Conservatoire National des Arts et Métiers

La Cohérence Dans les Applications Multimédia Interactives : du concert réparti sur Internet aux jeux multi-joueurs en réseau

THÈSE

présentée et soutenue publiquement le 14 novembre 2006

pour l'obtention du

Doctorat du Conservatoire National des Arts et Métiers

(spécialité informatique)

par

Nicolas Bouillot

Composition du jury

Directeur de thèse :	Eric Gressier-Soudan	Professeur au CNAM de Paris
Président :	Stéphane Natkin	Professeur Titulaire de la Chaire Multimédia au CNAM de Paris et à l'ENJMIN de Angoulême
Rapporteurs :	Jeremy R. Cooperstock	Associate Professor à l'université McGill au Canada
	Isabelle Demeure	Professeur à l'ENST, Paris
	Michel Raynal	Professeur à l'université de Rennes 1
Examinateurs :	Sophie Chabridon	Maître de conférence à l'INT de Evry
	Gérard Florin	Professeur au CNAM de Paris
	Cécile Leprado	Professeur Associé au CNAM de Paris et à l'ENJMIN de Angoulême
	Jean Vareille	Maître de conférence à L'UBO, Brest

Centre d'Étude et de Recherche en Informatique du CNAM – EA 1395 – Ecole Doctorale EDITE

Remerciements

Je tiens tout d'abord à remercier Eric Gressier-Soudan pour avoir su me soutenir, m'orienter et m'encourager. Ce fut un réel plaisir de travailler dans un tel contexte.

Mes très vifs remerciements vont à madame Isabelle Demeure, monsieur Jeremy Cooperstock et monsieur Michel Raynal pour m'avoir fait l'honneur d'accepter d'être les rapporteurs de cette thèse. Merci pour l'intérêt que vous y avez accordé.

Mes très vifs remerciements vont également à Sophie Chabridon, Gérard Florin, Cécile Leprado, Stéphane Natkin et Jean Vareille pour avoir accepté de faire partie de mon jury de thèse.

Je remercie également les chercheurs du laboratoire CEDRIC du CNAM pour m'avoir accueilli. Les discussions que j'ai pu avoir avec vous m'ont beaucoup enrichi.

Je remercie tous ceux qui ont participé aux démonstrations et aux tests du concert réparti :
- *Résonances 2003* : Rémy Bonafous, Samundeswary Ramachandra, Hans-Nikolas Locher, François Dechelle, Patrice Tisserant, Laurent Ghys, Arnaud Gomes, le quartet KumQuat, Jean Lochard
- *Fête de la science 2005* : Vincent Roudaut, Laurent Alibert, Julien Cordry, Nathalie Machetot, Suan Ajirent, Joël Berthelin, Patrice Bouillot, Rémy Bonafous, Laurent Dehoey, Jean Paul Etienne, Sylvain Fouqueray, Hans-Nikolas Locher, Samundeswary Ramachandra, Rodrigo Andrade B. Almeida, Jean Vareille
- *Tests utilisateurs de nJam* : Michael Escande, Julien Cordry, José Camarena, Laurent Alibert, Jean Frédéric Etienne, Jean-Paul Etienne, Jean Marc Farinone, Olivier Veneri et Patrice Bouillot.

Je remercie chaleureusement Marie-Christine, Safia, Rachèle et Nina pour les relectures et critiques de mes documents, ainsi que tous mes ami(e)s pour leur soutien.

Je remercie finalement de tout mon cœur mes parents et Rachèle pour m'avoir soutenu, fait confiance et encouragé durant ma thèse.

à Jeannine, Marie-Christine et Patrice

Résumé

Les Applications Multimédia Interactives et Distribuées (AMID) sur Internet ouvrent des perspectives nouvelles de travail et de loisir entre utilisateurs du réseau Internet. Dans ces applications, le sentiment de co-présence est obtenu par la numérisation et l'envoi sur le réseau de certaines de leurs actions. Cependant, le délai de transmission de ces actions introduit des ambiguïtés temporelles au niveau de l'interface. Ce problème est connu sous le nom de cohérence, mais il est généralement traité du point de vue de l'ordre dans lequel les actions sont prises en compte, et non du point de vue du temps physique. Afin de prendre en compte les contraintes psycho-perceptives liées à l'utilisation du système, nous proposons un formalisme pour l'étude et l'analyse de modèles de cohérence qui permet de spécifier les propriétés temporelles et les propriétés d'ordonnancement. Les propriétés développées dans ce formalisme sont l'instantanéité, la simultanéité, la Δ légalité et les conflits d'ordonnancement. Nous avons expérimenté un de nos protocoles de cohérence dans nJam, notre prototype de Performance Musicale Interactive et Distribuée (PMID). Pour ce type d'applications, nos résultats permettent de construire un métronome réparti partagé entre musiciens distants et cela malgré des latences importantes sur le réseau. A l'aide d'expérimentations publiques et de tests effectués avec des utilisateurs, nous montrons l'efficacité de nJam. De plus, nous montrons que nos résultats se transposent dans le domaine des jeux vidéo distribués et qu'ils peuvent améliorer les performances des mécanismes de cohérence existants dans ce domaine. Actuellement, dans le domaine des AMID, notre solution distribuée est la seule qui prend en compte les latences réseau tout en fournissant les propriétés définies par la Cohérence Perceptive.

Mots-clés : Applications Multimédia Interactives et Distribuées, modèles de cohérence, temps physique, instantanéité, simultanéité, Δ légalité, conflits d'ordonnancements, psycho-perception, Performances Musicales Interactives et Distribuées, jeux vidéo multijoueurs distribués.

Abstract

The design of Distributed Interactive Multimedia Applications (DIMA) on Internet opens new perspectives to work and to have fun in a collaborative way. In these applications, the feeling of co-presence between users is obtained with a digitizing of their actions and a network. However, network transmission delays introduce temporal ambiguities on the user interface. This problem is well-known under the name of consistency, but it is generally treated from an order point of view, instead of physical time one. Thus, we provide formal descriptions to study consistency models. It includes temporal (instantaneity, simultaneity and Δ legality) and scheduling properties and allows to take into account psycho-perceptive constraints. From these formal definitions, we propose some consistency models and consistency protocols. We implement one of them in nJam, our Networked Musical Performance system (NMP). Our results allow to build a distributed metronome shared between remote musicians, in spite of high latencies on the network. Thanks to public demonstrations and user experiments, we show the practical effectiveness of nJam. Moreover, we show that our results can improve performance of distributed games. Currently, in the DIMA domain, our results are the only ones that take into account network latencies while providing the Perceptive Consistency in a fully distributed way.

Keywords : Distributed Interactive Multimedia Applications, Consistency models, physical time, instantaneity, simultaneity, Δ legality, psycho-perception, Networked Musical Performance, distributed multiplayers games.

Table des matières

Table des figures

Première partie

Présentation du sujet et son contexte

Dans leurs usages, les machines à communiquer sont encore aujourd'hui, dans bien des cas, des machines à gérer des simulacres de présence.

[Jacques Perriault]
La logique de l'usage - Essai sur les machines à communiquer, 1988

1

Introduction : les Applications Multimédia Interactives et Distribuées

Sommaire

1.1 La recherche dans le domaine du multimédia

La notion de multimédia est apparue pour la première fois dans un système informatique à la fin des années soixante, avec un document numérique combinant du texte et des images [RJ05]. Peu de temps après, les médias continus comme l'audio et la vidéo ont été intégrés dans les applications, introduisant des nouvelles problématiques telles que la gestion du temps physique et la synchronisation des médias.

Par nature, tout objet Multimédia est construit pour être lu, vu, entendu et plus généralement *perçu* par un utilisateur. C'est pour cela que la recherche dans ce domaine est fondamentalement pluridisciplinaire : communauté système, réseau, traitement du signal, visualisation, base de données, interface hommes/machines, sciences cognitives, mais aussi les communautés spécifiques à certains types d'applications (éducation, architecture, art, etc.).

Une publication récente [RJ05] du groupe de recherche ACM SIGMM (Special Interest Group on Multimedia) propose un rapport de synthèse de l'opinion de représentants de la communauté scientifique sur l'état actuel de la recherche en Multimédia, ainsi que sur ses futures tendances. Il en résulte que jusqu'au milieu des années 90, la recherche en multimédia s'articulait principalement sur des problématiques techniques (entrées-sorties, ordonnancement, codage, compression, transmission réseau, etc.). Par la suite, la recherche s'est plutôt focalisée sur le développement d'infrastructures plus haut niveau prenant en compte des caractéristiques "temps réel" telles que les échéances ou l'isochronie.

Plusieurs principes forts semblent ressortir de cette décennie : les approches statistiques et les approches adaptatives (tant d'un point de vue système ou réseau qu'applicatif), mais aussi l'importance de la qualité de service.

Les trois grands défis qui ont été identifiés pour la recherche à venir sont les suivants :

- *Créer des outils et des solutions simples pour la construction d'objets multimédia complexes.* Cette problématique prend en compte le fait qu'un objet multimédia est en fait l'expression d'un spécialiste, d'un artiste, ou plus généralement d'un auteur qui cherche à faire passer un message quel qu'il soit. Ainsi, les outils de construction d'objets multimédia doivent être capables d'intégrer des outils et des fonctionnalités diverses et plus ou moins spécifiques au métier de l'auteur mais surtout le plus simplement et le plus efficacement possible.
- *Rendre les interactions entre êtres humains distants les plus proches possibles des interactions que les humains vivent localement avec d'autres humains et/ou avec l'environnement "physique".* Les principaux problèmes soulevés sont d'ordre émotionnel : l'immersion de l'utilisateur dans le système ainsi que le sentiment de collaborer avec d'autres à travers un système informatique. Le domaine des jeux vidéo multijoueurs est un exemple dans lequel cette problématique semble avoir été prise en compte le plus rapidement.
- *La capture, le stockage, la recherche et l'utilisation (et plus généralement la gestion) des objets multimédia dans l'environnement informatique.* En effet, les outils multimédia restent encore le plus souvent des outils de structuration des données, sans véritable lien entre le stockage du média et son contenu sémantique. Par exemple, le passage automatique du texte à la parole, la recherche d'un entretien radiophonique dans un système d'archive à partir d'une recherche textuelle, l'organisation des vidéos familiales, etc.

Les thèmes développés dans cette thèse sont étroitement liés au deuxième point. En effet, les Applications Multimédia Interactives et Distribuées (AMID) permettent à leurs utilisateurs de

collaborer en temps réel.

1.2 Classifications des applications Multimédia Interactives

L'interactivité supportée par un système informatique est une notion complexe qui retient l'attention de plusieurs communautés de recherche d'horizons différents. Ainsi, l'interactivité est définie différemment selon les critères de chaque communauté (Systèmes répartis, Interaction Homme Machine, Artistique, etc.). Lors de la conférence CSCW (Computer Supported Collaborative Work) de 1996, Steve Benford *et al.* [BBRG96] ont proposé une classification des AMID, selon trois axes :

- La *téléportation* qui caractérise l'ensemble des informations qui sont distantes mais qui se manifestent localement par un simulacre. [1]
- L'*artificialité* qui caractérise l'ensemble des informations prises en compte dans l'application qui sont issues de la perception d'événements réels simulés
- La *spatialité* qui caractérise le réalisme de la manifestation des simulacres dans l'interface, comme par exemple les dimensions de l'écran (projecteur, écran de téléphone portable), la position de l'utilisateur dans l'environnement (spatialisation du son ou pas), etc.

La figure 1.1 nous montre la classification des applications selon les critères de téléportation, d'artificialité et de spatialité.

L'*artificialité* est une notion complexe qui pose des problèmes de conception à différents niveaux. Elle soulève des problèmes techniques nouveaux lors de l'intégration d'éléments réels dans le virtuel (ou inversement). Par exemple, les systèmes de réalité augmentée peuvent être considérés comme des systèmes d'effets spéciaux temps réel (intégration graphique en temps réel [CLN+05]). Elle soulève aussi des questions en terme de génie logiciel car les transitions du monde virtuel vers le monde réel sont difficiles à décrire [NY05].

Les critères de *téléportation* et de *spatialité* se justifient surtout dans la partie virtuelle de l'application. En effet, il est nécessaire que les utilisateurs distants soient rapprochés à l'aide d'informations représentant la présence et les actions des autres (téléportation) et que la présentation des informations "téléportées" soit la plus réaliste possible en terme de perception (spatialité).

Avant de développer la problématique, nous pouvons mentionner les exemples suivants d'Applications Multimédia Interactives et Distribuées : la réalité augmentée, la réalité virtuelle, les jeux vidéo, les performances musicales distribuées (qui feront l'objet du chapitre 3), le théâtre virtuel, les groupwares interactifs (tableau blanc, etc.), les simulations militaires coopératives, les espaces partagés (shared spaces), la téléphonie sur IP, l'audio/vidéo conférence.

Nous allons maintenant proposer une définition de l'interactivité, du point de vue de l'utilisateur, puisque celui-ci manipule directement l'application en réaction aux simulacres mis en œuvre. Une des principales contraintes de la partie interactive de ce type de systèmes est de provoquer chez les utilisateurs les sentiments de co-présence [MRWJ03] et d'immersion, permettant de lui donner l'impression d'interagir avec un autre, et non avec une application. Ces sentiments seront obtenus à partir du moment où il va retrouver dans l'utilisation du système un certain nombre de signes, pratiques auxquelles il est accoutumé dans son environnement quotidien. En d'autres termes, dans un système interactif informatisé, la simulation des caractéristiques psycho-

[1] Un simulacre est ce qui n'a que l'apparence de ce qu'il prétend être. Par exemple dans une AMID, la transmission de vidéos est un simulacre de la vision d'événements et d'objets réels au cours du temps.

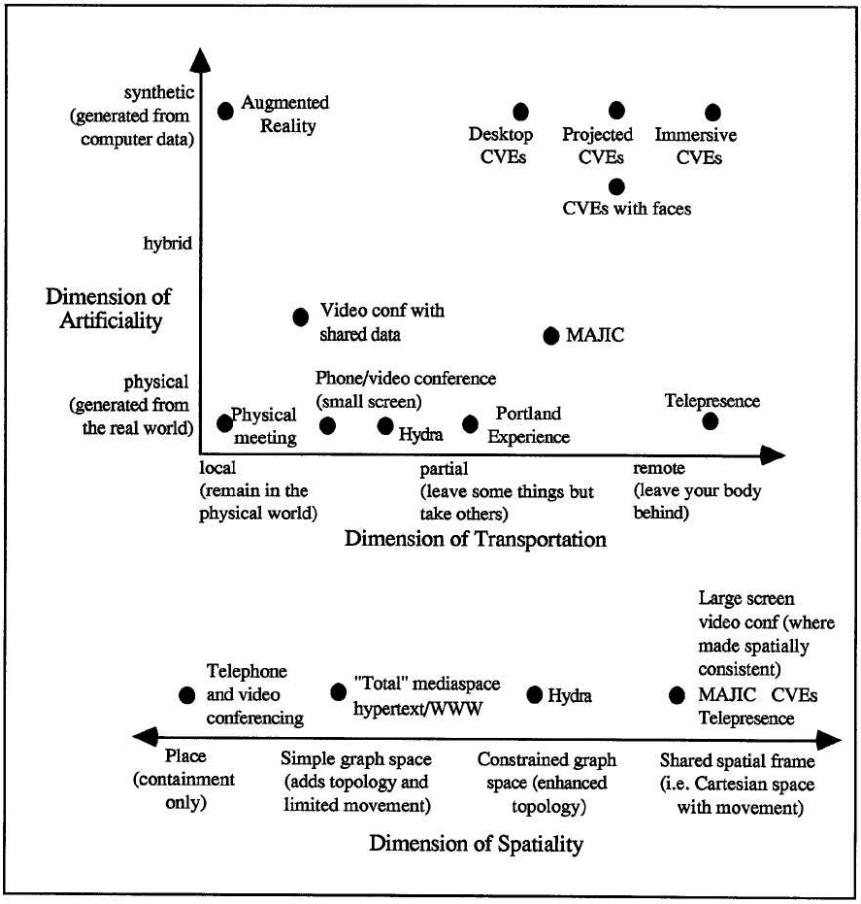


FIG. 1.1 – *La classification des Applications Multimédia Interactives par Steve Benford et al. (CVE est l'acronyme de Collaborative Virtual Environments)*

perceptives rend l'utilisation de l'application plus "naturelle"[2]. Sans tenir compte des contraintes propres à des cas d'utilisations précis, ces sensations vont être obtenues en construisant des *simulacres* de la réalité physique qui vont être perçus par les utilisateurs. Ils peuvent être d'ordre sociaux, sensoriels ou représentatifs.

Les *simulacres sensoriels* sont la base du processus d'interaction et/ou de co-présence puisqu'ils sont sensés être perçus par l'un des cinq sens humain : le codage des sons et des images ainsi que leurs systèmes d'acquisition/restitution sont des simulacres sensoriels largement étudiés et maîtrisés. Nous pouvons aussi mentionner toutes les recherches concernant les systèmes haptiques[3] et les interfaces maniables par contacts (claviers, souris, tablettes graphiques, etc.) tandis que les systèmes de simulacres sensoriels liés au goût et à l'odorat ne semblent pas avoir actuellement de bases technologiques suffisamment solides.

[2]L'intégration de paramètres psycho-perceptifs se retrouve par exemple dans les systèmes de codage de la voix où la bande de fréquences choisie dans l'acquisition correspond aux fréquences utilisées par la voix humaine.

[3]La perception haptique se réfère en grande partie à l'effet entre le mouvement de main et ses sensations dérivées, cutanés et/ou cinesthésiques, bien qu'elle soit parfois employée pour décrire des sensations vibratoires.

Les *simulacres représentatifs* correspondent aux sensations que le cerveau humain infère de l'interprétation des sens. Par exemple la notion d'espace peut être simulée par l'affichage d'un environnement en trois dimensions, la notion de position dans l'espace par de la spatialisation d'événements sonores, et la progression du temps peut être simulée par des systèmes de persistance d'états, par des événements périodiques, etc.

Le succès des systèmes de messageries instantanées comme MSN messenger, Yahoo messenger, etc. est probablement dû aux *simulacres sociaux* introduits par leurs concepteurs. En effet, l'utilisation de *smileys*, de listes de contacts ainsi que la visualisation de l'état d'activité des correspondants distants (absent, au téléphone, parti déjeuner, etc.) rend l'application plus "humaine" aux yeux de l'utilisateur. En fait, l'information même suggestive de la présence d'autres personnes semble être un élément motivant dans le processus de collaboration [EKLL03].

Nous pouvons alors redéfinir la téléportation et la spatialité avec les différents types de simulacres :

- La téléportation est définie par l'ensemble des simulacres sociaux et sensoriels
- La spatialité est définie par l'ensemble des simulacres représentatifs

Comme le suggère la notion de téléportation, les utilisateurs des AMID peuvent être distants. Cela suppose que l'application dispose d'un système de communication. Aujourd'hui, les réseaux les plus largement déployés sont la téléphonie mobile et l'Internet. Ces réseaux vont avoir des répercussions directes sur les applications car ils deviennent le support de transfert des informations, au même titre que le sont l'air et la lumière dans le monde réel. Ainsi, une AMID peut attendre différentes caractéristiques du réseau qu'elle utilise :

1- Que le réseau permette à l'application de faire communiquer un groupe d'utilisateurs (nous développerons cet aspect dans le chapitre 2 pour l'Internet).

2- Que le réseau permette à l'application de reproduire les simulacres sensoriels et représentatifs, notamment au niveau temporel avec les notions de continuité temporelle (ou encore appelée la Δ légalité), d'instantanéité et de simultanéité qui seront développées dans la deuxième partie de cette thèse.

1.3 Le multimédia du point de vue réseaux et systèmes répartis

Le champ d'utilisation du Multimédia dans l'Internet est en fait plus large que les Applications Multimédia Interactives. En effet, d'un point de vue système, les utilisateurs interagissent dans les AMID en étant à la fois acteurs et spectateurs de l'environnement simulé. On dit alors que la communication est de type n à m. D'autres applications, parfois dites interactives, fournissent une communication de 1 à 1, de 1 à n.

Les applications "1 à machine" ou "machine à 1" correspondent aux applications de streaming dans lesquelles les données multimédia sont envoyées en temps réel par un serveur. Dans cette catégorie, il y a les applications de transfert de flux vidéo en ligne (par exemple les bandes annonces de films ou la télévision sur IP) dans lesquelles les médias sont pré-enregistrés. Les applications de surveillance entrent aussi dans cette catégorie, mais les données vidéo/audio/etc. sont produites à la volée. Ces applications ont un intérêt commercial incontestable et utilisent les mêmes protocoles réseau que les applications multimédia avec communication de n à m.

La figure 1.2 nous présente les différentes catégories d'applications multimédia distribuées.

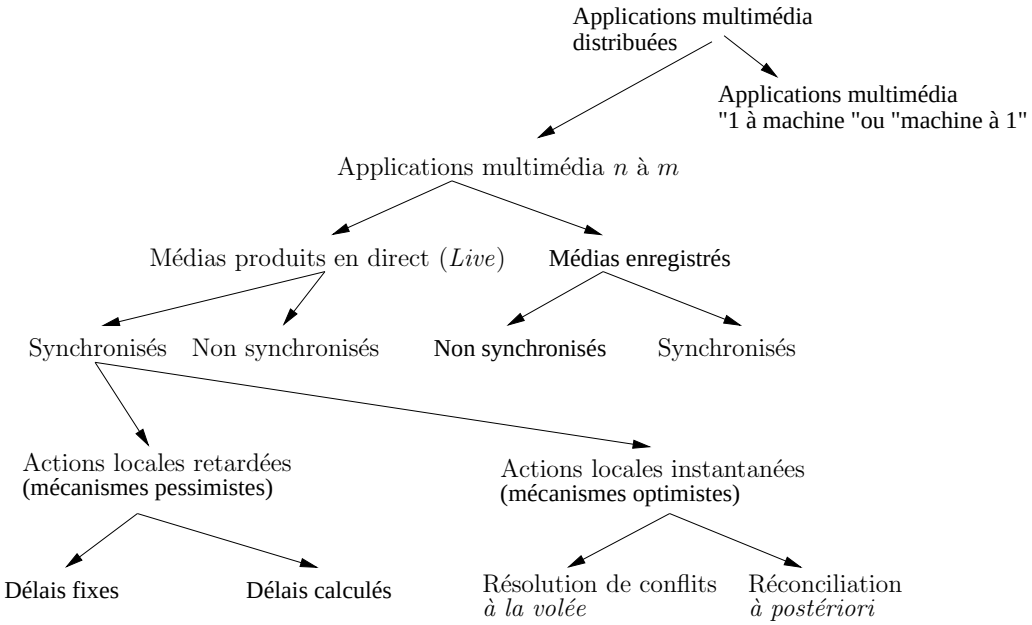


FIG. 1.2 – *Les entrées sorties, la synchronisation et les délais*

Ces applications ont toutes un point commun qui est l'utilisation de médias ayant des caractéristiques temps réel. Cependant, chacune d'entre elles aura ses propres caractéristiques : des plus simples avec la diffusion de médias pré-enregistrés aux plus complexes avec les AMID.

Les *médias enregistrés* sont des supports par définition passifs car ils ne sont pas le résultat direct de la manipulation du système par l'utilisateur (communication de 1 à 0). Malgré cela, les médias doivent être synchronisés dans certains cas (comme les transparents et le discours dans une présentation). Dans cette branche, les relations temporelles parmi les médias sont connues durant l'acquisition et un "timecode" (ou estampillage) commun peut être écrit directement dans le flux de données, que le média soit continu comme de l'audio ou discret comme des images. Ce timecode permettra au(x) récepteur(s) de lire les flux de façon synchronisée, même si les sources sont différentes.

Avec les *médias produits en direct*, les utilisateurs distants sont connectés les uns aux autres et sont capables de percevoir les actions effectuées par les autres. La catégorie *non-synchronisés* correspond aux applications où l'interaction consiste principalement à savoir que l'autre est là (appelées *see you see me* dans la littérature) tandis que la catégorie *synchronisés* correspond aux interactions avec des contraintes d'accès concurrents à des données ou avec des contraintes temporelles.

Dans l'approche *optimiste*, le système prend en compte immédiatement[4] des actions potentiellement conflictuelles. Cela peut engendrer une divergence d'états entre applications distantes. Pour réparer les erreurs introduites, un mécanisme va effectuer la convergence des états. Les techniques comme Timewarp [EM97], le Dead Reckoning [CLC99, AT00] s'effectuent à court

[4]Il faut comprendre par "immédiatement" le fait que le temps pris par une action pour être affiché est celui des périphériques et des couches logicielles locales.

terme tandis que la réconciliation [PSM03, KRSD01] et la gestion de versions [Ca02] s'effectuent à long terme.
Dans l'approche *pessimiste*, la synchronisation est effectuée avant l'affichage, introduisant un délai additionnel dans la notification d'une action locale au niveau de l'interface. Ces mécanismes ont un impact direct sur le temps de réponse de l'application : les délais devront donc être "bien" choisis, i.e. en fonction de l'application elle-même.

1.4 Problématique

Dans cette thèse, nous allons nous intéresser principalement aux concepts et aux mécanismes qui permettent d'élaborer des AMID qui intègrent des simulacres perceptifs et représentatifs. Nous allons pour cela développer la branche des applications interactives de type n à m avec des médias produits en direct, durant l'interaction.

Comme nous l'avons suggéré dans le début de ce chapitre, nous considérons que l'efficacité de telles applications est principalement due aux choix et à la conception des simulacres mis en œuvre dans l'application. Bien que nous allons garder à l'esprit le fait que ces simulacres sont multiples et de natures pluridisciplinaires (scénaristique, game-design, IHM, réseau, sociologie, etc.) nous allons nous focaliser sur les simulacres psycho-perceptifs, et particulièrement ceux liés à la perception du temps physique.

En effet, dans un contexte distribué, la notion de temps est cruciale car chaque partie du système possède sa propre perception des événements. Il est alors parfois nécessaire de contrôler globalement les dates physiques des opérations, voir leur ordre d'exécution. La formalisation de ce type de contraintes est effectuée avec la spécification de modèles de cohérence.

Aujourd'hui, il existe énormément de propositions de modèles de cohérence pour des applications telles que les Mémoires Réparties Partagées (MRP), les bases de données temps réel, etc. mais très peu pour les AMID. Cependant, avec la plupart des modèles actuels, il est impossible de formaliser des notions basées sur la perception du temps physique (comme par exemple la simultanéité entre événements) tout en connaissant les implications précises sur l'ordre d'exécution des opérations. Cela est dû notamment au fait que peu de modèles sont spécifiés à l'aide d'horloges physiques.

D'un autre coté, la communauté des chercheurs travaillant sur les AMID se focalise plutôt sur l'élaboration de simulacres perceptifs et sociaux. Ainsi, à notre connaissance, il n'existe aujourd'hui aucun cadre formel d'étude et d'analyse de modèles de cohérence permettant à la fois de spécifier des propriétés temporelles générales pour les AMID et de faire le lien avec les résultats obtenus sur la cohérence des données dans le domaine des MRP.

Nous estimons qu'un tel cadre est aujourd'hui nécessaire car pour les AMID, la gestion des données dans le temps physique fait appel à des caractéristiques qui sont peu étudiées dans les systèmes répartis. Ainsi, l'élaboration d'un formalisme de description de l'exécution répartie des AMID et la proposition de formalisation d'abstractions de haut niveau devraient permettre d'améliorer les performances de la gestion des données réparties. C'est pourquoi nous introduisons la notion de temps physique dans nos formalisations des AMID et des modèles de cohérence, en tenant compte à la fois des latences/gigues du réseau et de critères psycho-perceptifs.

Dans l'état actuel des choses, les AMID ayant des contraintes temps réel (de par l'interactivité qu'elles fournissent) utilisent des protocoles de cohérence qui sont souvent inappropriés, et quand

ils sont adaptés, il sont implantés sous forme centralisée. Les protocoles de cohérence pour les AMID complètement répartis ne tiennent pas compte des délais effectifs provoqués par le réseau (et le système matériel et logiciel). Il n'existe donc pas aujourd'hui de protocole de cohérence complètement distribué pour les AMID qui s'adapte à la latence du réseau. C'est pourquoi nous allons proposer des protocoles de cohérence et des mécanismes additionnels qui tiennent compte à la fois de la latence réseau et des contraintes psycho-perceptives des interactions des applications.

Comme cadre d'expérimentation, nous avons choisi de développer un système de Performances Musicales Interactives et Distribuées (PMID). En effet l'interactivité musicale est très dépendante de la perception dans le temps des notes, des sons, des événements audio, mais aussi de la notion de rythme, de cadence, de vitesse, etc. Autrement dit, l'exécution musicale interactive et distribuée est un domaine dans lequel il est crucial de maîtriser les délais et la cohérence. Initialement, ce projet non financé s'est monté en collaboration avec l'IRCAM[5] à travers la participation de François Déchelle et de Patrice Tisserant, programmeurs du logiciel jMax[6].

L'interactivité musicale possède des contraintes propres, notamment du point de vue temporel. En effet, dans le domaine des PMID la latence pose un problème important : les musiciens s'entendent les uns les autres avec un retard. Cela les empêche d'entendre les notes et les sons de façon simultanée, et donc d'avoir un tempo commun. Ce problème, bien que souvent mentionné par la communauté, n'a jamais été résolu dans le cadre de transmission de données audio échantillonnées. La difficulté de ce problème est que la solution doit tenir compte des délais réseau, du morceau de musique joué par les musiciens et des protocoles réseau utilisés. Notre principale solution est basée sur l'un de nos protocoles de cohérence, qui permet notamment de construire un métronome réparti à l'aide de l'introduction de retards locaux dans les retours des musiciens. Nous verrons comment cette solution a permis d'effectuer deux démonstrations de notre système devant un public.

Les recherches actuelles dans ce domaine traitent rarement de la gestion des latences, et considèrent que les avancées technologiques dans le domaine des réseaux devraient permettre d'avoir des latences suffisamment faibles pour permettre une interactivité musicale identique à celle obtenue en salle de concert. Cela est partiellement faux lorsque l'on prend en compte le fait que la vitesse de propagation de l'information est encore aujourd'hui limitée physiquement par la vitesse de propagation de la lumière. Les solutions que nous allons proposer tiennent compte de cela et permettent à des musiciens distants d'interagir ensemble, malgré des latences importantes (supérieures à celles du réseau Internet).

1.5 Présentation du document

Dans toute AMID, il est nécessaire qu'au niveau de la communication un système permettant de communiquer avec l'ensemble des autres parties du système soit disponible. Ainsi, dans le chapitre 2, nous proposons un état de l'art des systèmes de transmission de données sur réseaux IP (Internet) en abordant des problématiques de communication au sein d'un groupe (telles que la localisation, le routage et le transport en tant que tels), tout en tenant compte du fonctionnement du réseau et de la présence d'autres types d'applications sur celui-ci.

Nous présenterons ensuite au chapitre 3 un état de l'art sur les systèmes de Performances

[5] pour Institut de Recherche et de Coordination Acoustique-Musique.

[6] Malheureusement cette collaboration s'est arrêté durant la thèse en Février 2005, suite au départ de l'IRCAM de François Déchelle.

Musicales Interactives et Distribuées dans lequel nous essaierons d'identifier les simulacres que de tels systèmes doivent proposer, ainsi que les difficultés que cela soulève dans la conception du système.

Suite à ces études, nous allons développer dans la troisième partie la problématique de la perception des événements suivant un axe temporel. La cohérence y est considérée du point de vue des utilisateurs, qui perçoivent les événements venant du système informatique. Cependant, le thème de la cohérence a été largement étudié pour les systèmes de Mémoires réparties Partagées (MRP). C'est pourquoi nous allons proposer au chapitre 4 un formalisme pour décrire les AMID et les critères de cohérence[7]. Cette approche va nous permettre de comparer les modèles issus des deux catégories d'applications.

Ensuite, nous présenterons et proposerons au chapitre 5 les modèles de cohérence adaptés aux AMID, ainsi qu'une discussion sur la compatibilité mutuelle des critères de cohérence que nous proposons au chapitre 4. Finalement, nous proposerons au chapitre 6 des protocoles de cohérence pour certains modèles, dont l'adaptation du protocole de cohérence perceptive pour deux types d'applications : les jeux distribués et les systèmes de PMID.

Dans la quatrième partie, nous allons présenter les expérimentations que nous avons effectuées dans le domaine des Performances Musicales Interactives et Distribuées. Nous commencerons donc par présenter (au chapitre 7) nJam, le prototype dans lequel nous avons introduit un protocole de cohérence ainsi qu'un métronome partagé.

Nous allons ensuite étudier au chapitre 8 l'utilisation du prototype d'un point de vue utilisateur, mais aussi système multimédia. Pour cela nous allons présenter les démonstrations de musique interactives et distribuées que nous avons menées pour des événements publics. La première fut organisée pour les journées portes ouvertes de l'IRCAM en Octobre 2003 et la deuxième pour la fête de la science au CNAM en Octobre 2005. Ces expériences ont permis de confronter le prototype et les solutions qui y sont intégrées à la réalité des technologies actuellement déployées. Ensuite nous y présenterons une étude sur les usages de notre système, montrant que si la configuration du système est faite selon le tempo et la structure de la musique jouée, alors le système permet à des musiciens de jouer ensemble en conservant un confort de jeu notable.

Nous présenterons ensuite au chapitre 9 une étude de performance du prototype nJam, tel qu'il a fonctionné durant les démonstrations. Ces mesures nous permettent de mettre en valeur les faiblesses de différentes parties du système, comme le matériel utilisé, le système d'exploitation, le réseau, etc.

Nous finirons par conclure dans la cinquième partie. Nous exposerons pour cela un résumé des résultats et des avancés scientifiques obtenus, ainsi que les perspectives de travail dans les domaines des Performances Musicales Interactives et Distribuées et des Applications Multimédia Interactives et Distribuées en général.

[7] Ce formalisme est une adaptation du formalisme de description des MRP proposé par M. Raynal et A. Schiper dans [RS96].

Deuxième partie

État de l'art

Aussi, lorsqu'il frappait une note, la note identique résonnait-elle sur le clavier de ces pianos lointains, dont chaque touche était mue instantanément par le courant voltaïque !

[Jules Verne]
Une ville idéale, 1875

2

La Transmission de Flux Multimédia

Sommaire

Dans les AMID, les participants communiquent entre eux et interagissent à travers l'environnement. Pour cela, les applications des utilisateurs échangent des flux de données comme des flux de messages textuels, des flux audio/vidéo et des flux d'événements discrets. Ces données ont toutes des contraintes temps réel dépendant de l'application et du degré d'interactivité qu'elle fournit aux utilisateurs.

Les solutions standards permettant ce type de communication sur Internet sont récentes et quelquefois complexes à mettre en œuvre. Internet est connu pour sa capacité à mettre en relation un utilisateur avec un service (web, mail). Mais ce n'est que récemment que les messageries instantanées ou les systèmes d'échanges de fichiers ont montré au grand public que l'Internet pouvait être un support de communication temps réel parmi un groupe d'individus formant une communauté.

Bien que les systèmes populaires soient généralement basés sur des solutions applicatives propriétaires, il existe un certain nombre de standards de l'Internet qui permettent de mettre en œuvre des communications Multimédia en temps réel parmi un groupe d'utilisateurs. Pour cela, les mécanismes à mettre en œuvre se retrouvent à différents niveaux du réseau :

- La couche réseau IP effectue la localisation et le routage de proche en proche des paquets. Pour effectuer une communication de groupe, un routage spécifique doit être mis en œuvre car les destinataires des paquets sont multiples
- La couche transport doit tenir compte des caractéristiques temps réel des informations transportées. Elle doit aussi être compatible avec une communication de groupe
- Il faut un système de mise en relation des membres d'un même groupe
- L'application doit tenir compte des caractéristiques du réseau Internet : congestions et pertes potentielles.

Lors de la conception d'une AMID, il est nécessaire de choisir des solutions réseau pour établir une communication entre les utilisateurs. Dans ce chapitre, nous allons introduire les solutions que l'on pourrait placer au niveau de la couche réseau du modèle OSI au paragraphe 2.1. Nous introduisons aussi dans ce paragraphe des solutions issues de la couche application qui fournissent des services équivalents à ceux de la couche réseau : la localisation et le routage. Dans la conclusion de ce paragraphe, nous établirons une comparaison des solutions présentées selon les quatre axes suivants :

- Le système *passe à l'échelle* s'il supporte l'augmentation forte du nombre de flux de données et du nombre d'utilisateurs
- Il permet *l'interactivité* s'il permet l'échange bidirectionnel de données entre utilisateurs, leurs permettant de s'observer mutuellement. De plus, cet échange doit avoir la plus faible latence possible pour qu'ils puissent effectuer des tâches en commun
- Il permet de maintenir plus ou moins difficilement *la cohérence* des données
- Le *coût* de mise en œuvre de la solution.

Ensuite nous présenterons au paragraphe 2.2 des solutions que nous pouvons placer à la couche transport, en développant les problématiques de bout en bout telles que la latence, l'interopérabilité (avec la présentation), la gestion de session et la mise en relation des utilisateurs. Pour finir, nous aborderons au paragraphe 2.3 les problématiques de fiabilité et de congestion, qui doivent être gérées au niveau de l'application, à cause du caractère spécifique de la transmission de données multimédia.

2.1 L'acheminement des données multimédia parmi un groupe interactif

Des solutions existent à la fois au niveau de la couche réseau et de la couche application du modèle OSI. La solution de la couche réseau que nous allons développer, le Multicast IP, est générique et n'est pas construite spécifiquement pour les AMID. D'un autre coté, les solutions que nous pourrions placer au niveau de la couche application reconstruisent de façon plus ou moins complexe les services de routage et d'acheminement, voire des services de plus haut niveau comme la cohérence, la fiabilité, l'archivage ...

Les solutions Multicast se divisent en deux catégories distinctes. Le modèle SSM (Single Source Multicast) fournit une sémantique de 1 vers n. Il est plutôt destiné à la diffusion de contenu puisqu'un seul membre peut émettre des données à destination du groupe. Dans le modèle ASM (Any Source Multicast), le service offre une communication ouverte où chaque membre du groupe peut être à la fois émetteur et récepteur (communication dite de n à m). C'est un modèle adéquat pour les AMID, mais c'est aussi le plus complexe. Bien que cette classification soit issue des différents types de routage Multicast IP, elle constitue une classification sémantique pertinente des solutions présentées dans ce chapitre, puisque par définition la solution SSM est plutôt destinée à des applications non interactives.

2.1.1 Le Multicast IP

Dans cette solution, ce sont les routeurs des réseaux IP qui se coordonnent pour permettre aux membres du réseau de s'abonner à des groupes. Pour cela, le groupe est abstrait par une adresse IP[8]. La notion d'accessibilité globale qui prédomine dans la conception et le déploiement de l'Internet rend ce choix complexe à gérer, puisque à l'initialisation du système, il faut être capable de trouver ou retrouver l'adresse d'un groupe existant (ou d'en utiliser une qui ne l'est pas déjà).

Ainsi, certaines adresses IP Multicast sont réservées, notamment pour fournir une communication au sein d'un groupe d'équipements réseaux qui doivent se coordonner à l'aide d'un protocole prédéterminé. Généralement, ces équipements sont localisés dans des "zones" spécifiques (comme par exemple les zones DNS[9] ou les domaines Multicast). Voici quelques exemples d'adresses réservées :

- 224.0.0.0/8 pour diffusion sur le lien local
  (utilisées par IGMP, DVRMP, RIPv2, PIM, ...)
- 232.0.0.0/8 pour diffusion SSM
- 239.0.0.0/8 pour une portée privée (ex : site local, l'organisation locale)

Lorsque des applications veulent communiquer en Multicast, le choix de l'utilisation d'une adresse peut se faire de façon manuelle (ce qui peut générer des collisions d'adresses), mais aussi de façon dynamique. Session Announcement Protocol (SAP [HPW00]) permet d'annoncer une session Multicast (adresse IP + port) future ou en cours et permet à l'application de choisir

[8] de 224.0.0.0 à 239.255.255.255 en IPv4 et les adresses dérivées du préfixe FF00 : :/8 en IPv6.

[9] La notion de *zone DNS* s'oppose à la notion de *domaine DNS* par le fait qu'elle est administrée et que plusieurs sous domaines DNS peuvent dépendre de cette administration (serveurs mail, DNS, SIP, etc).

une adresse qui n'est pas actuellement annoncée (voir paragraphe 2.2.3). D'autres protocoles tels que GLOP [ML01] et MADCAP [HPS99] fournissent un service d'allocation d'adresses, mais au niveau réseau. Ces protocoles d'allocation d'adresses IPv4 font tous l'objet de RFC (Request For Comment), mais seul MADCAP, bien que souvent critiqué pour sa complexité[10], est au statut "proposed standard".

Durant la phase de transmission des données vers une adresse IP Multicast, les routeurs vont se coordonner pour assurer des fonctions telles que la mise à jour des tables de routage, la construction et la gestion des arbres de diffusion et la gestion des abonnés aux différents groupes[11]. La gestion des abonnés et des sources disponibles sur le lien local ne constitue pas un problème puisque le nombre de machines sur la zone d'administration IP est maîtrisable. C'est au niveau de la construction et de la maintenance de l'arbre de diffusion que les modèles ASM et SSM se différencient : en ASM la multiplicité des sources possibles au sein d'un groupe rend complexe les protocoles de routage (par exemple PIM-SM [EFH+98]) et pose des problèmes de contrôle d'accès et de sécurité. Le modèle SSM (avec par exemple PIM-SSM [Bha03]) laisse l'espoir d'un meilleur déploiement (mais fournit un canal dans lequel une seule source est identifiée).

Le Multicast sur le LAN Ethernet

Bien que certains protocoles permettent de transposer les techniques de routage Multicast au niveau du LAN (IGMP snooping, RGMP, CGMP), ils sont rarement utilisés : les datagrammes Multicast sont donc généralement encapsulés par les commutateurs (les switchs) dans des trames Ethernet ayant pour adresse de destination l'adresse de broadcast MAC. Cela permet à tous les hôtes du réseau de recevoir l'ensemble des trames Multicast et de faire remonter les paquets Multicast à la couche IP, si l'application a informé le pilote de la carte qu'elle est intéressée par le flux correspondant.

Suivant le principe d'Ethernet, cette technique est simple et basée sur l'idée du partage du lien. Cependant, elle provoque l'inondation des liens sur tout le LAN, ce qui peut potentiellement poser des problèmes de débit.

Cependant, ce sont les routeurs qui vont se coordonner pour effectuer l'acheminement des flux entre les différents LAN distants. Chaque routeur doit donc maintenir une liste de flux Multicast à acheminer sur les LANs dont il est la passerelle. C'est à l'aide du protocole IGMPv2 (Internet Group Management Protocol) de l'IETF que les routeurs peuvent interagir avec les hôtes de ces LANs. Ce protocole utilise des adresses Multicast dédiées et permet au routeur d'être informé de l'intérêt que porte les hôtes du LAN à des groupes Multicast.

La figure 2.1 illustre le fonctionnement du protocole IGMPv2 : le routeur demande périodiquement si un hôte du réseau veut s'abonner à un groupe Multicast (il utilise pour cela l'adresse 224.0.0.1). Les hôtes vont ensuite, après un délai aléatoire, répondre par des messages "IGMP report" à l'adresse du groupe Multicast qui les intéresse et si un autre hôte n'a pas déjà répondu pour la même adresse Multicast.

[10] Le fonctionnement de MADCAP s'appuie sur une architecture arborescente dans laquelle les nœuds sont des zones d'administrations IP, comme pour le DNS.

[11] Les protocoles existent aussi pour Ethernet et 802.3, mais quand ils ne sont pas disponibles, les équipements effectuent un broadcast des trames.

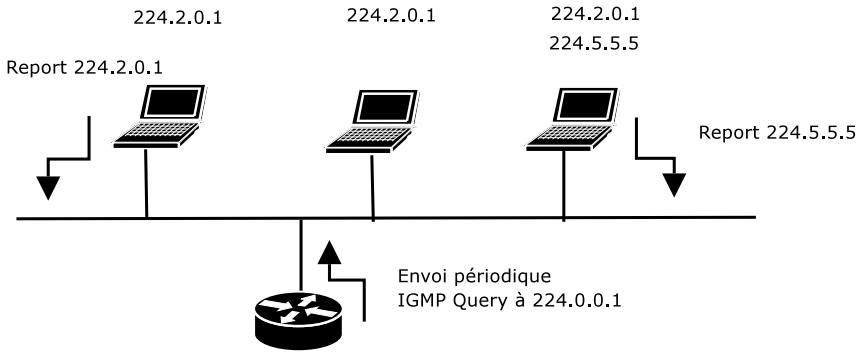


FIG. 2.1 – *Fonctionnement de IGMPv2*

En résumé, pour un hôte, le temps de réponse (le délai entre l'abonnement et la réception des premières données) est fonction de la période d'envoi des messages "IGMP query" du routeur. Pour le LAN Ethernet, l'activation du Multicast nécessite l'inondation des liens, pouvant poser d'éventuels problèmes de débit.

De la sorte, les routeurs sont chargés d'une mission qui fera l'objet du paragraphe suivant : la localisation des sources ainsi que le routage des flux Multicast vers les abonnés.

Les protocoles de routage Multicast

Pour les groupes multi-sources (**Multicast ASM**), le groupe est virtualisé par l'adresse IP Multicast uniquement. Il existe deux modes de construction des arbres : le *mode dense* qui consiste à inonder les liens connectant les routeurs pour élaguer les branches dans lesquelles il n'y a pas d'hôtes abonnés au groupe. Dans ce mode, tous les flux Multicast transitent un moment ou à un autre sur chaque lien. Il est donc stressant car il impose à chaque routeur du domaine Multicast le maintien d'un état pour chaque flux existant. Le *mode épars* consiste à greffer et à élaguer les branches des arbres, ce qui permet de limiter le travail Multicast des routeurs au travail à effectuer sur les flux dont il existe des membres actifs dans les branches logiques inférieures de l'arbre de diffusion.

Dans le déploiement du Multicast, c'est le mode épars et plus particulièrement le protocole PIM-SM (Protocol Independant Multicast, [EFH+98]) qui connaît le plus grand succès, probablement grâce à la maîtrise centralisée que ce protocole fournit. En effet, la construction de l'arbre Multicast se fait par un point de rendez-vous : lorsqu'un routeur a identifié un membre parmi un groupe (un émetteur ou un abonné), il en informe le point de rendez-vous qui commencera par être un relais du flux. Ensuite, grâce aux adresses IP des routeurs sources dans le flux Multicast lui-même, le ou les routeur(s) client(s) pourront se désabonner auprès du point de rendez-vous pour construire une branche ayant un chemin éventuellement plus court et qui ne passe plus par le point de rendez-vous. La racine de l'arbre de diffusion du flux devient alors la source.

Le principal inconvénient du Multicast ASM est qu'il nécessite un couplage important entre les routeurs. C'est pour cette raison que nous parlons de domaine Multicast, où un routeur "point de rendez-vous" (ou un ensemble de routeurs) est désigné pour effectuer le travail de mise

Protocole	ASM	SSM	Mode Dense	Mode épars
DVRMP	X		X	
MOSPF	X		X	
CBT	X			X
PIM-SM	X			X
PIM-DM	X		X	
PIM-SSM		X		

FIG. 2.2 – *Les protocoles de routage Multicast*

en relation des routeurs du domaine Multicast. Cela implique l'utilisation d'autres protocoles qui vont permettre aux différents domaines Multicast de l'Internet de partager les informations concernant les groupes Multicast actifs sur leurs propres domaines. Citons MSDP [FM03] pour découvrir les sources de données des groupes Multicast et MBGP [BCKR98] pour l'échange des informations de routage.

Il existe moins de protocoles pour le **Multicast SSM**, dans lequel il ne peut y avoir qu'une seule source par groupe. En effet, le groupe est virtualisé par une adresse Multicast plus l'adresse IP de la source. Bien qu'il soit plutôt dédié aux applications de diffusion de contenus pré-enregistrés, cette catégorie possède plusieurs propriétés intéressantes :

- Les risques de collisions dans l'adressage des groupes est beaucoup plus faible qu'en Multicast ASM, puisque l'adresse de source identifie un hôte unique sur le réseau
- Le contrôle d'accès à un groupe est possible puisque les hôtes s'abonnent à un flux, et non pas à un groupe comme en Multicast ASM
- Il n'y a pas besoin de protocoles inter-domaines Multicast puisque l'adresse IP de la source dans l'adresse du groupe permet de localiser directement le routeur qui sera la racine de l'arbre, grâce au routage IP classique (Unicast)

Notons que le Multicast SSM est encore en version intermédiaire (*draft*). Le tableau 2.2 présente les principaux protocoles Multicast ainsi que leurs caractéristiques.

Bien que le Multicast IP soit déployé un peu partout autour de la planète (le réseau MBONE[12]), il reste encore marginal et donc disponible à un nombre restreint d'utilisateurs sur l'Internet. C'est probablement pour cette raison que les applications comme les jeux vidéo massivement multijoueurs sont basés sur des architectures de serveurs qui gèrent la notion de groupe au niveau de la couche application.

2.1.2 Les solutions centralisées

Dans la solution centralisée, les membres du groupe communiquent entre eux par l'intermédiaire d'un ou plusieurs serveur(s), qui gèrent la présence des membres, le routage des informations, etc.

L'équivalent sémantique du modèle SSM du Multicast est déjà bien présent dans l'Internet et connaît un engouement commercial avec les caches multimédia [Ram05]. Ces architectures sont constituées de serveurs dédiés gérant des flux multimédia ainsi que leurs politiques d'accès. Elles

[12] `http://www.ietf.org/html.charters/mboned-charter.html`, URL consultée en Juin 2006

rencontrent un succès important auprès des FAIs car elles ramènent les contenus multimédia au plus près des utilisateurs, réduisant ainsi les temps d'accès. Les solutions les plus répandues sont Akamai, MiddleMan, QBIX, Video Caching Network... mais ne sont pas utilisables pour les AMID car elles ne permettent pas d'échanges bidirectionnels entre utilisateurs.

Dans les jeux massivement multijoueurs, l'utilisation de serveurs dédiés pour la gestion du groupe est largement répandue. La distribution des serveurs [Bos05] se fait généralement :

- soit par services (échange de positions d'avatars, chat, audio...)
- soit par réplication de serveur (une réplique à Paris, une à Melbourne...)
- soit par découpage de zones virtuelles (un serveur par région de l'espace virtuel, un serveur par carte ou morceau de carte...)

Ces solutions restent très spécifiques et propriétaires, ouvrant un débat sur les questions de réutilisabilité et d'interopérabilité. Par exemple, l'utilisation d'un service de messagerie instantanée propriétaire ne laissera pas le choix à l'utilisateur sur le logiciel utilisé. De plus, ce type de solutions impose un positionnement stratégique des serveurs sur le réseau, chez les fournisseurs d'accès (maintenance coûteuse). Une architecture complètement distribuée permettrait ainsi aux distributeurs de jeux vidéo de diminuer les coûts de maintenance et d'éviter les problèmes de sécurité liés à la centralisation des informations.

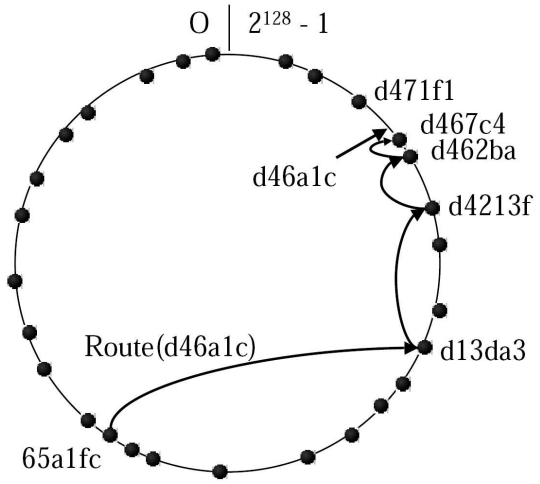
2.1.3 Les solutions Pair à Pair

Les solutions Pair à Pair consistent à construire une communauté virtuelle d'utilisateurs (éventuellement très grande) qui communiquent entre eux directement. Cependant, selon les générations, la localisation des pairs au sein du groupe se fait de façon centralisée ou par serveurs dédiés (deuxième génération[13]) ou de façon complètement distribuée (troisième génération). Les principaux problèmes de recherche que pose ce type d'applications est la dynamicité de la présence des pairs sur le réseau (bien qu'il existe généralement des pairs qui restent présents durant de longues périodes) et le passage à l'échelle d'un grand nombre de pairs.

Les systèmes de troisième génération (comme Chord [DBK+01], CAN [RFH+01] et Pastry [RD01] et Tapestry [ZKJ01]) sont généralement basés sur un système de hachage distribué (DHT) et mêlent la localisation et le routage. Ils fournissent donc un potentiel d'accessibilité important et une couche de routage point à point qui ne permettent pas vraiment la communication d'un groupe d'utilisateurs. En effet, le groupe n'est pas virtualisé par une adresse et les abonnements, le contrôle d'accès, etc ne font pas partie intégrante du routage.

Prenons l'exemple de Pastry : le routage s'effectue en utilisant des *nodeID* (identifiants des nœuds) qui peuvent être calculés en fonction des adresses IP des hôtes qui participent au réseau applicatif (l'overlay network). Chaque nœud va maintenir une table de routage contenant un nombre restreint d'entrées qui associent une adresse IP à un identifiant. Le routage va donc s'effectuer de proche en proche (comme en Multicast IP) et globalement, la répartition des entrées dans les tables de routage doit permettre de joindre tous les hôtes, sans créer de groupes isolés. La figure 2.3 nous montre un exemple de chemin pris par un message pour atteindre son destinataire. Nous pouvons y voir que la construction des tables de routages est faite de sorte que l'acheminement se fasse de façon dichotomique : dans une table de routage, un nœud sait localiser directement beaucoup de nœuds ayant un identifiant de même préfixe et de moins en

[13]Cette classification en générations peut varier en fonction des auteurs.



(A) Le routage d'un message

0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx
65a0x		65a2x	65a3x	65a4x	65a5x	65a6x	65a7x	65a8x	65a9x	65aax	65abx	65acx	65adx	65aex	65afx

(B) La table de routage du nœud 65a1x

FIG. 2.3 – *Le routage de proche en proche dans Pastry*

moins de nœuds ayant des préfixes différents. Cela est illustré par la table de routage du nœud 65a1x, qui sait localiser beaucoup de nœuds ayant un préfixe identique à son propre *nodeId* (les adresses IP associées aux nœuds ne sont pas indiquées dans la table).

Le système de DHT ne suffit pas pour construire plusieurs groupes, puisqu'ils fournissent un système de localisation et de routage global. Cependant, il est possible de construire des mécanismes de gestion d'arbres Multicast au dessus de ces systèmes. C'est ce que font CAN Multicast [RHKS01] (construit au dessus de CAN), SCRIBE [CDKR02] (construit au dessus de Pastry) et de Bayeux [ZZJ+01] (Multicast SSM construit au dessus de Tapestry).

Prenons l'exemple de Scribe, puisque nous avons développé un exemple de routage avec Pastry. S'inspirant probablement de PIM-SM, Scribe fonctionne sur le principe d'un point de rendez-vous, mais celui-ci sera différent pour chaque groupe Multicast. En effet, le point de rendez-vous sera le nœud ayant l'identifiant le plus proche de l'identifiant du groupe (dont la valeur est choisie par le créateur du groupe et dont la syntaxe est équivalente à celle d'un identifiant de nœud).

Pour l'initialisation d'un groupe, le nœud créateur va envoyer un message CREATE dans le système Pastry ayant pour destination l'identifiant du groupe. Ce message arrivera au nœud dont l'identifiant est le plus proche, il se désignera automatiquement comme point de rendez-vous.

Pour s'abonner à un groupe, un nœud va envoyer un message JOIN ayant pour destinataire l'identifiant du groupe (c'est donc le point de rendez-vous qui recevra le JOIN). Tous les nœuds qui vont participer au routage du message devenir des relais pour le groupe désigné (*forwarders*).

Ainsi, pour envoyer un message au groupe, il suffit de l'envoyer à l'aide de Pastry à destination de l'identifiant du groupe. De la sorte, le point de rendez-vous retransmettra le message aux *forwarders* fils, qui feront de même jusqu'aux feuilles.

Cette solution est intéressante car elle permet de faire du contrôle d'accès dans les groupes, grâce au point de rendez-vous "statique". Elle permet aussi d'effectuer une reconfiguration rapide de l'arbre (en relançant un requête JOIN). Cependant, les performances dépendent fortement du point de rendez-vous. Notons de plus que SCRIBE fait communiquer les nœuds pères et nœuds fils avec TCP, ce qui peut éventuellement introduire une gigue importante dans la transmission des messages (voir paragraphe 2.2).

Dans les jeux multijoueurs, la gestion distribuée du groupe passe quelquefois par un autre système que les DHT, et principalement sur les zones d'intérêts [Ben05, HL04, IHK04]. Cela consiste à mettre en relation les utilisateurs proches dans l'environnement virtuel et permet de limiter la quantité de messages échangés globalement (passage à l'échelle d'un grand nombre d'utilisateurs). Ce type de solution rend le système très dynamique car le départ et l'arrivée d'un membre dans un groupe dépend de la position du joueur dans le jeu. Cependant, ces solutions restent intéressantes car elles tiennent compte principalement du besoin de rapidité dans l'échange de message. Précisons qu'il existe d'autres systèmes de Multicast applicatif comme overcast (Multicast fiable avec source unique), Narada (dédié au streaming vidéo).

Par rapport aux architectures centralisées, les systèmes Pair à Pair ont l'avantage d'être complètement ou partiellement distribués, réduisant le coût de déploiement et de maintenance pour les distributeurs d'applications. Aujourd'hui, ces architectures restent peu utilisées dans l'industrie à cause des problèmes difficiles qu'elles soulèvent : manque de mécanisme de sécurité (comme par exemple la triche dans les jeux), cohérence entre les répliques des états du jeu, le modèle économique adapté.

De plus en plus, les architectures Pair à Pair dans les jeux fournissent des sémantiques de communication proche du Multicast IP. En fait, elles se situent à mi-chemin entre les systèmes centralisés et les systèmes de communications de groupe "optimisés" comme le Multicast[14]. Ils permettent en effet la construction d'un arbre Multicast au niveau application, imposant aux clients (qui peuvent quitter le système !) de participer au routage.

2.1.4 Synthèse et conclusion

Le tableau 2.4 présente une synthèse comparative de ces différentes solutions. Pour résumer les trois approches introduites, nous pouvons dire que la tendance actuelle est d'utiliser une architecture centralisée à base de serveur(s), dans le but d'atteindre les utilisateurs sur tout l'Internet (Université, câble, ADSL, etc). D'un autre coté, l'IETF et les chercheurs ont misé sur le Multicast IP, qui fournit un ensemble important de protocoles et de solutions pour la communication de groupe. Mais le Multicast souffre du manque de déploiement vers l'Internet grand public[15] ainsi que du manque d'applications phares pouvant le mettre en valeur. Aujourd'hui ces applications arrivent, mais peut-être trop tard car la communauté scientifique travaille déjà sur les architectures Pair à Pair qui recréent la sémantique du groupe au dessus d'un réseau IP point à point.

[14] En effet, l'envoie d'un message au groupe se fait avec un seul et unique message.
[15] Bien qu'il soit utilisé dans les cœurs de réseau des FAI pour transporter les flux télévisuels.

Solutions	Passage à l'échelle	Interactivité	Gestion de la cohérence	Coût
Multicast IP	Oui mais complexe (point de rendez-vous + MSDP + MBGP)	Délais abonnements / désabonnements longs mais transport "rapide"	Complexe car les applications sont complètement distribuées	Maintenance et configuration dans le cœur de réseau
Serveurs centralisés (sémantique ASM)	Limité par la capacité des serveurs	Le passage systématique par les serveurs augmente la latence	Simple, c'est le serveur qui possède l'état de référence	Important en maintenance
Pair à Pair	Oui	Latence importante (sur-couche de communication) et pertes (dynamicité)	Distribuée et complexe	Nul pour les distributeurs, important en ressources pour les utilisateurs

FIG. 2.4 – *Comparaison des solutions Multicast*

Nous constatons donc qu'il n'existe pas de technologie disponible à l'échelle de l'Internet qui permette d'acheminer des données multimédia à l'échelle d'un groupe interactif. Aujourd'hui, l'approche Pair à Pair manque encore de maturité, mais ouvre des perspectives intéressantes pour les applications futures. Cependant, les problématiques liées au transport de bout en bout de données multimédia ne sont pas à notre connaissance, abordées dans les systèmes Pair à Pair. En effet, la communauté de recherche se focalise aujourd'hui principalement sur la localisation et le routage.

2.2 Le transport des données multimédia

Dans l'acheminement de bout en bout, nous considérons que ce sont les applications qui dialoguent (et non plus des éléments du réseau). Cela suppose l'existence d'une abstraction de désignation équivalente aux ports de l'architecture TCP/IP, ainsi qu'un ensemble de solutions qui vont adapter la transmission en fonction de l'état de charge du réseau. Nous allons voir que la gestion de la transmission est aussi (et surtout) dépendante des types de données transportés et de leurs sémantiques.

Aujourd'hui, malgré l'augmentation importante des flux UDP[16] (télévision, téléphonie, bande annonces, extraits musicaux, etc) sur les réseaux, TCP reste le protocole le plus utilisé dans l'Internet, probablement parce qu'il fournit des services adaptatifs complets (et complexes) comme le contrôle de flux, la fiabilité, la délivrance en séquence et surtout le contrôle de congestion. De plus, TCP reste paramètrable par l'application, notamment dans la quantité de données à envoyer par paquets TCP. Par exemple le protocole Telnet [PR83] configure TCP pour qu'il envoie un seul caractère dans un paquet TCP, dans le but de respecter le caractère "interactif" du contrôle par caractères textuels.

Sachant que les données multimédia ont des caractéristiques de type temps réel (temps de

[16]En fait de l'augmentation de l'utilisation du protocole RTP, qui est un protocole basé sur UDP.

réponse, période d'échantillonnage ainsi des délais de synchronisation), l'application attend du transport qu'il limite (ou qu'il permette de contrôler) les délais et la gigue de bout en bout. La figure 2.5 nous montre les paramètres à prendre en compte dans une transmission isochrone (par exemple la transmission de musique échantillonnée). La période d'échantillonnage p (et donc de décodage) est un délai critique entre le décodage de deux échantillons audio successifs. A partir de cette contrainte, nous pouvons constater que lorsque le premier échantillon est joué (la latence entre le codage et le décodage de cet échantillon est noté t_a pour une transmission de Alice vers Bernard) alors le décodage de l'échantillon suivant possède une échéance, et ainsi de suite. De la sorte, les applications de streaming essayent de maintenir une latence constante en utilisant des buffers pour amortir la gigue du réseau.

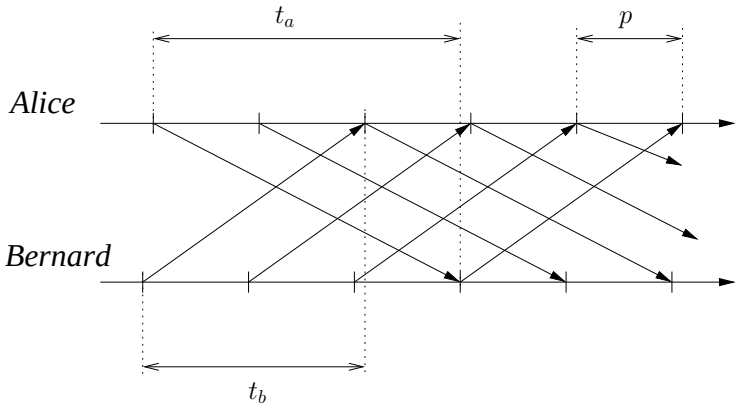


FIG. 2.5 – *Le streaming isochrone*

Ce besoin va avoir des répercussions directes sur le protocole de transport, puisqu'il ne doit pas retarder l'émission d'un message, au risque que le traitement des données qu'il contient ne rate son échéance. Cependant, cette contrainte peut être nuancée, par exemple pour la transmission de la voix. La figure 2.6 propose un exemple dans lequel Alice parle à Bernard : il faut garder une latence constante du codage au décodage uniquement sur le début et la fin d'un mot, mais il est possible de faire varier la latence durant les silences dans les phrases (par exemple entre les mots "Bonjour" et "je").

Le respect des échéances n'est pas la seule contrainte imposée à la transmission par les données multimédia :

- Les données doivent être délivrées en séquence au système de décodage
- Il est nécessaire que les applications puissent identifier le format du contenu afin d'utiliser l'algorithme de décodage approprié
- Il est courant que les applications envoient plusieurs flux multimédia de natures différentes, mais corrélés entre eux. Dans ce cas, il faudra que le transport apporte les informations nécessaires à la synchronisation des flux (comme par exemple une synchronisation des mouvements des lèvres sur un flux vidéo avec le flux audio)
- Les applications multimédia interactives peuvent être amenées à établir un dialogue entre un groupe d'utilisateurs. Comme nous l'avons vu dans le chapitre précédent, la communication vers un groupe se fait par l'envoi d'un message unique avec comme adresse de destination l'identifiant virtuel du groupe. Il faudra donc que l'application puisse identifier l'utilisateur qui est derrière la source d'un paquet particulier.

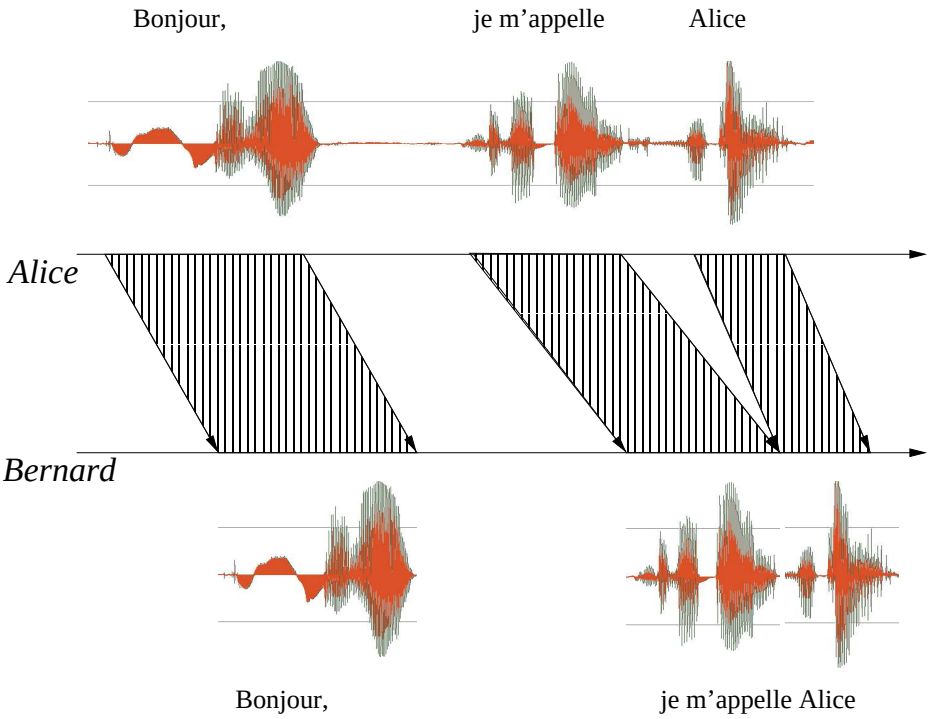


FIG. 2.6 – *Le streaming de la voix*

2.2.1 Que doit-on attendre d'un protocole de transmission de données multimédia ?

TCP n'est pas adapté à la transmission des données temps réel, en effet, le débit d'émission d'une connexion TCP est cadencé par les messages d'acquittement du récepteur ainsi que par la variation de la taille de la fenêtre d'émission. La subtile combinaison de la délivrance en séquence avec la fiabilité totale, le contrôle de flux et le contrôle de congestion fait que lorsque l'émetteur détecte une perte (soit par timeout, soit par réception d'un acquittement répliqué trois fois), il ralentit l'émission des données et reprend l'émission en séquence à partir du paquet perdu. La conséquence de cela est une augmentation de la gigue entre les données envoyées, ainsi que la retransmission de paquets ayant déjà été envoyés (et potentiellement reçus).

D'autre part, il est quelquefois préférable de ne pas retransmettre un paquet comme par exemple pour des flux audio/vidéo pour lesquels les utilisateurs tolèrent une certaine quantité de pertes (généralement d'environ 5%).

UDP n'est pas adapté non plus à la transmission de données multimédia puisqu'il ne fournit ni informations temporelles (séquencement ou date physique) ni informations de présentation du contenu ni d'identification des sources. Nous pouvons cependant le qualifier de rapide par rapport à TCP car il ne met pas en œuvre de mécanisme de fiabilité, de contrôle de congestion, etc.

Il est alors nécessaire que le protocole de transport assure des services de haut niveau que ni UDP ni TCP ne fournissent. Ces services sont les suivants [SCFJ98] :

- Le *séquencement temporel* : les paquets doivent être réordonnés en temps réel par l'application. 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 médias 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 vue d'ajuster les taux de transfert en fonction de la charge du réseau ou de la capacité du récepteur
- La *signalisation* : les informations comme les taux de perte ou le départ et l'arrivée des participants permet à l'application de gérer les communications

RTP/RTCP, les protocoles que nous allons présenter fournissent en fait plus de services que cela puisqu'ils couvrent les couches transport, session et présentation.

2.2.2 RTP, un protocole de transport de données temps réel

- Novembre 1995 : RTP/RTCP devient un standard de l'IETF (RFC 1889).
- Janvier 1996 : Netscape annonce "LiveMedia" (MPEG, H.261, GSM)
- 1996 : l'International Multimedia Teleconferencing Consortium (IMTC) encourage le développement d'une plate-forme téléphonique ouverte, basée sur les spécifications de l'International Telecommunications Union (ITU) et de l'IETF
- Avril 1998 : publication de RTSP, une couche de contrôle des flux multimédia pour serveurs de vidéos pré-enregistrées (fonctions retour rapide, pause, ...)
- Avril 1998 : publication de SDP (RFC 2327)
- 1999 : publication SIP (RFC 2543), un protocole de signalisation pour créer, modifier et terminer les sessions multimédia (téléphonie, conférence, etc)
- Octobre 2000 : Session Announcement Protocol (SAP, RFC 2974)
- 2003 : mise à jour de RTP (RFC 3550) et du profile audio/vidéo (RFC 3551)
- 1996-2006 : Publications régulières de RFCs "RTP payload format ..."
- 2004 : version **RTP sécurisé** (RFC 3711)

FIG. 2.7 – *L'historique de normalisation de RTP à l'IETF*

RTP (Real time Transport Protocol) est devenu un standard de l'IETF (the Internet Engi-

neering Task Force) en 1995 (voir l'historique, figure 2.7). Rapidement, Netscape et l'IMTC ont montré un intérêt pour ce protocole. La normalisation de RTP a été rapidement suivie par la proposition de protocoles permettant d'enrichir l'utilisation de RTP, en vue d'applications spécifiques : RTSP pour le contrôle de transmission de flux multimédia dans la diffusion de contenus, SIP pour la signalisation d'appels téléphoniques, SAP pour l'annonce de sessions multimédia. Notons que SDP (Session Description Protocol) est utilisé par RTSP, SIP et SAP pour décrire les informations nécessaires à l'établissement de sessions multimédia (adresse(s), port(s), outil(s), politique de sécurité, etc).

Contrairement à ASN1, CORBA et XML, la présentation des formats dans RTP se fait de façon statique, dans les RFCs (Request For Comments) accessibles librement sur Internet. Depuis 1996, nous pouvons recenser plus de 35 RFC décrivant des formats de présentation de médias divers et variés. Les premiers médias à avoir été "encapsulables" dans RTP sont les formats issus de la téléphonie (décrit dans le standard "RTP Profile for Audio and Video Conferences with Minimal Control"). Progressivement, les formats comme MPEG, DV, AC-3 ont été intégrés dans les médias compatibles avec RTP, montrant que RTP est un protocole qui trouve sa place chez les constructeurs. En effet, pour effectuer du streaming la plupart d'entre eux ont remplacé leurs protocoles propriétaires par RTP (Realplayer, Quicktime, Windows Media).

En plus de pouvoir transporter des médias, RTP peut transporter d'autres types de données, comme par exemple des données audio redondantes pour la compensation d'erreurs ou des codes correcteurs d'erreurs (nous reviendrons sur la compensation des erreurs de transmission de données multimédia au paragraphe 2.3.2). Progressivement, les médias dit discrets (voir la deuxième partie) comme la position d'un pointeur de souris, l'arrêt de transmission de données pendant un blanc dans une conversation (comme dans la figure 2.6) ou alors des données textuelles (pour les sous-titrages par exemple) sont devenus des médias transportables par RTP.

Bien que RTP puisse être utilisé au dessus de protocoles de transport tels que TCP, il est conseillé dans sa spécification de l'utiliser au dessus d'un protocole équivalent à UDP, qui est compatible avec le Multicast. La gestion des erreurs de transmission, le contrôle de flux ainsi que le contrôle de congestion ne sont assurés ni par RTP ni par RTCP (le protocole de contrôle associé à RTP qui est défini dans la même RFC). Si de tels mécanismes doivent être mis en œuvre, ce sera l'application qui devra s'en occuper, éventuellement à l'aide des informations statistiques fournies par RTCP.

Les paquets RTP

La deuxième version de RTP (champ V de la figure 2.8) est définie dans la RFC 3550. Comme dans sa précédente version, il fut publié conjointement avec un profil pour les conférences audio et vidéo. Dans ce profil, on trouve notamment les correspondances entre les codages traditionnels de la téléphonie comme GSM et les G7xx de l'ITU-T (l'International Telecommunication Union) ainsi que ceux de la vidéo-conférence comme CelB de SUN ou les h26x de l'ITU-T. Bien qu'il soit possible de définir d'autres profils, depuis la normalisation de RTP, seulement deux ont été définis : celui pour les conférences audio et vidéo et celui sur les communications sécurisées (SRTP). C'est dans ces profils que l'interprétation d'une valeur à 1 pour le champ M est définie. Il en est de même pour le champ X, qui précise si l'en-tête RTP possède une extension.

La spécification de RTP permet de tenir compte d'architectures complexes introduisant ce

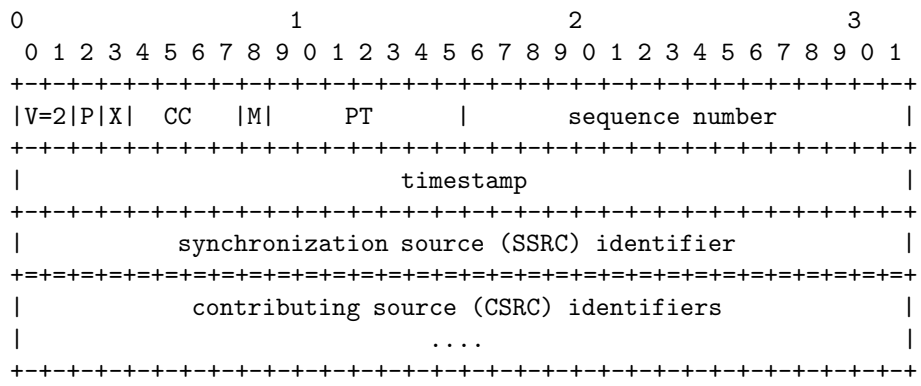


FIG. 2.8 – *L'en-tête RTP*

que l'on appelle des *mixer* et des *translator*. Un *mixer* est un système qui doit être capable de recevoir un certain nombre de flux RTP et de les agréger dans un même flux par mixage (addition des échantillons en audio, création de mosaïques pour des flux vidéo, etc.). Un *translator* va effectuer une conversion de format de codage du média.

Dans ce type d'architecture, les *mixer* et les *translator* doivent travailler sur les médias eux mêmes, donc travailler au niveau de l'application. Ils deviennent donc des nouvelles sources RTP dans le canal de communication, mais ne sont que des "relais". Si le champ CC (CSRC count) du paquet est égal ou supérieur à 1, alors la source du paquet a effectué une translation de codage ou un mixage d'un nombre de flux égal à CC. Dans ce cas, les champs $CSRC$ (contributing sources) permettra de spécifier les identifiants RTP des sources étant à l'origine des flux. Par contre, le champ $SSRC$ (synchronization source) identifie toujours l'émetteur du paquet RTP. Tous ces identifiants doivent être calculés par RTP, à l'initialisation, en essayant de ne pas générer de collisions.

Le champ PT (Payload Type) permet de spécifier le format du codage de la zone de données. Le champ *timestamp* va fournir une date spécifiée par l'application qui pourra être issue soit de l'horloge du système d'exploitation, soit de l'horloge d'échantillonnage du système d'acquisition. Cette date correspond à celle associée à la première unité de média qui est comprise dans le paquet et ne doit pas spécialement être synchronisée avec l'horloge universelle. Finalement, le champ *sequence number* correspond à une numérotation logique des paquets successifs du flux RTP, dans le but de permettre à l'application de reconstruire le séquencement.

Maintenant que nous avons détaillé les différents champs de RTP, nous sommes en mesure de constater que RTP fournit des informations qui font de lui :

- Un protocole de session puisqu'il permet d'identifier logiquement les sources de média. Comme cela n'est pas fait avec l'utilisation de l'adresse réseau de la source, cela permet à un hôte d'héberger plusieurs sources et de changer d'adresse en cours de communication (mobilité de l'équipement)
- Un protocole de présentation puisqu'il annonce le format du contenu
- Un protocole temps réel puisqu'il transporte les dates physiques associées au média transporté.

Cependant, il ne permet pas en l'état aux applications de contrôler l'état de la communication

(délais, gigue, taux de pertes) ainsi que la gestion de la session (départ d'un membre du groupe, informations spécifiques à un membre). Ces fonctionnalités sont fournies par le protocole associé à RTP et qui est défini dans la même RFC : RTCP (RTP Control Protocol). Remarquons cependant le manque de qualité de service (QoS) peut éventuellement mettre en échec une transmission de données ayant des contraintes temps réelles. En effet, si la bande passante disponible est inférieure au débit nécessaire au média, alors la signalisation fournie par RTCP ne pourra être utilisée et la transmission sera de très mauvaise qualité.

Les paquets RTCP

Pour que les applications puissent contrôler la session, chacun des membres de la session va envoyer périodiquement des paquets RTCP. Il existe cinq types de paquets :

- *RR* receiver report : il sera utilisé dans le but d'avoir un retour sur la qualité de réception d'un flux issu d'une source donnée.
- *SR* sender report : indispensable dans le calcul de délai aller retour, il informe les récepteurs de la position courante des estampilles logiques et temporelles du flux émis
- *SDES* Source description items : fournit le nom et le domaine DNS (Domain Name System) de l'utilisateur (champ CNAME) ainsi qu'éventuellement son nom usuel, son mail, etc
- *BYE* : message de fin de participation : permet de maintenir un état sur le nombre de membres actifs du groupe
- *APP* : message qui est laissé à l'application en cas de besoins spécifiques. Il peut être utilisé par exemple pour des synchronisations spécifiques (voir troisième partie) ou pour effectuer un contrôle de congestion (voir paragraphe 2.3.1).

Bien qu'il ne soit pas indispensable à la transmission, il est fortement conseillé d'utiliser RTCP, notamment pour obtenir le CNAME des membres du groupe, puisque celui-ci est la référence permettant d'associer plusieurs flux RTP issus de la même source, et éventuellement à synchroniser. De plus l'utilisation du CNAME permet de supporter les déconnexions, puisque le CNAME n'est pas sensé changer au cours du temps.

RTCP ne partageant pas le même numéro de port que RTP, il faut être capable de contrôler son débit et éviter que celui-ci ne crée des effets de bord trop importants. Pour cela, les paquets RTCP sont envoyés périodiquement et leur débit total ne doit pas dépasser 5% du débit total de la session RTP. Le partage de cette bande passante doit être d'environ de un quart pour les émetteurs et du reste pour les récepteurs. Ainsi, le processus RTCP d'un membre va effectuer un certain nombre de calculs statistiques sur le débit total de la session, le nombre de sources, le nombre de récepteurs, etc dans le but de calculer sa propre période d'émission de paquet RTCP.

Un paquet RTCP est composé de plusieurs zones : un en-tête commun qui permet de spécifier (champ *PT*) si le paquet est un sender report ou un receiver report et l'identifiant RTP de l'émetteur. Ensuite, le paquet est découpé en différentes zones. Dans le sender report (figure 2.9) il y a une zone pour les informations du flux émis ainsi que les informations sur les flux reçus (l'exemple montre une session interactive). Dans la zone *sender info* on retrouve la date d'émission du paquet (*NTPtimestamp*) ainsi que la position courante du flux RTP associé au SSRC.

Du coté récepteur (figure 2.10), les informations envoyées sont relatives à la gigue, les pertes perçues. Les champs *LSR* et *DLSR* sont utiles au calcul du délai aller retour (figure 2.11). Celui-ci va être calculé par l'émetteur grâce au sender report et au receiver report. A l'instant

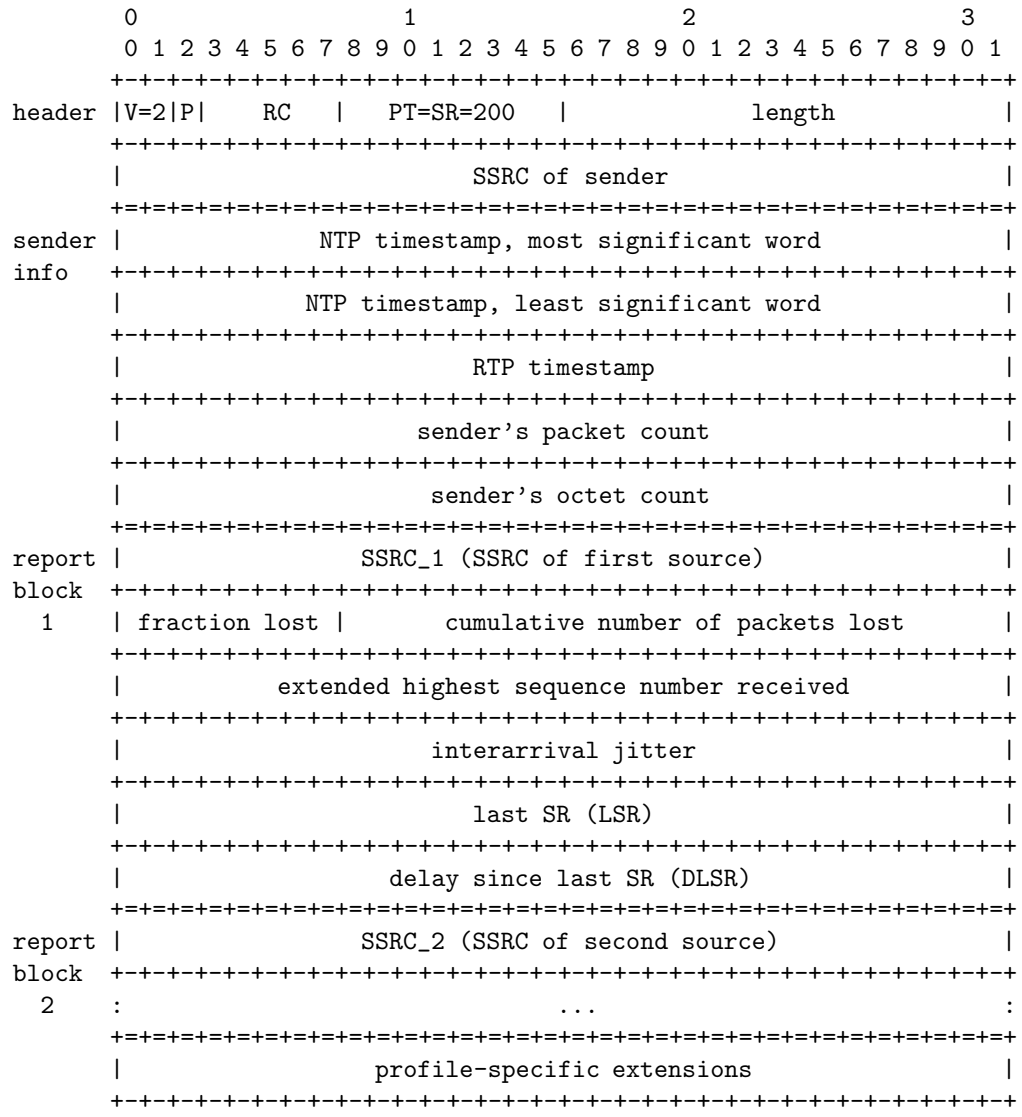


FIG. 2.9 – *Le format d'un sender report RTCP*

t_c, l'émetteur va envoyer un sender report, dans lequel le champ $NTPtimestamp$ va être mis à la valeur t_c. Sur réception d'un tel rapport, le récepteur RTP va déclencher un chronomètre pour calculer le délai entre la réception du rapport d'émission et l'envoi de son prochain rapport de réception (valeur $DLSR$, Delay since Last Sender Report). De plus, il va recopier t_c dans le champ LSR. L'émetteur va donc calculer le délai aller retour comme suit :

$$RTT = t_{reception} - LSR - DLSR$$

Cette façon de procéder a deux avantages majeurs. Tout d'abord l'émetteur ne doit pas se souvenir de toutes les dates d'émission de ses rapports, puisque les récepteurs lui répètent dans leurs rapports. De plus, le calcul se fait en comparant des différences temporelles qui restent locales, évitant d'avoir besoin d'horloges synchronisées. Ces avantages permettent d'améliorer le passage à l'échelle d'un grand nombre de membres dans un groupe.

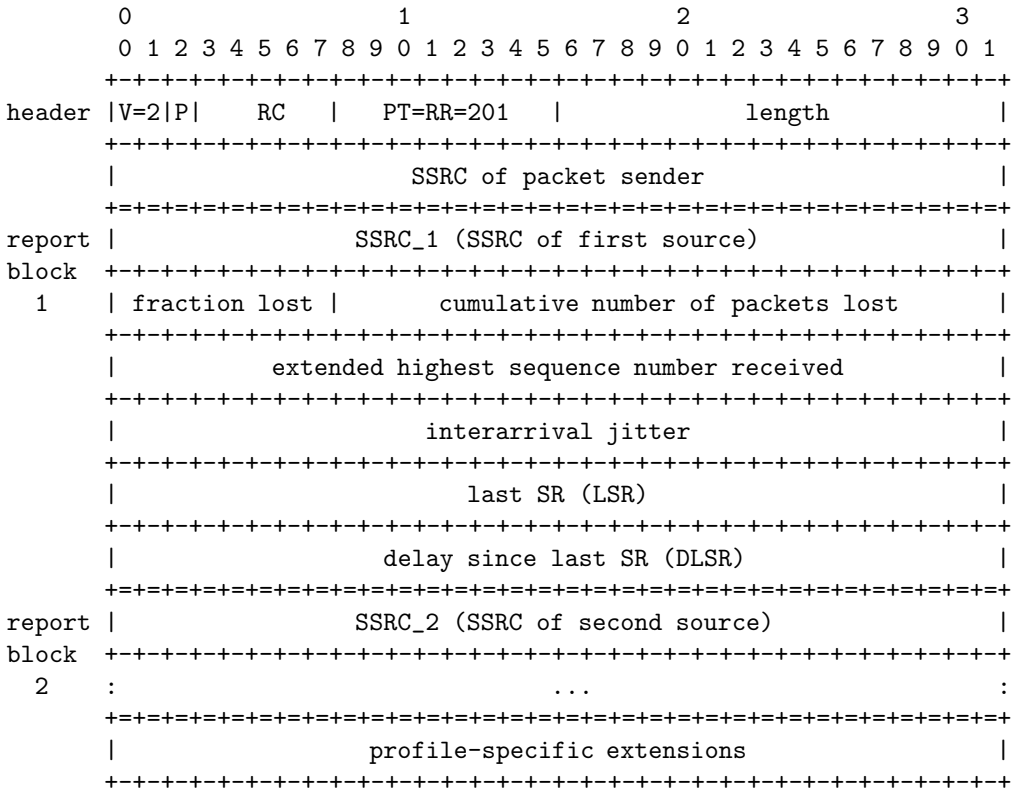


FIG. 2.10 – *Le format d'un receiver report dans RTCP*

Conclusion sur RTP/RTCP

Pour résumer le comportement et les possibilités offertes par RTP, nous allons dérouler un scénario type, décrit dans sa RFC :

Lors d'une conférence audio, la définition des formats des paquets (RFCs du type RTP payload format ...) fait qu'un paquet RTP ne contient pas plus de 20ms de voix (dans le but de limiter la latence). Le codage est alors détecté par les récepteurs en analysant un champ de l'en-tête. Si un émetteur veut changer de codage, par exemple pour diminuer le débit d'émission, il lui suffit de mettre à jour le bon champ pour en informer les récepteurs. Malgré tout, les informations temporelles contenues dans l'en-tête permettent de maintenir la continuité du flux audio.

Si l'on veut ajouter des flux vidéo dans cette conférence, alors les flux vidéo vont être envoyés dans une ou plusieurs autres sessions RTP différentes. Grâce aux informations sur les utilisateurs contenues dans RTCP, l'application pourra associer les flux vidéo et audio issus du même membre, permettant d'effectuer une synchronisation si nécessaire.

Bien que RTP/RTCP ne fournissent pas directement de solutions pour les problèmes classiques comme le contrôle de flux ; le contrôle de congestion et la fiabilité, il est une avancée majeure dans la transmission de données temps réel, tout en tenant compte du passage à l'échelle, du Multicast, de la gestion de session, l'interopérabilité, la mobilité et des déconnexions.

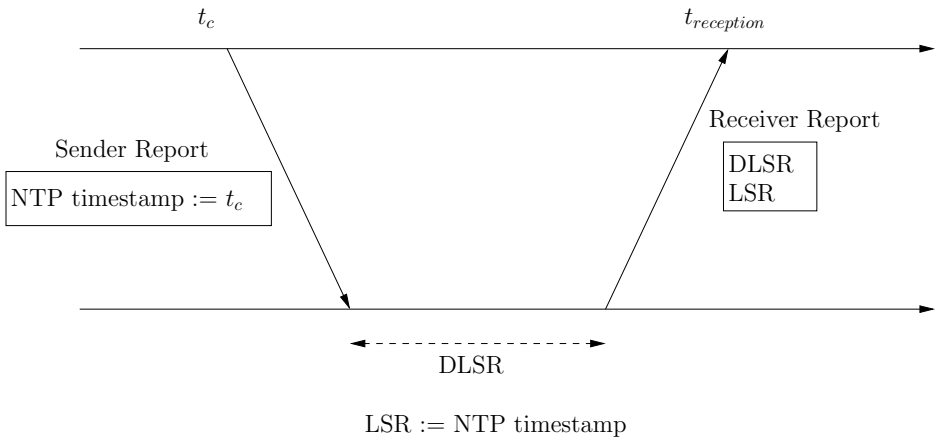


FIG. 2.11 – *Le calcul du délai aller retour (RTT) avec RTCP*

Cependant, pour qu'une session RTP/RTCP puisse avoir lieu, il est nécessaire qu'une procédure d'initialisation mette en relation les participants. En UDP et en TCP, la mise en relation (pour une communication point à point) se fait généralement avec des URL (Uniform Ressource Locators) comme par exemple `http://www.cnam.fr` qui définit à la fois un protocole et une machine. La résolution de l'adresse de la machine se fait grâce au système DNS (qui possède quelque part une entrée qui associe `www.cnam.fr` à l'adresse IP 163.173.128.121). Pour savoir quel port doit être utilisé pour le dialogue, les protocoles comme http sont associés à des "ports bien connus".

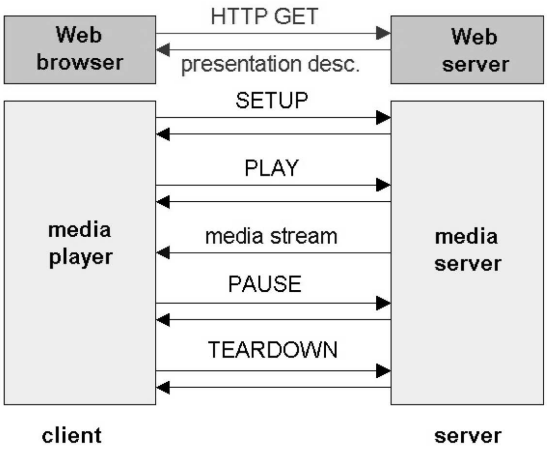
Dans le cas de RTP/RTCP, il faut être capable d'associer un utilisateur ou un groupe à une adresse IP de destination, ce qui est impossible en l'état actuel puisque les DNS ne référencent ni les utilisateurs ni les groupes. C'est pour cette raison que des protocoles ont été définis dans le but de mettre en relation des utilisateurs (les conférences audio/vidéo publiques ou privées, la téléphonie sur IP, etc.).

2.2.3 La mise en relation des utilisateurs

Dans toutes les applications multimédia interactives, la question de mise en relation des utilisateurs se pose. Différentes solutions ont été adoptées. Les solutions normalisées sont celles liées à la téléphonie sur IP (SIP), les conférences audio/vidéo (SAP) et au streaming audio/vidéo dans des pages web (RTSP). Nous allons détailler un peu plus ces solutions dans ce paragraphe.

Notons que chacun de ces protocoles utilise le protocole SDP (Session Description Protocol) pour décrire les paramètres nécessaires à l'établissement de la session. Ces paramètres sont les formats utilisés, le profile RTP, les adresses IP et les numéros de port, etc. Dans la figure 2.12, les informations issues de l'encapsulation de SDP seront surlignées en gris. Dans la figure 2.13, les informations issues de SDP sont affichées dans le "contenu de la session". Remarquons que SDP n'est pas limité aux protocoles que nous présentons.

Cependant, toutes les solutions ne sont pas normalisées : la mise en relation d'utilisateurs est aussi nécessaire dans d'autres types d'applications, pas exemple les jeux vidéo multijoueurs ou dans les messageries instantanées (MSN messenger, yahoo messenger, Skype, etc). Une so-



(A) L'échange de message

```
C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
   Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK
   Session 4231

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231
   Range: npt=0-

C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231
   Range: npt=37

C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231

S: 200 3 OK
```

(B) Le contenu des messages

FIG. 2.12 – Les interactions client serveur dans RTSP

lution intéressante que l'on peut retrouver dans les jeux multijoueurs (mais pas massivement multijoueurs) est la solution "salle d'attente". Le principe est d'ouvrir un espace dans lequel les joueurs vont "s'inscrire" à une partie et vont attendre qu'ils soient suffisamment nombreux pour commencer. Cette solution permet, du point de vue de l'application, d'effectuer un filtrage sur les joueurs en fonction de leurs caractéristiques réseau. Par exemple, on peut ouvrir une partie dont les joueurs sont à "moins de 150ms de ping" du serveur.

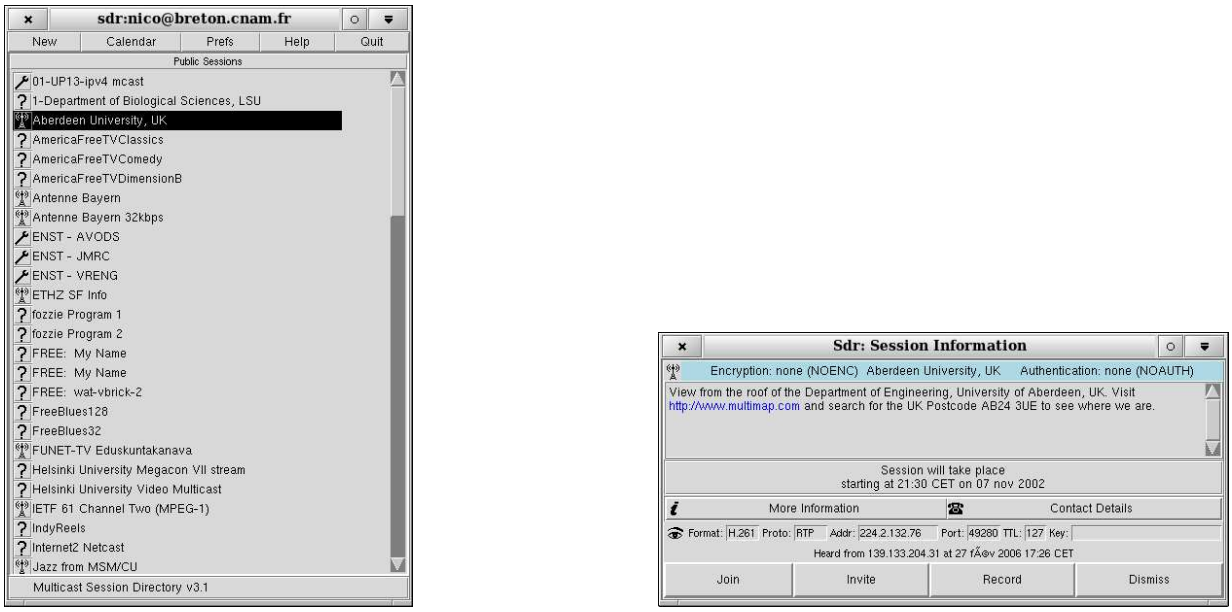
L'approche utilisée dans les systèmes comme Skype, MSN messenger, etc. est-elle aussi intéressante car il n'y a pas vraiment de mécanisme de découverte des utilisateurs, mais une liste de contact. La connaissance de nouveaux utilisateurs se fait par "réseautage", introduisant une forme "d'Internet social" qui connaît un succès important. Ces systèmes risquent de devenir les points de départ d'une grande partie des applications coopératives sur Internet, puisqu'ils peuvent initier d'autres applications coopératives (diaporama, jeux, présentations, etc).

RTSP : Real time Streaming Protocol

Comme le montre l'historique de RTP (figure 2.7), le premier protocole associé à RTP à avoir été normalisé par l'IETF fût RTSP (Real Time Streaming Protocol). Cela est probablement dû au besoin de plus en plus pressant d'intégrer des flux audio et vidéo dans les navigateurs web. RTSP sert donc au contrôle de lecture d'un flux qui est pré-enregistré (ou différé) comme une bande annonce pour un film. Ici, l'utilisateur ne va pas s'adresser à un groupe, mais à une machine unique qui va lui envoyer le flux du média. L'utilisation d'URL est alors possible (voir figure 2.12). Pour cela, le client s'adresse au serveur de contenus, qui va devoir maintenir un état par client (numéro de session dans la figure) pour que celui-ci puisse contrôler le lecture avec des fonctions telles que pause, retour rapide, etc.

SAP : Session Announcement Protocol

Le protocole SAP permet d'annoncer des sessions futures ou en cours à la manière d'un programme télévision. Pour cela, les annonces sont diffusées à l'aide d'adresses Multicast dédiées :



(A) Les session annoncées

(B) Le contenu d'un session

FIG. 2.13 – La consultation de l'annuaire SAP

224.2.127.124 pour une portée globale et 239.16.33.255 pour la zone DNS locale en SAPv2.

Avec un logiciel comme SDR (figure 2.13), un utilisateur peut consulter l'ensemble des sessions annoncées et consulter leurs contenus (données SDP). Ainsi il peut lancer de façon automatique (ou en recopiant les informations) une application qui sera alors membre de la session.

SIP : Session Initiation Protocol

SIP est un protocole de signalisation pour la téléphonie sur IP. Son fonctionnement s'appuie sur l'architecture DNS. Il nécessite la présence de serveurs dédiés dans les zones DNS. Ainsi, un utilisateur qui désire téléphoner à un autre va envoyer une requête à un serveur dont il a la connaissance (figure 2.14), comme cela se fait pour la résolution nom logique/adresse IP avec le DNS. Ce serveur va devoir localiser le destinataire, à l'aide de l'adresse SIP, qui est de la même forme qu'une adresse mail. Pour cela, le serveur dédié va demander au DNS (à la manière de SMTP, le service de messagerie) l'adresse IP du serveur SIP faisant autorité pour l'utilisateur. Finalement, après avoir localisé l'utilisateur, le serveur SIP de l'appelant va effectuer une signalisation propre à la téléphonie (sonnerie, fin d'appel, etc). Ensuite, la communication se fera à l'aide de RTP, puisque la phase de signalisation aura permis aux correspondants d'échanger les paramètres nécessaires.

Actuellement, SIP est moins utilisé que son équivalent normalisé par l'ITU-T : h323. Cependant il semblerait que la tendance soit de privilégier SIP car il est plus simple et que son architecture est modulaire (grâce à l'utilisation du DNS notamment).

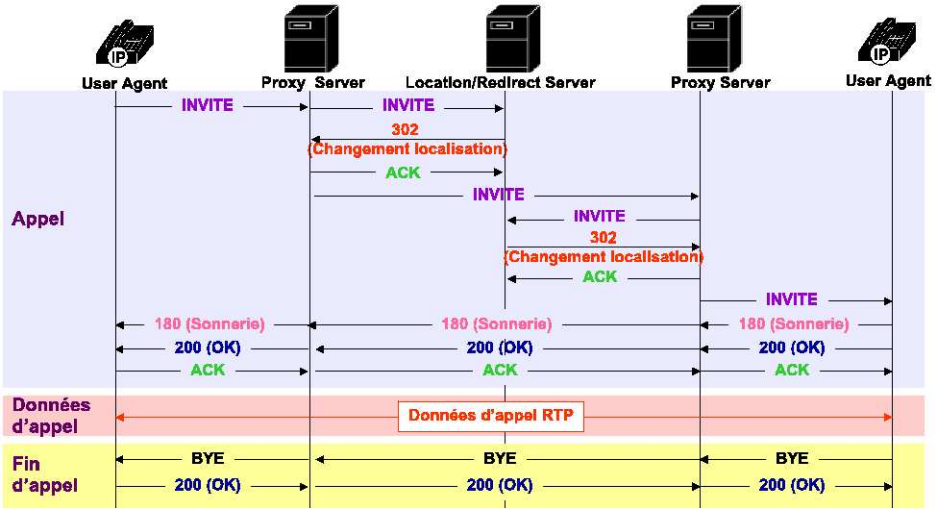


FIG. 2.14 – *Les interactions SIP*

2.2.4 Conclusion sur le transport

La solution de transport RTP est en fait une solution couvrant les couches de transport, de présentation et de session. Ce protocole permet de transporter de façon interopérable un nombre important de format de données multimédia, grâce à la description de leurs encapsulations dans les normes appelées "RTP payload for ...".

Contrairement à TCP, le protocole RTP n'introduit pas de latence et de gigue dans la transmission. De plus, son complément, le protocole de contrôle RTCP permet à l'application d'effectuer une enquête statistique sur la transmission. De plus RTP utilise un dialogue sources/récepteurs suffisamment découplé pour lui permettre d'être compatible avec n'importe quel type de communication Multicast. Cependant, à notre connaissance, il n'a été mis en œuvre qu'au dessus du Multicast IP.

Avec la normalisation de RTP sont arrivées des propositions de standard pour effectuer la mise en relation entre entités communicantes via un flux multimédia. Les technologies utilisées pour effectuer la mise en relation sont multiples : RTSP pour le web, SAP pour le Multicast et SIP pour le nommage logique des utilisateurs (équivalent aux adresses mail). Les solutions propriétaires, basées sur des annuaires centralisés sont souvent utilisées pour mettre en relation des utilisateurs de jeux multijoueurs, de messageries instantanées, etc.

La suite de protocoles que nous avons présentée fournit un ensemble de services nécessaire à la conception des AMID. Cependant, les contraintes temps réel des flux multimédia ainsi que la communication Multicast font que les mécanismes de contrôle de flux, de contrôle de congestion et de gestion des erreurs de transmission ne sont pas proposés par RTP/RTCP. En effet, ces mécanismes se heurtent à la spécificité de chacun des médias transportés : débit constant ou pas, taux de pertes toléré, variation de qualité acceptée par l'utilisateur, etc. Ainsi, ces mécanismes sont laissés à la couche application, qui va dans certains cas devoir être capable de s'adapter à l'état du réseau à l'aide d'une analyse statistique (avec RTP/RTCP) de la qualité de transmission

durant l'exécution.

2.3 Les mécanismes supportés par les applications

Nous allons présenter ici quelques solutions proposées dans la littérature pour adapter la transmission à l'état du réseau. A notre connaissance, les seules solutions pour le contrôle de flux consistent à demander à l'utilisateur, avant la transmission, la capacité en débit de sa connexion au réseau. Par exemple, pour visualiser une bande annonce de film sur un site web, l'utilisateur doit généralement choisir entre plusieurs flux ayant des débits plus ou moins importants. Nous ne développerons donc pas plus la problématique du contrôle de flux.

Cependant, comme nous l'avons vu avec les solutions mises en œuvre dans le protocole TCP, nous savons que les solutions classiques de contrôle de congestion et de gestion des erreurs ont tendance à ralentir la cadence d'émission des données, introduisant une gigue importante. Nous présentons ici des solutions issues de la littérature.

2.3.1 Le contrôle de congestion

La congestion est un phénomène qui apparaît dans un réseau lorsque l'ensemble des sollicitations des hôtes provoque un travail supérieur à la capacité d'au moins un des éléments d'acheminement intermédiaire (un routeur ou un commutateur LAN). C'est pour cette raison que TCP possède un contrôle de congestion [APS99] qui permet d'adapter le débit de la connexion à la bande passante disponible du réseau. L'algorithme utilisé est dit AIMD (Additive Increase, Multiplicative Decrease) et va faire varier la taille de la fenêtre d'émission[17] pour faire varier le débit. En effet, TCP utilise cette fenêtre comme "tampon" d'envoi des messages. Après l'envoi du contenu de cette fenêtre, il la remplit de nouveau et attend l'acquittement du récepteur pour envoyer la suivante. Ainsi, faire varier la taille de la fenêtre au cours du temps permet de contrôler le débit.

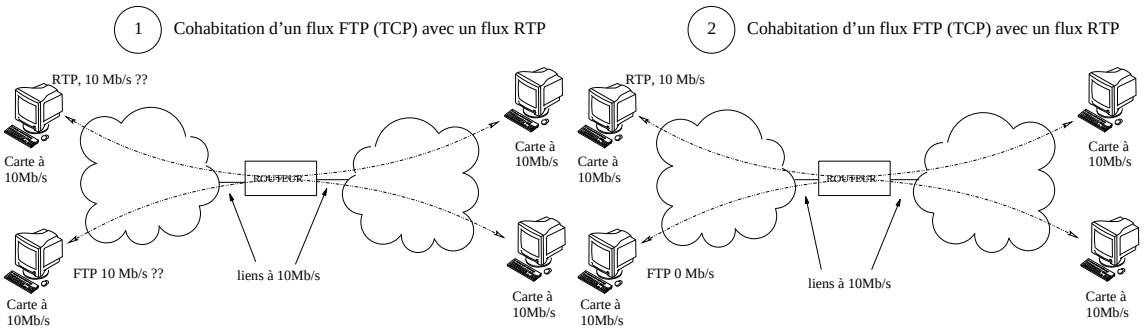
Dans le contrôle de congestion, le calcul de la taille de la fenêtre (noté W) se fait comme suit :

$$W \rightarrow \begin{cases} W + \alpha & si\ pas\ de\ perte \\ max(\lfloor \beta W \rfloor, 1) & sinon \end{cases}$$

où $\alpha = 1$ (croissance additive) et où $\beta = 1/2$ (décroissance multiplicative). Nous pouvons alors constater que le comportement de TCP est "statique" (le contrôle de la fenêtre n'est pas effectué en fonction d'une mesure fine de l'état du réseau). Cependant, ce principe à l'avantage de fonctionner et d'avoir porté TCP sur la plus haute marche du podium des protocoles de transport les plus utilisés de l'Internet.

Sur l'ensemble du réseau, le contrôle de congestion de TCP permet de partager équitablement la bande passante disponible, à condition que tous les hôtes du réseau jouent le jeu. Prenons l'exemple (figure 2.15) dans lequel une application utilisant RTP sans contrôle de congestion partage un lien avec un transfert de fichier FTP (basé sur TCP). Au départ de la transmission, le flux RTP va envoyer à un débit de 10Mb/s alors que TCP va progressivement "tester" la bande passante disponible. Il va alors s'adapter et tomber à 0Mb/s, puisque RTP occupe aveuglément

[17] En réalité, la taille de la fenêtre d'émission à un moment donné est calculée en fonction du contrôle de congestion et du contrôle de flux. Nous ne nous intéressons ici qu'au contrôle de congestion.



(A) Les capacités potentielles de transmission des deux flux

(B) Le partage après adaptation de TCP

FIG. 2.15 – *Le partage de la bande passante entre TCP et RTP*

toute la bande passante.

Au niveau de la couche réseau, il existe des solutions permettant d'éviter ou de limiter la congestion. Par exemple RSVP [ZDE93] et IntServ [RLS99] sont des solutions de réservation de ressources réseau qui annulent la notion de partage des liens, évitant ainsi complètement la congestion. Une autre approche, celle de DiffServ [RLS99] consiste à marquer les flux pour permettre aux routeurs de grouper le traitement des paquets dans des files d'attente différentes en fonction de la nature des flux (notion de classe de services). Cela crée des groupes distincts dans lesquels une congestion peut apparaître. Ces solutions demandent la participation de tous les routeurs du chemin emprunté, ce qui n'est pas possible à l'échelle d'Internet. En effet, il existe encore beaucoup d'équipements non compatibles et la tarification de telles solutions reste compliquée. De plus, les garanties fournies au niveau IP n'ont pas toujours leurs équivalents au niveau liaison, comme dans les commutateurs Ethernet dans lesquels la congestion peut aussi apparaître.

Le problème de congestion avec UDP/RTP peut aussi être géré au niveau de la couche transport, en ajoutant un contrôle de congestion au niveau de l'application. La conception d'un tel contrôle de congestion doit alors fournir les propriétés suivantes :

- Réagir vite, i.e. ralentir la cadence d'émission en cas de détection d'une congestion. Pour une transmission de données temps réel (possédant une période), il faudra dégrader la qualité des données (fréquence d'échantillonnage ou quantification) pour garder la cadence du média
- Être équitable avec les autres flux, qu'ils soient transportés avec TCP ou avec un protocole utilisant le même contrôle de congestion [Bou02]
- Éviter les grandes variations de débit et donc de qualité des données, pour le confort de l'utilisateur.

Le contrôle de congestion unicast

Il existe beaucoup de propositions de contrôle de congestion pour RTP qui sont issues de la recherche. Pour une transmission Unicast de bout en bout, il existe deux types d'algorithmes :

les algorithmes réactifs et les algorithmes proactifs (voir [Bou02] pour une comparaison et une description plus détaillée de ces algorithmes). La principale différence entre les deux catégories est la méthode de détection du phénomène de congestion.

Majoritairement, les algorithmes réactifs copient le comportement du contrôle de congestion AIMD de TCP et considèrent qu'une perte est le signe d'une congestion. Ensuite, ils *réagissent* à la congestion. Dans cette catégorie, ce sont les algorithmes binomiaux [BB00] qui semblent être les plus efficaces en terme de stabilité et d'équité à TCP. L'augmentation et la diminution de la taille de la fenêtre de congestion sont fonction de la fenêtre courante, réduisant ainsi la variabilité du débit sortant. Le choix de diminution se fait sur détection d'une perte (comme TCP). Voici un résumé de l'algorithme :

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

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

Où w_t est la taille de la fenêtre 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. Les auteurs de [BB00] montrent qu'un algorithme binomial cohabite correctement avec TCP si $k + l = 1$.

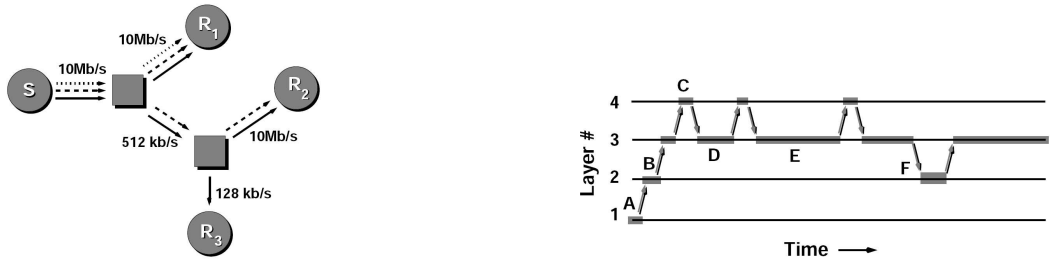
A la manière de TCP Vegas [BOP94], les algorithmes proactifs utilisent la gigue pour détecter une congestion future. En effet, une gigue importante est un signe d'augmentation du travail effectué par les routeurs et donc un signe de congestion dans un futur proche [Bol93]. L'algorithme LDA+ [SW00] utilise cette propriété pour prévoir une congestion et diminuer le débit. En utilisant les paquets de type APP du protocole RTCP, le récepteur informe l'émetteur de la mesure de la bande passante R du goulot d'étranglement qu'il a effectuée. Cette mesure est obtenue après que l'émetteur ait envoyé deux paquets d'investigation le plus rapidement possible. Avec la formule suivante, l'émetteur peut calculer R :

$$R = \frac{TaillePaquetInvestigation}{TempsEntreLaReceptionDeDeuxPaquets}$$

Ensuite, pour obtenir l'équité avec le protocole TCP, le débit sortant est le minimum entre le débit qu'aurait TCP (calculé à l'aide d'un modèle mathématique) et R.

Le contrôle de congestion Multicast

Paradoxalement, les solutions de contrôle de congestion pour les communications Multicast sont plus simples que celles proposées pour les communications Unicast. La plupart d'entre elles sont des adaptations de la solution RLM (pour Receiver-driven Layered Multicast [MJV96]). Dans cette solution, l'émetteur effectue plusieurs codages, à différents débits. De plus ces codages peuvent être complémentaires, i.e. la réception du deuxième flux permet d'augmenter la qualité du précédent. Ainsi, ce sont les récepteurs qui vont effectuer le contrôle de congestion en s'abonnant uniquement aux flux dans lesquels ils ne subissent pas de perte. De cette façon, les récepteurs peuvent dégrader harmonieusement la qualité de réception (voir figure 2.16(b)) et la transmission des flux sur le réseau va automatiquement s'équilibrer (voir figure 2.16(a)).



(A) La transmission avec RLM

(B) Le contrôle de congestion d'un récepteur

FIG. 2.16 – *RLM (Receiver-driven Layered Multicast)*

2.3.2 La compensation des erreurs de transmission

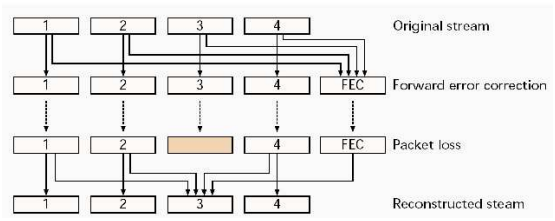
De même que pour le contrôle de congestion, le mécanisme de gestion des erreurs doit limiter la latence et la gigue de la transmission dans le but d'éviter que la donnée ne rate son échéance. C'est pour cela que la retransmission est une solution qui n'est généralement pas utilisée.

Ainsi la plupart des mécanismes mis en œuvre sont basés sur le caractère prédictif des données temps réel. En effet, elle sont souvent issues de l'échantillonnage de phénomènes continus, comme le son, les mouvements, la variation de température, etc.

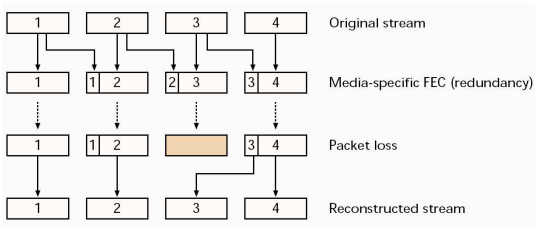
Dans certains cas, la sémantique des données fait que la transmission doit être complètement fiable. Par exemple dans les jeux vidéo, certaines informations sont utilisées pour prendre des décisions importantes sur l'évolution du scénario ou sur l'état du jeu. Pour ce type d'applications, la fiabilité est obtenue à tout prix : même si une donnée rate son échéance, alors elle sera enregistrée dans un état retardé temporellement dans le but d'activer un mécanisme de convergence des états répliqués. Nous proposons au paragraphe 6.3.2 une description plus détaillée d'un tel mécanisme : le *Dead Reckoning*.

D'un autre côté, pour la transmission de données temps réel, la normalisation s'est focalisée sur la spécification de mécanismes de compensation d'erreurs qui limitent la latence mais qui ne fournissent pas une fiabilité totale. Dans ce paragraphe, nous allons présenter succinctement les solutions qui sont proposées dans des RFCs "types de payload RTP", celles décrites dans [PHH98] ainsi que les techniques issues du traitement du signal.

L'approche protocolaire (FEC pour Forward Error Correction) consiste à ajouter de la redondance dans les flux pour compenser la perte d'un paquet. La figure 2.17(a) montre la technique du paquet de parité. Ce paquet est calculé en effectuant une opération "ou exclusif" parmi un certain nombre de paquets précédents (quatre dans l'exemple). Ainsi, si l'un d'entre eux est perdu, le paquet de parité permet de le reconstruire. Ce mécanisme fait l'objet d'une RFC [RS99] qui préconise l'utilisation de la même session RTP et l'utilisation du numéro de payload pour que le ou les récepteurs puissent différencier un paquet de donnée d'un paquet de réparation. Cette technique augmente la latence de bout en bout car l'application doit attendre que l'ensemble des paquets soit généré pour construire le FEC et l'envoyer. La figure 2.17(b) nous montre une



(A) Le FEC de parité



(B) Le FEC spécifique au média

FIG. 2.17 – *Les FEC (Forward Error Corrections)*

autre technique de FEC : l'émetteur ajoute une copie compressée (i.e. de plus faible qualité) d'un paquet dans son successeur. De cette façon, une perte est compensée dès la réception du paquet suivant [HSHW95, BVG98].

Le traitement du signal, grâce aux propriétés des médias, peut aussi être un support à la compensation des erreurs de transmission. On appelle ces techniques des *dissimulations* d'erreurs. Contrairement aux techniques protocolaires, seuls les récepteurs vont intervenir dans le mécanisme. D'après [PHH98], elles sont efficaces si les pertes plafonnent à 15% et si les paquets contiennent entre 4 et 40 ms (testé pour le transport de sons, 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). Les techniques les plus simples sont appelées *réparation par insertion*. Elles consistent à remplacer les données manquantes par un blanc, un bruit ou par le paquet de données précédent. Certains articles [ML50, War82] proposent des comparaisons de ces mécanismes. L'avantage de ces techniques est qu'elles ne sont pas coûteuses en calcul (pour les téléphones mobiles par exemple).

Les techniques de *réparation par interpolation ou régénération* sont plus complexes mais fournissent de meilleurs résultats. Elles sont basées sur des traitements du signal, comme par exemple le remplacement par des données issues d'un "morphing" fréquentiel entre les deux paquets entourant la perte [SSYG96].

2.3.3 Conclusion

Bien que ces contrôles de congestion (Unicast ou Multicast) soient élégants et efficaces, la mise en œuvre reste très difficile. En effet, il est parfois préférable d'interrompre une communication que de dégrader la qualité (comme cela est couramment fait par les utilisateurs de téléphones portables). Il faut donc dimensionner des bornes maximale et minimale pour le débit, ceci en fonction de l'application. De plus, il faut disposer de systèmes de codage des médias qui peuvent être capables de donner le débit désiré, ou alors changer de système/format en cours de communication. Ces algorithmes sont en pratique assez peu utilisés. De plus, avec ou sans contrôle de congestion, une application qui utilise RTP/RTCP reste sujet aux pertes provoquées par le réseau.

Incontestablement, les mécanismes de compensation d'erreurs forment un complément aux solutions de transport présentées dans ce chapitre. Cependant, elles restent majoritairement spécifiques aux médias et supposent que les pertes sont aléatoirement réparties durant la trans-

mission. Pour les réseaux câblés, cette supposition est rarement vérifiée, dans la mesure où la distribution des pertes de paquets est généralement uniforme.

Cependant, toutes les solutions de compensation présentées sont compatibles avec les mécanismes Unicast et Multicast, puisqu'elles ne demandent aucun couplage entre l'émetteur et son ou ses récepteur(s).

2.4 Conclusion

La conception d'Applications Multimédia Interactives et Distribuées apporte son lot de contraintes. En effet, la notion d'interactivité impose une certaine régularité dans les délais de communication et dans les débits disponibles. Ces contraintes entrent en contradiction avec la sémantique de partage des ressources de l'Internet, puisque celles-ci sont fonction du nombre d'utilisateurs présents et actifs sur le réseau. De plus, la construction de solutions de routage Multicast (au niveau IP) introduit une complexité importante qui freine la mise à disposition de tels services.

Bien que pour le transport de données au sein d'un groupe, la tendance actuelle soit de développer des solutions Multicast au niveau applicatif, il n'existe pas encore de solution complètement satisfaisante. En effet, le Multicast IP souffre toujours du problème de déploiement, les architectures centralisées sont difficiles à maintenir et ne passent pas toujours à l'échelle et les solutions Pair à Pair ne sont pas encore matures.

Au niveau du transport, de bout en bout, RTP est le protocole qui s'est imposé puisqu'il fournit le minimum des services nécessaires à des transmissions temps réel, tout en étant compatible avec les solutions Multicast et en laissant les applications gérer les mécanismes sensibles et spécifiques : le contrôle de congestion et la gestion des erreurs de transmission.

Nous allons maintenant nous intéresser aux applications fournissant une interactivité entre musiciens géographiquement distribués, appelées dans la littérature les NMP pour Networked Musical Performances [LW01]. Compte tenu des concepts présentés dans ce chapitre, nous allons proposer au chapitre suivant une analyse des systèmes de NMP qui tiendra compte des solutions réseaux utilisées (protocoles, mise en relation des utilisateurs, routage des informations, spécificité des formats des données, etc).

3

Le Concert Réparti

Sommaire

3.1 Introduction : le principe de l'interactivité musicale en ligne

Les systèmes de Performance Musicale Interactive Distribuée (PMID, en anglais NMP pour Networked Musical Performance) fournissent un cadre d'étude intéressant pour les AMID. En effet, la pratique musicale en groupe est effectuée généralement dans le contexte suivant :

- Les musiciens sont dans la même pièce
- Ils peuvent communiquer verbalement ou visuellement
- Ils peuvent jouer de la musique en partageant un tempo commun
- Chacun d'eux utilise ses sens pour pratiquer son instrument : les sens du toucher et de l'audition permettent de calibrer la façon de pratiquer l'instrument
- Une coordination entre les musiciens est nécessaire pour effectuer les réglages des volumes, leurs positions dans l'espace, les morceaux à jouer et la façon de les jouer (structures, tonalité, etc).

De plus, chaque style, chaque groupe possède ses astuces pour interagir. Par exemple dans les musiques improvisées comme le Jazz, le Flamenco ou la musique classique Indienne un ensemble de conventions, connues de tous les musiciens, leur permet de jouer ensemble facilement dès la première rencontre. Ces conventions, qu'elles soient rythmiques ou harmoniques sont importantes car elles permettent de simplifier au maximum la difficulté de la pratique musicale à plusieurs.

Ensuite, au moment du travail de groupe, l'interactivité, la spontanéité est une combinaison de beaucoup d'éléments spécifiques. En effet, qui n'a jamais vu un groupe de rock finir un morceau sur le saut d'un musicien, permettant de synchroniser la dernière note ? Il existe de nombreux exemples : le chef d'orchestre synchronise rythmiquement les différents musiciens d'un orchestre philharmonique et permet de contrôler une partie de l'interprétation des musiciens. Les phases d'appel sont souvent utilisées pour lancer un changement dans les ensembles de percussions ou dans les groupes de Jazz.

Ainsi, la conception d'une AMID pour ce type d'interactions couvre tous les aspects de l'interactivité telle qu'elle a été décrite dans le chapitre 1, en effet :

- Il faut "téléporter" les sons et éventuellement les images des autres musiciens (*simulacres sensoriels*) avec un système qui reproduit au mieux les conditions psycho-perceptives des musiciens, lorsqu'ils sont situés dans une même salle
- Le système doit être capable de fournir une interface de communication permettant d'avoir un tempo commun et éventuellement des informations de position dans l'espace (construction d'une scène virtuelle, spatialisation du son). Cela correspond aux simulacres représentatifs
- Le système doit permettre aux musiciens de se coordonner pour paramétrer le système en fonction de la musique qui est jouée. Par exemple en configurant les volumes ou les effets, en différenciant les rôles de chaque musicien (chef d'orchestre, soliste, etc), ou alors en configurant un métronome. Ce sont des *simulacres sociaux*.

La combinaison des simulacres à fournir pour se rapprocher d'une situation "classique" de concert pousse un système de PMID à avoir une architecture complexe dans laquelle il existe plusieurs types d'applications, propres aux rôles de chacun des acteurs d'un concert (voir figure 3.1). Pour l'instant nous considérons qu'il y a trois rôles possibles : musicien, ingénieur du son et spectateur.

Nous supposons que la partie de l'application propre aux musiciens doit être composée de "simulacres", dont celui de base est l'écoute mutuelle. Pour cela, l'application doit posséder un système de communication de n à m (qui lui permet de simuler l'environnement) et un système d'acquisition des sons joués par les musiciens.

L'ingénieur du son a par nature une fonction centralisée, puisqu'il est habituellement capable de contrôler les réglages pour les musiciens, comme les effets audio, les volumes des retours et ceux du public[18]. Il participe donc à l'interactivité du système et joue un rôle important dans le confort des musiciens.

Le troisième type d'acteur d'un système de PMID est le spectateur. La diffusion "Live" de la musique produite permettrait d'envisager des cas pratiques intéressants pour les PMID. Par exemple pour effectuer des auditions de musiciens en ligne, ou encore organiser des événements artistiques avec des musiciens populaires, voir même d'organiser un concours en ligne et faire voter les spectateurs par SMS.

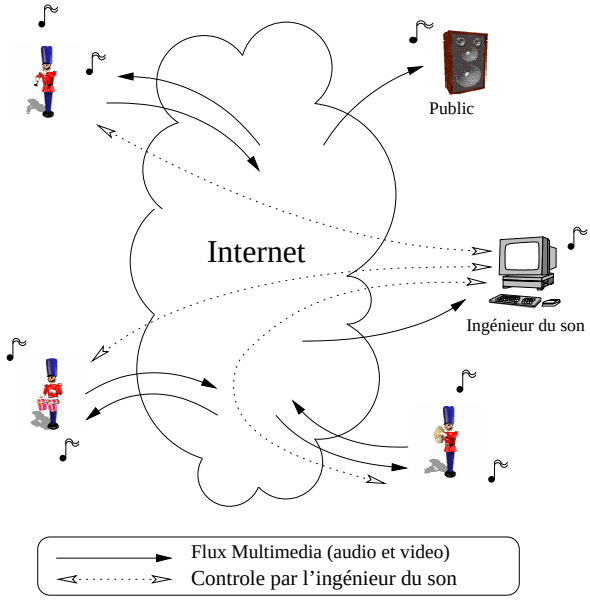


FIG. 3.1 – *Le concert réparti, l'architecture fonctionnelle*

Nous allons par la suite nous intéresser principalement à la partie du système qui est dédiée aux musiciens et présenter les systèmes de PMID existants.

Le principal problème qui se pose dans ces systèmes est la gestion de la latence et de la gigue introduite par le réseau. La latence est fonction des caractéristiques des divers éléments du réseau (cartes réseau, câbles, commutateurs, modems, routeurs, ...) mais reste soumise aux lois de la physique. En conséquence, la latence n'est pas seulement due au travail des éléments d'acheminement du réseau, mais aussi due à la distance physique entre les hôtes qui communiquent. D'autre part, la gigue peut être importante sur les réseaux IP (Internet) car la charge de travail supportée par les routeurs est fonction des flux qui transitent par ceux-ci.

[18]Sur scène le son de chaque musicien est mixé pour leur être renvoyé : c'est ce que l'on appelle les retours. Cela permet à chaque musicien d'avoir un espace sonore dédié dans lequel il peut choisir la balance des instruments selon son confort. Le mixage envoyé au public est lui différent des retours, puisque calibré en fonction des instruments qui doivent être mis en valeur (la voix, le soliste, etc).

La pratique musicale et la latence ne font pas bon ménage puisque la perception des musiciens est suffisamment fine pour qu'ils aient l'impression que les sons qu'ils jouent soient instantanément entendus. C'est pour cette raison que la configuration des buffers des cartes son dans les applications audionumériques est un cauchemar pour la majorité des musiciens. Sur Internet les délais sont tellement importants que certains auteurs pensent qu'il est impossible d'effectuer des performances audio distribuées interactives [KI98, BCF+98]. Cependant, nous allons voir dans cet état de l'art que cela est possible, dans la mesure où la pratique musicale peut être adaptée à Internet, introduisant éventuellement des nouveaux styles musicaux propres [CSZ+04].

Certaines études proposent d'évaluer la latence à laquelle il n'est plus possible pour des musiciens de jouer ensemble[Sch02, CSTZ05, CZS+05]. Bien que dans la pratique musicale nous pouvons trouver de nombreux exemples dans lesquels les musiciens s'adaptent parfaitement à une latence importante, la connaissance de ces seuils est d'une utilité incontestable pour la conception d'une application musicale interactive distribuée.

Au moyen âge, l'acoustique réverbérante des cathédrales imposait des contraintes sur le tempo : le chant grégorien devait être composé de longues notes afin de rendre les textes compréhensibles. Un autre exemple : dans un orchestre symphonique, le chef d'orchestre est généralement en avance vis-à-vis des musiciens. Cependant, l'exemple le plus proche de la transmission sur réseau est celui de l'organiste d'église : l'orgue utilise un mécanisme pneumatique qui provoque une latence importante entre la pression d'une touche et la production du son. Cette latence est suffisamment importante pour que l'organiste la perçoive et pour qu'un non-initié ne soit pas capable d'anticiper le retard pour jouer de façon synchronisée avec une chorale.

Ces exemples montrent que dans plusieurs cas de figure, la pratique musicale s'est adaptée au contexte d'exécution, mais avec un travail complexe d'apprentissage pour les musiciens. Le but n'étant pas de compliquer la tâche des musiciens mais de leur permettre de s'adapter au problème de la latence, un système de PMID devra obligatoirement offrir une solution pour la latence.

Les systèmes permettant à des musiciens d'interagir ensemble et en temps réel à l'aide d'un réseau sont des AMID à part entière. Nous allons présenter dans ce chapitre un état de l'art des systèmes et des problématiques spécifiques à cette application.

Au paragraphe 3.1, nous présentons les enjeux de l'interactivité musicale en ligne avec une proposition d'architecture basée sur le rôle de chaque utilisateur d'un tel système. Nous allons ensuite proposer une description des simulacres de base à mettre en œuvre : celui du son au paragraphe 3.2 avec une analyse de l'utilisation des codages MIDI et MIC, puis celui du temps physique au paragraphe 3.3 avec une analyse des ambiguïtés temporelles liées à la transmission des données avec latence. Nous proposons au paragraphe 3.4 une synthèse des systèmes existants pour finalement conclure au paragraphe 3.5.

3.2 La numérisation du son

Dans une application de PMID, la numérisation du son est un aspect important de l'interactivité entre musiciens. En effet, le codage et le décodage constituent le simulacre sensoriel auditif de base à l'écoute mutuelle. La qualité de la numérisation va alors avoir un impact direct sur la puissance immersive de l'application. Nous pouvons distinguer principalement deux types de numérisation du son : le codage "logique" (le MIDI pour Musical Instrument Digital Interface) et codage continu (le MIC pour Modulation par Impulsion et Codage, en anglais PCM pour Pulse

Code Modulation).

3.2.1 Le MIDI

L'IMA[19] a publié en 1983 les spécifications du MIDI 1.0. Les premiers instruments compatibles MIDI ont alors rapidement fait leur apparition sur le marché. L'objectif original était de pouvoir interconnecter des instruments de différents constructeurs pour permettre le contrôle de fonctions communes comme la pression sur une pédale, l'activation ou le relâchement d'une note, etc. Le codage MIDI est en fait plus qu'un codage, c'est un protocole de contrôle. Dans la partie codage, il est possible de décrire un son de façon logique avec l'envoi d'un événement du type "la note numéro 30 a été activée avec une vélocité de force 70, en utilisant un son de piano". A la réception d'un tel événement, le synthétiseur peut recréer le son de façon proche. Le protocole MIDI est asynchrone et utilisé au départ sur des liaisons série entre les instruments.

Ce type de codage peut être considéré comme un codage "sémantique" de la musique puisqu'il consiste à décrire une note avec son rythme (la durée d'une note est limitée par l'événement d'activation et celui de désactivation). Cependant, pour un codage donné, il existe une infinité d'interprétations possibles, puisque les synthétiseurs possèdent tous leurs propres banques de sons. De plus, la qualité de synthèse diffère selon le matériel, le logiciel, etc. D'un point de vue purement perceptif, ce codage efface toute les subtilités de jeu de l'instrumentiste et ne permet pas la numérisation de sons complexes, et particulièrement ceux qui ne peuvent pas être décrits avec seulement des notes et des transitions, comme l'articulation des mots par une bouche.

Bien que le MIDI soit critiquable sous beaucoup d'aspects (faible qualité du codage et asynchronisme du protocole), il a ouvert un champ d'applications important et très utile à la communauté des musiciens comme l'acquisition de partition, le contrôle de table de mixage, etc.

3.2.2 Le MIC

L'autre système de numérisation des sons, le MIC, est un codage adopté dans les standards de téléphonie numérique et audionumérique. Le principe consiste à numériser une grandeur analogique (le son est transformé en signal électrique à l'aide d'un microphone) en prélevant la valeur du signal à des instants donnés. La mesure du signal (amplitude) est alors destinée à être écrite dans une zone mémoire (fichier, buffer, etc).

L'amplitude maximum est définie par la taille de codage, définissant ainsi une échelle de quantification. Par exemple, si 16 bits sont utilisés pour coder une amplitude, alors l'amplitude peut prendre $2^{16} = 65536$ valeurs possibles. On dit alors que l'on utilise une quantification sur 16 bits (certaines normes utilisent une échelle de quantification logarithmique, d'autres une échelle linéaire). La quantification étant effectuée avec des valeurs discrètes, elle introduit une marge d'erreur qui est caractérisée par la notion de signal sur bruit (exprimée en dB pour décibels).

La mesure du signal se fait de façon périodique (l'intervalle d'échantillonnage), définissant ainsi une fréquence d'échantillonnage. Selon la relation de Shanon, les fréquences d'un signal seront restituées fidèlement si la fréquence d'échantillonnage est supérieure ou égale au double de la plus grande fréquence du signal.

Il existe alors deux paramètres principaux définissant la qualité de numérisation du son : la fréquence d'échantillonnage et la quantification. Dans le cas de la numérisation des sons, le degré

[19]International MIDI Association, composée notamment de Roland, Yamaha et Korg

de qualité doit être choisi en fonction de la capacité du système auditif humain et de l'application. Ainsi pour la téléphonie numérique, la contrainte principale est que les phrases doivent être intelligibles, la numérisation se fait alors à une fréquence de 8000 Hz et une quantification de 8 bits (débit de 64 kb/s). Pour la vente de musique sur CD, la restitution doit respecter le critère minimal de qualité pour l'oreille humaine, la qualité choisie est de 44100Hz avec une quantification à 16 bits sur deux canaux (stéréo), correspondant à un débit de 2 x 0,7 Mb/s. C'est la qualité minimale calculée à partir des propriétés perceptives de l'oreille humaine.

Les systèmes de compression audio (mp3, ogg, etc) sont généralement basés sur des codages MIC, puisque c'est le format de codage interne des cartes sons, des CD audio, de la téléphonie, etc.

3.2.3 Quel type de codage pour l'interaction musicale distribuée ?

Il est évident que dans un système de musique interactive distribué il est préférable de privilégier la qualité de restitution du son, puisque celle-ci permet de plonger les musiciens dans un environnement qui semble le plus réel possible. Le codage MIC doit donc être privilégié même s'il est beaucoup plus consommateur en bande passante que le codage MIDI. Son utilisation nécessite cependant un débit sortant d'au moins 0,7Mb/s.

Ajoutons que pour une transmission des sons sur un réseau, le MIDI et le MIC n'ont pas les mêmes besoins de fiabilité. Avec un codage MIC, la perte ponctuelle de données n'aura d'effet que sur la durée des données elle-même. Par exemple la perte d'un paquet de 20ms de données va provoquer un silence de 20ms lors de la restitution (s'il n'y a pas de mécanisme de compensation d'erreur, cf paragraphe 2.3.2).

En MIDI, la perte d'un paquet va avoir un impact plus important. Par exemple, si un paquet contenant l'information "noteof" est perdu, alors la dite note continuera à résonner tant que le musicien ne rejouera pas cette note (génération d'un nouveau noteof). L'intégration de données MIDI dans RTP fait actuellement l'objet d'une proposition de normalisation [LW06] qui propose une solution pour la fiabilité de la transmission.

Dans la littérature, le MIDI comme le MIC ont été choisis dans différents systèmes de PMID. Cependant, dans tous les cas, le système doit mixer les flux audio pour que les musiciens puissent s'écouter mutuellement. C'est ainsi que va apparaître le problème de la latence entre les applications des musiciens, ainsi que ses effets sur les systèmes de PMID.

3.3 Une transmission instantanée pour l'interaction musicale : le rêve de Jules Verne

Depuis l'apparition de l'électricité, les scientifiques ont imaginé que l'on pourrait rapprocher les gens malgré leur distance physique. Bien qu'ils n'aient pas encore imaginé à cette époque les techniques de numérisation utilisées aujourd'hui, les usages pour le grand public étaient décrits dans nombres d'ouvrages tels que [Per89].

Par exemple, Jules Verne, dans son texte nommé "une Ville Idéale" (1875) décrit un système permettant à un musicien d'effectuer un concert dans lequel le public n'est pas géographiquement localisé au même endroit que le concertiste. Voici un extrait de ce texte :

En tout cas, à gauche, se dressait un vaste monument de forme hexagonale, avec une superbe entrée. C'était à la fois un cirque et une salle de concert, assez grande pour permettre à l'Orphéon, à la Société Philharmonique, à l'Harmonie, à l'Union chorale, à l'Harmonie de la Neuville, à la Lyre Amicale, à la Fanfare du Faubourg de Beauvais et à la Fanfare municipale des Sapeurs-Pompiers volontaires, d'y fusionner leurs accords.

Dans cette salle – on l'entendait de reste – une foule immense applaudissait à la faire crouler. En dehors s'étendait une longue queue, à travers laquelle se propageait l'enthousiasme de l'intérieur. A la porte s'étalaient des affiches gigantesques, avec ce nom en lettres colossales :

PIANOWSKI PIANISTE DE L'EMPEREUR DES ÎLES SANDWICH

Je ne connaissais ni cet Empereur ni son virtuose ordinaire.
- Et quand Pianowski est-il arrivé ? demandais-je à un dilettante, reconnaissable à l'extraordinaire développement de ses oreilles.
- Il n'est pas arrivé, me répondit cet indigène, qui me regarda d'un air assez surpris.
- Alors, quand viendra-t-il ?
- Il ne viendra pas, répliqua le dilettante.
Et, cette fois, il avait parfaitement l'air de me dire : "Mais vous, d'où venez-vous donc ?"
- Mais, s'il ne vient pas, dis-je, quand donnera-t-il son concert ?
- Il le donne en ce moment !
- Ici ?
- Oui, ici, à Amiens, en même temps qu'à Londres, à Vienne, à Rome, à Petersbourg et à Pékin !
- Ah ça ! pensais-je, tous ces gens-là sont fous ! Est-ce que, par hasard, on aurait laissé fuir les pensionnaires de l'établissement de Clermont ?
- Monsieur, repris-je...
- Mais, monsieur, me répondit le dilettante, en haussant les épaules, lisez donc l'affiche ! Vous ne voyez pas que ce concert est un concert électrique !
Je lus l'affiche !... En effet, dans ce même moment, le célèbre broyeur d'ivoire, Pianowski, jouait à Paris, à la salle Hertz ; mais au moyen de fils électriques, son instrument était mis en communication avec des pianos de Londres, de Vienne, de Rome, de Pétersbourg et de Pékin. Aussi, lorsqu'il frappait une note, la note identique résonnait-elle sur le clavier de ces pianos lointains, dont chaque touche était mue instantanément par le courant voltaïque !

Je voulus entrer dans la salle ! Cela me fut impossible ! Ah ! je ne sais pas si le concert était électrique, mais je puis bien jurer que les spectateurs, eux, étaient électrisés !

Aujourd'hui, grâce aux techniques de numérisation des informations et du développement d'Internet dans les centres d'enseignement, les entreprises, chez les particuliers, etc, les expériences telles que celle décrite par Jules Verne peuvent être menées à bien. Avec l'arrivée de l'Internet, et plus particulièrement de la technologie IP, le réseau devient un outil qui passe à l'échelle d'un nombre exceptionnellement grand d'utilisateurs. En effet il est possible théoriquement de connecter plus de 4 milliard de machines en IP version 4 et plus de $3,4 \times 10^{38}$ en IPv6

(128 bits pour l'adresse).

L'état actuel des connaissances en physique et en informatique nous montre que la vision de Jules Verne n'est pas tout à fait exacte, du point de vue de la transmission d'informations : même si elle atteint sa limite physique (la vitesse de la lumière), la transmission d'informations ne peut se faire "instantanément". En effet, les délais sur les réseaux sont mesurables (même de façon inexacte) et prend des valeurs qui peuvent être suffisamment grandes pour qu'un utilisateur humain puisse les percevoir.

3.3.1 Les lois de la physique

Suivant les mêmes hypothèses que Jules Verne, la plupart des projets de PMID proposent comme argument que sur un réseau à faible latence la transmission d'informations (audio) est suffisamment rapide pour être "instantanée" (certains auteurs parlent de "téléportation" [CS01]). Cependant, même si la transmission se fait à la vitesse de la lumière, la latence reste trop importante pour des grandes distances (voir le tableau 6.4).

Galilé fut probablement le premier scientifique à vouloir mesurer la vitesse de la lumière, mais sans succès. Différentes expériences se sont succédées pour obtenir une valeur : Römer en 1676 avec les satellites de Jupiter, Fizeau en 1849 avec un procédé de roues dentées occultant périodiquement la lumière puis Foucault en 1850. En 1857 Kirshof montre avec l'équation des télégraphistes que la vitesse des ondes électriques est équivalente à celle de la lumière (confirmé par Maxwell en 1867). Finalement, nous estimons aujourd'hui la vitesse de la lumière à environ 300 000 km/s.

Le tableau 6.4 nous montre les délais pris par la lumière pour aller d'un point à l'autre du globe terrestre. Nous remarquons alors que pour deux points diamétralement opposés de la surface terrestre, la limite physique du délai de transmission est de 66ms. Cette valeur est supérieure à ce que nous pourrions appeler l'instantanéité (pour un être humain manipulant un instrument de musique). En effet, avec un son très bref envoyé à un système stéréo, en insérant progressivement un décalage entre les enceintes, l'oreille commence à percevoir deux sons distincts à partir de 20ms. C'est cette valeur que nous considérons comme la limite de l'instantanéité pour la musique.

Sachant que la vitesse du son (300 mètres seconde) est bien plus faible que la vitesse de la lumière, le problème d'instantanéité se retrouve en conditions réelles, comme dans les fanfares de rue ou dans les orchestres philharmoniques. En fait, à partir de 6 mètres le son met 20ms à arriver, ce qui est supérieur à la distance entre musiciens dans ce type de formation. C'est probablement la raison pour laquelle nous y trouvons systématiquement un chef d'orchestre qui assure la fonction de synchronisation entre les musiciens. Dans certaines fanfares de rue très longues, plusieurs chefs d'orchestre sont surélevés par rapport aux musiciens pour se voir mutuellement et donner le tempo aux musiciens qui sont proches.

3.3.2 Retour vers le futur : les ambiguïtés temporelles

Nous allons maintenant mettre en avant les ambiguïtés temporelles auxquelles les musiciens sont soumis dans un système de communication de groupe, si la latence est importante. Supposons deux musiciens géographiquement distants désirant interagir ensemble, ils doivent alors au moins

s'entendre mutuellement pour exécuter une pièce musicale. La musique étant composée de sons (éventuellement de notes, de rythmes, de parole), un système de transmission de flux multimédia (voir le chapitre 2) doit être capable de transmettre rapidement et avec un délai constant les sons produits par chacun. En effet, lorsqu'un musicien fait durer une note ou attend un certain temps entre deux notes consécutives, il le fait généralement de façon consciente et cette durée est la même pour toutes les personnes présentes dans la pièce.

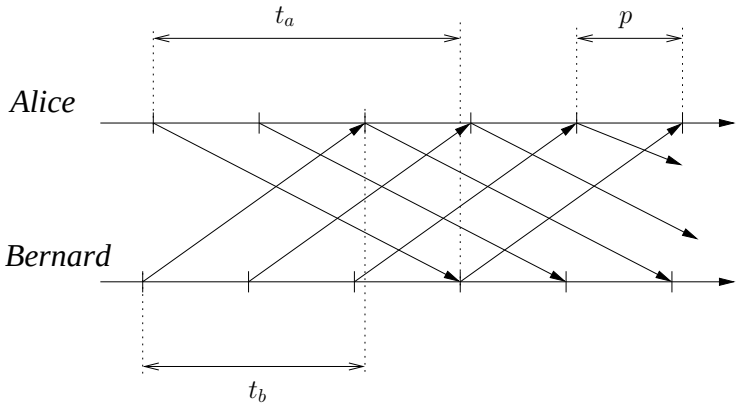


FIG. 3.2 – *La transmission mutuelle de musique*

Appelons ces musiciens Alice et Bernard (figure 3.2). La musique est numérisée et envoyée par paquets à travers le réseau. Ce sont les flèches allant de Alice vers Bernard pour le son de Alice et les flèches allant de Bernard vers Alice pour la musique produite par Bernard. Le système de transmission de flux multimédia, grâce à des techniques de bufferisation parvient à maintenir des latences constantes : t_a ms pour le flux provenant de Alice et t_b ms pour celui provenant de Bernard.

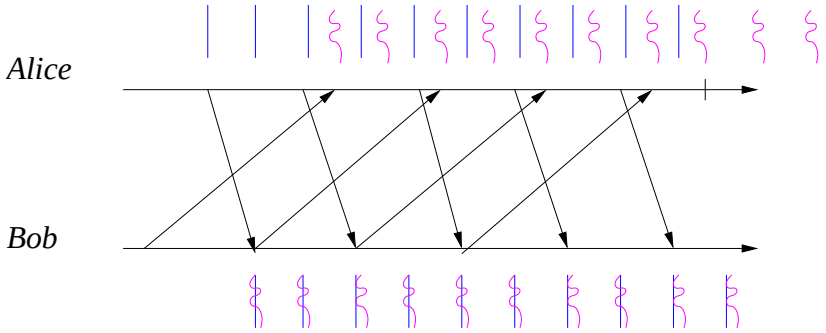


FIG. 3.3 – *L'ambiguïté temporelle liée aux délais*

Regardons maintenant ce qu'il va se produire si Alice et Bernard essaient de jouer ensemble un morceau dans lequel ils sont supposés jouer des notes simultanément, en suivant un tempo (figure 3.3). Le tempo en musique est normalement une composante temporelle commune à tous les musiciens qui jouent ensemble : il possède une période qui peut varier légèrement ou progressivement dans le temps.

Ainsi, Alice commence à jouer de la musique à un tempo donné (représenté par des barres verticales sur la figure 3.3). Bernard recevant la musique joué par Alice va suivre le tempo de

celle-ci. Cependant, la musique qu'il joue sera reçue par Alice avec le délai du réseau et de la bufferisation. Les notes entendues simultanément par Bernard seront perçues par Alice avec une différence de temps de $t_a + t_b$ ms (le tempo de Bernard est symbolisé par les vagues verticales). Sur la figure, il y a une désynchronisation gênante de la musique, puisqu'elle ne permet pas à Alice de partager le tempo avec Bernard.

Cependant, l'oreille humaine ne possède pas une précision extrême et tolère les latences jusqu'à un certain degré, que nous appellerons le seuil de perception des décalages. Avec une mesure de ce seuil, nous pouvons définir si oui ou non, pour une configuration donnée, la transmission peut être instantanée ou pas.

3.3.3 La mesure du seuil de perception des décalages temporels par l'oreille humaine

Selon les expériences décrites dans la littérature, le délai tolérable par les musiciens varie entre 20 et 80 ms. Cette variation dépend en fait de plusieurs facteurs qui entrent en compte durant les tests : le niveau de précision du musicien, le tempo du morceau (plus le morceau est lent, moins le délai introduit est gênant).

La plupart des tests ont été effectués sans le réseau, en retardant artificiellement le son des musiciens. Par exemple en plaçant deux musiciens dans deux pièces différentes [Sch02] et en analysant automatiquement la désynchronisation de rythmes battus avec les mains. L'asynchronisme fut mesuré avec des algorithmes détectant la simultanéité des battements et l'accélération ou le ralentissement. En effectuant un test avec différentes valeurs, il résulte que pour des délais supérieurs à 30 ms, les musiciens ont tendance à ralentir. De plus, si les musiciens se mettent d'accord sur le fait que l'un d'eux est "leader" du tempo, alors ce délai monte à 50 ms.

Ce test s'est effectué avec une séparation totale des musiciens (ils ne pouvaient se voir car dans deux pièces différentes). Dans les tests décrits dans [CSTZ05], les musiciens (professionnels) étaient situés face à face. Deux types d'expériences ont été proposées : une dans laquelle les sons provenant de l'autre musicien sont retardés et l'autre dans laquelle l'entrée audio de chaque musicien est retardée, impliquant que le musicien s'entend lui même avec le délai. Avec un retard local, sur le même morceau de musique, le seuil semble plafonner à 65 ms contre 50 ms pour la première configuration. Le seuil de jouabilité mesuré est d'environ 50 ms, avec un résultat intéressant : *l'introduction d'un retard local semble permettre aux musiciens de tolérer des délais plus importants*. En effet, ce test introduit le fait que les musiciens sont plongés dans plusieurs espaces sonores : celui de la musique jouée localement (la perception est instantanée via la transmission locale dans la salle) et celui provenant du réseau. L'introduction d'un retard dans la perception locale permet probablement aux musiciens de partager un tempo commun. Nous montrerons dans la troisième partie (paragraphe 8.4) que l'utilisation de délais importants (jusqu'à plusieurs secondes) est en fait très efficace dans la construction d'une interaction musicale synchrone, et cela grâce à une étude d'opinion sur notre système de PMID (pour Performance Musicale Interactive et Répartie), effectuée par des utilisateurs.

3.4 Les PMID : les Performances Musicales Interactives et Réparties

Depuis 2000, la mise en œuvre de PMID s'est considérablement développée, avec une augmentation significative des expérimentations à grande distance. En 2002, une performance fut

mise en place entre les universités de Stanford et McGill (Montréal)[20] en utilisant les réseaux basse latence Internet 2 [21] et le réseau CA*net 3 [22] à l'aide d'une application de conférence audio/vidéo. En 2003, c'est durant la conférence NIME 2003 (New Interface for Musical Expression) qu'une démonstration similaire [23] fut expérimentée entre McGill et Princeton. En 2005 pour la même conférence, une autre démonstration similaire fut testée avec trois sites distants : Vancouver, Marseille et Troy.

Ces différentes démonstrations, sans faire l'objet de publications scientifiques, montrent que les technologies réseau de pointe ont aujourd'hui atteint un niveau de performance permettant de supporter la charge de transmission de flux audio de type MIC, et avec des latences faibles. Cependant, la latence reste tout de même un problème pour l'interactivité musicale à des distances importantes sur le réseau.

Le principal enseignement de ce type d'expérience se trouve dans les documentations techniques : la complexité de l'organisation de tels événements [DD03] est due au budget, à la coordination multi-sites, à la qualité et à la synchronisation des équipements d'acquisition et de restitution, à la propriété intellectuelle des œuvres jouées, à l'enregistrement des performances et à l'éclairage. Ces aspects sont généralement peu abordés dans les publications scientifiques que nous allons présenter maintenant.

3.4.1 Les PMID MIDI

A partir du milieu des années 90 et jusqu'à environ 2002, la plupart des PMID étaient basées sur le standard MIDI, probablement à cause de la demande importante en débit des flux MIC de bonne qualité et la normalisation trop récente de RTP, le protocole de transport de données temps réel[24]. De plus, la grande majorité des publications sont issues de la conférence ICMC (International Computer Music Conference) dans laquelle la communauté réseau n'est pas présente.

C'est en 1996 que l'équipe de Masataka Goto [GHMM96, GNM97] a développé VirJa, un outil de Jazz sessions virtuelles où un humain peut jouer de la musique avec deux processeurs, limitant ainsi l'interactivité mais introduisant le réseau et la problématique de la latence. C'est dans [GN02] qu'ils proposent un système de PMID qui utilise le protocole RMCP (Remote Music Control Protocol) utilisé dans VirJa. Pour gérer la latence, le système *Open RemoteGIG* [GN02] retarde exagérément la musique, avec des délais qui sont calculés en fonction du rythme du morceau et de sa structure harmonique. En effet, beaucoup de musiques sont basées sur l'idée de faire tourner en boucle une suite d'accords, comme en Jazz, en Blues, en Rock, etc. Ainsi, pour jouer un blues à 12 mesures, composées chacune de quatre temps à un tempo 120 bpm[25] il faut introduire un délai équivalent à une boucle, soit 12 x $4/120 = 0,4$ minutes (24 secondes). L'utilisation d'un tel retard permet de synchroniser rythmiquement et harmoniquement la musique des autres musiciens, puisque chacun entend la musique des autres sur le bon accord, mais la musique jouée durant la boucle précédente.

Dans ce système, les musiciens des différents sites n'entendront pas le même résultat, puisqu'ils entendent les autres avec un délai, tandis qu'ils s'entendent personnellement instantané-

[20] `http://cim.mcgill.ca/sre`
[21] `http://www.internet2.edu/`
[22] `http://www.canarie.ca/canet4_f/index.html`
[23] `http://www.music.mcgill.ca/musictech/nime/`
[24] Le protocole RTP (Real Time Transport Protocol) est devenu un standard de l'IETF en 1995. Voir le chapitre 2.2.2
[25] Battements Par Minutes

ment [GNM97].

Le système VirJa est le premier à introduire la notion de synchronisme musical. Dans le système d'enseignement du piano de Young et Fujinaga [YF99], un unique musicien enseigne pendant que les élèves écoutent, il n'y a pas besoin ici de jouer à plusieurs de façon synchronisée. Cependant, c'est dans cet article que nous pouvons trouver la première discussion sur l'utilisation de TCP ou de UDP pour un système musical interactif.

C'est à la conférence ICMC 2000 que nous pouvons trouver le plus de publications proposant des systèmes interactifs musicaux. Plusieurs problématiques intéressantes ont été développées. L'article [Bur00] développe la problématique de la constitution d'une communauté sur le web, en utilisant un serveur web http (pour Hypertext Transfert Protocol) pour contrôler un moteur temps réel audio. La pratique musicale n'y est pas très évoluée, puisqu'elle consiste à programmer en ligne des boucles de sons avec la souris. Cependant, ce système a mis en avant un certain nombre de points essentiels : l'intégration d'un chat dans l'interface permet d'augmenter l'intérêt d'un tel système, puisqu'il permet d'ajouter une dimension sociale qui accroche les utilisateurs.

Comme dans [Bur00], l'article [PAF+00] propose un système dans lequel il n'est pas nécessaire de jouer d'un instrument pour avoir une interaction. Le système étant dédié à des non-musiciens : les utilisateurs ont un contrôle abstrait d'un instrument virtuel, au sein d'un groupe virtuel d'utilisateurs. La musique est calculée par un serveur qui va synchroniser les actions et faire des choix harmoniques dans la musique, permettant ainsi aux utilisateurs d'avoir un pouvoir sur le résultat sans connaissance musicale théorique.

C'est seulement en 2001 que Dannenberg introduit la notion d'horloges physiques synchronisées dans ce type de systèmes. Le MIDI étant généré avec des instruments par des musiciens, le système (fonctionnant sur LAN) doit jouer les notes le plus vite possible en conservant le rythme. Comme le standard MIDI ne contient pas de date (il a été conçu pour des liaisons série) Dannenberg *et al.* ont introduit un mécanisme de synchronisation d'horloges pour dater les échantillons. La solution est de type client/serveur, fonctionne sur LAN et tient compte de la dérive des horloges.

Finalement en 2001 Lazzaro *et al.* illustrent un autre problème de l'utilisation du MIDI sur le réseau : la fiabilité. En effet, le MIDI utilise des événements discrets pour décrire les notes, posant alors des problèmes en cas de perte d'événements. Lazzaro propose alors une extension pour rendre fiable le transport du MIDI au dessus de RTP, qui fait encore aujourd'hui l'objet d'une version en cours d'étude (draft) à l'IETF [LW06]. Dans leur système, Lazzaro *et al.* utilisent le protocole de signalisation SIP (voir 2.2.3) pour mettre en relation les utilisateurs à l'aide d'un nom logique.

Tous ces systèmes fournissent une expertise intéressante, notamment sur les problèmes soulevés durant la conception et l'utilisation : le besoin de synchronisation au niveau musical, le besoin d'adapter le système aux caractéristiques du réseau, la mise en relation des utilisateurs et le besoin d'intégration de simulacres supplémentaires pour enrichir la co-présence. En complément de cette étude, l'article de Fober *et al.* de 2001 [FOL01] fournit une synthèse sur le transport d'événements MIDI sur Internet avec la proposition d'un algorithme de détection de la dérive des horloges pour une communication point à point.

Cependant, le protocole MIDI fournit une qualité médiocre et ne permet pas à des musiciens de retrouver le confort auditif des conditions en salle (comme le réglage des sons, les subtilités

du toucher des instruments, etc). Nous allons maintenant étudier les PMID utilisant le codage MIC.

3.4.2 Les PMID MIC

La plupart des systèmes utilisant le codage MIC suppose que l'utilisateur dispose de la bande passante nécessaire, la plupart de ces systèmes ajoute la vidéo pour augmenter le sentiment de présence. Dès 1998, le projet DVP (Distributed Video Production), financé par la gouvernement Suisse et la commission européenne, a pour but d'expérimenter des répétitions musicales à distance. Le test présenté dans [KOGC98] se répartit sur deux sites (Bonn en Allemagne et Genève et Suisse), l'un hébergeant un chef d'orchestre et l'autre des musiciens de musique classique. L'objectif était de voir si le chef d'orchestre peut diriger facilement l'orchestre à distance. Il en résulte alors que :

- l'utilisation de projecteur et d'éclairage de studio améliore grandement la convivialité du système
- la spatialisation du son est très utile pour le chef d'orchestre
- une latence de 160 ms peut être tolérée par le chef d'orchestre
- il est nécessaire de synchroniser les flux audio pour le passage à une configuration multi-sites hébergeant des musiciens.

Cependant, l'utilisation de la vidéo a mis en évidence un nouveau problème, la localisation dans l'espace : en effet le chef d'orchestre ne peut pas désigner un musicien du doigt, car il pointe vers la caméra. Lors de la restitution de l'image du chef d'orchestre, tous les musiciens pensent alors être désignés.

Ces répétitions, bien que riches en enseignements ont été effectuées sur un réseau ATM (Asynchronous Transfert Mode). Le transfert de données MIC sur réseau IP pour l'interactivité musicale reste alors inexploré. Cette même année, en 1998 l'AES (l'Audio Engineering Society) publie un rapport [BCF+98] sur la faisabilité des systèmes interactifs sur le réseau Internet 2. Les principales limitations identifiées sont d'ordre technique : la latence, l'estampillage des messages les formats audio, le nombre de canaux audio.

Le transport de flux audio MIC temps réel a été expérimenté dans [XC00, CS01]. Les auteurs ont utilisé un studio d'enregistrement pour échantillonner une performance ayant lieu dans un autre pays. Les musiciens étant localisés au même endroit, une synchronisation multi-sites n'était alors pas nécessaire.

C'est à partir de 2003 que les expériences impliquant des musiciens sur différents sites ont commencé. Les expériences en réseau local ont d'abord mis en évidence l'intérêt de la synchronisation multi-sites [Bou03, CSZ+04, Bou04, GDNW04]. Aujourd'hui, seuls les systèmes décrits dans [Bou03, Bou04] proposent une synchronisation de flux MIC pour une configuration avec des musiciens répartis sur plusieurs sites. Ce travail sera décrit dans le chapitre 6. Une version centralisée de synchronisation est proposée dans [GDNW04], reprenant le principe de celle décrite dans le paragraphe 6.4 et publiée dans [Bou04]. Un autre système proche du notre est utilisable sur Internet et propose une solution basée sur un serveur de synchronisation : NinJam[26]. Cependant, à notre connaissance, ce système ne fait l'objet d'aucune publication scientifique.

[26] `http://www.ninjam.com/`

3.5 Conclusion

L'ensemble des travaux publiés par la communauté scientifique a permis de mettre en avant un certain nombre de difficultés. Du point de vue de la co-présence, chaque type de simulacre (voir chapitre 1) apporte son lot de complexité.

Les simulacres perceptifs : le codage à préférer est le MIC car il reproduit plus fidèlement le son produit par les musiciens que le codage MIDI. Pour la partie visuelle, le format de codage doit être suffisamment bon pour permettre une projection à taille réelle des utilisateurs distants. Pour l'audio comme pour la vidéo, le système d'acquisition doit être de la meilleure qualité possible (micro, vidéo, silence dans la pièce, éclairage, format vidéo, etc).

Les simulacres représentatifs : l'élément qui revient le plus souvent est le besoin de synchronisation : de bout en bout (conserver le déroulement du temps), entre deux types de flux différents, mais provenant du même site (audio/vidéo par exemple) et la synchronisation multi-sites.

Comme nous le verrons dans la partie suivante, la synchronisation multi-sites est étroitement liée à la latence du réseau. Le choix de solutions réseaux viables est donc crucial. De plus, dans ces choix, il faudra tenir compte de l'ordonnancement des systèmes d'exploitation (pour l'accès aux ressources d'acquisition et de restitution) et de la synchronisation des horloges.

La représentation de l'espace pose aussi des problèmes car les interfaces multimédias audio et vidéo classiques sont construites en deux dimensions (stéréo et écran). Pour une restitution en trois dimensions des données, il existe des systèmes de spatialisation du son et de visualisation 3D.

Les simulacres sociaux : en fonction de l'utilisateur ciblé, le système doit pouvoir ajouter des simulacres spécifiques à l'application : par exemple un métronome virtuel se configurant en bpm (battements par minute), une partition virtuelle, etc.

Un autre problème, qui est général aux applications multimédia interactives, est la mise en relation d'un groupe d'individus, la constitution d'une communauté. L'application doit pour cela être consciemment adaptée à sa portée sur le réseau (LAN, Internet, groupe de LAN) et disposer de mécanismes/solutions pour retrouver les correspondants (ou en découvrir de nouveaux).

Nous avons vu dans ce chapitre que la principale limitation à la faisabilité des PMID est la latence introduite par le système et la communication. En effet, cette latence introduit des ambiguïtés temporelles qui ne respectent pas les principes habituels de la pratique musicale (lorsque les musiciens sont dans la même salle). Ce type d'ambiguïtés se retrouve en fait dans toutes les AMID dans lesquelles l'interaction est fortement couplée entre les utilisateurs. Plus généralement dans les systèmes distribués, la cohérence des données est un domaine dans lequel le système de communication essaie de masquer les effets des délais sur les dates de prise en compte des messages/opérations. Le problème d'ambiguïté temporelle est en fait typiquement un problème de cohérence des données. C'est la raison pour laquelle nous proposons dans la partie suivante de cette thèse d'étudier la problématique de la cohérence pour les AMID, en effectuant un parallèle étroit avec les modèles de cohérence existant dans le domaine des Mémoires Réparties Partagées (MRP).

Nous développerons ensuite au chapitre 6 des protocoles de cohérence (perceptive) qui représentent des solutions au problème d'ambiguïté temporelle. L'utilisation de l'un de ces protocoles dans le domaine des PMID sera développé dans la quatrième partie de cette thèse. Cette solution est basée sur l'introduction d'un retard local dans les retours des musiciens.

Troisième partie

Des Mémoires Réparties Partagées aux Applications Multimédia Interactives : Nouveaux Modèles et Protocoles de cohérence

La perception n'est pas le constat d'une réalité objective, elle est la négociation d'une présence au monde.

[Derrick de Kerckhove]
Extrait de l'interview effectuée par *Libération - 29 septembre 2001*

4

Des Mémoires Réparties Partagées aux Applications Multimédia Interactives et Distribuées

Sommaire

4.1 Introduction : la cohérence, des MRP aux AMID

Les premiers travaux sur la cohérence sont à notre connaissance issus des protocoles d'échange et de partage de données dans les architectures de machines parallèles [CF78, Lam78]. C'est dans ces travaux que nous pouvons trouver les premières définitions sur la cohérence des données dont les critères de cohérence portaient notamment sur l'ordre d'exécution des opérations (les accès en lecture et en écriture). Par exemple, [Lam79] définit de façon "intuitive" la cohérence séquentielle (notée SC par la suite pour *sequential consistency* puis formalisée plus tard dans [RS96]) qui exprime le respect de l'ordre d'accès linéaire aux variables du programme. Dans ce contexte, la duplication des données ainsi que leur gestion étaient (et sont toujours) directement liées à la sémantique des Mémoires Réparties Partagées (MRP), et visent à augmenter les performances des programmes concurrents s'exécutant sur des machines parallèles.

Dans la littérature, et notamment dans l'étude de Mosberger [Mos93], les modèles de cohérence pour les MRP sont classés en modèles uniformes et hybrides. Les modèles initiaux ne prenaient en compte que les opérations de lecture et d'écriture tandis que d'autres modèles plus riches prenaient en compte les outils de synchronisation comme les barrières ou les mutex. Tandis que les premiers protocoles de cohérence étaient implémentés au cœur du système afin de maintenir le paradigme de l'ordre séquentiel, les propositions de modèles hybrides introduisaient déjà la nécessité d'une forme de prise de conscience par les programmeurs des problèmes de cohérence, afin d'augmenter les performances des machines parallèles.

Cependant, les applications de calculs parallèles possèdent chacune leurs propres besoins et permettent alors le relâchement du modèle de cohérence sous-jacent, augmentant encore une fois les performances du système. Cependant, comme le suggère Mosberger dans son étude, le gain en performances dû aux modèles plus souples apporte son lot de complexités supplémentaires aux programmeurs. Cela nous permet de mettre en évidence deux points essentiels dans la conception de modèles de cohérence : la dépendance de celui-ci au contexte visé (i.e. la sémantique de l'application et de sa plate-forme) ainsi que la complexité d'utilisation et/ou de programmation du système.

Aujourd'hui, la classe d'applications soumises à des contraintes de cohérence dépasse celle des applications de calculs distribués dans les machines parallèles car la duplication de données est utilisée dans les caches web, le Pair à Pair, le calcul sur grille/cluster de machines, les bases de données distribuées, les AMID[27] et plus généralement les systèmes distribués et/ou mobiles. Chacune de ces classes apporte son propre contexte et son propre ensemble de modèles de cohérence dont l'utilisateur doit avoir conscience.

Tout comme les MRP, les AMID ont leur propre contexte dans lequel la sémantique d'accès aux données est spécifique. En effet les MRP sont constituées de processus effectuant un calcul réparti et qui accèdent aux données tandis que dans les AMID, ce sont des utilisateurs humains qui les génèrent et les perçoivent grâce à un ensemble de périphériques. En conséquence, les modèles de cohérence pour les AMID seront basés sur la perception des données partagées (appelées des instances de média) dans un environnement qui doit simuler des propriétés locales (l'illusion d'être proche et de pouvoir interagir ensemble, la *co-présence*).

[27] cf chapitre 1. Nous avons par exemple la réalité virtuelle, la vidéo-conférence, les tableaux blancs partagés, les jeux multijoueurs, les simulations interactives distribuées et les performances musicales interactives

Autrement dit, les AMID sont composées de processus distants qui communiquent et maintiennent des répliques de l'état du système. Cependant, ce n'est pas l'ordre du programme qui est dominant mais la perception temporelle de l'interface par les utilisateurs. Entre autres, la conception d'une AMID doit tenir compte de la rapidité de retour à l'utilisateur des actions que celui-ci effectue, ainsi que du rapport temporel qu'il entretient avec des événements dont il n'est pas à l'origine, et enfin des délais de propagation de l'information.

Ajoutons que les utilisateurs sont des entités autonomes et intelligentes qui évoluent de façon parallèle. Cela constitue une différence essentielle entre les AMID et les MRP puisque dans ces dernières, c'est le paradigme de l'ordre séquentiel voire de la causalité du programme qui prédomine.

Le concert réparti (chapitre 3) est un exemple d'AMID où le sentiment de co-présence est primordial. Dans ce type d'application, l'interaction entre musiciens est possible à condition qu'ils disposent virtuellement d'un battement commun [Bou04, Bou03] (le *tempo*, dont l'unité est en BPM pour Battements Par Minute). En terme de cohérence, cette notion se traduit par le critère de *simultanéité*. Ce critère étant clairement lié au temps physique, d'autres AMID ont des contraintes liées à la notion d'ordonnancement des opérations sur les instances de média. C'est le cas des jeux vidéo de courses automobiles où les événements conduisant à une décision de victoire (passer la ligne d'arrivée) doivent être ordonnés de la même façon pour tous les joueurs. Nous verrons dans le chapitre 6 sur les protocoles pourquoi la simultanéité est aussi (et surtout) un critère pertinent pour ce type de jeux.

L'interactivité dans les AMID

Selon Edwards *et al.* [EM97], l'interactivité entre utilisateurs à travers un réseau se décompose en deux catégories : les *collaborations autonomes* dans lesquelles les participants travaillent indépendamment pendant une période donnée puis vont s'échanger leurs actions au cours d'une période de connexion au système qui va intégrer l'ensemble de leurs travaux. La conception distribuée de plan d'avion [CTCG01], les systèmes de gestion de versions comme CVS [Ca02], les agendas partagés [PSM03] etc, sont des exemples d'applications où l'on peut retrouver ce type d'interactions. Les contraintes liées aux actions des utilisateurs sont par exemple des relations d'antériorité (*Before*), de dépendance (*MustHave*) et de non-commutation [SK05].

Les *collaborations synchrones* correspondent à des actions fortement couplées entre utilisateurs où l'effort se fait en même temps. Les actions peuvent alors subir le même type de contraintes que pour les collaborations autonomes mais la notion de temps physique y est introduite. Le théâtre virtuel [JMS+00, SG00] ou la musique interactive [KOGC98, LBB+03] entrent dans cette catégorie. Notons que dans ce type d'applications, nous pouvons aussi effectuer un découpage sémantique des actions (*Before*, *MustHave* et *non-commutation*) pour optimiser la gestion de la cohérence. Cela va rendre plus complexe la conception de l'application (à cause des spécifications) ainsi que la gestion de la cohérence (il faut vérifier la composabilité de ces spécifications), ce qui sort du cadre de cette thèse.

Dans la suite de cette partie, nous considérerons que nous sommes dans le domaine des collaborations synchrones. Pour cela nous allons utiliser les critères suivants [Bou05b, Bou05a, BGS04], qui s'expriment du point de vue de l'interface utilisateur :

1) L'*instantanéité* : une action est instantanée si le délai entre la production de cette action et la notification de celle-ci à l'interface est inférieur à un délai perceptible par l'utilisateur. Ce critère fait référence aux interfaces à "effet immédiat" qui nécessitent un temps de réponse "rapide".
2) La Δ *légalité* : les délais entre les actions d'un utilisateur doivent être les mêmes lorsque celles-ci sont perçues que lorsqu'elles sont effectuées.
3) La *simultanéité* : deux actions sont dites simultanées si le délai local entre la perception de chacune de ces actions est le même pour tous. Remarquons que si ce délai est suffisamment faible pour être considéré comme nul par l'utilisateur, alors les deux actions ont lieu *en même temps* pour tous les utilisateurs. [28]
4) Les *conflits d'ordonnancements* des notifications des actions entre utilisateurs. Les *conflits d'ordonnancements* comme la causalité sont des propriétés largement étudiées dans les systèmes distribués. Nous reviendrons sur ce point au chapitre 4.3.2.

La *simultanéité* et *l'instantanéité* sont des notions clés dans la conception des AMID. En effet les interactions sont généralement issues de mouvements ou de pratiques de la vie courante (professionnelle ou pas) et rendues possibles grâce à la perception (audition, vue, etc) que nous avons des autres et de l'environnement. En fait, bien que les informations captées par nos sens aient elles aussi un délai de propagation (vitesse de son, de la lumière, etc) nos sens sont limités et nous fournissent l'illusion d'être dans un environnement complètement synchrone (en tout cas avec des personnes proches). Par exemple, dans une pièce meublée nous avons le sentiment que nos propres paroles sont instantanées (bien que le son réverbéré nous revienne aux oreilles avec un délai calculable) tandis que nous percevons clairement le délais de l'écho dans des espaces vastes (comme en montagne). Lorsque l'information circule sur des réseaux informatiques (dans lesquels les délais de transport peuvent atteindre des valeurs suffisamment grandes pour que l'utilisateur les perçoive) ; les AMID doivent reconstruire cette illusion.

En terme de cohérence des données, la *simultanéité* relève de la synchronisation des répliques de l'état du système. En effet cette contrainte ne s'exprime que pour les événements de notification d'un changement d'état à l'interface utilisateur. Cependant l'*instantanéité* peut être comprise comme une contrainte d'échéance sur la délivrance des données. Ainsi le succès du protocole de cohérence va dépendre de l'état et des performances du réseau et de l'ensemble des éléments introduisant de la latence sur la communication (système d'exploitation, périphériques, couches logicielles). Les concepteurs d'une AMID doivent alors faire un compromis entre le degré de cohérence et le temps de réponse [BBA98] de l'application.

Il est donc impératif de quantifier les effets de la latence (ou délai) sur les usages des interactions. Plus particulièrement, lors de la conception d'une AMID, les seuils de perception des utilisateurs doivent être déterminés afin de dimensionner l'application. Par exemple, les articles suivants [PW02a, PW02b] étudient l'impact de la latence réseau sur la technique du "Dead Reckoning" (utilisée dans les jeux répartis multijoueurs). Dans le contexte de la réalité virtuelle, [MRWJ03] établit les effets secondaires du délai sur le sentiment d'immersion. [Sch02] est une étude qui se focalise sur l'interactivité musicale distribuée (du point de vue du musicien) et sans synchronisation. Dans [PK99] l'étude se fait sur des actions fortement couplées (d'un point de vue plus général) avec différents niveaux de difficultés. Finalement, l'étude de Steinmetz [Ste96]

[28] Nous proposerons une simultanéité "système" qui fournit cette simultanéité utilisateur (voir paragraphe 4.3.1)

analyse l'influence des délais sur des désynchronisations comme celles des lèvres avec la parole. Toutes ces études concluent sur le fait qu'à partir d'un certain seuil de latence, pour un usage donné, l'utilisateur devient sérieusement gêné et finit par abandonner l'utilisation de l'application. Ces "seuils perceptifs" seront pris en compte dans l'analyse des modèles de cohérence (paragraphes 4.3.1 et 5.6).

Nous avons vu au chapitre 3 que les ambiguïtés temporelles dues à la latence et à la gigue dans la transmission des informations devaient être compensées pour établir l'interactivité entre utilisateurs d'un système de PMID. Plus généralement, ce problème est un problème de cohérence des données qui se retrouve dans les AMID. C'est pour cela que dans ce chapitre, nous proposons un cadre formel pour la spécification des communications entre AMID et pour l'étude et la description des critères de cohérence.

Dans le paragraphe 4.1, nous proposons un bref historique de la cohérence dans le domaine des Mémoires Réparties Partagées (MRP), ainsi qu'une définition informelle des critères tels que la Δ légalité, l'instantanéité et la simultanéité. Ensuite nous présenterons au paragraphe 4.2 un modèle de description des AMID qui reprend le modèle de description des MRP de [RS96]. Nous y détaillerons les modifications que nous y avons apportées (sur les opérations de lecture et les horloges).

Cela nous permet de décrire formellement au paragraphe 4.3 les contraintes de cohérence mentionnées ci-dessus, tout en montrant leurs caractéristiques d'ordonnancement, à savoir l'incompatibilité à la légalité causale et le respect d'un ordre d'exécution qui évite les conflits liés aux opérations non-commutatives.

4.2 Un modèle pour les AMID

Nous pouvons trouver dans la littérature des MRP une quantité très importante d'articles abordant la cohérence (et plus rarement sa formalisation). Pour les AMID, la tendance est plutôt d'apporter un nouveau modèle par auteur et par protocole. C'est pour cela que dans l'étude qui suit, nous allons reprendre le modèle issu de [RS96] et l'adapter aux AMID. Cela nous permet d'une part de mieux comprendre les liens et les différences entre les deux types de systèmes, mais aussi de voir dans quelle mesure les modèles de cohérence pour les AMID peuvent profiter de l'expérience et la maturité des modèles de cohérences issues du domaine des MRP.

4.2.1 Les opérations d'accès aux données

Nous considérons que l'interactivité et la co-présence sont atteintes dans les AMID grâce aux échanges de données Multimédia, que nous appelons des *instances de média*. Par exemple l'audio est une catégorie de média tandis que les deux flux audio d'une application de téléphonie sont deux instances de média. De même dans une application de réalité virtuelle, la succession des mises à jour des positions d'un utilisateur constitue une instance de média.

Dans les MRP, la valeur d'une donnée n'a d'intérêt que lorsqu'un appel à la fonction *read* est effectué. Cela s'oppose aux AMID, où les utilisateurs partagent des données média, non seulement issues de la numérisation de leurs actions, mais aussi perçues continuellement. De ce point de vue, l'accès aux données n'est pas initié par une demande explicite (opération *read*), mais par une *notification* des mises à jour (qui sera aussi notée comme une opération *handle*). Plus précisément, dans les AMID, la communication est de type "publish/subscribe"[EFGK03]. C'est pour ces raisons que dans notre modèle, l'utilisation de l'opération *handle* est préférable à l'opération *read*.

Le modèle que nous allons présenter supporte à la fois les *médias continus* et les *médias discrets*[29]. En effet, une instance de média est constituée d'Unités Logiques de Données (LDU pour Logical Data Units) [Ste96] qui sont *écrites* (*write*) et *attrapées* (*handled*) par le(s) système(s).

Nous pouvons alors illustrer le comportement "publish/subscribe" dans le domaine des applications à *médias continus* et *discrets* :

- Dans un système de transmission de flux vidéo, l'action de cliquer sur le bouton "play" est un abonnement à la notification régulière d'événements audiovisuels
- Dans la réalité virtuelle, la gestion de la communication est quelquefois basée sur les "zones d'intérêts" : les utilisateurs se situant dans des zones d'intérêt communes doivent être notifiés des opérations d'écriture effectuées par les autres.

Cependant, cette modification des opérations ne fait pas perdre sa généralité au modèle car l'opération *read* peut-être vue comme un cas particulier de l'opération *handle*. En effet :

- L'opération *read* est une simplification de l'opération *handle*. En effet, localement à une réplique, une lecture sur un objet renvoie la valeur retournée par la *dernière* opération *handle*. Sachant que *dernière* est non ambigu car basée sur une vision locale, celle de l'ordre du processus local
- L'opération *Handle* correspond à la mise à disposition d'une valeur considérée comme accessible à l'extérieur du système de cohérence. C'est donc le système qui décide de la date de mise à jour. Notre formalisme est donc compatible avec les systèmes distribués basés sur la notion de temps physique
- L'opération *Handle* convient à la fois aux médias continus et aux médias discrets.

Du point de vue du concepteur d'AMID, la notification d'une nouvelle valeur pour une instance de média va être immédiatement mise à jour au niveau de l'interface utilisateur ; on peut dire alors que la valeur est attrapée (*handled* ou *caught* en anglais) *quand* le système de gestion de la cohérence remonte la mise à jour cohérente. Ici, la date du *quand* sera décidée par le système de gestion de cohérence en fonction de critères d'ordre et/ou de temps physique. De la sorte et afin de simplifier les notations, nous allons considérer que le temps pris par l'application pour mettre à jour la donnée au niveau de l'interface est nul.

4.2.2 Notation : les opérations, les historiques, les horloges

La notation qui est présentée ici est inspirée de la notation présentée dans l'article de M. Raynal et A. Shiper [RS96], qui contient un ensemble de définitions pour la cohérence des Mémoires Réparties Partagées. La principale différence entre les notations est la modification de l'opération de lecture, qui devient l'opération de notification dans les modèles de cohérence pour les AMID. Ainsi, comme les MRP, un système d'AMID est composé d'un ensemble fini de processus séquentiels $P = (p_1, ..., p_n)$ qui interagissent par l'intermédiaire d'un ensemble fini d'objets $X = (x_1, ..., x_m)$. Chaque objet de X est une instance de média qui est accédée par des opérations

[29] Nous supposons que les médias continus sont caractérisés par une fréquence (constante ou variable) d'écriture sur l'instance de média (comme pour l'audio ou pour les mouvements d'un avatar dans un jeu) tandis que pour les médias discrets, les écritures résultent d'une action spontanée de l'utilisateur (écrire du texte dans un tableau blanc partagé)

de type *write* et *handle*.

Les opérations

L'écriture d'une valeur v sur un objet x par le processus p_i est notée $w_i(x)v$ (chaque valeur écrite v est considérée comme unique). De la même façon, l'opération *handle* associée à cette écriture est notée $c_i(x)v$. Afin de simplifier la notation, op sera utilisé pour noter soit w, soit c. Intuitivement, l'écriture d'une valeur correspond à la génération d'une nouvelle valeur pour une instance de média donnée (l'appel n'est pas bloquant car on ne peut pas bloquer l'utilisateur dans sa manipulation des périphériques). De même, l'opération handle est l'opération qui notifie l'utilisateur de la mise à jour d'une donnée, au niveau de l'interface.

L'historique

Précédemment introduit par [RS96], l'historique local $\widehat{h_i}$ de p_i est la séquence d'opérations exécutées par p_i. Si $op1$ et $op2$ sont exécutées par p_i et $op1$ est exécutée la première, alors $op1$ précède $op2$ dans l'ordre du processus p_i. Cela est noté $op1 \rightarrow_i op2$. Soit h_i l'ensemble des opérations exécutées par p_i ; l'historique local $\widehat{h_i}$ est l'ordre total $(h_i, \rightarrow_i)$.

Un historique $\widehat{H}$ d'une AMID est l'ordre partiel $\widehat{H} = (H, \rightarrow_H)$ tel que :
- $H = \bigcup_i h_i$
- $op1 \rightarrow_H op2$ si :
  i) $\exists p_i : op1 \rightarrow_i op2$ (dans ce cas, la relation $\rightarrow_H$ est appelée *process-order*)
  ou ii) $op1 = w_i(x)v \wedge op2 = c_j(x)v$ (*handled-from*)
  ou iii) $\exists op3 : op1 \rightarrow_H op3$ et $op3 \rightarrow_H op2$

Les horloges

Au niveau de chaque réplique, une horloge physique locale permet d'attribuer une date locale aux opérations. Nous savons, d'après les travaux de David Mills sur NTP qu'une synchronisation parfaite et absolue des horloges sur un réseau de type IP est impossible [Mil92] et que les synchronisations les plus précises sont construites au dessus de mécanismes spécifiques comme les horloges IEEE 1588 [IEE04a] ou GPS [IEE04b]. En conséquence, il n'est pas possible, dans un système distribué, d'utiliser une horloge globale pour déduire des relations d'ordre entre les opérations, sans introduire une marge d'erreur.

C'est la raison pour laquelle nous utilisons des horloges physiques locales pour définir les critères de cohérence basés sur le temps physique. Nous supposons alors que les horloges peuvent avoir un décalage absolu (*lag*) mais ne dérivent pas entre elles (*drift*). Dans la mesure où nous utilisons des horloges locales dans nos spécifications, la dérive des horloges n'aura pas d'influence sur les propriétés d'ordonnancement des opérations, mais en aura sur les propriétés temporelles. Nous reviendrons dessus au chapitre 9.

Plus formellement, nous notons t_i l'horloge physique locale du processus p_i. En conséquence, $t_i(w_i(x)v)$ date localement et physiquement l'écriture de la valeur v sur l'objet x. Notons que $t_i(w_j(x)v)$ n'a pas de sens puisque t_i ne peut dater que des opérations locales à p_i. Nous utiliserons donc le plus possible la notation t à la place de t_i, car il n'y a pas d'ambiguïté sur l'horloge utilisée.

Bien que l'on ne puisse avoir d'horloges parfaitement synchronisées (i.e. de façon absolue, sans *lag*), nous considérons qu'il existe virtuellement une horloge universelle qui permettrait de mesurer les délais du réseau. Nous allons utiliser une telle horloge, non pas pour décrire nos critères de cohérence, mais pour discuter de propriétés globales. Bien que cette horloge ne fasse pas partie du système, nous la notons T.

4.3 Les critères de cohérence

Nous allons dans ce paragraphe décrire de façon formelle les critères que nous allons utiliser dans les modèles de cohérence. Ces descriptions sont proposées du point de vue du système, mais en tenant compte des besoins de l'utilisateur.

4.3.1 Les contraintes temporelles

Dans la mesure où la *co-présence* (voir chapitre 1) consiste à donner l'illusion à des utilisateurs d'être physiquement ensemble, les simulacres perceptifs doivent être le plus réaliste possible, notamment pour la représentation temporelle des événements.

En effet, dans notre vie de tous les jours, lorsque nous interagissons avec d'autres personnes, nous possédons chacun notre propre perception de l'écoulement du temps. Dans ce référentiel, nous y percevons nos propres actions, comme un écho de nous même, mais aussi les actions venant d'éléments extérieurs. Ce sont ces caractéristiques que nous essayons de capturer avec les notions d'instantanéité, de Δ légalité et de simultanéité.

L'instantanéité

Les utilisateurs d'applications Multimédia sont habitués à un *effet immédiat* d'une action sur l'application. L'application doit donc avoir un temps de réponse faible. Par exemple, l'effet produit en tournant le volant d'un périphérique à retours d'efforts doit être perçu dans les mains du conducteur et sur le véhicule affiché dans le jeu.

Définition 1 *Une opération handle $c_i(x)v$ est instantanée si et seulement si :*
$t(c_i(x)v) - t(w_i(x)v) \leq \tau_{resp}(x)$

Où τ_{resp} est le délai toléré par un utilisateur entre une action et sa perception. Par exemple, dans les applications audio, en fonction de l'utilisateur, un délai de 5ms à 40ms peut être toléré. Ainsi, un historique $\widehat{H}$ est *instantané* si toutes les opérations *handle* sont instantanées.

La Δ légalité

La Δ légalité diffère de la δ légalité définie dans [TRAR99]. En effet la δ légalité caractérise le fait qu'une donnée doit être lue *avant* un délai $t+\delta$ (δ est une laxité) tandis que la Δ légalité caractérise le fait qu'une écriture doit être lue (en fait *handled*) à Δ unités de temps après l'écriture.

Les délais entre deux écritures locales doivent être répercutés au niveau des opérations *handle* associées (voir la figure 4.1). Par exemple pour des flux de type audio/vidéo, ou encore des positions dans un jeu de voiture (où la vitesse perçue par les autres joueurs doit être la même que

la vitesse effective).

Définition 2 *Un historique $\widehat{H}$ est Δ légal si :*
$\forall x \forall u, v \forall i, j : t_i(w_i(x)v) - t_i(w_i(x)u) = t_j(c_j(x)v) - t_j(c_j(x)u)$

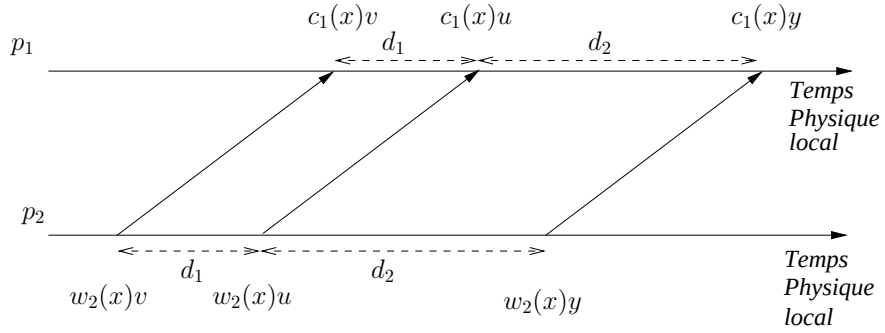


FIG. 4.1 – *La Δ légalité*

Nous pouvons remarquer que cette définition est en fait une contrainte de latence constante entre les écritures et leurs opérations de notifications. En effet si nous supposons que les horloges t_i et t_j sont parfaitement synchronisées, alors elles renvoient une date universelle qui permet de comparer leurs dates respectives. Nous avons alors :
$\forall u, v : t_j(c_j(x)u) - t_i(w_i(x)u) = t_j(c_j(x)v) - t_i(w_i(x)v)$
En d'autres termes, la latence entre p_i et p_j doit rester constante au cours du temps (la gigue doit être rendue nulle). Nous notons $\Delta_{x,i,j}$ une telle constante, qui correspond à une contrainte sur le délai réseau entre p_i et p_j pour l'instance de média x.

C'est principalement cette propriété qui va poser problème aux protocoles de cohérence perceptive, car la latence est variable sur les réseaux IP. Les protocoles de cohérence perceptive seront donc optimistes car ils supposeront que les messages vont arriver avant un certain délai (qui devra être calibré en fonction de l'état du réseau, nous reviendrons sur cet aspect au chapitre 6).

La simultanéité

Remarquons que les définitions de l'instantanéité et de la Δ légalité ne mettent en jeu qu'un écrivain vis-à-vis des autres processus. Cependant, il est nécessaire de caractériser les relations temporelles des écritures entre les différents écrivains, c'est le critère de *simultanéité.*

Parmi les premiers travaux sur l'étude de la perception du temps par un être humain, ceux de A. Einstein [Ein05] en 1905 attirent l'attention sur les décalages temporels dûs à la propagation de l'information (la vitesse de la lumière). Dans cet article, A. Einstein analyse la notion de simultanéité, mais le support de communication du système est la lumière. Plus particulièrement, dans la "partie cinématique", au début de l'article il propose la définition suivante :

> *Si [...] je dis que "le train arrive à 7 heures", cela veut dire : "Le fait que la petite aiguille de ma montre pointe sur 7 heures et l'arrivée du train sont deux événements simultanés".*

Dans le cas des AMID, la simultanéité peut être de la première importance. Par exemple dans un jeu de course automobile, il est important que plusieurs joueurs perçoivent certains événements simultanément, notamment lors du franchissement de la ligne d'arrivée. En effet, l'expression "lorsque la voiture A traverse la ligne d'arrivée, la voiture B est 2 mètres derrière" est équivalente à l'expression "la voiture A gagne la course" et "la voiture B est à 2 mètres de l'arrivée" sont deux événements simultanés. De plus cela est vrai du point de vue de B, mais aussi de celui de A. Remarquons que la lumière comme vecteur de transport des informations génère aussi son lot d'incohérences : deux événements que nous percevons comme simultanés ne le sont pas forcément de tous les points de vue. L'observation des étoiles en est une bonne illustration : nous pouvons percevoir des étoiles mortes depuis des centaines d'années, à cause du délai de propagation de la lumière. En fait, en différents points de l'espace, les simultanéités sont différentes à cause du voyage effectué par la lumière.

Dans la partie suivante de l'analyse de A. Einstein, "De la relativité du temps et de l'espace"[Ein05], il est montré que la simultanéité est propre à chaque référentiel temporel, et non à une définition globale du temps.

> *On ne peut donner de signification* absolue *au concept de simultanéité. Cependant, si deux événements sont perçus comme simultanés par un système d'entités autonomes, alors ces événements ne seront pas simultanés lorsque perçus par un système qui est en mouvement par rapport au premier.*

Lors du transport d'informations sur un réseau informatique, cette règle reste valable puisque l'information ne peut être transportée à une vitesse plus rapide que celle de la lumière. Cependant, nous pouvons essayer de reproduire la simultanéité existante entre entités proches, malgré des distances importantes. Autrement dit, un être humain perçoit deux événements simultanés uniquement dans son propre système temporel et la perception simultanée et partagée par plusieurs humains n'est physiquement possible que pour des personnes proches (dans une même pièce par exemple).

A partir de ce constat, nous pouvons définir formellement le concept de simultanéité, pour des personnes proches :

Définition 3 *Dans un historique $\widehat{H}$, les événements $c_j(x_k)v$ et $c_j(x_l)u$ (n'ayant pas forcément le même écrivain) sont simultanés si (voir figure 4.2) :*
$\forall p : t_j(c_j(x_k)v) - t_j(c_j(x_l)u) = t_p(c_p(x_k)v) - t_p(c_p(x_l)u)$

En fait, cela définit la perception de la simultanéité au cours du temps, et en des lieux différents. En effet la perception de deux événements simultanés $c(x_k)v$ et $c(x_l)u$ par deux personnes proches (p_j et p_p) est un cas spécial de la définition précédente, et est défini comme suit :
$t_j(c_j(x_k)v) \approx t_j(c_j(x_l)u)$ et $t_p(c_p(x_k)v) \approx t_p(c_p(x_l)u)$

Nous pouvons maintenant définir le concept d'historique simultané :

Définition 4 *$\widehat{H}$ est simultané si et seulement si :*
$\forall p \; \forall c_{p_1}, c_{p_2} \in H : c_{p_1}$ *et* c_{p_2} *sont des événements simultanés.*

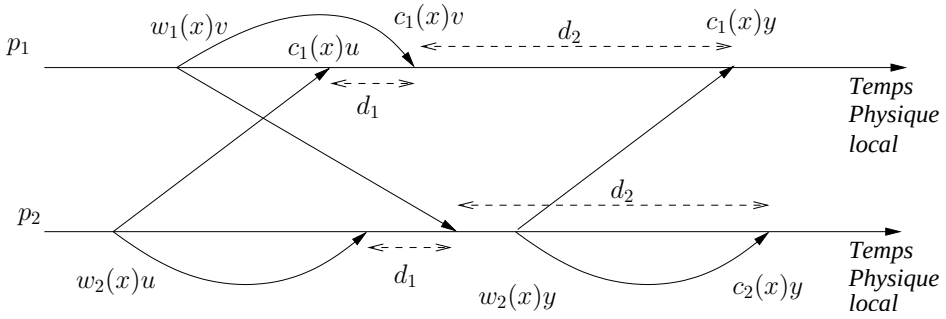


FIG. 4.2 – *La simultanéité*

Discussion au sujet de la simultanéité utilisateur et la simultanéité système

La définition 4 est une vision système. Du point de vue de l'utilisateur, la simultanéité peut être comprise au niveau sémantique et peut être "relâchée" grâce à la limitation de notre perception. Par exemple, nos sens tolèrent jusqu'à 80 milli-secondes de décalage entre un mot et le mouvement de lèvres correspondant [Ste96] (cette valeur est plus faible dans le cas où la vidéo suit l'audio, comme dans les clips par exemple). Ainsi, pour différents types de médias et différentes associations entre eux (par exemple pointeur/voix, texte/image), nous pouvons déterminer la borne à laquelle l'utilisateur va être gêné. Nous appelons ces bornes "**les seuils relatifs de perception**".

Formellement, soit $\tau_{rel}(x_i, x_j)$ le seuil relatif de perception pour deux instances de médias x_i et x_j, alors deux opérations *handle* sémantiquement couplées seront synchronisées du point de vue de p_j si :
$\tau_{rel}(x_i, x_j) \geq |t(c_j(x_i)u) - t(c_j(x_j)v)|$

Ainsi, la définition d'historique simultané prend tout son sens. En effet si un historique $\widehat{H}$ est simultané et qu'il existe un processus p_i qui respecte τ_{rel} pour un couple de médias, alors tous les autres processus respecteront τ_{rel} pour ce couple.

4.3.2 Les critères d'ordonnancement

La légalité causale

La causalité est la relation "happen before" introduite par Lamport [Lam78] pour les protocoles orientés messages et appliquée à la cohérence pour les mémoires partagées par Ahamad et al. [ABHN92]. La spécification fût unifiée par Raynal et Shiper [RS96] comme suit[30] :

Définition 5 *une opération [handle $c(x)v$] est causalement légale si :*
(i) $\exists w(x)v : w(x)v \rightarrow_H c(x)v$ *et (ii)* $\nexists op(x)u : (u \neq v) \wedge (w(x)v \rightarrow_H op(x)u \rightarrow_H c(x)v)$

Implicitement, cette définition suppose que les données sont manipulées par des processus programmés, ayant une logique séquentielle. En d'autres termes, le programmeur suppose que

[30]Nous avons remplacé l'opération d'écriture par une opération de notification, notée c

après chaque instruction, les variables ont été mises à jour. Ce qui explique cette définition : une ancienne valeur ne peut en effacer une plus récente et toutes les dépendances potentielles entre deux opérations successives sur une même donnée sont conservées. Nous pouvons alors considérer que la contrainte de causalité est couramment utilisée dans les MRP car elle exprime la sémantique temporelle d'exécution des mémoires à accès "programmés".

Dans les AMID, les données (instances de média(s)) sont modifiées à l'initiative de l'utilisateur qui évolue dans un environnement naturellement parallèle. Autrement dit, les actions des utilisateurs ne sont pas forcément dépendantes de la dernière opération effectuée, mais de la dernière "perçue". En fait, comme nous allons le montrer au paragraphe 5.3.2 le point (ii) est trop fort pour les AMID. C'est pour cette raison que [ZCTL02] relâche le critère de causalité par le critère de "causalité critique", qui pourrait être comparé au "MustHave" de [SK05]. Ces relations sont des relations exprimant une dépendance sémantique, dont nous ne présentons pas de formalisme ici.

Les conflits et la non-commutation

Malgré le fait que la causalité au sens de Lamport ne soit pas un critère d'ordre pertinent pour les AMID, celles-ci ne sont pas libres de contraintes d'ordonnancement. En effet il est nécessaire d'éviter les conflits causés par des ordonnancements locaux différents parmi les répliques. Cette contrainte est appelée la *non-commutation* dans [SK05] et nous l'appellerons la contrainte de conflit.

Formellement, nous définissons que deux opérations $op1$ et $op2$ sont conflictuelles si $\exists i, j$: $c_i(x)v \rightarrow_i c_i(x)u$ et $c_j(x)u \rightarrow_j c_j(x)v$. Le fait d'éviter ce type de conflit permet aux répliques de prendre des décisions immédiatement et de façon consensuelle sur l'effet qu'une action va avoir sur l'environnement. Par exemple une décision de succès dans un jeu, ou encore la couleur d'un cercle dans une application de tableau blanc.

Le conflit est vu ici dans le même sens que la concurrence dans les bases de données réparties : la cohérence d'une donnée est assurée malgré des accès concurrents et cela en fonction de la sémantique de l'application.

Dans [RS96], la cohérence séquentielle est définie avec le concept de **l'extension linéaire** $\widehat{S}$ de l'historique $\widehat{H}$. Cette définition correspond à un historique qui n'admet pas de conflits.

$\widehat{S}$ est un tri topologique de l'ordre partiel $\widehat{H}$, i.e. :
i) $S = H$
et ii) $op1 \rightarrow_H op2 \Rightarrow op1 \rightarrow_S op2$
et iii) $\rightarrow_S$ définit un ordre total

4.4 Conclusion

Le thème de la cohérence dans les systèmes répartis est vaste : il couvre un nombre important de disciplines de l'informatique. C'est le domaine des Mémoires Réparties Partagées (MRP) qui en a posé les bases formelles, apportant un héritage solide aux problématiques de cohérence dans les applications qui préoccupent la recherche moderne.

Dans ce paragraphe, nous avons proposé d'adapter le formalisme de description des MRP au AMID, ainsi que la description des critères de cohérence. Nous avons pour cela choisi de formaliser le système en tenant compte du temps physique et surtout de la perception des utilisateurs.

En effet, les AMID maintiennent des états locaux qui doivent être globalement cohérents, mais selon le jugement d'utilisateurs humains, contrairement aux MRP qui doivent rester globalement cohérentes vis-à-vis de l'exécution réparti d'un programme parallélisé.

Ainsi, ce paragraphe propose un ensemble de bases formelles qui exprime les contraintes que le système doit satisfaire pour construire un simulacre représentatif : celui de la perception temporelle partagé, telle qu'elle est décrite pour des personnes proches par A. Einstein en 1905 [Ein05]. Ces critères sont la Δ légalité, l'instantanéité et la simultanéité et les conflits.

Grâce à ces critères, nous allons dans le chapitre suivant proposer une spécification de modèles de cohérence des données pour les AMID. Nous pourrons alors estimer dans quelles mesures ces critères sont compatibles et en fonction des caractéristiques du réseau supportant la communication. De plus, l'utilisation d'un modèle d'application proche de celui des MRP nous permet de comparer les modèles "classiques" de la littérature des MRP avec les nouveaux modèles proposés et étudiés.

Cette approche formelle permet de mettre en valeur précisément les propriétés attendues par un protocole de cohérence. Nous proposerons ainsi deux protocoles de cohérence au chapitre 6.

5

Les modèles de cohérence pour les Applications Multimédia Interactives et Distribuées

Sommaire

5.1 Introduction

Dans ce paragraphe, nous allons introduire les modèles uniformes. Cependant, nous allons ici proposer une unification de la formalisation des modèles basés sur des critères de temps physique.

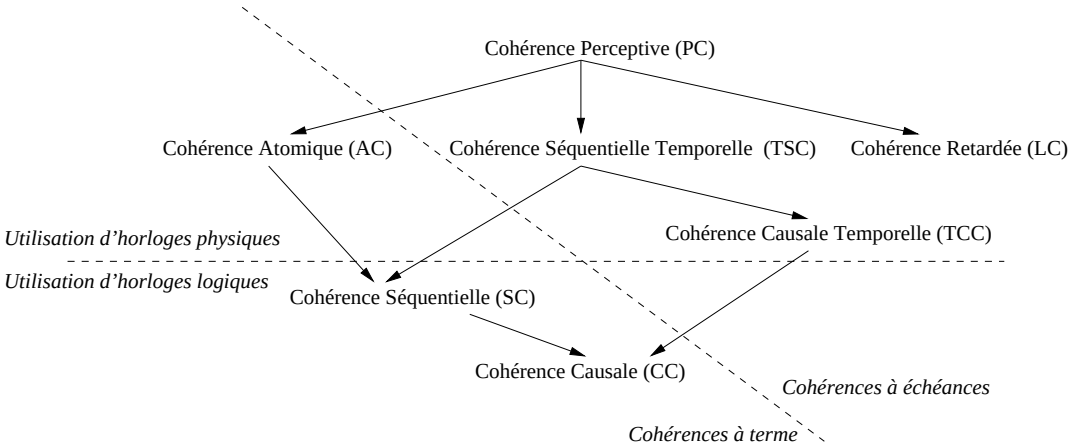


FIG. 5.1 – *Une classification des modèles de cohérences pour les AMID*

La figure 5.1 constitue une classification des modèles et nous donne un aperçu des relations entre eux. Une flèche du modèle A vers le modèle B indique que A possède des critères temporels plus contraignants que B. Dans cette figure, nous distinguons les modèles dont les critères sont basés sur le temps physique et ceux basés sur le temps logique. Pour les modèles "temps physique", il est considéré que les horloges physiques sont synchronisées et/ou ne dérivent pas les unes par rapport aux autres. Quand aux horloges logiques, elles sont incrémentées de façon monotone en fonction de l'apparition des événements.

Nous distinguons aussi les modèles de cohérence "à terme" avec ceux "à échéance". Dans les modèles à terme, les critères sont basés sur l'ordonnancement des opérations sans considération particulière sur la date physique à laquelle une opération sur une donnée est considérée comme cohérente (contrairement aux modèles à échéance).

Dans le but de simplifier la lecture de ce chapitre, le tableau 5.2 expose un résumé des critères associés à chaque modèle de cohérence que nous introduisons ici. Les cases cochées avec des parenthèses montrent que le critère ne fait pas partie de la spécification, mais qu'il est démontré dans ce chapitre s'il y est compatible ou pas[31].

Avant de présenter les descriptions formelles des modèles "à échéance", précisons que le terme échéance ne prend pas le même sens pour les différents modèles. Par exemple, les modèles de Cohérence Séquentielle Temporelle (TSC) et de Cohérence Causale Temporelle (TCC) considèrent que les données "doivent être lues à temps" [TRAR99] (ce critère est appelé la δ légalité par [SRH97, BPRS96]). Cette notion nécessite que si une opération est produite au temps t alors elle doit être visible (et cohérente) *pas plus tard que la date* $t + \delta$. La cohérence perceptive (PC) et

[31] A l'exception du critère d'instantanéité pour la cohérence perceptive, car sa compatibilité dépend d'un facteur extérieur. En effet, ce critère peut être satisfait par un protocole de cohérence perceptive à condition que la latence réseau soit suffisamment faible : cela est démontré au paragraphe 5.3.1.

Modèles \ Critères	Simultanéité	Δ légalité	δ légalité	Instantanéité	Extension linéaire, conflits	Causalité
CC						X
SC					X	X
AC					X	X
TCC	( )		X			X
TSC	( )		X		X	X
LC	( )	X		X	( )	( )
PC	X	X		(X)	(X)	( )

FIG. 5.2 – *Résumé des critères de cohérence de chaque modèle*

la cohérence retardée (LC) sont différentes car si une opération s'effectue à la date t, alors elle doit être visible *à la date* $t + \Delta$ (critère de Δ légalité).

Ces deux approches permettent de maîtriser la dimension temporelle de l'échange de données. En effet, si la valeur peut être rendue visible avec un δ ou un Δ inférieur au seuil de perception (cf. paragraphe 4.3.1) du média concerné, alors l'instantanéité sera atteinte. Dans le cas contraire, la valeur de δ ou de Δ permet de déterminer le temps de réponse du mécanisme de cohérence. Cependant, il ne faut pas oublier que ces contraintes temporelles seront respectées en fonction de l'état du réseau.

Dans ce chapitre, après avoir proposé au paragraphe 5.1 une classification informelle de différents modèles de cohérences, nous présentons de façon formelle les principaux modèles de cohérences issus de la littérature des MRP au paragraphe 5.2 (cohérences séquentielle, causale et atomique). Nous y expliquons en quoi ces modèles ne sont pas adaptés aux AMID, mais aussi en quoi ils fournissent une expertise nécessaire à la construction de modèles pour les AMID.

Nous présentons aux paragraphes 5.3 et 5.4 les modèles de cohérence perceptive (PC) et retardé (LC), qui sont basés sur les critères définis formellement au chapitre 4. Nous y présentons une étude sur la compatibilité de ces critères entre eux, ainsi qu'avec certains critères d'ordonnancements comme la légalité causale et les conflits liés à la non-commutation des opérations.

Finalement, nous étudions les modèles de cohérences temporelles causale et séquentielle au paragraphe 5.5 vis-à-vis de la causalité, de l'instantanéité et de la simultanéité. Cela nous permet de montrer dans quelle mesure le critère de légalité causale n'est pas adapté pour les AMID.

5.2 Les modèles "à terme" et les AMID

Les modèles que nous qualifions de "à terme" correspondent aux mêmes modèles que ceux appelés "eventual consistency" dans [PSYC03, GA02] où par "à terme" il faut comprendre "après un délai non prédéterminé". Ce délai dépend évidemment du protocole qui met en œuvre la cohérence. Les modèles de cohérences causale (CC), séquentielle (SC), atomique (AC) et les modèles plus faibles (comme ceux décrits dans [RS96]) tombent dans cette catégorie. Bien que par définition, le temps de réponse de ces modèles ne puisse être déterminé, ils représentent une base historique solide en ce qui concerne les critères d'ordonnancement des opérations sur les données. Rappelons leurs définitions [RS96] :

- La *cohérence causale* est respectée si l'historique $\widehat{H}$ est causalement légal
- La *cohérence séquentielle* est respectée si l'historique $\widehat{H}$ est causalement légal et s'il admet une extension linéaire

– La *cohérence atomique* est respectée si l'historique $\widehat{H}$ est causalement légal et s'il admet une extension linéaire telle que les relations d'ordre sont exprimées à partir de date physique d'horloges synchronisées.

Nous pouvons trouver dans la littérature des évaluations de protocoles de cohérences à terme dans des AMID. Par exemple, la cohérence séquentielle (qui sera notée SC par la suite) fut expérimentée par Strom et al. et par Li et al. dans [SBM+97, LZM00] ("chat" et tableaux blancs), par Lui et al. [LST99] (vidéo-conférence, jeux en réseau et simulations distribuées interactives) et par Galli et al. [GL00] (conception de plans architecturaux).

Dans ces expériences, le temps de réponse est géré en affichant instantanément les actions locales au niveau de l'interface, laissant diverger les répliques de l'état du système. Dans chacune de ces expériences, le défit est de faire converger l'état affiché avec l'état séquentiellement cohérent. Pour cela, [SBM+97] va afficher deux fenêtres à chaque utilisateur : une sur laquelle l'utilisateur va interagir et où le temps de réponse est instantané (la vue optimiste) ; et une maintenue avec un protocole de cohérence séquentielle (où les actions locales sont affichées avec le retard nécessaire pour assurer la cohérence). Pour les jeux vidéo, Lui et al. [LST99] utilisent la technique du Dead Reckoning (DR) pour l'état affiché et un état séquentiellement cohérent pour faire converger les états affichés. Nous reviendrons plus en détail sur la technique du Dead Reckoning au chapitre 6.3.2.

D'autres AMID utilisent des protocoles de cohérence causale pour ordonner les opérations (bien que ce choix n'évite pas les conflits). Par exemple Li et al. [LZM00], Sun et Chen [SC00, CS99] l'expérimentent pour une application de tableau blanc et Robert et al. [RS97] pour la réalité virtuelle. Ces solutions nécessitent un système de résolution de conflits additionnels qui tient compte de "l'intention de l'utilisateur"[32]. Pour cela, les auteurs de [SC00] utilisent un système de gestion de versions et ceux de [CS99] la convergence[33]. Remarquons qu'il existe d'autres approches pour éviter à la fois les conflits et conserver la causalité, comme le verrouillage des données. Par exemple Constantini et al. [CST+00] l'utilisent pour le prototypage virtuel d'avion en mode coopératif.

La cohérence atomique (AC) [RS96, Mos93] est un modèle en marge des deux modèles présentés précédemment (SC et CC) car il est fondé sur des horloges physiques synchronisées (avec un décalage plus faible que la date pouvant séparer deux événements successifs). Cela permet d'établir un ordre sur les opérations d'écriture. Les valeurs écrites doivent être lues dans le même ordre que l'ordre déterminé à partir des dates des écritures. Dans AC, les lectures vont s'effectuer dans le même ordre sur toutes les répliques. A notre connaissance, AC n'a jamais été utilisé pour assurer la cohérence dans une AMID.

L'utilisation de SC et CC montre plus ou moins de succès sur l'ordonnancement des opérations dans les AMID, en fonction de l'approche choisie. Cependant, aucun de ces modèles n'est capable de fournir la contrainte de simultanéité car aucun d'entre eux n'utilise le temps physique

[32] Cette intention doit être comprise comme le fait qu'un maximum d'opérations effectuées par l'utilisateur doit être conservé dans l'historique. En effet le système de résolution peut être amené à annuler certaines opérations pour assurer la cohérence

[33] La convergence autorise la lecture de données qui ne sont pas considérées comme cohérentes, mais celles-ci finiront par le devenir, grâce au système de convergence

pour dater les accès en lecture aux données (les notifications). De plus les protocoles utilisés dans [SBM+97, LZM00, LST99, GL00] ont été modifiés pour conserver un temps de réponse raisonnable, ce qui viole le critère de causalité (cf définition au paragraphe 4.3.2). Ces expérimentations utilisent alors des solutions additionnelles. Dans ce type de solution, les incohérences d'ordonnancement sont repérées tardivement et réparées en conséquence, laissant potentiellement l'utilisateur percevoir au niveau de l'interface des changements inattendus provoqués par le système.

5.3 Le modèle de cohérence perceptive (PC)

Dans ses premières définitions formelles (utilisant une horloge physique globale), le modèle PC fut spécifié avec une horloge universelle et appelée "cohérence absolue"[Qin02] et "local lag" [VM01]. Cependant, de par notre définition de la simultanéité du paragraphe 4.3.1, nous considérons que cette appellation ne convient pas.

Définition 6 *Un historique $\widehat{H}$ est perceptif cohérent si et seulement si il est simultané et Δ légal*

Ce modèle est sémantiquement proche du modèle "What You See Is What I See" (WYSIWIS) de [SBF+87]. Cependant, dans les WYSIWIS la perception de l'environnement virtuel n'est pas personnalisée selon des critères locaux (comme la position du joueur dans le jeu) mais strictement la même. La cohérence perceptive exprime un principe assez proche, mais s'applique à la gestion de l'état de l'environnement, alors que le modèle WYSIWIS s'applique à l'affichage directement.

5.3.1 PC et l'instantanéité

Nous allons maintenant analyser l'effet de la cohérence perceptive sur la notion d'instantanéité, et montrer que dans un historique conforme à PC, l'instantanéité est fonction des latences du réseau entre les répliques. Supposons deux écrivains p_i et p_j sur la même instance de média x, alors de la Δ légalité[34] nous avons (T est l'horloge universelle) :
(i) $T(c_j(x)v) = T(w_i(x)v) + \Delta_{x,i,j}$ et $T(c_i(x)v) = T(w_i(x)v) + \Delta_{x,i,i}$
(ii) $T(c_i(x)u) = T(w_j(x)u) + \Delta_{x,j,i}$ et $T(c_j(x)u) = T(w_j(x)u) + \Delta_{x,j,j}$
De la simultanéité, nous avons :
(iii) $T(c_j(x)v) - T(c_j(x)u) = T(c_i(x)v) - T(c_i(x)u)$
De (i) et (ii) dans (iii), nous avons :
$T(w_i(x)v) + \Delta_{x,i,j} - T(w_j(x)u) - \Delta_{x,j,j} = T(w_i(x)v) + \Delta_{x,i,i} - T(w_j(x)u) - \Delta_{x,j,i}$
alors $\Delta_{x,i,i} + \Delta_{x,j,j} = \Delta_{x,i,j} + \Delta_{x,j,i}$.

Il est donc préférable que les latences locales $\Delta_{x,i,i}$ et $\Delta_{x,j,j}$ soient choisies un peu plus grande que les latences du réseau entre p_i et p_j. Sinon, la plupart des opérations *handle* risque d'arriver tardivement, et donc d'être illégale. En d'autres termes, le délai $\Delta_{x,i,i} + \Delta_{x,j,j}$ doit être plus grand que le délai aller-retour entre p_i et p_j.

Cela rend l'instantanéité avec la simultanéité possible uniquement lorsque les latences sont telles que :

[34] D'après la définition de la Δ légalité du paragraphe 4.3.1, $\Delta_{x,i,j}$ est la latence maintenue constante entre les écritures effectuées par p_i et les lectures effectuées par p_j pour l'instance de média x

$\Delta_{x,i,j} \leq \tau_{resp}(x)$ et $\Delta_{x,j,i} \leq \tau_{resp}(x)$

5.3.2 PC et la causalité

La figure 5.3 nous montre que la causalité n'est pas respectée, en effet du point de vue du processus p_1 :
$w_i(x)v \rightarrow_H c_i(x)u \rightarrow_H c_i(x)v$

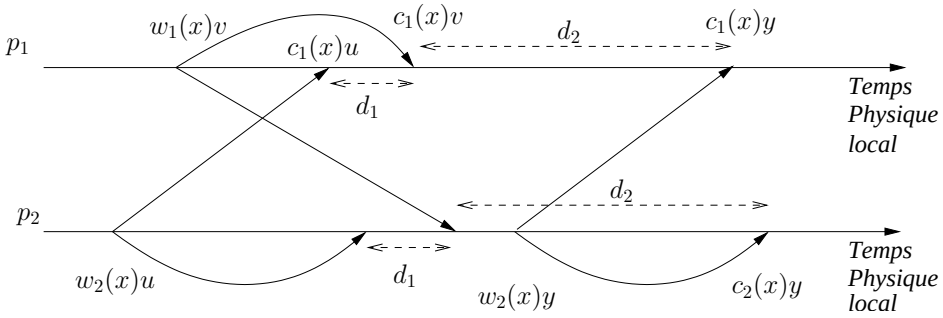


FIG. 5.3 – *Un historique PC cohérent*

Nous pouvons en déduire que le critère de causalité, s'il est couplé avec celui de simultanéité va avoir un impact sur le délai possible entre deux opérations qui s'effectuent sur la même instance de média. En effet il faut attendre la mise à jour locale provoquée par l'opération d'écriture, i.e. il ne faut avoir aucune opération sur x entre $w_1(x)v$ et $c_1(x)v$ (voir la figure 5.3). De plus, nous avons montré précédemment (paragraphe 5.3.1) que pour la simultanéité, le délai local à introduire pour une opération locale ($\delta_{x,i,i}$) dépendait de la latence entre l'écrivain et son lecteur ayant la plus grande latence.

La conséquence directe est la limitation de la fréquence possible d'apparitions des mises à jour de l'instance de média (la période entre deux mises à jour est limitée par la latence réseau). Cela rend le critère de causalité inefficace pour les applications à médias continus "verbeux".

5.3.3 PC et les conflits

Si un historique $\widehat{H}$ est perceptif cohérent, alors il n'admet pas de conflit. En effet, comme nous utilisons des horloges physiques locales, nous pouvons transformer la relation d'antériorité temporelle en relation d'ordre, i.e.
$t_j(c_j(x)v) < t_j(c_j(x)u) \Leftrightarrow c_j(x)v \rightarrow_j c_j(x)u$
Donc, en appliquant cela à la définition de la simultanéité, si un historique $\widehat{H}$ est simultané alors :
$\forall c_j(x)v, c_j(x)u \in H \ \forall k : (c_j(x)v \rightarrow_j c_j(x)u) \Leftrightarrow (c_k(x)v \rightarrow_k c_k(x)u)$

Ainsi, dans un historique $\widehat{H}$ simultané les processus exécutent les mises à jour dans le même ordre, évitant tous conflits tels qu'ils sont définis au paragraphe 4.3.2.

5.3.4 Les protocoles

Nous pouvons trouver des protocoles assurant la cohérence perceptive dans [Bou03, SYG03, Mau00, VM01, GD98, AY96].

Dans [VM01, GD98], les protocoles utilisent une horloge universelle afin de déterminer une latence fixe et identique entre tous les participants, et pour toutes les instances de média ($\forall j$: $T(c_j(x)v) = d + T(w(x)v)$). Pour des jeux en réseau, les deux travaux définissent une latence constante de respectivement $d = 150ms$ et $d = 100ms$.

Ces deux protocoles fournissent bien la cohérence perceptive mais ne tiennent pas compte de l'état réel du réseau, qui peut être sujet à des latences supérieures à d. Dans ce cas, chaque mise à jour arrivera tardivement et sera considérée comme incohérente.

Dans [SYG03, AY96], c'est grâce à une estimation du délai aller retour (RTT) que le système décide de la date à laquelle une valeur doit prendre effet. Pour cela l'écrivain estime son délai avec le(s) lecteur(s) et estime la date physique à laquelle la mise à jour doit avoir lieu. Le protocole de Akyildiz et al. [AY96] est complètement distribué tandis que celui de Saddik et al. [SYG03] est centralisé.

Le principal défaut de ces protocoles est l'utilisation d'une horloge universelle pour obtenir la propriété de simultanéité. L'erreur de synchronisation des horloges se répercute directement dans le protocole : soit par des erreurs temporelles (ou de dimensionnement des délais), soit par des conflits.

Finalement, [Bou03] propose le protocole le plus fidèle à la cohérence perceptive car il utilise les horloges locales des processus. Il effectue un "consensus" sur un vecteur de dates locales d'opérations *handle* qui doivent être jouées simultanément, et en fonction des délais réseau. Ce protocole fera l'objet du chapitre 6.

Compte tenu du fait que la réservation de ressources n'est pas toujours disponible sur l'Internet, les latences sont variables. Nous pouvons alors nous poser la question suivante : que faire si la donnée est perdue ou arrive plus tard que son échéance ? Le plus simple est de ne rien faire, mais cela n'est possible que si le média permet la perte ponctuelle de quelques échantillons. Les applications telles que le concert réparti [Bou03] et MiMaze, un jeu multijoueurs [GD98] entrent dans cette catégorie. La seconde approche consiste à rendre fiable la transmission des opérations importantes et d'effectuer une résolution de conflit si une mise à jour tardive en provoque un. Mauve propose une telle solution dans [MVHE04] en utilisant le mécanisme Timewarp [EM97]. Nous reviendrons sur ce type de technique, notamment celle des états de remorquage (TSS [CFKJ02]) au paragraphe 6.3.2.

5.4 Le modèle de cohérence retardée

La cohérence retardée (LC) fut formellement décrite par Qin dans [Qin02] (avec une horloge physique globale). C'est un relâchement de PC où les actions locales ont légalement le droit d'être mises à jour avant le délai nécessaire à assurer la cohérence perceptive. Cela a l'avantage d'augmenter la réactivité des protocoles de cohérence mais le désavantage de perdre les propriétés d'ordonnancement de la cohérence perceptive.

Définition 7 *Un historique $\widehat{H}$ est* ***tardivement cohérent*** *si*

i) $\forall i \forall x : t(c_i(x)v) = t(w_i(x)v)$ *et* $\nexists op : w_i(x)v \rightarrow_{h_i} op \rightarrow_{h_i} c_i(x)v$ *(les écritures locales sont instantanées)*
et ii) $\widehat{H}$ *est* Δ *légal*
et iii) La simultanéité est atteinte, excepté pour l'écrivain :$\forall k, l \forall u, v \forall p : w_i(x_k)v, w_i(x_l)u \notin h_i \Rightarrow (i \neq p \wedge t(c_i(x_k)v) - t(c_i(x_l)u) = t(c_p(x_k)v) - t(c_p(x_l)u))$

Dans cette définition, la formule $t(c_i(x)v) = t(w_i(x)v) + 1$ exprime le fait que la notification est immédiatement prise en compte. Le grain de l'horloge est alors suffisamment fin pour que $c_i(x)v$ et $w_i(x)v$ soient perçus en même temps par l'utilisateur.

Avec cette définition d'historique **tardivement cohérent**, nous pouvons remarquer que la simultanéité devra être construite à partir du moment où au moins quatre processus sont à la fois écrivains et lecteurs. En effet, si seulement trois communiquent alors tout couple de notifications ayant des écrivains différents laisse le troisième écrivain assurer la simultanéité seul. Dans ce cas, le seul critère de cohérence de LC est la Δ légalité telle que $\forall i : \delta_{x,i,i} = 0$.

5.4.1 LC et les conflits

La figure 5.4 nous montre un historique LC cohérent qui admet un conflit. En effet nous avons à la fois $c_1(x)v \rightarrow_H c_1(x)u$ et $c_2(x)u \rightarrow_H c_2(x)v$.

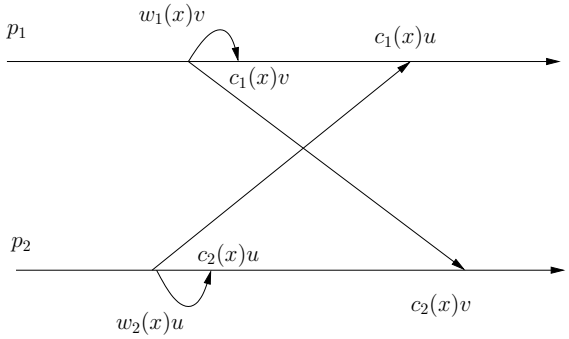


FIG. 5.4 – *Les conflits et la cohérence tardive*

5.4.2 LC et la causalité

La figure 5.5 nous montre un historique tardivement cohérent dans lequel la causalité n'est pas respectée. En effet, dans cet historique nous avons $w(x)v \rightarrow_H c(x)v \rightarrow_H w(x)u \rightarrow_H c(x)u \rightarrow_H c(x)v$.

5.4.3 Les protocoles

La cohérence tardive est un modèle qui privilégie le temps de réponse local dans la gestion de l'état de chaque réplique. A notre connaissance, un seul protocole met en œuvre un modèle de cohérence très proche de la cohérence tardive, le protocole proposé dans [Bou04] (ce protocole sera développé plus en détail dans le chapitre 6).

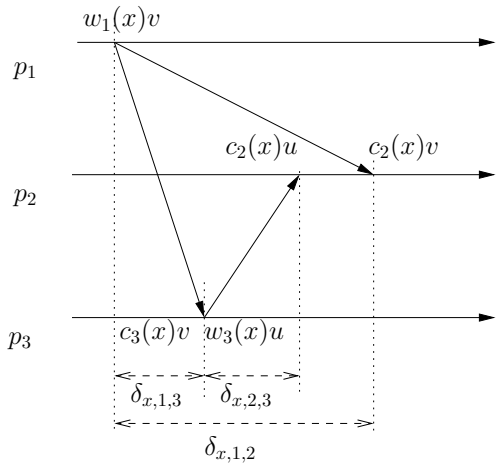


FIG. 5.5 – *Un historique LC cohérent mais pas causal*

LC ne fournit ni la simultanéité ni la garantie qu'il n'y aura pas de conflits. Il est en fait le fruit de compromis entre le temps de réponse désiré et les autres critères. Il correspond à la gestion temporelle du Dead Reckoning (première version, voir paragraphe 6.3.2). Ce modèle, bien que proche du Dead Reckoning deuxième version n'en est pas tout à fait équivalent puisque ce dernier "anticipe" les actions.

Beaucoup de jeux multijoueurs utilisent des protocoles de cohérence correspondant au modèle de cohérence retardé. Bien que celui-ci ne soit pas tout à fait "efficace" en terme d'ordonnancement et de simultanéité, il est le représentant même du besoin de modèles optimistes où l'instantanéité est le critère primordial pour la convivialité du système.

5.5 Les modèles de cohérences temporelles causale et séquentielle

La Cohérence Séquentielle Temporelle (TSC) et la Cohérence Causale Temporelle (TCC) ont été formalisées pour la première fois dans [TRAR99] et mises en œuvre notamment en informatique industrielle pour l'échange de données entre automates sur le bus de terrain FIP [GS02]. Dans ces modèles, les lectures (pour nous les notifications) sont considérées légales si l'historique $\widehat{H}$ est δ légal et s'il respecte les critères d'ordre ; respectivement la causalité et l'ordre séquentiel.

La δ légalité est un critère temporel basé sur le concept des échéances : une lecture doit retourner la valeur la plus récente, sans tenir compte des nouvelles écritures possibles depuis δ. Si l'on considère alors que l'accès en lecture aux données se fait par des notifications, alors le critère devient comme suit : "une notification sur une instance de média doit avoir lieu au plus à δ unités de temps avant la prochaine écriture".

Définition 8 *Une notification $c_j(x)v$ est δ* ***légale*** *si :*
i) $\exists w_i(x)v : T(c_j(x)v) - \delta_x > T(w_i(x)v)$
et ii) $T(w_i(x)v) < T(w_i(x)u) \Rightarrow T(c_j(x)v) - T(w_i(x)u) \leq \delta_x$
Un historique $\widehat{H}$ est δ légal si toutes ses opérations de notifications sont δ légales.

5.5.1 La δ légalité, la causalité et l'instantanéité

La causalité a en fait un impact important sur le délai minimum entre deux écritures successives. Remarquons tout d'abord que dans la figure 5.6, d'après l'ordre du processus p_1, nous avons $w_1(x)v \rightarrow_H c_1(x)u \rightarrow_H c_1(x)v$. L'historique n'est donc pas causalement légal (donc ni TSC ni TCC).

Nous en déduisons donc que si une écriture $w_i(x)v$ s'effectue entre $t(c_i(x)u)$ et $t(c_i(x)u) - \delta_x$:
$t(c_i(x)u) - \delta_x < t(w_i(x)v)$ et $t(w_i(x)v) < t(c_i(x)u)$
$\Rightarrow t(c_i(x)u < t(w_i(x)v) + \delta_x$ (et $t(w_i(x)v) < t(c_i(x)u)$)
donc, de la définition de la δ légalité, nous avons :
$t(c_i(x)u) < t(c_i(x)v)$ et $t(w_i(x)v) < t(c_i(x)u)$
donc $\widehat{H}$ n'est pas causalement légal.

Cela montre que la δ légalité se combine avec le critère de causalité de la même façon que la simultanéité, et que le délai entre deux écritures doit être supérieur au délai réseau δ_x. La figure 5.7 illustre cela en nous donnant un exemple d'historique combinant la δ légalité et la causalité.

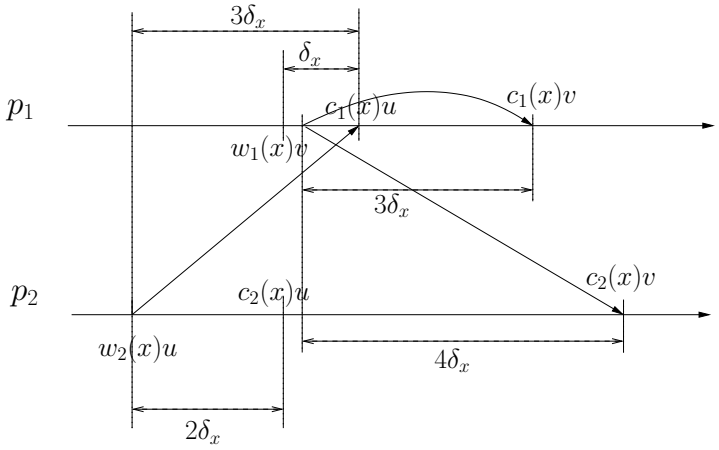


FIG. 5.6 – *Un historique δ légal*

De plus l'utilisation de TSC et de TCC pour des AMID peut avoir des effets inattendus, en particulier sur le temps de réponse et la gestion mémoire. Par exemple :

- La fréquence maximale tolérée pour une AMID sera alors celle correspondant à une période égale δ. Cette contrainte risque de mettre en échec les médias comme l'audio, la vidéo ou la numérisation des mouvements
- L'utilisateur ne peut générer plusieurs écritures en un temps inférieur à δ. Il faudrait alors bloquer l'interface.

5.5.2 La δ légalité et la simultanéité

La figure 5.7 respecte à la fois TSC et TCC mais la simultanéité n'y est pas respectée, en effet :
$t(c_1(x)v) - t(c_1(x)u) = d + 3\delta_x$ et $t(c_2(x)v) - t(c_2(x)u) = \delta_x + d + 4\delta_x$
$\Rightarrow t(c_1(x)v) - t(c_1(x)u) \neq t(c_2(x)v) - t(c_2(x)u)$
Ce qui contredit la définition de la simultanéité pour les opérations $c(x)v$ et $c(x)u$.

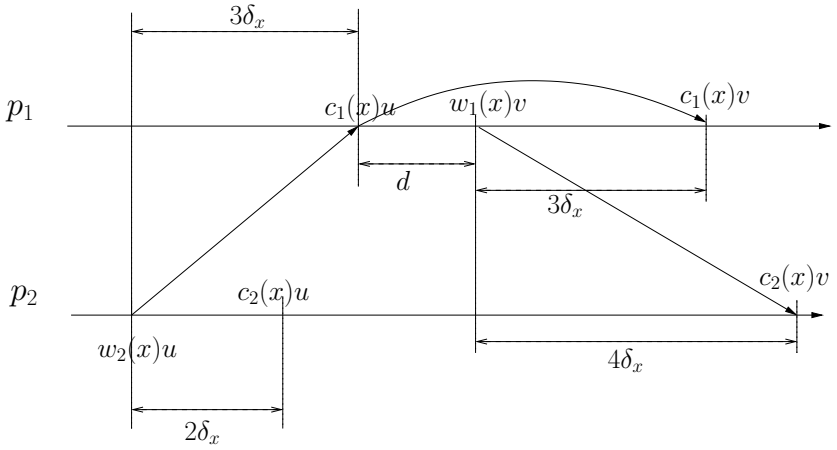


FIG. 5.7 – *Un historique TSC et TCC*

5.6 Conclusion

Les Applications Multimédia Interactives et Distribuées (AMID) sont naturellement soumises à des contraintes temporelles fortes. Dans ce chapitre, nous avons proposé pour ce type d'applications une suite de spécifications formelles de modèles de cohérences fondés sur la perception temporelle des événements par les utilisateurs. Nous avons fourni une analyse de ces modèles en réutilisant les historiques de [RS96] et en y modifiant la sémantique de l'opération de lecture, nous avons montré que la causalité est un critère contraignant pour les AMID, notamment sur la fréquence d'apparition des mises à jour. Par la proposition de propriétés temporelles orientées utilisateur (la simultanéité, l'instantanéité et la Δ légalité) que nous avons transformés en critère d'ordonnancement. Cela nous a permis de montrer que le modèle de cohérence le plus contraignant (la cohérence perceptive) fournit à la fois l'ordre total des opérations (évitant les conflits) et le respect des contraintes temporelles (la Δ légalité et la simultanéité). Nous avons aussi montré que le modèle de cohérence perceptive est le seul à fournir la simultanéité. C'est donc ce modèle que nous allons retenir pour les AMID proposant une interactivité fortement couplée entre utilisateurs, comme c'est le cas pour certain jeux vidéo et dans le domaine des PMID. Nous proposerons aussi au paragraphe 6.4 un protocole de cohérence retardée adapté au système de PMID.

Comme nous l'avons vu dans les paragraphes précédents, les AMID sont soumises à des critères temporels forts, bien qu'elles soient supposées être déployées sur des réseaux asynchrones. De ce fait, durant la conception de l'application, il est nécessaire d'évaluer la tolérance aux délais de la part des utilisateurs, dans le but de pouvoir appliquer une stratégie pertinente de gestion de la cohérence et des délais. On peut trouver dans la littérature quelques études à ce sujet (bien que très spécifiques à un mécanisme donné). Citons par exemple [PW02a, PW02b] où Pantel et Wolf étudient l'impact du Dead Reckoning retardé sur la jouabilité d'un jeu multijoueurs. [MRWJ03] est une étude sur les effets secondaires des délais locaux sur le sentiment d'immersion et [Sch02] sur la perception musicale de musiciens physiquement distants qui s'entendent réciproquement.

En terme de convivialité, les protocoles de cohérence des AMID doivent absolument tenir compte des limitations perceptives des utilisateurs et des limitations du réseau. Nous distinguons

trois approches pour effectuer le dimensionnement des variables liées à la gestion des délais :

- La conception de l'AMID ne prend en compte que les seuils utilisateurs, i.e. la Δ légalité est assurée avec les seuils perceptifs. Le principal défaut est que tout utilisateur se situant sur le réseau à une distance temporelle plus grande qu'au moins un des seuils va produire une quantité importante d'opérations incohérentes. C'est le problème dont souffre les protocoles tels que [GD98, MVHE04, Mau00]. Le protocole adapté au domaine des PMID que nous présenterons au paragraphe 6.4 peut être classé dans cette catégorie car l'interactivité entre musiciens s'effectue à la fois instantanément et simultanément : seul le chef d'orchestre perçoit un retard.
- La conception de l'AMID s'effectue en fonction des délais réseaux, i.e. la Δ légalité est assurée avec des mesures réseau durant l'exécution. Dans ce cas, la propriété de simultanéité ne pourra être obtenue pour chaque opération que après un délai équivalent au délai le plus important entre chaque couple d'utilisateur (voir paragraphe 5.3.1). Les protocoles décrits dans [Bou03, SYG03, AY96] et au paragraphe 6.2 utilisent cette approche.
- L'exécution de l'AMID introduit un contrôle d'accès des utilisateurs, en fonction de la latence mesurée vers les autres. Une latence réseau supérieure au plus petit seuil perceptif entraînera le refus d'accès de l'utilisateur au système. Cette approche est couramment utilisée dans les jeux en ligne en utilisant le système de la chambre d'accès (venue room). Cette approche "pragmatique" n'enlève rien au besoin de cohérence entre les utilisateurs élus.

Cela nous amène à penser que dans une certaine mesure, le réseau Internet possède des limites, non pas en terme d'accessibilité (elle est suffisamment bien gérée par les protocoles de routage), mais en terme de distances temporelles qui provoquent des périmètres "d'accessibilité interactive".

6

Les protocoles

Sommaire

6.1 Introduction

Comme nous l'avons vu dans les chapitres précédents, les sentiments de co-présence et d'immersion sont des éléments clés dans la conception d'une AMID. Cependant, chaque application soulève ses propres nuances. A partir des propriétés temporelles que nous avons formalisées au chapitre 4, nous pouvons spécifier les besoins temporels de plusieurs types d'Applications Multimédia Interactives et Distribuées.

Nous avons retenu dans le chapitre précédent le modèle de cohérence perceptive, car c'est le seul à fournir la propriété de *simultanéité*. Cette propriété est nécessaire dans un certain nombre d'AMID. Pour le concert réparti par exemple, la *simultanéité* permet d'assurer aux musiciens que les différents retours audio des musiciens soient identiques. Cette propriété est extrêmement importante car elle évite alors les ambiguïtés temporelles décrites au paragraphe 3.3.2. En fait, cette propriété permet aux musiciens de jouer "simultanément", durant l'exécution du morceau. Nous verrons au chapitre 7 que l'un des protocoles décrit dans ce chapitre permet à la fois de contrôler le délai local à introduire et de construire un métronome global. Après plusieurs expérimentations (présentées au chapitre 8), nous constatons que l'interactivité musicale est possible avec des retards locaux[35]. Ainsi, pour le concert réparti, la contrainte d'*instantanéité* n'est pas primordiale. Cependant si elle est atteinte, elle fournit un confort de jeu plus proche de l'interaction traditionnelle entre les musiciens, i.e. lorsqu'ils sont ensemble, dans la même salle. Nous proposons dans ce chapitre un deuxième protocole, proche de la cohérence retardé, qui permet d'obtenir la simultanéité et l'instantanéité. Nous n'avons pas développé et testé ce protocole dans la suite de cette thèse, puisque ce protocole impose aux musiciens de ne pas entendre la musique joué par les autres : il est basé sur la notion de chef d'orchestre.

Dans le domaine des jeux multijoueurs, la propriété de *simultanéité* est tout aussi importante, lorsque les joueurs effectuent des actions en parallèle et en collaboration. En effet, en fonction de certaines actions, il est souvent important qu'une décision commune soit prise, et cela dans un délai suffisamment faible pour que le temps de réponse de l'application n'affecte pas le confort de jeu. Dans ce domaine, la propriété de *simultanéité* prend une importance un peu différente, puisque ces décisions sont aussi faites en fonction de l'ordre de prise en compte des opérations de notification. Le modèle de cohérence perceptive reste alors un bon candidat car il permet d'éviter les conflits entre opérations (non-commutation, voir chapitre 5). Il y a actuellement une contraction dans les besoins soulevés pas ces jeux multijoueurs : il est nécessaire d'avoir un mécanisme de cohérence perceptive, tout en conservant un temps de réponse le plus faible possible[36]. Généralement, c'est un serveur qui assure la cohérence en prenant les décisions "litigieuses" et les temps de réponses sont réduits à l'aide de mécanismes de prédiction (le Dead Reckoning par exemple). Ainsi, un protocole de cohérence perceptive complètement réparti (comme celui que nous allons présenter) permettrait d'éviter la centralisation des décisions, et donc de réduire globalement les délais de communication entre les joueurs. Pour cela, nous proposons une solution basée sur la gestion locale de plusieurs états du jeux : l'état affiché est celui qui est géré avec un mécanisme de prédiction d'état, ayant ainsi un temps de réponse le "meilleur possible", mais risque d'avoir des incohérences. Le second état maintenu sera celui calculé avec la cohérence perceptive, avec un retard calculé "au mieux" pour obtenir la simultanéité. Ce second état permettra donc de

[35] Pour fournir la cohérence perceptive, la musique jouée localement est ré-entendue après un délai calculé. Autrement dit, le temps de réponse du système peut être important, tout en restant confortable.

[36] Nous avons montré au chapitre 5 que la cohérence perceptive ne pouvait être obtenue que après un délai équivalent au plus grand délai entre tout couple de joueurs.

détecter les incohérences et d'activer si nécessaire un mécanisme de réparation ou de convergence.

Ainsi, dans ce chapitre, nous présentons un protocole générique de cohérence perceptive, puis nous montrons comment celui-ci peut s'adapter aux spécificités du concert réparti au paragraphe 6.3.1 et aux spécificités des jeux multijoueurs au paragraphe 6.3.2. Nous présentons ensuite au paragraphe 6.4 un protocole de cohérence retardée basé sur le concept de chef d'orchestre (spécifique au concert réparti).

6.2 Un protocole de cohérence perceptive

6.2.1 Introduction

La construction d'un protocole de cohérence perceptive se heurte à plusieurs difficultés :

1. Fournir la propriété de Δ légalité,
2. Fournir la propriété de simultanéité,
3. Conserver la propriété d'ordonnancement qui découle de la Δ légalité et de la simultanéité,
4. Minimiser les incohérences dues à la latence et les gérer quand elles surviennent. Compenser les pertes,
5. Être complètement distribué,
6. Effectuer un traitement léger des opérations de notification, i.e. peu coûteux en messages de synchronisation et en calculs.

La Δ légalité consiste à établir une latence constante pour chaque couple de source/abonné d'une instance de média (1). Dans la mesure où nous distinguons les médias continus et les médias discrets, nous avons deux interprétations différentes de la procédure présentée au paragraphe 6.2.2. Dans les deux cas, l'objectif est d'estimer le moment où peuvent être jouées les mises à jour (les notifications) pour éviter les arrivées tardives. Cette phase aura lieu durant l'initialisation du système.

Nous avons montré dans le chapitre sur la formalisation que la simultanéité (2) ne pouvait être atteinte que si la latence locale était au moins égale à la latence du récepteur le plus éloigné temporellement. La procédure présentée au paragraphe 6.2.2 atteint cet objectif en fournissant une latence locale minimale, en se basant sur la Δ légalité et en étant complètement distribuée (5).

Le choix des horloges et de leurs fréquences doit permettre de conserver l'exactitude de l'ordonnancement (3). Nous supposons pour cela que les horloges sont configurées sur la même fréquence et ne dérivent pas entre elles. Cette hypothèse n'est pas viable sur des longues durées, mais reste acceptable avec du matériel de haute précision (voir le chapitre 9). Pour éviter toute dérive, il faut utiliser un protocole de synchronisation.

Selon la pertinence de la phase de calcul des délais pour établir la Δ légalité et selon la variabilité de l'état du réseau, les incohérences seront minimisées, mais elle peuvent encore survenir, en cas de gigue exceptionnellement grande ou en cas de perte du message (4). Selon le type de médias, il faudra gérer les messages tardifs (i.e. Δ illégaux). Cette gestion est simple dans le cas où le média permet de tolérer des pertes (comme l'audio), mais elle se complique si les données transmises doivent absolument être reçues (comme par exemple certaines informations dans les

jeux multijoueurs) .

De plus, les critères de la cohérence perceptive s'expriment principalement sur les notifications d'instances de médias. Compte tenu que ceux-ci peuvent avoir une fréquence éventuellement grande (jusqu'au Megahertz pour l'audio !). Il faut s'assurer que le système de cohérence ne devra pas négocier les dates des notifications durant l'exécution. Le traitement des messages de l'application doit être minimal (6).

Pour que l'on puisse utiliser cet algorithme, il faut connaître à l'avance le nombre de sources du système, le nombre de consommateurs, ainsi que les adresses multicast de réception des événements et de réception des statistiques nécessaires à la synchronisation. Il faut donc que le trafic multicast soit possible entre les différents sites. Nous supposons donc que les processus sont mis en relation à l'aide d'un système leur permettant de communiquer à l'aide de primitives de multicast. Cela peut être obtenu en développant l'application au dessus du multicast IP ou d'un système Pair-à-Pair adapté à la communication multimédia (voir le chapitre 2). Pour simplifier, nous considérons que le groupe est statique, i.e. qu'il n'y a ni départ ni arrivée de membre(s). La reconfiguration du groupe est cependant nécessaire dans certains types d'applications comme les jeux massivement multijoueurs ou certaines applications de réalité virtuelle.

De plus, nous considérons être dans un système interactif où le sentiment de co-présence est primordial. Les utilisateurs qui interagissent doivent avoir la connaissance de la présence des autres. Le système doit alors savoir à tout moment le nombre de membres du groupe interagissant[37]. Comme nous conservons les notations utilisées dans le chapitre 4, le nombre de processus distants est noté n et leur ensemble est noté $P = (p_1, ..., p_n)$. Il en sera de même pour les instances de médias, les opérations, etc.

6.2.2 L'algorithme

La première phase (le calcul des vecteurs locaux) consiste à dimensionner les délais de l'application grâce à une mesure de l'état du réseau. Dans la procédure que nous allons présenter, les latences ne sont pas calculées numériquement mais estimées les unes par rapport aux autres de façon répartie. Sachant que chaque processus envoie les notifications avec une date physique issue de l'horloge physique locale, chaque processus va calculer les dates des notifications qu'il est capable de jouer simultanément. Autrement dit, chaque processus envoie aux autres sa perception locale de la simultanéité. L'étape suivante (le calcul des délais locaux) consistera à calculer la simultanéité globale. Et enfin, nous exposerons comment chaque message reçu sera traité pour devenir une notification pour l'application.

Le calcul des vecteurs locaux

La figure 6.1 nous donne l'algorithme de la phase d'établissement de la Δ légalité (exécuté par chaque processus) :

$vecteur_{monID}[n]$ est le vecteur de sauvegarde des valeurs mesurées : il contient les différences

[37]Dans les systèmes hautement dynamiques comme les jeux massivement multijoueurs, le nombre total est inconnu, mais le nombre d'utilisateurs avec lesquels il y a interaction est connu ou borné

```
1  int vecteur_monID[n], initialisé à 0
2  Tant Que dimensionnement
3    Recevoir des notifications artificielles (w_i(x)u, t_i(w_i(x)u)) des p_i
4      Si DateCouranteLocale − t_i(w_i(x)u) > vecteur_monID[i]
5      Alors vecteur_monID[i] := DateCourante − t_i(w_i(x)u)
6  Fin Tant Que
7  Diffuser vecteur_monID[n] à tous les p_i
```

FIG. 6.1 – *Calcul des vecteurs locaux*

(absolues) entre l'horloge locale et les horloges distantes[38] (ligne 4). Si les horloges sont synchronisées de façon absolue (par exemple avec des horloges GPS ou avec NTP [Mil92]), alors cette valeur peut être considérée comme une estimation de la latence réelle. Cependant, le protocole fonctionne sans cette synchronisation (il tient compte du fait que les horloges sont locales) et tolère donc la comparaison de dates issues d'horloges ayant un décalage (*lag*).

Au début, chaque processus diffuse une série de notifications pour que les autres puissent estimer les écarts relatifs entre les instances de médias. Si les médias concernés sont continus, alors il suffit d'envoyer des notifications "à vide". Si les médias sont discrets, il suffit de générer de façon artificielle des notifications. Ces messages contiennent alors les champs $(w_i(x)u, t_i(w(x)u)$ où $w_i(x)u$ est la mise à jour et $t_i(w_i(x)u)$ sa date physique.

dimensionnement est la variable utilisée pour calculer la durée de la procédure, elle peut être exprimée en nombre de mises à jours reçues ou en temps physique. C'est à l'application de décider quelle est la durée pertinente. Cette variable sert en fait à déterminer la durée du test qui fera ressortir la gigue maximum considérée comme réaliste par rapport à l'état courant du réseau. En fait, l'algorithme calcule la différence entre la date du message et la date de sa réception (ligne 4) pour n'enregistrer que les différences maximums (ligne 5). Une fois la boucle *Tant Que* finie (ligne 6) le tableau $vecteur_{monID}[i]$ contient les écarts relatifs des horloges physiques pour une communication de p_i vers p_{monID}

A la ligne 7, nous considérons que les écarts relatifs des horloges (les différences) ont été calculés pour les messages provenant des différents processus[39].

Le calcul des délais locaux

Avec l'intégralité des vecteurs, chaque processus peut calculer les délais à introduire localement pour que chacun des autres processus puisse assurer la simultanéité. La procédure de calcul des délais locaux est présentée à la figure 6.2.

Quand un processus reçoit les n vecteurs (ligne 10), il devient conscient de la perception temporelle que chaque processus a des autres. Le principe de l'algorithme est de confronter chacun de ces tableaux pour ajouter localement un délai avant la notification de mises à jours à l'application, et en fonction du processus qui les reçoit avec le plus de retard.

[38] Nous supposons alors que celles-ci ne dérivent pas entre elles, ou suffisamment peu pour que cette valeur reste la même

[39] Le protocole présenté est optimiste car nous savons que la latence sur un réseau IP est variable de façon indéterministe

```
8     sync = false   variable de fin de calcul
9     dif_temp[n]

10    recevoir les n vecteurs
11    Tant Que sync != true
12       Choisir p_ref comme max(p_x) avec 1 ≤ x ≤ n et
            p_x n'a jamais été choisi
13       Pour i de 1 à n
14          p_ajust := p_i
15          Pour j de 1 à n
16             dif_temp[p_j] :=
                vecteur_j[p_ref] − vecteur_j[p_ajust]
17          Fin Pour
18          dif[p_i] := max_j(dif_temp[p_j])
19       Fin Pour ;
20       Si ∀i ∈ [1...n] dif[p_i] > 0 Alors
21         sync := true
22         Pour i de 1 à n
23           ajust[p_i] := dif[p_i]−
              (vecteur_monID[p_ref] − vecteur_monID[p_i])
24         Fin Pour
25       Fin SI
26    Fin Tant Que
```

FIG. 6.2 – *Calcul des délais locaux*

p_{ref} est le processus référence (ligne 12) utilisé pour effectuer ce calcul des différences relatives des horloges (ligne 16). Pour chaque tableau, nous calculons la différence maximum entre un processus donné p_{ajust} et le processus référence p_{ref} (ligne 13 à 18). Tous les processus seront p_{ajust} avant la ligne 20, les maximums calculés seront enregistrés dans le tableau $dif[]$ (ligne 18).

Ayant choisi un processus comme référence pour les calculs, il faut s'assurer que les maximums calculés sont tous positifs (ligne 20). Dans le cas contraire, le délai local à ajouter pour "attendre" le récepteur le plus tardif risquerait d'être négatif (ligne 23), alors il faudrait ajouter un délai négatif, ce qui est impossible. C'est pour cela que si une des valeurs du tableau $dif[]$ est négative, on choisit un autre processus pour être référence jusqu'à en obtenir un qui satisfasse la formule de la ligne 20.

Les processus doivent tous arriver à la ligne 21 en ayant choisi la même référence p_{ajust}. En effet, le même calcul est effectué par tous et avec les mêmes valeurs. C'est pour cela que l'algorithme choisit les processus références du processus de plus grand identifiant au processus ayant le plus petit (ligne 12). Le paragraphe 6.2.3 donne la preuve qu'au moins un des processus choisi comme référence satisfera la formule ligne 20, ce qui constitue une preuve de la terminaison de l'algorithme.

Lorsque l'algorithme arrive à la ligne 22, le tableau $dif[p_i]$ contient les différences des récepteurs les plus tardifs. Ce calcul est identique quelque soit le processus. Il faut maintenant regarder localement quels sont les délais à ajouter pour s'adapter au récepteur le plus tardif de chaque processus (ligne 23). Le tableau $ajust[]$ va alors contenir ces valeurs.

```
27  Tant Que(LesMessagesSontSupposesArriver)
28      Recevoir (w_i(x)u, t_i(w_i(x)u)) de p_i
29      d = DateCouranteLocale
30      écart = d − t_i(w_i(x)u)
31      Si écart ≤ vecteur_monID[i] + ajust[p_i] Alors
32        Notifier c_monID(x)u à l'application lorsque
33            t_monID == d
                +vecteur_monID[i] − écart
                +ajust[p_i]
34      Sinon traiter (w_i(x)u, t_i(w_i(x)u)) comme un message tardif
35  Fin Tant Que
```

FIG. 6.3 – *Le traitement des messages de mise à jour*

Le traitement des messages de mise à jour

Le traitement des messages consiste à déterminer le moment où les données contenues dans celui-ci seront des notifications. Lorsque le tableau $ajust[]$ est calculé, l'initialisation est terminée car chaque processus a effectué les calculs nécessaires pour obtenir les propriétés de Δ légalité et de simultanéité. La figure 6.3 nous montre alors comment le système de gestion de cohérence effectue ses notifications à l'application.

A la ligne 29, l'algorithme teste si le message reçu n'est pas un message tardif, en effet :
- *écart* est l'écart relatif des horloges pour ce message
- $vecteur_{monID}[i]$ est l'écart maximum calculé pour obtenir la Δ légalité
- $ajust[p_i]$ le délai local qu'il faut attendre avant la notification après avoir obtenu la Δ légalité.

Nous pouvons trouver à la ligne 31 le calcul du temps à attendre avant de notifier l'application, en effet $vecteur_{monID}[i] - écart$ est le temps à attendre depuis la date physique d pour obtenir la Δ légalité. Cette valeur peut être négative si $ajust[p_i]$ est suffisamment important (ligne 31). Cette attente peut être obtenue en utilisant un buffer circulaire dont la consommation régulière est contrôlée par une horloge physique (comme celle d'une carte son par exemple).

Le traitement d'un message tardif (ligne 34) est fortement dépendant de l'application et du média utilisé pour l'interactivité, nous reviendrons sur ce point dans les cas spécifiques du concert réparti au paragraphe 6.3.1 et des jeux multijoueurs au paragraphe 6.3.2.

6.2.3 Complexité et preuve de terminaison de l'algorithme

En nombre de messages, la complexité est de $\theta(n)$[40] pour une transmission en multicast (ligne 4), en effet le nombre de messages envoyés sera toujours de n. Si la transmission n'est pas multicast, la complexité augmentera à $\theta(n^2)$ et le nombre de messages à n^2.

Dans l'algorithme de synchronisation, en temps de calcul, la boucle (b) de la ligne 13 à 19 est en $\theta(n^2)$. Étant donné que l'algorithme se termine, la boucle (a) de la ligne 11 à 26 s'exécutera au maximum n fois. La boucle (b) a donc une complexité de $\theta(n^3)$. La boucle (c) de la ligne 22 à 24 ne s'exécutera qu'au dernier tour de la boucle (a) (la variable *sync* étant mise à *vrai* ligne 21). Sa complexité est de n. La complexité "pire cas" de l'algorithme de calcul des délais locaux

[40] θ définit une borne supérieure asymptotique

est donc de $\theta(n^3 + n)$.

Pour que l'algorithme termine, nous devons être certains que nous allons trouver un processus référence qui satisfasse la formule $dif[p_i] > 0$ (ligne 20). Autrement dit, il faut que chacun des processus p_i soit perçu au moins une fois en retard par rapport à p_{ref}. Cela peut être perçu par n'importe quel processus p_j puisqu'on ajuste les flux par rapport à celui qui met le plus de temps à recevoir les messages. Nous allons maintenant démontrer que cette propriété est vérifiée.

Nous excluons la comparaison du processus référence avec lui-même car cela n'a pas de sens d'être en avance ou en retard par rapport à soi même. En fait, nous considérons que seules les différences positives sont intéressantes.

Définition 9 *Soit $f(i,j) = i \xrightarrow{k} j$ la relation binaire telle que le processus p_i est perçu par un autre processus p_k comme en retard par rapport au processus p_j.*

De par sa nature, la relation f est anti-réflexive ($f(i,i)$ n'a pas de sens) et est anti-symétrique :

$$\neg(i \xrightarrow{k} j) \Leftrightarrow j \xrightarrow{k} i \tag{6.1}$$

En effet, pour le processus $p_k (1 < k < n)$, si p_i est perçu comme en avance (respectivement en retard) par rapport à p_j alors p_j est perçu par p_k comme en retard (respectivement en avance) par rapport à p_i. Notons qu'en cas d'estampilles identiques, p_k peut départager p_i et p_j avec leurs identifiants.

De plus, sachant que chaque processus dispose des mêmes valeurs :

$$\forall k(\neg(i \xrightarrow{k} j)) \Rightarrow \forall k(j \xrightarrow{k} i) \tag{6.2}$$

La relation f est transitive :

$$(i \xrightarrow{k} j) \wedge (j \xrightarrow{k} l) \Rightarrow i \xrightarrow{k} l$$

En effet, si pour p_k, p_i est en retard par rapport à p_j qui est lui même en retard par rapport à p_l, alors p_i est en retard par rapport à p_l.

De plus, de par la propriété de conjonction du quantificateur $\forall$:

$$\forall k(i \xrightarrow{k} j) \wedge \forall k(j \xrightarrow{k} l) \Rightarrow \forall k((i \xrightarrow{k} j) \wedge (j \xrightarrow{k} l)) \Rightarrow \forall k(i \xrightarrow{k} l) \tag{6.3}$$

Théorème 1 *Dans la procédure de traitement des estampilles, il existe toujours une source référence p_{ref} telle que :*
Chacun des processus p_i $(1 < i < n$ et $i \neq ref)$ du système est perçu par au moins un des processus p_k $(1 < k < n)$ comme ayant du retard par rapport à p_{ref}.

Le théorème 1 s'énonce donc comme suit (le processus p_j vérifiant cette formule est le processus référence) :

$$\exists j \forall i \exists k(i \xrightarrow{k} j) \tag{6.4}$$

A partir de maintenant, nous allons restreindre le domaine de i et j à $0 \leq i < n$, $0 \leq j < n$ et $i \neq j$ (conforme à la définition 9).

Preuve 1 *Nous allons montrer que le théorème 1 est vérifié. Pour cela, nous allons montrer que la formule 6.4 est vraie. Il suffit alors de démontrer que la contraposée de cette formule est fausse. Donc que la formule 6.5 est fausse :*

$$\neg \text{ (formule 6.4)} \Leftrightarrow \neg(\exists j \forall i \exists k (i \xrightarrow{k} j)) \Rightarrow \forall j \exists i \forall k \neg (i \xrightarrow{k} j)$$

Donc, par la formule 6.1 :

$$\Rightarrow \forall j \exists i \forall k (j \xrightarrow{k} i) \tag{6.5}$$

Nous allons procéder par l'absurde pour démontrer que cette formule est fausse. Nous allons alors la supposer vraie, donc si $j = x_0$ alors il existe $i = x_1$ tel que $\forall k(x_0 \xrightarrow{k} x_1)$. Sachant que i et j évoluent dans le même domaine, choisissons maintenant $j = x_1$. On sait qu'il existe $i = x_2$ tel que $\forall k(x_1 \xrightarrow{k} x_2)$. Nous pouvons continuer de même jusqu'à obtenir $\forall k(x_n \xrightarrow{k} x_{n+1})$ ($j = x_n$ et $i = x_{n_p+1}$). Nous aurons donc :

$$\forall k(x_0 \xrightarrow{k} x_1) \wedge \forall k(x_1 \xrightarrow{k} x_2) \wedge \ldots \wedge \forall k(x_n \xrightarrow{k} x_{n+1})$$

A ce moment, nous aurons trouvé au moins un doublon dans les valeurs que peuvent prendre les membres de gauche et les membres de droite des différents facteurs puisqu'il y en a n différents possibles et qu'ils sont au nombre de $n+2$. Prenons alors x_a et x_b ce doublon ($x_a = x_b$). Comme $\forall k(x_n \xrightarrow{k} x_{n+1})$, alors par définition, $x_n \neq x_{n+1}$. Il existe donc x_c tel que :

$$\forall k(x_a \xrightarrow{k} x_{a+1}) \wedge \ldots \wedge \forall k(x_{c-1} \xrightarrow{k} x_c) \wedge \forall k(x_c \xrightarrow{k} x_{c+1}) \wedge \ldots \wedge \forall k(x_{b-1} \xrightarrow{k} x_b) \tag{6.6}$$

Dans ce cas, en appliquant la propriété (6.3) à la formule précédente (6.6) sur les valeurs intermédiaires à x_a, x_b et x_c :

$$\forall k(x_a \xrightarrow{k} x_c) \wedge \forall k(x_c \xrightarrow{k} x_b)$$

Or $x_a = x_b$, nous obtenons donc une contradiction (propriété 6.2) suivante :

$$\forall k(x_a \xrightarrow{k} x_c) \wedge \forall k(x_c \xrightarrow{k} x_a) \tag{6.7}$$

Nous pouvons donc dire par l'absurde que la contraposée du théorème (formule 6.5) est fausse.

Le théorème 1 étant vérifié, l'algorithme s'arrêtera et appliquera les retards nécessaires à la synchronisation collective. Donc une décision collective sera prise par tous.

6.3 Application du protocole de cohérence perceptive

6.3.1 Application du protocole au concert réparti

L'interactivité musicale est soumise à des contraintes temporelles fortes : quand des musiciens sont dans la même pièce, ils s'entendent instantanément et simultanément. En effet, pour des sons brefs, l'oreille humaine ne perçoit pas de décalage d'un délai inférieur à 20ms (correspondant à 6 mètres si le son voyage à 300 mètres par seconde).

Il est vrai que si le son est transporté par un réseau informatique (dont le support initial est le signal électrique), nous pouvons transporter le son plus vite que sa vitesse physique, nous permettant potentiellement de faire communiquer des musiciens à de grandes distances avec les mêmes conditions temporelles qu'en salle de répétition (ou de concert).

Cependant, le tableau 6.4 nous montre le temps pris par un signal lumineux[41] pour aller d'un

[41] Rappelons que la vitesse de la lumière est de 300000 km/s

Couple de ville	**Distance en km**	**Temps du parcours en ms**
Lille - Perpignan	883	2,94
Miami - Seattle	5 309	17,7
circonférence terrestre/2	20 000	66,6

FIG. 6.4 – *Les limites physiques de la latence*

point à l'autre de la planète à vol d'oiseau. Cela constitue la limite physique de la latence. Ce tableau nous montre que si la vitesse des réseaux informatiques approche la vitesse de la lumière, alors l'interactivité musicale pourra se faire à l'échelle d'un continent avec un seul système de streaming basse latence, mais qu'à une échelle plus grande cet argument n'est plus valable : il faudra alors ajouter des astuces pour aider les musiciens à tolérer ces délais de communication, sachant que la condition première est la constitution d'un tempo simultané (voir chapitre 7).

Le protocole de cohérence s'adapte parfaitement à la problématique du concert réparti. En effet la musique est un moyen d'interaction fortement synchrone nécessitant une simultanéité fine parmi les musiciens qui doivent tous s'entendre mutuellement avec un tempo commun. L'interaction se fera à l'aide d'instruments de musique dont le son est numérisé (échantillons audio isochrones de type *PCM* pour Pulse Code Modulation [RdR99]).

La communication par échange d'échantillons audio entre musiciens apporte des propriétés intéressantes. D'une part, les systèmes de streaming sont déjà basés sur le principe de la Δ légalité, puisqu'ils utilisent des buffers pour amortir la gigue du réseau et fournir une latence constante entre un émetteur et son (ou ses) récepteur(s). Aussi, les opérations d'écriture ne sont pas concurrentes car les échantillons provenant de différentes sources sont simplement additionnés (*mixés*). De plus, pour un écrivain donné, l'écriture la plus récente "invalide" complètement les précédentes [42]. La procédure de traitement des messages tardifs de la ligne 34 du protocole est alors simple : le message est jeté.

Si les latences réseaux sont importantes, alors le protocole impose aux musiciens de jouer leur musique en ayant un retour du mixage des flux retardés. Ce retard peut sembler gênant (surtout s'il ne respecte pas le tempo du morceau) mais lors de nos tests, les musiciens s'en accommodaient très bien. Nous verrons dans le chapitre 7 comment nous avons procédé pour gérer ce problème.

6.3.2 Application du protocole aux jeux multijoueurs

Contrairement au concert réparti, dans les jeux multijoueurs il n'est pas possible de retarder la perception des actions effectuées. En effet, dans la pratique musicale, le simple fait de toucher son instrument, de souffler dedans, de le gratter ou encore d'entendre le son direct qu'il produit est une information permettant au musicien d'avoir un retour sensoriel de son action sur le système. Cela n'est généralement pas le cas dans les jeux vidéo sans retours d'efforts. Comme le retour à l'interface est le seul indice laissant comprendre à l'utilisateur que son action a été prise

[42] Nous ne considérons pas le codage PCM différentiel

en compte par le système, l'instantanéité est primordiale.

Dans ce paragraphe, nous considérons uniquement les jeux multijoueurs interactifs proposant des interactions fortement couplées. Cela couvre les jeux de sport, les FPS (First Person Shooter), les RPG (Role Playing Games), les RTS (Real Time Strategy) et plus généralement les jeux basés sur des interactions ayant des propriétés physiques spatio-temporelles (comme par exemple la réalité mixée [CFG+03]). Nous écartons alors les jeux de type "tour par tour", ainsi que les jeux dans lesquels les actions sont faiblement couplées.

Architecturalement, nous supposons que le système est complètement distribué, qu'il soit constitué uniquement de processus de joueurs ou qu'il soit constitué de serveurs miroirs de l'état du jeu (voir paragraphe 2.1.2).

Du point de vue du jeu, voici les contraintes temporelles attendues [BGS04] :

1 La jouabilité du jeu sera satisfaisante si les actions locales sont *instantanément* prises en compte dans l'affichage (visuel ou sonore)

2 Chaque action est prise dans le même ordre par chaque processus, quels que soient les écrivains

3 Le délai entre deux notifications d'actions quelconques doit être le même pour tous les joueurs (*simultanéité* et Δ *légalité*).

Voici un exemple intuitif de ces contraintes, si elles sont appliquées à un jeu de course automobile à deux joueurs :

1 Un changement de direction provoqué par une pression sur la manette doit s'effectuer immédiatement (on ne peut bloquer les écritures)

2 Lorsque deux joueurs sont proches dans la course et qu'ils passent la ligne d'arrivée, alors les deux joueurs doivent voir le même joueur passer la ligne le premier

3 Les collisions sont détectées avec les mêmes paramètres de direction/position/vitesse si le couple de voitures ainsi que leurs caractéristiques sont les mêmes à l'instant de la collision (*simultanéité*)

3bis La vitesse d'une voiture est conservée si le délai entre l'écriture de deux positions successives est le même pour les notifications de l'autre joueur (Δ *légalité*).

Ces contraintes semblent contradictoires car nous avons formellement montré que nous ne pouvions obtenir l'instantanéité avec la simultanéité si les latences du réseau sont trop grandes. Cependant le couplage de l'algorithme qui est proposé dans ce chapitre avec une technique de masquage de la latence permet d'atteindre quasiment ces objectifs. Nous allons présenter une telle technique dans le domaine des jeux multijoueurs : le Dead Reckoning.

Le Dead Reckoning

Certaines actions dans les jeux sont prévisibles, particulièrement les mouvements des objets. L'exemple le plus évident est une voiture roulant en ligne droite : connaissant son accélération, sa vitesse, sa position et sa direction, il est facile de prédire ses positions futures.

Cette technique a plusieurs intérêts [ABK+04]. Tout d'abord, elle permet d'économiser de la bande passante car l'écrivain n'a besoin d'envoyer que les écritures non prévisibles, voir même de ne pas envoyer une écriture, si l'erreur de prédiction est suffisamment faible.

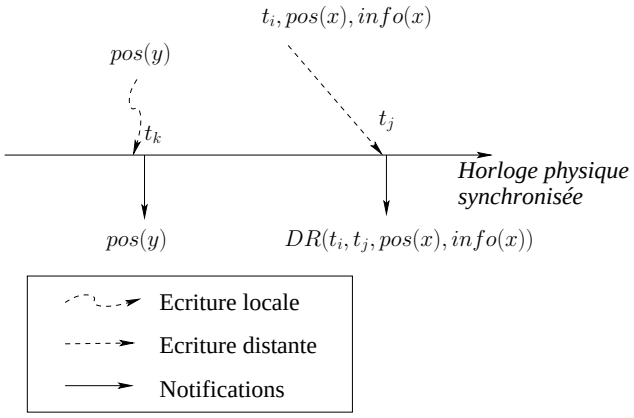


FIG. 6.5 – *Le Dead Reckoning et l'affichage des actions*

De plus, comme nous l'expose la figure 6.5 elle peut être utilisée pour "masquer" la latence du réseau : les écritures locales sur l'instance de média y, comme $pos(y)$, sont affichées aussi vite que possible, i.e. avec le temps pris par l'affichage (instantanéité). Par contre les messages reçus du réseau sont des actions qui ont été effectuées à la date de réception moins la date écrite dans le message (si les horloges sont synchronisées). La notification affichée sera alors la prédiction de la position courante t_j, si pour l'instance de média x l'objet associé était à la position $pos(x)$ avec les caractéristiques $info(x)$ à la date t_i. La précision de l'affichage dépend alors de deux facteurs : l'apparition d'actions imprévisibles et la fonction de prédiction elle-même (DR). Cette technique permet de minimiser les erreurs d'ordonnancement et de simultanéité tout en conservant l'instantanéité et la Δ légalité.

Notons que la définition du Dead Reckoning n'est pas toujours la même, par exemple dans [PW02b] où la réception d'une écriture implique son affichage direct, sans se préoccuper de la différence entre la date locale et la date du message.

Comme les actions sont générées par l'utilisateur, elles ne peuvent pas toutes être prédites par le Dead Reckoning, mais un mécanisme de convergence peut "ramener" l'état du média vers une position cohérente, une fois l'information correcte connue. Cela provoquant un effet inévitable de divergence / convergence / divergence ... [PW02b]. Même si certaines erreurs seront tolérées, il n'est pas impossible que lors d'une phase de divergence, une décision soit ratée (comme une collision entre deux voitures). Il faut donc un mécanisme supplémentaire pour détecter les incohérences et réparer l'état du jeu.

Un tel mécanisme, la synchronisation à états de remorquage (TSS pour Trailing States Synchronization) a été proposé dans [CFKJ02]. La solution consiste à maintenir plusieurs états du jeu qui seront perceptivement cohérents, chacun avec un retard différent. Cela veut dire que l'arrivée d'un message est acheminée vers l'état affiché (avec Dead Reckoning), mais aussi vers les états de remorquage, pour être pris en compte au moment pertinent. Lors de la détection d'une divergence par un des états de remorquage, le système va soit faire converger l'état d'affichage, soit prendre (ou modifier) une décision importante dans le jeu.

Les délais choisis pour les états de remorquage le sont de façon statique (le premier état est

retardé de 200 ms et le deuxième à 400 ms) avec un protocole équivalent à ceux de [MVHE04, GD98]. Comme expliqué au paragraphe 5.3, ces protocoles utilisent une horloge synchronisée avec NTP et choisissent la date de notification selon la formule suivante :
$T(c(x)v) = d + T(w(x)v)$ (où T est l'horloge synchronisée).

L'utilisation de notre protocole de cohérence perceptive comme premier état de remorquage permettrait d'optimiser ce mécanisme : cela réduirait le délai de détection des incohérences si la latence est faible sur le réseau et cela permettrait d'éviter de maintenir un état de remorquage à latence trop faible si la latence réelle du réseau est importante. Remarquons que le deuxième état de remorquage pourrait éventuellement prendre en compte les valeurs calculées par le protocole pour le premier.

Dans la mesure où notre protocole de cohérence perceptive hérite des propriétés de la cohérence perceptive, l'application de celui-ci dans les jeux multijoueurs permet d'éviter les conflits entre les opérations effectuées par les joueurs, et cela de façon complètement distribuée. Ce paragraphe fait l'objet d'un publication scientifique [Bou05a].

6.4 Un protocole de cohérence retardée spécifique au concert réparti

6.4.1 Introduction

Comme nous l'avons vu dans le paragraphe 6.3.1, l'écoute cohérente des autres musiciens dans un système d'interactivité musicale en réseau implique d'avoir un retard local. La solution que nous proposons ici est d'annuler cette latence grâce à une réduction de l'interactivité entre certains musiciens. Cela est possible si la pratique musicale le permet. C'est le cas par exemple de formations intégrant un chanteur ou une chanteuse, et où certains instrumentistes ont pour fonction principale d'accompagner celui ou celle-ci en contre chant. Nous pouvons citer par exemple l'accompagnement de la flûte dans la tradition vocale du style Tembang de la région de Sunda à Java Ouest [Wid94], ou encore le soutien de la vièle Kamânche dans la tradition du Mugam d'Azerbaïdjan [Pie93]. En fait, la propriété que nous cherchons à exploiter est le caractère quasi unidirectionnel que l'on peut trouver quelquefois dans les pratiques musicales. En effet, le chant n'a pas vraiment besoin d'entendre l'accompagnement du contre chant avec son propre retour.

La solution que nous proposons se base sur le concept de chef d'orchestre. C'est donc une architecture centralisée.

6.4.2 L'algorithme et l'architecture de communication

Pour un système à n musiciens, notons p_i un processus "musicien" et p_{chef} le processus "chef d'orchestre". Notons maintenant $t_{chef}(w_{chef}(x)u)$ la date du son u joué sur le flux x par le chef d'orchestre. Le chef d'orchestre utilisera cette date comme estampille dans le message contenant le son. Les musiciens possèdent aussi leurs propres horloges, mais l'estampille écrite pour l'envoi des écritures sera notée $e(w_i(x)u)$.

La différence entre la fonction de chef d'orchestre et la fonction de musicien réside dans l'utilisation de l'horloge. Le chef d'orchestre va utiliser l'horloge de sa carte son (chaque tic incrémente de un la date). Nous supposons alors que l'échantillonnage se fera à la même cadence chez les

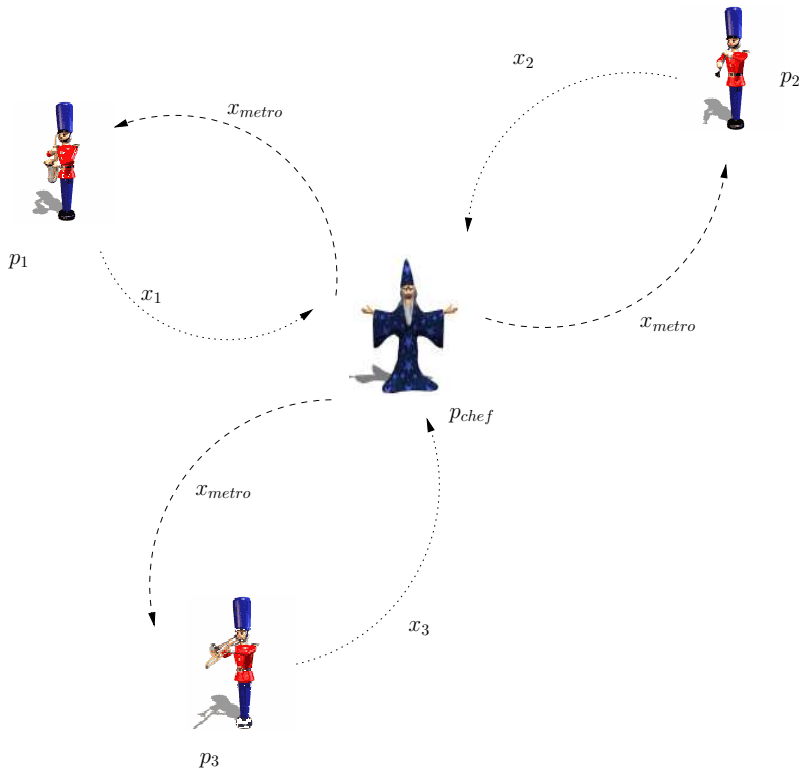


FIG. 6.6 – *L'architecture "chef d'orchestre"*

musiciens (et sans dérive). Nous pouvons maintenant imaginer la stratégie de communication présentée à la figure 6.6.

Le chef d'orchestre va jouer le rôle de métronome pour le groupe. Il va donc transmettre un flux audio x_{metro} (contenant un rythme par exemple) contenant les dates des échantillons transmis. Chaque musicien va recevoir ce flux avec une latence constante, grâce à un système de streaming, assurant la Δ légalité. Il pourra alors jouer sa musique en même temps qu'il entendra le son du chef d'orchestre et sans décalage local.

Pour effectuer la synchronisation, le processus du musicien p_i va renvoyer au chef d'orchestre le son joué qu'il joue, mais en datant ses échantillons (ses écritures) comme suit :

$$Si\ t_i(w_i(y)v) == t_i(c_i(x_{metro})u)\ alors\ e(w_i(y)v) := t_{chef}(w_{chef}(x_{metro})u) \tag{6.8}$$

Le musicien va donc dater chaque écriture avec la date d'origine de l'échantillon du flux de synchronisation entendu au moment de l'écriture. Cela permettra au chef d'orchestre de reconstituer le mixage des flux en conservant une synchronisation rythmique, puisque les musiciens auront pris une référence temporelle commune :

$$Si\ e(w_j(y)w) = e(w_i(x)v)\ alors\ t_{chef}(c_{chef}(y)w) = t_{chef}(c_{chef}(x)v) \tag{6.9}$$

A la réception des flux venant des musiciens, il suffira au chef d'orchestre (ou à un autre site dédié) de mixer les échantillons ayant la même estampille. La propriété de simultanéité sera alors conservée tout en garantissant l'instantanéité aux musiciens.

6.4.3 Une architecture spécifique pour des interactions spécifiques

Comme l'architecture du système est fortement dépendante du rôle de chacun dans la pièce musicale, il pourrait être nécessaire de pouvoir "configurer" la communication en fonction de l'interaction et permettre à des musiciens d'en entendre d'autres.

La figure 6.7 reprend l'exemple du chanteur et du musicien qui accompagne en contre chant. p_1, le chanteur reçoit un flux audio d'une rythmique produite par le chef d'orchestre p_{chef} et applique la synchronisation (formule 6.8) pour produire le flux estampillé x_1 qui sera envoyé à la fois au chef d'orchestre et au musicien p_2 (effectuant le contre chant). p_2 va alors prendre le flux x_1 comme flux de synchronisation et va l'utiliser pour produire le flux x_2, qui sera renvoyé au chef d'orchestre qui effectuera l'assemblage des flux x_1, x_2 et x_3 (formule 6.9). Pendant ce temps le musicien p_3 va pouvoir ajouter sa musique sur la rythmique du chef d'orchestre, à condition qu'il puisse se repérer dans la rythmique et qu'il connaisse sa partition.

Nous pouvons aussi ajouter un flux d'indications audio issu du chef d'orchestre dans lequel celui-ci pourrait diriger les musiciens en leur indiquant un changement de structure ou de façon de jouer. Cela pourrait être utile pour certains types de musiques improvisées où c'est le chef d'orchestre qui choisit le musicien qui va intervenir à un moment donné. Dans certains cas, il peut même construire le morceau en chantant une à une les parties de chaque musicien pendant le déroulement du morceau. Certaines compositions polyrythmiques du Jazz-man Steve Coleman (`http://www.m-base.com/`) sont basées sur ce principe et peuvent donc s'adapter à l'architecture "chef d'orchestre".

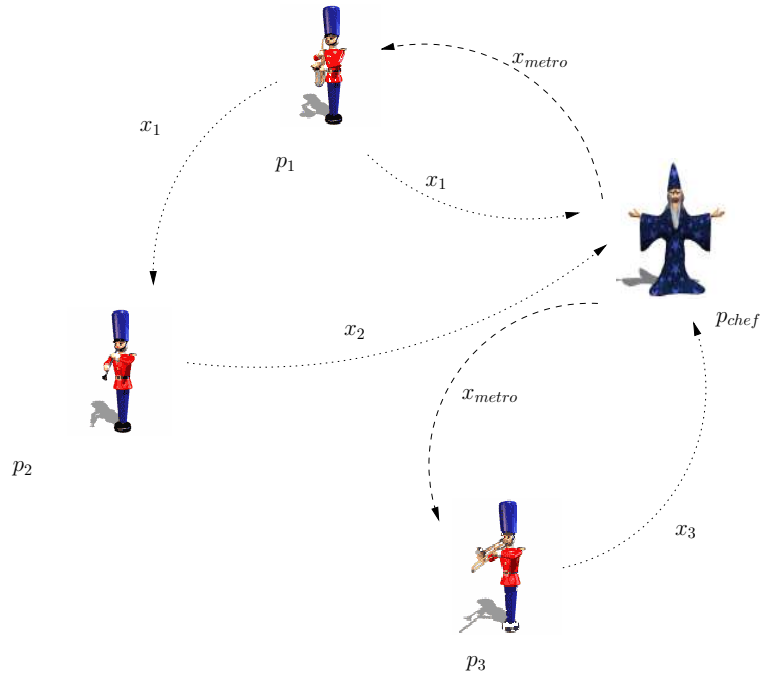


FIG. 6.7 – *L'architecture "chef d'orchestre" adaptée*

6.5 Conclusion

Dans ce chapitre, nous avons montré que les modèles de cohérence perceptive et de cohérence retardée pouvaient être instanciés par des protocoles qui satisfont les contraintes des AMID. De plus, nous avons illustré l'utilisation et la mise en œuvre du protocole de cohérence perceptive dans deux applications fournissant des types d'interactions distincts : le concert réparti et les jeux multijoueurs. Nous avons montré que le protocole de cohérence perceptive fournissait cette propriété avec une latence minimale : la plus grande latence réseau entre deux hôtes du système.

Dans ces deux types d'applications, les propriétés fournies par la cohérence perceptive sont correctement conservées, tout en garantissant un calcul léger durant l'exécution, i.e. sans message de synchronisation propre à la mise à jour et avec un minimum de calcul. Aujourd'hui l'implémentation de la solution dans un jeu multijoueurs reste à l'ordre du jour, mais l'implémentation du protocole dans le concert réparti, ainsi que les tests feront l'objet de la partie suivante.

De plus et du point de vue de la pratique musicale, il est rare que l'interaction soit unilatérale pour tous les musiciens, il sera alors nécessaire dans la mise en œuvre de donner aux musiciens soit une interaction "chef d'orchestre", soit une interaction perceptivement cohérente, et cela en fonction du rôle de chaque musicien dans l'œuvre musicale. Cette éventuelle composition des solutions ne pourra se faire qu'avec une connaissance fine des "codes métiers" de la musique, et donc dans une collaboration mélangeant les métiers de la musique et des systèmes répartis.

Nous proposons donc dans ce chapitre un ensemble de solutions théoriques nouvelles qui tiennent compte à la fois des formalismes et protocoles existants, mais aussi de la latence du réseau sous-jacent et des contraintes psycho-perceptives liées aux utilisateurs. Dans la partie

suivante de cette thèse, nous allons étudier la mise en œuvre du protocole de cohérence perceptive pour construire un système de PMID. Cette mise en œuvre va se faire avec l'ajout de mécanismes reposants sur le protocole : l'adaptation du retard local à la structure temporelle du morceau et le métronome global.

Quatrième partie

Mise en œuvre

When you hear music, after it's over, it's gone in the air. You can never capture it again.

[Eric Dolphy]
Extrait du disque *Last Date*, 1964

7

nJam, un prototype pour la musique interactive distribuée

Sommaire

7.1 Introduction

La conception d'un système de Performances Musicales Interactives Distribuées (PMID) nécessite un système de co-présence dans lequel les musiciens communiquent en multicast (le mot est pris au sens général, voir le paragraphe 2.1). Si le système de communication est considéré de bout en bout, i.e. des instruments de musique aux oreilles des musiciens, alors la latence introduite par le système est due à la propagation du son dans l'air avant l'acquisition, à l'acquisition elle-même, la numérisation et la restitution, aux traitements effectués par les applications communiquantes et par le réseau.

Dans les travaux que nous allons présenter, c'est le réseau qui est identifié comme la source principale de latence. Bien que cela soit vrai en cas de transmission de flux à distances importantes, ça ne l'est pas dans tous les cas puisque le traitement audionumérique effectué par les cartes sons et les logiciels utilisent des tampons, qui introduisent une latence équivalente à la durée correspondante aux données qu'ils contiennent.

Ainsi, pour fournir une latence faible, une carte son doit être capable d'effectuer des traitements sur un tampon de la plus petite taille possible. Les cartes bon marché comme celles qui sont intégrées dans les cartes mères des PCs ne sont généralement pas capables de travailler sur des tampons de moins de 1024 octets (soit 11 ms pour un échantillonnage à 44100 Hz 16 bits) sans introduire des pertes d'échantillons, causant des artefacts auditifs.

Le système de PMID que nous allons présenter doit donc être capable d'accéder aux périphériques audio avec la configuration la plus fine possible. Nous avons pour cela utilisé le logiciel jMax, qui est un outil de programmation d'application audionumérique temps réel. Nous présentons cet outil au chapitre 7.2.1. De plus, les latences de bout en bout ne sont pas compressibles, et introduisent des incohérences lors de l'écoute mutuelle parmi les musiciens. Nous proposons donc d'intégrer le protocole de cohérence perceptive ainsi qu'un système de métronome global, dans le but de compenser les latences du système.

Nous avons présenté au chapitre précédent un protocole de cohérence perceptive adapté au concert réparti. Nous présentons dans ce chapitre l'implantation de ce protocole. Pour cela, nous introduisons au paragraphe 7.2 l'outil de prototypage d'applications audionumériques appelé jMax. Ensuite, nous présentons au paragraphe 7.3 *nJam* (pour network jam), le module que nous avons implanté dans jMax pour établir la cohérence perceptive et un métronome réparti partagé. Nous expliquons ensuite au paragraphe 7.4 la solution que nous proposons aux musiciens pour jouer de la musique en temps réel et à distance. Cette solution utilise nJam et tient compte de paramètres issus du domaine de la musique.

7.2 jMax : un outil de prototypage pour l'audionumérique

7.2.1 Principes de fonctionnement et interface

De même que MAX[43], jMax [Déc00] est souvent présenté comme un outil de programmation visuelle : 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'autres modules (voir figure 7.1). Les connexions entre les modules représentent des communications de paramètres de contrôle ou de signaux, avec une sémantique d'envoi de messages.

[43] `http://www.cycling74.com/`

Notons qu'il existe deux grandes familles de modules : ceux de contrôle et ceux de traitement du signal. Le nom des modules traitant des signaux se terminent par le caractère "~".

Ainsi, un module peut être une unité de traitement (opérateur arithmétique simple ou transformation complexe du signal) ou être un module système d'entrées-sorties audio ou MIDI[44]. 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).

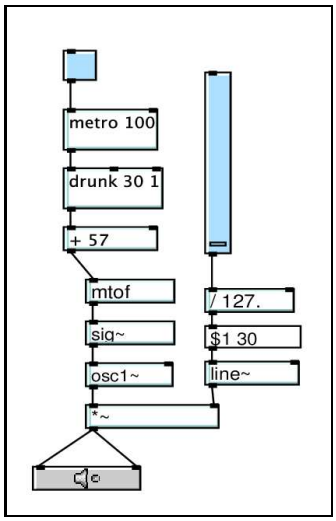


FIG. 7.1 – *Exemple de programme jMax (patch)*

La figure 7.1 nous montre l'exemple d'un patch jMax produisant une suite de notes aléatoires à un tempo donné. Le potentiomètre sert ici à régler le volume du son envoyé à la sortie audio située en bas de la figure (symbolisée par le module avec une baffle). L'interrupteur du haut active le module *metro* (un métronome) qui donne l'impulsion au module *drunk*, qui à son tour génère un nombre aléatoire choisi comme tempo de la mélodie. Les entiers seront convertis en notes grâce aux modules *mtof* (pour midi to frequency) et *sig* (pour signal).

7.2.2 L'architecture logicielle de jMax

Dans jMax, l'interface graphique utilisateur est découplée du moteur de calcul temps réel appelé FTS (pour Faster Than Sound). La communication entre les deux parties est de type client/serveur et utilise les "sockets" TCP/IP ou unix. Ainsi, le moteur FTS est programmé à l'aide du langage C tandis que l'interface est programmée en Java.

Le moteur FTS est une sorte de machine virtuelle qui ordonnance et exécute le code d'objets simples (les modules), qui peuvent communiquer entre eux par envoi de messages synchrones. Ces objets sont représentés à l'interface graphique par des boîtes (comme par exemple l'objet *metro* de la figure 7.1). La connexion entre objets se fait par points d'entrées et de sorties. La figure 7.1 se lit donc de haut en bas et les messages entrants sont interprétés par une fonction programmée dans le code de l'objet.

[44]Musical Instrument Digital Interface, voir paragraphe 3.2.1

Un patch jMax n'est donc rien de plus qu'un graphe d'objets connectés dans lequel les messages sortants d'un objet sont envoyés à tous ceux qui possèdent un point d'entrée (*inlet*) à un de ses points de sortie (*outlet*). L'utilisateur peut lui aussi intervenir en envoyant des messages lors de l'exécution, grâce à l'interface graphique : envoyant un signal MIDI ou en manipulant un objet de contrôle (interrupteur ou potentiomètre, voir la figure 7.1).

Le noyau de FTS doit donc exécuter périodiquement les fonctions de traitement des objets et gérer les événements provenant des utilisateurs et des objets. Ce noyau est en fait un ordonnanceur qui va effectuer un ensemble de sous tâches à chaque période :

1) Interpréter les messages
2) Exécuter les actions associées aux messages
3) Calculer les fonctions de traitement des objets du patch.

Le calcul du signal produit par chaque objet de traitement (les objets dont le nom finit par le caractère "~") est optimisé par une "pré-compilation" du graphe. Celle-ci trie les fonctions dans une liste ordonnée en fonction de la topologie des modules. Ainsi, à chaque cycle d'exécution, chacune des fonctions va traiter un vecteur d'échantillons, typiquement 64 échantillons (correspondant à 1,45 ms pour une fréquence d'échantillonnage de 44100Hz).

7.2.3 Les applications jMax

jMax est une application utilisée principalement par les musiciens pour les performances "Live" utilisant du traitement de signal ou du contrôle de périphérique (avec le MIDI). jMax est aussi une application de "suivi de partition" [OD01], permettant à des musiciens pratiquant des instruments acoustiques de jouer une partition qui sera suivie simultanément par jMax. Grâce à ce système, le compositeur peut programmer des effets spécifiques à appliquer à l'instant auquel se trouve le musicien dans la partition.

jMax est un environnement idéal pour prototyper une application musicale interactive distribuée, en effet :

- jMax implémente un système d'entrées sorties conviviales vers les périphériques audio. Il est capable de gérer une multiplicité de canaux audio, permettant la spatialisation du son
- jMax dispose d'un ensemble d'objets de traitement du signal, d'objets MIDI et d'objets de contrôle comme des opérateurs arithmétiques, des délais, métronomes, etc
- jMax peut être contrôlé par un utilisateur en cours d'exécution
- jMax possède une interface pour le développement de modules externes.

jMax ne possède pas de module natif permettant d'effectuer un transfert de données temps réel entre deux instances distribuées de jMax. La conception d'une application de musique interactive en ligne avec jMax va donc consister à intégrer une partie réseau dans jMax et à programmer un patch permettant de configurer les paramètres nécessaires à l'installation des musiciens.

Nous allons maintenant présenter notre module nJam[45] qui ajoute les fonctionnalités nécessaires à la mise en œuvre d'un système de Performances Musicales Interactives et Distribuées.

[45] Le prototype nJam a été intégré dans la distribution CVS de jMax

7.3 Introduction à nJam

nJam (pour Network Jam[46]) est développé comme un module externe à jMax et fournit un ensemble de fonctionnalités :

- nJam est un *moteur de streaming* permettant aux musiciens de s'entendre mutuellement. La communication utilise le multicast IP ASM (voir paragraphe 2.1.1) et le protocole RTP. Le format audio utilisé est le MIC avec une fréquence d'échantillonnage de 44100 Hz et une quantification de 16 bits (la qualité des CD audio). Pour cela, chaque instance de nJam gère des tampons de réception des flux audio.
- Les instances de nJam sont capables de se synchroniser entre elles pour fournir la *cohérence perceptive*
- Grâce à la cohérence perceptive, nJam fournit un *métronome global* qui est commun à tous les musiciens.

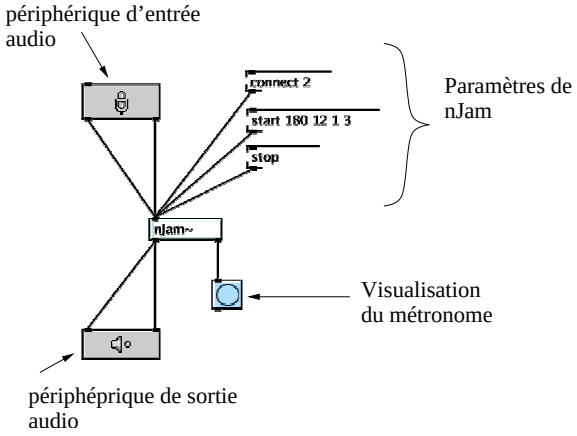


FIG. 7.2 – *Le module nJam dans jMax*

nJam (figure 7.2) est un objet de traitement du signal jMax, il accepte deux types de données en entrée : des échantillons audio pour effectuer un transfert de flux vers les autres instances de nJam et les données discrètes (le message de contrôle). Par exemple, le message "connect 2" sera transmis à nJam comme une donnée discrète, dans le but de configurer le nombre total d'instances de nJam qui vont communiquer (ici deux). Les messages "start 180 12 1 3" et "stop" servent à contrôler le métronome avec des paramètres de tempo qui seront expliqués dans le paragraphe 7.5. L'envoi du message est provoqué par un clic sur l'objet. Ces objets peuvent être édités pour que l'utilisateur puisse entrer ses propres valeurs.

nJam possède deux sorties, une qui renvoie un mixage de flux audio reçus des différentes instances de nJam (la sortie de gauche). L'autre envoie en signal périodique correspondant au métronome réparti.

7.3.1 L'écoute mutuelle entre musiciens

L'écoute mutuelle entre musiciens est le premier simulacre sensoriel à introduire dans la conception d'un système de Performance Musicale Interactive et Distribuée. Dans le but d'augmenter l'interactivité musicale, nous avons choisi de faire en sorte que les musiciens puissent

[46]"Jam" est un terme couramment utilisé par les musiciens pour désigner une rencontre ponctuelle dans laquelle ils jouent ensemble leur répertoire commun

interagir à travers la pratique instrumentale. De la sorte, le musicien utilise son expérience musicale, acquise suite à de longues heures d'entraînement.

Pour cela, le système doit effectuer une acquisition du son ambiant. Au paragraphe 3.2, nous avons identifié deux types de codages de la musique : le MIDI et le MIC. Le codage MIC est beaucoup plus fidèle à la restitution que le MIDI, puisqu'il permet de numériser tous les sons perçus par l'oreille humaine. Nous avons vu que la qualité du codage MIC dépend de deux paramètres : la fréquence d'échantillonnage et la quantification. Ainsi, pour une fréquence de 44100Hz avec une quantification sur 16 bits, la restitution d'un son est considérée comme fidèle au son d'origine pour une oreille humaine.

Ces paramètres d'échantillonnage du MIC sont utilisés dans le codage des CD audio, ainsi que par toutes les cartes sons. En conséquence, le MIC est le format audio utilisé par jMax et donc le format utilisé par les objets de traitement du signal. Selon la carte son, jMax propose plusieurs types de configurations des paramètres d'échantillonnage, dont la configuration à 44100Hz 16bits.

L'utilisation d'une horloge (caractérisée par la fréquence d'échantillonnage) introduit une contrainte temps réel durant la reconstitution du signal audio : après la lecture d'un échantillon, le système doit être capable de lire le suivant avant la période définie par la fréquence d'échantillonnage.

Avec nJam, les données MIC vont être transportées à travers le réseau via le protocole RTP (voir le paragraphe 2.2.2) et le Multicast IP. La qualité auditive que fournit le codage MIC impose un débit de données utiles de $44100 * 16 = 705600$ bit/s (soit 0,7 Mb/s) par canal. De plus jMax permet de faire communiquer les objets avec un flux audio à plusieurs canaux. Cela ouvre la perspective d'une acquisition et d'une restitution spatialisées du son.

Ainsi, une partie de nJam est un moteur de streaming. Pour cela nJam utilise la librairie RTP développée à l'UCL[47]. Bien que RTP est un protocole de session, l'interface de programmation ne permet pas de bénéficier de la gestion des utilisateurs présents et ne délivre pas les données de façon synchrone. Ainsi, nJam doit gérer les identifiants correspondant aux autres instances de nJam qui communiquent ainsi qu'un ensemble de tampons de réception.

Les tampons de nJam ont donc pour fonction principale de lisser la gigue du réseau. Ils fonctionnent comme un seau percé : les données arrivent en paquets avec la gigue introduite par le réseau. Ces données sont lues pour être ensuite traitées et envoyées à la sortie de nJam. Comme le trou du seau percé qui laisse l'eau s'écouler à flot constant, ces lectures doivent s'effectuer de façon régulière (c'est donc l'horloge de la carte son qui va cadencer l'exécution des objets de jMax). Le succès de ce mécanisme est fonction de la quantité d'eau qui est placée au départ dans le seau, et donc de l'indice de départ du tampon. Cet indice correspond à la gigue maximale que le mécanisme va "amortir"[48]. Cette valeur entre donc dans la chaîne des latences ajoutées par le système.

La librairie RTP fournit une interface dans le langage C, qui est le langage utilisé pour programmer l'objet nJam. La récupération des événements se fait grâce à des *handler*. Grâce au champ *ssrc* de RTP, nJam peut router les données dans le tampon correspondant. Ce mécanisme n'est pas cadencé par l'horloge de la carte son. Il est programmé dans un thread parallèle au

[47] UCL Common Multimedia Library,
http ://www-mice.cs.ucl.ac.uk/multimedia/software/common/

[48] La métaphore du seau percé est pédagogique, mais inexacte en réalité, puisque la vitesse d'écoulement est proportionnelle au remplissage du seau

processus *FTS* (le moteur de calcul de jMax).

C'est la fonction de calcul de nJam qui va lire les tampons, restituant alors l'isochronisme des flux audio. Ces flux seront mixés par une addition des échantillons provenant de chaque tampon puis restitués à la sortie de nJam (dans le patch de figure 7.2, la sortie de nJam est connectée directement au périphérique de sortie de la carte son).

Voici donc comment nJam utilise le protocole RTP :

- Dans la mesure où le prototype doit montrer la faisabilité de la conception d'un système de Performances Musicales Interactives Distribuées, nous n'avons cependant pas mis en œuvre de système de mise en relation des utilisateurs : l'adresse IP utilisée pour virtualiser le groupe est définie à l'avance. Nous n'avons jamais rencontré de problèmes de collision d'adresses durant nos tests
- Le format MIC utilisé correspond au format RTP appelé L16, utilisant le numéro de payload 11 (transmission mono-canal). Cependant, nJam a besoin de synchroniser les flux audio pour fournir un tempo commun. L'interopérabilité avec une autre application n'est alors pas primordiale, puisque celle-ci ne sera pas capable de synchroniser les flux
- Le champ *timestamp* de RTP est déterminé grâce à l'horloge de la carte son, en inscrivant la date du premier échantillon contenu dans le paquet. Ainsi, les horloges ne sont pas synchronisées de façon absolue, mais possèdent des fréquences proches. Nous reviendrons plus en détail sur les dérives des horloges dans le paragraphe 9.3.

Dans nJam, nous avons choisi d'utiliser les technologies qui permettent de fournir aux musiciens un système simulant au mieux la qualité d'écoute mutuelle. La technologie multicast permet d'être entendu par tous, la technique du seau percé couplée au codage MIC permet de restituer fidèlement le son et la technologie RTP permet de transporter les données de bout en bout dans le même flux.

Cependant, ces choix rendent l'application utilisable uniquement dans certaines conditions : le réseau utilisé doit supporter le multicast IP et supporter un débit relativement important. Durant nos tests et démonstrations, nous nous sommes efforcés de satisfaire ces contraintes, mais celles-ci rendent actuellement impossible l'utilisation de nJam sur la majorité des réseaux ADSL.

Comme nous l'avons vu au paragraphe 3.3.2, les ambiguïtés rythmiques dues à la latence de bout en bout font que l'écoute mutuelle seule n'est pas suffisante pour obtenir l'interactivité musicale. Nous allons maintenant décrire comment nJam fournit un tempo global aux musiciens, en utilisant le protocole de cohérence perceptive décrit au paragraphe 6.2.

7.3.2 Intégration du protocole de cohérence perceptive

Grâce au protocole de cohérence perceptive, les instances de nJam vont construire une référence de temps commune aux musiciens, qui leur permet de s'entendre tous avec un mixage identique. En effet, la propriété de simultanéité fait que si l'un des musiciens entend deux sons simultanément (sur un temps du tempo par exemple), alors tous les autres les entendront simultanément.

Le système d'écoute mutuelle fournit une communication isochrone entre les instances nJam, avec une latence constante de bout en bout, entre chaque instance de nJam. La propriété de Δ légalité est donc atteinte (voir paragraphe 4.1). Ainsi, les vecteurs locaux pourront être déterminés

sans utiliser la procédure de "calcul des vecteurs locaux" (figure 6.1).

Pour calculer les vecteurs locaux (les écarts relatifs entre horloges) automatiquement, chaque instance de nJam attend de recevoir des données de toutes les autres instances. Lors du mixage des échantillons audio, i.e. au moment auquel les données sont restituées avec une latence constante, il est facile d'enregistrer un vecteur contenant les dates des échantillons joués simultanément (un par instance).

Ce vecteur est alors équivalent à celui décrit dans la procédure de "calcul des vecteurs locaux", dans la mesure où les dates écrites dedans représentent les écarts relatifs des horloges distantes avec la date d'origine de l'horloge locale.

C'est donc ce vecteur qui va être diffusé par chaque instance de nJam aux autres. Pour cela, nJam utilise le message de type "application" (*APP*) du protocole RTCP, permettant d'utiliser un canal de communication déjà ouvert parmi les instances de nJam.

Après réception de chacun des vecteurs, la procédure de "calcul des délais locaux" (figure 6.2) est utilisée par chaque instance de nJam pour calculer le nombre d'échantillons à ajouter dans chacun des tampons pour obtenir la propriété de simultanéité. Une fois ces échantillons artificiels ajoutés, nJam va continuer la procédure de mixage des échantillons lus dans les différents tampons, maintenant ainsi les propriétés de simultanéité et de Δ légalité.

Bien que la cohérence perceptive soit nécessaire, elle n'est pas suffisante pour permettre une interactivité conviviale parmi les utilisateurs. En effet, comme nous allons le voir, l'introduction d'un retard local qui n'est pas dans le rythme du morceau provoque l'écoute simultanée de deux flux audio qui ne sont pas en rythme : le flux de sons joués par le musicien lui-même et celui issu du mixage des flux effectué par nJam.

Il est alors nécessaire de retarder l'ensemble des flux d'un délai permettant d'avoir un retard local adapté au tempo de la musique jouée. Nous allons maintenant expliquer comment nJam procède.

7.4 Adaptation du délai local au rythme du morceau

La somme des latences introduites de bout en bout dépasse généralement les 50 millisecondes acceptables par les musiciens, empêchant ceux-ci d'avoir un tempo commun (voir paragraphe 3.3.2 et 4.3.1).

Cependant l'algorithme de calcul des délais locaux a introduit des retards localement et pour chaque flux reçu, en fonction des latences qui ont été calculées pour maintenir une délivrance régulière des données. Cette latence peut donc être exprimée en millisecondes, en nombre d'échantillons, mais pas dans une unité de mesure propre à la musique. En effet, la musique possède sa propre échelle temporelle : le tempo. Le découpage du temps physique est ainsi exprimé en BPM (Battements Par Minute) et la structure musicale est découpée en mesures, dont le nombre de temps qui la compose est défini pas la métrique[49].

L'utilisation directe du système avec de tels retards risque de gêner les musiciens dans l'exécution d'un morceau, puisque les sons entendus par les musiciens sont en déphasage avec la musique qu'ils jouent directement. La figure 7.3 illustre ce problème : le mixage de la musique

[49]La métrique d'un morceau correspond au nombre de temps que compose une mesure. Par exemple une valse est composée de mesures à trois temps. Le rock utilise généralement des mesures à quatre temps.

jouée par Alice et Bernard est restitué de façon identique, avec un tempo partagé par les deux musiciens. Cependant les lignes pointillés montrent que le battement du rythme de la musique n'est pas en phase pour les deux flux. Ces déphasages correspondent au délai de bout en bout (d^{algo}_{Alice} et $d^{algo}_{Bernard}$), après l'établissement de la cohérence perceptive.

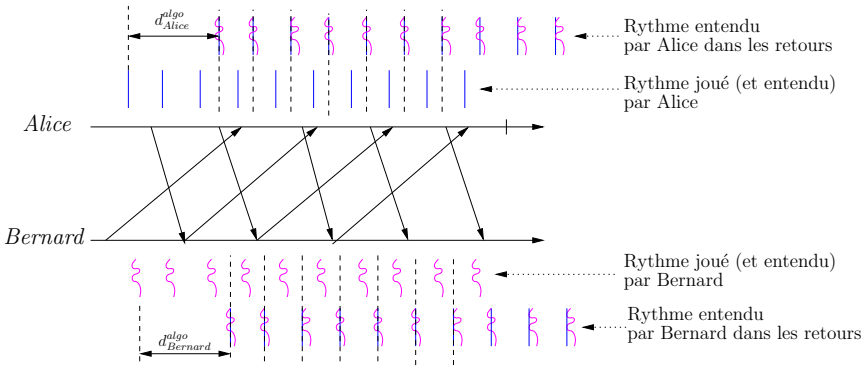


FIG. 7.3 – *L'application directe de la cohérence perceptive, un problème de déphasage*

Ce décalage de tempo est extrêmement gênant pour les musiciens car il provoque des erreurs rythmiques. Pour éviter cela, nous proposons d'exagérer la latence introduite dans la restitution des flux déjà synchronisés pour les mettre en phase avec le tempo d'entrée. Pour cela, les musiciens doivent spécifier le tempo dans lequel ils vont devoir jouer, ainsi que la durée du retard (exprimée en noires, en blanches, en croches, etc). La figure 7.4 montre alors la gestion des décalages introduits. Alice et Bernard vont entendre la musique synchronisée avec un retard plus important, mais qui aura un tempo en phase avec le tempo de la musique jouée.

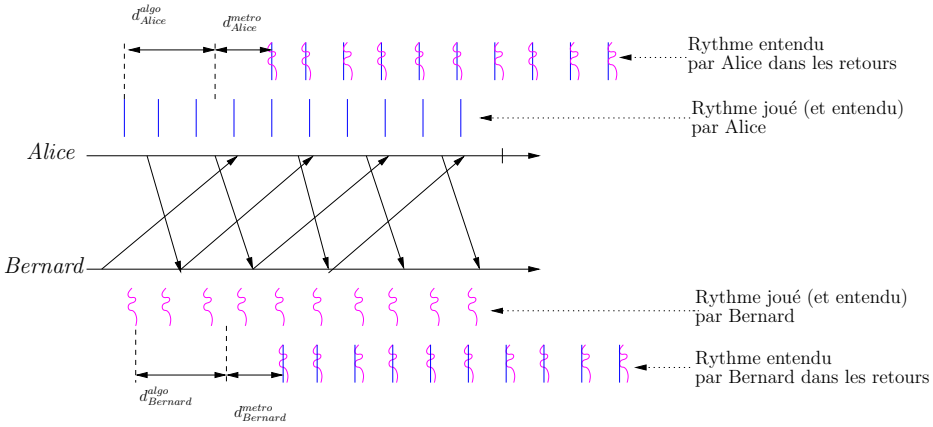


FIG. 7.4 – *La solution de gestion des latences*

Cette solution impose des contraintes aux musiciens puisqu'ils doivent gérer l'écoute de deux sources audio distinctes. Durant cette écoute, les notes des deux flux vont se superposer, introduisant alors d'éventuelles dissonances dans la musique entendue. Pour éviter ce phénomène, il suffit d'introduire un retard de durée égale à la durée d'une boucle harmonique, qui sera répétée à l'infini[50]. Nous reviendrons sur l'aspect contraignant de ce système dans le chapitre sur les

[50]Ce principe est proche du système VirJa qui est décrit au paragraphe 3.4.1 et qui introduit un retard de 12

démonstrations et les tests du prototype.

Remarquons que même s'il y a dissonances durant l'écoute, elles sont généralement moins gênantes pour un musicien que les ambiguïtés rythmiques. En effet, dans plusieurs types de musiques, les dissonances sont utilisées consciemment. Par exemple la musique du Jazzman Ornette Coleman était, dans les années 50 et 60 basée sur ce principe : tout en conservant un rythme commun, la rythmique et le soliste jouaient dans des harmonies différentes[51]. Ce style a largement été intégré par les Jazzmen depuis.

Dans certains styles musicaux, les dissonances sont introduites par les instruments eux mêmes. Par exemple, un Gamelan est composé d'un ensemble d'instruments qui sont fabriqués au même moment et à partir du même métal (des métalophones). Dans la mesure où le métal se dilate avec la chaleur et l'humidité, il est impossible d'accorder les instruments de l'ensemble, créant un spectre harmonique dissonant et unique à chaque ensemble de métalophones [Bal94].

Nous allons donc nous intéresser au rythme, car beaucoup de styles musicaux sont basés sur des rythmes spécifiques qui, à une certaine échelle, bouclent. Ces rythmes sont souvent appelés des patterns (motifs) dans lesquels les musiciens peuvent trouver une infinité de variations. Ces patterns sont caractéristiques et permettent de reconnaître à l'écoute un style, un compositeur ou un musicien très rapidement. Par exemple, il est facile pour un auditeur non musicien de différencier un morceau de Bossa Nova du Brésil d'un morceau de Georges Brassens, malgré le fait que ces deux musiques utilisent principalement des guitares acoustiques.

Le caractère cyclique de certains rythmes peut donc être utilisé pour superposer les flux audio, à la manière d'un canon. C'est pour cette raison que dans nJam, c'est le musicien qui va configurer son propre retard, en spécifiant la durée de la boucle. Dans la figure 7.2, le musicien va configurer le retard à l'aide du message suivant :

```
start 180 12 1 3
```

Le premier paramètre exprime le tempo du morceau en BPM (Battements par Minute). Le deuxième permet de choisir le décalage en nombre de battements (en noires). Les deux suivants servent à la configuration du métronome, nous reviendrons dessus au prochain paragraphe. Dans l'exemple, le décalage local à introduire est donc de $(12/180) * 60 = 4$ secondes. Ce délai peut être obtenu raisonnablement par le système. Si le délai demandé est inférieur au délai introduit par le réseau, alors un message d'erreur s'affiche.

Pour introduire le décalage demandé et conserver la simultanéité, nJam va insérer le même nombre d'échantillons audio dans chaque tampon. Pour calculer ce nombre, nJam va regarder le délai local qui a déjà été ajouté par le protocole de cohérence perceptive. Ce délai local est obtenu en mesurant à un instant donné la différence entre la date courante et la date de l'échantillon issu du flux local qui est mixé par la fonction de traitement audio de nJam. Ensuite nJam calcule la différence entre le délai demandé et le délai local, puis ajoute le nombre d'échantillons correspondants dans les tampons de réception des flux.

Grâce à cette méthode, nJam permet aux musiciens de pouvoir écouter la musique produite par le système, y compris la musique qu'il joue lui-même, sans ressentir de gêne rythmique.

mesures pour un blues

[51] Une description intuitive du style harmonique d'Ornette Coleman est effectuée à la fin de l'ouvrage de George Russel [Rus59]

Cependant, le décalage peut être important, comme le montre l'exemple que nous avons choisi, dans lequel le décalage correspond à une durée de 4 secondes. Le musicien doit alors à la fois anticiper le retard et rester dans le rythme.

Pour aider les musiciens à anticiper ce retard (en fait à rester en rythme), nJam fournit un métronome global qui s'appuie sur la propriété de simultanéité.

7.5 Le métronome réparti

La fonction de mixage introduit une propriété qui va être utilisée pour construire le métronome : le mixage est une addition des échantillons provenant des différentes instances de nJam qui communiquent, pour ensuite envoyer ce mixage à la carte son. Les échantillons mixés ensemble possèdent donc la même date de restitution.

Ainsi, le délai entre la restitution de deux échantillons mixés est nul. Avec la propriété de simultanéité, nous pouvons dire que si deux échantillons ont été mixés ensemble par une instance de nJam, alors ils le sont aussi par les autres instances.

A l'aide de cette propriété, il est possible de construire un métronome global. Une solution consiste à faire en sorte que chaque instance de nJam choisisse comme référence temporelle les dates de restitution des données provenant de l'instance de nJam choisie comme référence dans la procédure de calcul des délais locaux (voir paragraphe 6.2). Appelons l'instance référence ref.

Sachant que l'unité temporelle est l'échantillon, alors il y aura un battement envoyé à la sortie "métronome" de nJam, lorsque dans la fonction de mixage de nJam, l'échantillon u est mixé et que nous avons l'égalité suivante (suivant la notation introduite au paragraphe 4.2) :

$$t_{ref}(w_{ref}(x_{ref})u) \ modulo \ \delta_{metro} = 0$$

Sachant que δ_{metro} est la durée entre deux battements du métronome, dans l'unité de temps. Avant d'expliquer comment calculer cette valeur, remarquons que ce sont les dates des écritures effectuées par le processus référence qui permettent aux instances de décider d'un instant commun.

Pour calculer la durée entre deux battements, nJam utilise la configuration donnée par les musiciens. Reprenons le message de configuration utilisé dans le patch jMax :

```
start 180 12 1 3
```

Rappelons que le premier paramètre est le tempo du morceau. Le troisième paramètre permet de choisir l'écart temporel entre deux battements de métronome : la valeur 1 signifie que tous les temps du tempo seront battus par le métronome, une valeur de deux signifie "seulement un temps sur deux". Le quatrième paramètre indique la métrique du morceau, qui sera utilisée pour afficher un entier à la sortie métronome, correspondant à la position du courante du morceau dans la mesure.

Dans l'exemple que nous avons donné, l'écart entre deux battements de métronome est donc de $(1/180) * 60 = 0,3$ secondes. Nous verrons dans le chapitre suivant que ce métronome est très utile aux musiciens, notamment pour démarrer un morceau. Comme nous le verrons au paragraphe 8.3, l'utilité de métronome est moins importante durant le morceau.

7.6 Conclusion

Pour résumer le principe d'utilisation de nJam, voici succinctement les manipulations effectuées par chaque musicien :

1) Le musicien envoie le message "`connect n`" qui permet d'initialiser l'écoute mutuelle
2) Il envoie un message de configuration du métronome
3) nJam effectue le calcul des délais locaux
4) Un message informe le musicien que les flux sont synchronisés
5) Le métronome s'active, en fonction des dates du flux d'échantillons issus du processus référence
6) Le morceau peut commencer

L'ensemble des procédures de nJam (l'écoute cohérente, le décalage local rythmé et le métronome) est effectué durant l'initialisation. De la sorte, durant l'exécution du morceau, le seul travail que nJam doit effectuer est de maintenir le moteur de streaming, avec des latences de bout en bout déterminées par le protocole de cohérence perceptive, et par les paramètres donnés par les musiciens pour configurer le métronome.

Comme nous l'avons vu dans l'état de l'art du chapitre 3, les systèmes de Performances Musicales Interactives et Distribuées (PMID) comme nJam nécessitent la mise en œuvre d'un ensemble de simulacres de différentes natures. Dans nJam, les simulacres développés sont d'ordre auditif, car principalement basés sur la synchronisation des flux audio.

Ainsi, nJam est à notre connaissance le seul système de PMID qui intègre à la fois un streaming de données MIC de qualité CD, avec un système de synchronisation basé sur la notion de simultanéité. Bien que ces fonctionnalités soient jugées comme nécessaires par un ensemble de publications scientifiques [CSTZ05, Sch02, GN02, GDNW04], celles que nous proposons dans nJam n'ont jamais été mises en œuvre par d'autre(s) projet(s) en conditions de concert. Au prochain chapitre, nous allons présenter les tests que nous avons effectués à travers deux démonstrations publiques et un test effectué à l'aide de musiciens. Nous y montrerons la validité pratique des mécanismes présentés dans ce chapitre.

8

Les démonstrations et tests

Sommaire

8.1 Introduction

Nous avons expliqué le fonctionnement de nJam au chapitre précédent, en se focalisant principalement sur la gestion des délais du réseau. Ainsi, nJam propose de faire interagir des musiciens avec des latences pouvant aller jusqu'à plusieurs secondes, ce qui va à l'encontre de deux études utilisateurs du domaine des PMID, qui fixent la latence maximale acceptable par des musiciens à 65 ms [CSTZ05]. Dans ce chapitre, à travers deux expérimentations publiques effectuées avec nJam et une étude d'usage, nous montrons que l'interaction est possible avec des latences telles que celles proposées avec nJam, élargissant ainsi les possibilités de déploiement des systèmes de PMID à des réseaux à latences importantes, comme l'ADSL grand public.

Ainsi, dans ce chapitre, nous allons répondre aux questions suivantes :
- Les simulacres proposés par nJam sont-ils suffisamment conviviaux pour permettre à des musiciens de jouer ensemble facilement ?
- Est-il possible d'avoir une interaction musicale avec des latences entre musiciens allant jusqu'à plusieurs secondes ?
- Quels sont les simulacres à développer dans le système pour rendre l'application plus conviviale ?

Pour répondre à ces questions, nous présentons l'installation du système et des musiciens en conditions réelles. Nous allons donc présenter deux démonstrations qui ont été montées avec nJam. La première (paragraphe 8.2) a été montée pour la semaine "portes ouvertes" (*Résonances 2003*) de l'IRCAM en Octobre 2003. Cette démonstration nous a montré la complexité technique que représente l'organisation d'une telle expérience. En effet, la démonstration s'est répartie sur deux sites distincts : le CNAM Paris, situé rue Turbigo dans le troisième arrondissement de Paris, et l'IRCAM, situé place Igor Stravinsky dans le quatrième arrondissement. La deuxième démonstration (paragraphe 8.3) a été montée entièrement au CNAM entre le laboratoire CEDRIC et le musée des Arts et Métiers. Au chapitre 8.4, nous allons présenter une étude d'usage de nJam dans laquelle nous montrerons que l'interactivité musicale reste confortable (aux dires des musiciens) malgré des latences équivalentes à plusieurs secondes, et cela grâce à l'utilisation de nJam et de son métronome global.

Finalement nous concluons au paragraphe 8.5 sur un bilan positif, puisque nous avons montré la faisabilité d'un tel système, ainsi que de la validité pratique des solutions présentées dans les chapitres précédents.

8.2 *Résonances 2003*, une performance musicale interactive répartie durant les portes ouvertes de l'IRCAM

L'IRCAM et le CNAM sont connectés à Internet via deux fournisseurs d'accès différents (voir figure 8.1) mais interconnectés entre eux via les réseaux RAP[52] pour le CNAM et RENATER[53] pour l'IRCAM.

[52] Réseau Académique Parisien :
`http://www.rap.prd.fr/actualites/index.php`

[53] Réseau National de Télécommunications pour la Technologie, l'Enseignement et la Recherche :
`http://www.renater.fr/`

Durant cette expérience, nous avons planifié l'utilisation de trois salles différentes qui hébergeaient un ou deux musiciens chacunes (symbolisées par des machines sur la figure 8.1). Ainsi, l'unique salle située au CNAM a hébergé Sylvain Choinier (guitare) et Rafaël Quenehen (saxophone). Deux salles ont été prévues à l'IRCAM : une était destinée à accueillir le public et Jean Lochard (électronique temps réel), l'autre à accueillir Clément Lebrun (basse) et Julien Bloit (batterie). Durant la première répétition, malgré la mise en place technique de trois instances de nJam, nous n'avons fait interagir uniquement deux salles (celles hébergeant le bassiste, le guitariste, le saxophoniste et le batteur). En effet, nous avions prévu d'effectuer d'abord les tests techniques (réseau, balance et synchronisation), et seulement ensuite une interaction à trois sites.

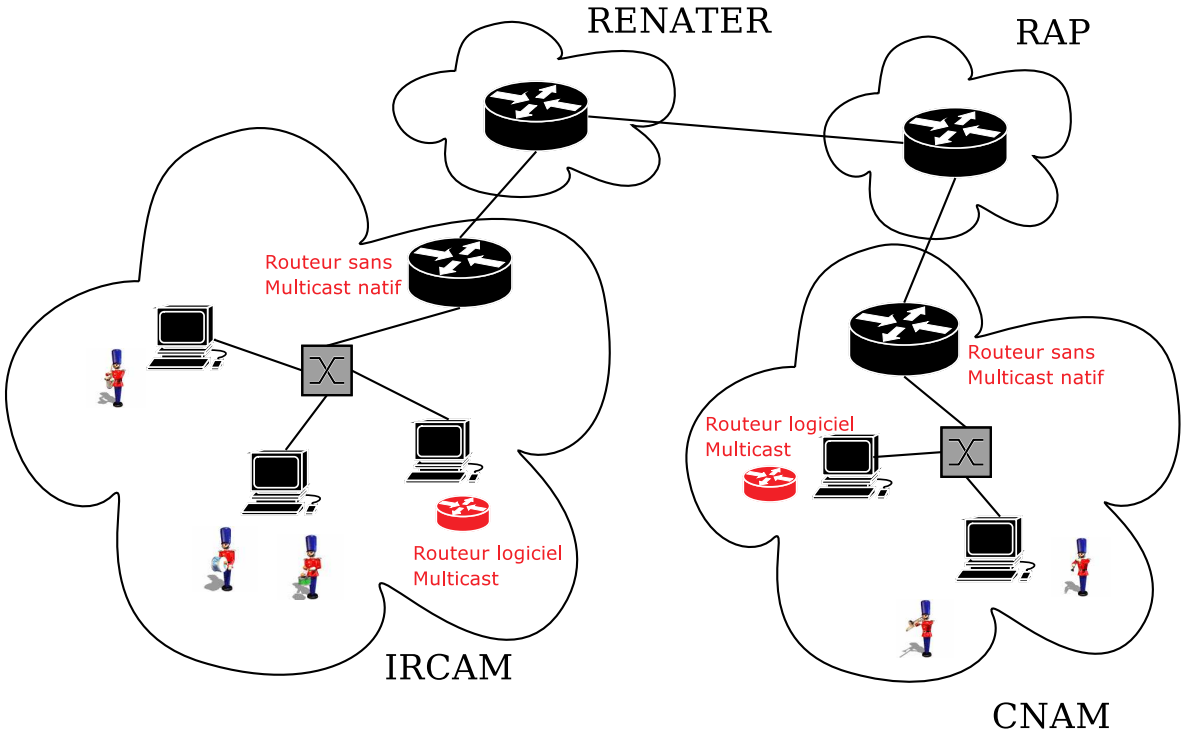


FIG. 8.1 – *L'infrastructure réseau durant résonances 2003*

Nous expliquerons au chapitre 9.2.1 le principe de tunnelisation que nous avons utilisé pour que nJam puisse disposer du Multicast IP. Nous présenterons aussi les modifications mineures que nous avons apportées au prototype nJam pour que celui-ci soit compatible à l'architecture réseau utilisée.

8.2.1 L'installation des musiciens

La configuration répartie des périphériques audio

Au paragraphe 7.3, nous avons donné un exemple simple d'utilisation de nJam. Cependant, lors de la mise en œuvre nous avons dû ajouter d'autres objets dans le patch jMax pour effectuer les fonctions suivantes :

- Régler le volume d'entrée
- Régler le volume de sortie

- Convertir le signal logique de la sortie métronome de nJam en un son de métronome (un clic bref)
- Régler le volume du métronome.

Ces réglages, lorsqu'ils doivent être effectués de façon répartie, sont très compliqués à faire puisque la modification de la valeur du volume de l'entrée locale va avoir un impact sur le volume de sortie des autres instances de nJam. Nous avions prévu alors d'utiliser une table de mixage virtuelle, qui permet à un ingénieur du son de contrôler de façon centralisée tous ces paramètres. Pour effectuer ce contrôle, Hans-Nikolas Locher a développé un objet jMax pouvant être contrôlé par une console distante, à l'aide de l'outil OpenTaz, une implémentation de la norme IEC TASE.2 [Gro96]. Bien qu'un premier prototype fut développé et testé [LBB+03, Loc03], nous n'avons pas pu développer à temps de version compatible avec la version de 4 de jMax.

Ainsi, lors de nos tests, les réglages de volume ont fait l'objet de longues minutes de négociations téléphoniques, montrant qu'un tel outil aurait été précieux.

La configuration locale des périphériques audio

Tout comme la configuration répartie, la configuration locale fut compliquée. En effet, le système doit gérer un ensemble de flux audio dans la pièce : les flux audio en entrée (si plusieurs musiciens sont présents devant la machine) et les flux audio en sortie. Par exemple, si l'acquisition du son joué par le musicien se fait par un micro (comme c'est le cas pour les instrument acoustiques), alors il faut éviter de restituer le son sortant de nJam avec des enceintes, au risque de faire ré-entrer le son retardé dans le micro.

Nous avons rencontré ce problème durant la première répétition, il peut être entendu dans l'enregistrement disponible à l'adresse suivante :

`http://cedric.cnam.fr/~bouill_n/MP3/CVRresonances03.mp3`

Une solution simple à ce problème est l'utilisation d'un casque. En effet, cela permet d'isoler complètement les flux audio dans la pièce. La figure 8.2 montre les musiciens, lorsqu'ils sont installés devant le système[54].

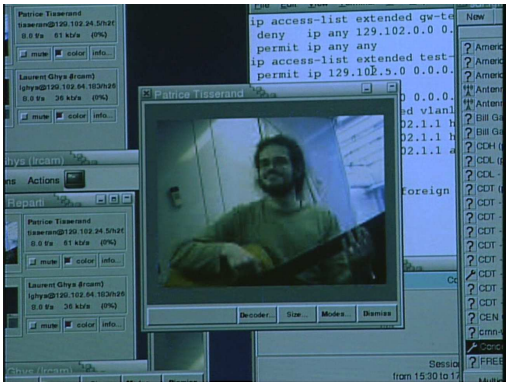
Un autre problème que pose cette présence de flux audio distincts dans la même pièce est que cela complique la configuration des volumes. En effet, si deux musiciens utilisent la même machine, alors il faut être capable de mixer le son des instruments et envoyer le résultat à nJam. Cette configuration nécessite l'utilisation d'une table de mixage, dont la configuration est décrite par la figure 8.3.

Cette configuration est complexe puisqu'elle nécessite que chaque musicien puisse entendre plusieurs flux audio :

- Le son qu'il produit lui même
- Le son produit directement par l'autre musicien présent dans la salle
- Le son provenant des autres musiciens distants
- Le son du métronome

[54]Les photos présentées sont extraites d'un reportage [GIP04] commandé par Renater auquel nous avons participé. Pour la transmission vidéo, nous avons utilisé VIC [HPH03], un logiciel de vidéo conférence utilisant le format de bas débit h.261 spécifié par l'ITU.

(A) L'installation d'un musicien avec nJam



(B) Le musicien distant à l'écran

FIG. 8.2 – *Les musiciens distants*

Notons qu'il est important de permettre à deux musiciens situés dans la même pièce de s'écouter directement, car ils peuvent éventuellement se coordonner instantanément, durant l'exécution du morceau. Cette configuration dépend en fait du choix des musiciens.

8.2.2 La première répétition

L'enregistrement effectué durant les répétitions montre la faisabilité du système. Tout d'abord, les musiciens arrivent à jouer un blues à 120 BPM avec une latence correspondant à une mesure à quatre temps (soit deux secondes) en suivant les changements harmoniques[55]. Ce résultat montre que le système permet aux musiciens d'utiliser leurs connaissances musicales pour s'adapter au système.

En somme, en quelques essais les musiciens ont réussi à jouer un morceau relativement complexe malgré une latence importante et cela grâce au protocole de cohérence perceptive et au métronome. Nous voyons alors que nJam permet aux musiciens de tolérer des latences beaucoup plus importantes que celles mesurées dans la littérature (environ 65 ms, nous proposons une étude plus détaillée au paragraphe 8.4).

Un autre résultat intéressant est qu'ils ont réussi à effectuer un départ synchronisé du morceau. Pour cela, il suffit à l'un des musiciens de compter "1 2 3 4" en rythme avec le métronome. Si tous les musiciens partent après avoir entendu le compte à rebours dans les retours, alors ils vont tous démarrer ensemble, au même instant.

La première répétition a donc apporté des résultats positifs, malgré les difficultés rencontrées pendant la configuration de l'application et le nouvel environnement auquel les musiciens ont du s'adapter. Durant la deuxième répétition et la démonstration, le réseau a joué en notre défaveur, puisque la gigue et les pertes sont devenues beaucoup plus importantes. Ce phénomène a eu un impact important sur nJam, notamment sur le système de gestion des tampons. Le système n'étant plus fonctionnel, nous avons du annuler la démonstration. Nous reviendrons sur le pro-

[55] Un changement harmonique correspond au passage d'un accord à un autre dans le morceau. Lors de ce changement, le soliste doit généralement utiliser d'autres notes pour rester "juste".

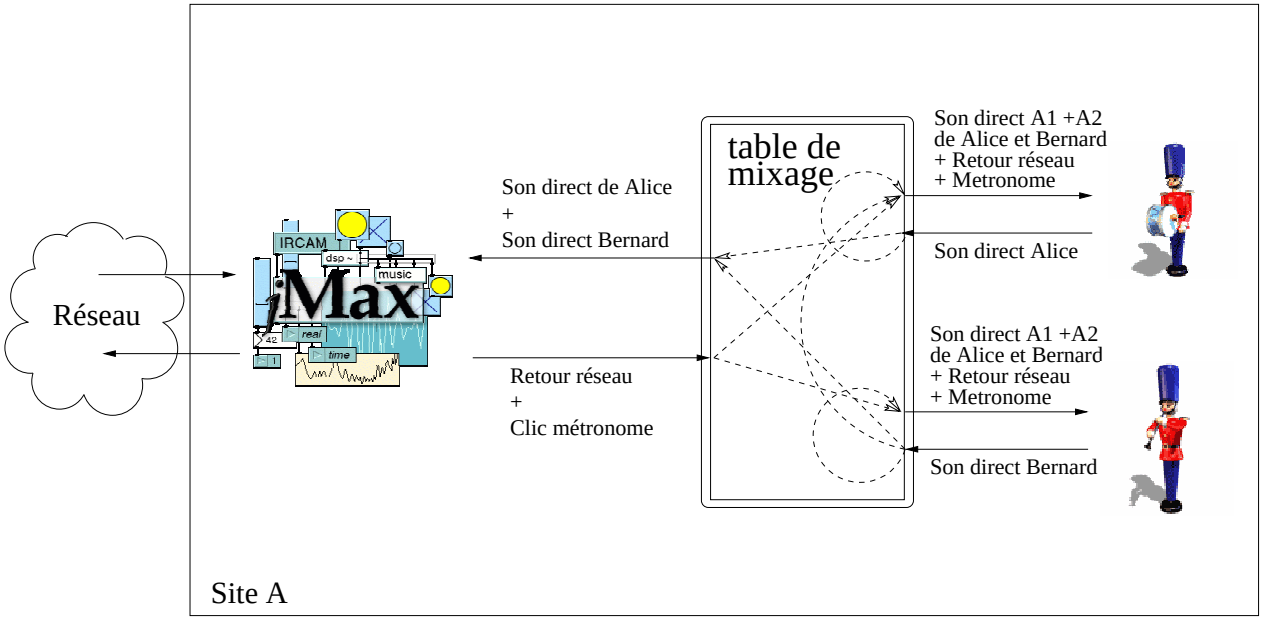


FIG. 8.3 – *L'installation des périphériques audio*

blème de gigue au paragraphe 9.3.

8.2.3 Conclusion sur l'expérience

Malgré l'annulation de la démonstration devant le public, la première répétition nous a permis d'obtenir des résultats intéressants sur l'interactivité entre les musiciens : grâce au métronome réparti et au protocole de cohérence perceptive, les musiciens ont pu jouer un morceau avec une latence d'une mesure (ce qui correspond à 2 secondes) avec un départ et une fin coordonnés. De plus, malgré les difficultés rencontrées et le peu de pratique, les mêmes musiciens n'ont mis que quelques minutes pour jouer un morceau différent lors d'un test que nous avons effectué quelques mois plus tard.

Du point de vue de l'organisation, nous avons vu qu'il était très complexe de configurer et de déployer le système. En effet, la partie réseau comme la partie audio demandent des connaissances techniques spécialisées dans chaque domaine de l'application.

8.3 *Il jouait du piano à distance* : démonstration au CNAM durant la fête de la science 2005

L'expérience de Résonances 2003 nous a montré la faisabilité du projet, mais aussi la complexité de son élaboration pour la fête de la science 2005. Nous avons initialement projeté de monter une expérience appelée "Agir à distance, comment et jusqu'où ?" qui visait à montrer au grand public des exemples d'Applications Multimédia Interactives et Distribuées. Pour cela, nous avions planifié trois animations différentes utilisant Internet pour l'interaction :

- *Reproduire l'action d'un musicien en plusieurs lieux*, à la façon texte "une ville idéale" de Jules Verne (voir le paragraphe 3.3). Cette démonstration a pour objectif de montrer que le réseau Internet peut aujourd'hui concrétiser les rêves d'hier, et notamment ceux de Jules Verne dont 2005 a été le centenaire de la mort. Pour reproduire l'expérience décrite par Jules Verne, les pianos de type Disklavier auraient pu être utilisés car ils sont capables de jouer mécaniquement les touches du clavier à partir de données MIDI, créant l'illusion de la présence d'un instrumentiste.
- *Agir ensemble à distance*, à la façon de l'expérience que nous avons montée pour *Résonances 2003*. L'objectif de cette démonstration permet de montrer que malgré la distance, des musiciens peuvent interagir ensemble.
- *Contrôler et agir à distance avec des machines.* Cette expérience reprend des idées utilisées durant les expériences de télémécanique d'Edouard Branly (réalisées au Trocadéro à Paris en 1905). L'expérience consiste à contrôler un bras mécanique à l'aide d'une page web dans laquelle l'utilisateur peut visualiser le bras avec un flux vidéo et choisir des programmes à faire exécuter par le robot.

Avec la collaboration de Jean Vareille de l'UBO (Université de Bretagne Ouest à Brest), nous avons décidé de monter ces démonstrations avec la participation de RENATER et le Laria à Amiens avec Christophe Loge. Pour cela nous avons répondu à l'appel à projets du Ministère de la Recherche. Malheureusement, notre projet n'a pas été retenu.

L'absence de financement du projet ne nous permettait alors pas de maintenir la programmation initiale du projet. En effet, le coût de la location de pianos de type Disklavier, de l'achat de matériel spécifique, de l'organisation de réunions était alors trop importants. Nous avons alors choisi de présenter une démonstration fonctionnellement similaire à celle présentée lors de *Résonances 2003*, mais au sein d'une seule zone d'administration IP : le CNAM et le Musée des Arts et Métiers.

8.3.1 Description de l'expérience et déploiement de l'application

Compte tenu des problèmes rencontrés durant la démonstration de Résonances 2003, nous avons choisi de simplifier la démonstration et de principalement mettre en avant l'étude de l'utilisation de nJam par des musiciens. Nous avons alors choisi de répartir les musiciens sur deux sites distincts dans lesquels le public pouvait être présent. Nous reviendrons sur les aspects systèmes de la démonstration au paragraphe 9.3.

La figure 8.4 nous montre la répartition géographique de l'événement. L'amphithéâtre Abbé Grégoire et la salle de la partie "communication" du musée sont suffisamment vastes pour accueillir un public. Nous avons alors choisi de baliser le chemin entre les deux salles, afin de permettre aux publics présents de passer d'une pièce à l'autre, au milieu de la représentation.

Avec la configuration que nous avons choisi, le son doit être diffusé dans les deux pièces, à cause de la présence du public. Ainsi, pour éviter les problèmes d'écho que nous avions rencontrés lors des répétitions de Résonances, nous avons choisi de faire jouer des musiciens dont l'acquisition du son peut se passer de microphone effectuant une acquisition du son ambiant. C'est le cas par exemple des instruments électriques comme la basse, le piano ou la guitare qui possèdent un système d'acquisition intégré. Ces instruments peuvent être directement branchés sur une table de mixage ou une carte son, permettant alors d'isoler les flux. Nous avons donc

FIG. 8.4 – *La répartition des musiciens et du public durant la démonstration de la Fête de la Science*

découpé la démonstration en plusieurs morceaux joués par trois musiciens : Laurent Alibert à la guitare, Julien Cordry à la basse et moi-même à la guitare.

Bien que cette solution nécessite du matériel spécifique, comme une table de mixage pour la salle hébergeant deux musiciens ainsi que deux cartes sons professionnelles, elle évite de faire porter un casque aux musiciens, leurs permettant d'interagir en utilisant des simulacres perceptifs naturels : l'instrument de musique et le son ambiant de la pièce (voir les photos de la figure 8.5). Bien que nous ayons eu besoin d'une table de mixage pour associer deux musiciens sur une même machine, nous n'avons pas eu besoin de configurer des retours spécifiques pour chacun d'eux, comme ce fut le cas pour la démonstration de *Résonances* (voir la figure 8.3).

L'expérience de *Résonances* nous a aussi montré que l'utilisation du Multicast IP était très contraignante, et que son utilisation sur une configuration multi-site fragilise l'application si le multicast n'est pas disponible en natif. La topologie du réseau du CNAM est en étoile : un unique routeur interconnecte des sous réseaux locaux Ethernet avec RAP le fournisseur d'accès Internet du CNAM. Les administrateurs utilisent alors des VLAN[56] pour déterminer de façon modulaire l'appartenance d'une machine à un sous réseau IP.

[56]Un VLAN Ethernet est réseau LAN virtuel, qui consiste à autoriser sur un même réseau physique la réalisation de plusieurs réseaux logiques totalement indépendants les uns des autres [Ser03]. L'utilisation de VLAN nécessite l'utilisation d'équipements Ethernet compatibles à cette fonctionnalité normalisée par l'IEEE sous le nom de 802.1Q.

Nous avons ainsi, grâce au service de Direction des Systèmes d'Informations du CNAM, bénéficié d'un VLAN dédié à la performance. La technologie VLAN, comme le LAN Ethernet correspond à un domaine de broadcast : le multicast IP se transpose naturellement en message de broadcast au niveau du VLAN. Ainsi, l'expérience ne nécessitait pas la prise en charge du multicast par le routeur puisque la communication multicast est implicitement gérée par la couche VLAN. Cette solution fut efficace puisque nous n'avons pas rencontré de problèmes liés au réseau.

Avant d'effectuer les répétitions avec les musiciens, nous avons testé le matériel et le système sur un réseau local. Nous nous sommes rendu compte que le système ne pouvait fonctionner plus de deux minutes. Après quelques analyses, nous avons pu identifier la cause de ce problème : la dérive des horloges des cartes sons. Nous reviendrons sur ce problème technique et la façon dont nous l'avons résolu au paragraphe 9.3.

8.3.2 Les répétitions et la démonstration

Les répétitions ont été effectuées en réseau local durant la semaine précédant la démonstration. Pour cela, nous avons utilisé deux pièces adjacentes du laboratoire d'informatique du CNAM (le CEDRIC), un extrait vidéo de ces répétitions est disponible à l'adresse suivante :

`http://cedric.cnam.fr/~bouill_n/FDS/CVR_FDS_Rep.avi`

L'utilisation d'un réseau local durant les répétitions a permis de se coordonner plus facilement, laissant plus de temps aux musiciens pour tester l'application et de discuter de la marche à suivre durant l'exécution des morceaux. Ainsi, nous avons planifié deux morceaux :

- Un morceau improvisé de Bossa Nova à un tempo de 180 BPM et un décalage de 4 mesures à quatre temps (soit 5,3 secondes environ)
- Une valse à un tempo de 4 mesures à 3 trois temps, dans un tempo de 180 BPM (soit 4 secondes).

Nous avons choisi de jouer des morceaux dans différents styles musicaux dans le but de montrer que le principe développé dans nJam peut s'adapter à plusieurs types de rythmes. Nous avons alors remarqué que les délais choisis pouvaient être utilisés dans l'exécution des morceaux. En effet, le musicien peut utiliser le son retardé par nJam comme base rythmique, dans le but de maintenir un tempo constant. Nous avons ainsi remarqué que nous pouvions nous passer du métronome pendant le morceau, mais que celui-ci restait très utile pour commencer. En effet, le tempo joué doit être exactement celui déterminé par le décalage introduit. Ainsi, pour démarrer un morceau, le musicien écoute le métronome, commence à jouer en rythme. Lorsque la première note qu'il joue est restituée par nJam, le son du métronome est coupé, puisqu'il devient inutile.

Grâce à cette superposition de flux audio, les musiciens n'entendent qu'un seul rythme et peuvent rester concentrés sur le morceau. Ainsi, comme nous l'entendons dans la vidéo des répétitions, les musiciens peuvent effectuer des changements dans la structure des morceaux, comme changer de soliste ou changer l'harmonie en cours d'exécution (par exemple dans le morceau de funk joué pendant la répétition).

Suite au succès de ces répétitions, nous avons planifié pour la démonstration le scénario suivant :

(A) Dans le musée

(B) Dans le musée

(C) Dans l'amphithéâtre

(D) Dans l'amphithéâtre

FIG. 8.5 – *Les photos de l'installation lors de la fête de la science 2005*

1) Explication de la démonstration devant le public de l'amphithéâtre[57]
2) Le guitariste, situé à l'amphithéâtre Abbé Grégoire et le bassiste situé au musée jouent la valse ensemble
3) Les deux musiciens ainsi que le public changent de salle, en suivant le chemin indiqué à la figure 8.4
4) Pour le dernier morceau, les musiciens jouent tous les trois la Bossa Nova, avec un guitariste et un bassiste dans l'amphithéâtre et l'autre guitariste dans le musée.

8.3.3 Conclusion sur l'expérience

Les photos présentées à la figure 8.5 montrent l'installation durant la démonstration. Comme pour *Résonances*, nous avons utilisé le téléphone pour effectuer la configuration répartie (photos 8.5(c) et 8.5(d). Cette coordination nous a surtout été utile pour lancer l'application. Cette tâche

[57]La vidéo est disponible à l'URL suivante :
`http://cedric.cnam.fr/~bouill_n/FDS/CVR_FDS_Speech.avi`

pourrait à l'avenir être supportée par un système de mise en relation des utilisateurs, ou par un système similaire à celui développé à partir de la norme TASE 2 [Gro96] et décrit dans [LBB+03].

Sur les vidéos comme sur les photos 8.5(a) et 8.5(c), nous constatons aussi que les musiciens utilisent très peu la vidéo-conférence durant l'exécution du morceau. Les vidéo ne sont pas synchronisée avec les flux audio de nJam : nous l'avons utilisée uniquement dans le but d'avoir un aperçu sur la disponibilité du joueur distant. De plus, cela a permis au public d'identifier le ou les musiciens distant(s).

Le résultat le plus intéressant pour l'analyse de l'interactivité musicale est l'utilisation nouvelle du métronome. Celui-ci a servi uniquement à donner le tempo servant à démarrer les morceaux (voir les vidéos). Ensuite, la configuration des retards sur plusieurs mesures a permis aux musiciens d'utiliser des boucles harmoniques plus longues. Ce confort leur a permis d'introduire des changements dans la structure des morceaux. Par exemple en inversant de soliste dans le morceau de Bossa Nova, ou en modifiant l'harmonie (d'un demi-ton) dans le morceau de funk des répétitions.

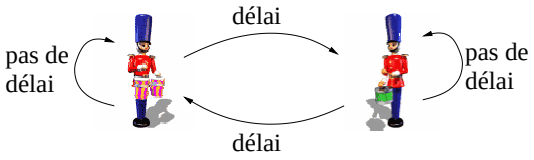
8.4 Expérimentation de nJam par des utilisateurs

Les événements *Résonances 2003* et la *fête de la science 2005* nous ont montré que les musiciens pouvaient interagir à travers un réseau avec des latences importantes et avec très peu d'entraînement. Contrairement à la plupart des systèmes de PMID, nJam ne va pas tenter d'obtenir une latence inaudible, ou suffisamment faible pour permettre aux musiciens d'interagir ensemble, mais proposer des latences importantes (de l'ordre de la seconde). Toutefois, quelque soit l'approche choisie, la latence va altérer la perception des musiciens et modifier le sentiment de jeu en groupe, d'interprétation et de création. Il est alors nécessaire de confronter les systèmes de PMID à l'opinion d'utilisateurs. De telles études ont été proposées dans la littérature [Sch02, CSTZ05, CZS+05]. Cependant, elles n'évaluent pas les conditions d'interactivité fournies par nJam puisqu'elles confrontent les musiciens à des latences faibles (jusqu'à environs 150 ms), dans le but de déterminer la borne maximale à laquelle l'interaction devient impossible. Nous avons alors mené notre propre étude sur le confort de jeu lorsque les musiciens sont soumis à des latences beaucoup plus importantes. Nous allons ainsi montrer dans quelle mesure ces latences peuvent être utilisées pour permettre l'interaction. Plus particulièrement, nous allons montrer que si la latence est choisie en fonction de la musique jouée (rythme et harmonie), alors l'interaction est possible et conserve un confort de jeu suffisant pour les musiciens.

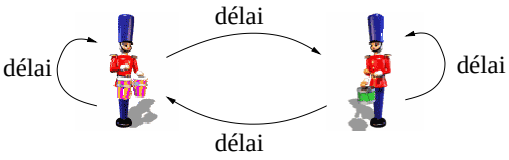
L'étude menée par N. Schuett [Sch02] a utilisé une configuration à deux musiciens situés dans deux pièces différentes. La latence entre les musiciens fût progressivement augmentée à l'aide d'une table de mixage digitale. La figure 8.6(a) montre la configuration audio utilisée durant cette étude : la variation de latence est effectuée uniquement entre les deux musiciens, sans modifier le retour local. La latence maximum acceptable est mesurée à l'aide d'un programme d'analyse de tempo appliqué sur les enregistrements des musiciens exécutant un court motif rythmique imposé. Voici les différents résultats obtenus :

- le tempo général a tendance à ralentir à partir de 30 ms
- avec une stratégie de suiveur/meneur, la limite s'établie aux environs de 50/70 ms.

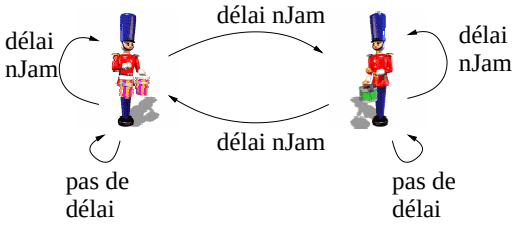
Cette étude, malgré l'intérêt qu'elle a pour le domaine des PMID, ne peut définir des seuils de



(A) La configuration de N. Schuett [Sch02]



(B) La configuration de E. Chew [CSTZ05]



(C) La configuration avec nJam

FIG. 8.6 – *La configuration audio des études utilisateurs pour PMID*

jouabilité pour nJam car les configurations audio dans lesquelles se trouvent les musiciens ne correspondent pas (voir figure 8.6) : sans retour local retardé, les musiciens sont immergés dans une configuration générant les ambiguïtés temporelles décrites au paragraphe 3.3.2.

La seconde étude que nous retenons est celle de E. Chew [CSTZ05, CZS+05]. Cette étude a expérimenté une deuxième configuration audio, en introduisant un retard local identique au délai de communication entre les musiciens (voir figure 8.6(b)). Le retour perçu par les musiciens est alors "perceptif cohérent" et évite les ambiguïtés temporelles décrites au paragraphe 3.3.2. En effet, l'égalité entre le délai local et le délai entre de communication entre les musiciens assure que les retours sont identiques pour chaque musiciens. Dans cette étude, les musiciens (deux pianistes) sont placés l'un en face de l'autre et doivent jouer une pièce musicale imposée dont le tempo varie au cours du temps. En faisant varier la latence à l'aide d'une table de mixage numérique, les résultats suivants sont obtenus :

- jusqu'à 50 ms, le délai permet aux musiciens d'interagir confortablement
- à 65 ms, le délai est encore acceptable, mais le système rend l'interactivité inconfortable.

Cette étude apporte des résultats qui sont exploitables pour l'étude de nJam. En effet, ces valeurs définissent les seuils de jouabilité avec un système de PMID synchronisé avec une latence minimale. Cela correspond donc au seuil de jouabilité avec nJam, si l'on n'adapte pas la latence au morceau et que l'on n'utilise pas le métronome global (voir chapitre 7).

Cependant, après la synchronisation, nJam ajoute une latence supplémentaire dans les retours dans le but de permettre aux musiciens d'avoir un retour "en rythme" avec ce qu'ils sont en train de jouer (voir paragraphe 7.4). De plus, nJam fournit un métronome global réparti pour aider les musiciens à "anticiper" la latence (voir paragraphe 7.5). Les résultats des deux précédentes études ne peuvent donc pas être appliqués directement à nJam. Nous avons donc effectué une étude sur l'utilisation de nJam que nous décrivons ici.

8.4.1 Description de l'étude d'opinion des utilisateurs à propos de nJam

Suivant la configuration audio de la figure 8.6(c), nous avons fait jouer les musiciens à l'aide de nJam (dans deux pièces différentes) et selon différents scénarios. Après l'exécution de chaque scénario, nous leurs avons demandé de répondre aux questions suivantes[58] :

- Pouvez-vous juger la difficulté du jeu à deux avec le système de retard ?
- Pouvez-vous juger la facilité de créer, d'improviser et/ou d'avoir une interprétation musicale avec le système ?
- Pensez vous que vous pourriez être capable de vous adapter à un tel système pour faire de la musique à l'aide d'un réseau informatique (Internet) ?

Le premier scénario a consisté à imposer un motif rythmique et harmonique aux musiciens, qu'ils devaient exécuter tout d'abord ensemble, puis avec nJam. Nous avons testé différentes configurations du retard local (la configuration est identique pour les deux musiciens). Durant l'exécution de ce scénario, nous n'avons pas activé le volume du métronome de nJam. Cela nous amène à une configuration audio proche de celle de la deuxième étude [CSTZ05, CZS+05].

Le deuxième scénario est identique au premier, mais le métronome donne le battement aux musiciens, et utilise donc toutes les fonctionnalités de nJam. Pour les deux scénarios, nous avons choisi les mêmes latences, basées sur un tempo de 100 battements par secondes :

Latence	300 ms	600 ms	2,4 s	7,2 s	9,6 s
Correspondant à	1 croche	1 noire	4 noires	12 noires	16 noires

Nous avons utilisé le même motif rythmique et harmonique (présenté par la figure 8.7) pour les deux premiers scénarios. Ce motif est emprunté au flamenco (musique traditionnelle espagnole) et présente la particularité de pouvoir être joué en boucle sur une longueur de 12 noires (soit 7,2 s pour un tempo de 100 BPM). Nous décrivons ce motif à l'aide de la figure 8.7 : le motif est constitué de 12 noires, numérotées de 1 à 12. Le motif débute au temps numéro 1. Les extrémités du polygone inscrit nous montrent les temps que les musiciens doivent marquer. Les accords à jouer à ces instants sont inscrits à l'extérieur du cercle (en notation anglaise, par exemple Bb7 correspond à l'accord "si bémol septième"). Cette notation est une notation intuitive de la musique qui est utilisée dans un article sur l'étude du rythme en flamenco [DBFG+04]. Elle est proche de celle utilisée dans un article sur l'étude de rythmes africains [Ank00].

L'utilisation d'un tel motif est intéressante pour plusieurs raisons :

- il est relativement long : il boucle en 7,2 secondes pour un tempo de 100 bpm. Il permet donc d'avoir des résultats sur une latence environ cent fois supérieure à celle mesurée dans les précédentes études
- le rythme est asymétrique : il est difficile de lui superposer d'autres rythmes, à part s'ils sont pensés pour fonctionner avec celui-ci
- la plupart des musiciens soumis aux tests ne sont pas accoutumés à ce type de rythme : nous pouvons alors comparer la difficulté que les musiciens ont à effectuer cet exercice ensemble, puis en réseau.

Le troisième scénario est beaucoup plus libre, puisque nous avons laissé aux musiciens le choix du motif à jouer. Nous avons donc configuré le retard en fonction de leurs choix (tempo et durée).

[58] Ces questions sont tirées de l'étude de E. Chew [CSTZ05, CZS+05].

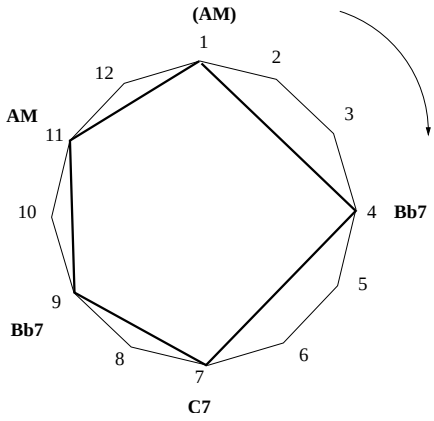


FIG. 8.7 – *Le motif imposé aux musiciens durant les deux premiers scénarios*

Ce scénario utilise le métronome global de nJam avec un retard calculé en fonction du morceau joué par les musiciens. Il est donc quasiment identique au deuxième scénario, excepté le fait que les musiciens ne sont pas contraints à l'exécution d'un motif imposé. Cela nous permet de voir comment ceux-ci réagissent au jeu en réseau lorsqu'ils jouent des pièces musicales auxquelles ils sont accoutumés.

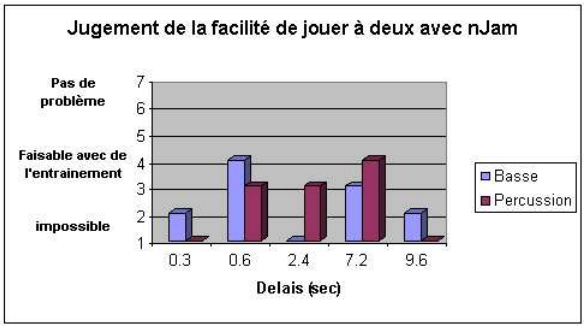
Pour chaque scénario et chaque configuration, nous avons enregistré différents flux audio (pour chaque musicien) : (1) le métronome mixé avec ce que joue le musicien[59], (2) ce que joue le musicien mixé avec le retour retardé et (3) le retour uniquement. Ces enregistrements nous permettent de vérifier la rigueur rythmique du musicien par rapport aux retours et au métronome. De plus, il permet de comparer ce qu'entend le musicien (2) avec ce qu'il produit avec l'autre (3).

8.4.2 Résultats du premier scénario

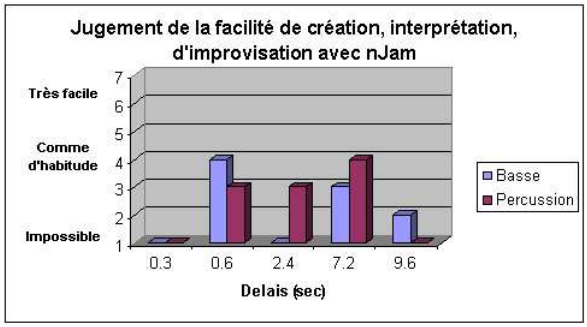
Nous avons expérimenté le premier scénario avec un couple de musiciens (un bassiste et un percussionniste) qui n'avaient jamais joué ensemble auparavant. Ces musiciens n'étaient pas vraiment accoutumés au motif, ainsi l'exécution de celui-ci leur a demandé un effort de concentration, même lorsqu'ils l'ont joué ensemble au préalable, dans la même pièce.

Sur les enregistrements audio, nous constatons que le bassiste suit le tempo du retour retardé. Pour être plus précis, le bassiste suit le rythme joué par le percussionniste. En effet, nous entendons sur l'enregistrement des retours (3) que par rapport au métronome le motif est décalé (ou "déphasé") d'une durée équivalente au retard choisi. Sur les enregistrements du percussionniste, nous constatons que celui-ci s'est positionné en "meneur", puisqu'il accélère ou décélère son tempo pour se resynchroniser en fonction de son propre retour (à l'exception de la configuration à 300 ms). Voici les valeurs de tempo que nous avons mesuré sur ces enregistrements :

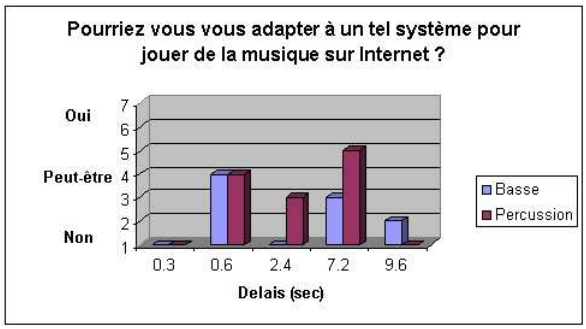
[59]Pour le premier scénario, l'enregistrement (1) ne contient que la musique jouée par le musicien, puisque le volume du métronome est muet. Dans ce cas, nous pouvons utiliser cet enregistrement pour estimer le tempo réellement joué par le musicien, en mesurant la période du motif à l'aide d'un éditeur de fichier audio.



(A) Facilité de jeu



(B) Facilité d'interprétation, improvisation, création musicale



(C) Adaptation au système

FIG. 8.8 – *Opinions des utilisateurs durant l'exécution du premier scénario*

Latence	300 ms	600 ms	2,4 s	7,2 s	9,6 s
Tempo mesuré (en BPM)	93	95-98	116-122	117	105

D'après les réponses au questionnaire (figure 8.8), nous constatons que les valeurs de 600 ms et de 7,2 s sont plus populaires. Nous pouvons supposer que cela provient du fait que ces latences ont une correspondance rythmique avec le motif demandé. Par exemple, pour le test à 600 ms de latence, "*le battement se plaque sur la latence*" comme l'a remarqué le percussionniste. Ceci est confirmé par les mesures de tempo : le test à 600 ms de latence est celui pour lequel les musiciens sont les plus proches de 100 BPM.

Le succès du test à 7,2 secondes semble provenir du fait que la latence est suffisamment importante pour que les musiciens puissent essayer de superposer le motif qu'ils jouent avec le motif entendu dans les retours. Nous constatons cependant sur l'enregistrement "entrée et retours" (2) du percussionniste que celui-ci est constamment en train de ralentir et d'accélérer pour trouver le tempo qui permet de mettre en phase l'entrée et la sortie.

Les autres valeurs de latence n'ont pas eu le même succès, car les musiciens y ont senti un décalage gênant au niveau rythmique. Voici donc les résultats à retenir issus de ce scénario (pour des latences importantes, i.e. au dessus de 300ms) :

- il y a une relation directe entre le rythme et la latence
- sans métronome de référence, les musiciens cherchent le battement approprié à une latence donnée.

8.4.3 Résultats du second scénario

Le second scénario s'est effectué dans les mêmes conditions, excepté la présence du métronome global de nJam (voir le paragraphe 7.5). L'utilisation de celui-ci a permis aux musiciens de jouer à un tempo régulier (100 BPM) dès les premières mesures. Ce deuxième scénario fût testé avec deux couples de musiciens : le couple basse et percussion du premier scénario et un couple guitare/chant. Ayant moi même joué la partie de guitare, je n'ai pas rempli de questionnaire.

Après écoute des enregistrements du retour uniquement (3), nous constatons que les musiciens suivent principalement les retours. Ainsi, le résultat produit est "déphasé". Autrement dit, si la latence n'est pas en phase avec le motif rythmique, les musiciens n'arrivent pas à exécuter "naturellement"[60] le motif ensemble, à l'unisson. Ainsi, seul le test à 7,2 secondes (12 noires à 100 BPM) produit un retour dans lequel les musiciens exécutent le motif à l'unisson. La figure 8.9 nous montre la superposition du motif joué avec le motif entendu dans les retours. Seule la configuration à 12 noires conserve le rythme en entrée en phase avec le rythme des retours.

Les résultats du questionnaire (figure 8.10) confirment cette hypothèse : l'expérience à 7,2 secondes de latence a obtenu un résultat nettement supérieur aux autres tests. Notons cependant que le chanteur a donné une opinion équivalente au test avec une latence correspondant à 4 noires. En effet, celui-ci, contrairement aux autres musiciens, ne chantait pas le motif, mais une phrase courte (moins de quatre temps) qui lui permettait d'entendre son propre retour à un instant auquel il ne chantait pas. Voici les commentaires du chanteur pour les tests à un, quatre et douze temps de latence :

- Une noire : "*cela peut être un bon moyen de création*"

[60] c.à.d. sans entraînement préalable.

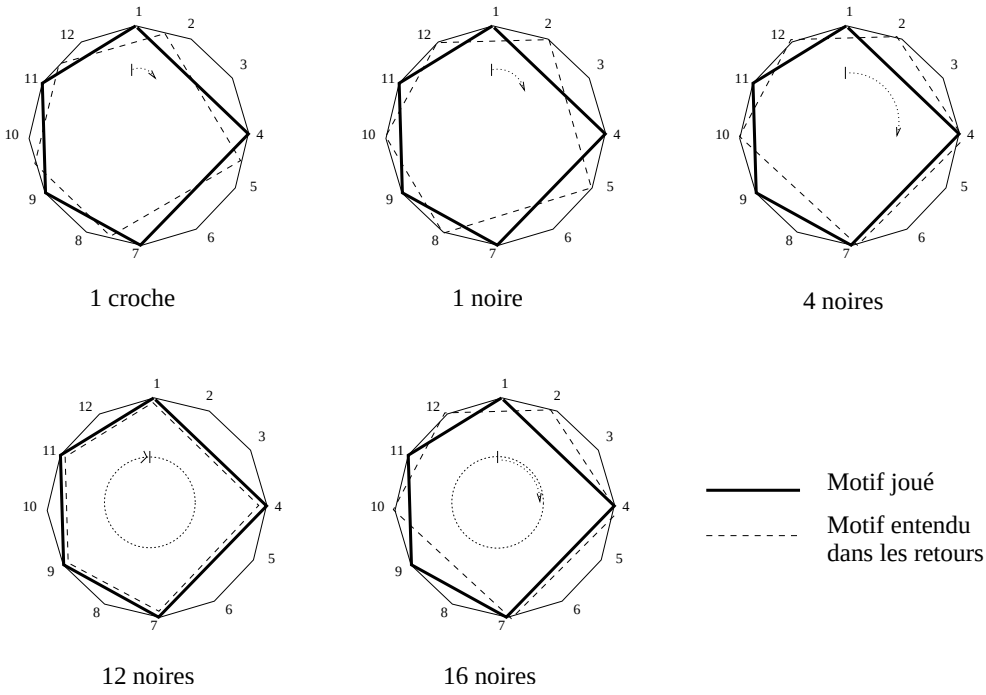


FIG. 8.9 – *Le déphasage rythmique entre le motif joué et le motif issu des retours*

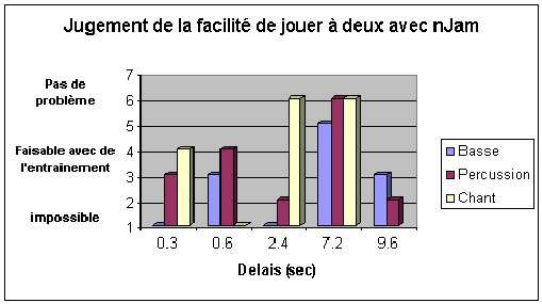
- Quatre noires : "*pas de problème car le retard arrive au moment où je ne chante pas (par contre le guitariste avait du mal, cela s'entend)*"
- Douze noires : "*très bien car le retard correspond à la mesure, ma voix s'écoute alors en cœur*".

En contrepartie, pour les tests à quatre et à seize noires de latence, nous constatons sur les enregistrements audio que les musiciens ont du mal à effectuer correctement le motif rythmique. Cela provient probablement du déphasage rythmique présenté à la figure 8.9. Pour les autres tests, malgré les différents niveaux de confort estimés, les musiciens ont réussi à exécuter le motif sans erreurs.

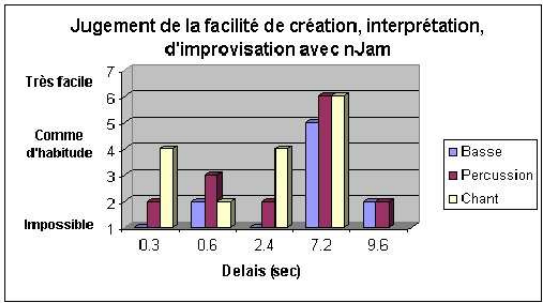
Ce scénario nous montre que le métronome est un outil utile dans la mesure où les musiciens n'ont plus besoin de ralentir ou d'accélérer pour trouver un tempo "en phase" avec la latence. De plus, ce test nous montre que le confort de jeu est beaucoup plus important quand la latence est choisie en fonction de la structure rythmique (et harmonique car les accords se superposent aussi !) de la pièce musicale jouée.

8.4.4 Résultats du troisième scénario

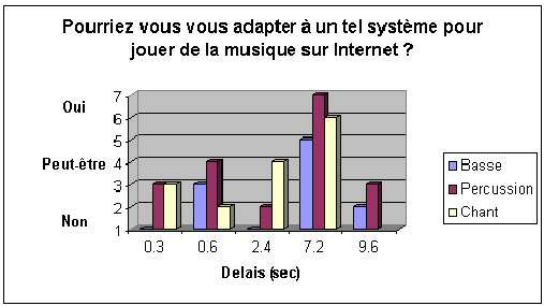
Pour le troisième scénario, nous avons choisi de ne pas imposer de motif rythmique et harmonique aux musiciens. Cela nous a permis d'étudier leurs réactions "spontanées" vis-à-vis du système (avec le métronome), puisqu'ils jouent la musique dont ils ont l'habitude. Ce test a été effectué par cinq couples de musiciens. J'ai moi-même participé à trois de ces couples, je n'ai donc pas répondu aux questions. Voici un descriptif des cinq couples (le retard configuré est équivalent à la durée des boucles) :



(A) Facilité de jeu



(B) Facilité d'interprétation, improvisation, création musicale



(C) Adaptation au système

FIG. 8.10 – *Opinions des utilisateurs durant l'exécution du deuxième scénario*

- le couple basse/percussion a choisi un motif rythmique et harmonique bouclant en 3,809 secondes (8 noires à 126 BPM)
- le couple chant/guitare a choisi de jouer un morceau long (sans boucle) avec un retard de 2,727 secondes (4 noires à 88 BPM)
- le couple basse/guitare a choisi d'improviser (solo de Jazz) sur une boucle harmonique de 8 secondes (16 noires à 120 BPM)
- Le couple guitare/guitare a choisi de jouer un motif de rock (un riff) de 7,619 secondes (16 noires à 126 BPM)
- le couple trombone/guitare a choisi de jouer un motif de Jazz d'une durée de 6 secondes (16 noires à 160 BPM).

Avec ce scénario, nous exploitons la configuration du deuxième scénario ayant obtenu la meilleure opinion. En effet, nous appliquons un retard dans les retours qui correspond à la durée de la boucle jouée par les musiciens. Cela nous permet d'estimer la part de difficulté qu'a amené l'obligation de jouer un motif imposé.

Comme nous le montrent les résultats du questionnaire (figure 8.11), ce scénario n'a pas fonctionné pour le morceau long effectué par le couple guitariste/chanteur. En effet, le retard n'était pas en phase avec la structure du morceau : les harmonies se superposent et il semble difficile, sans entraînement, d'anticiper le retard pour que le retour soit correct par rapport au morceau écrit. Cependant, avec anticipation et grâce à la connaissance du système, ce couple a réussi à effectuer un départ synchronisé (de la même manière que durant l'expérience de Résonances 2003), sans pouvoir pourtant continuer le morceau correctement. Remarquons tout de même que pour une configuration similaire (excepté le tempo), les musiciens de Résonances 2003 ont réussi à jouer un morceau long avec un retard d'une mesure.

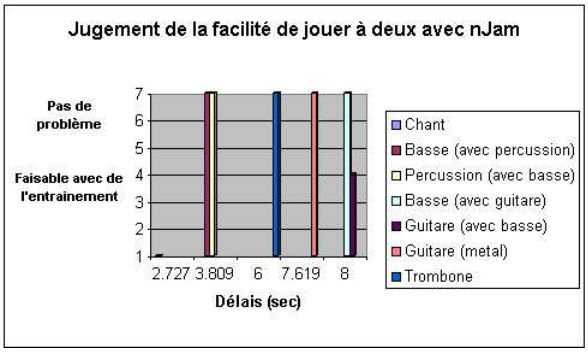
Mais la majorité des résultats des questionnaires montrent qu'avec ce principe, il n'est pas difficile de jouer de la musique dans une telle configuration. De plus, les musiciens ont différents niveaux d'études musicales, allant d'une année de cours à 8 ans de conservatoire. Ces résultats nous montrent aussi que les musiciens sont beaucoup plus à l'aise lorsqu'ils peuvent choisir ce qu'ils vont jouer (ce scénario est techniquement identique au deuxième scénario). Voici une liste de remarques effectuées par les musiciens :

- Le tromboniste : "*j'ai un peu d'expérience avec les systèmes de pédales qui construisent des boucles, je ne ressens donc pas de gêne*"
- Le bassiste : "*le métronome peut s'oublier après quelques mesures mais il aide en cas de problème. En effet, La période étant de 16 noires, il faut un certain temps pour retomber sur ses pieds en cas de pépin*"
- Le percussionniste : "*il est possible d'adapter un style de jeu avec ce paramètre de latence. Cela serait un nouveau concept d'improvisation. Par exemple, les "questions/réponses" se dérouleraient sur un temps distinct du "présent". La musique se déroulerait alors sur deux temporalités différentes !*"

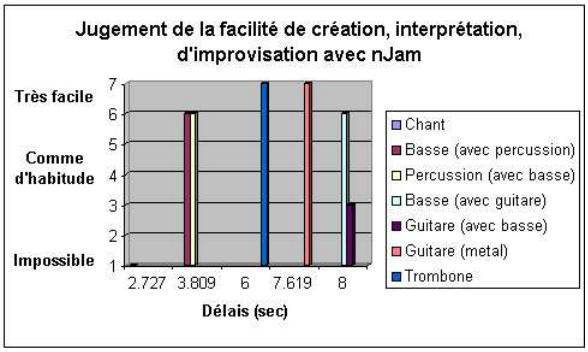
8.4.5 Conclusion de l'expérimentation par des utilisateurs

Avec cette expérience, nous avons montré qu'il était possible pour des musiciens d'avoir une interaction malgré des latences importantes (au moins jusqu'à 8 secondes). En effet, les études de la littérature fixait la borne maximale d'interactivité à une latence à 65 ms.

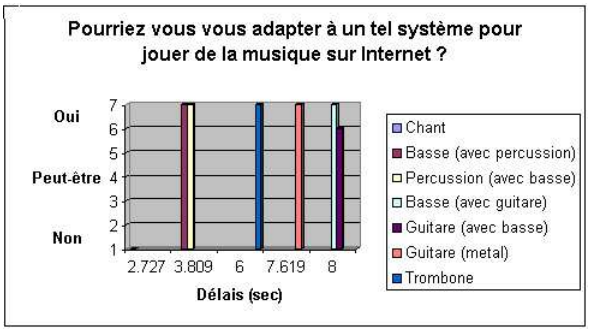
Tout d'abord, le premier scénario nous a permis de montrer que pour des latences importantes



(A) Facilité de jeu



(B) Facilité d'interprétation, improvisation, création musicale



(C) Adaptation au système

FIG. 8.11 – *Opinions des utilisateurs durant l'exécution du troisième scénario*

dans le retour audio des musiciens, il y avait une correspondance entre la latence soumise et la musique jouée par les musiciens (tempo et durée de la boucle). Ensuite, grâce au deuxième scénario, nous montrons qu'avec un rythme imposé et le métronome global de nJam, les problèmes de tempo sont annulés, rendant beaucoup plus confortable l'expérience de jeu des musiciens. Les résultats de l'exécution de ce scénario montrent aussi que le retard introduit doit être choisi en fonction de la structure du morceau. Plus particulièrement, si la latence est en phase avec l'harmonie et le rythme, alors le système devient beaucoup plus confortable. Cela est confirmé avec le troisième scénario : lorsque les musiciens choisissent leur décalage et la musique qu'ils vont jouer, nJam obtient les notes maximales.

Ces résultats nouveaux montrent aussi que le confort est possible dans certaines conditions uniquement : le tempo doit rester fixe et le morceau doit être basé sur une structure de boucle(s) (même s'il est probable qu'avec de l'entraînement, des musiciens puissent exécuter un morceau écrit et long grâce à la connaissance du système et à une anticipation de la latence). Cependant, malgré cette remarque, les musiciens ont suggéré que le type d'interaction fourni par nJam pourrait être intéressant dans un contexte de création, si l'interactivité musicale tenait compte des décalages de façon intrinsèque, ouvrant alors la porte à de nouveaux types de créations musicales.

Grâce à ces résultats, nous montrons qu'il est possible pour des musiciens répartis sur le réseau d'interagir à des distances très importantes, sur des réseaux à latences supérieures aux seuils de jouabilité définis par les études de la littérature.

8.5 Conclusion

A travers nos mises en œuvre, nous voyons que les Performances Multimédia Interactives Distribuées (PMID) restent encore aujourd'hui expérimentales et nécessitent encore des évolutions pour se rapprocher des cas d'utilisations. En effet, différents aspects de ce type de systèmes nécessitent encore des recherches pour découvrir des solutions et des techniques appropriées. Dans ce chapitre, à travers des expérimentations de notre système (nJam) et de tests utilisateur, nous avons identifié différentes pistes à explorer pour l'avenir et montré l'efficacité des solutions présentées dans les chapitres précédents.

Nos expériences ont montré l'utilité et l'efficacité du protocole de cohérence perceptive couplé au métronome global. En effet, malgré le fait que les musiciens ont eu peu de pratique avec nJam, ils ont su s'adapter rapidement au système. Remarquons de plus que les musiciens de la démonstration de *Résonances* avaient l'habitude de jouer en groupe ensemble, alors que certains musiciens de l'expérience de la démonstration de la *fête de la science* ne se connaissaient pas avant la première répétition.

L'utilisation d'une interface homme/machine minimale a aussi fait ses preuves. En effet, ce sont les instruments de musique qui sont utilisés par les musiciens pour interagir avec le système. Il n'ont donc pas eu besoin d'apprendre à manipuler l'interface, mais juste à comprendre ce qu'ils allaient entendre. Cela leur a permis très rapidement d'intégrer eux mêmes des usages du système non prévus initialement, comme le départ synchronisé et l'improvisation. En d'autres termes, les musiciens ont été capables d'utiliser des éléments de leur culture commune pour interagir avec le système.

Bien que les résultats des expérimentations soient très encourageants, nous avons constaté que la mise en œuvre de ces expériences est complexe et surtout qu'elle dépasse le cadre spécifique de la musique interactive. Par exemple nous avons énormément utilisé le téléphone pour effectuer la coordination entre les sites. Il aurait alors été logique d'intégrer un système de téléphonie sur IP dans l'application. Bien que l'intégration d'un tel système soit nécessaire à terme, il ne pourra être utilisé que sur des machines possédant un nombre suffisant de périphériques.

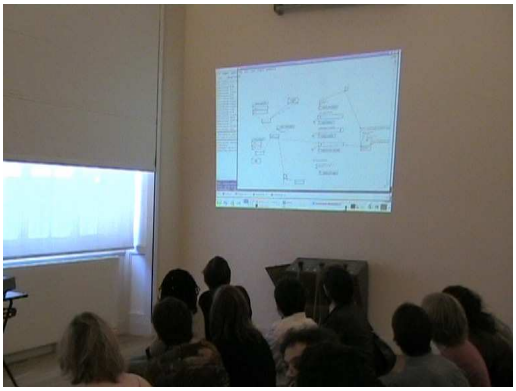


FIG. 8.12 – *L'apparition du contrôle (un patch jMax) dans la partie visuelle de la démonstration*

Nous pouvons faire la même remarque pour la partie vidéo : lors de la démonstration de la fête de la science, dans l'amphithéâtre nous avons pu projeter les vidéos des musiciens et garder les fenêtres de configuration à l'écran du moniteur, grâce à une carte graphique double écran. Cependant, nous n'avons pas pu obtenir la même configuration dans le musée, à cause d'un problème de compatibilité avec le matériel. Le public a donc vu des éléments liés à la configuration de la démonstration (voir la photo 8.12).

Ainsi, pour la partie audio comme pour la partie vidéo, il est nécessaire de pouvoir contrôler et router un ensemble de flux multimédia en fonction de leurs utilisations. De plus, chaque flux sera dédié à des utilisations spécifiques et à différents "corps de métiers". Cela montre que l'application qui sera capable de supporter des démonstrations à grande échelle devra être capable de gérer et d'orchestrer un ensemble de flux multimédia et un ensemble de périphériques d'entrées/sorties.

Dans ce chapitre, le principal résultat que nous fournissons au domaine des PMID est la démonstration de la faisabilité de l'interaction musicale, lorsque les musiciens sont soumis à des latences de plusieurs secondes. En effet, les précédentes études concluaient que l'interactivité entre musiciens n'était pas possible pour des latences supérieures à 65 ms. A l'aide de notre tests d'opinion effectués avec des musiciens, nous avons montré que les abstractions fournies par nJam permettent de conserver un certain confort dans l'interaction musicale, malgré des latences de plusieurs secondes (jusqu'à 8 secondes durant nos tests). Nous montrons à travers nos tests que la latence est largement acceptable par les musiciens, si celle-ci tient compte du tempo et de la structure rythmique et harmonique du morceau. De plus, nous montrons que dans ce cas, l'utilisation du métronome global permet aux musiciens d'anticiper la latence sans aucun effort. Ces résultats ouvrent de nouvelles perspectives pour le domaine des PMID : (1) la possibilité d'interagir avec des latences de plusieurs secondes rend possible l'interaction à grandes distances et sur une gamme de réseaux beaucoup plus large (2) l'interactivité avec retours retardé choisis

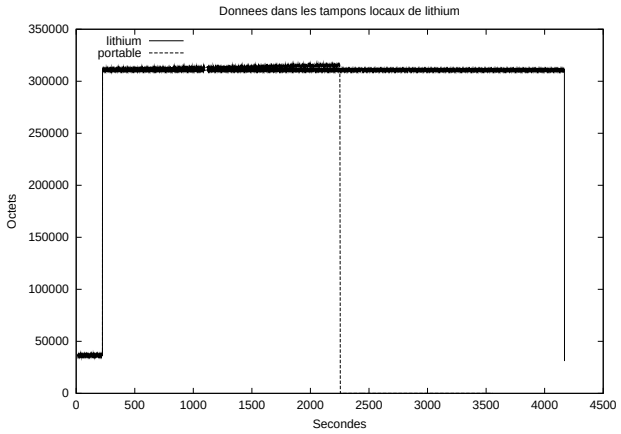
en fonction du morceau permet d'envisager de nouvelles façons d'interagir musicalement tout en conservant la pratique traditionnelle de jeu.

L'ensemble des résultats de ce chapitre montre concrètement la faisabilité de notre système. Un transfert de technologie vers la communauté culture est tout à fait envisageable, dans la mesure où le succès de la réalisation d'événements publics (ou non) passe par la disponibilité d'un nombre important de ressources matérielles (matériel multimédia, salle, réseau, etc.) et d'outils permettant de masquer la technologie sous-jacente.

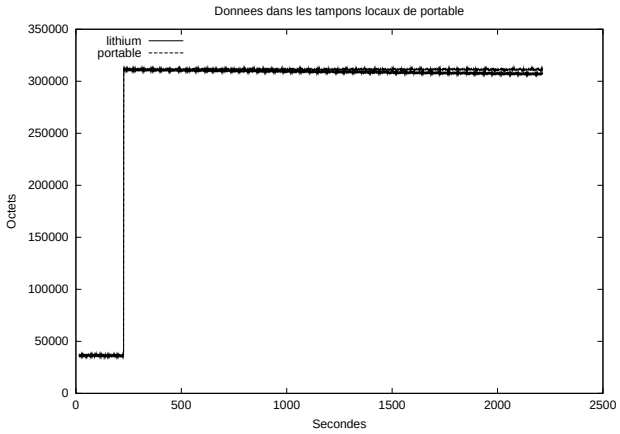
9

Mesures de performances du système

Sommaire



(A) Les tampons de la machine "portable"



(B) Les tampons de la machine "lithium"

FIG. 9.1 – *Comportement idéal des tampons de nJam*

9.1 Introduction

Rappelons que les tampons de nJam fonctionnent comme un seau percé (voir le paragraphe 7.3.1) : ils servent à lisser la gigue introduite par le réseau. Chaque schéma montre la quantité de données (exprimée en octets) dans le tampon de réception au cours du temps (exprimé en secondes). Ainsi, chaque figure présente plusieurs courbes : une pour chaque flux reçu, incluant le flux local, puisque celui-ci participe au protocole de cohérence perceptive. Nous présenterons aussi des mesures de pertes et de gigue.

En théorie, si les horloges des cartes sons fonctionnent exactement à la même fréquence, le débit moyen en entrée des tampons est identique au débit moyen en sortie. Chaque courbe doit donc ressembler à une droite de la forme $y = a$ ou a est la valeur de remplissage initiale du tampon. Notons que la valeur de a va changer lorsque le métronome sera initialisé, puisque nJam rajoute des silences dans les tampons pour assurer la cohérence perceptive durant la phase d'initialisation. De plus les courbes ne sont jamais complètement horizontales puisque la gigue du réseau introduit des variations de niveau.

Les courbes présentées à la figure 9.1 nous montrent le comportement des tampons de nJam dans un cas idéal : celui dans lequel l'application fonctionne sans artefacts auditifs. Ces mesures ont été effectuées à l'IRCAM, durant le tournage du reportage pour RENATER [GIP04] auquel nous avons participé avec l'IRCAM. Dans ce test, nous avons utilisé deux cartes sons professionnelles identiques (RME Hammerfall Multiface) sur deux machines distantes (nommées *portable* et *lithium*). Nous voyons sur ces courbes que le flux local reste horizontal au cours du temps, ce qui est logique puisque l'horloge locale ne dérive pas par rapport à elle même. Par contre, nous pouvons voir que le tampon de réception du flux distant augmente progressivement pour la machine "portable" et diminue (de façon symétrique) pour la machine "lithium". Cependant, la dérive n'est pas importante et permet au système de fonctionner pendant une durée suffisante (environ 40 minutes sur la figure).

Dans ce chapitre, nous proposons une analyse du comportement de nJam durant les expériences présentées au chapitre précédent. Nous nous intéressons donc maintenant aux aspects systèmes et réseaux du prototype et plus particulièrement aux obstacles que nous avons rencontrés durant la mise en œuvre. nJam est une AMID, une classe d'applications encore aujourd'hui peu répandue sur les réseaux des opérateurs. Ainsi, l'analyse de son fonctionnement en conditions réelles permet d'évaluer les conditions de faisabilité du déploiement de telles applications sur les réseaux et les équipements actuels.

Au paragraphe 9.2, nous présentons les analyses qui nous permettent d'effectuer des suppositions sur les raisons de l'annulation de la présentation publique de *Résonances 2003*, malgré le succès des répétitions. Ensuite, au paragraphe 9.3 nous présentons et analysons les performances de nJam durant la présentation publique de la *Fête de la science 2005*[61]. Finalement, nous concluons au paragraphe 9.4 en exposant les nouveaux problèmes de recherche identifiés.

9.2 *Résonances 2003* : un problème de performances du réseau

Comme nous l'avons vu dans le paragraphe 7.3, pour transporter les données de bout en bout, nJam utilise le protocole RTP et son protocole de contrôle RTCP (voir au paragraphe 2.2.2). Grâce au protocole RTCP, nous avons mesuré et sauvegardé au cours du temps des valeurs statistiques pour chaque flux des sessions, telles que le nombre de pertes, le délai aller-retour et la gigue.

9.2.1 La configuration du réseau

Comme nous l'avons vu au paragraphe 7.6, nJam repose sur le multicast IP. Le CNAM et l'IRCAM ne disposaient pas à l'époque de l'expérience d'abonnement au multicast auprès de RAP et de RENATER. Dans la mesure où nous n'avons pas obtenu de financement pour monter l'expérience, nous avons décidé d'installer un tunnel IP permettant de créer un domaine multicast contenant les deux réseaux locaux impliqués [Bon05].

La construction de ce tunnel nécessite alors l'installation d'un routeur logiciel par réseau local. Ainsi, le paquet IP ayant pour destination une adresse multicast est encapsulé dans une liaison point à point, à l'aide du protocole GRE [FLH+00]. Ce sont donc ces routeurs logiciels qui vont être responsables de l'acheminement des données multicast sur les deux réseaux locaux, imposant la désactivation des protocoles multicast sur les routeurs de bordures, gérés par les administrateurs réseaux du CNAM et de l'IRCAM[62].

Avant de mettre en place et de tester les routeurs logiciels sur l'infrastructure de la démonstration, nous avons testé le tunnel au sein de réseaux locaux du CNAM. L'infrastructure réseau du CNAM était alors composée d'un routeur auquel sont connectés des commutateurs Ethernet et des équipements ATM, émulant de l'Ethernet.

Durant ces tests (effectués en mai 2003), nous avons déployé trois instances de jMax/nJam qui communiquaient en multicast via le tunnel. Après un premier test, nous nous sommes aperçus que les routeurs logiciels n'étaient pas capables de supporter la charge du nombre de paquets à

[61] Voir la vidéo à l'URL suivante :
`http://cedric.cnam.fr/~bouill_n/FDS/CVR_FDS_Morceaux14h30.avi`

[62] Pour plus d'informations, voir le mémoire d'ingénieur de Rémy Bonafous [Bon05], qui a installé les routeurs logiciels pour les tests et la manifestation.

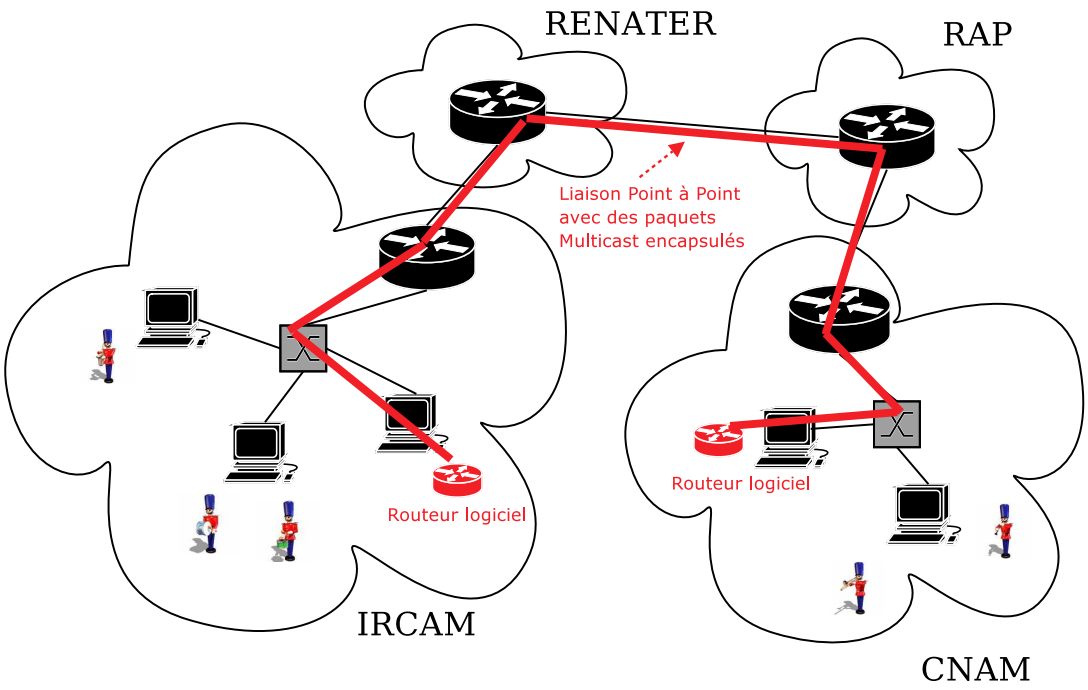


FIG. 9.2 – *L'infrastructure réseau durant résonances 2003*

router. En effet, chaque paquet contenait alors 64 échantillons audio MIC (le nombre d'échantillons d'une boucle de traitement de jMax) soit une charge de (44100/64) x 3 ≈ 2000 paquets par seconde pour trois flux audio mono échantillonnés à une fréquence 44100Hz. De plus, il faut ajouter dans le tunnel les messages du protocole de routage PIM-SM.

Nous avons alors choisi de remplir au maximum les paquets RTP, augmentant la latence de bout en bout. Pour cela, nJam dispose de 1460 octets (voir la figure 9.3), soit un remplissage de (1460 − (1460 *modulo* (64 x 16/8)) = 1408 octets, soit environ 16ms de données, pour un flux échantillonné avec une quantification de 16 bits et une fréquence de 44100 Hz.

Cela montre clairement que sans équipements spécifiques et performants, il est difficile d'obtenir une latence réseau très faible, i.e. inférieure à 16ms. Comme nous l'avons montré au chapitre précédent, nJam permet de fonctionner malgré des latences très importantes (l'exemple du paragraphe 7.4 préconise l'utilisation d'une latence de 4 secondes, ce qui reste une borne largement supérieure).

Lors du déploiement du prototype sur l'infrastructure de la démonstration, nous avons rencontré un autre problème lié à la fragmentation des paquets IP. En effet, l'encapsulation de paquet IP dans d'autres paquets IP à l'aide du protocole GRE augmente de 24 octets la taille du paquet IP d'origine. Ainsi, si le paquet IP d'origine fait au moins 1500−24+1 octets, alors le nouveau paquet IP va dépasser les 1500 octets de MTU[63] d'Ethernet. Les paquets RTP n'entraient

[63]Le MTU (pour Maximum Transfert Unit) est la taille maximum des paquets, exprimée en octets, qu'un réseau peut transmettre

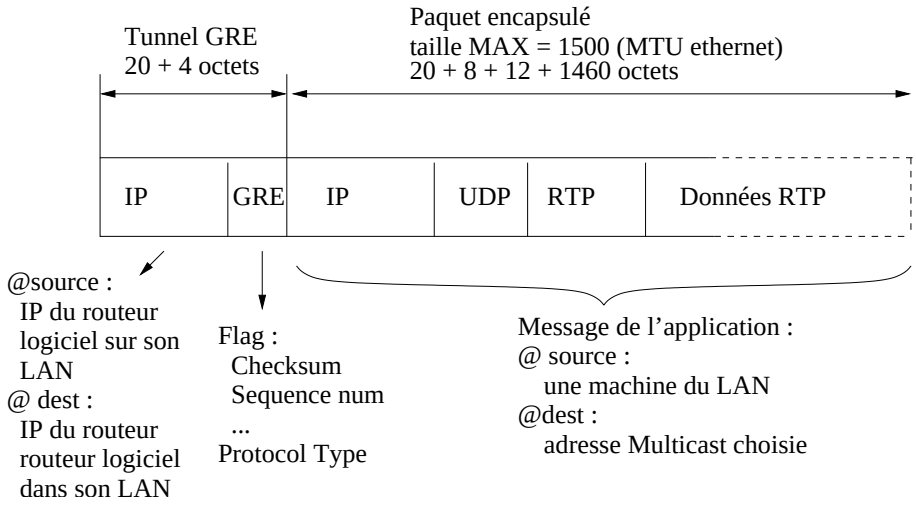


FIG. 9.3 – *L'encapsulation de RTP dans un tunnel GRE*

pas dans ce cas, contrairement aux paquets RTCP qui agrègent un maximum d'informations par paquets (voir le paragraphe 2.2.2).

Nous nous sommes alors rendu compte que le routeur logiciel n'effectuait pas de fragmentation IP et jetait les paquets trop longs, empêchant alors nJam de s'envoyer les informations de synchronisation contenues dans les paquets de type APP du protocole RTCP.

Ces tests et les modifications mineures de nJam qui ont suivies ont finalement permis de mettre les musiciens dans les conditions de la démonstration et d'effectuer des répétitions.

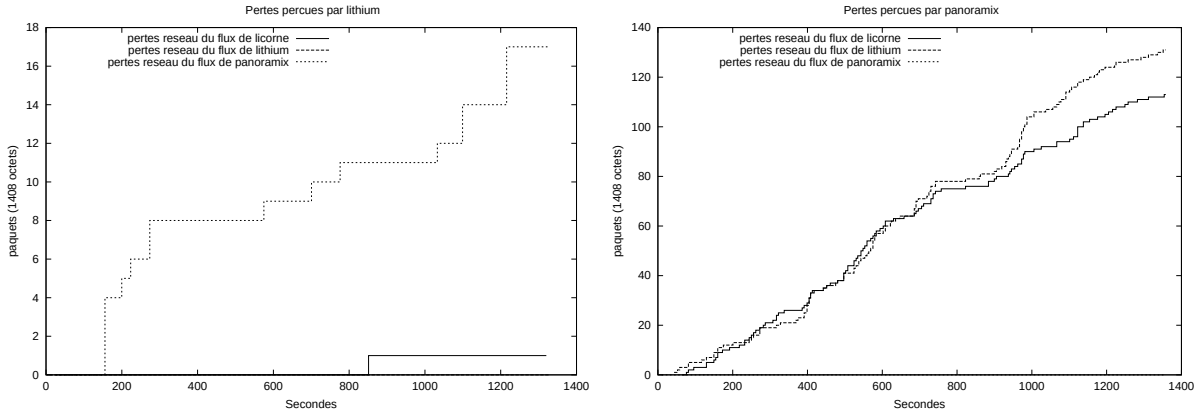
9.2.2 Les répétitions

C'est durant la première répétition que les musiciens ont pu exécuter le morceau utilisant une structure de blues. Les conditions réseau étaient alors suffisamment bonnes pour permettre l'interaction musicale entre trois machines : *lithium* et *licorne*, localisées à l'IRCAM et *panoramix*, située au CNAM.

Rappelons que chaque paquet RTP contient 1408 octets de données MIC échantillonnées à 44100Hz 16 bits. Le débit par flux est donc de (44100 x 16)/(1408 * 8) $\approx$ 63 paquets par seconde. La figure 9.4 nous montre le total cumulé de paquets perdus par flux, au cours de la session et en fonction du temps. Ainsi, du point du vue de la machine *lithium*, le taux de pertes peut être considéré comme nul. Du point de vue de la seule machine située au CNAM (*panoramix*), le taux de pertes pour la réception des deux flux issus du tunnel : environs 100 paquets en 1000 secondes, soit un taux de pertes de $100/(1000 \text{ x } 62,64) = 0,15$ % de pertes.

Ces mesures montrent que le tunnel et le réseau supportent correctement la charge de l'application, mais aussi que la qualité de transmission est légèrement plus faible dans le sens IRCAM vers CNAM.

La figure 9.5 nous montre la gigue mesurée durant ces mêmes répétitions, mais lors d'une autre session RTP. Sur une durée d'environ 3700 secondes (1 heure), la gigue dans le tunnel reste à peu près constante dans les deux sens. Elle est d'environ 450 échantillons MIC (c'est l'horloge



(A) lithium (B) panoramix

FIG. 9.4 – *Les pertes durant la répétition de Résonances*

d'échantillonnage qui est utilisée pour dater les messages RTP), soit une gigue de $450/44100 \approx 10$ ms. Cette gigue est probablement due aux imprécisions de mesures, car elle est équivalente à la durée correspondante à la quantité de données contenues dans un paquet RTP. Les mesures des délais aller-retour que nous effectuées en parallèle varient entre 6 et 16 ms environ, confirmant alors l'hypothèse.

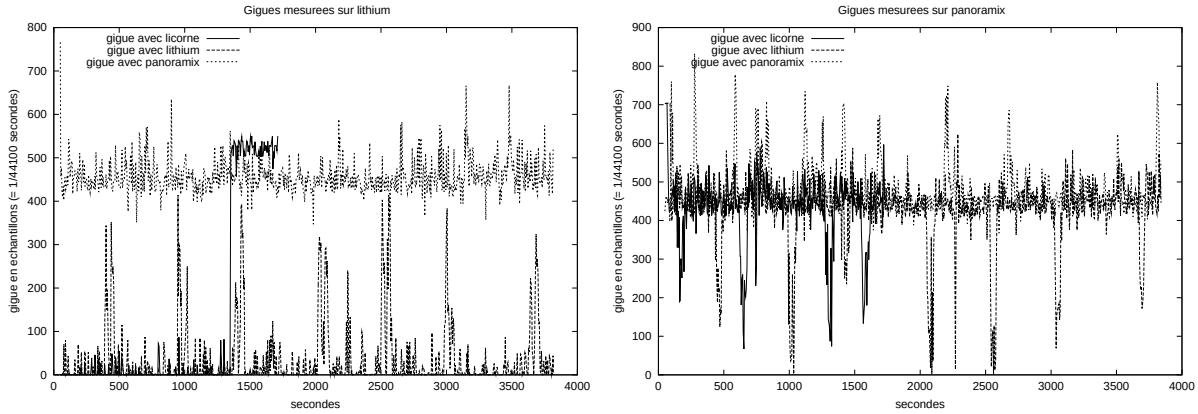
9.2.3 Les mesures durant la démonstration

Les figures 9.6 et 9.7 nous montrent les même mesures le jour de la démonstration, nous allons voir que les conditions du réseau se sont considérablement dégradées.

La figure 9.6 nous montre un comportement similaire des pertes pour *licorne* et *lithium*, les machines situées à l'IRCAM. En effet, nous constatons un taux de perte d'environ $8500/(250 \text{ x } 62,64) \approx 53\%$. Le taux de perte est moins important pour les deux flux allant vers le CNAM : environ 140 paquets en 250 secondes, soit $140/250 \text{ x } 62,64 \approx 0,9$ % de pertes réseau. Dans la mesure où les deux machines situées à l'IRCAM ne constatent pas de pertes entre elles, nous pouvons supposer qu'elles ne sont pas la cause de la défaillance.

La figure 9.7 nous montre la gigue mesurée au même moment, nous pouvons constater que celle-ci est trois fois plus importante pour les hôtes séparés par le tunnel.

Ces mesures mettent en évidence que les conditions du réseau ont joué un rôle important dans le succès de la répétition du week-end précédent, durant laquelle le CNAM et l'IRCAM étaient presque vides. Ce changement de conditions peut alors probablement s'expliquer par le fait que durant le week-end de l'événement, le trafic réseau a augmenté, puisqu'il correspondait à la fête de la science au CNAM et aux journées portes ouvertes à l'IRCAM. Nous avons aussi identifié ce jour là des utilisations "abusives" du réseau, comme l'utilisation d'une application de bureau distant transmettant un économiseur d'écran dans un flux TCP et un téléchargement avec une application pair à pair d'échange de fichiers.



(A) lithium (B) panoramix

FIG. 9.5 – *La gigue durant la répétition de Résonances 2003*

Ces mesures montrent aussi et surtout qu'il est crucial pour une application utilisant RTP (et qui veut maintenir un flux audio basse latence) d'être capable de s'adapter finement à l'état du réseau, en analysant la faisabilité de la communication avec une analyse à posteriori, ou avec des garanties plus solides sur la disponibilité des ressources.

9.3 La *fête de la science* 2005 : un problème matériel

L'expérience de la fête de la science s'est effectuée sur une unique zone d'administration IP : le CNAM. Nous avons donc pu éviter les problèmes liés au réseau en utilisant un VLAN dédié aux machines de la démonstration (voir le paragraphe 8.3). Nous n'avons ainsi pas rencontré de problèmes réseaux.

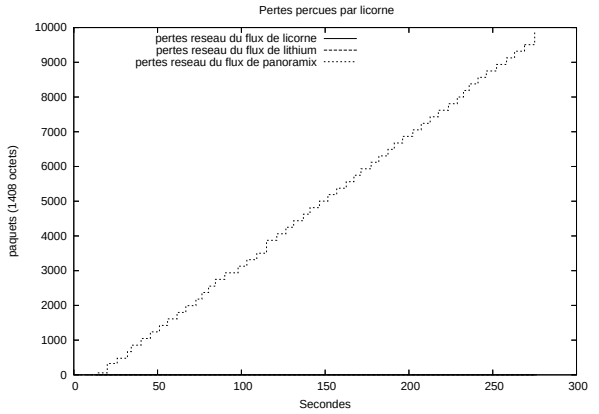
9.3.1 Introduction au problème de dérive d'horloge

nJam effectue un transfert de données isochrones à l'aide de RTP. Contrairement à la téléphonie classique, les horloges de codage et de décodage (les horloges des cartes sons) ne sont pas synchronisées[64].

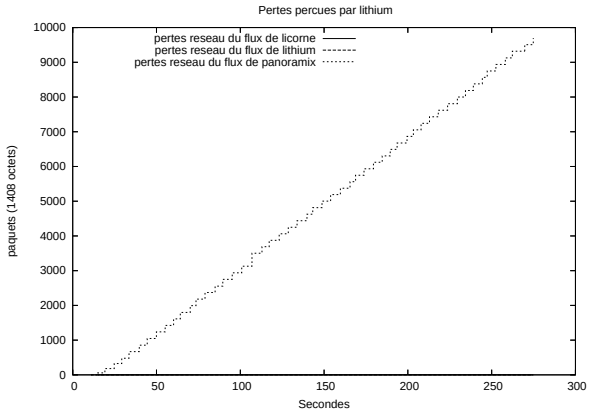
Notons alors f_e la fréquence de l'horloge utilisée pour réaliser l'échantillonnage et f_r la fréquence de l'horloge utilisée pour convertir le signal numérique en signal analogie (en son audio). Sachant que ces horloges peuvent varier plus ou moins faiblement de leurs valeurs nominales, alors les cas de figures suivants peuvent se produire :

1) Si $f_e > f_r$ alors les données sont produites plus rapidement qu'elles ne sont consommées, provoquant à terme un débordement du tampon, et donc des pertes de données.
2) Si $f_e < f_r$ alors les données sont produites plus lentement qu'elles ne sont consommées, provoquant à terme une famine du tampon de réception

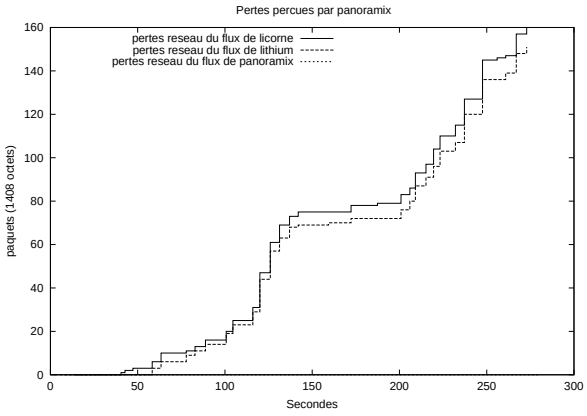
[64]En téléphonie RTC (Réseau Téléphonique Commuté), les horloges de codage et de décodage sont directement liées aux horloges de transmission des données, permettant d'asservir le décodage à l'horloge du codage [Ser03]



(A) licorne



(B) lithium



(C) panoramix

FIG. 9.6 – *Les pertes durant les essais de Résonances*

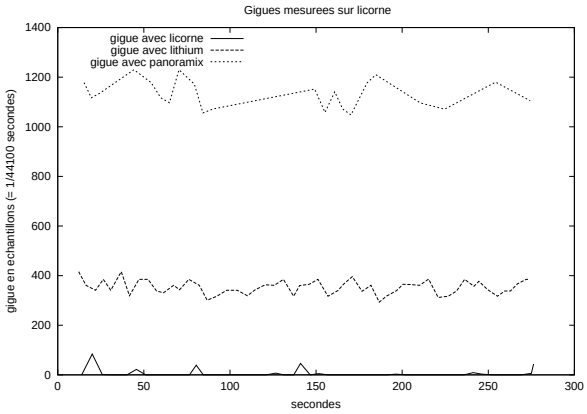
3) Si $f_e \approx f_r$ alors les tampons n'atteindront la famine ou le débordement qu'au bout du durée importante.

Pour les cas 1) et 2), la perte de données va se manifester par des artefacts auditifs très désagréables, mettant en échec l'interaction. Durant la démonstration de *Résonances* ainsi que durant le tournage du reportage pour RENATER [GIP04] nous avions utilisé deux cartes sons professionnelles identiques (RME Hammerfall Multiface) prêtées par l'équipe de l'IRCAM.

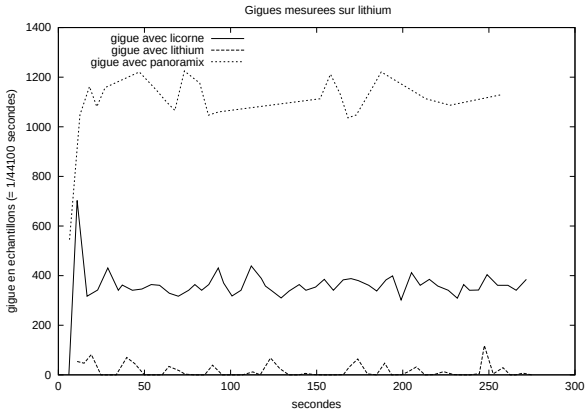
Grâce à la qualité du matériel utilisé, nous étions dans la situation 3), comme le montre les mesures effectuées durant le reportage (figure 9.1).

Lors de nos premiers tests de matériels avant les répétitions de la *fête de la science*, nous avons utilisé deux cartes sons différentes : une de type RME Hammerfall Multiface et une autre de type Creative Audigy2 ZS pro, une carte grand public de bonne qualité.

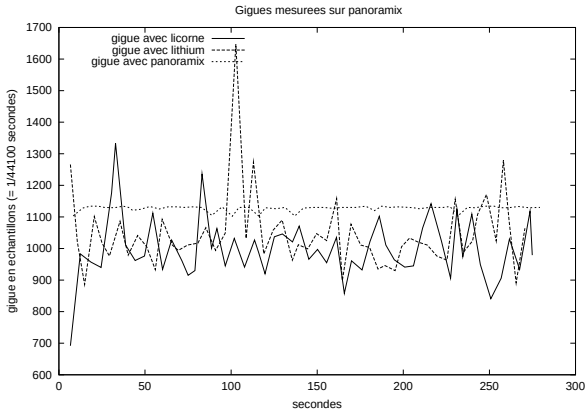
La figure 9.8 nous montre l'état des tampons des deux instances de nJam utilisant ces cartes. Nous pouvons constater que les dérives sont symétriques (la communication est bidirectionnelle).



(A) licorne



(B) lithium



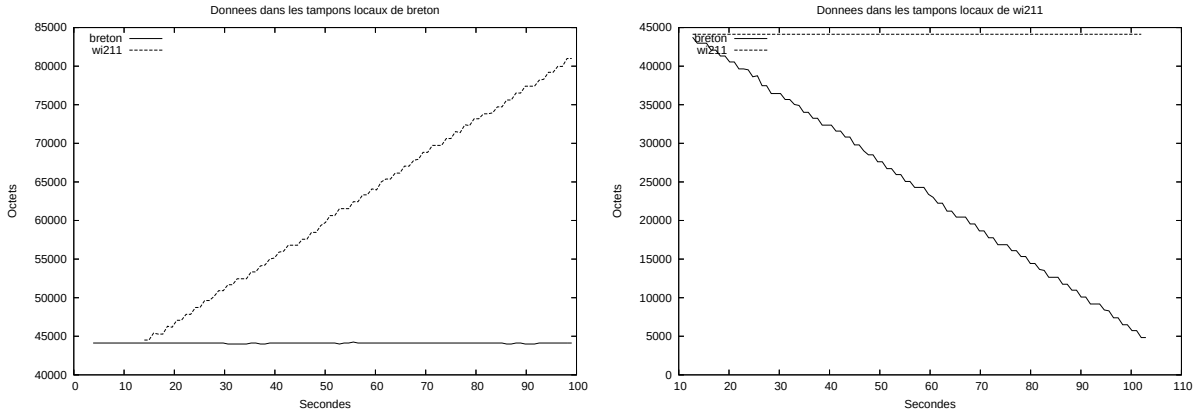
(C) panoramix

FIG. 9.7 – *La gigue durant les essais de Résonances*

En utilisant les valeurs des traces des deux instances de nJam enregistrées dans un fichier, nous pouvons calculer la pente de la courbe :

- figure 9.8(a) : $(80992 - 44512)/(99,042 - 14,024) \approx 430$ octets par seconde, soit 215 échantillons. La dérive mesurée entre les deux cartes sons est donc de 215 Hz (0,4 % de la valeur de 44100Hz nominale).
- figure 9.8(b) : $(4832 - 43744)/(102,929 - 12,771) \approx -431$ octets par seconde, soit la même dérive.

La dérive des horloges ne va pas avoir le même effet sur les deux instances de nJam. En effet la famine risque d'arriver plus vite car la valeur initiale du tampon est relativement faible, en fonction de la gigue que l'application veut tolérer. Sur la figure 9.8(b), la communication ne dépassera pas les deux minutes.



(A) breton (B) wi211

FIG. 9.8 – *Dérive audigy RME*

9.3.2 Les solutions existantes au problème de dérive des horloges des cartes sons

[Fob02, MST99] proposent un ensemble de méthodes de compensation de la dérive des horloges pour des transmissions de données audio sur réseaux IP. Il ont pour cela étudié trois stratégies différentes :

- Synchroniser les horloges des cartes sons par GPS
- Calculer la dérive des horloges puis resynchroniser
- 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 est difficile à mettre en œuvre car elle est très coûteuse : les cartes sons doivent être capables d'asservir leurs horloges sur un signal d'entrée, qu'il soit matériel ou logiciel. La plupart des cartes sons du marché en sont incapables. De plus, il faut installer sur la machine un récepteur GPS.

La deuxième solution consiste à mesurer la dérive des horloges, soit comme nous l'avons fait précédemment, soit en utilisant les horloges systèmes comme référence. Ensuite, le réechantillonnage permettrait de supprimer ou d'ajouter des échantillons sans introduire de discontinuité dans le signal.

La troisième solution consiste à maintenir deux tampons par flux : un tampon équivalent à ceux de nJam et un tampon de secours qui aurait du retard. Ainsi en cas de famine, avec un jeu sur les volumes (un *fade*), le système pourrait passer d'un tampon à l'autre.

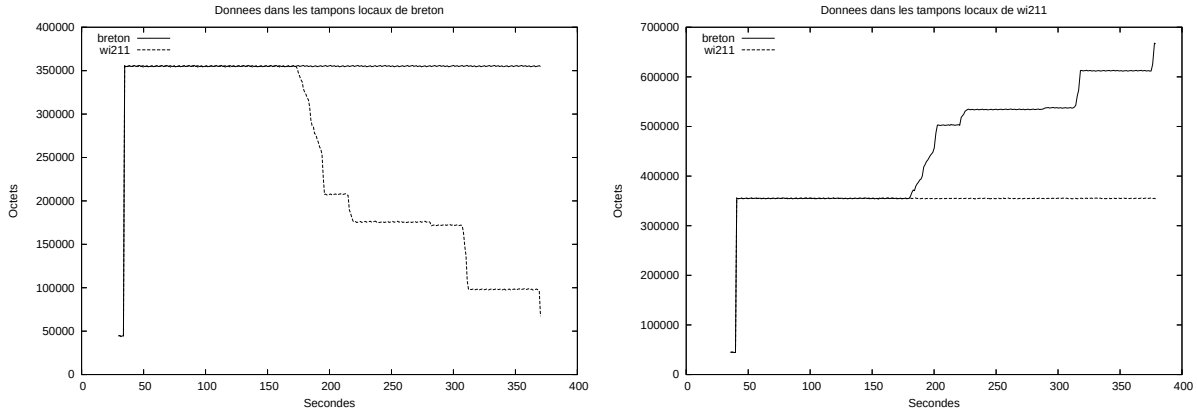
Nous n'avons retenu aucune de ces solutions pour la démonstration de la fête de la science. En effet, la solution GPS est plus coûteuse que de racheter une carte son RME Hammerfall Multiface, avec laquelle nous savons que le problème de dérive devient mineur. Remarquons que les deux autres solutions font appel à des connaissances spécifiques en audionumérique.

De plus, les solutions proposées par les auteurs de [Fob02] et [MST99] tiennent compte de la transmission d'un unique flux. Dans un système de performances musicales interactives et distribuées, il y a autant de dérive que de couples d'instances de nJam. A notre connaissance, seule la solution du GPS permet de résoudre aujourd'hui ce problème efficacement. Cependant,

cela nécessite un investissement considérable. Le musée des Arts et Métiers nous a donc financé l'achat d'une carte son supplémentaire de type RME Hammerfall Multiface.

9.3.3 L'ordonnancement du système

C'est lors des répétitions et de la démonstration que nous avons rencontré un autre problème lié au système. Nous pouvons constater ce problème sur les vidéos des répétitions et de la démonstration : à certains instants, un craquement (ou artefact) de quelques secondes peut être entendu. Ce craquement est gênant car il déconcentre les musiciens et est désagréable pour le public.



(A) breton (B) wi211

FIG. 9.9 – *Pendant la Fête de la science*

La figure 9.9 nous montre le comportement des tampons durant la démonstration. Nous voyons que la quantité de données reste à peu près constante sur des périodes qui sont relativement longues, montrant que les horloges ne dérivent pas beaucoup. Il y a cependant des pertes ou des augmentations brusques de données dans les tampons, correspondant aux instants auxquels les craquements peuvent être entendus (ce problème n'est survenu que sur la machine *wi211*, située dans le musée[65]).

Il faudrait évidemment faire des mesures plus précises pour déterminer de façon plus sûre le problème, mais il est fort probable que l'ordonnancement de jMax par Linux ait fait rater des accès systèmes à la carte son au niveau de la machine *wi211*, expliquant qu'elle ait alors pris du retard dans la consommation des échantillons, et donc provoqué les artefacts auditifs.

9.4 Conclusion

Durant les démonstrations et les tests, nous avons été confrontés à des problèmes liés à chaque élément intervenant dans la communication : de l'horloge de la carte son à l'ordonnanceur du

[65] voir la vidéo disponible à l'URL suivante :
`http://cedric.cnam.fr/~bouill_n/FDS/CVR_FDS_Morceaux14h30.avi`

système d'exploitation, en passant par les éléments du réseau lui même.

Excepté pour le réseau que l'application ne peut contrôler s'il n'y a pas de mécanisme de réservation de ressources disponible, il est possible de trouver des solutions logicielles. Pour le problème d'ordonnancement, nous avons expérimenté nJam avec une solution appelée BOSSA [LMB02], dont il est possible de trouver plus de détails dans les publications [CBB05b, CBB05a] et dans le rapport de stage de Julien Cordry [Cor04]. Malheureusement, l'utilisation d'une telle solution est complexe et nous n'avons pas pu l'utiliser pour la démonstration de la fête de la science.

Ces difficultés techniques ouvrent des nouvelles problématiques de recherche, comme le problèmes de dérive d'horloge, montrant ainsi que les systèmes de Performances Musicales Interactives et Distribuées représentent un défi à tous les niveaux : du système d'exploitation au réseau en passant par la conception logicielle, les interfaces hommes/machines et les systèmes distribués.

Cinquième partie

Conclusion

10

Conclusion et perspectives

Sommaire

10.1 Bilan et principaux résultats obtenus

Dans cette thèse, nous nous sommes intéressés aux Applications Multimédia Interactives et Distribuées (AMID). Ces applications simulent un environnement virtuel pour mettre directement en communication des utilisateurs qui effectuent des tâches communes. Ainsi, la coopération est effectuée via un ensemble de simulacres proposés par l'application.

A partir de critères psycho-perceptifs, nous avons proposé un modèle de cohérence des données Multimédia qui permet de tenir compte des aspects temporels de l'interactivité. Les critères que nous avons utilisés sont les suivants :

- L'*instantanéité* : une action d'un utilisateur est instantanée si le délai entre la production de cette action et la notification de celle-ci à l'interface est inférieure à un délai perceptible par l'utilisateur. Ce critère fait référence aux interfaces à "effet immédiat" qui nécessitent un temps de réponse "rapide".
- La Δ *légalité* : les délais entre les actions d'un utilisateur doivent être les mêmes lorsque celles-ci sont perçues que lorsque celles-ci sont effectuées.
- La *simultanéité* : deux actions sont dites simultanées si le délai local entre la perception de ces actions est le même pour tous. Remarquons que si ce délai est suffisamment faible pour être considéré comme nul par un utilisateur, alors les deux actions ont lieu *en même temps* pour chacun des utilisateurs.

Ces critères de cohérence sont des critères liés à la perception des utilisateurs, sans qu'il y ait immédiatement de corrélation formelle avec les modèles de systèmes distribués utilisés pour décrire des critères de cohérence.

Nous avons donc utilisé et modifié le formalisme de description des Mémoires Réparties Partagées proposé par Michel Raynal et André Schiper dans [RS96] pour nous permettre de spécifier la dimension temporelle de bout en bout. La principale modification du formalisme consiste à changer la sémantique de l'opération de lecture : l'opération de lecture devient une opération de notification, permettant alors au système de contrôler la date de mise à disposition de la donnée au niveau de l'interface utilisateur.

Après avoir proposé un formalisme pour les critères décris ci-dessus, ainsi qu'un nouveau formalisme pour les applications, nous avons extrait des propriétés d'ordonnancement induites par la spécification des critères temporels. Nous avons notamment montré que le critère de causalité proposé dans la littérature était un critère incompatible avec les trois critères ci-dessus. Nous avons aussi formellement montré qu'un système respectant les critères de Δ légalité et de simultanéité était un système dans lequel l'ensemble des opérations de notification était ordonné de la même façon pour tous les utilisateurs (et donc pour toutes les répliques de l'environnement virtuel).

Les critères de Δ légalité et de simultanéité (qui définissent la *cohérence perceptive*), permettant par exemple à des utilisateurs d'effectuer des tâches fortement couplées, de type "chorégraphique". Nous avons montré que ces critères sont incompatibles avec le critère d'instantanéité, si le réseau a une latence trop importante. Nous avons alors montré l'importance du dimensionnement des applications, à l'initialisation des communications entre utilisateurs. Ce dimensionnement peut être effectué par contrôle d'accès ou par découverte des performances du réseau.

La définition formelle des AMID, des critères de simultanéité, d'instantanéité et de Δ légalité,

ainsi que de la cohérence perceptive ont permis de décrire deux types d'AMID et leurs interactions spécifiques : les PMID et les jeux vidéo répartis temps réel dans lesquels les joueurs jouent en parallèle. Nous avons par ailleurs montré formellement que les modèles de cohérence issus du domaine des Mémoires Réparties Partagées (MRP) s'appliquent mal aux AMID, et donc qu'il est nécessaire de concevoir de nouveaux protocoles de cohérence plus adaptés.

Plus généralement, nous avons fourni dans cette thèse un cadre formel d'étude et d'analyse de modèles de cohérence permettant à la fois de spécifier des propriétés temporelles générales pour les AMID et de faire le lien avec les résultats obtenus sur la cohérence des données dans le domaine des MRP.

Nous avons ensuite proposé deux protocoles de cohérence. Les expérimentations se sont focalisées sur le protocole de *cohérence perceptive* qui tient compte des délais du réseau, et fournit la simultanéité en minimisant les délais additionnels introduits. Nous avons montré que ce protocole s'applique parfaitement aux systèmes de Performances Musicales Interactives et Distribuées (PMID) ainsi qu'aux jeux vidéo distribués. Nous avons aussi montré qu'il permet d'améliorer les performances des utilisateurs, notamment grâce à la construction d'un tempo commun dans les PMID. De plus, pour les jeux interactifs et distribués le protocole permet une réduction du temps de détection de conflits, et donc de limiter les effets de la latence sur la jouabilité du jeu.

Nous avons programmé ce protocole dans *nJam*, notre prototype pour la musique interactive et distribuée. Nous avons alors montré que malgré les latences potentiellement importantes du réseau, le protocole de cohérence perceptive permet la construction d'un temps commun (le tempo) et permet de construire un métronome réparti. D'après plusieurs publications, cette fonctionnalité est jugée nécessaire, mais aucune autre solution complètement distribuée ne semble exister actuellement. Le métronome réparti est configuré par les musiciens eux-mêmes avec des paramètres issus de la notation musicale : le tempo et la métrique. De plus, ils doivent configurer le délai local, pour que celui-ci soit en phase avec le rythme et la structure du morceau joué.

Pour tester les solutions proposées, nous avons mis en œuvre deux démonstrations publiques, l'une lors des journées portes ouvertes de l'IRCAM en 2003 (*Résonances 2003*) et l'autre lors de la fête de la science 2005 au CNAM. Ces deux expériences ont montré de façon pragmatique la validité du protocole de cohérence perceptive couplé au métronome réparti. Elles ont aussi permis de mettre en lumière un ensemble de problèmes communs aux Applications Multimédia Interactives (AMID) en général, notamment liées aux Interfaces Homme/Machine, au réseau et au contrôle réparti de l'application.

Nous avons ensuite effectué des tests avec des utilisateurs pour valider la méthode d'interaction. Dans ces tests, nous avons expérimenté trois scénarios, montrant que grâce à notre système (cohérence perceptive combinée au retard local calculé et au métronome global), l'interactivité musicale distribuée reste confortable malgré des latences de plusieurs secondes. Les études de la littératures fixaient auparavant la latence maximale acceptable à 65 ms.

En d'autres termes, nous avons montré que grâce à notre système, il est possible dans un système de PMID de compenser la latence du système et du réseau pour fournir aux musiciens une interactivité qui leur permet d'utiliser leurs connaissances musicales pour interagir en temps réel à travers Internet. En effet, l'interactivité est obtenue à l'aide de l'introduction d'un retard local dans les retours des musiciens. Ce retard est choisi en fonction de la structure du morceau et de son tempo.

Nous avons alors analysé le fonctionnement du système, à partir de traces effectuées sur

l'état des tampons de réception des instances de nJam, mettant ainsi en valeur un ensemble de problèmes de recherche en système (synchronisation d'horloges, ordonnancement et adaptabilité aux ressources).

Dans le domaine des jeux vidéo, nous avons montré que le protocole de cohérence perceptive permettait d'améliorer la rapidité des mécanismes existants qui touchent à la cohérence des données, tout en s'adaptant aux latences du réseau, lors de l'exécution du jeu. En effet, les solutions similaires à notre protocole utilisent des délais pré-établis et constants, limitant les performances des systèmes, quelque soit l'environnement dans lequel le jeu s'exécute (LAN, Internet, etc.).

10.2 Perspectives

L'ensemble des travaux exposés dans cette thèse ouvre de nombreuses perspectives de travail. A plus ou moins long terme et suite à nos résultats, les avancées scientifiques à prévoir correspondent à des enjeux techniques, économiques et sociaux.

Nous avons vu que la conception d'AMID se heurte encore aujourd'hui à un certain nombre de verrous. Ainsi à court terme, les perspectives sont les suivantes :

- Augmenter l'accessibilité, le passage à l'échelle et la portée des AMID avec la conception de systèmes Pair à Pair dédiés (ou compatibles) aux AMID. Dans le cas des applications de PMID, cela pourra se faire tant pour la partie "musiciens" (voir le paragraphe 2.1.3) que pour la partie "public" [Ram05].
  Il faudra alors s'assurer de la compatibilité des solutions proposées dans cette thèse. Notamment, notre protocole de cohérence perceptive est conçu dans un modèle statique, qui consiste à considérer qu'il existe n processus. Il serait alors nécessaire de prendre en compte l'aspect dynamique des systèmes Pair à Pair et d'adapter le protocole que nous avons proposé. En effet, les processus peuvent arriver et partir du système à n'importe quel moment [Mos05].
- Tenir compte des caractéristiques temps réel des applications au niveau du système d'exploitation, en utilisant notre expérience sur l'utilisation de l'outil BOSSA [CBB05b], un environnement de développement d'ordonnanceurs. En effet, selon l'environnement dans lequel est déployée l'AMID, ses processus vont cohabiter avec d'autres, plus ou moins gourmands en ressources. BOSSA permet de contrôler l'ordonnancement du système de façon modulaire et de fournir certaines garanties.
- Nos expériences ont montré la nécessité de la conception d'un outil de gestion et d'orchestration des flux Multimédia. En effet, la coordination entre utilisateurs distants est à la fois complexe, spécifique à l'application, et spécifique au métier de chaque utilisateur manipulant l'application. Le besoin d'un intergiciel capable de contrôler et de configurer à distance un ensemble de flux Multimédia et de paramètres se fait alors ressentir fortement. De plus, nous avons vu que la méthode de mise en relation entre utilisateurs est cruciale lors du déploiement de l'application, un tel intergiciel devra alors introduire un service de découverte des utilisateurs et de gestion des contacts.
  A notre connaissance, il n'existe pas aujourd'hui de solution permettant l'intégration de telles fonctionnalités pour la conception d'AMID.
- Lors de nos démonstrations, nous avons constaté que la dérive des horloges de l'application avait un impact important sur la stabilité du protocole de cohérence perceptive.

Ce problème de dérives multiples d'horloges n'a aujourd'hui, à notre connaissance, pas de solutions efficaces, dans la mesure où le système n'est pas maître de ses horloges.

- Sachant que le système nécessite une adaptation à la latence du réseau, le jeu musical traditionnel, i.e. lorsque les musiciens sont physiquement proches, est altéré par de nouvelles contraintes temporelles. Nous avons montré par nos tests avec des utilisateurs que le système fonctionne bien dans un cadre d'interaction spécifique : lorsque la musique jouée est basée sur des boucles harmoniques et rythmiques. Il serait alors intéressant d'effectuer des tests plus développés sur les types de musiques jouables, selon quels usages et éventuellement avec des compositeurs qui introduiraient l'interaction fournie par notre système à des pièces musicales dédiées.

  Dans le contexte spécifique de l'interactivité musicale et distribuée, d'autres types d'applications peuvent être testés comme l'enseignement à distance, les auditions réparties ou encore les enregistrements en studio.

Ces perspectives promettent des résultats importants dans les domaines scientifiques de l'algorithmique répartie (modèles et protocoles de cohérence pour le P2P), des systèmes (ordonnancement, synchronisation d'horloges), des intergiciels (langages, intégration) et des IHM (nouveaux modes d'interaction).

Sur le plan architectural et technique, nous approchons de plus en plus de systèmes complètement distribués, modulaires et adaptatifs. De tels résultats permettraient de concevoir rapidement des AMID et de les déployer facilement, efficacement et à faible coût sur l'Internet, chez les particuliers et les entreprises.

A moyen terme, les résultats présentés dans cette thèse pourront être appliqués à d'autres domaines de recherche et à d'autres types d'applications que les PMID. En effet, les critères de simultanéité et d'instantanéité se retrouvent dans les systèmes qui effectuent un contrôle de processus physiques, dans lesquels la notion de temps est primordiale. C'est le cas par exemple des systèmes d'informatique industrielle comprenant des réseaux de capteurs ou du contrôle de données (comme les bus de terrain [Tho05]).

Nous avons commencé à transférer nos solutions dans d'autres domaines, comme les jeux vidéo en ligne, en montrant les gains qu'elles y apportent. Cependant, nous pouvons aussi, à moyen terme, augmenter le champ d'applications de nos solutions aux jeux répartis sur téléphones mobiles, aux systèmes de réalité virtuelle et/ou augmentée. Dans ces applications, les caractéristiques du réseau et des applications sont proches, mais différentes de notre cadre d'expérimentation : les latences très importantes dans les réseaux mobiles nécessitent des mécanismes supplémentaires au niveau de l'application pour conserver un temps de réponse raisonnable pour l'utilisateur, la mobilité introduit la notion de déconnexions, les processus s'exécutant en environnements contraints, etc.

Il faudrait dans ce cas étudier la faisabilité de nos solutions dans un environnement embarqué et dynamique. Nos solutions pourraient, compte tenu des résultats actuels, réduire les temps de réponse et améliorer l'interactivité des systèmes et ouvrir la porte à de nouveaux types d'applications. Ce travail est conséquent, dans la mesure où nos solutions sont originales (d'un point de vue formel et technique) et que les chercheurs dans les différents domaines concernés ne sont pas toujours familiers de ce type d'approche.

A plus long terme, les résultats obtenus dans cette thèse ouvrent des perspectives nouvelles pour l'élaboration d'outils de communication en temps réel entre utilisateurs, notamment pour l'enseignement, le spectacle, les relations familiales (faire du sport à l'aide d'un système de réalité augmentée par exemple), les métiers de service, la conception collaborative, etc. Nous

pourrons envisager par exemple des spectacles "chorégraphiques" (impliquant des musiciens avec des danseurs et un public, le tout réparti), des entraînements sportifs en réalité augmentée (boxe, sports de raquettes, etc.), de la maintenance de matériel en équipe (expertise à plusieurs de matériels défaillants), etc.

De plus, la société d'aujourd'hui nous amène à avoir des compétences de plus en plus spécialisées, répartissant ainsi géographiquement l'expérience et les savoirs. Ainsi, les AMID pourraient avoir un rôle économique et social important dans un avenir où probablement les transports aériens, terrestres, maritimes et spatiaux seront limités par leur coût ou par la disponibilité des énergies.

Ainsi, au même titre que le courrier ou la téléphonie, les AMID seront, si leurs usages sont acceptés par le grand public et les entreprises, le vecteur d'un bouleversement important des habitudes sociales.

A

Résumé des publications

Les travaux publiés couvrent environ 70% de la thèse, plus quelques travaux qui n'y apparaissent pas, puisqu'ils ont été initiés ou sont actuellement en cours (monitoring, ordonnancement temps réel avec BOSSA). Les 30% restant correspondent à la description des mécanismes de nJam (métronome et latence configurable) et aux tests utilisateurs.

Revue Internationale

[1] N. Bouillot et E. Gressier-Soudan. *Consistency Models for Distributed Interactive Multimedia Applications* Operating Systems Review. Octobre 2004. Volume 38, issue 4.

Cet article présente une synthèse des modèles de cohérence présentés au chapitre 4. Sans que le formalisme n'y soit introduit, on y trouve une analyse des différents types d'Applications Multimédia Interactives et Distribuées (AMID), une classification des modèles de cohérence en fonction des critères d'instantanéité/simultanéité/Δ légalité/ordre, ainsi qu'une analyse sur l'utilisation des modèles de cohérence dit "à terme".

Revue électronique Internationale

[2] N. Bouillot *The auditory consistency in distributed music performance : a conductor based synchronization* ISDM (Info/com Sciences for Decision Making) vol. 13(0), Février 2004, pp. 129-137.

Cet article est une amélioration et réécriture en anglais de l'article publié à MAJECSTIC'03, car sélectionné parmi les meilleurs de la conférence. On y trouve une description du protocole de cohérence retardé (paragraphe 6.4) basé sur la notion de chef d'orchestre. Lors de la parution de la version française de cet article [9], il n'existait aucune publication effectuant une synchronisation de flux audio pour une application de Performance Musicale Interactive et Distribuée (PMID), même centralisée. Les idées développées dans cet article ont été reprises par l'équipe Communication et Multimédia l'université de Braunschweig (Allemagne) dans [GDNW04].

Conférence Internationale

[3] N. Bouillot *Fast Event Ordering and perceptive consistency in Time Sensitive Distributed Multi-Player Games CGAMES'05 (7th International Conference on Computer Games :*

Animation, Mobile, Educational and Serious Games), p.146-152, Angoulême, Nov. 2005.

Cet article propose une adaptation du protocole de cohérence perceptive développé dans le prototype du concert réparti (nJam) décrit dans [8] et dans le paragraphe 6.2. Il est montré dans l'article que la solution proposée permet d'améliorer les performances des mécanismes de cohérence existants dans les jeux (comme le Dead Reckoning et le Trailing States Synchronization), tout en étant complémentaire avec eux.
Pour un jeu, le protocole permet d'obtenir rapidement (d'une durée égale au plus grand délai mesuré entre les joueurs) des décisions communes sur une architecture complètement distribuée. Il permet aussi de détecter efficacement les collisions entre les objets d'un jeu. Le contenu de cet article se trouve au paragraphe 6.3.2.

[4] J. Cordry N. Bouillot et S. Bouzefrane. *Performing Real-Time Scheduling in an Interactive Audio-Streaming Application 7th International Conference on Enterprise Information Systems*, pp. 140-147, Miami, Mai 2005.

Nous proposons dans cet article d'ordonnancer les processus de l'application du concert réparti en utilisant une technique d'ordonnancement temps réel. Pour cela, nous avons choisi d'utiliser BOSSA, une plate-forme qui se greffe sur le noyau Linux en vue d'intégrer de nouveaux ordonnanceurs temps réel.
Nous proposons dans cet article un retour d'expérience sur l'utilisation de BOSSA, développé notamment à l'École des Mines de Nantes, ainsi qu'une étude de performance du prototype.

[5] H.N. Locher, N. Bouillot, E. Becquet, F. Dechelle et E. Gressier-Soudan. *Monitoring the Distributed Virtual Orchestra with a CORBA based Object Oriented Real-Time Data Distribution Service* DOA'03, International Symposium on Distributed Object Application, 2003. Catagne, Sicile.

Dans cet article nous présentons un service de distribution de données orienté objet, temps réel et basé sur CORBA. Nous utilisons ce service pour contrôler l'exécution du concert réparti (c'est une métaphore de la console de l'ingénieur du son).

Conférence nationale

[6] J. Cordry N. Bouillot et S. Bouzefrane *BOSSA et le Concert Virtuel Réparti, intégration et paramétrage souple d'une politique d'ordonnancement spécifique pour une application multimédia distribuée* Dans *RTS'05 (13th International Conference on Real-Time Systems)*, Paris le 5 Avril, pp. 18-39, 2005.

Cet article est la version française de [4].

[7] N. Bouillot *Le modèle de cohérence perceptive pour les applications multimédia interactives et distribuées. CDUR 2005 (Journées Francophones sur la Cohérence des Données en Univers Réparti)*, p. 15-21, Paris le 4 Novembre 2005.

Cet article présente le modèle d'AMID (développé au chapitre 4), qui reprend et adapte le modèle proposé dans [RS96] ainsi que des idées de A. Einstein, exposées dans [Ein05].

Nous y donnons une description formelle des propriétés de simultanéité, instantanéité, Δ légalité, ainsi que de la cohérence perceptive (voir chapitre 5). Dans cet article, nous mettons formellement en valeur les relations étroites entre la cohérence perceptive (basée sur la notion de temps physique) et la latence effective du réseau. De plus, nous montrons que ce modèle décrit indirectement une relation d'ordre qui évite tous conflits entre opérations non-commutatives.

[8] N. Bouillot. *Un algorithme d'auto synchronisation distribuée de flux audio dans le concert virtuel réparti* CFSE 3, Conférence Française sur les Systèmes d'Exploitation, Octobre 2003. La Colle sur Loup, France.

Dans cet article, nous proposons un protocole de cohérence perceptive spécifique au concert réparti. Le calcul est complètement distribué et fournit une base nécessaire à l'élaboration du métronome réparti et de la compensation de la latence du réseau dans l'exécution musicale interactive et répartie. A notre connaissance, jusqu'à aujourd'hui, c'est la seule solution proposée pour synchroniser de façon distribuée des flux audio. Le protocole correspond à celui décrit dans le chapitre 6.

[9] N. Bouillot. *Une architecture pour le jeu musical réparti avec jMax et RTP* MAJECSTIC'03, Octobre 2003. Marseille, France.

Version française de [2].

Séminaire

[10] N. Bouillot et H.N. Locher *Le concert virtuel réparti sur l'Internet : vers une approche composant* Groupe de recherche Systèmes temps réel Qualité de Service. Avril 2004, Paris.

[11] N. Bouillot *Distributed Musical Interactivity* Durant la présentation des travaux de recherche du CEDRIC-CNAM. Juin 2004. Paris.

[12] N. Bouillot, H.N. Locher et E. Gressier-Soudan. *The Distributed Virtual Concert* European seminar on Free Software for Multimedia Streaming over the Internet (FSMSI'04). Juin 2004. Paris.

B

Liste des stages relatifs au concert réparti

Année	Pourcentage d'encadrement et Co-encadrant	Cycle et établissement(s)	Intitulé du stage	Nom de l'étudiant	Soutenu
2005-2006 : CNAM (PARIS 75)	100%	troisième cycle, Master Conception d'Applications Multimédia (CNAM)	"Etude et mise en œuvre du concert virtuel réparti"	Vincent Roudaut	En Novembre 2005
2004-2005 : CNAM (PARIS 75)	75% (avec E. Gressier)	troisième cycle, Master Systèmes et Applications Réparties (CNAM, Paris6, ENST)	"Application des caches Multimédia et du P2P pour le concert virtuel réparti"	Sam Ramachandra	En octobre 2005
	50% (avec E. Gressier)	troisième cycle, Mémoire d'ingénieur (CNAM)	"Les grilles de calcul pour le Multimédia"	Hans Nikolas Locher	En cours
2003-2004 : CNAM (PARIS 75)	100%	troisième cycle, Mémoire d'ingénieur (IIE-CNAM)	"Communication visuelle comme support au jeu musical réparti en temps réel"	P.A. Baudrit	En Septembre 2004
	50% (avec S. Bouzefrane)	troisième cycle, stage du DESS Développement de logiciels Sûrs (CNAM)	"Intégration de la plate-forme temps réel jMax dans BOSSA, un noyau Linux temps réel et modulaire"	Julien Cordry	En Septembre 2004
2002-2003 : CNAM (PARIS 75)	25% (avec E. Gressier)	deuxième cycle, Stage de DEST (Diplôme d'Etudes Supérieures Techniques)	"Prototypage d'une table de mixage répartie basée sur la norme de contrôle industriel TASE 2"	Hans Nikolas Locher	En Janvier 2003
	25% (avec E. Gressier)	troisième cycle, Mémoire d'ingénieur (CNAM)	"Le Multicast IP et le concert virtuel réparti"	Rémy Bonafous	En Avril 2005
	25% (avec E. Gressier)	deuxième cycle, Stage de DEST (Diplôme d'Etudes Supérieures Techniques)	"étude et développement de l'extension RTP au sein de l'outil de monitoring NTOP"	Sam Ramachandra	En Juin 2003
	100%	Troisième cycle, module Conception d'Applications Multimédia (CNAM)	"Animation d'un avatar par la Danse Kathak dans jMax"	Sama Jean-Marie	En Juin 2003

C

Exemple d'exécution de l'algorithme d'initialisation du protocole de cohérence perceptive

Nous déroulons un exemple réel pour montrer les délais introduits en conditions de test. Nous supposons que le groupe de processus qui communiquent est statique (il n'y a pas de nouveaux arrivants en cours de session). L'exemple est issu d'un test réel sur LAN et $n = 3$. Trois processus participent et chacun d'eux envoie le son qu'il génère sur une adresse multicast, lui permettant de recevoir les flux des autres.

Soit trois répliques (p_1, p_2 et p_3) qui s'envoient mutuellement des échantillons audio. Après la phase de dimensionnement, voici le tableau dont dispose chacunes d'elles :

écrivains \ lecteurs	p_1	p_2	p_3
p_1	956928	963264	957824
p_2	1146880	1153472	1147840
p_3	436352	443008	437248

Chaque colonne contient le vecteur pris par la réplique indiquée. Notons que les dates mesurées sont en base décimale et l'horloge utilisée est celle de la carte son (elle est réglée sur une fréquence de 44100 Hz).

Regardons maintenant comment se déroule l'algorithme de traitement des estampilles.

p_3 a le plus grand indice, c'est donc ce processeur qui sera choisi comme référence pour les calculs.

En regardant les différences avec le processeur p_2 pour chaque récepteur p_i, p_3 est toujours en retard par rapport à p_2. Notre référence n'est donc pas bonne.

Nous allons donc changer de référence, la plus grande n'ayant pas encore été choisie (ligne 12 de l'algorithme) : $p_2 = p_{ref}$

Les différences sont calculées comme suit :
Dans la case (p_i x p_j) la différence est $dif_{temp}[p_j]$ avec $p_{ajust} == p_i$. La colonne "Différence max" contient le tableau $dif[pp_i]$.

	pc_1	pc_2	pc_3	Différence max
p_1	189952	190016	190208	190208
p_2	0	0	0	0
p_3	710528	710592	710464	710592

Ces calculs ont été effectués par chaque réplique, voyons maintenant les délais locaux ajoutés par chacune d'elle.
Pour chaque flux, on compare la différence locale et celle maximale, ensuite on obtient le nombre d'échantillons à ajouter pour obtenir exactement l'écart maximal par rapport au flux référence. Rappelons que l'horloge a une fréquence de 44100Hz ; un incrément de l'horloge correspond à une mise à jour de l'instance de média (un échantillon audio ou *sample*).

Du point de vue de p_1 :
$ajust[p_1] := dif[p_1] - (vecteur_1[p_{ref}] - vecteur_1[p_1]) = 190208 - 189952 = 256$
$ajust[p_2] := dif[p_2] - (vecteur_1[p_{ref}] - vecteur_1[p_2]) = 0 - 0 = 0$
$ajust[p_3] := dif[p_3] - (vecteur_1[p_{ref}] - vecteur_1[p_3]) = 710592 - 710528 = 64$

Soit 256 unités de temps (ou échantillons) à ajouter au flux venant de p_1, aucun pour p_2 et 64 pour p_3.

En faisant de même, p_3 va retarder son propre flux de p_3 de 128 échantillons et p_2 le flux issu de p_1 de 192 échantillons.

256 échantillons audio correspondent à un délai de 6ms. Cet exemple nous montre donc que dans des conditions réseau exceptionnelles (en LAN par exemple) la cohérence perceptive peut être atteinte à de très faibles latences. Le protocole présenté est alors beaucoup plus performant que les protocoles de cohérence perceptive à latence fixe présentés au chapitre 5.3.

Glossaire

Δ légalité (critère) :
Les délais entre les actions d'un utilisateur doivent être les mêmes lorsque celles-ci sont perçues que lorsque elles sont effectuées.

AMID :
voir Applications Multimédia Interactives et Distribuées.

Applications Multimédia Interactives et Distribuées :
Ensemble des applications faisant collaborer en temps réel et à l'aide d'un réseau un ensemble d'utilisateurs géographiquements distants.

Artificialité (dimension) :
Selon Steve Benford *et al.* [BBRG96], ce qui caractérise l'ensemble des informations prises en compte dans l'application qui sont issues de la perception d'événements réels simulés.

BPM (Battement Par Minute) :
Unité de mesure temporelle du battement du rythme dans le domaine de la musique.

Dead Reckoning :
Technique utilisée dans les jeux vidéo répartis qui consiste à prédire les états futurs d'une donnée (comme par exemple la position d'un objet dans l'espace). Cette technique permet d'économiser de la bande passante et de conserver un certain degré de cohérence.

IGMPv2 (Internet Group Management Protocol) :
Protocole permettant aux routeurs de connaître les abonnés à un groupe Multicast.

Instantanéité (critère) :
Une action est instantanée si le délai entre la production de cette action et la notification de celle-ci à l'interface est inférieur à un délai perceptible par l'utilisateur. Ce critère fait référence aux interfaces à "effet immédiat" qui nécessitent un temps de réponse "rapide".

Isochrone :
Un signal est dit isochrone lorsque l'intervalle de temps qui sépare deux instants significatifs quelconques est théoriquement égal à l'intervalle unitaire ou à un multiple de ce dernier.

Médias continus :
Médias caractérisés par une fréquence (constante ou variable) d'écriture sur l'instance de média (comme pour l'audio ou pour les mouvements d'un avatar dans un jeu).

Médias discrets :
Dans ce type de médias (en opposition aux média continus), les écritures résultent d'une action spontanée de l'utilisateur (comme par exemple écrire du texte dans un tableau blanc partagé).

Mémoires Réparties Partagées (MRP) :
Mémoires accessibles par un ensemble de processus, qui peuvent posséder une copie locale de l'état (ou d'une partie) de la mémoire.

Métronome :
Appareil mécanique ou électronique indiquant le degré de vitesse d'exécution d'un morceau de musique. Il permet à un musicien de contrôler sa régularité de tempo.

Mesure :

Division du morceau ou du thème musical en sections d'égale durée, séparées dans la notation par la barre de mesure et contenant chacune un même nombre de temps, exprimés en notes et silences dont les valeurs s'équilibrent en un total égal.

MIC (Modulation par Impulsions et Codage) :
Le MIC est un codage adopté dans les standards de téléphonie numérique et audionumérique. Son principe consiste à numériser une grandeur analogique (le son est transformé en signal électrique à l'aide d'un microphone) en prélevant la valeur du signal (un échantillon) à des instants donnés. La mesure du signal (amplitude) est cadencée par une fréquence d'échantillonnage.

MIDI (Musical Instrument Digital Interface) :
Le MIDI est un protocole de contrôle pour l'audionumérique. Dans sa partie codage, il est possible de décrire un son de façon logique avec l'envoi d'un événement du type "la note numéro 30 a été activée avec une vélocité de force 70, en utilisant un son de piano". A la réception d'un tel événement, le synthétiseur peut recréer le son de façon proche. Le protocole MIDI est asynchrone et fut utilisé au départ sur des liaisons série entre instruments.

Multicast ASM (Any Source Multicast) :
Appellation spécifique du Multicast IP qui comprend les protocoles permettant une communication optimisée d'émetteurs vers un groupe d'hôtes. La communication est dite de n vers n.

Multicast IP :
Technique réseau issue de la pile protocolaire TCP/IP. Elle permet l'envoi d'un datagramme unique à destination d'un groupe de récepteurs.

Multicast SSM (Single Source Multicast) :
Appellation spécifique du Multicast IP qui comprend les protocoles permettant une communication optimisée d'un émetteur vers un groupe d'hôtes. La communication est dite de 1 vers n.

NMP (Network Musical Performance) :
voir PMID

Pair à Pair (P2P) :
Les solutions Pair à Pair consistent à construire une communauté virtuelle d'utilisateurs (éventuellement très grande) qui communiquent entre eux directement.

PCM (Pulse Code Modulation) :
voir MIC

PIM-SM (Protocol Independant Multicast - Sparse Mode) :
Protocole de routage Multicast ASM basé sur l'utilisation d'un point de rendez-vous.

PMID (Performance Musicale Interactive et Répartie :
AMID qui vise à simuler l'interactivité musicale entre musiciens distants, éventuellement en situation de concert.

Pulse Code Modulation (PCM) :
Méthode de codage (ou échantillonnage) de l'information qui fait varier l'amplitude des impulsions. Par exemple sur les CD audio, le signal est échantillonné à une fréquence de 44100Hz avec une quantification sur 16 bits.

RTCP (RTP Control Protocol) :
Protocole de contrôle de RTP. Il permet de maintenir des informations sur les sessions RTP : le nombre d'émetteurs et de récepteurs, la qualité de la transmission, etc.

RTP (Real-time Transport Protocol) :
Protocole de transport limitant la latence de transmission et assurant le séquencement temporel, la synchronisation intra-média, la synchronisation inter-média, le typage des données transportées et la signalisation. Il est utilisé dans les applications de streaming et de téléphonie sur IP.

Simulacre :

Ce qui a l'apparence de ce qu'il prétend être. Par exemple dans une AMID, la transmission de vidéo est un simulacre de la vision d'événements et d'objets réels au cours du temps.

Simulacres représentatifs :
Simulacres qui correspondent aux sensations que le cerveau humain infère de l'interprétation des sens. Par exemple la notion d'espace peut être simulée par l'affichage d'un environnement en trois dimensions.

Simulacres sensoriels :
Simulacres qui correspondent à la simulation de l'un des cinq sens humain, comme l'affichage d'une image ou l'émission d'un son.

Simulacres sociaux :
Simulacres qui correspondent à la simulation notions sociales, comme par exemple les smileys qui expriment virtuellement des émotions.

Simultanéité (critère) :
Deux actions sont dites simultanées si le délai local entre la perception de chacune de ces actions est le même pour tous. Remarquons que si ce délai est suffisamment faible pour être considéré comme nul par l'utilisateur, alors les deux actions ont lieu *en même temps* pour tous les utilisateurs.

SIP (Session Initiation Protocol) :
SIP est un protocole de signalisation pour la téléphonie sur IP. Son fonctionnement s'appuie sur l'architecture DNS. Il permet notamment de localiser un utilisateur à partir d'une adresse SIP (format d'une adresse mail) pour initier une transmission, comme par exemple un appel téléphonique.

Spatialité (dimension) :
Selon Steve Benford *et al.* [BBRG96], ce qui caractérise le réalisme de la manifestion des simulacres dans l'interface, comme par exemple les dimensions de l'écran (projecteur, écran de téléphone portable), la position de l'utilisateur dans l'environnement (spatialisation du son ou pas), etc.

Synchronisation inter-média :
Dans une transmission temps réel de données multimédia, c'est la synchronisation entre différents flux. Par exemple synchroniser le son avec le mouvement des lèvres.

Synchronisation intra-média :
Dans une transmission temps réel de données multimédia, c'est la 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

Téléportation (dimension) :
Selon Steve Benford *et al.* [BBRG96], ce qui caractérise dans une AMID l'ensemble des informations qui sont distantes mais qui se manifestent localement par un simulacre.

Index

Bibliographie

[ABHN92] M. Ahamad, J.E. Burns, P. W. Hutto, and G. Neiger. Causal memory. In Proceedings of the 5th International Workshop on Distributed Algorithms, pages 9–30. Springer-Verlag, 1992.

[ABK+04] Sudhir Aggarwal, Hemant Banavar, Amit Khandelwal, Sarit Mukherjee, and Sampath Rangarajan. Accuracy in dead-reckoning based distributed multi-player games. In SIGCOMM 2004 Workshops : Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04, pages 161–165, New York, NY, USA, 2004. ACM Press.

[Ank00] Willie Anku. Circles and time : A theory of structural organization of rhythm in african music. Music Theory Online, vol. 6(num. 1), January 2000.

[APS99] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control. Internet RFC 2581, 1999.

[AT00] K. Alex and S. Taylor. Using determinism to improve the accuracy of dead reckoning algorithms. In Proceedings of the simulation Technology and training Conference, Sydney Australia, 2000.

[AY96] I. Akyildiz and W. Yen. Multimedia group synchronization protocols for integrated services networks. IEEE Jounal on selected area in communications, Vol. 14(No. 1) :p. 162, 1996.

[Bal94] Bali. Grands gong kebyar des années soixante. Disque Ocora - radio france, 1994. Harmonia Mundi, C 560057/58.

[BB00] D. Bansal and H. Balakrishnan. Tcp-friendly congestion control for real-time streaming applications. Tech. Rep., Massachusetts Institute of Technology, 2000.

[BBA98] S. Bhola, G. Banavar, and M. Ahamad. Responsiveness and consistency tradeoffs in interactive groupware. In Proc. of the 7th ACM Conference on Computer Supported Cooperative Work (CSCW'98), November 1998.

[BBRG96] Steve Benford, Chris Brown, Gail Reynard, and Chris Greenhalgh. Shared spaces : Transportation, artificiality, and spatiality. In Computer Supported Cooperative Work, pages 77–86, 1996.

[BCF+98] Robin Bargar, Steve Church, Akira Fukuda, James Grunke, Douglas Keislar, Bob Moses, Ben Novak, Bruce Pennycook, Zack Settel, John Strawn, Phil Wiser, and Wieslaw Woszczyk. AES white paper : Networking audio and music using internet2 and next-generation internet capabilities. Technical report, AES : Audio Engineering Society, 1998.

[BCKR98] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol extensions for BGP-4. Network Working Group Request for Comments : RFC 2283, feb. 1998.

[Ben05] A. Benslimane, editor. Multicast Multimédia sur Internet, chapter Le multipoint pour les environnements virtuels à grande échelle, pages 349–376. Hermes Lavoisier, 2005.

[BGS04] Nicolas Bouillot and Eric Gressier-Soudan. Consistency models for distributed interactive multimedia applications. SIGOPS Oper. Syst. Rev., 38(4) :20–32, 2004.

[Bha03] S. Bhattacharyya. An overview of source-specific multicast (SSM). Network Working Group Request for Comments : RFC 3569, July 2003.

[Bol93] J. Bolot. End-to-end packet delay and loss behavior in the internet. Proc. of ACM SIGCOMM 93, 1993.

[Bon05] Rémy Bonafous. Le multicast IP pour le concert virtuel réparti. Mémoire d'ingénieur CNAM, 2005. Soutenu le 12 Avril.

[BOP94] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. Tcp vegas : new techniques for congestion detection and avoidance. SIGCOMM Comput. Commun. Rev., 24(4) :24–35, 1994.

[Bos05] Anne-Gwenn Bosser. Réplications Distribuées pour la Définition des Interactions de Jeux Massivement Multi-Joueurs. PhD thesis, université Paris 7 - Denis Diderot, 2005.

[Bou02] Nicolas Bouillot. Transport du son produit en temps réel sur les réseaux best-effort. Rapport bibliographique de DEA, 2002. DEA SIR, P6, CNAM, ENST.

[Bou03] N. Bouillot. Un algorithme d'auto synchronisation distribuee de flux audio dans le concert virtuel reparti. In Proc. of The Conference Francaise sur les Systemes d'Exploitation (CFSE'3), La Colle sur Loup, France, October 2003.

[Bou04] N. Bouillot. The auditory consistency in distributed music performance : a conductor based synchronization. ISDM (Info et com Sciences for Decision Making), (13) :pp. 129–137, Février 2004.

[Bou05a] N. Bouillot. Fast event ordering and perceptive consistency in time sensitive distributed multiplayer games. In CGAMES'2005 7th International Conference on Computer Games, pages p.146–152, Angouleme, Nov. 2005.

[Bou05b] N. Bouillot. Le modèle de cohérence perceptive pour les applications multimédia interactives et distribuées. In CDUR'05 (Journées Francophones sur la Cohérence des Données en Univers Réparti), pages p. 15–21, Paris, 2005.

[BPRS96] R. Baldoni, R. Prakash, M. Raynal, and M. Singhal. Broadcast with time and causality constraints for multimedia applications. Technical Report RR-2976, INRIA, 1996.

[Bur00] Philip Burk. Jammin' on the web - a new client/server architecture for multi-user musical performance. In ICMC (International Computer Music Conference), pages pp 117–120, 2000.

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

[Ca02] P. Cederqvist and al. Version management with CVS, 2002.

[CBB05a] Julien CORDRY, Nicolas BOUILLOT, and Samia BOUZEFRANE. Bossa et le concert virtuel réparti, intégration et paramétrage souple d'une politique d'ordonnancement spécifique pour une application multimédia distribuée. In RTS'05, 13th International conférence on Real time Systems, Paris, Avril 2005.

[CBB05b] Julien CORDRY, Nicolas BOUILLOT, and Samia BOUZEFRANE. Performing real-time scheduling in an interactive audio-streaming application. In ICEIS'05, International Conference on Enterprise Information Systems, Miami, USA, mai 2005.

[CDKR02] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. Scribe : A large-scale and decentralised application-level multicast infrastructure. IEEE Journal on Selected Areas in Communication (JSAC), Vol. 20(No. 8), 2002.

[CF78] L. M. Censier and P. Feautrier. A new solution to coherence problems in multicache systems. IEEE Trans. Computers, vol 27(12) :pages 1112–1118, 1978.

[CFG+03] Adrian David Cheok, Siew Wan Fong, Kok Hwee Goh, Xubo Yang, Wei Liu, and Farzam Farzbiz. Human pacman : a sensing-based mobile entertainment system with ubiquitous computing and tangible interaction. In NETGAMES '03 : Proceedings of the 2nd workshop on Network and system support for games, pages 106–117, New York, NY, USA, 2003. ACM Press.

[CFKJ02] E. Cronin, B. Filstrup, A. Kurc, and S. Jamin. An efficient synchronization mechanism for mirrored game architectures. In ACM Netgames'02, Braunschweig, Germany, 2002.

[CLC99] W. Cai, F. Lee, and L. Chen. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In Proceedings of the thirteenth workshop on Parallel and distributed simulation, pages 82–89. IEEE Computer Society, 1999.

[CLN+05] A. Cheok, C.C. Li, T.H.D. Nguyen, S.P. Lee T.C.T. Qui, W. Liu, D. Diaz K.S. T., and C. Boj. Social and physical interactive paradigms for mixed reality entertainment. In CGAMES'2005 7th International Conference on Computer Games, pages p.8– 15, Angouleme, Nov. 2005.

[Cor04] Julien Cordry. Ordonnancement temps réel sous linux pour le concert réparti sur internet. Rapport de stage, DESS Développement de Logiciels Sûrs (CNAM), 2004.

[CS99] D. Chen and C. Sun. A distributed algorithm for graphic objects replication in real-time group editors. In Proceedings of the international ACM SIGGROUP conference on Supporting group work, pages 121–130. ACM Press, 1999.

[CS01] Jeremy R. Cooperstock and Stephen P. Spackman. The recording studio that spanned a continent. In IEEE International Conference on Web Delivering of Music,WEDELMUSIC, Florence, Italie, 2001.

[CST+00] F. Costantini, A. Sgambato, C. Toinard, N. Chevassus, and F. Gaillard. An internet based architecture satisfying the distributed building site metaphor. In IRMA2000 Multimedia Computing Track, pages pp.151–155, Anchorage, Alaska, 21-24 May 2000. IDEA Gro.

[CSTZ05] Elaine Chew, Alexander Sawchuk, Carley Tanoue, and Roger Zimmermann. Segmental Tempo Analysis of Performances in User-Centered Experiments in the Distributed Immersive Performance Project. In Proceedings of the Sound and Music Computing Conference, Salerno, Italy, November 24-26 2005.

[CSZ+04] E. Chew, A.A. Sawchuk, R. Zimmermann, V. Stoyanova, I. Tosheff, C. Kyriakakis, C. Papadopoulos, A.R.J. Francois, and A. Volk. Distributed Immersive Performance. In Proceedings of the 2004 Annual NASM Meeting, San Diego, CA, November 22 2004.

[CTCG01] F. Costantini, C. Toinard, N. Chevassus, and F. Gaillard. Collaborative design using distributed virtual reality over the internet. In SPIE Internet Imaging, San Jose, California, 2001.

[CZS+05] E. Chew, R. Zimmermann, A.A. Sawchuk, C. Papadopoulos, C. Kyriakakis, C. Tanoue, D. Desai, M.Pawar, R. Sinha, and W. Meyer. A second report on the user experiments in the distributed immersive performance project. In 5th Open Workshop of MUSICNETWORK, 2005.

[DBFG+04] J.M. Diaz-Banez, G. Farigu, F. Gomez, D. Rappaport, and G.T. Toussaint. El compas flamenco : A phylogennetic analysis. In proceedings of BRIDGES : Mathematical Connections in Art Music, and Science, pages 61–70, Winfield, Kansas, 2004.

[DBK+01] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with Chord, a distributed lookup service. In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), pages 81–86, 2001.

[DD03] Ann Doyle and Leilani Dawson. Current practices in capturing live performance events. Published by Internet2, 2003. Internet2 Working Group Draft.

[Déc00] F. Déchelle. jmax : un environnement pour la réalisation d'applications musicales temps réel sous linux. Actes des journées d'Informatique Musicale, 2000.

[EFGK03] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. ACM Comput. Surv., 35(2) :114–131, 2003.

[EFH+98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (pim-sm) : Protocol specification. Network Working Group Request for Comments : RFC 2362, june 1998.

[Ein05] A. Einstein. On the electrodynamics of moving bodies. Annalen der Physik, 17, June 1905. translated from german.

[EKLL03] K. M. Everitt, S. R. Klemmer, R. Lee, and J. A. Landay. Two worlds apart : bridging the gap between physical and virtual media for distributed design collaboration. In Proceedings of the conference on Human factors in computing systems, pages 553–560. ACM Press, 2003.

[EM97] W. Keith Edwards and E. D. Mynatt. Timewarp : Techniques for autonomous collaboration. In CHI, pages 218–225, 1997.

[FLH+00] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic routing encapsulation (GRE). Network Working Group Request for Comments : RFC 2784, March 2000.

[FM03] B. Fenner and D. Meyer. Multicast source discovery protocol (MSDP). Network Working Group Request for Comments : RFC 3618, oct. 2003.

[Fob02] D. Fober. Audio cards clock skew compensation over a local network. Technical report, 2002.

[FOL01] D. Fober, Y. Orlarey, and S. Letz. Real time musical events streaming over internet. In First International Conference on WEB Delivering of Music (WEDLMUSIC'01), page p. 147, 2001.

[GA02] S. Gustavsson and S. Andler. Self-stabilization and eventual consistency in replicated real-time databases. In Proceedings of the first workshop on Self-healing systems, pages 105–107. ACM Press, 2002.

[GD98] L. Gautier and C. Diot. Design and evaluation of MiMaze, a multi-player game on the internet. In International Conference on Multimedia Computing and Systems, pages 233–236, 1998.

[GDNW04] X. Gu, M. Dick, U. Noyer, and L. Wolf. NMP - a new networked music performance system. In Proceedings of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), Dallas, USA, Novembre 2004. IEEE.

[GHMM96] Masataka Goto, Isao Hidaka, Hideaki Matsumoto, and Yosuke Kuroda Yoichi Muraoka. A jazz session system for interplay among all players. In ICMC Proceedings, 1996.

[GIP04] Renater : Le réseau pensant, un film sur les usages du réseau renater. Film produit et réalisé par Ridgway Organisation, Juillet 2004.

[GL00] R. Galli and Y. Luo. Mu3D : a causal consistency protocol for a collaborative VRML editor. In Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML), pages 53–62. ACM Press, 2000.

[GN02] Masataka Goto and Ryo Neyama. Open RemoteGIG :an open-to-the-public distributed session system overcoming network latency. IPSJ JOURNAL, 43(2) :299–309, February 2002. (in japanese).

[GNM97] Masataka Goto, Ryo Neyama, and Yoichi Muraoka. RMCP : Remote music control protocol, design and applications. In ICMC Proceedings, 1997.

[Gro96] Utility Communications Specification Working Group. TASE.2 services and protocol. version 1996-08. iccp inter-control centre communication protocol version 6.1. Technical Report IEC 870-6-503, IEC, 1996.

[GS02] Eric Gressier-Soudan. Contribution aux messageries industrielles. Habilitation à Diriger des Recherches, LIFL, Lille, Décembre 2002.

[HL04] Shun-Yun Hu and Guan-Ming Liao. Scalable peer-to-peer networked virtual environment. In NetGames '04 : Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, pages 129–133, New York, NY, USA, 2004. ACM Press.

[HPH03] O. Hodson, C. Perkins, and V. Hardman. Videoconferencing tool (VIC), 2003.

[HPS99] S. Hanna, B. Patel, and M. Shah. Multicast address dynamic client allocation protocol (madcap). Network Working Group Request for Comments : RFC 2730, dec. 1999.

[HPW00] M. Handley, C. Perkins, and E. Whelan. Session Announcement Protocol. Network Working Group Request for Comments : RFC 2974, oct. 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.

[IEE04a] Precision clock synchronization protocol for networked measurement and control systems, 2004. IEC 61588 First edition 2004-09 ; IEEE 1588.

[IEE04b] Precision clock synchronization protocol for networked measurement and control systems. IEEE standard, 2004. IEC 61588 First edition 2004-09 ; IEEE 1588.

[IHK04] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers : a peer-to-peer approach to scalable multi-player online games. In NetGames '04 : Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, pages 116–120, New York, NY, USA, 2004. ACM Press.

[JMS+00] J.C.Oliveira, M.Hosseini, S. Shirmohammadi, M.Cordea, E.Petriu, D. Petriu, and N.D.Georganas. Virtual theater for industrial training : A collaborative virtual environment. In Proc. 4th World Multiconference on Circuits, Systems, Communications and Computers (CSCC 2000), Greece, July 2000.

[KI98] Fabio Kon and Fernando Iazzetta. Internet music : Dream or (virtual) reality. In Proceedings of the 5th Brazilian Symposium on Computer Music, Belo Horizonte, Brazil, 1998.

[KOGC98] D. Konstantas, Y. Orlarey, S. Gibbs, and O. Carbone. Design and implementation of an ATM based distributed musical rehearsal studio. In Proceedings of ECMAST'98, 3rd Eur. Conf. on Multimedia Applications, Services and Techniques, Berlin-Germany, 26 - 28 May 1998.

[KRSD01] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel. The icecube approach to the reconciliation of divergent replicas. In Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, pages 210–218. ACM Press, 2001.

[Lam78] L. Lamport. Time, clocks, and the ordering of events in a distributed system. Commun. ACM, 21(7) :558–565, 1978.

[Lam79] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Transaction on Computers, C28(9) :690–700, 1979.

[LBB+03] H-N. Locher, N. Bouillot, E. Becquet, F. Dechelle, and E. Gressier-Soudan. Monitoring the distributed virtual orchestra with a corba based object oriented real-time data distribution service. In Proceedings DOA'03 International Symposium on Distributed Objects and Applications, Catagna, Italy, November 2003.

[LMB02] J. Lawall, G. Muller, and L.P. Barreto. Capturing os expertise in an event type system : the bossa experience. In ACM SIGOPS European Workshop 2002 (EW'2002), Saint Emilion, France, Septembre 2002.

[Loc03] Hans-Nikolas Locher. Evaluation d'une messagerie industrielle pour le contrôle de périphériques son. Rapport de stage DUT informatique (CNAM Paris), 2003.

[LST99] J. Lui, O. So, and T. Tam. Deriving communication sub-graph and optimal synchronizing interval for a distributed virtual environment system. In ICMCS, Vol. 2, pages 357–361, 1999.

[LW01] John Lazzaro and John Wawrzynek. A case for network musical performance, 2001.

[LW06] John Lazzaro and John Wawrzynek. RTP payload format for MIDI. IETF Draft, draft-ietf-avt-rtp-midi-format-15.txt, January 2006.

[LZM00] D. Li, L. Zhou, and R. Muntz. A new paradigm of user intention preservation in realtime collaborative editing systems. In Proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'00), page 401. IEEE Computer Society, 2000.

[Mau00] M. Mauve. Consistency in replicated continuous interactive media. In Proceedings of the 2000 ACM conference on Computer supported cooperative work, pages 181–190. ACM Press, 2000.

[Mil92] D. Mills. Network time protocol (version 3) specification, implementation and analysis. Network Working Group Request for Comments : 1305, March 1992.

[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. Journal of the Acoustic Society of America, 22 : p.167-173, 1950.

[ML01] D. Meyer and P. Lothberg. Glop addressing in 233/8. Network Working Group Request for Comments : RFC 3180, sept. 2001.

[Mos93] D. Mosberger. Memory consistency models. SIGOPS Oper. Syst. Rev., 27(1) :18–26, 1993.

[Mos05] Achour Mostefaoui. Tutorial : From static distributed systems to dynamic distributed systems. In CDUR'05 (Journées Francophones sur la Cohérence des Données en Univers Réparti), Paris, 2005.

[MRWJ03] M. Meehan, S. Razzaque, M. C. Whitton, and F. P. Brooks Jr. Effect of latency on presence in stressful virtual environments. In IEEE Virtual Reality, page p. 141, Los Angeles, CA, March 22 - 26 2003.

[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.

[MVHE04] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and timewarp : Providing consistency for replicated continuous applications. IEEE Transactions on Multimedia, Vol. 6(Nr. 1), 2004.

[NY05] Stéphane Natkin and Chen Yan. A typology of the relationships between real and virtual worlds. In CGAMES'2005 7th International Conference on Computer Games, pages p.223–231, Angouleme, Nov. 2005.

[OD01] N. Orio and F. Déchelle. Score following using spectral analysis and hidden markov models. In ICMC : International Computer Music Conference, La Havane, 2001.

[PAF+00] Donald Pazel, Streven Abrams, Robert Fuhrer, Daniel Oppenheim, and James Wright. A distributed interactive music application using harmonic constraint. In ICMC (International Computer Music Conference), pages pp 113–115, 2000.

[Per89] Jacques Perriault. La logique de l'usage, essai sur les machines à communiquer. flammarion, 1989.

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

[Pie93] Bois Pierre. The mugam of azerbaïdjan. Sakine Ismaïlova, Antholgie du Mugam d'Azerbaïdjan, Vol. 5 Audio CD booklet, 1993.

[PK99] K. Shin Park and R. V. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In Proceedings of the IEEE Virtual Reality, page 104. IEEE Computer Society, 1999.

[PR83] J. Postel and J. Reynolds. Telnet protocol specification. Network Working Group Request for Comments : RFC 854, may 1983.

[PSM03] N. Preguica, M. Shapiro, and C. Matheson. Semantics-based reconciliation for collaborative and mobile environments. In Proc. Conf. on Cooperative Information Systems (CoopIS), Catania, Italy, Nov. 2003.

[PSYC03] J. Mazzola Paluska, D. Saff, T. Yeh, and K. Chen. Footloose : A case for physical eventual consistency and selective conflict resolution. In Fifth IEEE Workshop on Mobile Computing Systems and Applications, Monterey, California, October 09 - 10 2003.

[PW02a] L. Pantel and L. C. Wolf. On the impact of delay on real-time multiplayer games. In Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, pages 23–29. ACM Press, 2002.

[PW02b] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In Proceedings of the 1st workshop on Network and system support for games, pages 79–84. ACM Press, 2002.

[Qin02] X. Qin. Delayed consistency model for distributed interactive systems with real-time continuous media. Journal of Software, Vol.13(No.6) :1029–1039, June, 2002.

[Ram05] Samundeswary Ramachandra. Solution P2P pour la diffusion de flux audio : contribution au concert sur internet avec pastry. Mémoire d'ingénieur CNAM, octobre 2005.

[RD01] A. Rowstron and P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages pages 329–350, Heidelberg, Germany, November 2001.

[RdR99] Curtis Roads and Jean de Reydellet, editors. L'audionumérique. DUNOD, 1999.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, pages 161–172. ACM Press, 2001.

[RHKS01] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level multicast using content-addressable networks. In Proceedings of the Third International Workshop on Networked Group Communication, page pp. 14, London, UK, Nov. 2001.

[RJ05] Lawrence A. Rowe and Ramesh Jain. Acm sigmm retreat report on future directions in multimedia research. ACM Trans. Multimedia Comput. Commun. Appl., 1(1) :3–13, 2005.

[RLS99] P. Reichl, S. Leinen, and B. Stiller. A practical review of pricing and cost recovery for internet services. Proc. 2nd Internet Economics Workshop Berlin (IEW'99), Berlin, Germany, 1999.

[RS96] M. Raynal and A. Schiper. A suite of formal definitions for consistency criteria in distributed shared memories. In Proceedings Int Conf on Parallel and Distributed Computing (PDCS'96), pages 125–130, Dijon, France, September 1996.

[RS97] D J Roberts and P M Sharkey. Minimising the latency induced by consistency control within a large scale multi-user distributed virtual reality system. In IEEE International Conference on Systems, Man and Cybernetics, Orlando, Florida, 1997.

[RS99] J. Rosenberg and H. Schulzrinne. An RTP payload format for generic forward error correction. Network Working Group Request for Comments : RFC 2733, December 1999.

[Rus59] George Russell. The Lydian Chromatic Concept of Tonal Organization for Improvisation. Concept Publishing Company, 1959.

[SBF+87] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. WYSIWIS revised : early experiences with multiuser interfaces. ACM Trans. Inf. Syst., 5(2) :147–167, 1987.

[SBM+97] R. E. Strom, G. Banavar, K. Miller, A. Prakash, and M. Ward. Concurrency control and view notification algorithms for collaborative replicated objects. In International Conference on Distributed Computing Systems, 1997.

[SC00] C. Sun and D. Chen. A multi-version approach to conflict resolution in distributed groupware systems. In International Conference on Distributed Computing Systems, pages 316–325, 2000.

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

[Sch02] N. Schuett. Effects of Latency on Ensemble Performance. PhD thesis, Stanford University, May 2002.

[Ser03] Claude Servin. Réseaux et Télécoms, cours et exercices corrigés. Dunod, 2003.

[SG00] S.Shirmohammadi and N. D. Georganas. Collaborating in 3d virtual environments : A synchronous architecture. In Proc.IEEE 9th Inter. Workshops on Enab. Technol. Infr. For Collabor. Entreprises (WETICE) Knowledge Media Networking workshop, Washington DC, June 2000.

[SK05] M. Shapiro and Nishith Krishna. The three dimensions of data consistency. In CDUR'05 (Journées Francophones sur la Cohérence des Données en Univers Réparti), pages p. 54–58, Paris, 2005.

[SRH97] A. Singla, U. Ramachandran, and J. K. Hodgins. Temporal notions of synchronization and consistency in beehive. In ACM Symposium on Parallel Algorithms and Architectures, pages 211–220, 1997.

[SSYG96] H. Sanneck, A. Stenger, K. Younes, and B. Girod. A new technique for audio packet loss concealment. Proceedings IEEE Global Internet 1996 (Jon Crowcroft and Henning Schulzrinne, eds.), pp. 48–52, (London, England), 1996.

[Ste96] R. Steinmetz. Human perception of jitter and media synchronization. IEEE Jounal on selected area in communications, Vol. 14(No. 1) :p. 61, 1996.

[SW00] 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.

[SYG03] A. El Saddik, D. Yang, and N. D. Georganas. A lightweight multi-session synchronous multimedia collaborative environment. In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Tunisia, Tunis, July 2003.

[Tho05] J.P. Thomesse. Tutorial : coherence in industrial automation. In CDUR'05 (Journées Francophones sur la Cohérence des Données en Univers Réparti), Paris, 2005.

[TRAR99] F. J. Torres-Rojas, M. Ahamad, and M. Raynal. Timed consistency for shared distributed objects. In Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing (PODC '99), pages 163–172, Atlanta, Georgia, May 1999. ACM, ACM Press.

[VM01] J. Vogel and M. Mauve. Consistency control for distributed interactive media. In ACM Multimedia, pages 221–230, 2001.

[War82] R. Warren. Auditory perception. Pergamon Press Inc, 1982.

[Wid94] Ida Widawati. Java : Tembang sunda. Maison des Cultures du Monde, 1994. recorded live at the Rond-Point/Theatre Renaud-Barrault.

[XC00] Aoxiang Xu and Jeremy Cooperstock. Real-time streaming of multichannel audio data over internet. In AES 108th convension, Paris, 2000.

[YF99] J.P. Young and I. Fujinaga. Piano master classes via the internet. In Proceedings of the International Computer Music Conference, 1999.

[ZCTL02] S. Zhou, W. Cai, S. J. Turner, and F. B. S. Lee. Critical causality in distributed virtual environments. In Proceedings of the sixteenth workshop on Parallel and distributed simulation, pages 53–59. IEEE Computer Society, 2002.

[ZDE93] Lixia Zhang, Stephen Deering, and Deborah Estrin. RSVP : A new resource ReSer-Vation protocol. IEEE network, 7(5), September 1993.

[ZKJ01] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry : An infrastructure for fault-tolerant wide-area location and routing, April 2001.

[ZZJ+01] Shelly Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiatowicz. Bayeux : An architecture for scalable and fault tolerant wide-area data dissemination. In Proc. of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), Port Jefferson, NY, 2001.

Résumé

Les Applications Multimédia Interactives et Distribuées (AMID) sur Internet ouvrent des perspectives nouvelles de travail et de loisir entre utilisateurs du réseau Internet. Dans ces applications, le sentiment de co-présence est obtenu par la numérisation et l'envoi sur le réseau de certaines de leurs actions. Cependant, le délai de transmission de ces actions introduit des ambiguïtés temporelles au niveau de l'interface. Ce problème est connu sous le nom de cohérence, mais il est généralement traité du point de vue de l'ordre dans lequel les actions sont prises en compte, et non du point de vue du temps physique. Afin de prendre en compte les contraintes psycho-perceptives liées à l'utilisation du système, nous proposons un formalisme pour l'étude et l'analyse de modèles de cohérence qui permet de spécifier les propriétés temporelles et les propriétés d'ordonnancement. Les propriétés développées dans ce formalisme sont l'instantanéité, la simultanéité, la Δ légalité et les conflits d'ordonnancement. Nous avons expérimenté un de nos protocoles de cohérence dans nJam, notre prototype de Performance Musicale Interactive et Distribuée (PMID). Pour ce type d'applications, nos résultats permettent de construire un métronome réparti partagé entre musiciens distants et cela malgré des latences importantes sur le réseau. A l'aide d'expérimentations publiques et de tests effectués avec des utilisateurs, nous montrons l'efficacité de nJam. De plus, nous montrons que nos résultats se transposent dans le domaine des jeux vidéo distribués et qu'ils peuvent améliorer les performances des mécanismes de cohérence existants dans ce domaine. Actuellement, dans le domaine des AMID, notre solution distribuée est la seule qui prend en compte les latences réseau tout en fournissant les propriétés définies par la Cohérence Perceptive.

Mots-clés : Applications Multimédia Interactives et Distribuées, modèles de cohérence, temps physique, instantanéité, simultanéité, Δ légalité, conflits d'ordonnancements, psycho-perception, Performances Musicales Interactives et Distribuées, jeux vidéo multijoueurs distribués.

Abstract

The design of Distributed Interactive Multimedia Applications (DIMA) on Internet opens new perspectives to work and to have fun in a collaborative way. In these applications, the feeling of co-presence between users is obtained with a digitizing of their actions and a network. However, network transmission delays introduce temporal ambiguities on the user interface. This problem is well-known under the name of consistency, but it is generally treated from an order point of view, instead of physical time one. Thus, we provide formal descriptions to study consistency models. It includes temporal (instantaneity, simultaneity and Δ legality) and scheduling properties and allows to take into account psycho-perceptive constraints. From these formal definitions, we propose some consistency models and consistency protocols. We implement one of them in nJam, our Networked Musical Performance system (NMP). Our results allow to build a distributed metronome shared between remote musicians, in spite of high latencies on the network. Thanks to public demonstrations and user experiments, we show the practical effectiveness of nJam. Moreover, we show that our results can improve performance of distributed games. Currently, in the DIMA domain, our results are the only ones that take into account network latencies while providing the Perceptive Consistency in a fully distributed way.

Keywords : Distributed Interactive Multimedia Applications, Consistency models, physical time, instantaneity, simultaneity, Δ legality, psycho-perception, Networked Musical Performance, distributed multiplayers games.