GESTION DES FLUX DE MESSAGES ENTRE APPLICATIONS GRÂCE À UN PLUGIN DE SERVEUR OSC DANS LE PROCESSUS COMPOSITIONNEL ÉLECTROACOUSTIQUE ET MIXTE

Matthieu JACQUOT
err0r500.music@gmail.com

RÉSUMÉ

Cet article se propose d'étudier l'utilité dans la méthode de travail du compositeur de musique électroacoustique ou mixte d'un serveur OSC intégré sous forme de plugin à une station audionumérique standard (DAW), lui permettant ainsi de gérer les flux de messages entre applications de manière synchronisée avec le matériau sonore grâce, en particulier, au système d'automation. Le but est de lui permettre ainsi de tirer parti des points forts de chaque type d'application : montage, édition, synchronisation depuis une DAW classique ; traitement du signal grâce à un environnement de synthèse sonore (ESS)[1].

1. INTRODUCTION

Bien que de nombreux ESS à la fois puissants et flexibles soient à la disposition des compositeurs électroacoustique (Max, PureData ou SuperCollider pour ne citer que les plus communs), leur manque fréquent d'outils pratiques pour la synchronisation avec un matériau sonore en cours d'élaboration peut rendre leur utilisation mal aisée pour le compositeur travaillant, au moins de manière préliminaire, sur média fixe[2]. Nombreux sont alors les compositeurs contraints d'adopter l'une des deux stratégies suivantes :

- effectuer les traitements sonores au sein de leur DAW grâce à des plugins standards puis les redéployer dans un environnement plus adapté au temps-réel (pour le cas de la musique mixte) en tentant d'approcher autant que possible la première version. Cela les empêche néanmoins de tirer pleinement profit de la puissance des ESS et augmente aussi considérablement le temps de développement puisqu'un même travail est à effectuer deux fois, dans deux environnements différents.
- travailler directement dans un ESS mais en étant alors obligé de réintégrer dans la DAW le matériau sonore traité. Ils sont aussi contraints de modifier en temps-réel les paramètres de traitement et d'enregistrer le rendu ce qui pose alors le problème de la reproductibilité du traitement. L'autre option pouvant être de procéder par séquences successives au cours de l'œuvre grâce à l'utilisation de « cues » successifs permettant de basculer d'une configuration à une autre. Bien que cette solution soit tout à fait justifiable sur un plan artistique et technique, il est néanmoins nécessaire que cela ne reste qu'une option et non une contrainte, ce qui reviendrait sinon à considérer que ces environnements limitent d'un coté tout autant qu'ils libèrent d'un autre.

Nous reviendrons tout d'abord sur les différentes solutions disponibles actuellement puis verrons en quoi l'utilisation d'un serveur OSC directement intégré à une DAW peut être une solution viable pour chacun de ces cas de figure.

2. PROBLÉMATIQUE ACTUELLE

2.1. Utilisation du MIDI

Historiquement, le protocole standard pour l'échange de messages entre applications fut le MIDI. Son évolution en « High-Definition Protocol » bien qu'en développement depuis 2005, n'est à l'heure actuelle (février 2013) pas totalement standardisé et ne peut donc pas être discuté ici [8]. Il existe néanmoins une alternative moderne : le protocole OSC développé par le CNMAT de l'Université de Californie à Berkeley. Ce protocole possède de nombreuses caractéristiques permettant de palier aux limitations historiques ou structurelles du protocole MIDI.

2.1.1. Limitations liées au protocole

Les quelques avantages de l'OSC par rapport au MIDI n'étant pas sujets à débat [4][7] et nous intéressant dans la situation actuelle peuvent être rapidement énumérés :

- Le schéma de nommage, l'adresse, sous forme d'URL permet d'avoir un paramètre « lisible par un hu-

[1] Il est à noter que ce plugin permet de synchroniser l'hôte avec quelque application capable de gérer la réception de messages OSC ce qui ouvre la possibilité de gérer aussi bien du son que des interfaces physiques (Monome), de l'image (Processing, Jitter, Gem) ou même des environnements 3D (Blender).

[2] L'intérêt porté au projet « rs.delos » de Roby Steinmetzer se proposant de recréer un séquenceur linéaire minimaliste directement dans Max semble corroborer cette assertion.[10]

main » (*human-readable*) et pouvant donner une idée assez claire de son rôle ainsi que d'éviter les conflits involontaires entre les paramètres comme cela est aisé en assignant le même numéro de Control Changes MIDI à deux paramètres différents. Voici par exemple la « taille de la pièce » d'une réverbération appliquée sur la première piste :

Ex : /track1/reverb/roomSize

- Avec OSC, les valeurs transmises ont une précision sur 32-bit comparé au MIDI qui est sur 7-bit et peuvent ainsi s'adapter beaucoup plus facilement au type de paramètre contrôlé. Cela permet par exemple d'exprimer directement en Hertz la fréquence de coupure d'un filtre. Cette caractéristique reste néanmoins tributaire de la résolution des automations dans l'hôte[3].

Ex : /track1/reverb/lowPassFreq 3128 *(Hz)*

- Nous voyons de plus, par ces deux simples exemples la possibilité d'utiliser des « motifs » dans l'adresse.[4] Le plugin de serveur OSC présenté tire bénéfice de cela en proposant un premier champ « main pattern » qui ici pourrait être /track1/reverb/ permettant ensuite de n'avoir qu'à ajouter la suite de l'adresse puis à spécifier si nécessaire les bornes des valeurs transmises. On peut alors envisager de multiplier les instances du plugin, chacune ayant un champ d'action bien défini.

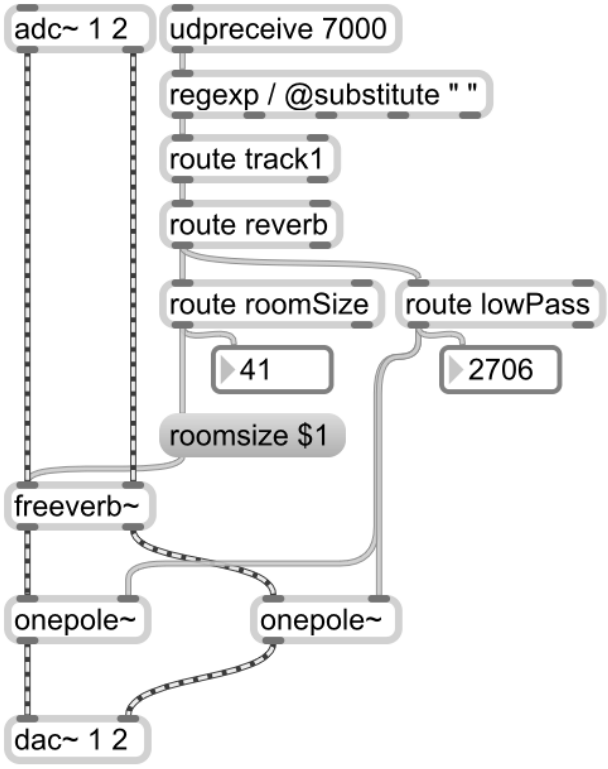

Figure 1. Contrôle d'une réverbération et d'un filtre passe-bas en utilisant le protocole OSC sous Max6.

- A cela il faut ajouter que l'OSC possède une précision temporelle de l'ordre de 10^{-12}s alors que le MIDI se situe aux alentours des 10^{-3}s.[4]

2.1.2. Limitations dues à l'implémentation logicielle

Il faut aussi noter que la prise en charge dans les DAW des messages MIDI (ici essentiellement des « Control Changes ») n'est pas pas toujours très adaptée à ce type d'utilisation car ils sont, pour des raisons historiques, plus souvent pensés comme une succession d'événements discrets plutôt que comme des flux et donc non dans l'optique d'une synchronisation avec le matériau sonore. De plus les logiciels font souvent la distinction entre piste audio ou MIDI ce qui signifie que la synchronisation doit s'effectuer entre deux pistes et non pas, comme c'est le cas pour les automations, comme des paramètres de la piste audio.

Il existe néanmoins des plugins permettant de transformer une enveloppe d'automation en message MIDI[4] mais cela n'affecte bien sûr pas les limitations liées au protocole lui-même et peut être la cause d'un nouveau problème.

En effet, router des messages MIDI en interne ne peut se faire que grâce à des logiciels conçus spécifiquement pour permettre l'interfaçage entre les applications au moyen de Bus virtuels. Néanmoins, la quantité de messages MIDI générée par les enveloppes peut rapidement se révéler problématique en conduisant à de faux-positifs dans la détection de boucles (*feedback loops*), certains de ces logiciels ont en effet une manière très sommaire de les détecter en se contentant de monitorer le volume de données transmises et en fixant un simple seuil, bloquant l'envoie de tous les messages dès que celui-ci est franchi.[5]

2.1.3. Avantages du MIDI

Un avantage indéniable du MIDI comparé à l'OSC est son taux de pénétration du marché. En effet, la prise en charge de l'OSC au niveau des interfaces et des logiciels est bien plus rare que le MIDI qui reste jusqu'à aujourd'hui le protocole de communication standard. Il est néanmoins nécessaire de remarquer qu'un nombre croissant de logiciels et d'interfaces le supportent.

2.2. Utilisation d'une solution « Max4Live »

Une alternative récente pourrait être la solution propriétaire Max4Live développé conjointement par Ableton et Cycling '74 et intégrant dans une même solution logicielle la flexibilité de l'ESS Max avec celui de la DAW généraliste Live, c'est-à-dire justement ce que nous tentons d'effectuer. Nous tenterons néanmoins de porter un regard critique sur cette solution qui, mis à part le fait que l'ESS soit évidemment limité à Max et que la DAW elle-même ne convient pas forcément à tous les types

[3] A titre d'exemple, la précision de l'automation dans le logiciel Reaper de Cockos va jusqu'à 6 décimales.

[4] Tel que ReaControlMIDI : http://www.reaper.fm/reaplugs/

[5] C'est le cas notamment de Loopbe1 développé par nerds.de

d'approches (par exemple, celles nécessitant un gros travail de montage de la source sonore traitée) :

2.2.1. La spécificité de la plate-forme contraint le compositeur à s'adapter à elle

L'avantage du serveur OSC est sa compatibilité avec n'importe quelle plate-forme supportant les plugins de type VST, AU ou RTAS c'est à dire quasiment toutes les DAW, y compris celle à laquelle le compositeur est déjà habitué et sur laquelle il a ses habitudes de travail. De plus il permet d'utiliser comme outil de traitement n'importe quelle application lui convenant, soit là encore par habitude, soit pour des raisons de spécificité du traitement à accomplir. Notons d'ailleurs aussi en faveur du serveur OSC qu'il permet d'utiliser plusieurs ESS simultanément. C'est donc ici l'environnement qui s'adapte au compositeur et non l'inverse.

2.2.2. Limitation de la possibilité de transmissibilité du traitement

Contrairement à l'approche électroacoustique dans laquelle le compositeur produit une œuvre « finie ». Le compositeur de musique mixte doit, au même titre que le compositeur de musique instrumentale, mettre en place un moyen pour des tiers de créer l'œuvre à sa place. Cela se fait sur deux plans : reproduire la partie instrumentale et le traitement sonore.

Si nous observons comment est gérée la partie instrumentale, nous nous rendons compte que les compositeurs et éditeurs ont naturellement opté pour le format le plus ouvert : un langage standardisé imprimé sur du papier ou diffusé via un format de fichier accessible à tous.

C'est pour la transmission du traitement sonore que le choix de l'ESS se révèle primordial et que l'ouverture rendue possible par VST2OSC prend toute son importance. En effet, un code source, qu'il soit sous forme de texte ou de schéma peut toujours faire sens pour celui qui le lit. A contrario, le format propriétaire adopté ici empêche légalement toute solution alternative d'avoir accès à ces informations, les fichiers de sessions étant par ailleurs sous forme binaire. Nous pouvons alors envisager deux conséquences :

- à court terme, car nous pouvons imaginer la situation à laquelle seraient confrontées les salles de concert en cas de prolifération de ce type de solution logicielle propriétaire du fait des coûts qu'elles engendrent.
- à long terme, car la maintenance d'un patch Max, PureData ou d'un code quel qu'il soit est déjà très problématique ; avec une solution fermée comme celle-ci, nous pouvons imaginer les problèmes auxquels se heurterait une tentative de reproduction d'ici plusieurs d'années. Pour s'en rendre compte il suffit tout simplement de lancer un regard sur les logiciels d'il y a 20 ans pour avoir un bon aperçu de l'ampleur de la tâche : pour pouvoir exploiter le « project » [6] il faudra retrouver une version du logiciel compatible avec celui-ci ce qui nécessitera sûrement l'emploi d'une version du système d'exploitation compatible ce qui implique alors aussi de l'installer sur une machine que ce dernier est capable de gérer.

Pour clore le parallèle avec la partition, nous nous retrouvons dans la même problématique que si les éditeurs exigeaient des interprètes de posséder le logiciel utilisé pour la gravure de la partition, dans une version compatible, pour pouvoir y avoir accès.

2.2.3. Nécessité de redéployer pour cette plate-forme spécifique des patchs Max existants

Bien que très proches, il est néanmoins nécessaire de redévelopper les patchs Max pour les passer sous forme de « device » Max4Live. Cela peut se révéler assez fastidieux dans le cas de gros patchs et ne présente comme unique avantage, par rapport à la solution proposée grâce au serveur OSC, la possibilité de faire des rendus « offline ».

2.3. ReWire

Les deux principaux problèmes liés à cette solution sont en premier lieu liés à la compatibilité. En effet, ReWire est une solution propriétaire de communication entre applications développé conjointement par Propellerhead et Steinberg. La licence restrictive de ce logiciel le rend de fait incompatible avec toute solution opensource ce qui signifie que, pour reprendre l'exemple des 3 ESS présentés ci-dessus, seul Max est compatible avec cette technologie. Les autres ne pourraient le devenir qu'en devenant propriétaires à leur tour. Ceci montre bien une grande réduction du panel d'applications à disposition du compositeur.

De plus la transmission des messages se fait en MIDI ce qui renvoie aux problèmes inhérents à celui-ci comme décrits ci-dessus (Cf 2.1).

2.4. OSCulator

Osculator est un logiciel propriétaire, disponible uniquement sous OS X. Sont but est, tout comme VST2OSC de permettre d'envoyer des messages OSC pour contrôler quelque logiciel susceptible de supporter le protocole. Un problème réside néanmoins dans le fait qu'il ne peut pas s'intégrer en tant que tel dans une DAW. Le seul moyen de le synchroniser avec un matériau sonore serait d'utiliser le MIDI pour faire l'interfaçage entre la DAW et OSCulator, ce qui nous renvoie aux mêmes problèmes que d'utiliser directement le protocole MIDI.

Il existe de nombreux autres logiciels permettant de contrôler des applications via des messages OSC mais tous sont pour l'instant pensés comme des moyens d'interfaçage geste-logiciel et ne permettent pas la reproductibilité du résultat.

[6] Nom donné aux sessions sous Ableton Live

3. SOLUTION PROPOSÉE

Le but est ici de trouver une solution de travail pour le compositeur qui soit à la fois intuitive, rapide, puissante et modulaire. Nous avons vu que jusqu'à présent les solutions étaient soit de consacrer beaucoup de temps par conviction au déploiement de solutions viables, soit d'accepter cet état de fait et de privilégier la rapidité au détriment de la pérennité.

3.1. Gestion des messages

3.1.1. Intégration

Le serveur OSC proposé, VST2OSC[6], se présente sous la forme d'un plugin[7], s'intègre alors parfaitement à n'importe quelle DAW et permet donc de mettre à profit les outils standards de cette dernière, en particulier les enveloppes, pour moduler des paramètres d'un patch de manière à ce qu'ils soient à la fois aisément enregistrables, éditables, reproductibles et synchronisés avec le son. On peut aussi envisager de faire transiter les messages d'un contrôleur dans la DAW où ils seront enregistrés en tant qu'enveloppes du plugin, ce dernier transmettant les messages à l'ESS. La performance pouvant ensuite être reproduite et éditée à volonté.

L'avantage de cette méthode étant à la fois de lier le travail compositionnel à l'écriture du patch de concert qu'il permet par ailleurs de rendre plus progressif et modulaire en se concentrant tout d'abord sur de petits modules de traitement du son au fur et à mesure que leur utilisation s'impose au compositeur et de ne se poser la question de leur regroupement au sein d'un grand patch pour le concert qu'une fois ceux-ci développés, rendant ainsi le portage extrêmement rapide et fiable puisque seule la partie prise en charge par la DAW reste à intégrer.

3.1.2. Détails de l'organisation du plugin

Figure 2. Interface graphique de VST2OSC

Bien que permettant de générer l'envoi des messages, l'interface graphique a été principalement conçue comme un éditeur permettant de formater les messages à envoyer. L'utilisation effective se faisant par le biais des enveloppes de la piste.

Le plugin est composé de deux grandes parties :

- la première prend en charge les paramètres globaux :

Champ	*Fonction*	*Valeur initiale*
Destination IP	Adresse IP du client	Ø (string)
Destination Port	Port sur lequel le client écoute	7000 (int)
Main Pattern	Partie de l'adresse commune à tous les modules	Ø (string)

Table 1. Paramètres globaux

- La seconde, contient 10 modules indépendants, chacun possédant les paramètres suivants :

Champ	*Fonction*	*Valeur initiale*
End Pattern	Fin de l'adresse	Ø (string)
min	Valeur minimale des données (data) à envoyer	0. (float)
max	Valeur maximale des données (data) à envoyer	127. (float)
steps	Pas de la valeur à transmettre	1. (float)

Table 2. Paramètres des modules

Au moment de l'envoi des messages, l'adresse est créée en concaténant les deux chaînes « Main Pattern » et « End Pattern ».

Les données envoyées, *f(x)* sont quant à elles un simple mappage des valeurs transmises par la DAW au VST, x étant de manière standard compris dans l'intervalle [0,1] :

$$f(x) = steps\left\lfloor \frac{min + x(max - min)}{steps} \right\rfloor \quad (1)$$

3.1.3. Fonctionnement interne

Le fonctionnement du plugin reste très basique.
Chaque milliseconde le plugin va vérifier pour chacun des 10 modules si la valeur actuelle a été modifiée depuis le dernier envoi. Si c'est le cas, il génère le message envoyé en UDP vers le port et l'adresse IP fournis dans les champs correspondants ; dans le cas contraire il passe au module suivant.

[7] Le développement du plugin a été fait en C++ en utilisant les bibliothèques JUCE et OSClib. Pour l'instant déployé uniquement sous forme de VST Windows, l'utilisation de JUCE permet néanmoins d'envisager aisément à l'avenir un déploiement cross-platform sous forme de VST mac, AU et RTAS.

3.2. Gestion des flux audio

Bien que cet article se concentre sur les envois de message, une application courante pour un compositeur ferait vraisemblablement aussi intervenir de l'audio. Nous allons donc très brièvement voir une solution possible.

Au niveau de l'échange de données audio en temps-réel entre applications, les serveurs sons JACK (open-source pour linux, windows, mac) ou Soundflower (propriétaire pour mac) de Cycling '74 permettent déjà un travail de ce type grâce au routing bidirectionnel entre applications (contrairement au ReWire qui est unidirectionnel : du client ReWire vers l'hôte ReWire).[9]

Pour prendre l'exemple de JACK qui est la solution la plus ouverte et polyvalente : le nombre de connexions dont dispose chaque application peut être fixée arbitrairement et sont alors vues comme autant d'entrées/sorties standards pour chaque logiciel. Elles peuvent de plus être rappelées et connectées entre elles automatiquement grâce au système de baies de brassage (patchbays).[3]

Suivant les DAW il devient alors envisageable d'utiliser l'ESS comme un simple effet en insert appliqué directement dans chaque piste à traiter, ce qui évite par ailleurs d'avoir à utiliser 2 pistes : l'une pour l'envoi à l'ESS, l'autre pour monitorer ce qu'il renvoie. La seule contrainte étant de pouvoir router le signal vers une sortie vue comme physique par l'application et le récupérer de la même manière.

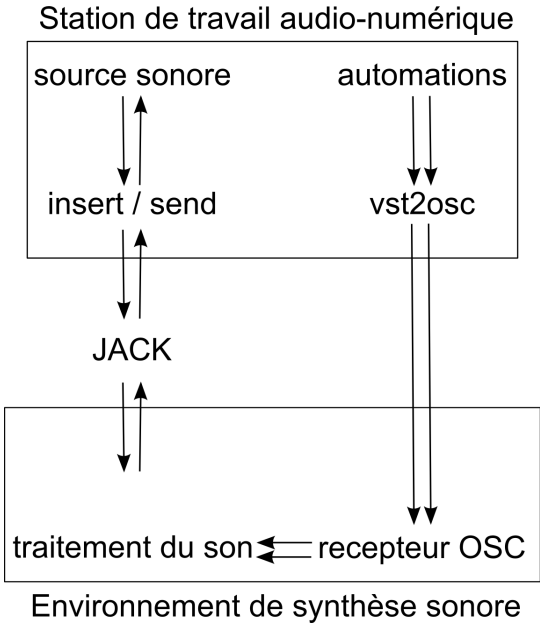

Figure 3. Les flux de données (audio + messages) de la solution proposée

4. EXEMPLES D'UTILISATION

4.1. Production

Pour imager l'utilité de ce système, il suffit de revenir à la genèse du plugin. En tant que compositeur l'auteur s'est trouvé face à un problème pour lequel le MIDI semblait être une solution peu pratique.

Après avoir codé un patch max permettant de générer la distorsion complexe d'une source sonore, il restait à trouver un moyen d'accéder aux 55 paramètres automatisables qu'il comptait.

L'utilisation de l'OSC semblait alors idéale puisqu'elle permettait grâce à un schéma de nommage logique de permettre une mémorisation rapide ainsi que d'éviter l'écueil potentiel d'un conflit dans les numéros de Control Change, surtout en cas de réutilisation ultérieure du patch.

4.2. Prototypage

L'auteur a récemment créé une installation multimédia interactive contrôlée par le public via une interface web distante[5]. Les modifications effectuées sur le site internet sont récupérées par l'ordinateur de l'installation puis transmises à un patch Max via un serveur OSC codé en Python.

Bien que VST2OSC ne soit pas utilisé lors des performances il permit un gain de temps considérable en court-circuitant cet interfaçage complexe et en permettant de tester l'aspect procédural, le traitement du son ainsi que la partie visuelle directement dans la DAW qui ne contenait que les extraits sonores nécessaires et 5 enveloppes d'automatisation correspondant aux 5 contrôleurs disponibles dans l'interface web.

5. CONCLUSION

5.1. Limitations

5.1.1. Interceptions des frappes par l'hôte

De part sa fonction et son interface assez peu conventionnelles pour un VST, certains hôtes peuvent rendre problématique l'entrée de texte dans le plugin. Certains l'interceptent purement et simplement, d'autres ne donnent pas le « focus » total au VST et certaines combinaisons de frappes entrent alors en conflit avec des raccourcis claviers, déclenchant les actions correspondantes dans l'hôte.

Néanmoins, et même si la liste ne peut bien sûr pas être exhaustive, des solutions de contournement par le biais d'options disponibles dans les paramètres (Reaper) ou de fichiers de configuration à modifier (Live) ont pour l'instant toujours été trouvées pour permettre un parfait fonctionnement du VST.[6]

5.1.2. Latence

Contrairement à un traitement du son opéré directement au sein de l'hôte, le fait de devoir échanger des données audio et des messages induit une latence. Pour mettre en valeur ce problème nous allons réaliser un simple test en situation, reprenant le schéma de la figure 2 : un bruit blanc est généré par l'hôte puis routé vers et depuis Max grâce au plugin ReaInsert[8].

Chaque 1000ms, VST2OSC envoie un message multipliant le signal dans Max par 0. (-∞ dB) puis par 1. (0 dB) après 500ms. Le son résultant est enregistré et fait ainsi apparaître la latence. Les résultats (Fig. 3) sont donnés avec « latence audio compensée »[9] La latence moyenne mesurée est d'approximativement 3ms, à noter aussi que ces résultats sont obtenus via une interface réseau « loopback » et sont donc susceptibles d'êtres supérieurs sur un réseau local.

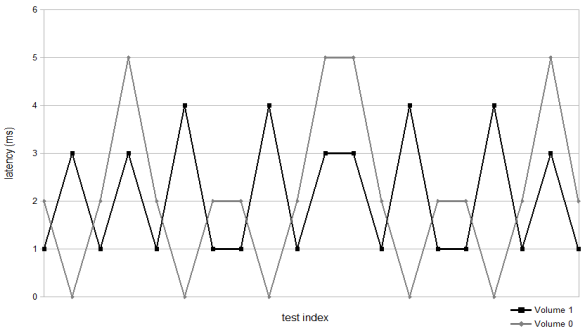

Figure 4. Test de latence

Il reste donc problématique d'utiliser ce type de configuration lors d'opérations nécessitant une grande précision temporelle mais cela semble néanmoins suffisant pour la plupart des utilisations musicales.

5.2. Améliorations

Dans le but de pouvoir multiplier les instances du plugin sans impacter la charge processeur, il est à envisager que seuls les modules pour lesquels un pattern a été défini ne soient vérifiés.

La taille du buffer dans lequel est stocké le message avant l'envoi est actuellement de 1024bit. Il pourrait être intéressant de laisser sa taille à la discrétion de l'utilisateur en cas par exemple de détection de perte de paquets. En effet, les messages sont transmis en UDP ; ceci permet par rapport au TCP de réduire la latence entre le serveur et le client mais la légèreté du protocole se fait au détriment de la fiabilité, rendant possible la perte d'information, la congestion du réseau et ne garanti pas l'ordre d'arrivée des paquets (deux paquets émis simultanément arriveront dans un ordre non prévisible)[2] ce qui peut se révéler très problématique. Le protocole RUDP[1] pourrait être une solution intermédiaire, s'il venait un jour à être standardisé.

[8]ReaInsert : http://wiki.cockos.com/wiki/index.php/ReaInsert

[9]Le plug-in ReaInsert permet de détecter et de compenser le délai dû à l'aller/retour entre applications.

6. RÉFÉRENCES

[1] Bova, T. Krivoruchka, T. "Reliable UDP Protocole", ietf.org, http://tools.ietf.org/html/draft-ietf-sigtran-reliable-udp-00, 1999.

[2] Chuah, C.-N. "Transport Layer: UDP vs. TCP", UC, Davis, USA, http://www.ece.ucdavis.edu/~chuah/classes/eec173A/eec189q-f04/lectures/L13_transport.pdf, 2003.

[3] Davis, P. "The JACK Audio Connection Kit", LAD Conference, Karlsruhe, RFA, http://lac.linuxaudio.org/2003/zkm/slides/paul_davis-jack/why.html, 2003.

[4] Freed, A. Schmeder, A. Zbyszynski, M. "Open Sound Control - A flexible protocol for sensor networking", CNMAT, Berkley, USA, http://opensoundcontrol.org/files/OSC-Demo.pdf, 2011.

[5] Jacquot, M. "--Manipulation--", peter1island.com, http://peter1island.com/art/manipulation/, 2013.

[6] Jacquot, M. "vst2osc", Paris, France, http://peter1island.com/technicalStuff/audio-plugin/vst2osc-rtas2osc.php, 2013.

[7] midi.org "MIDI Manufacturers Investigate HD Protocol", midi.org, http://www.midi.org/aboutus/news/hd.php, 2013.

[8] midi.org "White Paper: Comparison of MIDI and OSC", midi.org, http://www.midi.org/aboutmidi/midi-osc.php, 2008.

[9] propellerheads.com "ReWire - Technical information", propellerheads.com, http://www.propellerheads.se/developer/index.cfm?fuseaction=get_article&article=rewiretechinfo, 2013.

[10] Steinmetzer, R. "rs.delos, sort of a crossbreed of the pre-Max5 timeline object", arts.lu, http://arts.lu/roby/index.php/site/maxmsp/rs_delos, 2013.

NB : Chaque lien a été vérifié le 07 avril 2013.