin d'ajouter des mécanismes spécifiques pour

garantir l'équité du protocole, mécanismes qui, la plus part du temps, nuisent à l'efficacité du protocole. On va expliquer au paragraphe suivant comment appliquer le paradigme FS pour la conception d'un nouveau protocole de contrôle de congestion et en particulier par la conception d'un protocole de contrôle de congestion multipoint.

3.2.5 Conclusion

On vient de définir un nouveau paradigme, le paradigme FS, pour la conception de protocoles de contrôle de congestion *end-to-end*. On a montré que ce paradigme permettait de concevoir des protocoles de contrôle de congestion presque idéaux (au sens des propriétés que l'on a énoncé au § 3.2.3). Avec le paradigme FS, notre contribution majeure est que l'on a énoncé avec le même formalisme mathématique une définition de la notion de congestion, les propriétés requises pour un protocole de contrôle de congestion idéal et un nouveau paradigme pour la conception de protocoles de contrôle de congestion. De plus, ce formalisme nous a permis de prouver que le paradigme FS permettait de concevoir des protocoles de contrôle de congestion presque idéaux. Ainsi, le paradigme FS est le premier paradigme introduit et prouvé formellement. En annexe B.3.2, on donne quelques remarques sur le déploiement de la contrainte réseau et en annexe B.4, on compare les mérites respectifs du paradigme FS et du paradigme *TCP-friendly*.

Cependant, comment interpréter le paradigme FS pour concevoir un nouveau protocole de contrôle de congestion ? La contrainte du paradigme FS sur les systèmes terminaux est d'avoir des utilisateurs égoïstes et qui ne collaborent pas. De plus, cette condition est suffisante. Lors de la conception d'un nouveau protocole de contrôle de congestion avec le paradigme FS, on doit uniquement s'occuper des besoins de l'utilisateur et non des propriétés que l'on souhaiterait pour le protocole. Ces dernières seront automatiquement garanties par le paradigme FS. En fait, on n'a pas besoin de prendre en compte les différentes propriétés d'un protocole de contrôle de congestion comme l'équité ; on doit juste trouver un mécanisme qui satisfasse l'utilisateur. Le paradigme FS ne donne pas ce mécanisme mais simplifie considérablement la conception du protocole. À la différence du paradigme *TCP-friendly*, il permet de créer un schisme entre les propriétés requises pour un protocole de contrôle de congestion et les besoins de l'utilisateur, les propriétés étant garanties par le support du réseau. Ce schisme donne une grande latitude lors de la conception d'un nouveau protocole de contrôle de congestion.

Pour avoir une validation pragmatique du paradigme FS, on va concevoir un nouveau protocole de contrôle de congestion multipoint à couches cumulatives en se basant sur l'expérience acquise avec RLM et RLC (voir § 3.1). On a vu que le principal problème avec RLM et RLC venait de leur mécanisme de découverte de la bande passante. En effet, ces mécanismes sont basés sur des signaux de congestion, c'est-à-dire que leur seule information sur la bande passante disponible est au travers des signaux de congestion. Un signal de congestion est en général

une perte ou un signal ECN (*Early Congestion Notification*) [29]. Cependant, quel que soit le moyen pour signaler la congestion, un mécanisme de découverte de la bande passante basé sur un signal de congestion aura toujours les mêmes faiblesses :

- la file d'attente du goulot d'étranglement doit déborder pour que le signal de congestion soit généré ;
- le signal de congestion est reçu par le récepteur longtemps après que la congestion a commencé au goulot d'étranglement ;
- un signal de congestion ne permet pas d'avoir des informations sur la bande passante disponible.

La technique de l'envoi de paquets par paire introduite par Keshav [44] permet d'obtenir une notification explicite de la bande passante disponible. Il s'agit donc d'un mécanisme simple qui n'a aucun des inconvénients des signaux de congestion. D'après le paradigme FS, on devrait pouvoir concevoir facilement, à partir de la technique de l'envoi de paquets par paire, un nouveau protocole de contrôle de congestion multipoint presque idéal. On va décrire au paragraphe suivant un nouveau protocole de contrôle de congestion multipoint à couches cumulatives basé sur la technique de l'envoi de paquets par paire et on va montrer qu'il a des propriétés très proches de celles d'un protocole de contrôle de congestion idéal, validant ainsi le paradigme FS.

3.3 PLM : une validation du paradigme FS

3.3.1 Introduction

La distribution de contenu multimédia à un large groupe d'utilisateurs hétérogènes est un des problèmes les plus ardus pour le contrôle de congestion. Des protocoles comme RLM et RLC ont été proposés, mais ils souffrent de nombreux comportements pathologiques (voir § 3.1). Cependant, ils ont permis de débroussailler le terrain en proposant des choix stratégiques :

- la transmission multipoint est parfaitement adaptée à la diffusion à un large groupe ;
- l'envoi du contenu en couches cumulatives associé à un protocole orienté récepteur offre une solution efficace pour les groupes hétérogènes (le § 3.1.1 donne une introduction sur les notions de couches cumulatives et de protocoles orientés récepteur, et on pourra également consulter l'annexe C.3.1).

Pour valider le paradigme FS, on va concevoir un nouveau protocole de contrôle de congestion pour la distribution de contenu multimédia à un large groupe d'utilisateurs. Ce protocole

sera un protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur. Contrairement à RLM et RLC, on ne va pas utiliser un mécanisme de découverte de la bande passante basé sur des signaux de congestion, mais un mécanisme de découverte de la bande passante basé sur des notifications explicites de la bande passante disponible qui utilisent la technique de l'envoi de paquets par paire. Cette technique est rendue possible grâce à la contrainte réseau du paradigme FS.

3.3.2 La technique de l'envoi de paquets par paire

La technique de l'envoi de paquets par paire – dans la suite on parlera plus simplement de la technique PP par référence à Paquet par Paire – a été introduite par Keshav [44] pour permettre à une source de découvrir la bande passante disponible. Deux paquets envoyés par paire – on parlera plus simplement d'un PP – sont deux paquets envoyés aussi rapidement que possible. On dit, de manière imagée, que les deux paquets sont envoyés dos à dos (*back-to-back*). Lorsque l'on envoie un PP dans un réseau *Fair Scheduler*, les paquets du PP seront espacés au récepteur en fonction de la bande passante disponible sur le chemin entre la source et le récepteur. En envoyant fréquemment des PP, on peut suivre l'évolution de la bande passante.

Keshav utilisa la technique PP dans une version orientée source : la source envoie deux paquets par paire, le récepteur acquitte les deux paquets et la source mesure l'espacement des acquittements. Cependant, s'il y a un goulot d'étranglement sur le chemin entre le récepteur et la source – goulot d'étranglement pour les acquittements –, les acquittements seront espacés en fonction de la bande passante disponible sur le chemin entre le récepteur et la source et, par conséquent, la source mesurera la bande passante disponible sur le mauvais chemin, c'est-à-dire celui des acquittements et non des paquets de données. D'autre part, Keshav utilisa la technique PP pour un ajustement de la bande passante avec une granularité fine. Il eut donc besoin d'estimateurs complexes pour filtrer le bruit inhérent à la technique PP. Ce bruit – des erreurs dans les estimations – peut avoir de nombreuses sources : un goulot d'étranglement sur le chemin des acquittements, la politique d'ordonnancement qui est nécessairement une approximation d'un ordonnancement de type GPS, le partage de charge (*load balancing*), etc. On étudie plus en détail l'impact du bruit sur la technique PP en annexe C.3.2.

On va utiliser la technique PP d'une manière différente, moins sensible au bruit. Premièrement, on va considérer une version orientée récepteur de la technique PP qui supprime tous les problèmes dus au goulot d'étranglement sur le chemin des acquittements et réduit, par conséquent, considérablement le bruit inhérent à la technique PP [67]. Deuxièmement, on va utiliser cette technique pour un ajustement de la bande passante avec une large granularité. En effet, on va l'utiliser pour choisir à quelles couches s'abonner ou se désabonner. Étant donné que les couches ont une granularité en bande passante large, de petites erreurs dans l'estimation de la bande passante ne conduiront pas à un changement de couche et, par conséquent, n'auront pas

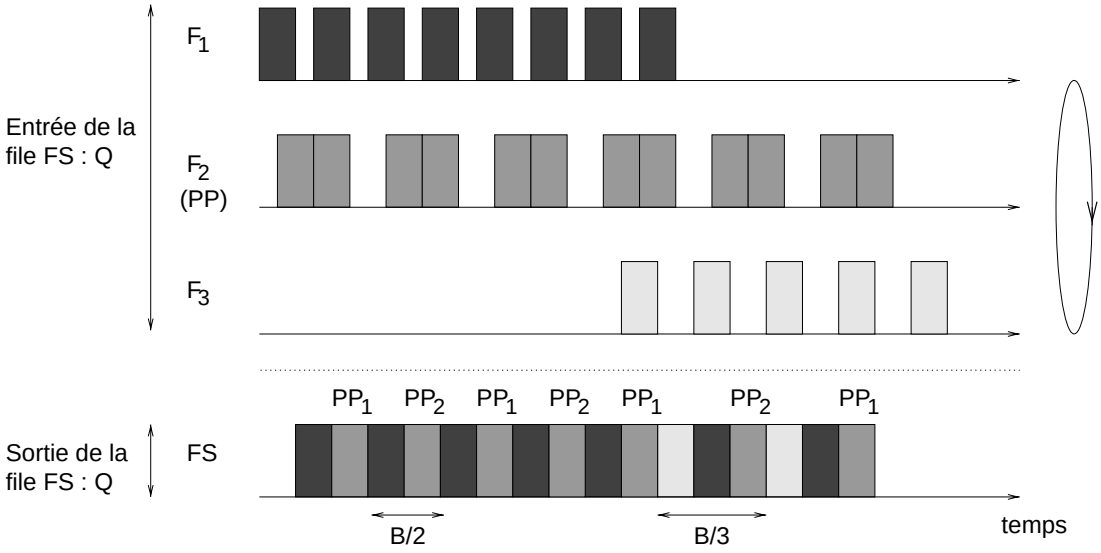


FIG. 3.1 – *Illustration de la technique PP dans un exemple simple.*

d'impact sur notre protocole utilisant la technique PP.

Mais la caractéristique la plus remarquable de la technique PP orientée récepteur – cette caractéristique existe toujours, mais à moindre titre, pour une version orientée source – est qu'un récepteur va détecter la congestion avant que la file d'attente du goulot d'étranglement ne se remplisse et bien avant que la file ne déborde ; c'est à dire, bien avant qu'il n'y ait des pertes. Les PP sont des notifications explicites de la bande passante disponible ; par conséquent, un PP signal de congestion sera un PP qui indiquera une bande passante disponible inférieure au débit actuel de la source pour le récepteur qui reçoit ce PP. Si l'on suppose qu'un seul PP est suffisant pour avoir une estimation de la bande passante disponible (filtre trivial), alors le premier PP qui quittera la file d'attente du goulot d'étranglement après que la congestion a commencé sera un signal de congestion. Le délai entre le début de la congestion et le moment où le récepteur est informé de cette congestion correspond approximativement au délai de transmission d'un paquet entre le goulot d'étranglement et le récepteur ! Pour montrer les performances de la technique PP, on va donner un exemple illustré à la figure 3.1. Ce dessin représente l'entrée et la sortie de la file d'attente – file FS – du goulot d'étranglement. En entrée de la file, il y a trois flux : F_1, F_2 et F_3. Le flux F_2 représente le flux utilisant la technique PP pour la découverte de la bande passante. Avant que le flux F_3 n'ait des paquets à l'entrée de la file FS, les PP sont espacés en fonction la bande passante disponible $\frac{B}{2}$. Un cycle du *Fair Scheduler* (*Fair Scheduler round*) après que le premier paquet de F_3 est entré dans la file – soit le temps de servir trois paquets –, un PP quitte la file, espacé de la nouvelle bande passante disponible $\frac{B}{3}$. De plus, le PP signal de congestion était déjà dans la file avant que le premier paquet du flux F_3 n'y entre ; ce PP est entré dans la file alors que la bande passante disponible était $\frac{B}{2}$, mais est sorti de la file, espacé de la bande passante disponible au moment de son service (au niveau du *Fair Scheduler*) soit

$\frac{B}{3}$. D'autre part, le PP a quitté la file alors qu'il n'y avait qu'un seul paquet du flux F_3 dans la file, c'est-à-dire bien avant que la file ne déborde.

En résumé, la technique PP permet de découvrir la bande passante et de réagir à la congestion avant que la file ne déborde, c'est-à-dire sans aucune perte induite.

3.3.3 Le protocole PLM

Le protocole *Packet Pair Layered Multicast* ou PLM est un protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur basé sur la technique de l'envoi de paquets par paire ; on fait donc l'hypothèse que l'on a un réseau qui autorise la transmission multipoint et on suppose que les données à envoyer peuvent être découpées en couches cumulatives. PLM est un protocole conçu d'après le paradigme FS ; on fait donc l'hypothèse que l'on a un réseau *Fair Scheduler*.

La source PLM est simple : elle envoie chaque couche dans un groupe multipoint différent et elle envoie les paquets de chaque couche par paire. En fait, la source n'envoie que des PP. Chaque PP reçu par un récepteur fournit une estimation de la bande passante disponible sur le chemin entre la source et ce récepteur. On obtient l'estimation de la bande passante en divisant le temps d'interarrivé des paquets de la paire par la taille d'un paquet. Le protocole au niveau du récepteur est le suivant :

– chaque fois qu'un récepteur reçoit un PP, c'est-à-dire les deux paquets de la même paire, il regarde si la bande passante disponible estimée par ce PP est plus petite que la bande passante demandée par le récepteur en fonction du nombre de couches auxquelles le récepteur est abonné. Si c'est le cas, le récepteur va immédiatement se désabonner du nombre de couches nécessaire pour que la bande passante qu'il demande soit inférieure à la bande passante disponible estimée par le PP.

– le récepteur n'ajoute des couches qu'en fonction de l'estimation minimum reçue durant une période de durée C (*Check value*) si toutes les estimations fournies par les PP sont supérieures à la bande passante demandée. Le récepteur ajoutera autant de couches qu'il faudra pour que la bande passante qu'il demande soit la plus haute possible sans dépasser la valeur donnée par l'estimation minimum reçue durant la période de durée C.

En résumé, un récepteur se désabonne à des couches en se basant sur un seul PP, mais s'abonne à des couches en se basant sur la valeur minimum de tous les PP reçus durant une période de durée C. De plus, un récepteur peut s'abonner ou se désabonner à plusieurs couches à la fois en fonction de la bande passante disponible ; tous les détails sur le protocole PLM sont donnés en annexe C.3.3. On note que ce paramètre C est le seul paramètre de PLM. Le plus marquant avec ce protocole est son extrême simplicité qui semble corroborer la validité du

paradigme FS. Cependant, pour valider le paradigme, il faut encore vérifier que PLM ait des propriétés proches de celles d'un protocole de contrôle de congestion idéal.

3.3.4 Évaluation du protocole PLM

On a évalué le comportement de PLM dans un grand nombre de configurations. Tous les détails sont donnés en annexes C.4 et C.5. On trace ici les grandes lignes de notre évaluation de PLM.

On a commencé par évaluer PLM dans des scénarios simples, qui ne sont pas destinés à être réalistes mais à permettre de comprendre le comportement de PLM dans des cas simples. On a commencé par considérer une seule session PLM sur une topologie hétérogène en bande passante et en délai. Dans ce scénario, tous les récepteurs de cette session PLM convergent après une période de durée C (*Check value*) vers la bande passante optimale sans aucune perte induite. Cette convergence est indépendante de la granularité des couches, du nombre de couches et des autres récepteurs dans la même session quel que soit le nombre de récepteurs. De plus, les récepteurs restent à cette bande passante optimale durant toute la simulation sans aucune perte induite.

On voit donc que dans des scénarios statiques, c'est-à-dire sans variation de la bande passante disponible, PLM se comporte idéalement. Dans une autre série de simulations, on a considéré trois sessions PLM partageant le goulot d'étranglement avec trois flux CBR (*Constant Bit Rate*). On utilise ces flux pour simuler une période de forte congestion. On constate que : les sessions PLM partagent équitablement la bande passante ; les récepteurs PLM convergent vers la bande passante disponible optimale ; les récepteurs PLM s'adaptent immédiatement à la période de congestion créée par les flux CBR et reprennent la bande passante disponible, après une période de durée C, lorsque les flux CBR s'arrêtent. Mais le plus remarquable dans ce scénario, c'est qu'aucun récepteur PLM n'observe de pertes durant toute la simulation même pendant la période de forte congestion. Dans la série suivante de simulations, on a étudié le comportement d'une session PLM partageant le goulot d'étranglement avec deux flux TCP. On observe exactement les mêmes résultats que précédemment : la session PLM s'adapte très rapidement aux variations de bande passante disponible sans aucune perte induite.

On voit donc que pour ces scénarios simples PLM se comporte idéalement, il s'adapte très vite aux variations de la bande passante disponible sans aucune perte induite. On a étudié dans une autre série de simulations le comportement de PLM lorsque l'on augmente le nombre de sessions PLM partageant le même goulot d'étranglement. On observe un bon comportement de PLM, mais avec, dans certains cas, un faible taux de pertes induites. On note, cependant, que ce scénario avec uniquement un grand nombre de sessions PLM est pathologique : il n'y a que des flux avec une adaptation à la bande passante à large granularité – protocole à couches. Or, PLM n'a pas été conçu pour fonctionner dans un tel environnement, mais dans un réseau *best*

effort avec d'autres flux ayant une adaptation à la bande passante à fine granularité comme TCP. Dans la série suivante de simulations, on a considéré le même scénario mais en mélangeant aux sessions PLM des flux TCP. Dans ce cas, PLM retrouve son comportement idéal sans aucune perte induite.

Pour finir les séries de simulations pour des scénarios simples, on a évalué le comportement de PLM lorsqu'il partage le goulot d'étranglement avec des flux constitués de paquets de taille différentes des tailles de paquets PLM. Dans certains cas, cela peut affecter la performance de PLM ; cependant, une grande taille de paquets pour les flux PLM et le multiplexage des flux permettent de réduire considérablement, et même de supprimer, les problèmes liés aux tailles de paquets.

En résumé de ces scénarios simples, PLM a un comportement idéal : il converge très rapidement vers la bande passante disponible, il suit les variations de celle-ci sans aucune perte induite, même lorsqu'il y a de sévères périodes de congestion. Cependant, le trafic exogène dans un réseau réel est loin d'être aussi simplifié que dans les scénarios précédents. Ce trafic est en fait autosimilaire et multifractal [27]. On va donc, dans la suite, tester le comportement d'une session PLM dans un tel environnement. Tous les détails du scénario qui permet d'obtenir un trafic autosimilaire et multifractal sont donnés dans l'annexe C.5.1. Le comportement de PLM dans un environnement réaliste aussi complexe est excellent : la session suit les évolutions de la bande passante disponible sans aucune perte induite durant les 4500 secondes de simulation. Étant donné que PLM est un protocole à couches cumulatives, son seul moyen de s'adapter à la bande passante disponible est de s'abonner ou de se désabonner à des couches. Or, comme le trafic exogène varie à de nombreuses échelles de temps, la session PLM va devoir s'adapter à ces variations pour exploiter la bande passante disponible. Cela va se traduire par des oscillations dans les abonnements aux couches. Cependant, comme on vient de l'expliquer, ces oscillations ne sont pas le résultat d'une instabilité de PLM, mais bien le résultat de sa grande efficacité. L'oscillation des abonnements aux couches peut avoir deux conséquences néfastes :

– dans le cas ou PLM est utilisé pour la transmission de contenu audio ou vidéo, les oscillations se traduisent par de fréquents changements de qualité ce qui peut être irritant pour un utilisateur. Cependant, le but d'un protocole de contrôle de congestion est d'offrir la plus grande satisfaction possible aux utilisateurs ; par exemple, un haut débit pour les applications multimédias. On affirme donc que c'est le rôle de l'application de lisser ces changements de qualité [58] et non le rôle du protocole de contrôle de congestion de diminuer son efficacité. Cependant, si cela est nécessaire, on pourra très facilement diminuer le nombre d'oscillations en augmentant le paramètre C (*check value*) sans pour autant diminuer radicalement l'efficacité de la transmission. Par exemple, pour une de nos simulations, on a obtenu, sur 4500 secondes, un débit moyen de 733 Kbit/s et 2090 changements de couches pour $C = 1$ seconde et un débit moyen de 561 Kbit/s et 417

changements de couches pour $C = 5$ secondes.

– les changements de couches génèrent du trafic au niveau du protocole de routage. Ce trafic de contrôle peut avoir un coût non négligeable. Cependant, si l'on reprend l'exemple précédent, pour un débit de 733 Kbit/s, on a environ un message de contrôle toutes les deux secondes, ce qui est modeste. De plus, une simple augmentation du paramètre C permet de faire chuter le nombre de messages de contrôle à un toutes les dix secondes, ce qui est négligeable.

L'oscillation des abonnements aux couches est le résultat de la grande efficacité de PLM. Mais, si cela est nécessaire, elle peut facilement être réduite.

En résumé, on a testé PLM dans une grande variété de configurations et on a trouvé que PLM était capable de suivre les évolutions de la bande passante disponible sans aucune perte induite, même dans un environnement autosimilaire et multifractal.

3.3.5 Conclusion

On a appliqué le paradigme FS pour la conception d'un nouveau protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur : PLM, basé sur la technique de l'envoi de paquets par paire. La conception de ce protocole devait permettre de valider le paradigme FS. Et, en effet, PLM a bien été une validation du paradigme FS. On rappelle que l'idée directrice du paradigme FS est qu'il suffit de trouver un mécanisme qui satisfasse les utilisateurs, le paradigme FS garantit alors toutes les propriétés d'un protocole de contrôle de congestion presque idéal. On a effectivement trouvé un mécanisme qui satisfasse les utilisateurs, la technique de l'envoi de paquets par paire, et on a conçu un protocole de contrôle de congestion autour de cette technique sans se préoccuper des propriétés spécifiques à un protocole de contrôle de congestion. On va maintenant vérifier si PLM a bien les propriétés d'un protocole de contrôle de congestion presque idéal comme le garantit le paradigme FS. PLM a les propriétés suivantes :

stabilité : les récepteurs PLM convergent rapidement (après une période de C secondes) et les oscillations des abonnements aux couches ne sont pas dues à une instabilité du protocole mais à une grande efficacité de ce dernier.

efficacité : les récepteurs PLM découvrent très rapidement la bande passante disponible et sont capables de suivre les évolutions de la bande passante disponible de très près. PLM surpasse RLM et RLC.

équité : une session PLM est équitable avec les autres sessions PLM et avec les flux TCP.

robustesse aux attaques : PLM est robuste aux sessions qui utilisent des autres protocoles de contrôle de congestion.

robustesse aux facteurs d'échelle : PLM est robuste aux facteurs d'échelle grâce au principe des couches cumulatives avec une gestion orientée récepteur.

faisabilité : PLM est un protocole simple qui peut facilement être évalué. De plus, PLM a été introduit dans la distribution du simulateur ns [62] ; il peut donc facilement être étudié.

En conclusion, PLM vérifie bien les propriétés définies dans le § 3.2.3, il valide donc le paradigme FS.

3.4 Une nouvelle politique d'allocation de la bande passante

3.4.1 Introduction

Lorsque cohabitent des flux multipoints et point à point se pose la question de l'allocation de la bande passante entre ces flux. En effet, comment allouer la bande passante d'un lien entre un flux point à point ne servant qu'un seul récepteur et un flux multipoint servant un million de récepteurs ? Cette question, loin d'être triviale, peut avoir de nombreuses réponses selon le but que l'on recherche et suivant que l'on adopte le point de vue du réseau ou le point de vue des utilisateurs.

Si l'on se place du point de vue du réseau, ou plus précisément du point de vue du fournisseur d'accès, l'utilisation de la transmission multipoint permet d'économiser de la bande passante et de déployer de nouveaux services comme la diffusion à un grand nombre d'utilisateurs de contenu audio et vidéo. Cependant, à cause de son coût élevé, la transmission multipoint n'est rentable, pour un fournisseur d'accès, que si l'on considère de grands groupes. Pour de petits groupes, la transmission point à point sera plus rentable [21]. Dans ce contexte, lorsque l'on parle de la rentabilité de la transmission multipoint, c'est toujours par rapport à la transmission point à point ; la transmission multipoint est rentable si l'économie réalisée avec le gain de bande passante – par rapport à la bande passante utilisée pour le même service en point à point – compense le coût du déploiement de la technologie nécessaire pour avoir un service multipoint. Dans cette notion de rentabilité, on ne tient pas compte du bénéfice apporté par l'ajout d'un nouveau service qui ne serait pas possible avec une transmission point à point. Par exemple, un fournisseur d'accès qui offre un service de diffusion audio et vidéo attirera de nouveaux utilisateurs ; cependant, ce type de bénéfice est difficile à évaluer et n'est pas le propos de cette thèse.

Si l'on se place du point de vue des utilisateurs, le mode de transmission – multipoint ou point à point – importe peu. Un utilisateur veut simplement augmenter sa satisfaction. Tous les

bénéfices de la transmission multipoint sont transparents pour l'utilisateur, sauf dans le cas où la transmission multipoint offre un service qui serait impossible à fournir avec la transmission point à point. Le fait que la transmission multipoint permette d'économiser les ressources du réseau peut conduire à diminuer le prix du service, ce qui aura un impact sur la satisfaction des utilisateurs. Cependant, il s'agit de considérations économiques et commerciales qui sont également hors du propos de cette thèse.

On vient de voir ce que la transmission multipoint pouvait apporter à un fournisseur d'accès et à un utilisateur. On n'a cependant pas expliqué comment allouer la bande passante entre flux multipoints et flux point à point. Étant donné que la transmission multipoint peut être rentable pour un fournisseur d'accès mais que ce mode de transmission est transparent pour un utilisateur, il faut inciter les utilisateurs à se servir de la transmission multipoint. L'incitation peut-être purement financière : un service utilisant la transmission multipoint sera moins cher qu'un service utilisant la transmission point à point. Cependant, il nous est apparu intéressant de mettre de côté cet aspect purement commercial pour se concentrer sur une incitation à utiliser la transmission multipoint basée sur l'allocation de la bande passante entre flux multipoints et flux point à point. Notre principale motivation est de redonner aux flux multipoints une partie de la bande passante qu'ils économisent ; bien que cela nous semble raisonnable, cette motivation peut être indéfiniment débattue. On va montrer qu'en redonnant aux flux multipoints une partie de la bande passante qu'ils économisent, on peut largement augmenter la satisfaction des utilisateurs de la transmission multipoint sans pour autant diminuer de manière significative la satisfaction des utilisateurs de la transmission point à point. C'est, à notre avis, un argument convaincant en faveur d'une politique d'allocation de la bande passante qui prenne en compte le nombre de récepteurs.

Le paradigme FS permet de concevoir des protocoles de contrôle de congestion multipoints et point à point. Par conséquent, la question de l'allocation de la bande passante entre flux multipoints et flux point à point est pertinente dans le contexte du paradigme FS. D'autre part, la contrainte principale de ce paradigme est d'avoir un réseau *Fair Scheduler* ; or un *Fair Scheduler* est un mécanisme d'ordonnancement pondéré, c'est-à-dire un mécanisme où l'on peut gérer l'allocation de la bande passante entre les flux en fonction de poids donnés à chaque flux. Le paradigme FS permet donc d'appliquer facilement de nouvelles politiques d'allocation de la bande passante.

On va, dans la suite, étudier trois politiques qui allouent localement sur chaque lien la bande passante entre les flux multipoints et les flux point à point. Après avoir introduit dans le paragraphe suivant les trois politiques d'allocation de la bande passante et les critères utilisés pour les comparer, on va, dans le § 3.4.3, donner les principaux résultats sur l'évaluation des trois politiques.

3.4.2 Définition des politiques d'allocation de la bande passante

On va considérer trois politiques pour allouer la bande passante de chaque lien l aux flux passant par ce lien. L'allocation de la bande passante va se faire en fonction du nombre de récepteurs en aval du lien l. On définit le nombre de récepteurs pour un flux en aval d'un lien l comme étant le nombre de récepteurs qu'il y a après ce lien, dans le sens source/récepteurs, pour le flux considéré. On va considérer les trois politiques d'allocation de la bande passante suivantes (les définitions mathématiques sont données en annexe D.2.2) :

indépendant des récepteurs (RI) : on alloue de manière égale la bande passante du lien l entre flux multipoints et flux point à point traversant ce lien l, indépendamment du nombre de récepteurs en aval de ce lien. Cette allocation ne représente aucun changement avec l'allocation actuelle. Cette politique va donc nous servir de référence.

dépendant linéairement du nombre de récepteurs (LinRD) : on alloue la bande passante du lien l entre les flux avec une fonction qui dépend linéairement du nombre de récepteurs pour chaque flux en aval de ce lien. Cette allocation correspond à la bande passante qui serait donnée aux flux point à point nécessaires pour servir les mêmes utilisateurs – c'est-à-dire une connexion point à point différente entre la source et chaque récepteur –, s'il n'y avait pas de service multipoint, .

dépendant de manière logarithmique du nombre de récepteurs (LogRD) : on alloue la bande passante du lien l entre les flux avec une fonction qui dépend de manière logarithmique du nombre de récepteurs pour chaque flux en aval de ce lien. Cette allocation correspond au gain global d'un flux multipoint qui est logarithmique avec le nombre de récepteurs [61, 68]. En annexe D.3.1, on définit la notion de gain multipoint et en annexe D.3.2 on discute l'impact global d'une politique d'allocation locale de la bande passante.

On a conservé les notations anglaises pour ne pas dérouter le lecteur habitué à ces notations et pour faciliter la lecture des annexes. On utilise RI pour *Receiver Independent*, LinRD pour *Linear Receiver Dependent* et LogRD pour *Logarithmic Receiver Dependent*.

Ces trois politiques sont des représentants de classes de politiques d'allocation de la bande passante. On ne prétend pas qu'elles sont les meilleures représentants des classes ni que les classes sont optimales, on dit simplement que ces trois représentants permettent de couvrir un large spectre de politiques d'allocation de la bande passante et surtout de comprendre comment introduire le nombre de récepteurs dans l'allocation de la bande passante.

Pour évaluer et comparer ces trois politiques d'allocation de la bande passante, on a besoin de critères de comparaison. On cherche à augmenter la satisfaction des utilisateurs sans pour autant diminuer significativement l'équité. On définit le critère de satisfaction d'un utilisateur

comme étant la bande passante qu'il obtient. Même s'il existe d'autres critères pour évaluer la satisfaction des utilisateurs comme le délai ou la gigue, la bande passante est un critère pertinent pour un grand nombre d'applications. On définit le critère d'équité entre des utilisateurs comme l'écart type de la bande passante vue par ces mêmes utilisateurs. Étant donné qu'il s'agit d'une notion globale qui peut cacher quelques utilisateurs ayant une très faible satisfaction, on va considérer, en plus de notre critère d'équité, le cas du pire utilisateur, c'est-à-dire le récepteur qui voit la bande passante la plus faible. L'annexe D.2.3 présente une discussion détaillée de ces critères de comparaison.

3.4.3 Évaluation des politiques

On a dans un premier temps évalué les trois politiques avec deux modèles analytiques simples. Le premier modèle est une topologie en étoile : on considère une session multipoint partageant un même goulot d'étranglement avec plusieurs sessions point à point. Le deuxième modèle est une topologie en chaîne : on considère une session multipoint partageant plusieurs liens avec des sessions point à point (une session point à point par lien). La description précise des scénarios se trouve en annexes D.3.3.1 et D.3.3.2. Notre choix des modèles analytiques considérés a été guidé par le fait qu'un réseau complexe est une composition de topologies en étoile et en chaîne. La grande concordance entre les résultats obtenus avec les modèles analytiques et les résultats obtenus avec les simulations sur une large topologie montre que nos modèles analytiques, bien que simples, permettent d'avoir une bonne appréhension de la réalité.

L'analyse des résultats obtenus avec nos modèles analytiques nous a permis d'arriver aux conclusions suivantes (une discussion détaillée se trouve en annexes D.3.3.1 et D.3.3.2) : les deux politiques $LinRD$ et $LogRD$ offrent une plus grande satisfaction aux utilisateurs que la politique RI mais offre une moins bonne équité ; de plus, la politique $LinRD$ est celle qui donne la plus grande satisfaction mais la plus mauvaise équité ; la politique $LogRD$, quant à elle, donne une satisfaction moindre que la politique $LinRD$ mais une meilleure équité. On en a conclu que la politique $LogRD$ était le meilleur compromis entre satisfaction et équité.

Pour approfondir les résultats obtenus avec nos modèles analytiques, on a fait des simulations sur une large topologie hiérarchique *RT* (*Random Topology*), qui représente les trois niveaux d'interconnexion que l'on peut trouver dans un réseau : les WAN (*Wide Area Network*), les MAN (*Metropolitan Area Network*) et les LAN (*Local Area Network*). *RT* interconnecte 180 LAN. Ce type de topologies hiérarchiques est considéré comme un bon modèle de l'internet [9, 23, 90]. On a cherché dans ces simulations à étudier l'introduction d'un service multipoint dans un environnement point à point. On a commencé par dimensionner l'environnement point à point qui consiste en des paires source/récepteur positionnées aléatoirement sur les LAN de la topologie *RT*. On a trouvé que 2000 sessions point à point permettaient d'avoir un environne-

ment point à point qui soit peu sensible à l'emplacement aléatoire des paires source/récepteur (voir annexe D.4.1).

On a réalisé deux séries de simulations : la première considère une seule session multipoint, dont on augmente la taille (de 1 à 6000 récepteurs), introduite dans l'environnement point à point ; la deuxième considère plusieurs sessions multipoints de taille fixe, dont on augmente le nombre, introduites dans l'environnement point à point. On parlera de satisfaction globale lorsque l'on considérera la moyenne de la satisfaction de tous les utilisateurs ; on parlera d'équité globale lorsque l'on calculera l'écart type de la satisfaction des utilisateurs sur tous les utilisateurs. De même, on parlera de satisfaction multipoint (resp. point à point) lorsque l'on considérera la moyenne de la satisfaction sur uniquement les utilisateurs multipoints (resp. point à point) ; on parlera d'équité multipoint (resp. point à point) lorsque l'on calculera l'écart type de la satisfaction des utilisateurs multipoints (resp. point à point).

On commence par résumer la première série de simulations. Les politiques $LinRD$ et $LogRD$ offrent une plus grande satisfaction globale que la politique RI, mais offrent une moins bonne équité globale. Par contre, lorsque l'on fait une distinction entre utilisateurs multipoints et utilisateurs point à point, on trouve que la politique $LinRD$ est celle qui offre le plus de satisfaction pour les utilisateurs multipoints, mais c'est la seule qui diminue fortement la satisfaction des utilisateurs point à point pour de grandes tailles de groupes. La politique $LogRD$, quant à elle, augmente la satisfaction des utilisateurs multipoints, mais ne diminue pas la satisfaction des utilisateurs point à point – par rapport à la politique de référence RI –, même pour les grandes tailles de groupes. L'équité des trois politiques est la même pour les utilisateurs point à point, puisque les politiques ne font de distinction qu'en fonction du nombre de récepteurs. L'équité des politiques RI et $LogRD$ est proche pour les utilisateurs multipoints, alors que l'équité de la politique $LinRD$ est plus mauvaise. Si l'on regarde la cas du pire récepteur, on constate que la politique $LinRD$ diminue très fortement la bande passante pour ce récepteur lorsque l'on augmente la taille du groupe multipoint, alors que la bande passante vue par le pire récepteur avec la politique $LogRD$ est très proche de celle vue avec la politique RI, même pour une grande taille de groupe. En résumé, la politique $LogRD$ est la seule à augmenter la satisfaction des utilisateurs multipoints sans pour autant diminuer significativement la satisfaction des utilisateurs point à point et en gardant une équité proche de celle de la politique RI.

La deuxième série de simulations va confirmer ces résultats. On a fait des simulations avec soit des tailles de groupes de 20 récepteurs, soit des tailles de groupes de 100 récepteurs. Dans les deux cas on arrive aux mêmes conclusions. La satisfaction et l'équité globale sont proches pour les trois politiques. Par contre, les deux politiques $LinRD$ et $LogRD$ donnent une satisfaction plus élevée pour les utilisateurs multipoints. La politique $LinRD$ diminue la satisfaction des utilisateurs point à point par rapport à la politique RI alors que la politique $LogRD$ conduit à une satisfaction proche de celle obtenue avec la politique RI. De plus, la bande passante vue

par le pire récepteur est très proche pour les politiques RI et $LogRD$, alors qu'elle est beaucoup plus basse pour la politique $LinRD$ que pour la politique RI. Pour cette série de simulations, la politique $LogRD$ est la seule à augmenter la satisfaction des utilisateurs multipoints sans diminuer significativement la satisfaction des utilisateurs point à point.

En résumé, la politique $LogRD$ est le meilleur compromis entre satisfaction et équité. De plus, on a montré que cette politique permettait d'augmenter largement la satisfaction des utilisateurs multipoints sans pour autant diminuer significativement la satisfaction des utilisateurs point à point et tout en gardant une équité proche de celle de la politique RI.

En annexe D.5, on discute plusieurs aspects liés au déploiement pratique de la politique $LogRD$ comme l'estimation du nombre de récepteurs en aval d'un lien, l'introduction de la politique $LogRD$ dans un réseau *Fair Scheduler* et le déploiement progressif de la politique $LogRD$.

3.4.4 Conclusion

On a introduit et évalué trois politiques d'allocation de la bande passante. On a utilisé pour l'évaluation des modèles analytiques simples mais pertinents et des simulations sur une large topologie hiérarchique. On en a conclu que la politique $LogRD$ offrait le meilleur compromis entre satisfaction et équité. Cette politique permet d'améliorer considérablement la satisfaction des utilisateurs multipoints sans pour autant diminuer de manière significative la satisfaction des utilisateurs point à point. Par conséquent, elle permet d'inciter les utilisateurs à se servir de la transmission multipoint et cela, sans un effet de bord néfaste pour les utilisateurs point à point. De plus, cette politique a permis d'apporter une réponse élégante à la question : « Comment allouer la bande passante d'un lien entre un flux point à point ne servant qu'un seul récepteur et un flux multipoint servant un million de récepteurs ? » Une allocation de la bande passante qui prend en compte de manière logarithmique le nombre de récepteurs est une allocation qui offre une solution raisonnable au problème de l'allocation de la bande passante entre flux multipoints et flux point à point.

Chapitre 4

Conclusion

4.1 Résumé des contributions

Une des clefs de l'amélioration de la qualité de service pour les réseaux *best effort* est le contrôle de congestion. On a exploré, dans cette thèse, une voie de recherche peu exploitée : comment améliorer les propriétés des protocoles de contrôle de congestion – point à point et multipoints – dans les réseaux *best effort* en s'affranchissant du paradigme *TCP-friendly* ? Notre étude des protocoles RLM et RLC nous a permis d'identifier quelques comportements pathologiques fondamentaux de ces protocoles qui rendent leur déploiement difficile. Ces comportements pathologiques sont difficiles à corriger dans le contexte actuel de l'internet, c'est-à-dire en respectant le paradigme *TCP-friendly*. Ceci nous a conduit à réfléchir au problème du contrôle de congestion dans le contexte plus général des réseaux *best effort*. On a, avec le même formalisme mathématique, redéfini la notion de congestion, défini les propriétés requises pour un protocole de contrôle de congestion idéal et défini un paradigme, le paradigme FS, pour la conception des protocoles de contrôle de congestion presque idéaux. Le paradigme FS est le premier paradigme pour la conception de protocoles de contrôle de congestion défini et prouvé formellement. Pour valider de manière pragmatique le paradigme FS, on a conçu, grâce à ce dernier, un nouveau protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur : PLM. Ce protocole est capable de suivre les évolutions de la bande passante disponible sans aucune perte induite, même dans un environnement autosimilaire et multifractal. PLM surpasse RLM et RLC et valide le paradigme FS. On a finalement défini une nouvelle politique d'allocation de la bande passante entre flux point à point et flux multipoints qui s'intègre parfaitement dans la contrainte de réseau du paradigme FS. Cette politique permet d'améliorer considérablement la satisfaction des utilisateurs multipoints sans nuire aux utilisateurs point à point. Cette politique appelée $LogRD$ donne une solution performante et élégante au problème de l'allocation de la bande passante entre flux point à point et flux multipoints.

4.2 Discussion sur les contributions

Depuis plusieurs années, on annonce le déploiement de solutions multipoints dans l'internet. Cependant, à part quelques exceptions, force est de constater que la transmission multipoint n'est pas accessible au grand public. Le multipoint dans l'internet est basé sur IP multipoint qui n'a pas été conçu avec l'idée d'être exploité commercialement [21]. De nombreuses fonctionnalités manquent à IP multipoint : la gestion de groupes multipoints, la sécurité multipoint, l'allocation d'adresses multipoints, la facturation de services multipoints, etc. Bien que d'actives recherches soient menées dans tous ces domaines, la communauté des réseaux se divise entre ceux qui pensent que tôt ou tard le multipoint sera déployé dans l'internet et ceux qui pensent que le multipoint n'est plus qu'un sujet académique sans aucun avenir. Étant donné qu'une grande partie de cette thèse est sur la transmission multipoint, doit-on considérer notre travail comme inutile si la transmission multipoint ne devient pas une fonctionnalité offerte au grand public ?

D'autre part, pour appliquer le paradigme FS il faut avoir un réseau FS, c'est-à-dire un réseau où tous les routeurs implémentent une politique d'ordonnancement de type FS. Or, il s'agit d'une hypothèse forte puisque, actuellement, soit les routeurs ne possèdent pas un mécanisme de type FS, soit les routeurs possèdent effectivement un mécanisme de type FS mais qui n'est pas activé. Doit-on considérer le paradigme FS comme un paradigme irréaliste et par conséquent sans intérêt ?

Notre réponse à ces deux questions est « non ! » Trop souvent, les pressions économiques et politiques poussent la communauté scientifique à considérer que les recherches qui ne sont pas applicables à court terme ne sont pas dignes d'intérêt. Cette habitude est dangereuse car il est très difficile, en travaillant à court terme, d'apporter des idées originales. Or c'est justement le rôle des chercheurs d'apporter des idées originales qui permettent de trouver de nouvelles orientations pour demain.

La transmission multipoint a posé et pose toujours de grands défis à la communauté scientifique. Cependant, les résultats de recherche sur la transmission multipoint ont souvent un champ d'application beaucoup plus large que la transmission multipoint elle même. L'exemple le plus flagrant est celui des réseaux *overlay* (*overlay network*). Cette technique permet, entre autres choses, la diffusion de contenus multimédias sur l'internet. Or si l'on regarde qui sont les pionniers des réseaux *overlay*, on retrouve des spécialistes de la transmission multipoint (par exemple Steven McCanne avec *Fast Forward Networks* [25] ou Jörg Nonnenmacher avec *Castify Networks* [10]). La technique des réseaux *overlay*, actuellement appliquée dans l'internet, n'aurait sans doute pas été possible sans la compréhension du problème de la diffusion de contenus multimédias avec la transmission multipoint.

Le paradigme FS, bien que basé sur une contrainte réseau qui n'est pas réalisable pour le

moment, a permis de montrer qu'il était possible de considérablement simplifier et améliorer la conception des protocoles de contrôle de congestion. Ce résultat fondamental montre qu'il peut être très profitable de prendre du recul par rapport au paradigme *TCP-friendly* et va, on l'espère, encourager les recherches dans cette voie.

En conclusion, bien que cette thèse ne présente pas de solutions que l'on puisse directement appliquer au contexte actuel de l'internet, elle donne un aperçu des solutions qui feront, on l'espère, le succès des réseaux *best effort* de demain.

Appendix A

Pathological Behaviors for RLM and RLC

Abstract

RLM [55] and RLC [87] are two well known receiver-driven cumulative layered multicast congestion control protocols. They both represent an indisputable advance in the area of congestion control for multimedia applications. However, there are very few studies that evaluate these protocols, and most of the time, these studies conclude that RLM and RLC perform reasonably well over a broad range of conditions.

In this paper, we evaluate both RLM and RLC and show that they exhibit fundamental pathological behaviors. We explain in which context these pathological behaviors happen, why they are harmful, and why they are inherent to the protocols themselves and cannot be easily corrected. Our aim is to shed some light on the fundamental problems with these protocols.

Keywords: RLM, RLC, Pathological behaviors, Congestion Control, Multimedia, Multicast, Cumulative layers.

A.1 Introduction

Multimedia applications will probably become some of the most popular applications in the Internet. One fundamental problem when introducing a new application in the Internet is to find an efficient way (for both the application and the network) to do congestion control. Cumulative layered multicast congestion control protocols are presented as the best solution for the dissemination of multimedia content to a heterogeneous set of receivers (see for instance [55, 87, 85, 50]). Therefore, these applications are the subject of active research.

Steven McCanne et al. introduced the first receiver-driven cumulative layered multicast congestion control protocol called RLM [55]. The behavior of RLM is determined by a state machine where transitions among the states are triggered by the expiration of timers (the join-timer and the detection-timer) or the detection of losses. The maintenance of the timers and the

loss estimator are fundamental parts of the RLM protocol. In order to scale with the number of receivers, RLM needs an additional mechanism called *shared learning*. McCanne evaluated RLM for simple scenarios and only considered inter-RLM interaction. He found that RLM can result in high inter-RLM unfairness. Bajaj et al. [2] explored the relative merits of uniform versus priority dropping for the transmission of layered video. They found that RLM performs reasonably well over a broad range of conditions, but performs poorly in extreme conditions like bursty traffic. Gopalakrishnan et al. [35] studied the behavior of RLM for VBR traffic and show that RLM exhibits high instability for VBR traffic, has very poor fairness properties in most of the cases, and achieves a low link utilization with VBR traffic.

A TCP-friendly version of RLM, called RLC, was introduced by Vicisano et al. [87]. RLC is based on the generation of periodic bursts that are used for bandwidth inference and on synchronization points (SP) that indicate when a receiver can join a layer. The TCP-friendly behavior is mainly due to the exponential distribution of the layers that results in an exponential decrease of the bandwidth consumed (like TCP) in case of losses. While the exponential distribution of the layers is not a requirement for the TCP-like behavior if the protocol drops the layers in an exponential way, it considerably simplifies the protocol. We are not aware of any study considering another layer distribution. Vicisano found that RLC can be unfair with TCP for large packet sizes.

According to these previous studies, RLM and RLC seem to perform reasonably well in a broad range of cases. However, in this paper, we evaluate both RLM and RLC with very simple scenarios and show that they exhibit pathological behaviors. We explain in which context these pathological behaviors happen, why they are harmful, and why they are inherent to the protocols themselves and cannot be easily corrected. Our aim is to shed some light on the fundamental problems with RLM or RLC.

The paper is organized as follows. In section A.2 we present the scenarios considered for the simulations. We discuss the results of the simulation for RLM in section A.3, and for RLC in section A.4. We conclude the paper in section A.5.

A.2 Simulation Topologies

Fig. A.1 shows the three topologies used to evaluate the behavior of RLM and RLC. A source and a receiver, when not specified, refer to a RLM (or RLC) source and receiver, respectively. The first topology, Top_1, consists of one source and four receivers. We evaluate the speed, the accuracy, and the stability of the convergence in the context of a large heterogeneity of link bandwidths and link delays. The second topology, Top_2, consists of one source and m receivers. For all the simulations, the links (N_1, R_M) have a bandwidth uniformly chosen in $[500, 1000]$ Kbit/s and a delay uniformly chosen in $[5, 150]$ ms. We evaluate the scalability with respect

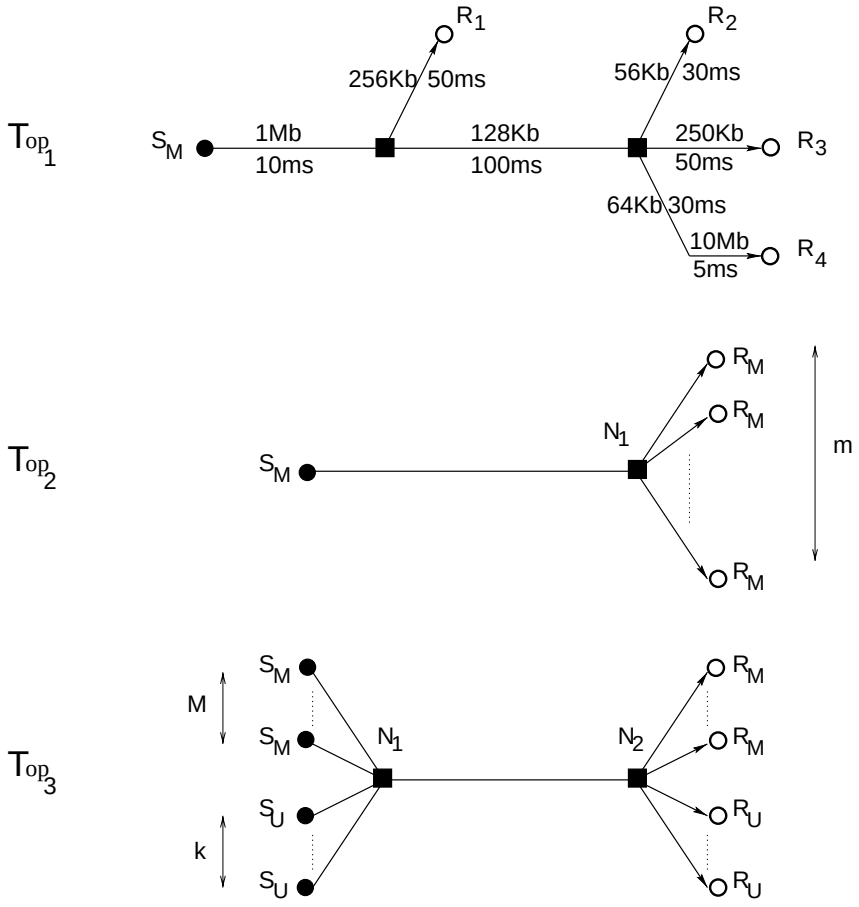


Figure A.1: Simulation Topologies.

to session size. The last topology, Top_3, consists of M multicast sources (with one receiver), and k unicast sources. For all the simulations, the links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We evaluate the scalability of the multicast protocol with an increasing number of multicast sessions and with an increasing number of unicast sessions. Also, we evaluate the fairness of the multicast protocol towards the unicast sessions.

We evaluate RLM and RLC using the ns [62] simulator. We use the following default parameters for our simulations: The multicast routing protocol is DVMRP (in particular graft and prune messages are simulated). We chose the packet size for all the flows (RLM, RLC, CBR, and TCP) to be 500 bytes.

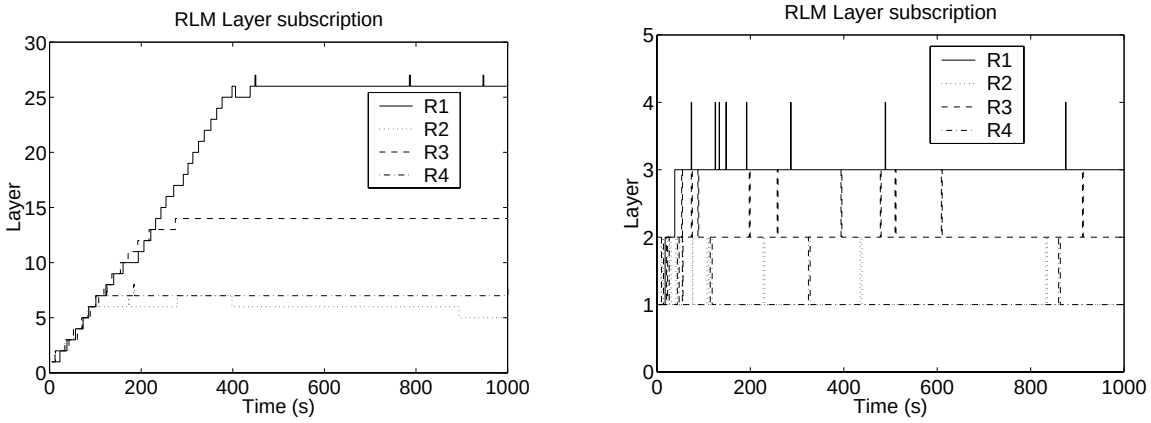
RLM and RLC are designed for FIFO scheduling. However, we made all the simulations for both FIFO and FQ scheduling; in a given simulation, all the queues are either Fair Queuing (FQ) queues with a shared buffer or FIFO queues. The main reason for considering FQ scheduling is to evaluate how FQ impacts the behavior of RLM and RLC[1].

[1]Another reason is the following: In [50] we introduce a new cumulative layered multicast congestion control

A.3 Pathological behaviors of RLM

We use the ns implementation of RLM with the parameters as chosen by McCanne in [55]. For all the simulations, the buffer size (or shared buffer size for FQ) is 20 packets. We run all the simulations for RLM for a duration of 1000 seconds.

In several places, in this section, we consider thin layers (typically 10 Kbit/s or 20 Kbit/s layers granularity). We do not argue that thin layers are reasonable, practically applicable, etc. (Linda Wu et al. [88] study an architecture exploiting thin layers/streams). In fact, we use thin layers as a diagnosis tool; thin layers clearly exhibit pathological behaviors that still hold with coarse layers. However, directly using coarse layers does not allow to easily find if there is a pathological behavior and what is the reason of this pathological behavior.



(a) Layer subscription for each RLM receivers, 10 Kbit/s layers.

(b) Layer subscription for each RLM receivers, exponential layers ($2^i \cdot 32Kbit/s$, $i = 0, 1, 2, 3, \cdots$).

Figure A.2: Speed, accuracy, and stability of RLM convergence for a single session, Top_1.

The first simulation evaluates the speed, the accuracy, and the stability of RLM convergence on Top_1. We consider 10 Kbit/s layer granularity. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). We see in Fig. A.2(a) the very slow convergence time of RLM. Receiver R_1 needs more than 400 seconds to converge to the optimal rate. Moreover, the mean loss rate for this simulation is

protocol called PLM. This protocol requires a Fair Queuing network (i.e. a network where every queue is a FQ queue). In order to compare PLM with RLM and RLC, we must consider the same scenarios (the scenarios in this paper are a subset of the scenarios in [50]) and in particular, the same scheduling discipline. Moreover, as FQ improves the performance of RLM and RLC, it is fair to consider FQ for the comparison between these protocols and PLM. We find that PLM outperforms in all the cases RLM and RLC.

3.2%. The 10 Kbit/s layers granularity is a tough test for RLM, and shows a pathological behavior of RLM in extreme cases. The slow convergence time is explained by the value of the minimum join-timer of RLM that is fixed to 5 seconds. The smaller the layer granularity, the slower the convergence. The significant loss rate is explained by the loss threshold of RLM set to 25%. With such small layers, we never enter in a congestion period where a receiver experiences a loss of more than 25% of the packets. Each receiver sees a persistent loss rate for the whole simulation that results in a mean loss rate of 3.2%. As a receiver can only do a join experiment if he does not see losses during a given period of time, there is very low number of join experiments in this simulation. We made another simulation with exponential layer sizes starting at 32 Kbit/s (the layer bandwidth distribution is $\{32,64,128,256,512,1024\}$ Kbit/s) and give the results in Fig. A.2(b). In this case RLM performs significantly better than in the previous case. The convergence time is reasonably fast. We clearly see the join experiments that are, in this case, the main reason for a mean loss rate of 0.81%.

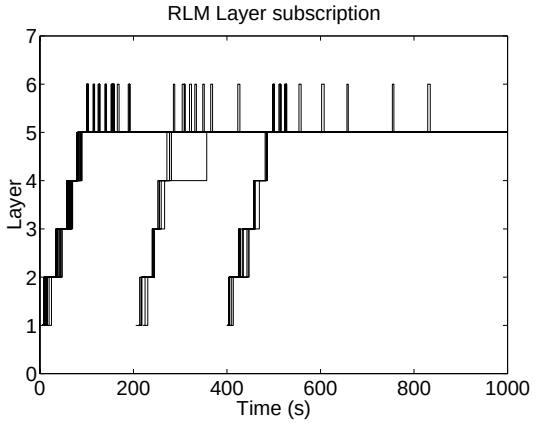


Figure A.3: Scaling of a RLM session with respect to the number of receivers, Top_2.

The second experiment evaluates the scaling of a single RLM session with respect to the number of receivers on topology Top_2. We consider 50 Kbit/s layer granularity. For this simulation, we consider the link (S_M, N_1) with a bandwidth of 280 Kbit/s and a delay of 20 ms. We start 20 RLM receivers at time $t = 5$ s then we add one receiver every five seconds from $t = 205$ s to $t = 225$ s, and at $t = 400$ s we add 5 more RLM receivers. The aim of this experiment is to evaluate the impact of the number of receivers on the convergence time and on the stability, and to evaluate the impact of late joins. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). The most interesting event in Fig. A.3 is the *receiver synchronization*. Due to the shared learning, receivers cannot join upper layers while there are some receivers subscribed only to lower layers. Indeed, the shared learning precludes a receiver to do a join experiment if there is a pending join experiment for a lower layer. Late joins can slow down the convergence time for RLM receivers. We

did the same experiment with exponential layers and observed a similar behavior.

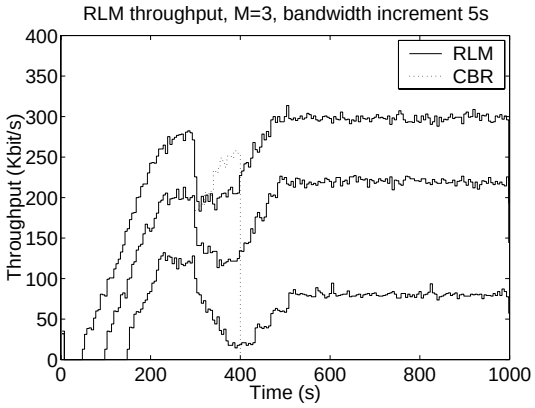
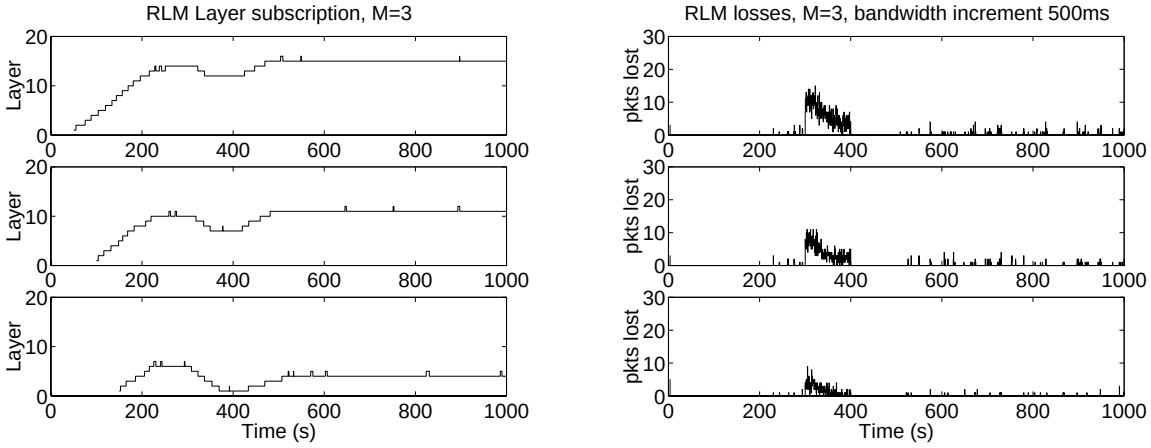


Figure A.4: Mean throughput of RLM and CBR flows sharing the same bottleneck, FIFO scheduling, Top_3.



(a) Layer subscription of each RLM session.

(b) Loss rate of each RLM session.

Figure A.5: RLM and CBR flows sharing the same bottleneck, FIFO scheduling, Top_3.

The third experiment considers a mix of RLM and CBR flows on Top_3. We consider a layer granularity of 20 Kbit/s. We comment this experiment for both FIFO and FQ scheduling. For FIFO scheduling, we consider $M = 3$ RLM sessions and $k = 1$ CBR flow. The bandwidth of link (N_1, N_2) is $200 * M = 600$ Kbit/s and the delay is 20 ms. We start each of the three RLM receivers at times $t = 50, 100, 150$ s and the CBR source at time $t = 300$ s; we stop the CBR source at $t = 400$ s. The CBR source rate is 300 Kbit/s, half the bottleneck bandwidth. The aim of this scenario is to study in the first part (before starting the CBR source) the behavior of RLM with an increasing number of RLM sessions, and in the second part (after starting the

CBR source) the behavior of RLM in case of severe congestion. When the CBR source stops we observe how fast RLM grabs the available bandwidth.

Fig. A.4 shows the mean throughput of the three RLM sessions and Fig. A.5(a) shows the layer subscription for the three RLM receivers. There is a slow convergence due to the small layer granularity. We see also a high unfairness among the sessions during the whole simulation. Moreover, the high period of congestion (when the CBR source sends packets) results in a large number a losses for the RLM sessions (see Fig. A.5(b)). When the CBR source starts and creates congestion, the RLM sessions start dropping layers. However, the process of dropping layers with RLM is very conservative (sluggish) and induces significant transitory losses (see Fig. A.5(b)). Indeed, a receiver can only drop one layer per detection-timer period. The mean loss rate is 2.3% in this experiment. We note the same effect as in experiment one: The small layers result in losses that never exceed the loss threshold (see Fig. A.5(b)), therefore never result in a layer drop, and result in a very low number of join experiments (see Fig. A.5(a)). We did the same simulation with exponential layers. As expected, the large layer granularity results in a higher reactivity for RLM. When the CBR source starts, RLM reacts fast to the congestion by dropping one layer (dropping one layer is enough in this case to avoid congestion). The resulting mean loss rate is reduced to 1.4%. However, RLM results in a very high unfairness in case of exponential layers as well. The first session gets roughly 500 Kbit/s, the second gets roughly 100 Kbit/s, and the third session must drop all the layers.

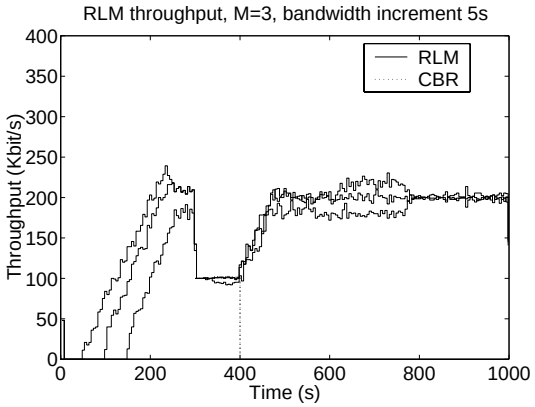
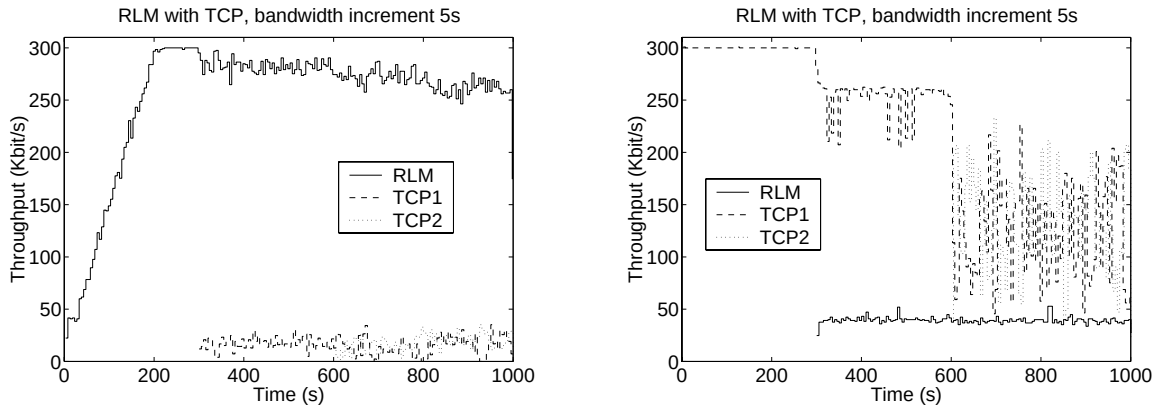


Figure A.6: Mean throughput averaged over 5s intervals, FQ scheduling, Top_3.

For FQ scheduling, we consider $M = 3$ RLM sessions and $k = 3$ CBR flows. The bandwidth of link (N_1, N_2) is $200 * M = 600$ Kbit/s and the delay is 20 ms. We start each of the three RLM receivers respectively at time $t = 50, 100, 150$ s. We start the CBR sources at time $t = 300$ s and stop the CBR sources at $t = 400$ s. The rate of each CBR source is 500 Kbit/s. We choose as many CBR sources as RLM sessions to simulate severe congestion. Indeed, with FQ, the only way to create congestion is to significantly increase the number of sessions. In this case, the three CBR sources grab half of the bottleneck bandwidth.

Fig. A.6 shows the mean throughput for the three RLM sessions. The most noticeable point, compared to the FIFO scheduling case, is the good fairness among the RLM sessions. However, even with FQ scheduling, the fairness is not ideal (see Fig. A.6 between $t = 400$ s and $t = 800$ s). The mean loss rate for this simulation is 4.6%. As FQ enforces fairness among all the flows, the RLM flows cannot grab more bandwidth than their fair share. While, with FIFO scheduling a RLM flow can grab more bandwidth than its fair share from the CBR flow. Therefore, the RLM receivers experience more losses with FQ than with FIFO. We do not notice any other significant difference compared to the FIFO scheduling case. We did the same simulation with exponential layers and observed a good fairness among the RLM flows (according to the layer granularity). RLM reacts fast to the congestion and the resulting mean loss rate is lower than 1%.



(a) RLM session starts first.

(b) RLM session starts after TCP1.

Figure A.7: Mean throughput of RLM and TCP flows sharing the same bottleneck, FIFO scheduling, Top_3.

The fourth experiment considers a mix of one RLM session and TCP flows on Top_3. We consider $M = 1$ RLM session and $k = 2$ TCP flows and a layer granularity of 20 Kbit/s. The bandwidth of link (N_1, N_2) is $100 * (M + k) = 300$ Kbit/s and the delay is 20 ms. We do all the simulations for FIFO and FQ scheduling. In a first set of simulations, we start RLM first at $t = 0$ s, then TCP1 at $t = 300$ s, and TCP2 at $t = 600$ s. In a second set of simulations, we start TCP1 first at $t = 0$ s, then RLM at $t = 300$ s, and TCP2 at $t = 600$ s. For FQ scheduling, the simulations do not bring any new results compared to the previous experiment. In summary, with FQ scheduling, RLM shares fairly the bandwidth with TCP (according to the layer granularity), and experience a transitory period of congestion when a new TCP flow starts. This period of congestion results in a significant loss rate (from to 2% to 8% according to the simulation scenario) with 20 Kbit/s layer granularity, and in a low loss rate (around 0.5% for all

the scenario) with exponential layers.

In the following we consider FIFO scheduling. Fig. A.7 shows the mean throughput averaged over 5 seconds intervals of the RLM and TCP flows for FIFO scheduling. When RLM starts first it grabs all the available bandwidth. TCP can only achieve a very small throughput (see Fig. A.7(a)) due to the hysterisis state and to the large RLM loss threshold of 25%. Indeed, when a RLM receiver is in the steady state, if he experiences congestion he enters the hysteresis state for a detection-timer period (in order to filter out transitory periods of congestion). At the end of the hysteresis state, the receiver measures the loss rate that must exceed the loss threshold to drop a layer. However, TCP is not able to create a large enough congestion and therefore fails to grab bandwidth from RLM. When RLM starts after TCP1, RLM is not able to grab bandwidth from TCP. This is due to the join experiment process of RLM. When a RLM receiver does a join experiment and experiences losses during this join experiment, he infers that it cannot join this layer. Moreover, in order to do a join experiment, a receiver must not see any loss during a given period of time. The key point is: whereas a RLM receiver in steady state needs a 25% loss rate to drop a layer, a RLM receiver needs only one loss to infer than he cannot join a layer or to preclude a join experiment (the reader can refer to [55] for all the details about the RLM protocol).

In conclusion, we found several pathological behaviors of RLM: i) The minimum join timer gives a large lower bound to the speed of convergence; ii) The high loss threshold can result in a high mean loss rate. Moreover, it results in a very aggressive behavior when competing with TCP. iii) The shared learning results in receiver synchronization; iv) The join experiment process results in a very conservative behavior when competing with TCP flows; v) The conservative drop process (one layer dropped per detection-timer) results in extended transient periods of losses in case of congestion.

Each of these pathological behaviors is very hard to correct as the parameters involved are the result of complex tradeoff. The minimum join timer is a tradeoff between the speed of convergence of the frequency of the join experiments. The loss threshold is a tradeoff between a conservative and a reactive behavior in case of loss. One solution is for both, the join timer and the loss threshold, to dynamically adjust these parameters according to the network conditions. However, that requires complex network inference mechanisms: an additional (large time scale) bandwidth inference mechanism to infer if a receiver needs to add several or only few layers to reach the equilibrium; an additional congestion inference mechanism to determine if the congestion is heavy (one needs to drop several layers to reach the equilibrium) or light (one needs to drop only one layer to reach the equilibrium). These questions need further research. The shared learning and the join experiment process are foundations of the RLM protocols and cannot be changed without redesigning the whole protocol. Finally, the conservative drop process is necessary for RLM to avoid over-reaction to losses and is, therefore, very hard to

tune.

A.4 Pathological behaviors of RLC

We use the ns implementation of RLC with the parameters as chosen by Vicisano in [87]. We identify behaviors in the ns version of RLC that are not conform with the description of RLC in [87]. We do not correct these behaviors as we do not know if they are intended by the authors or if they are the result of a bug. We always take into account these behaviors in our simulations and discuss them when they impact the results. The main *peculiar* behavior is that RLC drops the current layer when it experiences losses during a burst, whereas, according to [87], RLC should stay at the current layer and just infer that it cannot join an upper layer.

RLC can be considered a TCP-friendly version of RLM with the improvement of the synchronization points (data packets with a special flag) and a new bandwidth inference mechanism based on periodic bursts. In fact, we show that both the synchronization points and the periodic bursts lead to pathological behaviors, and that the RLC behavior is very sensitive to the queue size.

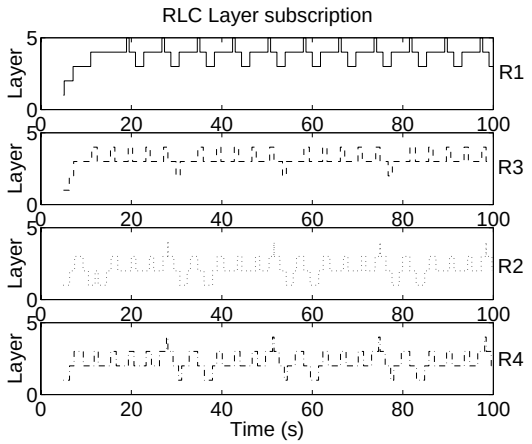


Figure A.8: Layer subscriptions for a single session, 4 receivers, Top_1.

For all the simulations with RLC, we just indicate the rate B_0 of the base layer L_0. The rate of layer L_i is $B_i = 2^i \cdot B_0$. If not specified, the default buffer size (or shared buffer size for FQ) is 20 packets. The first simulation evaluates the speed, the accuracy, and the stability of RLC convergence for Top_1. The rate of the base layer is 32 Kbit/s. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). The queue size is 15 packets. Fig. A.8 shows the layer subscription for the RLC receivers. The solid line is for R_1, the dashed line is for R_3, the dotted line is for R_2, and the dashed-dotted line is for R_4. This simple experiment shows one of the most fundamental

problem with RLC. For instance, when R_1 subscribes to layer 4, he receives 256 Kbit/s. As his bottleneck bandwidth is 256 Kbit/s, he experiences no loss. The source sends periodically a burst that doubles, over a short period of time, the sending rate to allow the receiver to infer if he can join a higher layer. However, the burst does not make the queue overflow, and R_1 infers that he can join layer 5. After a short period of time, R_1 will experience a large number of losses and will drop the layer. For receiver R_1, we observe a cascade drop from layer 5 to layer 3. However, this cascade drop is due to the peculiar behavior pointed out at the beginning of the section. Indeed, just after dropping layer 5, the queue will remain full (as the bottleneck bandwidth is equal to the layer 4 rate), the source will generate a burst that makes the queue overflow as the queue is already full before the burst. The receiver will experience losses during the burst and due to the peculiar behavior will drop the layer 4. We can explain the behavior of the other receivers in the same way. The periodic erroneous bandwidth inference leads to a mean loss rate up to 13%.

This experiment shows a fundamental pathological behavior of RLC. RLC's bandwidth inference is based on the generation of periodic bursts that aim to reduce the transitory period of congestion due to join experiments (see [87] for more details). To succeed, the burst must make the queue overflow when there is not enough bandwidth to accommodate a new layer. However, queue overflow happens in our simulations *only* for a very judicious choice of the queue sizes, which is impossible to do in a real network. As the bandwidth inference does not succeed, the receivers periodically join a layer when there is not enough bandwidth available to add this layer. That leads to periodic congestion and periodic losses.

To avoid cascade drop, RLC uses a deaf period of fixed length after dropping a layer during which it does not drop layers. However, this deaf period reflects the delay between the time the receiver sends a leave request and the time the receiver sees the effect of the leave request on the bottleneck router. This value varies highly over time and for different receivers. As the join experiments are sender-based in RLC, there is no way for a receiver to infer the appropriate duration for the deaf period without adding a complex protocol. This is a significant weakness of RLC as a correct static choice of the deaf period can be very difficult. If RLC must drop several layers to react to a severe period of congestion, the deaf period will significantly slow down the drop process. However, we note that with exponentially distributed layers, dropping one layer is most of the time sufficient to react to congestion.

The second experiment evaluates the scaling of a single RLC session with respect to the number of receivers on topology Top_2. For this simulation we consider the link (S_M, N_1) with a bandwidth of 250 Kbit/s and a delay of 20 ms. The queue size is 10 packets. We start 20 RLC receivers at time $t = 5$ s then we add one receiver every five seconds from $t = 30$ s to $t = 50$ s, and at $t = 80$ s we add 5 RLC receivers. The rate of the base layer is 8 Kbit/s. The aim of this experiment is to evaluate the impact of the number of receivers on the convergence

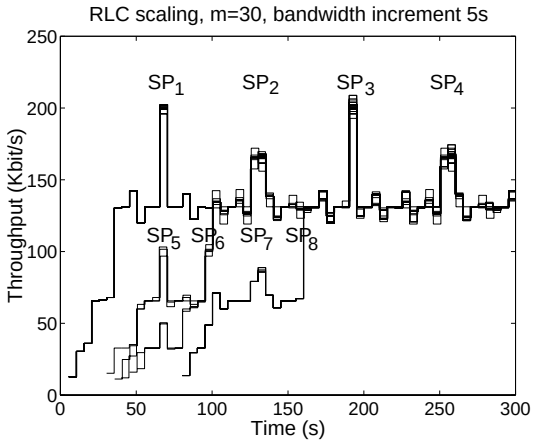


Figure A.9: Scaling of a RLC session with respect to the number of receivers, Top_2.

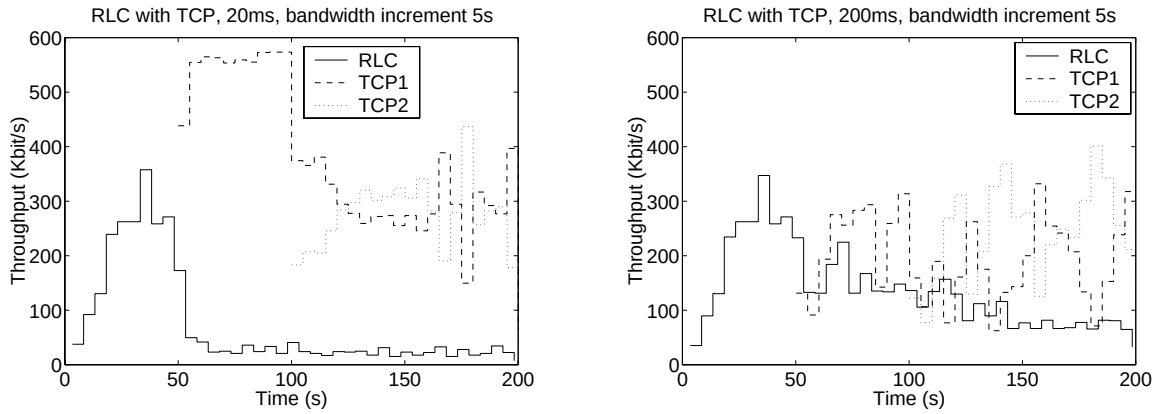
time and on the stability, and to evaluate the impact of late joins. We only present the results for FIFO scheduling (FQ scheduling gives the same result as, in this experiment, we have only one source). A receiver can only increase his number of layers at synchronization points (SP) if no losses are experienced during the burst preceding that SP. The distance between two SPs doubles at each layer, and the SPs at layer L_i are a subset of the SPs at layer L_{i-1} (see [87] for more details). Fig. A.9 shows the mean throughput for all the receivers. We first note that the small throughput oscillations around the mean throughput are due to the succession of periodic burst and silent period that slightly increases or decreases the mean throughput averaged over 5 seconds intervals. The annotations SP_i indicate the occurrence of some relevant SPs. In this simulation, the bandwidth inference using bursts never succeeds, i.e. the bursts never make the queue overflow, and the receivers join an additional layer that the network cannot support. We observe a new pathological behavior of RLC. Between $t = 30$ s and $t = 50$ s late joiners start. Around $t = 60$ s, at the synchronization point SP_5, some late joiners join layer 5[2] and the others join layer 4. But, as the synchronization point SP_1 is synchronized with SP_5, the first receivers (that join at $t = 5$ s) join layer 6 that cannot be supported. This results in a period of congestion that is misinterpreted by the late joiners who drop a layer. The late joiners can only subscribe to the highest layer supported at SP_6, which is not synchronized with an upper layer SP. We observe the same pathological behavior with the late joiners that start at $t = 80$ s. This pathological behavior significantly slows down the convergence speed. We note that, even if the burst succeeds in inferring the available bandwidth, the same problem persists. Indeed, if the burst (to join layer 6) makes the queue overflow, the first receivers will infer that they cannot join layer 6 and they will stay at the current layer at SP_1. However, the late joiners cannot join an upper layer at SP_5 as they will see losses, shared among all the layers, due to the burst on

[2] In this simulation layer 4 corresponds to a 64 Kbit/s, layer 5 corresponds to 128 Kbit/s, and layer 6 corresponds to 256 Kbit/s.

layer 5.

With the parameters choice in [87], the SPs are exponentially spaced. At layer i, the distance between the SPs is $2^i \cdot 8 \cdot \frac{s}{B_0}$, where s is the packet size and B_0 is the throughput of the base layer. For $B_0 = 16$ Kbit/s and $s = 256$ bytes, the distance between the SPs at layer i is roughly 2^i seconds. For instance, a receiver can only join layer 6 every 64 seconds. The exponentially spaced SPs can significantly slow down the convergence of the receivers to the highest layers.

We did a third experiment that considers the same scenarios than the third experiment for RLM. We do not give plots for this experiment as it does not exhibit pathological behaviors. For this experiment, RLC performs reasonably well. The RLC sessions share fairly the bandwidth among each other and adapt reasonably fast to the transitory period of congestion produced by the CBR source(s). The mean loss rate for all the scenarios range from 0.6% to 2.9%.



(a) The delay of the link (N_1, N_2) is 20 ms.

(b) The delay of the link (N_1, N_2) is 200 ms.

Figure A.10: Mean throughput of RLC and TCP flows sharing the same bottleneck, Top_3.

The fourth experiment on Top_3 considers a mix of RLC and TCP flows. We consider $M = 1$ RLC session and $k = 2$ TCP flows. The bandwidth of link (N_1, N_2) is $200 * (M + k) = 300$ Kbit/s and the delay varies from 20 ms to 400 ms. The rate of the base layer is 16 Kbit/s. We start RLC at $t = 0$ s, TCP1 at $t = 50$ s, and TCP2 at $t = 100$ s. We did all the simulations for both FIFO and FQ scheduling. For FQ scheduling, we do not see any pathological behavior and do not present the plots. In this case, RLC shares fairly (according to the layer granularity) the bandwidth with the TCP flows. For these scenarios, the mean loss rate range from 0.7% to 1.6%.

Now we comment the simulations for the fourth experiment with FIFO scheduling. Fig. A.10(a) shows the mean throughput averaged over 5 seconds intervals for the RLC and TCP flows when the delay of the link (N_1, N_2) is 20 ms. When TCP1 starts, RLC drops to layer 1 and then oscillates between layer 1 and layer 2. When TCP2 starts, we do not notice any

particular behavior for RLC. This experiment shows that RLC can be very conservative compared to TCP. Fig. A.10(b) shows the same experiment than previously except that the delay of the link (N_1, N_2) is 200 ms. We see that when TCP1 starts, RLC shares fairly the bandwidth with TCP1. When TCP2 starts, RLC gets a lower bandwidth than the two TCP flows. In a last experiment (we do not give the plot), we increase the delay of the link (N_1, N_2) to 400 ms. For this experiment, RLC fairly shares the bandwidth with TCP1 and TCP2. The explanation of this behavior is simple. The TCP cycle (i.e. the time between two losses) is shorter with a small RTT than with a large RTT. As a consequence, the smaller the RTT is, the larger the number of losses RLC experiences in a given time interval. As the RLC throughput is function of the number of losses, the higher the number of losses, the lower the RLC throughput.

In conclusion, we observed several pathological behaviors of RLC: i) The bandwidth inference mechanism based on burst leads to a high number of losses and does not succeed to make the queue overflow. ii) The synchronization points, as distributed in RLC, can significantly reduce the speed of convergence of late joiners. iii) The claimed TCP-friendly behavior of RLC results in a very conservative behavior of RLC compared to TCP.

Moreover, we cannot easily correct any of these pathological behaviors. For the periodic bursts to succeed, we must know how long the burst should persist in order to make the queue overflow. That requires a mechanism close to a bandwidth inference mechanism that renders to periodic burst useless. Moreover, the static choice of the burst length is a very difficult tradeoff between the probability to make the queue overflow and the amount of periodic congestion (and losses) generated. The pathological behaviors ii) and iii) raise new questions: Does RLC still achieve its claimed TCP-like behavior with non exponentially distributed layers? What is the influence of the placement of the SPs on the RLC behavior? These questions are for future research.

A.5 Conclusion

In this paper, we have evaluated RLM and RLC on simple scenarios. We show that both protocols exhibit pathological behaviors. We discuss which part of the protocol leads to a given pathological behavior and explain that most of the time these pathological behaviors are difficult to correct. We note that most of the problems come from the bandwidth inference mechanism used that is responsible for transient periods of congestion, instability, and periodic losses.

In [50] we present a new cumulative layered multicast congestion control protocol, called PLM, based on the generation of packet pairs (PP) to infer the available bandwidth. Bandwidth inference using PPs does not have any of the weaknesses of the bandwidth inference mechanisms of RLM and RLC, and PLM outperforms in all the cases RLM and RLC. However, PLM requires a Fair Queuing network. With a FIFO network, traditional solutions like RLM and

RLC are still necessary, but require improvements of the bandwidth inference mechanism. We hope that this paper contributes to identify the fundamental problems of these protocols, and will stimulate research to improve these protocols.

Appendix B

Beyond TCP-Friendliness: A New Paradigm for End-to-End Congestion Control

Abstract

The dominant paradigm for congestion control in the Internet today is based on the notion of TCP-friendliness. To be TCP-friendly, a source must behave in such a way to achieve a bandwidth that is similar to the bandwidth obtained by a TCP flow that would observe the same Round Trip Time (RTT) and the same loss rate. However, with the success of the Internet comes the deployment of an increasing number of applications that do not use TCP as a transport protocol. These applications can often improve their own performance by not being "TCP-friendly" which severely penalize TCP flows. Also, designing these new applications to be "TCP-friendly" is often a difficult task. For these reasons, we propose a new paradigm for end-to-end congestion control (the FS paradigm) that relies on a *Fair Scheduler* network and assumes only selfish and non-collaborative end users.

We rigorously define the properties of an ideal congestion control protocol and show that the FS paradigm allows to devise end-to-end congestion control protocols that meet almost all the properties of an ideal congestion control protocol. Moreover, the FS paradigm does not aversely impact the TCP flows. We show that the incremental deployment of the FS paradigm is feasible per ISP and leads to immediate benefits for the TCP flows.

Our main contribution is the formal statement of the congestion control problem as a whole that allows to rigorously prove the validity of the FS paradigm. Moreover, we explain how to apply the FS paradigm for the design of new congestion control protocols, and we introduce as a pragmatic validation of the FS paradigm a new multicast congestion control protocol called PLM.

Keywords: Congestion Control, Scheduling, Paradigm, Multicast, Unicast.

B.1 Introduction

Congestion Control has been a central research topic since the early days of computer networks. Nagle first identified the problems of congestion in the Internet[56]. The fundamental turning point in Internet congestion control took place in the eighties. Nagle proposed a strategy based on the round robin scheduling [57], whereas Jacobson proposed a strategy based on Slow Start (SS) and Congestion Avoidance (CA) [39]. Each of these solutions has its drawbacks. Nagle's solution has a high computational complexity and requires modifications to the routers. Jacobson's solution requires the collaboration of all the end users. The low performance of the routers and the small size of the Internet community at that time led to the adoption of Jacobson's proposal. SS and CA mechanisms were put into TCP. Ten years later, the Internet still uses Jacobson's mechanisms in a somewhat improved form [81].

We define the notion of *Paradigm for Congestion Control* as a model used to devise congestion control protocols that have the same set of properties. Practically, when one devises a congestion control protocol with a paradigm, one has the guarantee that this protocol will have a same set of properties as all the other congestion control protocols devised with this paradigm. However, the price to pay is that the paradigm imposes some constraints that need to be respected. The benefits of the paradigm come from the set of properties it guarantees. This notion of paradigm is not obvious in the Internet. A TCP-friendly paradigm was implicitly defined. However this paradigm was introduced after TCP, when new applications that can not use TCP had already appeared.

As TCP relies heavily on the collaboration of all the end users – collaboration is in the sense of the common mechanism used to achieve congestion control – the TCP-friendly paradigm was introduced (see [63], [30]) to devise congestion control protocols compatible to TCP. A TCP-friendly flow has to adapt its throughput T according to the equation:

$$T = \frac{C * MTU}{RTT * \sqrt{loss}} \tag{B.1}$$

where, C is a constant, MTU is the size of the packets used for the connection, RTT is the round trip time, and $loss$ is the loss rate experienced by the connection. To compute the throughput T, one needs to measure the loss rate and the RTT. The TCP-friendly equation models the TCP long-term behavior for low loss rate. Padhye et al. [64] introduced an TCP-friendly equation that is good approximation of the TCP long-term behavior even for high loss rate.

The throughput T for a TCP-friendly flow heavily decreases with the increase of loss rate $loss$. However, this behavior does not fit to many applications' requirements. For instance, audio and video applications are loss-tolerant and the degree of loss tolerance can be managed with FEC [7]. While these multimedia applications can tolerate a significant loss rate without a significant decrease in the quality perceived by the end users, they cannot tolerate frequent

variations of the throughput. The multicast flows suffer from TCP-friendliness since a source-based congestion control scheme for multicast flows has to adapt its sending rate to the worst receiver (in the sense of the throughput computed according to equation B.1), to follow the TCP-friendly paradigm. A receiver-based multicast congestion control scheme can be TCP-friendly but at the expense of a large granularity in the choice of the layer bandwidth [55] [87].

The TCP-friendly paradigm relies on the collaboration of all the users, which can no longer be assumed given the current size of the Internet [30]. This paradigm requires that *all* the applications adopt the same congestion control behavior based on Eq. (B.1), and it does not extend to the new applications being deployed across the Internet. Applications start to use non-TCP-friendly congestion control schemes (here congestion control may be a misleading expression, since the flows are often constant bit rate), as they observe better performance for audio and video applications than with TCP-friendly schemes. However, the benefit due to non-TCP-friendly schemes is transitory and an increasing use of non-TCP-friendly schemes may lead to a congestion collapse in the Internet [56]. Indeed, at the present time, most of the users access the Internet at 56 Kbit/s or less. However, with the deployment of xDSL most of the users will have, in a few years, an Internet access at more than 1 Mbit/s. It is easy to imagine the disastrous effect of hundred of thousands unresponsive flows at 1 Mbit/s crossing the Internet.

It is commonly agreed that router support can help congestion control. However there are several fears about router support. The end-to-end argument [78] is one of the major theoretical foundations of the Internet, and adding functionality inside the routers must not violate this principle. The end-to-end argument states that a service should only be implemented in the network if the network can provide the full service, or if this service is useful for all the clients. As TCP is the main congestion control protocol used in the Internet, router support must, at least, not penalize TCP flows [71]. Moreover it is not clear which kind of router support is desirable: router support can range from simple buffer management to active networking. One of the major reasons the research community distrusts network support is the lack of a clear statement about the use of network support for congestion control.

One simple way to use network support for congestion control is to change the scheduling discipline inside the routers. PGPS-like scheduling [65] is well known for its flow isolation property. This property sounds suitable for congestion control. However, the research community does neither agree on the utility of this scheduling discipline for congestion control, even if its flow isolation property is appreciated, nor on the way to use this scheduling discipline. We strongly believe that the lack of consensus is due to a fuzzy understanding about which properties a congestion control protocol should have and how a PGPS network, i.e. a network where each node implements a PGPS-like scheduler, can enforce these properties. The aim of this paper is to shed some light onto these questions.

A user acts selfishly if he only tries to maximize its own satisfaction without taking into account the other users (Shenker gives a good discussion of the selfishness hypothesis [79]). The TCP-friendly paradigm is based on cooperative and selfish users. We base our new paradigm called Fair Scheduler (FS) paradigm on non-cooperative and selfish users. We formally define the properties of an ideal congestion control protocol (see section B.2.2) and show that almost all these properties are verified with the FS paradigm when we assume a network support that simply consist in having a Fair Scheduler policy in the routers (see section B.2.3 for a definition of Fair Scheduler policy).

Our study shows that simply changing the scheduling policy allows to use the FS paradigm for congestion control, which outperforms the TCP-friendly paradigm. Indeed, the FS paradigm provides a basis for devising congestion control protocols tailored to the application needs. We do not want to replace or modify TCP. Instead, we propose an alternative to the TCP-friendly paradigm to devise *new* efficient congestion control protocols compatible with TCP. Important to us is that the FS paradigm does not violate the end-to-end argument, due to the network support. The weak network support that consists in changing the scheduling is of broad utility – we show that the Fair Scheduler policy significantly improves the performance of the TCP connections – and consequently does not violate the end-to-end argument [71]. While one part of our results is implicitly addressed in previous work (in particular [44] and [79]), we are making the step from an *implicit* definition of the problems to an *explicit* statement of the problem introducing a formalism that constitutes an indisputable contribution. Moreover, we show how to apply the FS paradigm for the design of a new congestion control protocol, and we introduce the protocol PLM as a pragmatic validation of the FS paradigm. We expect this study will stimulate the interest in the FS paradigm, which improves the behavior of the TCP flows and allows to devise end-to-end congestion control protocols that meet almost all the properties of an ideal congestion control protocol.

In section B.2 we define the FS paradigm for end-to-end congestion control. In section B.3, we study the practical aspects of the deployment of the FS paradigm in the Internet. Section B.4 compares the FS paradigm and the TCP-friendly paradigm. Section B.5 addresses the related work, while section B.6 summarizes our findings and concludes the paper.

B.2 The FS Paradigm

We formally define the FS paradigm in three steps. First, we define the notion of congestion. This definition is a slight modification of Keshav's definition[44]. Second, we formulate six properties that an ideal congestion control protocol must meet. These properties are abstractly defined, i.e. independent of any mechanism (for instance we talk about fairness but not about scheduling and buffer management, which are two mechanisms that influence fairness). Third,

we define the FS paradigm for congestion control. We show that almost all the properties of an ideal congestion control protocol are met by a congestion control protocol based on the FS paradigm.

We note that all the aspects of congestion control – from the definition of congestion to the definition of a paradigm to devise new congestion control protocols – are addressed with the same formalism. This formalism allows us to do a consistent study of the congestion control problem.

B.2.1 Definition of Congestion

The first point to clarify when we talk about congestion control is the definition of congestion. Congestion is a notion related to both user's satisfaction and network load. If we only take into account the user's satisfaction, we can imagine a scenario, where the user's satisfaction decreases due to jealousy, for instance, and not due to any modifications in the quality of the service a user receives. For instance, user A learns that user B has a better service and is no more satisfied with his own service. This can not be considered as congestion. If we only take into account the network load, congestion is only related to network performance, which can be a definition of congestion (for instance it is the definition in TCP), but we claim that we must take into account the user's satisfaction. We always have to keep in mind that a network exists to satisfy users. Our definition of congestion is:

Definition 1 *A network is said to be congested from the perspective of user i, if the satisfaction of i decreases due to a modification of the performance (bandwidth, delay, jitter, etc.) of his network connection.*

A similar definition was first introduced by Keshav [44]. Keshav's initial definition is : "A network is said to be congested from the perspective of user i if the satisfaction of i decreases due to an *increase* in network load". Our only one point of disagreement with Keshav is about the influence of network load. He says that only an *increase* in network load that results in a loss of satisfaction is a signal of congestion, whereas we claim that a *modification* (increase or decrease) in network load with a decrease of satisfaction is a signal of congestion. We give an example to illustrate our point of view.

Let the scheduling be WFQ [65], let the link capacity be 1 for all the links, and let the receiver's satisfaction depend linearly on the bandwidth seen (see Fig. B.1). The flow F_1 (sender S_1 and receiver R_1) has a weight of 1, the flow F_2 (sender S_2 and receiver R_2) has a weight of 2, the flow F_3 (sender S_3 and receiver R_3) has a weight of 1. In a first time the three sources have data to send, the satisfaction of R_1 is $\frac{1}{3}$, the satisfaction of R_2 is $\frac{2}{3}$ and satisfaction of R_3 is $\frac{2}{3}$. Then S_2 stops sending data, the satisfaction of R_1 becomes $\frac{1}{2}$ and the satisfaction of R_3

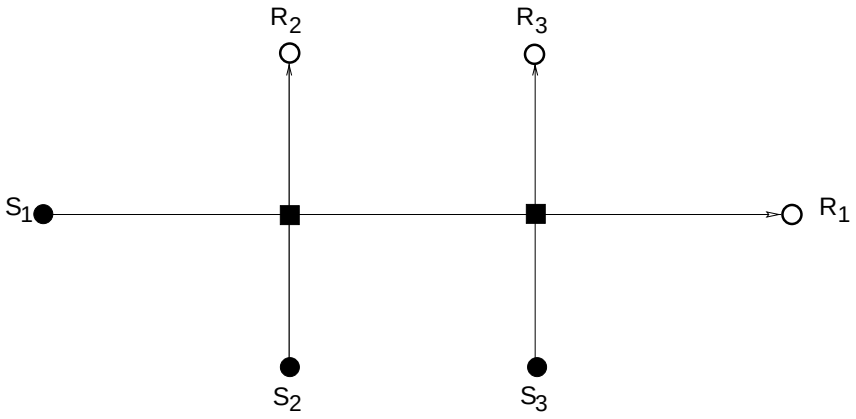


Figure B.1: Example for the definition of congestion.

becomes $\frac{1}{2}$. So when S_2 stops to send data, the network load decreases, but the satisfaction of R_3 decreases too. We consider this case as a congestion for R_3 in our definition, while Keshav's definition does not consider this case as congestion.

In the next section we will address the properties of an ideal congestion control protocol. We want such a congestion control protocol to avoid congestion! This is not trivial, in fact we want the congestion control protocol to avoid congestion in the sense of the congestion previously defined. This link is fundamental as it contributes to the consistency of our study.

B.2.2 Properties of an Ideal Congestion Control Protocol

We use through this section terminology from game theory and microeconomics; we define informally the terms used. The interested reader can refer to [79] for formal definitions. A network reaches a *Nash equilibrium* if, when every user acts selfishly, nobody can increase its own satisfaction. The bandwidth allocation A in a network is *Pareto optimal* if it does not exist another bandwidth allocation B such that all the users have a satisfaction with B higher or equal than the satisfaction with A, and at least one user has a satisfaction with B strictly higher than the satisfaction with A.

We discuss in the following a set of six abstract properties that an ideal congestion control protocol must meet. Whereas at the first sight these properties seem similar to the ones given by Keshav [44], they are fundamentally different. Indeed, most of our properties are expressed in mathematical terms that allow to rigorously prove that a congestion control protocol verifies these properties. Here, the only one assumption we make is the selfish behavior of the users. So these properties remain very general. The six properties of an ideal congestion control protocol are:

Stability Given each user is acting selfishly, we want the scheme to converge to a Nash equilibrium, where nobody can increase its own satisfaction. So this equilibrium makes sense

from the point of view of congestion control stability. Since the existence of more than one Nash equilibrium can lead to oscillations among these equilibria, the existence and the uniqueness of a Nash equilibrium are the conditions of stability.

Efficiency When the bandwidth allocation is Pareto optimal, nobody can have a higher satisfaction with another distribution of the network resources without decreasing the satisfaction of another user. This notion makes sense to guarantee the efficiency of a congestion control protocol. The convergence time of the scheme is another important parameter for efficiency. The faster the convergence, the more efficient the congestion control protocol. A fast convergence towards a Pareto optimal distribution of the network resources is the condition of efficiency.

Fairness It is perhaps the most delicate part of congestion control. Many criteria for fairness exist, but there is no criterion agreed on by the whole networking community. We choose to use max-min fairness as this is a reasonable notion of fairness. If we consider for all the users a utility function that is linearly dependent on the bandwidth seen, the max-min fair allocation is Pareto optimal. If a user does not have a utility function that depends linearly on the bandwidth seen he will not be able to achieve its fair share, in the sense of max-min fairness. Therefore max-min fairness defines/imposes an upper bound on the distribution of the bandwidth: If every user wants as much bandwidth as he can have, nobody will have more than its max-min share. But, if some users are willing to collaborate they can achieve another kind of fairness and in particular proportional fairness[43].

Robustness against misbehaving users. We suppose that all the users act selfishly, and as there is no restriction on the utility functions, the behavior of the users can be very aggressive. Such a user must not decrease the satisfaction of the other users. Moreover, he should not significantly modify the convergence speed of the scheme (see the efficiency property). Globally, the scheme must be robust against malicious, misbehaving, and greedy users.

Scalability The Internet evolves rapidly with respect to bandwidth, size, and the number of users. Inter-LAN, trans-MAN, and trans-WAN connections coexist. A congestion control protocol must scale on many axes: from an inter-LAN connections to a trans-WAN connections, from a 28.8 Kbit/s modem to a 2.4 Gbit/s line. Moreover, a congestion control protocol must scale with the number of receivers.

Feasibility This property contains all the technical requirements. We restrict ourself to the Internet architecture. The Internet connects a wide range of hardware and software systems, thus a congestion control protocol must cope with this heterogeneity. On the other hand, a congestion control protocol has to be simple enough to be efficiently implemented. To

be accepted as an international standard, a protocol needs to be extensively studied, the simplicity of the protocol will favor this process.

We believe that these properties are necessary and sufficient properties of an ideal congestion control protocol. Indeed these properties cover all the aspects of a congestion control protocol, from the theoretical notion of efficiency to the practical aspect of feasibility. However, it is not clear how we can devise a congestion control protocol that meets all these properties. In the next section, we establish the FS paradigm that allows to devise congestion control protocols that assure almost all of congestion control properties.

B.2.3 Definition and Validity of the FS Paradigm

A paradigm for congestion control is a model used to devise new congestion control protocols. A paradigm makes assumptions and under these assumptions we can devise compatible congestion control protocols; compatible means that the protocols have a same set of properties. Therefore, to define a new paradigm, we must clearly express the *assumptions* made and the *properties* guaranteed by the paradigm. To be viable in the Internet, the paradigm must be compliant with the end-to-end argument [78]. Mainly, the congestion control protocols devised with the paradigm have to be end-to-end and should not have to rely on specific network support. These issues are addressed in this section.

We first define the notion of Fair Scheduler policy.

Definition 2 (Fair Scheduler policy) *A Fair Scheduler policy is a per-packet approximation of a fluid GPS scheduling policy [65] with longest queue drop buffer management.*

We note that there are many approximations of the GPS scheduling policy (see [65], [20], and [4] for some examples). The better the approximation, the better the properties guaranteed by the FS paradigm. The WF^2Q scheduling policy [3] is a good approximation of the GPS fluid model that perfectly suits our paradigm.

For sake of simplicity, we make a distinction between the assumption that involves the network support, which we call that the Network Part of the paradigm (NP), and the assumptions that involve the end systems, which we call that the End System Part of the paradigm (ESP). The assumptions required for our new paradigm are:

- For the NP of the paradigm we assume a *Fair Scheduler* network, i.e. a network where every router implements a Fair Scheduler policy;
- For the ESP, the end users are assumed to be selfish and non-collaborative. This is a sufficient but not a necessary condition. In particular, collaboration among the users is possible if that increases their satisfaction.

We call this paradigm the Fair Scheduler (FS) paradigm[1]. With the TCP-friendly paradigm, the equation B.1 guarantees efficiency, stability, and fairness, however not in the sense as these three properties were defined for an ideal congestion control protocol in section B.2.2. Since TCP guarantees efficiency, stability, and fairness *by only one* mechanism at the end system, compromises between the three properties are unavoidable. The idea of the FS paradigm is to rely on the support of the network to guarantee the properties required for an ideal congestion control protocol, and to let the protocol at the end system only address the application needs. We note that the FS paradigm, unlike the TCP-friendly paradigm, does not make any assumptions on the mechanism used at the end systems. The FS paradigm assumes full freedom when devising a congestion control protocol. This characteristic of the paradigm is very appealing but may lead to a high diversity of the congestion control mechanisms used. Therefore, one may ask the question about the set of properties enforced by the FS paradigm. If the FS paradigm enforces fewer properties than the TCP-friendly paradigm, the FS paradigm does not make any sense. We show, in the following, that our simple FS paradigm enforces almost all the properties of an ideal congestion control protocol and consequently outperforms the TCP-friendly paradigm.

Stability Under the NP and ESP hypothesis, the existence and uniqueness of a Nash equilibrium is assured (see [79]). The congestion control protocols devised with the FS paradigm therefore meet the condition of stability.

Efficiency Under the NP and ESP hypothesis, even a simple optimization algorithm (like a hill climbing algorithm) converges fast to the Nash equilibrium. However, the Nash equilibrium is not Pareto optimal in the general case. If all the users have the same utility function, the Nash equilibrium is Pareto optimal. One can point out that ideal efficiency can be achieved with full collaboration of the users (see [79]). However, it is contrary to the ESP assumptions. The congestion control scheme devised with our new paradigm does not have necessarily ideal efficiency.

Fairness Every Fair Scheduler policy achieves max-min fairness. Moreover, as a Fair Scheduler policy is implemented in every network node, every flow achieves its max-min fairness rate on the long term average (see [36]). Our NP assumption enforces fairness.

Robustness Using a Fair Scheduler enforces that the network is protected against malicious, misbehaving, and greedy users (see [20]). While a user by opening multiple connections can increase its share of the bottleneck, we do not expect this multiple connections effect

[1]Like the TCP-friendly paradigm, we compose the name of our new paradigm using the name of the fundamental mechanism involved in the paradigm, namely the Fair Scheduler policy.

to be a significant weakness of the robustness property, as the number of connections that a single user can open is limited in practice.

Scalability According to the ESP assumption, selfish and non-collaborative end users is a sufficient condition. Unlike the TCP-friendly paradigm, the designer has a great flexibility to devise scalable end-to-end congestion control protocols with the FS paradigm.

Feasibility A Fair Scheduler policy (HPFQ [4]) can be implemented today in Gigabit routers (see [45]). So the practical application of the NP assumption is no longer an issue (see section B.3.2 for a discussion on the practical deployment of Fair Schedulers policy in the Internet). Moreover, even a simple algorithm will lead to an efficient congestion control protocol. The protocol will be easier to devise and easier to evaluate.

We see that the FS paradigm does not allow to devise an ideal efficient congestion control protocol, because the Nash equilibrium can not be guaranteed to be Pareto optimal. The simple case that consists in considering the user satisfaction of everyone using the same linear function of the bandwidth seen leads to ideal efficiency, as every user has the same utility function. However, in the general case ideal efficiency is not achieved. According to the NP assumption, every network node implements a Fair Scheduler policy, so we can manage the tradeoff among the three main performance parameters: bandwidth, delay, and loss (see [65]). This tradeoff can not be made with the TCP-friendly paradigm, therefore our paradigm leads to a significantly higher efficiency, in the sense of the satisfaction of end users, than the TCP-friendly paradigm.

We have given the assumptions made and the properties enforced by the FS paradigm. The NP contains only the Fair Scheduler assumption. As this mechanism is of broad utility – we will show in section B.3.1 that a Fair Scheduler has a positive impact on TCP flows – it does not violate the end-to-end argument [71]. The issues related to the practical introduction of the paradigm are studied in section B.3.

The FS paradigm, like the TCP-friendly paradigm, applies for both unicast and multicast since the paradigm does not make any assumption on the transmission mode. Moreover, the FS paradigm enforces properties of great benefits for multicast flows (see section B.3.3).

In conclusion, we have defined a simple paradigm for end-to-end congestion control, called FS paradigm, that relies on a Fair Scheduler network and only makes the assumption that the end users are selfish and non-collaborative. We note that the FS paradigm is less restrictive than the TCP-friendly paradigm, as it does not make any assumptions on the mechanism used by the end users. Whereas the benefits of the FS paradigm with respect to flow isolation are commonly agreed on by the research community, its benefits for congestion control have been less clear since the congestion control properties are often not clearly defined. We showed that the FS paradigm allows to devise end-to-end congestion control protocols that meet almost all the properties of an ideal congestion control protocol. The remarkable point is that simply

using Fair Schedulers allows to devise end-to-end congestion control protocols tailored to the application needs, due to the great flexibility when devising the congestion control protocol and due to the tradeoff possible among the performance parameters, while being nearly ideal congestion control protocols.

In section B.3.3 we address how to devise a new congestion control protocol according to the FS paradigm.

B.3 Practical Aspects of the FS Paradigm

In the previous sections we defined the FS paradigm. Now we investigate the practical issues that come with the introduction of such a paradigm in the Internet.

B.3.1 Behavior of TCP with the FS Paradigm

In this section, we evaluate the impact of the NP assumption of the FS paradigm on the today's Internet. A central question if we want to deploy the FS paradigm in the today's Internet is: As the NP assumption requires modifications in the network nodes, how will the use of a Fair Scheduler affect the behavior and performance of TCP flows? Suter showed the benefits of a fair scheduler for TCP flows [83]. While his results are very promising, they are based on simulations for a very simple topology. We decided to explore the influence of the NP hypothesis on TCP with simulations on a large topology.

The generation of realistic network topologies is a subject of active research [23]. It is commonly agreed that hierarchical topologies better represent a real Internetwork than do flat topologies. We use `tiers` ([23]) to create hierarchical topologies consisting of three levels: WAN, MAN, and LAN that aim to model the structure of the Internet topology [23] and call this Random Topology *RT*.

We give a brief description of the topology used for all the simulations. The random topology *RT* is generated with `tiers` v1.1 using the command line parameters `tiers 1 20 9 5 2 1 3 1 1 1 1`. A WAN consists of 5 nodes and 6 links and connects 20 MANs, each consisting of 2 nodes and 2 links. To each MAN, 9 LANs are connected. Therefore, the core topology consists of $5 + 40 + 20 \cdot 9 = 225$ nodes. The capacity of WAN links is 155Mbit/s, the capacity of MAN links is 55Mbit/s, and the capacity of LAN links is 10Mbit/s. The WAN link delay is uniformly chosen in [100,150] ms, the MAN link delay is uniformly chosen in [20,40] ms, and the LAN link delay is 10 ms. Each LAN is represented as a single leaf node in the tiers topology. All the hosts connected to the same LAN are connected to the same leaf node and send their data on the same 10 Mbit/s link.

The Network Simulator ns [62] is commonly agreed to be the best simulator for the study

of Internet protocols. We use ns with the topology generated by tiers. We choose, for each simulation, either a small queue length (50 packets) or a large queue length (500 packets) for both FIFO and FQ scheduling, i.e. the FQ shared buffer is 50 or 500 packets large. The buffer management used with FIFO scheduling is drop tail and the buffer management used with FQ scheduling is longest queue drop with tail drop. The TCP flows are simulated using the ns implementation of TCP Reno, with a packet size of 1000 bytes and a receiver window of 5000 packets, large enough not to bias the simulations. The TCP sources have always a packet to send.

Our simulation scenarios are the following. We add from $k = 50$ to $k = 1600$ TCP flows randomly distributed on the topology RT, i.e. the source and the receiver of a flow are randomly distributed among the LANs. We do, for each configuration of the TCP flows, an experiment with FIFO scheduling and an experiment with FQ scheduling, for both with a queue size of 50 and 500 packets. These experiments show the impact of the NP assumption on unicast flows. All the simulations are repeated five times and the average is taken over the five repetitions. All the plots are with 95% confidence intervals. We choose a simulated time of 50 seconds, large enough to obtain significant results. All the TCP flows start randomly within the first simulated second. We compute the mean throughput $\bar{F}_i$ over the whole simulation for each TCP flows i, $i = 1, ..., k$.

We consider three measures to evaluate the results:

- the mean throughput $\bar{B} = \frac{1}{k} \sum_{i=1}^{i=k} \bar{F}_i$. $\bar{B}$ shows the efficiency of the scheduling discipline in the sense of the satisfaction of the users if we consider a utility function that is linearly dependent of the bandwidth seen for each receiver.

- the minimum throughput $\min_{i=1,...,k} F_i$ shows the worst case performance for any receiver. We say that an allocation is max-min fair if the smallest assigned bandwidth seen by a user is as large as possible and, subject to that constraint, the second-smallest assigned bandwidth is as large as possible, etc. (see [36]). So the minimum throughput shows which scheduling discipline leads to the bandwidth allocation closest to the max-min fair allocation.

- the standard deviation $\sigma = \sqrt{\frac{1}{k-1} \sum_{i=1}^{i=k} (\bar{F}_i - \bar{B})^2}$ gives an indication about the uniformity of the bandwidth distribution among the users.

The Fig. B.2 shows the mean throughput for all the receivers as the number of TCP flows increases, and table B.1 gives the loss rate for a 50 second and a 200 second long simulation with 1000 TCP flows in function of the scheduling policy and of the queue size. We first note, in Fig. B.2, that a larger queue size leads to a higher mean throughput. Indeed, as the buffer size increases the amount of time the bottleneck link is fully utilized increases too. Therefore,

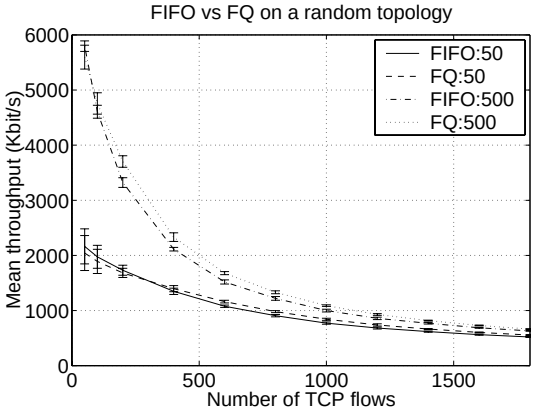


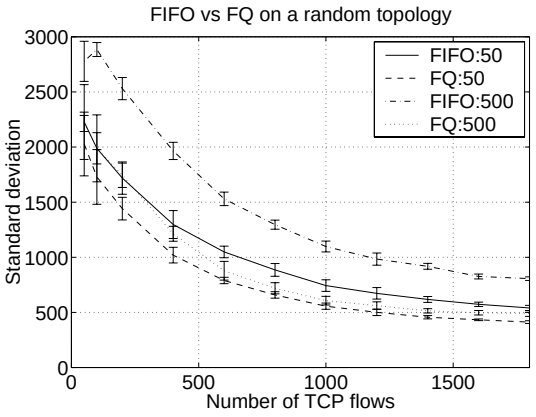
Figure B.2: FIFO versus FQ, mean throughput $\bar{B}$ for an increasing the number of unicast flows $k = 50, ..., 1600$ and for two size of queue length.

the mean throughput will increase. On the other hand, when we increase the buffer size, the amount of time required for a source to notice the congestion will increase(i.e. buffer overflow), resulting in an increase of the loss rate as shown in table B.1.

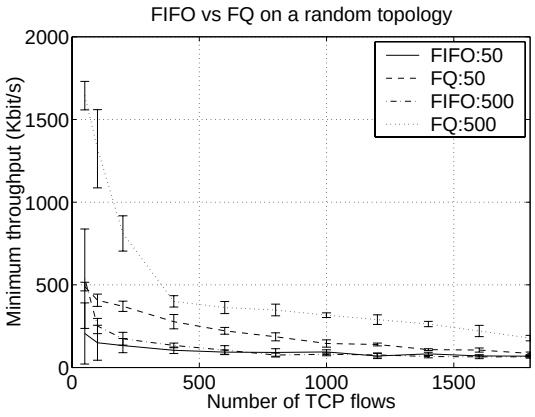
buffer size	Duration of the simulation			
	50 seconds		200 seconds	
	FIFO	FQ	FIFO	FQ
50 packets	1%	0.82%	0.35%	0.33%
500 packets	2.3%	1.8%	0.57%	0.52%

Table B.1: Loss rate for a 50 second and 200 second long simulation with 1000 TCP flows as a function of the queue size and the scheduling policy.

In all cases, we choose static scenarios, i.e. scenarios where all the TCP flows start at the beginning of the simulation and where there is no arriving nor departing flows. Our aim, with this kind of scenarios, is to avoid noise due to dynamic scenarios. At the beginning of a simulation, all the TCP sources must discover the available bandwidth. Therefore, there is a high probability that the bottleneck queues overflow during a slow start phase. However, the additive increase multiplicative decrease mechanism of TCP leads to an equilibrium. When a TCP flow reaches the equilibrium, the bottleneck queue overflows during a congestion avoidance phase. Therefore, the TCP source sees only one loss per TCP cycle. When the system comes close to the equilibrium, the TCP sources see bottleneck queues overflow during congestion avoidance phases. The mean loss rate decreases, as a bottleneck queue overflow, during a congestion avoidance phase, leads to only one loss whereas a bottleneck overflow, during a slow start phase, leads to a large number of losses.



(a) standard deviation σ of the mean throughput F_i.



(b) minimum throughput.

Figure B.3: FIFO versus FQ, increasing the number of unicast flows $k = 50, ..., 1600$ and for two size of queue length.

We see in table B.1 that for a longer simulation time (200 seconds versus 50 seconds) the difference in the loss rate between a queue size of 50 packets and of 500 packets becomes smaller. Indeed, the longer the simulated time is, the closer to the equilibrium the system is. For a system close to the equilibrium most of the bottleneck queues overflow during congestion avoidance phases, and the source detects the overflow with only one loss, independently of the queue size. The closer to the equilibrium the system is, the more independent to the queue size the loss rate is. The loss rate is a good indicator of the stability of the system.

In Fig. B.2, we see that the FQ scheduling leads to a higher mean throughput than FIFO scheduling. For instance, for 1000 TCP flows ($k = 1000$) the mean throughput $\bar{B}$ obtained with FQ scheduling is 9% higher than with FIFO scheduling for both small and large queue sizes. We see in table B.1 that the loss rate is lower with FQ scheduling than with FIFO scheduling. Since the loss rate is a good indicator of the stability of the system, FQ scheduling improves the stability of the system and, therefore, improves the speed of convergence of the TCP flows toward equilibrium. As TCP is the most efficient at the equilibrium, FQ scheduling leads to a higher throughput than FIFO scheduling. We note, on Fig. B.2, that for a small number of TCP flows, the mean throughput obtained with FIFO scheduling is higher than with FQ scheduling. However, as the confidence intervals largely overlap (the mean value of one measure in contained in the confidence interval of the other one), this result is not statistically significant.

FQ scheduling increases the stability of the system, improves the speed of convergence toward the equilibrium and the mean throughput of the TCP flows. Figures B.3 shows that FQ scheduling significantly improves fairness among the TCP flows. Indeed, Fig. B.3(a) shows

that FQ scheduling always leads to a lower standard deviation than FIFO scheduling, and the minimum throughput (see Fig. B.3(b)) is higher with FQ scheduling than with FIFO scheduling. Therefore, FQ scheduling leads to a fairness closer to max-min fairness than FIFO scheduling.

In conclusion, whereas the NP assumption requires changes in the network, which is a hard task, our simulations show that already the increase in TCP performances justifies the NP assumption.

B.3.2 Remarks on the Deployment of the New Paradigm

One practical question concerning the FS paradigm is its deployment in the Internet. First one can note that the issues concerning the deployment of the paradigm are only related to the deployment of the *Fair Scheduler* capability in the routers. The deployment of the end-to-end protocols is not an issue due to the NP assumption, since the paradigm enforces no constraint at the end system. For a new application, one can easily develop an end-to-end congestion control protocol for this application and distribute this protocol with the application. On the other hand, for existing applications, we can develop end-to-end congestion control protocols and so incrementally upgrade these applications without negative impact on the other applications. Indeed, the ones who use the new protocol will see a significant enhancement in the performance whereas the others, who do not upgrade yet, do not see a significant modification in their performance. So the FS paradigm allows for an easy deployment of the end-to-end protocols. This is not the case with the TCP-friendly paradigm, since it heavily relies on the collaboration of **all** the end users. If one wants, in the case of a collaborative paradigm, to add a new congestion control protocol, it has to implement the same mechanism than the previous congestion control protocols. If one wants to change this mechanism, one has to change it in every end user, which is practically infeasible.

Second, the deployment of the NP requires that every router implements a Fair Scheduler. If we deploy an end-to-end protocol without the NP assumption, we can cause congestion collapse. Deploying the NP in the Internet seems unrealistic. However we have to take into account the administrative reality of the Internet. The Internet is an interconnection of ISPs. Each ISP has the full control of its network and offers specific services on its network, independent of the rest of the Internet. For instance, some ISPs start providing the multicast functionality inside their network whereas Internet, as a whole, is still not multicast capable[2]. ISPs are operating in a competitive environment that forces them to innovate and improve their service offered to keep the customers. In the past, ISPs have continuously upgraded the capacity of their links and installed, for instance, caches to improve their service. If an ISP has installed caches, his

[2]We can note similarities in the deployment of the multicast functionality per ISP and the deployment of the FS paradigm per ISP as both require that all the routers support the respective capability.

client will find with a probability P (as P ranges between 0.5 and 0.7 according to [76]) the Web documents they access in the ISP's cache. Upgrading all the routers within an ISP with a *Fair Scheduler* will give a number of immediate benefits. Customers surfing on the Web will have a higher TCP performance (around 10% higher, see section B.3.1) and therefore shorter download times (with a probability P) whenever a document is in the cache or on a server directly connected to the same ISP. If the ISP is also multicast capable, its clients can also use new end-to-end protocols that significantly improve the performance of the multicast connection, like PLM [50]. The deployment of the FS paradigm will be very easy for a new ISP who has no existing "legacy infrastructures".

In conclusion, the deployment of the new paradigm can be incremental. For an ISP, upgrading all its routers with Fair Schedulers is a substantial investment, but we believe that this investment will improve the quality of the service, which can be a significant commercial argument. So the ISPs have a financial interest in the deployment of this paradigm.

B.3.3 PLM: A Pragmatic Validation of the FS Paradigm

In this section we explain how to apply the FS paradigm for the design of a new congestion control protocol through an example: PLM. We just give an overview of the PLM protocol, for details the reader is referred to [50]. The ESP part of the FS paradigm says that the assumption of selfish and non-collaborative end users is a sufficient but not a necessary condition. Therefore, when devising a new congestion control protocol with the FS paradigm, we just address the application needs, and we do not have to take care about the properties required for a congestion control protocol. These properties will automatically be enforced by the paradigm. For instance, we do not have to care explicitly about fairness, we just have to find a mechanism the satisfies the users. This fact considerably simplifies the design of new congestion control protocols.

Unlike the TCP-friendly paradigm, the FS paradigm allows to make a separation between the properties required by the designer for a congestion control protocol and the requirement of the users. We note that the properties required by the designer and the requirements of the users may overlap.

We introduced a new paradigm that, in theory, considerably simply the design of new congestion control protocols. To validate our claim, we apply the FS paradigm for the design of a new cumulative layered multicast congestion control protocol. We showed in [49] that the two most popular cumulative layered multicast congestion control protocols RLM [55] and RLC [87] suffer from pathological behaviors. Our conclusion was that the design of a cumulative layered multicast congestion control protocol with the TCP-friendly paradigm is very complex. In fact, most of the problems in RLM and RLC come from the bandwidth inference mechanism that must guarantee properties like efficiency, stability, and fairness. The bandwidth inference

mechanism is based on congestion signal, such as loss or an ECN [29]. However, congestion signals have many weaknesses: the bottleneck queue must overflow; the congestion signal, for instance a gap in the sequence number of the packets, is received far after congestion has started; the congestion signal does not give information on the available bandwidth.

The Packet Pair (PP) bandwidth inference mechanism [44], introduced by Keshav, allows to obtain an explicit available bandwidth notification. Indeed, Keshav showed that when one sends a PP, i.e. two packets sent as fast as possible (back-to-back) into a network where every router is a Fair Queuing router, the packets of the PP will be spaced out at the receiver by the available bandwidth on the path between the source and the receiver. The PP bandwidth inference mechanism is simple and does not have the drawbacks of the bandwidth inference mechanisms based on congestion signals.

We decided to devise an new cumulative layered multicast congestion control protocol, called PLM, based on the PP mechanism. We do not use any complex filtering mechanism. At the receiver, we simply collect the PP estimates of the available bandwidth and add or drop layers according to these estimates (for more details see [50]). We do not add any mechanism to improve stability or fairness.

Our evaluation of the PLM protocol showed that PLM is a nearly ideal congestion control protocol. PLM is stable, the receivers converge fast to the available bandwidth and do not suffer from pathological oscillations. PLM is efficient, PLM converges fast to the available bandwidth and tracks this available bandwidth with no loss induced, even in a self similar and multifractal environment. PLM is fair with the other PLM sessions and with TCP. PLM is robust against misbehaving sources. PLM is scalable due to the cumulative layered architecture. PLM is feasible, it is a very simple protocol that is easy to evaluate. Moreover, PLM was introduced in the ns [62] distribution and can easily be evaluated. PLM outperforms all the previous cumulative layered multicast congestion control protocols like RLM and RLC.

In summary, the FS paradigm makes it very easy to devise PLM, a nearly ideal congestion control protocol. PLM is clearly a pragmatic validation of the FS paradigm.

B.4 The FS Paradigm versus the TCP-friendly Paradigm

TCP, which has been for many years the main congestion control protocol, has indisputably contributed to the stability and the efficiency of the Internet. However, every new congestion control protocol deployed in the Internet must be TCP-friendly.

Both the TCP-friendly and the FS paradigm allow to devise end-to-end congestion control protocols compatible with TCP. A paradigm is only a formal way to define how to devise congestion control protocols. To compare two paradigms we must look at the properties of the protocols devised with these paradigms. We compare the congestion control protocols accord-

ing to the properties of an ideal congestion control protocol. The results are summarized in table B.2 where a $+$ shows which paradigm outperforms the other one for a given property.

Properties	FS paradigm	TCP-friendly paradigm
Stability	$+$	$-$
Efficiency	$+$	$-$
Fairness	$+$	$-$
Robustness	$+$	$-$
Scalability	$+$	$+$
Feasibility	$-$	$+$

Table B.2: The FS paradigm versus the TCP-friendly paradigm.

The TCP-friendly paradigm does not lead to ideal stability neither efficiency, due to the lack of an assumption on the scheduling discipline (with selfish users only a Fair Scheduling can lead to ideal stability and in some case to ideal efficiency [79]). The FS paradigm does not lead to ideal efficiency in the general case either. However, the FS paradigm allows a tradeoff among the performance parameters bandwidth, delay, and loss which is impossible with the TCP friendly paradigm. The TCP-friendly paradigm does not lead to ideal fairness, the fairness of this paradigm is biased by the RTT. The weakest point of the TCP-friendly paradigm is its lack of robustness: As this paradigm relies on the collaboration of the end users, it is easy to grab the bandwidth from the TCP-friendly flows. Both the TCP-friendly paradigm and the FS paradigm are scalable.

The weakest point of the FS paradigm is feasibility. The TCP-friendly paradigm is the most feasible paradigm because it does not require any modification in the current Internet. The FS paradigm requires modification of the scheduling inside routers. We showed in section B.3.2 that this deployment is feasible per ISP and that ISPs have a financial interest in this deployment. We believe that the FS paradigm is an appealing solution. In particular, the FS paradigm shows that with a reasonable network support, we can considerably simplify the design of new congestion control protocols, whereas the design of new congestion control protocols with the TCP-friendly paradigm is one of the most complex problem in networking.

B.5 Related Work

There is surprisingly little literature on congestion control paradigms. Most of the studies are about how to devise TCP-friendly end-to-end congestion control schemes. See [33] and [30] for unicast congestion control, and see [55], and [73] for multicast congestion control.

Keshav [44] presents a comprehensive study of congestion control. While we agree with him in many points, our approach to the problem is fundamentally different. Keshav's aim was to study the problems of congestion control and to present as a solution a new unicast congestion control scheme. Our aim is to define a model (*a new paradigm*) to devise end-to-end congestion control schemes. To achieve this goal, we define a set of properties for congestion control schemes. The definitions are abstract (they do not take into account any mechanism) and use a mathematical foundation. This formalism allows us to prove the feasibility of the FS paradigm (see section B.2.3) and to define a general background for the study of end-to-end congestion control.

Shenker applies game theory to study congestion control [79] and is complementary to ours. He shows that one can achieve, with the selfish and non-collaborative behavior of the users, a congestion control that has a set of desired good properties. The only requirement is to have switching with a *fair share allocation function*. Shenker shows the benefits of the fair share policy for congestion control. However, he does not clearly identify the properties of an ideal congestion control protocol and does not define the paradigm for devising congestion control protocols. We formally define the problem of congestion control and propose a paradigm for congestion control. Shenker presents mathematical results that validate our work.

Lefelhocz *et al.* discuss a new paradigm for best effort congestion control [46] and provide a good discussion of the question: "Why do we need a new paradigm?" The solution proposed is a set of four mechanisms required for congestion control: scheduling, buffer management, feedback, and end adjustment. These mechanisms meet the FS paradigm: the scheduling and the buffer management are part of our NP; the feedback and the end adjustment are part of the end-to-end protocol. Our study shows why these mechanisms are sufficient. Moreover, we show that selfish and non-collaborative end users can achieve nearly ideal congestion control. In their study, Lefelhocz *et al.* explain why they *believe* the four mechanisms are necessary and sufficient, we develop the formalism needed to *show* why they are necessary and sufficient. Our results can be seen as a generalization of their study.

Another way to devise a new paradigm is the Diffserv or Intserv paradigm. There is active research on these topics, but to the best of our knowledge, there is no similar study to ours with these paradigms. Moreover the Diffserv and Intserv paradigms lead to much more complex mechanisms than the FS paradigm, for instance these paradigms are not viable without pricing (see [13]). We believe that, even in a network with quality of service, a best effort class will always be popular and useful. The FS paradigm is a paradigm for best effort networks and, in particular, it applies to a best effort class.

B.6 Conclusion

We defined a new paradigm, called FS paradigm, for end-to-end congestion control protocols. This paradigm relies on a Fair Scheduler network and makes the assumption that the end users are selfish and non-collaborative. Whereas the FS paradigm is commonly agreed to have interesting properties, the research community has no clear understanding of what these properties precisely are. This lack of formalism leads to a mistrust toward this paradigm, which explains why end-to-end congestion control protocols have not been studied with the FS paradigm.

We start the paper with a definition of the notion of congestion and formally define a set of six properties for an ideal congestion control protocol. These properties are based on notions of game theory and microeconomics, thus allowing the use of the formally proven results previously established using these theories. The rigorous definition of the properties is important since this definition is highly reusable (we only make the assumption of selfishness for the definitions) and this definition allows to rigorously prove the validity of the FS paradigm. Then, we define the FS paradigm. We show that this new paradigm allows to devise congestion control protocols that have almost all the properties of an ideal congestion control protocol. The main strength of the FS paradigm is *the separation between the properties required by the designer of the protocol and the requirements of the end system*. There is no restriction on the end system when devising a new congestion control protocol, and the FS paradigm guarantees almost all the properties of an ideal congestion control protocol. To the best of our knowledge we are the first that define the properties of an ideal congestion control protocol, define a paradigm for the design of end-to-end congestion control protocols with such a formalism, and show the validity of this paradigm, in the sense of the properties of an ideal congestion control protocol.

The second part of our study is about the practical aspects related with the introduction of the FS paradigm in the Internet. Our simulations on a large topology show the great benefits of the Fair Scheduler policy for TCP flows. The Fair Scheduler policy improves the stability of a system of TCP flows and increases the mean throughput of the TCP flows by roughly 10% compared to the FIFO scheduling policy. As indicated, the incremental deployment by a single ISP will yield immediate benefits to the ISP's clients. In conclusion, the FS paradigm, applied in the today's Internet, leads immediately to great benefits for the TCP flows and opens a new way in devising very efficient unicast and multicast end-to-end congestion control protocols. The FS paradigm offers an appealing alternative to the TCP-friendly paradigm. Finally, we showed how to apply the FS paradigm to the design of a new congestion control protocol. We devised, according to the FS paradigm, a new cumulative layered multicast congestion control protocol based on the packet pair mechanism. This protocol, called PLM [50], outperforms all the previous cumulative layered multicast congestion control protocols, and it verifies the properties of an ideal congestion control protocol, as predicted by the FS paradigm, whereas

we do not address any of these properties in the design of the protocol. PLM is a pragmatic validation of the FS paradigm.

Appendix C

PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes

Abstract

A major challenge in the Internet is to deliver live audio/video content with a good quality and to transfer files to a large number of heterogeneous receivers. Multicast and cumulative layered transmission are two mechanisms of interest to accomplish this task efficiently. However, protocols using these mechanisms suffer from slow convergence time, lack of inter-protocol fairness or TCP-fairness, and loss induced by the join experiments.

In this paper we define and investigate the properties of a new multicast congestion control protocol (called PLM) for audio/video and file transfer applications based on a cumulative layered multicast transmission. A fundamental contribution of this paper is the introduction and evaluation of a *new* and *efficient* technique based on packet pair to infer which layers to join. We evaluated PLM for a large variety of scenarios and show that it converges fast to the optimal link utilization, induces no loss to track the available bandwidth, has inter-protocol fairness and TCP-fairness, and scales with the number of receivers and the number of sessions. Moreover, all these properties hold in self similar and multifractal environment.

Keywords: Congestion Control, Multicast, Capacity inference, Cumulative layers, Packet Pair, FS-paradigm.

C.1 Introduction

Multimedia applications (audio and video) take a growing place in the Internet. If multiple users want to receive the same audio/video data at the same time, multicast distribution is the most efficient way of transmission. To accommodate heterogeneity, one can use a layered source coding where each layer is sent to a different multicast address and the receivers subscribe to

as many layers as their bottleneck bandwidth permits. The multimedia applications can easily be transmitted using cumulative layers: each higher layer contains a refinement of the signal transmitted in the lower layers. File transfer to a large number of receivers will probably become an important application for software updates or electronic newspaper posting. Multicast distribution with a cumulative layer coding based on FEC (see [86]) is an efficient solution to this problem.

A receiver-driven cumulative layered multicast congestion control protocol (RLM) was first introduced by Steven McCanne [55] for video transmission over the Internet. RLM has several benefits: First, the cumulative layered transmission uses a natural striping of the multimedia streams and achieves a very efficient use of the bandwidth as the different layers do not contain redundant information but refinements. Second, the receiver-driven approach allows each receiver to obtain as much bandwidth as the path between the source and this receiver allows. However, RLM has also some fundamental weaknesses. RLM is not fair (neither inter-RLM fair nor TCP-fair), RLM converges slowly to the optimal rate and tracks this optimal rate slowly (after a long equilibrium period, RLM can take several minutes to do a join experiment and so to discover bandwidth that became recently available), finally RLM induces losses.

A TCP-friendly version of a cumulative layered receiver-driven congestion control protocol was introduced by Vicisano [87]. Whereas this protocol solves some fairness issues it does not solve issues related to the convergence time (the subscription to the higher layers is longer than the subscription to the lower layers), and does not solve the issues related to the losses induced.

We want a congestion control protocol for multimedia and file transfer applications that guarantees fast convergence, high throughput and does not induce losses. We introduce in [48] a *paradigm* to devise end-to-end congestion control protocols only by taking into account the requirements of the application (congestion control protocols tailor-made to the application needs). Our paradigm is based on the assumption of a Fair Scheduler network i.e. a network where every router implements a PGPS-like [65] scheduling with longest queue drop buffer management. We show that this assumption is practically feasible. Moreover this paradigm only assumes selfish and non-collaborative end users, and guarantees under these assumptions nearly ideal congestion control protocols.

To practically validate the theoretical claims of our paradigm, we devise a new multicast congestion control protocol for multimedia (audio and video) and file transfer applications. We devise a receiver-driven cumulative layered multicast congestion control protocol that converges fast to the optimal rate and tracks this optimal rate without inducing any loss. The cornerstone of our congestion control protocol is the use of packet pair (PP) to discover the available bandwidth (see [44]). We call the protocol *packet Pair receiver-driven cumulative Layered Multicast* (PLM).

In section C.2 we introduce the FS-paradigm. Section C.3 presents the PLM protocol. We

evaluate PLM in simple environments to understand its major features in section C.4 and in a realistic environment in section C.5. Section C.6 explores the practical validation of the theoretical claims of the FS-paradigm, section C.7 presents the related work, and we conclude the paper with section C.8.

C.2 The FS Paradigm and Its Application

A paradigm for congestion control is a model used to devise new congestion control protocols. A paradigm makes assumptions and under these assumptions we can devise compatible congestion control protocols; compatible means that the protocols have the same set of properties. Therefore, to define a new paradigm, we must clearly express the *assumptions* made and the *properties* enforced by the paradigm.

In the context of a formal study of the congestion control problem as a whole, we defined the Fair Scheduler (FS) paradigm (see [48]). We define a Fair Scheduler to be a Packet Generalized Processor Sharing scheduler with longest queue drop buffer management(see [65], [82], [20], and [4] for some examples). For clarity, we make a distinction between the assumption that involves the network support – we call this the *Network Part* of the paradigm (NP) – and the assumptions that involve the end systems – we call this the *End System Part* of the paradigm (ESP).

The assumptions required for the FS paradigm are:

- For the NP of the paradigm we assume a *Fair Scheduler* network, i.e. a network where every router implements a Fair Scheduler.
- For the ESP, the end users are assumed to be selfish and non-collaborative.

The strength of this paradigm is that under these assumptions we can devise nearly ideal end-to-end congestion control protocols (in particular fair with TCP), i.e. different protocols that have the following set of properties: stability, efficiency, fairness, robustness, scalability, and feasibility. The main constraint of the FS-paradigm is the deployment of FS routers. However, we explained in [48] how and why this deployment is feasible per ISP. The only assumption that the paradigm makes on the end-user is its selfish and non-collaborative behavior (we do not require these properties, we just do not need anything else to achieve the properties of an ideal congestion control protocol).

We consider for the PLM congestion control protocol multimedia (audio and video) and file transfer applications. The requirements of multimedia applications are very specific. We must identify how to increase the satisfaction of a user of a multimedia application: (i) A user wants to receive the highest quality (high throughput, low number of losses) and (ii) wants to avoid frequent modifications in the quality perceived. The requirement of a file transfer application is

a small transfer time (high throughput, low loss rate). In the next section we define mechanisms that allow to meet these requirements. We devise the PLM protocol with the FS-paradigm. We assume a Fair Scheduler network and all the mechanisms at the end-system try to maximize the satisfaction of the users (selfish behavior). What is remarkable with this paradigm is that whereas the end-users are selfish, we achieve the properties of an ideal end-to-end congestion control protocol.

To understand why the FS-paradigm is of great benefit to devise congestion control protocols we take a simple example (examples specific to PLM are presented in section C.6). First we have to identify the requirements of a user (i.e. how to increase his satisfaction). For our purpose we suppose that the user wants to converge fast to an optimal rate and to be stable at this optimal rate. The FS-paradigm guarantees that even a *simple* congestion control algorithm will converge and be stable at this optimal rate. This is the cornerstone of the practical application of the FS-paradigm. We do not have to devise complicated congestion control protocols to converge to the optimal rate and to stay at this optimal rate. Of course, the FS-paradigm does not give this simple algorithm, but if one finds a simple algorithm that converges to the optimal rate, this algorithm leads to a congestion control protocol that will converge fast and will be stable.

PLM is a demonstration of the practical application of the FS-paradigm. We have a simple mechanism, Packet Pair, and do not introduce any complicated mechanism to improve the convergence nor the stability. We discuss in section C.6 some implications of the FS-paradigm on the design of PLM.

C.3 Packet Pair Receiver-Driven Layered Multicast (PLM)

Our protocol PLM is based on a cumulative layered scheme and on the use of packet pair to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. PLM assumes that the routers are multicast capable but does not make any assumption on the multicast routing protocol used. PLM is receiver driven, so all the burden of the congestion control mechanism is at the receivers side. The only assumption we make on the sender is the ability to send data via cumulative layers and to emit for each layer packets in pairs (two packets are sent back-to-back). We devise PLM with the FS-paradigm, in particular we assume a Fair Scheduler network. In the next two sections we define the two basic mechanisms of PLM: The receiver-driven cumulative layered multicast principle and the packet pair mechanism.

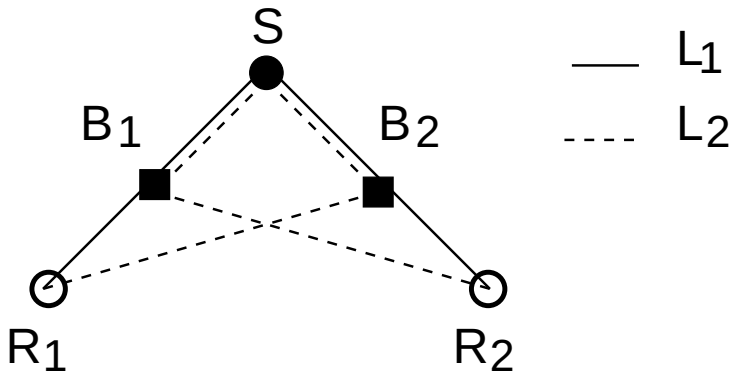


Figure C.1: Example of two layers following two different multicast trees.

C.3.1 Introduction to the Receiver-Driven Cumulative Layered Multicast Principle

Coding and striping multimedia data onto a set of n cumulative layers $L_1, \cdots, L_n$ simply means that each subset $\{L_1, \cdots, L_i\}_{i \leq n}$ has the same content but with an increase in the quality as i increases. This kind of coding is well suited for audio or video applications. For instance, a video codec can encode the signal in a base layer and several enhancement layers. In this case, each subset $\{L_1, \cdots, L_i\}$ has the same content and the higher number of layers we have, the higher quality video signal we obtain. For audio and video applications, the cumulative layered organization is highly dependent of the codec used. Vicisano in [86] studies two cumulative layered organizations of data, based on FEC, for file transfer. In this case the increase in the quality perceived is related to the transfer time.

Once we have a cumulative layer organization it is easy for a source to send each layer on a different multicast group. In the following, we use indifferently the terminology *multicast group* and *layer* for a multicast group that carries a single layer. To reap full benefits of the cumulative layered multicast approach for congestion control, a receiver-driven congestion control protocol is needed. When congestion control is receiver-driven, it is up to the receivers to add and drop layers (i.e. to join and leave multicast group) according to the congestion seen. The source has only a passive role, consisting in sending data in multiple layers. Such a receiver-driven approach is highly scalable and does not need any kind of feedback, consequently solves the feedback implosion problem.

One fundamental requirement with cumulative layered congestion control is that all the layers must follow the same multicast routing tree. In Fig. C.1 we have one multicast source and two receivers. The source sends data on two layers, each layer following a different multicast tree. Imagine congestion at the bottleneck B_1, receiver R_1 will infer that it should reduce its number of layers. As we use cumulative layers we can only drop the highest layer: L_2. However this layer drop will not reduce congestion at bottleneck B_1. When the layers do not follow the

same multicast routing tree, the receivers can not react properly to congestion.

One of the weakness of the cumulative layered congestion control protocols is the layer granularity. In fact this granularity is not a weakness for audio and video applications. Indeed, it makes no sense to adjust a rate with a granularity of, for instance, 10 Kbyte/s, if this adjustment does not improve the satisfaction of the users. Moreover a user may not perceive fine-grain quality adjustments. We strongly believe that a standardization effort should be made on the characteristics of the perceived quality compared to the bandwidth used. These characteristics are codec dependent. Imagine, for the purpose of illustration, the following classification for audio broadcast: quality 1: 10 Kbit/s (GSM quality); quality 2: 32 Kbit/s (LW radio quality); quality 3: 64 Kbit/s (quality 2 stereo); quality 4: 128 Kbit/s (FM radio quality); quality 5: 256 Kbit/s (quality 4 stereo). It is clear, in this example, that there is no benefit in creating an intermediate layer as this layer will not create a significant modification in the perceived quality. If a user does not have the minimum bandwidth required, he can not connect to the session. Who can have satisfaction in listening a sonata of J.S. Bach with a lower quality than GSM quality? Therefore, we do not believe that the layer granularity is a weakness for congestion control for audio and video applications.

For file transfer applications, the layer granularity leads to a higher transfer time (dependent on the layer distribution) than rate/window based solutions in case of small homogeneous groups. However, a sender rate/window based multicast congestion control protocol must adapt to the slowest receiver. In case of high heterogeneity of bandwidth, the layered scheme is clearly the most efficient. It is not the purpose of this paper to study how much bandwidth should be given to each layer.

All the previous multicast layered congestion control schemes are based on CBR layers. The protocols like RLM heavily rely on the accurate knowledge of the throughput of each layer to infer the available bandwidth. If a layer is added whereas this layer uses less than its regular throughput, the join experiment becomes meaningless. In the case of PLM, as the bandwidth inference is not based on join experiments but on PP estimate, even if a layer uses less than its regular throughput, the bandwidth inference is still accurate. Therefore, PLM can accommodate VBR layers if we can define an upper bound of the bandwidth reaches by each layer. PLM will simply assume that each layer is CBR with a bandwidth defined by the upper bound of the VBR layers. However, this solution can be very inefficient in case of VBR layers with a large standard deviation and a small mean throughput. One solution is to have a protocol that dynamically adapt its layers to the VBR layers. Managing dynamic layers is very complex and is an area for future research. In fact, we do not see any strong argument in favor of VBR encoding compared to CBR encoding. Even if CBR encoding can result in a slight decrease in quality, the ease of exploitation of CBR layers is a strong argument in favor of CBR codec. The study of a codec to get the appropriate layer distribution is beyond the scope of this paper.

In the following, we simply assume that we have a given set of CBR layers, without making any assumptions on the layer granularity.

C.3.2 Receiver-Driven Packet Pair Bandwidth Inference

The packet pair (PP) mechanism was first introduced by Keshav [44] to allow a *source* to infer the available bandwidth. We define a *receiver-driven* version of packet pair. Let the *bottleneck bandwidth* be the bandwidth of the slowest link on the path between the source and a receiver. Let the *available bandwidth* be the maximum bandwidth a flow can obtain. We assume a network where every router implements a Fair Scheduler. If a source sends two packets back to back (i.e. a packet pair), the receiver can infer the available bandwidth for that flow from the spacing of the packet pair and the packet size. By periodically sending packet pairs, the receiver can track the available bandwidth. The main feature of the PP bandwidth inference mechanism, unlike TCP, is that it *does not require losses*. Indeed, the bandwidth inference mechanism is based on measuring the spacing of the PPs and not on measuring loss.

For the packet pair bandwidth inference mechanism to succeed, the Fair Scheduler must be a fine approximation of the fluid Generalized Processor Sharing (GPS). Bennet and Zhang show that the Packet Generalized Processor Sharing (PGPS) is not a fine enough approximation of the GPS system for the packet pair mechanism to succeed. However, they propose a new packet approximation algorithm called WF^2Q that perfectly suits the packet pair bandwidth inference mechanism (see [3] for a discussion on the impact of the packet approximation of GPS system and for the details of the WF^2Q algorithm). In the following, we assume an algorithm for the Fair Scheduler that is a fine approximation (in the sense of the packet pair mechanism) of the GPS system like the WF^2Q algorithm.

The great interest of a receiver-based version of packet pair is twofold. First, we have considerably less noise in the measurement (see [67]). In the sender-based version, the packet pair generates two acknowledgments at the receiver and it is the spacing of these Acks that is evaluated at the sender to derive the available bandwidth. However, if we have bottleneck on the back-channel, the Acks will be spaced by the back-channel bottleneck and not by the data channel bottleneck. Second, the receiver can detect congestion before the bottleneck queue starts to build and far before the bottleneck queue overflows. A signal of congestion is a packet pair estimate[1] of the available bandwidth lower than the current source throughput. In the simplest case where an estimate is given by a PP measurement, the first PP that leaves the queue after congestion occurs is a signal of this congestion. The delay between the congestion event at the bottleneck and the receiver action (to this congestion) is the delay for the PP to go

[1] The appropriate estimator must be defined in the congestion control protocol. We define the (simple) estimator for PLM in section C.3.3.

from the bottleneck to the receiver (roughly the propagation delay from the bottleneck to the receiver). The PP bandwidth inference mechanism does not need losses to discover the available bandwidth and its receiver-driven version allows a receiver to react to congestion before the bottleneck queue overflows. We say that the receiver driven PP bandwidth inference mechanism *does not induce losses* when discovering the available bandwidth. An original consequence (unlike all the congestion control protocols that consider losses as signals of congestion) is that PLM can work without modification and no loss of performance on a lossy medium like a wireless link.

It is commonly argued that PP is very sensitive to network conditions. We identify two major components that can adversely impact PP. First, the physical network characteristics (load balancing, MAC layer, etc.). Second, the traffic pattern (PP estimate in a self similar and multifractal environment).

The physical network characteristics can indeed adversely impact PP. However, they can aversely impact all the congestion control protocols. For instance, load balancing on a packet basis clearly renders the PP bandwidth inference mechanisms meaningless but the TCP bandwidth inference mechanisms as well. How can you estimate the RTT if one can not assume that all the packets take the same path (or at least if one can not identify which packet takes which path). Most of the physical network noise can be filtered with appropriate estimators (see [44]). We leave this question as a future research.

The traffic pattern *does not adversely impact* PP measurements. A PP leaving the bottleneck queue will be spaced by the available bandwidth for the relevant flow. As real traffic in the Internet is self similar and multifractal [27], the PP estimate of the available bandwidth will highly fluctuate. The oscillations can be misinterpreted as instability of the PP estimation method. In fact, as the background traffic is highly variable, it is natural that the available bandwidth at the bottleneck is highly variable. The oscillations in the available bandwidth estimation are not due to instability but to the high efficiency of the PP method. It is the task of the congestion control protocol to filter the PP estimates in order to react with a reasonable latency (i.e. the congestion control protocol must not overreact to PP estimates).

C.3.3 PLM Protocol

We assume the source sends via cumulative layers and emits the packets as packet pairs on each of the layers, i.e. *all the packets on all the layers are sent in pairs* (we thus maximize the number of estimates). Moreover, we assume that the set of layers of a same session is considered as a *single* flow at the level of a Fair Scheduler. Now we describe the basic mechanisms of PLM that takes place at the receiver side. When a receiver just joined a session, it needs to know the bandwidth used by each layer. How to obtain this information is not the purpose of this paper. However, a simple way that avoids (source) implosion is to consider a multicast announcement

session where all the sources send informations about their streams (for instance the name of the movie, the summary, etc.) and in particular the layer distribution used. A receiver who wants to join a session, first joins the session announcement and then joins the session. In the following, we assume that the receivers who want to join the session know the bandwidth distribution of the layers.

Let PP_t be the bandwidth inferred with the packet pair received at time t, and let B_n be the current bandwidth obtained with n cumulative layers: $B_n = \sum_{i=1}^{n} L_i$ where layer i carries data at a bandwidth L_i. Let $\hat{B}_e$ be the estimate of the available bandwidth.

At the beginning of the session, the receiver just joins the base layer and waits for its first packet pair. If after a predefined timeout the receiver does not receive any packet we infer that the receiver does not have enough bandwidth available to receive the base layer, and therefore can not join the session. At the reception of the first packet pair, at time t, the receiver sets the check-timer $T_c := t + C$, where C is the check value (we find in our simulations that a check value C of 1 second is a very good compromise between stability and fast convergence). We use the terminology check (for both T_c and C) because when T_c expires after a period of C seconds the receiver checks whether he must add or drop layers. When the receiver sees a packet pair a time t_i:

- `if` $PP_{t_i} < B_n$ `then /*drop layers*/`
  - $T_c := t_i + C$
  - `until` $B_n < PP_{t_i}$ `do` (1)
    - drop layer n
    - n `:=` $n - 1$
- `elseif` $PP_{t_i} \geq B_n$ `and` $T_c < t_i$ `/*have received PPs during at least` C `units of time*/` `then /*add layers*/`
  - $\hat{B}_e := \min_{T_c - C < t \leq t_i} PP_t$ `/*take the minimum bandwidth estimate*/`
  - $T_c := t_i + C$
  - `if` $\hat{B}_e \geq B_n$ `then while` $B_{n+1} < \hat{B}_e$ `do` (2)
    - add layer $n + 1$
    - n `:=` $n + 1$

In summary, we drop a layer each time we have a sample PP_t value lower than the current layer subscription B_n, but we add layers according to the minimum PP_t value received during

a period C (if all the samples are $PP_t \geq B_n$ during this period). The algorithm is conservative by using strict inequalities ($<$) in (1) and (2). We can consider a less conservative version of the same algorithm by using ($\leq$) in (1) and (2). For all the simulations we take the conservative version with strict inequalities.

In case of high congestion (with a FS network, high congestion can only happen when a large number of new flows appear in a period of less than C seconds) packet pairs could never reach the receiver (one pathological case is when the second packet in a PP is always lost). So we need a mechanism to reduce congestion enough to receive a packet pair. If after a predefined timeout period we do not receive any packet, we drop a layer; if the layer dropped is the lowest, we exit the session. We identify losses using the packets sequence number, and use a one bit field that marks the first packet of a PP burst (the other packets of the burst must have the following sequence numbers). So each packet contains a layer id, a sequence number, and a one bit field. If we receive some packets, however no PP, we evaluate the loss rate with a short term estimator and drop a layer if this estimator exceeds a predefined threshold (we fix empirically the threshold to 10% of loss). After dropping a layer we wait for a period called a blind period (we fix empirically the blind period to 500 ms) before re-estimating the loss rate. This blind period helps to not over react to loss. We call that the aim of PLM is to allow a PP to reach the receiver in order to obtain an accurate estimate of the available bandwidth rather than dropping layer on loss.

Unlike RLM (see [49]) that produces losses with join experiments and unlike Vicisano's TCP-friendly protocol RLC (see [49]) that produces losses at join attempts (periodic bursts), PLM has the fundamental property that it does not induce any loss to discover the available bandwidth.

When a receiver joins a layer, all the other receivers downstream of the same bottleneck must join the same layer otherwise there is link underutilization (other benefits of the join synchronization are demonstrated in [77]). When a receiver drops a layer due to congestion, all the other receivers downstream of the same bottleneck must drop the layer otherwise the congestion persists. PLM achieves both, join and drop synchronization. Drop synchronization is obvious as the signal of drop (without high congestion) is a PP: Every receiver downstream of the same bottleneck will receive the same PP at the same time (roughly the same time according to the distance between the receivers and the bottleneck) and therefore will decide to drop layers at the same time. The join synchronization is less obvious at first sight. If the receivers do not start at the same time (the more probable situation) their timer T_c will not be synchronized and therefore they will join layers at different times. This is not a drawback as late joiners can not be synchronized before reaching the optimal rate. However, at the first drop inferred (due to $PP_t < B_n$) all the check-timers will be resynchronized and so all the following joins. This re-synchronization avoids the problems due to clock drift as well.

One problem for PLM can be the slow IGMP response in case of drop. IGMP can take several seconds before ”pruning” a group. This is a problem related to IGMP (and not to PLM). Rizzo in [74] introduces predictive techniques for fast IGMP leave. Moreover the leave synchronization of PLM tends to attenuate the drawback of the large leave delay in IGMP.

We have defined a multicast congestion control protocol that achieves and tracks the available bandwidth without loss induced to discover the available bandwidth. Moreover, a significant contribution of PLM is the introduction of a simple and efficient technique (based on PP) to infer which layer to join. As PLM is devised in the context of the FS-paradigm, all the properties of an ideal congestion control protocol – stability, efficiency, fairness, robustness, scalability, and feasibility – are theoretically guaranteed by the FS paradigm. We check via simulations if all these appealing properties really hold.

C.4 Initial Simulations

The initial set of simulations does not aim to evaluate PLM on realistic Internet scenarios but rather provides insights on how PLM behaves for specific and relevant configurations.

C.4.1 Evaluation Criteria

Three PLM parameters influence the behavior of PLM.

- The granularity of the layers: As PLM infers the bandwidth available and tracks the appropriate layers with a PP estimate, the less bandwidth per layer there is, the less stable the layer subscription will be. Moreover, an inaccurate estimate leads to a modification of the layer subscription, small layers emphasize this behavior.

- The check value C: A larger C leads to a higher stability as we can add layers only once each C interval, but leads to a lower speed of convergence.

- The burst size: In section C.3.3 we only consider PP, packet bursts of *size two*. However, PLM is not limited to a burst of size two. Increasing the burst size increases the accuracy of the estimate, but increases the probability of loss due to the bursty nature of each layers.

The behavior of PLM can be influenced by other flows (PLM sessions, TCP flows, etc.). To evaluate the behavior of PLM under these various internal and external parameters we consider three evaluation criteria. The first criterion is the bandwidth seen by each receiver. We want the mean bandwidth achieved by each receiver to be close to the available bandwidth. Moreover, we want the receivers to converge fast to the optimal rate and to be stable at this optimal rate. Our second criterion is therefore the number of layers subscribed. However, a bandwidth close to

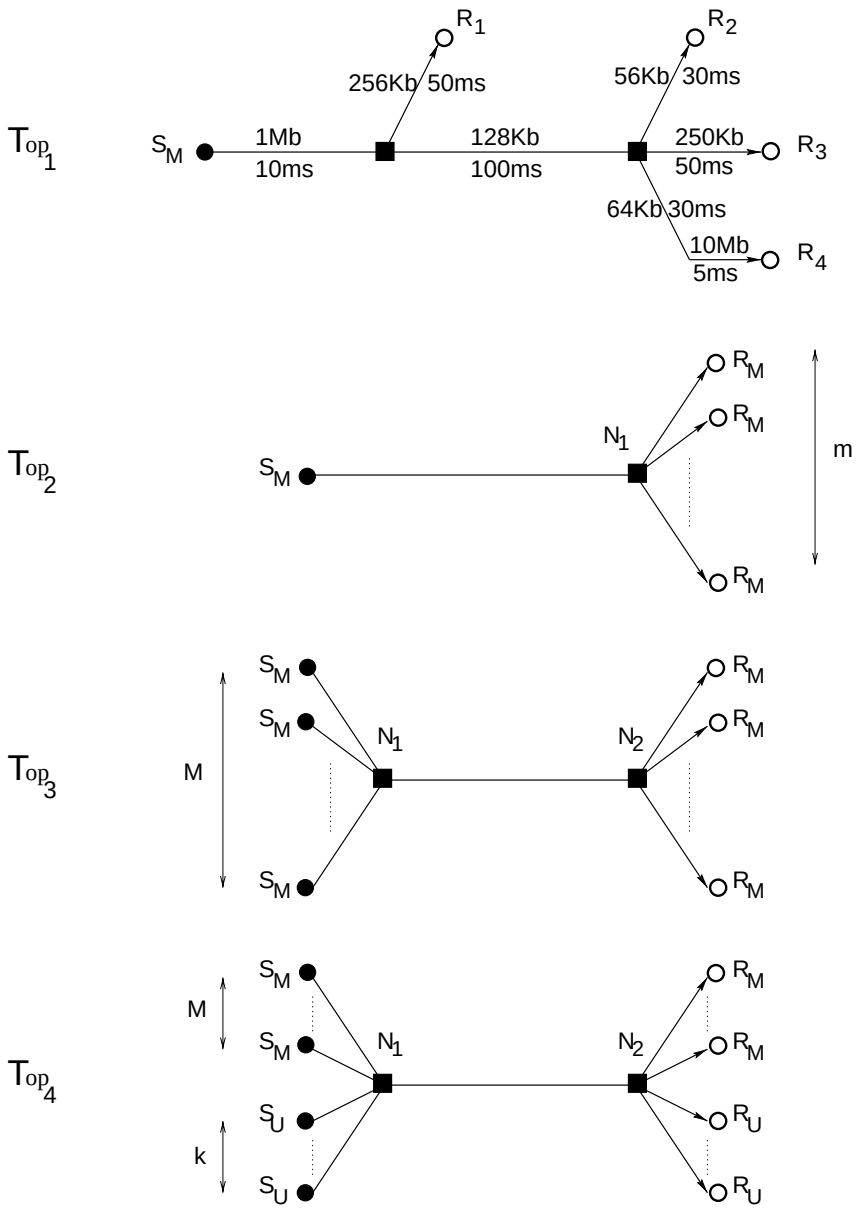


Figure C.2: Simulation Topologies.

the optimal rate and a stable layer subscription are not sufficient. Therefore, our third evaluation criterion is the loss rate.

C.4.2 Initial Simulation Topologies

Fig. C.2 shows the four topologies simulated to evaluate the PLM behavior. The first topology Top_1 consists of one PLM source and four PLM receivers. We evaluate the speed and the accuracy of the convergence in the context of a large heterogeneity of link bandwidths and link delays. The second topology Top_2 consists of one PLM source and m PLM receivers. We

evaluate the scalability of PLM with respect to session size. The third topology Top_3 consists in M PLM sources and one PLM receiver per source. We evaluate the scalability of PLM with respect to the number of sessions and the inter-PLM session fairness. The last topology Top_4 consists of M PLM sessions (with one receiver), and k unicast sessions. We evaluate the scalability of PLM with an increasing number of unicast sessions, and the fairness of PLM with respect to the unicast sessions.

We have implemented the PLM protocol in the ns [62] simulator. We use the following default parameters for our simulations. If we do not explicitly specify a parameter, the default parameters are used. The routing protocol is DVMRP (in particular graft and prune messages are simulated). All the queues are Fair Queuing (FQ) queues and each flow (PLM session or unicast flow) has a queue size of 20 packets. We chose the packet size of all the flows (PLM and TCP) to be 500 bytes. Each layer of a PLM source is simulated with a CBR source sending packets using PP. To avoid synchronization of the CBR sources we add randomness in the generation time of the PP. The check value is $C = 1$ second. For the experiment that simulates TCP flows, we use the ns implementation of TCP Reno. To exclude the influence of the max-window, we choose a max-window of 4000 packets.

C.4.3 Initial PLM Simulations Results

C.4.3.1 Basic Scenarios

We start our simulations evaluating the speed and the accuracy of the PLM convergence on topology Top_1. We choose 10 Kbit/s per layer. This very fine grain choice of bandwidth increment is a tough test for PLM as pointed out in section C.4.1. Fig.C.3 shows the results of this simulation. All receivers join the session at $t = 5$ s. Before $t = 7$ s (a check value plus the time to graft the appropriate groups) all the receivers converge to the optimal layer (i.e. the highest number of layers possible to use the available bandwidth). Moreover, the receivers stay at the optimal layer during the whole simulation and without loss induced (there is no loss during the whole simulation). In conclusion of this first experiment, PLM converges to the optimal link utilization in the order of a check value C and stays at this rate with no loss induced.

Our second simulation on topology Top_2 considers the scaling of PLM with respect to the number of receivers. We chose 50 Kbit/s bandwidth per layer. For this simulation we consider the link (S_M, N_1) with a bandwidth of 280 Kbit/s and a delay of 20 ms. The links (N_1, R_M) have a bandwidth uniformly chosen in $[500, 1000]$ Kbit/s and a delay uniformly chosen in $[5, 150]$ ms. The first set of experiments considers m receivers, $m = 1, ..., 100$ all started at $t = 5$ s. For this set of experiments, all the receivers converge to the optimal layer at the same time and stay at this layer without loss induced, whatever the number of receivers is. We do not give the plot

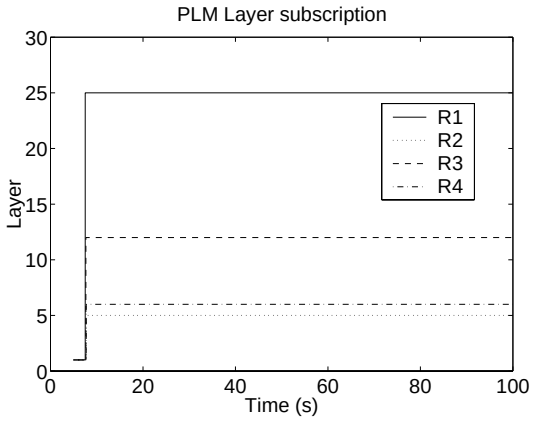


Figure C.3: Speed, accuracy, and stability of PLM convergence for a single session, Top_1.

of these experiments as Fig.C.4 shows the same behavior. In Fig.C.4 we show the results of an experiment where we start 20 PLM receivers at time $t = 5$ s then we add one receiver every five seconds from $t = 30$ s to $t = 50$ s, and at $t = 80$ s we add 5 PLM receivers. The aim of this last experiment is to evaluate the impact of the number of receivers on the convergence time and on the stability, and to evaluate the impact of late joins. When a receiver joins the session he starts at layer one and after a check value C he joins the optimal layer five (the vertical lines in Fig.C.4). We see in Fig.C.4 that neither the number of receivers nor the late joins have an influence on the convergence time and on the stability. This is not trivial for a receiver-driven protocol. Indeed, RLM for instance uses shared learning that leads to join synchronization (see [49]). Therefore, for RLM late joins have an influence on the convergence time of the other receivers. Once again, for all the experiments on topology Top_2 we do not observe any packet loss, and once the optimal layer is reached, the receivers stay at this layer for the whole simulation. We can easily explain these results as the layers subscribed only depend on the PP received; the receivers joining the session do not have an impact on the PP received by the others receivers. Multiple receivers joining the session at the same time, late joins, multiple late joins do not have any influence on the PLM session. In conclusion PLM perfectly scales with the number of receivers.

As we see in the previous experiments the number of receivers does not have any influence on the dynamics of a PLM session. Therefore, for all the other experiments we always consider PLM sessions with only one receiver.

Up to now we see a perfectly stable behavior of PLM, however to really evaluate the stability of PLM we must consider dynamic scenarios. The simulations on topology Top_4 consider dynamic scenarios. The links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We consider a layer granularity of 20 Kbit/s.

The first experiment on topology Top_4 considers a mix of PLM and CBR flows. We consider

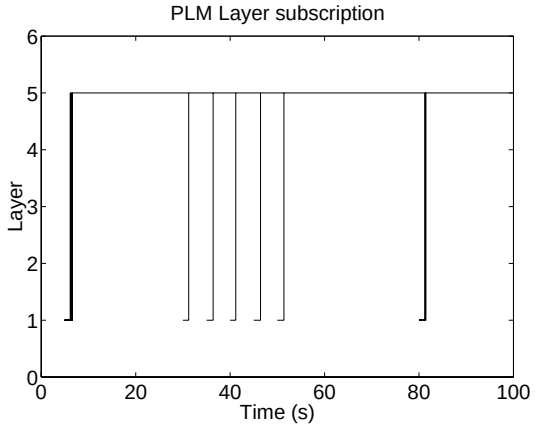
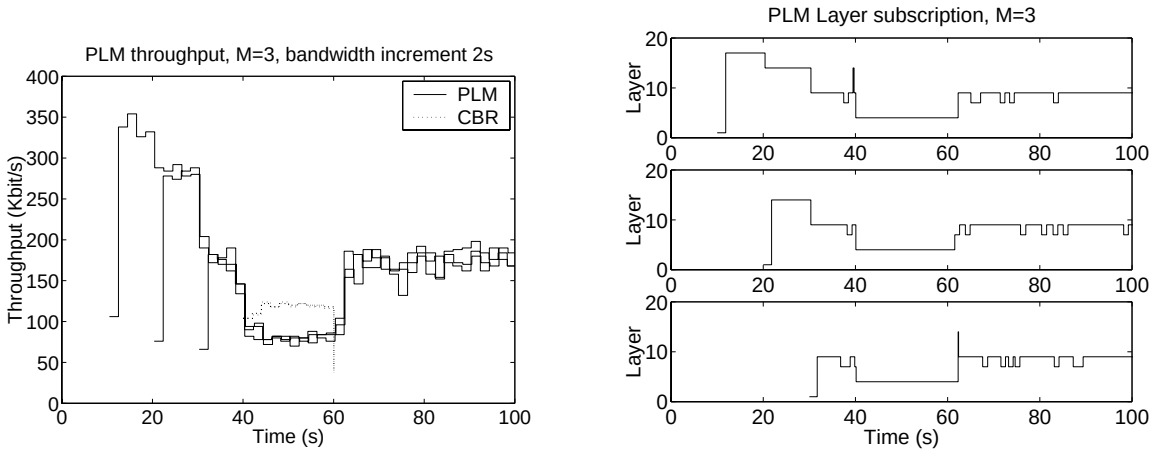


Figure C.4: Scaling of a PLM session with respect to the number of receivers, Top_2.

$M = 3$ PLM sessions and $k = 3$ CBR flows for the experiment plotted in Fig. C.5. The bandwidth of link (N_1, N_2) is $200 * M = 600$ Kbit/s and the delay is 20 ms. We start each of the three PLM receivers respectively at time $t = 10, 20, 30$ s. We start the three CBR sources at time $t = 40$ s and stop the CBR sources at $t = 60$ s. The aim of this scenario is to study in the first part (before starting the CBR sources) the behavior of PLM with an increasing number of PLM sessions, and in the second part (after starting the CBR sources) the behavior of PLM in case of severe congestion. When the CBR sources stop we observe the speed of PLM to grab the available bandwidth. We choose as many CBR sources as PLM sessions to simulate severe congestion. Indeed, with FQ, the only way to create congestion is to significantly increase the number of sessions. We have a bottleneck link at 600 Kbit/s, but the first PLM receiver can only get 340 Kbit/s (see Fig. C.5(a)). This is only due to the limited number of layers, in our simulation we have 17 layers, so $17 * 20 = 340$ Kbit/s if the receivers subscribe all the layers. When a new PLM session starts, the PLM sessions share equally the bottleneck at roughly (due to the layer granularity) $600/M$ Kbit/s where M is the number of PLM sessions. When the CBR sources start, the PLM sessions share the bottleneck at roughly $600/(M + k)$ Kbit/s where k is the number of CBR flows. We note that the CBR flows have a slightly higher share than the PLM flows. This is simply due to the layer granularity. The theoretical share is $600/6 = 100$ Kbit/s, however as we have chosen the conservative algorithm for PLM (see section C.3.3), PLM infers an available bandwidth of 80 Kbit/s and therefore joins 4 layers. All the three PLM sessions have the same behavior. The CBR flows grab has much available bandwidth they can, in this case 120 Kbit/s each. When the CBR sources stop, the PLM sessions increase their subscriptions to layers until reaching the highest link utilization (according to the layer granularity). PLM is very reactive and takes all the bandwidth available. In Fig. C.5(b) we see that changes in layer subscription to adapt exactly to the available bandwidth are within the time of one check value C (here $C = 1$ second). Moreover during the whole simulation (and even



(a) Bandwidth of each PLM and CBR receiver.

(b) Layer subscription of the PLM receivers.

Figure C.5: PLM and CBR flows sharing the same bottleneck, Top_4.

during the period of severe congestion) the PLM sessions experience no loss. PLM does not need large buffer sizes to absorb transient period of congestion (from the time the congestion starts to the time the reaction of PLM has an effect on the buffer). Indeed, the congestion is detected by a PP already when the congestion starts (we do not need a buffer to fill or to overflow to detect congestion) and the period of transient congestion (until the prune sent by the PLM receiver reaches the bottleneck) lasts less than a RTT. We repeated this experiment with different number of PLM sessions ($M = k = 1, 5, 10$) and different burst sizes (from 2 to 4 packets in a burst) and did not see any significant modification in the behavior of PLM. We observed some losses for the PLM sessions for a burst of 4 packets. However, these losses are infrequent and appear principally when a new session starts.

The second experiment on topology Top_4 considers a mix of a PLM sessions and TCP flows. We consider $M = 1$ PLM session and $k = 2$ TCP flows for the experiment plotted in Fig. C.6. The bandwidth of link (N_1, N_2) is $100 * (M + k) = 300$ Kbit/s and the delay is 20 ms. We start the first TCP flow at $t = 0$ s, then we start the PLM session at $t = 20$ s and finally start the second TCP flow at $t = 60$ s. The aim of this experiment is to study the behavior of PLM with TCP flows in a simple experiment. We see that a single PLM session perfectly adapts to the available bandwidth in presence of TCP flows. Here again this adaptation is in the order of one check value and induces no loss for the PLM sessions.

We can draw a partial conclusion at this point of our study: PLM reaches the optimal layer in the order of one check value C and tracks the optimal layer in a dynamic environment without loss induced for the PLM sessions.

In [49] we evaluate RLM [55] and RLC [87] on a subset of these basic scenarios. In order

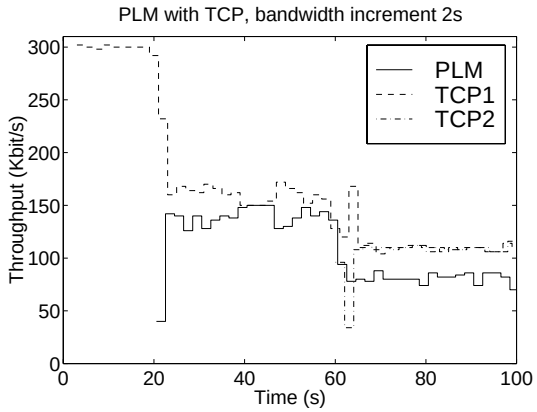


Figure C.6: PLM and TCP flows sharing the same bottleneck, Top_4.

to compare RLM and RLC with PLM, we must consider FQ scheduling for all the simulations. However, both RLM and RLC were designed with FIFO scheduling. As all the simulations done with FQ scheduling (for RLM and RLC) outperform the simulations done with FIFO scheduling, we consider that the use of FQ scheduling in the simulations for RLM and RLC is at the benefit of these protocols. Whereas PLM perfectly behaves in these basic scenarios, we shown in [49] that both RLM and RLC exhibit fundamental pathological behaviors on these same scenarios. PLM clearly outperforms RLM and RLC.

C.4.3.2 Multiple PLM Sessions

To further explore the properties of PLM we do experiments to study the scalability of PLM for an increasing number of PLM sessions. This experiment is on topology Top_3. The links (S_M, N_1), and (N_2, R_M) have a bandwidth of 10 Mbit/s and a delay of 5 ms. The link (N_1, N_2) has a bandwidth of $200 * M$ Kbit/s where M is the number of PLM sessions. All the sessions start randomly in $[5, 10]$ seconds. We do a measurement from $t = 20$ s to $t = 100$ s. For this experiment we vary the value of C (1 and 5 seconds), the value of the PP $Burst$ size (2, 3, and 4 packets), and the bandwidth distribution of the layers (20 Kbit/s and 50 Kbit/s). For each simulation we vary the number of PLM sessions. We only show the individual plots for $C = 1$, $Burst = 2$, and a layer granularity 50 Kbit/s, as this experiment is pathological and shows interesting behaviors (see Fig. C.7 and Fig. C.8). In Fig. C.7 a 'x' shows the mean bandwidth for a simple receiver. The solid line shows the mean bandwidth for all the receivers. We see that the mean bandwidth increases from 150 Kbit/s for a small number of sessions, to 195 Kbit/s for a large number of sessions. The layer granularity is 50 Kbit/s and the available bandwidth for each session is 200 Kbit/s. Due to the conservative algorithm (see section C.3.3) the PLM session infers a bandwidth at 150 Kbit/s and so joins 3 layers. However, the number of sessions increases, the number of sessions that do not join the fourth layer increases, and so the number

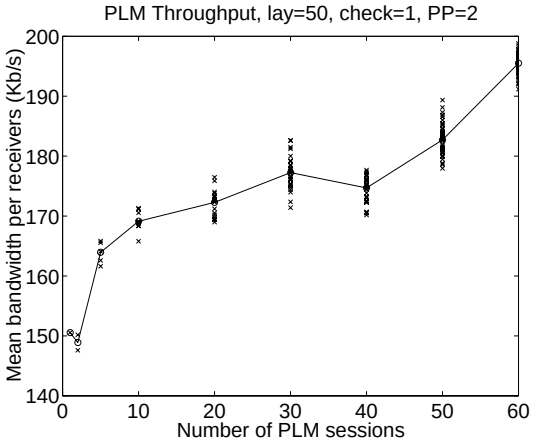
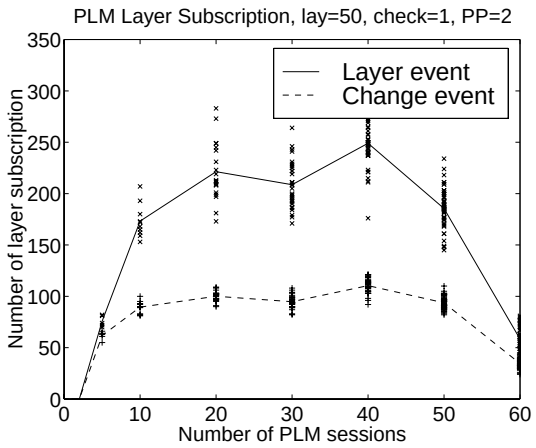


Figure C.7: PLM throughput, C=1, layer granularity 50 Kbit/s, Burst of 2 packets, Top_3.

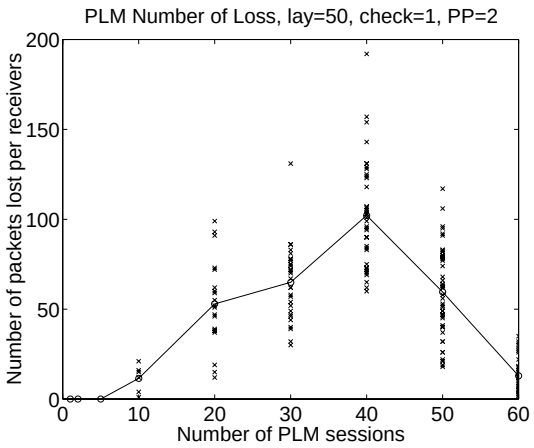
of sessions that can get this available bandwidth. Therefore, even with a large layer granularity the PLM sessions reach the available bandwidth in the case of a high degree of multiplexing. We obtained the same results for the mean bandwidth with the other parameters studied. We note that PLM achieves good inter-PLM fairness as the mean bandwidth for the individual receivers is close ($\pm 5\%$)to the mean bandwidth for all the receivers (small standard deviation).

Concerning the number of layers subscribed, we distinguish in Fig. C.8(a) between the total number of individual layers a receiver joins during the whole simulation experiment (*layer event* in Fig. C.8(a)) and the total number of events to change layers (*change event* in Fig. C.8(a)). For instance, if a receiver is at layer 2 and at time t he infers enough available bandwidth to join layer 5, we say that at t there are 3 layer events but 1 change event.

This scenario, with PLM sessions only, is pathological and is not realistic in the Internet where we have a mix of fine grain adjustment (1 packet per RTT like TCP) protocols and large grain adjustment (one layer like PLM) protocols. In Fig. C.8(a) we see the total number of layer subscriptions during the simulation. Several layers are added or dropped in a single event, roughly two layers per event because the number of layer events is roughly two times the number of change events. In fact, if the routing protocol allows cumulative graft, the change event curve has to be considered (as all the layers change in a single event can be send in one cumulative graft or prune), otherwise the layer event curve is considered (this curve shows the number of graft or prune sent in case of non cumulative graft or prune). We note the striking similarity between Fig. C.8(a) and Fig. C.8(b). The shape of both plots is due to the same phenomenon. The available bandwidth for each session is the bottleneck bandwidth divided by the number session that share the same bottleneck. A PP can only infer the right available bandwidth if, at the moment the PP reaches the bottleneck, all the sessions are back-logged in the queue. If a session use less than its fair share, another session can get the bandwidth. However, even if some sessions use more than their fair share (because other sessions use less), a PP infers the



(a) PLM layer subscription.

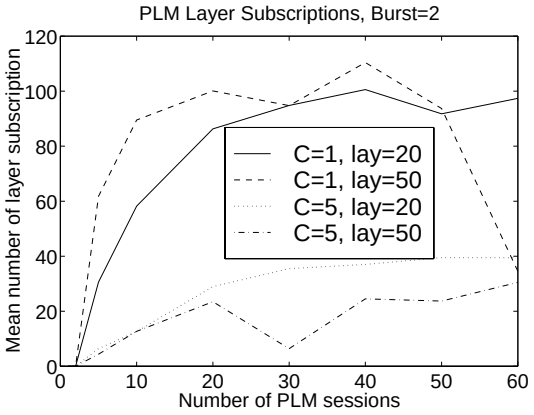


(b) PLM losses.

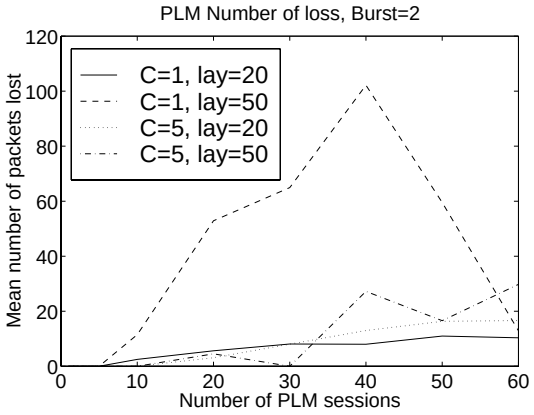
Figure C.8: PLM layer subscription and losses, C=1, layer granularity 50 Kbit/s, Burst of 2 packets, Top_3.

bandwidth available only dependent on the number of sessions back-logged but independent on the bandwidth really used by these sessions. Therefore, several sessions can infer in parallel a higher available bandwidth than their fair share. That leads to oscillation (see Fig. C.8(a)) and losses (see Fig. C.8(b)). For a large number of sessions there is such a high multiplexing that the sessions use, in the mean, a bandwidth close to the available bandwidth. There is less free bandwidth to grab, the oscillations and the losses significantly diminished (see Fig. C.8).

This simulation scenario studied represents a worst case result for the layer subscription oscillation and the loss rate for all our simulations. This is pathological since we only have PLM sessions with a large layer granularity compared to the available bandwidth for each flow. That results in a wrong estimation of the available bandwidth causing oscillation and, due to the large layer granularity, losses. Even in this pathological case PLM performs reasonably well, as the oscillations do not lead to a decrease in the mean bandwidth. In fact, a session can never infer an available bandwidth lower than the real bandwidth available when all the session are back-logged. Moreover the losses induced by these oscillations are reasonably low. The mean loss is, even in the worst case, less than 2% of packets lost. However this pathological case can be avoided by adjusting correctly the PLM parameters (check value, bandwidth granularity), and is improbable in a real network. In Fig. C.9 we have the layer subscription (change event) and the number of losses for various scenarios. We see that increasing the check value C significantly reduces the number of losses (see Fig. C.9(b)) and the number of oscillations (see Fig. C.9(a)). In a real network we would have a mix of adaptation granularity: Large granularity in the case of PLM and fine granularity in the case of TCP. Such a multiplexing of PLM and

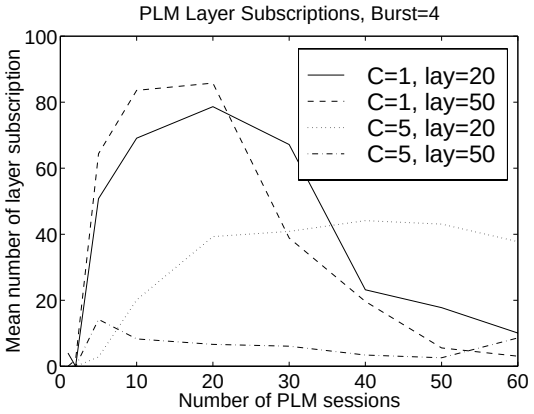


(a) PLM layer subscriptions.

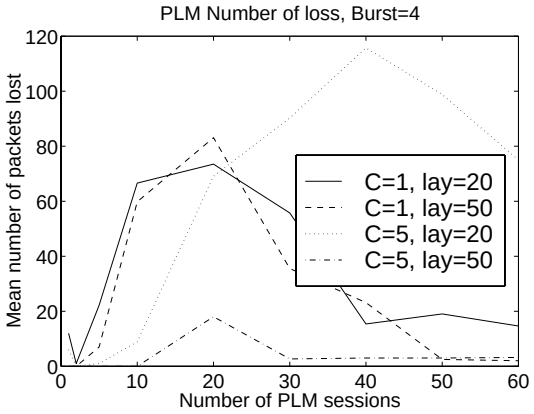


(b) PLM losses.

Figure C.9: PLM layer subscription and losses, Burst of 2 packets, Top_3.



(a) PLM layer subscription.



(b) PLM losses.

Figure C.10: PLM layer subscription and losses, Burst of 4 packets, Top_3.

TCP flows will avoid the presence of free available bandwidth, and so reduce the oscillations and the number of losses. This assertion is evaluated in the following with a mix of PLM and TCP flows.

Finally, increasing the size of the burst does not significantly change the mean bandwidth (we do not give the plot). Fig. C.10(a) shows the layer subscriptions for a burst of 4 packets. We see a slightly lower number of oscillations compared to the case of a burst of size of 2 packets (see Fig. C.9(a)), and we see that the curve has its maximum for a lower number of sessions. However, the large burst size significantly increases the number of losses (see Fig. C.10(b)).

In conclusion, PLM scales well with the number of sessions, achieves good inter-PLM

fairness, reaches the optimal rate in the order of one check value C, and leads to a low number of losses (even in the pathological case). The oscillations of PLM should not be considered as instability, they rather reflect the fact that PLM does not want to leave bandwidth unused (when there is bandwidth available PLM sessions try to get it). We saw that oscillations can be significantly reduced by increasing the check value. However, we will see that these oscillations disappear when there is a mix of TCP and PLM flows.

C.4.3.3 Multiple PLM Sessions and TCP Flows

Our next experiment on topology Top_4 considers a mix of PLM sessions and TCP flows. We consider M PLM and $k = M$ TCP sessions. The bandwidth of link (N_1, N_2) is $100 * (M + k)$ Kbit/s and the delay is 20 ms. The links (S_M, N_1), and (N_2, R_M) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We start the TCP flows randomly in $[5, 10]$ s and the PLM sessions randomly in $[20, 25]$ s. We do the measurement between $t = 30$ s and $t = 100$ s. The aim of this experiment is to study the scaling properties of PLM with concurrent TCP flows. We repeat this simulation for various check values ($C = 1, 5$), burst sizes (2, 3, and 4 packets), and bandwidth granularities (20 Kbit/s and 50 Kbit/s). We plot the results for $C = 1$, a burst of 2 packets, and a layer granularity of 20 Kbit/s, as this experiment is very representative.

Fig. C.11 shows the throughput for each receiver for the PLM and the TCP flows. Due to the layer granularity, for a small number of sessions, PLM sessions can only get 80 Kbit/s (compared to the 100 Kbit/s available for each flows). When the number of sessions increases, the PLM sessions get more bandwidth due to multiplexing. The TCP flows have exactly the opposite behavior of the PLM sessions. PLM achieves a good inter-PLM fairness, indeed the mean bandwidth for the individual receivers remains close to the mean bandwidth for all the receivers.

Fig. C.12(a) shows the layer subscription. The change event curve and the layer event curve are identical. That means there is never a change of more than 1 layer at the same time. Moreover, the number of oscillations is low with $C = 1$ and can be significantly reduced with $C = 5$. We have few oscillations, and these oscillations lead to *no loss* for a burst of 2 packets (see Fig. C.12(b)) and a very low number of losses for a burst of 4 packets. We see that with concurrent TCP flows, PLM behaves remarkably well. We have an optimal link utilization, inter-PLM fairness, a very low number of oscillations, and no loss induced.

C.4.3.4 Variable Packet Size

For all the previous simulations we always considered packets of the same size. We know that FQ guarantees an equal share of the bottleneck to each flow. Therefore, if flows have different packet sizes, the PP may lead to wrong estimate of the available bandwidth.

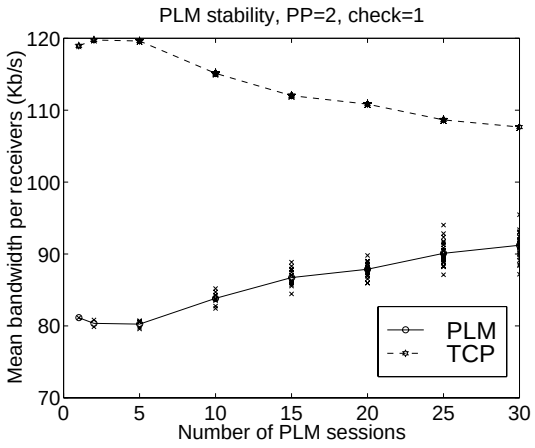
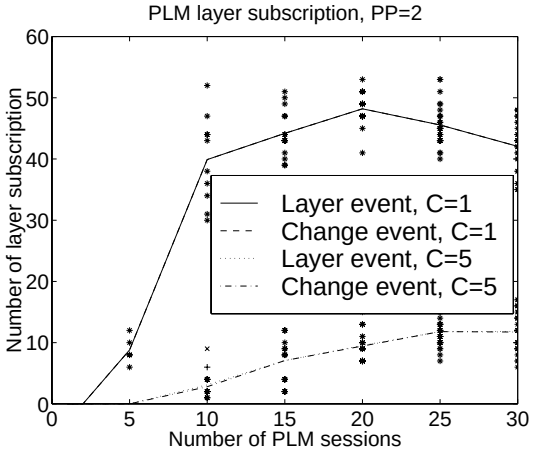
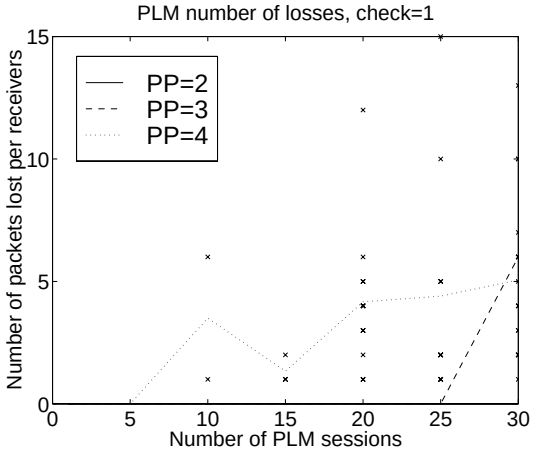


Figure C.11: Throughput for a mix of PLM and TCP flows, C=1, burst of 2 packets, 20 Kbit/s layer granularity, Top_4.



(a) PLM layer subscription, C=1,5, Burst of 2 packets.



(b) PLM losses, C=1, Burst of 2, 3, and 4 packets.

Figure C.12: Layer subscription and losses for the PLM sessions for a mix of PLM and TCP flows, 20 Kbit/s layer granularity, Top_4.

Imagine a bottleneck queue with a capacity of B and two concurrent flows F_1 and F_2 with a packet size of 500 and 300 Bytes respectively. Fig. C.13 shows the packet service time of the buffered packets of each flow. The available bandwidth inferred by PP_1 of F_2 is $\frac{300}{t_3-t_1} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth inferred by PP_2 of F_2 is $\frac{300}{t_6-t_5} = \frac{300}{\frac{300}{B}} = B > \frac{B}{2}$, the available bandwidth inferred by PP_3 of F_2 is $\frac{300}{t_{10}-t_8} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, etc. The available bandwidth inferred by PP_1 of F_1 is $\frac{300}{t_4-t_2} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth inferred by PP_2 of F_1 is $\frac{300}{t_9-t_7} = \frac{300}{\frac{300}{B}+\frac{500}{B}} = \frac{300}{800}B < \frac{B}{2}$, the available bandwidth

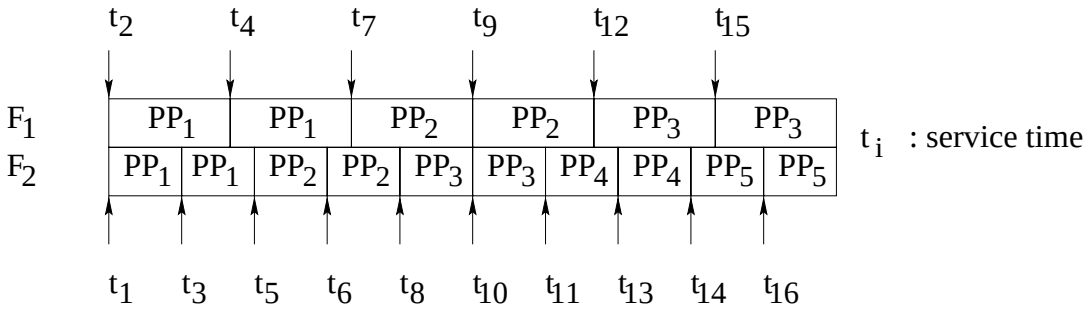


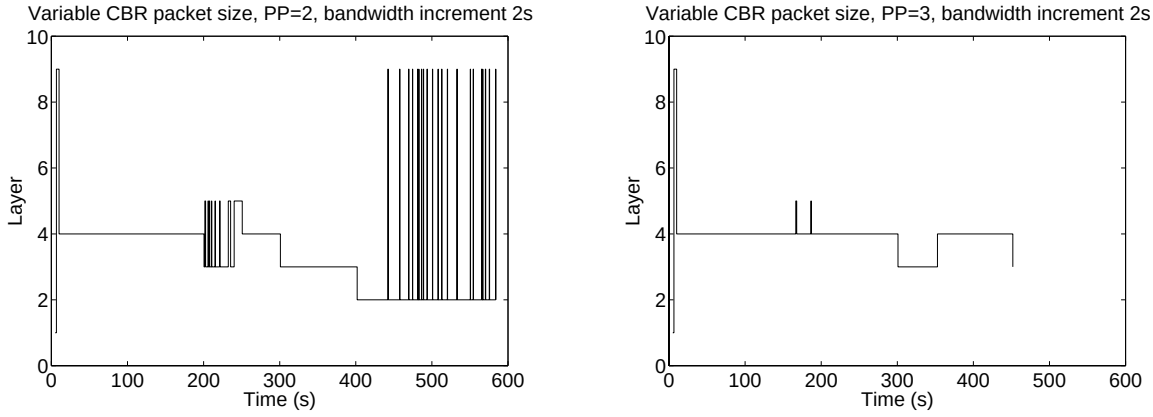
Figure C.13: Service time of packets of variable size in a single FQ queue.

inferred by PP_3 of F_1 is $\frac{300}{t_{15}-t_{12}} = \frac{300}{\frac{300}{B}+\frac{300}{B}+\frac{500}{B}} = \frac{300}{1100}B < \frac{B}{2}$. In this simple example, when the packet size of the flows is different, the PP bandwidth inference is not accurate. We see that the most significant problem comes from small packets compared to the size of the packets of all the other flows crossing the same FQ queue. Indeed, this is the only case where a PP can be served in a single round and so infers an available bandwidth equal to the bottleneck bandwidth (see PP_2 for flow F_2). If we are not in the case of small packets, the PP can never overestimate the available bandwidth.

However, there are ways to avoid this problem. First, it is often possible to take large packets for multimedia streams. If, however, the packets must be small, we can increase the size of the burst and so avoid the overestimating effect due to the small packets. Second, multiplexing of flows significantly reduces the problem due to the packet size. Indeed, with large number of flows crossing a single FS router, the number of flows becomes more important than the packet size of each flow. These two remarks are confirmed in the following simulations.

In Fig. C.14 we study the impact of the packet size on the PP bandwidth inference. We use topology Top_4, the links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. The link (N_1, N_2) has a bandwidth of 400 Kbit/s and a delay of 20 ms. We consider 20 Kbit/s layers. We have $M = 1$ PLM session, and $k = 1$ CBR flow. We start the PLM session at $t = 5$ s. We vary the packet size of the CBR flow during the time while fixing the same size for the PLM packet to 500 Bytes. We start at $t = 10$ s a CBR flow with packets of 50 Bytes. At $t = 50$ s and every 50 seconds the packet size of the CBR flow changes to the following values: 100, 200, 300, 400, 500, 750, 1000, 1250, 1500, 1750, and 2000 bytes. We see that packet sizes lower than the PLM packets can lead to a wrong estimate, however the worst estimates are for packet size larger than the PLM packet size (see Fig. C.14(a)). Increasing the burst size to three significantly reduce the error in the estimate (see Fig. C.14(b)). The reason is that the number of bytes in a burst is larger than the largest packet size of concurrent CBR flow. Therefore, a FQ queue can never serve a whole burst in a single round.

Fig. C.15 presents the effect of the flow multiplexing for the PP inference when each flow



(a) Burst=2.

(b) Burst=3.

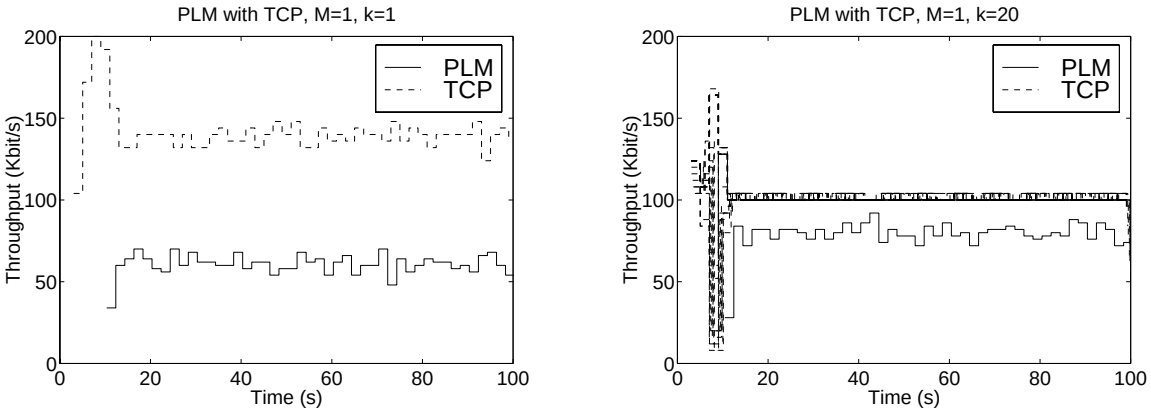
Figure C.14: Mix of PLM and CBR flows. Influence of the Burst size on the bandwidth inference for variable packet size, Top_4.

has a different packet size. We use topology Top_4, the links (S_M, N_1), (S_U, N_1), (N_2, R_M), and (N_2, R_U) have a bandwidth of 10 Mbit/s and a delay of 5 ms. We consider 20 Kbit/s layers. We have $M = 1$ PLM session, and $k = 1, 20$ TCP flows. The link (N_1, N_2) has a bandwidth of $100 * (k + M)$ Kbit/s and a delay of 20 ms. We start the PLM session at $t = 5$ s and randomly start the TCP flow(s) in $[0, 5]$ s. The PLM packet size is 500 Bytes, the TCP packet size is 1000 Bytes. Fig. C.15(a) shows the plot for $k = 1$ TCP flow. The PLM flow gets only 60 Kbit/s instead of 80 Kbit/s due to the wrong estimates. However when we increase the number of TCP flows, the wrong estimates disappear and PLM gets its available bandwidth at 80 Kbit/s.

In conclusion, whereas different packet size can lead to a wrong estimate, choosing larger packet sizes for the PLM flows, the natural multiplexing of the flows, and increasing the burst size significantly reduce the problems related to the correct bandwidth estimation in case of packets of different size.

C.5 Simulations with a Realistic Background Traffic

Recent works show the evidence of self similar [51] and even multifractal [27] traffic in the Internet. The burstiness over many time scales of such a traffic can adversely impact congestion control protocols. Moreover, it is commonly argued that the PP bandwidth inference mechanism is highly sensitive to the traffic pattern. These arguments motivate this set of simulation that aims to study the performances of PLM with self similar and multifractal background traffic.



(a) One PLM session and one TCP flow.

(b) One PLM session and twenty TCP flows.

Figure C.15: Mix of PLM and TCP flows. Influence of the multiplexing on bandwidth inference. PLM packet size: 500 Bytes, CBR packet size 1000 Bytes, Top_4.

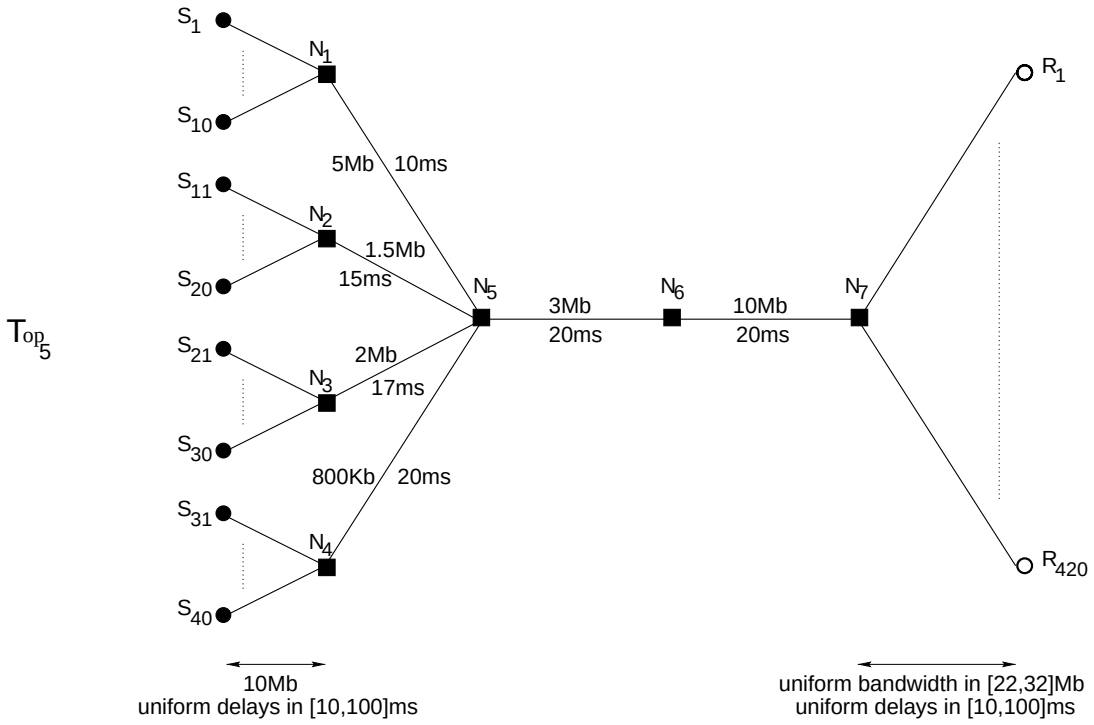


Figure C.16: Simulation topology Top_5 for the realistic background traffic.

C.5.1 Simulation Scenario

To simulate self similar and multifractal traffic we reuse a large part of the ns scripts and of the scenarios used in [27]. In the following, we explain the scenario we considered in this

paper. Fig. C.16 shows the topology considered for our simulations. Let $\{S_i, i = 1, \ldots, 40\}$ be the set of web servers and $\{R_i, i = 1, \ldots, 420\}$ be the set of web clients. Client and server communicate with a ns version of HTTP 1.0 and of TCP Reno. Let N_S be the number of sessions, each session contains 300 pages, each page contains one object. Each session is defined for a client randomly chosen among the set of clients. For a given session, the client requests each page on a server randomly chosen among the set of servers. All the objects of a same page are requested on the same server. The inter-session starting time is exponentially distributed with a mean of 1s, the inter-page starting time is exponentially distributed with a mean of 15s. The inter-page starting time is the same that the inter-object starting time as there is only one object per page. The object size is pareto distributed (pareto II) with a mean of 12 packets/object and a shape of 1.2. The TCP packet size is 1000 Bytes. We run all our simulations for 4500 simulated seconds.

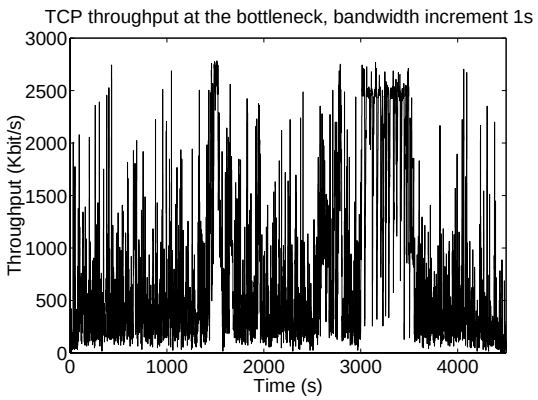
Feldmann et al. shown that this scenario produces a traffic close to a real Internet traffic. In particular, the traffic is self similar at large and medium time scales and multifractal at small time scales (see [27] for details).

Topology Top_5 has 4 bottlenecks, the links (N_2, N_5), (N_3, N_5), (N_4, N_5), and (N_5, N_6). All the queues are fair queuing with a shared buffer of 100000 Bytes. A flow for the FQ scheduler is defined as one (source,receiver) pair for the TCP flows, a PLM session is considered as one flow. We place a PLM source at node S_{10} and a PLM receiver at node R_{100}. The bottleneck link for the PLM flow is the link (N_5, N_6). If we do not specify, the bottleneck link always refers to the PLM bottleneck link. For all the simulations we start the PLM session at $t = 50$ s. Remember that PLM scales perfectly with the number of receivers. Increasing the number of receivers in the simulations does not change our results. In the following we take only one PLM receiver for the PLM session.

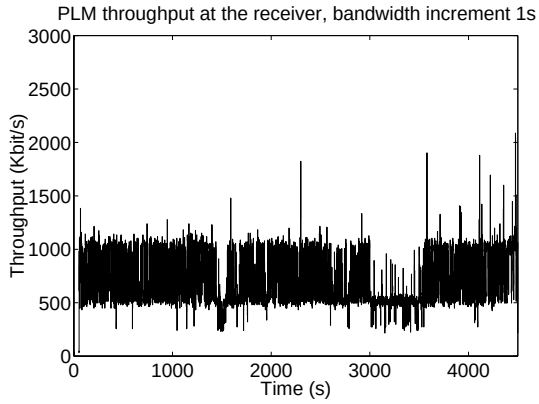
C.5.2 PLM Simulations Results with Realistic Background Traffic

We call the self similar and multifractal web traffic the background traffic. We consider two kinds of background traffic: a lightly loaded with $N_S = 100$ sessions; a heavy loaded with $N_S = 400$ sessions. The lightly loaded scenario is characterized by a mean throughput for the background traffic of about 600 Kbit/s and a loss rate around 0.4%. The heavy loaded scenario is characterized by a mean throughput for the background traffic of about 2200 Kbit/s and a loss rate around 1.5%. All the other parameters varied during the simulations are related to PLM. We consider various check values ($C \in \{0.2, 0.5, 1, 5\}$), two layer distributions (100 Kbit/s layers, exponentially distributed layers: 32 Kbit/s, 64 Kbit/s, 128 Kbit/s, etc.), and two PLM packet sizes (500 Bytes and 1000 Bytes).

In order to get a benchmark for the efficiency of PLM, we replace for the two kinds of background traffic the PLM session with a TCP flow where the TCP sender always has data



(a) Mean throughput averaged on 1s intervals for the background traffic at the bottleneck (N_5, N_6).



(b) Mean throughput averaged on 1s intervals for the PLM flow at the PLM receiver.

Figure C.17: $N_S = 100$, $C = 1$, 1000 bytes PLM packet size, exponential layers.

to send (the TCP source is at node S_{10} and the TCP receiver at node R_{100}). We call this TCP flow the benchmark TCP flow, or TCP$_b$. For the lightly loaded scenario, the mean throughput for the background traffic is 569 Kbit/s. The mean throughput achieved by the benchmark TCP connection is 1453 Kbit/s with a loss rate of 0.185%. For the heavy loaded scenario, the mean throughput for the background traffic is 2318 Kbit/s. The mean throughput achieved by the benchmark TCP connection is 479 Kbit/s with a loss rate of 0.809%. Remember that the bottleneck bandwidth for TCP$_b$ (and PLM) is 3 Mbit/s.

Fig. C.17 shows the results of a simulation with a lightly loaded background traffic ($N_S = 100$). The PLM check value is $C = 1$, the PLM packet size is 1000 Bytes, and the layers are exponentially distributed. We first note that the PLM session closely follows the evolutions of the throughput of the background traffic (for instance look at $t = 1500$ s). Moreover, PLM does not experience any loss during the whole simulation. PLM is able to track the available bandwidth without loss induced even in complex environments. The mean throughput for the whole simulation for the background traffic is 737 Kbit/s, and is 733 Kbit/s for the PLM session.

Fig. C.18 shows the layer subscription of the PLM receiver during the simulation. Most of the layer subscription oscillations are between layer 5 (512 Kbit/s) and layer 6 (1024 Kbit/s). There are 2090 layer changes during the simulation (roughly 1 layer change every 2 seconds). The layer subscription oscillation is not a weakness of PLM, but a consequence of the high efficiency of PLM. As the background traffic fluctuates in a large range of time scales, PLM must join and drop layers in a large range of time scales in order to get the available bandwidth. PLM is not unstable, it is the background traffic that is “unstable” and PLM simply follows its variations. We see in all the simulations a tradeoff between the layer subscription oscillation

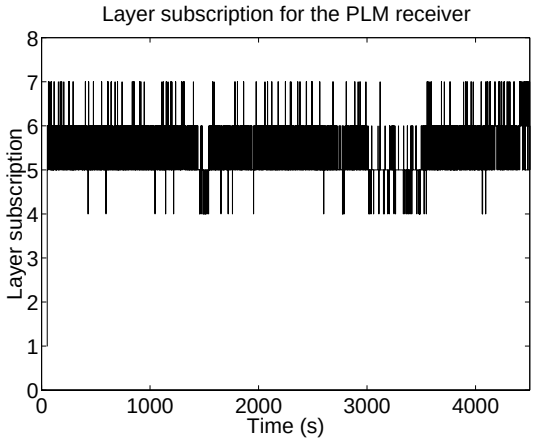


Figure C.18: Layer subscription for the PLM receiver.

and the efficiency of PLM. Increasing the layer granularity (from 100 Kbit/s to exponentially distributed layers) or increasing the check value reduces the layer subscription oscillation but reduces the efficiency of PLM as well (because the more the layer subscription oscillates, the more efficient PLM is).

For a check value $C = 0.2$ s and 100 Kbit/s layers, we obtain a surprising result. The mean throughput for the PLM session is 1507 Kbit/s, higher than the TCP benchmark. It is the only one experiment for the lightly loaded background traffic where observe some losses for the PLM flow. In fact, we get 22 losses corresponding to a loss rate lower than 3.10^{-5}. The PLM receiver generates 14010 change events and 124328 layer changes. We explain the big difference between layer change and the change event because the large variations in the available bandwidth enforce the PLM receiver to join and drop multiple layers in a single event. As explained in section C.4.3.2 with cumulative graft and prune the value of interest is the number of change events.

This result is fundamental for the evaluation of the efficiency of PLM. For the case of a single PLM multicast session from sender S to receivers $R_1, \ldots, R_n$ we observed that PLM could achieve for each receiver R_i a throughput B_i^{PLM} that is higher than the throughput B_i^{TCP} obtained by a single unicast TCP connection between S and that receiver R_i. This was not case for any previously proposed multicast congestion control protocol MCC where there was always, for at least one receiver R_i, $B_i^{MCC} < B_i^{TCP}$. However, the high number of layer changes is not possible with the actual multicast routing protocols. This experiment simply aims to show the high efficiency of PLM even in complex environments and the high efficiency of the packet pair bandwidth inference mechanism as we use it in PLM.

Stability in the layer subscription does not mean low efficiency. For a check value $C = 5$ s and exponentially distributed layers, the mean throughput for the PLM receiver is 561 Kbit/s with only 417 layer changes (roughly 1 layer change every 10 seconds) and no loss. Fig. C.19

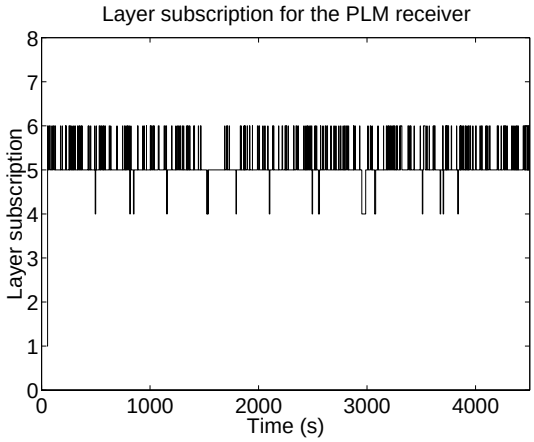


Figure C.19: $N_S = 100$, $C = 5$, 1000 bytes PLM packet size, exponential layers. Layer subscription of the PLM receiver.

shows the evolution of the layer subscription for this scenario. We see that layer subscription oscillate around layer 5 (512 Kbit/s).

For file transfer applications, layer subscription oscillation is not a drawback. These applications want to achieve the highest throughput. The limit in the number of layer subscription oscillations is only imposed by the routing protocol. For multimedia applications, frequent variations in the throughput (and thus in the quality perceived) can be a serious drawback. However, we believe it is the task of the application to filter the oscillations (for instance see [72]). If the application is not smart enough to filter the oscillations, we shown that an appropriate choice in the check value parameter can significantly reduce layer subscription oscillations with a reasonable decrease in the efficiency. In [47] we studied a bandwidth allocation policy that significantly rewards multicast sessions with respect to the number of receivers. Thus, a lower efficiency than TCP does not necessarily mean a lower throughput for a multicast session.

The set of experiments for the heavy loaded scenario ($N_S = 400$) confirms our conclusions and does not show other significant new results. The set of experiments with a PLM packet size of 500 Bytes simply shows a slight decrease in the mean throughput for PLM and a slight increase in the layer subscription oscillations. This is due to the error in the estimate with packets of different sizes (see section C.4.3.4). However, the multiplexing of the flows is sufficient to avoid large estimation errors. We do not notice any instability or significant inefficiency of PLM in these simulations.

In conclusion, we tested PLM in a large variety of situations and with different values of parameters and always found that PLM is efficient, stable, and induces no loss (or a negligible loss rate in some extreme situations) even with a realistic background traffic.

C.6 Validation of the FS-paradigm

We have devised PLM using the FS paradigm. So theoretically PLM must have the properties of an ideal congestion control protocol. In this section we check which properties of an ideal congestion control protocol PLM meets.

Stability In some pathological cases (only PLM sessions) PLM oscillates, however these oscillations are around the optimal rate and are fairly distributed among the receivers. Even in complex environments, PLM reaches and tracks the available bandwidth without loss induced.

Efficiency PLM is clearly efficient in the sense of satisfying the multimedia and file transfer applications: PLM allows for each receiver to get the maximum bandwidth available on the path between the source and this receiver.

Fairness PLM achieves inter-PLM fairness and TCP-fairness[2].

Robustness PLM robustness is guaranteed by the FS network.

Scalability PLM is highly scalable due to the receiver-driven cumulative layered scheme. PLM does not require any signaling nor feedback. The burden of PLM is distributed among the receivers.

Feasibility PLM requires a FS network, this issue is discussed in [48] and it is shown to be feasible per ISP (Internet Service Provider). PLM simply requires an IP-multicast network.

In conclusion, PLM meets all the properties of an ideal congestion control protocol. However the fundamental point (and the success) of the FS-paradigm is that we do not add any specific mechanism *in PLM* to meet these properties. For instance, to be satisfied, the user of a multimedia application requires: (i) to receive the highest quality (high throughput, low number of losses) and (ii) to avoid frequent modification in the quality perceived. Of course these two requirements are related with the properties of an ideal congestion control protocol, but we do not specifically address the properties of an ideal congestion control protocol in the design of PLM. The benefits of the FS-paradigm become obvious when we look at the number and the influence of the PLM parameters, which are the check value C, the burst size, and the layer granularity. We have only three parameters whose exact value is not very important for the correct behavior of PLM. The fine tuning of the check value, for instance, does not fundamentally change the properties of PLM but defines a slightly different operating point that can improve

[2]We talk about TCP-fairness that is different from TCP friendliness. TCP-fairness means that PLM does not significantly affect the performance (throughput, delay, etc.) of TCP flows when sharing the same bottleneck.

the satisfaction of a user. In the contrary to PLM, RLM has numerous parameters, where each parameter was introduced to improve the stability and the efficiency of RLM. A wrong choice of any such parameter can significantly decrease the performance of RLM. Its TCP-friendly version needs fine tuning of the parameters to effectively achieve TCP-friendliness.

PLM is significantly simpler than the previous cumulative layered receiver-driven protocols and yet outperforms these protocols. We therefore consider PLM as a practical proof of the validity of the FS-paradigm.

C.7 Related Work

Steven McCanne [55] first presented a cumulative layered scheme combined with a receiver-driven driven protocol to join and leave layers. His protocol, RLM, is the basis of most of the studies on cumulative layered protocols. These studies explored the properties of RLM ([35], [2]), or tried to improve the performance of RLM ([87], [88], [85]). Bajaj et al. explore the relative merits of uniform versus priority dropping for the transmission of layered video. As pointed out by the authors their results are “ambiguous” and can not be easily summarized. We refer the interested reader to [2]. Gopalakrishnan et al. study the behavior of RLM in various scenarios and find that RLM shows high instability for VBR traffic, has very poor fairness properties in most of the cases, and achieves a low link utilization with VBR traffic. Vicisano et al. devise a TCP-friendly receiver-driven protocol and propose how to achieve synchronization of the layer subscription. A TCP-friendly version of RLM was introduced in [85] where the layer subscription depends on a TCP-friendly formula. Since RLM leads to large oscillations under network congestion, Wu et al. propose to use thin streams [88].

The use of Packet Pair for bandwidth inference was introduced by Keshav [44] and used by Paxson (he introduces a more refined technique, PBM see[67]) and Ratnasamy [70] in the context of a FIFO network to discover the *bottleneck* bandwidth. The bottleneck bandwidth gives at most an upper bound for the *available* bandwidth. While the bottleneck bandwidth is interesting for many problems, it does not have a significant interest for congestion control.

C.8 Conclusion

We started our study with an introduction to the FS-paradigm. This paradigm allows to theoretically devise nearly ideal congestion control protocols. Then, we devised the PLM protocol according to the FS-paradigm. PLM is based on a cumulative layered multicast scheme and on the use of Packet Pair to discover the available bandwidth without doing any join experiment unlike RLM. We investigate the properties of PLM under various conditions and see that PLM

converges in the order of one check value to the optimal link utilization and stays at this optimal layer subscription during the whole simulation. Then we investigated the scalability of PLM with respect to the number of receivers and to the number of sessions. PLM behavior is fully independent of the number of receivers. However, concurrent PLM sessions can slightly decrease the performance of PLM. But adjusting correctly the PLM parameters significantly reduces the number of losses and the layer subscription oscillations. In a mix of TCP flows and PLM sessions, PLM performs well. PLM has a very low number of layer subscription oscillations and no loss is induced. We investigated the effect of variable packet size and saw that the multiplexing of the sessions significantly reduces the problems of convergence due to the packet size. Finally, we explore the impact of realistic background traffic on PLM. We obtain a confirmation of our results. PLM converges fast to the available bandwidth and track this available bandwidth without loss induced.

The introduction of our available bandwidth estimator based on packet pair solves the problems (efficiency, stability, loss induced, simplicity, fairness, etc.) of the previous layered multicast congestion control protocols. This is therefore a significant contribution to the field of multicast congestion control that makes PLM the new benchmark for the layered multicast protocols.

Another implication of the good performance of PLM is the practical validation of the FS-paradigm. The simulation results show that the simple mechanisms introduced in PLM lead to a very efficient congestion control protocol, as predicted by the FS-paradigm.

In conclusion, we have devised a layered multicast congestion control protocol that outperforms all the previous congestion control protocols like RLM and its TCP-friendly variants. Indeed PLM converges fast to the available bandwidth, induces no loss to discover and track the available bandwidth, requires no signaling, and scales with both the number of receivers and the number of sessions. All these properties hold in complex environments such as self similar and multifractal traffic.

Appendix D

Bandwidth Allocation Policies for Unicast and Multicast Flows

Abstract

Using multicast delivery to multiple receivers reduces the aggregate bandwidth required from the network compared to using unicast delivery to each receiver. However multicast is not yet widely deployed in the Internet. One reason is the lack of incentive to use multicast delivery. To encourage the use of multicast delivery, we define a new bandwidth allocation policy, called $LogRD$, taking into account the number of downstream receivers. This policy gives more bandwidth to a multicast flow as compared to a unicast flow that shares the same bottleneck, however without starving the unicast flows. The $LogRD$ policy provides also an answer to the question on how to treat a multicast flow compared to a unicast flow sharing the same bottleneck.

We investigate three bandwidth allocation policies for multicast flows and evaluate their impact on both receiver satisfaction and fairness using a simple analytical study and a comprehensive set of simulations. The policy that allocates the available bandwidth as a logarithmic function of the number of receivers downstream of the bottleneck achieves the best trade-off between receiver satisfaction and fairness.

Keywords: Unicast, Multicast, Bandwidth Allocation Policies.

D.1 Introduction

There is an increasing number of applications such as software distribution, audio/video conferences, and audio/video broadcasts where data is destined to multiple receivers. During the last decade, multicast routing and multicast delivery have evolved from being a pure research topic [18] to being experimentally deployed in the MBONE [24] to being supported by major

router manufacturers and offered as a service by some ISPs. As a result, the Internet is becoming increasingly multicast capable. Multicast routing establishes a **tree** that connects the source with the receivers. The multicast tree is rooted at the sender and the leaves are the receivers. Multicast delivery sends data across this tree towards the receivers. As opposed to unicast delivery, data is not copied at the source, but is copied inside the network at branch points of the multicast distribution tree. The fact that only a *single copy* of data is sent over a link that leads to multiple receivers results in a bandwidth gain of multicast over unicast whenever a sender needs to send simultaneously to multiple receivers. Given R receivers, the **multicast gain** for the network is defined as the ratio of unicast bandwidth cost to multicast bandwidth cost, where bandwidth cost is the product of the delivery cost of one packet on one link and the number of links the packet traverses from the sender to the R receivers for a particular transmission (unicast or multicast). In case of shortest path unicast and multicast routing between source and receivers, the multicast gain for the model of a full o-ary multicast tree is[1]:

$$log_o(R) \cdot \frac{R}{R-1} \cdot \frac{o-1}{o}$$

Even for random networks and multicast trees different from the idealized full o-ary tree, the multicast gain is largely determined by the logarithm of the number of receivers [61, 68].

Despite the widespread deployment of multicast capable networks, multicast is rarely provided as a service and network providers keep the multicast delivery option in their routers turned off. However, multicast results in bandwidth savings for the ISPs and allows the deployment of new services like audio/video broadcast. Several reasons contribute to the unavailability of multicast; multicast address allocation, security, network management, billing, lack of congestion control, lack of an incentive to use multicast are among the reasons that slow down the deployment of multicast (see [21] for a detailed discussion about the deployment issues for the IP multicast service). In this paper, we address how to increase the incentive to use multicast from the receiver's point of view. It could be argued that as multicast consumes less resources than unicast, a service using multicast should be charged less than the same service using unicast. However, as multicast is expensive to be deployed and probably more expensive to be managed (group management, pricing, security, etc.) than unicast, it is not clear whether a provider will charge less for multicast than for unicast. As discussed by Diot [21], multicast is only cost-effective for an ISP when it results in significant bandwidth savings. Indeed, as multicast is significantly more expensive than unicast, it is most of the time worthwhile to support small groups with unicast. We believe that the main incentive for a provider to use multicast is that multicast enables the deployment of new services that scale with a large number of receivers, for example audio and video broadcast.

[1] See section D.3.1 for some insights on the multicast gain and appendix D.7 for a rigorous proof of the results.

The problem of providing receivers with an incentive to use multicast is very difficult. In general, users want high satisfaction, but do not care whether the provider uses unicast or multicast to deliver the content. The argument that multicast allows applications to scale with a large number of receivers is not a good argument for a user because it does not change the user's satisfaction, except if the service cannot be provided without multicast due to a very large number of receivers. If we give more bandwidth to multicast, a multicast user will experience a higher satisfaction than a unicast user which results in an incentive to use multicast.

We saw that it is not easy to establish precisely who benefits how much from multicast. However, we saw that multicast allows to deploy new services. Therefore, it is very important to give a receiver-incentive to use multicast in order to give to the receivers an indisputable benefit to use multicast. We want to give an incentive to use multicast by rewarding the multicast gain in the network to the receivers; at the same time we want to treat[2] unicast traffic fairly relative to multicast traffic. The two motivations for increasing the bandwidth share for multicast compared to unicast are: First, to give a receiver-incentive to use multicast; Second, to favor multicast due to its significant bandwidth saving. We believe that the second point can be highly controversial. It does not seem fair to give the same amount of bandwidth to a flow serving one receiver and to another one serving ten millions receivers. However, the notion of fairness is subjective and debatable.

We investigate bandwidth allocation policies that allocate the bandwidth locally at each single link to unicast and multicast traffic, and we evaluate globally the bandwidth perceived by the receivers. For three different bandwidth allocation policies, we examine the case where a unicast network is augmented with a multicast delivery service and evaluate the receiver satisfaction and the fairness among receivers.

The rest of the paper is organized as follows. In Section D.2 we present the three bandwidth allocation strategies, and introduce the model and the assumptions for their comparison. In Section D.3 we give some insights into the multicast gain, and we analytically study the strategies for simple network topologies. In Section D.4 we show the effect of different bandwidth allocation policies on a hierarchical network topology. In Section D.5 we discuss the practical issues of our strategies, and Section D.6 concludes the paper.

[2]The problem of treating fairly unicast and multicast traffics is related to the more general question of how multicast flows should be treated in comparison to a unicast flow sharing the same bottleneck.

D.2 Model

D.2.1 Assumptions

We examine, in this paper, how to best allocate the bandwidth of a link between competing unicast and multicast traffic. We consider scenarios with a given number k of unicast sources, a given number m of multicast sources, a different number M of receivers per multicast source, and a different bandwidth C for each network link to be allocated among the source-destination(s) pairs.

For this study, we make several assumptions and simplifications. The assumptions are: i) Knowledge in every network node about every flow S_i through an outgoing link l. ii) Knowledge in every network node about the number of receivers $R(S_i, l)$ for flow S_i reached via an outgoing link l. iii) Each node is making the bandwidth allocation independently. A particular receiver sees the bandwidth that is the minimum bandwidth of all the bandwidth allocations on the links from the source to this receiver. iv) The sources have the capability to send through different bottlenecks via a cumulative layered transmission [55, 50]. For receivers of the same multicast delivery, the (bottleneck) bandwidth seen by different receivers may be different. In fact, each receiver sees the maximum available bandwidth on the path between the source and the receiver.

These assumptions are not restrictive in the sense that they do not simplify or limit the model. Indeed, i) and ii) are mandatory for per-flow bandwidth allocation with respect to the number of receivers. Weakening assumption ii) to require only, for instance, the knowledge in some network nodes about roughly the number of receivers per flow reached via an outgoing link, is a area for future research. Assumption iii) simply considers independent nodes, and iv) guarantees that the sources are able to get all the available bandwidth.

However, in order to make the evaluation of the model more tractable, we make two simplifications concerning the traffic: i) A constant bit rate traffic for every flow. ii) No arriving or departing flows. Simplification i) means that we do not consider the throughput variations of a flow, for instance, due to congestion control. Therefore, the sources immediately get all the available bandwidth. Simplification ii) means that we do not consider the dynamics of the flows, for instance, in the case of Web traffic (multiple arriving and departing flows to get a web page). As we consider a static scenario, the sources remain stable at the optimal rate. These simplifications are useful to eliminate all side effects and interferences due to dynamic scenarios. We do not claim our model to take into account the dynamics of the real Internet, but to provide a snapshot. At a given moment in time, we evaluate the impact of different bandwidth allocation policies for a given scenario. Adding dynamics to our model would not improve our study, but simply adds complexity in the evaluation of the bandwidth allocation policies. Indeed, the dynamics is not related to the bandwidth allocation policies, but to the ability of the sources to get

the available bandwidth. The impact of the dynamics of the flows on the bandwidth allocation policies is, however, an avenue for future research.

D.2.2 Bandwidth Allocation Strategies

We present three bandwidth allocation policies. It is important to us to employ the bandwidth-efficient multicast without starving unicast traffic and to give at the same time an incentive for receivers to connect via multicast, rather than via unicast. Our objective is twofold: On one hand we want to increase the average receiver satisfaction, on the other hand, we want to assure a fairness among different receivers.

We assume a network of nodes connected via links. At the beginning, we assume every network link l has a link bandwidth C_l. We compare three different strategies for allocating the link bandwidth C_l to the flows flowing across link l. Let n_l be the number of flows over a link l. Each of the flows originates at a source S_i, $i \in \{1, \ldots, n_l\}$. We say that a receiver r is **downstream** of link l if the data sent from the source to receiver r is transmitted across link l. Then, for a flow originating at source S_i, $R(S_i, l)$ denotes the **number of receivers that are downstream** of link l. For an allocation policy p, $B_p(S_i, l)$ denotes the bandwidth shared of link l allocated to the receivers of S_i that are downstream of l.

The three bandwidth allocation strategies for the bandwidth of a single link l are:

- **Receiver Independent (RI):** Bandwidth is allocated in equal shares among all flows through a link – independent of the number of receivers downstream. At a link l, each flow is allocated the share:

$$B_{RI}(S_i, l) = \frac{1}{n_l} C_l$$

  The motivation for this strategy is: the RI strategy does not represent any changes in the current bandwidth allocation policy. This allocation policy weighs multicast and unicast traffic equally. We consider this policy as the benchmark against which we compare the other two policies.

- **Linear Receiver Dependent (LinRD):** The share of bandwidth of link l allocated to a particular flow S_i depends linearly on the number of receivers $R(S_i, l)$ that are downstream of link l:

$$B_{LinRD}(S_i, l) = \frac{R(S_i, l)}{\sum_{j=1}^{n_l} R(S_j, l)} C_l$$

  The motivation for this strategy is: given R receivers for S_i downstream of link l, the absence of multicast forces the separate delivery to each of those R receivers via a separate

unicast flow[3]. For a multicast flow, we allocate a share that corresponds to the aggregate bandwidth of R separate unicast flows.

- **Logarithmic Receiver Dependent (LogRD):** The share of bandwidth of link l allocated to a particular stream S_i depends logarithmically on the number of receivers $R(S_i, l)$ that are downstream of link l:

$$B_{LogRD}(S_i, l) = \frac{1 + \ln R(S_i, l)}{\sum_{j=1}^{n_l}(1 + \ln R(S_j, l))} C_l$$

  The motivation for this strategy is: multicast receivers are rewarded with the multicast gain from the network. The bandwidth of link l allocated to a particular flow is, just like the multicast gain, logarithmic in the number of receivers that are downstream of link l:

Our three strategies are representatives of *classes* of strategies. We do not claim that the strategies we pick are the best representatives of each class. It is not the aim of this paper to find the best representative of a class, but to study the trends between the classes. One can define numerous classes of strategies. We do not claim that one of the three classes of strategies is optimal. However, we restrict ourselves to these three strategies as we believe these policies shed light on the fundamental issues that come with the introduction of the number of receivers in the bandwidth allocation.

The following example illustrates the bandwidth allocation for the case of the *Linear Receiver Dependent* policy. We have two multicast flows originating at S_1 and S_2 with three receivers each (see Fig. D.1).

For link 1, the available bandwidth C_1 is allocated as follows: Since $R(S_1, 1) = 3$ and $R(S_2, 1) = 3$, we get $B_{LinRD}(S_1, 1) = B_{LinRD}(S_2, 1) = \frac{3}{3+3} C_1 = 0.5C_1$. For link 4, we have $R(S_1, 4) = 2$ and $R(S_2, 4) = 1$. Therefore we get $B_{LinRD}(S_1, 4) = \frac{2}{3} C_4$ and $B_{LinRD}(S_2, 4) = \frac{1}{3} C_4$. Given these bandwidth allocations, the bandwidth seen by a particular receiver r is the bandwidth of the bottleneck link on the path from the source to r. For example, the bandwidth seen by receiver R_1^3 is $\min(\frac{1}{2}C_1, \frac{2}{3}C_4, \frac{1}{2}C_6)$.

The way we allocate bandwidth could lead to scenarios where bandwidth needs to be reallocated, we call this the *bandwidth reallocation problem*. Imagine three flows F_1, F_2, and F_3 with only one receiver each. The flows F_1 and F_2 share a link l_C of bandwidth C, and flows F_2 and F_3 share a link $l_{\frac{C}{2}}$ of bandwidth $\frac{C}{2}$. With any of the three policies, the bandwidth allocated on link l_C is $\frac{C}{2}$ for F_1 and F_2 and the bandwidth allocated on link $l_{\frac{C}{2}}$ is $\frac{C}{4}$ for F_2 and F_3. Therefore, F_2 cannot use its allocated bandwidth $\frac{C}{2}$ on link l_C. However, as we consider static scenarios with constant bit rate flows, the bandwidth that is not used by F_2 cannot be reallocated to F_1. This is the *bandwidth reallocation problem*. This problem could adversely impact the results of the

[3]We assume shortest path routing in the case of unicast and multicast.

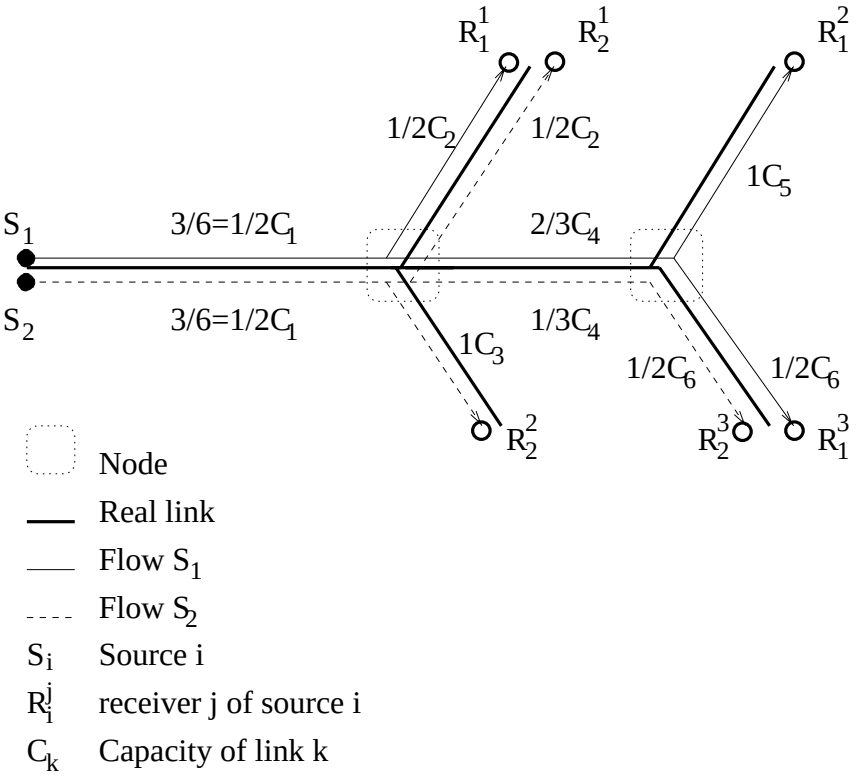


Figure D.1: Bandwidth allocation for linear receiver-dependent policy.

simulation. One way to solve this problem is to consider dynamic flows which grab the available bandwidth in case of unused bandwidth. This is contrary to the simplifications required by our model. Another way to solve this problem is to statically reallocate the unused bandwidth. However, in case of a complex topology, this leads to convergence problems that are beyond the scope of this paper. In fact, we decided to evaluate, for each simulation, the amount of unused bandwidth, and we found that there is very few unused bandwidth. Therefore, we do not expect the bandwidth reallocation problem to adversely impact the results of our simulations.

D.2.3 Criteria for Comparing the Strategies

Our goal is to increase the mean *receiver satisfaction*, however, not at the detriment of *fairness*. In order to evaluate *receiver satisfaction* and *fairness*, we define two basic measures, one describing the average user satisfaction, the other one describing the fairness among users.

Receiver Satisfaction

There are many ways to define receiver satisfaction and the most accurate is receiver utility. Unfortunately, utility is a theoretical notion that does not allow to compare the utility of two different receivers and does not give an absolute (i.e. for all receivers) scale of utility [26]. We

measure **receiver satisfaction** as the bandwidth an average receiver sees[4]. Let r be a receiver of a source S and let $(l_1, l_2, \ldots, l_L)$ be the path of L links from the source to r, then, the bandwidth seen by the receiver r is:

$$B_p^r = \min_{i=1,\ldots,L} \{B_p(S, l_i)\} \quad , \qquad p \in \{RI, LinRD, LogRD\}$$

With the total number of receivers R of all sources we define the **mean bandwidth** $\bar{B}_p$ as:

$$\bar{B}_p = \frac{1}{R} \sum_{r=1}^{R} B_p^r \quad , \qquad p \in \{RI, LinRD, LogRD\} \tag{D.1}$$

Jiang et al. [41] introduced a global measure for the throughput delivered via the whole network that is defined as the sum of the mean throughput over all the flows. For the global throughput measure, it is possible to weight multicast flows with a factor R^α, where R is the number of receivers and $0 < \alpha < 1$. To the best of the authors knowledge, the approach of Jiang et al. [41] is the only one taking into account the number of receivers of a multicast flow. While their approach takes into account the number of receivers to measure the global network throughput, our approach is different in two aspects: First, we take the number of receivers into account for the *allocation* of the bandwidth on links and use a policy ($LogRD$) that weights multicast flows in the allocation with the logarithm of the number of receivers. Second, we measure receiver satisfaction with respect to all receivers, not just the ones of a single group.

Fairness

For inter-receiver fairness, several measures exist, including the product measure [6] and the fairness index [40]. For a discussion of the different measures see [28].

Jiang et al. [41] defined inter-receiver fairness for a single multicast flow as the sum of the receiver's utilities, where utility is highest around the fair share. Due to the intricacies coming with the utility function, we do not consider a utility function and use a fairness measure that takes into account all receivers of all flows.

We use the standard deviation of the bandwidth among receivers to be the measure of choice for inter-receiver fairness.

$$\sigma = \sqrt{\frac{1}{R} \sum_{r=1}^{R} (\bar{B}_p - B_p^r)^2} \quad , \qquad p \in \{RI, LinRD, LogRD\} \tag{D.2}$$

The key point with this fairness measure is that we consider a notion of fairness independent of the network and of the localization of the bottlenecks. Indeed, each receiver has a given satisfaction. The feeling of fairness for each receiver only depends on the satisfaction of the

[4]While there are other criteria to measure satisfaction such as delay or jitter, bandwidth is a measure of interest to the largest number of applications.

other receivers, but is independent of any network parameters. For instance, if a receiver has a satisfaction lower than all the other receivers, he will feel a high unfairness even if his low satisfaction is due to a slow modem.

We define **ideal fairness** as the case where all receivers receive the same bandwidth. For **ideal fairness** our measure $\sigma = 0$ has its lowest value. In all other cases, the bandwidth sharing among receivers is unfair and $\sigma > 0$.

Optimality

The question now is how to optimize both *receiver satisfaction* and *fairness*. For the strategy p and the scenario s, let $\sigma(p, s)$ be the function that defines our fairness criteria and $\bar{B}(p, s)$ be the function that defines our receiver satisfaction. An accurate definition of s is: $s + p$ defines the full knowledge of all parameters that have an influence on receiver satisfaction and fairness. So s defines all the parameters without the strategy p. We define

$$\sigma_{max}(s) = \min_p \sigma(p, s)$$

and

$$\bar{B}_{max}(s) = \max_p \bar{B}(p, s)$$

We want to find a function $F(s)$ such as $\forall\ s$: $\sigma(F(s), s) = \sigma_{max}(s)$ and $\forall\ s$: $\bar{B}(F(s), s) = \bar{B}_{max}(s)$. If such a function $F(s)$ exists for all s, it means that there exists a pair $(F(s), s)$ that defines for all s an optimal point for both *receiver satisfaction* and *fairness*. Feldman [26] shows that *receiver satisfaction* is inconsistent with *fairness*[5], which means it is impossible to find such a function $F(s)$ that defines an optimal point for both *receiver satisfaction* and *fairness* for all s. So we cannot give a general mathematical criteria to decide which bandwidth allocation strategy is the best. Moreover, in most of the cases it is impossible to find an optimal point for both $\bar{B}$ and σ.

Therefore, we evaluate the allocation policies with respect to the tradeoff between *receiver satisfaction* and *fairness*. Of course, we can define criteria that can apply in our scenarios, for instance, strategy A is better than strategy B if $\frac{\sigma_A}{\sigma_B} \leq L_f$ and $\frac{\bar{B}_A}{\bar{B}_B} \geq I_s$ where L_f is the maximum loss of *fairness* accepted for strategy A and I_s is the minimum increase of *receiver satisfaction* for strategy A. But, the choice of L_f and I_s needs a fine tuning and seems pretty artificial to us.

Receiver satisfaction and fairness are criteria for comparison that are meaningful only in the same experiment. It does not make sense to compare the satisfaction and the fairness among different sets of users. Moreover, it is impossible to define an absolute level in satisfaction and fairness. In particular, it is not trivial to decide whether a certain increase in satisfaction

[5]In terms of mathematical economics we can say that Pareto optimality is inconsistent with fairness criteria [26].

is worthwhile when it comes at the price of a decrease in fairness. Hopefully, for our study the behavior of the three strategies will be different enough to define distinct operating points. Therefore, the evaluation of the tradeoff between *receiver satisfaction* and *fairness* does not pose any problem.

D.3 Analytical Study

We first give some insights into the multicast gain and the global impact of a local bandwidth allocation policy. A rigorous discussion of both points is given in appendix D.7 and appendix D.8. Then, we compare the three bandwidth allocation policies from Section D.2 for basic network topologies in order to gain some insights in their behavior. In Section D.4 we study the policies for a hierarchical network topology.

D.3.1 Insights on Multicast Gain

We can define the multicast gain in multiple ways and each definition may capture very different elements. We restrict ourselves to the case of a full o-ary distribution tree with either receivers at the leaves – in this case we model a point-to-point network – or with broadcast LANs at the leaves. We consider one case where the unicast and the multicast cost only depends on the number of links (the unlimited bandwidth case) and another case where the unicast and the multicast cost depends on the bandwidth used (the limited bandwidth case).

We define the **bandwidth cost** as the sum of all the bandwidths consumed on all the links of the tree. We define the **link cost** as the sum of all the links used on the tree; we count the same link n times when the same data are sent n times on this link. Let C_U be the unicast bandwidth/link cost from the sender to all of the receivers and C_M the multicast bandwidth/link cost from the same sender to the same receivers.

For the bandwidth-unlimited case, every link of the tree has unlimited bandwidth. Let C_U and C_M be the link cost for unicast and multicast, respectively. We define the multicast gain as the ratio $\frac{C_U}{C_M}$. If we consider one receiver on each leaf of the tree, the multicast gain depends logarithmically on the number of receivers. If we consider one LAN on each leaf of the tree, the multicast gain depends logarithmically on the number of LANs and linearly on the number of receivers per LAN (see appendix D.7.1 for more details).

For the bandwidth-limited case, every link of the tree has a capacity C. Let C_U and C_M be the bandwidth cost for unicast and multicast, respectively. Unfortunately, for the bandwidth-limited case, the multicast gain defined as $\frac{C_U}{C_M}$ makes no sense because it is smaller than 1 for a large number of multicast receivers (see appendix D.7.2 for more details). We define another measure that combines the satisfaction and the cost that we call cost per satisfaction $GB =$

$\frac{\text{global cost}}{\text{global satisfaction}}$, that tells us how much bandwidth we invest to get a unit of satisfaction. Now, we define the multicast gain as $\frac{GB_U}{GB_M}$ where GB_U and GB_M are the unicast and multicast cost per satisfaction, respectively. If we consider one receiver on each leaf of the tree, the gain depends logarithmically on the number of receivers. If we consider one LAN on each leaf of the multicast tree, the gain depends logarithmically on the number of LANs and linearly on the number of receivers per LAN (see appendix D.7.2 for more details).

In conclusion, for both the bandwidth unlimited and limited case, the multicast gain has a logarithmic trend with the number of receivers in case of point-to-point networks. The multicast gain has also a logarithmic trend with the number of LANs, but a linear trend with the number of receivers per LAN. Therefore, with a small number of receivers per LANs the multicast gain is logarithmic but with a large number of receivers per LANs the multicast gain is linear. Appendix D.7 gives an analytical proof of these results.

D.3.2 Insights on the Global Impact of a Local Bandwidth Allocation Policy

In section D.2.2, we suggest the $LogRD$ policy because we want to reward the multicast receivers with the multicast gain. However, it is not clear whether allocating locally the bandwidth as a logarithmic function of the number of downstream receivers achieves to reward the multicast receivers with the multicast gain, which is a global notion.

To clarify this point, we consider a full o-ary tree for the bandwidth-unlimited case when there is one receiver per leaf. We find (see appendix D.8 for a proof) that the policy that rewards multicast with its gain is the $LinRD$ policy and not the $LogRD$ policy as expected. If we reward multicast with its real gain using the $LinRD$ policy, we will give to multicast the bandwidth that corresponds to the aggregate bandwidth of R separate unicast flows (see section D.2.2). However, we have to consider that we use multicast in order to save bandwidth. If we allocate to a multicast flow the same bandwidth than the bandwidth used by R separate unicast flows, the use of multicast makes no sense as it does not save bandwidth compared to unicast. Therefore, rewarding a multicast flow with its gain (as defined in appendix D.7) makes no sense.

In the following, we will see that the $LinRD$ is a very aggressive policy for unicast flows while the $LogRD$ policy gives very good results for both the unicast and multicast flows.

D.3.3 Comparison of the Bandwidth Allocation Policies

D.3.3.1 Star Topology

We consider the case where k unicast flows need to share the link bandwidth C with a single multicast flow with m downstream receivers, see Fig. D.2.

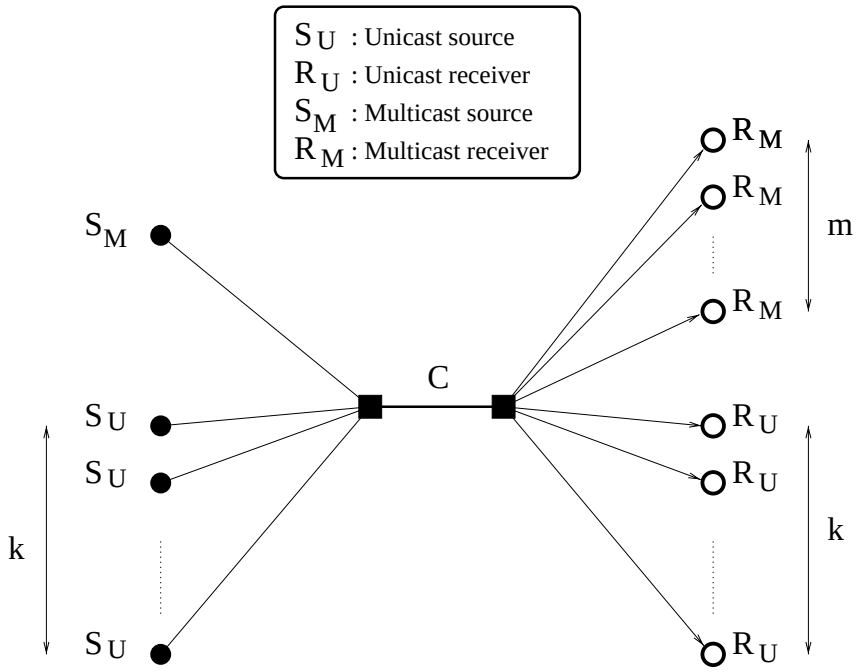


Figure D.2: One multicast flow and k unicast flows over a single link.

With the RI strategy, the bandwidth share of a link is $\frac{1}{k+1}C$ for both a unicast and a multicast flow. The $LinRD$ strategy gives a share of $\frac{1}{m+k}C$ to each unicast flow and a share of $\frac{m}{m+k}C$ to the multicast flow. The $LogRD$ strategy results in a bandwidth of $\frac{1}{k+(1+\ln m)}C$ for a unicast flow and $\frac{1+\ln m}{k+(1+\ln m)}C$ for the multicast flow.

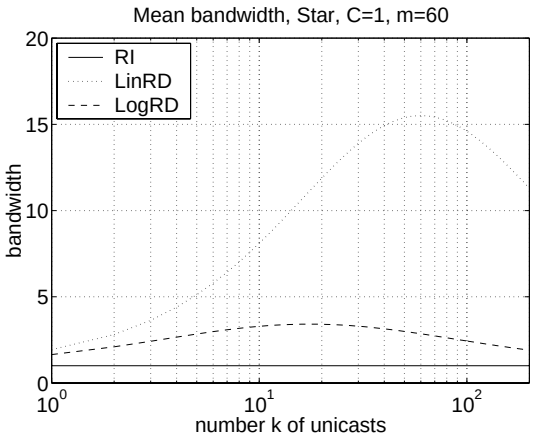
The mean receiver bandwidths over all receivers (unicast and multicast) for the three policies are:

$$\begin{aligned}
\bar{B}_{RI} &= \frac{1}{k+m}\sum_{i=1}^{k+m}\frac{C}{k+1} = \frac{C}{k+1} \\
\bar{B}_{LinRD} &= \frac{1}{k+m}\left(\sum_{i=1}^{k}\frac{C}{m+k} + \sum_{i=1}^{m}\frac{mC}{m+k}\right) = \frac{k+m^2}{(k+m)^2}C \\
\bar{B}_{LogRD} &= \frac{1}{k+m}\left(\sum_{i=1}^{k}\frac{C}{k+(1+\ln m)} + \sum_{i=1}^{m}\frac{C(1+\ln m)}{k+(1+\ln m)}\right) = \frac{k+m(1+\ln m)}{(k+m)(k+1+\ln m)}C
\end{aligned}$$

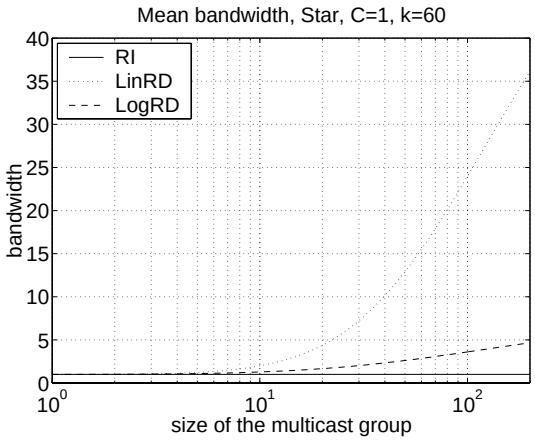
By comparing the equations for any number of multicast receivers, $m > 1$, and any number of unicast flows $k > 1$ we obtain:

$$\bar{B}_{LinRD} > \bar{B}_{LogRD} > \bar{B}_{RI} \tag{D.3}$$

The receiver-dependent bandwidth allocation strategies, $LinRD$ and $LogRD$, outperform the receiver-independent strategy RI by providing a higher bandwidth to an average receiver.



(a) Increasing the number k of unicasts; 60 multicast receivers.



(b) Increasing the size m of the multicast group; 60 unicasts.

Figure D.3: Normalized mean bandwidth for the Star topology.

This is shown in Fig. D.3, where the mean bandwidths are normalized by $\bar{B}_{RI}$, in which case the values depicted express the bandwidth gain of any policy over RI.

Fig. D.3(a) shows the mean bandwidth for $m = 60$ multicast receivers and an increasing number of unicasts $k = 1, \cdots, 200$. The receiver-dependent policies $LinRD$ and $LogRD$ show an increase in the mean bandwidth when the number of unicasts is small compared to the number of multicast receivers. The increase with the $LogRD$ policy is less significant than the increase with the $LinRD$ policy since the $LogRD$ policy gives less bandwidth to the multicast flow than the $LinRD$ policy for the same number of receivers. Additionally, more link bandwidth is allocated to the multicast flow than in the case of a higher number of unicasts, which result in a lower share for multicast. With an increasing number of unicasts, the gain of $LinRD$ and $LogRD$ decreases.

After assessing the bandwidth gain of $LinRD$ and $LogRD$ for a number of unicast receivers higher than the number of multicast receivers, we turn our attention to the case where the number of multicast receivers is increasing $m = 1, \cdots, 200$ and becomes much higher than the number of unicasts ($k = 60$). Fig. D.3(b) shows that the mean bandwidth for $LinRD$ and $LogRD$ is increasing to multiples of the bandwidth of RI.

We saw that the receiver-dependent policies significantly reward multicast receivers and that the $LinRD$ policy is better than the $LogRD$ policy with respect to the receiver satisfaction. Now, we have to study the impact of the receiver-dependent policies on the fairness.

The following equations give the standard deviation over all receivers for the three policies:

$$\sigma_{RI} \quad = \quad 0$$

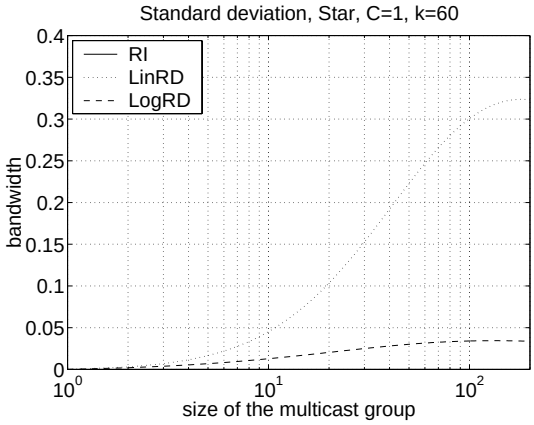


Figure D.4: Standard deviation for the Star topology. Increasing the size $m = 1, ..., 200$ of the multicast group; $k = 60$ unicasts.

$$\sigma_{LinRD} = C(m-1)\sqrt{\frac{k \cdot m}{(k+m)^3(k+m-1)}}$$

$$\sigma_{LogRD} = \frac{C \cdot \ln m}{k+1+\ln m}\sqrt{\frac{k \cdot m}{(k+m)(k+m-1)}}$$

By comparing the equations for any number of multicast receivers, $m > 1$, and any number of unicast flows $k > 1$ we obtain:

$$\sigma_{LinRD} > \sigma_{LogRD} > \sigma_{RI} \tag{D.4}$$

While the $LinRD$ is the best policy among our three policies with respect to the receiver satisfaction, it is the worst policy in terms of fairness. Fig. D.4 shows the standard deviation for $k = 60$ unicast flows and an increasing multicast group $m = 1, ..., 200$. With the Star topology, all unicast receivers see the same bandwidth and all multicast receivers see the same bandwidth. Between unicast receivers and multicast receivers no difference exists for the RI strategy. For the $LinRD$ strategy a multicast receiver receives m times more bandwidth than a unicast receiver and for the $LogRD$ strategy a multicast receiver receives $(1 + \ln m)$ times more bandwidth than a unicast receiver. The standard deviation for all the receivers is slightly increased with the $LogRD$ policy compared to the RI policy, and is more significantly increased with the $LinRD$ policy compared to the RI policy (see Fig. D.4).

The high bandwidth gains of the $LinRD$ strategy result in a high unfairness for the unicast receivers. For $LogRD$, the repartitioning of the link bandwidth between unicast and multicast receivers is less unequal than in the case of $LinRD$. In summary, the $LogRD$ policy leads to a significant increase in receiver satisfaction, while it introduces only a small decrease in fairness. We can conclude that among the three strategies $LogRD$ makes the best tradeoff between receiver satisfaction and fairness.

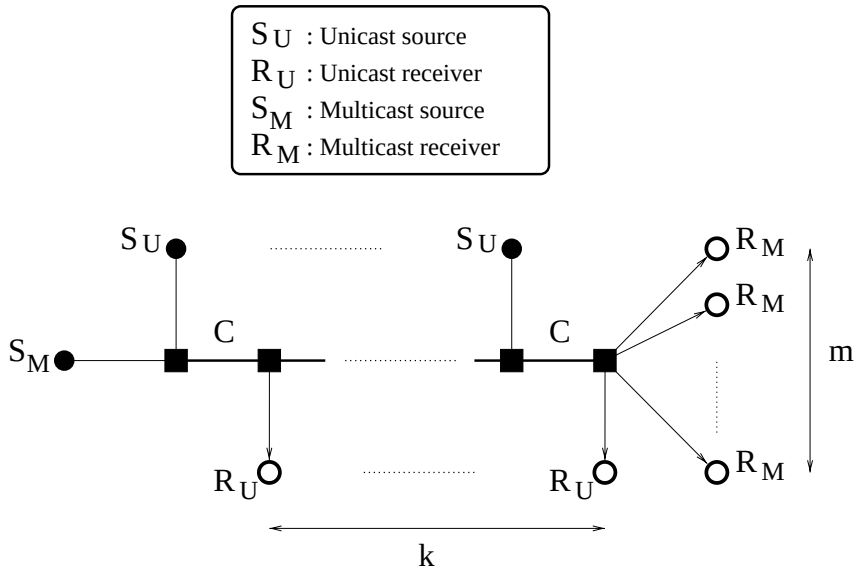


Figure D.5: One multicast flow and k unicast flows over a chain of links.

Surprisingly we will obtain nearly the same results in Section D.4.3 when we examine the three policies on a large random network. The similarity of the Fig. D.3(b), and D.4, with the figures of Section D.4.3 indicates that the simple Star topology with a single shared link can serve as a model for large networks.

D.3.3.2 Chain Topology

We now study bandwidth allocation for the case where a multicast flow is traversing a unicast environment of several links. We use a chain topology, as shown in Fig. D.5, where k unicast flows need to share the bandwidth with a single multicast flow leading to m receivers. However, the unicast flows do not share bandwidth among each other, as opposed to the previous single shared link case for the star topology.

At each link, the RI strategy allocates in $\frac{1}{2}C$ for both the unicast flow and the multicast flow. The $LinRD$ strategy results in a share of $\frac{1}{m+1}C$ for the unicast flow and $\frac{m}{m+1}C$ for the multicast flow. The $LogRD$ strategy results in a share of $\frac{1}{2+\ln m}C$ for the unicast flow and a share of $\frac{1+\ln m}{2+\ln m}C$ for the multicast flow.

The mean receiver bandwidth for the three cases is:

$$\begin{aligned} \bar{B}_{RI} &= \frac{1}{k+m}\sum_{i=1}^{k+m}\frac{C}{2} = \frac{C}{2} \\ \bar{B}_{LinRD} &= \frac{1}{k+m}\left(\sum_{i=1}^{k}\frac{C}{m+1} + \sum_{i=1}^{m}\frac{m\cdot C}{m+1}\right) = \frac{k+m^2}{(k+m)(m+1)}C \end{aligned}$$

$$\bar{B}_{LogRD} = \frac{1}{k+m}\left(\sum_{i=1}^{k}\frac{C}{2+\ln m}+\sum_{i=1}^{m}\frac{C(1+\ln m)}{2+\ln m}\right)=\frac{k+m+m\cdot\ln m}{(k+m)(2+\ln m)}C$$

The strategy with the highest mean bandwidth depends on the relation between the number of multicast receivers and the number of unicast flows. If the number of unicasts equals the number of multicast receivers, $k = m$, then all policies result in the same average receiver bandwidth of $C/2$. For all other cases, with $k > 1$ and $m > 1$ we have:

$$\begin{aligned}\bar{B}_{RI} > \bar{B}_{LogRD} > \bar{B}_{LinRD} \quad , \qquad k > m \\ \bar{B}_{LinRD} > \bar{B}_{LogRD} > \bar{B}_{RI} \quad , \qquad k < m\end{aligned} \tag{D.5}$$

The receiver-dependent policies $LinRD$ and $LogRD$ perform better than the RI policy when the size of the multicast group is larger than the number of unicast sessions. While the number of multicast receivers can increase to large numbers and is only limited by the number of hosts in the network, the number of unicast crossing traffic is limited by the length of the path source-receiver. This is shown in Fig. D.6, where the mean bandwidths are normalized by $\bar{B}_{RI}$, in which case the values depicted express the bandwidth gain of any policy over RI.

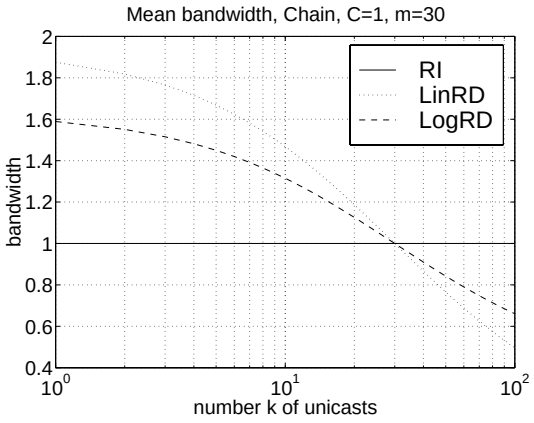
Fig. D.6(a) shows the mean bandwidth for $m = 30$ multicast receivers and an increasing number of unicast sessions $k = 1, ..., 200$. As the number of unicasts increases, receiver-dependent policies become worse than RI policy. Fig. D.6(b) shows the mean bandwidth for $k = 30$ unicast receivers and an increasing number of multicast receivers. The receiver-dependent policies perform worse than the RI policy for small multicast group sizes, but as the size of the multicast group increases the bandwidth gain for receiver dependent policies increases rapidly. In Fig. D.6(b), for the multicast group size $m = 30$, the three policies lead to the same mean bandwidth, for the multicast group size $m = 50$, the $LinRD$ policy yields to more than 20% gain over the RI policy and the $LogRD$ policy yields to more than 15% gain over the RI policy.

We see that, concerning the receiver satisfaction, the receiver-dependent policies have a more complex behavior with a chain topology than with a star topology. To complete the study of the chain topology, we look at the fairness.

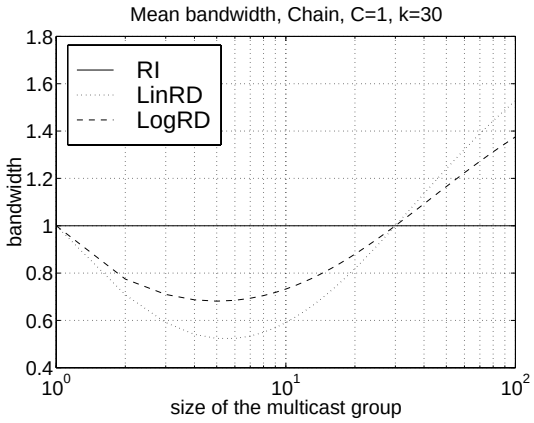
The standard deviation over all the receivers for the three policies is:

$$\begin{aligned}\sigma_{RI} &= 0 \\ \sigma_{LinRD} &= \frac{C(m-1)}{m+1}\sqrt{\frac{k\cdot m}{(k+m)(k+m-1)}} \\ \sigma_{LogRD} &= \frac{C\cdot\ln m}{2+\ln m}\sqrt{\frac{k\cdot m}{k+m}k+m-1}\end{aligned}$$

By comparing the equations for any number of multicast receivers, $m > 1$, and any number



(a) Increasing the number k of unicasts, 10 multicast receivers.



(b) Increasing the size m of the multicast group, 10 unicasts.

Figure D.6: Normalized mean bandwidth for the Chain topology.

of unicast flows $k > 1$ we obtain:

$$\sigma_{LinRD} > \sigma_{LogRD} > \sigma_{RI} \tag{D.6}$$

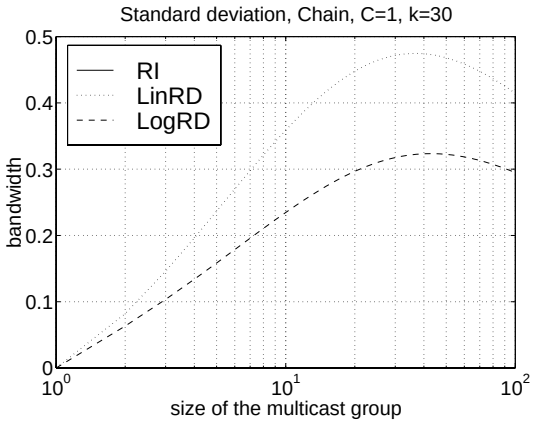


Figure D.7: Standard deviation for the Chain topology as a function of the size m of the multicast group for $k = 30$ unicasts.

The $LinRD$ policy, as for the star topology, has to the worst fairness. Fig. D.7 shows the standard deviation for $k = 30$ unicast flows and an increasing multicast group $m = 1, ..., 200$. For RI, unicast receivers and multicast receivers obtain the same share, for $LinRD$ a multicast receiver receives m times more bandwidth than a unicast receiver and for $LogRD$ a multicast receiver receives $(1 + \ln m)$ times more bandwidth than a unicast receiver. As the multicast session size m increases, the unicast flows get less bandwidth under the $LinRD$ and the $LogRD$

strategy, while the RI strategy gives the same bandwidth to unicast and multicast receivers. The $LinRD$ policy leads to a worse fairness than the $LogRD$ policy, however, the gap between the two policies is smaller that with the *Star* topology (compare Fig. D.7 and Fig. D.4).

We conclude that among the three strategies the $LogRD$ strategy achieves for large group sizes the best compromise between receiver satisfaction and fairness. However, for the *Chain* topology the superiority of the $LogRD$ policy is not as obvious as for the *Star* topology.

This simple analytical study allowed to identify some principal trends in the allocation behavior of the three strategies studied. The $LogRD$ policy seems to be the best compromise between receiver satisfaction and fairness. To deepen the insight gained with our analytical study, we will study the three strategies via simulation on a large hierarchical topology.

D.4 Simulation

We now examine the allocation strategies on network topologies that are richer in connectivity. The generation of realistic network topologies is subject of active research [9, 23, 90, 91]. It is commonly agreed that hierarchical topologies better represent a real Internetwork than do flat topologies. We use `tiers` [23] to create hierarchical topologies consisting of three levels: WAN, MAN, and LAN that aim to model the structure of the Internet topology [23]. For details about the network generation with `tiers` and the used parameters the reader is referred to Appendix D.9.

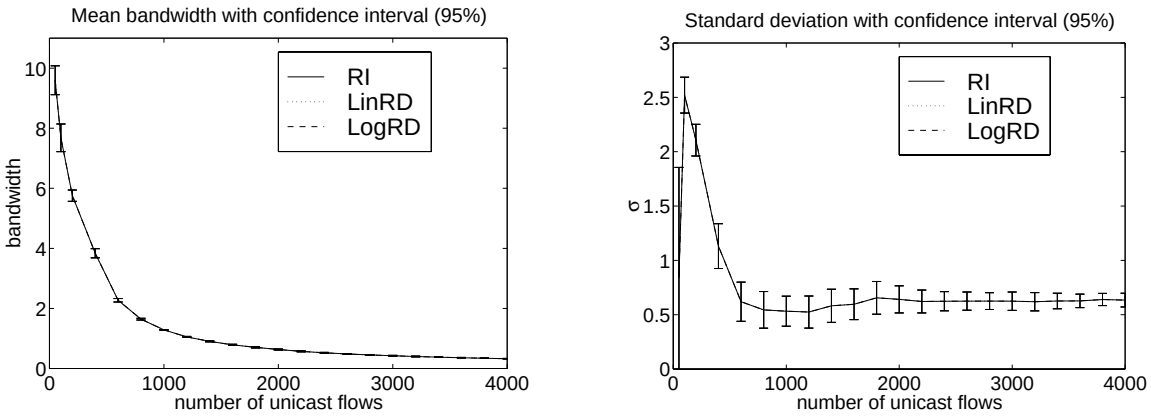
D.4.1 Unicast Flows Only

Our first simulation aims to determine the right number of unicast flows to define a meaningful unicast environment. We start with our random topology *RT* and add at random locations of the LAN-leaves unicast senders and unicast receivers. The number of unicast flows ranges from 50 to 4000. Each simulation is repeated five times and averages are taken over the five repetitions. We compute for each plot 95% confidence intervals.

First of all, we see in Fig. D.8 that the 3 allocation policies give the same allocation. Indeed, there are only unicast flows and the differences of behavior between the policies depend only on the number of receivers downstream a link for a flow, which is always one in this example.

Secondly, the mean bandwidth (Fig. D.8(a)) decreases as the number of unicast flows increases. An added unicast flows decreases the average share. For instance, if we take one link of capacity C shared by all unicast flows, k unicast flows on that link obtain a bandwidth of $\frac{C}{k}$ each.

We plot the standard deviation in Fig. D.8(b). For a small number of unicast flows, we have high standard deviation. Since there are few unicast flows with respect to the network size,



(a) Mean bandwidth.

(b) Standard deviation.

Figure D.8: Mean bandwidth (Mbit/s) and standard deviation of all receivers for an increasing number of unicast flows, $k = [50, ..., 4000]$.

the random locations of the unicast hosts have a great impact on the bandwidth allocated. The number of LANs in our topology is 180. So, 180 unicast flows lead on average to one receiver per LAN. A number of unicast flows chosen too small for a large network results in links shared only by a small number of flows. Hence, the statistical measure becomes meaningless. When the network is lightly loaded adding one flow can heavily change the bandwidth allocated to other flows, and we observe a large heterogeneity in the bandwidth allocated to the different receivers. On the other hand, for 1800 unicast flows, the mean number of receivers per LAN is 10, so the heterogeneity due to the random distribution of the pairs sender-receiver does not lead to high standard deviation. According to Fig. D.8(b), we chose our unicast environment with 2000 unicast flows to obtain a low bias due to the random location of the sender-receiver pairs.

D.4.2 Simulation Setup

For our simulations we proceed as follows.

- 2000 unicast sources and 2000 unicast receivers are chosen at random locations among the hosts.

- One multicast source and $1, \cdots, 6000$ receivers are chosen at random locations. Depending on the experiment, this may be repeated several times to obtain several multicast trees, each with a single source and the same number of receivers.

- We use shortest path routing [15] through the network to connect the 2000 unicast source-receiver pairs and to build the source-receivers multicast tree [22]. As routing metric, the length of the link as generated by `tiers` is used.

- For every network link, the number of flows across that link is calculated. By tracing back the paths from the receivers to the source, the number of receivers downstream is determined for each flow on every link.

- At each link using the information about the number of flows and the number of receivers downstream, the bandwidth for each flow traversing that link is allocated via one of the three strategies: RI, $LinRD$, and $LogRD$.

- In order to determine the bandwidth seen by a receiver r, the minimum bandwidth allocated to a flow on all the links along the path from source to receiver is taken as the bandwidth B_p^r seen by r for strategy p (see section D.2.3).

The result of the simulation gives the mean bandwidth $\bar{B}_p$ for the three bandwidth allocation strategies. We conduct different experiments with a single and with multiple multicast groups.

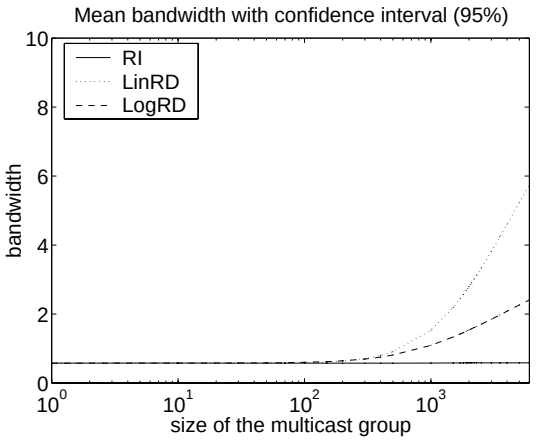
D.4.3 Single Multicast Group

For this experiment, we add one multicast group to the 2000 unicast flows. The size of the multicast group varies from 1 up to 6000 receivers. There are 70 hosts on each LAN and the number of potential senders/receivers is therefore 12600. This experiment shows the impact of the group size on the bandwidth allocated to the receivers under the three allocation strategies. This simulation is repeated five times and averages are taken over the five repetitions.

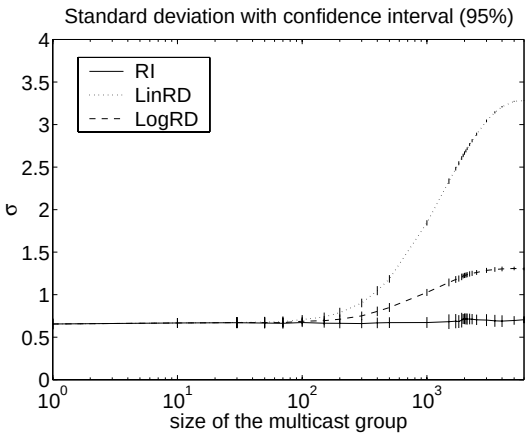
We simulate small groups sizes ($m = [1, ..., 100]$), then large groups sizes ($m = [100, ..., 3000]$), and finally evaluate the asymptotic behavior of our policies ($m = [3000, ..., 6000]$). The asymptotic case does not aim to model a real scenario, but gives an indication about the behavior of our policies in extreme cases. While 6000 multicast receivers seems a lot compared to the 2000 unicast flows, this case gives a good indication about the robustness of the policies. We display the results with a logarithmic x-axis.

Fig. D.9(a) shows that the average user receives more bandwidth when the allocation depends on the number of receivers. A significant difference between the allocation strategies appears for a group size m greater than 100. For small group sizes, unicast flows determine the mean bandwidth due to the high amount of unicast receivers compared to multicast receivers. We claim that receiver-dependent policies increase receiver satisfaction.

A more accurate analysis needs to distinguish between unicast and multicast receivers. Multicast receivers are rewarded with a higher bandwidth than unicast receivers for using mul-

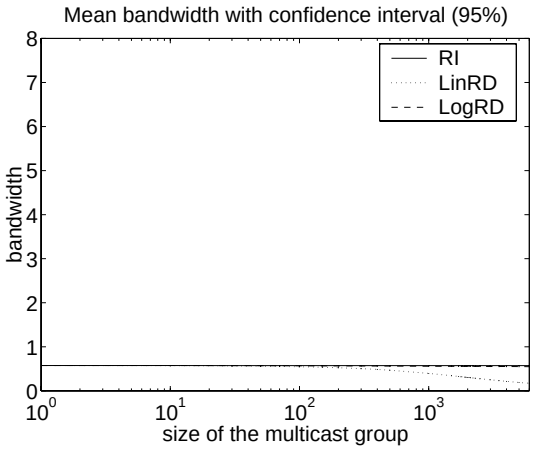


(a) Mean bandwidth.

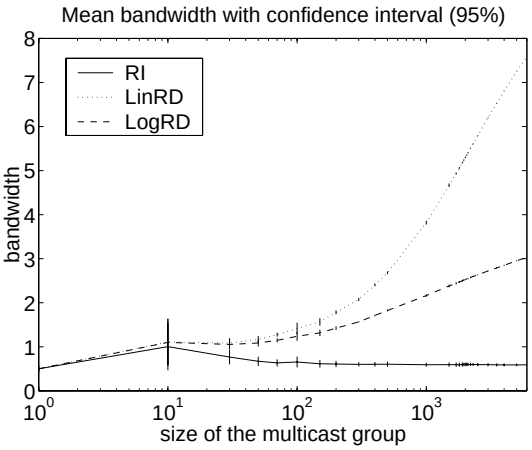


(b) Standard deviation.

Figure D.9: Mean bandwidth (Mbit/s) and standard deviation of all receivers for an increasing multicast group size $m = [1, ..., 6000]$, $k = 2000$, $M = 1$.



(a) Unicast receivers.

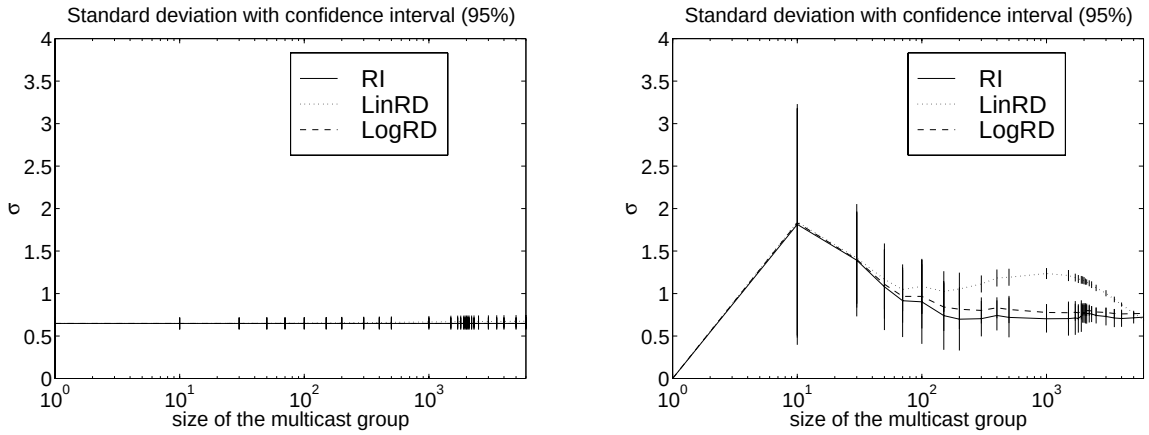


(b) Multicast receivers.

Figure D.10: Mean bandwidth (Mbit/s) of unicast and multicast receivers with confidence interval (95%) for an increasing multicast group size $m = [1, ..., 6000]$, $k = 2000$, $M = 1$.

ticast as the comparison between Fig. D.10(a) and Fig. D.10(b) shows. This is not surprising as our policies reward using multicast. Moreover, the increase in bandwidth allocated to multicast receivers leads to a significant decrease of bandwidth available for unicast receivers for the $LinRD$ policy, while the decrease of bandwidth is negligible for the $LogRD$ policy (Fig. D.10(a)) even in the asymptotic case. In conclusion, the $LogRD$ policy is the only policy among the three policies that leads to a significant increase of receiver satisfaction for the

average multicast receiver without affecting the receiver satisfaction for the average unicast receiver.



(a) Unicast receivers.

(b) Multicast receivers.

Figure D.11: Standard deviation of unicast and multicast receivers with confidence interval (95%) for an increasing multicast group size $m = [1, ..., 6000]$, $k = 2000$, $M = 1$.

The standard deviation for the average user increases with the size of the multicast group for the receiver-dependent policies (Fig. D.9(b)). This unfairness is caused by the difference of the lower bandwidth allocated to the unicast flows compared to the higher bandwidth given to the a multicast flow (Fig. D.10(a) and D.10(b)). For $LinRD$ and $LogRD$, σ tends to flatten for large group sizes, since the multicast receivers determine, due to their large number, the standard deviation. The standard deviation for unicast receivers (Fig. D.11(a)) is independent of the multicast group size and of the policies. For a small increasing group size, fairness first becomes worse among multicast receivers, as indicated by the increasing standard deviation in Fig. D.11(b), since the sparse multicast receiver setting results in a high heterogeneity of the allocated bandwidth. As the group size increases further, multicast flows are allocated more bandwidth due to an increasing number of receivers downstream. Therefore, the standard deviation decreases with the number of receivers. In the asymptotic part, the standard deviation for the $LinRD$ policy decreases faster than for the $LogRD$ policy since as the number of receivers increases, the amount of bandwidth allocated to the multicast flow approaches the maximum bandwidth (the bandwidth of a LAN), see Fig. D.10(b). Therefore, all the receivers see a high bandwidth near the maximum, which leads to low standard deviation. Another interesting observation is that the multicast receivers among each other have a higher heterogeneity in the bandwidth received than have the unicast receivers, compare Fig. D.11(a) and Fig. D.11(b). A few bottlenecks are sufficient to split the multicast receivers in large subgroups with significant differences in bandwidth allocation that subsequently result in a higher standard deviation. For

the 2000 unicast receivers, the same bottlenecks affect only a few receivers.

The standard deviation taken over all the receivers hides the worst case performance experienced by any individual receiver. To complete our study, we measure the minimum bandwidth, which gives an indication about the worst case behavior seen by any receiver. The minimum bandwidth over all the receivers is dictated by the minimum bandwidth over the *unicast* receivers (we give only one plot, Fig. D.12(a)). As the size of the multicast group increases, the minimum bandwidth seen by the unicast receivers dramatically decreases for the $LinRD$ policy, whereas the minimum bandwidth for the $LogRD$ policy remains close to the one for the RI policy even in the asymptotic part of the curve. We can point out another interesting result: the minimum bandwidth for the RI policy stays constant even for very large group sizes; the $LinRD$ policy that simulates the bandwidth that would be allocated if we replace the multicast flow by an equivalent number of unicast flows, results in a minimum bandwidth the rapidly decreases toward zero. Therefore, we note the positive impact of multicast on the bandwidth allocated, and multicast *greatly improves the worst case bandwidth allocation*. We see in Fig. D.12(b) that the minimum bandwidth increases for *multicast* receivers with the size of the multicast group for the receiver-dependent policies. In conclusion, the $LinRD$ policy leads to important degradation of the fairness when the multicast group size increases, whereas the $LogRD$ policy always remains close to the RI policy.

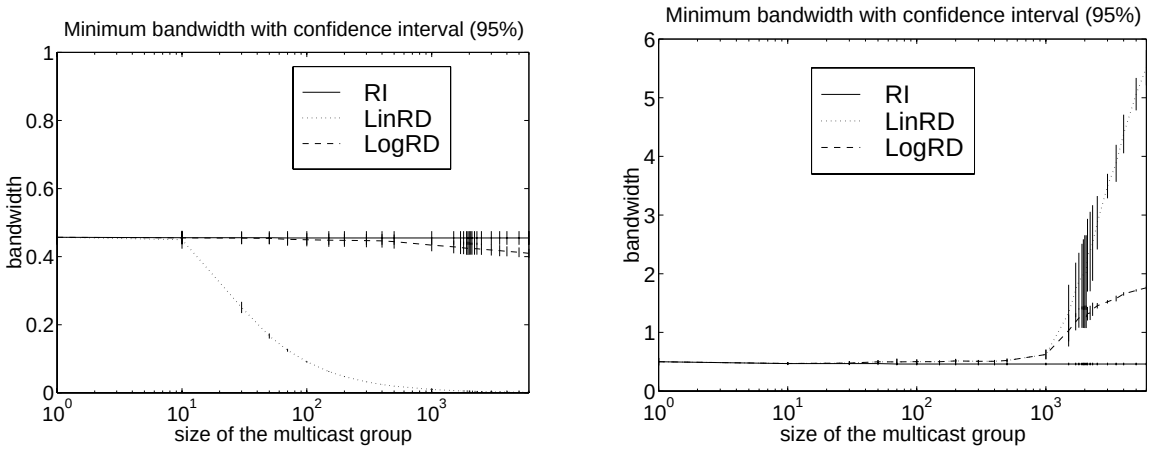
For the RI policy, we see that the increase in the multicast group size does not influence the average user satisfaction (Fig. D.9(a)), nor the fairness among different receivers (Fig. D.9(b)). Also, the difference between unicast and multicast receivers is minor concerning the bandwidth both received (Fig. D.10(a) and D.10(b)), and the unfairness (Fig. D.11(a) and D.11(b)). The $LogRD$ policy is the only policy among our policies that significantly increases receiver satisfaction (Fig. D.9(a)), keeps fairness close to the one of the RI policy (Fig. D.9(b)), and does not starve unicast flows, even in asymptotic cases (Fig. D.12(a)).

Finally, one also should note the similarity between Fig. D.9(a), D.9(b) obtained by simulation for a large network and Fig. D.3(b), D.4 obtained by analysis of the star topology. This suggests that the star topology is a good model to study the impact of the three different bandwidth allocation policies.

D.4.4 Multiple Multicast Groups

We now consider the case of multiple multicast groups and 2000 unicast sessions. We add to the 2000 unicast sessions multicast sessions of 100 receivers each. The number of multicast sessions ranges from 2 to 100. There are 100 hosts on each LAN, the number of potential receivers/senders is therefore 18000. The simulations were repeated five times and average are taken over the five repetitions.

In this section, each plot can be partitioned into three parts: the first part shows the re-



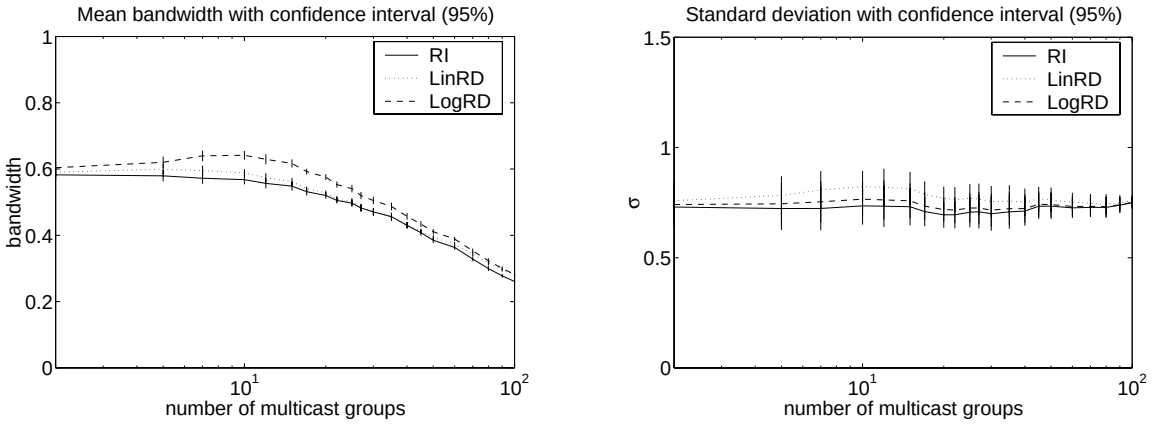
(a) Minimum bandwidth of unicast receivers.

(b) Minimum bandwidth of multicast receivers.

Figure D.12: Minimum bandwidth (Mbit/s) with confidence interval (95%) of the unicast receivers and of the multicast receivers for an increasing multicast group size $m = [1, ..., 6000]$, $k = 2000$, $M = 1$.

sults for a small number of multicast receivers with respect to the number of unicast receivers ($M = [1, ..., 10]$ groups), the second part shows the results for a large number of multicast receivers compared to the number of unicast receivers ($M = [10, ..., 50]$ groups), and the third part indicates the asymptotic behavior of our policies ($M = [50, ..., 100]$ groups). Again, the asymptotic case gives an indication about the behavior of our policies in extreme cases and about the robustness of our policies.

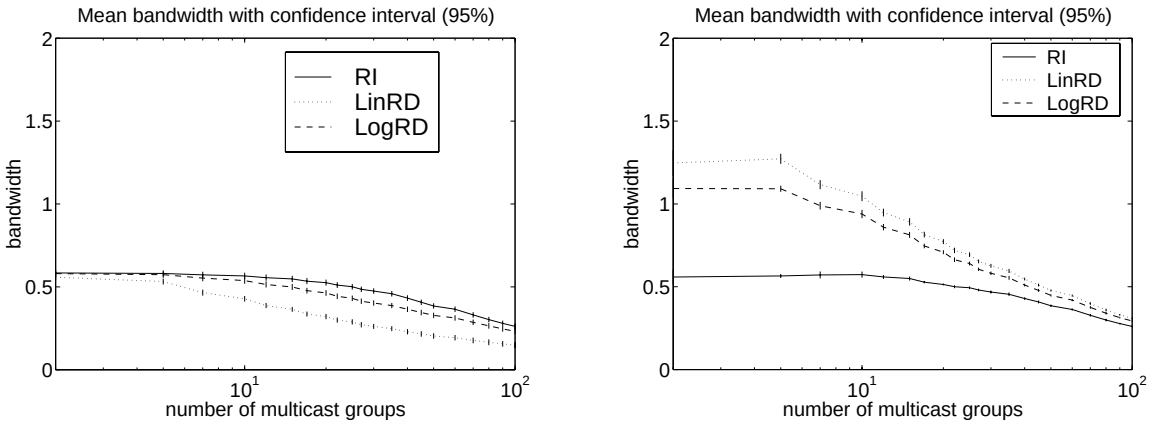
The three policies give nearly the same mean bandwidth over all the receivers (Fig. D.13(a)). The $LogRD$ policy is the best policy for the mean bandwidth over all the receivers. We can explain this behavior with our simple analytical study. We see for the chain topology that there are some cases where the $LinRD$ strategy gives worse results than the $LogRD$ and the RI strategy. We can consider a real network as a composition of stars and chains, therefore, it is not surprising to observe, for a large topology, a composition of the behavior of both the star and chain topology. We see that the bandwidth allocation of the $LogRD$ policy over the RI policy first slightly increases as the number of multicast groups increases (until $M = 10$), and then decreases with the number of multicast groups. For $M \leq 10$, the number of multicast receivers that benefits from the receiver-dependent policies increases and so the differences between receiver-dependent and receiver-independent policies increase. However, in the second part of the curves ($M > 10$), the number of multicast sessions tends to have more impact than the number of multicast receivers. Indeed, when the number of multicast session increases we have two behaviors: i) As the number of sessions (unicast or multicast) increases the bandwidth



(a) Mean bandwidth.

(b) Standard deviation.

Figure D.13: Mean bandwidth (Mbit/s) and standard deviation of all the receivers for an increasing number of multicast sessions, $k = 2000$, $M = [2, ..., 100]$, $m = 100$.

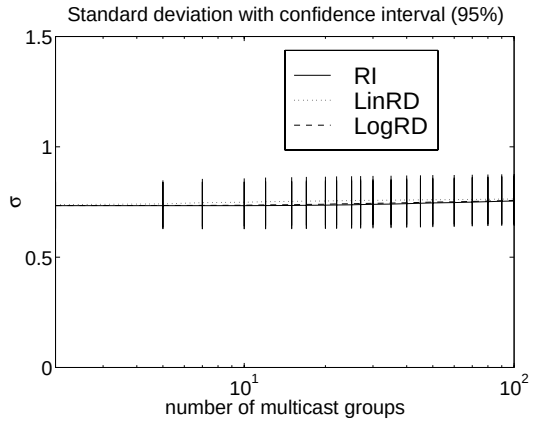


(a) Unicast receivers.

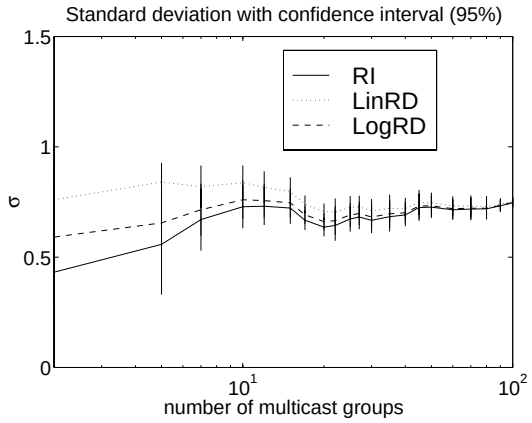
(b) Multicast receivers.

Figure D.14: Mean bandwidth (Mbit/s) of unicast and multicast receivers with confidence interval (95%) for an increasing number of multicast sessions, $k = 2000$, $M = [2, ..., 100]$, $m = 100$.

available for each session decreases, and therefore, the benefits due to receiver-dependent policies decreases; ii) The receiver-dependent policies reward multicast flows as a function of the number of receivers. But, if all the flows have the same number of receivers, receiver-dependent policies do not make any significant difference. Fig. D.14(a) shows that the $LogRD$ policy gives roughly the same bandwidth than the RI policy for unicast receivers whereas the $LinRD$ policy leads to a lower bandwidth. Fig. D.14(b) shows a very important result, the receiver-dependent

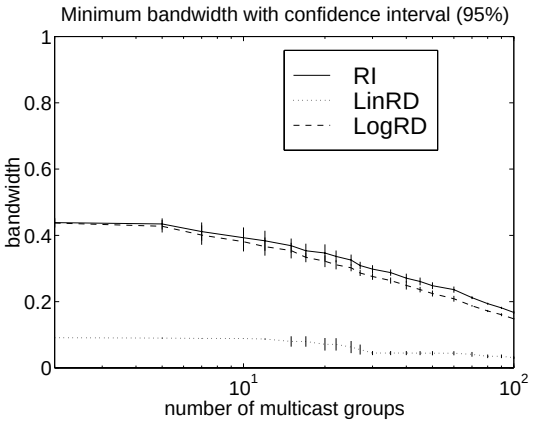


(a) Unicast receivers.

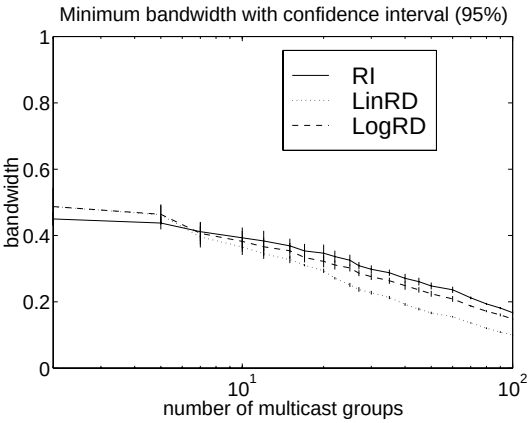


(b) Multicast receivers.

Figure D.15: Standard deviation of unicast and multicast receivers with confidence interval (95%) for an increasing number of multicast sessions, $k = 2000$, $M = [2, ..., 100]$, $m = 100$.



(a) Minimum bandwidth of unicast receivers.



(b) Minimum bandwidth of multicast receivers.

Figure D.16: Minimum bandwidth (Mbit/s) with confidence interval (95%) of the unicast receivers and of the multicast receivers for an increasing number of multicast sessions, $k = 2000$, $M = [2, ..., 100]$, $m = 100$.

policies significantly reward the multicast receivers compared to the RI policy. As the number of multicast groups increases, the differences between the policies decrease, since the number of multicast sessions tends to have more impact on the mean bandwidth than the number of multicast receivers. Fig. D.14(b) shows that the receiver dependent policies achieve their objective, which is to reward multicast flows.

Fig. D.13(b) shows that standard deviation is roughly the same for the three bandwidth allo-

cation policies. Fig. D.15(b) shows that the multicast receivers have higher standard deviation with the receiver-dependent policies than with RI. The standard deviation is roughly the same for the three bandwidth allocation policies for the unicast receivers (Fig. D.15(a)). As the number of multicast sessions increases, multicast flows dominate due to the high amount of multicast receivers compared to unicast receivers, and therefore, the standard deviation of multicast receivers for the three bandwidth allocation strategies becomes close due to the high homogeneity of the sessions. The minimum bandwidth is dictated by the unicast receivers, so the plots for all the receivers and for the unicast receivers are the same. Fig. D.16(a) shows an interesting result. The $LinRD$ policy gives very little bandwidth to unicast receivers, whereas the $LogRD$ policy allocates roughly the same minimum bandwidth than the RI policy. Fig. D.16(b) shows the minimum bandwidth for multicast receivers is slightly better for the receiver-dependent policy than for RI for a small number of multicast sessions, and the minimum bandwidth is slightly worse for a large number of multicast sessions. Indeed, for a small number of multicast sessions the interaction between sessions is low, therefore the probability that a multicast session decreases the bandwidth seen by a multicast receiver of another session is low. But, for a large number of multicast sessions, the interaction between multicast sessions is high, and the probability that a multicast session decreases the bandwidth seen by a multicast receiver of another session is higher.

We did another experiment that aims to model small conferencing groups where multicast groups of a size 20 are added. But the results of this experiment do not differ from the results of the experiment with multicast group sizes of 100 receivers and we do not present these results.

In conclusion, the receiver satisfaction and fairness of all the receivers are roughly the same for the three bandwidth allocation strategies (Fig. D.13), but the $LogRD$ policy is the only policy that greatly improves the average bandwidth allocated to multicast receivers (Fig. D.14(b)) without starving unicast flows (Fig. D.16(a)).

D.5 Practical Aspects

D.5.1 Estimating the Number of Downstream Receivers

Up to now, we quantified the advantages of using bandwidth allocation strategies based on the number of downstream receivers. Estimating the number of receivers downstream of a network node has a certain cost but has other benefits that largely outweigh this cost. Two examples of these benefits are feedback accumulation and multicast charging.

One of the important points of the feedback accumulation process is the estimation of the number of downstream receivers. Given the number of receivers is known in the network nodes, the distributed process of feedback accumulation [66], or feedback filtering in network nodes

becomes possible and has a condition to terminate upon.

While multicast saves bandwidth, it is currently not widely offered by network operators due to the lack of a valid charging model [14, 37]. By knowing the number of receivers at the network nodes, different charging models for multicast can be applied, including charging models that use the number of receivers. In the case of a single source and multiple receivers, the amount of resources used with multicast depends on the number of receivers. For an ISP, in order to charge the source according to the resources consumed, the number of receivers is needed. The bandwidth allocation policy used impacts the charging in the sense that the allocation policy changes the number of resources consumed by a multicast flow, and changes the cost of a multicast flow for the ISP. However, in appendix D.8, we see that a simple local bandwidth allocation policy leads to a global cost that is a complex function of the number of receivers. It is not clear to us whether an ISP can charge a multicast flow with a simple linear or logarithmic function of the number of receivers. Moreover, several ISPs (see [21]) use flat rate pricing for multicast due to the lack of valid charging model. Even in the case of flat rate pricing, the number of downstream receivers is useful when a multicast tree spans multiple ISPs. In this case, we have a means to identify the number of receivers in each ISP. The charging issue is orthogonal to our paper and is an important area for future research.

The estimation of the number of downstream receivers is feasible, for instance, with the Express multicast routing protocol [37]. The cost of estimating the number of downstream receivers is highly dependent on the method used and the accuracy of the estimate required. As our policy is based on a logarithmic function, we only need a coarse estimate of the number of downstream receivers. Holbrook [37] describes a low overhead method for the estimation of the number of downstream receivers.

D.5.2 Introduction of the LogRD Policy

Another important question is how to introduce the $LogRD$ policy in a real network without starving unicast flows. In section D.4, we show that even in asymptotic cases the $LogRD$ strategy does not starve unicast flows, but we do not have a hard guarantee about the bandwidth allocated to unicast receivers. For instance, one multicast flow with 1 million downstream receivers sharing the same bottleneck than a unicast flow will grab 93% of the available bandwidth. This is a large amount of the bandwidth, but that does not lead to a starvation of the unicast flow.

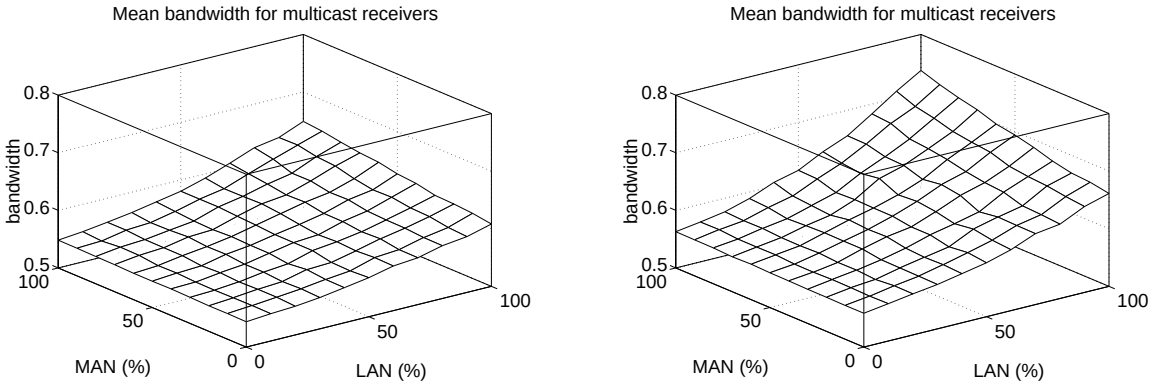
The $LogRD$ policy will asymptotically – when the number of multicast receivers tends toward infinity – lead to an optimal receiver satisfaction (limited by the capacity of the network) and to a low fairness. In particular, the multicast flow will grab all the available bandwidth of the bottleneck link and starve all the unicast flows sharing this bottleneck link. It is possible to devise a strategy based on the $LogRD$ policy that allocates to the multicast flows never more

than K times the bandwidth allocated to the unicast flows sharing the same bottleneck. We can imagine the $LogRD$ strategy to be used in a hierarchical link sharing scheme (see [31, 4] for hierarchical link sharing models). The idea is to introduce our policy in the general scheduler [31] (for instance we can configure the weight of a PGPS [65] scheduler with the $LogRD$ policy to achieve our goal), and to add an administrative constraint in the link sharing scheduler (for instance we guarantee that unicast traffic receives at least x% of the link bandwidth). This is a simple way to allocate the bandwidth with respect to the $LogRD$ policy, and to guarantee a minimum bandwidth for the unicast flows. Moreover, Kumar et al. [45] show that it is possible to integrate efficiently a mechanism like HWFQ [4] in a Gigabit router, and WFQ is already available in the recent routers [12].

D.5.3 Incremental Deployment

An important practical aspect is whether it is possible to incrementally deploy the $LogRD$ policy. To answer this question we make the following experiment. We consider the random topology used in section D.4 and a unicast environment consisting of 2000 unicast flows. We add to this unicast environment 20 multicast flows with a uniform group size of 50 multicast receivers randomly distributed. The simulation consists in varying the percentage of LANs, MANs, and WANs that use the $LogRD$ policy compared to the RI policy. We make the assumption that each LAN, MAN, and WAN is an autonomous system managed by a single organization. So when an organization decides to use the $LogRD$ policy, it changes the policy in all the routers of the LAN, MAN, or WAN it is responsible for. We say that a LAN, MAN or WAN is $LogRD$ if all the routers use the $LogRD$ policy. The simulation consists in varying the number of $LogRD$ LANs and MANs from 0% to 100%, for the WAN we only look at a full support (all routers are $LogRD$) or no support (all routers are RI). We call these percentages respectively perLAN, perMAN, and perWAN. This simulation is repeated five times and averages are taken over the five repetitions. The results are given with a confidence interval of 95% $\pm$ 20Kbit/s around the mean bandwidth.

The main behavior we see in Fig. D.17 is the *interdependency* of the parameters perLAN, perMAN, and perWAN on the mean bandwidth for the multicast receivers. An isolated deployment of the $LogRD$ in just the LANs, MANs, or WANs does not allow to achieve a mean bandwidth close to the mean bandwidth obtained when the whole network is $LogRD$. For instance, the perMAN parameter does not have a significant influence on the mean bandwidth when $perLAN = 0$. However, when $perLAN = 100$ and $perWAN = 100$, the perMAN parameter has a significant influence on the mean bandwidth. The results obtained depend on the network configuration (number of LANs, MANs, and WANs, link bandwidth, etc.). However, we believe the property of interdependency of the parameters perLAN, perMAN, and perWAN to hold in all the cases.



(a) 100% of RI links in the WAN

(b) 100% of $LogRD$ links in the WAN

Figure D.17: Influence on the mean bandwidth (Mbit/s) for the multicast receivers for an hierarchical incremental deployment of the $LogRD$ policy, $k = 2000$, $M = 20$, $m = 50$.

In conclusion, to reap the full benefit of the $LogRD$ policy, a coordinated deployment is necessary. However, as the lack of links using the $LogRD$ allocation does not lead to any performance degradation for the network, an incremental deployment is possible.

D.6 Conclusion

If one wants to introduce multicast in the Internet, one should give an incentive to use it. We propose a simple mechanism that takes into account the number of receivers downstream. Our proposal does not starve unicast flows and greatly increases multicast receiver satisfaction.

We defined three different bandwidth allocation strategies as well as criteria to compare these strategies. We compared the three strategies analytically and through simulations. Analytically, we studied two simple topologies: a star, and a chain. We showed that the $LogRD$ policy leads to the best tradeoff between receiver satisfaction and fairness. The striking similarities in the results for the analytical study and the simulations confirm that we had chosen valid models.

To simulate real networks, we defined a large topology consisting of WANs, MANs, and LANs. In a first round of experiments, we determined the right number of unicast receivers. We studied the introduction of multicast in a unicast environment with three different bandwidth allocation policies. The aim was to understand the impact of multicast in the real Internet. We showed that allocating link bandwidth dependent on the flows' number of downstream receivers results in a higher receiver satisfaction. The $LogRD$ policy provides the best tradeoff between the receiver satisfaction and the fairness among receivers. Indeed, the $LogRD$ policy always

leads to higher receiver satisfaction than the RI policy for roughly the same fairness, whereas the $LinRD$ policy leads to higher receiver satisfaction than the $LogRD$ policy, however, at the expense of unacceptable decrease in fairness.

Our contribution in this paper is the definition and evaluation of a new bandwidth allocation policy called $LogRD$ that gives a real incentive to use multicast. Also, the $logRD$ policy gives a relevant answer to the open question on how to treat a multicast flow compared to a unicast flow sharing the same bottleneck. To the best of our knowledge, we are the first that take into account the number of multicast receivers to reward multicast flows. Moreover, we show that the deployment of the $LogRD$ policy is feasible when deployed per ISP at the same time as the ISP upgrades its network to be multicast capable.

D.7 Discussion on Multicast Gain

To evaluate the bandwidth multicast gain, we restrict ourselves to the case of a full o-ary tree with receivers at the leaves – in this case we model a point to point network – or with broadcast LAN at the leaves. We consider one case where the unicast and the multicast cost only depends on the number of links (the unlimited bandwidth case) and one case where the unicast and the multicast cost depends on the bandwidth used (the limited bandwidth case).

Let the full o-ary tree be of height h. We assume the sender to be at the root, so there are $R = o^h$ receivers or $N = o^h$ LANs with R_N receivers on each LAN ($R = R_N \cdot N$). We define the **bandwidth cost** as the sum of all the bandwidths consumed on all the links of the tree. We define the **link cost** as the sum of all the links used on the tree, we count n times the same link when the same data are sent n times on this link. Let C_U be the unicast bandwidth/link cost from the sender to all of the receivers and C_M the multicast bandwidth/link cost from the same sender to the same receivers.

D.7.1 Bandwidth-Unlimited Case

We assume that every link of the tree has unlimited bandwidth. Let C_U and C_M be the link cost for unicast and multicast, respectively.

If we consider one receiver on each leaf of the tree we have:

$$C_U = o^h + o^{h-1} \cdot o + \cdots + o^1 \cdot o^{h-1} = h \cdot o^h = h \cdot R = R \cdot log_o(R) \tag{D.7}$$

$$C_M = \sum_{i=1}^{h} o^i = \frac{o^{h+1} - o}{o - 1} = \frac{o}{o-1}(R - 1)$$

We define the multicast gain as the ratio:

$$\frac{C_U}{C_M} = log_o(R)\frac{R}{R-1} \cdot \frac{o-1}{o}$$

The multicast gain depends logarithmically on the number of receivers.

If we consider one LAN on each leaf of the tree we have:

$$C_U = h \cdot R = h \cdot N \cdot R_N = R_N \cdot N \cdot log_o(N)$$

$$C_M = \sum_{i=1}^{h} o^i = \frac{o^{h+1} - o}{o-1} = \frac{o}{o-1}(N-1)$$

We define the multicast gain as the ratio:

$$\frac{C_U}{C_M} = \frac{o-1}{o} \cdot R_N \cdot \frac{1}{1-\frac{1}{N}} \cdot log_o(N)$$

The gain depends logarithmically on the number of LANs and linearly on the number of receivers per LAN.

D.7.2 Bandwidth-Limited Case

Every link of the tree has a capacity C. Let C_U and C_M be the bandwidth cost for unicast and multicast, respectively.

If we consider one receiver on each leaf of the tree we have:

$$\begin{aligned} C_U &= o \cdot C + o^2 \cdot \frac{C}{o} + o^3 \cdot \frac{C}{o^2} + \cdots + o^h \cdot \frac{C}{o^{h-1}} \\ &= \sum_{i=1}^{h} C \cdot o = h \cdot C \cdot o = C \cdot o \cdot log_o(R) \\ C_M &= C \sum_{i=1}^{h} o^i = C \cdot \frac{o^{h+1} - o}{o-1} = C \cdot \frac{o}{o-1}(R-1) \end{aligned}$$

The multicast gain is:

$$\frac{C_U}{C_M} = (o-1)\frac{\log_o(R)}{R-1}$$

This means that there is a multicast gain smaller than 1 for large R. But, of course, in the unicast case (which is now globally less expensive), we also have much smaller receiver satisfaction due to the bandwidth-limited links close to the source. Therefore, the definition for the standard multicast gain does not make sense in the bandwidth-limited case. For the unlimited case, receivers are equally satisfied, since they receive the same bandwidth and the multicast gain *makes sense*.

We need to define another measure that combines the satisfaction and the cost. We use cost per satisfaction. We look at the ratio of bandwidth cost per satisfaction that tells us how much bandwidth we need to invest to get a unit of satisfaction.

We now employ: $GB = \frac{\text{global cost}}{\text{global satisfaction}}$. To compute the global satisfaction, we add the satisfaction over all receivers. Let the global satisfaction be S_U for unicast and S_M for multicast.

$$\begin{aligned} S_U &= R \cdot C \cdot \frac{1}{o^{h-1}} = R \cdot C \cdot \frac{o}{o^h} = R \cdot C \cdot \frac{o}{R} = C \cdot o \\ S_M &= R \cdot C \end{aligned}$$

Then $GB = \frac{\text{global cost}}{\text{global satisfaction}}$ is :

$$GB_U = \frac{C_U}{S_U} = \frac{C \cdot o \cdot log_o(R)}{C \cdot o} = log_o(R)$$

$$GB_M = \frac{C_M}{S_M} = \frac{(R-1)}{R} \cdot \frac{o}{o-1}$$

Now the new multicast gain is:

$$\frac{GB_U}{GB_M} = \frac{o-1}{o} \cdot \frac{R}{R-1} \cdot log_o(R)$$

The gain depends logarithmically on the number of receivers.

If we consider one LAN on each leaf of the multicast tree we have:

$$\begin{aligned} C_U &= o \cdot C + o^2 \cdot \frac{C}{o} + o^3 \cdot \frac{C}{o^2} + \cdots + o^h \cdot \frac{C}{o^{h-1}} = C \cdot o \cdot log_o(N) \\ C_M &= C \sum_{i=1}^{h} o^i = C \cdot \frac{o^{h+1} - o}{o-1} = C \cdot \frac{o}{o-1}(N-1) \end{aligned}$$

The multicast gain is:

$$\frac{C_U}{C_M} = (o-1)\frac{\log_o(N)}{N-1}$$

Once again the multicast gain smaller than 1 for large N. The global satisfaction is:

$$\begin{aligned} S_U &= R \cdot C \cdot \frac{1}{o^{h-1} R_N} = C \cdot o \\ S_M &= R \cdot C \end{aligned}$$

Then $GB = \frac{\text{global cost}}{\text{global satisfaction}}$ is :

$$GB_U = \frac{C_U}{S_U} = log_o(N)$$

$$GB_M = \frac{C_M}{S_M} = \frac{N-1}{R_N \cdot N} \cdot \frac{o}{o-1}$$

Now the new multicast gain is:

$$\frac{GB_U}{GB_M} = \frac{o-1}{o} \cdot \frac{R_N \cdot N}{N-1} \cdot log_o(N)$$

The gain depends logarithmically on the number of LANs and linearly on the number of receivers per LAN.

In conclusion, for both the unlimited and the limited bandwidth case, the multicast gain has a logarithmic trend with the number of receivers in case of point-to-point networks. For broadcast LANs at the leaves of the multicast distribution tree, the multicast gain has a logarithmic trend with the number of LANs, but a linear trend with the number of receivers per LAN. Therefore, with a small number of receivers per LAN the multicast gain is logarithmic but with a large number of receivers per LANs the multicast gain is linear.

D.8 Global Impact of a Local Bandwidth Allocation Policy

We consider a full o-ary tree for the unlimited bandwidth case when there is a receiver per leaf. The unicast link cost is $C_U = h \cdot R$ (see Eq. D.7). Now we consider the multicast link cost for the RI, the $LinRD$, and $LogRD$ policies. For instance when there are 2 receivers downstream of link l, the $LinRD$ policy allocates the equivalent of 2 units of bandwidth and the $LogRD$ policy allocates the equivalent of $1 + \ln(2)$ units of bandwidth compared to the RI policy which allocates 1 unit of bandwidth.

The multicast link cost for the RI policy is:

$$C_M^{RI} = \sum_{i=1}^{h} o^i = \frac{o}{o-1}(R-1)$$

The multicast link cost for the $LinRD$ policy is:

$$C_M^{LinRD} = o \cdot \frac{R}{o} + o^2 \cdot \frac{R}{o^2} + \cdots + o^h \cdot \frac{R}{o^h} = h \cdot R = C_U$$

The multicast link cost for the $LogRD$ policy is:

$$C_M^{LogRD} = o \cdot (1 + \ln \frac{R}{o}) + o^2 \cdot (1 + \ln \frac{R}{o^2}) + \cdots + o^h \cdot (1 + \ln \frac{R}{o^h}) = \sum_{i=1}^{h} o^i (1 + \ln \frac{R}{o^i})$$

We have $1 + \ln \frac{R}{o^i} \leq \frac{R}{o^i}$ and $1 + \ln \frac{R}{o^i} < \frac{R}{o^i}$ for $\frac{R}{o^i} \neq 1$. So for $h > 1$ and $o > 1$ we have $C_M^{LogRD} < C_M^{LinRD}$.

In conclusion we see that the policy that rewards multicast with its gain is the $LinRD$ policy and not the $LogRD$ policy as expected.

D.9 Tiers Setup

We give a brief description of the topology used for all the simulations. The random topology *RT* is generated with `tiers` v1.1 using the command line parameters `tiers 1 20 9 5 2 1 3 1 1 1 1`. A WAN consists of 5 nodes and 6 links and connects 20 MANs, each consisting of 2 nodes and 2 links. To each MAN, 9 LANs are connected. Therefore, the core topology consists of $5 + 40 + 20 \cdot 9 = 225$ nodes. The capacity of WAN links is 155Mbit/s, the capacity of MAN links is 55Mbit/s, and the capacity of LAN links is 10Mbit/s.

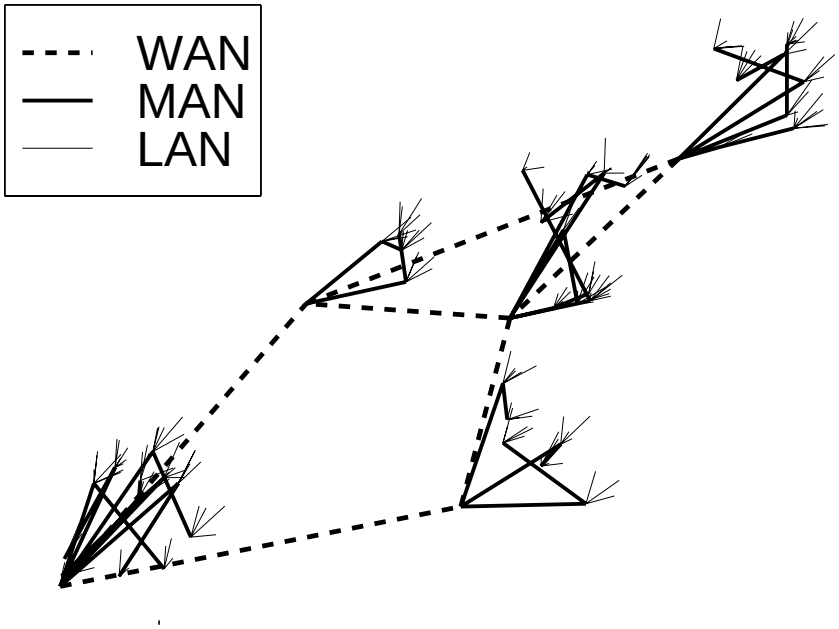


Figure D.18: The random topology *RT*

Each LAN is represented as a single node and connects several hosts via a 10Mbit/s link. The number of hosts connected to a LAN changes from experiment to experiment to speed up simulation. However, the number of hosts is always chosen larger than the sum of the receivers and the sources all together.

Bibliographie

[1] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, Internet Engineering Task Force, April 1999.

[2] S. Bajaj, L. Breslau, and S. Shenker, "Uniform versus Priority Dropping for Layered Video", In *Proc. of ACM SIGCOMM'98*, pp. 131–143, Vancouver, British Columbia, CANADA, September 1998.

[3] J. Bennett and H. Zhang, "WF2Q: Worst-case Fair Weighted Fair Queueing", In *Proc. of IEEE INFOCOM'96*, pp. 120–128, San Francisco, CA, USA, March 1996.

[4] J. C. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms", *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997.

[5] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1992.

[6] K. Bharat-Kumar and J. Jaffe, "A new Approach to Performance-Oriented Flow Control", *IEEE Transactions on Communications*, 29(4):427–435, 1981.

[7] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-Based error control for Internet Telephony", In *Proc. of IEEE INFOCOM'99*, pp. 1453–1460, New York, March 1999.

[8] J. Bolot, T. Turletti, and I. Wakeman, "Scalable Feedback Control for Multicast Video Distribution in the Internet", In *Proc. of ACM SIGCOMM'94*, pp. 58–67, September 1994.

[9] K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet Topology", *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[10] "Castify Networks", http://www.castify.net.

[11] V. Cerf, Y. Dalal, and C. Sunshine, "Specification of Internet Transmission Control Program", RFC 675, December 1974.

[12] Cisco, "Advanced QoS Services for the Intelligent Internet", White Paper, May 1997.

[13] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in Computer Networks: Motivation, Formulation, and Example", *IEEE/ACM Transactions on Networking*, 1(6):614–627, December 1993.

[14] R. Comerford, "State of the Internet: Roundtable 4.0", *IEEE Spectrum*, 35(10):69–79, October 1998.

[15] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[16] S. Deering, "Host Extensions for IP Multicasting", *Internet Request for Comments*, RFC 1112, August 1989.

[17] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM Architecture for Wide–Area Multicast Routing", *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.

[18] S. E. Deering, "Multicast routing in internetworks and extended LANs", In *Proc. ACM SIGCOMM 88*, pp. 55–64, Stanford, CA, August 1988.

[19] D. DeLucia and K. Obraczka, "A Multicast Congestion Control Mechanism Using Representatives", Technical report 97-651, Computer Science Department - University of Southern California, May 1997.

[20] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", In *Proc. of ACM SIGCOMM'89*, pp. 1–12, Austin, Texas, September 1989.

[21] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture", *IEEE Network magazine special issue on Multicasting*, 14(1):78–88, January/February 2000.

[22] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing", In *Proceedings of IEEE INFOCOM'93*, volume 1, pp. 82–89, 1993.

[23] M. B. Doar, "A Better Model for Generating Test Networks", In *Proceedings of IEEE Global Internet*, pp. 86–93, London, UK, November 1996, IEEE.

[24] H. Eriksson, "MBONE: The Multicast Backbone", *Communications of the ACM*, 37(8):54–60, August 1994.

[25] "FastForward Networks", http://www.ffnet.com.

[26] A. Feldman, *Welfare economics and social choice theory*, Martinus Nijhoff Publishing, Boston, 1980.

[27] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control", In *Proc. of ACM SIGCOMM'99*, pp. 301–313, September 1999.

[28] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1:One-way Traffic", *Computer Communications Review*, 21(5):30–47, October 1991.

[29] S. Floyd, "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, 24(5):10–23, October 1994.

[30] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[31] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", *IEEE/ACM Transactions on Networking,*, 3(4):365–386, August 1995.

[32] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[33] J. S. Golestani and S. Bhattacharyya, "End-to-End Congestion Control for the Internet: A Global Optimization Framework", In *Proc 6th Int. Conf. on Network Protocols*, pp. 137–150, October 1998.

[34] S. J. Golestani and K. K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet", In *Proc. of INFOCOM'99*, pp. 990–1000, New York, USA, March 1999.

[35] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. J. Sreenan, "Stability and Fairness Issues in Layered Multicast", In *Proc. of NOSSDAV'99*, pp. 31–44, Basking Ridge, NJ, USA, June 1999.

[36] E. L. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks", *IEEE Journal on Selected Areas in Communications*, 9(7):1024–1039, September 1991.

[37] H. W. Holbrook and D. R. Cheriton, "IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications", In *Proc. of ACM SIGCOMM'99*, pp. 65–78, Harvard, Massachusetts, USA, September 1999.

[38] C. Huitema, *Et Dieu Créa l'INTERNET*, Eyrolles, 1995.

[39] V. Jacobson, "Congestion Avoidance and Control", In *Proc. of ACM SIGCOMM'88*, pp. 314–329, Stanford, CA, August 1988.

[40] R. Jain, D. M. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems", Technical report 301, DEC, Littleton, MA, September 1984.

[41] T. Jiang, M. H. Ammar, and E. W. Zegura, "Inter-Receiver Fairness: A Novel Performance Measure for Multicast ABR Sessions", In *Proc. of ACM Sigmetrics*, pp. 202–211, June 1998.

[42] F. P. Kelly, "Charging and rate control for elastic traffic", *European Transactions on Telecommunications*, 8:33–37, 1997.

[43] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability", *Journal of the Operational Research Society*, 49:237–252, March 1998.

[44] S. Keshav, *Congestion Control in Computer Networks*, Ph.D. Thesis, EECS, University of Berkeley, CA 94720, USA, September 1991.

[45] V. P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", *IEEE Communications Magazine*, 36(5):152–164, May 1998.

[46] C. Lefelhocz, B. Lyles, S. Shenker, and L. Zhang, "Congestion Control for Best-Effort Service : Why We Need a New Paradigm", *IEEE Network*, pp. 10–19, January/February 1996.

[47] A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth Allocation Policies for Unicast and Multicast Flows", In *Proc. of IEEE INFOCOM'99*, pp. 254–261, New York, NY, USA, March 1999.

[48] A. Legout and E. W. Biersack, "Beyond TCP-Friendliness: A New Paradigm for End-to-End Congestion Control", Technical report, Institut Eurecom, November 1999, http://www.eurecom.fr/ legout/Research/research.html.

[49] A. Legout and E. W. Biersack, "Pathological Behaviors for RLM and RLC", In *Proc. of NOSSDAV'00*, pp. 164–172, Chapel Hill, North Carolina, USA, June 2000.

[50] A. Legout and E. W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", In *Proc. of ACM SIGMETRICS'2000*, pp. 13–22, Santa Clara, CA, USA, June 2000.

[51] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic", In *Proc. of ACM SIGCOMM'93*, pp. 183–193, September 1993.

[52] K. Leonard, "Research Areas in Computer Communication", In *Computer Communication Review, ACM SIGCOMM*, volume 4, July 1974.

[53] M. R. Macedonia and D. P. Brutzmann, "MBone Provides Audio and Video Across the Internet", *IEEE Computer*, 7(4):30–36, April 1994.

[54] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", *Computer Communication Review, ACM SIGCOMM*, 27(3):67–82, July 1997.

[55] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", In *SIGCOMM 96*, pp. 117–130, August 1996.

[56] J. Nagle, "Congestion control in TCP/IP internetworks", *Computer Communication Review*, 14(4):11–17, October 1984.

[57] J. Nagle, "On packet switches with infinite storage", *IEEE Transactions on Communications*, COM-35(4):435–438, April 1987.

[58] S. Nelakuditi, R. R. Harinath, E. Kusmierek, and Z.-L. Zhang, "Providing Smoother Quality Layered Video Stream", In *Proceedings of NOSSDAV'00*, Chapel Hill, North Carolina, USA, June 2000.

[59] J. Nonnenmacher, *Reliable Multicast to Large Groups*, Ph.D. Thesis, EPFL, Lausanne, Switzerland, July 1998.

[60] J. Nonnenmacher and E. W. Biersack, "Scalable Feedback for Large Groups", *IEEE/ACM Transactions on Networking*, 7(3):375–386, June 1999.

[61] J. Nonnenmacher and E. Biersack, "Asynchronous Multicast Push: AMP", In *Proceedings of ICCC'97*, pp. 419–430, Cannes, France, November 1997.

[62] NS, UCB/LBNL/VINT Network Simulator - ns (version 2), http://www.isi.edu/nsnam/ns.

[63] T. Ott, J. Kemperman, and M. Mathis, "The stationay distribution of ideal TCP Congestion Avoidance", Technical report, Bellcore, August 1996.

[64] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", In *Proc. of ACM SIGCOMM'98*, pp. 303–314, Vancouver, Canada, August 1998.

[65] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", In *Proc. IEEE INFOCOM'93*, pp. 521–530, 1993.

[66] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)", *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, 15(3):407 – 421, April 1997.

[67] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. Thesis, University of California, Berkeley, April 1997.

[68] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of Multicast Trees: Comments on the Chuang-Sirbu Scaling Law", In *Proc. of ACM SIGCOMM'99*, pp. 41–51, Harvard, Massachusetts, USA, September 1999.

[69] J. Postel, "Transmission Control Protocol – Protocol Specification", Request for Comments (Standard) RFC 793, Information Sciences Institute, USC, September 1981.

[70] S. Ratnasamy and S. McCanne, "Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-End Measurements", In *Proc. of IEEE INFOCOM'99*, pp. 353–360, New York, USA, March 1999.

[71] D. P. Reed, J. H. Saltzer, and D. D. Clark, "Commentaries on Active Networking and End to End Arguments", *IEEE Network*, 12(3):66–71, May/June 1998.

[72] R. Rejaie, M. Handley, and D. Estrin, "Quality Adaptation for Congestion Controlled Video Playback over the Internet", In *Proc. of ACM SIGCOMM'99*, pp. 189–200, Cambridge, MA, USA, September 1999.

[73] I. Rhee, N. Ballaguru, and G. N. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast", Technical report TR-98-01, North Carolina State University, North Carolina, January 1998.

[74] L. Rizzo, "Fast Group Management in IGMP", In *Proc. of Hipparc'98*, 1998.

[75] L. Rizzo, "pgmcc: A TCP-friendly Single-Rate Multicast Congestion Control Scheme", In *Proc. of ACM SIGCOMM'00*, Stockholm, Sweden, August 2000.

[76] P. Rodriguez, K. W. Ross, and E. W. Biersack, "Distributing Frequently-Changing Documents in the Web: Multicasting or Hierarchical Caching", *Computer Networks and ISDN Systems. Selected Papers of the 3rd International Caching Workshop*, pp. 2223–2245, 1998.

[77] D. Rubenstein, J. Kurose, and D. Towsley, "The Impact of Multicast Layering on Network Fairness", In *Proc. of ACM SIGCOMM'99*, pp. 27–38, September 1999.

[78] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to end arguments in system design", *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[79] S. Shenker, "Making Greed Work in Networks: A Game-Theoric Analysis of Switch Service Disciplines", In *Proc. of ACM SIGCOMM'94*, pp. 47–57, University College London, London, UK, October 1994.

[80] D. Sisalem and A. Wolisz, "MLDA: A TCP-friendly congestion control framework for heterogenous multicast environments", In *Proc. of IWQoS 2000*, Pittsburgh, USA, June 2000.

[81] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", Request for Comments RFC 2001, Internet Engineering Task Force, January 1997.

[82] D. Stiliadis and A. Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms", In *Proc. of IEEE INFOCOM '97*, pp. 326–335, April 1997.

[83] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury, "Design Considerations for Supporting TCP with Per-flow Queueing", In *Proc. of IEEE INFOCOM'98*, pp. 299–306, April 1998.

[84] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols", *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, 1997.

[85] T. Turletti, S. Fosse-Parisis, and J. Bolot, "Experiments with a Layered Transmission Scheme over the Internet", Research report, INRIA, B.P.93, Sophia-Antipolis Cedex, France, November 1997.

[86] L. Vicisano, "Notes on a cumulative layered organization of data packets accross multiple streams with different rates", Technical report, UCL London, January 1997.

[87] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", In *Proc. of IEEE INFOCOM'98*, pp. 996–1003, San Francisco, CA, USA, March 1998.

[88] L. Wu, R. Sharma, and B. Smith, "Thin Streams: An Architecture for Multicasting Layered Video", In *Proc. of NOSSDAV'97*, pp. 173–182, St Louis, Missouri, USA, May 1997.

[89] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network", In *Proceedings of IEEE Global Internet*, London, UK, November 1996.

[90] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork", In *Infocom '96*, pp. 594–602, March 1996.

[91] E. W. Zegura, K. Calvert, and M. J. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology", *IEEE/ACM Transactions on Networking*, 5(6):770–783, December 1997.

Publications

Journal

A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth Allocation Policies for Unicast and Multicast Flows", Submission under revision for IEEE/ACM *Transactions on Networking*, September 2000.

A. Legout and E. W. Biersack, "Beyond TCP-Friendliness: A New Paradigm for End-to-End Congestion Control", Submitted to Special Issue of the IEEE Network Magazine on Control of Best Effort Traffic, September 2000.

Conférence

A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth Allocation Policies for Unicast and Multicast Flows", In *Proc. of IEEE INFOCOM'99*, pp. 254–261, New York, NY, USA, March 1999.

A. Legout and E. W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", In *Proc. of ACM SIGMETRICS'2000*, pp. 13–22, Santa Clara, CA, USA, June 2000.

A. Legout and E. W. Biersack, "Pathological Behaviors for RLM and RLC", In *Proc. of NOSS-DAV'00*, pp. 164–172, Chapel Hill, North Carolina, USA, June 2000.

Résumé

Une des clefs de l'amélioration de la qualité de service pour les réseaux *best effort* est le contrôle de congestion. Dans cette thèse, on a étudié le problème du contrôle de congestion pour la transmission multipoint dans les réseaux *best effort*. Cette thèse présente quatre contributions majeures. On a commencé par étudier deux protocoles de contrôle de congestion multipoints RLM et RLC. On a identifié des comportements pathologiques pour chaque protocole. Ceux-ci sont extrêmement difficiles à corriger dans le contexte actuel de l'internet, c'est-à-dire en respectant le paradigme *TCP-friendly*. On a alors réfléchi au problème du contrôle de congestion dans le contexte plus général des réseaux *best effort*. Ceci nous a conduit à redéfinir la notion de congestion, définir les propriétés requises par un protocole de contrôle de congestion idéal et définir un nouveau paradigme pour la conception de protocoles de contrôle de congestion presque idéaux. On a introduit à cet effet le paradigme *Fair Scheduler* (FS). L'approche que l'on a utilisée pour définir ce nouveau paradigme est purement formelle. Pour valider cette approche théorique, on a conçu grâce au paradigme FS un nouveau protocole de contrôle de congestion multipoint à couches cumulatives et orienté récepteur : PLM, qui est capable de suivre les évolutions de la bande passante disponible sans aucune perte induite, même dans un environnement autosimilaire et multifractal. PLM surpasse RLM et RLC et valide le paradigme FS. Comme ce paradigme permet de concevoir des protocoles de contrôle de congestion multipoints et point à point, on a défini une nouvelle politique d'allocation de la bande passante entre flux multipoints et flux point à point. Cette politique, appelée $LogRD$, permet d'améliorer considérablement la satisfaction des utilisateurs multipoints sans nuire aux utilisateurs point à point.

Abstract

An efficient way to improve quality of service for best effort networks is through congestion control. We present in this thesis a study of multicast congestion control for best effort networks. This thesis shows four major contributions. We first exhibit some pathological behaviors for the multicast congestion control protocols RLM and RLC. As these pathological behaviors are extremely hard to fix in the context of the current Internet (i.e. with the TCP-friendly paradigm), we thought about the problem of congestion control in the more general case of best effort networks. We give a new definition of congestion, we define the properties required by an ideal congestion control protocol, and we define a paradigm, the fair scheduler (FS) paradigm, for the design of nearly ideal end to end congestion control protocols. We define this paradigm in a formal way. To validate this paradigm in a pragmatic way, we design with the FS paradigm a new multicast congestion control protocol: PLM. This protocol converges fast to the available bandwidth and tracks this available bandwidth without loss induced even in a self similar and multifractal environment. PLM outperforms RLM and RLC and validates the FS paradigm claims. As the FS paradigm allows to devise multicast and unicast congestion control protocols, we define a new bandwidth allocation policy for unicast and multicast flows. This policy called $LogRD$ allows to increase the multicast receiver satisfaction without significantly decreasing the unicast receiver satisfaction.