Implémentation d'un flot de données MIDI sur RTP

Nicolas Falquet, Dominique Fober

IMPLÉMENTATION D'UN FLOT DE DONNÉES MIDI SUR RTP

Nicolas Falquet
INSA Lyon
Institut national des sciences appliquées
nicolas.falquet@insa-lyon.fr

Dominique Fober
GRAME
Centre national de création musicale
fober@grame.fr

RÉSUMÉ

RTP MIDI est une extension du protocole RTP (Real-Time Transport Protocol) pour la transmission de flux MIDI. Ce standard décrit un format permettant la paquetisation de toute commande pouvant apparaître légalement sur un câble DIN MIDI 1.0. Il utilise le cadre générique fourni par RTP et les outils qui lui sont associés afin de permettre une transmission de ces données MIDI en temps réel. Un tel protocole pourrait être utilisé dans le cadre d'applications interactives ou pour le streaming de fichiers. Le format propose également un mécanisme de réparation en cas de pertes de paquets lorsque le flux est transmis à travers un environnement réseau incertain. Après une brève présentation de MIDI et de RTP, l'article présente le domaine et les enjeux de RTP MIDI puis le format des paquets et les mécanismes de réparation décrits dans ce standard ainsi que, pour finir, nos travaux d'implémentation d'une bibliothèque permettant la création et l'analyse de payloads au format RTP MIDI et pouvant s'intégrer au système MidiShare.

1. INTRODUCTION

1.1. MIDI, de l'électronique à l'informatique

Le protocole MIDI (Musical Instrument Digital Interface) [9], conçu il y a plus de vingt ans, reste toujours une référence pour la représentation et la communication d'évènements musicaux. Son objectif premier était de fournir un protocole de communication standard filaire entre instruments, synthétiseurs, séquenceurs, etc. via une connectique DIN et des interfaces physiques standard. Son usage a pu s'étendre à l'utilisation d'applications logicielles comme composants de systèmes MIDI puis comme langage général de représentation d'événements musicaux, autant pour leur transmission que pour les représentations internes des applications ou que pour leur stockage grâce au Standard MIDI File (SMF) [9].

Certains travaux plus récents proposent de répondre aux critiques faites à MIDI (limitations de la bande passante, faible résolution des données, etc.) par de nouveaux protocoles mieux adaptés aux réseaux modernes. C'est par exemple le cas du protocole ZIPI [10] puis, plus récemment, d'OpenSound Control [14]. On peut aussi évoquer SASL, le langage de description de partitions de MPEG-4 Structured Audio [5].

Une suite logique au développement du standard MIDI du monde de l'électronique au monde de l'informatique était la mise au point de mécanismes de transmission à distance de ces signaux qui ne dépendent plus, cette fois-ci, d'interfaces physiques spécifiques et qui puissent, en revanche, profiter de l'universalité des réseaux IP. C'est l'objectif de RTP MIDI. D'autres travaux avaient déjà été menés dans cet objectif [2].

1.2. RTP : Real-Time Protocol

Le protocole RTP (Real-Time Transport Protocol) [13] est un protocole conçu pour le transport de bout en bout de données de type temps réel au dessus de TCP ou d'UDP, en mode unicast ou en mode multicast. Il peut s'adapter à la fois à des applications interactives à faible latence (comme la téléphonie ou la vidéoconférence) et à des applications de transmission de contenu (comme le streaming de fichiers). Il n'assure pas, en soi, de garanties quant à la qualité du service fourni (temps de latence, gigue, perte de paquets, ...). Ces aspects de qualité de service doivent être assurés par les couches basses du réseau (Figure 1).

RTP est un protocole générique dont les extensions pour différents formats audio ou vidéo doivent être définies à part. De nombreux formats ont déjà été spécifiés sous la coordination du groupe de travail AVT (Audio / Video Transport) de l'IETF (Internet Engineering Task Force). Ces extensions spécifient, entre autre, le format des “payloads” qui seront inclus dans les paquets RTP et qui contiennent les données utiles, spécifiques au format.

La norme RTP définit, en plus de la description des paquets RTP, le protocole de contrôle RTCP (RTP Control Protocol) permettant l'échange entre les différents acteurs d'une session RTP de données de contrôle (rapports sur la qualité de réception, contrôle et informations sur les participants, etc.).

Plusieurs autres protocoles peuvent être utilisés en complément de RTP : SIP (Session Initiation Protocol) peut, par exemple, permettre d'établir la connexion entre les différentes parties d'une session et d'en négocier les paramètres, SDP (Session Description Protocol) permet de décrire les paramètres d'une session, etc.

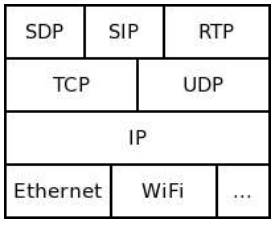

Figure 1. RTP et les outils associés dans la pile IP.

2. RTP MIDI

Une proposition de standard décrivant un format de payload RTP pour les signaux MIDI et nommé “RTP MIDI” est en cours d'élaboration depuis 2001 au sein de l'AVT. Les deux drafts présentant d'une part le standard RTP MIDI [6] et d'autre part un guide d'implémentation [7] ont maintenant atteint un état stable et vont bientôt faire l'objet d'un “Last Call” de la part de l'IETF.

2.1. Domaines d'application

Les domaines dans lesquels ce type de protocole peut trouver son utilité sont les suivants :

- Des instruments ou des applications MIDI pourraient utiliser RTP comme protocole de transport au dessus de réseaux IP. L'intérêt repose ici dans l'universalité du protocole IP (Niveau 3) qui peut lui-même s'adapter à une grande variété de protocoles réseaux de Niveau 2 : Ethernet, WiFi, FireWire, etc. La qualité de service du flux dépendra des caractéristiques de ces protocoles de Niveau 2. Ceci permettrait, d'une part, de transmettre des flux MIDI sur des interfaces plus génériques, plus communes et plus performantes que les interfaces physiques MIDI mais également d'utiliser les propriétés génériques de RTP pour imaginer une transmission combinée de signaux MIDI, de vidéo et d'audio utilisant également RTP mais pour d'autres formats et ce dans la même session RTP ou sur le même réseau.

- Des musiciens pourraient interagir à distance au coeur d'un système MIDI comme ils le feraient s'ils se trouvaient dans la même pièce. Le système MIDI étant, cette fois-ci, distribué à travers Internet. On pourrait alors parler de performance musicale en réseau. Un tel cas de “Network Musical Performance” est décrit dans [8]. L'application s'apparenterait à une application de téléphonie ou de conférence pour des signaux MIDI.

- Les signaux MIDI pourraient offrir la possibilité d'un streaming de contenu musical à faible coût. D'après [9], une séquence MIDI classique peut nécessiter typiquement moins de 10 ko de données par minute. Ces signaux pourraient ensuite être utilisés par une méthode de synthèse standard performante telle que le MPEG-4 Structured Audio [5].

2.2. Aperçu des fonctionnalités.

Le standard RTP MIDI s'est d'abord appelé, dans ses premières versions, MIDI Wire Protocol Packetization (MWPP). Ce nom traduit bien la volonté première du protocole : proposer la simulation, à travers le réseau, d'un câble MIDI virtuel. Nous appellerons “flux” l'ensemble des commandes d'un unique câble MIDI virtuel de 16 canaux.

Les commandes émises sont encapsulées dans un paquet RTP. Elles peuvent y être encapsulées une par une ou groupées à plusieurs dans un même paquet. Ceci dépend du domaine d'application et peut être soumis à un compromis entre bande passante et temps de latence.

Un flux, comme un câble MIDI, est unidirectionnel. Il est cependant possible de combiner plusieurs flux au sein d'une même session RTP MIDI. Ces différents flux peuvent alors représenter différents ports MIDI ou bien encore permettre une communication bidirectionnelle entre les parties.

En plus de la simple transmission des commandes, RTP MIDI intègre un mécanisme de marquage temporel des commandes pouvant permettre, à la réception, de reconstituer correctement leur séquence. Contrairement à MIDI, RTP MIDI rend également possible la transmission de commandes pouvant être considérées comme simultanées.

2.3. Enjeux

RTP n'offre pas, nous l'avons déjà dit, en soi, de mécanismes de gestion de la qualité de service et dépend, pour cela, du réseau sous-jacent. Ceci implique, si l'on se trouve sur un transport UDP par exemple, que certains paquets d'un flux RTP peuvent être perdus, que d'autres peuvent arriver dans le désordre ou peuvent être dupliqués.

Or, dans le cas de transmissions de paquets avec des intentions temps réel, il est recommandé d'utiliser UDP (User Datagram Protocol) [11] comme protocole de transport : celui-ci ne garantit pas non plus un transfert fiable des paquets mais les mécanismes de TCP (Transmission Control Protocol) [12] qui eux le fournissent le font au détriment d'une transmission “au plus vite” des paquets.

Dans le cas de signaux MIDI, les conséquences d'une perte de paquet sont décrites, dans le standard RTP MIDI, comme pouvant être de deux types :

- des artefacts transitoires (*transient artifacts*) qui correspondent à des dégradations du flux MIDI immédiates mais de courte durée. L'exemple canonique est le cas de la perte d'un paquet contenant une commande Note On : la perturbation ne durera que jusqu'à la réception de la commande Note Off correspondante. C'est un cas d'artefact transitoire.

- des artefacts à durée indéterminée (*indefinite artifacts*) qui produisent des dégradations de longue durée. La perte d'un paquet contenant une commande Note Off, par exemple, pourra faire sonner une note indéfiniment. Une perte de commande Control Change pour le controlleur 7 (Channel Volume) pourra altérer le volume d'un canal de manière durable. Ce sont deux exemples d'artefacts à durée indéterminée.

Si les artefacts transitoires sont gênants et devraient être évités au maximum, les artefacts à durée indéterminée sont en revanche intolérables. Un des objectif de RTP MIDI était donc, dans le cas d'une transmission de paquets non fiable, de limiter, dans la mesure du possible, les effets des artefacts transitoires et d'éliminer complètement les artefacts à durée indéterminée.

Dans le standard, ces objectifs sont assurés par la présence au sein de chaque payload d'un journal de réparation (*recovery journal*) dont le principe et la structure seront décrits en partie 4.

3. FORMAT DU PAYLOAD

Les paquets RTP MIDI suivent la structure présentée Figure 2. La première partie du paquet (RTP header) correspond à l'entête RTP du paquet. Cette partie est commune à tout paquet RTP et est décrite dans [13]. Très peu d'éléments sont redéfinis dans la spécification de RTP MIDI. Cet entête permet notamment de numéroter le paquet au sein du flux (ce qui permettra de détecter les paquets perdus ou désordonnés). Il permet aussi de dater le paquet.

Les deux parties suivantes, qui elles sont spécifiques à RTP MIDI et définies dans [6], constituent ce que l'on appelle le “payload” du paquet RTP, sa charge utile.

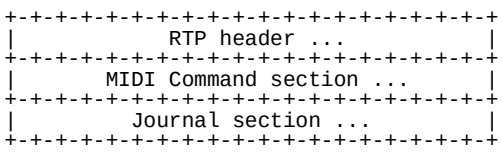

Figure 2. Structure d'un paquet RTP MIDI.

3.1. Section des commandes MIDI

La deuxième partie (MIDI Command section) fournit l'essentiel du service de RTP MIDI. Elle code une liste de commandes MIDI. Ces commandes peuvent correspondre à toute commande pouvant apparaître légalement sur un câble DIN MIDI 1.0. Les commandes d'un unique flux RTP MIDI représentent, comme celles d'un flux sur câble DIN, les commandes d'un groupe de 16 canaux ainsi qu'un ensemble de commandes systèmes. Sa structure (Figure 3) comporte un entête de 1 ou 2 octets dont le champ `LEN` peut coder une longueur de section allant jusqu'à 4095 octets. S'ensuit la liste des commandes à proprement parler.

À chaque commande est associée une valeur de delta time permettant de lui reconstituer une date. Ce delta time est codé en longueur variable comme les delta times des SMF. La date de la n^{ième} commande du paquet est égale à la somme de la date de l'entête RTP et des delta times 0 à n. Différentes sémantiques peuvent être associées à ces dates en fonction de la configuration de la session : date d'exécution de la commande, date du premier ou du dernier bit de la commande sur le câble virtuel, etc.

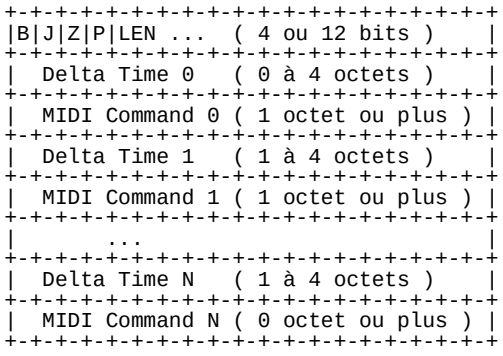

Figure 3. Structure de la section MIDI Command.

Chaque commande MIDI est codée telle quelle dans un champ “MIDI Command”, cependant, dans le cas de commandes System Exclusive, qui peuvent être relativement longues, RTP MIDI propose également un mécanisme permettant de fragmenter ces commandes sur plusieurs payloads.

4. JOURNAL DE RÉPARATION

La troisième et dernière partie du paquet (Journal section) est la section en charge de réaliser les impératifs de réparation du flux qui ont été fixés au paragraphe 2.3.

4.1. Principe

Cette section code, dans le paquet i, un historique du flux jusqu'au paquet i-1. En cas de perte de paquets, le récepteur utilise ces données sur l'historique pour réparer les dégradations du flux dues aux commandes perdues.

La réparation du flux ne nécessite donc pas de retransmission de la part de l'émetteur lorsqu'un ou plusieurs paquets se perdent. Le récepteur, qui ne détecte, dans les faits, une perte de paquet qu'au moment où il reçoit le paquet suivant (il s'aperçoit d'un saut dans la numérotation des paquets) peut alors réparer le flux immédiatement : la réparation se fait au plus tôt. Ce schéma pourrait également s'avérer pertinent dans le cas d'une diffusion multicast faisant intervenir un seul émetteur et plusieurs récepteurs : chaque récepteur pourra réparer les dégradations dues aux paquets qui ne lui sont pas parvenus indépendamment des pertes subies par les autres récepteurs et sans nécessiter d'intervention de la part de l'émetteur. Les principes mis en jeu dans ce système de journal de réparation s'apparentent aux principes de Forward Error Correction.

Si le journal codait simplement un historique brut du flux (par exemple, la liste de toutes les commandes ayant déjà été transmises), sa taille exploserait rapidement. Deux mécanismes permettent de limiter la taille de ce journal de réparation.

Premièrement, les informations codées ne représentent que le minimum permettant de réparer le flux de possibles artefacts à durée indéterminée. Plus qu'un simple log elles codent généralement l'état du flux. Un travail important a été fourni sur la conception d'un format optimal de stockage de ces informations. L'historique des commandes Note Off, par exemple, est codé sur un champ de bits où chaque bit correspond à une note et indique si elle doit être éteinte. Autre exemple, pour un même contrôleur, une seule information au maximum figure dans le journal.

Deuxièment, un paquet de “checkpoint” peut être convenu entre l'émetteur et le récepteur afin de limiter l'étendue de l'historique couvert par le journal de réparation. Dans ce cas, la section de journal du paquet i codera seulement des informations sur les commandes transmises depuis le paquet de checkpoint et jusqu'au paquet i-1. La mise à jour de ce checkpoint sera faite via le protocole RTCP, protocole de contrôle de RTP.

4.2. Structure

Le journal de réparation a une structure hiérarchique (Figure 4). L'élément de base du journal est le chapitre qui code les informations correspondant à un type de

commande en particulier. Le journal de réparation se compose, sur premier niveau, d'une série de chapitres concernant les commandes systèmes (System Exclusive, System Common et System Real-Time) puis d'un journal pour chacun des 16 canaux. Ce journal de canal se subdivise à son tour en une série de chapitres traitant des commandes d'un certain type pour ce canal (Note Off, Note On, Key Pressure, Control Change, Program Change, Channel Pressure et Pitch Bend).

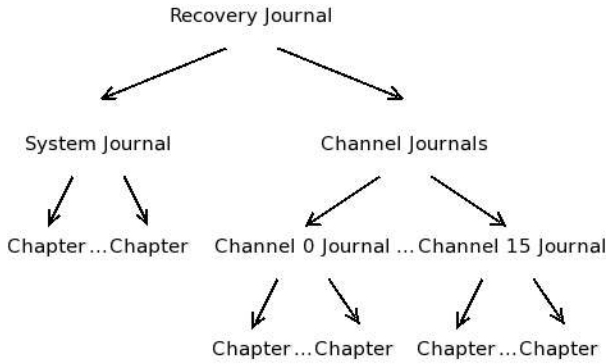

Figure 4. Structure du journal de réparation.

Voici la liste des chapitres du journal des commandes systèmes :

```
Chapter D : Simple System
Chapter V : Active Sense
Chapter Q : Sequencer State
Chapter X : System Exclusive
```

Et celle des chapitres des journaux de canaux :

```
Chapter P : Program Change
Chapter C : Control Change
Chapter M : Parameter System
Chapter W : Pitch Wheel
Chapter N : NoteOff, NoteOn
Chapter E : Node Command Extras
Chapter T : Channel Aftertouch
Chapter A : Poly Aftertouch
```

Il existe aussi, parmi les informations du journal de réparation, un mécanisme permettant de survoler rapidement ses informations et de savoir quels chapitres ont été modifiés par les commandes du dernier payload envoyé. Ce système accélère l'analyse du journal dans le cas simple d'une perte de paquet unique.

5. BIBLIOTHÈQUE RTP MIDI

Nous allons maintenant présenter nos premiers travaux d'implémentation de ce standard RTP MIDI. Ils concernent, pour l'instant, l'implémentation d'une bibliothèque permettant d'automatiser la création et l'analyse de payloads au format RTP MIDI. Cette bibliothèque sera publiée sous licence libre GNU Lesser General Public License [1]. Une première version devrait être disponible en juin 2005.

5.1. Généricité de RTP MIDI

Le format des payloads que l'on vient de décrire n'impose pas la manière dont un client (émetteur ou récepteur) doit les traiter. Celle-ci dépend des spécificités de l'application à réaliser ainsi que du réseau et des protocoles sous-jacents. RTP MIDI reste donc assez générique. Il définit cependant des outils permettant de décrire la session en utilisant le protocole SDP (Session Description Protocol) [4]. Il est ainsi possible de configuer plusieurs aspects de la session :

- le système de journal de réparation (utilisation ou non d'un journal, type de journal utilisé, règles d'inclusion des chapitres, etc.)
- la sémantique des dates
- la présence de plusieurs flux dans une même session et leur relations de dépendance
- etc.

5.2. Services de la bibliothèque

Au vue de la relative complexité de RTP MIDI et de sa nécessaire généricité, nous nous sommes proposés de construire un premier outil qui puisse servir à tout client voulant émettre ou recevoir des flux RTP MIDI. Notre bibliothèque automatise la création et l'analyse de payloads au format RTP MIDI.

Elle traite des parties “MIDI Command section” et “Journal section” des payloads mais ne s'occupe pas des entêtes RTP des paquets. Elle permet, dans le cas d'un client voulant émettre un flux RTP MIDI de créer, à partir de commandes MIDI datées, les payloads à inclure dans les paquets RTP. Dans le cas d'un client récepteur d'un flux RTP MIDI, elle permet d'extraire les commandes MIDI inclues dans les payloads et de créer, en cas de perte de paquets, les commandes de réparation permettant de réparer le flux (Figure 5).

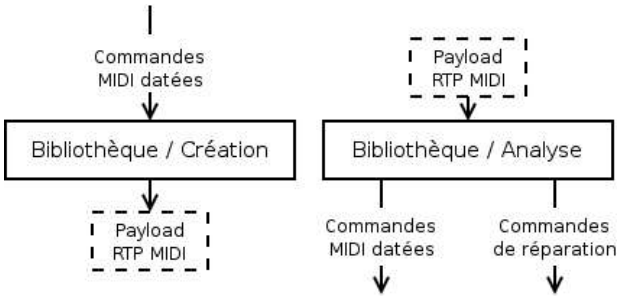

Figure 5. Services de la bibliothèque.

Le client reste responsable des interactions avec le réseau : envoi et réception des paquets via RTP, contrôle du flux via RTCP et description de la session via SDP.

Cette solution offre l'avantage de préserver la généricité de RTP MIDI tout en automatisant les aspects les plus complexes du standard (notamment l'écriture et la lecture du journal de réparation). La création et l'analyse des payloads peut se faire indépendamment des outils RTP utilisés et des algorithmes que le client souhaite utiliser pour l'envoi ou la réception des paquets, ceux ci dépendant fortement du type d'application et des compromis de performance souhaités.

Les aspects négatifs de cette solution sont qu'elle déporte beaucoup de complexité du côté du client : il

aura de grosses responsabilités de coordination entre les différents outils (RTP MIDI, RTP, RTCP et SDP) et devra également implémenter lui-même certaines parties normatives du standard comme la description de la session via SDP. Mais il existe par ailleurs plusieurs bibliothèques RTP pouvant simplifier ces tâches.

5.3. Les évènements MidiShare

MidiShare [3] est un système d'exploitation temps réel multitâches conçu pour le développement d'applications musicales.

Nous utiliserons ici, plus particulièrement, son gestionnaire d'évènements qui propose des méthodes à la fois pratiques et efficaces de stockage et de traitements de commandes MIDI, et ce sous forme d'évènements MidiShare. C'est en plus un gestionnaire dynamique de mémoire spécialement conçu pour les opérations temps réel.

Un évènement MidiShare peut représenter toute commande MIDI légale et sa structure permet également d'y associer certaines informations utiles : type de la commande, date, etc. (Figure 6).

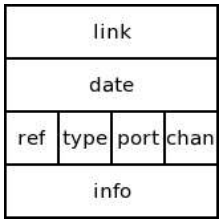

Figure 6. Structure d'un évènement MidiShare.

Afin de permettre une meilleure intégration de cette bibliothèque au système MidiShare et de bénéficier de ces représentations, les commandes échangées entre le client et la bibliothèque le sont sous la forme d'évènements MidiShare.

5.4. Interfaces

La bibliothèque est développée en C++ et deux classes servent d'interfaces uniques aux services de la bibliothèque : l'une pour la création de payloads (StreamWriter) et l'autre pour l'analyse de payloads (StreamReader).

Les informations de delta time associées à chaque commande de la section MIDI Command seront échangées avec le client par l'intermédiaire du champ `date` de l'évenement MidiShare.

5.4.1.StreamWriter

Voici les principaux éléments d'interface de la classe StreamWriter, utilisée par un client émetteur d'un flux pour l'écriture des payloads :

```
void
StreamWriter::newPayload (
  const uint32   payloadNumber,
  uint8 * buffer,
  const unsigned short size );
```

Cet appel débute la création d'un nouveau payload. Le payload sera créé dans le buffer `buffer` et ne dépassera pas la taille `size`. Le paramètre `payloadNumber` et servira à la mise-à-jour, en interne, de l'historique du flux ; il pourra être, simplement, le numéro du paquet RTP correspondant (*extended sequence number*).

```
bool
StreamWriter::putCommand (
  const MidiEvPtr command );
```

Cet appel rajoute l'évenement MidiShare `command` au payload en train d'être créé. La valeur de retour indique si la commande a bien été ajoutée au payload. Une valeur de retour `false` signale que le payload est plein et que la commande n'a pas été prise en compte.

```
int
StreamWriter::finalizePayload ( );
```

La fin de création d'un payload doit être demandée explicitement par cet appel (le client peut ainsi décider d'envoyer un paquet même s'il n'est pas complètement plein). Après cet appel seulement, le payload est prêt à être envoyé.

5.4.2.StreamReader

Voici les principaux éléments d'interface de la classe StreamReader, utilisée par un client récepteur d'un flux pour la lecture des payloads :

```
void
StreamReader::newPayload (
  const uint32 payloadNumber,
  uint8 * buffer,
  const unsigned short size );
```

Cet appel débute la lecture d'un nouveau payload. Le payload (sans l'entête RTP du paquet) devra se trouver à l'emplacement `buffer` et être long de `size` octets. Le paramètre `payloadNumber` sert également à la mise-à-jour de l'historique du flux.

```
MidiEvPtr
StreamReader::getRecoveryCommand (
  );
```

Après un appel à `newPayload`, le client peut demander des commandes de réparation pour le flux. Ces commandes seront créées par la bibliothèque grâce au système de journal de réparation. Cela ne fait de sens que si le payload suit immédiatement une perte de paquet, dans le cas contraire aucune commande de réparation ne sera créée.

Cet appel retourne une commande de réparation ou `NULL` si aucune commande de réparation n'est disponible. Les commandes de réparation ne sont pas datées.

```
MidiEvPtr
StreamReader::getCommand ( );
```

Cet appel retourne les commandes régulières du payload : celles qui sont inclues dans la partie “MIDI

Command section" du payload. L'appel renvoie `NULL` si aucune commande ne peut être extraite du payload.

5.4.3.Configuration du journal

Comme il a déjà été évoqué, le standard RTP MIDI prévoit que le type de journal de réparation et les règles d'inclusion des différents chapitres du journal puissent être spécifiés dans la description de la session. Notre bibliothèque doit donc offrir les moyens de personnaliser ces aspects lors de la création des payloads. Voici les éléments d'interface de la classe StreamWriter correspondants :

```
typedef enum { RECJ, NONE }
journallingMethod;

void
StreamWriter::setJournallingMethod
   ( journallingMethod j );
```

Cet appel, commun aux classes StreamWriter et StreamReader, permet de spécifier le type de journal de réparation utilisé pour le flux. La valeur par défaut du paramètre est `RECJ`, qui correspond au journal de réparation décrit dans le standard. Si la valeur `NONE` est spécifiée, aucun journal n'est utilisé.

```
typedef enum { UNUSED, NEVER,
DEFAULT, ANCHOR, ACTIVE }
chapterInclusion;

void
StreamWriter::setChapterInclusion (
   char chapterName,
   chapterInclusion c
   short channel = -1 );
```

Cet appel permet de modifier les règles d'inclusion des chapitres dans le journal de réparation par rapport aux comportements par défaut spécifiés dans le standard. Le paramètre `chapterName` détermine le nom du chapitre concerné. `UNUSED` interdit l'utilisation du type de commande concernant le chapitre. `NEVER` désactive l'utilisation du chapitre. `ANCHOR` supprime la purge du chapitre grâce au checkpoint. `ACTIVE` altère la sémantique de certaines commandes de reset (voir [6] Appendix C.1.3 pour plus de détails). Le paramètre `Channel` permet de spécifier quel est le canal dont le chapitre est concerné. Sa valeur par défaut permet d'appliquer la modification aux 16 canaux.

5.5. Modularité de l'architecture

Afin de supporter au mieux ces possibilités de modification des règles d'inclusion et des comportements du système de journal de réparation, l'architecture de la bibliothèque a été conçue de manière modulaire. Par exemple, la gestion de l'écriture et de la lecture de chacun des chapitres du journal de réparation est assurée par un composant autonome qui peut être ajouté ou enlevé simplement du journal aussi bien à la compilation qu'à l'execution. Il est également pensable de concevoir un système de journal alternatif et de l'intégrer facilement à la bibliothèque.

6. CONLUSION ET TRAVAUX FUTURS

Une grande partie de l'intérêt de RTP MIDI repose donc dans le concept du journal de réparation qui lui permet de pouvoir supporter des couches de transport peu fiables qui peuvent imposer des pertes de paquets. Aucun mécanisme similaire n'a encore été proposé, à notre connaissance, pour les standards plus récents permettant la transmission en temps réel d'évènements musicaux (ZIPI, OSC ou SASL).

Mais on peut également estimer que la solution proposée actuellement n'est pas complètement satisfaisante : la solution de journal de réparation proposé actuellement est complexe et l'on peut aussi craindre que sa taille pose rapidement des problèmes d'efficacité d'une part et de viabilité d'autre part, dans le cas de réseaux pouvant imposer une taille limite de paquet, et c'est notamment le cas d'UDP sur Ethernet.

Une fois l'implémentation de cette bibliothèque terminée. Nos travaux se porteront d'abord sur une évaluation du protocole et sur une étude de son optimalité dans différents cas d'utilisation. Nous chercherons ensuite à proposer une solution alternative au journal de réparation défini dans le standard.

Un autre axe de travail pourrait être le développement d'un deuxième composant proposant aux clients des services de plus haut niveau qui les soulageraient d'une part des responsabilités d'interactions avec RTP et SDP.

7. RÉFÉRENCES

[1] Free Software Foundation, *GNU Lesser General Public License*, 1999.

[2] Fober, D. "Real-Time Midi data flow on Ethernet and the software architecture of MidiShare", *Proceedings of the International Computer Music Conference*, Aarhus, Denmark, 1994.

[3] Fober, D., Letz, S. et Orlarey, Y. "MidiShare : une architecture logicielle pour la musique", *Informatique musicale : du signal au signe musical*, p.175-194, Hermes, 2004.

[4] Handley, M., Jacobson, V. et Perkins, C. *SDP : Session Description Protocol*, Internet-Draft, MMUSIC, IETF, 2005. (work in progress)

[5] International Standards Organisation. *ISO/IEC 14496 MPEG-4, Part 3 (Audio), Subpart (Structured Audio)*, 2001.

[6] Lazzaro, J. et Wawrzynek, J. *RTP Payload Format for MIDI*, Internet-Draft, AVT, IETF, 2004. (work in progress)

[7] Lazzaro, J. et Wawrzynek, J. *An Implementation Guide for RTP MIDI*, Internet-Draft, AVT, IETF, 2004. (work in progress)

[8] Lazzaro, J. et Wawrzynek, J. "A Case for Network Musical Performance", *11th International Workshop on Network and Operating Systems Support for Digital Audio*

and Video (NOSSDAV 2001), Port Jefferson, New York, USA, 2001.

[9] MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification*, Los Angeles, USA, 1996.

[10] McMillen, K., Wessel, D. et Wright, M. "The ZIPI Music Parameter Description Language", *Computer Music Journal*, Volume 18, Number 4, 1994.

[11] Postel, J. *User Datagram Protocol*, RFC 768, IETF, 1980.

[12] Postel, J. *Transmission Control Protocol*, RFC 793, IETF, 1981.

[13] Schulzrinne, H., Casner, S., Frederick, R., et Jacobson, V. *RTP : A Transport Protocol for Real-Time Applications*, RFC 3550, IETF, 2003.

[14] Wright, M. et Freed, A. "OpenSound Control : A New Protocol for Communicating with Sound Synthesizers", *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.