Trafics bidirectionnels et asymétrie : mécanismes pour optimiser les performances et maximiser l'utilisation des liens de l'Internet

Fatma Louati

Université de Nice Sophia Antipolis

N° d'ordre: ...

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE NICE SOPHIA ANTIPOLIS
Spécialité INFORMATIQUE

préparée et présentée par

Fatma LOUATI

Ecole doctorale: STIC
Laboratoire d'accueil: INRIA-SOPHIA ANTIPOLIS
Equipe d'accueil: PLANETE
Directeur de thèse: Pr. Walid Dabbous

Titre de la thèse:

Trafics bidirectionnels et Asymétrie: Mécanismes pour optimiser les performances et maximiser l'utilisation des liens de l'Internet

soutenue le 18 Fevrier 2004 devant la commission d'examen:

M. :	Michel	RIVEILL	Président
MM. :	Andrezj	DUDA	Rapporteurs
	Pierre	VINCENT	
MM. :	Farouk	KAMOUN	Examinateurs
	Ahmed	SERHROUCHNI	
M. :	Walid	DABBOUS	Directeur de thèse

Remerciements

Je remercie Pr. Walid DABBOUS pour m'avoir fait l'honneur de diriger cette thèse dans les meilleures conditions. Je lui suis reconnaissante de tout ce qu'il m'a appris, de sa disponibilité et de ses conseils. Sa gentillesse et sa sympathie m'ont profondément touchée. Qu'il trouve ici l'expression de ma profonde gratitude.

Mes vifs remerciements s'adressent tout particulièrement à Mr. Chadi BARAKAT pour les nombreux et fructueux entretiens qu'il a bien voulu m'accorder, pour sa collaboration active et son aide précieuse dans l'élaboration de ce travail.

Je remercie Pr. Pierre VINCENT et Pr. Andrezj DUDA d'avoir bien voulu accepter la charge de rapporteurs et ce, en dépit de leurs nombreuses occupations et responsabilités.

Je remercie Pr. Michel RIVEILL qui me fait l'honneur de présider ce jury. Je remercie également Pr. Farouk KAMOUN et Pr. Ahmed SERHROUCHNI d'avoir bien voulu juger ce travail.

Je remercie très chaleureusement tous les membres de l'équipe PLANETE pour l'ambiance de travail agréable et l'amitié qui s'est établie entre nous. Nos discussions cosmopolites pendant les repas m'ont permis de m'enrichir personnellement, je les en remercie vivement.

Mes sincères remerciement s'adressent à Mahmoud pour sa patience et pour la motivation et le courage qu'il a su me transmettre. Quoi qu'il arrive, je ne le remercierai jamais assez.

Je remercie également mes parents à qui je dédie cette thèse. Merci aussi à Hajer, Hela, Leila, Ahmed, Rafik, mes magnifiques nièces, ma chère gand-mère et les autres membres de la famille pour leurs encouragements. Merci aussi à Rim, mon amie de toujours. Un merci particulier s'adresse à mon oncle et ma tante Mustapha et Kahena pour leur précieux soutien pendant ces années passées en France.

Je tiens finalement à remercier vivement tous mes amis, voisins et toutes les personnes qui de près ou de loin ont contribué au bon déroulement de cette thèse. Je ne les citerai pas pour ne pas risquer d'en oublier mais je souhaite qu'ils trouvent ici l'expression de ma profonde gratitude.

Résumé de la thèse

Les réseaux asymétriques sont des réseaux ne présentant pas la même capacité en terme de bande passante sur le lien ascendant et sur le lien descendant. Si de plus nous avons un trafic bidirectionnel sur de tels réseau, paquets de données et paquets d'acquittement se partagent les ressources. Cette cohabitation de paquets pose d'énormes probèmes au protocole TCP. Afin d'y remédier, une multitude de solutions ont été proposées dans la littérature.
Dans cette thèse, nous analysons les problèmes qu'une telle situation engendre, puis nous décrivons quelques unes des solutions les plus prometteuses. Nous démontrons aussi par une étude de cas leur inéfficacité.
Nous proposons ensuite deux nouveaux ordonnanceurs ayant pour but commun de maximiser l'utilisation d'un lien asymétrique et satisfaire ainsi l'utilisateur de tels liens. Nous proposons **ACQ** de l'anglais *Adaptive Class-based Queuing*. ACQ manipule des agrégats de trafics, classés dans deux catégories et dont l'ordonanncement s'adapte au trafic qui passe afin d'avoir toujours une utilisation maximale du lien asymétrique. Nous proposons aussi **VAQ** de l'anglais *Virtual Ack-based Queueing*. VAQ suit une approche à granularité fine, c'est à dire qu'il manipule directement des paquets de données et d'acquittement en ayant toujours pour objectif de maximiser l'utilisation du lien asymétrique. Nous mettons en évidence dans cette thèse l'augmentation importante de l'utilisation des liens asymétriques obtenue avec ACQ et VAQ. Nous montrons d'autre part la robustesse de ACQ et de VAQ face à des changements dans les paramètres et la toplogie du réseau ainsi que leur impact favorable sur d'autres critères de la qualité de service.

*Mots-Clés:*TCP, trafics bidirectionnels, asymétrie de bande passante, mécanismes d'ordonnacements, fonctions d'utilité, satisfaction de l'utilisateur

Abstract of the thesis

Asymmetric networks are networks that do not have the same capacity in term of bandwidth available in their forward and reverse directions. Typical examples are asymmetric satellite networks and ADSL lines. If these kinds of networks are crossed by two-way TCP traffic, we are in a situation where data packets and acknowledgments (ACK) share the same scarce resource in the upload direction. This sharing implies lot of problems to the TCP protocol. In order to alleviate these problems, many solutions have been proposed in the literature. However, all these solutions try to improve per-

formances of one traffic at the expense of the opposite. For this reason, we propose in this thesis two solutions, that differ by their approach, and that have another objective: satisfy the user of a two-way traffic by considering at the same its upload and download traffic. This satisfaction is obtained by maximizing the utilisation of both links.
Our first solution is called Adaptive Class-based Queuing mechanism **ACQ** and aims to handle two-way TCP traffic over links that exhibit bandwidth asymmetry. ACQ runs at the entry of the slow link and relies on two separate classes, one for Ack packets and one for Data. ACQ proposes to adapt the weights of both classes according to the crossing traffic. We present also **VAQ** - for Virtual Ack based Queuing -, our second solution that uses a fine-grained approach at the packet level to schedule data and ACKs at the entry of the slow reverse link of the asymmetric access network. VAQ uses two parallel queues: one for ACK packets and one for data packets. It grants a credit for the data packets queue, and proposes to adapt this credit by monitoring the ACKs of the download traffic.
We show by simulations that our mechanisms are able to reach a good utilization of the available resources, managing then to maximize the satisfaction of the user. ACQ and VAQ are also robust when changing the network conditions.

Key Words: TCP, two-way traffic, bandwidth asymmetry, schedulers, work conserving, utility function, user satisfaction

Abréviation

AFA	Association des Fournisseurs d'Accès et de service Internet
ACQ	Adaptive Class-based Queuing
ADSL	Asymmetric Digital Subscriber Line
AF	Acknowledgement Filtering
AR	Acknowledgement Reconstruction
AQM	Active Queue Management
CARD	Congestion Avoidance by Round-trip Delay
CBQ	Class Based Queing
CBR	Constant Bit Rate
DVB	Digital Video Broadcasting
ECN	Early Congestion Notification
FIFO	First In First Out
FQ	Fair Queuing
FTP	File Transfert Protocol
IETF	The Internet Engineering Task Force
IP	Internet Protocol
IRC	Internet Relay Chat
MSS	Maximum Segment Size
PC	Personal Computer
PQ	Priority Queuing
PILC	Performance Implication of Link Characteristics
QoS	Quality of Service
RED	Random Early Detection
RFC	Request For Comment
RTT	Round Trip Time
SACK	Selective ACKnowledgements
TCP	Transport Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VAQ	Virtual Ack-based Queuing
VSAT	Very Small Aperture Terminal
WFQ	Weighted Fair Queuing
WRED	Weighted RED
WRR	Weighted Round Robin
WWW	World Wide Web

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

COMPTES D'ACCES A L'INTERNET OUVERTS PAR LES MEMBRES DE L'AFA:

	Abonnements RTC	**ADSL et câble**	**Heures de connexions**
Juin 2003	7.338.000	2.053.000	90.137.000
Mars 2003	7.490.000	1.807.000	94.617.000
Dec 2002	7.469.000	1.456.000	91.502.000
Sept 2002	7.425.000	1.040.000	80.150.000
Juin 2002	7.056.000	884.000	73.600.000
Mars 2002	6.990.000	734 000	80.895.000

Organisme : AFA France
Url : www.afa-france.com/chiffres/

Tous les jours, des comptes rendus de récentes statistiques sont publiés, ils renvoient tous les mêmes résultats à savoir une incessante croissance du nombre de liaisons haut débit achetées et du nombre d'utilisateurs connectés [15]. L'Internet a évolué de façon très importante ces dernières années et cette évolution touche aussi bien l'usage actuel du réseau, son architecture et toutes les piles protocolaires qui le constituent. Nos motivations dans cette thèse sont de s'assurer de toujours avoir une utilisation optimale des ressources disponibles, en particulier une utilisation optimale des liens du réseau. Une utilisation maximale des liens se traduit par des débits élevés des trafics circulant sur le maximum des liens qui constituent le réseau et permettrait une optimisation des performances de l'Internet.
Nous commençons dans la section qui suit par faire un état de l'art de l'Internet afin de rapporter les évolutions majeures qu'a connu récemment ce réseau. Cet état de l'art nous permettra de démontrer l'importance de notre domaine de recherche dans cette thèse.

1.1 Etat de l'art du réseau Internet

Au début, Internet a été conçu pour interconnecter des machines telles que des PCs de bureau, des stations de travail UNIX, et des serveurs. Le développement de l'usage d'Internet a été possible par l'apparition depuis 1994 de nouveaux acteurs économiques et techniques, les Fournisseurs d'Accès Internet s'adressant aux utilisateurs particuliers. Aujourd'hui, des machines informatiques non traditionnelles telles que des téléviseurs Web, des ordinateurs portables et des organisateurs personnels se sont greffées au fur et à mesure des années au réseau Internet. Toutes ces machines sont interconnectées par des liens de communications aussi différents que des câbles coaxiaux, des fibres optiques et des liaisons radio. Chaque type de liens possède ses propriétés en termes de capacité de bande passante, de délai de propagation et de taux d'erreur de bit (*Bit Error Rate*), par conséquent différents liens peuvent transmettre une même information à des débits différents.

Les liens de communications aussi ont évolué ces dernières années, et nous assistons à la prolifération de nouveaux liens tels que les liens satellites [73] et les liens ADSL [44]. Ces nouveaux types de liens ont de nouvelles caractéristiques que nous allons expliciter un peu plus loin dans ce chapitre et qu'il est nécessaire de prendre en considération afin d'obtenir le meilleur service du réseau.

Une fois connectées, les extrémités de connexions doivent pouvoir communiquer à travers un protocole de communication qui définit préalablement aussi bien les formats et l'ordre des messages échangés, que les diffèrentes actions qui doivent être effectuées lors de la transmission ou de la réception de messages ou de tout autre évènement. Deux protocoles se sont imposés comme les standards de l'Internet; il s'agit de IP *Internet Protocol* et de TCP *Transmission Control Protocol* [38, 34, 52]. IP assure les fonctions suivantes qui parviennent à fournir un service *Best Effort* sans garantie de qualité de service:

- La fonction de relais (*forwarding*) où le routeur prend l'information (ou paquets IP) d'un de ses liens entrants et la transmet sur l'un de ses liens sortants.
- La fonction de routage (*routing*) où le routeur choisit le chemin sur lequel doit transiter l'information afin de réduire les temps de transmission et d'augmenter les débits des connexions.
- La fonction d'ordonnancement (*scheduling*) où le routeur décide entre plusieurs paquets arrivants celui qui sera transmis en premier. L'ordonnancement FIFO *First In First On* est utilisé par le protocole IP pour garantir un service *Best Effort*. Cependant les politiques d'ordonnancement peuvent varier afin d'assurer quelques critères de qualité de service, elles sont décrites plus en détail plus loin dans cette thèse (c.F. Chapitre 2).

Le protocole TCP de son coté est conçu afin de rendre la transmission des paquets IP plus fiable. En effet, TCP est pourvu de mécanismes de contrôle de congestion et de recouvrement d'erreurs qui permettent une utilisation efficace du réseau Internet et une optimisation des performances des connexions qui y transitent. TCP considère les

connexions comme des séquences d'octets divisées en segments pour être transmis. Plusieurs types de segments peuvent transiter par un routeur. En considérant uniquement les phases de transfert des données, il existe deux grandes familles de paquets:

1. **Les segments de données** (ou par abus de langage paquets TCP): ce sont les paquets qui contiennent l'information proprement dites et qui circulent de la source TCP vers la destination ou récepteur TCP. Leur taille maximale (MSS *Maximum Segment Size*) est égale à l'unité maximale de transmission (MTU *Maximum Transmission Unit*) moins 40 [14]. Les valeurs des MTU varient de 576 octets pour une réseau X.25 à 17914 octets pour un *Token Ring* à 16 Mbit/s [19], par conséquent les paquets de données sont généralement volumineux. D'autre part, les segments de données répondent aux notifications de congestion de l'algorithme de contrôle de congestion du protocole TCP (c.f. chapitre 1) en adaptant leur débit de transmission.

2. **Les paquets "accusés de réception"** qui témoignent de la bonne réception de l'information par le récepteur et qui donc circulent du récepteur vers la source TCP. Ces paquets sont appelés tout au long de cette thèse paquets d'acquittements (de l'anglais *acknowledgement packets*). De même, l'action d'accuser la réception d'un paquet de données est appelée acquitter.

Internet est aujourd'hui la plus grande interconnexion de réseaux opérationnelle qui soit (c.f. le site caida.org). Cependant, Internet et la technologie TCP/IP ne cessent d'évoluer. Nous retenons dans ce qui suit deux types de tendances qui stimulent cette évolution de TCP/IP: les nouvelles applications et les nouveaux types de bande passante.

1.1.1 De plus en plus de nouvelles applications

Avec l'apparition des fournisseurs de services et la grande concurrence à laquelle ils sont confrontés, la notion d'Accès Internet s'est très rapidement étendue à celle de Services Internet. L'accès au réseau permet l'usage de l'ensemble des technologies interactives (Vidéo conférences, jeux de simulations...), de capacités multimédia en ligne (Audio et Vidéo à la demande) et la pratique du téléchargement de logiciels et de contenus. A la consultation s'ajoute la capacité d'émission d'information vers le réseau Internet ou vers les autres utilisateurs, par la messagerie, les forums interactifs, l'édition et la publication web/HTML.

D'autre part, un utilisateur de l'Internet se voit migrer vers de nouveaux modes de communication autres que la communication entre 2 individus par le biais de mail ou de "*Chat*". En effet, des bases d'informations sont aujourd'hui mises à la disposition du public telles que les serveurs web. De plus la communication de groupe a vu le jour avec les listes de diffusion pour les groupes de travail, les "*newsgroups*", les "*ytalk*" et autres "*irc*". Nous assistons aussi à l'émergence des collecticiels dans l'Internet avec la diversité des média (texte, image, son, vidéo), avec les algorithmes de compression, les typage MIME (*Multipurpose Internet Mail Extension*) et les technologies du Multicast et de la diffusion de groupe.

Avec tous ces nouveaux modes de communication et la notion d'accès au réseau il était en fait tout à fait normal d'assister à de nouvelles applications issues d'incessants nouveaux besoins de l'utilisateur. Téléphoner, faire ses courses et faire des audio ou vidéo conférences sont devenues des applications qu'un utilisateur effectue couramment via le réseau Internet. Aujourd'hui, l'accès aux services d'information, de jeux, de consultations d'extrait de compte ou d'achats en ligne bénéficient des apports et de la généralisation de nouvelles interfaces de types navigateur. Ces applications ont rendu obsolète toute logique de développement propriétaire. Rien que pour les applications d'audio-vidéo conférences nous pouvons énumérer les recherches sur des technnologies telles que vat (*Visual Audio Tool*), rat (*Robust Audio Tool*), fphone (*Free Phone*), ivs (*Inria Videoconferencing System*), vic (*Video Internet Conferencing*), wb (*White-Board*), nt (*Network Text*), sdr (*Session Directory*), *netmeeting*, etc. En effet, des applications comme l'audio-vidéo ne peuvent être introduites sur l'Internet sans modification de ses performances et les recherches portant sur ce sujet ne cessent d'augmenter. L'extension logique de l'intégration de la vidéo et de la voix à la messagerie électronique est un service de remise en temps réel de la voix et de l'image. Un tel service peut être utilisé pour faire de la téléconférence.

1.1.2 Concept de bande passante

La bande passante se mesure en Hertz; elle représente la largeur de bande de signal qui peut être transmise sur un lien, par abus de langage on dit qu'elle représente le débit ou la vitesse d'une connexion. Plus précisément, il s'agit de la quantité de données transmissible par unité de temps. La bande passante se mesure alors en bits par seconde (bit/s) et s'exprime souvent en milliers (K) ou en millions (M).
Pour choisir un type de connexion, l'utilisateur doit tenir compte du débit en émission et du débit en réception, car ils diffèrent souvent. Le débit en émission ou en mode ascendant (en anglais *upstream bandwidth*) est la quantité de données par unité de temps qui peut transiter de l'ordinateur d'un utilisateur jusqu'au commutateur du réseau. Le débit en réception ou en mode descendant (en anglais *downstream bandwidth*) est la quantité de données par unité de temps qui peut transiter en sens inverse. Il est souvent utile d'avoir un bon débit en réception. Le débit en émission, en revanche, importe surtout pour la transmission de fichiers ou l'hébergement de sites Web.

Plusieurs types de liaisons existent aujourd'hui pour permettre l'accès au réseau. Les connexions téléphoniques sont les plus répandues, les plus économiques et les plus faciles à utiliser, mais les moins rapides (les débits peuvent atteindre à peine 56 Kbit/s). De plus, une liaison de cette sorte monopolise la ligne téléphonique.
La liaison RNIS (Réseau Numérique à Intégration de Services) utilise quant à elle une ligne numérique louée. Elle établit un canal de transmission entre l'ordinateur de l'utilisateur et le fournisseur de services Internet. Contrairement aux liaisons basées sur des technologies analogiques moins fiables, les liaisons RNIS utilisent des technologies numériques et ont un débit stable ("garanti") de 64 ou 128 Kbit/s en émission et en réception. Cependant elles restent relativement onéreuses et inaccessibles par endroits faute de déploiements.

Les liaisons par câble utilisent en revanche le réseau de câblodistribution. Une telle connexion est permanente et ne produit aucun effet sur la réception des signaux de télévision. Les débits atteints sont élevés en émission et en réception (Jusqu'à 2 Mbit/s). Cependant les débits varient sensiblement selon le nombre de personnes qui se partagent simultanément le service dans un secteur donné.

Liaison ADSL

La liaison ADSL (de l'anglais *Asymmetric Digital Subscriber Line*) ou LNPA pour Ligne Numérique à Paire Asymétrique [44] utilise aussi une ligne téléphonique, mais son débit est nettement supérieur. La technologie ADSL permet de mieux utiliser la bande passante disponible sur la "paire torsadée" de la ligne téléphonique. Ceci est possible grâce à la technologie ATM qui est greffée derrière ces liaisons ADSL. Alors que les téléphones analogiques utilisent les fréquences comprises entre 300 et 3400Hz, les fréquences les plus élevées demeurent inutilisées. L'ADSL utilise les fréquences de 30KHz à 1.1MHz pour transporter les données et ne perturbe donc aucunement la qualité des appels téléphoniques.
De plus, d'après les études effectuées par les fournisseurs de services ADSL, un utilisateur résidentiel typique télécharge plus (par exemple en surfant sur un site web) qu'il n'envoie de documents sur l'Internet (par exemple en envoyant un courrier joint d'un fichier d'image ou de vidéo). Ces fournisseurs ont donc jugé qu'il vaudrait mieux privilégier la bande passante des flux dits "descendants" de la ligne asymétrique.
Les limites de débits théoriques sont de 1Mbits/s pour les flux dit montants et de 8Mbits/s pour les flux dit descendants. La véritable vitesse obtenue grâce à l'ADSL dépend de beaucoup de facteurs: le contrat avec l'Opérateur/FAI ADSL, l'interface du modem (Ethernet, USB...), les diverses caractéristiques de l'ordinateur, les logiciels utilisés, etc.

Les connexions permanentes ADSL permettent donc des débits relativement élevés en réception (De 1,5 à 9 Mbit/s). Elles ne monopolisent bien évidemment pas les lignes et donc n'interfèrent pas avec les communications vocales ou les transmissions par fax. Cependant le débit reste relativement bas en émission (De 16 à 640 Kbit/s) et le service inaccessible par endroits. Les connexions ADSL sont aussi confrontées aux erreurs dues à des atténuations de transmission.

Liaison par satellites

Dans certaines régions éloignées des réseaux des compagnies de téléphone pour les liaisons ADSL et dépourvues de câble TV, la liaison par satellite est souvent le seul moyen d'avoir un accès haute vitesse à Internet [73, 72]. L'Internet par satellite comprend deux types de stations, les stations émettrices (*uplink*) qui envoient les données et les stations réceptrices (*downlink*) équipées d'une antenne de réception pour pouvoir recevoir les données diffusées par le satellite. Les liens satellites ont plusieurs caractéristiques qui les différencient des liens terrestres, certaines de ces caractéristiques ont un impact négatif sur les performances du protocole TCP: un temps de retour

long, un produit *delai * bande passante* important, un taux d'erreurs de transmission élevé, l'utilisation souvent imposée de réseaux asymétriques afin d'atteindre le haut débit à moindre coût. Cependant les liaisons satellites sont dix fois plus rapides qu'un modem normal et offrent un grand nombre d'avantages (déploiement) favorisant ainsi leur intégration dans le réseau Internet. En effet, les liaisons par satellite ont l'avantage d'être accessibles partout dans le sens de réception de données. Elles bénéficient d'un débit en réception élevé (jusqu'à 64 Mbit/s pour les satellites LEO). Cependant elles sont onéreuses et le débit en émission est faible (De 28 à 128 Kbit/s), c'est pourquoi il y a souvent recours à des liaisons asymétriques.

Actuellement la grande majorité des services multimédia empruntant les liaisons satellites sont de par leur nature point multipoint. Elles génèrent un trafic qui est essentiellement asymétrique. Ainsi la quasitotalité des consultations sur le net suppose avant tout un transfert de données du serveur vers l'utilisateur, ce qui est encore plus vrai avec les serveurs audio et vidéo. Par conséquent, et à cause du coût élevé des équipements utilisés pour envoyer les données aux satellites, l'utilisation des réseaux asymétriques est souvent imposée. Un hôte connecté à un réseau satellite envoie des requêtes via un lien terrestre et reçoit les données qui lui sont destinées à partir d'un canal satellite.

1.2 Motivations

Le domaine des réseaux informatique est un domaine très vaste. Sécurité, mesures et modélisation des performances, réseaux locaux, réseaux sans fil, etc. La liste des sous-domaines est longue [15]. Cette thèse s'inscrit dans le cadre de l'amélioration des performances des trafics circulant sur l'Internet et l'optimisation de l'utilisation des ressources du réseau. Nous nous intéressons plus particulièrement à l'étude de trafic bidirectionnel passant par des liens asymétriques en terme de bande passante disponible:

1. Les trafics bidirectionnels sont des trafics ayant des paquets de données circulant de part et d'autre d'un lien. Leurs paquets de données et paquets d'acquittement se partagent par conséquent les mêmes ressources à savoir files d'attentes, liens physiques, etc. De tels réseaux sont de plus en plus présents dans l'Internet suite aux nouvelles applications qui transitent dans le réseau. Par exemple, le pair à pair ou *peer-to-peer* fournit d'énormes bénéfices aux compagnies qui l'utilisent, les applications se basant sur cette technologie (*Napster* entre autres) ont de très bonnes performances qui leur permettent de se multiplier. Un groupe de travail a même été spécialement créé par des firmes aussi prestigieuses que Hewlett-Packard, IBM et Intel, afin d'étudier les questions relatives au peer-to-peer telles que sécurité et robustesse (*Peer-to-Peer Working Group*).
   Cependant, les performances de TCP en cas de trafics bidirectionnels sont affectées. En effet, les acquittements subissent des retards ou des pertes dans le cas où leur flux est perturbé par la présence de paquets de données. La propriété *A*ck-Clocked de TCP est alors perdue. Les algorithmes de contrôle de congestion, de contrôle de flux et de recouvrement d'erreurs de TCP sont de ce fait moins

efficaces, les performances des connexions concernées sont médiocres et l'utilisation des ressources du réseau devient inacceptable.
Pour toutes ces raisons, l'étude des performances de TCP dans le cas de trafics bidirectionnels représente une de nos motivations dans cette thèse.

2. L'asymétrie de bande passante représente la seconde motivation de nos travaux. En effet, il existe une forte probabilité que l'utilisateur d'Internet soit connecté au réseau via une liaison asymétrique, c'est à dire une liaison à bas débit dans le sens ascendant (de l'utilisateur vers le réseau) et à très haut débit dans le sens descendant (du réseau vers l'utilisateur). Selon la section précédente, les modems standards reliés à une ligne téléphonique transmettent à 28,8 Kbps ou 33,6 Kbps, et ceux se conformant à la norme V.90 jusqu'à 56 Kbps. Le modem-câble offert par les entreprises de câblodistribution permet d'atteindre une vitesse de réception pouvant s'approcher de 1 Mbps. Le satellite direct permet de se relier à Internet à plus de 1 Mbps. Enfin, la technologie ADSL permet d'établir des liens de communication de plusieurs mégabits par seconde en exploitant l'infrastructure actuelle de fils à paire torsadée du réseau téléphonique. Dans ces trois derniers cas, la vitesse est dite asymétrique, car elle est plus rapide pour la réception que pour l'envoi.
Les conséquences de l'asymétrie de bande passante sur TCP sont explicitées en détail dans le chapitre 1 de cette thèse. Ce qui en ressort c'est la dégradation de performances du protocole impliquant des débits médiocres et une trop faible utilisation des ressources du réseau.

L'association des deux situations qui engendre donc un trafic bidirectionnel circulant sur un réseau asymétrique perturbe d'autant plus le protocole TCP [48] qui sera par conséquent doublement affecté. TCP n'a pas été conçu pour de telles situations et présente donc des difficultés à optimiser ses performances [83].

FIG. 1.1 – *Architecture courante des connexions engendrées par un utilisateur de l'Internet*

La Figure 1.1 est une illustration d'une telle situation de plus en plus fréquente de nos jours. Notre objectif dans cette thèse est de maximiser l'utilisation du (des) lien(s) asymétrique(s) sur lequel transitent les trafics bidirectionnels. Maximiser l'utilisation

du lien revient à maximiser une fonction d'utilité préalablement définie et qui sera explicitée plus loin dans cette thèse (chapitre 3 et 4). Ceci permettra en outre de maximiser les débits des différentes connexions et d'atteindre une satisfaction maximale de l'utilisateur de tels trafics.

1.3 Contributions de la thèse

Les réseaux asymétriques d'accès à l'Internet sont des réseaux ne présentant pas la même capacité en terme de bande passante sur le lien ascendant (qu'on appellera aussi le lien retour ou lien en émission), et sur le lien descendant (qu'on appellera aussi lien aller ou lien en réception). Si de plus nous avons un trafic bidirectionnel sur un tel réseau, les paquets de données et les paquets d'acquittements se partagent les ressources à savoir files d'attente des routeurs, bande passante, etc. Ce partage cause des dégradations des performances du protocole TCP. Afin d'y remédier, une multitude de solutions ont été proposées dans la littérature.

Dans cette thèse après une description de quelques unes de ces solutions telles que le filtrage puis la reconstruction des acquittements et le *Per-Flow Queing*, nous démontrons par une étude de cas leur inéfficacité dans le cas de trafics bidirectionnels. Nous proposons ensuite deux nouveaux mécanismes d'ordonnancement ayant pour but commun de maximiser l'utilisation d'un lien asymétrique et optimiser les performances des connexions qui y circulent. Nos ordonnanceurs diffèrent par l'approche qu'ils suivent pour ordonnancer les paquets de données et les paquets d'ordonnancements:

- Nous proposons **ACQ** de l'anglais *Adaptive Class-based Queuing* ou ordonnancement adaptatif de paquets se basant sur des classes de trafics. ACQ manipule des agrégats de trafics, classés dans deux files d'attente dont les poids relatifs à l'allocation de bande passante s'adapte au trafic qui passe afin d'avoir toujours une utilisation maximale du lien asymétrique.

- Nous proposons aussi **VAQ** de l'anglais *Virtual Ack-based Queueing* ou ordonnancement se basant sur les "tailles virtuelles" des paquets d'acquittements. VAQ suit une approche à fine granularité (*fine grained*), c'est à dire qu'il ne manipule plus des agrégats de trafics mais manipule directement des paquets de données et d'acquittements en ayant toujours pour objectif de maximiser l'utilisation du lien asymétrique.

Nous décrivons chacun de nos ordonnanceurs et par de nombreuses simulations, nous mettons en évidence l'augmentation importante de l'utilisation des liens asymétriques obtenues avec ACQ et VAQ en comparaison avec d'autres techniques d'ordonnancement entre paquets de données et paquets d'acquittements. Nous montrons d'autre part la robustesse de ACQ et de VAQ face à des changements dans les paramètres et la topologie du réseau ainsi que leur impact favorable sur d'autres critères de la qualité de service.

1.4 Structuration de la thèse

Cette thèse est organisée de la maniére suivante: dans le chapitre 2, nous commençons par faire un rappel concis sur le protocole TCP et les modifications les plus importantes qui y ont été apportées afin d'améliorer ses performances. Ensuite, nous faisons une description complète des réseaux asymétriques d'accès à Internet, des problèmes que rencontre TCP face à de tels réseaux et des principales solutions proposées dans la littérature pour y remédier. Nous insistons aussi sur l'impact de l'ajout d'un trafic bidirectionnel. Afin d'argumenter nos déclarations, nous faisons à la fin du chapitre 2, une étude de cas de trafic bidirectionnel passant par un lien asymétrique et montrons à travers cette étude la faiblesse de TCP ainsi que des solutions proposées face à une telle situation, mettant ainsi en évidence la nécessité de concevoir de nouvelles solutions pour y remédier.

Dans le chapitre 3, nous présentons notre première proposition, à savoir ACQ (*Adaptive Class-based Queuing*). ACQ est placé à l'entrée du lien bas débit. Il utilise deux classes de trafic, une pour les paquets de données et une pour les paquets d'acquittement et propose d'adapter les poids de chaque classe en fonction du trafic qui passe afin de maximiser l'utilisation du lien asymétrique et d'augmenter les débits des connexions. Nous rapportons ensuite les résultats de simulations prouvant l'efficacité de ACQ et l'augmentation de l'utilisation du lien que son application implique.

Notre deuxième proposition est présentée dans le chapitre 4: VAQ (*Virtual Ack-based Queuing*). VAQ est aussi placé à l'entrée du lien à bas débit la différence avec ACQ est l'approche que ce mécanisme suit. En effet, VAQ manipule les paquets individuellement, on dit qu'il suit une approche à fine granularité. VAQ utilise deux files d'attente, une pour les paquets de données et une pour les paquets d'acquittements, accorde un crédit à la file d'attente des données et selon un algorithme spécifique met à jour ce crédit afin de maximiser l'utilisation du lien. Nous démontrons dans le chapitre 4 avec les résultats de simulations l'augmentation de l'utilisation du lien obtenue avec VAQ.

Le chapitre 5 est consacré à l'étude de ACQ et de VAQ face à des conditions de trafics différentes. Nous varions les topologies des simulations le nombre de connexions, le nombre de liens asymétriques, etc. afin de vérifier la robustesse de nos mécanismes. Nos simulations sont incluses dans un très faible intervalle de confiance. Ce chapitre montre clairement que ACQ et VAQ sont des ordonnanceurs très robustes et que par conséquent, grâce aux excellentes performances qu'ils fournissent, leur utilisation est vivement recommandée.

Nous concluons cette thèse avec une discussion de nos propositions et quelques perspectives de travaux futurs.

Chapitre 2

Faiblesse de TCP face à un trafic bidirectionnel passant par un lien asymétrique

Comme décrit dans l'introduction, avec l'évolution des réseaux satellites, de l'ADSL et autres nouvelles technologies, nous assistons de plus en plus fréquemment à la situation d'un trafic bidirectionnel passant par un lien asymétrique. L'objectif de ce chapitre est d'étudier les performances de TCP dans un environnement bidirectionnel avec liens asymétriques et d'analyser les raisons d'éventuelles dégradations de performance. Pour ce faire, nous commençons par une étude des principes et algorithmes du protocole TCP. Nous rapportons aussi les différentes versions du protocole. Ensuite, nous décrivons la problématique apportée par une asymétrie de bande passante dans un réseau. Nous insistons sur les conséquences de trafics bidirectionnels dans un tel environnement. Plusieurs solutions ont en effet été proposées dans la littérature, nous rapportons dans ce chapitre quelques unes des plus pertinentes. Nous concluons ce chapitre par une étude de cas qui nous permet d'argumenter notre étude par des simulations élaborées avec l'outil NS [62].

2.1 Le protocole TCP

Le protocole TCP (Transmission Control Protocol) [38, 3] constitue les fondements de l'Internet d'aujourd'hui. En effet il assure la délivrance des paquets en séquence, le contrôle du débit de transmission ainsi que la retransmission des paquets perdus dans le réseau. Les algorithmes de contrôle de la congestion et de contrôle de flux permettent une utilisation efficace de la bande passante disponible et une stabilité du réseau.
TCP est un protocole basé sur un principe de fenêtre de congestion, dont la taille varie dynamiquement en fonction de l'état du réseau. La philosophie de TCP est de sonder pour la capacité disponible en augmentant la fenêtre (*Additive Increase*), et de réduire la fenêtre si une perte de paquets est detectée*Multiplicative Decrease*). TCP suppose que toute perte est dûe à une congestion du réseau.

TCP repose sur le principe de fenêtre glissante ou fenêtre d'anticipation (*sliding window*). Le protocole place un petit nombre de paquets dans la fenêtre et transmet tous les paquets qui se trouvent à l'intérieur. Ceci permet d'améliorer l'utilisation de la bande passante des réseaux en permettant à l'émetteur d'envoyer plusieurs paquets avant de devoir attendre un accusé de réception. Notons par W(t) la taille de la fenêtre à l'instant t. W(t) est égale donc au nombre maximum permis de paquets non acquittés. A la réception d'un paquet, le récepteur est supposé envoyer immédiatement un acquittement en retour (en fait ceci était vrai pour la première version de TCP. Les versions de TCP implémentées dans l'Internet attendent la réception de deux paquets avant d'envoyer un acquittement, ,c'est le mécanisme d'acquittements différés (*Delayed Acks*) que nous explicitons un peu plus tard dans ce chapitre). Ces acquittements sont cumulatifs, c'est à dire que le dernier acquittement d'une même connexion contient toute l'information utile sur les acquittements précédents pour pouvoir mener à bien la connexion. L'émetteur utilise cet acquittement pour déterminer l'instant où elle doit envoyer un nouveau paquet de données. Ce mécanisme repose donc sur le flux des acquittements pour réguler le débit des connexions; il est dit ***Ack-Clocked***.

FIG. 2.1 – *Evolution de la fenêtre de congestion de TCP Tahoe*

Pour éviter la congestion, TCP gère un seuil appelé fenêtre de congestion (*congestion window*), le protocole passe alors par deux phases schématisées dans la figure 2.1:

1. La phase de *slow start* ou de démarrage lent: lorsque l'émission de trafic commence sur une nouvelle connexion ou qu'elle reprend après une période de congestion, l'émetteur commence par une fenêtre de congestion limitée à un seul segment et incrémente la fenêtre de congestion d'un segment à la fois, chaque fois qu'un acquittement est reçu. L'augmentation de la fenêtre se fait donc exponentiellement afin d'identifier rapidement les éventuel goulots d'étranglement dans le réseau tout en permettant au récepteur d'établir un flux d'acquittement correctement espacés. L'augmentation de la fenêtre pendant cette phase suit ce qu'on appelle un *Exponential Increase*.

2. La phase de *congestion avoidance* ou de diminution dichotomique: le protocole se place en cette phase dès la détection d'une perte de paquets dans le réseau. En cas de perte d'un segment, l'émetteur réduit sa fenêtre de congestion de moitié, la fenêtre subit alors un (*multiplicative decrease*). Pour les segments qui restent dans la fenêtre, TCP augmente la temporisation de retransmission. Pendant cette phase, l'incrémentation des fenêtres se fait linéairement et lentement, la fenêtre augmente de 1 à chaque envoi d'une fenêtre entière (*Additive Increase*) jusqu'à arriver à la taille maximale de la fenêtre ou jusqu'à ce qu'une nouvelle perte de paquets soit détéctée.

En réalité, deux autres phases ont été rajoutées au déroulement du protocole TCP, ce sont les phases de *fast retransmit* et *fast recovery* que l'on explitera dans la section 2.1.2. Mais auparavant, et vu l'importance des paquets d'acquittement pour le protocole TCP, nous leur consacrons la section suivante.

2.1.1 Les acquittements dans TCP

Pour assurer la fiabilité du transfert des informations, TCP utilise un mécanisme d'acquittements et de retransmissions des paquets. Dans cette section, nous mettons en évidence l'importance des paquets d'acquittements pour le bon fonctionnement du protocole.
Un récepteur TCP transmet un acquittement à l'émetteur à toute réception d'un segment de données. Ceci permet d'obtenir un degré de fiabilité puisque l'émetteur retransmet tout segment qui n'a pas été acquitté. Puisque TCP est un protocole à fenêtre glissante (c.f. 2.1), les paquets acquittements entrants permettent les transmissions de nouveaux paquets de données. Les acquittements sont utilisés par l'algorithme de contrôle de congestion de TCP pour l'augmentation de la quantité de données que l'émetteur est autorisé à injecter dans le réseau. Comme décrit à l'origine dans [5], les récepteurs TCP génère un acquittement pour tout segment entrant. Un acquittement porte donc le numéro du paquet de donnée qu'il acquitte. Ces acquittements sont de plus cumulatifs et acquittent tous les segments précédents arrivés en séquence (dans l'ordre) au récepteur. Cette propriété de cumul des acquittements est fondamentale pour le bon déroulement du protocole, et ceci soit dans le cas où le réseau est faiblement chargé ou dans le cas où le réseau est fortement chargé:

– Si le réseau est faiblement chargé, l'information contenue dans chaque acquittement rend tous les acquittements antérieurs redondants. Cette redondance est en fait bénéfique dans ce cas puisqu'elle permet au protocole de se protéger contre des retards ou des pertes de paquets d'acquittements. En effet une perte ou un retard d'un acquittement ne perturbera pas la propriété *ack-clocked* de TCP et impliquera simplement un trafic de données en rafale.

– Si le réseau est fortement chargé, la propriété cumulative des acquittements permet l'enclenchement du mécanisme de *Fast Retransmit* (c.f. 2.1.2) de TCP qui va récupérer de ces erreurs sans perte d'informations. En effet, tout paquet d'acquittement entrant à un récepteur TCP porte toute l'information nécessaire au bon

fonctionnement des algorithmes de contrôle de congestion et de contrôle d'erreurs de TCP.

Dans le cas où un paquet hors séquence *out-of-order* arrive, un acquittement est transmis. Seulement, il n'acquitte pas le paquet qui vient d'arriver mais acquitte encore le dernier paquet de données reçu en séquence. C'est un acquittement dupliqué (*duplicate ACK*) qui sert à indiquer l'existence d'un problème à l'émetteur. Si un certain nombre d'acquittements dupliqués est reçu, l'émtteur considère qu'il y a congestion sur le chemin des données et agit en conséquence (c.f. 2.1.2). Jacobson a fixé ce nombre à trois.

Le traitement des paquets d'acquittements a connu des modifications afin d'améliorer encore les performances de TCP. Parmi ces modifications nous présentons les acquittements retardés et les acquittements sélectifs.

2.1.1.1 Mécanismes de retardement des acquittements (*Delayed ACK*)

Cette option de TCP est définie dans [5]. *Delayed ACK* donne à un récepteur TCP la possibilité de ne plus envoyer un acquittement pour chaque segment de données mais le récepteur doit envoyer un acquittement à la réception d'un second segment tant que le temps entre la réception de deux segments ne dépasse pas 500 ms.
Le mécanisme de *delayed ACK* donne à TCP l'opportunité de réduire sensiblement les coûts et temps de traitement des paquets, ce qui a un impact positif sur les performances de transferts TCP en rafales dans certains réseaux [37], en particulier les réseaux asymétriques (c.f. 2.2) [10]. Il permet aussi aux ressources d'être moins encombrées et mieux utilisées. Dans [66], il a été montré que le mécanisme *delayed ACK* est souvent utilisé dans des implémentations de TCP.
Ce mécanisme est utilisé comme une solution au problème de congestion sur le chemin des acquittements (c.f. 2.3.1.2), en effet il permet de réduire le nombre de paquets d'acquittements, libérant de l'espace et réduisant la congestion. Cependant, des retards excessifs sur les paquets de données peuvent perturber les délais aller retour et la propriété *ack-clocked* de TCP, c'est pourquoi il est important de limiter le retardement des acquittements.

2.1.1.2 Des acquittements sélectifs (*SACK*)

Les performances de TCP souffrent à cause de timers trop larges dans le cas d'une perte de plusieurs segments dans une seule fenêtre (c.f. 2.1.3.1), et ceci est dû aux acquittements cumulatifs. C'est pourquoi le principe de SACK a été introduit comme une option de TCP par Fall et Floyd dans [26]. Cette option a été standardisée en une RFC [61]. Elle permet au récepteur de donner plus d'information à l'émetteur à propos des données reçues et de lui dire exactement quelle information a été reçue. L'utilisation de cette option est négociée lors de l'établissement de la connexion. La spécification de SACK est sous la forme de blocs inclus dans les acquittements à chaque "trou" dans la séquence des octets reçus. Chaque bloc SACK spécifie le début et la fin d'une séquence contigüe de données reçues. L'émetteur utilise alors les blocs SACK pour en savoir

plus sur l'emplacement des espaces et retransmettre seulement les données non reçues [22, 60]. Notons que le nombre de blocs est limité par la longueur de l'entête TCP. L'utilisation des acquittements sélectifs améliore considérablement les performances quand le réseau présente un taux de pertes élevé au niveau physique (e.g. liaisons satellites, réseau sans fil...).

2.1.2 Les versions de TCP

Plusieurs versions de TCP existent [34], chacune opère de façon differente concernant les deux phases de contrôle de congestion. Dans la version **Reno** de TCP [67], les concepteurs du protocole TCP ont à juste titre ajouté les phases de *fast retransmission* et de *fast recovery* aux phases de *slow start* et de *congestion avoidance*, . Avant de décrire ces deux phases, nous rappelons que TCP doit générer un paquet d'acquittement immédiat, qui est appelé un acquittement dupliqué, lorsqu'un paquet n'est pas reçu dans l'ordre (un tel paquet est appelé *out-of-order*). Un acquittement dupliqué indique donc à l'émetteur qu'un paquet *out-of-order* est reçu. Chaque acquittement dupliqué indique aussi qu'un *nouveau* paquet est arrivé chez le récepteur, c'est à dire que ce paquet a *quitté* le réseau. Etant donné que l'émetteur ne sait pas si cet acquittement dupliqué est causé par un paquet perdu ou juste par un paquet *out-of-order*, l'émetteur attend de recevoir un nombre restreint d'acquittements dupliqués avant d'effectuer la moindre action. Si l'émetteur reçoit trois acquittements dupliqués à la suite, il interprète cela comme une indication de perte de paquets et effectue deux actions:

- Une action de contrôle d'erreurs: l'émetteur retransmet le paquet qui paraît avoir été perdu, sans attendre l'expiration du compteur de retransmission. C'est la phase de *fast retransmission* ou retransmission rapide. Afin d'éviter de retransmettre inutilement des paquets à cause du risque de déséquencement dans l'Internet, la phase de *fast retransmission* est déclenchée après la réception de trois acquittements dupliqués. Après cette phase, TCP initialise le seuil de la fenêtre de congestion à la moitié de la valeur courante.

- Une action de contrôle de congestion: Étant donné que des paquets sont encore dans le réseau, l'émetteur ne se remet pas en mode *slow start*. Mais, la taille de la fenêtre de congestion est divisée par deux, et pour chaque acquittement dupliqué reçu on augmente la taille de la fenêtre de congestion de 1 (on augmente donc la fenêtre de 3). Ceci oblige l'émetteur à stopper son envoi de paquet pendant environ la moitié d'un délai aller-retour (RTT), c'est la phase de *fast recovery* ou recouvrement rapide. Dès qu'un "nouveau" acquittement est reçu, la fenêtre reprend sa taille initiale (celle qu'elle avait pendant la phase de *fast retransmission*).

Dans la version TCP **NewReno** par contre [27], l'émetteur reste dans la phase de *fast recovery* lorsqu'un nouvel acquittement "partiel" se produit, l'objectif étant de pallier les pertes multiples dans une fenêtre. Un acquittement partiel est défini comme étant un acquittement cumulatif qui n'acquitte pas la fenêtre en entier mais seulement

une partie des segments déjà transmis au moment de passage à la phase de *fast recovery*. Dans Reno, les acquittements partiels permettent de sortir de la phase de *fast recovery* et de rentrer dans la phase *congestion avoidance*. Dans NewReno en revanche, les acquittements partiels constituent une indication de la perte du segment succédant le paquet acquitté. Chaque acquittement partiel déclenche donc la retransmission d'un paquet perdu. L'émetteur continue d'envoyer des segments, ce qui donnerait plus d'aquittements cumulatifs et permettrait une éventuelle *fast retransmission* sans passer par des expiration de compteurs (*timers*). TCP Newreno utilise mieux l'information contenue dans les acquittements et permet des performances comparables à celles de TCP avec des acquittements sélectifs (c.f. 2.1.1.2).

TCP **Vegas** est une autre version qui, au lieu de réagir à d'éventuelles congestion dans le réseau, tente de les détecter et de les éviter, permettant ainsi de mieux utiliser la bande passante disponible [59]. Cette version est en fait une alternative entre pertes de paquets et indicateurs explicites de congestion (ECN, c.f. 2.1.2.2). Le principe fondamental est CARD: *Congestion Avoidance by Round-trip Delay*. CARD remarque l'augmentation des files d'attente des routeurs dans le cas d'une congestion, et par conséquent remarque une augmentation des délais aller-retour (RTT: Round Trip Time). Il est donc possible pour un émetteur de conclure à une congestion dans le routeurs suite à une augmentation de son RTT et de réagir en conséquence. Cependant TCP Vegas peut être très sensible au bruit pour le calcul du RTT. De plus dans le cas de connexions Web qui sont très courtes, l'émetteur peut ne jamais avoir toutes les mesures nécessaires pour son calcul du RTT.

2.1.3 Autres Améliorations de TCP: Contributions des routeurs IP

Depuis sa création et face au succès fulgurant de l'Internet, TCP a subit quelques changements afin d'améliorer ses performances et de lui permettre de réagir plus efficacement à d'éventuelles pertes de paquets.

2.1.3.1 Gestion des temporisateurs (*timeout*)

Un des paramètres qui affecte les performances de TCP est le temporisateur de retransmission ou *timout* [65]: si ce paramètre est sous-estimé, TCP peut retransmettre inutilement des segments de données et s'il est surestimé, une perte sera détectée tardivement entraînant une moindre efficacité des algorithmes de récupération de TCP cités plus haut. A chaque fois qu'un émetteur reçoit un nouvel acquittement qui augmente la taille de sa fenêtre de congestion, le *timeout* est recalculé à partir de la moyenne ($SRTT$) et de la déviation moyenne ($RTTVAR$) d'un échantillon de délai aller retour ($SampleRTT$) [38]. Les implémentations actuelles de TCP (Tahoe, Reno, NewReno et SACK) incluent en plus le principe de backoff exponentiel qui consiste à initialiser la valeur du timeout au double de sa valeur pécédente à chaque fois qu'un paquet est retransmis [42].

2.1.3.2 Le mécanisme RED (*Random Early Detection*)

Comme nous l'avons explicité dans les paragraphes précédents, les sources TCP réagissent à la perte de paquets par la réduction de leurs fenêtres de congestion et par conséquent de leur débit d'émission. L'idée de RED [29] est de concevoir un ordonnanceur qui au lieu d'attendre qu'une file d'attente de routeur soit pleine pour rejeter les paquets qui arrivent, rejette de façon aléatoire les paquets quand le remplissage de la file atteint un certain seuil. [32, 74].
RED est un mécanismes de gestion active des files attente des routeurs (*Active Queue Management AQM*) [57, 28, 35] qui sont en fait des mécanismes de contrôle de congestion permettant aux routeurs de détecter activement des congestions imminentes et de le notifier aux extrémités du système. Cette notification peut être réalisée par le rejet d'un paquet (*Drop*), l'affectation d'un bit dans l'entête d'un paquet *ECN*[70] ou tout autre moyen compréhensible par les protocole de la couche transport de la source. Pour ce faire la méthode utilisée dans RED est de dropper aléatoirement ou marquer les paquets.

Un routeur RED calcule la taille moyenne de sa file d'attente. Cette taille moyenne est comparée à deux seuils, un seuil *minimum* et un seuil *maximum*.

- Si la taille moyenne est inférieure au seuil minimum aucun paquet n'est marqué,
- Si elle est supérieure au seuil max tous les paquets qui arrivent sont marqués,
- Si elle est comprise entre les deux seuils, les paquets qui arrivent seront marqués avec une probabilité proportionnelle à la taille actuelle de la file d'attente.

Donc, la probabilité de rejeter les paquets augmente au fur et à mesure que la taille moyenne de la file d'attente augmente. L'algorithme RED consiste en fait en deux grandes parties: l'estimation de la taille moyenne de la file d'attente et la décision de rejeter ou non un paquet qui arrive[1].

(a) Estimation de la taille moyenne de la file d'attente

(1) $avg \leftarrow 0$
(2) `for each paquet arrival`
(3) `    if the queue is nonempty`
(4) $\quad avg \leftarrow (1 - w_q)avg + w_q q$
(5) `    else`
(6) $\quad m \leftarrow f(time - qtime)$
(7) $\quad avg \leftarrow (1 - w_q)^m avg$
(8) `    endif`
(9) `endfor`

1. Il est à noter à ce niveau que l'estimation de la taille moyenne de la file d'attente ainsi que la décision de rejeter un paquet peuvent être appliquées pour des paquets, ignorant ainsi la taille des paquets, ou sur des octets et prenant ainsi en considération la taille des paquets arrivants.

Où *time* est l'instant courant et *qtime* est le début de la période *idle* de la file. w_q définit le poids de la file d'attente pour calculer la moyenne. Cette variable constitue une sorte de filtre qui permet de ne pas avoir une augmentation importante de la taille moyenne de la file dans le cas d'une augmentation à court terme de la taille réelle de la file (suite à une arrivée en rafale de paquet par exemple ou à une congestion transitoire). Une valeur de w_q trop grande ne permettra pas de filtrer les congestions transitoires et au contraire une valeur trop petite rendra le routeur incapable de détecter un début de congestion car l'estimation de la taille moyenne de la file d'attente ne reflètera pas rapidement les changements dans la file.

Suite à cet algorithme, le routeur calcule en fait le degré de "*burstiness*" qu'il pourra accepter dans sa file d'attente. L'algorithme prend en compte la période où la file est vide en évaluant le nombre m de petits paquets qui *auraient pu* être transmis par le routeur durant cette période. Après cette période, le routeur calcule la taille moyenne de la file comme si m paquets étaient arrivés à une file vide pendant la période *idle*.

(b) Décision de rejet de paquets

Dans la deuxième partie de l'algorithme, un routeur RED décide si oui ou non il aura à rejeter un paquet arrivant. Deux paramètres de RED, min_{th} et max_{th} figurent au premier plan dans ce processus de décision. min_{th} spécifie la taille moyenne de la file en dessous de laquelle aucun paquet ne sera rejeté, alors que max_{th} représente la taille moyenne au dessus de laquelle tous les paquets seront rejetés. Au fur et à mesure que la taille varie entre min_{th} et max_{th}, les paquets seront rejetés avec une probabilités qui varie linéairement entre 0 et max_p.

```
(1) count ← −1
(2) for each paquet arrival
(3)     Calculate the avg. queue size
(4)     if min_th <= avg <= max_th
(5)             increment count
(6)             calculate probability p_a:
(7)                   p_b ← max_p(avg − min_th)/(max_th − min_th)
(8)                   p_a ← p_b(1 − count ∗ p_b)
(9)             with probabilty p_a:
(10)                  mark the arriving paquet
(11)                  count ← 0
(12)    else if max_th <= avg
(13)            mark the arriving paquet
(14)            count ← 0
(15)    else count ← −1
(16)    endif
(17) endif
```

Cet algorithme montre l'importance des paramètres min_{th} et max_{th}: leurs valeurs

dépendent de la taille moyenne désirée de la file d'attente. Pour un trafic typiquement en rafale, min_{th} doit être assez grand pour avoir un taux acceptable d'utilisation du lien. La valeur de max_{th} dépend en partie du délai de propagation moyen permis par le routeur.

Floyd et Jabobson ont montré que RED apporte plusieurs améliorations par rapport à une file FIFO classique. Dans [84], les auteurs ont prouvé que dans un environnement unidirectionnel et asymétrique, TCP fournit des performances plus élevées avec RED. Cependant ce mécanisme est très controversé [53, 82, 31, 47, 64] et prouver son efficacité de façon formelle est très difficile dû à la complexité de la mise en place de ses nombreux paramètres qui peuvent donner des performances médiocres s'ils sont mal fixés. RED est déjà implémenté dans des routeurs disponibles commercialement.

2.1.3.3 Notification explicite de congestion (*ECN*)

Dans [20], Floyd introduit la notification explicite de congestion ECN (*Explicit Congestion Notification*) comme autre moyen pour détecter les congestions dans le réseau. Lorsqu'un routeur est congestionné ou sur le point de le devenir, il met à 1 un bit de l'entête du paquet puis le transmet. Un routeur qui opère en mode Random Early Detection (RED) est parfait pour faire ce genre de manipulations où dès que la taille moyenne de la file d'attente approche la taille maximale, la conclusion d'une congestion sera faite, et un paquet choisi aléatoirement se voit attribuer son bit ECN à 1. Cette notification est répercutée jusqu'à l'émetteur qui réduit alors sa fenêtre de congestion. ECN est donc capable de faire un contrôle efficace de congestion sans laisser les files d'attentes des routeurs se remplir et donc sans pertes de paquets. Le mécanisme ECN ainsi que les réactions de TCP à ECN sont des standards de l'IETF (*Internet Engineering Task Force*) [70].

2.1.4 Conclusion sur le protocole TCP

Depuis sa création durant les années 70, le protocole TCP a pour objectif premier d'améliorer le service simple de *best effort* fournit par la couche IP de l'architecture Internet. TCP est utilisé par un grand nombre d'applications de transfert de données (FTP, Telnet, SMTP, HTTP, etc.) et permet de ne pas implémenter de mécanismes de fiabilité pour chacune des applications facilitant ainsi considérablemnet le travail des programmeurs du réseau.
Un autre objectif de TCP est d'assurer un contrôle de flux et de congestion afin de ne pas surcharger les ressources du réseau suite à l'augmentation incessante du trafic. Le contrôle de flux est asssuré par le mécanisme de fenêtre de congestion au niveau des sources et destination, le contrôle de congestion lui est assuré par les paquets d'acquittements. L'association de ces deux contrôles a permis à TCP de constituer une partie importante du trafic de l'Internet, (95% du trafic est en effet constitué de trafic TCP) et constitue une garantie de sa stabilité pour le protocole. Comme explicité plus haut dans cette section, ce sont les paquets d'acquittements qui permettent au protocole

d'effectuer ce contrôle de congestion puisqu'ils indiquent aux émetteurs TCP l'état de congestion du réseau, auquel cas l'émetteur réagit en conséquence.

Seulement, TCP suppose que les seuls problèmes de congestion possibles sont ceux survenant sur le chemin des paquets de données. En effet, une perturbation qui se traduit par des pertes dans le flux des acquittements est traduite automatiquement par l'émetteur en une congestion sur le chemin des données. Les deux situations suivantes peuvent arriver fréquemmnent comme nous l'avons indiqué dans le chapitre 1, pourtant elles n'ont pas été prises en compte par les concepteurs du protocole TCP:

- Le cas de trafic bidirectionnel où des paquets de données partagent le chemin emprunté par les acquittements. Dans ce cas, les acquittments peuvent être fortement retardés ou même perdus.
- le cas d'asymétrie dans la bande passante des liens du chemin emprunté par les paquets de données et celui emprunté par les acquittement. Ces derniers peuvent aussi être retardés ou perdus.

Dans les deux cas, le flux d'acquittements n'est plus indicateur de l'état du chemin des données, d'où le danger pour le fonctionnement de TCP. Les conséquences de telles situations sont explicitées dans la section suivante.

2.2 Les réseaux asymétriques, les trafics bidirectionnels et les conséquences sur TCP

Nous retenons de l'étude précédente du fonctionnement du protocole TCP la propriété de cumul des acquittements. Nous avons montré l'importance de cette propriété pour le bon déroulement du protocole puisqu'elle permet d'avoir toujours un protocole *ack clocked*, régi par la cadence du flux d'acquittements.
Seulement les réseaux d'aujourd'hui ont évolué et il peut arriver que le flux des acquittements soit perturbé sur le chemin de retour impliquant ainsi une perturbation du fonctionnement de TCP [9, 10]. Dans cette section, nous définissons les notions de réseau asymétrique et de trafic bidirectionnel. Nous expliquons ensuite les conséquences de telle situations sur le fonctionnement de TCP.

2.2.1 Les réseaux asymétriques

Nous disposons d'un trafic circulant d'un émetteur à un récepteur comme le montre la figure 2.2. On appelle réseau asymétrique tout réseau ayant un chemin de retour pas suffisament rapide pour porter le flux d'acquittements générés par le récepteur TCP.
Notons:
- C_{aller}: la capacité disponible sur le lien aller
- C_{retour}: la capacité du lien retour
- t_{data}: la taille des paquets de données
- t_{ack}: la taille des paquets d'acquittemnets

FIG. 2.2 – *Schématisation d'une communication*

On définit le rapport normalisé de la bande passante (*the normalized bandwidth ratio*) r comme étant le rapport entre les différences de capacités des liens aller et des liens retour sur la différence entre les tailles des paquets de données circulant sur le chemin aller et les paquets d'acquittements circulant sur le chemin retour:

$$\mathrm{r} = \frac{\frac{C_{aller}}{C_{retour}}}{\frac{t_{data}}{t_{ack}}}$$

r représente le nombre minimal de paquets de données qui doivent être acquittés par un seul acquittement pour ne pas saturer le lien de retour. En d'autres termes, si le récepteur génère des acquittements avec une cadence supérieure à 1 acquittement tous les k paquets de données reçus, alors le chemin de retour sera congestionné avant le chemin d'aller. Ceci réduira fortement le débit dans le chemin d'aller puisque les acquittements ne sont plus capables d'indiquer les éventuelles congestion: le *ack-clocking* de TCP est rompu. r est d'autant plus important lorsque le chemin retour dispose de grandes capacités de stockage. En effet, si les acquittements ne sont pas perdus sur le chemin retour, dès que r dépasse 1 (ou 0.5 si le mécanisme *delayed ACK est utilisé*, le *ack-clocking* est rompu.

2.2.1.1 Le problème de regroupement d'acquittements (*Ack Compression*)

Comme cité dans la section précédente, le protocole TCP se base sur les acquittements pour réguler son trafic et contrôler la congestion. Un réseau ayant TCP comme protocole de transport est comme une boîte noire et, pour un émetteur transmettant des paquets à travers un tel réseau, les seules indications de l'état du réseau sont les acquittements qui lui parviennent de la part du récepteur. Les acquittements sont donc en fait une sorte d'horloge interne du réseau.
Un lent chemin de retour entraîne une congestion au niveau du trafic d'acquittements. Dans le cas où les files d'attente des différents routeurs intermédiaires permettent une grande capacité de stockage, la congestion sur le lien retour résulte en un retard important des acquittements. Dans le cas où les files d'attente sont de tailles réduites, la congestion sur le lien retour entraine une perte d'acquittements. Un retard d'acquittements favorise et précipite le recours de TCP au *timeout* et une perte d'acquittements augmente le nombre d'acquittements dupliqués. Dans les deux cas, l'émetteur

interprète alors ce retard ou cette perte comme un problème sur le lien aller, et réagit en conséquence selon le mode où il se trouve:

- Si l'émetteur est en mode *slow start* et qu'il reçoit une notification de congestion dans le réseau (sous forme d'expiration de *timer* ou d'acquittements dupliqués), les performances de la connexion seront fortement dégradées [12]. En effet, l'émetteur conclue que le seuil de *slow start* actuel est trop élevé par rapport aux capacités du réseau. Il le réduit et reprend rapidement la transmission avec le débit réduit en se mettant en mode *congestion avoidance*. L'émetteur augmente alors plus lentement sa fenètre, et le débit finalement atteint sera médiocre.

- Si l'émetteur est en mode *congestion avoidance*, il interprète la perte ou retard des acquittement comme une notification de congestion dans le réseau, et réduit immédiatement sa fenêtre de congestion. Selon les versions de TCP, l'émetteur peut alors enclencher un mécanisme de *fast retransmission*, ce qui va générer encore plus de paquets d'acquittements et saturer encore plus le lien de retour. Cette situation implique une importante dégradation de performances de TCP.

On appelle phénomème de "**regroupement ou compression d'acquittements**" le retard ou la perte de paquets d'acquittements suite à une congestion sur le lien de retour (*the Ack Compression Problem*). Les conséquneces néfastes [12, 56] de ce problème sur le fonctionnement du protocole TCP sont présentés dans la section suivante.

2.2.1.2 Conséquences sur le fonctionnement de TCP

TCP est très sensible à des retards ou pertes d'acquittements [11]. Le problème de *Ack Compression* est en effet à l'origine de phénomènes dont vont souffrir les émetteurs et récepteurs TCP.

(a) **Lenteur de l'incrémentation de la fenêtre de congestion**
Un émetteur TCP se base sur l'arrivée des acquittements pour augmenter sa fenêtre de congestion, lui permettant ainsi de "sonder" la capacité du réseau pendant la phase *slow start* et de réguler son débit de transmission à cette capacité pendant la phase *congestion avoidance*. Dans un réseau asymétrique, les acquittements arrivent à l'émetteur en nombre réduit puisqu'une grande partie de ces acquittements est perdue au niveau du lien saturé. On voit alors un ralentissement de l'incrémentation de la fenêtre de congestion. Ce ralentissement entraîne des débits faibles malgré l'existence de ressources suffisantes au niveau du chemin des paquets données. Ceci peut être inacceptable pour certaines applications exigentes en terme de débit.

(b) **Augmentation des délais aller-retour (*RTT*)**
Dans un réseau asymétrique, la probabilité qu'un acquittement rencontre une file d'attente non vide est importante. Une conséquence immédiate pour cet acquittement est une attente au niveau de cette file d'attente. Le temps pour arriver à l'émetteur est donc plus grand: le délai aller-retour de la connexion s'en trouve augmenté. Bien sûr un

plus grand délai aller-retour n'est pas grave pour des applications telles que le courrier électronique, mais pour une application de vidéo conférence, un délai aller-retour plus important n'est pas du tout souhaitable.

(c) **Trafic TCP en rafale**
Un émetteur TCP transmet les données en larges rafales de paquets limitées seulement par la fenêtre de congestion disponible. Si l'émetteur reçoit moins d'acquittements à cause de la congestion sur le chemin de retour, c'est à dire s'il reçoit un acquittement pour k paquets de données, il transmet les paquets de données en rafales de k ou plus. Les routeurs actuels étant incapables de bien gérer de grandes rafales de paquets, nous auront une augmentation du risque de perte pour les paquets de données sur le chemin aller, surtout si k est grand.

(d) **Impossibilité pour TCP de récupérer rapidement des pertes**
"Fast Retransmission" et "Fast Recovery" sont des mécanismes de récupération de pertes pour TCP comme nous l'avons décrit dans la section 2.1.1. Ces mécanismes se basent sur le flux des acquittements pour bien fonctionner. Dans le cas de congestion sur le lien retour, nous devons considérer le risque que l'émetteur ne reçoive pas le nombre d'acquittements dupliqués pourtant envoyés par le récepteur, et nécessaires à la mise en marche d'un "fast recovery" ou d'un "fast rentransmission". L'émetteur doit alors toujours attendre l'expiration du *timer* et ne pourra donc pas récupérer rapidement des pertes.

(e) **Augmentation de l'inéquité de TCP**
En présence d'asymétrie, lorsque plusieurs connexions partagent le chemin de retour, le partage de la bande passante n'est pas équitable même pour des connexions ayant le même délai aller-retour. En effet, dans TCP régit le mécanisme d'équité proportionnelle (*proportional fairness*). Les connexions ayant les délais aller-retour les plus longs ont une plus priorité plus basse vis a vis de l'allocation de la bande passante comparées aux connexions à faibles délais aller-retour [86]. Par conséquent, des connexions ayant théoriquement le même délai aller-retour devraient avoir une allocation équitable de la bande passante. Or, dans le cas de congestion sur le chemin des acquittements, subissent des retards dans leur flux d'acquittements et donc voient leur délais aller-retour augmenter. Elles se verront alors attribuer des part de bande passante inférieures aux autres connexions.

(f) **Problème de Lock-out**
Ce problème touche particulièrement une nouvelle connexion voulant partager le chemin de retour avec d'autres connexions déjà en place. A cause de la saturation de la file d'attente, cette nouvelle connexion subit la perte de son premier acquittement ce qui l'empêche d'incrémenter sa fenêtre de congestion. Ce "*deadlock*" continue jusqu'à ce que les connexions dominantes réduisent leurs débits.

La gravité de ces conséquences varie avec le type de connexions impliquées. A titre

d'exemple pour des connexions à longue durée, la lenteur d'incrémentation des fenêtres de congestion cause la dégradation des performances. Pour des connexions temps réel, augmenter le délai aller-retour et avoir un trafic en rafale serait très mauvais.
C'est pourquoi les solutions varient selon le scénario et les connexions mises en jeu. Dans la section 2.3, nous rapportons les solutions publiées ces dernières années et qui tentent de remédier au problème de *Ack Compression*.

2.2.2 Le trafic bidirectionnel

Avec l'incessante évolution des machines et le développement d'applications de plus en plus accessibles aux particuliers, il est plus que réaliste d'imaginer le cas d'un utilisateur lançant plusieurs applications simultanément (tel que téléchargeant une page Web tout en envoyant un mail ou un fichier), les applications *peer-to-peer*, etc. De telles situations engendrent des trafics TCP bidirectionnels. Dans de tels environnements, il a été démontré dans [45, **?**] que TCP subit des baisses importantes de débits dans un sens du trafic ou même dans les deux. Les deux trafics partagent liens et ressources du réseau. Le problème est que les deux trafics sont composés de paquets complètement différents qui se partagent ces ressources, à savoir paquets de données et paquets d'acquittements. Leur différence est expliquée dans ce qui suit:

- D'une part, les acquittements sont des petits paquets, non encombrants. Cependant ils ne sont pas *"TCP-Friendly"*, en d'autres termes ils ne répondent pas à des éventuelles notifications de congestion et ne réduisent pas leurs débits en conséquence.

- D'autre part les paquets de données réagissent eux à des notifications de congestion dans le réseau en réduisant leur débit. Le problème est leur taille volumineuse qui peut rapidement monopoliser des files d'attente des routeurs.

De plus, dans le cas d'un trafic bidirectionnel, ces paquets de données et d'acquittements dépendent les uns des autres. En effet un retard ou perte d'acquittements dans un sens, notons sens 1, engendre des rafales de paquets de données dans l'autre sens, sens 2. Ce qui engendre une perte ou retard de paquets d'acquittements dans le sens 2, ce qui engendre des rafales de paquets de données dans le sens 1. Ceci mène donc à des dégradation de performances pour le protocole TCP. Et cette situation est évidemment plus grave dans le cas où asymétrie de bande passante est déjà existante au niveau de la bande passante des liens.

Plusieurs approches sont possibles pour remédier à cette "cohabitation" de paquets: (i) donner la priorité à l'un ou l'autre des trafics ce qui implique une famine de l'autre trafic, (ii) utiliser une allocation de bande passante par flux, (iii) limiter le taux des paquets de données dans le réseau, (iv) effectuer aussi un contrôle de congestion pour les acquittements, etc. Nous décrivons plus en détail dans la section 2.3, les solutions visant à remédier au problème de trafic bidirectionnel.

2.3 Solutions proposées pour l'asymétrie de bande passante et le trafic bidirectionnel

Les problèmes engendrés par une asymétrie dans les liens aller et retour d'un réseau ont interessé la communauté de l'Internet et quelques solutions ont été proposées pour y remédier [78]. Nous allons dans cette section les classer en deux catégories:

- Solutions qui impliquent la participation des extrémités d'une connexion (l'émetteur et le récepteur). Pour mettre en place ces solutions, des changements dans TCP sont souvent nécessaires.

- Solutions qui impliquent les routeurs du réseaux. Ces solutions sont aussi dites "transparentes" puisqu'elles ne nécessitent pas la participation de l'utilisateur.

2.3.1 Agir sur les extrémités

Dans cette section, nous faisons une description fidèle des solutions proposées afin de remédier au problème de compression d'acquittement en intégrant les mécanismes appropriés au niveau des extrémités. Nous commençons par une solution qui agit sur les deux extrémités en même temps, à savoir la compression des entêtes des paquets d'acquittements. Ensuite, nous énumérons les solutions qui agissent sur la source, nous parlerons en particulier de l'augmentation de la taille des segments, de l'espacement des paquets de données et du *"Byte Counting"*. Enfin, nous abordons les solutions qui agissent sur le récepteur, à savoir la modification de la génération des paquets d'acquittements, l'estimation de la fenêtre de congestion et le contrôle de congestion pour les paquets d'acquittements.

2.3.1.1 Agir sur les deux extrémités

Compresser les entêtes des paquets d'acquittements Cette solution [36] nécessite des changements dans les deux extrémités d'un lien montant. En effet, la taille des acquittements est réduite en compressant les entêtes de ces paquets, ce qui permet d'avoir plus d'espace sur le lien et de diminuer le niveau de la congestion.
Il est recommandé d'utiliser cette solution seulement dans des réseaux où le taux d'asymétrie n'est pas très important et où le taux d'erreur est très faible. Mais dans le cas général, la complexité de cette solution est trop élevée par rapport aux éventuelles améliorations qu'elle peut apporter, c'est pourquoi nous ne recommandons pas de l'utiliser.

2.3.1.2 Agir sur l'émetteur

Voici une description de trois solutions qui nécessite des modifications au niveau de l'émetteur d'une connexion passant par un lien asymétrique. Toutes ces solutions proposent de réduire les effets du problème de compression d'acquittement.

Augmenter les tailles maximales des segments TCP Une solution immédiate serait pour l'émetteur d'augmenter la taille des segments TCP (MSS *Maximum Segement Size*) [9]. En effet ceci réduirait le nombre d'acquittements sur le chemin de retour et éviterait ainsi une congestion sur le chemin des acquittements. Une implémentation de cette solution est possible si l'émetteur n'utilise pas le mécanisme *Path MTU (Maximum Transmission Unit) Discovery* [54] et les routeurs appliquent une fragmentation IP sur le chemin des données [65]. Ceci va en effet permettre d'avoir un MSS plus grand.

Seulement, augmenter les tailles maximales des segments TCP n'est pas la meilleure solution. En effet, tout d'abord on aura augmenté la différence entre l'unité de correction d'erreur *error recovery* qu'est le segment TCP et entre l'unité de transmission qu'est le paquet IP. Ceci implique une transmission d'un grand nombre de paquets après la perte d'un seul paquet IP, et une aggravation de la congestion du réseau.
Ensuite, à chaque lancement d'une connexion ou après l'expiration d'un *timer*, l'émetteur transmet un plus grand nombre de paquets IP, à savoir une rafale de paquets IP pour chaque segment TCP. La phase de *slow start* est par conséquent plus agressive et la connexion causera une dégradation de performances des autres connexions et une mauvaise utilisation du réseau.
Enfin, en augmentant les tailles des segments TCP, on diminue le nombre de paquets d'acquittements générés. En cas de congestion, la probabilité d'avoir des paquets d'acquittements dupliqués afin de notifier d'une congestion va automatiquement diminuer. Les *timeouts* sont alors les principaux indicateurs pour l'émetteur d'un cas de congestion dans le réseau. Cette solution entraîne donc une réduction de l'efficacité des algorithmes de *fast retransmission* et de *fast recovery* qui se déclenchent tardivement. De plus, l'incrémentation de la fenêtre de congestion de l'émetteur restera toujours lente, étant donné que le nombre de paquets d'acquittements arrivant à l'émetteur diminue. Toutes ces raisons justifient le fait que la solution de choisir un plus grand segment TCP n'est pas recommandée par le groupe de travail PILC [69].

Espacer l'envoi des paquets de données Cette solution consiste à contrôler la cadence d'envoi des paquets par la source (*TCP Sender Pacing*)[4]. Ce contrôle vient s'ajouter au contrôle de congestion de TCP et vise à espacer les paquets de données envoyés afin de diminuer le nombre d'acquittements sur le chemin de retour. Cette solution effectue un contrôle de débit (*rate control*) puisque TCP ne se base plus uniquement sur le *Ack Clocking* mais ajoute du contrôle de débit. Le nombre de paquets envoyés les uns à la suite des autres est limité à un seuil maximum autre que celui de la fenêtre de congestion. Ce seuil est calculé par l'émetteur en fonction de la taille de la fenêtre de congestion et de la moyenne du délai aller retour (*SRTT*).

Contrôler le débit favorise la diminution de transmission de grosses rafales de paquets et aide à espacer par la même occasion la réception des paquets d'acquittements. Cette solution est rejetée par la communauté. En effet, elle nécessite des changements importants dans les implémentations des sources TCP et implique des coûts de traitement supplémentaires à chaque transmission de paquet de données [4]. L'algorithme optimal de calcul du meilleur débit d'envoi des paquets TCP n'a toujours pas été spécifié vu la complexité qu'il représente en terme de compromis entre performances du réseau

et coût de traitement.

Changer le mode d'incrémentation des fenêtres de congestion Il s'agit ici de considérer pour l'incrémentation de la fenêtre de congestion TCP le nombre total d'octets acquittés par un paquet d'acquittement plutôt que le nombre d'acquittements reçus. Cette solution est connue sous le nom de *TCP Byte Counting* [1, 2]). En d'autres termes, si un émetteur TCP reçoit 1 paquet d'acquittement, il regarde le nombre total de segments TCP réellement acquittés. Si l'acquittement acquitte d paquets de données, l'émetteur incrémente sa fenêtre de congestion d et non de 1 comme dans le cas général (il augmente de d/2 si le mécanismes *delayed ACK* est utilisé). On aura ainsi résolu le problème de lenteur de l'incrémentation de la fenêtre de congestion causée par le problème de *Ack Compression*.

De toutes les solutions qui agissent sur les sources TCP, celle ci est la solution la plus intéressante et celle qui nécessite le moins de modification chez l'émetteur TCP. En effet, l'incémentation de la fenêtre de congestion d'un émetteur TCP est relative à la capacité disponible sur le chemin des données, le nombre d'acquittements reçus traduit moins cette capacité que le nombre total d'octets effectivement acquittés.
La solution de *TCP Bytes Counting* doit en revanche être utilisée avec un autre mécanisme pour diminuer le risque de congestion sur le chemin d'acquittements. En effet, une augmentation plus rapide de la fenêtre de congestion engendre indirectement plus de paquets d'acquittements sur le chemin retour, le risque de congestion accroît alors à moins d'utiliser un *TCP Sender Spacing* par exemple (c.f. paragraphe précédent) ou autre technique de contrôle de congestion des paquets d'acquittements. D'autre part, il est à noter que le *TCP Byte Counting* provoque des rafales importantes de paquets de données qui peuvent être cause d'un déséquencement de paquets IP. Un mécanisme pour remédier aux rafales doit aussi lui être associé.

2.3.1.3 Agir sur le récepteur

Concernant les solutions qui agissent sur les récepteurs des connexions TCP passant par un lien asymétrique, nous allons dans cette section décrire les trois plus prometteuses.

Modifier la génération des paquets d'acquittements Les acquittements retournés à l'émetteur peuvent être envoyés à chaque réception d'un paquets de données ou alors plus généralement, à chaque réception de "d" paquets de données. Un tel mécanisme d'envoi tardif d'acquittements est sans aucun risque pour la connexion puisque les acquittements sont cumulatifs. Cette solution est appelée *"Delayed Acks"* (c.f. 2.1.1.1) et consiste à attendre de recevoir d paquets de données de la connexion avant d'envoyer un acquittement. Ceci réduit le nombre de paquets d'acquittement sur le chemin de retour et donc de libérer de l'espace sur le chemin retour et de réduire les effets de l'asymétrie. La solution *Modified Delayed ACK* est une variante de *Delayed ACK* tel spécifiée dans TCP. La seule différence est de permettre dans le cas de *Modified Delayed ACK* d'avoir "d" > 2, c'est à dire de permettre d'attendre la réception de plus de deux paquets de

données avant d'envoyer un paquets d'acquittements. En effet, comme cité dans 2.1.1.1, il est important dans le cas de *Delayed ACK* de ne pas trop retarder les acquittements pour éviter les *timeout*, "d" devrait être au maximum égal à 2 [63, 3]. Afin de pouvoir diminuer le débit d'envoi des paquets d'acquittements et réduire la congestion sur le chemin retour, la solution de *Modified Delayed ACK* permet d'avoir "d" > 2.

Cette solution n'est pas intéressante parce qu'elle présente plusieurs inconvénients: des délais aller-retour plus importants, une lenteur dans l'incrémentation des fenêtres de congestion, trafic en rafale, etc. Elle ne résout donc pas les problèmes de *Ack Compression*. De plus, le récepteur doit être conscient qu'il se trouve à l'extremité d'un lien asymétrique et effectuer la modification au niveau du nombre de paquets qu'il doit recevoir avant l'envoi de paquet d'acquittement (augmenter la valeur "d"). En effet, une telle modification à TCP tout entier peut amener des performances dégradées du réseau. Le déploiement de la solution *Modified Delayed ACK* est par conséquent très improbable.

Estimer la fenêtre de congestions de l'émetteur avant d'acquitter Cette solution vient améliorer la solution précédente (*Modified Delayed ACK*) en proposant un moyen de fixer correctement la valeur "d" relative au nombre de paquets de données que le récepteur doit attendre avant l'envoi d'un paquet d'acquittement. En effet, le principe de cette solution est de varier le nombre de paquets qu'un acquittement acquitte en fonction de la valeur estimée de la fenêtre de congestion de l'émetteur. Deux propositions se basant sur ce principe ont été publiées [16, 65]. Avant d'envoyer les acquittements, le récepteur tente de mesurer la fenêtre de congestion à la source. Pour cela, dans [16], le récepteur reconstruit le comportement de la fenêtre de congestion de l'émetteur en manipulant une variable qui imite le seuil de la phase *slow start*. Dans [65], l'émetteur estime le délai aller retour (RTT) puis il compte le nombre de paquets reçus pendant un RTT. Dans les deux version de cette solution, l'envoi des acquittements se fait en fonction de la fenêtre de congestion de la source. Si la fenêtre de congestion est petite, le nombre de paquets concernés par un acquittement est petit. Ceci implique un plus grand nombre de paquets d'acquittements et permettra ainsi d'accélérer la construction de la fenêtre de congestion au niveau de l'émetteur. Si la fenêtre de congestion de l'émetteur est grande, le nombre de paquets de données acquittés par un paquet d'acquittement est plus grand, ce qui engendre moins de paquets d'acquittements générés et donc libère l'espace sur le lien retour.

Cette solution est encore au stade expérimental, les différents algorithmes proposés pour l'estimation de la fenêtre de congestion de l'émetteur sont encore au stade de recherche. Cependant, nous pouvons noter la complexité de ces algorithmes, notamment dans l'estimation des délais aller-retour et dans l'imitation de l'évolution de la fenêtre de congestion. Le traitement supplémentaire que cette solution nécessite au niveau du récepteur peut causer des délais et des coût plus ou moins importants selon le type d'application. Prouver leur utilité n'est pas chose facile.

Contrôle de congestion pour les acquittements Cette solution propose d'effectuer un contrôle de congestion même sur les paquets d'acquittements. Ces paquets sont

considérés utilisateurs de ressources du réseau autant que les paquets de données. Il faut donc les rendre aussi sensibles à des notifications de congestion dans le réseau, auquel cas ils seront amenés à réduire leur débit. Des paquets adoptant un tel comportement sont dits *"TCP-friendly"*. Cette solution nécessite l'aide des routeurs du réseau qui doivent faire parvenir au récepteur l'information qu'il y a congestion sur le lien retour. Afin d'effectuer cette notification, les concepteurs de cette solutions recommandent l'utilisation de RED-ECN (*Random Early Detection with Explicit Congestion Notification*, c.f. 2.1.3) dans les routeurs. Lorsqu'un récepteur reçoit un paquet de données marqué traduisant une notification de congestion sur le chemin aller, il augmente exponentiellement le nombre de paquets de données qu'il doit recevoir avant de pouvoir envoyer un paquet d'acquittement (il augmente le facteur *delayed-Ack* "d"). Le récepteur permet ainsi de réduire exponentiellement le nombre d'acquittements sur le lien retour. Si aucune indication de congestion n'est notée, le récepteur réduit linéairement la valeur de "d" augmentant ainsi le nombre de paquets acquittement sur le lien retour. La valeur maximale de "d" est fixée par la taille de la fenêtre de l'émetteur. Le récepteur applique donc aux paquets d'acquittements le contrôle de congestion qu'applique TCP aux paquets de données.

Cette solution nécessite de grands changements aussi bien dans les récepteurs TCP que dans le réseau lui même. De plus elle nécessite l'aide d'un algorithme de gestion active de file d'attente dans les routeurs (AQM), et du fait que ces algorithmes n'ont pas encore prouvé leur efficacité, il est difficile de prouver celle de cette solution. Les coûts de traitement qu'elle implique sont aussi non négligeables. Pour toutes ces raisons, nous rejetons cette solution.

2.3.2 Agir sur les routeurs

Dans les trois sous-sections précédentes, nous avons cité quelques unes des solutions qui proposent de réduire les effets du problème de *Ack Compression* en agissant directement sur les émetteurs et les récepteurs TCP des connexions passant par des liens asymétriques. Toutes ces solutions nécessitent beaucoup de changements dans les implémentations de TCP.

Les solutions qu'on énumère dans cette section sont dites "transparentes" puisqu'elles agissent sur les routeurs su réseau. Elles ne modifient donc ni les émetteurs ni les récepteurs TCP, ce qui leur donne l'avantage d'être plus facilement déloyables dans le réseau. et sont les plus recommandées par la communauté pour remédier aux problèmes de l'asymétrie de bande passante. En effet, de célèbres sociétés informatiques se sont "appropriées" aujourd'hui les logiciels et programmes d'accès aux services de l'internet. Ces sociétés adoptent la technologie TCP/IP, et implémenter un nouveau mécanisme relatif à un des aspects de cette technologie semble donc très difficile. Les solutions transparentes quant à elles sont adressées aux opérateurs auxquels les routeurs appartiennent. Ces opérateurs sont toujours soucieux de maximiser les performances de leurs réseaux et sont donc toujours intéressés par de nouveaux mécanismes.

2.3.2.1 Filtrer les paquets acquittement (AF)

Les acquittements sont cumulatifs, ce qui veut dire que le dernier acquittement d'une connexion porte toute l'information utile nécessaire à cette connexion et peut ainsi remplacer tous les acquittements antérieurs. La solution de filtrer les acquittement se base sur cette caractéristique des acquittements.
A l'entrée de la file d'attente du lien congestionné, un agent effectue un filtrage des acquittements:

> Si un acquittement A_i arrive et qu'il appartient à la même connexion qu'un autre acquittement A_{i-1} déjà présent dans la file d'attente, A_{i-1} sera supprimé et A_i rajouté à la fin de la file.

Cette opération est appelée "filtrage", elle résulte en une libération de l'espace dans la file. Le risque de congestion est par conséquent diminué.

FIG. 2.3 – *Filtrage d'un acquittements*

Une schématisation de cette solution est représentée dans la figure **??**.
Une mise en garde cependant pour le filtrage d'acquittements est de ne pas filtrer les acquittements spéciaux de TCP tels que les acquittements de signalisation ou les acquittements selectifs (SACK). Supprimer ces acquittements entraînerait de graves disfonctionnements du protocole TCP.
Deux versions du mécanisme de filtrage d'acquittements existent:

1. Commencer à "filtrer" dès la réception d'un deuxième acquittement d'une même connexion. C'est la version la plus simple qui garantit que le nombre d'acquittements dans la file d'attente soit maintenu à une petite valeur, ce qui diminue la congestion sur le chemin retour sans perdre la moindre information. Cette version

du mécanisme de filtrage des paquets d'acquittements présente tout de même un problème: elle réduit le nombre d'acquittements au détriment du nombre d'acquittements passés aux sources TCP. En effet un mécanisme de filtrage qui permet à une connexion TCP d'avoir un seul acquittement dans la file d'attente pourrait résulter en un rejet d'acquittements même lorsque le débit moyen d'acquittements est inférieur à la bande passante disponible sur le chemin de retour. Il suffit pour cela que les acquittements arrivent en rafales avec un débit moyen inférieur à la bande passante, comme c'est le cas pendant la phase de slow start ou en cas de regroupement de paquets de données.

2. La deuxième version du filtrage d'acquittements, plus performante mais plus compliquée, est de fixer un seuil à partir duquel nous commençons le filtrage [7]. Dans ce cas, les rafales d'acquittements peuvent être absorbées sans aucun impact négatif sur les performances, ce qui assure une accélération de l'augmentation de la fenêtre au niveau de la source. Là aussi les auteurs définissent deux schémas pour le filtrage des acquittements. Le premier est le schéma statique ou déterministe et consiste à commencer le filtrage dès que le nombre d'acquittements dans la file d'attente atteint la valeur du seuil fixé à l'avance. Le deuxième schéma est le schéma dynamique ou adaptatif et consiste à mesurer l'utilisation de la bande passante sur le chemin de retour et à utiliser cette mesure comme information sur le début et l'arrêt du filtrage. Les acquittements sont filtrés uniquement lorsque le chemin de retour est bien chargé.

Le filtrage d'acquittements est donc la solution proposée dans la littérature à la congestion sur le chemin de retour la plus intéressante. Cependant, de part son principe même, cette solution réduit le nombre de paquets d'acquittement qui arrivent à l'émetteur et, par conséquent, provoque un ralentissement de l'incrémentation de la fenêtre de congestion au niveau de la source. On a donc réduit la congestion au niveau du chemin de retour mais le problème de l'incrémentation de la fenêtre de congestion est toujours présent, voire même exacerbé. C'est pourquoi le filtrage d'acquittements tout seul ne suffit pas. Un autre mécanisme de gestion des acquittements est nécessaire pour rendre cette solution applicable en pratique. Ce mécanisme est appelé reconstruction d'acquittements et corrige le problème de la rareté des acquittements qui arrivent à l'émetteur après le passage par un mécanisme de filtrage d'acquittements.

2.3.2.2 Reconstruction de paquets d'acquittements (AR)

Le filtrage des acquittements est une solution pour remédier au problème de congestion des acquittements sur le chemin de retour. Seulement, cette solution nécessite un second élément actif qui atténue l'effet du débit réduit des acquittements après leur filtrage et l'effet des acquittements en rafale. L'élément actif en question est appelé reconstructeur d'acquittements et agit à la sortie du lien de retour congestionné. Il aura pour rôle de remplir les "trous" d'acquittements pour chaque connexion; trous formés après le filtrage de quelques acquittements de cette connexion. Les acquittements supplémentaires reconstruits permettront à l'émetteur d'incrémenter sa fenêtre

de congestion comme si rien ne s'était passé. De plus, en reconstruisant les paquets d'acquittements, nous aurons restauré le comportement *"self-clocked"* du protocole TCP et diminué considérablement les rafales envoyées par les sources TCP.

FIG. 2.4 – *Reconstruction du flux d'acquittements*

La reconstruction des acquittements est représentée dans la figure 2.4. Elle se fait en se basant sur le flux d'acquittements reçu. Le reconstructeur évalue le nombre de paquets d'acquittements qui ont été filtrés en regardant dans les numéros de séquence des acquittements qui lui parviennent [2]. Par exemple, si deux acquittements successifs arrivent séparés par x, le reconstructeur doit générer au maximum:

$$\frac{x}{seuil} - 2$$

Où seuil est une constante qui limite le nombre d'acquittements à générer généralement égale au double de la taille maximale des paquets (le double correspond à la politique de retardement des acquittements de facteur d=2).
Le reconstructeur tient aussi en compte du débit d'arrivée des acquittements pour regénérer le flux selon un débit adéquat c'est à dire en respectant un espacement entre les acquittements générés. Le reconstructeur mesure le débit d'arrivée en utilisant un estimateur mobile exponentiel. Ce débit dépend du débit de sortie de la part du chemin de retour et de la présence d'autres trafics sur le lien. L'estimateur fournit donc l'espacement temporel moyen pour les acquittements. En fait, si on effectue les notations

2. TCP est un protocole "*byte stream*" et il n'exite pas actuellement de moyen de pouvoir avoir une historique des paquets d'acquittement qui passent par un routeur. Il est difficile par conséquent de concevoir une implémentation réelle de AR qui puisse regarder le numéro de séquence des acquittements. Cependant, AR a été étudié dans un cas de simulateur de réseau où accéder à cette information est possible. C'est pourquoi nous en parlons, dans l'attente d'un moyen de l'implémenter réellement

suivantes:
- δ_a: débit d'arrivée des acquittements au reconstructeur
- δ_t: espacement temporel moyen selon lequel les acquittements arrivent au reconstructeur
- espaceAcquittement: l'espacement de débit entre les acquittements générés
- intervalAcquittement: l'espacement de temps entre les acquittements générés
Le reconstructeur obeit à l'équation suivante pour déterminer le débit de reconstruction des acquittemnets:

$$\frac{\delta_a}{\delta_t} = \frac{espaceAcquittement}{intervalAcquittement}$$

Le mécanisme de reconstruction des acquittements est en effet assez complexe à mettre en oeuvre mais, en collaboration avec le filtrage des acquittements, il permet de préserver la sémantique bout-en-bout de TCP et donc de considérablemet améliorer les performances de connexions passant par des chemins asymétriques. Cependant, le reconstructeur risque de générer un trafic d'acquittements érronés si les conditions suivantes ne sont pas respectées :

– les acquittements doivent parvenir au reconstructeur dans l'ordre,

– tous les acquittements doivent passer par le reconstructeur,

– les acquittements ne portent pas des informations supplémentaires.

Le mécanisme de reconstruction des acquittements obéit à un compromis entre amélioration des performances des sources TCP en terme de débit atteint et augmentation du délai aller-retour dûe à la manipulation des aquittements sur le chemin de retour.

2.3.3 Solutions pour le trafic bidirectionnel

Un trafic bidirectionnel amplifie l'effet de l'asymétrie dans le réseau [9, 6].

FIG. 2.5 – *Architecture d'un trafic bidirectionnel*

En effet, dans ce cas, le lien de retour est non seulement moins rapide que le lien aller mais en plus il est partagé par deux types de paquets complètement différents de

par leur rôle et leur réactivité dans le protocole TCP, à savoir les paquets de données et les paquets d'acquittements. La figure 4.1 illustre une situation de trafic bidirectionnel sur un lien asymétrique.
Des solutions ont été proposées pour rendre la cohabitation de ces deux types de paquets la plus harmonieuse possible.

Limiter le nombre de paquets de données *Backpressure* Cette proposition [45] consiste à limiter le nombre de paquets de données autorisés à être stockés dans la file d'attente à l'entrée du lien retour. La file d'attente informe alors l'émetteur du nombre de paquets de données qu'elle stocke. Celui ci limite son envoi de paquets de données à un seuil au delà duquel les paquets de données ne sont plus acceptés dans la file d'attente. Cette solution consiste donc en fait à limiter les paquets de données dans le réseau à un certain seuil, même si les fenêtres de congestion des émetteurs TCP relatifs à ces paquets permettent encore l'envoi de paquets. Elle applique donc le mécanisme de *rate control* à la connexion dont les paquets de données circulent sur le chemin retour. Le seuil à partir duquel plus aucun paquet de données n'est autorisé à rentrer dans le réseau est déterminé à partir des débits des différentes connexions (c.f. [45].

Cette proposition a pour objectif en fait d'éviter aux paquets d'acquittement d'être stockés après les paquets de données dans la file d'attente du routeur à l'entrée du lien retour. Les paquets de données sont généralement volumineux, elle évite ainsi une famine des paquets d'acquittemenst. Cependant, les paquets d'acquittements ayant donc une priorité plus élevés peuvent à leur tour causer la famine des paquets de données. Cette solution est par conséquent inéquitable vis à vis des trafics ascendants.

Donner la priorité aux paquets d'acquittement: *AckFirst Scheduling* Il s'agit d'une autre proposition qui donne toujours une priorité plus élevée aux paquets d'acquittements dans les files d'attente des routeurs par rapport aux paquets de données [9]. Plus clairement, paquets de données et paquets d'acquittements partagent une seule et même file d'attente de type FIFO à l'entrée du routeur du lien retour. Seulement, les paquets d'acquittements sont toujours servis en priorité afin d'éviter aux paquets d'acquittement une éventuelle longue attente dans les routeurs derrière les paquets de données généralement volumineux (c.f. chapitre1).

Assurer une priorité plus élevée aux paquets d'acquittements peut être obtenu en appliquant un ordonnancement de *Priority Queuing* dans le routeur. Sous l'ordonnancement "*Priority Queueing*" [81], les paquets arrivant au lien de sortie sont classés en une, deux ou plusieurs classes de priorités. Une classe de priorité de paquets dépend d'un marquage explicite présent dans l'entête du paquets (par exemple le bit ToS "Type of Service" dans les paquets IPv4). La priorité peut aussi être spécifiée par l'adresse source du paquet, son adresse destination ou d'autres critères. Chaque classe de priorité possède sa propre file d'attente. Lors du choix du paquet à transmettre, la politique de l'ordonnancement "*Priority Queuing*" est de transmettre un paquet de la classe de priorité la plus haute vers la plus basse. Pour plusieurs paquets de même priorité, le choix se fait par une politique FIFO.

Cette proposition, à force de donner l'avantage aux paquets d'acquittements, risque fort d'empêcher complètement les paquets de données de transiter, surtout dans un cas de connexions multiples où le trafic d'acquittements est de longue durée. Les simulations que nous avons menées et que nous présentons dans 2.4 montrent la famine dont souffrent les paquets de données dans le cas où un mécanisme *AckFirst Scheduling* est utilisé à l'entrée d'un lien retour bas débit. Il a été proposé dans cette solution de compacter les entêtes des paquets d'acquittements afin d'éviter une telle famine pour les paquets de données. Elle demeure cependant inéquitable. Une autre solution serait de l'associer à un mécanisme qui réduit le débit des paquets d'acquittements. Les concepteurs de cette solution recommandent l'utilisation d'un contrôle de congestion pour les acquittements, qui permettra alors de contôler le débit des acquittements et de protéger les paquets de données. La complexité d'une telle association de mécanismes nous oblige à ne pas la recommander.

Partage équitable du lien de retour: *Per-Flow queuing* Cette proposition consiste à utiliser des files d'attente séparées au niveau du routeur à l'entrée du lien retour bas débit, une file d'attente pour chaque flux. Les paquets entrants sont donc placés dans la file correspondante et seront ordonnancés selon un algorithme équitable tel que le *Weighted Round Robin* (WRR) ou par tout autre algorithme modifié selon les besoins des opérateurs. Un ordonnanceur modifié est présenté dans [45] et propose de limiter le nombre de paquets d'acquittements qu'une extrémité est autorisée à envoyer avant d'envoyer un paquet de données. Ceci permet d'utiliser au mieux les deux bandes passantes, celle du lien aller et celle du lien retour. Nous proposons d'ailleurs dans le chapitre 3 un mécanisme qui s'inspire de cet algorithme. CBQ (*Class Based Queuing*) est un exemple de *Per-Flow Queuing*, il est explicité plus loin dans cette thèse (c.f. 3.1.2).

C'est la solution la plus propre et la plus équitable pour les deux types de paquets. Elle n'a aucun impact sur les algorithmes de contrôle de congestion de TCP et peut être déployée facilement dans l'Internet. Cependant, les parts de bande passante allouée aux differentes classes sont constantes et ne s'adaptent pas aux trafics, ce qui peut impliquer une mauvaise utilisation du lien et une dégradation de performances de TCP.

Nous avons donc énuméré quelques unes des solutions qui visent à réduire les problèmes rencontrés par des trafics bidirectionnels passant par des liens asymétriques, ces solutions sont résumées dans la figure 2.6. La solution de *Per-Flow Queuing* est celle qui cause le moins d'inconvénients puisque c'est celle qui demande le moins de traitements particuliers. Nous nous inspirons d'ailleurs de cette technique dans notre mécanisme d'ordonnancement présenté dans le chapitre 3.

Dans la section suivante, nous analysons plus en détail les propositions de filtrage et reconstruction des acquittements et la proposition de donner la priorité la plus élevée aux acquittements. En effet, à priori, la combinaison filtrage et reconstruction des acquittements nous semble la plus intéressante dans le cas de passage par des liens asymétriques. Et pour un trafic bidirectionnel, il suffirait d'appliquer le mécanisme de

FIG. 2.6 – *Solutions pour l'asymétrie et le trafic bidirectionnel*

AckFirst Scheduling afin d'éviter aux acquittements d'être longuement retardés.

2.4 Etude d'un cas de trafic bidirectionnel passant par un lien asymétique

Le protocole TCP a prouvé son efficacité et sa robustesse pour assurer un trafic fiable dans un cas "générique". Cependant, TCP présente des difficultés dans un environnement asymétrique, limites dûes au fait que TCP se base sur le flux des acquittements pour réguler son trafic et récupérer des pertes. Dans les deux sections précédentes, nous avons d'abord expliqué l'impact de l'asymétrie de bande passante sur le bon fonctionnement de TCP puis nous avons rapporté les solutions les plus importantes et les plus prometteuses citées dans la littérature. Notre intérêt était plus particulièrement porté sur deux solutions dites les plus prometteuses: acquittement Filtering (AF) et acquittement Reconstruction (AR) pour remédier au probléme de compression d'acquittements. Concernant le trafic bidirectionnel, c'est la solution de privilégier les paquets d'acquittements qui est la plus recommandée.

Dans cette section, nous étudions par des simulations le cas d'un trafic bidirectionnel passant par un lien asymétrique. Des simulations nous aideront à avoir une idée plus concrète des effets de l'asymétrie sur un trafic bidirectionnel. Nous étudierons en fait l'impact de la combinaison de solutions suivante:

"AF+AR+AckFirst Scheduling".

Cette combinaison de solution est très recommandée dans la littérature et plus parti-

culièrement dans la RFC du groupe PILC [9].
Dans le reste de cette étude nous utiliserons les nomenclatures suivantes:
- Lien descendant: Le lien du chemin aller.
- Lien ascendant: Le lien du chemin de retour.
- Trafic descendant: Le trafic qui a ses paquets de données sur le chemin aller.
- Trafic ascendant: Le trafic qui a ses paquets de données sur le chemin de retour.

2.4.1 La topologie des simulations

FIG. 2.7 – *Topologies des simulations*

Nous avons effectué des simulations en utilisant ns-2 comme simulateur de réseaux [62, 24]. La topologie est représentée dans la figure 2.7. N connexions TCP à longue durée commencent simultanément de chaque coté d'un lien asymétique. Nous prenons N égal à 10. Nous avons fixé les capacités des différents liens de façon à toujours avoir le lien asymétrique comme étant le seul lien congestionné de notre topologie (le lien de l'utilisateur vers le réseau). Toutes les connexions sont de type FTP infinies[3] utilisant le protocole TCP dans sa version Reno comme protocole de transport, elles commencent toutes à émettre aléatoirement entre la première et la cinquième seconde du début de la simulation. Les paquets de données sont de tailles 1000 octets, les paquets d'aquittements sont de taille 40 octets. Tous les routeurs sont munis de files d'attente de type FIFO dont les tampons peuvent contenir jusqu'à 20 paquets. Les simulations durent 500 ms.

Nous rappelons notre objectif suite à ces simulations: tester l'efficacité des mécanismes de filtrage et de reconstruction des acquittements dans un environnement de trafic bidirectionnel passant par un lien asymétrique. Nous voulons aussi voir si donner la priorité aux paquets d'acquittements est favorable à l'optimisation des performances dans un tel environnement. Les mécanismes de filtrage et de reconstruction des paquets sont

3. La majorité du trafic dans l'Internet est certes du trafic Web à courte durée. Par souci de simplification, nos simulations présentent des connexions FTP infinies illustrant du téléchargement de gros fichier. Les résultats de ces simulations sont négatifs pour de telles connexions FTP, ils le seront à fortiori aussi, sinon pires, pour des connexions Web à courte durée.

donc appliqués respectivement à l'entrée et à la sortie du lien retour bas débit. Le *Ack-First scheduling* est appliqué dans la file d'attente à l'entrée du lien.
Nous notons dans nos figure pour le reste de cette section T_0 le trafic ascendant (de l'utilisateur vers le réseau) et T_1 le trafic descendant (du réseau vers l'utilisateur).

2.4.2 Impact de AF/AR et *AckFirst* sur l'évolution des débits

Un utilisateur de trafic bidirectionnel cherche avant tout à avoir un compromis acceptable entre le débit moyen du trafic ascendant et le débit moyen du trafic descendant. Concernant le trafic ascendant, ce sont les paquets de données appartenant à ce trafic qui subissent le passage par un lien à faible capacité: ils seront donc stockés dans des files d'attente, retardés ou même perdus. Pour le trafic descendant, ce sont plutôt les paquets d'acquittements qui subissent des retards ou des pertes. Il est donc intéressant de voir concrètement l'impact de cette asymétrie de bande passante du lien sur les débits moyens des deux trafics.
Nous regardons dans cette section la variation du débit en utilisant:

1. un simple ordonnancement FIFO à l'entrée du lien retour,
2. un ordonnanceur FIFO auquel on associe les mécanismes AF/AR de filtrage et reconstruction des paquets d'acquittements, et
3. un ordonnanceur FIFO auquel on associe les mécanismes AF/AR de filtrage et reconstruction des paquets d'acquittements et en donnant une plus haute priorité aux acquittements par rapport aux paquets de données.

Nos simulations on été répétées plusieurs fois et ont montré une appartenance à un intervalle de confiance de l'ordre de 0.95.

2.4.2.1 Débit moyen du trafic ascendant

Dans la figure 2.8, nous voyons les variations du débit du trafic ascendant dans le cas d'un ordonnancement FIFO, dans le cas de l'utilisation AF/AR et dans le cas de AF/AR avec AckFirst. L'utilisation de AF/AR a été clairement bénéfique à ce trafic puisqu'on voit son débit moyen augmenter considérablemnt et se stabiliser à 6Kbps (notons que ce débit correspond au débit moyen d'une connexion appartenant au trafic ascendant).
En appliquant un filtrage des acquittements appartenant au trafic descendant, le routeur à l'entrée du lien libére de l'espace dans ses tampons. Cette place va être prise par les paquets de données du trafic ascendant qui se trouve donc privilegié aux niveau des routeurs et parvient à monopoliser les files d'attente et à augmenter son débit moyen.

Quant à l'utilisation d'un ordonnancement qui donne la priorité aux paquet d'acquittement, la figure 2.8 montre une réduction importante du débit moyen du trafic ascendant. Ce trafic finit par atteindre à peine un débit moyen de 0.2 kbps, c'est à dire un débit moyen inacceptable pour l'utilisateur. En effet en faisant toujours passer les

FIG. 2.8 – *Variation du débit moyen du trafic ascendant T_0*

paquets d'acquittements qui appartiennent au trafic descendant devant les paquets de données qui appartiennent au trafic ascendant, l'ordonnanceur cause une famine pour le trafic ascendant puisque les connexions du trafic descendant sont permanentes.

2.4.2.2 Débit moyen du trafic descendant

FIG. 2.9 – *Variation du débit moyen du trafic descendant T_1*

Pour le trafic descendant la situation est complètement réciproque. En effet, comme le montre la figure 2.9, l'emploi des mécanismes de filtrage et reconstruction des acquittements AF/AR diminue fortement le débit moyen des connexions du trafic descendant.

Ce trafic subit le filtrage de ces acquittements. Et vu que le lien est aussi partagé par les paquets de données du trafic ascendant, l'espace libéré dans les routeurs sera utilisé par les paquets de données ascendant. Le trafic descendant perd aussi au change et est lourdement affecté. L'ajout de la reconstruction des paquets d'acquittements AR n'arrive pas à compenser cette longue attente derrière les volumineux paquets de données appartenant au trafic ascendant. AF/AR entraîne donc une augmentation du débit du trafic ascendant au détriment du débit des connexions descendantes, comme le montre la figure 2.9.
Si nous appliquons maintenant une priorité aux paquets d'acquittement appartenant au trafic descendant, nous arrivons à protéger les acquittements et à obtenir un meilleur débit moyen pour le trafic descendant.

2.4.2.3 Impact de AF/AR et *AckFirst*

A ce stade de notre étude, nous nous trouvons en présence d'une impasse. En effet l'application des mécanismes de filtrage et de reconstruction d'acquittements AF/AR améliore les performances du trafic ascendant T_0 mais réduit fortement les performances du trafic descendant T_1. Réciproquement, donner la priorité aux paquets d'acquittements augmente les débits du trafic descendant T_1 mais annule pratiquement le débit du trafic ascendant T_0.

Nous pensons qu'en fait la raison principale de cette impasse est la monopolisation de la file d'attente par un type de trafic au dépend de l'autre. En d'autres termes:

- dans le cas où on applique le filtrage et la reconstruction des acquittements, on filtre les paquets d'acquittements dans la file d'attente à l'entrée du lien bas débit, libérant ainsi de l'espace pour les paquets de données. Du fait de la grande taille de ces paquets, ils finissent par monopoliser la file d'attente et les paquets d'acquittements sont donc incapables de transiter via le routeur,

- dans le cas par contre où on donne une priorité aux paquets d'acquittement et du fait que ces paquets sont permanents, ils finissent pas monopoliser la file d'attente et les paquets de données ne transitent plus par le routeur.

Une première solution est donc d'éviter cette monopolisation de la file d'attente. Un mécanisme tel que RED (c.f. 2.1.3.2) a justement pour objectif de toujours garder une taille moyenne de file d'attente réduite, c'est pourquoi nous l'avons appliqué à l'entrée du lien asymétrique, associé toujours aux mécanismes AF, AR et *AckFirst*. Les variations des débits avec une telle configuration sont reportées dans la section suivante.

2.4.3 RED pour avoir un compromis entre les débit des deux trafics

Nous testons les performances d'un ordonnanceur RED auquel on associe les mécanismes AF/AR de filtrage et de reconstruction des paquets d'acquittements et l'impact d'une telle configuration sur les débits des trafics ascendants et descendant.

FIG. 2.10 – *Variation du débit moyen des trafics avec RED*

Nous choisissons RED avec les paramètres suivants: 18 pour le seuil maximal de la taille de la file d'attente th_max, 10 pour le seuil minimum th_min, 0.011 pour l'estimateur exponentiel w_q et 0.45 pour le taux maximum d'erreurs max_p (c.f. 2.1.3.2). En choisissant ces paramètres, nous voulions que la file d'attente soit capable d'absorber des rafales de trafics, c'est pourquoi nous fixons d'assez grandes valeurs pour les seuils. D'autre part, nous voulions que RED soit sévère vis à vis des paquets qui ont dépassé le th_max pour ne pas monopoliser la file d'attente, nous avons donc fixé max_p à 0.45 [23]. Les résultats sont dans le figure 2.10. Concernant le trafic ascendant T_0, l'application de RED donne des débits moyens des connexions aux alentours de 6 Kbit/s, ce qui est considéré acceptable vu que nous avons ici 10 connexions se partageant une bande passante de 56 Kbit/s [4]. RED associé à AF/AR est donc bénéfique pour le trafic ascendant.
Concernant le trafic descendant T_1, RED avec AF/AR réussit à améliorer les débits moyens de ce trafic (100 Kbit/s en fin de connexion) comparé à FIFO avec AF/AR (60 Kbit/s). Ces débits moyens sont tout de même plus petits que ceux atteints en appliquant le mécanisme *AckFirst*(140 Kbit/s).

Nous pouvons donc dire que RED avec AF/AR parvient à avoir un bon compromis entre augmenter les débits moyens du trafic ascendant et augmenter les débits moyens du trafic descendant. Dans ce qui suit, nous détaillons plus cette étude en regardant la distribution des débits entre les différentes connexions des trafics. En effet, ceci va nous permettre d'avoir une idée plus précise sur l'impact de AF/AR avec ou sans AckFirst, avec ou sans le mécanisme RED sur les trafics ascendants et descendants.

2.4.3.1 Distribution des débits moyens des connexions du trafic ascendant

Afin d'avoir une idée plus précise sur la distribution des débits moyens entre les différentes connexions des trafics, nous effectuons des simulations qui représentent la

4. Les approximations de simulations sont la cause de cette légère différence entre $10 * 6Kbit/s$ et 56Kbit/s.

FIG. 2.11 – *Densité des débits moyens du trafic ascendant*

distribution des débits. Nous avons rajouté des simulations impliquant RED *Random Early Detection* dans les routeurs avec les mécanismes AF/AR. Le problème étant ici une saturation de la file d'attente du routeur à l'entrée du lien lent, nous avons choisi d'utiliser RED pour voir si une gestion active de file d'attente est suffisante pour améliorer les débits moyens des connexions.

Concernant le trafic ascendant, la distribution des débits des connexions est représentée dans la figure 2.11. La figure montre qu'une bonne partie des connexions de ce trafic, à savoir 35 pour cent, a un débit moyen aux alentour de 5 Kbps dans le cas où l'on a FIFO avec AF/AR dans le routeur à l'entrée du lien asymétrique. L'application d'une plus haute priorité aux acquittements donne 100 pour cent des connexions ayant un débit moyen nul. Ceci renforce notre raisonnement: l'ordonnancement AckFirst dans le cas d'un trafic bidirectionnel passant par un lien asymétrique est absolument à rejeter si on veut optimiser les performances des connexions du trafic ascendant. L'utilisation de RED avec AF/AR améliore légèrement les débits et donne 30 pour cent des connexions ayant leur débit moyen aux alentours de 6 Kbps.

2.4.3.2 Distribution des débits moyens des connexions du trafic descendant

Concernant le trafic descendant T_1, les résultats des variations de la distribution des débits moyens entre les 10 connexions sont reportés dans la figure 2.12. La figure montre qu'en cas d'utilisation de AF/AR, une bonne partie des connexions obtient un débit moyen aux alentours de 50 Kbit/s (7.5% des connexions a un débit moyen égal à 50 Kbps, 7.5% a un débit moyen égal à 49 Kbit/s et 7.5% a un débit moyen égal à 48 Kbit/s). L'utilisation de RED améliore ce résultat avec la même proportion des connexions (7.5%) atteignant un débit moyen aux alentours de 75 Kbps. Finalement, l'utilisation de AckFirst donne 27 pour cent des connexions ayant un débit moyen de 200 Kbps. AF/AR avec AckFirst obtient encore un fois les meilleures performances si

FIG. 2.12 – *Densité des débits moyens du trafic descendant*

le but est d'optimiser les performances du trafic descendant.

2.4.4 Influence du degré d'asymétrie

FIG. 2.13 – *Influence du degré d'asymétrie sur le débit moyen du trafic ascendant* T_0

Nous définissons le degré d'asymétrie comme étant le rapport entre la capacité disponible sur le chemin aller et la capacité sur le chemin retour, c'est à dire que nous avons:

$$k = \frac{CapaciteLienAller}{CapaciteLienRetour}$$

Dans cette section, nous nous intéressons à étudier l'impact d'un changement dans le degré d'asymétrie k sur le comportement des mécanismes de filtrage et de reconstruction des paquets d'acquittement.
Nous avons effectué des simulations en changeant la capacité du lien bas débit. Pour le trafic ascendant T_0, les résultats des simulations sont reportés dans les figures 2.13. Nous avons varié k entre la valeur 1 (qui traduit donc un lien symétrique) et la valeur 200 (qui traduit une très forte asymétrie en terme de capacité en bande passante disponible).
Dans le cas d'un trafic symétrique, la présence d'un trafic bidirectionnel implique une faible asymétrie implicite puisque les paquets de données et les paquets d'acquittements partagent les mêmes liens et les mêmes ressources. Dans ce cas, AF/AR avec *AckFirst* arrive à obtenir un bon compromis entre les débits des trafics descendants et ascendants, le débit moyen du trafic ascendant est donc bon.
Cependant dès que k dépasse 10, les débits deviennent catastrophiques pour le trafic ascendant, et ce dans le cas où on utilise AF/AR avec ou sans *AckFirst*. AF/AR n'arrive donc pas à surmonter les difficultés dans lesquelles se retrouve le trafic ascendant T_0.

FIG. 2.14 – *Influence du degré d'asymétrie sur le débit moyen du trafic descendant* T_1

Concernant le trafic descendant, les résultats des simulations sont représentés dans la figure 2.14. Comme le montre la figure, l'impact de k est moins important que pour le trafic ascendant mais les débits diminuent aussi fortement si on augmente le degré d'asymétrie, sutout en ayant AF/AR comme ordonnanceur au niveau du routeur. AF/AR avec *AckFirst* donne les meilleurs résultats concernant le trafic descendant en cas de forte asymétrie. On ne peut visiblement pas affirmer non plus que RED avec AF/AR aide le trafic descendant à ne pas succomber aux problèmes causés par de fortes asymétries.

2.4.5 Influence du nombre de connexions

FIG. 2.15 – *Influence du nombre de connexions sur le débits moyens du trafic ascendant* T_0

Nous nous sommes aussi intéressés à l'impact d'un changement dans le nombre de connexions sur les débits moyens des connexions [85]. Nous savons que le mécanisme de filtrage des paquets d'acquittement filtre des acquittements de connexions déjà présentes dans la file d'attente du routeur. Intuitivement, nous en concluons que l'augmentation du nombre de connexions impliquées dans les simulations aura un impact négatif sur le mécanisme de filtrage des acquittements. En effet, AF va effectuer son opération de filtrage des acquittements moins souvent, vu qu'en augmentant le nombre de connexions, la probabilité que deux paquets d'acquittements appartenant à la même connexion se retrouvent dans la file d'attente diminue.

Les résultats des simulations concernant le trafic ascendant sont représentés dans les figures 2.15. Pour T_0, dès que le nombre de connexions dépasse cinq connexions, l'utilisation du mécanisme *AckFirst* donne des débits très faibles et inacceptables pour l'utilisateur.

Concernant le trafic descendant, les résultats des simulations sont repésentés dans la figure 2.16. En comparaison avec le trafic ascendant, T_1 est moins sensible au changement dans le nombre de connexions et AF/AR avec AckFirst est, pour ce trafic, la solution qui donne les plus fort débits.

2.4.6 Récapitulatif des résultats des simulations

L'étude que nous avons effectué dans cette section nous permet de dire que la coopération "AF+AR+AckFirst" est incapable de résoudre le problème que rencontre tout trafic TCP bidirectionnel passant par un lien asymétrique et d'améliorer les débits

FIG. 2.16 – *Influence du nombre de connexions sur les débits moyens du trafic descendant* T_1

moyens des trafics. En effet:

- Soit nous appliquons AF/AR avec le mécanisme AckFirst donc en donnant une plus forte priorité aux paquets d'acquittements. Dans ce cas nous augmentons les débits moyens des connexions appartenant au trafic descendant mais nous réduisons fortement les débits moyens des connexions appartenant au trafic descendant. Cette solution est donc à rejeter.
- Soit nous appliquons AF/AR sans le mécanisme AckFirst. Dans ce cas nous augmentons les débits moyens des connexions appartenant au trafic ascendant mais nous réduisons fortement les débits moyens des connexions appartenant au trafic descendant. Cette solution est à rejeter aussi.

D'autre part, le moindre changement dans les paramètres de la topologie des simulations a un impact direct sur les résultats. Une diminution dans la capacité en bande passante du lien ascendant traduisant donc une assez forte asymétrie du lien implique des performances médiocres pour les deux trafics, particulièrement pour le trafic ascendant, et aucun des mécanismes proposés ne parvient à améliorer ces performances. Une augmentation dans le nombre des connexions impliquées traduisant un assez forte charge du réseau implique aussi des performances médiocres. L'hétérogénéité du trafic du réseau Internet ne cesse de croître, ce qui implique la nécessité de s'assurer de la robustesse de tout mécanisme visant à maximiser les performances.

2.5 Conclusion

Dans ce chapitre, nous avons décrit l'état de l'art actuel concernant les réseaux asymétriques dans l'Internet et en particulier concernant les trafic bidirectionnels qui

empruntent ce type de réseaux. Une asymétrie dans la bande passante disponible sur le lien aller et sur le lien retour cause d'importantes difficultés pour le protocole TCP qui devient incapable de réguler correctement son trafic et de se protéger des éventuelles congestions du réseau. L'ajout d'un trafic bidirectionnel exacerbe ces difficultés puisque paquets d'acquittements et paquets de données se partagent des liens et des ressources. Ce partage est absolument à éviter puisque ces paquets sont très différents de par leur taille et leur réaction aux notifications de congestion.

Nous avons ensuite énuméré les solutions qui ont été publiées dans la littérature afin de remédier à ces problèmes. Nous avons analysé plus en détail la solution de filtrage des acquittements (AF) suivie de leur reconstruction (AR). Dans le cas de trafic bidirectionnel, on recommande l'utilisation d'une plus forte priorité pour les paquets d'acquittements afin d'optimiser les performances (*AckFirst*). L'ajout du mécanisme RED dans le routeur à l'entrée du lien bas débit permet de réduire la taille moyenne de la file d'attente et évite de monopoliser l'accès au lien par l'un ou l'autre des trafics.

Nous avons mené des simulations qui ont démontré cependant clairement que ces solutions ne suffisent pas du tout à optimiser les performances de trafic bidirectionnel passant par un lien asymétrique. En effet, soit on augmente les performances du trafic descendant au dépens du trafic ascendant soit c'est le trafic ascendant qui est optimisé réduisant alors fortement les performances du trafic descendant. Nos simulations mettent clairement en évidence l'insuffisance de ces solutions, et ceci même si elles sont associées à des mécanismes de gestion active de file d'attente tel que RED puisque nous avons pu avoir un bon compromis entre performances de trafic ascendant et performances de trafic descendant. Ce compromis est toutefois très instable et très sensible au moindre changement dans les paramètres de la topologie, donnant dans certains cas des perforamnces médiocres. Nous avons donc mis en évidence la nécessité de concevoir des mécanismes capables d'optimiser à la fois les deux trafics descendant et ascendant et d'assurer ainsi une utilisation optimale des ressources du réseau.

Dans le chapitre suivant, nous proposons un premier mécanisme qui vise à maximiser les performances de trafics bidirectionnels passant par un lien asymétrique. L'objectif plus précisément est de maximiser l'utilisation du lien asymétrique, utilisation qui sera représentée par une fonction d'utilité. Ce mécanisme est appelé ACQ de l'anglais *Adapative Class-based Queuing* et parvient à améliorer les performances et à maximiser la fonction d'utilité.

Chapitre 3

ACQ: un mécanisme d'ordonnancement adaptatif

Dans ce chapitre, nous développons un mécanisme d'ordonnancement entre paquets de données et paquets d'acquittements à l'entrée d'un lien asymétrique, afin d'améliorer les performances globales[1]. Nous présentons ACQ (***A****daptive* ***C****lass-based* ***Q****ueuing*), qui est un mécanisme adaptatif se basant sur les classes de trafics pour optimiser les performances dans un environnement de trafic bidirectionnel passant par un lien asymétrique. Nos motivations pour concevoir un mécanisme se basant sur les classes de trafics et non sur une adaptation de la gestion active de files d'attente (tel que RED utilisé et simulé dans le chapitre 2) sont explicitées clairement dans la suite de ce chapitre (c.f. 3.1.2). ACQ est donc un ordonnanceur à placer à l'entrée du lien bas débit. Il nécessite la mise en place de deux classes de paquets, une pour les paquets de données et une pour les paquets d'acquittements. La particularité de ACQ est de pouvoir s'adapter au trafic qui passe sur le lien asymétrique en modifiant les poids des deux classes afin de maximiser les performances des deux trafics.
Avant de présenter plus en détail ACQ, nous commençons par rappeler quelques principes utilisés recemment par les concepteurs de réseaux afin d'assurer une certaine qualité de service à l'utilisateur.

3.1 Nouvelles approches pour améliorer le service de l'Internet

Le réseau Internet a été conçu afin d'assurer un service dit *"best-effort"*, c'est à dire assurant à l'utilisateur un acheminement de paquets sans garantie de fiabilité ou de délai. De nouvelles applications ont vu le jour: elles ont des exigences en terme de débit, de délais ou de tout autre critère de qualité de service. En effet, et avec l'évolution de l'Internet vers des applications de plus en plus variées, tout critère de validation d'une

1. Notre ordonnanceur ACQ est aussi utilisable dans le cas d'un lien symétrique, ses performances sont aussi optimales dans un tel cas.

conception doit répondre d'abord à la contrainte de satisfaction de l'utilisateur. Il ne sera donc plus question uniquement de mesurer les performances d'un réseau en terme de quantités spécifiques au réseau telles que le nombre de paquets rejetés ou la puissance des signaux émis mais il faudra également évaluer le degré avec lequel le réseau satisfait ou pas les critères de service de chaque utilisateur d'application.

Dans cette section, nous décrivons plus en détail la signification du terme "satisfaction de l'utilisateur". En faisant référence à la littérature, nous répondons ensuite à la question suivante: comment quantifier cette satisfaction afin de pouvoir la maximiser? Ensuite nous decrivons les outils les plus significatifs déjà existants qui permettent d'atteindre cette satisfaction. Cette section nous aidera à expliquer les principes de notre ordonnanceur ACQ.

3.1.1 Satisfaction de l'utilisateur et fonctions d'utilité

Un utilisateur d'une application spécifique du réseau est plus ou moins satisfait des services proposés par l'opérateur de ce réseau selon que ses critères de service sont plus ou moins atteints. Ces critères peuvent être des délais minimums d'attente, des débits importants, etc. L'opérateur doit donc pouvoir modéliser les critères de qualité de service des utilisateurs de son réseau afin de leur offrir le maximum de qualité de service et de s'assurer de leur fidélité.
Nous appelons satisfaction de l'utilisateur la maximisation d'une fonction d'utilité fixée auparavant [43]. Une fonction d'utilité est une fonction qui dépend, selon sa complexité, d'une ou de plusieures variables et dont l'étude de variation permet de connaître la ou les valeurs de ces variables qui permettent à cette fonction d'être optimale en ce ou ces points. L'opérateur devra donc oeuvrer à avoir toujours cette fonction d'utilité aux alentours de ces valeurs optimales [17].

Il est possible de formaliser la fonction d'utilité comme étant la trace du service fourni tout au long de l'exécution de l'application. On note cette fonction U. Augmenter U revient à augmenter les performances de l'application en question. La fonction d'utilité traduit en fait la dépendance entre service fourni et performances de l'application.
Les fonctions d'utilité varient avec l'application [8, 50]. Il faut donc définir les fonctions d'utilité en fonction des besoins des applications et donc de l'utilisateur de ces applications. Il peut s'agir de maximiser les débits aller et retour, maximiser l'utilisation des liens, maximiser l'équité entre les deux connexions aller et retour, etc. Dans la littérature, plusieurs fonctions d'utilité sont connues comme des standards pour quelques critères de qualité de service. En voici une liste exhaustive classées selon les critères auxquels elles obéissent:

Respecter des critères de débits Il est possible de quantifier la qualité de service d'une source audio ou autre en fonction du débit. Les applications de données traditionnelles comme le transfert de fichiers sont tolérantes aux pertes et retard de paquets. Leur fonction d'utilité est alors intuitivement strictement concave. D'autres applications sont plus sensibles aux débit, telles que les applications multimédia, et

auront donc des fonctions d'utilité convexe autour de zéro. Les variations de la fonction

FIG. 3.1 – *Fonctions d'utilité dépendant du débit moyen*

d'utilité qui dépend du débit sont représentées dans la figure 3.1. La première figure montre une fonction d'utilité concave autour de zéro représentant par exemple une applications FTP de transfert de fichier. Une telle courbe de la variation de la fonction d'utilité veut dire que donner à deux applications un débit de 1 donne une somme des valeurs d'utilité supérieure à celle obtenue en donnant un débit de 2 à une seule application. La deuxième courbe en revanche est convexe, et donner le maximum à une seule application est souhaitable dans ce cas afin de maximiser l'utilité. Elle reflète une application multimédia par exemple où un minimum de débit est exigé et où la qualité augmente lentement pour de très bas débits inférieurs au débit minimum éxigé par l'application, puis augmente rapidement pour des débits intermédiaires et encore lentement pour des débits élevés.

Respecter des critères de délai Avec les applications nouvelles telles que l'audio et la vidéo, le délai représente un des critères les plus importants que l'opérateur se doit de pouvoir assurer aux utilisateurs de ces applications. Plusieurs critères de délai ont été étudiés dans la littérature, et à chacun une fonction d'utilité que l'opérateur doit optimiser a été établie.
Plusieurs études ont établi que, pour les applications interactives, le délai de bout-en-bout doit approximer 150 ms. D'autre part, il est aussi reconnu qu'un utilisateur d'application de téléphonie trouvant des délais supérieur à 300ms ne se considère plus en conversations interactives mais plutôt en connexions semi-duplex. De ces considérations sont nées des fonctions d'utilité dont les courbes suivent les modèles de la figure 3.2 et qui suivent le comportement suivant: pour des délais inférieurs au seuil critique de 150 ms, la qualité décroit très lentement de telle sorte que l'utilisateur ne bénéficie pas d'avoir un délai de bout-en-bout plus bas. Pour des délais supérieurs, la qualité diminue brutalement de telle sorte que la moindre augmentation de délai altère fortement l'interactivité. Enfin, si le délai dépasse 300 ms et puisque la conversation est

FIG. 3.2 – *Variation d'une fonction d'utilité dépendant du délai moyen*

considérée de toute façon en semi-duplex, toute augmentation supplémentaire du délai affecte seulement la qualité qui est déjà trop basse.

Combiner plusieurs critères Certaines applications nécessitent d'associer plusieurs critères et autres contraintes de qualité de service. Par exemple, l'utilisateur peut exiger des contraintes de délai et des contraintes de débit pour des applications où, à part le délai de bout-en-bout, un débit minimum est fondamental pour optimiser les performances de ces applications. Les variations de telles fonctions d'utilités sont dans ce genre de situation beaucoup plus complexes que des cas linéaires (dépendant d'une seule variable) et correspondent plus à des variations en N-dimensions que l'opérateur doit optimiser.

$\Longrightarrow$ De nouveaux principes de conception sont utilisés de nos jours. Utiliser différentes fonctions d'utilités afin d'assurer une satisfaction de l'ensemble des utilisateurs permet aux opérateurs et autres fournisseurs de service, de modéliser les critères de leurs utilisateurs, de les quantifier et enfin de pouvoir y répondre le plus fidèlement possible. Les fonctions d'utilité représentent dans l'état actuel du réseau Internet une approche plus réaliste parce qu'elle tient en compte des besoins réels des utilisateurs. C'est pourquoi, toute étude d'un nouveau mécanisme d'ordonnancement ou de routage, se doit d'optimiser une fonction d'utilité établie auparavant afin de cibler directement les critères de satisfaction et afin d'intéresser les opérateurs par l'applicabilité de ces mécanismes.

3.1.2 Supports pour assurer une qualité de service au sein du réseau

Il est de plus en plus proposé dans de récentes publications de ne plus considérer le réseau Internet comme une boîte noire, mais d'y inclure des mécanismes ayant pour objectifs d'améliorer les performances du protocole TCP [71]. Inclure des mécanismes se fait entre autres au niveau des routeurs soit en rajoutant une gestion active de leur file d'attente (AQM:*Active Queue Management*), soit en incluant de nouveaux mécanismes

d'ordonnancement de paquets. Dans le chapitre précédent, nous avons mené des simulations visant à évaluer les performances des mécanismes AF/AR et *AckFirst* dans un environnement bidirectionnel asymétrique avec un mécanismes de gestion active de file d'attente, en l'occurence RED (*Random Early Detection*), dans les routeurs. La question est de savoir si les mécanismes de AQM en général, et RED en particulier, peuvent fournir les supports nécessaires et adéquats afin d'assurer une qualité de service au sein du réseau. Comme décrit dans 2.1.3.2, en résumé, les routeurs munis de la discipline de gestion active de file d'attente permettent d'atteindre les objectifs suivants:

- Réduire le nombre de paquets rejetés: puisque réduire la taille des files permet d'absorber des *bursts* et par conséquent un plus grand nombre de paquets seront acceptés.

- Réduire le délai pour des services interactifs: en trouvant des tailles de files d'attente réduites, les paquets attendent moins et donc parviennent plus vite à destination.

- Remédier au problème de *Lock-Out*: c'est le fait d'avoir le monopole de la file d'attente par une seule connexion ou un flux de trafic.

Ceci dit, il est important de noter que, appliquer un algorithme de gestion active de files d'attente à des routeurs ne suffit pas pour acquérir l'équité entre les flux, par équité on entend justice entre les différentes classes de services ou *fairness*. Plusieurs type d'équité sont définies dans TCP telle que l'équité max-min ou l'équité *equal share* **??**. Cependant, celle qui est la plus utilisée avec les protocoles de congestion et de flux de TCP est l'équité proportionnelle. Cette équité va permettre de répondre aux exigences de qualité de service d'un trafic. En effet dans un routeur utilisant une gestion active de file d'attente tout en appliquant un ordonnancement de type FIFO, il a été démontré dans [71] que le débit d'une source TCP (ou ayant un comportement similaire auquel cas elle est appelée *TCP friendly*) peut s'exprimer par la relation:

$$Débit_{TCP} = Cste\frac{MTU}{RTT.\sqrt{Loss}}$$

où
MTU: Taille des paquets, RTT: délai d'aller-retour et Loss: taux de perte de paquets

Par conséquent, deux flux TCP peuvent recevoir des débits très différents seulement parce qu'ils ont des RTTs différents, et un flux qui n'utilise pas de contrôle de congestion peut se voir attribuer un débit plus grand qu'un autre qui l'utilise.
Afin d'avoir un niveau maximum d'équité et donc d'avoir l'opportunité d'assurer un degré maximal de qualité de service, il est nécessaire d'avoir un état pour chaque flux de trafic ou un état pour chaque agrégat de trafic (un agrégat de trafic est un ensemble de flux présentant une ou plusieurs caractéristiques communs qui définissent un critère particulier).

Ceci est obtenu en utilisant des algorithmes d'ordonnancement tels que CBQ(*Class-Based Queuing*) ou FQ (*Fair Queuing*, c.f. 4.1.1.2). Nous décrivons dans ce qui suit CBQ: un mécanisme de support de qualité de service dans l'Internet se basant sur la classification des trafics et sur lequel repose notre mécanisme ACQ proposé dans ce chapitre.

Class-Based Queuing (CBQ) Dans [30], Jacobson et Floyd proposent une organisation hiérarchique en classes des paquets. En effet, les paquets sont classifiés selon certains critères répondant aux demandes des utilisateurs, et constituant ainsi une hiérarchie de classes. A chaque classe est alors allouée une portion de la bande passante. Un mécanisme similaire aux horloges virtuelles permet de surveiller la consommation de chaque classe, de distribuer un excédent de bande passante non utilisé par une classe entre les classes restantes et de "punir" les classes "gourmandes" qui ne respectent pas leur part de bande passante. En fait, on mémorise pour chaque classe la fin théorique d'émission du paquet F_i. On mesure ensuite un intervalle moyen entre deux arrivées IM_i suivant la formule suivante dite formule du lissage exponentiel:

$$IM_i = (1 - a) \times IM_{i-1} + a \times (t - F_i)$$

où a est une constante et t la date réelle d'arrivée du paquet. Dans le cas où IM_i est positif, la classe est "sous-consommatrice", sinon elle est "sur-consommatrice". Ce calcul est effectué à tous les niveaux de la hiérarchie.

FIG. 3.3 – *Politique de partage de bande passante dans CBQ*

CBQ est illustré dans la figure 3.3. Il permet d'éviter qu'un trafic monopolise les ressources du réseau, il a été essentiellement implémenté par Wakeman, Ghosh et Crowcroft [80], d'après la spécification de Jacobson et Floyd. Le trafic dans CBQ étant séparé

en classes, il devient possible de spécifier des prévisions d'utilisation des liens par profil d'applications et de répondre ainsi aux demandes des utilisateurs du réseau [21].

L'ordonnancement entre les files d'attente des diffrentes classes est régi par le mécanisme WRR *Weighted Round Robin* qui partage la bande passante entre ses classes en utilisant la technique du "tourniquet pondéré". Un poids est associé à chaque classe. Toutes les classes qui ont suffisamment de demandes obtiendront la bande passante proportionnellement à leur poids. Ces poids peuvent être configurés pour décroître automatiquement pour les classes transférant moins de données [21]. Une fois qu'une classe est servie, WRR passe à la classe d'après et ainsi de suite. CBQ/WRR permet d'atteindre un niveau d'équité optimal.

Dans la section suivante, nous décrivons notre modèle d'ordonnancement des trafics bidirectionnels passant par un lien asymétrique. Notre modèle se base sur une organisation hiérarchique des classes. Nous avons choisi au départ une fonction d'utilité afin de pouvoir mener notre modélisation. Nous avons ensuite étendu notre modèle à d'autres fonctions d'utilité impliquant d'autres besoins d'utilisateurs.

3.2 Le modèle ACQ: Adaptive Class-based Queuing

Comme dans le chapitre précédent, nous appelons connexions descendantes les connexions ayant leurs paquets de données sur le chemin aller à haut débit et réciproquement, nous appelons connexions ascendantes les connexions ayant leurs paquets de données sur le chemin de retour à bas débit. Avant de décrire le modèle de l'ordonnanceur ACQ, précisons les notations suivantes:

- x_f: débit de la connexion descendante ("f" pour dire débit *forward*)
- x_r: débit de la connexion ascendante ("r" de *reverse*)
- $U_f(x_f)$: Satisfaction de l'utilisateur des connexions descendantes
- $U_r(x_r)$: Satisfaction de l'utilisateur des connexions ascendantes

Le problème est donc de trouver les valeurs de x_f et de x_r qui maximisent la satisfaction globale de l'utilisateur qui correspond à la somme des deux satisfactions. En d'autres termes nous avons :

$$U(x_f,x_r) = U_f(x_f) + U_r(x_r) \tag{3.1}$$

Le principe de ACQ est de séparer les flux de données et les flux d'acquittements présents sur le lien ascendant en deux classes, par un mécanisme *"two-class CBQ"* décrit dans la section précédente. Les paquets d'acquittements et les paquets de données sont donc separés en deux classes et servis tour à tour selon un mécanisme de *Weighted Round Robin* (c.f. 3.1.2). La question qu'on pourrait se poser est comment choisir les poids des deux classes; en effet donner un plus grand poids à la classe des données

pénalise le flux d'acquittements qui à son tour pénalise automatiquement le flux de données sur le lien aller. Inversement, donner plus de poids aux acquittements pénalise fortement les connexions ascendantes. En résumé, l'allocation des poids des classes doit être faite de manière à maximiser l'utilité "globale" et assurer la satisfaction de l'utilisateur qui ne veut voir affectées ni ses connexions ascendantes et ses connexions descendantes. Les poids des classes doivent s'adapter à d'éventuels changements dans les conditions de trafics tout en conservant une satisfaction maximale pour l'utilisateur d'un tel environnement.

Le principe de l'ordonnancement ACQ se base sur un algorithme simple pour adapter les poids des classes à l'entrée du lien ascendant. On considère que les acquittements et les paquets de données sont mis dans deux files d'attente séparées et sont servis selon un mécanisme de *"Weighted Round Robin"*. On suppose que le buffer ACQ a la faculté de mesurer le débit des paquets de données dans les deux directions. Il est en effet tout à fait possible de mesurer le débit des données sur le lien descendant:

- Si ACQ est dans un routeur propriétaire d'accès au réseau, il suffit de regarder l'espacement d'arrivée des acquittements: cette solution est possible malgrè le fait que le trafic TCP est "byte-stream". En effet, ACQ manipule des agrégats de trafics, il n'est donc pas nécessaire d'avoir un état de chaque connexion, et un simple compteur du nombre total d'acquittements qui passent suffit, sans distinguer entre les connexions.

- Si ACQ se trouve dans un routeur interne du réseau, dans le cas où le routeur est à faible capacité de stockage et donc ne permet pas d'avoir en mémoire l'espacement d'arrivée des acquittements, la mise en place d'un mécanisme de signalisation entre ce routeur et celui d'accès au lien descendant permet d'accéder à l'information du débit sur le lien descendant.

A chaque intervalle de temps T, notre algorithme mesure le débit des deux flux opposés et adapte le debit alloué à la classe des données ascendantes en changeant le poids de la classe. La classe des acquittements se verra allouée le reste de la bande passante.
Soient les notations suivantes:

- r_n: débit alloué à la connexion ascendante à l'instant nT (n $\in$ N). Ce débit reste le même jusqu'à l'instant $(n+1)$T

- $x_r(n)$: débit mesuré de la connexion ascendante entre l'instant nT et (n+1)T

- $x_f(n)$: débit mesuré de la connexion descendante entre l'instant nT et (n+1)T

On utilise la méthode de la projection du gradient pour mettre à jour la valeur de r_n [50, 39] de facon à approcher le point optimal,

$$r_n = x_r(n) + \gamma \frac{dU(x_f,x_r)}{dx_r}(n) \tag{3.2}$$

Cette méthode suppose l'existence d'un maximum d'utilité dans la région de définition de x_r et x_f. γ est une constante qui effectue un compromis entre stabilité et rapidité de convergence. Nous en parlerons plus en détail dans la section suivante. L'expérimentation nous donnera la valeur optimale de γ.
On écrit,

$$r_n = x_r(n) + \gamma \frac{dU(x_r(n))}{dx_r} + \gamma \frac{dU(x_f)}{dx_f} \cdot \frac{dx_f}{dx_r}(n) \tag{3.3}$$

Etant donné qu'on ne connaît pas la relation entre dx_r et dx_f, nous proposons l'approximation suivante:

$$\frac{dx_f(n)}{dx_r} = \frac{x_f(n)-x_f(n\text{-}1)}{x_r(n)-x_r(n\text{-}1)} \tag{3.4}$$

Nous obtenons alors la règle suivante pour mettre à jour le débit alloué aux connexions circulant sur le lien retour:

$$r_n = x_r(n) + \gamma \frac{dU(x_r(n))}{dx_r} + \gamma \frac{dU(x_f)(n)}{dx_f} \cdot \frac{x_f(n)-x_f(n\text{-}1)}{x_r(n)-x_f(n\text{-}1)} \tag{3.5}$$

Fonction d'utilité Comme dit précédemment dans ce chapitre, la fonction d'utilité dépend du type d'application et des exigences de l'utilisateur. Elle dépend de l'application utilisant TCP, du coût en terme de bits par seconde dans chaque direction. Elle peut aussi dépendre du taux d'équité ou "*fairness*" entre les différentes connexions dans un des sens ou dans les deux. Plusieurs fonctions d'utilité peuvent être imaginées, nous allons étudier celles qui nous ont semblées les plus probables de la part de l'utilisateur ou de l'opérateur.

Nous considèrons que notre fonction d'utilité que ACQ tente de maximiser est égale à l'utilisation de la bande passante du lien asymétrique. Afin d'obtenir la satisfaction de l'utilisateur, il faudrait donc maximiser la somme des utilisations des bandes passante dans les deux sens. Cette fonction d'utilité varie avec les poids des deux classes allouées par ACQ. Elle suit une courbe de la forme de la Figure 3.4. Cette courbe montre bien la dépendance entre les liens aller et les liens retour

FIG. 3.4 – *Variation de la fonction d'utilité utilisée par ACQ*

puisque la fonction d'utilité part d'un minimum égal à 1 et correspondant à une utilisation maximale d'un lien au dépens de l'autre. La courbe montre l'existence d'un maximum théorique égal à 2 et correspondant à une maximisation de l'utilisation des deux liens en même temps pour une certaine allocation de la bande passante entre les deux trafics. Le but est donc pour ACQ de toujours fournir les portions de la bande passante du lien ascendant à allouer aux deux classes (données et acquittements) afin d'atteindre le maximum de la fonction d'utilité.

Soit C_r la bande passante disponible sur le lien ascendant et réciproquement C_f la bande passante disponible sur le lien descendant. L'utilisation de la bande passante ascendante (repectivement descendante) est en fait le rapport entre le débit atteint par les connexions ascendantes (respectivement descendantes) et la capacité maximale du lien ascendant (respectivement descendant). On a donc

$U_r(x_r)$ = Utilisation de bande passante du lien ascendant = $\frac{x_r}{C_r}$
Nous avons donc $\frac{dU(x_r)}{dx_r} = \frac{1}{C_r}$

$U_f(x_f)$ = Utilisation de bande passante du lien descendant = $\frac{x_f}{C_f}$
Et donc $\frac{dU(x_f)}{dx_f} = \frac{1}{C_f}$

Ce qui implique l'équation de mise à jour des poids des classes à laquelle obéit ACQ suivante en remplaçant les valeurs de l'équation (3.5),

$$r_n = x_r(n) + \gamma \left(\frac{1}{C_r} + \frac{1}{C_f} \frac{x_f(n) - x_f(n\text{-}1)}{x_r(n) - x_r(n\text{-}1)} \right) \tag{3.6}$$

A chaque intervalle de temps T, le routeur remet à jour les poids alloués à la classe des paquets de données en respectant cette équation. La classe des acquittements se verra attribuer le reste de la bande passante encore disponible.

Seulement, un trafic TCP est totalement hétérogène et certains effets de bords peuvent arriver amenant à une situation un peu spéciale où une des deux classes tente de monopoliser toute la bande passante disponible. Vu que ACQ parvient à un état stable de façon progressive, une telle situation peu vite nous éloigner de l'état stable et est donc fortement à éviter. C'est pour cela que nous avons aussi mis en place un comportement spécial de ACQ dans ces situations.

Nous définissons le poids de la classe des paquets de données comme étant la variable $X_r(n) = \frac{r_n}{C_r}$, $X_r(n) \in [0,1]$.
Le poids de la classe des acquittements est donc égal à 1-$X_r(n)$.

Dans le cas où c'est la classe des paquets de données qui veut monopoliser la bande nous aurons

$$X_r(n) = C_r$$

sinon nous obtiendrons le résultat suivant pour le débit de la classe de données

$$X_r(n) = 0.$$

Ces deux cas sont absolument à éviter parce qu'ils sont l'un et l'autre contraire à ce qu'on a défini comme étant la satisfaction de l'utilisateur. C'est pourquoi nous avons fixé à ACQ des limites dans son allocation de la bande passante. Si les mesures et le calcul de l'équation (3.6) fournit un résultat où une classe tend à monopoliser la bande passante, ACQ garde toujours une portion de la bande passante à l'autre classe, afin de la protéger et de ne pas s'éloigner du cas optimal. Nous avons fixé cette portion à vingt pour cent, en d'autres termes:
- si $X_r(n) \geq C_r \Rightarrow X_r(\mathrm{n}) = 80\ \%$ de C_r
- si $X_r(n) \leq 0 \Rightarrow X_r(\mathrm{n}) = 20\ \%$ de C_r.

Nos simulations ont cependant démontré que de tels cas sont vraiment particuliers, c'est à dire que grâce au principe même de ACQ, l'équation (3.6) fournit toujours des résultats inclus dans la bande passante disponible. De tels cas arrivent par exemple si un gros trafic s'effondre brusquement. Nous avons tout de même choisi la sécurité en fixant ce comportement de ACQ si de tels cas arrivaient.

Nous montrons dans les sections suivantes que ACQ ainsi configuré permet d'atteindre la satisfaction de l'utilisateur puisqu'il parvient à maximiser l'utilisation du lien asymétrique et à optimiser les débits des deux trafics. Mais avant cela, nous décrivons plus en détail les deux seuls paramètres de cet ordonnanceur: T et γ. Nous donnons dans cette section une stratégie de choix des valeurs optimales de ces paramètres.

3.2.1 Description détaillée de ACQ

FIG. 3.5 – *Architecture de ACQ*

Nous reprenons la figure 1.1 et rajoutons l'emplacement et le mode de fonctionnement de ACQ. La Figure 3.5 est donc une représentation plus détaillée. ACQ est placé à l'entrée du lien bas débit et attribue des classes différentes aux paquets de données et paquets d'acquittements. Au lancement de l'ordonnanceur ACQ, les deux classes recoivent deux part équivalente de la bande passante. L'objectif de ACQ est de trouver le plus vite, le schéma d'allocation de la bande qui permet une performance optimale du trafic.

A chaque intervalle de temps T, ACQ mets à jour $X_r(n)$ selon l'équation (3.6). Pour assurer un fonctionnement optimal de ACQ, deux questions se posent:

Quelle doit être la fréquence de rafraîchissement de l'ordonnanceur?
Quelle doit être le valeur de la variable γ?

Dans cette section, nous expliquons les deux seuls parametres de cet ordonnaceur. Nous ne prétendons pas l'existence de deux constantes fixes optimales pour T et γ, nous donnons cependant une stratégie pour leur choix en section 3.2 où nous nous appuyons en plus sur les résultats de simulation qui confirment notre stratégie.

3.2.1.1 La fréquence de rafraîchissment de ACQ

Notre objectif ici est de définir la valeur de l'intervalle T au bout duquel l'ordonnanceur doit mettre à jour les allocations des deux classes. T représente en fait un compromis entre stabilité et réponse du système.
Faisons une schématisation un peu simpliste du système mais qui va nous permettre de mieux comprendre l'importance de ce paramètre. Lorsque ACQ alloue une importante portion de la bande passante à une des classes, celle ci doit en tirer parti et profiter de

ce qui lui a été accordé pour maximiser son débit. Si jamais au prochain intervalle T, il se révèle que cette source n'en a pas profité, ACQ interprête cela comme un non besoin de cette source pour la bande passante, il va alors l'accorder à l'autre source. Et ainsi de suite.
Par conséquent, T doit être suffisamment grand pour permettre aux sources de réagir à la bande passante qui leur ont été accordées. Ce qui signifie que T doit être de l'ordre de quelques délai aller-retour, 2 au minimum puisque le protocole TCP adapte sa fenêtre de congestion tous les deux RTTs (en présence de l'option *Delayed Ack*).
D'un autre coté, si l'intervalle de rafraichissement devient trop grand, le système mettra beaucoup de temps pour se stabiliser et donner l'allocation optimale de la bande passante. Ceci empêchera aussi le système de réagir rapidement à tout changement dans les paramètres du trafic.

3.2.1.2 Valeur de γ

Concernant la valeur de γ, ce facteur décide de la quantité par laquelle les débits sont mis à jour à chaque intervalle T. Si nous attribuons une valeur élevée à γ, nous allons vite basculer vers un état instable du système. Si, au contraire, nous donnons à γ une petite valeur, le système converge trop lentement. γ est la variable qui doit aider le système à éviter de grandes oscillations autour des valeurs optimales des poids des classes et à converger vers un état stable en un minimum d'intervalle T.
Selon l'équation (3.6), l'unité de γ est $Kbit/s^2$.

3.3 Validation des performances de ACQ

Nous avons dans les sections précédentes proposé un algorithme d'ordonnancement des paquets dans un environnement bidirectionnel passant par un lien asymétrique. Cet algorithme s'appuie sur deux classes, une pour les paquets de données et une pour les paquets d'acquittements et propose d'adapter les poids des classes en fonction du trafic qui passe.

Dans la section suivante nous reportons les résultats des simulations que nous avons menées afin de valider ACQ. Nous avons utilisé NS (Network Simulator) dans sa deuxième version.

3.3.1 Topologie des simulations

Les simulations présentées dans cette section obéissent au scénario représenté par la figure 3.6. N connexions TCP sont lancées aléatoirement entre l'instant 0 et 5 ms de chaque coté du lien asymétrique. Toutes les connexions sont dans un premier temps des applications TCP infinies de type FTP transmettant indéfiniment, et ayant Reno comme version de TCP. Nous présentons dans le chapitre 5 des simulations mettant en évidence l'impact de ACQ sur des applications de courtes durée. La taille des paquets

FIG. 3.6 – *Topologie des simulations*

est fixées à 1500 octets pour les paquets de données et 40 octets pour les paquets d'acquittement. Tous les routeurs sont pourvus de l'ordonnanceur FIFO, sauf le routeur à l'entrée du lien asymétrique qui est pourvu de l'ordonnanceur ACQ.

3.3.2 Comportement de ACQ

Commençons d'abord par regarder de plus près le fonctionnement de ACQ en tant qu'ordonnanceur. Étant donné qu'il implique deux paramètres, à savoir T intervalle de rafraichissement et γ, nous commençons par voir les variations de ces paramètres afin de déduire une stratégie pour le choix des valeurs.

3.3.2.1 Stabilité de ACQ avec les valeurs de T et γ

Comme spécifié dans la section 3.2, il est important de choisir les valeurs des paramètres T et γ qui permettent à ACQ de fournir au lien asymétrique son utilisation maximale. Nous avons fait varier les valeurs des paramètres puis calculé les valeurs des fonctions d'utilité.

La Figure 3.7 représente les variations de la fonction d'utilité en fonction des valeurs de T et de γ. Les simulations ont impliqué 10 connexions de chaque coté du lien asymétrique et ont duré 10000 secondes. Elles ont en outre été répétées dix fois, donnant à chaque fois des résultats inclus dans un intervalle de confiance de l'ordre de 0.95. La figure représente en fait une moyenne entre les dix résultats.
On remarque que pour chaque valeur de T il existe une valeur maximale pour l'utilité correspondant à une valeur optimale pour γ. On remarque aussi que plus T est grand, plus la valeur optimale de γ est petite. La figure montre trois paires de valeurs donnant des utilités maximales:
- T = 20 secondes et $\gamma = 50\ Kbps^2$,
- T = 10 secondes et $\gamma = 100\ Kbps^2$, et
- T = 5 secondes et $\gamma = 150\ Kbps^2$.

FIG. 3.7 – *Variation de la fonction d'utilité avec T et γ*

Par contre pour T égal à 2 secondes, les valeurs atteintes par la fonction d'utilité sont trop basses et donc non acceptables. Ceci est normal: les délais moyens d'aller retour comprenant temps de propagation et délais d'attente dans les routeurs de ce système sont aux alentours de 1.2 secondes. Donc T égal à 2 secondes n'est pas un intervalle suffisant pour permettre au système de réagir à des changements de partage de bande passante. En fait, étant donnée une valeur pour T, γ doit aider le système à être réactif à un changement dans le partage de la bande passante. Et donc si la valeur de T est trop petite telle que 2 secondes pour cette topologie, γ doit être grand afin de permettre au système de réagir. γ doit donc être plus grande que 250 $Kbit/s^2$.
Dans le cas contraire où T a une valeur suffisamment grande pour permettre au système de converger vers le meilleur schéma de partage de la bande passante, γ ne doit pas être trop grande pour empêcher le système d'osciller autour de cet optimum. γ joue le rôle de modérateur afin d'amener le système le plus subtilement possible à l'optimal.
D'un autre coté, dès que γ atteint de trop petites valeurs de l'ordre de 10 $Kbit/s^2$, les valeurs des fonctions d'utilité deviennent trop petites. En effet, dans ce cas, le système converge trop lentement, et ce quelque soit la valeur de T.

Afin de comprendre mieux les paramètres T et γ, nous avons fait aussi des simulations avec des scénarios différents et représentant des situations extrêmes:

– **Changement dans le nombre de connexions**: nous avons d'abord considéré une seule connexion dans chaque sens, puis 20 connexions.

  La Figure 3.8 montre les résultats des simulations avec une seule connexion dans chaque direction du lien asymétrique. Cette figure montre que dans le cas d'une seule connexion de chaque coté, l'utilité globale est faible. En effet, l'utilité est représentée par la somme des débits du trafic ascendant sur le lien ascendant

FIG. 3.8 – *Nombre de connexions impliquées = 1*

FIG. 3.9 – *Nombre de connexions = 20*

et du trafic descendant sur le lien descendant. En ayant une seule connexion descendante, celle-ci va subir les effets de l'asymétrie du lien et donc son débit sur le lien descendant sera médiocre. Etant donné que c'est le seul trafic dont nous disposons, le trafic lui même va être faible dans ce cas. Si la valeur de T est petite, par exemple 2 secondes dans la figure 3.8, le système n'étant pas assez chargé, il ne parvient pas à réagir comme il faudrait. Il faut alors avoir de grandes valeurs de T, telles que 20 secondes afin de permettre au trafic et plus précisément aux sources de réagir et à ACQ de donner la meilleure utilisation du lien. Dans le cas d'une charge faible du réseau, et une fois fixée une grande valeur de T, nous devons aussi fixer une grande valeur de γ afin d'aider encore plus le système.

Dans le cas où le système est suffisamment chargé avec 20 connexions de chaque côté du lien, dès que T atteint 5 secondes nous obtenons de bonnes valeurs pour l'utilité. En effet, dans ce cas, il est plus facile de charger le lien descendant même

si les connexions seront confrontées à l'asymétrie et ACQ (qui se base justement sur les débits ascendants et descendants) est capable de bien partager la bande passante même pour de petites valeurs de γ. Comme le montre la Figure 3.9, l'utilité atteint rapidement des valeurs aux alentours de 1.6 dès que T est égal à 5 secondes.

FIG. 3.10 – *RTT moyen = 300ms*

FIG. 3.11 – *RTT moyen = 800ms*

– **Changement dans la valeur du délai aller-retour**: nous avons refait les simulations avec un délai aller-retour faible de 300ms puis un délai un peu plus élevé de 800ms. Les résultats sont reportés dans les Figures 3.10 et Figure 3.11. On remarque que plus le RTT est grand, plus T doit être grand afin, encore une fois, de permettre au système de réagir. Pour le premier cas où le RTT est petit, pour un intervalle de rafraichissement de 20 secondes il a été possible d'atteindre de bons résultats, à savoir 1.7. En revanche, si nous augmentons la valeur du RTT, l'utilité diminue et il faudra alors augmenter la valeur de T sinon le système ne parvient pas à réagir et converger vers le meilleur schéma de partage de la bande passante.

Nous définissons la notion de "réseau suffisamment chargé" comme suit:
Par suffisamment chargé, nous entendons que la charge du réseau permet, dans un cas d'ordonnancement FIFO, d'avoir un des débits, ascendant ou descendant (ou les deux), certes médiocre mais non nul. L'ordonnanceur ACQ est dans ce cas capable d'améliorer les performances de ce trafic tout en protégeant le trafic opposé.
En revanche, le réseau est considéré "non suffisamment chargé" si, toujours dans le cas d'un ordonnancement FIFO, un des trafics, ascendant ou descendant, est pratiquement nul, à cause des effets de l'asymétrie. ACQ a donc la capacité de permettre au trafic nul de circuler et d'avoir des débits satisfaisants tout en protégeant le trafic opposé.

A chaque scénario correspond des valeurs optimales qu'il faut fixer selon la stratégie suivante:

- Si le réseau n'est pas suffisamment chargé T et γ doivent avoir des valeurs larges
- si le réseau est suffisamment chargé:
  - Si le RTT est grand alors T doit être grand. Dans ce cas γ doit être petite afin d'éviter d'éventuelles oscillations du système.
  - si le RTT est petit alors T peut être petit mais c'est à γ d'être grand pour mener le système le plus rapidement possible au partage de files d'attente qui fournit l'utilisation optimale du lien.

En suivant ces principes simples, il est possible pour chaque cas de figure de trouver les paramètres optimaux pour l'ordonnanceur ACQ. Dans la suite de cette section, nous rapportons les résultats du comportement de ACQ en terme de partage de la bande passante et d'utilisation moyenne du lien asymétrique. Selon notre stratégie, nous avons pris T égal à 10 secondes et γ égale à 50 $Kbps^2$ pendant le reste de cette section.

3.3.2.2 Partage de la bande passante avec ACQ

Le principe de notre ordonnanceur ACQ est de trouver rapidement le partage de la bande passante idéal entre les deux classes qui permettra à l'utilisateur d'être satisfait des performances des trafics bidirectionnels sur le réseau d'accès asymétrique. Ceci en supposant que les trafics qui parcourent le lien asymetrique sont infinis en durée.
Dans la figure 3.12, nous avons la représentation des poids des deux classes en fonction du temps de la simulation. A noter que nous avons laissé les simulations durer 10000 secondes afin de laisser aux différentes sources tout le temps qu'ils leur faut pour réguler leur trafic avec ACQ. Cependant, nous remarquons que le système atteint rapidement un état stable.
Comme le montre la figure 3.12, la classe des paquets de données acquièrt 80% de la bande passante laissant donc les 20% restants pour les paquets d'acquittement. Dans le cas de notre scénario, cette répartition des poids des classes semble être optimale pour la fonction d'utilité choisie puisque le système se stabilise.

FIG. 3.12 – *Partage de la bande passante*

ACQ face à différents scénarios de début et de fin des trafics Nous nous intéressons dans ce paragraphe au comportement de notre ordonnanceur ACQ dans le cas où les instants de début et de fin des différentes connexions impliquées changeraient, c'est à dire dans le cas où les connexions dans les deux directions commencent et finissent à n'importe quel moment. Bien évidemment, les cas de figures possibles sont infinis mais nous testons dans ce qui suit trois scénarios de simulation très communs et qui peuvent donc avoir lieu dans l'Internet. Notre but est de vérifier que ACQ parvient toujours à être flexible dans son allocation de la bande passante afin de maximiser l'utilisation du lien asymétrique.

1. *Scénario 1:* Les deux trafics T_r et T_f commencent à l'instant 0 (ou, plus précisémment, aléatoirement entre 0 et 5 secondes. T_r s'achève au bout de 4000 secondes. T_f, en revanche, continue de transmettre jusqu'à la fin de la simulation (10000 secondes). Comme le montre la figure 3.13, lorsque le trafic T_r s'arrête, ACQ parvient à utiliser la bande passante nouvellement disponible en accordant la part auparavant allouée au trafic T_r au trafic T_f. Cette flexibilité de ACQ représente un avantage non négligeable puisque ACQ tente toujours de ne pas laisser une part de la bande passante du lien retour non utilisée tout en maximisant l'utilisation de la bande passante disponible sur le lien aller.

2. *Scénario 2:* Les deux trafics T_r et T_f commencent à l'instant 0. T_r s'achève au bout de 4000 secondes puis recommence à transmettre à la 6000 ème seconde. T_f, en revanche, ne change pas du tout et continue de transmettre jusqu'à la fin de la simulation (10000 secondes). Ce scénario nous permet en fait de tester la stabilité de ACQ dans le cas où les trafics ne démarrent ni ne finissent en même temps. Le changement concernant le trafic T_r est clairement représenté dans la figure 3.14,

FIG. 3.13 – *Partage de la bande passante; T_r s'achève à t=4000s*

FIG. 3.14 – *Partage de la bande passante; T_r s'achève à t=4000s puis redémarre à t=6000s*

où, à la 4000 ème seconde, la totalité de la bande passante est allouée au trafic T_f, et ce jusqu'au lancement à nouveau du trafic T_r à la 6000 ème seconde. En effet, dès que T_r redémarre, ACQ parvient à réadapter l'allocation de la bande passnte. Ce scénario met en évidence la facilité d'adaptation de l'ordonnanceur ACQ aux conditions du trafic.

3. *Scénario 3:* Le nombre des connexions impliquées dans le trafic T_f augmente fortement pendant le déroulement de la simulation. À la 4000 ème seconde exactement, nous impliquons quarante nouvelles connexions appartenant au trafic T_f c'est à dire ayant leurs paquets de données sur le chemin aller.
   La variation de l'allocation de la bande passante avec ACQ dans ce scénario est rapportée dans la figure 3.15. Nous voyons clairement que ACQ est tout à fait stable dans le cas où un grand nombre de connexions appartenant au trafic T_f est

FIG. 3.15 – *Partage de la bande passante dans le cas d'un changement brusque dans l'un des trafics*

rajouté pendant la simulation. Nous pouvons donc dire qu'en cas de changement brusque dans l'un ou l'autre des trafic, ACQ reste stable dans son allocation de la bande passante.

3.3.3 Comparaison de ACQ avec d'autres ordonnanceurs

Dans cette section, nous allons comparer l'utilisation moyenne du lien mesurée en utilisant ACQ avec quatre autres schémas d'ordonnancements:

- Ordonnanceur simple FIFO, premier arrivé premier servi
- FIFO avec le mécanisme AF/AR de filtrage des acquittements et reconstruction après passage du lien ascendant.
- Ordonnancement CBQ basé sur deux classes, une pour les acquittements et une pour les paquets de données. La différence entre CBQ et ACQ est que CBQ n'adapte pas les poids des classes en fonction du trafic.
- CBQ avec AF/AR dans la file d'attente des paquets d'acquittements

Pour ACQ aussi nous avons d'abord effectué des simulations sans utiliser AF/AR dans la file d'attente des acquittements puis refait les simulations avec AF/AR. A chaque simulation, nous avons dix connexions de chaque côté du lien asymétrique. Nous avons fixé les valeurs de T et γ à 10 secondes et 50 $Kbps^2$ respectivement.
Nous séparons les simulations en deux classes: sans utilisation de filtrage et reconstruction des acquittements et avec utilisation de AF/AR. Dans chacun des deux prochains paragraphes, nous rapportons les résultats des simulations relativement aux deux parties de la simulation.

Application d'ACQ sans utilisation de AF/AR Pour commencer, nous avons effectué des simulations impliquant CBQ, FIFO, et ACQ dans le routeur à l'entrée du lien asymétrique. Nous n'avons pas utilisé de filtrage ni de reconstruction d'acquittements. Concernant CBQ, nous avons impliqué deux variantes à savoir un partage de la bande passante à 50-50 entre les deux classes et un partage plus avantageux pour les acquittements qui reçoivent 70 pour cent de la bande passante. Les résultats sont

FIG. 3.16 – *Résultats des simulations sans utilisation de AF/AR*

représentés dans la figure 3.16. Sans l'utilisation de AF/AR, ACQ atteint une utilisation moyenne du lien asymétrique de 1.62, dépassant ainsi l'utilisation atteinte par CBQ 50-50 égale à 1.4 et celle atteinte par CBQ 30-70 égale à 1.2. En effet le partage mesuré par ACQ se trouve être le meilleur dans ce scénario pour atteindre une utilisation optimale de la ressource du réseau. Il est à noter que l'intervalle de confiance de ces simulations est très petit, en effet nous avons 95 pour cent de chance que la valeur de l'utilité soit dans l'intervalle [1.621481 - 0.005186, 1.621481 + 0.005186]. Nous pouvons donc affirmer que la valeur de l'utilité avec ACQ atteint 1.62.

Application de ACQ avec utilisation de AF/AR Afin de mieux affronter l'asymétrie, nous ajoutons le filtrage et la reconstruction des acquittements dans la file d'attente de la classe des acquittements. Le filtrage se fait avant de passer dans le lien asymétrique, et la reconstruction se fait donc à la sortie du lien tel qu'expliqué dans la section 1.3.2. Intuitivement, nous pensons qu'appliquer AF/AR pour la classe des acquittements va aider les connexions descendantes à améliorer leurs performances sans pour autant affecter les connexions ascendantes qui sont protégées par ACQ.

Les résultats des simulations sont dans la figure 3.17. Tout d'abord, nous remarquons que la convergence de ACQ vers la valeur finale de la fonction d'utilité est plus lente avec l'utilisation de AF/AR. Ceci est parfaitement compréhensible: en effet ACQ se base

FIG. 3.17 – *Résultats des simulations avec utilisation de AF/AR*

sur les débits des trafics descendants et ascendants pour calculer à chaque intervalle de temps T les nouveaux poids à accorder aux deux classes tel que spécifié dans l'équation (3.6). Or le filtrage puis la reconstruction des acquittements dans le lien ascendant fausse le trafic descendant. En d'autre termes, AF/AR "trichent" en quelque sorte sur le vrai flux des acquittements et faussent par conséquent le trafic descendant. Tout ceci implique qu'il faut un peu plus de temps à ACQ pour détecter le vrai trafic descendant et donc la convergence vers la meilleure répartition des poids de classes demandera un plus grand nombre d'intervalles T.
D'autre part, nous remarquons qu'à la fin de la connexion, la fonction d'utilité atteint 1.72 et est donc plus élevée pour ACQ avec AF/AR que sans AF/AR. Il y a donc un compromis à faire entre atteindre vite une satisfaisante répartition des poids des classes ou atteindre en un peu plus de temps une répartition des poids très satisfaisante. En fait, selon le type de trafic, nous pouvons dire s'il est conseillé ou pas d'utiliser AF/AR. Pour des connexions très longues, l'utilisation de AF/AR est recommandée, pour des connexions un peu moins longues l'utilisation de AF/AR n'est pas recommandée.
La figure 3.17 montre clairement que ACQ donne la meilleure utilisation du lien, à savoir 1.72 comparé à 1.5 pour CBQ et 1.2 pour FIFO avec AF/AR. Nous recommandons donc d'utiliser ACQ en tant qu'ordonnanceur pour un trafic bidirectionnel passant par un lien asymétrique afin de maximiser l'utilisation des liens.

3.3.4 Récapitulation des résultats de simulations

ACQ est placé à l'entrée d'un lien asymétrique dans le but d'en optimiser l'utilisation globale. En effet, en adaptant les poids des deux classes qui le caractérisent en fonction du trafic qui passe, ACQ vise à maximiser en même temps l'utilisation du lien aller et du lien retour et permettre ainsi de satisfaire l'utilisateur de trafic bidirectionnel

passant par un tel lien.
Les simulations menées dans cette section montrent bien que ACQ atteint ses objectifs, qu'il soit ou pas utilisé avec les mécanismes de filtrage et reconstruction des paquets d'acquittements dans la classe des acquittements. En effet, partant d'un choix adéquat des paramètres, ACQ se stabilise en un temps très court et trouve rapidement le partage de la bande passante du lien retour qui permet d'avoir la meilleure utilisation globale du lien asymétrique. La mise en place des paramètres doit, quant à elle, respecter une stratégie assez simple et applicable dans toutes les situations, permettant ainsi de toujours trouver les paramètres optimaux.

ACQ est donc un ordonnanceur qui a des performances meilleures que l'ordonnanceur FIFO appliqué avec les mécanismes de filtrage et reconstruction des paquets d'acquittement proposés jusque là pour améliorer l'utilisation d'un lien asymétrique sur lequel transite un trafic bidirectionnel. Grâce à son adaptation au trafic et à sa flexibilité concernant les poids des classes, ses performances sont aussi supérieures à celle de CBQ. Nous avons d'autre part testé la robustesse et la stabilité de ACQ en cas de changement dans les paramètres du réseau, les résultats très satisfaisants sont explicités dans le chapitre 5.

3.4 Conclusion

Il est de plus en plus fréquent d'avoir le cas d'un utilisateur téléchargeant un gros fichier de l'Internet à travers un lien haut débit tout en envoyant un message par mail. Dans ce chapitre, nous avons proposé ACQ qui est un ordonnanceur à placer à l'entrée du lien à bas débit et qui a pour seul objectif de maximiser l'utilisation du lien asymétrique considéré comme importante ressource du réseau.
ACQ se base sur la philosophie des classes de trafics et sur le mécanisme CBQ (*Class Based Queuing*) mais a l'avantage d'adapter les poids des classes en fonction du trafic, évitant ainsi des sous-utilisations du lien en réservant de la bande passante non utilisée. Nous avons décrit ACQ ainsi que les paramètres qui le caractérisent et qui, configurés convenablement, permettent d'optimiser les performances de ACQ. Nous avons par ailleurs donné une stratégie efficace qui permet de configurer justement les paramètres. Nous avons mené des simulations qui ont montré clairement l'augmentation des performances obtenues avec ACQ comparé au simple FIFO et à CBQ avec les mécanismes conçus pour remédier aux problème d'asymétrie de bande passante.
Il est à préciser que dans ACQ présenté dans ce chapitre, les deux actions que mène l'utilisateur ont la même importance pour lui, il est donc hors de question pour cet utilisateur de privilégier un des trafics aux dépends de l'autre. Ceci dit, l'utilisateur peut aussi vouloir donner une plus haute priorité à un trafic circulant dans un sens par rapport à l'autre. Par exemple, on pourrait imaginer facilement un utilisateur payant plus cher tout trafic descendant, provenant des liens satellites. La fonction d'utilité dans ce cas donnerait plus la priorité au trafic ascendant. Même pour de tels cas relatifs à d'autres fonctions d'utilité, ACQ est aussi applicable. En effet, nous pouvons tout à fait envisager plusieurs "extensions" de ACQ. ACQ adapte les poids des classes

de données et d'acquittements selon l'équation (3.1). Cette équation a été élaborée en ayant pour objectif de maximiser l'utilisation du lien asymétrique, c'est à dire en maximisant l'utilisation du lien aller et celle du lien retour. Seulement il peut être envisageable d'imaginer d'autres fonctions d'utilité qui répondraient à d'autres exigences des utilisateurs. Pour ce faire, il suffirait de changer la définition de la fonction U, ce qui changerait donc simplement l'équation (3.1) et permettrait à ACQ d'ordonnancer les paquets de données et les paquets d'acquittements en fonction de cette nouvelle fonction d'utilité. Cette souplesse de l'ordonnanceur ACQ est très avantageuse.

Avec un ordonnancement ACQ, tous les trafics descendants sont aggrégés et traités tel un seul trafic, idem pour le trafic ascendant. Cette approche est utilisée par des opérateurs qui connaissent parfaitement l'ensemble du trafic assez régulier qui circulent sur leur réseau. Elle a pour avantage de nécessiter moins de manipulations puisque les traitements effectués se feront sur un ensemble de paquets plutôt que sur tout paquet qui transite par le lien. C'est donc une approche plus simple mais elle présente pour inconvénient majeur d'être moins précise et pas très spécifique. Dans le prochain chapitre, nous présentons un autre ordonnanceur qui se base sur une approche à fine granularité, c'est à dire qui traîte chaque paquet individuellement, et qui peut être préféré à ACQ de la part d'opérateurs connaissant moins bien l'ensemble de leur trafic appelé à changer souvent. Cette nouvelle approche nécessite plus de manipulations et est donc plus complexe. En revanche, elle fournit des résultats plus précis. Le mécanisme se basant sur cette approche (VAQ) est présenté et discuté dans le chapitre suivant.

Chapitre 4

Maximiser l'utilisation des ressources en suivant une approche à fine granularité

Dans le chapitre 3, nous avons présenté ACQ qui est un ordonnanceur se basant sur deux classes distinctes de trafic pour ordonnancer les paquets de données et les paquets d'acquittement et atteindre une maximisation de l'utilité établie auparavant. ACQ agrège tous les trafics descendants d'un coté et tous les trafics ascendants d'un autre côté en deux classes, et, en adaptant les poids respectifs des deux classes, il parvient à améliorer l'utilisation globale du lien asymétrique. Nous avons aussi mis en place une stratégie pour fixer les paramètres nécessaires au bon fonctionnement de ACQ. En effet, ACQ se base sur le calcul de débit moyen sur une large période de temps sur les liens ascendants et descendants. Pour ce faire, il est fondamental de fixer une valeur optimale de l'intervalle de rafraichissement T pendant lequel se fait ce calcul de débit moyen. Cette valeur optimale (et celle du facteur d'amortissement γ) varie avec les paramètres du réseau. L'utilisation de ACQ est donc souhaitable par des opérateurs qui connaissent parfaitement leur trafic. Pour de tels opérateurs, ACQ parvient à fournir une utilté maximale, et, avantage qui n'est pas des moindres: la définition de cette utilité peut être aussi variée que ces opérateurs le souhaitent.

Dans ce chapitre, nous présentons un autre mécanisme pour ordonnancer les paquets de données et d'acquittement dans un réseau asymétrique. Notre but est toujours le même à savoir maximiser une utilité préalablement définie. Il s'agit donc de maximiser l'utilisation du lien asymétrique. Cependant, nous voulons atteindre cet objectif en évitant d'avoir à fixer des paramètres tels que T et γ. Le calcul de l'utilité ne devrait donc plus dépendre du temps de rafraichissment et du calcul d'un débit moyen. La solution que nous avons choisie est de calculer l'utilité en fonction d'un autre élément qui traduit un trafic TCP, à savoir les acquittements. Nous agissons directement sur chaque paquet d'acquittement pour en extraire les informations qui nous indiquent l'état du réseau en terme de débit sur les liens ascendants et descendants. Nous appelons le

mécanisme présenté dans ce chapitre VAQ pour *Virtual Ack-based Queuing*, il se base sur la notion de "taille virtuelle" des paquets d'acquittements. Nous allons expliciter en détail cette notion plus loin dans ce chapitre, mais nous pouvons déjà dire que le principe est de ne plus considérer la taille d'un paquet d'acquittement mais de prendre en compte le nombre total d'octets qu'il acquitte.
VAQ peut être préférée à ACQ par des opérateurs ne pouvant pas fixer un profil constant de leur trafic puisque le fait d'agir sur chaque paquet individuellement rendra ce deuxième ordonnanceur plus flexible à des changements dans les caractéristiques du trafic. Cependant, il est important de signaler que, à la différence de ACQ, VAQ tel présenté dans ce chapitre, répond uniquement à la fonction d'utilité que nous avons présentée dans le chapitre précédent (c.f. 3.2). En effet, dans ce cas, VAQ ne nécessite aucune mesure de trafic et ne dépend d'aucun paramètres, ce qui lui confère un avantage majeur par rapport à ACQ. Afin de l'appliquer à d'autres fonctions d'utilité et lui permettre de couvrir un plus large spectre des fonctions d'utilité traduisant les critères de qualité de service, des mesures de trafic deviennet nécessaires.

Avant de présenter VAQ, nous faisons un rappel des mécanismes de contrôle de trafic de l'Internet. Nous décrivons ensuite le mécanisme VAQ et décrivons les résultats de simulations visant à tester ses performances dans un environnement bidirectionnel asymétrique.

4.1 Notion de contrôle de trafic

La notion de contrôle de trafic a été introduite dans l'Internet afin d'assurer une optimisation des performances des nouvelles applications (c.f. 1.1.1) et de maximiser la satisfaction des utilisateurs de telles applications (c.f. 3.1.1). Ce rappel nous aidera à présenter les principes de VAQ et à justifier notre choix concernant sa politique d'ordonnancement.

Nous avons vu dans le chapitre 2 que le protocole TCP est muni d'un ensemble d'algorithmes permettant de faire un contrôle de congestion du réseau et un contrôle d'erreurs. Le contrôle de congestion désigne l'ensemble des actions permettant d'éviter l'encombrement du réseau et de minimiser l'impact d'une congestion du réseau, le contrôle et recouvrement d'erreurs désigne l'ensemble d'actions permettant de rétablir au plus vite en cas de congestion l'état "normal" non congestionné du réseau. Seulement, avec l'apparition des nouvelles applications, il y a eu besoin de "prévenir" la congestion considérée fatale pour certaines de ces applications et d'adapter les débits de transmission aux capacités des ressources du réseau. Le **contrôle de trafic** désigne l'ensemble des actions permettant de prévenir la congestion et d'adapter le trafic au réseau.

Le contrôle de trafic peut se faire selon deux approches:

– Mécanismes à l'intérieur du réseau: constitués de mécanismes de gestion active des files d'attente tels que RED (c.f. 2.1.3.2) qui visent à garder des tailles moyennes

de files d'attente réduites, et des politiques d'ordonnancement dans les routeurs internes tels que WFQ (*Weighted Fair Queuing* [13].

– Mécansimes à l'entrée du réseau:

  – Contrôle d'accès (CAC "*Connection Admission Control*"): c'est l'algorithme déroulé à l'entrée du réseau ([77]) pour décider si une nouvelle connexion peut être acceptée ou pas. Une nouvelle connexion est acceptée seulement si le réseau peut garantir la qualité de service qu'elle exige.

  – Lissage du trafic (*Traffic Shaping*): caractérisé par un ensemble d'algorithmes visant à imposer au trafic d'être conforme à un modèle prélablement défini. Pour ce faire, le trafic passe d'abord par un "descripteur de trafic" qui forme avec les paramètres de QoS négociés avec le réseau le "contrat de trafic". Les algorithmes de lissage de trafic vérifient que le trafic est toujours conforme au contrat.

Nous décrivons plus en détail dans ce qui suit le principe de lissage de trafic.

4.1.1 Lissage du trafic: *Shaping*

Nous pouvons identifier trois importants critères pour effectuer la fonction de lissage:

– Le débit moyen ou débit soutenu (*Average Rate*): représente une estimation du débit de la source à long terme. Le réseau peut vouloir limiter à long terme le débit moyen d'injection de paquets dans le réseau par tel ou tel flux [1].

– Le débit maximal ou débit crête (*Peak Rate*): ce critère limite le nombre maximal de paquets qui peuvent être transmis sur le réseau sur une courte période de temps.

– La taille de la rafale (*Burst Size*): Le réseau peut aussi vouloir limiter le nombre maximal de paquets transmis dans le réseau sur une très courte période de temps (à limiter les rafales de paquets). En faisant tendre cette periode de temps vers zéro, la taille des rafales limite le nombre de paquets qui peuvent être transmis simultanément sur le réseau.

Seau percé "*Leaky Bucket*" (LB) et Seau à jetons "*Token Bucket*"

Il s'agit d'une abstraction des critères de lissage cités plus haut. On dispose d'un buffer illustré par un seau (*bucket*) avec une taille limitée (b) et un taux de sortie constant. Il existe plusieurs variantes de l'algorithme LB [46]. Un des modèles de LB les plus couramment utilisés est le *Token Bucket* (TB) décrit dans [77]. Dans TB, des jetons sont générés avec un taux de r jetons par secondes. Si le seau est rempli avec

1. Un flux qui a un débit moyen limité à 100 paquets par secondes subit plus de contraintes qu'un autre flux limité à 6000 paquets par minute, même si les deux ont le même débit moyen sur une période de temps assez longue

moins de b jetons quand un jeton est généré, le nouveau jeton est rajouté au seau, sinon le nouveau jeton est ignoré. Avant qu'un paquet ne soit transmis dans le réseau, un jeton du seau est enlevé. Si le seau est vide, le paquet reste en attente jusqu'à la génération d'un nouveau jeton. Puisque le nombre maximum de jetons est b, la taille maximale des rafales est donc égale à b. Aussi, puisque le taux de génération des jetons est égal à r, le nombre maximum de paquets qui peuvent entrer dans le réseau pour tout intervalle de temps t est donc $rt + b$. Notons que la notion de buffer introduite pas l'algorithme LB est virtuelle et ne signifie pas que le service est à débit constant ou que les paquets qui arrivent quand le buffer virtuel n'est pas vide subiront un délai supplémentaire.

4.1.2 Conclusion

Dans cette section, nous avons fait une étude des mécanismes de contrôle de trafic se bansant sur le lissage visant à assurer certains critères de la qualité de service qu'un utilisateur exige pour certaines applications. Parmi ces critères nous pouvons citer un débit moyen maximum pour les connexions. C'est en effet ce critère qui nous interesse dans ce chapitre. Nous rappelons que nous voulons concevoir un ordonnanceur capable d'assurer une utilisation maximale d'un lien asymétrique traversé par un trafic bidirectionnel. Pour ce faire, nous optons pour la notion de lissage de trafic. Cependant, les algorithmes existants ne nous satisfont pas à cause de leur caractère non conservatif *non Work-Conserving* puisque le lien peut être inactif malgré la présence de paquets en attente d'être transmis. Ceci est contraire à une maximisation de l'utilisation du lien. Nous concevons par conséquent VAQ un ordonnanceur qui sépare à l'entrée du lien bas débit le trafic de données et le trafic d'acquittements en deux files d'attente, et effectue un lissage sur le trafic de données selon un algorithme *Work Conserving* que nous détaillons dans la section 4.2.

4.2 Le modèle VAQ

Dans cette section, nous présentons VAQ, un ordonnanceur entre paquets de données et paquets d'acquittement dans un environnement asymétrique. Notre objectif est toujours de maximiser l'utilisation des ressources du réseau dans un environnement bidirectionnel asymétrique et de satisfaire ainsi l'utilisateur. Satisfaire l'utilisateur revient à ordonnancer intelligemment les paquets de données et les paquets d'acquittement afin d'utiliser au mieux les ressources de cet environnement, à savoir le lien asymétrique. Nous reprenons la figure 4.1 illustrant un trafic bidirectionnel passant par un lien asymétrique. VAQ est placé à l'entrée du lien congestionné afin de gérer au mieux le partage de la bande passante entre le trafic dans le sens aller et le trafic dans le sens retour et assurer une utilisation maximale des liens. Pour ce faire, nous utilisons un mécanisme de crédit alloué à la classe des paquets de données. En fait, VAQ fait un contrôle supplémentaire du débit des paquets de données. A priori, ce contrôle est d'une part bénéfique pour les paquets d'acquittements puisque les paquets de données sont volumineux et donc risquent de saturer le lien bas débit et d'autre part il affecte

FIG. 4.1 – *Architecture d'un trafic bidirectionnel*

à moindre mesure les paquets de données puisqu'ils sont capables de "répondre" à une éventuelle perte de paquets et d'agir autant que possible afin d'y remédier. En effet, au niveau du routeur, un contrôle de débit effectué sur les paquets de données aurait moins d'effets que sur les paquets d'acquittements.

VAQ associe à chaque type de trafic ascendant et descendant une file d'attente. Il attribue un crédit à la file d'attente des paquets de données et se propose d'adapter ce crédit en fonction du trafic qui passe en suivant une politique d'ordonnancement *Weighted Fair Queuing*.

4.2.1 Philosophie de VAQ: Émuler un lien virtuel

VAQ se base sur le principe de conservation de travail lors d'un partage de lien entre plusieurs paquets. VAQ émule un lien virtuel qui est en fait la concaténation du lien aller et du lien retour et a pour objectif d'ordonnancer les paquets sur ce lien virtuel de façon à toujours conserver le travail et toujours avoir un paquet à transmettre, c'est le principe du *Work Conserving* où un serveur n'est jamais inactif tant qu'il y a un paquet à envoyer.
VAQ agit paquet par paquet, sans agrégation de trafic. Chaque paquet est traité individuellement. Afin d'émuler le lien virtuel, il faut reconsidérer les paquets qui peuvent transiter par ce lien virtuel et qui émulent le trafic ascendant d'une part et le trafic descendant d'autre part, nous énumérons donc:

1. D'une part les paquets de données appartenant au trafic ascendant. Ils émulent donc naturellement le trafic ascendant.

2. Et d'autre part, les paquets d'acquittement appartenant au trafic descendant et qui serviront à émuler le trafic descendant. Pour ce faire, nous devons les redimensionner. Nous considérons alors que ces paquets ont une taille "virtuelle"

égale aux nombres d'octets réellement acquittés par chaque paquet d'acquittement. D'où le nom de cet ordonnanceur VAQ : ordonnancement basé sur la taille "virtuelle" des acquittements (*Virtual Ack-based Queuing*).

FIG. 4.2 – *Schématisation de la philosophie de VAQ*

La "philosophie" de VAQ est schématisée dans la figure 4.2. Maximiser l'utilisation du lien asymétrique revient donc à maximiser l'utilisation du lien virtuel ainsi configuré de manière à suivre une politique conservative *work-conserving*.

En fait, nous pouvons présenter autrement la philosophie de VAQ. Avec une telle reconfiguration du lien virtuel, on aura à ordonnancer entre paquets de tailles équivalentes. En fait il est judicieux de considèrer le nombre d'octets que chaque acquittement acquitte puisque en notant B ce nombre, faire passer un paquet d'acquittement relatif à B octets veut dire que dans l'intervalle aller-retour précédent, il y a eu B octets de paquets appartenant au trafic descendant qui sont passés sur le lien descendant et qui ont donc consommé une portion de la bande passante descendante. De façon plus triviale, nous pouvons dire que VAQ tente alors de rétablir l'équilibre entre trafic descendant et trafic ascendant en allouant un équivalent de bande passante ascendante au trafic ascendant. La politique d'ordonnancement est expliquée plus clairement dans la section suivante.

4.2.2 Politique d'ordonnancement de VAQ

Nous conservons les mêmes notations que celles citées dans le chapitre 2, à savoir T_f pour le trafic descendant, T_r pour le trafic ascendant et C_f et C_r pour la bande passante disponible respectivement sur le chemin descendant et ascendant. Nous ajoutons dans ce chapitre les notations suivantes:

- FileDonnees: file d'attente associée aux paquets de données, tout paquet de données qui arrive à l'entrée du lien lent sera stocké dans cette file d'attente si le lien en question est déjà occupé.
- TeteFileDonnees: le premier paquet de données dans la file d'attente, VAQ dispose en effet d'un pointeur sur le premier paquet prêt à être transmis sur le lien lent. Si la file d'attente associée aux paquets de données est vide, ce pointeur sera nul.
- FileAcks: file d'attente associée aux paquets d'acquittement, cette file d'attente sera consultée par VAQ au cas où la file d'attente réservée aux paquets de données est vide ou que la valeur du crédit ne permet pas l'envoi d'un paquet de données.
- TeteFileAcks: le premier paquet d'acquittement dans la file d'attente, idem que pour TeteQueueDatas.
- NbrOctetsAcquittes(p): nombre total d'octets aquittés par un paquet d'aquittement donné p, ce nombre est calculé par VAQ pour chaque paquet d'acquittement par une opération assez simple qu'on décrira un peu plus tard dans ce chapitre.
- Taille(p): taille en octets d'un paquet donné p, information contenue dans l'entête du paquet de données.

Lorsqu'un paquet d'acquittement est transmis sur le lien ascendant, VAQ regarde le nombre total d'octets que ce paquet acquitte. En effet, si un acquittement acquitte plus d'une taille de segment MSS ceci veut dire que le lien de retour a été utilisé par plus d'un paquet de données au détriment des acquittements du trafic ascendant. Il est donc souhaitable maintenant de céder un peu plus de la bande passante disponible sur le lien ascendant pour le trafic ascendant afin d'atteindre une utilisation plus grande des liens par les paquets de données.
L'utilisation du lien par un paquet d'acquittement est en fait l'utilisation du lien descendant. Elle correspond donc au rapport du nombre total d'octets acquittés par cet acquittement divisé par la capacité totale du lien descendant. Dans notre cas c'est donc:

$$\frac{NbrOctetsAcquittes}{C_f}$$

Cette utilisation dans le lien descendant doit donc être émulée par les paquets d'acquittement passant sur le lien ascendant. VAQ redimensionne alors les paquets d'acquittement par leur équivalent sur le lien ascendant correspondant à cette utilisation. En d'autres termes, cette utilisation dans le lien descendant implique à fortiori une perte d'utilisation dans le lien ascendant puisque les paquets d'acquittements et les paquets

de données se partagent le lien lent. Afin de compenser cette perte, il faudra que des données ascendantes aient une utilisation du lien ascendant correspondant à la valeur suivante:

$$\frac{NbrOctetsAcquittes * C_r}{C_f}$$

C'est la taille virtuelle qui doit être attribuée aux paquets d'acquittement, et c'est donc aussi la valeur du crédit qui sera accordé au trafic ascendant T_r pour tout passage d'un acquittement sur le lien ascendant afin d'assurer une maximisation de l'utilisation globale.
D'autre part lorsqu'un paquet de données passe sur le lien ascendant, VAQ estime que ce paquet a utilisé une partie du crédit alloué à la file d'attente des paquets de données. Il met à jour alors le crédit en le diminuant de la valeur correspondant à la taille en octets de ce paquet de données.

La politique d'ordonnancement de VAQ obéit à un algorithme décrit dans ce qui suit :

```
Si ( FileDonnees est vide ) Alors
      p = PrendrePaquet(FileAcks)
      Si ( p est non NULL ) Alors
         crédit = crédit + NbrOctetsAcquittes(p)*C_r / C_f
Sinon Si (FileAcks est vide) Alors
           p = PrendrePaquet(FileDonnees)
           crédit = crédit - Taille(p)
         Sinon Si (crédit >= Taille(TeteFileDonnees)) Alors
                    p = PrendrePaquet(FileDonnees)
                    crédit = crédit - Taille(p)
               Sinon
                    p = PrendrePaquet(FileAcks)
                    crédit = crédit + NbrOctetsAcquittes(p)*C_r / C_f
```

On a donc à chaque passage de paquet sur le lien ascendant une mise à jour du crédit alloué à la classe des paquets de données. En effet, de par leur grande taille et leur réponse aux notifications de congestion, les paquets de données sont ici considérés non prioritaires et c'est pourquoi leur passage par le lien ascendant est régi par un crédit. D'autre part, les paquets d'acquittement ne doivent pas monopoliser le lien non plus. En fait, en accordant un crédit à la file d'attente des paquets de données, nous défavorisons cette file d'attente, puisqu'elle peut contenir des paquets à transmettre mais ne pas en avoir le droit à cause d'un crédit insuffisant. Cependant, comme le montre l'algorithme, cette file d'attente est consultée avant celle des paquets d'acquittements, et ces derniers ne commencent à transmettre que si la file des données est vide ou que leur crédit est insuffisant. Ceci permet donc aux acquittements de ne pas monopoliser le lien. La "priorité" donnée aux paquets d'acquittement avec le crédit pour la file de données est par conséquent réduite.

4.2.3 Détails d'implémentation de VAQ

FIG. 4.3 – *Format d'un segment TCP*

Dans cette section nous expliquons plus en détail le mode exact d'action de l'ordonnanceur sur un segment TCP, dont le format est représenté dans la figure 4.3. Le champs *Numéro de séquence* définit le numéro de séquence du premier bit de données du segment. Le champs *Numéro d'acquittement* contient la valeur du prochain numéro de séquence la source s'attend à recevoir, si le bit de contrôle de l'acquittement est mis à 1. (voir RFC 793 pour plus de détails). VAQ traîte les paquets différemment selon qu'il s'agit d'un paquet de données ou un paquet d'acquittement.

- Si le paquet manipulé par VAQ est un paquet de données, VAQ regarde la valeur du crédit avant de décider si oui ou non il peut transmettre ce paquet. Si la valeur du crédit permet l'envoi du paquet, c'est à dire si le crédit est supérieur à la taille du paquet de donnée en tête de la file d'attente, VAQ doit regarder la taille en octets du paquet pour mettre à jour le crédit et respecter l'algorithme d'ordonnancement. Cette taille est calculée avec les valeurs des bits du *"checksum"* comme le montre la figure 4.3. La manipulation d'un paquet de données se traduit donc par un accès au champs *"checksum"* dans l'entête du paquet de données suivie d'un simple calcul permettant de fournie l'information de la taille du paquet en question.
- Si le paquet est un acquittement, VAQ a besoin d'avoir une information fondamentale qui est le nombre total d'octets de données réellement acquitté par ce paquet d'acquittement. Pour ce faire, VAQ calcule la différence entre la dernière séquence d'acquittement reçue et la séquence attendue par le récepteur. La deuxième information à savoir la séquence attendue par le récepteur est contenue dans l'entête du

paquet d'acquittement (comme le montre la figure 4.3). Elle est donc directement accessible. La première information, à savoir la dernière séquence d'acquittement reçu, peut être accessible aussi dans l'entête du paquet d'acquittement en lui affectant un des champs disponible. Ceci nécessite une manipulation très simple du paquet d'acquittement. Une fois le nombre total d'octets acquittés calculé, VAQ est en position de mettre à jour le crédit. La séquence attendue par le récepteur peut aussi être stockée dans le routeur, auquel cas nous éviterions de manipuler les paquets d'acquittements. En effet, la capacité des routeurs de l'Internet sont de plus en plus large et il n'est par conséquent pas réducteur pour VAQ de considérer du stockage par flux dans les routeurs (*Per-Flow State*).

Suite à ces explications dans les manipulations des deux types de paquets par l'ordonnanceur VAQ, son implémentation devient assez simple puisque VAQ accède aux informations dont il a besoin directement sur TCP.

4.2.4 Paramètres du modèle VAQ

Dans cette section nous précisons les valeurs des paramètres et autres métriques qui peuvent avoir une influence sur les résultats donnés par VAQ. Nous appellons paramètres les constantes à fixer au début du lancement de l'ordonnanceurs. Nous regardons aussi tout autre facteur extérieur qui pourrait avoir un impact, positif ou négatif, sur le fonctionnement de VAQ.

FIG. 4.4 – *Architecture de VAQ*

Dans la figure 4.4 nous illustrons le fonctionnement de VAQ. Il se place à l'entrée du lien bas débit et met à jour la valeur du crédit comme il est spécifié dans 4.2.2.

4.2.4.1 Paramètres du modèle

VAQ est un ordonnanceur qui met en place deux files d'attente virtuelles, une pour les paquets de données et une pour les acquittements. Il accorde un crédit à la file des données et adapte ce crédit en fonction du trafic qui passe afin d'avoir toujours une utilisation maximale du lien asymétrique, comme illustré dans la figure 4.4.
Les seuls paramètres nécessaires à VAQ sont les tailles maximales des files d'attentes virtuelles et les tailles des paquets constituant le trafic. Intuitivement c'est la taille maximale de la file d'attente des acquittements qui aura le plus d'importance sur le comportement de VAQ. En effet, les paquets de données sont réactifs à des notifications de congestion de la part du réseau. Par conséquent, même s'ils sont confrontés à des délais importants d'attente dans les routeurs, ils sont capables d'y remédier "tous seuls". Par contre, les acquittements ne doivent pas être trop retardés sinon le problème de compression des acquittements sera encore présent et perturbera encore considérablement le fonctionnement de TCP et ceci aura pour conséquence de diminuer l'utilisation du lien.
La taille des données et des acquittements est un autre paramètre qui peut avoir un impact sur le fonctionnement de VAQ. En effet, le crédit de la file des données est mis à jour proportionnellement à la taille des paquets de données. Seulement, nous avons effectué des simulations dans la section 4.3.3 qui prouvent que les tailles des paquets n'influent pas sur les résultats fournis par VAQ.

4.2.4.2 Autre facteurs extérieurs

Notre objectif dans ce chapitre est d'avoir un mécanisme d'ordonnancement entre paquets de données et paquets d'acquittement qui soit le plus indépendant possible d'autres paramétres ou facteurs extérieurs. Nous avons vu dans la section 4.2.2 que VAQ ne demande pas la mise en place de paramètres, ce qui en fait un mécanisme intéressant et prometteur. Cependant nous nous sommes demandé si quelques parametres extérieur peuvent influer le bon déroulement de VAQ. Dans le chapitre suivant (chapitre 5) , nous effectuons des simulations permettant de répondre à cette question.

4.3 Validation des performances de VAQ

Dans cette section, nous rapportons les résultats des simulations effectuées afin de tester l'ordonnanceur VAQ. Notre objectif est toujours d'atteindre la satisfaction de l'utilisateur, satisfaction traduite par une utilisation maximale du lien asymétrique.

4.3.1 Topologie des simulations

Nous reprenons la topologie du chapitre 2 (Figure 3.6). La taille maximale de la file d'attente virtuelle des acquittements est de 10 et celle des données est de 100. Nous avons d'autre part des paquets de taille 40 octets pour les acquittement et 1500 octets pour les paquets de données. La topologie est représentée dans la figure 4.5.

FIG. 4.5 – *Topologie des simulations*

4.3.2 Partage de la bande passante

Dans cette section, nous regardons de plus près le partage de la bande passante entre les paquets de données et les paquets d'acquittement obtenu en ayant VAQ comme ordonnanceur au niveau du routeur. Ce partage va nous permettre d'avoir une idée sur l'équité de VAQ.

FIG. 4.6 – *Partage de la bande passante avec VAQ*

Comme le montre la Figure 4.6, VAQ donne aux alentours de 80 pour cent de la bande passante au traffic T_r (*Reverse Traffic*) représenté sur le lien par les paquets de données et 20 pour cent pour le trafic T_f (*Forward Traffic*) représenté par les paquets d'acquittement. Cette distribution de la bande passante entre les différents paquets permet d'avoir une utilisation maximale du lien. De plus, VAQ y parvient après un intervalle de temps très court équivalent à seulement quelques secondes.

Dans la section 3.3.2.2, nous obtenons approximativement la même distribution de la bande passante avec ACQ, la différence est que VAQ n'a pas besoin de mettre en place des paramètres tels que T et γ. Cette difference fait de VAQ un ordonnanceur bien plus intéressant et plus facile à mettre en œuvre puisque ne nécessitant pas de mise en place ni de mise à jour de paramètres.

4.3.3 Variation de l'utilisation du lien en utilisant VAQ

Nous mesurons dans cette série de simulations les variations de l'utilisation globale du lien asymétrique, égale à:

$$\text{Utilisation Globale} = \frac{DebitMoyenT_f}{C_f} + \frac{DebitMoyenT_r}{C_r}$$

Nous reprenons les résultats obtenus avec ACQ, FIFO et CBQ rapportés dans la figure 3.17 et nous y ajoutons les résulalts de VAQ. Nous avons ainsi directement une comparaison entre les différents ordonnanceurs.

FIG. 4.7 – *Variation de la fonction d'utilité*

Les résultats des simulations sont reportés dans la figure 4.7, où sont représentées les variations de la fonction d'utilité moyenne en ayant respectivement dans le rôle de l'ordonnanceur entre les paquets de données et les paquets d'acquittement VAQ, ACQ, ACQ avec AF/AR, FIFO avec AF/AR puis CBQ avec AF/AR. Ces simulations ont été répétées plusieurs fois donnant un intervalle de confiance très intéressant de l'ordre de 0.95.

La figure 4.7 montre clairement que la courbe la plus haute qui traduit l'utilisation maximale du lien correspond à celle de VAQ. En effet, VAQ parvient à atteindre une utilité maximale aux alentours de 1.75. De plus, le système atteint cette valeur en un

temps très court. L'utilisation du lien atteinte avec VAQ dépasse toutes les autres, ce qui prouve que VAQ est l'ordonnanceur qui fournit l'utilité maximale dans le cas de trafic bidirectionnel passant par un lien asymétrique. En adaptant intelligemment le crédit de la file d'attente allouée aux paquets de données, VAQ parvient à fournir rapidement la meilleure utilisation du lien.

Nous avons présenté le modèle VAQ dans les sections précédentes, ainsi que les paramètres dont il dépend. Nous avons dit que les seuls paramètres sont:

- Les tailles des files d'attente relatives respectivement aux paquets de données et aux paquets d'acquittement
- et les tailles des paquets de données et d'acquittements.

Nous nous devons de voir les variations des performances de VAQ en fonction de ces paramètres. Dans la suite de cette section nous étudions les variations de l'utilisation globale du lien en fonction de la taille de la file d'attente virtuelle des paquets de données (qu'on note *tailleFileData*), en fonction de la taille de la file d'attente virtuelle des paquets d'acquittements (*tailleFileAck*), en fonction de la taille des paquets de données (*taillePaquetData*) et enfin en fonction de la taille des paquets d'acquittement (*taillePaquetAck*).

4.3.3.1 Impact des tailles des files d'attente

Nous regardons d'abord l'impact des tailles des files d'attente des données et des acquittements sur le fonctionnement de VAQ. Afin de parcourir un large spectre des situations possibles, nous varions ces valeurs entre 10, 50 et 100 afin d'être le plus réaliste possible. Nous avons donc le cas de petits buffers, le cas de buffers moyens et le cas de grands buffers. Toutes les combinaisons ont été simulées en utilisant le simulateur de réseau NS afin de montrer les variations de l'utilité moyenne. Rappelons encore que notre fonction d'utilité représente l'utilisation du lien asymétrique $\Longleftrightarrow$

$$\text{Fonction d'utilité} = \frac{DebitMoyenTraficForward}{CapaciteLienForward} + \frac{DebitMoyenTraficReverse}{CapaciteLienReverse}$$

TAB. 4.1 – *Stabilité de VAQ face à la taille des files d'attente*

Le tableau 4.1 reporte les résultats de ces simulations. Ces résultats sont très clairs et montrent que l'impact tailleFileData et de tailleFileAck est négligeable, ainsi que

nous l'avions prévu dans la section précédente. Le tableau montre que varier la taille maximale de la file d'attente de données n'influe aucunement l'utilité. D'autre part, varier la taille maximale de la file d'attente d'acquittement change le résultat de la fonction d'utilité atteinte, ce changement n'est certes pas très important, mais il rejoint notre raisonnement de la section 4.2.4.1.
Nous remarquons que la meilleure utilité moyenne est atteinte avec 10 paquets et 100 paquets respectivement pour tailleFileData et pour tailleFileAck, à savoir 1.75. Nous recommandons donc de ne pas utiliser de très larges files d'attente pour les paquets d'acquittement.

4.3.3.2 Impact des tailles des paquets de données et d'acquittements

Dans l'opération de mise à jour du crédit de la file d'attente virtuelle des données, la taille de paquets de données apparaît comme une variable qui pourrait influer sur le fonctionnement de l'ordonnanceur VAQ. C'est pourquoi nous effectuons ici des simulations mettant en évidence la stabilité de VAQ dans le cas où les paquets qu'il manipule sont différents en terme de taille. Nous avons utilisé des valeurs réalistes des différents cas qu'on peut rencontrer dans l'Internet, 500 octets, 1000 octets et 1500 octets pour les paquets de données et 20 octets, 40 octets et 100 octets pour les paquets d'acquittements.

TAB. 4.2 – *Stabilité de VAQ face à la taille des paquets*

Les résultats sont présentés dans le tableau 4.2. Nous remarquons que la taille des paquets n'a pratiquement aucun impact sur le fonctionnement de VAQ. En effet, l'ordonnancement parvient à réguler le trafic quel que soit la taille de paquets.

4.3.4 Conclusions sur les résultats des simulations

VAQ se base sur un ordonnancement en files d'attente séparées entre deux files d'attente, une pour les paquets de données et une pour les paquets d'acquittement. Il utilise la notion de taille virtuelle concernant les paquets d'acquittement. VAQ met à jour au fur et à mesure un crédit alloué à la file d'attente des données de manière à réguler un trafic bidirectionnel passant par un lien asymétrique et atteindre une utilisation maximale du lien. Les simulations rapportées dans cette section montrent clairement l'avantage d'utiliser VAQ en tant qu'ordonnanceur puisque la valeur de l'utilité moyenne telle que décrite dans 3.1.1 est maximale avec VAQ. De plus, le partage "idéal" de la bande passante entre trafic ascendant et trafic descendant atteint rapidement un état

stable permettant d'affirmer que VAQ parvient à trouver le partage optimal entre les paquets de données et les paquets d'acquittements. D'autres simulations testant la robustesse de l'ordonnanceur VAQ en cas de changements dans les paramètres du réseau sont présentées dans le chapitre 5.

4.4 Conclusion

De plus en plus de mécanismes visant à améliorer les performances d'Internet se basent sur des approches à fine granularité, c'est à dire des approches qui traitent les paquets individuellement et qui font le moins possible appel à des agrégats de trafic. Dans ce chapitre nous avons présenté VAQ, un ordonnanceur entre paquets de données et paquets d'acquittements circulant sur un lien asymétrique. VAQ manipule chaque paquet qui passe sur le lien à bas débit différemment selon que c'est un paquet de donnés ou un paquet d'acquittement et se base sur une taille virtuelle des paquets d'acquittements pour réguler le trafic. Par taille virtuelle des acquittements nous entendons le nombre total d'octets de données réellement acquittés par le paquets. VAQ atteint alors un équilibre entre les allocations de la bande passante lui permettant d'atteindre une utilisation maximale du lien asymétrique et de satisfaire l'utilisateur.

ACQ présenté dans le chapitre 3 et VAQ présenté dans ce chapitre sont deux ordonnanceurs visant à atteindre une maximisation du lien asymétrique et satisfaire ainsi d'une part l'utilisateur d'un trafic bidirectionnel passant par de tels liens qui voit ses débits maximaux et d'autre part de satisfaire les opérateurs ou fournisseurs des services du réseau qui voient leurs liens utilisés de façon optimale. Seulement, avant de parler de déploiement de ces ordonnanceurs, il est important de pouvoir répondre aux deux questions suivantes:

1. Tout d'abord, l'utilisation de ACQ ou de VAQ perturbe-t-elle d'autres critères de qualité de service tels que le délai de transmission?
   En effet, ACQ et VAQ sont appliqués au niveau du routeur d'accès au lien bas débit, ils agissent afin de maximiser une utilité que nous avons définie comme étant la maximisation des débits du trafic ascendant et du trafic descendant. L'objectif a certes été atteint par ACQ et VAQ mais il est important maintenant de voir si pour ce faire le délai a été affecté et jusqu'à quel point.

2. La deuxième question est: ACQ et VAQ fournissent-ils une utilité maximale en cas d'un changement dans les paramètres du trafic ou du réseau?
   En effet, s'assurer que ACQ et VAQ sont résistants à des changements dans la topologie du réseau est fondamental afin de généraliser ces ordonnanceurs. Appliqués dans des topologies avec, par exemple, un nombre différents de connexions impliquées, un degré de symétrie plus important ou un taux d'erreur élevé, ACQ et VAQ fournissent des performances différentes de celles obtenues dans les simulations des chapitres 3 et 4. Sont-elles toujours optimales?

Dans le chapitre suivant, nous menons une large série de simulations qui répondent à ces deux questions.

Chapitre 5

Interaction entre les ordonnanceurs proposés et les conditions du réseau

Dans les chapitres 3 et 4, nous avons proposé deux ordonnaceurs, ACQ et VAQ, avec pour objectif de fournir une utilisation maximale d'un lien asymétrique traversé par un trafic bidirectionnel.

Dans ce chapitre, nous analysons dans une première partie l'interaction entre nos ordonnanceurs et les conditions variables du réseau. Nous regardons dans un premier temps l'impact de leur utilisation sur d'autres critères de qualité de service. En effet, bien que notre objectif de départ est de maximiser l'utilisation du lien asymétrique, il est important que cette maximisation ne se fasse pas aux dépens d'autres métriques de la qualité de service. Nous considérons le délai des trafics et nous rapportons l'impact de l'utilisation de ACQ et VAQ sur les délais de transmission des paquets des connexions. Nous montrons que selon que la priorité de l'utilisateur est plus pour le trafic sur le chemin aller ou pour le trafic sur le chemin retour, il est judicieux d'utiliser ACQ ou VAQ, respectivement.

Dans la deuxième partie de ce chapitre (section 5.2), nous étudions la robustesse de ACQ et VAQ avec différent paramètres du réseau. Nous envisageons les cas suivants:

1. Le nombre de connexions impliquées dans les trafics varie entre une et plusieurs connexions pour chaque trafic. Dans nos simulations, pour valider les performances des ordonnanceurs (3.3 et 4.3), nous avions considéré dix connexions formant le trafic ascendant et descendant. Nous étendons ici nos simulations afin de voir l'impact du nombre de connexions sur les performances de ACQ et VAQ.

2. La capacité du lien bas débit varie traduisant une variation du degré d'asymétrie du lien. En effet nos ordonnanceurs travaillent à optimiser l'utilisation du lien asymétrique et ont montré leur efficacité dans le cas où le degré d'asymétrie tel que défini dans 2. est aux alentours de 17, correspondant à une capacité descendante

et ascendante égales respectivement à 1 Mbit/s et 56 Kbit/s. Nous analysons ici l'impact de la variation du degré d'asymétrie (en variant la capacité du lien ascendant) sur les performances de ACQ et VAQ.

3. Le réseau subit des erreurs autres que les erreurs de saturations de tampons des files d'attente des routeurs. En effet, un lien tel qu'un lien satellite subit des erreurs à cause de phénomènes d'atténuations de signal dus aux longues distances et de la transmission radio. Ces erreurs sont traduites par des pertes aléatoires de paquets de données ou d'acquittements. Nous avons donc étudié l'impact de telles pertes sur le fonctionnement de ACQ et VAQ.

4. Les trafics passent par plusieurs liens asymétriques. Il est en effet tout à fait normal qu'un trafic bidirectionnel emprunte plus d'un lien asymétrique. Nous plaçons donc ACQ et VAQ à l'entrée de chaque lien bas débit et mesurons les utilisations moyennes de tous les liens asymétriques.

5. D'autres types de trafic traversent le lien bas débit. TCP constitue 95% du trafic circulant sur Internet. Cependant, le nombre d'applications dont le trafic suit un modéle exponentiel (source On/Off) ou à débit constant (CBR) augmente. Nous étudions pour cela les performances de ACQ et VAQ avec de tels trafics.

Les résultats des simulations de toutes ces situations sont rapportés dans la partie 2 de ce chapitre.

5.1 Partie 1: Impact de ACQ et VAQ sur les délais de transmission des trafics

Nous avons vu que ACQ et VAQ donnent le meilleur résultat en terme d'utilisation moyenne du lien asymétrique, dépassant FIFO et CBQ. ACQ et VAQ présentent donc un interêt certain, et cet interêt serait d'autant plus complet si on pouvait affirmer que leur application ne nuit pas à d'autres critères de qualité de service et de satisfaction de l'utilisateur. Dans cette section, nous reportons quelques résultats de simulations mettant en évidence l'impact de l'utilisation de ACQ et de VAQ sur les délais de transmission.

Les paquets qui traversent un routeur ayant ACQ ou VAQ comme ordonnanceur vont certainement subir un délai supplémentaire à ceux qui passent par un simple FIFO vu qu'un traitement plus compexe leur est appliqué. Il s'agit en effet du délai de traitement des algorithmes d'ordonnancement de ACQ et de VAQ. La question ici est de savoir si ACQ et VAQ portent préjudice en terme de délai de transmission de paquets ou s'ils parviennent malgré tout à optimiser les débits des trafics sans affecter le délai de transmission.

Nous reprenons la topologie de simulations représentée dans 3.6. Nous comparons le délai donné par VAQ avec celui donné par ACQ, par ACQ avec les mécanismes de

filtrage et de reconstruction des acquittements AF/AR avec FIFO et AF/AR et enfin avec CBQ et AF/AR. L'idéal est d'avoir le meilleur compromis entre délai d'aller retour moyen du trafic ascendant et délai aller retour moyen du trafic descendant. En utilisant ACQ ou VAQ, les paquets d'acquittements appartenant au trafic descendant subissent un traitement particulier dans le sens retour alors que pour le trafic ascendant ce sont les paquets de données qui subissent ce traitement.
Nous avons toujours les mêmes notations, à savoir T_r pour le trafic ascendant et T_f pour le trafic descendant.

5.1.1 VAQ favorable au délai du trafic ascendant

FIG. 5.1 – *Variation du délai aller-retour du trafic ascendant*

Les résultats des simulations sont rapportés dans la Figure 5.1.
En appliquant un traitement *class-based*, un ordonnanceur fait passer les paquets par plusieurs étapes à savoir classification, shaping, etc. Si de plus ce sont les paquets de données qui subissent toutes ces étapes, le trafic est d'autant plus retardé. La figure 5.1 montre bien le retard que subi le trafic T_r lorsque ACQ ou CBQ jouent le rôle de l'ordonnanceur.
Les résultats montrent d'autre part que c'est VAQ qui donne le délai moyen le plus bas concernant le trafic T_r. Le gain en utilisation de lien et efficacité de transmission compense alors le délai de traitement des paquets de données. Ceci représente un fort bon point pour VAQ.

5.1.2 Délai du trafic descendant

Les résultats des simulations sont reportés dans la Figure 5.2. Concernant le trafic T_f, nous remarquons déjà que les délais moyens sont en général plus bas que pour le

FIG. 5.2 – *Variation du délai aller-retour du trafic descendant*

trafic T_r, ce qui est tout à fait attendu vu que ce sont ici les paquets d'acquittement qui subissent le retard et que TCP possède les moyens d'y remédier et de diminuer ce retard.
ACQ avec AF/AR fournit le délai moyen le plus intéressant pour T_f tandis que ACQ sans AF/AR fournit le délai moyen le plus élevé. L'explication est évidente puisque AF/AR est justement appliqué pour aider les paquets d'acquittement à ne pas attendre longtemps dans les files d'attente des routeurs. Il est donc recommandé d'appliquer ACQ avec AF/AR si le trafic T_f est continu et que le critère de délai pour T_f est plus important que la rapidité de convergence (cf Figure 3.17).
VAQ ne donne pas le délai moyen de T_f le plus bas mais ce délai est tout de même acceptable et plus intéressant que celui donné par FIFO .

5.1.3 Récapitulatif: délai moyen de trafic bidirectionnels passant par un lien asymétrique

Le choix de l'ordonnanceur relativement au délai moyen des connexions revient à utiliser l'ordonnanceur qui répond le mieux aux critères des utilisateurs. Plusieurs situations sont possibles, et à chacune des situations il est recommandé d'utiliser l'ordonnanceur qui répond le mieux. En voici quelques unes:

- Si le délai du trafic descendant a considérablement plus d'importance pour l'utilisateur que le délai du trafic ascendant, il serait préférable d'utiliser ACQ avec AF/AR ou CBQ avec AF/AR. Nous notons cependant que l'utilisation du lien est plus importante avec ACQ, il est donc recommandé d'appliquer ACQ plutôt que CBQ afin de permettre d'avoir une combinaison entre délai réduit et utilisation maximale du lien.

– Si le délai du trafic ascendant est plus important pour l'utilisateur, il faut absolument utiliser VAQ et ne surtout pas utiliser CBQ puisque CBQ donne le délai moyen le plus élevé pour le trafic ascendant.

– S'il s'agit d'obtenir un compromis entre les délais dans le cas où l'utilisateur accorde une priorité équivalente au trafic ascendant et au trafic descendant, nous recommandons fortement d'utiliser VAQ. En effet VAQ parvient d'une part incontestablement à optimiser l'utilisation du lien (cf Figure 4.7) et d'autre part à obtenir le meilleur compromis entre délai du trafic aller et délai du trafic retour.

5.2 Partie 2: Robustesse des mécanismes d'odonnancement

Dans le chapitre précédent, nous avons montré que le mécanisme ACQ provoquait une augmentation de l'utilisation du lien ascendant. Le critère de maximiser l'utilisation des liens est d'ailleurs très réaliste et pratiquement tous les opérateurs et fournisseurs de service exigeraient d'un réseau asymétrique une utilisation du lien congestionné optimale afin de maximiser leur profit.
Dans cette section, nous testons la robustesse de ACQ et de VAQ face à différentes conditions extérieures du réseau. En effet, dans l'Internet, une infinité de cas de figure peut exister. Ces cas diffèrent par les capacités des liens, le nombre de connexions qui traversent ces liens, le type de trafic des connexions, etc. Il est par conséquent important de s'assurer qu'un ordonnanceur puisse s'adapter à de telles topologies.
Nous reprenons la même topologie que celle de la figure 3.6 puis effectuons quelques changements dans la topologie des simulations. Nous montrons que ACQ et VAQ sont capable de s'adapter à differents changements dans le trafic et de se stabiliser rapidement donnant ainsi une valeur très acceptable de la fonction d'utilité. Evidemmnent, la quantité de scénarios qu'on pourrait imaginer est infinie mais nous rapportons ici quelques uns des scénarios les plus réalistes et les plus significatifs dans l'Internet actuel.

5.2.1 Implication d'un grand nombre d'utilisateurs

Dans un premier temps, nous varions le nombre de connexions qui peuvent traverser le lien asymétrique. Nous regardons donc l'influence du nombre de connexions impliquées dans la simulation sur le déroulement de ACQ. Nous commençons par regarder le cas d'un seul utilisateur. Nous avons augmenté ensuite le nombre de connexions jusqu'à atteindre 100 connexions de chaque part du lien asymétrique, c'est à dire jusqu'à atteindre un réseau bien chargé.
Nous avons donc fait des simulations avec (en reprenant les notations de la section 2.2):

– 1 connexion pour T_f et 1 connexion pour T_r

– 5 connexions pour T_f et 5 connexions pour T_r

– 10 connexions pour T_f et 10 connexions pour T_r

– 100 connexions pour T_f et 100 connexions pour T_r

FIG. 5.3 – *ACQ et VAQ sont robustes en cas de changements dans le nombre de connexions dans chaque direction du lien asymétrique impliquées dans les simulations*

Nous calculons dans chaque cas la moyenne d'utilité atteinte. Les résultats de ces simulations sont représentés dans la figure 5.3. Nous rapportons les résultats dans le cas où nous utilisons juste ACQ et dans le cas ou nous rajoutons aussi les mécanismes AF/AR. Nous notons aussi qu'à chaque configuration, nous mettons en place les paramètres de ACQ les plus adéquats selon la stratégie décrite en 2.4.2. Par exemple, dans le cas de réseau faiblement chargé, T est égal à 20 secondes et γ to 400 $kbps^2$.

Dans le cas d'une très faible charge du réseau nous remarquons que l'utilité atteinte est assez faible, et ce, que ce soit avec l'application ou pas de AF/AR. En fait nous nous attendions à ce genre de résultat puisque le réseau n'est pas suffisamment chargé pour pouvoir bien réagir, et ceci quand bien même nous aurions arrangé au mieux les valeurs des paramètres.

Dès que le nombre de connexions dans le réseau dépasse 5 pour chaque type de traffic, nous remarquons que ACQ devient capable de donner une bonne utilité. Et puisque comme cité plus haut, le cas d'un réseau bien chargé est le cas le plus probable, ces simulations sont satisfaisantes. Une remarque tout de même pour le cas d'un nombre de connexions supérieur à 70; nous remarquons que l'utilisation des mécanismes AF/AR diminue la valeur moyenne de l'utilité. Ceci s'explique par le fait qu'augmenter le nombre de connexions implique augmenter les opérations de filtrage et de reconstruction des acquittements. Ceci génère plus de délais dans les routeurs et donc les débits moyens diminuent pour les connexions et l'utilisation des liens aussi. Il est par conséquent préférable de ne pas utiliser AF/AR si le réseau est surchargé, mais d'appliquer ACQ seul afin d'atteindre une moyenne d'utilisation du lien optimale.

Les résultats des différentes simulations sont reportés dans la Figure 5.3. Il est clair que, d'après la figure, VAQ est bien robuste face à une variation dans le nombre de connexions impliquées. En effet aussi bien pour un petit nombre d'utilisateurs que pour un grand nombre d'utilisateurs, la valeur moyenne de la fonction d'utilité obtenue reste invariable égale à 1.75.

Ceci veut dire que même si la charge du réseau est très faible, VAQ est capable de fournir une utilisation maximale des ressources du réseau. Et réciproquement, si le réseau est très chargé, VAQ parvient tout de même à trouver la meilleure répartition de la bande passante entre les différents paquets de données et d'acquittements pour encore satisfaire au maximum les utilisateurs.

5.2.2 Changement du degré d'asymétrie

Dans cette section, nous varions la capacité du lien ascendant, passant ainsi d'une forte asymétrie à un cas de lien symétrique. ACQ et VAQ sont conçus pour ordonnancer les trafics bidirectionnels passant par un lien asymétrique. Il est donc important de savoir si ACQ et VAQ sont robustes face à différents degrés d'asymétrie, en d'autres termes la question que nous nous sommes posée est:

> jusqu'à quel degré d'asymétrie d'un lien ACQ et VAQ sont-ils capables d'adapter les débits d'un trafic bidirectionnel traversant ce lien et de fournir ainsi une utilisation maximale?

La topologie des simulations est similaire à celle représentée dans la figure 4.5. Nous avons toujours 1 Mbit/s pour la capacité du lien descendant C_f et varions la capacité du lien ascendant C_r. Nous avons fait des simulations avec:

- 9.6 Kbit/s comme capacité du lien C_r : ce cas représente le cas "pire" et traduit un accès au réseau via une liaison téléphonique classique à très bas débit.
- 28.8 Kbit/s comme capacité du lien C_r : ceci traduit le cas d'une liaison téléphonique actuelle.
- 56 Kbit/s comme capacité du lien C_r: l'accès au réseau est fait avec un modem avec un débit en sortie de 56 Kbit/s.
- 128 Kbit/s comme capacité du lien C_r: l'accès au réseau est fait avec un modem avec un débit en sortie de 128 Kbit/s.
- 560 Kbit/s comme capacité du lien C_r: l'accès au réseau est fait avec une liaison numéris haut débit en sortie de 560 Kbit/s.
- 1 Mbit/s comme capacité du lien C_r: cas de lien symétrique. Cependant la présence de trafic bidirectionnel signifie que le lien de retour est partagé par les paquets de données et les paquets d'acquittements. On a donc toujours le problème de compression de paquets d'acquittements. De plus, ce problème certe amoindri se trouve maintenant aussi bien sur le lien descendant que sur le lien ascendant.

Concernant ACQ, nous nous attendons à avoir des performances affectées par une très forte asymétrie. En effet, une très forte asymétrie implique un trafic des acquittements sur le chemin ascendant pratiquement inexistant. Ceci induit automatiquement un trafic de données sur le chemin descendant perturbé d'abord puis fortement réduit. Et dans ce cas, et puisque ACQ se base sur le débit moyen du trafic de données sur le lien descendant, il faudra donner à ACQ une période de rafraichissement T et une valeur de γ très élevée.

FIG. 5.4 – *ACQ et VAQ sont robustes en cas de une variation de la capacité du lien ascendant*

Les résultats des simulations traduisant les variations de l'utilité moyenne en fonction de la capacité du lien ascendant sont représentés dans la Figure 5.4. Un zoom de la figure concernant les petites valeurs de la capacité du lien ascendant est représenté dans la figure 5.5.

Concernant ACQ, l'évolution de l'utilité en fonction de la capacité du lien ascendant est globalement convexe pour de petites valeurs de C_r. Si la capacité du lien ascendant est faible (pour 9.6, 28.8 et 56 Kbit/s), l'utilisation de AF/AR augmente légèrement l'utilisation moyenne. En effet, dans ce cas le filtrage puis reconstruction des acquittements va aider l'ordonnanceur à atteindre une bonne utilisation du lien. Cependant, comme nous l'avons montré dans 3.3.3, l'utilisation de AF/AR va réduire la vitesse de convergence. Nous remarquons aussi que CBQ+AF/AR donnait une meilleure utilité que ACQ avec ou sans AF/AR. En effet, dans le cas d'une forte asymétrie, l'isolation constante du trafic d'acquittements sur le lien ascendante est souhaitable. En revanche, au delà de 40 Kbit/s environ, les performances de ACQ sont très bonnes. Aussi, pour des asymétries "raisonnables", l'utilisation de AF/AR avec ACQ n'est pas favorable à l'augmentation de l'utilisation du lien.

FIG. 5.5 – *ACQ et VAQ sont robustes en cas de une variation de la capacité du lien ascendant*

Les figures 5.4 et 5.5 représentent aussi le comportement de VAQ face à une variation de la capacité du lien ascendant. Nous remarquons aisément que les résultats de l'utilité obtenue avec VAQ sont très satisfaisants. En effet, l'utilité démarre avec un minimum de 1.05 pour un cas de très forte asymétrie, dépassant tous les autres ordonnanceurs, et ne cesse de croître jusqu'à approcher de très près la valeur maximale de 2.

Si nous reprenons la variable r définie dans le chapitre 2 (section 2.2.1) défini comme suit:

$$\mathrm{r} = \frac{\frac{C_{aller}}{C_{retour}}}{\frac{t_{data}}{t_{ack}}}$$

Nous pouvons résumer les résultats de nos simulations traduisant les performances de nos ordonnanceurs avec différentes valeurs de r comme suit:

- $r < 0.5$: Nous sommes ici dans le cas d'une grande capacité du lien ascendant traduisant donc une faible asymétrie du lien. Nous recommandons dans ce cas l'utilisation de ACQ sans les mécanismes AF/AR afin d'atteindre une utilité avoisinant 1.8. VAQ donne aussi des performances optimales, nous recommandons aussi son application à l'entrée du lien ascendant.

- $0.5 < r < 1$: L'asymétrie est dans ce cas moyenne. ACQ donne des performances légèrement meilleures avec AF/AR (entre 1.24 et 1.62). VAQ quant à lui fournit encore une fois une utilité optimale, de l'ordre 1.7. Nous recommandons donc l'application de VAQ.

– $r > 1$: L'asymétrie est alors très importante. La moyenne de l'utilité est aux alentours de 1.016 si on applique ACQ sans AF/AR, 1.107 si on applique ACQ avec AF/AR et 1.36 si on applique VAQ. A titre de comparaisons FIFO et CBQ donnent resprectivement 1.01 et 1.175. Si l'asymétrie est donc très forte, c'est VAQ que nous recommandons le plus.

5.2.3 Présence d'erreurs dans le réseau

Dans ce chapitre, nous testons la robustesse de nos ordonnanceurs ACQ et VAQ face à des changements des paramètres du réseau. Notre objectif est de nous rapprocher le plus possible des cas réels qui peuvent avoir lieu dans l'Internet actuel et donc d'étudier l'impact et la possibilité d'un éventuel déploiement de nos ordonnanceurs. Il faudrait par conséquent considérer des liens avec un taux d'erreur plus ou moins important. L'étude du comportement de ACQ et VAQ face à des erreurs dans le réseau représente le sujet de cette section.
Nous avons effectué des simulations avec différents cas de figure:

– Présence d'erreurs sur le lien retour T_r: dans ce cas le lien à faible capacité est en plus confronté à des erreurs aléatoires autres que celles causées par des saturations des files d'attente des routeurs. Ce cas n'est pas très fréquent dans la réalité et pourrait représenter des situations où le lien terrestre par lequel l'utilisateur est relié à l'Internet est de faible qualité.

– Présence d'erreurs sur le lien aller T_f: ce lien est à forte capacité, il peut représenter des liens satellites par exemple. Ces liens couvrent généralement de grandes distances, et acheminent des transimission radio. Des pertes aléatoires sont par conséquent tout à fait plausibles et envisageables vu les problèmes d'atténuations de signal.

– Et présence d'erreurs sur les deux liens en même temps: on a ici envisagé le pire des cas.

Dans chaque situation, nous rapportons les variations de l'utilisation moyenne du lien asymétrique. Nous mesurons les erreurs par un taux correspondant au nombre de paquets perdus pour cause d'erreurs sur le lien par rapport au nombre de paquets correctement transmis.

5.2.3.1 ACQ et VAQ face à des erreurs sur le lien retour

Dans un premier temps, nous testons la robustesse des ordonnanceurs ACQ et VAQ dans le cas où le lien à bas débit est confronté à des pertes de paquets dont les raisons sont autres que des saturations des files d'attente. Nous avons effectué des simulations avec différentes valeurs du taux d'erreur et calculé les moyennes des utilités atteintes.

Les résultats des simulations sont rapportés dans la figure 5.6. Les erreurs sont ici simulées sur le lien ascendant c'est à dire le lien où s'effectuent tous les traitements de nos ordonnanceurs. Les erreurs ou pertes de paquets sont appliquées aléatoirement et

FIG. 5.6 – *ACQ et VAQ sont robustes pour des taux d'erreur sur T_r inférieurs à 30%*

nous perdons sur le lien ascendant des paquets de données ou des paquets d'acquittements. Regardons d'un point de vue conceptuel plus en détail l'effet théorique de ces pertes sur nos ordonnanceurs.

Commençons par étudier l'impact que peuvent avoir des erreurs sur le lien ascendant sur le comportement de VAQ. Un paquet de données du trafic T_r est lancé par VAQ sur le lien ascendant après avoir décrémenté la valeur du crédit. Si ce paquet de données est rejeté avant d'arriver à destination à cause d'erreurs provoquées sur le lien ascendant, du coté du récepteur aucun acquittement du trafic T_r ne sera généré et la source doit alors envoyer à nouveau le paquet de données du trafic T_r. Seulement, elle ne pourra le faire selon la philosophie de VAQ que si le crédit a été incrémenté entre temps suite au passage d'un acquittement appartenant au trafic T_f. Dans le cas d'un fort taux d'erreur, le lien descendant est progressivement exploité uniquement par le trafic T_f, le nombre d'octets acquittés par un acquittement de T_f est donc réduit à un et l'incrémentation du crédit se fait donc trop lentement pour permettre aux données de T_r de passer. Dans le cas où le taux d'erreur atteint et dépasse 30% comme le montre la figure 5.6. VAQ est incapable de compenser et de donner de bonnes valeurs de l'utilité.
Si au contraire ce sont les paquets d'acquittements du trafic T_f qui subissent le plus de pertes du lien ascendant, le récepteur se voit contraint d'envoyer de nouveau les paquets de données du trafic T_f encore et encore sur le lien descendant, lien qui sera donc monopolisé par les données de T_f et nous nous retrouverons confrontés aux mêmes conséquences. C'est pourquoi, nous nous attendons à ce qu'un fort taux d'erreur sur le lien ascendant affecte fortement VAQ.

Concernant ACQ, l'impact d'erreurs sur le lien retour est moins important et ceci grâce à la propriété de large granularité de ACQ. En effet ACQ agit sur le trafic en

modifiant le poids des classes constituées par les aggrégats de trafic et en regardant dans le passé l'effet de chaque allocation sur les classes. Si des erreurs surviennent sur le lien ascendant, ACQ est capable de réguler son trafic en fonction de ces erreurs et de donner le maximum d'utilisation du lien. C'est ce que montre la figure 5.6. La figure montre aussi sans surprise que ACQ, sans l'application de AF/AR, résiste mieux aux erreurs, puisque encore une fois, AF/AR est responsable d'erreurs de jugement d'ACQ pour l'état actuel des trafics T_r et T_f. L'utilisation des classes de trafics différentes pour le trafic T_r et T_f permet aussi à CBQ d'être robuste face à la présence d'erreurs sur le lien retour.

Pour un ordonnancement de type FIFO avec les mécanismes AF/AR, en appliquant des erreurs sur le lien ascendant, nous libérons en fait de l'espace sur ce lien congestionné et à faible capacité. Nous réduisons ainsi l'effet du problème de compression des acquittements et nous évitons la cohabitation entre paquets d'acquittements et paquets de données. Nous remarquons donc dans la Figure 5.6 que pour un taux d'erreur de l'ordre de 10%, un simple ordonnancement FIFO avec les mécanismes AF/AR parvient à atteindre une excellente moyenne de fonction d'utilité avoisinant les 1.9. Cependant dès que le taux d'erreur devient plus important, les performances de AF/AR deviennent aussi insuffisantes.

5.2.3.2 ACQ et VAQ face à des erreurs sur le lien aller

Nous nous sommes aussi intéressés au comportement de nos ordonnanceurs dans le cas où c'est le lien à haut débit qui est confronté à des erreurs. Le lien lent est considéré sans pertes autres que celle causées par des saturations de files d'attente.

FIG. 5.7 – *Comportement de ACQ et VAQ face à des erreurs sur T_f*

Avoir des pertes de paquets sur le chemin aller ne devrait pas avoir un impact aussi catastrophique sur VAQ que le cas d'erreurs sur le chemin retour. En fait, le plus

important est de ne pas risquer de perdre des paquets d'acquittements de T_f puisque ce sont ces paquets qui servent à mettre à jour le crédit de la file d'attente des paquets de données. Des pertes de paquets de données appartenant au trafic T_f sur le chemin aller impliquent moins d'acquittements de ce même trafic sur le chemin retour. Ce qui signifie que le lien congestionné le sera moins puisque les acquittements de T_f cèdent leur place aux paquets de données de T_r. VAQ est, dans ce cas, plus favorable aux paquets de données. Si, au contraire, ce sont des paquets d'acquittement appartenant à T_r qui sont perdus sur le lien aller, VAQ sera certes envahi par des paquets de données retransmis de T_r, mais il sera apte à savoir les ordonnancer avec les paquets d'acquittements de T_f, et ce, grâce au crédit alloué à la file d'attente des paquets de données. Les résultats sont reportés dans la figure 5.7. Nous voyons bien que VAQ est effectivement l'ordonnanceur qui résiste le mieux à des erreurs sur le lien descendant.

La robustesse d'ACQ face à des erreurs sur le lien aller est mise en évidence uniquement dans le cas où nous n'utilisons pas les mécanismes AF/AR, (voir figure 5.7). En effet, dans le cas de taux d'erreurs importants sur le lien aller, un nombre important de paquets de données appartenant au trafic T_f sont perdus et necessitent l'envoi de plus en plus de paquets d'acquittement sur le chemin retour. Ce qui implique que les opérations de filtrage et reconstruction d'acquittements seront trop fréquentes. Les délais qu'elles causent ainsi que leur coût de traitement devient important. Dans le cas de ACQ et de CBQ nous avons en plus des délais de classification et des portions de bande passante réservée à certaines classes. La figure 5.7 montre bien que CBQ avec AF/AR donne aussi des performances médiocres. La concaténation de la présence de classes et de la fréquence des mécanismes AF/AR rendent ACQ avec AF/AR incapable de maximiser l'utilisation du lien dans le cas de forts taux d'erreurs sur le lien aller.

5.2.3.3 ACQ et VAQ face à des erreurs sur le lien aller et le lien retour

Cette section représente en quelque sorte une récapitulation des deux précédentes. Nous simulons maintenant des erreurs sur les deux liens, aller et retour. Pour ce faire nous définissons le degré d'erreur du lien asymétrique comme étant la paire:

$$e = (\text{TauxErreurLienRetour}, \text{TauxErreurLienAller})$$

Nous varions e dans l'ensemble suivant: { (1, 40), (1, 4), (10, 10), (20, 5), (40, 1) } afin d'avoir un large spectre des possibilités.
La figure 5.8 montre les valeurs moyennes des utilités pour chaque cas de taux d'erreur sur le lien retour et sur le lien aller. Nous remarquons que globalement ACQ donne toujours des résultats satisfaisants, tant qu'il n'est pas utilisé avec AF/AR. Si, en revanche, nous appliquons ACQ avec AF/AR la présence d'un taux d'erreur sur le lien ascendant ou sur le lien descendant affecte souvent l'utilisation globale du lien.
Concernant VAQ, un grand taux d'erreur sur le lien ascendant implique de mauvaises performances pour l'ordonnanceur. Dans tous les autres cas, VAQ augmente considérablement l'utilisation du lien asymétrique.

FIG. 5.8 – *Comportement de ACQ et VAQ face à des erreurs sur le lien aller et le lien retour*

5.2.4 Passage par plusieurs liens asymétriques

Nous faisons ici subir à ACQ et à VAQ le passage de trafic bidirectionnel par plus d'un lien asymétrique. En effet, nous ne pouvons pas tester les performances des ordonnanceurs dans le cas de trafic bidirectionnel passant par un seul lien asymétrique puisque la probabilité de passer par plus d'un lien est élevée. La nouvelle topologie des

FIG. 5.9 – *Topologie des simulations*

simulations est représentée dans la figure 5.9. Nous avons fait des simulations avec un trafic bidirectionnel de 10 connexions pour T_r et 10 connexions pour T_f. Toutes les connexions démarrent au début du premier lien et ont leur destination après le dernier lien considéré. Les capacités des liens sont 1 Mbps pour tous les liens descendants (aller) et 56 Kbps pour tous les liens ascendants (retour). Nous faisons passer les trafics T_f et T_r par un seul lien, deux liens, trois, cinq et dix liens.

Les résultats des simulations sont reportés dans la figure 5.10, et la conclusion suivante apparait évidente: ACQ n'est pas robuste face à une augmentation du nombre

FIG. 5.10 – *Comportement de ACQ et VAQ face à un passage par plus d'un lien asymétrique*

de liens par lesquels passent le trafic bidirectionnel . En effet, plus le nombre de liens intermédiaires augmente, plus la valeur moyenne de la fonction d'utilité diminue.

Les simulations sont représentées dans la figure 5.10. Encore une fois, VAQ nous satisfait pleinement. En effet le comportement de VAQ n'est perturbé que très légèrement si les trafics T_r et T_f passent par plusieurs liens. La fonction d'utilité est toujours parfaitement intéressante, les liens asymétriques sont utilisés de façon optimale et l'utilisateur est toujours satisfait même si les connexions qu'il engendre passent par dix liens asymétriques.

5.2.5 Impliquer différents types de trafic

Une grande proportion du trafic circulant sur les réseaux de l'Internet est de type TCP. Cependant, d'autres types de trafic voient de plus en plus fréquemment le jour avec la prolifération de nouvelles applications exigentes en terme de débits ou de délais. Parmi ces nouveaux trafics, le trafic à courte durée modéisé par des sources On/Off et le trafic régi par un protocole autre que TCP modélisé par CBR (*Constant Bit Rate*). Nous décrivons un peu plus loins dans cette section plus en détail les caractéristiques des deux trafics. Nous avons testé la robustesse de nos ordonnanceurs dans le cas où ils sont confrontés à ce genre de trafic sur le lien ascendant.

5.2.5.1 T_r est un trafic On/Off à durée de distribution exponentionelle

Jusqu'à présent le modèle de prédilection dans les évaluations d'algorithmes de télécommunication était le trafic uniforme de Poisson. De récentes statistiques de tra-

fic[1] ont montré que certains trafics de données s'écartaient très sensiblement du modèle poissonnien. Des études récentes montrent que de tels trafics peuvent être modélisés par des superpositions de processus On/Off, ou sources interrompues. Une source interrompue se forme de la façon suivante: la source alterne entre les phases actives ("On") et les phases silencieuses ("Off"). Les durées de ces périodes sont aléatoires. Pendant la phase "Off", le processus d'arrivée est gelé et aucun client n'arrive. Une communication téléphonique produit typiquement un tel profil de trafic. En moyenne il y a des périodes de silence pendant 1/3 du temps [39].
Les paramètres d'un tel trafic sont décrits dans ce qui suit:

- L'intervalle de temps pendant lequel la source de ce trafic est en mode "On".
- L'intervalle de temps pendant lequel la source de ce trafic est en mode "Off".
- Le débit moyen de transmission des paquets pendant la phase "On".
- La taille constante moyenne des paquets transmis pendant la phase "On".

Un processus interrompu a la propriété suivante: si on enlève les périodes "Off" et on recolle les périodes "On", on obtient un processus poissonnien. Les sources On/Off servent à modéliser des trafics à courtes durée dont les conséquences pratiques de leur existence dans l'Internet sont essentiellement les dimensionnements des buffers au niveau des routeurs. En effet les niveaux d'occupation des files d'attente suivent dans le cas de trafic On/Off une loi à décroissance polynomiale alors que sous le modèle poissonnien uniforme la décroissance est exponentielle.

Nous simulons ici un tel modèle de trafic en mettant en place un trafic On/Off au dessus du protocole TCP sur le lien retour, faisant office donc de trafic T_r. Nous simulons un trafic On/Off dont les durées des phases On et Off sont prises selon une distribution exponentielle[2]. Nous fixons les paramètres d'entrée à 0.5 secondes également pour la durée de la phase "On" et pour la durée de la phase "Off", 64Kb pour le débit moyen de transmission des paquets pendant la phase "On" et 210 octets pour la taille moyenne des paquets transmsis. Ce trafic circule sur le lien retour en tant que le trafic T_r. Nous laissons le trafic T_f tel quel, c'est à dire un trafic à longue durée TCP afin d'avoir des paquets d'acquittements circulant sur le lien ascendant et voir le fonctionnement de ACQ et VAQ en cas de cohabitation de ces paquets d'acquittements avec des paquets de trafic On/Off à distribution exponentielle.

Dans cette série de simulations, nous avons 10 secondes comme intervalle de rafraichissement pour ACQ. Toutes les 10 secondes, ACQ calcule les débits moyens atteints sur les liens ascendants et descendants. Le trafic On/Off circule sur le lien ascendant avec une fréquence de phase "On" égale à 0.5 secondes.
Concernant ACQ, avoir un trafic TCP de type On/Off avec une durée à distribution

1. Études observées notamment à Bellcore à New Jersey (USA)
2. Certes, en simulant les trafics TCP à courtes durée par des modèles On/Off nous ne représentons pas exactement la réalité mais nous nous y approchons au maximum

FIG. 5.11 – *Comportement de ACQ et VAQ face à un trafic de type On/Off à distribution exponentielle*

exponentielle améliore les performances (1.75). VAQ, quant à lui, subit une légère baisse des performances

Afin de s'assurer que les bons résultats obtenus dans la figure 5.11 ne dépendent pas des paramètres du trafic T_r, nous effectuons dans ce qui suit d'autres simulations en variant les durées des phases "On" et des phases "Off", les débits des connexions ainsi que les tailles des paquets. Nous notons l'expérience précédente Exp1. Voici les caractéristiques des deux autres séries de simulations effectuées:

Exp2 avec une durée de la phase "On" et de la phase "Off" toutes deux égales à 5 secondes, un débit moyen constant de 640 Kb obtenu pendant la phase "On" et enfin une taille constante des paquets transmis égale à 520 octets.

Exp3 avec une durée de la phase "On" et de la phase "Off" toutes deux égales à 20 secondes, un débit moyen constant de 1 Mb obtenu pendant la phase "On" et enfin une taille constante des paquets transmis égale à 1500 octets.

Les variations des résultats obtenus par VAQ et ACQ pour les trois expériences sont reportés dans la figure 5.12. La figure montre clairement la stabilité des résultats des utilisations du lien asymétrique. En effet, changer les paramètres du trafic T_r n'influe ni sur le comportement de VAQ ni sur celui de ACQ. Nous notons tout de même que pour ACQ, l'utilisation moyenne du lien asymétrique est légèrement moins importante pour Exp2 et pour Exp3. En effet la durée de la phase "On" est plus grande pour Exp2 et encore plus grande pour Exp3. Il aurait donc fallu augmenter aussi la période de rafraichissement T de ACQ pour permettre à l'ordonnanceur de réagir et d'être encore plus efficace.

FIG. 5.12 – *Variation des fonctions d'utilité avec ACQ et VAQ face à plusieurs trafic de type On/Off à distribution exponentielle*

Nous pouvons donc affirmer suite à toutes les simulations effectuées dans cette section que nos ordonnanceurs ACQ et VAQ sont aussi efficaces dans le cas où le trafic T_r est constitué d'un trafic On/Off à courte durée, dans nos simulations nous avons modelé de tel trafic par une distribution exponentielle. L'utilisation de ACQ ou de VAQ permet d'utiliser au mieux le lien asymétrique et est donc fortement recommandée dans un tel cas.

5.2.5.2 T_r est un trafic CBR à débit constant

Le trafic CBR ou *Constant Bit Rate* est à débit constant idéalement utilisé afin de modéliser un trafic temps réel (*Streaming*) ou un trafic Audio à débit constant. L'objectif du modèle de trafic CBR est conceptuellement simple - de permettre à une connexion réseau de ressembler le plus possible à une ligne spécialisée ou à une réservation exclusive de câble ou de fibre optique entre la source et le récepteur. Ces avantages qui ne sont pas des moindres en terme de satisfaction de l'utilisateur ont bien évidemment séduit les opérateurs et fournisseurs de service actuels et l'on retrouve souvent du trafic CBR au dessus de UDP. Nous utilisons ici une modélisation d'un tel trafic présente dans NS dont les paramètres:

- Le débit constant de transmission de paquets
- La taille moyenne des paquets, constante aussi
- Le `random` qui indique si oui ou non il faut rajouter du bruit aléatoire pendant les temps programmés de départ (généralement cette valeur est nulle).

– Le nombre maximum de paquet qui doivent être envoyés (par défaut cette valeur vaut 2^28)

Nous reprenons les simulations ayant un trafic CBR au dessus du protocole UDP en tant que trafic T_r circulant donc sur le lien de retour. Nous fixons les paramètres d'entrée à 448 Kbit/s concernant le débit constant de transmission de paquets, 210 octets concernant la taille constante des paquets.

FIG. 5.13 – *Comportement de ACQ et VAQ face à un trafic de type CBR comme* T_r

Le lien aller dans cette configuration est exclusivement reservé au trafic T_f. En effet, T_r est un trafic UDP ne nécessitant donc pas de paquets d'acquittement sur le lien aller. Le nombre de paquets du trafic T_f perdus est donc moins importants que dans le cas où le lien est aussi partagé par des paquets d'acquittement appartenant au trafic T_r. Par conséquent, au niveau du routeur à l'entrée du lien lent, le nombre de paquets d'acquittements acquittant plus d'un paquet est aussi moins important. D'après la philosophie de VAQ, le nombre de paquets de données autorisés à passer va aussi être moins important impliquant donc une moindre utilisation du lien retour.

Les utilités atteintes par une telle configuration sont représentées dans la figure 5.13. La figure montre les résultats atteints en ayant respectivement VAQ comme ordonnanceur dans le routeur, ACQ, ACQ avec les mécanismes AF/AR, FIFO avec AF/AR et enfin CBQ avec AF/AR. Nous remarquons que les performances atteintes en utilisant nos ordonnaceurs, aussi bien ACQ que VAQ sont les meilleures comparées à FIFO et à CBQ. Cependant, dans ce cas de figure, VAQ est moins bon que ACQ. La moyenne de l'utilisation du lien avec VAQ est égale à 1.65 tandis que ACQ fournit une utilisation avoisinant 1.71 s'il est appliqué avec AF/AR et 1.75 sans AF/AR.

Nous notons l'expérience précédente "Exp1" puis, comme dans la section 5.2.5.1, nous varions les paramètres d'entrée afin d'avoir un large spectre des trafics CBR.

Exp2 avec 540Kbit/s comme débit constant du trafic et 520 octets comme taille constante des paquets

Exp3 avec 1Mbit/s comme débit constant et 1500 octets comme taille de paquets

FIG. 5.14 – *Variation des fonctions d'utilité avec ACQ et VAQ face à plusieurs trafic de type CBR*

Les variations des résultats obtenus par VAQ et ACQ pour les trois expériences sont reportés dans la figure 5.14.

5.2.5.3 T_r est un trafic hétérogène

Nous étudions maintenant les performances de ACQ et VAQ dans le cas où le trafic ascendant T_r est en fait un "mélange" de trafics. Nous avons envisagé deux cas:

- Un premier cas où T_r est un mélange de trafic FTP et de trafic On/Off. Le trafic FTP est constitué de 10 connexions à longue durée avec une taille de segments de données égale à 1500 octets et une taille de paquets d'acquittements de 40 octets, le protocole est TCP Reno. Le trafic On/Off est aussi constitué de 10 connexions sur TCP, avec comme paramètres du trafic: 0.5 secondes pour la durée de la phase "On" et pour la durée de la phase "Off", 64Kbit/s pour le débit moyen de transmission des paquets pendant la phase "On" et 210 octets pour la taille moyenne des paquets transmsis.

- Un deuxième cas où T_r est un mélange de trafic FTP et de trafic CBR. Le trafic FTP est aussi constitué de 10 connexions à longue durée avec une taille de

FIG. 5.15 – *T_r est un mélange de TCP et On/Off*

FIG. 5.16 – *T_r est un mélange de TCP et CBR*

segments de données égale à 1500 octets et une taille de paquets d'acquittements de 40 octets, le protocole est TCP Reno. Le trafic CBR est aussi constitué de 10 connexions sur UDP, avec comme paramètres du trafic: 448 Kbit/s concernant le débit constant de transmission de paquets, 210 octets concernant la taille constante des paquets.

Les résultats sont dans les figures 5.15 et 5.16. Comme dans 5.2.5.1 et 5.2.5.2, l'utilisation de ACQ permet dans ces deux cas de trafic ascendant hétérogène d'atteindre une utilisation moyenne optimale du lien. Concernant VAQ, nous remarquons que rajouter des paquets de données appartenant à un trafic de type FTP augmente les performances de l'ordonnanceur VAQ comparé aux résultats des simulations dans 5.2.5.1 et 5.2.5.2.

5.2.5.4 ACQ et VAQ sont robustes en cas de trafics variés

Les simulations de cette section montrent donc que ACQ et VAQ fournissent aussi d'excellents résultats si des trafics différents de trafics TCP à longues durées caractérisent le trafic circulant sur le lien ascendant. En effet, nous avons fait des simulation en ayant un trafic TCP On/Off comme trafic T_r et ACQ a réussi à atteindre une valeur optimale de la fonction d'utilité. Pour VAQ, les performances se sont trouvées légèrement réduites mais toujours supérieures à celles données par FIFO ou CBQ.
Les simulations avec un trafic UDP à débit constant montrent aussi que ACQ et VAQ sont des ordonnanceurs qui dans tous les cas optimisent l'utilisation du lien asymétrique dans le cas où un trafic bidirectionnel le traverse.

5.3 Récapitulations des résultats

Avant de recommander de déployer un ordonnanceur ou un quelconque mécanisme dans l'immense réseau qu'est Internet, il est fondamental de s'assurer de la robustesse du mécanisme et de son efficacité même lorsque les conditions du réseau varient. De plus, il est aussi important de savoir que le dit mécanisme ne risque pas d'avoir des conséquences néfastes sur des critères autres que ceux pour lesquels il a été conçu, en d'autres termes savoir qu'un ordonnanceur tel que ACQ ou VAQ conçu pour maximiser l'utilisation d'un lien asymétrique n'altère pas d'autres critères tels que les délais de transmission ne peut qu'être favorable au déploiement de ACQ et VAQ. Nos simulations ont montré que, idéalement, selon qu'on donne une priorité aux délais des connexions ascendantes, aux connexions descendantes qu'on ne donne aucune priorité, il est préférable d'utiliser ACQ ou VAQ respectivement.

En effet, c'est exactement le cas dans ce chapitre. Nous avons mené des simulations qui montrent que ni ACQ ni VAQ n'affecte le délai aller retour (RTT) des connexions que ces ordonnanceurs manipulent. De plus, ACQ et VAQ se sont montrés robustes face à des changements dans le nombre de connexions impliquées dans le réseau de part et d'autre du lien asymétrique. Un changement dans le degré d'asymétrie du lien n'affecte en rien les bonnes performances des ordonnanceurs ACQ et VAQ. Dans le cas de présence d'erreurs dans les liens du réseau, VAQ et ACQ donnent de bonnes performances. Nous avons vu cependant que VAQ n'est pas recommandé si le taux d'erreur dépasse les 30 % sur le lien retour.
Nous avons aussi vu le comportement de ACQ et de VAQ dans le cas où le trafic passe par plus d'un lien asymétrique. Les performances de VAQ ont été excellentes dans ce cas. En effet VAQ est robuste et donne toujours une utilisation maximale de l'ensemble des liens. Ceci malheureusemenet n'a pas été le cas de ACQ.
Enfin, nous avons changé le type des trafics ascendants. Nous avons impliqué des trafic TCP mais à courtes durées (On/Off) et des trafics autres que TCP (UDP à débit constant CBR). Nos simulations ont montré que ACQ donnait d'excellents résultats. VAQ, de son coté, voyait ses performances légèrement réduites.

En conclusion, globalemenet ACQ et VAQ sont tous les deux robustes face à des changements dans les conditions du trafic. Leur usage est recommandé tant que le nombre de liens asymétrique est faible. Dès que le trafic passe par plusieurs liens asymétrique, il est recommandé d'utiliser VAQ qui a prouvé sa stabilité dans toutes les circonstances. De toute manière, utiliser juste FIFO avec les mécanismes de filtrage et de reconstruction des acquittements n'est même plus envisageable vu les résultats des simulations menées dans ce chapitre où FIFO a montré clairement ses limites. De plus, l'utilisation de CBQ avec aussi les mécanismes de filtrage et de reconstruction des acquittements est certes simple mais a de très mauvaises conséquences sur l'utilisation du lien asymétrique dans certains cas comme nous l'avons montré dans ce chapitre. Il est donc de meilleur usage d'utiliser ACQ ou VAQ afin de permettre une utilisation optimale des liens et une satisfaction de l'utilisateur.

Chapitre 6

Conclusion générale et perspectives de travaux futurs

l'Internet est un ensemble de ressources partagées. La bande passante des liens du réseaux ainsi que les files d'attente des routeurs sont des instances de ressources partagées. Pendant les moments de surcharge, le réseau peut être amené à rejeter des paquets pour cause d'insuffisance de ressources disponibles. La couche IP fournit uniquement un service sans garantie, *"Best-Effort"*. Elle ne garantie pas que tous les paquets injectés dans le réseau soient correctement délivrés à la destination. Puisque c'est le protocole TCP qui fournit le service de transmission de données fiable, il doit contenir des mécanismes de détection et de correction de pertes de paquets. Les paquets d'accusé de réception (paquets d'acquittement) représentent les fondements de détection et du recouvrement des erreurs du protocole TCP.

D'autre part, TCP est un protocole *full-duplex*, c'est à dire que chaque point d'extrémité d'une connexion peut envoyer des données à son point d'extrémité paire. Les paquets d'acquittements correspondant au transfert de données d'une direction peuvent donc se retrouver "mélangés" aux paquets de données. Le protocole TCP se retrouve alors dans une situation assez imprévisible où paquets de données et paquets d'acquittements se partagent l'accès à la bande passante. Les paquets de données sont généralement très volumineux ce qui implique une probable longue attente des paquets d'acquittements qui seront donc liés ensemble et possiblement rejetés. Les paquets d'acquittement, quant à eux, ne répondent pas à des notifications de congestion du réseau et pourraient aisément monopoliser l'accès au lien au dépens des paquets de données.

Le problème de partage des ressources entre paquets de données et paquets d'acquittements est amplifié dans le cas de passage par des liens asymétriques. En effet, les recherches sur le protocole TCP connaissent une nouvelle dimension qu'est l'analyse des performances dans des réseaux à accès asymétrique, tels que les réseaux câblés, satellites ou ADSL. Seuls de tels réseaux sont capables d'atteindre les hauts débits éxigés par les nouveaux utilisateurs du réseau, c'est pourquoi ils prolifèrent chaque jour et

connaissent un franc succés. Seulement, ce type de réseau représente un défi pour le protocole TCP puisque la bande passante disponible dans la direction de tranfert des paquets de données est largement supérieure à celle disponible dans la direction opposée qui sert à transférer les paquets d'acquittements. Ces derniers vont se retrouver fortement retardés voir même rejetés si les ressources sont en plus partagées par des paquets de données d'un trafic de sens opposés.

6.1 Contributions de la thèse

Dans un premier temps, nous avons détaillé les conséquences d'une interaction entre paquets de données et paquets d'acquittements sur un lien asymétrique. La conséquence majeure d'une telle situation est le regroupement des acquittements qui engendre à son tour des problèmes tels que lenteur de l'incrémentation des fenêtres de congestion et rafales de paquets de données.

Dans ce contexte, nous avons proposé deux mécanismes qui ont pour objectif de maximiser l'utilisation d'un lien asymétrique sur lequel transite un trafic bidirectionnel. L'utilisation de ces mécanismes de la part des différents opérateurs et fournisseurs de service est recommandée afin d'atteindre la satisfaction de leurs utilisateurs. Nos mécanismes se basent sur deux approches différentes: une approche d'agrégation du trafic pour ACQ et une approche à plus fine granularité qui agit sur les paquets pour VAQ. Mais ils se rejoignent dans leur principe de différencier entre paquets de données et paquets d'acquittements et de leur infliger des "manipulations" différentes. Nous nous sommes intéressés au cas d'un trafic ascendant dont les paquets de données circulent sur le lien haut débit et d'un trafic descendant dont les paquets de données empruntent le lien bas débit. La distinction entre paquets de données et paquets d'acquittements se fait grâce à un bit dans l'entête du paquet et au calcul de la taille des paquets puisque les paquets d'acquittements ont une longueur égale à zéro. Cependant dans le cas où la source TCP du trafic ascendant est aussi celle du trafic descendant, les paquets d'acquittements sont dans ce cas "encastrés" dans des paquets de données (*piggybacked ACK*). Nos mécanismes sont aussi utilisables dans un tel cas, il suffit alors de considérer un tel paquet d'acquittement comme une concaténation entre un paquet de données et un paquet d'acquittement. Le traitement diffère alors selon l'ordonnanceur.

6.1.1 ACQ pour satisfaire l'utilisateur en agrégeant les trafics

Le premier mécanisme que nous proposons dans cette thèse se base sur une agrégation des trafics afin de maximiser l'utilisation d'un lien asymétrique traversé par un trafic bidirectionnel. ACQ est un ordonnanceur qui doit être placé à l'entrée du lien bas débit. Il utilise deux classes de trafics de même priorité: une classe pour les paquets de données et une classe pour les paquets d'acquittement. Si un paquet d'acquittement est *piggybacked* il sera considéré comme paquet de données et placé dans la classe de données. Le principe de ACQ est d'adapter les poids des classes en fonction du trafic qui passe dans le but de toujours maximiser l'utilité préalablement définie. Dans un

premier temps, nous avons défini la satisfaction de l'utilisateur comme étant l'utilisation du lien asymétrique, mais nous avons montré que ACQ est facilement capable de s'adapter à d'autres critères définissant la satisfaction de l'utilisateur.
ACQ a démontré son efficacité, il permet de maximiser l'utilisation du lien asymétrique. Nous avons mené des simulations qui montrent que ACQ améliore l'utilisation du lien de 60% comparé à un simple ordonnanceur FIFO sur lequel on applique les mécanismes de filtrage et de reconstruction des acquittements afin de remédier aux effets de l'asymétrie du lien. La capacité de ACQ de s'adapter au trafic pour l'allocation de la bande passante lui donne aussi de forts avantages comparé au mécanisme CBQ. En effet, nos simulations ont clairement démontré que CBQ était souvent la cause de sous-utilisation du lien du fait de son partage fixe de la bande passante.

6.1.2 VAQ pour satisfaire l'utilisateur en ayant une vue à fine granularité du trafic

VAQ est le second mécanisme que nous proposons dans cette thèse afin d'ordonnancer entre paquets de données et paquets d'acquittements et de maximiser ainsi l'utilisation d'un lien asymétrique traversé par un ou plusieurs trafics bidirectionnels. VAQ classe aussi les paquets en deux files d'attente, une pour les paquets de données et une pour les paquets d'acquittements. À l'opposé de ACQ, VAQ utilise une approche à fine granularité sans agrégation de trafic et traîte chaque paquet individuellement. VAQ se base sur la taille virtuelle des paquets d'acquittements représentée par le nombre total d'octets que chaque paquet d'acquittement accuse la reception à la source et tente à chaque passage de paquet par le lien à bas débit d'allouer la bande passante relativement à cette taille virtuelle des paquets d'acquittements. Pour ce faire, il accorde un crédit à la file des données et met à jour la valeur du crédit en fonction du trafic. Si un paquet d'acquittement arrivant est *piggybacked*, il est placé dans la file d'attente des paquets de données mais VAQ utilise tout de même les informations contenues dans son entête pour mettre à jour la valeur du crédit (c'est comme si VAQ recevait un acquittement avec une taille nulle).
Les performances de VAQ ont été testées dans cette thèse via des simulations qui ont montré l'efficacité de VAQ. En effet, en utilisant VAQ, nous avons pu atteindre près de 87.5% de l'utilisation globale du lien asymétrique et satisfaire ainsi l'utilisateur. VAQ est d'autre part simple à mettre en place, ne nécessitant ni le stockage d'information dans les routeurs ni un choix cornélien de paramètres, ce qui favorise son utilisation de la part des opérateurs et fournisseurs de service.

6.1.3 Confrontation des ordonnanceurs proposés

Une première différence évidente entre les deux ordonnanceurs ACQ et VAQ est l'approche retenue et suivie par chacun des ordonnanceurs. Un opérateur ou fournisseur de service susceptible d'être interessé par ACQ peut en fait être intéréssé par l'approche à large granularité s'il connait parfaitement son trafic et si ce dernier n'est pas appelé à changer souvent. En effet, dans ce cas, un ordonnanceur de type ACQ

qui agrège les trafics en deux uniques classes est préférable vu qu'il ne nécessite pas de manipulation de paquets[1] et parvient tout de même à maximiser l'utilisation globale du lien asymétrique et satisfaire l'utilisateur. En revanche, dans le cas où le trafic varie souvent, l'opérateur ne peut pas se contenter d'un ordonnanceur qui agrège le trafic et se doit d'utiliser un ordonnanceur ayant une approche à fine granularité de type VAQ. En manipulant ainsi chaque paquet, VAQ est en effet plus "finement" capable d'atteindre ses objectifs que ACQ; il atteint plus rapidement une utilisation globale optimale du lien asymétrique.

Une deuxième différence sont les fonctions d'utilités auxquelles les deux ordonnanceurs peuvent répondre. En effet, ACQ a la faculté de pouvoir s'adapter à toutes les fonctions possibles, répondant ainsi à plusieurs critères aussi bien de l'utilisateur que de l'opérateur. En effet, et vu que les prix des liens ascendants sont généralement les plus chers, un utilisateur peut vouloir plutôt donner la priorité à la maximisation de l'utilisation de ce lien. Dans ce cas, la fonction d'utilité à maximiser est différentes de celle utilisé dans notre étude. Cependant, ACQ tel que conçu dans le chapitre 3 peut parfaitement s'adapter à d'autres fonctions selon l'équation (3.6). En revanche, VAQ tel que conçu dans le chapitre 4 répond uniquement à la fonction d'utilité qui est de maximiser l'utilisation des deux liens, à savoir le lien ascendant et le lien descendant, et les résultats de VAQ sont alors meilleurs que ceux de ACQ. Certe, nous pouvons l'adapter à d'autres fonctions d'utilité mais il necessiterait alors plus de mesures de trafics.
D'un autre coté, dans le cas d'un trafic bidirectionnel composé d'un trafic UDP dans le sens descendant, les paquets d'acquittements sur le chemin ascendant sont inexistants. Il est par conséquent plus difficile pour VAQ d'optimiser les performances dans un tel cas puisque cet ordonnanceur agit sur les paquets d'acquittements. Cependant, il est possible dans ce cas d'utiliser les rapports de RTCP (*Real-time Transfert Control Protocol*) basé sur des transmissions périodiques de paquets de contrôle par tous les participants dans un trafic UDP. ACQ en revanche n'aura pas de difficulté à s'adapter à une telle situation.

Une autre différence entre ACQ et VAQ est leur impact, bien que minime, sur les délais de transmission des trafics. Nous sommes donc en présence d'un trafic dont les paquets de données circulent sur le lien à haut débit, appelé trafic aller et d'un autre trafic dont les paquets de données circulent sur le lien bas débit appelé trafic retour. Bien évidemment, le fait d'appliquer un ordonnanceur particulier plus complexe que FIFO à l'entrée d'un lien va forcément augmenter le délai aller retour de toute connexion qui passe par cet ordonnanceur. ACQ et VAQ augmentent en effet le délai aller retour (RTT) des trafics aller et retour, mais cette augmentation est vraiment très légère et elle dépend de plus de l'ordonnanceur. En effet, si le délai du trafic aller doit impérativement être bas (si, par exemple, le trafic aller est constitué d'un trafic à temps réel circulant au dessus de TCP). Notre étude a montré que dans ce cas il est

1. autre que la classification

préférable d'utiliser ACQ. Si, au contraire, c'est le trafic retour qui doit absolument répondre à une contrainte de délai, l'ordonnanceur à utiliser est VAQ. Quoi qu'il en soit, notre étude a montré que ACQ et VAQ n'influent que très légèrement sur les délais des trafics.

ACQ et VAQ se sont montrés stables et robustes en cas de changements des conditions du trafic. Nous avons en effet effectué un grand nombre de simulations qui ont démontré que ACQ et VAQ fournissent de très bonnes performances si l'on modifiait quelques paramètres du réseau. Il est à noter que toutes les modifications que nous avons effectué représentent désormais des situations des plus courantes et fréquentes dans l'Internet d'aujourd'hui. C'est pourquoi, il était important de tester la robustesse de nos ordonnanceurs dans ces cas là.
ACQ et VAQ sont robustes en cas de nombre de connexions variables. En effet, que nous soyons dans le cas d'un réseau peu chargé ou alors en cas de surcharge du trafic dans le réseau, ACQ et VAQ donnnent toujours une utilisation optimale du lien asymétrique. Nous avons aussi testé la robustesse de ACQ et VAQ en cas de degré d'asymétrie différents, c'est à dire en variant les capacités des liens aller et retour. Encore une fois, ACQ et VAQ donnent d'excellents résultats et sont capables de fournir une utilisation maximale du lien même en cas d'importante asymétrie. Dans le cas fréquent de présence d'erreurs dans le réseau et même en cas de présence de trafics autre que du long terme TCP dans le réseau, nos ordonnanceurs ont montré de bonnes performances, avec une légère préférence pour l'ordonnanceur ACQ. Cette préférence est compensée par notre dernier test, où VAQ s'est révélé être plus stable en cas de passage du trafic par plus d'un lien asymétrique.

Nos simulations constituent en fait un pas important vers un éventuel déploiement de nos ordonnanaceurs. Tester ces ordonnaneurs dans un réseau expérimental est le prochain pas que nous nous fixons. En effet, tous ces résultats de simulations encouragent le déploiement de ACQ et de VAQ dans l'Internet, et il était important de s'en assurer. Cependant, il est aussi important maintenant de faire un "*mapping*" de ACQ et VAQ dans un noyau UNIX et de voir leurs performances dans l'utilisation d'un lien asymétrique sur lequel transite un trafic bidirectionnel. Pour ce faire, ACQ et VAQ seront implémentés dans un routeur d'accès. Après la classification, les paquets passent alors par nos ordonnanceurs avant dêtre injectés dans le réseau. Ce *mapping* constitue une de nos perspectives de travaux futurs que nous décrivons immédiatement dans la section suivante.

6.2 Perspectives et travaux futurs

Le futur de l'Internet dépendra en réalité des usages auxquels il répondra. Le web, le *chat* et l'e-mail sont aujourd'hui les trois services les plus utilisés sur l'Internet. De nouveaux services émergent, dont certains nécessiteront des architectures complexes. Par ailleurs, les possibilités des réseaux et des interfaces ne cessent de croître, certaines

sont des moyens d'accès non standards à l'Internet. L'Internet est amené à évoluer et à considérer ces nouveaux acteurs et les achitectures doivent nécessairement s'adapter. La notion de l'Internet *Everywhere* aboutira à une architecture dont le centre ne peut en aucun cas être vide, il faudra notamment gérer des informations à propos des utilisateurs eux-mêmes.
A l'heure actuelle, on accède à l'Internet essentiellement à partir de réseaux de bureaux ou résidentiels, selon un modèle client-serveur. La tendance actuelle est à la multiplication des réseaux d'accès et à "l'hétérogénéisation" de l'Internet. Ainsi sont apparus les réseaux WLAN qui sont des réseaux locaux sans fil, fondés sur les protocoles tels que le 802.11. On assiste ainsi à une convergence partielle de ces différents réseaux. Pour réaliser cette convergence, il y a besoin de mettre en place des structures et des mécanismes qui permettent la communication entre ces sous-réseaux et surtout qui fournissent des performances optimales. Nos travaux futurs s'inscrivent dans ce cadre et porterons essentiellement sur la conception de ces structures afin de permettre au caractère hétérogène de l'Internet de ne plus impliquer des dégradations de performances.

Bibliographie

[1] M. Allman "TCP Byte Counting Refinements" Mark Allman, ACM Computer Communication Review, Volume=3, Numéro 29, 1999.

[2] M. Allman "TCP Congestion Control with Appropriate Byte Counting (ABC)" RFC 3465.

[3] M. Allman, V. Paxson, W. Stevens "TCP Congestion Control" RFC 2581.

[4] A. Aggarwal, S. Savage, T. Anderson. "Understanding the Performance of TCP Pacing" INFOCOM 2000.

[5] R. Braden "Requirements for Internet Hosts - Communication Layers" Octobre 1989, RFC 1122.

[6] C. Barakat, E. Altman, W. Dabbous "On TCP Performance in a Heterogeneous Network: A Survey" URL: http://www.citeseer.nj.nec.com/barakat00tcp.html

[7] C. Barakat, and E. Altman. "On ACK Filtering on a Slow Reverse Channel", proceedings of the first international workshop on Quality of future Internet Services (QofIS), Berlin, Germany, September 2000.

[8] C. Boutremans, J.Y. Le Boudec "Adaptive Delay Aware Error Control for Internet Telephony" Proceedings of 2nd IPTelephony workshop, Columbia University, New York, April 2001, pp 81–92.

[9] H. Balakrishnan, V. N. Padmanabhan, G.Fairhurst, M. Sooriyabandara "TCP Performance Implications of network Path Asymmetry" IETF RFC 3449 December 2002 .

[10] H. Balakrishnan, V.N. Padmanabhan, and R.H. Katz. "The Effects of Asymmetry on TCP Performance" ACM Mobile Networks and Applications (MONET), 4(3), 1999.

[11] H. Balakrishnan PhD Thesis "Challenges to Reliable Data Transport over Heterogeneous Wireless Networks" Aug 1998.

[12] C. Barakat "Evaluation des performances du contrôle de congestion dans l'Internet" PhD équipe Mitral INRIA Sophia Antipolis 2001.

[13] J. Bennet, H. Zhang "WF2Q: Worst-case Fair Weighted Fair Queueing", Proceedings Of IEEE Infocom '96, March, 1996.

[14] Cisco TechNotes, "Adjusting IP MTU, TCP MSS, and PMTUD on Windows and Sun Systems" http://www.cisco.com/warp/public/105/38.shtml

[15] K. claffy, S. McCreary. "Internet Measurement and Data Analysis: Topology, Workload Performance and Routing Statistics" NAE Workshop, 1999. http://www.caida.org/Papers/Nae/

[16] A. Calveras, J. Linares, J. Paradells "Window Prediction Mechanism for Improving TCP in Wireless Asymmetric Links" Proceedings of IEEE Global Communications Conference (GLOBECOM) Sydney, November 1998, pp.533-538.

[17] T. Cover, J. Thomas "Elements of Information theory" John Wiley and Sons, 1991.

[18] J. Davis, E. Lemar, A. Mohr "Investigating Tradeoffs in Drop-Tail Queuing"

[19] A. Duda "Advanced Computer Networks" URL: http://duda.imag.fr/3at

[20] S. Floyd "TCP and Explicit Congestion Notification" ACM Computer Communication Review, Volume 24, Numéro 5, October 1994, pp. 10–23.

[21] S.Floyd. "Notes on CBQ and Guranteed Service" Draft document, July 1995. Disponible à l'URL http:// www.aciri.org/floyd/cbq.html.

[22] S. Floyd "Issues of TCP with SACK" Technical report, January 1996.

[23] S. Floyd. "RED: Discussions of Setting Parameters", November 1997, URL: http://www.icir.org/floyd/REDparameters.txt.

[24] K. Fall."Network Emulation in the Vint/NS Simulator" Proceedings of the fourth IEEE Symposium on Computers and Communications, 1999.

[25] Wu-chang Feng "Improving Internet Congestion Control and Queue Management Algorithms" PhD thesis, University of Michigan, 1999.

[26] K. Fall, S. Floyd "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP" Computer Communication Review, Volume 26, Numéro 3, July 1996, pp. 5–21.

[27] S. Floyd and T. Henderson "The NewReno modification to TCP's fast recovery algorithm" Request for Comments 2582, Internet Engineering Task Force, Apr. 1999.

[28] V. Firoiu, M. Borden. "A Study of Active Queue Management for Congestion Control" INFOCOM 2000.

[29] S. Floyd, V. Jacobson. "Random Early Detection gateways for Congestion Avoidance" Volume 1, Numéro 4, August 1993, pp. 397–413.

[30] S. Floyd, V. Jacobson. "Link-sharing and Resource Management Models for Packet Networks" IEEE/ACM Transactions on Networking, Volume 3, Noméro=4, August 1995.

[31] W.C. Feng, D.D. Kandlur, D. Saha, K.G. Shin "A Self-Configuring RED Gateway" Proceedings of INFOCOM 99.

[32] Wu-chang Feng. "Improving Internet Congestion Control and Queue Management Algorithms". PhD thesis, University of Michigan, 1999.

[33] J. Hoe "Improving the Start-up Behavior of a Congestion Control Scheme for TCP" ACM SIGCOMM August 1996.

[34] O. Ait Hellal, E. Altman "Analysis of TCP Vegas and TCP Reno" Proc. IEEE ICC'97, 1997.

[35] C. V. Hollot, V. Misra, D. Towsley, Wei-Bo Gong. "A Control Theoretic Analysis of RED" INFOCOM 2001.

[36] V. Jacobson."Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[37] S. Johnson. "Increasing TCP Throughput by Using an Extented Acknowledgement Interval" Master's Thesis, Ohio University, June 1995.

[38] V. Jacobson. "Congestion avoidance and control", ACM SIGCOMM, August 1998.

[39] R. Jain "The art of computer systems performance analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling" Edition WILEY.

[40] W. Jiang, T.F. Williams "Detecting And Measuring Asymmetric Links In An IP Network" Technical Report CUCS009 -99, Columbia University, New York, January 1999.

[41] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, J. Mahler "Real-time voice over packet-switched networks" IEEE Networks, pp. 18–27, january/february 1998.

[42] P. Karn, C. Partridge "Estimating round-trip times in reliable transport protocols". Proceedings of SIGCOMM '87, August 1987, ACM.

[43] S. Kunniyur, R. Srikant. "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", INFOCOM 2000.

[44] S Kalyanaraman, D. Shekhar, K. Kidambi. "TCP/IP Performance Optimization over ADSL", GI2000.

[45] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. "Improving TCP throughput over two-way asymmetric links: analysis and solutions", In Proceedings of Sigmetrics, 1998.

[46] M. Lenercier. "Architecture et Evaluation d'un mécanisme à Rejets Selectifs Multiples". Thèse de l'Université de Paris VI, 1995.

[47] D. Lin, R. Morris. "Dynamics of Random Early Detection" Proceedings of SIGCOMM '97.

[48] T.V. Lakshman, U. Madhow, and B. Suter. "Window-based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance", IEEE INFOCOM, April 1997.

[49] T.V. Lakshman, B. Suter. "TCP/IP Performance with Random Loss and Bidirectional Congestion", IEEE/ACM transactions on networking, Volume 8, Numéro 5, October 2000.

[50] Z.Q. Luo, P. Tseng "Analysis Of An Approximate Gradient Projection Method With Applications To The Backpropagation Algorithm" Hamilton, Ontario, L8S 4L7, Canada, 1993.

[51] U. Madhow, "Dynamic congestion control and error recovery over a heterogeneous Internet" (invited paper), IEEE CDC, 1997.

[52] R. Morris, "Scalable TCP Congestion Control", IEEE INFOCOM 2000, March 2000, pages 1176-1183.

[53] M. May, J. Bolot, C. Diot, B. Lyles. "Reasons not to deploy RED" Proc. of 7th. International Workshop on Quality of Service (IWQoS'99), London.

[54] J. Mogul, S. Deering "Path MTU discovery" RFC 1191.

[55] M. May, C. Diot, B. Lyles, J. Bolot. "Influence of active queue management parameters on aggregate traffic performance". Research Report, Institut National de Recherche en Informatique et en Automatique, April 2000.

[56] R. Mahajan, S. Floyd, D. Wetherall. "Controlling High-Bandwidth Flows at the Congested Router" ACM 9th International Conference on Network Protocols (ICNP), November 2001.

[57] V. Misra, W. Gong, D. Towsley "A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED" Proceedings of ACM SIGCOMM'00, (Stockholm, Sweden, September 2000).

[58] I.T. Ming-Chit, D. Jinsong, W. Wang "Improving TCP performance Over Asymmetric Networks", ACM SIGCOMM, ACM Computer Communications Review (CCR), Volume 30, Numéro 2, 2000.

[59] J. Mo, R.J. La, V. Anantharam, J. Walrand "Analysis and Comparison of TCP Reno and Vegas" Proceedings of INFOCOM, 1999.

[60] M. Mathis, J. Mahdavi "Forward Acknowledgement: Refining TCP Congestion Control" SIGCOMM 96, August 1996.

[61] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow "TCP Selective Acknowledgement Options" RFC 2018, April 1996.

[62] Ns network simulator, available via http://www-nrg.ee.lbl.gov/ns/

[63] Y. Onoe, Y. Atsumi, F. Sato, T. Mizuno "A Dynamic Delayed ACK Control Scheme on MobileIP Networks" 2001 International Conference on Computer Networks and Mobile Computing (ICCNMC'01) October 16 - 19, 2001, Beijing, CHINA.

[64] T. J. Ott, T. V. Lakshman, L. Wong, "SRED: Stabilized RED," Proceedings of IEEE INFOCOM'99, March 1999.

[65] J. Postel "Transmission Control Protocol" September 1981. RFC 793.

[66] V. Paxson "Automated Packet Trace Analysis of TCP Implementation" ACM SIGCOMM, September 1997.

[67] J. Padhye, V. Firoiu, D. Towsley and J. Kurose (2000) "Modeling TCP Reno performance: a simple model and its empirical validation". IEEE/ACM Transactions on Networking 8, pp 133–145.

[68] A.K. Parekh, R.G. Gallager. "A generalized processor sharing approach to flow control in integrated services networks: the single-node case" IEEE/ACM Transactions on Networking (TON) ,Volume 1, Issue 3, pp 334–357, June 1993.

[69] PILC: Performance Implications of Link Characteristics Working Group, URL: http://www.ietf.org/html.charters/pilc-charter.html.

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

[71] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998

[72] N.K.G. Samaraweera "Return link optimization for internet service provision using DVB-S networks" ACM SIGCOMM Computer Communication Review, Volume 29, Issue 3, Pages: 4–13, July 1999.

[73] M. Allman, S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, J. Semke. "Ongoing TCP Research Related to Satellites" RFC 2760.

[74] B. Suter, T.V. Lakshman, D. Stiliadis, A. Choudhury. "Efficient Active Queue Management for Internet Routers" Proceedings of Interop Engineers Conference, Las Vegas, NV, May 1998.

[75] D. Shekhar, H. Qin, S. Kalyanaraman, K. Kidambi, "Performance Optimization of TCP/IP over Asymmetric Wired and Wireless Links" Invited paper at European Wireless 2002, February 2002.

[76] M. Shreedhar, G. Varghese. "Efficient Fair Queuing using Deficit Round Robin" SIGCOMM 1995.

[77] F. Toutain. "Internet à Integration de Services", acte de GRES97, Rennes, 1997.

[78] I. Tam Ming-Chit, D. Jinsong, W. Wang. "Improving TCP performance over asymmetric networks" ACM SIGCOMM Computer Communication Review, Volume 30, Issue 3, pp 45–54, July 2000.

[79] S. Varma "Performance and Buffering Requirements of TCP Applications in Asymmetric Networks" INFOCOM 1999, pp 1548–1555.

[80] I. Wakeman, A. Ghosh and Jon Crowcroft. "Implementing Real Time Packet Forwarding Policies Using Streams", USENIX Winter, 1995.

[81] H. Zhang, D. Ferrai "Rate Controlled Static Priority Queuing" Proceeding of INFOCOM'93 pp 227–236.

[82] T. Ziegler, S. Fdida, U. Hofmann, "RED+ Gateways for Identification and Discrimination of unfriendly best-effort Flows in the Internet", Proceedings of IFIP Broadband Communications 99, November 1999.

[83] L. Zhang, S. Shenker, and D.D. Clark. "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic", In Proc. SIGCOMM '91 Symposium on Communications Architectures and Protocols, pp 133–147, Zurich, September 1991.

[84] Y. Zhang and L. Qiu, "Understanding the end-to-end performance impact of RED in a heterogeneous environment," Cornell CS Technical Report 2000-1802, July 2000.

[85] L. Qiu, Y. Zhang, S. Keshav. "Understanding the Performance of Many TCP Flows" Computer Networks (Amsterdam, Netherlands: 1999).

[86] J.Y. Le Boudec. "Rate Adaptation, Congestion Control and Fairness: A Tutorial" Ecole Polytechnique Fédérale de Lausanne (EPFL) Dec.2000.