

UNIVERSITÉ DE MONTRÉAL

AMÉLIORATION DE LA QUALITÉ DE SERVICE DES APPLICATIONS DE
VIDÉOCONFÉRENCE DANS LES RÉSEAUX AD HOC

SERGE EL HELOU
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
SEPTEMBRE 2008



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-46046-7

Our file Notre référence

ISBN: 978-0-494-46046-7

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

CE MÉMOIRE INTITULÉ :

AMÉLIORATION DE LA QUALITÉ DE SERVICE DES APPLICATIONS DE
VIDÉOCONFÉRENCE DANS LES RÉSEAUX AD HOC

présenté par : EL HELOU Serge

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

Mme. BOUCHENEB Hanifa, Doctorat, présidente

M. QUINTERO Alejandro, Doct., membre et directeur de recherche

M. KHENDEK Ferhat, Ph.D., membre et codirecteur de recherche

M. PIERRE Samuel, Ph.D., membre

*Si nous savions ce que nous faisons, cela ne
s'appellerait plus de la recherche, n'est ce pas?*

Einstein

*Celui qui n'a pas d'objectifs ne risque pas de
les atteindre.*

Sun Tzu

REMERCIEMENTS

Je voudrai d'abord remercier MM. Alejandro Quintero et Ferhat Khendek, mes directeur et co-directeur de recherche, pour leurs conseils, leur soutien et leur encouragements nécessaires à l'accomplissement de mon travail de recherche.

Je tiens aussi à remercier tous les étudiants membres du LARIM, qui ont partagé avec moi leurs connaissances, ainsi que leurs expériences enrichissantes.

Je voudrai finalement remercier la femme sans qui ce travail, et moi, n'aurions jamais vus le jour, ma mère, dont tous les sacrifices, la confiance, et le soutien financier et moral ont fait de moi ce que je suis. Et tous les membres de ma famille qui m'ont soutenu tant financièrement que moralement tout au long de mes études.

Merci à tous.

RÉSUMÉ

L'ubiquité est le mot d'or de notre ère. Les équipements de communication nomades sont en pleine effervescence, et les technologies de support de ce genre de communication en expansion rapide. Après les réseaux sans fil traditionnels, les réseaux ad hoc prennent le relais, et poursuivent la course vers des communications sans fil de bout en bout. Mais ces réseaux présentent encore un problème majeur, la qualité de service. Les réseaux ad hoc actuels ainsi que les protocoles de routage et d'accès au réseau utilisés sur ces réseaux, n'offrent aucune garantie de qualité de service. De ce fait toute la pression de la qualité de service repose sur l'application.

Notre objectif principal est de trouver une méthode pour améliorer la qualité des applications de vidéoconférence sur les réseaux ad hoc. L'approche que nous adoptons consiste à rendre l'application de vidéoconférence utilisée adaptative à l'état du réseau. Pour cela, nous allons définir une architecture *cross-layer* qui va permettre à l'application de collecter de l'information, en temps réel, sur l'état du réseau. De même, nous allons définir plusieurs modes de fonctionnement de l'application à la source. Ainsi, et en se basant sur l'information collectée du réseau, notre application va s'adapter dynamiquement à l'état actuel du réseau, en passant d'un mode de fonctionnement à l'autre, suivant un ordre prédéfini.

Les résultats obtenus suite à la simulation de la solution que nous proposons, sont très satisfaisants, et prouvent qu'en rendant l'application de vidéoconférence adaptative à l'état du réseau, nous parvenons à maintenir une qualité de la vidéo à la réception relativement stable. En comparant les résultats ainsi obtenus, aux résultats obtenus dans le cas d'applications non adaptatives, nous remarquons qu'en plus de la stabilité, notre solution améliore la qualité globale de la vidéo à la réception.

ABSTRACT

Ubiquity is a very popular concept nowadays. This decade has witnessed a booming of the nomad communication equipments and of the technologies supporting this kind of communications. So the traditional wireless networks make way for the new ad hoc networks, which brings us one step closer to the end-to-end wireless communications. But the ad hoc networks have a major limitation, quality of service, which is not yet supported in this kind of networks. In fact, the ad hoc networks, as well as the routing protocols and the medium access protocols used, offer no guaranty of service whatsoever. Thus, all the burden of quality of service lays on the application layer.

Our main objective is to find a way to improve videoconferencing applications' quality of service in ad hoc networks. Our approach consists of making the videoconferencing application adaptive to the network's state. In order for the application to collect the needed information on network, we define a cross-layer architecture. Furthermore, we define different functioning modes for the application at the source. Thus, based on the information collected via the cross-layer architecture, the application will dynamically adapt to the network's state, by passing from one functioning mode to the other in a predefined order.

The results obtained after the simulation of our solution are very satisfying, and prove that we can reach a relative stability of the quality of the received video by dynamically adapting the application at the source to the network's state. Furthermore, the simulations results show that besides stability, the overall quality of the video received is better when using our solution.

TABLE DES MATIÈRES

DÉDICACE	iv
REMERCIEMENTS.....	v
RÉSUMÉ.....	vi
ABSTRACT.....	vii
TABLE DES MATIÈRES	viii
LISTE DES TABLEAUX.....	xii
LISTE DES SIGLES ET ABRÉVIATIONS	xv
CHAPITRE 1 INTRODUCTION	1
1.1 Définition et historique	2
1.2 Eléments de la problématique	3
1.3 Objectifs de recherches et résultats attendus	4
1.4 Plan du mémoire	5
CHAPITRE 2 RÉSEAUX MOBILES AD HOC ET ADAPTABILITÉ DES APPLICATIONS	6
2.1 Limitations et problèmes des réseaux ad hoc	7
2.2 Protocoles de routage dans les réseaux ad hoc	8
2.2.1 Caractéristiques des protocoles de routage	8
2.2.2 Comparaison de différents protocoles de routage.....	9
2.2.3 Caractéristiques de AODV	10
2.3 Protocoles de transport.....	12
2.3.1 Real-time transport protocol (RTP)	13
2.3.2 Caractéristiques de RTP/RTCP.....	13
2.4 Standards de vidéoconférence.....	15

2.4.1	Comparaison des standards de vidéoconférence.....	16
2.4.2	Caractéristiques de H.264/AVC.....	17
2.5	Architecture Cross-Layer.....	19
2.6	Etudes existantes dans le domaine.....	20
2.6.1	Délais et gigue dans la vidéoconférence.....	20
2.6.2	Erreurs de transmission dans la vidéoconférence.....	22
2.6.3	Effet des coupures de liens sur la qualité.....	23
2.6.4	Application adaptative.....	24
CHAPITRE 3 MÉTHODE D'ADAPTATION DE L'APPLICATION.....		26
3.1	Hypothèses.....	26
3.2	États du réseau et mode de fonctionnement de l'application.....	27
3.2.1	Aperçu sur les modes de fonctionnement.....	27
3.2.2	Etats du réseau.....	28
3.2.2.1	Chemin établi.....	28
3.2.2.2	Chemin établi avec grande gigue.....	30
3.2.2.3	Chemin établi avec congestion.....	31
3.2.2.4	Coupure de lien intermédiaire.....	32
3.3	Caractéristiques des modes de fonctionnement.....	36
3.3.1	Paramètres niveau application.....	36
3.3.2	Paramètres niveau transport.....	38
3.3.3	Mode High.....	39
3.3.4	Mode Normal1.....	40
3.3.5	Mode Normal2.....	42
3.3.6	Mode Congestion Prevention.....	43
3.3.7	Mode Prevention.....	44
3.3.8	Mode Recovery.....	45
3.3.9	Mode Congestion Recovery.....	47
3.4	Architecture cross-layer.....	48
3.4.1	Architecture cross-layer choisie.....	49

3.4.2	Paramètres échangés entre les couches.....	49
3.5	Méthode de signalisation	52
CHAPITRE 4 IMPLÉMENTATION ET ÉVALUATION DES PERFORMANCES.		54
4.1	Environnement de simulation	54
4.1.1	Choix du simulateur.....	54
4.1.2	Générateur de trafic vidéo.....	55
4.2	Implémentation de la solution.....	57
4.2.1	Stratégie d'évaluation	57
4.2.2	Etapas de l'implémentation de la solution	58
4.2.2.1	Codage de la vidéo.....	58
4.2.2.2	Création de fichier MP4.....	60
4.2.2.3	Envoie de la vidéo à une destination.....	61
4.2.2.4	Simulation du réseau ad hoc	62
4.2.2.5	Reconstruction de la vidéo reçue	66
4.2.2.6	Décodage de la vidéo	67
4.2.2.7	Calcul du PSNR	67
4.3	Résultats de simulation	68
4.3.1	Débit et PSNR en sortie du codeur	68
4.3.2	PSNR et temps de décodage en sortie du décodeur	70
4.3.3	Résultats pour différents cas de coupure de liens	71
4.3.3.1	Faible taux de coupure de liens.....	71
4.3.3.2	Fort taux de coupure de liens	73
4.3.3.3	Fort taux de coupure de liens avec grande vitesse de réparation.....	75
4.3.3.4	Simulation de notre solution	77
4.3.4	Résultats pour différents cas de congestion	79
4.3.4.1	Mode <i>High</i>	79
4.3.4.2	Mode <i>Normal1</i>	81
4.3.4.3	Mode <i>Normal2</i>	82
4.3.4.4	Mode <i>Congestion Prevention</i>	83

4.3.4.5	Mode <i>Prevention</i>	85
4.3.4.6	Mode <i>Rocover</i>	86
4.3.4.7	Mode <i>Congestion Recovery</i>	87
4.3.4.8	Solution proposée.....	89
CHAPITRE 5 CONCLUSIONS.....		92
5.1	Synthèse du travail et contribution	92
5.2	Limitations de notre travail.....	93
5.3	Travaux futurs.....	94
RÉFÉRENCES		95

LISTE DES TABLEAUX

Tableau 4.1 Résultats de simulation dans un réseau à faible taux de coupure de liens	73
Tableau 4.2 Résultats de simulation dans un réseau à fort taux de coupure de liens	75
Tableau 4.1 Résultats de simulation dans un réseau à fort taux de coupure de liens avec une grande vitesse de réparation	77
Tableau 4.4 Résultats de simulation de la solution proposée	79
Tableau 4.5 Résultats de simulation du mode <i>High</i> dans un réseau à différents états de congestion	80
Tableau 4.6 Résultats de simulation du mode <i>Normal1</i> dans un réseau à différents états de congestion	82
Tableau 4.7 Résultats de simulation du mode <i>Normal2</i> dans un réseau à différents états de congestion	83
Tableau 4.8 Résultats de simulation du mode <i>CP</i> dans un réseau à différents états de congestion	84
Tableau 4.9 Résultats de simulation du mode <i>Prevention</i> dans un réseau à différents états de congestion	86
Tableau 4.10 Résultats de simulation du mode <i>Recovery</i> dans un réseau à différents états de congestion	87
Tableau 4.11 Résultats de simulation du mode <i>CR</i> dans un réseau à différents états de congestion	89
Tableau 4.12 Résultats de simulation de notre solution dans un réseau à différents états de congestion	90

LISTE DES FIGURES

Figure 3.1 Chemin établi.....	30
Figure 3.2 Chemin établi avec grande gigue	31
Figure 3.3 Chemin établi avec congestion.....	32
Figure 3.4 Coupure de lien intermédiaire	33
Figure 3.5 Lien réparé localement	33
Figure 3.6 Lien réparé de la source.....	34
Figure 3.7 Lien réparé de la source et localement	35
Figure 3.8 Architecture cross-layer choisie	51
Figure 3.9 Fonctionnement de l'application	53
Figure 4.1 Débit et moyenne des PSNRs en sortie du codeur pour les différents modes de fonctionnement	69
Figure 4.2 Temps de décodage et moyenne des PSNRs en sortie du décodeur pour les différents modes de fonctionnement.....	70
Figure 4.3 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à faible taux de coupure de liens.....	72
Figure 4.4 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à grand taux de coupure de liens.....	74
Figure 4.5 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à grand taux de coupure de liens et grande vitesse de réparation	76

Figure 4.6 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau passant par différents états	78
Figure 4.7 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>High</i> dans un réseau dans différents états de congestion	80
Figure 4.8 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>Normal1</i> dans un réseau dans différents états de congestion	81
Figure 4.9 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>Normal2</i> dans un réseau dans différents états de congestion	82
Figure 4.10 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>CP</i> dans un réseau dans différents états de congestion	84
Figure 4.11 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>Prevention</i> dans un réseau dans différents états de congestion	85
Figure 4.12 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>Recovery</i> dans un réseau dans différents états de congestion	86
Figure 4.13 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour le mode <i>CR</i> dans un réseau dans différents états de congestion	88
Figure 4.14 Trames reçues, PSNR et <i>Quality</i> en sortie du décodeur pour la solution proposée dans un réseau dans différents états de congestion	89

LISTE DES SIGLES ET ABRÉVIATIONS

ASO	Arbitrary Slice Ordering
AVC	Advanced Video Coding
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable-Length Coding
CDMA	Code Division Multiple Access
CP	Congestion Prevention
CR	Congestion Recovery
CSMA	Carrier Sense Medium Access
CSMA/CA	CSMA with Collision Avoidance
DARPA	Defense Advanced Research Projects Agency
DON	Decoding Order Number
DONB	Decoding Order Number Base
DOND	Decoding Order Number Difference
FEC	Forward Error Correction
FMO	Flexible Macrobloc Ordering
FU	Fragmentation Unit
GOP	Group Of Picture
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union - Telecommunication
JVT	Joint Video Team
MANET	Mobile Ad hoc NETWORK
MB	MacroBlock
MPEG	Moving Picture Experts Group
MTAP	Multi-Time Aggregation Packet
MTU	Maximum Transmit Unit

PSNR	Peak Signal-Noise Ratio
PDA	Personal Digital Assistant
PRNET	Packet Radio NETwork
QoS	Quality of Service
QP	Quantization Parameter
RERR	Route ERRor
RR	Receiver Report
RREP	Route REPlY
RREQ	Route REQuest
RTCP	RTp Control Protocol
RTP	Real-time Transport Protocol
STAP	Single-Time Aggregation Packet
SURAN	SURvivable Radio Networks
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TTL	Time To Live
UDP	User Datagram Protocol
VCEG	Video Coding Experts Group
VCL	Video Coding Layer

CHAPITRE 1 INTRODUCTION

Dans tous les domaines de la science, l'évolution nous mène vers des technologies de plus en plus adaptées au mode de vie actuel, au prix d'une complexité toujours croissante. En informatique, et plus spécialement en réseautique, l'évolution nous a menés à adapter nos réseaux à l'ubiquité des utilisateurs. En effets les utilisateurs des réseaux informatiques sont passés de personnes fixes assis à leurs bureau et travaillants sur leur ordinateur de bureau, à des personnes en déplacement continu, travaillants sur des équipements mobiles, tel que les ordinateurs portables, les téléphones cellulaires ou les PDA (Personal Digital assistant). Ainsi pour suivre cette évolution, les réseaux informatiques sont aussi passés des réseaux fixes et complètement câblés de bout en bout, aux réseaux mobiles soutenus par des infrastructures fixes, les points d'accès. Ces réseaux mobiles continuent leur évolution pour donner naissance à une nouvelle génération de réseaux, les réseaux ad hoc. Ces réseaux sont sans fil de bout en bout. Ca veut dire que la communication entre deux nœuds n'utilise aucune infrastructure fixe. De meme, tous les nœuds à travers lesquels sont établies les connexions sont des nœuds mobiles. Ces avantages des réseaux ad hoc, à savoir la simplicité de création du réseau, ainsi que la mobilité des nœuds, entraînent du même coup leurs inconvénients. En effet, la mobilité des nœuds et l'absence d'infrastructure fixe, rends ces réseaux difficiles à gérer, et limite ainsi la qualité de service que ces réseaux présentent.

Ce travail est une étude sur les méthodes d'amélioration de la qualité de service des applications de vidéoconférence dans les réseaux ad hoc. Dans ce premier chapitre nous allons introduire brièvement les concepts des réseaux ad hoc, puis nous présenterons la problématique traitée dans ce document, pour ensuite survoler les éléments de la solution et les résultats prévus, pour finir avec un plan de ce document.

1.1 Définition et historique

Ad hoc en latin signifie « pour cette raison » ou « à cette fin ». Cette terminaison est donc utilisée pour définir quelque chose qui est fait au moment même, et sans planification à l'avance, pour résoudre un problème spécifique. Un réseau ad hoc est donc un réseau improvisé, et de ce fait il n'est pas supporté par une infrastructure fixe. Ce réseau est mis en place pour répondre à un besoin de communication entre deux entités, dans des cas où un réseau fixe n'est pas disponible.

MANET (Mobile Ad hoc NETwork), est un réseau ad hoc dont les nœuds sont mobiles, et communiquent entre eux en sans fil. La configuration d'un MANET change dynamiquement et aléatoirement. De plus la structure de ce réseau est plane, tous les nœuds sont au même niveau, chacun fonctionnant comme nœud terminal et nœud de routage en même temps. En effet, dans un réseau fixe, ou un réseau mobile avec infrastructure fixe, pour que les paquets passent de la source à la destination, ils doivent passer par des équipements fixes formant l'infrastructure du réseau, tel que les concentrateurs, les routeurs et les points d'accès. Ce qui n'est pas le cas dans les réseaux ad hoc où ce sont les nœuds mobiles qui jouent ce rôle d'équipements d'interconnexion, en même temps que celui de nœud transmetteur.

Les premiers réseaux sans fils furent créés en 1973 par la *DARPA (Defense Advanced Research Projects Agency)*. Les PRNET (Packet Radio NETWORKs) se basaient sur CSMA (Carrier Sense Medium Access) et ALOHA pour l'accès au medium. Ces premiers réseaux furent testés pour des environnements de combats. Les réseaux PRNETs ont évolués en réseaux SURANs (Survivable Radio Networks) en 1980, pour rendre les équipements plus petits et moins gourmands en énergie, et les réseaux plus robustes. Les protocoles utilisés dans ces réseaux étaient capables de supporter jusqu'à plusieurs milliers de nœuds. Finalement avec l'arrivée des ordinateurs portables en 1990, les MANETs sont devenus de plus en plus utilisés dans divers domaines.

1.2 Éléments de la problématique

Les réseaux informatiques, ainsi que l'internet ont embarqués dans la locomotive de l'évolution. Une locomotive en accélération continue, et à vrai dire incontrôlable. En effet, les utilisateurs des réseaux actuels, que ce soit les développeurs d'applications, ou les utilisateurs de ces application, demandent continuellement plus de ressources. Du coté des développeurs, les applications développées actuellement sont très gourmandes en ressource, que ce soit en bande passante, en énergie ou en puissance de traitement dans les nœuds. De même, les utilisateurs de leurs coté, sont toujours plus demandant en qualité de service. Dans les réseaux fixes, les technologies développées, ont permis de trouver un certain équilibre, entre les exigences des développeurs et des utilisateurs d'un coté, et les ressources disponibles dans les réseaux de l'autre.

Dans les réseaux sans fil, et spécialement dans les réseaux ad hoc, les ressources sont très limitées. En effet, les réseaux ad hoc sont limités en bande passante, car ils utilisent l'air comme medium de transmission. De plus, étant mobiles, les nœuds dans un réseau ad hoc n'ont pas accès à une source de courant, ce qui limite l'énergie dans les nœuds. De même, et du fait de leur mobilité, les nœuds sont souvent petits et ainsi leur puissance de traitement limitée. Toutes ces limitations physiques, limitent aussi la qualité de service que ces réseaux peuvent offrir, qualité de service qui est le critère le plus important pour les utilisateurs des applications. De plus, ces limitations propres aux réseaux ad hoc, rendent les technologies développées pour les réseaux fixes, inutilisables dans ce genre de réseaux. D'où le besoin de développer de nouvelles technologies spéciales pour les réseaux ad hoc. C'est pourquoi on témoigne de nos jours d'une avalanche de littérature traitant les problèmes de ces réseaux.

D'un autre cote, avec l'évolution d'internet, qui est devenu une nécessité pour tous, on a eut une vague d'applications développés dans différents domaines, utilisables sur internet. Mais le domaine qui a connu une expansion extraordinaire est celui des multimédias. Ces derniers consomment aujourd'hui la majorité des ressources d'internet, et la consommation n'est que croissante. En effet, téléchargement de music, de vidéo, le *streaming* et les vidéoconférences sont devenus des activités régulières dans

la vie de chacun de nous. La vidéoconférence est particulièrement intéressante dans les réseaux ad hoc. En ajoutant la mobilité et la simplicité de ces réseaux, à l'utilité de la communication audio et vidéo en temps réel qu'offre la vidéoconférence, on obtient une application primordiale dans différents domaines. En effet, cette application est d'une utilité sans pareil dans le domaine militaire, ainsi que dans les cas de désastres naturels ou non, qui rendent toute infrastructure fixe non opérationnelle. Malheureusement, l'application de vidéoconférence est très gourmande en ressources. De même la qualité de service est très importante pour les utilisateurs de cette application. Actuellement les réseaux ad hoc ne sont pas capables d'offrir les ressources nécessaires pour avoir une bonne qualité de la voix et la vidéo. Pour cette raison il est important de développer des applications qui peuvent s'adapter aux ressources offerts par les réseaux, et non inversement, pour offrir une certaine qualité de service aux utilisateurs. L'application doit donc être informée à tout moment de l'état du réseau. Cela peut être atteint à l'aide d'une architecture *cross-layer*, qui est de plus en plus utilisée dans les réseaux ad hoc.

1.3 Objectifs de recherches et résultats attendus

Dans ce mémoire nous proposons une méthode qui vise à améliorer la qualité de service des applications de vidéoconférence dans les réseaux ad hoc. Ceci en rendant l'application adaptative à l'état du réseau, en collectant l'information nécessaire du réseau grâce à une architecture *cross-layer*. Les étapes suivies pour atteindre se but sont :

- choix des protocoles et standards les plus adaptés à notre application :
 - protocole de routage offrant les meilleures performances dans les réseaux ad hoc. Ces dernières seront mesurées en temps d'établissement du chemin, et temps de réparation en cas de coupure de lien ;
 - Protocole de transport permettant une certaine forme de qualité de service, et permettant d'avoir des informations sur l'état du réseau ;
 - Un standard de vidéoconférence offrant plusieurs niveau de qualité de l'audio et de la vidéo, ainsi que des outils de correction d'erreurs.

- recherche des solutions proposées dans la littérature, analyse de ces solutions pour retenir les éléments les plus intéressants, et définir les limitations des études faites ;
- conception et implémentation d'une méthode d'adaptation dynamique de l'application de vidéoconférence à l'état du réseau :
 - étude des besoins des réseaux dans ses différents états possibles ;
 - définition du mode de fonctionnement dans chaque état du réseau, et de la méthode de passage d'un mode à l'autre ;
 - identification des paramètres à changer au niveau de la source pour avoir les différents modes de fonctionnements.
- évaluation de la méthode proposée, et analyse des résultats obtenus, en les comparants aux résultats obtenus dans le cas d'application non adaptative.

1.4 Plan du mémoire

Ce mémoire est formé de cinq chapitres. Le premier introduit les réseaux ad hoc, dans ce chapitre on trouve aussi les éléments de la problématique, ainsi que la méthodologie de travail suivie pour atteindre les résultats souhaités. Dans le deuxième on trouvera une explication plus détaillée des concepts et des technologies utilisées dans les réseaux ad hoc, suivie d'une étude sur les solutions proposées dans le domaine. Le troisième chapitre expose les détails de la solution proposée, en développant les différentes étapes du développement de la solution. Dans le chapitre suivant les performances de la solution proposée seront évaluées. Le dernier chapitre va conclure ce mémoire en analysant les résultats obtenus, et les limitations et faiblesses de la solution proposée, et offrant des voies future pour la recherche dans ce domaine.

CHAPITRE 2 RÉSEAUX MOBILES AD HOC ET ADAPTABILITÉ DES APPLICATIONS

Contrairement aux réseaux câblés, les réseaux sans fil sont des réseaux qui utilisent l'air comme medium de communication. Ces réseaux peuvent avoir une infrastructure fixe (i.e. les stations de base), pour supporter les nœuds mobiles, et présenter une certaine forme de centralisation dans le réseau. Les réseaux les plus utilisés appartenant à cette famille sont les réseaux cellulaires, où les téléphones portables sont les nœuds mobiles, et les stations de base sont les points d'accès à l'infrastructure fixe du réseau. Une autre famille des réseaux sans fils, c'est les réseaux mobiles ad hoc (MANETs, Mobile Ad hoc NETWORKs). Ces réseaux sont la nouvelle génération de réseaux sans fils. Ils ne présentent aucune forme d'infrastructure fixe, ou de centralisation. Tous les nœuds dans ces réseaux peuvent se déplacer aléatoirement dans l'espace, et offrent des fonctionnalités d'interconnexion entre eux. Donc si deux nœuds distants désirent communiquer, leurs messages doivent passer par des nœuds mobiles intermédiaires. Mais du fait de cette mobilité, les liens sont toujours sujets à des coupures imprévues et fréquentes. C'est pourquoi ces réseaux ne peuvent garantir aucune qualité de service. Ce manque en qualité de service, est la raison majeur qui retarde l'explosion attendu de ces réseaux dans le domaine des télécommunications.

Dans ce chapitre nous présenterons les caractéristiques des réseaux mobiles ad hoc, en faisant le point sur leurs limitations. Puis nous continueront en présentant les protocoles de routages utilisés dans ces réseaux, avant de passer à la couche supérieure en étudiant les protocoles de transports les plus adaptés à ce genre de réseau. Finalement nous exposerons un état de l'art des solutions présentes dans la littérature pour améliorer la qualité de service des applications de vidéoconférence sur les réseaux ad hoc.

2.1 Limitations et problèmes des réseaux ad hoc

Etant en sans fil, les MANETs utilisent l'air comme support de communication, avec les technologies développées par l'IETF tel que 802.11a/b/g. Par sa nature, l'air est un medium très peu fiable. La raison est que l'air est saturé de signaux électromagnétiques qui proviennent de différents équipements : domestiques, de communication ou industriels. Ces signaux électromagnétiques interfèrent entre eux et altèrent ainsi le message envoyé. De plus le signal voyageant dans l'air est sujet au phénomène d'évanouissement causé par des déplacements aléatoires du nœud mobile ainsi que des objets entourant ce nœud. Ces évanouissements imprévisibles causent aussi des erreurs à la réception. La première limitation des MANETs est donc le taux élevé d'erreur de canal.

Il existe aussi des limitations de routage dans les MANETs. Les nœuds dans ces réseaux se déplacent aléatoirement, et leur emplacement géographique ainsi que leurs nœuds de voisinage changent aléatoirement dans le temps. Pour que le routage soit donc efficace, ces nœuds doivent envoyer continuellement des messages de mise à jour de leur emplacement dans le réseau. Ces messages utilisent une partie de la bande passante qui est une ressource très limitée dans les réseaux ad hoc. Cette limitation de la bande passante est aussi un handicap dans les réseaux ad hoc, car les techniques de modulation utilisées dans les réseaux sans fil, telles que CDMA (Code Division Multiple Access) et TDMA (Time Division Multiple Access), se basent sur un équipement central qui assigne les différents codes (CDMA) et/ou *time slot* (TDMA). Or les MANETs sont des réseaux décentralisés, où, comme nous l'avons déjà mentionné, tous les nœuds sont au même niveau. La technique d'accès au medium la plus utilisée jusqu'à présent est donc le CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), qui est basé sur une compétition entre les nœuds, pour gagner l'accès en premier au medium.

La dernière limitation des MANETs concerne les nœuds mobiles. Ces derniers sont limités en énergie, espace mémoire et puissance de traitement. En effet, du fait du déplacement des nœuds, une source d'énergie « inépuisable » n'est pas disponible. Donc le nœud doit se contenter de l'énergie contenue dans sa pile. De plus, la taille des

nœuds mobiles est généralement petite ce qui limite d'abord la taille de la mémoire contenue dans ces nœuds, puis la puissance des processeurs qui ne sont pas aussi bien refroidis que dans les équipement fixes de taille plus grande.

Toutes les limitations exposées là-dessus, introduisent un problème majeur aux réseaux ad hoc, le problème de qualité de service. En effet pour que le réseau soit capable d'offrir une certaine qualité de service, il doit pouvoir garantir la bande passante nécessaire à l'application. De même il doit pouvoir offrir un minimum de stabilité pour les liens pour garder une certaine forme de continuité dans la communication. De plus au niveau des nœuds ils doivent être assez puissants, d'abord en énergie, pour transmettre avec le minimum d'erreurs le plus loin possible, et en traitement, pour pouvoir implémenter des méthodes de correction d'erreurs.

2.2 Protocoles de routage dans les réseaux ad hoc

Un protocole de routage est un protocole qui fonctionne à la couche 3 du model OSI. Ce protocole définit la méthode avec laquelle les nœuds d'un réseau communiquent entre eux. Ainsi, un nœud source désirant envoyer de l'information à un ou plusieurs nœuds destinations, commence par lancer l'opération de découverte de chemin. Cette opération permet au nœud source de trouver le meilleur chemin que ces paquets doivent prendre pour atteindre la destination. Comme dans n'importe quel déplacement, la découverte de chemin entre la source et la destination se base sur une adresse, IP dans le cas des réseaux, pour savoir dans quelle direction chercher.

2.2.1 Caractéristiques des protocoles de routage

Les protocoles de routage sont divisés en plusieurs familles. Les protocoles proactifs, où l'information sur le chemin à suivre pour atteindre une destination existe déjà dans les nœuds; les protocoles réactifs, où l'information sur le chemin à suivre est collectée sur demande dans chaque nœud, et les protocoles hybrides. Cette dernière famille combine les deux modes de fonctionnement des deux familles précédentes.

2.2.2 Comparaison de différents protocoles de routage

Pour qu'un protocole de routage soit bien adapté aux réseaux ad hoc, il doit prendre en considération les limitations que présente ce type de réseaux (voir partie 2.1). A savoir, étant donné que dans un réseau ad hoc les nœuds sont mobiles, les liens entre différents nœuds ne sont pas stables, et des coupures de liens peuvent survenir en tout moment. Donc le protocole de routage doit être capable de détecter rapidement cette coupure, et d'y remédier aussi rapidement, en trouvant un nouveau chemin pour la suite de la communication. De plus, comme déjà mentionné, la bande passante dans les réseaux ad hoc est très limitée, donc le protocole de routage ne doit pas ajouter beaucoup de trafic inutile sur le réseau, et la découverte de chemin doit être faite avec le minimum possible de signalisation. Finalement, vu la faible puissance de traitement des équipements mobiles, il est important que le protocole de routage soit léger, et ne demande pas beaucoup de traitement dans les nœuds. Ceci est très important dans les réseaux ad hoc, puisqu'en plus de la faible puissance de traitement dans les équipements, aucun équipement de centralisation n'existe dans ce type de réseau pour centraliser le traitement de l'information.

Plusieurs protocoles de routages pour les réseaux ad hoc sont définis dans la littérature. Mais répondre en même temps à toutes les exigences mentionnées plus haut est quasiment impossible, pour cela, les développeurs de ces protocoles se trouvent contraints à faire des compromis, en favorisant un paramètre par rapport à l'autre.

Une comparaison entre les plus importants protocoles de routage dans les réseaux ad hoc a été effectuée dans [5].

Dans ce document [5], les auteurs analysent les performances de H.264 [6] sur MANETs basés sur 802.11b. Les performances de H.264 sont évaluées en utilisant différents protocoles de routage pour enfin trouver le protocole le plus adapté à ce standard.

Les protocoles de routage utilisés sont OLSR [1], AODV [2], DSR [3] et TORA [4], AODV-H, et les tests se font dans le cas de faible et de forte mobilité.

Le premier test est fait dans un environnement normal de mobilité. Dans ce test, les auteurs évaluent le temps de réparation de lien des différents protocoles de routage. Les résultats obtenus [5], montrent que AODV répond le plus rapidement à une coupure de lien, et donc a le plus court temps de réparation.

Le deuxième test vise à étudier l'impact des différents protocoles de routage sur la qualité de la vidéo (PSNR) et le taux de perte de paquets. Les résultats obtenus [5], montrent une légère supériorité de AODV en termes de PSNR et taux de perte, mais à forte mobilité AODV devient moins performant que TORA, qui lui ne présente pas une grande différence entre faible et forte mobilité.

D'autres tests ont été effectués sur TORA et AODV, dans le même environnement que précédemment, mais avec plusieurs connexions TCP dans le premier cas, et plusieurs connexions vidéo dans le deuxième. Les résultats de ces tests montrent que TORA perd de sa performance plus rapidement que AODV, et ceci quand le nombre de connexions, que ce soit TCP ou vidéo, augmente.

Donc les auteurs dans [5] arrivent à la conclusion que AODV est le protocole de routage le plus adapté à la transmission d'une vidéo codée avec le standard H.264, sur un réseau ad hoc.

2.2.3 Caractéristiques de AODV

AODV [7] est un protocole de routage fait spécialement pour les réseaux ad hoc. Ce protocole appartient à la famille des protocoles réactifs (voir partie 2.2.1), donc il ne conserve pas dans la mémoire du nœud l'information sur le chemin à prendre pour atteindre une destination. Au besoin, ce protocole lance le processus de découverte de chemin entre la source et la destination. Les principaux avantages de AODV sont sa vitesse de convergence en cas de changement dans la topologie du réseau, et le concept de numéro de séquence de la destination. Ce dernier permet, d'un côté de s'assurer que le chemin pris est sans boucle, de l'autre, au niveau de la source de faire le choix de chemin, dans le cas où plusieurs chemins sont disponibles.

AODV se base sur 3 types de message pour gérer une connexion [7], *Route Request* RREQ, *Route Reply* RREP et *Route Error* RERR.

Route Request : quand un nœud source a besoin de communiquer avec un nœud destination, il a recours aux services de AODV pour la découverte de chemin. Ce dernier génère un message de requête de chemin (RREQ) [7], qu'il envoie en diffusion dans tout le réseau, avec une limite de diffusion spécifiée dans le champ *Time To Live (TTL)*. Le RREQ contient deux champs importants qui sont, *Originator Sequence Number* (numero de sequence de la source), c'est l'identifiant de la source, à être utilisé dans la table de routage des nœuds intermédiaires comme source de ce chemin. Ainsi que le *Destination Sequence Number* (numéro de séquence de la destination), c'est le dernier numéro de séquence reçu par la source, et identifiant une route vers le nœud destination. Cet identifiant permet aux nœuds intermédiaires de vérifier la validité des routes qu'ils ont en mémoire, et ainsi en cas de numéro de séquence supérieur ou égale à celui contenu dans le RREQ, le nœud intermédiaire peut répondre directement à la source, par un RREP avec le chemin à prendre vers le nœud destination.

Route Reply : quand un RREQ atteint la destination ou un nœud intermédiaire connaissant un chemin valide vers la destination, ce nœud répond à la requête par un RREP [7], contenant le numéro de séquence de la destination. Au cas où le nœud répondant au RREQ est un nœud intermédiaire, ce dernier met le numéro de séquence et la valeur de *lifetime* (durée de validité du lien) contenus dans sa table de routage, alors que si le nœud répondant est le nœud destination, ce dernier incrémente son numéro de séquence avant de l'ajouter à RREP et ajoute la valeur *MY_ROUTE_TIMEOUT* [7] dans le champ *lifetime*. Enfin le RREP est envoyé en unicast à la source en utilisant le chemin contenu dans la table de routage du nœud générant ce RREP. Au passage dans chaque nœud, le champ *Hop Count* (nombre de sauts) est incrémente de 1 de sorte que la source sache la distance qui la sépare de la destination.

Route Error : quand une destination devient inaccessible, à cause d'une coupure dans un lien ou de nœud désactivé, le nœud qui détecte cette coupure, envoie un message à tous les voisins utilisant ce lien pour atteindre une destination. Ce message contient la

liste de destinations rendue inaccessible suite à cette coupure de lien, dans le champ *Unreachable Destination Sequence Number* [7]. Ainsi que le champ *N* mis à 1 si le nœud a effectué une réparation locale du chemin. L'état des liens est connu dans les chemins actifs à l'aide des messages *Hello*.

Hello : un message *Hello* est un paquet RREP avec TTL mis à 1. Ce message est envoyé par un nœud faisant parti d'un chemin actif, pour informer ses voisins que ce lien est encore disponible. Pour minimiser la signalisation, les messages *Hello* sont envoyés uniquement par les nœuds faisant parti d'un chemin actif, et seulement si aucun paquet n'a été envoyé sur ce lien pendant une période *HELLO_INTERVAL* (intervalle de temps entre deux messages *Hello* consécutifs). Ainsi, si le voisin ne reçoit aucun paquet ou message *Hello* pendant $ALLOWED_HELLO_LOSS * HELLO_INTERVAL$ secondes (durée de temps pendant laquelle un message *Hello* peut ne pas arriver au nœud, à cause d'une perte de paquet), ça veut dire qu'il y a coupure de lien et il génère un RERR.

2.3 Protocoles de transport

Un protocole de transport définit une méthode de transfert de données entre deux nœuds. Il permet le « transport » de l'information d'un nœud à l'autre, d'une façon transparente pour l'application qui désire effectuer ce transfert. Dans le modèle de communication, que ce soit OSI ou TCP/IP, mis en place pour permettre une séparation entre les différents niveaux d'une communication, la couche de transport est probablement la plus importante. Cette couche permet une séparation entre ce qu'on appelle les couches basses, à savoir les couches réseau, liaison, et physique, dans le modèle OSI, ou leurs équivalents dans les modèles TCP/IP, et les couches hautes, à savoir les couches présentation, session et application, dans le modèle OSI, ou leurs équivalents dans les modèles TCP/IP. De même, elle a une importante tâche qui consiste à gérer la livraison des données aux couches hautes.

Contrairement à TCP (Transmission Control Protocol), UDP (User Datagram Protocol) n'offre aucune garantie de service, ou mécanisme de détection ou correction d'erreurs.

Toute cette légèreté et simplicité, rend ce protocole bien adapté aux applications où le délai est plus important que le taux de perte, comme les applications temps réel (vidéoconférence par exemple). Pour cette raison il est d'habitude utilisé avec un protocole de transport plus fiable pour les applications audio et vidéo, RTP (Real-time Transport Protocol) [8].

2.3.1 Real-time transport protocol (RTP)

Real-time Transport Protocol est défini dans la couche application du modèle TCP/IP, et dans la couche transport du modèle OSI. C'est pourquoi il est intéressant de l'exposer dans la partie concernant les protocoles de transport.

RTP est un standard qui définit le format des paquets utilisés pour le transport d'un trafic temps réel. Comme UDP ne présente aucune garantie de service, donc aucune QoS (Quality of Service), RTP vient s'ajouter à UDP pour présenter une certaine QoS pour les applications temps réel. Même si RTP ne garantit pas directement une QoS, mais le paquet envoyé contient assez d'informations pour que la destination puisse traiter correctement les données reçues.

Pour avoir plus de contrôle sur la communication, un protocole de contrôle RTCP (RTP Control Protocol) [8] est utilisé avec RTP. Ce protocole permet à la destination d'envoyer à la source un rapport sur la qualité de la communication et des paquets reçus. A la réception de ce rapport, la source peut décider de modifier ces paramètres (diminuer le débit par exemple) pour améliorer la qualité de la communication.

2.3.2 Caractéristiques de RTP/RTCP

A chaque sortie d'un nouveau standard de compression audio et/ou vidéo, un nouveau format de paquet RTP est défini spécialement, pour le transport de la vidéo et/ou l'audio suivant ce standard. Dans [8] on définit le format du paquet RTP pour le transport de vidéo suivant le standard H.264 [9]. Comme mentionné précédemment, le paquet RTP contient l'information nécessaire, pour qu'à la destination, l'application puisse utiliser de façon efficace les données. Parmi ces champs, les plus intéressants sont :

- *Payload Type (PT)* : champs qui contient le type du contenu du paquet. Donc sous quel format les données sont présentées dans le champ de données du paquet.
- *Sequence number* : c'est le numéro de séquence du paquet reçu. Comme dans UDP aucun mécanisme de remise en ordre n'existe, RTP vient combler ce manque avec son champ «sequence number » qui permet à la source de remettre en ordre les paquets reçus.
- *TimeStamp* : ce champ contient le temps auquel la source a fait l'échantillonnage du contenu de ce paquet. Tout comme le champ « Sequence number », ce champ permet à la source de s'assurer de la position de ce paquet, ou de cet échantillon reçu, parmi les autres reçus avant ou après.

De même, dans [8] on définit trois types de paquets, suivant le format des données qu'il contient :

- Paquet à une seule unité NAL : c'est un paquet RTP qui ne contient qu'une seule unité NAL [9].
- Paquet d'agrégats : c'est un paquet RTP qui contient plusieurs unités NAL et qui se présente sous quatre formes différentes :
 - STAP-A : Paquet d'agrégat à temps unique sans champs DON (Decoding Order Number).
 - STAP-B : Paquet d'agrégat à temps unique avec champs DON.
 - MTAP16 : Paquet d'agrégat à temps multiples avec un champ *offset* de 16 bits.
 - MTAP24 : Paquet d'agrégat à temps multiples avec un champ *offset* de 24 bits.
- Unité de Fragmentation (FU) : paquet contenant des fragments des unités NAL, ce paquet se présente sous deux formes :
 - FU-A : Paquet de fragmentation sans champ DON.
 - FU-B : Paquet de fragmentation avec un champ DON.

Pour les paquets de types STAP-A, STAP-B, MTAP16, MTAP24, FU-A et FU-B ce document spécifie six nouveaux types d'unité NAL qui étaient non définis dans la spécification du JVT H.264 [9], ces types prennent les valeurs 24, 25, 26, 27, 28 et 29 respectivement.

De plus, ce document définit trois types de paquetage :

- Mode à unité NAL unique : mode où les unités NAL sont envoyées l'une à la suite de l'autre dans différents paquets RTP et suivant le bon ordre de décodage.
- Mode Non mêlé : mode où les unités NAL peuvent être envoyées par agrégats ou fragments mais doivent garder le bon ordre de décodage.
- Mode mêlé : mode où les unités NAL peuvent être envoyées par agrégats ou fragments et ne suivent pas nécessairement le bon ordre de décodage. Ce mode utilise le champ DON pour remettre les unités NAL dans le bon ordre au décodeur. Pour les paquets de type MTAP deux champs différents sont ajoutées, *Decoding Order Number Base (DONB)* qui a la même valeur que DON, et *Decoding Order Number Difference (DOND)* qui est la différence entre DONB et le DON d'une unité NAL, si cette dernière était envoyée dans un paquet RTP toute seule.

Pour notre recherche nous allons retenir les paquets à unité NAL unique, les paquets STP-A et les paquets FU-A. les autres paquets avec un champ DON ne nous intéressent pas puisque le profile de H.264 que nous utilisons ne permet pas l'utilisation de paquets mêlés. De plus le changement de mode de paquetage se fera dynamiquement suivant l'état du réseau, information qui sera recueillie grâce au protocole de routage AODV, ainsi que grâce au rapport du récepteur (RR) du protocole RTCP.

2.4 Standards de vidéoconférence

Une vidéoconférence c'est une application qui permet à deux ou plusieurs personnes de communiquer entre eux. Cette communication est en audio et vidéo et se passe en temps réel. Avant de transmettre la voix et la vidéo sur le réseau, on procède à une

compression des deux. Les organismes de standardisation travaillant dans le domaine, définissent la façon avec laquelle la voix et la vidéo sont compressées, et présentées de l'application vers le réseau et inversement. Les deux organismes les plus actifs dans le domaine sont ISO/IEC avec son groupe de travail MPEG (Moving Picture Experts Group). Cet organisme a déjà défini plusieurs standards, les plus importants sont MPEG-1 [10], MPEG-2 [11] et MPEG-4 [12]. Le deuxième organisme est ITU-T avec son groupe de travail VCEG (Video Coding Experts Group). Les deux standards les plus importants définis par cet organisme sont H.261 [13], H.263 [14]. En 2003 les deux organismes mentionnés plus haut joignent leurs efforts et expertises dans le JVT (Joint Video Team), pour créer un standard qui unit les avantages des standards créés par les deux organismes. Le résultat, le standard H.264 /AVC (H.264/Advanced Video Coding) [9].

2.4.1 Comparaison des standards de vidéoconférence

La comparaison de différents standards de vidéoconférence a été effectuée dans [15] où on compare les performances du standard H.264 par rapport aux standards MPEG2, MPEG4 et H.263. Les auteurs effectuent une comparaison pour des séquences vidéo à faible débit, d'où l'utilisation du format d'image QCIF, avec des séquences de 150 trames. De plus, comme l'œil humain est plus sensible à la luminance qu'à la chrominance, les auteurs ne considèrent que la composante de luminance dans l'évaluation. La comparaison se fait par rapport au débit généré en sortie du codeur, et le PSNR (Peak Signal Noise Ratio) de la séquence codée.

Dans le document [15] on trouve les résultats obtenus pour les quatre types de codage avec différentes séquences vidéo à différents débits. Ces résultats montrent une visible supériorité dans les performances de H.264 par rapport aux autres standards simulés.

Notre recherche consiste à rendre les applications de vidéoconférence adaptatives aux réseaux ad hoc, et ainsi améliorer la qualité de service. Suite aux résultats obtenus dans [15] nous remarquons que H.264 est plus intéressant pour être notre application de base que les autres standards. En effet, dans son fonctionnement normal, H.264 présente déjà

des avantages par rapport aux autres standards, ce qui le rend plus adapté aux réseaux ad hoc.

2.4.2 Caractéristiques de H.264/AVC

L'aspect le plus important dans le standard H.264/AVC est la séparation entre le niveau de codage et le niveau de paquetisation. En effet dans le standard H.264/AVC deux niveaux sont définis, le VCL (Video Coding Layer), niveau de codage de la vidéo, et le NAL (Network Abstraction Layer) niveau d'abstraction du réseau, qui rend le même paquet codé utilisable sur différents types de réseaux. De plus ce standard présente plusieurs outils et améliorations par rapport aux standards précédents. Ces outils seront présentés rapidement dans ce document. Pour plus de détails sur les outils de H.264/AVC nous vous conseillons de vous référer à [16].

Le codage de la vidéo se fait au niveau du VCL. Pour bien comprendre le codage, il est important de savoir qu'une image est constituée de petites composantes. Chaque composante représente soit la luminance Y, soit la chrominance Cb et Cr. La luminance étant la luminosité de cette composante, et la chrominance étant la couleur de cette composante. Ces composantes sont regroupées entre elles pour former un *macroblock* (MB). Un MB peut prendre plusieurs tailles représentées par le nombre de composantes en longueur multiplié par le nombre de composantes en largeur, soit donc de la forme $X \times Y$. Comme l'œil humain est plus sensible à la luminance qu'à la chrominance, un MB contient plus de composantes de luminances que de chrominance. Les MBs sont regroupés en tranches (*Slices*), une image peut être constituée d'une ou plusieurs tranches. Il existe plusieurs types de tranches, les plus importants sont les tranches de type I, type P et type B. Les tranches de types I sont codées en *intra*, donc chaque composante est codée par rapport aux autres composantes dans la même trame, on parle de codage spatial, car les composantes de prédiction varient dans l'espace mais pas dans le temps. Les tranches de types P sont codées en *intra* et *inter*, donc les composantes peuvent être codés par rapport à d'autres composantes dans d'autres trames, en utilisant un seul vecteur de compensation de mouvement [9], on parle de codage spatio-

temporel, car les composantes de prédiction varient dans l'espace ainsi que dans le temps. Les tranches de type B sont codées comme ceux de type P mais avec la possibilité d'utiliser deux vecteurs de compensation de mouvement.

En plus du codage de base, plusieurs outils ont été ajoutés dans H.264/AVC pour rendre le codage plus efficace. Les plus importants sont la compensation de mouvement en utilisant plusieurs images de références, la compensation de mouvement utilisant des blocs de tailles différents, la séparation entre l'ordre de référencement et celui de présentation, deux types de codages entropiques (entropy coding) CAVLC (*Context-Adaptive Variable-Length Coding*) et CABAC (*Context-Adaptive Binary Arithmetic Coding*) ainsi que d'autres outils [16]. De même H.264 présente des outils de correction d'erreurs qui le rend plus robuste que ses prédécesseurs aux paquets perdus ou erronés. Parmi ces outils nous citons la structure de l'unité NAL, taille de tranches flexible, rangement de MB flexible, ainsi que d'autres [16].

Après le codage de la vidéo, la couche NAL permet de modéliser l'information obtenue de façon compatible avec différents genres de réseaux. La couche NAL construit des paquets à partir des bits générés par la couche VCL. L'entête du paquet NAL contient l'information sur la façon avec laquelle le contenu est codé.

Deux types de paquets NAL existent, le premier contient l'information produite par la couche VCL, l'autre contient de l'information sur la façon dont la série (plus ou moins longue) d'unités NAL doit être décodée. Cette information est l'ensemble de paramètres (*parameter set*). De même, deux types d'ensemble de paramètres existent, les paramètres de séquence (*sequence parameter set*) et les paramètres de l'image (*picture parameter set*). Une séquence est une suite d'images codées, qui constitue une partie de la vidéo codée. Les unités NAL contenant un VCL appartenant à une séquence, pointent vers l'ensemble de paramètres déjà reçus pour le décodage de cette séquence. D'autre part, les paramètres de l'image sont utilisés pour le décodage d'une certaine image dans une séquence. Ainsi on minimise les entêtes, puisque l'information sur le décodage de plusieurs paquets est reçue une seule fois.

Une fois le paquet NAL construit, il peut être envoyé sur des réseaux en flot de bits, pour cela il doit être délimité par un code de début et de fin. Comme il peut être envoyé sur un réseau de flot de paquets où les code début et de fin ne sont pas nécessaires.

2.5 Architecture Cross-Layer

La révolution *cross-layer* à surgie suite à la commercialisation des réseaux sans fil. Dans ces réseaux, les modèles existants, comme la pile OSI ou TCP/IP, ne sont pas adaptés. Pour cela les chercheurs se sont orienté vers la modification de ces standards, où les couches ne communiquent qu'avec les couches adjacents, pour permettre aux couches de communiquer avec des couches qui ne leur sont pas adjacents. Dans [17] on défini le design *cross-layer* comme tout design qui viole l'architecture de référence pour la communication entre couches.

De plus dans [17] on défini quatre types de design *cross-layer* dépendamment du type de violation qui a eu lieu :

- Création de nouvelles interfaces : ca revient à créer de nouvelles interfaces pour que les couches non adjacentes puissent communiquer entre elles. De plus les auteurs dans [17] divisent cette catégorie en trois sous-catégories dépendamment du sens du flot entre les couches, de haut en bas, de bas en haut ou dans les deux sens.
- Le fusionnement de couches adjacentes : ca revient à fusionner deux couches adjacentes en une seule, sans le besoin de créer de nouvelles interfaces pour cette nouvelle super-couche.
- Couplage entre couches sans la création de nouvelles interfaces : ca revient à créer des protocoles de couches en prenant en compte le fonctionnement des autres couches, même celles qui ne sont pas adjacentes.
- Calibrage vertical à travers les couches : ca revient à faire une modification des paramètres qui passent par toutes les couches.

En plus de la définition de différents types de design, dans [17] on défini différents type de communication entre les couches :

- Communication directe : où les couches communiquent directement l'une avec l'autre, à travers de nouvelles interfaces.
- Une base de données partagée : où une base de données est créée, comme une nouvelle couche qui interagit avec toutes les autres couches. Ainsi une couche peut introduire une variable à la base de données, qu'une autre couche peut lire et utiliser.
- Une toute nouvelle architecture : où la communication entre couche ne suit pas les modèles déjà définis.

Pour notre recherche nous allons utiliser une architecture cross-layer avec calibrage vertical, avec création d'une base de données partagée. Cette architecture étant la plus facile à implémenter et la mieux adaptée à notre recherche.

2.6 Etudes existantes dans le domaine

2.6.1 Délais et gigue dans la vidéoconférence

Dans [5], AODV a été choisi comme protocole de routage, et les performances de H.264 ont été testées dans différents environnements.

Le premier test, pour évaluer l'impacte de la mobilité et de la congestion sur le PSNR, montre que la mobilité cause des pertes de qualité moins fréquentes mais plus visibles que ceux causées par la congestion. De plus les tests sur le délai et la gigue, montrent que la mobilité et la congestion influencent différemment ces deux paramètres. A forte mobilité très peu de paquets ont un très grand délai de plus de 6s, les autres ont un délai acceptable. Alors qu'en cas de congestion, tous les paquets qui arrivent à destination, arrivent avec un délai de moins de 1s. En ce qui concerne la gigue, les résultats sont aussi différents pour la mobilité et la congestion. Pour la première, la gigue est moins fréquente mais plus aigue, alors que pour la deuxième la gigue est distribuée moyennement tout au long de la connexion. Les résultats obtenus sont reportés dans [5]. Pour remédier à ces problèmes, les auteurs ont recouru à deux paramètres dans H.264.

Le premier est le nombre de trames de références, et leurs tests montrent que l'utilisation d'une seule trame de référence donne les meilleurs résultats en terme de

vitesse de correction. Le deuxième est le *MacroBlock Intra-Updating*. Plusieurs modes ont été testés et les résultats montrent que le *1/3 random MB update* donne les meilleurs PSNR pour un taux de perte de 20% ainsi que celui de 4%.

D'autre part, l'étude dans [19] se base sur le nombre de sauts entre source et destination. En effet, Dans ce document on propose deux solutions pour améliorer la qualité de la transmission vidéo en temps réel sur un réseau ad hoc. Les deux solutions visent à minimiser le délai de bout en bout, et ceci pour répondre aux contraintes de délais dans les applications temps réel.

Ce but est atteint premièrement en augmentant la puissance de transmission des nœuds. En faisant ceci on parvient à minimiser le nombre de sauts entre la source et la destination, mais on augmente aussi l'interférence entre les nœuds. L'étude faite dans ce document montre que si on veut diminuer le délai de bout en bout, on doit diminuer aussi le nombre d'utilisateurs dans le réseau conformément à l'équation suivante [19]:

$$n' \approx \left(\frac{L'}{L} \right)^2 n$$

Où L et L' sont l'ancienne et la nouvelle distance respectivement en sauts entre la source et la destination, et n et n' sont l'ancien et le nouveau nombre d'utilisateurs respectivement dans le réseau.

D'autre part, cette étude montre que si on ne veut pas diminuer le nombre d'utilisateurs, on ne peut pas augmenter la puissance de transmission des nœuds. Par contre ce qu'on propose dans ce document serait de diminuer la taille des paquets conformément au débit disponible.

Dans le document [19] on trouve les résultats obtenus pour des simulations faites avec des séquences vidéo de 100 trames à 10 trames/s en QCIF et 300 trames à 30 trames/s en CIF, sur un réseau ad hoc de bande passante partagée de 3.6 Mbps.

Les résultats montrent que quand la distance entre la source et la destination augmente, le débit vidéo doit être diminué en fonction, et ceci entraîne une dégradation dans la qualité de la vidéo.

Mais cette méthode reste statique puisque la décision sur le débit ou le nombre d'utilisateurs se fait au lancement de l'application. En effet, aucun mécanisme d'adaptation de l'application au réseau ou vice versa en temps réel n'est proposé.

2.6.2 Erreurs de transmission dans la vidéoconférence

Dans un autre document [18], les auteurs testent les performances du standard H.264 en ce qui concerne la résistance aux erreurs induits par les réseaux, tel que les réseaux ad hoc. La simulation a été faite en utilisant le model de test JM3.9a, avec des séquences vidéos de 150 et 300 trames de long, en format QCIF. Les paramètres pris en comptes pour l'évaluation des performances sont le PSNR, ainsi qu'un paramètre introduit par les auteurs, et qui représente la robustesse (R), qui est la capacité du standard à maintenir la qualité de l'image même en présence d'erreurs.

Les simulations on été faites dans différents états du réseau. Et ainsi en variant différents paramètres les auteurs ont essayé de parvenir au meilleur rendement.

Le premier test effectuer est un test de référence, où tous les paramètres de H.264 sont laissés à leur valeur par défaut. Suite à ce test, un deuxième test est effectué où les paramètres ont été réglés à leur valeur optimale. Les paramètres principaux qui ont été activés sont : *Hadamart Transform*, le codage *CABAC* et le *Rate Distortion Optimization*. De même le mode complètement flexible de la fonction de transformation adaptative du block a été activé.

Les résultats montrent que dans un réseau à forte perte de paquets (jusqu'à 3%), le rendement augmente en augmentant la fréquence des *intra frames*. De plus, dans les réseaux à pertes en rafales du quart ou trois quarts de la taille de la trame, l'utilisation de FMO (*Flexible Macroblock Ordering*) avec un envoi aléatoire de *intra Macroblock* pour la mise à jour d'une partie des *macroblock* (le 1/3 dans une trame) donne un taux d'erreur de seulement 1.4%.

Donc la qualité de la vidéo transmise sur un réseau à fort taux d'erreur, est améliorée en réglant les paramètres introduits dans ce document. Mais les tests ne sont pas complets. Premièrement ces tests ne prennent pas en considération les pertes de paquets suite à des coupures de liens dans les réseaux, bien que ce problème est très fréquent dans les réseaux ad hoc. Deuxièmement les tests se font pour un débit constant, alors que dans un réseau ad hoc nous voudrions plutôt que le débit s'adapte dynamiquement à l'état du réseau. Ainsi si le réseau est congestionné ou un lien est coupé, nous voudrions que le débit diminue. Et inversement si le réseau fonctionne normalement nous voulons que le débit augmente.

2.6.3 Effet des coupures de liens sur la qualité

Une étude conduite sur la taille des paquets pour l'amélioration de la PSNR est faite dans [20]. Dans cette méthode, la décision quant à la taille des paquets est prise en fonction de la valeur de disponibilité des liens. L'étude expérimentale a été faite dans deux environnements. Le premier est un environnement où la disponibilité des liens et l'interférence sont grandes (zone urbaine), et l'autre où la disponibilité des liens et l'interférence sont faibles (autoroute). Suite à cette étude de la corrélation entre l'état des liens, la taille des paquets et le PSNR, les auteurs sont arrivés à la conclusion que pour une disponibilité des liens de plus de 95%, l'utilisation de paquets de petite taille donne un meilleur PSNR que l'utilisation de paquets de grande taille. D'autre part, pour une disponibilité de moins de 95%, c'est l'utilisation de paquets de grande taille qui donne un meilleur PSNR. Ces résultats ont menés au développement d'un algorithme qui décide sur la taille des paquets en fonction de la disponibilité des liens. La disponibilité des liens est mise à jour continuellement durant la transmission pour pouvoir changer dynamiquement la politique à utiliser pour la taille des paquets. Cet algorithme a permis d'améliorer le PSNR de H.264, de 5 dB pour le scénario sur l'autoroute et de 2 dB pour le scénario en zone urbaine, par rapport à H.264 avec une taille fixe des paquets.

Cette méthode ne prend en considération que la taille des paquets, sans prendre en compte le contenu des paquets. En effet, en plus de l'état du réseau, on doit savoir le type de données contenues dans les paquets pour décider de la meilleure taille. Par exemple pour des *I-Frames*, il serait meilleur d'utiliser des paquets de petite taille même si la disponibilité et l'interférence sont faibles, ce qui n'est pas le cas pour les *P-Frames* ou les *B-Frames*.

2.6.4 Application adaptative

La seule proposition existant dans la littérature, pour rendre l'application de vidéoconférence adaptative à l'état du réseau, est faite dans [21]. Dans ce document, on propose une méthode pour améliorer la qualité de service des applications de vidéoconférence, basée sur l'échange d'informations entre couches.

Ce modèle est basé sur la variation dynamique des paramètres de codage et de correction d'erreur, en fonction du nombre de sauts entre la source et la destination.

Les auteurs montrent d'abord comment la bande passante disponible sur un chemin diminue quand le nombre de sauts sur ce chemin augmente. Il en résulte que pour garder une bonne qualité de service, on doit diminuer le débit de l'application quand le nombre de saut sur le chemin augmente.

Un protocole de routage comme AODV contient l'information sur le nombre de sauts dans sa table de routage. L'échange d'information entre couche, permet à la couche application de retrouver cette information et d'adapter son débit en fonction. Ainsi quand le protocole de routage effectue un changement de route, l'application d'abord arrête l'envoi de paquets jusqu'à ce qu'une nouvelle route soit établie, les paquets perdus sont envoyés sous forme de paquets redondants contenant seulement le vecteur de déplacement, ensuite l'application à la source commence la transmission des nouveaux paquets par une *I-frame*, puis effectue un changement de débit en changeant le paramètre du quantificateur et continue à envoyer des *P_frames*.

Donc en plus du changement de débit dynamique, cette méthode permet une meilleure protection contre les erreurs en utilisant des paquets redondants pour le *P-frames*,

contenant seulement les informations indispensables au bon décodage de l'image, ainsi que le FEC pour les I-frames.

Les résultats des simulations effectuées montrent que l'augmentation du QP, du FEC et de paquets redondants, quand le nombre de sauts entre la source et la destination augmente, diminue la distorsion de l'image au décodeur.

Les améliorations atteintes par cette méthode sont très importantes, surtout que le *Cross-Layer Feedback* est utilisé, et que des paramètres de l'application H.264 sont changés dynamiquement. Mais ce travail n'est pas complet puisque d'autres paramètres de H.264 peuvent être exploités dans le même esprit, afin d'atteindre encore de meilleurs résultats.

CHAPITRE 3 MÉTHODE D'ADAPTATION DE L'APPLICATION

Pour améliorer la qualité de service de l'application de vidéoconférence dans les réseaux ad hoc, nous décidons de rendre l'application adaptative à l'état du réseau. L'architecture *cross-layer* que nous utilisons permet à l'application de connaître en tout moment l'état du réseau, en ayant accès à l'information collectée par les couches basses. Ainsi, et dépendamment de l'état du réseau, l'application passera d'un mode de fonctionnement à un autre dynamiquement. Pour atteindre ce but, nous allons commencer par définir les états possibles d'un réseau ad hoc. Pour chaque état du réseau nous allons définir le comportement de l'application de vidéoconférence, et la façon avec laquelle le passage d'un mode de fonctionnement à l'autre va se faire. Puis nous donnerons les caractéristiques de chaque mode de fonctionnement défini, donc les valeurs des paramètres qui définissent un mode de fonctionnement de l'application à la source. Pour la détection de l'état du réseau, et le choix du mode de fonctionnement de l'application, nous définirons l'architecture *cross-layer* la mieux adaptée à notre application, ainsi que les informations échangées entre les différentes couches. Finalement nous ferons une synthèse du chapitre en faisant une représentation en diagramme du fonctionnement de notre application.

3.1 Hypothèses

Les hypothèses considérées dans le développement de notre solution sont les suivantes :

- Le réseau considéré est un réseau ad hoc, où tous les nœuds sont mobiles, et où l'état du réseau ne peut pas être prévu à l'avance.
- Le protocole de routage utilisé dans ce réseau est AODV.
- Le standard de vidéoconférence utilisé est H.264/AVC.
- Le protocole de transport est RTP/RTCP sur UDP.

3.2 États du réseau et mode de fonctionnement de l'application

Nous définissons sept états dans lesquels le réseau ad hoc peut être au moment de la session de vidéoconférences. Chaque état n'est pas fixe durant toute la session, mais le réseau peut passer d'un état à l'autre à tout instant, sans que ce passage ne puisse être prévu à l'avance. Nous définirons de même le mode de fonctionnement de l'application à la source dépendamment de l'état dans lequel se trouve le réseau.

3.2.1 Aperçu sur les modes de fonctionnement

Avant d'introduire les différents états possibles d'un réseau ad hoc, nous trouvons qu'il serait intéressant de définir brièvement les modes de fonctionnements de l'application à la source. Pour plus de détails sur ces modes de fonctionnement nous vous referons à la partie 3.3.

- Mode *High* : Ce mode offre une vidéo de haute qualité. Il a été conçu dans le but d'envoyer beaucoup d'information sur le réseau. Ainsi nous aurons assez de paquets propres à notre communication dans le réseau, pour maintenir le flot de paquet jusqu'à la source pour une certaine durée, même en cas de coupure de liens.
- Mode *Normal1* : ce mode offre une vidéo de bonne qualité. Il a été conçu dans le but de maintenir un bon nombre de paquets propres à notre communication en circulation dans le réseau. Mais son débit est inférieur à celui du mode *High*, pour ne pas causer des congestions dans le réseau.
- Mode *Normal2* : le but et les caractéristiques de ce mode, ne diffèrent pas beaucoup de ceux du mode *Normal1*. Ce mode a été quand même conçu pour améliorer légèrement la qualité de la vidéo quand l'état du réseau le permet.
- Mode *Congestion Prevention (CP)* : ce mode offre une vidéo de qualité moyenne. Il est utilisé dans des cas où l'information collectée du réseau informe d'une possibilité de congestion qui se forme. Nous allons donc

utiliser ce mode pour diminuer le trafic et rendre la communication mieux adaptée à l'état du réseau.

- Mode *Prevention* : ce mode offre une vidéo de qualité moyenne. Il a été conçu pour être utilisé dans des cas où l'information collectée du réseau informe d'une possibilité de coupure de lien. Nous utilisons ce mode pour prendre des mesures préventives en cas de coupure de liens, en diminuant le débit, et rendant la communication plus robuste contre les erreurs.
- Mode *Recovery* : ce mode offre une vidéo de qualité médiocre. Il est utilisé seulement dans le cas où la coupure de liens est confirmée. Dans ce cas, nous allons minimiser le débit au maximum en attendant qu'un nouveau chemin vers la destination soit trouvé.
- Mode *Congestion Recovery (CR)* : ce mode offre une vidéo de qualité médiocre. Il est utilisé dans le cas où la congestion est confirmée dans le réseau. Dans ce cas, nous allons diminuer le débit tout en conservant une qualité de la vidéo meilleure que celle obtenue dans le mode *Recovery*.

3.2.2 Etats du réseau

3.2.2.1 Chemin établi

C'est l'état normal de fonctionnement d'un réseau ad hoc. Cet état est atteint suite à une demande de connexion lancée par la source. Le protocole de routage (AODV dans notre cas) effectue une recherche de chemin entre la source et la destination, et retourne dans le cas normal le plus court chemin *figure 3.1*. Pour les premières millisecondes après l'établissement de la connexion, nous considérons que le chemin est stable et que les mémoires tampons dans les nœuds intermédiaires ne sont pas surchargées. Nous exploitons donc cette stabilité en envoyant pour une courte durée beaucoup d'information sur le réseau en utilisant de grands paquets. C'est pour cette raison que du côté de l'application à la source, après l'établissement du chemin entre cette dernière et la destination, l'application se met en mode *High* pour un temps équivalent au temps que met un paquet à faire un aller retour sur le réseau (*RoundTripTime ms*). Dans ce

mode de fonctionnement l'application envoie beaucoup d'information avec une haute qualité de la vidéo. Suite à cette étape, nous considérons que les premiers paquets sont déjà arrivés à la destination. De même nous considérons que le réseau contient assez de paquets propres à cette session pour maintenir le flot d'information vers la destination en cas de coupure de liens, jusqu'à ce qu'un nouveau chemin soit établi. Après *RoundTripTime* ms nous pouvons passer en mode *normal1*. Dans ce mode l'application à la source fonctionne normalement, donc la qualité de la vidéo est un peu réduite et de même le débit sur le réseau. Ce mode est maintenu jusqu'à la réception du premier rapport du récepteur RR (Receiver Report). Quand ce rapport est reçu, on vérifie les valeurs des champs *interarrival jitter* et *fraction lost*. Si ces valeurs sont petites, nous considérons que le réseau fonctionne bien, et que nous pouvons augmenter un peu la qualité de la vidéo et par suite le débit sur le réseau, donc nous passons dans le mode *normal2*. Le fonctionnement dans les modes *Normal1* et *Normal2* est pour une durée équivalente à la durée de validité du chemin, moins la durée de l'intervalle de temps durant lequel nous devons recevoir un message pour confirmer la disponibilité du lien, message *HELLO* dans le cas de AODV, (*Lifetime* – *HELLO_INTERVAL* ms). Si après cette durée, la source n'a pas reçue un message *HELLO* ou une mise à jour de ce chemin, elle passe en mode *Prevention*. Le passage à ce mode est fait, parce que nous présumons qu'il peut y avoir une coupure de lien quelque part sur le chemin. En passant dans ce mode, nous diminuons légèrement le débit sur le réseau pour ne pas causer un débordement des mémoires tampons des nœuds intermédiaires, jusqu'à ce que l'état actuel du réseau soit confirmé. AODV accepte un certain nombre de pertes de paquets *HELLO* (*ALLOWED_HELLO_LOSS*), avant de décider de l'invalidité d'un lien. Puisque l'intervalle d'attente d'un message *HELLO* est de *HELLO_INTERVAL*, cela nous donne *ALLOWED_HELLO_LOSS * HELLO_INTERVAL* ms. Si après cette durée nous recevons un nouveau message *HELLO*, nous considérons que le chemin est encore valable et donc nous repassons en mode *normal1*. Sinon, au cas où nous ne recevons pas de message *HELLO*, ou nous recevons un message *RERR*, ça veut dire qu'il y a une coupure de lien.

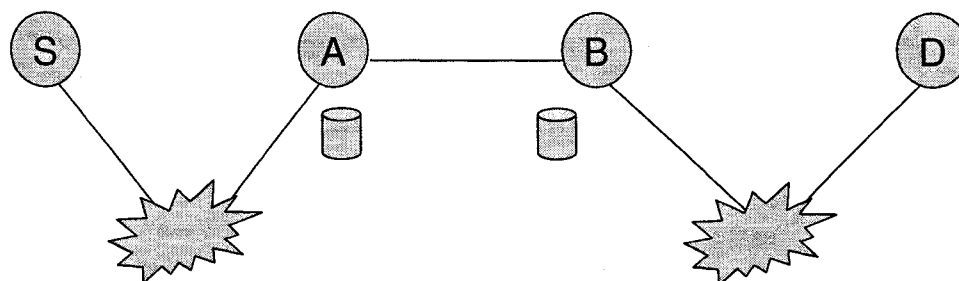


Figure 3.1 Chemin établi

3.2.2.2 Chemin établi avec grande gigue

Le paquet de contrôle RTCP *Receiver Report (RR)* contient deux champs qui nous intéressent. Le champ contenant la fraction de paquets perdus dans le réseau (*fraction lost*), et la valeur moyenne de la gigue entre deux arrivées consécutives (*interarrival jitter*). Si un RR est reçu avec une valeur grande dans le champ *interarrival jitter* mais une valeur petite dans le champ *fraction lost*, nous considérons que le chemin pris présente beaucoup de variation dans la capacité des liens *figure 3.2* et qu'une congestion peut se produire si le débit sur le réseau reste le même. Comme prévention, pour ne pas avoir de congestion sur le réseau, l'application à la source passe en mode *congestion prevention* pour une durée *RoundTripTime* ms. En passant dans ce mode nous diminuons le débit de l'application, ainsi que la taille des paquets, donc nous allégeons un peu la charge sur les mémoires des nœuds intermédiaires, ce qui doit minimiser les chances d'avoir une congestion. De plus, en utilisant des paquets de petite taille, nous diminuons la possibilité de rejet de paquet, ainsi que la gigue [31]. Après *RoundTripTime* ms, nous aurons essayé d'alléger les mémoires tampons des nœuds intermédiaires, donc nous pouvons essayer de fonctionner dans un mode normal. Nous repassons donc au mode *normal*, ceci pour ne pas vider complètement les mémoires tampon des nœuds intermédiaires, et pour utiliser le plus efficacement possible les ressources.

Dans les deux cas, nous devons diminuer le débit d'avantage, c'est pourquoi nous passons dans le mode de fonctionnement propre au cas de coupures de liens, le mode *recovery*.

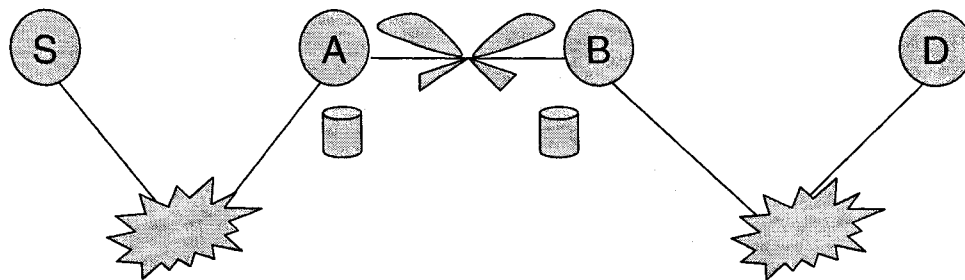


Figure 3.3 Chemin établi avec congestion

3.2.2.4 Coupure de lien intermédiaire

Une coupure de lien intermédiaire peut être détectée suite à réception d'un RERR avec le champ N mis à 0 et le numéro de séquence du dernier paquet envoyé correctement. Le nœud où la coupure a eut lieu, nœud A dans *figure 3.4*, envoie donc un RERR et essaie de réparer localement le lien. Puis A et tous les liens entre A et la source rejettent tous les paquets peu importants appartenant à la communication, dans leur mémoire tampon. B continue à envoyer les paquets dans sa mémoire vers la destination. La source passe en mode *recovery* et réinitialise la découverte de chemin avec un nouvel identificateur (*DestSeqNum*). En fonctionnant ainsi nous diminuons au maximum le débit de la source sur le réseau, sans pour autant arrêter complètement l'envoi de la vidéo, au cas où le nœud intermédiaire arrive à rétablir une connexion vers la destination. De même en réinitialisant la découverte de chemin, nous ne perdons pas de temps à la source, au cas où A ne trouve pas un autre chemin vers la destination.

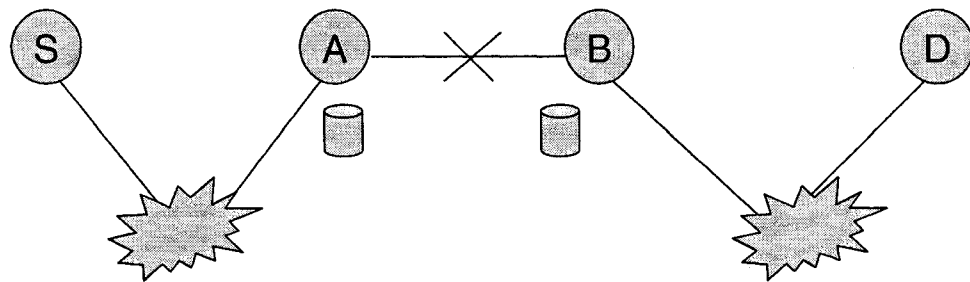


Figure 3.4 Coupure de lien intermédiaire

Suite à cet état, trois cas se présentent. Premièrement, nous pouvons avoir le nœud A qui répare localement cette coupure de lien *figure 3.5*. Dans ce cas A envoie un message d'erreur (*RERR*) à la source avec le champ N mis à 1, nous informant d'une réparation locale, et l'information sur le nouveau chemin établi entre elle et la destination. Puis A continue l'envoi des paquets normalement sur le nouveau chemin. Du côté de la source, nous passons en mode *high* pour une durée d'un trajet dans un sens sur le réseau ($RoundTripTime/2$), pour re-remplir les mémoires des nœuds intermédiaires. Puis nous repassons en mode *normal* et nous reprenons le fonctionnement normal de l'application.

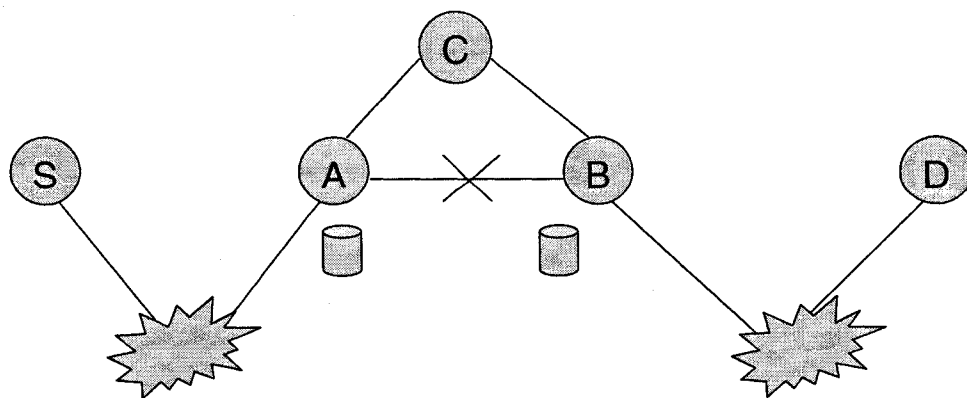


Figure 3.5 Lien réparé localement

Deuxièmes, un nouveau chemin est trouvé de la source à la destination *figure 3.6*. Dans ce cas aussi, du côté de la source, nous passons en mode *high* pour une durée *RoundTripTime* pour re-remplir les mémoires des nœuds intermédiaires. Puis nous repassons en mode *normal* et nous reprenons le fonctionnement normal de l'application. De plus, tous les nœuds dans l'ancien chemin jettent tous les paquets appartenant à la communication, et la source recommence le transfert à partir du dernier paquet envoyé correctement.

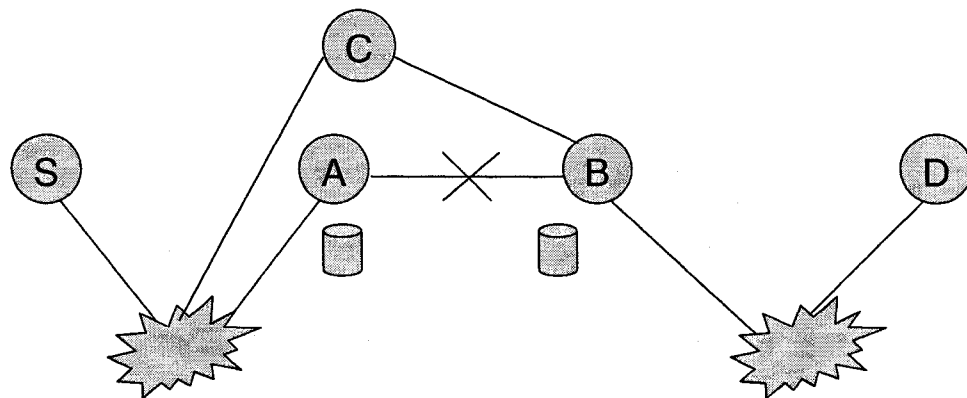


Figure 3.6 Lien réparé de la source

Et finalement, le troisième cas possible, la source et le nœud intermédiaire trouvent tous les deux un nouveau chemin *figure 3.7*. Dans ce cas A envoie un RERR à la source avec le champ N mis à 1 l'informant d'une réparation locale, et l'information sur le nouveau chemin établi entre elle et la destination. Puis elle continue l'envoi des paquets normalement sur le nouveau chemin. La source continue l'envoi des paquets mais sur le nouveau chemin trouvé. Puis elle compare les deux chemins trouvés et garde le

meilleur. Après le choix du meilleur chemin, nous passons en mode *high* pour une durée *RoundTripTime* pour re-remplir les mémoires des nœuds intermédiaires. Puis nous repassons en mode *normal* et nous reprenons le fonctionnement normal de l'application.

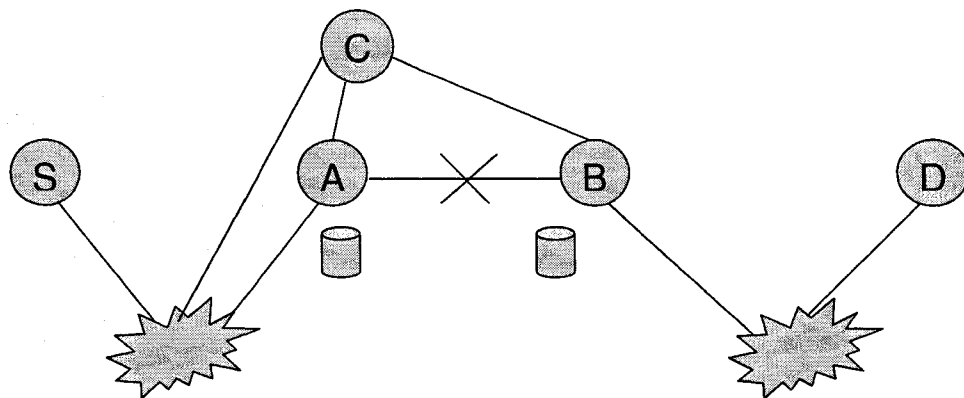


Figure 3.7 Lien réparé de la source et localement

Ainsi nous aurons couvert tous les états possibles du réseau, ses besoins dans chacun des cas en termes de débit à la source et de type de paquets échangés. De plus, nous aurons précisé le mode de fonctionnement dans chaque cas, ainsi que les modalités de passage d'un mode à l'autre selon le besoin.

Notre but est de garder la qualité de la vidéo à la destination la plus stable possible. Pour cela nous allons essayer de définir les modes de fonctionnement en sorte d'avoir premièrement, le plus de paquets qui parviennent à la destination, et deuxièmement, le meilleur niveau de PSNR pour les paquets reçus. De bonnes valeurs pour ces deux critères ne sont pas toujours possibles, c'est pourquoi, et dépendamment de l'état du réseau, nous allons décider d'améliorer l'une au dépend de l'autre. Ainsi, quand le réseau fonctionne normalement, notre objectif va être d'envoyer une vidéo de bonne

qualité. Le nombre de paquets reçus ne doit pas être un problème, puisque l'état du réseau est favorable à la bonne réception des paquets à la destination. D'autre part quant le réseau est congestionné, ou qu'il y a une coupure de lien, notre but va être de recevoir le maximum de paquets, même si nous savons que ce nombre ne sera pas très grand. La qualité de la vidéo va nécessairement se détériorer, premièrement car on doit minimiser le débit sur le réseau, deuxièmement car un réseau dans cet état induit beaucoup d'erreurs dans les paquets.

Ayant fixé ces objectifs, il nous est facile de prendre les décisions concernant les paramètres à donner à l'application à la source.

3.3 Caractéristiques des modes de fonctionnement

Dans la partie précédente, nous avons défini les différents modes de fonctionnement, ainsi que leur champ d'application. Dans cette partie nous allons préciser les caractéristiques de chaque mode de fonctionnement, en précisant les paramètres à utiliser à la source. Les paramètres à modifier pour obtenir les différents modes, appartiennent à différentes couches. En effet nos modifications porteront sur les paramètres de codage niveau VCL et NAL de la couche application, ainsi que les paramètres de RTP de la couche transport.

3.3.1 Paramètres niveau application

Au niveau de l'application, nous utilisons les paramètres qui influencent le débit sur le réseau, ainsi que la qualité de la vidéo transmise.

Le standard H.264/AVC définit plusieurs profils de fonctionnement, qui sont *Baseline*, *Extended*, *Main*, *High*, *High 10*, *High 4 :2 :2* et *High 4 :4 :4 Predictive*. Chacun de ces profils offre une série de fonctionnalités qui le différencie des autres profils. Les fonctionnalités que nous recherchons dans le profil que nous utiliserons sont :

- Les fonctionnalités de base, tel que les trois types de trames I qui est une trame codée en spatial seulement, sans aucune référence vers les trames précédentes, P qui est une trame codée en temporel et spatial, avec un seul vecteur de référence vers les trames précédentes et B qui est une trame codée

en temporel et spatial avec plusieurs vecteurs de référence vers les trames précédentes.

- La possibilité d'utilisation de plusieurs trames de référence.
- Possibilité d'utilisation des deux codages entropique, CAVLC (Context-Adaptive Variable-Length Coding), qui est un codage efficace et peu complexe, et CABAC (Context-Adaptive Binary Arithmetic Coding), qui offre un codage sans perte mais au prix d'une complexité élevée. Nous allons préférer cette fonctionnalité à FMO (Flexible Macroblock Ordering) et ASO (Arbitrary Slice Ordering), puisqu'elle présente un grand avantage côté débit et qualité de la vidéo.
- Possibilité d'utilisation du format 4 :2 :0 et d'échantillonnage sur 8 bits.

Ayant ces caractéristiques bien définies, nous pouvons choisir le profil dans lequel l'application doit fonctionner. En ce référent à l'annexe A de [9] nous pouvons voir que le meilleur profil qui répond à nos besoins est le profil *High*. En effet ce profil, même s'il ne permet pas d'avoir un ordre flexible pour les *MacroBlock* comme le profil *Extended*, mais il offre deux types de codage entropique, CAVLC et CABAC, qui sont très importants pour notre application.

Ayant décidé du profil à utiliser, l'étape suivante serait de choisir les niveaux dans lesquels l'application doit fonctionner. L'annexe A de [9], définit 16 niveaux de fonctionnement, allant de 1 à 5.1. Chacun de ces niveaux permet d'atteindre un certain débit maximal, qui augmente avec le niveau utilisé. De plus chaque niveau permet d'utiliser un certain nombre de fonctionnalités différentes. Dans ce même annexe on définit les paramètres permis pour chaque niveau de fonctionnement, ainsi que le débit maximal atteint par chaque niveau de fonctionnement. Pour notre recherche nous allons utiliser les niveaux les moins exigeants en paramètres et en débit. Cette décision est prise d'abord dans le but de minimiser la puissance requise pour le traitement de la vidéo à la source et à la destination, ainsi que l'espace mémoire requis pour le stockage de la vidéo codée. De plus, puisque le réseau utilisé est un réseau ad

hoc, et puisque nous désirons que le réseau puisse supporter un grand nombre de nœuds communiquant en même temps entre eux, nous allons choisir les niveaux de fonctionnement les moins gourmands en bande passante. Donc après avoir fixé ces buts, et en se référant à l'annexe A de [9], nous avons décidé d'utiliser les niveaux 1, 1b, 1.1 et 1.2, pour atteindre un débit maximal de 12.288 Mbits/s.

D'autre part, la qualité de codage de la vidéo est directement liée aux valeurs des paramètres de quantification (QP) à la source. La qualité avec laquelle les trois types de trames, I, P et B sont codées dépend de la valeur donnée aux paramètres QPI, QPP et QPB respectivement. Plus cette valeur est petite et plus la qualité augmente, et inversement. Mais en augmentant la qualité, donc en diminuant les valeurs des QPs, nous augmentons le débit de la source. Ces paramètres vont donc être utilisés dans les modes de fonctionnement pour faire varier le débit et la qualité de la vidéo codée.

De plus, la vidéo codée est envoyée par séquences sur le réseau. La couche NAL s'occupe de la forme de ces séquences. La forme de la séquence est le type des trames qui la constituent, le nombre de trames de chaque type et la succession des trames de chaque type. Dans une séquence, plus il y a des trames de types I et P, et plus il y a de chance que cette séquence soit décodée avec une bonne qualité, et inversement. Mais comme les trames de types I et P sont plus grandes que ceux de type B, en augmentant leur nombre, nous augmentons aussi le débit. Donc la forme des séquences tout comme les QPs sera un des paramètres qui définira les différents modes de fonctionnement de la source.

3.3.2 Paramètres niveau transport

Le format des paquets au niveau transport est aussi un paramètre important dans les différents niveaux de fonctionnement de la source. Comme nous l'avons vu précédemment (voir partie 2.2), RTP définit plusieurs types de paquets pour le transport de la vidéo suivant le standard H.264/AVC. Les différents types de paquets définis, sont caractérisés par différents type d'encapsulation, et différentes tailles.

Dans les réseaux à fort taux d'erreurs, comme les réseaux ad hoc, les paquets de petite taille sont moins sensibles aux erreurs que ceux de grande taille. De même, dans les réseaux à fort trafic, donc forte probabilité de congestion, la taille des paquets utilisés doit être petite [22]. Alors que dans le cas de grande possibilité de coupure de liens, la taille de paquets utilisés doit être grande [20]. La décision sur la taille des paquets niveau transport, dépend donc de l'information collectée sur l'état du réseau des couches plus basses, et de l'information sur le type de données envoyées par les couches supérieures. En effet une IDR [9] est plus importante que des trame de type P ou B, alors il est plus intéressant de l'envoyer dans de petits paquets, même si le réseau fonctionne sans coupure de lien ni congestion.

3.3.3 Mode High

Ce mode est utilisé pour *RoundTripTime* ms après chaque nouvel établissement de chemin. En utilisant ce mode, nous essayons d'introduire un grand nombre de paquets sur le réseau. Ces paquets vont remplir les mémoires tampons des nœuds intermédiaires sur le chemin établi, et ainsi, même en cas de coupure de lien, l'arrivée des paquets à la destination se poursuivra pour une certaine durée, dépendamment du lieu de coupure, donc aucune coupure dans la vidéo ne sera ressentie à la destination. Cela va augmenter les chances de retrouver un nouveau chemin vers la destination, sans que cette dernière ne soit affectée par la coupure du lien.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1.2, c'est le niveau qui offre le plus de débit parmi les niveaux retenus. Il est utilisé puisque dans ce mode de fonctionnement notre but est d'augmenter le débit à la source.
- Codage entropique : CAVLC, ce codage est plus léger que CABAC, donc le traitement de la vidéo à la source et à la destination est plus rapide. De plus, même si CABAC est plus robuste contre les erreurs, son utilisation n'est pas avantageuse dans ce mode de fonctionnement car ce dernier sera utiliser

seulement dans le cas où le réseau fonctionne normalement, donc quand le taux d'erreurs dans le réseau est faible.

- Paramètres de quantification : QPI et QPP de petites valeurs, et QPB de grande valeur. Comme les trames de type I et P sont plus importantes pour le décodage que les trames de type B, nous allons leur donner des valeurs de QPs plus petites. Ainsi nous aurons une bonne qualité à haut débit, sans consommation de bande passante inutilement pour des trames de type B.
- Séquence vidéo : nous commencerons la transmission dans ce mode avec une trame IDR pour continuer la séquence avec la suite normale de trames, avec des trames de type B fréquents. Le passage à ce mode de fonctionnement peut suivre une phase de coupure de lien. Dans cette phase beaucoup de paquets ont été perdu dans le réseau, et donc la qualité de l'image au récepteur est médiocre. Il est donc important de procéder à une mise à jour de l'image au récepteur avant de continuer la transmission normale de la vidéo. De plus l'IDR va annuler toutes les trames de référence le précédents, donc les erreurs reçues dans les anciennes trames n'affecteront plus les nouvelles trames décodées.
- Taille des paquets : FU-A puis STAP-A. Ainsi nous utilisons les paquets de petite taille, de type FU-A qui contient des fragments du paquet à envoyer pour envoyer le IDR, puis nous passons à des paquets de grande taille, de type STAP-A qui contient un agglomérat de paquets à envoyer, pour la suite de la séquence.

3.3.4 Mode Normal1

Le passage à ce mode est fait après le mode *High*, donc après un temps *RoundTripTime* ms. Après avoir repli les mémoires des nœuds intermédiaires, nous devons réduire le débit de la source, pour éviter des congestions dans le réseau. Nous passons donc en mode de fonctionnement normale, où nous diminuons le débit de l'application, et du meme coup la qualité de la vidéo. Mais il se peut que le réseau soit capable de supporter

un plus grand trafic, donc ca nous permet d'avoir une meilleure qualité de la vidéo. A priori, jusqu'à la réception du premier RR de la destination, l'état du réseau nous est inconnue, donc il est important de prendre les mesures nécessaires pour éviter de le surcharger, et ce, jusqu'à la réception d'information prouvant le contraire.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1.1, ce niveau est moins gourmand en bande passante que le niveau 1.2, donc il va permettre de réduire le débit de l'application, pour cela il est utiliser pour ce mode de fonctionnement.
- Codage entropique : CAVLC, ce codage est plus léger que CABAC, donc le traitement de la vidéo à la source et à la destination est plus rapide. De plus, même si CABAC est plus robuste contre les erreurs, sont utilisation n'est pas avantageuse dans ce mode de fonctionnement car ce dernier sera utiliser seulement dans le cas où le réseau fonctionne normalement, donc quand le taux d'erreurs dans le réseau est faible.
- Paramètres de quantification : nous allons augmenter les valeurs de QPI et QPP en concervant la meme valeur pour QPB. Ainsi nous allons diminuer légèrement la qualité des trames de type I et P en augmentant la valeur de leur QP, nous gardons la même qualité pour les trames de type B, puisque celle-ci est déjà faible et ne consomme pas beaucoup de bande passante. Ainsi nous aurons une qualité moyenne à débit plus faible que le mode *High*.
- Séquence vidéo : format normal, trames de types B fréquents. Puisque le passage à ce mode de fonctionnement ce fait généralement après le mode *High*. Nous pouvons donc considérer que la qualité de l'image au récepteur est bonne. La mise à jour de l'image a été faite récemment, donc nous n'avons pas besoin de faire une nouvelle mise à jour. Pour cette raison, nous n'effectuons pas de changements dans la forme de la séquence vidéo.
- Taille des paquets : STAP-A. Nous considérons que le passage à ce mode c'est effectuer assez rapidement, et que la topologie du réseau n'a pas eut le

temps de changer. Le taux d'erreurs de transmission est donc toujours faible. Par contre durant ce mode une coupure de lien peut avoir lieu à tout moment. Pour cette raison, et suivant les résultats obtenus dans [20] et [22], l'utilisation de paquets de grande taille est préférable.

3.3.5 Mode Normal2

Après la réception d'un RR de la destination, si l'information contenue dans ce rapport informe d'un bon fonctionnement du réseau, nous pouvons passer au mode *Normal2*. Le réseau n'étant pas surchargé, nous décidons d'améliorer la qualité de la vidéo transmise, en augmentant ainsi légèrement le débit à la source.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1.1, nous gardons le même niveau que dans le mode *Normal1* pour garder le débit dans les mêmes limites.
- Codage entropique : CAVLC, ce codage est plus léger que CABAC, donc le traitement de la vidéo à la source et à la destination est plus rapide. De plus, même si CABAC est plus robuste contre les erreurs, son utilisation n'est pas avantageuse dans ce mode de fonctionnement car ce dernier sera utilisé seulement dans le cas où le réseau fonctionne normalement, donc quand le taux d'erreurs dans le réseau est faible.
- Paramètres de quantification : nous allons diminuer légèrement le QPP pour améliorer la qualité des trames de type P, en gardant les autres valeurs fixes. Le changement effectué à ce niveau, même si mineur, va permettre néanmoins une meilleure qualité de la vidéo, surtout que les trames de type P peuvent être aussi fréquentes que celles de type B.
- Séquence vidéo : format normal, trames de types B fréquentes. Puisque le passage à ce mode de fonctionnement se fait généralement après le mode *Normal1*, et n'est fait que si l'état du réseau est assez bon pour le permettre. Nous pouvons donc considérer que la qualité de l'image au récepteur est

bonne. Une mise à jour n'est donc pas nécessaire, sauf si cette mise à jour est prévue dans la forme normale de la séquence vidéo. Dans tous les cas, aucune modification n'est apportée à la forme de la séquence vidéo.

- Taille des paquets : STAP-A. Le RR reçu, informant d'un faible taux d'erreur dans le réseau, ainsi que d'un faible taux de perte. Nous n'avons donc pas besoin de changer le format des paquets utilisés dans ce mode de fonctionnement.

3.3.6 Mode Congestion Prevention

Le passage à ce mode est fait en cas de possibilité de congestion dans le réseau. Ce risque de congestion peut être détecté grâce au RR reçu de la destination. Si le rapport reçu informe d'une possibilité, ou d'un début de congestion dans le réseau, nous devons d'abord diminuer le débit sur le réseau, pour permettre aux mémoires tampons des nœuds de se vider. De plus comme le rapport nous informe d'une valeur croissante de la gigue dans le réseau, nous allons utiliser un format de paquets adéquats pour remédier à ce problème de gigue qui dégrade la qualité de la vidéo au récepteur.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1.1, nous gardons le même niveau que dans le mode *Normal1* et *Normal2*, car même s'il y a possibilité de congestion, nous allons essayer de diminuer le débit mais sans pour autant vider complètement les mémoires tampons des nœuds intermédiaires.
- Codage entropique : CAVLC, ce codage est plus léger que CABAC, donc le traitement de la vidéo à la source et à la destination est plus rapide. De plus, à ce niveau nous n'avons pas encore un taux élevé d'erreurs dans le réseau, donc l'utilisation de CABAC n'est toujours pas nécessaire, ni avantageuse.
- Paramètres de quantification : nous allons augmenter la valeur de QPI et QPP en gardant la valeur de QPB fixe. Les valeurs choisies pour QPI, QPP et QPB vont permettre de garder une qualité acceptable de la vidéo à la

sortie du codeur, tout en permettant de diminuer le débit sur le réseau. Ceci permettra de garder la qualité de la vidéo stable au récepteur, et en même temps d'alléger la charge du réseau.

- Séquence vidéo : format normal de la séquence vidéo, trames de type B fréquents. Puisqu'il y a possibilité de congestion sur le réseau, nous n'allons pas augmenter la fréquence des trames de type I ou P, même si cela va améliorer la qualité de la vidéo au décodeur, mais cela va aussi augmenter le débit à la source. Cette augmentation de débit va amplifier la congestion, et causer plus de pertes dans le réseau, et donc une dégradation de la qualité de la vidéo au décodeur.
- Taille des paquets : FU-A. Le RR reçu, informant d'une gigue augmentante dans le réseau, nous utiliserons des paquets de petite taille pour remédier à ce problème et repasser à une valeur acceptable de la gigue dans le réseau.

3.3.7 Mode Prevention

Le passage à ce mode est fait en cas de possibilité de coupure de liens dans le réseau. Ce risque de coupure peut être détecté grâce au RR reçu de la destination. Suite à la réception de ce rapport, nous allons diminuer le débit à la source. De cette façon, si la coupure se confirme, nous aurons déjà commencé la phase de prévention en allégeant les mémoires tampons des nœuds intermédiaires, et ainsi minimisant les pertes dues à cette coupure. De plus comme le rapport nous informe d'une valeur croissante du taux d'erreurs dans le réseau, nous utiliserons un format de paquets adéquats pour minimiser les erreurs.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1.1, nous gardons le même niveau que dans le mode *Normal1* et *Normal2*, car même si il y a possibilité de coupure de liens, nous essayerons de diminuer le débit mais sans pour autant vider complètement les mémoires tampon des nœuds intermédiaires.

- Codage entropique : CABAC. Ce codage est plus robuste que CAVLC contre les erreurs, et l'état actuel du réseau présente une grande probabilité d'avoir un grand taux d'erreurs. De plus, même si le codage et le décodage en CABAC sont plus lents qu'en CAVLC, cela ne dérange pas le fonctionnement de l'application à la source et à la destination, au contraire il le ralentit en donnant plus de temps au réseau de se reconfigurer correctement.
- Paramètres de quantification : nous conservons les mêmes valeurs pour QPI, QPP et QPB que pour le mode *Congestion Prevention*, car à ce niveau, la coupure n'est toujours pas certaine, donc nous allons essayer de fonctionner comme dans le cas de prévention de congestion.
- Séquence vidéo : trames de type P plus fréquentes. Même si cela va augmenter le débit sur le réseau, mais comme il y a possibilité de coupure, il y a donc grande probabilité de perte de paquets, ainsi que grand taux d'erreurs. Les trames de type P sont moins dépendants sur les trames précédentes pour être décodés. Leur utilisation dans ces conditions est donc très avantageuse pour remédier aux erreurs induites par le réseau.
- Taille des paquets : unite NAL unique, ce sont des paquets qui contiennent chacun un seul paquet NAL. Puisque nous prévoyons une coupure de lien sur le chemin, nous allons augmenter la taille des paquets [22]. Mais avec les coupures de liens le taux d'erreur augmente aussi, donc nous n'allons pas agrandir beaucoup la taille des paquets. Nous utiliserons une taille intermédiaire, entre les petites FU-A et les grandes STAP-A.

3.3.8 Mode Recovery

Le passage à ce mode se fait si la coupure de lien est confirmée par une absence de message *hello* ou bien par la réception d'un message *RERR*. Dans le cas de coupure certaine de lien, nous passerons à un mode de fonctionnement basique. Notre but dans cette phase se résume à ne pas faire déborder les mémoires tampons des nœuds

intermédiaires. Nous conserverons dans ces nœuds exclusivement l'information nécessaire au bon décodage de la vidéo. Jusqu'à la découverte d'un nouveau chemin, nous minimiserons le débit sur le réseau. A ce niveau, la qualité de la vidéo n'est pas plus importante que la quantité des paquets qui parviennent à la destination, donc nous diminuons considérablement la qualité de la vidéo. De plus nous utilisons le format de paquet adéquat à l'état de blocage total.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1b, dans ce mode de fonctionnement nous passons au fonctionnement basic à la source. Pour cette raison nous utilisons le niveau ayant le moins d'exigences en débit.
- Codage entropique : CABAC. Il est encore important pour nous d'utiliser un codage robuste. La qualité de la vidéo étant dégradée à la source, il est important que le minimum d'erreurs soit induit par le réseau, pour que la qualité à la destination reste acceptable. De plus, la lenteur de cette technique de codage reste un avantage et pas le contraire, pour ralentir le débit de paquets sur le réseau au niveau de la source, ainsi que le traitement des paquets à la réception, au niveau de la destination.
- Paramètres de quantification : nous allons augmenter encore plus les valeurs de QPI, QPP et QPB, pour sacrifier la qualité de la vidéo au profit d'un débit plus faible. A ce niveau la qualité de la vidéo au décodeur repose plus sur la quantité de paquets reçus que sur la qualité de ces derniers. De plus nous désirons minimiser le débit à la source jusqu'à la découverte d'un nouveau chemin. nous donnons donc de grandes valeurs aux QPs des trois types de trames.
- Séquence vidéo : trames de type I et P plus fréquentes. Puisque ces trames ne sont plus très grandes, et puisqu'une grande partie des paquets ne parviendront pas à la destination, nous allons faire en sorte que ceux qui y

parviennent soit décodable en ayant recours au minimum de paquets précédents, qui peuvent ne pas être reçus.

- Taille des paquets : FU-A. L'état du réseau ne permet pas l'utilisation de grands paquets. Le taux d'erreurs est d'abord très grand, de plus, l'état des nœuds est similaire à celui en cas de congestion. Il est donc plus intéressant d'utiliser des paquets de petite taille dans ce mode de fonctionnement.

3.3.9 Mode Congestion Recovery

Si, étant dans le mode *Congestion Prevention* nous recevons un autre RR avec une grande valeur du champ *Interarrival Jitter*, nous considérons que la congestion est survenue dans le réseau, et que ce n'est plus juste une possibilité. Nous devons donc prendre les mesures nécessaires pour alléger au maximum les mémoires tampons des nœuds intermédiaires sur le réseau. Comme dans le cas de coupure de lien, nous allons accorder plus d'importance à la quantité des paquets qui parviennent à la destination, qu'à la qualité de la vidéo codée à la source. Ce qui implique, comme dans les modes précédents, que nous devons minimiser le débit sur le réseau, et ceci en diminuant la qualité de la vidéo codée. De plus nous utiliserons un format de paquets adéquat à l'état de congestion dans lequel se trouve le réseau.

Les paramètres à la source qui définissent ce mode de fonctionnement sont les suivants :

- Profile : High
- Niveau : 1. Ce mode de fonctionnement est utilisé quand le réseau est dans un état plus désastreux que pour le mode *Congestion Prevention*, mais moins désastreux que pour le mode *Recovery*. Donc le niveau utilisé pour ce mode est un niveau intermédiaire entre les niveaux utilisés dans les deux cas précédemment cités.
- Codage entropique : CABAC. Comme dans le mode *Recovery* nous devons utiliser un codage robuste à la source. Premièrement pour compenser la dégradation de la qualité de la vidéo au niveau de la source. De plus parce que le taux d'erreurs dans le réseau est fort quand ce dernier est dans un état

de congestion. D'autre part la lenteur de cette méthode de codage reste un avantage pour nous dans ce mode de fonctionnement.

- Paramètres de quantification : les valeurs de QPI et QPP sont plus petite que celles utilisées dans le mode *Recovery*, mais plus grandes que celles utilisées dans le mode *CP*. Nous allons améliorer un peu la qualité de la vidéo codée au niveau de la source par rapport à sa qualité dans le mode *Recovery*. En effet, dans ce mode il y a congestion mais le réseau n'est pas complètement bloqué comme dans le cas de coupure de lien. Nous pouvons donc nous permettre d'améliorer un peu la qualité de la vidéo au prix d'une légère augmentation du débit à la source.
- Séquence vidéo : trames de type I et P plus fréquents. Puisque ces trames ne sont plus très grandes, et puisqu'une grande partie des paquets ne parviendront pas à la destination, nous faisons en sorte que ceux qui y parviennent soient décodables en ayant recours au minimum d'images de référence, qui peuvent ne pas être reçus.
- Taille des paquets : FU-A. L'état du réseau ne permet pas l'utilisation de grands paquets. Le taux d'erreurs est d'abord très grand, de plus le réseau est congestionné. Il est donc plus intéressant d'utiliser des paquets de petite taille dans ce mode de fonctionnement.

3.4 Architecture cross-layer

L'architecture OSI ou TCP/IP standard est très bien adaptée aux réseaux câblés. Mais cette architecture présente des limitations dans les réseaux ad hoc. En effet ces réseaux n'ont pas les mêmes propriétés que les réseaux câblés. La différence majeure entre ces deux types de réseau est que, inversement aux réseaux câblés, la topologie dans les réseaux ad hoc varie dynamiquement et aléatoirement, et l'état du réseau change en conséquence continuellement. D'où la nécessité d'informer les différents niveaux de la pile de l'état courant du réseau. Cet échange d'information entre les couches est interdit dans l'architecture normale des standards OSI et TCP/IP. Nous nous trouvons donc

obligés de violer cette architecture pour améliorer son rendement dans les réseaux ad hoc.

L'application que nous avons définie dans ce chapitre est sensée s'adapter dynamiquement à l'état du réseau. Elle doit donc connaître à tout moment l'état dans lequel se trouve le réseau. Cette information sur l'état du réseau est collectée par les couches réseau et transport. D'où la nécessité de définir une architecture qui permet la communication temps réel entre ces trois couches.

3.4.1 Architecture cross-layer choisie

Le but de notre recherche n'est pas de développer une architecture cross-layer, mais de développer une application adaptative à l'état des réseaux ad hoc. Pour cette raison nous utiliserons une architecture déjà définie dans la littérature. De plus nous choisirons une architecture rapide à implémenter, et facile à configurer. Dans [17] on trouve un état de l'art sur les architectures cross-layer existants dans la littérature (voir partie 2.5). Des architectures exposées dans ce document, nous trouvons que la mieux adaptée à notre application est l'architecture calibrage verticale. Cette architecture permet le passage d'un paramètre à travers les couches jusqu'à l'arrivée à la couche cible. Pour ce faire, nous allons créer une couche supplémentaire, qui est en fait une base de données, que nous appellerons niveau *infoXchange*, voir figure 3.8. Cette base de données contiendra de l'information insérée par les couches réseaux, transport et application. Ces mêmes couches pourront aussi lire l'information insérée dans cette base de données par les autres couches. Donc cette base de données aura une interface de lecture/écriture avec les trois couches mentionnées plus haut.

3.4.2 Paramètres échangés entre les couches

Trois couches vont s'échanger de l'information suivant notre architecture. Cette information échangée concerne l'état du réseau, donc l'état des liens, le nombre de sauts, ainsi que la qualité offerte par ce réseau. De même il y aura un échange d'information concernant l'application, et plus précisément, le type de paquets NAL envoyés sur le réseau.

Ceci dit il nous est facile de définir les paramètres à insérer dans la couche *infoXchange*.

Au niveau application :

- *nal_unit_type* : ce paramètre contient le type d'unité NAL envoyé sur le réseau [9]. Cette information sera utile pour savoir quel type de paquet utiliser à la couche transport (voir partie 3.3).

Au niveau transport :

- *numéro de séquence du dernier paquet* : ce paramètre contient le numéro de séquence du dernier paquet reçu à la destination. Cette information sera utile au niveau de l'application pour savoir le nombre de paquets à retransmettre.
- *Gigue inter-arrivé* : ce paramètre contient la valeur de la gigue sur le réseau entre la source et la destination. Cette information sera utilisée au niveau de l'application pour prendre une décision sur le mode de fonctionnement devant être utilisé.
- *Temps d'aller retour* : ce paramètre contient le temps d'aller retour entre la source et la destination. Cette information sera utilisée par l'application pour la durée du fonctionnement dans quelques modes (voir partie 3.3.3).
- *Fraction perdue* : ce paramètre contient la fraction de paquets perdus dans le réseau. Cette information sera utilisée au niveau de l'application pour prendre une décision sur le mode de fonctionnement devant être utilisé.

Au niveau réseau :

- *Valeur de N* : ce paramètre contient la valeur du champ N dans le paquet d'erreur reçu. Cette information sera utilisée au niveau de l'application pour prendre une décision sur le mode de fonctionnement devant être utilisé.
- *Valeur de lifetime* : ce paramètre contient la durée de temps durant laquelle le chemin trouvé reste fonctionnel. Cette information sera utilisée au niveau de l'application pour prendre une décision sur le mode de fonctionnement devant être utilisé (voir partie 3.2.1).

- Etat du chemin : ce paramètre contient l'état du chemin établi entre la source et la destination. Cette information sera utilisée au niveau de l'application pour prendre une décision sur le mode de fonctionnement devant être utilisé.
- Nombre de saut : ce paramètre contient le nombre de saut entre la source et la destination. Cette information permet de prendre une décision sur la taille des paquets à utiliser niveau transport.

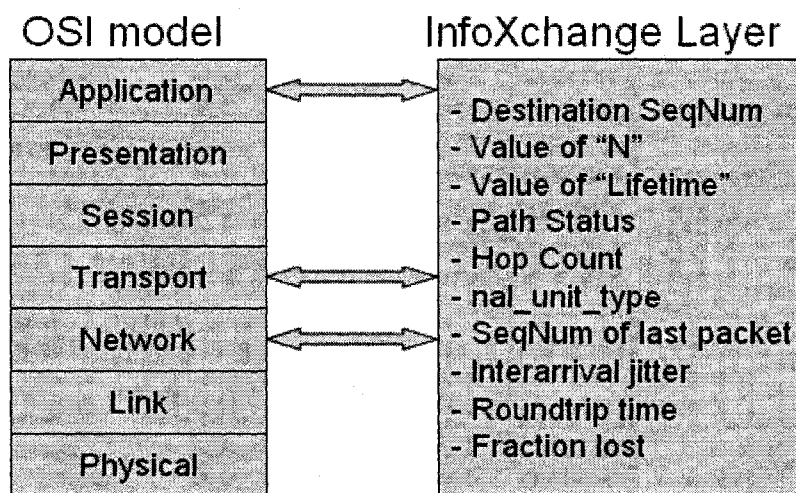


Figure 3.8 Architecture cross-layer choisie

Les états que nous avons cités dans cette partie ne sont pas les seuls qui peuvent exister. Nous pouvons définir une infinité d'états en changeant à chaque fois un seul paramètre. Bien entendu plus nous avons d'états et plus le résultat sera satisfaisant coté qualité de la vidéo à la réception. Mais un grand nombre de mode de fonctionnement rendra l'application difficile à gérer. Notre étude empirique c'est basée sur 15 modes de fonctionnement que nous avons défini à la première étape de notre recherche. Mais les résultats obtenus ne montraient pas une différence très significative entre les états intermédiaires. Pour cette raison et dans le but de rendre l'application plus facilement

gérable, nous avons décidé de retenir les sept modes de fonctionnements que nous avons définis dans ce chapitre.

3.5 Méthode de signalisation

Après la définition des modes de fonctionnements de la source, et des valeurs des différents paramètres du codeur à la source, nous pouvons apporter des modifications aux informations concernant les paramètres de séquence et les paramètres d'image envoyés à la destination. En effet, si les modes de fonctionnements sont bien définis et connus par la source et la destination, les paquets concernant les paramètres de séquence et d'image ne contiendront plus que le numéro du mode dans lequel fonctionne actuellement la source. Cela va permettre de diminuer considérablement la taille des paquets utilisés pour communiquer les valeurs de ces paramètres entre la source et la destination, voir même les éliminer complètement.

On propose donc d'associer un numéro d'identification à chaque mode de fonctionnement, et d'inclure ce numéro directement dans l'entête NAL du premier paquet de la séquence, ou de l'image si un changement est effectué au niveau de l'image.

Ainsi dans ce chapitre nous avons énuméré les différents états dans lesquels un réseau ad hoc peut passer durant une communication. De même, nous avons cité les différents modes de fonctionnement dans lesquels une application de vidéoconférence devra fonctionner, suivant l'état du réseau, figure 3.9. Pour chaque mode de fonctionnement possible, nous avons fixé les valeurs des paramètres qui le définissent à la source. Finalement nous avons proposé une méthode pour optimiser l'utilisation de ces modes pour diminuer le débit associé à la signalisation entre la source et la destination.

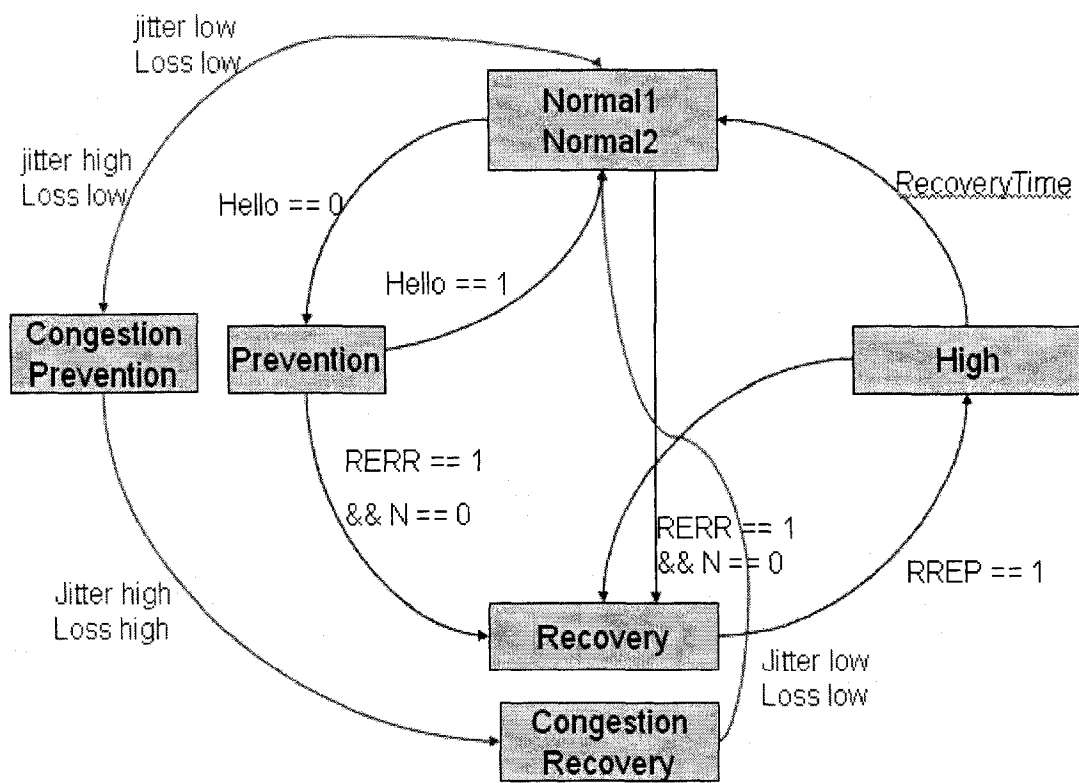


Figure 3.9 Fonctionnement de l'application

CHAPITRE 4 IMPLÉMENTATION ET ÉVALUATION DES PERFORMANCES

Dans le chapitre 3, nous avons énuméré les différents états dans lesquels le réseau ad hoc peut se trouver à un instant donné. De même nous avons présenté le mode de fonctionnement de l'application dans chaque état du réseau. De plus nous avons donné la méthode avec laquelle le passage d'un mode de fonctionnement à l'autre doit se faire. Dans ce chapitre nous allons simuler la solution proposée, puis évaluer ces performances. Pour cela nous présenterons tous d'abord l'environnement de simulation. Nous continuerons par le choix du simulateur le plus adapté à notre application. Finalement nous présenterons les résultats des différentes simulations représentant chaque mode de fonctionnement de l'application dans différents états du réseau, que nous comparerons enfin aux résultats obtenus en combinant les différents modes de fonctionnement dans une application adaptative.

4.1 Environnement de simulation

Dans cette partie nous allons présenter l'environnement utilisé pour l'implémentation et la simulation de notre solution. Nous commencerons par une présentation de l'équipement sur lequel nous travaillons, nous présenterons ensuite le simulateur utilisé pour implémenter notre solution, ainsi que les différents algorithmes auxquels nous avons fait recours pour simuler l'environnement d'application de notre solution.

4.1.1 Choix du simulateur

Dans notre recherche pour un simulateur pour tester notre solution, nous nous sommes fixés des critères. L'utilisation du simulateur doit être répandue dans la communauté de recherche, il doit être bien adapté pour simuler des réseaux ad hoc, il doit permettre le contrôle de mobilité et de congestion, et doit être facile à implémenter tout en étant assez flexible dans la représentation des fonctionnalités importantes dans les réseaux sans fil, et surtout les réseaux ad hoc.

Dans [23] on trouve une analyse de différents simulateurs pour les réseaux ad hoc. Même si cette analyse est faite pour retrouver le meilleur simulateur pour l'implémentation d'une application différente de la notre. Mais les forces et les faiblesses de quatre simulateurs, OPNET, GloMoSim/QualNet, NS-2 et OMNeT++, sont étudiées dans ce document. Les caractéristiques de NS-2 [24] présentées dans ce document, en font un candidat très intéressant pour être notre simulateur. En effet, NS-2 est très utilisé dans le domaine de la recherche que ce soit dans les réseaux câblés, que les réseaux sans fil, et spécialement les réseaux ad hoc. De plus ce simulateur est complet du côté des outils qui permettent de construire un réseau, représentant le plus fidèlement possible les réseaux réels. Finalement ce simulateur est gratuit dans sa version la plus complète, ce qui le rend avantageux par rapport aux autres simulateurs qui peuvent être très dispendieux. D'autre part, les limitations de NS-2 exposés dans [23] ne s'appliquent pas à notre application. En effet, nous allons simuler un réseau de 30 nœuds ou moins, donc l'extensibilité dans NS-2 ne pose pas de problèmes pour nous, puisque NS-2 est très bien adapté aux petits réseaux formés d'une centaine de nœuds. De plus nous n'avons pas besoin d'apporter des modifications majeures aux protocoles existants dans NS-2, et la seule programmation que nous avons à faire c'est le programme de base pour former le réseau et définir le mouvement des nœuds et le trafic TCP et vidéo dans ces derniers, donc la complexité de programmation ne présente pas non plus une limitation à considérer dans NS-2.

En conclusion, et en raison de tous les avantages cités plus hauts, nous trouvons que NS-2 est bien le simulateur le plus adapté à notre application, et donc ce sera le logiciel que nous utiliserons pour nos simulations.

4.1.2 Générateur de trafic vidéo

NS-2 offre un outil de génération de trafic, mais pour notre application nous avons besoin d'injecter dans le réseau un trafic provenant d'un codeur vidéo suivant le standard H.264/AVC. Comme les outils de NS-2 ne permettent pas automatiquement la création d'un tel trafic, nous avons eu recours à un outil disponible sur internet. EvalVid

[26][27][28] est un outil développé pour permettre l'évaluation de la qualité de la vidéo, suite à une transmission sur un réseau. Dans sa version modifiée, myEvalVid [29] permet aussi de modifier le format des paquets UDP dans NS2 pour inclure l'information sur le type de trames transmis. L'utilisation de cet outil nécessite l'utilisation de logiciels appropriés à la source et à la destination :

Les logiciels à la source sont :

- logiciel de codage et de compression de la vidéo : dans notre cas nous voulons utiliser le standard H.264/AVC donc nous avons le choix entre le logiciel de référence H.264/AVC, ou le codeur x264. Nous avons décidé d'utiliser x264 puisqu'il est plus rapide et moins encombrant que le logiciel de référence H.264/AVC.
- Logiciel de création d'un fichier de type ISO MP4 : ce logiciel va permettre de créer un fichier de type ISO MP4 contenant les trames codées de la séquence vidéo, en plus d'une allusion sur le type de paquets RTP à utiliser pour chaque trame. Le logiciel propose par Evalvid et que nous allons utiliser est le MP4Box.
- Le logiciel de trace : ce logiciel permet de transmettre le fichier MP4 obtenu précédemment, en utilisant les protocoles RTP sur UDP, vers une certaine destination. Cette transmission génère un fichier qui va être utile pour la simulation sur NS2. Le logiciel inclu dans EvalVid est le logiciel mp4trace, c'est ce logiciel que nous allons utiliser dans notre simulation.
- Le logiciel de simulation : c'est le logiciel que nous utilisons pour la simulation du réseau. Dans notre cas ce logiciel est NS2.

Les logiciels à la destination :

- le logiciel de reconstruction de la vidéo : ce logiciel va permettre de reconstruire la vidéo, à travers les paquets reçus, ainsi que le fichier MP4 généré à la source. Le logiciel utilisé est etmp4.

- Le logiciel de décodage de la vidéo : pour faire la comparaison entre la vidéo reçue et la vidéo transmise, nous devons décoder la vidéo reçue. Cette vidéo est en format MP4, et nous devons la transformer en format de base YUV. Pour cela nous utilisons le logiciel proposé dans EvalVid, ffmpeg.
- Logiciel de calcul de la qualité de service : ce logiciel permet de faire la comparaison entre la vidéo reçue et la vidéo transmise, ainsi que le calcul de la PSNR (Peak Signal-to-Noise Ratio) de chaque trame reçue, puis la moyenne de la PSNR de toutes les trames reçue.

Donc à l'aide de EvalVid, nous parvenons à créer un générateur de trafic vidéo suivant le standard H.264. De plus, en ajoutant les modifications proposées dans myEvalVid [29] à NS2, nous parvenons à changer le format des paquets UDP pour inclure le type de trame transporté par ce paquet. Cet outil sera donc utilisé dans notre simulation comme générateur de trafic.

4.2 Implémentation de la solution

Dans la partie précédente nous avons exposé en détails les différents équipements, logiciels et outils que nous utiliserons pour l'implémentation de notre application et la simulation de la solution que nous proposons. Cette partie sera consacrée aux détails de l'implémentation de notre application dans l'environnement précédemment cité, en utilisant les outils présentés dans la partie précédente. D'abord nous exposerons la stratégie d'évaluation de notre solution, ensuite nous détaillerons les différentes étapes suivies dans son implémentation.

4.2.1 Stratégie d'évaluation

Notre but est d'évaluer la qualité de la vidéo au récepteur, suite à une transmission sur un réseau ad hoc. La vidéo transmise est codée suivant le standard H.264/AVC, et le réseau ad hoc passe par plusieurs états durant la transmission. Dans son fonctionnement normal, le codage de la vidéo est indépendant de l'état du réseau, donc le codeur ne s'adapte pas dynamiquement à l'état de ce dernier. Nous commencerons donc par

évaluer chaque mode de fonctionnement, séparément, d'une façon non adaptative, puis nous évaluerons la solution proposée quand le réseau passe par plusieurs états.

Pour cela nous commencerons par coder la même séquence vidéo plusieurs fois. A chaque fois nous changerons les paramètres au codeur suivant ceux définis pour chaque mode de fonctionnement. Ainsi nous obtenons sept fichiers vidéo, qui représentent la vidéo codée suivant les sept modes de fonctionnements définis pour le codeur. Ensuite, nous créerons différents scenarios de simulations sur NS2, représentant chacun un état possible du réseau ad hoc. Puis nous utiliserons EvalVid, pour transmettre les sept fichiers vidéo précédemment obtenus sur le réseau ad hoc simulé dans ses différents états possibles. A la fin de la simulation nous évaluerons la qualité de la vidéo obtenue pour les différents modes de fonctionnements dans les différents états du réseau. Finalement nous simulerons notre solution dans un réseau qui passe successivement dans les différents états possibles du réseau, et nous comparerons les résultats obtenus à ceux des simulations précédentes.

4.2.2 Etapes de l'implémentation de la solution

Comme nous l'avons mentionné précédemment, nous allons implémenter notre solution en utilisant le simulateur NS2, avec l'outil d'évaluation de la vidéo EvalVid. Nous allons suivre les étapes décrites dans [29]. Dans cette partie nous donnerons les détails de chaque étape de l'implémentation de la solution. Nous commencerons par les étapes qui simulent la source de la vidéo. Commenant par le codage de la vidéo, ensuite la construction du fichier de vidéo format MP4 et du fichier de trace obtenu suite à la transmission de la vidéo. Après cela nous passerons à la simulation du réseau, où nous présenterons et expliquerons le code source en OTcl de la simulation sur NS2. Pour finir les étapes qui simulent la destination de la vidéo. Incluant la reconstruction de la vidéo, son décodage et le calcul de la qualité dans laquelle elle est reçue.

4.2.2.1 Codage de la vidéo

Pour coder la vidéo nous avons utilisée le codeur x264 [30]. Ce codeur permet de coder une vidéo de format YUV (format de base de la video), suivant le standard H.264/AVC.

Il est très fidèle au standard qu'il suit, et offre la possibilité à l'utilisateur de choisir les paramètres de codage qu'il désire utiliser. Cet outil inclut beaucoup des paramètres de H.264/AVC, mais dans cette partie nous présenterons seulement ceux qui sont intéressants pour notre application.

Les paramètres d'entrée sortie :

- *fps* : ce paramètre permet de déterminer le taux de transmission en trames par seconde.
- *frames* : ce paramètre permet de spécifier le nombre total de trame à être codées.
- *output* : ce paramètre permet de spécifier le nom du fichier dans lequel on désire sauvegarder le résultat du codage de la vidéo.
- *no-psnr* : ce paramètre permet d'arrêter la computation du PSNR.
- *level* : ce paramètre permet de spécifier le niveau qu'on désire utiliser pour le codage de la vidéo, comme décrit dans l'annexe A de [9].

Les paramètres des trames :

- *keyint* : ce paramètre permet de spécifier la taille maximale de la GOP (Groupe Of Picture), qui est l'intervalle en trames entre deux IDR (Instantaneous Decoding Refresh) consécutifs.
- *bframes* : ce paramètre permet de spécifier le nombre de trames de type B entre les trames de types I et ceux de type P.
- *no-cabac* : ce paramètre permet d'arrêter l'utilisation de CABAC comme méthode de codage entropique, dans ce cas CAVLC est utilisée à la place.
- *ref* : ce paramètre permet de spécifier le nombre de trames de référence.

Les paramètres de contrôle de débit :

- *qp* : ce paramètre permet de spécifier le QP utilisé pour le codage des trames de types P.

- *ipratio* : ce paramètre permet de spécifier le rapport entre le QP utilisé pour le codage des trames de type I et celui utilisé pour le codage des trames de type P. Ce paramètre permet donc indirectement de spécifier le QP utilisé pour le codage des trames de type I.
- *pbratio* : ce paramètre à la même fonction que *ipratio* mais pour les trames de type B.

Ces différents paramètres vont nous permettre de définir les différents modes de fonctionnement de l'application à la source. Pour cela leur valeur n'est pas fixe et la commande pour le codage de la vidéo en utilisant cet outil varie d'un mode de fonctionnement à l'autre. Pour cette raison nous avons écrit de petits scripts dans des fichiers de format *Batch*, chaque scripte contenant la commande à exécuter pour simuler un mode de fonctionnement.

4.2.2.2 Création de fichier MP4

Suite au codage de la vidéo, nous passons à l'étape suivante, comme décrit dans [29]. Cette étape consiste à transformer le fichier vidéo de format H.264 en un fichier vidéo de format MP4. Pour cela nous utilisons le logiciel MP4Box, qui va créer le fichier MP4 tout en ajoutant une allusion au format de paquet RTP à utiliser pour la trame. Cet outil offre plusieurs options, celles qui nous intéressent sont les suivantes:

- *hint* : cette option permet d'ajouter une allusion (hint en anglais) au fichier, pour spécifier le type de paquets RTP à utiliser pour chaque trame.
- *mtu* : cette option permet de spécifier la taille maximale de l'unité de transmission (MTU).
- *fps* : cette option permet de forcer la fréquence d'import de la vidéo et des sous-titrages, en trames par seconde.

Les valeurs de ces paramètres, et par suite la commande globale ne changent pas d'un mode de fonctionnement à l'autre. En effet, les modifications apportées aux formats des

paquets RTP utilisés dans les différents modes de fonctionnement, ne seront pas définis à ce niveau, mais directement dans les paramètres de la simulation dans NS2.

A la fin de cette étape, nous aurons des fichiers de format MPEG-4, contenant chacun la version MPEG-4 de la séquence vidéo codée suivant un mode de fonctionnement du codeur H.264.

4.2.2.3 Envoie de la vidéo à une destination

Cette étape consiste à envoyer la vidéo de format MPEG-4 reçue précédemment, vers une certaine destination spécifiée dans la commande. Pour cela nous utilisons l'outil mp4trace. Cet outil va permettre d'envoyer la vidéo suivant les allusions ajoutées aux fichiers MP4 par l'outil MP4Box. En sortie, cet outil va créer un fichier qui contient la trace de l'envoi, c'est ce fichier de trace qui sera utilisé pour simuler le trafic en entrée du simulateur NS2. Les options offerts par cet outil ne sont pas nombreuses, celles que nous allons utiliser sont :

- *f/p* : cette option permet de définir le type des trames envoyées, donc si c'est une trame ou une image complète.
- *s* : cette option permet de spécifier l'adresse et le port de la destination de la vidéo.

Cet outil ne présente pas beaucoup de flexibilité et par suite les seules options qu'il présente ne nous intéressent pas pour différencier les différents modes de fonctionnement du codeur. Pour cela les mêmes options seront utilisées pour tous les modes de fonctionnement, de même tout ce qui nous intéresse c'est le fichier de trace que nous obtenons en sortie.

Ainsi suite à cette étape, nous obtenons des fichiers de traces qui représentent le trafic vidéo à injecter dans la simulation NS2. Chaque fichier de trace représente le trafic suivant un certain mode de fonctionnement, donc chaque fichier sera la simulation d'une source qui fonctionne suivant l'un des modes définis dans la partie 3.3.

4.2.2.4 Simulation du réseau ad hoc

Ainsi et après avoir créé le fichier qui va simuler le trafic à la source, nous parvenons à la partie de simulation du réseau. Pour la simulation de notre réseau ad hoc nous avons décidé d'utiliser le simulateur NS2. Dans cette partie nous présentons le programme que nous avons écrit pour la simulation du réseau. Nous définirons les caractéristiques de notre réseau simulé, puis nous expliquerons le fonctionnement global de notre programme.

Les détails et les paramètres de notre programme sont donnés dans ce qui suit.

Paramètres du réseau :

- Type de canal : nous simulons un réseau sans-fil MANET, donc nous utilisons un canal sans-fil.
- Type de propagation : nous utilisons un type de propagation qui permet de modéliser le plus réellement possible la propagation entre les nœuds. Pour cela, nous avons décidé d'utiliser la propagation directe avec réflexion au sol (two-ray ground en anglais). Ce modèle de propagation est réaliste dans le cas où la distance entre les nœuds est grande.
- Protocole de routage : dans l'étude théorique faite précédemment, nous avons fixé l'hypothèse que le protocole de routage utilisé dans le réseau est le protocole AODV. Dans notre simulation nous utilisons donc ce protocole pour le routage dans le réseau.
- Nombre de nœuds dans le réseau : notre réseau est constitué d'un nombre fixe de nœuds mobile, 30 dans notre cas.
- Espace de mouvement : l'espace dans lequel évoluent les nœuds de notre réseau est de taille variable. En effet nous allons utiliser plusieurs dimensions pour simuler les différents états du réseau. Ainsi en augmentant les dimensions de l'espace, nous simulerons un réseau avec une plus grande probabilité de coupure de liens. Les dimensions de notre réseau prennent les valeurs (950 x 950) m², (1000 x 1000) m² et (1100 x 1100) m², pour une faible, normale et grande probabilité de coupure de liens respectivement.

- Taille de paquets : les paquets transmis dans notre réseau sont de taille variable. En effet comme expliqué dans l'étude théorique, voir partie 3.3, la taille des paquets vidéo varie suivant le mode de fonctionnement de la source. ainsi la taille des paquets prend les valeurs 750 octets, 1024 octets et 1350 octets, suivant le mode de fonctionnement simulé.

Paramètres des nœuds :

- Interface physique : l'interface physique installée sur les nœuds du réseau, est une interface sans-fil puisque les nœuds font partie d'un réseau MANET.
- Antennes : tous nos nœuds sont équipés d'une antenne omnidirectionnelle située au centre du nœud et à une hauteur de 1.5 m, puisque nous considérons que les nœuds mobiles sont des équipements portés par des personnes ou dans des véhicules.
- Niveau MAC : la technologie sans-fil utilisée au niveau MAC de nos nœuds mobiles, est la technologie ieee 802.11.
- Niveau liaison : au niveau liaison (Link Layer) nous utilisons le protocole LL propre à ce niveau.
- File d'attente : la file d'attente dans les nœuds mobiles est de type rejet de queue (DropTail en anglais), avec priorité. Ce type de file d'attente va rejeter les paquets qui arrivent en dernier, tout en allouant une priorité particulière aux paquets de routage. Nous pouvons modifier le fonctionnement de cette file d'attente pour allouer une priorité aux paquets contenant des trames de type I et P par rapport à ceux contenant des trames de type B.
- Taille de la file d'attente : la taille de la file d'attente en paquets varie selon la taille des paquets utilisée. Ainsi la taille est de 69, 50 et 38 paquets, suivant que la taille des paquets est de 750, 1024 ou 1350 octets respectivement.

Paramètres de mouvement :

Le mouvement des nœuds dans le réseau est aléatoire. Pour générer ce mouvement aléatoire, nous utilisons un générateur de mouvement aléatoire offert par NS-2, *setdest*. Cet outil permet de générer un fichier qui contient les caractéristiques du mouvement de chaque nœud dans le réseau. Malgré que cet outil permet de générer un mouvement aléatoire, il offre des options dans la commande qui permettent à l'utilisateur de spécifier les paramètres qu'il désire utiliser pour le mouvement des nœuds. Parmi ces paramètres, ceux qui nous intéressent sont les suivants :

- nombre de nœuds, à l'aide de l'option *-n* : cette option permet de spécifier le nombre de nœuds dans le réseau, pour lesquels nous voulons générer un fichier de mouvement aléatoire. Notre réseau est formé de 30 nœuds, donc la valeur de cette option dans notre commande est 30.
- Le temps de pause entre mouvement, à l'aide de l'option *-p* : cette option permet de spécifier le temps de pause entre deux mouvements consécutifs d'un nœud. Dans notre commande, nous allons donner la valeur 2.0 secondes.
- La vitesse maximale, à l'aide de l'option *-M* : cette option permet de spécifier la vitesse maximale de déplacement des nœuds mobiles. La valeur de ce paramètre n'est pas fixe, mais elle varie selon l'état simulé du réseau. Ainsi dans un réseau à forte probabilité de coupure de liens, les nœuds mobiles se déplacent rapidement, soit à une vitesse maximale de 15 m/s, alors que dans un réseau à faible probabilité de coupure de liens, la vitesse maximale de déplacement des nœuds est de 10 m/s. ces valeurs sont choisies en considérant que les nœuds mobiles, sont des équipements portés par de piétons, ou embarqués dans des véhicules.
- La durée de la simulation, à l'aide de l'option *-t* : cette option permet de spécifier la durée de la simulation. Dans notre cas, la séquence vidéo utilisée est de 10 secondes, donc nous désirons arrêter la simulation après 15 secondes. Pour cela nous allons donner la valeur 15 à ce paramètre.

- Dimensions, à l'aide des options *-x* et *-y* : ces deux options permettent de spécifier les dimensions en longueur et en largeur, respectivement de l'espace dans lequel évoluent les nœuds mobiles. Comme nous avons dit précédemment, les dimensions de notre espace varient suivant l'état simulé du réseau, ainsi ces options prennent les valeurs 950, 1000 et 1100.

Paramètres de trafic :

Le trafic injecté dans réseau ad hoc simulé est différent, suivant les états du réseau simulé. Ainsi, quand nous testons le comportement de notre application dans des cas de coupure de liens, nous ne nous intéressons pas aux congestions dans le réseau. Pour cela, dans ce cas nous allons injecter seulement le trafic vidéo, en utilisant le fichier de trace créé précédemment à l'aide de l'outil *mp4trace*. Alors que quand nous testons le comportement de notre application dans des cas de congestion dans le réseau, nous injectons en plus du trafic vidéo, du trafic additionnel pour causer des congestions. Pour créer le trafic additionnel, nous allons utiliser un outil offert par NS-2, *cbrgen*. Cet outil permet de générer un fichier qui contient les caractéristiques du trafic de chaque nœud dans le réseau. Malgré que cet outil permet de générer du trafic aléatoire, il offre des options, dans la commande, qui permettent à l'utilisateur de spécifier les paramètres qu'il désire utiliser pour le trafic généré. Parmi ces paramètres, ceux qui nous intéressent sont les suivants :

- Le type de trafic, à l'aide de l'option *-type* : cette option permet de spécifier le type de trafic que nous voulons générer. Dans notre cas nous générons le trafic simplement pour créer de la congestion dans le réseau, pour cela nous allons générer du trafic constant CBR (Constant Bit Rate), puisqu'il est le plus facile à gérer.
- Le nombre de nœuds, à l'aide de l'option *-nn* : cette option permet de spécifier le nombre de nœuds dans le réseau. Notre réseau est formé de 30 nœuds, donc la valeur de cette option est 30 dans notre cas.

- La valeur de départ, à l'aide de l'option *-seed* : cette option permet de spécifier la valeur de départ de notre trafic.
- Le nombre maximal de connexions, à l'aide de l'option *-mc* : cette option permet de spécifier le nombre maximal de connexions simultanées dans le réseau. La valeur de cette option variera suivant le taux de congestion que nous voulons avoir dans le réseau. Ainsi le nombre de connexions va passer de 5 à 12 pour augmenter graduellement le taux de congestion dans le réseau.
- Le débit, à l'aide de l'option *-rate* : cette option permet de spécifier le débit en paquets par seconde de chaque communication entre deux nœuds. Nous allons garder ce débit fixe à 100 paquets par seconde, et faire varier le nombre de connexions pour faire varier le taux de congestion. De cette façon la congestion sera aléatoire dans le réseau, et ainsi le réseau plus proche d'un réseau réel.

A la sortie du simulateur, nous voulons avoir des fichiers contenant les paquets vidéo transmis par la source, et ceux reçus à la destination, et cela pour pouvoir reconstituer la séquence vidéo comme elle est reçue. Pour cela, nous attachons un fichier de trace à l'agent du nœud source, et un autre à celui du nœud destination. Ces deux fichiers seront utilisés par la suite pour reconstruire la séquence vidéo reçue à la destination.

4.2.2.5 Reconstruction de la vidéo reçue

Cette étape de la simulation consiste à reconstruire la séquence vidéo transmise, tel que reçue à la destination. Pour cela nous utilisons l'outil *etmp4*. En entrée de *etmp4*, nous donnons les fichiers de trace à la source et à la destination, en plus du fichier de trace obtenu en sortie de l'outil *mp4trace*, ainsi que le fichier contenant la vidéo format mp4. A la sortie de cet outil, nous obtenons un fichier contenant la séquence vidéo telle que reçue à la destination.

4.2.2.6 Décodage de la vidéo

Après avoir reconstruit la vidéo reçue à la destination, nous devons la décoder pour l'avoir en format YUV. Pour le décodage nous utilisons l'outil *ffmpeg*. Cet outil va permettre de transformer la séquence vidéo reçue, qui est en format H.264/AVC, en une séquence vidéo YUV que nous pouvons comparer à la séquence YUV initiale à la source. Cet outil prend en entrée la séquence vidéo reçue, et donne en sortie la séquence vidéo format YUV.

4.2.2.7 Calcul du PSNR

La dernière étape de notre simulation consiste à calculer le PSNR de la vidéo reçue par rapport à la vidéo envoyée. Pour cela nous utilisons l'outil *psnr*. Cet outil va permettre de comparer la vidéo reçue à la vidéo initiale, envoyée, et de calculer ainsi le PSNR de la vidéo reçue. *Psnr* prend en entrée le format de la vidéo, sous forme de taille de trame en échantillons horizontales et verticales, ainsi que la vidéo initiale à la source et la vidéo reçue à la destination, toutes les deux format YUV. En sortie, cet outil nous donne le PSNR de chaque trame reçue, et le PSNR moyen de toutes les trames reçues. C'est cette dernière valeur qui sera utilisée pour évaluer la qualité globale de la vidéo à la réception.

Ainsi et suite à cette dernière étape, nous aurons simulé la source, le réseau et la destination, et nous aurons obtenus à la fin la qualité de la vidéo en PSNR. Mais cette méthode présente une limitation. Le décodeur utilisé à la destination, n'est pas un décodeur fidèle au modèle de référence H.264/AVC, et donc ne présente pas toutes les fonctionnalités possible d'un décodeur réel. Ainsi toutes les méthodes de correction d'erreurs qu'un tel décodeur peut offrir ne sont pas testées par cette méthode. Cela n'est pas très significatif pour notre application, puisque les modes de fonctionnement définis à la source ne dépendent pas du décodeur. En effet nous n'utilisons aucune méthode de codage qui se base sur des outils ou des fonctionnalités offerts par le décodeur. Par suite, les outils utilisés dans notre simulation, sont largement suffisantes pour simuler notre application. Mais si dans des travaux futurs nous désirons améliorer encore plus la

performance de notre application, et cela en utilisant les fonctionnalités offertes par les décodeurs, nous devons utiliser un décodeur plus fidèle au modèle de référence.

4.3 Résultats de simulation

Dans cette section, nous présenterons les résultats obtenus suite aux différentes simulations. Comme nous l'avons mentionné précédemment, la première partie de la simulation, sera consacrée à la simulation des différents modes de fonctionnement séparément. Ensuite nous passerons à la deuxième partie qui consiste à simuler la solution proposée.

4.3.1 Débit et PSNR en sortie du codeur

La figure 4.1 montre le débit obtenu en sortie du décodeur, pour chaque mode de fonctionnement. De même sur cette figure nous pouvons voir la moyenne du PSNR des composantes Y, U et V des 300 trames de la séquence vidéo codée. Ces résultats sont obtenus suite au codage de la séquence vidéo akiyo_cif.yuv dans le logiciel de référence H.264/AVC, jm13.0. Le débit et les PSNR sont représentés sur le même graphe pour que l'impacte du débit sur le PSNR soit plus clair.

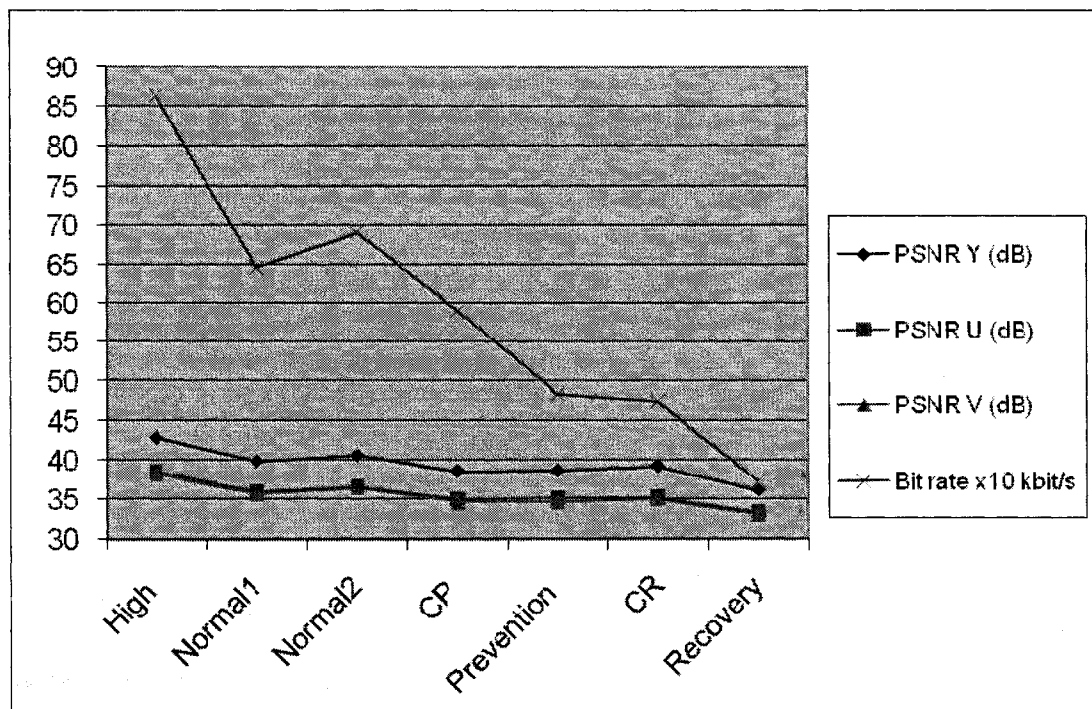


Figure 4.1 Débit et moyenne des PSNRs en sortie du codeur pour les différents modes de fonctionnement

On peut très clairement voir dans la figure 4.1 qu'en passant du mode *High* au mode *Recovery*, le débit en sortie du codeur diminue considérablement, pour passer de 863 kbits/s à 374 kbits/s. De plus, nous pouvons voir que cette diminution du débit n'entraîne pas une grande détérioration de la qualité de la vidéo. En effet, en regardant les moyennes des PSNRs des trois composantes de l'image, nous remarquons que ces valeurs chutent seulement de 8 dB (15.7 %) au maximum, en passant de 42.93 dB à 36.17 dB pour la composante Y qui est la plus importante dans l'évaluation de la qualité de l'image, quand le débit diminue de plus que la moitié. Ces résultats montrent qu'à la sortie du codeur, et en utilisant les modes de fonctionnement, nous parvenons à garder la qualité relativement stable, tout en diminuant largement de débit.

4.3.2 PSNR et temps de décodage en sortie du décodeur

La figure 4.2 montre les résultats obtenus après le décodage de la vidéo. Les résultats en sortie du décodeur sont donnés en PSNR des trois composantes Y, U et V de l'image, ainsi que le temps mis pour le décodage de la séquence vidéo. La vidéo décodée à ce niveau parvient directement de la sortie du codeur, et non pas de la sortie du réseau simulé. Le décodeur utilisé est le décodeur de référence jm13.0. Le temps de décodage et les PSNRs sont représentés sur le même graphe pour que l'impacte de la complexité de décodage sur les PSNRs soit clair.

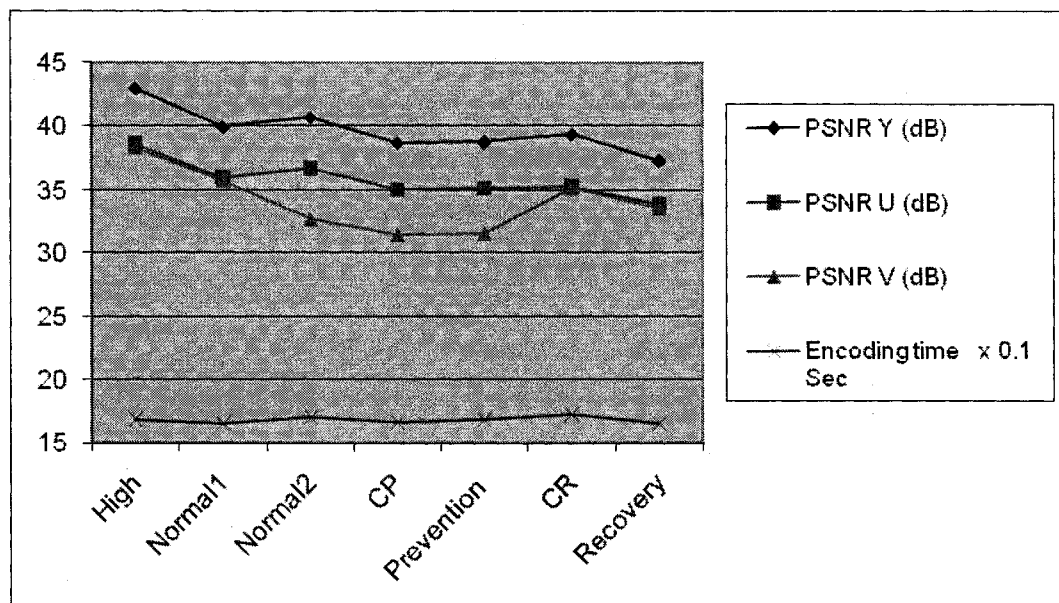


Figure 4.2 Temps de décodage et moyenne des PSNRs en sortie du décodeur pour les différents modes de fonctionnement

Suite à ces résultats, nous pouvons voir que les valeurs des PSNRs au décodeur sont très proches de ceux obtenus au codeur. Cela est normal, parce que nous utilisons un codage sans perte, et on décode la vidéo directement après sa sortie du codeur. Ainsi, nous n'avons pas de pertes induites par le codage de la vidéo, ni de pertes de paquets à cause

de la transmission sur un réseau. D'autre part, nous pouvons remarquer que le temps de décodage est très stable pour tous les modes de fonctionnement. En effet, le temps de décodage varie au maximum de 4,2 % de sa valeur minimale. Ces résultats nous montrent que les modes de fonctionnements définis, codes sans perte significative la vidéo, sans pour cela augmenter la complexité du décodage.

4.3.3 Résultats pour différents cas de coupure de liens

Dans cette partie nous allons envoyer la vidéo à travers un réseau ad hoc. Nous allons simuler différents états de coupures de liens dans ce réseau et analyser les résultats obtenus, nous commencerons par présenter les différents résultats obtenus, que nous comparerons ensuite au résultat obtenu pour notre solution proposée. Dans ces résultats, nous avons introduit un nouveau paramètre pour le calcul de la qualité de la vidéo reçue. Ce paramètre donne la moyenne de la qualité des 300 trames envoyées en fonction des trames reçues. Le calcul de ce paramètre *Quality* se fait de la façon suivante :

$$Quality = moyenne\ PSNR * nombre\ de\ trames\ reçues / nombre\ de\ trames\ envoyées$$

A l'aide de ce paramètre, nous allons donc pouvoir évaluer la qualité globale de la vidéo, suivant le nombre de trames reçues et leur PSNR.

4.3.3.1 Faible taux de coupure de liens

La figure 4.3 montre les résultats obtenus, suite à la simulation, dans le cas où les nœuds se déplacent dans un espace restreint. Puisque l'espace est petit, mais le nombre de nœuds reste le même, le taux de coupure de liens va nécessairement être petit. Ainsi nous serons capables de tester les performances de chaque mode de fonctionnement dans un réseau à faible probabilité de coupure de liens.

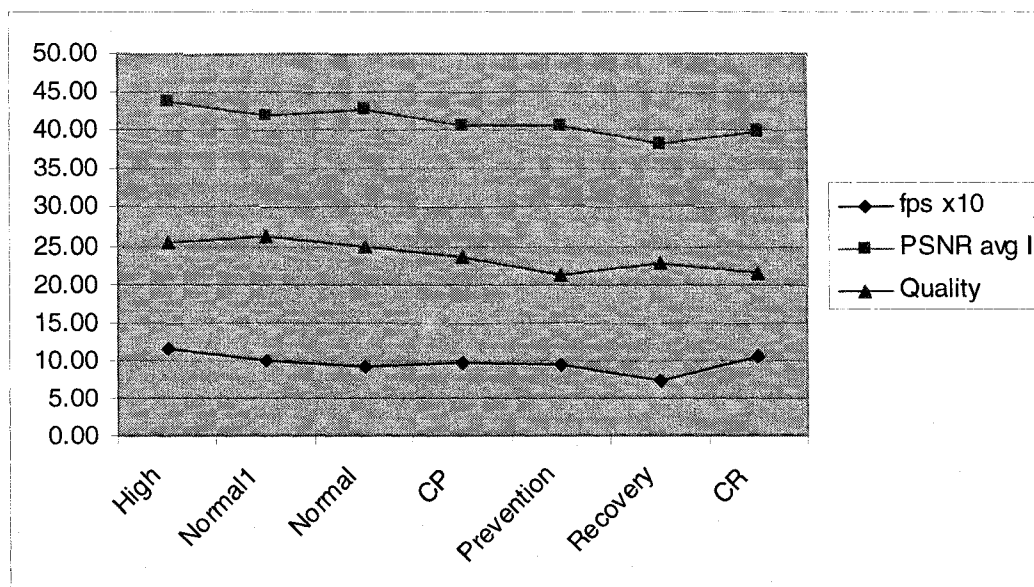


Figure 4.3 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à faible taux de coupure de liens

Comme nous pouvons le remarquer dans la figure 4.3, les modes de fonctionnement *High*, *Normal1* et *Normal2* sont plus performants dans les réseaux à faible taux de coupure de liens. En effet, nous remarquons que pour ces trois modes de fonctionnement, la valeur de *Quality* reste relativement proche de 25 dB, voir tableau 4.1. Alors que pour les autres modes, cette valeur diminue rapidement, pour atteindre un minimum de 21.20 dB pour le mode *Prevention*. Nous pouvons donc conclure, que quand le réseau fonctionne parfaitement, ce qui est généralement le cas pour une certaine durée de temps après l'établissement d'un chemin, les modes *High*, *Normal1* et *Normal2*, sont les plus avantageux à utiliser.

Tableau 4.1 Résultats de simulation dans un réseau à faible taux de coupure de liens

Faible taux de coupure de liens			
	PSNR avg	nbr frames avg x10	Quality
High	26.77	28.50	25.43
Normal1	26.22	30.00	26.22
Normal2	25.40	29.24	24.75
CP	25.44	27.64	23.44
Prevention	22.66	28.06	21.20
Recovery	25.50	26.86	22.83
CR	22.88	28.06	21.40

4.3.3.2 Fort taux de coupure de liens

Suite au teste d'un faible taux de coupure de lien, nous avons effectué un test dans un environnement à fort taux de coupure de liens. Les résultats obtenus sont présentés dans la figure 4.4. Pour simuler un réseau à fort taux de coupure de liens, nous avons augmenté la vitesse des nœuds mobiles. Ainsi, dans cette simulation, les nœuds se déplacent aune vitesse maximale de 15 m/s, dans un espace de 1 km², ce qui va induire de coupures de liens plus nombreux dans le réseau.

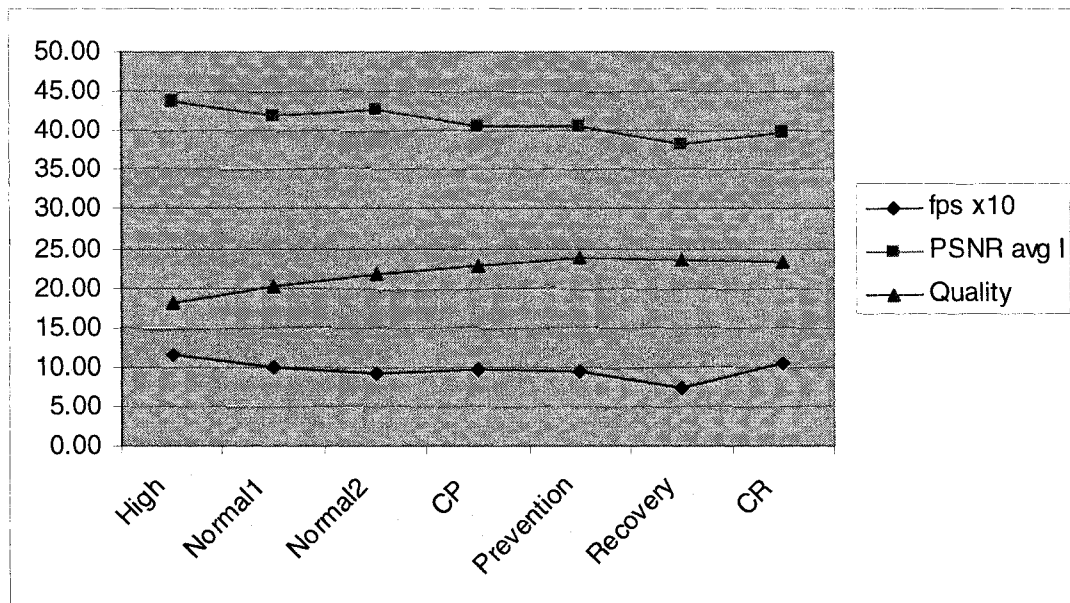


Figure 4.4 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à grand taux de coupure de liens

Les résultats obtenus dans ce cas, sont en quelque sorte l'inverse des résultats dans le cas de réseau à faible taux de coupure de liens, ce qui est normal. En effet, les résultats présentés dans la figure 4.4, montrent une amélioration de la qualité de la vidéo à la réception, pour les modes de fonctionnement *Prevention*, *Recovery* et *Congestion Recovery*. Nous remarquons que pour un fort taux de coupure de liens la performance de ces modes de fonctionnement s'améliore par rapport aux modes à haut débit, à savoir, les modes *High*, *Normal1* et *Normal2*, voir tableau 4.2. Ce qui nous mène à la conclusion qu'en cas de coupure de liens, les modes les plus avantageux à utiliser sont les modes *Prevention* et *Recovery*.

Tableau 4.2 Résultats de simulation dans un réseau à fort taux de coupure de liens

Fort taux de coupure de liens			
	PSNR avg	nbr frames avg x10	Quality
High	21.70	24.90	18.01
Normal1	26.05	23.36	20.28
Normal2	22.66	28.76	21.72
CP	24.08	28.48	22.86
Prevention	26.47	26.86	23.70
Recovery	26.74	26.42	23.55
CR	24.03	29.13	23.33

4.3.3.3 Fort taux de coupure de liens avec grande vitesse de réparation

Dans la figure 4.5, nous présentons les résultats obtenus suite à une simulation avec un réseau à fort taux de coupure de liens, mais une grande disponibilité de liens pour la réparation. Pour simuler ce réseau, nous allons augmenter la vitesse des nœuds mobiles, ce qui augmente le taux de coupure de liens, mais nous allons aussi diminuer l'espace dans lequel se déplacent les nœuds mobiles. Ceci va permettre une grande disponibilité de liens de réparation, donc les coupures sont fréquentes mais la réparation rapide.

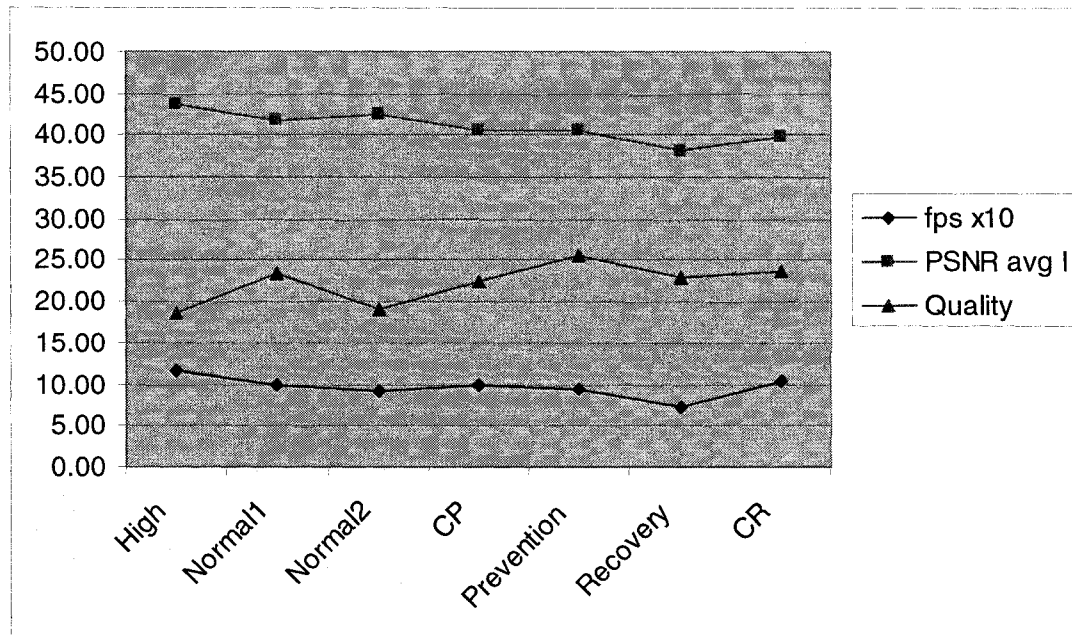


Figure 4.5 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau à grand taux de coupure de liens et grande vitesse de réparation

Les résultats obtenus suite à cette simulation, montrent que les modes de fonctionnements *Prevention*, *Recovery*, *Congestion recovery*, ainsi que le mode *Normal1* offrent une meilleure qualité de vidéo que les autres modes, pour lesquels nous remarquons une grande détérioration du facteur *Quality*. En effet, alors que les valeurs obtenus pour les modes *High* et *Normal2* passent en dessous de 20 dB, voir tableau 4.3, les valeurs des autres modes restent proches de 25 dB, alors que le mode *Congestion Prevention* n'offre aucun changement majeur durant les différentes simulations. Nous pouvons donc conclure que pour les réseaux à fort taux de coupure de liens, et une grande rapidité de réparation, l'utilisation des modes de fonctionnement à faible débit est avantageuse.

Tableau 4.3 Résultats de simulation dans un réseau à fort taux de coupure de liens avec une grande vitesse de réparation

Fort taux de coupure de liens avec grande vitesse de réparation			
	PSNR avg	nbr frames avg x10	Quality
High	23.74	23.40	18.52
Normal1	25.89	27.12	23.40
Normal	24.74	23.04	19.00
CP	23.85	28.38	22.56
Prevention	26.55	28.80	25.49
Recovery	24.03	28.58	22.90
CR	24.52	28.82	23.55

4.3.3.4 Simulation de notre solution

Suite à la simulation de chaque mode de fonctionnement séparément, dans différents états de coupure de lien du réseau, nous allons simuler la solution que nous proposons. Le réseau utilisé passera par les trois états précédemment décrits, à savoir, un faible taux de coupure de liens, puis un fort taux de coupure de lien avec une réparation lente, et enfin un fort taux de coupure de lien avec une réparation rapide. Les résultats obtenus sont présentés dans la figure 4.6.

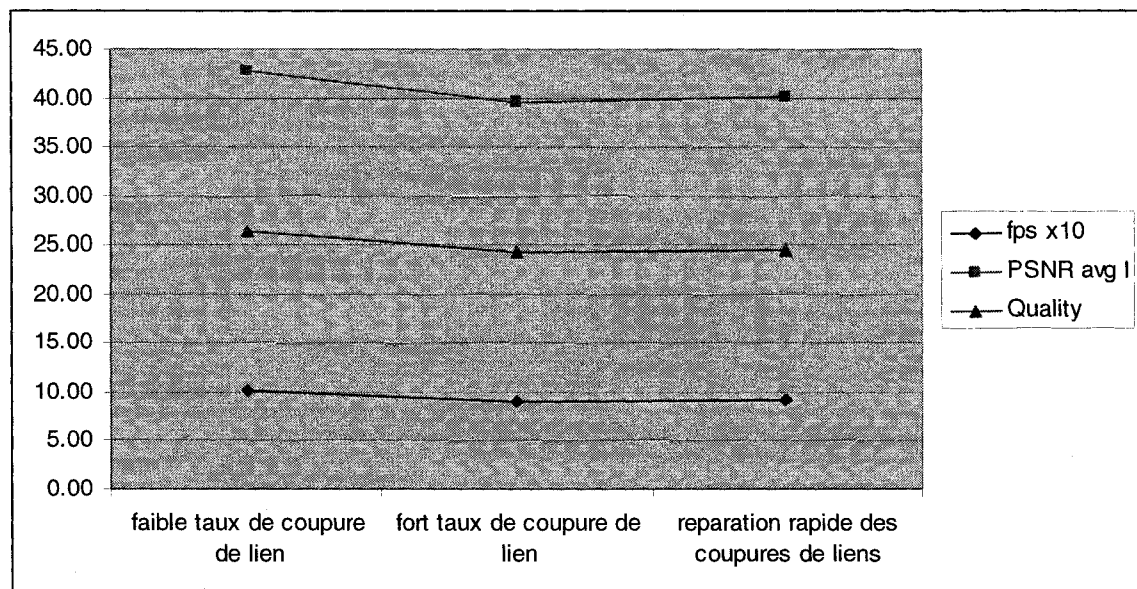


Figure 4.6 Débit et PSNR à l'entrée, et qualité de la vidéo en sortie du réseau passant par différents états

Les résultats présentés dans la figure 4.6 montrent une grande stabilité de la qualité de la vidéo à la réception. En effet, nous pouvons remarquer que la valeur de *Quality* reste toujours très proche de 25 dB, voir tableau 4.4. De plus nous remarquons que les résultats obtenus suite à l'utilisation de notre solution sont au pire des cas, meilleurs que les résultats obtenus pour chaque mode de fonctionnement séparément. Ainsi, quand le réseau est à faible taux de coupure de lien, la valeur du paramètre *Quality* est au pire des cas, meilleur de 0.4 % des résultats précédents, l'amélioration s'élève à 24.30 % pour les modes les moins performants, mais il est difficile d'améliorer plus les performances dans le cas de faible taux de coupure, puisque le réseau fonctionne parfaitement, donc la meilleure qualité de la vidéo à la source donnera la meilleure qualité à la destination. Alors que quand les coupures deviennent fréquentes, l'amélioration est au minimum de 10.3 %, au maximum de 35.17 %. De même pour les cas de réparation rapide, l'adaptabilité de notre application améliore les résultats de 1.1 % par rapport au meilleur résultat obtenus précédemment, et de 39.17 % par rapport au pire résultat. Nous

pouvons donc conclure que pour les différents états de coupures de liens, notre solution donne de meilleurs résultats que l'utilisation d'une application non adaptative.

Tableau 4.4 Résultats de simulation de la solution proposée

	fps x10	PSNR avg	Quality
Faible taux de coupure de lien	10.20	42.69	26.35
Fort taux de coupure de lien	9.00	39.57	24.34
Réparation rapide des coupures de liens	9.21	40.14	25.77

4.3.4 Résultats pour différents cas de congestion

Dans cette partie nous étudierons l'impacte des congestions sur la transmission de la vidéo. Nous simulons le réseau ad hoc dans différents états de congestion. La congestion augmente graduellement en augmentant le nombre de connexions simultanées sur le réseau. Ainsi le nombre de connexions commence à 4, puis passe à 5, 7, 8, 10 puis le nombre maximum de connexions simultanées est de 12. Nous analyserons les résultats obtenus pour les différents modes de fonctionnement séparément, ensuite, nous simulerons notre solution dans un réseau passant par plusieurs états de congestion, pour enfin comparer les résultats obtenus à cette simulation, aux résultats des différents modes.

4.3.4.1 Mode *High*

La figure 4.7 présente les résultats obtenus suite à la simulation de la source fonctionnant en mode *High*, sur un réseau dans différents état de congestion.

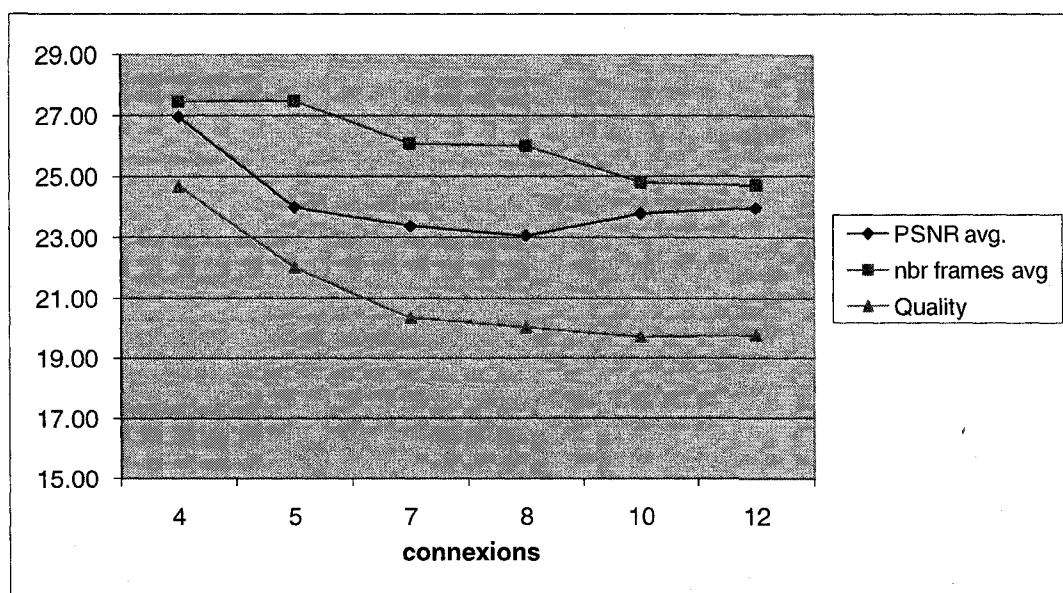


Figure 4.7 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *High* dans un réseau dans différents états de congestion

Les résultats présentés dans la figure 4.7, montrent que le mode *High* offre de bons résultats pour un faible nombre de connexions, soit 4 et 5 connexions. En effet, pour 4 connexions simultanées, ce mode offre une valeur de *Quality* de près de 25 dB, voir tableau 4.5, puis à partir de 7 connexions, cette valeur chute de plus de 17 %. Nous pouvons donc conclure que ce mode est bien adapté à un réseau à faible congestion.

Tableau 4.5 Résultats de simulation du mode *High* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	26.97	27.48	24.71
5	24.02	27.50	22.01
7	23.41	26.08	20.35
8	23.09	26.02	20.02
10	23.82	19.70	19.70
12	23.99	19.77	19.77

4.3.4.2 Mode *Normal1*

Dans la figure 4.8 nous représentons les résultats obtenus suite à la simulation de la source fonctionnant en mode *Normal1*, sur un réseau qui devient graduellement plus congestionné.

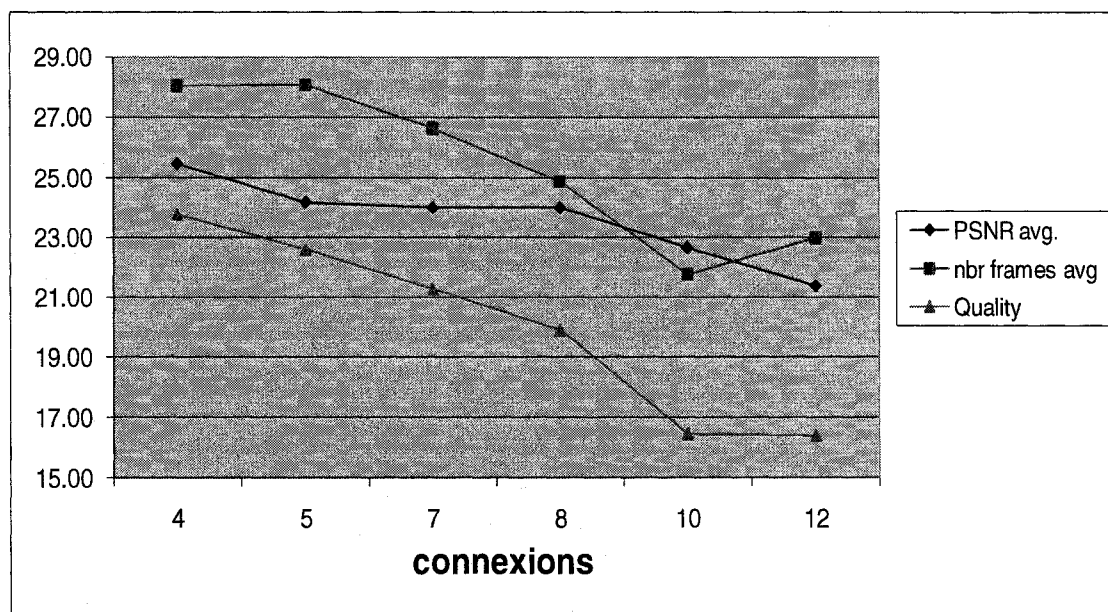


Figure 4.8 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *Normal1* dans un réseau dans différents états de congestion

Comme nous pouvons le voir dans la figure 4.8, le mode *Normal1* est moins performant que le mode *High* pour le cas de 4 connexions simultanées. Mais la détérioration de la valeur de *Quality* avec l'augmentation de la congestion est plus faible. Ainsi, entre 4 connexions et 7 connexions, la valeur de *Quality* chute de 10.48 % pour ce mode, voir tableau 4.6, alors qu'elle chutait de plus de 17 % pour le mode *High*. Nous concluons donc que dans un réseau stable nous pouvons utiliser le mode *Normal1* en conservant la stabilité du réseau, ainsi que la qualité de la vidéo.

Tableau 4.6 Résultats de simulation du mode *Normal1* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	25.47	28.02	23.78
5	24.17	28.06	22.61
7	24.00	26.62	21.29
8	24.00	24.88	19.90
10	22.65	21.78	16.45
12	21.38	23.00	16.39

4.3.4.3 Mode *Normal2*

La figure 4.9 présente les résultats obtenus suite à la simulation en mode *Normal2*. Dans les mêmes conditions que les testes de congestion précédents.

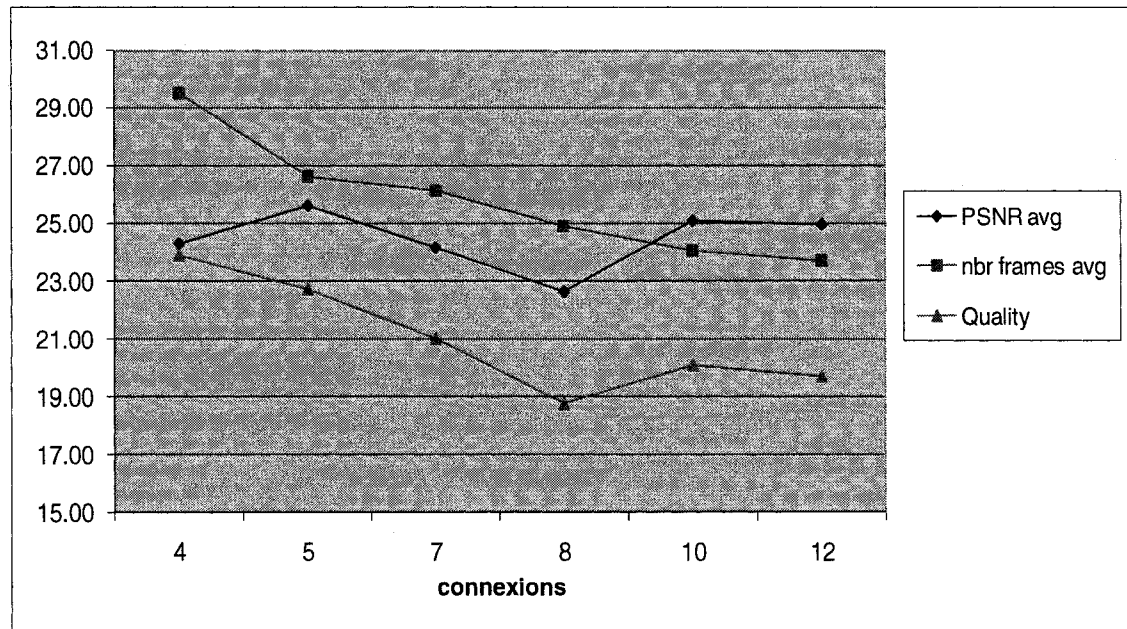


Figure 4.9 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *Normal2* dans un réseau dans différents états de congestion

Ces résultats sont équivalents à ceux obtenus précédemment pour le mode *Normal1*. Le mode *Niormal2* n'étant pas très différent du mode précédent, il est normal d'avoir des résultats qui ont un comportement similaire, voir tableau 4.7. Donc l'analyse et l'utilisation de ce mode seront les même que ceux du mode *Normal1*.

Tableau 4.7 Résultats de simulation du mode *Normal2* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	24.29	29.50	23.88
5	25.61	26.62	22.73
7	24.14	26.12	21.02
8	22.63	24.88	18.77
10	25.08	24.04	20.10
12	24.96	23.68	19.70

4.3.4.4 Mode Congestion Prevention

Les résultats de la figure 4.10 sont obtenus suite à la simulation du mode *CP*. Dans ces simulations aussi, l'état du réseau change, en augmentant graduellement la congestion, afin de tester la validité de ce mode en cas de congestion éventuelle.

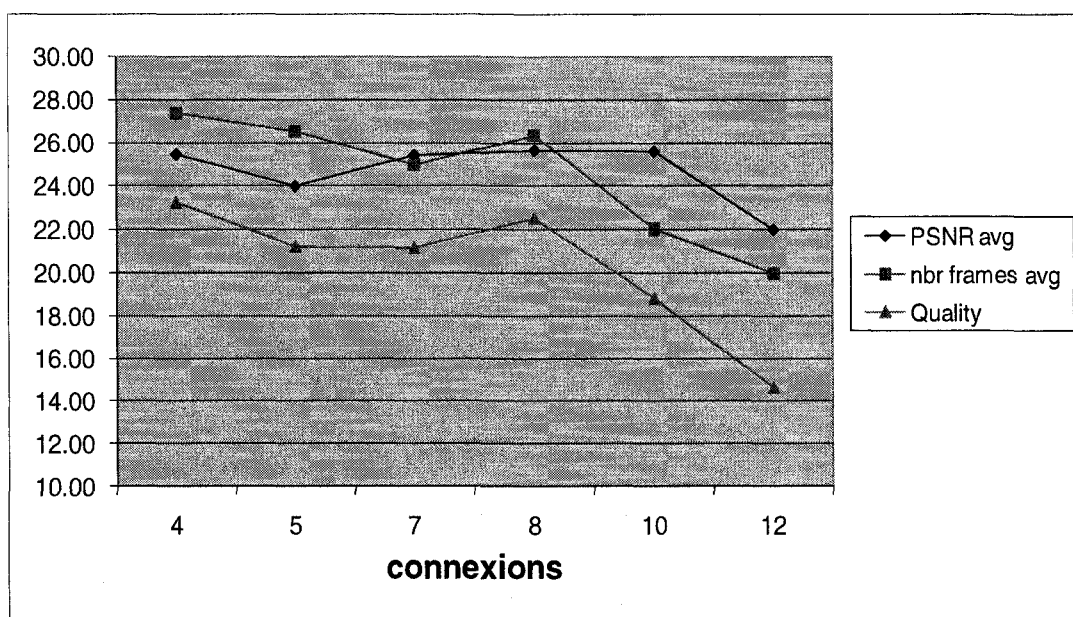


Figure 4.10 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *CP* dans un réseau dans différents états de congestion

Dans la figure 4.10, nous pouvons voir clairement l'importance de ce mode en cas de début de congestion. La courbe de cette figure montre une certaine stabilisée pour un nombre de connexions inférieur à 8. En effet, la valeur de *Quality* fluctue autour de 22 dB pour les cas à 4, 5, 7 et 8 connexions, voir tableau 4.8. Mais pour plus que 8 connexions cette valeur diminue rapidement de plus de 35 % pour atteindre 14.64 dB.

Tableau 4.8 Résultats de simulation du mode *CP* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	25.47	27.43	23.26
5	24.00	26.54	21.23
7	25.43	24.98	21.18
8	25.66	26.34	22.53
10	25.62	22.04	18.82
12	21.98	19.98	14.64

4.3.4.5 Mode *Prevention*

La figure 4.11 présente les résultats obtenus pour la simulation du mode *Prevention* sur un réseau à congestion augmentante.

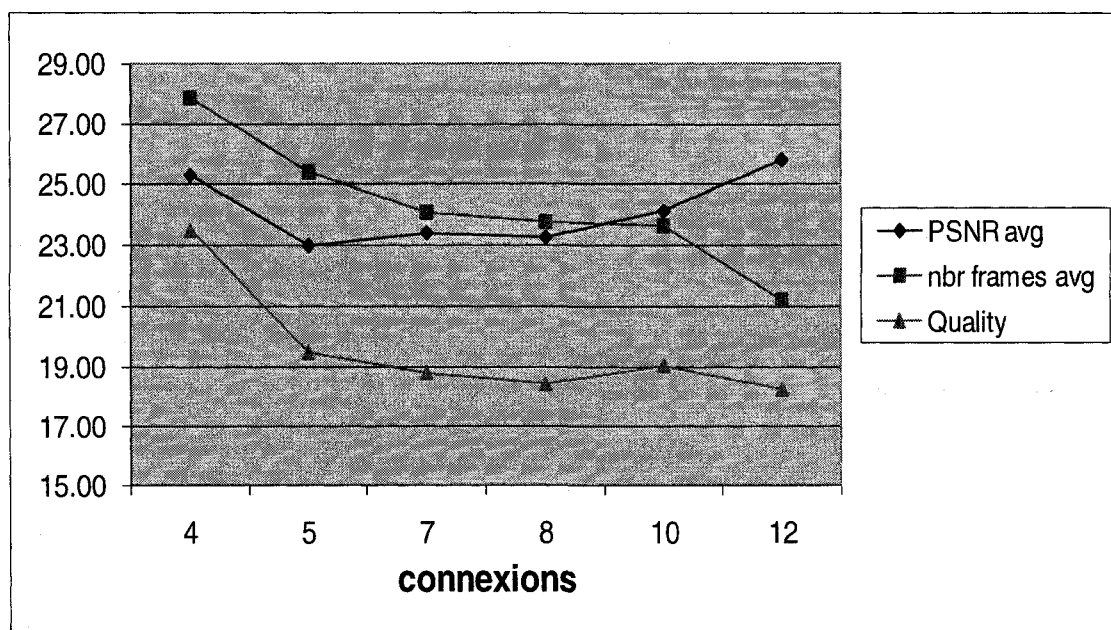


Figure 4.11 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *Prevention* dans un réseau dans différents états de congestion

Comme pour les modes précédents, les résultats du mode *Prevention* présentent une détérioration qui augmente avec le nombre de connexions. Mais à partir de 7 connexions, nous retrouvons une certaine stabilité. En effet pour les simulations avec plus de 7 connexions, nous obtenons des valeurs de *Quality* qui fluctuent autour de 19 dB, voir tableau 4.9. Cette valeur est faible certainement, mais la stabilité atteinte est importante pour notre application.

Tableau 4.9 Résultats de simulation du mode *Prevention* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	25.28	27.88	23.49
5	22.96	25.40	19.44
7	23.39	24.08	18.77
8	23.24	23.76	18.41
10	24.11	23.66	19.02
12	25.80	21.20	18.23

4.3.4.6 Mode *Recovery*

Les résultats obtenus suite à la simulation du mode *Recovery* sont présentés dans la figure 4.12. Ce mode défini pour des cas de coupure de liens est aussi testé dans des cas de congestions de réseau.

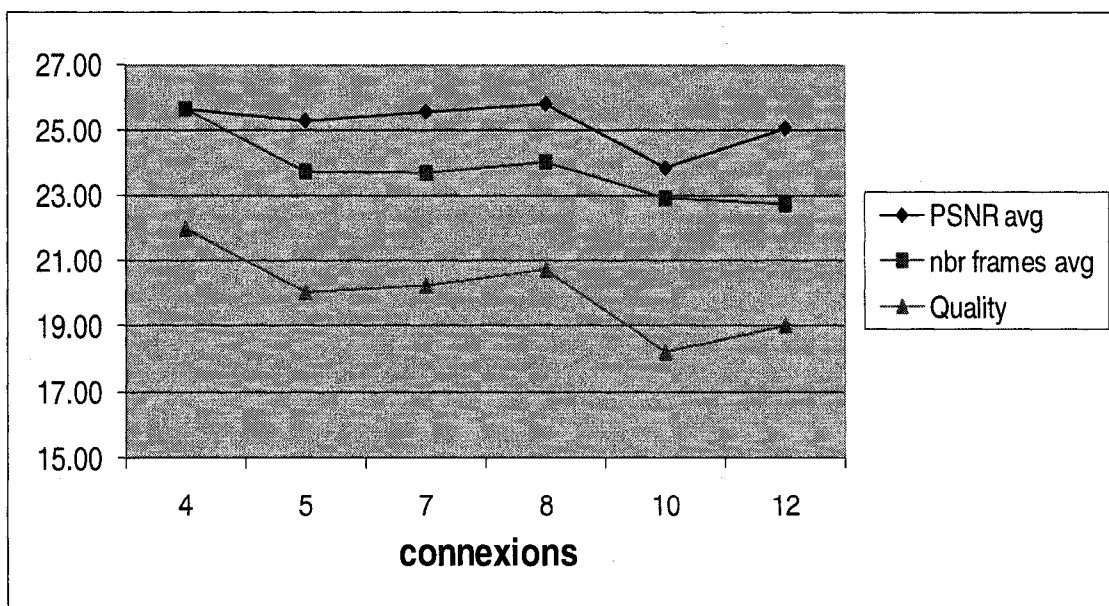


Figure 4.12 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *Recovery* dans un réseau dans différents états de congestion

Les résultats obtenus pour ce mode montrent une détérioration continue de la qualité de la vidéo. Même si cette détérioration n'est pas très aigue, mais pour un faible nombre de connexions aussi la qualité de la vidéo est médiocre, et elle fluctue majoritairement entre 19 dB et 21 dB, voir tableau 4.10. Ceci s'explique par le fait que ce mode est défini pour être utilisé quand le réseau est dans son pire état. Le but de ce mode est donc simplement de faire parvenir à tout prix une image à la destination, pour cela nous avons minimisé la qualité de l'image pour minimiser le débit à la source, donc même si la vidéo parvient à la destination elle sera de qualité très médiocre.

Tableau 4.10 Résultats de simulation du mode *Recovery* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	25.68	25.66	21.97
5	25.32	23.72	20.02
7	25.59	23.70	20.22
8	25.83	24.04	20.70
10	23.84	22.92	18.22
12	25.07	22.74	19.00

4.3.4.7 Mode *Congestion Recovery*

La figure 4.12 présente les résultats obtenus pour le dernier mode de fonctionnement simulé, le mode *CR*. L'environnement de simulation est le même que pour toutes les simulations précédentes.

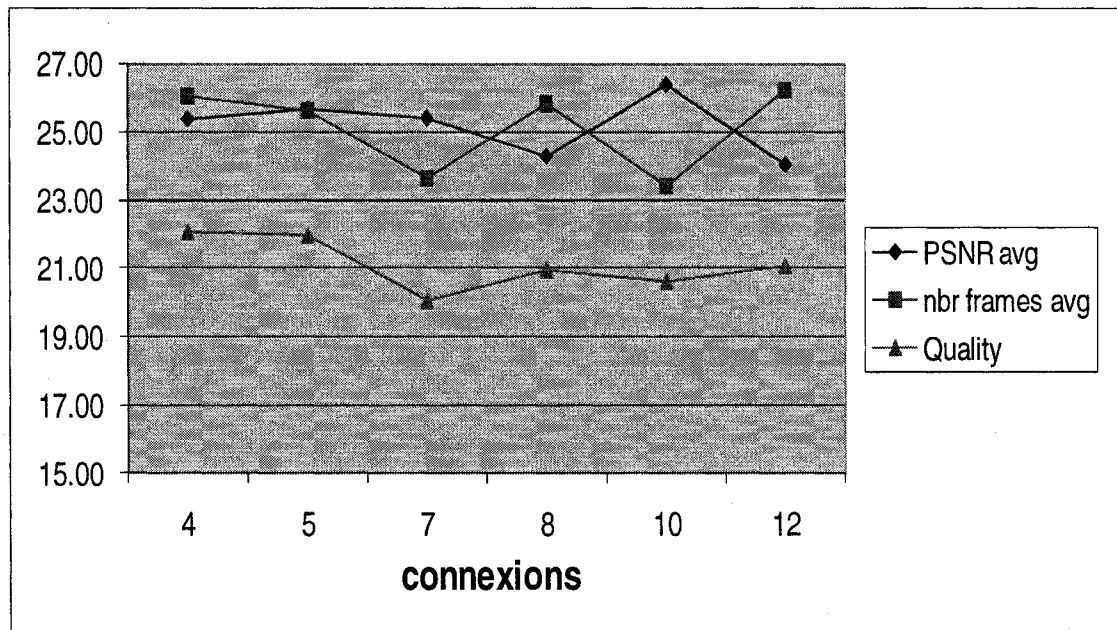


Figure 4.13 Trames reçues, PSNR et *Quality* en sortie du décodeur pour le mode *CR* dans un réseau dans différents états de congestion

La courbe de la figure 4.12, montre une stabilité pour les différents nombre de connexions. En effet, nous remarquons que la valeur de *Quality* fluctue autour de 21 dB pour le nombre de connexions allant de 4 à 12, voir tableau 4.11. Nous pouvons donc conclure que ce mode de fonctionnement, même si moins avantageux dans les réseaux à faible congestion, offre de meilleurs résultats, et une grande stabilité de la qualité de la vidéo quand la congestion augmente. Ce qui confirme les prévisions théoriques de la partie 3.3.

Tableau 4.11 Résultats de simulation du mode *CR* dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	25.41	26.08	22.09
5	25.70	25.66	21.98
7	25.44	23.66	20.06
8	24.31	25.86	20.96
10	26.41	23.42	20.62
12	24.06	26.26	21.06

4.3.4.8 Solution proposée

Dans la figure 4.13, nous voyons les résultats obtenus suite à la simulation de notre solution. La simulation est faite sur un réseau passant par plusieurs états de congestion. En comparant ces résultats aux résultats précédents, nous allons pouvoir arriver à une conclusion concernant l'avantage de l'utilisation de notre solution.

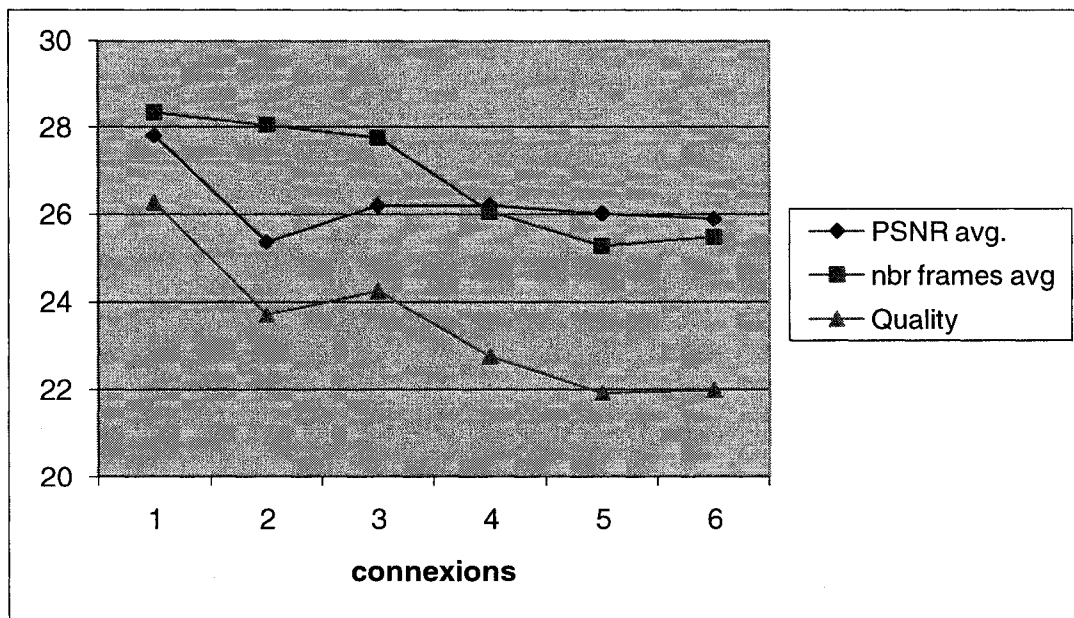


Figure 4.14 Trames reçues, PSNR et *Quality* en sortie du décodeur pour la solution proposée dans un réseau dans différents états de congestion

Dans cette figure, nous avons les résultats obtenus suite à la simulation de notre solution, dans un réseau passant par plusieurs états de congestion. Ces résultats montrent une grande stabilité de la qualité de la vidéo reçue. En effet, la valeur de *Quality*, chute seulement de 4 dB entre le minimum et le maximum de congestion, voir tableau 4.12. En plus de la stabilité, ces résultats sont, en tout point, meilleurs que les résultats obtenus en simulant chaque mode tout seul. Ainsi, le meilleur résultat obtenu en utilisant notre solution est meilleur de 6% que le meilleur résultat obtenu précédemment, et le pire, est meilleur de 34 % que le pire résultat obtenu précédemment.

Tableau 4.12 Résultats de simulation de notre solution dans un réseau à différents états de congestion

connections	PSNR avg.	nbr frames avg	Quality
4	27.81	28.33	26.26
5	25.35	28.04	23.69
7	26.19	27.76	24.23
8	26.20	26.05	22.75
10	26.01	25.27	21.91
12	25.88	25.48	21.98

Ainsi, et suite à la simulation des différents modes de fonctionnement séparément, puis notre solution proposée, dans le même réseau, nous arrivons à la conclusion qu'en rendant l'application de vidéoconférence adaptative à l'état du réseau, et en utilisant les modes de fonctionnement que nous proposons, nous parvenons à améliorer la qualité de la vidéo à la réception.

Dans ce chapitre nous avons décrit la méthodologie d'évaluation de la solution proposée. Nous avons présenté en détail les outils utilisés et les simulations effectuées. Ensuite nous avons présenté les résultats obtenus suite à chaque simulation, en divisant

les résultats entre ceux obtenus pour les coupures de liens, et ceux obtenus pour les différents taux de congestion. Et finalement nous avons analysé, puis comparé les résultats obtenus pour chaque mode de fonctionnement, aux résultats obtenus pour l'application adaptative proposée.

CHAPITRE 5 CONCLUSIONS

Ce travail présente une méthode pour améliorer la qualité de service des applications de vidéoconférence dans les réseaux mobiles ad hoc. La solution proposée repose sur l'adaptabilité dynamique de l'application à l'état du réseau. Dans ce dernier chapitre, nous présentons une synthèse du travail effectué, ainsi que les contributions de ce travail, nous exposons les limitations essentielles de la solution proposée, pour enfin proposer des directions de recherches futures dans le domaine.

5.1 Synthèse du travail et contribution

Les réseaux mobiles ad hoc (MANet) remplacent progressivement les réseaux câblés, ainsi que les réseaux sans fil traditionnels. De même, la diversité des applications sur ces réseaux est de plus en plus grande. La majorité de ces applications, surtout les applications de vidéoconférence, ont des contraintes de qualité de service difficilement satisfaites par les réseaux ad hoc. En effet ce genre de réseau présente plusieurs limitations, d'abord en bande passante due au médium utilisé qui est l'air, puis en disponibilité puisque la topologie change dynamiquement ce qui induit des coupures aléatoires des liens, et enfin des limitations des équipements mobiles utilisés dans ces réseaux, tel que la puissance de traitement et l'énergie disponible.

La solution proposée dans ce mémoire repose sur une méthode d'adaptabilité dynamique de l'application de vidéoconférence à l'état du réseau. Ainsi nous avons défini sept états dans lesquels un réseau ad hoc peut se trouver durant une transmission. Puis en se basant sur les résultats et les propositions existants dans la littérature, nous avons tiré plusieurs paramètres qui influencent la qualité de la transmission vidéo sur les réseaux ad hoc. Ainsi en faisant varier les valeurs de ces différents paramètres, nous avons défini sept modes de fonctionnement de l'application à la source, un pour chaque état possible du réseau. Finalement nous avons défini une architecture *cross-layer* pour permettre à l'application à la source de connaître à tout moment l'état du réseau, et ainsi de prendre une décision sur le mode dans lequel elle doit fonctionner.

Pour évaluer la solution que nous proposons, nous avons fait une série de simulation. Dans un premier temps nous avons simulé les différents modes de fonctionnement séparément dans les différents états possibles du réseau. Ensuite nous avons simulé la solution proposée dans un réseau passant successivement dans les différents états possibles. Finalement, nous avons comparé les résultats obtenus pour les deux séries de simulations.

Les résultats obtenus suites aux simulations, montrent qu'en rendant l'application dynamiquement adaptative à l'état du réseau, nous obtenons des résultats toujours meilleurs que ceux obtenus pour une application non adaptative, peu importe le mode utilisé. Ainsi, les résultats obtenus suite à la simulation de notre solution sur un réseau passant par les différents états possibles, montrent une grande stabilité de la qualité de la vidéo à la réception. De même, ces résultats montrent une supériorité de la qualité de la vidéo reçue, mesurée en dB, en utilisant une application adaptative, par rapport à l'application statique fonctionnant tout le temps dans le même mode. Ce qui valide l'étude théorique et la solution proposée dans ce document.

5.2 Limitations de notre travail

La solution proposée, permet des améliorations des applications de vidéoconférence existantes dans la littérature. Mais d'autres améliorations sont envisageables. En effet, notre solution repose sur le changement dynamique de l'application de vidéoconférence au niveau de la source, pour la rendre adaptative à l'état du réseau. Mais aucun mécanisme de correction d'erreur n'est proposé au niveau de la destination, bien que le standard H.264/AVC offre plusieurs de ces mécanismes, qui, si correctement utilisés, permettent d'atteindre de meilleurs résultats, ainsi que des niveaux d'amélioration encore plus élevés.

De plus nous avons proposé dans la partie 3.5 une méthode pour minimiser le débit propre aux messages de contrôles niveau application, échangés entre la source et la destination, sans donner les détails et les méthodes d'intégration possible de cette méthode.

5.3 Travaux futurs

Le travail sur la qualité de service dans les réseaux ad hoc, et surtout la qualité des applications de vidéoconférence dans ces réseaux, est loin d'être finie. Notre solution présente une nouvelle approche dans le domaine de la vidéoconférence sur ces réseaux, et peut être utilisée comme base pour des travaux futurs.

L'étude que nous avons faite pour les réseaux mobile ad hoc (MANet), peut s'étendre aux réseaux sans fil personnels IEEE 802.15. (WPAN). En effet la recherche sur la transmission d'image et de vidéo sur ces réseaux est en pleine expansion. Les réseaux WPANs présentent des limitations plus strictes que ceux dans les réseaux MANets, surtout en termes de débit, qui est très limité dans ces réseaux, et de topologie qui peut changer plus rapidement dans ces réseaux que dans les autres types de réseaux, même les MANets. Comme vu précédemment, notre solution permet d'atteindre des débits assez faibles, et de s'adapter à des conditions très difficiles du réseau, tout en offrant une bonne qualité de la vidéo, ainsi qu'une grande stabilité, peu importe l'état du réseau. Nous croyons donc qu'il serait intéressant de se baser sur la solution proposée dans ce document, ou de l'étendre pour la rendre applicable aux réseaux personnels WPAN.

RÉFÉRENCES

- [1] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. "Optimized Link State Routing protocol for ad hoc networks", In *Proceedings of the IEEE International Multitopic Conference (INMIC 2001), Pakistan, 2001*.
- [2] C. E Perkins, E. M. Royer "Ad hoc On-Demand Distance Vector Routing", *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100*.
- [3] D. B. Johnson, D. A. Maltz "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing, Tomasz Imielinski and Hank Korth (Editors), Vol. 353, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996*.
- [4] V. Park, S. Corson "Temporally-Ordered Routing Algorithm (TORA) Version 1", *Functional Specification, Internet Draft, IETF MANET Working Group, June 2001*
- [5] C.T. Calafate, M. P. Malumbres, and P. Manzoni, "Performance of H.264 compressed video streams over 802.11b based MANETs", in *proceedings of the 24th international conference, Distributed Computing Systems Workshops, 23-24 March 2004, pp 776-781*.
- [6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", in *IEEE transactions on circuits and systems for video technology, vol. 13, no. 7, July 2003, pp 560-576*.
- [7] C. Perkins, E. M. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", *Experimental RFC 3561, University of California, Santa Barbara, July 2003*.
- [8] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer, "RTP Payload Format for H.264 Video", *Internet proposed standard RFC 3984, February 2005*.

- [9] ITU-T, "Advanced Video Coding for Generic Audiovisual Services", *ITU-T Recommendation H.264*, 2003.
- [10] D. Lagall, "MPEG: A Video Compression Standard for Multimedia Application", *Commun. ACM*, vol. 34, pp. 46-58, April 1991.
- [11] ISO/IEC, "The MPEG-2 International Coding Standard", *Reference number ISO/IEC 13818-2*, 1996.
- [12] T. Sikora, "The MPEG-4 Video Standard Verification Model", *In IEEE Transactions for Circuits and Systems for Video Technology*, vol. 7, pp 19-31, February 1997.
- [13] ITU-T, "Video Coding for Audiovisual Services at p x 64 bits", *ITU-T Recommendation H.261*, March 1993.
- [14] ITU-T, "Video Coding for Low bit rate Communication", *ITU-T Recommendation H.263; version 1*, November 1995; *version 2*, January 1998; *version 3*, November 2000.
- [15] G. Raja, M.J. Mirza, "JVT H.264/AVC: Evaluation with Existing Standards for low bit rate Video Coding", *In the 17th international conference on microelectronics*, pp. 301-304, 13-15 December 2005.
- [16] T. Wiegand, G. J. Sullivan, G. Bjntegaard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard", *In IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, issue 7, pp. 560-576, July 2003.
- [17] V. Srivastava, M. Motani, "Cross-Layer Design: a Survey and the Road Ahead", *In IEEE Communication Magazine*, vol. 43, issue 12, pp. 112-119, December 2005.
- [18] C. Calafate, M. Malumbres, "Testing the h.264 error-resilience on wireless ad-hoc networks", *In Proc. EURASIP Video/Image Processing and Multimedia Communications Conference (VIPMCC)*, pp. 789-796, 2003.

- [19] Y. Pei, V.S. Ambetkar, J. W. Modestino, Q. Qu, X. Wang, “ Enabling real-time H.264 video services over wireless ad hoc networks using joint admission and transmission power control”, *In Telecommunication Systems*, vol. 28, issue 2, pp. 231-244, 2005.
- [20] P. Buccioli, E. Masala, J. C. DE Martin, “ Adaptive H.264 video transmission over 802.11 inter-vehicular ad hoc networks”, *In Proc. of Workshop on DSP for in-vehicule and Mobile Systems*, September 2005.
- [21] H. Gharavi, K. Ban, “Cross-layer feedback control for video communications via ad-hoc networks”, *In IEEE 58th Vehicular Technology Conference*, vol. 5, pp. 2941-2945, 6-9 October 2003.
- [22] W. H. He, Z. H. Ge, Y. P. Hu, “Optimizing UDP Packet Sizes in Ad Hoc Networks”, *In international conference on wireless communications, networking and mobile computing*, 2007. *WiCom 2007*, pp. 1617-1619, 21-25 September 2007.
- [23] G. A. Di Caro, “Analysis of Simulation Environment for Mobile Ad Hoc Networks”, *Technical Report No. IDSIA-24-03*, Dale Molle Institute for Artificial Intelligent, 2003.
- [24] NS-2 Network Simulator, <http://www.isi.edu/nsnam/ns/>.
- [25] OTcl, <http://bmrc.berkeley.edu/research/cmt/cmtdoc/otcl/>.
- [26] J. Klaue, B. Rathke, and A. Wolisz, “EvalVid – A Framework for Video Transmission and Quality Evaluation”, *In Proc. of the 13th international conference on modeling techniques and tools for computer performance evaluation*, Urbana, Illinois, USA, September 2003.

- [27] A. Lie, and J. Klaue, "EvalVid-RA: Trace Driven Simulation of Rate Adaptive MPEG-4 VBR Video:", *Multimedia Systems*, 10.1007/s00530-007-0110-0, November 2007.
- [28] EvalVid – A Video Quality Evaluation Tool-set, <http://www.tkn.tu-berlin.de/research/evalvid/>.
- [29] How to evaluate H.263/H.264/MPEG4 video Transmission using the NS2 simulator, <http://140.116.72.80/~smallko/ns2/myevalvid2.htm>.
- [30] x264 – a free h264/avc encoder, <http://www.videolan.org/developers/x264.html>.
- [31] H. Toral, D. Torres, C. Hernandez, L. Estrada, "Self-Similarity, Packet Loss, Jitter, and Packet Size: Empirical Relationship for VoIP". In *18th international conference on electronics, communications and computers*, 2008. CONIELECOMP, pp. 11-16, 12 March 2008