Transport du son produit en temps réel sur les réseaux best effort

Bouillot Nicolas *

20 janvier 2003

*Un Merci particulier à Eric Gressier-Soudan pour l'initiative de ce rapport et pour ses nombreuses relectures

Table des matières

1 Introduction

L'émergence récente des applications multimédia dans les réseaux a amené de nouvelles contraintes que les protocoles traditionnels ne savent pas satisfaire de façon appropriée. Ce sont ces contraintes et plus particulièrement les solutions proposées afin de les satisfaire qui feront l'objet de cette étude. Pour être plus précis, nous allons nous focaliser sur les applications multimédia interactives avec plus de deux acteurs et dont les médias sont produits à la volé (Une conférence sur Internet par exemple). De par leur interactivité et leur volatilité (les données multimédia ne sont pas stockées mais consommées en temps réel) il faut contrôler les délais de bout en bout, la gigue et le taux de pertes pour chacun des acteurs tout en considérant leur hétérogénéité. Nous allons donc considérer le transport de la voix par rapport à un réseau de type Internet, la voix sur *Frame Relay* (VoFR) et la voix sur *ATM* (VoATM) seront donc exclues. De plus, le son étant envoyé sur le réseau dès qu'il est produit, nous ne considérerons que les méthodes de compression isochrone.

Cette synthèse est organisée comme suit. Le paragraphe 2 présentera le contexte réseau et transport dans lequel nous situons notre étude. Le paragraphe 3 exposera certains protocoles standards mis en place afin d'aider le développement des applications multimédia nous intéressant. Le paragraphe 4 présentera quant à lui les efforts de la communauté scientifique pour mettre en place un algorithme de contrôle de congestion qui tient compte de l'état du réseau sans affecter son comportement. Le paragraphe 5 proposera des mécanismes de réparation d'erreurs sur le flux multimédia, et cela directement au niveau application. Enfin nous conclurons dans le paragraphe 6.

2 Contexte

L'implantation d'applications multimédia distribuées ne se fait pas sans considération de leur l'environnement. En effet nous pouvons rapidement nous voir confronté à différents problèmes intrinsèques aux architectures des réseaux et des systèmes d'exploitations, principalement la surcharge entraînant des pertes.

Afin de conserver un aspect interactif dans ce type d'application, nous devons conserver un délai constant ou quasi-constant entre la production du média et le moment où il sera joué, cela tout en conservant une qualité minimale. La téléphonie sur IP est représentative de ce type de contraintes. Le son produit par l'interlocuteur doit être *intelligible* (ce qui constitue un critère minimum de qualité sur les données transportées) : s'il y a trop de pertes et que le discours entendu est incompréhensible, il serait préférable d'interrompre la communication.

De part l'irrégularité de la présence des utilisateurs sur le réseau et du caractère imprévisible de la charge qui va leur être nécessaire, nous pouvons assister à des périodes de congestion du réseau (surcharge) entraînant des pertes de paquets dans les files d'attente des routeurs. Les figures ci-dessous illustrent cette situation.

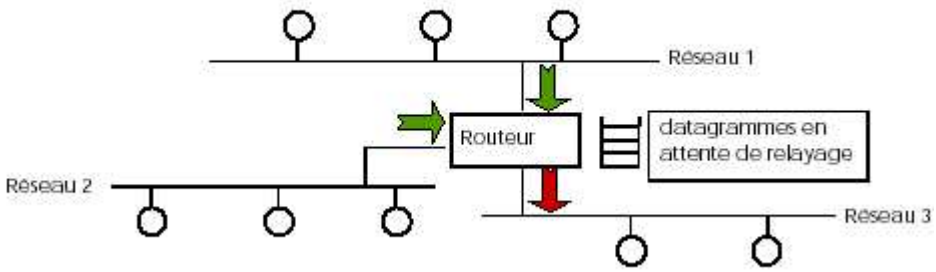

FIG. 1 – *Le goulot d'étranglement*

La figure 1 montre un goulot d'étranglement (le routeur) : les sous-réseaux 1 et 2 partagent le même routeur pour atteindre le sous-réseau 3. Le nombre de paquets dans la file d'attente du routeur risque de dépasser sa capacité et entraîner des pertes.

Sans contrôle de congestion ni contrôle de flux(figure 2), la bande passante disponible sur le réseau décroît en fonction du nombre d'émetteurs. Lorsque la capacité de ces émetteurs est supérieure à la bande passante disponible, il y a congestion.

Dans le cas de transmission fiable (i.e. transmission sans perte, FTP par exemple),

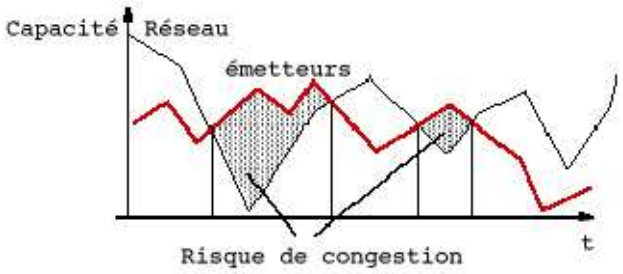

FIG. 2 – *Risque de congestion*

il est nécessaire de retransmettre un paquet perdu, ce qui provoque une augmentation de la latence et introduit de la gigue. De plus, la retransmission des paquets perdus augmente la quantité de données envoyées à travers le réseau. Ce qui peut alimenter l'effet de congestion au niveau du goulot d'étranglement. Le débit utile du réseau s'en voit alors fortement dégradé.

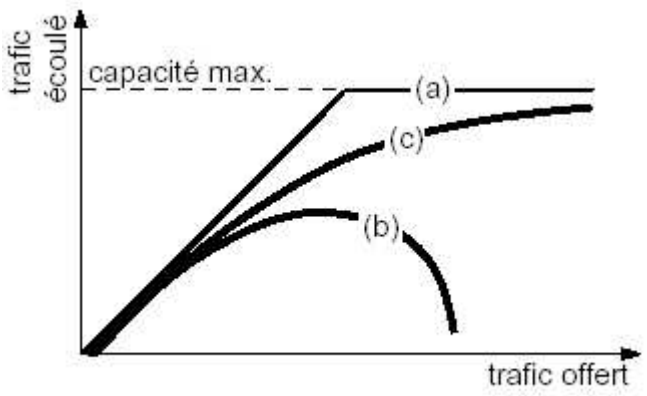

FIG. 3 – *Le débit utile*

Figure 3 : La courbe (a) représente le débit utile optimal. La courbe (b) représente le débit constaté sans contrôle de congestion, les pertes entraînent les retransmissions (qui augmentent l'influence du goulot d'étranglement) entraînant les pertes... La courbe (c) est l'évolution espérée du débit utile avec contrôle de congestion.

Ce problème peut se transposer au niveau système d'exploitation, un ordonnancement classique peut faire échouer la continuité d'un média. En effet, s'il y a trop de processus à ordonnancer, l'application multimédia peut être mise en attente trop longtemps et vider son buffer de sortie qui produira un blanc au moment de jouer

le flux. Ce type de problème ne sera pas abordé dans cette étude.

2.1 TCP pour le multimédia ?

TCP est le protocole le plus utilisé d'Internet (95% du trafic). Ce protocole s'adapte à la bande passante disponible du réseau en diminuant son débit lorsqu'une perte est détectée. La stratégie d'adaptation est AIMD (Additive Increase, Multiplicative decrease) et permet un contrôle de congestion vis-à-vis du réseau. Les figures ci-dessous montrent le fonctionnement du contrôle de congestion de TCP [1] :

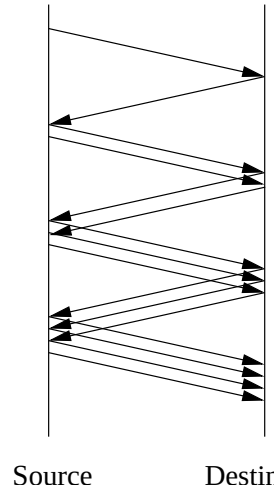

FIG. 4 – *L'augmentation linéaire de la fenêtre de contrôle de congestion après le Slow Start*

L'*Additive increase* (figure 4) : Le récepteur envoie l'acquittement estampillé par le numéro du dernier paquet reçu. Si aucun paquet n'a été perdu, l'émetteur envoie un paquet de plus que précédemment avant d'attendre le prochain acquittement suivant. Cette étape suit le *Slow Start* : augmentation exponentielle de la taille de la fenêtre (sa taille est multipliée par deux à chaque pas) jusqu'à un certain seuil auquel TCP passe à l'*additive increase*.

Le *Multiplicative Decrease* (figure 5) : Dès qu'une perte est constatée, l'émetteur divise le seuil par deux, il repart de 0 en doublant son débit par deux jusqu'au nouveau seuil ou TCP repasse au *Slow Start*.

TCP choisira ensuite le débit minimum entre celui déduit de l'algorithme de contrôle de congestion et celui déduit de l'algorithme de contrôle de flux.

[1] [Ste95] : une description détaillée du fonctionnement de TCP.

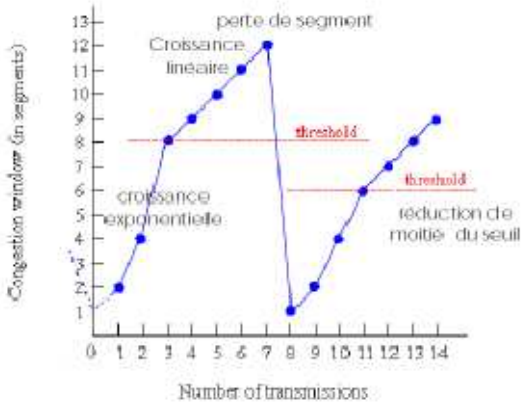

FIG. 5 – *Diminution exponentielle du seuil*

Afin de minimiser les effets de la congestion en période de charge du réseau, il est souhaitable qu'au niveau de chaque site, la gestion du débit soumis soit la moins agressive possible (Les oscillations de TCP sont brutales). Ayant des contraintes temporelles fortes, les applications multimédia doivent activement participer au contrôle de congestion (être adaptative) afin d'obtenir un service optimal de la part du réseau pour tous ses utilisateurs.

L'aspect distribué des applications multimédia pose des contraintes supplémentaires, il faut pouvoir gérer les communications multicast de la façon la moins coûteuse possible en terme de nombre de messages et quantité d'informations envoyées sur le réseau.

Les protocoles dit traditionnels (UDP et TCP) ne sont pas de bon candidats pour une transmission multimédia. En effet, UDP n'est pas fiable (pour le multimédia, ce problème reste marginal tant que le taux d'erreur est faible), mais surtout ne fournit pas le numéro de séquence des paquets ni l'estampillage temporel, sans oublier qu'il ne participe pas au contrôle de congestion, il n'est donc pas possible de l'utiliser abruptement. TCP est un protocole de transport adaptatif et fiable. Seulement, son algorithme de contrôle de congestion [APS99] n'est pas adapté : il est réactif et provoque la congestion pour finalement se rétracter et diminuer le débit (en effet le débit est augmenté linéairement jusqu'à ce qu'une perte soit constatée). C'est une approche agressive qui ne prend pas en compte les variations des délais de bout en bout (qui peuvent nous donner des information plus fine sur le réseau). Nous pouvons ajouter que les facteurs additif et multiplicatif étant constants, le débit oscille autour du débit optimal sans converger vers celui-ci, ce qui serait souhaitable afin d'exploiter au mieux la capacité du réseaux.

2.2 Quelles sont les solutions proposées ?

Une alternative, TCP Vegas [BP95] propose de modifier TCP pour le rendre plus performant. TCP Vegas à été implémenté au dessus d'une version de TCP existante (TCP Reno [Jac88]) et améliore le débit de 37 % à 70 % . Dans cet implémentation, trois techniques ont été testées, la première consiste à choisir la date à laquelle une retransmission doit avoir lieu, la deuxième vise à anticiper la congestion du réseau et d'y ajuster son débit d'émission, tandis que la troisième modifie le slow start (phase initiale de l'algorithme de contrôle de congestion de TCP) tout en essayant de trouver la bande passante disponible sur le réseau. La première technique est basée sur le paradigme suivant : le client peut notifier la perte d'un paquet (expiration d'un temps d'attente) alors que celui-ci est encore sur le réseau. Dans ce cas, une attente un peu plus longue peut suffire pour finalement recevoir l'acquittement du paquet. Les mesures effectuées dans [BP95] montrent une économie importante de retransmissions.

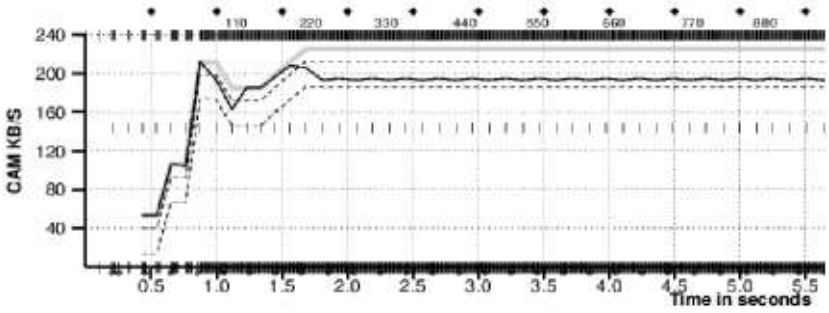

FIG. 6 – *Contrôle de congestion de TCP Vegas*

Sur la figure 6, L'algorithme d'anticipation de congestion est basé sur la variation du RTT (délai aller retour).

- Les traits verticaux représentent chaque RTT. C'est à ce moment que l'algorithme de contrôle de congestion prend une décision. Cela permet de réagir directement en fonction de l'état du réseau.
- La ligne grise épaisse est le débit prévu, i.e. le débit auquel on peut envoyer tout le contenu de la fenêtre en un RTT.
- Le ligne la plus foncée montre le débit actuel (calculé sur le temps du précédent RTT).
- Les lignes en pointillés sont des seuils permettant de réguler la taille de la fenêtre de contrôle de congestion.

En comparaison, Vegas est plus performant 92 % du temps que TCP Reno, l'équité entre plusieurs connexions sur une même bande passante est comparable sur les deux versions. D'autre travaux d'expérimentation de TCP Vegas ont été

publiés [ADLY95, HBG00] concluant sur les mêmes résultats que les auteurs du protocole original [BP95].

D'autres variantes de TCP ont été proposées pour encore optimiser ses performances en vue de transporter du multimédia. Dooly [DGS98] propose de diminuer la latence introduite par les acquittements retardés. Le principe est simple, après réception de n paquets, alors il faut calculer l'arrangement de sous ensembles de paquets (le dernier paquet du sous-ensemble sera acquitté) tel que la latence introduite par les acquittements retardés soit minimale tout en essayant de maximiser le nombre de paquets acquittés en une fois. TLTCP [Muk00] est encore une autre variante de TCP qui elle est destinée aux applications multimédia. TLTCP utilise la retransmission mais n'est pas un protocole fiable. Il s'agit en fait de mesurer la latence de bout en bout et de réémettre seulement si le paquet à une chance d'arriver dans les délais de jouabilité du coté du récepteur. L'algorithme de contrôle de congestion est identique à celui de TCP. Dans la même direction, nous pouvons citer [PFOC00] qui propose un protocole de retransmission sélective. L'application doit spécifier (ou typer) les paquets devant impérativement être reçus et ceux pouvant être perdus. Par exemple, considérant un flux MPEG 2, une frame intermédiaire peut être perdue mais pas une frame complète. La fiabilité du transport passe par la retransmission (comme TLTCP [Muk00] une retransmission a lieu si elle a une chance d'être utile). Nous pouvons aussi citer les efforts de TCP-Real [ZT01] afin de modifier TCP pour fonctionner sur un réseau sans fils. Les erreurs dues aux liens sans fil peuvent être perçues comme une congestion par TCP, il faut donc limiter cela tout en prenant en compte le caractère asymétrique des liens en terme de bande passante.

2.3 L'impasse ?

Malgré tous ces efforts, TCP reste TCP et n'est pas parfaitement approprié au transport de données temps réel. En effet, TCP est un protocole fiable basé sur la retransmission. Le simple fait de retransmettre un paquet augmente de deux fois la latence de bout en bout et cela si tout ce passe bien durant la retransmission. Cela ne répond pas aux contraintes des flux multimédia interactifs. Ces flux sont exigeants principalement sur la latence et la gigue. L'exigence sur les pertes est motivée par la qualité du résultat que l'on veut obtenir chez le récepteur, un paquet perdu pouvant être réparé (le taux de perte doit quand même rester au dessous d'un seuil maximum, voir paragraphe 5). Il s'agit donc de faire un compromis au niveau du protocole de la couche application entre les pertes que l'on veut pouvoir tolérer et la latence désirée.

Afin de répondre à ces besoins, nous pouvons distinguer plusieurs approches. D'une part l'approche orientée réseau voit émerger deux écoles, la première propose de réserver les ressources directement au coeur de l'infrastructure du réseau

même[ZDE93] ou alors d'intégrer des mécanismes de priorités sur les paquets ordonnancés par les routeurs[RLS99]. La deuxième place ses efforts au niveau de la couche application en ré-écrivant des protocoles de transport intégrant un contrôle de congestion grâce à une adaptation la plus fine possible du débit d'émission. L'autre approche plutôt orientée système réparti englobe les deux approches précédentes. Elle consiste à utiliser des environnements logiciels tels que les différents ORB temps réel[FWDC+00] afin que ceux-ci prennent en charge tous les efforts nécessaires à la satisfaction des contraintes temporelles[FSL97]. Ces applications visent principalement à réduire le temps de développement et sont généralement destinées aux applications multimédia distribuées mais non interactives (à cause des fortes latences introduites). C'est pour cette raison que nous n'explorerons pas cette voie dans ce document.

De par l'aspect distribué d'Internet et des réseaux interconnectés, la mise en place de mécanismes de réservation de ressources ou de priorités est difficile. En effet, leur garantie de fonctionnement suppose une participation collective de tous les sous-réseaux de bout en bout, ce que les réseaux actuels n'intègrent que rarement. De plus, le traitement privilégié de certains paquets a un coût en terme financier (des équipement plus perfectionnés sont nécessaires), et cela contrairement à un transport de données dit classique. Prenons l'exemple d'un site Web voulant diffuser de la radio, un mécanisme de réservation payante devient complexe si l'on considère l'hétérogénéité des auditeurs quant à la bande passante de leur réseau, voir la monnaie utilisée. Il n'est pas impossible non plus qu'un auditeur préfère avoir un service gratuit avec une qualité plus faible (surtout s'il ne se connecte pas aux heures de congestion du réseau). Toutefois, ces approches orientées réseau ont une qualité unique vis-à-vis des autres citées ci-dessus, elles *garantissent* une qualité de réception, tandis que les autres techniques font du mieux qu'elles peuvent. Dans cette étude, nous nous focaliserons particulièrement sur l'approche architecture de communication qui vise à ré-écrire les protocoles de la couche application [2] (paragraphe 4).

[2]Nous allons présenter RTP qui est un protocole de session(en particulier grâce à son protocole associé RTCP qui fourni l'estampillage) et de présentation(les données transporté sont typées par des Payload)

3 Transport de média et standards

Les flux de données temps réel tels que les flux audio et vidéo, comme la téléphonie sur IP ou la vidéo-conférence possèdent certaines contraintes qui ne sont pas prises en compte dans les protocoles de base d'Internet.

Le séquencement temporel : les paquets doivent être réordonnés temporellement et en temps réel par le récepteur. Si un paquet n'est pas reçu à temps alors il doit être détecté et compensé sans retransmission.

La synchronisation intra-média : synchronisation de bout en bout. Par exemple si l'on ne transmet pas de données pendant un silence, il faut reconstruire ce silence au niveau du récepteur.

La synchronisation inter-média : il faut synchroniser les flux de différents média entre eux. Par exemple synchroniser le son avec le mouvement des lèvres.

Le type des données transportées (Payload) : le typage est utilisé pour les modifications dynamiques des données transportée dans le flux en vu d'ajuster les taux de transfert en fonction de la charge du réseau ou de la capacité du récepteur.

Le format des messages est contraint par la structuration propre du média transporté : La vidéo et l'audio sont envoyés par unités (Frames) de taille variable selon le format d'encodage. Il est donc nécessaire d'indiquer au récepteur le début et la fin de chacune d'elle.

Nous allons présenter dans ce paragraphe les efforts de la communauté scientifique dans l'élaboration de protocoles visant à aider le concepteur d'application multimédia à implémenter une politique d'adaptation du débit. Les protocoles présentés sont des emballages adéquats permettant d'avoir le maximum d'information sur le lien entre le récepteur et l'émetteur des donnés multimédia.

3.1 RTP et RTCP

RTP [SCFJ98] est un protocole de transport de données temps réel qui vise à aider à satisfaire les contraintes ci-dessus. Il repose sur IP et UDP et fournit des services tels que la reconstruction, la détection de pertes, la sécurité, l'identification du contenu du flux transporté. Pour cela, il met à disposition **l'estampillage temporel** qui sera utilisé pour déterminer quand doit être joué un paquet par rapport à un autre et synchroniser les différents flux (la synchronisation se fait au niveau de l'application). Le **numéro de séquence** sert à réordonner les paquets et à détecter des pertes. L'**identificateur de type de charge** permet de spécifier le format des

données encapsulées. L'application réceptrice sait donc comment jouer les paquets en sortie. Les payloads standards sont définies dans [Sch98]. Un émetteur ne peut utiliser qu'un seul payload à la fois mais peut le changer durant la transmission. L'**identifiant de la source** permet de différencier les flux venant de sources différentes.

Pour plusieurs raisons, RTP fonctionne au dessus de UDP. D'une part, UDP est fait pour le multicast, TCP ne passerait pas à l'échelle à cause des acquittements et réémissions de paquets, d'autre part la fiabilité n'est pas aussi importante que le temps de délivrance des paquets. En pratique, la récupération des paquets perdus et le contrôle de congestion sont implémentés au niveau de l'application grâce au protocole RTCP qui assure un compte-rendu du récepteur. Pour cela, RTCP transmet cinq types de messages que nous présentons ici afin de connaître exactement la sémantique disponible au niveau application(qui implémente le contrôle de congestion si elle le désire !!) :

- *RR* receiver report. Envoyé par le récepteur, il contient le numéro du plus grand paquet reçu, les numéros des paquets perdus, la différence de temps entre l'arrivée des paquets et l'estampillage temporel. Permet de calculer le délai aller-retour (L'estampillage temporel étant relatif et non absolu, il n'y a pas besoin de synchronisation d'horloge entre l'émetteur et le récepteur).
- *SR* sender report. Contient les informations de synchronisation entre les médias, un compteur de paquets et le nombre d'octets envoyés.
- *SDES* Source description items. Information de description des sources.
- *BYE* Message de fin de participation.
- *APP* Option spécifique à l'application.

RTCP étant un support pour le contrôle de congestion, l'identification de la source et la synchronisation entre les médias, il fournit le pilotage de la QoS. Il envoie ses paquets de façon périodique mais ne doit pas dépasser 5 % du trafic de la session. Il faut donc adapter le délai en fonction du nombre de participants.

Nous allons décrire comment les délais aller-retour sont calculés. L'émetteur introduit la date d'émission dans son paquet de contrôle (champ *SR* de RTCP). Le récepteur calcule le temps écoulé entre la réception de l'ancien rapport de l'émetteur et le courant (T_{pass}) et l'inclut dans son paquet de contrôle (champ *RR* de RTCP). Grâce à ce rapport, l'émetteur peut calculer le délai comme le temps passé entre l'émission de ses deux rapports concernés moins T_{pass}.

RTP supporte les notions de *translators* et *mixers* qui peuvent être considérées comme des systèmes intermédiaires. Bien que ces notions compliquent le protocole, le besoin de les utiliser ce fait ressentir lorsque l'on expérimente le multicast audio ou vidéo sur Internet. Un *translator* transmet les paquets reçus avec les mêmes identifiants de source mais peut modifier le contenu des données (changement de payload), peut faire passer le flux de multicast en unicast, convertir le flux d'IP4 à

IP6.Les *mixers* reçoivent d'une ou plusieurs sources, peuvent changer le format des données, combiner les paquets et retransmettre dans un seul flux.

Les protocoles RTP et RTCP fournissent donc un support intéressant à l'application qui l'utilise pour contrôler la congestion du réseau. Grâce au rapport du récepteur, le calcul du délai aller retour, le taux de perte, la synchronisation des médias et la gestion d'une politique d'adaptation vis à vis du réseau [DHT95] sont grandement simplifiés. De plus les *mixers* et *translators* permettent de construire une architecture répartie efficace afin d'assurer le passage à l'échelle sur le réseau. Nous verrons dans la paragraphe 4 les algorithmes exploitant les rapports RTCP du ou des récepteurs afin de gérer le débit de la source.

3.2 RTSP

Real Time Streaming Protocol [SRL98] est un protocole du niveau application permettant de contrôler la délivrance de données ayant des propriétés temps réel comme l'audio et la vidéo à la demande. Il permet de faire des retours arrière, des pauses pendant la lecture d'un média à distance. RTSP peut contrôler plusieurs sessions simultanément de façon synchronisée, il permet donc de choisir le support du média parmi UDP, multicast UDP et TCP ou même RTP.

Ce protocole est fortement orienté vers les transmissions de médias stockés sur disque (messagerie vocale par exemple *VoD*) mais n'est pas adapté aux flux interactif comme les conférences ou même la téléphonie sur IP.

4 Approche Rate Control et TCP friendly, le contrôle du débit

L'Internet actuel est un réseau best-effort sans réservation de ressources et sans garantie de qualité de service (QoS). Les applications multimédia (destinées à fonctionner sur les réseaux best-effort) se voient confrontées à de grandes variations sur la bande passante disponible, la latence et la gigue. Pour que ces applications puissent survivre dans cet environnement hautement dynamique, l'adaptation du débit d'émission grâce au compte-rendu du récepteur s'impose. A l'aide de ce compte-rendu, l'application (ou le protocole de transport utilisé) doit faire un compromis sur la qualité (tolérer un taux de perte plus grand, diminuer la taille de l'écran contenant le film joué, utiliser un échantillonnage de plus faible qualité) afin de ne pas provoquer de surcharge du réseau par une transmission trop massive. Pour l'utilisateur final, il est préférable, en cas de dégradation de la qualité, de la dégrader le plus harmonieusement possible. L'adaptation vise une utilisation optimale du réseau, cependant il faut tenir compte des autres flux de données transitant et donc fournir un partage équitable de la bande passante. De plus l'adaptation est vitale pour la survie du réseau, l'Internet actuel est stable grâce au contrôle de congestion de TCP (qui représente 95% du trafic), la sur-utilisation d'applications multimédia sans mécanisme d'adaptation provoquerait de nouvelles congestions collapses qui casseraient la stabilité du réseau.

Nous allons dans cette section étudier certains de ces mécanismes d'adaptation. Notre objectif est de comprendre comment faire fonctionner une application multimédia interactive à plus de deux intervenants. Nous allons donc différencier les mécanismes orientés unicast et les mécanismes multicast. Certains d'eux sont construits pour les deux types de communication, ce qui sera discuté plus tard.

L'étude présentée ici section 4.1 et 4.3 va se focaliser sur les mécanismes dédiés aux réseaux best-effort. Nous pouvons toutefois citer quelques travaux intéressant comme [Gia01] qui vise à élaborer un algorithme de contrôle de congestion pour les réseaux de mobiles caractérisés par de forts taux d'erreurs sur les paquets (qui sont interprétés par TCP comme de la congestion). Des paquets (avec une priorité minimale) sont envoyés en éclaireur sur le réseau afin de mesurer les ressources disponibles. [DHLB98] propose un protocole de transport pour les communications multimédia sur réseau à priorité. Ce protocole propose de séparer les flux de données fiables et de données non fiables afin de leur appliquer un traitement différent. Cela permet dans le cas de transfert de flux MPEG 2 par exemple d'utiliser un flux fiable pour les trames entières et un flux non fiable pour les trames intermédiaires.

Dans [ACG+97], un constat est fait sur TCP : lors d'une transmission, tous les

paquets sont reçus dans l'ordre de leur émission, ce qui n'est pas toujours nécessaire. Les auteurs proposent un nouveau type de service *partially-ordered, partially-reliable*. [RDSD99] utilise ce type de service afin de construire un protocole adaptatif pour un serveur de vidéo. Les réseaux de pétris sont utilisés pour spécifier l'ordre partiel des séquences vidéo. La fiabilité est obtenue en envoyant un paquet correcteur d'erreur si une perte est détectée (cf paragraphe 5).

4.1 Mécanismes unicast

4.1.1 Un mauvais exemple

Le protocole VDP proposé dans [CTCL95] est utilisé dans le navigateur Vosaic pouvant intégrer de la vidéo dans une page Web. Le type d'encodage de la vidéo est spécifié dans l'URL de la même façon que lorsque l'on spécifie le type de protocole (FTP, HTTP...). VDP propose deux types de canaux pour le transport du média, un canal fiable pour transporter les donnés de contrôle (play, pause..) et un canal pour le flux multimédia. Le client envoie un compte-rendu toutes les 30 images. Ce compte-rendu contient le nombre d'images n'ayant pas été jouées par manque de CPU et le nombre d'images perdues sur le réseau. Le serveur peut donc adapter son débit. Il dégrade la qualité si 15% de pertes sont détectées et l'augmente si moins de 5% de pertes sont constatées. Seules les images importantes sont retransmises. Tenir compte du nombre final de paquets joués est une approche intéressante. Cependant, l'augmentation du débit continue tant que des pertes sont constatées mais restant inférieures à 5% des paquets du trafic. Nous pouvons facilement imaginer une connexion TCP qui partagerait la bande passante avec VDP, TCP diminuerait son débit d'émission dès qu'une perte est détectée, tandis que VDP augmenterait son débit. En d'autre terme, VDP risque de ¨ manger¨ la bande passante et créer sa propre congestion collapse (en cas d'utilisation à grande échelle bien sûr). Les comptes-rendus du récepteur sont envoyés toutes les 30 images. En cas de réduction du débit, le délai entre deux comptes-rendus augmente, la précision des mesures du réseau perd en qualité. Une mesure plus fine serait préférable dans cette situation afin de réagir plus rapidement à une libération de bande passante sur le réseau.
Voici un tableau récapitulant la modification du débit en fonction des pertes :

Pertes	de 0 à 5%	de 5 à 15%	de 15 à 100%
Débit	$\nearrow$	$=$	$\searrow$

4.1.2 Les TCP-like, approche réactive

Nous allons présenter dans ce paragraphe quelques protocoles et particulièrement leurs algorithmes de contrôle de congestion. Le principe ici est d'imiter le compor-

tement de TCP afin d'être équitable tout en le modifiant pour que le protocole construit convienne au transport de données multimédia.

TFRCP, TCP-Friendly Rate Control Protocol

Afin de s'assurer de l'équité du partage de la bande passante, la méthode la plus évidente est d'utiliser le même algorithme de contrôle de congestion que TCP mais en multipliant par deux le débit si aucune perte n'est détectée (slow start de TCP). TFRCP [PKTK98] est construit selon ce principe. Chaque paquet est acquitté afin de permettre à l'émetteur de calculer le RTT et de calculer les pertes. Un vecteur de 8 bits permet de confirmer ou non l'arrivée d'un paquet (cela évite une mauvaise interprétation d'un acquittement perdu). L'émetteur recalcule son débit à intervalle de temps régulier de façon similaire à TCP. TFRCP n'intègre aucun mécanisme de récupération d'erreur. Par implémentation, les auteurs présentent ce protocole comme équitable vis à vis de TCP, cependant doubler le débit d'émission en cas de sous-charge du réseau est très agressif et risque de provoquer rapidement des pertes. Les mesures effectuées par les auteurs montrent que le débit de TFRCP oscille beaucoup au court du temps.

SCP, Streaming Control Protocol

SCP [CPW97] utilise le même principe, se servir du contrôle de congestion de TCP pour assurer le partage équitable de la bande passante avec d'autres connexions. L'algorithme de slow start est identique afin de découvrir rapidement la bande passante disponible. Les paquets perdus ne sont pas retransmis. Chaque paquet est acquitté afin de mesurer les délais aller-retour. L'augmentation du nombre de paquets dans la fenêtre d'émission est fonction du débit des acquittements reçus, du RTT mesuré. Cela permet de suivre les fluctuations du réseau. Cependant, on ne pourra constater d'augmentation du débit d'acquittements que si SCP augmente lui aussi son débit. C'est un problème du type ¨ l'oeuf et la poule ¨ . Pour remédier à cela, l'augmentation du débit est aussi fonction de la variation du nombre de paquets sur le réseau. S'il y a plus de paquets sur le réseau que précédemment et que l'on ne constate pas de perte alors on peut continuer à augmenter le débit de façon similaire. La diminution du débit se fait de moitié lorsqu'une perte est détectée. Les résultats montrent que plusieurs sessions SCP ne partagent pas de façon équitable la bande passante. En fait lorsque SCP devient à peu près stable, il est difficile pour un autre flux SCP d'obtenir autant de bande passante que lui. De plus en cas de dégradation du débit, on observe de fortes fluctuations à cause de la réduction par deux de la taille de la fenêtre d'émission. La qualité s'en voit fortement dégradée d'un coup.

Vic, un outil de vidéo-conférence

Busse, Deffner et Schulzrinne [BDS96] mettent en place un mécanisme d'adaptation de débit du flux pour le logiciel de vidéo conférence Vic. L'adaptation (basée sur RTP) se fait sur détection de pertes lors du compte-rendu du récepteur. L'objectif ici est d'avoir la meilleure qualité possible, donc la bande passante optimale. Les pertes causées par les erreurs de transmissions étant considérées comme faibles par rapport aux pertes de paquets dans les files d'attente des routeurs, les auteurs considèrent toutes les pertes comme étant causées par la congestion. Afin de déterminer l'état du réseau, les auteurs utilisent des paliers sur le taux des pertes détectées. Les seuils sont déterminés par expérimentation. Sous 2% de pertes détectées, le réseau est considéré comme UNLOADED (sous chargé), entre 2 et 4 % LOADED (chargé) et au dessus de 4% comme CONGESTED (en période de congestion). Une fois l'état du réseau établit, l'adaptation se fait de façon similaire à TCP avec un algorithme AIMD. On augmente linéairement le débit si l'état est UNLOADED, on le diminue exponentiellement si l'état est CONGESTED. Nous pouvons noter que la modification du débit ne se fait pas à l'aveugle, des seuils minimum et maximum pour la bande passante peuvent être proposés afin de ne pas dépasser des valeurs aberrantes. Comme VDP, ce mécanisme n'est clairement pas équitable avec TCP. Nous verrons dans la section mécanisme multicast comment adapter cette méthode à un groupe de récepteurs.

Algorithmes binomiaux

Bansal et Balakrishnan [BB00] ont généralisés le fonctionnement du contrôle de congestion de TCP à travers les algorithmes de contrôle de congestion appelés *algorithme binomiaux*. Ces algorithmes sont *AIMD* (additive increase, multiplicative decrease) comme TCP. Une augmentation du débit se fait de façon inversement proportionnelle à la taille de la fenêtre d'émission en utilisant un paramètre k (fonction I). Pour une diminution du débit, le paramètre est l (fonction D).

$$I : w_{t+RTT} \leftarrow w_t + \alpha / {w_t}^k; \alpha > 0 \tag{1}$$

$$D : w_{t+\delta t} \leftarrow w_t - \beta {w_t}^l; 0 < \beta < 1 \tag{2}$$

Où w_t est la taille de la fenêtre d'émission du contrôle de congestion, α est le facteur additif, RTT le délai aller-retour, δt le temps de détection d'une perte depuis le dernier changement de taille de la fenêtre, β le facteur multiplicatif.Pour TCP, $w_{t+RTT} = w_t/2$ en cas de perte donc $\beta = 1/2$. $\alpha = 1$ car $w_{t+RTT} = w_t + 1$ après le Slow Start (et qu'aucune perte n'est constatée). Les paramètres $k = 0$ et $l = 1$ modélise donc son comportement (équation (1) et (2)).
Les auteurs montrent qu'un algorithme binomial cohabite correctement avec TCP si

$k + l = 1$. D'après eux, pour un taux de perte et des conditions identiques, les algorithmes binomiaux obtiennent le même débit que TCP. Les expérimentations se font à travers deux algorithmes binomiaux, IIAD (Inverse increase additive decrease avec $k = 1$, $l = 0$) et SQRT (SQuare RooT avec $k = l = 0, 5$). D'après les tests effectués, IIAD ne passe pas à l'échelle d'un réseau surchargé. Cependant, l'équité avec TCP est conservée puisque k+l=1. Selon les mesures, les débits d'émission ne semblent pas osciller. Le mécanisme proposé ici représente une contribution intéressante au domaine des algorithmes d'adaptations. Ceux présentés ici permettent de réagir moins violemment que TCP le fait. Cependant, cette approche reste une approche réactive qui va provoquer la congestion pour découvrir la bande passante maximale disponible. Ce protocole à été implémenté dans le logiciel d'audio-conférence Vat[JM92].

4.1.3 L'approche par équation, une approche prédictive

Comme dans le paragraphe précédent, nous allons présenter des algorithmes de contrôle de congestion. Ici le calcule des facteurs AIMD est basé sur une modélisation du débit de TCP (sur des estimations de son comportement) mais aussi quelquefois sur une estimation de l'état du réseau (de la bande passante du goulot d'étranglement).

Un modèle du comportement de TCP

Afin de mieux comprendre le fonctionnement de TCP, Mahdavi et Floyd [MF97] proposent une modélisation de l'effet des pertes sur une connexion TCP. L'effet des pertes de paquets sur une connexion TCP est donné par la formule suivante :

$$Bandepassante = C\frac{MTU}{RTT\sqrt{Loss}} \tag{3}$$

où MTU est la taille des paquets envoyés, RTT le temps d'un aller-retour, $Loss$ les pertes constatées par la connexion (mise a jours le plus souvent possible) et C une constante choisie entre 0,7 et 1,3. Lors d'une augmentation de débit, les auteurs conseillent de ne pas dépasser la bande passante calculée par cette formule. Par simulation [SK97] pour $C = 1, 22$ et si les pertes restent inférieures à 5% un algorithme utilisant cette formule est TCP friendly. En contrepartie, la formule ne simule pas correctement TCP si les pertes passent au dessus de 15% . Dans cette situation, les retransmissions commencent à jouer de façon significative sur la latence.

TFRCP nouvelle version

Suite à ces travaux, une nouvelle version de TFRCP est proposée dans [FHPW00] utilisant une équation plus fine que l'équation (3). L'utilisation de cette équation

permet de ne pas provoquer la congestion pour trouver la bande passante disponible. Les expérimentations montrent que TFRCP nouvelle version ne ¨ mange¨ pas la bande passante d'autres connexions TCP et que les oscillations du débit dues aux calculs de l'algorithme d'ajustement sont harmonieuses (on ne constatera pas de dégradation abrupte de la qualité).

TFRP, TCP-friendly rate based transport protocol

Tan et Zakhor [TZ99] utilisent aussi la formule (3) afin d'implémenter un logiciel de transmission vidéo. L'équité avec TCP est le principal objectif. Pour cela les auteurs construisent un protocole TFRP qui joue sur la taille des paquets envoyés. La synchronisation entre plusieurs sources devient complexe à cause de la longueur variable (en temps) du média contenu dans le paquet. On peut imaginer de garder un temps constant sur les paquets en modifiant la qualité de l'encodage à chaque modification de la taille. Il faut disposer d'un encodeur à taux dynamiques et à grain fin.

Un modèle de mesure de la congestion du réseau

Afin de comprendre plus finement le comportement de TCP, Bolot dans [Bol93] présente une étude sur le comportement des pertes et des délais des paquets sur le réseau. L'auteur mesure le temps aller retour à l'aide de paquets UDP de petite taille et envoyés à intervalles réguliers. De cette façon, il est possible de mesurer la gigue (la variation de la latence) et d'en déduire la charge du réseau et son évolution. Ce résultat est important puisque il permet d'estimer la congestion future et donc dans le cas d'une application multimédia permet de diminuer le débit avant constatation de la congestion.

LDA+, enhanced loss-delay based adaptation

Sisalem et Wolisz [SW00b] vont utiliser ce résultat pour construire LDA+, un algorithme d'adaptation basé sur RTP. Grâce aux mesures effectuées par le récepteur et transportées par RTCP, l'émetteur va réguler le trafic par rapport aux pertes, aux délais et à la bande passante disponible de façon dynamique. Le délai aller-retour est calculé comme décrit dans la paragraphe 3.1. Afin de calculer la charge du réseau selon la méthode de Bolot [Bol93], les auteurs envoient des paquets servant à calibrer la bande passante du réseau. Pour cela, il rajoutent (grâce au champ *APP* de RTCP) dans les informations de contrôle de l'émetteur le numéro de l'enquête sur la gigue et le nombre de paquets n servant à celle-ci. Les n paquets sont envoyés aussi rapidement que le permet le système de l'émetteur. Le récepteur peut calculer

la bande passante du goulot d'étranglement R par :

$$R = \frac{TaillePaquetInvestigation}{TempsEntreReceptionDeDeuxPaquets} \tag{4}$$

Sachant qu'il est nécessaire d'utiliser des filtres sur les informations à cause des pertes éventuelles de paquets d'investigation.

Ces information calculées, l'algorithme LDA+ peut commencer à travailler. Il est de type AIMD. En période de pertes, LDA+ calcule la bande passante disponible r_{TCP} à l'aide du modèle analytique de Padhye et al. [PKTK98] (calcule de la bande passante équitable avec TCP) :

$$r_{TCP} = \frac{M}{t_{RTT}\sqrt{\frac{2Dl}{3}} + t_{out}min(1, 3\sqrt{\frac{3Dl}{8}})l(1+32l^2)} \tag{5}$$

où M est la taille des paquets, l le pourcentage de pertes, t_{out} est la valeur du temps d'attente (timeout) de retransmission de TCP, t_{RTT} est le délai aller-retour et D le nombre de paquets acquittés en une fois. Si aucune perte n'est détectée, cette formule permet de plafonner l'augmentation du débit.

Nous allons décrire comment le facteur additif est calculé. Ce facteur est le minimum entre A_{add_m}, A_{exp} et A_{TCP}. A_{exp} est déterminé par une fonction convergeant vers la bande passante du goulot d'étranglement (pour éviter d'arriver à la congestion). A_{TCP} est obtenu par une fonction du délai estimant l'augmentation de la fenêtre d'émission d'une connexion TCP (A_{TCP} sert à être équitable vis à vis de TCP) et enfin A_{add_m} est obtenu par une fonction de R (équation (4)). Cette stratégie permet une approche complètement proactive sur la congestion sans la provoquer.

Le facteur multiplicatif est $1 - \sqrt{l}$, l étant les pertes. Cette modification du débit sera faite si elle est plus grande que r_{TCP} (équation (5)). Cela permet de réagir en fonction de TCP (avec r_{TCP}) où en fonction des pertes détectées.

Les auteurs ont testé cet algorithme par implémentation sur un réseau à deux routeurs étant considérés comme le goulot d'étranglement. L'équité entre un nombre égal de flux FTP et LDA+ (jusqu'a 80 flux de chaque) s'établit autour de 80% (la somme des débits des connexions TCP divisé par ceux des connexion LDA+). Les mêmes résultats sont obtenus pour FTP, WWW et LDA+ en même temps sur le goulot d'étranglement. Nous pouvons remarquer que le protocole construit avec RTP et LDA+ s'adapte de façon fine à l'état du réseau. Dans le cas d'une application multimédia, les fréquences d'encodages (l'échantillonnage) ne sont pas aussi fine, cela veut dire que LDA+ n'est pas applicable directement. C'est pourquoi Sisalem et Wolisz [SW01] proposent un modèle permettant à une application multimédia de spécifier la Qualité de service désirée (QoS). Ils proposent aussi le Framework CTFAF qui fera le lien entre les spécifications et LDA+. Les paramètres pris en compte sont :

- Bornes minimum et maximum sur les débits acceptés par l'application. (Si le réseau ne peut satisfaire ces contraintes alors l'application peut prendre la décision d'interrompre la communication)
- L'adaptation granulaire pour modifier le type d'encodage du média lors d'une adaptation du débit (Par exemple passer d'un encodage PCM à un encodage MP3)
- Agressivité de la modification du flux pour avoir une dégradation ou amélioration harmonieuse de la qualité. (On peut spécifier par exemple pour de la vidéo de ne pas passer de 24 frames à 0 mais 12 minimum)
- Tolérance aux pertes : le taux de perte toléré par le média.

Pour répondre à ces contraintes, les auteurs introduisent le concept de Bande Passante Virtuelle qui est calculée selon ces paramètres. L'encodage et le débit choisis est un compromis entre les valeurs obtenue par LDA+ et les contraintes spécifiées. Ce compromis fait la balance entre une dégradation disgracieuse (de forte variation de qualité) et les variations proposées par LDA+.

RAP, Rate Adaptation Protocol

RAP [RHE99] est un autre algorithme de contrôle de débit basé sur le fonctionnement AIMD de TCP. Bien qu'il soit CPU-intensif pour un serveur d'adapter l'encodage du média en fonction des clients actifs, RAP fonctionne avec un encodeur de ce type. La source envoie les paquets de données avec un numéro de séquence. Chacun d'entre eux est acquitté par le récepteur (avec redondance sur les huit paquets précédents pour éviter au maximum la perte des acquittements). Cela permet de calculer (au niveau de la source) le RTT et le taux de pertes. L'adaptation se fait de la manière suivante :

- fonction de décision : Si aucune congestion n'est détectée alors augmenter le débit de façon *périodique*. Dès que la congestion est détectée, *immédiatement* baisser le débit (la congestion est détectée dès qu'une perte est constatée)
- l'algorithme AIMD : L'augmentation du débit se fait par pas de $\alpha = \frac{TailleDesPaquets}{gigue^2}$ ce qui estime la bande passante disponible. En cas de diminution, le débit est divisé par 2 .
- La fréquence de décision : Le changement du débit n'est pas décidé régulièrement. Le délai doit être plus grand qu'un RTT pour éviter un comportement trop impulsif. En fait le délai entre deux prises de décision est similaire au temps que prendrait TCP pour envoyer une fenêtre pleine.

Les auteurs précisent que comme TCP , RAP n'est pas équitable avec les flux ayant un RTT plus grand.

4.2 Conclusion sur les mécanismes unicast

Nous pouvons distinguer deux approches, l'approche réactive et l'approche prédictive. Le principal atout de l'approche réactive est une équité évidente avec TCP (qui représente la grande majorité des flux sur l'Internet). Cependant, cette approche hérite des défauts de TCP car elle copie son comportement (on constate de grandes variations du débit et l'adaptation à l'état du réseau n'est pas optimale puisque l'on provoque la congestion pour la guérir). L'approche prédictive tente de modéliser TCP et l'état du réseau pour réagir en fonction des estimations déduites. Malgré le caractère approximatif des modèles utilisés, les mesures montrent une adaptation fine au réseau et une équité acceptable à TCP. Cette approche ne provoque pas la congestion car une augmentation du débit sera fonction des estimations du réseau et du comportement de TCP.

4.3 Les mécanismes Multicast

Nous allons présenter dans ce paragraphe les mécanismes d'adaptation du débit pour un environnement à plusieurs émetteurs et/ou plusieurs récepteurs. L'adaptation orientée émetteur consiste à appliquer directement les protocoles présentés dans le paragraphe précédent à plusieurs récepteurs. L'adaptation orientée récepteur utilise les adresses IP multicast pour émettre un flux à plusieurs débits afin que le récepteur s'abonne à celui qui lui correspond le mieux. Cette approche, pour passer à grande échelle suppose la présence des récepteurs et émetteur(s) sur un réseau supportant le routage multicast (par exemple MBone).

4.3.1 L'adaptation orientée émetteur

MICE, un logiciel d'audio-conférence

Lors d'expérimentations d'une politique d'ajustement du débit sur le logiciel d'audio-conférence MICE, Bolot [BVG96] constate que la mauvaise qualité perçue est généralement l'effet de la mauvaise qualité des micros qui n'offrent qu'un débit constant en sortie. Afin de pouvoir choisir la fréquence d'encodage optimale par rapport au débit souhaité, il va utiliser plusieurs encodeurs en même temps. Cela permet de choisir à la volée le débit souhaité. Pour minimiser l'effet des pertes sur la qualité d'écoute, il va rajouter de la redondance dans les paquets pour pouvoir réparer les pertes directement chez le récepteur (voir paragraphe 5). Ce mécanisme n'ajoutant pas une redondance constante dans le flux, il sera directement combiné avec le mécanisme d'adaptation du débit (i.e. quelle quantité de redondance peut-on mettre dans le flux en fonction du débit choisi ?). Pour obtenir un compte rendu du récepteur, Bolot utilise RTP et RTCP.

La première tache du mécanisme de contrôle est de mesurer l'état de chacun des récepteurs. Ensuite, on choisit la plus petite mesure de QoS (parmi les récepteurs) supérieure à 90% (moins de 10% de pertes) et on essaie de maintenir le taux de pertes du récepteur correspondant entre 1% et 10% . Le seuil de 90% est choisi de façon arbitraire. Dans cette méthode, les récepteurs ne sont pas tous logés à la même enseigne : ceux ayant un taux de perte supérieure à 90% sont exclus du mécanisme. Ces participants sont ceux dont la bande passante est la plus faible, pourtant l'émetteur ne va pas s'adapter selon eux mais selon ceux qui observent un taux de pertes inférieure aux pertes du seuil prévu. On peut donc globalement voir deux clans : les abandonnés qui constateront beaucoup de pertes et dont la qualité n'ira probablement pas en s'améliorant puisque l'émetteur ne s'adapte pas en fonction d'eux et les privilégiés qui sont contraints de suivre le plus mauvais d'entre-eux.

Une fois que le récepteur sur lequel l'adaptation va se faire est choisi, l'algorithme d'adaptation est AIMD avec des facteurs multiplicatifs et additifs tenant compte de la taille ajoutée ou enlevée par la redondance.

4.3.2 L'approche adaptation par le récepteur

RLM, Receiver-driven Layered Multicast

Comme nous l'avons vu dans la section précédente, l'adaptation orientée émetteur n'est pas efficace pour les environnements multicast à récepteurs hétérogènes. En effet la plupart de ces mécanismes sont issus des résultats obtenus dans le domaine de la transmission de données multimédia en unicast. Les exigences conflictuelles des récepteurs en terme de bande passante ne peuvent être satisfaites simultanément avec l'émission d'un unique flux, en effet un récepteur peut demander une meilleure qualité car il ne constate pas de pertes alors qu'un autre fera l'inverse. Comment choisir sans pénaliser l'un ou l'autre ? Sans remettre en cause l'intérêt évident des mécanismes unicast, McCanne et Jacobson dans [MJV96] proposent un nouveau mécanisme d'adaptation multicast orienté récepteur (c'est le récepteur qui va choisir à quel débit il va recevoir le flux). Ce mécanisme (RLM pour Receiver-driven Layered Multicast) combine un encodage à différents taux produit par l'encodeur et une émission de chacun des flux résultant sur une adresse IP multicast différente. Ce mécanisme est complètement orienté récepteur, en effet ce sont les récepteurs qui construisent l'arbre de distribution multicast en fonction de leur intérêt pour un flux donné sans que l'émetteur n'est à se préoccuper de qui désire recevoir le flux.
Les auteurs définissent deux façons de transmettre les flux. L'approche émission simultanée où chaque flux est indépendant l'un de l'autre, seules les fréquences d'encodages sont différentes sur chacune des émissions. L'approche émission cumulative où chaque flux possède les données complémentaires du flux de qualité directement inférieure. C'est l'approche cumulative qui sera retenue dans [MJV96] tout en sachant que l'approche émission simultanée n'est pas incompatible avec RLM. L'avantage

de cumuler les flux est un gain évident de bande passante sur le réseau puisque l'on évite la redondance d'informations entre les flux.

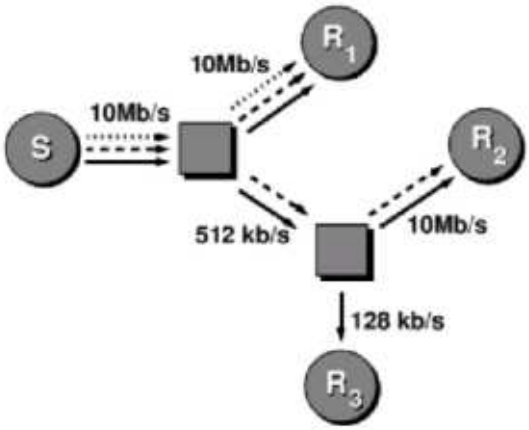

FIG. 7 – *Architecture de RLM*

La figure 7 montre le fonctionnement de RLM. S est l'émetteur et les R_i sont les récepteurs. S émet à 3 niveaux d'encodage. R_1 dispose d'une bande passante importante donc s'abonne à tous les flux. Contrairement à R_3 qui ne s'abonne qu'au flux principal (128 kb/s).
Le désavantage principal de l'approche cumulative est l'inégalité d'importance des flux. En effet, le récepteur sera moins tolérant aux pertes du flux principal sur lequel les autres flux sont basés. En contrepartie, sur un réseau à priorité, une priorité maximum sur les paquets de ce flux devrait donner de bon résultats.
Le protocole d'adaptation résultant est le plus simple possible :

- En cas de congestion (détection d'une perte) : le récepteur se désabonne au flux lui offrant la plus grande qualité.
- En sous charge (pas de pertes constatées) : le récepteur s'abonne à un nouveau flux pour avoir une meilleure qualité.

La figure 8 montre comment un récepteur passe d'un groupe à l'autre au cours du temps. Le désabonnement aux flux se fait un par un afin de dégrader la qualité le plus finement possible.
Un algorithme d'apprentissage vient compléter ce protocole. Si le récepteur remarque qu'à chaque fois qu'il s'abonne à un flux, une congestion est constatée tout de suite après, alors il ne demandera plus ce flux ni aucun autre apportant une qualité supérieure.
D'après les auteurs, le caractère hautement dynamique des abonnements et désabonnements des récepteurs peut nuire au passage à l'échelle. Afin de supprimer ce problème, un mécanisme d'apprentissage collectif est mis en place (comme celui dé-

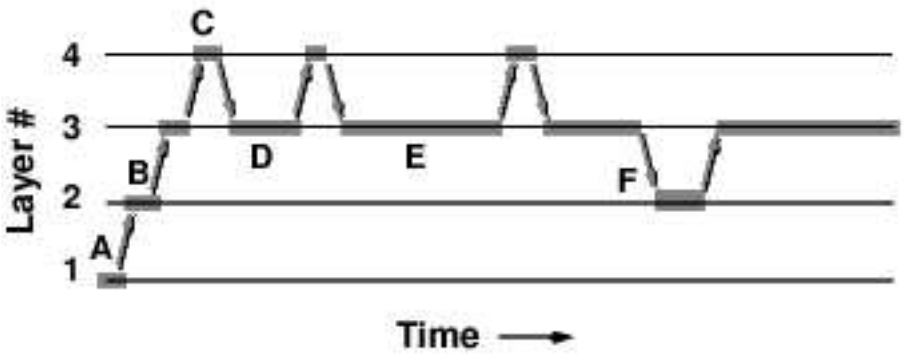

FIG. 8 – *Adaptation coté récepteur dans RLM*

crit plus tôt mais en collectivité).
Les performances de RLM dépendent directement des latences introduites par les abonnements et désabonnements des récepteurs, en effet le temps que le désabonnement se propage sur le réseau est ajouté au temps de congestion perçu par le récepteur. RLM tel qu'il est décrit dans [MJV96] ne fournit pas l'équité vis-à-vis des autres flux sur le réseau.

SOT, Self Organised Transcoding

S'inspirant de RLM, Kouvela, Hardman et Crowcroft [KHC98] présentent une architecture de contrôle de congestion multicast passant à l'échelle d'un réseau à récepteurs hétérogènes. La transmission des données se fait par émission simultanée (chaque flux est indépendant l'un de l'autre). La notion de groupe est introduite : plusieurs récepteurs proches physiquement et constatant des pertes vont former un groupe. Une fois ce groupe formé, si un membre n'a pas reçu un paquet, il pourra le demander à un autre membre du groupe. La notion de groupe permet aussi de placer des transcodeurs en amont du flux. Leur fonction est de convertir le flux entrant en un autre flux plus léger en bande passante. Cela permet d'alléger le réseau sur lequel sont situés les récepteurs. La figure suivante montre le fonctionnement des transcodeurs :

Les récepteurs (figure 9) percevant des pertes et proches sur l'arbre de multicast IP vont former un groupe (new group), le récepteur directement placé plus haut dans l'arbre que le lien surchargé (congested link) va servir de transcodeur. Il fournira au groupe un flux moins ¨ gourmand ¨ en bande passante. Cependant un représentant du groupe doit être élu afin de recevoir le flux d'origine afin de prévoir un éventuel abandon du transcodeur. Différents algorithmes sont présentés dans [KHC98] afin

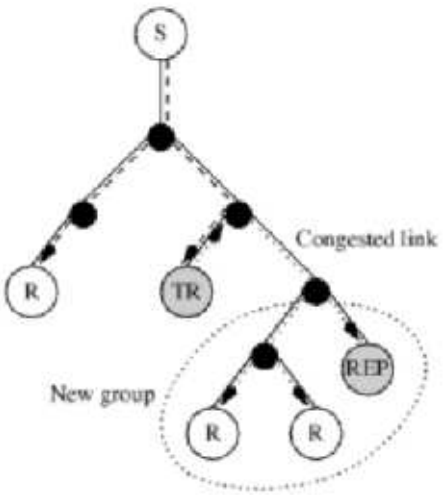

FIG. 9 – *Formation des groupes*

de gérer ces groupes. Le contrôle de congestion se fait par compte rendu du groupe au transcodeur qui détermine alors à quel débit il doit faire correspondre son flux en sortie.

Un contrôle de congestion *TCP-like*

Vicisano, Rizzo et Crowcroft dans [VRC98] considèrent que joindre et quitter le groupe multicast afin de recevoir les données au débit adapté à l'état du réseau est efficace seulement si les récepteurs agissent en coordination et si les algorithmes passent à l'échelle d'un grand nombre d'utilisateurs. Les auteurs mettent en place un mécanisme de coordination des décisions des récepteurs sans gestion de groupe explicite. En fait, chaque récepteur va prendre ses décisions de façon autonome, mais toutes les décisions des récepteurs seront synchronisées (En effet, le désabonnement à un flux n'a aucun effet tant que tous les récepteurs derrière le goulot d'étranglement ne se sont pas désabonnés). Lors de la détection d'une perte, le récepteur va calculer le débit auquel il doit adhérer grâce à un modèle de TCP (équation 3). Cela permet de rester équitable à TCP dans la prise de décision. Pour que ces décisions soient utiles le plus rapidement possible au niveau de l'arbre multicast, des points de synchronisation sont mis par l'émetteur dans le protocole. Un changement de niveau de souscription ne pourra se faire qu'après réception d'un de ces points de synchronisation.

MLDA, Multicast Loss-Delay based Adaptation

RLM et le travail de Vicisano, Rizzo et Crowcroft se basent sur une adaptation uniquement orienté récepteur, MLDA [SW00a] est un complément de ces techniques. Il combine RLM [KHC98] et LDA+ [SW00b], c'est donc un mécanisme d'adaptation orienté à la fois émetteur et récepteur. L'émetteur envoie le média à différents taux d'échantillonnage et les récepteurs envoient un compte rendu sur l'état de la réception du flux. Le récepteur détermine un temps T_{wait} en fonction du RTT calculé, ce temps est aussi petit que l'est le RTT. Dès que T_{wait} a expiré, le récepteur envoie son compte rendu à l'émetteur qui engagera l'adaptation de type LDA+ sur le flux concerné. Si un récepteur reçoit un compte rendu qui implique plus de restriction que lui en demanderait, alors il abandonne l'émission de son propre compte rendu, cela permet de ne pas surcharger l'émetteur de comptes rendus inutiles. Nous pouvons noter que le récepteur participe aux comptes rendu d'un flux s'il y est abonné et s'il ne remet pas en cause les bornes minimum et maximum du débit de ce flux (si c'est le cas, il s'abonnera au nouveau flux idéal pour lui). Pour mettre ce mécanisme en place, il est aussi nécessaire de synchroniser les comptes-rendus des récepteurs, la fin de la période d'observation du réseau est utilisé comme point de synchronisation explicite entre les récepteurs (la fin de la période d'observation est marquée par un envoi de compte-rendu). Les mesures sur l'implémentation de ce protocole faite par les auteurs montrent que MLDA est TCP équitable de façon plus fine que les mécanismes orientés uniquement récepteur, en effet, cette méthode permet de minimiser les variations de niveaux d'encodage entre les différents flux proposés par l'émetteur.

4.4 Conclusion sur les mécanismes multicast

Pour les connexions Multicast, la nécessité de disposer d'un réseau de type MBone est cruciale, si ce n'est pas le cas, nous pouvons envisager un mécanisme d'adaptation orienté émetteur tout en sachant qu'il ne sera pas possible de fournir un résultat équitable à chacun des récepteurs. Avec les mécanismes d'adaptation orientés récepteurs, il est possible de satisfaire un grand nombre de récepteurs hétérogènes (en bande passante) sans privilégier l'un ou l'autre. De plus, on peut espérer que les performances du système restent identiques quelque soit le nombre de récepteurs.

5 Forward Error Correction, compensation d'erreurs

Tandis que les techniques d'adaptation du débit d'émission des paquets visent à minimiser les pertes de paquets à venir, les pertes effectives restent un problème en matière de qualité sur les flux multimédia transportés. C'est pour cette raison que nous allons introduire dans cette section les mécanismes de type FORWARD ERROR CORRECTION (FEC) qui visent à faire de la redondance sur le flux afin de réparer les paquets perdus sans ou avec un minimum de retransmission de la part de l'émetteur. Une grande majorité des mécanismes exposés ici sont décrits dans [PHH98].Parmi les mécanismes de base, nous pouvons distinguer deux approches principales, la première est l'approche orientée émetteur tandis que la deuxième est orientée récepteur.

5.1 Réparation orientée émetteur

Nous pouvons distinguer encore deux types de réparations d'erreur ; les réparations avec retransmissions actives et les retransmissions utilisant un codage avec redondance rendant l'émetteur passif vis à vis de la réparation.

Parity FEC

Cette technique est indépendante du média transporté et tolère une perte tous les n paquets pour être efficace. Un paquet correcteur d'erreur est produit pour n paquets émis(figure 10). Ce paquet peut être construit en appliquant un ou exclusif sur chacun des bits relatifs des paquets émis.

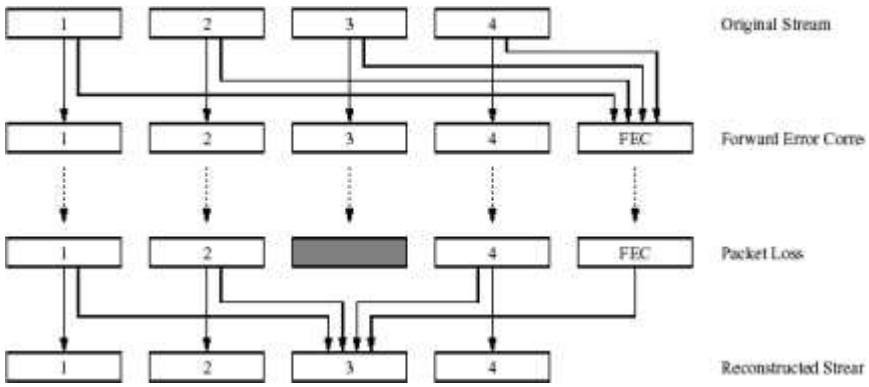

FIG. 10 – *Le FEC de parité*

On peut donc paramétrer le taux de perte toléré par cette méthode. De plus, le paquet correcteur peut être émis systématiquement ou à la demande selon le temps

de latence autorisé de bout en bout. Par contre, attendre le nième paquet afin de réparer une erreur augmente le temps de latence. Ce mécanisme fait l'objet d'un Payload RTP particulier [RS98].

FEC spécifique au média

L'idée ici est de rajouter directement une copie compressée du paquet dans le paquet suivant. De cette façon, une perte de paquet est compensée dès la réception du paquet suivant (figure 11). Cette méthode à été introduite par Hardman [HSHW95] et Bolot [BVG98].

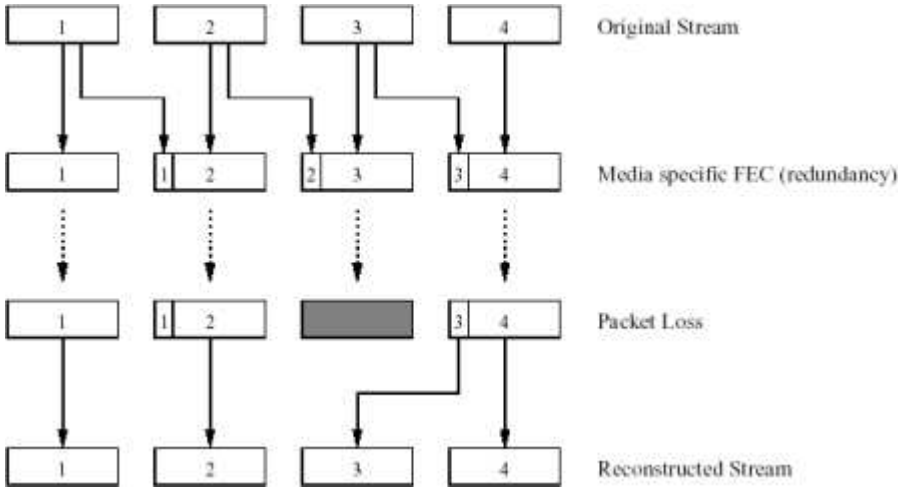

FIG. 11 – *FEC spécifique au média*

Cette méthode est efficace s'il n'y a que des périodes de perte courtes, si deux paquets l'un à la suite de l'autre sont perdus alors on ne pourra pas réparer le premier.

L'entrelacement

Si l'on fait l'hypothèse que la taille du grain du média est plus petite que la taille du paquet et que le délai de bout en bout peut être choisi grand, alors on peut utiliser l'entrelacement [RO70].

Sur n paquets, on répartit les paquets audio parmi ces paquets et l'on émet. Si un paquet est perdu alors au lieu d'avoir un trou de la taille d'un paquet, on a plusieurs trous plus petits qui passent plus facilement inaperçus.

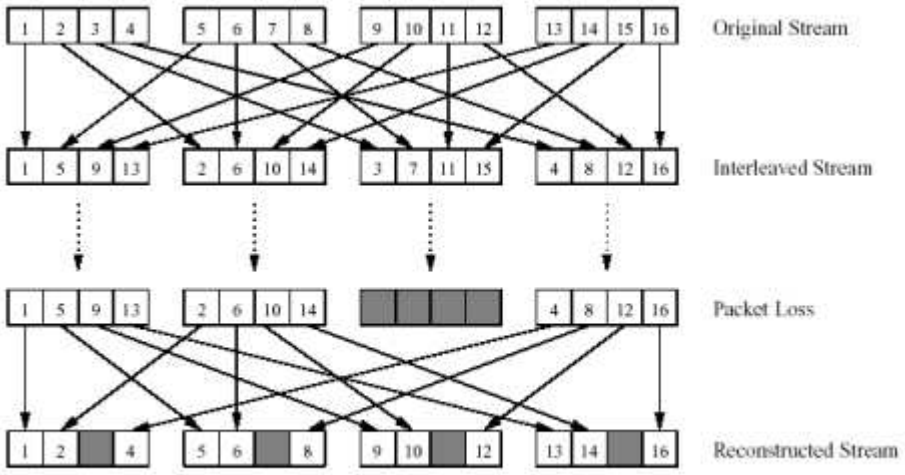

FIG. 12 – *FEC par entrelacement*

Retransmission et multicast

Floyd et *al.* [FJL+97] ont proposés un protocole de correction d'erreurs (Scalable Reliable Multicast protocol) basé sur la retransmission des paquets perdus en cas d'émission par multicast IP. Le principe est simple : si un membre perd un paquet alors il attend un temps déterminé par sa distance avec la source du paquet puis envoie un paquet de demande de retransmission sauf si entre temps un autre site à pris l'initiative de faire une demande de retransmission du même paquet. De cette façon, on minimise le nombre de demandes de retransmission et évite ainsi le produire un goulot d'étranglement au niveau de la source.

Cette technique étant orientée réémission, il faut pouvoir tolérer une latence forte de bout en bout. En effet, chaque site peut détecter des pertes de paquets différentes. Finalement chaque paquet ou la plupart devront être réémis. Ce phénomène est constaté par Xu dans [XMZY97], une implémentation d'un protocole SRM-like. D'autres techniques basées sur le même principe ont été formalisées comme [NBT98](retransmission d'un paquet FEC pour corriger plusieurs erreurs à la fois).

5.2 Dissimulation d'erreur

Les mécanismes présentés ici sont spécifiques au streaming audio. Ces techniques de dissimulation d'erreur sont orientées récepteur, d'après [PHH98], elles sont efficaces si les pertes plafonnent à 15% et si les paquets contiennent entre 4 et 40 ms de son. L'hypothèse forte considérée est qu'il y a une importante redondance sur de

très courts instants dans un flux audio, en particulier pour la voix.

Réparation par insertion

Il s'agit ici d'insérer un blanc, un bruit ou un paquet précédent afin de remplacer le paquet perdu. La qualité du son joué en est donc altérée. Dans le cas d'insertion de silence, la qualité est fortement dégradée [HSHW95]. L'insertion d'un bruit est bien plus efficace, elle fournit une meilleure qualité et rend le discours plus intelligible [ML50, War82]. [Sch98] propose ce mécanisme dans le cadre d'une RFC, il s'agit d'émettre un paquets *Confort Noise* afin de guider le mécanisme de réparation du récepteur à insérer un bruit particulier. La répétition de paquet est utilisée en GSM, il suffit de remplacer le paquet perdu par le paquet précédent. En cas de perte plus longue, on répète le même paquet en diminuant le volume (pendant environs 320 ms). Cette méthode est souvent utilisée pour la voix car elle fait un bon compromis entre simplicité d'implémentation et qualité restituée.

Réparation par interpolation ou régénération

Quelques travaux utilisent ce principe qui est fortement lié au signal audio même. [SSYG96] propose de prolonger les fréquences des paquets entourant une perte et d'en faire une moyenne afin de réparer l'erreur. Cette méthode demande de plus gros calculs mais fournit une meilleure qualité que les méthodes de réparation par insertion. Il existe d'autres mécanismes basés sur la comparaison de modèles et permettant de synthétiser le paquet perdu.

Les techniques présentées ci-dessus ne sont pas disjointes et peuvent être utilisées en conjonction. Selon le type d'utilisation, pour une application multimédia non-interactive (comme la radio), on va privilégier un temps de latence le plus fort possible et permettre une meilleure qualité, on peut alors imaginer d'utiliser l'entrelacement avec dissimulation d'erreur. Ce scénario n'est pas envisageable pour une application multimédia interactive où les temps de latence doivent généralement être plus faible. De toute façon, si l'on veut un délai acceptable, il faut impérativement accepter des pertes et donc nécessairement utiliser un mécanisme de dissimulation d'erreur.

5.3 Conclusion sur la compensation d'erreur

Incontestablement, les mécanismes de compensation d'erreurs sont un complément aux protocoles présentés au 4. Toutes les réparations présentées sont compatibles avec les mécanismes unicast et multicast. Cependant certains types de réparations donnent de meilleurs résultats sur la qualité. Si l'on veut transporter de la musique, la qualité est une contrainte forte, le Parity FEC, le FEC spécifique, l'entrelacement ou même la réparation par insertion de bruit risquent de laisser passer des

blancs dans le flux ou alors altérer la qualité du son. Une qualité optimale ne peut être obtenue qu'avec une réparation par interpolation ou régénération. En contrepartie, si l'on veut transporter de la voix dans le but de communiquer, la principale contrainte est de conserver le caractère intelligible du discours (on peut tolérer des pertes tant que le sens des phrases est compréhensible). Dans ce cas, nous pouvons privilégier le simplicité dans la réparation et donc utiliser une réparation par insertion de bruit par exemple. RAT (Robust Audio Tool) utilise la répétition de paquet, l'insertion de silence et l'insertion de bruit. L'avantage principal des réparations orientées récepteurs est la non-redondance d'informations dans les flux multimédia. Ces mécanismes se basent sur certains paradigmes de psycho-acoustique et donnent de très bon résultats.

6 Conclusion

Comme nous pouvons le constater, l'intégration d'applications multimédia interactives à large échelle est un véritable pari. Il est crucial pour ces applications de s'adapter de façon la plus fine possible à l'état du réseau mais aussi de cohabiter avec les autres flux présents (principalement des flux TCP). La contrainte temps réel se manifestant principalement sur les délais de bout en bout et la gigue, il est nécessaire à ces applications de disposer de protocoles fournissant des informations de synchronisation (estampillages), des informations sur les pertes, des informations sur le type des données transportées (afin de modifier le type d'encodage du flux de façon dynamique). Un tel protocole doit aussi tenir compte du grain du média pour éviter au maximum l'enchevêtrement des ses unités sur l'unité de transfert du protocole (il faut éviter par exemple qu'une même trame audio s'enchevêtre dans deux paquets consécutifs). RTP (Real-time Transport Protocol paragraphe 3.1) étant un standard issu de l'IETF, est le candidat le plus évident aujourd'hui pour transporter des flux multimédia en temps réel.

En contrepartie, le contrôle de congestion associé à RTP est laissé au bon vouloir de l'application. Pour ne pas mettre en cause la stabilité globale du réseau, le contrôle de congestion doit s'adapter correctement à l'état du réseau en minimisant le taux de perte. Il doit aussi coexister avec les autres flux sans ¨ manger¨ leur bande passante. Ce problème, comme nous l'avons vu dans le paragraphe 4 motive la recherche et n'est pas encore un problème totalement résolu. Bien que certains algorithmes comme LDA+ fournissent de très bon résultats, il reste nécessaire aux applications de spécifier les différents taux d'encodage disponibles. Surtout, il ne faut pas oublier que les performances des différents algorithmes d'adaptation sont directement liées à la granularité de l'adaptation, si l'application gère un grain trop gros, l'algorithme de contrôle de congestion risque de réagir de façon complètement différente. Dans le cas du multicast, le principal pari est de répondre à l'hétérogénéité des récepteurs. Se basant sur l'IP multicast, les mécanismes présentés au paragraphe 4.3.2 ne sont

pas directement applicables à l'échelle d'Internet (mais le sont sur un réseau local supportant évidement le Multicast IP).

Comme nous l'avons vu dans le paragraphe 5, l'addition de mécanismes de réparation d'erreur permet d'optimiser le transport en évitant les retransmissions de paquets perdus. Le choix du type de réparation est directement lié au type du média transporté et de sa fonction. Si la plate-forme d'accueil possède une puissance de calcul importante, une réparation par interpolation ou régénération donnera les meilleurs résultats.

L'objectif de toutes ces techniques reste la minimisation du taux de pertes perçu. Dans le cas du transport de la voix ou d'une image où les délais ne sont pas strictes (une conférence par exemple) il reste possible de jouer sur les temps de latence de bout en bout. Lorsque un paquet est arrivé chez le récepteur, il faut choisir le moment auquel il va être joué. Il se peut que l'on choisisse une latence trop faible. Dans ce cas, les paquets suivant risquent d'être détectés comme perdus alors qu'ils arrivent seulement un peu trop tard, une solution possible à ce problème est de retarder le moment où le paquet sera joué en retardant le plus possible des phrases entières (dans le cas de transport de la voix). [KK01] propose un algorithme calculant la variation du moment auquel les paquets doivent être joués. Ce type d'optimisation est directement liée à la sémantique même du flux (décaler un petit bout de phrase ne posera pas problème sur le résultat). Ce mécanisme n'est pas envisageable si l'on veut transporter du son avec une qualité optimale.

Références

[ACG+97] Amer, Conrad, Golden, Iren, and Caro. Partially-ordered, partially-reliable transport service for multimedia applications, 1997.

[ADLY95] Jong Suk Ahn, Peter B. Danzig, Zhen Liu, and Limin Yan. Evaluation of TCP vegas : Emulation and experiment. In *SIGCOMM*, pages 185–205, 1995.

[APS99] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control, 1999.

[BB00] D. Bansal and H. Balakrishnan. Tcp-friendly congestion control for real-time streaming applications, 2000.

[BDS96] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic qos control of multimedia applications based on rtp, 1996.

[Bol93] J. Bolot. End-to-end packet delay and loss behavior in the internet, 1993.

[BP95] Lawrence S. Brakmo and Larry L. Peterson. TCP vegas : End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8) :1465–1480, 1995.

[BVG96] Jean-Chrysostome Bolot and Andres Vega-Garcia. Control mechanisms for packet audio in the internet. In *INFOCOM (1)*, pages 232–239, 1996.

[BVG98] Jean-Crysostome Bolot and André Vega-Garcia. The case for FEC-based error control for packet audio in the Internet. *to appear in ACM Multimedia Systems*, 1998.

[CPW97] Shanwei Cen, Calton Pu, and Jonathan Walpole. Flow and congestion control for internet media streaming applications, 1997.

[CTCL95] Z. Chen, S. Tan, R. Campbell, and Y. Li. Real time video and audio in the World Wide Web. *Proc. Fourth International World Wide Web Conference*, 1995.

[DGS98] Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. Tcp dynamic acknowledgment delay : Theory and practice, 1998.

[DHLB98] Dane Dwyer, Sungwon Ha, Jia-Ru Li, and Vaduvur Bharghavan. An adaptive transport protocol for multimedia communication. In *International Conference on Multimedia Computing and Systems*, pages 23–32, 1998.

[DHT95] C. Diot, C. Huitema, and T. Turletti. Multimedia applications should be adaptive. *Proc. HPCS'95, Mystic (CN)*, 1995.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, Auguest 2000.

[FJL+97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6) :784–803, 1997.

[FSL97] W. Feng, U. Syyid, and J. Liu. Providing for an open, real-time corba, 1997.

[FWDC+00] Victor Fay-Wolfe, Lisa C. DiPippo, Gregory Cooper, Russell Johnston, Peter Kortmann, and Bhavani M. Thuraisingham. Real-time CORBA. *IEEE Transactions on Parallel and Distributed Systems*, 11(10) :1073–1089, 2000.

[Gia01] Jin Tang Giacomo. Rcs : A rate control scheme for real-time traffic in networks with high bandwidth-delay products and high bit error rates, 2001.

[HBG00] Urs Hengartner, Jurg Bolliger, and Thomas Gross. TCP vegas revisited. In *INFOCOM (3)*, pages 1546–1555, 2000.

[HSHW95] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. *Proceedings of INET, Oahu, Hawaii*, 1995.

[Jac88] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.

[JM92] JACOBSON and McCanne. The lbl audio tool vat, 1992.

[KHC98] I. KOUVELAS, V. HARDMAN, and J. CROWCROFT. Network adaptive continuous-media applications through self organised transcoding, 1998.

[KK01] Aman Kansal and Abhay Karandikar. Adaptive delay adjustment for low jitter audio over internet. In *IEEE Globecom*, 2001.

[MF97] Jamshid Mahdavi and Sally Floyd. Tcp-friendly unicast rate-based flow control, 1997.

[MJV96] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, volume 26,4, pages 117–130, New York, 26–30 1996. ACM Press.

[ML50] G. Miller and J. Licklider. The intelligibility of interrupted speech, 1950.

[Muk00] Biswaroop Mukherjee. Time-lined tcp : a transport protocol for delivery of streaming media over the internet, 2000.

[NBT98] Jörg Nonnenmacher, Ernst W. Biersack, and Don Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4) :349–361, 1998.

[PFOC00] Mike Piecuch, Ken French, George Oprica, and Mark Claypool. A selective retransmission protocol for multimedia on the internet. In *SPIE International Symposium on Multimedia Systems and Applications*, 2000.

[PHH98] C. Perkins, O. Hodson, and V. Hardman. A survey of packet-loss recovery techniques for streaming audio, 1998.

[PKTK98] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. *UMass-CMPSCI Technical Report TR 98-04*, 1998.

[RDSD99] L. Rojas-Cardenas, L. Dairaine, P. Senac, and M. Diaz. An adaptive transport service for multimedia stream. *IEEE Transactions on Multimedia*, 1999.

[RHE99] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP : An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *INFOCOM (3)*, pages 1337–1345, 1999.

[RLS99] P. Reichl, S. Leinen, and B. Stiller. A practical review of pricing and cost recovery for internet services, 1999.

[RO70] J. Ramsey and R. Optimum. Ieee transactions on information theory, 1970.

[RS98] J. Rosenberg and H. Schulzrinne. An RTP payload format for generic forward error correction. *Internet-Draft draft-ietf-avt-fec-03.txt (work in progress)*, 1998.

[SCFJ98] Schulzrinne, Casner, Frederick, and Jacobson. RTP : A transport protocol for real-time applications. *Internet-Draft ietf-avt-rtp-new-01.txt (work inprogress)*, 1998.

[Sch98] Schulzrinne. RTP profile for audio and video conferences with minimal control. *Internet-Draft ietf-avt-profile-new-03.txt (work in progress)*, 1998.

[SK97] Floyd S. and Fall K. Router mechanisms to support end-to-end congestion control, 1997.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp, 1998.

[SSYG96] H. Sanneck, A. Stenger, K. Younes, and B. Girod. A new technique for audio packet loss concealment, 1996.

[Ste95] R. Stevens. *TCP Illustrated*, volume 1. Addison Wesley, 1995.

[SW00a] D. Sisalem and A. Wolisz. MLDA : A TCP-friendly congestion control framework for heterogeneous multicast environments, 2000.

[SW00b] Dorgham Sisalem and Adam Wolisz. LDA+ : A TCP-friendly adaptation scheme for multimedia communication. In *IEEE International Conference on Multimedia and Expo (III)*, pages 1619–1622, 2000.

[SW01] Dorgham Sisalem and Adam Wolisz. Constrained TCP-friendly congestion control for multimedia communication. In *QofIS*, pages 17–31, 2001.

[TZ99] Wai Tan and Avideh Zakhor. Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol. *IEEE Transactions on Multimedia*, 1(2) :172–186, 1999.

[VRC98] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-like congestion control for layered multicast data transfer. In *INFOCOM (3)*, pages 996–1003, 1998.

[War82] R. Warren. Pergamon press inc, 1982.

[XMZY97] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications, 1997.

[ZDE93] Lixia Zhang, Stephen Deering, and Deborah Estrin. RSVP : A new resource ReSerVation protocol. *IEEE network*, 7(5) :8–?, September 1993.

[ZT01] C. Zhang and V. Tsaoussidis. Tcp real : Improving real-time capabilities of tcp over heterogeneous networks, 2001.