N° d'ordre : 2009telb0129

THÈSE

Présentée

École nationale supérieure des télécomunications de Bretagne

en habilitation conjointe avec l'Université de Rennes 1

pour obtenir le grade de
Docteur de Télécom Bretagne

Mention : *Informatique*

par

Andrés Emilio ARCIA MORET

Modifications du mécanisme d'acquittement du protocole TCP : évaluation et application aux réseaux filaires et sans fils

---

Soutenue à Rennes, le 3 décembre 2009

Composition du Jury :

Président :	Prof. Gerardo Rubino
Rapporteurs :	Prof. Tijani Chahed Prof. Jean-Jacques Pansiot
Examinateurs :	Prof. Xavier Lagrange (Directeur de la thèse) Dr. David Ros (Encadrant de la thèse) Dr. Michael Welzl

To my wife, Yoli and my sons, Andrés Alejandro & Camilo Rafael

This work has been funded by:

- Convenio de cooperación internacional FONACIT-ECOS NORD, Código No. PI-2003000207 (código francés: V04M01)
- Fundación para el Desarrollo de las Telecomunicaciones (FIDETEL).
- Universidad de Los Andes.
- TELECOM Bretagne.
- The WiMAX Networking Engineering and Multihomming (WiNEM) project.

Acknowledgements

This thesis has been carried until its end thanks to many people I would profoundly like to express my gratitude. My wife Yolibeth, and my sons Andrés and Camilo have been at my side during all this special period of my life giving me support and love; I love you all...

My parents and my brothers, have also believed in me and gave me support during all these years. Thank you.

I would like to specially thank to Sally Floyd for great discussions during the conception of the ACK-CC mechanism. Thanks also to Janardhan Iyengar, who has also been available for fruitful discussions on the same subject.

I would like to thank to the *Fund for the Development of the Telecommunications of Venezuela* (FIDETEL), the *Universidad de Los Andes* and *TELECOM Bretagne* for this wonderful opportunity of doing my PhD studies abroad. Without their funding and support it would not be possible to make it.

I specially thank to Nicolas Montavont of TELECOM Bretagne, who has been an endless source of support, encouragement and guide during the most difficult period of my thesis.

Finally, I would like to thank my mentor David Ros, who has been a guide through the whole process of the PhD, and who has always known to make fair critics to improve the quality of the work and to push it further! Thank you.

Abstract

Nowadays the Transmission Control Protocol (TCP) is omnipresent in the Internet. TCP conveys roughly 90% of the Internet traffic from applications, such as web, email, file transfers and even multimedia traffic. In this thesis we develop methods and algorithms to improve the fundamental working of TCP: the clocking on data packets through the timely reception of acknowledgements (ACKs). Thus, we study two complementary aspects on data transmission, i.e., the controlling of the congestion of ACKs and the accelerating of the ACKs frequency. First, we describe in detail a mechanism, called ACK congestion control (ACK-CC), allowing TCP to deal with the impairments produced by the congestion of ACKs in bandwidth asymmetrical networks. Then, we evaluate the functioning of ACK-CC in both phases of the transmission, namely slow start and congestion avoidance. On the other hand, for the acceleration of the ACKs frequency (and thus the acceleration of data transmission) we assess the ACK division technique, through both simple mathematical modeling and simulation.

In this dissertation we also investigate the performance of TCP over WiMAX networks. In WiMAX access networks, a Base Station (BS) serves a number of Subscriber Stations (SSs) forming the so-called last-mile access. Since the bandwidth management in WiMAX systems is centralized at the BS, the SSs have to contend for bandwidth to send their Best Effort traffic (i.e., TCP) in the uplink (from the SS to BS). Hence, we focus on two particular issues: the impact of the WiMAX MAC-layer asymmetry on TCP performance, and the improvement of the bandwidth perception management on the BS to accelerate downlink data transfer.

Keywords: TCP, WiMAX, ACK congestion control, ACK division, bandwidth perception.

Résumé

Aujourd'hui, le Protocole de Contrôle de Transmission (TCP) est le protocole de transport le plus déployé sur Internet. En effet, TCP transporte environ 90% du trafic Internet issu d'applications telles que le web, le courriel, le transfert de fichiers et même du trafic multimédia. Dans cette thèse nous développons des méthodes et des algorithmes pour améliorer le fonctionnement d'un mécanisme fondamental de TCP : l'envoi de paquets de données a travers la réception d'acquittements (ACKs) de façon réguliere. Nous étudions deux aspects complémentaires dans la transmission de données, c'est-a-dire, le contrôle de congestion des acquittements et l'accélération de la fréquence d'envoi des acquittements. Tout d'abord, nous détaillons un mécanisme pour le contrôle de la congestion d'acquittements, dit ACK-CC, pour que TCP puisse gérer le ralentissement produit par la congestion d'ACKs á cause de l'asymétrie de bande passante dans le réseau. Ensuite, nous évaluons le fonctionnement d'ACK-CC dans les deux phases de la transmission : *slow start* et *congestion avoidance*. En ce qui concerne l'augmentation de la fréquence d'envoi des acquittements, nous étudions la technique de la division d'acquittements á travers un modèle mathématique simple et des simulations. Nous avons evalué un grand nombre des scénarios en nous focalisant sur la variation de la durée de la transmission pour pouvoir analyser l'impact des *divacks* dans les deux phases de TCP.

Dans cette thèse nous avons aussi examiné la performance de TCP dans les réseaux WiMAX. Dans les réseaux d'accès WiMAX une Station de Base (BS) gére un ensemble de Stations Clientes (*Subscriber Stations*, SS). Étant donné que la gestion de la bande passante dans un système WiMAX est centralisé dans la BS, les SS entrent en compétition pour transmettre le trafic dit *Best Effort* (transporté par TCP) sur la voie montante (des SS vers la BS). Ainsi, nous traitons deux problèmes : l'impact sur TCP de l'asymmétrie de la couche MAC du WiMAX, et l'amélioration de la gestion de la perception de la bande passante dans la BS pour accélérer le transfert de données.

Mots clés: TCP, WiMAX, contrôle de la congestion des acquittements, division des acquittements, perception de la bande passante.

Contents

CHAPTER 1

Modifications du mécanisme d'acquittement du protocole TCP : évaluation et application aux réseaux filaires et sans fils

1.1 Introduction

Aujourd'hui le protocole de contrôle de transmission (TCP) est le protocole le plus déployé sur Internet. En effet, TCP transporte 90% du trafic de l'Internet < que le web, le courriel, le transfert de fichiers et même du trafic multimédia. Des études récentes de métrologie rapportent que TCP et non seulement utilisé pour le transport de données, mais il est également utilisé pour le transport des flux "streaming" [27].

Dans le cadre de cette thèse, nous évaluons et nous proposons des méthodes et des algorithmes pour l'amélioration du transfert des données à travers la modification du schéma d'envoi des acquittements TCP. Particulièrement, nous explorons l'effet de la modification de la fréquence des acquittements et l'effet sur la performance de TCP. Par ailleurs, nous étudions un cas d'asymétrie particulière dans des réseaux IEEE 802.16 et son impact sur TCP.

1.1.1 Contexte

Dans TCP [23], les acquittements (ACKs) peuvent être utilisées afin d'accomplir des objectifs complémentaires. Tout d'abord, les ACKs sont utilisés par le noeud récepteur pour acquitter des paquets des données. Ceci est fait à travers les champs `ACK number` et `Window` dans l'entête TCP, les ACKs dirigent un mécanisme de fenêtre glissante qui permet à un récepteur *lent* de ralentir un émetteur *rapide*. Les acquittements TCP permettent à l'émetteur de détecter des pertes et de réagir en adaptant son débit d'émission à travers le mécanisme

d'adaptation de la fenêtre d'émission pour le contrôle de la congestion [2]. Le contrôle de flux et de congestion ont également un autre objectif : l'amélioration de l'utilisation du canal. Le contrôle de flux (basé sur un mécanisme d'adaptation de la taille de la fenêtre) permet à l'émetteur d'envoyer au réseau un bon nombre de paquets sans attendre à l'arrivé d'un ACK. En revanche, le contrôle de congestion de TCP essaie d'utiliser toute la bande passante possible à travers l'augmentation de la fenêtre de congestion quand il n'y a pas de pertes de paquets de données.

Quand un acquittement `ACK number` $= X$ arrive à l'émetteur, il est considéré comme neuf si $X > Y$, étant Y le numéro d'acquittement porté par l'acquittement précédent. C'est-à-dire, qu'un nouvel ACK confirme explicitement à l'émetteur que de nouvelles données sont bien arrivées au récepteur. Cette information est utilisée par les mécanismes de contrôle de flux et de contrôle de congestion pour faire évoluer leurs propres fenêtres dans le temps.

Le transfert des données avec TCP est basé sur le principe d'auto-régulation de l'horloge de TCP (c.f., "self-clocking") [12] : Tant que le récepteur envoie des ACKs avec un débit approximativement équivalent au débit de paquets de données, l'émetteur adaptera sont débit d'émission à la capacité disponible du réseau. Le rôle des algorithmes qui contrôlent l'augmentation de la fenêtre est de découvrir le débit "approprié" à un instant donné.

La propriété cumulative d'acquittement est importante. Un ACK avec `ACK number` $= X$ informe l'émetteur que toutes les données, avec un numéro de séquence $\leq X-1$ envoyé vers le récepteur, sont bien arrivées. Cette propriété fait en sorte que le récepteur puisse éviter l'envoi d'ACKs pour un ou plusieurs paquets de données. Le mécanisme d'acquittements retardés, ("Delayed ACKs") [6, 2] consistant en l'envoi d'un ACK tous les deux paquets de données, est basé sur cette propriété.

À partir de la première proposition de standardisation de TCP [23], plusieurs adaptations ont été proposées pour les nouvelles conditions du réseau. Tel que les différentes approximations le suggèrent, les solutions "middle-box" peuvent être utilisé pour compenser (filtrer ou reconstruire) les imperfections dans la fréquence des acquittements TCP à l'arrivée [5, 18, 16, 17]. Cependant, comme Internet à été profondément inspiré du principe de bout-en-bout ("end-to-end principle") [25], de nouvelles solutions à l'intérieur du protocole sont encore dans l'intérêt de la communauté.

Tel qu'il est dit dans la spécification [23], TCP à été conçu pour l'envoi d'un paquet de donnée par acquittement. L'algorithme d'acquittement retardés à été conçu pour : (1) la diminution du temps de traitement des petit paquets (En effet, la planification du traitement de paquets est une tâche à coût élevé pour le système d'exploitation [6, 13]). (2) pour réduire le trafic des acquittements dans les connections du type "telnet", c.f., on peut épargner 1/3 des acquittements en faisant la jonction dans un seul acquittement pour l'actualisation de la fenêtre et le caractère pour l'écho [6]. On pourrait également améliorer le rendement des liens partagés et des liens asymétriques, en envoyant un acquittement pour deux paquets de données [15, 3, 4].

La réduction du débit d'émission des paquets de données est la première conséquence de la réduction du nombre total des acquittements. Par exemple, en utilisant des acquittements retardés, on diminue de moitié le débit par rapport à la version d'un acquittement par paquet de données. Cette perte de débit peut être compensée par des techniques implémentées au niveau de l'émetteur dites de "byte-counting". Cependant, une paramétrage spécial est nécessaire pour contrôler l'agressivité de l'émetteur dans la phase de démarrage rapide, dont en général un acquittement couvrant X paquets peut induire une rafale de $2X$ paquets des

données. Tel qu'il sera présenté, les stratégies qui limitent la croissance de la fenêtre *cwnd* ou la taille de rafales réduisent les problèmes potentiels présentés dans cette phase.

Les modifications des règles pour la croissance de la fenêtre de congestion, tel que "Appropriate Byte Counting (ABC)", ont été proposées pour compenser la perte de performance produite par les acquittements retardés. Par ailleurs, ABC uniformise la croissance de la fenêtre de congestion. À l'origine, TCP a été conçu pour traiter des *segments*. Cette stratégie de conception a établi, entre autres, des guides pour la simplification du code source dans les noyaux des systèmes d'exploitation. Toutefois, la transmission des données est orientée flux. Donc, en interprétant l'argument de bout-en-bout on pourrait défendre la séparation des deux fonctionnalités et en même temps permettre leur coexistence dans le protocole [25]. Dans tous les cas, il doit avoir une relation entre la segmentation des données et l'ouverture de la fenêtre de congestion, c'est-à-dire, que la fenêtre devrait s'ouvrir en relation avec le nombre des segments qui ont été injectés dans le réseau. Une disparité entre les deux fonctionnalités (c.f., la segmentation et l'ouverture) induit des inconsistances profitables pour les récepteurs, et qui peuvent bien être corrigées par des techniques de "byte counting".

1.2 Contrôle de la Congestion des Acquittements

Idéalement, la fréquence avec laquelle les acquittements de TCP sont envoyés devrait être adaptée aux conditions du réseau. En effet, le mécanisme d'acquittements retardés peut être vu comme une manière simple de gérer la congestion du trafic des acquittements.

Le mécanisme de contrôle de congestion des acquittements est en charge d'adapter dynamiquement le taux d'envoi des acquittements à l'intérieur du protocole. Dans notre cas particulier, les pertes des acquittements sont considérés comme des signaux de congestion. De plus, nous considérons comme motivation principale pour la conception du mécanisme, les scénarios avec une asymétrie forte de bande passante ou de la contention dans des liens sans fils, pour lesquels TCP montre des problèmes de performance [4, 7, 18].

1.2.1 Spécification du Mécanisme

L'objectif du mécanisme est de contrôler le trafic des acquittements dits "purs" dans le chemin de retour, c'est-à-dire du récepteur à l'émetteur TCP. Même si notre approche est comme dans l'approche suivi par le protocole CCID 2 de DCCP [10, 19], c'est-à-dire purement de bout-en-bout, nous pouvons également tirer profit de l'information transportée par les marqueurs ECN [24].

Comme le montre la Figure 1.1, les bouts de la connexion TCP négocient l'utilisation du contrôle de congestion des acquittements avec une option TCP dit "ACK congestion control permitted". Cette option peut être uniquement envoyée dans des paquets d'ouverture de connexion qui ont le bit SYN activé. Remarquons dans la figure que si le noeud "A" reçoit de la part du noeud "B", l'option TCP indiquant l'accord pour contrôler la congestion des acquittements, les extrémités TCP peuvent contrôler la congestion des acquittements purs envoyés de l'extrémité "A" vers l'extrémité "B". De la même façon, il suffit d'avoir le cas contraire (l'échange de l'option commencé par l'extrémité A) pour contrôler la congestion d'acquittements dans le sens opposé.

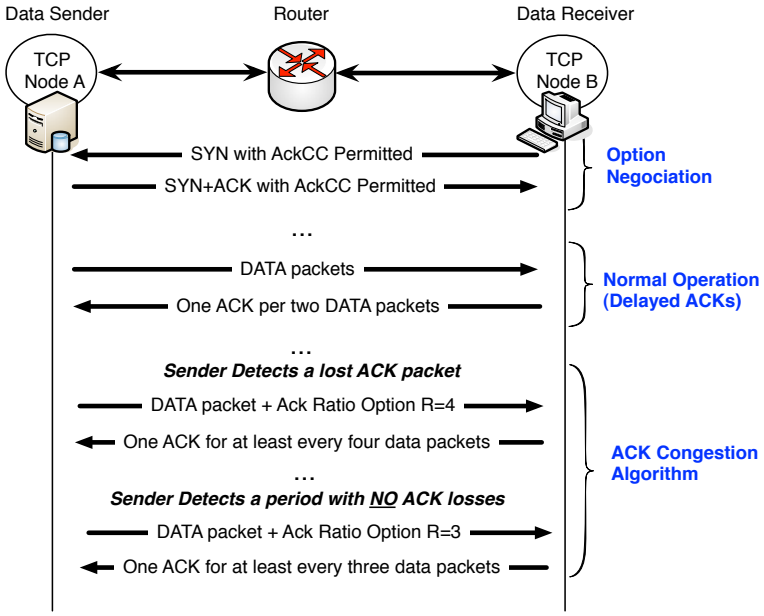


Figure 1.1: Acknowledgement Congestion Control, Node B is the connection initiator in a connection without ECN.

Le contrôle de la congestion des acquittements se fait, à la base, par l'émetteur en envoyant un Ratio (R) qui représente un nombre maximal de paquets de données par acquittement, mis à jour depuis le début de la connexion. Après l'ouverture de la connexion, l'émetteur informe la valeur par défaut $R = 2$. Ensuite, l'émetteur utilise une deuxième option dite "ACK Ratio" pour mettre à jour R chez le récepteur. Cette option est uniquement envoyée dans des paquets de données, donc l'émetteur peut bien distinguer si le récepteur a bien reçu chaque mise-à-jour du ratio.

Le ratio R indique aux récepteurs qu'il devrait envoyer un acquittement pour R paquets des données qui ont été reçus de façon ordonnée, à moins que le temporisateur des acquittements retardés se déclenche en avance. Le récepteur peut avoir ses propres limitations au moment de suivre un ratio quelconque, c.f., un récepteur avec des limitations d'espace dans le tampon de réception peut bien utiliser un ratio $L < R$.

Tel qu'il est signalé en [1], pendant la récupération de paquets perdus, le récepteur doit acquitter chaque paquet hors séquence en envoyant un acquittement dupliqué pour chaque paquet de données. Une fois que les trous dans les numéros de séquence sont remplis, partiellement ou complètement, le récepteur envoie sans tarder un nouvel acquittement pour signaler le nouveau numéro de séquence à l'émetteur.

1.2.2 Résultat de Simulation

Les réseaux d'accès ont souvent une asymétrie de bande passante. Ceci est dû au fait que les clients sont des consommateurs plutôt que de générateurs d'information [21, 20]. Donc, les scénarios de simulations traités dans cette section, ont été conçus pour tester l'impact et les gains produits par le contrôle de congestion des acquittements dans le deux phases de TCP : le démarrage rapide et l'évitement de la congestion.

1.2.2.1 ACK-CC dans le Démarrage Rapide

Puisque notre but est d'accentuer la phase de démarrage rapide, nous considérons le transfert de petits fichiers en émulant du trafic WEB. Comme nous le montrons dans la Fig. 1.2, nous employons une topologie dite "dumbbell" dont le lien central est asymétrique et les délais des les liens d'accès sont obtenus aléatoirement avec une distribution uniforme entre 0 ms et 10 ms.

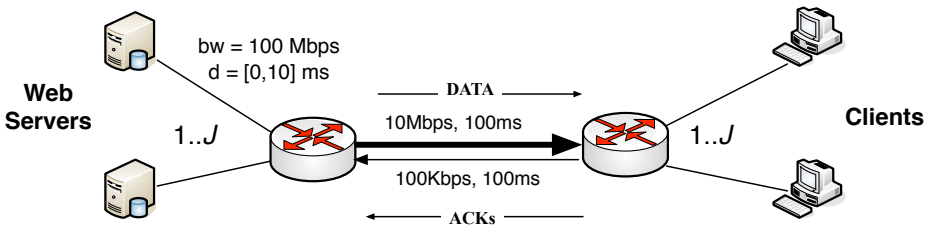


Figure 1.2: Asymmetric topology configuration for the Web Clients scenario.

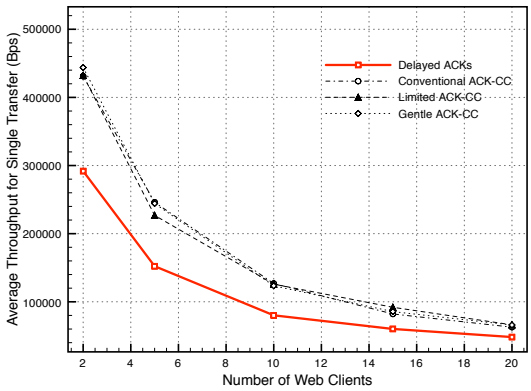
Dans ce scénario nous avons recréé les mêmes conditions de [5]. Donc, nous envoyons des fichiers dont les tailles varient entre 10 KB et 10 MB obtenus par une distribution uniforme. Nous avons fixé la taille du paquet de données à 1000 octets et la taille des acquittements à 40 octets. La taille du tampon de l'émetteur est suffisamment grande et donc un fichier quelconque d'un client WEB peut être envoyé dans la phase de démarrage rapide. En plus, le seuil du démarrage rapide est fixé à une valeur suffisamment large. Dans cette expérience, nous utilisons une stratégie simplifiée pour le traitement des paquets dans le tampon. Nous utilisons un tampon "drop-tail" dont la taille est de 60 paquets avec une bande passante de 100 Kbps.

Dans la Fig. 1.3a et Fig. 1.3b nous montrons le débit moyen par fichier sans "Appropriate Byte Counting" (ABC) et avec ABC respectivement. Nous observons que ACK-CC améliore la performance par rapport aux acquittements retardés. En outre, les Fig. 1.3c et Fig. 1.3d montrent le nombre total de fichiers envoyés pendant une période de 1000 s. Observons qu'il y a une amélioration notable de la performance définie par le débit (en octets par seconde) par transfert du fichier et par le nombre total de fichiers transmis dans la période de simulation. Plus précisément, ACK-CC permet une amélioration moyenne du débit de 1.5 fois pour le transfert de petits fichiers.

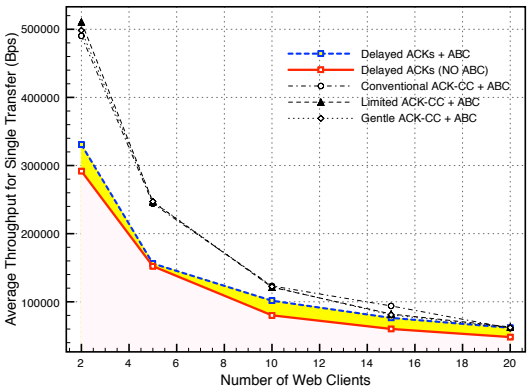
Cette expérience montre que l'algorithme des acquittements retardés empêche les paquets de données d'être libérés plus rapidement. ACK-CC est donc capable d'induire des rafales des paquets plus grandes chez l'émetteur qui sont absorbées par le tampon. Observons que le tampon du chemin des acquittements est du type "drop-tail", ainsi ACK-CC permet un partage équitable de la bande passante.

1.2.2.2 ACK-CC dans l'Evitement de la Congestion

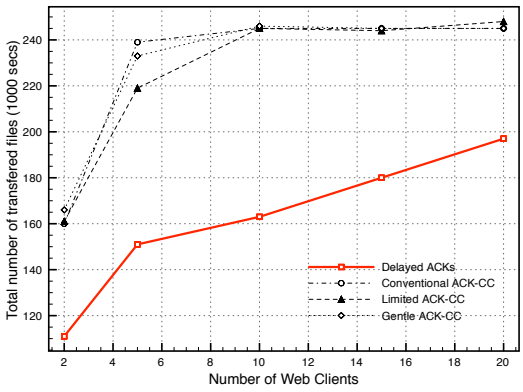
Dans cette section nous évaluons la performance d'ACK-CC dans la phase d'évitement de la congestion. Nous utilisons donc des longs transferts puisque dans ce cas la phase d'évitement de la congestion est active la plupart du temps. Tel qu'on l'a fait dans les essais de la phase de démarrage rapide, on évalue l'impact d'ABC dans des longs transferts. Remarquons que dans le cas de transferts courts l'utilisation d'ABC est efficace pour compenser la croissance



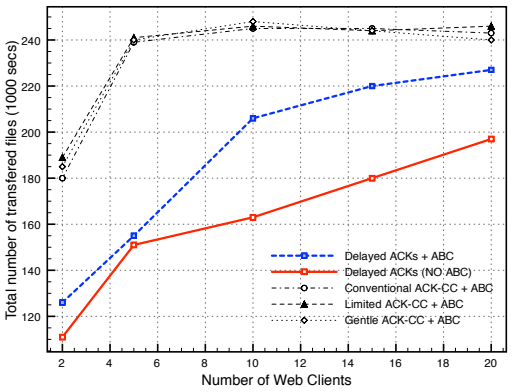
(a) Average throughput per file transfer *without* ABC.



(b) Average throughput per file transfer using ABC.



(c) Total number of transfered files *without* ABC.



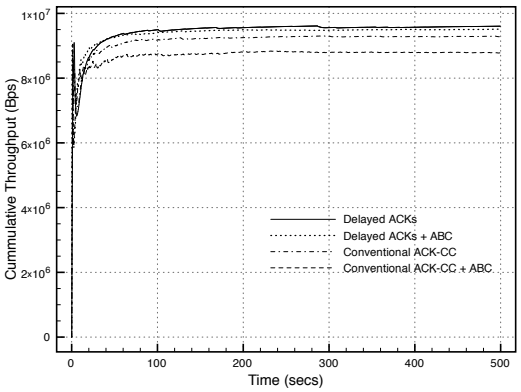
(d) Total number of transfered files using ABC.

Figure 1.3: Comparison between ACK-CC and Delayed ACKs for short files transfer.

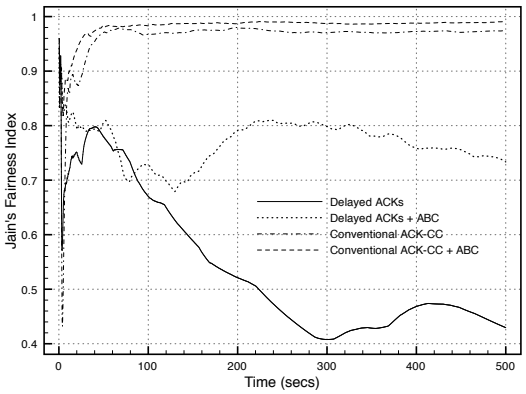
de la fenêtre. Les expériences avec de longs transferts confirment la tendance, en revanche dans certains cas l'utilisation d'ABC dégrade la performance.

Dans la Fig. 1.4 nous analysons la performance de flux TCP de longue durée. Pour cette expérience nous essayons avec 10 flux de longue durée dans une topologie comme dans la Fig. 1.2, de plus nous essayons deux tailles différentes de paquets de données pour intensifier l'asymétrie du réseau. Les paquets de données sont transmis dans la voie rapide et les acquittements dans la voie lente. Remarquons que cette expérience est un échantillon représentatif des autres charges de trafic étudiés. Dans la colonne gauche de la Fig. 1.4 nous montrons le débit accumulé et dans la colonne droite nous montrons l'indice d'équité de Jain pour différentes tailles de paquets. La première file montre la performance de transferts en utilisant un paquet de 1500 octets et dans la deuxième file pour un paquet de 1000 octets.

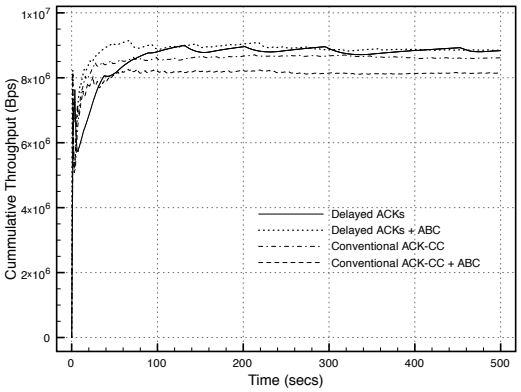
Pour l'algorithme des acquittements retardés nous observons dans les configurations du type asymétrique que, contrairement à l'intuition, les flux se synchronisent facilement est la performance est liée au délai d'accès. Autrement dit, il y a un débit plus grand pour les flux qui suivent les chemins les plus longs. Observons que la synchronisation est pire pour le transfert avec des paquets de données avec une taille plus petite (c.f., 1000 octets). Ce phénomène dit de phase [9], disparait en utilisant ACK-CC. Tel que l'on peut le voir dans les Figs. 1.4b et 1.4d, ACK-CC compense largement le manque d'équité comparé avec les flux qu'utilisent des acquittements retardés.



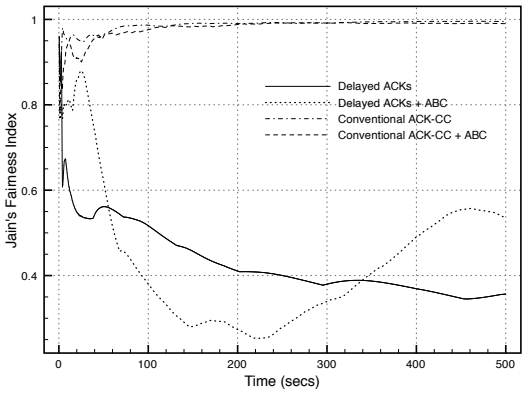
(a) Cumulative throughput for 10 downloading long-lived flows with a data packet size of 1500.



(b) Fairness for 10 downloading long-lived flows with a data packet size of 1500.



(c) Cumulative throughput for 10 downloading long-lived flows with a data packet size of 1000.



(d) Fairness for 10 downloading long-lived flows with a data packet size of 1000.

Figure 1.4: Jain's fairness index and throughput at the bottleneck for 10 downloading TCP flows.

Notons, toutefois, qu'il y a une différence dans le débit dû aux différentes configurations des clients TCP. L'utilisation d'ABC n'a pas un effet significatif dans le débit, mais il améliore notablement l'indice d'équité, même quand ACK-CC n'est pas utilisé. Dans tous les cas, quand l'agressivité de l'émetteur augmente, c'est-à-dire en utilisant ABC, ACK-CC ou tout le deux, il y a une petite perte dans l'utilisation du canal. Cette perte est justifiée par l'augmentation du taux de pertes de données, mais les gains en équité induit par ACK-CC sont beaucoup plus importants pour la performance de l'ensemble des flux.

1.3 Croissance du Taux d'Envoi des Acquittements

Comme on a vu dans la section précédente, TCP retarde l'émission des acquittements pour des raisons d'efficacité. Cependant, le récepteur TCP peut également envoyer plus d'un acquittement par segment sans casser la sémantique fondamentale des acquittements. Ce phénomène dit "division des acquittements" qui a été étudié à l'origine par [26], peut être exploité par des utilisateurs malveillantes pour améliorer de façon égoïste leurs propres performances. D'un autre côte la division des acquittements a été proposée pour améliorer la

performance de TCP dans des réseaux d'accès sans fils. Remarquons que dans les deux situations l'idée est de profiter de la croissance élevée de la fenêtre de congestion grâce à la division des acquittements.

1.3.1 Utilisation de la division des acquittements

La division des acquittements et les mécanismes pour retarder l'émission des acquittements (c.f., acquittements retardés ou ACK-CC), peuvent être vues comme un continuum de stratégies pour l'envoi des acquittements.

Appelons $r > 0$ le ratio entre nombre des nouveaux acquittements qu'un récepteur peut envoyer pour acquitter un seul segment de donnée, et le nombre des segments (de taille maximale) qu'un récepteur peut recevoir avant l'envoi d'un nouvel acquittement. Remarquons que quand $1/r$ est un entier, la valeur $R = 1/r$ équivaut au Ratio utilisé par DCCP [19] ou ACK-CC décrit dans la section précédente. Donc, le récepteur fait de la division des acquittements quand $r > 1$, un acquittement par paquet de données quand $r = 1$, et des acquittements retardés quand $r = 1/2$. La taille maximale de r est de $r = SMSS$, la taille maximale d'un segment ; cela correspond au cas limite, c'est-à-dire quand le récepteur acquitte un segment de taille maximale en envoyant un acquittement par octet.

Même si le contrôle de congestion et le contrôle de flux dans TCP sont dirigés par les acquittements, chacun a sa propre façon pour la mise à jour de la fenêtre. Le contrôle de flux est fait sous forme d'octets, c'est-à-dire que la séquence de numéros correspond aux octets dans le flux de données ; quand un nouvel acquittement arrive à l'émetteur, la fenêtre est mise à jour avec la quantité de nouveaux octets acquittés. D'un autre côté, le contrôle de congestion de TCP à toujours été sous forme de paquets. Ainsi, la croissance de la fenêtre de congestion dans une période de temps est une fonction du nombre total d'acquittements reçus, indépendamment de la quantité de données reçues [23, 1].

1.3.1.1 Division des acquittements pour les récepteurs gourmandes

La disparité dans la manière de traiter l'information dans les acquittements TCP peut conduire vers une forme de non équité sévère entre un ensemble des flux TCP [26]. Un flux TCP gourmand peut augmenter artificiellement son débit tout simplement en envoyant des acquittements divisés ; plus le nombre d'acquittements divisés par segment est grand (c.f., un r plus grand), plus le débit de croissance de la fenêtre de congestion est grand. Ce comportement permet au récepteur d'avoir une portion plus grande de la bande passante disponible dans le goulot d'étranglement, tel qu'il est illustré en [26].

Savage et al [26] montrent comment un client peut envoyer de façon agressive (presque sans aucun contrôle) des acquittements divisés pour télécharger de petites fichiers en passant à travers un goulot congestionné. Dans l'expérience un client envoie au moins 47 acquittements divisés au début du transfert et induit une croissance agressive de la fenêtre de congestion. Plus tard, 47 segments correspondant au fichier HTML sont arrivés dans une seule rafale. On peut, donc, en déduire que les acquittements retardés sont utilisés dans l'expérience comme point de comparaison. Finalement, le temps de transfert à été réduit de 70% approximativement à travers des acquittements divisés.

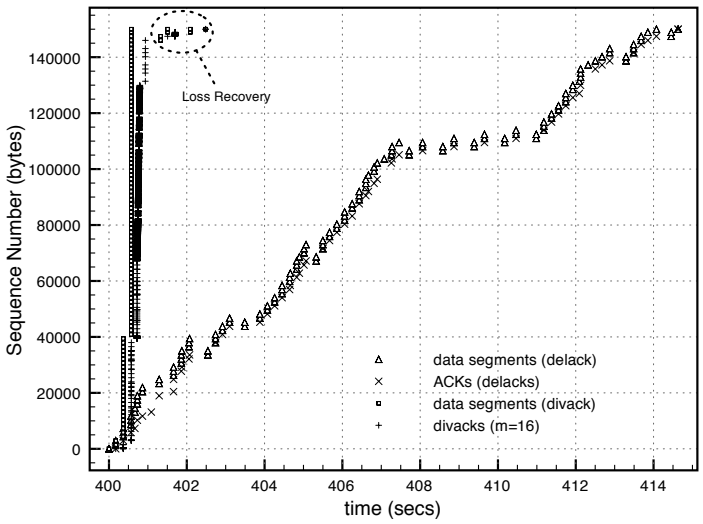


Figure 1.5: Comparison of a medium-sized file transfer (150 Kb) using Delayed ACKs versus *divacks* in slow start.

Dans une expérience par simulation, nous avons atteint des performances similaires à celles présentées dans [26] pour le transfert d'un seul fichier à travers un goulot d'étranglement. Dans la figure 1.5 nous montrons, donc, le transfert d'un fichier de 150 KB à travers une topologie "dumbbell". Pour cette expérience nous avons envoyé 16 acquittements divisés par segment pendant la phase de démarrage rapide. Nous avons amélioré le temps de transfert en environs 83% avec une congestion définie par 64 flux dans chaque sens (c.f. voie montante et descendante).

Nous observons que les acquittements divisés dans cette topologie sont efficaces pour 2 raisons. Premièrement, les deux rafales induites par les acquittements divisés ont été suffisamment chanceuses pour passer à travers le goulot d'étranglement avec juste très peu des pertes à la fin de la deuxième rafale. Deuxièmement malgré le temps supplémentaire d'aller-retour employés dans la récupération des 6 paquet perdus, il y a une amélioration significative dans le temps de transfert comparé avec la référence.

1.3.1.2 Division des acquittements dans les réseaux sans fils

Plusieurs chercheurs ont proposé de tirer profit de la division des acquittements pour améliorer la performance de TCP dans les réseaux sans fils, sans intervenir sur l'émetteur TCP. Dans les différents travaux, la division des acquittements est produite d'une façon contrôlée, pour aider l'émetteur dans la récupération rapide de la valeur de la fenêtre de congestion quand elle a été réduite de façon appropriée.

Jin et al. [14] proposent une méthode pour d'éviter les réductions de la fenêtre de congestion dues aux pertes aléatoires dans des scénarios d'accès sans fils. Ils supposent un émetteur TCP dans un hôte fixe (HF) dans la partie filaire du réseau et le récepteur TCP dans un hôte dans la partie sans fils. La station de base (SB) maintient un temporisateur parallèle au temporisateur de retransmission de TCP. Quand il y a une perte et le temporisateur de la SB se déclenche, la SB déduit que l'émetteur rentre dans la phase de démarrage rapide. En suite, la SB envoie des rafales d'acquittements divisés au HF quand les paquets retransmis

arrivent au récepteur. Les auteurs montrent par simulation que la fenêtre de congestion est rapidement récupérée à la même taille que l'émetteur avait avant la perte aléatoire.

Hasegawa et al. [11] ont développé une solution de bout-en-bout orienté récepteur pour la récupération après de mauvaises réductions de la fenêtre de congestion dues aux pertes aléatoires dans un réseau d'accès sans fils. Sa méthode considère trois aspects : un mécanisme pour la différentiation entre les pertes aléatoires et les pertes dues à la congestion, le contrôle de la durée de l'intervalle de temps pour générer et le contrôle du débit d'envoi des acquittements. Idéalement, les acquittements divisés, devraient seulement être envoyés quand il y a des pertes aléatoires. Ils ont essayé différentes méthodes pour la différentiation du type de pertes pour trouver la plus adapté à leur scénario. Pour le contrôle de la durée la période d'envoi des acquittements divisés, ils maintiennent dans le noeud mobile une estimation de la taille de la fenêtre de congestion. Ainsi, quand une perte aléatoire est détectée, le noeud envoie autant d'acquittements divisés que nécessaire pour faire agrandir la fenêtre de congestion à la taille atteinte juste avant la perte aléatoire. Comme l'envoi des acquittements dans la voie montante intensifie la congestion, ils observent et contrôlent la dynamique de la file d'attente du noeud mobile. Quand la file d'attente n'est pas pleine le numéro des acquittement divisés par segment augmente d'un par segment (c.f., $r = r + 1$). Au contraire, r est diminué par le nombre de paquets en attente dans la file. L'idée du mécanisme est, donc, d'estimer la bande passante disponible dans la voie montante.

Matsushita et al. [22] utilisent les acquittement divisés pour une adaptation rapide de la fenêtre de congestion dans des "handover" verticaux. Ils supposent que le noeud mobile peut détecter la bande passante disponible dans un nouveau réseau d'accès avec une capacité plus grande. Une fois que le noeud rentre dans le nouveau réseau, il envoie des acquittements divisés pour faire monter rapidement la fenêtre de congestion. Cet algorithme suppose que le récepteur est dans la phase d'évitement de la congestion.

1.3.2 Division des acquittements et ses effets dans la congestion

Pour l'évaluation de l'effet de la division des acquittements dans la performance de TCP, nous avons fait un étude exhaustive par simulation. L'un des buts était de répondre à la question : "La division d'acquittement toujours est-elle bénéfique ?". C'est-à-dire, dans quelle mesure le récepteur TCP peut améliorer systématiquement son "gootput" à travers l'envoi massive des acquittements, en supposant que l'émetteur répond toujours en ouvrant la fenêtre pour chaque acquittement reçu. Nous évaluons de même le cas quand le récepteur est plus conservateur, autrement dit nous contrôlons la manière dont on envoi les acquittements. L'objectif de cette étude est d'évaluer si la division des acquittements peut, de façon générale, améliorer les temps de transfert des fichiers de taille différente dans des conditions de congestion variable.

1.3.2.1 Les politiques de division des acquittements

Nous avons conçu et simulé différentes politiques d'envoi des acquittements en utilisant le simulateur réseau *ns-2*. Nous avons fixé comme références les politiques standard d'envoi des acquittements : acquittements retardés ($r = 0.5$) et un acquittement par paquet de donnée ($r = 1$). Les simulations ont été faites en utilisant les versions NewReno et SACK, cependant nous analysons le cas de SACK, dont les acquittements divisés améliorent le "goodput".

Tableau 1.1: ACK division policies.

Set	Policy	ACK division activated when the sender is in	Number of *divacks* sent when ACK division is activated	ACK sending mechanism when ACK division is deactivated
I	*divss1* *divca1* *divssca1*	SS CA SS+CA	m for every in-order data packet ($r = m$)	One ACK per in-order data packet ($r = 1$)
II	*divss2* *divca2* *divssca2*	SS CA SS+CA	m every other in-order data packet ($r = m/2$)	One ACK every other in-order data packet ($r = 0.5$)

Dans le Tableau 1.1 nous présentons un résumé des politiques utilisées. Les politiques peuvent être groupé en deux types. Considérons r comme le nombre moyen des acquittements divisés par paquet de donnée. Autrement dit, quand on fait la division des acquittements, le récepteur TCP utilise les politiques dites du type-I pour m acquittements divisés par chaque paquet de donnée (c.f., $r = m$). De l'autre côté, le récepteur utilise les politiques dites du type-II pour envoyer m acquittements divisés pour chaque deux paquets des données (donc, en moyenne nous avons $r = m/2$). En outre, un récepteur pourrait générer exclusivement des acquittements divisés quand l'émetteur est en dans la phase de démarrage rapide (SS), en évitement de la congestion (CA) ou dans n'importe quelle phase. Ce dernier correspond au cas d'un récepteur gourmand.

Pour les quatre politiques les plus conservatrices (*divss1*, *divca1*, *divss2*, *divca2*), c'est-à-dire quand on utilise d'acquittement divisés dans une seule phase, nous envoyons un acquittement par paquet de donnée pour les politiques du type-I et un acquittement pour deux paquets de données pour les type-II. Nous supposons que le récepteur est capable de connaitre l'état de l'émetteur avec un délai d'un demi RTT. Autrement dit, nous considérons le cas idéal avec une information parfaite (retardé).

Dans tous les cas, la division des acquittements est désactivé pendant un événement de réordennancement déclenché par une perte. Donc, le récepteur émet des acquittements divisés seulement quand il n'y a pas de trous dans les numéros de séquence reçues. De plus, les acquittements divisés sont toujours envoyés en rafales.

1.3.2.2 Scénario de Simulation

Pour l'expérimentation nous avons considéré une topologie dit "dumbbell" tel qu'on le montre dans la Fig. 1.6. Les liens du goulot d'étranglement ont un délai de 50 ms et une bande passante de X Mb/s. Nous avons simulé pour $X = 10$ et $X = 1.5$, mais comme nous avons trouvé des tendances similaires pour tous les deux, nous allons montrer des résultats pour $X = 10$. Nous utilisons des tampons "drop-tail" en mode paquet, avec une taille égale à la capacité du réseau (avec une taille de paquet de 1500 octets). Donc, nous avons utilisé un tampon de 83-paquets pour le lien de 10 Mpbs, et un autre de 13-paquets pour le lien de 1.5 Mbps. Les liens d'accès ont un délai de 1 ms et une bande passante de 4 fois la bande passante du goulot d'étranglement X.

Remarquons que tous les flux ont le même temps d'aller-retour. Nous justifions cette configuration particulier, parce que nous voulons éviter la favorisation naturel du débit pro-

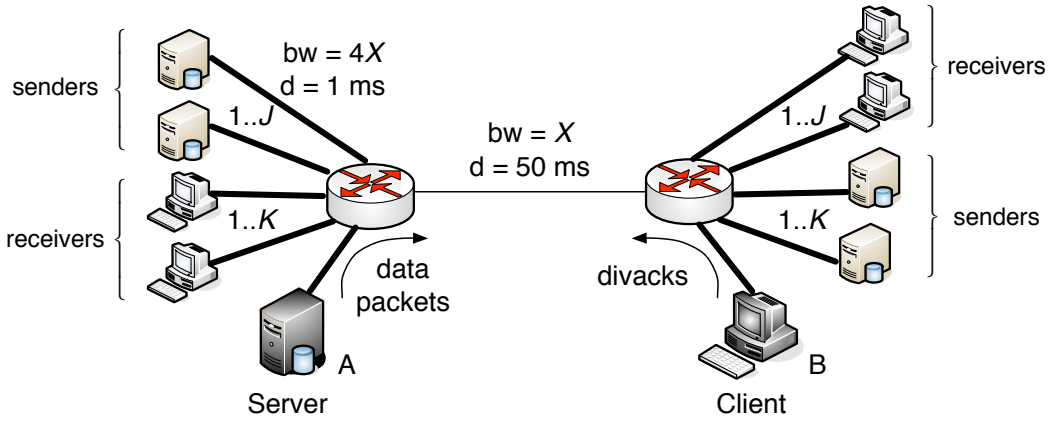


Figure 1.6: Topologie utilisé pour la simulation.

duit par les chemins les plus longs. Nous voulons que les gains en débit soient produit par les acquittements divisés.

Dans chaque scénario, il y a un seul récepteur qui fait de la division des acquittements (le "client" dans la Fig. 1.6), et un nombre fixe de flux TCP qui utilisent les acquittement retardées. Nous considérons donc deux types des scénarios, en termes du niveau de la congestion dans le chemin des acquittements divisés. Dans le premier type de scénario, il y a seulement J flux en arrière-plan dans la direction aller (les paquets de données vont dans la direction $A \rightarrow B$), et donc $K = 0$. Pour cette configuration il n'y pas de trafic des données dans le sens de retour. Dans le deuxième scénario, nous fixons $K = J > 0$ pour qu'il y ait des paquets de données dans les deux directions. Notre intention est de créer des différents configurations pour la compétition de ressources pour les acquittements divisés. Toutes les flux en arrière-plan sont de longue durée et utilisent l'algorithme SACK pour la récupération des pertes. Le but de ce choix est d'avoir une récupération efficace dans les flux en arrière-plan. En quelque sorte, nous voulons compenser avec SACK l'augmentation dans le taux des pertes, produite par le flux qu'utilise des acquittements divisés.

Pour tous les flux TCP, la fenêtre de réception est fixée dans une valeur suffisamment grande pour que les émetteurs ne soient pas limités par les récepteurs. D'autre part, le seuil du démarrage rapide est fixé aussi à une grande valeur.

Dans un scénario quelconque, le récepteur qui fait des acquittements divisés, télécharge un fichier après une période de réchauffement de 400 s pour le trafic en arrière-plan. Remarquons que chaque flux TCP du trafic en arrière-plan a un temps de démarrage qui va entre [0, 20] ms. Nous avons testé trois tailles différentes des fichiers : "petit" (15 kB), "moyen" (150 kB) et "grand" (1.5 MB). Nous utilisons des tailles différentes pour observer l'effet des acquittements divisés dans les deux phases : démarrage rapide et évitement de la congestion. Nous avons fait chaque transfert avec chacune des politiques pour l'envoi des acquittements divisés, de plus, nous avons également évalué les politiques des acquittements retardées et un acquittement par paquet de donnée. Donc, un scénario en particulier correspond a un choix de : taille du fichier, politique d'envoi des acquittements, nombre des acquittements divisés par paquet de donnée m (si des acquittements divisés sont envoyées), nombre de flux en arrière-plan pour l'aller J et pour le retour K. Pour chaque scénario, 30 simulations ont été faites.

Tableau 1.2: Effect of *divacks* on a single transmission performance when there is *one-way* and *cross* traffic congestion.

	divacks sent during :		
file size	*SS*	*CA*	*SS+CA*
small (15 kB)	improves	*no-op*	improves
medium (150 kB)	improves	*no-op*	improves
large (1.5 MB)	mostly degrades	OK	mostly degrades

1.3.2.3 Résultat de Simulation

Dans tout l'ensemble des simulations, la politique d'un acquittement par paquet de donnée a montré une meilleure performance que les acquittements retardé. Par conséquent, le temps de transfert pour n'importe quelle taille du fichier était plus bas pour un acquittement par paquet de donnée que pour des acquittements retardés. Donc, nous fixons ces deux temps comme référence pour l'évaluation de temps de transfert pour les autres politiques. C'est-à-dire, aucun flux se comporte pire que les acquittement retardés, et dans le cas d'amélioration de la performance, un flux quelconque doit se comporte mieux qu'un acquittement par paquet de donnée.

Quand les temps de transfert pour les fichiers étant transmisses avec n'importe quel algorithme de division des acquittements, sont entouré par les valeurs de référence, nous interprétons cet résultat comme un gain par rapport aux acquittements retardés, mais comme un perte par rapport à un acquittement par paquet de donnée. En outre, le fait de ne pas améliorer la performance d'un acquittement par paquet de donnée indique que les acquittements divisés dans certain cas sont pas nécessaire. Donc, dans les scénarios étudiés nous nous concentrons dans des cas spécifiques pour les pertes et les gains de performance quand nous utilisons des acquittements divisés. De plus, nous évaluons l'effet de l'augmentation du débit de flux des acquittements dans des autres flux qui partagent le chemin de retour ($B \rightarrow A$).

Dans le Tableau 1.2 nous présentons un résumé des temps de transfert que nous avons observé dans notre étude. Les entrées dans le tableau montrent des résultats généraux dans les deux cas de figure : avec de la congestion dans un seul sens ($K = 0$) et dans les deux sens ($K = J > 0$) pour des fichiers de tailles différentes.

Nous avons observé qu'il y a une claire amélioration dans les temps de transfert avec les acquittements divisés. Cette amélioration, de façon générale correspond d'une part, aux cas dont la majorité d'un fichier est transmit pendant la phase de démarrage rapide. C'est le cas de "petit" fichiers (15 kB) et des fichiers "moyens". D'autre part, les transferts des "grands" fichiers sont améliorés quand les acquittements divisés sont transmis dans la phase d'évitement de la congestion.

Du coté négatif, nous avons observé également que les acquittements divisés font facilement dépasser la capacité des tampons pour les émetteurs dans la phase de démarrage rapide. Cet effet est plus notoire quand les tampons du goulot d'étranglement sont plus petits que la capacité du réseau, ou quand la congestion en arrière-plan augmente. Donc, l'augmentation des pertes, dans certains cas spécifiques, augmente le temps de transfert pour des fichiers de taille moyenne, et dans la plupart des cas augmentent le temps de transfert pour les "grand"

fichiers. Nous avons observé également que, comme une tendance générale, il est nécessaire un débit des acquittements divisés (r) plus grand pour avoir une amélioration claire quand la congestion en arrière-plan est dans les deux sens.

1.4 Etude et amélioration de la Performance de TCP dans un Réseau 802.16

Le standard IEEE 802.16 pour les réseaux sans fils d'accès métropolitain, également connue par WiMAX, définit des mécanismes des haute-performance qui fournissent accès du dernier-mile à haute vitesse. Ce type d'accès est considéré par des uns, dans certains cas de figures, comme un remplaçant pour les technologies filaires et pourrait même offrir une connexion sans fils aux utilisateurs nomades et mobiles.

Dans un système IEEE 802.16, une station de base (BS) alloue des ressources réseaux aux stations clientes (SS). Le standard IEEE 802.16 permet une réservation à la demande, dynamique (trame par trame), de la bande passante de la voie montante (du SS vers la BS). De cette manière, le système est capable de gérer plusieurs types de trafic. Pour supporter les différents types de trafic (c.f. temps réel versus non temps réel), le standard IEEE 802.16 définit cinq classes de services qui permettent donner des priorités au trafic des utilisateurs. Avec ce mécanisme, nous pouvons conditionner le débit, le délai et la gigue expérimentée par les flux des données des utilisateurs. La mise en priorité est implémentée dans la couche MAC à travers un classificateur, un réordonnanceur et un système de contrôle d'admission.

1.4.1 Gestion de la bande passante de la voie montante

Dans les réseaux 802.16, le mécanisme de requête et assignation de la bande passante (BRGM) est le responsable de la gestion et satisfaction des besoins de bande passantes des SSs. Ce mécanisme permet aux SS signaler dynamiquement leurs besoins de bande passante des SS (qui varient dans le temps), la gestion de la perception du BS des besoins de bande passante des SS, et satisfaire leur besoin en octroyant de la bande passante.

Un SS informe ses besoins de bande passante au BS en envoyant un message de requête de bande passante (BW-REQ). Ce message signale la quantité des octets qu'attendent pour la transmission dans une file d'attente d'un SS. Le standard décrit des différents méthodes dont un SS peut envoyer des BW-REQ au BS.

Dans la méthode "polling", la BS interroge les SSs pour savoir leurs besoins de bande passante. En fait, le mécanisme de "polling" peut être déclanché de deux manières différentes : ou bien à travers de l'expiration d'un timer dans la BS, ou bien à travers une requête explicite géré par le bit "poll-me" dans le sous-entête de gestion des concessions. Le "polling" peut être fait de façon unicast, c'est-à-dire en allouant à un seul SS suffisamment de bande passante pour l'envoi d'un BW-REQ dans une rafale du SS vers le BS. De même, la BS peut interroger un groupe de SSs (dit "multicast polling") ou même toutes les SS (dit "broadcast polling").

"Multicast Polling" ou "Broadcast Polling" sont utilisées pour l'habilitation d'une période de contention pour faire passer les requêtes. Avec cette méthode, les SSs se battent pour l'envoi des messages de BW-REQ en utilisant les créneaux physiques disponibles dans une partie de la sous-trame de la voie montante, dites créneaux pour les requêtes de bande passante.

Comme les BW-REQ sont pas acquitté, un SS considère qu'une requête est perdue s'il n'y a pas une concession pendant une période de temps dit T16 dans le standard IEEE 802.16-2004. Par exemple, une perte peut être occasionnée par une collision (c.f., deux ou plus SSs utilisent le même créneau de contention), ou parce que le BS décide de ne pas octroyer de la bande passante sollicitée. Les pertes de requêtes sont géré par un algorithme de "backoff" binaire exponentielle qui permet la retransmission de BW-REQ perdues. Finalement, la méthode de "piggybacking" utilise le sous-entête de gestion de concessions pour faire passer une requête de bande passante à l'intérieur d'un paquet de donnée dans la voie montante.

Afin de satisfaire les besoins de bande passante des SSs, la BS doit suivre les requêtes de bande passante rentrantes, et les requêtes non résolues. A savoir, la BS doit avoir une *perception* de l'état des files d'attente dans chaque SS. Cette information peut être stockée dans le tableau d'allocation, également appelé tableau du gestion du trafic. Nous appellerons Gestionnaire de la Perception de Bande Passante (GPBP) à l'algorithme utilisé pour la mise à jour du tableau d'allocation. L'algorithme utilise comme information de base les tailles des files d'attente et les requêtes servies. Comme nous le verrons, les politiques utilisées pour la gestion de la perception de la bande passante peuvent introduire un effet de desynchronisation entre les SSs et la BS.

La BS satisfait les besoins de bande passante des SSs en allouant des créneaux physiques dans la sous-trame de la voie montante. Les allocations, définies dans la carte pour la voie montante (UL-MAP), sont généré par le ordonnanceur pour la voie montante dans la BS. Un algorithme particulier de ordonnancement utilisée est dépendent de l'implémentation. Le standard spécifie que les requêtes de bande passante doivent être faites par chaque connexion dans la voie montante, permettant une implémentation de l'ordonnanceur dans la BS qui garantie de la équité et la qualité de service. Ainsi, chaque requête doit être associée avec une connexion en utilisant un identificateur de connexion (CID). Cependant, nous remarquons que les concessions ne sont pas faites par connexion, mais par SS.

1.4.2 Gestion de la Perception dans les Reseaux 802.16

La BS peut changer la perception des besoins de bande passante des SS à travers la réception de BW-REQs. Le standard spécifie deux types de BW-REQ qu'un SS peut utiliser : les requêtes "incremental" et "aggregate". Une requête "agregate" communique à la BS l'état actuel (c.f., la quantité totale de bande passante requisse) d'une file d'attente d'une connexion particulière. Alors que, une requête "incremental" permet à un SS de demander plus de bande passante pour un CID particulier.

Le comportement par défaut d'un SS est d'envoyer des requêtes "incremental" quand la file d'attente d'une ou plusieurs connexions s'agrandissent. Les requêtes "aggregate" sont nécessaire pour assurer l'auto-correction propre au mécanisme de requête-concession de bande passante. En effet, la perception dans la BS pourrait perdre la synchronisation avec les besoins réels des SS, pour des divers raisons : des pertes aléatoires, des retards propre du système de gestion de la bande passante (dans la BS ou le SS), etc. Remarquons qu'un SS quelconque ne peut pas être au courant de la politique particulière est utilisée par la BS pour gérer les requêtes. Cette situation pourrait conduire vers une mauvaise interprétation des politiques, et donc une raison de plus pour la désynchronisation.

Afin de montrer comme un SS et son BS peuvent être desynchronisé, nous discutons la politique appelée *Décrément à l'Arrivé des Données avec traitement Immédiat du BW-REQ*

(DAD-i). Après nous proposons une modification au mécanisme appelé *Décrément à l'Arrivé des Données avec traitement Retardé du BW-REQ* (DAD-d) qui permet à la BS de décaler la mise à jour de la perception des besoins de bande passante d'un SS quelconque.

1.4.3 Evaluation de la Performance

Nous avons étendu le modèle WiMAX développé à l'Institut National de Standards et Technologie (NIST), qui est une extension du simulateur *ns-2*. Pour cette étude, nous avons fixé un nombre des flux de longue durée en variant aléatoirement le temps de démarrage de chaque flux pour éviter la synchronisation.

Nos résultat sont valides pour des conditions idéales de canal, pour une couche physique *WirelessMAN-OFDM* et en mode *Time Division Duplexing* (TDD) pour l'accès au media. L'ordonnanceur de la BS envoi les paquets de données dans une façon *round-robin*. En conséquence, pour N connexions dans la voie descendante, quand il y a des paquets à transmettre dans une file d'attente d'un CID i, la BS remplie la sous-trame de la voie descendante autant que possible avec les paquets de la connexion i. S'il n'y a pas suffisamment d'espace dans la sous-trame, le SS choisi doit attendre une ronde complète pour être encore servie. Cette stratégie est également appliquée pour les autres SSs dans le cas ou on a servi une ou plus SS et il y a encore de l'espace dans la sous-trame.

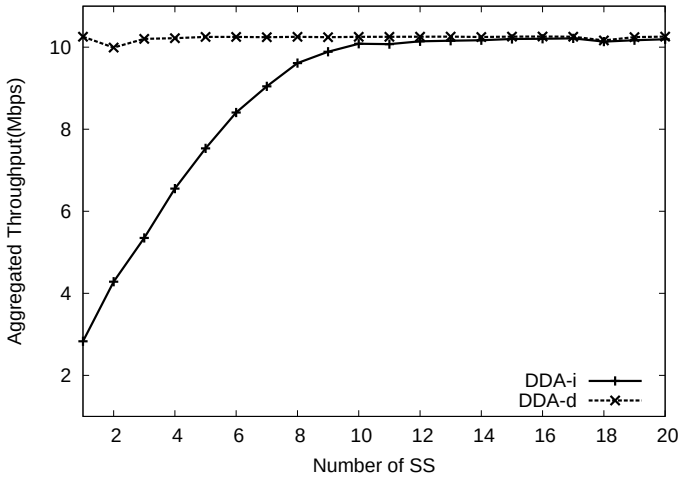
Les paramètres les plus importants dans la simulation sont :

- Durée de la trame : 5 ms
- Modulation et codification : 64 QAM 3/4
- Ratio DL :UL 0.5
- Version du TCP : Newreno
- Taille du segment TCP (sans entêtes) : 960 *octets*
- Configuration des files d'attente dans le BS et SS : 50 paquets, *drop-tail*
- Durée de la simulation : 1000 s

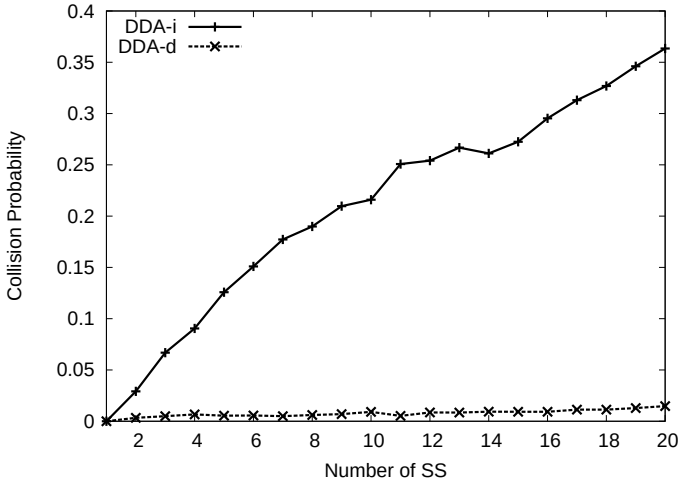
1.4.4 Résultats de Simulation

En utilisant cette configuration nous avons incrémenté le nombre total de SSs dans le système. Chaque SS démarre un flux TCP de longue durée pour le téléchargement des données d'un serveur localisée dans la partie filaire du réseau. Nous observons l'effet de la politique DAD-i dans la Fig. 1.7.

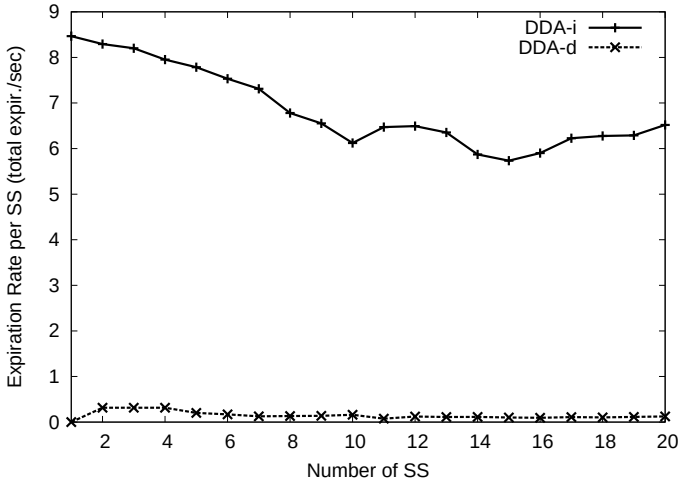
Tel que le montre la Fig. 1.7a, DAD-i est la seule politique qui présente une sous-utilisation significatif de la bande passante de la voie descendante quand il y a peut des SS. Le reste de politique se comportent de façon similaire mieux que DAD-i. La performance dégradé pour DAD-i est la desynchronisation propre à la politique DDA-i. En fait, comme il y a une sous-estimation des besoins de bande passante pour les SSs, la politique DAD-i permet la congestion dans la voie montante, c'est-à-dire pour les acquittements de TCP. Eventuellement les émetteurs TCP rentrent en timeouts. Ce phénomène est également observé à travers de l'augmentation de la probabilité de collision dans la Fig. 1.7b et le taux d'expiration du T16 dans la Fig. 1.7c qui sont notablement supérieur par rapport au DAD-d. De l'autre côté, le débit agrégé atteint le maximum pour la politique DAD-i quand il y a suffisament de SSs. Cette amélioration est possible parce que pendant que certaines stations sont interrompues,



(a) Aggregated Throughput



(b) BW-REQ Collision Probability



(c) T16 Expiration Rate per SS

Figure 1.7: Download-only traffic scenario, using exclusively aggregate requests (BW-REQs) for asking for uplink bandwidth.

des autres ont des paquets prêts à être envoyés. Finalement, la politique DAD-d garantie une base probabilité de collision et un bas débit d'expiration de T16 dans cet scénario.

1.5 Conclusions

Dans cette thèse nous avons investigué deux sujets complémentaires sur la performance de TCP. Dans un premier temps, nous avons traité à profondeur l'effet de modifications au mécanisme d'acquittement du protocole TCP. Nous avons réussi à améliorer la performance de TCP dans des réseaux asymétriques et nous avons évalué le comportement du protocole quand il y a une accélération dans l'envoi des acquittements. Deuxièmement, nous avons étudié la performance de TCP dans des réseaux WiMAX. Nos contributions peuvent être résumés comme suit :

- Nous avons définie complètement et nous avons évalué un mécanisme pour le contrôle de congestion des acquittements de TCP.
- Nous avons étudié l'effet de la division des acquittements quand elle est utilisée pour accélérer la transmission des données. En outre, pour la première fois nous avons confirmé que l'algorithme "Appropriate Byte Counting" régule la croissance de la fenêtre de congestion face aux acquittements divisés.
- Nous avons étudié et étendue la performance de TCP dans des réseaux WiMAX en proposant un mécanisme de perception de bande passante qui permet gérer la bande passante dans la voie montante de façon plus efficace.

1.5.1 Impact des modification au mécanisme des acquittements

La transmission des données est basée dans le principe du "clocking" des acquittements, consistant en l'arrivé opportun des acquittements qui reproduisent chez l'émetteur le pattern d'arrivé de paquets des données dans le récepteur [12]. Le "clocking" des acquittements est, donc définie pour deux processus : l'injection en continue de paquets de données par chaque acquittement reçu (également connue par le principe de conservation des paquets) et l'augmentation de débit à travers de l'injection de plus des paquets de donnée avec une fréquence d'un temps d'aller-retour. Dans cette thèse nous avons réuni de la évidence pour montrer que le "clocking" des acquittements est sensible aux modifications de la fréquence d'envoi des acquittements.

Quand une connexion TCP voit une augmentation du délai dans le chemin des acquittements, les paquets des données sont anormalement délayé. Autrement dit, le flux des acquittements détériore le débit des paquets des données, et donc le "clocking" devient complètement dépendent du pattern d'arrivé des acquittements. Dans cet cas les acquittements ne reproduisent plus le pattern de temps avec lequel les paquets des données ont été envoyée. Au contraire, les acquittements déforment le pattern d'arrivé des paquets des données.

Nous avons corrigé cet problème potentiel avec un mécanisme proposé à l'IETF pour diminuer le délai des acquittements TCP [8]. Le mécanisme de contrôle de congestion des acquittements mitige les effets produits par les acquittements délayés qui traversent le chemin lent. L'émetteur est capable de détecter la congestion des acquittements à travers de l'inspection des numéros de séquence. L'émetteur contrôle le ratio des acquittements puisqu'il est

bien placé pour détecter et corriger l'arrivée opportune des acquittements, en indiquant au récepteur une fréquence d'envoi adaptative.

De l'autre côté nous avons étudié l'effet produit par l'accélération du "clocking " des acquittements. Pour produire l'accélération nous avons utilisé une inconsistance du protocole consistant en l'envoi de plus d'un acquittement par paquet de donnée. La division des acquittements diffère de l'envoi courant des acquittements, parce que les acquittements divisés acquittent des parties plus petites qu'un segment et donc par fois, un acquittement n'est pas capable de déclencher l'envoi d'un paquet de donnée. Tel que la spécification de base le dit, les acquittements divisés augmentent la fenêtre de congestion d'un paquet pendant le démarrage rapide, et accélèrent considérablement l'augmentation de la fenêtre pendant la phase d'évitement de la congestion.

Nous avons évalué la division des acquittements à travers d'un modèle mathématique simple pour montrer quelques limites de la technique. Donc, les acquittements divisés induisent des longs rafales chez l'émetteur quand il est dans la phase de démarrage rapide. En conséquence, la taille du tampon pour les paquets des données doit être dimensionnée proportionnellement à l'agressivité induite par la technique, c.f., proportionnel au nombre des acquittements divisés par paquet de donnée.

Nous avons également évalué la possibilité de profiter de la technique systématiquement. Nous avons testé donc, plusieurs algorithmes pour l'envoi des acquittements divisés et nous avons observé que les acquittements divisés sont effectives s'ils sont envoyée dans la phase qui prédomine dans un transfert quelconque. Par exemple, pour les petites fichiers l'envoi des acquittements divisés dans le démarrage rapide est bénéfique, même que pour les grands fichiers dans la phase d'évitement de la congestion.

Finalement, nous avons constaté que le mécanisme "Appropriate Byte Counting" régule la croissance de la fenêtre de congestion dans toutes les deux cas, quand il y a peu et beaucoup des acquittements.

1.5.2 Transmission des données dans des réseaux WiMAX

Dans cette thèse nous avons traité également la transmission des données dans des réseaux d'accès IEEE 802.16, dans le cas résidentielle en utilisant la couche physique OFDM. Nous nous étions concentrés dans quelque sous-mécanismes spécifié dans le standard qu'on un effet sur TCP : la structure de la trame, la gestion de la bande passante dans la voie montante et l'organisation de la couche MAC.

Nous avons montré que la performance de TCP en termes de débit brute et débit utile, dépends du ratio de distribution de la bande passante pour la voie montante et descendante, la durée de la trame, la direction de flux de données, la charge et la modulation et codification dans la cellule.

Nous avons démontré également que, en dépendant des politiques pour le traitement et gestion de requêtes de bande passante, la station de base peut avoir une perception erronée des besoins de bande passante des stations clientes. Ce phénomène fait augmenter la probabilité de collision pour le BW-REQs, l'inéquité et la dégradation du débit. Pour palier cet problème, propre du mécanisme de requête-concession de bande passante, nous avons proposé une méthode pour la mise à jour efficace de la perception des besoins de bande passante des stations clientes. Cette politique, consistant en délayer le traitement des requêtes de bande

passante, a amélioré la performance de TCP de façon générale en améliorant la perception dans la BS.

Bibliography

[1] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. Internet Standards Track RFC 5681, IETF, 2009.

[2] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Internet Standards Track RFC 2581, IETF, April 1999.

[3] E. Altman and T. Jiménez. Novel delayed ACK techniques for improving TCP performance in multihop wireless networks. In *Proceedings of PWC 2003*, number 2775 in Lectures Notes in Computer Science, pages 237–250, Venice, September 2003. Springer.

[4] H. Balakrishnan and V. N. Padmanabhan. How network asymmetry affects TCP. *IEEE Communications Magazine*, pages 60–67, April 2001.

[5] C. Barakat and E. Altman. On ACK Filtering on a Slow Reverse Channel. *Internatinal Journal of Satellite Communications.*, 21:241–258, 2003.

[6] R. Braden. Requirements for Internet Hosts – Communication Layers. Internet Standards Track RFC 1122, IETF, October 1989.

[7] A. Eshete, A. Arcia, D. Ros, and Y. Jiang. Impact of WiMAX Network Asymmetry on TCP. In *Proceedings of IEEE WCNC 2009*, Budapest, Hungary, April 2009.

[8] S. Floyd, A. Arcia, D. Ros, and J.R. Iyengar. Adding Acknowledgement Congestion Control to TCP. Internet Draft draft-floyd-tcpm-ackcc-06, work in progress, 2009.

[9] S. Floyd and V. Jacobson. Traffic phase effects in packet-switched gateways. *ACM SIGCOMM Computer Communications Review*, 21(2):26–42, 1991.

[10] S. Floyd and E. Kohler. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. Internet Standards Track RFC 4341, IETF, March 2006.

[11] G. Hasegawa, M. Nakata, and H. Nakano. Receiver-based ACK splitting mechanism for TCP over wired/wireless heterogeneous networks. *IEICE Transactions on Communications*, E90-B(5):1132–1141, May 2007.

[12] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM'88*, Stanford (CA), USA, August 1988.

[13] I. Järvinen and M. Kojo. Improving processing performance of linux TCP SACK implementation. In *Proceedings of International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, Akihabara, Tokio, Japan, May 21-22 2009.

[14] K. Jin, K. Kim, and J. Lee. SPACK: Rapid recovery of the TCP performance using split-ACK in mobile communication environments. In *Proceedings of IEEE TENCON*, pages 761–764, Cheju, South Korea, September 1999.

[15] S. Johnson. Increasing TCP throughput by using an extended acknowledgment interval. Master's thesis, Ohio University, 1995.

[16] F. Keceli, I. Inan, and E. Ayanoglu. TCP ACK congestion control and filtering for fairness provision in the uplink of ieee 802.11 infrastructure basic service set. In *IEEE International Conference on Communications, 2007 (ICC '07)*, pages 4512–4517, June 2007.

[17] F. Keceli, I. Inan, and E. Ayanoglu. Achieving fair TCP access in the IEEE 802.11 infrastructure basic service set. In *IEEE International Conference on Communications, 2008 (ICC '08).*, pages 2637–2643, May 2008.

[18] H. Kim, H. Lee, S. Shin, and I. Kang. On the cross-layer impact of TCP ACK thinning on IEEE 802.11 wireless mac dynamics. *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pages 1–6, 2006.

[19] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). Internet Standards Track RFC 4340, IETF, March 2006.

[20] G. J. Lampard. Performance of TCP over cable modems and ADSL. In *Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST '98)*, pages 380–387, London, UK, 1998. Springer-Verlag.

[21] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, November 2009.

[22] Y. Matsushita, T. Matsuda, and M. Yamamoto. TCP congestion control with ACK-pacing for vertical handover. In *Proceedings of IEEE WCNC*, pages 1497–1502, New Orleans, March 2005.

[23] J. Postel (ed.). Transmission Control Protocol. Internet Standards Track RFC 793, IETF, September 1981.

[24] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Internet Standards Track RFC 3168, IETF, September 2001.

[25] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions in Computer Systems*, 2(4):277–288, 1984.

[26] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *ACM SIGCOMM Computer Communications Review*, 29(5), October 1999.

[27] K. Sripanidkulchai and B. Maggs. An analysis of live streaming workloads on the Internet. In *In Proceedings of ACM IMC*, pages 41–54. ACM Press, 2004.

List of Figures

List of Tables

Glossary

1-apdp	One ACK per Data Packet.
ABC	Appropriate Byte Counting.
ACK	TCP Acknowledgment.
ACK-CC	Acknowledgment Congestion Control.
ADSL	Asymmetric Digital Subscriber Networks.
AIMD	Additive Increase Multiplicative Decrease.
ARQ	Automatic Repeat Request.
ASN	Asymmetric Satellite Networks.
BDP	Bandwidth Delay Product.
BE	Best Effort.
BEB	Binary Exponential Backoff.
BPSK	Binary Phase Shift Keying.
BS	Base Station.
BW-REQ	Bandwidth Request.
CAC	Connection Admission Control.
CoS	Class of Service.
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance.
CTS	Clear to Send.
cwnd	Congestion Window.
DCCP	Datagram Congestion Control Protocol.
DCD	Downlink Channel Descriptor.
DCF	Distributed Coordination Function.
DDA-d	Decrease at Data Arrival with delayed BW-REQ handling.
DDA-i	Decrease at Data Arrival with immediate BW-REQ handling.
DDF	divacks's Data Flow.
DL-MAP	Downlink Map.
DOCSIS	Data-Over-Cable Service Interface Specification.
DPG	Decrease per Grant.
FCH	Frame Control Header.
FDD	Frame Division Duplexing.
FEC	Forward Error Correction.
FIFO	First Input First Output.
FTP	File Transfer Protocol.

MAC	Medium Access Control.
MCS	Modulation and Coding Scheme.
MSS	Maximum Segment Size.
nrtPS	Non Real Time Polling Service.
OFDM	Orthogonal Frequency Division Multiplexing.
PEP	Performance-Enhancing Proxy.
PMP	Point to Multipoint.
PS	Physical Symbol.
QAM	Quadrature Amplitude Modulation.
QoS	Quality of Service.
QPSK	Quadrature Phase-Shift Keying.
RBSCP	Rate Based Satellite Control Protocol.
RPG	Reset per Grant.
RTO	Retransmission Timeout.
rtPS	Real Time Polling Service.
RTS	Request to Send.
RTT	Round Trip Time.
SACK	Selective Acknowledgements.
SCTP	Stream Control Transfer Protocol.
SNR	Signal to Noise Ratio.
SS	Subscriber Station.
T16	Timer 16.
TCP	Transmission Control Protocol.
TDD	Time Division Duplexing.
TDM	Time Division Multiplexing.
TDMA	Time Division Multiple Access.
UCD	Uplink Channel Descriptor.
UGS	Unsolicited Grant Service.
UIUC	Uplink Interval Usage Code.
UL-MAP	Uplink Map.
WiMAX	Worldwide Interoperability for Microwave Access.
WWW	World Wide Web.