

Rapport de stage
Métaphore de l'orchestre virtuel
Étude des contraintes systèmes et réseaux, puis
prototypage.

Bouillot Nicolas

18 novembre 2002

Table des matières

1	Présentation du stage	5
1.1	Présentation des équipes	5
1.2	jMax, un environnement visuel de programmation audio	6
1.3	Le projet et le stage	7
2	Études préliminaires des protocoles de communications pour l'audio	9
2.1	Contexte	9
2.1.1	TCP pour le multimédia?	11
2.1.2	L'impasse?	13
2.2	Transport de média et standards	14
2.2.1	RTP et RTCP	15
2.3	Approche Rate Control et TCP friendly, le contrôle du débit	17
2.3.1	Mécanismes unicast	18
2.3.2	Conclusion sur les mécanismes unicast	21
2.3.3	Les mécanismes Multicast	21
2.3.4	Conclusion sur les mécanismes multicast	23
2.4	Forward Error Correction, compensation d'erreurs	24
2.4.1	Réparation orientée émetteur	24
2.4.2	Dissimulation d'erreur	26
2.4.3	Conclusion sur la compensation d'erreur	27
2.5	Conclusion de l'étude	27
3	Mise en œuvre des interactions pour l'orchestre	29
3.1	Intégration à jMax	29
3.1.1	Le périphérique RTPaudioport, le transport du son	29
3.1.2	Le son transporté	31
3.2	Les objets RTPin et RTPout, faire coopérer les musiciens	31
3.2.1	Synchronisation des flux des différents musiciens entre eux	31
3.2.2	Résolution du problème de dérive des horloges d'échantillonnages entre les cartes sons des musiciens	32
3.2.3	Les objets RTPin et RTPout	34
3.2.4	Architecture des objets RTPin et RTPout	35
3.3	Problèmes rencontrés	36
3.3.1	Appréhension d'un nouveau domaine, l'audionumérique	36
3.3.2	L'installation et prise en main	36
3.3.3	Utilisation de la librairie RTP dans jMax	37
3.3.4	Mesures	37

4	Travaux futurs	40
4.1	Interaction entre les musiciens	40
4.2	Le mixage et le publique	41
5	Conclusion	42

Table des figures

1	<i>Exemple de patch jMax</i>	7
2	<i>Le concert réparti</i>	8
3	<i>Le débit utile</i>	10
4	<i>L'augmentation linéaire de la fenêtre de contrôle de congestion après le Slow Start</i>	11
5	<i>Diminution exponentielle du seuil</i>	12
6	<i>Message intervenant dans le calcul du RTT</i>	16
7	<i>Architecture de RLM</i>	23
8	<i>Le FEC de parité</i>	25
9	<i>FEC spécifique au média</i>	25
10	<i>exemple de patch jMax utilisant RTPaudioport</i>	30
11	<i>Exemple d'utilisation des objets RTPin et RTPout</i>	34
12	<i>Architecture des objet RTPin et RTPout</i>	35
13	<i>Mesure de la taille du tampon audio</i>	38
14	<i>Mesure du délai aller-retour</i>	47
15	<i>Exemple d'interaction sans délai sur son propre instrument</i>	48

1 Présentation du stage

Ce stage s'effectue dans le cadre d'un projet commun entre l'IRCAM ¹ et le CEDRIC ² sur le thème des services systèmes pour les applications multimédia réparties.

1.1 Présentation des équipes

L'IRCAM mène des recherches fondamentales sur les apports de l'informatique, de la physique et de l'acoustique à la problématique musicale. L'équipe Acoustique des salles étudie l'effet de la propagation du son dans les lieux d'écoute, une part importante de son activité est consacrée à la simulation acoustique, intégrant à la fois les paramètres de localisation des sources, d'effet de salle et l'adaptation à divers dispositifs de restitution à travers des procédures de codage adaptées. L'équipe Systèmes temps-réel met au point des dispositifs informatiques destinés au traitement en temps-réel des informations musicales et sonores (entre autre jMax voir paragraphe 1.2), ces systèmes sont utilisés par les compositeurs pour concevoir des œuvres associant de manière interactive des parties instrumentales et des parties électroniques calculées par ordinateur.

jMax est donc un environnement de programmation sonore. Les objets disponibles sont multiples. Il est possible de lire un fichier audio, d'utiliser un micro comme entrée, ajouter des effets sur un flux audio, produire des notes par synthèse, détecter la hauteur des notes... La variété des objets disponibles dans jMax rendent celui-ci utilisable comme outil d'expérimentation sonore (utilisation bas niveau sur le son lui-même) mais aussi comme outil de plus haut niveau (mixeur, séquenceur ou même comme instrument).

Le CEDRIC mène des travaux dans l'ensemble des domaines de l'informatique avec un axe principal orienté vers la conception, la modélisation et la validation de systèmes. Le CEDRIC, de par son appartenance au CNAM, joue un rôle privilégié en matière de transfert de technologie de la recherche vers l'industrie. Le CEDRIC entretient des rapports avec les principaux acteurs industriels et publics des Nouvelles Technologies de l'information et de la Communication (NTIC). Il participe aux actions de recherche des réseaux français et européens.

Les travaux en cour de l'équipe portent de façon générale sur les interfaces multimodales pour le travail coopératif. Certains de ses travaux sont d'un ordre théorique (spécification des interfaces, sécurité des systèmes multimédia), d'autres expérimentales (Interface symbolique 3D, spatialisation du son, Architectures pour le multimédia) pour aboutir à des mises en œuvre dans diverses applications (bibliothèque

¹contacts : Francois Déchelle, Francois.Dechelle@ircam.fr et Olivier Warusfeld warusfel@ircam.fr

²contact : Eric Gressier-Soudan, gressier@cnam.fr

numérique, Supervision, CAO à distance, EAD). L'année 2000 a vu l'équipe s'étoffer de plusieurs membres apportant des compétences dans les domaines des architectures et du traitement du son.

1.2 jMax, un environnement visuel de programmation audio

De même que MAX (son prédécesseur), le langage jMax est souvent présenté comme un langage de programmation visuel : l'interface, permet à l'utilisateur de construire une application sous forme de *patch* (morceau de programme jMax pouvant être réutilisé dans un autre programme jMax) par connexion de modules. Chaque module se présente sous la forme d'une *boîte* ayant des entrées sorties reliées à d'autre module (voir figure 1). Les connections entre les modules représentent des communications de paramètres de contrôle ou de signaux, avec une sémantique d'envoi de messages. Un module peut être une unité de traitement (opérateur arithmétique simple ou transformation complexe du signal) ; il peut aussi représenter des données (ex : une ligne de retard), ou enfin être un module système d'entrée-sortie audio ou MIDI³. Les modules peuvent être des primitives, appelées *objets*. Ces modules peuvent être eux-mêmes des patches, donnant ainsi aux patches une structure hiérarchique. Enfin, certains modules contrôleurs ont un comportement graphique interactif et permettent l'envoi de messages depuis l'interface graphique (par exemple l'activation d'un interrupteur).

Notons qu'il existe deux grandes familles d'objets : ceux traitant des paramètres de contrôle et ceux traitant les signaux. Usuellement, le nom des objets traitant des signaux se terminent par le caractère tilda.

La figure 1 nous montre l'exemple d'un patch jMax produisant une suite de notes aléatoires à un tempo aléatoire et variable. La ligne sous le menu nous permet de choisir le type d'objet que l'on veut insérer dans le patch. Par exemple un potentiomètre (qui sert ici à régler le volume) ou encore un objet comme la sortie audio (en bas de la figure, noté dac). L'interrupteur du haut active l'objet *metro* (un métronome) qui donne l'impulsion à l'objet *drunk*. Cet objet génère un nombre aléatoire qui sera choisit comme tempo de la mélodie. Il en est de même pour générer des entiers (deuxième interrupteur) qui seront convertis en notes grâce aux objets *mtof* et *sig*.

Pour plus d'informations sur jMax : voir [DBC⁺98a, DBC⁺98b, DBC⁺99, Déc00]

³Musical Instrument Digital Interface

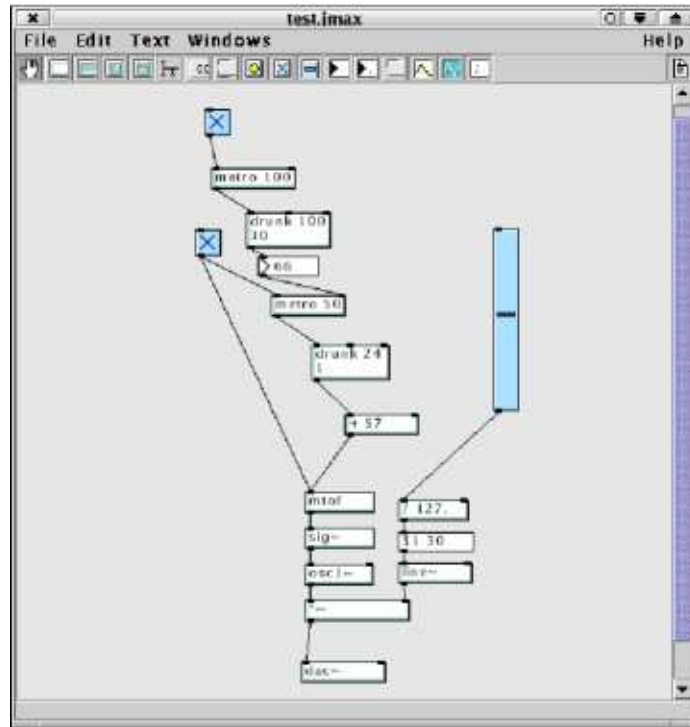


FIG. 1 – Exemple de patch jMax

1.3 Le projet et le stage

La conjonction des travaux des équipes de l'IRCAM et du CNAM-CEDRIC doit permettre la mise en place d'un orchestre virtuellement réparti avec des musiciens physiquement éloignés. Le Cnam-Cedric est chargé de la conception d'une architecture système distribué pour supporter cette métaphore de l'orchestre virtuel.

La figure 2 nous présente l'architecture générale du projet. Les musiciens sont physiquement répartis mais doivent jouer ensemble en temps réel. Afin de permettre à un publique d'assister au concert virtuel, un ingénieur du son collectera les flux audio des musiciens pour faire le mixage sonore et spatial de la scène. Le mixage spatial consiste à positionner les sources sonores dans un espace à trois dimensions. Afin de permettre ce type de mixage, le son transporté sera du son quatre canaux⁴.

Le transport de flux produit par des musiciens existe déjà, il a été mis en place entre Montréal et Los Angeles par Jeremy Cooperstock et Stephen Spackman de l'université de Montréal. Le principe est de faire jouer des musiciens ensemble dans la même pièce, de transporter chacun des flux (non compressé) et de les mixer à

⁴permettant la reproduction d'un un effet tridimensionnel (positionnement dans l'espace du son)

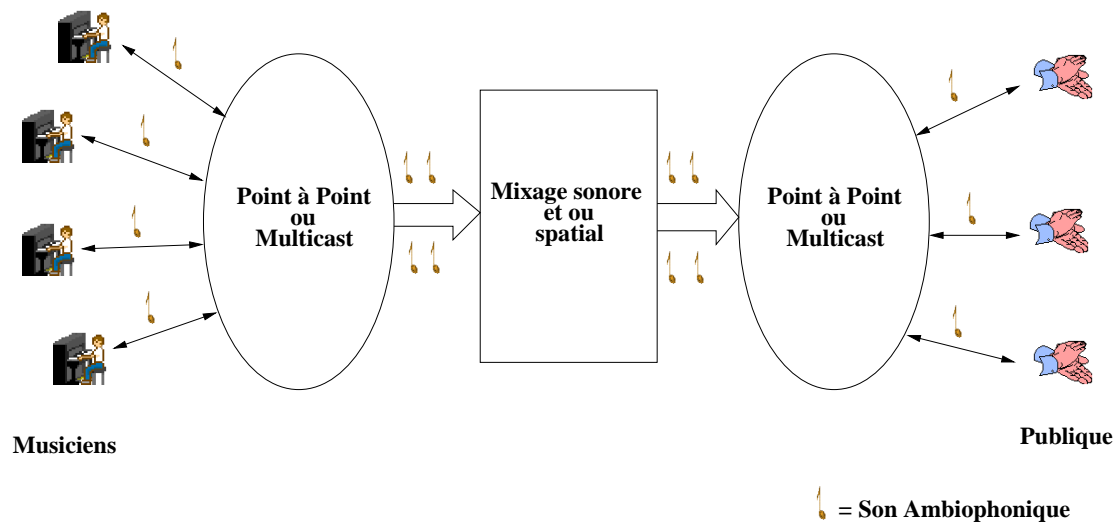


FIG. 2 – *Le concert réparti*

distance. Une description plus détaillée peut-être trouvée dans [CS01]. Cependant, le résultat obtenu n'est pas destiné à être écouté en direct et les musiciens ne vont pas interagir entre eux au travers d'un réseau.

L'objectif du stage de DEA est d'étudier et de prototyper les bases d'une architecture système et réseau susceptible de permettre à un orchestre dont les membres sont répartis géographiquement d'offrir un concert à un public éventuellement réparti lui aussi. Pour cela, le prototype issu du stage doit permettre aux musiciens de jouer de la musique ensemble de façon synchronisée et interactive au travers d'instances du logiciel jMax.

Le prototype devra transférer les échantillons de N canaux simultanés ($N = 4$ ou $N = 8$) codés chacun sur 24 bits de façon synchrone. La restitution des sons nécessite une latence constante. Pour les besoins du prototypage dans le cadre du stage de DEA, un réseau local Ethernet 100Mb/s sera utilisé. Dans un second temps, un réseau MAN haut débit auquel l'IRCAM devrait être raccordé pourrait servir de base aux expérimentations ultérieures.

Le stage cible plus spécifiquement la partie communication entre les musiciens. Le paragraphe 2 présente une étude des protocoles de communication pour l'audio en temps réel. Le paragraphe 3 décrit le prototype, le déroulement de son développement et les difficultés rencontrées. Le paragraphe 4 expose les travaux futurs et les possibles évolutions du prototype et enfin le paragraphe 5 conclut le rapport.

2 Études préliminaires des protocoles de communications pour l'audio

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 ce paragraphe. 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 volée (Un orchestre virtuel 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 perte pour chacun des acteurs tout en considérant leur hétérogénéité. Le son étant envoyé sur le réseau dès qu'il est produit, nous ne considérerons que les méthodes de compression isochrone. Nous préciserons aussi en quoi les mécanismes étudiés peuvent contribuer au projet du concert virtuel.

Nous pourrions trouver plus de détails sur les sujets abordés dans ma synthèse bibliographique de DEA [Bou02].

Cette synthèse est organisée comme suit. Le paragraphe 2.1 présentera le contexte réseau et transport dans lequel nous situons notre étude. Le paragraphe 2.2 exposera certains protocoles standards mis en place afin d'aider le développement des applications multimédia nous intéressant. Le paragraphe 2.3 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 2.4 proposera des mécanismes de réparation d'erreurs sur le flux multimédia, et cela directement au niveau application. Enfin nous concluons l'étude dans le paragraphe 2.5.

2.1 Contexte

L'implantation d'applications multimédia distribuées ne se fait pas sans considération de leur environnement. En effet nous pouvons rapidement nous voir confrontés à différents problèmes intrinsèques aux architectures des réseaux et des systèmes d'exploitation, 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.

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

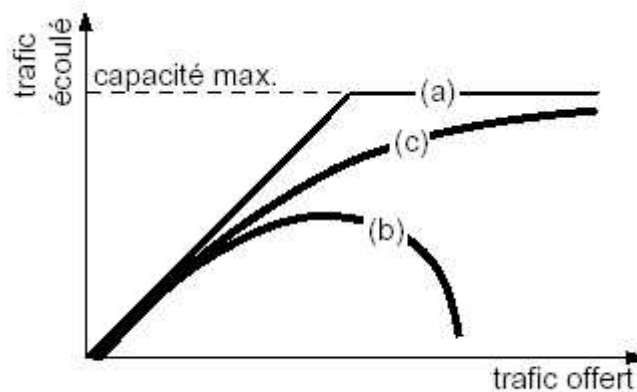


FIG. 3 – *Le débit utile*

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.

2.1.1 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 ⁵ :

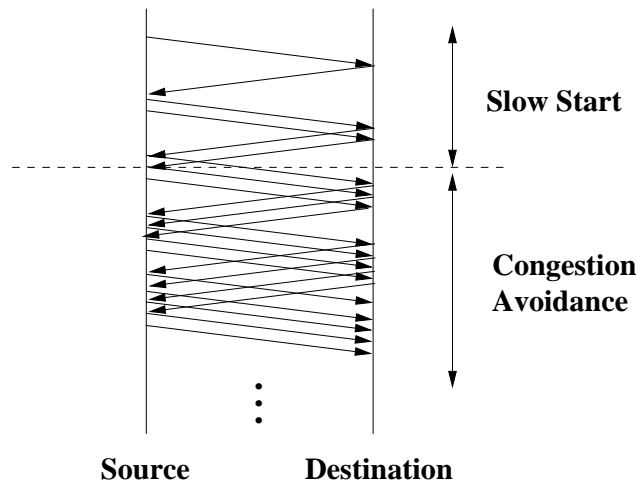


FIG. 4 – L'augmentation linéaire de la fenêtre de contrôle de congestion après le *Slow Start*

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 où TCP repasse au *congestion Avoidance*.

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.

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 tem-

⁵[Ste95] : une description détaillée du fonctionnement de TCP.

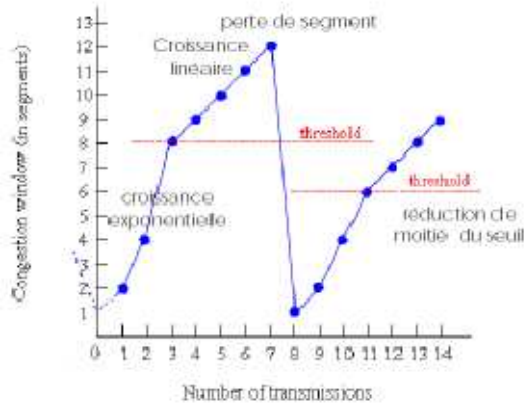


FIG. 5 – *Diminution exponentielle du seuil*

poelles fortes, les applications multimédia doivent activement participer au contrôle de congestion (être adaptatives) 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 bons 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.

2.1.2 L'impasse ?

TCP 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 se 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 2.4). 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 le délai 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.

Une 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 travail.

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 ⁶ (paragraphe 2.3).

2.2 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 séquençement 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 transportées 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 données multimédia.

⁶Nous 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)

2.2.1 RTP et RTCP

RTP [SCFJ98] est un protocole de transport de données temps réel qui vise 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 (charge utile) 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 ou des récepteurs vers la source. Pour cela, RTCP transmet cinq types de messages que nous présentons ici afin de connaître exactement la sémantique disponible au niveau application (qui ainsi peut implémenter 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.

La figure 6 montre comment est calculé le RTT (Round Trip Time, le temps aller retour). Au temps T_1 l'émetteur envoie un paquet RTCP. Dès sa réception, le

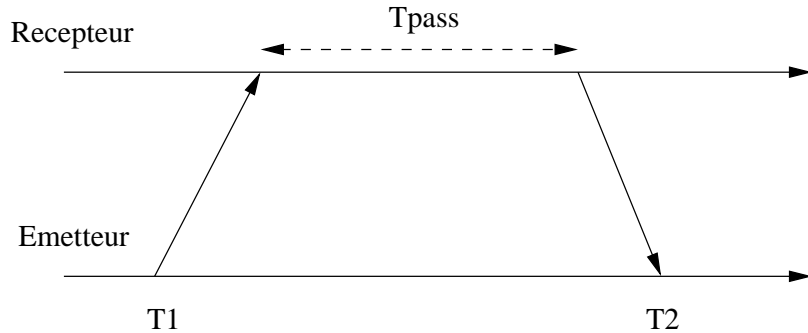


FIG. 6 – Message intervenant dans le calcul du RTT

récepteur compte le temps T_{pass} , le temps qui s'écoule entre la réception du rapport de l'émetteur et le moment où il va lui-même renvoyer un rapport dans lequel il va communiquer T_{pass} . Lorsque que l'émetteur recevra ce rapport au temps T_2 , il pourra estimer le RTT comme suit :

$$RTT = (T_1 - T_2) - T_{pass} \quad (1)$$

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 se 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 2.3 les algorithmes exploitant les rapports RTCP du ou des récepteurs afin de gérer le débit de la source.

Dans le cadre de l'orchestre virtuel réparti, nous allons travailler sur un réseau local avec du son de haute qualité. Il ne sera pas possible de faire varier la charge du flux sur le réseau en jouant sur les débits. Toutefois, les atouts de RTP restent très intéressants pour notre utilisation. En effet, l'utilisation de ce protocole nous fournira les différents types de synchronisation non disponible avec TCP ou UDP. Notons qu'un précédent prototype existe pour transporter du son entre deux ins-

tances de jMax (voir le rapport de stage de F. Dufourd [Duf00]). Le protocole repose sur UDP, la synchronisation a dû être programmée dans le protocole lui-même de façon monolithique. Cela contrairement aux applications fondées sur RTP qui grâce à l'intermédiaire des payloads peuvent être compatible entre elles (Il faut juste que les deux applications soient capables de comprendre le même payload).

2.3 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 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.

Le son transporté pour l'orchestre virtuel étant de haute qualité, il ne sera pas possible d'utiliser ce genre de mécanisme directement dans le prototype du stage. En effet, il ne sera pas possible de jouer sur le débit en fonction de l'état du réseau, le son devant rester constamment à la même qualité. Toutefois, dans le cadre du concert réparti avec diffusion à grande échelle (sur Internet), l'hétérogénéité des récepteurs en terme de latence et bande passante rendra ce type de mécanisme essentiel.

2.3.1 Mécanismes unicast

Les TCP-like, approche réactive

Nous allons présenter dans ce paragraphe un protocole et particulièrement son algorithme de contrôle de congestion. Le principe ici est d'imiter le comportement de TCP afin d'être équitable tout en le modifiant pour que le protocole construit convienne au transport de données multimédia. Nous pouvons citer parmi les protocoles de ce type [PKTK98, CPW97, BDS96].

Algorithmes binomiaux Bansal et Balakrishnan [BB00] ont généralisé le fonctionnement du contrôle de congestion de TCP à travers les algorithmes de contrôle de congestion appelés *algorithmes 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).

$$I : w_{t+RTT} \leftarrow w_t + \alpha/w_t^k; \alpha > 0 \quad (2)$$

$$D : w_{t+\delta t} \leftarrow w_t - \beta w_t^l; 0 < \beta < 1 \quad (3)$$

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 $l = 1$ modélisent 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 a été implémenté dans le logiciel d'audio-conférence Vat [JM92].

L'approche par équation, une approche prédictive

Comme dans le paragraphe précédent, nous allons présenter des algorithmes de contrôle de congestion. Ici le calcul 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 d'un goulot d'étranglement). Nous pouvons citer parmi les protocoles de ce type [TZ99, RHE99].

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 la bande passante d'une connexion TCP. L'effet des pertes de paquets sur une connexion TCP est donné par la formule suivante :

$$\text{Bandepassante} = C \frac{MTU}{RTT \sqrt{Loss}} \quad (4)$$

où MTU est la taille des paquets envoyés, RTT le temps d'un aller-retour, $Loss$ les pertes constatées par la connexion (mise à 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.

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 2.2.1. 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, ils 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 :

$$R = \frac{\text{TaillePaquetInvestigation}}{\text{TempsEntreReceptionDeDeuxPaquets}} \quad (5)$$

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

Ces informations 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] (calcul de la bande passante équitable avec TCP) :

$$r_{TCP} = \frac{M}{t_{RTT}\sqrt{\frac{2Dl}{3}} + t_{out}\min(1, 3\sqrt{\frac{3Dl}{8}})l(1 + 32l^2)} \quad (6)$$

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 considérés comme le goulot d'étranglement. L'équité entre un nombre égal de flux FTP et LDA+ (jusqu'à 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 fines, 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 CTFAP 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 à tolérer 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 sont un compromis entre les valeurs obtenues 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+.

2.3.2 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.

2.3.3 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 [BVG96]. 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).

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'ait 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.

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 :

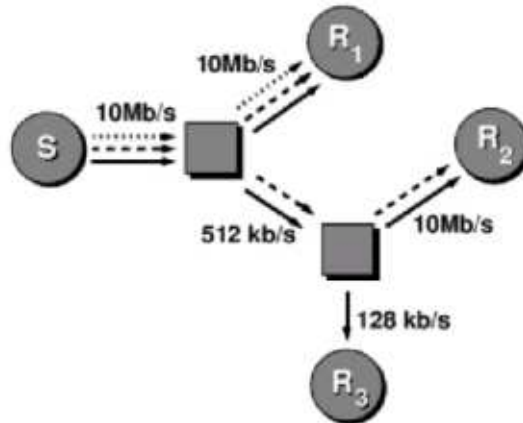


FIG. 7 – Architecture de RLM

- 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.

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 décrit 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.

2.3.4 Conclusion sur les mécanismes multicast

Pour les connexions Multicast, la nécessité de disposer d'un réseau supportant le multicast 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. Nous pouvons citer parmi les protocoles de ce type [KHC98, VRC98, SW00a].

2.4 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.

L'intérêt de ce type de mécanisme est évident pour chacun des flux du concert réparti (pas de réémission et amélioration de la qualité du son sans réémission de paquets). Dans le prototype issu du stage, aucun mécanisme de réparation n'a été introduit. En effet, comme nous allons le voir, le mécanisme à choisir dépend fortement du type du flux à protéger (latence, unicast, multicast...). La réparation d'erreur fut mise de côté dans un premier temps pour privilégier l'architecture du prototype. Toutefois nous ajouterons très certainement la compensation d'erreur au concert virtuel.

2.4.1 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 8). Ce paquet peut être construit en appliquant un ou exclusif sur chacun des bits relatifs des paquets émis.

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].

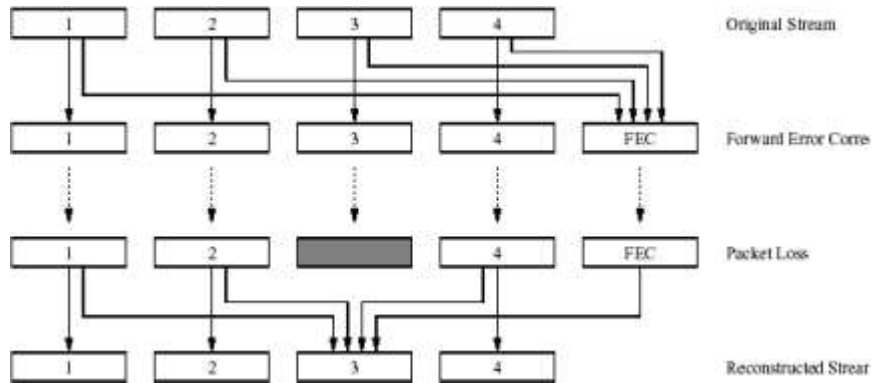


FIG. 8 – Le FEC de parité

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 9). Cette méthode à été introduite par Hardman [HSHW95] et Bolot [BVG98].

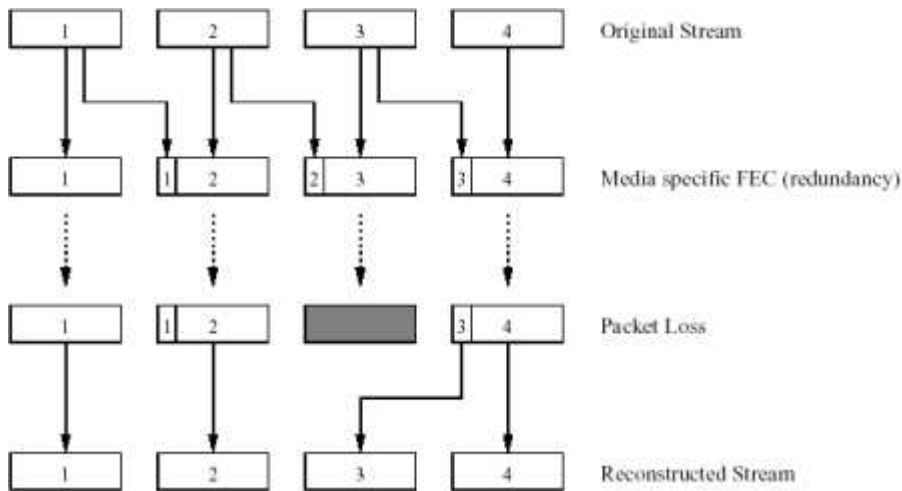


FIG. 9 – FEC spécifique au média

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.

Retransmission et multicast Floyd et al. [FJL⁺97] ont proposés un protocole de correction d'erreurs (Scalable Reliable Multicast protocol) basé sur la retransmis-

sion 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 a 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 de 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).

2.4.2 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 paquet *Comfort 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 environ 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 faibles. 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.

2.4.3 Conclusion sur la compensation d'erreur

Incontestablement, les mécanismes de compensation d'erreurs sont un complément aux protocoles présentés au 2.3. 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, 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 la 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 bons résultats. Pour la communication entre les différents musiciens, le son doit rester de bonne qualité, nous pouvons exclure directement les mécanismes de type insertion (de bruit ou de silence). Le mécanisme à choisir doit faire l'objet de plus amples réflexions car le choix à faire mêle l'audio-numérique de haute qualité, les contraintes réseau et l'architecture de l'orchestre et du concert (les auditeurs du concert n'ayant pas les mêmes contraintes que les musiciens, un mécanisme différent pourra être appliqué).

2.5 Conclusion de l'étude

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 de 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 2.2.1) 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 2.3 motive la recherche et n'est pas encore un problème totalement résolu. Bien que certains algorithmes comme LDA+ fournissent de très bons 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 2.3.3 ne sont pas directement applicables à l'échelle d'Internet (mais le sont sur un réseau local supportant évidemment le Multicast IP).

Comme nous l'avons vu dans le paragraphe 2.4, 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. 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.

3 Mise en œuvre des interactions pour l'orchestre

Contrairement à une application d'audio-conférence sur Internet, l'orchestre réparti doit fonctionner avec un signal de très bonne qualité afin d'éviter toute surcharge de calcul et de latence engendrées par les algorithmes de compression et de décompression. Ce choix rend l'orchestre virtuel possible sur réseau local ou métropolitain mais non applicable à l'échelle d'Internet. En effet, les flux transportés sont non modifiables, il est donc impossible d'adapter la qualité du son en fonction du réseau (en dehors de la variation de la fréquence d'échantillonnage).

Toutefois, la présentation de l'étude préliminaire à l'équipe système temps réel de l'IRCAM a confirmé que l'utilisation du protocole RTP est une solution adéquate au transport du son produit par chaque musicien. Bien que l'adaptation du débit de flux ne soit pas possible entre les musiciens, RTP permet entre autre l'estampillage à l'échantillon (l'estampille est incrémenté de un pour un échantillon), la possibilité d'estimation du RTT⁷. De plus, l'objectif à long terme est le concert réparti, une stratégie d'adaptation des flux en fonction du réseau sera nécessaire pour les récepteurs du concert. Utiliser RTP pour l'orchestre maintient une cohérence entre les protocoles de bout en bout du concert.

3.1 Intégration à jMax

Afin de pouvoir faire jouer des musiciens entre eux, il fut choisit de créer deux objets jMax, un modélisant l'envoi du flux et l'autre modélisant sa réception. De cette façon, les musiciens peuvent communiquer directement grâce à jMax mais aussi traiter le flux reçu d'un partenaire (mettre un effet, un retard...).

3.1.1 Le périphérique RTPaudioport, le transport du son

La fonctionnalité de flux audio temps réel (dans un premier temps) est intégrée à jMax sous forme d'un nouveau périphérique *FTS*⁸. Ce périphérique est programmée en langage C sur le modèle des périphériques audio de Linux (OSS et ALSA), interface entre *FTS* et le pilote audio. Configurées en entrée sortie au démarrage de jMax, l'utilisation de ce périphérique (RTPaudioport) dans jMax se fait de la même façon que lorsque l'on manipule un microphone ou des haut-parleurs (il faut toutefois passer en paramètre les informations nécessaires à la connection RTP). La figure 10 montre un exemple de patch utilisant le périphérique RTPaudioport.

⁷Round Trip Time : le temps aller-retour d'un paquet sur le réseau

⁸Faster Than Sound : le noyau calculateur de jMax

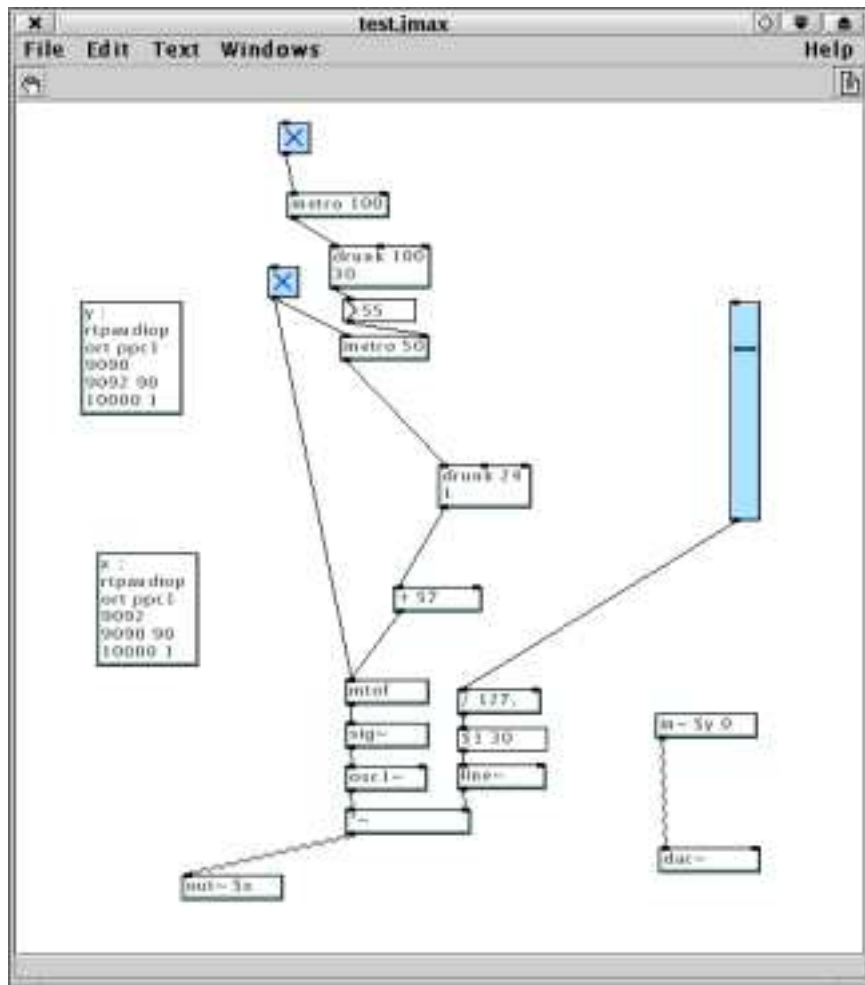


FIG. 10 – exemple de patch jMax utilisant RTPaudioport

La mise en œuvre du module RTPaudioport ne sera pas détaillée ici. En effet, le module RTPaudioport transporte un flux entre deux instances du logiciel jMax. Cela n'est pas encore suffisant pour que deux musiciens puissent jouer de la musique ensemble en temps réel. Pour que cela soit possible, il faut que le musicien puisse avoir une estimation du délai que mettent les paquets pour arriver chez le récepteur. Cette information lui permettrait d'ajouter un retard sur le son produit par son instrument (afin de s'entendre de façon synchronisée avec le son reçu). De par sa nature, un port audio (OSS, ALSA ou RTPaudioport) est déclaré dans une variable. L'obtention des flux en entrée et en sortie se fait par les objets `in~` et `out~` (voir figure 10). Il n'est donc pas possible d'obtenir d'autre information (comme la latence estimée) dans une outlet (sortie d'un objet jMax).

3.1.2 Le son transporté

Dans un premier temps, le matériel de prise et de restitution de son ambiophonique n'étant pas disponible, le prototype a été testé avec du son à deux canaux (stéréo). En fait, l'utilisateur choisit le nombre de canaux qu'il veut utiliser, il lui est donc possible de transférer du son monophonique, stéréophonique ou multi-canaux.

jMax fonctionne selon le principe de boucles DSP⁹. Le programmeur d'objet jMax doit fournir une fonction traitant les n échantillons de la boucle (généralement $n = 64$). L'intégration des données dans un paquet RTP se fait comme suit : les n échantillons de la boucle DSP sont encapsulés dans un paquet RTP puis envoyé au réseau. L'estampillage se fait à l'échantillon, (l'incrément par paquet est de n).

Le format des échantillons étant variable (nombre de canaux variable et fréquence d'échantillonnage non spécifié) il est nécessaire que les deux instances de jMax (celles du producteur et du consommateur) soit configurées sur la même fréquence d'échantillonnage et sur le même nombre de canaux. Cela pose aussi un problème en ce qui concerne le format des données transportées par RTP (payload). En effet, il n'existe aucun payload standard possédant plus de deux canaux simultanés et il n'y a que peu de choix pour le son haute qualité. Nous avons choisi d'utiliser un numéro de payload non utilisé. Cela rend notre prototype incompatible en émission avec les autres logiciels utilisant RTP (il ne pourront pas comprendre le flux issu de jMax). Toutefois il reste possible d'implémenter une émission avec un payload existant si le son produit y correspond. Il est aussi possible de traiter les informations reçues par jMax selon le payload du paquet, rendant jMax compatible avec d'autres logiciels RTP (comme VAT par exemple [JM92]).

3.2 Les objets RTPin et RTPout, faire coopérer les musiciens

Désirant obtenir des informations statistiques sur les flux, il fut convenu d'abandonner le périphérique RTPaudioport. En effet, il n'est pas possible d'obtenir des outlets¹⁰ sur un port audio dans jMax. D'autre part, il est nécessaire de simplifier la configuration de jMax (la configuration des périphériques audio comme le MIDI, OSS, etc...est amenée à changer dans les futures versions de jMax).

3.2.1 Synchronisation des flux des différents musiciens entre eux

Le périphérique RTPaudioport fournissant le transport du son entre un producteur et un auditeur, il fallait un mécanisme pour que les musiciens puissent s'entendre

⁹Digital Signal Processing : traitement du signal numérique

¹⁰sortie d'un objet jMax

de façon synchronisée. Sachant que pour un échange entre deux instances de jMax, chaque musicien soit à la fois producteur et auditeur.

Intégrer au flux reçu par le RTPaudioport une sortie donnant la latence entre la source et le RTPaudioport étant impossible, la solution proposée fut d'écrire un objet DSP "RTPout" (pour la réception du flux) dont l'Outlet serait la latence estimée et dont la seule Inlet serait le flux à envoyer. Rappelons qu'avec RTP, la latence est estimée chez l'émetteur.

Le travail immédiat consistait donc à développer deux objets DSP "RTPin" et "RTPout" modélisant la réception et l'envoi d'un flux. L'objet "RTPout" devra fournir un Outlet informant l'utilisateur de la latence entre la production et la réception du flux de l'autre instance de jMax.

Fournir la latence au musicien lui permettra de choisir sa stratégie d'interaction avec le ou les autres musiciens distants. Par exemple, si les musiciens désirent obtenir une interaction l'un vis à vis de l'autre, ils pourront placer un objet de retard (insérant un retard en fonction de la latence constatée) sur leurs micros (ou sur le son qu'ils produisent). De cette façon, chacun des musiciens jouera en avance par rapport à ce qu'ils entendent. En contrepartie, ils pourront tous s'entendre de façon synchronisée et en direct (si bien entendu ils mettent tous en place le même mécanisme).

Cette stratégie laisse aux musiciens le choix d'une interaction unidirectionnelle ou bidirectionnelle.

3.2.2 Résolution du problème de dérive des horloges d'échantillonnages entre les cartes sons des musiciens

Considérons une interaction entre deux musiciens en temps réel sur un réseau. Le premier musicien va produire du son, ce son sera échantillonné à une certaine fréquence cadencée par l'horloge de sa carte son. Le deuxième va consommer les échantillons à la même fréquence mais cadencée elle aussi par sa propre carte son. Si les horloges des cartes sons des deux musiciens ne sont pas synchronisées, nous pouvons rencontrer des problèmes de famine ou d'engorgement d'échantillons. Par exemple, le son produit par l'un des musiciens sera échantillonné à la fréquence de l'horloge de sa carte son locale. L'horloge peut échantillonner à 44099 Hz alors que la carte son prétendra échantillonner à 44100Hz. Une fois le son envoyé, il sera consommé à la fréquence prétendue par la source, soit 44100Hz. Dans ce cas, le récepteur consomme les échantillons plus vite que l'émetteur en produit. Nous risquons alors de vider (ou de surcharger dans le cas inverse) le tampon de lecture. Cela

provoquera des clics non désirés sur le son. [Fob02, MST99] proposent une méthode de compensation de ce problème.

Il est alors nécessaire de mettre en place un dispositif de compensation de l'erreur.

Pour résoudre ce problème, trois stratégies furent étudiées :

- Calculer la dérive des horloges puis resynchroniser.
- Synchroniser les horloges des cartes son par GPS
- Surveiller l'état du tampon de réception puis compenser le retard ou l'avance avec un deuxième tampon.

La synchronisation des horloges des cartes sons n'a pas été retenue. En effet, cela demande du matériel supplémentaire pour chacun des musiciens. Cela suppose aussi une carte son possédant une entrée horloge. Seules les cartes sons professionnelles disposent de cette entrée et cela risque de restreindre le nombre d'utilisateurs potentiels des objets RTPin et RTPout.

Le calcul de la dérive des horloges afin de resynchroniser les flux peut être obtenu en synchronisant les horloges des différents terminaux à l'aide du protocole NTP (Network Time Protocol). Il suffit ensuite de comparer les échantillons produits en un temps t chez l'émetteur avec le nombre d'échantillons consommés chez le récepteur pendant ce temps t . De cette façon, nous pourrions obtenir la dérive des horloges. Dans ce cas, il est alors possible d'échantillonner 43999 échantillons en 44000 (pour l'exemple précédent)

La troisième solution suggère de faire des statistiques sur la variation de la taille du tampon de réception. Lorsque nous remarquerons une baisse (ou augmentation) significative du nombre d'échantillons dans le tampon, il faudra alors compenser le décalage. La compensation de ce décalage s'effectuerait grâce à un deuxième tampon. Lors de la réception d'échantillons, deux tampons seraient remplis. Quand un échantillon sera consommé dans le tampon audio, il restera dans le deuxième. Lorsque les statistiques détecteront une famine ou une surcharge, le tampon de sauvegarde deviendra le tampon audio. L'indice courant du nouveau tampon sera choisi de façon à compenser l'erreur d'échantillonnage. La transition entre les deux tampons se fera grâce à un "fade" (jeu de volume entre les deux tampons).

Nous avons privilégié la solution statistique pour son faible coût en temps de travail CPU. Dans un premier temps, l'idée est de fournir le nombre de paquets présents dans le tampon audio au travers d'une Outlet de l'objet "RTPin". Un autre objet DSP (à développer) pourrait récolter le flux et les statistiques afin de compenser l'erreur.

3.2.3 Les objets RTPin et RTPout

La figure 11 montre un exemple d'utilisation de ces deux objets.

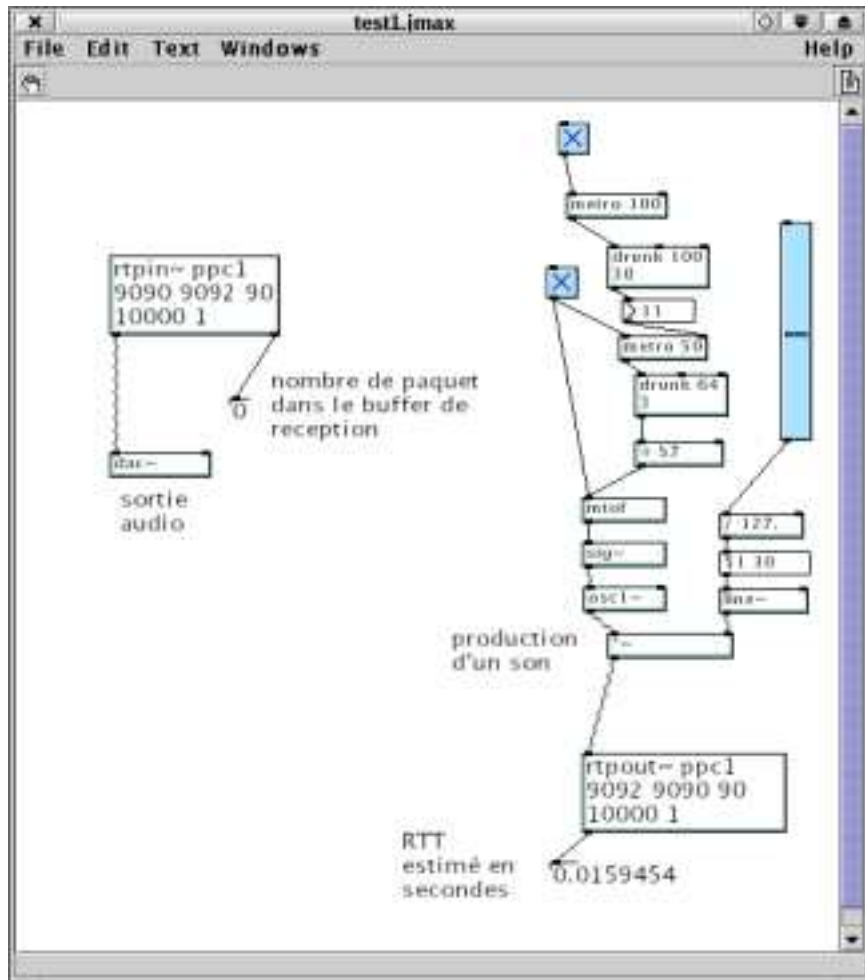


FIG. 11 – Exemple d'utilisation des objets RTPin et RTPout

Les paramètres des objets RTPout et RTPin sont :

- le nom de la machine distante (ou une adresse IP)
- le port entrant et sortant
- le TTL : Time To Live : nombre maximum de nœuds du réseau à traverser
- la bande passante que l'on laisse à disposition

Le son est envoyé à l'objet RTPout. Celui-ci se charge de l'envoyer à l'objet RTPin sur la machine distante au numéro de port indiqué par le port sortant. L'objet RTPin reçoit le flux audio pour le restituer en sortie.

L'émetteur (possédant l'objet RTPout) dispose d'une estimation du délai aller retour entre les deux objets RTPin et RTPout. Le récepteur dispose lui du nombre de paquets en attente dans son tampon audio.

Dans le cas où les musiciens désirent une interaction unidirectionnelle (chaque musicien est soit consommateur soit producteur), le récepteur va juste ouvrir les objets RTPin nécessaires à la réception de chacun des flux. Cette approche est directement adaptée à une communication point à point, c'est un choix qui a été fait.

Lorsque l'on désire une interaction directe entre tous les musiciens, chacun d'entre eux possédera les objets RTPin et RTPout nécessaires à l'envoi et à la réception de tous les flux. L'entente synchronisée chez chacun des musiciens se fera grâce à l'inclusion de retard sur d'une part ce que le musicien joue (un retard sur le son qu'il produit) et éventuellement sur ce qu'il reçoit des autres (le délai aller-retour obtenu étant estimé, il peut être nécessaire de rajouter de la latence).

3.2.4 Architecture des objets RTPin et RTPout

La figure 12 prend comme exemple une communication unidirectionnelle entre deux musiciens afin de détailler le fonctionnement de ceux-ci.

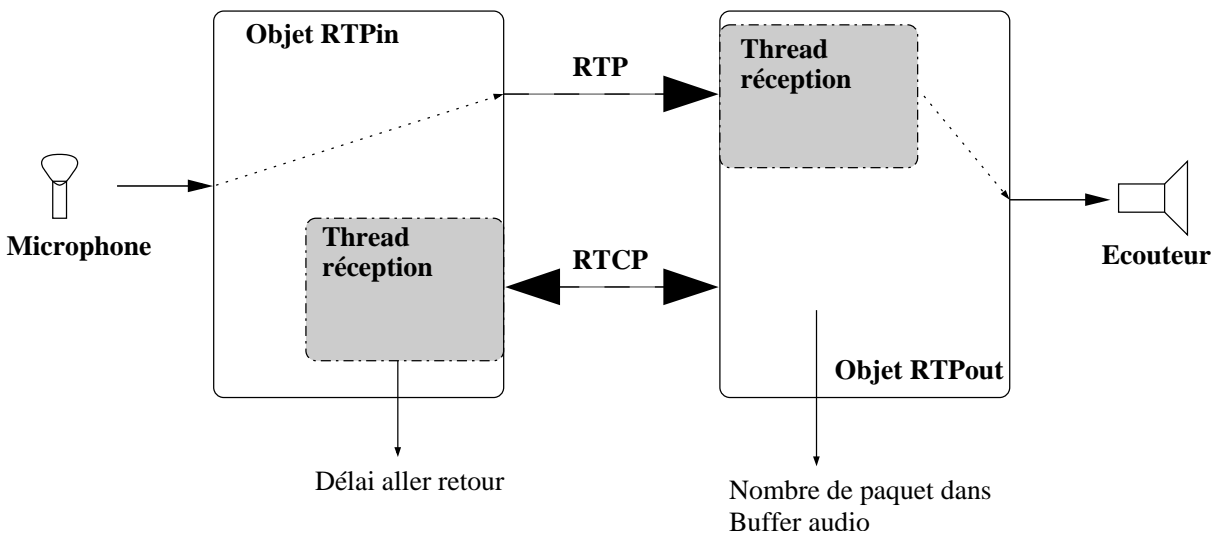


FIG. 12 – Architecture des objet RTPin et RTPout

Comme nous l'avons vu dans le paragraphe précédent, le son est envoyé directement à l'objet RTPin qui traite les échantillons reçus par nombre constant (boucle

DSP¹¹). A chaque boucle est envoyé un paquet RTP contenant les données récoltées. Le récepteur fait la même chose mais dans le sens inverse : il reçoit les paquets dans un tampon qui sera lu par la boucle de traitement pour être envoyé vers sa sortie (ici un haut-parleur).

Notons que le tampon maintient à jours le nombre de paquets qu'il contient. Grâce à un système d'évènements (les alarmes), à chaque boucle de traitement, le nombre de paquets dans le tampon audio est mis à jour en sortie de l'objet RTPin.

La bibliothèque RTP utilisée gère elle-même les messages RTCP, sur réception d'un rapport du récepteur (dans l'objet RTPin), un évènement déclenche le calcul le délai aller retour puis la transmission de celui-ci en sortie.

3.3 Problèmes rencontrés

3.3.1 Appréhension d'un nouveau domaine, l'audio numérique

L'audio numérique ne m'étant pas familière, il m'a fallu me documenter pour comprendre les enjeux du projet. La lecture de [RdR99] de Curtis Roads m'a permis de me plonger dans ce domaine et d'appréhender au mieux le logiciel jMax. L'utilisation de celui-ci pouvant être de très bas niveau (il est possible de travailler sur des tampons d'échantillon), il est crucial de connaître au maximum les subtilités de l'audio numérique.

Cette culture fut aussi nécessaire afin d'avoir un dialogue constructif avec les spécialistes du temps réel et de l'audio numérique à l'IRCAM (François Déchelle et Norbert Schnell). En effet, jMax est aussi un puissant outil temps réel, les objets de jMax ont un fonctionnement bien particulier : le programmeur d'un objet doit programmer une fonction de traitement des échantillons qui sera appelée régulièrement par FTS¹² lors de l'exécution. Les systèmes d'exploitations n'étant généralement pas temps réel, FTS se charge d'ordonnancer au mieux les fonctions de calculs des objets placés dans le programme de l'utilisateur. Le développement d'objets tels que RTPin et RTPout requiert donc des connaissances en audio numérique, en temps réel mais aussi en réseau et systèmes répartis.

3.3.2 L'installation et prise en main

L'histoire de la gestion du son sous Linux est complexe. Les pilotes historiques de son de Linux sont ceux de OSS¹³. Depuis Janvier 1998, certains pilotes OSS sont devenus des produits commerciaux. Dès lors, ALSA¹⁴ a vu le jour afin de continuer à

¹¹Digital Signal Processing

¹²Faster Than Sound, le noyau de jMax

¹³Open Sound System

¹⁴Advanced Linux Sound Architecture

faire évoluer les pilotes de son dans un cadre gratuit. jMax fonctionne avec les deux types de pilotes (un seul à la fois), à condition qu'il soit correctement configuré. Suite à de nombreux échecs de l'installation d'ALSA (problèmes de carte son, problèmes de version) OSS fut choisit comme pilote audio pour jMax. La configuration de celui-ci requiert juste une ligne dans le fichier `.jmaxrc` :

```
defaultAudio ossaudioport /dev/audio 2
```

Les deux paramètres sont le fichier interface avec OSS et le nombre de canaux utilisées.

3.3.3 Utilisation de la librairie RTP dans jMax

Pour les raisons évoquées dans le paragraphe 2, RTP est le protocole choisi pour transporter les échantillons audio à travers le réseau. La complexité de ce protocole nous a imposé l'utilisation d'une librairie existante, UCL¹⁵. Le lien entre cette librairie et l'interface d'objets de jMax ne fut pas facile. En effet, la réception des messages nécessite des temporisateurs et le traitement de ceux-ci un récupérateur d'événements (handler). jMax étant un outil temps réel, un appel bloquant ne doit pas être actif (utilise du temps de calcul CPU) au risque de faire échouer l'ordonnement de FTS. Les fonctions associées aux objets jMax sont exécutées directement par FTS, nous avons donc besoin de garder un lien vers l'objet (et donc ses paramètres) dans chacune des fonctions associées à cet objet. Le traitement des messages par le récupérateur d'évènement risquait de poser problème car il n'est pas possible d'utiliser des variables globales par objet (au risque de ne pouvoir se servir que d'un seul et unique de ces objets, i.e. un seul RTPin par exemple). Les auteurs de cette librairie ayant prévu une situation de ce type, il nous fut possible de passer en paramètre un lien vers l'objet jMax dans la fonction de récupération d'évènements. Comme le montre la figure 12 nous avons dû placer des tampons intermédiaires entre l'objet jMax et la librairie RTP.

3.3.4 Mesures

Le prototype des objets RTPin et RTPout représente environ 1000 lignes de code C. La difficulté étant principalement l'assemblage de la librairie RTP avec l'interface de programmation d'objet de jMax (la compréhension comprise). Il fut testé et implémenté sur un Linux SuSE 8.0. Le réseau sur lequel les tests ont été faits est un réseau local 100Mb par seconde avec deux machines, l'une (la machine *A*) possède un processeur de 660MHz et une distribution de Linux, la SuSE 8.0, l'autre (machine *B*) un processeur de 450MHz et une distribution de Linux, la SuSE 7.6.

¹⁵UCL Common Multimedia Library, <http://www-mice.cs.ucl.ac.uk/multimedia/software/common/>

La figure 13 nous montre la taille du tampon intermédiaire entre les paquets reçus du réseau et l'envoi de ceux-ci à la carte son. Cette mesure nous a permis de constater l'effet produit par la non synchronisation des horloges des cartes sons (voir 3.2.2). Cette mesure fut effectuée sur la machine *A* recevant un flux audio de la machine *B*. L'unité de temps est la seconde.

Nous voyons qu'au début de la réception, quatre paquets sont en attente dans le tampon de la machine *A*. Jusqu'à n'avoir plus aucun paquet dans le tampon intermédiaire, leur nombre décroît régulièrement à cause de la légère dérive des horloges (nous observons tout de même quelques fluctuations à cause du réseau). C'est au moment où le tampon contient 0 paquet qu'un clic est entendu sur la machine *A* (rien n'est lu dans le tampon, le clic entendu dure le temps du contenu d'un paquet). Le tampon ayant à ce moment là compensé le léger retard, nous reprenons un paquet d'avance. Finalement, le retard va reprendre le dessus et le tampon ne va contenir encore une fois aucun paquet, un nouveau clic va se produire. A partir de là, le phénomène se répétera en boucle avec une nombre de paquets dans le tampon oscillant entre 0 et 1. Dans ce cas là, la machine *A* consomme les échantillons plus vite que n'en produit la machine *B*.

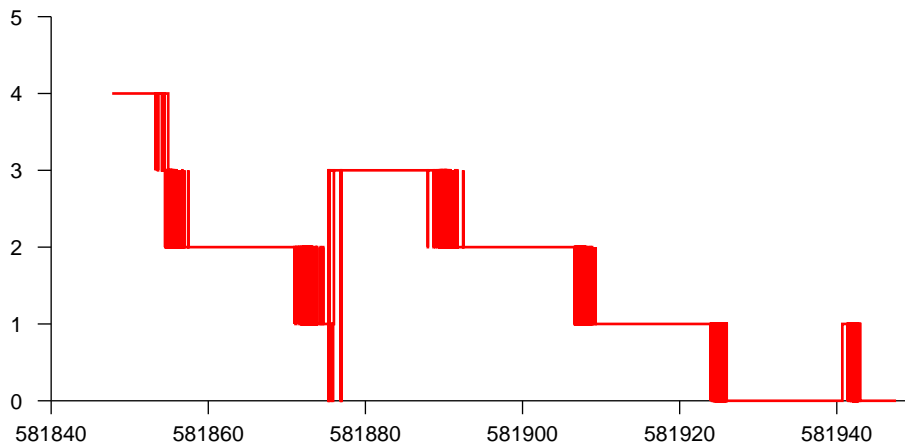


FIG. 13 – *Mesure de la taille du tampon audio*

La figure 14(a) nous montre le délai aller-retour estimé grâce à RTCP sur la machine *A* hébergeant à la fois l'émetteur et le récepteur. Le déroulement du temps est exprimé en secondes sur l'axe des abscisses, le délai est exprimé en secondes sur l'axe des ordonnées. On peut voir que le délai oscille entre 15,8 millisecondes et 16 millisecondes. Sachant qu'aucune perte n'est constatée, nous pouvons considérer le délai comme quasi-constant.

La figure 14(b) nous montre le délai aller-retour estimé grâce à RTCP sur la machine *A* émettant vers la machine *B*. Le délai oscille entre 15,6 et 16 millisecondes. Les résultats sont comparables au test en local, ce qui nous montre que le réseau utilisé pour le test est largement capable de supporter un flux de ce type.

4 Travaux futurs

Ce stage de DEA a permis de tester une approche d'interaction à distance entre plusieurs musiciens (qui ont été simulés pour les tests). Le prototype obtenu permet de faire jouer plusieurs musiciens en temps réel. Cependant, des améliorations sont encore nécessaires afin de simplifier l'utilisation des objets RTPin et RTPout. D'autre part, rien n'est encore disponible pour qu'un ingénieur du son puisse mixer les flux et les retransmettre au public.

4.1 Interaction entre les musiciens

Comme nous l'avons vu dans le paragraphe 3.2.3 un objet RTPin ou RTPout se connecte à un et un seul autre objet RTP. Dans le cas où plusieurs musiciens (plus que deux) désirent jouer de la musique ensemble au travers d'un réseau, ils vont tous devoir se connecter les uns aux autres (autant d'objets RTPin et RTPout). Cela risque de poser plusieurs problèmes. Tout d'abord, cela augmente la charge de calcul de jMax. En effet jMax utilise son propre ordonnanceur pour gérer l'exécution des boucles de traitement de chacun des objets. D'autre part, le réglage du délai à ajouter sur son propre flux ne sera pas suffisant, il faudra aussi ajouter un léger retard sur les flux de chacun car ceux-ci ne sont pas synchronisés entre eux. Le fait que le délai affiché par l'objet RTPout soit une estimation risque lui aussi de poser un problème supplémentaire : chaque flux sera synchronisé de façon approximative. Notons toutefois que ce problème est mineur pour une interaction à deux musiciens : une fois que les retards sont réglés, les deux musiciens peuvent se synchroniser l'un par rapport à l'autre à l'aide de leurs oreilles.

Une solution possible à ce problème serait d'utiliser un unique objet RTPin pour chacun des flux reçus. Cela permettrait à la fois une baisse des ressources nécessaires au traitement des échantillons mais aussi d'avoir une gestion unique des flux reçus. Le problème étant principalement leur synchronisation, cette méthode permettrait de simplifier les réglages. Les flux étant datés à l'échantillon, il suffirait de jouer au même moment les échantillons de chacun d'eux possédant la même estampille, la synchronisation se fera sur l'échantillon et non sur le temps. De plus, pour éviter d'avoir des décalages important de numéro d'échantillons entre les différents musiciens, il faudra mettre en place un dispositif d'initiation de l'interaction afin de partir à peu près en même temps au même numéro d'échantillon. Le protocole SIP¹⁶ [RSC⁺02] pourrait être utilisé dans cette optique.

Un orchestre à plusieurs musiciens jouant à distance reste donc une vision réaliste, même sans réglages de délai. Un chef d'orchestre pourrait jouer le rôle de métronome pour le groupe. L'instance de jMax hébergeant celui-ci enverrait le flux qu'il produit

¹⁶Session Initiation Protocol

(n'importe quel instrument ou rythme). Les autres musiciens pourraient recevoir ce flux et renvoyer au chef d'orchestre le nouveau flux avec l'estampille correspondante à l'estampille reçue du chef. La synchronisation entre tous les musiciens se ferait alors grâce au chef d'orchestre et sans réglage. De plus, chacun d'eux s'entendra sans décalage (mais n'entendra pas les autres). En contrepartie, chaque musicien n'entendra que le chef d'orchestre. Dans ce cas, la structure du morceau joué devra être préalablement définie. Si elle ne l'est pas, nous pouvons imaginer un deuxième canal du chef d'orchestre vers les musiciens ou celui-ci donnerait les indications nécessaires au déroulement du morceau. C'est la situation illustrée par la figure 15.

Comme nous l'avons précisé au paragraphe 1.3, le son transporté est non compressé par souci de qualité. Afin de conserver une qualité optimale, le mécanisme de compensation de dérive des horloges d'échantillonnage des cartes sons devra être ajouté. Il reste les pertes possibles sur le réseau, un mécanisme du type de ceux présentés dans le paragraphe 2.4 pourra être inclut dans le prochain prototype.

4.2 Le mixage et le publique

Hans-Nicolas Locher est actuellement en stage sur le contrôle commande à distance de l'ingénieur du son. L'idée (s'appuyant sur le concept de messagerie industrielle temps réel) est de superviser et de commander en temps réel les périphériques d'une salle de restitution et d'acquisition du son en trois dimensions. L'ingénieur devra gérer le mixage au sens classique du terme (réglage des volumes) mais aussi les positions spatiales des sons dans la pièce.

Il est aussi envisageable que l'ingénieur du son puisse mixer le son dans un format plus classique (MP3 ou PCM par exemple) afin que des auditeurs puissent entendre le concert sur Internet. Le mixage pourrait dans ce cas là se faire de façon automatique en prévoyant un mode auditeur dans jMax où l'on recevrait les flux de tout le monde. Un mécanisme d'adaptation au réseau tel que ceux présentés dans le paragraphe 2.3 serait alors à envisager pour répondre au besoin de tous.

5 Conclusion

Le projet de l'orchestre et du concert virtuel réparti n'est pas projet un facile à appréhender. L'interaction entre les musiciens est délicate, il est crucial de contraindre le moins possible les musiciens tout en ayant une interaction et un confort d'entente les plus précis possible. L'étude préliminaire présentée ici est très importante pour le bon déroulement de la suite du projet. Elle a permis de déterminer quels outils restent à notre disposition pour faire progresser le projet, mais aussi d'avoir une vision réaliste de ce qu'il est possible de faire, cela notamment grâce au domaine des logiciels d'audio-conférences et du domaine de la téléphonie sur IP. Elle a permis aussi de prévoir à l'avance les problèmes que nous risquons de rencontrer lors d'une utilisation à plus large échelle (sur Internet) de notre outil, mais aussi d'envisager comment les résoudre grâce au contrôle de congestion et à la compensation d'erreur.

Le prototype issu du stage permet d'envoyer du son entre deux instances du logiciel jMax. Le choix de la gestion de l'interaction entre les musiciens est laissé à ceux-ci grâce à la disponibilité d'une information, le délai aller retour entre les deux musiciens. Cette approche nous a permis de comprendre comment améliorer le principe d'interaction entre les musiciens.

Les résultats obtenus par le stage et par l'étude nous permettent d'être tout à fait optimistes sur le bon déroulement de la suite du projet.

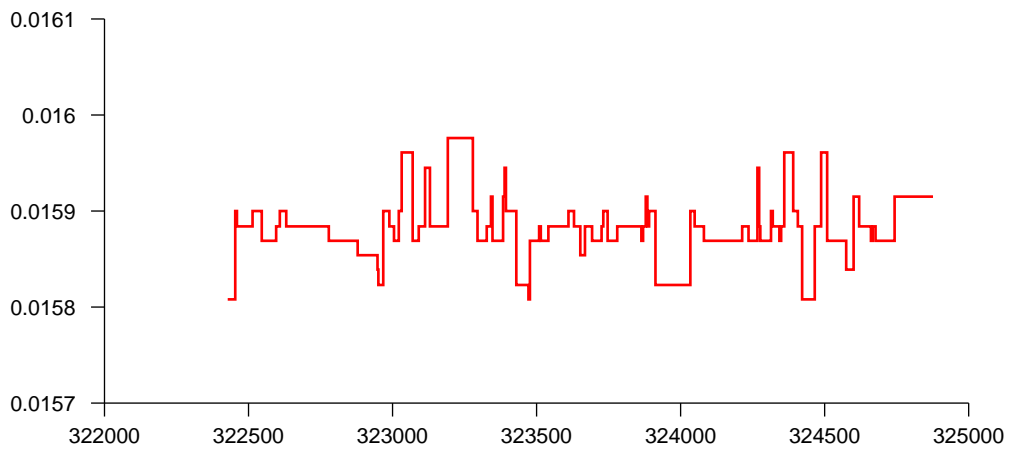
Références

- [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.
- [Bou02] Nicolas Bouillot. Transport du son produit en temps réel sur les réseaux best effort, 2002.
- [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.
- [CS01] J.R. Cooperstock and S. Spackman. The recording studio that spanned a continent. *IEEE International Conference on Web Delivering of Music (WEDELMUSIC), Florence*, 2001.
- [DBC⁺98a] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rován, and N. Schnell. jmax : a new JAVA-based editing and control system for real-time musical applications. *Proceedings of the International Computer Music Conference*, 1998.
- [DBC⁺98b] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rován, and N. Schnell. jmax : demonstration of an integrated environment for real time musical applications. *Proceedings of the International Computer Music Conference*, 1998.
- [DBC⁺99] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rován, and N. Schnell. jmax : an environment for real-time musical applications. *ComputerMusic Journal*, 23(3) :50–58, 1999.
- [Déc00] F. Déchelle. jmax : un environnement pour la réalisation d’applications musicales temps réel sous linux. *Actes des journées d’Informatique Musicale*, 2000.
- [DHT95] C. Diot, C. Huitema, and T. Turetletti. Multimedia applications should be adaptive. *Proc. HPCS’95, Mystic (CN)*, 1995.
- [Duf00] F. Dufourd. Streaming audio temps réel sur réseau avec jmax, 2000.

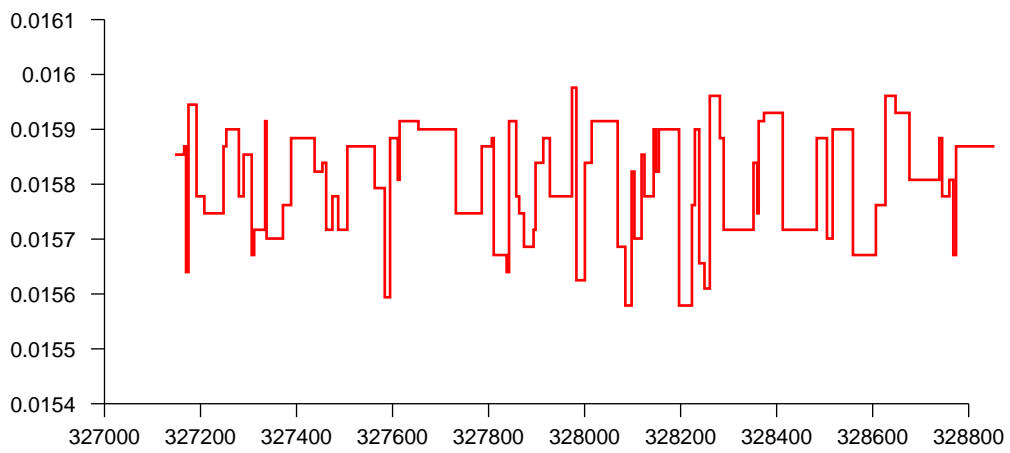
- [FJL⁺97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6) :784–803, 1997.
- [Fob02] D. Fober. Audio cards clock skew compensation over a local network. Technical report, 2002.
- [FSL97] W. Feng, U. Syid, 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.
- [HSHW95] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. *Proceedings of INET, Oahu, Hawaii*, 1995.
- [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.
- [MST99] Sue Moon, Paul Skelley, and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *IEEE INFOCOM '99*, New York, mars 1999.
- [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.
- [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.
- [RdR99] Curtis Roads and Jean de Reydellet, editors. *L'audionumérique*. DUNOD, 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.
- [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.
- [RSC⁺02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip session initiation protocol rfc 3261, 2002.
- [SCFJ98] Schulzrinne, Casner, Frederick, and Jacobson. RTP : A transport protocol for real-time applications. *Internet-Draft ietf-avt-rtp-new-01.txt (work in progress)*, 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.
- [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 Avidah 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.



(A) *Mesure du délai aller-retour en local*



(B) *Mesure du délai aller-retour entre la machine A et la machine B*

FIG. 14 – Mesure du délai aller-retour

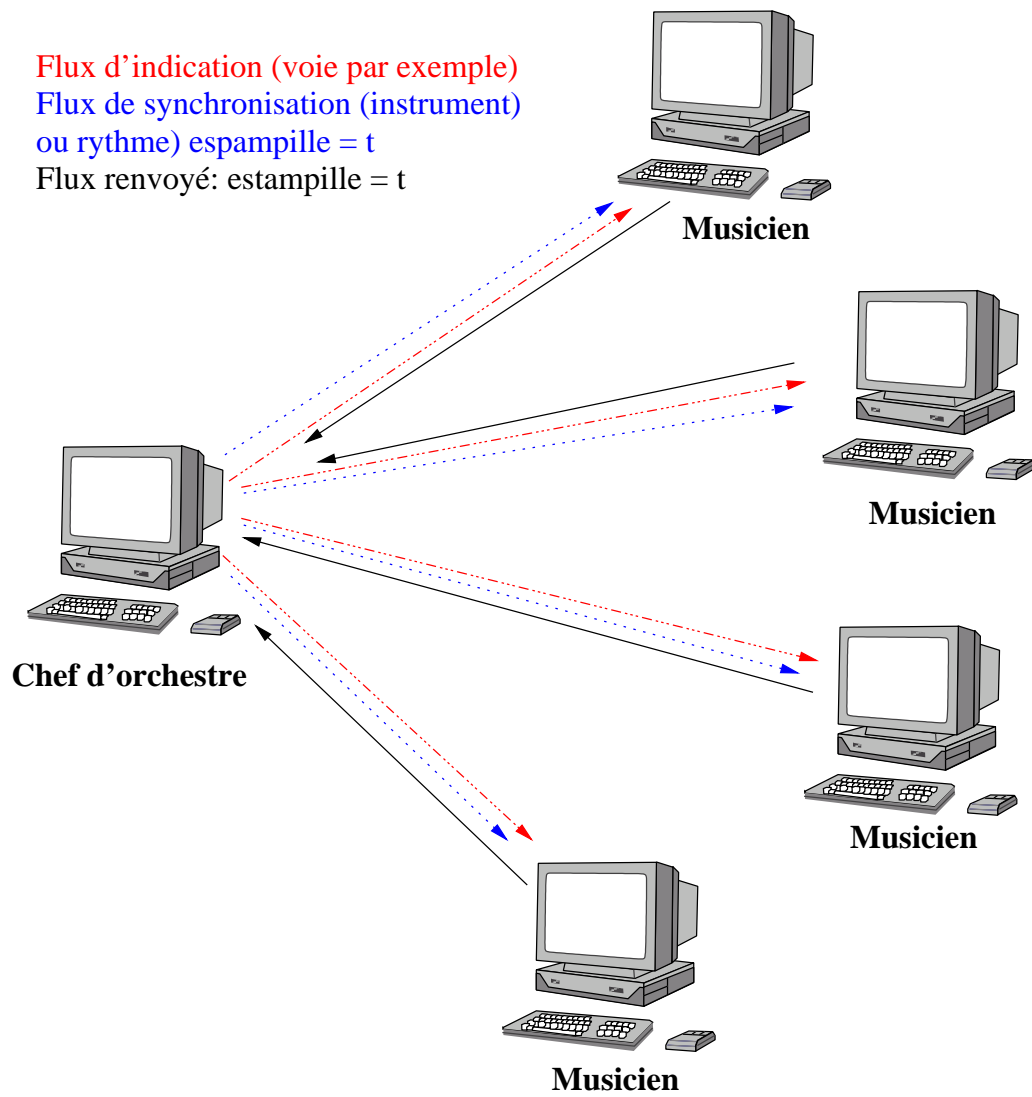


FIG. 15 – Exemple d'interaction sans délai sur son propre instrument