Université du Québec
Institut national de la recherche scientifique
Centre Énergie Matériaux Télécommunications

UN ALGORITHME D'ADAPTATION DYNAMIQUE DE DÉBIT POUR LES COMMUNICATIONS VIDÉO INTERACTIVES SUR INTERNET

Par

Fakher Oueslati

Mémoire présenté pour l'obtention du grade de
Maître es Sciences, M.Sc.
en télécommunications

Jury d'évaluation

Président du jury et examinateur interne	Monsieur André Girard INRS Énergie Matériaux Télécommunications
Examinateur externe	Monsieur Hechmi Khlifi Nuance Communications Inc.
Directeur de recherche	Monsieur Jean-Charles Grégoire INRS Énergie Matériaux Télécommunications

Avant-propos

Ce travail a été élaboré en collaboration avec l'entreprise Summit-Tech dans le but d'améliorer leur solution de communication vidéo interactive sur Internet.

Je tiens à exprimer ma gratitude et mes respects les plus sincères à mon directeur de recherche, le Professeur Jean-Charles Grégoire pour l'aide qu'il a bien voulu m'accorder tout au long des différentes étapes de mes travaux de recherche, pour ses critiques constructives et ses suggestions pertinentes. Je m'incline pardevant ses qualités humaines et morales dont j'étais toujours impressionné, sa volonté permanente d'aider et ses idées innovantes qui m'ont permis d'élaborer mes travaux de recherches.

Je remercie également Mr Ahmad Vakil, chercheur à Summit-Tech, pour son soutien et son aide. Sa disponibilité, ses suggestions et ses conseils judicieux m'ont aidé à réussir ce mémoire.

J'exprime, enfin, mes profondes reconnaissances à tous les enseignants qui m'ont soutenu au long de mon parcours académique. Que messieurs les membres du jury soient remerciés pour l'attention qu'ils voudraient porter à ce mémoire.

Résumé

Ces dernières années, l'utilisation des services de communication vidéo interactive sur Internet a connu une augmentation très importante. À ce jour, ces services ne peuvent pas garantir une bonne qualité à tout moment à cause de la variation de la disponibilité des ressources du réseau Internet.

Pour résoudre ce problème et pour améliorer la qualité de ces services, nous proposons dans ce mémoire un algorithme d'adaptation dynamique du débit vidéo aux conditions du réseau. Cet algorithme envoie des paquets Forward Error Correction (FEC) redondants en parallèle avec le trafic vidéo afin d'estimer la bande passante disponible. Lorsqu'il y a assez de bande passante, notre algorithme remplace les paquets FEC par des paquets vidéo utiles afin d'améliorer la qualité offerte. Le trafic FEC est utilisé aussi pour augmenter la robustesse du flux vidéo et pour le protéger contrer les flux concurrents.

Dans la phase d'évaluation, nous avons développé une application Web de communication vidéo interactive en utilisant l'API WebRTC qui utilise notre algorithme pour adapter le débit de la vidéo émis aux conditions du réseau. Nous l'avons ensuite testée en utilisant plusieurs bancs de test, chacun d'eux émulant une situation spécifique qui peut se produire dans le réseau Internet. Notre algorithme a satisfait la majorité des tests. En effet, les résultats obtenus ont montré son efficacité à s'adapter rapidement aux conditions du réseau et son rôle dans l'amélioration de la QoE perçue par l'usager.

Mots-clés : Communication vidéo interactive, contrôle de congestion, adaptation de débit, estimation de la bande passante disponible, correction d'erreur sans voie de retour.

Abstract

In the recent past, the use of interactive video communication services on the internet has increased rapidly. However, to this day, the different applications that are offering such services are not able to guarantee a good quality at all times due to the continuous change in the availability of Internet resources.

To resolve this problem, we propose in this master's thesis a dynamic rate adaptation algorithm for interactive video communication services. This algorithm adapts the video bitrate to network conditions. It sends Forward Error Correction redundant packets to probe for available bandwidth. If there is enough available bandwidth, it replaces these FEC packets with video packets to increase the quality of the sent video. The FEC packets are also used to increase the robustness of the video stream against packet loss that is caused by competing traffic.

In the evaluation phase, we have developed a Web application for interactive video communication using the WebRTC API, in which we implemented our algorithm. This application is then executed in several test beds, each one of them emulates a specific condition that could occur on the Internet. The results show that our algorithm achieves very good performance : It adapts rapidly the video bitrate to network conditions and preserves the quality of the communication.

Key-words : Interactive video communication, Congestion control, Rate adaptation, Available bandwidth estimation, Forward Error Correction.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

ALF	Application Level Framing
APP	Application-specific message
BSS	Binary Search Start
CABAC	Context-adaptive Binary Arithmetic Coder
CAVLC	Context-adaptive Variable Length Coder
CNAME	Canonical Name
CPU	Central Processing Unit
CRF	Constant Rate Factor
ECC	Error-Correcting Code
FBRA	FEC-based Rate Adaptation
FEC	Forward Error Correction
GCC	Google Congestion Control
IETF	Internet Engineering Task Force
IP	Internet Protocol
MaxS	Max Start
NACK	Negative Acknowledgement
OTT	Over-The-Top
OWD	One-Way Delay
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QoS	Quality Of Service
RCS	Rich Communication Services
REMB	Receiver Estimated Maximum Bitrate
RFC	Request for Comments
RR	Receiver Report
RTCP RR	RTCP Receiver report

RTP	Real-time Transport Protocol
RTT	Round Time Trip
SDES	Source Description
SIP	Session Initiation Protocol
SR	Sender Report
SSIM	Structural Similarity
SSRC	Synchronization source identifier
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
TMMBR	Temporary Maximum Media Stream Bit Rate Requests
UDP	User Datagram Protocol
VoIP	Voice over IP
VVoIP	Voice and Video over IP

Chapitre 1

Introduction

1.1 Contexte et motivation

De nos jours, les services de communication sont devenus partie intégrale de notre vie moderne. Le désir d'être accessible et d'accéder à des services voix, vidéo et données de n'importe où et à tout moment a amené le nombre d'abonnés aux réseaux mobiles à dépasser les 6.9 milliards en 2014 [1]. Ceci a permis aux opérateurs de télécommunications de dégager des revenus énormes au cours des deux dernières décennies. Le succès des services liés à l'Internet mobile et l'apparition des appareils mobiles innovants tels que les téléphones intelligents et les tablettes ont contribué énormément aux revenus des opérateurs de télécommunications. Avec les progrès technologiques et l'évolution du réseau Internet, il est devenu possible d'exécuter des services de communications multimédia interactives sur le réseau Internet. Ainsi, plusieurs acteurs du monde de l'informatique ont proposé de nouveaux services de communication sur Internet plus riches que les services offerts par les opérateurs et avec un coût très faible. Le faible coût et la richesse de ces services ont capté l'intérêt de centaines de millions d'utilisateurs. Parmi ces services nous citons Skype, Viber et Whatsapp.

Ce genre de services est connu sous le nom service de communication Over-The-Top (OTT). Comme le nom l'indique les services OTT s'insèrent par-dessus des infrastructures de transmission existantes, que ce soient les réseaux téléphoniques, les réseaux sans fils ou les réseaux cellulaires. Autrement dit, les fournisseurs de services OTT ne contrôlent pas le réseau et n'ont pas à payer pour son développement et son entretien, mais ils l'utilisent pour livrer leurs services.

Les fournisseurs de services OTT, qui sont généralement des entreprises de création de logiciels, ont généré des milliards de dollars de bénéfices grâce à leurs services. En même temps, ils ont causé des pertes financière énormes aux opérateurs et au secteur de télécommunications en général. Par exemple, on rapporte dans [2] que Skype a dépassé 280 millions d'utilisateurs actifs mensuels en 2013, qui utilisent 2 milliards de minutes quotidiennement. Skype, indépendamment des autres services de communications populaires de type OTT, est en train de causer des pertes pour l'industrie de télécommunication évaluées à 100 millions de dollars quotidiennement pour un total de 36,5 milliards de dollars chaque année. Les services de communication OTT offrent des fonctionnalités plus riches que celles offertes par les opérateurs téléphoniques, comme la visiophonie, l'envoi de fichiers et plusieurs autres. Cependant, à cause de la nature du réseau Internet, la qualité de ces services est très pauvre. Ils opèrent par-dessus des réseaux qui ont une bande passante disponible limitée et variable, soit parce qu'elle est partagée par de nombreux utilisateurs, soit par la mobilité de l'usager. La gratuité de ces services de communication et leur richesse sont les facteurs principaux de leur succès.

Afin de limiter les pertes causées par les services de communications OTT, les opérateurs développent une nouvelle technologie connue sous le nom de Rich Communication Services (RCS) [3]. Les opérateurs veulent exploiter les réseaux LTE et son cœur tout IP pour fournir des services de communication voix et vidéo interactifs ayant un coût très faible et offrant une qualité de service hautement supérieure à celle offerte par les applications OTT. Ces services seront aussi interopérables entre les différents opérateurs. Le problème de RCS est que son adoption par tous les opérateurs à travers le monde nécessite beaucoup de temps. En conséquence, les opérateurs ont créé un nouveau concept, appelé Telco-OTT. Ce concept définit un scénario dans lequel les opérateurs de télécommunications offrent leurs propres applications OTT. Ceci permet non seulement de minimiser les pertes causées par les applications de communications OTT existantes, mais aussi de faciliter la commercialisation et l'adoption de RCS dans le futur. Pour concurrencer les services existants, les opérateurs doivent donc offrir un service ayant une meilleure qualité.

Le sujet de ce mémoire s'inscrit dans ce contexte. Nous proposons une méthode permettant d'améliorer la qualité des services de communications vidéo sur Internet en adaptant le débit vidéo dynamiquement aux conditions du réseaux.

1.2 Problématique

Les services de communications vidéo sur Internet sont les services les plus exigeants en termes de ressources. Pour pouvoir offrir une bonne qualité aux utilisateurs, ces services doivent être utilisés dans un réseau permettant de garantir un délai et une variation de délai faible, une bande passante disponible suffisante et un taux de pertes presque nul. Le réseau Internet ne permet pas de satisfaire toutes ces exigences. Ce réseau a été conçu initialement pour offrir des services basés sur l'échange de données binaires et textuelles. Ces services ont des contraintes très différentes de celles des services de communication vidéo interactive.

Ainsi, plusieurs technologies et protocoles ont été développés permettant l'exécution de ces services sur les réseaux Internet. Malheureusement, elles ne permettent pas de garantir le bon fonctionnement à tout moment et dans toutes les conditions. En effet, le réseau Internet est un réseau partagé entre plusieurs utilisateurs dont les caractéristiques peuvent varier au cours du temps et même dans des intervalles de courte durée. Un réseau qui garantit les différents critères exigés par un service des communications vidéo interactif à un moment donné peut ne plus le garantir plus tard, par exemple lorsque plusieurs autres utilisateurs se connectent depuis le même réseau d'accès et commencent à utiliser des services très gourmands en terme de bande passante tels que le transfert de gros fichier.

Afin d'offrir une bonne qualité, il faut adapter dynamiquement le service aux conditions du réseau. En conséquence, une application offrant le service de communication vidéo interactive doit adapter le flux vidéo aux variations des conditions du réseau. Elle doit améliorer la qualité de la vidéo transmise en réduisant le taux de compression lorsque les conditions du réseau le permettent et elle doit dégrader la qualité de la vidéo en augmentant le taux de compression lorsqu'il n'y a plus suffisamment de bande passante disponible. L'augmentation du taux de compression dégrade la qualité légèrement, mais permet d'éviter les coupures et les arrêts du flux vidéo qui peuvent rendre la qualité offerte inacceptable.

Quelques algorithmes offrant ce mécanisme d'adaptation ont été développés récemment. Cependant, ils n'offrent pas encore les performances souhaitables. Dans ce mémoire, nous proposons donc un algorithme d'adaptation dynamique de débit pour les communications vidéo interactives sur Internet.

1.3 Contribution

Nous proposons dans ce mémoire un algorithme d'adaptation dynamique du débit vidéo aux conditions du réseau. Cet algorithme envoie des paquets Forward Error Correction (FEC) redondants en parallèle avec le trafic vidéo afin d'estimer la bande passante disponible. Lorsqu'il y a assez de bande passante, notre algorithme remplace les paquets FEC par des paquets vidéo utiles afin d'améliorer la qualité offerte. Le trafic FEC est utilisé aussi pour augmenter la robustesse du flux vidéo et pour le protéger contrer les flux concurrents.

Pour évaluer notre algorithme, nous avons développé une application Web de communication vidéo interactive en utilisant l'API WebRTC qui utilise notre algorithme pour adapter le débit de la vidéo émis aux conditions du réseau. Nous l'avons ensuite testée en utilisant plusieurs bancs de test, chacun d'eux émulant une situation spécifique qui peut se produire dans le réseau Internet.

1.4 Structure du mémoire

Afin de communiquer clairement le travail accompli ainsi que les contributions de ce mémoire, nous avons structuré ce document comme suit :

- Le chapitre 2 explique les techniques utilisées pour pouvoir exécuter les services de communication multimédia sur Internet. Nous mettons l'accent principalement sur les services de communication vidéo interactive sur Internet, car notre algorithme est développé pour ce genre de service. Nous commençons ce chapitre par la présentation des défis et des barrières de l'exécution des services de communications multimédias sur Internet. Ensuite, nous décrivons l'architecture principale usuelle des services de communication vidéo interactive. Nous détaillons après les technologies et les mécanismes permettant l'exécution de ces services sur le réseau Internet. Finalement, nous présentons les deux types de technique de mesure permettant d'évaluer la qualité offerte par ces services.
- Le chapitre 3 présente les causes de la congestion et son effet sur les services de communication vidéo interactive. Il décrit aussi deux algorithmes permettant d'adapter la qualité vidéo dans ces services aux conditions du réseau, ce qui permet d'éviter et de réduire l'effet de la congestion.
- Le chapitre 4 décrit le nouvel algorithme d'adaptation de débit dynamique pour les communications vidéo interactives sur Internet que nous proposons. Nous commençons par une présentation générale de l'algorithme. Nous présentons le principe directeur et les différents indices permettant de détecter la

congestion que nous utilisons. Ensuite, nous détaillons les différentes parties et états de l'algorithme. Finalement, nous présentons deux nouveaux mécanismes permettant l'adaptation aux conditions initiales du réseau pendant la phase de démarrage du service.

– Le chapitre 5 présente les résultats de l'évaluation de performance de notre algorithme. Dans ce chapitre nous exposons l'architecture de l'application que nous avons développée pour nos tests. Nous énumérons aussi les métriques d'évaluation de performance que nous utilisons. Nous présentons également les différents cas de test exécutés et une interprétation des différents résultats obtenus pour chaque test.
– Le chapitre 6 conclut ce document avec un résumé des travaux effectués ainsi que les résultats obtenus. Nous identifions également les travaux futurs afin d'améliorer l'algorithme proposé.

Chapitre 2

Communication vidéo interactive sur Internet

2.1 Introduction

Le réseau Internet a été initialement conçu pour transmettre un trafic de données [4]. La majorité des services utilisés au début de l'apparition de cette innovation ont été généralement basés sur l'échange de données telles que les courriels et les fichiers. Le trafic voix a été desservi exclusivement par les réseaux téléphoniques. Avec les progrès des technologies et l'explosion du nombre d'applications et des utilisateurs d'Internet, le besoin de la communication multimédia est apparu. Le trafic multimédia pose de nouvelles exigences qui sont différents d'un service multimédia à un autre. En effet, il existe plusieurs particularités pour chaque trafic multimédia associé à un type spécifique de service. En conséquence, divers techniques et protocoles permettant l'exécution des services multimédia sur Internet ont été créés.

Dans ce chapitre, nous détaillons les défis de l'exécution des services multimédia sur Internet et nous présentons les différentes techniques utilisées afin de rendre leur exécution possible. Nous mettons l'accent sur les communications vidéo interactives qui sont les services de communications les plus exigeants en ressources, car l'algorithme que nous proposons est spécifique à ce type de service.

2.2 Défis de l'exécution des services multimédia sur Internet

L'exécution des services multimédia sur Internet présente un grand défi à cause de la nature du réseau Internet d'une part et à cause de la nature du trafic multimédia d'autre part.

Il est possible de classifier le trafic de communication sur Internet en plusieurs classes. Chaque classe a des besoins spécifiques en matière de ressources. La liste suivante est un exemple des différentes classes de trafic qu'on peut trouver dans le réseau Internet. Elle est ordonnée de la classe la moins exigeante en ressources jusqu'à la classe la plus exigeante :

- Trafic « Best effort » : généré par les navigateurs.
- Trafic de données en masse « Bulk data » : généré généralement par le transfert des fichiers et les courriels.
- Trafic de données transactionnel : généré généralement par les applications utilisant un modèle client-serveur.
- Trafic multimédia non interactif : généré généralement par les applications de streaming vidéo et audio.
- Trafic multimédia interactif : généré généralement par les applications de communication voix/vidéo interactives.

Les applications multimédias non interactives peuvent être divisées en deux sous-catégories : la diffusion des fichiers vidéo/audio pré-enregistrés et la diffusion des flux vidéo/audio en direct. Les applications appartenant à la première sous-catégorie délivrent les flux vidéo et audio sauvegardés dans un serveur vers les clients. YouTube est un exemple de compagnie offrant ce genre de service. Dans ce type d'application, l'utilisateur commence la lecture du fichier (vidéo ou audio) pendant que le fichier est en train d'être reçu. Les applications de cette catégorie ne sont pas très sensibles au délai de bout en bout. La seule condition requise est qu'une fois la lecture du flux commencée, elle ne s'arrête pas avant la fin de la vidéo ou l'arrêt commandé par l'utilisateur. Le contenu sauvegardé peut être mis dans une mémoire tampon afin de réduire les effets du délai et de la gigue du réseau (c.-à-d. la variation du délai). Dans la deuxième sous-catégorie, les applications affichent à l'utilisateur un contenu diffusé en direct. Elles sont donc plus strictes en termes de délai et d'exigence de QoS.

Dans les applications multimédia interactives, telles que la voix sur IP (VoIP) et la vidéoconférence (Vidéo sur IP), toutes les communications doivent être en temps réel et les données ne peuvent pas être mises dans des mémoires tampons pour une longue période afin de diminuer l'effet de délai et de la gigue. Ces applications sont donc plus strictes en termes d'exigences de QoS. Par exemple, les applications VVoIP (Voix

et vidéo sur IP) interactives doivent respecter un délai inférieur à 150 ms et une gigue inférieure à 50 ms afin d'offrir une bonne qualité d'expérience utilisateur [5]. Il faut noter que les applications de communication vidéo interactives sont plus exigeantes que celles n'utilisant que la voix car le débit des vidéos est très supérieure à la taille des données audio. Elles ont donc besoin d'une grande disponibilité de bande passante afin d'offrir une bonne qualité.

La disponibilité de la bande passante sur Internet est considérée comme une barrière à l'exécution des services multimédias sur ce réseau. La bande passante offerte à l'utilisateur final est généralement trop limitée et partagée avec plusieurs autres utilisateurs. À cause de la taille importante du contenu média il est presque impossible d'exécuter des services multimédias sur Internet sans utiliser un moyen permettant de réduire la taille du contenu. Ainsi, il faut compresser les contenus voix et vidéo avant de les envoyer d'un utilisateur à un autre. Nous présentons dans la section 2.4 une méthode de compression utilisée pour réduire la taille des flux vidéo.

Le modèle best-effort d'Internet sur IP a montré son efficacité pour le transport de données textuelles ou binaires (c.-à-d. le trafic de données). Dans ce modèle de service, le protocole IP fait de son mieux pour livrer les paquets aux récepteurs, mais sans fournir de garanties. Par conséquent, les paquets envoyés peuvent être perdus ou délivrés dans un ordre différent, et avec un délai imprévisible. L'avantage de ce modèle est qu'il est possible de fournir le service Internet en utilisant des routeurs très simples et une architecture évolutive, deux caractéristiques cruciales pour le succès d'Internet. Les services de données, par contre, ne tolèrent pas la perte de paquets et afin d'augmenter la fiabilité des services de bout en bout, les données sont transportées par le protocole TCP, qui est le protocole le plus utilisé dans les réseaux IP. TCP a des mécanismes permettant de minimiser les inconvénients d'IP et d'offrir un canal réseau fiable de bout en bout. Par exemple, TCP retransmet les paquets perdus et maintient leur ordre de transmission. TCP a aussi des mécanismes qui contrôlent et évitent la congestion. De ce fait, TCP est la solution optimale pour le transport du trafic de données sur Internet. IP, qui est implémenté dans tous les nœuds, est simple et évolutif, et TCP, qui est implémenté dans tous les nœuds terminaux, offre la fiabilité nécessaire. Cependant, cette solution a plusieurs problèmes lorsqu'on essaye de transporter le trafic multimédia. Les applications multimédias sont généralement très sensibles au délai de bout en bout et à sa variation (c.-à-d. la gigue), mais peuvent tolérer un faible taux de perte de paquets. TCP prend soin des pertes de paquets, mais ne garantit pas de limite ni pour le délai, ni pour la gigue. De ce fait, les applications multimédias ne fonctionnent pas correctement sous TCP. Par exemple, le fait de retransmettre des paquets contenant une partie d'une scène vidéo perdue pendant la transmission initiale peut aggraver le délai. Ce trafic vidéo retransmis peut chevaucher le trafic

d'une nouvelle scène vidéo et l'endommager. Pour éviter ce problème, la majorité du trafic multimédia est transporté par UDP qui est un autre protocole de transport sur IP [6].

À cause des exigences en termes de délai et de gigue, les applications multimédias ont besoin d'un protocole de transport ayant des caractéristiques différentes de TCP, mais avec plus de fonctionnalités que celles offertes par une simple encapsulation applicative d'IP. UDP est néanmoins insuffisant pour le trafic multimédia, et ne permet pas de transporter des informations permettant, par exemple, la synchronisation des flux multimédias. Un autre protocole est donc nécessaire, qui peut être transporté par UDP. Nous présentons ce protocole dans la section 2.5.1.

2.3 Fonctionnement de la communication vidéo sur Internet

Généralement, tous les systèmes de communications ont une architecture globale semblable. Ils sont divisés en deux parties : l'émetteur et le récepteur. La figure 2.1 montre l'architecture usuelle d'un système de communication vidéo. L'émetteur est composé de 4 entités : la caméra, l'encodeur, un empaqueteur et une mémoire tampon d'émission. La caméra capture la vidéo sous forme de données brutes non compressées (p. ex. en format RGB ou YUV) [7] et l'envoie vers l'encodeur. Ce dernier compresse les données reçues en utilisant un codec vidéo spécifique, voir section 2.4. Actuellement, le codec le plus utilisé pour les communications vidéo est le codec H.264/AVC. La vidéo compressée est ensuite envoyée vers l'entité suivante. L'empaqueteur met les données reçues dans des paquets RTP et les envoie vers la mémoire tampon de réception avant qu'elles soient envoyées vers le récepteur.

Le récepteur est aussi composé de 4 entités : la mémoire tampon de réception (connue sous le nom mémoire tampon de la gigue ou jitter buffer en anglais), le dépaqueteur, le décodeur et le lecteur. Les paquets reçus sont mis dans la mémoire tampon de réception. C'est une mémoire de faible taille qui permet de minimiser l'effet de la gigue sans introduire beaucoup de délais. Elle réordonne aussi les paquets avant de les transmettre vers le décodeur. De plus, elle supprime les paquets arrivant trop tard pour être lus. Le dépaqueteur enlève l'entête RTP des paquets reçus et les envoie vers le décodeur. Ce dernier décode les données reçues. Finalement, le lecteur lit la vidéo à partir des données décodées et l'affiche à l'utilisateur.

Pour les communications interactives, chaque participant doit avoir les deux parties afin de pouvoir communiquer : un émetteur pour envoyer sa vidéo et un récepteur pour voir la vidéo de l'autre participant.

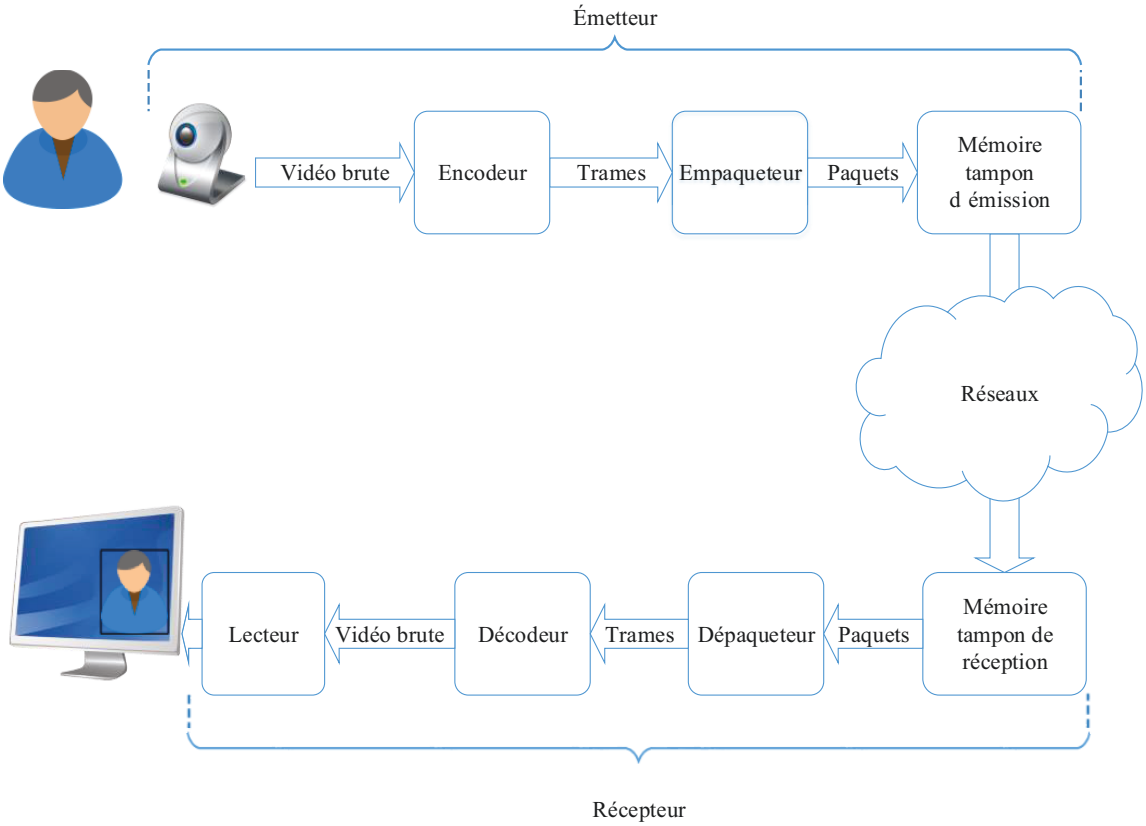


FIGURE 2.1 – Architecture des applications de communication vidéo

Pour pouvoir fonctionner, les applications de communications interactives doivent utiliser deux catégories de protocoles : des protocoles de transport de flux multimédia et des protocoles de signalisation.

Les protocoles de transport de flux multimédia sont des protocoles qui s'occupent du transport des données audio et/ou vidéo de l'émetteur vers le récepteur. Le protocole le plus connu et le plus utilisé de cette catégorie est le protocole RTP. Les protocoles de signalisation sont des protocoles qui offrent des mécanismes de gestion d'appel comme l'établissement et la terminaison d'appel. Le protocole SIP [8] appartient à cette catégorie. Puisque nous nous intéressons à la partie multimédia dans ce mémoire, nous nous intéresserons qu'au protocole RTP, que nous présentons dans la section 2.5.1.

2.4 Compression vidéo

Les flux vidéo sont très gourmands en bande passante. Pour pouvoir offrir des services de communication vidéo interactive, il faut réduire considérablement la taille de la vidéo avant de la transmettre sur Internet. De ce fait, il est donc nécessaire d'utiliser une technique de compression.

La compression vidéo consiste ici en la réduction du nombre de bits utilisés pour représenter la vidéo. Il existe plusieurs normes de compression. De nos jours, la norme de compression la plus utilisée pour les communications interactives est la norme H.264/AVC [9] et nous nous basons sur elle pour la description ci-dessous.

Une vidéo est composée d'une succession de trames (images) ordonnées en fonction du temps. Chaque image est d'abord compressé en exploitant la redondance spatiale existante. L'encodeur divise chaque image en plusieurs petits blocs et applique ensuite diverses techniques (DCT, quantification, codage par plage, codage entropique, etc.) [10] pour réduire la taille des images. Pour appliquer une compression plus élevée, l'encodeur exploite aussi la redondance temporelle. En effet, les trames vidéo consécutives contiennent des images semblables, il est donc inutile d'envoyer toutes les informations de chaque trame. Par conséquent, l'encodeur divise les trames en différents types pour minimiser la taille de la vidéo:

- Trame-I (Intra-coded frame) : La trame-I est une image fixe compressée spatialement. Elle contient des informations concernant toute la trame. Elle peut être décodée indépendamment des autres trames. Pour cela, sa taille est beaucoup plus grande que celle des autres.
- Trame-P (Predicted frame) : La trame-P contient les informations concernant la différence incrémentale avec la trame I et P précédente. Par exemple, pour encoder une séquence contenant un arrière-plan fixe et un ballon en mouvement, il suffit d'encoder une trame-I et une série de trame-P contenant seulement les informations de mouvement du ballon.
- Trame-B (Bi-predictive frame) : La trame-B est une image prédite d'une façon bidirectionnelle, c'est-à-dire qu'elle est construite depuis une image future et d'une image passée. Elle a une taille plus petite que les trames-P, mais nécessite plus de temps de calcul pour être encodée et décodée. Pour cela, elle n'est pas généralement utilisée dans les communications vidéo interactives.

La figure 2.2 montre des groupes d'images (GOP). Un GOP est une partie de la vidéo composée par un trame-I suivie de plusieurs trames-P et -B.

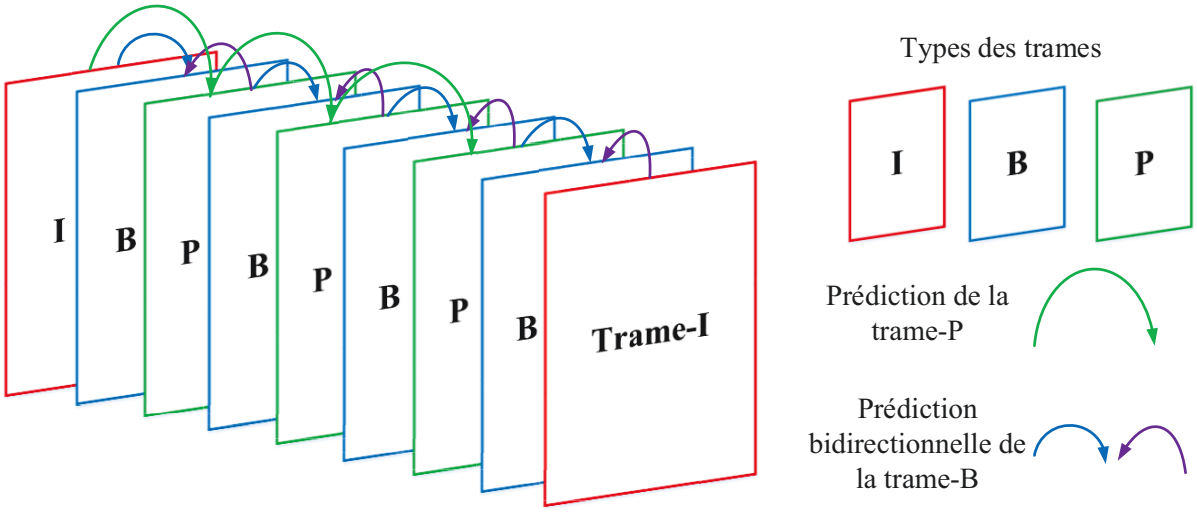


FIGURE 2.2 – GOP avec trames-I, -P et -B

Le codec H.264 contient plusieurs autres techniques afin d'offrir un bon niveau de compression. Il est aussi très flexible et peut couvrir un large éventail d'applications, en temps-réel ou non, de grand à faible débit. Cette flexibilité est due au fait qu'il offre à l'utilisateur un grand nombre de mécanismes optionnels et de paramètres de configuration. À cause de sa grande souplesse, le codec H.264 nous sert de référence dans ce travail. Notre algorithme change les paramètres de ce codec en fonction des conditions du réseau. Si les conditions sont bonnes, il diminue le taux de compression afin d'augmenter la qualité et si les conditions ne sont pas bonnes, il augmente le taux de compression pour diminuer le débit utilisé, mais ceci diminue la qualité de la vidéo. Une explication plus détaillée du fonctionnement de notre algorithme est présentée dans le chapitre 4

H.264 offre un grand nombre de paramètres de configuration. Chaque paramètre peut influencer un ou plusieurs des éléments suivants : le débit de la vidéo, le temps nécessaire pour la compression, le CPU utilisé et la qualité de la vidéo. Parmi ces paramètres nous citons [10]:

- **keyint** :définit l'intervalle maximal entre deux trames de référence, « Key frames ». Ces trames sont des trames-I qui sont utilisées en tant que référence pour décoder les trames-P et -B suivants.
- **ref** : contrôle la taille de la mémoire tampon appelé « Decoded Picture Buffer» (DBF). Cette mémoire stocke un nombre de trames de référence qui peuvent être utilisées par les trames-P. Ce paramètre accepte une valeur entre 0 et 16. Par exemple, si cette valeur est fixée à 8, La DBF peut stocker

8 trames de référence. Ceci signifie que les trames-P peuvent utiliser jusqu'à 8 trames en tant que références de prédiction.

- **no-cabac** : désactive l'utilisation de la technique de compression appelée Context-adaptive Binary Arithmetic Coder (CABAC). En conséquence, l'encodeur utilise la technique de compression Context-adaptive Variable Length Coder (CAVLC). Ceci permet de réduire le temps de compression au coût du taux de compression (réduit de 10 à 20 %).
- **qp** : active et détermine le premier mode de contrôle de débit connu sous le nom de Constant Quantizer mode. Dans ce mode, l'encodeur utilise un quantificateur constant pendant l'encodage des trame-P. Ce mode n'est pas recommandé, car le mode CRF, expliqué ci-dessous, permet d'offrir une meilleure qualité subjective pour le même débit.
- **bitrate** : active le deuxième mode de contrôle de débit, connu sous le nom de Constant Bitrate Rate. Lorsque ce paramètre est initié, l'encodeur encode la vidéo avec une méthode permettant de fixer le débit au cours du temps, mais la qualité ne sera pas fixe.
- **Constant Rate Factor (CRF)** : active le troisième mode de contrôle de débit dans H.264. Dans ce mode l'encodeur fixe la qualité de la vidéo. Ceci est réalisé en utilisant pour chaque trame un taux de compression différent. L'œil humain est moins sensible aux détails lorsqu'un objet est en mouvement, et par conséquent l'encodeur peut appliquer un taux de compression élevé pour les objets en mouvements et un taux de compression moins élevé pour ceux qui sont fixes. Ce mode permet donc d'avoir une qualité subjective fixe tout au long de la vidéo, mais avec un débit variable. Ce paramètre peut prendre une valeur entre 0 et 51 (0 compression sans perte et 51 compression très élevée).
- **Mode de quantification adaptative (aq-mode)** : permet de mieux répartir les bits disponibles entre tous les macroblocs de la vidéo.
- **Complexité de l'estimation des sous-pixels (subme)** : contrôle la complexité de l'estimation des sous-pixels. Ce paramètre peut prendre une valeur qui varie entre 0 et 11. Les valeurs les plus hautes offrent une meilleure qualité et un haute compression au coût de la rapidité de compression.
- **Estimation de mouvement (me)** : définit la méthode d'estimation de mouvement à utiliser. Les méthodes d'estimation possible sont : diamond (dia), hexagon, uneven multi-hex (umh), exhaustive multi-hex (esa), transformed exhaustive (tesa).

Pour une communication vidéo interactive, il importe de pouvoir garantir un délai minimal. Pour cela, il faut fixer les paramètres permettant de minimiser le temps nécessaire pour la compression. Afin d'adapter la vidéo à la bande passante disponible du réseau, l'algorithme doit varier les paramètres ayant un effet sur le débit de la vidéo tout en essayant de maintenir un débit minimal et une qualité maximale. Notons que la va-

riation de certains paramètres peut augmenter le débit de la vidéo sans causer une amélioration remarquable de la qualité. Il est donc nécessaire de les détecter et d'éviter de les modifier. Ceci peut augmenter considérablement les performances de notre algorithme. Nous présentons les paramètres que notre algorithme va contrôler dans le chapitre 4.

2.5 Protocoles de transport multimédia

Comme nous l'avons déjà vu dans la section 2.2, les protocoles TCP et UDP sur IP ne permettent pas de satisfaire les exigences des applications multimédias interactives. Pour cela, la norme RTP a été créée. Le standard RTP a défini une paire de protocoles : RTP et RTCP. Le protocole RTP est utilisé pour le transport des données en temps réel. Le protocole RTCP fournit des statistiques et des informations de contrôle relatif au flux RTP associé.

2.5.1 RTP

RTP est un protocole de transport de trafic temps réel comme la voix et la vidéo sur les réseaux IP. Il a été défini par l'IETF dans le RFC 3550 [11] et il est utilisé principalement dans les services voix et vidéo sur IP. RTP fonctionne sur UDP/IP, mais il existe plusieurs initiatives afin de le rendre indépendant d'UDP pour qu'il soit possible de le transporter sur d'autres protocoles.

Le protocole RTP est un protocole léger qui offre un support aux applications interactives pour détecter les pertes, identifier le type du contenu et le sécuriser. RTP a été conçu en suivant le principe du « Application Level Framing » (ALF) proposé par Clark et Tennenhouse en 1990 [12]. ALF est basé sur le fait que les applications peuvent comprendre leurs besoins mieux que les réseaux. Ce qui signifie que l'intelligence doit être placée au niveau de l'application et que le réseau doit rester simple. Par exemple, une application de communication vidéo utilisant le codec H.264 sait mieux comment réagir lorsqu'une trame I ou B est perdue.

2.5.2 RTCP

RTCP est le protocole de contrôle du standard RTP. Il est conçu pour fonctionner en collaboration avec le protocole RTP et il est aussi défini dans le RFC 3550 [11]. RTP a été conçu pour une diffusion point à

multipoint, par-dessus IP multicast et permet à tous les récipiendaires d'envoyer périodiquement un feed-back, sous forme de paquets RTCP, à la source.

Les paquets RTCP contiennent des rapports de l'expéditeur et/ou du récepteur comportant plusieurs statistiques comme le nombre de paquets envoyés, le nombre de paquets perdus, la gigue des paquets reçus. Ces informations peuvent être utiles pour des applications adaptatives qui peuvent les exploiter afin d'envoyer des données de haute ou de basse qualité selon le niveau de congestion. Ces applications doivent augmenter le taux de compression lorsque la bande passante disponible est limitée et doivent diminuer le taux de compression, ce qui augmentera la qualité, lorsqu'il y a assez de bande passante. Ces informations peuvent être aussi utilisées pour des raisons de diagnostic afin de localiser un problème particulier. RTCP offre aussi la possibilité de synchroniser plusieurs flux média qui viennent de la même source.

Le RFC 3550 [11] définit 5 types de paquets RTCP pour transporter les informations de contrôle. Ces 5 types sont les suivants :

- Rapport du récepteur (RR) : Les RR sont envoyés périodiquement par les récepteurs d'un trafic RTP, qui n'ont pas envoyé des paquets RTP. Ces rapports contiennent des informations de feed-back sur la qualité des données reçues. Parmi ces informations, on trouve le nombre des paquets reçu, le nombre de paquets perdus, la gigue des paquets reçus et un horodatage permettant le calcul de la RTT entre l'émetteur et le récepteur.
- Rapport de l'émetteur (SR) : Les SR sont générés par les participants actifs, ceux qui envoient des paquets RTP. Outre les informations de feed-back similaire à celles se trouvant dans les RR, ils contiennent une section d'informations relatives à l'émetteur. Cette section contient des informations concernant la synchronisation entre les flux média envoyés et le nombre de paquets envoyés.
- Description de la source (SDES) : Les SDES contiennent des informations décrivant les sources. Dans les paquets RTP, les sources sont identifiées par un nombre de 32 bits choisi arbitrairement. Ces identifiants ne sont pas pratiques pour les êtres humains et les SDES contiennent donc des informations textuelles, nommées CNAME, qui sont des identifiants globaux des participants de la session. CNAME peut contenir des informations supplémentaires telles que le nom de l'émetteur, son numéro de téléphone, son adresse e-mail et d'autres informations.
- BYE : Une source envoie le message BYE pour arrêter les flux média. Il permet à un participant d'annoncer qu'il va quitter la conférence.
- Message spécifique à l'application (APP) : Les APP fournissent un mécanisme pour concevoir des extensions pour de nouvelles fonctionnalités.

Les paquets RTP et RTCP sont envoyés sur la même adresse IP, mais sur des ports différents. Les rapports RTCP doivent être envoyés périodiquement par tous les participants, même dans une session multicast contenant un nombre très élevé de participants.

Il faut noter que, bien que le protocole RTCP offre des informations de feed-back concernant la QoS, il ne spécifie pas comment ces informations doivent être utilisées et laisse la décision à la couche d'application. En effet, comme nous l'avons déjà indiqué, chaque type d'application multimédia a des exigences spécifiques en matière de bande passante, délai, perte de paquets, etc. En conséquence, il n'est pas possible de trouver un ensemble unique d'opérations permettant de satisfaire les besoins de tous les types d'application multimédia. Il est donc nécessaire pour chaque application d'utiliser ses propres opérations d'adaptation du flux aux conditions du réseau en se basant sur les informations de feed-back afin d'améliorer la QoS. Ceci est très différent de TCP qui offre une interface uniforme pour un nombre important d'applications ayant des exigences homogènes en terme de QoS : insensibilité aux variations de délai et la bande passante, mais très sensible aux pertes des paquets (donc qui nécessitent une fiabilité parfaite).

L'algorithme d'adaptation de débit que nous proposons dans le chapitre 4 utilise principalement les informations de QoS provenant des rapports du récepteur RR pour décider quelles opérations il faut exécuter.

2.5.3 Les profils RTP

Le standard RTP est conçu d'une façon lui permettant de supporter une grande variété d'applications. Il offre des mécanismes avec lesquels de nouvelles applications peuvent être développées sans devoir le mettre à jour. Pour chaque classe d'applications, RTP définit un profil et un ou plusieurs formats de charges utiles. Les profils offrent un ensemble d'information permettant à l'application des classes associées de comprendre le contenu des champs des entêtes RTP. Les profils peuvent aussi spécifier des extensions et des modifications pour RTP qui sont spécifiques à une classe particulière d'applications. Les spécifications de format de charges utiles expliquent comment les données qui suivent l'entête RTP doivent être interprétées. Actuellement, 4 profils RTP ont été définis : RTP/AVP [13], RTP/SAVP [14], RTP/AVPF [15], RTP/SAVPF [16].

Le profil RTP/AVP

Le profil RTP/AVP (RTP Profile for Audio and Video Conferences with Minimal Control) est le profil le plus ancien du standard RTP. Ce profil définit une spécification et un ensemble de règles permettant

d'utiliser les protocoles RTP et RTCP pour le transport des données vidéo et audio. Ce profil permet donc d'utiliser RTP dans les applications de communication vidéo et audio sur Internet. Il présente des aspects qui n'existent pas dans le standard RTP et il offre une interprétation des champs génériques de RTP spécifique pour les communications vidéo et audio. Ce profil définit aussi une liste de spécifications d'encodage dont chacune décrit comment chaque type de contenu multimédia doit être encapsulé dans les paquets RTP. Il présente aussi une grande liste d'entêtes de charge utile. Chaque entête est spécifique à un format vidéo ou audio différent.

Le profil RTP/SAVP

Le profil RTP/SAVP, connue sous le nom Secure Real-time Transport Protocol (SRTP), offre plusieurs fonctionnalités permettant de sécuriser le contenu média se trouvant dans les paquets RTP. Il peut assurer la confidentialité, l'authentification et l'intégrité des messages et la protection contre les attaques de rejeu « Replay Attack » des données RTP et RTCP. SRTP définit un ensemble d'algorithmes de chiffrement à utiliser et offre la possibilité d'utiliser de nouveaux algorithmes.

Le profil RTP/AVPF

Le profil RTP/AVPF (Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feed-back) est une extension du profil RTP/AVP. Il permet aux récepteurs de fournir un feed-back plus rapide vers l'émetteur. Le profil RTP/AVPF présente un nouveau type de paquet RTCP, appelé RTCP Feed-back message (FB message). Ce paquet permet de signaler la réception où la perte d'un morceau particulier du flux média ou de donner une autre forme de feed-back immédiat concernant les données reçues. Plusieurs types de messages FB ont été définis :

- NACK générique : Ce message est utilisé pour indiquer la perte d'un ou plusieurs paquets RTP.
- Picture Loss Indication (PLI) : Le décodeur informe l'encodeur de la perte d'un morceau de données vidéo encodé appartenant à une ou plusieurs images en utilisant le message PLI. Ce message est utile en lien avec un codec basé sur la prédiction inter-image (p. ex. H.264). Lors de la réception de PLI, l'encodeur se rend compte que la chaine de prédiction est rompue et il envoie donc immédiatement une trame-I au lieu d'envoyer une trame-P ou –B qui ne peuvent pas être décodées.
- Slice Loss Indication (SLI) : En utilisant ce message, l'encodeur peut informer le décodeur qu'il a détecté la perte consécutive d'un ou plusieurs macroblocs [9].

– Reference Picture Selection indication (RPSI) : Les codecs vidéo modernes (p.ex. H.264, H.265) permettent non seulement d'utiliser la trame vidéo la plus récente pour le décodage prédictif, mais aussi des trames de référence plus anciennes. Par conséquent, si l'encodeur apprend quelle image de référence a été correctement reçue, il peut l'utiliser en tant que référence pour encoder les images suivantes. Afin d'informer l'encodeur de la dernière trame de référence correctement ou incorrectement reçu, le message RPSI est utilisé.

Dans le standard RTP défini par le RFC 3550 [11], les paquets RTCP sont envoyés suivant un intervalle fixe. Il est recommandé que cet intervalle soit supérieur à la limite de 5 s. Il est possible d'utiliser une limite inférieure à 5 s, qui est égale à 360 s divisé par la bande passante utilisée par le flux RTP en Kb/s lorsque certaines conditions définies dans [11] sont remplies. Cette nouvelle limite est inférieure à 5 s lorsque la bande passante utilisée par le flux RTP est supérieure à 72 Kb/s. Dans RTP/AVPF, cette limite n'est plus prise en compte. Il est donc possible d'envoyer des rapports RTCP en utilisant des intervalles très courts. De plus, si un évènement particulier se produit (p. ex. la perte d'un paquet), le récepteur peut envoyer un message FB sans attendre l'écoulement de l'intervalle RTCP. Ceci permet donc de minimiser le temps de réponse de l'émetteur et donc d'implémenter des mécanismes de correction d'erreurs basés sur le feed-back, comme la retransmission NACK.

Nous utilisons ce profil dans notre travail, car il permet d'envoyer des messages de feed-back rapide, ce qui est essentiel pour minimiser le temps de réponse de notre algorithme et pour détecter les congestions le plus tôt possible. De plus, ce profil permet d'implémenter le mécanisme de retransmission NACK, détaillé dans la section 2.6 , en utilisant les messages FB de type NACK générique.

Le profil RTP/SAVPF

Le profil RTP/SAVP (Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feed-back) permet de sécuriser les flux RTP et RTCP utilisant le profil RTP/AVPF. Il est une combinaison du profil RTP/SAVP et du profil RTP/AVPF.

2.6 Mécanismes de correction d'erreurs

Les services de communication multimédia sont moins sensibles aux pertes des paquets que les services basés sur l'échange de données [5]. Cependant, il faut noter que les services multimédias ne peuvent tolérer

qu'un faible taux de perte. Le seuil de tolérance varie énormément d'un type de service à un autre. Pour que la dégradation de la qualité ne soit pas perceptible, le taux de pertes pour le service VoIP ne doit pas dépasser le seuil de 1 % [17]. Les communications vidéo sur IP sont plus sensibles aux pertes de paquets, car elles utilisent des paquets de plus grandes tailles et la perte d'un seul paquet peut engendrer la perte de plusieurs trames vidéo successives (p. ex. la perte d'un paquet appartenant à une trame-I). Pour cela, le taux de perte de paquets pour ce genre de service ne doit pas dépasser le seuil de 0.01 % [17].

Nous avons déjà vu dans les sections précédentes que RTP sur UDP/IP offre un service non fiable qui peut seulement détecter le taux de perte de paquets. Pour cela, il faut utiliser des mécanismes permettant de corriger les erreurs causées par les pertes de paquets et/ou des mécanismes permettant de minimiser l'effet de la perte sur la perception de l'utilisateur, communément appelés mécanismes de dissimulation d'erreur. Dans cette section, nous nous concentrons seulement sur les mécanismes de correction, car notre algorithme les utilise afin d'augmenter la robustesse des flux vidéo face aux pertes.

Les mécanismes de correction d'erreurs sont des techniques utilisées par l'émetteur qui aident le récepteur à corriger les problèmes causés par les pertes de paquets ou d'autres erreurs de transmission. Ces techniques se divisent principalement en deux catégories de base : la retransmission et la Forward Error Correction (FEC).

2.6.1 Retransmission

Lorsqu'un paquet est perdu, le récepteur demande à l'émetteur de le retransmettre en utilisant un message de feed-back spécifique. Afin de standardiser le mécanisme de retransmission, un mécanisme connu sous le nom d'acquittement négatif (NACK) a été défini dans le profil RTP/AVPF. Ce profil offre la possibilité d'envoyer un message NACK indiquant quel paquet a été récemment perdu. Dans les communications multimédias interactives, le récepteur ne doit pas envoyer un message NACK pour tous les paquets perdus. En effet, avant d'envoyer un message NACK, le récepteur doit vérifier si le paquet qui sera retransmis aura la chance d'être livré avant qu'il ne soit joué en se basant sur RTT et le délai de mémoire tampon. Lorsque l'émetteur reçoit un message NACK, il doit renvoyer tous les paquets perdus indiqués dans ce message. Malgré la vérification du récepteur, les paquets retransmis peuvent arriver en retard. Il existe donc deux stratégies possibles pour la manipulation de ces paquets. Le récepteur peut ignorer ces paquets et continuer le décodage des paquets reçus. Ceci peut causer des artefacts pour la vidéo et des courtes modifications dans le son reçu. Le récepteur peut aussi interrompre l'encodage et attendre la réception du paquet. Une fois le paquet reçu,

le décodeur reprend son activité. Ceci engendre de petites périodes de pause dans le flux média. L'utilisation de la deuxième stratégie est plus efficace surtout pour communication vidéo, car la perte d'un paquet appartenant à une trame peut causer des artefacts qui persistent dans toutes les trames suivantes, jusqu'à la réception d'une trame-I.

Le format des paquets NACK et des paquets RTP retransmis sont définis dans le RFC 4588 [18]. Les paquets RTP retransmis sont séparés du flux média principal en utilisant un identifiant de source de synchronisation (SSRC) différent ou une nouvelle session pour ne pas perturber les statistiques de RTCP.

Afin de bénéficier des avantages de la retransmission, nous utilisons le mécanisme NACK dans l'implémentation de notre algorithme. Notons que ce mécanisme augmente l'utilisation de la bande passante et peut augmenter la perte de paquets lorsque la perte originale est causée par la congestion du réseau [19]. Pour cela, il faut prendre en compte la bande passante utilisée par le NACK dans notre algorithme et de le désactiver lorsqu'il y a une congestion.

La retransmission de paquets est une technique très efficace qui permet de diminuer énormément la sensibilité des applications multimédias interactives face aux pertes dans les réseaux à faible RTT, tout en utilisant une bande passante limitée. Cependant, son efficacité est réduite dans les réseaux ayant des délais élevés comme les réseaux cellulaires. Pour cela, d'autres mécanismes comme le FEC ont été créés.

2.6.2 Forward Error Correction

La correction d'erreur sans voie de retour (« Forward Error Correction », ou FEC) est un mécanisme qui permet de détecter et corriger les erreurs pendant la transmission des données sur un lien de communication non fiable [20]. Dans le mécanisme FEC, l'émetteur ajoute au message à envoyer une information redondante appelée code correcteur d'erreur (ECC). Cette redondance aide le récepteur à détecter un nombre limité d'erreurs et à les corriger sans besoin de demander au récepteur de retransmettre le message corrompu, mais au coût de l'utilisation d'une bande passante supplémentaire. Ainsi, le mécanisme FEC est généralement utilisé lorsque la retransmission des paquets perdus est coûteuse ou impossible.

Le premier code correcteur d'erreur, le code de Hamming (4, 7), a été créé en 1950 par le mathématicien américain Richard Hamming en 1950 [21]. De nos jours, plusieurs ECC ont été créés tels que code à parité, le code Reed-Solomon, ou encore les codes Turbo. La technique FEC est généralement utilisée dans la couche

physique mais il est possible de l'appliquer dans d'autres couches comme la couche transport ou la couche application [22].

Comme la retransmission des paquets est coûteuse et même parfois impossible pour les communications interactives lorsque le délai du réseau est trop élevé, plusieurs méthodes FEC ont été standardisées et adaptées pour être utilisées dans les applications RTP [23–25]. Nous détaillons seulement la méthode FEC basée sur les codes de parité définie dans le RFC 5109 parce que nous allons l'utiliser dans l'implémentation de notre algorithme. Nous avons choisi l'utilisation de ce type de FEC pour plusieurs raisons. La raison principale est que les codes de parité sont les codes correcteurs les moins coûteux en matière de charge de calcul. De ce fait, l'utilisation du RFC 5109 permet d'offrir un délai très faible par rapport aux méthodes définies dans d'autres RFCs, ce qui est une contrainte indispensable dans les communications interactives [23]. Le code correcteur d'erreurs de la méthode FEC du RFC 5109 est basé sur une simple opération de parité. Il est considéré comme l'un des codes correcteurs d'erreurs les plus simples en existence. L'opération de parité est un ou-exclusif ($\oplus$) des bits du flux média.

Le fait de changer un bit unique dans l'opération $\oplus$ changera le résultat obtenu. Ceci permet à un seul bit de parité de détecter seulement l'existence toute erreur singulière mais sans pouvoir la corriger. Cependant, lorsque plusieurs bits de parité sont utilisés, il devient possible de détecter plusieurs erreurs et de les corriger.

Pour l'utilisation de RTP sur UDP/IP où on se soucie de la perte de paquets plutôt que d'erreurs sur des bits, il faut envoyer les bits de parité dans des paquets séparés des données qu'ils protègent. S'il y en a assez de bits de parité, il est possible de les utiliser pour récupérer un contenu complet d'un paquet perdu dans un groupe déterminé. La propriété qui rend cette opération possible est :

$$A \oplus B \oplus B = A, \tag{2.1}$$

pour n'importe quelle valeur de A et B. Si les trois informations A, B et A $\oplus$ B sont envoyés séparément, le récepteur a seulement besoin de deux de ces informations afin de récupérer les valeurs de A ou B. La figure 2.3 montre un exemple dans lequel un groupe de 7 bits perdus est récupéré en utilisant cette méthode. Ce processus peut-être directement appliqué au paquet RTP.

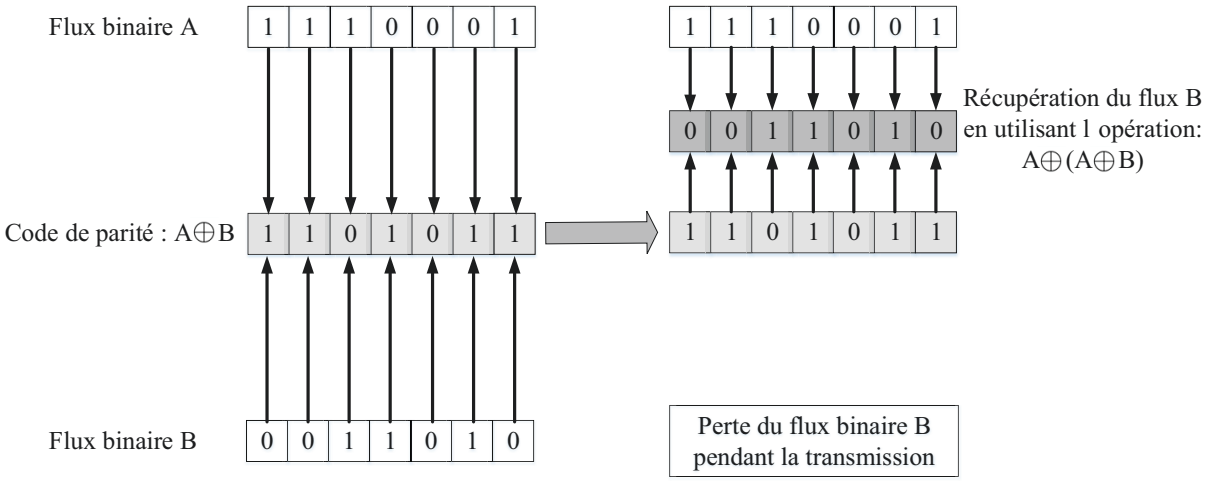


FIGURE 2.3 – Utilisation de la parité entre les bits pour récupérer la perte des données

Ceci est effectué en traitant la totalité du paquet en tant qu'un ensemble de bits et en calculant les paquets de parité qui sont le résultat de l'opération $\oplus$ de tous les paquets de données, voir la figure 2.4. Ces paquets de parité sont utilisés pour reconstruire les paquets RTP perdus. Chaque paquet FEC permet de reconstituer un seul paquet perdu dans un groupe de paquets déterminé. Si le nombre de paquets perdus dans ce dernier est supérieur à 1, il n'est pas possible de les récupérer. Ainsi, pour avoir un bon niveau de protection, il faut utiliser des séquences de paquets de petite taille pour la génération des paquets FEC. Cependant, la diminution de la taille des séquences augmente la bande passante utilisée par les paquets redondants. La protection de paquets en utilisant FEC est donc considérée comme un compromis entre la bande passante disponible et le degré de protection.

2.7 La qualité des communications interactives

Les applications de communications interactives peuvent offrir des qualités très différentes. Il est donc nécessaire d'utiliser une métrique permettant de mesurer la qualité perçue par l'utilisateur. Ainsi, 2 types de mesure permettant d'évaluer la qualité offerte par les services de communications interactives ont été définis : la qualité de service (QoS) et la qualité d'expérience (QoE).

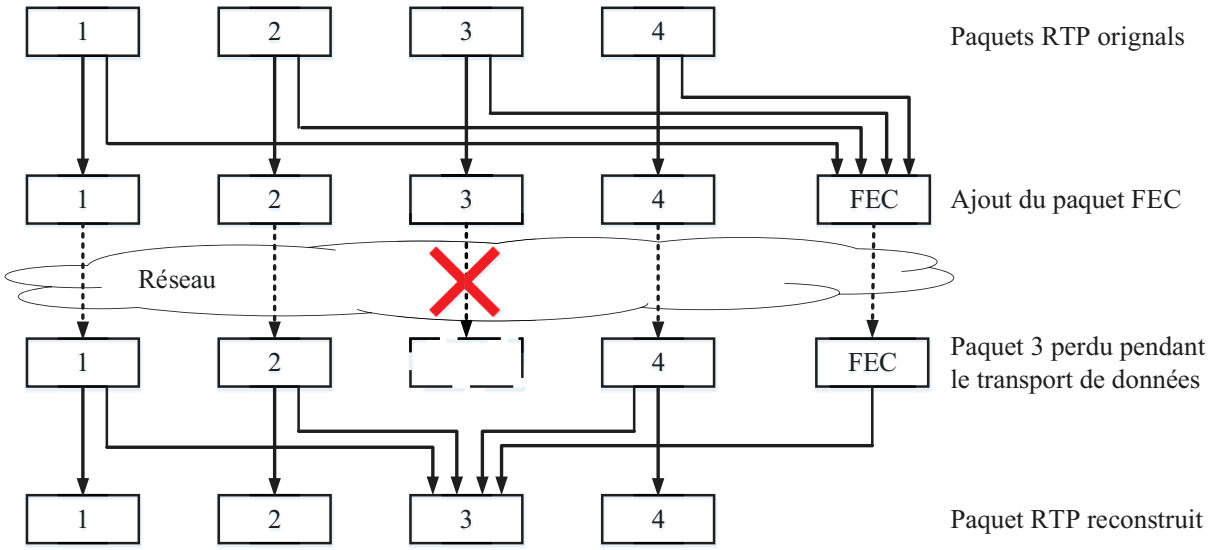


FIGURE 2.4 – Récupération des paquets RTP avec FEC

2.7.1 Qualité de services (QoS)

La qualité de service est la capacité d'un service à répondre par ses caractéristiques aux différents besoins de ses utilisateurs ou consommateurs [26]. Les caractéristiques prises en compte pour déterminer la QoS d'un service sont généralement variables en fonctions du service proposé. D'un point de vue technique, la qualité de service dans un réseau IP peut être mesurée objectivement en mesurant les valeurs des facteurs tels que le taux de pertes de paquets, le délai, la gigue. Ces paramètres peuvent se dégrader rapidement lors de la présence d'une congestion. Nous allons étudier ces paramètres en détail dans la section 3.2.

2.7.2 Qualité d'expérience (QoE)

Contrairement à la QoS, la QoE est une mesure subjective. Elle évalue l'adéquation d'un service par rapport aux attentes d'un utilisateur. La valeur de la QoE est très liée à la QoS, mais elle ne peut pas être mesurée seulement à partir des paramètres de QoS. La QoE dépend aussi de plusieurs autres paramètres tels que le degré de satisfaction de l'utilisateur, la disponibilité, la fiabilité et la facilité d'utilisation. Afin de mesurer la QoE d'un service vidéo, on utilise traditionnellement plusieurs observateurs. Ces observateurs utilisent le service pendant une durée déterminée. Ensuite, ils donnent une note qui varie généralement entre 1 et 5, communément appelé Mean Opinion Score (MOS). Cette note reflète le niveau de qualité perçue de la vidéo reçu. La valeur 1 signifie que la qualité de la vidéo est très mauvaise et 5 que la qualité est excellente.

Le niveau de QoE offert par le service est égal à la moyenne des différentes valeurs MOS. Cette technique est très coûteuse en termes de temps et d'argent. Pour cela, plusieurs métriques sont utilisées permettant de donner objectivement une approximation de la valeur de la QoE. Pour les services vidéo, les métriques SSIM et PSNR sont généralement utilisées. La signification et la méthode de mesure de ces métriques sont présentées dans la section 5.3.

Travailler avec la QoE plutôt que la QoS est important dans notre étude. Alors que la QoS peut se dégrader de manière non contrôlée, nous allons chercher à adapter la communication vidéo à un niveau acceptable pour l'usager, donc à préserver la QoE.

2.8 Conclusion

Nous avons présenté dans ce chapitre les difficultés et les défis de l'exécution des applications multimédia interactives sur Internet. Nous avons également détaillé les mécanismes et les technologies permettant de rendre l'exécution de ce genre de services possibles.

Nous avons vu aussi que contrairement à TCP, le protocole RTP de transport des données multimédia n'offre pas de mécanismes de contrôle de congestion. Il laisse le choix aux concepteurs et aux développeurs d'implémenter leurs propres algorithmes selon les besoins du service développé.

Chapitre 3

Contrôle de congestion et adaptation de débit

3.1 Introduction

La congestion du réseau se traduit généralement par des dépassements de la taille des tampons des routeurs, lorsque les noeuds envoient plus de paquets que le réseau ne peut en acheminer sur certain liens. Ceci entraine la dégradation des paramètres de QoS. Afin d'éviter ce phénomène, les émetteurs doivent diminuer le taux de données envoyées avant ou dès son apparition. Ce mécanisme s'appelle le contrôle de congestion. Plusieurs algorithmes de contrôle de congestion ont été développés pour TCP, comme TCP New Reno [27], TCP BIC [28], TCP CUBIC [29], etc. Ce n'est par contre pas possible pour UDP et les applications basées sur ce protocole doivent implémenter leurs propres algorithmes de contrôle de congestion qui permettent de préserver un niveau de QoE adéquat malgré la dégradation de la QoS. Dans les applications de communication vidéo interactive, l'émetteur peut adapter le débit de la vidéo envoyée aux conditions du réseau. En d'autres termes, l'algorithme augmente le taux de compression de la vidéo pour diminuer la bande passante utilisée, ce qui permet d'éviter la congestion.

Les services de communications interactives sur Internet utilisent de nos jours des algorithmes de congestion propriétaires divers qui n'offrent pas une QoE optimale. Pour améliorer la QoE et standardiser les algorithmes de contrôle de congestion, le groupe RTP Media Congestion Avoidance Techniques (RMCAT) de l'IETF a été créé en 2013. Le but de ce groupe est de concevoir un algorithme de contrôle de congestion

standard pour les communications interactives qui permet d'offrir une QoE optimale pour des conditions réseaux données.

Dans ce chapitre, nous présentons les différents paramètres de QoS qui sont généralement affectés par la congestion. Ensuite, nous présentons les mécanismes de contrôle de congestion et d'adaptation de débit utilisé en détaillant deux algorithmes récents définis par le groupe RMCAT.

3.2 La congestion et ses effets sur les paramètres de QoS

La congestion réseau se produit lorsqu'un lien ou un nœud reçoit une quantité de données qui dépasse sa capacité de traitement. Ainsi, la file d'attente du lien ou du nœud se remplit. Les nouveaux paquets qui arrivent et ne trouvent pas de place dans la file d'attente sont rejetés. Ceci peut causer une énorme dégradation dans la QoS.

Les paramètres principaux de QoS pour la communication vidéo interactive sont : le délai de bout en bout, la gigue, la bande passante disponible, le taux de perte de paquets.

3.2.1 Délai

Les paquets se déplaçant d'un hôte vers un autre sont sujets à des délais sur des liens réseau. Ces délais peuvent être causés par plusieurs obstacles le long du réseau. Certains délais sont trop insignifiants pour être pris en considération. D'autres, par contre, prennent plus d'importance selon la longueur des paquets ou la longueur du lien de la source à la destination. Il existe plusieurs types de délais [30] :

- Délai de traitement (D_t) : C'est le temps nécessaire pour analyser l'entête du paquet IP et déterminer où il faut le transmettre.
- Délai de transmission (D_{tr}) : C'est le temps nécessaire pour transmettre l'entièreté d'un paquet sur un lien de communication. Ce délai est représenté par le rapport entre le nombre de bits du paquet et la vitesse du lien (Q bits/S bps), où Q est la longueur du paquet et S est la vitesse de transmission du lien.
- Délai de propagation (D_{pr}) : C'est le temps nécessaire pour qu'un bit transmis atteigne l'autre extrémité du lien de communication. Ce délai est représenté par le rapport entre la longueur du lien et la vitesse de propagation de l'information dans le médium.

– Délai de mise en file d'attente (D_a) : C'est un délai variable, qui peut prendre beaucoup d'importance dans le réseau. Ce délai est principalement causé par le temps pris pour transmettre des paquets en attente sur un lien de communication. Les nouveaux paquets qui arrivent au routeur doivent attendre la fin de la transmission du ou des paquets précédents pour être transmis à leur tour sur la même route.

3.2.2 Le délai de bout en bout et le temps d'aller-retour

Le délai de bout en bout, où « One-Way Delay » (OWD), est le temps nécessaire à un paquet pour aller de la source à la destination. Il comprend les délais de propagation, les délais de transmission et les délais de mise en file d'attente de tous les nœuds intermédiaires (routeurs et commutateurs) [30]. Pour avoir un bon niveau d'interactivité dans les applications de communication vidéo, l'OWD doit être inférieur à 150 ms. Le niveau d'interactivité reste acceptable lorsque l'OWD se trouve dans l'intervalle 150 ms à 400 ms. Un OWD supérieur à 400 ms n'est pas acceptable pour les applications de communication interactive. À cause des problèmes de synchronisation d'horloge, il est difficile de calculer l'OWD en pratique [31]. Pour cela, on utilise généralement le temps aller-retour, qui est plus simple à mesurer, pour calculer le délai de bout en bout dans un réseau. Le temps aller-retour, connu sous le nom de « Round-Trip Time (RTT) » est le délai de la source à la destination et de la destination à la source. Pour calculer l'OWD, il suffit d'appliquer la formule suivante :

$$OWD = \frac{RTT}{2}. \tag{3.1}$$

Notons que la valeur ainsi obtenue du OWD est une valeur approximative, car le paquet peut utiliser un lien de retour différent du celui de l'arrivée [32].

3.2.3 La gigue

La variation du délai cause une fluctuation dans le temps d'arrivée séparant les paquets. Cette fluctuation s'appelle la gigue. Une gigue élevée peut causer une forte dégradation de la qualité des applications multimédia interactives. Une mémoire tampon au niveau du récepteur, communément appelée « Jitter Buffer » , est généralement utilisée pour minimiser la variation des paquets et les réordonner, le cas échéant.

3.2.4 Bande passante disponible

La bande passante est la capacité maximale du canal de communication. Elle définit le débit maximal auquel les paquets peuvent être transmis par le lien [30]. La bande passante disponible sur un lien i est égale à :

$$B_i = Ci - Xi, \tag{3.2}$$

où, C_i est égal à la capacité du lien i et X_i est égal à l'intensité du trafic passant par ce lien. Le lien ayant la plus petite valeur de bande passante disponible d'un chemin s'appelle le goulot d'étranglement et il définit la bande passante disponible de tout le chemin. Ainsi, la bande passante disponible sur un chemin, composé de N liens, entre un émetteur et un récepteur peut être exprimée par la bande passante minimale de tous les liens :

$$B = \underset{i=1,\ldots,N}{min}(C_i - X_i). \tag{3.3}$$

3.2.5 Perte de paquets

Lorsque la file d'attente d'un nœud réseau se remplit, les paquets peuvent être rejetés à cause de la technique de gestion utilisée dans la file d'attente. Il existe plusieurs techniques de gestion de file d'attente. La plus simple et la plus utilisée dans le réseau Internet s'appelle « Drop Tail ». Dans cette technique, lorsque la file d'attente se remplit jusqu'au niveau maximal, les nouveaux paquets arrivant sont rejetés jusqu'à ce que la file d'attente ait assez de place pour accepter de nouveaux paquets. Les paquets peuvent aussi être perdus lorsqu'il y a une collision. Dans un système de communication vidéo interactive, les paquets peuvent être aussi rejetés par la mémoire tampon de réception lorsqu'ils arrivent trop tard. La perte de paquets peut causer une dégradation énorme dans la qualité de service, même si elle n'est pas trop élevée. Pour cela, il faut éviter et de la minimiser en utilisant les techniques décrites dans la section 2.6.2. Lorsqu'on utilise le mécanisme FEC, plusieurs paquets perdus peuvent être récupérés en utilisant le code correcteur d'erreur. Parfois, il n'est pas possible de récupérer tous les paquets. Les taux de perte de paquet après application du FEC s'appellent le taux de perte de paquets résiduels.

3.3 Algorithmes de contrôle de congestion

La congestion augmente considérablement le taux de perte de paquets, le délai de bout en bout et la gigue. Il est donc nécessaire de l'éviter pour offrir une bonne QoE utilisateur. Pour cela, plusieurs algorithmes de contrôle de congestion ont été développés. Les algorithmes qui ont eu un grand succès sont ceux développés pour TCP [27–29].

À cause de leurs différentes exigences, les applications multimédias interactives doivent être implémentées en utilisant RTP sur UDP. Puisque UDP ne fournit pas le contrôle de congestion, les applications RTP ont implémenté leurs propres algorithmes de contrôle de congestion dans la couche application. Ces algorithmes peuvent être groupés en 4 catégories différentes [33] : disjoncteur humain (« human circuit breaker »), disjoncteur (« circuit breaker »), algorithme propriétaire et algorithme standard expérimental.

3.3.1 Disjoncteur humain

Malgré la nécessité de l'implémentation d'un algorithme de contrôle de congestion indiqué dans la spécification du protocole RTP et ces profils AVP et AVPF, certaines applications multimédias ne les implémentent pas. Pendant la conception de ces applications, les développeurs prennent en compte que les utilisateurs vont arrêter manuellement les flux média ayant une qualité détériorée à cause d'une congestion. Le disjoncteur humain offre donc une QoE médiocre.

3.3.2 Disjoncteur

C'est une forme simple de contrôle de congestion. En effet, après l'observation d'une augmentation importante du taux de pertes pour une période de temps significative, certaines applications arrêtent la transmission des données audio et/ou vidéo. Les disjoncteurs permettent d'éviter de causer de graves congestions, mais ils causent une grande dégradation dans la qualité de l'expérience utilisateur. Cependant, l'arrêt automatique des flux multimédias est préféré à l'arrêt manuel qui nécessite l'intervention de l'utilisateur comme dans les disjoncteurs humain. Le document [34] définit un exemple de disjoncteur RTP.

3.3.3 Algorithme de contrôle de congestion non standard

Les applications de communication connues, comme Skype, utilisent des algorithmes de contrôle de congestion propriétaires. Ces algorithmes sont parmi les principales clefs du succès de ce genre d'application, car ils permettent d'offrir une QoE acceptable.

3.3.4 Algorithme de contrôle de congestion standard expérimental

Certaines applications expérimentales utilisent des algorithmes standardisés qui n'ont pas été beaucoup utilisés pour plusieurs raisons. Parmi ces algorithmes, citons TFRC. TFRC est un protocole de contrôle de congestion qui a été défini pour la première fois en 2003 dans le RFC 3448 [35]. La dernière mise à jour de TFRC est définie dans le RFC 5348 [36]. Ce protocole peut être implémenté en utilisant les informations se trouvant dans les rapports de feed-back standard de RTP. Cependant, TFRC nécessite un feed-back pour chaque paquet [37]. Ceci peut causer une augmentation ou une diminution du débit du flux média dans un intervalle de temps très court (dents de scie) [38, 39]. Par conséquent, TFRC n'offre pas une QoE optimale [40].

Le groupe RMCAT IETF a été créé afin de proposer un algorithme de contrôle de congestion standard, qui offre une très bonne QoE pour les communications interactives. Il est prévu que le processus de création de ce standard va durer plusieurs années [41]. À la date de l'écriture de ce mémoire, les membres de RMCAT ont défini les spécifications du standard [42] et un ensemble de critère d'évaluation [43]. Ils ont aussi proposé 3 algorithmes expérimentaux de contrôle de congestion : NADA [44], GCC [45] et FBRA [46, 47]. Dans les sections suivantes nous détaillons seulement le fonctionnement des algorithmes GCC et FBRA. Le but de ce mémoire est de proposer une solution qui ne nécessite pas la modification du réseau utilisé. Pour cela, nous ne détaillons pas NADA car il est basé sur une extension optionnelle des protocoles TCP et IP, la notification explicite de congestion (ECN).

3.3.5 L'algorithme Google Congestion Control (GCC)

Afin d'améliorer la qualité d'expérience offerte par leur implémentation de la norme WebRTC, détaillée dans la section 5.2.1, Google a conçu un nouvel algorithme de contrôle de congestion qui permet d'adapter dynamiquement le débit de la vidéo aux conditions du réseau. Cet algorithme est appelé Google Congestion

Control (GCC) et il est décrit dans [45]. Il est composé de deux parties : une partie de contrôle côté récepteur et une partie de contrôle côté émetteur. La partie de contrôle côté récepteur utilise le temps d'inter-arrivée des paquets et un filtre de Kalman afin d'estimer le débit d'envoi. La partie de contrôle côté émetteur utilise l'équation TFRC [35] pour estimer le débit d'envoi en se basant sur le taux de perte, RTT et le nombre d'octets envoyé.

Partie de contrôle côté récepteur

La partie de contrôle côté récepteur est composée de 3 modules : un filtre d'arrivée, un détecteur de sur-utilisation (over-use detector) et un contrôleur de débit distant.

Filtre d'arrivée L'algorithme de la partie de contrôle côté récepteur utilise un filtre adaptatif qui met à jour l'estimation des paramètres du réseau en continu en se basant sur le temps d'arrivée des trames reçues. Ce filtre est appliqué sur le temps d'inter-arrivée $d(i)$:

$$d(i) = (t(i) - t(i-1)) - (T(i) - T(i-1)), \tag{3.4}$$

où $t(i)$ est le temps d'arrivée du paquet i et $T(i)$ est l'horodatage du paquet i (i.e. le moment où le paquet a quitté l'émetteur). Le temps t_s pour envoyer une trame de taille L sur un lien avec une capacité C peut être mesurée en utilisant l'équation suivante :

$$t_s = \frac{L}{C}. \tag{3.5}$$

En utilisant l'équation 3.5, le temps d'inter-arrivée peut être modélisé comme suit :

$$d(i) = \frac{L(i) - L(i-1)}{C} + w(i) = \frac{dL(i)}{C} + w(i), \tag{3.6}$$

où $w(i)$ est une variable aléatoire du processus stochastique W, qui est une fonction de la capacité C, du trafic en transit $X(i)$ et du débit d'expédition $R(i)$. W est modélisé pour un bruit blanc gaussien. Si le lien est surutilisé, $w(i)$ doit augmenter, et si la file d'attente du goulot d'étranglement est en train de se vider, $w(i)$ va diminuer; sinon la moyenne de $w(i)$ sera égale à 0. En séparant la moyenne $m(i)$ de $w(i)$, il est possible de modéliser le temps d'inter-arrivée comme suit :

$$d(i) = \frac{dL(i)}{C} + m(i) + v(i). \tag{3.7}$$

L'équation 3.7 est l'équation finale utilisée pour modéliser le temps d'inter-arrivée. Les paramètres $d(i)$ et $dL(i)$ sont disponibles pour chaque trame $i > 1$. Les paramètres $C(i)$ et $m(i)$ sont nécessaires pour estimer si la bande passante disponible actuelle est surutilisée ou non. Afin d'estimer ces paramètres, l'algorithme de la partie de contrôle côté récepteur utilise un filtre de Kalman.

Détecteur de sur-utilisation Après avoir estimé les paramètres $C(i)$ et $m(i)$, l'algorithme utilise un module appelé détecteur de sur-utilisation (over-use detector) pour détecter si la bande passante disponible est sur- ou sous- utilisée. Ceci est effectué en comparant la valeur estimée de $m(i)$ avec un seuil γ_1. Lorsque $m(i) > \gamma_1$, le détecteur de sur-utilisation indique que la bande passante disponible est surutilisée et il indique que la bande passante disponible est sous-utilisée lorsque $m(i) < -\gamma_1$. Sinon, le détecteur est maintenu à l'état normal.

Contrôleur de débit distant Le contrôleur de débit distant utilise les informations provenant du détecteur de sur-utilisation pour adapter le débit de la vidéo. Il est conçu pour augmenter la valeur estimée de la bande passante du côté récepteur ($\hat{A}_r$) tant que l'état normal du détecteur est détecté. Cela permet d'assurer que l'algorithme va atteindre, tôt ou tard, la bande passante disponible et va détecter une sur-utilisation. Dès la détection d'une sur-utilisation, l'algorithme réduira la valeur estimée de la bande passante disponible. Le contrôleur de débit distant est exécuté périodiquement. Il contient 3 états :

- L'état Increase : aucune congestion n'est détectée.
- L'état Decrease : une congestion est détectée.
- L'état Hold : attente que les files d'attente se vident avant d'aller à l'état Increase.

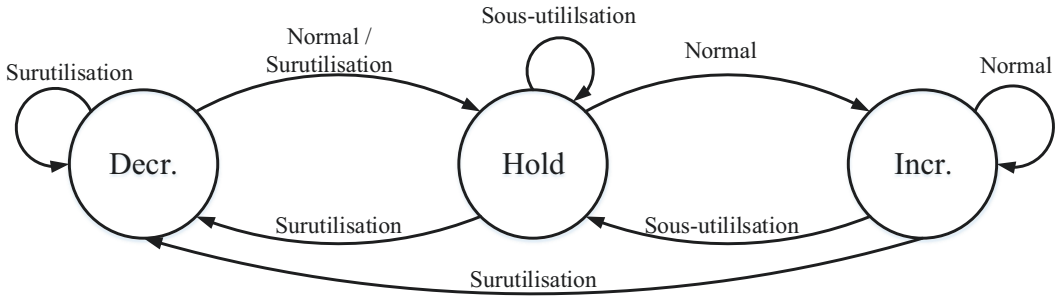


FIGURE 3.1 – Machine à états finis de GCC [48]

La figure 3.1 présente la machine à états finis du contrôleur. Ce dernier commence à l'état Increase et y reste, jusqu'à la détection d'une sur- ou sous-utilisation. Dans cet état, l'algorithme augmente la valeur estimée de

la bande passante disponible en la multipliant par un facteur $eta()$.

$$\hat{A}_r(i) = eta()\hat{A}_r(i-1), \tag{3.8}$$

où $eta()$ est une fonction de plusieurs paramètres du système, détaillés dans [45]. Lorsqu'une sur-utilisation est détectée, le système passe à l'état Decrease. Dans cet état, l'algorithme réduit la valeur estimée de la bande passante en utilisant l'équation suivante :

$$\hat{A}_r(i) = \alpha\hat{R}, \tag{3.9}$$

où $\hat{R}$ est le débit des paquets reçus mesuré pendant une fenêtre de T secondes et α est une constante, typiquement choisie dans l'intervalle $[0.8, 0.955]$.

Le contrôleur de débit distant est aussi responsable de l'envoi des messages de feed-back vers l'émetteur. Ces messages sont les rapports de feed-back usuels de RTCP et des rapports supplémentaires permettant d'envoyer la valeur de la bande passante estimée. Pour envoyer la valeur de la bande passante vers l'émetteur, le contrôleur peut utiliser les rapports de feed-back Temporary Maximum Media Stream Bit Rate Requests (TMMBR) du profil AVPF défini dans le RFC 5104 [49]. Il peut aussi utiliser les rapports de feed-back REMB définis dans [50]. Les rapports de feed-back sont généralement envoyés périodiquement. Cependant, si la valeur de la bande passante estimée change fortement, le contrôleur envoie un nouveau rapport avant que la période ne s'écoule.

Partie de contrôle côté émetteur

La bande passante estimée par le récepteur est fiable seulement lorsque les files d'attente le long du chemin ont une taille assez large. Si les files d'attente sont courtes, la sur-utilisation est seulement visible lorsqu'il y a des pertes de paquets. Puisque le taux de perte de paquets n'est pas utilisé dans l'algorithme de la partie de contrôle côté récepteur, l'émetteur utilise un mécanisme d'adaptation de débit supplémentaire. Pour estimer la bande passante disponible, il se base sur le RTT, le taux de perte et la bande passante disponible estimée par la partie de contrôle côté récepteur. L'algorithme est exécuté pour chaque rapport reçu. Il estime

la bande passante disponible côté émetteur ($A_s(i)$) en utilisant le taux de perte de paquets p :

$$A_s(i) = \begin{cases} A_s(i-1) \times (1 - 0.5p) & p > 0.10 \\ A_s(i-1) & 0.02 \leq p \leq 0.1 \\ 1.05(A_s(i-1) + 1 \text{ Kb/s}) & p < 0.02. \end{cases} \tag{3.10}$$

La nouvelle valeur estimée $\hat{A}_s(i)$ est limitée par deux valeurs : la valeur estimée de la bande passante côté récepteur $\hat{A}_r(i)$ et la valeur déterminée en utilisant la formule de contrôle de débit de TFRC [36].

Critiques

L'algorithme GCC utilise deux algorithmes de contrôle de congestion, le premier est basé sur le délai et le deuxième est basé sur la perte de paquets. L'utilisation de ces deux algorithmes offre une bonne adaptation et un bon niveau d'utilisation de la bande passante disponible. Le GCC a aussi quelques inconvénients. L'inconvénient majeur est le fait qu'un flux vidéo contrôlé par GCC ne peut pas fournir une bonne QoE lorsqu'il partage un goulot d'étranglement de capacité limitée avec un autre flux TCP de longue durée. En effet, dans un scénario pareil, le flux TCP utilise la majorité de la bande passante et le flux vidéo GCC utilise une bande passante très limitée [48]. Le deuxième inconvénient de GCC est qu'il offre de mauvaises performances lorsqu'il est utilisé dans un réseau ayant une bande passante disponible qui varie rapidement, comme les réseaux cellulaires ou les points d'accès Wi-Fi. De plus, lorsque 2 flux GCC partagent le même goulot d'étranglement, l'algorithme se comporte d'une façon imprévisible et présente un mauvais taux d'équité [48]. La performance de GCC se dégrade aussi énormément lorsque le délai de bout en bout dépasse 200 ms [51].

3.3.6 L'algorithme FEC-based Rate Adaptation (FBRA)

Description

FBRA est un algorithme de contrôle de congestion pour les communications interactives sur Internet. Il est basé sur l'utilisation du FEC [47]. L'idée générale de cet algorithme est d'activer un trafic redondant (FEC) à côté du flux média et de l'utiliser en tant que sonde pour estimer la bande passante disponible. Si les conditions du réseau sont bonnes et stables après l'activation du trafic FEC, l'algorithme augmente le débit vidéo d'une quantité égale au débit du trafic FEC ajouté et annule le trafic FEC. L'avantage de cette méthode est que l'algorithme peut être plus agressif pendant la phase de sonde pour la bande passante disponible,

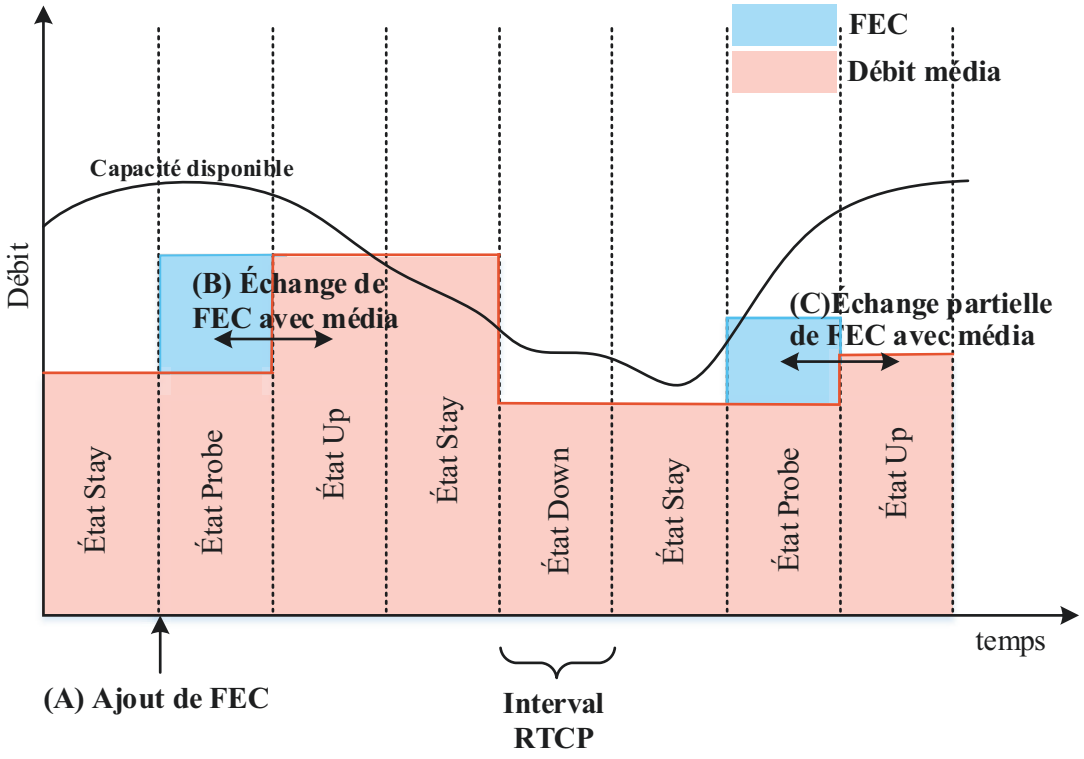


FIGURE 3.2 – Le concept de l'utilisation de FEC pour l'adaptation du débit [47]

car le trafic FEC permet de compenser les pertes de paquets causées par la surexploitation du lien. La figure 3.2 illustre le concept de l'utilisation du FEC pour l'adaptation de débit : lorsque les indices de congestion indiquent l'absence de congestion, le flux FEC est activé afin de sonder pour plus de bande passante. Si le RTCP RR suivant signale des pertes de paquets, l'algorithme réduit le débit vidéo; sinon il l'augmente. L'algorithme contient 4 états :

- L'état Stay : l'algorithme ne génère pas de paquets FEC et ne modifie pas le débit vidéo.
- L'état Probe : l'algorithme maintient le débit vidéo constant, mais envoie des paquets FEC en parallèle au flux vidéo. Si les rapports RTCP RR montrent des signes de congestion, l'algorithme désactive le trafic FEC et passe à l'état Stay. Dans le cas contraire, il passe à l'état Up.
- L'état Up : l'algorithme augmente la qualité du flux vidéo en remplaçant le trafic FEC par un trafic vidéo additionnel. Si une congestion est détectée, l'algorithme passe à l'état Down. Sinon, il passe à l'état Stay.
- L'état Down : l'algorithme réduit le taux d'envoi du trafic vidéo. Si le rapport RTCP RR suivant ne rapporte pas une congestion, l'algorithme passe à l'état Stay. Sinon, il reste dans l'état Down.

Critiques

Dans [47], M. Nagy et al. ont montré que FBRA donne de bonnes performances. L'avantage de cet algorithme est qu'il offre un meilleur taux d'utilisation de la bande passante disponible. Il est aussi moins sensible à la variation rapide de la bande passante. Pour cela, cet algorithme peut offrir une meilleure QoE dans les réseaux cellulaires et les réseaux ayant un accès partagé entre plusieurs utilisateurs. L'utilisation de FEC en tant que sonde a apporté des avantages à FBRA, mais aussi plusieurs inconvénients. FEC est le seul mécanisme existant qui permet d'augmenter la fiabilité des trafics média sans causer une augmentation remarquable dans le délai de bout en bout. Le fait de l'utiliser en tant que sonde uniquement diminue la fiabilité du flux vidéo. De ce fait, l'algorithme offre une mauvaise QoE lorsqu'il partage un goulot d'étranglement d'une capacité limitée avec un ou plusieurs flux TCP. Lorsque la bande passante est très limitée, le trafic TCP concurrent peut causer un taux élevé de pertes dans le trafic vidéo qui oblige FBRA à passer à l'état Down et y rester jusqu'à la terminaison du flux TCP concurrent. Ceci causera une dégradation importante dans la QoE offerte.

3.4 Conclusion

Afin d'améliorer la QoE offerte par les applications de communications vidéo interactives sur Internet, il faut développer un algorithme très performant d'adaptation de débit aux conditions du réseau. L'importance d'un tel algorithme est devenue critique en raison de l'augmentation fulgurante du nombre d'utilisateurs de services multimédias interactifs sur Internet. Pour cela, le groupe RMCAT de l'IETF a été créé. Ce groupe a entrepris la création de quelques algorithmes expérimentaux. Ces algorithmes n'offrent pas les performances souhaitables. Chaque algorithme a ses propres inconvénients et avantages. Dans le prochain chapitre, nous proposons un nouvel algorithme d'adaptation de débit. Pendant sa conception, nous avons pris en comptes les critères d'évaluations proposées par RMCAT. Ceci nous a permis de créer un algorithme offrant une bonne QoE même si la bande passante disponible est très limitée. Notre algorithme n'a pas les inconvénients des deux algorithmes présentés.

Chapitre 4

Algorithme proposé

4.1 Introduction

Afin d'améliorer la QoE offerte par les applications vidéo interactives sur Internet, il est nécessaire de développer un algorithme très performant d'adaptation du débit aux conditions du réseau. Comme nous venons de le montrer, le groupe RMCAT de l'IETF a entrepris l'étude de plusieurs algorithmes expérimentaux. Ces algorithmes n'offrent pas, à ce point, les performances souhaitables: chaque algorithme a ses propres inconvénients et avantages.

Dans ce chapitre, nous proposons un nouvel algorithme d'adaptation de débit. Pendant sa conception, nous avons pris en compte les critères d'évaluations et les spécifications proposées par RMCAT. Ceci nous a permis de créer un algorithme offrant une bonne QoE, même si la bande passante disponible est très limitée, et qui évite les inconvénients des propositions actuelles du RMCAT.

4.2 Présentation générale

Le principe directeur de l'algorithme que nous proposons est d'ajouter un trafic redondant parallèle au trafic vidéo. Si le trafic ne cause pas de perte de paquet et/ou une augmentation de délai, l'algorithme remplace le trafic redondant par un trafic vidéo utile. Pour ne pas provoquer une congestion soudaine, qui peut dégrader la QoE de la communication, l'algorithme doit ajouter un faible taux de trafic redondant et l'aug-

menter progressivement. Si une perte de paquets est détectée, l'algorithme réduit ou annule le taux du trafic redondant. Si la quantité du trafic redondant atteint un niveau prédéfini, l'algorithme remplace ce trafic par un trafic vidéo afin d'améliorer la QoE. On pourrait imaginer que l'amélioration directe de la qualité vidéo est plus efficace que notre technique mais ceci n'est pas correct, spécialement pour les réseaux ayant des variations rapides et brusques dans la quantité de la bande passante disponible tels que les réseaux cellulaires et les réseaux d'accès partagés entre plusieurs utilisateurs (p. ex. points d'accès public, connexion Internet dans une entreprise ou une université). En effet, dans ce type de réseaux, l'algorithme doit adapter les paramètres de l'encodeur très rapidement pour répondre à la variation de la bande passante disponible. Cette variation rapide de qualité à pour effet de dégrader la QoE [40]. Pour éviter cette dégradation, les applications de communication vidéo interactive existantes qui améliorent la QoE en changeant directement les paramètres du codec vidéo essayent de ne pas dépasser une limite largement inférieure à la bande passante réellement disponible. En utilisant notre algorithme, il est possible d'utiliser plus de bande passante sans causer de variations rapides dans la qualité de la vidéo. Ceci permet donc d'offrir une meilleure QoE. Lorsqu'il n'y a pas assez de bande passante disponible, l'ajout du trafic redondant peut perturber le flux vidéo et causer des pertes de paquets. Pour cela, il est nécessaire de choisir un trafic redondant qui permet de compenser la perte de paquets en utilisant une quantité limitée de bande passante. Par conséquent, nous choisissons le trafic FEC, présenté dans la section 2.6.2, en tant que trafic redondant.

Notons que notre algorithme a quelques similarités avec l'algorithme FBRA présenté dans la section 3.3.6, mais il contient aussi plusieurs différences qui lui permettent d'offrir une meilleure QoE. L'algorithme FBRA utilise le FEC pour estimer la bande passante disponible et améliorer la qualité vidéo suivant cette estimation. Notre algorithme effectue une opération similaire, mais il permet aussi d'utiliser plus de bande passante et de garantir une bonne QoE même si la bande passante disponible est très faible. L'augmentation de la proportion de FEC provoque une augmentation du taux de perte de paquets dans le réseau lorsqu'il n'y a pas assez de bande passante disponible. Cette augmentation du taux de perte oblige les autres flux concurrents à diminuer leur débit. Cependant, le flux vidéo peut conserver le même débit utilisé, car les paquets perdus sont recréés grâce au trafic FEC. Cette technique permet, du moins en principe, d'améliorer la qualité de la vidéo tout en laissant une proportion de bande passante acceptable pour les autres flux concurrents. L'algorithme FBRA améliore la qualité de la vidéo en augmentant un seul paramètre, le débit de la vidéo. Par contre, notre algorithme change plusieurs paramètres afin d'offrir une meilleure qualité en utilisant moins de bande passante. Pour cette raison, il utilise plusieurs profils prédéfinis.

4.2.1 Profils

Chaque profil est défini par un ensemble de valeurs de paramètres du codec H.264. Les paramètres sont choisis de façon à permettre à chaque profil d'avoir une bande passante et un niveau de qualité spécifique. Nous avons décidé d'utiliser les paramètres résolution, cadence de trame, CRF, Aq-mode et subme (expliqués à la section 2.4) pour définir les différents profils, car ils ont l'influence la plus notable sur la qualité de la vidéo et le débit. Nous avons aussi fixé les valeurs des paramètres restants à des valeurs optimales pour la communication temps réel. Le choix de profils et des paramètres à utiliser pour définir ces profils est basé principalement sur une étude déjà effectuée au sein de notre laboratoire, décrite dans [52] et par une étude similaire effectuée par notre partenaire industriel. Le profil P_1 (ou P_{Min}) est le profil qui offre la plus basse qualité (c.-à-d. qu'il utilise une faible quantité de bande passante) et le profil P_{Max} (= 3000 Kb/s) est le profil qui offre la meilleure qualité (c.-à-d. qu'il utilise une quantité élevée de bande passante). Les débits vidéo des différents profils sont présentés dans l'annexe B.

4.2.2 Description de l'algorithme

Nous présentons ici le concept général de l'algorithme; une description plus détaillée suivra dans la section 4.3. Notre algorithme est exécuté pour chaque RTCP RR reçu et repose sur une machine de 5 états. Pour chaque exécution, il peut choisir entre rester dans le même état ou passer à un autre état. La transition d'un état à un autre est basée sur plusieurs indices de congestion (taux de perte de paquets, taux de perte résiduelle de paquets et RTT) et d'autres conditions, voir section 4.3. Pour simplifier le diagramme et faciliter l'explication générale de l'algorithme, nous utilisons les conditions générales suivantes pour présenter la transition d'un état à un autre :

- **S1** : Les conditions du réseau sont bonnes. La perte des paquets est nulle.
- **S2** : La perte des paquets n'est pas nulle, mais tous les paquets sont récupérés grâce à la charge FEC (c.-à-d. le taux de perte résiduelle de paquets est nul).
- **S3** : Le taux de perte de paquets est faible et le FEC ne permet pas de récupérer tous les paquets (c.-à-d. le taux de perte résiduelle est non nul).
- **S4** : Le taux de perte de paquets est très élevé et l'ajout du trafic FEC ne permet pas de compenser la totalité de la perte.

Une description détaillée de toutes les conditions nécessaires pour passer d'un état à un autre sera présentée dans la prochaine section. La figure 4.1 présente un diagramme d'états qui décrit le concept général de l'algorithme. Notre algorithme contient les 5 états suivants :

- **HOLD** : l'algorithme ne change aucun paramètre. Il conserve les mêmes valeurs des profils vidéo et du taux de FEC. Le taux de FEC est le pourcentage du trafic FEC par rapport au trafic vidéo. l'état HOLD est un état transitoire dans lequel l'algorithme reste jusqu'à la collecte des informations nécessaires lui permettant de passer à un nouvel état.
- **PROBE** : l'algorithme augmente progressivement le taux de FEC jusqu'à la détection d'une congestion ou jusqu'à atteindre un débit de FEC qui peut-être remplacé par un profil ayant une meilleure qualité.
- **PROTECT** : lorsqu'un faible taux de perte est détecté, l'algorithme passe de l'état HOLD à cet état. Dans cet état, l'algorithme augmente graduellement le taux de FEC jusqu'à l'annulation du taux de perte résiduelle. Si le taux de FEC est égal à 100 % et le taux de perte résiduelle reste non nul, ce qui signifie généralement que le taux de perte est élevé (> 15 %), l'algorithme passe à l'état DECREASE.
- **INCREASE** : si les sondes FEC ne permettent pas de détecter une congestion, l'algorithme passe de l'état PROBE à cet état. On passe également de l'état HOLD à cet état si le taux de perte résiduelle reste nul pour une durée bien définie. Dans tous les cas, l'algorithme remplace le trafic FEC par un trafic vidéo afin d'améliorer la qualité de la vidéo.
- **DECREASE** : On passe à cet état lorsque plusieurs paquets sont perdus à cause d'une congestion. Dans cet état, l'algorithme remplace le profil courant par un profil ayant une qualité inférieure pour éviter la congestion et la perte de paquets.

4.2.3 Indices de congestion

Notre algorithme utilise plusieurs indices de congestion pour décider d'une transition vers un nouvel état.

Taux de perte et taux de perte résiduelles

Lorsqu'une congestion se produit, plusieurs paquets sont perdus. En conséquence, l'algorithme doit s'appuyer sur les valeurs de taux de perte de paquets pour passer à l'état DECREASE et éviter la dégradation de la qualité de l'expérience de la vidéo. Dans la pratique, la perte de paquets varie rapidement, voir figure 4.2.

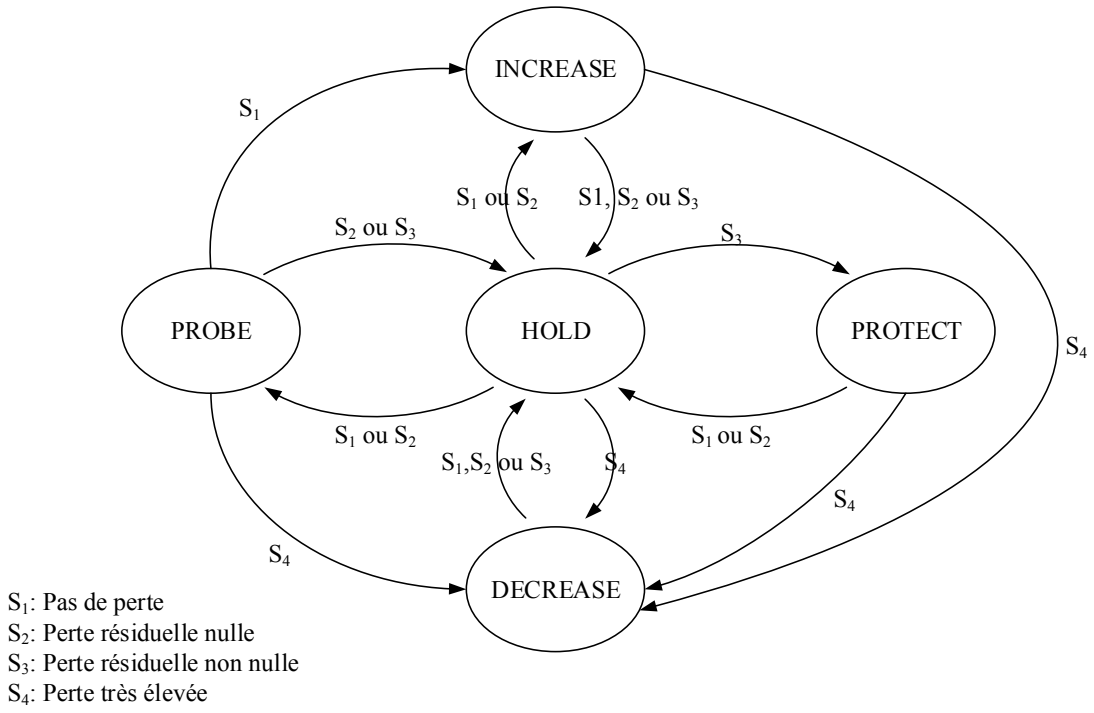


FIGURE 4.1 – Diagramme d'états de l'algorithme

Pour cela, au lieu de compter sur une seule valeur de perte de paquets, l'algorithme doit utiliser la moyenne des 10 dernières valeurs rapportées par les rapports RTCP RR.

Des flux TCP concurrents peuvent aussi causer un faible taux de perte. Il est très simple de distinguer entre le motif de pertes de paquets causées par une congestion sévère et le motif de pertes de paquets causées par les flux TCP concurrents. Lorsqu'il y a congestion, la majorité des rapports RTCP RR indiquent un taux de perte non nul, tandis que quelques-uns d'entre eux indiquent un taux de perte nul. En présence de flux TCP concurrents, la majorité des rapports RTCP RR indiquent un taux de perte nul, tandis que quelques-uns d'entre eux indiquent un taux de perte non nul. La figure 4.2 présente le motif de pertes de paquets lorsque le goodput de la vidéo envoyée est égal à la capacité du goulot d'étranglement. La figure 4.3 présente la capacité du goulot d'étranglement lorsque 2 flux TCP sont en concurrence avec un flux vidéo. Nous choisissons d'utiliser la médiane de la perte au lieu de la moyenne, car elle est moins sensible aux variations brusques et aux événements rares. Ainsi, les pertes, causée par des flux TCP concurrents, se produisent occasionnellement, et la médiane est le plus souvent nulle, où très faible.

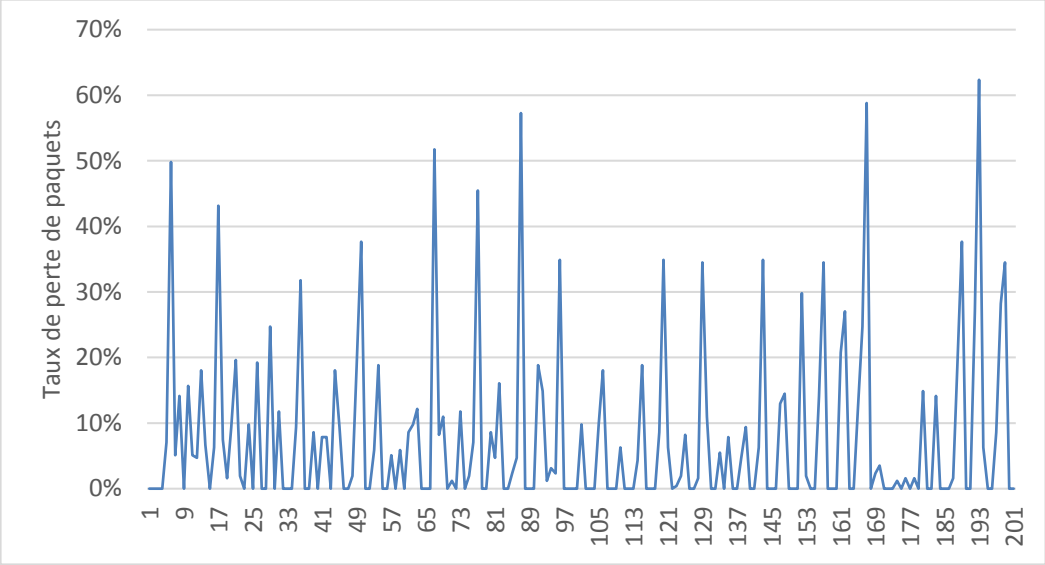


FIGURE 4.2 – Motif de taux de perte (Goodput vidéo = 2 Mb/s, capacité = 2 Mb/s)

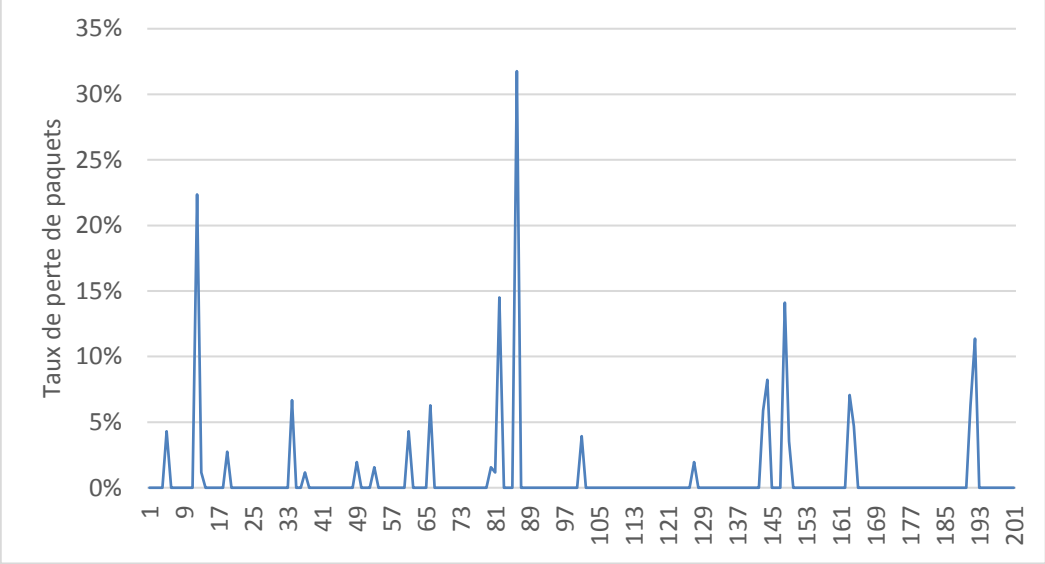


FIGURE 4.3 – Motif de taux de perte à la présence de deux flux de longue durée (Goodput vidéo = 2 Mb/s, capacité = 6 Mb/s)

Puisque le FEC peut compenser la perte de paquets, notre algorithme utilise le taux de perte résiduelle de paquets comme indicateur, voir section 3.2.5. Cet indice de congestion est un indicateur de la nécessité d'augmenter le niveau de FEC utilisé pour la protection. D'une manière similaire au taux de perte, notre algorithme utilise la médiane des 10 dernières valeurs au lieu d'une seule valeur.

RTT

La variation du RTT aide à détecter la congestion avant qu'elle ne cause une perte de paquets. Pour détecter les premiers signes de congestion, l'algorithme calcule la médiane des 10 dernières valeurs de RTT lorsque le taux de perte de paquets est nul. Une fois une nouvelle valeur de RTT reportée, notre algorithme la compare à la médiane mesurée. Si la différence entre la nouvelle valeur de la RTT et la médiane est supérieure à RTT_{Limit}, l'algorithme commence à diminuer le taux de FEC utilisé en tant que sonde.

4.3 Description des états de l'algorithme

Dans cette section, nous donnons une description détaillée de tous les états de notre algorithme.

4.3.1 État HOLD

L'état HOLD est l'état principal de notre algorithme. L'algorithme de la fonction associée à cet état est présenté dans Algorithme 1. C'est un état transitoire dans lequel les paramètres de l'algorithme (taux de FEC et profil) restent constants. L'algorithme reste dans cet état jusqu'à la collecte des informations nécessaires lui permettant de passer à un nouvel état.

algorithme 1 Procédure de l'état HOLD

```
1: function HOLD(params)
2:     if medianResidualLoss = 0 ∧ ¬recentDECREASE then
3:         if timeSinceLastPROTECT > Prot_time ∧ FEC_Protect > 0 then
4:             newState ← INCREASE                          ▷ Passe à l'état INCREASE
5:         else if timesincelastPROBE > ProbeInterval then
6:             if currentProfile < P_max then
7:                 newState ← PROBE                         ▷ Passe à l'état HOLD
8:             else
9:                 FEC_Probe = 0
10:            end if
11:        else
12:            return                                       ▷ Reste dans l'état HOLD
13:        end if
14:    else if medianResidualLoss > 0 ∧ medianLoss < Loss_Limit ∧ FEC_Protect < 1 ∧
       ¬recentDECREASE then
15:        newState ← PROTECT                               ▷ Passe à l'état PROTECT
16:    else if (medianResidualLoss > 0 ∧ FEC_Protect = 1) ∨ medianLoss ≥ Loss_Limit then
17:        newState ← DECREASE                              ▷ Passe à l'état DECREASE
18:    else
19:        return                                           ▷ Reste dans l'état HOLD
20:    end if
21: end function
```

(2) : une médiane de taux de perte résiduelle de paquets nulle signifie qu'il n'y a pas de perte de paquets ou que tous les paquets perdus sont récupérés en utilisant le trafic FEC. L'algorithme doit donc passer à l'état INCREASE ou PROBE afin d'augmenter le débit vidéo utilisé. Un temps de visite de l'état PROTECT supérieur à une limite $Prot_{Time}$ (= 20 s) signifie que les conditions du réseau se sont probablement améliorées. Pour cela, un taux de FEC moins élevé peut suffire pour protéger le flux média en compensant toutes les pertes de paquets. L'algorithme passe donc à l'état INCREASE. Cet état permet de réduire le taux de FEC utilisé pour la protection et change le profil utilisé pour un autre de meilleure qualité tout en gardant la bande passante totale utilisée constante. Si le taux de FEC utilisé pour la protection est nul ou si le temps depuis la

visite de l'état PROTECT est inférieur à $Prot_{Time}$, l'algorithme passe à l'état PROBE. $ProbeInterval$ est l'intervalle séparant deux sondes successives.

(14): une médiane de taux de perte de paquets résiduelle non nulle signifie que le taux de FEC utilisé pour la protection ne suffit pas à compenser tous les paquets perdus. Dans cette situation, l'algorithme passe à l'état PROTECT.

(16): si la médiane de taux de perte de paquets est non nulle et l'algorithme a atteint le taux maximal de FEC (c.-à-d. $FEC_{Protect} = 1$), l'algorithme passe à l'état DECREASE. Si la médiane est très élevée ($> Loss_{Limit} = 0,15$), l'algorithme passe à l'état DECREASE même si $FEC_{Protect}$ est inférieur à 1. En effet, lorsque la perte de paquets est trop élevée, même à un taux maximal, le FEC ne serait pas capable de récupérer tous les paquets perdus. De plus, l'augmentation du FEC lorsque le taux de perte est trop élevé rend la congestion plus grave et augmente la perte de paquets.

Nous avons fixé la valeur de $Loss_{Limit}$ à 0,15, car nous avons remarqué d'après plusieurs tests effectués que lorsque la médiane du taux de perte de paquets est supérieur à 15 %, le trafic FEC (même avec un taux de 100 %) ne permet pas de garantir un taux de perte résiduel nulle. Ainsi, il est plus judicieux de passer directement à l'état DECREASE que d'ajouter du trafic FEC. Il faut noter que cette valeur est spécifique à l'implémentation du RFC 5109 que nous avons utilisée. Elle peut varier dans d'autres implémentations.

L'algorithme passe à l'état DECREASE lorsque les conditions du réseau se sont dégradées. Pour diminuer le risque de création d'une nouvelle congestion, l'algorithme doit attendre un peu de temps dans l'état HOLD avant de passer à un nouvel état différent de DECREASE.

4.3.2 État PROBE

Dans l'état PROBE, l'algorithme augmente progressivement le taux de FEC utilisé en tant que sonde, (FEC_{Probe}). L'augmentation est effectuée seulement lorsque la différence entre le RTT actuel et la médiane des RTT précédents ne dépasse pas RTT_{Limit} (= 50 ms).

Dans nos expériences, nous avons remarqué que le fait de fixer RTT_{limit} à 50 ms permet d'avoir de meilleurs résultats. Avec des valeurs inférieures à cette limite, l'algorithme devient trop sensible à la variation de délai. En conséquence, notre algorithme diminue le taux de FEC utilisé en tant que sonde et évite d'envoyer des sondes lorsqu'il y a une faible variation de délai qui n'est pas due à la congestion. Avec des

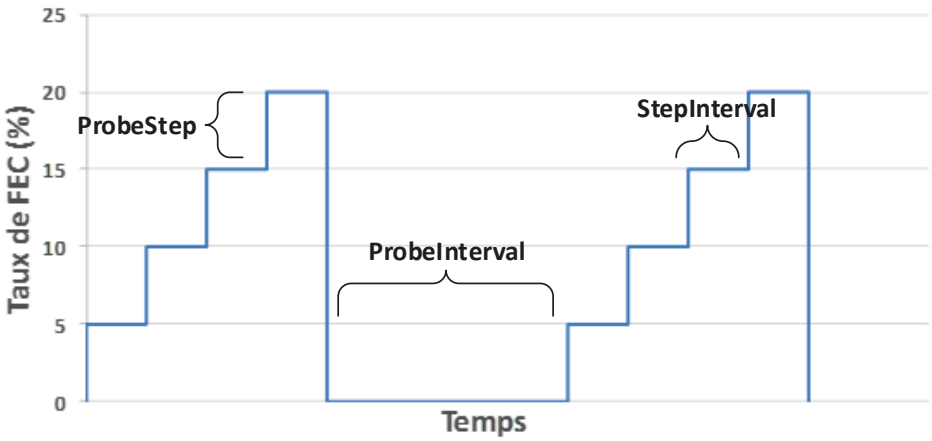


FIGURE 4.4 – Les paramètres des sondes FEC

valeurs supérieures à cette limite, l'algorithme échoue fréquemment à détecter la congestion avant qu'elle se produise. En conséquence, une perte de paquet peut se produire. Pour déterminer RTT_{limit}, nous avons effectué plusieurs tests, chacun utilisant une valeur de RTT spécifique. Ensuite, nous avons conservé la valeur qui permet d'utiliser un taux élevé de FEC et ayant un taux de perte très faible. Il faut noter que le choix de cette valeur tend à être spécifique à la taille des mémoires tampons des nœuds se trouvant tout au long du chemin parcouru. Il est donc nécessaire de changer la valeur de RTT_{Limit} lorsqu'elle est utilisée dans des réseaux ayant des nœuds de grande mémoire tampon comme dans le cas des réseaux cellulaires.

Pour éviter la création d'une congestion sévère qui peut être causée par les sondes, l'augmentation du taux de FEC est effectuée progressivement en utilisant des pas égaux à $probeStep$ ($probeStep = 5\,\%$ du débit vidéo dans le mode $Normal$). À cause du délai du réseau, les indices de congestion indiqués dans des rapports RTCP RR ne signalent pas l'apparition d'une congestion immédiatement. Pour cette raison, l'algorithme attend un laps de temps très court égal à $probeStepInterval$ avant d'augmenter de nouveau le taux du FEC. Après le succès d'une sonde (c.-à-d. le succès du changement du trafic FEC par un trafic vidéo) ou après l'échec d'une sonde, l'algorithme attend un laps de temps égal à $probeInterval$ avant d'envoyer une autre sonde. La figure 4.4 donne une représentation simple des paramètres de la sonde : $probeStep$, $probeStepInterval$ et $probeInterval$. L'état probe peut fonctionner dans les 3 modes suivants :

Le mode *Normal* est le mode le plus utilisé. Dans ce mode l'algorithme envoie des sondes avec un rythme modéré. Les valeurs des paramètres utilisés dans ce mode sont : $probeStep = 5\,\%$, $probeStepInterval =$

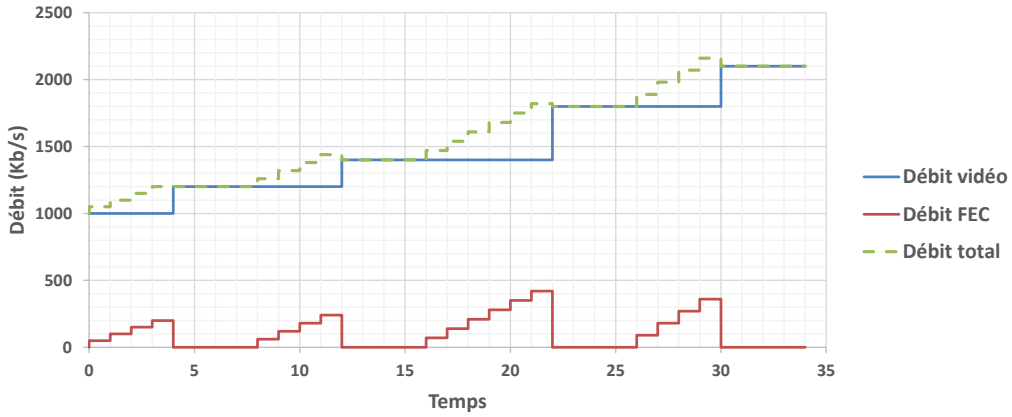


FIGURE 4.5 – Algorithme utilisant le mode probe à l'état *Normal*

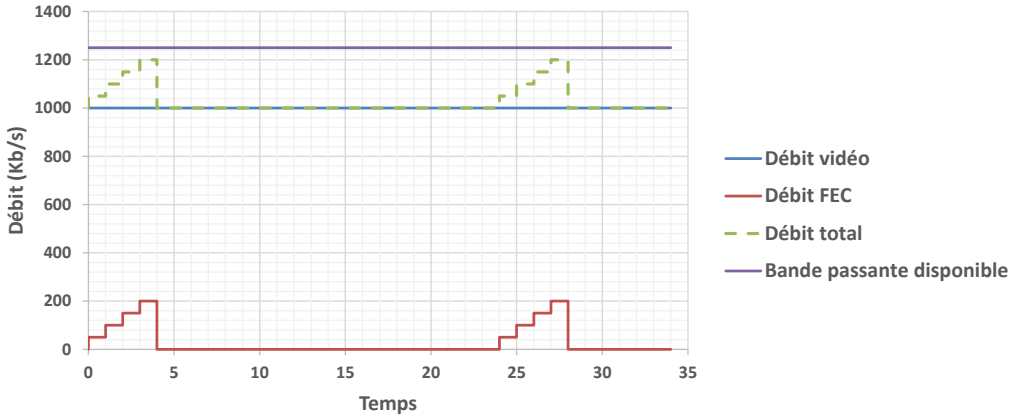


FIGURE 4.6 – Algorithme utilisant le mode prudent à l'état PROBE

$100\,ms$ et $probeInterval = 2000\,ms$. La figure 4.5 présente un exemple du fonctionnement de l'état PROBE dans le mode $Normal$.

Le deuxième mode est le mode prudent (*Slow*). Lorsque la bande passante disponible est très limitée, l'envoi de plusieurs sondes peut augmenter la perte des paquets. Ces paquets peuvent ne pas être récupérés, car le taux de FEC utilisé par les sondes est trop faible. Pour éviter ce problème, il faut réduire le nombre de sondes envoyées, d'où l'utilité du mode prudent. Dans ce mode, l'intervalle entre les différentes sondes est élevé. Il est activé après l'échec de plusieurs sondes successives. Les valeurs des paramètres utilisés dans ce mode sont : $probeStep = 5\,\%$, $probeStepInterval = 100\,ms$ et $probeInterval = 10\,000\,ms$. La figure 4.6 présente un exemple du fonctionnement de l'état PROBE dans le mode prudent.

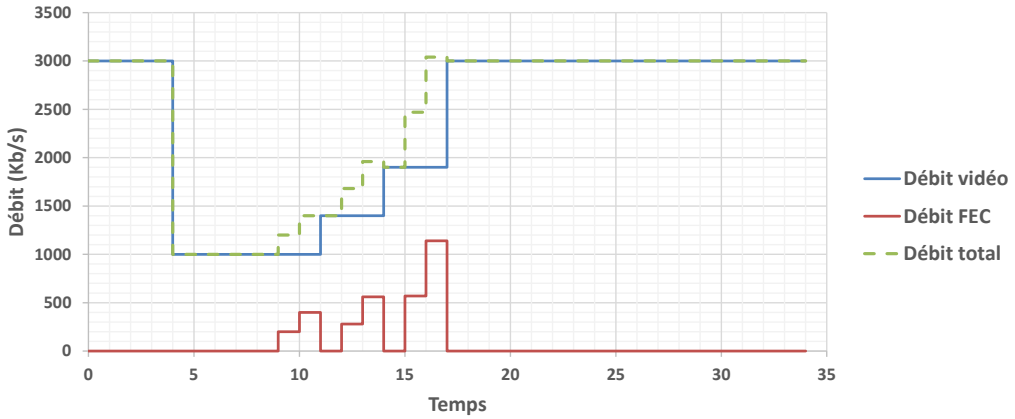


FIGURE 4.7 – Algorithme utilisant le mode agressif à l'état probe

Le troisième mode est le mode agressif (*Fast Recovery*). Généralement, une diminution importante et soudaine de la bande passante est un évènement passager (p. ex. l'utilisateur s'éloigne du point d'accès, or un autre utilisateur commence le téléchargement d'un fichier). Pour cela, après le succès de 3 sondes successives, l'algorithme utilise ce mode pour revenir rapidement au débit vidéo utilisé avant la diminution de la bande passante. Les valeurs des paramètres utilisés dans ce mode sont : $probeStep = 20\,\%$, $probeStepInterval = 50\,ms$ et $probeInterval = 50\,ms$. Ce mode est seulement activé lorsque le débit vidéo a diminué de plus de $40\,\%$. Pendant la phase d'augmentation rapide, l'algorithme désactive le mode agressif et active le mode normal dès que le débit vidéo atteint $80\,\%$ de l'ancien débit (celui utilisé avant la diminution rapide de la bande passante). Ceci permet de diminuer le risque de la création d'une nouvelle congestion au cas où la somme du débit du trafic vidéo et du trafic FEC de la sonde avec, rappelons-le, un probeStep égal à 20 % dépasse la bande passante disponible. Si une sonde échoue pendant le mode agressif, l'algorithme active immédiatement le mode $Normal$. La figure 4.7 présente un exemple du fonctionnement de l'état PROBE dans le mode agressif.

algorithme 2 Procédure de l'état PROBE

1: **function** PROBE(params)
2: **if** successiveFailedProbes > 3 **then**
3: setProbingMode(slow)
4: **end if**
5: **if** rtt $-$ medianRtt $<$ RTT_{Limit} $\wedge$ timeSinceProbeStep $>$ $probeStepInterval$ $\wedge$ timeSinceLastProbe $>$ $probeInterval$ **then**
6: **if** fecProbeBitrate $<$ β **then** $\triangleright$ β = Débit du meilleur profil suivant $-$ Débit du profil courant
7: $FEC_{Probe} \leftarrow FEC_{Probe} + probeStep$
8: **else**
9: newState $\leftarrow$ INCREASE
10: **end if**
11: **else if** rtt $-$ mediaRtt $> RTT_{Limit} \wedge$ packetLoss $= 0$ **then**
12: $FEC_{Probe} \leftarrow FEC_{Probe} -$ probeStep
13: **else if** packetLoss > 0 **then**
14: $FEC_{Probe} \leftarrow 0$
15: successiveFailedProbes $\leftarrow$ successiveFailedProbes $+ 1$
16: **end if**
17: **end function**

Lorsque la somme de la bande passante utilisée par le profil courant et celle du FEC utilisé en tant que sonde est supérieure à la bande passante utilisée par le meilleur profil, l'algorithme passe à ce profil et annule le taux du FEC utilisé en tant que sonde. Une valeur de RTT supérieure à la médiane indique que les files d'attente du point de congestion sont en train de se remplir ce qui est un indicateur d'une congestion possible. Pour cette raison, l'algorithme commence la réduction de FEC_{Probe} progressivement lorsque la différence entre la valeur de la RTT et la valeur $medianRTT$ est supérieure à RTT_{Limit}. Une perte de paquet qui se produit pendant la phase de sonde signifie généralement qu'il n'y a plus de bande passante disponible. Pour cela, si le taux de perte de paquets augmente, l'algorithme annule la charge FEC utilisée pour sonder. Les paquets perdus sont souvent récupérés par le FEC utilisé en tant que sonde. L'algorithme de l'état PROBE de cet état est présenté dans Algorithme 2. Pour simplifier la présentation de l'algorithme, l'activation et la désactivation du mode agressif ne sont pas inclus.

4.3.3 État PROTECT

L'algorithme de l'état PROTECT est présenté dans Algorithme 3. Dans cet état, l'algorithme augmente le taux du FEC de protection ($FEC_{Protect}$). $FEC_{Protect}$ est utilisé pour protéger le flux média contre les pertes passagères qui peuvent être causées par les flux TCP concurrents. L'algorithme augmente le $FEC_{Protect}$ progressivement en utilisant des pas égaux à 5 % du débit vidéo. Un trafic de FEC non nul signifie que l'algorithme a été récemment dans l'état PROBE et qu'il y a une forte probabilité pour que le taux de pertes observé soit causé par les sondes FEC. Pour cette raison, l'algorithme doit annuler le trafic FEC_{Probe} avant d'essayer d'augmenter le trafic $FEC_{Protect}$. Sans cette condition, l'état PROTECT pourrait dégrader la qualité de la communication vidéo au lieu de l'améliorer, particulièrement lorsque la capacité du lien utilisé est limitée.

Prenons un scénario dans lequel la capacité du lien utilisé est égale à 2 Mb/s et l'algorithme utilise un profil ayant un débit presque égal à 2 Mb/s. S'il n'y a pas de perte de paquets, l'algorithme envoie des sondes FEC pour estimer la quantité de la bande passante disponible. Ceci peut causer un faible taux de perte de paquets qui oblige l'algorithme à passer à l'état PROTECT. Dans cet état, l'algorithme augmente la charge du FEC_{Probe}. Ceci provoque une congestion sévère qui oblige l'algorithme à passer à l'état DECREASE et à utiliser un profil ayant une qualité inférieure. Ainsi, lorsque le débit total utilisé (débit vidéo + débit FEC) est presque égal à la capacité du réseau, l'algorithme n'augmente pas seulement la charge $FEC_{Protect}$, mais il utilise aussi un profil de qualité inférieure pour éviter la congestion. La fonction getBestProfile(A) retourne le profil de meilleure qualité qui utilise un débit inférieur à A.

algorithme 3 Procédure de l'état PROTECT

```
1:  function PROTECT(params)
2:      if FEC_Probe > 0 then
3:          FEC_Probe ← 0
4:      else
5:          FEC_Protect ← FEC_Protect + 0.05
6:          if curUsedBandwidth > capacity × 0.9 then
7:              currentProfile ← getBestProfile(currentProfileBitrate ∗ 0.95)
8:          end if
9:      end if
10:     Prot_time = 20
11: end function
```

4.3.4 État INCREASE

Cet état remplace le trafic $FEC_{Protect}$ et le trafic FEC_{Probe} par un trafic vidéo. N'ayant pas observé de perte de paquets résiduelle pendant une période de temps égale à $Prot_{Time}$, l'algorithme réduit la charge $FEC_{Protect}$ de 10 % du débit vidéo et change le profil utilisé pour un profil ayant une meilleure qualité de sorte que la bande passante totale utilisée ($totalProtectBitrate$) est maintenue constante. Cette procédure est effectuée progressivement (une diminution de 10 % de $FEC_{Protect}$ chaque 0,5 s) pour augmenter la qualité d'une façon harmonieuse et pour éviter la dégradation de la qualité lorsque le nouveau taux de $FEC_{Protect}$ ne permet plus de récupérer tous les paquets. L'algorithme de l'état INCREASE est présenté dans Algorithme 4.

algorithme 4 Procédure de l'état INCREASE

```
1: function INCREASE(params)
2:     if FEC_probe > 0 then
3:         currentProfile ← getBestProfile(currentProfileBitrate + FECProbeBitrate)
4:         FEC_Probe ← 0
5:     end if
6:     if FEC_Protect > 0 then
7:         if Prot_time = 20s then
8:             totalProtectBitrate ← currentProfileBitrate + FECProtectBitrate
9:         end if
10:        Prot_time = 0.5s
11:        FEC_Protect ← FEC_Protect − 0.1
12:        currentProfile ← getBestProfile(totalProtectBitrate − FECProtectBitrate)
13:    end if
14: end function
```

4.3.5 État DECREASE

L'algorithme passe de l'état HOLD à cet état lorsque la médiane des taux de pertes résiduelles des paquets est non nulle et qu'il n'est plus possible d'augmenter $FEC_{Protect}$ (c.-à-d. $FEC_{Protect} = 1$) ou lorsque la perte de paquets est trop élevée et l'ajout de plus de FEC ne permettrait pas de récupérer tous les paquets. Si la congestion n'est pas sévère (c.-à-d. si $medianResidualLoss < 0{,}3$), l'algorithme change le profil actuel pour un profil d'une qualité inférieure qui a un débit égal à currentProfileBitrate $\times (1 -$ medianResidualLoss $\times 0,5)$. Sinon, il désactive la FEC, car lors d'une congestion sévère, la charge FEC peut empirer la situation et change le profil actuel par un profil d'une qualité inférieure qui a un débit égal à currentProfileBitrate $\times$ $(1 -$ medianLoss $\times 0,5)$. Ce mécanisme augmente considérablement la bande passante utilisée lorsqu'il y a une congestion sévère, ce qui permet d'augmenter la perte de paquets et d'empirer la situation. L'algorithme de l'état DECREASE est présenté dans Algorithme 5.

algorithme 5 Procédure de l'état DECREASE

```
1:  function DECREASE(params)
2:      if FEC_probe > 0 then
3:          FEC_probe = 0
4:      end if
5:      if currProbingMode = fastrecovery then
6:          setProbingMode(normal)
7:      end if
8:      if timeSinceLastDecrease > 1s then
9:          if medianResidualLoss < 0.3 then
10:             currentProfile ← getBestProfile(currentProfileBitrate × (1 − medianResidualLoss × 0.5))
11:         else
12:             FEC_Protect ← 0
13:             DisableNack()
14:             currentProfile ← getBestProfile(currentProfileBitrate × (1 − medianLoss × 0.5))
15:         end if
16:     end if
17: end function
```

4.4 Phase de démarrage

L'algorithme d'adaptation de débit doit s'adapter rapidement aux conditions initiales du réseau. En commençant par le profil qui utilise le moins de bande passante (c.-à-d. le profil ayant la pire qualité), notre algorithme prend plus de 50 s, en utilisant les états INCREASE et PROBE en mode Normal, pour atteindre le profil ayant la meilleure qualité. Ce temps est considéré comme supérieur aux attentes de la spécification définie par le groupe RMCAT. Pour cette raison, il faut utiliser une méthode permettant de trouver le meilleur profil à utiliser pendant le début de la communication vidéo.

Dans un premiers temps, nous avions pensé utiliser une méthode d'estimation de la bande passante spécifiquement pour le démarrage. Une fois la quantité de la bande passante disponible au démarrage connue, l'algorithme choisit le meilleur profil à utiliser. Malheureusement, après une recherche approfondie, nous avons trouvé que toutes les méthodes permettant d'estimer la quantité de la bande passante pour les réseaux

filaires, telles que BART [53] ou PathChirp [54], ne peuvent pas être utilisées pour estimer la bande passante pour les réseaux sans fils et cellulaire. Nous avons aussi trouvé que les méthodes développées spécifiquement pour estimer la bande passante dans les réseaux sans fils ou cellulaire, telles que WBest [55] et ProbeGap [56] sont seulement efficaces dans des bancs de test exécutés en laboratoire. Ces méthodes donnent de fausses estimations lorsqu'elles sont utilisées dans un environnement réel comme les réseaux des opérateurs ou les points d'accès Wi-Fi public [57].

Nous avons donc décidé d'utiliser une autre méthode plus simple, mais très efficace. Pendant la phase de démarrage de la communication, nous augmentons le débit vidéo directement (c.-à-d. sans avoir recours aux FEC) et sans faire attention aux différents indices de congestion. Le fait de fonctionner de cette manière peut engendrer une importante perte de paquets qui dégradera la qualité de la communication vidéo. Pour que l'utilisateur ne soit pas gêné par cette dégradation, l'application cache la vidéo pendant la phase de démarrage et affiche un écran de chargement. Une fois la phase de démarrage achevée (c.-à-d. lorsque l'algorithme a trouvé le bon profil à utiliser), l'algorithme enlève l'écran de chargement et affiche la vidéo reçue. En utilisant cette méthode, il est important de minimiser le temps nécessaire pour que la phase d'adaptation au démarrage s'achève sans ennuyer l'appelé. Dans ce mémoire, nous proposons deux algorithmes d'adaptation au démarrage différent : Binary Search Start (BSS) et Max Start (MaxS).

4.4.1 BSS

Nous pensons que le problème d'adaptation au démarrage est similaire à un problème de recherche : Il faut trouver le profil qui peut être utilisé dans les conditions initiales du réseau. Le problème est similaire à la recherche d'une valeur (la valeur de la bande passante disponible) dans un tableau ordonné (un tableau contenant la bande passante disponible de chaque profil). La valeur de la bande passante disponible n'est pas connue mais notre système peut envoyer un feed-back qui indique si le profil courant a une bande passante supérieure ou inférieure à la bande passante disponible. En d'autres termes, si le RTCP RR indique que la perte de paquet est nulle, le profil actuel est en train d'utiliser une bande passante inférieure à la bande passante disponible. Sinon, si la perte de paquets est non nulle, le profil actuel est en train d'utiliser une bande passante supérieure à la bande passante disponible. Ce problème de recherche peut être résolu en utilisant l'un des algorithmes les plus rapides pour la recherche d'une valeur dans un tableau ordonné [58]. L'idée de BSS est de commencer par un profil utilisant une bande passante égale à peu près à la moyenne ($midPoint$),

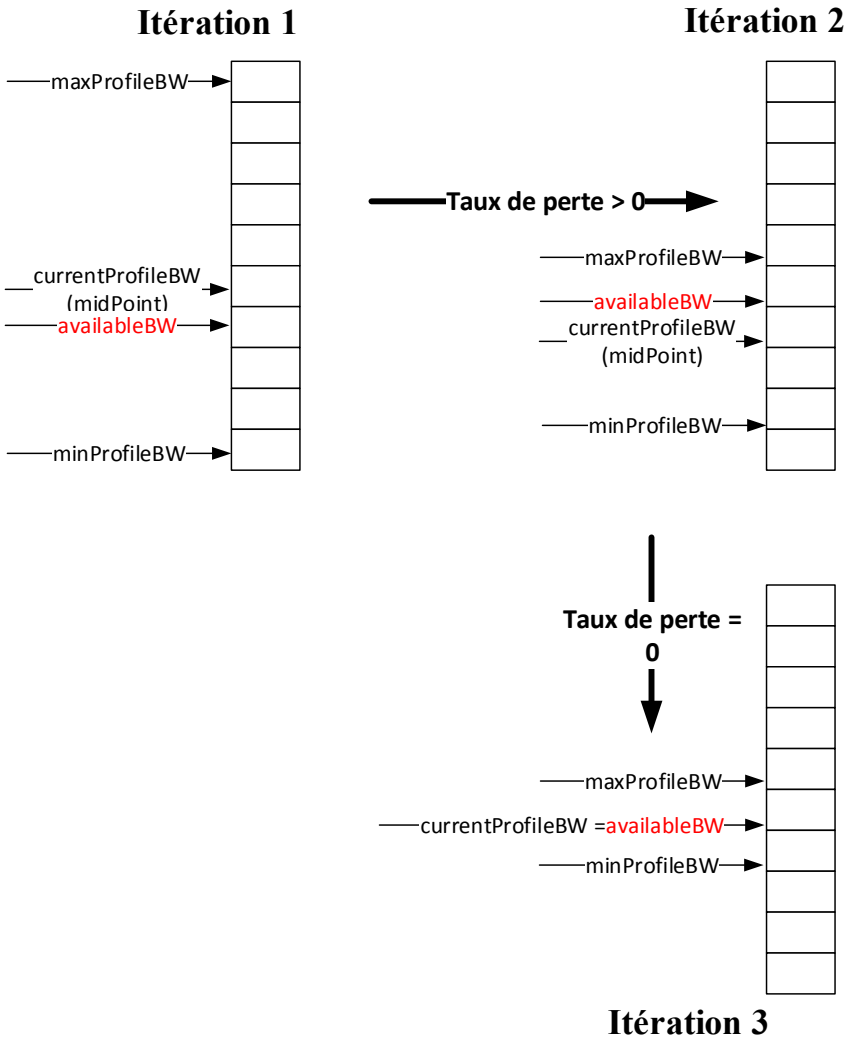


FIGURE 4.8 – Exemple d'exécution de l'algorithme BSS

avec :

$$\text{midPoint} = \frac{\text{Bande passante de } P_{max} + \text{Bande passante de } P_{min}}{2}. \tag{4.1}$$

Un taux de perte non nul signifie que la valeur de la bande passante disponible se trouve entre la bande passante du profil P_{min} ($minProfileBW$) et $midPoint$. Un taux de perte nul signifie que la valeur de la bande passante disponible se trouve entre $midPoint$ et la bande passante du profil P_{max} ($maxProfileBW$). La figure 4.8 montre un exemple dans lequel le système a reporté une perte de paquet non nul dès la première itération. Dans ce cas, l'algorithme réduit l'espace de recherche entre $currentProfileBW$ et $minProfileBW$ et utilise un profil ayant une bande passante égale à la nouvelle valeur de $midPoint$. Le système ne signale pas de perte de paquets ce qui signifie que la valeur de bande passante disponible se

trouve entre $minProfileBW$ et la nouvelle valeur de $maxProfileBW$. L'algorithme continue l'exécution de ces itérations jusqu'à trouver le profil adéquat. Puisque notre système n'estime pas les mesures de la bande passante disponible et compte seulement sur les taux de pertes reportés, le BSS doit utiliser une condition d'arrêt pour qu'il converge. Pour cette raison, nous le limitons par un temps d'exécution maximal (max_{Time}) et par d'autres conditions d'arrêt: si le taux de perte de paquets est entre $pLoss_{Limit1}$ (= 5 %) et $pLoss_{Limit2}$ (= 1 %), l'algorithme change le profil utilisé pour le deuxième profil de qualité inférieur. Si le taux de perte de paquets est entre $pLoss_{Limit2}$ et 0, l'algorithme change le profil utilisé pour le profil de qualité inférieure suivant. Si le temps limite d'exécution est atteint, l'algorithme utilise le profil ayant la bande passante minimale dans l'espace de recherche courant (c.-à-d. la valeur de $minProfileBW$ la plus récente). L'algorithme de BSS est présenté dans Algorithme 6.

algorithme 6 Procédure d'adaptation au démarrage en utilisant BSS

```
1:  function STARTUPADAPTATION(params)
2:      if packetLoss > pl_limit1 then
3:          maxProfileBW ← midpointBW
4:          midpointBW ← (maxProfileBW + minProfileBW)/2
5:          if currentTime < max_Time then
6:              currentProfile ← getBestProfile(midpointBW)
7:          else
8:              currentProfile ← getBestProfile(minProfileBW)
9:              return
10:         end if
11:     else if packetLoss ≤ pl_limit1 & packetLoss > pl_limit2 then
12:         currentProfile ← getBestProfile(currentProfiletBW) − 2
13:         return
14:     else if packetLoss ≤ pl_limit2 & packetLoss > 0 then
15:         currentProfile ← getBestProfile(currentProfileBW) − 1
16:         return
17:     else if packetLoss == 0 then
18:         if currentTime < max_Time then
19:             minProfileBW ← midpointBW
20:             midpointBW ← (maxProfileBW + minProfileBW)/2
21:             currentProfile ← getBestProfile(midpointBW, 1)
22:         else
23:             return
24:         end if
25:     end if
26: end function
```

4.4.2 MaxS

L'algorithme de Max Startup est plus simple que l'algorithme de BSS. Il commence par utiliser le profil ayant la meilleure qualité, P_{max}. Si une perte de paquet se produit, l'algorithme effectue une diminution proportionnelle aux taux de perte. Il change le profil courant par un profil ayant une bande passante B qui satisfait l'équation suivante, où α est une constante entre 0 et 1 :

$$B = B_{curr} \times (1 - \alpha \times \text{packetLoss}). \tag{4.2}$$

Nous comparons les performances des deux algorithme, MaxS et BSS, dans le chapitre 5.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre algorithme d'adaptation de débit dynamique aux conditions du réseau. Cet algorithme utilise le rapport RTCP RR et des sondes FEC afin d'estimer les conditions du réseau. En utilisant cette estimation, l'algorithme décide d'améliorer la qualité de la vidéo, en diminuant son taux de compression, ou de dégrader la qualité de la vidéo, en augmentant son taux de compression. En utilisant les estimations des conditions du réseau, l'algorithme décide aussi entre l'augmentation ou la diminution de la robustesse de la vidéo face aux pertes. Il contient aussi une phase d'adaptation au démarrage qui lui permet d'estimer les conditions initiales du réseau et d'adapter la vidéo rapidement à ces conditions.

Chapitre 5

Évaluation des performances

5.1 Introduction

Le réseau Internet est un assemblage de technologies diverses et d'une grande variété d'équipements. Les performances offertes au sein de ce réseau peuvent varier d'une technologie d'accès à une autre, d'un opérateur à un autre et d'un équipement à un autre. Elles peuvent varier même au sein du réseau du même opérateur en raison d'une configuration ou d'une topologie différentes d'un endroit à un autre. Les performances de ce réseau dépendent aussi énormément du comportement des utilisateurs qui l'utilisent. Un réseau d'accès peut offrir des performances très variées dans des intervalles de temps différents. Ainsi, un réseau peut donner de bonnes performances entre 4 h et 6 h du matin lorsque la majorité des utilisateurs ne sont pas encore réveillés et de très mauvaises performances entre 19 h et 21 h pendant l'heure de pointe du trafic [59].

Toutes ces conditions et évènements aléatoires affectant le réseau Internet présentent un grand défi pour l'évaluation de la performance des algorithmes de contrôle de congestion. En effet, pour développer un bon algorithme de contrôle de congestion, il faut que ce dernier offre de bonnes performances dans toutes les conditions. Cependant, il n'est pas possible de tester systématiquement la solution proposée dans un réseau réel, car le développeur n'a pas le contrôle sur le réseau et ne peut pas connaitre exactement les types d'activités et d'évènements qui sont en train de se dérouler sur le réseau pendant la réalisation du test.

Ainsi, pour pouvoir développer un algorithme de contrôle de congestion performant, il est nécessaire dans une première étape de le tester en utilisant plusieurs bancs de test : chaque banc de test émule un cas ou une condition particulière qui peut se produire sur les réseaux d'Internet. Pour cela, le groupe RMCAT

a proposé plusieurs critères d'évaluations et plusieurs exemples de banc de test permettant d'évaluer ces différents critères. Dans ce chapitre, nous présentons la démarche suivie afin d'évaluer les performances de notre algorithme et les différents résultats obtenus. Nous commençons par décrire notre environnement d'évaluation et les différents éléments qui le composent. Aussi, nous dévoilons les métriques d'évaluation et les méthodes utilisées pour les mesurer. Finalement, nous présentons les différents scénarios de test exécutés et nous discutons les résultats obtenus pour chaque scénario.

5.2 Implémentation et environnement d'évaluation

Pour évaluer les performances de notre algorithme, nous utilisons une implémentation réelle au lieu d'une simulation. Ainsi, nous développons une application de communication vidéo interactive qui utilise notre algorithme afin d'adapter le débit vidéo aux conditions du réseau. Nous utilisons l'API WebRTC pour développer notre application parce qu'elle permet d'accélérer la phase d'implémentation de notre algorithme.

5.2.1 WebRTC

Web Real-Time Communication (WebRTC) est une implémentation du protocole RTCWeb (Real Time Communication on WEB-browsers) [60] qui est en cours de développement par l'IETF. WebRTC offre plusieurs composants et fonctions fondamentaux pour le développement des applications de communication interactive audio et vidéo sur un navigateur. Ces composants, quand ils sont implémentés dans le navigateur, peuvent être manipulés via une API JavaScript nommé WebRTC API qui est définie par le World Wide Web Consortium (W3C). Ceci permet aux développeurs d'implémenter facilement et rapidement des applications Web de communications interactives. Les applications basées sur WebRTC peuvent être exécutées directement sur le navigateur sans l'installation d'applications supplémentaires ou de plug-ins comme c'est le cas pour la majorité des services de communications interactive sur Internet existants. Cette technologie vise à donner à la communauté de développement une technologie gratuite et ouverte permettant la création d'application de communication temps réel de très bonne qualité. La figure 5.1 représente l'architecture globale de WebRTC. WebRTC contient les modules suivants :

– Web API : C'est une API qui peut être utilisée par des développeurs tiers pour développer des applications Web de communications temps réel.

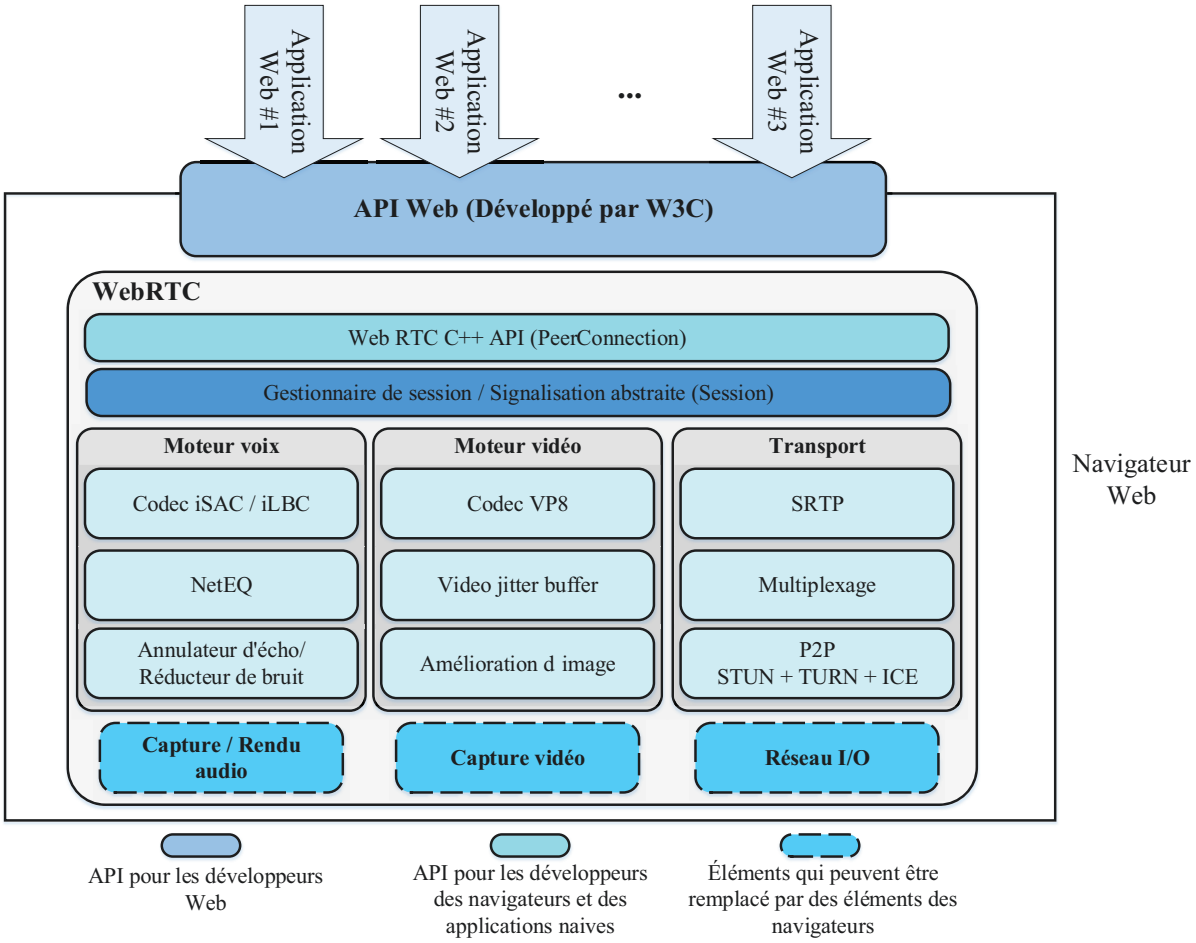


FIGURE 5.1 – Architecture de WebRTC

– WebRTC Native C++ API : C'est la partie intégrée dans le navigateur, utilisée par les développeurs de navigateur Web pour éviter l'installation de plug-ins supplémentaires. Cette partie peut être aussi utilisée pour développer des applications natives.

– Transport / Session : Ce module contient une implémentation des protocoles RTP et RTCP. Cette implémentation supporte le profil RTP/SAVPF. En conséquence, il supporte le mécanisme de retransmission NACK. Ce module supporte aussi le mécanisme FEC défini par le RFC 5109 et implémente des mécanismes permettant d'établir la communication P2P à travers différentes conditions d'accès au réseau comme STUN [61], TURN [62] et ICE [63].

– Gestionnaire de session : C'est une couche de session abstraite pour permettre l'établissement d'appel et la gestion de la session. Cette couche offre donc la possibilité aux développeurs d'utiliser n'importe quel protocole de signalisation pour le développement de leurs applications.

– Moteur voix : c'est un Framework pour le traitement de la voix dans toutes les étapes, de la carte son jusqu'au réseau. Il contient les codecs audio iSAC [64], iLBC [65] et Opus [66]. Il contient aussi un composant appelé NetEQ. Ce composant comprend un « Jitter buffer » et un algorithme de dissimulation d'erreurs permettant de dissimuler les effets négatifs de la gigue du réseau et des pertes de paquets. Le moteur voix comprend aussi un réducteur de bruit permettant de supprimer certains types de bruit de fond.
– Moteur vidéo : C'est un Framework pour le traitement de la vidéo dans les différentes étapes de la chaine média, du camera vers le réseau et du réseau vers l'écran. Il supporte le codec VP8 [67]. Il contient un « Jitter Buffer » pour la vidéo qui aide à dissimuler l'effet de la gigue et de la perte de paquet sur la qualité de la vidéo.
– Module d'amélioration d'image : Ce module améliore la qualité des images capturée par la webcam par exemple en supprimant le bruit des images capturées.

L'API WebRTC nous offre plusieurs avantages qui permettent d'économiser beaucoup de temps pendant la phase d'implémentation de notre algorithme, mais elle présente aussi un inconvénient. En effet, présentement, WebRTC ne supporte pas le codec H.264. Il supporte seulement le codec VP8. En conséquence, au lieu d'utiliser les paramètres de profil définis dans la section 4.2.1, nous changeons seulement le débit vidéo. Chaque profil aura donc un débit différent.

5.2.2 Architecture de l'application

Pour effectuer les tests présentés dans la section 5.4, nous développons une application Web de communication vidéo interactive. L'application est composée de deux parties : un client et un serveur d'application. La partie client est une application développée en utilisant l'API WebRTC, JavaScript et HTML5. Elle peut jouer le rôle d'un récepteur ou un émetteur. L'émetteur envoie une vidéo capturée depuis la Webcam vers le récepteur en temps réel. Le récepteur reçoit la vidéo et l'affiche sur l'écran. Le serveur d'application s'occupe de la gestion de la signalisation. La figure 5.2 montre l'architecture d'un banc de test. L'émetteur et le récepteur sont connectés l'un à l'autre par un relai qui joue le rôle d'un point de congestion. Pour conserver les mêmes conditions entre les différents tests et les différentes exécutions de chaque test, l'émetteur envoie une vidéo enregistrée d'une tête parlante au lieu d'envoyer des images réelles capturées par la webcam. Nous utilisons une vidéo de tête parlante, car notre algorithme est destiné pour être utilisé avec les applications qui permettent aux utilisateurs de passer des appels vidéo via Internet (p. ex. Skype).

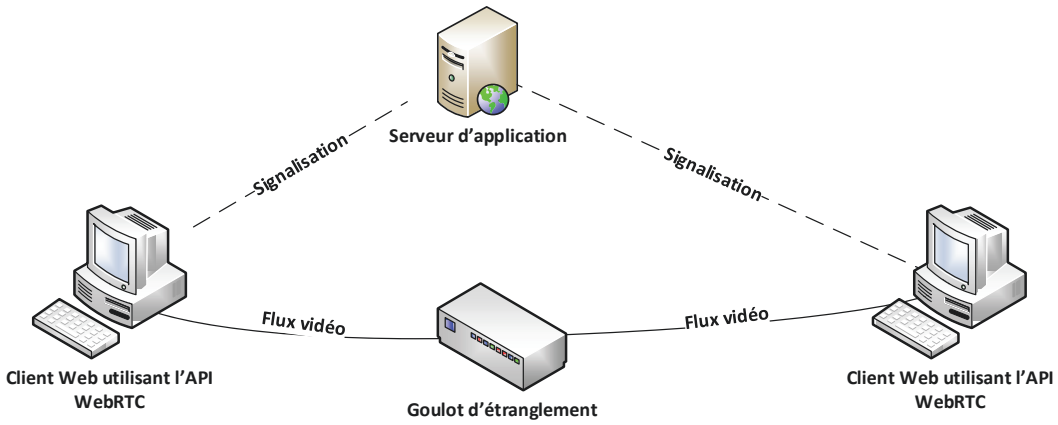


FIGURE 5.2 – Architecture de notre application de communication vidéo interactive

5.2.3 Émulation réseau

Afin de créer le goulot d'étranglement, nous utilisons une solution d'émulation de réseau appelé Dummynet. Le principe d'un émulateur de réseau est de placer des mécanismes en des points particuliers d'une plate-forme pour reproduire le comportement souhaité : médium de communication engendrant des pertes, du délai, limite du débit... À la différence de la simulation, l'émulation s'appuie sur des équipements réels, ce qui permet de mettre en place physiquement une plate-forme d'expérimentation. De nombreux émulateurs sont disponibles avec des logiciels sous licence commerciale ou libre. Dans ce mémoire, nous avons décidé d'utiliser l'émulateur Dummynet [68] pour son efficacité et sa souplesse. Dummynet est un logiciel intégré au pare-feu disponible dans le système d'exploitation FreeBSD. Il fait une mise en forme du trafic et a été développé à l'origine pour tester les connexions TCP. Il permet de modifier les propriétés du trafic traversant l'émulateur en appliquant un délai, une bande passante et un taux d'erreur paquet. Le pare-feu délivré par la distribution de FreeBSD est appelé IP firewall et est accessible par la commande ipfw. Il intercepte les paquets, en fonction de filtres, au niveau des interfaces physiques de l'émulateur et les redirige vers le logiciel. Pour le filtrage, il possède un ensemble de règles qui permettent de rediriger uniquement les paquets souhaités. Les règles se construisent à partir des champs des en-têtes des paquets IP, et permettent de filtrer selon un ou plusieurs des paramètres suivants [69]:

- interface physique d'émission ou de réception
- adresse IP source et/ou destination
- port destination
- protocole

Pour les paquets correspondant au filtre d'une règle, Dummynet permet de :

- Limiter le débit du flux de ces paquets concernés
- Modifier les délais de transit de ces paquets
- Introduire des pertes de paquets

Dans les tests, nous avons utilisé Dummynet pour contrôler le délai et limiter le débit du goulot d'étranglement.

5.3 Outils de mesures et métriques d'évaluation

Comme nous l'avons indiqué dans le chapitre 3, le but principal de tous les algorithmes de contrôle de congestion est de minimiser la perte de paquets tout en maximisant la bande passante utilisée. En conséquence, un bon algorithme de contrôle de congestion est un algorithme qui diminue rapidement la quantité de bande passante utilisée lorsqu'il y a une congestion pour diminuer le taux de perte de paquets, et qui l'augmente rapidement lorsque les conditions le permet.

Nous utilisons plusieurs métriques afin d'évaluer les performances de notre algorithme. Ces métriques peuvent être groupées en deux types : des métriques de trafic et des métriques de qualité vidéo.

5.3.1 Métrique de trafic

Les métriques de trafic sont des métriques qui quantifient la qualité du trafic envoyé. Nous avons utilisé les métriques suivantes :

- Débit vidéo : Le débit de la vidéo donne une information sur la qualité de la vidéo envoyée. Un faible débit vidéo signifie que l'algorithme est en train d'utiliser un profil de mauvaise qualité, alors qu'un débit vidéo élevé signifie que notre algorithme est en train d'utiliser un profil de bonne qualité.
- Débit FEC : Le débit moyen du trafic FEC reflète la quantité de trafic FEC utilisé. Notre algorithme utilise le FEC en tant que sonde. L'algorithme doit essayer d'utiliser un taux faible de FEC pour ne pas gaspiller les ressources disponibles.
- Taux de pertes de paquets (%) : Un taux de perte très faible ou nul signifie que notre algorithme réussit à se protéger contre la congestion et les flux concurrents.

Nous traçons pour chaque test la courbe du débit vidéo et la courbe de la capacité du goulot d'étranglement en fonction du temps. Ces courbes permettent d'évaluer l'adaptabilité, la stabilité et le temps de réponse de

notre algorithme. Nous traçons aussi la courbe du débit de FEC en fonction du temps. Cette courbe reflète la quantité du trafic FEC utilisé au cours du temps. Les différentes métriques ci-dessous sont mesurées en ajoutant des morceaux de code permettant d'enregistrer les statistiques des différents types de paquets envoyés et les statistiques des rapports RTCP RR.

5.3.2 Métrique de qualité vidéo

Les métriques de qualité vidéo permettent d'apprécier la QoE de façon approximative, mais par un calcul objectif. Les métriques que nous utilisons sont les suivantes : Peak Signal-to-Noise Ratio (PSNR) et Structural Similarity (SSIM).

Peak Signal-to-Noise Ratio (PSNR)

Le PSNR est une mesure de distorsion qui est souvent utilisée pour évaluer la qualité des images compressée. Elle permet de quantifier la performance des codeurs en mesurant la qualité de la reconstruction de l'image compressée par rapport à l'image originale. Le PSNR est défini par :

$$PSNR = 10 \times \log_{10} \left(\frac{d^2}{EQM} \right), \tag{5.1}$$

où d est la dynamique du signal (la valeur maximale possible pour coder un pixel). Dans le cas standard d'une image où les composants d'un pixel sont codés sur 8 bits, d est égale à 255 et EQM est l'erreur quadratique. Elle est définie pour 2 images I_1 et I_2 de taille $M \times N$ comme :

$$EQM = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \parallel I_1(i,j) - I_2(i,j) \parallel^2 . \tag{5.2}$$

Le PSNR de la vidéo est la moyenne des valeurs de PSNR de toutes les images qui la composent.

Structural Similarity (SSIM)

Le SSIM permet aussi de mesurer la qualité visuelle d'une image par rapport à l'image originale. Il permet de mesurer la similarité de structure entre les deux images, plutôt qu'une différence pixel à pixel comme le fait le PSNR. La raison de la création de cette métrique est l'hypothèse qui stipule que l'œil

humain est plus sensible aux changements de la structure de l'image que des pixels. La métrique SSIM est calculée sur plusieurs blocs d'une image. Un bloc est une partie de l'image. Il contient ainsi plusieurs pixels. La mesure entre deux blocs $x = \{x_i | i = 1, 2, ..., M \times N\}$ (x_i est un pixel appartenant au bloc x) et $y = \{y_i | i = 1, 2, ..., M \times N\}$ (y_i est un pixel appartenant au bloc y) de taille $M \times N$ est [70, 71]:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2cov_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{5.3}$$

où

- μ_x est la moyenne de x;
- μ_y est la moyenne de y;
- σ_x^2 est la variance de x;
- σ_y^2 est la variance de y;
- cov_{xy} est la covariance de x et y;
- $c_1 = (k_1 L)^2$, et $c_2 = (k_2 L)^2$ sont deux variables destinées à stabiliser la division quand le dénominateur est très faible;
- L est la dynamique des valeurs des pixels, soit 255 pour des images codées sur 8 bits;
- $k_1 = 0,01$ et $k_2 = 0,03$ par défaut.

5.4 Tests et résultats

Pour évaluer les performances de notre algorithme, nous exécutons plusieurs scénarios de tests. Le but de cette étape est de vérifier si notre algorithme respecte les différentes conditions requises énoncées dans le document de spécification définie par le groupe RMCAT [42]. Afin d'atteindre ce but, chaque test doit évaluer un ou plusieurs critères parmi la liste des critères d'évaluation établie par RMCAT dans [43]. La majorité de ces tests a été inspirée des différents tests proposés par RMCAT dans [72].

5.4.1 Test 1 : Phase de démarrage

Le but de ce test est d'évaluer la rapidité et l'efficacité des estimations effectuées par les deux algorithmes BSS et MaxS. Afin de les tester, nous avons utilisé un banc de test simple composé d'un récepteur et d'un émetteur qui sont connectés entre eux par un relai Dummynet jouant le rôle d'un goulot d'étranglement. Nous exécutons plusieurs tests : dans chaque cas, nous avons changé la capacité du goulot d'étranglement.

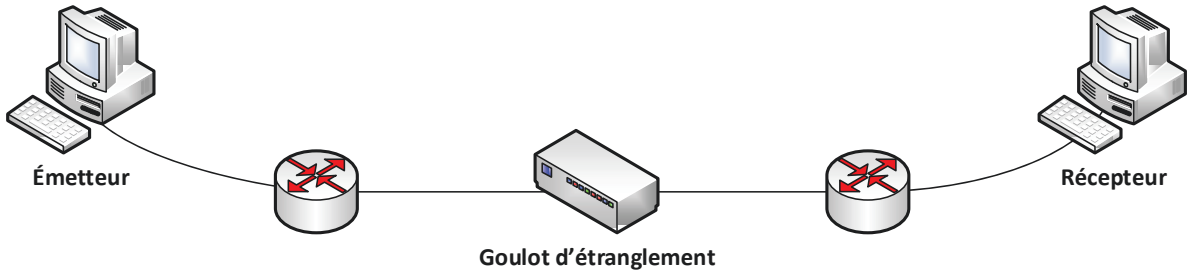


FIGURE 5.3 – Architecture du banc de test 1

Tableau 5.1 – Les résultats de BSS et de MaxS

Bande passante disponible (Kb/s)	Temps de convergence (s)		Efficacité	
	BSS	MaxS	BSS	MaxS
200	9	10	0,92	0,80
500	6	6	0,96	0,72
1000	14	4	1	0,80
1500	13	6	0,92	0,84
2000	13	5	1	0,96
3000	14	5	1	1

La figure 5.3 montre l'architecture du banc de test utilisé pour la réalisation de ce scénario. Pour chaque test, nous mesurons le temps de convergence et l'efficacité. L'efficacité est mesurée en utilisant la formule suivante :

$$\text{Efficacité} = 1 - \frac{P_p - P_c}{Nbr_p}, \tag{5.4}$$

où, P_c est le profil qui sera utilisé après la convergence, P_p est le profil qui offre la meilleure qualité pour la bande passante disponible et Nbr_p est le nombre total des profils.

L'efficacité varie entre 0 et 1 : la valeur 0 indique que l'algorithme utilise le plus mauvais choix et la valeur de 1 indique qu'il choisit le meilleur profil. Les résultats de ce test sont illustrés dans le tableau 5.1. Selon les résultats obtenus, nous remarquons que MaxS sous-estime la bande passante disponible et commence avec un profil ayant une bande passante significativement inférieure à la bande passante disponible. Cependant, il converge très rapidement, particulièrement lorsque la bande passante disponible est supérieure à 500 Kb/s. BSS est très efficace, car il utilise un profil ayant une bande passante presque égale à la bande passante disponible. Cependant, il a un temps de convergence supérieur à celui de MaxS, particulièrement lorsque la bande passante disponible est supérieure à 500 Kb/s. Chaque algorithme a donc ses forces et ses

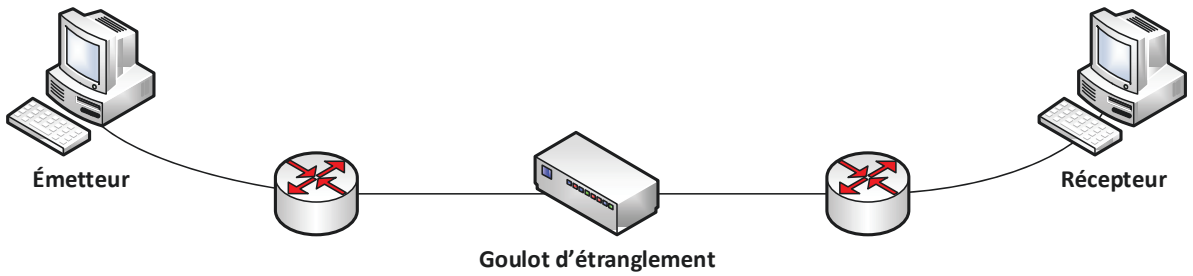


FIGURE 5.4 – Architecture du banc de test 2

faiblesses. Si l'opérateur préfère la rapidité de l'estimation à l'efficacité, il doit choisir MaxS. Dans le cas contraire, il doit choisir l'algorithme BSS.

Notons que nous utilisons l'algorithme MaxS dans les tests suivants. La raison principale de ce choix est de montrer comment notre algorithme peut adapter le débit et choisir un meilleur profil après la convergence de MaxS. Si nous utilisons BSS, nous n'observons aucune adaptation au débit de l'exécution de chaque test, car le meilleur profil possible est utilisé.

5.4.2 Test 2 : Test des modes de l'état PROBE

Description générale

Le but de ce test est d'évaluer l'apport du mode prudent et du mode agressif. Dans ce test, nous varions la capacité du goulot d'étranglement afin de voir le comportement de notre algorithme. Ce test est exécuté 2 fois : une fois en utilisant seulement le mode normal et une deuxième fois en utilisant tous les modes.

Banc de test

La topologie du banc de test est présentée par la figure 5.4. Le banc de test est composé de deux clients WebRTC, un émetteur et un récepteur, connectés par un relai Dummynet qui joue le rôle d'un goulot d'étranglement. Le test est exécuté pendant 235 s. Pendant les premières 55 s, la capacité du relai est fixée à 4000 Kb/s. Ensuite, elle sera fixée à 1000 Kb/s pendant 100 s et finalement, nous augmentons la capacité à 4000 Kb/s pour terminer le test. La liste détaillée des valeurs des paramètres de ce banc de test est représentée dans l'annexe A.

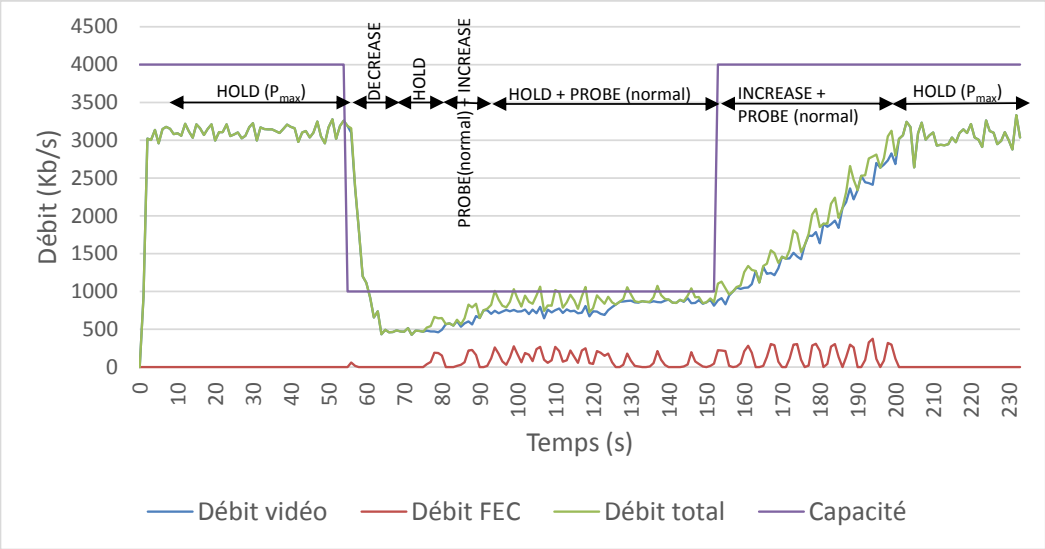


FIGURE 5.5 – Test des modes de l'état PROBE (seulement le mode normal)

Résultats et observations

La figure 5.5 montre les résultats de l'exécution de ce test en utilisant uniquement le mode normal. Grâce à MaxS, l'algorithme commence par le profil P_{max}, car il y a assez de bande passante disponible. Au début, l'algorithme garde les paramètres constants en restant dans l'état HOLD. Lorsqu'on diminue la capacité du goulot d'étranglement, l'algorithme passe à l'état DECREASE et diminue la quantité de la bande passante utilisée en utilisant un profil ayant une qualité plus basse. Ensuite, il commence à sonder pour plus de bande passante jusqu'à 155 s en utilisant l'état PROBE. Puisqu'il n y a pas de bande passante disponible, les sondes échouent, ce qui oblige l'algorithme à retourner à l'état HOLD.

La version qui utilise tous les modes, figure 5.6, utilise moins de sondes FEC que l'autre version parce qu'elle utilise le mode prudent. Ceci permet de diminuer le taux de pertes de paquets et le délai qui peuvent être causés par les sondes échouées. En effet, l'envoi de plusieurs sondes FEC pendant que la bande passante disponible est très limitée cause de légères congestions successives. Ces dernières augmentent le délai, sa variation et le risque d'avoir des pertes de paquets qui ne peuvent pas être récupérées par le trafic FEC envoyé. À 156 s, nous augmentons la capacité à 4000 Kb/s. l'algorithme s'adapte rapidement, en utilisant l'état PROBE pour envoyer des sondes FEC et en utilisant l'état INCREASE afin de remplacer le trafic FEC par du trafic vidéo. Dans la figure 5.6, l'algorithme active le mode Fast Recovery ce qui permet d'atteindre le profil maximal en 30 s. Dans la figure 5.5, l'algorithme prend plus de 50 s pour atteindre le profil maximal, car il n'utilise pas le mode agressif.

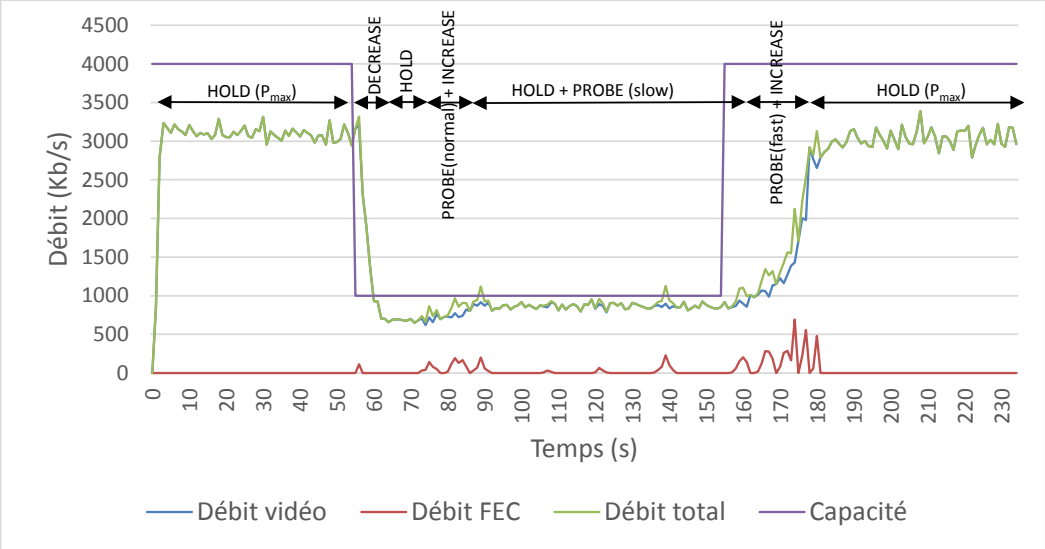


FIGURE 5.6 – Test des modes de l'état PROBE (tous les modes)

Tableau 5.2 – Tableau des résultats du Test 2

	Mode normal seulement	Tous les modes
Débit vidéo moyen (Kb/s)	1874	2079
Taux de perte (%)	1,19	1,03
Débit de FEC (Kb/s)	62,15	27,75
PSNR (dB)	39,19	39,63
SSIM	0.947	0.952

Le tableau 5.2 montre le débit moyen de la vidéo, le taux de perte, le PSNR et le SSIM obtenus depuis les deux exécutions, celle qui utilise seulement le mode normal et celle utilise les trois modes. Les valeurs de ce tableau sont mesurées après la fin de la phase de démarrage (MaxS).

D'après le tableau 5.2, nous remarquons que l'utilisation de tous les modes permet d'augmenter le débit moyen utilisé de plus que 200 Kb/s. En conséquence, la qualité de la vidéo est améliorée légèrement puisque le PSNR et le SSIM ont augmenté légèrement. Nous remarquons aussi que l'ajout des modes prudent et agressif ont permis de diminuer le taux de FEC utilisé de plus de 50 % et de légèrement diminuer le taux de perte.

D'après les résultats obtenus dans ce test, nous concluons que l'utilisation des modes prudent et agressif a plusieurs avantages. En conséquence, nous utilisons ces modes dans tous les tests qui suivent.

5.4.3 Test 3 : Capacité de lien variable

Description générale

Dans ce test, la capacité du goulot d'étranglement entre deux participants varie en fonction du temps. Ce test se concentre principalement sur l'évaluation de l'adaptabilité, la stabilité et le temps de réponse de notre algorithme lors d'une variation rapide dans la capacité du lien séparant les deux participants. Ce test tente aussi de répondre aux requis suivants définis par RMCAT dans [42]:

- L'algorithme d'adaptation doit adapter le flux vidéo aux changements brusques dans la capacité du lien de bout en bout : le changement de capacité est généralement dû au changement d'un lien à cause du routage. Il peut être aussi dû au changement de l'environnement des terminaux sans fils. Par exemple, lorsqu'un utilisateur s'éloigne d'un point d'accès Wi-Fi, la capacité du lien sans fils peut changer rapidement. L'algorithme doit minimiser la bande passante utilisée lorsque la capacité diminue afin d'éviter la perte de paquets et doit augmenter la bande passante utilisée lorsque la capacité augmente pour améliorer la qualité de la vidéo.
- L'algorithme d'adaptation doit adapter le flux vidéo lorsque le débit maximal possible est supérieur à la capacité du lien : dans ce cas, l'algorithme essaye d'atteindre le profil maximal pour maximiser la bande passante disponible, puisque la capacité du lien est limitée à une valeur plus faible, l'algorithme doit stabiliser le débit d'envoi à une valeur très proche de la capacité disponible du goulot d'étranglement.

Banc de test

La topologie du banc de test est la même que celle du test précédent présenté dans la figure 5.4. Le banc de test est composé de deux clients WebRTC, un émetteur et un récepteur, connectés par un relai Dummynet. Au début, nous fixons la capacité du goulot d'étranglement à 2000 Kb/s pendant 45 s. Ensuite, nous fixons la capacité à 5000 Kb/s pendant 40 s. Par la suite, nous la changeons à 1200 Kb/s pendant 40 s. Finalement, nous augmentons la capacité du goulot à 2000 Kb/s pour le reste du test. La liste détaillée des valeurs des paramètres de ce banc de test est présentée dans l'annexe A.

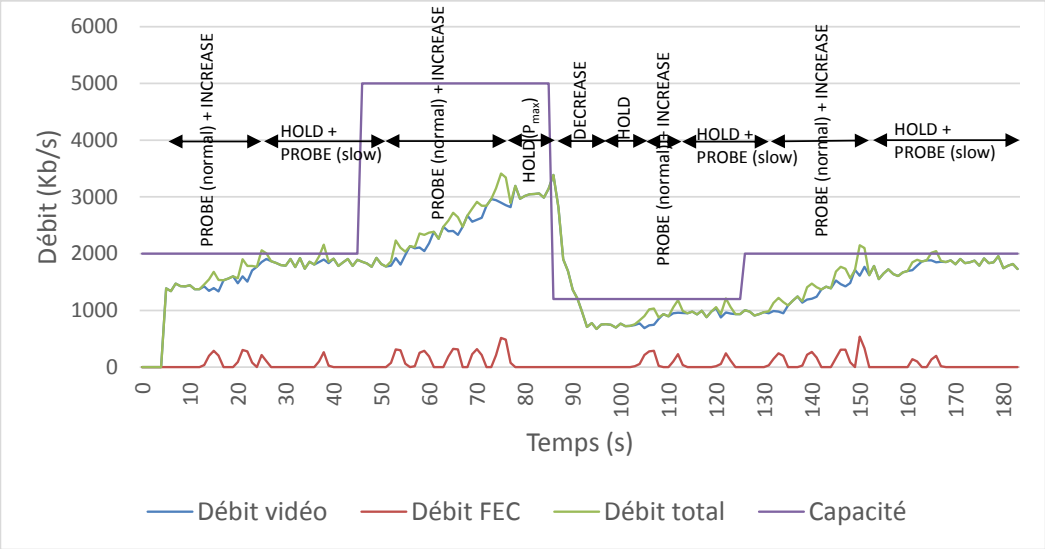


FIGURE 5.7 – Capacité de lien variable

Résultats et observations

La figure 5.7 montre le résultat de ce test. Au début, l'algorithme utilise MaxS pour estimer la quantité de la bande passante disponible et démarrer la communication vidéo en utilisant le profil le plus adapté aux conditions du réseau. Puisque la bande passante disponible est égale à 2000 Kb/s, notre algorithme démarre la vidéo avec un profil utilisant à peu près 1400 Kb/s. L'estimation de la bande passante au démarrage prend environ 5 s. À 15 s, l'algorithme passe à l'état PROBE pour sonder pour plus de bande passante et utilise l'état INCREASE pour passer progressivement aux profils supérieurs. À 78 s, notre algorithme atteint le profil maximal, P_{max}. Il passe donc à l'état HOLD pour conserver les différents paramètres utilisés. Lorsque nous diminuons la capacité du goulot d'étranglement, l'algorithme passe à l'état DECREASE et diminue rapidement le débit utilisé. Le temps nécessaire pour atteindre un profil utilisant une bande passante inférieure à la capacité du lien est égale à 6 s. Durant ce temps, nous observons des périodes d'arrêt dans la vidéo à cause des pertes de paquets. À 93 s, l'algorithme se stabilise en passant l'état HOLD. La bande passante utilisée pendant la stabilisation est légèrement supérieure à 750 Kb/s. À 103 s, l'algorithme améliore la qualité légèrement et utilise toute la bande passante disponible. Puisqu'il n'y a plus de bande passante disponible, les sondes FEC échouent, ce qui oblige l'algorithme à se stabiliser de nouveau et à activer le mode prudent. Après avoir augmenté la capacité du goulot d'étranglement à 125 s, l'algorithme utilise les états PROBE et HOLD pour améliorer la qualité progressivement jusqu'à l'utilisation de toute la bande passante disponible. Notre algorithme a réussi ce test. Il a pu adapter le débit de la vidéo au changement brusque de la capacité.

Il a réussi aussi à se stabiliser et utiliser une bande passante inférieure à celle du profil maximal lorsqu'il n'y a pas assez de bande passante disponible.

5.4.4 Test 4 : Équité entre plusieurs flux vidéo

Description générale

Dans ce test 3 flux vidéo partagent le même goulot d'étranglement et chacun d'entre eux utilise notre algorithme de congestion. Ce scénario peut se produire lorsque plusieurs utilisateurs utilisent la même passerelle réseau pour se connecter au réseau Internet et effectuent un appel vidéo vers d'autres utilisateurs distants. La capacité de la passerelle est limitée et elle ne peut pas supporter l'utilisation du profil maximal par tous les flux en même temps. Le but de ce test est d'évaluer l'équité entre plusieurs flux vidéo utilisant notre algorithme d'adaptation. Pour réussir ce test, tous les flux doivent partager la bande passante d'une façon équitable. Notre algorithme doit aussi offrir une qualité vidéo similaire à tous les récepteurs.

Banc de test

La topologie du banc de test est présentée dans la figure 5.8. Il est composé de 3 émetteurs et de 3 récepteurs. Chaque émetteur envoie un flux vidéo à un récepteur. La capacité du goulot d'étranglement est égale à 7000 Kb/s. Elle peut donc supporter deux flux utilisant simultanément le profil $P_{max}(= 3000\,Kb/s)$. Au début, nous démarrons le premier flux entre l'émetteur 1 et le récepteur 1. Ensuite, après 40 s, nous démarrons le deuxième flux entre l'émetteur 2 et le récepteur 2. Finalement, après 40 s, nous lançons le troisième flux entre l'émetteur 3 et le récepteur 3. Tous les flux traversent le même goulot d'étranglement. La liste détaillée des valeurs des paramètres de ce banc de test est présentée dans l'annexe A.

Résultats et observations

À cause de quelques phénomènes aléatoires liés à l'utilisation de vrais équipements, les résultats de ce test peuvent varier d'une exécution à une autre. Ainsi, pour la pertinence statistique, nous l'avons exécuté 5 fois. Les figures 5.9 et 5.10 montrent les résultats de la meilleure exécution et les figures 5.11 et 5.12 montrent les résultats pour la pire exécution. Dans l'exécution qui donne les meilleurs résultats, les deux premiers utilisent le profil maximal, car il y a assez de bande passante. Quand le troisième flux démarre,

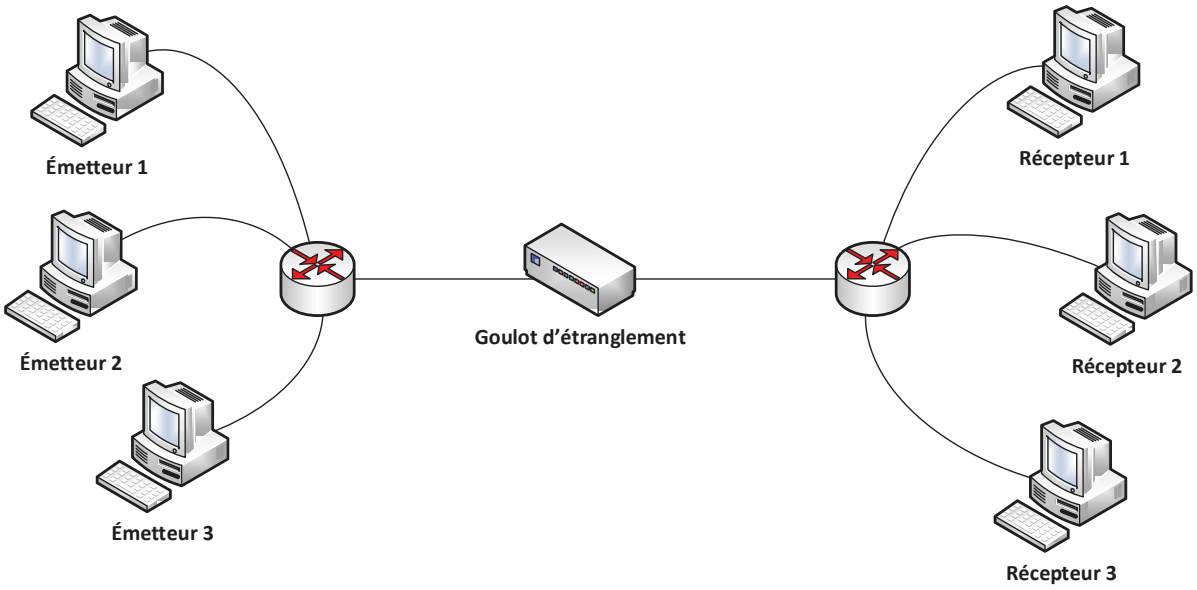
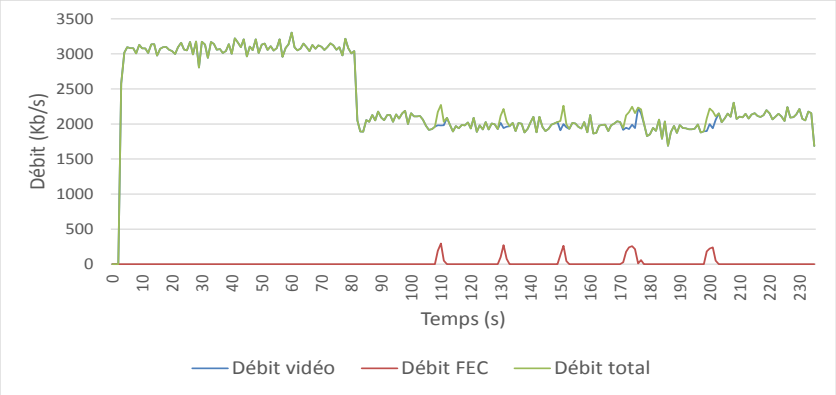


FIGURE 5.8 – Architecture du banc de test 4

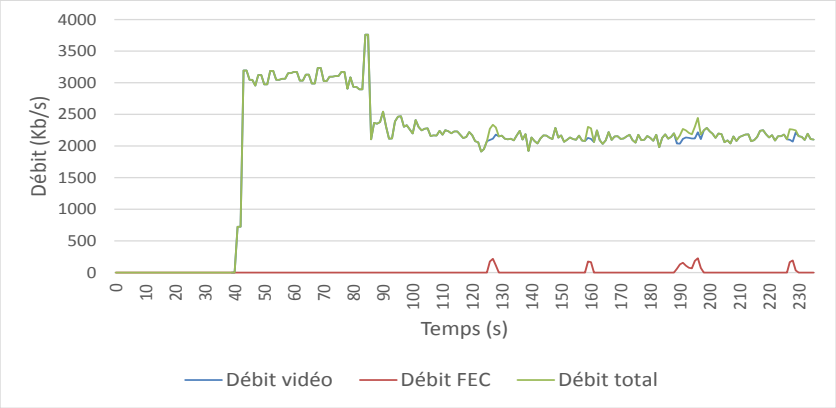
sa phase MaxS provoque une augmentation dans le taux de pertes de paquets des 2 autres flux, ce que les oblige à passer à l'état DECREASE et à basculer vers des profils de qualité inférieure jusqu'à l'annulation du taux de perte de paquets. Après la phase de démarrage, le flux 3 envoie des sondes FEC pour estimer de la bande passante disponible et passe à des profils de meilleur qualité jusqu'à ce qu'il n'y ait plus de bande passante disponible. Après la stabilisation, le flux 1 utilise environ 2000 Kb/s, le flux 2 utilise environ 2100 Kb/s et le flux 3 utilise environ 2400 Kb/s. Pour la pire exécution, les 3 flux prennent à peu près 80 s pour se stabiliser. Après la stabilisation, le flux 1 utilise un débit vidéo égal à 1500 Kb/s, le flux 2 utilise environ 2100 Kb/s et le flux 3 utilise environ 1800 Kb/s. Pour les 5 cas de tests, le flux 1 utilise un débit vidéo moyen égal à 1981 Kb/s, le flux 2 utilise un débit moyen de 2123 Kb/s et le flux 3 utilise un débit moyen de 2341 Kb/s. En utilisant les résultats obtenus, nous pouvons conclure que tous les flux sont grosso-modo équitables entre eux. L'algorithme offre une bonne QoE pour 3 flux. Cependant, nous remarquons une augmentation légère du nombre de trames perdues et quelques arrêts dans la vidéo à cause de cette perte. La raison de ce problème est que, lorsque 2 ou 3 flux envoient des sondes FEC en même temps, le nombre de paquets perdus augmente légèrement. De plus, la petite quantité du trafic FEC ne peut pas compenser tous les paquets perdus. Ce problème ne provoque pas une diminution importante dans la QoE car il se produit rarement.

Le tableau 5.3 montre les valeurs moyennes des résultats obtenus des 5 exécutions. D'après le tableau

(a) Flux 1



(b) Flux 2



(c) Flux 3

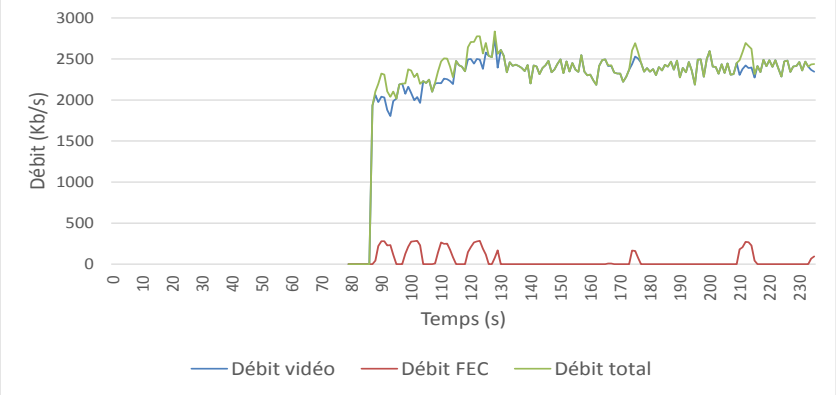


FIGURE 5.9 – Équité entre plusieurs flux vidéo - Meilleure exécution

5.3, nous remarquons que les valeurs des PSNR et des SSIM pour les trois flux sont presque égales. Ceci signifie que la qualité de la vidéo offerte par les 3 flux est presque identique. Les débits moyens du trafic FEC envoyé des différents flux sont aussi très proches. Le flux 3 a un taux plus élevé que les autres, car au début

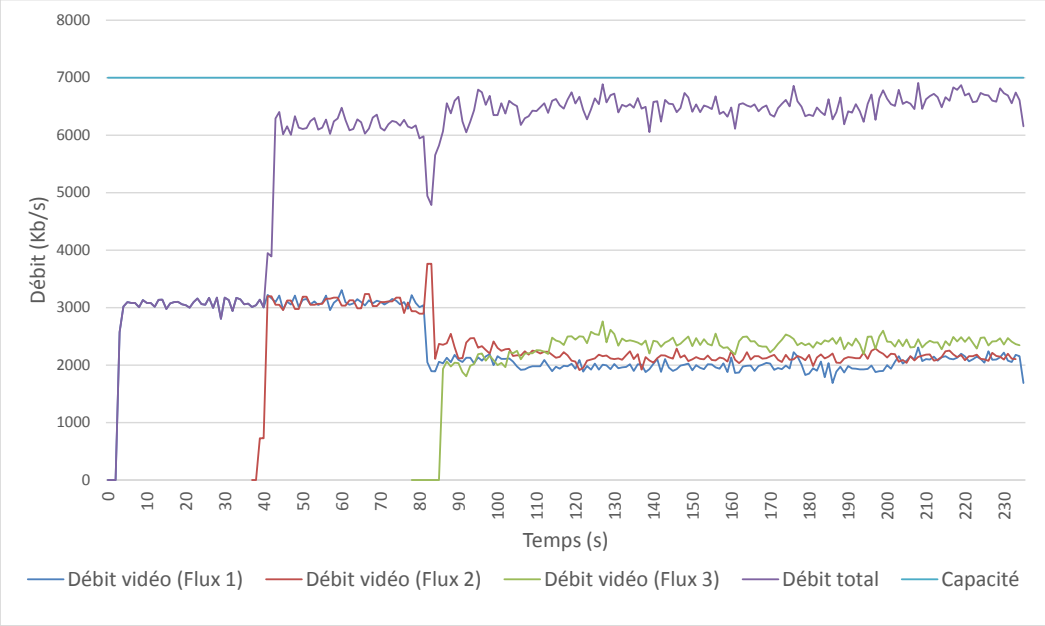


FIGURE 5.10 – Les résultats de la meilleure exécution - 3 flux

Tableau 5.3 – Tableau des résultats du Test 4

	Flux 1	Flux 2	Flux 3
Débit vidéo moyen (Kb/s)	1981	2123	2341
Taux de perte (%)	1,32	1,20	0,81
Débit de FEC (Kb/s)	34,63	29,66	57,68
PSNR (dB)	39,41	39,75	39,86
SSIM	0,951	0,958	0,957

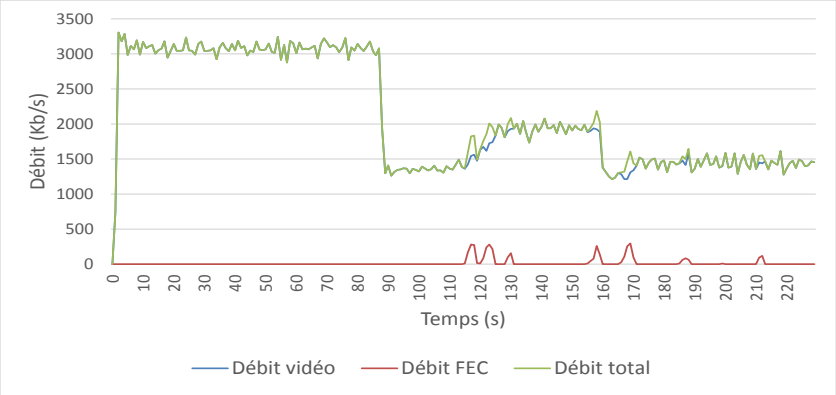
de son exécution le flux commence avec un débit inférieur aux débits des autres flux et utilise plus de sondes FEC pour améliorer sa qualité. Nous remarquons aussi que le taux de pertes des flux 1 et 2 est légèrement supérieur à celui du flux 3. En effet, ce dernier utilise la phase MaxS au démarrage. Cette phase provoque une congestion sévère qui augmente énormément le taux de perte des flux 1 et flux 2. D'après les différents résultats obtenus, nous pouvons conclure que notre algorithme a réussi ce test.

5.4.5 Test 5 : Concurrence entre un flux vidéo et plusieurs flux TCP de longue durée

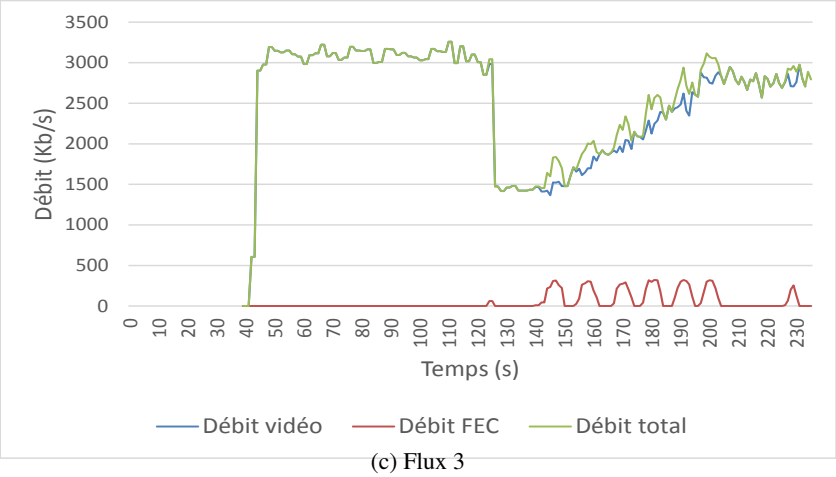
Description générale

Dans ce cas de test, un flux vidéo utilisant notre algorithme d'adaptation partage le goulot d'étranglement avec plusieurs flux TCP de longue durée. Ce scénario peut se produire lorsque l'application multimédia

(a) Flux 1



(b) Flux 2



(c) Flux 3

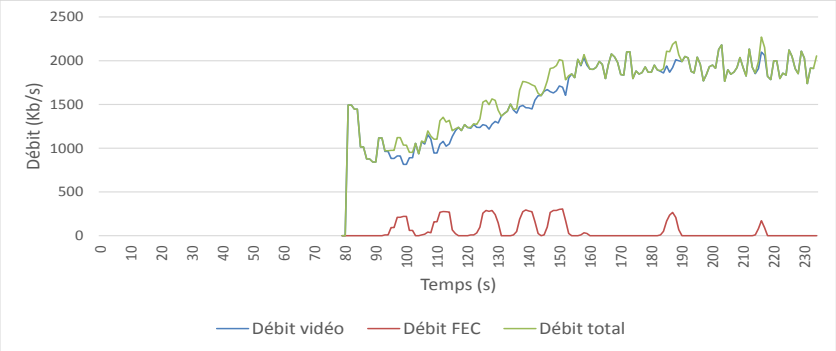


FIGURE 5.11 – Équité entre plusieurs flux vidéo - Pire exécution

coexiste avec des téléchargements de fichiers de grandes tailles. Le but de ce test est d'évaluer le niveau d'adaptabilité et le dégrée de compétitivité contre les flux TCP de longue durée.

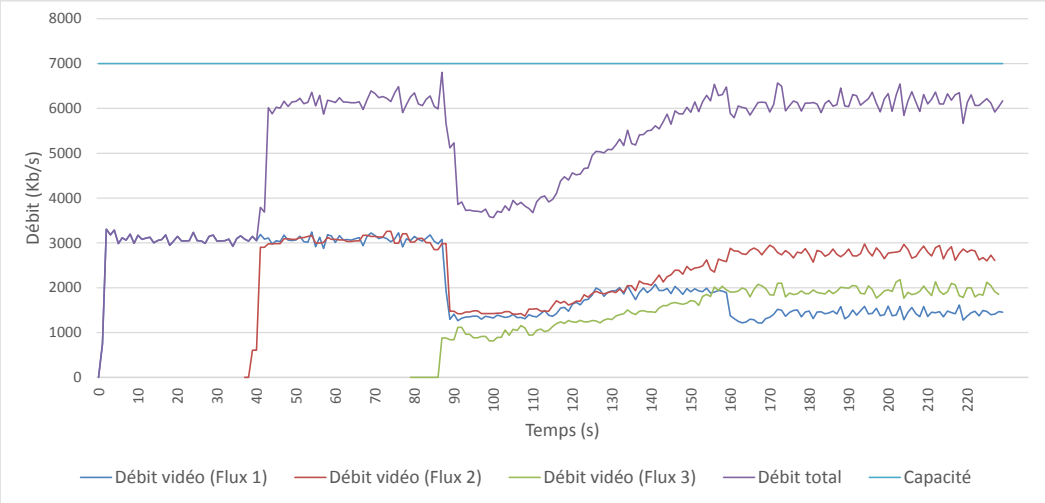


FIGURE 5.12 – Les résultats de la pire exécution - 3 flux

Banc de test

La topologie du banc de test est présentée dans la figure 5.13. Il est composé d'un émetteur vidéo, un récepteur vidéo, un serveur et 4 autres clients. Ces clients téléchargent des fichiers de grandes tailles depuis le récepteur. Au début du test, nous démarrons le flux vidéo entre l'émetteur et le récepteur. Après 40 s, nous démarrons le premier flux TCP de longue durée. Ensuite, après 40 s, nous démarrons 3 flux TCP supplémentaires. À partir de 100 s, 5 flux (1 flux vidéo et 4 flux TCP) partagent le même goulot d'étranglement. Finalement, nous arrêtons tous les flux TCP après l'écoulement de 170 s depuis le début du test. La liste détaillée des valeurs des paramètres de ce banc de test est présentée dans l'annexe A.

Résultats et observations

La figure 5.14 présente les résultats de ce test. Grâce à MaxS, l'algorithme commence avec le profil maximal, car il y a assez de bande passante. À 40 s, le premier client commence le téléchargement d'un fichier depuis le serveur. Ce flux TCP de longue durée provoque un faible taux de perte de paquets en rafale, comme présenté dans la section 4.2.3. Afin de réduire l'effet de la perte de paquets, l'algorithme passe à l'état PROTECT : Il commence donc à augmenter le taux de FEC et à passer à des profils de plus faible qualité. A 79 s, l'algorithme se stabilise. La perte de paquets est totalement compensée en utilisant un taux de FEC égal à 36 % du trafic vidéo. Pendant la stabilisation, l'algorithme reste dans l'état HOLD et collecte les nouvelles statistiques reportées par les rapports RTCP RR sans changer aucun paramètre. À 100 s, les 3 autres

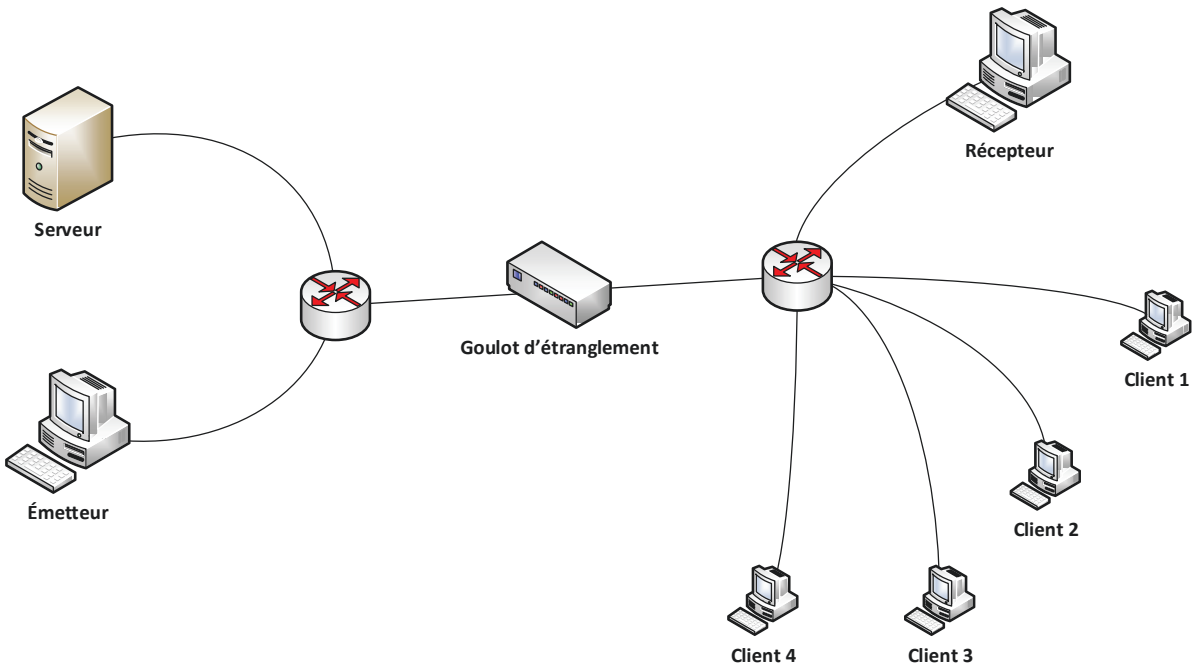


FIGURE 5.13 – Architecture du banc de test 5

clients commencent le téléchargement des fichiers à partir du serveur. À ce moment, 4 flux TCP et le flux vidéo traversent le même goulot d'étranglement. Ceci provoque un niveau de congestion très élevé qui oblige l'algorithme à passer à l'état DECREASE. L'algorithme remplace le profil courant par un profil de plus basse qualité. Il passe ensuite à l'état PROTECT pour augmenter la robustesse du flux vidéo et le protéger contre les pertes causées par les flux TCP concurrents. L'algorithme augmente donc le taux de FEC graduellement. À 150 s, l'algorithme passe à l'état DECREASE et passe à un profil plus bas, car l'augmentation du taux de FEC ne permet pas de compenser toutes les pertes. Après la diminution du débit vidéo, l'algorithme se stabilise. Il utilise un débit vidéo égal à environ 1500 Kb/s et un taux de FEC légèrement supérieur à 75 % du trafic vidéo. Lorsque nous arrêtons les flux TCP à 170 s, le taux de perte s'annule. En conséquence, l'algorithme attend 5 s et passe à l'état INCREASE. Il commence donc à remplacer le trafic FEC par du trafic vidéo. En d'autres termes, il diminue le taux de FEC utilisé pour la protection et remplace le profil utilisé par des profils de meilleure qualité progressivement. Une fois le trafic FEC utilisé pour la protection annulé (à 195 s), l'algorithme passe à l'état PROBE pour envoyer des sondes FEC et utilise l'état INCREASE pour passer à des profils de meilleure qualité jusqu'à atteindre le profil P_{max}.

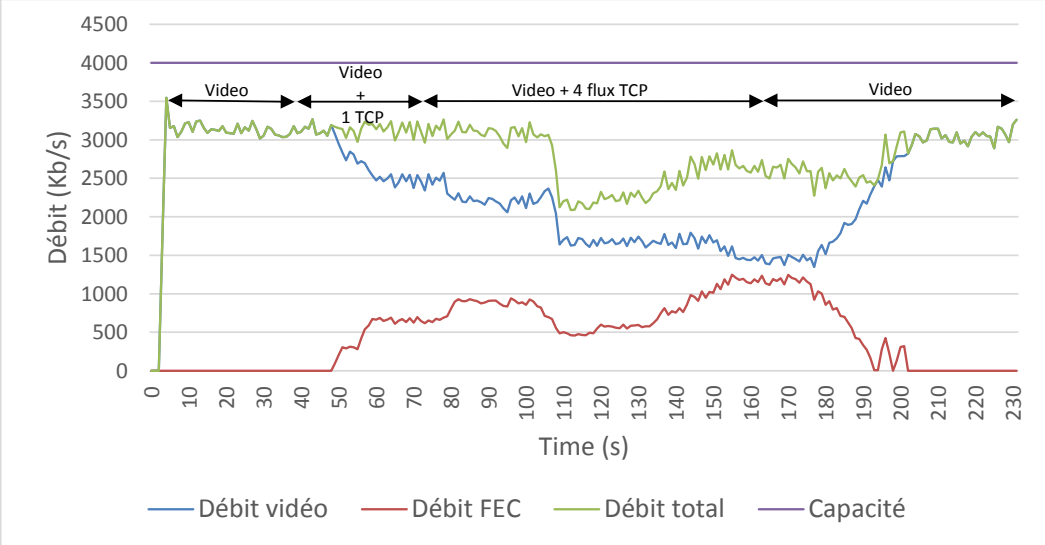


FIGURE 5.14 – Concurrence entre un flux vidéo et plusieurs flux TCP de longue durée

5.4.6 Test 6 : Concurrence entre un flux vidéo et plusieurs flux TCP de courte durée

Description générale

Dans ce cas de test, un flux vidéo utilisant notre algorithme partage le goulot d'étranglement avec plusieurs flux de courte durée. Les flux de courte durée sont générés par un client qui télécharge plusieurs pages Web depuis un serveur distant. Ce scénario peut se produire lorsque plusieurs utilisateurs qui naviguent sur Internet partagent le même lien avec un autre utilisateur qui est en train de communiquer par vidéo en utilisant notre algorithme. Le but de ce test est d'évaluer le niveau d'adaptabilité et le degré de compétitivité contre les flux TCP de courte durée.

Banc de test

La topologie du banc de test est présentée dans la figure 5.15. Il est composé d'un émetteur, un récepteur, un client et un serveur Web. Le client utilise plusieurs fenêtres de son navigateur pour télécharger des pages depuis le serveur Web. Les pages contiennent un contenu textuel accompagné de quelques images. La liste détaillée des valeurs des paramètres de ce banc de test est présentée dans l'annexe A.

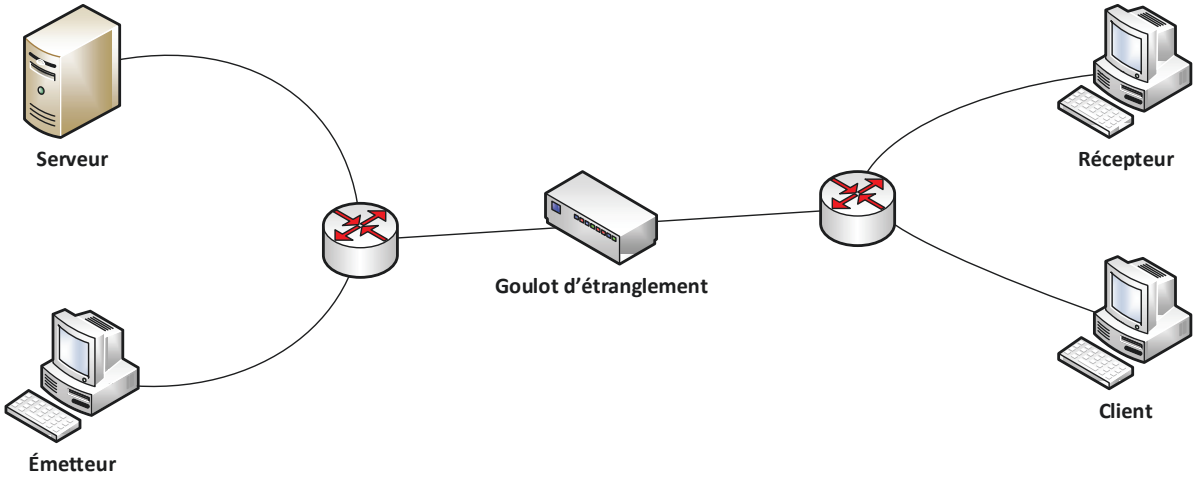


FIGURE 5.15 – Architecture du banc de test 6

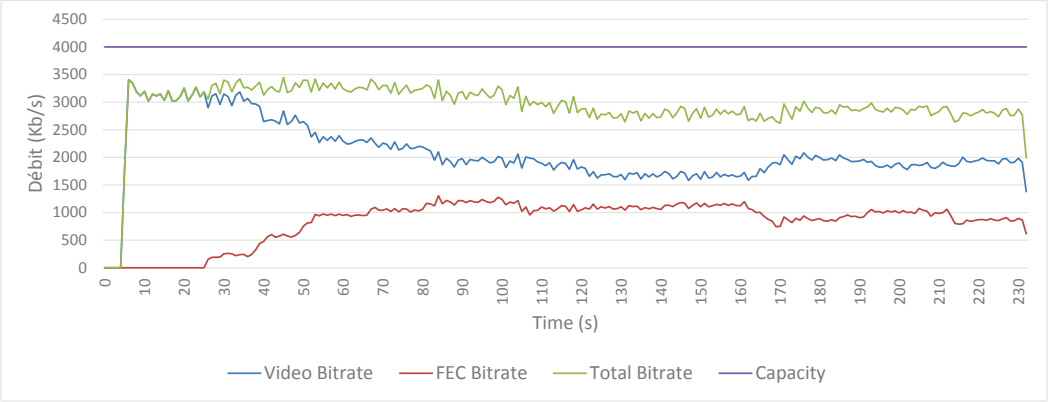


FIGURE 5.16 – Concurrence entre un flux vidéo et plusieurs flux TCP de courte durée

Résultats et observations

La figure 5.16 présente les résultats de ce test. Lorsque le navigateur du client charge une page, plusieurs flux TCP de courte durée sont initiés. Ces flux causent une augmentation du taux de perte du flux vidéo. L'algorithme passe donc à l'état PROTECT afin de compenser les paquets perdus. L'algorithme augmente donc le taux de FEC graduellement et réduit le débit de la vidéo jusqu'à l'annulation du taux de perte résiduelle de paquets. Si le taux de perte diminue, l'algorithme passe à l'état INCREASE, augmente la qualité de la vidéo et réduit le taux de FEC graduellement (figure 5.16 à 163 s). Si le taux de perte résiduelle augmente, l'algorithme repasse au dernier profil utilisé et augmente le taux de FEC (figure 5.16 à 190 s).

Ensuite, il passe à l'état HOLD. Grâce à l'état PROTECT, l'algorithme a réussi à offrir une bonne QoE malgré la présence de plusieurs flux TCP de courte durée.

5.4.7 Test 7 : Lien de retour congestionné

Description générale

Dans ce cas test, nous évaluons l'adaptabilité, la stabilité et le temps de réponse pendant le changement de la capacité du lien montant lorsque le lien descendant est congestionné. En effet, les rapports de feed-back sont envoyés depuis le récepteur vers l'émetteur en utilisant le lien descendant. Lorsque ce lien est congestionné, il est possible que quelques rapports soient perdus ou retardés. Ce test permet donc de voir l'effet de la congestion du lien montant.

Banc de test

La topologie du banc de test est présentée dans la figure 5.17. Il est composé d'un émetteur, d'un récepteur, un client et un serveur. Le client télécharge un fichier de grande taille depuis le serveur en utilisant le lien montant. Le téléchargement commence au début et dure pendant tout le scénario. Ce test est exécuté deux fois. La première fois, nous démarrons un seul flux TCP de longue durée. Dans ce cas le lien de descendant est donc moyennement congestionné. La deuxième fois, nous démarrons 4 flux TCP de longue durée simultanément. Dans ce cas le lien de retour est donc sévèrement congestionné. La liste détaillée des valeurs des paramètres de ce banc de test est présentée dans l'annexe A.

Résultats et observations

La figure 5.18 présente les résultats de l'exécution de ce test en utilisant un lien de retour moyennement congestionné et la figure 5.19 présente les résultats de l'exécution de ce test en utilisant un lien de retour sévèrement congestionné.

Nous remarquons depuis la figure 5.18 que l'adaptation au changement de la capacité est plus lente que l'adaptation dans le test présenté dans la section 5.4.3, où le lien de feed-back n'était pas congestionné. L'utilisation de RTT en tant qu'indice de congestion est la raison de ce comportement. En effet, comme nous

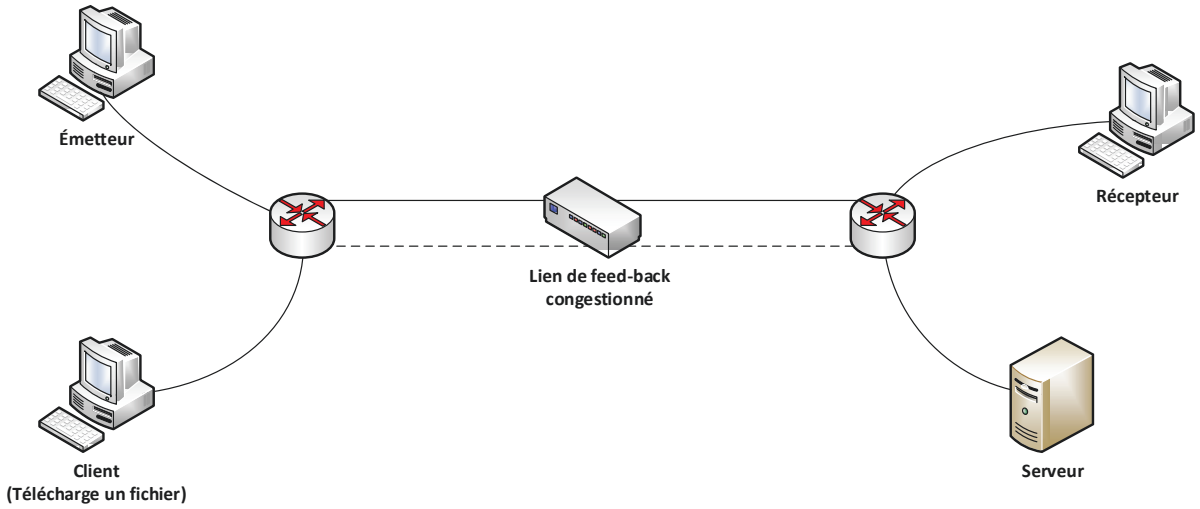


FIGURE 5.17 – Architecture du banc de test 7

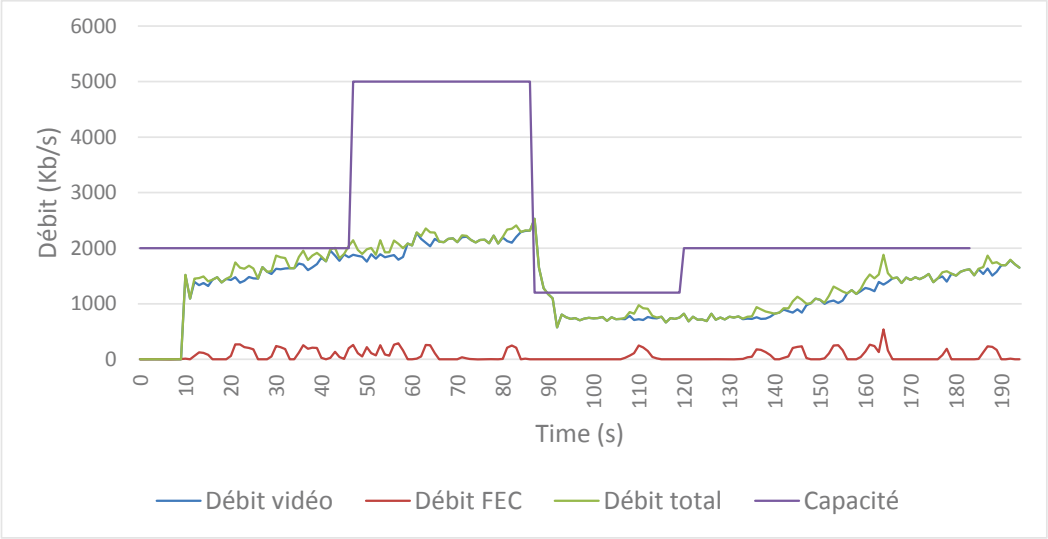


FIGURE 5.18 – Lien de retour moyennement congestionné

l'avons déjà indiqué dans la section 4.2.3, si la RTT annoncée est largement supérieure à la médiane des 10 dernières valeurs de RTT, l'algorithme n'envoie pas de nouvelles sondes et diminue le taux de FEC utilisé pour sonder. Le flux TCP de longue durée provoque des variations successives du RTT. Si les variations sont élevées, l'algorithme n'envoie pas de sonde FEC ce qui l'oblige à s'adapter lentement à l'augmentation de la capacité.

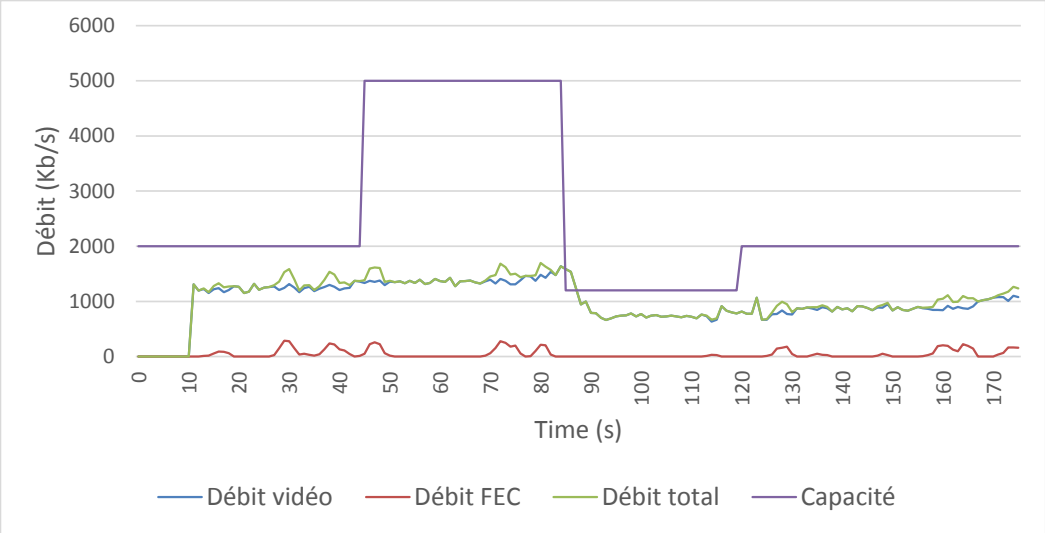


FIGURE 5.19 – Lien de retour sévèrement congestionné

Tableau 5.4 – Tableau des résultats du Test 7

	Lien de retour non congestionné	Lien moyennement congestionné	Lien sévèrement congestionné
Débit vidéo moyen (Kb/s)	1657	1384	1063
Taux de perte (%)	0,98	0,71	0,40
Débit de FEC (Kb/s)	66,36	68,53	46,25
PSNR (dB)	38,72	37,95	36,83
SSIM	0,941	0,936	0,930

Si la congestion est très sévère, comme c'est le cas dans la deuxième exécution de ce test, plusieurs sondes successives seront considérées comme ayant échoué. En conséquence, l'algorithme utilise le mode prudent au lieu du mode normal. Ceci rend l'adaptation plus lente, comme montré dans la figure 5.19. Ce problème affecte uniquement les états INCREASE et PROBE. Pour les autres états, l'algorithme fonctionne comme prévu.

Le tableau 5.4 montre les résultats obtenus en utilisant un lien de retour moyennement congestionné et un lien de retour sévèrement congestionné par rapport au résultat d'un test ayant un lien de feed-back non congestionné. D'après ces résultats, nous remarquons que le débit moyen diminue énormément lorsque le lien de retour est congestionné. Il diminue de plus de 250 Kb/s pour un lien de feed-back moyennement congestionné et de presque 600 Kb/s pour le lien sévèrement congestionné. Cette diminution du débit utilisé entraine une dégradation dans la qualité offerte. Ceci peut être observé par la diminution des valeurs de PSNR

et SSIM. Nous remarquons que plus le lien est congestionné plus la perte de paquets diminue. Ceci s'explique par le fait que, lorsque le lien de feed-back est congestionné, l'algorithme ne réussit pas à augmenter son débit lorsque nous augmentons la bande passante disponible à 45 s. En conséquence, quand nous diminuons la bande passante disponible à 85 s, le niveau de congestion sera plus faible.

D'après les résultats obtenus, nous concluons que notre algorithme a échoué à ce test. Pour le réussir, il faut utiliser l'OWD en tant qu'indice de congestion au lieu de la RTT.

5.5 Comparaison de notre algorithme avec GCC

Dans cette section, nous exécutons quelques scénarios de tests en utilisant l'algorithme GCC et nous comparons les résultats avec ceux obtenus en utilisant notre algorithme. Il faut noter que nous avons fait une première comparaison avec la version 34M de WebRTC. Dans cette comparaison, les résultats de l'algorithme GCC ont été largement inférieurs aux résultats de notre algorithme. Les résultats de GCC se sont améliorés dans la version 36M, qui a apparu fin juillet 2014. Malgré les améliorations effectuées, notre algorithme offre de meilleures performances dans la majorité des scénarios. Les résultats présentés dans cette section sont obtenus en utilisant la version 36M de WebRTC.

5.5.1 Capacité de lien variable

La description du scénario de ce test est présentée dans la section 5.4.2 La seule différence est que les clients WebRTC utilisent l'algorithme GCC au lieu de notre algorithme pour adapter le débit vidéo aux conditions du réseau. La figure 5.20 montre les résultats de l'exécution de ce test en utilisant notre algorithme et l'algorithme GCC.

D'après la figure 5.20, nous remarquons que GCC prend beaucoup de temps pour adapter le débit vidéo à la bande passante disponible initiale. En effet, contrairement à notre algorithme, GCC n'a pas de mécanisme spécifique au démarrage. Lorsqu'on diminue la capacité du goulot d'étranglement, GCC réussit à s'adapter rapidement, mais il sous-estime la bande passante disponible. Il utilise donc un débit vidéo très bas par rapport à celui utilisé par notre algorithme. Ensuite, il améliore la qualité de la vidéo lentement. Après la diminution de la capacité, notre algorithme offre une meilleure qualité pendant plus de 50 secondes. À 156 s,

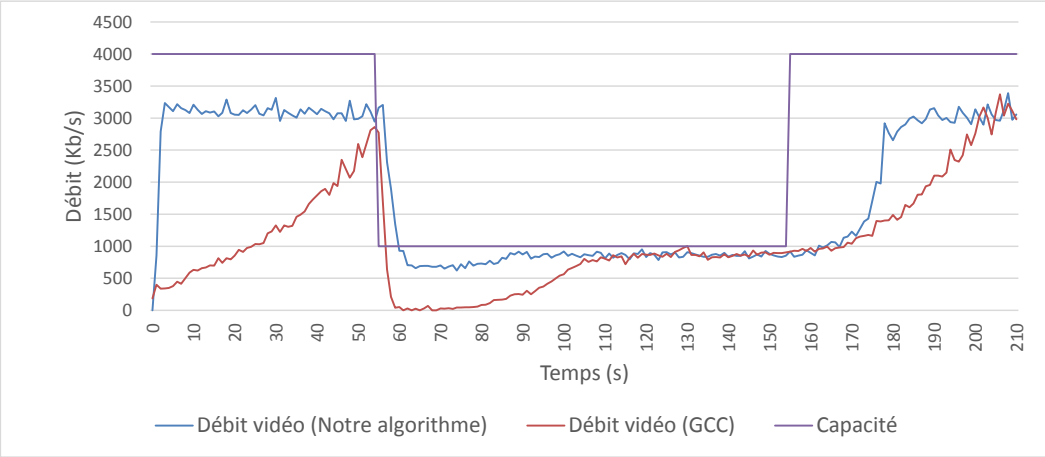


FIGURE 5.20 – Capacité de lien variable - Comparaison avec GCC

nous augmentons la capacité à 4000 Kb/s. Après cette augmentation, notre algorithme réussit à s'adapter plus rapidement grâce au mode agressif.

Dans ce scénario, notre algorithme réussit à offrir une vidéo ayant une meilleure qualité que celle offerte par GCC.

5.5.2 Capacité de lien variable (variations multiples)

La description du scénario de ce test est présentée dans la section 5.4.3. La seule différence est que les clients WebRTC utilisent l'algorithme GCC au lieu de notre algorithme pour adapter le débit vidéo aux conditions du réseau. La figure 5.21 montre les résultats de l'exécution de ce test en utilisant notre algorithme et l'algorithme GCC.

Comme indiqué dans le scénario précédent, GCC prend beaucoup de temps pour s'adapter aux conditions initiales du réseau. Après la diminution de la capacité du goulot d'étranglement, GCC offre une qualité inférieure à celle offerte par notre algorithme pour une courte durée. Dans les autres parties, notre algorithme et GCC utilisent des débits vidéo très proches.

Dans ce scénario, notre algorithme offre une qualité légèrement meilleure de celle offerte par GCC.

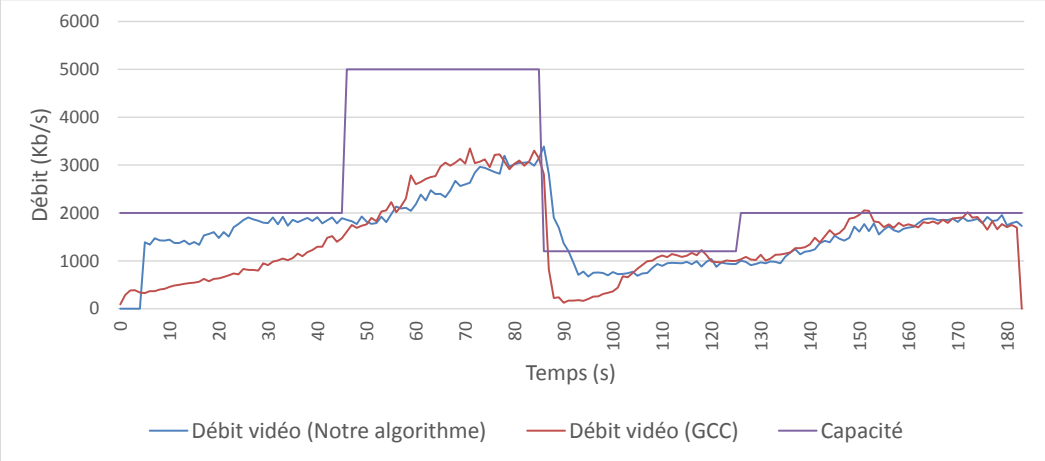


FIGURE 5.21 – Capacité de lien variable (variations multiples) - Comparaison avec GCC

5.5.3 Concurrence entre un flux vidéo et plusieurs flux TCP de longue durée

La description du scénario de ce test est présentée dans la section 5.4.5. La seule différence est que les clients WebRTC utilisent l'algorithme GCC au lieu de notre algorithme pour adapter le débit vidéo aux conditions du réseau. La figure 5.22 montre les résultats de l'exécution de ce test en utilisant notre algorithme et l'algorithme GCC.

D'après la figure 5.22, notre algorithme utilise un débit vidéo supérieur à celui offert par GCC. En conséquence, notre algorithme offre une meilleure qualité par rapport GCC lorsque le goulot d'étranglement est partagé avec des flux TCP de longue durée. Lorsque nous arrêtons les flux TCP à 170 s, GCC s'adapte plus rapidement que notre algorithme. Il est possible d'adapter le débit vidéo plus rapidement que GCC en remplaçant tout le trafic FEC de protection par du trafic vidéo. Cependant, comme nous l'avons indiqué dans la section 4.3.4, nous avons choisi de remplacer le trafic FEC de protection progressivement pour augmenter la qualité d'une façon harmonieuse et pour éviter la dégradation de la qualité lorsque le nouveau taux de FEC ne permet plus de récupérer tous les paquets.

5.5.4 Lien de retour sévèrement congestionné

La description du scénario de ce test est présentée dans la section 5.4.7. La seule différence est que les clients WebRTC utilisent l'algorithme GCC au lieu de notre algorithme pour adapter le débit vidéo aux

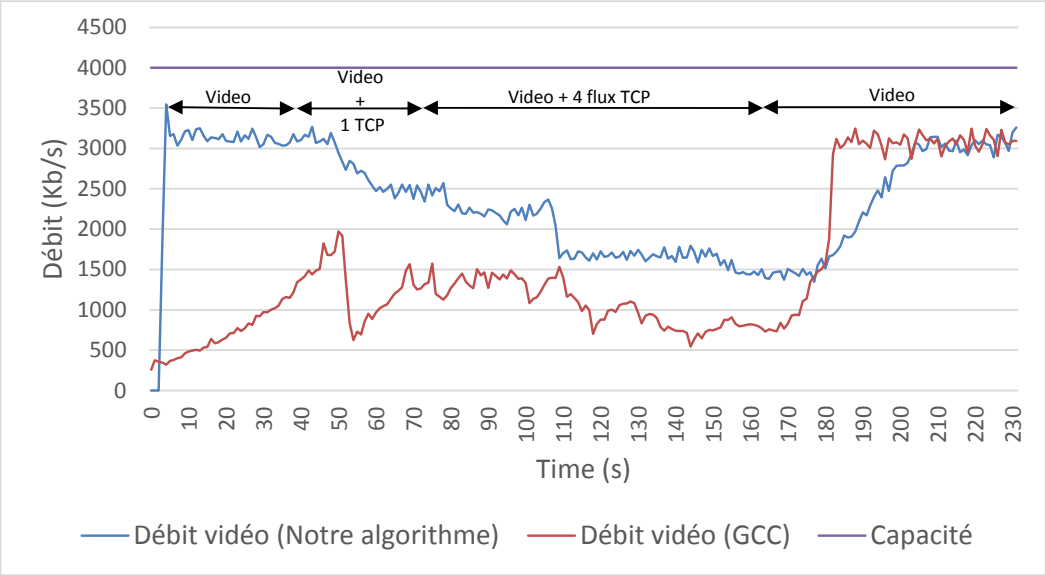


FIGURE 5.22 – Concurrence entre un flux vidéo et plusieurs flux TCP de longue durée - Comparaison avec GCC)

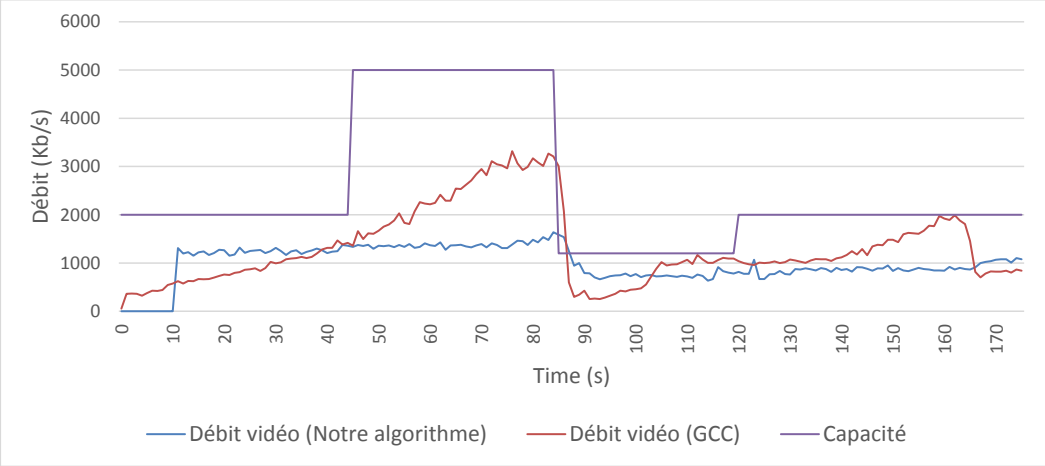


FIGURE 5.23 – Lien de retour sévèrement congestionné - (Comparaison avec GCC)

conditions du réseau. La figure 5.23 montre les résultats de l'exécution de ce test en utilisant notre algorithme et l'algorithme GCC.

L'algorithme GCC s'adapte plus rapidement à la variation de la bande passante disponible lorsque le lien de feed-back est congestionné. Ainsi, il offre une meilleure qualité. La congestion du lien de retour n'influence pas l'algorithme GCC car il contient une partie de contrôle côté récepteur. Cette partie utilise le temps d'inter-arrivée des paquets. Ce temps n'est pas affecté lorsque le lien de retour est congestionné.

5.6 Conclusion

Dans ce chapitre, nous avons détaillé la démarche suivie afin d'évaluer les performances de notre algorithme d'adaptation dynamique aux conditions du réseau. Nous avons également présenté les différents cas de test exécuté pour évaluer ces différents critères et dévoilé les résultats obtenus pour chaque test. Notre algorithme a réussi la majorité des tests et montré de très bonnes performances. Il a ainsi répondu à la majorité d'exigences définies par le groupe RMCAT. Il faut noter aussi que notre algorithme a échoué à un test, celui du lien de feed-back congestionné. Nous avons pu détecter la principale raison de cet échec, ce qui va nous permettre de le corriger dans une prochaine version et développer une nouvelle version plus performante de notre algorithme.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion

Les services de communications interactives sur Internet permettent d'offrir une expérience de communication très riche à faible coût et c'est pourquoi le nombre d'utilisateurs de cette technologie a explosé. Les services les plus utilisés de nos jours sont les services de communication interactive de voix et de messagerie, car ils peuvent offrir une bonne QoE même si les ressources disponibles sur l'Internet sont limitées. Les services de communication vidéo interactive sont les plus exigeants en matière de ressources. Pour fonctionner convenablement et offrir une bonne qualité, il faut utiliser un réseau qui peut garantir une bande passante disponible suffisante, un faible délai, une très faible gigue et une perte de paquets presque nuls. Le réseau Internet est un réseau très hétérogène partagé entre un nombre énorme d'utilisateurs. Il est conçu d'une façon lui permettant de faire de son mieux afin de délivrer les informations vers leur destination, mais il ne permet pas de garantir constamment les ressources nécessaires pour le bon fonctionnement des services de communications vidéo interactives. Pour assurer le bon fonctionnement de ces services, plusieurs technologies et protocoles ont été créés. Malgré l'utilisation de ces technologies, la majorité des services existants ne peuvent pas offrir un bon niveau de qualité de service lorsqu'ils sont exécutés sur le réseau Internet. Ceci est dû à la variation rapide et continue de la disponibilité des ressources. Avec l'augmentation exponentielle du nombre des utilisateurs des services de communication sur Internet, il est devenu important d'offrir une QoE qui correspond aux attentes des utilisateurs.

Dans ce mémoire, nous avons proposé un algorithme d'adaptation dynamique du débit pour les communications vidéo interactives sur Internet. L'algorithme estime la bande passante disponible et il maximise le débit utilisé pour améliorer la qualité de la vidéo sans créer une congestion. Si une congestion se produit, l'algorithme compresse la vidéo afin de diminuer le débit utilisé, ce qui permet de supprimer la congestion et éviter ses effets nocifs pour la qualité. Afin d'estimer la bande passante disponible, notre algorithme ajoute un trafic FEC en parallèle au trafic vidéo. Si ce trafic ne cause pas de perte de paquets ou une augmentation de délai, il remplace le trafic redondant par un trafic vidéo utile. Il comporte aussi une phase de démarrage lui permettant d'adapter le débit aux conditions initiales du réseau très rapidement. En utilisant plusieurs indices de congestion, notre algorithme peut détecter l'occurrence d'une congestion. À ce moment-là, il réduit la bande passante utilisée en compressant davantage la vidéo envoyée. Il permet aussi de détecter la différence entre une congestion sévère, qui peut être due à une capacité limitée, ou une congestion causée par un trafic TCP concurrent. Dans la deuxième situation, notre algorithme utilise le trafic FEC pour augmenter la robustesse du trafic vidéo et le rendre moins sensible aux pertes qui sont provoquées par le trafic TCP concurrent. Ce mécanisme lui permet de garantir une bonne QoE dans les réseaux chargés.

Pour évaluer notre algorithme, nous avons développé une application Web de communication vidéo interactive en utilisant l'API WebRTC qui utilise notre algorithme pour adapter le débit de la vidéo émis aux conditions du réseau. Pendant la phase d'évaluation, nous avons suivi les directives du groupe RMCAT. En conséquence, nous avons créé plusieurs bancs de test, chacun d'eux émulant une condition spécifique qui peut se produire dans le réseau Internet. Ensuite, nous avons testé notre algorithme sur ces différents bancs de test.

Notre algorithme a réussi la majorité des tests. En effet, les résultats obtenus ont montré son efficacité à s'adapter rapidement aux conditions du réseau et son rôle dans l'amélioration de la QoE offerte. Pendant la phase démarrage, il a pu s'adapter très rapidement aux conditions initiales du réseau. Il a pu aussi adapter le débit de la vidéo aux différentes variations de la bande passante disponible. Il a réussi aussi à offrir une bonne QoE pendant la présence de plusieurs flux TCP de courtes et longues durées concurrents. La qualité offerte a été bonne même dans des conditions de concurrence très sévère, comme le fait d'avoir 4 flux concurrents qui partagent un goulot d'étranglement de capacité très limité (4000 kbit/s). Malgré son caractère agressif face aux autres flux concurrents, un ensemble de flux vidéo utilisant notre algorithme sont équitables entre eux.

6.2 Perspectives

Dans le chapitre 5, nous avons découvert que notre algorithme n'offre pas les performances souhaitables lorsqu'il est exécuté dans un réseau ayant un lien de feed-back congestionné. La principale cause de ce résultat est le fait que nous avons utilisé le RTT en tant qu'indice de congestion. En effet, lorsque le lien de feed-back est congestionné, le RTT varie énormément. Cette variation oblige l'algorithme à diminuer ou annuler le taux de FEC utilisé en tant que sonde, ce qui ralentit l'amélioration de la qualité de la vidéo lorsqu'il y a une augmentation dans la bande passante disponible. Afin de résoudre ce problème, nous pourrons remplacer le RTT par l'OWD. Cette métrique permet de mesurer le délai de transmission sur chaque lien séparément. Ainsi, les performances de l'algorithme ne seront plus affectées par la congestion du lien de feedback. Le défi principal du calcul de l'OWD est que l'émetteur et le récepteur doivent utiliser des horloges synchronisées. La synchronisation des horloges de l'émetteur et du récepteur n'est pas une tache évidente à implémenter dans une application de communication vidéo interactive destinée au grand public. Pour pouvoir utiliser l'OWD, il est donc nécessaire de trouver une méthode qui peut donner une estimation précise de l'OWD sans la nécessité de la synchronisation. Une première voie est de faire une investigation sur la méthode proposée dans [73] et l'implémenter.

Dans ce mémoire, nous avons évalué les performances de notre algorithme dans plusieurs bancs de test en suivant les différents critères énoncés par le groupe RMCAT. Il est intéressant de voir les performances des nouveaux algorithmes expérimentaux, comme FBRA et GCC dans ces bancs de tests et de comparer les différents résultats obtenus avec les résultats de notre algorithme.

L'évaluation a été faite dans des bancs de test émulant plusieurs cas particuliers qui peuvent se produire dans un environnement réel. Un algorithme qui offre une bonne QoE dans les différents tests exécutés dans un environnement expérimental peut ne pas offrir le même bon niveau de QoE lorsqu'il est exécuté dans un environnement réel (dans un vrai réseau Internet). Pour cela, il faut tester notre algorithme en utilisant un environnement réel. Il est aussi nécessaire de le tester dans plusieurs types de réseaux d'accès et dans des réseaux de différents opérateurs. Le problème de ce genre de test est que nous n'avons pas le contrôle sur ces réseaux. Nous ne pouvons pas donc deviner les raisons des décisions qui pourraient être prises par notre algorithme. Ainsi, au lieu de se concentrer sur le temps d'adaptation aux conditions du réseau. Il sera mieux de comparer la QoE offerte par notre algorithme au QoE offerte par FBRA et GCC. Même en utilisant cette approche, nous pouvons avoir une différence énorme entre deux exécutions successives. En conséquence, il faut exécuter des tests de longue durée et dans de différents intervalles de temps pendant la journée. Les tests

doivent être aussi répétés plusieurs fois en alternant entre les différents algorithmes. Ceci permet d'avoir des valeurs moyennes et de minimiser l'effet des évènements brusques et rares qui peuvent se produire pendant les différents tests.

À l'heure actuelle, plusieurs nouveaux codecs audio ont été créés (p. ex. Opus [66], AMR-WB [74], etc.). Ces codecs ont aussi des paramètres configurables permettant d'améliorer ou de dégrader la qualité offerte en augmentant ou diminuant le taux de compression. Par exemple, le codec Opus peut supporter un débit audio allant de 6 kb/s jusqu'à 512 kb/s. Il sera donc intéressant d'adapter notre méthode au trafic audio utilisant ces nouveaux codecs. Ceci permettra donc d'avoir une qualité audio adaptative aux conditions du réseau. Dans cette perspective, l'algorithme proposé doit se concentrer sur l'adaptation du couple flux vidéo / flux audio aux conditions du réseau et essayer d'offrir la meilleure QoE possible. Il doit donc avoir un nouveau module intelligent permettant de décider quand il est préférable d'améliorer la qualité du trafic audio et quand il est préférable d'améliorer la qualité vidéo. Il est aussi nécessaire d'étudier l'effet des sondes FEC d'un flux audio/vidéo sur l'autre flux vidéo/audio.

Références

[1] ITU, "ICT Facts and Figures," ITU, Brochure, April 2014. [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf

[2] Mobilesquared, "OTT Services Blow Up the Mobile Universe. Operators Must Act NOW!" Tyntec, Whitepaper, Septembre 2013. [Online]. Available: https://www.tyntec.com/fileadmin/tyntec.com/whitepapers/Whitepaper_Operator-survey-OTT-blows-up-mobile-universe.pdf

[3] GSMA, "Rich Communication Suite RCS API Detailed Requirements," GSMA, Detailed Requirements, June 2014. [Online]. Available: http://www.gsma.com/network2020/rcs/specs-and-product-docs/

[4] Zurawski Richard, *The industrial information technology handbook*. CRC Press, 2004.

[5] Yan Chen, Toni Farley et Nong Ye, "QoS requirements of network applications on the Internet," *Information, Knowledge, Systems Management*, vol. 4, no. 1, pp. 55–76, 2004.

[6] J. Postel, "User Datagram Protocol," RFC 768, Août 1980.

[7] Jenq-Neng Hwang, *Multimedia Networking: From Theory to Practice*. Cambridge University Press, 2009.

[8] Jonathan Rosenberg, Hennmg Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley et autres, "SIP: session initiation protocol," RFC 3261, Juin 2002.

[9] Li Ze-Nian, Mark S. Drew et Jiangchuan Liu, *Fundamentals of Multimedia*. Springer, 2nd ed. 2014.

[10] Iain E. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2nd ed. 2010.

[11] Henning Schulzrinne, Stephen L. Casner, Ron Frederick et Van Jacobson, "RTP: a transport protocol for real-time applications," RFC 3550, Juillet 2003.

[12] David D. Clark et David L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4. ACM, 1990, pp. 200–208.

[13] Henning Schulzrinne et Stephen L. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control," RFC 3551, Juillet 2003.

[14] Mark Baugher, Elisabetta Carrara, David A. McGrew, Mats Naslund et Karl Norrman, "The Secure Real-time Transport Protocol (SRTP)," RFC 3711, Mars 2004.

[15] Jörg Ott, Stephan Wenger, Noriyuki Sato, Carsten Burmeister et Jose Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)," RFC 4585, Juillet 2006.

[16] Jörg Ott et Elisabetta Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)," RFC 5124, Février 2008.

[17] Rafael Cacheda, Daniel García, Antonio Cuevas, Francisco Castaño, Javier Sánchez et Georgios Koltsidas, "QoS requirements for multimedia services," in *Resource Management in Satellite Networks*. Springer, 2007, pp. 67–94.

[18] Jose Rey, Akihiro Miyazaki, Viktor Varsa et Rolf Hakenberg, “RTP Retransmission Payload Format,” RFC 4588, Juillet 2006.

[19] Colin Perkins, Magnus Westerlund et Jörg Ott, “Web Real-Time Communication (WebRTC): Media Transport and Use of RTP,” IETF, Internet Draft – work in progress 17, Août 2014. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rtcweb-rtp-usage/

[20] Charles Wang, Dean Sklar, Diana Johnson, “Forward Error-Correction Coding,” *The Aerospace Corporation magazine of advances in aerospace technology*, 2002.

[21] Richard Hamming, “Error detecting and error correcting codes,” *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[22] Jonathan Rosenberg et Henning Schulzrinne, “An RTP Payload Format for Generic Forward Error Correction,” RFC 2733, Décembre 1999.

[23] Adam H. Li, “RTP Payload Format for Generic Forward Error Correction,” RFC 5109, Décembre 2007.

[24] Mark Watson, Thomas Stockhammer et Michael Luby, “Raptor Forward Error Correction (FEC) Schemes for FECFRAME,” RFC 6681, Août 2012.

[25] Ali Begen, “RTP Payload Format for 1-D Interleaved Parity Forward Error Correction (FEC),” RFC 6015, Octobre 2010.

[26] Janusz Gozdecki, Andrzej Jajszczyk et Rafal Stankiewicz, “Quality of service terminology in IP networks,” *Communications Magazine, IEEE*, vol. 41, no. 3, pp. 153–159, 2003.

[27] Tom Henderson, Sally Floyd, Andrei Gurtov and Yoshifumi Nishida, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” RFC 6582, Avril 2012.

[28] Xu. Lisong, Harfoush Khaled et Rhee Injong, “Binary increase congestion control (BIC) for fast long-distance networks,” in *INFOCOM, Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2514–2524.

[29] Injong Rhee et Lisong Xu , “CUBIC: a new TCP-friendly high-speed TCP variant,” *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[30] Fabien Michaut et Francis Lepage, “Application-oriented network metrology: Metrics and active measurement tools,” *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, 2005.

[31] Ahmad Vakili et Jean-Charles Grégoire, “Accurate One-Way Delay Estimation: Limitations and Improvements,” *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 9, pp. 2428–2435, 2012.

[32] Ngamwongwattana Boonchai et Thompson Richard, “Sync & sense: VoIP measurement methodology for assessing one-way delay without clock synchronization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 5, pp. 1318–1326, 2010.

[33] Mo Zanaty, Varun Singh, Suhas Nandakumar et Zaheduzzaman Sarker, “RTP Application Interaction with Congestion Control,” IETF, Internet Draft – work in progress 00, Août 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rmcat-app-interaction/

[34] Colin Perkins et Varun Singh, “Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions,” IETF, Internet Draft – work in progress 06, Juillet 2014. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-avtcore-rtp-circuit-breakers

[35] Mark Handley, Sally Floyd, Jitendra Padhye et Joerg Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification,” RFC 3448, Janvier 2003.

[36] Sally Floyd, Mark Handley, Jitendra Padhye and Joerg Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification,” RFC 5348, Septembre 2008.

[37] Ladan Gharai et Colin Perkins, "RTP with TCP Friendly Rate Control," *Work in Progress*, 2007.

[38] Alvaro Saurin, "Congestion control for video-conferencing applications," *Master's Thesis, University of Glasgow*, 2006.

[39] Varun Singh, Jörg Ott et Igor D. D. Curcio, "Rate adaptation for conversational 3G video," in *INFOCOM Workshops*. IEEE, 2009, pp. 1–7.

[40] Michael Zink, Oliver Künzel, Jens Schmitt et Ralf Steinmetz, "Subjective impression of variations in layer encoded videos," in *International Workshop on Quality of Service (IWQoS 2003)*. Springer, 2003, pp. 137–154.

[41] Cullen Jennings, Ted Hardie et Magnus Westerlund, "Real-time communications for the web," *Communications Magazine, IEEE*, vol. 51, no. 4, pp. 20–26, 2013.

[42] Randell Jesup, "Congestion Control Requirements For RMCAT," IETF, Internet Draft – work in progress 05, Juillet 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rmcat-cc-requirements/

[43] Varun Singh et Jörg Ott, "Evaluating Congestion Control for Interactive Real-time Media," IETF, Internet Draft – work in progress 02, Juillet 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rmcat-eval-criteria/

[44] Xiaoqing Zhu, Rong Pan, Michael A. Ramalho, Sergio Mena de la Cruz, Charles Ganzhorn, Paul E. Jones et Stefano D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media," IETF, Internet Draft – work in progress 04, Septembre 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-zhu-rmcat-nada/

[45] Stefan Holmer, Luca De Cicco, Saverio Mascolo et Harald Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication," IETF, Internet Draft – work in progress 02, Août 2014. [Online]. Available: https://datatracker.ietf.org/doc/draft-alvestrand-rmcat-congestion/

[46] Varun Singh, Marcin Nagy, Jörg Ott et Lars Eggert, "Congestion Control Using FEC for Conversational Media," IETF, Internet Draft – work in progress 00, Juillet 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-singh-rmcat-adaptive-fec/

[47] Marcin Nagy, Varun Singh, Jörg Ott et Lars Eggert, "Congestion control using FEC for conversational multimedia communication," in *Proceedings of the 5th ACM Multimedia Systems Conference*. ACM, 2014, pp. 191–202.

[48] Luca De Cicco, Gaetano Carlucci and Saverio Mascolo, "Experimental investigation of the Google congestion control for real-time flows," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*. ACM, 2013, pp. 21–26.

[49] Stephan Wenger, Umesh Chandra, Magnus Westerlund et Bo Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)," RFC 5104, Février 2008.

[50] Harald Alvestrand, "RTCP message for Receiver Estimated Maximum Bitrate," IETF, Internet Draft – work in progress 03, Octobre 2013. [Online]. Available: https://tools.ietf.org/html/draft-alvestrand-rmcat-remb-03

[51] Varun Singh, Albert Abello Lozano et Jörg Ott, "Performance analysis of receive-side real-time congestion control for WebRTC," in *Proc. of IEEE Packet Video*, vol. 2013, 2013.

[52] Ahmad Vakili et Jean-Charles Grégoire, "QoE management for video conferencing applications," *Computer Networks*, vol. 57, no. 7, pp. 1726–1738, 2013.

[53] Svante Ekelin , Martin Nilsson, Erik Hartikainen, Andreas Johnsson, Jan-Erik Mångs, Bob Melander et Mats Björkman, "Real-time measurement of end-to-end available bandwidth using kalman filtering," in *Network Operations and Management Symposium (NOMS)*. IEEE, 2006, pp. 73–84.

[54] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil et Les Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and active measurement workshop*, vol. 4, 2003.

[55] Mingzhe Li, Mark Claypool et Robert Kinicki, "WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks," in *33rd IEEE Conference on Local Computer Networks (ICN)*. IEEE, 2008, pp. 374–381.

[56] Karthik Lakshminarayanan, Venkata N. Padmanabhan et Jitendra Padhye, "Bandwidth estimation in broadband access networks," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 314–321.

[57] Dimitrios Koutsonikolas et Y. Charlie Hu, "On the feasibility of bandwidth estimation in wireless access networks," *Wireless Networks*, vol. 17, no. 6, pp. 1561–1580, 2011.

[58] Robert Sedgewick et Kevin Wayne, *Algorithms (4th Edition)*. Addison-Wesley Professional, 2011.

[59] Christina Lagerstedt, Andreas Aurelius, Hemamali Pathirana, Claus Popp Larsen et Olle Findahl, "Unified methodology for broadband behavior measurements in the acreo national testbed," in *ICDT 2012, The Seventh International Conference on Digital Telecommunications*, 2012, pp. 96–100.

[60] H. Alvestrand, "Overview: Real Time Protocols for Browser-based Applications," IETF, Internet Draft – work in progress 11, Août 2014. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-overview-11

[61] Jonathan Rosenberg, Rohan Mahy, Philip Matthews et Dan Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, Octobre 2008.

[62] Rohan Mahy, Philip Matthews et Jonathan Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," RFC 5766, Avril 2010.

[63] Jonathan Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," RFC 5245, Avril 2010.

[64] Tina le Grand, Paul E. Jones, Pascal Huart, Turaj Zakizadeh Shabestary et Harald Alvestrand, "RTP Payload Format for the iSAC Codec," IETF, Internet Draft – work in progress 04, Février 2013. [Online]. Available: https://tools.ietf.org/html/draft-ietf-avt-rtp-isac-04

[65] Soren Vang Andersen, Alan Duric, Henrik Astrom, Roar Hagen, W. Bastiaan Kleijn et Jan Linden, "Internet Low Bit Rate Codec (iLBC)," RFC 3951, Décembre 2004.

[66] Rämö Anssi et Toukomaa Henri, "Voice Quality Characterization of IETF Opus Codec." in *INTERSPEECH*, 2011, pp. 2541–2544.

[67] James Bankoski, John Koleszar, Lou Quillio, Janne Salonen, Paul Wilkins et Yaowu Xu, "VP8 Data Format et Decoding Guide," RFC 6386, Novembre 2011 2011.

[68] Carbone, Marta et Rizzo, Luigi, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.

[69] Auriol Guillaume, "Spécification et implémentation d'une architecture de signalisation à gestion automatique de la QdS dans un environnement IP multi domaines," Ph.D. dissertation, INSA de Toulouse, 2004.

[70] Zhou Wang, Eero P. Simoncelli et Alan C. Bovik, "Multiscale structural similarity for image quality assessment," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, 2004*, vol. 2. IEEE, 2003, pp. 1398–1402.

[71] Richard Dosselmann et Xue Dong Yang, "A comprehensive assessment of the structural similarity index," *Signal, Image and Video Processing*, vol. 5, no. 1, pp. 81–91, 2011.

[72] Zaheduzzaman Sarker, Varun Singh, Xiaoqing Zhu et Michael A. Ramalho, “Test Cases for Evaluating RMCAT Proposals,” IETF, Internet Draft – work in progress 01, Juin 2014. [Online]. Available: http://datatracker.ietf.org/doc/draft-sarker-rmcat-eval-test/

[73] Ngamwongwattana, Boonchai and Thompson, Richard, “Measuring one-way delay of VoIP packets without clock synchronization,” in *Instrumentation and Measurement Technology Conference (I2MTC'09)*. IEEE, 2009, pp. 532–535.

[74] Johan Sjoberg, Magnus Westerlund, Ari Lakaniemi et Qiaobing Xie, “RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs,” RFC 4867, Avril 2007.

Annexe A

Les paramètres des bancs de test

Dans cette annexe, nous détaillons les paramètres des différents bancs de test.

A.1 Test 1 : Phase de démarrage

- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: Variable
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média : vidéo
    - Direction : lien montant
    - Nombre de sources : 1
    - Encodeur média : VP8
    - Contenu média : vidéo de tête parlante
    - Trafic concurrent : aucun

A.2 Test 2 : Test des modes de l'état PROBE

- Durée du test : 240 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: variable (Uplink)
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:

  - Type du média : vidéo
  - Direction : lien montant
  - Nombre de sources : 1
  - Encodeur média : VP8
  - Contenu média : vidéo de tête parlante
  - Chronologie des médias :
    - Flux N°: 1
    - Temps de départ : 0 s
    - Temps de fin : 240 s
  - Trafic concurrent : aucun

A.3 Test 3 : Capacité de lien variable

- Durée du test : 180 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: variable (Uplink)
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média :vidéo
    - Direction : lien montant
    - Nombre de sources : 3
    - Encodeur média : VP8
    - Contenu média : Vidéo de tête parlante
    - Chronologie des médias :
      - Flux N°: 1
      - Temps de départ : 0 s
      - Temps de fin : 180 s
    - Trafic concurrent : aucun

A.4 Test 4 : Équité entre plusieurs flux vidéo

- Durée du test : 240 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: 7 Mbps (Uplink)
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média : vidéo

  - Direction : lien montant
  - Nombre de sources : 3
  - Encodeur média : VP8
  - Contenu média : Vidéo de tête parlante
  - Chronologie des médias :
    - Flux N°: 1
    - Temps de départ : 0 s
    - Temps de fin : 240 s
    - Flux N°: 2
    - Temps de départ : 40 s
    - Temps de fin : 240 s
    - Flux N°: 3
    - Temps de départ : 80 s
    - Temps de fin : 240 s
  - Trafic concurrent : aucun

A.5 Test 5 : Concurrence entre un flux vidéo et plusieurs flux TCP de longue durée

- Durée du test : 240 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: 4 Mbps (Uplink)
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média : vidéo
    - Direction : lien montant
    - Nombre de sources : 1
    - Encodeur média : VP8
    - Contenu média : Vidéo de tête parlante
    - Trafic concurrent : 4 Flux TCP
    - Chronologie des flux TCP :
      - Flux N°: 1
      - Temps de départ : 40 s
      - Temps de fin : 170 s
      - Flux N°: 2
      - Temps de départ : 80 s
      - Temps de fin : 170 s
      - Flux N°: 3
      - Temps de départ : 80 s
      - Temps de fin : 170 s
      - Flux N°: 4
      - Temps de départ : 80 s

      - Temps de fin : 170 s

A.6 Test 6 : Concurrence entre un flux vidéo et plusieurs flux TCP de courte durée

- Durée du test : 240 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: 4 Mbps (Uplink)
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média : vidéo
    - Direction : lien montant
    - Nombre de sources : 1
    - Encodeur média : VP8
    - Contenu média : Vidéo de tête parlante
    - Chronologie des médias :
      - Flux N°: 1
      - Temps de départ : 0 s
      - Temps de fin : 240 s
    - Trafic concurrent : plusieurs flux TCP de courte durée

A.7 Test 7 : Lien de retour congestionné

- Durée du test : 180 s
- Les caractéristiques du lien :
  - Capacité du goulot d'étranglement: variable (Uplink), 2 Mbps
  - OWD : 50 ms
  - Type de la file d'attente du goulot d'étranglement : Drop Tail
  - Taille de la file : 75 KB
  - Taux de perte de paquets : 0
- Les caractéristiques du trafic :
  - Source média:
    - Type du média : vidéo
    - Direction : lien montant
    - Nombre de sources : 1
    - Encodeur média : VP8
    - Contenu média : Vidéo de tête parlante
    - Trafic concurrent : Flux TCP de longue durée sur le lien de feed-back

Annexe B

Le débit des profils utilisés

Notre algorithme utilise 35 profils en total. Le tableau B.1 contient le débit vidéo de chaque profil.

Tableau B.1 – Débit vidéo des profils

Profil	Débit vidéo (Kb/s)
P_{min}	50,000
P_2	73,516
P_3	91,648
P_4	110,671
P_5	140,894
P_6	155,035
P_7	175,901
P_8	222,589
P_9	324,495
P_{10}	459,228
P_{11}	563,646
P_{12}	660,738
P_{13}	712,000
P_{14}	839,300
P_{15}	927,750
P_{16}	1015,680
P_{17}	1157,421
P_{18}	1204,163
P_{19}	1309,453
P_{20}	1404,540
P_{21}	1530,186
P_{22}	1630,565
P_{23}	1790,262
P_{24}	1870,055
P_{25}	1934,095
P_{26}	2066,562
P_{27}	2131,568
P_{28}	2230,234
P_{29}	2305,621
P_{30}	2401,549
P_{31}	2516,061
P_{32}	2601,531
P_{33}	2704,240
P_{34}	2810,651
P_{max}	2954,942