Diffusion Multicast et cache Multimédia pour le Concert Virtuel Réparti : Expérimentations avec Pastry/SplitStream/Past

Samundeswary Ramachandra et Nicolas Bouillot
Laboratoire CEDRIC
Conservatoire National des Arts et Métiers
292 rue St Martin
75141 Paris cedex 03

Résumé : Nous proposons une étude sur l'utilisation de Pastry et de ses services associés dans la cadre de la diffusion "Live" de musique, vers un public. Nous montrons que l'utilisation d'un tel système permet de construire rapidement une application Multimédia temps réel avec des fonctionnalités de haut niveau comme le retours rapide, mais aussi de réduire les coûts de déploiement d'applications multimédia interactives et distribuées à large échelle. Pour cela, nous avons implanté une passerelle spécifique au Concert Réparti permettant de convertir des flux Multicast IP vers le réseau Pair à Pair.
Mots-clés : Application Multimédia Interactive, Pair à Pair, Concert Virtuel Réparti, Cache Multimédia

1 INTRODUCTION

Les Applications Multimédia Interactives et Distribuées (AMID) sur Internet, comme les jeux vidéo, les systèmes de réalité augmentée, ou encore les messageries instantanées, ouvrent des perspectives nouvelles de travail et d'amusement 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.

Les contraintes imposées par ce type d'applications sont directement liées aux interactions qu'elles fournissent à leurs utilisateurs. Dans [Bouillot, 2005], nous proposons de décrire ses contraintes comme suit : la Δ légalité caractérise le besoin de régularité des mises à jours au niveau de l'interface utilisateur (comme pour du streaming audio par exemple), l'instantanéité caractérise le besoin d'un temps de réponse faible, la simultanéité le fait que les actions des utilisateurs qui sont perçues comme étant simultané par l'un d'entre eux le sera pour tous.

Ces contraintes montrent que l'interactivité est améliorée dans les AMID si le système de communication sous-jacent permet d'obtenir une gigue nulle et une latence faible.

D'un point de vue système, dans les AMID, les participants communiquent entre eux et interagissent à travers un environnement partagé (le mélange des flux audio pour le Concert Virtuel Réparti). 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 dépendantes de l'application et du degré d'interactivité qu'elle fournit aux utilisateurs.

Le Concert Virtuel Réparti est une AMID dans laquelle idéalement ces propriétés doivent être respectée. Sa conception nécessite alors un système qui permet à la fois d'établir une communication de groupe, entre les musiciens eux même et vers un public, mais aussi une communication dans laquelle les délais peuvent être contrôlé au niveau applicatif.
Au niveau de la couche IP, la technologie multicast permet aux applications de communiquer en groupe. Pour cela, l'application envoie un unique message à destination d'un adresse IP Multicast, qui abstrait le groupe. Pour recevoir les messages à destination d'un groupe, un hôte doit s'abonner au groupe auprès de son routeur.
Ainsi, les routeurs vont effectuer une mise en relation des hôtes appartenant au même groupe. Ils se coordonnent alors 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.
Pour assurer ces fonctions complexes, les routeurs doivent être compatibles à un ensemble important de protocoles. En effet, il doivent établir une liste des groupes à transmettre sur une interface de sortie donnée, établir les tables de routage, échanger les sources des flux, etc.

L'utilisation du protocole de transport RTP [Schulzrinne, 1998] avec le multicast permet d'obtenir des performances intéressante pour la conception d'AMID. En effet, RTP est basé sur UDP, ce qui le rend compatible au Multicast IP. Il est aussi conçu pour que les applications puissent communiquer en groupe (identifiants de participants) avec une gigue et une latence faible[1].

[1]En fait, RTP ne va pas introduire de gigue dans la communication

Cependant, le fonctionnement du Multicast IP impose une configuration fine et complexe au administrateurs qui gèrent les routeurs. En conséquence, il représente un coût important en maintenance. Il est aujourd'hui marginal sur la majorité des réseau et donc disponible à un nombre restreint d'utilisateurs sur l'Internet[2], même s'il est déployé un peu partout autour de la planète dans le milieu académique (le réseau MBONE[3]).

En conséquence, les AMID les plus populaires (les jeux vidéo par exemple [Bosser, 2005]) sont basées sur des architectures centralisées, qui gèrent à la fois la communication entre utilisateurs et un ensemble de services applicatifs, comme la cohérence des données. Cependant, dans certains cas (comme dans le domaine des jeux massivements multijoueurs par exemple) les architectures centralisés souffrent de problèmes de performances (temps de calcul, mémoire, bande passante réseau, etc.). De plus, les serveurs doivent idéalement être installés en des lieux stratégiques pour minimiser la latence et améliorer l'interactivité entre les utilisateurs, ce qui implique une charge financière importante pour la maintenance et la sécurité des serveurs.

Aujourd'hui les architectures Pair à Pair complètement distribuées offrent une sémantique de communication proche du multicast, en évitant les problèmes de configuration, maintenance et de coût qui sont liés à l'utilisation de serveurs. De plus, la maîtrise des nœuds dans le système permet d'ajouter des services de haut niveau, comme la gestion d'arbres Multicast, des systèmes d'archivage (ou de cache), etc.
Nous avons choisi de mettre en œuvre la diffusion de la musique produite par des musiciens distribués à l'aide d'un système Pair à Pair : Pastry. L'originalité de notre système et qu'il combine un envoi de données produites à la volée à un nombre potentiellement grand d'auditeurs avec un système de cache réparti qui leur permet de revenir en arrière et écouter le concert depuis le début.

Au paragraphe 2, nous présentons un état de l'art des solutions Pair à Pair ainsi que la description de Pastry, Past, Scribe et SplitStream (les outils que nous avons utilisés). Nous expliquerons ensuite l'architecture actuelle du concert réparti ainsi que l'application que nous avons conçu pour l'auditoire du concert au paragraphe 3. Ensuite nous exposerons nos mesures de performance au paragraphe 4 pour finalement conclure au paragraphe 5.

2 PASTRY ET 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[4]) 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 présent durant de longues périodes) et le passage à l'échelle d'un grand nombre de pairs.
Les systèmes de troisième génération (comme Chord [Dabek, 2001], CAN [Ratnasamy, 2001a] et Pastry [Rowston, 2001a] et Tapestry [Zhao, 2001]) sont généralement basés sur un système de hachage distribué (DHT) et mêlent la localisation et le routage. Il 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.

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 [Turletti, 2005]. Cela consiste à mettre en relation les utilisateurs proches dans l'environnement virtuel et permet de limiter la quantité de messages échangé 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. Nous pouvons aussi citer les systèmes de Multicast applicatif comme overcast (Multicast fiable avec source unique), Narada (dédié au streaming vidéo).

2.1 Pastry

Dans 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 1 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

car il n'effectue ni reprise sur erreur, ni contrôle de flux, ni contrôle de congestion.

[2]Remarquons que les fournisseurs d'accès Internet utilisent de plus en plus le Multicast en interne pour transmettre les flux télévisuels, mais ne permettent pas à leurs abonnés de d'exécuter leurs propres applications Multicast.

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

[4]Cette classification en générations peut varier en fonction des auteurs

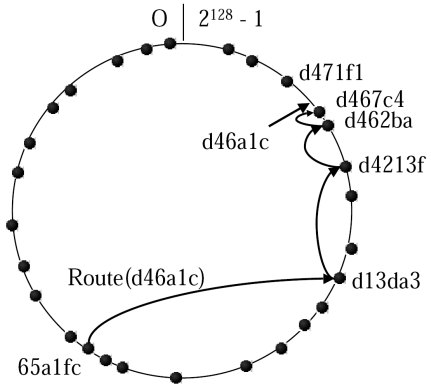

FIG. 1 – *Le routage d'un message de proche en proche dans Pastry*

identifiant de même préfixe et de moins en 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).

0	1	2	3	4	5		7	8	9	a	b	c	d	e	f
x	x	x	x	x	x		x	x	x	x	x	x	x	x	x
6	6	6	6	6		6	6	6	6	6	6	6	6	6	6
0	1	2	3	4		6	7	8	9	a	b	c	d	e	f
x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
6	6	6	6	6	6	6	6	6	6		6	6	6	6	6
5	5	5	5	5	5	5	5	5	5		5	5	5	5	5
0	1	2	3	4	5	6	7	8	9		b	c	d	e	f
x	x	x	x	x	x	x	x	x	x		x	x	x	x	x
6		6	6	6	6	6	6	6	6	6	6	6	6	6	6
5		5	5	5	5	5	5	5	5	5	5	5	5	5	5
a		a	a	a	a	a	a	a	a	a	a	a	a	a	a
0		2	3	4	5	6	7	8	9	a	b	c	d	e	f
x		x	x	x	x	x	x	x	x	x	x	x	x	x	x

FIG. 2 – *La table de routage du nœud 65a1x*

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 [Ratnasamy, 2001b] (construit au dessus de CAN), SCRIBE [Castro, 2002] (construit au dessus de Pastry) et de Bayeux [Zhuang, 2001] (Multicast SSM[5] construit au dessus de Tapestry).

Prenons l'exemple de Scribe, puisque nous avons développé un exemple de routage avec Pastry. 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.

2.2 Past et SplitStream, des services de haut niveau

Past [Rowston, 2001b] est un système de d'archivage et de cache construit au dessus de Pastry. Dans ce système, chaque fichier obtient un identifiant calculé en fonction du nom. Ensuite, le fichier (ou de l'une de ses parties) va être localisé sur le nœud ayant l'identifiant le plus proche. Le fichier peut être répliqué k fois dans le système (k est spécifié par l'utilisateur).
Past permet à l'application qui l'utilise d'insérer un fichier dans le système, d'en récupérer une copie ou de la supprimer du système. Pour cela, il utilise les techniques de localisation de Pastry.

SplitStream [Castro, 2003] est un système dédié aux environnements coopératifs permettant d'effectuer des transferts haut débit de données vers un groupe de récepteur. Il est construit au dessus de Scribe (voir le paragraphe 2). Cette sur couche à Scribe se justifie dans ce type d'environnement pour les raisons suivantes :

- L'utilisation d'un seul arbre de routage multicast pour un flux limite la bande passante de bout en bout à celle du nœud intermédiaire qui en dispose le moins.
- En cas de départ de l'un des nœuds de l'arbre, la reconfiguration de celui-ci interrompt la transmission. Cela risque notamment d'augmenter de façon significative la latence de bout en bout.

Dans le but de minimiser ces problèmes, lors de l'émission, SplitStream découpe les flux en sous flux (par un système de tour par tour) qui vont être en-

[5] La Multicast SSM est une variante du multicast dans laquelle une seule source est autorisée à émettre vers un groupe

voyés sur leurs propres arbre Multicast. La forêt ainsi construite permet d'équilibrer la charge appliquée aux nœuds des arbres, et donc de limiter l'impact d'un goulot d'étranglement.
De plus, suite au départ de l'un des nœud, seul l'un des sous arbres est affecté, minimisant ainsi la latence due à la reconfiguration.
Comme Past, les mécanismes interne à SplitStream reposent sur le système de DHT de Pastry.

3 LE CONCERT RÉPARTI ET PASTRY/SPLITSTREAM/PAST

3.1 Présentation du concert réparti

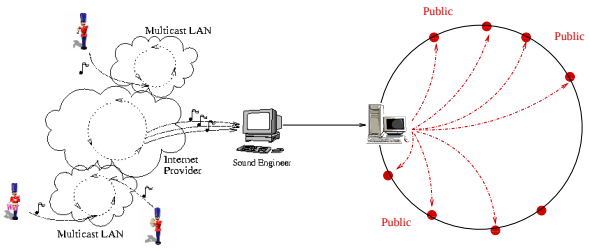

FIG. 3 – L'architecture du Concert Réparti

La figure 3 présente l'architecture générale du concert réparti. Elle construite sur l'idée qu'il y a plusieurs rôles possibles pour les utilisateurs.
La partie dédiée aux musiciens utilise actuellement le Multicast IP pour effectuer la diffusion en temps réel de la musique produite par chacun d'entre eux, leur permettant ainsi de s'entendre tous mutuellement. La transmission des données audio échantillonnées (MIC) se fait à l'aide du protocole RTP [Schulzrinne, 1998] qui repose sur le protocole UDP, qui est compatible avec le Multicast. Nous avons implanté un protocole de cohérence perceptive [Bouillot, 2005] qui synchronise de façon répartie les différents flux audio et permet aux musiciens d'avoir une perception identique du mixage des flux. Grâce à cette synchronisation, nous avons construit un métronome partagé et réparti. L'implantation s'appelle *nJam* pour "Network Jam", elle est disponible avec la distribution du logiciel audio temps réel jMax[6]
Dans la mesure où cette partie est complètement distribuée, une coordination entre les musiciens est nécessaire (réglage des volumes, positions dans l'espace dans cas de spatialisation audio, choix des morceaux à jouer, etc.)

Cela nous amène à la deuxième partie de l'architecture. 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. 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. C'est en fait une métaphore de la table de mixage, qui permet à un ingénieur du son de contrôler de façon centralisée tous ces paramètres. Le prototype que nous avons développé s'appuie sur l'outil OpenTaz, une implantation de la norme IEC TASE.2 [Locher, 2003].
Dans la mesure où l'ingénieur du son effectue le réglage, il doit être capable d'entre le mixage des différents son joués par les musiciens. La machine hébergeant la table de mixage doit donc être placée de façon stratégique pour pouvoir s'abonner aux flux RTP/RTCP envoyés par les différents musiciens. Nous avons choisi d'utiliser cette machine pour être le relais vers les auditeurs du concert, situés sur le réseau Pair à Pair. Dans la suite de cet article, nous l'appellerons la *passerelle*.

La troisième partie de l'architecture est celle dédiée aux auditeurs du concert : à l'aide du réseau Pastry et du système de diffusion associé SplitStream, les auditeurs peuvent s'abonner au flux en cours, qui sont envoyés en temps réel par la passerelle dans le réseau Pair à Pair. Le public peut ainsi assister au concert en "Live".
Dans le but d'augmenter la convivialité du système pour le public, nous avons projeté d'ajouter les fonctions classique d'un lecteur de média, a savoir le retour et l'avance rapide ainsi que l'enregistrement. Pour cela, nous utilisons Past, un système de réplication de données basé sur Pastry. Actuellement, seule la fonction d'enregistrement est implantée.

3.2 Mise en œuvre de la partie public

Dans le réseau Pair à Pair, la passerelle va jouer le rôle de la source du flux de données. Dans le réseau multicast IP, elle va être abonnée au flux, effectuer la synchronisation grâce aux informations contenues dans les paquets RTCP (voir [Bouillot, 2003] pour une explication de la synchronisation). Elle va donc effectuer les traitements suivants (nous ne précisons que ceux nécessaires à la transmission vers le public) :

1) Effectuer un abonnement au groupe multicast permettant de recevoir les flux audio des musiciens.
2) Effectuer la synchronisation pour avoir un mixage cohérent avec ce que les musiciens jouent (cette fonctionnalité n'est pas encore implantée).
3) Convertir les données audio au format utilisé dans le système Pair à Pair (nous avons utilisé un encodeur mp3 pour réduire la bande passante nécessaire).
4) Transmettre les données audio vers les clients à l'aide de SplitStream. Il devra enregistrer des blocs de données audio pour effectuer régulièrement un envoi vers le système d'archivage (PAST).

La figure 4 présente l'architecture en couches sur laquelle va reposer l'application cliente (CVRApplication) pour la

[6] `http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=11`, URL visitée en Juin 2006.

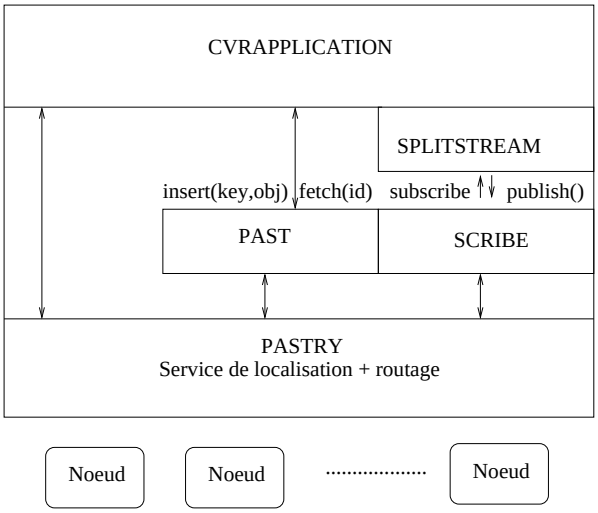

FIG. 4 – L'architecture en couches de l'application

distribution et la réplication de contenu multimédia. L'utilisation des librairies de Pastry et de SplitStream vont permettre de localiser un groupe de diffusion et de recevoir les données qui y sont émises. La librairie de Past permet quant à elle de récupérer les données du flux qui ne sont plus émise, ouvrant alors la possibilité d'effectuer des retours rapides où des enregistrements.

Les fonctions à effectuer par l'application "CVRApplication" sont donc les suivantes :

1) S'abonner au flux désiré.
2) Recevoir les données à travers SplitStream.
3) Faire une demande de données à Past si l'utilisateur fait une demande d'enregistrement ou de retour rapide. L'application devra retrouver les identifiants des différent bloc mis en cache, faire des demandes indépendente pour les récupérer, puis les réordonner.

Les interactions entre le serveur et les clients Après avoir récupéré les données audio issues de RTP et du processus de synchronisation et de mixage, la passerelle (le serveur) va agréger les données en blocs pour les envoyer au client (avec SplitStream) et au cache réparti (avec Past). La fréquence de l'envoi des données étant importante, nous alimentons Past et SplitStream avec des blocs correspondant à une durée de 10s de son (les données envoyées durant ce temps sont sauvegardées dans un buffer qui sera libéré à chaque insertion).

La figure 5 présente l'ensemble de messages échangés entre un serveur et deux clients. "client 1" arrive dès l'initialisation du serveur, il reçoit les flux en cours de SplitStream et participe à la mise en cache (avec le message "insert", le serveur lui envoie les 10 première secondes à sauvegarder). Le client 2 arrive plus tard, il s'abonne à la réception du flux et récupère le début auprès du "client 1" à l'aide du message "fetch". Pendant ce temps, le serveur continue d'envoyer le flux. "client 2" va maintenant participer à la mise en cache. Il va jouer le flux en cours tout en enregistrant le flux depuis le début.

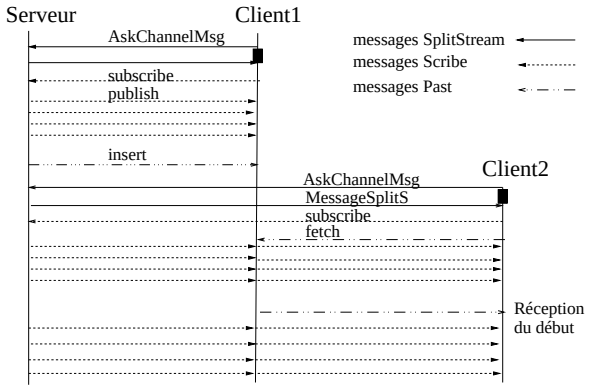

FIG. 5 – Les messages échangés entre diffuseur et auditeurs

4 MESURE DE PERFORMANCE

Nous avons effectué un test à l'aide de 6 machines ayant des processeurs à 3.2GHz, 2Go de mémoire, sur un réseau Ethernet 100Mb/s.

L'une d'elles envoie des flux RTP multicast, simulant ainsi les musiciens. La machine "passerelle" récupère ce flux, effectue les traitements décrits au paragraphe 3.2 et envoie les données vers les quatre autres machines. Durant ce test, la musique produite par les clients était clairement intelligible, montrant qu'il n'y avait pas de pertes importante de données.

Nous avons fait arriver le premier client au bout de 40 secondes, puis un toutes les 70 secondes environ. A l'aide du logiciel IPtraf[7], nous avons mesuré les débit entrant et sortant pour la passerelle ainsi que pour chacun des clients.

Machine	Objet Past récupéré	Débit (in/out kbits/s)	Débit Serveur au même moment (in/out kbits/s)
passerelle (initial)	0	0	0
passerelle (lancement)	0	750/0	750/0
client 1	4	1000/25	760/890-1000
client 2	11	1030/320	890/1600
client 3	20	1000/187	800/-
client 4	26	1041/300	900/2300

Le débit entrant initial de la passerelle (750 bit/seconde) est dû à la réception du flux Multicast provenant de la machine simulant les musiciens. En effet, les données transmises dans ce flux sont de type MIC avec une fréquence d'échantillonnage à 44100Hz pour une quantification à 16 bits, soit $44100 * 16 = 705600$ bits/secondes (environ 0,7Mb/s), sans compter les en-têtes du protocole de transport RTP.

[7] `http://iptraf.seul.org/`

Avec ces mesures, nous constatons que malgré la compression, le débit entrant des clients reste important, et même supérieur au flux initial, cela est probablement dû à l'envoi simultané par le serveur du flux lui-même avec SplitStream et des blocs du début (avec Past).
Cependant, nous constatons aussi que le débit sortant du serveur n'augmente pas linéairement, en fonction de l'arrivée des clients.

5 CONCLUSION

Dans cette article nous avons expérimenté l'utilisation de la suite logicielle Pastry/SplitStream, qui est conçue pour fournir un environnement multicast au niveau applicatif, pour la conception d'un auditoire réparti au concert sur Internet.
Nous avons de plus expérimenté Past, un système d'archivage de données basé sur Pastry, comme un cache Multimédia de données produites et consommées en temps réel. L'utilisation d'un tel outil nous a permis d'enrichir les fonctionnalités offertes aux membres du public. En effet, il permet alors à un spectateur de récupérer les données produites avant son arrivé dans le système.

Notre implantation de ces outils dans le concert réparti, ainsi que les test que nous avons effectué ont montré la faisabilité d'un tel système, avec des spectateurs répartis sur Internet. En effet, le déploiement actuel du Multicast ne permet pas d'envisager son utilisation à cette échelle. De plus, l'utilisation d'un système Pair à Pair permet de limiter les coût de maintenance de serveurs dédié, puisque la localisation, la mise en cache et le routage multicast se font de manière distribuée.

Cependant, bien que nos tests aient montré la faisabilité d'un tel système, la passerelle qui effectue la liaison entre les musiciens et le public représente un goulot d'étranglement puisqu'elle doit gérer à la fois la réception des flux Multicast issus des musiciens, la conversion des données, l'envoi de celles-ci dans le réseau Pair à Pair et la transmission des données dans le cache.
Ainsi, une architecture basé uniquement sur un réseau Pair à Pair permettrait :

- d'équilibrer la charge de flux sur l'ensemble du système (les flux mixés sont aussi disponibles chez les musiciens)
- d'éviter des conversions de protocoles (Multicast IP vers SplitStream, voir SplitStream vers Multicast IP)
- de mettre en œuvre des protocoles réseau de bout en bout adaptés à la transmission de données temps réel. Par exemple en encapsulant RTP dans SPlitStream ou Scribe.

L'interactivité entre musiciens nécessite une maîtrise des délais de bout en bout beaucoup plus importante que pour les spectateurs. Pour montrer la faisabilité d'une telle architecture, nous devrons alors comparer les performances de nJam (l'outil que nous avons développé pour les musiciens) que nous avons obtenu avec le Multicast IP avec les performances que nous obtiendrons avec un système Pair à Pair.
En effet, l'utilisation du Multicast IP avec RTP repose sur le protocole UDP, qui ne met pas les données en attente en cas de perte. Cela permet de privilégier une latence faible à une fiabilité totale. Avec une transmission basée sur Pastry, la communication entre chaque Pair est effectué avec TCP, qui pratique l'envoi en séquence des données et la reprise sur erreurs.

De plus, pour notre système, les performances peuvent encore être améliorées, en évitant d'envoyer deux fois les mêmes données à un client, comme cela est le cas s'il est élu par Past pour être le cache d'un bloc de données. En effet, en combinant Past et SplitStream, ce client pourrait enregistrer le bloc dès sa réception par SplitStream.

Les travaux que nous envisageons à cours terme concernent le passage à l'échelle de l'Internet avec des spectateurs sur ADSL. Cela nous permettrait d'étudier la validité d'un déploiement réel et de quantifier la tenue de l'application. Pour cela, nous envisageons de modéliser nos solutions avec un outil de simulation. Nous envisageons aussi d'introduire de la qualité de service dans le routage Pair à Pair [Linnolahti, 2004], dans le but d'optimiser l'équilibrage de charge.

BIBLIOGRAPHIE

[Turletti, 2005] A. Benslimane, editor. *Multicast Multimédia sur Internet*, chapitre Le multipoint pour les environements virtuels à grande échelle, pages 349–376. Hermes Lavoisier, 2005.

[Bosser, 2005] Anne-Gwenn Bosser. *Réplications Distribuées pour la Définition des Interactions de Jeux Massivement Multi-Joueurs*. PhD thesis, université Paris 7 - Denis Diderot, 2005.

[Bouillot, 2003] 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.

[Bouillot, 2005] 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.

[Castro, 2003] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream : High-bandwidth multicast in a cooperative environment. SOSP'03,Lake Bolton, New York, October 2003.

[Castro, 2002] 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.

[Dabek, 2001] 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.

[Locher, 2003] 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.

[Ramachandra, 2005] Samundeswary Ramachandra. Solution P2P pour la diffusion de flux audio : contribution au concert sur internet avec pastry. Mémoire d'ingénieur CNAM, octobre 2005.

[Ratnasamy, 2001a] 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.

[Ratnasamy, 2001b] 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.

[Rowston, 2001a] 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.

[Rowston, 2001b] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *18th ACM SOSP'01*, Lake Louise, Alberta, Canada, October 2001.

[Schulzrinne, 1998] Schulzrinne, Casner, Frederick, and Jacobson. RTP : A transport protocol for real-time applications. *RFC 1889*, 1998.

[Zhao, 2001] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry : An infrastructure for fault-tolerant wide-area location and routing, April 2001.

[Zhuang, 2001] 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.

[Linnolahti, 2004] Janne Linnolahti. QoS routing for P2P networking. *HUT T-110.551 Seminar on Internetworking*, 2004.