Université de Franche-Comté **UFR Sciences et Techniques**

LIFC **École doctorale SPIM**

Numéro d'ordre : 111

Optimisation et adaptation des communications dans un réseau hétérogène

THÈSE

présentée et soutenue publiquement le 02 12 2008

pour l'obtention du

Grade de Docteur de l'Université de Franche-Comté

École Doctorale SPIM - Spécialité Informatique

par

Sébastien LINCK

Composition du jury

Directeur de thèse :	François Spies	Professeur à l'Université de Franche-Comté
Président :	Jean-Christophe Lapayre	Professeur à l'Université de Franche-Comté
Rapporteurs :	Pascale Minet	Chargée de recherche Habilité à Diriger des Recherches, INRIA Rocquencourt
	Jean-Jacques Pansiot	Professeur à l'Université Louis Pasteur de Strasbourg
Examinateur :	Benoît Hilt	Maître de Conférences à l'Université de Haute-Alsace
Co-encadrant :	Eugen Dedu	Maître de Conférences à l'Université de Franche-Comté

Mis en page avec la classe thloria.

A mon père ...

Remerciements

Le bon déroulement de ma thèse et la rédaction de ce mémoire n'auraient pu avoir lieu sans le concours de nombreuses personnes. Je tiens à les en remercier, et tout particulièrement :

François Spies, Professeur au LIFC, qui m'a accueilli au sein de l'équipe *OMNI* Optimisation, Mobility, Networking de Montbéliard et suivi tout au long de cette thèse.

Eugen Dedu, Maître de conférences au LIFC qui a supervisé le bon déroulement du projet et qui m'a prodigué des conseils avisés durant celui-ci, forts utiles au demeurant.

Pascal Chatonnay, Philippe Canalda, Gérard Cécé, Dominique Dhoutaut, Damien Charlet maîtres de conférences, Julien Bourgeois, Professeur au LIFC pour leur coopération lors de nos différentes réunions de recherche.

L'ensemble des doctorants du LIFC en particulier Frédéric Lassabe, Jean-Baptiste Ernst-Desmulier pour leur patience et leur aide.

Toutes les personnes du LIFC et de l'IUT de Belfort-Montbéliard que j'ai eu l'occasion de côtoyer, pour leur sympathie et l'aide qu'ils m'ont apporté, ainsi que pour l'accueil qui m'a été offert sur le site de Montbéliard.

L'ensemble des étudiants avec qui j'ai partagé de très bons moments tout au long de ces années.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

Contexte scientifique et institutionnel

Cette thèse a été effectuée à Montbéliard, dans le Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC), au sein de l'équipe OMNI Optimisation, Mobility, Networking. Elle a débuté en septembre 2004.

Ma thèse a été encadrée par Monsieur François Spies, Professeur, et par Monsieur Eugen Dedu, Maître de conférences dans le laboratoire LIFC. Elle a été financée par la Communauté d'Agglomération du Pays de Montbéliard (CAPM) et par l'Université de Franche-Comté. Elle présente mes travaux sur l'optimisation et l'adaptation des communications dans un réseau hétérogène.

Cette thèse s'inscrit dans la thématique de l'équipe OMNI, qui est d'étudier et de proposer des solutions dans la manière de concevoir les réseaux mobiles et de transmettre des données.

Cette thèse a donné naissance à plusieurs articles et présentations.

Conférences internationales :

1. S. Linck, E. Dedu, F. Spies. Distance-Dependent RED Policy (DDRED). In ICN, Sainte-Luce, Martinique, April 2007.
2. S. Linck, E. Mory, J. Bourgeois, E. Dedu, F. Spies. Video quality estimation of DCCP over wireless networks. In PDP, Montbéliard, France, February 2006.
3. E. Dedu, S. Linck, F. Spies. Removing the MAC retransmission times from the RTT in TCP. In Euromedia, Toulouse, France, April 2005.

Introduction générale

Depuis le commencement de l'Internet, le type de données transitant sur le réseau s'est très largement étendu. Les simples transferts de texte brut ont fait place aujourd'hui à des transferts de type multimédia beaucoup plus complexes dans leurs structures et leurs caractéristiques. Les données qui nous intéressent dans ce rapport sont les données les plus complexes de par leur composition : les vidéos.

Mais cette progression des transferts est à mettre en relation avec la progression des technologie de l'information : là où il y a une dizaine d'année le grand public accédait au réseau Internet par un simple modem RTC, aujourd'hui une majorité des internautes a accès à des réseaux haut débit asymétriques ou non et sur différentes technologies (xDSL, câble, fibre optique, ...)

Pour transférer un fichier de type vidéo sur le réseau, il existe aujourd'hui deux modes de lecture du flux et donc deux modes de transmission associée :

Le playback : Le système télécharge un fichier complet sur l'ordinateur de l'utilisateur puis exécute l'application permettant de lire le fichier. Les données sont donc lues depuis la machine locale. La transmission de la vidéo est équivalente à une transmission de fichier simple pas forcément multimédia.

La transmission d'une vidéo progressivement (ou streaming) : Si un serveur transmet une vidéo vers un client en temps réel, celle-ci est lue au fur et à mesure de sa transmission par ce dernier. Les données de la vidéo sont décodées puis lues par l'application cliente. Le flux de données est permanent pendant la lecture, réception et lecture sont conjointes à un même instant mais pas sur les mêmes données. On parle de *streaming* car le client n'a accès qu'aux données déjà transmises ou présentes à l'instant t. Une lecture de fichier en streaming suppose une transmission adaptée au type de données et d'actions possibles sur ces données (pause, lecture, retour avance, ...)

Mais pour cette méthode de diffusion existe à nouveau plusieurs formes :

La diffusion statique La manière statique de diffuser du streaming est de préparer le fichier en le formatant pour le streaming et de le déposer sur un site Web ou Intranet. Dès la demande de lecture du fichier votre lecteur va reconnaître le format, mettre une partie de la vidéo en mémoire et ensuite commencer sa lecture.

La diffusion dynamique Par opposition au streaming statique, le streaming dynamique nécessite un serveur spécialisé ou mixeur. Ce serveur ajustera la vidéo à la qualité de la bande passante du client grâce au format du fichier qui contiendra plusieurs qualités de compressions pour satisfaire toutes les qualités de connexions possibles des clients. Un client avec un modem 56Kbps pourra visualiser le fichier de la même manière que le ferait un client avec une connexion plus rapide parce que le serveur fournira les deux compressions nécessaires à ses deux clients. La qualité du flux s'ajustera aussi à la qualité de bande passante lorsqu'elle variera dans le temps.

Nous nous plaçons dans le cadre du *streaming* vidéo avec adaptation. Cette diffusion d'information se fait dans un environnement de communication informatique mixte : le réseau est constitué d'éléments actifs (serveurs, routeurs, ...) connectés par un réseau filaire et d'équipements reliés par des liaisons sans fil.

Les applications vidéo ne se développent pas sur internet autant qu'elles le pourraient car il n'existe pas pour l'instant de méthode de transport véritablement adaptée.

L'originalité du protocole de transport que nous proposons est de s'adapter aux conditions d'utilisation du réseau traversé, car ni UDP, ni TCP ne sont adaptés à cette tâche. Pour cela, les équipements actifs du réseau appliquent des stratégies de congestion prioritaires permettant de dégrader la qualité de la vidéo et retardant au maximum la coupure du flux vidéo. Pour tirer au mieux parti de cette nouvelle fonctionnalité de la couche transport, les algorithmes de codage vidéo doivent cependant être modifiés.

La transmission de données au niveau réseau est gérée principalement par les protocoles de transport (couche 4 de l'architecture OSI). De nos jours, la vidéo transmise en streaming sur Internet utilise des protocoles qui ne sont pas adaptés aux données multimédias. Ceux-ci ne tiennent pas compte des propriétés du flux vidéo.

Le contrôle de flux en matière de transfert multimédia temps réel est ce qui différencie principalement ce type de flux des transferts de flux plus simple. En effet :

- les pertes sont acceptables jusqu'à un certain taux : à 25 images par seconde, une image perdue toutes les secondes a un impact limité sur la qualité visuelle de la vidéo ;
- il y a des contraintes temps-réel : une image perdue dont le temps d'affichage est passé n'a plus d'intérêt d'être retransmise ;
- si la vidéo est composée de plusieurs flux (audio, vidéo, texte, etc.), la synchronisation doit être prise en compte.

Dans le cas d'une transmission de flux multimédia en temps réel, on remarque qu'au dessus d'IP, les protocoles de transport actuels ne sont pas adaptés. Ceux qui sont utilisés principalement à l'heure actuelle dans le cas de transmission vidéo sont le plus souvent TCP (Transmission Control Protocol) ou UDP (User Datagram Protocol). Ces protocoles ont des caractéristiques très différentes (par exemple le premier garantit une fiabilité de 100% alors que le deuxième n'offre pas une garantie de fiabilité). Les protocoles de vidéo proposés dans la littérature, comme RTP (Real Time Protocol), sont d'un niveau supérieur à UDP et en pratique, se basent sur celui-ci. Ainsi, ils rajoutent une surcharge au niveau transport. De même, UDP étant par définition non fiable, le réseau le traite de manière moins prioritaire (par rapport à TCP par exemple). D'autres protocoles étudiés dans le domaine de la recherche comme SCTP (Stream Control Transmission Protocol), TCP-CM (TCP for Continuous Media) ou DCCP (Datagram Congestion Control Protocol) s'attachent à la transmission de flux multimédia plus spécifiquement sans toutefois prendre en compte toutes les particularités de ce flux vidéo.

Un protocole de même niveau que UDP ou TCP, avec une gestion des caractéristiques de la vidéo (délai, synchronisation, ...) et un contrôle de congestion adapté, permettrait de combler les inconvénients présents dans le différents protocoles actuels. Ce protocole devra être compréhensible par l'ensemble des nœuds du réseau, que ce soit le serveur, le client ou tout autre

intermédiaire réparti entre ces extrémités (routeurs, points d'accès, ...).

Contributions et organisation du document

La motivation de cette thèse est l'absence de mécanismes de contrôle de congestion et de transport qui répondent correctement aux exigences d'applications spécifiques. Pour cette raison, il est essentiel de savoir comment interviennent ces mécanismes en différents points du réseau. L'importance de connaître précisément les limites de l'existant permet d'améliorer les mécanismes et de concevoir des stratégies optimales pour leurs mises en œuvre.

Dans un premier temps, nous nous sommes intéressés à l'amélioration du contrôle de la congestion au niveau d'une architecture de réseau filaire. Nous avons créé une extension pour les politiques de gestion de files d'attente au niveau des routeurs prenant en compte l'historique et l'ancienneté d'un flux dans ce réseau. Nous avons intégré cette extension dans une telle politique reconnue nommée RED (*Random Early Detection*) pour former une nouvelle politique nommée DDRED (*Distant-Dependent RED*). Cette dernière nous permet d'augmenter les ressources réseaux disponibles en rejetant judicieusement certains paquets de données plutôt que d'autres.

Mais si le cœur de réseau reste filaire, le réseau d'accès tend vers des systèmes sans fil comme le Wi-Fi. C'est pourquoi, dans un deuxième temps, nous proposons une optimisation des protocoles classiques de transport pour ces nouveaux liens. Ceux-ci sont sujets plus que d'autres aux perturbations et à l'environnement et entraînent ainsi des variations dans les caractéristiques premières de la liaison (bande passante, taux d'erreurs, etc.). Nous nous proposons de prendre en compte les perturbations et ainsi de différencier les erreurs dues à des problèmes d'interférences, de celles dues à une congestion sur le réseau sans fil ou sur le réseau filaire en amont. On étudie plus spécifiquement l'augmentation des temps de transmission. Cette dernière est causée soit par une augmentation des files d'attente des routeurs et donc c'est une congestion, soit par les retransmissions générées par la couche MAC dans les réseaux 802.11abg. Dans ce cas il ne s'agit pas d'une congestion mais d'une simple perturbation, ce qui ne demande pas la réaction de l'émetteur.

Enfin,après s'être intéressé aux problèmes techniques, nous avons conçu un système applicatif complet de diffusion vidéo. Il permet de proposer une qualité de *streaming* optimale et dépendant directement du client et/ou de la bande passante disponible à chaque instant pour la vidéo diffusée. Il utilise DCCP (Datagram Congestion Control Protocol) qui est un protocole de transport récent et modulable au niveau transport comme au niveau contrôle de congestion. Cette qualité est obtenue grâce à une communication entre les couches transport et application de la pile OSI : sur la source, DCCP détermine la bande passante disponible et l'encodeur vidéo choisit ou non si un réencodage à la volée de la vidéo est nécessaire.

D'un point de vue technique, toutes ces améliorations ont été développées et validées grâce à un simulateur de réseau : Network Simulator v2. Un tel outil nous permet de construire rapidement un réseau hétérogène et de tester sur celui-ci différentes théories tout en ayant les mêmes conditions initiales de réseau, ce qui est impossible lors d'expérimentations réelles, sur un réseau sans fil par exemple.

Comme nous venons de le présenter, cette thèse a pour objectif la spécification, la modélisation et l'évaluation sur des outils de simulation de ces propositions.

Première partie

Approche globale de la congestion

CHAPITRE 1

Etat de l'art de la gestion des files d'attente

Sommaire

1.1 Introduction

Dans ce chapitre, nous allons nous intéresser à la gestion des communications dans les équipements qui forment le cœur de tout réseau informatique y compris le plus grand de tous, Internet. Ce cœur de réseau est principalement composé d'une multitude d'équipements appelés des routeurs. Un routeur peut se définir comme un outil logiciel ou matériel qui permet de diriger les données à travers un réseau.

On peut identifier différents types de flux aujourd'hui sur le réseau Internet et le catégoriser simplement de la manière suivante :

- Les flux TCP
  - pour le transfert de données importantes comme un transfert FTP par exemple,
  - de type transactionnel pour la création d'une connexion, les requêtes HTTP, ...
- Les flux UDP
  - de type transactionnel comme les échanges DNS ou ARP,
  - pour la diffusion de vidéos
    - en directe,
    - avec un contrôle de congestion applicatif avec ou sans dégradation du contenu,
    - avec un tampon de réception pour anticiper la baisse de débit (échange le plus souvent fait sur HTTP)

Tous ces types de transfert passent par les mêmes équipements réseaux (câbles, routeurs, commutateurs, ...). Tant que le routeur dispose de ressources physiques (CPU, mémoire, bande passante en amont et en aval, ...) suffisantes, aucune intervention n'est nécessaire : le système peut traiter tous les flux de données entrants et sortants. Par contre dans certains cas, le système peut entrer en phase de congestion c'est-à-dire que ce système, et plus particulièrement les routeurs qui le composent, sont dans un état dans lequel les ressources du réseau (mémoire, liaisons entre les routeurs) sont pratiquement toutes occupées.

Nous nous intéressons ici à la gestion des liens et plus précisément, aux files d'attente des routeurs qui en constituent les extrémités. Une file d'attente se définit un hébergement temporaire pour les messages ou paquets attendant d'être traiter. Chaque file d'attente représente un ensemble d'éléments traités par un système dans un ordre spécifique. Il existe deux niveaux d'études des files d'attente : les mécanismes d'ordonnancement et ceux de gestion de ces files. L'ordonnancement est la fonction qui distribue les ressources entre les différentes classes de service. La gestion de file d'attente est la fonction qui, au sein d'une même classe, détermine quels paquets éliminer en cas de congestion.

Ce chapitre présente différents mécanismes d'ordonnancement et de gestion de file d'attente qui peuvent être utilisés pour mettre en œuvre une politique particulière de qualité de service.

1.2 L'ordonnancement des paquets

Quelle que soit la politique de qualité de service utilisée, c'est au niveau des files d'attente des routeurs que cette politique est mise en œuvre.

La qualité de service permet de traiter les paquets d'une manière différenciée contrairement à la méthode *best-effort*. Il existe plusieurs types d'ordonnanceurs suivant les niveaux de qualité attendue.

1.2.1 FIFO (*First In First Out*)

Par défaut, la politique sur Internet est de type *best-effort* : tous les paquets arrivant dans un routeur sont placés dans une file d'attente de type *First In First Out* (FIFO) et pour un traitement équitable des flux.

Cet algorithme ne s'applique que dans le cas de la présence d'une seule file d'attente. Les paquets sont ajoutés en queue de la file d'attente et c'est le paquet en tête de file qui est traité en premier. Cependant cette gestion des paquets entraîne une variation des délais (gigue) entre l'arrivée des paquets dans le cas où plusieurs flux sont présents.

Tant que le système ne subit pas de congestion, le traitement des files d'attente est basé sur une file FIFO. Par contre en cas de congestion, de nouveaux ordonnancements s'appliquent suivant les besoins en qualité de service qui ont été définis sur un routeur donné ou sur l'ensemble d'un réseau.

1.2.2 FQ (*Fair Queueing*)

Cet algorithme permet le partage équitable des ressources entre les différents flux. Si deux files d'un routeur utilisent FQ, les flux passant dans chaque file d'attente auront le même débit en cas de congestion. FQ prend en compte la taille de paquets en donnant une plus grande priorité aux petits paquets.

1.2.3 WFQ (Weighted Fair Queueing)

WFQ est basé sur FQ. Il permet de donner une priorité pondérée à certains flux. WFQ permet de séparer le trafic arrivant dans un routeur en plusieurs flux, et d'allouer à chaque flux une fraction de la bande passante. Le routeur va créer dynamiquement une file par flux détecté. Malgré ses avantages, tels la gestion des priorités dynamiques des flux, WFQ est coûteux en ressources CPU car le routeur doit calculer à chaque émission combien de paquets de chaque file seront émis.

1.2.4 PQ (*Priority Queueing*)

PQ permet d'affecter une priorité stricte à une file d'attente. Les flux entrant sont classés par types de protocole (IP, IPX, SNA, ...), ports d'entrées, ...

Ainsi si le routeur possède trois files A, B et C, et qu'il est configuré avec un algorithme PQ spécifiant que A est prioritaire sur B qui est elle-même prioritaire sur C on aura le fonctionnement suivant :

- un paquet de la file B ne sera servi que si la file A est vide ;
- un paquet de la file C ne sera servi que si les files A et B sont vides.

Cet algorithme est intéressant pour gérer le trafic critique mais ne doit pas être utilisé pour un nombre important de flux car les flux non prioritaires seraient systématiquement rejetés.

1.2.5 CBQ (*Class Based Queueing*)

CBQ permet d'allouer une certaine proportion de bande passante pour une classe de trafic donnée. Les classes peuvent être basés sur une variété de paramètres, telles que la priorité, l'interface, ou l'application originaire. CBQ abaisse la bande passante réelle au débit configuré en calculant le temps qui devrait s'écouler entre des paquets de taille moyenne pour atteindre ce débit. Il est aussi possible de spécifier combien de bits ou de paquets seront émis à chaque passage dans une file active.

CBQ est intéressant si l'on souhaite allouer de manière fixe une partie de la bande passante à un flux spécifique ou à une classe de flux. CBQ est plus simple que FQ car il ne fait pas de distinction dans la taille des paquets. Il est plus efficace que PQ car le système de classe permet une gestion plus précise des paquets et ne se limite pas à des priorités strictes. Par exemple on peut réserver 30% de la bande passante à un flux vidéo et laisser le reste en *best-effort*.

1.2.6 Récapitulatif des ordonnanceurs

Nom	Avantages et Inconvénients
FIFO First In First Out	Simple, rapide peu gourmand en CPU Pas de qualité de services
FQ Fair Queue	Partage équitable de la bande passante Pas de gestion de la priorité
WFQ Weighted Fair Queue	Priorités dynamiques en fonction des flux Coûteux en ressources CPU
PQ Priority Queue	Simple, rapide et peu gourmand en ressources CPU Risque de famine pour les flux avec une priorité basse
CBQ Class Based Queueing	Allocation d'une bande passante fixe par flux Coûteux en ressources CPU

TABLE 1.1 – Tableau comparatif récapitulatif des ordonnanceurs

1.3 La gestion classique des files d'attente

Dans la gestion des files d'attente sur les routeurs, on distingue deux méthodes :
- la gestion passive où l'on ne s'occupe de savoir que s'il reste de la place dans la file d'attente avant d'accepter un paquet,
- la gestion active ou *Active Queue Management* (AQM) qui est obtenu par un retour d'information aux utilisateurs (sources et destinations) sur l'état du réseau par l'élimination de paquets dans les files d'attente des routeurs. Les émetteurs de trafic sur le réseau réagissent alors à la perte de paquets en réduisant la quantité de données envoyées.

Ces deux méthodes s'illustrent très bien par les deux exemples représentatifs de chacune d'elle : Drop Tail pour le gestion passive et RED (Random Early Detection) pour la gestion active.

1.3.1 Gestion passive : Drop Tail

Le comportement le plus utilisé pour la gestion des files d'attente des routeurs sur Internet est appelé *Drop Tail* ou *Tail Drop* (traduction : « élimine le reste »). La politique Drop Tail consiste à mettre en file d'attente les paquets qui arrivent, et à rejeter tous les paquets quand la taille de la file d'attente dépasse la taille du tampon.

La probabilité de rejet assez simple de Drop Tail $d(k)$ est donnée par l'équation suivante :

$$d(k) = \begin{cases} 0 \ si \ k \leq K \\ 1 \ si \ k > K \end{cases} \tag{1.1}$$

où k est la taille de la file d'attente et K la taille du tampon.

Cela conduit à des phénomènes de retransmissions de synchronisation pour les flux TCP par exemple. Quand une retransmission de synchronisation a lieu, la brusque rafale de rejets d'un routeur qui a atteint sa limite entraînera une rafale de retransmissions retardée qui inondera à nouveau le routeur congestionné.

Dans le but de régler les problèmes de congestion due à des rafales de paquets, les routeurs intègrent souvent des files d'attente de grande taille. Malheureusement, bien que ces files d'attente

offrent un bon débit, elles peuvent augmenter sensiblement les temps de latence et entraîner un comportement très saccadé des connexions TCP pendant la congestion.

Les files Drop Tail sont monopolisées par certains types de flux et empêchent ainsi les autres transmissions de trouver une place dans la file. C'est le problème du *lock-out* de Drop Tail : tous les flux n'ont pas la même chance d'entrer dans la file d'attente. Les problèmes avec Drop Tail deviennent de plus en plus préoccupants avec l'augmentation de l'utilisation d'applications « hostiles »au réseau comme les applications de diffusion vidéo.

Il existe plusieurs autres techniques, côté routeur, pour obtenir une utilisation optimale des ressources et/ou adapter les applications aux conditions dynamiques du réseau.

1.3.2 Gestion active : Random Early Detection (RED)

Sam Floyd et Van Jacobson ont donc défini RED (*Random Early Detection*). Ils l'ont développé afin de proposer beaucoup d'avantages par rapport à Drop Tail, notamment dans le cas de flux agressifs. La méthode RED [**?**] permet de détecter la congestion au niveau d'un routeur avant que cette dernière ne se produise. Le routeur surveille la taille de sa file d'attente et suivant sa valeur, rejette suivant une certaine probabilité des paquets à leur arrivée par mesure préventive afin que ces flux pénalisés réduisent leur débit.

Plutôt que d'attendre que la file soit pleine, on élimine un paquet qui arrive avec une certaine probabilité d'élimination (*drop probability*) quand la file d'attente atteint un certain niveau (*drop level*). La taille moyenne de la file d'attente est calculée selon une moyenne exponentielle *Exponentially Weighted Moving Average* (EWMA) Pour calculer la longueur moyenne de la file d'attente *AvgLen*, on applique la formule suivante :

$$AvgLen = (1 - Weight) \times AvgLen + Weight \times SampleLen \tag{1.2}$$

Avec $0 < Weight < 1$ (habituellement 0.002) et $SampleLen$ qui est la longueur de la file à chaque fois qu'un nouveau paquet arrive.

La file d'attente d'un routeur est définie par les deux valeurs de la taille de cette file que sont les seuils $MinThreshold$ et $MaxThreshold$ (voir **??**). En fonction de la longueur moyenne de la file et des seuils de celle-ci, plusieurs actions sont possibles :

- Si $AvgLen \leq MinThreshold$ alors mettre le paquet dans la file d'attente
- Si $MinThreshold < AvgLen < MaxThreshold$ alors
  - calcul de la probabilité P
  - élimination du paquet avec la probabilité P
- Si $MaxThreshold \leq AvgLen$ alors éliminer le paquet

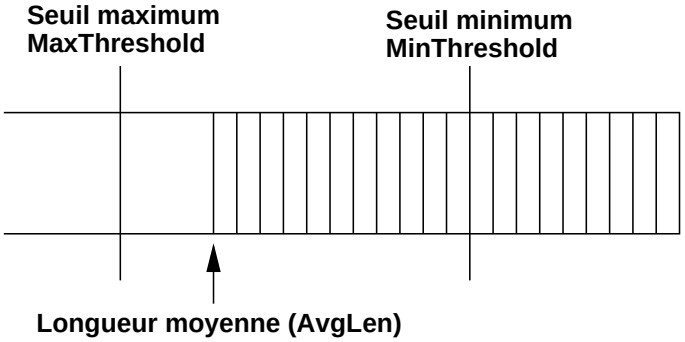


FIGURE 1.1 – Schéma d'une file d'attente

$$P = \begin{cases} 0 \; si \; AvgLen < MinThreshold \\ 1 \; si \; AvgLen > MaxThreshold \\ \frac{AvgLen - MinThreshold}{MaxThreshold - MinThreshold} \times P_{max} \; sinon \end{cases} \tag{1.3}$$

Le calcul de la probabilité P de rejet de paquet s'exprime de la manière suivante : (voir graphique **??**)

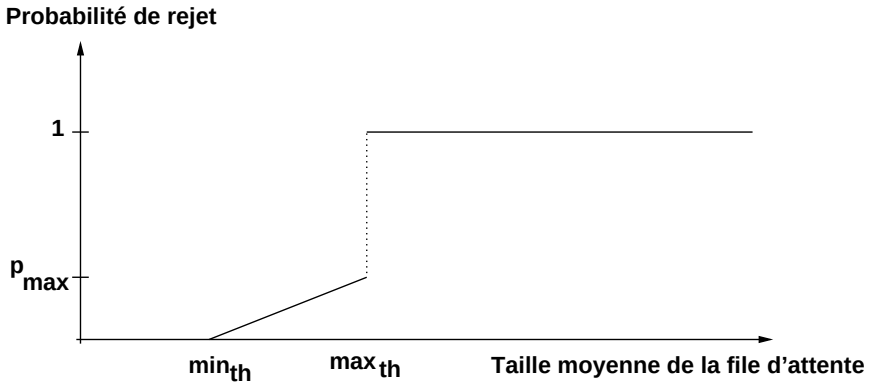


FIGURE 1.2 – Courbe représentant la probabilité d'élimination P dans RED

Défauts de RED

Configuration

Bien que RED est été en avant pour un déploiement à l'échelle de l'Internet, des lignes directrices pour la configuration des paramètres de RED sont toujours sujettes à discussion.

Des études [**?**, **?**] ont montré que l'attention portée au réglage des paramètres de RED est nécessaire afin d'obtenir les performances promises par RED et ces files d'attente par rapport à *Drop Tail*. Comme les performances de la file d'attente dépendent de cette configuration correcte, ceci entraîne aussi un problème d'évolutivité et de mise à l'échelle.

Prise en compte d'autres informations

Les paramètres de RED ne lui permettent pas de prendre en compte les données contenues dans le paquet ou même simplement le type d'application associée à chaque paquet de la file d'attente.

Si on prend comme exemple ce réseau en ligne (figure **??**). On considère un flux fixe que l'on qualifiera de « long »entre les machines numérotées 1 et 14. Tous les routeurs de cet exemple implémentent comme politique de gestion de file d'attente RED. On met en concurrence un flux « court »en terme de distance source-destination (et donc de RTT) avec le long que l'on positionne de différentes manières :

- Le flux court est au début du flux long (2 vers 3)
- Le flux court est au milieu du flux long (7 vers 8)
- Le flux court est à la fin du flux long (12 vers 13)

Pour chaque cas, on examine la bande passante de chaque flux (voir figure **??**). Le flux court est démarré en $t = 4s$ et arrêté en $t = 12s$. On remarque clairement que dans tous les cas de concurrence, la priorité est donnée au flux court.

D'autres mécanismes de gestion des files d'attente se sont développés à partir du principe de RED.

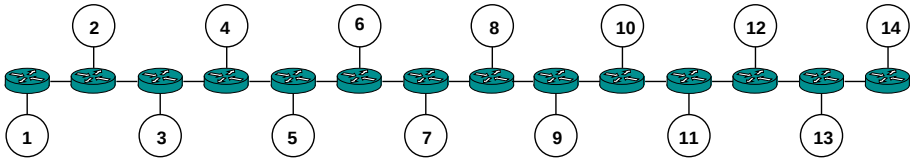


FIGURE 1.3 – Exemple de test

1.4 Les dérivées de RED

Un certain nombre de chercheurs ont proposé des modifications ou des alternatives à RED dans le but de résoudre ses lacunes et améliorer ses performances. Les variantes portent sur la modification des seuils de contrôle et/ou de la fonction de calcul de probabilité. La longueur de la file d'attente (ou longueur moyenne de la file d'attente) est largement utilisé dans RED et la plupart de ses variantes (ARED, Gentle RED, ...) font de même. D'autres, comme le SRED utilisent le cas de dépassement de la file ou les cas où la file d'attente est vide.

Sur la base de leurs paramètres de contrôle (seuils, taille de la file, ...), les variantes de RED peuvent être classées dans les principales catégories suivantes :

- RED avec de meilleures performances (WRED, DSRED, ARED, HRED),
- RED avec une meilleure stabilité (Gentle RED, SRED, FRED, XRED, MRED).

1.4.1 Amélioration de performance

Weighted RED (WRED)

WRED [**?**] (Weighted RED) de Cisco inclut une préférence dans RED en fournissant des seuils et les poids différents pour différentes adresses IP. En attribuant un P_{max} et des seuils $MinThreshold$ et $MaxThreshold$ différents, il est possible d'offrir une distribution différente des pertes par flux. WRED peut réaliser cette fonction en utilisant un jeu de paramètres pour chaque niveau de priorité.

Comme RED, WRED ne résiste pas à un facteur d'échelle trop important car il est nécessaire de stocker les différents paramètres pour chaque flux traversant le routeur. Dans WRED, le calcul de l'occupation moyenne reste celui de RED. A l'arrivée de chaque paquet, $AvgLen$ est actualisée, par contre, c'est dans le calcul de P_{max} que la différenciation s'effectue.

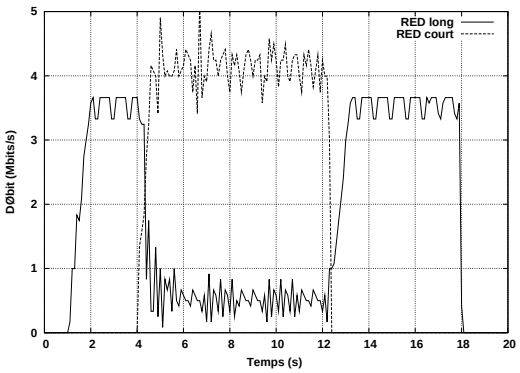
La figure **??** montre un choix de paramètres qui pourrait servir pour WRED. Elle illustre deux types de comportement de la file d'attente :

- Pour les paquets de faible importance (avec $P_{max} = P_{max2}$, l'algorithme est très agressif : il ne leur permet pas de traverser le routeur dès que la file dépasse une limite relativement courte.
- Par contre, pour les paquets plus importants ($P_{max} = P_{max0}$, la politique en autorise plus à passer.

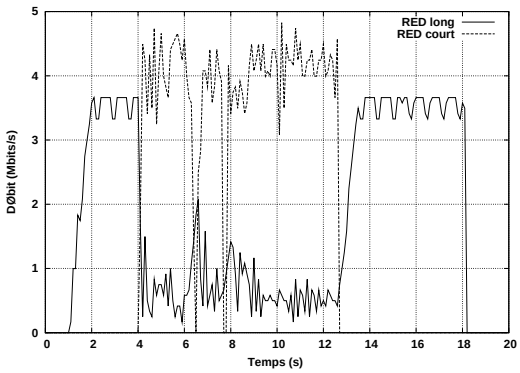
Double Slope RED (DSRED)

Double Slope RED (DSRED) [**?**] a été créé dans le but d'améliorer les performances de RED en ce qui concerne le débit et les délais des paquets transmis. Comme Gentle RED, DSRED a une courbe de probabilité de rejet continue qui change dynamiquement en fonction du niveau de congestion.

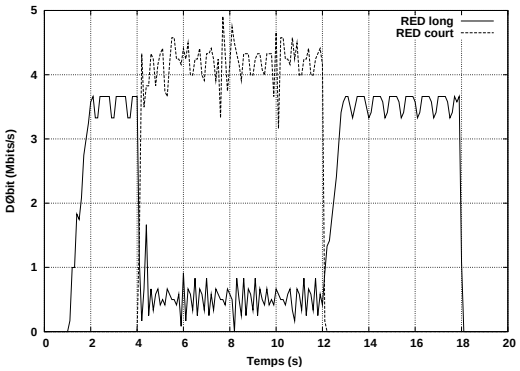
DSRED définit de nouvelles variables pour le calcul de la probabilité de rejet (voir figure **??** :



(a) Concurrence au début



(b) Concurrence au milieu



(c) Concurrence à la fin

FIGURE 1.4 – Bandes passantes de flux TCP New Reno en concurrence.

- K_l l'équivalent de $MinThreshold$
- K_m le seuil de changement de fonction de rejet
- K_h l'équivalent de $MaxThreshold$
- α la pente de la courbe de rejet entre K_l et K_h
- β la pente de la courbe de rejet entre K_h et K_m
- γ le paramètre définissant les deux pentes α et β des courbes de rejet

Comme RED, DSRED utilise la taille moyenne de la file d'attente K_{cur} comme indicateur de niveau de congestion.

$$p_{DSRED}(K_{cur}) = \begin{cases} 0 & si\ K_{cur} < K_l \\ \alpha(K_{cur} - K_l) & si\ K_l \leq K_{cur} < K_m \\ 1 - \gamma + \beta(K_{cur} - K_m) & si\ K_m \leq K_{cur} \leq K_h \\ 1 & si\ K_h \leq K_{cur} < N \end{cases} \tag{1.4}$$

α est la pente de la courbe de rejet correspondant à une taille moyenne de la file d'attente comprise entre K_l et K_m. De même β est celle qui correspond à une taille comprise entre K_m et K_h. Elles sont définies de la manière suivante :

$$\alpha = \frac{2(1-\gamma)}{K_h - K_l} \tag{1.5}$$

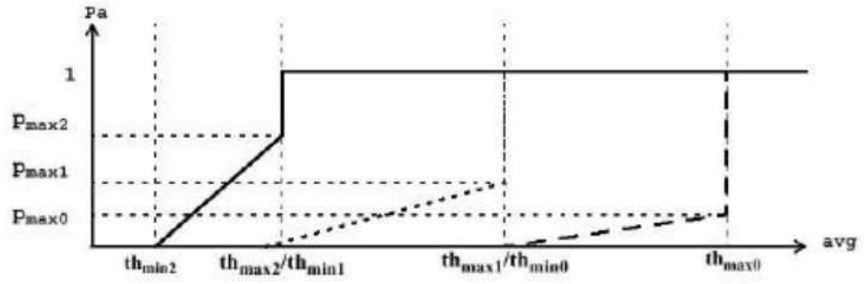


FIGURE 1.5 – Courbes représentants différents exemples de gestion des pertes dans WRED

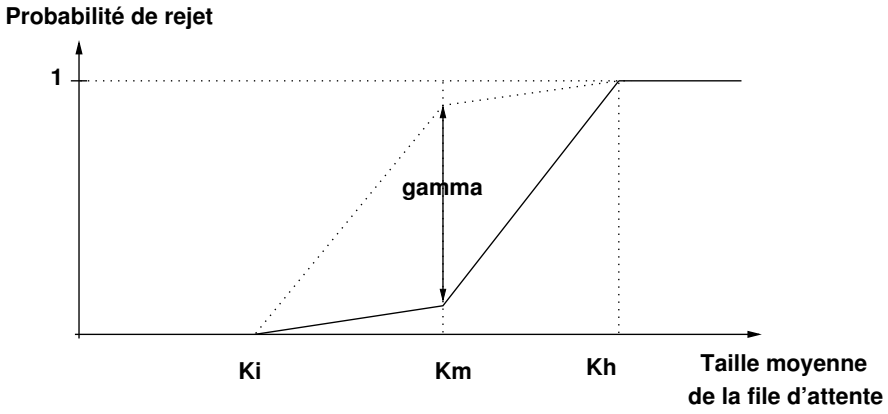


FIGURE 1.6 – Courbe représentant différents niveaux de perte dans DSRED

$$\beta = \frac{2\gamma}{K_h - K_l} \tag{1.6}$$

γ permet d'obtenir différents modes de comportement de DSRED. Il permet à DSRED d'intervenir dans différentes situations de congestion. On peut ainsi augmenter le taux de rejet avec une probabilité plus élevée en cas d'augmentation du nombre de congestion et de réduire ce dernier en cas de niveau de congestion bas.

Adaptative RED (ARED)

Dans [**?**], Feng propose une version adaptative de RED (Adaptative RED notée ARED) afin de maximiser l'utilisation des liens et réduire le taux de perte. Durant une congestion lourde, afin d'éviter la perte de paquets due au débordement de sa file, RED doit signaler la congestion à un nombre suffisant de sources de sorte que sa charge soit réduite. Afin d'empêcher une sous-utilisation soudaine du lien, RED ne doit pas avertir toutes les sources dont les paquets le traverse.

Avec N sources partageant un même goulot d'étranglement, envoyer un avis de congestion à une source permet de réduire la charge par un facteur de $(1 - \frac{1}{2N})$ puisque TCP réduit la taille de sa fenêtre de congestion de moitié. Pour une valeur grande de N, l'impact d'un seul avis de congestion est minime mais il augmente pour une petite valeur de N. Par conséquent, pour N grand, comme RED n'est pas assez agressif, la file d'attente reste presque entièrement occupée et elle se comporte ainsi comme Drop Tail. Par contre si N est petit, comme RED est

trop agressif, trop de sources reçoivent les avis de congestion et comme elles réduisent leur taux de transmission, on a une sous-utilisation des liens.

Pour optimiser les performances, les auteurs ont fait les recommandations suivantes : pour un petit nombre de connexions, le dépistage précoce devrait être conservateur afin de permettre un taux élevé d'utilisation des liens. Mais en cas de grand nombre de connexions, une sur-utilisation des liens est obtenue quelle que soit la valeur de P_{max}. Cependant, le mécanisme de détection de congestion devrait être agressif afin de réduire le taux de perte de paquets.

En conséquence, les auteurs concluent qu'une adaptation du paramètre P_{max} selon la charge du routeur améliore l'exécution RED et propose l'algorithme ARED à cette fin. L'algorithme emploie la longueur moyenne de file d'attente pour influencer le comportement agressif ou conservateur de la détection précoce de congestion. Si la taille moyenne de file d'attente est autour de $MinThreshold$, le mécanisme de détection est trop agressif. En conséquence, la valeur de P_{max} est diminuée d'un facteur α constant. D'un autre côté, si la taille moyenne de file d'attente est autour de $MaxThreshold$, cela signifie que la détection de congestion est trop conservatrice. Dans ce cas, la valeur de P_{max} est augmentée d'un facteur β. Cependant, si la taille moyenne de file d'attente oscille entre $MinThreshold$ et $MaxThreshold$ les auteurs n'adaptent pas P_{max} parce qu'ils considèrent que le mécanisme de détection de congestion se comporte comme désiré et peut éviter un taux de perte élevé de paquet et la sous-utilisation des liens.

Hybrid RED

Comme dans toutes les variantes de RED, [**?**] pose la question du choix de l'indicateur « taille de la file d'attente »dans le marquage ou la perte d'un paquet. Les auteurs proposent l'utilisation conjointe des tailles instantanées des files d'attente et de leur moyenne exponentielle (EWMA) respective pour la gestion des paquets.

L'utilisation conjointe de ces deux informations pour le marquage et le rejet de paquets permet de diminuer l'utilisation des ressources CPU du routeur pour le calcul de EWMA lorsque la taille est faible et ensuite d'utiliser les seuils classiques ($MinThreshold$) pour un certain nombre de paquets consécutivement arrivés. Ainsi, de nouvelles directives pour améliorer le marquage ou le rejet de paquet et obtenir une réponse plus rapide des algorithmes de type RED sont définis. L'algorithme résultant a été testé avec des simulations NS2 selon différents scénarios, et il offre une meilleure utilisation de la bande passante du réseau et fait diminuer le taux de perte de paquets.

1.4.2 Augmentation de la stabilité

Gentle RED

Dans la suite de ce mémoire nous utiliserons souvent en simulation la politique *Gentle RED* qui a été introduit par Vincent Rosolen, Olivier Bonaventure et Guy Leduc [**?**].

Cet algorithme est en tout point identique à RED en prenant une fonction de calcul de probabilité qui ne fait pas de saut à l'ordonnée P_{max} , mais qui croît linéairement de P_{max} en $MaxThreshold$ à 1 à la taille maximale moyenne de la file d'attente (voir figure **??**).

Elle est souvent définie comme étant deux fois $MaxThreshold$. Cette version de RED présente l'avantage de ne pas introduire de discontinuité dans l'évolution des rejets de paquets.

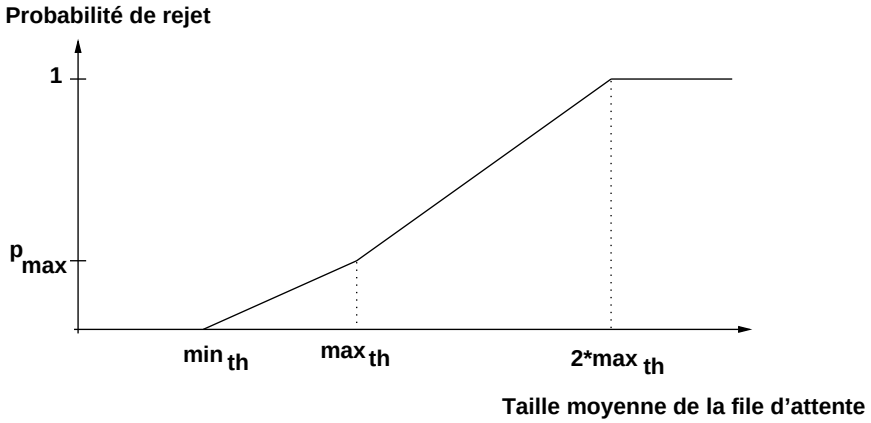


FIGURE 1.7 – Courbe représentant la probabilité d'élimination P dans Gentle RED

Stabilized RED (SRED)

SRED (Stabilized RED) est une variante de RED proposée par plusieurs auteurs dans [**?**]. Elle a pour but de stabiliser la taille des files d'attente d'un routeur. Plus précisément, SRED estime le nombre de connexions en utilisant une liste de « zombis »afin de sauvegarder les informations sur les flux qui ont récemment envoyé des paquets à travers le routeur.

Pour initialiser cette file, le routeur la remplit avec les identifiants de flux des différents paquets entrants. Pour chaque index de cette liste, le compteur correspondant est initialisé à la valeur zéro.

Une fois que la liste est pleine, quand un paquet arrive dans les tampons du routeur, il est comparé à un zombi aléatoirement choisi dans la liste de zombis. Si le paquet entrant a le même identifiant de flux que le zombi (on a une « touche »ou *hit*), alors le compteur du zombi est incrémenté de un. Avec la probabilité p, l'identifiant de flux du zombi est remplacé par l'identifiant du flux du paquet entrant ayant servi à la comparaison et le compteur du zombi est placé à 0. Avec la probabilité $1 - p$, il n'y a aucun changement dans la liste de zombis.

Ce mécanisme de « touche »peut être employé pour identifier les « mauvais »flux sans garder un état de chaque flux rencontré. En effet, ces flux sont plus enclin à causer des touches que les autres et le compteur d'un zombi venant de ces flux est élevé. Ils estiment le nombre de flux actifs à partir du taux moyen de touche. Pour cela, ils calculent d'abord la fréquence de hit $P(n)$ à l'arrivée du nième paquet selon la formule suivante :

$$P(n) = (1 - \alpha)P(n - 1) + \alpha Hit(n) \tag{1.7}$$

où α est une constante comprise entre 0 et 1 et :

$$Hit(n) = \begin{cases} 0 \ si \ pas \ de \ hit \\ 1 \ sinon \end{cases} \tag{1.8}$$

$1/P(n)$ est une bonne évaluation du nombre N de flux actifs jusqu'à l'arrivée du nième paquet. Au lieu d'utiliser la taille moyenne de file d'attente comme dans RED, SRED calcule la probabilité de rejet p_{zap} à partir de l'estimation du nombre de flux actifs et de la taille instantanée

q. La probabilité p_{SRED} est définie comme suit :

$$p_{SRED}(p_{zap}) = \begin{cases} 0 & si\ 0 \leq p_{zap} < B/6 \\ \frac{1}{4} \times p_{max} & si\ B/6 \leq p_{zap} < B/3 \\ p_{max} & si\ B/3 \leq p_{zap} < B \end{cases} \tag{1.9}$$

où B est la taille du tampon de la file d'attente, p_{max} est la probabilité maximale de rejet et N le nombre de flux actifs. On note que p_{SRED} a seulement trois niveaux possibles de rejet (0, $p_{max}/4$ et p_{max}), et que p_{SRED} ne prend pas en compte les variations de taille de la file d'attente q. La courbe **??** illustre ces niveaux de perte.

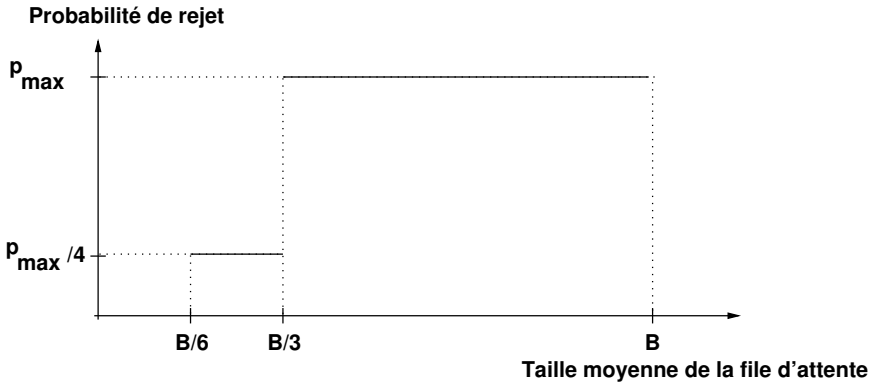


FIGURE 1.8 – Courbe représentant différents niveaux de pertes dans SRED

Les auteurs proposent deux versions du calcul de la probabilité de rejet dans SRED :
– le calcul simple qui ne tient compte que de la taille a un instant t de la file d'attente

$$p_{zap} = p_{SRED}(p_{zap}) \times min\left(1, \frac{N^2}{256^2}\right) \tag{1.10}$$

– la version « full SRED »où la probabilité de rejet p_{zap} dépend en plus du fait qu'un paquet soit responsable ou non d'un *hit*. Dans ce cas p_{zap} est une fonction croissante du nombre de flux actifs dans un routeur.

$$p_{zap} = p_{SRED}(p_{zap}) \times min\left(1, \frac{N^2}{256^2}\right) \times (1 + Hit(t) \times N) \tag{1.11}$$

Les auteurs présentent des résultats montrant que SRED stabilise l'occupation des files d'attente mais entraîne un débit plus faible en sortie de routeur.

Flow Random Early Drop (FRED)

Afin de prendre en compte le problème des flux non affectés par une congestion, Lin et Morris ont proposé dans [**?**] un mécanisme appelé Flow RED (FRED) pour protéger les routeurs de la congestion tout en réduisant l'effet d'injustice présent dans RED, notamment le problème d'équité entre les flux TCP et d'UDP.

Au lieu d'indiquer la congestion d'un routeur à des flux aléatoirement choisis en rejetant leurs paquets suivant une certain probabilité, FRED filtre les flux qui ont un grand nombre de paquets dans la file d'attente. Cette politique calcule un taux de perte de chaque flux selon la taille occupée par ce flux dans la file d'attente.

Si un flux occupe continuellement une grande quantité d'espace de la file d'attente, il est détecté et sa place dans la file d'attente est limitée. Le coût de cette gestion par flux est proportionnel à la taille de la file et indépendant du nombre de flux, dans la mesure où l'utilisation de la file d'attente ne dépend que du nombre de flux actifs, et non du nombre total de flux.

Les auteurs classent le type de trafic dans 3 catégories :

- Le trafic non adaptatif : c'est-à-dire qui est transporté par un protocole n'intégrant pas un contrôle de congestion comme par exemple les transmissions faites sur UDP : applications audio, vidéo, flux UDP, etc.
- Le trafic robuste : pour les connexions qui utilisent un protocole de transport fiable tel TCP, intégrant un contrôle de congestion. Ce trafic consomme de la bande passante jusqu'à ce qu'une congestion soit détectée (exemple : transfert de données FTP).
- Le trafic fragile : pour les connexions qui s'adaptent lentement à une augmentation de la bande passante disponible ou qui sont sensibles aux pertes de paquet (exemple : telnet).

L'algorithme de FRED retient les flux non adaptatifs en augmentant leur taux de perte, protège les flux fragiles et traite les flux robustes comme le ferait RED. Pour identifier les différents types de flux à l'intérieur du routeur, FRED a besoin les paramètres suivants :

- q : la taille actuelle de la file d'attente,
- Min_q : le nombre minimum de paquets de chaque flux dans la file d'attente,
- Max_q : le nombre maximum de paquets de chaque flux dans la file d'attente,
- $avgcq$: une évaluation du nombre moyen de paquet par flux. Des flux ayant moins de $avgcq$ paquets dans la file sont favorisés par rapport aux autres flux,
- $qlen$: le compteur de paquet pour chaque flux ayant des paquets dans la file,
- $strike$: le nombre de fois où un flux n'a pas pris en compte la congestion.

Le calcul de probabilité de rejet d'un paquet pour un flux i dans FRED est fait de la manière suivante (voir figure **??**) :

$$p_{FRED}(AvgLen) = \begin{cases} p_{RED} & si\ MinThreshold \leq AvgLen < MaxThreshold\ et\ qlen_i \geq MAX(min_q, avgc \\ 1 & si\ qlen_i \geq Max_q \\ & ouAvgLen \geq MaxThreshold\ et\ qlen_i > 2avgcq \\ & ouqlen_i \geq avgcq\ et\ strike_i > 1 \\ & ouAvgLen \geq MaxThreshold(commeRED) \\ 0 & sinon \end{cases} \tag{1.12}$$

La probabilité de perdre un paquet sur un routeur augmente avec le nombre de paquets non perdus depuis la dernière perte sur ce routeur. Cette perte d'un paquet suivant la probabilité RED intervient pour les flux robustes.

Pour protéger des flux fragiles, un paquet entrant est toujours accepté s'il a moins de Min_q paquets dans la file et si la taille moyenne de la file est inférieure à $MaxThreshold$. Autrement les paquets sont sujets à la probabilité de RED. À la différence du RED qui estime la longueur moyenne de la file d'attente à chaque arrivée de paquet et ne traite donc pas les cas où la file est vide, FRED fait sa moyenne à l'arrivée et au départ d'un paquet.

Par exemple si un paquet arrive au temps zéro quand la longueur instantanée et moyenne de la file d'attente ont la même valeur égale à 100 paquets et si les prochains paquets arrivent après le départ de 50 paquets, alors la taille instantanée de file d'attente sera égale à 50 tandis que la taille moyenne de file d'attente restera inchangée et égale à 100 paquets. FRED évite une telle erreur de calcul qui pourrait avoir comme conséquence la sous-utilisation des liens. De plus, FRED ne modifie pas la taille moyenne de file d'attente si le paquet entrant est rejeté à moins que la file d'attente soit vide. FRED calcule la longueur moyenne de $avgcq$ par flux en divisant

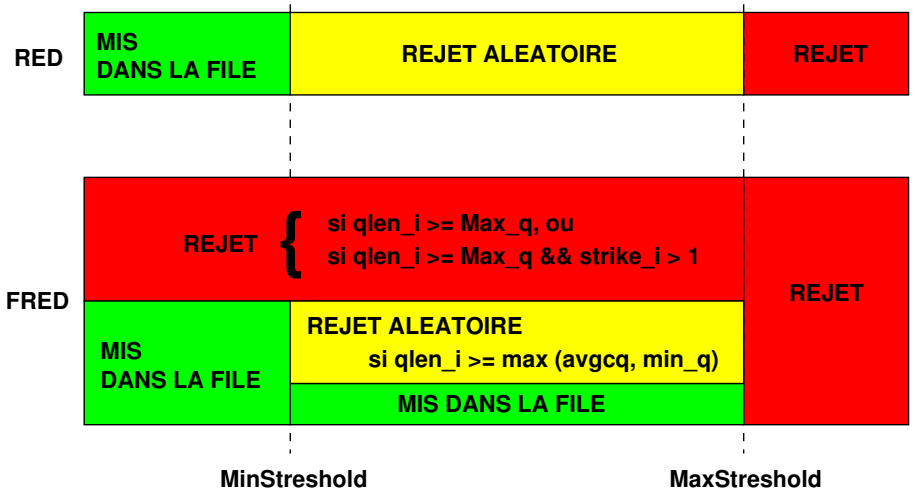


FIGURE 1.9 – Comparaison de la gestion des pertes entre FRED et RED

la longueur moyenne globale *AvgLen* par le nombre de flux actifs au même instant.

Du fait que FRED fournit une forme d'équité entre les flux, elle pourrait résister au problème de mise à l'échelle mais exige également un tampon suffisamment grand pour contenir assez de paquets afin de détecter les différents types de flux.

XRED

Hutschenreuther et Schill présente XRED dans [**?**]. XRED a pour but de réduire la perte de bande passante pour la transmission de vidéo MPEG dans les routeurs en utilisant AQM. En effet, RED rejette sans distinction des paquets. Quelques paquets peuvent être plus importants que d'autres. Par exemple, la perte de quelques paquets produits par une application de diffusion vidéo MPEG peut rendre inutilisable l'application à la destination. Ceci a comme conséquence un gaspillage de la bande passante du routeur.

L'idée de XRED passe par la description d'un paquet suivant trois paramètres :

- *FlowID* pour identifier le flux,
- *ADUID* pour décrire des données spécifiques à l'application,
- le *Content Priority* pour les différentes priorités de contenu (les types d'images par exemple).

Une liste qui enregistre ces trois paramètres est sauvegardée sur le routeur. Quand des images MPEG sont fragmentées dans des paquets IP, XRED initialise chaque paquet avec *FlowID*, *ADUID* et *Content Priority*. Quand un paquet est rejeté, son *FlowID*, *ADUID* et *Content Priority* sont écrits dans la liste, et chaque paquet arrivant sur le routeur est vérifié en comparant ses paramètres aux paramètres stockés. Le paquet est rejeté si la variable *Content Priority* est inférieure à celle dans la liste.

Bien que les résultats de simulation prouvent que XRED réduit la perte de bande passante, XRED a l'inconvénient d'avoir besoin de champs supplémentaires dans l'en-tête IP. D'ailleurs, il doit enregistrer trois paramètres pour chaque flux d'application, provoquant des problèmes de mémoire en cas de mise à l'échelle. Enfin, il a besoin d'actions supplémentaires pour écrire et lire dans la liste et comparer les trois paramètres ce qui demande une quantité significative de puissance de calcul au routeur.

Modified RED

MRED[**?**] (Modified RED) s'intéresse au trafic en rafale. Avec un trafic en rafale, la taille de la file d'attente d'un routeur se remplit très rapidement. Lorsque la valeur moyenne de la file dépasse $MaxThreshold$, le routeur commence à éjecter/marquer des paquets pour faire baisser la taille de la file d'attente, ce qui diminuera celle-ci.

Mais comme la taille de la file est élevée, cela prend un temps considérable pour que la moyenne descende au-dessous de $MaxThreshold$. Cette intervention réduit le débit efficace et l'utilisation du lien du routeur.

En partant de cette constatation, MRED proposes une modification de RED pour faire face aux problèmes précédents. Il n'est pas nécessaire d'éjecter ou de marquer tous les paquets lorsque la taille de la file d'attente est inférieure à $MaxThreshold$ (ou même vide) et la moyenne est au-dessus de $MaxThreshold$. La principale modification de MRED est que la condition d'éjection/marquage des paquets :

1. $MaxTrehreshold < moyenne$
2. $MaxTrehreshold < taille\ de\ la\ file$

Le reste de l'algorithme reste identique à RED. Grâce à MRED, l'amélioration de l'utilisation du lien varie, d'après les auteurs, de 1,3% pour RED à 22,8% pour MRED.

1.4.3 Synthèse

Toutes ces variantes décrites RED ont permis d'améliorer les performances de la version originale. Toutefois, chaque variante a ses propres défauts et avantages. Le choix d'une variante de RED devrait alors être faite selon les exigences des applications et du réseau.

Les différentes variantes de RED calcule la probabilité de rejet différemment. Il existe des variantes de RED qui utilisent une fonction linéaire, d'autres qui utilisent une fonction par palier. Dans le tableau **??**, nous allons décrire brièvement le fonctionnement de certaines variantes de RED.

Nom	Description
RED	Politique basée uniquement sur la taille moyenne de sa file d'attente du routeur
Gentle RED	Version continue de la courbe de probabilité de rejet de RED
WRED	Basée sur le fonctionnement de RED mais avec des paramètres (probabilité et seuils) différents pour chaque flux
SRED	Sauvegarde d'une liste de flux récents
DSRED	Changement dynamique de la pente de la courbe de probabilité en fonction du niveau de congestion du routeur
FRED	Taille de chaque flux dans la file
XRED	Prise en compte du contenu pour les vidéos de type MPEG
ARED	Variation de P_{max}
Modified RED	Adaptation aux variations brusque de la taille de la file d'attente
Hybrid RED	Utilisation de la taille instantanée et moyenne de la file d'attente comme indicateur

Table 1.2 – Tableau comparatif et récapitulatif de RED et ses dérivées.

1.5 Explicit Congestion Notification

L'*Explicit Congestion Notification* ou ECN est une addition à la pile TCP/IP conçue en 1994 et standardisée en septembre 2001 par l'IETF sous la RFC 3168 [**?**]. C'est un mécanisme qui permet d'anticiper la congestion d'un routeur.

ECN est seulement employé lorsque les deux protagonistes (source et destination) signalent qu'ils sont d'accord pour employer cette extension. Avec cette méthode, un drapeau ECN est placé dans l'en-tête IP pour informer explicitement l'émetteur que le routeur est congestionné. ECN est une alternative à la notification indirecte (voir utilisation figure **??**) de congestion faite par l'effacement d'un paquet exécuté par RED et ses algorithmes dérivés comme WRED.

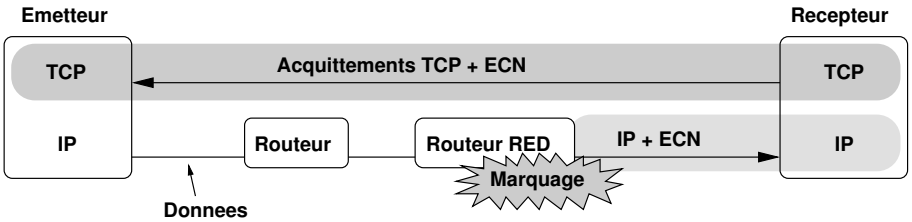


FIGURE 1.10 – La notification indirecte dans TCP grâce à RED et ECN

ECN est donc constitué de deux ajouts à des niveaux différents et constituant les deux phases du contrôle de congestion : détection et réaction.

Le premier niveau est la détection d'une congestion et la propagation de cette information. Quand un routeur reçoit un paquet marqué comme *ECN-Capable Transport* (ECT) et prévoit (en utilisant RED) la congestion, il placera un drapeau demandant à l'expéditeur de diminuer la taille de sa fenêtre de congestion. Le but de ce marquage est d'éviter de renvoyer un paquet que l'on aurait dû normalement éliminer. Le routeur informe la source d'une congestion sans qu'il n'y ait eu ni expiration du temps (*timeout*) ni perte d'un paquet.

L'ajout consiste en 2 bits représentant 4 drapeaux :

- 00 ou *Not ECN-Capable Transport* (Not-ECT) indiquant que le paquet ne supporte pas ECN,
- 01 et 10 traités de la même manière et signifiant ECT,
- 11 pour *Congestion Experienced* (CE) placé par un routeur sur un paquet au lieu de le rejeter pour informer d'une congestion.

Pour l'en-tête IP, ECN se place dans l'en-tête régulier et non dans les options IP pour éviter les mauvaises interprétations par certains routeurs. Ces drapeaux ont été ajoutés à la place des 6 et 7ème bits des champs IPv4 Type of Service et IPv6 Traffic Class. (voir figure **??**)

Code	Description
Not-ECT	Paquet incompatible ECN
ECT (0)	ECN Capable Transport (Nonce 0)
ECT (1)	ECN Capable Transport (Nonce 1)
CE	Congestion Experienced

TABLE 1.3 – Drapeaux ECN dans les en-têtes IP et TCP

La congestion est engendrée par la source du paquet qui a été marqué CE par un routeur sur le chemin. Pour réagir à cette congestion, il faut que le message soit transmis du routeur à la source du paquet. Or le routeur, lorsqu'il marque un paquet, n'informe à aucun moment la

source de celui-ci. Ce paquet marqué rejoint normalement sa destination. C'est la destination qui doit rediriger l'information vers la source. Pour cela, elle ajoute un drapeau dans l'en-tête TCP de ses accusés de réception pour indiquer à l'expéditeur de réduire la quantité d'information qu'il envoie. Ces drapeaux (voir figure **??**) sont :

- CE-Echo (ECE) placé sur chaque accusé sortant lorsque le récepteur reçoit un paquet CE afin de notifier la congestion et de demander une réaction (consistant en une réduction de la fenêtre de congestion grâce à l'algorithme de *Congestion Avoidance*),
- *Congestion Window Reduced* (CWR) envoyé par la source après la réception d'un segment ECE pour indiquer d'arrêter de placer le marquage ECE.

Lorsque la source TCP reçoit un paquet IP marqué *Congestion Experienced Echo*, le protocole de transport utilisé pour cette transmission doit déclencher les mêmes mécanismes de contrôle de congestion que s'il y avait eu détection d'une perte de paquets. ECN est basé sur le fait que la source (qui corrigera la congestion) utilise les algorithmes *Slow Start*, *Fast Retransmit* et *Fast Recovery* décrits dans la RFC 2581. De plus, pour éviter de bloquer totalement une transmission, il ne peut y avoir qu'une seule réduction de la fenêtre de congestion (*cwnd*) par fenêtre TCP (déterminant la quantité de données envoyée sur le réseau, par une source TCP, avant de recevoir un accusé).

Pour tous ces drapeaux, il existe une phase de négociation similaire à un *3 way handshake*, appelée *TCP ECN Negociation* qui peut être effectuée à tout moment de la transmission. Tant qu'elle n'est pas complètement effectuée, aucun paquet avec le marquage ECT ne doit être envoyé. Elle consiste à l'envoi d'un segment ECN-setup SYN dans lequel les drapeaux ECE et CWR sont activés indiquant que cet hôte est ECT. En réponse à ce paquet, un segment ECN-setup SYN-ACK est envoyé ne comportant que le marquage ECE, indiquant que cet hôte est également ECT.

Cas spéciaux

Les paquets d'acquittement seuls, ne comportant donc aucune donnée, ne sont pas marqués « ECN Capable »mais sont marqués Not-ECT car TCP ne possède aucun mécanisme capable d'éviter la congestion sur ces accusés (aucune fenêtre à réduire). Le marquage ECT est également annulé sur les paquets retransmis, car l'algorithme Fast retransmit corrige déjà la congestion.

Enfin, lors des sondages ou tests de la taille de la fenêtre (déclenchés par la machine source suite à une fenêtre du récepteur réduite à zéro), le marquage ECT et le marquage CWR sont également interdits pour la source et la destination, puisque leur perte n'est pas détectée en temps normal comme un signe de congestion et qu'il transporte peu de données.

Pour résumer, ECN utilise les drapeaux ECT et CE dans l'en-tête IP pour signaler la congestion entre les routeurs et les extrémités de la connexion, et les drapeaux ECE et CWR dans l'en-tête TCP pour signaler la congestion entre hôtes TCP.

1.5.1 Extension d'ECN

citezheng02 présente une version adaptative de ECN : Adaptative ECN (AECN). Cette version a pour objectif d'améliorer davantage les performances de ECN en ce qui concerne le débit et l'équité de flux en ajustant correctement les paramètres de ECN.

AECN ajoute à l'en-tête TCP un champ contenant des informations sur le RTT (*Round Trip Time*) d'un flux. Le champ est placé par l'expéditeur et lu par les routeurs. Les routeurs ont un ensemble d'intervalles de RTT, divisant les flux entrant en trois groupes : les fragiles,

les moyens et les robustes. Chaque groupe est associé à une sous-file d'attente correspondante. Chaque paquet est mis dans la sous-file d'attente appropriée, basée sur son RTT.

AECN se comporte de la manière suivant :

- Si $AvgLen \geqslant MaxThreshold$, on rejette le paquet entrant comme pour ECN ;
- Si $AvgLen < MaxThreshold$, on ajoute le paquet dans la file d'attente du routeur puis on détermine si le flux est robuste, moyen ou fragile. Ensuite on l'ajoute à la file d'attente du groupe ;
- Si a $MinThreshold < AvgLen < MaxThreshold$, on détermine la probabilité P_{max}. On marque suivant celle-ci le premier paquet non-marqué à l'avant de la file d'attente du groupe ;

La probabilité calculée est fonction de la probabilité classique pondérée d'un paramètre propre à chaque groupe.

Malheureusement, le RTT inscrit dans le paquet donne le temps pour faire l'aller-retour et est indépendant de la localisation du routeur, sur lequel il est lu, dans le trajet du paquet. On ne sait donc pas où l'on se situe par rapport à l'ensemble du parcours.

1.6 Conclusion

Les mécanismes de gestion de file d'attente décrits dans ce chapitre réalisent des actions nécessaires au maintien d'une certaine Qualité de Service dans le réseau. Dans un premier temps, l'identification d'un flux IP permet de déterminer le niveau de QoS associé. Ensuite, l'application des mécanismes de gestion de file d'attente permet de satisfaire les besoins de QoS que chacun des flux transmis réclament.

Beaucoup d'algorithmes de gestion de file d'attente de type AQM se sont inspirés de RED. Leur principal but est de provoquer des pertes anticipées pour contrôler les utilisateurs de TCP qui produisent le trafic traversant les routeurs. Tout comme RED, ces algorithmes ont un problème d'adaptation. Ils sont très difficiles à paramétrer à cause de la multitude des situations possibles : chaque combinaison de types de flux (simple transfert, transfert vidéo, données sécurisées, etc) demande une gestion différente. C'est à cause de ce problème que, bien que la politique RED soit implémentée dans tous les routeurs, elle n'est presque jamais activée.

Dans le chapitre qui suit, nous présenterons notre contribution qui est une nouvelle variante de RED appelée DDRED avec gestion du trajet d'un paquet. Nous comparerons les performances de DDRED avec certaines variantes existantes.

CHAPITRE 2

Proposition : Distance-Dependent RED

Sommaire

2.1 Introduction

Aujourd'hui de par la nature des échanges effectués sur le réseau Internet, diffusion de contenu multimédia (son et vidéo) et transfert de fichiers de grande taille sur des réseaux pair-à-pair, on observe une augmentation de la taille moyenne et donc de la durée moyenne des transferts.

Cette augmentation du trafic depuis ces dernières années n'a pas forcément été suivie d'une augmentation des équipements réseau intermédiaires. On constate donc une congestion des routeurs sur l'Internet. Suivant les mécanismes de gestion de files d'attente qu'ils mettent en œuvre, des paquets de données peuvent ou non être mis dans la file d'attente des routeurs.

La grande majorité des transmissions est basée sur le protocole de transport TCP. C'est un protocole fiable et robuste [**?**] qui par un système de contrôle de flux et d'acquittements des données émises va estimer l'état du réseau. La fiabilité est obtenue grâce à la retransmission des paquets perdus. Ce trafic additionnel ainsi produit conduit par conséquent à une réduction du débit utile et de plus grands temps de transmission. Une congestion crée déjà un ralentissement global du réseau donc il faut minimiser l'impact des retransmissions pour revenir le plus rapidement possible à un état général stable.

Le contrôle de congestion de TCP est crucial afin d'éviter la saturation des divers types d'équipements entre la source et la destination. Si un flux TCP éloigné fait entrer un routeur en congestion, il y aura des rejets de paquets. Quand un paquet est rejeté, il exige une retransmission, par conséquent il a inutilement employé les ressources réseau.

Nous verrons dans ce chapitre comment tenir compte de l'utilisation des ressources réseau et quels critères nous devons surveiller pour estimer au mieux cette consommation. Nous présenterons une intégration de notre traitement des pertes dans une politique de gestion de file d'attente qui a fait ses preuves.

2.2 Mécanisme

La file d'attente est le composant central de l'architecture des routeurs. Les divers algorithmes ou politiques de gestion de ces files, mis en place sur les routeurs, permettent de créer une

différenciation de services dans ces files. Dans un réseau avec une bande passante insuffisante, à un certain moment, la file d'attente déborde et les paquets suivants sont jetés. Des stratégies de gestions différentes limitent la dégradation du service réseau, en permettant à certains trafics d'être traités.

Nous proposons une autre politique de rejet de paquets dans ces files d'attente. Elle consiste à éliminer, lorsque qu'une perte de paquet est nécessaire, un paquet ayant consommé peu de ressources réseau. Cette faible consommation réseau peut se traduire par le nombre d'équipements traversés : un paquet avec une faible consommation est un paquet venant d'une source proche du routeur encombré. En éliminant les paquets de cette source, la retransmission occupera moins de routeurs.

Cette idée d'aide à l'élimination d'un paquet peut être vue comme un module décisionnel pour le choix d'un paquet « victime ». Notre politique est donc une extension d'une politique de gestion existante. Elle peut être intégrée dans n'importe quel algorithme de rejet de paquets dans un routeur, comme Drop Tail ou RED [**?**] (Random Early Detection), pour devenir, dans notre cas précis, DDRED (Distance-Dependent RED).

Le champ TTL (Time to Live) dans l'en-tête d'IP [**?**] contient déjà la notion de distance sur le réseau. Mais sa valeur initiale est fixée par l'expéditeur [**?**] (sur les systèmes Unix elle est généralement fixée à 64 voir fichier `/proc/sys/net/ipv4/ip_default_ttl`), donc elle est inconnue des routeurs. Par conséquent, pour mesurer la distance nous avons besoin de nouvelles informations pour notre politique de gestion de paquets dans les files d'attente des routeurs.

2.2.1 Principe

En un point du réseau, on définit la distance d_r d'un paquet comme étant le nombre d'équipements réseaux, tels les routeurs, entre ce point et la source d'émission du paquet. Si l'équipement réseau doit éliminer un paquet, il est préférable de rejeter un paquet avec une petite valeur d_r. En effet, un paquet proche de sa source sera rapidement retransmis et consommera peu de ressources réseau ; par contre un paquet plus éloigné de son émetteur utilisera un nombre plus important de liens et d'équipements (voir figure **??**).

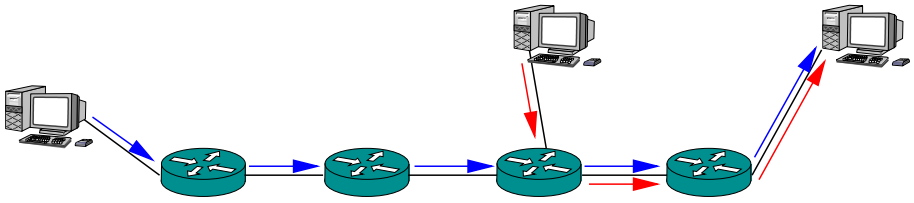


FIGURE 2.1 – Cas d'application avec chemin long et court

On peut voir ce principe comme le passage d'un point de non-retour : le chemin déjà parcouru contre celui restant à parcourir. Il est plus cher en coût d'équipements traversés de faire retransmettre un paquet que de le laisser continuer sa route jusqu'à sa destination. Nous utilisons cette distance pour développer de nouveaux types de politiques de rejet de paquets dans la gestion de file d'attente.

Le TTL ne pouvant pas être utilisé (comme précisé plus haut), nous créons nos propres champs dans l'en-tête IP, définis comme options IP par exemple. Nous présentons deux méthodes pour mettre en place ces champs.

La première méthode tient compte uniquement de la distance d_r. Ceci peut être fait simplement en utilisant un seul champ additionnel définissant le TTL initial (TTL_i). TTL_i est placé

dans chaque paquet avec la valeur initiale du champ du TTL de la source. En calculant la différence entre TTL_i et le TTL lu sur le routeur, on peut déterminer la distance $d_r = TTL_i - TTL$ couverte par le paquet. Cette distance correspond, si on la ramène au TTL initial, à un pourcentage de la distance à parcourir. De cette manière, on tient compte des différents chemins possibles. Des changements de parcours peuvent intervenir dans le cas de configurations multi-chemins dans les routeurs pour des raisons d'équilibrage de charge ou de panne d'un ou plusieurs équipements.

La seconde méthode est plus précise. Elle prend en compte le pourcentage de l'itinéraire que le paquet a déjà parcouru par rapport à la distance entre sa source et sa destination. Cette méthode est plus juste dans son principe, car elle elle tient compte de la distance en amont et en aval du point de congestion. Prenons l'exemple d'un paquet arrivant sur un routeur congestionné et devant normalement éliminer un paquet, le paquet entrant est proche de sa source et doit donc être éliminé. Malheureusement si sa destination est elle aussi proche, il n'aura pas la même valeur (en terme de coût d'équipements réseau traversés) qu'un paquet avec une source plus éloignée et une destination qui l'est encore plus.

Contrairement à la première, le seconde méthode implique deux champs de l'en-tête IP : TTL_i et le TTL final du paquet à sa réception : TTL_f. Ce dernier est fixé, soit à partir des paquets échangés durant la phase de synchronisation de TCP, soit en cours de transfert par la destination à partir du TTL du paquet précédemment reçu. En cas de routage multi-chemins, TTL_f est la distance du plus long chemin utilisé entre la source et la destination. On choisit le TTL_f du tout premier paquet de donnée, les paquets de synchronisation SYN ne sont pas utilisés.

2.3 Mise en œuvre

Nous avons étudié plusieurs manières de mettre en place cette idée en créant de toute pièce une nouvelle politique de gestion de file d'attente ou bien en combinant la proposition à une politique existante ayant fait ses preuves. En utilisant le champ Option de la trame IP, les routeurs implémentant notre politique et ceux ne la connaissant pas peuvent cohabiter sur un même réseau. En effet ce champ doit être défini, mais si un équipement ne sait pas le lire, il n'en tiendra pas compte. Un déploiement total n'est donc pas nécessaire pour faire fonctionner notre proposition. Mais pour une meilleure efficacité, les routeurs les plus souvent congestionnés (nœuds de raccordements, jonction inter-réseau, etc.) ont besoin de le connaître.

Nous présenterons ici deux cas de mise en œuvre de cette gestion dans une politique simple : Drop Tail, puis plus complexe, avec la politique RED.

2.3.1 La gestion de file d'attente DropTail

Drop Tail, ou Tail Drop, est un algorithme simple de gestion de file d'attente. Il s'agit d'une gestion passive de la file d'attente. Comme dans beaucoup de politique de gestion de file d'attente, il n'y a pas de différenciation des flux entrants, tous sont traités de la même manière par le routeur. Mais dans ce cas, quand la file d'attente est remplie à sa capacité maximum, les paquets nouvellement arrivés sont rejetés jusqu'à ce qu'une place soit libérée dans la file d'attente pour accepter le trafic entrant.

Pour mettre en place notre politique, au lieu de rejeter toujours le paquet entrant, nous cherchons dans la file celui dont la distance d_r ou son pourcentage de parcours est le plus petit, donc concernant le flux ayant la source la plus proche, et il est rejeté, tandis que l'autre est mis

en queue de la file d'attente. Si le paquet entrant est un paquet d'une source proche, c'est donc ce dernier qui est éliminé.

2.3.2 Une implémentation dans RED : DDRED

Comme nous l'avons vue dans **??**, RED est une politique de gestion plus complexe que Drop Tail. Il existe plusieurs façons de mettre en place notre module d'aide à la décision de rejet d'un paquet.

Méthode 1 : Pondération de la probabilité

Dans la politique de gestion des files d'attente RED, la décision de rejeter des paquets de données mettant le routeur en phase de congestion est basée sur un calcul de probabilité. Cette probabilité de rejet est uniquement définie en fonction de la taille moyenne de la file d'attente du routeur pendant un court instant. Tout en gardant cette évaluation statistique de la file d'attente, on peut modifier le principe de calcul de celle-ci en incluant le paramètre distance (voir figure **??**).

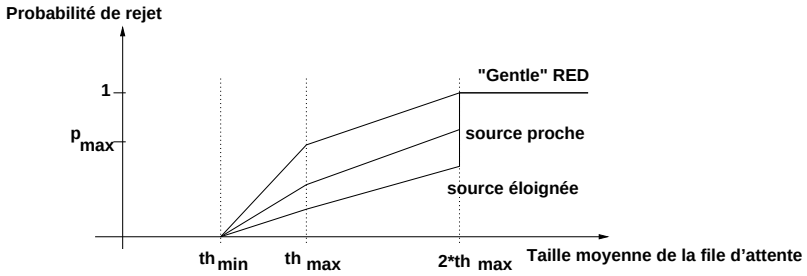


FIGURE 2.2 – Modification de la probabilité de rejet dans Gentle RED (identique pour RED)

En pondérant la probabilité avec un pourcentage de distance parcourue, on améliore cette gestion de file en tenant compte de tout le parcours. On obtient donc une nouvelle probabilité P_{TTL} :

$$P_{TTL} = P \times \left(\frac{TTL}{TTL_i} \right) \tag{2.1}$$

avec TTL le TTL lu sur le routeur, TTL_i celui de démarrage et P la probabilité calculée normalement lors de la décision de rejeter un paquet ou bien

$$P_{TTL} = P \times \left(\frac{TTL_f - TTL}{TTL_i - TTL_f} \right) \tag{2.2}$$

si l'on considère le chemin à réellement parcourir en tenant compte du TTL final TTL_f.

Distance moyenne Dans ce cas d'application, on pourra remarquer que l'on diminue globalement la probabilité de rejet d'un paquet. La nouvelle probabilité est toujours inférieure voire très inférieure à la probabilité de RED. Nous ne voulons pas modifier les propriétés de RED mais seulement le paquet rejeté. On risque donc de surcharger les files d'attente des routeurs et

donc de créer de nouveaux points de congestion. Cet inconvénient peut être supprimé grâce à un recentrage de la probabilité.

$$P_{TTL} = P \times \left(\frac{TTL_f - TTL - d_r m}{TTL_i - TTL_f} \right) \tag{2.3}$$

Pour qu'en moyenne le routeur se comporte de la même manière avec et sans gestion des distances, il faut que sur un réseau donné, le comportement classique de RED corresponde au comportement pour une distance moyenne $d_r m$ entre source et destination pour ce réseau-là. Mais on parle de millions de transferts par seconde donc la distance moyenne devra être actualisée fréquemment. Ceci est irréalisable du fait que pour connaître la distance moyenne entre source et destination, il faut pouvoir dénombrer tous les flux et partager cette information entre tous les équipements actifs du réseau. Le coût induit par la détermination de cette distance moyenne nous a conduit à ne pas considérer cette amélioration dans notre étude.

Méthode 2 : Changement de cible

Nous avons vu que dans RED, le rejet de paquet était soumis à une probabilité que nous avons essayé de changer. Au lieu de changer cette formule de calcul, nous avons décidé de changer seulement le paquet auquel cette probabilité s'applique. Quand la probabilité exige de rejeter le paquet entrant, le routeur recherche dans sa file d'attente le paquet avec la plus petite distance d_r et le traite comme s'il s'agissait du paquet entrant (voir figure **??**).

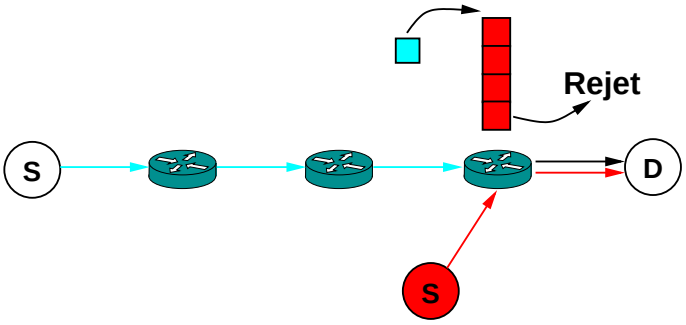


FIGURE 2.3 – Changement de cible

Si la probabilité indique que le paquet doit être rejeté de la file, on rejette la nouvelle cible désignée à la place du paquet entrant. Pour des flux ECN (Explicit Congestion Notification), le même mécanisme s'applique sauf qu'un paquet rejeté devient un paquet marqué.

Cette gestion pose un problème d'équité entre les flux : si deux flux ou plusieurs flux sont en concurrence au niveau d'un même routeur congestionné, le changement de cible affectera toujours le même flux. De même, si on considère le marquage ECN d'un paquet à la place de son rejet, un paquet peut même être marqué plusieurs fois. Comme le drapeau ECN sur un paquet est une information importante dans la régulation du trafic sur le réseau, il vaut mieux dans ce cas ne pas surcharger un seul paquet.

Il faut donc introduire un choix équitable de la cible tout en gardant en mémoire la règle : « Je rejette un des paquets les plus proches ». Pour cela on introduit une forme d'historique des flux déjà concernés par l'élimination d'un de leurs paquets afin de ne pas choisir à nouveau un flux diminué. Le marquage ECN pourra se faire non pas que sur les flux proches mais sur tous afin de réguler au mieux les flux traversant notre routeur congestionné.

Afin de choisir le paquet dans la file d'attente, deux méthodes s'offrent à nous : la recherche descendante ou ascendante. L'information véhiculée par le drapeau ECN étant renvoyée à l'émetteur par les accusés de réception TCP, plus le paquet choisi sera en tête de la file d'attente, plus l'information (perte d'un paquet, marquage ECN) sera rapidement connue de l'émetteur du flux subissant la congestion. Le chois d'un paquet à rejeter est différent d'un paquet à marquer, il est plus logique de rejet un paquet en queue qu'un paquet ayant déjà consommé des places dans la file. Mais cette consommation de la file est minime par rapport à la consommation des ressources réseau et marquage ou rejet peuvent être traité de la même manière sans perte de performance. De cette constatation, il semble donc logique de faire une recherche de la cible en démarrant de la tête de la file d'attente afin de faire connaitre la situation de congestion au plus vite de l'émetteur.

Historique d'un paquet

Sur le même principe que l'historique des flux déjà éliminés, on peut introduire un historique plus précis sur les paquets déjà éliminés. Dans certains cas, il pourrait arriver, à cause de la rapidité de retransmission d'un paquet venant d'une source proche, que celui-ci soit à nouveau éliminé de la file d'attente.

Pour éviter cet acharnement sur un paquet, on peut considérer en plus de son origine, son temps passé dans le réseau. Un paquet constamment retransmis bloque une retransmission : au lieu de couper une transmission en éliminant un même paquet à chaque fois, pourquoi ne pas le laisser continuer après quelques essais et éliminer ensuite le suivant dans la transmission pour le flux concerné.

Mais cet historique peut être élargi à l'ensemble des routeurs du réseau. Si une instance d'un paquet (le paquet contient les mêmes données mais c'est la énième retransmission de celui-ci) a déjà été éliminé sur un routeur lors de sa transmission précédente, cela peut diminuer ses chances d'être à nouveau rejeté de la file d'un autre routeur sur le réseau. Cela suppose bien sûr qu'il existe une communication entre routeur ou bien un système centralisé.

2.4 Étude de cas

Avoir plusieurs estimations des différentes méthodes, le principe du changement de cible a été choisi. Pour valider le fonctionnement de notre algorithme de rejet, nous avons utilisé un simulateur de réseau : Network Simulator 2 (NS2). Un simulateur de réseau nous permet, contrairement à une expérimentation réelle, de pouvoir reproduire rapidement l'expérience sans en changer les comportements de base. Il devient alors possible de mettre en vis-à-vis des résultats avec et sans notre amélioration sur des transferts identiques. De plus on peut aussi créer facilement des jeux de simulations avec des différences minimes : comparaison de deux politiques dans un même scénario et confrontation de différents scénarios.

Nous avons créé un patch NS2 v2.29 qui modifie la politique de rejet dans l'algorithme RED comme présenté précédemment. Pour plus de fiabilité, chaque étude de cas (appelée Sr_n pour Simulation run n) a été simulée avec différentes graines du générateur de nombres aléatoires, donnant différents scénarios. Chaque scénario est simulé deux fois avec les mêmes conditions initiales (taille de transfert et heure de départ de transfert) : dans le premier, tous les routeurs mettent en application l'algorithme RED et dans le second, l'algorithme DDRED.

Pour comparer les performances des politiques RED et DDRED, les critères suivants ont été utilisés :

– *le nombre de paquets rejetés* des files d'attente des routeurs, qui fournissent des informations sur les congestions et donc sur l'utilisation des ressources réseau,
– *les temps de transfert*, parce qu'ils sont visibles par tous les utilisateurs,
– *le nombre d'emplacements de la file d'attente consommés.*

Avoir des pertes sur un réseau est signe de congestion, dans notre cas le nombre de pertes a une signification différente. Le plus important ce n'est pas le fait qu'un paquet soit perdu sur son trajet, mais qu'il ait consommé des ressources (des processeurs et de la bande passante sur les routeurs). Les pertes ne sont donc pas à comptabiliser en tant que telles, mais c'est leurs parcours qui est important : par exemple un paquet perdu au dixième routeur est plus coûteux en terme de ressources que trois paquets perdus au deuxième routeur. Le but premier n'est pas de minimiser les pertes mais de déterminer à un instant et sur un routeur, quel est le paquet que je peux me permettre de perdre.

2.4.1 Topologie simple de réseau

Afin de valider notre politique, nous allons la tester dans un premier temps sur une organisation très simple de topologie de réseau : la chaîne. Tous nos ordinateurs sont reliés à une même ligne de transmission constituée d'une chaîne de routeurs identiques

La figure **??** présente la topologie de réseau en bus que nous utiliserons. Cette topologie comporte 8 routeurs et 8 stations de travail. Tous les liens entre les stations de travail et les routeurs et entre les routeurs ont une bande passante de 10 Mbits/s. 100 transferts FTP, basés sur TCP New Reno, sont créés dans une période de 60 secondes et la simulation est arrêtée quand tous les transferts sont accomplis. La source et la destination de chacune de ces derniers flux sont aléatoirement choisies parmi les postes de travail. La taille des données transférées est aussi aléatoirement choisie entre 100 Ko et 6 Mo. Grâce au simulateur, tous les paramètres réseau sont reproductibles.

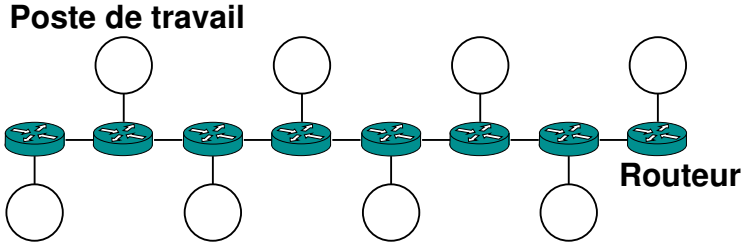


FIGURE 2.4 – Exemple de topologie réseau en chaîne.

Résultats

La figure **??** présente le nombre total de paquets perdus sur les routeurs durant la simulation. Le nombre moyen de paquets perdus pour les 100 flux est de $\overline{L_{RED}} = 2724$ et $\overline{L_{DDRED}} = 3160$.

Cette courbe montre que l'on a plus de pertes avec notre politique DDRED.

Afin de mesurer les ressources consommées, pour chaque distance couverte d_r on totalise le nombre de paquets perdus L_d par l'ensemble des flux circulant pendant la simulation. On pondère ensuite ce nombre de pertes par leur distance respectivement couverte :

$$S = \sum_{d_r=1}^{d_{rmax}} (d_r \times L_d) \tag{2.4}$$

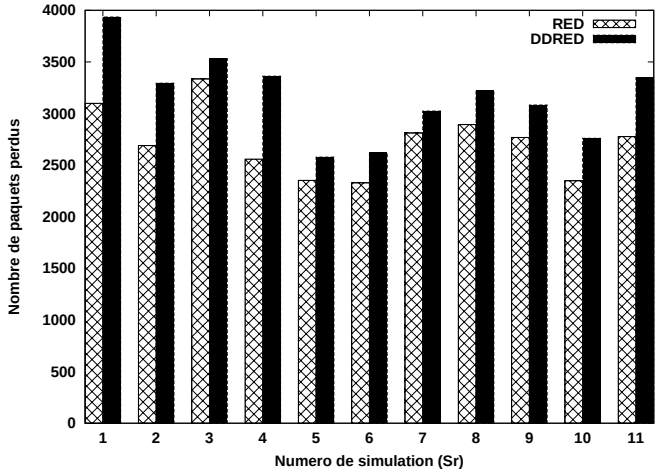


FIGURE 2.5 – Topologie chaîne, nombre de paquets perdus.

La figure **??** montre un exemple des répartitions $L_d(d)$ pour un des scénarios de simulation. Comme nous l'avons voulu, les paquets perdus par DDRED sont rapprochés de la source de transfert. La figure **??** présente cette somme pondérée pour chaque exécution Sr_n de la simulation. Nous économisons ainsi de l'espace ou slots dans les files d'attente des routeurs (un slot est l'espace pris par un paquet de taille moyenne dans la file d'attente d'un routeur). Le nombre de slots consommés est réduit en moyenne de $\overline{S_{RED}} = 5418$ à $\overline{S_{DDRED}} = 3259$.

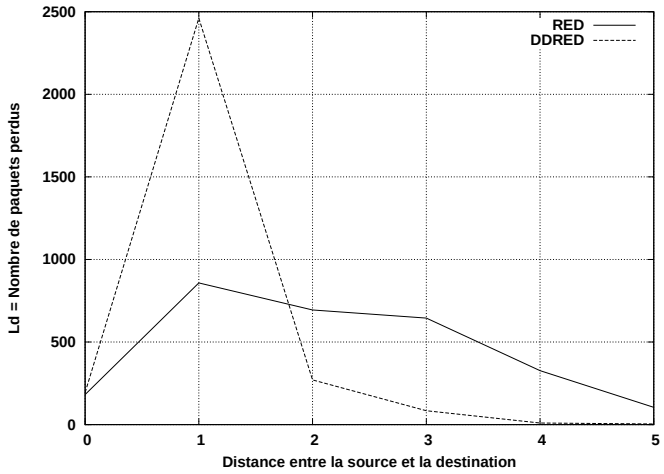


FIGURE 2.6 – Topologie chaîne, répartition des perte pour une simulation(Sr_6).

La figure **??** présente la somme des temps de transmission de tous les transferts pour chaque simulation Sr_j. Cette somme des temps de transmission (différence entre le temps du dernier paquet reçu T_{end}et celui du premier envoyé T_{begin}) de nos cent flux est donnée par :

$$T_{Sr_j} = \sum_{i=1}^{100} (t_{end_i} - t_{begin_i}) \; avec \; 0 < j < 12 \tag{2.5}$$

Généralement, nos courbes de résultats montrent une baisse de la somme totale des temps de

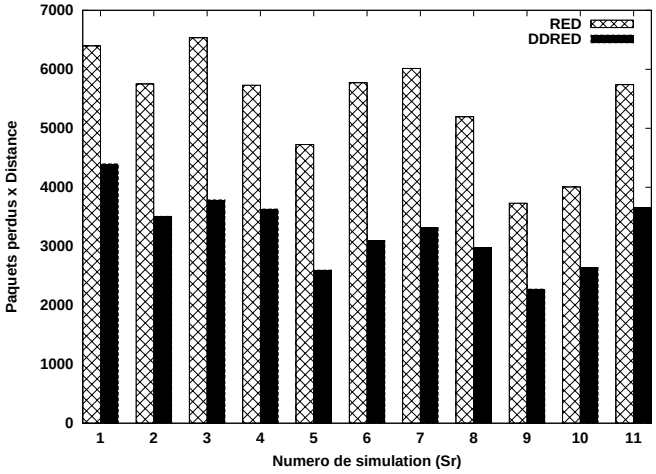


FIGURE 2.7 – Topologie chaîne, pertes pondérées par leur distance couverte.

transferts. On obtient en moyenne un bénéfice pour DDRED de 2.5% en temps total ($\overline{T_{RED}}$ = 3196 s et $\overline{T_{DDRED}}$ = 3125 s). On peut penser que les slots sauvés dans la file d'attente de DDRED sur le routeur sont libres pour d'autres flux et c'est pour cela que les transferts de bout-en-bout s'effectuent plus rapidement.

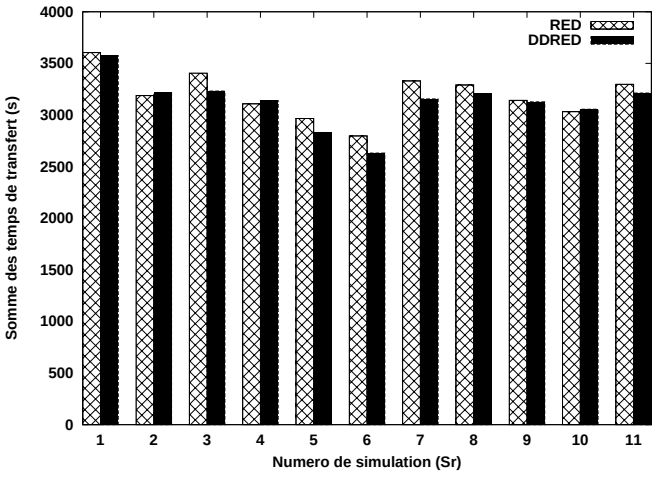


FIGURE 2.8 – Topologie chaîne, somme de tous les temps de transmission.

Le réseau bus est un réseau très simple nous permettant de bien mettre en évidence le comportement de notre politique en cas de congestion sur le réseau. La chaîne de routeurs entraîne une propagation de la congestion le long de celle-ci. C'est un cas d'étude peu réaliste mais qui nous permet de valider l'algorithme de rejet en cas de congestion multiple.

2.4.2 Topologie du réseau Fleur

Afin d'avoir un scénario plus réaliste, nous avons défini une nouvelle forme de topologie : un réseau en « fleur »(voir la figure **??**). Après une petite étude du cœur du réseau xDSL d'un

fournisseur d'accès[1], la plupart de ces réseaux sont construits autour d'un noyau central où plusieurs boucles sont reliées. Ces boucles se composent d'un nombre restreint de routeurs. Le but premier des boucles dans ce réseau est d'avoir, par le biais de chemins multiples, une tolérance aux pannes.

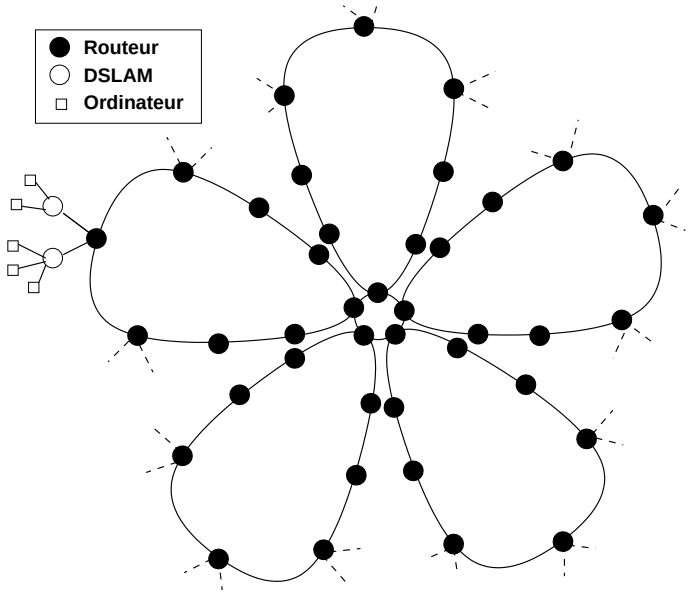


FIGURE 2.9 – Topologie « fleur »pour le cœur du réseau.

Nous considérons une topologie en fleur avec cinq boucles et huit routeurs composent chacune d'elles. Sur chaque routeur est greffé un arbre représentant les équipements d'un fournisseur d'accès xDSL à Internet. Dans cette simulation 500 connexions (supposées indépendantes) sont initialisées dans la même première minute et la simulation s'arrête à la fin de la dernière transmission. La source et la destination sont choisies au hasard dans tout le réseau.

Résultats

La figure **??** présente le nombre de paquets perdus par chaque politique. On peut remarquer que DDRED perd moins de paquets que RED, approximativement 5.8 % ($\overline{L_{RED}} = 33659$ à $\overline{L_{DDRED}} = 31738$. On observe déjà une amélioration au niveau de la gestion des files d'attente.

Un paquet rejeté d'une file d'attente a consommé inutilement des slots dans les files d'attente qu'il a traversées. Dans la figure **??**, on peut voir le nombre d'emplacements utilisés avec ces politiques de file d'attente. Il est en moyenne de $\overline{S_{RED}} = 150218$ et $\overline{S_{DDRED}} = 112023$ paquets. On obtient en moyenne une économie d'environ 25% des ressources réseau. Les emplacements économisés, disponible pour d'autres flux, peuvent augmenter jusqu'à 35% comparés à RED. Cette disponibilité n'est pas mise en valeur dans ces résultats, par une augmentation du nombre de flux en circulation sur le réseau, car les flux sont définis de manière statique. Il n'y a pas de nouveaux flux créés, seuls ceux prévus par la simulation le sont.

La somme des temps de transfert avec DDRED est en moyenne 5,5% plus petite (voir figure **??**, où $\overline{T_{RED}} = 44272$ s et $\overline{T_{DDRED}} = 41866$ s). Nous ne nous intéressons pas à chaque flux car tous les transferts ne sont pas plus rapides, mais en moyenne on gagne du temps. Suivant les indications de la figure **??**, 237 flux sont plus rapides et 214 flux sont plus lents comparés

1. `http://support.free.fr/reseau/province.png`

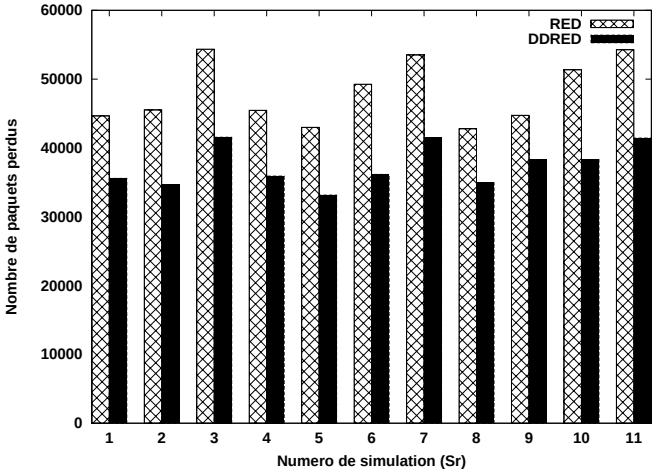


FIGURE 2.10 – Nombre de paquets perdus sur l'ensemble des routeurs.

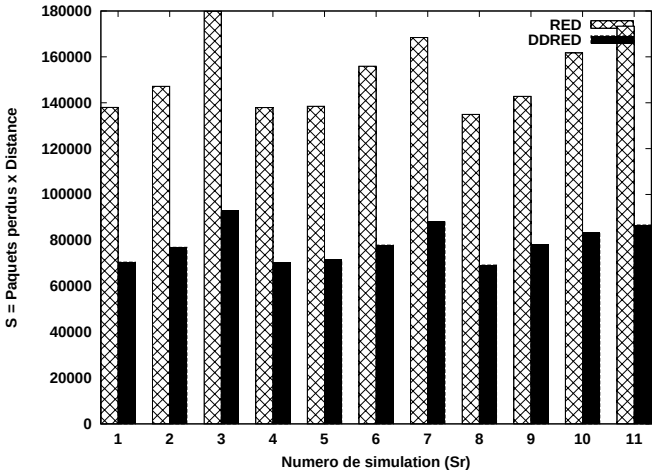


FIGURE 2.11 – Pondération des paquets perdus par leur distances couvertes.

à RED. La surface des flux gagnants est de 11362, celle des flux perdants est de 8142, ainsi DDRED donne une bonne distribution du gain et des pertes sur chaque flux. Les flux courts étant défavorisés par rapport aux longs, si un tel flux était en plus d'une taille importante, on obtient forcément une augmentation de la durée de transfert.

2.5 Équité et famine des flux

DDRED est une politique RED modifiée. Il est donc nécessaire de connaître les implications de cette nouvelle politique dans un réseau. Dans [**?**], RED est présenté comme offrant trois avantages par rapport à DropTail : éviter le comportement de famine d'un flux, en réduisant le nombre de paquets perdus dans les routeurs et en fournissant de moindre retards dans les services interactifs.

Cette section présente une étude sur l'équité de la concurrence entre des flux TCP. Contrairement à RED, DDRED réalise une corrélation des pertes, car elle choisit toujours de supprimer les paquets de la plus courte connexion. Toutefois, ce flux est le plus rapide en terme de re-

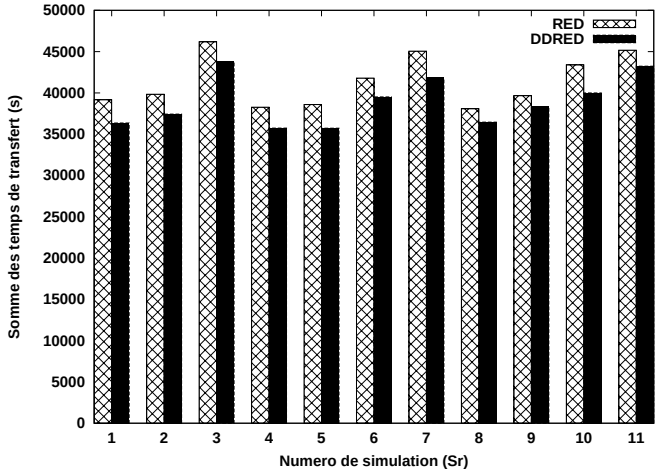


FIGURE 2.12 – Somme des temps de transmission de tous les flux.

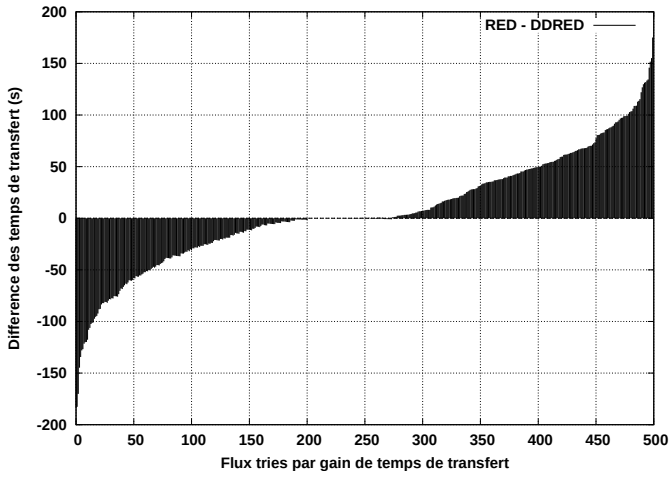


FIGURE 2.13 – Durée des transmissions de chaque flux dans un scénario (Sr_3).

couvrement des pertes de paquets, ce qui réduit ce désavantage. En outre, contrairement à une file DropTail, les rafales de pertes sont évitées, en raison de l'utilisation des probabilités proposées par RED. DDRED peut être considéré comme une file d'attente comprise entre RED et DropTail.

2.5.1 Exemple de réseau

Au cours de la congestion, un flux long a toujours la priorité sur un court, qui peut être sévèrement sanctionné dans certaines circonstances. Supposons le réseau de test suivant, le routeur R2 utilise DDRED :

```
PCs1 --- R1 --- R2 --- PCd
                 |
                PCs2
```

Chaque lien a une latence de 1 ms et une bande passante de 10Mb/s. Deux connexions TCP, de PCs1 vers PCD, et de PCs2 vers PCD, démarrent au temps $t = 0$. L'intervalle de temps pris en compte est entre la seconde 0 à la seconde 10 (temps d'arrivée du paquet), et la taille des données transmise est infinie.

2.5.2 Discussion

Compte tenu de notre algorithme basé sur le TTL, PCs1 a toujours la priorité sur PCs2 au niveau de R2. Les accusés de réception sont traités exactement comme les paquets de données.

Étape 1 : deux connexions existent dans la file d'attente. Lorsque R2 devient encombré, PCs2 diminue son débit d'envoi, mais PCs1 continue d'augmenter régulièrement sa cadence. PCs2 continue de baisser sa cadence, alors que PCs1 augmente la sienne (comportement de TCP en l'absence de paquets perdus). Cela se produit jusqu'à ce que le débit de PCs1 dépasse la vitesse de traitement des paquets du routeur R2. Le résultat final est que la bande passante disponible pour PCs2 diminue régulièrement.

Étape 2 : il n'y a que le flux de PCs1 dans la file d'attente. Maintenant, un paquet de PCs1 est rejeté, cependant, les accusés continuent d'être correctement reçus. PCs1 réduit donc son débit d'envoi, permettant ainsi à PCs2 d'accroître sa bande passante, mais PCs2 est plus rapide : comme la bande passante disponible pour PCs2 augmente, il accroit sa transmission de données. Comme il est plus proche que l'autre, son accélération est plus élevée que celle de PCs1. La conséquence de ceci est que PCs2 reprend sa transmission plus vite que PCs1, mais PCs1 redémarre d'un débit diminué de moitié alors que PCs2 recommence de 0. Ensuite on recommence à l'étape 1.

Résultat de ces étapes : en raison de son RTT supérieur, PCs1 ne peut pas toujours obtenir une part majoritaire de la bande passante.

Lorsque la différence de RTT est faible, PCs1 est beaucoup moins pénalisé et, lorsqu'il l'est, recouvre son débit initial presque aussi vite que PCs2. Cette situation est clairement favorable à PCs1, mais cela ne conduit pas à un arrêt de PCs2. Il n'y a donc pas famine d'un flux. Les simulations ci-dessous montrent que PCs2 peut encore gagner 15 % de bande passante dans un tel cas défavorable.

Lorsque la différence de RTT est élevée, PCs1 a besoin de plus de temps pour se remettre de l'erreur survenue dans l'étape 2, suffisamment pour permettre à PCs2 d'envoyer de nombreux paquets dans une courte période de temps (voire même plus que PCs1). Cette situation est aggravée par le fait que les paquets sont marqués dans un ordre descendant de la file d'attente, ainsi PCs2 n'est pas informé immédiatement de la congestion et a le temps d'envoyer encore plus de paquets. Cette situation, où le débit de PCs1 est supérieur à PCs2 (étape 1) et inversement (étape 2), est illustré par la figure **??**, obtenues par la simulation présentée ci-dessous.

2.5.3 Simulations

Plusieurs simulations ont été réalisées avec une latence du lien PCs1-R1 allant de 1ms à 100ms. Le tableau **??** présente certains résultats sous forme d'un ratio de débits $PCs1/PCs2$.

Les RTT pour la connexion longue sont égaux à la somme des latences des liens traversés multiplié par deux. Dans l'en-tête TCP, le champ *rwnd*, fenêtre de réception (*received window*), permet à un nœud TCP d'informer les autres de l'espace disponible encore dans son tampon de réception [**?**]. Par conséquent, au cours d'un RTT, un émetteur ne peut envoyer plus de *rwnd* octets. Le débit d'une connexion est donc limitée par *rwnd* : $max = rwnd/RTT$. Par exemple,

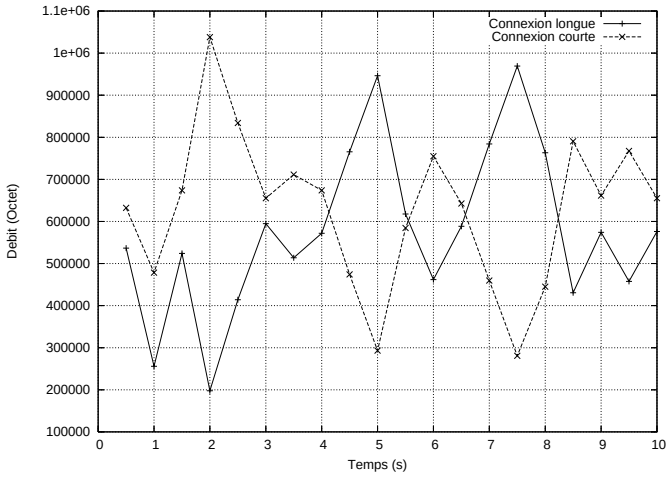


FIGURE 2.14 – Les deux flux ont alternativement le plus haut débit.

Latence PCs1-R1 (ms)	1	5	20	50	100
RTT de la connexion longue (ms)	6	14	44	104	204
rwnd=64Ko/s					
PCs1 max rwnd bw (Ko/s)	10666	4571	1454	615	313
Ratio Débit PCs1/PCs2 RED (Ko/s)	509/727	428/804	230/998	136/1078	65/1144
Ratio Débit PCs1/PCs2 DDRED (Ko/s)	991/214	847/340	616/590	363/798	280/873
rwnd=256Ko/s					
PCs1 max rwnd bw (Ko/s)	42666	18285	5818	2461	1254
Ratio Débit PCs1/PCs2 RED (Ko/s)	599/641	338/901	235/998	171/1025	123/1079
Ratio Débit PCs1/PCs2 DDRED (Ko/s)	1024/198	851/339	577/625	449/686	128/1076

TABLE 2.1 – Comparaison entre les résultats des simulations RED et DDRED.

pour la première colonne de données, PCs1 $max\ rwnd$ est $256Ko/6ms = 42666Ko/s$ (avec $rwnd = 256Ko$ pour toutes les simulations).

En réalité, un flux est limité par la bande passante disponible sur un lien, et non par *rwnd*. La raison en est que l'option du protocole TCP « window scaling » [**?**] peut être utilisé, ce qui permet d'adapter *rwnd* au moment de l'initialisation de la connexion.

Toutefois, dans NS2, *rwnd* est fixe [2] et par défaut égal à 20Ko. Les essais avec $rwnd = 64Ko$ ne sont pas suffisants, parce que le débit est limité par *rwnd*, comme le montre par exemple la colonne avec une latence de 100ms. Par conséquent, un autre essai a été fait avec un nouvelle valeur de *rwnd* fixée à 256Ko, comme le montre le tableau **??** ci-dessus, qui ne limite pas non plus le débit

Pour comparer l'équité de DDRED à RED, le ratio de la bande passante entre les deux flux est illustré par la figure **??**. Plus la courbe est proche de 1, plus équitable est l'algorithme. (Le rapport entre les bandes passantes de deux flux TCP n'est pas exactement de 1, mais dépend de leur RTT, voir par exemple [**?**]. Les simulations montrent que, pour une différence de temps de latence inférieur à 10 ms (différence de RTT inférieur à 20ms), RED a un ratio plus proche de 1, c'est pourquoi RED est juste en terme d'équité. Toutefois, entre 10 ms et 100 ms, DDRED a un ratio plus proche de 1, alors DDRED est aussi équitable. Lorsque le temps de latence est de

2. Limitations de NS2 : `http://www.isi.edu/nsnam/ns/ns-limitations.html`

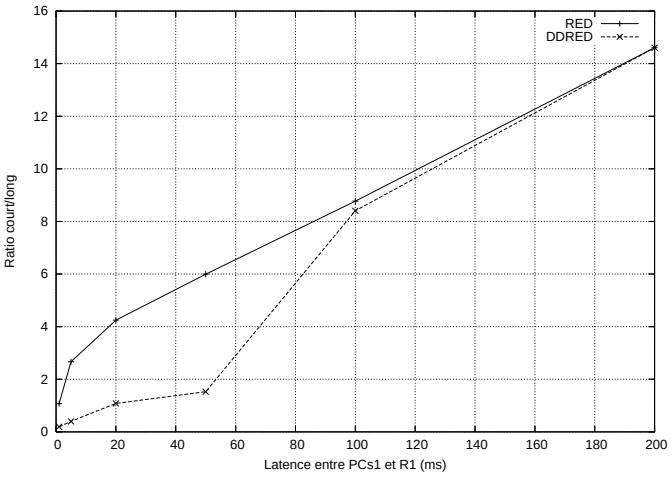


FIGURE 2.15 – Ratio de bande passante entre le flux court et le flux long.

plus de 50ms, l'équité des deux stratégies est rompue, même si DDRED a de meilleures valeurs que RED.

L'une des conclusions est que DDRED n'est pas adapté à des réseaux où des latences sont de petite taille, ou « réseaux fermés », mais fonctionne mieux que RED en ce qui concerne les « réseaux ouverts », où les latences sont plus variées. Tous les systèmes autonomes dans le contexte actuel de l'Internet peuvent alors être considérés comme des réseaux ouverts.

Enfin, il est intéressant de noter que DDRED est non seulement plus juste (c'est-à-dire le rapport entre les flux est plus proche de 1) que RED, mais aussi a un débit plus élevé.

TCP est un protocole avec un contrôle de congestion basé sur une fenêtre (de congestion). Il envoie des paquets après réception d'accusés de réception. Le taux d'envoi est limité supérieurement par les paquets perdus et inférieurement par le taux de réception de ses accusés. Comme les flux adaptent leur débit à la bande passante disponible sur un lien, TCP est considéré comme un protocole équitable.

Lorsque des flux concurrents ont des RTT différents, leur taux de réception d'accusés est différents, et le partage de la bande passante entre eux est proportionnelle à leur RTT. Lorsque deux flux symétriques (même RTT) coexistent sur un même lien, la bande passante du lien est partagée de manière sensiblement égale entre eux.

L'algorithme présenté, DDRED, change aussi les proportions du partage de la bande passante d'un lien. Au cours de la congestion, il favorise les flux lointains en rejetant les paquets des flux proches. Néanmoins, nous estimons que cette politique ne change pas l'équité des protocoles :

- Dans le contexte actuel l'algorithme de TCP, la bande passante prise par les flux entre deux routeurs n'est pas identique, mais dépend de leur RTT.
- Si tous les routeurs dans un réseau DDRED utilisent cette politique, alors les paquets de tous les flux passeraient d'un état défavorable, quand ils sont au routeur près de la source, à l'état favorable, au routeur proche de la destination, au cours de leur transmission. On peut parler de *qualité de service progressive*.

En outre, DDRED augmente le taux d'utilisation des ressources et le trafic utile global.

2.6 Conclusion

Pendant la congestion, les routeurs rejettent des paquets, indépendamment de la politique de file d'attente. Quand un paquet est rejeté, toutes les ressources réseau qu'il a consommées sont gaspillées. Nous avons présenté un nouvel algorithme de rejet de paquets sur les routeurs qui tient compte du chemin déjà parcouru par un paquet. Des paquets loin de leurs sources sont favorisés comparés aux paquets près de leurs sources. Cet algorithme peut s'appliquer au sein de diverses politiques de gestion des files d'attente pendant la congestion.

Les simulations avec NS2 de cet algorithme dans RED montrent qu'avec cette nouvelle politique de gestion de file d'attente, le débit d'un flux favorisé augmente relativement aux autres flux. De même, pendant que les paquets appartenant aux flux longs sont favorisés, nous économisons globalement des ressources réseau. Cela nous permet d'avoir une bande passante globale plus importante sans pour autant sacrifier la propriété d'équité TCP des flux.

Il s'avère que le cheminement des paquets joue un rôle significatif dans les problèmes de contrôle de congestion et de priorité de flux. Décider quel chemin prendra chaque paquet, router, est un aspect complexe du problème dont l'efficacité détermine directement les performances du réseau.

En se basant sur les réseaux simples et en ajoutant quelques modifications particulières dans la politique de la gestion de file d'attente aux niveaux des routeurs, nous montrons, en employant des simulations sur NS2, qu'il est possible d'améliorer significativement les performances des réseaux courants. Nous favorisons les rejets de paquets sur le transfert court car celui-ci retrouve plus rapidement un débit stable et de ce fait nous augmentons la disponibilité du réseau sur le chemin plus long. Avec notre solution, nous épargnons globalement les ressources réseau. Ceci nous permet d'avoir une bande passante globale plus élevée et par conséquent un plus haut débit par flux en moyenne.

Quelques inconvénients subsistent toujours à cette politique de traitement des files d'attente. Le principal problème vient du temps CPU nécessaire au traitement des files d'attente sur un routeur. En effet, chaque routeur est composé de plusieurs files qui peuvent contenir chacune des dizaines de paquets. Le temps supplémentaire pour traiter ces paquets peut induire une nouvelle forme de congestion : un routeur qui arriverait à traiter un trafic normal pourrait se retrouver surchargé par ce même trafic à cause de notre gestion des files d'attente. Toutefois la puissance CPU des routeurs a augmenté énormément depuis plusieurs années.

Deuxième partie

Contrôle de congestion sans fil

CHAPITRE 3

Etat de l'art : Wi-Fi, Transport en réseaux sans fil et différenciation des pertes

Sommaire

3.1 Introduction

Internet, et tous les réseaux informatiques en général, autrefois exclusivement des réseaux câblés, sont complétés aujourd'hui par les réseaux sans fil (WLANs) et les réseaux de téléphonie mobile. Beaucoup de commerces, universités, lieux publics (gare, etc.) proposent maintenant un accès à Internet sans fil. Simultanément, se développent de manière croissante des applications multimédia en temps réel comme la diffusion audio et/ou vidéo, la téléphonie sur IP, la visiophonie, les jeux en réseau, etc. Pour permettre de telles applications multimédia sur un réseau mixte, des mécanismes efficaces de contrôle de congestion doivent être mis en place sur ces supports.

La majorité des protocoles de transport sur Internet est basée sur le premier d'entre eux : TCP [**?**]. Ils se comportent généralement de la même manière en considérant qu'une perte de paquet est due à une congestion du réseau. Or cette association perte-congestion, si elle reste valable dans les réseaux classiques avec un câble en cuivre ou une fibre optique, ne l'est plus en ce qui concerne les transmissions radio utilisées dans les réseaux sans fil de type Wi-Fi.

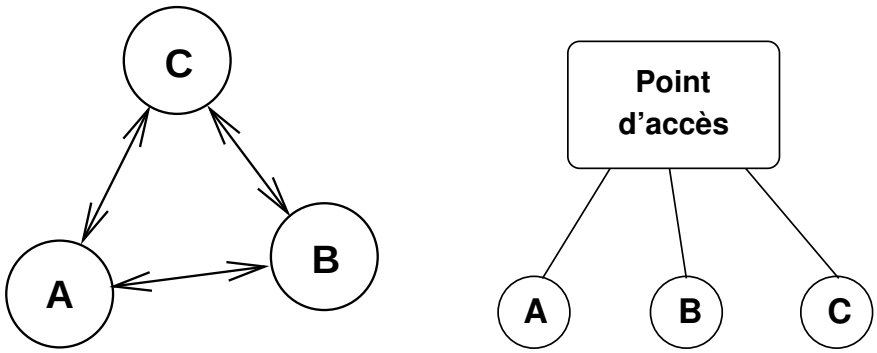
Ce chapitre présente dans un premier temps le fonctionnement et les propriétés des réseaux sans fil. Ensuite nous verrons un ensemble d'améliorations de protocole de transport dans le cas de réseaux peu fiables. Enfin Nous présenterons différents travaux sur le problème de la différenciation des pertes dues à la congestion des routeurs des pertes causées par les liaisons radio.

3.2 Les transmissions Wi-Fi

3.2.1 Équipements

Le Wi-Fi est une technologie de réseau informatique sans fil mise en place pour fonctionner en réseau d'extrémité : il est souvent le dernier maillon faisant la connexion entre le client et le œur du réseau. Il est devenu un moyen d'accès à haut débit à Internet. Il est basé sur la norme

IEEE 802.11. C'est un réseau qui peut fonctionner suivant plusieurs modes, les deux principaux utilisés dans un cadre normal étant le mode *ad-hoc* et le mode infrastructure (voir figure **??**).



(a) Ad-hoc (b) Infrastructure

FIGURE 3.1 – Deux modes de fonctionnement du Wi-Fi.

Les réseaux informatiques *ad-hoc* **??**, en latin : « qui va vers ce vers quoi il doit aller », sont des réseaux sans fil capables de s'organiser sans infrastructure définie préalablement. Chaque équipement mobile peut jouer différents rôles à différents instants.

Quant au mode infrastructure **??**, il est composé d'une ou plusieurs stations de base (BS pour Base Station) appelées aussi point d'accès AP généralement interconnectées entre elles par un réseau de type filaire. La zone couverte par un point d'accès est appelée BSS (Basic Service Set). Les BSSs de plusieurs stations de base peuvent être regroupées en un ESS (Extended Service Set). La mobilité entre les ESSs est supportée en Wi-Fi grâce au mécanisme de *handover* qui prend entre 60 et 400ms pour se réaliser.

3.2.2 Évolution de la liaison radio

Au début les connexions sans fil se faisaient principalement entre stations ou entre stations et point d'accès. Depuis quelques années maintenant, on trouve une nouvelle utilisation des liaisons radio : il ne s'agit plus de liaisons isolées mais d'un réseau complet relié par des liaisons sans fil.

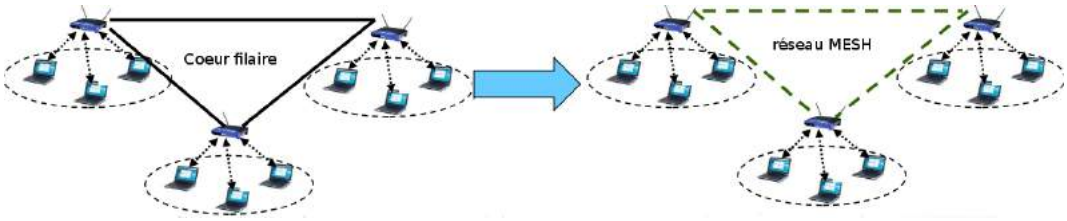


FIGURE 3.2 – Évolution du réseau filaire vers un réseau mesh.

Ce réseau sans fil est un réseau où chaque récepteur (station ou point d'accès) est également un relai voir figure **??**). Ce type de système baptisé *Mesh Network* (ou NAN pour *Neighborhood Area Network* traduit par réseau de "voisinage") permet de construire un réseau maillé (traduction de *mesh* reposant sur des technologies radio. Chaque station de base forme alors un nœud capable de communiquer avec tous les autres en même temps.

La technologie *mesh* intègre aussi la gestion de la connexion et de la déconnexion de nouveaux relais sans recourir à la reconfiguration manuelle du réseau.

3.2.3 Accès au médium

Dans la norme IEEE 802.11, l'algorithme utilisé pour l'accès au média, CSMA/CA (Carrier Sense Multiple Acces with Collision Avoidance), présente des similarités à celui utilisé dans les réseaux filaires CSMA/CD (avec Collision Detection).

Dans un réseau local Ethernet classique, la méthode d'accès utilisée est le CSMA/CD (Carrier Sense Multiple Access with Collision Detection), pour lequel chaque machine est libre de communiquer à n'importe quel moment. Chaque station envoyant un message vérifie qu'aucun autre message n'a été envoyé en même temps par une autre station. Si c'est le cas, les deux stations patientent pendant un temps aléatoire avant de recommencer à émettre.

L'atténuation du signal radio en fonction de la distance est bien plus importante que celle d'un câble. Le signal émis est reçu par son émetteur avec une puissance très supérieure à tout signal provenant des autres mobiles présents dans le réseau. Tout signal émis localement, de par sa puissance d'émission élevée, couvre donc tous les autres signaux qu'il reçoit. De plus, dans un environnement radio, deux stations communiquant avec un récepteur ne s'entendent pas forcément mutuellement en raison de leur rayon de portée. De ce fait, il est impossible pour un émetteur de détecter une collision dans un réseau sans fil. Le récepteur peut recevoir simultanément plusieurs signaux avec des puissances comparables. Dans la pratique, des collisions se produisent le plus souvent au niveau des récepteurs.

Ainsi la norme 802.11 propose un protocole appelé CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

Backoff

Avec Ethernet, on observe l'état du canal avant d'émettre. Si le canal est libre, alors nous pouvons envoyer notre trame et si nous détectons une collision, nous réémettons la trame un peu plus tard. Or dans un réseau radio il n'était pas possible de détecter directement les collisions. Si, comme pour Ethernet, nous attendrons simplement que le canal soit libre pour émettre, plusieurs mobiles en état d'attente détecteraient tous le canal libre et émettraient au même instant.

Pour 802.11, lorsque le canal devient libre, il a été décidé d'attendre une durée aléatoire supplémentaire appelée *backoff* avant d'émettre. Ce mécanisme s'applique chaque fois que le canal devient libre. De ce fait, si plusieurs mobiles veulent émettre, il y a peu de chances pour qu'ils aient choisi le même *backoff*. Celui qui a choisi le plus petit va commencer à émettre, et les autres se rendent alors compte que le canal est utilisé, vont attendre. La figure **??** représente le déroulement d'une communication de deux mobiles avec un troisième sur un canal libre.

Pour que le canal devienne libre, il faut qu'il le reste pour une période DIFS (*DCF Inter-Frame Space*). Si cela est vérifié, alors les mobiles désirant émettre choisissent un *backoff* aléatoire exprimé en un nombre de *time slots* d'une durée fixe de 20μs. Le *backoff* est choisit au hasard dans un intervalle appelé Contention Window (CW) qui est par défaut [0 ;31]. Dans l'exemple de la figure **??**, le mobile 1 a tiré 3 et le mobile 2 a tiré 5. Une fois le *backoff* choisi, tant que le canal reste libre, les mobiles décrémentent leur *backoff*. Dès que l'un d'eux a terminé (ici le mobile 1), il émet. L'autre mobile, dès qu'il détecte une activité sur le canal stoppe la décrémentation de son *backoff* et entre en période de *defering*.

Le mobile dans cet état ne pourra reprendre la décrémentation de son *backoff* que si le canal est à nouveau libre pendant DIFS. La période SIFS (*Short Inter-Frame Space*), qui sépare un paquet de données de son acquittement, empêche une prise inopportune du canal entre les données et leur acquittement.

Lorsque les données du mobile 1 ont été acquittées et que DIFS s'est écoulé sans activité sur

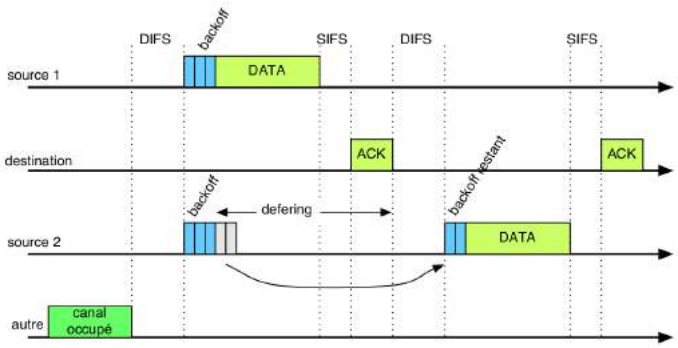


FIGURE 3.3 – Mécanisme de communication du protocole CSMA/CA.

le canal, le mobile 2 peut reprendre la décrémentation de son *backoff*. Ici, aucun autre mobile ne vient l'empêcher de terminer et il peut donc finalement envoyer ses données.

Le mécanisme de *backoff* limite seulement les risques de collision. S'il y a collision (absence d'aquitement), un nouveau *backoff* est tiré au hasard. Mais à chaque collision consécutive, la taille de la fenêtre va doubler afin de diminuer les risques de collisions. La borne inférieure de la Contention Window est toujours fixée à zéro, et la borne supérieure est comprise entre les valeurs *aCWmin* et *aCWmax* définies par la norme comme des puissances de 2 moins 1. La borne supérieure de la fenêtre est ré-initialisée à aCWmin après la transmission correcte d'un paquet.

RTS/CTS

Dans la norme 802.11, on peut utiliser un mécanisme optionnel, appelé RTS/CTS, afin d'éviter les collisions dans le cas où les différents émetteurs ne s'entendent pas. Il est basé sur un principe d'accusé de réception réciproque entre l'émetteur et le récepteur. Cette option n'est utilisable que si la taille des paquets est supérieure à un certain seuil.

Un exemple d'enchaînement des messages est donné par la figure **??**.

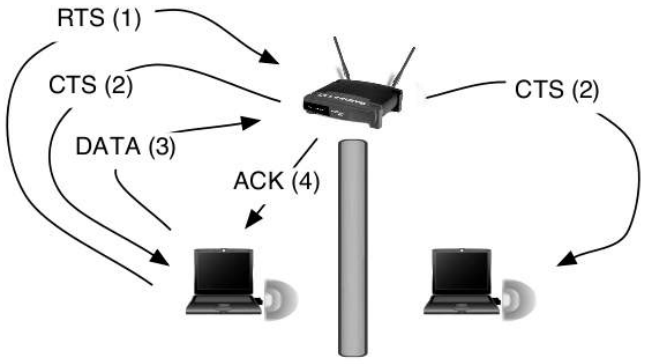


FIGURE 3.4 – Enchaînement des messages RTS/CTS.

La station de gauche voulant émettre, écoute le réseau. Si le réseau est encombré, la transmission est retardée. Dans le cas contraire, si le média est libre pendant un temps donné, appelé DIFS (Distributed Inter Frame Space), alors la station peut émettre. Dans le cadre d'une transmission à risque d'un point de vue puissance de signal ou des présence de mobiles non-détecté

(mobile de droite), émetteur et récepteur peuvent choisir de sécuriser la transmission en transmettant un message appelé *Ready To Send* (noté RTS signifiant « prêt à émettre ») contenant des informations sur le volume des données qu'elle souhaite émettre et sa vitesse de transmission. Le récepteur (généralement un point d'accès) répond par un *Clear To Send* (CTS, signifiant « le champ est libre pour émettre »), alors seulement la station commence l'émission des données.

À la réception de toutes les données émises par la station émettrice, le récepteur envoie un accusé de réception (ACK). Toutes les stations voisines patientent alors pendant un temps qu'elles considèrent être celui nécessaire à la transmission du volume d'information à la vitesse annoncée.

3.2.4 Modèles de propagation dans Network Simulator 2

Actuellement, trois modèles de propagation sans fil sont mis en œuvre dans NS2 [**?**] : *Free Space*, *Two ray ground* et *Shadowing*.

Les deux premiers modèles sont de type « tout ou rien » (figure **??**) : si la distance d entre les équipements mobiles en communication est inférieure à une certaine valeur, tous les paquets envoyés sont correctement reçus. Le modèle *Free Space*, le plus simple, ne prend en considération que le lien « en ligne droite sans fil entre l'émetteur et le récepteur. Le modèle *Two ray ground* est une amélioration du précédent par la prise en compte, en plus de l'onde « directe », de sa réflexion sur un sol. *Free space* correspond à une transmission sans fil libre de tout obstacle (dans le ciel par exemple). Le modèle *Two ray ground* définit une propagation dans un environnement avec un sol (sur terre par exemple).

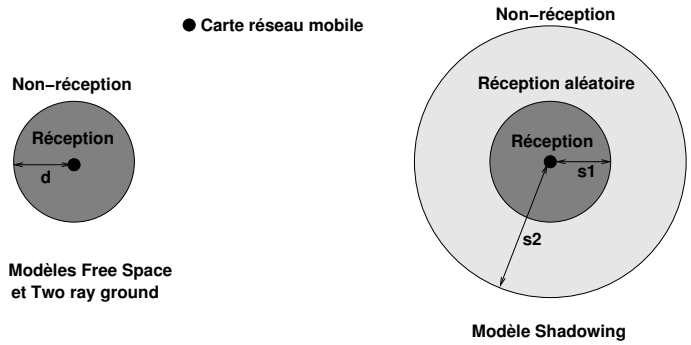


FIGURE 3.5 – Représentation des modèles de propagation dans NS2.

Si d est plus grande, aucun paquet n'est reçu. Ces deux modèles ne sont pas appropriés à une étude réaliste d'un réseau sans fil, car ils ne sont pas conformes à la réalité.

Là où les modèles précédents ne prennent en compte qu'une distance, le modèle *Shadowing* (figure **??**), introduit des événements aléatoires durant la transmission. Comme pour les autres modèles, les paquets sont toujours reçus pour une distance $d \leqslant s1$, toujours perdus pour $d \geq s2$ et reçus avec une certaine probabilité si $s1 < d < s2$. Ce dernier modèle est le plus adapté parce qu'il ressemble plus à de vrais liens sans fil et *c'est celui-ci que nous utiliserons dans toutes nos simulations.*

Dans tous les cas, la zone de réception forme un cercle autour de l'émetteur.

3.2.5 Fiabilité du réseau

L'onde radio se propage sur une certaine distance qui est dépendante de l'équipement utilisé et de l'environneemnt dans lequel elle se déplace. La qualité du signal reçu à cette distance est

fonction des obstacles traversés (murs, fenêtres, personnes, etc.). Grâce aux accusés de la couche MAC, on fiabilise la communication sans fil. Alors que dans les réseaux filaires, la congestion est la raison principale cause de perte de paquets, dans les réseaux sans fil les sources des pertes peuvent être de plusieurs types :

- la mobilité des équipements clients engendre beaucoup de déconnexion et de reconnexion suivant la distance entre la station de base, les obstacles et les mobiles. Par conséquent, beaucoup de ruptures de signal se produisent entre la station de base et le récepteur.
- un changement de route entre l'émetteur et le récepteur en cas de changement de points d'accès, ou dans les cas de réseaux *ad-hoc* auto-organisés, peut causer un désordonnancement des paquets et ceci augmente la probabilité d'un *timeout* d'un paquet et de problèmes au niveau du protocole de transport.
- le taux d'erreur dans les réseaux sans fil est élevé ce qui cause généralement des pertes aléatoires des paquets. Ce taux élevé s'explique par des interférences liées à l'environnement ou provoquées par d'autres communications travaillant sur une même bande de fréquence ou sur des bandes de fréquence voisines.
- le principe d'accès au canal radio *Mac Layer Contention* a un impact significatif sur l'augmentation du délai d'acheminement des paquets. La durée aléatoire attribuée à chaque nœud à la fin de chaque transmission permet de fournir un accès équitable à tous les clients mobiles voulant utiliser le canal mais elle rallonge le transfert et augmente donc le risque des pertes de données.

3.3 Protocoles de transports dans les réseaux sans fil

3.3.1 TCP Santa Cruz

Cette version de TCP, TCP Santa Cruz [**?**], est adaptée aux réseaux asymétriques. Il s'agit, plus précisément, de réseaux dont la bande passante descendante est généralement plus élevée que la bande passante montante comme par exemple l'ADSL. De plus, TCP Santa Cruz est bien adaptée aux réseaux sans fil.

La méthode est basée sur le délai de propagation de l'information pour arriver jusqu'à sa destination plutôt que sur le RTT des paquets contenant cette information. De plus elle permet d'identifier le sens 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 considérations est que Vegas regarde les variations dans le RTT. Le délai relatif entre les paquets est mesuré sur la base d'un marquage inscrit 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 croît ou décroît. La fenêtre de congestion est alors adaptée en fonction de l'évolution des résultats.

[**?**] présente une amélioration spécifique à la différenciation de perte pour TCP Santa Cruz.

3.3.2 TCP WestWood+

TCP Westwood [**?**] est une variante de TCP spécialement développée pour améliorer les réseaux avec une grande bande passante disponible et avec des pertes potentielles de paquets dues à la transmission ou à d'autres erreurs. Dans cette version, on utilise le taux de réception des accusés TCP pour estimer la bande passante B disponible, ajuster de la fenêtre de congestion, adapter les seuils dans les algorithmes de traitement de congestion.

Le débit d'un émetteur TCP dépend de l'arrivée des accusés de réception selon l'espacement avec lequel le récepteur les a envoyés. Si ces accusés restent plus longtemps dans les files d'attente sur le réseau, leur espacement peut être modifié. Quand ils arrivent plus rapprochés qu'ils n'ont été envoyés, l'expéditeur peut être trompé et envoyer plus de données que le réseau ne peut en accepter, ce qui mènerait à la congestion et à la perte de données. Ces accusés rapprochés sont dits *compressés*.

TCP Westwood+ [**?**] améliore l'algorithme de TCP Westwood en employant un algorithme légèrement modifié d'évaluation de bande passante qui fonctionne correctement en présence d'accusés de réception compressés. Westwood+ permet une meilleure initialisation du paramètre *ssthresh*.

Algorithme de TCP Westwood+

- On appelle DupACK un accusé identique au précédent. Si le paquet N arrive au récepteur avant $N-1$ alors son accusé est identique à l'accusé de $N-2$. Lorsque 3 DupACKs sont reçus par la source :
  $ssthresh = \max(2, (B * RTTmin)/segsize$;
  avec $RTTmin$ le plus petit RTT mesuré par la source et *segsize* la taille d'un segment TCP délivré.
  $cwnd = ssthresh$; $cwnd$ = fenêtre de congestion
- En cas de Timeout :
  $ssthresh = \max(2, (B * RTTmin)/segsize$;
  $cwnd = 1$;
- Si les accusés sont correctement reçus, la fenêtre de congestion est augmentée suivant l'algorithme de TCP Reno.

3.3.3 TIBET : Time Intervals based Bandwidth Estimation Technique

TIBET [**?**] est un nouveau système d'estimation de bande passante qui peut être mis en œuvre dans le contrôle de congestion de TCP en modifiant seulement la partie émettrice d'une connexion. Ce contrôle de congestion permet d'obtenir une estimation correcte de la *bande passante* en utilisant le taux de réception des accusés compressés ou non.

Soit n le nombre de paquets appartenant à une transmission et L_1, L_2 ... L_n la longueur en bits de ces derniers et T la durée de cette transmission (voir figure **??**). La bande passante moyenne, Bw utilisée par la connexion est donnée par $\frac{1}{T}\sum_{i=1}^{n} L_i$ et peut donc s'exprimer de la manière suivante

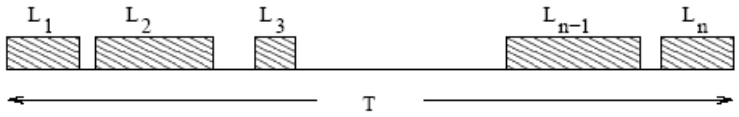


FIGURE 3.6 – Organisation temporelle des paquets

$$Bw = \frac{n\overline{L}}{T} = \frac{\overline{L}}{\frac{T}{n}}$$

L'idée de base de cet algorithme est d'effectuer une estimation sur l'émetteur de la longueur moyenne des paquets, $\overline{L}$, et la moyenne des temps entre les arrivées des paquets, $\frac{T}{n}$, séparément.

3.3.4 TCP Vegas

La version de TCP Vegas [**?**] améliore la méthode de variation de la taille des fenêtres par rapport aux autres versions de TCP. Son principe est d'évaluer la taille des tampons en entrée des routeurs. De plus elle observe l'évolution de ces tampons grâce à des calculs faits sur les mesures du 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. Cette version ne régule donc plus ses fenêtres uniquement grâce à la réception des accusés de réception. Vegas calcule le taux de transmission attendu par :

$$Taux\ attendu = \frac{Taille\ de\ la\ fenetre\ en\ cours}{RTTBase}$$

où *RTTBase* est calculé sur base d'un paquet envoyé dans le réseau sans qu'il y ait congestion (le plus petit mesuré jusqu'à présent).

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

$$Taux\ actuel = \frac{Nombre\ d'octets\ transmis}{RTT}$$

Vegas ajuste son taux de transmission pour le garder inférieur au taux de transmission attendu. Ce taux de transmission, Vegas l'ajustera en fonction du délai de propagation et du délai dû aux files d'attente.

$$Diff = Taux\ attendu - Taux\ actuel$$

Cette valeur est forcement positive ou nulle. D'autre part on définit deux seuils, α et β, qui correspondent au fait de n'avoir pas assez ou trop de paquets en transit dans le réseau. On applique ensuite les règles suivantes :

- $Diff < \alpha$, alors TCP Vegas accroît sa fenêtre de congestion linéairement durant le prochain RTT,
- $Diff > \beta$, alors TCP Vegas réduit sa fenêtre de congestion linéairement durant le prochain RTT,
- $\alpha < Diff < \beta$, alors TCP Vegas ne modifie pas sa fenêtre.

Contrôle de congestion pro-actif

Reno ne permet pas d'évaluation précise des tailles des files sur le chemin de la 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 instants 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).

Dans la phase de *Congestion Avoidance*, 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, l'algorithme n'entraîne pas de diminution de la taille de la fenêtre pour une perte remarquée après cette diminution, il ne tient compte que des paquets envoyés avant cette réduction de la taille de la fenêtre de congestion. En effet, il n'est pas sûr que pour la dernière taille de celle-ci obtenue après réduction, il y ait encore congestion.

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.

3.4 Différenciation des pertes

Les méthodes utilisées pour faire la différence entre un paquet perdu dû à une congestion et un autre perdu à cause d'une autre raison sont nombreuses. Elles seront en partie détaillées dans cette section.

Le but de la différenciation de pertes est de différencier les pertes radio des pertes de congestion. Ces mécanismes peuvent être implicites ou explicites. Les méthodes explicites sont celles qui se servent des agents déployés sur des nœuds intermédiaires du réseau. Elles essayent de différencier les pertes au niveau du récepteur sans se servir d'aucun nœud intermédiaire. Elles ont besoin du récepteur pour informer l'émetteur du type de perte : elles sont donc aussi appelées méthodes de bout-en-bout.

3.4.1 Mécanismes à information explicite

TCP Aware Local Retransmission

Le protocole *TCP Aware Local Retransmission* [**?**] utilise un système de coopération entre la couche transport et la couche liaison. Il utilise les numéros de séquences de la couche liaison avec ceux de la source TCP pour détecter la perte de données. Il utilise l'*Explicit Retransmission Notification* (ERN) afin d'éviter un mauvais comportement de la part de la source TCP. Cette méthode de gestion des pertes est destinée aux réseaux peu fiables et aux réseaux sans fil.

La méthode est mise en œuvre comme suit :

1. Lorsque le point d'accès reçoit un nouveau paquet de données TCP à partir d'une source TCP, il insère dans le paquet un en-tête LAC-PDU avec l'estampille d'un premier numéro d'ordre local. Le paquet se résume à un paquet LAC-PDU (*Link Aggregation Control Protocol Data Unit*) composé des en-têtes LAC-PDU, IP, TCP et des données, puis est envoyé à sa destination.
2. Lorsque le terminal reçoit un paquet de données TCP, il produit un accusé de réception (ACK), qui comprend un numéro (*Acknowledgement Number* AN). Il y insère également un en-tête LAC-PDU avec l'estampille d'un deuxième numéro d'ordre local.
3. Au niveau du point d'accès, une détection est faite pour savoir s'il y a eu une perte de paquets. Cette détection est fonction du numéro de l'accusé de réception (AN), la date du deuxième numéro d'ordre local, tous deux reçus dans l'accusé de réception, ainsi que de la date du premier numéro d'ordre local, qui est stocké dans le point d'accès. Si une perte de paquet est détectée, comme pour l'étape 1, le point d'accès met à jour ses temps d'enregistrement du premier numéro d'ordre local dans l'en-tête PDU-LAC, et le retransmet. Lorsque des pertes dues à la congestion sont improbables, l'accusé de réception (ACK), dont le bit ERN est activé, est livré à la source TCP.
4. La retransmission explicite ERN mentionné ci-dessus, est un champ d'un bit. Lorsqu'un paquet de données, correspondant au numéro du paquet d'acquittement (AN), est sur l'équipement gérant le lien de faible fiabilité (le point d'accès dans le cas des réseaux sans fil), le bit de retour ERN pour le paquet d'acquittement (ACK), devant être envoyé à la

source TCP, est initialisé. Lorsque la source TCP reçoit un accusé de réception (ACK) avec le bit mis, en même temps la retransmission (Fast-Retransmission ou timeout retransmission) des paquets de données TCP se produit, ce qui correspond à l'acquittement (ACK), le paquet de données concerné est retransmis sans aucune opération de réduction de la fenêtre d'envoi et donc de la fenêtre de congestion.

Cette méthode simple de différentiation des pertes présente l'avantage de ne pas nécessiter de redéfinition d'un équipement de communication sans fil et peut donc s'adapter à toutes les transmissions TCP sur IP par une modification des extrémités seules.

Snoop

La différenciation explicite de perte pour des flux TCP peut aussi être utilisée par l'utilisation des agents Snoop [**?**]. Des agents Snoop sont appropriés à la topologie sans fil pour les extrémités du réseau (premier ou dernier saut en réseaux mobiles). Considérons un flux TCP provenant d'un serveur fixe en direction d'un client mobile. L'agent Snoop dans le point d'accès surveille les paquets TCP expédiés et les accusés retournés, maintenant ainsi un historique des paquets TCP expédiés mais non acquittés. Il détecte une perte de paquet sur le lien sans fil en voyant les doublons d'accusés (*DupACKs*) ou par la fin de ses temporisateurs locaux. Après la détection, il retransmet, s'il a été mis en cache, le paquet perdu au client, puisque ce sont des paquets perdus sur le lien sans fil. En plus, il supprime les DupACKs correspondant aux pertes sans fil, de ce fait il évite de transmettre toutes les fausses informations de congestion à l'expéditeur. L'agent Snoop supprime de la partie radio les pertes de congestion sans aucun changement sur le protocole. Par contre il n'évite pas l'expiration du timeout dû au retard des paquets.

Par contre les agents Snoop ne sont pas appropriés aux flux temps réel puisque ces flux sont généralement transmis par des protocoles non-fiables. Pour que les agents Snoop détectent la perte de paquet, le récepteur doit reconnaître chaque paquet reçu. Les agents Snoop sont une méthode possible pour des mécanismes de contrôle de congestion comme le contrôle de congestion binomial [**?**] (où le récepteur accuse chaque paquet comme pour TCP). Mais la suppression des DupACKs pour les paquets perdus sur le lien sans fil peut entraîner une augmentation du retard des paquets qui est préjudiciable aux applications temps réel.

ELN

ECN est également un mécanisme possible pour des flux en temps réel. ECN est employé par les routeurs pour signaler la congestion dans le réseau par le marquage de paquet IP. Malheureusement les expériences dans [**?**] ont montré que l'ECN est un mauvais prédicateur de pertes de congestion.

La méthode ELN [**?**], pour *Explicit Loss Notification* utilise le même principe que le mécanisme ECN pour TCP. Le bit renommé ELN de l'en-tête TCP des accusés de réception est initialisé à 1 afin d'indiquer à l'émetteur que la non réception d'un ou plusieurs paquets(s) est due à une perte sur le réseau sans fil. Ce mécanisme nécessite un agent Snoop sur le point d'accès. Cet agent doit être capable d'analyser les paquets qui transitent à travers le point d'accès et plus précisément de garder en mémoire les numéros des paquets non transmis (paquets n'ayant pas reçu d'acquittement au niveau MAC 802.11). Le point d'accès renvoie les acquittements TCP demandant les paquets perdus et mis en mémoire vers l'émetteur en positionnant le bit ELN à 1. Pour éviter de marquer une perte par congestion comme étant une perte sans fil, le système ne met pas en mémoire les paquets perdus sur la partie sans fil si le nombre de paquets présents dans la file d'attente du point d'accès n'est pas proche du maximum de la taille de cette dernière.

Quand l'émetteur reçoit un paquet marqué ELN, il retransmet le paquet manquant et met à jour une variable afin de garder une trace de la dernière retransmission induite par ELN. Ceci permet d'éviter les multiples retransmissions du même paquet pour chaque accusé marqué ELN reçu. On différencie ainsi les pertes sur la partie sans fil des pertes de congestion en différenciant les accusés avec ELN ou sans.

Lorsqu'il retransmet un paquet grâce à l'information ELN, l'émetteur ne réduit pas sa fenêtre de congestion. Il court-circuite le système Fast Recovery de TCP et continue d'augmenter sa fenêtre pour chaque accusé reçu. L'émetteur maintient ainsi son augmentation normale de débit (voir figure **??** en ne tenant compte que des pertes dues à la congestion).

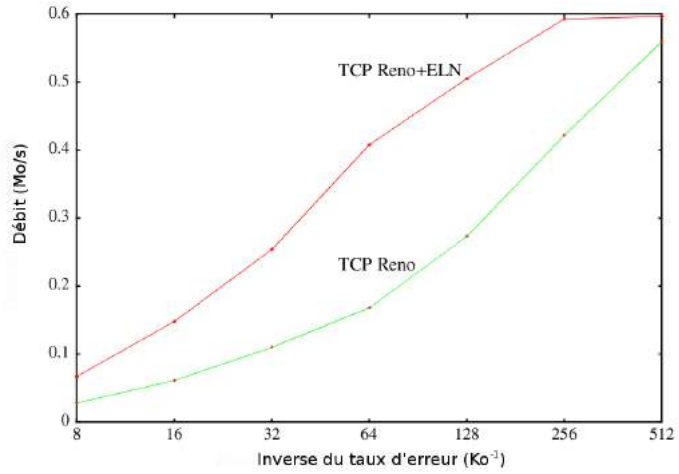


FIGURE 3.7 – Comparaison entre le débit de TCP Reno et celui de TCP Reno avec ELN par rapport au taux de perte [**?**].

Cette méthode de différenciation implique de se placer dans un réseau de type infrastructure ce qui exclut les réseaux de type *ad-hoc*. De plus, l'agent installé dans le point d'accès intervient au niveau TCP, ce qui suppose une modification conséquente du micro-code de la carte. Enfin, la notification intervient après la perte (les acquittements TCP doivent arriver au point d'accès).

3.4.2 Information implicite : les mécanismes de bout-en-bout

Les mécanismes de bout-en-bout fonctionnent au niveau de la couche transport. Ils se basent sur les temps de transmissions des paquets et sur le délai entre la réception des paquets.

Méthodes basées sur le délai entre paquets

On s'intéresse par exemple à la variation du délai dans une seule direction. Cette méthode a été utilisée par Garcia-Luna Aceves [**?**] qui interprète la perte en tant que signe de congestion si ce délai augmente.

Sinon, c'est un signe de perte dans le réseau sans fil. Barman et Matta [**?**] proposent que lorsque la variation du RTT est élevée, on peut estimer que le réseau entre la source et la destination est congestionné. Si la variation reste faible, le réseau est stable (congestionné ou non).

Vidya et Biaz [**?**] ont proposé un mécanisme basé sur les délais entre les arrivées de paquet (*Inter-Arrival Time*) IAT pour le dernier lien sans fil, qui correspond au goulot d'étranglement. Ils approximent le minimum des délais entre les arrivées de paquet pendant une connexion (T_{min})

par le temps pris par un paquet pour traverser le dernier lien radio. Si un paquet est perdu et que le paquet suivant arrive approximativement après T_{min}, la perte est classifiée en tant que perte de congestion donc due aux files d'attente. Si le paquet suivant le paquet perdu arrive après plus ou moins $2 \times T_{min}$, la perte est considérée comme une perte due au réseau sans fil.

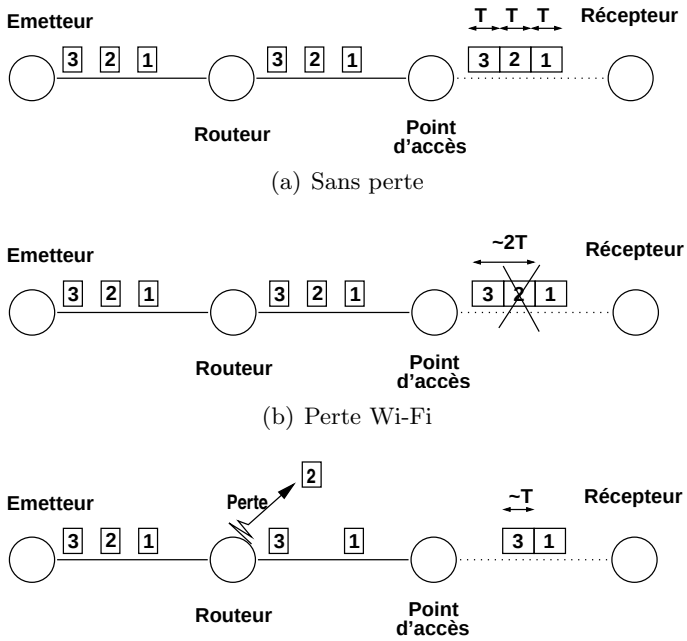


FIGURE 3.8 – Cas simple d'utilisation de la méthode Biaz.

Généralisons cette démonstration au cas de n pertes de paquets successives. Soit T_i le temps entre l'arrivée du paquet i et le paquet $i + n + 1$. On admet que les n paquets sont perdus sur la liaison sans fil si et seulement si :

$$(n + 1)T_{min} \leqslant T_i < (n + 2)T_{min}$$

FIGURE 3.9 – Méthode mBiaz

La mauvaise classification d'un nombre important de pertes causées par des congestions réseau a conduit à une évolution de l'algorithme de Biaz : *Modified Biaz* ou *mBiaz* [**?**]. Dans cette amélioration, les auteurs ont choisi de redéfinir la limite supérieure de la zone de perte due au réseau sans fil :

$$(n + 1)T_{min} \leqslant T_i < (n + 1.25)T_{min}$$

Cette nouvelle borne supérieure est obtenue de manière empirique et détaillée dans [**?**].

Statistical Packet Loss Discrimination

[**?**] propose une nouvelle méthode *Statistical Packet Loss Discrimination* SPLD de discrimination des pertes de paquets au niveau d'un récepteur mobile dans un réseau sans fil. Cette approche est fondée sur la valeur statistique de l'IAT des paquets reçus comme Biaz. Par simulation, il montre que SPLD est plus précis que celui de Biaz pour deux cas : pour un seul flux ou pour de multiples flux dans le réseau. Ce système peut être utilisé indifféremment avec TCP et le contrôle de congestion TFRC ainsi que dans le cas de diffusion vidéo dans la maison par liaisons sans fil.

Le *monitoring module* collecte des paquets $N_{packets}$ pendant un certain temps. S'il n'y a pas de perte pendant ce temps, un module *statistic manager* met à jour plusieurs statistiques :

- l'IAT courant $IAT_{curr} = \sum T_i/(N_{packets}$ avec $1 < i < n$,
- la moyenne des IAT, IAT_{stable}.

Lorsqu'il y a des pertes, un module *discriminator* vérifie l'inéquation suivante : si l'inéquation $IAT_{curr} \leqslant IAT_{stable}$ est vraie, alors les pertes sont dues à une congestion sur le réseau, sinon il s'agit de pertes provoquées par le réseau sans fil.

Spike

Tobe [**?**] emploie le Relative One-way Trip Time $ROTT$ qui est la mesure du temps pris par un paquet pour aller de la source à la destination. Ce mécanisme est basé sur le fait que lorsqu'on trace la courbe du $ROTT$ en fonction du temps sur le récepteur, on peut observer des pics pendant la congestion. À partir de cette observation, on peut assimiler les pertes pendant ces pics comme de pertes engendrées par une congestion du réseau.

Lorsqu'il y a des pertes en dehors de ces pics graphiques, elles sont classifiées en pertes radio. Le problème avec ce mécanisme est que la congestion peut se produire sur un ou plusieurs routeurs à la fois, ayant pour résultat des pics de tailles variables. La difficulté est donc de détecter des pics de différentes tailles et de définir le contenu de ces pics. Ces pics sont définis, dans ce premier article, par des seuils codés en dur dans l'algorithme et fonction uniquement du ROTT (voir figure **??**) :

- $B_{spikestart} = ROTT_{min} + 20ms$
- $B_{spikeend} = ROTT_{min} + 5ms$ où $ROTT_{min}$ est le ROTT minimum observé jusqu'à maintenant.

Le mode *Spike* est défini par ces deux seuils. A partir de cela, on admet que si une perte qui intervient pendant le mode *spike* alors c'est une perte de congestion, sinon c'est considérée comme une perte due à la liaison radio.

Mais dans le cas où la connexion ne subissait pas de variations assez importantes, on ne pouvait fournir aucune informations sur la classification des pertes. Si une valeur de $B_{spikestart}$ est surestimée à cause d'un niveau élevé des files d'attente des routeurs, l'algorithme se trompe dans l'estimation des pertes de congestion et ceci conduit à une mauvaise classification des pertes et peut provoquer une forte congestion dans le réseau suivant les actions menées par la source en fonction de la classification.

[**?**] présente un autre algorithme basé sur le mode *Spike*. Ce nouvel algorithme modifie les valeurs des seuils en fonction de l'amplitude de la variation du ROTT au cours du temps :

- $B_{spikestart} = ROTT_{min} + \alpha(ROTT_{max} - ROTT_{min})$ où $ROTT_{max}$ est le ROTT maximum avant d'entrer en mode *Spike*
- $B_{spikeend} = ROTT_{min} + \beta(ROTT_{max} - ROTT_{min})$ où $ROTT_{min}$ est le ROTT minimum avant de quitter le mode *Spike*

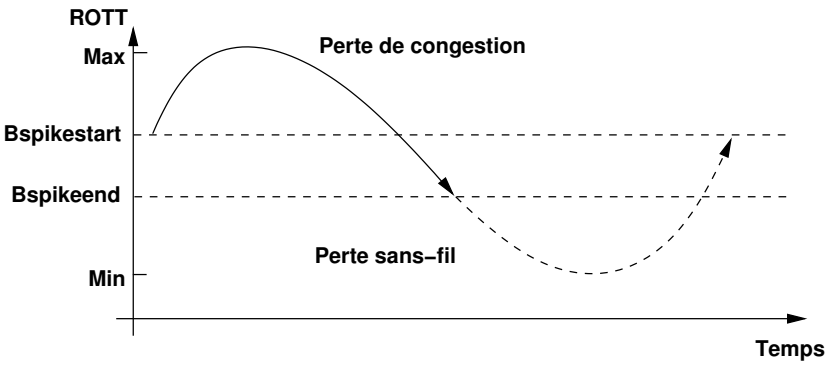


FIGURE 3.10 – Méthode Spike

– $\alpha \geqslant \beta$ avec $\alpha = 1/2$ et $\beta = 1/3$ (résultats optimaux empiriques)

Packet Loss Classification

Basée sur la même idée, la méthode PLC [**?**] pour *Packet Loss Classification* s'attarde, elle, un peu plus sur ce que les autres appellent la zone de mode *spike*. La PLC ne revient pas sur le principe de classement des pertes dans le cas où le ROTT est faible ou proche de son maximum. Ces pertes restent définies respectivement comme perte sur le lien radio et perte de congestion. Cette méthode redéfinit de nouveaux seuils TG_{low} et TG_{up} qui correspondent exactement à $B_{spikestart}$ et $B_{spikeend}$ aux constantes près α et β (ici $\alpha = 10.8$ et $\beta = 0.3$).

Ce qui différencie PLC de Spike, c'est dans la zone centrale de détermination, lorsque ROTT est entre TG_{up} et TG_{low} :

– Si le ROTT observé augmente c'est que les pertes sont de type perte de congestion.
– Si le ROTT observé baisse alors c'est une perte due au réseau sans fil

Zigzag

La méthode Zigzag [**?**] se sert également de la variable $ROTT$. Le récepteur garde en mémoire une évaluation du $ROTT$ moyen, $ROTT_{mean}$, et de sa variance $ROTT_{dev}$. En se basant sur le nombre de pertes et la différence entre le $ROTT$ et le $ROTT_{mean}$ (moins sa déviation $ROTT_{variance}$) courants, ils proposent certaines conditions pour la classification.

$ROTT_{mean}$ et $ROTT_{dev}$ sont calculés comme suit : $\alpha = 1/32$ empiriquement

$ROTT_{mean} = (1 - \alpha)ROTT_{mean} + \alpha \times ROTT$

$ROTT_{dev} = (1 - 2\alpha)ROTT_{dev} + 2\alpha \times \mid ROTT - ROTT_{mean} \mid$

Selon les auteurs, le délai dans un seul sens augmente de façon monotone en cas de congestion et le délai entre les paquets augmente en cas de perte radio donc l'augmentation du ROTT en fonction du nombre de paquet perdu est plus ou moins importante suivant la nature des pertes (voir répartition sur la figure **??**).

ZBS

ZBS [**?**] est une méthode hybride de différenciation des pertes. Elle utilise plusieurs algorithmes que nous venons de détailler Biaz, mBiaz, Spike et ZigZag. ZBS change dynamiquement entre les différentes méthodes de différenciation d'après les règles suivantes :

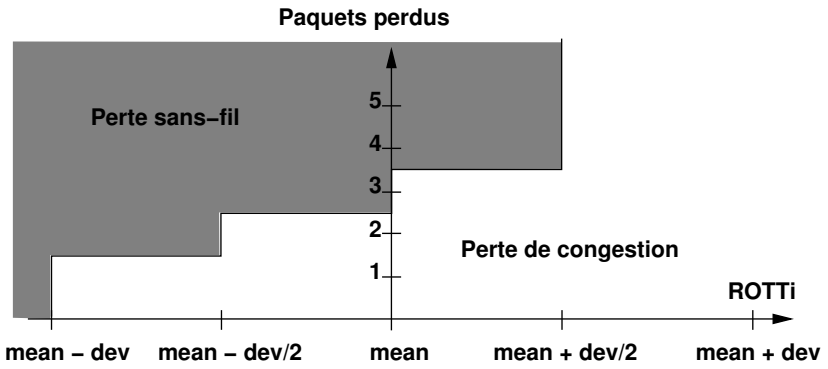


FIGURE 3.11 – Méthode Zigzag

Si $ROTT < (ROTT_{min} + 0.05 \times T_{min})$ utiliser Spike
Sinon :
Si $T_{narr} < 0.875$ utiliser ZigZag
SinonSi $T_{narr} < 1.5$ utiliser mBiaz
SinonSi $T_{narr} < 2$ utiliser ZigZag
Sinon utiliser Spike
avec $T_{narr} = T_{avg}/T_{min}$

Explication : ZBS, comme le montre la figure **??**, utilise Spike quand la moyenne de ROTT est très proche de celui de $ROTT_{min}$, ce qui est la cause d'une sous-utilisation du réseau, soit lorsque $T_{narr} \geqslant 2$ indiquant l'existence de multiples flux en concurrence sur le réseau.

Il applique mBiaz si le goulot d'étranglement n'est pas sous utilisée et que les conditions du réseau indiquent que la liaison sans fil n'est pas partagée ($T_{narr} \simeq 1$). Enfin il implémente ZigZag si les conditions du réseau ne sont pas définies surtout en début de connexion et lors du départ ou l'arrivée de flux durant la connexion.

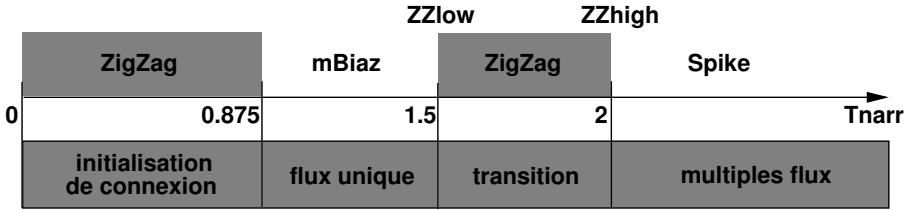


FIGURE 3.12 – Décomposition de la méthode ZBS.

Trend-Loss-Density-based TD

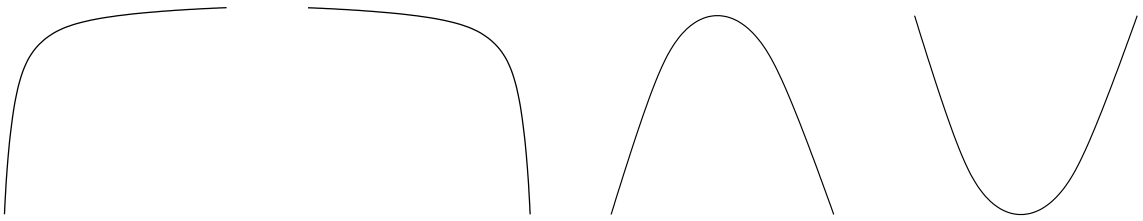
Lorsque le ROTT augmente, cela indique une tendance de surcharge du réseau ce qui causera probablement un rejet de paquets à cause du remplissage des files d'attente des routeurs. La perte de paquet provoque alors une diminution du débit et par conséquent une baisse de la valeur du ROTT.

L'algorithme de différenciation proposé dans la méthode Trend-Loss-Density-based (TD) dans [**?**] repose sur deux facteurs principaux, la tendance de perte et sa densité :

1. La tendance (*Trend*) précise où la perte est survenue (autour d'un pic de la courbe ou non).
2. La densité de perte (*Loss Density*) spécifie combien de fois la perte arrive

La tendance : Soit $f(x) = ax^2 + bx + c$ la fonction qui donne la courbe de ROTT. Pour n paquet reçus, avec les notations x_1 pour le premier paquet de la séquence arrivé, x_n le dernier et x_{apex} pour le sommet de la courbe, il y a quatre types de courbes de tendance (voir figure **??**) :

1. Haute : si $f(x_1) < f(x_n)$ avec $(x_n < x_{apex} \parallel x_1 > x_{apex})$.
2. Basse : si $f(x_1) > f(x_n)$ avec $(x_n < x_{apex} \parallel x_1 > x_{apex})$.
3. Convexe : si $(x_1 < x_{apex} < x_n)$ avec $c > 0$.
4. Concave : si $(x_1 < x_{apex} < x_n)$ avec $c < 0$.



(a) Haute (b) Basse (c) Convexe (d) Concave

FIGURE 3.13 – Type de courbe de ROTT.

La densité : Prendre en considération les caractéristiques du réseau sans fil pour calculer le seuil de la densité pour chaque courbe de tendance. Si la densité de perte excède ce seuil, les pertes sont considérées comme pertes de congestion. Pour une séquence de paquets où il y a une perte, TD utilise la fonction d'autocorrélation pour calculer la densité de perte. Considérons un paquet x_i perdu comme le centre d'une séquence s_i contenant les paquets en corrélation avec x_i et qu'il y a autant de paquets dans les deux côté (à distance $rd = 7$ paquets de chaque côté).

Classification : S'il n'y a qu'une seule perte x_i dans s_i ==> perte radio
sinon :
Si la tendance de x_i appartient à Basse, ConvexeBasse ou ConcaveBasse alors il s'agit surement d'une perte de congestion
Si la tendance de x_i appartient à ConvexeHaute ==> le seuil = la densité (5 paquets)
Si la tendance de x_i appartient à Haute, ou ConcaveHaute ==> seuil = la densité (2 paquets)

S'il y a une perte et qu'il existe plusieurs paquets perdus autour d'elle, cette perte est considérée comme perte de congestion. Si les pertes arrivent de manières séparées, ce sont des pertes radio.

Récapitulatif

Biaz, mBiaz et SPLD sont tous basés sur l'IAT *Inter Arrival Time*. Ils ont une faible performance car ils n'utilisent pas les informations pour vérifier les conditions du réseau. Par contre,

dans le cas d'une connexion unique à un réseau sans fil problématique, Biaz et mBiaz marchent bien.

SPLD marche mieux en présence de plusieurs flux sur le réseau.

ZigZag repose sur le nombre de pertes et la déviation de ROTT mais il ne marche pas bien dans beaucoup de situations.

ZBS sélectionne différentes méthodes (Spike, ZigZag ou mBiaz) mais sa performance globale est plus faible que Spike seul.

Spike et PLC ont souvent de très bonnes performances car ils utilisent des valeurs ajustées du maximum et du minimum de ROTT.

TD et le meilleur quand le niveau de congestion est très élevé. Mais il est moins bon que Spike et PLC quand le trafic sur le réseau est très faible.

Inconvénients

Une majorité de ces mécanismes de bout-en-bout ont été comparés dans [**?**]. Ils ont généralement des contraintes topologiques et donc le besoin d'être adaptés pour des topologies plus spécifiques.

Malheureusement, les mécanismes basés sur l'IAT ont deux problèmes principaux. Premièrement, avec plus d'un flux, les paquets de différents flux sont mélangés et le paquet passant après qu'une perte sans fil peut ne pas venir juste après T_{min}. Deuxièmement, dans la pratique, la bande passante disponible sur le lien réseau sans fil varie souvent au cours du temps : il y a variation de la distance entre le mobile et le point d'accès ou encore une variation de la répartition des mobiles présents dans un réseau *ad-hoc* et le temps pris par un paquet pour traverser le lien sans fil ne cesse de changer, causant ainsi une fausse interprétation.

Les méthodes de différenciation basées sur le ROTT ne permettent pas réellement de distinguer entre une perte de congestion et une perte du sans fil. Elles permettent de décider quel est le type de perte le plus dominant, elles fournissent une sorte de degré de congestion du réseau mixte. Bien que certains de ces mécanismes améliorent le débit, ils souffrent généralement de fausse classification. Ainsi les pertes de congestion dans le réseau peuvent augmenter et la propriété TCP-Friendly des flux peut diminuer. L'émetteur ne peut donc pas utiliser les accusés de réception pour ajuster son débit puisque les informations peuvent être invalides.

3.5 Propriété TCP-Friendly

Une des exigences essentielles pour un nouveau protocole, c'est qu'il doit être TCP-Friendly ou TCP-équitable. Cette propriété étant une demande inévitable pour un nouveau protocole, de nombreux chercheurs l'utilisent pour tester si leur proposition est acceptable ou non. Maintenant, si il n'existe pas de définition claire de cette exigence, les chercheurs ont chacun leur idée claire sur la façon de mesurer cette propriété.

L'article de Floyd et Fall de 1999 [**?**] est souvent pris pour référence dans ce besoin d'une définition claire et crédible. Il dit : « Nous disons qu'un flux est TCP-Friendly si son taux d'arrivée de paquets ne dépasse pas celui d'une connexion conforme à TCP dans les mêmes circonstances. »

L'indice d'équité de Chiu et Jain [**?**] créé dans les années 80, une époque où ni les topologies de réseau ni les tendances du trafic actuelles n'ont été envisagées, est souvent appliquée pour vérifier si un flux non-TCP répond à la définition de [**?**]. Dans des recherches récentes, d'autres définitions sont apparues. Par exemple, dans [**?**] de 2004, la propriété TCP-Friendly est définie comme suit : « Le débit d'un flux non-TCP ne doit être ni supérieur, ni inférieur à long terme

au débit d'une connexion TCP fonctionnant sur le même chemin et dans les mêmes conditions réseau. »

Cette définition est reprise dans de nombreux articles [**?**], [**?**] et [**?**].

Les auteurs citent un article de Widmer [**?**], où une définition plus générale est donnée : « L'équité TCP pour l'unicast - Un flux unicast est considéré comme TCP-Friendly quand il ne réduit pas à long-terme le débit de tout flux TCP coexistant plus qu'un autre flux TCP le ferait sur le même chemin dans les mêmes conditions réseau. L'équité TCP pour le multicast - Un flux multicast est défini comme TCP-Friendly quand, pour chaque paire émetteur-récepteur, le flux multicast a la propriété d'être unicast TCP-Friendly. »

Avec cette définition, un flux non-TCP peut surpasser TCP et toujours être TCP-Friendly.

Lors d'une évaluation du contrôle de congestion TFRC face à celui de TCP sur des liens sans fil et dans le cas de transmissions vidéo [**?**], les auteurs introduisent le terme d'« équité opportuniste »avec une définition similaire à celle utilisée par Widmer : « L'équité opportuniste est la capacité d'un nouveau flux d'utiliser la bande passante qui ne serait pas utilisée par les flux existants. »

L'article [**?**] approfondit cette dernière définition. Un nouveau flux peut utiliser la bande passante qu'un flux aurait utilisé dans les mêmes circonstances et de toute la bande passante inutilisée par les flux existants. On remarquera que dans cette définition, le mot "TCP" est absent. Ainsi, elle peut également se référer à un autre type de flux comme TFRC. Dans [**?**], les auteurs écrivent qu'ils ne connaissent pas de définition qui s'applique à la fois aux réseaux filaires et sans fil, de sorte qu'ils en créent une eux-même : « Un protocole est TCP-Friendly si il est équitable avec TCP dans les réseaux filaires (selon la définition Floyd), et peut obtenir de meilleures performances que TCP dans les réseaux sans fil. »

Les différentes définitions ci-dessus, semblent être équivalent à première vue, mais ont conduit à des conclusions différentes de la part des chercheurs et ont surement freiné le développement d'une nouvelle norme. Dans un environnement Internet de plus en plus mobile, les performances des protocoles dans les réseaux sans fil doivent être la principale priorité. Comme les versions traditionnelles de TCP ne fonctionnent pas de façon optimale, il n'a pas donc pas sens à essayer de limiter le débit de TCP. Par conséquent, les trois dernières définitions semblent plus aptes aux travaux actuels et futurs dans ce domaine.

3.6 Conclusion

On a pu voir dans ce chapitre qu'obtenir de bonnes performances dans les réseaux sans fil est beaucoup plus complexe que les réseaux filaires. Cette complexité est due à la mobilité des équipements et une qualité de liaison variable au cours du temps et de l'espace. Si des erreurs dans la transmission de paquets sont presque inimaginables sur un câble de cuivre ou la fibre optique, elles font partie intégrante de la liaison radio.

Les principaux protocoles de transport dans le domaine des réseaux sans fil s'attardent sur le fait d'estimer au plus juste la bande passante sans aller jusqu'à la saturation de celle disponible.

On remarque que beaucoup d'équipes se sont intéressées à la différenciation des pertes sur des réseaux hétérogènes. Les deux méthodes, explicite ou implicite, demandent chacune plus ou moins de déploiement sur les équipements de liaison entre les réseaux filaires et sans fil (pour la méthode explicite) ou sur les émetteurs et récepteurs au point de vue protocole de transport (pour la méthode implicite).

La différenciation des pertes sans fil et de celles de congestion est une étape nécessaire au bon fonctionnement des protocoles de transport existants et à venir sur des réseaux hybrides.

CHAPITRE 4

Différenciation de pertes dans les réseaux sans fil

Sommaire

4.1 Introduction

Il est maintenant montré que le protocole TCP n'est pas adapté aux connexions sans fil. La principale préoccupation est que les pertes/retards sont provoquées par des *interférences* souvent temporaires, donc aucun mécanisme de congestion ne devrait être activé.

De nombreuses études traitent l'effet des pertes de paquets dans les réseaux liens sans fil (802.11) sur TCP, mais seulement quelques unes traitent des effets du retard des paquets au cours de la retransmission. Il est intéressant de noter que des retransmissions MAC qui aboutissent sont beaucoup plus fréquentes que les pertes au niveau MAC.

Dans les réseaux sans fil, les retransmissions MAC sont les effets des interférences radio. Parce que ces pertes sont temporaires et apparaissent de façon aléatoire, elles n'ont aucune valeur utile, mais induisent une erreur d'appréciation. Par conséquent, plusieurs ouvrages proposent des méthodes pour éviter leurs effets. Mais des ouvrages ayant trait aux retransmissions MAC se concentrent généralement sur l'influence du RTT sur le RTO (*Retransmission TimeOut*) du protocole TCP [**?**, **?**, **?**].

Toutefois, il existe plusieurs versions TCP qui utilisent leur RTT pour déterminer leur débit d'envoi, par exemple Vegas [**?**] utilise tous les échantillons de RTT, Westwood+ [**?**] et Tibet [**?**] utilisent le plus petit RTT. [**?**] montre que, dans certains cas, un gain de 10 % en débit est possible si le RTT réel est remplacé par le RTT ajusté.

Dans ce chapitre, nous considèrerons un réseau d'accès sans fil maillé, un réseau mesh. De ce réseau nous extrayons une toute petite partie, la liaison entre un point d'accès et une ou plusieurs stations.

Nous proposons un mécanisme qui permet à l'expéditeur de prendre en compte le temps pris par les retransmissions. Une option TCP est ajoutée, avec un champ. Aussi, une minuterie est ajoutée à la carte réseau. Chaque fois qu'un paquet est transmis, la carte réseau conserve en interne le temps pris pour sa retransmission, en utilisant l'horloge interne. Le champ est retourné à la source dans un accusé de réception. La source est donc consciente du temps perdu dans toutes les retransmissions MAC.

4.2 Différenciation des pertes

4.2.1 Définition d'une perte de paquet

Une perte de paquet peut être causée de deux manières et en deux endroits différents sur un réseau hétérogène :

Sur les routeurs : il s'agit alors d'une perte immédiate due à la surcharge en amont ou en aval de l'équipement. On parle alors de goulot d'étranglement ou de congestion.

Sur un équipement du réseau Wi-Fi : les mécanismes de 802.11 font que si un paquet est perdu lors d'une transmission, il est automatiquement retransmis. Mais le nombre de retransmissions est limité. Dans cette partie de la thèse, nous considérons comme paquet perdu, un paquet perdu au sens de la couche Transport et donc ayant épuisé tous ses essais de transmissions sur le réseau sans fil.

4.2.2 Fausse congestion

On supposera que le client mobile restera dans la zone de couverture la plus petite, présentant la bande passante la plus grande pour la norme utilisée : 11Mbps pour le 802.11b, 54Mbps pour le 802.11g, etc.

Dans le cas d'un réseau mixte (avec une partie filaire et une partie sans fil) on doit pouvoir faire la différence entre une congestion survenant sur la partie filaire et une perte sur la partie sans fil. Cette perte en Wi-Fi est généralement matérialisée par un nombre de retransmissions au niveau MAC, supérieur à une valeur définie au niveau matériel.

La perte d'un paquet au niveau de la transmission sans fil sera remontée à la source par le biais du protocole de niveau transport, TCP par exemple. Cette perte sera considérée par la source comme étant intervenue sur la transmission dans sa globalité et sera donc traitée par une réduction de la fenêtre de congestion suivant l'algorithme du protocole utilisé.

Dans le cas que nous considérons, il n'est pas obligatoire de réduire la fenêtre de congestion du serveur, en effet le client mobile peut être momentanément en limite de zone de couverture ou subir une série d'interférences dues à une modification temporaire de son environnement immédiat.

Ces interférences multiples mais temporaires ne correspondent pas à une baisse de la bande passante mais à une interruption temporaire de signal. La capacité nominale du lien radio est théoriquement toujours la même. Réduire la fenêtre de congestion entraînerait une baisse du taux de transfert et donc de la qualité du rendu du transfert (dans le cas d'un transfert vidéo par exemple) sur une longue période avant le retour à la valeur maximale de celle-ci.

Nous nous proposons de faire en sorte d'éviter cette réduction de la fenêtre de congestion dans le cas d'une perte de paquets due à un nombre trop important de retransmissions au niveau de la couche MAC de la norme Wi-Fi. Il nous faut donc différencier une perte au niveau de la partie filaire d'un réseau, d'une perte au niveau de celle sans fil.

Pour cela plusieurs méthodes de contrôle des pertes s'offrent à nous :

– **Définitive** : On peut ne pas tenir compte du tout des pertes dues aux retransmissions successives au niveau du réseau sans fil Wi-Fi. Cela signifie que la source, dans ce cas là, réagit comme s'il n'y avait eu aucune pertes. Elle continue d'envoyer ses données au même débit et elle ne réagit qu'aux pertes causées par les files d'attente des routeurs.

– **Temporaire** : On peut sauvegarder la valeur du seuil *SSthresh* effective ainsi que la taille de la fenêtre de congestion avant la perte de paquet, traiter la perte comme étant une congestion avec utilisation de l'algorithme actuellement en place (TCP Reno, New Reno, etc), puis au bout de renvois réussis, revenir à notre ancienne configuration en ce qui concerne la taille de la fenêtre de congestion et la valeur de *SSthresh*.

Pour pouvoir appliquer un de ces contrôles de congestion spécifiques, la source doit être informée de la perte due aux retransmissions multiples au niveau Wi-Fi. Plusieurs possibilités quant à la méthode à utiliser pour faire transiter cette information sont disponibles : directe et indirecte, présentées dans les sections suivantes

4.2.3 L'information explicite directe

Comme les routeurs peuvent informer une source de données d'une congestion au moyen d'un *paquet ICMP* (voir [**?**]), on peut supposer qu'un point d'accès sans fil pourrait faire de même afin de signaler qu'un paquet a été retransmis plusieurs fois sans succès (c'est-à-dire sans que l'émetteur ait reçu l'accusé de réception correspondant à ce paquet). Le paquet ICMP contiendrait le numéro de séquence TCP du paquet en question. La source ainsi informée pourrait retransmettre directement le paquet ayant le même numéro de séquence sans attendre la réception de trois *DUPAcks* portant ce numéro.

Plus tard, la source, à la réception des trois *DUPAcks* ne devra pas en tenir compte. Cette méthode d'information voit son efficacité prendre de l'importance avec l'utilisation de l'option *Selective Ack* (SACK) de TCP. En effet, dans ce cas, la source est informée de tous les paquets reçus, il y a donc un gain de temps au niveau récepteur dans l'envoi des accusés.

Cette méthode supprime donc la gestion des pertes au niveau d'une liaison sans fil. Elle est associée avec la méthode de contrôle définitive.

4.2.4 L'information explicite indirecte

Calqué sur le principe d'ECN, on peut transmettre une information à l'émetteur par le récepteur.

Méthode ECN

Le mécanisme d'ECN peut nous permettre de déterminer deux informations essentielles : le niveau et le type de congestion sur le réseau. Tant qu'il y a de paquets marqués CE (Congestion Experienced), le réseau est congestionné et il faut par conséquent diminuer le débit.

Une indication utile pour pouvoir faire la différence entre un paquet perdu à cause d'une congestion et une perte due à une erreur provoquée par le comportement de la transmission sur le canal radio. En fait, le récepteur examine l'information d'ECN fournie par le réseau, mise dans l'en-tête du paquet et retourné à l'émetteur dans le vecteur d'accusés. S'il y a une perte et que le paquet n'est pas marqué CE, la source diminue aussi son débit mais d'un tout petit peu (comme nous l'avons déjà évoqué, c'est le problème dont souffrent généralement les protocoles de transport dans les réseaux radio). Cette différenciation impose que les routeurs congestionnés ainsi que la source et la destination soient compatibles ECN.

Enfin, en ce qui concerne la différenciation de pertes, si nous détectons une perte et que cette perte arrive alors que le réseau n'est pas congestionné (pas d'ECN), nous diminuons le débit de 1/8 de la fenêtre de congestion actuelle, donc $cwnd = cwnd * 7/8$. Cette valeur de diminution est obtenue empiriquement.

ERLN (*Explicit Retransmission Loss Notification*)

Suivant le même principe de transmission d'information qui est mis en place dans le cas de ECN, on peut définir un nouvel ajout à TCP et à IP dans le cas d'un réseau sans fil : ERLN (voir figure **??**).

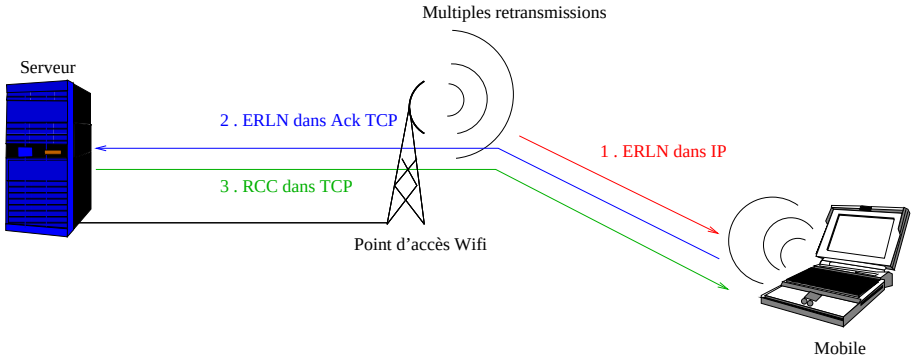


FIGURE 4.1 – Schéma d'enchaînement des messages avec ERLN

1. Comme pour ECN sur les routeurs, le point d'accès, en cas de retransmissions d'un paquet sans accusés de réception, modifierait un bit de l'en-tête IP afin d'informer le destinataire d'un problème d'interférences.
2. Cette information serait ensuite routée vers la source dans les accusés de réception TCP au moyen d'un bit *ERLN-E* (E pour *Experienced*) dans l'en-tête de ceux-ci.
3. La source appliquerait en cas de réception d'un message marqué ERLN-E au niveau des accusés TCP, l'algorithme de contrôle de congestion implémenté.
4. Pour prévenir la destination de la prise en compte du paquet marqué ERLN-E, la source modifierait un autre bit que celui précédemment utilisé pour ERLN, dans l'en-tête IP, afin de spécifier au récepteur d'arrêter d'envoyer des paquets marqués ERLN-E. Ce bit serait appelé *ERLN-C* (C pour *Confirmation*) et correspondrait au bit *CWD* dans ECN.
5. L'information comme quoi la source est revenue à un état normal après un laps de temps en phase adaptée aux retransmissions n'est pas obligatoire mais peut être signalé par un bit du même style que RCC au niveau de l'en-tête TCP.

Cette méthode d'information de la source semble mieux s'associer avec un contrôle de congestion associant une gestion temporaire des pertes au niveau Wi-Fi.

L'inconvénient de cette solution vient principalement de la supposition que le point d'accès est capable de lire et écrire au niveau des en-têtes IP. Le point d'accès n'est pas un équipement de haut niveau contrairement aux routeurs :il ne travaille qu'au niveau IP et pas au niveau Transport dans les couches du modèle ISO. Cette évolution du matériel, envisageable, peut être simulée dans un premier temps par l'utilisation d'une station de travail équipée de deux interfaces réseaux, une filaire et une sans fil, et un système de pont entre celles-ci.

4.3 Différenciation des délais

En raison du manque de fiabilité de la propagation sans fil, 802.11 permet des retransmissions MAC. On peut dire que 802.11 tente de transformer un réseau avec des pertes et un retard

prévisible en un réseau sans perte et avec un retard variable. Les pertes de paquet sont généralement dues à des interférences supposées provisoires, donc le RTT ne devrait pas être influencé par ces dernières.

Chaque interférence ou chaque sortie de la zone de couverture en réseau sans fil entraîne un certain nombre de retransmissions automatiques (un maximum est défini par défaut) au niveau de la couche liaison 802.11. Ces retransmissions exécutées pour des raisons techniques et non liées à un problème d'occupation du lien, entraîneront une augmentation du RTT. Cette augmentation du RTT peut être interprétée par la source, de manière erronée, comme un problème de congestion, et celle-ci diminuera sa fenêtre de congestion alors qu'il n'y a pas lieu de le faire. En connaissant le nombre de retransmissions effectuées au niveau de la couche liaison Wi-Fi, il devient possible de corriger le RTT en le diminuant du temps pris par les retransmissions.

Nous nous plaçons dans le cas où une transmission de données sur un réseau Wi-Fi subit un grand nombre d'interférences mais avec un nombre de paquets perdus peu important. On souhaite donc obtenir une régulation de l'émission des paquets par les pertes et non en fonction des délais occasionnés par les multiples retransmissions de chaque paquet. Nous proposons donc un mécanisme pour enlever le temps perdu par ces retransmissions de la couche MAC Wi-Fi.

4.3.1 Théorie

Notre principe s'applique pour tous les protocoles de transport qui ont un contrôle de congestion basé sur l'étude du RTT ou de sa variation au cours du temps.

Backoff

De manière plus importante que dans un réseau filaire classique, le tirage au sort du moment de transmission ou *backoff* entre chaque retransmission interfère sur le calcul du temps de retransmission moyen. Il faut donc tenir compte du nombre de retransmissions et du *backoff* moyen correspondant à chacune de ces retransmissions. Le *backoff* est tiré au hasard dans la fenêtre de collision (*Contention Window* CW) qui double à chaque retransmission. La fenêtre de départ est dans la majorité des cas comprise entre 0 et 31 ($2^5 - 1$ en fait) unités (*time slot* t_s) avec $t_s = 20$ μs dans le cas d'un réseau Wi-Fi 802.11b (voir explication détaillée dans [**?**]).

Prenons le cas d'un serveur désirant transmettre sur le réseau sans fil au moyen d'un point d'accès faisant la liaison entre les réseaux filaires et sans fil. Si pour la transmission d'un paquet le serveur obtient un RTT de 4,3 ms et qu'il sait qu'il y a eu 2 retransmissions, le RTT sans retransmission, RTT_r, peut donc être calculé :

- Le backoff moyen à la première transmission = $\frac{2^5-1}{2} = 15,5$ *time slots*
- Le backoff moyen à la première retransmission = $\frac{2^6-1}{2} = 31,5$ *time slots*
- Le backoff moyen à la deuxième retransmission = $\frac{2^7-1}{2} = 63,5$ *time slots*

D'où $RTT_r = 4,3 - t_s \times (31,5 + 63,5)$

$RTT_r = 2,4$ *ms*

Nous proposons la formule générique suivante valable uniquement dans le cas d'une présence d'un unique émetteur sur le réseau sans fil (avec n le nombre de retransmissions et t_s la durée d'un time slot en μs) :

$$RTT_r = RTT - \frac{t_s}{2} \times \left[\sum_{i=1}^{n} \left(2^{5+i} - 1 \right) \right]$$

Ou sans l'opérateur somme :

$$RTT_r = RTT - \frac{t_s}{2} \times \left(2^{6+n} - n - 33\right)$$

C'est grâce au calcul du RTT que la source sait comment adapter la taille de sa fenêtre de congestion. Avec le RTT normal, elle considérerait qu'il y a un problème de transmission ou de traitement de celle-ci et réduirait donc sa fenêtre de congestion alors qu'il n'y en a pas lieu. En utilisant le RTT_r, la source ne voit plus apparaître les temps de retransmission. Si RTT_r est grand par rapport à la normale, c'est qu'il y a un vrai problème autre que celui d'interférences causant des retransmissions. Dans le cas de retransmissions multiples à répétition, il se peut qu'un flux bloque les transmissions au niveau d'un point d'accès et comme le débit n'est pas réduit, on peut causer un blocage complet de tous les flux. La source pourrait alors passer en mode « normal », c'est-à-dire comme s'il y avait eu congestion.

Temps de transmission

Comme indiqué auparavant, chaque carte de réseau a un temporisateur. Chaque fois qu'un nouveau paquet est traité, une nouvelle variable est employée pour initialiser le temporisateur. Ceci permet de prendre en compte les pertes de temps cumulées, par exemple dans le cas d'un paquet d'ACK contenant déjà la valeur perdue de temps du paquet de données correspondant. Pendant chaque transmission, la valeur du temporisateur est stockée dans la variable à l'intérieur du paquet.

Le temporisateur peut-être considéré comme un simple chronomètre qui serait déclenché avant la première tentative d'envoi d'un paquet et arrêté au moment de la réception de l'accusé de récepteur de la couche MAC Wi-Fi ou au moment du dépassement du seuil des retransmissions. Dans ce dernier cas, il nous serait alors d'aucune utilité puisque le paquet est alors considéré comme perdu par le réseau.

Matériellement, la carte réseau sans fil doit savoir quand elle doit retransmettre un paquet. Pour cela à partir de la taille du paquet, de la bande passante négociée à la connexion et des caractéristiques de la transmission radio Wi-Fi (enchaînement des messages, longueur d'onde radio, etc), elle détermine l'instant précis de réception de l'accusé.

$$tempsperdu = tpspaquet + delai + SIFS + tpsack + delai$$

avec $tps_paquet = taille_paquet \times rate$ et $txack = acklg \times rate$

où *tpspaquet* est la durée de transmission d'un paquet de donnée et *tpsack* celle d'un accusé, $SIFS$ (*Short Interframe Space*) est le temps utilisé pour séparer les transmissions d'un même dialogue et *delai* correspond au délai de propagation sur le réseau sans fil utilisé.

S'il y a fragmentation au niveau de la couche MAC, cela n'influence en rien notre mécanisme. En effet, quand un paquet est réduit en fragments, la perte de temps est nulle si chaque fragment arrive à destination sans retransmission sinon le temps de retransmission total est égal à la somme des temps des fragments.

4.3.2 Comportement

Pour intégrer la variable dans laquelle le temps de retransmission est sauvegardé à l'intérieur d'un paquet, nous définissons une option supplémentaire `rets` dans l'en-tête IP d'un paquet. Notre option `rets` a seulement un champ, contenant une valeur de temps. Si le champ a 2 octets et l'unité de mesure est le temps en slot $t_s = 20\mu s$ du calcul du backoff 802.11, alors le champ débordera à la fois $t = 65536 \times t_s = 65536 \times 20\mu s \approx 1.3s$. Dans la norme 802.11b la fenêtre

maximum de congestion (CW) est de 1023 paquets, par conséquent une longueur de 2 octets est suffisante. Si le champ avait 4 octets, il débordera après $t \approx 65536 \times 1.3s \approx 1\ jour$, ce qui est très largement suffisant. La source place ce champ à zéro. Pendant le transfert du paquet, le champ peut être modifié par le point d'accès sans fil.

Ce mécanisme est déployable et utilisable de proche en proche, chaque communication au sein d'une chaîne pouvant se définir en tant que relation émetteur-récepteur comme par exemple dans le cas d'un réseau *ad-hoc*.

Dans le cas d'un réseau de type infrastructure, il donne des valeurs utiles seulement si l'expéditeur, le récepteur et le point d'accès reconnaissent cette option. Si l'expéditeur ou le client ne sont pas informés de cette option, elle n'est pas activée en raison de la négociation des options du protocole de transport. Sinon, la source ajoute l'option et place le champ à 0. Si le point d'accès et/ou le récepteur ne connaissent pas cette option, l'expéditeur ne reçoit aucune option en retour, ou une option avec valeur nulle, ce qui ne change rien non plus.

Dans le cas de TCP Vegas, on modifie le comportement de manière plus poussée. On a vu dans le chapitre **??** que TCP Vegas a un contrôle de congestion basé sur la différence entre un débit attendu et le débit actuel. Cette différence est ensuite comparée à deux seuils α et β . Nous ne changeons pas le principe de fonctionnement pour des valeurs de Δ supérieur à β et inférieur à α car notre modification du RTT influe déjà dans ces zones. Par contre dans la partie centrale sans « influence », on prend en compte la présence de retransmissions selon leur nombre en augmentant plus ou moins la fenêtre de congestion.

4.4 Étude de cas

4.4.1 TCP Vegas

Nous nous proposons d'étudier, au moyen du simulateur de réseau NS2, l'évolution du débit utilisé lors d'une transmission entre un client mobile et une station de travail fixe avec un point d'accès Wi-Fi comme intermédiaire. Dans cette étude, la modification supplémentaire que nous faisons à TCP Vegas en plus de notre proposition est : si notre flux subit des interférences et donc des retransmissions mais reste d'un point de vue TCP Vegas dans la zone stable (entre α et β), nous choisissons d'augmenter la fenêtre de congestion de $\frac{1}{cwnd}$ comme dans le cas où $\Delta < \alpha$.

Scénario simple

Nous prenons ici le cas d'un mobile unique, connecté à un point d'accès, lui-même connecté à un ordinateur fixe par un lien filaire avec une latence et une bande passante disponible suffisamment grands par rapport à celles du lien radio pour éviter toute influence de ce lien sur le transfert.

Nous effectuons une transmission de type FTP basée sur le protocole de transport TCP Vegas. Le transfert se fait avec un volume d'information transmise fixe. Le mobile effectue un déplacement programmé et minuté en trois phases comme présenté dans la figure **??** :

- Il reste immobile à une distance de couverture de la borne suffisante pour subir quelques retransmissions pendant une durée de 200 secondes,
- ensuite il se déplace à la vitesse de $1,7m.s^{-1}$ pour atteindre un emplacement prédéfini,
- il atteint sa destination en 200 secondes environ puis s'immobilise à cet endroit.

En suivant cette feuille de route, on obtient trois zones de couverture du réseau radio : une zone avec peu de perturbations, une zone de mouvement et de perturbation moyenne et une

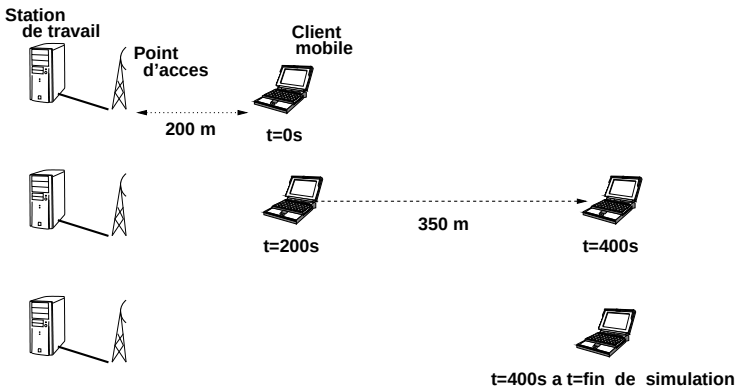


FIGURE 4.2 – Scénario de simulation.

dernière subissant de fortes interférences dues à la faible qualité du signal.

Nous ne nous sommes pas intéressé ici à un cas de *non-perturbation* de notre signal par des interférences, car notre solution n'est prévue que dans les cas où il y a interférences. La transmission sans interférence restant parfaite avec ou sans notre protocole de transport.

Les résultats présentés dans cette partie font suite à deux simulations successives. La première correspond au cas d'utilisation classique du protocole de transport TCP Vegas ; La seconde est l'exacte copie de la première du point de vue scénaristique, le seul changement étant les modifications proposées sur le contrôle de congestion de TCP Vegas.

Comme nos deux simulations ont un même volume transféré, nous les différencions et les comparons grâce au temps de transmission de chacune d'elle et donc de leurs débits respectifs.

Ce scénario de simulation a été renouvelé dix fois de suite en changeant à chaque exécution la valeur de la graine du générateur de nombres aléatoires.

Validation du cas simple

La première figure **??** représente l'évolution du débit de chacun de nos transferts au cours du temps. On remarque bien les trois phases de la simulation : la zone proche non perturbée (t = 0 à 200), la zone de mouvement (t=200 à 400) et enfin la zone en bordure de couverture (t=400 à 600). On observe que notre débit est la majeure partie du temps supérieur au débit du transfert obtenu avec TCP Vegas.

On remarque que grâce à l'augmentation du débit, notre protocole de transport accélère le transfert des données. Le tableau **??** présente les gains de temps obtenus. Un gain négatif correspond à du temps perdu. Ces résultats montrent une augmentation de la vitesse de transfert d'environ 7,15% (moyenne des 10 simulations exécutées).

La seconde figure **??** présente les résultats obtenus pour le débit dans chacune des simulations sous forme d'une différence. On observe dans ce graphique très clairement la supériorité de notre transmission quelque soit la situation géographique du client mobile par rapport à la borne d'accès sans fil. Le creux présent à la fin du graphique ($t > 560s$) est négatif mais ne signifie pas que le transfert classique est plus rapide : notre protocole de transport ayant accéléré le transfert, celui-ci est déjà fini. Ce creux indique simplement que la méthode classique est la seule en cours d'exécution durant ce laps de temps.

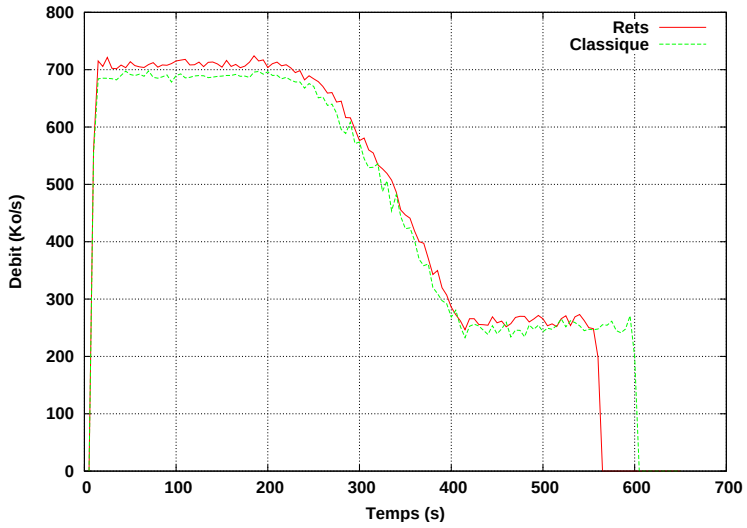


FIGURE 4.3 – Débit d'un transfert rets par rapport à un transfert classique.

Simulation (Valeur de la graine)	Gain en %	Gain en s
1	6.73	40.27
2	8.12	49.51
3	7.31	43.78
4	6.85	40.99
5	8.79	53.67
6	6.70	40.00
7	7.19	43.06
8	6.89	41.09
9	6.86	41.02
10	6.08	36.27

TABLE 4.1 – Gains obtenus pour différentes simulations.

Scénarios avec plusieurs clients

Pour valider notre première approche de la gestion de retransmissions, nous avons multiplié les clients mobiles pour un point d'accès. Nous avons testé pour deux puis trois et enfin quatre clients mobiles identiques.

Afin de ne pas multiplier les scénarios de déplacements, nous avons choisi un modèle de déplacement fixe que nous avons adapté en fonction du nombre de mobiles en mouvement. Dans le cas d'un mobile unique, nous n'avons déplacé ce client que dans un sens et de manière linéaire entre deux points. Dans le cas de plusieurs clients connectés à une même infrastructure, un déplacement identique n'est pas sérieux : nous avons donc décidé de déplacer nos mobiles entre deux points fixes (séparé par une distance L) avec un départ et une arrivée au même endroit. La position de départ de chaque client mobile ainsi que leur sens de déplacement sont définis afin de couvrir une distance globale parcourue de $2L$ à une vitesse fixe et identique pour tous les mobiles (voir figure **??**).

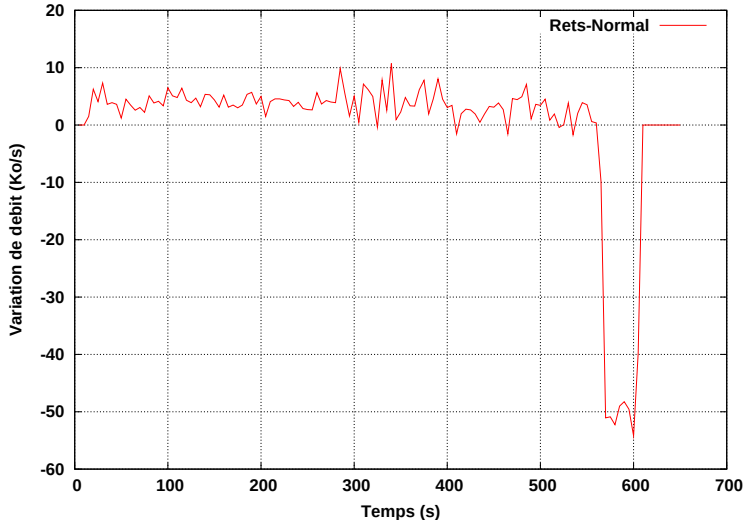


FIGURE 4.4 – Différence de débit entre un transfert rets et un classique.

Chaque mobile se déplace parallèlement aux autres et toutes les trajectoires sont centrées par rapport au point d'accès. Comme pour le scénario à un seul client, chaque mobile reçoit un quantité de données identiques. Ces transferts sont moins importants que dans le cas d'un unique client connecté afin de pallier la baisse de bande passante disponible due à l'augmentation des clients accédant au canal. La simulation dure comme précédemment, environ cinq minutes.

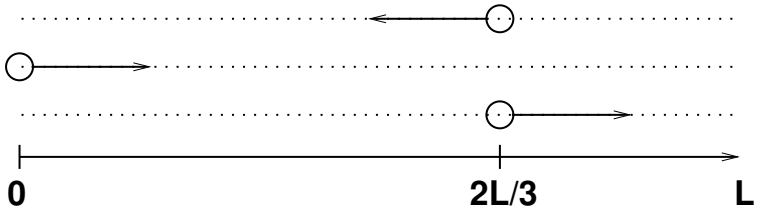


FIGURE 4.5 – Exemple de positionnement pour une simulation à trois clients.

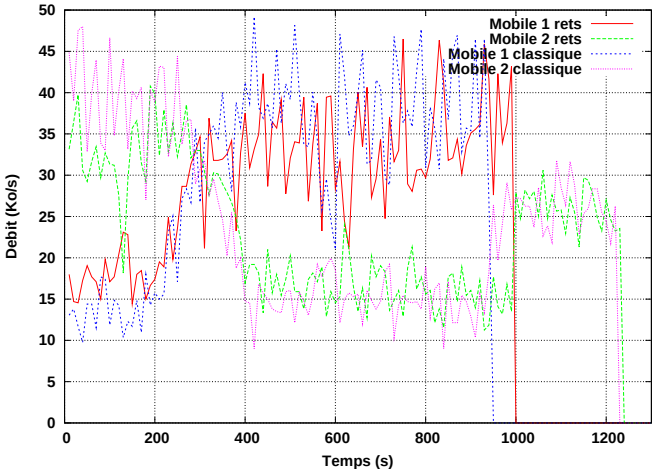
Résultats

La validation était parfaite pour un mobile unique connecté par un lien d'infrastructure. Dans le cas de multiple mobiles, il devient difficile de « faire gagner »tout le monde. Si l'on gagne de la bande passante et donc du temps dans le scénario de validation, cela est dû au fait que la bande passante disponible n'est pas totalement utilisée par un client mobile, il nous reste donc un peu de place disponible.

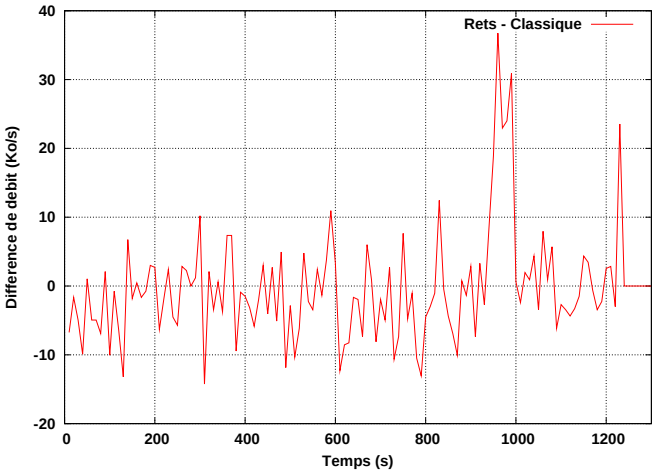
Dans un cas plus réaliste avec plusieurs mobiles connectés à un même point d'accès, la marge disponible est plus faible. On vérifie très bien en matière de bande passante réseau le principe qui veut que la bande passante totale utilisée sur un lien soit fonction du nombre de connexions sur ce lien : plus on a de connexions, plus le lien est utilisé dans un réseau sans fil. Ce principe est mis en défaut à cause de la méthode d'accès au canal : plus il y a de monde, plus un client mobile passera de temps à attendre et plus il y aura de collisions lors de communications sans fil. Ces temps d'attente diminuent fortement le débit d'un client et donc la bande passante globale utilisée.

Les figures **??**, **??** et **??** présentent toutes des résultats cohérents mais à différentes échelles de réseaux.

Les résultats obtenus diffèrent quelque peu des attentes, basés sur les gains observés dans le cas d'un mobile simple, mais restent cohérents et explicables.



(a) Débits des transferts avec rets par rapport à des transferts classiques.



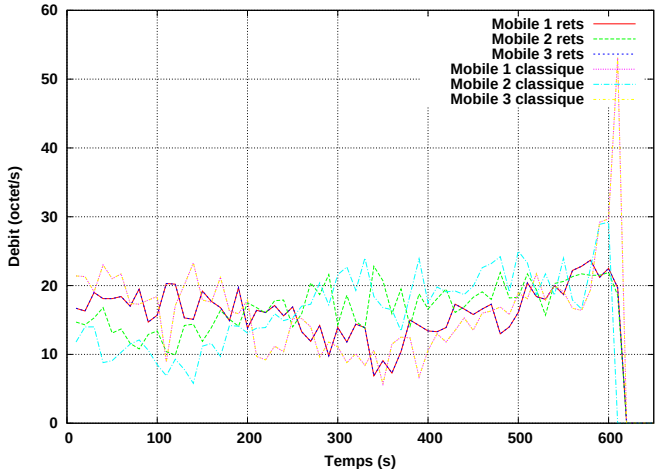
(b) Différence de débit entre les transferts rets et les classiques.

FIGURE 4.6 – Scénario avec deux mobiles connectés.

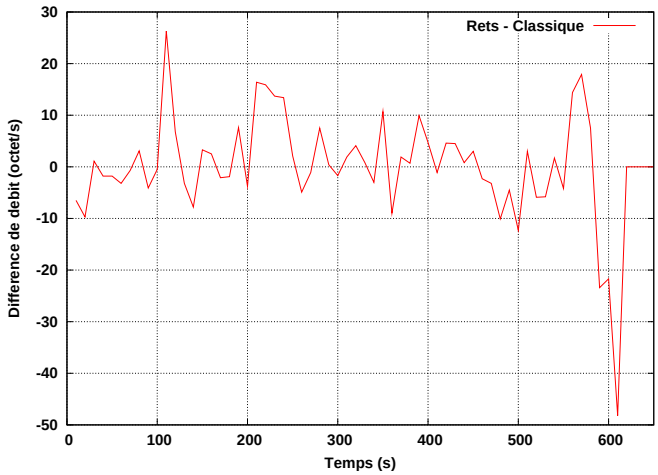
Comme nous avons pu le constater dans l'exemple étudié précédemment, on n'obtient aucun gain s'il n'y a pas de retransmissions : observation logique de par la conception de l'algorithme. Dans le cas de multiples clients connectés à un point d'accès, certains se trouvent statistiquement proches de l'équipement d'infrastructure et ne peuvent donc bénéficier de notre amélioration.

Par contre, ceux qui subissent des interférences et donc des retransmissions, acquièrent cette possibilité de gain de bande passante. Mais comme la bande passante n'est pas infinie, ce qui est gagné est forcément pris quelque part. La bande passante des mobiles proches devient la source des gains de bande passante pour des mobiles défavorisés. On évite ainsi le principe de la double peine appliquée aux mobiles éloignés : la baisse de débit et les nombreuses interférences.

D'un côté, l'augmentation des collisions, dues aux gains de bande passante de certains clients



(a) Débits des transferts avec rets par rapport à des transferts classiques.



(b) Différence de débit entre les transferts rets et les classiques.

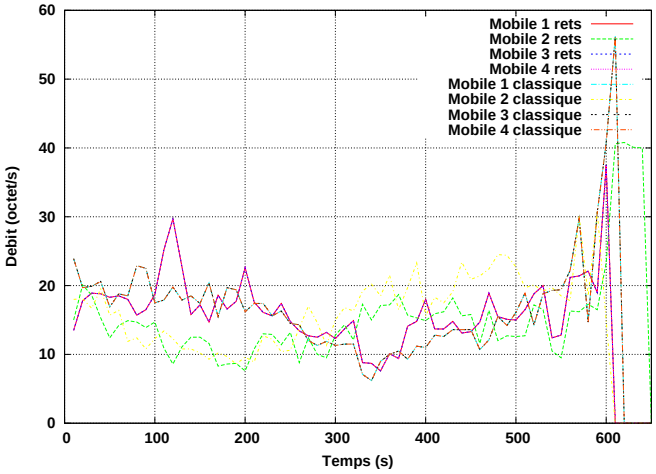
FIGURE 4.7 – Scénario avec trois mobiles connectés.

mobiles, diminuent légèrement la bande passante globale. D'un autre, les clients plus faibles (en terme de bande passante accessible) sont aidées avec un rendement supérieur à la perte des plus forts. En conclusion, l'ensemble est plus équilibré mais peut être légèrement déficitaire par rapport à l'environnement de transmission classique. L'utilisation de la bande passante disponible est beaucoup plus équitable.

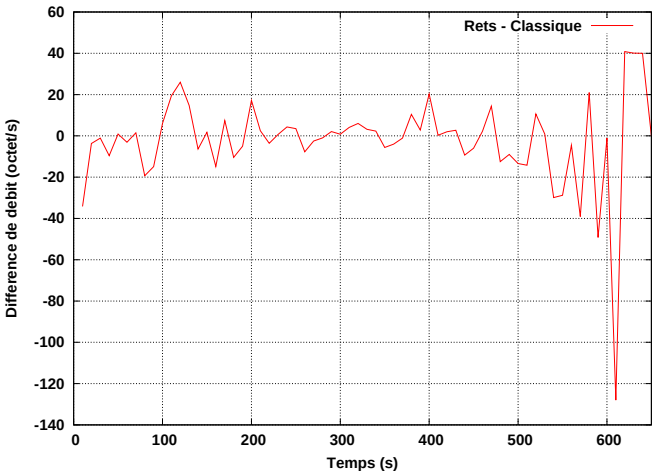
4.4.2 DCCP/TFRC-LD

En plus de proposer un évolution de TCP Vegas, nous avons, dans le cadre de l'étude d'un nouveau protocole de transport, choisi de nous intéresser à DCCP. Notre solution est appelée le DCCP/TFRC-LD (*TFRC Loss Differenciation*). Une option, appelée `rets` comme pour TCP Vegas, est ajoutée l'en-tête DCCP.

Contrairement à ce qui a été proposé pour TCP Vegas, dans DCCP/TFRC-LD, il appartient au récepteur de prendre des mesures appropriées pour le contrôle de congestion. Par exemple,



(a) Débits des transferts avec rets par rapport à des transferts classiques.



(b) Différence de débit entre les transferts rets et les classiques.

FIGURE 4.8 – Scénario avec quatre mobiles connectés.

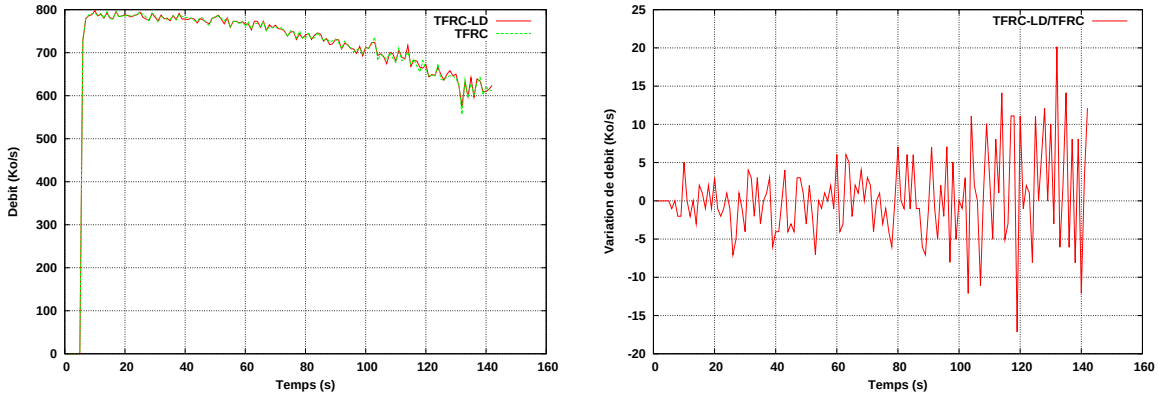
nous proposons que le récepteur DCCP/TFRC-LD utilise le RTT corrigé dans sa formule d'estimation du RTT. Grâce à cette évaluation l'expéditeur emploiera un débit d'émission plus adapté.

Les simulations diffèrent de celles utilisées dans des connexions TCP Vegas vu précédemment. La modification est nécessaire de par la conception du simulateur de réseau utilisé, en effet celui-ci ne gère pas correctement les connexions de type DCCP multiples ainsi que les échanges de données dans le cas de réseaux sans fil.

DCCP ne faisant pas de retransmission, on ne peut pas se baser sur le temps de transfert pour constater un éventuel gain de performance. Il faut donc s'intéresser à la quantité de données reçues ainsi qu'aux pertes de paquets obtenues durant le transfert pour qualifier notre proposition.

La courbe du débit pour un transfert DCCP-LD (voir figure **??**) est similaire à celle d'un transfert DCCP classique. On n'observe pas directement d'amélioration. La variation de différence entre ces deux débits, présenté dans le graphique **??**, montre quand à elle une légère

augmentation du débit dans le cas de transfert perturbé.



(a) Débits de transferts avec rets par rapport à des transferts classiques.

(b) Différence de débit entre un transfert rets et un classique.

FIGURE 4.9 – Scénario DCCP avec un mobile.

Le tableau **??** présente les différents résultats obtenus pour un ensemble de simulations. comme le temps de transfert n'est pas une valeur sûre de mesure de performances, nous étudions les valeurs caractéristiques d'un transfert de fichiers : le nombre de paquets reçus et perdus. Comme les temps de transfert subissent un écart inférieur à 0.5% on peut considérer cette différence négligeable dans ce cas. Pour pouvoir comparer efficacement les données obtenues, nous définissons un ratio paquets perdus sur paquets reçus à la destination ceci afin de prendre en compte les différences entre les quantités de données arrivées à destination.

Paquets reçus rets	Paquets reçus norm	Perte Rets	Pertes Norm	Temps Rets	Temps Norm	Ratio Rets Reçus/Pertes	Ratio Norm Reçus/Pertes	Gain
99634	99568	369	435	142.49	142.48	.3703 %	.4368 %	15.22 %
99596	99610	407	394	142.39	142.50	.4086 %	.3955 %	-3.31 %
99606	99606	398	397	142.50	142.44	.3995 %	.3985 %	-.25 %
99637	99600	366	403	142.40	142.50	.3673 %	.4046 %	9.21 %
99634	99554	369	450	142.66	142.37	.3703 %	.4520 %	18.07 %
99575	99559	428	444	142.45	142.34	.4298 %	.4459 %	3.61 %
99597	99585	406	418	142.50	142.53	.4076 %	.4197 %	2.88 %
99611	99587	392	416	142.50	142.54	.3935 %	.4177 %	5.79 %
99640	99630	363	374	142.71	142.50	.3643 %	.3753 %	2.93 %
99618	99566	385	437	142.40	142.42	.3864 %	.4389 %	11.96 %

TABLE 4.2 – Gains obtenus pour différentes simulations DCCP.

On peut remarquer que notre ratio est dans la majorité des cas, meilleur que celui obtenu dans le cas d'un transfert DCCP classique. On ne va pas forcément plus vite dans nos transferts mais le nombre de pertes que l'on subit par rapport au temps de transmission est plus faible. Suivant ces résultats, on peut dire que l'on a une meilleure utilisation de la bande passante disponible.

4.5 Conclusion

Ce chapitre présente une méthode pour supprimer l'effet des retransmissions de la couche Mac, dans un réseau sans fil de type Wi-Fi, sur la valeur du RTT. Il définit une option d'un protocole de transport contenant une valeur temporelle. Cette valeur est augmentée par les cartes réseau sans fil chaque fois qu'un paquet est retransmis par la couche Mac. Le champ d'information se propage à la source, comme l'option TCP timestamp [**?**]. Cela permet à l'expéditeur de connaître exactement le temps passé dans les retransmissions.

Les simulations ont montré une amélioration de débit avec notre méthode dans le cadre de tests simples et ce quelque soit le protocole de transport testé (TCP Vegas ou DCCP/TFRC). Nous prévoyons de faire des expériences réelles, afin de valider nos simulations. En cas de succès, nous envisageons de faire une proposition complète pour l'amélioration d'un protocole de transport dans le cas de retransmissions faites sur les connexions sans fil.

Troisième partie

Contrôle de congestion vidéo

CHAPITRE 5

Etat de l'art

Sommaire

5.1 Introduction

TCP a transporté avec succès des données de tout type de manière fiable et ordonnée. Avec l'augmentation de bande passante sur le réseaux ces dernières années, de nombreuses applications multimédia proposant des flux vidéo et audio sur Internet ont vu le jour. La forte demande de ce type de support a entraîné une augmentation de la présence de ces données multimédia. Les protocoles de transport doivent donc être pensé afin que la bande passante disponible sur le réseau soit partagée entre les transferts multimédia et les autres types de connexion comme TCP, UDP etc.

Les données multimédia ne sont généralement pas transmises en utilisant le protocole TCP parce que les services fournis par ce dernier sont adaptés pour les applications nécessitant le transfert fiable de données, comme la navigation web, les e-mail et les transferts FTP. Ces dernières applications accordent une plus grande priorité à la transmission fiable, plutôt qu'au temps de transfert des données. En revanche, la diffusion vidéo demande une latence de transfert plus stricte.

La popularité croissante de la diffusion vidéo est une cause d'inquiétude pour la stabilité de l'Internet parce que la plupart des contenus vidéo est actuellement proposée sur UDP, sans aucun contrôle de congestion. Il existe donc un intérêt grandissant à la mise en place d'un système de contrôle de congestion à la fois équitable pour TCP et efficace dans la mise à disposition de flux vidéo temps réel.

5.2 Caractéristiques d'un flux vidéo

Un flux vidéo est un flux multimédia (images et sons) temps réel. Toute transmission multimédia a des propriétés bien spécifiques. Comme pour la compression, si lors de la transmission de textes ou d'images une partie des données se voit modifiée, le résultat risque d'être inutilisable. Par contre le rafraîchissement constant d'une vidéo permet de ne pas se contraindre à fournir un contrôle de validité strict à tous les instants. Lors de transmissions traditionnelles, la durée et le débit peuvent varier sans pour autant affecter le résultat final, tandis que dans le cas du

multimédia, le débit doit être suffisamment élevé et régulier pour supporter le flux continu et fournir assez d'informations de manière assez régulière pour afficher les images et diffuser le son d'une façon fluide et non hachée.

Transporter un flux de ce type ne signifie pas obligatoirement le délivrer le plus rapidement possible mais le délivrer le plus régulièrement possible à l'application réceptrice. Certains paramètres du système de communication impactent sur la qualité perçue par l'utilisateur final :

- Le délai : ce paramètre est défini en tant que temps pris pour établir un service particulier de la demande initiale de l'utilisateur à l'instant de réception de l'information spécifique une fois le service établi. Le délai a un impact très direct sur la satisfaction d'utilisateur selon l'application, et inclut du délai dans le système destinataire et les systèmes intermédiaires.
- La perte d'information : ce paramètre a un effet très direct sur la qualité de l'information finale présentée à l'utilisateur. Dans ce contexte, la perte de l'information n'est pas limitée aux effets des erreurs de bit ou de la perte de paquet pendant la transmission, mais inclut également les effets de toutes dégradations introduites volontairement par le codage des médias pour une transmission plus efficace.

Du point de vue utilisateur, d'autres paramètres de communication entrent en compte, citons :

- La gigue : la variation de délai.
- Le débit : la quantité d'informations transmise via un canal de communication selon un intervalle de temps donné. Le débit est principalement exprimé en octets/seconde (o/s).
- La synchronisation multimédia : c'est ce qui correspond à la synchronisation du flux audio avec le flux vidéo.

Pour un flux vidéo la régularité de réception implique un contrôle de flux avec un envoi plus ou moins fiable des données (gestion des pertes et erreurs). Un flux vidéo est aussi un flux limité temporellement et cette durée de transmission peut être importante : dans le cas de la Video on Demand (VoD) par exemple la transmission d'un film peut durer plus de deux heures. Il faut donc gérer l'occupation du réseau par ce flux multimédia afin qu'il ne pénalise pas les autres applications et que d'un autre côté, elles n'influencent pas la transmission du flux multimédia (altération des paquets ou concurrence au sein d'une file d'attente).

La transmission de vidéos requiert une connexion rapide avec un protocole offrant une qualité de service (QoS) de bout en bout aussi proche que possible du temps réel et une gigue assez faible, ou dans le cas de diffusions à débit variable une QoS assez prévisible pour avoir l'assurance de transmettre le flux désiré dans le temps voulu.

On voit bien que ces spécificités ne correspondent pas aux protocoles utilisés habituellement sur Internet. C'est pourquoi des protocoles répondant aux besoins des flux multimédias temps réel ont été développés. Deux choix s'offrent alors pour la création d'un protocole spécifique à la transmission vidéo :

1. ajouter à un protocole temps réel, donc non-fiable, un peu de fiabilité par l'intermédiaire d'un contrôle de flux,
2. enlever à un protocole fiable, un peu de cette fiabilité, afin de prendre ne compte la propriété temps réel.

Les protocoles de transport fiable ne sont pas adaptés au transfert de flux vidéo car ils privilégient la qualité au profit du délai. En effet, la transmission fiable n'est pas adaptée au flux continu en cas de congestion et donc de retransmission des paquets perdus. Ces retransmissions assurant la fiabilité se font au détriment d'une des caractéristiques principales de la vidéo en *streaming* : le temps réel.

5.3 Plus de fiabilité

5.3.1 Le protocole non-fiable par excellence, UDP

UDP défini dans la RFC768 [**?**] est un protocole de transport non fiable. À la différence de TCP, il n'est pas orienté connexion et ne fournit pas de contrôle des pertes de paquets. C'est le protocole le plus simple pour le transport de données : la communication est « unidirectionnelle », c'est-à-dire que les données sont envoyées par une source à un destinataire mais il n'y a pas de retour d'information de la part de ce dernier (pas d'acquittement ni d'information sur les paquets perdus). Un grand nombre d'applications manipulant des flux continus multimédia se sont mises à utiliser des protocoles, non fiables, comme UDP plus adaptés à la manipulation de tels flux.

Cependant, les services fournis par UDP ne répondent pas vraiment aux besoins des applications temps-réel. Pour pallier certains manques, UDP a été détrôné au profit de protocoles comme RTP (voir section **??**) qui utilise directement UDP depuis une couche supérieure ou DCCP (voir section **??**) qui est une amélioration de UDP mais de même niveau, que nous verrons dans les sections suivantes. En effet, comme celui-ci est dépourvu de tout contrôle, on peut facilement créer un nouveau protocole par ajout d'éléments à UDP.

5.3.2 et sa sur-couche vidéo RTP

Le protocole RTP est décrit dans les documents RFC 3550 [**?**]. C'est un protocole de niveau applicatif et est donc indépendant des couches inférieures. En effet, il peut être utilisé soit directement au dessus de IP (Internet Protocol), soit, comme c'est généralement le cas, au dessus du protocole UDP.

Ce protocole de transport a été conçu spécialement pour la diffusion continue de données sur les réseaux. C'est la norme de diffusion en continu la plus importante. Tous les flux médias, quelque soit leur format et leur contenu, sont encapsulés dans des paquets RTP. L'en-tête RTP offre plusieurs champs qui ne sont pas présents dans l'en-tête UDP, notamment un horodateur et un numéro de séquence. Le protocole RTP s'exécute sur UDP et en utilise les fonctions de multiplexage et de somme de contrôle. Il permet de contrôler le serveur afin que le flux vidéo soit transmis à la bonne vitesse. Le destinataire est alors capable de réassembler les paquets RTP reçus dans le bon ordre et de les lire à la vitesse voulue.

Le protocole RTP transmet les paquets en temps réel, ne gère pas les pertes et les délais mais réalise l'ordonnancement des paquets. Les paquets perdus ou endommagés ne sont pas retransmis. Ces aptitudes temps réel se font cependant au détriment de beaucoup de paramètres tout aussi importants comme nous avons pu le voir ou comme nous le verrons :

- pas de qualité de service
- pas de fiabilité de transport
- pas de contrôle de congestion
- aucune priorité du flux en cas de concurrence avec TCP par exemple en cas de rejet de paquets (au niveau des routeurs)

RTCP : Real-Time Control Protocol

Le protocole RTCP est associé au protocole RTP et utilise le protocole UDP pour les connexions entre le client et le serveur. Il envoie au fournisseur de services les informations de retour sur la qualité de réception transmises par chaque participant à une session RTP. Ces messages contiennent des rapports sur le nombre de paquets perdus et des statistiques sur la

gigue (arrivées précoces ou tardives). Ces informations peuvent être utilisées par des applications de niveau supérieur pour contrôler la session et améliorer la transmission ; par exemple, le débit d'un flux peut être modifié pour faire face à l'encombrement d'un réseau.

Messages RTCP

RTCP est basé sur la transmission périodique des paquets de contrôle à tous les participants à la session, et sa fonction principale est d'obtenir des rapports de réception des flux vidéos. Il y a cinq types de messages RTCP :

- SR (Sender Report) : Il contient des informations concernant les données envoyées par cette source, et des statistiques sur la réception des flux envoyés par les autres sources dans le cas d'une session à plusieurs sources.
- RR (Receiver Report) : Il indique son état de réception (sa bande passante disponible et son taux de perte). Dans le cas d'une session à plusieurs sources, ce rapport peut contenir des statistiques sur la réception d'au plus 31 sources.
- SDES (Source description RTCP packet) : Chaque élément de la session (source ou récepteur) transmet des informations textuelles appelées *canonical names* comme des identificateurs uniques pour les participants de la session dans un paquet SDES. Les paquets RTCP SDES contiennent des informations textuelles comme identificateurs uniques pour les participants de la session. Ces identificateurs peuvent inclure le nom des utilisateurs, leur numéro de téléphone, adresse e-mail et autres informations.
  Dans le cas où la source émet plusieurs types de données, par exemple, un flux audio et un flux vidéo, un même paquet ne peut pas transporter ces différents types. Dans ce cas, RTP affecte à la session des identificateurs SSRC pour chaque application, à condition que le SSRC de chaque élément (source ou récepteur) soit nouveau pour chaque application.
- BYE : Chaque élément de la session doit envoyer ce paquet pour indiquer la fin de sa contribution à la session.
- APP : Dans le cas des applications spécifiques, les données peuvent être transmises dans des paquets spécifiques de type APP. Ce paquet consiste en un en-tête et un champ de données qui est fonction de l'application considérée.

Services RTCP

En plus de la génération de ces 5 types de paquets, RTCP offre les services de :

- Contrôle de la congestion : Ceci constitue la fonction primordiale de RTCP. RTCP fournit un compte-rendu à l'application au sujet de la qualité de distribution des données. L'information de contrôle est utile aux émetteurs et aux récepteurs. L'émetteur peut ajuster sa transmission en se basant sur le rapport du récepteur. Les récepteurs peuvent déterminer si une congestion est locale, régionale ou globale.
- Identification de la source : Dans des paquets de données RTP, les sources sont identifiées par des identificateurs de 32 bits générés de manière aléatoire.
- Synchronisation inter-média : Les rapports des émetteurs RTCP contiennent une indication de temps réel et l'estampille de temps RTP correspondante. Ceci peut être utilisé pour la synchronisation inter-média (vidéo).
- Information de contrôle : Les paquets RTCP sont envoyés périodiquement parmi les participants. Quand le nombre de participants augmente (ce qui peut être le cas dans une session multicast), il est nécessaire de limiter les informations de contrôle pour éviter que le trafic de contrôle ne vienne saturer les ressources du réseau. RTCP limite le trafic de contrôle à 5% du trafic total de la session.

RTSP : Real-Time Session Protocol

Le protocole de transfert de flux continu en temps réel RTSP est un protocole de niveau application qui sert au contrôle des données diffusées avec RTP. Il se base sur le protocole RTP comme protocole de distribution de données et offre un contrôle pratique de la diffusion du flux vidéo à l'utilisateur : lecture, arrêt, pause, avance rapide et rembobinage rapide, ainsi que l'accès à n'importe quelle partie de la vidéo.

Le protocole RTSP permet au serveur de réguler son débit à la bande passante disponible sur le réseau. Pour se faire, RTSP maintient une horloge qui est associée à chaque paquet transmis. Cette horloge permet de déterminer, à chaque instant, si une portion de donnée à transmettre a déjà dépassé son échéance. Dans ce cas, il est inutile de l'envoyer au client qui la rejetterait à sa réception du fait de son retard. On évite ainsi un encombrement du réseau avec des données inutilisables

RTSP peut choisir le canal de distribution optimal pour le client. Par exemple, si le protocole UDP ne peut pas être utilisé (certains pare-feu privés ne l'acceptent pas), le serveur de diffusion en continu doit offrir un choix de protocoles de distribution, UDP multi-destinataire ou TCP, pour s'adapter aux différents clients.

Le protocole RTSP est similaire au protocole HTTP en termes de syntaxe et de fonctionnement, mais il en diffère sur plusieurs aspects importants. Avec le protocole RTSP, le client et le serveur peuvent envoyer des demandes pendant les interactions, contrairement à HTTP dans lequel c'est toujours le client qui émet les demandes.

5.3.3 VRP : Variable Reliability Protocol

VRP est un protocole de niveau application s'appuyant sur UDP et développé dans [**?**]. Il est intéressant parce qu'il y a une *spécification des pertes* autorisées par le destinataire. Le protocole UDP ne sert qu'à l'envoi des données sur le réseau, VRP gère au niveau supérieur le découpage des données sous forme de datagrammes. Les données sont séparées en *frames* puis découpées sous forme de datagrammes UDP de taille fixe. Ceux-ci sont envoyés en ordre à la couche UDP par une fenêtre de transmission qui définit le nombre maximal de paquets envoyés sur le réseau (appelée *Sliding Window*).

La détection de perte n'est gérée qu'au niveau récepteur par un système d'accusés de réception. Le récepteur peut accuser un paquet perdu si c'est une perte acceptable et sinon il renvoie un NACK (Negative ACKnowledgment) afin de demander une retransmission de ce paquet. L'application réceptrice spécifie deux types de contraintes sur les pertes :

- le pourcentage de pertes tolérable dans la fenêtre de réception,
- le nombre maximum autorisé de paquets perdus consécutivement dans cette fenêtre.

Pour gérer ces pertes, il existe deux fenêtres au niveau récepteur (voir figure **??**) :

- la première est une fenêtre de réception basée sur le même principe que la fenêtre côté émetteur, *reliability window*,
- la seconde est une fenêtre *history* permettant de garder un historique des paquets perdus afin que l'application vérifie si les conditions de perte sont vérifiées.

5.4 Moins de fiabilité et plus de temps réel

Afin de profiter d'un minimum de fiabilité dans un transfert de données continues, beaucoup se sont penchés sur l'amélioration de l'existant : ajouter une dimension temps-réel à des protocoles de transport de fichiers.

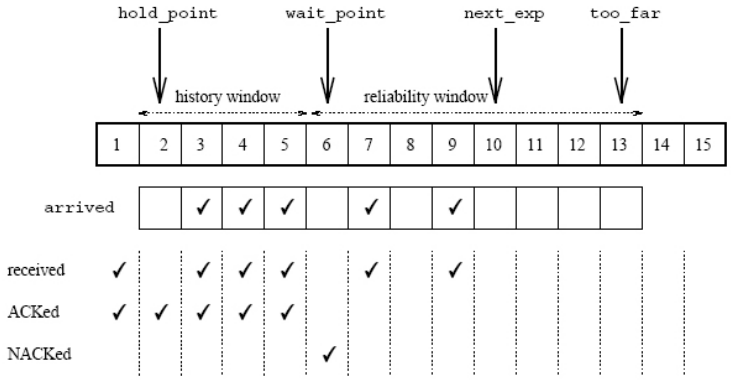


FIGURE 5.1 – Fenêtres au niveau récepteur

5.4.1 TCP pour la vidéo

Transmission Control Protocol - Continuous Media ou TCP-CM [**?**] est une modification de TCP pour les applications de médias continus sur Internet. C'est un protocole de même niveau que TCP, il appartient à la couche transport.

Il n'y a pas de retransmission des paquets perdus, on donne la priorité au flux en cours. La source envoie des données en continu à l'application cliente. Dans TCP pour pouvoir transmettre des paquets, la fenêtre de congestion ou d'émission doit être non pleine. Mais en cas de congestion, celle-ci reste pleine et la transmission est stoppée. Avec TCP-CM l'émetteur peut gérer sa fenêtre de congestion et peut choisir une méthode d'arrangement de sa fenêtre de congestion :

- First In First Out (FIFO) : les nouveaux paquets remplacent les plus anciens et sont mis en queue ;
- Last In First Out (LIFO) : les nouveaux paquets remplacent les plus anciens et sont mis en tête.

De plus l'émetteur, pour conserver une caractéristique de flux continu, ne retransmet pas les paquets perdus mais envoie à la place des paquets factices (*dummy packets*) afin de « boucher les trous »du côté client. En effet, ce protocole est basé sur TCP et ce dernier retransmet normalement tous les paquets non acquittés. Ainsi, on obtient une version de TCP sans retransmission de données en effet les paquets factices retransmis ne contiennent aucune information hormis leur numéro de séquence.

Cette gestion suppose un groupage des données par paquets au niveau application. Mais TCP gère les données par octet, TCP-CM a donc été modifié pour avoir une gestion par paquet comme UDP et non par octet comme TCP, il supporte donc l'ALF (Application Level Framing). Il n'y a qu'un APDUs (Application Protocol Data Unit) à l'intérieur d'un paquet envoyé.

5.4.2 PR-SCTP : Partial Reliability - Stream Control Transfert Protocol

Stream Control Transfert Protocol (SCTP) a été développé originellement dans l'optique de remplacer le système de signalisation numéro 7 (SS7) [**?**] utilisé dans les télécommunications. Puis il a évolué vers un protocole de la couche transport. Le protocole SCTP, défini dans la RFC 4960 [**?**], intègre plusieurs options lui permettant d'acquérir des particularités à la fois des protocoles TCP et UDP. Tout comme TCP, il permet de transférer de manière fiable des données sur le réseau tout en ayant la possibilité de les délivrer de dans le désordre à l'application comme UDP.

L'implémentation d'une forme de *fiabilité partielle* dans SCTP est décrite dans le RFC 3758 [**?**]. Cette extension de SCTP permet à la source de définir une fiabilité différente sur des messages différents. La source peut ainsi choisir quelle politique de retransmission adopter pour chaque message. Par contre cette politique est propre au serveur et est donc inconnue des clients. La seule politique spécifiée aujourd'hui est une politique de fiabilité temporelle. Mais d'autres politiques peuvent être envisagées.

La fiabilité temporelle permet de spécifier la durée de vie d'un paquet. Dès que cette durée de vie est expirée, le serveur stoppe les retransmissions et oublie le paquet en indiquant au client d'avancer le moment de son prochain accusé de réception cumulatif. Le fait d'avancer ce point permet de considérer le paquet en fin de vie comme reçu et accusé.

Dans [**?**] est proposée une application de ce principe au transfert d'un flux vidéo MPEG : une fiabilité temporelle différente est appliquée aux images I et P. En effet, une image I est plus importante mais a une date limite d'utilisation au-delà de laquelle elle n'a plus aucune utilité. En se basant sur ce principe, il est possible de définir une fiabilité temporelle pour chaque image correctement mise en paquet au niveau application.

5.4.3 VTP : Video Transport Protocol

Dans ce papier [**?**], les auteurs conçoivent et mettent en œuvre un protocole visant à maximiser la qualité de la diffusion en temps réel d'un flux vidéo MPEG-4 tout en proposant un contrôle de congestion de bout en bout. Ce qui différencie le protocole Video Transport Protocol (VTP) est l'utilisation de l'estimation de la bande passante au niveau du récepteur.

VTP implémente un unique mécanisme de contrôle de congestion pour obtenir un débit régulier et équitable : il suit en permanence le débit de réception *Achieved Rate* (AR). La communication entre émetteur et récepteur se fait au moyen de messages d'acquittements réguliers.

AR est une mesure juste pour définir le débit d'envoi et l'encodage de la vidéo (vidéo pré-encodée).

$$débit\ d'envoi\ =\ débit\ recu$$

Cette mesure évite la forte réduction du débit en cas de congestion pour TCP. Ceci bloque temporairement l'augmentation du débit jusqu'à ce que TCP revienne à un débit normal.

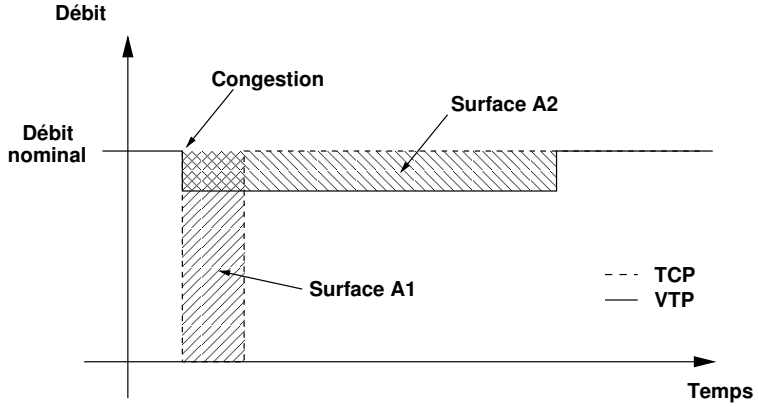


FIGURE 5.2 – Équité TCP / VTP

La figure **??** illustre le principe d'équité du contrôle de congestion de VTP avec TCP. Alors que TCP casse radicalement son débit (phase de *SlowStart*) : VTP diminue plus légèrement le

sien mais reste dans cet état plus longtemps. La propriété TCP-Friendly est vérifiée si l'égalité des surfaces est conservée $A1 = A2$.

De plus VTP inclut un système de différenciation des pertes afin de reconnaître les zones de réelle congestion des pertes dues au réseau sans fil.

Les résultats montrent que VTP offre une qualité vidéo acceptable dans les réseaux peu congestionnés et partage de manière équitable la bande passante avec des flux TCP (sous certaines conditions de congestion).

5.5 Un protocole modulaire : DCCP

UDP est un protocole adapté pour la transmission de flux temps-réel mais sans aucun système de contrôle. *Datagram Congestion Control Protocol* (DCCP) est un protocole de la couche transport apportant à UDP le contrôle de congestion et de flux. Il possède les propriétés suivantes :

- une transmission non fiable de datagrammes avec des acquittements ;
- des mécanismes fiables d'établissement et de libération de connexion ;
- un mécanisme de négociation d'options fiable qui inclut la négociation d'un protocole de contrôle de congestion ;
- un mécanisme permettant à un serveur de ne pas stocker d'information concernant des connexions non acquittées ou déjà terminées ;
- un mécanisme optionnel qui permet à l'émetteur de savoir avec précision quels sont les paquets qui ont atteint le récepteur ;
- le mécanisme de contrôle de congestion incorpore l'ECN ;
- la découverte du chemin possédant la plus grande *Maximum Transmission Unit* MTU.

DCCP s'adresse aux applications qui ne souhaitent pas une transmission fiable et ordonné des paquets (par exemple celles de *streaming* multimédia). Une connexion DCCP contient un flux de contrôle (informe l'émetteur si le paquet est arrivé et s'il a eu un marquage ECN) et de données.

Une particularité de DCCP est de proposer plusieurs mécanismes de contrôle de congestion : un premier basé sur une division par deux de la fenêtre de congestion lorsqu'il y a une perte nommé *TCP-Like* (son système de contrôle de congestion est identique à celui de TCP), un second basé sur *TFRC* qui minimise les changements abrupts dans le débit d'émission. D'autres contrôle de congestion sont intégrables comme nous pourrons le voir. TCP-Like étant identique à TCP, nous ne détaillerons pas son fonctionnement.

Dans DCCP il est possible de différencier la partie montante de la partie descendante de la connexion. On parle alors de demi-connexion (ceci consiste en l'ensemble des paquets envoyés de A vers B et l'ensemble des acquittements envoyés de B vers A), ainsi une connexion est un ensemble de deux demi-connexions. Chaque demi-connexion peut adopter un mécanisme de contrôle de congestion spécifique.

5.5.1 Contrôle de congestion TFRC

Le mécanisme *TCP Friendly Rate Control*, TFRF, a été proposé initialement dans [**?**]. Il s'agit d'un nouveau mécanisme de contrôle de congestion qui repose sur le calcul du taux d'événements des pertes de paquet. Il est *TCP-Friendly* et permettra aux applications de *streaming* multimédia de disposer d'un taux de transfert moins irrégulier que celui actuellement disponible avec les mise en œuvre courantes de TCP. En effet, pour TCP l'occurrence d'une perte (expiration d'un *timeout* par exemple) entraîne la division par deux de la fenêtre de congestion.

TFRC, au contraire calcule régulièrement, une fois par RTT, par l'intermédiaire du récepteur, le taux de pertes et en informe la source. Cette dernière peut ainsi adapter son débit en fonction de l'évolution de l'état du réseau. Il en résulte que l'évolution du débit proposé par TFRC est plus régulière. Il espace aussi régulièrement l'envoi des paquets depuis l'émetteur. La bande passante disponible est calculé par TFRC en utilisant l'équation suivante :

$$R_{TFRC} \cong \frac{s}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)}$$

où R_{TFRC} représente le débit du flux en cours de transmission, s la taille d'un paquet, p le taux de perte, t_{RTO} le *timeout* et b le nombre de paquets accusé par un accusé de réception.

Augmentation de la qualité de service

Le mécanisme de contrôle de congestion de TFRC est utilisé par les applications qui souhaitent avoir un débit régulier car il minimise les changements brutaux de débit. Le principe de ce contrôle de congestion est le suivant :

- L'émetteur envoie un flux de paquets de données au récepteur à un débit donné ;
- Le récepteur envoie un paquet de réponse à l'émetteur au moins une fois par RTT. En se basant sur les informations contenues dans ces paquets, l'émetteur ajuste son débit d'émission en accord avec les équations de débit de TCP pour rester *TCP-Friendly* au niveau du débit généré.
- Si aucune information de réponse n'est reçue du récepteur en deux RTT, l'émetteur réduit de moitié son débit d'émission. Trois valeurs sont nécessaires pour pouvoir mettre en place ce mécanisme : le RTT, la valeur de base du *timeout* et le taux de perte. Le calcul de ces valeurs est réparti entre émetteur et récepteur. Les deux premières sont à la charge de l'émetteur tandis que la dernière est à la charge du récepteur.

Il est important de noter que ce comportement lisse de façon importante le trafic généré par ce protocole. Ainsi, comparé à la même quantité de trafic généré par une version de TCP, le profil de trafic de TFRC comporte moins de pics de débit. En effet, TFRC devrait faire baisser la variabilité du trafic pour les flux qui l'utiliseront. Ces caractéristiques pourraient apporter des gains aux utilisateurs en terme de QoS.

5.5.2 DCCP et diffusion vidéo

DCCP ajoute un contrôle de congestion à UDP et il a été utilisé avec succès en vidéo streaming. [**?**] présente une combinaison de plusieurs éléments qui forme un ensemble de diffusions de vidéo sur ce protocole : une mise en œuvre de TFRC sous Linux, un codec et un système de vidéoconférence à faible latence. Il divise le système vidéo en trois composantes : le codec, le module de DCCP avec le contrôle de congestion TFRC, et l'algorithme qui décide de la qualité vidéo à envoyer. Les variables utilisées pour faire évoluer la qualité de la vidéo au cours du temps sont les suivantes : la résolution, la qualité des images (dans ce cas, il s'agit d'images de types JPEG) et le débit de la vidéo (qualité d'encodage).

MFRC

MFRC [**?**] propose un mécanisme simple de contrôle de congestion en fonction des caractéristiques des flux de médias. Il a été créée comme une solution aux problèmes que les applications

multimédia expérimentent avec les TFRC. MFRC a été pensé pour être mis en œuvre conjointement avec DCCP.

MFRC montre que le démarrage dans le mécanisme TFRC n'apporte rien de bénéfique pour les diffusions de médias et qu'en général il provoque un retard inutile pour le démarrage de l'application. L'approche proposée pour faire face à ces problèmes est de considérer l'application émettrice « innocente jusqu'à preuve de sa culpabilité ». Il s'agit de laisser l'application se comporter librement jusqu'à ce que la congestion soit détectée. Afin de mettre en œuvre cette approche dans TFRC conjointement, trois différentes phases sont proposées : peu congestionné, congestionné et récupération.

Phase non congestionnée (*Uncongested Phase*) C'est la phase de lancement. Au cours de cette phase, l'application peut faire varier le débit de la vidéo jusqu'à une valeur maximale. Ce taux maximal est un paramètre fixé à la création de la connexion et peu enclin à causer une congestion. Une hypothèse, concernant ce taux maximum, est que l'application n'aura pas besoin de plus de bande passante même si elle est disponible. L'application reste dans cette phase jusqu'à ce qu'un message informant d'une perte soit reçu ou que le temps d'attente de ce message soit arrivé à expiration. Ensuite le mécanisme de contrôle est passé en phase de congestion. Le délai entre deux « messages d'information sur les pertes »est fixé à 2 RTT au lieu de 4 comme dans TFRC.

Phase de congestion (*Congested Phase*) Au cours de cette deuxième phase, le taux d'envoi est égal à la moitié du taux maximal d'envoi. Ce premier taux est réduit de moitié chaque RTT en cas de notifications de perte de paquets ou si son délai expire. La valeur exacte du taux réduit de moitié d'envoi est $min(X/2, X_{recv}/2)$ où X est le taux calculé par l'expéditeur et X_{recv} le taux renvoyé par le récepteur.

Lorsque l'émetteur détecte que la congestion a disparu, il se met en phase de récupération. La congestion est considérée comme terminée après un délai ou un nombre de perte de paquets raisonnable.

Phase de récupération (*Recovery Phase*) Le débit d'envoi initial de cette phase est le même que celui de la phase de congestion. À partir de cette valeur, le mécanisme TFRC est appliqué comme indiqué dans [**?**]. L'émetteur reste dans cet état jusqu'à ce que le taux déterminé soit égal au taux maximal de la phase initiale, puis l'expéditeur se met dans cette phase.

5.6 Conclusion

On peut remarquer que la fiabilité du transfert et le principe d'un transfert continu temps réel sont deux concepts difficilement associables sans favoriser de façon perceptible l'un des deux. Deux méthodes d'association se sont distinguées dans notre présentation :

1. la diminution de fiabilité d'un protocole fiable afin de le rendre moins sensible aux pertes et donc plus temps réel par exemple TCP-CM ou PR-SCTP,
2. l'ajout à un protocole temps réel, donc non-fiable, d'un principe de contrôle de flux par exemple DCCP.

TFRC est pressenti comme le futur mécanisme de contrôle de congestion qui véhiculerait les informations de *streaming* multimédia. Cette évolution devrait permettre une diminution

à long terme des protocoles non *TCP-Friendly* (UDP par exemple) qui réalisent cette tâche actuellement.

En conclusion de cette étude, les caractéristiques de DCCP font qu'il semble le protocole le plus adapté à la diffusion multimédia continue. Néanmoins, il garde la possibilité de véhiculer des informations moins critiques (données d'une page HTML par exemple) de façon moins optimale grâce à la présence d'un contrôle de congestion fonctionnant sur le modèle des implémentations actuelles de TCP. De plus, il présente des possibilités d'évolution pour les trafics qui ne sont pas identifiés à l'heure actuelle.

Ainsi, il représente une synthèse intéressante des meilleures solutions proposées au niveau transport pour mettre en place une QoS de bout en bout dans une architecture hétérogène.

CHAPITRE 6

Contrôle de congestion vidéo

Sommaire

6.1 Introduction

La diffusion multimédia sur Internet est désormais courante et les solutions de diffusion sont nombreuses. Mais actuellement il n'y a pas de solution idéale. Certaines d'entre elles sont normalisés par l'IETF, tels que RTP [**?**], RTSP [**?**], d'autres sont propriétaires. Actuellement, les fournisseurs de contenus multimédia sur Internet comme `http://youtube.com` ou `http://dailymotion.com` envoient seulement des fichiers vidéo fixe. Ainsi, ils transmettent des fichiers vidéo de mauvaise qualité avec de faibles définitions afin de prévenir la baisse de bande passante. Si la bande passante disponible est en baisse au-dessous du *bitrate* vidéo au cours de la transmission, le tampon du lecteur vidéo risque de se vider et la vidéo de s'arrêter. Les diffuseurs de vidéos proposent parfois une meilleure qualité pour quelques vidéos, mais c'est l'utilisateur qui prend la responsabilité de les regarder. Ils n'offrent pas le choix d'une qualité automatique afin d'éviter toute responsabilité, si un arrêt se produit.

Au niveau du transport, plusieurs solutions existent aussi. Certaines d'entre elles n'utilisent aucun contrôle de congestion, tels que RTP sur UDP, d'autres utilisent le contrôle de congestion pour les fichiers « normaux »c'est-à-dire indépendants de la nature des informations transmises, tels que HTTP/TCP. Même si beaucoup de propositions ont été avancées, elles n'ont pas remplacé les solutions actuelles. DCCP est un nouveau protocole utile, car il offre des mécanismes internes qui permettent de définir de nouvelles stratégies du contrôle de la congestion. En effet, DCCP permet de mettre en œuvre et de comparer les stratégies adaptées au transport de contenus multimédia.

Une de ces stratégies est l'adaptation d'une vidéo à la bande passante disponible. Cela permet de respecter le flux temps réel. Ce type d'opération ne peut être réalisé que par un mixeur, tels que définis dans [**?**]. Le contrôle de congestion utilisé par le protocole TCP vidéo doit être convivial, utiliser la bande passante allouée et enfin offrir le meilleur aspect visuel de la vidéo reçue.

Des topologies complexes sont souvent utilisées. L'architecture sans fil est spécifique, par exemple les retransmissions MAC, les paquets perdus et des retards. Le multimédia en temps réel s'ajoute à cette complexité, non seulement parce que plusieurs protocoles peuvent être utilisés,

mais principalement parce que les données reçues sur le client doivent être transmises selon les données de la vidéo d'entrée. Aussi, les données multimédia reçues sont sensibles aux spécificités du réseau sans fil. Être capable de simuler une architecture de Vidéo à la Demande (VoD), composée d'un serveur, d'un mixeur (voir ci-dessous) et d'un client, ouvre des possibilités réelles d'optimisation de comparaison de stratégies dans un contexte de flux réels. Par conséquent, une plate-forme modulable est nécessaire pour traiter différents protocoles de transport.

6.2 Plate-forme de diffusion

Elle consiste en une plate-forme d'essai pour la simulation générique multimédia, avec les principales caractéristiques suivantes :

Flexibilité : divers protocoles peuvent être simulés ;

Modularité horizontale et verticale : il est facile de se brancher sur les protocoles de simulation de l'environnement, à la fois verticalement, c'est-à-dire interposer/modifier les protocoles entre client et serveur (comme DCCP), et horizontalement, c'est-à-dire interposer/modifier des éléments de la chaîne de diffusion (comme le mixeur).

Basée sur un simulateur : Network Simulator (NS2 [**?**]) est fréquemment utilisé dans le domaine de la recherche.

Utilisation des données d'applications réelles : même si le réseau est simulé, **les données ne sont pas simulées**, ce qui permet de donner des résultats précis lors de l'analyse de protocoles de contrôle de congestion, par exemple. Notre banc d'essai évalue la qualité finale de la vidéo dans NS2. Ceci est réalisé en intégrant de nouveaux modules de contenu réel et d'évaluation de la qualité visuelle de la vidéo grâce au calcul du PSNR (Peak Signal-to-Noise Ratio) et du SSIM (Structural SIMilarity)[**?**].

Le Peak Signal-to-Noise Ratio (PSNR) mesure la distorsion utilisée dans la compression d'images qui permettent de quantifier les performances du codec par mesure de la qualité de l'image compressée par rapport à l'original. Le PSNR est exprimé en décibels et une image de bonne qualité donne un PSNR variant entre 30 et 40dB. Cette valeur ne doit pas être prise pour une valeur objective de la qualité visuelle d'une image.

Le Structural SIMilarity [?] (SSIM) est une mesure de similarité entre deux images numériques, développée pour comparer la qualité visuelle d'une image compressée par rapport à l'image originale. Cette mesure comble les lacunes du PSNR en tenant compte des caractéristiques du système visuel humain plus sensible à l'évolution de la structure d'une image, plutôt qu'une différence entre deux pixels. La valeur du SSIM est une valeur décimale comprise entre 0 et 1. Un 0 signifie qu'il n'y a aucune corrélation avec la photo originale, et 1 signifiant que les deux images sont identiques.

6.2.1 Architecture générale

L'architecture générale de notre banc d'essai de simulation est composée de deux parties principales : le mixeur et le client.

La figure **??** présente le modèle mixeur tel qu'il a été modélisé dans NS2. Trois types de données vidéo peuvent être utilisées comme entrée du mixeur :

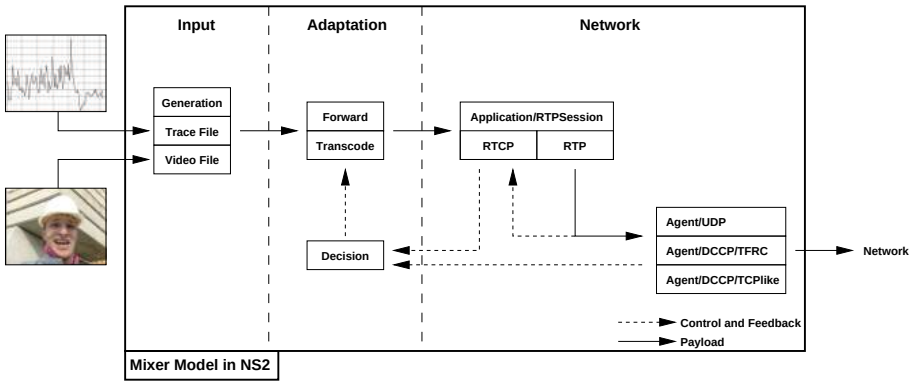


FIGURE 6.1 – Modèle du mixeur dans NS2.

- Données générées : Le mixeur génère ses propres données à l'aide de divers algorithmes. La répartition des différents types d'images (I, P et B) est donnée par le générateur de flux GISMO [**?**]. GISMO est un outil développé à l'université de Boston qui permet de générer à la fois les spécificités d'ensemble d'une vidéo et le déroulement des sessions des clients y accédant.
- Traces réelles : Lors d'une transmission réelle, la taille des paquets envoyés sur le réseau est analysée et les caractéristiques des paquets sont stockées, afin qu'elles puissent être utilisées dans NS2.
- Diffusion réelle : Ce mode permet d'utiliser les vraies vidéos dans NS2. La mise sous forme de paquets des données est effectuée par le mixeur. Les paquets sont envoyés à travers les différents modules NS2 et la vidéo est véritablement transmise entre le serveur et le client final.

La vidéo en entrée est envoyée au noyau du mixeur, c'est-à-dire au module d'adaptation, qui décidera soit tout simplement de transmettre le flux, soit de le transcoder en vue de l'adapter au mieux à la bande passante disponible. Le mixeur RTP est détaillé dans la section suivante. Les paquets sont envoyés au module réseau qui comprend une application qui gère les agents RTP et RTCP puis à l'agent de transport qui peut être par exemple UDP ou DCCP.

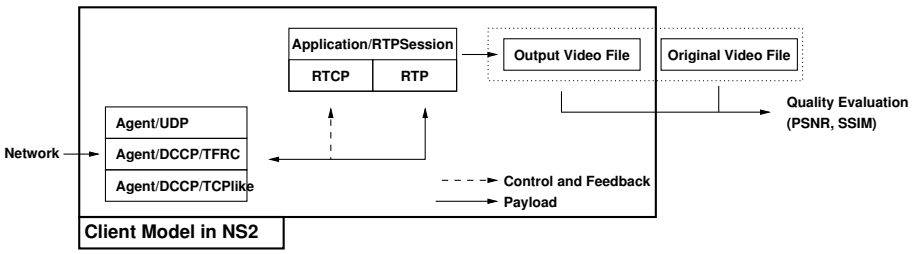


FIGURE 6.2 – Modèle du client dans NS2.

La figure **??** présente le modèle du client, tel qu'il a été modélisée dans NS2. Le client reçoit les paquets en provenance du réseau, et il peut, grâce à RTPSession, reconstruire la vidéo exactement comme s'il l'avait jouée directement sur son écran. Cette vidéo est renvoyée vers un fichier vidéo compressé qui peut être comparé à la vidéo originale. Enfin, le PSNR et le SSIM peuvent être calculés pour voir exactement quels effets ont les pertes et la gigue sur la vidéo.

Ce banc d'essai de simulation nous permet de tester différentes stratégies de flux avec un contenu multimédia avec divers protocoles de transport. De plus, elle nous permet aussi de voir les effets des problèmes de réseau sur des flux de données clairement orientés « contenu ». En effet, lors de la diffusion multimédia, les pertes de paquets n'auront pas le même effet sur la qualité de la vidéo. Certains paquets vont affecter gravement la qualité visuelle de la vidéo, alors que d'autres ne le feront pas. Cette différence dépend de divers paramètres tels que le type de paquets perdus, l'instant où le paquet est perdu dans le groupe d'images (*Group Of Pictures* GOP), etc. En fait, si la dernière image P d'un GOP est perdue, la qualité ne sera pas affectée parce que, juste après cette image, une image I sera décodée. Comme la dernière image P d'un GOP et la première image de I à la suite du GOP n'ont pas de dépendance temporelle, *une seule image* sera endommagée.

Autre exemple : lorsqu'un paquet est perdu sur une image P juste avant un mouvement de la caméra, l'image sera endommagée, mais le mouvement de la caméra va supprimer cette erreur.

C'est pourquoi il est nécessaire de calculer le PSNR et le SSIM, et pas seulement de compter les paquets perdus pour évaluer la qualité de la vidéo.

6.2.2 Extensions de NS2

Par rapport à la version originale de NS2, nous avons fait les extensions suivantes :

- Actuellement, NS2 ne contient pas DCCP. La mise à jour proposée par Mattsson[**?**] a été utilisée pour la simulation.
- Cependant, ce patch ne fonctionnant pas d'origine pour les liaisons sans fil, nous l'avons modifié [3] dans le but de travailler avec des liaisons sans fil.
- Nous avons également modifié le code de DCCP afin d'ajouter et d'utiliser l'option qui stocke les temps de retransmission.

6.2.3 Le mixeur

Architecture du mixeur

Un mixeur RTP [**?**] reçoit des paquets RTP provenant d'une ou plusieurs sources, modifie éventuellement le format des données, combine les paquets d'une certaine manière, puis transmet un nouveau paquet RTP.

Ce mixeur a été développée dans le cadre du projet NetMoVie [**?**], qui fait partie d'un projet plus vaste appelé MoVie de notre laboratoire. Le mixeur n'est pas seulement compatible avec le standard RTP, mais a également étendu les fonctionnalités comme le transcodage à la volée ou l'adaptabilité des contenus à des contraintes réseau.

Il est situé au plus près du client afin de réagir le plus tôt possible à la variation de la bande passante. Comme les clients sont connectés à un réseau sans fil, le mixeur devrait être idéalement situé dans la passerelle entre le réseau filaire et le réseau sans fil.

Le mixeur modifie les données transmises en vue de les adapter au client ou d'optimiser la bande passante disponible. Le mixeur est également en mesure de faire face à plusieurs types de codage, comme le codage hiérarchique ou le codage vidéo MPEG. Pour réaliser cette adaptation, le mixeur comporte trois niveaux de fonctionnalités :

- Le niveau 1 correspond à la sélection des couches (correspondant à différentes qualités d'images) de la vidéo à transmettre dans le cas d'un codage hiérarchique. Selon les rapports

3. Le nouveau patch est disponible à l'adresse `http://lifc.univ-fcomte.fr/~dedu/ns2/`.

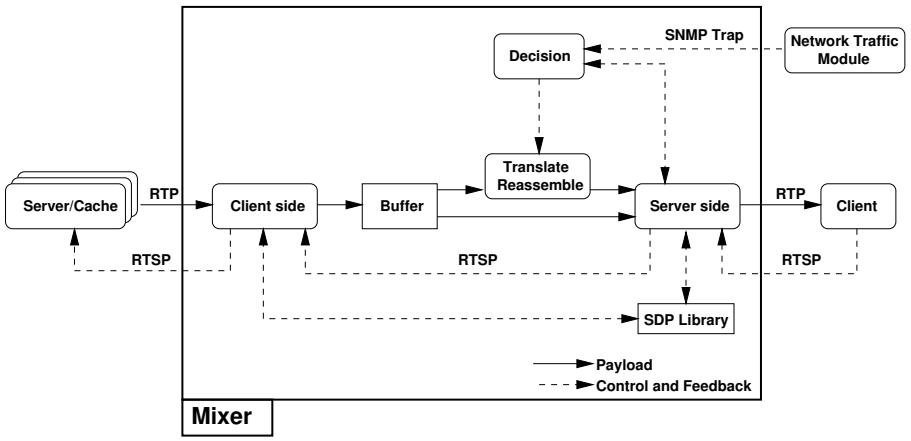


FIGURE 6.3 – Architecture interne de notre mixeur.

RTCP, le mixeur choisit le nombre de couches d'amélioration à transmettre en plus pour garantir une qualité vidéo optimale pour le client.

- Le niveau 2 transforme une vidéo hiérarchique en une vidéo ayant un codage traditionnel pour assurer la compatibilité avec les acteurs qui ne peuvent pas décoder de vidéo hiérarchique.
- Le niveau 3 représente la capacité de transcoder une vidéo. Par exemple, si la bande passante disponible est en baisse, le mixeur peut transcoder la vidéo à un débit (bitrate) plus faible afin d'éviter les pertes de paquet.

Dans son environnement réel, le mixeur peut être interfacé avec n'importe quel serveur qui utilise les protocoles standards RTP/RTCP et RTSP de diffusion de contenu. Le mixeur utilise aussi les bibliothèques de codage vidéo qui regroupe plusieurs des codecs comme *FFmpeg*[4] et *XviD*[5]. Des essais ont été effectués avec un serveur vidéo *Darwin Streaming Server* et des clients comme *Quicktime* ou *Videolan*.

Notre banc d'essai de simulation est utilisé pour tester les diverses fonctionnalités du mixeur, dans un environnement qui serait difficile à reproduire dans la réalité.

Les modules du mixeur

La figure **??** nous montre les différents modules du mixeur :

côté client Ce module permet au mixeur d'être connecté à des serveurs vidéo.

côté serveur Ce module consiste en la réalisation d'un serveur RTP/RTSP. Le mixeur est considéré comme un serveur du point de vue du client.

tampon La mémoire tampon sert à stocker un certain nombre d'images avant le début de la diffusion vers le client. Le tampon et le module de transcodage sont les deux éléments les plus complexes à gérer. Le mixeur doit surtout être réactif pour être en mesure de s'adapter aussi vite que possible à un changement dans les conditions de diffusion du flux. Mais le tampon ne doit pas être trop rempli pour que la diffusion soit effectuée en temps presque réel.

4. Home page : `http://ffmpeg.mplayerhq.hu`.
5. Home page : `http://www.xvid.org`.

Transcodage/réassemblage Ce module est l'élément qui permet d'ajuster la qualité du flux en fonction des diverses contraintes qui agissent sur la transmission. L'un des principaux points de ce module est le choix de la modification de la politique d'adaptation. Une vidéo dont la compression comprend les dépendances temporelles ne peut être décodée directement à tout moment. En effet, il est nécessaire d'attendre une nouvelle image I, ou de revenir à une image I précédente et de décoder toutes les images jusqu'à l'obtention de l'image actuelle. Afin d'optimiser la mémoire, il n'y aura jamais plus que l'actuel GOP stocké dans la mémoire tampon.

Décision Ce module prend en compte tous les paramètres qui lui sont conférés : des informations de retour de la part du module serveur (rapports RTCP, par exemple) ou des informations sur la bande passante disponible en provenance du module réseau. Ensuite, il choisit la meilleure vidéo pour le client en fonction de la bande passante disponible et la qualité vidéo disponible sur les serveurs.

Description d'une session

1. Un client se connecte au mixeur par l'intermédiaire du module serveur et lui demande le démarrage de la visualisation de la vidéo.
2. La demande est transmise au module de décision qui demandera à l'architecture globale les différentes qualités disponibles pour la vidéo requise.
3. Le module de décision choisit la vidéo qui est la plus adaptée aux caractéristiques du client et aux contraintes du réseau.
4. Le module client demande la vidéo choisie au serveur sélectionné. Dans le meilleur des cas, une vidéo, de la qualité qui répond aux besoins, est directement disponible dans le système et aucun transcodage ou adaptation n'est nécessaire. Si ce n'est pas le cas, le module de décision choisit la méthode de transcodage.
5. Dès que le flux est reçu par le module serveur, il est renvoyé directement au client.
6. Si un changement de qualité est nécessaire, le module de décision demande une vidéo au serveur si une vidéo plus adaptée est disponible. Mais, dans le but d'adapter la qualité au plus vite, il demande au module de transcodage de changer la qualité du flux dans l'attente d'un nouveau.
7. Aussitôt que la nouvelle vidéo est disponible, le mixeur arrête le transcodage et diffuse le nouveau flux vidéo.

6.3 Contrôle de flux vidéo

Nous verrons ici, quels points considérer dans la gestion d'un flux vidéo et comment cela a été pris en compte dans notre plateforme de diffusion.

6.3.1 Évaluation de la bande passante

La détermination de la bande passante disponible dans un réseau de tout type a toujours été un défi lorsque les conditions du réseau ne sont pas stables. Elle dépend du nombre de routeurs dans un réseau, du nombre de connexions simultanées et de la qualité des services offerts entre chaque point du réseau.

En général, il est normal d'estimer que la bande passante disponible dans un réseau ne devrait pas varier beaucoup dans un laps de temps réduit. Cette présomption est vraie pour les réseaux

câblés, et il devient alors facile d'estimer la quantité de données qui peuvent être transférées entre deux points d'un tel réseau. Les problèmes apparaissent cependant lorsqu'il se réfère à des réseaux sans fil.

Les conditions dans les réseaux sans fil varient beaucoup en raison de facteurs physiques : obstacles sur le chemin entre le point d'accès et le client, de la distance entre le point d'accès et le client, source d'énergie du récepteur, les interférences entre flux, l'intensité du signal en général. Il est donc impossible de conserver une qualité de service constante sur une période de temps pour un appareil mobile. De plus les réseaux ont été élargis à une si grande ampleur que l'on trouve souvent une combinaison de réseaux câblés et sans fil entre deux points d'une communication.

L'évaluation de la bande passante peut se faire avant et/ou pendant la diffusion de la vidéo.

Avant la diffusion

Nous nous basons sur l'algorithme de [**?**] pour estimer la bande passante disponible. En fait nous n'utiliserons pas directement le résultat de cette estimation, nous chercherons à choisir un fichier vidéo encodé pour la bande passante disponible. Un test peut se résumer de manière assez simple (voir figure **??** : on envoie un train de paquets du serveur vers le client par le protocole UDP et on récupère le temps de parcours de chaque paquet. Ces essais sont répétés jusqu'à ce qu'on obtienne trois résultats consécutifs égaux. On peut alors considérer que nous avons une bonne estimation initiale pour lancer le transfert des données.

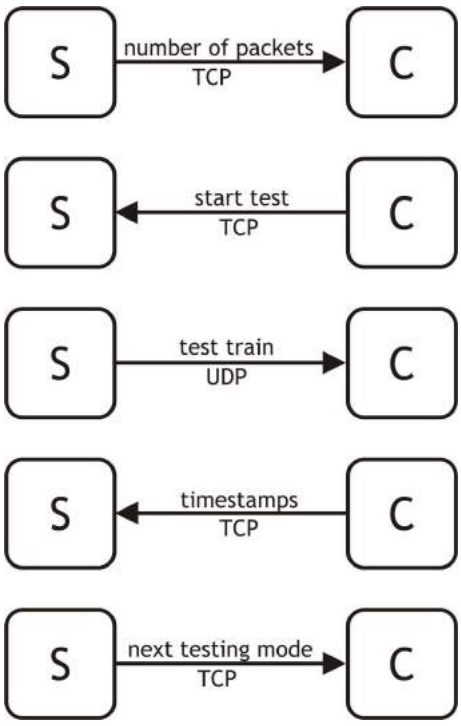


Figure 6.4 – Étapes de l'estimation de la bande passante.

Les temps à partir desquels les paquets sont envoyés (par le serveur) puis reçus (par le client) sont stockées dans deux tableaux. Nous faisons une différence entre les deux tableaux. Ce que nous obtenons n'est pas le temps de déplacement pour chaque paquet, mais le temps

de déplacement plus un décalage donné par le « décalage horaire »entre le serveur et le client. Il ne perturbe pas nos résultats parce que nous avons l'intention de faire deux moyennes : la moyenne pour les 60 premiers pourcents des paquets et une autre moyenne pour les 40 derniers pourcents. Le décalage est constant tout au long de la totalité du train de paquets. Si les 40 derniers pourcents des éléments de la gamme ont une moyenne plus élevée que les 60 premiers pourcents cela signifie qu'il y a eu un retard pour les derniers paquets, donc la bande passante était saturée.

Nous avons donc ajouté une condition basée sur un « temps total de test », ce qui signifie que si un test dure au total plus de 1,5 secondes par exemple, il n'est pas valide. Si le chemin d'accès est déjà saturé avant de commencer le test, tous les paquets seront retardés et la congestion ne sera pas détectée pour notre algorithme. Notre estimation prendra alors plus de temps que la normale. Cette limite de temps permet d'éviter ce faux test.

Premièrement, le serveur calcule le nombre de paquets qu'il doit utiliser pour estimer la bande passante et le transmet au client par une connexion TCP. Ensuite, le client envoie un message de synchronisation en TCP lorsqu'il est prêt à commencer le test. Enfin, le train de paquets est envoyé sur UDP, à un débit défini par le calcul du temps de sommeil entre deux paquets. Le client dispose d'un délai d'attente de 100ms fixé uniquement dans le cas où les paquets se perdraient en cours de route, de sorte qu'il puisse continuer à recevoir le reste du train de test.

Deuxièmement le serveur et le client stockent les temps d'envoi des paquets lorsqu'ils sont envoyés ou reçus et le client les envoie au serveur dans un paquet (TCP) afin que celui-ci puisse les traiter et faire le calcul des retards. Le prochain test est alors fixé : s'il y a eu des retards pour les derniers paquets ou que l'essai a pris plus de 1,5 secondes, le test suivant utilise un débit inférieur, sinon le serveur vise une valeur supérieure dans le tableau des qualités disponibles. Si les trois derniers essais ont donné le même résultat, le test est terminé et la variable « mode »est fixée à *Vrai* ce qui signifie que nous avons une première approximation. Quelle que soit la valeur de « mode », le client reçoit cette variable, il peut donc se préparer pour un nouveau test ($mode = Faux$) ou sortie de son mode « estimation de la bande passante »(mode=Vrai).

Pendant la diffusion

Durant la diffusion de la vidéo, on utilise une approche multi-couche ou *Cross Layer* afin que notre équipement applicatif, notre mixeur, obtienne des informations des modules chargés du transport des données multimédia. DCCP est fourni avec deux contrôles de congestion : TCP-Like ou TFRC.

Le contrôle de congestion TCP-Like est une adaptation directe du principe de contrôle de TCP, c'est-à-dire basé uniquement sur les pertes de paquets. Or ce principe est peu utilisable pour des données continues de type multimédia car une perte entraîne directement une perte de qualité de la vidéo sur le client. Les données étant transmises en temps réel, une perte de paquet est généralement suivie d'une suite de pertes qui elles entraîneraient non pas une simple baisse de qualité mais une absence d'image sur le client voire une coupure nette de la connexion avec obligation de rejouer complètement la vidéo. Pour garder une qualité de diffusion correcte, il faut pouvoir réagir à la baisse de bande passante avant d'obtenir une perte d'un paquet ou du moins éviter la perte consécutive de plusieurs paquets.

6.3.2 Modifications RTP et DCCP pour la diffusion vidéo

Pour calculer et estimer la bande passante, donc faire du contrôle de congestion, il est nécessaire d'obtenir un retour d'information de la part du client. La méthode classique est d'utiliser le

protocole RTCP. Malheureusement, la fréquence des messages RTCP empêche un calcul précis. Une autre méthode consiste à utiliser d'autres protocoles, par exemple *Darwin Streaming Server* utilise un protocole appelé *Reliable UDP* [**?**], qui est basé sur les messages RTCP/APP. Mais la norme RTP / RTCP n'est pas respectée, car la quantité de paquets RTCP est supérieure à ce qui est recommandé dans le protocole.

Pour faire face à ces inconvénients, nous avons décidé d'utiliser une approche multi-couches, où la couche réseau fournit l'information « bande passante réseau »au mixeur. Le contrôle de congestion de DCCP fonctionne de la manière suivante :

- Le récepteur mesure le taux d'erreur et renvoie ces informations à l'expéditeur.
- Grâce au RTT et au taux d'erreur précédemment acquis, le module DCCP/TFRC sur l'expéditeur calcule le taux possible d'envoi des paquets .
- Le mixeur demande périodiquement au module DCCP/TFRC son estimation de la bande passante disponible. Ces informations seront utilisées pour sélectionner le taux de transcodage de la vidéo, qui est détaillé ci-dessous.

La bande passante disponible calculée par TFRC est utilisée comme suit : pour être sûr que la vidéo transcodée ne soit pas bloquée sur le réseau (l'estimation n'est pas toujours correcte), nous choisissons de prendre en compte seulement 90% de la valeur calculée de la bande passante. Ceci permet d'être le plus souvent juste et absorbe la plupart des pics dans le *bitrate* vidéo.

DCCP/TFRC calcule la bande passante une fois par RTT. Transcoder une fois par RTT consomme beaucoup de temps et de ressources au mixeur et la qualité proposée peut s'en ressentir. Ainsi, nous proposons un algorithme de transcodage comme suit :

- Nous définissons un seuil supérieur et un seuil inférieur associés au débit actuel. L'opération de transcodage est réalisée uniquement lorsque le débit moyen entre l'ancienne et l'actuelle bande passante est en dehors de l'intervalle [inférieur, supérieur]. Cette moyenne empêche de trop fort changement de bande passante. Les bornes inférieures et supérieures dépendent du taux actuel et d'une valeur de PSNR. Cette valeur de PSNR assure que le nouveau taux d'encodage est optimal (en terme de qualité vidéo).
- Les valeurs du PSNR sont empiriquement associés à des débits. Étant donné que les petites différences dans PSNR ne sont pas significatives aux yeux humains, une différence d'au moins 2dB est nécessaire. Ainsi, pour le débit actuel, les débits donnant une différence de PSNR d'au moins 2 sont considérés comme bornes inférieures et supérieures de transcodage.
- Enfin, le transcodage est limité à des extrema évidents : un taux minimum garantissant le lisibilité de la vidéo et un taux maximum, défini par le *bitrate* initial de la vidéo.

6.3.3 Demande du client

Si les capacités du réseau de diffusion sont un point important à prendre en compte dans la transmission d'une vidéo, le client final doit lui aussi définir des contraintes. Ces contraintes peuvent être :

- physiques comme la taille de l'écran sur le client influence la résolution envoyée. Il n'est pas nécessaire et surtout inutile d'envoyer une vidéo en 720x576 sur un téléphone portable avec un écran de 320x240. La contrainte physique du client amène une adaptation de la vidéo et peut en même temps fournir un gain de qualité : une vidéo plus petite en taille mais encoder dans une meilleure qualité. La puissance CPU disponible sur le client permet aussi d'adapter le choix de l'encodage, donc de la qualité et donc du débit.
- logicielles comme les logiciels de lecture de vidéos présents sur le client et/ou les encodages supportés mais le lecteur. Beaucoup de lecteur de vidéo savent décoder de multiples codecs

mais avec plus ou moins de faciliter et de fluidité.

Cette description du client existe déjà pour un protocole de transmission audio très utilisé pour la voix sur IP (*Voice Over Ip* VoIP), le protocole SIP *Session Initiation Protocol* [**?**, **?**]. Le client échange avec le serveur, ou directement son interlocuteur, la liste des encodeurs et décodeurs qu'il supporte. Le message INVITE, envoyé par le client en début de connexion, permet de demander l'ouverture d'une session. Ce message contient une syntaxe SDP [**?**] (*Session Description Protocol*). SDP est un format de description et d'initialisation de paramètres. Cette structure de description consiste en plusieurs lignes qui décrivent les caractéristiques du média et donc du client.

Ce même principe peut-être utilisé pour la description d'une session de transmission vidéo et transmettre les caractéristiques physiques et logicielles du client (voir figure **??**). Ce type de contraintes de diffusion ne sera pas étudié dans les expérimentations et simulations présentes dans cette thèse.

6.3.4 Comparaison de la qualité des vidéos

Pour obtenir une évaluation de la vidéo reçue, il est nécessaire de comparer celle reçue à celle présente dans l'espace de stockage des vidéos. Pour comparer deux vidéos, il est nécessaire de comparer les deux images, qui doivent être présentées en même temps. Mais, les vidéos produites par simulation souffrent d'erreurs durant la transmission. Pour comparer les images, nous devons combler les trous faits par les pertes en ajoutant une ou plusieurs photos au début de chacune d'elles. Ensuite, pour chaque ajout, nous faisons une comparaison puis nous ajoutons une autre image, et on compare les vidéos à nouveau, jusqu'à l'obtention d'un nombre identique de photos entre la vidéo initiale et la vidéo générée.

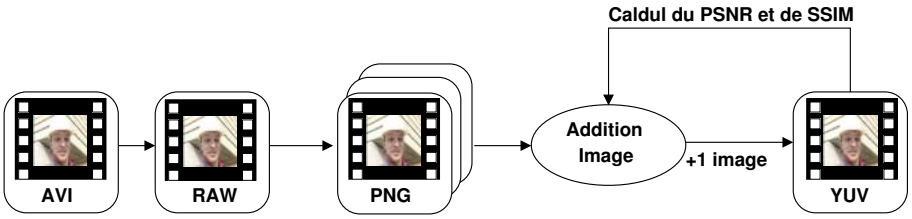


FIGURE 6.5 – Reconstruction de la vidéo transmise.

Comme indiqué dans la figure **??**, pour comparer les vidéos reçues par le client, on utilise les deux méthodes de comparaison : PSNR et SSIM.

Pour effectuer ces différentes comparaisons, nous utilisons *MSU Video Quality Measurement Tool*[6] développé par MSU Graphics & Media Lab.

6.4 Expérimentations réelles

Nous avons testé de manière pratique l'utilisation DCCP au lieu de TCP ou UDP comme protocole de transport de contenus multimédia.

Toutes les autres expériences effectuées jusqu'ici, dans le cadre d'autres projets, ont été fait en utilisant soit des simulateurs, soit de véritables réseaux mais n'utilisant que des transferts de données par paquet. Cette section présente une réelle diffusion multimédia en streaming sur DCCP.

6. `http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html`

Cette expérimentation sert uniquement à valider les aspects pratiques de DCCP tout en effectuant des tests en continu sur différents réseaux. Sachant qu'il n'y a que quelques applications qui ont déjà mis en œuvre le protocole DCCP, ceci à constitué un vrai défi de configurer les applications afin qu'elles soient en mesure de gérer des flux médias DCCP en raison des fonctionnalités manquantes. GStreamer est le média streaming utilisé dans ce projet d'expérimentation avec le protocole DCCP. Une fois la plateforme configurée, des essais ont été réalisés et les résultats obtenus ont été comparés à des tests identiques utilisant les protocoles de transport TCP et UDP sur RTP.

6.4.1 Définition de la plate-forme

Logiciels utilisés

DCCP est approuvé en tant que norme depuis 2006, mais sa mise en œuvre dans le noyau Linux est encore en cours de développement, et il n'est pas encore stable.

Ainsi, comme le protocole lui-même n'est pas entièrement pris en charge par les systèmes d'exploitation, bien évidemment, toutes les applications qui utilisent actuellement le protocole DCCP sont également en cours de développement, ce qui signifie que les programmes sont instables et pas forcément entièrement opérationnels.

Les deux applications testées en tant que serveur et client seront *VLC Videolan player* et *GStreamer*.

Scénarios

Pour cette partie, deux jeux de tests sont effectués pour chaque scénario testé. Malheureusement ces tests ne peuvent avoir valeur de généralité, surtout dans le cas d'un réseau sans fil au vu des conditions d'expérimentations non exactement reproductibles. Voici les spécifications techniques de ces tests :

Sur le réseau filaire : Cette expérience a été réalisée en utilisant un réseau câblé entre le serveur et le client. Ces conditions réseaux sont optimales. Pour introduire une baisse de la bande passante disponible, nous utilisons un limiteur de débit sur l'interface réseau du serveur.

Sur un réseau sans fil : Cette expérience a été réalisée en utilisant un point d'accès Wi-Fi entre le serveur fixe et le client mobile.

Différents scénarios correspondant à différentes positions du client mobile sont expérimentées : client et serveur proche (dans la même pièce), client et serveur séparés par un mur et enfin client et serveur éloignés de plusieurs pièces.

Nous n'utilisons pas dans ce cas de limiteur de débit car les interférences et l'atténuation du signal suffisent à limiter la bande passante de manière drastique.

6.4.2 Résultats

Les résultats présentés se limiteront, dans le cas de l'expérimentation réelle, à l'estimation de la qualité de la vidéo reçue au moyen de la valeur du PSNR de ses images. Ces expérimentations ont simplement pour but de valider les capacités de DCCP dans le transport de vidéo.

Nous ne présenterons pas tous les scénarios testés car ceux-ci montrent les mêmes types de courbes. Les tests effectués vont tous dans le sens des conclusions présentées ci-dessous.

Le graphique **??** démontre clairement la supériorité de TCP en terme de fiabilité de transfert. Les courbes DCCP comme TCP ne sont pas visible sur le graphique car elle représente une ligne droite avec une ordonnée maximale égale à une valeur de PSNR de 100. DCCP comme TCP fournit une remarquable qualité d'image. Ces tests filaires ne présentent pas de nouveaux résultats probants. Ils nous permettent de certifier la faisabilité et le bon fonctionnement du protocole DCCP comme un autre protocole de transport pour la vidéo.

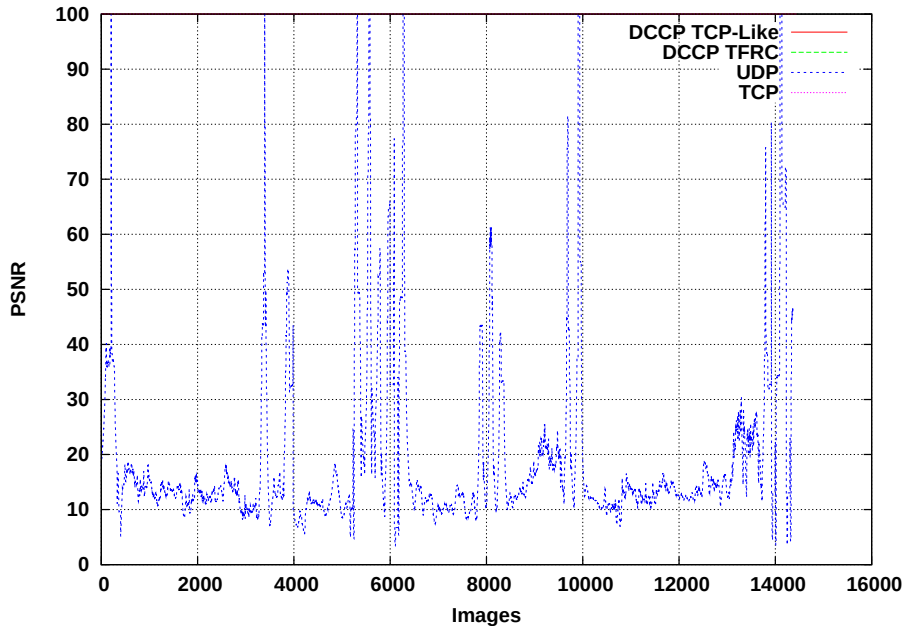


FIGURE 6.6 – PSNR de la vidéo transmise lors d'un transfert filaire.

La courbe **??** montre clairement les avantages de DCCP par rapport au protocole existant et actuellement utilisé, UDP, en terme de qualité vidéo perçue. TCP présente une qualité vidéo parfaite, cela correspond au fait qu'il s'agit d'un protocole fiable dont tous les paquets transmis sont reçus. Par contre TCP introduit, de par ses retransmissions, un délai important mais d'un autre côté, la vidéo reçue est identique à l'originale. Les pics observés pour DCCP sont la conséquence de l'adaptation de son débit à la bande passante disponible.

6.5 Simulations

6.5.1 Protocole de test

Scénarios de simulations

Le simulateur de réseau Network Simulator Version 2.30, fréquemment utilisé dans la recherche, est utilisé pour effectuer des simulations. Un serveur de diffusion vidéo, un point d'accès (AP), et des clients mobiles sont créés pour la simulation. Cette simulation correspond à un homme marchant dans la rue tout en regardant les nouvelles à la demande (*News On Demand* NoD). Le mobile se déplace selon le scénario suivant (figure **??**) :

1. Le mobile s'éloigne linéairement du point d'accès AP.
2. À $t = 100s$, il s'arrête et reste immobile pendant $50s$.
3. Puis, il retourne à sa position initiale et la simulation se termine à $t = 250s$.

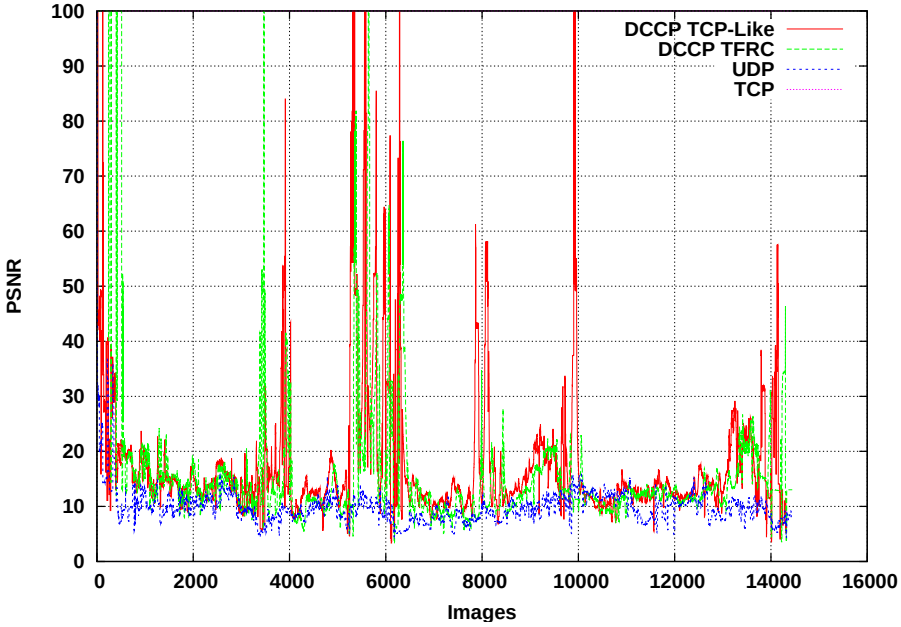


FIGURE 6.7 – PSNR de la vidéo transmise lors d'un transfert sans fil.

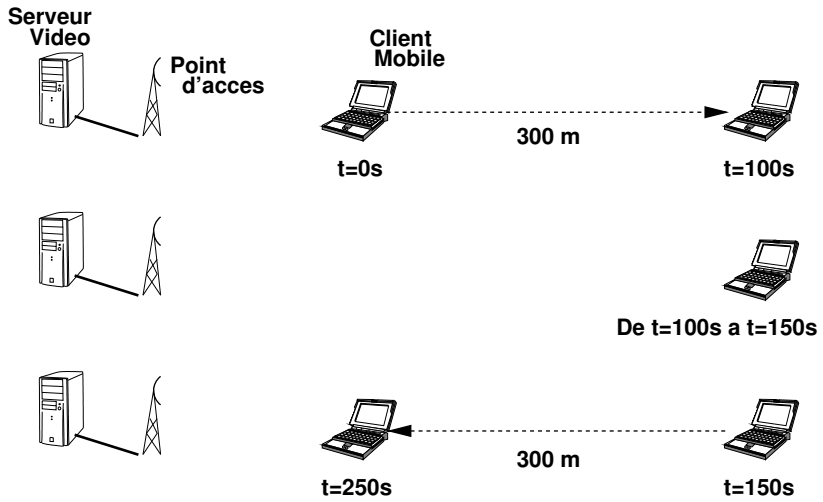


FIGURE 6.8 – Scénario de la simulation.

Pendant la période où le client est immobile, le mobile est situé au bord de la zone de couverture du point d'accès sans fil, le débit utile chute (figure **??**) à l'endroit où la plupart des *retransmissions* vont apparaître.

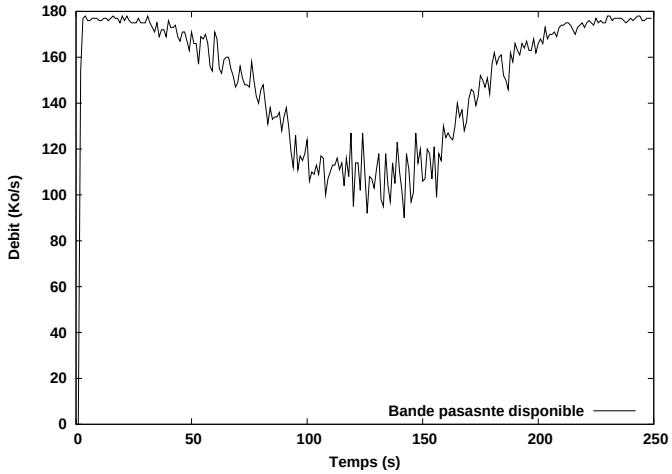


FIGURE 6.9 – Bande passante disponible durant notre scénario.

Nous transférons trois fois la même vidéo avec le même scénario de déplacement. Seul le protocole de transport varie d'une simulation à l'autre. Nous utilisons :

- RTP sur UDP, le protocole de transport originel pour la diffusion de vidéo,
- RTP sur DCCP/TFRC avec adaptation de la vidéo. Le mixeur adapte la vidéo à la bande passante disponible,
- DCCP/TFRC avec notre modification du RTT : DCCP/TFRC-LD.

6.5.2 Résultats pour UDP vs DCCP/TFRC

Débit

La figure **??** montre les trois étapes de la circulation du client. Au cours des 50 premières secondes les débits des deux transferts, DCCP/TFRC et UDP, sont plus faibles que la bande passante disponible parce que la vidéo contient principalement encore des images fixes ou avec de petits mouvements. En effet, le débit de la vidéo est inférieur à la bande passante disponible. Après cet intervalle de temps, le débit est plus proche de la bande passante disponible et les deux courbes de débits sont presque les mêmes. Ce résultat doit être mis en relation avec les pertes de paquets. UDP n'est pas un protocole avec un contrôle de congestion et doit nécessairement subir plus de pertes que DCCP/TFRC.

Les pertes de paquets

Dans la figure **??**, durant les 250 secondes de la transmission vidéo RTP/UDP, 2063 paquets sont perdus. Avec DCCP/TFRC, ce nombre est réduit à un total de 180 paquets seulement. Dans le cas d'un transfert sur UDP, la plupart de ces pertes apparaissent lorsque la bande passante disponible est plus petite que le *bitrate* vidéo. Ce graphique montre l'efficacité de DCCP/TFRC pour évaluer la bande passante disponible et réduire ainsi au minimum les pertes en fonction de cette évaluation. On observe toujours des pertes régulières de paquets mais celles-ci sont moins fortes donc la baisse de qualité est moins flagrante.

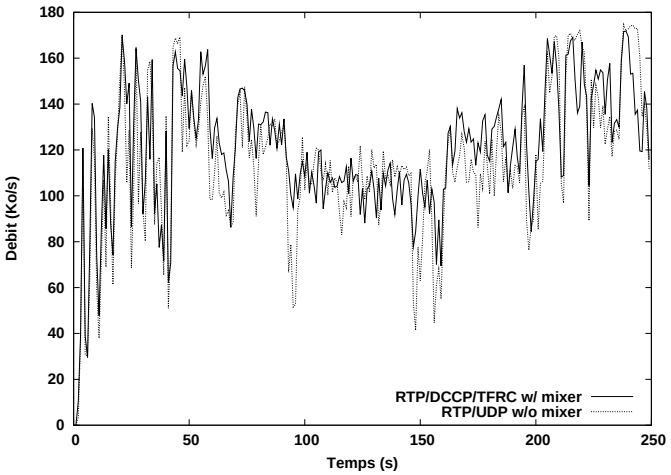


FIGURE 6.10 – Comparaison entre le débit de la diffusion de la vidéo sous UDP et DCCP.

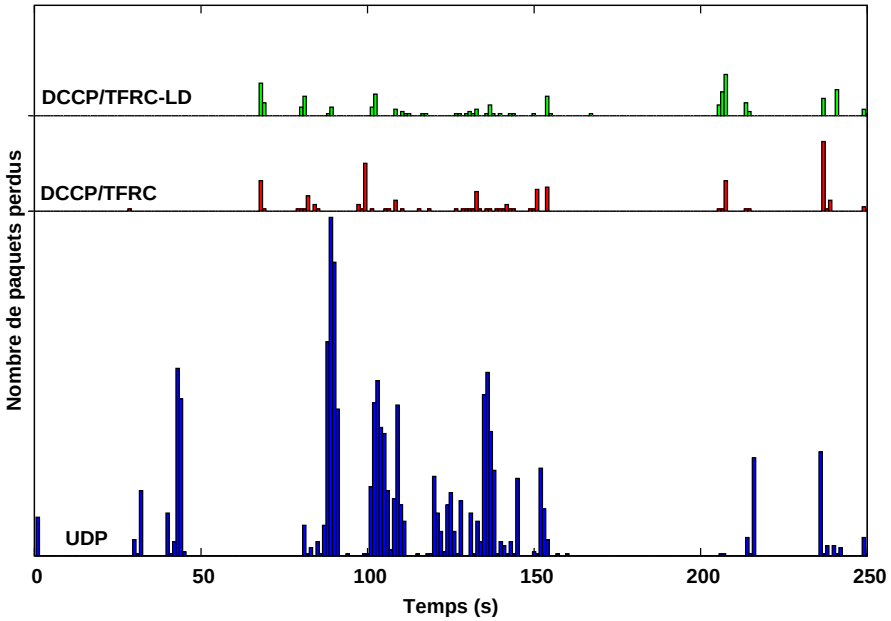


FIGURE 6.11 – Comparaison des pertes de paquets entre UDP, DCCP/TFRC et DCCP/TFRC-LD.

Comparaison des vidéos

Pour chaque protocole de transport que l'on simulera, la vidéo originale et la vidéo reçue sont comparées en utilisant PSNR figure **??**, et le SSIM, figure **??**.

La figure **??** présente le nombre d'images reçues pour chaque qualité (pour chaque unité de PSNR). Nous pouvons voir que la courbe pleine de DCCP/TFRC a plus d'images de bonne qualité (supérieur à 35dB) que la courbe en pointillé de l'UDP. Cela est dû au fait que TFRC évalue mieux la bande passante et que le mixeur vidéo effectue l'adaptation. La vidéo lue par le client sur son portable a donc une meilleure qualité selon les critères PSNR.

Le ratio global du temps pendant lequel le mixeur est actif peut être estimé sur la figure **??** grâce à la différence entre les parties en pic de la courbe. Cette valeur est d'environ $(350 - 300)/350 = 14\%$, ce qui est très faible par rapport aux améliorations de la qualité des flux vidéo.

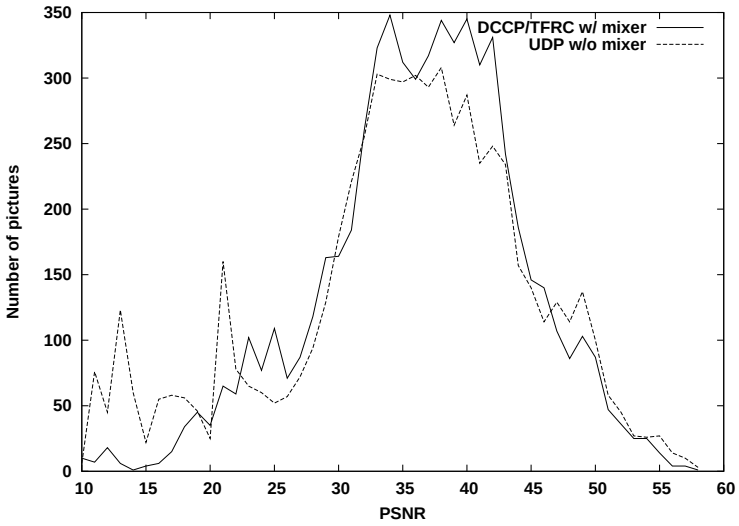


FIGURE 6.12 – Comparaison du PSNR entre UDP et DCCP/TFRC.

La figure **??** présente le pourcentage d'images trouvées dans chaque segment de qualité (mauvaise qualité, moyenne -, moyenne +, bonne qualité). Nous pouvons voir que les deux vidéos ont le même nombre d'images de bonne qualité, mais que DCCP/TFRC présente beaucoup plus d'images dans l'intervalle « moyenne + »qu'UDP. La capture d'écran (figure **??**) faite lors de la diffusion synchronisée des vidéos récupérées sur les clients DCCP/TFRC et UDP montre bien la baisse de qualité due à la perte de paquets. La valeur SSIM moyenne et l'écart type des valeurs pour UDP sont respectivement de 0,95 et 0,14 , qui sont plus mauvaises que les valeurs de la moyenne et l'écart type pour DCCP/TFRC (0,99 et 0,044). Ceci est en accord avec les résultats obtenus dans le cadre de la comparaison des valeurs du PSNR.

6.5.3 Résultats pour DCCP/TFRC vs DCCP/TFRC-LD

Les paquets perdus

En regardant la figure **??**, on remarque qu'il y a moins de paquets perdus sur la courbe DCCP/TFRC-LD qui inclut la modification du RTT que sur la courbe représentant le protocole simple DCCP/TFRC. Le nombre de pertes de paquet diminue de presque 10 % entre DCCP/TFRC et DCCP/TFRC-LD.

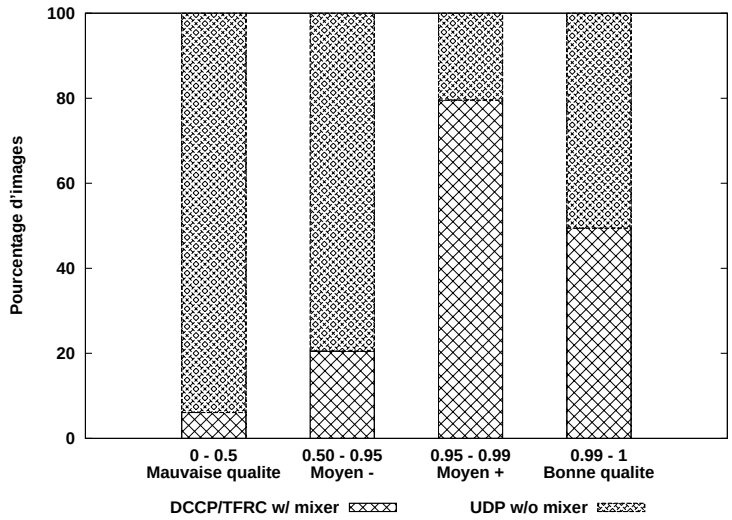


FIGURE 6.13 – Comparaison SSIM entre UDP et DCCP/TFRC.

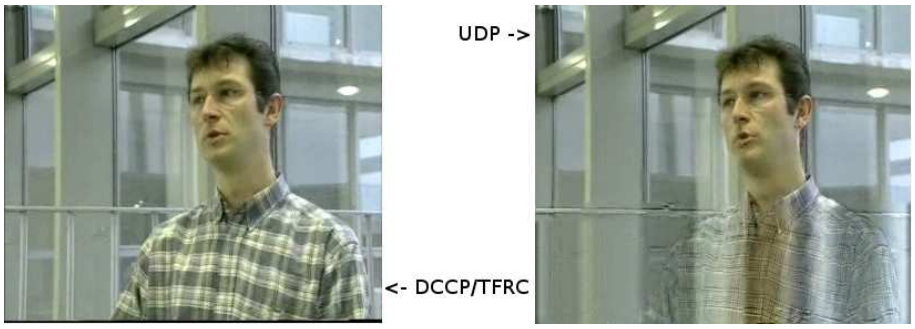


FIGURE 6.14 – Captures d'écran des vidéos diffusées par UDP et DCCP/TFRC.

Comparaison des vidéos

La figure **??** montre que DCCP avec sa modification du RTT (DCCP/TFRC-LD) est essentiellement équivalente à l'autre courbe. Il y a quelques bonnes images supplémentaires (de plus de 40 dB) pour le protocole DCCP/TFRC-LD par rapport à celui sans modification.

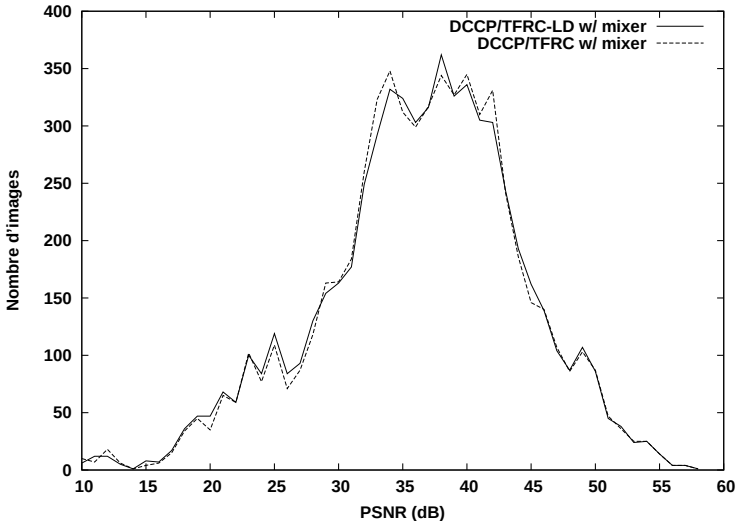


FIGURE 6.15 – Comparaison des valeurs du PSNR entre DCCP/TFRC et DCCP/TFRC-LD.

La figure **??** présente le pourcentage d'images trouvées dans chaque intervalle de qualité (mauvaise qualité, moyenne, moyenne +, bonne qualité). Nous pouvons voir que les deux vidéos sont globalement équivalentes avec exactement les mêmes répartitions. Cependant, l'écart-type de la valeur pour DCCP/TFRC-LD est plus faible avec une valeur de 0,040, ce qui signifie que la vidéo est un peu plus stable que dans le flux DCCP/TFRC.

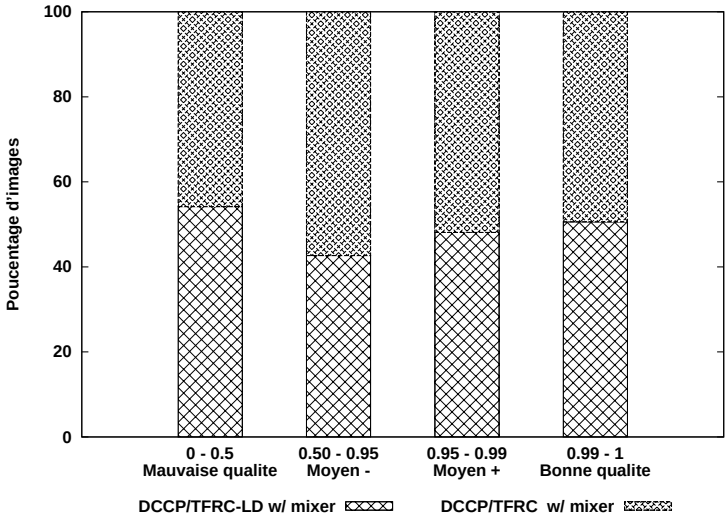


FIGURE 6.16 – Comparaison des valeurs SSIM entre DCCP/TFRC et DCCP/TFRC-LD.

6.6 Conclusion

Ce chapitre met en avant les défauts des protocoles utilisés actuellement dans la diffusion de contenu multimédia sur des réseaux non-fiables. Expérimentalement dans un premier temps, nous avons mis en avant les lacunes d'UDP face à un protocole plus récent : DCCP. Ensuite d'un point de vue simulation, une architecture vidéo complète est présentée. Elle comprend un mixeur, un serveur et un client mobile. Nous avons démontré que ce modèle de simulation peut contribuer à optimiser et à mettre en œuvre un contrôle de congestion dédié à un usage multimédia.

La mesure des améliorations de notre banc d'essai par rapport à la diffusion vidéo classique est importante. Deux types de résultats ont été présentés, pour le réseau (débit et perte de paquets) et pour la visualisation de vidéo reçue. Notre solution donne de meilleurs résultats que le couple classique RTP/UDP pour deux raisons : l'amélioration de la qualité vidéo mesurée à l'aide des fonctions PSNR et SSIM et la capacité de transmettre sur Internet l'ensemble grâce à un contrôle de congestion TCP qui est TCP-Friendly.

En outre, l'utilisation du simulateur de réseau permet la réalisation de scénarios successifs reproductibles avec des données différentes pour valider plusieurs parties d'un développement, comme l'adaptation du mixeur NetMoVie. Grâce à l'utilisation réelle de la vidéo, le contenu multimédia étant effectivement transporté, on redonne au contenu toute son importance au cours de la simulation de la transmission.

L'intérêt d'un mixeur d'adaptation dans une chaîne de transmission a été démontrée. Dans les scénarios, y compris avec adaptation, les vidéos reçues par le client sont de meilleure qualité, que ce soit mathématiquement ou visuellement. Les solutions que nous avons sélectionnées et développées respectent les différentes normes mises en œuvre.

Nous avons proposé de nouvelles solutions de contrôle de congestion combinées avec un mixeur RTP afin d'adapter la diffusion vidéo aux contraintes des réseaux filaires et sans fil. Grâce à notre modèle de simulation, il sera plus rapide et plus facile de proposer de nouvelles stratégies efficaces pour fournir des contenus multimédias. Plus généralement, à notre avis, l'équilibrage de la bande passante dans un réseau sans fil entre un ou plusieurs flux vidéo sera un des grands défis des années à venir.

Conclusion

Dans cette thèse, nous nous sommes intéressés à l'optimisation et à l'adaptation des communications dans un réseau hétérogène. Cette problématique nous à conduit à prendre en compte différents types de transmissions sur différents réseaux afin d'optimiser celles-ci sur la globalité du chemin parcouru.

Dans un premier temps nous nous sommes proposés de réorganiser la circulation des flux informatiques sur le cœur d'un réseau à grande échelle comme Internet (qui est le plus grand d'entre eux). Pour cela nous avons choisi de modifier les priorités de traitements des flux et plus précisément des paquets appartenant à une certaine catégorie de transfert. Beaucoup se sont intéressés à l'amélioration de la gestion des équipements de cœur de réseau mais peu l'ont fait en considérant le type de transfert sur celui-ci. Après avoir présenté un éventail non exhaustif des différentes formes de gestion des files d'attente dans les routeurs, nous avons développé notre propre proposition.

Pour valoriser un réseau, il s'agit de l'utiliser au mieux c'est-à-dire sans en gaspiller ses ressources souvent limitées et coûteuses. Nous avons introduit et défini la notion de distance parcourue par un paquet comme paramètre supplémentaire dans le traitement de ceux-ci par les politiques de gestion de file d'attente dans les routeurs. Nous avons choisi d'intégrer ce critère de décision dans une politique déjà certifiée et éprouvée depuis longtemps : RED.

RED garde, élimine ou marque les paquets arrivants suivant une probabilité dépendante de la file d'attente dans laquelle on se trouve. Sans pour autant changer cette probabilité, nous avons pensé la détourner de sa cible pour l'appliquer à un autre paquet que l'on qualifiera de moins coûteux. En éliminant les paquets proches de leur source, on évite le gaspillage des ressources (liens, routeurs, ...) réseaux tout en augmentant les ressources disponibles (car non encore utilisées). Notre nouvelle politique de gestion de file d'attente se nomme DDRED pour *Distance-Dependent Random Early Detection*.

Les simulations de notre politique ont montré qu'en prenant comme critère d'optimisation, l'évitement du rejet d'un paquet déjà éloigné de sa source, nous avons amélioré la capacité globale de traitement du réseau.

Dans la deuxième parte de cette thèse, nous avons changé de position dans notre architecture réseau : après le cœur de celui-ci, nous voici en fin de parcours. Cette fin de réseau revêt dans notre cas un caractère précis : c'est une liaison sans fil. La liaison radio, contrairement à la liaison filaire et quelle que soit la technologie utilisée dans chacun de ces cas, n'est pas une liaison fiable.

La technologie radio de par ses propriétés physiques (fréquences, puissance d'émission, ...) et l'environnement (extérieur, intérieur, nombre d'équipements présents, ...) dans lequel on l'utilise subit des interférences. Nous nous sommes attardés plus particulièrement sur la technologie Wi-Fi 802.11abg en considérant ainsi un maximum de types de clients mobiles différents (ordinateurs, assistants personnels, téléphones portables, ...).

La gestion des accusés de réception au niveau MAC dans les réseaux Wi-Fi permet de tenir compte des imperfections techniques de ce type de transport. En corrigeant les effets des retransmissions sur le réseau sans fil et des pertes dues à la multiplication de celles-ci, on permet ainsi de rapprocher les réseaux sans fil des réseaux filaires en ce qui concerne la gestion de la fenêtre de congestion tout en gardant une émulation de la fiabilité grâce aux retransmissions MAC.

Dans la dernière partie de ma thèse, nous avons pris le cas d'un type de transfert précis et plus difficile à traiter que les transferts classiques de fichiers : le transfert de données continues complexes ou plus vulgairement le transfert de vidéos. Là où un transfert classique a besoin soit de fiabilité soit de rapidité, la vidéo, elle, a besoin des deux sans pour autant casser ce qui fait sa spécificité : la continuité visuelle.

Pour améliorer le transfert de vidéo, nous nous sommes placés dans le cas d'une architecture avec mixeur RTP, architecture bien spécifique et définie dans les standards de transmission vidéo. Ce mixeur permet le transcodage et le réencodage à la volée de la vidéo. Ce module est inséré dans la chaîne de diffusion au plus proche des clients afin de ne pas multiplier les formats transférés sur le réseau.

Perspectives

Dans le domaine des politiques de gestion de file d'attente des routeurs, la prochaine étape de notre travail sera de calculer la distance moyenne entre les flux de source et de destination mobile. Cela nous permettrait d'augmenter la précision dans l'estimation du chemin parcouru par rapport au trajet complet et d'ainsi améliorer la probabilité de rejet de paquet proposée dans la politique RED. Une autre amélioration sera de proposer une nouvelle mise en œuvre de la distance entre deux machines. Le champ Option qui est utilisé afin de sauvegarder cette information consomme lui aussi des ressources. Une incorporation de la distance « en cours »(TTL) et « totale »est envisageable à l'intérieur de l'unique champ TTL préexistant.

Dans le domaine de la simulation de réseaux, beaucoup d'améliorations sont possible et maintenant disponible en partie :

- Le passage à un générateur de trafic beaucoup plus réaliste et sur un plus longue durée permettra de tester le passage à une échelle nationale réaliste par exemple.
- La nouvelle version de Network Simulator apporte son lot d'avancées technologiques en terme d'intégration avec l'environnement d'utilisation : plus de réécriture des protocoles de transport mais une utilisation directe du protocole du système.
- Avec ses avancées du simulateur, le passage vers une émulation à la place de la simulation semble possible.

DCCP est un nouveau protocole de transport défini dans une RFC et donc considéré comme un standard. Malgré cette standardisation, beaucoup de points concernant la mise en place de ce protocole tant au point de vue réel dans les systèmes d'exploitation qu'au point vue intégration dans les simulateurs réseaux, restent encore à l'état expérimental et varient même d'une mise en œuvre à l'autre. Dans le cadre de l'étude sur la retransmission dans les réseaux sans fil comme dans celui de la diffusion vidéo, cette approximation du standard est source d'erreurs

voire même d'impossibilité d'exécution. Un temps attente est donc nécessaire afin de pouvoir disposer de mise en œuvre stable, quel que soit le domaine d'utilisation de ce protocole.

L'intérêt portée aux équipements mobiles comme clients amène bien sûr à s'intéresser à la gestion de ces clients mobile et plus principalement à leur autonomie. On veut diffuser de la vidéo mais il faut que le client est les capacités requises pour la lire dans les meilleures conditions. L'économie d'énergie est un sujet très actuel.

Il reste de nombreuses pistes à exploiter dans le domaine de la transmission de contenus multimédia en temps réel et du contrôle de congestion. Nous nous sommes intéressés au flux dans ses caractéristiques réseaux (débit, latence). Mais notre contrôle de congestion devra être basé aussi sur un contrôle des données en elles-mêmes. Il nous faut donc maintenant nous intéresser à la gestion non plus des flux mais au contenu de ces flux : les données brutes de la vidéo.

De plus nous nous sommes focalisé sur un type unique de trafic, le transfert unicast. Il s'agit maintenant d'évoluer vers la gestion de transfert multicast beaucoup plus utilisés dans le domaine de diffusion vidéo et plus précisément encore, l'accès aux chaînes de télévision par Internet.

Résumé

Mots-clés: Politique d'ordonnancement dans les routeurs, retransmission MAC, protocole de transport, DCCP, diffusion vidéo, simulation

Dans le cadre de la diffusion de flux vidéo sur un réseau hétérogène (filaire et radio), cette thèse a pour objectif la spécification, la modélisation et l'évaluation sur des outils de simulation d'un ensemble de propositions. La gestion du transfert que nous proposons s'adapte aux conditions d'utilisation du réseau traversé. Dans un premier temps, nous avons créé une nouvelle politique de gestion de file d'attente nommée DDRED (Distant-Dependent RED) au niveau des routeurs prenant en compte les distances parcourues des flux dans le réseau. Elle nous permet d'augmenter les ressources réseaux disponibles en rejetant judicieusement certains paquets de données plutôt que d'autres. Sur l'extrémité du réseau, nous proposons aussi de prendre en compte les perturbations Wi-Fi et ainsi de différencier les erreurs dues aux interférences, de celles dues à une congestion. L'augmentation des temps de transmission est causée soit par une augmentation des files d'attente des routeurs et donc c'est une congestion, soit par les retransmissions générées par la couche MAC dans les réseaux sans fil. De plus nous avons conçu un système complet de diffusion vidéo qui utilise DCCP. Il permet de proposer une qualité de diffusion optimale et dépendant directement de la bande passante disponible. Cette qualité est obtenue par une communication entre les couches transport et application. DCCP détermine la bande passante disponible et l'encodeur vidéo définit si un réencodage est nécessaire. D'un point de vue technique, toutes ces améliorations ont été développées, validées et évaluées grâce au simulateur de réseau NS2.

Abstract

Keywords: Queue managment policy in routers, MAC retransmission, transport protocol, DCCP, streaming, simulation

As part of the video streaming on a heterogeneous network (wired and wireless), this thesis aims to specification, modelling and evaluation with simulation tools of a package of proposals. The management transfer that we offer adapts to use the network through. In a first step, we created a new queue management policy named DDRED (Distant-Dependent RED) at routers taking into account the length of a flow in the network. It enables us to increase the available network resources by wisely rejecting some data packets rather than others. At the end of the network, we also propose to take into account Wi-Fi retransmissions and thus differentiate errors due to interference from those due to congestion. The transmission time increase is caused either by an increase of queues routers and therefore it is a congestion, either by retransmissions generated by the MAC layer in wireless networks. In addition, we have designed a streming architecture that uses video DCCP. It can offer optimal broadcast quality and directly dependent to the available bandwidth. This quality is achieved by a crosss-layer communication between transport and application. DCCP determines the available bandwidth and video encoder defines whether a re-encoding is necessary. From a technical point of view, all these improvements have been developed, validated and evaluated through the network simulator NS2.