HELSINKI UNIVERSITY OF TECHNOLOGY
Telecommunications Software and Multimedia Laboratory
20.12.2006
T-111.5080 Seminar on content creation
Fall 2006: Interactive Digital Theatre

Latency Issues in Distributed Musical Performance

Jari Kleimola

30742A

Latency Issues in Distributed Musical Performance

Jari Kleimola
HUT, Telecommunications Software and Multimedia Laboratory

Jari.Kleimola@tml.hut.fi

Abstract

Distributed performance occurs when a group of artists and audiences in geographically separated locations collaborate with each other in virtual performance spaces. A pre-requisite for such an event is that the interaction happens in near real time. However, even the highest speed network links have latencies which result lack of presence. This paper investigates the components, causes and tolerance of latency and jitter, and their impact to distributed musical performance. Latency reduction and compensation possibilities are discussed, and audio delay plugin with ping modulated delay time parameter is described, and two experiments using various parameters are conducted. State of the art survey of distributed musical performance software is presented before conclusion.

1 INTRODUCTION

Distributed performance can be defined as a collective form of art that occurs simultaneously in multiple locations, using networked interaction between physically dispersed participants (Green *et al.*, 2006). Performers, hardware equipment and computer software constitute a virtual performance space, which can be experienced not only by on-site audience(s), but also by global communities via especially crafted internet websites. Audiences can be passive, or actively collaborating in the virtual performance space, making world wide joint events possible in near real time.

As the amount of expression that can be achieved by a distributed performance is remarkable, many experiments and works employing distribution have been conducted in forms of sonic arts, theatre, dance and cinema. In distributed cinema, a large canvas is divided into tiles which can be used to display content from distributed sources. For example, *SPECFLIC 1.0* performed at the UCSD campus (Jenik *et al.*, 2005) used on location 'Sousveillance Grid' to display video originating from live cameras, mobile phones and webcams, which was also published via a web page. *DANCEPOD 2006* connected six simultaneous dance parties from USA and Europe via web, sharing a virtual performance space by projecting visual streams from each participant club into large screens beside dance floors, and playing audio streams mixed by DJ's of individual clubs (PICA 2006). In distributed theatre, a play is dramatized so that actors interact with each other using real time video and audio streams which are shown to local audiences in each participating space. As an example, *Distributed Improv* organized in

2003, a group of theatrical improvisers perfomed at UCLA and Stanford using realtime audio and video streams to connect performers and audiences (CyberSImps, 2003). Distributed musical performance strives to produce a collective audio stream, which is mixed together from networked sources consisting of individual players or groups of musicians. *Point25* concert connected a quartet and audiences from Sweden and USA into a shared jam session (Point25, 2004).

There are also numerous research projects around the world addressing issues in distributed performance. To name a few, CCRMA's SoundWIRE Group at Stanford have investigated sonification of network's QoS parameters, high quality audio streaming between remote locations, and effects of time delay in ensemble accuracy (SoundWIRE 2001). Lazzaro and Wavrzynek (2001) have studied network musical performance using MIDI and RTP, and have conducted practical experiments on the subject. Distributed Immersive Performance group at USC have concentrated so far on duo performances (DIP 2006). A comprehensive list of research activities can be found in (Grether 2003) and (Barbosa, 2006).

The focus of this paper is in distributed musical performance, and in particular in more or less strictly tempo related material. Thus, temporally stochastic or purely event-driven sonic artwork is scoped out, and emphasis is in more traditional forms of music. The aim of the study is to gather background information of a scenario in which multiple home networks are connected via internet into an adhoc virtual music making community, related problems of such an environment, and to provide current state of the art software overview.

First, a generic architecture of a distributed performance system is presented, followed by an example system suited for musical purposes. Problem areas are pointed out, and some theoretical calculation results are presented in order to get the magnitude of the latency issues. Next, components, tolerance and compensation of latency are discussed. A test experiment to subjectively validate latency issues is prepared and conducted, followed by a survey into state of the art software available for distributed musical performance in domestic environment. Finally, conclusion is given in chapter 6.

2 ARCHITECTURE OF A DISTRIBUTED PERFORMANCE SYSTEM

A distributed performance system comprise performers and audiences in physically separated locations, and supporting technical equipment in each location consisting of cameras, microphones, screens, loudspeakers, control desks and computer hardware and software. Locations or performance nodes are interconnected using a data network, which can be a LAN, or most usually the internet. Figure 1 shows an example system with three locations and a remote audience access point. One of the nodes may function as the session management master.

The interconnecting network between nodes should be able to transmit audio, video, control, and communication related data in near real time. Each of these streams may contain multichannel data, depending on installation. As a result, the bandwidth requirements of such a system are high, so fast fixed links between nodes are preferred.

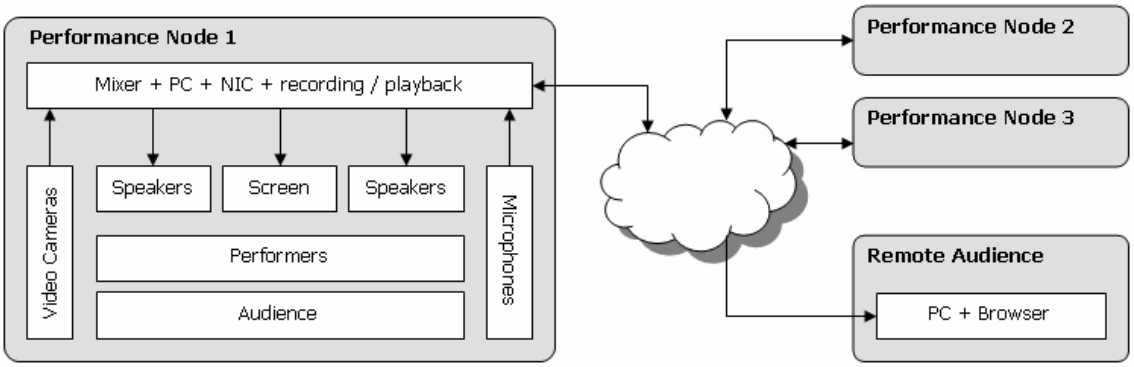

Figure 1: Generic distributed performance system. Three geographically separated nodes interconnected by a network.

On the other end of the scale, a simplified semi-synchronous version intended for distributed musical performance can be realized with domestic equipment and basic ISP accounts, as shown in figure 2.

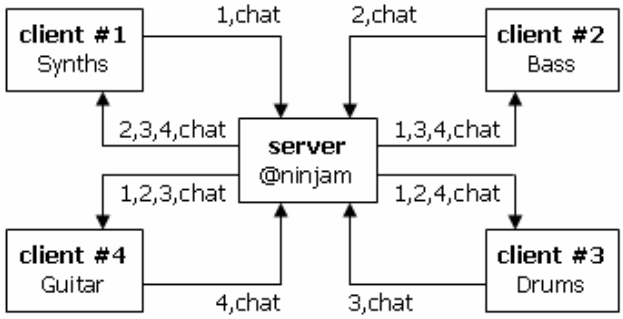

Figure 2: NINJAM architecture (Underwood, 2006). Four clients and a centralized server.

Regardless of the complexity of the setup, the problems inherent in distributed networked performance are related to a reduced sense of presence. This is akin to synchronization between performers (or lack of it), and is subject to the quality of service of the underlying infrastructure. Communication delays and asymmetric connections make responsive interaction between performers more difficult. Packet loss resulting from a trade off between a) near real time quality and b) transport reliability, cause interrupts in streams that are sensed as clicks or pops in audio, or freezed frames in video material. This might also be a consequence of jitter.

3 TIMING IN DISTRIBUTED MUSICAL PERFORMANCE

3.1 Real World Scenarios

In ideal conditions, when a signal travels at the speed of light and ignoring all networking delays, it takes nearly 67 milliseconds for a signal to reach a point located at the opposite side of the globe. At the speed of sound, this translates to 22.7 metres. On the other hand, the distance between a first violin player and a double bass player in a full-sized symphony orchestra (both sitting at the fifth row when counted from the conductor's point), is about 30 metres. A delay between the conductor and a trumpet

player (sitting at the back row of the orchestra) is in the order of 46 milliseconds. Temporal threshold of human hearing is according to psychoacoustic research around 25-35 ms (Haas, 1949).

In musical timescale, a 16th note at the tempo of 120 beats per minute (bpm) lasts 62.5 ms. Minimum time resolution of MIDI is 640 µs, considering 10 bits per byte, 2 bytes per note event (using running status), and 31.25 kbps communication speed.

3.2 Components of Latency

Latency in distributed musical performance consists of local and network latencies. Local latency is introduced by instrument's performance interface, air and audio signal paths from sound source to the computer / from computer to speakers, and by computer's audio interface coupled with codec's (de)compression algorithm delays. Network latency is defined as the time that is consumed after data is ready to be transmitted in sender's application layer before it is available at the same application layer at the receiving end (Delaney *et al*., 2006).

General guidelines for instrument interface design have dictated that performance latency should not exceed 10 ms, although this has been later argued by Mäki-Patola (2005) to be relaxed, and allow even 60 ms for continuous sounding instruments. These values might be thought as extremes, because initial attack characteristics of various instrument families do differ substantially. The path from source to the computer can be ignored when electric instruments are used, but when performing with acoustic instruments, each meter between source and microphone increases latency by approximately 3 ms. Modern audio interfaces with state of the art device drivers achieve latency figures that are below 5 ms. If audio synthesizers are used as the source, the complexity of the patch may need to be taken into account also, because depending on synthesis method, a single voice with effects processing may take 60-360 µs processing time (Kleimola, 2005), and chords may require six or even more voices. MIDI event routing from input port to the sound generating algorithm takes 50 µs per event on the average. For six note chords the synthesis overhead is of the same magnitude as if acoustic material was inputted through AD converter instead. According to Underwood (2006), perceptual codec latencies are at least 20 ms.

Network latency consists of i) *transmission delay*, which is equal to packet size divided by communication speed, ii) *propagation delay* which is dependent on physical distance of the nodes and is about 5 µs per km (assuming 70% speed of light signal propagation speed), and iii) *internetworking device delays* caused by store/forward buffering, queuing and packet processing actions.

Because of routing and queuing, the network latency component is usually not constant over time. When individual packets have different latencies, jitter occurs. For streaming audio, it is important that packets arrive in time so that they can be played without gaps in packet stream. Buffering at the receiving end can be introduced to guarantee that packets are available in time.

3.3 Latency Tolerance

As already noted in 3.1, the precedence effect says that sounds are perceived as coming from a single source when they occur within 25-35 ms time window (Haas, 1949). Later research has suggested that we can subconsciously perceive latencies below 10 ms, but in musical instrument performance context latency tolerance can be even higher than 50 ms. Chew *et al.* (2004) have studied duo performances, and in four-hand piano experiment they have found that delays below 50 ms were generally considered tolerable, and that when local delay was introduced, the tolerance increased to 65 ms.

They also concluded that latency tolerance is dependant on tempo (slower tempo equals higher tolerance), and on instrumentation. For example, if a synthesizer pad type sound is used in ensemble performance, the musician has to anticipate chord changes so that slow attack is matched with the song tempo (corresponding latencies of several hundred milliseconds), whereas transient sounds result lower latency tolerance levels. Musical genre affects also tolerance, because it often presets tempo and instrumentation choices, and it is clear that a loop-based tightly quantized machine groove is less tolerant than a rubato style ballad. Individual playing styles do have an impact as well.

3.4 Analysis

Local Delays (one-way)	ms	m
instrument performance interface	10	3.4
air + audio signal path from source to PC	3	1.0
PC audio interface / synthesis	5	1.7
compression (codec)	20	6.8
decompression (codec)	20	6.8
PC audio interface	5	1.7
air + audio signal path to speakers	3	1.0
total	**66**	**22.4**

Table 1. Local delays. Second column shows the delay in milliseconds, and the third column corresponding distance at the speed of sound.

Table 1 summarizes local delays for one-way trip, including both sender and receiver nodes. As can be seen, most time is spent in audio (de)compression actions (40 ms), leaving 16 ms to acoustic delays that would be present even in traditional musical performance, and 10 ms to AD/DA conversions and sound synthesis algorithms. 10 ms equals 3.4 m over the air distance, which is not uncommon in live staging, while 40 ms ≅ 13.4 m requires some kind of visual interaction for player synchronization.

Table 2 shows calculation results for transmission delays and variable networking delays for links between Espoo and three university servers located in Stockholm, New York and Los Angeles. Transmission delays were calculated by assuming that a 128 kbps CBR audio stream is to be transmitted in 20 ms slices. Then payload for single slice is 20*128/8 = 320 bytes, and total packet size with IP,UDP and RTP headers becomes 320+20+8+16 = 364 bytes. For bit rates of 512 and 1024 kbps the transmission delay is therefore 0.7 and 0.35 ms, respectively. For internetworking device delays, 125 µs per hop packet processing delay was assumed, and packet size of 364 bytes with 1024 kbps bit rate was used for store/forward delay calculations.

Transmission Delays	ms	m
@ 512 kbps	0.7	0.2
@ 1024 kbps	0.35	0.1
Variable Networking Delays	ms	m
propagation (one way, 5 us per km)		
Stockholm (660 km)	3	1.0
New York (7600 km)	38	12.9
Los Angeles (8900 km)	45	15.3
internetworking devices (one way)		
Stockholm (www.kth.se, 14 hops)	7	2.3
New York (www.columbia.edu, 23 hops)	11	3.7
Los Angeles (www.ucla.edu, 30 hops)	14	4.9
total (one way)		
Stockholm	**11**	3.7
New York	**50**	17.0
Los Angeles	**60**	20.4
Ping (one way)	ms	m
Stockholm (www.kth.se)	11	3.7
New York (www.columbia.edu)	69	23.5
Los Angeles (www.ucla.edu)	103	35.0

Table 2. Static and destination dependent one-way networking delays, and pinged values (measured from Espoo, Finland).

Congestion delay is missing from calculations, so there is no latency in figures caused by queuing (which for loads of 50, 75 and 90% can multiply internetworking device delays by factors of 2, 4 and 10). Furthermore, because exact network topology was not known (and because it is dynamic), propagation delays were calculated using a straight line approximation for end node distances. It was also impossible to determine the transmission media between internetworking nodes and the latency caused by interconnecting equipment circuits. These error sources make calculations purely theoretical, and they should therefore be considered as minimum latencies between nodes. However, it is possible to see that if the local upstream link speed is 1024 kbps instead of 512, total latency is decreased only by 0.35 ms.

Popular utility for network performance measurement is *ping*, which uses ICMP echo requests and responses to calculate round trip times between two nodes. Bottom part of table 2 gives pinged values (halved for one-way link) for the three servers, and these are generally larger than the theoretical values calculated above. The reason for difference is that ping includes queuing delays and that it travels through real network topology path. However, pinged results should be considered as maximum latencies, because internetworking devices usually give ICMP low priority (meaning that queuing might take longer than with actual data packets when network load is high).

Total latencies for the three cases presented above become 87, 136 and 146 ms. When this is compared to the acceptable range stated in (Chew *et al.* 2004), it seems that the delays are too high for enjoyable musical performance. Latency tolerance depends on a number of factors however, and when the figures are compared to those presented in topic 3.1, we find that 136 ms corresponds roughly to a 8^{th} note at tempo of 120 bpm, and that 87 ms corresponds to the distance of two players sitting at the opposite ends of a symphony orchestra. Furthermore, if codec delays are eliminated from total delay values, we get 47, 96 and 106 ms, assuming that all other conditions stay intact.

3.5 Latency Reduction and Compensation

There is always latency in networked musical performance, but the overall performance experience can be enhanced by optimizing the technical environment and by masking of the unidealities with acoustic makeup. At local side, instruments may be fine tuned so that their responsiveness feels comfortable in the performance environment. Low latency computer audio interfaces with decent device drivers are mandatory, and new processor architectures can be used to support efficient streaming algorithms. Codec delays can be eliminated entirely by streaming raw audio, but this has a negative effect on the amount of packets or packet sizes that must be transmitted.

On transmission phase, careful selection of streaming protocol is important, and UDP-based packets should be favored over the reliable TCP transmission methods (which provide re-transmission of lost packets even if they are not going to be delivered in time, and has built-in data rate management that scales poorly for real time scenarios). Jitter can be decreased by buffering, but ideal buffer size cannot be determined by jitter elimination alone, because bigger buffers mean longer latencies. Number of samples per packet affect also internetworking device buffering delay, because routers and gateways start forwarding packets only after they have been fully received. By reducing the audio quality the amount of transmitted data can be decreased. An obvious way to reduce latency is to use appropriate Service Level Agreement to get guaranteed bounds for latency and jitter.

If all peers share identical sound engines, it might be feasible to increase the semantic level of transmitted data, and pass only performance related data over the network using MIDI or SASL (Structured Audio Score Language of MPEG-4), and leave rendering to the client synthesizers.

Latency tolerance may be increased by introducing delays into locally generated streams so that each performer hears his own lines as they are presented to other participants. Because of network asymmetry, this can only be achieved in special cases, and should be adjustable locally according to performer preferences. Adding reverb, low pass filtering, amplitude scaling or other DSP effects to the audio streams can make latency artifacts sound smoother, but their use depends on musical genre.

4 PING DELAY

4.1 Description

To subjectively test the effects of network latency to playing, digital audio delay effect that uses *ping* to dynamically control the delay time parameter was developed. It runs inside a host audio sequencer employing VST plugins, and consists of a simple circular buffer delay line and a delay time modulation component that constantly pings a given internet address, and changes the delay line length accordingly. The architecture of the plugin is shown in figure 3 below. A screenshot of the plugin is displayed in figure 4b.

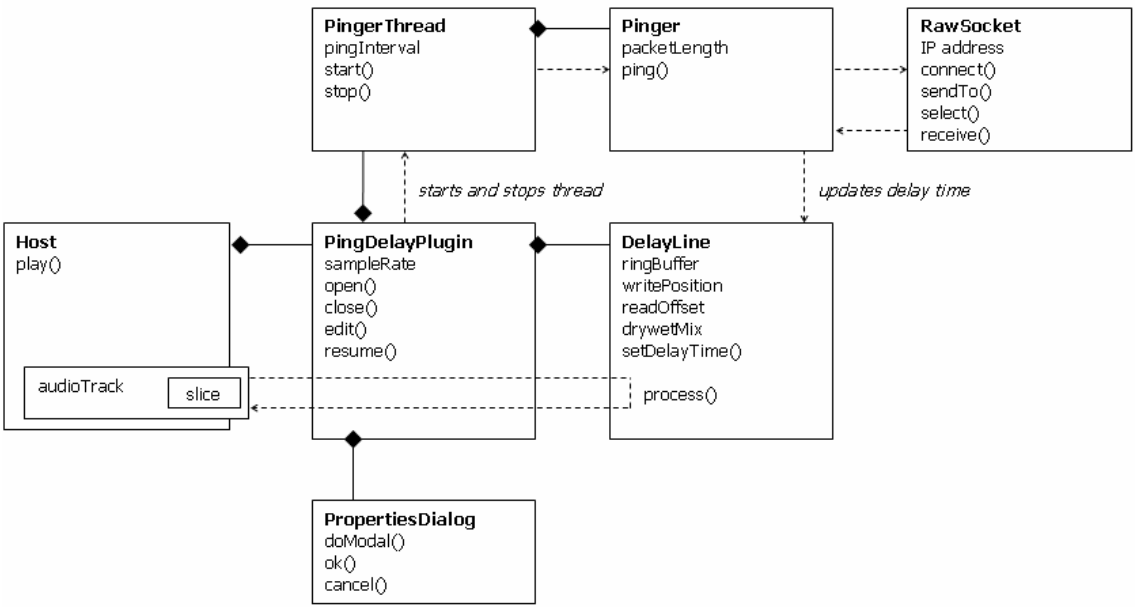

Figure 3: Ping audio delay plugin architecture.

The test setup is shown in figure 4a. Audio or MIDI streams are loaded into sequencer tracks, and an instance of ping delay is inserted into each stream's signal path before signal enters a mixer channel. Internet address or fixed delay time, jitter range and rate, dry-wet mix ratio (to set volume of local vs. delayed stream) and bypass settings can be adjusted for each instance separately. One of the sequencer's tracks is fed from a software synthesizer that is controlled live from a local MIDI keyboard, and is also fed through a ping delay plugin instance so that various (fixed) delay times can be tried out.

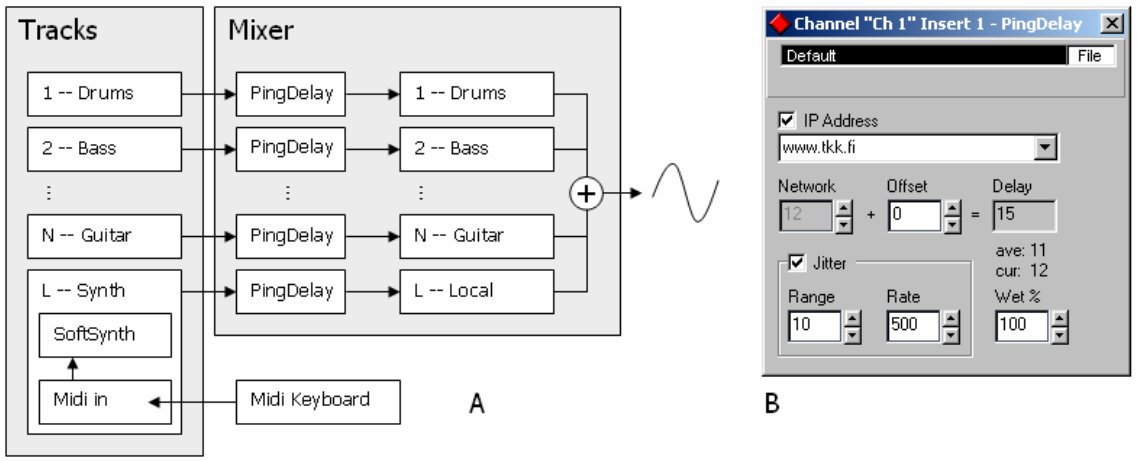

Figure 4: a) Test Setup. b) Ping Delay front panel.

4.2 Tests

Two experiments were conducted. The first one was instrumented as a quintet (drums, bass, clavinet, electric organ and Rhodes) augmented with a local brass part. Three different four bar funk grooves were looped at tempi of 80, 120 and 160 bpm, and ping delays were addressed to locations that provided average one-way latencies of 0, 11, 69 and 103 milliseconds. Local live part was adjusted to use various fixed delay times. Ping

destinations were changed so that different delays could be tested with different instruments of the quintet. All parts were initially recorded without quantization, and the shortest note duration was a 16^{th} note.

In second experiment, a pre-recorded vocal part was imported into sequencer's audio track, and random jitter values were applied in the plugin, so that the vocal part had a constantly fluctuating tempo. This was done in order to simulate vocalist's actions to synchronize herself into the delayed accompaniment coming from another performance node. Local piano part was played as an accompaniment, first dry, then enhanced with big room reverb, and finally layered with a string pad sound. The genre was a rock ballad, with average tempo of 64 bpm.

4.3 Results

4.3.1 Experiment #1

When 50% dry-wet mix ratio was used, 11 ms delay sounded like a chorus effect, and was generally acceptable for all instrument timbres, possibly excluding the drums. 69 ms was sensed as a (slapback) echo, and delay of 103 ms was considered annoying. For 100% wet signal (ie. only delayed part audible), the group play did not suffer from 11 ms delay, but 69 ms was considered unacceptable in particular between drum and bass parts. This is not surprising, because the genre requires particular tight interaction with drummer and bass player, and delays over 50 ms was felt too sloppy in 120 bpm tempo (16th note length is 62.5 ms). Keyboard tracks suffered the least from latencies. Large latencies were felt as if band members did not pay any attention to each other. As expected, slower tempos adjusted better to delays.

When local brass part was played live over the groove, it was felt most natural to synchronize to the drum part. If all other parts were bypassed from the latency effects, and the drummer was late, few tens of milliseconds local delay towards drum delay made live playing sound more in time (although adding local delay would suggest quite the opposite). After few minutes of rehearsal it was relatively easy to anticipate the delay by playing ahead of the beat, and the end result was like the sound was programmed to have a longer attack time.

4.3.2 Experiment #2

In this experiment, no pinging nor fixed delays were used. The vocal part was not originally sung over quantized accompaniment, so there was some amount of tempo fluctuation already present. A little practice was needed to get into the original feel of the song. After that, 1000 ms was set for the jitter rate (little more than length of one beat at 64 bpm), and 10 ms was set for the jitter max range to make the vocal part tempo to fluctuate between 61 and 67 bpm. Because jitter delay value was drawn randomly, vocal track had to be constantly monitored, and this was unpleasant. Reverb and string pad layer did not have a direct impact on jitter feel compensation, but made the overall playing experience more enjoyable.

4.3.3 Discussion

Adding 86 ms offset (the sum of local delay calculated in 3.4 and sound packet buffering delay) to the delay values was felt uncomfortable, suggesting that codec latencies should be eliminated from the delay budget, and that 10 ms sound packets might be preferred over 20 ms buffering. Experiment 1 showed the importance of conductor track, as most natural synchronization source was the drum track. This is more or less different to all participants, so further study should be organized to investigate available synchronization solutions. The benefit of experiment 2 is questionable, because in real performance situation tempo is not changed randomly, but according to mutual artistic expression between soloist and accompanist.

5 SOFTWARE FOR DMP

Despite numerous research projects addressing distributed musical performance using realtime audio streams, at the time of writing, end user level synchronous applications are available only in MIDI domain. Following the pioneering ResRocket Surfer software (which is no longer in existence), *eJamming* is a standalone general MIDI compatible distributed studio environment for Windows and Macintosh computers that enables realtime jamming over internet (eJamming, 2006). Local MIDI streams are delayed so that they can be synchronized into remote events, and late notes can be rejected from live performance (although they are recorded for later in-sync playback). User can configure the amount of delay and the threshold for late note rejection. Sessions can be public or private, and they are managed via a community site. Full-duplex VoIP is used for in-session communication.

Audio streams are only supported in asynchronous (or semi-synchronous) mode for the time being. *VSTunnel* is a vst/au plugin that connects multiple audio sequencers together over internet using an insert effect slot on sequencer's master out channel (Miller, 2005). The plugin detects changes in local audio, compresses the signal and then transmits the timestamped packets into other sequencers. Received data is uncompressed and inserted into a sample accurate location inside target sequencer (which can be in loop playing mode when certain section of a song is being worked on). As the output of each performer is transferred separately, streams can later be mixed at will. User is able to define when his audio is transmitted over to other participants, as well as the quality of the audio stream. Sessions can be public or private, and public sessions can be pre-auditioned before joining in (sessions can be discovered via community forum pages). Textual chat channel is used for communication between performers during session.

NINJAM is standalone software for Windows, Mac and Linux, which can stream local audio tracks to other participants, and receive and mix streams created by others (Underwood, 2006). Interchanged streams are quantized in measure boundaries, so local musician plays against other collaborators' previous take, and as soon as local quantized interval is exceeded, the stream is packed in client machine into OGG Vorbis format and sent to the NINJAM server, which then redistributes it to other clients in session. Like in VSTunnel, each collaborative stream is transferred separately, and can thus be mixed in each client. Modest 512 kbps dowstream is required for typical eight person jam session, with 100 kbps upstream. Manufacturer hosted servers allow free

anonymous login into sessions, and may require 3 Mbps outbound bandwidth for a similar eight person session.

Digital Musician Net is an online recording community, which provides a central meeting place for musicians and producers for news, self promotion and session management purposes (Digital Musician, 2006). Additionally, it offers Digital Musician Link VST plugin for latency free (i.e. timestamped semi-synchronous) audio and MIDI content transfer between peer-to-peer connected audio sequencers. There can be only one active connection at a single instance, and the process resembles remote audio capture scenario: The performer starts his sequencer, peer puts his sequencer into recording mode and starts streaming already recorded material to the performer. The performer then plays his part, which is buffered in recording machine and later synchronized into existing material. 768 kbps downstream and 128 kbps upstream connection is required, and realtime video and text chat is used for in-session communication.

6 CONCLUSION

There is no apparent reason why an ordinary domestic internet connection point could not serve as a distributed musical performance collaboration node, because the initial upstream speed does not have a great impact on the achieved latency figures. Most profound local delay source is the audio codec, which should be as efficient as possible, or eliminated from the delay budget altogether by transmitting raw audio data.

However, when networked distributed musical performance is compared to conventional music playing scenarios, the latencies might still be too high for enjoyable experience, particularly in intercontinental lineups. If participants are within a low-latency region (20 ms or so), the performance has a fair chance to be successfull. Region can be extended by using MIDI or some other higher level abstraction for control data. Tightly quantized rhythmical material might be the hardest to handle under delayed conditions, and for such scenario the NINJAM -like semantically delayed model could be the most appropriate option.

Traditional band session is also very much a social happening, so there should be at least one shared visual, speech or textual communication channel available for all performance nodes. There should also be a way to discover public sessions, a mechanism to arrange private ones, and means to manage the session during its lifetime. Models for this could be borrowed from real world. For example, there could be a virtual jazz club with house band, and anyone could listen for the jam, join in for a couple of choruses and then step down from the stage to release the network bandwidth for someone else. Participants could also collaborate in different roles, like conductors, performers or mixers.

In time, network ports might replace audio jacks in electric music instruments. There are already models from Gibson and Hartmann that employ ethernet connectivity, and Yamaha's mLan (which is Firewire-based) is supported by company's digital mixing desks and high end synthesizers. WLAN might prove to be more attractive solution however, and in future musician may just power up his synthesizer and have instant

connection to a networked music making community with participating members all over the world.

REFERENCES

Barbosa, A. 2006. *Computer-Supported Cooperative Work for Music Applications*. PhD Thesis. Universitat Pompeu Fabra. 2006. [referenced 20.12.2006]. Available: www.iua.upf.edu/mtg/publications/86df45-PhD-Abarbosa-2006.pdf

Bouillot, N. 2004. The auditory consistency in distributed music performance: a conductor based synchronization. Information Sciences for Decision Making. [referenced 20.12.2006]. Available: http://isdm.univ-tln.fr/PDF/isdm13/isdm13a123_bouillot.pdf.

Chew, E.; Sawchuk, A. 2004. Distributed Immersive Performance. *Proc. of National Association of the Schools of Music Annual Meeting (NASM).* San Diego. USA. November 22, 2004. [referenced 20.12.2006]. Available: http://www-rcf.usc.edu/~echew/papers/NASM2004/DIP-2004NASM-proceedings.pdf

CyberSImps 2003. *Distributed Improv*. CCRMA. Stanford. USA. [referenced 20.12.2006]. Available: http://ccrma.stanford.edu/groups/soundwire/cybersimps/index.html

Delaney, D.; Ward T.; McLoone S. 2006. On Consistency and Network Latency in Distributed Interactive Applications: A Survey. *Presence*. Vol. 15. No. 2. pp. 218-234.

Digital Musician GmbH. 2006. *digital musician net homepage* [online]. [referenced 20.12.2006]. Available: http://www.digitalmusician.net/index.php

DIP 2006. Distributed Immersive Performance. University of Southern California. USA. [online]. [referenced 20.12.2006]. Available: http://imsc.usc.edu/dip/

eJamming. 2006. *eJamming homepage* [online]. [referenced 20.12.2006]. Available: http://www.ejamming.com

Green, J.; Riel, M.; Thorington, H. 2006. *networked_performance* [online]. A research blog about network-enabled performance. [referenced 20.12.2006]. Available: http://www.turbulence.org/blog/

Grether, R. 2003. Virtual Performance Research Area. [online]. Institute for Applied Theatre Studies. Giessen University. Germany. [referenced 20.12.2006]. Available: http://www.netzwissenschaft.de/perfa.htm

Haas, H. 1949. The Influence of a Single Echo on the Audibility of Speech. Reproduced in *J. Audio Eng. Soc.* Vol. 20 (Mar. 1972). pp. 145-159.

Jenik, A. *et al*. 2005. *SPECFLIC* 1.0, A Speculative Distributed Cinema Project. [referenced 20.12.2006]. [online] weblog available: http://va-grad.ucsd.edu/~specflic/

Kleimola, J. 2005. *Design and Implementation of a Software Sound Synthesizer*. Master's Thesis. Helsinki University of Technology . 2006. [referenced 20.12.2006]. Available: http://www.acoustics.hut.fi/publications/files/theses/kleimola_mst/.

Lazzaro, J.; Wavrzynek, J. 2001. A Case for Network Musical Performance. *The 11th International Workshop on Network and Operating Systems Support for Digital*

Audio and Video (NOSSDAV 2001). June 25-26. 2001. Port Jefferson. New York. USA. [referenced 20.12.2006]. Available: http://www.cs.berkeley.edu/~lazzaro/sa/pubs/pdf/nossdav01.pdf

Miller, M. 2005. *VSTunnel homepage*. [online]. [referenced 20.12.2006]. Available: http://www.vstunnel.com/en/

Mäki-Patola, T. 2005. Musical Effects of Instrument Latency. *Proc. of Suomen musiikintutkijoiden 9. valtakunnallinen symposium*. Jyväskylä. Finland. March 17-19, 2005.

PICA 2006. *DANCEPOD 2006*, T.B.A. Festival Event. Portland Institute for Contemporary Arts. [referenced 20.12.2006]. [online] Available: http://www.pica.org/tba/tba06/detail.aspx?eventid=79

Point25 2004. A concert within 'Connected Performance Spaces' project. [referenced 20.12.2006]. Available: http://www.r1.kth.se/point25/

SoundWIRE 2001. Sound Waves on the Internet from Real-time Echoes. CCRMA. Stanford. USA. 1996-2001. [referenced 20.12.2006]. Available: http://ccrma.stanford.edu/groups/soundwire/

Underwood, B. 2006. *NINJAM Realtime Music Collaboration Software* [online]. [referenced 20.12.2006]. Available: http://www.ninjam.com/index.php