Ferramentas livres para distribuição de áudio em rede

Flávio Luiz Schiavoni[1], **André Jucovsky Bianchi**[1], **Marcelo Queiroz**[1]

[1]Instituto de Matemática e Estatísticas – Universidade de São Paulo (USP)
Rua do Matão, 1010 - Cidade Universitária - São Paulo - SP - Brasil - CEP 05508-090

{fls, ajb, mqz}@ime.usp.br

Abstract. *The increasing interest in colective/collaborative musical works led to the development of several tools that simplify network connections for real time music distribution. On the other hand, lack of knowledge of these tools and their requirements, besides lack of proper documentation, are barriers for actual collaborative network music development. This article presents some of these tools, aiming to help interested users in overcoming some of these barriers, and making it easier for them to explore network music distribution and effectively applying it in artistic context.*

Resumo. *O interesse crescente em trabalhos musicais coletivos/colaborativos levou ao desenvolvimento de diversas ferramentas que simplificam a criação de conexões em rede para distribuição musical em tempo real. Por outro lado, o desconhecimento da existência ou dos requisitos de tais ferramentas, além da falta de documentação adequada, são barreiras para o desenvolvimento de fato de música colaborativa usando redes. Este artigo apresenta algumas destas ferramentas, com o intuito de ajudar usuários interessados a transpôr algumas destas barreiras, tornando mais fácil a exploração da distribuição de música em rede e a aplicação efetiva desta em contexto artístico.*

1. Introdução

A interdisciplinaridade das áreas de computação musical e música ubíqua e a conscientização dos problemas e demandas que surgem em cada fronteira interdisciplinar sugerem que esforços coletivos sejam despendidos para a criação de ferramentas que apoiem a exploração e a criação artísticas nos mais diversos contextos. No caso da performance musical distribuída (usando redes de computadores), este esforço tem gerado ferramentas que tanto auxiliam o músico a gerenciar conexões de áudio em tempo real entre localidades distantes, como também servem de pano de fundo para explorações e reformulações conceituais e estéticas. Várias composições distribuídas e apresentações musicais em rede têm sido realizadas nas últimas duas décadas [Tanaka 1999, Young and Fujinaga 1999, Young 2001, Chafe et al. 2000], porém as dificuldades técnicas têm se mostrado recorrentes neste tipo de cenário.

No contexto da pesquisa e desenvolvimento de uma nova ferramenta para distribuição de sinais de áudio em rede, é fundamental conhecer os trabalhos já existentes nesta direção. Restrições ou especificações quanto ao escopo de aplicação, tecnologias utilizadas, licenças de publicação ou de uso do código, entre outras características, são importantes tanto para caracterizar as ferramentas já existentes, quanto para decidir se o desenvolvimento deve ou não se apoiar nestes trabalhos.

Dentre as restrições acima mencionadas, a licença de publicação de uma ferramenta talvez esteja entre os aspectos mais importantes do ponto de vista do desenvolvimento de software. Uma grande dificuldade no contexto de desenvolvimento de FLOSS (Free/Libre Open Source Software) é a dependência, ainda que indireta, de ferramentas ou bibliotecas fechadas, pois desta forma não há garantia permanente da disponibilidade de tais recursos frente a necessidades de adaptação ou atualizações de sistemas operacionais e outras bibliotecas utilizadas. Sem o código disponível, mesmo uma ferramenta ou biblioteca gratuita poderia se tornar inutilizável poucos meses após o abandono do desenvolvimento por parte de seus autores.

Nas próximas seções, apresentaremos uma série de ferramentas (publicadas sob licenças livres) de transmissão de sinais de áudio e controle em rede, suas características e possibilidades que oferecem à prática musical distribuída, tanto na performance quanto na criação colaborativa.

2. Ferramentas livres para música em rede

Um amplo levantamento bibliográfico e correspondente busca por ferramentas para música em rede permitiu a compilação de uma lista extensa de aplicativos que se propõem a resolver o problema de transmissão em tempo real de conteúdo sonoro e musical através de redes de computadores. Para os propósitos deste trabalho, é fundamental que as ferramentas selecionadas tenham seu código fonte publicado sob licenças livres, pois assim são passíveis de análises e investigações mais detalhadas, além de permitir a introdução de novas funcionalidades e posterior republicação do código-fonte, para alcançar outros objetivos além dos originalmente propostos pela ferramenta.

As ferramentas selecionadas para esta análise são: Jacktrip, Netjack, SoundJack, Ninjam, jack.udp, llcon e os *externals*[1] do Pure Data `streamin~` e `streamout~`. Esta lista acima representa uma seleção abrangente de ferramentas que satisfazem os critérios aqui estabelecidos: estarem publicadas sob licenças livres e permitirem a transmissão de áudio em tempo real.

Tal lista não é de forma alguma exaustiva, pois há uma série de outras ferramentas para música em rede, algumas com código fechado, outras com objetivos diferentes dos considerados aqui, como por exemplo ferramentas para transmissão assíncrona ou para transmissão exclusiva de dados de controle. Apresentamos a seguir uma pequena lista de aplicativos que se relacionam com a distribuição de conteúdo musical em redes, mas que não se encaixam nos critérios deste texto:

- O protocolo de comunicação OSC [Wright 2005] tem sido muito utilizado para a criação de novos instrumentos e distribuição de sinais de controle para aplicações musicais conectadas por rede [Schmeder et al. 2010], sem contudo possuir como objetivo a distribuição de áudio [Wright 2005].
- O software OtherSide[2] [Anagnostopoulos 2009] permite a síntese remota de sinais em um servidor web controlado por meio de mensagens OSC, enviadas através de uma interface CGI:IRC para navegadores web. Como no exemplo acima, apenas sinais de controle (e não de áudio) são transmitidos por rede.

[1] Objetos Pure Data escritos em C.
[2] http://otherside.gnufunk.org/

- O software Quintet.net [Hajdu 2003] é uma ferramenta feita em MAX/MSP que permite utilizar som e vídeo distribuído pela Internet. Por ser dependente do ambiente fechado MAX/MSP, ela não pode ser estendida no contexto de FLOSS, pois suas dependências (neste caso, o próprio ambiente MAX/MSP) não podem ser distribuídas livremente junto com outras extensões.
- O LDAS (Low Delay Audio Streamer) [Sæbø and Svensson 2006] era uma ferramenta para Linux que usava o protocolo UDP para o transporte de áudio capturado via ALSA. Foi lançado em 2005 e apresentado no Linux Audio Conference de 2006, e possuía vários objetivos em comum com o presente trabalho. Lamentavelmente o código-fonte não está mais disponível, sendo que a página do projeto está atualmente fora do ar.

Além destas, há ainda outras ferramentas que se propõem a resolver o problema de transmissão de áudio em tempo real por rede, mas que não possuem código aberto para uma análise detalhada de seu funcionamento ou extensão dentro do paradigma FLOSS. Entre estas ferramentas, é possível citar o eJamming[3] e o Rewire[4].

Nas próximas seções descreveremos com certo nível de detalhamento as ferramentas que se conformam aos objetivos deste trabalho.

2.1. JackTrip

O Jacktrip[5] foi desenvolvido por Chris Chafe e por Juan-Pablo Cáceres no CCRMA da Universidade de Stanford, Califórnia, em 2008 [Chafe 2011]. Esta ferramenta foi utilizada em diversas apresentações do grupo de pesquisa SoundWIRE (Sound Waves on the Internet from Real-time Echoes) e combina diversas técnicas computacionais para lidar com o problema de latência e perda de pacotes [Cáceres and Chafe 2009b]. O Jacktrip é integrado ao servidor de som Jack [Cáceres and Chafe 2009a].

Desenvolvido em C++ e utilizando a biblioteca Qt [Cáceres and Chafe 2009a], o Jacktrip possibilita a utilização de um sistema de redundância de pacotes para garantir a integridade no recebimento dos dados. Cada mensagem UDP possui um número sequencial e um *buffer* circular é utilizado para ordenar os pacotes recebidos. Como no protocolo UDP os segmentos de dados podem ser entregues mais de uma vez, o mecanismo de alimentação do buffer circular trata da triagem dos segmentos que ainda não foram recebidos e daqueles duplicados e que podem ser descartados. Além disso, o uso de um buffer permite amortizar as variações no atraso da rede (conhecidas como *jitter*), garantindo que a entrega dos dados mantenha uma taxa constante de atualização, às custas de um pequeno aumento na latência[6].

O Jacktrip utiliza como padrão uma taxa de amostragem do sinal de áudio igual a 44.100 Hz com amostras de tamanho 16 bits. Para realizar compactação (com perdas) e redução da largura de banda, necessárias para o tráfego de vários canais em rede, o codec CELT pode ser utilizado.

[3] http://ejamming.com/

[4] http://www.propellerheads.se/products/reason/index.cfm

[5] http://code.google.com/p/jacktrip/

[6] A transmissão de dados sem nenhum buffer é extremamente arriscada, pois neste caso todo pacote perdido se converteria em ruído

O modelo conceitual de comunicação implementado no Jacktrip é uma arquitetura do tipo cliente-servidor, porém restrito a apenas um cliente por servidor, tornando-o na prática um sistema *peer-to-peer*. Quando o servidor é instanciado e o cliente se conecta, uma mesma quantidade de canais é criada para envio e recebimento de sinais, permitindo assim a comunicação full duplex.

2.2. NetJack

O NetJack[7] foi desenvolvido por Torben Hohn, Dan Mills e Robert Jonsson. É atualmente a ferramenta de distribuição de sinais áudio e controle que funciona integrada ao servidor de áudio Jack [Letz et al. 2009]. A arquitetura desta ferramenta utiliza um modelo mestre/escravo no qual os nós mestres utilizam o backend de áudio do Jack e um cliente interno de rede para trocar conteúdo musical com os escravos, que utilizam o backend de rede do Jack [Carôt et al. 2009].

Neste modelo, para cada par de nós que queira estabelecer conexão para trocar sinais de áudio é necessário que um deles seja mestre e o outro escravo. Um nó que roda a aplicação como mestre pode se conectar a vários escravos, mas cada instância da aplicação configurada como nó escravo só se conecta a um único mestre. Veja que nada impede que um mesmo nó faça por um lado o papel de mestre de um certo conjunto de nós e por outro o papel de escravo de um ou mais mestres (abrindo várias instâncias escravas). Um detalhe interessante é que os nós escravos são totalmente configurados pelo nó mestre (taxa de amostragem, tamanho de bloco, etc).

O Netjack utiliza o protocolo UDP e adiciona uma identificação temporal em cada pacote para reordená-los no buffer circular do receptor. Esta identificação é utilizada para sincronizar as placas de som como um relógio mundial distribuído, onde as máquinas escravas devem converter sua taxa de amostragem (Sample Rate Conversion – SRC), para compensar o *clock drift*[8] entre escravos diferentes.

O codec CELT pode ser utilizado para codificação de áudio. Além da transmissão de áudio, o NetJack também permite o envio de MIDI e o controle remoto da funcionalidade Transport do Jack, responsável pela sincronização de processos remotos (por exemplo, o início da reprodução em uma máquina concomitante com o início da gravação em outra máquina).

O NetJack possui ainda três modos de operação: rápido, normal e lento. Estes modos diferem quanto à latência extra adicionada pelo sistema para lidar com diferentes condições de tráfego na rede. No modo rápido, para redes rápidas e pouca transmissão de dados, o mestre envia um bloco de amostras para o escravo, que deve recebê-las, produzir seu próprio conteúdo e responder para o mestre ainda no mesmo ciclo. No modo normal o mestre espera o conteúdo do escravo no ciclo seguinte, e no modo lento espera-se o resultado do escravo com 2 ciclos DSP de diferença.

[7] http://netjack.sourceforge.net/

[8] Duas interfaces de áudio podem apresentar desvios na medição do tempo devido ao fato de seus pulsos serem providos por cristais que vibram aproximadamente (mas não exatamente) na mesma frequência.

2.3. Ninjam

O Ninjam [9] está sendo desenvolvido desde 2005 pela Cockos Incorporated em um projeto liderado por Justin Frankel, um dos criadores do programa de compartilhamento de arquivos Gnutella e do reprodutor de mídia Winamp. Ao contrário dos demais programas que tentam evitar latência, o Ninjam adiciona propositalmente um compasso de latência[10] por meio de um buffer, de forma que os músicos sempre escutam o sinal um compasso atrás do que estão tocando – e portanto precisam tocar sempre um compasso à frente do que estão escutando. Com o uso de uma latência tão alta a ferramenta consegue evitar o jitter da rede, introduzindo no entanto um descompasso na relação natural entre escutar e reagir, por parte dos participantes da performance. Segundo os desenvolvedores, o Ninjam foi desenvolvido com ênfase em experimentações e expressões musicais onde este descompasso possa ser visto como parte da proposta estética e não como limitação técnica.

O programa utiliza uma arquitetura cliente-servidor, na qual é necessário manter um servidor ativo para que os clientes possam se conectar. Para diminuir largura de banda necessária para o tráfego de áudio na rede, o Ninjam utiliza o codec OGG Vorbis. O tráfego de dados é feito por meio do protocolo TCP (https://github.com/wahjam/wahjam/wiki/Ninjam-Protocol), que garante ausência de perdas de pacotes através de um sistema de mensagens de confirmação de recebimento. Finalmente, o Ninjam possui distribuição para Windows (por meio da API de multimídia nativa), Linux (por meio do ALSA) e MacOSX (por meio do Core Audio).

2.4. SoundJack

O SoundJack[11] foi desenvolvido por Alexander Carôt no ISNM da Universidade de Lübeck/Alemanha, e fornece uma maneira bastante direta de conectar a entrada da placa de som à interface de rede. Isto permite transformar um computador em um servidor de *streams* UDP para trabalhar com performances em rede [Carôt et al. 2006b, Carôt et al. 2006a].

O SoundJack foi otimizado para os sistemas Windows, Linux e OSX e pode utilizar os codecs CELT ou ULD para diminuir a quantidade de dados transferidos [Carôt and Werner 2008]. Segundo o autor, estas opções foram incluídas pois a intenção desta ferramenta é funcionar sobre uma conexão ADSL comum. O desenvolvimento deste software foi feito utilizando a biblioteca Qt em C++, e inclui o desenvolvimento de uma interface gráfica.

O Soundjack utiliza um sistema de reamostragem para ajustar diferenças na taxa de amostragem entre máquinas diferentes (world-clock drift). Para isto, em todos os pacotes de áudio enviados é enviado também um timestamp. Ao receber um pacote de áudio, a ferramenta verifica o instante em que o mesmo teria sido enviado (pela taxa de amostragem e tamanho de bloco), calculando a diferença que o pacote

[9] http://www.ninjam.com

[10] No Ninjam a configuração da latência do sistema usa a noção de *compasso* em um sistema métrico-musical, mas na prática essa configuração se refere a uma medida de tempo fixada previamente, que os músicos/usuários usarão como referência para a sincronização.

[11] http://www.soundjack.eu/

tem em relação ao seu tempo lógico local, para então garantir a sincronização dos blocos [Carôt and Werner 2009].

O SoundJack também possui uma espécie de *ping* de áudio: um pacote ICMP que contém dados de áudio e que pode ser respondido por qualquer máquina que possua habilitada a resposta para o comando PING. Assim, é possível medir tempo de latência utilizando o protocolo ping tradicional, mas com dados reais de áudio, sem que nenhuma configuração adicional seja adicionada à máquina destino [Carôt et al. 2008].

2.5. Jack.udp

O *Jack.udp*[12] foi desenvolvido por Rohan Drape em 2003 [Sæbø and Svensson 2006], e corresponde a um mecanismo de transporte UDP para o servidor de som Jack, sendo distribuído através de um pacote chamado "jack tools" nas distribuições Linux Debian e Ubuntu. Esta aplicação pode ser executada em dois modos: modo de envio e modo de recebimento. No modo de envio a aplicação lê pacotes de áudio de portas de entrada do Jack e envia pacotes UDP para uma determinada porta/IP na frequência dos ciclos DSP determinada pelo Jack local. No modo de recebimento a aplicação lê pacotes recebidos pela rede na porta indicada e escreve os dados recebidos em um buffer associado a um porta de saída do Jack. Além dos dados de conexão de rede como IP e porta, a ferramenta permite configurar o tamanho do buffer circular entre a rede e o Jack, sendo o padrão 4096 amostras Além disto, o padrão da ferramenta é utilizar a porta 57120, o endereço 127.0.0.1 e dois canais (sendo que a ferramenta oferece suporte para um máximo de 32 canais).

2.6. llcon

O *llcon*[13] (Low-Latency (internet) Connection tool), depois renomeado para *Internet Jam Session software*, está sendo desenvolvido por Volker Fischer desde 2004. Este software foi feito para utilizar conexões de redes limitadas como, por exemplo, conexões ADSL com 256kbps [Anagnostopoulos 2009] podendo ser utilizado para conectar vários usuários a um servidor central. A ferramenta trabalha com uma topologia em estrela, sendo sempre necessária a presença de um servidor central para garantir a conexão entre clientes. O servidor central irá mixar todos os sinais recebidos e devolverá aos clientes os sinais mixados. Caso um cliente queira enviar vários canais de áudio ou um canal estéreo, os mesmos serão mixados localmente de modo que apenas um canal seja enviado ao servidor central.

O software trabalha com uma frequência de amostragem de 24kHz com o codec de áudio CELT (o codec anterior IMA-ADPCM foi abandonado a partir da versão 3 da ferramenta pois acarretava um atraso maior por causa da codificação). Assim, independentemente da codificação utilizada nos clientes, os dados serão sempre codificados dessa maneira ao trafegar pela rede, e decodificados localmente para uso de outras aplicações.

O llcon funciona em Windows por meio de um driver ASIO, em MacOSX com Core Audio e em Linux usando o servidor Jack.

[12] http://www.slavepianos.org/rd/sw/sw-23/

[13] https://sourceforge.net/apps/mediawiki/llcon/index.php

2.7. streamin~/streamout~

Desenvolvido em 1999 por Guenter Geiger, estes *externals* permitem enviar e receber fluxos (*streams*) de áudio dentro do Pure Data. Desenvolvidos para troca de áudio em redes locais, os mesmos não possuem nenhum tipo de compactação ou correções para perdas de pacotes ou para sincronização entre os pares. O único parâmetro configurável que permite diminuir a largura de banda necessária para o tráfego de dados é a resolução das amostras, que pode utilizar 8 bits, 16 bits ou 32 bits. Para resolver diferenças entre os pares, informações sobre o número de canais e resolução são enviadas juntamente com as amostras de áudio.

Distribuídos com a versão Vanilla do Pure Data, estes externals encontram-se disponíveis em versões do Pure Data para todos os sistemas operacionais.

3. Comparação das ferramentas relacionadas

A comparação entre as ferramentas apresentadas neste artigo não é uma tarefa simples pois elas não foram desenvolvidas para um mesmo propósito. É possível imaginarmos diferentes cenários de aplicação onde cada uma das ferramentas poderia ser considerada ideal e melhor que as outras. A Tabela 1 traz alguns parâmetros importantes de comparação que podem servir para identificar tais cenários, e que serão discutidos a seguir.

Tabela 1. Resumo tabular das ferramentas apresentadas

	API	CODEC	Interface	Rede	Arquitetura
Jacktrip	Jack	CELT	TUI	WAN	Ponto a ponto
Netjack	Jack	CELT	TUI	WAN	Estrela
Ninjam	Várias	OGG Vorbis	GUI/TUI	ADSL	Estrela
SoundJack	Jack	CELT/ULD	GUI	ADSL	Ponto a ponto
jack.udp	Jack	Nenhum	UI	LAN	Ponto a ponto
llcon	Várias	CELT	GUI	ADSL	Estrela
Pd stream	PD	Nenhum	PD	LAN	Ponto a ponto

A API de áudio utilizada influi na maneira como o usuário interage com a ferramenta. Usuários acostumados a trabalhar com o servidor de som Jack conseguem adicionar facilmente as ferramentas de música em rede às suas ferramentas de áudio habituais. Já para usuários que não estão acostumados a usar o Jack, ferramentas de rede que trabalham diretamente sobre a API do sistema operacional podem ser mais fáceis de interagir. Já o caso do external para Pure Data permite que a distribuição seja feita dentro deste ambiente de criação e performance, o que flexibiliza muito o uso da distribuição de áudio por parte dos usuários desta ferramenta.

A utilização de um codec de áudio permite diminuir o volume de dados a serem enviados pela rede. No caso de uma rede como a Internet, onde o roteamento normalmente não é conhecido a priori, a diminuição do fluxo de dados pode ajudar a garantir a entrega de todos os pacotes. Por outro lado, qualquer codec de áudio que utilize compactação com perdas altera a qualidade do áudio. Assim, ferramentas que permitem ao usuário utilizar ou não um codec de áudio tornam-se mais flexíveis para diferentes cenários.

A interface do usuário (UI=User Interface) de uma ferramenta também pode ser um fator decisivo em relação à sua simplicidade de uso. Interfaces em modo texto (TUI) são geralmente mais flexíveis para automação de processos, porém podem ser menos intuitivas por não se apoiarem em paradigmas espaciais para disposição de informações e controles, como é o caso das interfaces gráficas (GUI).

O tipo de rede para o qual uma ferramenta foi desenhada, seja ela rede local (LAN), redes de longa distância (WAN) ou conexões domésticas (ADSL), influi na maneira como a ferramenta irá funcionar e no tipo de cenário de aplicação ideal. É importante ressaltar que o fato de uma ferramenta ter sido desenvolvida para um tipo de conexão não impede sua utilização em outro contexto. A diferença principal é o fato de o desenvolvimento ter sido guiado pelos problemas e soluções que cada tipo de conexão possui, dotando a ferramenta de funcionalidades para resolver ou contornar estes problemas específicos.

Por último, a arquitetura de uma ferramenta pode influenciar na simplicidade de sua utilização. Conexões ponto a ponto implicam em conectar apenas dois computadores na distribuição do áudio, o que pode ser suficiente para performance distribuída em rede envolvendo duas localidades, onde cada ponto de captação em um local é transmitido diretamente para o outro local, sem atrasos adicionais por conta de servidores centrais. Já no cenário de um ensaio de um grupo de n músicos, seriam necessárias $n(n-1)$ conexões unidirecionais ponto a ponto para que todos se escutem, enquanto o mesmo resultado seria produzido por n conexões em uma topologia estrela.

4. Conclusão

Este artigo teve como objetivo apresentar e comparar ferramentas disponíveis para música em rede. Foi dada maior ênfase para ferramentas livres que permitem distribuição de áudio, que além de poderem ser utilizadas diretamente, permitem também o desenho e implementação de extensões de suas funcionalidades atuais, seja para flexibilizar seu uso e aumentar o número de cenários de aplicação, ou para melhorar sua interface com o usuário.

A popularização da tecnologia necessária para o compartilhamento de fluxos de áudio em rede é um dos primeiros passos na direção de permitir uma exploração mais sistemática das redes de computadores como verdadeiros espaços de interesse acústico e estético, com características específicas que as tornam intrinsicamente distintas de espaços acústicos usuais, por exemplo na medida em que latência e distância não são necessariamente conceitualmente traduzíveis ou quantitativamente redutíveis um em relação ao outro.

Além disto, a popularização desta mesma tecnologia é também essencial para o redirecionamento dos esforços de desenvolvimento de software para um contexto de aplicação verdadeiramente ubíquo, onde não apenas as necessidades do músico profissional ou do performer experimental estariam em jogo, mas também as possibilidades de aplicação do compartilhamento de áudio via rede em tempo real a atividades mais gerais, de interesse artístico, didático, lúdico ou uma combinação destes, através de dispositivos de uso cotidiano, por exemplo dotados de conexões Wi-Fi ou bluetooth.

5. Agradecimento

Esta pesquisa é realizada com o apoio do CNPq (proc. nº 141730/2010-2) e da FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo (proc. nº 2008/08632-8).

Referências

Anagnostopoulos, I. (2009). The otherside web-based collaborative multimedia system. In LAC, editor, *Proceedings of Linux Audio Conference 2009*, pages 131–137.

Cáceres, J.-P. and Chafe, C. (2009a). Jacktrip: Under the hood of an engine for network audio. In *Proceedings of International Computer Music Conference*, page 509–512, San Francisco, California: International Computer Music Association.

Cáceres, J.-P. and Chafe, C. (2009b). Jacktrip/Soundwire meets server farm. In *In Proceedings of the SMC 2009 - 6th Sound and Music Computing Conference*, pages 95–98, Porto, Portugal.

Carôt, A., Hohn, T., and Werner, C. (2009). Netjack–remote music collaboration with electronic sequencers on the internet. In *In Proceedings of the Linux Audio Conference*, page 118, Parma, Italy.

Carôt, A., Kramer, U., and Schuller, G. (2006a). Network music performance (NMP) in narrow band networks. In *Proceedings of the 120th AES Convention*, Paris, France.

Carôt, A., Renaud, A., and Verbrugghe, B. (2006b). Network music performance (nmp) with soundjack. In *In Proceedings of NIME 2006 Network Performance Workshop*.

Carôt, A., Renaud, A. B., and Werner, C. (2008). Audible icmp echo responses for monitoring ultra low delayed audio streams. In *AES 124th Convention*, Amsterdam, The Netherlands.

Carôt, A. and Werner, C. (2008). Distributed network music workshop with soundjack. In *In proceedings of the Tonmeistertagung*, Leipzig, Germany.

Carôt, A. and Werner, C. (2009). External latency-optimized soundcard synchronization for applications in wide-area networks. In *AES 14th Regional Convention*, Tokio, Japan.

Chafe, C. (2011). Living with net lag. In *Proceedings of AES 43RD INTERNATIONAL CONFERENCE*, Pohang, Korea.

Chafe, C., Wilson, S., Leistikow, A., Chisholm, D., and Scavone, G. (2000). A simplified approach to high quality music and sound over IP. In *In Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00*, pages 159–164.

Hajdu, G. (2003). Quintet.net - a quintet on the internet. In *Proceedings of International Computer Music Conference*, Singapore.

Letz, S., Arnaudov, N., and Moret, R. (2009). What's new in JACK2? In LAC, editor, *Proceedings of Linux Audio Conference*, page 1.

Sæbø, A. and Svensson, U. P. (2006). A low-latency full-duplex audio over IP streamer. In *Proceedings of the Linux Audio Conference*, pages 25–31, Karlsruhe, Germany.

Schmeder, A., Freed, A., and Wessel, D. (2010). Best practices for open sound control. In *Linux Audio Conference*, page 131, Utrecht, NL.

Tanaka, A. (1999). Network audio performance and installation. In *Proceedings of International Computer Music Conference*, pages 519–522, San Francisco, California: International Computer Music Association.

Wright, M. (2005). Open Sound Control: an enabling technology for musical networking. *Org. Sound*, 10:193–200.

Young, J. and Fujinaga, I. (1999). Piano master classes via the internet. In *In Proceedings of International Computer Music Conference*, pages 519–522, Beijing, China.

Young, J. P. (2001). Using the Web for live interactive music. In *Proceedings of International Computer Music Conference*, pages 302–305, Habana, Cuba.