Amélioration de la transmission de contenus vidéo et de données dans les réseaux sans-fil

Wassim Ramadan

N° d'ordre 134
Année 2011

Université de Franche-Comté
Sciences, Techniques et Gestion de l'Industrie
École doctorale SPIM
Laboratoire d'Informatique de l'Université de Franche-Comté – EA 4269

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

Spécialité : **Informatique**

Amélioration de la transmission de contenus vidéo et de données dans les réseaux sans-fil

par **Wassim RAMADAN**

Soutenue le : 4 juillet 2011

Composition du jury

Rapporteurs	Congduc PHAM	Professeur à l'Université de Pau
	Toufik AHMED	Professeur à l'École Nationale Supérieure d'Électronique, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux
Examinateur	Pascal LORENZ	Professeur à l'Université de Haute-Alsace
Directeur de Thèse	Julien BOURGEOIS	Professeur à l'Université de Franche-Comté
Co-directeur de Thèse	Eugen DEDU	Maître de conférences à l'Université de Franche-Comté

À celle qui a tout sacrifié pour moi, mon épouse et mon amour Nisreen

À celui qui a tout changé à mes yeux, mon petit ange Yosseph

Remerciements

C'est avec un grand plaisir que je réserve ces lignes en témoignage de ma reconnaissance à tous ceux qui m'ont accompagné, soutenu et aidé pour accomplir les travaux présentés dans ce manuscrit.

J'exprime mes profonds remerciements à M. Julien Bourgeois, mon directeur de thèse, qui a bien accepté de superviser mes travaux. Ensuite, à M. Eugen Dedu, qui a encadré mes travaux pendant cette thèse. Je tiens à lui exprimer ma gratitude pour les pistes de réflexion qu'il m'a fait prendre.

J'adresse aussi mes remerciements à M. Congduc Pham et à M. Toufik Ahmed pour m'avoir fait l'honneur d'accepter d'être rapporteurs de cette thèse. Que M. Pascal Lorenz soit également remercié pour avoir accepté d'être membre de mon jury.

Je suis très reconnaissant envers M. Jean-Laurent Hippolyte et Mlle Chloé Balzinge qui m'ont aidé à la relecture de ce document.

Je ne manquerais pas de remercier l'ensemble des doctorants à Montbéliard Wahabou, Soumaya, Bogdan, Matteo et mes collègues de bureau Mouhannad et Raheel.

J'exprime aussi ma gratitude à l'ensemble des membres de l'équipe OMNI (Optimisation, Mobility, Networking) de Montbéliard.

Enfin, je tiens aussi à remercier toute ma famille en Syrie pour leur soutien inconditionnel, et surtout mon père et ma mère qui n'ont jamais cessé de m'encourager de si loin.

À tous, mon infinie gratitude ...

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

Première partie

Introduction générale

Chapitre 1

INTRODUCTION

Depuis l'avènement de l'informatique moderne mais plus particulièrement depuis l'avènement de l'informatique grand public dans les années 80, les ordinateurs ont été utilisés pour afficher du contenu multimédia. Au début ces contenus étaient relativement simples puis, au fil du temps, les ordinateurs sont devenus capables d'afficher des contenus de plus en plus complexes. Puis, avec l'arrivée d'Internet, l'idée de transmettre ces vidéos en temps-réel d'une machine à une autre est née. Il restait cependant de nombreux défis à relever. En effet, transmettre des vidéos en temps-réel nécessite des processeurs assez puissants pour supporter le décodage de la vidéo ainsi qu'une bande passante assez élevée pour le transmettre. Cependant, le débit disponible était très limité et la seule solution viable était de télécharger le média pour le visionner ultérieurement.

A partir de 2000, le développement d'Internet s'est accéléré. Le débit disponible est devenu beaucoup plus important avec la popularisation de l'accès à Internet haut-débit. Dans le même temps le nombre de personnes connectées a explosé. De nombreux services de *streaming* ont alors vu le jour. Pourtant, le *streaming* est resté limité aux formats d'encodage bas débit. Les formats d'encodages qui nécessitaient une plus grande bande passante, comme le HD (*High Definition*), n'étaient pas encore utilisés.
Durant cette période de popularité croissante d'Internet, les utilisateurs se connectaient à Internet par des liaisons filaires, ce qui limitait leur mobilité. Des solutions ont été développées, notamment la technologie Wi-Fi qui permet à un dispositif (par exemple un ordinateur, une console de jeux, un smartphone) compatible de se connecter à Internet par le moyen d'un point d'accès sans avoir besoin d'y être relié par un câble (connexion dite sans-fil). Le Wi-Fi permet donc une très grande liberté de mouvement par rapport à une connexion filaire. Mais cette liberté est limitée car la portée du signal entre le mobile et le point d'accès est de l'ordre d'une dizaine ou d'une centaine de mètres.
D'autres solutions s'appuyant sur les technologies de communication mobile comme par exemple la 3G ont fait leur apparition afin de pallier à cette limitation de portée et offrir aux utilisateurs une liberté de mouvement supérieure. Ainsi, un utilisateur connecté en 3G a la possibilité d'accéder au contenu d'Internet et d'être joignable n'importe où.
Avec cette évolution vers une plus grande liberté de mouvement/mobilité, presque toute personne a la possibilité d'être connectée à Internet par le biais d'une de ces technologies. De plus,

ces personnes sont, pour la plupart, équipées de dispositifs permettant la production de contenu multimédia, tels que des caméscopes, des appareils photos numériques, des téléphones portables, etc. Ainsi, la production de contenu multimédia est devenue accessible à tout public et à faible coût.
Nous pouvons classer ceux qui produisent du contenu multimédia en deux catégories : les amateurs et les professionnels. Afin de répondre aux attentes différentes de ces deux catégories, les formats d'encodage vidéo se sont multipliés et le contenu produit devient de plus en plus complexe.
Quelle que soit leur catégorie les gens veulent partager leurs contenus avec les autres et bien que le téléchargement soit encore le moyen de partage de contenu vidéo le plus utilisé, il ne répond plus aux attentes des utilisateurs. Le téléchargement prend beaucoup de temps avant que le contenu ne soit disponible et ce contenu n'est disponible que sur un seul poste à moins de le télécharger de nouveau. Enfin, le contenu téléchargé nécessite, en fonction du format de la vidéo et sa durée, des espaces de stockages très importants et coûteux.
Les utilisateurs désirant un accès direct au contenu partagé tendent vers un nouveau type de services vidéo, c'est la transmission vidéo en continu (*streaming*). Le *streaming* permet de restituer presque instantanément le contenu vidéo et de démarrer le visionnage dès que les premières secondes sont téléchargées. De plus, ce mode de transmission permet de se passer d'un grand espace de stockage car le contenu est stocké en partie seulement sur le disque dur, tandis que le visionnage se fait au fur et à mesure. Également, si l'utilisateur souhaite arrêter la transmission, seul le contenu visualisé a été transféré sur le réseau et non pas toute la vidéo. L'utilisation du streaming a énormément augmenté ces dernières années renforcée par un dispositif législatif interdisant le téléchargement de contenu sous licence mais ne se prononçant pas sur le *streaming*. Les sites officiels de streaming les plus utilisés depuis la France sont sans conteste Dailymotion, lancé depuis 2005, et Youtube, lancé en France depuis 2007.

Les données vidéo ont besoin d'un protocole de transport pour être acheminées entre la source et la destination. La conception d'un tel protocole doit prendre en compte la contrainte temps réel qu'impose le visionnage en direct du contenu. UDP (*User Datagram Protocol*) répond à cette exigence mais il ne garantit pas l'acheminement des données et c'est à l'application réceptrice de détecter les pertes de données et de les restituer. De plus, il est dépourvu d'un mécanisme de contrôle de congestion, ce qui rend ses paquets non-prioritaires dans les files d'attente des routeurs. TCP (*Transmission Control Protocol*), qui garantit l'acheminement de chaque bit du média transmis, a d'autres inconvénients. Il bloque, par exemple, l'application réceptrice lorsqu'il y a des données perdues afin de procéder à leur retransmission. D'autres protocoles de transport tels que SCTP (*Stream Control Transmission Protocol*) et DCCP (*Datagram Congestion Control Protocol*) ont été conçus afin de pallier ces problèmes.
Nous nous sommes intéressés aux problèmes que posent les réseaux sans-fil au *streaming*. D'une part, tous ces protocoles ont été développés pour les réseaux filaires. Bien que leur fonctionnement soit acceptable sur un réseau filaire, ils souffrent, comme tous les protocoles de transport ayant un mécanisme de contrôle de congestion, des erreurs de transmission sur les réseaux sans-fil (pertes), car ils ne les traitent pas convenablement. En effet, les protocoles de transport considèrent toute perte de données comme une perte de congestion, et donc ils réagissent en réduisant de moitié le débit de transmission alors qu'ils ne devraient pas le faire pour les pertes dues aux erreurs du canal sans-fil.
D'autre part, les caractéristiques du *streaming* ne sont pas compatibles avec les réseaux sans-fil. D'un côté, le *streaming* impose la contrainte temps réel avec un débit variable de données. De

l'autre côté, les réseaux sans-fil eux aussi ont une bande passante variable en fonction de la portée du mobile de sa station de base.
Cette incompatibilité entre le *streaming* et le sans-fil, ainsi que le mauvais fonctionnement des protocoles de transport sur réseaux sans-fil imposent qu'il y ait un traitement spécifique des transmissions sur ce genre de réseaux.
Les solutions proposées devraient être générales pour avoir la possibilité d'être applicables indépendamment de l'architecture actuelle. Pour déployer des solutions efficaces pour le transport des médias modernes de façon interopérable et universelle, il faut concevoir un moyen adéquat de transport des données multimédia.

Les travaux qui tentent d'atteindre cet objectif sont divisés en deux catégories. La première catégorie s'intéresse plutôt au moyen de transmission et tente de l'optimiser afin qu'il soit plus efficace sur les réseaux sans-fil. Certains proposent qu'on ajoute un agent spécial dans les points d'accès. Cet agent a pour rôle de distinguer les deux parties de la connexion : la partie de la connexion sur le réseau filaire et celle sur le réseau sans-fil. Un traitement spécifique pour les pertes sur chaque partie est effectué, par exemple [BW04b] [ZN00] [BB95]. D'autres proposent plutôt d'utiliser les informations de la connexion de bout-en-bout afin de réaliser ce traitement et d'empêcher le contrôle de congestion de diminuer le débit si les pertes ne sont pas dues à la congestion, comme par exemple [PSJ06] [CHL06] [HCR$^+$05].
La deuxième catégorie s'intéresse directement au traitement du contenu vidéo. Une solution basique est de télécharger une partie de la vidéo. Lorsqu'il y a suffisamment de données, le visionnage de ces données commence, puis s'arrête en attendant que d'autres arrivent. Autrement dit, cette solution consiste à alterner lecture et pause durant le visionnage. Mais en réalité, cela constitue une forme de téléchargement qui ne prend pas en compte l'aspect temps réel du *streaming*. D'autres solutions existent comme les travaux sur l'optimisation de l'encodage vidéo afin qu'il soit plus adapté aux transmissions sur différents types de réseaux.
La solution la plus efficace devra transmettre la vidéo en continu d'une manière adaptative et utilisera forcément une approche (*cross-layer*) pour connaître les conditions du réseau. Dans la littérature, les solutions qui utilisent ce genre d'approches sont soit développées pour des formats spécifiques [ETHQ09] [GT10], soit demandent des modifications importantes sur plusieurs couches en même temps, ce qui rend leur application difficile [LC08] [KZ07].

Les travaux présentés dans ce manuscrit portent sur l'amélioration de la transmission de contenu vidéo et de données dans les réseaux sans-fil. Nos contributions se déclinent en deux volets : une amélioration des protocoles de transport sur les réseaux sans-fil par la différenciation de pertes et une amélioration du transfert du contenu multimédia par adaptation de la vidéo au niveau de la couche application.
Le premier volet traite le problème des pertes dues aux erreurs du canal sans-fil qui ne sont pas traitées convenablement. En effet, comme nous l'avons déjà mentionné, les protocoles de transport considèrent toute perte de données comme due à la congestion alors que dans les réseaux sans-fil elles ne peuvent avoir d'autres causes. Nous proposons dans cette thèse une méthode permettant de différencier les pertes de congestion des pertes sans-fil et de les traiter de manière adéquate. Cette méthode est appelée RELD (*RTT ECN Loss Differentiation*). RELD est basé sur le RTT (*Round Trip Time*) et la signalisation ECN (*Explicit Congestion Notification*) afin de prendre la décision de différenciation.
Le deuxième volet s'intéresse à la transmission de la vidéo en continu de manière adaptative. Pour le *streaming* vidéo sur un réseau qui a un débit variable, l'adaptation de contenu est un

moyen d'adapter le bitrate de la vidéo transmise aux caractéristiques du réseau, c'est-à-dire qu'elle aligne le bitrate au débit disponible. Elle permet ainsi d'améliorer la qualité de la vidéo reçue par l'utilisateur final. La méthode d'adaptation de la vidéo que nous proposons (VAAL, *Video Adaptation at Application Layer*, « Adaptation de la vidéo au niveau de la couche application ») utilise le débordement du tampon du protocole de transport comme solution pour découvrir le débit disponible et adapter le bitrate de la vidéo transmise au débit disponible estimé en conséquence.
VAAL peut révéler certains inconvénients. En effet, lorsque le débit disponible est situé entre deux bitrates, la qualité alterne constamment entre eux. Afin d'éviter cet inconvénient et d'améliorer la qualité de la vidéo transmise, une deuxième partie de ce volet traite ce problème. La solution proposée est générale car elle peut être utilisée par d'autres méthodes d'adaptation de la vidéo. Elle est appelée ZAAL (**Z**igzag **A**voidance **AL**gorithm). ZAAL utilise une valeur indiquant la capacité de réussite de chaque bitrate et donc s'il est potentiellement supérieur à la capacité du réseau ou non. Cette valeur est utilisée par la suite pour décider si une qualité supérieure peut être transmise ou non.

Cette thèse a donc pour objectif d'optimiser la transmission de données sur les réseaux sans-fil et acheminer efficacement et d'une manière adaptative des données vidéo sur différents types de réseaux. Nous donnons une grande importance au déploiement en réalité de nos solutions. Par conséquent, nos solutions doivent être générales et simples à implémenter. De plus, elles assurent un bon fonctionnement en utilisant les informations disponibles sur plusieurs couches avec un minimum de changement sur l'architecture actuelle.

Plan du document

Cette thèse est structurée en deux parties traitant les deux volets de nos contributions :

- **amélioration des protocoles de transport sur les réseaux sans-fil par la différenciation de pertes :** cette partie est composée de trois chapitres. Le premier chapitre 4 présente l'état de l'art sur les méthodes utilisées pour différencier les pertes de congestion des pertes sans-fil.
  Le chapitre suivant 5 montre l'impact des pertes de congestion et sans-fil sur le RTT, ce qui aide à trouver une formule pour les différencier ; il présente également RELD comme une nouvelle méthode de différenciation de pertes.
  Le troisième chapitre 6 introduit l'environnement de simulation, en particulier le modèle d'erreur sans-fil utilisé et évalue les performances de RELD par la simulation.
- **amélioration du transfert du contenu multimédia par adaptation de la vidéo au niveau de la couche application :** cette partie est composée de cinq chapitres. Le premier chapitre 8 présente des travaux antérieurs sur les méthodes d'adaptation de la vidéo. Le deuxième chapitre 9 présente d'abord un aperçu général des solutions possibles afin d'effectuer une adaptation de la vidéo. VAAL ainsi que son implémentation sur GNU/Linux sont expliqués en détail par la suite dans la section 9.3.
  Le chapitre 10 commence par introduire la plateforme de réseau réel utilisée pour effectuer les expérimentations, le modèle de limitation du trafic employé et les différents scénarios réalisés. Les performances de VAAL sont aussi évaluées dans la section suivante 10.2 qui, premièrement, montre l'adaptation réalisée en utilisant VAAL, et deuxièmement compare VAAL à une transmission classique avec des bitrates statiques sans adaptation.

Le quatrième chapitre 11 décrit un problème (« Zigzag Quality Switching ») résultant de l'utilisation de certaines méthodes d'adaptation de la vidéo telle que notre méthode VAAL ; en particulier les méthodes effectuant une adaptation de la vidéo avec de multiples bitrates ou bien avec des vidéos encodées en plusieurs couches. ZAAL est expliqué dans la section 11.3, puis il est utilisé afin de débarrasser VAAL du problème de zigzag et de le rendre plus robuste et fonctionnel. Enfin, une évaluation de ZAAL est faite dans la section 11.4.

Ce manuscrit s'achève par une conclusion synthétisant nos contributions et mettant en relief les perspectives de recherches menées dans cette thèse. Il est à noter que tous les sigles sont répertoriés dans un glossaire se trouvant à la fin du présent document.

Chapitre 2

PRÉ-REQUIS

Nous présentons dans ce chapitre les connaissances qui vont aider à la compréhension de cette thèse.

2.1 Contrôle de congestion

L'effondrement congestif est une situation qu'un réseau informatique peut atteindre, quand peu ou pas de communications aboutissent à cause de la congestion. La congestion se produit généralement dans les goulots d'étranglement du réseau, où le débit des données entrantes dépasse le débit disponible sortant. Lorsqu'un réseau est dans cet état, la demande de trafic est élevé, mais peu de débit utile est disponible. Des niveaux élevés de retard sont constatés et des pertes de paquets sont causées par les routeurs qui les rejettent parce que leurs files d'attente sont pleines. La qualité générale du service est donc extrêmement mauvaise.

Réduire la congestion a été identifié comme un problème dès 1984 (RFC 896 [Nag84]). Les premières tentatives de réduire la congestion sont apparues avec la mise en œuvre du mécanisme de contrôle de congestion Van Jacobson entre 1987 et 1988. Le contrôle de congestion est donc un mécanisme qui permet d'adapter le débit d'un ou de plusieurs flux au débit disponible dans un réseau.

La plupart des mécanismes de contrôle de congestion utilisent des méthodes dites de bout-en-bout afin d'estimer le débit disponible. Le protocole le plus utilisé actuellement est TCP. Il existe de nombreuses versions de TCP dont la plus utilisé est TCP NewReno.

TCP est connu pour ne pas être bien adapté aux transmissions des données multimédia. C'est pourquoi d'autres protocoles spécifiques à ce genre de transmissions existent tels que UDP et DCCP. Le point commun entre ces deux protocoles est la non retransmission des paquets perdus sachant qu'elles ne sont pas toujours utiles.

2.2 Principe d'ECN

ECN (*Explicit Congestion Notification*) est une extension d'IP (Internet Protocol) définie dans la RFC3168 [RFB01]. ECN nécessite pour son fonctionnement un gestionnaire actif de file d'attente (Active Queue Management (AQM)) comme par exemple RED (décrit dans la section suivante), et supporte une notification de congestion de bout en bout sans perte de paquets. Son utilisation est facultative et elle n'est utilisée que lorsque les deux extrémités d'une même connexion souhaitent le faire.

ECN utilise les deux bits les moins importants (plus à droite) bits du champ DiffServ dans l'en-tête IP. Cela donne la possibilité d'avoir quatre valeurs qui sont :
- 00 : non ECN-Capable
- 10 et 01 : ECN Capable - ECT
- 11 : congestion rencontrés - CE

Les flux voulant utiliser ECN modifient l'en-tête de la couche transport DCCP au début de la transmission afin de négocier leurs volontés de l'utiliser. Si la source et la destination acceptent son utilisation, les paquets envoyés entre elles seront marqués ECN capable (ECT 01 ou 10) dans l'en-tête IP.

Quand un routeur compatible ECN entre en congestion, il met à jour les deux bits à 11 dans l'en-tête IP du paquet, que cela soit un paquet de donnée ou un accusé de réception, pour indiquer une congestion imminente. En réponse à ce signal ECN, le destinataire du paquet acquitte le paquet en indiquant les paquets marqués. En détectant la présence d'ECN dans l'acquittement, la source réagit, à son tour, en réduisant son débit comme si c'était un paquet perdu. Si l'acquittement en revanche est marqué lui aussi la source renégocie avec la destination le taux avec lequel les accusé de réception seront envoyés en diminuant ce taux.

ECN sera utilisé également par notre méthode de différenciation de pertes RELD, présentée et expliquée dans le chapitre 5, en tant que facteur supplémentaire de différenciation entre les pertes de congestion et sans-fil.

2.3 Gestion active de la file d'attente : RED

De nos jours, les routeurs implémentant une stratégie active de gestion des files d'attente comme RED (*Random Early Detection*) sont nombreux. L'utilisation de RED conduit à un meilleur partage entre les différents flux passant par un routeur. Il permet également de gérer la congestion sur le réseau en fournissant des informations de retour négatives « *feed-back* » aux émetteurs par le rejet aléatoire des paquets de sa file d'attente lors d'une congestion imminente. La RFC3168 [RFB01] propose de marquer ces paquets au lieu de les rejeter. Ainsi, si l'utilisation d'ECN est activée dans le routeur et que le flux est compatible avec ECN, RED marque ces paquets au lieu de les rejeter.

Pour ce faire, RED utilise nombreuses valeurs : la taille de la file d'attente q_l ; la moyenne de la file d'attente q_{ave} ; le seuil minimum de la file d'attente q_{th_min} et le seuil maximum de la file d'attente q_{th_max}. Le traitement du paquet dépend de q_{avg} :
- Si $q_{ave} < q_{th\_min}$, tous les paquets passent sans rejet ou marquage ECN.
- Si $q_{th\_min} < q_{ave} < q_{th\_max}$, les paquets sont marqués ou rejetés avec une probabilité qui augmente avec l'augmentation de q_{ave}.
- Finalement, quand $q_{ave} > q_{th\_max}$ tous les paquets sont rejetés.

2.4 Simulateur de réseau : ns2

Le simulateur de réseaux ns2 [Net] fournit au monde de la recherche un support pour simuler un réseau avec différents protocoles de transport, protocoles de routage, gestions de files d'attente et d'autres applications sur les réseaux filaire et sans-fil.

ns2 a été conçu de sorte que le code offre ré-utilisabilité et modularité. L'utilisation du simulateur ns2 est simple et flexible. De plus, les résultats des simulations sont reproductibles, ce qui permet de bien décrire des événements apparus pendant la simulation et d'en tirer des conclusions. Cela offre également la possibilité de vérifier très facilement l'influence d'un changement du code sur les résultats.

ns2 est un simulateur à événements discret écrit en C++ avec le langage orienté objet Tcl (OTcl) pour configurer les objets. Ces objets sont des nœuds, des liaisons, des agents ou des applications. Les nœuds et les liaisons définissent la topologie du réseau. Les agents représentent les points finaux, tandis que les applications sont des sources de trafic qui envoient et reçoivent les données.

Le simulateur réseau ns2 sera utilisé dans la partie II de cette thèse afin de faire des analyses sur les pertes sans-fil. Ensuite, il sera utilisé comme outil de validation de notre contribution de différenciation de pertes RELD.

2.5 Réseaux sans-fil

Un réseau sans-fil est un réseau informatique qui connecte différents postes entre eux par des ondes radio. Il est composé d'un point d'accès AP (*Access Point*) auquel sont connectées plusieurs stations mobiles. L'AP est généralement connecté lui-même à Internet par le moyen d'un réseau filaire.

Les liaisons sans-fil ne sont pas aussi robustes que les liaisons filaires. Le débit disponible est souvent moins important et les erreurs de transmission sont plus fréquentes. De plus, la puissance de signal change en fonction de la distance et les obstacles entre le mobile et le point d'accès. Finalement, les ondes radio souffrent d'interférences qui résultent de l'influence de l'environnement ou de bandes de fréquences très proches. Ils soufrent également de l'effet des chemins multiples.

Dans le but de protéger les protocoles des couches de haut niveau des erreurs de transmission, la correction d'erreur et la retransmission peuvent être utilisées dans les couches de bas niveau aussi. Dans ce but, un protocole de type ARQ (Automatic Repeat reQuest) est défini pour continuer à retransmettre des segments jusqu'à ce qu'ils soient acquittés ou qu'un nombre élevé de retransmission soit atteint.

Le Wi-Fi est une technologie de réseau informatique sans-fil mise en place pour fonctionner en réseau interne. Il est devenu un moyen d'accès à haut débit à Internet. Il est basé sur la norme IEEE 802.11 [CWKS97]. C'est un réseau qui fonctionne en deux modes : Ad-Hoc et infrastructure. Nous nous intéressons au mode infrastructure qui est composé d'un ou plusieurs AP interconnectés généralement entre eux par un réseau, normalement filaire. Dans la norme standard de IEEE 802.11, l'algorithme utilisé pour l'accès au média, CSMA/CA (*Carrier Sense Multiple Acces with Collision Avoidance*), est similaire à celui utilisé dans les réseaux filaires CSMA/CD (avec *Collision Detection*).

Les principaux problèmes d'une communication sur un réseau sans-fil se résument somme suit :

- la mobilité engendre une rapidité de déconnexion et de reconnexion en fonction de la distance entre le mobile et le point d'accès ainsi que les obstacles qui les séparent. En conséquence, un grand nombre de ruptures de signal sont constatées sur ces réseaux ;
- le changement de chemin entre l'émetteur et le récepteur en cas de changement d'AP peut causer un taux élevé de paquets acheminés en désordre ;
- le taux d'erreur dans les réseaux sans-fil est élevé, ce qui cause généralement des pertes aléatoires de 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 très étroites ;
- la contention au niveau de la couche MAC (*Mac layer contention*) qui a un impact sur l'augmentation du délai d'acheminement des paquets (RTT). La période de contention (*Contention period*) est une durée aléatoire attribuée à chaque nœud à la fin de chaque transmission sur le canal sans-fil. Cela sert à donner un accès égal à tous les nœuds voulant utiliser le canal.

2.6 Utilisation du contrôle de trafic sur Linux

TC (*Traffic Control*) est un outil de contrôle de trafic intégré dans le noyau Linux. Il permet de spécifier la limite de bande passante pour certains utilisateurs pendant un certain temps.

Le contrôle de trafic peut se faire de deux manières. La première concerne le trafic sortant et est appelée « limitation du trafic » (*traffic shaping*). Elle est généralement réalisée du côté source (serveur). La seconde concerne le trafic entrant et est appelée « régulation du trafic » (*traffic policing*). Elle se fait généralement au niveau du récepteur (client).

La limitation et la régulation du trafic sont effectuées par l'ajout d'un gestionnaire de la file d'attente (Qdisc *queueing discipline*) avec un débit de données spécifique assigné à l'interface réseau concernée. À chaque fois qu'un paquet est envoyé par le noyau à l'interface correspondante, il est d'abord mis dans la file d'attente de son Qdisc. Ensuite, si le débit des paquets livrés par le noyau est plus petit que le débit spécifié au Qdisc, le paquet est livré à l'interface pour être envoyé sur le réseau. Les paquets qui arrivent à un taux plus élevé sont retardés, c'est-à-dire qu'ils sont ajoutés au tampon du Qdisc pour être acheminés plus tard. Ils sont en revanche supprimés si la taille du tampon est atteinte.

Il y a de nombreux types de gestionnaires de limitation du trafic. L'un d'entre eux est HTB (*Hierarchical Token Bucket*) qui est basé sur une classe complète, c'est-à-dire qu'il peut utiliser des filtres, de type TBF (*Token Bucket Filter*). TBF est un Qdisc simple qui permet de transmettre les paquets arrivant à un certain taux défini. Au contraire, la régulation du trafic n'a qu'un seul gestionnaire qui est le Qdisc appelé *Ingress policer*. Il permet d'appliquer des filtres sur les paquets entrant dans l'interface réseau avant d'arriver à la pile IP.

L'utilisation des gestionnaires Qdisc est très simple. La première étape consiste à choisir un type de Qdisc et à le joindre à l'interface réseau concernée. La deuxième étape nécessite l'ajout d'un filtre au Qdisc lui-même. Ce filtre spécifie les paquets qui peuvent passer par le Qdisc, par exemple par leur taux d'arrivé ou par leur type (paquets TCP, UDP, etc.) La troisième et dernière étape est d'activer le Qdisc. De plus, pour le Qdisc de type *Ingress policer*, une valeur indiquant le taux de paquets en rafale possible doit être fournie pour atteindre le débit désiré.

Nous avons testé ces deux possibilités mais quelques problèmes sont apparus. Pour la limitation du trafic sur l'expéditeur, à chaque fois qu'un paquet est rejeté par le Qdisc, DCCP envoie un nouveau paquet immédiatement, ce qui fausse le taux d'envoi de DCCP; en réalité, il aurait dû attendre un certain temps (régulé par son contrôle de congestion) avant d'envoyer un nouveau paquet. Cela empêche notre méthode VAAL de fonctionner correctement, parce que les paquets ne sont plus rejetés du tampon du socket DCCP. Par conséquent, la valeur du WFP est toujours zéro (voir le chapitre 9 pour voir le fonctionnement de VAAL). Cela s'explique par le fait que le taux effectif d'envoi de DCCP est largement plus grand que le taux d'envoi calculé par DCCP. Comme solution à ce problème, nous avons utilisé pendant nos expérimentations une troisième machine entre le serveur et le récepteur et nous avons placé sur celle-ci la limitation du trafic. Cette solution est de toute façon la méthode conseillée par les développeurs DCCP qui mettent en garde les utilisateurs de ne pas mettre la limitation du trafic sur la même machine que l'émetteur.

Dans le cas de la régulation du trafic, le taux de données spécifié n'a jamais été stable. Il a souvent été dépassé en raison de la valeur du rafale. Une petite valeur fait que le taux désiré n'est pas atteint. Une grande valeur amène à un dépassement important du débit spécifié. Nous n'avons donc pas utilisé la régulation de trafic.

TC sera utilisé dans le chapitre 10 de cette thèse afin de limiter le trafic du réseau employé pour faire des expérimentations et valider nos contributions.

2.7 Modèles de propagation existants

De nombreux modèles existent, chacun ayant ses points forts et ses points faibles. Selon le contexte, il faut être particulièrement attentif lors du choix du modèle utilisé. La liste suivante présente les principales familles de modèles qui peuvent être trouvées dans la littérature :

- Les modèles utilisant une équation d'atténuation continue du signal radio. Ils ne prennent en compte que la distance entre le mobile et le point d'accès. Le modèle proposé par *Friis* par exemple [Fri46] gère la propagation dans des environnements complètement libres d'obstacles. D'autres modèles proposent une atténuation plus rapide du signal selon la distance, tels que le *tworayground* qui considère une propagation à deux trajets entraînant une auto-altération du signal reçu.
- Les modèles dérivés des précédents mais y intégrant une sorte de *fast-fading* provoquant des pertes de paquets. Certains utilisent le modèle de *Gilbert-Elliot* (une chaîne de Markov pour déterminer si un paquet doit être rejeté ou non en fonction de l'état actuel du lien sans-fil). Ce modèle cause une variation très rapide de la qualité du lien. Un autre modèle dérivé, appelé *shadowing*, propose un facteur aléatoire de la qualité de chaque paquet en plus de l'atténuation provoquée par la distance. Pour chaque paquet, il y a une probabilité de rejet qui augmente avec la distance. Le réglage des paramètres d'un tel modèle peut être fait en utilisant des données empiriques obtenues à partir de mesures réelles (par exemple, le facteur d'atténuation est élevé dans l'environnement intérieur, moyen dans une ville et bas dans un champ ouvert : l'écart-type peut également être obtenu à partir des expérimentations). Dans [YEY+04], un modèle multi-chemin a été défini et combiné avec un effet doppler afin de calculer le taux d'erreur par bit (BER *Bit Error Rate*) en fonction du rapport signal/bruit et de la vitesse.

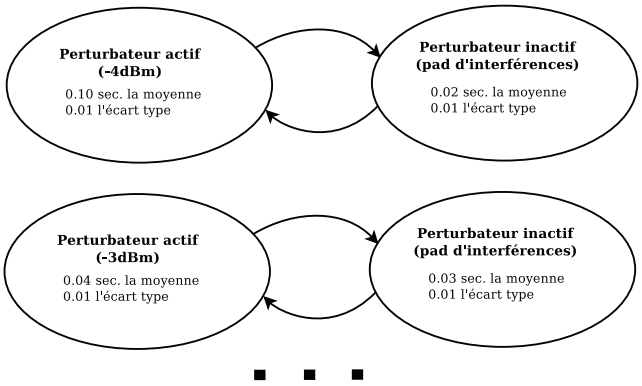


Figure 2.1 – Le graphique d'état des deux types de perturbateurs

- Des modèles plus complexes existent. Ils nécessitent une modélisation de l'ensemble de l'environnement (ou du moins les parties les plus pertinentes de celui-ci) avec ses obstacles. Les techniques de type *Raytracing* sont utilisées pour calculer le niveau du signal en fonction de la position relative de l'émetteur et du récepteur dans cet environnement complexe. Les algorithmes utilisés nécessitent des calculs intensifs, mais peuvent être pré-calculées pour un environnement donné [SHR05].
- Même les algorithmes de propagation les plus avancés sont disponibles. Ils n'utilisent pas seulement de nombreux obstacles, mais ils en font aussi plusieurs compositions ainsi que des propriétés de leur surface. Cela influence beaucoup sur la propagation du signal radio. Avec ces modèles ce ne sont pas seulement les occlusions qui sont causées par des obstacles modélisées, mais aussi les réfractions et les diffractions. Cela permet des calculs détaillés de propagation pour les environnements intérieurs [DLRRJRG05].
- Enfin, certains modèles ont une approche différente et utilisent essentiellement des données réelles collectées [EDS05]. Leur principal avantage est de proposer des valeurs très réalistes, mais qui sont bien sûr fortement liées à un environnement particulier.

Shadowing-pattern

Ce modèle reproduit la réalité dans laquelle différents éléments dans l'environnement ont des effets cumulatifs sur la puissance du signal au niveau du récepteur. Il se sert de ce qu'on appelle perturbateurs, qui sont une abstraction des éléments du monde réel qui ont un impact sur la puissance du signal. Ces éléments peuvent être aussi variés que les personnes ou les voitures qui passent, l'ouverture et le clôture des portes, les autres nœuds envoyant des données sur le canal radio, etc. Les perturbateurs affectent une zone limitée et configurable de l'environnement virtuel. Ils alternent entre deux états, actif et inactif.

- Dans l'état actif, les perturbateurs affectent (c'est-à-dire généralement ils réduisent) la force de n'importe quel signal reçu par un nœud à l'intérieur de la zone d'effet du perturbateur.
- Dans l'état inactif, ils n'ont aucun effet du tout.

Chaque perturbateur est donc défini par le temps et l'écart-type du temps passé dans les deux états, ainsi que sa puissance lorsqu'il est actif. La figure 2.1 montre le graphique d'état

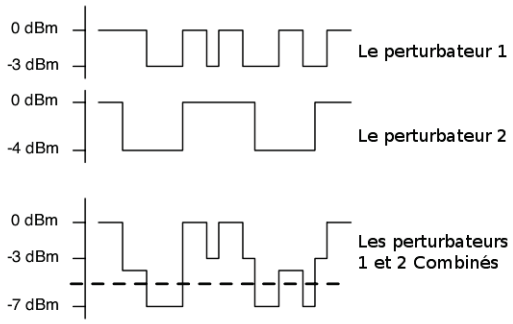


Figure 2.2 – Les effets des perturbateurs au fil du tempos et leurs combinaisons

des deux types de perturbateurs. Quand un perturbateur est à l'état actif, son effet (en dBm) est simplement une sorte de modificateur qui est ajouté au signal au niveau du récepteur. Ce dernier prend déjà en compte la distance, étant lui-même calculé par une équation de type Friis. La figure 2.2 montre comment les effets des perturbateurs individuels évoluent dans le temps et comment ils peuvent être combinés pour parfois dépasser leur seuil individuel.

Comme expliqué dans [DRS06], cette technique ne vise pas à une modélisation précise d'un environnement particulier. Il vise plutôt à produire des comportements extrêmement réalistes statistiquement, notamment en terme de distribution des pertes au fil du temps. Il est également facile à configurer et ne nécessite que des calculs simples.

En fait, lorsque le modèle *shadowing-pattern* est utilisé, il suffit de choisir un ensemble de *perturbateurs* et de configurer leurs paramètres. Comme leurs effets se combinent, un ensemble de quelques-uns (2–5) *perturbateurs* est généralement suffisant pour décrire un environnement tout à fait réaliste. Et comme toute moyenne et écart type ne peuvent être donnés, il peut être utilisé pour modéliser des phénomènes du monde réel de toute échelle de temps.

Le *shadowing-pattern* sera utilisé dans les chapitres 5 et 6 comme modèle de propagation sur le réseau sans-fil.

Deuxième partie

Amélioration des protocoles de transport sur les réseaux sans-fil par la différenciation de pertes

Chapitre 3

INTRODUCTION

Le nombre de stations mobiles augmente de façon considérable ces dernières années. Les protocoles de transport fonctionnent moins bien sur les réseaux sans-fil que sur un réseau filaire. Or, les services du *streaming* vidéo sur Internet sont actuellement présents partout et accessibles par tout le monde. Cela crée un besoin important d'améliorer le fonctionnement de ces protocoles de transport sur les réseau sans-fil et en particulier pour les applications ayant des contraintes spécifiques, tels que le délai et la stabilité du débit du transfert.

De nos jours, les réseaux sans-fil sont largement déployés et sont couramment utilisés pour accéder aux services sur Internet. Comparés aux réseaux filaires, les réseaux sans-fil sont inférieurs en terme de performance (vitesse et qualité de service) [BSK95, BPSK97]. De plus, les pertes dans les réseaux filaires sont principalement dues à la congestion dans les routeurs. Car en cas de congestion, les routeurs rejettent généralement des paquets reçus lorsque leurs files d'attente sont, ou presque pleines. Ces paquets rejetés sur le chemin, au niveau d'un routeur intermédiaire entre un émetteur et un récepteur, sont par la suite signalés par ce dernier au premier en utilisant des accusés de réception (ACKs) en tant que paquets perdus. Mais tout ceci est différent dans les réseaux sans-fil où les pertes se produisent souvent pour diverses autres raisons. En effet, elles peuvent se produire à cause d'une mauvaise qualité de la liaison sans-fil si par exemple la distance est élevée entre l'appareil mobile et la station de base à la quelle il est connecté, ou à cause des interférences.

Cette différence entre les deux types de réseaux, filaires et sans-fil, est bien connue et elle est déjà prise en compte dans la conception des caractéristiques physiques des réseaux sans-fil. En effet, IEEE 802.11 [CWKS97] inclut des mécanismes pour lutter contre les nombreuses pertes sur un lien radio. Les appareils sans-fil sont conçu de sorte qu'un paquet, envoyé entre deux terminaux sans-fil, soit retransmis au niveau de la couche MAC un certain nombre de fois (6 par exemple). Toutefois, en cas de longue perturbation, il arrive qu'un paquet puisse être perdu 7 fois de suite sur un lien sans-fil. Dans ce cas, l'appareil émetteur sans-fil le rejette et il sera par la suite remarqué à la couche transport du récepteur. Nous nous sommes intéressés au traitement de la perte des paquets sur les réseaux sans-fil au niveau de la couche transport.

Malgré cette prise en compte des caractéristiques des liaisons radio par les concepteurs des technologies sans-fil, les performances des protocoles de transport sur réseaux sans-fil sont tou-

jours dégradées. Ceci est due à la manière dont le protocole TCP (*Transmission Control Protocol*) [Pos81] réagit aux signalements de pertes. TCP, qui est communément utilisé par les applications Internet et initialement conçu pour les réseaux filaires, classe toute perte de données comme une perte de congestion, et donc il réagit en réduisant de moitié le débit de transmission. Toutefois, comme vu ci-dessus, cette supposition n'est plus nécessairement vraie. Dans la littérature, il y a de nombreuses propositions sur la manière dont la performance du protocole de transport peut être optimisée sur les réseaux sans-fil. L'idée principale est que **les protocoles de transport devraient réduire leur débit de transmission seulement dans le cas de la congestion et non pas si des paquets sont perdus pour d'autres raisons** [BPSK97, BW04b, BV99, BK98].

Le contrôle de congestion est un aspect crucial de TCP. Il l'utilise afin d'atteindre la plus haute performance et d'éviter l'effondrement du réseau à cause de la congestion (*congestion collapse*), où la performance du réseau peut énormément chuter. Un mécanisme de contrôle de congestion maintient le débit entrant dans le réseau en dessous d'un certain taux qui peut provoquer l'effondrement. Le récepteur informe l'émetteur de la réception de chaque paquet de donnée en utilisant des accusés de réception (ACKs). L'émetteur, à son tour, peut déduire l'état du réseau à partir de ces ACKs ou, parfois, par leur manque. Il augmente ainsi son débit si les conditions réseau sont bonnes. Sinon, il fait le contraire. Ce mécanisme utilise principalement deux algorithmes liés: le mode de démarrage lent (*Slow Start* (SS)) et le mode d'évitement de la congestion (*Congestion Avoidance* (CA)). En mode de démarrage lent, la source augmente le taux d'envoi de façon exponentielle par rapport au temps. Par contre, en mode d'évitement de congestion ce taux est augmenté de façon linéaire. À chaque fois que la source détecte une perte, il réduit sa vitesse d'envoi et retransmet les paquets perdus.

Les versions originales de TCP utilisent un système cumulatif d'acquittements afin de reconnaître les paquets perdus de données. Dans ce système, si un groupe de paquets est envoyé et que un des premiers d'entre eux est perdu, le récepteur ne peut pas acquitter les autres paquets. Ainsi, l'émetteur ne sait pas si d'autres paquets ont été perdus. Pour pallier à cette faiblesse, un autre système d'acquittements sélectif est proposé, c'est l'option SACK, définie dans RFC 2018 [MMFR96]. Le récepteur est autorisé à accuser la réception d'un bloc de paquets ayant été correctement reçus. Ainsi, la source peut retransmettre plus rapidement les paquets perdus.

Le type d'application réseau utilisé a un impact direct et très important sur le choix du protocole de transport employé et par la suite le système d'acquittements voulu. Certaines applications, notamment pour le transfert de fichiers, ont besoin d'un protocole de transport fiable à 100% comme TCP qui retransmet tous les paquets perdus. En revanche, il y a de plus en plus d'applications sur Internet qui peuvent accepter un certain taux de pertes (ITU.T [IT07]), par exemple les applications temps réel comme le *streaming* multimédia. TCP n'est pas un bon choix pour ce genre d'applications. C'est UDP (*User Datagram Protocol*) [Pos80] qui est donc utilisé. Or, UDP, qui ne présente pas les inconvénients de TCP, n'a ni de contrôle de congestion ni de mécanismes de contrôle de flux.
Afin d'améliorer la qualité d'une transmission multimédia, RTP (*Real-time Transport Protocol*), qui est un protocole de couche application [SCFJ03], a été conçu pour opérer au dessus d'un protocole de transport comme TCP ou UDP. Il permet par exemple au récepteur de réorganiser les paquets reçus grâce au numéro de séquence inclus dans l'en-tête des paquets RTP et de faire la synchronisation. Néanmoins, RTP, comme UDP, ne traite pas les conditions réseau, car il lui manque aussi un contrôle de congestion. Le choix peut se faire actuellement par l'emploi d'un nouveau protocole de transport DCCP (*Datagram Congestion Control Protocol*) récemment

standardisé comme RFC 4340 [KHF06]. Il n'est pas un protocole fiable, ce qui convient aux applications temps réels, mais il permet en revanche l'utilisation d'un mécanisme de contrôle de congestion comme TCPlike [FK06] ou TFRC [FHPW08], ce qui préserve la performance du réseau.

Quelque soit le choix du protocole de transport utilisé, fiable ou non, recourir à un contrôle de congestion s'avère très important afin d'empêcher une sous utilisation de la capacité du réseau. Se servir d'un tel mécanisme signifie que le débit d'envoi doit être réduit en cas de détection de la congestion. De ce fait, les protocoles employant un contrôle de congestion comme DCCP) souffrent du même problème que TCP lorsqu'ils opèrent sur un réseau sans-fil, c'est-à-dire qu'ils considèrent toute perte de données comme causée par la congestion. Ceci mène à une sous-utilisation de la capacité du réseau.

À partir de toutes ces constatations et parce que les réseaux filaires et sans-fil sont souvent utilisés conjointement, il y a un besoin croissant d'un nouveau protocole prenant en compte les propriétés des liaisons sans-fil et les diverses raisons pour la perte de données. Dans cette partie, nous proposons une nouvelle méthode de différenciation de pertes appelée RELD (*RTT ECN Loss Differentiation*). **RELD est basée sur le RTT (*Round Trip Time*) et la signalisation ECN (*Explicit Congestion Notification*) pour différencier les pertes dues à la congestion de celles dues aux erreurs de transmission sur le canal sans-fil**, voir la section 5.3 pour avoir les détails sur son fonctionnement. RELD [RDDB11] est une évolution de notre méthode précédente EcnLD [RDB09b, RDB09a] de différenciation pertes.

Chapitre 4

ÉTAT DE L'ART DES MÉTHODES DE DIFFÉRENCIATION DE PERTES

Un réseau sans-fil est un type de réseaux informatiques permettant la connexion entre différents postes sans être reliés physiquement entre eux par un câble. Il y a de nombreux types de réseaux sans-fil comme par exemple les réseaux satellites, les réseaux cellulaires, les réseaux Wi-Fi qui comportent deux modes de communications : un mode de communication à infrastructure et un autre mode dit Ad-Hoc. Dans cette thèse, nous nous sommes intéressés aux réseaux sans-fil à infrastructure, voir la section 2.5 pour plus d'informations.

Dans la littérature, de nombreuses méthodes ont été proposées pour améliorer les performances des protocoles de transport sur réseaux sans-fil. Dans le contexte de cette thèse, seule la différenciation de pertes nous intéresse. Elle amène la source à continuer à émettre des données à un débit égal ou supérieur si les paquets signalés comme perdus ne sont pas causés par la congestion, c'est-à-dire qu'ils sont perdus pour une autre raison. Cela veut dire que la différenciation de pertes rend l'émetteur capable de distinguer les pertes de congestion des pertes causées par des erreurs sur le canal radio.

Nous pouvons distinguer deux types de méthodes de différenciation de pertes entre les pertes sans-fil et les pertes de congestion, selon qu'elles utilisent des informations explicites ou implicites.

Le premier type (dit explicite) nécessite la mise en place d'un agent intermédiaire spécialisé entre la source et la destination qui est normalement localisé sur la station de base. Cet agent a pour rôle de préciser pourquoi un paquet a été perdu. Ce type d'approches est particulièrement intéressant parce qu'il permet une bonne différenciation de pertes, main il est nécessaire d'apporter des modifications aux équipements réseau existants, en particulier à la station de base. En outre, il nécessite plus de traitement sur les équipements intermédiaires. Pour ces raisons, nous pensons que le déploiement de ce type d'approches est très difficile, voire impossible.

Le deuxième type différenciation de pertes (dit implicite) ne demande pas de changement spécifique à l'infrastructure des réseaux existants. Il utilise en revanche des approches dites de bout-en-bout. En fonction des informations disponibles d'une connexion et grâce à la coopération du récepteur, l'émetteur peut effectuer la différenciation de pertes et ainsi améliorer le débit.

Il existe actuellement trois catégories de méthodes ayant recours à ce type d'approche qui sont basées sur le RTT (*Round Trip Time*), le ROTT (*Relative One-way Trip Time*) et l'IAT (*Inter Arrival Time*). Étant donnée que ce type d'approches ne change pas l'infrastructure du réseau, nous pensons qu'il a plus de chances d'être déployé.

Nous présentons dans cette section TCP Westwood et Westwood+ comme exemples des protocoles qui améliorent la transmission sur les réseaux sans-fil et puis nous détaillons les deux types des méthodes de différenciation.

4.1 Protocoles de transport pour les réseaux sans-fil

4.1.1 TCP Westwood et Westwood+

TCP Westwood est un mécanisme de contrôle de congestion de bout-en-bout [CGM+02]. En effet, c'est une modification de TCP Reno du côté émetteur. Son but est d'améliorer la performance de TCP sur les réseaux filaires et sans-fil. Le plus grand avantage de TCP Westwood est qu'il n'est pas sensible aux pertes aléatoires se produisant pendant une transmission tel que le fait TCP Reno. Il parvient à cet effet en mesurant la taux des acquittements reçus ce qui est utilisé pour calculer le débit disponible. Ce débit est par la suite utilisé pour réajuster la taille de la fenêtre de congestion cwnd et le seuil de la phase de démarrage lent ssthresh (*Slow Start threshold*).

Après réception d'un acquittement, l'émetteur peut tout simplement savoir si oui ou non, il y a eu des pertes et combien de paquets ont été reçus. Dans le cas où il n'y a pas de pertes, l'émetteur calcule le débit disponible BE en faisant la moyenne des données transmises sur le temps et cela en utilisant l'équation suivante : $BE_t = \alpha_t BE_{t-1} + (1 - \alpha_t)\left(\frac{BE_t + BE_{t-1}}{2}\right)$ où t et le temps de réception d'un ACK quelconque et α_t dépend de l'intervalle $BE_t - BE_{t-1}$. Dans le cas contraire, c'est-à-dire qu'il y a eu des acquittements dupliqués DupACKs ou un *time out*, TCP Westwood règle cwnd et ssthresh comme suit :

- en cas de DupACKs :
  1. $ssthresh = (BE * RTTmin)/segsize$
  2. si $(cwnd > ssthresh)$ mettre $cwnd = ssthresh$
- en cas de *time out* :
  1. $cwnd = 1$
  2. $ssthresh = (BE * RTTmin)/segsize$

  où RTTmin est le RTT minimum rencontré pendant la transmission.

TCP Westwood+ [GM04] est une évolution du protocole TCP Westwood. En effet, l'algorithme d'estimation de bande passante Westwood ne fonctionne pas bien en présence de trafic inverse en raison de la compression des acquittements.

Donc, Westwood+ modifie cwnd et ssthresh comme suit :

- en cas de DuppACKs :
  1. $ssthresh = max(2, (BE * RTTmin)/segsize)$
  2. $cwnd = ssthresh$
- en cas de *time out* :

1. $cwnd = 1$
2. $ssthresh = max(2, (BE * RTTmin)/segsize)$

4.2 Méthodes explicites de différenciation de pertes

Ce type de méthodes nécessitent la mise en place d'un agent intermédiaire entre la source et la destination qui est normalement localisé dans la station de base.

4.2.1 *Snoop*

L'idée principale de ce protocole est de mettre en *cache* des paquets au niveau de la station de base et d'effectuer des retransmissions locales à travers la liaison sans-fil [BSK95, BSAK95]. Des modifications sont faites uniquement sur le code du protocole de routage à la station de base afin de transférer des données entre un émetteur fixe et un récepteur sans-fil. Ces modifications sont, d'une part, la mise en tampon des données non acquittées, et d'autre part une politique de retransmissions locale basée sur les acquittements (ACKs) et l'expiration du temporisateur (*timeout*).

Snoop consiste en un module ajouté à la station de base. Il assure le suivi de tous les accusés de réception du récepteur mobile. Quand une perte est détectée par l'arrivée d'un accusé de réception dupliqué, (DupACK), ou par l'expiration du temporisateur local, il retransmet le paquet perdu. Ainsi, la station de base cache la perte de paquets à l'émetteur en supprimant les acquittements dupliqués, ce qui empêche de déclencher le mécanisme de contrôle de congestion inutilement.

Le module de *snoop* a deux procédures principales : « *snoop* data() » et « *snoop* ack() ». *snoop* data() traite les paquets de l'émetteur de la manière suivante :

1. Lorsqu'un paquet (ayant un nouveau numéro de séquence) arrive, il est sauvegardé et transféré au récepteur.
2. Lorsqu'un paquet (ayant un numéro hors-séquence (en désordre) qui a déjà été sauvegardé, représentant une retransmission d'un paquet perdu par congestion) arrive, il est tout simplement transféré au récepteur.
3. lorsqu'un paquet (ayant un numéro hors-séquence qui n'est pas encore sauvegardé) arrive, il est marqué comme congestion dans le tampon de l'AP et puis transféré au récepteur.

snoop ack() en revanche surveille et traite les accusés de réception (ACKs) et effectue différentes opérations selon le type et le nombre d'accusés de réception qu'il reçoit. Ces accusés de réception peuvent tomber dans l'une des trois catégories:

1. Un nouvel ACK
2. Un ACK parasite
3. Un DupACK

Quand un ACK est reçu, il y a deux possibilités : premièrement, si l'ACK est nouveau, son paquet est libéré et puis l'ACK est transféré à l'émetteur. Deuxièmement, si c'est le premier DupACK, le paquet et directement retransmis localement. Dans tous les autres cas, l'ACK est ignoré.

4.2.2 ELN

La notification explicite de perte (ELN, *Explicit Loss Notification*) [BK98] est un mécanisme par lequel la raison de la perte d'un paquet peut être communiquée à l'expéditeur. En effet, avec ELN l'expéditeur est informé que la perte est due à d'autres raisons, non liées à la congestion du réseau (par exemple, en raison d'erreurs sur le réseau sans-fil). Ainsi, les retransmissions des paquets perdus peuvent être dissociées du contrôle de congestion. Si le récepteur (ou une station de base) sait avec certitude que la perte d'un paquet n'est pas due à la congestion, il définit un bit (ELN) dans l'entête TCP et le propage à la source.

ELN met un agent à la station de base afin de surveiller tous les segments TCP qui arrivent sur le lien sans-fil. Contrairement à d'autres méthodes (comme Snoop ou TCP-aware), l'agent ELN ne tamponne aucun paquet car il ne fait pas de retransmissions local, par contre, il conserve la trace des trous dans les numéros de séquences des paquets de données qu'il reçoit. Un trou est un intervalle manquant dans l'espace des numéros de séquences ce qui correspond aux paquets ayant été perdus sur la liaison sans-fil.

Lorsqu'un DupACK arrive à la station de base, ELN consulte sa liste des numéros de séquences. Si le numéro de séquence du paquet acquitté est dans cette liste, l'agent défini le bit ELN dans l'ACK avant de le transférer à l'émetteur. Quand la source reçoit un ACK avec le bit ELN activé, elle retransmet le paquet mais ne prend aucune mesure de contrôle de congestion en considérant la perte comme due au canal sans-fil.

Toutefois, il se pourrait également que le paquet ait été perdu en raison de la congestion dans la file d'attente de la station de base. C'est pourquoi et afin d'éviter le marquage à tort d'une perte de congestion comme étant une perte sans-fil, ELN conserve en mémoire les numéros de séquence des paquets perdus, seulement si le nombre de paquets présents dans la file d'attente de la station de base n'est pas proche du maximum de la taille de cette dernière.

ELN nécessite des traitements au niveau de la couche transport (TCP) ce qui entraîne des modifications importantes dans la station de base et aussi plus de traitements.

4.2.3 TCP-Aware Link-Layer Retransmission

TCP-Aware est un schéma introduisant la coopération entre la couche transport et la couche liaison. Dans ce schéma, d'un côté, des estampilles des paquets (*time stamps*) sont utilisées pour offrir un transfert fiable des segments en désordre. De l'autre côté, une notification explicite de retransmission (ERN, *Explicit Retransmission Notification*) est utilisée pour éviter la réduction de la fenêtre de congestion et les multiples retransmissions à partir de la source [ZN00].

Au niveau de la station de base sans-fil, chaque segment TCP est encapsulé dans un paquet avec une entête « LAC-PDU » (Link Access Control-Protocol Data Unit), puis ce paquet est transféré à travers la liaison sans-fil. Chaque entête LAC-PDU est donnée un numéro de séquence spéciale à la couche liaison et le segment correspondant est mis en *cache* à la station de base. La couche liaison du récepteur sans-fil acquitte le paquet en y mettant aussi l'entête LAC-PDU. Ces retours des numéros de séquence LAC-PDU permettent à la station de base de détecter et de retransmettre les paquets perdus. Les paquets retransmis reçoivent par ailleurs un nouveau numéro de séquence.

En outre, la station de base utilise la notification explicite de retransmission (ERN) pour éviter que l'expéditeur réduise sa fenêtre de congestion. À la réception d'un accusé dupliqué

(DupACK), si le paquet concerné est toujours présent dans le tampon de la station de base, il est donc retransmis localement et le bit ERN de l'entête de l'ACK est activé avant d'être envoyé à l'expéditeur. La source voit le fait que ce bit ERN est activé comme un signal de perte sans-fil. Par conséquent, elle préserve la taille de sa fenêtre de congestion et annule la retransmission devenue inutile étant donné que le paquet a été localement retransmis au niveau de la station de base.

Malgré l'intérêt que peut apporter une telle méthode, elle exige un taux élevé de traitement à la station de base et une surcharge de sa mémoire tampon, étant donné que les paquets doivent y être mis en *cache*.

4.2.4 I-TCP

I-TCP, (*Indirect-TCP*) n'a pas été conçu pour différencier les pertes mais pour améliorer TCP sur les réseaux sans-fil [BB95]. C'est son fonctionnement qui fait de lui un moyen de pallier le problème des pertes sans-fil. En principe, I-TCP divise la connexion en deux. Une première connexion se fait sur la partie filaire (entre une extrémité et la station de base). Une deuxième connexion est effectuée sur la partie sans-fil (entre la station de base et l'autre extrémité). Une connexion TCP différente est effectuée pour chacune de ces deux parties.

L'avantage de cette méthode est que cela permet de séparer le contrôle de congestion sur le réseau filaire de celui du réseau sans-fil. Cette séparation entraîne effectivement une amélioration de la bande passante utilisée par l'émetteur.

Cependant, cela signifie aussi qu'il est possible d'accuser la réception des paquets à l'expéditeur avant même qu'ils atteignent effectivement le récepteur ce qui pose des problèmes dans certains cas.

4.2.5 ECN

La notification explicite de congestion (ECN, *Explicit Congestion Notification*) est décrite dans la RFC 3168 [RFB01] comme un mécanisme de notification de congestion à l'émetteur (voir la section 2.2 pour plus d'informations sur son fonctionnement). Certains ont aussi analysé la possibilité de l'utiliser comme un mécanisme permettant de différencier les pertes causées par la congestion de celles dues aux erreurs de la liaison sans-fil. L'idée principale de son utilisation pour différencier les pertes est de regarder le dernier intervalle de temps dans lequel une perte s'est produit ; si la source a déjà reçu un ECN, cela indique la présence d'une congestion, sinon et s'il existe des paquets perdus, on conclu que c'est dû au sans-fil.

TCP-Eaglet

Dans [BW04b] Biaz *et al.* montrent que, dans un cas normal, ECN ne peut pas être utilisée avec certitude pour distinguer la congestion des pertes sans-fil. Par contre, si une gestion active de la file d'attente des routeurs intermédiaires telle que RED (*Randon Early Detection*) est employée et que ses paramètres sont bien définis, l'utilisation d'ECN devient plus adéquate (c'est-à-dire les routeurs signalent efficacement une imminente congestion). En réponse à ce signalement ECN, la source devrait forcement éviter la congestion. Par conséquent, TCP-Eaglet propose d'utiliser ECN pour la différenciation de pertes (si les conditions précédentes sont vérifiées) de la manière

suivante :
Vu la nature de la phase de démarrage lent (*Slow Start mode*) d'une connexion TCP où le débit augmente exponentiellement, la source ne devrait pas procéder à un mécanisme de différenciation de perte pendant cette phase. Par contre, lorsque la connexion est en phase d'évitement de congestion (*Congestion Avoidance mode*), où le débit augmente linéairement, la source réduit son débit d'envoi seulement lorsqu'elle reçoit des acquittements avec un signalement des paquets marqués ECN. Ainsi, dans cette dernière phase, toute perte est considérée comme étant due au sans-fil et le débit est maintenu.

Nous pensons que cette hypothèse est très forte et qu'elle est pas certaine. Des paquets perdus dans un réseau sans-fil à cause de la congestion peuvent toujours se produire en phase d'évitement de congestion même si ECN est utilisée. À titre d'exemple, si des paquets marqués ECN sont perdus sur le canal sans-fil, la source n'en a pas conscience. Par conséquent, ces pertes seront considérées comme dues au sans-fil ce qui n'est pas vrai. La section 6.2.4 en présente une discussion détaillée.

4.3 Méthodes implicites de différentiation de pertes

Les méthodes implicites de différentiation de pertes regroupent les mécanismes de bout en bout qui ne nécessitent aucune modification des équipements réseau. Ces méthodes peuvent être classées en trois catégories : les méthodes basées sur le RTT (*Round Trip Time*) ; les méthodes basées sur IAT (*Inter Arrival Time*) et celles basées sur le ROTT (*Relative One-way Trip Time*).

4.3.1 Méthodes basées sur le RTT

Le RTT est le délai entre le moment d'envoi d'un paquet et la réception de son acquittement. Cette section détaille les méthodes de différenciation de pertes basées sur le RTT.

NewReno-FF

Barma et Matta, dans [BM02], proposent une méthode de différenciation de pertes basée sur la variation du RTT appelée « NewReno-FF (*Flip Flop Filter*) » afin d'améliorer la version New Reno de TCP.

Le « Flip Flop Filter » permet d'avoir un calcul adéquat du RTT, sa moyenne et ses variations. Il utilise deux filtres basés sur la moyenne mobile exponentielle et pondérée (EWMA, *Exponential Weighted Moving Average*) pour calculer le RTT (contrairement à TCP qui utilise un seul filtre EWMA stable). Un filtre est stable et l'autre est dynamique afin de donner plus de poids (plus d'importance) aux valeurs du RTT récemment observées. L'idée principale est d'utiliser, tant que possible, le filtre dynamique et de passer à l'autre stable lorsque les RTTs varient considérablement et deviennent bruyants.

NewReno-FF définit un seuil appelé « limite de contrôle ». Lorsqu'un ACK arrive et que le RTT augmente au delà de cette limite, un compteur est incrémenté. Au cas où ce compteur dépasse un certain nombre $n = 5$ et si l'émetteur détecte un ou plusieurs paquets perdus, il considère les pertes comme dues à la congestion et il réduit son débit. Par contre, si le compteur

est inférieur à n, les pertes sont classifiées comme dues au sans-fil et le débit est maintenu. Le compteur est calculé sur les $l = 10$ derniers ACKs reçus.

En résumé, la technique NewReno-FF de différenciation de pertes considère que la variation du RTT est élevée pour les pertes de congestion et qu'elle est faible pour les pertes sans-fil. Cette conclusion est le résultat des simulations sur un simple réseau filaire avec un modèle d'erreur de transmission pour simuler les pertes sur un réseau sans-fil [1]. Nous pensons que ces résultats sont basés sur une fausse supposition du modèle théorique utilisé (voir la section 5.2 pour des informations détaillées).

DCCP/TFRC-LD (*TFRC Loss Differenciation*)

TFRC est un contrôle de congestion basé équation. Dans TRFC le récepteur calcule le RTT et le renvoie à l'émetteur avec des événements de pertes. À la base de ces informations, l'émetteur calcule le débit d'envoi. TFRC-LD propose que le récepteur utilise un calcul propre du RTT appelé « le RTT corrigé » afin d'estimer sa valeur en prenant en compte les pertes sur le réseau sans-fil.

L'idée de la méthode DCCP/TFRC-LD [Lin08, DLS05] est dérivée de TCP Vegas qui utilise le RTT pour la régulation du débit d'envoi. Pour faire le calcul du RTT corrigé, elle ajoute une option, appelée « rets », à l'entête DCCP (contrairement à ce qui se fait en TCP Vegas, dans DCCP/TFRC-LD il appartient au récepteur de prendre des mesures appropriées pour le contrôle de congestion). « rets » est un compteur du nombre de retransmissions au niveau MAC qui ont été faites. Puis, le temps pris par ces retransmissions est calculé et retiré du RTT estimé par le récepteur donnant ce qui a été appelé le RTT corrigé. L'émetteur reçoit les acquittements avec un RTT habituel comme si c'était sur un réseau filaire. De cette manière, l'influence des pertes sans-fil est supposée être annulée. Ainsi, l'émetteur calcule son débit sans se soucier des pertes dues aux erreurs du canal sans-fil. Par conséquent, le débit n'est pas réduit dans le cas où des pertes sans-fil se produisent.

Grâce à TFRC-LD, l'expéditeur emploie un débit d'émission plus adapté. Cette méthode nécessite en revanche un traitement supplémentaire à la station de base et une modification de son code.

4.3.2 Méthodes basées sur le ROTT

Étant donné que les temps d'envoi et de réception d'un paquet sont mesurés séparément chez l'expéditeur et chez le récepteur, la valeur absolue du délai entre les deux est difficile à obtenir sauf s'ils sont parfaitement synchronisés, ce qui est impossible en réalité. C'est pourquoi un délai relatif (la différence entre le temps de réception et le temps d'envoi indiqué dans l'entête du paquet) est utilisé et appelé ROTT (*Relative One-Way Trip Time*). Cette section détaille les méthodes de différenciation de pertes basées sur ce délai (ROTT).

1. « Dans le réseau de simulation, des liens filaires représentent des liaisons sans-fil avec des erreurs de transmission

TCP Santa Cruz Loss Differentiation

Le protocole de transport « TCP Santa Cruz », expliqué dans [PGLA99], présente une version améliorée de TCP. Cette version permet non seulement de détecter la congestion dans le réseau mais aussi le sens dans lequel elle s'est produite (c'est-à-dire sur le chemin ascendant ou descendant de la connexion). De cette manière, la congestion arrivant sur le chemin de retour (le chemin ascendant) n'a aucun effet sur le contrôle de congestion (c'est-à-dire que la fenêtre de congestion n'est pas réduite dans ce cas).

La congestion est déterminée en calculant le temps relatif du retard qu'un paquet peut expérimenter à l'égard d'autres paquets. Il est utilisé pour estimer le nombre de paquets résidant dans la file d'attente du goulot d'étranglement ; l'algorithme de contrôle de congestion maintient le nombre de paquets dans la file d'attente du goulot d'étranglement à un niveau minimum en ajustant la fenêtre de congestion de la source TCP.

La fenêtre de congestion est réduite si la taille de la file d'attente du goulot d'étranglement (en réponse à l'augmentation de la congestion dans le réseau) augmente au-delà d'un certain nombre fixé de paquets (n). La fenêtre est augmentée lorsque la source détecte la disponibilité de bande passante supplémentaire dans le réseau, par exemple après une diminution de la taille de la file d'attente.

À partir de constatations faites dans [PGLA99], Parsa *et al.* introduisent une méthode de différenciation de pertes dans [PGLA00] et l'appellent TCP Santa Cruz. Cette méthode identifie une perte, qui est précédée par une augmentation de ROTT, comme une perte de congestion. Pour TCP santa Cruz, ROTT augmente lorsque la taille de la file d'attente du routeur surchargé augmente. Une perte sans-fil, d'autre part, peut être identifiée comme une perte aléatoire qui n'est pas précédée par une accumulation dans la file d'attente.

TCP Santa Cruz surveille les changements dans la file d'attente sur un intervalle égal au temps qu'il faut pour transmettre une fenêtre de données et de recevoir ses accusés de réception. Si la taille de la fenêtre augmente (c'est-à-dire que ROTT augmente), un compteur (initialement 0) est mis à 1. Si elle augmente encore, le compteur est mis à 2. Pour chaque diminution de cette taille (c'est-à-dire que ROTT diminue), le compteur est décrémenté. Pour chaque perte signalée à l'émetteur alors que le compteur est à 2, elle est considérée comme congestion et la fenêtre de congestion est divisée par deux. Pour tout autre cas, la perte est traitée en tant qu'une perte sans-fil et la source maintient son débit.

En résumé, TCP Santa Cruz considère une perte comme un signe de congestion si ROTT augmente, et sans-fil sinon. Pour correspondre à l'hypothèse de cette méthode, les pertes sans-fil doivent arriver d'une façon isolée ce qui n'est pas le cas en réalité car des rafales de pertes arrivent souvent. Le goulot d'étranglement peut être aussi sur le lien sans-fil et un grand nombre de pertes sans-fil entraîne également une augmentation de la taille de la file d'attente ce qui peut diminuer sa viabilité.

Spike

Dans Spike [TTM+00], le ROTT est utilisé pour identifier l'état de la connexion en cours. La connexion peut ainsi être dans un de deux états possibles : *spike* ou *non spike*. Une perte est considérée comme congestion si la connexion est dans l'état *spike*, sinon la perte est classifiée comme due au sans-fil.

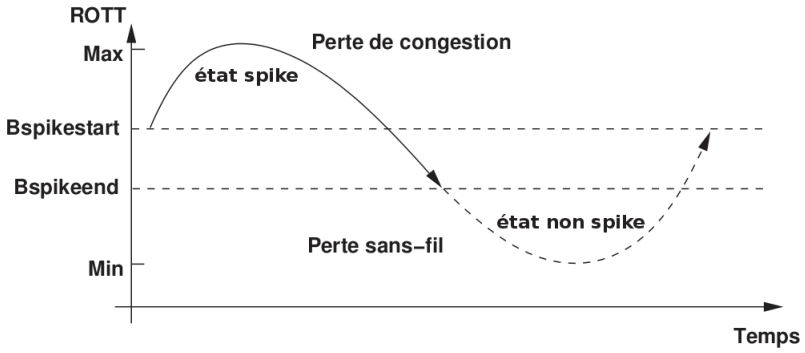


Figure 4.1 – l'état de la connexion avec la méthode Spike.

Au début, une connexion est dans un état *non spike*. On entre en état *spike* si la connexion n'est pas dans cet état et si à la réception d'un paquet i, le délai ROTT de i est supérieur à un seuil prédéfini $B_{spikestart}$ (c'est-à-dire que c'est le seuil indiquant le ROTT maximum possible dans l'état *spike*). La connexion quitte cet état si ROTT de i est inférieur à un autre seuil $B_{spikeend}$ indiquant le minimum de ROTT dans l'état *spike*.

Les valeurs de $B_{spikestart}$ et $B_{spikeend}$ sont définies comme suite :
$B_{spikestart} = ROTTmin + 20ms$ et
$B_{spikeend} = ROTTmin + 5ms$.
$ROTTmin$ est le minimum de ROTT observé depuis le début de la connexion jusqu'à la réception de i.

Spike considère de cette manière que les pertes de congestion se produisent autour des pics de ROTT. La figure 4.1 montre les deux états de la connexion ainsi que les seuils de Spike. Nous pouvons y constater :

1. Si pertes et ROTT $> B_{spikestart}$, ce sont toujours des pertes de congestion
2. Si pertes et ROTT $< B_{spikeend}$, ce sont toujours des pertes sans-fil
3. Si pertes et $B_{spikeend}$ < ROTT < $B_{spikestart}$, ce sont des pertes de congestion ou sans-fil en fonction de l'état de la connexion.

Dans Spike, tout dépend des valeurs prédéfinies des deux seuils. Pour une connexion qui n'a jamais expérimenté des délais de ROTT inférieurs à 5ms ou supérieures à 20ms, l'algorithme est tout simplement inutile.

PLC

L'algorithme de PLC (*Packet Loss Classification*), proposé dans [HCR+05], est semblable à celui de Spike et il est également basé sur le temps relatif ROTT. En effet, PLC est une version améliorée de Spike. D'une part, PLC redéfinit les seuils de Spike comme suit :
$Spike_{start} = rott_{min} + \alpha * (rott_{max} - rott_{min})$, où $\alpha = 10.8$
$Spike_{end} = rott_{min} + \beta * (rott_{max} - rott_{min})$, où $\beta = 0.3$

D'autre part, PLC redéfinit la classification des pertes de la manière suivante : lorsque les pertes de paquets sont détectées (en vérifiant les numéros de séquence), elles sont supposées être

des pertes sans-fil si ROTT est inférieur à $Spike_{end}$, et de congestion si ROTT est supérieur à $Spike_{start}$.

ZigZag

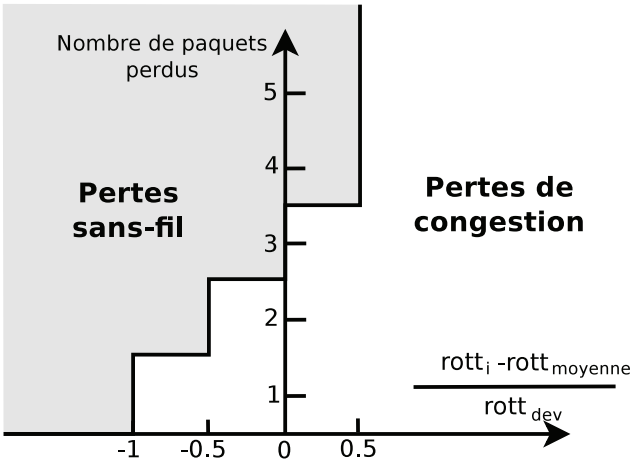


Figure 4.2 – Le schéma de ZigZag.

La figure 4.2 montre le schéma de cette méthode. En plus du nombre de pertes n, ZigZag [CCV03] est basé sur la moyenne ($rott_{mean}$) de ROTT et sa déviation ($rott_{dev}$) afin de classifier les pertes sans-fil. Ces deux valeurs sont calculées comme suit, ($\alpha = 1/32$):

- $rott_{mean} = (1 - \alpha) * rott_{mean} + \alpha * rott$
- $rott_{dev} = (1 - 2\alpha) * rott_{dev} + 2\alpha * rott - (rott - mean)$

Ainsi, si :

1. ($n = 1$ et $rott_i < rott_{mean} - rott_{dev}$)
2. ou ($n = 2$ et $rott_i < rott_{mean} - rott_{dev}/2$)
3. ou ($n = 3$ et $rott_i < rott_{mean}$)
4. ou ($n > 3$ et $rott_i < rott_{mean} + rott_{dev}/2$)

les n pertes sont considérées comme dues aux erreurs de transmission sur la liaison sans-fil. Toute autre perte est classifiée comme due à la congestion.

De cet manière, ZigZag prétend classifier 84% des pertes de congestion avec ($rott_i < rott_{mean} - rott_{dev}$). ZigZag suppose que les pertes sans-fil sont isolées la plupart du temps (une seule perte sans-fil à la fois), or c'est la congestion qui est accompagnée d'une augmentation de délai. Donc l'augmentation de ROTT identifierait la majorité des pertes de congestion. D'ailleurs, dans la figure 4.2, plus le nombre de paquets perdus augmente, plus le seuil ROTT utilisé pour la différenciation de pertes bascule vers la droite (sur l'axe des abscisses).

L'hypothèse de ZigZag (que les pertes sans-fil arrivent individuellement) n'est pas adéquate, parce qu'en réalité les pertes sans-fil arrivent en rafale, surtout si une période d'interférence est assez longue.

ZBS

ZBS, décrit dans [CCV03], est un algorithme hybride entre ZigZag, mBiaz (présenté plus loin) et Spike. Il choisit l'un d'entre eux en fonction des conditions réseau et en particulier si le lien sans-fil est le dernier saut du réseau (WLH, *Wireless Last Hop*) entre l'émetteur et le récepteur et combien de flux N partagent le goulot d'étranglement.

ZBS estime que dans un réseau WLH, ZigZag fait mieux que Spike. Par contre, si ce lien n'est pas partagé, mBiaz est meilleur. Dans un réseau backbone sans-fil (WB, *Wireless Backbone*), c'est Spike qui est le meilleur. ZBS fait aussi la remarque que dans un réseau WLH, $Tnarr = IAT_{avg}/IAT_{min} \approx 1$, tandis que $Tnarr = N$ dans un réseau WB. Ainsi, l'algorithme de ZBS est le suivant (c'est la 3ème et dernière version ZBS3) :

si $(rott - rott_{min} < 0.05 * IAT_{min})$, utiliser Spike ;
sinon si $(rott_{dev} < 0.5 * IAT_{min})$, utiliser ZigZag ;
sinon
  si $(Tnarr < 0.95)$, utiliser ZigZag ;
  sinon si $(Tnarr < 1.1)$, utiliser mBiaz ;
  sinon si $(Tnarr < 1.5)$, utiliser ZigZag ;
  sinon, utiliser Spike.

ZBS choisit ces seuils en faisant quelques suppositions pour que son algorithme fonctionne ; par exemple $Tnarr$ est < 1.5 dans un réseau WB partagé entre 2 utilisateurs et le goulot d'étranglement est complétement utilisé. Par conséquent, ces suppositions font que ZBS peut mal identifier des pertes de congestion dans les autres cas.

TD *Trend-Loss-Density-based*

TD *Trend-Loss-Density-based* utilise les courbes de ROTT afin de déterminer la nature des pertes [CHL06]. Il considère que les pertes de congestion surviennent autour et après un pic de ROTT. Lorsque le ROTT commence à augmenter, cela indique que le réseau commence à être surchargé. Donc la probabilité d'avoir des pertes dues à la congestion augmente. Si le débit est réduit suite aux pertes, le réseau devient moins chargé et donc ROTT diminue. TD estime également que les pertes dues à la congestion sont rarement isolées (c'est-à-dire elles sont nombreuses). Donc une perte isolée est toujours traitée comme perte sans-fil.

TD propose d'identifier la nature des pertes en utilisant la densité des pertes et leurs tendance : a) la tendance indique si les pertes sont survenues autours d'un pic de la courbe de ROTT ou non, b) la densité des pertes examine combien de pertes se trouvent autour de ce pic.

De plus, pour chaque courbe de tendance, TD détermine un seuil de densité. Si à un moment donné une ou plusieurs pertes se produisent et que la densité de perte autour de la courbe de tendance de ROTT excède son seuil défini, la ou les pertes sont considérées comme dues à la congestion.

TD définit quatre types différents de tendance de ROTT, présentés dans la figure 4.3. Ainsi, une courbe ROTT peut être : Haute, Basse, Concave et Convexe. Ces courbes sont tracées avec une fonction prenant en compte les n derniers paquets (en regardant leurs numéros de séquence).

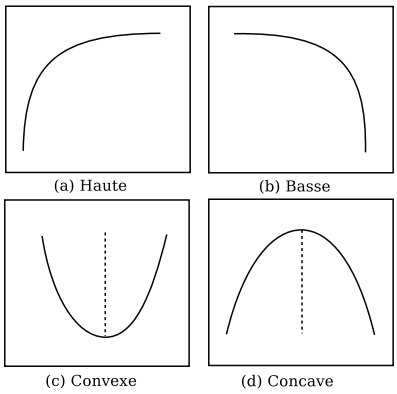


Figure 4.3 – Courbes de tendance de ROTT dans TD.

TD estime que les pertes de congestion arrivent souvent sur les parties descendantes de chaque courbe (Basse, ConvexeBasse ou ConcaveBasse), par conséquent le seuil de densité est mis à 0 pour ces parties. De l'autre côté, si la perte se produit sur les parties montantes, il est difficile d'identifier la nature de perte. Un seuil de 5 paquets est mis pour la courbe ConvexeHaute et il est de 2 pour les courbes Haute et ConcaveHaute.

En utilisant ces seuils de densité, TD classifie les pertes comme suit :
- Toute perte isolée est une perte sans-fil.
- Si la courbe de tendance est de type Basse, ConvexeBasse ou ConcaveBasse (les côtés descendants), les pertes sont classifiées comme dues à la congestion.
- Si la courbe de tendance est de type ConvexeHaute et le nombre de pertes excède le seuil de densité de 5, elles sont classifiées comme congestion.
- Enfin, si la courbe de tendance est de type Haute ou ConcaveHaute avec plus de deux pertes, c'est de la congestion, sinon, c'est des pertes sans-fil.

C'est un algorithme très compliqué car il difficile à implémenter. Il consomme également beaucoup de ressources processeur.

4.3.3 Méthodes basées sur l'IAT

Le délai entre paquets, IAT (*Inter Arrival Time*), indique le temps qui sépare la réception de deux paquets. Cette section détaille les méthodes de différenciation de pertes basées sur l'IAT et plus précisément sur sa variation.

Biaz et mBiaz

Biaz [BV99] repose sur l'IAT et sa valeur minimum (IAT_{min}) déjà observée par le récepteur dans son schéma de différenciation de pertes. Quand à un moment donné le récepteur reçoit un paquet q_i après n paquets perdus, il considère que les n pertes sont dues aux erreurs de transmission sur le canal sans-fil si et seulement si :

$$(n+1)\,IAT_{min} \leq IAT_{q_i,q_j} < (n+2)\,IAT_{min} \tag{4.1}$$

où q_j est le dernier paquet reçu avant les n pertes et IAT_{q_i,q_j} est le délai entre la réception des paquets q_i et q_j.

Ainsi, Biaz considère que les pertes de congestion diminuent la valeur d'IAT (une perte de congestion entraîne moins de temps de transmission sur le canal sans-fil et donc un IAT plus petit). Par contre, les pertes sans-fil occupent leur temps de transmission. Par conséquent, le premier paquet reçu après ce type de perte arrive à son temps prévu de réception sans conséquence sur la valeur d'IAT.

Pour que ce schéma soit viable, Biaz fait trois suppositions :

1. Il y a un seul lien sans-fil entre l'émetteur et le récepteur et c'est le dernier saut du réseau.
2. Le lien sans-fil est le goulot d'étranglement de la connexion.
3. L'émetteur transmet une grande quantité de données.

De plus, Biaz a un taux élevé de mauvaise classification des pertes de congestion. C'est pourquoi les auteurs ont modifié le seuil supérieur de l'équation 4.1 comme suit :

$$(n+1)\,IAT_{min} \leq IAT_{q_i,q_j} < (n+1.25)\,IAT_{min} \tag{4.2}$$

Cette version améliorée appelée mBiaz [CCV03] permet de mieux identifier les pertes de congestion mais au détriment des pertes sans-fil. Biaz et mBiaz ne sont pas conçus pour n'importe quelle condition de réseau (il y a en effet les trois suppositions à respecter). De plus, ces méthodes sont conçues pour fonctionner avec un seul flux sur le réseau. Donc, leur application reste très restreinte.

SPLD

SPLD (*Statistical Packet Loss Discrimination*) [PSJ06], tout comme Biaz, repose sur le délai entre paquets (IAT) et fait la différenciation de pertes chez le récepteur. La différence entre SPLD et Biaz est que SPLD pallie l'imprécision de Biaz lorsque multiples flux partagent le goulot d'étranglement.

Comme montré dans la figure 4.4, SPLD comporte trois modules. Un module de surveillance recueille des informations sur les paquets reçus telles que le numéro de séquence et l'estampille du paquet, et ceci pendant un temps défini dit temps de surveillance. S'il n'y a pas de pertes pendant ce temps, le deuxième module, appelé le gestionnaire statistique, met à jour la moyenne d'IAT (autrement appelée la valeur stable, IAT_{stable}). Par contre, si une perte se produit, le troisième module, appelé le discriminant, utilise les informations recueillies par le gestionnaire statistique pour classifier les pertes.

SPLD considère la perte comme due à la congestion si la valeur actuelle d'IAT ($IAT_{cur} = \sum_{i=1}^{n} IAT_i/N$), calculée pendant le temps de surveillance (N paquets reçus pendant cette période), est inférieure à IAT_{stable}. Sinon, c'est une perte sans-fil.

4.4 Discussion

Plusieurs études de comparaison des méthodes permettant d'améliorer la performance des protocoles des transport sur réseaux sans-fil existent. En particulier, l'étude faite dans [BJP07] compare un grand nombre des méthodes de différenciation de pertes de bout-en-bout. Cette étude donne plusieurs conclusions concernant les méthodes comparées (Biaz, mBiaz, SPLD, Spike, PLC, ZigZag, ZBS et TD) et certaines s'accordent avec ce que nous avons avancé :

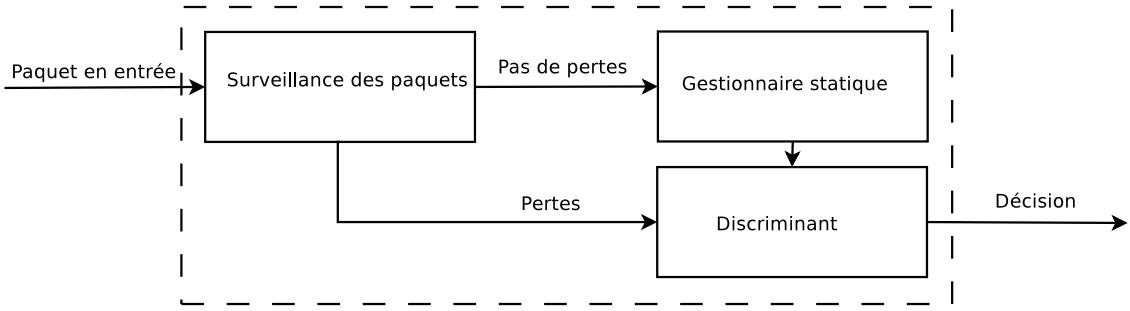


Figure 4.4 – Le mécanisme de SPLD.

- Les méthodes basées sur ROTT sont meilleures que celles basées sur IAT.
- Parmi les méthodes basées sur IAT, SPLD est meilleure que Biaz et mBiaz lorsque de multiples flux partagent la bande passante.
- Parmi les méthodes basées sur ROTT, TD est la meilleure quand le niveau de congestion est très élevé mais Spike et PLC sont meilleurs quand il y a un trafic léger sur le réseau. Quant à ZBS utilisant plusieurs schémas, sa performance est même inférieure à Spike seule. ZigZag a également un faible taux de bonne classification.

Quant aux méthodes explicites de différenciation, elles permettent de mieux protéger l'émetteur contre les pertes sans-fil en lui cachant leur existence, mais la connexion souffre souvent à cause des *times out* générés par ces mêmes paquets cachés [BPSK97]. En conséquence, la performance du protocole du transport devient pire en utilisant ce genre de méthodes.

Le tableau 4.1 récapitule et compare les propriétés principales des méthodes de différenciation de pertes. La comparaison est faite en terme d'efficacité, de facilité, de généralité et de l'endroit où la modification devrait se faire :

- l'efficacité est représentée en un nombre d'étoile sur cinq. Cinq étoiles indique que la méthode est très efficace et elle est capable de différencier toutes les pertes ;
- la simplicité, représentée en un nombre d'étoile sur trois, montre à quel point l'algorithme de la méthode est facile à comprendre et à implémenter ;
- la généralité indique si la méthode peut être intégré dans n'importe quel protocole de transport ou non. Et si non, où elle est appliquée.
- la particularité précise sur quelle extrémité (émetteur ou récepteur), équipement (Routeur ou point d'accès AP) ou couche OSI (le numéro de la couche est mis entre deux parenthèses) fonctionne la méthode.

Il est à noter que RELD ne sera présenté que dans la section suivante.

4.5 Conclusion

Le nombre de stations mobiles augmente d'une façon considérable ces dernières années. Les protocoles de transport fonctionnent moins bien sur réseaux sans-fil que sur un réseau filaire. Or, les services de *streaming* vidéo sur Internet sont actuellement présents partout et accessibles par tout le monde. Tout cela a donc créé un besoin important d'améliorer le fonctionnement de ces protocoles de transport sur réseau sans-fil et en particulier pour les applications ayant des contraintes spécifiques tels que le délai et la stabilité du transfert.

Méthode	Efficace	Simple	Générale	Extensible	Particularité
Snoop	*****	*	Oui	*	AP + table de routage (3)
ELN	*****	**	Oui	**	AP + entête IP (3)
TCP-aware	*****	*	Oui	*	AP + couche liaison (2)
I-TCP	*****	***	Oui	*	AP + deux connexions (4)
ECN	*	***	Oui	**	Routeurs + émetteur (4)
NewReno-FF	**	***	Non (NewReno)	***	Émetteur (4)
DCCP/TFRC-LD		**	Non (TFRC)	**	AP (2) + récepteur (4)
TCP Santa Cruz	**	**	Non (TCP)	***	Récepteur (4)
Spike	***	**	Oui	***	Récepteur (4)
PLC	***	**	Oui	***	Récepteur (4)
ZigZag	**	***	Oui	***	Récepteur (4)
ZBS	**	*	Oui	**	Récepteur (4)
TD	****	*	Oui	*	Récepteur (4)
Biaz	*	***	Non (un seul flux)	***	Récepteur (4)
mBiaz	**	***	Non (un seul flux)	***	Récepteur (4)
SPLD	***	***	Oui	***	Récepteur (4)
RELD	****	***	Oui	***	Émetteur (4)

Tableau 4.1 – Comparaison des méthodes de différenciation de pertes.

Des solutions variées ont été proposées afin d'atteindre ce but, en particulier afin d'armer les protocoles de transport contre de mauvaises réactions en cas de pertes sans-fil. Elles ont été souvent créées sous des conditions spécifiques ou pour améliorer le débit dans un type particulier de réseaux. Certaines sont parties de mauvaises suppositions et ont donc eu de mauvais résultats de classification. D'autres ont tout simplement restreint leur application aux cas spéciaux, qui ne concordent pas tout le temps à la réalité. Tandis que celles qui donnent de bons résultats sont soit compliquées et difficiles à implémenter, soit nécessitent des modifications importantes sur les équipements du réseau. Par conséquent, leurs déploiement est toujours difficile voire impossible dans le monde réel.

Donc à présent aucune méthode de différenciation de pertes n'est vraiment appliquée. Nous sommes partis du principe que, d'une part, trouver une méthode simple et efficace permettra de faciliter son implémentation dans des protocoles de transport déjà utilisés et déployés, et d'autre part, qu'il y a de la place pour améliorer les performances de ces protocoles sur réseaux sans-fil et par conséquent optimiser l'utilisation de ce genre de réseaux. C'est ce que nous proposons dans cette partie de thèse.

La suite de cette partie de thèse est consacrée à étudier les caractéristiques d'une transmission sur un réseau sans-fil, de soumettre des améliorations et d'évaluer la performance de la solution proposée.

Chapitre 5

DIFFÉRENCIATION DE PERTES EN UTILISANT LE RTT ET ECN

Avant le déploiement des réseaux sans-fil à large échelle – ils sont aujourd'hui omniprésents – les pertes étaient traditionnellement considérées comme causées par la congestion. En effet, la probabilité qu'un lien filaire soit responsable de la survenue d'une erreur (une perte par la suite) est très faible, les pertes dues aux erreurs du lien filaire sont négligeables. Ainsi, un paquet perdu sur un lien filaire est principalement dû au dépassement de la capacité des files d'attente sur les routeurs lorsque le débit des paquets entrants excède la capacité du lien sortant, c'est-à-dire que le réseau est surchargé, autrement dit congestionné. Avec l'arrivée des réseaux sans-fil, les causes de pertes sont nombreuses et leur nombre augmente avec la dégradation du signal en fonction de différentes circonstances de l'environnement : distance entre la station de base et les mobiles, taux élevé des bits erronés liés à la liaison sans-fil et ainsi de suite. Ces pertes amènent la source des paquets à réduire son taux d'envoi, car elle considère les pertes, parfois à tort, comme un signe de congestion. Afin d'avoir une meilleure performance lorsqu'une transmission se fait sur un réseau sans-fil, **la source doit être en mesure de distinguer les pertes dues à la congestion des pertes sans-fil**.

Comme la signalisation ECN apparaît avant la congestion, un émetteur utilisant ECN peut éviter la congestion dans un réseau filaire. Dans la plupart des cas il réduit son taux d'envoi en réponse à ECN. Mais dans le cas où une rafale de paquets arrive au routeur, la file d'attente peut devenir rapidement pleine sans que sa moyenne (q_{avg}) n'ait beaucoup variée. Le routeur rejette par conséquent les paquets sans les marquer. Dans de tels cas, les pertes sont nombreuses et ECN n'est pas utile. Si on rajoute à cela le fait qu'un paquet marqué peut être perdu sur un lien radio, nous pouvons conclure alors qu'ECN seul ne peut être utilisé comme un facteur de différenciation entre les pertes sans-fil et les pertes de congestion.

Pour cette raison, il est nécessaire de coupler ECN avec un autre facteur. Nous proposons, dans ce chapitre, de montrer que le **RTT en conjonction avec ECN peut être utilisé pour différencier efficacement les pertes**. Mais avant cela, nous allons analyser l'utilité de la différenciation de pertes. En effet, comment les pertes causées par des erreurs du canal sans-fil peuvent-elles affecter le débit effectif (throughput) ? Est-ce que la différenciation peut aider à augmenter ce débit ? La réponse à cette question est effectivement oui. Cela sera démontré par une étude analytique dans la section suivante. Ensuite, après avoir montré l'utilité de la

différenciation des pertes, l'impact des pertes sur le RTT sera analysé et étudié selon deux études différentes. Une première étude montre théoriquement que les pertes causées par des erreurs sans-fil provoquent une augmentation du RTT et que les pertes réduisent sa valeur. Une deuxième étude valide cette conclusion par la simulation.

À noter que par « différenciation de pertes », on entend « différenciation entre les paquets perdus à cause de la congestion et ceux dus aux erreurs du canal sans-fil ». À noter également que le RTT utilisé dans la présente section est le RTT du paquet *suivant* le paquet perdu (le temps entre la réception de l'ACK et son paquet de données correspondant), qui donne des informations sur le paquet perdu.

5.1 Étude analytique sur l'utilité de la différenciation de pertes

Dans cette section, une étude analytique est présentée pour analyser l'effet de classification de pertes sur le débit global. Le modèle utilisé ici est dérivé de [Kel01, BM02] et adapté à l'algorithme AIMD (*Additive Increase Multiplicative Decrease*) de TCPlike. Dans le contrôle de congestion de TCPlike, le taux d'acquittements (autrement dit la fréquence des accusés de réception des paquets reçus, *ACK ratio*), noté ici par R, est défini comme un paramètre. [1]

Pour chaque ACK reçu, la fenêtre de congestion (*cwnd*) augmente de $R/cwnd$ lorsque $cwnd \geq ssthresh$ (le seuil du mode de démarrage lent (*Slow Start mode*)). Cela signifie que la fenêtre de congestion est augmentée par un paquet pour chaque fenêtre de données acquittée sans paquet perdu ou marqué. D'autre part, si l'accusé de réception indique des pertes ou de paquets marqués, *cwnd* est divisée par 2 (c'est-à-dire $cwnd = cwnd/2$).

Soient p la probabilité de perte de paquets pour lesquels *cwnd* est réduit de moitié (c'est généralement la probabilité de baisse sur l'ensemble des réseaux filaires et sans fil), et RTT (*Round Trip Time*) le temps de parcours . Ainsi, le changement attendu de *cwnd* sur chaque ACK reçu sera:

$$E[\Delta cwnd] = \frac{(1-p) * R}{cwnd} - \frac{cwnd * p}{2} \tag{5.1}$$

Dans le cas d'un accusé de réception tous les R paquets reçus, le temps entre deux mises à jour de *cwnd* est de $\frac{R*RTT}{cwnd}$. Ainsi, le changement de débit dans ce laps de temps est le suivant :

$$\frac{dx(t)}{dt} = \frac{\left(\frac{(1-p)*R}{cwnd} - \frac{cwnd*p}{2}\right)/RTT}{\frac{R*RTT}{cwnd}} \tag{5.2}$$

Cette équation différentielle s'écrit ainsi :

$$\frac{dx(t)}{dt} = \frac{1-p}{RTT^2} - \frac{p}{2R}\left(\frac{cwnd}{RTT}\right)^2 \tag{5.3}$$

Donc :

1. En DCCP, R peut changer au cours d'une communication. En TCP, R est fixe et est égal à 1 ou 2.

$$\frac{dx(t)}{dt} = \frac{1-p}{RTT^2} - \frac{p}{2R}x^2(t) \tag{5.4}$$

Soient $a = \frac{1-p}{RTT^2}$ et $b = \frac{p}{2R}$. En intégrant, nous obtenons :

$$\int_0^{x(t)} \frac{1}{a - bx^2(t)}\, dx(t) = \int_0^t dt \tag{5.5}$$

qui donne :

$$\frac{\tanh^{-1}\left(\sqrt{\frac{b}{a}}x(t)\right)}{\sqrt{ab}} = t + c \tag{5.6}$$

$$\frac{\ln\left(1 + \sqrt{\frac{b}{a}}x(t)\right) - \ln\left(1 - \sqrt{\frac{b}{a}}x(t)\right)}{2\sqrt{ab}} = t + c \tag{5.7}$$

$$\ln(1 + \sqrt{\frac{b}{a}}x(t)) = 2t\sqrt{ab} + C + \ln(1 - \sqrt{\frac{b}{a}}x(t)) \tag{5.8}$$

$$1 + \sqrt{\frac{b}{a}}x(t) = e^{2t\sqrt{ab}+C}(1 - \sqrt{\frac{b}{a}}x(t)) \tag{5.9}$$

c'est-à-dire

$$x(t) = \sqrt{\frac{a}{b}} * \frac{e^{2t\sqrt{ab}+C} - 1}{e^{2t\sqrt{ab}+C} + 1} \tag{5.10}$$

Ainsi, le débit de l'état d'équilibre de TCPlike est donné par :

$$x = \lim_{t\to\infty} x(t) = \sqrt{\frac{a}{b}} \tag{5.11}$$

En remplaçant a et b par leurs valeurs, nous obtenons :

$$x = \frac{1}{RTT}\sqrt{\frac{2R(1-p)}{p}} = \frac{1}{RTT}\sqrt{2R\left(\frac{1}{p} - 1\right)} \tag{5.12}$$

Cette équation (5.12) donne l'influence de p sur le débit effectif. Comparons maintenant l'équation 5.12 en cas de classification de pertes et sans classification. Soient p_c la probabilité qu'un paquet soit perdu en raison de la congestion, et p_w la probabilité qu'un paquet soit perdu à cause d'une erreur sur le lien sans-fil. Dans le cas où l'algorithme de différenciation de pertes est utilisé (c'est-à-dire que les pertes sur le canal sans-fil n'entraînent pas une division de la fenêtre de congestion *cwnd* par deux), $p = p_c$. Dans le cas classique, quand aucune différenciation n'est utilisée, $p = p_c + p_w$. Notons également le débit effectif de la méthode classique par x_c et le débit de la méthode avec différenciation par x_l. De cette manière, le gain attendu du débit est le rapport entre x_l et x_c :

$$\frac{x_l}{x_c} = \frac{\frac{1}{RTT}\sqrt{2R\left(\frac{1}{p_c}-1\right)}}{\frac{1}{RTT}\sqrt{2R\left(\frac{1}{p_c+p_w}-1\right)}} = \sqrt{\frac{\frac{1}{p_c}-1}{\frac{1}{p_c+p_w}-1}} \tag{5.13}$$

À partir de cette équation (5.13), nous pouvons conclure que :

1. si $p_w = 0$ donc $x_l = x_c$ (si toutes les pertes sont filaires, les classifier ne change pas le débit)
2. si $p_w > 0$ donc $x_l > x_c$ et plus p_w augmente, plus le rapport x_l/x_c augmente.

Autrement dit, **l'utilité de la classification de pertes augmente lorsque le taux de pertes des paquets causées par des erreurs sans-fil augmente également**.

5.2 Influence des pertes sur le RTT

Nous avons vu dans la section précédente que faire une différenciation de pertes permet d'augmenter le débit effectif de transmission. Dans cette section, nous allons essayer de concevoir une nouvelle méthode permettant de savoir comment faire cette différenciation. Force est de constater (voir le chapitre 4) que la plupart des méthodes existantes reconnaissent l'existence d'un lien entre les pertes et la variation du temps de contact entre une source et une destination d'une même communication, que ça soit le RTT ou le ROTT. Nous nous sommes également intéressés à ce lien en se basant sur le RTT. Comment les pertes peuvent influencer le RTT ? Les pertes dues à la congestion ont-elles le même impact que celles dues aux erreurs du canal sans-fil ? Si oui, quelle est cette influence ? Et sinon, pouvons-nous en profiter pour avoir une classification entre les deux types de pertes ? Afin de répondre à ces questions, nous allons étudier cette influence des pertes sur la valeur du RTT de deux façons : théorique et par la simulation.

5.2.1 Influence des pertes sur le RTT en théorie

Afin de voir si tous les types de pertes dues à la congestion ou aux erreurs de transmission sans-fil ont la même influence sur le RTT, nous les avons étudiés séparément. Dans ce qui suit, nous entendons par pertes de congestion « des paquets perdus à cause de la congestion ». De la même manière, nous entendons par pertes sans-fil « des paquets perdus à cause des erreurs dues au canal sans-fil ».

Influence des pertes de congestion sur le RTT

Lorsqu'un flux capable ECN est utilisé sur un réseau avec une gestion active de la file active comme RED, la variation du RTT tend généralement à avoir une petite valeur. Cela vient d'une part, du fait que RED a été conçu pour diminuer la variation de la taille de son tampon, et d'autre part, de l'utilisation d'ECN, qui permet d'anticiper la congestion et de diminuer le débit avant de remplir la file d'attente du routeur. De cette manière, un paquet rejeté de la mémoire tampon du routeur à cause de la congestion arrive souvent avant le remplissage de la file d'attente.

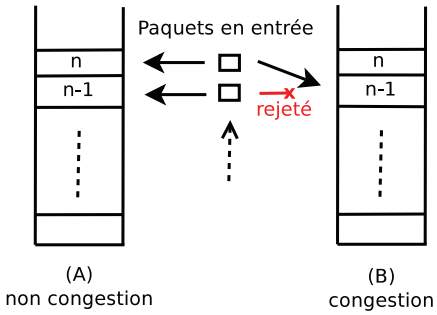


Figure 5.1 – Impact théorique de pertes de congestion sur le RTT.

Ainsi, soit s le temps nécessaire pris par un routeur pour traiter et envoyer un paquet, c'est-à-dire le temps de service de la file d'attente. Supposons qu'un paquet p_i est mis dans une file d'attente d'un routeur (voir figure 5.1). S'il n'y a pas de congestion et que le paquet précédent p_j a été mis en file d'attente à la position $n-1$, la position de p_i est n. Si, au contraire, le paquet précédent p_j a été supprimé avant d'être traité (par exemple à cause de la congestion), alors le nouveau paquet p_i aurait été placé à la position $n-1$. Dans ce cas, il faut à p_i, s moins de temps pour être traité par le routeur. Autrement dit, après un rejet d'un paquet à cause de la congestion sur un routeur, le RTT du paquet suivant voit sa valeur diminuer d'un temps égal à s.

Prenons un exemple : supposons que nous avons un routeur avec une interface 100Mb/s, et des paquets de taille 1000 octets. L'interface envoie à 100Mb/s = 12.8MB/s = 12.8k paquets/s. Cela signifie qu'un paquet prend 1/12,8ms pour être traité, donc $s \approx 0,1$ ms. Pour les interfaces à plus grande vitesse, s est plus petit. La figure 5.2 montre des résultats similaires, avec des différences de moins de 1ms en général.

Pour conclure, le RTT d'un paquet devrait *diminuer* après une perte causée par la congestion.

Influence des pertes sans-fil sur le RTT

Comme cela est dans la norme IEEE 802.11 [CWKS97], quand un paquet est perdu sur le canal d'un réseau sans-fil, il sera retransmis au niveau de la couche MAC, soit jusqu'à ce qu'il soit reçu, soit jusqu'à ce que la limite de tentatives de retransmission soit atteinte. Il y a généralement 7 tentatives de transmission dont une transmission et puis 6 retransmissions. Cela permet de récupérer un paquet perdu plus rapidement par rapport à une retransmission au niveau de la couche transport. Si toutefois la limite est atteinte et que le paquet n'arrive toujours pas, par exemple dans le cas d'une période de perturbation assez longue, il est tout simplement rejeté. Malgré l'inexistence de la congestion pour ce genre de paquets, ils n'arriveront pas à la couche transport. Le paquet rejeté par la couche MAC sera par la suite signalé à la source comme étant perdu par la couche transport. La source procède ainsi à sa retransmission au niveau de la couche transport si TCP est utilisé et il sera seulement reporté à l'application (si elle fait la demande) sans retransmission si c'est DCCP qui est utilisé.

Les cartes sans-fil attendent, pour chaque retransmission MAC et conformément à la norme, un certain *backoff* (par exemple un nombre d'intervalles de temps (*timeslot*)). Ce *backoff* est employé par le mécanisme d'accès au médium basé sur le protocole CSMA/CA (*Carrier Sense*

Multiple Access with Collision Avoidance) dans le but de réduire le nombre de collisions dans le réseau sans-fil. Un *timeslot*) s est égal à $20\mu s$, et le *backoff* est tiré au hasard dans une fenêtre de contention (CW). CW est initialisée à $2^5 - 1$ pour la première transmission de chaque paquet. Si la carte ne reçoit pas d'accusé de réception pour le paquet envoyé, CW est doublée (sans toutefois dépasser 1023) et la transmission est tentée de nouveau. La même chose est répétée pour chaque retransmission (par exemple pour la première retransmission CW=$2^6 - 1$, pour la deuxième retransmission CW=$2^7 - 1$, et plus généralement pour la n-ième retransmission, CW=$\min(2^{5+n} - 1, 1023)$).

Dans un environnement réel, et surtout si le récepteur est proche de la limite de portée du signal, les conditions du canal peuvent être assez mauvaises pour empêcher plusieurs tentatives de retransmission successives. Si toutes les retransmissions MAC échouent, le paquet est perdu. Par conséquent, pour un paquet perdu sur le réseau [2], ses *backoff*s constituent un temps supplémentaire de : $s \cdot \sum_{i=1}^{i=n} \min(rand(2^{5+i-1} - 1), 1023)$. Ce temps est ajouté au RTT du paquet suivant à envoyer, qui attend dans la file d'attente. En cas de ré-essai égal à 6 (cette valeur est utilisée dans les réseaux sans-fil réels et également dans le simulateur du réseau ns2) le temps supplémentaire est égal en moyenne à:
$20\mu s \times (63 + 127 + 255 + 511 + 1023 + 1023)/2 = 30.33$ms.

En conclusion, le RTT augmente d'environ 30 ms à cause d'une perte sans-fil [3]. La figure 5.3 montre des résultats similaires, par exemple à la seconde 30, la différence entre le RTT moyen et le RTT du paquet arrivant après une perte sans-fil est de 15 ms ; la même différence est de 35 ms à la seconde 32.
Pour conclure, le RTT d'un paquet *augmente* en présence d'une perte causée par une erreur sur un réseau sans-fil.

5.2.2 Influence des pertes sur le RTT par la simulation

Pour évaluer l'impact du type de perte (congestion ou sans-fil) sur le RTT dans la simulation, nous utilisons le réseau présenté dans la section 6.1.3, et le modèle de propagation *shadowing-pattern* présenté dans la section 2.7. Nous présentons dans cette section les résultats de deux tests : avec un perturbateur fort et sans perturbateur, en utilisant le contrôle de congestion d'origine TCPlike en dessous de DCCP dans ns2.

Les sources ns2 ont été modifiées afin de pouvoir tracer les valeurs du RTT, du RTT moyen et la variation du RTT des paquets qui suivent une ou plusieurs pertes. Ces valeurs ont été sauvegardées dans un fichier propre à chaque simulation. La raison des pertes a été obtenue à partir des fichiers de trace de la simulation générés par le simulateur ns2. **Donc les RTT tracés dans les courbes représentent le RTT du paquet suivant une perte.**

La figure 5.2 présente l'évolution du RTT en présence des pertes de congestion ; dans ce test, il n'y a pas de perturbations utilisées sur le sans-fil (c'est-à-dire que il n'y a pas de pertes sans-fil provoquées. Nous pouvons remarquer que le RTT est généralement stable car il est généralement compris entre 0,034 et 0,04 secondes). En outre, la plupart des pertes de congestion ont un RTT plus petit que la moyenne, comme on le voit aux secondes 37, 38 et 39 par exemple.

2. Comparé au temps de transmission le plus petit (un paquet qui est arrivé dès la première transmission et avec *backoff* entre 0 et slots).

3. C'est la contribution du *backoff* seulement ; d'autres facteurs, de moindre importance dans notre étude, peuvent modifier cette valeur, tels que la transmission elle-même ou d'autres transmissions au cours de *backoff*

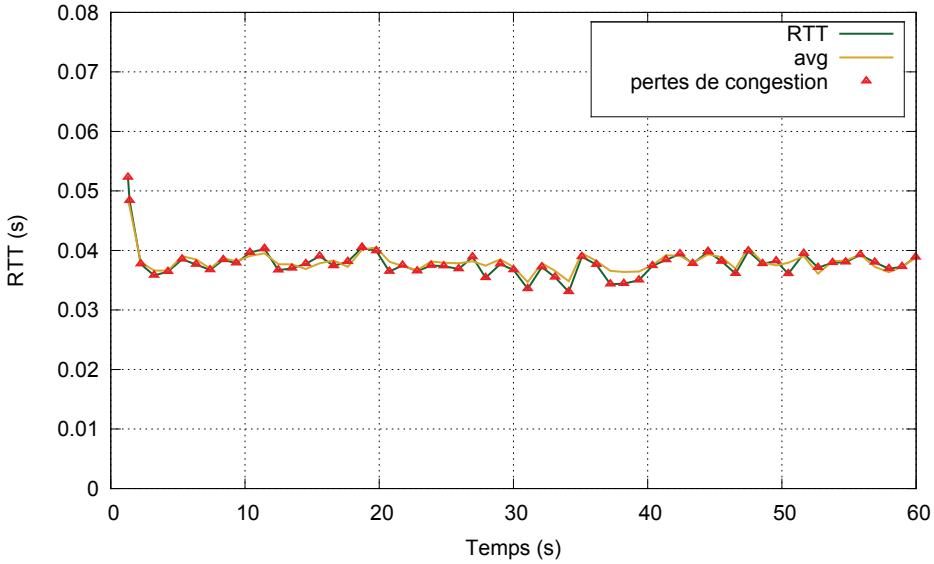


Figure 5.2 – Influence des pertes de congestion sur le RTT.

La figure 5.3 présente l'évolution RTT en présence des deux types de pertes (congestion et sans-fil) ; dans cette figure, un perturbateur (numéro 7) est utilisé sur le canal sans-fil. À noter d'abord que la perturbation sur le réseau sans-fil rend le RTT instable car il est compris entre 0,025 à 0,07 secondes. Deuxièmement, la plupart des pertes de congestion apparaissent lorsque le RTT est inférieur à la moyenne, par exemple entre 53 et 57 secondes, alors que la plupart des pertes sans-fil apparaît lorsque le RTT est au-dessus de la moyenne, par exemple à 20 secondes, 22 et 30.

Les deux sections précédentes conduisent aux mêmes conclusions :

- **Le RTT pour les pertes de congestion est généralement *plus petit* que le RTT moyen.**
- **Au contraire, le RTT pour les pertes sans-fil est généralement *plus grand* que le RTT moyen.**

En outre, il semble que la variation du RTT est généralement plus importante pour les pertes sans-fil que pour les pertes de congestion.

Cette section a permis de voir qu'il existe un rapport entre le nature des pertes et la variation du RTT. Dans la section suivante, ce rapport est identifié et un seuil du RTT est défini pour distinguer les pertes de congestion des pertes sans-fil.

5.2.3 Choix du seuil du RTT pour différencier les pertes

Les sections précédentes ont montré qu'il est possible de classifier les pertes en utilisant le RTT. Cette section va plus loin en analysant en détail l'évolution du RTT pour les pertes de congestion et les pertes sans-fil par rapport à la moyenne du RTT et sa variation. Elle vise à trouver un seuil entre les valeurs des RTT après des pertes sans-fil et les valeurs des RTT après une congestion.

Afin de pouvoir trouver précisément ce seuil, 51 tests ont été faits en utilisant le simulateur

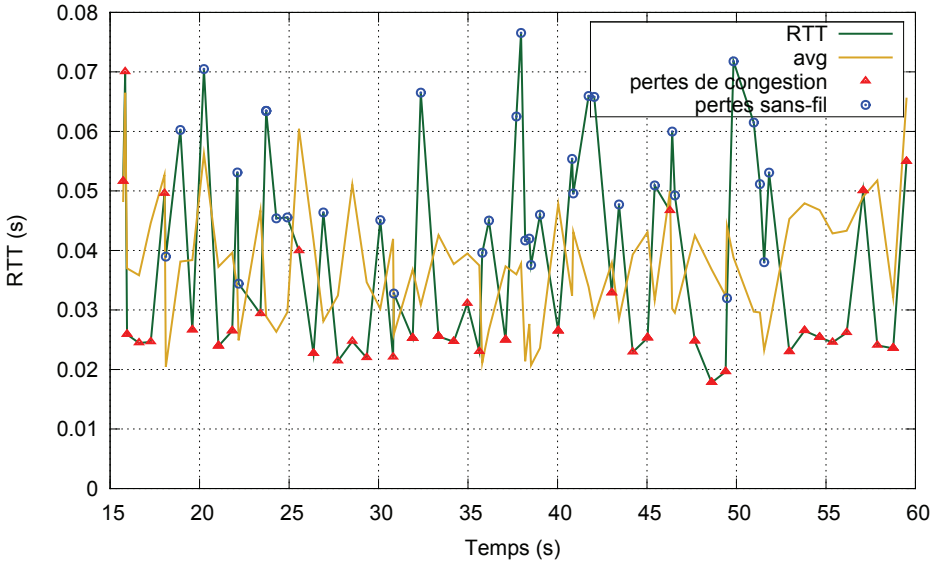


Figure 5.3 – Influence des pertes sans-fil sur le RTT.

ns2 version 2.34 (voir section 6.1 pour plus de détails sur la description des tests). Le protocole de transport utilisé est toujours DCCP avec TCPlike comme contrôle de congestion. TCPlike indique à la source, lors de l'arrivée d'un ACK, quels sont les paquets bien reçus par la destination et quels sont ceux qui sont perdus.

Les simulations ont été préparées de sorte que les informations importantes, telles que les RTT, la moyenne (avg) et la variation (var) de ces RTT, ainsi que la nature des pertes (congestion ou sans-fil) soient facilement extraites et analysées. Cela veut dire que la trace de toutes ces informations pour tous les paquets arrivant après une ou plusieurs pertes a été conservée et analysée après la fin de la simulation. Les résultats sont présentés comme une moyenne de l'ensemble des 51 tests.

La figure 5.4 présente la distribution des pertes de congestion et des pertes sans-fil autour de la moyenne du RTT en utilisant un pas ($step$) d'un dixième de la variation du RTT : cela permet d'avoir plus d'intervalles (environ 42 entre $-2var$ et $+2var$) de petites tailles, ce qui diminue l'incertitude sur la valeur estimée. La valeur du RTT actuel est comparée aux deux extrémités de chaque intervalle pour préciser à quel intervalle de valeurs elle appartient. Par exemple, pour un intervalle i (où $-20 <= i <= +20$) les deux extrémités sont «$avg + i * var/step$» et «$avg + (i+1) * var/step$» ; les étapes à l'extrême deux cas sont «$RTT < -2 * step$» et «$RTT > 2 * step$».

Tout d'abord, ce graphique confirme la conclusion de la section précédente à savoir que les RTT après des pertes de congestion sont généralement plus petits que la moyenne avg, et que les RTT après des pertes sans-fil sont généralement plus grands que la moyenne. À noter également dans ce graphique que pour la plupart des pertes de congestion le RTT est compris entre «$avg - 1.8var$» et «$avg - 1.3var$». Par contre, il est compris entre «$avg + var$» et «$avg + 1.9var$» pour les pertes sans-fil.

Afin d'avoir une meilleure visualisation du seuil séparant les RTT après des pertes de congestion et les RTT après des pertes sans-fil, nous utilisons des RTT cumulatifs. Dans la figure 5.5,

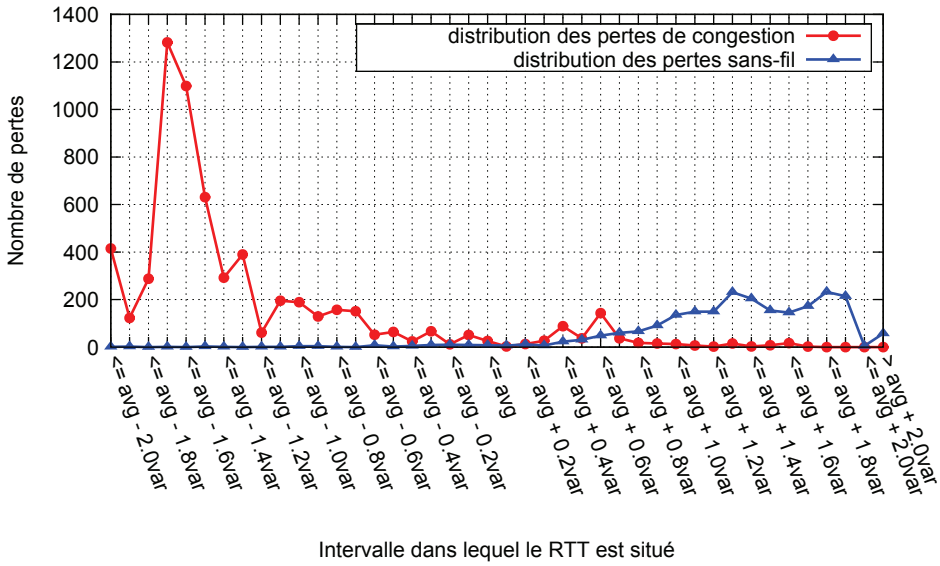


Figure 5.4 – Répartition des pertes en se basant sur des intervalles de RTT.

chaque barre indique le pourcentage des pertes de *congestion* qui donnent une valeur de RTT inférieure à la valeur de son extrémité droite indiquée sur l'axe des x ; autrement dit chaque barre représente la somme de tous les pourcentages précédents plus l'actuel.

À remarquer que dans ce graphique pour les 51 tests, environ 90% des pertes de la congestion sont classifiées si RTT $\leq avg$. Ce pourcentage atteint environ 98% si RTT $\leq avg + 0.6var$ et 100% si RTT $\leq avg + 1.5var$.
Par conséquent, un seuil du RTT inférieur à la moyenne n'est pas l'idéal car même si la grande majorité des pertes de congestion est classifiée, il reste un pourcentage non négligeables qui n'est pas pris en compte et qui provoque une légère augmentation du RTT.

Voyons cette fois du côté des pertes sans-fil. La figure 5.6 présente la même distribution, mais pour les pertes *sans-fil*. Nous pouvons tout de suite constater que seulement 3% des pertes sans-fil se trouvent à gauche du seuil RTT $\leq avg$. Ce même pourcentage est de 8% pour les RTT $\leq avg + 0.6var$ et il est supérieur à 60% pour les RTT $\leq avg + 1.5var$. **Par conséquent, la majorité des pertes sans-fil correspond à une valeur du RTT supérieure à la moyenne**.

Les deux figures montrent aussi qu'il est beaucoup plus fréquent d'avoir des pertes de congestion avec des RTT $\geq avg$ que les pertes sans fil avec RTT $\leq avg$. Enfin, classifier parfaitement toutes les pertes en utilisant RTT est impossible. **Cependant, le choix d'un seuil situé entre** avg **et** $avg + 0.6var$ **permet de classifier correctement la majorité des pertes de congestion et engendre très peu de classifications erronées pour les pertes sans-fil (entre 3% et 8%).**

Comme la valeur de 0,6 est choisie expérimentalement, nous ne pouvons pas soutenir que c'est la *meilleure* valeur pour tous les cas. Néanmoins, nous soutenons que le meilleur seuil devrait être supérieur à avg. Dans le reste de cet article nous allons utiliser un seuil de $avg + 0.6var$.

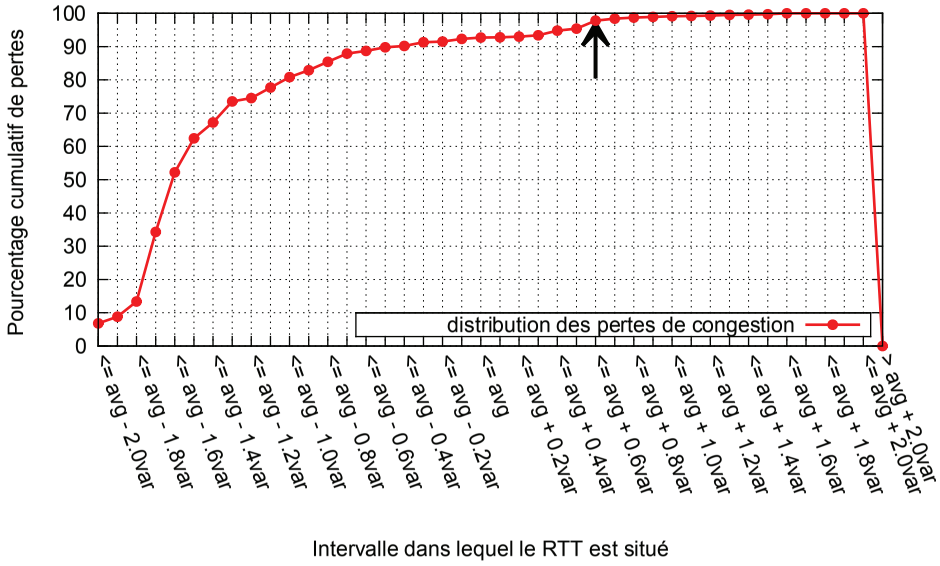


Figure 5.5 – Distribution cumulative des pertes de congestion.

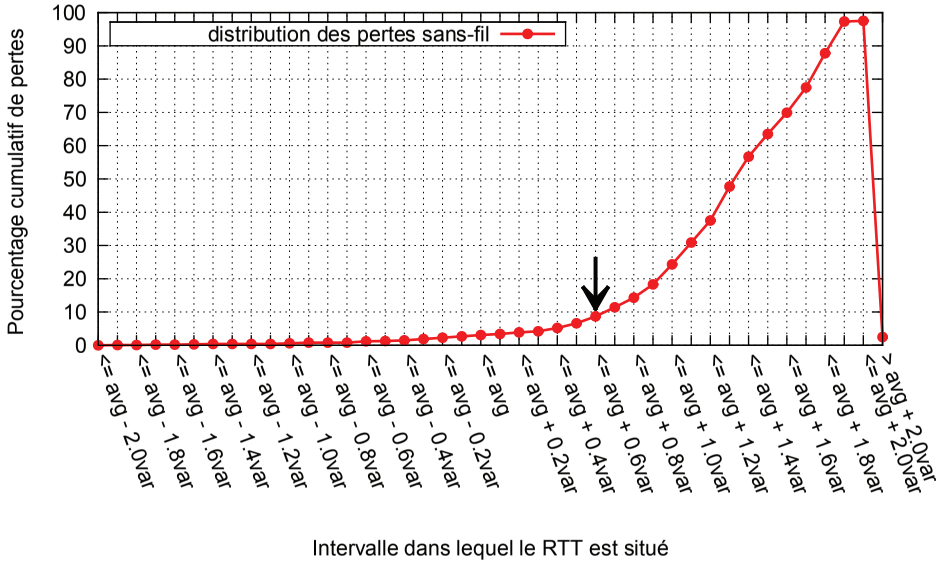


Figure 5.6 – Distribution cumulative des pertes sans-fil.

5.3 RELD (*RTT ECN Loss Differentiation*)

Notre objectif est d'améliorer le contrôle de congestion pour éviter la congestion sur le réseau tout en maintenant un taux d'envoi maximal en cas de pertes sans-fil. Ceci permettrait aux protocoles de transport de pouvoir profiter pleinement de toute la capacité du réseau. Ce nouveau contrôle de congestion fait appel à une méthode de différenciation de pertes (RELD, *RTT ECN Loss Differentiation*) pour arriver à ses fins. RELD utilise le RTT avec ECN afin de différencier les pertes de congestion des pertes sur le canal sans-fil. Cette méthode nécessite par ailleurs que les routeurs intermédiaires entre l'émetteur et le récepteur soient compatibles avec ECN. Pour cela, il est important que les routeurs emploient un gestionnaire actif de la file d'attente *Active queue management* tel que RED (*Random Early Detection*) [FJ93]. Dans cette section, RELD en tant que nouvelle méthode de différenciation de pertes est présentée et détaillée.

5.3.1 Détails et le fonctionnement de RELD

La signalisation ECN permet d'éviter la congestion dans beaucoup de cas mais pas tous. Utiliser une telle signalisation aide à garder le réseau actif et à éviter son effondrement mais cela ne permet pas de différencier les pertes de congestion de celles dues au réseau sans-fil. Pour pallier ce problème, un autre facteur a été étudié, c'est le RTT. Comme nous avons pu le voir dans la section 5.2.3, la valeur du RTT par rapport à sa moyenne et à sa variation permet, en présence des signalisations ECN, de faire la différence entre une perte due à la congestion et une autre due à un problème d'envoi sur le canal sans-fil.

Le fonctionnement de RELD est donc simple : tant que l'émetteur reçoit des acquittements indiquant l'existence de paquets marqués ECN, le réseau semble être congestionné. En conséquence, il réduit son débit. Quand il n'y a pas de paquets marqués et que l'acquittement informe de l'existence d'un ou plusieurs paquets perdus, l'émetteur doit vérifier si ces pertes sont dues à un réseau congestionné ou non avant de diminuer ou non son débit. Ainsi, le RTT est extrait et il est comparé à la moyenne. En prenant en compte le seuil ($avg + 0.6var$) trouvé dans la section 5.2.3, RELD en déduit que la cause des pertes est, soit due aux erreurs du canal sans-fil si le RTT est supérieur à ce seuil et le débit doit être maintenu, soit due à la congestion du réseau et le débit doit être réduit.

En résumé, après la réception d'un acquittement indiquant l'existence d'une ou de plusieurs paquets perdus entre l'émetteur et le récepteur, on considère qu'il y une congestion si et seulement si :

1. $ecn > 0$
   ou
2. $n > 0$ et RTT $< avg + 0.6var$
   où ecn : le nombre de paquets marqués EC, n : le nombre de pertes indiquées dans l'accusé de réception, RTT : le RTT actuel, avg : la moyenne du RTT et var : la variation du RTT.

À noter que les valeurs du RTT moyen et la variation du RTT sont calculées par DCCP. DCCP utilise deux formules (voir le pseudo-code 1 de l'algorithme RELD) pour lisser leurs valeurs comme conseillé dans [Jac88b]. Le lissage (le calcul) est considéré depuis le début de la transmission.

Contrôle de congestion basé sur RELD

RELD peut être implémentée dans n'importe quel protocole de contrôle de congestion tant que l'utilisation d'ECN est permise ainsi que le calcul du RTT, sa moyenne et sa variation. Nous avons choisi de l'intégrer au contrôle de congestion TCPlike de DCCP. Tout comme TCPlike, la connexion commence en phase de démarrage lent et puis la fenêtre de congestion augmente. Lorsque l'émetteur reçoit un acquittement avec une indication de perte, il ne divise sa fenêtre de congestion par 2 que si les pertes sont considérés par RELD comme un signe de congestion (comme décrit sur la formule 2 ci-dessus).

Pseudo-code de l'algorithme RELD

Le pseudo-code de l'algorithme de RELD, qui consiste à calculer le RTT, avg et var et faire la différenciation de pertes est le suivant :

Algorithme 1 – Pseudo-code de algorithme RELD

Paramètres : $\boldsymbol{\alpha = 0.125, \beta = 0.25}$: constantes prédéfinies dans le code DCCP de ns2
Variables :
▷ *RTT : le temps d'aller-retour*
▷ *avg : la moyenne du RTT*
▷ *var : la variation du RTT par rapport à la moyenne*
▷ *ecn* : $= 1$ *si ECN est signalé, et* $= 0$ *sinon*
▷ $n : \geq$ *le nombre de pertes signalées dans l'ACK*
▷ *cwnd : la fenêtre de congestion*

$ecn \leftarrow 0$
$n \leftarrow 0$
pour *chaque ACK reçu* **faire**
  $ecn \leftarrow 1$ si ECN est signalé
  $n \leftarrow$ nombre de pertes indiquées
  $RTT \leftarrow$ temps actuel – le temps d'envoi du paquet acquitté
  $var \leftarrow (1 - \beta) \times var + \beta \times abs(avg - RTT)$
  $avg \leftarrow (1 - \alpha) \times avg + \alpha \times RTT$
  si $(ecn \parallel ((n > 0)\&\&(RTT < avg + 0.6 \times var)))$ **alors**
    $cwnd \leftarrow cwnd/2$ (congestion)
  sinon
    ne rien faire (pertes sans-fil)
  fin
fin

Caractéristiques de RELD

RELD possède les caractéristiques suivantes :

1. il est simple et facile à implémenter ;
2. il utilise des informations disponibles et ne nécessite pas l'implication d'autres couches réseau ;

3. la complexité de la formule 2 est constante car les calculs de RTT, *avg* et *var* sont faits par une équation simple en utilisant des valeurs constantes et seulement deux valeurs enregistrées à chaque fois ;
4. il peut être intégré dans plusieurs contrôles de congestion existants, comme TCP et TCP-Plike, par exemple.

Comparaison entre RELD et TCP-Eaglet

A notre connaissance, il n'y a dans la littérature que *TCP-Eaglet* [BW04a] qui fait appel à ECN pour une classification de pertes. D'une part, *TCP-Eaglet* ne gère pas (c'est-à-dire qu'il ne différencie pas) les pertes arrivant en phase de démarrage lent (*Slow Start mode*), la première phase d'une connexion TCP, où la fenêtre de congestion augmente de façon exponentielle à chaque RTT. D'autre part, il ne prend pas en considération le cas où une rafale de paquets arrive sur un routeur. Dans ce cas, la taille des données peut dépasser soudainement la capacité de sa file d'attente. Par conséquent, il peut y avoir un nombre important de pertes sans marquage ECN même en phase d'évitement de congestion (*Congestion Avoidance mode*), la phase de longue durée d'une connexion TCP, où la fenêtre de congestion augmente linéairement à chaque RTT.

Contrairement à TCP-Eaglet, RELD prend en compte ces situations. D'abord, il ne fait pas de différence entre les modes *Slow Start* et *Congestion Avoidance*. Ensuite, il a recours au RTT pour palier les insuffisances d'ECN. Cela lui permet d'avoir de meilleures performances, comme nous le verrons dans la section 6.2.4.

5.4 Conclusion

Dans ce chapitre, nous avons proposé une méthode générale pour résoudre certains problèmes liés à la faible performance des protocoles de transport dans les réseaux sans-fil. Nous avons expliqué que le contrôle de congestion est plus efficace sur les réseaux sans-fil si une différenciation est correctement effectuée entre les pertes dues aux erreurs du canal sans-fil et les pertes dues à la congestion.

La solution proposée appelée RELD (*RTT ECN Loss Differentiation*), pour « différenciation de pertes basée sur le RTT et l'ECN », constate que, sur un réseau sans-fil où ECN est utilisée, le RTT du paquet arrivant après une perte sans-fil augmente au delà de la moyenne. RELD utilise ainsi un seuil du RTT pour connaitre la nature des pertes et de décider si oui (pour des pertes de congestion) ou non (pour des pertes sans-fil) il faut réduire le débit. RELD considère que le RTT est supérieur à ce seuil si des pertes de sans-fil se produisent. Au contraire, le RTT est inférieur à ce seuil après des pertes de congestion.

Dans le chapitre suivant nous allons évaluer cette hypothèse et montrer l'efficacité de RELD.

Chapitre 6

ÉVALUATION DES PERFORMANCES PAR LA SIMULATION

6.1 Environnement de simulation

Les outils de simulation sont très largement utilisés lors de l'évaluation et la mise au point de nouveaux protocoles. Afin d'évaluer notre contribution, nous avons opté pour l'utilisation du simulateur ns2 version 2.34 (voir la section 2.4 pour plus d'informations sur ce simulateur). La particularité de ns2 pour nos travaux est la présence, à notre disposition, du code source pour le protocole DCCP sous ns2 ainsi bien évidement le code de TCPlike. En réalité, nous avons eu recours au patch écrit initialement par Mattson [Mat04] et maintenu actuellement par nos soins [1].

6.1.1 Modifications apportées aux sources de ns2

Des modifications ont été apportées aux sources de ns2 afin que celles-ci répondent aux besoins spécifiques de nos simulations. La modification la plus importante est le changement de code pour que RED (*Random Early Detection*) fonctionne sur un réseau sans-fil, détaillée par la suite. Les autres changements concernent la modification des fichiers traces de ns2 pour stocker et afficher plus d'informations.

Après avoir réalisé de nombreux tests pour évaluer les performances de RELD dans ns2 sur un réseau sans-fil, nous nous sommes aperçus que le point d'accès ne générait pas de signalements ECN pour informer les extrémités d'une connexion de l'arrivée imminente d'une congestion. Nous nous sommes intéressés à la raison de ce manque de signalement ECN et nous avons trouvé que le code RED (sous ns2) pour la gestion de la file d'attente n'était pas compatible avec les réseaux sans-fil. En réalité, lors de la création du réseau sans-fil, le gestionnaire choisi pour la file d'attente est ajouté sans que ses paramètres ne soient initialisés. Cela peut très bien fonctionner avec un gestionnaire tel que DropTail qui ne nécessite pas une telle initialisation. En revanche, si c'est le modèle *TwoRayGround* qui est ajouté aux nœuds sans-fil, RED fonctionne mais sans

1. `http://eugen.dedu.free.fr/ns2`

signalement ECN. Sinon si c'est le modèle de propagation *Shadowing*, RED ne fonctionne pas car ses paramètres ne sont pas initialisés.

Étant donnée que RELD nécessite, pour son bon fonctionnement, un gestionnaire de la file d'attente qui permet l'utilisation d'ECN sur les routeurs intermédiaires, une modification du code s'avérait indispensable. De cette manière, les paramètres de RED doivent être préalablement initialisés lors de l'ajout de RED dans le point d'accès. Cela se fait dans le fichier « ns-mobilenode.tcl ».

Voici une copie de notre patch permettant de faire fonctionner RED sur un réseau sans-fil (et par la suite avoir des signalements ECN) :

```
--- ns-2.34_orig/tcl/lib/ns-mobilenode.tcl
+++ ns-2.34/tcl/lib/ns-mobilenode.tcl
@@ -373,6 +373,16 @@
  set netif_($t)   [new $iftype]    ;# interface
  set mac_($t)  [new $mactype]      ;# mac layer
  set ifq_($t)  [new $qtype]        ;# interface queue
+ if {$qtype == "Queue/RED"} {
+     set bw_ 54Mb                  ;#ou 11Mb pour un réseau wifi 11Mb/s, etc
+     set link_ [new $lltype]       ;#Créer un nouveau lien
+     $link_ set bandwidth_ $bw_    ;#La bande passante
+     $link_ set delay_ 10ms        ;#Il doit être initialisé et peut avoir
+                                   ;#n'importe quelle valeur (0ms, 10ms ...) sans
+                                   ;#que cela ne change le fonctionnement de RED.
+     $ifq_($t) link $link_         ;#Affecter le nouveau lien à la liaison sans-fil
+     $ifq_($t) set setbit_ true    ;#Il active ECN
+ }
  set ll_($t)  [new $lltype]        ;# link layer
  set ant_($t) [new $anttype]
```

6.1.2 Modèles de pertes et de propagation

Au fur et à mesure, il nous a paru nécessaire de disposer de modèles plus réalistes concernant la propagation des ondes radio et le contrôle d'accès au médium, car nous nous intéressons à l'impact de l'environnement radio réel sur les communications DCCP et les moyens de surmonter les problèmes résultants.

Du point de vue d'un protocole de niveau 4 tel que DCCP, seuls les paquets perdus sont réellement pris en compte. En effet, grâce à la superposition des couches OSI, DCCP n'est pas conçu pour être au courant de l'atténuation du signal radio, des collisions ou des interférences. Toutefois, la façon dont ces pertes apparaissent, leur fréquence et leur distribution dans le temps ont une grande incidence sur le comportement de DCCP et les améliorations que nous proposons dans le présent manuscrit.

Il existe différentes façons de simuler des pertes. Une première solution, très simple, consiste à utiliser un modèle de propagation radio traditionnel (comme les modèles natifs de ns2 *tworayground, Friis* ou *shadowing*), puis d'ajouter un modèle de perte ou d'erreur qui rejette un certain pourcentage fixe de paquets. Ces modèles ne sont pas détaillés, ni suffisamment réalistes pour que nous les utilisions ici, en particulier en raison de leurs distributions simplistes de pertes au fil du temps. De l'autre coté, les modèles qui nécessitent une modélisation extrêmement détaillée de l'environnement (pour une utilisation avec le *raytracing* ou des techniques similaires), produisent des résultats qui ne sont valables que dans le contexte spécifique qui a été modélisé.

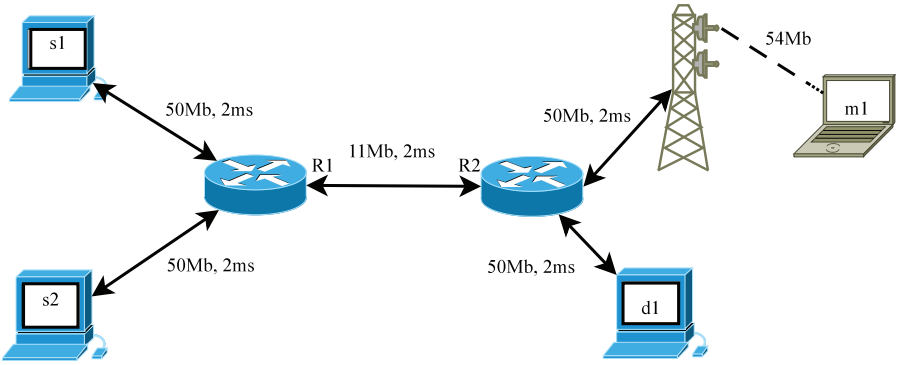


Figure 6.1 – ns2, la topologie du réseau de la simulation.

En raison de ces préoccupations, nous avons décidé d'utiliser un modèle de propagation plus réaliste. C'est le modèle *shadowing-pattern* conçu au sein de notre équipe et décrit dans [DRS06] et [DS07] (voir aussi section 2.7 pour plus d'informations). Il est basé sur le *shadowing* standard, en y ajoutant la capacité de changer la puissance du signal d'une manière rafale, qui à son tour génère des pertes statistiquement réalistes en rafale. Le modèle *shadowing-pattern* a été initialement conçu pour une utilisation dans des réseaux ad-hoc de véhicules, mais il est assez polyvalent. Dans ce modèle les pertes sur la liaison radio sont toutes provoquées par le modèle lui-même. En revanche, il peut être facilement lié ou issu d'un environnement réel.

6.1.3 Topologie du réseau de simulation

La figure 6.1 présente la topologie du réseau employé pour réaliser les simulations. Dans cette topologie, nous utilisons un réseau mixte filaire/sans-fil. Il est composé de deux émetteurs filaires (s1 et s2) établissant des connexions avec deux récepteurs (d1 étant un récepteur filaire et m1 étant un récepteur sans-fil). La liaison entre les routeurs R1 et R2 est de capacité C de 11Mb/s. Les liens entre le routeur R1 et les deux émetteurs (s1 et s2) ainsi que les liens entre le routeur R2 et d'une part le récepteur (d1) et d'autre part le point d'accès (AP) sont d'une capacité de 50Mb/s. Cette capacité est supérieure à C afin que le lien entre R1 et R2 soit le goulot d'étranglement du réseau filaire. Le réseau sans-fil utilise la norme 802.11 avec une bande passante de 54Mb/s. Chaque nœud utilise RED pour la gestion de la file d'attente (les valeurs par défaut de RED sous ns2 sont utilisées). ECN est activé sur tous les routeurs ainsi que pour tous les flux circulant sur ce réseau. La taille du paquet est de 500 octets et le temps de chaque simulation est de 60 secondes. Voir le tableau 6.1 pour les paramètres de ce réseau.

6.1.4 Configuration du modèle de propagation

Dans les sections suivantes et afin d'évaluer nos propositions dans un vaste éventail de contextes réalistes, nous allons exécuter 51 tests différents en utilisant la topologie précédente tout en modifiant les paramètres du modèle de propagation radio.

La différence entre les 51 tests est le niveau de perturbation ajouté au réseau sans-fil. Les perturbations sur le canal sans-fil sont effectuées à l'aide de perturbateurs de type *shadowing-pattern*. Un certain nombre (c'est-à-dire un mélange de zéro jusqu'à trois parmi les sept perturbateurs configurés) de perturbateurs est utilisé dans chacun de ces tests. Le tableau 6.2 montre

Nom du paramètre	Valeur du paramètre
Distance	20m
Mac/802_11: basicRate_	54Mb/s
Mac/802_11: PLCPDataRate_	54Mb/s
Mac/802_11: dataRate_	54Mb/s
Mac/802_11: RTSThreshold_	3000
Queue/RED: setbit_	true
node-config: −ifqLen	Queue/RED
node-config: −adhocRouting	DSDV
node-config: −propType	TSShadowingPattern
TSShadowingPattern: pathlossExp_	4
TSShadowingPattern: std_db_	0

Tableau 6.1 – Paramètres du réseau utilisé pour les simulations.

ces sept perturbateurs et leur puissances lorsqu'ils sont actifs et également quand ils ne le sont pas.

Comme les simulations présentées sont axées sur les réseaux sans-fil standard (un ordinateur portable accède au réseau grâce à un point d'accès (AP)) et que nous nous intéressons aux flux DCCP, nous n'utiliserons que des fréquences relativement élevées (*perturbateurs*), avec des effets individuels ne durant pas plus d'un centième de seconde. Il est évident que les phénomènes qui pourraient empêcher les communications dans le réseau pendant plusieurs secondes, voire quelques minutes sont au-delà du cadre de l'optimisation de notre thèse. Il faut garder à l'esprit que l'effet de chaque *perturbateur* ne dure pas plus de quelques centièmes de seconde et que les effets de plusieurs *perturbateurs* et leurs durées peuvent se chevaucher. Dans un tel événement, les communications peuvent être, bien sûr, bloquées pour une période plus longue.

En raison de la topologie choisie, aucun perturbateur, sauf le numéro 7, n'a d'effet sur le seuil de réception de canal sans-fil s'il est activé de façon isolée. Lorsque deux d'entre eux sont combinés et actifs en même temps, l'atténuation du signal résultant rend le signal sans-fil inférieur au seuil de réception. Cela se traduit par la perte de paquets sur le canal sans-fil. Les perturbateurs numéros 1 et 2 ont la même puissance mais avec un temps d'effet différent (c'est-à-dire 2 est plus fort que 1). Le même constat se fait pour les 3 et 4 (c'est-à-dire 4 est plus fort que 3). Lorsque les perturbateurs 1 et 3 sont combinés avec d'autres, les perturbations obtenues sont très faibles.

De cette manière, nous avons au total un seul test sans perturbation et 50 autres qui provoquent des perturbations (avec un à trois perturbateurs mélangés). Ces 51 tests utilisent les combinaisons suivantes des perturbateurs : 0, 7, 15, 16, 17, 25, 26, 27, 34, 35, 36, 37, 45, 46, 47, 56, 57, 67, 123, 124, 125, 126, 127, 134, 135, 136, 137, 145, 146, 147, 156, 157, 167, 234, 235, 236, 237, 245, 246, 247, 257, 267, 345, 346, 347, 356, 357, 367, 457, 467 et 567. Par exemple, la combinaison 357 utilise les perturbateurs 3, 5 et 7. Cette variété de tests vise à produire un groupe d'environnements sans-fil réaliste, allant des canaux sans-fil sans perturbation jusqu'à ceux qui le sont d'une manière importante.

# perturbateurs	puissance	temps d'inactivité	écart type	temps d'activité	écart type
1	-2	0.03	0.005	0.01	0.01
2	-2	0.01	0.02	0.03	0.01
3	-3	0.03	0.005	0.01	0.01
4	-3	0.01	0.02	0.03	0.01
5	-4	0.03	0.01	0.02	0.01
6	-5	0.04	0.01	0.02	0.01
7	-6	0.04	0.01	0.01	0.01

Tableau 6.2 – Les sept perturbateurs utilisés dans les simulations.

6.1.5 Scénarios de simulation

Deux scénarios de simulation ont été réalisés. Le premier scénario emploie un seul flux sans concurrence aux ressources du réseau avec d'autre flux, c'est-à-direqu'un seul flux occupe toute la capacité disponible de la bande passante du réseau. Nous allons appeler ce scénario par la suite : « scénario sans compétition ». Ce scénario permet de tester dans les cas idéaux les performances de notre proposition sans que d'autres facteurs entrent en jeu pour influencer les résultats obtenus. Cela permet, par ailleurs, d'avoir une meilleure compréhension des modifications apportées.

Le deuxième scénario ajoute un trafic d'arrière-plan au premier scénario afin de faire de la concurrence aux ressources réseau disponibles, c'est-à-dire que plusieurs flux partagent ensemble le débit disponible. Nous allons appeler ce scénario par la suite : « scénario avec compétition ». Ce deuxième scénario permet de tester les performances de notre méthode en présence d'un autre trafic, ce qui donne une idée plus précise du comportement des cas plus réels de la méthode testée.

La concurrence est simulée au moyen d'un flux TCP concurrent qui est ajouté au réseau, entre s2 comme émetteur et d1 comme récepteur. Ce flux apparaît deux fois pendant la simulation: de 1 à 20 secondes et de 25 secondes à 45. Son objectif est de créer du trafic en phase de démarrage lent (*Slow Start*), et cela apparait plusieurs fois au cours de la simulation notamment lorsque les files d'attente sont susceptibles de devenir pleines. Cela augmente la probabilité qu'un paquet soit rejeté sans notification ECN.

Ces deux scénarios ont été exécutés séparément pour chacun des 51 tests expliqués précédemment. Ils ont été aussi répétés trois fois : pour RELD ; pour TCPlike et pour TCP-Eaglet. Les traces de chacun de ces tests ont été sauvegardées et par la suite analysées. Ces traces précisent quel paquet a été perdu et quelle est la nature de sa perte (congestion ou sans-fil), les valeurs de RTT pour les paquets arrivant après une perte et bien d'autres informations.

Les résultats de ces tests sont analysés, présentés et discutés en détail dans la section 6.2.

6.2 Résultats des simulations

Après avoir expliqué en détail l'environnement de simulation dans la section précédente, nous allons dans celle-ci, par le biais d'un vaste nombre de simulations, analyser le seuil de RELD ainsi que son taux de classification afin de savoir si elle distingue bien les pertes de congestion de celles sans-fil. Ensuite, nous allons quantifier le gain de performance par rapport au TCPlike original. Et enfin, comparer RELD et TCP-Eaglet.

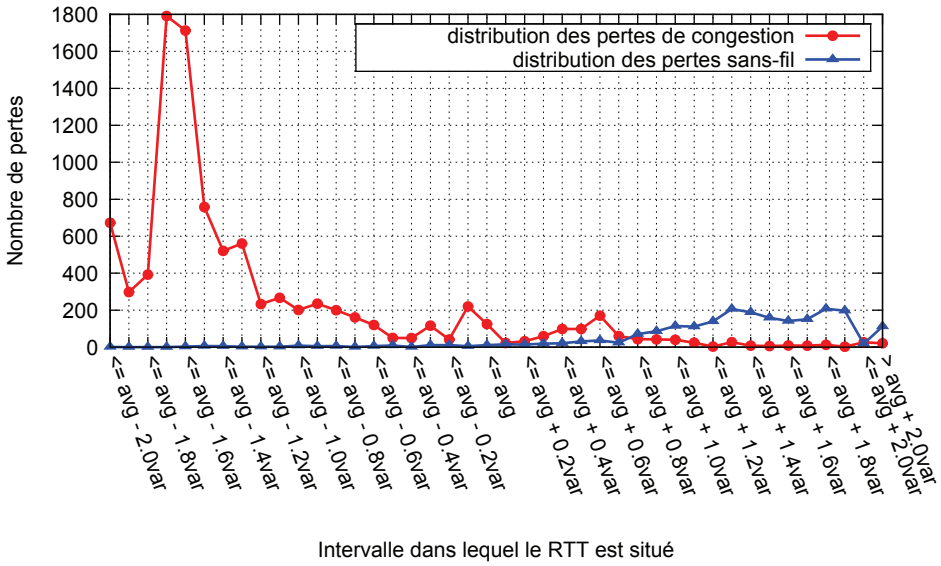


Figure 6.2 – RELD, sans compétition: répartition des pertes par intervalle de RTT.

6.2.1 RELD : vérification du seuil du RTT choisi

Dans la section 5.2.3, un seuil (avg+0.6var) a été choisi à partir des résultats statistiques pour permettre de distinguer efficacement les pertes de congestion des pertes sans-fil. Cette section a pour but, d'une part, de valider ce choix en utilisant cette fois-ci RELD comme contrôle de congestion de DCCP, et d'autre part, de vérifier que ce nouvel algorithme ne modifie pas complétement la pertinence de la différenciation.

Scénario sans compétition

Les résultats de cette analyse sont exposés dans les figures 6.2, 6.3 et 6.4. D'abord, la figure 6.2 confirme les deux conclusions de la section 5.2: le RTT après des pertes de congestion est généralement plus petit que la moyenne RTT, tandis qu'il est généralement plus grand que la moyenne après des pertes sans-fil.

Les figures 6.3 et 6.4 renforcent le choix de $avg + 0.6var$ dans le cas où il n'y a pas de concurrence avec d'autres flux. Comme en témoigne la figure 6.3, seulement 3% des pertes de congestion ne sont pas incluses dans la formule de RELD (c'est-à-dire 3% des pertes ont un RTT $\geq avg + 0.6var$). Pour les pertes sans-fil, la figure 6.4 montre que seulement 10% d'entre elles ne sont pas incluses dans la formule de RELD.

Scénario avec compétition

Les résultats des 51 tests pour les flux RELD avec compétition sont présentés dans les trois figures suivantes : la figure 6.5 présente la répartition des pertes. À noter d'abord qu'il y a deux fois moins de pertes de congestion dans l'intervalle entre $avg - 1.8dev$ et $avg - 1.6dev$, qui représente le pic des pertes, que ceux du même intervalle dans la figure 6.2 (elles sont 1800 dans 6.5 comparées à 900 dans 6.2). Deuxièmement, les pertes de congestion sont moins regroupées

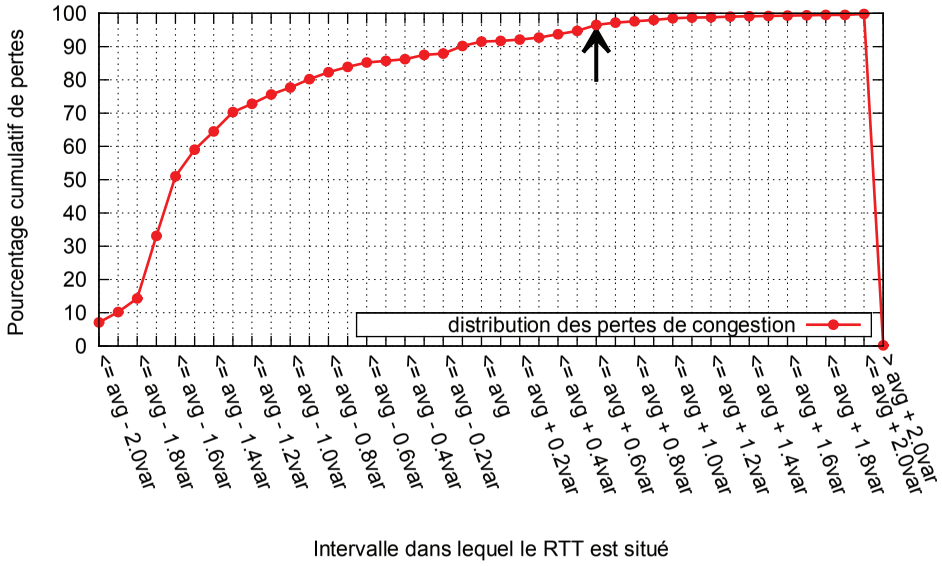


Figure 6.3 – RELD, sans compétition: distribution cumulative des pertes de congestion.

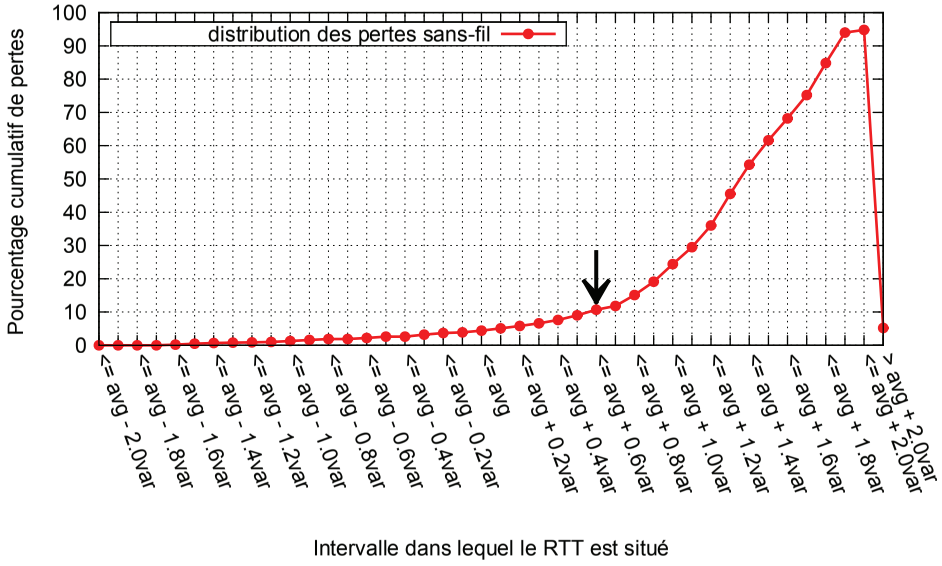


Figure 6.4 – RELD, sans compétition: distribution cumulative des pertes sans-fil.

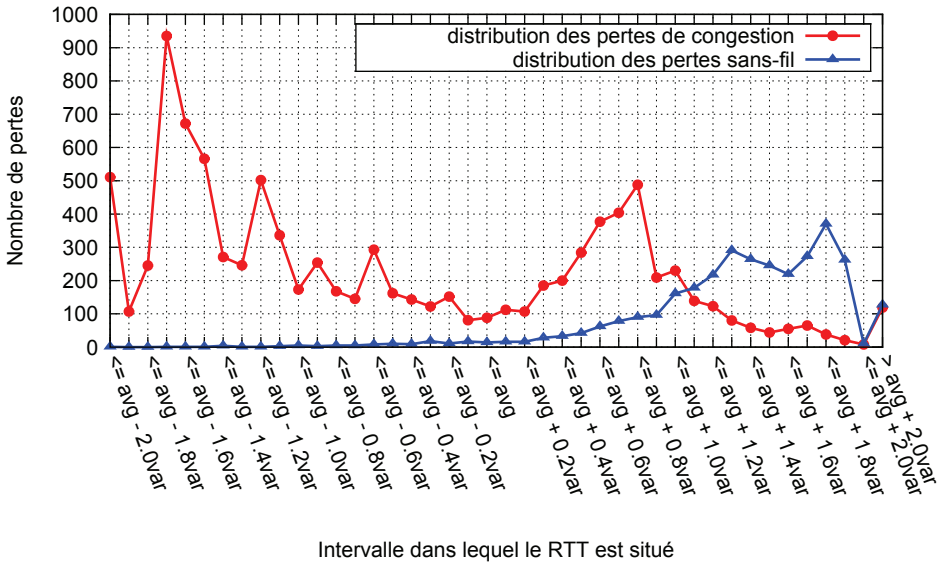


Figure 6.5 – RELD, avec compétition : répartition des pertes par intervalle de RTT.

du côté gauche du graphique. En effet, les valeurs des RTT après des pertes de congestion sont plus équitablement réparties. Troisièmement, ces RTT se déplacent plus vers la droite, ce qui signifie que la différentiation de pertes est plus difficile. Et quatrièmement, comme avant, la plupart des pertes dues à la congestion sont à gauche et celles dues aux erreurs du canal sans-fil sont à droite.

La figure 6.6 révèle que moins de pertes de congestion (78%) sont incluses dans la formule de RELD (RTT $\leq avg + 0.6var$). Bien sûr, le choix d'un plus grand seuil peut réduire les pertes de congestion ayant une classification erronée, mais cela se traduira par un taux d'erreur supérieur de classification sans-fil.

Enfin, pour les pertes sans-fil, présentées sur la figure 6.7, la même distribution du RTT est remarquée, ce qui est normal parce que les mêmes perturbateurs sont utilisés.

La conclusion de cette section est que le seuil de RELD permet de classifier correctement environ 78% des pertes de congestion (en cas de compétition avec d'autres trafics) et 97% (dans le cas sans compétition). En ce qui concerne les pertes sans-fil, ce pourcentage est environ 90% dans les deux cas.

6.2.2 RELD : évaluation de la viabilité de classification

Dans cette section, nous évaluons les performances de RELD et sa capacité à distinguer les pertes de congestion de celles sans-fil. Cette performance est évaluée pour un grand nombre de conditions du canal sans-fil. Ces conditions sont exprimées par les identifiants des perturbateurs utilisés pour une simulation donnée. « 0 » signifie que le perturbateur numéro 0 a été utilisé, « 234 » signifie que les perturbateurs 2, 3 et 4 ont été utilisés conjointement et ainsi de suite.

La classification utilise plusieurs paramètres, expliqués dans la suite. Supposons que lors d'un test les résultats obtenus sont les suivants : il y a 80 pertes de congestion et 20 pertes dues aux erreurs du canal sans-fil. Parmi les pertes de congestion, 70 d'entre elles sont correctement

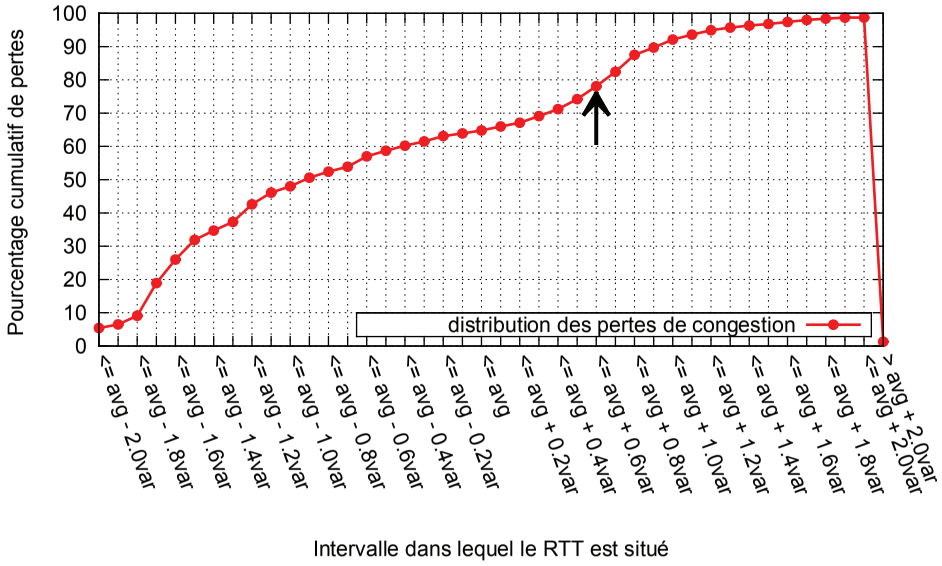


Figure 6.6 – RELD, avec compétition : distribution cumulative des pertes de congestion.

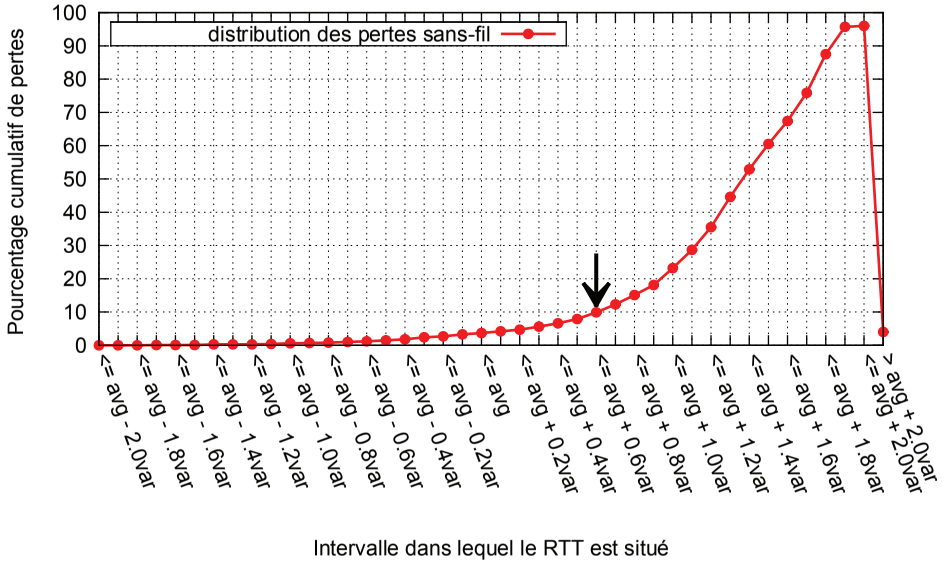


Figure 6.7 – RELD, avec compétition : distribution cumulative des pertes sans-fil.

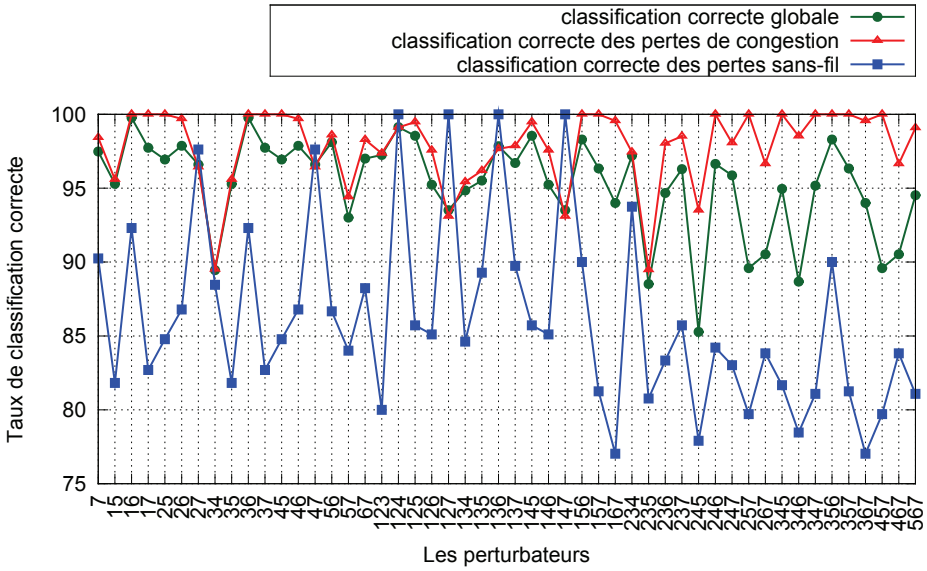


Figure 6.8 – RELD, sans compétition: le taux de classification correcte.

identifiés comme dues à la congestion, et 10, à tort comme des pertes sans-fil. Pour les pertes sans-fil, 5 sont mal classifiées et 15 sont correctement reconnues. Ceci est repris dans le tableau suivant (c signifie des pertes de congestion, w signifie des pertes sans-fil) :

Réelles	80c		20w	
Classifiées	70c	10w	5c	15w

Le pourcentage de classification correcte des pertes de congestion est de 70/80=87%, et celui des pertes sans-fil est de 15/20=75%. Le pourcentage de classification correcte total est le nombre global de pertes correctement classifiées divisé par le nombre total de ces pertes : $(70 + 15)/(80 + 20) = 85\%$.

La vraie raison pour chaque perte de paquets (congestion ou sans-fil) est extraite des fichiers de trace de la simulation. D'autre part, la raison supposée par la source, qui utilise la classification RELD, est affichée en modifiant le code source ns2.

Scénario sans compétition

La figure 6.8 présente trois courbes : la première représente le pourcentage de classification correcte pour l'ensemble des paquets perdus, toute sorte confondue. La deuxième et la troisième représentent le taux de classification correcte des pertes de congestion et sans-fil respectivement. En fait, la première courbe est une moyenne pondérée des deux autres courbes.

D'abord, nous pouvons constater que le taux de classification correcte global varie entre 85% et environ 99% pour la plupart des cas, et il est d'environ 92% en moyenne. Ensuite, le taux de classification correcte des pertes de congestion dans ce scénario sans compétition est très élevé (environ 97%) grâce au seuil de RELD qui identifie la majorité d'entre elles. D'autre part, le taux de classification correcte des pertes sans-fil est lui aussi élevé, il varie entre 78% et 100%.

La figure 6.9 donne le nombre de paquets perdus dans chaque test de simulation. Comme le montre cette figure, le nombre de pertes de congestion est très faible puisqu'il représente environ

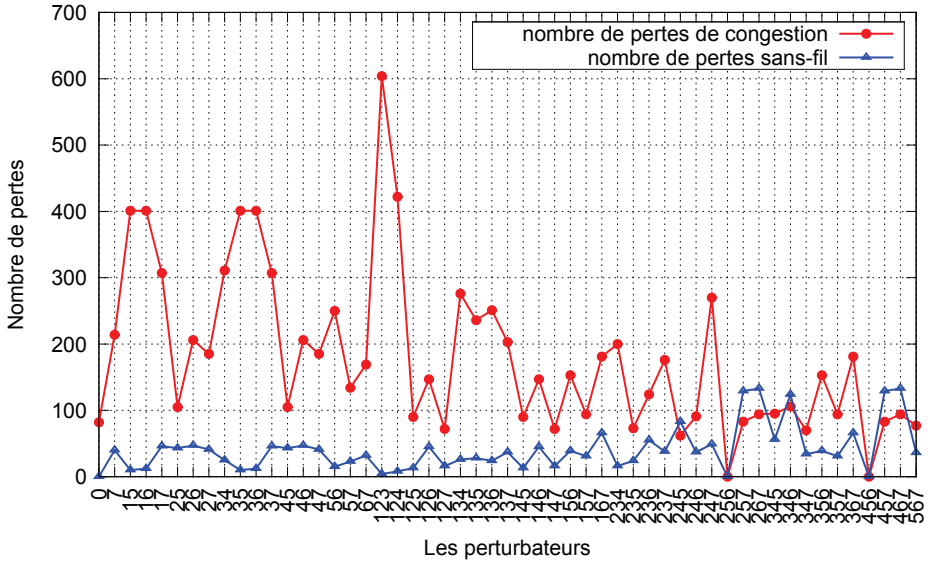


Figure 6.9 – RELD, sans compétition: nombre de pertes de congestion et sans-fil.

600 pertes, comparées au nombre de paquets envoyés entre 15000 et 115000. Ceci signifie que le rapport entre les paquets reçus et les paquets envoyés est très élevé. En d'autres termes, puisque le protocole de transport utilisé est DCCP qui a été spécialement conçu pour les transmissions multimédia, ce rapport s'avère très important. En effet, un rapport élevé diminue la nécessité de retransmettre des paquets perdus, d'ailleurs un grand nombre de paquets perdus conduit à une dégradation de la qualité de la média transmise.

Scénario avec compétition

Contrairement à la figure 6.8, la figure 6.10 indique que le taux de classification correcte des pertes sans-fil est supérieur à celui des pertes de congestion. Ce résultat s'explique par le fait que le seuil de RELD tente d'équilibrer la classification de chaque type de pertes. Dans tous les tests, le taux de classification correcte est très élevé, 80% en moyenne pour la totalité des pertes ainsi que pour les pertes de congestion. Le taux correct augmente lorsque le *nombre* de pertes sans-fil augmente, par exemple dans des tests où le numéro de l'identifiant des perturbateurs actifs est supérieur à 123 (voire le paragraphe suivant).

Le nombre de paquets perdus (congestion et sans-fil) est donné dans la figure 6.11. Nous pouvons constater que le nombre de paquets perdus à cause de la congestion est très petit. Il est de l'ordre de quelques centaines de paquets comparé à quelques dizaines de milliers de paquets envoyés. Par conséquent, le taux de paquets reçus est aussi très élevé.

Nous concluons alors que même si le taux de classification dans la simulation avec compétition est moindre, **RELD ne provoque par une importante quantité de paquets perdus.**

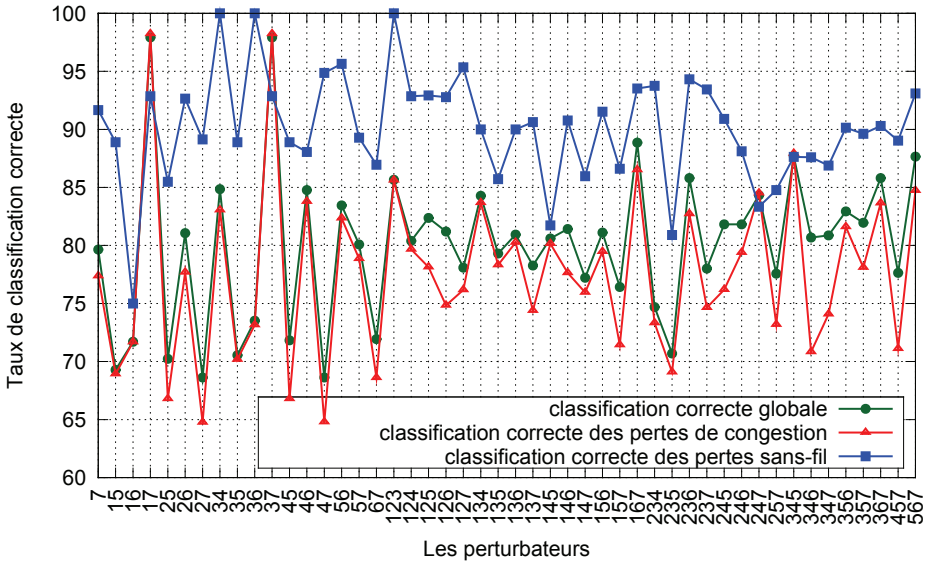


Figure 6.10 – RELD, avec compétition : le taux de classification correcte.

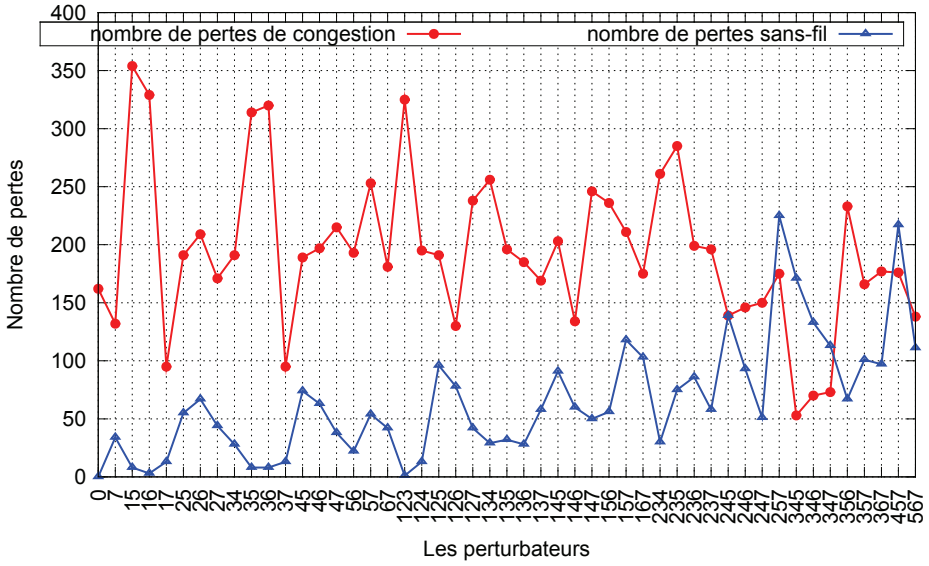


Figure 6.11 – RELD, avec compétition : nombre de pertes de congestion et sans-fil.

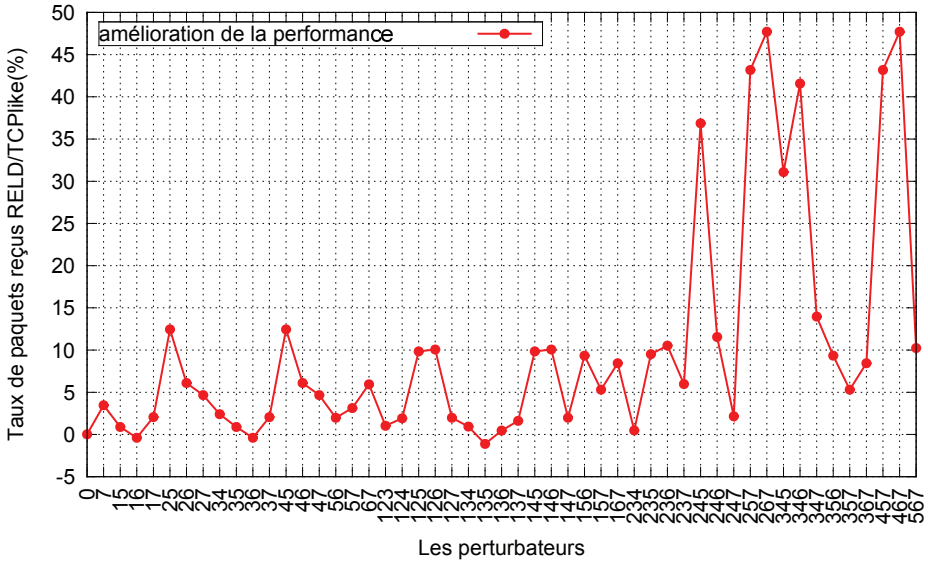


Figure 6.12 – RELD vs TCPlike, sans compétition: l'amélioration de performance.

6.2.3 RELD vs TCPlike

Une méthode de classification de pertes est censée améliorer les performances du protocole de transport utilisé dans un environnement sans-fil. Pour vérifier l'amélioration des performances apportée à DCCP par l'algorithme de classification de pertes RELD, nous comparons le nombre de paquets reçus par le récepteur avec TCPlike et avec RELD pour le même réseau. Les résultats sont exprimés comme un ratio entre le nombre de paquets reçus par le récepteur RELD et par le récepteur TCPlike. Pour rendre la valeur plus lisible, nous en enlevons 1 et puis le reste est multiplié par 100. La dernière valeur obtenue indique ainsi l'amélioration (si elle est positive), l'égalité (si elle est égale à zéro) ou la dégradation des performances (si elle est négative).

Scénario sans compétition

Les résultats de comparaison des tests sont présentés dans la figure 6.12. D'abord, il est à noter que, sauf dans de très rares cas, le ratio de paquets reçus/envoyés est plus grand pour RELD que pour TCPlike. Deuxièmement, l'amélioration est élevée, avec une moyenne de 10% et un maximum de 47%. Troisièmement, le gain de performance est plus élevé lorsque le nombre de pertes sans-fil est très grand, les perturbateurs 257, 267, 346, 457 et 467 ont le plus grand nombre de pertes sans-fil (comme vu aussi dans la figure 6.9) et ce sont ceux-la même aussi qui ont le gain le plus élevé de performance. Au contraire, la plus faible amélioration apparaît lorsque le nombre de pertes de congestion est élevé et le nombre de pertes sans-fil est faible, à savoir les perturbateurs 35, 36 et 135.

Scénario avec compétition

Dans le cas du scénario avec compétition, les résultats sont présentés dans la figure 6.13. Des résultats obtenus sont encore meilleurs de ce que nous avons obtenu dans le scénario sans

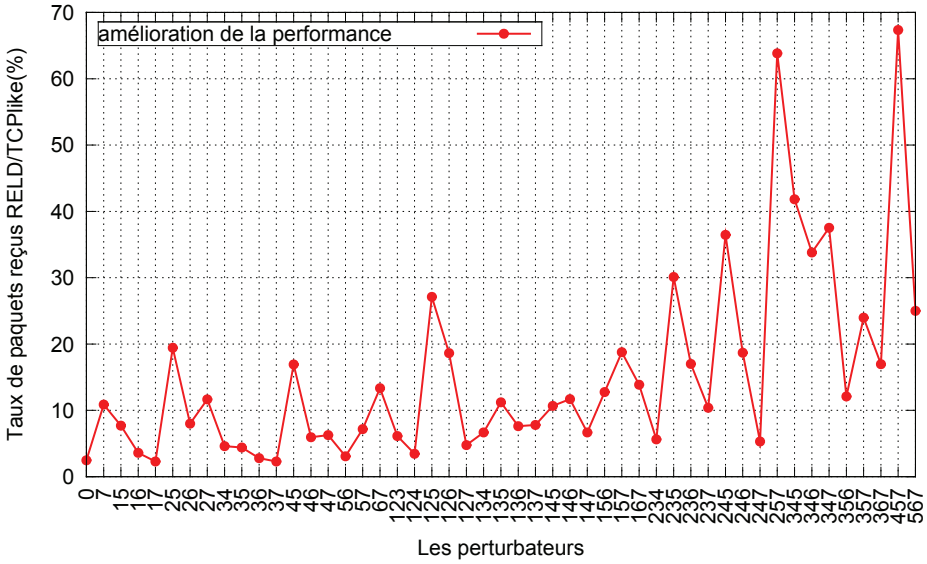


Figure 6.13 – RELD vs TCPlike, avec compétition : l'amélioration de performance.

compétition. Nous avons 15% de paquets reçus en plus en moyenne et 68% en maximum.

La conclusion est que RELD améliore la performance du protocole de transport à la fois avec et sans compétition. Le gain de performance est plus élevé lorsque les pertes sans-fil sont plus nombreuses, et il est plus petit lorsque les pertes de congestion sont élevées.

6.2.4 RELD vs TCP-Eaglet

TCP-Eaglet utilise ECN afin de différencier les pertes de congestion de celles dues au sans-fil. Cette méthode considère que si l'expéditeur n'est pas en phase de démarrage lent (c'est-à-dire qu'il est en phase d'évitement de congestion), il estime la perte comme une perte sans-fil et ne réduit pas de moitié la taille de sa fenêtre de congestion (voir section 4.2.5). Pour TCP-Eaglet, un paquet perdu, sans une signalisation ECN, pendant la phase d'évitement de congestion signifie une perte sans-fil, tandis qu'un paquet marqué ECN signifie une congestion. Nos résultats montrent que cette hypothèse n'est pas toujours valable, en particulier dans un environnement sans-fil très perturbé, et, en tant que telle, cette méthode peut conduire à la congestion dans le réseau.

La version originale de TCP-Eaglet a été conçue pour le protocole TCP, et non pour DCCP. Cependant, nous avons voulu comparer notre approche avec cette idée. Ainsi, nous avons implémenté TCP-Eaglet sous DCCP dans ns2. Dans cette section, nous comparons RELD et TCP-Eaglet dans un scénario sans compétition. Comme cela a été fait pour RELD (figures 6.8 et 6.9), les résultats de classification correcte et le nombre de pertes de congestion et sans-fil sont présentés pour TCP-Eaglet (figures 6.14 et 6.15).

TCP-Eaglet a une moyenne de 95% de taux de classification correcte pour les pertes sans-fil (voir figure 6.14). Or, le taux de classification correct de TCP-Eaglet est très faible, seulement 4,4% en moyenne (même figure). En réalité, il y a environ entre 4000 et 18000 de paquets qui

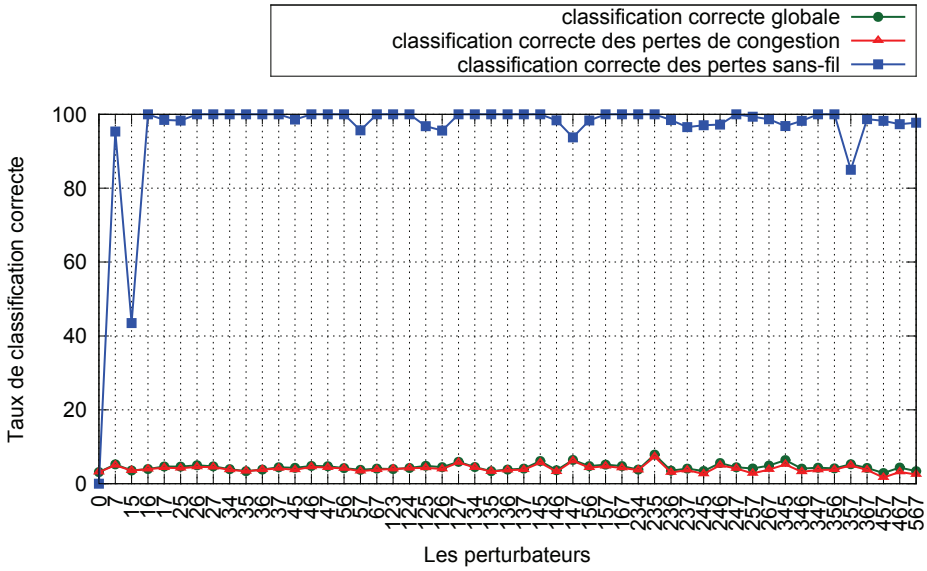


Figure 6.14 – TCP-Eaglet, sans compétition: le taux de classification correcte.

sont perdus (voir figure 6.15) car TCP-Eaglet classifie les pertes sans signalement ECN comme des pertes sans-fil ce qui n'est pas correct.

Étant donné que les résultats de TCP-Eaglet ne sont pas bons, nous n'avons pas testé ses performances dans le scénario avec compétition.

Discussion

Pour tous les tests, les paramètres par défaut de RED ont été utilisés. TCP-Eaglet ([BW04b]) indique que ces paramètres doivent être ajustés, ce qui pourrait être l'une des raisons de ses mauvais résultats.

Une deuxième raison est que, dans les réseaux filaires, un paquet marqué ECN a une faible chance d'être perdu par la suite. En revanche, une telle perte peut survenir dans les réseaux sans-fil, ceux-ci sont en effet connus pour connaître un nombre élevé de pertes par moments. La figure 6.16 montre un tel exemple, où l'émetteur s1 envoie 3 paquets, p1, p2 et p3 et le routeur R1 marque le bit EC de p2 (*Experienced Congestion*), mais tous les paquets sont rejetés par la suite sur le canal sans-fil à cause d'une interférence. Quand un paquet précédemment marqué ECN est perdu sur un lien sans-fil, la notification ECN n'arrive pas à l'émetteur, qui va continuer à augmenter son débit d'envoi comme d'habitude.

Un exemple extrême est le suivant, comme indiqué dans la section 2.3, RED a trois états (quatre états pour « gentle RED »): tous les paquets passent, les paquets sont marqués de manière probabiliste, les paquets sont rejetés. Si par malchance tous les paquets *marqués* au cours de $q_{th\_min} < q_{ave} < q_{th\_max}$ sont perdus sur le canal sans-fil plus tard, l'émetteur ne reçoit pas de notifications ECN et donc il continue à augmenter son taux d'envoi. Par conséquent, la taille de la file d'attente continue d'augmenter elle aussi ce qui provoque une hausse de la taille moyenne de cette file et c'est au-delà du son seuil maximum. Autrement dit, RED entre donc dans l'état $q_{ave} > q_{th\_max}$. Dans cet état, tous les paquets sont rejetés (sans marquage/notification). Étant

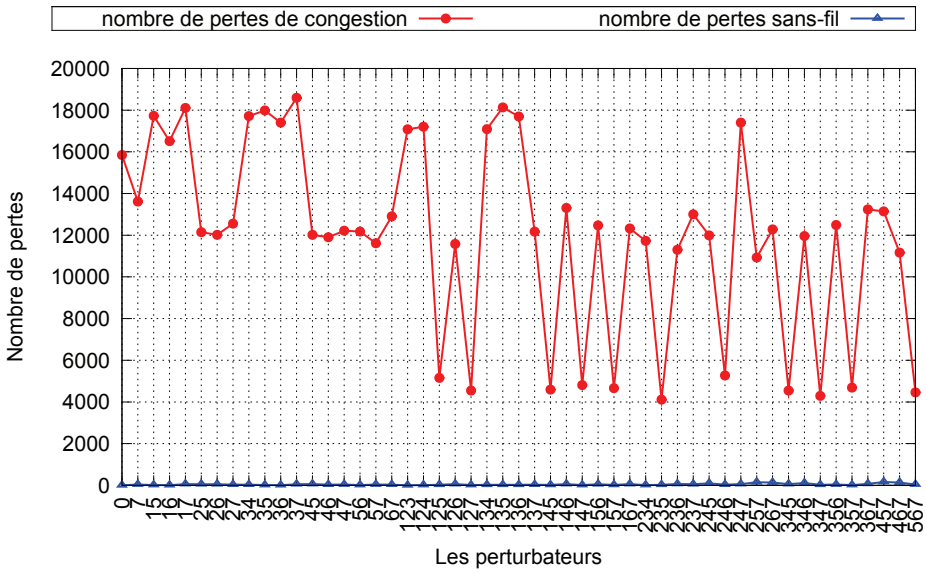


Figure 6.15 – TCP-Eaglet, sans compétition: nombre de pertes de congestion et sans-fil.

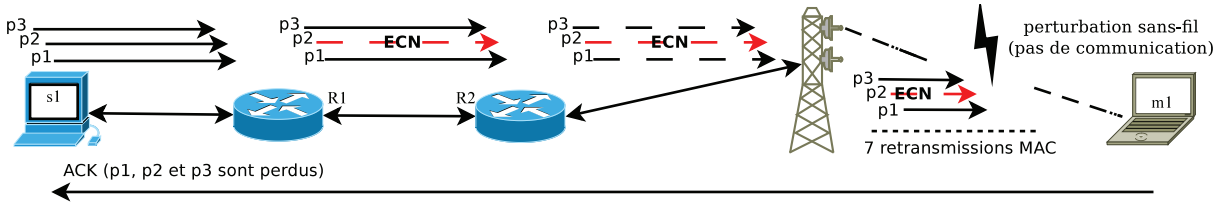


Figure 6.16 – Un paquet marqué ECN peut être perdu par la suite sur un canal sans-fil.

donné que l'émetteur TCP-Eaglet ne reçoit pas de notification ECN pour ces derniers paquets, il continue à augmenter le taux d'envoi comme d'habitude, conduisant à de nombreuses pertes.

De son côté, RELD compense ce problème grâce à l'utilisation d'un autre facteur (le RTT). Dans ce cas, lorsque les paquets marqués sont perdus sur le canal sans-fil, le RTT va augmenter, et RELD se comporte comme TCP-Eaglet. Toutefois, pour les dernières pertes (pendant la phase $q_{ave} > q_{th\_max}$), comme elles sont dues à la congestion, leurs RTT diminue, et RELD les considère comme dues à la congestion, par conséquent, il diminue son débit d'envoi.

Cela confirme que l'utilisation d'ECN seul afin de distinguer le type de pertes n'est pas satisfaisante. Ainsi, l'émetteur RELD réagit plus efficacement à la congestion que TCP-Eaglet.

6.3 Conclusion

Dans le but d'améliorer les performances des protocoles de transport dans les réseaux sans-fil, nous avons proposé dans le chapitre 5 une méthode de différenciation de perte appelée RELD. Ce chapitre a été consacré à l'évaluation et la validation par la simulation sous NS2 des performances de cette contribution.

Nos résultats statistiques réalisés par des simulations qui utilisent un modèle réaliste d'erreur sur le canal sans-fil, confirment que le RTT augmente en cas de pertes dues au canal sans-fil. Ainsi nous avons pu, tout d'abord, montré que RELD différencie bien les pertes de congestion et les pertes sans-fil en utilisant le seuil du RTT étudié dans le chapitre 5. Deuxièmement, nous avons validé la viabilité de RELD et montré qu'elle atteint généralement des taux de classification correcte très élevés (entre 80% et 97% pour les pertes de congestion et d'à peu près 90% pour les pertes sans-fil). Troisièmement, nous avons souligné que RELD surpasse TCPlike original sur lequel est basée l'implémentation de son algorithme. L'utilisation de RELD permet d'atteindre une amélioration d'environ 68% de la performance du protocole du contrôle de congestion.

Dans ce chapitre, nous avons expliqué également que l'utilisation d'ECN, qui est proposée afin d'éviter la congestion sur un réseau filaire, ne suffit pas pour distinguer les pertes sur un réseau sans-fil. Notre méthode qui utilise le RTT en combinaison avec la signalisation ECN est ainsi un moyen efficace pour réaliser la différenciation de pertes.

Troisième partie

Amélioration du transfert du contenu multimédia par adaptation de la vidéo au niveau de la couche application

Chapitre 7

INTRODUCTION

7.1 Contexte

Aujourd'hui, le nombre de vidéos encodées en plusieurs bitrates et accessibles à tout le monde augmente de façon significative. En outre, la qualité d'une vidéoconférence peut beaucoup varier en fonction de la qualité d'encodage choisie par l'expéditeur. Leur contenu est généralement livré à l'utilisateur final en utilisant des services de *streaming* sur Internet. Ces services, ainsi que le nombre de clients demandant une meilleure qualité de la vidéo sont en constante progression. Les nouvelles normes pour la vidéo comme les vidéos en HD et en 3D demandent de plus en plus de bande passante. Les variations du débit disponible doivent également être prises en compte afin de raccourcir le temps de chargement de la vidéo en mémoire tampon à la destination.

Regarder de telles vidéos sur un réseau sans-fil n'est pas une expérience agréable, car différentes technologies de réseau sans-fil avec des caractéristiques différentes cohabitent et évoluent au fil du temps. Ainsi, le débit disponible n'est pas toujours stable pour de nombreuses raisons (interférences, mobilité etc.) En outre, le partage de la bande passante entre plusieurs utilisateurs pourrait rendre le débit disponible inférieur au bitrate requis pour la vidéo (que ce soit sur un réseau filaire ou sans-fil).

Actuellement, une vidéo avec un seul bitrate est choisie au début d'une transmission de vidéo en *streaming* ; la transmission est contrôlée au niveau de la couche réseau (TCP ou UDP) et l'application n'est pas du tout impliquée. Ainsi, deux choix existent lors de la lecture d'une vidéo en contenu : le premier, et le cas général, est de choisir un bitrate bas et la vidéo est jouée directement sans interruption car le tampon de réception est alimenté de façon continu et donc ne se vide pas. Le deuxième, est de choisir un bitrate plus élevé que le débit disponible moyen, charger les données vidéo en mémoire tampon chez l'utilisateur, et lire la vidéo lorsque la mémoire tampon a suffisamment de données. Lorsque ce tampon se vide l'utilisateur sera confronté à de nombreux *play/pause* pendant l'affichage. Les deux cas sont désagréables à l'œil de l'utilisateur : le premier a l'avantage de regarder une vidéo fluide, mais de faible qualité, tandis que le second permet d'avoir une bonne qualité mais avec un temps d'attente plus ou moins long et avec de fréquentes interruptions. Les utilisateurs qui souhaitent avoir un accès instantané sans délai au contenu multimédia et avec la meilleure qualité possible peuvent se

servir d'un nouveau type de services de transmission multimédia en continu : « l'adaptation de contenu multimédia » (*multimedia content adaptation*).

Pour le *streaming* vidéo en réseau avec un débit disponible très variable, *l'adaptation de contenu* est un moyen d'adapter le bitrate de la vidéo transmise aux caractéristiques du réseau (c'est-à-dire aligner le bitrate au débit disponible). Elle permet d'améliorer la qualité de la vidéo reçue par l'utilisateur final. Plusieurs solutions sont envisageables afin de réaliser l'adaptation. Une solution simple est d'utiliser une approche coopérative entre la couche application et une autre couche OSI (par exemple la couche réseau ou la couche transport). Le protocole de transport gère le contrôle de congestion côté réseau, tandis que l'application contrôle le bitrate de la vidéo côté serveur. Ce contrôle de bitrate peut se faire en changeant les paramètres de la vidéo transmise (par exemple, quantification, nombre de trames par seconde FPS (*Frames Per Second*), etc.)

7.2 Choix du protocole de transport

De plus en plus d'applications réseau (par exemple, les applications en temps réel comme le *streaming* audio et vidéo) peuvent accepter un certain niveau de paquets perdus. D'une part, celles qui utilisent le protocole TCP (*Transmission Control Protocol* [Pos81]) doivent payer le prix de sa fiabilité, avec une grande latence. D'autre part, celles qui ont recours à UDP (*User Datagram Protocol* [Pos80]) doivent faire face à son manque de contrôle de congestion. RTP (*Real-time Transport Protocol* [SCFJ03]), protocole largement utilisé pour la transmission multimédia, est un protocole au niveau application. En tant que tel, il fonctionne au-dessus d'un protocole de transport, tels que TCP ou UDP, et n'est donc pas concerné par les problèmes des protocoles de transport.

D'autres protocoles de transport prometteur pour ces applications tels que DCCP (*Datagram Congestion Control Protocol*), récemment normalisé en tant que RFC4340 [KHF06], et SCTP (*Stream Control Transmission Protocol*) sont apparus. Ils peuvent être considérés comme un TCP sans fiabilité et sans acheminement ordonné obligatoire des paquets (deux points essentiels en *streaming* vidéo, qui le rendent plus adapté pour la diffusion de données multimédia), ou encore comme un UDP muni d'un contrôle de congestion.

Selon l'étude de comparaison faite dans [ZMA+09], DCCP a un meilleur débit par rapport à SCTP et UDP. Il a également le plus petit délai de transmission ainsi que la plus petite gigue (*jitter*). DCCP assure mieux que les autres la qualité de service demandée pour un protocole de transport conçu pour transmettre les données vidéos en continu.

Pour notre propos, DCCP possède deux points intéressants: premièrement, il permet de choisir le contrôle de congestion utilisé lors de la communication, deuxièmement, il utilise les accusés de réception. Parmi les trois contrôles de congestion actuellement normalisés, TFRC (*TCP-Friendly Rate Control*) est le plus adapté à la vidéo en *streaming* [FHPW08]. Comme décrit dans [KHF06], DCCP implémente des connexions bidirectionnelles et en *unicast*. Il possède, en outre : la négociation du mécanisme de contrôle de congestion et du mécanisme d'acquittement afin de communiquer à la source des informations concernant les paquets perdus et les paquets marqués ECN (*Explicit Congestion Notification*).

Pour les protocoles ci-dessus, en particulier DCCP et TCP, la transmission vidéo est contrôlée au niveau de la couche transport et l'application n'est pas du tout concernée. Pour le *streaming*

vidéo en réseau avec un débit disponible très variable au cours d'une connexion, une adaptation de la vidéo aux caractéristiques du réseau semble être cruciale. Une approche coopérative entre la couche application et la couche transport peut améliorer la qualité de la vidéo perceptible par l'utilisateur final.

7.3 Contributions

La méthode d'adaptation de la vidéo que nous proposons (VAAL, *Video Adaptation at Application Layer*, « Adaptation de la vidéo au niveau de la couche application ») utilise le débordement du tampon du protocole de transport comme solution pour la couche application pour découvrir le débit disponible et adapter le bitrate de la vidéo transmise au débit disponible estimé en conséquence. Toutes les n secondes, l'application serveur calcule le pourcentage du nombre de paquets qui n'ont pas pu être écrits dans le tampon. Ce pourcentage est utilisé par la suite pour contrôler le bitrate de la vidéo. Un pourcentage élevé signifie un dépassement du débit disponible et le bitrate devrait diminuer. Un pourcentage nul indique soit un débit disponible égal au bitrate soit plus grande ; ainsi, le bitrate de la vidéo envoyée peut être augmenté. De cette manière, le débit de la vidéo regardée est fixe au cours de chaque période de n secondes, et peut changer uniquement entre les périodes.

L'adaptation précédente, connue dans la littérature sous le nom de (*rate adaptive video control*, « contrôle de la vidéo par débit adaptatif »), peut être réalisée en contrôlant d'autres paramètres de la vidéo, comme le nombre d'images par seconde (FPS, *Frames Per Second*) et la résolution de la vidéo. Ces possibilités peuvent générer une vidéo hétérogène chez le client, et l'application cliente doit être capable de reconstituer une vidéo lisible.

Les trois paramètres présentés précédemment permettent d'optimiser *la partie réseau* du *streaming* de la vidéo. Pour des résultats encore meilleurs, ces méthodes *pourraient être associées à d'autres méthodes*. Par exemple, utiliser une méthode de différentiation de perte telle que RELD sur un réseau sans-fil (voir le chapitre 5.3.1 pour plus d'informations sur comment fonctionne RELD). Un autre exemple intéressant sur les liens avec perte est la FEC (*Forward Error Correction*) ([SYL03] par exemple). De plus, *des techniques spécifiques à la vidéo* permettent par exemple de donner la priorité [GAA05] ou de ne retransmettre [HI06] que les paquets importants (paquets I dans une vidéo encodée en MPEG) côté serveur.

Le contrôle du débit n'est pas une idée nouvelle pour améliorer la transmission vidéo. Dans la littérature, il existe une multitude d'articles sur ce sujet. La plupart d'entre eux ont été écrits il y a 5 à 10 ans. À l'époque, les solutions utilisant de nouveaux protocoles de transport modifiés ont été utilisées (telle que [Kaz02], qui *n'est pas déployée en réalité* et risquée du point de vue contrôle de congestion). En outre, peu d'entre elles utilisent des expérimentations. Les documents que nous avons trouvés sur le contrôle de débit au dessus de DCCP sont [Hau07, LK08], et ils utilisent des simulations.

Dans ce contexte, ce manuscrit fait partie des *rares documents qui analysent l'adaptation du contenu vidéo* ([Hau07] en est un autre), et c'est *le premier qui utilise DCCP avec de véritables expérimentations dans un réseau sans-fil*. En outre, c'est la première fois que *le débordement du tampon est proposé pour l'adaptation de la vidéo*. Notre solution est *très simple à déployer*, seule l'application, du côté émetteur, doit être modifiée (il n'est pas nécessaire de modifier le récepteur, ni le protocole de transport). En corollaire, notre méthode fonctionne avec n'importe quel protocole de transport qui a un contrôle de congestion.

7.4 Motivations

Dans cette section, nous montrons les avantages de l'adaptation vidéo et pourquoi elle est plus intéressante que la transmission vidéo classique.

Aujourd'hui, le *streaming* vidéo est fait comme un transfert inélastique (c'est-à-dire qu'il ne s'adapte pas aux conditions du réseau) contrairement au transfert classique de type TCP (qui est élastique afin d'empêcher un effondrement du réseau). Une adaptation de la vidéo rend élastique également le *streaming* de la vidéo côté serveur.

7.4.1 Avantages de l'adaptation de la vidéo sur le codage statique

Il y a deux cas où l'adaptation vidéo est particulièrement utile : le premier cas et le plus important est dans les réseaux sans-fil, où le débit des données envoyées change en fonction de la qualité du lien radio. En voici deux :

1. les interférences dues à l'environnement ou à la présence d'un autre matériel fonctionnant sur la même bande de fréquences, ce qui diminue le rapport signal/bruit pour une courte période ;
2. la mobilité qui, selon la distance entre le mobile et le point d'accès, conduit à une atténuation du signal et peut à la limite de porté causer un changement de bande passante (54, 11, 5, 1 etc.)

Le débit disponible est très variable dans un réseau sans-fil ; nous simulons ce cas en utilisant un modèle de limitation du trafic (TC, *Traffic Shaping*) comme décrit dans la section 10.1.

Le second cas est quand un nombre variable de flux partagent la même liaison, par conséquent le débit disponible pour le *streaming* est également partagé et variable pour chaque flux. Nous illustrons ce cas en utilisant un nombre différent de flux simultanés comme décrit dans le chapitre 10 (expérimentations).

Pour ces deux cas, l'adaptation de la vidéo permet de profiter de tout le débit disponible et d'éviter de nombreuses pertes de paquets.

7.4.2 Étude de cas

Une transmission classique de la vidéo utilise le même bitrate du début à la fin de la transmission. A titre de comparaison, nous donnons quelques résultats théoriques basés sur un débit disponible, entre l'expéditeur et le récepteur, qui change selon la figure 7.1. Cela simule (à plus grande échelle) la dynamique du débit disponible entre deux hôtes sur un réseau. Comme nous le savons, de nombreux facteurs contribuent à cette dynamique, par exemple quand un utilisateur s'approche ou s'éloigne du point d'accès, ou lorsque plus ou moins de paquets circulent dans le réseau. Ce changement du débit disponible pose des problèmes pour la transmission vidéo car lorsque le bitrate est inférieur au débit disponible, le réseau est sous-utilisé, et quand il est supérieur au débit disponible, des paquets seront rejetés et donc la qualité sera négativement affectée.

La figure 7.1 présente les bitrates disponibles d'une vidéo hypothétique: 512kb/s (Q0), 1Mb/s (Q1), 2Mb/s (Q2) et 3Mb/s (Q3). En outre, comme le montre la figure, pendant les 180 secondes

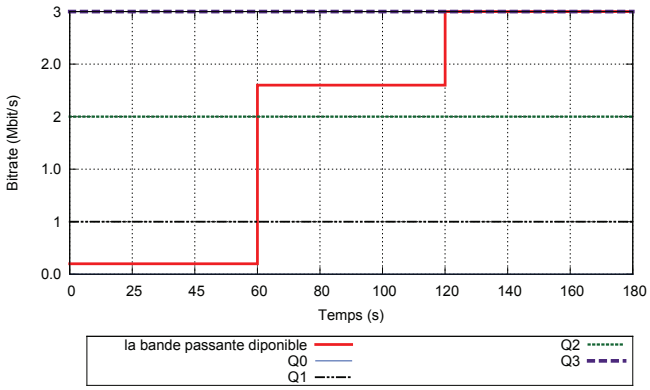


Figure 7.1 – Débit disponible vs bitrates disponibles.

de la transmission vidéo, le débit disponible change à trois reprises: 600kb/s pour la première minute où un bitrate de la vidéo de 512kb/s est le meilleur choix disponible ; 2300kb/s pour la deuxième minute où seulement la qualité de 3Mb/s ne doit pas être utilisée, et 3Mb/s pour la dernière minute, où toutes les qualités disponibles pourront être utilisées.

Dans cet exemple, nous pouvons remarquer qu'aucun des quatre bitrates disponibles n'est adapté à la transmission pendant toute sa durée:

1. Q0 est un bon choix pendant la première minute mais empêche l'utilisateur de profiter de la totalité de son débit disponible pendant le reste de la transmission.
2. Q1 n'est pas un bon choix pendant la première minute car supérieure au débit disponible, ni pendant le reste de la transmission car beaucoup inférieure à sa capacité.
3. Q2 est un bon choix pendant la deuxième minute, mais sur-utilise le débit disponible pendant la première minute et a la même faiblaisse que Q0 durant la troisième minute.
4. Finalement, Q3 n'est pas le bon choix ni la première ni la deuxième minute. Elle peut être utilisée seulement pendant la dernière minute.

Afin de comparer les qualités, nous calculons pour chacune d'entre elles le nombre de paquets envoyés, le nombre de paquets pouvant potentiellement être reçus et le pourcentage de paquets perdus qui en résultera. Les résultats sont donnés dans le tableau 7.1 (la taille de paquet s=1024 octets). Par exemple, les informations de Q1 sont obtenues comme ceci :

- 1Mb/s / (s=1024*8b) * (t=180s) = 23040 paquets envoyés ;
- 0.6Mb/s / (s=1024*8b) * (t=60) + 1Mb/s / (s=1024*8b) * (t=120) = 19968 paquets reçus (Q1 est supérieure au débit disponible pendant les premières 60 secondes, ce qui vaut dire que seulement 600kb/s sont reçus des 1Mb/s envoyés) ;
- 23040 – 19968 = 3072, c'est-à-dire que 13.3% des paquets sont perdus.

Le cas idéal apparaît lorsque le bitrate s'aligne au débit disponible, à savoir 512kb/s pour la première minute, 2Mb/s pour la deuxième minute et 3Mb/s pour la troisième minute. Il ne conduit à aucune perte et à un nombre égal à « maxb*time » des paquets envoyés et reçus, où maxb est à tout moment le maximum de bitrate inférieur ou égal au débit disponible. Dans notre exemple, cela donne 42240 de paquets envoyés et autant reçus.

Tout d'abord, ce tableau confirme ce que nous venons de dire : aucun bitrate statique n'est adapté à l'ensemble de la transmission. Deuxièmement, l'adaptation est clairement meilleure

Qualité	*#Pkts envoyés*	*#Pkts reçus*	*%Pkts perdus*
Q3	69120	45312	34.4%
Q2	46080	35328	23.3%
Q1	23040	19986	13.3%
Q0	11520	11520	0%
Idéal	42240	42240	0%

Tableau 7.1 – Nombre de paquets envoyés et reçus pour chaque bitrate statique comparé au cas idéal.

que chacun des bitrates statiques. La seule qualité sans perte de paquets est Q0, mais elle est pire que le cas idéal, car elle n'a que 11520 paquets reçus comparé à 42240 pour le cas idéal. À noter que le nombre de paquets reçus pour le Q3 est plus élevé que le cas idéal, car elle conduit à des paquets perdus, par exemple pour les 60 premières secondes, le cas idéal envoie à 512kb/s et reçoit au même débit, en revanche Q3 envoie à 3Mb/s et reçoit à 600kb/s.

Enfin, il est difficile, voire impossible, pour l'utilisateur ou un programme de décider au début d'une transmission vidéo quel bitrate statique est le meilleur à utiliser pour l'ensemble de la transmission. Au contraire, l'adaptation est automatique, c'est-à-dire il ne requiert aucune action de l'utilisateur. Dans la section 10.2.2 nous allons comparer les résultats de notre méthode d'adaptation à la transmission classique avec des bitrates statiques.

Chapitre 8

ÉTAT DE L'ART DES MÉTHODES D'ADAPTATION DE LA VIDÉO

L'augmentation du nombre d'applications temps réel, comme par exemple QuickTime et Real Player, et des sites de *streaming*, comme Youtube et DailyMotion, montre l'intérêt croissant des utilisateurs sur Internet d'avoir de nouveaux services de la vidéo. Basé sur IP, Internet reste cependant un réseau de type *best effort* et n'offre pas de qualité de service aux utilisateurs. La congestion, les pertes, la variation du débit disponible et l'augmentation du délai sont donc des caractéristiques imposées par la conception du réseau Internet et qui doivent être prises en compte comme contraintes afin d'améliorer le transfert des données vidéo et de donner à l'utilisateur final la meilleure qualité vidéo.

Pour cela, nous avons identifié trois catégories d'optimisation du transfert des données vidéo :

1. optimisation des formats vidéo ;
2. optimisation des protocoles de transport dédiés ;
3. optimisation du choix des données vidéo à transmettre.

La figure 8.1 montre sur quelle partie du transfert agit chacune de ces catégories. Il est à noter qu'une catégorie peut avoir plusieurs méthodes et qu'une méthode d'optimisation du transfert peut appartenir à plusieurs catégories en même temps, c'est-à-dire qu'elle propose des idées qui agissent sur plusieurs parties du transfert.

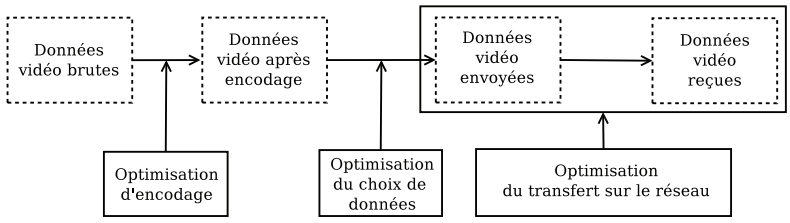


Figure 8.1 – Optimisation du transfert des données vidéo.

La première catégorie consiste plus précisément à optimiser l'encodage de la vidéo pour le transfert sur le réseau. La communauté du traitement de signal travaille sans cesse pour créer de nouveaux standard d'encodage de la vidéo qui augmente la performance de compression tout

en gardant la meileure qualité de la vidéo possible. Le standard d'encodage vidéo H.264 en est un exemple. Durant ces dernières années, cet encodage est devenu le standard de transmission vidéo en continu sur Internet, étant donné que la plupart des sites de *streaming* les plus populaires proposent leurs contenus encodés en H.264. L'utilisation de H.264 n'est pas gratuite, c'est pourquoi des alternatives existent comme par exemple le format de compression vidéo Theora qui est *open source*. Il existe également un nouveau format de compression dédié au *streaming*, c'est le VP8 récemment acheté par Google qui en a fait un format ouvert dans le cadre du projet WebM.

La deuxième catégorie cible l'amélioration de la partie réseau de la transmission. Des méthodes appartenant à cette catégorie améliorent la manière dont les paquets sont acheminés sans changer les données vidéo elles-mêmes. Pour cela elles agissent sur de nombreux paramètres. Il y a d'abord des protocoles de transport dédiés au *streaming*, tels que DCCP ou SCTP. Un des avantages de ce type de protocoles est leurs mécanismes d'évitement de congestion afin de minimiser les pertes et les délais d'acheminement. De plus, ils ont été conçus afin de prendre en compte les caractéristiques des données vidéo tels que le fait que les données vidéo soient temps-réel. Ensuite, il y a des méthodes qui modifient les protocoles de transport afin de transmettre les paquets sélectivement, en fonction du débit disponible [GAA05], ou de retransmettre les paquets de manière sélective en fonction de leur importance, par exemple dans une vidéo de type MPEG encodée en groupe d'images (I, P et B), les images I sont toujours retransmises, contrairement aux images P et B [HI06]. Enfin, il y a des méthodes qui améliorent les performances des protocoles de transport sur certains types de réseau afin que le débit d'envoi soit plus lisse et adapté au streaming. Nous avons présenté une contribution dans la partie précédente qui permet d'améliorer les performances des protocoles de transport sur les réseaux sans-fil.

Enfin, les méthodes dans la troisième catégorie agissent directement sur les données vidéo à transmettre en choisissant par exemple le bitrate à utiliser pour encoder la vidéo. Il y a essentiellement deux types de méthodes dans cette catégorie. Le premier type fait appel aux techniques de correction d'erreurs dans la vidéo en continu. FEC (Forward Error Correction), le code correcteur d'erreur, est une stratégie pour diminuer le taux d'erreurs pour la vidéo reçue. Cela fonctionne en ajoutant des bits supplémentaires aux données de la vidéo. Ces bits permettent de protéger l'application contre les erreurs de transmission au détriment des ressources réseau additionnelles utilisées pour ces données.

Le deuxième type consiste à transmettre la vidéo de manière adéquate sur le réseau (*network-friendly*). Nous allons utiliser le mot adaptation pour désigner ce type de méthodes. L'adaptation peut se faire de trois manières :

- En utilisant une méthode de commutation de flux. La vidéo est encodée en plusieurs qualités (bitrates) et la méthode d'adaptation change la qualité entre les différents flux afin que le bitrate corresponde au débit disponible du réseau.
- En utilisant une méthode de commutation de couches. La vidéo est encodée en plusieurs couches : une couche de base et des couches d'amélioration. La méthode d'adaptation envoie la couche de base et autant de couches d'amélioration que le débit disponible permette de passer.
- En utilisant une méthode d'encodage direct. La méthode d'adaptation ré-encode la vidéo transmise périodiquement en fonction du débit disponible.

De toute évidence, l'une des couches OSI (*Open System Interconnection*) devrait se charger de l'adaptation. Nous classons les travaux actuels en fonction de la couche qui se charge de cette tâche.

8.1 Méthodes basées sur la couche application

Cet approche modifie uniquement la couche application. L'application utilise les informations fournies par les couches inférieures. Ce sont les caractéristiques de l'encodage vidéo qui déterminent la façon d'adapter le bitrate de la vidéo au débit disponible. Par exemple, si la vidéo est encodée en plusieurs qualités, on fait une commutation de qualité; si elle est encodée en plusieurs couches, on envoie plus ou moins de couches.

8.1.1 RAAHS, *Rate Adaptation Algorithm for HTTP Streaming*

RAAHS propose un algorithme d'adaptation pour le *streaming* adaptatif en HTTP [LBG11]. Cet algorithme, appliqué chez le client, utilise des mesures spécifiques pour détecter le changement du débit disponible. Il se base sur le temps de chargement d'un segment de la vidéo SFT (*Segment Fetch Time*) pour détecter la congestion et calculer le débit de transmission en HTTP. Pour cela, cet algorithme compare le SFT avec la durée du segment MSD (*Media Segment Duration*) qui a une durée de 5 à 10 secondes. Puis la qualité de la vidéo est contrôlée chez l'utilisateur par une méthode qui l'augmente par étape et qui la diminue de manière importante en cas de congestion.

Si le temps de chargement d'un segment SFT est supérieur à sa durée, cela signifie que le débit TCP est inférieur au bitrate de la vidéo. Sinon, le débit est supérieur au bitrate. Ainsi, le ratio μ entre SFT et la durée du segment est utilisé pour détecter la congestion et le débit à transmettre, $\mu = \frac{MSD}{SFT}$. Donc, le débit r à transmettre est égal à $\mu * b$, où b est le bitrate du segment actuel.

Plus précisément si $\mu > 1 + \epsilon$ le bitrate suivant supérieur est choisi où $\epsilon = max\{\frac{b_{i+1}-b_i}{b_i}, \forall i = [0, 1, ..., N-1]\}$, b_i étant le bitrate du segment i et N le plus grand bitrate. Sinon, si $\mu < \gamma$, où γ est un seuil prédéfini dans RAAHS, il y a congestion et il faut baisser le bitrate : le bitrate à utiliser ensuite est b_i qui vérifie $b_i < \mu * b$.

Cet algorithme ressemble au nôtre du point de vue augmentation par étape/diminution agressive du bitrate. Il est en revanche différent du côté qui fait l'adaptation, c'est-à-dire serveur pour notre algorithme et client pour RAAHS. De plus, la durée d'adaptation de 10 secondes est très large par rapport à 2 secondes de notre méthode. La détection de la congestion est faite par le calcul de μ sur toute la durée de 10 secondes, ce qui n'est pas réaliste car le débit disponible peut être petit au début et peut redevenir suffisant ce qui est difficilement détectable par SFT. Finalement, le choix de TCP et non de DCCP peut rendre l'estimation de la bande passante via FST fausse à cause de la retransmission des paquets perdus. En effet, un paquet perdu bloque le tampon de réception TCP pendant un certain temps et augmente ainsi le temps de chargement du segment transmis.

8.1.2 AHDVS, *Akamai HD Video Streaming*

AHDVS [DCM10] emploie des connexions HTTP pour transmettre des vidéos entre le serveur et le client. Les vidéos sont encodées en cinq qualités avec le codec H.264 et un espacement de temps de 1,2s entre deux images clés I, ce qui veut dire que l'adaptation se fait toutes les 1,2s. Le client désirant regarder une vidéo télécharge une application en Flash pour exécuter l'algorithme d'adaptation. Le client fournit régulièrement au serveur des informations nécessaires

à la transmission et surtout à l'algorithme d'adaptation telles que la bande passante estimée par le récepteur, la taille de la mémoire tampon de réception, le nombre d'images reçues par seconde et le bitrate actuel reçu. Ainsi, AHDVS met à la disposition du client une liste de commandes afin de communiquer avec le serveur. Les commandes les plus importantes sont SWITCH_UP qui demande au serveur d'augmenter la qualité et BUFFER_FAILURE qui indique que la mémoire tampon est en difficulté et qu'il faut baisser la qualité.

Afin de décider quelle qualité il faut transmettre, l'algorithme utilise la taille du tampon de réception q et un seuil prédéfini q_t de la manière suivante :

$$r = l\frac{max((1 + \frac{q_t - q}{q_t})100, 10)}{100} \tag{8.1}$$

où r est le taux de données auquel l'application devrait alimenter le tampon TCP et l est le bitrate actuel. Dans cette équation, lorsque q est équivalent à q_t, r est équivalant à l et quand $q_t - q$ augmente la qualité augmente.

Cette méthode nécessite l'envoi de beaucoup d'informations entre le client et le serveur ce qui augmente le trafic sur le réseau, il faut également modifier des deux extrémités de la connexion.

8.1.3 QAC, *Quality Adaptation Controller*

Le contrôleur adaptatif de la qualité QAC [DCMP11] utilise une théorie à contrôle de *feedback* pour la transmission vidéo en continu. L'adaptation de la vidéo se fait du côté serveur afin de minimiser les changements nécessaires au cas où il y ait une amélioration sur l'algorithme du contrôleur. Le client se charge uniquement du décodage de la vidéo reçue.

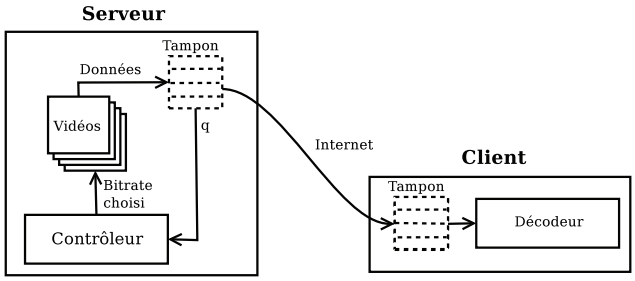


Figure 8.2 – Architecture de *streaming* de QAC.

Au début d'un transmission le contrôleur choisit une qualité, la stocke dans la mémoire tampon de l'émetteur et puis commence à la transmettre via une connexion TCP. Ensuite, QAC surveille en permanence la taille la mémoire tampon q de TCP afin qu'il y ait toujours des données à transmettre. Pour cela il choisit un bitrate en fonction du taux entre q et un seuil prédéfini q_t de sorte que ce bitrate soit le plus élevé et reste inférieur au débit disponible. La figure 8.2 montre l'architecture de *streaming* de QAC.

8.1.4 CBVA

CBVA, décrit dans [Fen02] et [SSF01], est un mécanisme qui supporte le *streaming* et l'adaptation des fichiers stockés de haute qualité. Il utilise des informations de la vidéo telles que celles

des trames de la vidéo afin de prendre une décision de *streaming*. En effet, CBVA combine le taux de trames par seconde (*Frame Rate*), la qualité de la trame et la mise en tampon des données reçues au niveau récepteur, dans une seule plateforme d'adaptation de la vidéo. De cette manière, pour CBVA, il existe des points de fonctionnement qui mettent en corrélation le taux des trames et la qualité de la trame.

Ainsi, un point de fonctionnement définit un groupe de combinaisons du taux des trames et la qualité du trame tout en gardant un certain équilibre entre eux (c'est-à-dire qu'il n'y a pas un paramètre qui est préféré à l'autre). Il représente autrement une vidéo encodée avec une certaine qualité et un certain taux de trames. Lorsque le débit disponible varie, CBVA bascule d'un point de fonctionnement prédéfini à un autre.

Pour chaque point de fonctionnement, une valeur de bande passante critique est calculée ($critBW$) permettant de déterminer un plan raisonnable de l'acheminement des paquets (c'est-à-dire si le débit disponible est suffisant pour délivrer des paquets avec la plus haute qualité disponible pendant une seconde donnée i, il est préférable d'envoyer à une qualité un peu inférieure afin de préserver le reste de la bande passante pour la seconde j où elle n'est pas suffisante). La figure 8.3 donne un exemple où aux secondes 3 et 4 le débit disponible nécessaire est inférieur à celle demandée pour les secondes 5 et 6. Ainsi, même s'il est possible d'envoyer à la plus haute qualité pendant les secondes 3 et 4, un peu de bande passante sera réservée pour les secondes 5 et 6. C'est pour cela qu'un débit disponible critique est défini pour chaque groupe (par exemple elle est de 3.29 à la seconde 6). De cette manière, à chaque changement de qualité, le point de fonctionnement choisi est celui ayant un débit disponible critique inférieur à celle disponible.

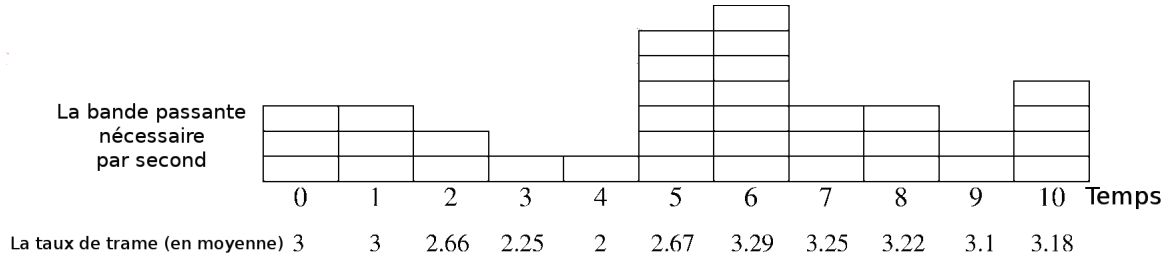


Figure 8.3 – Un exemple de la bande passante critique

Pour créer des points de fonctionnement, CBVA suppose que la vidéo est encodée en plusieurs groupes de taux de trames/qualités. Chaque groupe de trames (GOP) est traité en tant qu'une seule entité (généralement chaque demi-seconde). Il existe également une deadline limitée pour chaque GOP permettant de préciser si le temps de transmission est dépassé ou non. Ainsi, si la deadline est largement respecté, la qualité peut être augmentée (c'est-à-dire qu'il est possible d'utiliser un point de fonctionnement d'une qualité meilleure). Si, en revanche, la deadline est largement dépassée, CBVA doit basculer vers un point de fonctionnement d'une qualité inférieure.

Afin de garder la compatibilité avec les protocoles existants, autrement dit la propriété de *TCP-friendly*), CBVA emploie TCP comme protocole de transport. En planifiant correctement la livraison des données, CBVA garantit que même les paquets perdus seront retransmis. Ainsi, lorsqu'une trame est envoyée, elle arrivera à la destination.

Cette méthode garantit un acheminement adaptatif des données vidéo à la destination, mais

elle ne fonctionne que pour des vidéos stockées parce qu'elle a besoin de définir des paramètres de chaque vidéo en avance. De plus, elle suppose que le tampon de réception a une taille illimitée afin de pouvoir stocker toutes les données reçues, ce qui ne peut pas être garanti tout le temps en réalité.

Plusieurs méthodes se sont intéressées au format d'encodage vidéo SVC *Scalable Video Coding* qui introduit une extensibilité de la qualité spatiale et temporelle présentées par la suite.

8.1.5 Kofler *et al.*

Kofler *et al.* [KST+08] présentent une méthode qui traite le streaming vers un terminal sans-fil. La méthode présentée utilise une approche de type multi-couche basée sur la plate-forme de streaming MPEG-21. Cette méthode d'adaptation nécessite d'opérer essentiellement sur deux couches : la couche application et la couche transport ; d'autres couches, comme la couche physique, peuvent elles aussi fournir des informations. L'unité d'adaptation basée sur MPEG-21 se situe du côté serveur. Une description des trames de la vidéo et leurs correspondants niveau d'abstraction NAL (*Network Abstraction Layer*) se fait au niveau de l'outil appelé gBSD (*generic Bitstream Syntax Description*) de la plate-forme MPEG-21. L'adaptation se réalise donc en deux étapes : enlever la description des parties de la vidéo qui peuvent être supprimées, et adapter la vidéo à la nouvelle description. L'adaptation ainsi faite n'est effectuée que sur une partie de la vidéo, par exemple une image, une unité NAL ou un paquet. La vidéo adaptée est envoyée par la suite via le protocole RTP.

Les auteurs implémentent également l'équation du débit utilisée par TFRC tout en utilisant un mécanisme de *feed-back* basé sur RTSP afin de délivrer les informations du réseau (les paramètres d'entrée de l'équation de débit TFRC) du client au serveur. Ces informations sont envoyées périodiquement (une fois par RTT). D'autres informations concernent le bitrate maximum du décodeur et les résolutions d'écran de l'utilisateur.

8.1.6 Eberhard *et al.*

Eberhard *et al.* présentent aussi une plateforme de *streaming* adaptatif [ETHQ09] pour la vidéo à la demande VoD (*Video on Demand*) et le *multicast* basé sur le MPEG-21 et le format d'encodage SVC. Une extension, MXM *MPEG Extensible Middleware*, pour le client et le serveur est fournie. Cette extension utilise les méta-informations en XML (*eXtensible Markup Language*) de MPEG-21 pour offrir une adaptation des médias en continu.

Pour la VOD, l'unité d'adaptation ADTE (*Adaptation Decision Taking-Engine*) a été intégrée dans le serveur en utilisant le lecteur vidéo VLC (*Video LAN Client*). Elle se situe dans le modèle de mise en paquets qui reçoit les données avant d'être encapsulées en paquet RTP ce qui lui permet d'être indépendant de sa structure.

Après avoir reçu en HTTP des demandes de la part du client formées en utilisant son MXM, le MXM du serveur répond, en HTTP aussi, en envoyant des séquences disponibles. L'utilisateur choisit sa séquence et précise ses paramètres d'adaptation qui seront envoyer au serveur, toujours en HTTP, en utilisant la description MPEG-21. À ce moment le serveur utilise RTP/RTCP pour transmettre la séquence demandée.

Une description pour chaque média à transmettre est créée. Elle précise quelles sont les parties à conserver et celles qui peuvent être rejetées ou modifiées. Durant la transmission, Eberhard

et al. utilise cette description afin de procéder à l'adaptation en fonction de l'environnement du client.

Cette manière d'adaptation génère beaucoup de trafic entre le serveur et le client afin d'échanger les informations nécessaires à l'adaptation. De plus, elle nécessite la création et la sauvegarde de beaucoup d'informations pour chaque média.

8.1.7 Gorkemli *et al.*

Gorkemli *et al.* dans [GT10] créent une plateforme adaptative de *streaming*. L'application présentée transmet une vidéo encodée en SVC en « multi-couche ». La qualité de la vidéo transmise est déterminée par les conditions du réseau via DCCP. L'expéditeur estime le taux d'envoi sur le réseau, extrait un groupe d'images (GOP) de la vidéo de sorte que leur taux corresponde au débit disponible estimé et envoie les paquets vidéo extraits au récepteur. Comme les paquets arrivent au récepteur, ils sont insérés dans une mémoire tampon spéciale pour le décodeur, assez grand pour filtrer les variations dans le réseau. Le décodage commence dès que la moitié de la mémoire tampon est pleine. Afin d'éviter le débordement du tampon du décodeur, l'expéditeur estime également son occupation et ralentit sa transmission, si nécessaire.

Étant donné que les couches d'améliorations ont besoin de la couche de base pour être décodées, l'application utilise un schéma adaptatif de requête automatique de répétition ARQ (*Automatic Repeat reQuest*) pour diminuer le trafic de retransmission. Ce schéma demande uniquement la retransmission des paquets perdus sur le réseau appartenant à la couche de base lorsque la capacité du réseau est faible, et demande la retransmission de tous les paquets manquants lorsque le réseau le permet.

Malheureusement, la méthode d'adaptation n'est pas décrite.

8.1.8 Cazoulat *et al.*

Les auteurs de [CGH+07] créent une architecture de *streaming* basée sur la norme MPEG-21 et des vidéos encodées en SVC. Dans cette architecture les informations et les préférences statiques de chaque utilisateur sont avant tout récupérées et envoyés vers un nœud spécial, qui est responsable de l'adaptation de la vidéo, du côté serveur. Lorsqu'un des utilisateurs demande une vidéo, une représentation de type MPEG-21 est effectuée afin d'adapter la vidéo transmise à son profil. Après avoir préparé la représentation MPEG-21, l'application transmet la vidéo au récepteur via un autre nœud chargé de l'adaptation.

Si au cours de la transmission une partie de ce contexte change, comme par exemple le débit disponible ou la performance du lecteur vidéo, le nœud responsable de l'adaptation est informé de ce changement. Cela permet à l'adaptateur de réadapter les paramètres utilisés pour la transmission en cours comme par exemple l'allocation plus ou moins de bande passante pour chaque média transmise. Si en revanche les changements sont sévères, l'adaptateur peut reconfigurer la représentation de transmission en envoyant par exemple un autre format de la vidéo, une image fixe ou bien seulement la bande son de la vidéo demandée.

Outre la configuration présentée, l'architecture prend également en charge une transmission provenant d'une source en direct. Le rôle du nœud d'adaptation est dans ce cas simplifié et se compose essentiellement de relais avec adaptation. Un transfert en *multicast* en direct est effectué

et divisé par la suite en plusieurs transferts en *unicast*, voir la figure 8.4. Cependant, les médias de chaque transfert *unicast* sont adaptées en fonction de chaque bande passante disponible.

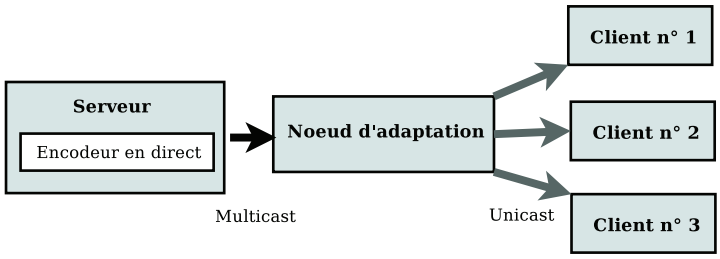


Figure 8.4 – Architecture de l'encodage en direct et la transmission vidéo en SVC de Cazoulat *et al.*.

8.1.9 DRDOBS, *Delay-Aware Rate Distortion Optimized Bitstream Switching*

DRDOBS est une méthode d'adaptation de la vidéo basée sur le délai de la transmission [XZ04]. Elle utilise un approche de tampon virtuel afin d'estimer ce délai, éviter la perte des paquets et leur re-mise en cache.

Le tampon virtuel se situe dans le réseau entre le serveur et le client. Pour une application de *streaming* la capacité du tampon B devrait vérifier l'équation suivante (à un instant i) :

$$0 \leq B(i) \leq \sum_{j=i+1}^{i+N_t} C(j) \tag{8.2}$$

où N_t est l'intervalle de temps pour le décodage des N images suivantes et $C(j)$ représente la capacité du réseau. Le taux d'envoi et de réception sont supposés être égaux.

DRDOBS propose de changer la qualité entre deux bitrates. Le bitrate le plus élevé sera utilisé tant que l'équation précédente n'est pas violée ce qui permet d'éviter une sous-utilisation du tampon de réception. Dans le cas contraire, l'algorithme bascule vers le bitrate inférieur.

Cette méthode a été validée par la simulation ce qui a permis de créer un environnement où les suppositions spécifiques sont devenues possibles. Ces suppositions sont irréalistes et difficilement applicables en réalité.

8.1.10 MVCBF, *Multiple Virtual Client Buffer Feedback*

MVCBF [LBG10] est défini comme un nouveau type de *feed-back* RTCP au niveau application. Il est proposé afin d'améliorer les messages d'information entre le client et le serveur dans une transmission vidéo en continu de type AVC. En effet, la solution proposée définit un tampon virtuel pour chaque couche de la vidéo transmise, chez le client, afin de stocker ses données. Puis le client envoie des *feed-back* MVCBF pour chaque tampon virtuel VCB *Virtual Client Buffer*. Après la réception d'un *feed-back* le temps de la vidéo MT pour chaque couche est calculé, puis il est comparé aux 5 seuils prédéfinis ($T1$ à $T5$) par l'algorithme afin de décider de l'adaptation appropriée à effectuer.

– Si $MT > T5$ ajouter une couche.
– Sinon si $T2 < MT < T3$ maintenir la qualité à la couche actuelle.
– Sinon si $T1 < MT < T2$ augmenter le nombre d'images par seconde.
– Sinon si $MT < T1$ rejeter une couche de la vidéo transmise.

Le seuil $T4$ est utilisé pour contrôler le temps de la couche de base. Si $MT < T4$ pour cette couche, il faut directement rejeter la dernière couche ajoutée. De l'autre côté, si pendant un certain temps il n'y a pas de *feed-back* MVCBF, une nouvelle couche est ajoutée.

8.1.11 Positionnement

Les méthodes qui ont été décrites dans cette catégorie se divisent en deux types : des méthodes effectuant l'adaptation du côté émetteur et celles qui la font du côté récepteur. Bien que le deuxième type puisse prendre en compte la taille du tampon de réception, cela nécessite de surcharger le réseau par du trafic supplémentaire entre le client et le serveur pour faire l'adaptation. De plus, cela demande de faire un changement sur les applications clientes ce qui n'est pas pratique.

Pour le deuxième type, nous remarquons surtout que soit les méthodes sont proposées pour un format spécifique de la vidéo tel que AVC, soit elles sont optimisées pour un type particulier de protocoles de transport tel que TCP, soit elles fonctionnent sous certaines conditions. Cela rend ces solutions moins générales et applicables seulement dans certains cas.

Notre méthode se situe aussi dans cette catégorie, celle où seulement la couche application est modifiée. Nous pensons que c'est la meilleure couche pour réaliser l'adaptation. Nous pensons également que, si une méthode d'adaptation du débit doit être déployée, elle devrait entraîner un minimum de changements aux systèmes d'exploitation existants et aux applications et surtout qu'elle devrait être générale et indépendante d'autres conditions tels que le mécanisme de contrôle de congestion utilisé et le format de la vidéo.

8.2 Méthodes basées sur plusieurs couches

Une autre approche est d'effectuer des changements sur deux couches à la fois : les couches inférieures donnent des *feed-back* sur l'état du réseau à la couche application, qui agit en conséquence, par exemple en adaptant le bitrate de la vidéo. Les *feed-back* peuvent être de type *pull*, c'est-à-dire que l'application demande des informations, ou de type *push*, c'est-à-dire que le réseau contacte l'application pour l'informer.

8.2.1 Occupation de la mémoire tampon du récepteur

L'adaptation de la vidéo appelée *Buffer-driven* [LC08] définit un mécanisme de *streaming* qui prend en compte la mémoire tampon du récepteur. L'objectif principal de cet article est d'éviter la sous-utilisation et la sur-utilisation de cette mémoire. *Buffer-driven* repose sur les protocoles RTP/RTCP afin de déterminer les conditions réseau (bande passante disponible) ainsi que l'occupation de la mémoire tampon du récepteur, voir la figure 8.5 qui décrit son architecture. Ces informations sont utilisées par la suite pour adapter le débit de la vidéo transmise. *Buffer-driven* procède ainsi à deux types d'adaptation : le taux d'envoi et le bitrate de la vidéo transmise.

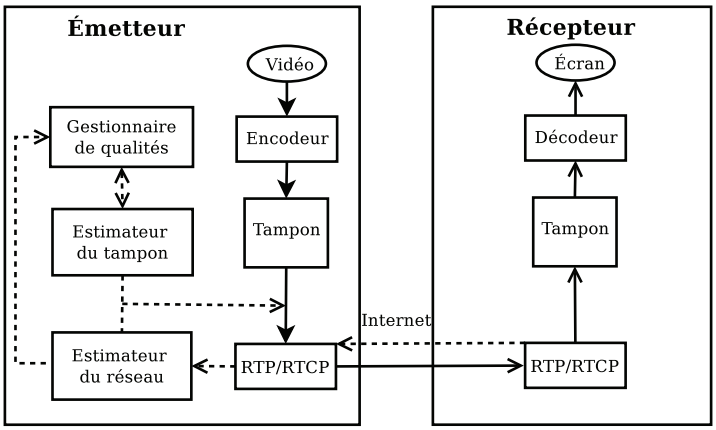


Figure 8.5 – Architecture de transmission de *Buffer-driven*.

Buffer-driven définit donc deux seuils permettant l'adaptation en fonction des informations calculées : *th_min* et *th_max* (voir équations 8.3 et 8.4). Si l'occupation de la mémoire de réception est entre 0 et *th_min* la qualité de la vidéo est augmentée afin d'éviter une sous-utilisation de la mémoire. De l'autre côté, si elle est entre *th_max* et 100% la qualité est diminuée afin d'éviter une sur-utilisation de la mémoire. En revanche si cette occupation est entre les deux seuils (*th_min* et *th_max*), *Buffer-driven* maintient la qualité et soit il diminue le taux d'envoi si le réseau est congestionné, c'est-à-dire qu'il y a des paquets perdus, soit il augmente le taux d'envoi dans le cas contraire. Les deux seuils sont calculés comme suit :

$$th\_min = \frac{R(bps)/8bits}{B(bytes)} * 100\% \tag{8.3}$$

$$th\_max = 100\% - th\_min \tag{8.4}$$

où R est le taux d'encodage et B est la taille totale de la mémoire tampon.

Le calcul de l'occupation de la mémoire tampon du récepteur B_E se fait sur l'émetteur afin de minimiser le trafic généré dans le cas où c'est le récepteur qui le fait et qui envoie l'information à la source (ce calcul a été présenté dans [LC06] pour les mêmes auteurs) :

$$B_E = B_C + (\frac{pktnum * s}{i} - R) * RTT \tag{8.5}$$

où *pktnum* est le nombre de paquets, s est la taille du paquet, i est l'intervalle de RTCP, R est le taux d'encodage et B_C est la dernière occupation calculée.

Ce mécanisme présente des calculs assez compliqués et risque de ne pas être efficace car c'est l'émetteur qui réalise le mesure des informations qui sont à la disposition du récepteur.

Un autre mécanisme d'adaptation basé sur l'occupation de la mémoire tampon du récepteur est décrit dans [YWZW02]. Ce mécanisme, contrairement à *Buffer-driven*, ne prend pas en compte la stabilité et l'équité dans le réseau car il utilise UDP comme protocole de transport pour faire le *streaming*.

8.2.2 Dieu *et al.*

La méthode de *streaming* décrite dans [NO07] crée un nouveau mécanisme de contrôle de congestion pour mieux adapter le *streaming* des vidéos encodées en H.264/AVC. Le client envoie au serveur les accusés de réception uniquement pour les paquets de la couche de base, l'état du tampon de réception et la bande passante estimée. Avec ces informations, le contrôle de congestion décidera jusqu'à quelle couche sera envoyé au client. Pour extraire ces couches de la vidéo, un extracteur de *bitstream* est utilisé. Les paquets appartenant à la couche de base ont une très haute priorité concernant la retransmission tandis que les paquets des autres couches ont une priorité basse.

Afin d'estimer le débit disponible, uniquement au début de la transmission, cette méthode modifie la technique PTR (*Packet Transmission Rate*) [HS03] qui est une variation de la technique appelée « la paire de paquets ». Cette modification consiste à envoyer 30 paquets de sondage d'une taille de 500 octets du serveur au client.

Le contrôle de congestion présenté utilise cette estimation initiale de la bande passante B, l'occupation du tampon de réception ainsi que les paquets perdus afin d'adapter le bitrate au débit disponible et d'éviter une sous-utilisation de la mémoire tampon du récepteur. Donc, si B est supérieur au taux de données actuel des résolutions spatio-temporelles, il y a deux choix : le premier, augmenter la résolution temporelle (la fréquence d'images) si B est inférieure à la résolution spatiale suivante supérieure, sinon le deuxième, augmenter la résolution spatiale. Dans le cas contraire, on diminue le taux de données des résolutions spatio-temporelles.

Cette méthode repose essentiellement sur une seule estimation de la bande passante ce qui n'est pas suffisant. De plus, elle se concentre d'une façon intensive sur les caractéristiques de la vidéo ce qui nécessite un traitement différent pour chaque vidéo.

8.2.3 iTCP, *interactive TCP*

Le protocole de transport interactive iTCP [KZ07] est une solution de type *pull*, c'est-à-dire qu'il fournit des informations et c'est à la charge de qui en a besoin de les demander. Il modifie l'implémentation du protocole TCP sur l'expéditeur. Il ajoute à TCP un mécanisme actif de *feed-back* pour fournir des informations sur les événements du réseau. La retransmission due *timeout* et la réception d'un troisième accusé de réception sont des exemples de tels événements. Une application intéressée s'inscrit à iTCP pour récupérer ces *feed-back* afin de s'adapter aux conditions du réseau. En outre, lorsque de tels événements apparaissent, cette application peut accéder à des variables pertinentes de la connexion comme la taille de la fenêtre de congestion, le seuil de démarrage lent (ssthresh) et le RTT. Les auteurs de iTCP l'ont implémenté dans le système d'exploitation FreeBSD. Ils ont réalisé plusieurs expérimentations, notamment le transcodage de la vidéo en se basant sur le temporisateur de retransmission.

Un grand avantage de iTCP est sa transparence pour le réseau, il n'y a donc aucun problème pour son déploiement. Mais à l'opposé, iTCP a l'inconvénient que, du côté du récepteur, si un ou plusieurs paquets sont perdus, l'application réceptrice est bloquée jusqu'à l'arrivée des retransmissions. Si la retransmission est par malchance perdue aussi, le retard devient très élevée. DCCP par exemple ne souffre pas de ce problème, car ce n'est pas un protocole fiable, donc plus approprié pour le *streaming* vidéo. Contrairement à iTCP, le mécanisme que nous utilisons ne consomme pas beaucoup de ressources CPU, et ne demande pas de création de *threads* spéciaux,

car il est passif, autrement dit axé sur les applications, il est juste une question de lecture d'une variable et en prenant les mesures appropriées.

8.2.4 VTP, *Video Transport Protocol*

Video Transport Protocol (VTP) [BMGS03] est un nouveau protocole au niveau application. Il a été conçu pour transmettre spécifiquement les vidéos encodées en MPEG-4. Le principal but de VTP est de minimiser les pertes des images clefs (image I) d'une vidéo transmise en envoyant moins de paquets lorsque le réseau est congestionné et par conséquent avoir moins d'images I supprimées à la réception. Pour cela VTP joue sur deux facteurs : le bitrate de la vidéo et le taux d'envoi. La vidéo devrait donc être encodée préalablement en plusieurs bitrates. En ce qui concerne le taux d'envoi, il est calculé à la source, grâce à une interaction spéciale avec la destination qui envoie régulièrement, à chaque intervalle prédéfini, des acquittements. Ainsi, VTP est basé sur un taux de transmission et non sur une fenêtre de congestion comme TCP.

VTP exerce un contrôle sur la transmission tous les k paquets envoyés. L'émetteur envoie une demande d'acquittement et le récepteur lui répond. Cette échange permet à l'émetteur de calculer le RTT, le taux d'envoi, et par la suite le meilleur bitrate à envoyer. VTP présente ainsi les inconvénients inhérents à l'implémentation du contrôle de congestion à ce niveau. Il nécessite aussi une modification de l'application du côté du récepteur, et il y a un risque qu'il ne soit pas compatible avec TCP (*TCP-friendly*).

La méthode décrite dans [Kaz02] agit de la même façon que VTP, mais elle crée un nouveau protocole de transport.

8.2.5 Schierl et Wiegand

Schierl et Wiegand [SW04] proposent également une méthode d'adaptation de la vidéo basée sur H.264/AVC. Leur méthode combine une technique d'extensibilité temporelle de la vidéo et une technique de commutation de flux en fonction du débit disponible. La première technique consiste à rejeter les couches la deuxième couche d'amélioration et puis la première couche en réponse à une réduction du débit disponible ce qui en résulte d'une réduction entre 30% et 40% du taux de données à envoyer. Dans la deuxième technique, la vidéo est encodée avec de multiples valeurs pour le paramètre de quantification QP (*Quantization Parameter*) ce qui donne une vidéo avec plusieurs bitrates disponible au serveur. En fonction du débit disponible, qui est calculé en utilisant un algorithme de type IIAD (*Inverse Increase Aditive Decrease*), la méthode d'adaptation commute entre les bitrates disponibles.

IIAD [BB01] fonctionne de la manière suivante :
- Quand un paquet perdu est rapporté à l'émetteur, la fenêtre de congestion W sera réduite selon l'équation : $W = W_t - \beta$, où W_t est la fenêtre de congestion actuelle et $0 < \beta < 1$ est une constante.
- S'il n'y a pas de pertes, W augmente comme suit : $W = W_t + \alpha/W_t$, où $\alpha > 0$ est une constante aussi.

L'algorithme d'adaptation, voir la figure 8.6 a été implémenté avec les protocoles RTP/RTCP modifiés pour évaluer les feed-back du récepteur et procéder à la retransmission des paquets perdus. Cet algorithme définit trois seuils ($s1 = 500ms$, $s2 = 1000ms$ et $s3 = 1500ms$) pour le retard Δ des paquets à la destination pour déterminer quelle technique d'adaptation utiliser.

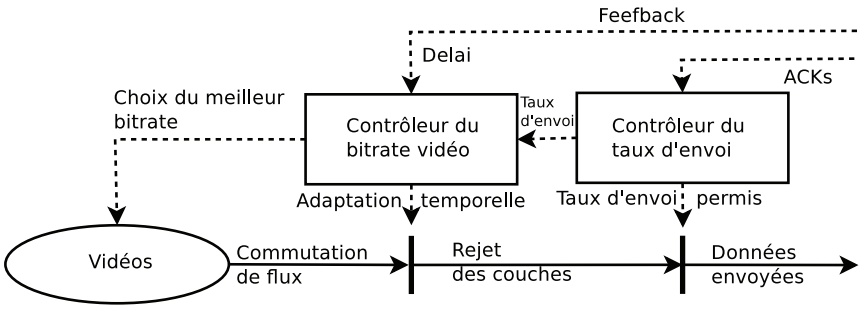


Figure 8.6 – Algorithme d'adaptation de Schierl et Wiegand.

1. Si $\Delta < s1$ utiliser la technique de commutation de flux.
2. Si $s1 < \Delta < s2$ utiliser la première technique et rejeter la première couche d'amélioration.
3. Si $s2 < \Delta < s3$ utiliser la première technique et rejeter la première et la deuxième couche d'amélioration.
4. Si $\Delta > s3$ utiliser la première technique et rejeter les trois couches d'amélioration.

8.2.6 Méthodes utilisant le ré-encodage

MPEG-TFRCP (*MPEG-TCP-Friendly Rate Control Protocol*) [WMM01] est une méthode qui utilise le ré-encodage en direct pour adapter le bitrate de la vidéo transmise au débit disponible du réseau. Cela fonctionne uniquement avec des vidéo encodées en MPEG-2 et en changeant le paramètre de quantification du codeur. Les auteurs proposent d'effectuer l'adaptation chaque période de temps équivalente à 32 fois le temps d'un RTT. Pour connaître le débit disponible du réseau nécessaire à l'adaptation, les auteurs modifie le protocole de transport en utilisent une équation basée sur le RTT et la probabilité de pertes p des paquets :

$$r_i = \frac{MTU}{RTT\sqrt{\frac{2p}{3}} + T_0 \min(1, 3\sqrt{\frac{3p}{8}})p(1+32p^2)} \ , \text{ (Si p > 0)} \tag{8.6}$$

$$r_i = 2 * r_{i-1} \ , \text{ (Sinon)} \tag{8.7}$$

où r_i est le débit calculé, T_0 le *timeout* de retransmission, et MTU la taille maximum d'une unité de transmission (*Maximum Transmission Unit*).

Dans nos travaux précédents [LMB+06], le ré-encodage a été utilisé aussi avec le protocole DCCP. La solution apportée ne se concentre pas sur les problèmes de réseau, mais sur l'architecture complète du *streaming*, y compris un mixeur RTP entre le serveur et le client. À noter que le ré-encodage (adaptation de contenu) a été fait dans le mixeur, qui a besoin d'avoir accès aux données DCCP à travers un mécanisme inter-couches.

Toute autre qui utilise le ré-encodage direct n'est pas pratique en réalité car elle nécessite d'énormes ressources CPU et un délai supplémentaire avant qu'une nouvelle séquence ne soit prête. La plupart des méthodes d'adaptation actuelles tendent vers l'utilisation des vidéos pré-encodées en plusieurs qualités.

8.2.7 Méthodes basées sur la couche réseau

Cet approche consiste à laisser la couche application inchangée tout en changeant une des couches basses ([CMC+10]). L'application envoie la vidéo encodées en couches en utilisant le codec H.264/SVC. La couche *Internet Protocol* (IP) les reçoit toutes et envoie seulement celles qui sont avantageuses pour le réseau (*network-friendly*). La bande passante est estimée par des mesures régulières.

8.2.8 Positionnement

Les méthodes proposées dans cette catégorie partagent toutes la même idée : pour transmettre la vidéo en continu d'une façon adaptative, plusieurs couches doivent être changées. Elles proposent, soit de recréer le contrôle de congestion du protocole de transport afin qu'il soit plus adapté au *streaming* tout en donnant leur propre solution pour le choix de la qualité convenable, soit de changer le mécanisme de retransmission pour qu'il soit plus adapté aux caractéristiques de la vidéo. Ces méthodes nécessitent cependant des changements importants sur le système d'exploitation, ce qui n'est pas souhaitable étant donné que ce genre de solutions a beaucoup de mal à s'imposer. De l'autre côté, des protocoles de transport adaptés à la transmission de la vidéo existent, ce qui diminue le besoin de faire des changements sur plusieurs couches.

8.3 Outils de *streaming* adaptatif existants

8.3.1 *IIS Smooth Streaming*

IIS Smooth Streaming [Zam09] est un service de transmission vidéo adaptative en direct fournis par Microsoft. C'est une solution basée sur le Web qui nécessite l'installation d'un plug-in disponible pour Windows et iPhone OS version 3.0. IIS Smooth Streaming est un service agnostique qui emploie un algorithme de commutation de flux où différentes versions de la vidéo peuvent être encodées avec des débits configurables et des résolutions allant jusqu'à 1080p. Dans la configuration par défaut, la vidéo est encodée en sept couches allant de 300 kbps à 2,4 Mbps.

8.3.2 *Adobe Dynamic Streaming*

Adobe Dynamic Streaming [Dyn10] est un service de *streaming* adaptatif basé sur le Web et développé par Adobe qui est disponible pour tous les appareils utilisant un navigateur avec le plug-in d'Adobe Flash. Le serveur stocke plusieurs flux avec différentes qualités et résolutions. Le service commute entre les différente qualités pendant la lecture, afin de correspondre au débit disponible des utilisateurs et leur CPUs. Le service utilise le protocole de transmission en continu RTMP [Rea09]. Les codecs vidéo supportés sont H.264 et VP6, qui sont inclus dans le plug-in d'Adobe Flash.

8.3.3 HTTP *Adaptive Live Streaming*

Apple de son côté a aussi publié récemment un client HTTP de *streaming* adaptatif *Adaptive Live Streaming* [PWM11]. Le serveur coupe le contenu vidéo en plusieurs morceaux encodés en

multiples qualités et avec des durées configurables. Le serveur expose une liste de lecture (avec l'extension .m3u8) contenant tous les segments de vidéo disponibles. Le client télécharge les segments consécutifs de la vidéo et il choisit dynamiquement la qualité vidéo en utilisant un algorithme non divulgué. L'algorithme emploie le codec H.264 en utilisant le conteneur MPEG2 TS. Ce client est disponible sur n'importe quel appareil fonctionnant sous le système d'exploitation iPhone OS version 3.0 ou ultérieur (y compris l'iPad), ou sur n'importe quel ordinateur ayant QuickTime X ou une autre version ultérieure installée.

8.3.4 Positionnement

Ces outils ont été conçus par les grandes sociétés sur Internet. Bien qu'ils aient eu un bon appui en terme de développement et de financement, ils restent fermés et restreints en terme de possibilités et des codecs supportés. De plus, ces outils ont été développés dans un but commercial ce qui fait que leur utilisation, même s'ils sont parfois gratuits, est soumise à des conditions d'utilisation très restrictives.

8.4 Récapitulatif des propriétés des méthodes d'adaptation

Le tableau 8.1 récapitule et compare les propriétés principales des méthodes d'adaptation de la vidéo. La comparaison est faite en termes des critères suivants.

1. Les niveaux où les modifications demandées par la méthode apparaissent. Cela est indiqué par « A » si cela implique un changement sur la couche l'application, « S » sur le serveur (Émetteur), « C » pour le client et « CL *Cross Layer* » si les modifications sont sur plusieurs couches OSI.
2. La généralité concernant le codec. Nous considérons qu'une méthode est générale si elle ne dépend pas d'un codec précis pour son fonctionnement.
3. La complexité de la méthode et son déploiement.
4. L'utilisabilité de la méthode pour la transmission en direct, c'est-à-dire que les données ne sont pas disponibles à l'avance (vidéoconférence).
5. La nécessité de générer un trafic réseau supplémentaire (en plus de la vidéo elle-même) pour son fonctionnement.
6. Enfin, le respect de la propriété temps réel (sans pause) d'une transmission vidéo en continu.

8.5 Conclusion

Il y a de plus en plus d'intérêt pour le domaine du *streaming* multimédia. Étant donné la nature de type *best effort* du réseau IP et la variation du débit, beaucoup de travaux se sont concentrés sur le *streaming* adaptatif de la vidéo. Pourtant, soit le problème est resté non résolu, soit les solutions proposées sont difficilement applicables en réalité.

Dans cette thèse, nous proposons une solution générale qui peut être appliquée sur n'importe quel serveur et pour la plupart des formats vidéo disponibles. De plus, cette solution nécessite un minimum de changements et surtout n'en demande que sur l'application du côté émetteur.

Caractéristiques / Méthode	Modifications	Codec utilisable	Déploiement facile	Support Trans. directe	Trafic supplémentaire	Temps réel
RAAHS	A, C	Général	Oui	Oui	Oui	Oui
AHDVS	A, C	Flash/H.264	Non	Oui	Oui	Oui
QAC	A, S	Général	Oui	Oui	Non	Oui
CBVA	A, S	MPEG	Non	Non	Non	Non
A. et al.	A, S, C	AVC/MPEG-21	Non	Non	Oui	Oui
E. et al.	A, S	AVC/MPEG-21	Non	Non	Oui	Oui
G. et al.	A, S	AVC	-	-	Oui	Non
C. et al.	A, C, S	AVC/MPEG-21	Non	Non	Oui	Oui
DRDOBS	CL, C, S	Général	Non	Non	Oui	Oui
MVCBF	CL, C, S	AVC	Non	Non	Oui	Oui
Recever-driven	CL, C	Général	Oui	Oui	Oui	Oui
Dieu	CL, C, S	AVC	Non	Oui	Oui	Oui
iTCP	CL, S	Général	Oui	Oui	Non	Oui
VTP	CL, C, S	MPEG-4	Oui	Oui	Oui	Oui
S. et W.	CL, S	H.264/AVC	Non	Oui	Non	Oui
MPEG-TFRCP	CL, C, S	MPEG-2	Non	Oui	Oui	Oui
VAAL/ZAAL	A, S	Général	Oui	Oui	Non	Oui

Tableau 8.1 – Comparaison des propriétés des méthodes d'adaptation.

La suite de cette partie de thèse est consacrée à proposer une solution d'adaptation vidéo au niveau de la couche application et d'évaluer ses performances.

Chapitre 9

ADAPTATION DE LA VIDÉO AU NIVEAU DE LA COUCHE APPLICATION

9.1 Différentes possibilités d'adaptation de la vidéo au niveau de la couche application

Pour être en mesure de savoir s'il faut augmenter, maintenir ou diminuer la qualité de la vidéo, l'application émettrice doit avoir des informations sur le débit disponible. Ainsi, une approche inter-couches, entre l'application et le protocole de transport, ou tout simplement un retour d'informations (*feed-back*) à la demande, à partir du protocole de transport, est nécessaire. Nous avons identifié les méthodes suivantes pour atteindre cet objectif :

1. Le protocole de transport estime le débit disponible (ou un autre paramètre spécifique au réseau) : généralement, le contrôle de congestion calcule la valeur du débit de transmission effectif sur le réseau. Cette valeur peut être transmise à la couche application, à sa demande, qui peut, à son tour, l'utiliser pour contrôler la qualité de la vidéo.
2. Le pourcentage (taux) de remplissage du tampon de la socket du protocole de transport (autrement dit, le pourcentage des paquets de données dans la mémoire tampon) : l'application expéditrice peut respectivement augmenter/diminuer la qualité lorsque le pourcentage de remplissage est faible/élevé. Cette information est actuellement fournie pour TCP par certains systèmes d'exploitation [1]. L'implémentation de cette méthode pour d'autres protocoles de transport nécessite une modification du noyau, qui n'est ni simple, ni facile à utiliser.
3. Un tampon au niveau de l'application : si la méthode précédente n'est pas disponible, l'expéditeur peut la simuler en créant son propre tampon au niveau de l'application. L'expéditeur met ses paquets dans cette zone tampon, tandis qu'un autre processus (*thread*) essaie en permanence de déplacer les données de la mémoire tampon application au tampon de la socket du protocole de transport. Ensuite, l'expéditeur se comporte comme dans

1. Sur GNU/Linux, une structure de type `tcp_info` avec une telle information est renvoyée par `getsockopt` avec un `TCP_info` comme paramètre.

la méthode précédente. En mesurant le taux de remplissage de ce tampon, l'application peut obtenir le débit disponible.

4. Notre méthode VAAL, expliquée plus en détail dans la suite. Le débit disponible est mesuré par le nombre de paquets qui n'ont pas pu être écrits dans la socket du protocole de transport (*buffer overflow*). Plus ce nombre est grand, plus le débit disponible est petit. Régulièrement, VAAL calcule le pourcentage du nombre de ces paquets en échec et change le bitrate de la vidéo en conséquence. VAAL augmente donc la qualité si ce pourcentage est nul, et la diminue tant que le pourcentage augmente.

Dans toutes ces méthodes, l'expéditeur doit essayer de réduire au minimum la fluctuation de la qualité. Il n'est pas préférable, à notre opinion, de changer la qualité à chaque paquet envoyé, mais sur une plus grande échelle, par exemple chaque 100ms, chaque 2sec ou chaque image I pour une vidéo MPEG. Toutefois, certains codecs peuvent avoir des contraintes sur le moment du changement de qualité. L'application doit être donc configurée de sorte que le changement de qualité ne tombe pas en contradiction avec le codec utilisé. Ce point mérite plus de discussion et des tests pour être étendu à tous les codecs actuels.

À noter que toutes ces méthodes nécessitent un protocole de transport avec un mécanisme de contrôle de congestion sur le réseau, peu importe lequel, afin d'obtenir des retours (*feed-back*) du réseau. De plus, elles utilisent toutes des algorithmes simples (l'algorithme 3 montre celui de VAAL). Enfin, elles opèrent au niveau de la couche application et ne nécessitent aucune modification au noyau du système ou du réseau.

9.2 Transmission de la vidéo en continu

9.2.1 Avantages de DCCP pour la transmission vidéo en continu

Lors de toute transmission de données, un protocole de transport comme TCP, UDP ou DCCP doit être utilisé. DCCP a plusieurs avantages pour le *streaming* de la vidéo :

- Comme UDP, DCCP ne retransmet pas les données perdues. En conséquence, un paquet perdu ne bloque pas l'application cliente jusqu'à ce que le paquet soit retransmis et reçu. D'autre part, comme TCP, il utilise les accusés de réception, c'est-à-dire que l'expéditeur DCCP est informé des paquets qui ont été perdus afin qu'il puisse prendre une décision de retransmettre ou non certains d'entre eux.
- Comme UDP, DCCP est orienté paquet, donc l'application peut utiliser la technique de ALF (*Application-Level Framing*), qui est une technique utile pour le *streaming* vidéo.
- Comme TCP, DCCP fournit des informations sur les conditions du réseau (telles que le taux d'envoi calculé par l'émetteur et le taux de réception calculé par le récepteur) à l'application. L'application peut adapter son débit, par conséquent, en fonction de ces conditions.
- Comme TCP, DCCP utilise un contrôle de congestion de type *TCP-friendly*. En outre, plusieurs contrôles de la congestion sont disponibles. TFRC est le plus approprié pour le *streaming* vidéo [FHPW08].

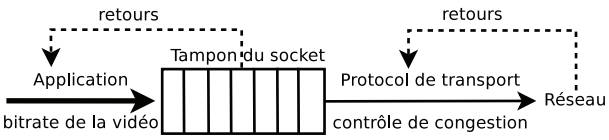


Figure 9.1 – Le flux des données de la vidéo côté émetteur.

9.3 Méthode d'adaptation VAAL

Nous avons implémenté la méthode d'adaptation de la vidéo VAAL (***V**ideo **A**daptation at **A**pplication **L**ayer*) au niveau de la couche application sur une machine GNU/Linux ayant un noyau 2.6.35, sans aucune modification sur le noyau du système. Les paquets en échec sont directement supprimés (l'explication a été donnée plus haut dans la section 9.2), c'est-à-dire qu'ils ne sont pas stockés ni retransmis plus tard. VAAL utilise DCCP en tant que protocole de transport et TFRC en tant que contrôle de congestion (pour des raisons fournies précédemment).

Actuellement, il existe deux versions implémentées de VAAL. Nous notons la première par VAALv1 (le pseudo-code 3 présente la boucle principale de l'algorithme de VAALv1). Pour VAALv2, qui est une amélioration de VAALv1, nous avons étudié quatre nouvelles idées (modifications). Le pseudo-code 4 présente la boucle principale de l'algorithme de VAALv2. VAALv2 ainsi que VAALv1 sont présentées et expliquées par la suite.

9.3.1 Explication générale pour les deux versions

Comme le montre la figure 9.1, l'application écrit des paquets dans la mémoire tampon de la socket du protocole de transport à un taux égal au débit de la vidéo en cours. D'un autre côté, le protocole de transport a un contrôle de congestion qui fournit la vitesse à laquelle les paquets peuvent quitter la machine et entrer dans le réseau. Le but de VAAL est d'adapter le bitrate de la vidéo au débit estimé par le protocole de transport. Ainsi, l'algorithme de VAAL est divisé en deux étapes (présentées dans la figure 9.2) :

- **la découverte de l'état du réseau :** le débit disponible comme information basée sur le débordement du tampon du protocole du transport ;
- **la sélection de la qualité :** l'action à réaliser.

Ces deux étapes sont exécutées à chaque période de n secondes.

Découverte de l'état du réseau

Lorsque l'application génère des paquets à un taux supérieur de ce que le protocole de transport peut envoyer vers les couches inférieures, notamment la couche réseau, les paquets sont mis en attente dans le tampon de la socket. Si cette mémoire tampon est pleine, les nouveaux paquets générés par l'application ne peuvent pas y être écrits (débordement du tampon), et l'application reçoit un code d'erreur. VAAL surveille ainsi le débit disponible au travers de débordement de tampon de la socket du protocole de transport lorsque l'application essaie d'y écrire un paquet. À chaque période, VAAL calcule le pourcentage de paquets en échec, que nous appelons WFP (*Write Failure Percentage*, pourcentage d'échecs d'écriture). Une période d'adaptation est l'intervalle de temps pendant lequel le bitrate est ne change pas. WFP est donc

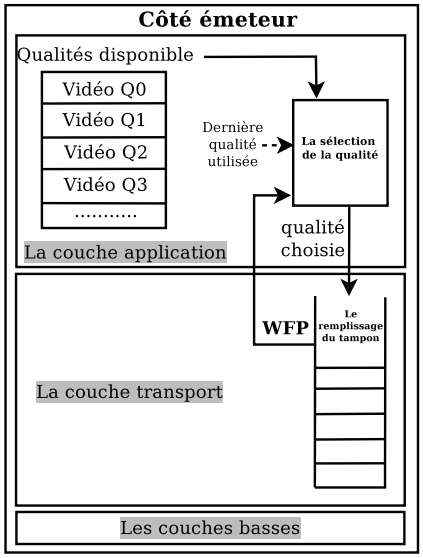


Figure 9.2 – Le fonctionnement de la méthode VAAL du côté émetteur.

une indication de l'état du réseau : plus WFP est grand, plus le débit disponible est inférieur au débit des données envoyées par l'application.

Sélection de qualité

À la fin de chaque période, VAAL lit la valeur de WFP, donnée par la première étape sur la période qui vient de s'écouler, et :

- Si WFP est nul (car il n'y a pas de paquets en échec lors de l'écriture dans le tampon), VAAL choisit le niveau supérieur de qualité suivante, un débit plus élevé, sauf si la qualité actuelle est déjà la plus élevée.
- Sinon, si WFP est tolérable (moins de 5 %), la qualité est maintenue au même niveau. ITU-T G.1070 [IT07] recommande que le taux de perte de bout en bout des paquets IP en *streaming* vidéo devrait être inférieur à 10 %. Par conséquent, nous avons choisi un seuil de 5 % de taux de perte au niveau de la mémoire tampon de l'émetteur (WFP <= 5 %), les 5 % autres étant disponible afin de supporter les pertes sur le réseau.
- Enfin, lorsque WFP est supérieur à 5 %, et à moins que la qualité la plus basse ne soit déjà en cours d'utilisation, VAAL recherche une qualité inférieure q' qui vérifie l'équation suivante:

$$q' \leq q(1 - \mathrm{WFP})r \tag{9.1}$$

où q est la qualité actuelle. Dans cette formule, $q(1-\mathrm{WFP})$ représente le débit des données injectées dans le réseau, donc le débit disponible pour la période qui vient de s'achever, tandis que $r > 0$ est un facteur qui représente l'agressivité du t5 %ransfert. r, s'il est différent de 1, permet de choisir une qualité avec un débit différent du débit disponible.

À noter que VAAL nécessite un protocole de transport avec contrôle de congestion, peu importe lequel. En outre, VAAL est particulièrement utile dans la vidéoconférence

(vidéo à la volée), car il n'exige pas de ré-encoder la vidéo, mais simplement de changer le paramètre qui influence le bitrate, c'est-à-dire choisir un autre taux d'encodage.

9.3.2 VAALv1

Avec cette version de VAAL, WFP est calculée toutes les deux secondes, $n = 2$. Ces 2 secondes ne constituent pas un temps absolu car il peut varier d'une transmission à l'autre en fonction des vidéos disponibles. En effet, 2 secondes ont été choisies de sorte qu'elles correspondent à la présence d'une image I dans nos vidéos utilisées qui ont été encodées avec une image I toutes les 2 secondes. Autrement dit ces 2 secondes représentent le temps d'une période pour VAAL et elles correspondent normalement au temps de transmission pendant lequel un changement de qualité s'avère être difficile voire impossible.

Puisque pour un taux de perte inférieur ou égal à 5 % VAAL maintient le débit, **nous avons choisi une valeur d'agressivité $r = 1,05$, c'est-à-dire qu'on envoie 5 % de plus que le débit disponible, donc un tolérance de 5 % de paquets en échec est proposée**. Finalement, VAALv1 débute la connexion en envoyant la plus haute qualité disponible à partir de la source vidéo.

Pseudo-code de l'algorithme de VAALv1

Voici tout d'abord le pseudo-code pour une transmission vidéo classique avec un codage statique :

Algorithme 2 – Le pseudo-code d'une transmission classique sans adaptation

```
   Données : La vidéo à envoyer
1  répéter
2  |  envoyer un paquet
3  |  sleep
4  jusqu'à la fin de la vidéo
```

Il est évident que la seule chose à faire est d'écrire les données de la vidéo dans la mémoire tampon de la socket du protocole de transport et d'attendre pour reprendre la main.

En revanche, lorsque la méthode d'adaptation de la vidéo VAALv1 est utilisée, le pseudo-code de la transmission reste simple mais devient :

Algorithme 3 – Le pseudo-code de l'algorithme de VAALv1

```
   Données : La vidéo à envoyer
   Variables :
   ▷ err : le nombre de paquets en échec
   ▷ pkts : le nombre de paquets envoyés
   ▷ wfp : le pourcentage de paquets en échec
 1 pour chaque période d'adaptation faire
 2 |   err ← 0
 3 |   pkts ← 0
 4 |   répéter
 5 |   |   pkts ⟵ pkts + 1
 6 |   |   envoyer un paquet
 7 |   |   si paquet en échec alors
 8 |   |   |   err ⟵ err + 1
 9 |   |   fin
10 |   |   sleep
11 |   jusqu'à la fin de la période
12 |   wfp ⟵ err/pkts
13 |   si wfp = 0 alors
14 |   |   augmenter le bitrate
15 |   sinon si wfp < 5 % alors
16 |   |   maintenir le bitrate
17 |   sinon
18 |   |   baisser le bitrate
19 |   fin
20 fin
```

9.3.3 VAALv2

Nous avons étudié quatre idées afin d'améliorer les résultats de VAALv1. Les trois premières idées améliorent les performances tandis que la dernière donne des résultats similaires. Ces quatre idées sont : 1) un changement sur le bitrate initial ; 2) un changement sur l'intervalle de calcul de WFP ; 3) un changement sur la valeur de r et 4) une comparaison entre le bitrate théorique et le bitrate réel.

Bitrate initial

VAALv1 commence la connexion en envoyant la plus haute qualité disponible à partir de la source vidéo. Dans nos expérimentions concernant VAALv1 nous avons noté qu'au début de chaque transmission vidéo le nombre de paquets perdus est élevé. Nous pouvons expliquer cela non seulement parce que le débit disponible initial ne suffit pas nécessairement pour le bitrate le plus grand, mais le plus important, parce qu'au début de la communication le protocole de transport, comme DCCP, estime le débit disponible lentement. Ainsi, commencer la connexion avec un bitrate élevé obligera la mémoire tampon du protocole de transport à rejeter un grand nombre de paquets générés, ce qui a un impact négatif sur la qualité de la vidéo au début de

chaque transmission.

En conclusion, **dans VAALv2 nous décidons de ne pas commencer le transfert avec un bitrate élevé, mais avec un bitrate faible**. Notre choix a également été de ne pas utiliser le plus petit bitrate, car il faudra quelques secondes pour atteindre le plus haut bitrate au cas où le débit disponible est assez élevé pour le soutenir dès le départ. Dans notre implémentation actuelle VAALv2, **le bitrate initial utilisé est de 1 Mb/s**, qui représente le plus grand bitrate inférieur ou égal au bitrate moyen.

Cette idée de bitrate initial est en réalité bien connue et des solutions de type *Slow Start* existent, d'où nous avons dérivée notre solution.

Intervalle de calcul de WFP

WFP est le paramètre utilisé par VAAL pour estimer le débit disponible. Son calcul dans VAALv1 était fait sur toute la durée de la période de deux secondes. Toutefois, le débit disponible peut varier rapidement dans un intervalle de temps largement inférieur à 2 secondes. C'est pourquoi nous avons considéré qu'il valait mieux calculer WFP sur un plus petit intervalle de temps, à la fin de la période de 2 secondes. Nous avons examiné deux idées pour cet intervalle de temps : utiliser le dernier RTT, ou bien utiliser l'intervalle de temps qui correspond à l'envoi des p derniers paquets envoyés ; p ne devrait pas être inférieur à 20, afin de permettre l'apparition d'un pourcentage de paquets en échec (WFP) entre 95 % et 100 %, (cf. la sélection de qualité, section 9.3.1). Cependant, parce que le RTT (selon la topologie du réseau) et le nombre de paquets pendant cette période pourraient être très faibles, nous choisissons une solution hybride utilisant à la fois les deux idées. Donc, dans cette solution hybride, l'intervalle de temps sur lequel WFP est calculé est égal à max(dernier RTT, le temps d'envoi des 20 derniers paquets).

Le calcul hybride de WFP fonctionne comme suit :

- Stocker le temps d'envoi et le numéro de séquence de chaque paquet envoyé.
- Calculer le dernier RTT.
- Calculer $numSPkts$ (le nombre de paquets envoyés pendant le dernier RTT).
- Si $numSPkts > 20$, calculer le pourcentage (WFP) pendant ce RTT
  Sinon calculer WFP = le pourcentage des paquets rejetés sur les 20 derniers.

La section 9.3.4 analyse l'influence de ce paramètre.

Valeur de l'agressivité r

r représente l'agressivité du transfert. Comme nous l'avons déjà montré (équation 9.1, page 98), en cas de baisse de qualité, la qualité suivante choisie est la plus haute qualité disponible q' qui vérifie $q' \leq w*r$, où w est le débit disponible. Si $r = 1$, la qualité suivante sera au plus égale à w, tandis que si $r = 1,1$ par exemple, la qualité suivante pourrait être aussi grande que $1,1w$. Toutefois, ce dernier cas ne signifie pas nécessairement que la qualité sera réellement plus grande que w. Si par exemple les qualités disponibles de la vidéo transmise augmentent en tant que puissances de 2 (512kb/s, 1Mb/s, 2Mb/s etc.), la plus haute qualité disponible inférieure ou égale à $1,1w$ se situe entre $1,1w/2$ et $1,1w$, avec une moyenne de $1,1w * 3/4 = 0.825w$, ce qui est plus petit que w. D'autre part, si toutes les qualités sont disponibles, par exemple par un encodage à la volée très fin, la qualité suivante sera donc proche ou égale à $1,1w$, ce qui est supérieur à w, donc pas une bonne solution.

En conclusion, r devrait être choisi en fonction des qualités disponibles de la vidéo. **Plus le ratio des bitrates entre les qualités consécutives est grand, c'est-à-dire que la qualité supérieure divisée par la qualité inférieure, plus la valeur de r devrait être grande**.

Dans une transmission vidéo avec un encodage direct, où le bitrate peut être précisé finement, la valeur idéale est $r = 1$ si le codec est assez précis parce que le bitrate idéal peut être utilisé. Bitrate idéal signifie qu'il est exactement égal au débit disponible.

En VAALv1, $r = 1,05$ avait été choisi pour tolérer jusqu'à 5 % de paquets perdus. Dans la version actuelle, plusieurs valeurs pour r ont été testées. Leur impact sur le débit et le taux de pertes est analysé dans la section 9.3.4, et $r = 1,1$ est utilisée dans VAALv2 suite à cet analyse.

Bitrate réel vs le bitrate théorique

Nous savons que, parfois, les encodeurs ne peuvent pas encoder une vidéo à un débit précisé, autrement dit le bitrate de la vidéo générée n'est pas égal, mais est à proximité du débit demandé. Nous notons par q_t le bitrate théorique (ou demandé) pour une vidéo et par q_r le bitrate réel (ou produit).

Comme spécifié précédemment, lorsque VAAL est à l'étape de sélection de qualité et après calcul du WFP, la qualité suivante q' (pour les 2 secondes qui suivent) est choisie de telle sorte que $q' \leq q(1 - \text{WFP})r$ (formule 9.1), où q est la qualité utilisée dans les 2 dernières secondes. q et q' peuvent avoir deux significations différentes, autrement dit deux manières de les prendre en compte dans la formule :

- Pour q' : si le bitrate réel auquel la vidéo est encodée pendant les deux prochaines secondes est connu, par exemple pour le *streaming* vidéo en utilisant des fichiers pré-enregistrés, q' peut prendre cette valeur qui est q'_r. Sinon, $q' = q'_t \leq q(1 - \text{WFP})$.
- Pour q : parfois le bitrate réel auquel la vidéo est encodée à tout moment n'est pas connu, par exemple pour la vidéoconférence interactive lorsque le codec n'est pas très précis, mais il est calculable à la fin de chaque période. Donc, q peut prendre, dans la formule, une de deux valeurs possibles : q_t la valeur théorique qui a été utilisée il y a deux secondes, ou q_r la valeur du bitrate calculé à la fin de la période de 2 secondes. Toutefois, le débit théorique q_t utilisé 2 secondes auparavant n'existe que si à l'étape précédente c'est q'_t qui a été choisi.

Puisque q'_r n'est pas toujours disponible, nous préférons choisir q'_t. Il existe maintenant un choix entre q_t ou q_r. Section 9.3.4 analyse l'influence de ce choix pour VAALv2 dans certains cas. VAALv1 utilisait q_t et q'_t.

9.3.4 Études expérimentales pour l'amélioration de VAAL

Choix de l'intervalle de calcul de WFP

Dans cette section VAALv2 est dérivée de VAALv1 avec la modification suivante : WFP est calculé en utilisant la formule hybride (le dernier RTT ou les 20 derniers paquets). Deux séries d'expérimentations ont été faites pour démontrer l'impact de cette modification sur la performance de VAAL : une série avec cinq flux concurrents et une autre avec dix flux concurrents, chacune d'elle est répétée cinq fois. Le réseau utilisé pour ces tests est présenté dans la

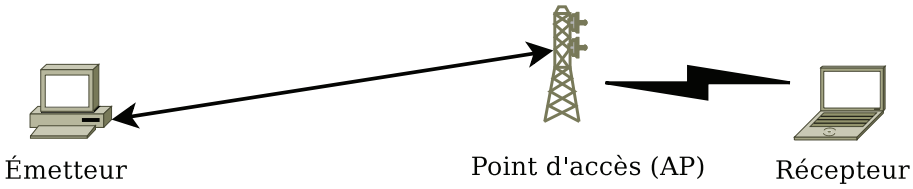


Figure 9.3 – Topologie du réseau utilisé pour comparer VAALv1 et VAALv2.

figure 9.3, qui fait appel à la même topologie du réseau qui a été employé pour l'évaluation des performances de VAALv1. Ce réseau est composé d'un émetteur relié par une connexion câblée à un point d'accès (AP), et d'un récepteur qui est relié également au même AP par une liaison sans-fil.

Les résultats sont présentés dans quatre figures : 9.4(a) et 9.4(b)) pour cinq flux, 9.5(a) et 9.5(b) pour dix flux simultanés. Les deux figures de chaque série montrent en une répétition représentative, et la moyenne de toutes les répétitions.

Dans tous ces graphiques il y a quatre courbes, dont deux sont marquées avec un cercle pour indiquer le nombre de paquets envoyés et reçus pour la version ancienne VAALv1 et deux courbes marquées avec un carré pour leur équivalents pour VAALv2. L'axe des abscisses représente le numéro du flux pour la répétition représentative et le numéro de la répétition pour la moyenne de toutes les répétitions.

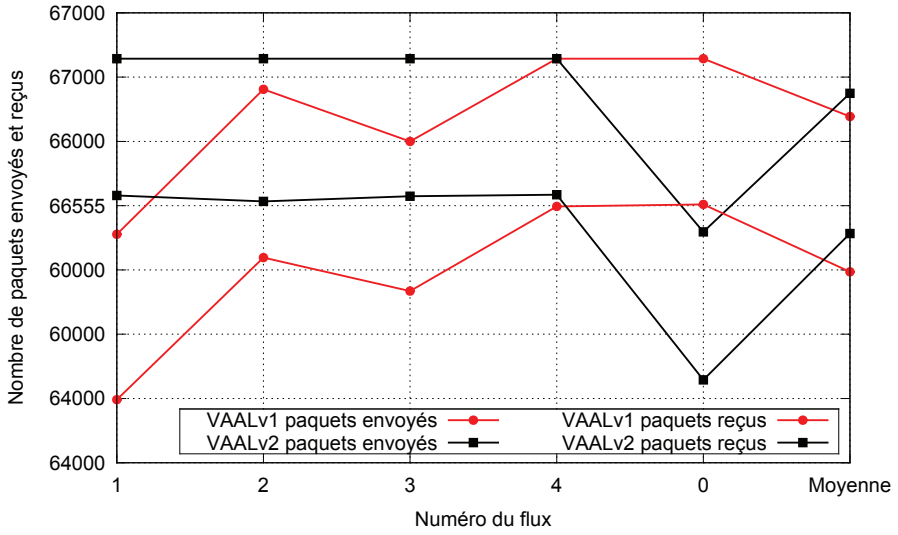
Comme le montre la figure 9.4(a), pour un test représentatif, tous les flux de VAAL ont approximativement le même débit, c'est-à-dire qu'ils ont presque le même nombre de paquets envoyés et reçus et leur moyenne est exactement la même. Ceci est valable pour les deux versions (VAALv1 et VAALv2). Cette équivalence est due au fait que le débit disponible est suffisamment élevé pour laisser passer cinq flux simultanés à leur plus haute qualité (3Mb/s). Le taux de paquets perdus pour les deux versions est très faible.

La moyenne des cinq répétitions, figure 9.4(b), montre que VAALv2 est légèrement meilleure dans la plupart des répétitions, par exemple, les répétitions numéros 2, 4 et 5, mais elle est pire pour la répétition numéro 3. La moyenne de toutes les répétitions, nous pouvons conclure que **si le débit disponible est assez élevé pour la plus haute qualité, VAALv1 et VAALv2 donnent des résultats similaires**. En fait, le taux de perte est très faible, par conséquent WFP est souvent nul, peu importe s'il est calculé sur le dernier RTT ou sur les deux dernières secondes, ce qui conduit à des résultats similaires.

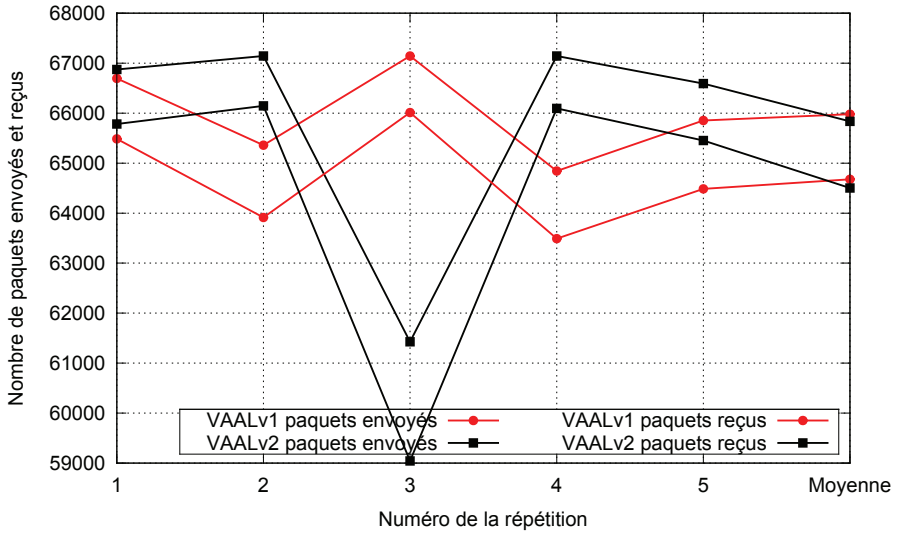
D'autre part, dans le cas du test avec dix flux concurrents, le débit disponible n'est pas assez élevé pour laisser passer tous les flux. La performance de VAALv1 et VAALv2 est maintenant très différente. Dans la figure 9.5(a) l'ensemble des flux de VAALv2 envoient et reçoivent plus de paquets que les flux équivalents de VAALv1. En outre, presque tous les paquets supplémentaires envoyés sont reçues, ce qui indique que la version modifiée de l'intervalle de calcul du WFP (RTT ou 20 derniers paquets) estime mieux le débit disponible au moment de la décision.

La même conclusion est tirée en regardant la moyenne des cinq répétitions dans la figure 9.5(b) : **VAALv2 est toujours meilleure ; elle permet d'obtenir environ 10 % de plus sur le nombre de paquets envoyés et reçus**.

Dans les tests suivants c'est VAALv2 qui est utilisée avec un bitrate initial de 1Mb/s et la formule hybride du calcul de WFP.

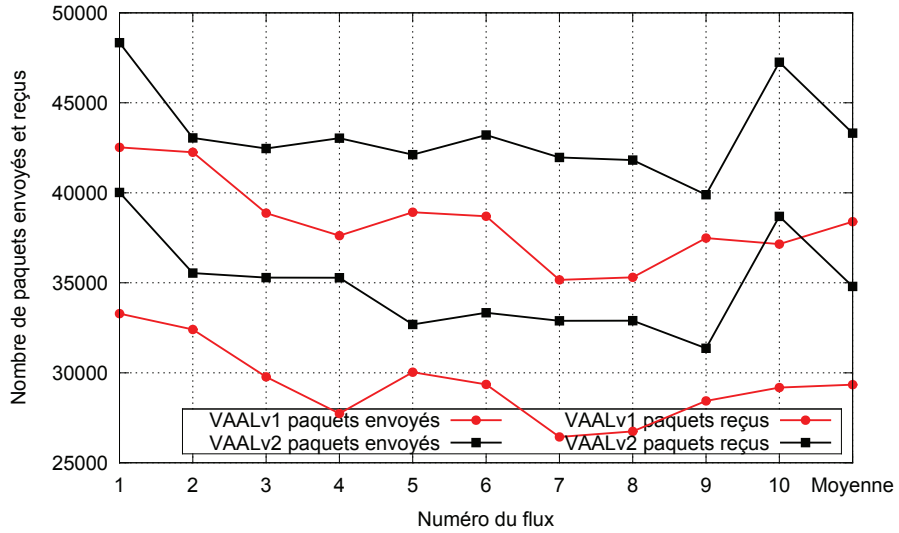


(a) Une répétition représentative : il y a une équivalence de débit entre tous les flux pour les deux versions

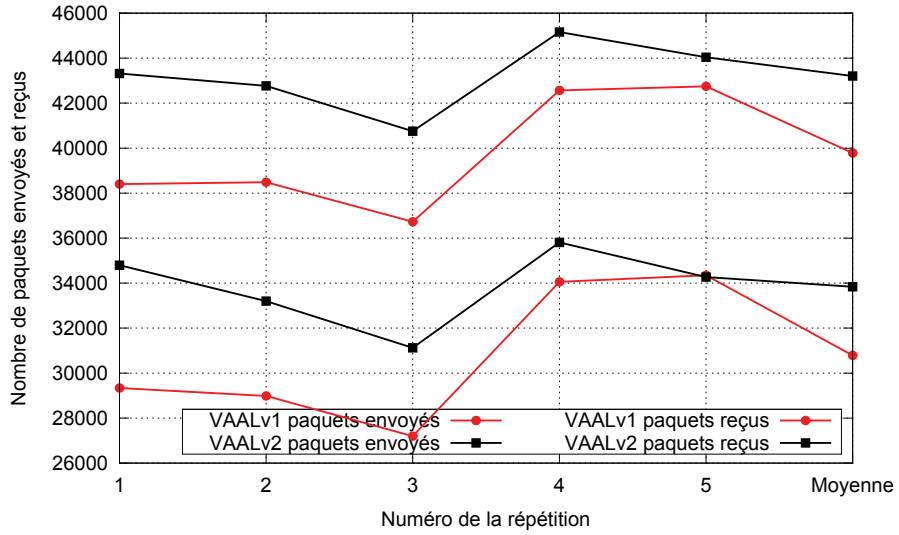


(b) La moyenne des cinq répétitions : VAALv1 et VAALv2 sont similaires si le débit disponible est assez élevé

Figure 9.4 – Comparaison entre VAALv1 et VAALv2 pour cinq flux.



(a) Une répétition représentative : tous les flux de VAALv2 ont un débit plus élevé que les flux de VAALv1.



(b) La moyenne des cinq répétitions : VAALv2 est meilleure que VAALv2 dans toutes les répétitions.

Figure 9.5 – Comparaison entre VAALv1 et VAALv2 pour dix flux.

Choix de r et de la qualité

Une série d'expérimentions a été effectuée pour comparer les différentes valeurs de r et le choix de la qualité. Le réseau utilisé est le même que celui de la section des expérimentions (voir la section 10.1).

Les résultats sont présentés dans les figures 9.6 et 9.7. Les deux graphiques possèdent huit courbes : le nombre de paquets envoyés et reçus pour r égal à 1 et 1,1, et l'utilisation de q_r et q_t. Nous en déduisons les conclusions suivantes :

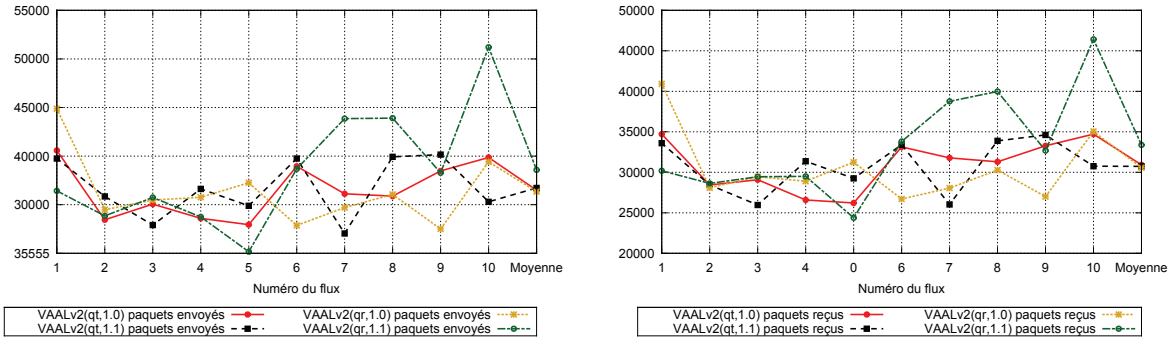
- Plus la valeur de r est élevée, plus le nombre de paquets envoyés et reçus est grand. Cette remarque est tirée de la figure 9.7 où pour q_t, la courbe notée avec des carrés pleins ($r = 1,1$) a une moyenne plus grande que la courbe notée avec des cercles pleins ($r = 1,0$). De même, pour q_r, $r = 1,1$ (cercles vides) a une moyenne de paquet envoyés et reçus plus grande que $r = 1,0$ (étoiles). Cependant, l'influence de r reste faible, c'est-à-dire que la différence est petit entre les courbes des deux valeurs de r .
- L'utilisation de q_t rend le transfert plus agressif qu'avec l'utilisation de q_r, ce qui donne un débit plus élevé. En effet, dans le même graphique, les courbes avec cercles et carrés pleins pour q_t ont plus de paquets envoyés et reçus que les courbes avec étoiles et cercles vides pour q_r.
- Les valeurs testées de r : 1 ; 1,1 et 0,9 qui n'est pas représentée ici par souci de simplicité et le choix de q (q_r ou q_t) n'ont pas d'impact important sur le taux de perte, mais sur le débit. Presque tous les paquets supplémentaires envoyés sont reçus et c'est pour tous les flux, voir la figure 9.6 où toutes les courbes ont presque la même différence entre le nombre de paquets envoyés et reçus.
- La moyenne des 5 répétitions (voir la figure 9.7) indique que (pour trois répétitions (numéro 1, 2 et 4) sur cinq) VAALv2(q_t, 1,1) a un plus grand nombre de paquets reçus et un plus grand nombre de paquets envoyés aussi que VAALv2(q_r, 1,1). De l'autre côté, VAALv2(q_r, 1,1) a un taux de perte moins important. Comme il sera montré plus tard, en figure 10.2 page 112, q_t est généralement supérieur à q_r, autrement dit q_t est plus agressif, ce qui conduit à un débit plus élevé et à un taux plus élevé de perte des paquets. **Le choix d'une qualité moindre, q_r, est donc préférable pour un taux réduit de pertes, tandis que le choix d'une meilleure qualité q_t, est préférable si plus de débit est demandé** (le client accepte plus de pertes).

En VAALv2, en se basant sur des résultats ci-dessus, nous avons fixé $r = 1,1$. Dans les expérimentions qui suivent, q_t *est utilisé pour décider de la qualité suivante.*

Par souci de simplicité, nous utiliserons VAAL dans le reste de cette thèse pour parler de la *deuxième* version VAALv2.

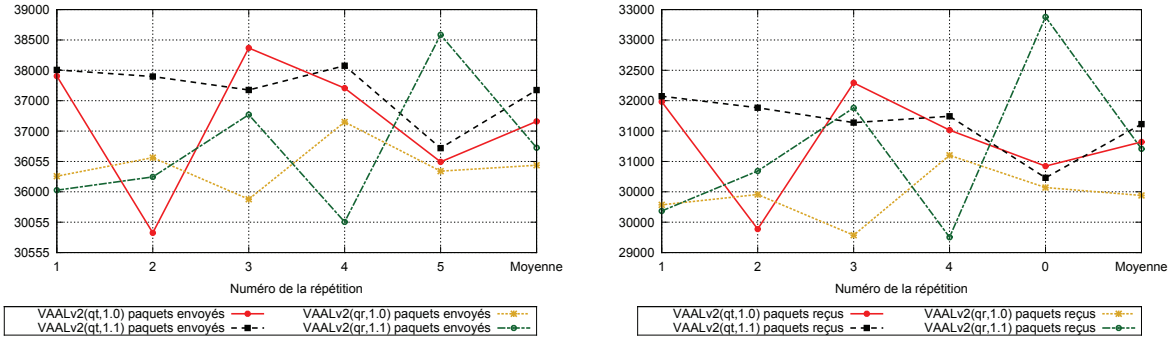
9.3.5 Pseudo-code de l'algorithme de VAALv2

Le pseudo-code pour une transmission vidéo en utilisant VAALv2 comme méthode de choix d'adaptation de la vidéo est présenté dans l'algo 4.



(a) Nombre de paquets envoyés (b) Nombre de paquets reçus

Figure 9.6 – Comparaison entre VAALv1 et VAALv2 pour dix flux sans écart (une répétition représentative) : r influence sur le débit et pas sur le taux de pertes..



(a) Nombre de paquets envoyés (b) Nombre de paquets reçus

Figure 9.7 – Comparaison entre VAALv1 et VAALv2 pour dix flux sans écart (la moyenne des cinq répétitions) : q_r réduit le taux de pertes, tandis que q_t améliore le débit.

9.4 Conclusions

Nous avons présenté une méthode simple mais efficace (VAAL, *Video Adaptation at Application Layer*) pour adapter le contenu de la vidéo pendant la diffusion. Cette méthode utilise le débordement de la mémoire tampon de la socket du protocole de transport sur le serveur afin d'estimer le débit disponible pour la vidéo. Intuitivement, cette méthode devrait mener à une meilleure qualité de la vidéo en *streaming*.

En revanche, nous avons remarqué que notre méthode VAAL a deux types possibles d'inconvénients quand au choix du bitrate. Le premier est la surestimation du débit disponible, ce qui conduit à un bitrate de la vidéo transmise supérieur au débit disponible. Ce sur-bitrate peut se produire dans les cas suivants :

1. Dans le milieu des deux secondes, période de temps pendant la quelle VAAL ne change pas de qualité car le débit disponible pourrait brusquement baisser ; autant de paquets seront perdus jusqu'à la fin de cette période de deux secondes.
2. Lorsque VAAL passe à un bitrate plus élevé, mais que le débit disponible est inférieur à sa valeur car VAAL ne peut pas prédire la valeur exacte du débit disponible car cela dépendra du futur.

Le second cas est l'utilisation d'une vidéo avec bitrate plus petit que le débit disponible du réseau. C'est une sous-estimation du débit disponible. De la même manière que ci-dessus, ceci peut apparaître :

1. Lorsque, au milieu des deux secondes, le débit disponible va brusquement augmenter.
2. Lorsque le bitrate passe à la valeur suivante immédiatement supérieure, mais le débit disponible est encore plus élevé que cela.

Une solution à ces inconvénients est de s'adapter aux conditions du réseau plus souvent. Or, cette solution est soumise à des contraintes du codec utilisé. Nous avons donc étudié une autre solution qui minimise le nombre de paquets perdus. Un historique de mauvaises décisions est utilisé, permettant de connaître les bitrates surestimant le débit disponible, afin de ne pas les utiliser pendant un certain temps. Comme il s'agit d'une solution qui fonctionne avec toute méthode d'adaptation de la vidéo, nous lui avons dédié un chapitre à part, le chapitre 11.

Dans le chapitre suivant, la performance de VAAL est évaluée par des expérimentions. Elle est également comparée à une méthode classique de transmission de la vidéo avec des bitrates statiques afin de se comparer.

Algorithme 4 – Le pseudo-code de l'algorithme de VAALv2

Données : La vidéo à envoyer
Variables :
▷ *err : le nombre de paquets en échec pendant l'intervalle de calcul de WFP*
▷ *pkts : le nombre de paquets envoyés par l'application pendant l'intervalle de calcul de WFP*
▷ *wfp : le pourcentage de paquets en échec*
▷ *RTT : le temps d'aller-retour*
▷ *pktsList : liste de tous les paquets envoyés pendant la période d'adaptation*
▷ *failList : liste de tous les paquets en échec pendant la période d'adaptation*
▷ *pktTime : le temps d'envoi d'un paquet*
▷ *pktNum : le numéro de séquence d'un paquet*

```
pour chaque période d'adaptation faire
    err ← 0
    pkts ← 0
    répéter
        envoyer un paquet
        pktsList ⟵ pktTime et pktNum
        si paquet en échec alors
            failList ⟵ pktTime et pktNum
        fin
        calculer le RTT
        sleep
    jusqu'à la fin de la période
    pkts ⟵ le nombre de paquets envoyés pendant le dernier RTT
    si pkts < 20 alors
        err ⟵ le nombre de paquets en échec sur les 20 derniers paquets envoyés
        wfp ⟵ err/20
    sinon
        err ⟵ le nombre de paquets en échec pendant le dernier RTT
        wfp ⟵ err/pkts
    fin
    si wfp = 0 alors
        augmenter le bitrate
    sinon si wfp < 5 % alors
        maintenir le bitrate
    sinon
        baisser le bitrate
    fin
fin
```

Chapitre 10

ÉVALUATION DES PERFORMANCES PAR DES EXPÉRIMENTATIONS RÉELLES

10.1 Présentation des tests

La figure 10.1 montre le réseau réel utilisé pour réaliser les expérimentations avec le programme présenté dans la section précédente. Une transmission vidéo en continu est mise en place entre un émetteur et un récepteur. Les deux sont connectés au réseau par une interface câblée. Une machine de limitation de trafic, appelée par la suite TCM, *Traffic Control Machine*, est ajoutée entre l'émetteur et le récepteur [1]. La machine TCM a deux cartes réseau (deux interfaces) : une interface filaire connectée à l'expéditeur et une autre sans-fil (54Mb/s) connectée à un point d'accès (AP pour *Access Point*). Le récepteur est connecté au même AP par son interface filaire. Des informations détaillées sur les paramètres des expérimentations sont données dans le tableau 10.1. Le *streaming* vidéo utilise une vraie vidéo, encodée en quatre qualités, 3Mb/s, 2Mb/s, 1Mb/s et 512kb/s, comme indiqué précédemment. La vidéo dure 180 secondes et le bitrate réel (à chaque seconde) pour chacun des quatre bitrates demandés est donné dans la figure 10.2.

Trois séries de tests sont effectuées. Pour la première série (série de *limitation du trafic*),

1. Il est préférable d'ajouter une machine intermédiaire parce que la limitation du trafic sur l'expéditeur DCCP ne fonctionne pas actuellement (voir le thread « DCCP_BUG called », initié par nous, sur la liste de diffusion DCCP, disponible sur `http://www.spinics.net/lists/dccp/msg04264.html`). La mettre sur le point d'accès n'est pas intéressant non plus, parce qu'il a un ancien noyau Linux. Enfin, la limitation de trafic sur le récepteur ne donne pas des résultats précis.

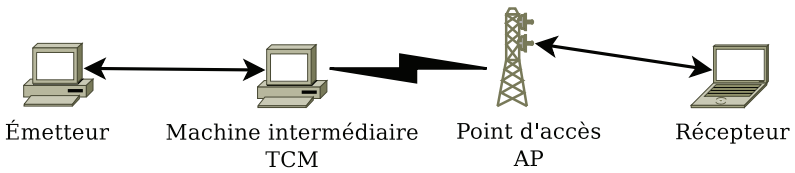


Figure 10.1 – Topologie du réseau utilisé pour réaliser les expérimentations.

Paramètre	Valuer du paramètre
Lieu	dans un bâtiment
Taille du paquet	1024 octets
PC1 (émetteur): carte filaire	
Produit	82567LM-3
Technologie	Gigabit Connection
Marque	Intel Corporation
PC2 (shaper machine): carte sans-fil	
Produit	BCM4312
Technologie	802.11b/g
Marque	Broadcom Corporation
PC2 (shaper machine): carte filaire	
Produit	RTL8111/8168B
Technologie	Gigabit Connection
Marque	Realtek
PC3 (recepteur): carte filaire	
Produit	BCM5755M
Technologie	Gigabit Connection
Marque	Broadcom Corporation
PC1,2&3 La bande passante filaire	100Mb/s
PC1,2&3 OS	Linux (Ubuntu 64bits)
PC1,2&3 OS, le noyau	2.6.35 generic
DCCP	version du noyau
Point d'accès (AP)	LINKSYS
Technologie de l'AP	Wireless-G
Bande passante sans-fil	54Mb/s
Distance (AP $\leftrightarrow$ PC2)	50cm
Réseau partagé	no

Tableau 10.1 – Paramètres réels du réseau utilisés pour réaliser les expérimentations.

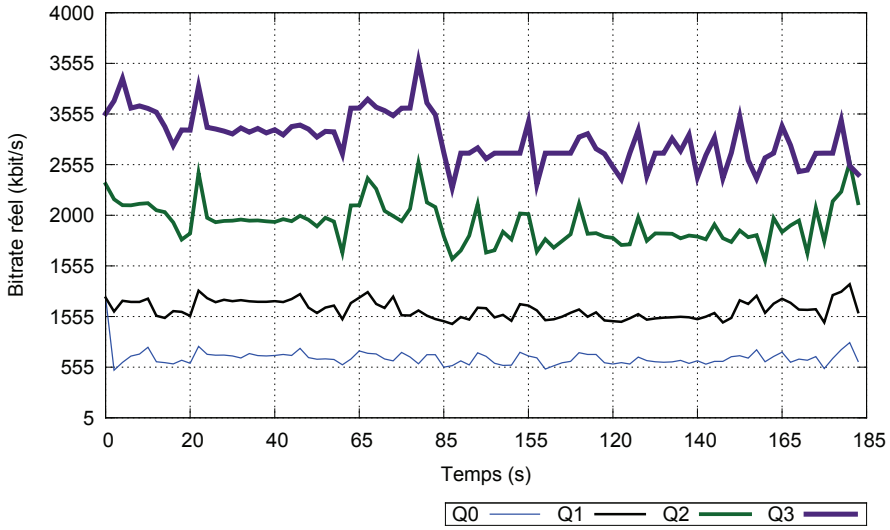


Figure 10.2 – Distribution de bitrate réel pour les quatre bitrates théoriques (demandés) : 512kb/s, 1Mb/s, 2Mb/s and 3Mb/s.

un seul flux est présent à tout moment au cours de la transmission vidéo ; ce flux peut donc utiliser tout le débit disponible. D'autre part, le débit disponible en sortie de la machine TCM est changée trois fois afin de simuler une bande passante variable : 600kb/s pour la première minute, 2300kb/s pour la deuxième minute, et la limitation du trafic a été interrompue pendant la dernière minute, ce qui rend le débit disponible en sortie pendant cette minute égal au débit disponible sans-fil d'origine [2], voir figure 7.1, page 77.

Pour la deuxième série, deux nombres de flux simultanés sont présents pendant toute la transmission : cinq flux et dix flux. Cette série sera appelée dans la suite « série à *flux sans écart* »). Cela permet de voir ce qui se passe lorsque plusieurs flux se partagent la bande passante, en particulier pour vérifier si cela conduit à une oscillation importante des performances globales. En réalité, le débit des données envoyées par cinq flux avec le bitrate maximum (3Mb/s) est comparable au débit disponible du réseau (5*3Mb/s=15Mb/s, ce qui est équivalent au débit disponible effectif fourni par un réseau sans-fil ayant 54Mb/s théorique). Pour dix flux avec le bitrate maximum (3Mb/s), le débit total dépasse largement le débit disponible (10*3Mb/s=30Mb/s). Lorsque le nombre de flux concurrents passe de cinq à dix, la fréquence d'adaptation de la vidéo, autrement dit le nombre de changement de qualité, sera plus grande.

Dans la troisième série, un nombre variable de flux jusqu'à un maximum de douze flux sont présents en même temps. Chaque flux commence 30 secondes après le début du flux précédent, sauf le premier qui commence au temps 0. Cette série sera appelée par la suite « série à *flux avec écart* ».

Aditionnellement, pour les deuxième et troisième séries, chaque flux attend un temps aléatoire entre 0 et 2 secondes avant de démarrer, afin d'éviter que les flux changent de bitrate exactement en même temps toutes les deux secondes. Tous les flux d'une même expérimentation utilisent des algorithmes identiques (autrement dit tous utilisent l'adaptation, ou tous utilisent une vidéo avec un bitrate fixe).

En outre, la première série est répétée dix fois tandis que les deuxième et troisième séries d'expérimentations sont répétées cinq fois chacune. En réalité, lors d'une expérimentation, une méthode peut bénéficier de bonnes conditions réseau et ainsi prendre l'avantage sur les autres. Effectuer 5 répétitions minimise la probabilité qu'un tel phénomène se reproduise. Une seule répétition représentative des cinq est considérée ici, ainsi que la moyenne globale de toutes les répétitions d'expérimentations. À noter qu'il n'y a pas de retransmission des paquets perdus dans l'ensemble de nos tests (comportement de DCCP par défaut).

Ce chapitre a donc pour objectif de :

1. Vérifier si l'adaptation de la vidéo faite par VAAL fonctionne bien en prenant en compte les retours (feed-back) du tampon DCCP. Cela est présenté dans la section 10.2.1, « La variation de la qualité ».
2. Évaluer les performances de VAAL par rapport à une transmission classique de la vidéo avec un encodage statique. Ce point est présenté dans la section 10.2.2, « Évaluation de la performance de l'adaptation ».

2. Des détails sur l'utilisation de la limitation du trafic sous un système d'exploitation (OS) Linux est donnée dans la section 2.6.

10.2 Évaluation de VAAL

10.2.1 Variation de la qualité (l'adaptation)

Comme précédemment, VAAL mesure le pourcentage d'échec d'écriture (WFP) dans le tampon du socket de DCCP toutes les deux secondes. VAAL décide ensuite d'augmenter, de maintenir ou de diminuer le bitrate de la vidéo envoyée. Les résultats d'un seul flux en cas de limitation de trafic sont présentés, puis pour cinq et dix flux de la série à flux sans écart. Pour une meilleure visualisation, le pourcentage de *réussite* d'écriture (WSP, *Write Success Percentage*) est tracé au lieu de WFP. WSP est tout simplement égal à 1 – le pourcentage d'échec d'écriture (WSP = 1–WFP).

Pour toutes les figures présentées dans cette section, l'axe des abscisses représente le temps de 0 à 180s, la durée de la transmission vidéo en continu dans nos expérimentations. Le bitrate de la vidéo transmise et le pourcentage de réussite d'écriture (WSP) sont mis sur l'axe des ordonnées. Trois courbes sont tracées dans chaque graphique : la première courbe trace le bitrate théorique de la vidéo en cours q_t, demandé par VAAL lors de la transmission et utilisé pour calculer la qualité de la période suivante (pendant la phase de sélection de la qualité). La deuxième courbe montre le bitrate réel de la vidéo q_r au moment de la transmission. La troisième courbe donne le pourcentage de réussite d'écriture (WSP).

Un seul flux en cas de limitation de trafic

Les résultats de ce test sont illustrés dans la figure 10.3. Comme prévu, pendant la première minute, durant laquelle le débit disponible est limité à 600kb/s, VAAL alterne entre la plus basse qualité (512kb/s), avec le WFP=0, et 1Mb/s deux secondes plus tard, avec une grande valeur pour WFP. Au cours de la deuxième minute, où le débit disponible est limité à 2300kb/s, VAAL découvre la nouvelle bande passante disponible et utilise la plupart du temps un bitrate vidéo entre 2 et 3 Mb/s qui sont supérieures à 512Kb et 1Mb/s utilisés pendant la première minute. Dans la dernière période, la machine intermédiaire TCM stoppe la limitation du trafic, et VAAL envoie le plus haut bitrate (haute qualité). Enfin, la dernière minute montre également que lorsque le débit disponible est plus élevé que la plus haute qualité, la commutation entre les qualités ne se produit pas, donc VAAL est comparable à une transmission de la plus haute qualité sans adaptation.

Bien que l'adaptation fonctionne, ce graphique montre un défaut important de VAAL. Pendant la première minute et aussi pendant la deuxième minute, VAAL change constamment la qualité entre deux valeurs (512kb/s et 1Mb/s), une valeur plus petite que la capacité de bande passante et l'autre plus grande. Une telle oscillation est gênant pour l'utilisateur final. Ce problème sera traité dans le chapitre 11.

Cependant **VAAL détecte bien le débit disponible et change la qualité autour de sa valeur.**

Cinq flux concurrents sans écart de temps de démarrage

Dans cette série d'expérimentations, cinq flux concurrents sont en cours de transmission. Ils utilisent tous la bande passante en même temps. La figure 10.4 présente les résultats pour un

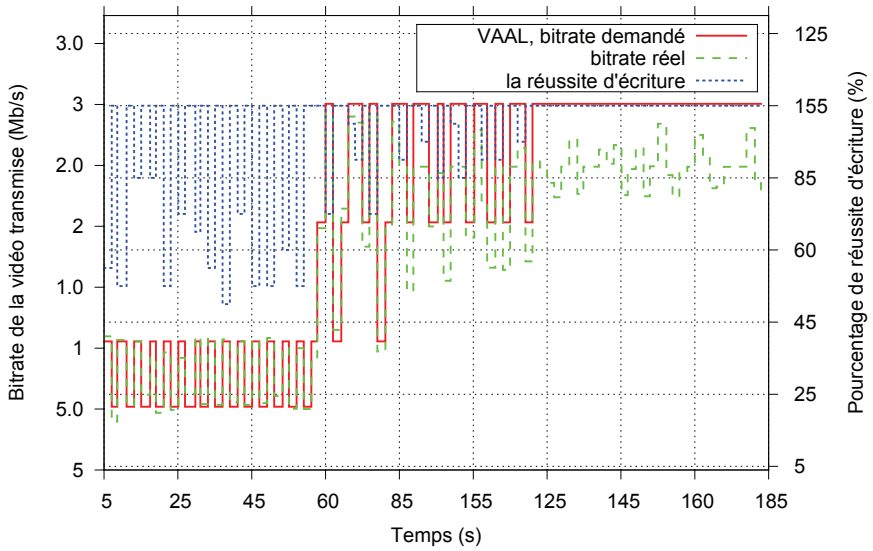


Figure 10.3 – Adaptation de la vidéo en cas de limitation de trafic.

flux représentatif de ce test.

Tout d'abord, on remarque que la sélection de la qualité initie la connexion à un bitrate de 1Mb/s, comme expliqué précédemment, puis l'augmente lentement pour atteindre la plus haute qualité. Par la suite, la qualité est souvent à 3Mb/s, car le débit disponible est en principe suffisant pour permettre à cinq flux concurrents de transmettre une vidéo avec une haute qualité. Lorsque le débit disponible n'est plus suffisant (certaines périodes entre 108 et 124 secondes, à cause des interférences sur la liaison sans-fil par exemple) la qualité est diminuée. À constater également que lorsque le pourcentage de réussite d'écriture (WSP) est très faible, la qualité est directement passée à la plus petite (par exemple à la seconde 108). Au contraire, lorsque WSP est à 100%, la qualité est augmentée lentement (basculée au bitrate immédiatement supérieur toutes les deux secondes, par exemple à la seconde 110, puis à 112 et après à 114). En cas de baisse temporaire de WSP, la diminution de la qualité est temporaire aussi et la qualité repasse au bitrate précédent (à la seconde 140 par exemple).

VAAL utilise donc efficacement le débit disponible et ne diminue la qualité que si le débit disponible devient moins important afin de garder la qualité au maximum possible.

Dix flux concurrents sans écart de temps de démarrage

Ce test est similaire au précédent, mais les flux concurrents sont en nombre de dix et non de cinq, afin d'augmenter la charge du réseau par plus de trafic. La figure 10.5 présente les résultats pour un seul flux aussi. Il est noté tout d'abord que l'adaptation se fait, car le flux ne reste pas à un seul bitrate. Ensuite, comme pour cinq flux, il est remarqué que lorsque WSP est très faible, la qualité choisie par VAAL est très faible aussi, par exemple à la seconde 20. En outre, étant donné qu'il y a dix flux qui sont en concurrence pour partager la bande passante, le bitrate est fréquemment changé. Par conséquent, VAAL assure que la qualité est une fonction du débit disponible.

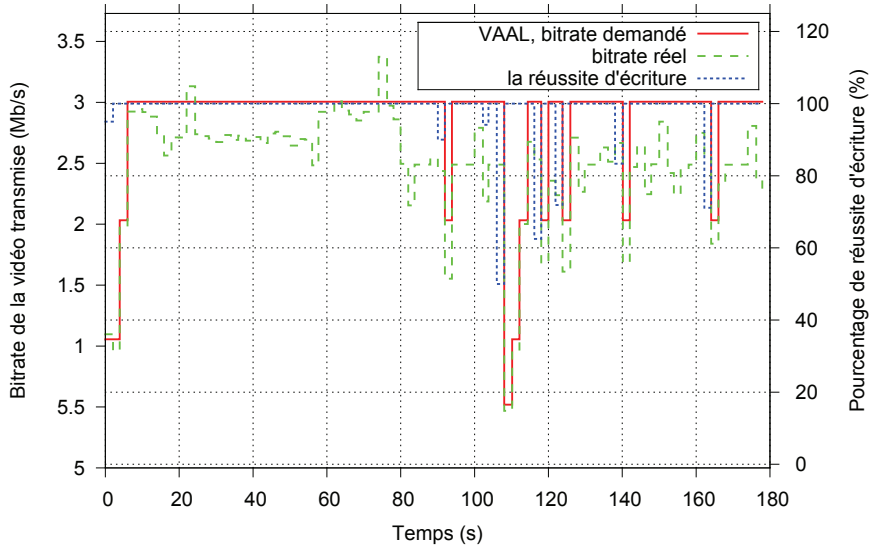


Figure 10.4 – Adaptation de la vidéo pour cinq flux concurrents sans écart de temps de démarrage (un des cinq flux).

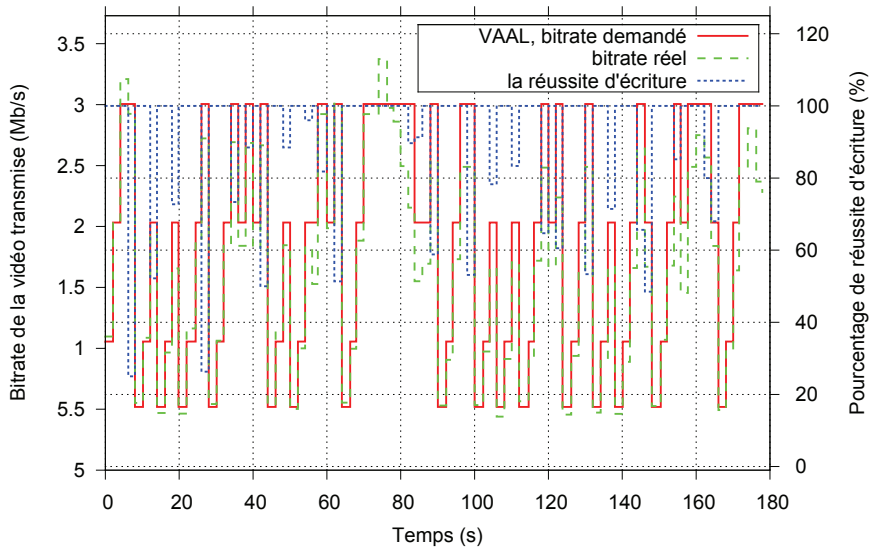


Figure 10.5 – Adaptation de la vidéo pour dix flux concurrents sans écart de temps de démarrage (un des dix flux).

Bien que VAAL envoi parfois à un bitrate supérieur au débit disponible quand WSP=100%, mais il diminue tout de suite après. **Cela permet à VAAL de tenir un bitrate autour du débit d'envoi et une qualité maximal.** Comme cela a été indiqué, un traitement plus efficace est proposé au chapitre 11.

10.2.2 Évaluation des performances de VAAL

On compare les transmissions vidéo (en utilisant DCCP) avec et sans VAAL. Puisque cette thèse se place dans un cadre d'optimisation du transfert (afin d'améliorer la qualité de la vidéo), les métriques de comparaison utilisés sont le nombre de paquets reçus et le nombre de paquets perdus. On suppose que si une nouvelle méthode est en mesure de maximiser le nombre de paquets reçus, tout en minimisant le nombre de paquets perdus, elle permettra d'améliorer la qualité vidéo reçue du côté réseau.

Il faudrait bien sûr prendre en compte le fait que VAAL a tendance à baisser son débit si le débit disponible diminue. Il y aura donc moins de paquets envoyés et aussi moins de paquets reçus. Ainsi, VAAL ne peut pas forcément amener à plus de paquets reçus qu'une méthode classique envoyant au maximum possible. En effet, le nombre de paquets reçus doit être plus ou moins équivalent pour les deux méthodes. En revanche, les différences seront visibles au niveau du taux de pertes et surtout sur la qualité de la vidéo visualisée par l'utilisateur.

Les résultats pour VAAL sont issus des expérimentations présentées au début de ce chapitre. Pour DCCP sans adaptation vidéo, les trois mêmes séries d'expérimentations ont été effectuées, séparément pour chacune des quatre qualités (Q3=3Mb/s, Q2=2Mb/s, Q1=1Mb/s et Q0=512 kb/s).

Dans chaque graphique qui suit il y a dix courbes, qui donnent le nombre de paquets envoyés et reçus pour chaque type de *streaming* : Q0, Q1, Q2, Q3 et VAAL. Afin de les distinguer plus facilement, les courbes pour les paquets envoyés et reçus pour chaque type emploient le même style de symbole, par exemple c'est un cercle plein pour VAAL.

À noter aussi que, même si toutes les courbes sont tracées sur le même graphique, l'exécution a eu lieu à des *moments différents*. En outre, même si les points forment des lignes pour une meilleure visualisation, les flux et les répétitions, sur l'axe des abscisses, sont indépendants.

Un seul flux en cas de limitation de trafic

Les résultats de cette série d'expérimentations sont présentés dans la figure 10.6 pour les 10 répétitions. Cette figure montre que pour chaque répétition les résultats sont très proches les uns des autres (même nombre de paquets envoyés et reçus), ce qui est prévisible car il n'y a qu'un seul flux impliqué dans chaque test.

Plusieurs conclusions peuvent encore être tirées :

- le nombre de paquets reçus pour VAAL est légèrement plus petit que DCCP Q3, mais il y a une énorme différence entre eux en ce qui concerne le taux de perte (environ 35% de paquets perdus pour DCCP Q3 contre 13% pour VAAL). VAAL est donc meilleur que DCCP Q3.
- le nombre de paquets reçus pour VAAL est 15% plus grand que DCCP Q2, tandis que son nombre de paquets envoyés est seulement 2,5% plus grand. VAAL est donc incontestablement meilleur que DCCP Q2.

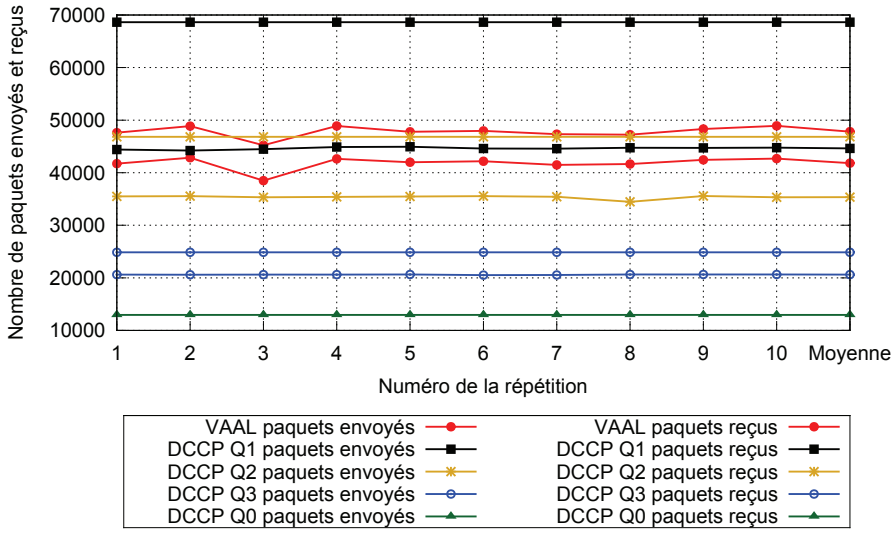


Figure 10.6 – Comparaison du nombre de paquets envoyés et reçus pour un seul flux en cas de limitation de trafic : VAAL a le taux de pertes le plus petit.

Qualité	*Nombre de paquets reçus*	
	(théorie)	*(expérimentation)*
Q3	45312	44635
Q2	35328	35355
Q1	19986	20604
Q0	11520	12142
Ideal	42240	41811

Tableau 10.2 – Comparaison entre la théorie et les expérimentations en cas de limitation de trafic.

– il est clair que VAAL est meilleur que DCCP Q1 et DCCP Q0 tant l'écart est important avec eux.

La comparaison avec les résultats théoriques, données dans le tableau 7.1 page 78, est très intéressante. Non seulement les résultats expérimentaux montrent, comme en théorie, que l'adaptation est meilleure que la transmission avec un bitrate statique (voir figure 10.6), mais surtout que les nombres de paquets reçus sont similaires, cf. table 10.2. Cette similitude valide ainsi les résultats de cette série de tests.

En conclusion, VAAL surpasse toute méthode statique dans le cas d'une importante variation du débit disponible.

Enfin, il y a quelques commentaires sur le taux de perte de paquets de 13% pour VAAL. Comme on peut le voir également dans certaines expérimentations qui suivent, ce taux est soit proche de 0 dans le cas où le débit disponible permet l'envoi de la plus haute qualité, c'est-à-dire que la commutation de la qualité se produit rarement, soit entre 10% et 20% lorsque la commutation de la qualité se produit souvent car le débit disponible n'est pas assez élevé. L'utilisation de ZAAL (cf chapitre 11) permet de réduire encore ce pourcentage de pertes.

Cinq flux concurrents sans écart de temps de démarrage

La figure 10.7(a) présente les résultats d'une répétition représentative de la série d'expérimentations avec cinq flux sans écart. Comme prévu, le débit disponible laisse passer les cinq flux avec le bitrate le plus élevé. Le nombre de paquets envoyés et reçus pour VAAL est comparable à une transmission classique avec une qualité Q3 ; pour certains flux, VAAL a des performances légèrement meilleures, tandis que pour d'autres, DCCP Q3 est légèrement meilleure. Il peut également être remarqué que le nombre moyen de paquets reçus pour VAAL et DCCP Q3 est largement supérieur à Q2, Q1 et Q0, comme prévu lorsque le débit disponible est suffisamment élevé.

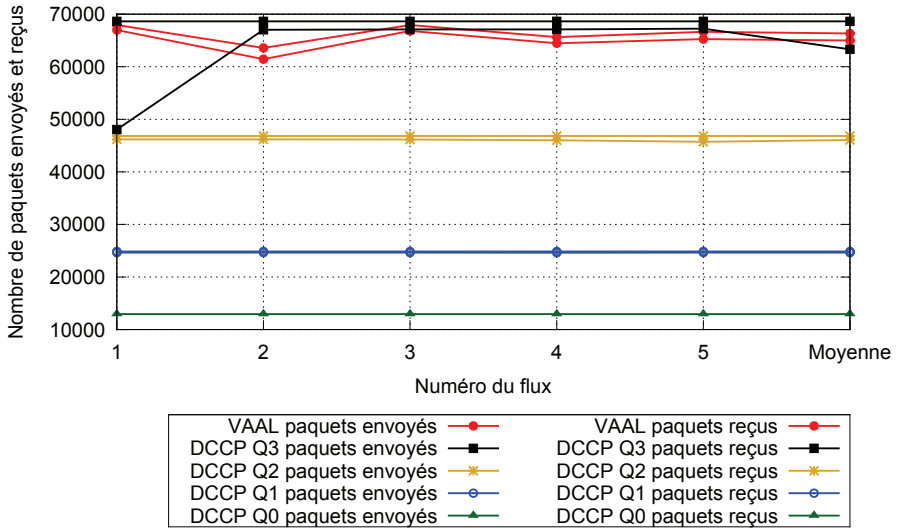
La figure 10.7(b) (moyenne des cinq répétitions) donne les mêmes résultats. À l'exception de la répétition 2, VAAL est meilleure qu'une transmission classique : bien que son nombre de paquets reçus soit similaire à DCCP Q3, son nombre de paquets perdus est largement inférieur à Q3. Il est également important de noter que la répétition 2 (où VAAL ne fonctionne pas bien, sans doute à cause de certaines conditions de réseau difficile lorsque le test a été exécuté) montre clairement que VAAL a été en mesure de s'adapter aux conditions du réseau : il a diminué le nombre de paquets envoyés, par conséquent il maintient un taux de perte relativement faible, ce qui n'est pas le cas de la transmission classique.

Dix flux concurrents sans écart de temps de démarrage

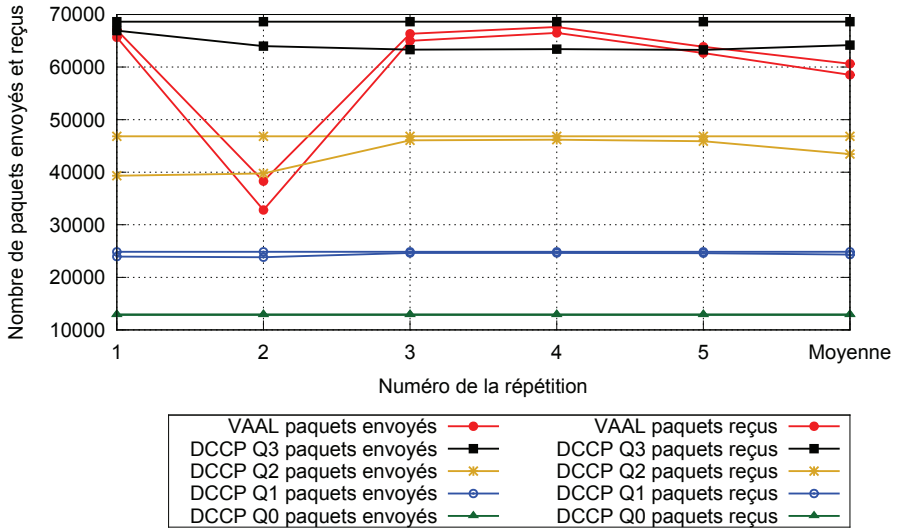
Cette série d'expérimentations est similaire à la précédente mais emploie dix flux concurrents. La figure 10.8(a) présente une répétition représentative. En ce qui concerne VAAL, tous les flux adaptent leur taux d'envoi au débit disponible. Il est important de noter que la différence entre les paquets envoyés et reçus, la différence entre les deux courbes de VAAL dénotées avec des cercles pleins, est identique pour tous les flux. En outre, tous les flux partagent équitablement le débit disponible car ils ont à peu près le même nombre de paquets envoyés et reçus. Il est à noter que le flux 8 de DCCP Q2 a un nombre important de pertes, mais, contrairement à VAAL dans la série précédente d'expérimentations, il continue d'envoyer à la même vitesse, ce qui démontre une fois de plus l'avantage de l'adaptation par rapport à une transmission classique avec des bitrates statiques.

La figure 10.8(b) (moyenne des cinq répétitions) montre que le nombre de paquets reçus pour tous les flux est largement plus petit qu'à la série précédente pour cinq flux (c'est le cas aussi pour DCCP avec Q3 et Q2). Cela était attendu compte tenu de l'augmentation du nombre de flux (de 5 à 10) partageant le même lien. Ensuite, par rapport à DCCP Q3, VAAL a environ 12% de moins de paquets reçus, mais DCCP Q3 a un grand nombre de paquets perdus pour tous ses flux, environ 38% des paquets envoyés, ce qui indique que Q3 est en effet trop agressif pour ce réseau. Ce taux de perte élevé de paquets affecte considérablement la qualité vidéo du côté du récepteur. Par rapport à DCCP Q2, VAAL envoie 5,5% de moins de paquets et reçoit 3,4% moins des paquets, il est donc difficile de dire qui est meilleure.

La même conclusion peut être tirée : **avec la transmission statique classique, l'application continue à envoyer à la même vitesse quel que soit l'état du réseau**. Dans des conditions de l'expérimentation, le débit disponible est presque suffisant pour faire passer les dix flux avec un bitrate de 2Mb/s, et VAAL est en mesure de trouver cette valeur.

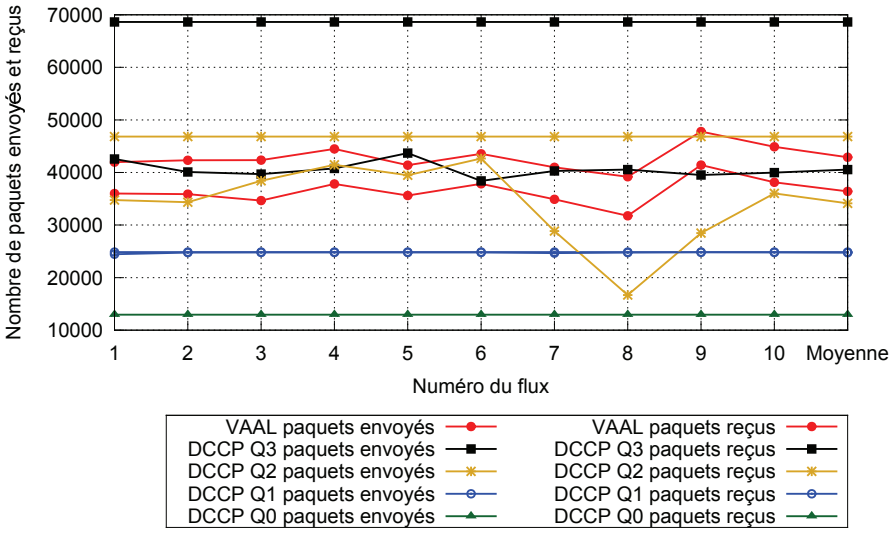


(a) Une répétition représentative : tous les flux de VAAL envoient à la plus haute qualité, tout comme DCCP Q3.

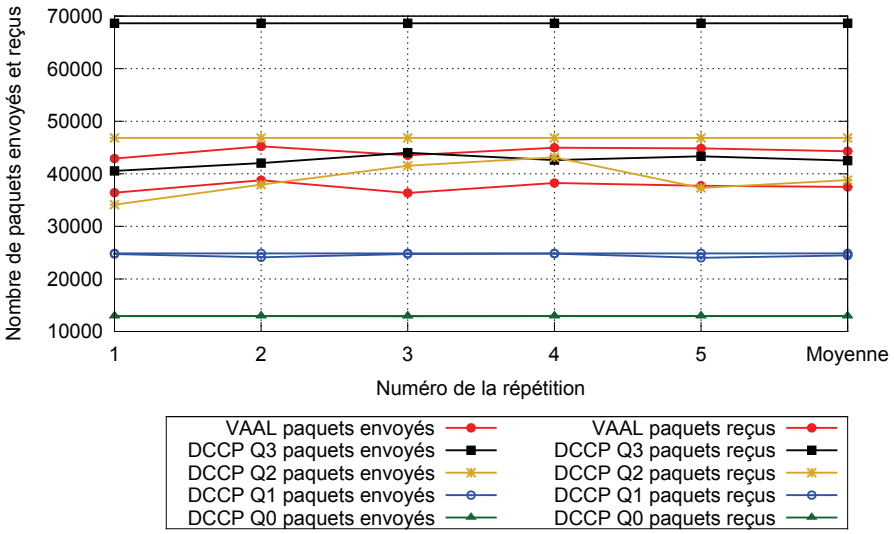


(b) La moyenne des cinq répétitions : étant donné que le débit disponible est suffisant pour tous les flux concurrents, VAAL est comparable à la transmission classique de la plus haute qualité.

Figure 10.7 – Comparaison du nombre de paquets envoyés et reçus pour cinq flux sans écart de temps de démarrage.



(a) Une répétition représentative : tous les flux de VAAL adaptent leur bitrates, ce qui donnent un taux de pertes beaucoup inférieur au flux de DCCP Q3.



(b) La moyenne des cinq répétitions : VAAL a un taux de réception et de pertes comparable à DCCP Q2.

Figure 10.8 – Comparaison du nombre de paquets envoyés et reçus pour dix flux sans écart de temps de démarrage.

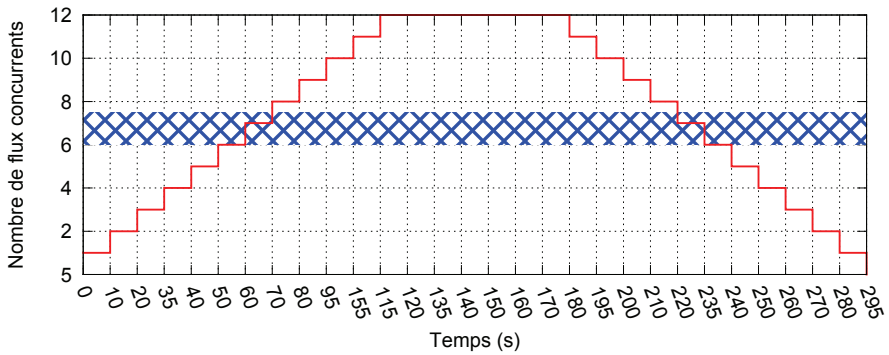


Figure 10.9 – Nombre de flux simultanés à tout moment pour douze flux avec un écart de temps de démarrage de 10s.

Jusqu'à 12 flux concurrents avec un écart de temps de démarrage de 10 secondes.

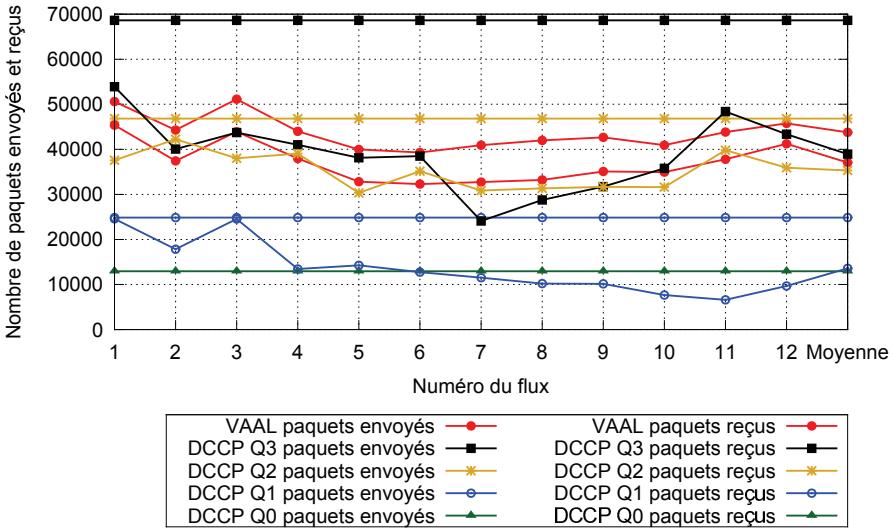
Dans cette série d'expérimentations, jusqu'à 12 flux simultanés sont en concurrence pour le débit disponible réseau. Cela permet de comparer les performances dans une situation dynamique plus réaliste où le nombre de flux varie avec le temps. En conséquence, le meilleur bitrate statique ne peut pas être connu avant le début de la transmission.

Comme déjà précisé, il y a environ 10 secondes d'écart entre le début de deux flux consécutifs. La figure 10.9 présente le temps de départ des 12 flux concurrents et le nombre de flux simultanés à tout moment. Par exemple, le flux numéro 7 commence à la seconde 60, et il y a 6 flux simultanés entre 50 et 60 secondes. Si on ajoute 180s au temps de départ, cela donne le temps de fin de transmission pour chaque flux, par exemple, ce temps est de 60+180=240s pour le flux 7. Il y a aussi une zone floue pour indiquer théoriquement le nombre de flux simultanés possibles à la plus haute qualité (3Mb/s) sans dépasser le débit disponible (c'est plus ou moins 7 flux). À remarquer que les flux avec le plus haut degré de simultanéité sont 6 et 7 : au cours de leur vie, le nombre de flux simultanés est compris entre 6 et 12. Le degré de concurrence subi par un flux diminue lorsqu'il s'éloigne de 6 ou de 7 (flux 5 et 8 expérimentent au moins 5 flux simultanés, les flux 4 et 9 ont au moins 4 flux simultanés, etc.) Pour l'énoncer simplement, le degré de la concurrence diminue avec la distance entre le numéro de flux et le milieu (6,5).

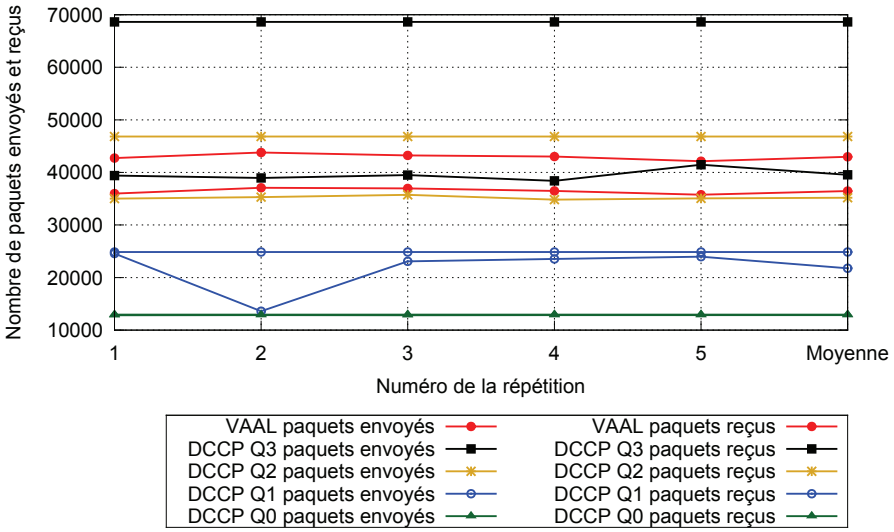
La figure 10.10(a) présente une répétition représentative. Comme prévu, les flux au milieu, par exemple les flux 5 à 8, ont moins de paquets reçus. Pour VAAL, la courbe du nombre de paquets reçus ainsi celle des paquets envoyés diminuent généralement à partir du flux numéro 1 jusqu'aux flux au milieu (6 et 7) et augmentent ensuite, c'est-à-dire que généralement les flux envoient/reçoivent des paquets en fonction du degré de concurrence qu'ils subissent. Cela est contraire à la transmission statique (Q3 et Q2, sans adaptation), où les flux au milieu expérimentent une baisse de nombre de paquets reçus beaucoup plus élevé que les autres flux.

Dans le même graphique, on peut prêter attention à DCCP Q1, qui en présence de certaines mauvaises conditions du réseau, peu de paquets seulement sont reçus, a un taux d'échec très élevé pour presque tous ses flux, ce qui bien sûr conduit à une mauvaise qualité vidéo lors de cette période.

La moyenne générale pour toutes les répétitions, illustrée dans la figure 10.10(b), confirme la supériorité de VAAL sur toutes les transmissions statiques classiques, par exemple VAAL et



(a) Une répétition représentative : les flux au milieu subissent plus de concurrence, mais quand l'adaptation est utilisée ils ne perdent pas plus de paquets par rapport à leurs équivalents sans adaptation.



(b) La moyenne des cinq répétitions : grâce à VAAL, le taux de pertes est le plus petit par rapport à DCCP Q2 et Q3.

Figure 10.10 – Comparaison du nombre de paquets envoyés et reçus pour douze flux avec un écart de temps de démarrage de 10 sec.

DCCP Q3 ont environ le même nombre de paquets reçus, mais VAAL a 40% de pertes en moins ; aussi, Q2 et VAAL DCCP ont presque le même nombre de paquets reçus, mais VAAL envoie moins de paquets et a 10% de pertes de paquets en moins.

En conclusion, une fois de plus, avec une adaptation vidéo (VAAL) la bande passante est utilisée plus efficacement, surtout quand elle change de manière dynamique ou lorsque le réseau expérimente de mauvaises conditions.

10.3 Conclusions

Nos expérimentations réelles confirment notre hypothèse, à savoir que le débit de la vidéo transmise par VAAL est façonné par le débit disponible du réseau, et en général VAAL soit limite le taux de pertes de paquets, soit évite une sous-utilisation du débit disponible réseau. En outre, l'utilisation d'un protocole de transport (DCCP dans notre implémentation) avec un contrôle de congestion (TFRC) garantit la compatibilité de notre méthode avec le protocole de transport classique TCP, VAAL est *TCP-friendly*.

Ce chapitre a par conséquence validé deux idées :

1. VAAL prend bien en compte les retours de DCCP afin de trouver le débit disponible, et par conséquent d'adapter le bitrate ;
2. l'adaptation faite par VAAL est meilleure qu'une transmission classique sans adaptation.

Ce chapitre révèle également quelques points intéressants qui, d'une part, méritent d'être mentionnés et, d'autre part, appellent à plus d'analyse :

- Lorsque le débit disponible est supérieur à la plus haute qualité disponible et que sa variation est petit, VAAL se comporte comme une méthode classique de transmission avec la plus haute qualité disponible. Tout l'intérêt de VAAL dans ce cas est qu'elle transmet cette qualité sans intervention de l'utilisateur final, donc le choix de la meilleure qualité disponible est automatique.
- En termes de paquets envoyés et reçus, une méthode d'adaptation n'est pas forcément toujours la meilleure. En revanche, l'adaptation est dynamique et permet de se repositionner au-dessus du débit disponible si les conditions réseau changent. Imaginons par exemple qu'une personne regarde une vidéo avec un débit disponible suffisant pour la meilleure qualité. Si à un moment donné cet utilisateur commence à télécharger d'autres données, le débit disponible sera partagé par les différents flux et ne sera plus suffisant pour la vidéo affichée. Une méthode d'adaptation telle que VAAL changera de qualité de façon autonome, empêchant ainsi les pauses intempestives de la vidéo.
- Un point négatif cette fois-ci, qui a été déjà mentionné dans ce chapitre, est que dans son état actuel et lorsque le débit disponible se situe entre deux qualités, VAAL continue à commuter entre les deux valeurs, ce qui provoque un taux important de commutations de qualité et un très grand nombre de paquets perdus.

Le chapitre suivant traite de ce dernier point. Il présente un algorithme (ZAAL) permettant de réduire davantage le nombre de paquets perdus, en utilisant un historique des décisions prises par VAAL.

Chapitre 11

ÉVITEMENT DE ZIGZAG PRODUIT PAR UN ALGORITHME D'ADAPTATION

Comme nous l'avons vu dans le chapitre précédent, l'adaptation peut parfois conduire à un changement constant entre deux qualités (bitrates) dont l'une est plus petite que le débit disponible et l'autre est plus grande. Par exemple, pour un utilisateur connecté à Internet avec un débit disponible de 2.5Mb/s et pour une vidéo encodée en 2 qualités, 2Mb/s et 3Mb/s, un algorithme adaptatif de *streaming*, comme VAAL, alterne constamment entre ces deux qualités si bien sûr aucune mesure n'est prise en compte. Évidemment, la meilleure solution serait de rester avec 2Mb/s beaucoup plus de temps avant de retenter la qualité 3Mb/s. Puisque nous ne savons pas à quel moment le débit disponible pourrait être plus grand que 3Mb/s, l'algorithme d'adaptation devrait retenter une qualité supérieure de temps en temps. Ce changement constant de qualité entre deux valeurs crée ce que nous appelons « commutation de qualité en zigzag », (ZQS, *zigzag quality switching*).

ZQS est causé par toute application adaptative de la vidéo en *streaming* lors de l'utilisation d'un bitrate supérieur à la capacité du réseau, et puis juste après revenir à un bitrate inférieur. Afin de délivrer la meilleure qualité visuelle possible de la vidéo à l'utilisateur, cette application devrait éviter ce problème de zigzag. ZQS peut être considéré comme un problème de congestion, mais sur la partie serveur et non pas sur la partie réseau. Or, malgré la similitude des deux problèmes, les hypothèses et les buts sont différents, et les solutions aussi. En outre, les techniques d'estimation de la bande passante représentent une solution potentielle pour le problème de zigzag, mais leur utilisation entraîne des modifications très complexes et peu pratiques. Nous discutons ces deux points dans la section suivante.

Peu de travaux traitent cette question, et ils ne la résolvent pas vraiment. Ce chapitre présente une solution à ce problème, appelée ZAAL (**Z**igzag **A**voidance **AL**gorithm). ZAAL utilise une valeur indiquant la capacité de réussite de chaque bitrate. Cette valeur est utilisée par le suite pour décider si une qualité supérieure peut être choisie ou non, empêchant ainsi le zigzag.

Nous présentons également l'intégration de ZAAL dans la méthode d'adaptation de la vidéo au niveau de la couche application sur le serveur (VAAL). VAAL se comporte ainsi comme il a été expliqué dans le chapitre 9 sauf qu'au moment du changement de qualité dans la phase de sélection de qualité, elle consulte ZAAL pour savoir si une qualité supérieure est autorisée ou

non. Si ZAAL décide que oui, ce qui veut dire que la capacité de réussite du bitrate supérieur est acceptable, VAAL augmente la qualité. Sinon, elle la maintient au niveau actuel.

Dans ce chapitre, le problème que nous essayons de résoudre (ZQS) est formulé dans la section 11.1. Ensuite, ZQS est comparé à d'autres problèmes similaires existants dans la section 11.2. L'algorithme de ZAAL est présenté et analysé dans la section 11.3 comme une solution au problème ZQS, et sa performance est évaluée par des expérimentations réelles dans la section 11.4.

11.1 Formulation du problème ZQS

La « commutation de qualité en zigzag » est le fait qu'une application émettrice d'une vidéo en streaming adaptatif change constamment la qualité de la vidéo entre deux valeurs, surtout quand une valeur est plus grande du débit disponible et l'autre est plus petite.

Dans un réseau avec un débit disponible variable, l'adaptation vidéo est très importante. Elle vise à adapter le débit de la vidéo aux caractéristiques du réseau et donc à améliorer la qualité de la vidéo perceptible par l'utilisateur final. Comme nous l'avons présenté dans le manuscrit de cette thèse, l'adaptation peut se faire à plusieurs couches OSI, dont la couche application par exemple. Pour adapter la vidéo à l'état du réseau, l'application émettrice contrôle des paramètres vidéo. Pour atteindre cet objectif, l'application doit vérifier constamment le débit disponible (chaque 2 sec. ou chaque image I etc.) et d'adapter le bitrate de la vidéo à la valeur obtenue en conséquence.

Comme exemple concret, dans VAAL le contrôle de congestion du côté réseau est laissé à la charge du protocole de transport. C'est lui qui décide à quel débit les données peuvent quitter la machine et entrer dans le réseau. L'application se contente de mettre ses données dans la mémoire tampon de la socket du protocole de transport (voir la figure 9.1 page 97). Si le débit disponible est inférieur au débit des données envoyées par l'application, la mémoire tampon se remplit, et lorsqu'elle est pleine, les paquets supplémentaires générés par l'application ne pourront pas y être écrits. Nous avons appelé ces paquets « paquets en échec ». Dans ce cas, le débit des données générées par l'application doit baisser. En revanche, lorsque VAAL choisit un bitrate qui ne provoque pas de paquets en échec, l'application tente une qualité supérieure pour se réadapter au débit disponible et améliorer la qualité de vidéo transmise [1]. Ce processus est répété à la fin de chaque période d'adaptation, à savoir toutes les 2 secondes dans nos expérimentations. Maintenant, si la valeur du débit disponible est entre deux bitrates, VAAL va constamment, également toutes les 2 secondes, commuter la qualité entre eux créant un zigzag (le problème ZQS).

Ainsi, toute méthode d'adaptation de la vidéo, comme VAAL, commute constamment la qualité entre deux valeurs : une avec un bitrate plus grand que le débit disponible et avec des paquets en échec, et une autre avec un bitrate plus petit que le débit disponible et sans paquets en échec. Nous avons remarqué ce phénomène dans nos expérimentations. La figure 11.3(a) (page 137) donne un exemple clair, où durant la première minute le bitrate de la vidéo est continuellement basculée entre 0,5Mb/s et 1 Mb/s.

Ce problème de commutation de qualité en zigzag (ZQS) a deux effets indésirables :

1. Le chapitre 9 détaille la méthode d'adaptation VAAL.

– De nombreux changements de bitrate, conduisant à une qualité vidéo instable. Il est préférable de conserver une qualité vidéo stable autant que possible tout en la maximisant.
– Gaspillage de ressources du processeur CPU et/ou du réseau, lorsque le bitrate est supérieur au débit disponible, entraînant une baisse de la qualité vidéo et des paquets perdus. En effet, les vidéos sont généralement plus dégradées par les pertes de paquets que par des bitrates plus faibles.

À noter que *le problème à résoudre n'est pas la réduction du nombre de changements de qualité*, par exemple lorsque de tels changements utilisent beaucoup de ressources (CPU, HDD etc.) *Il ne s'agit pas non plus de l'utilisation de la plus haute qualité disponible à un certain moment*, ce qui peut potentiellement conduire à un zigzag, par exemple, lorsque tout de suite après le débit disponible baisse. **Le problème à résoudre est le *zigzag*, ce qui signifie d'éviter de transmettre une qualité qui a été récemment utilisée une voire plusieurs fois sans succès**. En tant que tel, nous pensons qu'une solution qui ne tient pas compte du passé récent, d'une manière ou d'une autre, ne peut pas résoudre ce problème.

11.2 Positionnement par rapport à d'autres travaux

Nous avons constaté que ZQS a des similitudes avec d'autres problèmes, et avec deux techniques déjà trouvées dans la littérature.

11.2.1 Méthodes similaires

Il existe quelques méthodes qui traitent directement ou indirectement du problème de zigzag ZQS. Nous les présentons par la suite.

RLM, *Receiver-driven Layered Multicast*

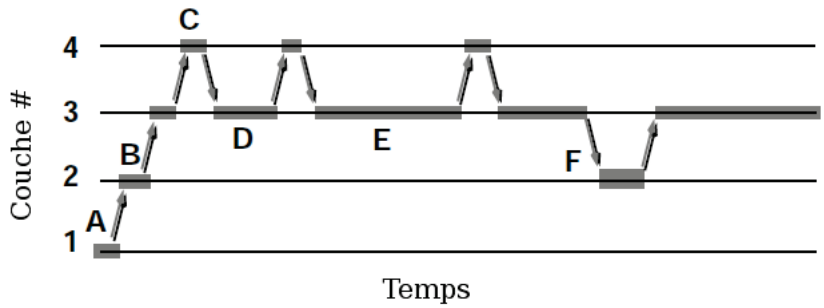


Figure 11.1 – RLM : démonstration de l'évitement du Zigzag

Une méthode qui peut être utilisée pour résoudre le zigzag est présentée dans [MJV96a] et [MJV96b]. Elle est proposée pour l'adaptation de la vidéo en *multicast*, mais elle peut être adaptée aux transmissions en *unicast*.

L'expéditeur envoie plusieurs couches, une couche de base et des couches d'amélioration, et le récepteur souscrit dynamiquement à une ou plusieurs d'entre elles. Le récepteur utilise un

temporisateur, qui est court au début, pour chaque niveau d'abonnement (couche). Lorsque le temporisateur expire et qu'aucune perte n'a été détectée, le récepteur s'abonne à un niveau supérieur et son temporisateur est démarré. Si au contraire le niveau conduit à des paquets perdus, le récepteur revient au niveau précédent et le temporisateur pour le niveau qui a provoqué des pertes est multiplicativement augmenté. Ainsi, RLM utilise un *backoff* exponentiel pour chaque couche ajoutée qui pose problème, autrement dit qui cause des pertes.

Pour mieux comprendre RLM, la figure 11.1 illustre sa stratégie du *backoff* exponentiel pour un hôte recevant jusqu'à quatre couches. Initialement, l'hôte (le récepteur) s'abonne à la couche 1 et définit un temporisateur (A). À ce stade, la durée du temporisateur est courte parce que la couche n'a pas encore posé problème. Une fois le temporisateur expiré, le récepteur s'abonne à la couche 2 et établit un nouveau temporisateur (B). Encore une fois, le temporisateur est court et la couche 3 est bientôt ajoutée. Le processus se répète à la couche 4, mais à ce stade une congestion se produit (C). Une fois que le récepteur détecte des paquets perdus, il redescend à la couche 3. Le temporisateur de la couche 3 est donc multiplicativement augmenté et un autre délai d'attente est prévu (D). Encore une fois, le processus se répète, la congestion est rencontrée, et le temporisateur est encore augmenté (E). Plus tard, une autre congestion transitoire amène le récepteur à descendre à la couche 2 (F). À ce point, puisque le temporisateur de la couche 3 est court, la couche est rapidement rétablie.

Dans cette méthode, il n'y a pas de limite supérieure pour le temporisateur, ce qui signifie que l'augmentation de la qualité peut être pénalisée pour un temps très long (pas d'adaptation pendant cette période), même si le débit disponible devient plus grand pendant le temps d'attente. Mais surtout, à chaque fois, seul le temporisateur du bitrate actuel est mis à jour, ce qui signifie que de bonnes conditions pour les bitrates supérieurs n'améliorent pas le temporisateur des bitrates inférieurs. Ces deux caractéristiques ne sont pas réalistes. Au contraire, notre algorithme basé sur EWMA ne présente pas ces inconvénients.

EFR, *Effective Frame Rate*

EFR, donné dans [Fen02], est un métrique pour évaluer les saccades (*jerkiness*) de la vidéo en utilisant le nombre de changements de la qualité. EFR utilise une formule pour calculer le nombre effectif d'images par seconde (*Effective Frame Rate*) d'une vidéo :

$$EFR = \sum_{i=1}^{N-1} \left(\frac{fps_i - P * qualitychange(max(i-W,0),i)}{N} \right) \tag{11.1}$$

où fps_i est le nombre d'images clés livrées dans la $i$ème seconde de la vidéo, W est la taille de la fenêtre, P est un facteur de pondération pour les variations du taux d'images, et $qualitychange(max(i-W,0),i)$ est le nombre de changements de qualité dans l'intervalle entre $i-W$ et i.

Cette formule est utilisée pour limiter le nombre de changements de qualité au cours de chaque fenêtre (W). Ainsi, EFR compte le nombre de changements de qualité, peu importe *où* ils apparaissent dans la fenêtre, qui est cependant un paramètre important de la saccade. Par exemple, 10 changements de qualité dans les 2 premières secondes d'une vidéo de 1 minute sont pires visuellement que 10 changements de qualité dispersés à travers toute la durée de la vidéo. Le but de EWMA, utilisé dans notre solution, est exactement de prendre aussi en compte le temps du changement, c'est-à-dire le temps de début de chaque période d'adaptation. En outre,

EFR utilise une fenêtre statique, c'est-à-dire qu'il regarde le nombre de changements pendant une période et après il attend la fin pour changer de période et refaire le calcul (déplacement à grain grossier), et fixe, c'est-à-dire que la fenêtre à toujours la même taille. En revanche, dans notre cas, la fenêtre est glissante et dynamique.

Enfin, notre objectif n'est pas de limiter le nombre de changements de qualité, mais d'éviter des augmentations de qualité qui entraînent des baisses de qualité tout de suite après.

Feamster et al.

La méthode d'adaptation décrite dans [FBB01] ne s'attaque pas directement au problème du zigzag, mais présente une façon pour lisser le bitrate de la vidéo envoyée et réduire la fréquence des changements de qualité vidéo. L'idée principale est que l'application ne passe pas à une qualité vidéo plus élevée, ou bien n'ajoute pas une couche supplémentaire dans le cas d'un encodage vidéo hiérarchique, jusqu'à ce que l'expéditeur soit certain que la vidéo continuera à jouer même après une réduction de la fenêtre de congestion.

Les auteurs de [FBB01] présentent des résultats pour un contrôle de congestion de type AIMD ou SQRT pour une transmission vidéo en *streaming*. Pour garantir la lecture vidéo en continu, la qualité de la vidéo envoyée doit être la plus haute qualité qui vérifie : $C < R - \beta R^l$, où C est le bitrate auquel la vidéo est encodée, R est le taux de transmission et $\beta = 1/2$ et $l = 1$ pour le contrôle de congestion de type AIMD et $l = 1/2$ pour le contrôle de congestion de type SQRT. Cette formule implique que la meilleure qualité vidéo ne dépasse pas la moitié du taux de transmission pour AIMD, et elle est de $\beta\sqrt{R}$ plus petite que le débit disponible pour SQRT.

Les résultats donnés dans [FBB01] montrent que cette formule fonctionne bien pour une application de *streaming* avec un contrôle de congestion AIMD. En revanche, le bitrate de la vidéo transmise est la plupart du temps bien inférieur au débit disponible. En ce qui concerne SQRT, l'algorithme a peu d'impact sur le zigzag. Les changements de qualité ne sont pas par conséquent significativement réduits. Enfin et surtout, cette formule ne résout pas vraiment le problème de zigzag ZQS car elle ne fait que réduire le bitrate artificiellement.

11.2.2 Positionnement par rapport aux techniques d'estimation de la bande passante

Étant donné qu'il existe des techniques qui visent à estimer le débit disponible au moment de la mesure, elles permettent également à un expéditeur d'utiliser l'information obtenue pour déterminer le bitrate qui correspond le mieux au débit disponible et d'interdire l'utilisation de ceux qui y sont supérieurs.

La technique de la paire de paquets

Une technique bien connue est la paire de paquets (*packet pair*) [Jac88a]. L'idée principale de cette technique est que l'expéditeur envoie une paire de paquets de la même taille vers une certaine destination, un serveur spécialisé ou simplement un récepteur informé. La destination répond alors en envoyant un écho pour chaque paquet reçu. En mesurant les changements dans

le temps d'espacement entre les deux paquets, l'expéditeur peut estimer le débit disponible sur le réseau comme suit :

$$bw = \frac{s}{t_2 - t_1} \tag{11.2}$$

où:

- bw la bande passante du goulot d'étranglement de la liaison
- s la taille du paquet
- t_1 et t_2 sont les temps d'arrivée du premier et du deuxième paquet respectivement.

La précision de cette technique repose sur plusieurs hypothèses. La plus importante est que les deux paquets doivent être mis *l'un après l'autre* en file d'attente du goulot d'étranglement du réseau, ce qui n'est pas garanti si un routeur a une file d'attente de type non FIFO (*First In First Out*) ou quand un autre paquet, venant d'un autre flux, est inséré entre les deux paquets. D'autres variantes de la technique de la paire de paquets ont été mis au point [CC96, Kes95, DRM04] pour atténuer cet effet sans toutefois le faire disparaître.

La technique du train de paquets

Une autre technique connue est celle appelée « le train de paquets » (*packet train*) [JR86]. Une rafale de paquets est envoyée entre la source et la destination. Seuls les paquets appartenant au même train sont utilisés dans la mesure. Un train est formé des paquets tels que l'espacement entre deux paquets consécutifs reçus ne dépasse pas un certain écart entre les paquets (inter-train). Comme dans la technique la paire de paquets, les temps entre l'arrivée des paquets du même train sont utilisés pour estimer le débit disponible.

Positionnement

Toutes ces méthodes ont plusieurs inconvénients :

1. L'envoi/la réception de paquets supplémentaires de données est nécessaire pour estimer le débit disponible. Si au contraire les paquets de données vidéo eux-mêmes sont utilisés pour cela, le récepteur doit les distinguer des autres données, par exemple en ajoutant un nouveau champ dans l'en-tête du paquet.
2. Les paquets utilisés pour l'estimation doivent être spécifiquement dispersés, c'est-à-dire qu'ils ont besoin d'un timing précis quand ils sont envoyés. Cela rend l'utilisation des paquets de données vidéo eux-mêmes, comme paquets d'estimation, difficiles.
3. Si une nouvelle connexion est utilisée pour l'estimation, l'émetteur et le récepteur doivent envoyer et écouter sur d'autres sockets (ports).
4. Non seulement l'application émettrice, mais aussi l'application réceptrice doit être modifiée pour être en mesure de répondre à ces paquets de sondage. Le changement des deux extrémités d'une connexion est connu pour être très difficile à déployer dans la réalité.

Enfin, et le plus important, ces techniques ont un objectif différent du nôtre. Comme ils ne prennent pas en compte les bitrates utilisés dans le passé, elles ne sont pas appropriées pour résoudre notre problème ZQS.

11.2.3 Positionnement par rapport aux mécanismes de contrôle de congestion

Notre problème est similaire au problème de contrôle de congestion classique du réseau, mais pas identique. Nous considérons à la fois les mécanismes basés sur une fenêtre de congestion, écrits comme TCP dans la suite, et ceux basés sur une équation du taux de transfert, écrit comme TFRC dans la suite.

Plus précisément, à la fois TCP/TFRC et ZAAL tentent de résoudre un problème d'optimisation multi-critère, mais les critères concernés pour atteindre cet objectif ne sont pas vraiment identiques, comme expliqué dans la suite.

Le zigzag

ZAAL vise à éviter le zigzag entre deux qualités consécutives, ce qui se produit lorsque (1) la qualité est inférieure au débit disponible, donc pas de perte, donc la qualité est augmentée, *et* (2) quand la qualité supérieure est plus grande que le débit disponible, donc des pertes, donc la qualité est ré-réduite à la qualité inférieure. TCP/TFRC ont un taux d'envoi à petit grains ; puisque TCP est un protocole de transport de *données*, il ne fait pas attention au zigzag, tandis que TFRC lisse le débit d'émission, sans pour autant éviter le zigzag.

Les pertes

TCP/TFRC visent à envoyer au débit maximal possible, même si cela donne un peu de pertes. Au contraire, ZAAL vise à éviter les pertes autant que possible. En réalité, les pertes ne peuvent pas être complètement évitées, sauf si un débit très faible est utilisé. Donc, pour ZAAL il est préférable de maintenir une qualité sans ou avec peu de pertes, au lieu de la qualité supérieure suivante avec un taux élevé de pertes. A titre d'exemple, lors d'une transmission vidéo, ZAAL préfère envoyer à 2Mb/s sans pertes que d'envoyer à 3Mb/s avec 20% de pertes, sachant que 3Mb/s a 50% de plus de paquets envoyés.

Le taux d'envoi

TCP/TFRC visent à envoyer au plus haut débit possible. C'est pourquoi ils *ne cessent d'augmenter* le taux d'envoi. VAAL a le même objectif mais il n'augmente pas la qualité si ZAAL estime que la qualité supérieure suivante ne devrait pas être utilisée à ce moment-là. Ainsi VAAL peut être considérée comme une proposition d'augmentation/diminution de la qualité et ZAAL comme uniquement un bloquant d'augmentation de la qualité.

Cette comparaison montre que des algorithmes classiques de contrôle de congestion ont des objectifs différents, donc ils ne sont pas appropriées pour résoudre le problème de zigzag ZQS.

11.3 Présentation de l'algorithme d'évitement de zigzag ZAAL

Pour résoudre le problème de zigzag causé par une méthode d'adaptation vidéo, un algorithme spécifique supplémentaire peut être utilisé, comme ZAAL. L'algorithme de ZAAL évite l'utilisation constante des bitrates supérieurs au débit disponible. Pour atteindre cet objectif,

ZAAL maintient une valeur moyenne pour chaque bitrate, appelée « la capacité de réussite » (*successfulness*) par la suite. Lorsque l'algorithme d'adaptation décide d'augmenter le bitrate et seulement dans ce cas, ZAAL vérifie si sa capacité de réussite est supérieure à un seuil prédéfini appelé β ; si tel est le cas, l'application est autorisée à choisir un bitrate supérieur. Autrement dit, **l'application bascule vers une qualité meilleure disponible i si et seulement si la capacité de réussite S_i de ce bitrate est supérieure à β**. Après ce processus, la valeur de la capacité de réussite est mise à jour.

À noter que ZAAL n'est pas une méthode d'adaptation. Elle est utilisée seulement pour empêcher une méthode d'adaptation de basculer constamment la qualité vers un bitrate supérieur au débit disponible. La seule information dont ZAAL a besoin est un *feed-back* obtenu de la méthode d'adaptation employée. Ce *feed-back* est une réponse à la question suivante : est-ce que l'utilisation du bitrate actuel était un succès ou non ?

11.3.1 Algorithme

L'algorithme de ZAAL utilise la valeur de la *capacité de réussite* à chaque fois qu'une période d'adaptation se termine. Après avoir été utilisé par l'application, cette valeur est mise à jour séparément pour chaque bitrate i disponible par une formule avec des poids différents ; elle est notée par S_i. Cette valeur indique si le bitrate i peut être utilisé pour la prochaine période ou non. Ainsi, cette valeur reflète le résultat des dernières tentatives de l'application d'utiliser le bitrate correspondant i. En bref, quand un bitrate ne cause pas de rejet de paquets de la mémoire tampon du protocole de transport, la valeur exprimant sa capacité de réussite est considérablement augmentée. En revanche, lorsque i cause des paquets en échec, la valeur de sa capacité de réussite est considérablement réduite. Et finalement, quand il n'y a pas de changements de qualité pendant une longue période, les valeurs de la capacité de réussite correspondant à des bitrates plus élevés sont un peu augmentées. **En règle générale, plus la valeur de la capacité de réussite est petite, plus le bitrate correspondant peut causer des paquets en échec et l'application doit éviter de le choisir pour la prochaine période d'adaptation.**

La capacité de réussite S_i, où i est l'indice du bitrate actuel, change en fonction de son passé récent. Elle est calculée en utilisant un algorithme EWMA (*Exponential Weighted Moving Average*). L'utilisation de EWMA permet de donner plus de poids à la valeur la plus récente de l'historique par rapport aux anciennes valeurs, puisque de toute évidence le débit disponible actuel est mieux exprimée par l'usage des derniers bitrates que par les plus anciens. En outre, des poids différents sont utilisés, en fonction du bitrate concerné.

Au début d'une transmission vidéo, tous les valeurs S_i sont mises à 1. Ensuite, elles sont mises à jour à chaque fois que la méthode d'application veut réajuster le débit de la vidéo au débit disponible. Cela se fait en utilisant la formule générale suivante :

$$S_i = (1 - \alpha/d)S_i + s(\alpha/d) \tag{11.3}$$

où :

- s est la réussite au moment de la mesure, autrement dit l'observation actuelle. Elle prend une des deux valeurs 0 ou 1. La valeur 0 est utilisée lorsque le bitrate ne donne pas de bons résultats, donc sa valeur de capacité de réussite va diminuer. En revanche, la valeur 1 est utilisée lorsque le bitrate est un succès, soit parce qu'il n'a pas causé de paquets perdus,

soit parce qu'il n'a pas été utilisé récemment, donc sa valeur de capacité de réussite va augmenter.
- α est le degré de pondération de la diminution/augmentation de S_i, un facteur constant de lissage ayant une valeur réelle entre 0 et 1. Plus α est grand, plus les valeurs de capacité de réussite se déconnectent plus vite du passé, c'est-à-dire que les valeurs anciennes perdent plus rapidement leurs influences.
- d est un facteur de division qui permet d'accélérer ou de ralentir l'augmentation de la valeur S_i en fonction du bitrate concerné. Dans notre algorithme, d a trois valeurs : 1, 2 et 4. Pour une $s = 1$, plus d est grand, plus la hausse de S_i est importante. Ces valeurs sont expliquées plus tard.

La capacité de réussite de chaque bitrate est une valeur réelle arithmétique entre 0 et 1. Elle peut changer dans trois cas :

1. Tout d'abord, lorsque l'application bascule vers un bitrate k supérieur à celui actuellement utilisé j. Ceci apparaît lorsque l'application ne détecte pas de paquets perdus pendant un certain temps avec la qualité actuelle de la vidéo. Dans ce cas la capacité de réussite devrait augmenter ($s = 1$) pour tous les bitrates i inférieur ou égal au débit actuel j. La vitesse d'augmentation doit être élevée, ce qui se traduit par le facteur de division le plus petit possible, $d = 1$. Cela donne l'équation suivante :

$$S_{i|i\leq j} = (1 - \alpha)S_i + \alpha \tag{11.4}$$

2. Ensuite, lorsque l'application maintient la qualité. Cela apparaît soit lorsque des pertes de paquets se produisent mais que leur taux est acceptable pour laisser l'application maintenir la qualité actuelle j, soit lorsque la capacité de réussite (S_k) du bitrate supérieur suivant k est faible ($S_k < \beta$) et puis l'application devrait éviter de s'en servir. En conséquence, la valeur de la capacité de réussite augmente pour toutes les qualités mais avec différentes valeurs de facteur de division (des vitesses différentes). Ainsi :
   - $d = 1$ and $s = 1$ (grande augmentation) pour tous les bitrates i qui sont inférieurs au bitrate courant j :
   $$S_{i|i<j} = (1 - \alpha)S_i + \alpha \tag{11.5}$$
   - $d = 2$ and $s = 1$ (légère augmentation) pour le bitrate courant j :
   $$S_j = (1 - \alpha/2)S_j + \alpha/2 \tag{11.6}$$
   - $d = 4$ and $s = 1$ (très légère augmentation) pour le bitrate $k = j+1$ supérieur au bitrate courant j :
   $$S_{k(k=j+1)} = (1 - \alpha/4)S_k + \alpha/4 \tag{11.7}$$

3. En troisième lieu, l'application diminue le bitrate de la vidéo (la phase de décroissance). Ce cas apparaît quand le débit disponible diminue pendant la période actuelle, ou lorsque l'application essaie un bitrate supérieur au débit disponible. Dans ce cas, la valeur de la capacité de réussite doit être réduite pour le bitrate actuel j et ne doit pas changer pour les autres. Par conséquent, $s = 0$ et $d = 1$ (très grande diminution). Cela se traduit par l'équation :
$$S_j = (1 - \alpha)S_j \tag{11.8}$$

Dans nos expérimentations, nous mettons intuitivement $\alpha = 0,3$ et $\beta = 1-\alpha = 0,7$. D'autres valeurs ont été analysés, or soit elles conduisent à un grand nombre d'itérations de l'algorithme d'adaptation et un temps de convergence très long, soit elles minimisent les effets car le zigzag apparaît plus souvent. Déterminer les meilleures valeurs de α et β et comment elles affectent la stabilité et l'adaptabilité de ZAAL est encore un travail à venir.

11.3.2 Discussion

Nous présentons dans cette section certaines caractéristiques de l'algorithme d'évitement de zigzag ZAAL. Pour la simplicité, nous prenons les valeurs $\alpha = 0,3$ et $\beta = 0,7$ en considération dans la discussion :

1. Au début, les valeurs S_i, qui sont mises à 1, sont supérieures à $\beta = 0,7$, de sorte qu'il n'y a pas de restriction sur l'utilisation de n'importe quel bitrate, c'est-à-dire que l'application est autorisée à utiliser tous les bitrates disponibles.
2. En analysant la phase de décroissance de l'algorithme (équation 11.8), nous pouvons noter :
   - Comme mentionné précédemment, lorsque le bitrate est supérieur au débit disponible, l'application évite de s'en servir pendant un certain temps en diminuant sa valeur de capacité de réussite. En se basant sur cette équation, lorsque cela se produit pour la première fois, elle donne $S_i = 1 - 0.3 * 1 = 0.7$ ce qui n'est pas supérieur à β. **Ainsi, un bitrate qui provoque des paquets en échec, ne peut pas être utilisé à nouveau tout de suite après.**
   - La plus petite valeur de la capacité de réussite pour un bitrate i peut apparaître quand il est utilisé avec $S_i = 0,7 + \epsilon$ et il conduit à des paquets en échec. Ainsi, la nouvelle valeur de est égale à $S_i = 0.7 * (0.7 + \epsilon) = 0.49 + \epsilon$.
   - Un bitrate plus élevé ne signifie pas nécessairement une plus petite valeur de la capacité de réussite. Lors de la réduction de la capacité de réussite pour un bitrate i, les valeurs de la capacité de réussite des bitrates supérieurs ne changent pas.
3. En étudiant la phase de croissance de ZAAL, nous pouvons noter :
   - La plus longue durée pendant laquelle ZAAL empêche l'application de choisir un bitrate supérieur k commence lorsque k a la plus petite valeur de la capacité de réussite ($S_k = 0,49 + \epsilon$, voir le point précédent). Elle se termine lorsque S_k devient $> 0,7$. En utilisant l'équation 11.7, cela correspond à 7 tentatives consécutives échouées de la part de l'application d'aller vers une qualité supérieure k, suivie d'une huitième tentative réussite. La figure 11.2 montre l'évolution de cette valeur : il y a 7 tentatives échouées entre $0,49 + \epsilon$ et une tentative avec un capacité de réussite supérieure à $\beta = 0,7$.
   - Des valeurs de la capacité de réussite entre 0,7 et 1 ne sont possibles que lorsque le bitrate ne cause pas de paquets en échec. Ceci arrive dans deux cas : la capacité de réussite augmente lentement au-delà de 0,7 lorsque la qualité est stable (voir l'équation 11.6), ou elle augmente rapidement au-delà de 0,7 lorsque les qualités supérieures sont autorisées (voir l'équation 11.5).
4. Enfin, **utiliser une moyenne exponentielle plutôt qu'une moyenne linéaire permet de garder un effet pour les valeurs anciennes pour plus longtemps** lors de l'augmentation des valeurs de capacité de réussite, et de donner plus de poids aux éléments récents de l'historique par rapport aux anciens.

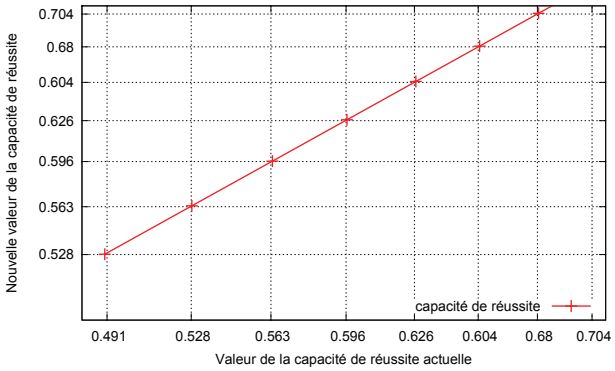


Figure 11.2 – L'augmentation de la capacité de réussite au cours de la plus grande période de temps.

11.3.3 Complexité de ZAAL

- **ZAAL est un algorithme simple et facile à mettre en œuvre**. En effet, ZAAL se compose de trois instructions conditionnelles (`if`) avec une boucle pour tous les bitrates à l'intérieur de chaque instruction conditionnelle.
- **ZAAL est un algorithme extensible** (*scalable*) ; il a une **complexité linéaire en nombre de bitrates** car il n'utilise qu'une seule boucle sur tous les bitrates. Il a également une complexité linéaire **en nombre d'utilisateurs** car il est exécuté indépendamment pour chacun d'eux. En outre, le temps d'exécution de l'algorithme de ZAAL est négligeable. En effet, pour son fonctionnement, il maintient une seule valeur par bitrate disponible. Cette valeur est calculée à la fin de chaque période d'adaptation en utilisant une formule simple, comme indiqué ci-dessus. Ainsi, **l'usage de ZAAL n'affecte pas la capacité de traitement du serveur lorsque le nombre de clients augmente**. Le faible impact supplémentaire de ZAAL peut être facilement compensé par ses autres côtés positifs, c'est-à-dire qu'en se servant de ZAAL, le serveur permet d'éviter le traitement des paquets qui seront autrement perdus sur le réseau. Nous pouvons donc ajouter que les clients desservis seront plus satisfaits de ce qu'ils reçoivent car la qualité sera meilleure.

11.4 Évaluation de ZAAL

Le réseau utilisé (figure 10.1) ainsi que les paramètres (tableau 10.1) et les conditions des expérimentations sont exactement les mêmes que ceux utilisés pour évaluer VAAL dans le chapitre précédent (voir la section 10.2 pour voir plus d'explications). À savoir, une transmission vidéo en continu a eu lieu entre un émetteur et un récepteur avec une interface câblée pour les deux. Une machine intermédiaire TCM pour contrôler le trafic est ajoutée entre la source et la destination. Le *streaming* vidéo utilise une vraie vidéo, disponible en quatre qualités : 3Mb/s, 2Mb/s, 1Mb/s et 512kb/s. La vidéo a 180s.

Deux séries de tests sont effectuées. Pour la première série (série de limitation du trafic), un seul flux est présent à tout moment au cours de la transmission vidéo ; ce flux peut donc utiliser toute le débit disponible. D'autre part, la bande passante en sortie de la machine TCM est changée trois fois afin de simuler une bande passante variable : 600kb/s pour la première

minute, 2300kb/s pour la deuxième minute, et la limitation du trafic a été interrompue pendant la dernière minute, ce qui donne un débit disponible égal à celle du réseau sans-fil d'origine. Cette série permet de voir l'effet de ZAAL dans un réseau avec un débit disponible qui change au fil du temps.

Pour la deuxième série, dix flux simultanés sont présents pendant toute la transmission. Ainsi, le débit des données envoyés par dix flux avec un maximum de bitrate (3Mb/s) dépasse largement le débit disponible (10*3Mb/s=30Mb/s). Cela permet de voir ce qui se passe lorsque multiples flux se concurrencent pour le débit disponible, en particulier pour vérifier si cela conduit à une oscillation importante des performances globales.

Cette section a pour objectif de :

1. Vérifier si ZAAL évite le problème de zigzag de la commutation de qualité (ZQS).
2. S'assurer que l'utilisation de ZAAL conduit aux mêmes résultats quantitatifs pour tous les flux, autrement dit, est-ce que les flux partagent équitablement le débit disponible ?

11.4.1 Évitement du zigzag

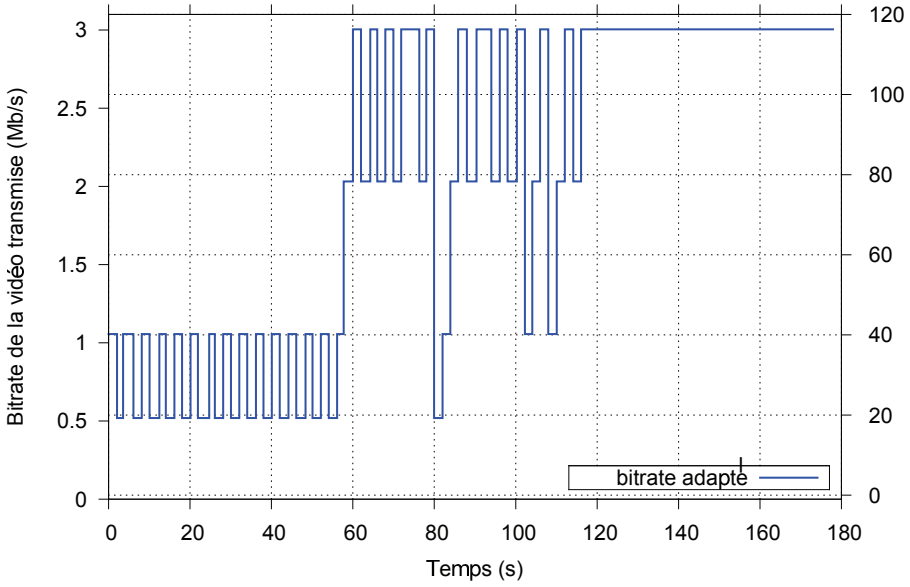
Cette section présente la variation de qualité avec et sans ZAAL. Dans les figures suivantes (11.3, 11.4), l'axe des abscisses représente le temps de 0 à 180s, la durée d'une transmission vidéo dans les expérimentations, et l'axe des ordonnées indique le bitrate de la vidéo (avec ou sans ZAAL).

Un seul flux en cas de limitation de trafic

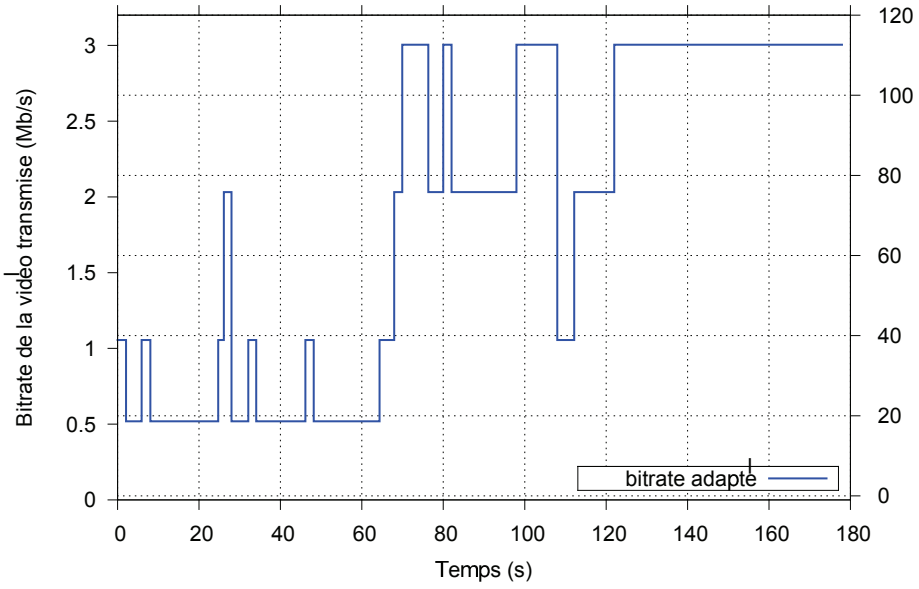
Comme prévu, ZAAL minimise l'effet de zigzag : pendant la première minute dans la figure 11.3(a), où ZAAL n'est pas utilisé, le bitrate de la vidéo est continuellement changé entre 0,5Mb/s et 1Mb/s, tandis que lorsque ZAAL est utilisé (figure 11.3(b)) l'application utilise le bitrate 0,5Mb/s la plupart du temps. Les mêmes conclusions peuvent être tirées de la transmission durant la deuxième minute. Au cours de la dernière minute, le débit disponible est assez large pour faire passer 3Mb/s, et ZAAL permet d'utiliser ce bitrate.

Une analyse plus poussée de la transmission avec ZAAL (voir la figure 11.3(b)) confirme les propriétés utiles de l'algorithme de ZAAL. Tout d'abord, l'application attend pendant au moins 2 périodes de temps avant d'essayer un bitrate qui a récemment causé des pertes. Par exemple, au début 1Mb/s cause des paquets en échec entre les secondes 0 et 2, par conséquent, il n'a pas été réutilisé à la seconde 4, mais plutôt à la seconde 6. Deuxièmement, quand un bitrate entraîne des pertes plusieurs fois de suite, l'application attend de plus en plus de temps pour s'en servir. Par exemple 1Mb/s a été essayé à la seconde 24 pour une fois, puis 8 secondes plus tard à la seconde 32 pour une deuxième fois, ensuite après 14 secondes pour la troisième fois à la seconde 46, et enfin l'application a attendu 18 secondes (à la seconde 64s) pour le réutiliser sans pertes. Troisièmement, la période maximale pendant laquelle un bitrate n'était pas choisi est de 16 s, par exemple entre les secondes 8 et 24, comme indiqué dans la section 11.3.2, le troisième point. Le même exemple peut être appliqué pour les secondes 48 et 64 ; pendant ce temps il n'y avait pas de pertes car le débit disponible était supérieur au bitrate, mais ZAAL a correctement empêché l'augmentation du bitrate évitant ainsi le zigzag.

Plus précisément, le tableau 11.1 présente le nombre de zigzags au cours de la première et la deuxième minute (la troisième minute est ôtée car il n'y a pas de différence). Il est clair



(a) sans ZAAL: de nombreux zigzags



(b) avec ZAAL: peu de zigzags

Figure 11.3 – Adaptation de la qualité pour un seul flux en cas de limitation de trafic, dans les mêmes conditions.

Méthode	*Première minute*	*Deuxième minute*	*Total*
Sans ZAAL	13	10	24
Avec ZAAL	4	1	5

Tableau 11.1 – Nombre de zigzags avec et sans ZAAL.

que ZAAL conduit à beaucoup moins de zigzags : 4 contre 13 pendant la première minute et 1 contre 10 pendant la deuxième, soit environ 80 % de moins au total. Naturellement, cela a un impact très important sur la qualité vidéo perçue. Le peu de zigzags apparu avec ZAAL est le résultat de quelques tentatives d'amélioration de la qualité (adaptation) sans pour autant en abuser.

Dix flux

Dans cette expérimentation (voir figure 11.4 pour un des dix flux concurrents), il est noté que tous les flux ont la même tendance d'adaptation concernant leur choix de bitrates. Les changements de bitrate sont souvent entre 1Mb/s et 2Mb/s, moyenne correspondant à chacun des dix flux. Cela indique qu'un seul flux ne reste pas tout le temps à un bitrate élevé, par exemple 3Mb/s ; si cela arrive, il provoquerait la réduction du débit disponible pour les autres flux. En outre, comme dans le test précédent pour un seul flux, l'application adapte souvent la qualité de la vidéo tout en évitant des bitrates provoquant des paquets perdus. Par exemple, le bitrate 3Mb/s cause des paquets perdus à la seconde 26, il n'est donc pas utilisé jusqu'à la seconde 68. ZAAL évite les changements fréquents de bitrate vidéo lors de la transmission, tout en améliorant l'utilisation de la bande passante.

Ce test vérifie également l'équité entre les dix flux concurrents lors de l'utilisation de ZAAL. Selon les résultats de ce test, tous les flux ont un pourcentage presque égal de paquets envoyés (voir la figure 11.5 qui montre le pourcentage de paquets envoyés par chaque flux au niveau applicatif avec ZAAL). Cette figure montre donc que ZAAL maintient l'équité entre les flux simultanés sur le serveur. D'autre part, l'équité sur le réseau est garantie le contrôle de congestion de type *TCP-friendly* sur lequel se base l'algorithme d'adaptation de la vidéo.

11.4.2 Comparaison des performances

Même si la réduction du taux de perte de paquets n'est pas l'objectif principal de ZAAL, il est important de savoir comment il est influencé par ZAAL. C'est le nombre de paquets reçus et le nombre de paquets perdus qui sont pris en compte dans cette comparaison. Le tableau 11.2 présente les résultats numériques des expérimentations effectuées.

Pour un flux en cas de limitation de trafic, le nombre de paquets reçus est plus faible lorsque ZAAL est utilisé. En effet, il y a 40263 paquets reçus pour ZAAL contre 42043 sans ZAAL. Cela s'explique naturellement par le fait que ZAAL prive l'application d'aller vers une qualité supérieure pendant une certaine période. D'autre part, lorsque ZAAL est utilisé, le bitrate a 30% moins de paquets perdus dans les mêmes conditions réseau, ce qui compense largement le 4% de paquets reçus en moins. La moyenne des dix flux concurrentes donne des résultats encore meilleurs pour ZAAL. En effet, le nombre de paquets reçus est environ 5% plus grand pour ZAAL, et le taux de perte est environ 50% plus petit aussi.

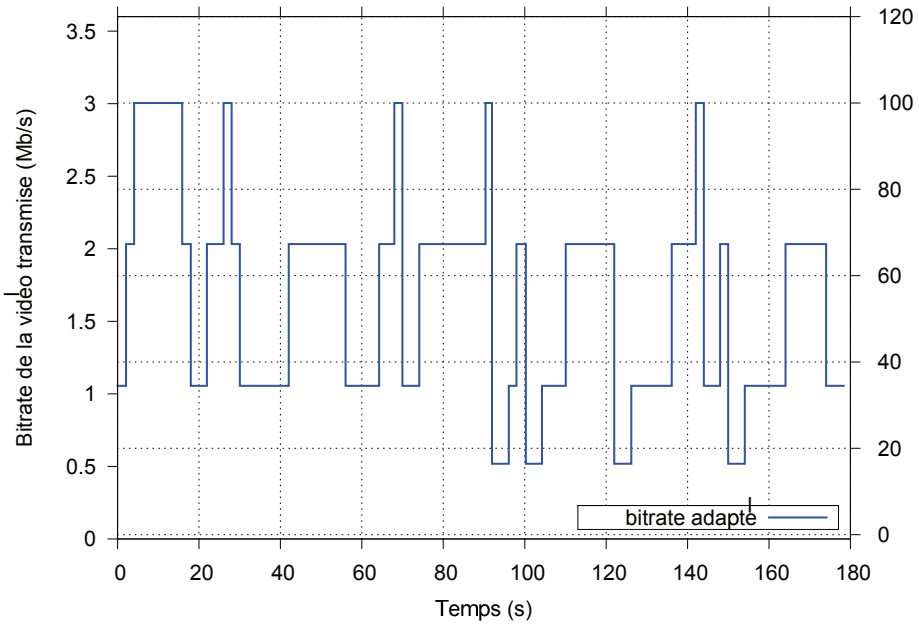


Figure 11.4 – Adaptation de la qualité avec ZAAL pour un des dix flux concurrents : peu de zigzags.

	Limitation de trafic			*10 flux concurrents*		
	Paquets			*Paquets*		
Méthode	*Envoyés*	*Reçus*	*Perdus (%)*	*Envoyés*	*Reçus*	*Perdus (%)*
Sans ZAAL	47795	42043	5752 (12%)	41191	32307	8884 (21.6%)
Avec ZAAL	43913	40263	3650 (8.4%)	38105	33889	4216 (11%)

Tableau 11.2 – Comparaison du nombre de paquets envoyés et reçus (pour tous les flux) sans et avec ZAAL.

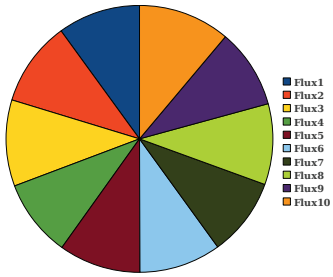


Figure 11.5 – Pourcentage de paquets envoyés par l'application de chaque flux en utilisant ZAAL : le pourcentage est à peu près le même.

Pour résumer :

- dans la première expérimentation, il est difficile de décider quelle est la meilleure solution en termes de nombre de paquets reçus et le taux de perte, mais on constate que l'utilisation de ZAAL est bénéfique car il évite le zigzag et conduit à une qualité vidéo plus stable ;
- dans la deuxième expérimentation, ZAAL est meilleur en termes de paquets envoyés et reçus, évitant le zigzag dans le même temps.

En conclusion, **ZAAL est une amélioration utile de VAAL et même si son nombre de paquets reçus est inférieur dans certains cas, il réduit le taux de paquets perdus, tout en maximisant l'utilisation de la bande passante**.

11.5 Conclusion

Dans ce chapitre a été présenté une méthode simple afin d'éviter la commutation constante et indésirable de la qualité survenant lors d'une transmission adaptative de la vidéo en continu. Cette méthode appelée ZAAL (*Zigzag Avoidance ALgorithm*) représente une solution générale, car elle peut être intégrée à n'importe quelle méthode d'adaptation de la vidéo, et peu importe si la vidéo est encodée en multiples qualités ou en multi-couches. Cette solution utilise un historique d'utilisation des bitrates pour calculer une valeur indiquant leurs capacités de réussite, à utiliser quand l'application souhaite les choisir à un moment donné. C'est une méthode non intrusive, c'est-à-dire qu'elle ne modifie pas la manière dont la transmission vidéo est effectuée et n'a pas besoin de *feed-back* du réseau car elle se contente des retours applications. Les expérimentations confirment que, grâce à cette solution, le problème de zigzag de la commutation de qualité n'apparaît que très rarement, sans grande influence sur le débit vidéo.

Quatrième partie

Conclusion générale et perspectives

Chapitre 12

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les services de diffusion et les services en mobilité ont récemment commencé à converger vers une plus grande liberté de mouvement, et cela s'étendra à d'autres services. Les développements dans le domaine du multimédia impliquent que divers contenus et services seront offerts sur différents réseaux avec différentes contraintes. Les utilisateurs s'attendent à consommer ces services en fonction de la disponibilité et l'accessibilité du réseau. Des services dynamiques et efficaces doivent être donc développés afin de répondre à ces exigences.

Cette thèse a porté sur l'amélioration du transfert de données, d'une part sur les réseaux sans-fils et d'autre part pour des données continues telles que la vidéo. Des simulations sous ns2 et des expérimentations réelles ont été effectuées pour valider nos propositions.

Les contributions de cette thèse ainsi que leur validations sont synthétisées par la suite.

Synthèse

➤ Dans le chapitre 5, nous nous sommes intéressés à l'amélioration du mécanisme de contrôle de congestion afin qu'il soit plus efficace lorsqu'il est utilisé sur un réseau sans-fil. La solution proposée, RELD (*RTT ECN Loss Differentiation*), donne à ce mécanisme le moyen de différencier les pertes de congestion des pertes sans-fil, et par conséquent de mieux traiter les pertes dues aux erreurs du canal sans-fil. En effet, lorsque RELD est utilisé, la source d'une connexion ne diminue son taux d'envoi qu'en réaction aux pertes de congestion. En revanche, il continue à envoyer au même débit au cas où RELD indique que les pertes sont dues aux erreurs sur le sans-fil.

Pour différencier ces pertes sur l'émetteur des données, RELD utilise à la fois ECN (Explicit Congestion Notification) et le changement sur le RTT du paquet qui suit la perte. Le fonctionnement de RELD est ainsi simple : tant que l'émetteur reçoit des accusés de réception avec des paquets marqués ECN, il en déduit que le réseau est congestionné et réduit donc son débit en conséquence. En revanche, s'il n'y a pas de marquage ECN alors que la source détecte des pertes sur le réseau, il faut vérifier s'il y a une congestion ou non avant de diviser par deux la taille de sa fenêtre de congestion. Pour cela, RELD extrait le RTT et le compare à un seuil qui prend

en compte la moyenne et la variation du RTT ($avg + 0.6var$). RELD déduit ainsi que les pertes sont soit dues aux erreurs du canal sans-fil si le RTT est supérieur à ce seuil et le débit doit donc être maintenu, soit dues à la congestion du réseau et le débit doit donc être réduit.

➤ Dans le chapitre 6, nous avons validé les performances de RELD par la simulation. Afin d'avoir des résultats auxquels nous pouvons avoir confiance, nous avons utilisé un modèle réaliste d'erreur sur le canal sans-fil. Ce modèle (*shadowing-pattern*), développé dans notre laboratoire, reproduit la réalité dans laquelle différents éléments de l'environnement réel fait cumuler les effets qu'ont sur la puissance du signal au niveau du récepteur. Il en résulte ainsi des pertes qui ressemblent à celles qui arrivent dans un environnement réel.

Grâce aux simulations effectuées, nous avons pu valider les points suivants :

1. le RTT augmente en cas de pertes dues aux erreurs de transmission sur le canal sans-fil ;
2. RELD différencie bien les pertes de congestion des pertes sans-fil en utilisant le seuil $avg + 0.6var$;
3. RELD atteint généralement des taux de classification correcte très élevés (entre 80% et 97% pour les pertes de congestion et d'à peu près 90% pour les pertes sans-fil) ;
4. RELD améliore d'environ 68% la performance du protocole du contrôle de congestion ;
5. l'utilisation d'ECN, qui est proposée afin d'éviter la congestion sur un réseau filaire, ne suffit pas pour distinguer les pertes sur un réseau mixte filaire et sans-fil.

➤ Le chapitre 9 a présenté VAAL *Video Adaptation at Application Layer*, une méthode originale d'adaptation de la vidéo au niveau de la couche application sur l'émetteur. C'est une méthode très simple à implémenter, puisqu'elle ne requiert qu'une modification côté émetteur au niveau de la couche application. Elle adapte en permanence le bitrate de la vidéo aux conditions du réseau, autrement dit elle fait un contrôle de congestion sur l'émetteur. La visioconférence est un cas d'application idéal. Cette méthode fonctionne au-dessus de tout protocole de transport avec contrôle de congestion (TCP, DCCP), ce qui lui confère aussi la propriété de *TCP-friendliness*.

VAAL utilise le débordement du tampon du protocole de transport pour estimer le débit disponible et adapter le bitrate de la vidéo transmise en conséquence. Toutes les 2 secondes, l'application serveur calcule le pourcentage des paquets en échec dans le tampon. Ce pourcentage est utilisé par la suite pour contrôler le bitrate de la vidéo. Plus le pourcentage des paquets en échec est élevé, plus le bitrate actuel dépasse le débit disponible ce qui signifie qu'il devrait être diminué. Un pourcentage nul indique soit un débit disponible égal au bitrate, soit un débit disponible plus grand, donc le bitrate de la vidéo envoyée peut être augmenté.

➤ Dans le chapitre 10, nous avons validé les performances de VAAL par des expérimentations. Une application de transmission de la vidéo en continu, écrite en C, a été développée du côté client et une autre du côté serveur. Cette application utilise DCCP comme protocole de transport multimédia. Les expérimentations ont été réalisées par une transmission en continu utilisant une vraie vidéo d'une durée de 180 secondes. Cette vidéo est encodée en quatre qualités, 3Mb/s, 2Mb/s, 1Mb/s et 512kb/s, avec une image clé (I) toutes les 2 secondes.

Grâce aux expérimentations effectuées, nous avons pu valider les points suivants :

1. le pourcentage de paquets en échec est un moyen efficace pour estimer le débit d'envoi, et par conséquent d'adapter le bitrate au débit disponible ;
2. l'adaptation est meilleure qu'une transmission classique sans adaptation ;
3. lorsque le débit est supérieur à la meilleure qualité disponible, VAAL est équivalent à une transmission classique de la plus haute qualité ; VAAL assure donc le meilleur choix sans l'intervention de l'utilisateur final ;
4. VAAL change de qualité de façon autonome, empêchant ainsi les pauses intempestives de la vidéo ;
5. l'adaptation faite par VAAL est dynamique et permet de réajuster rapidement le bitrate si les conditions réseau changent.

➤ Enfin, dans le chapitre 11 nous avons présenté une méthode permettant de limiter la commutation de qualité en zigzag ZQS (*zigzag quality switching*). ZQS est défini comme étant le problème rencontré par une application de *streaming* adaptatif lorsqu'elle change constamment la qualité de la vidéo entre deux valeurs.

Dans ce chapitre, nous avons proposé une solution à ce problème, que nous avons appelée ZAAL (**Z**igzag **A**voidance **AL**gorithm). ZAAL utilise un historique des bitrates afin d'extraire un indicateur pour chacun d'entre eux. Cet indicateur exprime la capacité de réussite de chacun des bitrates disponibles si la méthode d'adaptation décide de s'en servir. Ainsi, cette valeur est utilisée pour décider si une qualité supérieure peut être choisie ou non, empêchant ainsi le zigzag.

L'algorithme de ZAAL est simple et facile à implémenter. C'est une méthode non intrusive, c'est-à-dire qu'elle ne modifie pas la manière dont la transmission vidéo est effectuée et n'a pas besoin de *feed-back* du réseau car elle se contente des retours de la couche application.

Ce chapitre a présenté aussi une intégration de l'algorithme de ZAAL avec VAAL qui se résume comme suit. VAAL débute une transmission vidéo avec un bitrate donné. Ensuite, toutes les 2 secondes, VAAL adapte la qualité de la vidéo aux conditions réseau. Si le bitrate est supérieur au débit disponible, VAAL se comporte comme d'habitude en choisissant un bitrate inférieur. ZAAL, en revanche, met à jours les valeurs de capacité de réussite des bitrates impliqués. Sinon, si VAAL trouve qu'une qualité supérieure doit être choisie, il consulte ZAAL qui décide en fonction de la capacité de réussite de la qualité supérieure. Si et seulement si cette capacité de réussite est supérieure à un seuil prédéfini β, VAAL est autorisé à augmenter le bitrate, puis ZAAL met à jour les valeurs de capacité de réussite des bitrates concernés.

La même méthodologie de validation de VAAL a été utilisée pour valider les performances de ZAAL. Les expérimentations ont validé et confirmé les points suivants :

1. le problème de commutation de qualité en zigzag n'apparaît que très rarement ;
2. ZAAL améliore la stabilité de VAAL ;
3. ZAAL réduit le taux de paquets perdus, tout en maximisant l'utilisation de la bande passante ;

Enfin, ZAAL peut être intégrée à n'importe quelle méthode d'adaptation de la vidéo, et peu importe si la vidéo est encodée en multiples qualités ou en multi-couches.

Bilan

Les méthodes proposées dans cette thèse ont répondu aux objectifs principaux que nous avions fixés, à savoir :

- optimiser la transmission de données sur les réseaux sans-fil ;
- acheminer efficacement et d'une manière adaptative des données vidéo sur différents types de réseaux ;
- améliorer la qualité de service en utilisant les informations disponibles sur plusieurs couches avec un minimum de changement sur l'architecture actuelle ;
- supporter la transmission en continu des contenus multimédia directs ;
- prendre en considération les caractéristiques des données multimédia telles que le temps réel (un minimum de délai) ainsi que les formats existants ;
- employer des algorithmes simples et faciles à implémenter ;
- être les plus générales possible.

Nos travaux ont abouti à cinq publications dont une dans un journal international [1], trois articles dans des conférences internationales [2, 3, 4] et une conférence internationale [5], toutes avec comité de sélection. Un de ces article [3] a également reçu le prix A'Doc [6] décerné par un jury pluridisciplinaire de l'Université de Franche-Comté au meilleur article de recherche fait par un doctorant de cette Université.

Perspectives

Les perspectives de cette thèse se divisent en deux catégories : court et long terme. Les perspectives à court terme se caractérisent par trois points :

- Rendre dynamique la durée de la période utilisée par l'algorithme d'adaptation. Dans l'implémentation actuelle de VAAL, la période d'adaptation est de 2 secondes. Elle a été choisie pour correspondre aux fréquences de temps des images clés dans la vidéo MPEG utilisée. La prochaine version de VAAL devrait faire de sorte que cette valeur soit extraite dynamiquement de la vidéo pour ajuster l'algorithme d'adaptation à cette valeur.

- En corollaire du point précédent, étudier plus de formats vidéo, notamment ceux qui n'utilisent pas la notion des images clés, afin de préciser les moments de début et de fin de chaque période, et donc la durée d'une période d'adaptation.

- Trouver les meilleures valeurs de α et de β pour l'algorithme de ZAAL. Dans les expérimentations, $\alpha = 0,3$ et $\beta = 0,7$ ont été choisies intuitivement. Une prochaine étude sera de trouver des valeurs qui permettent un petit nombre d'itérations de l'algorithme d'adaptation et un petit temps de convergence vers la meilleure qualité de la vidéo. L'étude devrait montrer également comment les valeurs de α et de β affectent la stabilité et l'adaptabilité de l'algorithme.

En ce qui concerne les perspectives à long terme, elles reposent sur quatre points. D'abord, pour la méthode de différenciation de pertes :

- Implémenter l'algorithme de différenciation de pertes RELD dans le code DCCP de Linux même, puis évaluer ses performances dans un environnement sans-fil réel mais également sur un réseau purement filaire. Cela donnera la possibilité de vérifier si RELD peut co-exister avec d'autre trafic n'utilisant pas la différenciation et partageant les mêmes ressources réseau. Cela permettra également de tester des transmissions réelles de la vidéo en continu sur un réseau sans-fil lorsque les performances du protocole du transport sont optimisées pour ce réseau.

Ensuite, pour la méthode d'adaptation de la vidéo :

- Créer une solution hybride entre ZAAL et les techniques d'estimation de la bande passante pour, d'une part, limiter encore le nombre de zigzags et, d'autre part, diminuer encore le taux de pertes dû aux tentatives ratées.

  Par exemple, au moment où la capacité de réussite d'un bitrate est assez élevée pour que ZAAL autorise l'application à augmenter le bitrate, l'application vérifie d'abord si le débit disponible à ce moment précis est suffisant pour le bitrate supérieur. Si c'est le cas, l'application prend la décision d'augmenter la qualité. Sinon, elle maintient encore la qualité actuelle. Ainsi, ZAAL servirait à éviter le zigzag et, lorsque la période maximale de non-augmentation imposée par ZAAL est atteinte, l'estimation de la bande passante empêcherait l'application de tenter inutilement un bitrate potentiellement supérieur au débit disponible.

  Toutefois, un travail très intéressant à venir est de concevoir un algorithme avec des contraintes VAAL/ZAAL, mais qui soit plus général, et applicable à toute la classe des méthodes de contrôle de congestion de réseau.

- Créer une plateforme de streaming HTTP en utilisant comme protocole de transport DCCP. À l'état actuel nous disposons des applications de transfert serveur et client. Un travail à venir est de faire de sorte que le client puisse choisir la vidéo avec le format d'encodage désiré. L'application s'occuperait ensuite de la transmission adaptative de la vidéo. Une intégration de l'application dans un lecteur Flash est envisageable mais pas exclusive.

- Étendre les travaux actuels sur des réseaux multi-radio. De plus en plus d'équipements sont connectés à plusieurs technologies réseau (multi-réseau), notamment radio. Par exemple, il est courant qu'un téléphone portable ait des connexions Bluetooth, GPRS, 3G et Wi-Fi. Nous nous plaçons dans le cas où plusieurs utilisateurs avec des appareils multi-réseau s'échangent des données vidéo en streaming.

  Le point principal est le choix dynamique du réseau : plusieurs paramètres entrent en compte lors du choix du réseau auquel l'appareil est connecté, par exemple le débit, le coût, le taux d'erreurs et la consommation d'énergie. Certains paramètres sont dynamiques, par exemple le débit dépend de la distance jusqu'à la borne ou du nombre d'utilisateurs connectés.

  Notre but final est d'augmenter la qualité des vidéos transmises en continu sur Internet, tout en laissant aux machines et au réseau le soin de s'occuper du paramétrage des caractéristiques intrinsèques de la vidéo.

* * *

BIBLIOGRAPHIE PERSONNELLE ET PRIX

Revues internationales avec comité de sélection

[1] Wassim Ramadan, Eugen Dedu, Dominique Dhoutaut et Julien Bourgeois. RELD, RTT ECN Loss Differentiation to optimize the performance of transport protocols on wireless networks. *Telecommunications Systems, special issue on Mobile Computing and Networking Technologies*, 2011. To appear.

Conférences internationales avec comité de sélection

[2] Wassim Ramadan, Eugen Dedu et Julien Bourgeois. EcnLD, ECN loss differentiation to optimize the performance of transport protocols on wireless networks. Dans *International Conference on Ultra Modern Telecommunications and Workshops (ICUMT), WMCNT workshop*, 1, p. 1–6. IEEE, Saint Petersburg, Russia, octobre 2009.

[3] Wassim Ramadan, Eugen Dedu et Julien Bourgeois. VAAL, video adaptation at application layer and experiments using DCCP. Dans Marcelo S. Alencar et Valdemar C. da Rocha, Jr. (éds.), *WPMC, 13-th Int. Symposium on Wireless Personal Multimedia Communications*, p. 1–5. Springer, Recife, Brazil, octobre 2010. Proceedings on CD-ROM.

[4] Wassim Ramadan, Eugen Dedu et Julien Bourgeois. Avoiding zigzag quality switching in real content adaptive video streaming. Dans *DICTAP, International Conference on Digital Information and Communication Technology and its Applications*, CCIS, Communications in Computer and Information Science. Springer, Dijon, France, juin 2011.

Conférences nationales avec comité de sélection

[5] Wassim Ramadan, Eugen Dedu et Julien Bourgeois. Une méthode de différenciation de pertes pour améliorer la performance des protocoles de transport sur réseaux sans-fil. Dans Alexandre Caminada (éd.), *JDIR, 10èmes Journées doctorales en informatique et réseaux*, p. 128–133. Belfort, France, février 2009.

Prix

[6] Wassim Ramadan. Améliorer la qualité de la vidéo sur internet, Juin 2011. Prix A'Doc - Association des Jeunes Chercheurs de Franche-Comté, décerné par un jury pluridisciplinaire de l'Université de Franche-Comté. Publication dans la presse universitaire de l'Université de Franche-Comté.

* * *

GLOSSAIRE

3D three-dimensional video (3D), une vidéo à trois dimensions ou en relief. Elle donne la possibilité d'apprécier des images en trois dimensions par l'intermédiaire de techniques permettant au cerveau humain de simuler une perception de profondeur.

3G Third Generation (3G), la troisième génération (3G) désigne une norme de technologie de téléphonie mobile apparue au début des années 2000, permettant un accès à Internet, appels vidéo et TV dans un environnement mobile.

802.11 IEEE 802.11 est un ensemble de normes concernant les réseaux sans fil.

ACK ACKnowledgement (ou acquittement) est l'accusé envoyé par une destination spécifique à une source afin de l'informer de la réception ou non des paquets envoyés. Il est utilisé généralement par les protocoles de transport dans leur mécanisme de contrôle de congestion.

AIMD Additive increase/multiplicative decrease (AIMD). C'est un algorithme d'évitement de congestion utilisé par TCP. Il alterne des phases de croissance linéaire du taux de transmission et de réduction multiplicative en cas de congestion.

AP Access Point, un point d'accès est un dispositif qui permet aux périphériques sans-fil de se connecter entre eux ou à un réseau filaire en utilisant le Wi-Fi, Bluetooth ou des technologies similaires.

ARQ Automatic Repeat reQuest (ARQ), requête automatique de répétition. C'est une méthode de contrôle d'erreur pour la transmission de données. Elle utilise des acquittements et des timeouts pour parvenir à une transmission efficace de l'information.

bitrate le débit de la vidéo exprimé en kbps ou kilo bits par seconde. Plus le bitrate a une valeur importante, plus la qualité de la vidéo est bonne mais également plus grande est sa taille.

C++ C'est un langage de programmation permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique.

CA Congestion Avoidance, la phase d'évitement de congestion dans le mécanisme de contrôle de congestion (CC).

CC Congestion Control, le contrôle de congestion est un mécanisme qui permet d'empêcher la source d'envoyer des paquets à un débit supérieur à la capacité du réseau tout en gardant ce débit au maximum possible.

CPU Central Processing Unit (CPU), unité centrale de traitement, elle constitue le composant essentiel d'un ordinateur, son rôle est d'interpréter les instructions et de traiter les données d'un programme.

DCCP Datagram Congestion Control Protocol (DCCP) est un protocole de communication orienté paquet (classé dans la couche de transport (4) du modèle OSI). Il a été normalisé par l'IETF dans le RFC 4340. Il a actuellement trois contrôles de congestion, TCPlike, TFRC et TFRC-lite, et utilise ECN par défaut.

DupACK Duplicate Acknowledgement, un accusé de réception dupliqué. C'est un acquittement identique au précédent, qui a pour but d'informer la source de la réception d'un segment en désordre. Après avoir reçu 3 DupACKs, TCP considère que le premier paquet non acquitté est perdu sur le réseau.

ECN Explicit Congestion Notification (ECN) est une extension du protocole de transport qui permet la notification de la congestion du réseau préalable à la perte de paquets.

EWMA Exponential Weighted Moving Average (EWMA), la moyenne mobile exponentielle et pondérée. C'est un type de moyenne statistique utilisée pour analyser des séries ordonnées de données, le plus souvent des séries temporelles. Elle est mobile parce qu'elle est recalculée de façon continue, en utilisant à chaque calcul un sous-ensemble d'éléments dans lequel un nouvel élément remplace le plus ancien ou s'ajoute au sous-ensemble. La moyenne mobile exponentielle utilise une pondération des termes qui décroît exponentiellement. Le poids de chaque valeur participant à la moyenne est un facteur plus grand que la valeur qui le précède dans la série, ce qui donne plus d'importance aux observations plus récentes, sans toutefois supprimer complètement l'effet des valeurs plus anciennes.

FEC Forward Error Correction, code correcteur d'erreur.

FIFO First In First Out, une méthode d'organisation des données en vue de leur traitement : premier arrivé, premier servi.

Flash Il se réfère à Adobe Flash Player et à une plateforme utilisée pour créer et ajouter des contenus multimédia, tels que les animations, les jeux ou des vidéos, aux pages web.

FPS Frames Per Second (FPS), le nombre d'images par seconde d'une vidéo. Plus le nombre d'images est élevé, plus la vidéo est fluide et agréable à regarder.

FreeBSD FreeBSD, un système d'exploitation.

GNU GNU est une distribution de logiciels composé exclusivement de logiciels libres. Son nom est un acronyme récursif de « Gnu's Not Unix » qui signifie GNU n'est pas UNIX.

GOP Group Of Pictures (GOP), groupe d'images. Un GOP est constitué d'une suite d'images qui sont indépendantes d'autres images de la vidéo. Les images visibles sont générées à partir des images codées contenues dans un GOP.

H.264 H.264, ou MPEG-4 AVC (Advanced Video Coding), ou MPEG-4 Part 10, est une norme de codage vidéo développée conjointement par l'ITU-T et l'ISO/CEI Moving Picture Experts Group (MPEG). La norme H.264 comprend de nombreuses techniques nouvelles

qui lui permettent de compresser beaucoup plus efficacement les vidéos que les normes précédentes et fournit plus de flexibilité aux applications dans un grand nombre d'environnements réseau.

HD High Definition video, une vidéo en haute définition. Elle se réfère à toute vidéo d'une résolution supérieure à celle ayant une définition standard (SD, Standard Definition). Généralement, cela nécessite des résolutions d'affichage de 1280x720 pixels (720p) ou 1920x1080 pixels (1080i/1080p).

HDD Hard Disk Drive, le disque dur est un support magnétique intégré dans un ordinateur permettant le stockage de données.

HTTP HyperText Transfer Protocol, protocole de transfert hypertexte. C'est un protocole de communication client-serveur développé pour le World Wide Web (WWW). Il est classé au niveau de la couche application (7) du modèle OSI.

IAT Inter-Arrival Time, le temps qui sépare la réception de deux paquets.

IEEE Institute of Electrical and Electronics Engineers (l'« Institut des ingénieurs électriciens et électroniciens ») est une organisation à but non lucratif dédiée à faire avancer l'innovation technologique, analogue de l'ITU-T.

IETF Internet Engineering Task Force, un groupe informel, international, ouvert à tout individu, qui participe à l'élaboration de standards pour Internet.

IP Internet Protocol, le protocole de communication fondamental de la suite des protocoles Internet. C'est un protocole de communication de réseau informatique, classé dans la couche réseau (3) du modèle OSI.

ISO International Standardization Organization, un organisme de normalisation international composé de représentants d'organisations nationales de normalisation de 158 pays.

ITU-T International Telecommunications Union-Telecommunications standardization, la plus ancienne organisation internationale technique de coordination entre les États et le secteur privé. Elle est rattachée directement aux Nations Unies depuis 1947. Elle est chargée de la réglementation et de la planification des télécommunications dans le monde.

MPEG Moving Pictures Experts Group, groupe d'experts de l'ISO chargé du développement de normes internationales pour la compression, la décompression et le traitement de la vidéo, de l'audio et de leur combinaison, de façon à satisfaire une large gamme d'applications.

MPEG-21 Une norme internationale développée par MPEG dont l'objectif est de spécifier une architecture permettant l'interopérabilité et l'utilisation transparente des représentations audiovisuelles numériques.

ns2 Network Simulator, un simulateur de réseau open source permettant l'évaluation et la mise au point de nouveaux protocoles.

OS Operating System, système d'exploitation.

OSI Open System Interconnection (OSI), modèle d'interconnexion des systèmes en réseau, norme proposée par l'ISO pour les communications entre ordinateurs. Il y a 7 couches : 1- la couche physique ; 2- la couche liaison de données ; 3- la couche réseau ; 4- la couche transport ; 5- la couche session ; 6- la couche présentation ; 7- la couche application.

PC Personal Computer (PC), ordinateur personnel.

RFC Request for Comments (RFC), littéralement « demande de commentaires », sont une série numérotée de documents officiels décrivant les aspects techniques d'Internet, ou de différents matériels informatiques.

ROTT Relative One-Way Trip Time (ROTT), le délai relatif pour la réception d'un paquet dès son départ de chez l'émetteur.

RTP Real-Time Transport Protocol, protocole pour le transfert temps réel (au niveau de la couche application (7) du modèle OSI).

RTT Round Trip Time (RTT) est le temps nécessaire pour un signal ou un paquet d'aller d'une source spécifique à une destination spécifique et de recevoir son acquittement.

SACK Selective Acknowledgment (SACK), l'acquittement sélectif sous la forme d'une option qu'on peut ajouter au protocole de transport (e.g. TCP), c'est une forme d'acquittement où le récepteur liste explicitement les paquets reçus (négativement ou positivement).

SCTP Stream Control Transmission Protocol (SCTP) est un protocole défini en 2000 par l'IETF dans la RFC 4960. Il offre certaines des caractéristiques des services fournies à la fois par UDP et par TCP : il est orienté message comme UDP et assure un transport fiable et en séquence des messages avec contrôle de congestion comme TCP.

socket en français « prise », signifie un canal de transmission. Le socket permet à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau.

SQRT le racine carré. C'est un type de contrôle de congestion similaire à AIMD sauf que la réduction de la fenêtre de congestion w est proportionnelle à $\sqrt{w}$ et pas multiplicative.

SVC Scalable Video Coding (SVC), le codage vidéo évolutif, est le nom donné à une norme de compression vidéo développée conjointement par UIT-T et l'ISO. L'objectif de SVC est d'offrir un contenu qui peut être encodé une fois et offrir ensuite différents débits avec différentes qualités.

TC Traffic Shaping, TC est un programme (open source) en ligne de commande conçu pour contrôler le trafic entrant et sortant des machines connectées au réseau.

TCP Transmission Control Protocol, protocole de contrôle de transmission. C'est un protocole de transport fiable sur un réseau. Il est situé au niveau de la couche transport (4) du modèle OSI.

TCPlike Un contrôle de congestion semblable à celui de TCP. C'est aussi un des contrôles de congestion de DCCP. Il est approprié pour les flux DCCP qui aimeraient recevoir autant de bande passante que possible sur le long terme.

TFRC TCP-Friendly Rate Control (TFRC) est un mécanisme de contrôle de congestion dont l'objectif est de faire une concurrence loyale avec le trafic TCP sur des périodes moyennes, et d'être beaucoup moins variable que TCP sur des délais courts. Il fait partie des contrôles de congestion de DCCP.

Theora Theora est un format de compression vidéo libre et gratuit de la Fondation Xiph.org. Il peut être utilisé pour distribuer des films et des vidéos en ligne et sur disque sans les droits de licence et de redevances d'un fournisseur associé à d'autres formats.

UDP User Datagram Protocol, protocole de datagramme utilisateur. C'est l'un des principaux protocoles de transport utilisés sur Internet. Il fait partie de la couche transport (4) du modèle OSI.

VLC Video LAN Client, lecteur multimédia dont le développement très populaire.

VoD Video on Demand, vidéo à la demande.

Vorbis Vorbis, un algorithme de compression et de décompression audio numérique ouvert et libre.

VP6 C'est un codec vidéo propriétaire développé par On2 Technologies en 2003.

VP8 VP8, un codec vidéo de la société On2, acheté par Google, qui en a fait un format ouvert dans le cadre du projet WebM.

W3C World Wide Web Consortium (W3C), un organisme de normalisation chargé de promouvoir la compatibilité des technologies du WWW.

WebM WebM est un format multimédia ouvert principalement destiné à un usage sur le Web. Il regroupe des flux vidéos encodés en VP8 et des flux audios encodés en Vorbis.

Wi-Fi Wi-Fi est un ensemble de protocoles de communication sans-fil régis par les normes du groupe IEEE 802.11. Un réseau Wi-Fi permet de relier sans fil plusieurs appareils informatiques (ordinateur, routeur, décodeur Internet, etc.) au sein d'un réseau informatique.

WWW World Wide Web (WWW), communément appelé le Web, signifie littéralement la toile (d'araignée) mondiale ou la Toile. C'est un système hypertexte public fonctionnant sur Internet et qui permet de consulter, avec un navigateur, des pages mises en ligne dans des sites. L'image de la toile vient des hyperliens qui lient les pages web entre elles.

XML eXtensible Markup Language (XML), un langage de balisage extensible. C'est un standard du W3C permettant l'échange de contenus entre systèmes hétérogènes.

BIBLIOGRAPHIE

[BB95] Ajay Bakre et B.R. Badrinath : I-TCP, indirect TCP for mobile hosts. *In 15th International Conference on Distributed Computing Systems*, pages 136–143, Vancouver, BC, Canada, May 1995.

[BB01] Deepak Bansal et Hari Balakrishnan : Binomial congestion control algorithms. *In IEEE/INFOCOM Conference on Computer and Communications*, pages 631–640, Anchorage, AK, USA, April 2001.

[BJP07] Azzedine Boukerche, Guihua Jia et Richard Werner Nelem Pazzi : Performance evaluation of packet loss differentiation algorithms for wireless networks. *In PM2HW2N : Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 50–52, New York, NY, USA, Octobre 2007. ACM.

[BK98] Hari Balakrishnan et Randy Katz : Explicit Loss Notification and Wireless Web Performance. *In IEEE GLOBECOM Global Internet*, Sydney, Australia, Novembre 1998.

[BM02] Dhiman Barman et Ibrahim Matta : Effectiveness of loss labeling in improving TCP performance in wired/wireless networks. *In ICNP, IEEE International Conference on Network Protocols*, pages 2–11, Washington, DC, USA, 2002. IEEE Computer Society.

[BMGS03] Alex Balk, Dario Maggiorini, Mario Gerla et M. Y. Sanadidi : Adaptive MPEG-4 video streaming with bandwidth estimation. *In International Workshop on Quality of Service in Multiservice IP Networks*, 2, pages 525–538, London, UK, February 2003. Springer-Verlag.

[BPSK97] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan et Randy H. Katz : A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, décembre 1997.

[BSAK95] Hari Balakrishnan, Srinivasan Seshan, Elan Amir et Randy H. Katz : Improving TCP/IP performance over wireless networks. *In the 1st annual international conference on Mobile computing and networking*, MobiCom '95, pages 2–11, New York, NY, USA, 1995. ACM.

[BSK95] Hari Balakrishnan, Srinivasan Seshan et Randy H. Katz : Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4):469–481, décembre 1995.

[BV99] Saâd Biaz et Nitin H. Vaidya : Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. *In IEEE ASSET Symposium on Application - Specific Systems and Software Engineering and Technology*, pages 10–17, Washington, DC, USA, 1999. IEEE Computer Society.

[BW04a] Saâd Biaz et Xia Wang : Can ECN be used to differentiate congestion losses from wireless losses ? Rapport technique CSSE04-04, Auburn University, 2004.

[BW04b] Saâd Biaz et Xia Wang : RED for improving TCP over wireless networks. *In ICWN : International Conference on Wireless Networks*, pages 628–636, Las Vegas, USA, June 2004.

[CC96] Robert L. Carter et Mark E. Crovella : Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27-28:297–318, octobre 1996.

[CCV03] Song Cen, Pamela C. Cosman et Geoffrey M. Voelker : End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking*, 11(5):703–717, octobre 2003.

[CGH+07] R. Cazoulat, A. Graffunder, A. Hutter, P. Amon et Wien : Real-Time system for adaptive video streaming based on SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1227–1237, 2007.

[CGM+02] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi et Ren Wang : TCP Westwood : end-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8:467–479, September 2002.

[CHL06] Cheng-Fu Chou, Min-Way Hsu et Ching-Ju Lin : A trend-loss-density-based differential scheme in wired-cum-wireless networks. *In IWCMC : international conference on Wireless communications and mobile computing*, pages 239–244, Vancouver, British Columbia, Canada, July 2006.

[CMC+10] Pedro A. Chaparro, Janio Monteiro, Carlos T. Calafate, Jesus Alcober, Juan-Carlos Cano et Pietro Manzoni : Supporting scalable video transmission in MANETs through distributed admission control mechanisms. *In Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 18, pages 238–245, Pisa, Italy, février 2010. IEEE.

[CWKS97] B.P. Crow, I. Widjaja, J.G. Kim et P.T. Sakai : IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine*, pages 116–126, September 1997.

[DCM10] Luca De Cicco et Saverio Mascolo : An experimental investigation of the akamai adaptive video streaming. *In 6th international conference on HCI in work and learning, life and leisure : workgroup human-computer interaction and usability engineering*, USAB'10, pages 447–464, Berlin, 2010. Springer-Verlag.

[DCMP11] Luca De Cicco, Saverio Mascolo et Vittorio Palmisano : Feedback control for adaptive live video streaming. *In Second annual ACM conference on Multimedia systems*, MMSys, pages 145–156, New York, NY, USA, 2011. ACM.

[DLRRJRG05] Guillaume De La Roche, Raphael Rebeyrortte, Katia Jaffrès Runser et Jean-Marie Gorce : A New Strategy for Indoor Propagation Fast Computation with MR-FDPF Algorithm. *In Antennas, Radar and Wave Propagation*, Banff Canada, 2005. IASTED.

[DLS05] Eugen Dedu, Sébastien Linck et François Spies : Removing the MAC retransmission times from the RTT in TCP. *In Euromedia Conference, Workshop on Distributed Multimedia Databases and Multimedia Adaptation*, 11, pages 190–193, Toulouse, France, avril 2005. Eurosis.

[DRM04] Constantinos Dovrolis, Parameswaran Ramanathan et David Moore : Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions Networking*, 12:963–977, December 2004.

[DRS06] Dominique Dhoutaut, Anthony Régis et François Spies : Impact of radio propagation models in vehicular ad hoc networks simulations. *In VANET : the 3rd international workshop on Vehicular ad hoc networks*, pages 40–49, New York, NY, USA, 2006. ACM.

[DS07] Dominique Dhoutaut et François Spies : Adding geographical interferences into the shadowing pattern model for vehicular ad hoc networks simulations. *In ITST, 7th International Conference on Intelligent Transport Systems Communications*, pages 1–6, Sophia Antipolis, France, 2007.

[Dyn10] Dynamic streaming in flash media server 3.5. Disponible à `http://www.adobe.com/devnet/flashmediaserver/`, 2010. Adobe Systems Inc.

[EDS05] Nicolas Eude, Bertrand Ducourthial et Mohamed Shawsky : Enhancing ns-2 simulator for high mobility ad hoc networks in car-to-car communication context. *In the 7th IFIP International Conference on Mobile and Wireless Communications Networks MWCN*, Morocco, september 2005.

[ETHQ09] Michael Eberhard, Christian Timmerer, Hermann Hellwagner et Emanuele Quacchio : An interoperable delivery framework for scalable media resources. *Wireless Communication*, 16:58–63, October 2009.

[FBB01] Nick Feamster, Deepak Bansal et Hari Balakrishnan : On the interactions between layered quality adaptation and congestion control for streaming video. *In 11th International Packet Video Workshop*, Kyongiu, Korea, April 2001.

[Fen02] Wu-chi Feng : On the efficacy of quality, frame rate, and buffer management for video streaming across best-effort networks. *Journal of High Speed Networks*, 11:199–214, 2002.

[FHPW08] S. Floyd, M. Handley, J. Padhye et J. Widmer : TCP Friendly Rate Control (TFRC) : Protocol Specification. RFC 5348 (Proposed Standard), septembre 2008.

[FJ93] Sally Floyd et Van Jacobson : Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, août 1993.

[FK06] S. Floyd et E. Kohler : Profile for Datagram Congestion Control Protocol (DCCP) congestion control ID 2 : TCP-like congestion control. RFC 4341, mars 2006.

[Fri46] H.T. Friis : A note on a simple transmission formula. *Proc. IRE*, 34, 1946.

[GAA05] Eren Gürses, Gozde Bozdagi Akar et Nail Akar : A simple and effective mechanism for stored video streaming with TCP transport and server-side adaptive frame discard. *Computer Networks*, 48:489–501, juillet 2005.

[GM04] Luigi A. Grieco et Saverio Mascolo : Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *SIGCOMM, Computer Communication Review*, 34:25–38, April 2004.

[GT10] Burak Gorkemli et A. Murat Tekalp : Adaptation strategies for streaming SVC video. *In ICIP, The International Conference on Image Processing*, pages 2913–2916, Hong Kong, Septembre 2010.

[Hau07] Torgeir Haukaas : Rate adaptive video streaming over wireless networks. Mémoire de D.E.A., Norwegian University of Science and Technology, Trondheim, Norway, juin 2007.

[HCR+05] Hsu-Feng Hsiao, Aik Chindapol, J. Ritcey, Yaw-Chung Chen et Jenq-Neng Hwang : A new multimedia packet loss classification algorithm for congestion control over wired/wireless channels. *In ICASSP, IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1105–1108, Philadelphia, Pennsylvania, USA, mars 2005.

[HI06] Árpád Huszák et Sándor Imre : Selective retransmission of MPEG video streams over IP networks. *In International Symposium on Communication System Networks and Digital Signal Processing (CSNDSP)*, 5, pages 125–128, Patras, Greece, juillet 2006.

[HS03] Ningning Hu et P. Steenkiste : Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.

[IT07] ITU-T : Opinion model for video-telephony applications, avril 2007.

[Jac88a] V. Jacobson : Congestion avoidance and control. *SIGCOMM Computer Communication Review*, 18:314–329, August 1988.

[Jac88b] Van Jacobson : Congestion avoidance and control. *In ACM SIGCOMM '88*, pages 314–329, août 1988.

[JR86] Raj Jain et Shawn A. Routhier : Packet trains : Measurements and a new model for computer network traffic. *IEEE Journal on Selected Areas in Communications*, 4:986–995, 1986.

[Kaz02] Matheos Ioannis Kazantzidis : *Adaptive Multimedia in Wireless IP Networks.* Thèse de doctorat, University of California, Los Angeles, USA, 2002.

[Kel01] Frank Kelly : Mathematical modelling of the Internet. *In Mathematics Unlimited — 2001 and Beyond*, pages 685–702, Berlin, Germany, 2001. Springer-Verlag.

[Kes95] Srinivasan Keshav : Packet-pair flow control. *IEEE/ACM Transactions on Networking*, February 1995.

[KHF06] E. Kohler, M. Handley et S. Floyd : Datagram Congestion Control Protocol (DCCP). RFC 4340, mars 2006.

[KST+08] Ingo Kofler, Joachim Seidl, Christian Timmerer, Hermann Hellwagner, Ismail Djama et Toufik Ahmed : Using MPEG-21 for cross-layer multimedia content adaptation. *Signal, Image and Video Processing*, 2(4):355–370, 2008.

[KZ07] Javed Khan et Raid Zaghal : Symbiotic rate adaptation for time sensitive elastic traffic with interactive transport. *Computer Networks*, 51(1):239–257, janvier 2007.

[LBG10] C. Liu, I. Bouazizi et M. Gabbouj : Advanced rate adaption for unicast streaming of scalable video. *In IEEE International Conference on Communications*, pages 1–5, Cape Town, South Africa, May 2010.

[LBG11] Chenghao Liu, Imed Bouazizi et Moncef Gabbouj : Rate adaptation for adaptive HTTP streaming. *In Second annual ACM conference on Multimedia systems*, MMSys, pages 169–174, New York, NY, USA, 2011. ACM.

[LC06] Sunhun Lee et Kwangsue Chung : Mavis : Media-aware video streaming mechanism. *In IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*, pages 74–84, Dublin, Ireland, October 2006.

[LC08] Sunhun Lee et Kwangsue Chung : Buffer-driven adaptive video streaming with TCP-friendliness. *Computer Communications*, 31:2621–2630, June 2008.

[Lin08] Sébastien Linck : *Optimisation et adaptation des communications dans un réseau hétérogène*. Thèse de doctorat, Université de Franche-Comté, Decembre 2008.

[LK08] Arne Lie et Jirka Klaue : Evalvid-RA : trace driven simulation of rate adaptive MPEG-4 VBR video. *Multimedia Systems*, 14(1):33–50, juin 2008.

[LMB+06] Sébastien Linck, Emmanuel Mory, Julien Bourgeois, Eugen Dedu et François Spies : Video quality estimation of DCCP streaming over wireless networks. *In Euromicro Conference on Parallel, Distributed and Network-based Processing*, 14, pages 405–412, Montbéliard, France, février 2006. IEEE.

[Mat04] Nils-Erik Mattsson : A DCCP module for ns-2. Mémoire de D.E.A., Luleå University of Technology, Sweden, mai 2004.

[MJV96a] Steven McCanne, Van Jacobson et Martin Vetterli : Receiver-driven layered multicast. *In Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM, pages 117–130, New York, NY, USA, 1996. ACM.

[MJV96b] Steven McCanne, Van Jacobson et Martin Vetterli : Receiver-driven layered multicast. *SIGCOMM Computer Communication Review*, 26:117–130, August 1996.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd et A. Romanow : TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), octobre 1996.

[Nag84] J. Nagle : Congestion Control in IP/TCP Internetworks. RFC 896, janvier 1984.

[Net] Network simulator — ns-2. `http://www.isi.edu/nsnam/ns/`.

[NO07] Dieu Thanh Nguyen et Joern Ostermann : Congestion control for scalable video streaming using the scalability extension of H.264/AVC. *IEEE Journal of Selected Topics in Signal Processing*, 1(2):246–253, 2007.

[PGLA99] Christina Parsa et J. J. Garcia-Luna-Aceves : Improving TCP congestion control over internets with heterogeneous transmission media. *In ICNP : IEEE*

International Conference on Network Protocols, pages 213–221, Toronto, Ontario, Canada, octobre-novembre 1999.

[PGLA00] Christina Parsa et J.J. Garcia-Luna-Aceves : Differentiating congestion vs. random loss : A method for improving TCP performance over wireless links. *In WCNC :Wireless Communications and Networking Conference, IEEE*, volume 1, pages 90–93, Chicago, IL, USA, septembre 2000.

[Pos80] J. Postel : User Datagram Protocol. RFC 768, août 1980.

[Pos81] J. Postel : Transmission control protocol. RFC 793, septembre 1981.

[PSJ06] Min Kyu Park, Kue-Hwan Sihn et Jun Ho Jeong : A statistical method of packet loss type discrimination in wired-wireless network. *In CCNC : IEEE Consumer Communications and Networking Conference*, pages 458 – 462, Las Vegas, USA, janvier 2006.

[PWM11] Roger Pantos et Jr. William May : HTTP live streaming. IETF Draft, March 2011. Apple Inc.

[RDB09a] Wassim Ramadan, Eugen Dedu et Julien Bourgeois : EcnLD, ECN loss differentiation to optimize the performance of transport protocols on wireless networks. *In International Conference on Ultra Modern Telecommunications and Workshops (ICUMT), WMCNT workshop*, 1, pages 1–6, Saint Petersburg, Russia, octobre 2009. IEEE.

[RDB09b] Wassim Ramadan, Eugen Dedu et Julien Bourgeois : Une méthode de différenciation de pertes pour améliorer la performance des protocoles de transport sur réseaux sans-fil. *In JDIR, 10èmes Journées doctorales en informatique et réseaux*, pages 128–133, Belfort, France, février 2009.

[RDDB11] Wassim Ramadan, Eugen Dedu, Dominique Dhoutaut et Julien Bourgeois : RELD, RTT ECN Loss Differentiation to optimize the performance of transport protocols on wireless networks. *Telecommunications Systems, special issue on Mobile Computing and Networking Technologies*, 2011.

[Rea09] Real-time messaging protocol (RTMP). Specification, 2009. Adobe Systems Inc.

[RFB01] K. Ramakrishnan, S. Floyd et D. Black : The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, septembre 2001.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson : RTP : A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), juillet 2003. Updated by RFCs 5506, 5761, 6051.

[SHR05] Illya Stepanov, Daniel Herrscher et Kurt Rothermel : On the impact of radio propagation models on manet simulation results. *In the 7th IFIP International Conference on Mobile and Wireless Communication Networks (MWCN)*, Marrakech, Morocco, september 2005.

[SSF01] Nagasuresh Seelam, Pankaj Sethi et Wu-chi Feng : A hysteresis based approach for quality, frame rate, and buffer management for video streaming using TCP. *In Management of Multimedia on the Internet*, volume 2216 de *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin, 2001.

[SW04] Thomas Schierl et Thomas Wiegand : H.264/AVC rate adaptation for internet streaming. *In 14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.

[SYL03] Yih-Ching Su, Chu-Sing Yang et Chen-Wei Lee : Optimal FEC assignment for scalable video transmission over burst error channel with loss rate feedback. *Signal Processing : Image Communication*, 18(7):537–547, août 2003.

[TTM+00] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh et H. Tokuda : Achieving moderate fairness for UDP flows by path-status classification. *In LCN : IEEE Local Computer Networks*, pages 252–264, Washington, DC, USA, août 2000.

[WMM01] Naoki Wakamiya, Masayuki Murata et Hideo Miyahara : TCP-friendly video transfer. *In Internet Quality and Performance and Control of Network Systems*, volume 4211, pages 25–35. SPIE, 2001.

[XZ04] Bo Xie et Wenjun Zeng : Rate-distortion optimized dynamic bitstream switching for scalable video streaming. *In IEEE ICME, International Conference on Multimedia and Expo*, volume 2, pages 1327–1330, Taipei, Taiwan, 2004.

[YEY+04] Jijun Yin, Tamer ElBatt, Gavin Yeung, Bo Ryu, Stephen Habermas, Hariharan Krishnan et Timothy Talty : Performance evaluation of safety applications over dsrc vehicular ad hoc networks. *In VANET, A Vehicular Ad-Hoc Network*, pages 1–9, 2004.

[YWZW02] D. Ye, X. Wang, Z. Zhang et Q. Wu : A buffer-driven approach to adaptively stream stored video over internet. *In International Conference on High Speed Networks and Multimedia Communications*, pages 81–85, Beijing, China, 2002.

[Zam09] Alex Zambelli : IIS smooth streaming technical overview, March 2009. Microsoft Corporation.

[ZMA+09] Yousaf Bin Zikria, Shahzad A. Malik, Hassan Ahmed, Sumera Nosheen, Naeem Zafar Azeemi et Shahid A. Khan : Video transport over heterogeneous networks using SCTP and DCCP. *In Wireless Networks, Information Processing and Systems*, volume 20 de *Communications in Computer and Information Science*, pages 180–190. Springer Berlin Heidelberg, 2009.

[ZN00] Jing Zhu et Zhisheng Niu : A reliable TCP-aware link layer retransmission for wireless networks. *In Communication Technology. WCC-ICCT*, volume 1, pages 900–905, Beijing, China, August 2000.

* * *

Résumé

Cette thèse traite de l'amélioration du transfert de données, d'une part sur les réseaux sans-fils et d'autre part pour des données continues telles que la vidéo. Pour améliorer les transmissions sur les réseaux sans-fils nous nous sommes intéressés au contrôle de congestion des protocoles de transport mais nous avons également proposé une méthode pratique d'adaptation de la vidéo aux conditions du réseau.

Cette thèse contient donc deux volets. La première porte sur la différenciation de pertes entre les pertes de congestion et les pertes sur le réseau sans fil. Il est connu que lors d'une perte, les protocoles de transport actuels réduisent le débit (par deux par exemple). Or, pour les pertes sans fil, cela n'a pas d'intérêt. Pour différencier ces pertes sur l'émetteur des données, nous proposons une méthode originale qui utilise à la fois ECN (Explicit Congestion Notification) et le changement sur le RTT du paquet qui suit la perte.

La seconde propose une méthode originale d'adaptation vidéo au niveau de la couche application sur l'émetteur. Avec l'arrivée des vidéos à bitrate élevés (HD, 3D) et l'augmentation constante mais irrégulière des bandes passantes réseau, la qualité vidéo à l'utilisateur reste à la traîne : elle est non-optimale (bitrate beaucoup plus petit ou plus grand que le débit disponible) et non adaptable (aux conditions dynamiques du réseau). Nous proposons une méthode très simple à implémenter, puisqu'elle ne requiert qu'une modification côté émetteur au niveau de la couche application. Elle adapte en permanence le bitrate de la vidéo aux conditions du réseau, autrement dit elle fait un contrôle de congestion sur l'émetteur. La visioconférence est un cas d'application idéal. Cette méthode fonctionne au-dessus de tout protocole de transport avec contrôle de congestion (TCP, DCCP), ce qui lui confère aussi la propriété de TCP-friendliness.

Des simulations sous ns2 et des expérimentations réelles ont été effectués pour valider nos algorithmes. Nous avons donné une grande importance au réalisme des simulations (en utilisant un modèle de pertes sans fil plus réaliste) et à l'implémentation complète des expérimentations. Les résultats montrent tout l'intérêt, à la fois théorique et pratique, de nos propositions.

Mots-clefs : streaming, multimédia, adaptation vidéo.

Abstract

This thesis deals in improving the data transfer on wireless networks and for the continuous data such as video. To improve transmission over wireless networks, we were interested in congestion control transport protocols and we also proposed a practical method for adjusting the video rate to network conditions.

This thesis composes of two parts. The first parts concerns the loss differentiation between congestion losses and losses on the wireless network. It is known that when there is a loss, transport protocols reduce the current sending rate (e.g. by two). However, for wireless losses, it has no interest in reducing the rate. To differentiate these losses on the data senders side, we propose a novel method that uses both the ECN (Explicit Congestion Notification) and the change of RTT of the packet following the loss.

The second part proposes a novel method for video adaptation at the application layer of the sender. With the advent of high bitrate video (e.g. HD, 3D) and steadily increasing but irregular network bandwidth, video quality to the user lags. It is non-optimal (bitrate is highly smaller or larger than the available bandwidth) and not adaptable (to the dynamic conditions of the network). We propose a simple method to implement, since it requires a change only at the application layer of the sender. It adapts the bitrate of the video to the network conditions, i.e. it is a congestion control on the transmitter. Videoconferencing is an ideal case for the application of adaptation. This method works over any transport protocol with congestion control (e.g. TCP, DCCP), which also confers the property of TCP-friendliness.

ns2 simulations and actual experiments were conducted to validate our algorithms. We gave importance to the realism of simulations (using a model that is more realistic for wireless losses) and the complete implementation of the experiments. The results show the interest both theoretical and practical of our proposals.

Keywords: streaming, multimedia, video adaptation.