Sound Exchange and Performance on Internet2

Mara Helmuth

University of Cincinnati, College Conservatory of Music, Center for Computer Music
Cincinnati, OH 45221-0003

ABSTRACT

Internet Sound Exchange (ISX), an application for computer music composition, performance and improvisation for Internet2, is being developed to allow sound exchanges between multiple hosts. Sound data is located on all participating hosts and processing is distributed among these machines. This program, based on the RTcmix music programming language, runs on SGI¤ workstations. It sends sounds samples to remote hosts on the internet. A graphical user interface allows the composer or improviser to create sound textures by source sound selection, sound window timings and stochastic qualities. Before a performance event using this software, musicians on each host create source sounds to be used in the event by any means they choose, and load the sounds into the program. They may also save patterns or sound gesture parameters which can be triggered later in real time, which will structure the mix. In the performance phase, each host sends and receives mixes of sounds, based on their own sounds and chosen windowing parameters, to each other via sockets. The GUI allows the human performer on a host to make sound selection and processing choices in real time through slider manipulation or triggering of stored patterns. Any remaining decisions not made by the performer are made stochastically. Windowing of the source sounds allows for interesting and unique textural and timbral combinations, always based on the host's source sounds. The high bandwidth and quality of service of Internet2 is required for peaks in the density of sounds, and for situations involving larger numbers of participating hosts. Exchanges between three Universities are planned for the year 2000.

INTRODUCTION

Interest in improvisation comes from the earliest human musical experiences, and permeates the music of many styles and cultures. Music composition may spring from improvisation, and improvisation can be a realtime form of improvisation. My earlier experiments on the computer include PieceNow, improvisation software running on the NeXT computer around 1990, created at Columbia University, the SoundColors installation also running on the NeXT (Helmuth, 1996), for which the earlier version of the current program was named, and acoustic/electronic improvisations occurring in the Live Electronic Music courses with RTcmix taught with percussionist-composer Allen Otte at the University of Cincinnati in 1997, 1998 and 2000.

Perhaps one of the most recent developments in improvisation concepts is to involve participants on computers in remote location using technologies such as ISDN. This project is concerned with realizing performance situations between improvisers and composers to create distinctly personal and interactive sound exchanges. The participants using the Internet Sound Exchange (ISX) application may interact musically with high quality digital sound created with the most sophisticated techniques. Using precomposed sound samples in a highly flexible algorithmic mixing environment gives the improvisers control and quality of sound as well as spontaneity and interactivity in performance. As the results are dependent on each individual improviserÕs sound sets and mixing strategies, as well as the how this musical layer combines with the othersÕ contributions, unique music results from each performance and host configuration.

Flexible and reliable sound exchange requires high bandwidth, and low latency and jitter, all of which are goals of Internet2. Processing and sound storage is distributed among a number of machines on the internet, but a fast connection allows musicians to interact musically as if they were in the same room.

BACKGROUND

The ability to connect over the internet musically has been improving throughout the last decade, from the email and ftp exchanges of software and ideas, to current web streaming audio applications and performances. As the internet does not have currently the bandwidth to sustain high quality sound transmission, applications for sound exchange are appropriate uses for the high bandwidth infrastructure being developed by Internet2¤, the not-for-profit consortium led by universities in the United States, and including some international and corporate members. The University of Cincinnati presently has an OC-3 (theoretically 155.52 Mbps) bandwidth connection to the Abilene advanced backbone network.

Brad Garton and Dave TopperÕs RTCmix (Garton, 1997), found at http://www.music.columbia.edu/cmix/, are extensions to Paul LanskyÕs Cmix, found at http://www.music.princeton.edu/ in the Princeton Sound Kitchen. Garton introduced socket programming for transfer of Cmix (text score) data so that real time cmix

processing programs could be initiated on remote machines. Socket programming (Stevens, 1998) allows interprocess communication with listening and sending routines and structures by which data is transferred. For example, a host can send a message to another host to play a sequence of Karplus-Strong notes with the cmix STRUM instrument. The host that receives the message runs the cmix program to synthesize and play the notes in real time. Garton also wrote a prototype addition that sends and receives/plays sound samples. These programs were a starting point for ISX.

APPLICATION DESCRIPTION

Overview

ISX is an application which allows many hosts to send algorithmically controlled mixes of sound samples to each other. Sending the sound samples rather than score data puts the responsibility and control over the sound produced on the host which initiates the sound rather than the receiver, insures that exactly the correct sound is sent, and allows both sender and receiver to hear the sound. Each host can both send and receive sound, allowing a performer at one of the host computers to improvise in response to other performers. Components: The application consists of three components: 1., a modified RTCmix software ÒinstrumentÓNMIX, 2., an ÒplayÓ program which listens to particular sockets and plays the incoming audio data, and 3., a GUI with which the improviser interacts to hear and play sound. Other programming contributors to this project are Ico Bukvic, Ryan Meyer, and Carl McTague.

RTcmix program

The RTcmix NMIX instrument directs audio data to a remote host on a particular socket, and may play the sound on the current host as well or write to disk. This instrument was created with several additions to the RTcmix2.0 release in the front end (Minc) and system functions code modules.

Listener/Player

The ÒplayÓC program listens to sockets and plays received sounds using the SGI¤ audio library.

User Interface

The C++ Motif GUI contains a file selection browser, and track control widgets which allow the improviser to select and change source sounds, control how they are played, the densities, durations and timing characteristics of the sounds, and to make socket selections. The improviser may determine functions for envelopes and gesture control, including the probability distributions found in RTcmix. The GUI controls both the listener/player program and the initialization of NMIX commands which send samples to remote systems.

HOST CONFIGURATIONS AND BANDWIDTH

Configuration

The above design allows a maximum of flexibility in host configurations. Any host can send sound to any other host by means of UNIX sockets. Of course, the host configurations can vary widely for each performance. Each configuration is structured anarchistically by the participants as they choose to whom to listen and respond. If the people involved decide which sockets on which to send and receive, they can create a situation in which any host may improvise with any subset of the group of hosts at any time, subject to the processing limits of the workstations.

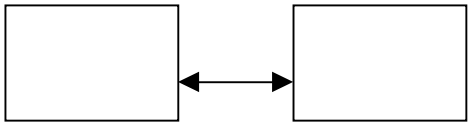

fig. 1 Two hosts exchanging stereo sound.

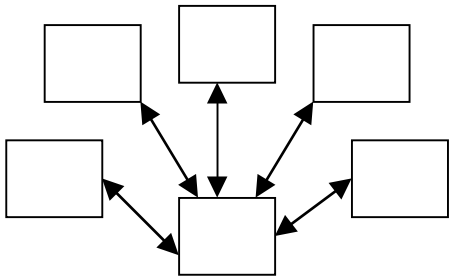

fig. 2 One host interacting with 5 hosts.

Bandwidth

While two hosts sending one stream of stereo 44.1KHz audio data to each other (fig. 1) will require a bandwidth about 350 kilobytes of data per second or 2.8 Mbps, if five hosts each send one stream of data to our host, and we send our data back to the five other hosts (fig. 2), the bandwidth needed is about 14 Mbps. If each of the five hosts sends three independent streams of audio data, the amount of data is approximately 5.3 MB/sec or 43 Mbps. If our host could transmit to and receive from five hosts, three streams of 4-channel data each, 85 Mbps speed would be required. While not every moment in the mix may be this dense, the bandwidth is needed for the maximum amount of transmission at the heaviest layerings. While the processing power of each machine has limits, and it can only assimilate so much data, the high bandwidth scenario overall allows for very dense sound layering at peak points, and flexibility for controlling each stream of data independently.

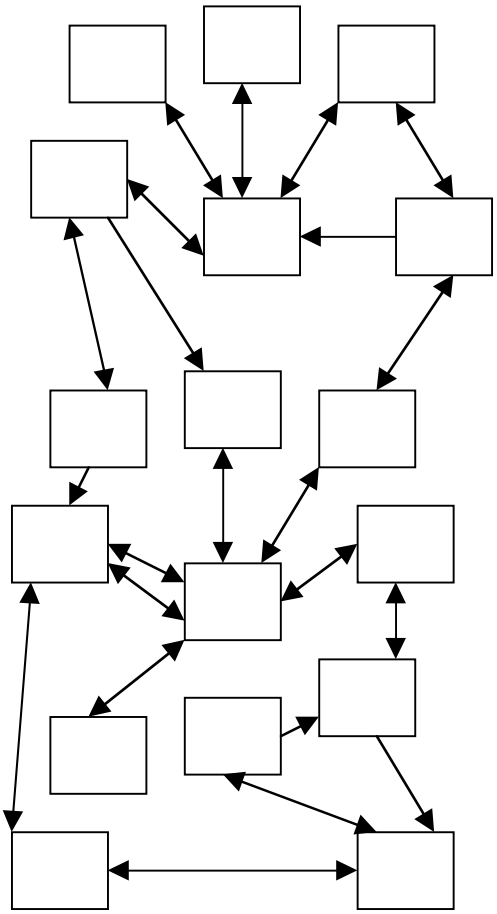

fig 3. Improvisation configuration with many hosts. This is one socket connection configuration which may exist at a particular point in a performance.

IMPROVISATION EVENTS

A sound exchange using ISX consists of two phases: setup and performance.

Setup
First, the software running on each host must be loaded with the improviserÕs own soundfiles which will form the source sound for the hostÕs audio output. In planning the improvisation, the musician can consider that designed families of sounds with cataloged relationships can make the performance easier to control and more unified. However, aleatoric aspects of the improvisation introduce unpredictability and the likelihood of unexpectedly interesting mixes to occur. After sound selections are loaded, windowing parameters for the processing of the source sound are set. For example, a long cymbal sound sample, stored on the disk as an NeXT/.au format soundfile of sampling rate 44.1KHZ, may be broken into .5 second windows occurring .25 seconds apart. These timing values can be entered into GUI fields. The performer can also choose randomized values, within a particular range and around a preferred value, for stochastic sound groupings. Socket numbers are chosen to send and listen on. The performer chooses sounds and values for a set of tracks, which can be allocated to the same or different sockets.

Performance
When the run button for each track is pressed, audio data is sent on the sockets to the other hosts. The performer can modify the window parameters and source sound distributions with interface values and sliders. The incoming sound sockets are selected and that sound is heard as well.

Events so far
A prototype version of the program was tested on the internet in spring of 1999 between Cincinnati and Oberlin CollegeÕs TIMARA studio, with composer Pauline Oliveros and Corey Arcangel, but the low bandwidth of the commercial internet speed caused breakup of the sounds into unintelligibility creating a Ò teletypeÓ sound, according to Oliveros. Test exchanges with hosts in the Cincinnati studio and the Computer Music Center at Columbia University are happening in the summer of 2000 with SGI Indy and O2¤ workstations on Internet2, so far having sent stereo 44.1 KHz audio data on 5 sockets simultaneously (about 7 Mbps) with sound received generally intactly. However any slight latency causes big problems for audio: gapping or noisy signals, and sending the same amount of data the other way causes significantly worse audio results. Further work must be done to analyze the problem, which could relate to bandwidth or workstation processing power issues, and to find a solution. Plans for more complex exchanges and exchanges with other universities including the University of Virginia and Southern Methodist University are made for the rest of this year.

IN-PROGRESS OR DESIRABLE EXTENSIONS

Expansions of the implementation are under investigation in these areas:
1. Incorporating additional synthesis or processing algorithms in addition to mixing sampled sound. As the Cmix programs already exist, it simply means recompiling them with the new version of RTcmix and involving them in the GUI musical logic. For example, layers of the real time granular synthesis instrument SGRAN could be mixed with the sampled sound layers. Processing of incoming sounds from other hosts is another option.
2. Graphical control of envelope and gesture shaping functions, as well as other parameters, to enhance performance.
3. Interactions between larger numbers of hosts, and in more complex configurations will be explored. One host may act as a server, and mix signals from many other hosts, and sending the information back to the other hosts. Multicast routing should allow efficient transmission between many hosts.

4. Multichannel work will be incorporated into the GUI. RTcmix2.0 allows 4 channels, and higher numbers of channels are already in use in some studios with later versions. This will allow the option of partitioning the incoming channels of sound from a particular host into a selected location, aiding the improviserÕs sense of interacting with sound from a particular place and person. For example, if stereo sound from Sylvia on host 2 is placed on channels 3 and 4 in the rear of the listening space for me on host 1, I will feel as though Sylvia is behind me in our virtual improvisation space.
5. The software will be made more easily customizable and ported to Linux which will encourage participation of other computer musicians.
6. Using one communication channel, or socket, for information passing between the GUIÕs of various hosts to convey information about which host will send or listen on each socket. The improviser could enter information about the sounds and sockets s/he is using into a control socket communication received by the other hosts so that the performers know which sockets to listen for.
7. Applying MIDI controller input to sound selection, windowing and track-layering parameters of sound to affect the windowing parameters, or file or socket selection.

CONCLUSION

This initial exploration of high quality algorithmic digital audio improvisation makes use of new networking technologies to allow a virtual experimental jam session to occur between remote performers on the computer. The goal to make the application available to the computer music and Internet2 communities should encourage new kinds of musical interaction, which have been so far largely untapped.

REFERENCES

Garton, B. and D. Topper. 1997. "RTCmix -- Using CMIX in Real Time." *Proceedings of the 1997 International Computer Music Conference*, pp. 399-402. San Francisco: International Computer Music Association.

Helmuth, M. 1996. "Collage: Sound Colors Installation Software." *Proceedings of the 1996 International Computer Music Conference*, pp. 251-252. San Francisco: International Computer Music Association.

Stevens, W. Richard. 1998. *UNIX network programming.v.1*. Upper Saddle River, NJ : Prentice Hall PTR.