Transmission d'événements musicaux en temps réel sur Internet

Dominique Fober, Yann Orlarey, Stéphane Letz

Transmission d'événements musicaux en temps réel sur Internet

Dominique Fober Yann Orlarey Stephane Letz

Grame - Centre National de Création Musicale
9, rue du Garet BP 1185
69202 LYON CEDEX 01
Tél +33 (0)4 720 737 00 Fax +33 (0)4 720 737 01
[fober, orlarey, letz]@grame.fr

Résumé

Nous présentons un nouveau protocole s'appuyant sur UDP, permettant de transmettre des événements datés en temps réel et fournissant au récepteur, les moyens d'une restitution temporelle correcte. Ce protocole inclus des mécanismes permettant de compenser la latence du réseau et d'optimiser l'utilisation de la bande passante. Il prend également en compte les dérives d'horloges des différentes machines impliquées dans une transmission. Il est particulièrement adapté à la transmission d'événements musicaux tels que les messages MIDI.

1 INTRODUCTION

Les problèmes liés à la transmission en temps réel de données multimédia sur des réseaux ont été identifiés au début des années 90 en terme ressources nécessaires pour la communication temps réel [1], prenant en compte les délais de transmission, la fiabilité ainsi que les besoins en bande passante. Les flots de données multimédia sont généralement gourmants en bande passante, c'est pourquoi une attention particulière a été accordée au débit et des protocoles tels que Internet Stream Protocol (ST2) [2] ou Resource ReSerVation Protocol (RSVP) [3] [4] ont été développés pour offrir des services de transmission efficaces aux applications ayant des contraintes de qualité de service. Ces protocoles opèrent notamment grace à une politique de réservation de ressources tout au long des noeuds situés sur le chemin de distribution des données. Bien que cette politique n'affecte pas le routage en lui même, elle suppose la collaboration des routeurs afin de maintenir l'allocation de ces ressources durant toute la session. Le principe de la réservation de ressouces a également été pris en compte dans des couches de protocoles plus bas niveau tel que ATM (specifié en 1991 par CCITT I.361), et l'intégration de services temps réel dans une architecture de réseau IP-ATM a été envisagée en utilisant ST2 ou RSVP [5].

Elaboré en dehors des préoccupations de qualité de service, le protocole RTP (Transport Protocol for Real-Time Applications) [6] quant à lui, ne fournit aucune garantie concernant les services temps réel. Il opère cependant en parallèle d'un protocole de contrôle (RTCP) qui permet un contrôle de la transmission ajustable à de larges réseaux multicast, notamment par le biais d'envois réguliers de reports RTCP contenant des informations sur la qualité de la transmission. Comme RTP et RTCP ont été conçu de manière indépendante des couches de transport sous jacentes, ils peuvent constituer une solution tout à fait efficace en association avec des protocoles de réservation de ressource tel que ST2.

A condition que leur taille soit limitée, la transmission en temps réel d'événements diffère des flots de données audio ou vidéo, notamment parce qu'ils n'ont pas de besoin particulier en bande passante. La transmission de données au format MIDI peut être effectuée en beaucoup moins de temps que nécessaire pour les jouer : l'Invention à deux voix en do majeur de J.S. Bach (BWV 772) sous forme d'un fichier MIDI de 6646 octets, représentant environ 1 minute 30 de musique, peut être transmise en 2 secondes via UDP sur une liaison à 28.8 kbps. C'est pourquoi la transmission brute de fichier est suffisante pour la plupart des applications, éventuellement associée à des techniques de bufferisation dans le cas de fichiers importants,

pour permettre leur restitution avant la fin du transfert.

Cette technique n'est cependant pas utilisable quand les données sont générées en temps réel : dans ce cas, des conventions supplémentaires entre l'émetteur et le récepteur sont nécessaires. De même que pour les flots de données audio, elles devront prendre en compte les délais de transmission et la latence du réseau, mais d'une manière particulière puisqu'il faudra à réception, pouvoir restituer l'ordonnancement temporel des événements transmis.

Pour ce qui est de la bande passante et bien qu'elle ne pose problème, une attention particulière doit être donnée à l'optimisation des paquets, car des transmissions trop fréquentes d'événements pourraient alors encombrer le réseau : un plein débit MIDI par exemple, représente un paquet de 3 octets MIDI toutes les millisecondes. Dans le cas d'une transmission via UDP, le surcoût des couches de transport sous-jacentes représente plus de 91% du contenu du paquet, ce qui ne constitue pas une solution satisfaisante du point de vue efficacité.

Enfin, un des problèmes non traité par les travaux précédents est lié à la synchronisation du temps de l'émetteur et du récepteur. Dans les faits, nous ne nous intéresserons qu'à la différence de fréquence entre les horloges. En l'absence de mécanisme de correction, la dérive du temps pourra apparaitre comme une augmentation constante de la latence du réseau et à terme, empêcher une restitution correcte des événements transmis.

Il y a peu de travaux traitant de flots de données temps réel dans le contexte particulier d'événements datés. Young et Fujinaga ont présenté un travail appliqué à des master classes de piano via Internet [7]. Basé sur OTUDP (Open Transport UDP), un objet Max transportant des données via UDP, il est essentiellement axé sur la perte de paquets. La synchronisation du temps n'y est pas traitée. Un autre travail appliqué à la transmission temps réel de données MIDI sur Ethernet [8] inclus des techniques d'optimisation de la bande passante qui seront reprises dans le protocole présenté.

La suite de cet article est structurée comme suit : la section 2 présente les techniques utilisées pour compenser la latence du réseau et pour optimiser la transmission des paquets. La section 3 traite des mécanismes permettant de corriger les dérives du temps. La section 4 présente l'implémentation du protocole et des résultats expérimentaux. La section 5 conclut en soulignant les développements futurs de ce travail.

1.1 Terminologie

Quelques termes utilisés dans cet article sont définis comme suit :

restitution temporelle : le processus de restitution temporelle garantit que des événements ordonnés dans le temps seront restitués avec le même ordonnancement temporel du coté du récepteur.

la latence du transport : représente le temps de transport d'un niveau de protocole au même niveau de protocole sur la machine distante. Le niveau utilisé ici est celui du protocol proposé, ce qui signifie que la latence du transport inclus aussi bien la latence du réseau que celle des couches logicielles qui traitent les différentes couches de protocole.

le délai de restitution : soit d_a, la date d'un événement sur une machine A et d_b, la date de restitution de cet événement sur une machine B. Considérant que d_a et d_b sont exprimés dans une référence de temps commune à A et B, le délai de restitution D est définit comme d_b - d_a. Il inclut la latence du transport mais également le délai introduit pour la restitution temporelle.

le décalage d'horloges : représente la différence de temps entre deux horloges.

la dérive d'horloges : représente la différence de fréquence de deux horloges.

le décalage d'horloge apparent : c'est le décalage d'horloge augmenté de la latence de transport courante au moment de la mesure.

2 TRANSPORT ET RESTITUTION

Les mécanismes permettant de transmettre et de restituer les événements sont basés sur des travaux précédents concernant des flots de données MIDI temps réel sur Ethernet [8]. Ils reposent sur une *période de groupage* et sur une *latence maximale autorisée* pour effectuer une restitution temporelle correcte des événements transmis.

2.1 Période de groupage

La période de groupage a pour but de minimiser la fréquence des paquets transmis et d'optimiser le rapport entre la quantité de données transmise et le surcoût des couches de transport sous-jacentes. Elle représente la période pendant laquelle les événements seront accumulés avant d'être transmis sur le réseau. La valeur de la période de groupage fixe également la limite de fréquence de transmission des paquets. La figure 1 présente le nombre d'octets transmis par paquets en fonction de la fréquence des événements et pour des périodes de groupage allant de 10 à 200 ms. La fréquence des événements est exprimée en nombre d'événements par seconde. On considère que chaque événement est codé sur 5 octets. La figure 2 présente les rapports correspondants entre l'en-tête des protocoles sous-jacents et les quantités de données transmises. On considère que la taille de l'en-tête est de 44 octets.

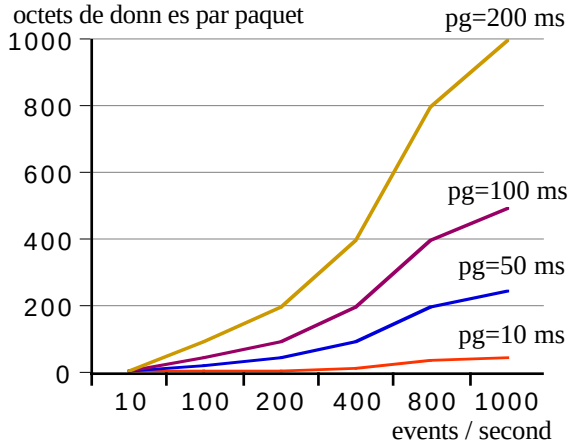

Figure 1: octets de données par paquet en fonction de la période de groupage et de la fréquence des événements.

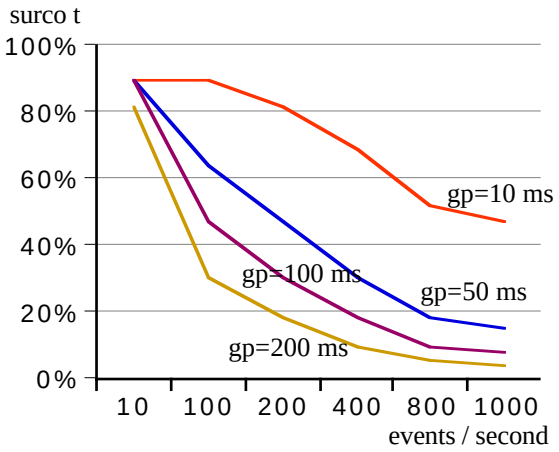

Figure 2: surcoût des protocoles sous-jacents en fonction de la période de groupage et de la fréquence des événements.

La période de groupage affecte le comportement de l'émetteur. Elle est incluse dans le délai de restitution. Ainsi que le montrent les travaux de Bolot [9], elle a également pour effet supplémentaire de minimiser la probabilité de perte de paquets et d'inversion due à des routes différentes de la source à la destination. Sa valeur sera fixée en fonction du contexte de transmission du réseau et de la fréquence des événements attendue. En particulier, cette valeur pourra être très basse si le protocole est destiné à opérer sur un réseau local (LAN).

2.2 Restitution temporelle

Comme la section suivante traite de la dérive d'horloges, nous allons considérer qu'elle est nulle pour expliquer le mécanisme de base de la restitution temporelle. Ce mécanisme repose sur une *latence maximale autorisée* afin de garantir un ordonnancement correct des événements transmis lors de leur restitution. Il opère de manière similaire aux techniques de bufferisation : la restitution des événements est différée dans le temps en fonction de la latence courante, ce qui revient à accumuler ces événements durant une période égale à la latence maximale autorisée.

Le paramètre de latence maximale autorisée affecte le comprtement du récepteur et sa valeur est également incluse dans le délai de restitution.

2.2.1 Evaluation de la variation de la latence.

En pratique, nous ne nous intéresserons uniquement à la variation de la latence du transport. Dans un premier temps, le décalage d'horloge apparent est mesuré à l'initialisation du protocole. Chaque paquet transmis comporte une date d'émission: soit A_n la date d'émission du nième paquet transmis par une machine A et B_n, la date de réception de ce paquet sur une machine B ; le décalage d'horloge apparent vu par la machine B est alors :

$$\Theta = B_o - A_o \qquad (1)$$

Il inclut alors la latence du transport.
Pour toute transmission ultérieure, la variation de la latence de A à B est évaluée comme suit :

$$\delta_n = B_n - A_n - \Theta \qquad (2)$$

2.2.2 Date des événements

Chaque événement transmis est daté par l'émetteur sous forme d'offset par rapport à la date du paquet qui le transporte. Le récepteur ajoute cet offset à la date de réception du paquet pour produire la *date locale de l'événement*. Soit B_n, la date de réception du nième paquet et o_i, l'offset de l'événement *i* relativement au paquet . La date locale de l'événement e_n^i est alors :

$$e_n^i = B_n + o_i \qquad (3)$$

Pour compenser la latence du transport, la date de restitution r_n^i de l'événement e_n^i est alors calculée comme suit :

$$r_n^i = e_n^i + L_{\max} - \delta_n \qquad (4)$$

où L_{max} est la latence maximale autorisée.
A partir de (1, 2, 3), r_n^i peut être exprimé également par :

$$r_n^i = B_o + A_n - A_o + L_{\max} + o_i \qquad (5)$$

ce qui est indépendant de la variation de la latence δ_n.
Cependant, pour que la restitution temporelle soit correcte, il faut s'assurer que la date de resitution r_n^i est supérieure ou égale à la date de réception du paquet B_n. Dérivé de (2), cela signifie que

$$B_o + A_n - A_o + L_{\max} + o_i \geq A_n + \Theta + \delta_n \qquad (6)$$

et déduit de (1), c'est équivalent à :

$$L_{\max} + o_i \geq \delta_n \qquad (7)$$

Considérant que o_i est nul pour le premier événement d'un paquet, cela signifie que la variation de la latence ne doit pas dépasser la latence maximale autorisée pour que la restitution temporelle soit correcte.

3 DERIVE D'HORLOGES

La dérive d'horloges peut être vue du coté récepteur, comme une augmentation constante de la latence du transport : soit R, le rapport de fréquence des horloges de deux machines A et B. Si l'on considère que la latence du transport est nulle, alors en déduction de (2) nous avons :

$$B_n = A_n + \Theta \qquad (8)$$

cependant, et en fonction de la dérive d'horloges, la date de réception du nième paquet sur la machine B peut être exprimée comme :

$$B_n = \frac{(A_n + \Theta)}{R} \qquad (9)$$

ce qui signifie que la variation de la latence vue par la machine B est alors :

$$\delta_n = (A_n + \Theta)\frac{1-R}{R} \qquad (10)$$

Si le rapport de fréquence R entre les horloges A et B est égal à 1, alors la variation de la latence δ_n est effectivement nulle, mais si R est plus petit que 1 (ie l'horloge B est plus rapide que l'horloge A), alors δ_n augmente avec A_n et atteindra plus ou moins vite la latence maximale autorisée, empêchant ainsi une restitution temporelle correcte des événements transmis·

3.1 Techniques relatives

La synchronisation d'horloges a fait l'objet de nombreux travaux. Parmis eux, le protocole NTP (Network Time Protocol) [10] permet la synchronisation d'horloges réparties sur Internet. Son utilisation en parallèle du protocole n'est pas envisagée en raison de certaines limitations : en particulier, la précision donnée par NTP est directement dépendante du temps mis pour l'obtenir et plusieurs heures et des douzaine de mesures sont nécessaires pour maintenir le temps avec une précision de l'ordre de la milliseconde.

La synchronisation d'horloge fait également partie des problématique de base des systèmes distribués. Dans ce cadre, un certain nombre d'algorithmes de synchronisation d'horloge ont été développés [11, 12, 13] pour permettre à des processus physiquement dispersés d'acquérir une notion du temps commune par le biais d'horloges physiques locales et d'échanges de messages via un réseau. Bien qu'en moins grand nombre, certaines recherches s'appliquent plus spécifiquement la synchronisation de fréquence des horloges [14, 15]. Tous ces travaux s'appliquent généralement à des systèmes répartis comportant de nombreux noeuds. Une approche commune consiste à utiliser des algorithmes de synchronisation tolérants aux fautes et notamment par le biais d'algorithmes de convergence interactive [16, 17] tels que le *sliding window algorithme* (SWA), le *fault-tolerant midpoint algorithme* (FTMA), l'*adaptative exponential fault-tolerant midpoint algorithme* (AEFTMA) ou le *multistep interactive convergence algorithme* (m-ICV).

La solution que nous avons adopté pour la détection de la dérive d'horloges est basée sur ces algorithmes de convergence.

3.2 Algorithme de détection

Comparé aux travaux précédents sur la synchronisation d'horloge et de fréquence, notre contexte d'opération constitue un cas particulier de plusieurs points de vue :

- seulement deux noeuds sont impliqués dans le processus de détection de la dérive d'horloges,
- il n'est pas nécessaire d'obtenir un consensus sur une base temporelle commune: chaque noeud peut estimer la dérive de son horloge de manière indépendante,
- comme la variation de la latence est utilisée pour détecter la dérive d'horloge, nous ne pouvons pas considérer certaines de ces valeurs comme étant fautives.

3.2.1 Filtrage des pics de latence

Nous avons fait un certain nombre de mesures de variation de latence en utilisant des routes de réseau différentes. Il apparait que cette variation est contenue globalement dans un éventail borné avec des pics plus ou moins fréquents. La largeur de l'éventail ainsi que l'amplitude des pics peut varier de manière importante en fonction des fournisseurs d'accès à Internet et des routes empruntées. La figure 3 montre les résultats de mesures continues effectuées via une connection modem à 42,6 kbps et un fournisseur d'accès libre (sans abonnement), à une heure réputée comme étant une heure de forte fréquentation. La mesure a été faite en envoyant des paquets datés toutes les 200 millisecondes. La légère pente de l'augmentation constante de

latence correspond à la dérive d'horloge mesurée entre les 2 machines.
La figure 4 donne la route correspondante sur le réseau.

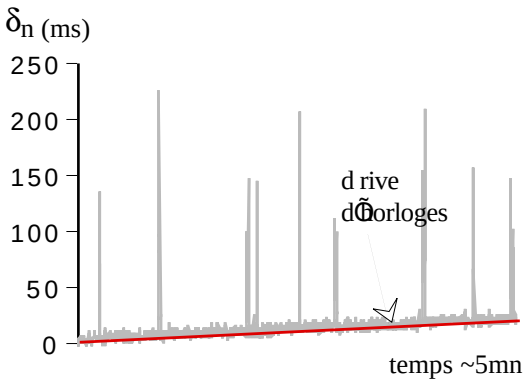

Figure 3: mesure de la variation de la latence pendant 5 mn

1	lyon2-2-58-254.dial.proxad.net
2	paris11-2-a1.routers.proxad.net
3	telehouse-6.routers.proxad.net
4	teleglobe.net
5	if-0-0.core1.paris.teleglobe.net
6	No host name found.
7	if-1-7.core1.newyork.teleglobe.net
8	ix-1-8.core1.newyork.teleglobe.net
9	jfk-core-02.inet.qwest.net
10	jfk-core-01.inet.qwest.net
11	chi-core-01.inet.qwest.net
12	chi-edge-01.inet.qwest.net
13	ar-chicago-cern.ch
14	cernh9-pos500.cern.ch
15	in2p3-fddi.in2p3.fr
16	lyon-inter.in2p3.fr
17	lyon-tif.in2p3.fr
18	grame-cisco.in2p3.fr
19	bach.grame.fr

Figure 4: routage correspondant à la figure 3

La solution proposée consiste à filtrer les pics de latence et à calculer le point de convergence des valeurs restantes. Nous avons appliqué un algorithme dérivé de FTMA à la variation de la latence pour obtenir ce que nous nommerons un *profil de dérive d'horloges*.

Appliqué à la synchronisation d'horloges, le *fault-tolerant midpoint algorithme* (FTMA) repose sur l'hypothèse qu'au plus k horloges sont fautives lors de chaque opération de resynchronisation. Dans ce cas, il faut que le système comporte au moins $n = 3k + 1$ noeuds pour tolérer des fautes k Byzantines [21]. Pour déterminer la correction d'horloge, FTMA élimine les k déviations d'horloges supérieures et inférieures et calcule ensuite la moyenne arithmétique des déviations extrêmes restantes [16].

Notre algorithme, que nous nommerons *peak-tolerant midpoint algorithm* (PTMA) opère de manière similaire avec cependant les différences suivantes :

- le vecteur ordonné de déviations d'horloge, constitué par FTMA à partir des valeurs collectées auprès des n noeuds du système, est transformé en un vecteur ordonné de variations de latence collecté dans le temps,
- le nombre de valeurs éliminées n'est pas limité aux tolérances k Byzantine,
- la moyenne arithmétique est calculée en utilisant l'ensemble des valeurs résiduelles (et non plus les seules extrêmes).

Soit w, la taille de la fenêtre temporelle de collection de la variation de la variation de latence et k, le nombre de valeurs éliminées par fenêtre. A une date t, le vecteur ordonné de la variation de latence est

$$\overline{\Delta}_t = \left[\delta_{t-w} \ldots \delta_i \ldots \delta_t\right], \delta_i \leq \delta_{i+1}$$

et le point de convergence de cette variation est calculé comme suit :

$$LV_t = \frac{\sum_{i=t-w+k/2}^{i=t-k/2} \delta i}{w-k} \qquad (11)$$

De manière intuitive, l'opération effectuée par l'algorithme peut être vue comme une sélection des variations de la latence (il élimine les variations les plus fortes et les plus faibles), mais également comme une sélection dans l'historique des variations étant donné qu'il peut ne retenir qu'un petit sous-ensemble discontinu des valeurs d'une fenêtre temporelle. La figure 5 illustre le résultat de cet algorithme appliqué aux mesures présentées ci-dessus. La ligne de

correction d'horloge représente la sortie de l'algorithme PTMA.

Pour cette application, la taille de la fenêtre temporelle est de 50 et 10 valeurs sont retenues par fenêtre. Les mesures ont été effectuées toutes les 200 ms, cela signifie qu'un sous-ensemble représentant 500 mls de variations est retenu pour 10 secondes de mesures.

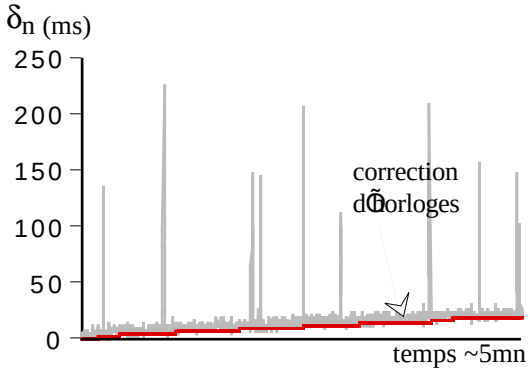

Figure 5: correction en sortie de PTMA

Notez que ce profil de dérive d'horloges peut également inclure des changements globals à long terme dans la latence du transport, ce qui satisfait également à nos besoins.

3.2.2 Lissage du profil de dérive d'horloges

Dans l'exemple précédent, les conditions de transmissions étaient plutôt bonnes comparées à d'autres mesures (effectuées via d'autres fournisseurs d'accès à Internet et donc d'autres routes sur le réseau) qui laissent apparaitre des conditions de transmission beaucoup plus chaotiques, avec des pics de latence dépassant les 3 secondes et un éventail constant pouvant atteindre 200 ms sur des périodes dépassant la largeur de la fenêtre temporelle. Nous avons donc étendu l'algorithme PTMA, que nous nommerons alors *exponential peak-tolerant midpoint average algorithm* (EPTMA) en appliquant un lissage exponentiel aux valeurs en sortie de PTMA.

Soit LV_t, la valeur du profil de dérive d'horloges à une date t, EPTMA calcule la valeur lissée corespondante comme:

$$LV^t_{EPTMA} = \alpha.LV_t + (1-\alpha).LV^{t-1}_{EPTMA} \qquad (12)$$

où le facteur de pondération α est tel que $0 \le \alpha \le 1$.

Une petite valeur pour α donne plus de poids aux valeurs passées et minimise les effets de variations brutales tout en les prolongeant dans le temps. Supposons que la latence a une valeur constante L et qu'à une date t, cette valeur passe sans transition à une valeur L'. Soit R, le rapport entre L et L'. A une date $t+\theta$, le rapport entre la valeur lissée et L' est exprimé par :

$$1 - \frac{(1-\alpha)^{\theta+1}(R-1)}{R} \qquad (13)$$

La figure 6 illustre la réponse du filtrage dans le temps pour $R = 1.3$ et différentes valeurs du facteur de pondération α.

Il apparait que même avec un facteur très petit, l'erreur de correction peut être considérée comme négligeable (au regard de nos besoins) après une minute.

La figure 7 illustre un profil de dérive d'horloges lissé, constitué à partir des pires conditions de transport mesurées. Le facteur de pondération α correspondant est de 0,01. La dernière partie du diagramme montre des distortions importantes dans le profil de dérive des horloges. Bien que ces distortions soient partiellement reportées sur le profil lissé, sa forme globale reste correcte et la distortion introduite dans la restitution temporelle est grandement minimisée. La résolution du profil de dérive lissé est la milliseconde, ce qui explique les paliers de la courbe.

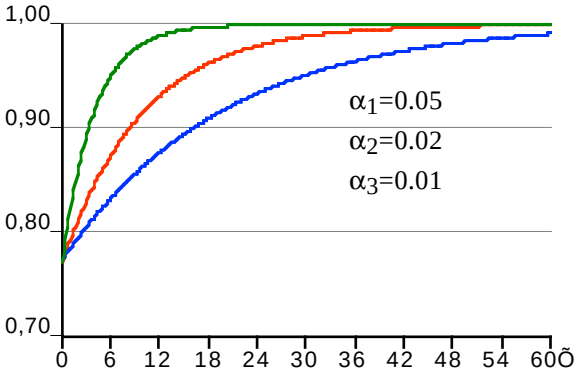

Figure 6: réponse dans le temps à une variation de la latence.

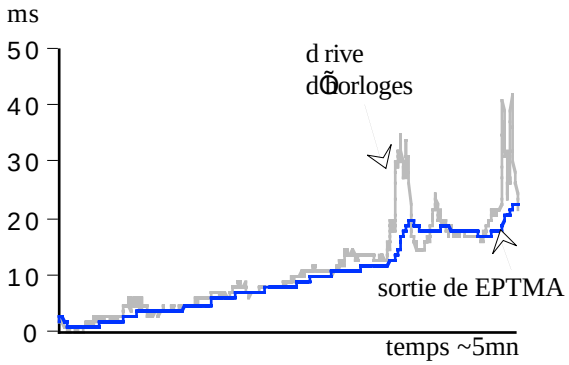

Figure 7: sortie de EPTMA comparé à PTMA

3.2.3 Correction de la dérive d'horloges

Ainsi que décrit au paragraphe 2.2, la restitution temporelle des événements est basée sur le décalage d'horloges apparent mesuré à l'initialisation du protocole. Une machine B connectée à une machine A maintient 2 dates : la date d'initialisation locale B_O et la date apparente correspondante de la machine distante A_O. Pour compenser la dérive d'horloges, la date r_n^i de restitution d'un événement e_n^i exprimée en (5) peut maintenant être corrigée de la manière suivante :

$$r_n^i = B_o + LV_{EPTMA}^n + A_n - A_o + L_{\max} + o_i \quad (14)$$

où LV_{EPTMA}^n représente la valeur lissée du profil de dérive calculée à la date *n*. C'est également équivalent à l'ajout de tout changement dans les valeurs produites par EPTMA à la date locale d'initialisation B_O de manière à maintenir un décalage d'horloge apparent constant dans la référence temporelle de A. Cette technique pour compenser la dérive d'horloges permet d'éviter le calcul d'un rapport de fréquences ainsi que les conversions de dates d'une référence temporelle à une autre. De plus, le mécanisme de compensation peut alors opérer de manière indépendante du mécanisme de restitution temporelle, ce qui facilite grandement l'implémentation du protocole.

4 IMPLEMENTATION DU PROTOCOLE

L'implémentation actuelle du protocole transmet des événements MidiShare [18, 19, 20] qui sont des événements de haut niveau, structurés et datés avec une résolution de la milliseconde. Leur typologie inclut des événements au format MIDI ainsi qu'au format MIDIFILE. La couche de transport sous-jacente utilisée est UDP (User Datagram Protocol). Deux variations sur le protocole de base sont présentées, conçues pour optimiser les implémentations LAN et WAN.

4.1 Format des paquets de base

Tous les messages ont un en-tête commun illustré par la figure 8. Les différents champs de l'en-tête ont la signification suivante:

- ID: identificateur de protocole sur 16 bits, permet d'éliminer les paquets étrangers au protocole.
- Version: numéro de version du protocole sur 8 bits, actuellement *un* (1).
- Type: identificateur de message sur 8 bits, permet de différencier les types de paquets.

Le protocole s'appuie sur 2 paquets de base : les *paquets d'événements*, qui transportent des événements datés, et les *paquets d'identification*, dont le but est à la fois de permettre l'identification des machines sur le réseau et d'assurer la continuité de la détection de dérive

d'horloges quand il n'y a pas de paquets d'événements à transmettre. Les deux types de paquets sont datés avec une résolution à la milliseconde.

4.1.1 Paquets d'événements

Le format des paquets d'événements est illustré en figure 9.

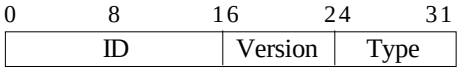

Figure 8: en-tête commun à tous les messages

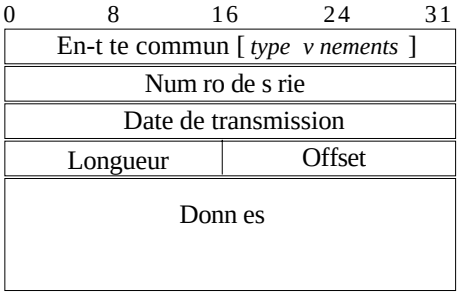

Figure 9: format des paquets d'événements

Les différents champs sont les suivants :

- En-tête commun : ainsi que décrit ci-dessus
- Numéro de série : un numéro de série unique sur 32 bits, incrémenté pour chaque paquet transmis et permettant de détecter les pertes de paquets.
- Date de transmission : une date en millisecondes sur 32 bits, exprimée dans la référence temporelle de l'émetteur.
- Longueur : le nombre de données transmises codé sur 16 bits, il représente la longueur du champ *Données*.
- Offset: une valeur sur 16 bits qui représente l'offset de début du premier événement dans le champs *Données*. La plupart du temps, sa valeur sera de 0, elle pourra être supérieure dans le cas d'un gros événement ne tenant pas dans un seul paquet. Il permet d'assurer la continuité de la lecture des données dans le cas de pertes de paquets.
- Données : un champ de longueur variable contenant les données correspondant aux événements transmis. La seule contrainte concernant le format de ces données concerne la date des événements : ainsi qu'indiqué précédemment, chaque événement doit être daté sous forme d'offset par rapport à la date du paquet qui le transporte. Cette date est actuellement expimée avec une résolution d'une milliseconde et 2 octets sont suffisants pour couvrir l'étendue des offsets possibles.

La définition actuelle du protocole ne fournit pas de mécanisme de récupération en cas de perte de paquet.

4.1.2 Paquets d'identification

Le format des paquets d'identification est illustré en figure 10. De même que pour les paquets d'événements, la date de transmission est un champ de 32 bits contenant une date exprimée en millisecondes dans le temps de référence de l'émetteur. Le *Nom* est un champ de longueur variable contenant le nom symbolique de l'émetteur.

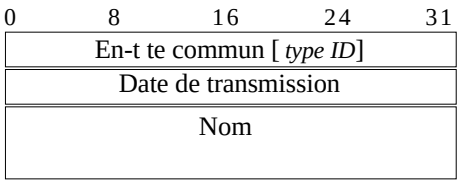

Figure 10: format d'un paquet d'identification

L'utilisation actuelle des paquets d'identification diffère légèrement dans les implémentations LAN et WAN. Toutefois, leur fonction commune est d'assurer la continuité de la détection de dérive d'horloges.

4.2 Opérations spécifiques LAN et WAN

Les implémentations pour LAN et WAN diffèrent dans leur manière d'établir une connexion et de maintenir la continuité de la détection de dérive d'horloges. Ils utilisent également des types de paquets supplémentaires pour la gestion des connexions. Enfin, pour l'implémentation LAN, les paramètres de période de groupage et de latence maximale autorisée sont définis par défaut de manière à minimiser le délai de restitution.

4.2.1 Processus de connexion

L'implémentation LAN inclus un processus de connexion automatique : à l'initialisation du protocole, un paquet d'identification est envoyé sur l'adresse de broadcast du réseau pour être émis ensuite toutes les 200 ms. Toute machine sur le réseau recevant un paquet d'identification est censée allouer les ressources nécessaires pour gérer la connexion correspondante (si elle n'existe pas encore). Quand ce paquet disparait, la machine correspondante est considérée comme inaccessible et la connexion doit être détruite.
L'implémentation WAN utilise des transactions TCP pour la gestion des connexions.

4.2.2 Continuité de la détection de dérive d'horloges

Pour maintenir la détection de dérive d'horloges dans le temps, une machine doit recevoir des paquets datés à une fréquence suffisante. L'émission régulière de paquets d'identification dans l'implémentation LAN assure cette continuité dans le temps.
L'implémentation WAN quant à elle opère de manière à minimiser la transmission de paquets : les paquets d'identification ne sont envoyés que s'il n'y a pas de paquet d'événements à transmettre.

4.2.3 Paquets optionnels

Deux types de paquets supplémentaires sont définis pour une meilleure gestion des connexions

- un paquet *Connection Refusée* : utilisé uniquement dans l'implémentation WAN et via TCP, il contient la raison du refus.
- un paquet *Bye* : envoyé lors de la fermeture d'une connexion, il contient des informations statistiques sur le déroulement de la session parmi lesquelles : le nombre de paquets perdus, la valeur du paramètre de latence maximale et le nombre de dépassements de latence. Pour l'implémentation LAN, c'est le seul moyen de fermer proprement une session.

4.3 Expérimentation

4.3.1 Paramétrage du protocole

Les expérimentations du protocole faites sur LAN et sur WAN ont utilisé les paramétrages suivants :

paramètre	*valeur LAN*	*valeur WAN*
période de groupage	10 ms	200 ms
latence maximale	10 ms	1500 ms

Les délais de restitution correspondants étaient donc de 20 ms sur LAN et au moins de 1700 ms sur WAN.
L'algorithme EPTMA faisait également usage de paramétrages différents :

paramètre	*valeur LAN*	*valeur WAN*
fenêtre temporelle	10	50
valeurs retenues	6	10
facteur de pondération	0.1	0.01

4.3.2 Mise en place d'un serveur Internet

Afin de valider le protocole, nous avons mis en place un serveur qui diffuse un flot d'événements MIDI en continu à partir de l'adresse "radio-hd.grame.fr". Basé sur le principe d'une radio, ce serveur génère son contenu musical en temps réel, à partir de n'importe quel fichier présent sur le disque dur du serveur, qui est hébergé actuellement sur une machine Linux. Des clients spécifiques sont disponibles pour Macinosh et Linux à l'adresse suivante: http://www.grame.fr/radio-hd.

5 CONCLUSION

Nous avons proposé un protocole simple pour transmettre en temps réel sur Internet et restituer des événements ordonnés dans le temps. La solution proposée présente de nombreux avantages : absence de transaction pour pouvoir opérer, indépendance des machines connectées, évaluation asymétrique de la dérive d'horloges, facilité et légèreté de l'implémentation. Toutefois, des améliorations possibles restent à explorer : en particulier concernant la dérive d'horloges, des algorithmes adaptatifs basés sur PTMA et EPTMA pourraient produire de meilleurs résultats. Le support d'adresses mulitcast permettra également d'optimiser le traffic pour des utilisations de type client / serveur.
Le code source de l'implémentation actuelle est disponible sous license LGPL (Library General Public License) à l'adresse suivante: http://www.grame.fr/MidiShare/SCPP/.

REMERCIEMENTS

Cette recherche a été conduite avec la participation de la Société Mil Productions (Villefranche/Saone - France). Nous tenons à la remercier pour sa contribution.

REFERENCES

[1] D.Ferrari. Client requirements for real-time communication services. *RFC 1193 (Status: informational)* Nov-01-1990.

[2] L. Delgrossi, L. Berger. Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+.*RFC 1819 (Status: experimental)* August 1995.

[3] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin. Resource ReSerVation Protocol (RSVP) *RFC 2205 (Status: proposed standard)* September 1997.

[4] S. Herzog. RSVP Extensions for Policy Control. *RFC 2750 (Status: proposed standard)* January 2000.

[5] M. Borden, E. Crawley, B. Davie, S. Batsell. Integration of Real-time Services in an IP-ATM Network Architecture. RFC 1821 August 1995.

[6] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Audio-Video Transport Working Group. *RFC 1889 (Status: informational)* January 1996.

[7] J.P. Young, I. Fujinaga. Piano master classes via the Internet. *Proceedings of the International Computer Music Conference 1999.* ICMA San Francisco, 1999, pp.135-137

[8] M. Wright. Implementation and performances issues with OpenSound Control. *Proceedings of the International Computer Music Conference 1998*, ICMA San Francisco, 1998, pp. 224-227

[8] D. Fober. Real-Time Midi data flow on Ethernet and the software architecture of MidiShare - *Proceedings of the International Computer Music Conference 1994*, ICMA San Francisco, 1994, pp. 447-450

[9] J.C. Bolot. End-to-end packet delay and loss behavior in the internet. *Conference proceedings on Communications architectures, protocols and applications.* ACM, 1993, pp.289-298

[10] D.L. Mills. Network Time Protocol (Version 3) Specification, Implementation. *RFC 1305 (Status: draft standard)* March 1992.

[11] T.K. Srikanth, S. Toueg. Optimal Clock Synchronization. *Journal of the ACM*, vol. 34, pp. 626–645, July 1987.

[12] B. Patt-Shamir, S. Rajsbaum. A theory of clock synchronization (extended abstract). *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994, pp. 810 - 819

[13] R. Ostrovsky, B. Patt-Shamir. Optimal and efficient clock synchronization under drifting clocks. *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, 1999, pp. 3 - 12

[14] K. Schossmaier. An interval-based framework for clock rate synchronization. *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, 1997, pp. 169 - 178

[15] K. Schossmaier, B. Weiss. An Algorithm for Fault-Tolerant Clock State and Rate Synchronization. *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 1998

[16] M.M. de Azevedo and D.M. Blough, "Fault-Tolerant Clock Synchronization for Distributed Systems with High Message Delay Variation," *1994 IEEE Workshop Fault-Tolerant Parallel and Distributed Systems. (Appears in Fault-Tolerant Parallel and Distributed Systems, D. Pradhan and D. Avresky, eds., pp. 268–277, IEEE CS Press, 1995.*) Computing, vol. 27, pp. 1–14, May 1995.

[17] M.M. de Azevedo, D.M. Blough. Multistep Interactive Convergence: An Efficient Approach to the Fault-Tolerant Clock Synchronization of Large Multicomputers. *IEEE Transactions on Parallel and Distributed Systems 9(12)*, 1998, pp. 1195-1212

[18] Y. Orlarey, H. Lequay. MidiShare : a Real Time multi-tasks software module for Midi applications - *Proceedings of the International Computer Music Conference 1989*, ICMA San Francisco, 1989, pp.234-237

[19] D.Fober, Y. Orlarey, S. Letz. Recent developments of MidiShare - *Proceedings of the International Computer Music Conference 1996,* ICMA San Francisco, 1996, pp.40-42

[20] D.Fober, Y. Orlarey, S. Letz. MidiShare joins the Open Source Softwares - *Proceedings of the International Computer Music Conference 1999* ICMA San Francisco, 1999, pp.311-313

[21] L. Lamport, R. Shostak, M. Pease. The Byzantine Generals Problem, *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 3, July 1982, pp.382-401.