Two Different Paradigms for Network Audio Performance with a Laptop Ensemble

Eric Sheffield
School of Music
Louisiana State University
esheff1@lsu.edu

William Thompson
School of Music
Louisiana State University
wthom53@lsu.edu

Edgar Berdahl
School of Music & CCT
Louisiana State University
edgarberdahl@lsu.edu

ABSTRACT

Two different paradigms for network audio performance in a laptop ensemble are explored. On one hand, the composition "Telephone" demonstrates that wireless networking technology is feasible for sending discrete sound samples back and forth between machines for a game-like interaction. On the other hand, realtime audio streaming between laptops is currently not very feasible using 802.11 wireless technology and is best realized with a wired network. As an example, the composition "Resuscitation" uses a wired network to stream audio in realtime, distributing a physics-based virtual instrument by creating a digital waveguide network spanning between the laptops.

1. INTRODUCTION

Compelling networked musical performance predates modern networking standards. For example, Max Neuhaus' *Broadcast Works* utilized telephones and radio to invite thousands of participants into a "virtual aural space [1]." As another example, the members of the well-known League of Automatic Composers connected their Commodore KIM-1 microcomputers in a circular network, each transmitting and receiving one stream of data that was then uniquely mapped to musical parameters according to the programmer's specifications [2].

Further advancements in technology have facilitated the flexibility of musical networks, prompting explorations of collaboration, interaction, and the representation of musical time and space. A great deal of research in networked performance has focused on telematic music, in which musicians in remote locations can perform together over the internet [3, 4, 5]. Telematic research often deals with the practical and conceptual issues of the format, including rhythmic synchronization [6] and the establishment of stable high-fidelity audio streams [7].

Digital music ensembles provide a readymade and reconfigurable platform for collaborative performance on local computer networks [8]. In particular, the standard laptop ensemble arrangement usually includes a wired and/or wireless network, means for individual amplification, and consistent hardware and software configurations for all players [9]. To date, most of the explorations of local networking in laptop ensembles have focused on the distribution of control data, resource management, ensemble communication, and performance synchronization rather than audio [10, 11, 12, 13].

The compositions described in this paper, *Telephone* and *Resuscitation*, present two different methods of networked audio realized in a laptop ensemble setting. We explore applications for sharing audio using wireless and wired networks, respectively.

2. A WIRELESS NETWORKED AUDIO COMPOSITION

2.1 Concept

All other things being equal, wireless technology might, in the long term, become the convenient networking option for laptop ensembles. However, the authors believe that current 802.11 scheduling protocols do not allow for audio to be reliably streamed live over a network with realtime latencies [14]. This means that it is currently challenging to realize networked audio compositions with wireless technology.

However, when the second author proposed the concept for the composition *Telephone*, some excitement ensued because it was realized that it would be feasible to realize *Telephone* wirelessly. The reason for this is that *Telephone* specifies the recording of a 10-second audio sample, which is then "given away" to another laptop station that processes the sample further, and then the cycle continues with the sample getting passed on further.

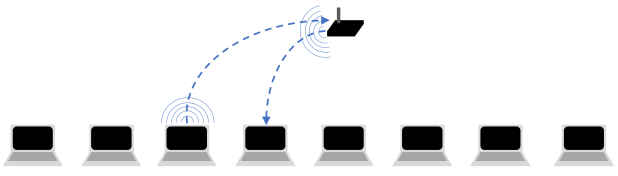

Figure 1. An illustration of a complete audio sample (`mysound.wav`) being sent wirelessly from laptop 3 to laptop 4. The time to transfer and confirm takes about 1 to 2 seconds.

2.2 Details About the Composition

Telephone takes its name and operation from the child's game telephone in which a message is whispered in the ear of each child going down the line. The final and usually surprising resulting phrase is revealed at the end of the line.

However, in this piece, performers and audience members hear the message as it develops and is passed from

one ensemble member to another. In *Telephone*, performers take turns updating an audio buffer with new versions of a wave file which can be sent using `scp` to any of the computers on the local network. The manipulation of audio by performers is accomplished through use of phase vocoders and pitch shifting. As the initial recorded sound is passed from one ensemble member to another, it begins to resemble its original form less and less taking on a new sound all its own. A recording of *Telephone* can be heard at the link below. [1] Figure 2 provides an indication of how a particular performance of *Telephone* might proceed.

Section I. The performers are arranged in a line across the stage. The composition begins with two performers, one at each end of the line, speaking something into a microphone. This initial recording is recorded by the person speaking and is limited to a ten second window of time. Both outer-line performers would immediately launch their sampled voice in their Max patch interface. Once launched, the next performers (moving towards the middle of either extreme of the line) re-record the received audio sample while altering it using a phase vocoder and pitch shifting as desired. After the recording process, the performers send the new audio to the next performer in line and this process starts again.

After any performer sends recorded audio, her or his volume will automatically fade away in five seconds while the performer who receives the message will begin making sound fading in over five seconds. The process continues until the line of re-samplers is complete. So, eventually we will have a sample of a sample of a sample of a sample, etc. Because there were two initial samples started at the extremes of the line, at some point these two traveling ideas will cross paths. Once the two separate samples reach the middle two computers, the performer who sends the next message must skip over the next person in line. This done to avoid losing one line of messaging since a message from the other path of origin would delete the opposite. Once this is accomplished the message can continue to be sent from both paths until it reaches the end of the line.

Section II. Once each sample reaches the other side of the line, all performers observe a grand pause. Following this pause, each performer will chose one horizontal point on the screen on which to focus his our her mouse and simultaneously fade in from the bottom of the screen, labeled *ppp*, to the top of the screen, labeled *fff*, or until the desired volume has been achieved. Together, the ensemble holds this sound and dynamic for a maximum of one minute. Next each performer make very small circles around his or her chosen point. Performers continue to make circle motions and gradually increase the diameter of each circle until each performers circle diameter is the size of the computer screen. This process should take no longer than 20 seconds. Section II ends as all performers decrescendo into silence before proceeding to section III.

Section III. At the beginning of section III, ensemble members on each end of the line become active again and now possess the altered audio file started by the opposite side of the line. These audio files crescendo back to a desired volume and are then sent to a performer (computers 1-8) chosen by the two now active performers. For the remainder of the piece, these two audio samples are sent to remaining ensemble members in any chosen order. Note, during this section performers must be careful not to send audio immediately to the computer which now possesses the other audio sample, unless this is a desired action. Once each performer has sent this second audio file, he or she walks off stage, leaving any remaining audio samples with one performer who decrescendos to silence to end the composition.

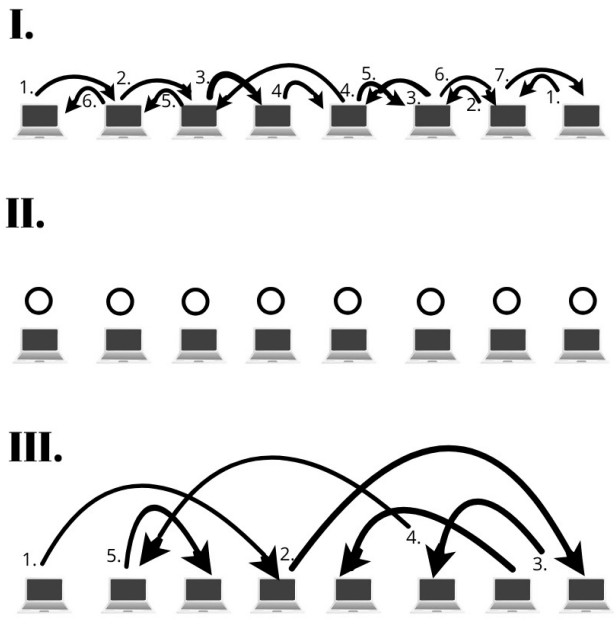

Figure 2. A graphical representation of how the composition *Telephone* might proceed. Each arrow represents a sound sample being sent to a new target laptop station, which processes the sample further while it is played back.

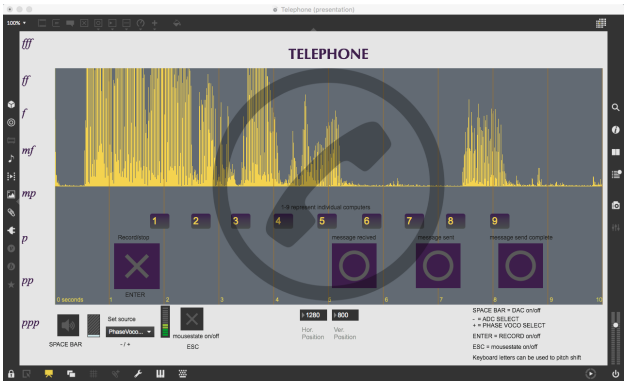

Figure 3. The Max patch performance interface for *Telephone* allows performers to record the initial sound, manipulate it, and send it to other ensemble members. Additionally, the interface serves as a way to navigation trough time by moving the mouse state horizontally. Vertical mouse state adjustment controls dynamics (the top of the screen being the loudest and the bottom being the most quiet).

2.3 Technological Realization

Telephone was realized using a single version of a Max patch that was deployed to eight Apple MacBook Pro laptops. The design of the Max patch is built around the realization that each laptop needs only to have a single copy of

[1] `https://www.dropbox.com/s/3utaj6tzxu224ne/Telephone%20Audio%20Submition3.wav?dl=0`

its sound sample `mysound.wav`.

The `shell` object in Max made it possible to call the `scp` terminal program directly from Max. In this manner, the Max patch enables each laptop performer to send her or his copy of `mysound.wav` to any other laptop in the configuration. This will then cause the receiver's copy of `mysound.wav` to be instantly overwritten.

Each laptop also uses the `filewatch` object to watch the file `mysound.wav` so that if it is ever overwritten, it is immediately loaded into Max, the volume is turned up, and the person is invited to perform with the sound. This is one of the important aspects of *Telephone* that invites game play. Each performer always has the ability to overwrite the other performers' sounds, even potentially all of them at almost the same time.

With the authors' particular laptop ensemble setup using 802.11n, it took about 1 to 2 seconds for each transfer to complete. (Indeed, despite the fast wifi machine-to-router mean ping time of 3.2 ms (std dev 1.1 ms), it took much longer to send a whole file under load than simple ping messages.) This was clearly too slow for realtime audio, but nonetheless fast enough for the game-like interaction to run its course at a sufficiently fast speed.

There were some security implications of allowing sound files to be sent from any station to any other station. Max needed to the `scp` command to run without a password, so each computer needed to have an ssh key setup for it on *every other station* [15]. In the case of eight computers, this involved setting up $8 \times 7 = 56$ ssh keys. However, the authors were able to setup and test all of these ssh keys within under one hour, which enabled the wireless realization of the networked audio composition *Telephone*.

3. A WIRED NETWORKED AUDIO COMPOSITION WITH REALTIME STREAMING

3.1 Concept

The laptop quartet *Resuscitation* by the first author was motivated by an interest in composing for virtual multi-user instruments that afford interdependent actions by the performers. Related work on interdependent performance has been accomplished using networked systems, e.g. the three computer setup of the League of Automatic Composers [2] and Weinberg's Beatbugs [16]. These examples are built on a framework of data mappings, topological organizations, and/or systemic hierarchies that allow for the organization and interpretation of information exchanged among players. In order to explore more direct and *physically plausible* interdependencies, we decided to use physics-based modeling synthesis for instrument design.

A mental model of the instrument and the desired performer interactions was sketched out before any software work began. The virtual instrument resembles an assemblage of four strings coupled to a resonant plate. Each player is assigned a string and one corner of the plate and performs the instrument by exciting the string or plate, changing the parameters of the string or plate, and damping the plate. As such, the primary interdependencies occur through simultaneous damping and excitation gestures by different players on different parts of the instrument and sympathetic resonances that occur through the string-plate coupling.

The realization of the instrument in *Resuscitation* is accomplished using digital waveguides distributed over the network through realtime audio streaming. In contrast to *Telephone*, this required the use of a wired network. Further technical details about the instrument's design are discussed below.

3.2 Details About The Composition

Resuscitation was approached much like a percussion quartet, driven primarily by rhythmic passages traded among players and explorations of timbre. Sections of the piece are delimited by parameter changes to the instrument that change its timbre and resonant behaviors.

Although each of the four strings of the instrument can produce clear pitches, they are not perfectly tuned to a Western scale and change pitch infrequently. This negates most potential for melodic material. Instead, the focus is put on drones and sustained dissonance.

Instructions to excite the plate with taps and strikes, excite the string with plucks, and damp the plate are specifically notated in the score using a notation system similar to that used by multi-percussion literature in which specific actions are given their own line or space on a musical staff. Rhythmic passages are often a result of the combination of plate taps or strikes by one or more players in conjunction with rhythmically notated damping gestures by another player.

A video of a live performance of *Resuscitation* can be found at the link below. [2]

3.3 Technological Realization

The technical design of *Resuscitation* was driven by a focus on accessibility. The aim was to make a piece of electroacoustic chamber music that could be feasibly organized by performers or ensembles with modest technological proficiency and limited resources. Additionally, our hope was that it would appeal to both electronic performers as well as instrumentalists, especially technically savvy percussionists. As such, we decided on a combination of software and hardware that is familiar, inexpensive or free, and easy to configure using provided instructions.

3.3.1 Software

Since realtime audio transmission on a local wired network is still subject to small amounts of latency, the software instrument for *Resuscitation* exploits the delays that occur between computers by using the digital waveguide synthesis technique [17]. Chafe et al. used a similar approach to build digital waveguide string models over the internet in SoundWIRE, but the significantly greater transmission time limits fundamental pitch range near the bottom of human perception [18].

Our instrument is distributed among four similar Max patches running on four separate computers. Each patch includes a portion of the instrument built primarily within Max's `gen~` environment, where patching operations occur at the sample level. The computers transmit audio to

[2] `https://youtu.be/dxOa_NOT9Rw`

each other using the NetJack2 component of Jack2. [3] Laptop 1 operates as the NetJack2 master, with laptops 2-4 operating as slaves.

The plate portion of the instrument is a 2D digital waveguide network (see Figure 4 in which ≋ indicates a lowpass filter). Each laptop hosts half of a bidirectional delay line with a single inverting junction acting as an edge. Laptops 2-4 send and receive audio to a 4-port scattering junction [4] that connects the waveguides on laptop 1. The delay line on laptop 1 is also lengthened to compensate for the delay incurred by NetJack2 transmission. This extra delay is optional but can assist with predictable tuning of the digital waveguide network.

The initial interest in distributing the plate part of the instrument among multiple laptops for *Resuscitation* was primarily for practical purposes. Additional software work aside, it was more feasible to implement a fully connected local wired network in our standard laptop ensemble configuration than a custom setup with additional multi-channel audio interfaces and instrument interfaces that required extra connections between stations. However, it is easy to imagine further potential for physics-based virtual instruments distributed among multiple laptops. The 4-port scattering junction used on laptop 1 for connecting the waveguides could be designed to flexibly accommodate additional computers, changing the characteristics of the instrument in relation to the size of the ensemble. Additionally, although digital waveguide synthesis is computationally efficient, more complex models could be realized on older or less advanced hardware if parts of the instrument are distributed among multiple devices.

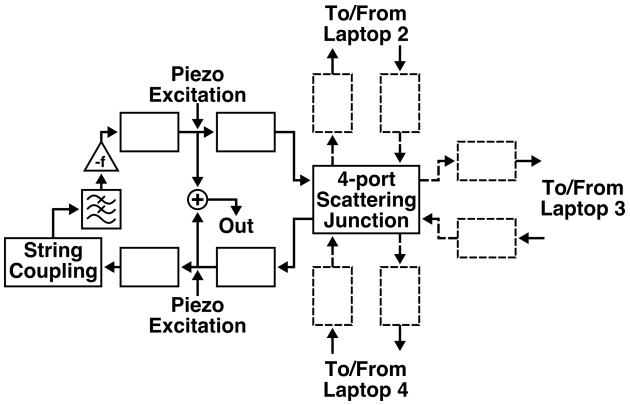

Figure 4. A diagram of the portion of the digital waveguide network hosted on each computer. Computer 1 adds a 4-port scattering junction that connects the digital waveguides together from all four computers.

The strings of the instrument are 2-axis digital waveguides (see Figure 5). Each laptop runs its own string instance, and one axis of each string is coupled to the local portion of the waveguide network. Coupling of string axes and string-plate coupling both refer to the example provided by Berdahl and Smith [19].

Jack2 is initialized through Terminal, which allows for most of the startup process to initiate from a script that can be customized for each setup. For example, to create master-slave connections in NetJack2, the hostname of each computer is referenced in the name of any send or return. This requires some minor script editing the first time the piece is run on a particular arrangement of computers.

[3] http://jackaudio.org

[4] https://ccrma.stanford.edu/~jos/pasp/Rectilinear_2D_Mesh.html

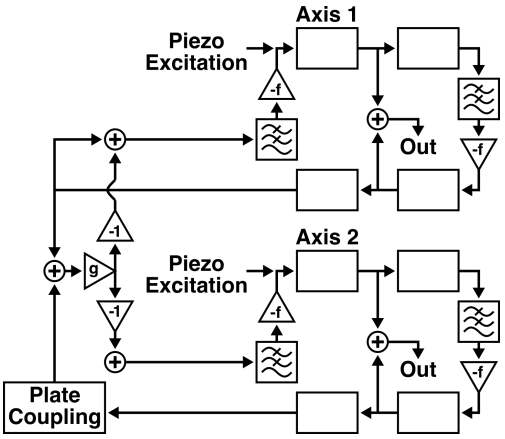

Figure 5. The 2-axis digital waveguide string component, which is identical on all four computers.

3.3.2 Hardware

The current version of *Resuscitation* requires four Mac computers, four 2-channel audio interfaces (one per computer), four custom Arduino and piezo based interfaces (one per computer), and a wired ethernet router. Testing and composition occurred primarily using laptop and desktop models of various vintages from 2011-2018 running different versions of macOS. All models were connected using gigabit ethernet ports (built-in or via USB-C ethernet adapters) to a home gigabit ethernet router. The goal while designing and constructing the software instrument was to try and recreate a flexible laptop ensemble situation in which computers may not be uniform or recent models.

Initial tests for the instrument in *Resuscitation* were conducted in late 2017 using a Max patch running on a single laptop. Players performed the piece by using standard MIDI controllers, triggering enveloped white noise bursts to excite the model and changing control parameters to modify damping characteristics. Dissatisfaction with the responsiveness and expressiveness of the MIDI controllers in conjunction with the physics-based model lead to the design of a custom interface using affordable, non-specialized, and easily sourced parts, including piezos, scrap wood, and miscellaneous hardware and electronics (see Figure 6). Efforts were made to ensure that they could be constructed using minimal electronics knowledge and tools. Cost of the interfaces is estimated to be around $30 USD each if purchasing third party Arduino-compatible clones, and this can be further reduced by using parts or materials at hand. Detailed instructions for interface construction are included with the software for *Resuscitation*.

The use of piezos and contact mics for physics-based model excitation can be found in several projects, e.g. [19, 20, 21]. This method of audio-rate excitation allows for nuanced and intuitive control as variations in playing technique (e.g. soft and hard tapping, scraping, use of auxiliary implements) have noticeable effects on the resulting sound of the model [22]. Our interface uses a piezo disc mounted under a damped plastic plate for excitation of the waveg-

uide network and a piezo tine for excitation of the string. Each piezo is connected to a separate input channel on the audio interface. Players perform the piece by tapping or striking the plastic plate and plucking the tine.

A Force Sensitive Resistor (FSR) connected to an Arduino Nano transmitting serial data to the Max patch is used to control the damping characteristics of the instrument. Tests were conducted using homemade pressure sensors in an attempt to further reduce costs; however, sensor range and dependability using readily available materials were insufficient when compared to commercial versions. This FSR setup proved to be very responsive and physically satisfying as it mirrored the actions that would be performed to dampen a real resonant plate or membrane.

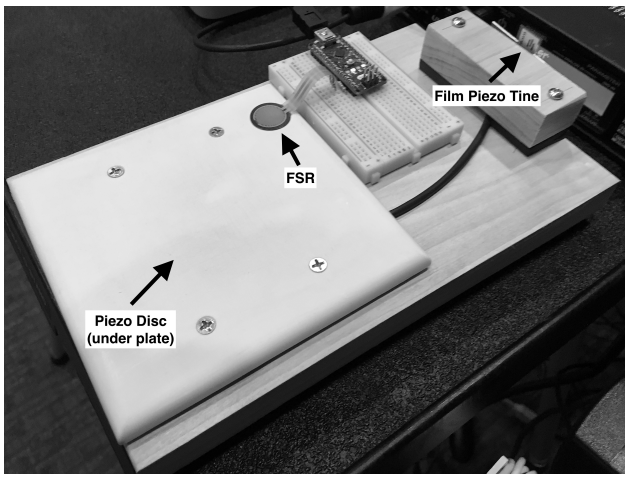

Figure 6. The custom interface used for *Resuscitation*.

3.3.3 Tuning the Network

In the interest of further evaluating the efficacy of NetJack2 for digital waveguide synthesis, tests were conducted at multiple settings. It should be noted that on a NetJack2 master, audio is synced with the local sound interface, so no resampling is required. However, NetJack2 slaves must each resample the audio as it arrives and is transmitted. This allows the locally simulated audio on each slave to be synchronized with the remotely received and transmitted audio, which is synced to the clock of the master. As such, the following measurements, though we found them to be consistent, should be considered approximate as all instances of NetJack2 are not referencing the same clock. Additionally, these findings relate specifically to our combination of Max and NetJack2 in *sync* mode running on four computers and may not be consistent using other applications, *async* mode, or additional channels.

Network latency, and therefore length of delay lines and waveguide fundamental frequency limits, is primarily determined by two parameters of NetJack2 frames per period *-p*, which is set on the master, and network latency *-l*, which is set on each slave. Higher values for either of these settings incur additional latency. In our testing, roundtrip latency *L* of audio signals was found to be consistently determined by the simple equation:

$$\mathbf{L} = p(l + 2) \tag{1}$$

where *p* is the NetJack2 *-p* setting and *l* is the *-l* setting. For example, a *-p* of `64` and *-l* of `2` produced an *L* of 256 samples at a sampling rate of 44.1kHz. Using the following equation from [17]:

$$f_S/f_1 = \mathbf{N} \geq \mathbf{L}, \tag{2}$$

where $\mathbf{N}$ is the total delay around the digital waveguide loop, f_S is the sampling rate, f_1 is the fundamental frequency, and it is for simplicity momentarily assumed that there are no filters installed in the loop. Consequently, it would follow that $f_1 \leq f_S/\mathbf{L}$. With *-p* of `64` and *-l*, this enables a fundamental frequency range limited to a maximum of approximately 172Hz. While such a setting would likely provide for interesting low register compositions, through further testing, we were pleasantly surprised to be able to maintain a stable NetJack2 digital waveguide network with *-p* `32` and *-l* `1` on our test configuration for at least ten minutes with no audible artifacts or errors reported by Jack2. Referencing the equations, this setting allows fundamental frequencies up to 459Hz. For context, this is sufficient to reproduce fundamental frequencies beyond the entire left side of a piano or all open strings of a guitar in standard tuning. However, further testing on other computer and audio interface configurations was less stable at these extreme settings and seemed to be dependent not only on the processing speed of the computer but also the driver used by the audio interface. As such, it is recommended to use more conservative settings (i.e. higher *-p* values) that still meet the needs of the desired instrument design. For *Resuscitation*, we utilized the settings *-p* `64` and *-l* `1`.

4. CONCLUSIONS

The exploration of network audio performance is especially appealing within the context of a laptop ensemble. Through the experiences of composing these two pieces of music and implementing the technology needed to realize them, it is evident that the wired and streaming aspects of *Resuscitation* create interest in the immediacy of their effects within the composition. On the other hand, *Telephone* demonstrates that the limitations created by using a wireless network can create value in the compositional structure of a piece of music.

Specific observations within *Resuscitation* include the following: NetJack2 is a viable platform for digital waveguide synthesis in a laptop ensemble setting, with a wide pitch range and stable operation. Distributing a physics-based virtual instrument over a network provides for practical distribution of model excitation and control from multiple performers on individual laptops and distribution of localized audio. Predictable and consistent roundtrip delay times using NetJack2 allows for deliberate pitch assignments if desired. Additionally, a prototype instrument can be created on a single laptop during the composition phase if network delay times are properly emulated in software.

When considering *Telephone*, it has been noted that it is generally challenging to realize networked audio compositions with wireless technology. However, the message-based compositional structure as seen in *Telephone* is a viable procedure. Any networked audio scenario that requires transfer times of 1 to 2 seconds or more, given a sim-

ilar file size, can be considered feasible. With this knowledge, it can be concluded that there must be a vast number of yet imagined compositions in which wireless networked audio is a satisfactory and useful solution. The messaging technique in *Telephone*, using `scp` and ssh key generation, is very flexible and efficient. Additionally, this wireless audio network can be set up relatively quickly. Once configured, any computer has the ability to send messages with ease to any of the computers on the local network. This aspect of *Telephone* is particularly interesting. With this freedom comes a unique relationship between performers and creates a need for emphasis regarding the order of messaging. For example one performer can easily send his or her message to every other performer in the ensemble and therefore overwriting all audio. While this makes performance a bit more difficult, it can be seen as an exciting option for "game-like" play in future compositions. In this case, our task has been interrupted by a mutiny. The technical aspects of *Telephone* are eminent and can be seen as not only one composition but an instrument for which to compose or improvise many pieces of music.

Overall, the question of realtime versus non-realtime control is a perennial issue in computer music. Realtime control can afford exciting and compelling interactions, but realizing realtime control often involves trying to overcome a wide array of technical challenges (latency, drivers, bandwidth, etc.). In contrast, non-realtime control can become more appealing due to its simplicity, and in the case of this work, non-realtime control enabled the realization of a new composition over a wireless network.

5. REFERENCES

[1] M. Neuhaus, "The BROADCAST WORKS and AUDIUM," in *Zeitgleich: The Symposium, the Seminar, the Exhibition*. Transit, 1994.

[2] J. Bischoff, R. Gold, and J. Horton, "Music for an Interactive Network of Microcomputers," *Computer Music Journal*, vol. 2, no. 3, pp. 24–29, 1978.

[3] R. Rowe and N. Rolnick, "The Technophobe and the Madman: An Internet-2 Distributed Musical," in *Proceedings of ICMC*, Coral Gables, FL, 2004.

[4] P. Oliveros, S. Weaver, M. Dresser, J. Pitcher, J. Braasch, and C. Chafe, "Telematic music: six perspectives." *Leonardo Music Journal*, vol. 19, no. 1, 2009.

[5] M. Burtner, S. Kemper, and D. Topper, "Network Socio-Synthesis and Emergence in NOMADS," *Organised Sound*, vol. 17, no. 1, pp. 45–55, 2012.

[6] J. Caceres and A. Renaud, "Playing the Network : the Use of Time Delays As Musical Devices," in *Proceedings of ICMC*, Belfast, Northern Ireland, 2008, pp. 24–29.

[7] J. Caceres and C. Chafe, "JackTrip: Under the Hood of an Engine for Network Audio," *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, 2010.

[8] K. Umezaki and I. Hattwick, "Approaches to Collaboration in a Digital Music Ensemble," in *Proceedings of NIME*, Ann Arbor, MI, 2012, pp. 466–469.

[9] D. Trueman, "Why a laptop orchestra?" *Organised Sound*, vol. 12, no. 2, pp. 171–179, 2007.

[10] C. Burns and G. Surges, "NRCI: Software Tools for Laptop Ensemble," in *Proceedings of ICMC*, Belfast, Northern Ireland, 2008.

[11] L. Wyse and N. Mitani, "Bridges for Networked Music Ensembles," in *Proceedings of ICMC*, Montreal, Canada, 2009, pp. 443–446.

[12] S. D. Beck and C. Branton, "Reconsidering Laptop Orchestras as a Computational Grid for Music Performance," in *Proceedings of ICMC*, Huddersfield, UK, 2011, pp. 460–463.

[13] S. W. Lee, J. Freeman, and A. Collela, "Real-Time Music Notation, Collaborative Improvisation, and Laptop Ensembles," in *Proceedings of NIME*, Ann Arbor, MI, 2012, pp. 21–23.

[14] S. Letz, N. Arnaudov, and R. Moret, "Whats new in JACK2?" in *Proceedings of the Linux Audio Conference*, Parma, Italy, 2009.

[15] D. J. Barrett, R. E. Silverman, and R. G. Byrnes, *SSH, The Secure Shell: The Definitive Guide*. Sebastopol, CA: O'Reilly Media, Inc., 2005.

[16] G. Weinberg, R. Aimi, and K. Jennings, "The Beatbug Network A Rhythmic System for Interdependent Group Collaboration," in *Proceedings of NIME*, Dublin, Ireland, 2002, pp. 186–191.

[17] J. O. Smith, "Physical Modeling Using Digital Waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[18] C. Chafe, S. Wilson, and D. Walling, "Physical model synthesis with application to internet acoustics," in *Proceedings of ICASSP*, Orlando, FL, 2002, pp. 4056–4059.

[19] E. Berdahl and J. O. Smith, "A Tangible Virtual Vibrating String: A Physically Motivated Virtual Musical Instrument Interface," in *Proceedings of NIME*, Genoa, Italy, 2008, pp. 299–302.

[20] D. Schlessinger and J. O. Smith, "The Kalichord: A Physically Modeled Electro-Acoustic Plucked String Instrument," in *Proceedings of NIME*, Pittsburgh, PA, 2009, pp. 98–101.

[21] P. Dahlstedt, "Physical Interactions with Digital Strings - A hybrid approach to a digital keyboard instrument," in *Proceedings of NIME*, Copenhagen, Denmark, 2017, pp. 115–120.

[22] R. H. Jack, J. Harrison, F. Morreale, and A. McPherson, "Democratising DMIs: The Relationship of Expertise and Control Intimacy," in *Proceedings of NIME*, Blacksburg, VA, 2018, pp. 184–189.