UNIVERSITE LIBRE DE BRUXELLES
Faculté des Sciences Appliquées
Ecole Polytechnique
Service d'Informatique et Réseaux

Année académique 2002-2003

Evaluation de performances d'une amélioration de TCP Westwood

Directeur de mémoire:
Prof. Marie-Ange Remiche

Dissertation présentée par
JOËL CANNAU en vue de
l'obtention du Diplôme d'études
Approfondies en Sciences
Appliquées

Remerciements

Ce mémoire est l'aboutissement de longues heures de travail portant sur un sujet très pointu et passionnant du domaine de la transmission de l'information. Le monde très vaste des télécommunications ne peut se prétendre d'être entièrement connu, mais ce travail m'a permis d'en découvrir quelques facettes.

Je voudrais remercier tout particulièrement mon promoteur, le Professeur Marie-Ange Remiche, de m'avoir supervisé et encouragé tout au long de ces deux années de recherche dans la poursuite quotidienne de mon travail. Sa passion pour la précision m'a permis de développer un esprit plus critique et d'aborder les problèmes avec plus de rigueur. Son expérience et ses connaissances m'ont apporté une aide précieuse dans la réalisation de ce travail.

Merci également pour la confiance que m'a accordée le Professeur Guy Latouche, et de m'avoir permis de partager l'expérience des groupes de travail portant sur les recherches satellitaires et de télécommunication en général.

Je remercie aussi le Professeur Esteban Zimanyi, directeur du Service Informatique et Réseaux, pour l'encadrement considérable et le soutien qu'il m'a apporté depuis déjà plusieurs années. Merci aussi au Professeur Paul Van Binst, directeur du service de Télématique et communication pour les contacts apportés dans le monde industriel.

Merci enfin à ceux que je n'ai pas cités, mais qui se reconnaîtront, pour les excellents moments passés en leur compagnie et sûrement les longues conversations que nous avons pu tenir.

Introduction

Le début du 21ième siècle, empreint par l'éclosion de l'ère des télécommunications sans fil, ouvre la voie à de nouveaux objectifs et défis pour l'élaboration de nouvelles techniques de transmission mais aussi pour la mise en place de systèmes permettant la cohabitation avec les technologies déjà existantes du monde des réseaux câblés.

Aujourd'hui, la grande majorité des réseaux sont constitués d'une infrastructure centrale, dotée d'une capacité de transmission à haut débit, et la plupart du temps construite sur base d'un réseau câblé et statique. Comme celui-ci ne permet pas une mobilité étendue des équipements qui y sont connectés, une nouvelle génération de réseaux fut définie. Par celle-ci, des périphériques jusqu'alors voués à une place fixe, sont dotés d'équipement radio par lesquels ils peuvent entrer en dialogue avec d'autres tout en leur assurant une mobilité plus ou moins large.

Cette inauguration de nouveaux comportements a mis à jour des faiblesses dans les transmissions de l'information suite à l'hétérogénéité des milieux de propagation. Les réseaux fixes initialement conçus pour des communications de bout en bout transitant exclusivement par voie câblée sont aujourd'hui mis à contribution pour le transit entre et vers des réseaux sans fil. Ces modifications de comportement ne sont pas sans conséquences pour la plupart des algorithmes conçus pour le contrôle de l'acheminement de l'information. Par exemple une perte de données pourra être interprétée de diverses manières selon qu'elle provienne d'une saturation d'un lien du réseau ou d'une perte subie par cause d'un milieu de propagation peu fiable, ou encore par suite d'une défaillance technique survenue dans un équipement assurant la transmission. Les algorithmes développés jusqu'à présent ne font pas la distinction entre ces différentes interprétations.

Notre travail dans ce DEA s'inscrit dans la lignée de la thèse entreprise dans le cadre de l'évaluation de performances de réseaux hétérogènes. Dans la thèse, nous désirons déceler les enjeux et les contraintes liés au passage des données issues des points d'accès sans fil pour lesquels une qualité de service est souhaitée, vers les réseaux câblés. Ces points d'accès peuvent être constitués par des mobiles supportant une compatibilité avec différentes normes pour lesquelles des transmissions multimédia temps réel sont réalisables. Différents facteurs peuvent intervenir dans ces contraintes mais le plus important sera sans doute celui lié au manque inévitable de qualité du transfert, causé par les pertes de transmission dans l'air.

Pour mieux comprendre ces influences, nous avons commencé par scinder l'étude du problème en deux parties:

- La première tend à comprendre les caractéristiques inhérentes aux protocoles de transport de bout en bout développés dans un contexte de réseaux câblés.

- La seconde porte sur la transmission entre les points finaux mobiles.

La première partie débute par une étude bibliographique concernant les procédures et méthodes assurant la certitude du transfert de l'information de bout en bout. Ces algorithmes sont localisés au niveau des émetteurs et des récepteurs finaux. Cette localisation offre une accessibilité aisée pour toute modification éventuelle. Il n'entrait pas dans notre optique de nous focaliser sur les algorithmes de routage de l'information qui résident au niveau des nœuds intermédiaires du réseau.

La seconde partie, permettra d'avoir une connaissance plus globale des problèmes de télécommunication liée à la mobilité des sources et des récepteurs. Son approche est facilitée par la compréhension des phénomènes de transmission que nous aurons étudiés dans la première partie. Ce volet, qui s'inscrit dans notre travail de thèse, a pour objectif d'aborder les problèmes de performances de solutions appropriées au monde des réseaux sans fil. On y abordera, notamment, l'étude des boosters de protocoles.

Ce travail de DEA est représentatif de la première partie de l'objectif. Le DEA a pour objectif d'avoir une connaissance suffisante et précise des protocoles construits sur base d'une transmission sur réseaux câblés. Dans celui-ci nous analysons un ensemble d'algorithmes d'évitement de congestion des réseaux et l'influence des pertes de données sur les prises de décision de ces algorithmes. Ces algorithmes sont situés au niveau de la couche transport du modèle OSI. Nous pourrons observer que si aucune modification ne leur est apportée, ils contribueront à rendre les performances de transmission plutôt désastreuses sur les réseaux hétérogènes. En effet, les protocoles de transport de l'information tels que conçus précédemment et qui se basent sur les pertes de paquets pour ajuster leurs paramètres sont moins performants que les algorithmes dans lesquels on se base sur une estimation de la bande passante disponible pour également ajuster ces paramètres. Les pertes aléatoires de paquets propres au réseaux sans fils ne sont que mal traitées.

Ces algorithmes de départ ont fait l'objet de multiples recherches jusqu'à aujourd'hui de manière à les rendre plus performants. Nous essayerons, dans le contexte de ce DEA, d'y apporter notre contribution et d'analyser la validité de notre proposition. L'étape de recherche bibliographique des solutions déjà existante est indispensable à cet effet. Celle-ci permet de repérer les éventuels problèmes potentiels. Nous nous sommes focalisés sur un problème de fin de sessions multiples de connexions ftp établies par un client.

Le principe de l'amélioration est le suivant:

Il s'agit d'augmenter, sur base d'une prise en considération de fin de sessions, de manière proportionnelle au débit libéré, les connexions encore actives du client.

Cette modification de protocole sera présentée via des simulations réalisées à l'aide de NS, qui est un outil de simulation de protocoles réseau.

Le travail présenté dans ce rapport sera composé de 4 chapitres:

Le chapitre 1 présente les protocoles de contrôle de la couche transport et leur utilité dans le schéma des connexions avec garantie de délivrance. Les algorithmes utilisés dans les protocoles de contrôle de congestion sont présentés. Un inventaire de quelque uns des protocoles de contrôle sera dressé.

Le chapitre 2 présente une version particulière d'un protocole de contrôle de congestion : TCP Westwood. Il s'agit d'un protocole dans lequel l'émetteur peut récupérer plus rapidement son débit de transmission après des pertes et plus particulièrement sur des connections sans fil.

Le chapitre 3 présente les améliorations que nous avons apportées au protocole TCP. Nous y présentons le résultat de nos simulations. Ceux-ci sont commentés et nous discutons des performances de la solution proposée.

Le chapitre 4 présente les conclusions du DEA ainsi que sa mise en rapport avec la thèse. Nous y formulons nos objectifs futurs.

Chapitre 1

À propos de TCP

Ce chapitre vise à présenter les protocoles de transmission de bout en bout, leur utilité et leur fonctionnement. La première partie présente les mécanismes de base de ces protocoles. La seconde dresse un aperçu de quelques uns de ces protocoles. La troisième partie formule quelques remarques sur un mécanisme présent dans les protocoles de transport.

1.1 Considérations générales

1.1.1 TCP et le modèle OSI

Le protocole auquel nous allons porter notre attention durant tout ce travail fait partie de la suite des protocoles du modèle OSI. La suite de protocoles OSI, ou Open System Interconnection protocols, regroupe un ensemble de protocoles qui sont basés sur le modèle de référence OSI. Ces protocoles font partie d'un programme international qui recherche à développer des protocoles de télécommunication et des standards pour assurer une interopérabilité entre les équipements de différents vendeurs. Les spécifications du modèle OSI ont été conçues et implémentées en 1984 par deux organisations internationales : l'International Organisation for Standardization (ISO) et l'Internation Telecommunication Union (ITU). Le modèle de référence OSI décrit comment les informations issues d'un logiciel sur un ordinateur rejoint par l'intermédiaire d'un réseau, une autre application située sur un autre ordinateur. Le modèle OSI est un modèle conceptuel subdivisé en sept couches, chacune spécifiant une fonction du réseau particulière. Il permet de diviser les fonctions impliquées pour l'acheminement de l'information en sept groupes de tâches plus indépendamment gérables. Un groupe de tâches est donc associé à chacune des sept couches. Les tâches d'un groupe sont suffisamment bien définies de sorte qu'elles peuvent être implémentées de manière autonome. Ces couches sont présentées sur la figure suivante [Fig 1.1] On peut encore effectuer une subdivision entre les différentes couches en les classifiant entre les couches hautes et les couches basses. Les couches hautes traitent des caractéristiques d'implémentation des applications et sont proches de l'utilisateur final, alors que les couches basses traitent du transport des données proprement dit.

Application
Présentation
Session
Transport
Réseau
Lien
Physique

[Fig 1.1]

Du modèle OSI, nous allons nous intéresser plus particulièrement aux protocoles de la couche transport et plus particulièrement au protocole TCP et ses différentes versions. La couche de transport défini les services de transport de données émises avec fiabilité ou non. De manière générale on y retrouve les fonctions qui traitent les contrôles de flux, le multiplexage, la gestion des circuits virtuels, les corrections et traitement des erreurs. Le contrôle de flux gère la transmission entre les périphériques de sorte que ceux-ci n'envoient pas plus de données sur le réseau que ce que le périphérique récepteur ne peut recevoir. Le protocole TCP, ou Transport Control Protocol, issu de la suite des protocoles TCP/IP est une implémentation de protocole de la couche transport.

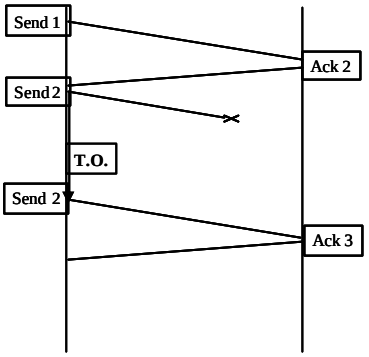
Les réseaux sur lesquels nous travaillons sont des réseaux basés sur la transmission à commutation de paquets à l'opposé des réseaux à commutation de circuit. Comme des paquets envoyés vers une même destination peuvent chacun emprunter des routes différentes pour la rejoindre, il faut leur doter d'un système de numérotation pour qu'à l'arrivée le récepteur puisse les agencer dans le bon ordre; il se peut qu'une route plus rapide qu'une autre permette à un paquet émis ultérieurement par rapport à un autre, arrive avant ce dernier à destination.

Le protocole TCP, inventé en 1981 (voir J. Postel [1]) , est un protocole qui principalement assure l'arrivée des paquets envoyés à bonne destination et qui gère la congestion du réseau qui peut être engendrée suite à une émission trop intensive d'informations sur celui-ci. C'est également lui qui se charge de la numérotation des paquets. Il se sert de mécanismes basés sur des 'accusés de réception' pour s'assurer de l'arrivée des paquets à bonne destination et de mécanismes basés sur des 'fenêtres' (voir Clark, D. David [2]) de transmission et de réception pour gérer les congestions.

1.1.2 Définitions

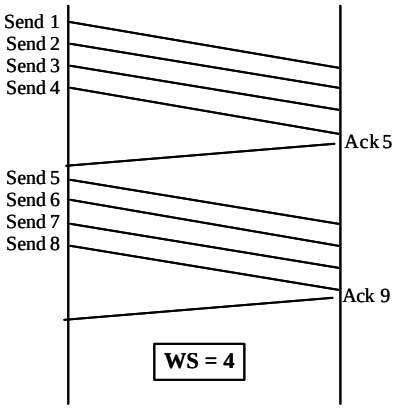
Nous présentons ici quelques définitions de base des termes rencontrés fréquemment dans la littérature et également présents dans ce travail.

L'accusé de réception est une information émise depuis le récepteur vers l'émetteur pour lui signaler que le paquet envoyé lui est bien parvenu. Il s'agit du mécanisme au cœur du fonctionnement de TCP. Les mécanismes de contrôle de congestion font varier leurs paramètres, dont la fenêtre, sur base de la réception de ceux-ci. Une mauvaise utilisation de ce paramètre peut engendrer des défauts de comportement comme par exemple le syndrome de la fenêtre stupide [2]. On notera aussi que si l'émetteur ne reçoit pas un accusé de réception pour un des paquets envoyés, signe que le paquet est perdu, il devra le renvoyer (voir fig 1.2). Cette prise de décision de renvoyer le paquet, s'effectue après déclenchement d'un 'watchdog' ou d'un timer. Après avoir attendu un certain temps, l'émetteur estime que le paquet a du arriver à destination et si durant ce temps aucun accusé de réception n'est parvenu, il faut retransmettre l'information.



[Fig 1.2]

La fenêtre d'émission d'un émetteur correspond au nombre de paquets que l'on peut envoyer sur un canal de transmission avant de devoir recevoir obligatoirement un accusé de réception pour poursuivre la suite de la transmission (voir fig 1.3).



[Fig 1.3]

La fenêtre de réception, au niveau du récepteur, précise le nombre maximum d'informations que le récepteur est capable de recevoir. Il s'agit en quelques sortes de la taille du buffer en entrée du récepteur. Si le récepteur reçoit plus de paquets que cette limite, il n'en tiendra pas compte et ces paquets seront perdus.

Le round trip time: Il s'agit du temps mis pour envoyer un paquet et recevoir son accusé de réception (voir H. Molly et al. [3] et P. Karn, C. Partridge[7]). Nous parlons de Round Trip Time de fenêtre pour désigner le temps entre lequel le premier paquet d'une fenêtre est émis et celui où l'accusé de réception du dernier paquet correspondant à cette fenêtre est reçu.

Les segments émis par TCP sont de taille fixée par le minimum entre le SMSS (Sender Maximum Segment Size) et le RMSS (Receiver Maximum Segment Size). Cette valeur peut être basée sur l'unité de transmission maximum du réseau (MTU) ou une de ses fonctions (voir M. Allman et al [5]).

1.1.3 Remarques d'implémentation

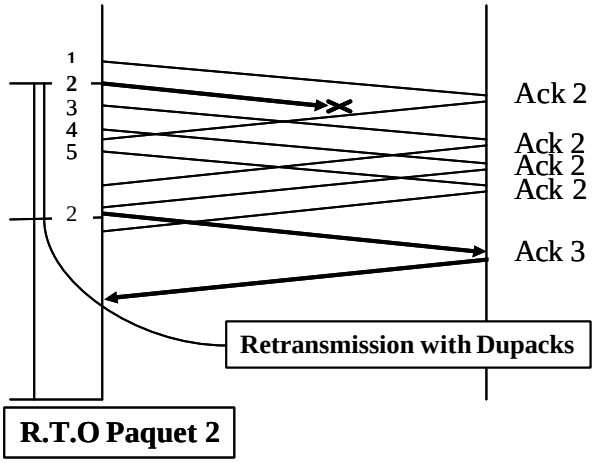
Si une version archaïque de TCP est construite sur base des critères qui viennent d'être définis, et sans aucun autre mécanisme supplémentaire, on arrive à un modèle de transmission dans lequel TCP ne fait que peu pour minimiser les congestions du réseau (voir K. Fall, S. Floyd [6]). Il s'agit en fait d'un système cadencé uniquement par les arrivées des accusés de réception sans aucune autre mesure.

Notons que par les mécanismes de TCP plus de données sont envoyées sur le réseau que ce qu'il faudrait pour que le taux de transmission corresponde exactement à la bande passante disponible du réseau. Ceci a pour conséquence de remplir les buffers intermédiaires mais a surtout pour but de pouvoir répondre assez rapidement aux transitoires de libération de bande passante du réseau. Un juste partage entre le nombre de données en excès par rapport au nombre de données nécessaires permet d'éviter l'entrée en congestion trop fréquente.

Il est bon de signaler l'existence du protocole UDP également situé au niveau de la couche transport mais pour lequel il n'y a pas d'algorithmes de contrôle de congestion ni de mécanisme assurant une certitude de la réception de l'information. Ce protocole permet dès lors des transmissions plus rapides. Il est donc pratique pour l'émission de données multimédia, puisqu'il est moins élaboré techniquement. Cependant, il n'est pas recommandé pour des transmissions dont on espère une fiabilité totale au niveau de la réception. La rapidité va au détriment de la fiabilité de la transmission.

1.1.4 Les types d'accusés de réception

Les récepteurs peuvent envoyer deux types d'accusé de réception cumulatifs. Ils envoient les acks positifs (ACKs) pour les segments qui sont reçus correctement et dans l'ordre et ils envoient des DUPACKs pour les segments qui sont reçus correctement mais pas dans le bon ordre. Un DUPACK accuse réception du même numéro de segment que celui accusé par le dernier ACK reçu (voir fig 1.4). Un DUPACK ne communique donc aucune information explicite à l'émetteur à propos des segments qui ont été reçu correctement. Il est à remarquer qu'un accusé de réception informe l'émetteur du numéro de segment que le récepteur espère attendre après avoir envoyé cet accusé de réception. Lors de la mise en route d'une communication entre l'émetteur et le récepteur il y a une phase de synchronisation (SYNC) par laquelle ils s'échangent des informations sur les numéros de segment par lesquels ils vont débuter la transmission.



[Fig 1.4]

Un récepteur peut retarder les accusés de réception qu'il envoi (voir R. Braden [13]). Cette fonctionalité est celle des 'Delayed Ack'. Le récepteur ne peut cependant pas les retarder trop exagérément. Un ACK doit être généré au moins tous les seconds segments de taille maximum reçus et doit être généré endéans les 500 ms qui succèdent la réception d'un paquet encore non accusé (voir [5]). Cette fonctionalité est à prendre en considération au niveau des algorithmes d'évaluation de la bande passante qui se basent sur la réception des ACKs pour leurs calculs. Il faut remarquer que si des delayed Acks sont utilisés, les algorithmes comme le Slow Start qui incrémente la fenêtre de congestion sur base de la réception des ACKs, l'augmentera plus lentement, et en principe deux fois plus lentement. En effet, un accusé de réception serait uniquement renvoyé tous les deux paquets reçus.

1.2 Quelques versions du protocole TCP

Dans cette partie nous donnons un aperçu des versions de TCP qu'il nous a paru bon de présenter. Il est évident que ceci ne représente pas une liste exhaustive.

1.2.1 TCP Tahoe

Jacobson (voir V. Jacobson [4]) donne les premières améliorations de TCP au niveau de ses algorithmes de contrôle de congestion. Il définit un nouveau concept important qui est la fenêtre de congestion sur laquelle se base deux algorithmes principaux de TCP: Les algorithmes de Slow Start et de Congestion Avoidance. Cette version de TCP porte le nom de TCP Tahoe et date de 1988. Cette version n'est plus utilisée aujourd'hui.

La fenêtre de congestion (Cwnd) (voir [4]) est une sous fenêtre mobile de la fenêtre de transmission. Sa taille varie en fonction des algorithmes de Slow Start (SS) et de Congestion Avoidance (CA). Son principe de fonctionnement est identique au principe de la fenêtre de transmission. A savoir: lémetteur n'envoi pas plus de paquets sur le réseau que ce que la fenêtre de congestion ne le permet. L'émetteur choisira cependant le minimum entre la taille de la fenêtre de congestion et la fenêtre d'émission.

L'algorithme de Slow Start (voir [4]) a pour but d'avoir une estimation de la limite de débit au-delà de laquelle il risque d'y avoir une congestion. Pour arriver à cette estimation, la taille de la fenêtre de congestion est augmentée progressivement jusqu'à observer une perte de paquet. Cette croissance de la taille de la fenêtre de congestion est de type

exponentielle. Au moment ou un accusé de réception n'est pas retourné, la taille de la fenêtre est enregistrée dans une variable de seuil de slow start (SStresh). La phase de Slow Start est un mécanisme pour lequel la fenêtre de congestion est initialisée à une taille de 1 paquet. On double ensuite la taille de la fenêtre lors de la réception d'un nombre d'accusé de réception correspondant à la taille de la fenêtre. La valeur optimale du seuil de slow start est celle qui correspond au nombre de segments en transit dans le conduit d'une capacité égale à la bande passante disponible (voir J.C. Hoe [25]).

L'algorithme de Congestion Avoidance (voir [4]) est déclenché après la première perte d'un paquet. A son initialisation, la fenêtre de congestion est réduite de moitié (SStresh/2) et l'algorithme permet à l'émetteur d'augmenter de façon plus douce le débit d'émission pour ne pas ré-entrer à nouveau en phase de congestion et devoir retransmettre l'ensemble des paquets perdus.

Le temps endéans lequel un accusé de réception doit être perçu pour un paquet envoyé ('Time Out' ou 'T.O.'), n'est pas une variable constante dans le temps. En effet, la dynamique du réseau force cette variable à être recalculée périodiquement puisque le réseau est mis à contribution par plusieurs utilisateurs initiant et finissant des transactions à des moments aléatoires. Les T.O. sont adaptés en fonctions de l'état de congestion du réseau (voir V. Jacobson et al [8]). Cela permet d'éviter le phénomène de "Congestion Collapse" (voir J. Nagle [9]). Ce temps se calcule sur base d'échantillonnages de la mesure du RTT et sa variance (voir [7]). Mais le calcul du RTT ne se révèle pas évident comme peuvent le montrer R. Jain dans [10], P. Karn dans [11] et L. Zhang dans [12]. Il n'est pas possible de calculer des mesures de RTT si l'on tient compte des temps issus des paquets retransmis suite à des pertes de données sur le réseau. C'est le principe de l'algorithme de Karn (voir P. Karn, C. Partridge [7]). Mais des solutions existent comme le montre Jacobson (voir [8]) en modifiant des options dans le champs de TCP en y incluant une emprunte temporelle (Timestamp). Le récepteur reproduit cette information dans les Acks. Par un calcul issu sur base de ce Timestamp, le récepteur possède une information plus précise pour chaque segment accusé. C'est le mécanisme du RTTM (Round-Trip Time Measurement).

1.2.2 TCP Reno

Jacobson propose également dans [4] un autre algorithme qui rend la transmission plus efficace. Cette transmission est qualifiée de plus pro-active lors de la détection d'une perte lors de la transmission. En effet, dans la version Tahoe, ce n'est qu'après apparition du time-out que l'algorithme se rend compte d'une perte de paquet. La solution proposée dans l'algorithme de 'Fast Retransmit' est de partir sur cette constatation et de s'apercevoir d'une perte de paquet avant le time-out. L'algorithme de Fast Retransmit préconise que dès réception d'un nombre n de DUPACKs, un paquet a pu se perdre. Après réception de ce nombre de DUPACK, on prend les dispositions requises. N est généralement égal à 3 (voir fig 1.4).

Une perte de paquet, signe d'une congestion, déclenche un algorithme pour lequel on retransmet les paquets perdus. Cette retransmission s'effectue avec une fenêtre de congestion unitaire, même si la perte s'est produite avec une taille de fenêtre élevée. Ceci n'est pas efficace puisque la congestion s'est produite pour une cwnd supérieure à cette valeur unitaire, et il n'est pas utile de la réduire autant. Le compromis proposé par Jacobson est de re-initier une transmission avec une valeur de fenêtre de congestion

réduite de moitié et pour laquelle on continue la transmission en congestion avoidance. Il s'agit de l'algorithme de Faster Recovery.

L'ensemble de ces derniers apports à la version de TCP Tahoe forme ce que l'on appelle la version de TCP Reno.

Cette version garde l'inconvénient d'être peu rapide pour la réémission de données suite à un Time Out à cause d'une granularité peu fine utilisée dans le système d'horloge pour le calcul des RTT (voir V. Jacobson [14]).

1.2.3 TCP Vegas

La version de TCP Vegas (voir L.S. Brakmo et al [16]) change le procédé par lequel elle fait varier les tailles des fenêtres par rapport aux autres versions de TCP. Son principe est d'évaluer la taille des buffers en entrée des routeurs et d'en observer leur évolution sur base d'informations calculées à partir des mesures de RTT. Un algorithme fait varier la fenêtre à partir d'une comparaison du taux de transmission attendu par rapport au taux de transmission en cours. Ce n'est donc plus un système régulé uniquement par la réception des accusés de réception.

Vegas calcule le taux de transmission attendu par :

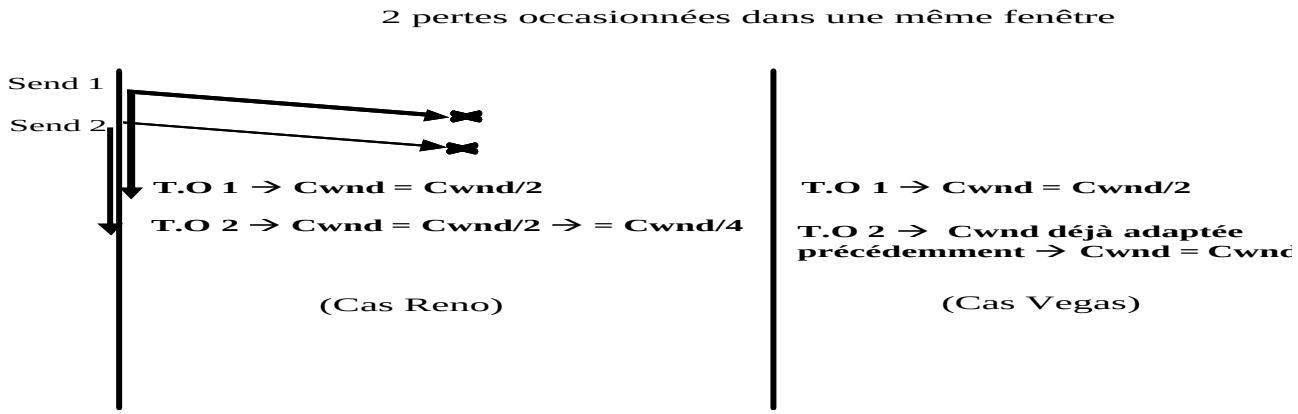
Taille de la fenêtre en cours / RTT

où le RTT est calculé sur base d'un segment envoyé dans le réseau sans qu'il y ait congestion.

Pour obtenir le taux actuel de transmission, il faut compter le nombre d'informations envoyées sur le réseau entre le temps pour lequel un paquet est émis et le temps pour lequel son accusé de réception est reçu.

Vegas ajuste son taux de transmission pour le garder inférieur au taux de transmission attendu (voir S. Low et al [23]). Ce taux de transmission, Vegas l'ajustera en proportion avec le délai de propagation et avec le délai du aux files d'attente (voir R. Wang et al [28]).

Reno ne permettait pas d'évaluation précise des tailles des files sur le chemin de transmission car il se base sur des calculs de RTT cadencés par une horloge peu fine, fournissant des impulsions par multiples de 500 ms. Dans le cas de Vegas, l'algorithme enregistre les temps auxquels un segment est émis et pour lequel l'accusé de réception correspondant est reçu. Ce sont donc des estimations beaucoup plus précises. Cette manière de faire dans Vegas donne une méthode de perception plus rapide des paquets perdus et donc de la retransmission après un DUPACK (on n'attend plus 3 DUPACKs). Il ne donne pas lieu comme Reno à plusieurs réductions de taille de fenêtre pour des pertes occasionnées dans une même fenêtre de transmission. Dans Vegas, à l'inverse de Reno, l'algorithme ne donne pas lieu à une diminution de taille de fenêtre pour une perte remarquée après cette diminution, mais pour un paquet envoyé avant cette diminution de taille de fenêtre. En effet, il n'est pas sûr que pour la dernière taille de fenêtre obtenue après réduction, il y ait encore congestion (voir fig 1.5).



[Fig 1.5]

Vegas propose également un mécanisme de détection de congestion au sein de la phase de Slow Start. Il n'augmente les fenêtres que lorsqu'il n'y a pas de modification importante entre les RTT mesurés par rapport aux RTT précédents.

L'avantage mis en évidence dans l'algorithme de Vegas, est d'obtenir un taux de transmission plus élevé avec moins de retransmissions.

Cependant, des problèmes d'équité ont étés décelés en fonction de l'implémentation de l'algorithme. Celui-ci peut baser ses mesures en nombre de paquets transmis par RTT ou par seconde (voir S. Low [23], J. Mo et al [22], G. Hasegawa et al [24]).

Si Vegas partage une connexion avec Reno, Reno sera plus agressif et aura tendance à occuper plus de place dans les buffers, ce qui aura pour conséquence que Vegas interprétera une augmentation de délai dans les buffers et donc un signe de congestion. De même que les connexion Vegas entre elles mêmes peuvent être injustes. Une connexion qui démarre plus tard qu'une autre peut observer des RTT plus importants et donc obtient une Cwnd plus faible. Or il faut partager la bande passante équitablement entre les différents utilisateurs. L'analyse de l'équité est assez complexe et ne sera pas abordée dans ce travail.

1.2.4 TCP New Reno

Il s'agit d'une petite modification apportée au protocole Reno (voir K. Fall, S. Floyd [40]) qui lui permet d'éviter de devoir attendre pour qu'un déclenchement de retransmission se produise lorsque plusieurs paquets sont perdus au sein d'une même fenêtre de congestion. Les changements concernent l'émetteur lorsqu'il reçoit un accusé partiel pour seulement une partie des segments perdus qu'il a retransmis au cours de l'algorithme de Fast Recovery. Avec New-Reno, à la différence de Reno, un accusé de réception partiel ne fait pas quitter l'algorithme de Fast Recovery. Un accusé partiel permet de se rendre compte que parmi les paquets retransmis une première fois, certains ont à nouveau été perdus. Dès lors, cette version n'attend pas un Time Out pour retransmettre ces paquets perdus.

La version de TCP New Reno est parmi les versions les plus utilisées de TCP actuellement.

1.2.5 TCP Sack

Dans [35] & [36] est proposé une méthode où est apportée une modification au contenu des accusés de réception. Alors que les accusés de réception ne permettaient pas de savoir avec certitude le numéro du ou des segments qui étaient perdus lors d'une congestion du réseau. TCP Sack apporte une solution à ce problème. Sack, utilise un champ particulier dans la donnée d'ACK.

Les accusés contiennent des informations 'bloc' où chacun des blocs indique une discontinuité entre les paquets réceptionnés. Les blocs contiennent ainsi l'information qui permet de savoir quel est, ou quels sont les bons paquets à retransmettre.

Comme il s'agit d'une option de TCP, l'algorithme implémentant SACK inclut les versions de TCP Reno et New Reno.

Il est fortement conseillé d'utiliser le mécanisme SACK car engendre des accroissements de performance importants.

1.2.6 TCP Byte counting

Quelques problèmes peuvent être montrés lorsque l'algorithme de 'Delayed Acks' est utilisé. En effet, les algorithmes peuvent ne pas être suffisamment bien cadencés par l'arrivée de œux-ci puisqu'ils sont retardés. Les algorithmes ne reçoivent pas d'accusés pour tous les paquets transmis mais reçoivent des accusés pour plusieurs paquets groupés. Dès lors, s'ils se basent sur le nombre de d'accusés de reçus pour évaluer le nombre de paquets transmis, il peut y avoir des confusions.

Pour gagner en performance face aux pertes occasionnées par les retardements des accusés, M. Allman (voir [37]) propose une solution par laquelle le récepteur regarde, au niveau des accusés, le nombre de données effectivement reçues, plutôt que de se baser sur un compteur d'accusés de réception. Cependant cette manière de faire rend cette méthodologie trop agressive dans certaines situations (voir [39]) Une proposition d'adaptation de cet algorithme est donnée dans [39]. L''Adapted Byte counting' y est présenté pour résoudre le problème.

1.2.7 TCP Santa Cruz

Cette version de TCP Santa Cruz (voir C. Parsa, J.J. Garcia-Luna-Aceves [38][39]) est adaptée dans des conditions où les vitesses de propagation dans les réseaux sont asymétriques. En effet, les réseaux d'aujourd'hui sont dissymétriques dans le sens ou la bande passante descendante est généralement plus élevée que la bande passante montante (ex: ADSL). [39] Montre également que TCP Santa cruz est particulièrement bien adaptée aux télécommunications transitant dans des milieux sans-fil. Il s'agit également d'un protocole qui ne fait pas appel au niveau de la couche lien du modèle OSI, ni de proxy pour tenir compte des pertes dues au milieu de propagation.

La méthode est basée sur le délai de propagation de l'information pour arriver jusqu'à sa destination plutôt que sur le RTT des données. De plus elle identifie l'origine de la congestion (soit sur le chemin ascendant, soit le chemin descendant). Un calcul du délai relatif entre les paquets est utilisé dans l'algorithme de contrôle de congestion. La différence par rapport à Vegas qui repose sur les mêmes œonsidérations est que Vegas

regarde les variations dans le RTT. Le délai relatif entre les paquets est mesuré sur base d'une empreinte temporelle inscrite au niveau des accusés de réception par le récepteur. En effectuant une comparaison entre l'évolution des délais en fonction du temps on peut évaluer si la taille des files d'attente intermédiaires sont occupées à croître ou à décroître. La fenêtre de congestion est alors adaptée sur base de calculs sur ces valeurs.

1.2.8 TCP Fast Start

TCP Fast Start (voir M. Allwam et al [33]) se propose d'initialiser la fenêtre de congestion à une valeur supérieure à l'unité pour le démarrage de l'algorithme de Slow Start. Celui-ci mettra dès lors moins de temps (RTT) pour atteindre des fenêtre de congestion de taille élevée. Cet algorithme est dès lors pratique dans les connexions présentant de grands délais et des produits 'bande passante – délai' élevé comme par exemple les connexions satellites.

1.2.9 TCP Smooth Start

La version proposée par TCP Smooth Start (voir H. Wand et al [34]) se fonde sur les problèmes suivants:

- il faut trop de RTT pour que Slow Start arrive à une valeur suffisamment élevée
- lorsque les fenêtres de congestion sont 'grandes', Slow Start augmente trop rapidement le débit et fait entrer le réseau en congestion. Ceci est du au fait que SS n'a pas connaissance de l'environnement sur lequel il enverra les données et que par conséquent, il est initialisé en début de connexion avec des valeurs par défaut (65535 bytes pour le SStresh)

Une solution proposée pour éviter le premier problème est donnée dans TCP Fast Start (voir [33]).

La solution proposée par Smooth Start est de prévoir une transition plus lisse entre la phase de Congestion avoidance et la phase de Slow Start en modifiant la manière dont l'émetteur atteint la limite de Slow Start. Une nouvelle variable SmsTresh est introduite et correspond à une valeur inférieure au SStresh: SStresh/(2^d). On désire que durant l'intervalle SmsTresh-SStresh, l'algorithme continue à augmenter la taille de la fenêtre de congestion de manière exponentielle mais à une cadence moins élevée en introduisant plus de variation entre deux tailles de fenêtres qu'aurait donné TCP Reno.

1.3 Incrémentation / décrémentation de la fenêtre de congestion

L'incrémentation et la décrémentation de la fenêtre de congestion lors de la détection de paquet perdu sont des conséquences de l'interprétation d'une congestion. Les algorithmes utilisés dans les versions basées sur Reno, pour ces gestion de contrôle de congestion, font partie des algorithmes dit AIMD : Additive Increase, Multiplicative Decrease (voir D. Chiu, R. Jain [17]).

L'objectif de tout algorithmes de contrôle de congestion est de tenter de garder le réseau en deçà d'une zone pour laquelle l'augmentation en charge n'entraîne pas une chute brutale du taux de transmission et pour laquelle on garde des temps de réponse non excessifs. Alors que l'objectif des algorithmes d'évitement de congestion est d'autoriser

l'utilisateur à augmenter son débit pour autant qu'il n'y a pas congestion si non en sortir au plus vite. On est donc en présence d'algorithmes qui forceront des oscillations dans les débits de transmission des sources.

[17][18][19] montrent que pour obtenir une convergence et une équité, les algorithmes devraient être du type AIMD de types linéaires.

Bien que l'on montre que les fenêtres de congestion soient diminuées de moitié pour sortir d'une congestion, il existe des solutions pour lesquelles ce facteur réductif est différent. Les objectifs à garder en vue pour ces algorithmes sont d'être efficaces et équitables vis-à-vis des autres utilisateurs, qu'ils soient distribués dans le réseau (Il est difficile d'avoir un contrôle central) et qu'ils présentent une convergence vers une solution adéquate à partir d'un état quelconque et avec une vitesse assez rapide. Ainsi Z.Wang et J. Crowcroft (voir [20]) proposent l'algorithme DUAL dans lequel la fenêtre de congestion est réduite d'un facteur 8 en fonction des RTT mesurés tous les round-trip. De même que R. Jain (voir [21]) propose une solution TRI-S dans laquelle la fenêtre est également réduite d'un facteur 8.

1.4 Tableau Récapitulatif

Nom	*Caractéristiques*		
Tahoe	Slow Start + Congestion Avoidance Fast Retransmit		
Reno	Tahoe + Fast Recovery	⇨ Réactif	⇨ Mauvaise granularité ⇨ Plus d'une réduction possible de taille de fenêtre pour un RTT de fenêtre
New Reno			
Vegas	⇨ Extension de l'algorithme de retransmission de Reno (Fast Retransmit) ⇨ Modification du Slow start pour incorporer un début de détection de congestion	⇨ Plus rapide que Fast Retransmit ⇨ Granularité précise ⇨ Pro-actif	⇨ Equité
Westwood	⇨ Estimation de type 'Fair Share' de la bande passante	⇨ Meilleure utilisation en cas de pertes aléatoires ⇨ Gain significatif en cas de produit BP-délai élevé ⇨ équitable	
GFR	⇨ Sur base de BWE adaptation de SS		
FR	⇨ Sur base de BWE adaptation de CA		
Net Reno	⇨ Réduit retransmission T.O.		
Smooth Start	⇨ Plus faible agressivité de TCP Reno par élongation de la période de Slow Start	⇨ Réduction du nombre de perte de paquets au sein d'une Cwnd	
Fast Start	⇨ Cwnd(0) = 2	⇨ efficacité pour produit BP,delai élevé	
Byte counting	=> utilisation de la quantité de données accusées pour Slow Start plutôt qu'un compteur d'accusés	=> Efficacité pour les délayed Ack	
Santa Cruz	Algorithmes basés sur le temps de transmission de l'information et non du RTT	Mise à jour de l'origine de la congestion (ascendant, descendant) Appliqué au réseaux asymétriques	

Chapitre 2

TCP Westwood

2.1 Introduction

La version de TCP que nous analysons dans ce chapitre est présentée actuellement comme une version de TCP efficace pour le transport d'informations pouvant circuler sur des réseaux sans fil. Un ensemble d'analyses et de simulations montrent l'efficacité de ce protocole (voir R. Wang et al [30] et M. Gerla et al [31]). Celui-ci se base sur une estimation de la bande passante disponible pour régler les mouvements de la fenêtre de congestion en adaptant les seuils présents dans les algorithmes de traitement de congestion sur base de ces estimations. TCP Westwood est fondé sur l'algorithme de Faster Recovery (voir C. Casetti [26]) qui réalise de meilleures performances de débit. M. Gerla (voir [31]) montre également que ce protocole est adéquat pour les réseaux profitant des fonctionnalités de balancement de charge en partageant le trafic issu d'une connexion, sur les divers chemins qui mènent à la destination.

Un algorithme d'estimation de bande passante placé au niveau du récepteur donnera de meilleurs résultats que s'il est placé au niveau de l'émetteur. Les réseaux asymétriques, par exemple, peuvent engendrer des pertes au niveau des Ack et fausser les mesures au niveau de l'émetteur. Cette éventualité n'est pas envisagée ici.

2.2 Estimation de la bande passante disponible

Introduisons avant tout quelques concepts de mesures que nous pouvons associer à un lien physique:

- La bande passante résiduelle: la bande passante, libre, qui n'est pas utilisée par les flux de données.
- La capacité : la bande passante physique du lien
- Le juste partage de la bande passante: correspondrait au partage de la capacité par le nombre de flots TCP.

Plusieurs modèles d'estimation de bande passante ont étés développés jusqu'à présent comme le montrent les solutions de "Packet Tailgating", [XX1 Kevin Lai, Mary Baker] "Pathchar" [jacobson 1997, Downey 1999], "B|C Probe" [Downey 1999] et "Packet Dispersion Measurements" [Dovrolis, Ramanathan Moore 2001].

Pour évaluer le débit disponible, TCP Westwood travaille avec des informations entre les interlocuteurs de bout à bout. Pour l'instant TCP se base uniquement sur les informations implicites qu'il détecte comme le RTT, les DUPACKs et les Timeout pour effectuer les évaluations de congestion puisqu'il ne peut recevoir que difficilement des informations explicites des routeurs par lesquels transite l'information envoyée.

Le principe sur lequel repose l'évaluation de TCP W, et calqué sur Faster Recovery, est basé sur la mesure du taux de retournement d'accusés de réception (voir S. Mascolo et al [15]). Pour cela, l'émetteur doit déduire le nombre de données envoyées vers le récepteur

en fonction du temps. Lorsqu'un ACK parvient à la source, cela exprime qu'une certaine quantité d'information est arrivée à destination. En effectuant une simple moyenne du nombre de données envoyées en fonction du temps, et s'il n'y a pas de pertes, on obtient une estimation de la bande passante utilisée par la source.

Si les paquets transmis par la source sont de tailles variables, et qu'une perte survient et que celle-ci est annoncée par la présence de DUPACKs, il n'est pas possible de savoir quel paquet s'est perdu. Il est alors utile d'effectuer des échantillonnages de mesures de taille de paquets pour pouvoir évaluer la quantité d'information perdue. Par la suite on gardera, pour la facilité des simulations, des tailles de paquets constantes.

Un autre moyen d'obtenir une estimation de la bande passante consiste à utiliser la méthode du 'Packet – Pair Flow Control' (voir C. Dovrolis et al [29]) par lequel le taux de trafic réalisé par une source est mesuré. Le principe est basé sur la transmission de deux paquets l'un directement après l'autre et à observer le temps séparant les accusés de réception reçus. Cet écart de temps est proportionnel à la bande passante du lien le moins rapide si l'algorithme de Round Robin est utilisé dans les serveurs.

2.3 Faster Recovery

On estime le taux d'arrivée des ACKS qui est proportionnel au taux de données effectivement transmises à la destination. Ce taux peut être inférieur au taux de données effectivement émises. Le taux est estimé au moyen d'un 'moyennage' exponentiel. Interviennent dans les calculs également les DUPACKs qui sont, en plus d'être signe d'un désordre, également signe d'une arrivée de paquets. Sont alors calculées la 'bande passante estimée par échantillonnage' (sample_BSE) et la 'bande passante estimée' (BWE) sur base de la sample_BSE [26].

When ACK: [
    Sample_BSE = pkt_size*8/(now-lastacktime)
    BWE = BWE*alpha + sample_BWE * (1-alpha)
]

Ces valeurs sont utilisées pour les algorithmes de contrôle de congestion pour former l'algorithme de FAster Recovery.

3 DUPACKs ➔ sstresh = (BWE * RTTmin)/a
Cwin = sstresh

Time Out ➔ sstresh = (BWE * RTTmin)/a
Cwin = 1

Où *a* est un facteur correctif.

2.4 Gradual Faster Recovery

Une idée est également proposée dans la littérature (voir C. Casetti et al [26]) afin de récupérer une taille de fenêtre proportionnelle à la bande passante disponible en phase de congestion avoidance. Dans ce cas, la bande passante disponible est estimée périodiquement (toutes les 500 ms) et si il y a moyen d'augmenter le débit en cours par rapport à cette estimation, on effectue une modification du SStresh.

L'algorithme qui s'adapte pour les versions de TCP basées sur Reno et Tahoe est:

```
If (Cwin > sstresh) && (Cwin < BWE*RTT_min)
        Sstresh = sstresh + (BWE*RTT_min – Sstresh)/2
```

Il permet d'éviter d'épargner un certains nombre de RTT pour atteindre une fenêtre de congestion plus élevée en phase de Congestion Avoidance.

2.5 Algorithme Westwood

Comme il vient de l'être présenté, Westwood se base sur FR et utilise un filtre pour estimer la bande passante. Il s'agit d'une estimation de type 'Fair Share' (voir R. Wang et al [28]). L'algorithme général mis en place est le suivant:

```
When Ack:
    Sample_BWE[k] = (acked * pkt_size*8)/(now-lastacktime)
    BWE[k] = (19/21) * BWE[k-1] + (1/21) * (sample_bwe[k] + sample_BWE[k-1])
```

Les mesures récupérées par échantillonnage sont filtrées à l'aide d'un filtre passe bas à temps discret de type 'exponentiel' [30].

Il faut cependant se méfier de l'interprétation des accusés de réception qui parviennent à l'émetteur. Les accusés de réception cumulatifs perçus après une perte importante de paquets peuvent fausser considérablement l'interprétation en donnant une évaluation soudainement plus brutale de Sample_BWE[k] comme le précise S. Mascolo et al (voir [15]). Dès lors l'algorithme précédent est modifié et [15] présente les procédures qui tiennent compte des facteurs correctifs à apporter.

Sur base des estimations précédentes, et toujours calqué sur Faster Recovery, Westwood adapte le SStresh et sa Cwnd dès apparition d'une congestion pour approcher SStresh du produit 'Délai - Bande passante'.

```
When Dupacks:
        If (C.A. phase):
                SStresh = f1(BWE*RTTmin)
                Cwin=SStresh
        If (S.S phase):
                SStresh=f2(BWE*RTTmin)
                If Cwin>SStresh
                        Cwin=SStresh
```

De la même manière pour un Time Out, le SStresh est adapté en fonction du produit 'BWE – RTTmin' mais au lieu de repartir en Slow Start avec une fenêtre unitaire, comme l'aurait fait Reno, l'algorithme démarre avec une fenêtre de congestion également fonction du produit 'BWE – RTTmin'.

Pour améliorer l'efficacité et l'équité de TCP Westwood, R. Wang (voir [30]) propose de calculer une 'Rate Estimation (RE)' sur base de la 'Bandwith Estimation (BWE)' pour former une 'Combined Rate and Bandwith Estimation (CRB)'. La BWE peut dans

certaines conditions surestimer le partage équitable d'une connexion et la RE le sous-estimer (voir [28]). Un algorithme d' 'Adaptative Bandwith Share Estimation' est également proposé par R. Wang dans [28]. Celui-ci consiste en une adaptation continue du taux de transmission au niveau de congestion.

Chapitre 3

Problème et résolution

Dans cette partie nous présentons un problème d'évolution du débit d'une source suite à l'observation d'une fin de session d'une autre source. Les sources réalisent des transferts de données au moyen de sessions ftp. Une session ftp est une période de temps au cours de laquelle un émetteur transmet des données vers un récepteur. La transmission s'effectue au moyen du protocole TCP. Les versions du protocole TCP que nous allons analyser sont TCP Reno et TCP Westwood.

Sur base d'observations, nous proposons une solution permettant un accroissement sensible du rendement de transmission de la source encore active suite à l'observation de la fin de session. Les performances de la solution proposée seront analysées et ensuite discutées.

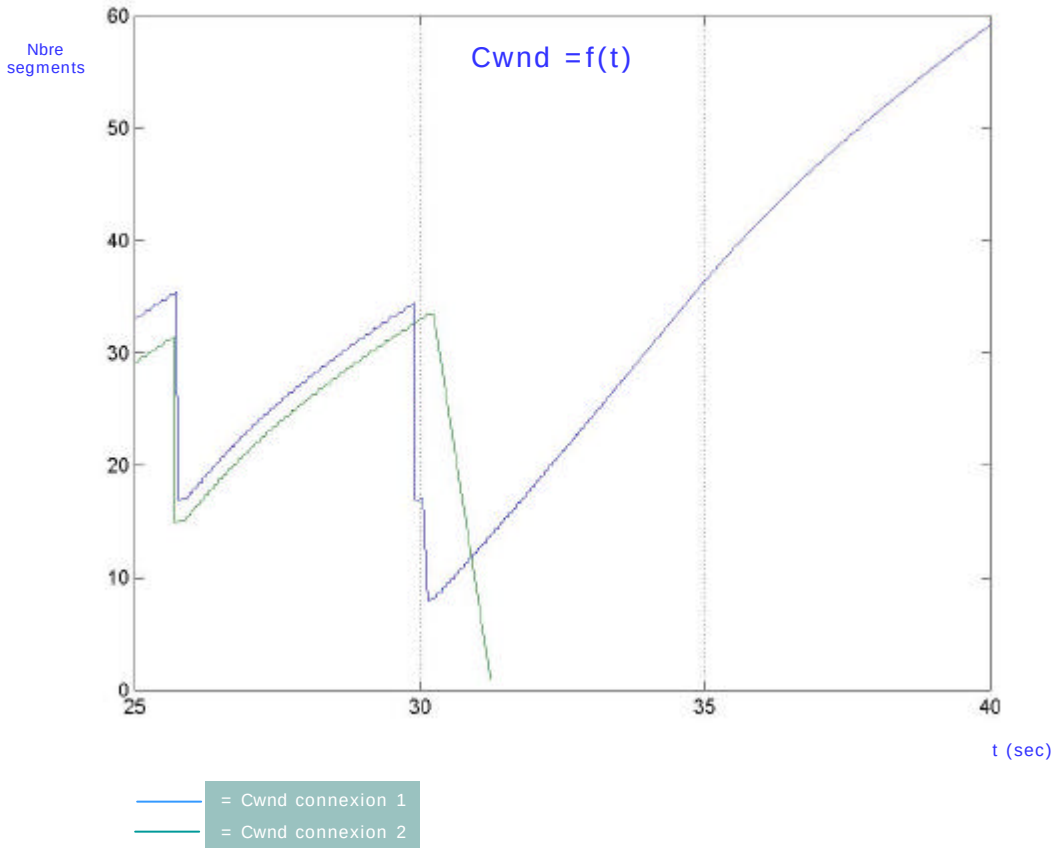
3.1 Définition du problème

Nous désirons analyser la dynamique de récupération de la bande passante libérée lors de la fin de service d'une session ftp. Au cours d'une session ftp, l'émetteur dispose toujours d'informations à envoyer vers le récepteur. L'émetteur régule donc son taux de transmission uniquement sur base des règles édictées par la version du protocole TCP utilisée.

Nous désignerons par S1 et S2 les sources réalisant les transmissions. Soit le cas dans lequel la source S1 continue son transfert d'information après la fin de session de la source S2. Nous ne formulons pas d'hypothèse sur la manière dont ont été initiés les transferts, mais nous supposons le système en régime.

La situation dans laquelle une mauvaise utilisation de la bande passante risque de se produire est celle où la source S1 est en phase de congestion avoidance alors que la session initiée par S2 prend fin. Nous montrerons que dans ce cas, la source S1 peut se retrouver dans une phase de congestion avoidance assez longue et dès lors n'augmente son débit de transmission que lentement. [voir figure 3.1]

En effet, puisqu'une session prend fin, il y a une libération de bande passante. Or, si la source S1 est en phase de congestion avoidance, elle recherche le niveau du débit au delà duquel le réseau entrera en congestion. La particularité de la phase de congestion avoidance est de n'augmenter que doucement la fenêtre de congestion. Dès lors, le niveau de congestion sera d'autant plus lent à atteindre que la libération de bande passante est grande. Il y a donc lieu d'apporter une amélioration pour que la source se rende compte plus rapidement de cette libération de bande passante et accélère la manière dont elle atteindra le seuil de congestion.



[Fig 3.1]

3.2 Plan des étapes de l'analyse

L'analyse et la résolution du problème se dérouleront en plusieurs étapes. Nous reprenons ici la structure qui nous permettra d'aboutir à nos conclusions.

1. Création d'un modèle
2. Mise en place du modèle sous un simulateur de réseau et génération de simulations.
3. Elaboration d'une solution au problème sur base d'observations
4. Intégration de la solution au sein du simulateur
5. Réalisation de simulations et comparaison entre les situations avec et sans apport de la solution. Ces simulations seront:

I. Cas sans apport de solution complémentaire

A: Dans le cas d'un réseau sans perte de donnée dues au support de transmission (une installation filaire)

1: Cas où les sources émettent avec le protocole TCP Reno
2: Cas où les sources émettent avec le protocole TCP Westwood

B: dans le cas d'un réseau présentant des pertes de données dues au support de transmission (une installation sans fil)

1: Cas où les sources émettent avec le protocole TCP Reno
2: Cas où les sources émettent avec le protocole TCP Westwood

II. Cas avec apport de solution complémentaire

A: Dans le cas d'un réseau sans perte de donnée dues au support de transmission (une installation filaire)

1: Cas où les sources émettent avec le protocole TCP Reno
2: Cas où les sources émettent avec le protocole TCP Westwood

B: dans le cas d'un réseau présentant des pertes de données dues au support de transmission (une installation sans fil)

1: Cas où les sources émettent avec le protocole TCP Reno
2: Cas où les sources émettent avec le protocole TCP Westwood

6. Mise en place d'outils de mesure de performance
7. Analyse des résultats

3.3 Mise en place du modèle

Pour effectuer les simulations, nous utiliserons le simulateur ns [voir 46]. NS permet de simuler l'ensemble des événements qui se déroulent sur un réseau en y modélisant une topologie et des objets. Les objets sont des nœuds du réseau, des sources et des récepteurs qui implémentent certains protocoles. Il est alors possible d'extraire les événements auxquels:

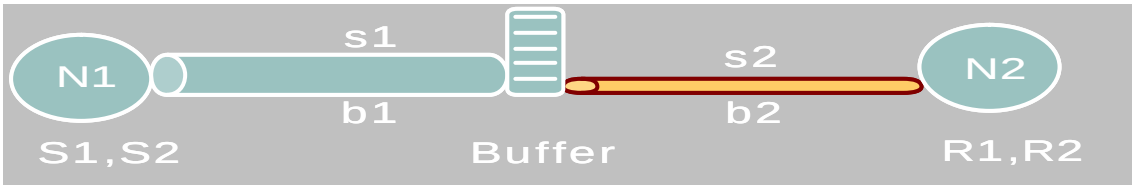
- Des paquets quittent les sources
- Des paquets entrent dans des buffer
- Des paquets quittent des buffer
- Des paquets entrent dans les récepteurs

Il permet également de renvoyer un certain nombre de variables internes du système propres à chacun des éléments modélisés. Une des variables à laquelle nous allons nous intéresser est par exemple la taille de la fenêtre de congestion. D'autres paramètres comme les valeur des RTT, des seuils et bien d'autres, sont également récupérables.

Pour réaliser les simulations, nous devons mettre en place un modèle simple qui permettra de réaliser les observations. Comme notre problème consiste en l'observation du comportement du débit d'une source suite à la fin de session d'une autre, nous utiliserons deux nœuds sur un réseau. L'un sera considéré comme le nœud auquel sont attachées les sources émettrices (N1) et l'autre possèdera les récepteurs (N2) (voir figure 3.2).

[Fig 3.2]

L'émetteur et le récepteur pourront entrer en contact au moyen d'un lien réseau. Ce réseau doit être modélisé de la façon la plus générale possible. Il présentera donc un goulot d'étranglement de manière à faire apparaître des congestions. Dès lors le réseau sera modélisé par deux segments de bande passante différente. L'interconnexion de ses deux segments présentera une mémoire tampon "buffer" (voir figure 3.3). Ce buffer pourra accepter un nombre limité de paquets et sera de taille prédéfinie.



[Fig 3.3]

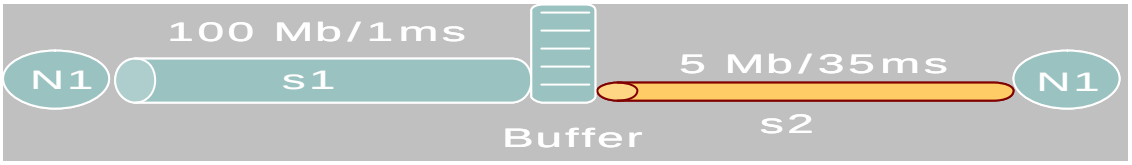
Sur la figure 3.3 on représente par

S1 et S2 : Les sources
R1 et R2 : Les récepteurs

N1 : Le nœud auquel sont associés les source émettrices S1 et S2.
N2 : Le nœud auquel sont associés les récepteurs R1 et R2 qui reçoivent respectivement les informations émises par les sources S1 et S2
s1 et s2 : Les segments du réseau de débit b1 et b2.
Buffer : Le buffer intermédiaire

Nous devons encore choisir les caractéristiques des bandes passantes des segments. Nous avons choisi des valeurs réalistes qui correspondent à des bandes passantes que nous pouvons rencontrer dans la plupart des réseaux. Le segment s1 a une bande passante de 100 Mb et le segment s2 de 5Mb. Cependant ces valeurs restent dans une grande mesure arbitraires (voir figure 3.4).

A chacun des segments correspond également un délai associé. Comme pour les bandes passantes, les valeurs de ceux-ci sont des valeurs plausibles mais restent arbitraires. Toutes les topologies étant particulières, il n'est pas possible de choisir une topologie meilleure qu'une autre.



[Fig 3.4]

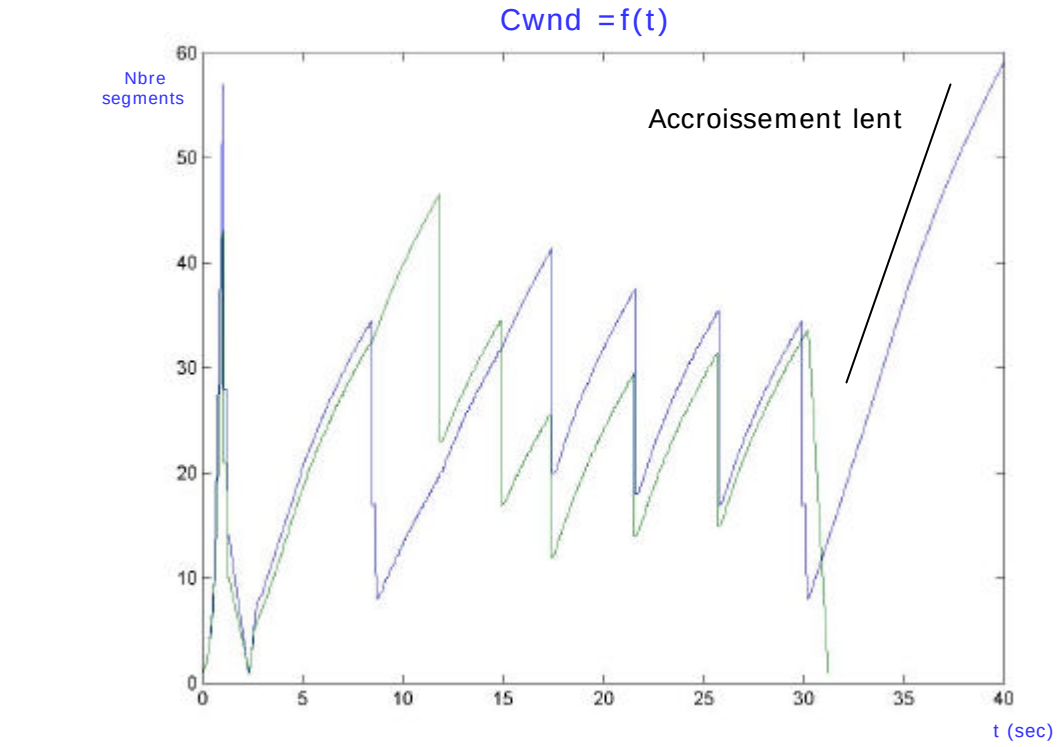
Suite aux caractéristiques intrinsèques des liens, nous pouvons définir une taille de buffer appropriée. Celle-ci peut être calculée sur base de la bande passante du lien le plus lent et sur base du délai total de transmission (voir B.L. Tierney [41] et Semke, J. Mahdavi, M. Mathis [42]).

Taille_Buffer = (Bande passante du lien le plus faible)*delai / 8 / octet par paquet
Délai = (Délai s1 + Délai s2) * 2 = 72 ms

Si nous définissons la taille des paquets à 1400 bytes nous obtenons une taille de buffer de 32 paquets pour la topologie construite.

3.4 Analyse du problème

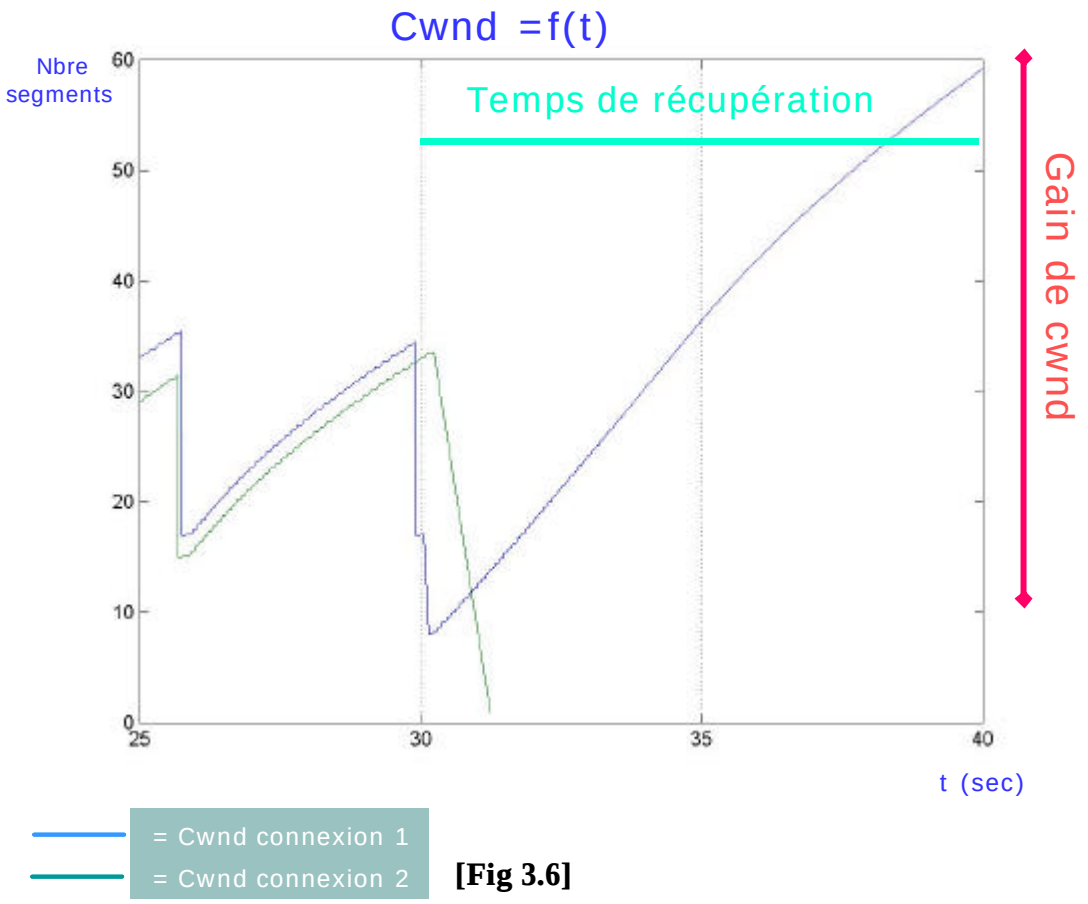
Après simulation avec des sources émettant suivant le protocole Reno, nous pouvons extraire des résultats générés, la variation de la taille de la fenêtre des congestion des sources en fonction du temps (voir figure 3.5).



Cwnd = Congestion Window
= Cwnd connexion 1
= Cwnd connexion 2

[Fig 3.5]

Dans cette simulation, nous arrêtons le transfert réalisé par la source 2 au temps t=30 sec. Alors que la source 1 pourrait se rendre compte de cette fin de session et augmenter de façon plus agressive la progression de sa fenêtre de congestion, elle reste dans la phase de congestion avoidance et n'atteint le seuil de congestion que très lentement.



[Fig 3.6]

Le résultat de la simulation donnée sur la figure [3.6] permet d'obtenir une idée du temps qu'il faudra mettre à la source S1 pour atteindre le seuil de congestion en phase de congestion avoidance.

Sur base de cette simple simulation il est possible de se poser quelques questions; à savoir:

1. Quel accroissement de taille de fenêtre de S1 est permis
2. Quand cet accroissement peut-il être effectué
3. De quelle manière réaliser l'accroissement
4. Quel sera l'effet d'un milieu de propagation à pertes élevées

3.5 Elaboration de la solution

3.5.1 Valeur de l'accroissement

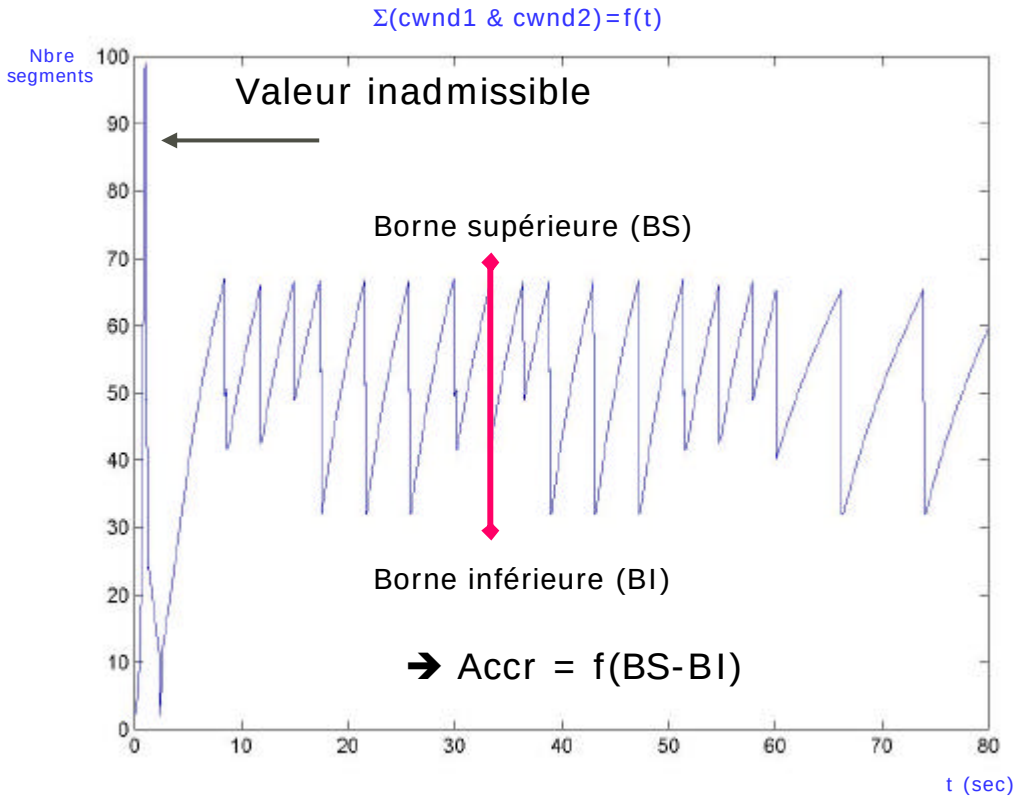
Sur la figure suivante [3.7] nous représentons l'évolution de la somme des fenêtres de congestion des deux sources en fonction du temps. Cette somme nous permet d'avoir une idée de la valeur totale du nombre de paquets que les sources peuvent émettre au cours du temps. Il faut cependant prendre garde à cette interprétation. Car si ces valeurs de taille de fenêtre cumulées nous renseignent sur les paquets émis, il ne s'agit pas d'office du nombre de paquets qui arrivent à destination. Certains d'entre eux peuvent se perdre en cours de route suite à une congestion.

Par observation, nous voyons que la valeur de cette somme de taille de fenêtre, oscille entre deux seuils en régime. L'origine, moment où les deux sources démarrent leur transfert, l'algorithme de slow start est exécuté, et donne lieu à des valeurs de fenêtre trop grande. Ceci provoque une entrée très rapide en congestion. Nous ne pouvons donc pas prendre en considération la somme des tailles de fenêtre obtenues à cet instant.

L'accroissement des fenêtres est cadencé par l'arrivée des Acks sous TCP Reno; Une augmentation de la valeur des fenêtres qui briserait cet équilibre n'est donc pas à envisager. Il faut donc choisir une valeur d'accroissement calculée sur base des tailles des fenêtres précédentes. L'accroissement à apporter à la source S1 sera donc une fonction de la valeur des deux seuils observés.

Dans la solution que nous désirons apporter, nous avons choisi d'augmenter la valeur de cwnd1 d'une quantité égale à la valeur de cwnd2 juste avant que la source S2 n'arrête son transfert. Ce choix est motivé de la façon suivante: l'accroissement de la fenêtre de S2 avant la terminaison de la session s'effectue par les algorithmes habituels de TCP. Cette fenêtre est elle donc aussi cadencée correctement par l'arrivée des accusés de réception. Si l'on ajoute à cwnd1 cette quantité cwnd2, on ne dérégulera pas le système.

Cette procédure est valable dans le cas où les sources S1 et S2 émettaient en direction d'un même nœud du réseau en empruntant le même chemin. Nous n'avons pas encore envisagé le cas des chemins multiples ni de sources émettant à partir d'un même nœud en direction de nœuds différents.



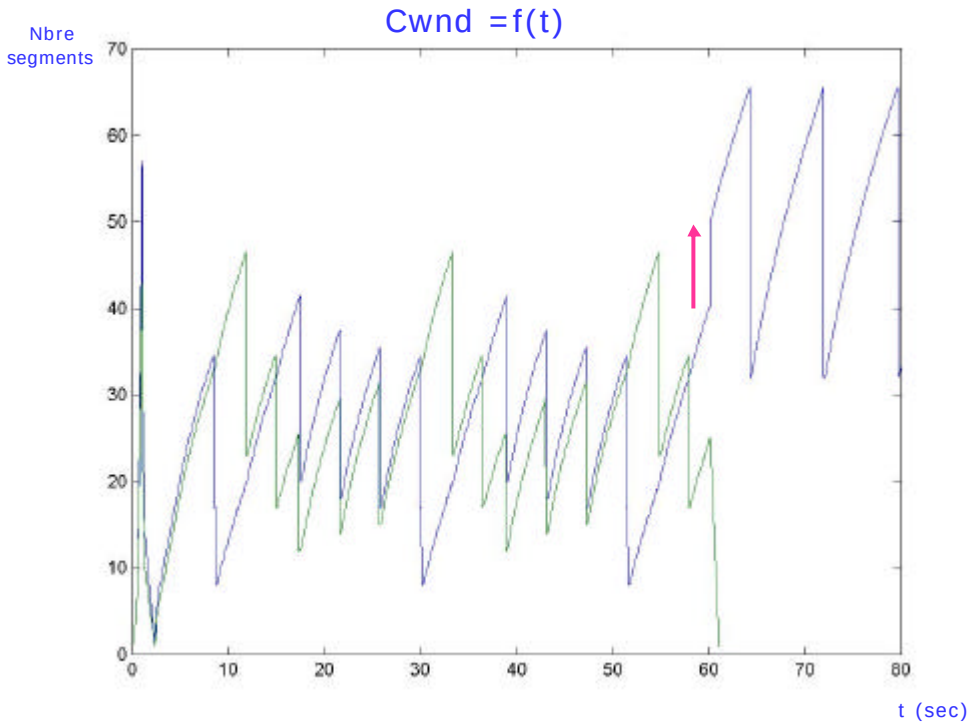
[Fig 3.7]

3.5.2 Réalisation de l'accroissement

Il nous est possible d'accroître la fenêtre de congestion de la source S1 de plusieurs manières différentes. Parmi celles-ci nous pouvons:

- Effectuer une modification du SStresh et accroître la fenêtre de congestion en mode de Slow Start jusqu'à ce qu'elle atteigne la valeur désirée
- Réaliser une augmentation brutale de la fenêtre de congestion en modifiant immédiatement sa valeur.

Nous avons choisi la deuxième méthode dans notre solution. La figure suivante [3.8] montre le résultat d'une simulation avec celle-ci. Nous pouvons y voir l'accroissement de la fenêtre de cwnd1 suite à la fin de session de S2. S1 poursuit, après cet accroissement, une phase de congestion avoidance tant qu'aucune perte de paquet ou DupAcks n'apparaissent. Il reste cependant à constater que, malgré l'accroissement apporté à la fenêtre de congestion, la source S1 continue d'avoir de grandes oscillations de taille de fenêtre. Plus la congestion apparaît avec des fenêtres de grandes valeurs, plus ces oscillations seront d'amplitude élevées.

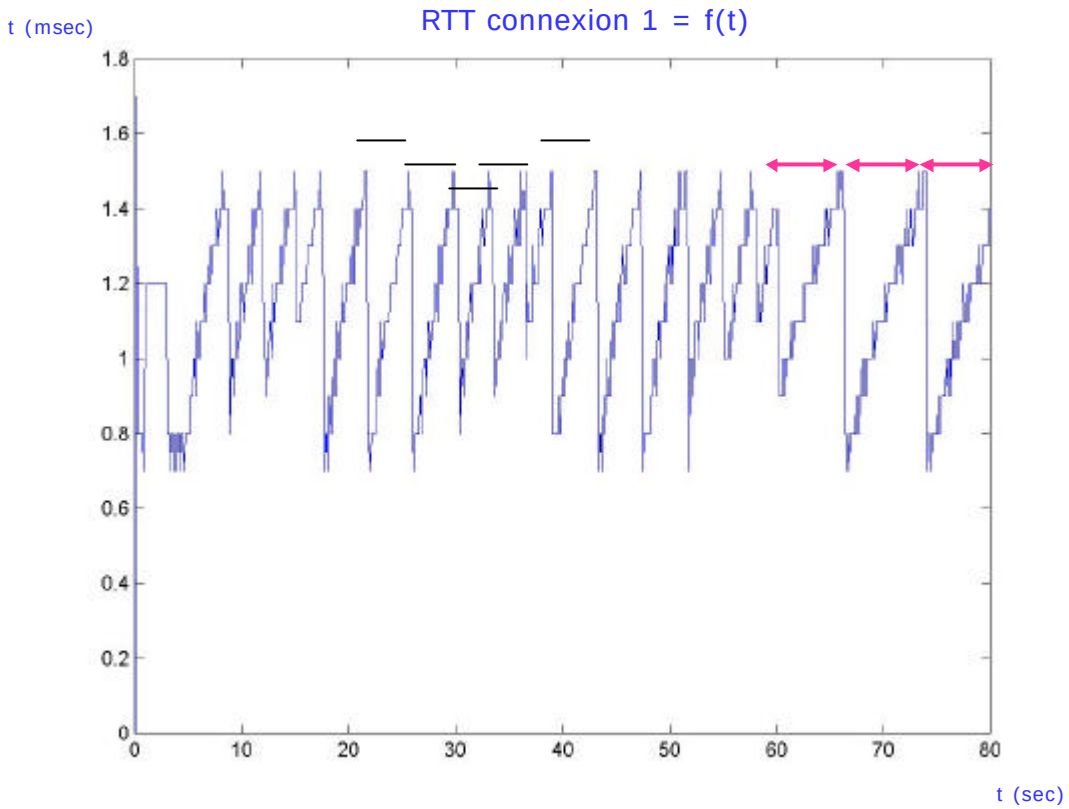


[Fig 3.8]

L'augmentation de la valeur de la fenêtre de congestion doit être déclenché après la fin d'une session. Il est facile au niveau de la couche applicative de savoir quand une session prend fin: soit l'utilisateur arrête le transfert, soit la totalité des données à transmettre ont été transférées. Il est envisageable de mettre en place un système par lequel cette information serait communiquée au niveau des couches inférieures et en particulier au niveau de la couche transport. Dans notre cas, nous voudrions, dans un premier temps, que ce soit la couche transport, qui de manière autonome, réalise qu'une fin de session s'est produite. Le cas d'un système de message transmis de la couche applicative vers la couche transport n'est pas envisagé dans ce travail.

Il faut donc trouver un critère de déclenchement de l'augmentation de la valeur de la fenêtre à partir des paramètres accessibles au niveau de la couche transport.

Parmi eux, le RTT peut nous renseigner sur de informations utiles. Le RTT dépend de la taille des buffers intermédiaires. Lorsqu'une source arrête d'émettre, elle n'envoie plus de paquets. Les buffers se remplissent donc par un nombre moins élevé de sources. Comme conséquence les paquets transmis par les autres sources encore actives accusent moins de délai. Le RTT des paquets est donc moins élevé. Pour nous en rendre compte nous pouvons tracer le graphique de l'évolution du RTT en fonction du temps.



[Fig 3.9]

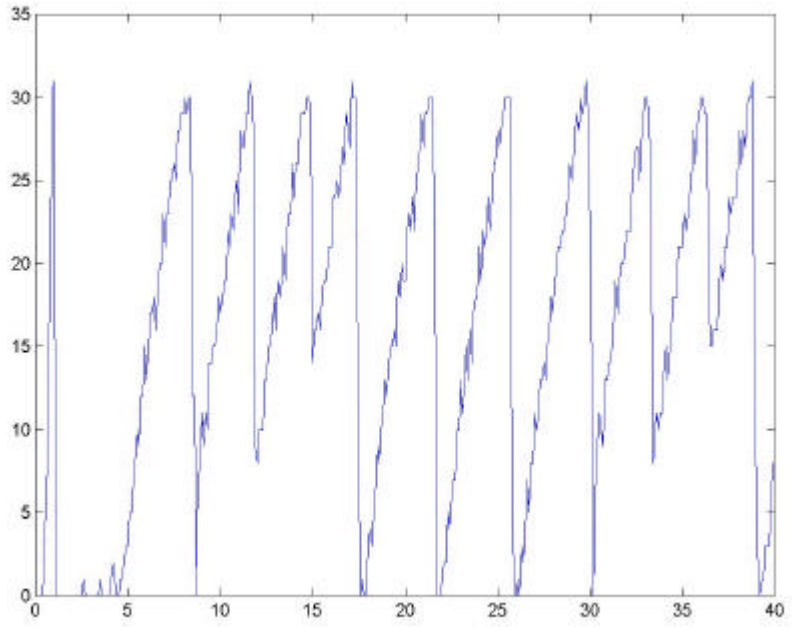
Sur la figure [3.9] nous représentons l'estimation du RTT par la source S1 dans le cas de base suivant: 2 sources émettent simultanément avec le protocole Reno avec le modèle défini précédemment. Au temps t=60 sec on arrête le transfert de la source S2 alors que la session de la source S1 se poursuit.

Nous observons que lorsque les deux sources émettent simultanément la fréquence d'oscillation du RTT est plus élevée que lorsqu'il n'y a plus qu'une seule source qui émet (t<60 sec & t>60 sec).

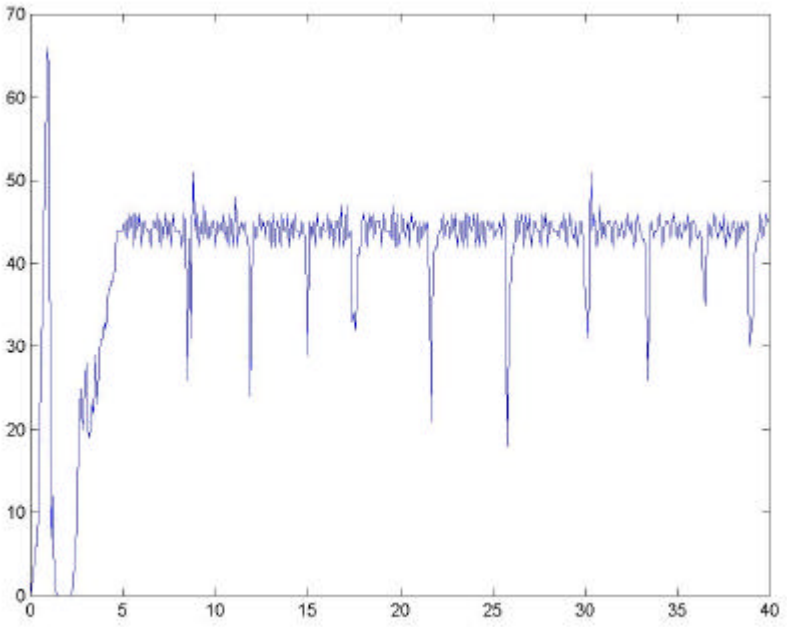
Si dans cette situation, le RTT oscille, c'est dû au comportement du buffer intermédiaire. Celui-ci se remplit jusqu'à atteindre sa limite. Il y a alors une perte de paquet. La source qui constate cette perte réduit sa fenêtre de congestion et le buffer a sur ce temps l'occasion de se vider. Le RTT des paquets transmis après cette perte sont ainsi plus faibles qu'à l'instant précédent la perte.

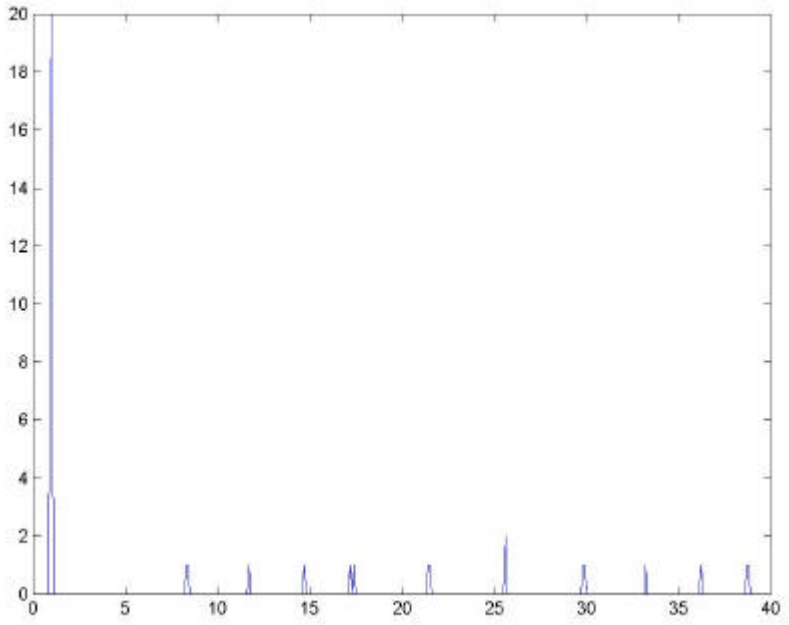
- La figure [3.10] montre le nombre instantané de paquets dans le buffer.
- La figure [3.11] montre le taux de paquets entrant dans le buffer en fonction du temps
- La figure [3.12] montre le nombre de paquets perdus en fonction du temps

- La figure [3.13] montre le taux de paquets sortant du buffer en fonction du temps

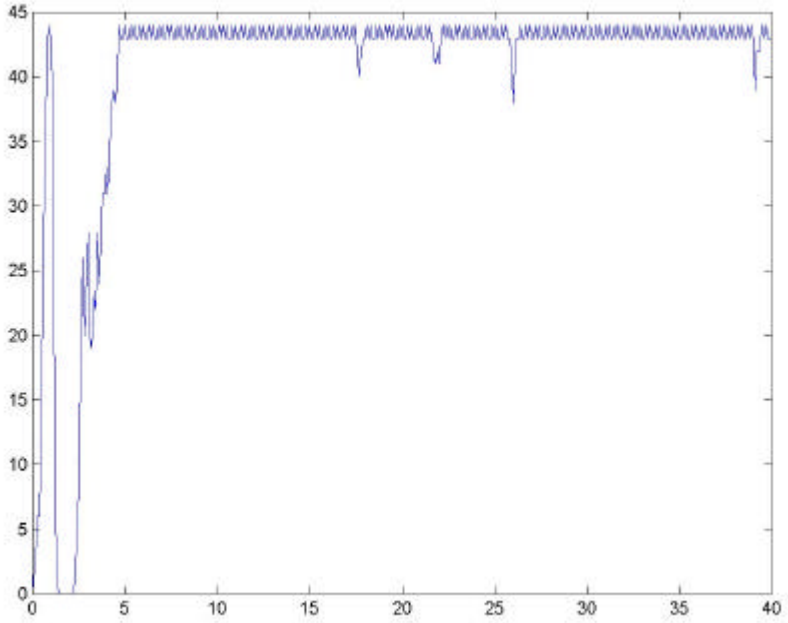


En haut **[Fig 3.10]** En bas **[Fig 3.11]**





[Fig 3.12]



[Fig 3.13]

Lorsque nous revenons à la figure 3.9, nous voyons qu'après la fin d'une session, la fréquence d'oscillation est plus faible. Un mécanisme que nous pouvons mettre en place est de retenir les temps séparant deux extremas consécutifs. Lorsqu'une variation apparaît entre ces extremas, il y a forcément une variation qui s'est produite au sein du système.

Nous pouvons donc nous baser sur la mesure de cette nouvelle variable interne pour déclencher notre mécanisme de modification de fenêtrage. Cette partie de travail est actuellement en cours d'élaboration. Dans le travail présenté dans ce rapport nous nous

contentons d'observations récoltées après un déclenchement manuel de l'accroissement de la fenêtre de congestion.

3.6 Simulations

En suivant le plan donné au point 3.2 nous avons réalisé les simulations en implantant dans le simulateur la solution développée. Nous envisageons deux cas généraux d'étude du problème:

- Modélisation avec un lien sans perte liée au médium de transmission
- Modélisation avec un lien générant des pertes de transmission

Toutes les simulations réalisées sont effectuées sur base du modèle présenté plus tôt. La durée d'une simulation est de 20 secondes. Au cours de cette simulation, nous initions les sessions des source S1 et S2 au temps t=0 sec. Au temps t=10 sec nous arrêtons la session de la source S2. La session de la source S1 se poursuit jusqu'au temps t=20 sec.

Chaque cas envisagé par simulation est simulé 20 fois de manière à ne pas établir de conclusions hâtives basées sur une expérience unique. Les conclusions présentées seront fournies sur base de l'entièreté de ces résultats.

3.6.1 Modélisation sans perte de donnée

Les seules pertes occasionnées dans la simulation sont celles causées par le remplissage des buffers. Cette modélisation permet de représenter la situation d'un lien filaire classique. Les simulations sont réalisées d'une part avec les sources émettant sous le protocole Reno et d'autre part avec le protocole Westwood.

Il n'y a pour cette famille de modélisation aucune modification à apporter au modèle déjà établi.

3.6.2 Modélisation avec pertes de données

Quelques modifications sont à apporter au modèle pour représenter le milieu de transmission avec perte.

Au segment S2 du modèle nous allons associer une variable aléatoire de distribution uniforme. Celle-ci nous permettra d'ajouter un paramètre de modélisation de perte au segment s2. Ce module d'erreur est ajouté en aval du buffer et réside sur le lien s2. Nous pouvons spécifier si ce module d'erreur génèrera des erreurs en fonction du nombre de paquets envoyés, ou du nombre de bits transmis ou encore s'il génèrera des erreurs aléatoire en fonction du temps. Dans le cas de nos simulations nous utiliserons un module d'erreur qui portera sur le nombre de paquets transmis. Il s'agit du modèle d'erreur par défaut proposé par le simulateur.

Nous envisagerons les cas du lien s2 présentant 2,5,10,15 et 20% d'erreurs durant toute la durée de la simulation. Nous nenvisagerons pas le cas où le taux d'erreur varie en fonction du temps.

3.7 Outils d'analyse des résultats

Les résultats que nous voulons connaître portent sur les comparaisons des performances des protocoles avec et sans apport de notre solution. Nous devons dès lors définir des outils qui nous permettront d'établir ces comparaisons.

Nous allons nous intéresser au rendement de transmission de l'information de la source S1 puisque c'est cette source qui subit les modifications de protocole.

A cet effet nous allons mesurer la bande passante utilisée par le flux de source. Plusieurs choix s'offrent à nous pour ce calcul. Un descriptif plus complet peut est proposé par N. Malouch et al (voir [45]).

3.7.1 Troughput en fonction du RTT

M. Mathis et al (voir [44]) présentent un moyen de calculer le troughput d'une source TCP en fonction du RTT et la probabilité de perte *p* des paquets que subit cette source.

Thrpt= (1/RTT)* sqrt(3/2p)

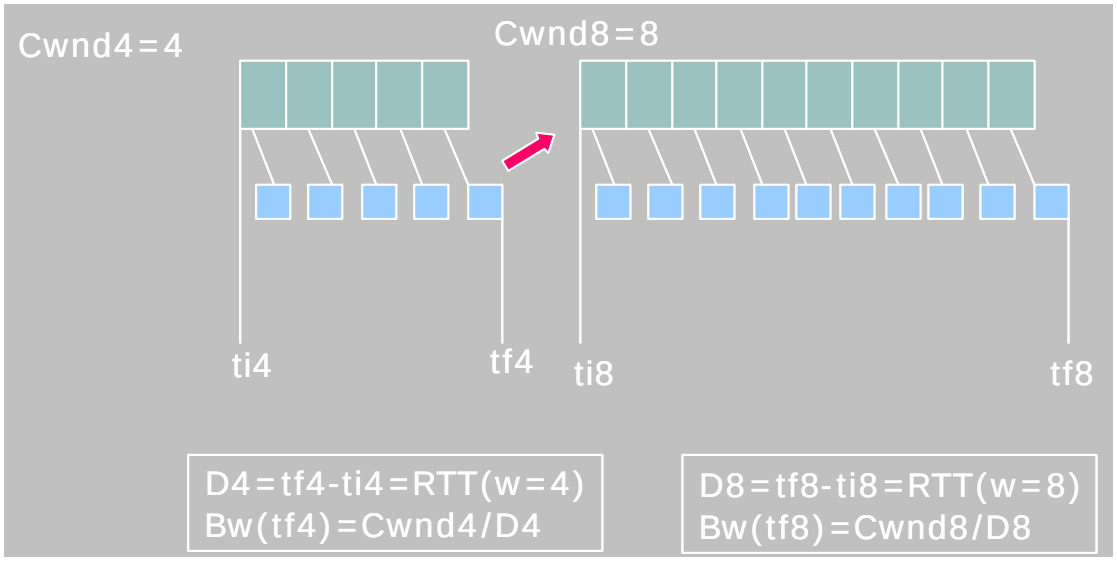
Un autre développement proposé dans [44] donne sur base d'observation du graphique de l'évolution de la taille de la fenêtre en fonction du temps:

B= 3W/4RTT dans le cas où il y a peu de Time Out.

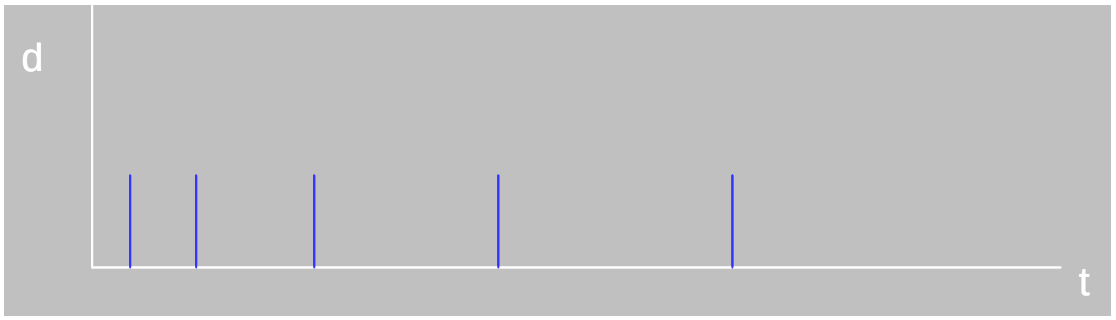
Le débit d'une source TCP est complètement déterminé par les caractéristiques de la fenêtre en état stationnaire.

Par [43] nous pouvons également l'estimer par W/RTT [voir figure 3.14].

Nous n'utiliserons pas cette méthode actuellement car elle laisserait apparaître des calculs de débit de manière irrégulière. Le calcul serait exécuté tout les RTT ce qui donne lieu à des calculs apériodiques [figure 3.15].



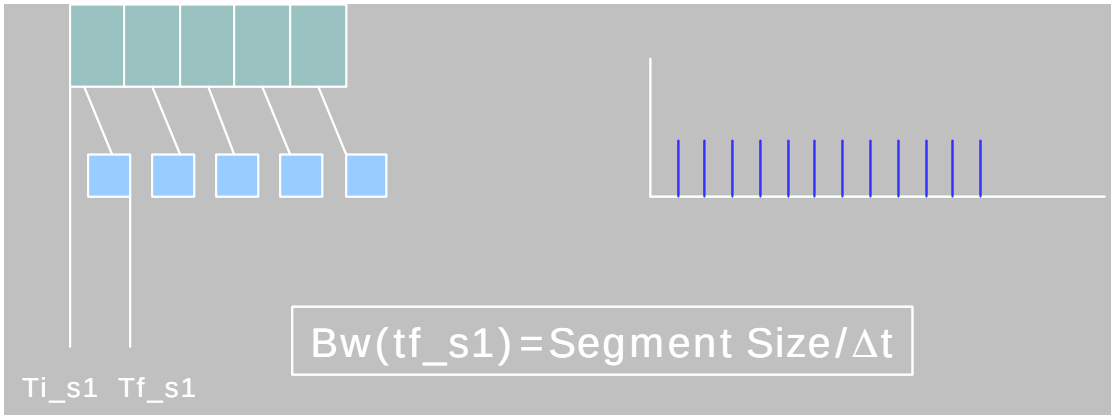
[Fig 3.14]



[Fig 3.15]

3.7.2 Throughput par segment

Le débit pourrait être calculé de manière similaire au point 4.7.1 si ce n'est que le calcul serait effectué après toute réception d'un accusé de réception (voir figure 3.16). Ceci ne constitue pas pour nous une idée raisonnable car nécessiterait beaucoup de temps de calcul par simulation.



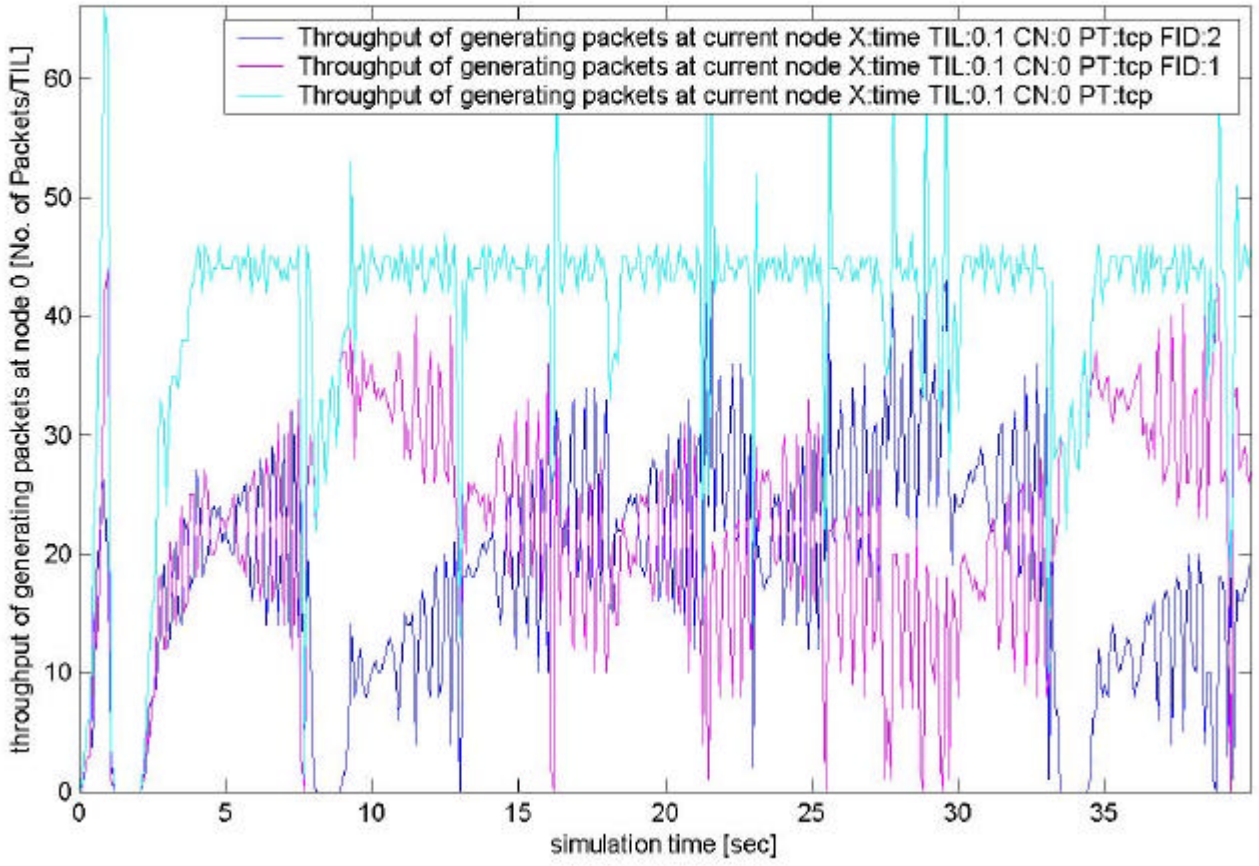
[Fig 3.16]

3.7.3 Throughput / Goodput par ns

La méthode que nous avons choisie pour analyser les résultats consiste à relever périodiquement le nombre de paquets émis par l'émetteur ainsi que le nombre de paquets reçus au niveau du récepteur. Cette méthode reflètera exactement le comportement des sources puisqu'il s'agit de mesures exactes.

Nous effectuons ces prélèvements tous les dixième de seconde durant toute la durée de la simulation. Ces prélèvements sont effectués séparément pour les flux issus de la source S1 et de la source S2. Nous effectuons également le prélèvement des mesures globales au niveau des nœuds.

Pour une simulation nous obtenons ainsi un graphe similaire à la figure [3.17]. Dans cette simulation, le prélèvement a été effectué au niveau du nœud '0' pour la source S1, S2 et le nœud auquel sont attachées ces mêmes sources. Nous mesurons le throughput des paquets générés.



[Fig 3.17]

A chaque simulation, nous effectuons ce prélèvement périodique de throughput. Des calculs statistiques sur les valeurs obtenues sont ensuite opérés pour récolter entre autre leurs moyennes et leurs variances.

Les calculs des rendements sont obtenus en effectuant les rapports entre le nombre de paquets transmis par le nombre de paquets reçus (goodput).

3.8 Présentation des résultats

Dans les tableaux ci-dessous (T1, T2, T3, T4) sont rapportés les principaux résultats des simulations. F1 et F2 font référence au flux des sources S1 et S2. Le rendement total donne le rendement associé au cumul des trafics issus des sources. On présente les valeurs pour le protocole Westwood et Reno avec et sans apport de modification. L'apport de la modification est repéré par le signe '+'.

Reno

Pertes (%)	Rendement Tot	Rendement F1	Rendement F2
0	0,988521837	0,987555721	0,981756907
1	0,982448434	0,983012924	1,007830777
5	0,948488984	0,946869279	0,971401495
10	0,902343057	0,949840027	0,93039633
15	0,857169451	0,897352922	0,97589913
20	0,839886893	0,925162204	1,026693869

[Tableau T1]

Reno +

Pertes	Rendement Tot	Rendement F1	Rendement F2
0	0,984225694	0,981584937	0,981756907
1	0,985884004	0,999921085	0,982743506
5	0,949222257	0,94998031	0,978671763
10	0,926508214	0,980732725	0,933721736
15	0,89134446	0,951716602	1,04513902
20	0,824165921	0,900394341	0,898799723

[Tableau T2]

Westwood

Pertes	Rendement Tot	Rendement F1	Rendement F2
0	0,990492539	0,992334717	0,981693364
1	0,984774838	0,984655984	1,030120599
5	0,948663296	0,949676443	0,948433857
10	0,910572653	0,944976148	0,986518253
15	0,890437488	1,207676569	0,935924483
20	0,822146908	0,849164882	1,070006712

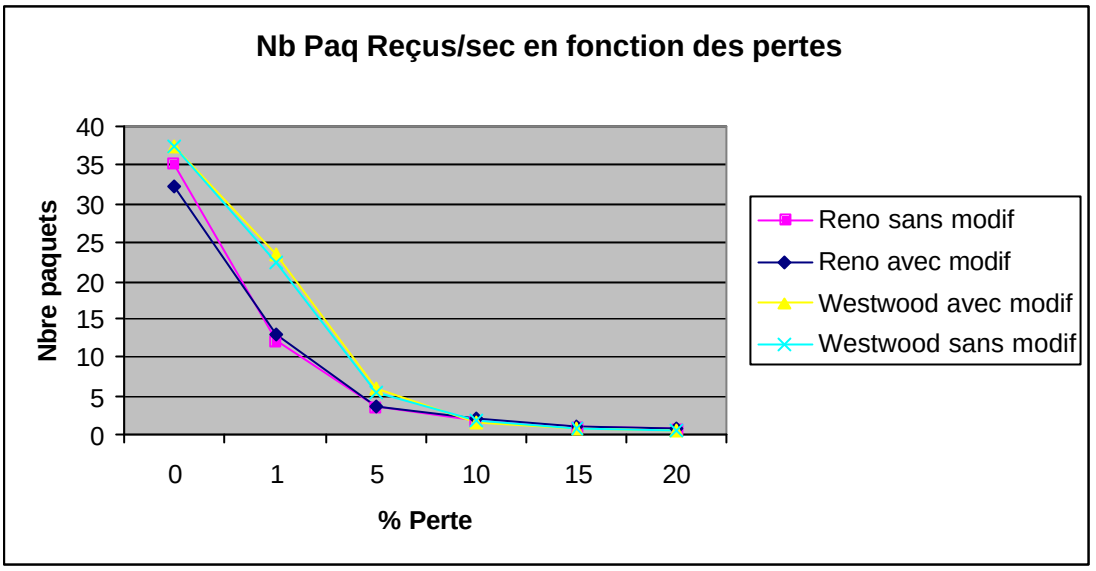
[Tableau T3]

Westwood +

Pertes	Rendement Tot	Rendement F1	Rendement F2
0	0,989447302	0,991068581	0,981693364
1	0,984402754	0,983811913	0,992224051
5	0,948181452	0,953608308	1,000162005
10	0,903718424	0,932981399	0,913762838
15	0,879349868	0,946569072	0,946488502
20	0,8437837	0,86780182	1,155128925

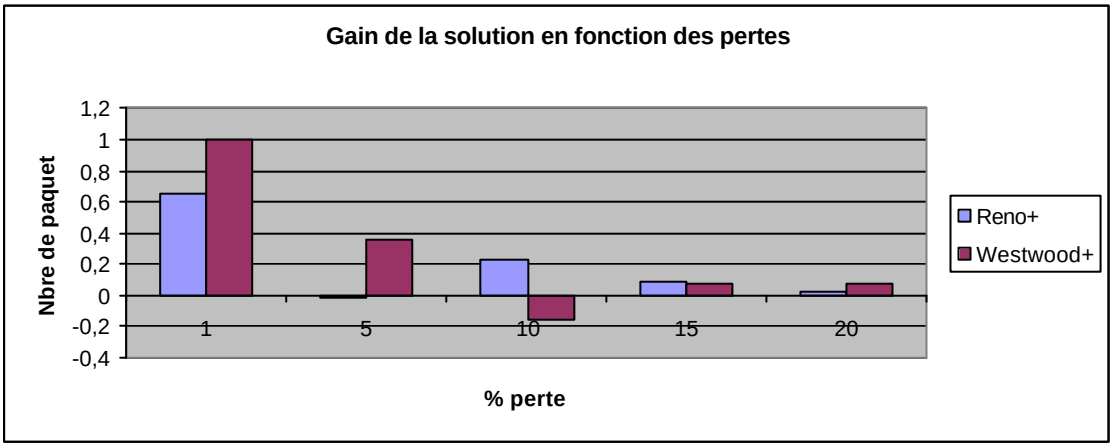
[Tableau T4]

Le graphe de la figure suivante [voir figure 3.18] renseigne dans l'ensemble des cas simulés le nombre moyen de paquets reçus par seconde par le récepteur en fonction du pourcentage de perte.



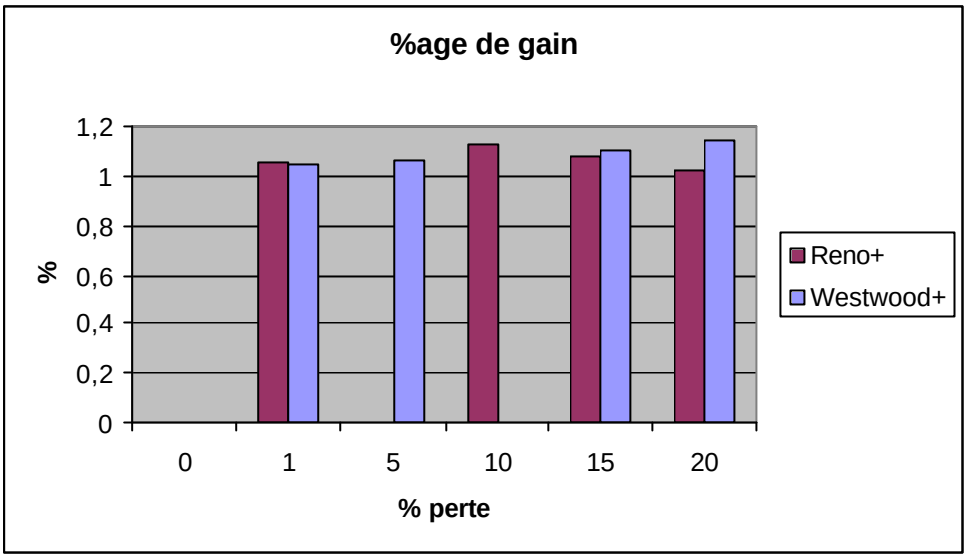
[Figure 3.18]

Le tableau suivant (voir tableau T5) donne le nombre de paquets par seconde gagnés en fonction du pourcentage de perte par rapport au protocole sans implémentation de la solution développée.



[Tableau T5]

Le tableau suivant (voir tableau T6) présente le pourcentage de paquets supplémentaires que l'on a pu transmettre par seconde à l'aide de la nouvelle solution implémentée. Nous ne représentons pas les résultats pour lesquelles la simulation a donné des résultats moins performants qu'avec les protocoles sans implémentations de la nouvelle solution.



[Tableau T6]

3.9 Remarques et interprétation des résultats

Avant d'interpréter les résultats, nous allons formuler quelques remarques. Ces remarques portent sur les divers cas de figure qui se sont présentés dans les simulations ainsi que sur les calculs qui nous permettent d'obtenir des résultats.

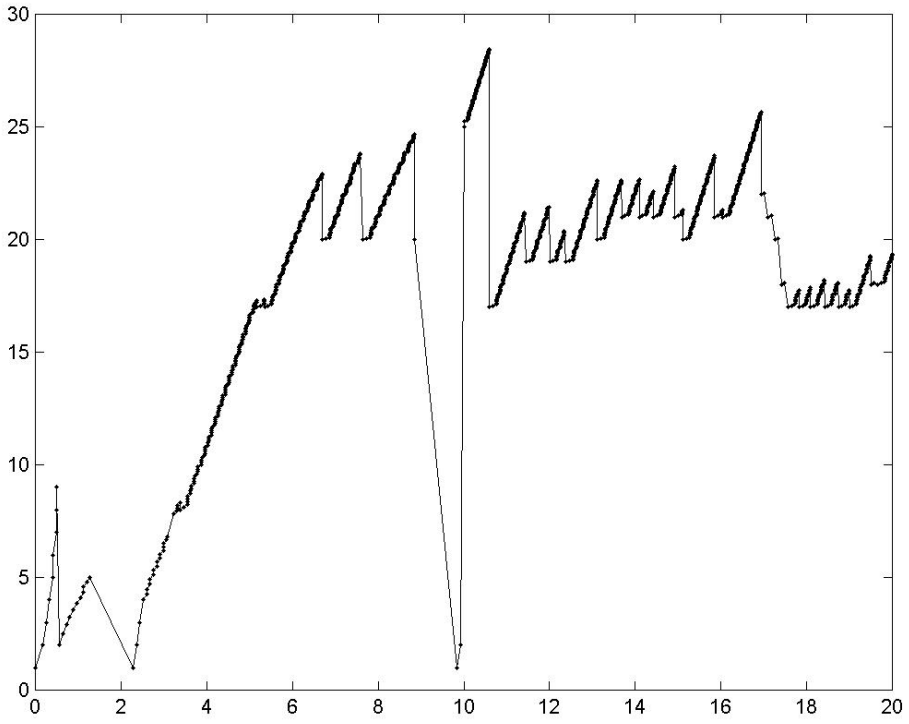
Nous analyserons ensuite les résultats obtenus.

3.9.1 Remarque 1

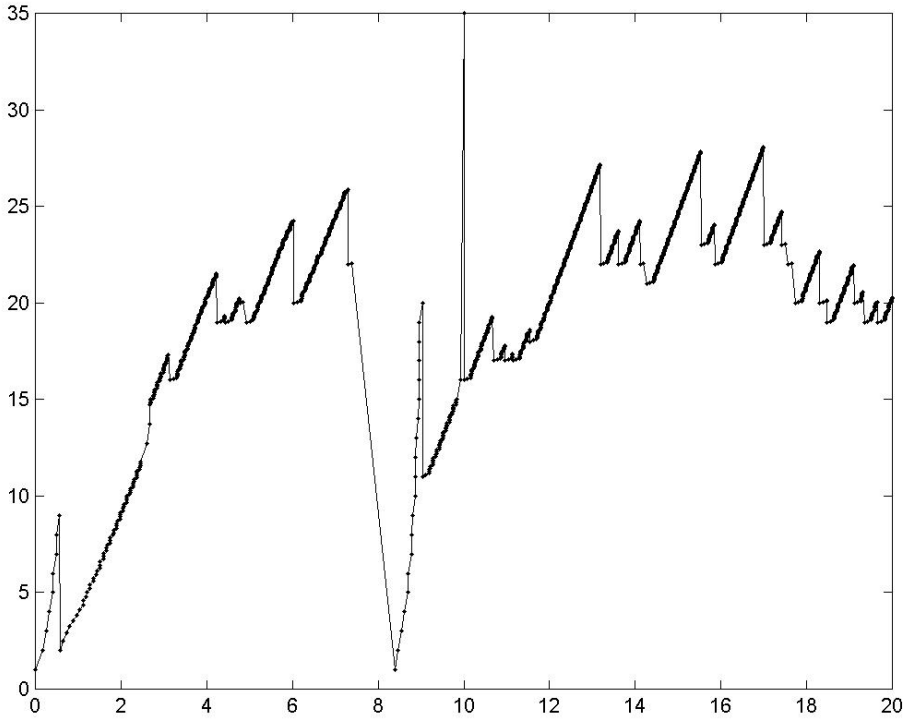
L'application du nouvel algorithme au protocole peut apporter un bénéfice à la connexion en cours, s'il n'y a pas de perte qui survient immédiatement après le redressement de la fenêtre. Ces deux situations sont présentées sur les deux figures suivantes [3.19] et [3.20].
Sur la première figure, on retrouve le cas favorable ou l'apport de l'amélioration est bénéfique à la source. L'accroissement de la fenêtre de congestion augmente le seuil à partir duquel on continue la transmission en mode Congestion Avoidance.

D'autre part, sur la seconde figure, on retrouve le cas ou une perte de paquet survient immédiatement après le redressement. Ce cas n'apporte aucune amélioration au protocole et au contraire pourrait même entrer en sa défaveur. En effet, lorsque l'augmentation de fenêtre survient, l'émetteur transmet un nombre important de paquets et s'en suit un nombre important de retransmissions.

Un travail à réaliser serait de comparer le taux de fin de sessions qui se produisent par rapport au taux de perte de paquets. Nous pourrions ainsi nous rendre compte lequel de ces cas est le plus courant.



[Fig 3.19]



[Fig 3.20]

3.9.2 Remarque 2

Dans les simulations que nous avons réalisées, nous n'avons travaillé qu'avec deux sources émettant vers une même destination. Cette destination est atteinte par les deux sources en empruntant un même chemin. Cette situation, ne représente en rien une généralité par rapport aux sessions ftp initiées par une source. La solution apportée est applicable et favorable dans ce cas. Une étude doit être menée dans le cas ou une source émet deux sessions ftp vers des destinations différentes.

Il faut également prendre en considération le fait qu'une source peut générer un nombre n de sessions ftp. Dans notre cas, nous ne considérons que deux sources. L'étude générale pour un nombre n de sources en direction de n récepteurs est un cas complexe.

3.9.3 Remarque 3

Dans les calculs de throughput des flux issus des sources, nous avons rencontré le cas où le taux moyen de paquets reçus par un récepteur était plus élevé que le taux moyen de paquets émis par la source. Cette constatation provient de la précision avec laquelle nous échantillonnons nos données. Un compromis entre sur la fréquence d'échantillonnage doit être réalisé pour obtenir des mesures cohérentes.

3.9.4 Remarque 4

Nos simulations sont réalisées sur une période de 20 secondes. Les valeurs des rendements et des performances obtenus sont estimées dans le cas ou une session ftp se termine au cours de ces 20 secondes. Si des simulations ou des études doivent être réalisées à partir de ces résultats, il faut prendre garde à les reporter aux mêmes unités.

3.10 Conclusion

Nous voyons que l'apport de la nouvelle méthode accroît très faiblement le rendement des protocoles de base. Cette faiblesse est une conséquence de la remarque du point 4.9.4. Il faut préciser que ces performances sont à rapporter pour une fin de session ftp sur une durée de 20 secondes. Il est évident que plus la simulation sera de grande durée, en considérant une fin de session, plus le gain paraîtra insatisfaisant. Dès lors il faut pour mieux se rendre compte du gain réel multiplier le gain obtenu dans les simulations par la durée de la simulation.

Sur base de ces considérations nous constatons que l'apport de la solution apporte environ 1% de performance supplémentaire par temps de simulation, dans tous les cas simulés.

Cette augmentation de rendement est malgré tout assez faible. De meilleurs résultats pourraient être présentés avec une topologie ou les temps de propagations seraient plus élevés, comme par exemple dans les liens satellites. Dans ce cas, la cadence de réception des ACKs est plus lente suite aux délais. Le mécanisme de Congestion Avoidance prend dès lors beaucoup plus de temps pour permettre à la fenêtre d'arriver à des valeurs proches du seuil de congestion. La simulation qui présente ces résultas n'est pas donnée dans le cadre de ce travail.

Après avoir analysé le comportement des fenêtres de congestion dans le cas de simulation de lien avec perte, nous pouvons constater:

- Il est difficile de trouver un facteur permettant de détecter la fin d'une session suite au caractère aléatoire des fenêtres provoqué par les pertes
- L'apport du gain suite à notre algorithme se révèle peu utile s'il est suivit d'une perte.

Dès lors, il serait préférable d'utiliser une autre méthode pour annoncer à la source encore active, une fin de session. Il faudrait un système par lequel la couche transport puisse recevoir de la couche applicative des messages. Ces messages consisteraient à annoncer les fins de session. Le protocole TCP se baserait sur ces messages pour mettre en place de nouveaux mécanismes qui lui assureraient de meilleures performances.

Il est à constater que dans tous les cas simulés, le protocole TCP Westwood présente de meilleures performances que le protocole TCP Reno. Ces simulations corroborent les résultats présentés dans la littérature.

Notons l'importance du choix du temps auquel il faut démarrer l'accroissement de la fenêtre de congestion. Si celui-ci se fait de manière immédiate à la fin de la session ftp, l'émetteur envoit soudainement trop de paquets que le buffer ne peut réceptionner. Il y a dès lors des retransmissions qui sont à prévoir et donc une baisse de rendement. Un court instant entre la fin de la session et l'accroissement de la fenêtre est à insérer.

Chapitre 4

Conclusions

Dans le travail que nous venons de présenter, nous avons proposé un petit éventail des diverses versions des protocoles de transport de type TCP. Nous n'avons pas relevé la globalité des solutions proposées dans la littérature car ceci consisterait principalement en une tâche bibliographique. Cette étape fût importante et nécessaire pour comprendre les enjeux liés aux protocoles de transport et d'en connaître les mécanismes. Aussi, c'est grâce à elle qu'il fût possible de cibler un problème potentiel.

Actuellement des travaux sont toujours en cours pour l'amélioration de ces protocoles. Ce sont les problèmes relatifs à l'interconnexion des réseaux sans fils et des réseaux avec fils qui sont en cours d'étude et qui offrent le plus de perspective de recherche.

Dans ce travail de DEA, nous avons tenté de présenter une amélioration du protocole TCP dans le but d'accroître ses performances suite à la détection de fin de sessions. Ce problème nous a permis de nous rendre compte qu'il existait encore quelques améliorations à apporter au protocole. Notre solution s'avère plus adaptée à implémenter dans le cas de réseaux sans pertes de données.

Dans la perspective de la thèse nous avons compris quels étaient les moyens pratiques de modélisation de protocoles et mis en place un ensemble d'outils permettant d'en mesurer les performances. Le DEA a permis d'ouvrir des pistes concernant la manière dont nous allons aborder les problèmes liés à la transmission des données en milieu hétérogène.

Notre thèse se poursuivra dans l'idée de l'étude de performance des boosters de protocoles. Les boosters de protocoles permettent de tenir compte des milieux de propagation afin d'adapter de manière efficace les algorithmes contenus dans les protocoles de contrôle de congestion. Actuellement nous travaillons sur une version d'un protocole Booster développé par l'Université de Gand. Nous étudions l'influence du protocole TCP Westwood lorsque ce Booster est introduit dans le réseau.

L'étude des problèmes liés aux pertes de paquets lors des Hand Off est également poursuivie. Nous désirons étudier les comportements des algorithmes TCP utilisés dans les sources mobiles: quels sont les comportements liés au passage d'une antenne à une autre.

Références

[1] J. Postel, "Transmission Control Protocol – DARPA Internet Program Protocol Specification", RFC 793, DARPA, Septembre 1981.

[2] Clark, D. David, "Window and Acknowledgement Strategy in TCP", RFC 813, Juillet 1982.

[3] H. Molly Shor, K. Li, J. Walpole, D. Steere, C. Pu, "Application of control theory to modeling and analysis of computer Systems", DARPA.

[4] V. Jacobson, "Congestion avoidance and Control", *SIGCOMM symposium on Communications Architectures and Protocols*, pages 314-329, 1988.

[5] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control", RFC 2581, avril 1999.

[6] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP," in *Computer Communications Review*, 26(3), Juillet 1996.

[7] P. Karn, C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," in *ACM Trans. On Computer Systems*, vol. 9, no 4, pp. 364-373, Novembre 1991.

[8] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, Mai 1992.

[9] J. Nagle, "Congestion Control in IP/TCP Internetworks", RFC 896, FACC, Janvier 1984.

[10] R. Jain, "Divergence of Timeout Algorithms for Packet Retransmission", in *Proc. Fifth Phoenix conference on Computer and Communications*, Scottsdale, Arizona, Mai 1986.

[11] P. Karn, C. Partridge, "Estimating Round-Trip Times in Reliable Transport Protocols," in *Proceedings SIGCOMM '87*, Stowe, VT, Aout 1987.

[12] L. Zhang, "Why TCP Timers don't work well," in *Proc. SIGCOMM '86*, Stowe, Vt., Aout 1986.

[13] R. Braden, "Requirements for Internet hosts – Communication Layers", RFC 1122, Octobre 1989.

[14] V. Jacobson, "Berkeley TCP Evolution from 4.3 Tahoe to 4.3 Reno," *Proceedings of the 18th Internet Engineering Task Force, University of British Colombia, Vancouver, BC*, Septembre 1990.

[15] S. Mascolo, C. Casetti, M.Gerla, S.S. Lee, M. Sanadidi, "TCP Westwood: congestion control with faster recovery," in *MILCOM 2000,* Los Angeles, CA, Octobre 2000.

[16] L.S. Brakmo, S. W.O'Malley, L.L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of ACM SIGCOMM'94,* pp. 24-35, Octobre 1994.

[17] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," in *Computer Networks and ISDN Systems*, 17:1-14, 1989.

[18] K.K. Ramakrishnan, D.M. Chiu, R. Jain, "Congestion Avoidance in Computer Networks with a connectionless Network Layer", Technical Report DEC-TR-509, Digital Equipment Corporation, 1987.

[19] K.K. Ramakrishnan, E. Jain, "A binary feedback Scheme for congestion avoidance in computer networks with a connectionless Network Layer," in *Proceedings of ACM SIGCOMM'88,* 1988.

[20] Z. Wang, J. Crowcroft, "Eliminating Periodic Packet Losses in 4.3 Tahoe BSD TCP Congestion Control Algorithm," in *proceedings of ACM Computer Communication Review*, 22(2):9-16, avril 1992.

[21] R. Jain, "A Delay-Based Approach for Congestion Avoidance in Inteconnected Heterogeneous Computer Networks," in *proceedings of ACM Computer Communication Review*, 19(5):56-71, octobre 1989.

[22] J. Mo, R. La, V. Anantharam, J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *proceedings of IEEE Infocom*, Mars 1999.

[23] S. Low, L. Peterson, L. Wang, "Understanding TCP Vegas: A Duality Model," in *SIGMETRICS*, pages 226-235, 2001.

[24] G. Hasegawa, K. Kuruta, M. Murata, "Fairness and Stability of Congestion Control Mechanisms of TCP," in *Proceedings of INFOCOM'99*, pages 1329-1336, Mars 1999.

[25] J.C. Hoe, "Improving the start-ip Behavior of a Congestion Control Scheme for TCP," in *ACM SIGCOMM'96*, Stanford, CA, USA, 1996.

[26] C. Casetti, M. Gerla, S.S. Lee, S. Mascolo, M. Sanadidi, "TCP With Faster Recovery," in *IEEE INFOCOM'2000,* Tel Aviv, Israel, Mars 2000.

[27] D. Lin, H.T. Kung, "TCP fast recovery strategies: Analysis and improvements," in *INFOCOM'98,* 1998.

[28] R. Wang, M. Valla, M.Y. Sanadidi, M. Gerla, "Adaptive Bandwith Share Estimation in TCP Westwood," Technical Report, also submitted to Globecom 2002.

[29] C. Dovrolis, P. Ramanathan, D. Moore, "What do packet Dispersion techniques measure," in *INFOCOM*'2001, pages 905-914, 2001.

[30] R. Wang, M. Valla, M.Y. Sanadidi, B. Kwok Fai Ng, M. Gerla, "Efficiency/Friendliness Tradeoffs in TCP Westwood," in *IEEE Symposium on Computers and Communications*, Taormina, Italy, Juillet 2002.

[31] M. Gerla, S.S. Lee, G. Pau, "TCP Westwood Simulation Studies in Multiple-Path Cases," Technical Report, 2002.

[32] A. Zanella, G. Procissi, M. Gerla, M.Y. Sanadidi, "TCP Westwood: Analytic Model and Performance Evaluation", In *Proceedings of IEEE Globecom*, pages 1703--1707, 2001.

[33] M. Allwam, S. floyd, C. Partridge, "Increasing TCP's initial Window," Internet Draft, Avril 1998.

[34] H. Wand, H. Xin, D.S. Reeves, K.G. Shin, "A Simple Refinement of Slow Start of TCP Congestion Control," in *IEEE Symposiums on Computers and Communications*.

[35] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement option," RFC 2018 IETF, Octobre 1996.

[36] S. Floyd, "Issues of TCP With Sack", Technical report, Mar. 1996. URL ftp://ftp.ee.lbl.gov/papers/issues sa.ps.Z.

[37] M. Allman, "TCP Byte Counting Refinements," in *ACM computer communication Review*, Juillet 1999.

[38] C. Parsa, J.J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media", in *International Conference on Network Protocols*, p 213-21. IEEE, Octobre 1999.

[39] C. Parsa, J.J. Garcia-Luna-Aceves, "Differentiating Congestion vs Random Loss: A Method for Improving TCP Performance over Wireless Links," in *Proc IEEE WCNC* 2000.

[40] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and Sack TCP", in *Computer Communication Review*, vol. 26, pp. 5--21, July 1996.

[41] B. L. Tierney, "TCP Tuning guide for distributed application on wide area networks" .

[42] Semke, J. Mahdavi, M. Mathis, "Automatic TCP Buffer Tuning", in *Computer Communication Review*, ACM SIGCOMM, volume 28, number 4, Oct 1998.

[43] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP Throughput: a simple model and its empirical validation", in *ACM SIGCOMM'98*, 1998.

[44] M. Mathis, J. Semake, J. Mahdavi, T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm", in *Computer Communication Review*, juillet 1997.

[45] N. Malouch, Z. Liu ,"On steady state analysis of TCP in networks with differentiated services," in *Proceedings of Seventeenth International Teletraffic Congress, ITC'17*, Décembre 2001.

[46] K. Fall and K. Varadhan, "ns Notes and documentation", LBNL, Août 1998, http://www-math.cs.berkeley.edu/ns.