# WAI-KNOT
# (Wireless Audio Interactive Knot)

Adam Douglas Smith

B.A. Physics and Studio Art, Bowdoin College, Maine 1999
B.S. Mechanical Engineering, Columbia University, N.Y. 1999

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning in Partial Fulfillment
for the degree of Master of Science in Media Arts and
Sciences at the Massachusetts Institute of Technology

September, 2001



**Adam Douglas Smith**
Program in Media Arts and Sciences
August 10, 2001

Certified by
**Dr. Michael Hawley**, Co-Advisor
Me

Ce
**Dr**
Senior Research Scientist
Program in Media Arts and Sciences

aduate Studies
Program in Media Arts and Sciences

# WAI-KNOT
# (Wireless Audio Interactive Knot)

Adam Douglas Smith

Submitted to the Program in Media Arts and Sciences,School of Architecture and Planning in Partial Fulfillment for the degree of Master of Science in Media Arts and Sciences

**Abstract**

The Sound Transformer is a new type of musical instrument. It looks a little like a saxophone, but when you sing or "kazoo" into it, astonishing transforms and mutations come out. What actually happens is that the input sound is sent via 802.11 wireless link to a net server that transforms the sound and sends it back to the instrument's speaker. In other words, instead of a resonant acoustic body, or a local computer synthesizer, this architecture allows sound to be sourced or transformed by an infinite array of online services, and channeled through a gesturally expressive handheld.

Emerging infrastructures (802.11, Bluetooth, 3G and 4G, etc) seem to aim at this new class of instrument. But can such an architecture really work? In particular, given the delays incurred by decoupling the sound transformation from the instrument over a wireless network, are interactive music applications feasible? My thesis is that they are. To prove this, I built a platform called WAI-KNOT (for Wireless Audio Interactive Knot) in order to examine the latency issues as well as other design elements, and test their viability and impact on real music making. The Sound Transformer is a WAI-KNOT application.

Thesis Co-Advisor: Dr Michael Hawley
Thesis Co-Advisor: Dr Andrew Lippman

# WAI-KNOT
# (Wireless Audio Interactive Knot)

Adam Douglas Smith

The following people have served as readers for this Thesis:

Media Laboratory

**Dr. Andrew Lippman**
Senior Research Scientist
Program in Media Arts and Sciences

**Tod Machover**
Professor of Music and Media
Program in Media Arts and Sciences

**Mark Dolson**
Director of Audio Research for the Creative Advanced Technology Center
Creative Labs

# Mark Dolson Biography

Mark Dolson is the Director of Audio Research for the Creative Advanced Technology Center. Dolson received his B.S.E.E. from Carnegie-Mellon University in 1976 and Ph.D in Electrical Engineering from Caltech in 1983. He worked for 11 years as a Research Scientist at the University of California, San Diego, conducting research in the areas of music perception, digital audio signal processing, speech recognition, and neural networks. In 1994, he joined Creative as a Research Scientist and was promoted to Manager of the DSP Group in 1996. He assumed the role of Director in April, 2000, and am responsible for the continuing development of proprietary audio digital signal processing technology for games, music, and voice.

# Contents

# List of Figures

# Acknowledgements

The following persons, in many ways, known best by each of them, have made gracious and indispensable contributions which have materially assisted me in the preparation of this thesis.

Michael Hawley

Tod Machover

Tristan Jehan

Ari Epstein, Joan and Noah

Mike White

My parents

Brooks Parker

Dana Spiegel

Heather Childress

Phil Frei

Mike Ananny

My gratitude, debt and thanks to each of them will always be remembered.

# 1 Introduction

Broadband wireless technology, employing computer and Internet access, is rapidly developing, improving and expanding. Present technology for wireless devices has opened a window of opportunity for researchers, designers and developers to explore and investigate the rich variety of relatively new applications now available. I have chosen for this thesis the challenge of building a software system, within the context of new wireless technology, to investigate the potential of transporting and transforming high quality sound and music among wireless handheld devices operating in real time.

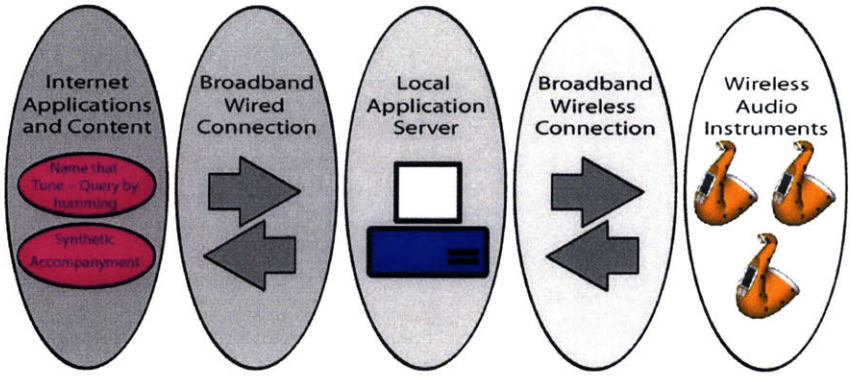
Handheld devices, reminiscent of personal music players, may record and play sound. Likewise, they may be wirelessly linked to Internet and audio services. Developers and users, for example, could tap global music and sound effects libraries; implement Karaoke synthetic accompaniment with tempo tracking; or perform other audio transformations. The system I have developed is a platform to test the viability of such applications. The system is named WAI-KNOT (for Wireless Audio Interactive Knot).

The WAI-KNOT system employs server-client architecture. The server uses an open architecture to facilitate the receipt and transmission of audio streams from many sources including handheld devices. The handhelds are designed to record and playback CD quality sound in real time. I have examined in detail the issue of management of the delay in the roundtrip of audio from handheld to server to handheld.

A wealth of online "soft" media services has recently been developed. Streaming media systems, such as RealAudio and QuickTime, connect a PC to live radio and, additionally, to vast databases of music indexing systems, including Napster and CDDB, which help to organize and search for

**Figure 1:**

*Future Wireless Architecture*



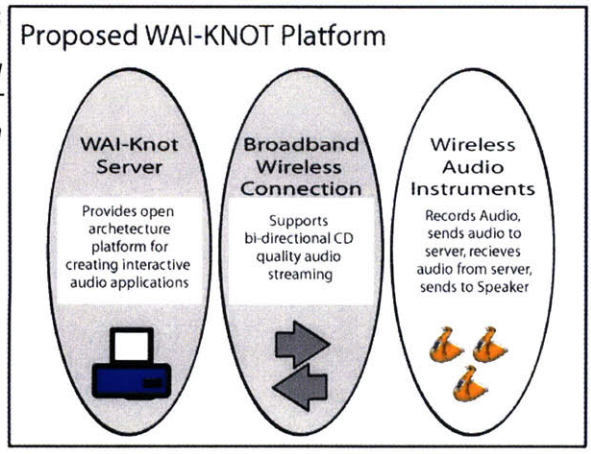

desired services. Communication systems, exemplified by Net2Phone and Dialpad, provide "free" Internet phone services. A wave of handheld devices has simultaneously emerged, including pocket MP3 music players and cell phones that receive radio and play MP3 recordings. Handhelds, however, have heretofore been simple caches. A user may download into the handheld, and play back. User interaction is minimal. Existing handhelds are passive devices. They are not designed for interactivity with the user, such as a musical instrument.

New wireless networks will emerge and substantially change things. Handheld devices will be augmented with fluid networked links. Today, for example, a person may walk a mile down Market Street in San Francisco and wirelessly connect to the net via 802.11b at 11MB/sec. Current cell phones and Walkman radios do not access the net for music, but someday they will, and when they do, a virtually unlimited library of audio, and more intriguingly, live audio processing services, will be available.

In the future instead of downloading a song into an MP3 handheld, the user could simply hum a tune into the handheld; the hum could be transmitted to an online "name that tune" lookup service, the song would be retrieved, and the user would hear the music. And if the user continued singing, the playback service could fade out the recorded vocal

**Figure 2:**

*Proposed WAI-KNOT system*



track and keep the instruments in synch and in tune-with the user's voice. While jogging, this new species of "Walkman" could synch your stride and adjust the music to the user's pace. The music could be used, in fact, to push the user faster and faster.

A new system architecture could synthesize some important but previously disjointed elements. The new handheld net device will be a two-way, high quality audio device connected via the wireless network to online services. Servers will be capable of listening to the audio from a handheld device's microphone, process the audio and transmit audio back to the handhelds. The system could, in theory, support an extraordinary variety of new exciting applications.

A telephone service, for example, could provide, perhaps for the first time, CD quality stereo, a user's own background music, sound effects, an easy "party line" capability, and more. The new system will usher in many expressive possibilities. An open server architecture will allow designers and developers to build in a free and liberal manner.

Quality live music, of course, offers wonderfully demanding and enjoyable applications. The handheld, in such a system, becomes a musical instrument itself, which accesses online service to amplify the expression. An orchestra of synthesizers, with infinite variety and scale, will be available, and the sound to and from many performers will be knitted together and transformed in new ways.

All MIDI systems, to date, have been limited by the contents of the rack. Even though an Internet link may be available, the existing systems employ particular hardware, and are not designed for a post-internet world. The prospect of a fresh new approach, employing a new generation of wireless net handhelds, real time online processing, and live interaction, is both exciting and challenging.

Major issues involve the design of new software architecture to permit the new applications. Constraints to network performance must be assessed and addressed. Latency must be managed within the requirements of net-

work channeling and processing. This thesis contends that the fundamental latencies can be managed. The platform, which I have designed, addresses the problem of latency.

# 2 Motivation

Interactive projects in the MIT Media Lab and current wireless technology provide the motivation for the WAI-KNOT platform. Researchers at the Media Lab have produced a number of projects, which utilize interactive audio applications, including synthetic accompaniment, duets at a distance, melody retrieval on the Internet (a query by humming system) digitally collaborative musical instruments, hyperinstruments, etc. [Chai 00] [Blackburn 98] [Oliver 97][Machover 94] [Rigopolous 94] [Machover 92].

The Toy Symphony, another Media Lab project, seeks to develop new musical toys and instruments that encourage children to create music. These new electronically enhanced toys will help remove some of the skill barriers to enjoying the creative experience of composing music. The WAI-KNOT platform, presented in this thesis, is a direct descendant of the Toy Symphony project.

Disadvantages of some of the current implementations involve limited portability, large size, and the difficulty of scaling. Wireless systems, in contrast, provide portability, small size, and scalability. Present wireless applications, however, incorporate more passive interaction due to latency. WAI-KNOT addresses latency within a closed network, providing near real time functionality. WAI-KNOT, this thesis contends, combines the many benefits of a wireless system and provides a useful and practical platform for designing and building a rich variety of interactive audio applications.

# 3 WAI-KNOT Platform

The WAI-KNOT platform includes a collection of software programs which provide full duplex wireless communication of high quality audio between handheld wireless devices and a remote server. The platform contains two major parts, the remote server and the handheld devices. To further describe the WAI-KNOT platform, I will first discuss the design goals for the system, then describe building the WAI-KNOT, and conclude with an explanation of the featured implementation.

## 3.1 Design Goals

When designing the WAI-KNOT platform, I combined a number of design goals for the entire system. The three most important considerations were latency, quality of sound, and scalability.

In creating a platform capable of real time interactivity, the most important design goal involved the problem of latency. Latency in electric or electronic system design may be described as follows. When pressing a key on a piano, there is a very small delay between the finger strike and the sound. This delay is so short that it may be called immediate. But with a pipe organ, the delay is much larger. When pressing an organ key, a relay (which may involve a pneumatic switch) sends a signal to a pipe to open a gate to allow air to flow through the pipe. Only after some air has passed through the pipe, does the organ begin to create sound. This delay differs for every organ and must be managed by the organist. The WAI-KNOT platform is similar in that there is a delay between input and output audio of the handheld device. If this delay is too high, users must learn to compensate. To relieve this user burden, the first design goal for the

WAI-KNOT platform involved the difficult problem of minimizing latency.

I also designed the handheld devices for small size, high quality sound recording and playback, and high bandwidth wireless connectivity to the server. I wanted to make the clients as small as reasonably possible without sacrificing their capabilities. Since the WAI-KNOT platform utilizes wireless connectivity, I considered that the clients should provide as much mobility and portability as possible.

The next design goal for the Knots involved sound recording and playback. WAI-KNOT clients have to be capable of real-time full duplex sound recording and playback. Since the project was intended for interactive musical applications, it was important that WAI-KNOT facilitate high quality audio. Thus, to perform CD quality encoding and decoding, the WAI-KNOT client must be capable of 44.1KHz sampling with at least 16bit(s) resolution.

Not only must the handhelds devices record and playback excellent audio, they must be capable of simultaneously wirelessly transmitting the data to the server. Assuming no compression, sending and receiving audio at 44.1KHz with 16bit(s) resolution requires a minimum bandwidth of 1.4MB (44.1KHz * 16 (number of bits) * 2 (full duplex)). Thus the server requires a bandwidth of 14MB to accommodate up to 10 clients.

There is also an interesting question of where to do processing: on the server or the clients. Later, I will describe the first WAI-KNOT application in which all of the processing could have been done locally on the client. However, many of the applications envisioned for the platform do involve interactions among multiple clients.

Thus the system server was employed to provide the bulk of the communication and processing of the audio data incoming from and outgoing to the clients. The server was designed to provide a flexible programming environment for building WAI-KNOT applications, and was also optimized for speed.

# 4 Building the WAI-KNOT Platform

After reviewing the WAI-KNOT platform system goals, I began to build the platform. I first had to choose the appropriate hardware and software.

## 4.1 Choosing the Hardware

The first hardware challenge involved choosing the appropriate wireless network system. I immediately chose to work with RF communication because of the high bandwidth requirements of my system. I then had the choice of one of three options; namely build my own, Bluetooth, or 802.11. Since this thesis is to prove the feasibility of wireless communication of high quality audio to and from a server, I decided it made the most sence to use currently available hardware. Custom hardware has the disadvantage of not scaling very well.

I decided to use 802.11 for the WAI-KNOT platform. An 802.11 network has a number of advantages: thoroughly tested hardware, wide availability, and wide adoption. Currently there are a growing number of 802.11 pc cards in use. There are at least six manufacturers building 802.11 hardware. Also 802.11 offers a bandwidth of 11 Megabit(s) which should theoretically connect up to 7 clients to one hub.

Subsequently, I began testing for latency in the 802.11 system. Since 802.11 was optimized for data transfer, there was no guarantee that it would provide an acceptable data rate and latency. I created a program to test the time required to send a packet from my desktop computer to my laptop using an 802.11 PC card. I ran the test with an Apple airport hub connecting my desktop via a 10baseT Ethernet connection and my laptop with an 802.11b PC Card.

I also considered using Bluetooth for my wireless system. Bluetooth has the advantage of small size, low power, and easy connectivity. I was fortunate to receive a couple of early development boards from DigiAnswer in the fall of 2000. At the time, no commercial Bluetooth hardware was available. I built two projects demonstrating the idea of remote processing. Unfortunately the development cards provided only limited support for multiple connections. I finally decided not to use Bluetooth for one reason. Bluetooth now offers a maximum bandwidth of 1 Megabit. To send CD quality audio, my system needed more bandwidth than Bluetooth now provides. However, I believe as Bluetooth matures, it could become an interesting addition to the WAI-KNOT platform.

I next tested for latency using various packet sizes. I built a test program measuring the time to send various sized packets from the iPaq to the MAC server and back. At first, I used the Media Lab wireless network; however, the network traffic greatly affected my results. At this point I decided to use a closed wireless network so I could control the amount of network traffic.

Mono CD quality audio requires a data rate (also commonly referred to as bandwidth) of 750 Kbyte(s)/second (44.1 Kbyte(s) * 16 bit), or 88Kbyte(s)/s. I first discoverd that TCP packets resulted in latencies 10 times

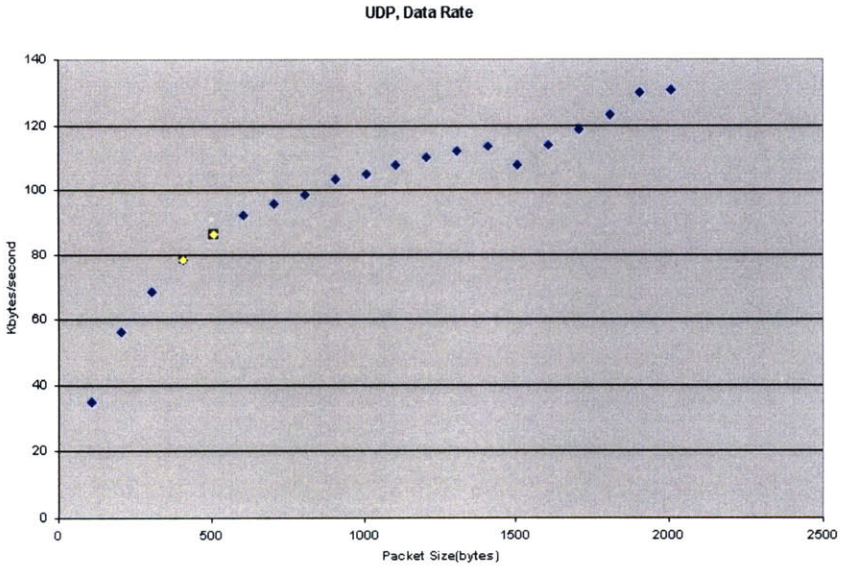
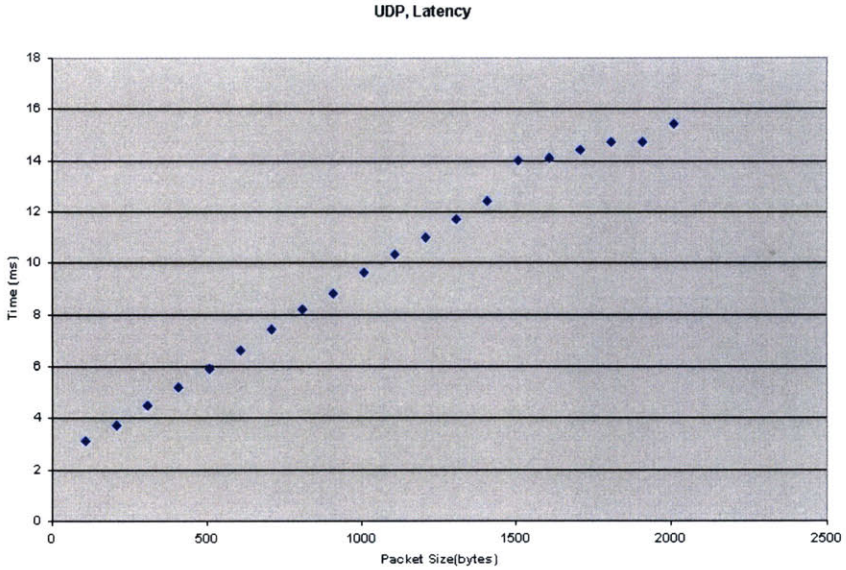



**Figure 3:**

*Resulting data rate from sending various sized packets over an 802.11 network using UDP*

**Figure 4:**

*Resulting latency from sending various sized packets over an 802.11 network using UDP*




those of UDP over an 802.11 network. For a description of TCP and UDP, see glossary. Using UDP, the minimum packet size for the required bandwidth is 600 bytes with a corresponding latency of 7ms (See Figures 4 and 5). I measured a standard deviation of 1.5 ms. These tests demonstrated that an 802.11 closed network would provide an acceptable solution for the WAI-KNOT platform.

Next I considered which handheld devices could best interface with 802.11. Currently the smallest computer capable of running an 802.11b network card is a Compaq iPaq. The iPaq has a 200MHz arm processor with sound card capable of full duplex 44.1KHz 16bit audio. Though the iPaq comes with Windows CE installed, I discovered that Windows CE doesn't support full duplex audio. Therefore, I installed Linux on the iPaq. More information on running Linux on a Pocket PC can be found at www.handhelds.org.

**Figure 5:**

*Compaq iPaq with Lucent 802.11 card*

For the server component I ultimately decided on using a MAC G4. I decided on a MAC server because of the large number of 3rd party audio programs available. One program which I used for the WAI-KNOT server

software is only available on the MAC.

## 4.2 Building the Software

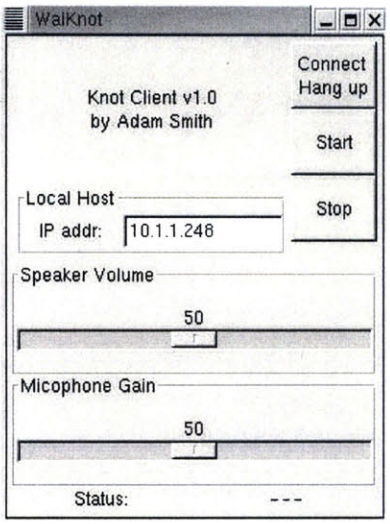
After choosing the hardware, I began building the software for the WAI-KNOT platform. This involved designing both the client program for the iPaq and the Mac Server program.

Building the client software to run on the iPaq posed another challenge of minimizing latency. Additionally, writing software for Linux running on a pocket PC presented other challenges. Currently, no native compilers are available for the iPaq without installing a micro drive. To build native applications for the iPaq, one must either cross compile on a Linux machine or compile on a skiffcluster. Skiffclusters are servers which run on Arm processors. I chose to use the skiffclusters, hosted by Compaq Research, as a number of researchers have found cross compiling to be unreliable. Thus to create the application, I used emacs to write the code; I then sent the code to the skiffclusters and compiled. Finally, I copied the compiled program to the iPaq.

Once the capability of building programs was established, I then began testing the iPaq sound card. Sound cards contain software buffers that offer stability by letting the processor spend less time monitoring the sound card. These buffers, however, also create latency. Linux provides access to altering the size of the buffers, but only at some risk. Following experimentation, I found the smallest operational buffer size.

I then began experimenting to find the optimal packet size. Since 802.11 is a packet-switch network, in order to send data, one has to break the data into packets. I knew the minimum packet size for the required 88Kbyte(s) using UDP, namely 600 bytes. Later I will describe the choice of using the program MAX/MSP. I found that scheduling in MAX/MSP required a larger packet size. The trade off is latency versus stability. After experimenting with various sizes, I settled on 1024.

**Figure 6:** *Screen shot of Knot client program*



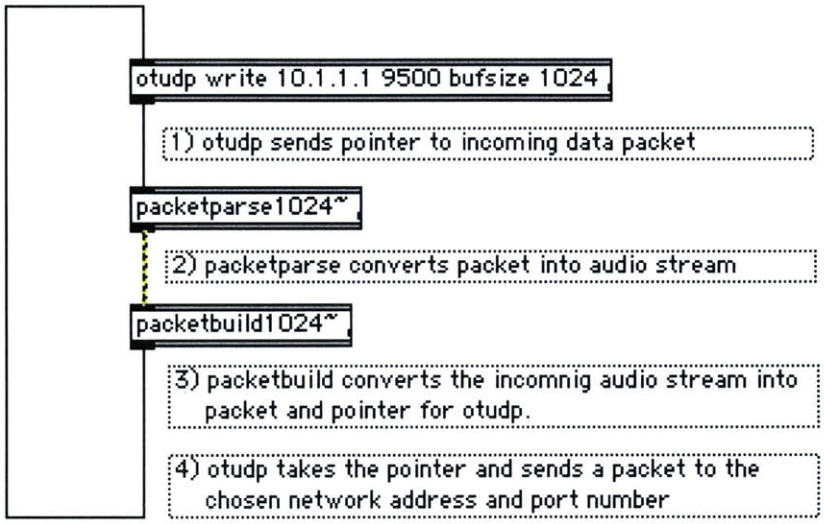

Next, I created a graphical user interface which allows the user to connect to a remote computer over a UDP socket using 1024 byte packets. See figure 6. After connecting, the user then presses the start button. Once the start button is pressed, my program records audio packets from the internal microphone and sends them to the remote computer. The program also listens for audio packets from the remote computer. When receiving an audio packet, it sends the audio data to the sound card and out through the internal speaker.

In designing the server program, I decided to use the audio program MAX/MSP. Initially MAX was written by Miller Puckette to provide an environment for creating MIDI applications. Puckette later built another program called Pure Data for real time musical and multimedia performances. David Zicarelli used the engine behind Pure Data and created MSP to work with MAX. [Puckette] MAX/MSP provides a suitable environment for creating audio applications. The program works with predefined objects which users can connect. For example, one may use a noise object to create an audio stream of white noise. A low pass filter may be added so that only the low waves will pass.

I decided to use MAX/MSP as the foundation for the WAI-KNOT server program. MAX/MSP, however, does not contain any objects for sending and receiving audio packets from a remote computer. Mike White, a graduate student at the University of California at Berkley built an object that could send and receive MIDI messages. I connected to his objects and wrote two new objects that could send and receive audio data. White's object, otudp monitors the chosen network address and port number for incoming packets. Upon receiving a packet, otudp sends a pointer to my packetparse object. Packetparse copies the data from the pointer and then converts the data into a MAX/MSP audio stream. My other object, Packetbuild, converts

**Figure 7:**

*WAI-KNOT server program*




the incoming MAX/MSP audio stream and passes a pointer to otudp. Otudp uses this pointer to create a packet and then sends packet to the chosen network address and port. See figure 7.

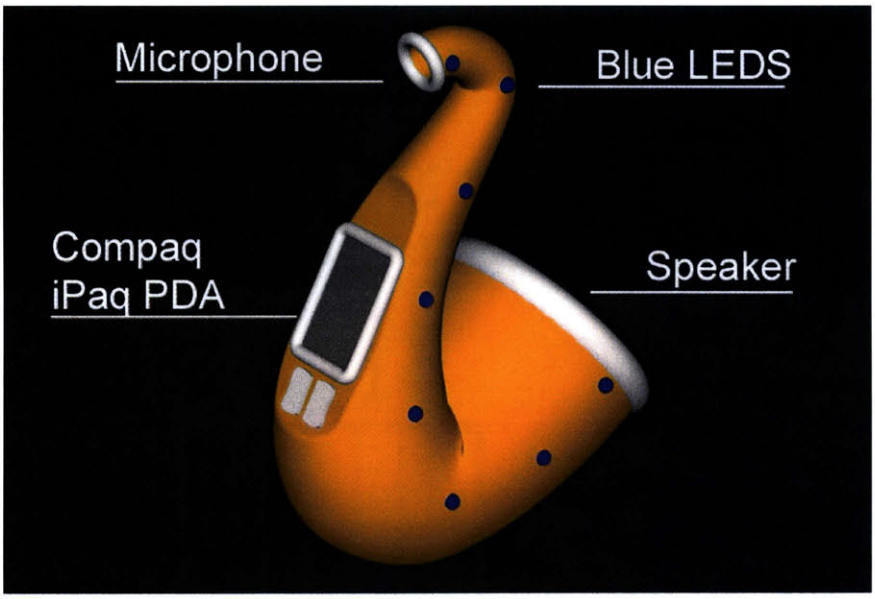
Thus, one could use the predefined MAX/MSP objects and my network audio stream objects to create interactive music applications.

# 5 WAI-KNOT Application

The original application that I have designed for the WAI-KNOT system is the sound transformer application (Sound Transformer). The Sound Transformer is a saxophone-like device which uses voice input to create instrumental music. See figure 8.

**Figure 8:**

*Concept sketch of Sound Transformer*




When the user sings into the device, the audio is wirelessly transmitted to a remote server. The audio content is then analyzed for pitch, volume, brightness, and spectrum. The analytical results are then fed into a software synthesizer to create an audio stream. This transformed audio is then returned to the device, amplified, and played out loud.

## 5.1 Sound Transformer

The Sound Transformer was designed for use in the Toy Symphony project.

The Toy Symphony project seeks to unite children with experienced orchestra musicians for the purpose of jointly creating and performing music. New toy instruments, including the Sound Transformer, have been and are now being designed with the use of available technology to help bridge the gap between a professional musician and an enthusiastic child.

The Sound Transformer contains two main components, the analysis and audio synthesis. In the analysis the incoming audio from the client is analyzed for pitch, volume, brightness, etc. In the synthesis, the results from the analysis are used to create audio that can then be sent back to the clients.

### 5.1.1 Sound Transformer Design Goals

The primary design goal for the Sound Transformer concerns the use and transformation of exclusively vocal input. The Toy Symphony project involves the participation of children (ages four to twelve years) in the creation of music. Young children (as well as most adults) have more control over their voices than their hands when used for making music.

### 5.1.2 Analysis Tools

I will first describe the tools used for the analysis. Although the MAX/MSP provides some predefined analytical patches. I utilized six independent patches which were developed by Tristan Jehan [Jehan]. Jehan developed these patches to analyze the incoming audio originating on the strings of an electric violin. Jehan's tools are optimized for speed, making them well suited for my time sensitive application. I first experimented with Jehan's six analytical tools. (Pitch 1.0; Loudness 1.1; Brightness 1.1; Noisiness 1.0; Bark 1.1; and Analyzer 1.10) The six analytical tools are further described as follows:

- Pitch 1.0: uses a Fast Fourier Transform to extract pitch. The output comes in two varieties, frequency and corresponding Midi note with

semi-tones. These Midi notes ranged from 45 to 85 when using vocal input. The output is also easily represented in an active visualizer.

- Loudness 1.1 measures the volume by taking psycho acoustics into account. For example when two notes are played temporarily close, one hears the first note as louder. The output can also be represented in an active visualizer.

- Brightness 1.1 measures the audio's center of gravity. Incoming audio with many high frequencies results in high brightness, while audio with more low frequencies will return low brightness.

- Noisiness 1.1 measures the number of frequencies relative to their intensity. An incoming noisy audio stream will contain many different frequencies with relatively similar intensities. A non-noisy incoming audio stream would have one intense single frequency.

- Bark 1.1 measures the relative intensity of 10 frequency ranges. Bark provides a discrete look at the frequency domain of the incoming audio stream.

- Analyzer 1.1 combines all of the above patches into one cohesive patch providing individual output for each aspect

Using an external microphone input in MAX/MSP, I began experimenting with the Analyzer 1.1. After performing a series of audio tests, I found the operating range of each parameter, such as pitch, loudness, and brightness. One should bear in mind that this range differs from one microphone to the next due to sensitivity and general hardware specifications, and must be adjusted for individual input configurations.

### 5.1.3 Audio Synthesis

To return the signal to the client output speaker and complete the circle, the output from the audio analyzer must be processed into an appropriate MIDI

signal via a mathematical processing patch. This MIDI signal must then be routed into a separate third-party patch (ReWire), specifically built for the synthesizer application. ReWire provides the ability to send MIDI messages from MAX/MSP to a software synthesizer program and then have the resulting audio returned to MAX/MSP.

I chose to use the Propellerheads Software synthesizer "Reason" for its versatility, power, and ease of MIDI mapping as well as its compatibility with ReWire. Experimenting with the synthesizer, it was evident that a system must be established to control the analyzed audio data and provide output within a desired range. I now had to define the parameters for that desired range.

### 5.1.4 Patching Analysis to Synthesis

I experimented with a number of mappings of audio analysis to synthesis. I decided to use the analyzed pitch to drive a "virtual" analog synthesizer, known as a Subtractor in Reason. I also used the semi-tone output to drive the pitch bend to prevent MIDI quantinization. I used the analyszd loudness to control the volume level on the Subtractor. The result is an application that matches the pitch and volume of the input audio.

I also experiment with a number of other mappings. I created a drum track that would begin when the user began singing at a certain volume. I found the brightness variable when combined with loudness to make a great mode change. Thus when the user wants to change mode, the loudly hisses causing the loudness and brightness to trip a change.

# 6 Evaluation

Completion of the project constituting the basis of this thesis has included building the WAI-KNOT platform and the Sound Transformer application. This evaluation includes answers to some original questions contained in the thesis proposal to build the WAI-KNOT platform for the purpose of investigating the potential of transporting and transforming high quality sound and music among wireless devices in real time. My research program involved designing and building a prototype interactive audio and musical application for handheld wireless devices which utilizes the WAI-KNOT remote processing platform.

These original questions were: What is the maximum latency acceptable for real time performance? How does the WAI-KNOT system match to these criteria? Is the latency inherent in digital wireless systems manageable for real time vocal feedback?

To answer these questions, I first offer a general discussion of latency in electronic audio systems. Next, I will describe the latency in the WAI-KNOT platform and the Sound Transformer application. This will be followed with a qualitative description of the manner in which the WAI-KNOT platform provides an open system with a manageable amount of latency. Finally, I will evaluate the Sound Transformer as a musical instrument in general as well as one for the Toy Symphony.

## 6.1 Latency in Electronic Audio Systems

Latency exists in all electronic audio systems for several reasons. First, physical limits govern transmission of audio signals through various media. For example, light, electromagnetic waves and electricity travel at 1x10^9ft/s; and sound travels at 1116 ft/s. Second, there exist practical limits on the

speed of processing a signal. Third, latency is accumulative. Measurement of latency involves the addition of all signal transmission times and all processing delays.

Consider the amount of latency which is compatible with system objectives. For sound, 30 ms is the transition to conscious perception. For example, a person when hearing two sounds separated by less than 30 ms, hears the two sounds as one long sound. The telephone industry aims to keep latencies on telephone conversations below 100 ms [Cheshire96] because tests suggest that conversations with delays over 100 ms lead to break downs in the normal unconscious etiquette of human conversation [Cheshire 96]. The effect is most apparent in telephone calls over satellite connections with 250 ms delays. One of the goals of this thesis is to quantify and understand the maximum amount of latency tolerable for WAI-KNOT applications.

## 6.2 Latency in WAI-KNOT Platform

The primary challenge in building the WAI-KNOT platform involved building a capable system for sending high quality audio to and from wireless handheld devices or nodes with a manageable amount of latency. In this section I will describe the sources of latency, how I measured each source, and then conclude with a summary of all latencies in the WAI-KNOT platform.

### 6.2.1 Sources of Latency in WAI-KNOT Platform

The following time delays cause latency, including imposed latency in the WAI-KNOT platform:

(A) Sound traveling from the user to the microphone of the handheld device.

(B) Conversion of the sound into a digital signal and storage in memory as an audio packet.

(C) Transmission of the audio packet to the server.

(D) Converting audio packet into MAX/MSP audio stream.

(E) Converting MAX/MSP audio stream in audio packet.

(F) Transmitting the audio packet to handheld device and storage in memory.

(G) Conversion of digital into analog audio and transmission to speaker.

(H) Transmission of audio from speaker back to the user.

The WAI-KNOT system was designed and built in an effort to minimize, as much as possible, the foregoing time delays and provide compatible latency.

### 6.2.2 Measuring Latency Sources in WAI-KNOT Platform

Evaluation and measurement of the WAI-KNOT platform latency proved challenging. Measurement and computation of the latency was accomplished as follows:

I assumed a user who is singing, or otherwise inputting audio, from a position approximately 4 inches from the microphone and about two feet from the speaker. Taking into account the speed of sound (1116 ft/s), the latency for (A) is found to be 1ms (1 ft * (1/1116 s/ft) and for (H) 2ms, using the same formula.

Next I measured the latency in (B). The Compaq iPaq, similar to other computers, contains an audio processing sound card. The latency in (B) includes the conversion of analog to digital, the speed and buffer size of the sound card, and the driver and hardware speed of copying the buffered signal of the soundcard into memory.

I built a program which reads from the microphone and immediately records the data on the speaker to measure this delay. I connected two o-scope leads to the iPaq, one to the input microphone and the other to the speaker. Next, I connected a function generator to an external speaker and positioned the speaker above the iPaq microphone. The function generator was employed to send a square wave every second into the iPaq

**Figure 9:**

*Using o-scope to determine latency by measuring the delay between an input square wave and the resulting audio*


microphone. The o-scope was then used to measure the shift from the incoming square at the microphone to the outgoing square leaving the iPaq. Ssee figure 9.

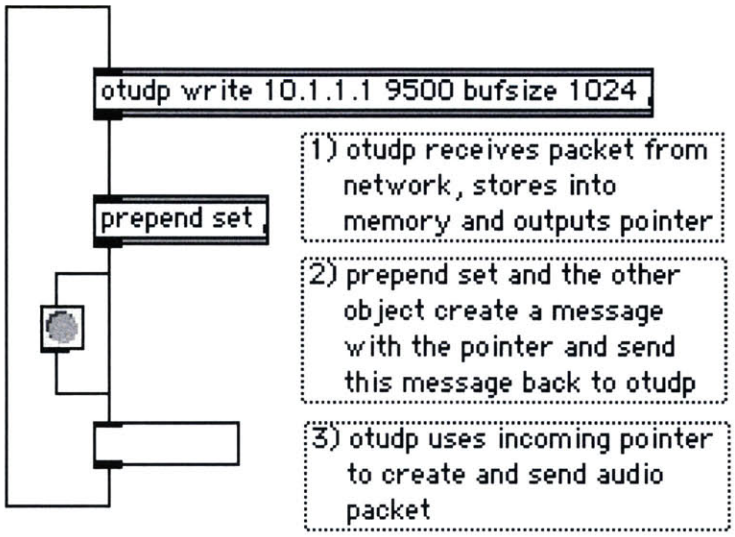
The latency from both (B) and (G) was measured in the above manner. The total latency was 34 ms. The latency in (B) and (G) are each 17ms, assuming the two are equivalent.

The above described (C) and (F) involve the latency incurred in sending and receiving a packet between the iPaq and the MAC server. For this test, I used a MAC G4 titanium laptop, an Apple Airport hub, and the iPaq client. I first created a set of test programs, one for the iPaq and the other for the MAC server. The iPaq program started a timer then sent a packet to the MAC server. The server upon recieving the packet, would send the packet back to the iPaq. Upon receiving the packet from the server, the iPaq program would stop the timer. Thus using a packet size of 1024, I measured a delay of 8 ms.

The foregoing accounts for the time of wireless transmission. It does not, however, consider the time consumed during the iPaq program communication with the network drives, and the time required for the MAX/MSP to

**Figure 10:**

*WAI-KNOT server program for latency testing*




interact with the network driver. To test these delays, I created two more programs, one for the iPaq and the other for the MAC server. The iPaq program sends input from the microphone to the server. It simultaneously checks for audio packets from the server. This program directs all incoming audio packets to the speaker.

MAX/MSP was used to create the MAC server program. This program checks for incoming packets. The arriving packet is copied into memory and returned to the iPaq. See Figure 10, for MAX/MSP server program. Again using the function generator and o-scope, I sent a test burst square wave and recorded the time between the input burst into the microphone and the iPaq speaker output burst. This test revealed a delay of 48 ms. The delay in (C) and (F) is therefore 14 ms (48 -34ms). The (C) and (F) delays are 7ms each, assuming they are symmetrical.

The latency in (D) involves the time required to convert an audio packet into a MAX/MSP audio stream, while the latency of (E) consists of the delay in converting a MAX/MSP audio stream into an audio packet. These delays are best understood by considering the MAX/MSP audio streams. MAX/MSP allows designers to connect objects with both MIDI messages and audio streams. Objects connected with audio streams process a certain number of samples which are sent by each object to the next one. The number of samples is referred to as the vector size, which may range from 64

to 2048 samples. Smaller vector sizes incur higher processor overhead but produce the beneficial result of lower latency.

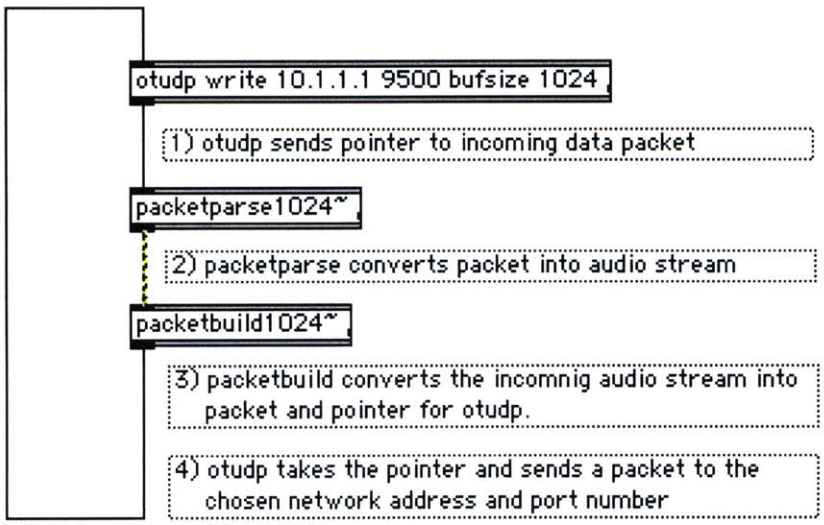
Consider the vector size as providing the buffer for each object. Minimal buffers produce less delay. A vector size of 64 with 44.1 KHz sampling was selected and used. With this vector size, each object must process 64 samples every 1.5 ms.

The MAX/MSP expects 64 samples every 1.5 ms as the audio packets arrive from the iPaq. Unfortunately, the audio packets arrive at the MAC server with variable delays between packets. Although technically, this delay should be rather small, I measured the MAX/MSP delays to ranges from 1 ms to 4 ms. These delays were probably caused by the MAX/MSP scheduling, the communication between the MAX/MSP and the MAC operating system, or the exchange between the operating system and the network card. The problem was corrected by inserting a buffer with a latency of approximately 6 ms.

To test total system latency, I created a program in MAX/MSP similar to the one used earlier. The new test program, however, included latency involved in converting the incoming packets with MAX/MSP audio streams and reconverting them into audio packets. See Figure 11 for test MAX/MSP program.

**Figure 11:**

*WAI-KNOT server program for testing latency with audio streaming*



Utilizing this test, a delay of 57 ms was measured with a worst case of 100 ms. The conversion delay, therefore, from audio packets in MAX/MSP, both to and from audio stream, results in a delay of 9 ms (57 ms- 48 ms). Since the audio conversion has a built in 6 ms buffer, I estimate the delay in (D) to be 6ms and in (E) to be 3ms.

### 6.2.3 Summary of Latency in WAI-Knot Platform

The WAI-KNOT sound transformer application was evaluated to possess segment and total latency as follows

(A) 1 ms

(B) 17 ms

(C) 7 ms

(D) 6 ms

(E) 3 ms

(F) 7 ms

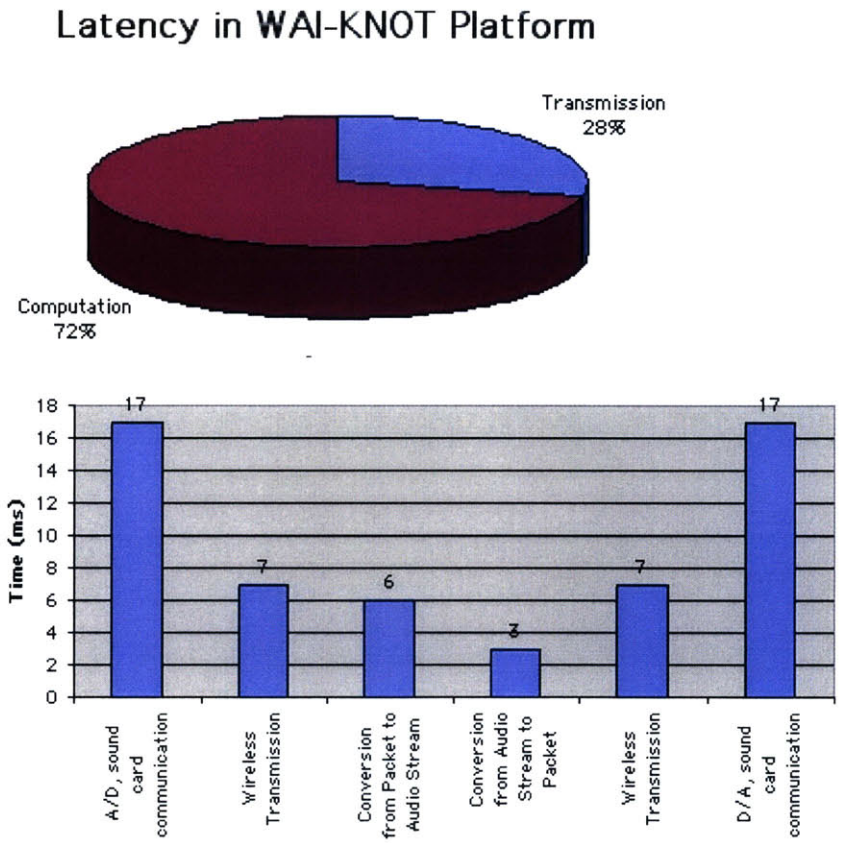
(G) 17 ms

(H) 2 ms

Total 60 ms

From figure 12, one can see the majority of the latency in the WAI-KNOT platform is in computation and not wireless transmission. The most sensitive variable leading to the latency is the packet size. Every delay is a function of the packet size. I observed a packet size of 1400 resulted in a 90ms latency, while a packet size of 1024 created a delay of 60ms.

Figure 12: *Latency in WAI-KNOT Platform*




## 6.3 Latency in Sound Transformer Application

The Sound Transformer application, like all audio feedback applications, also depends on manageable latency. This section will describe the sources of latencies in the Sound Transformer application, as well as how I measured the various sources. I will conclude this section with a summary of latency in the Sound Transformer application.

### 6.3.1 Sources of Latency in Sound Transformer Application

The Sound Transformer application presents the following causes of latency:

(A) Analyzing MAX/MSP audio system to extract pitch, loudness, brightness, noisiness and discrete picture of the frequency domain.

(B) Converting pitch, loudness, and brightness into MIDI messages.

(C) Sending MIDI messages to Rewire object.

(D) Transmission of MIDI-Messages from MAX/MSP to Reason.

(E) Converting MIDI-Messages into audio stream from Reason synthesizers.

(F) Transmission of audio stream from Reason back to MAX/MSP.

The foregoing causes of latency were measured to help evaluate performance of the Sound Transformer application.

### 6.3.2 Measuring Latency Sources in Sound Transformer Application

The Sound Transformer application begins with an analysis of the incoming audio. The WAI-KNOT platform supplies this audio in the form of a MAX/MSP audio stream. The latency involved in (A), the analysis segment, results from the number of samples used in the Fast Fournier Transform (FFT). The FFT requires a minimum number of samples for converting the incoming audio wave into the frequency domain. (The glossary contains additional information regarding an FFT).

The analysis object utilized in the Sound Transformer application requires 1024 samples before taking an FFT. This results for (A) in a delay of 23 ms (1024*1/44.1 KHz).

The delay in (B) is caused by the scheduler, governing the travel of MIDI messages in MAX/MSP. From my experience with the scdeduling in MAX/MSP, I estimate the delay in (B) to be approximately 2ms.

The delay in (C), (D), and (E) is only a fraction of a millisecond, according to the vendors of the software Rewire and Reason. The conversion

of audio from Reason to a MAX/MSP stream, however, includes a buffer of 512 samples, which consumes 12 ms. Therefore the latency value of (F) is about 12 ms.

I created another MAX/MAP program to test these values. An analysis object on the incoming audio from the iPaq was used to create MIDI messages. The program sends these MIDI messages to Reason via Rewire. In Reason, the MIDI messages activate a synthesizer which creates augmented audio that is returned to the MAX/MSP. A rewire MAX/MSP object receives this audio and then returns it to the iPaq.

Again by using the o-scope and function generator, I measured a latency of 120 ms. The delay added by the analysis and synthesis minus, the time for the WAI-KNOT platform, appears to be 60 ms. (120 ms-60ms). I had predicted the latency in (A), (B), (C), (D), (E) and (F) to be approximately 37 ms. It appears, therefore, that something is causing an additional 23 ms delay. I believe it is somewhere between (C), (D) and (E), caused by the processor load.

### 6.3.3 Summary of Latency in Sound Transformer Application

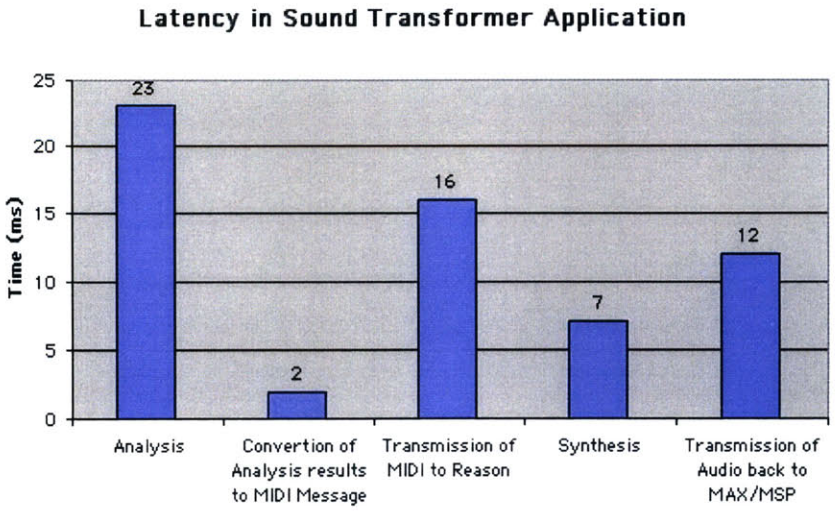



**Figure 13:**

*Latency in Sound Transformer Application*

The following is a summary of the Sound Transformer latency:

(A) 23 ms.

(B) 2 ms

(C) 7ms

(D) 9 ms

(E) 7 ms

(F) 12 ms

Total 60 ms

This total of 60ms represents the latency in the Sound Transformer by itself. The complete latency for the Sound Transformer application when combined with the WAI-KNOT platform is 120 ms.

This resulting latency of 120 ms is rather high. This high latency makes the Sound Transformer best suited for slow melodies. Users tend to slow down their rhythm as they sing into the device because of the delay. I discovered that adding reverb and delay helped to mask the latency.

## 6.4 Qualitative Evaluation of WAI-Knot Platform

The WAI-KNOT platform, originally proposed for this thesis, has been evaluated as meeting the basic design objectives and requirements. The Sound Transformer application demonstrates that the platform may be used to create a large number of new interactive musical applications. The relatively high latency of the present system may impair its use by musicians. WAI-KNOT does provide a workable platform for designing, testing and prototyping new applications. Additional research and development should be directed at further reducing latency. User testing is needed to help evaluate the many differnt mappings of audio analysis to synthesis. Development of a new custom version of MAX/MSP, based on network audio packets, would contribute, substantially in my judgment, to further reduction of latency in the WAI-KNOT.

## 6.5 Qualitative Evaluation of Sound Transformer

The Sound Transformer demonstrates the feasibility and potential for the future use of remote wireless devices for audio analysis and synthesis. These devices present a challenging opportunity for transporting, transforming, and enjoying high quality sound and music. Researchers, designers and developers should fully explore, conceive, and construct useful new applications which are now possibilities. Further development of broadband wireless technology will magnify the depth and range of promising new applications.

# 7 Conclusion

The Sound Transformer Application demonstrates the feasibility of deploying wireless architecture to create a new class of electronic musical instruments. The WAI-KNOT platform is the first step in designing and building new wireless interactive musical instruments. Additional research effort should be focused and conducted to further reduce the system latency now incurred from all sources, previously identified and described in both the WAI-KNOT platform and the Sound Transformer application. I believe further development time will effectively eliminate these problems and permit the satisfactory management of latency within the requirements of network channeling and processing.

Building the WAI-KNOT platform provided an array of both surprisingly easy and difficult problems. First, the challenge of full duplex audio streaming over an 802.11 network proved surprisingly easy. When using a closed network, transmission times were surprisingly consistent. Second, the 802.11 wireless network incurs a relatively small latency for small packets. As for difficulty, the beta Linux release contains a number of strange bugs. I found that the socket on the iPaq becomes unstable following five to ten minutes of operation. Next, MAX/MSP is rather slow at reading incoming network packets.

The successful management of latency, now possible and further augmented with future improvements, will support many applications which may be built by using the WAI-KNOT platform. These new applications include enhanced telephony, interactive music and collaborative music making. The WAI-KNOT platform, for example is well adapted for building telephone systems that offer CD quality audio communication. Creation of additional interactive music could be the subject of another interesting application

through the use of the WAI-KNOT platform. Another application, which could be built, would investigate musical collaboration.

The only limit to an exciting variety of new applications available through the use of broadband wireless technology is one's imagination. Home, office, and other network severs will provide instant computer and Internet access for wireless devices. These new servers will accommodate novel interaction and communication among electronic devices of all types and new computer and Internet programs. The WAI-KNOT platform and Sound Transformer application demonstrate the potential of diverting and transforming sound from an instrument over a wireless network, with the resultant possibility of producing a viable impact on real music making.

# 8 Glossary

| | |
|---|---|
| 3G | 3G is an ITU specification for the third generation (analog cellular was the first generation, digital PCS the second) of mobile communications technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationery or moving at pedestrian speed, 128 Kbps in a car, and 2 Mbps in fixed applications. 3G will work over wireless air interfaces such as GSM, TDMA, and CDMA. The new EDGE air interface has been developed specifically to meet the bandwidth needs of 3G. |
| 802.11 | Refers to a family of specifications developed by the IEEE for wireless LAN technology. |
| Analog | Almost everything in the world can be described or represented in one of two forms: *analog* or *digital*. The principal feature of analog representations is that they are continuous. In contrast, digital representations consist of values measured at discrete intervals. |
| Beta | A test for a computer product prior to commercial release. Beta testing is the last stage of testing, and normally involves sending the product to *beta test sites* outside the company for real-world exposure. Beta testing is often preceded by a round of testing called alpha testing. |
| Bluetooth | A set of specifications for wireless communication. Named for a 10th Century Danish King. A very short-range wireless standard. |
| Broadband | Form of data transmission where parallel channels pass through a cable. Commonly used to refer to systems with high data rate capabilities. |
| Buffer | A temporary storage area, usually in RAM. The purpose of most buffers is to act as a holding area, enabling the CPU to manipulate data before transferring it to a device. |
| Burst | A data transmission mode in which data is sent faster than normal. |
| CDDB | Compact Disk Data Base. |

| | |
|---|---|
| CDMA | Code Division Multiple Access Wireless Communications Protocol based on digital technology. Wireless Network. |
| Client/server | Computer model where one program (client) requests information from another (server). |
| Compile | A program that translates *source code* into *object code*. The compiler derives its name from the way it works, looking at the entire piece of source code and collecting and reorganizing the instructions. Thus, a compiler differs from an *interpreter*, which analyzes and executes each line of source code in succession, without looking at the entire program. The advantage of interpreters is that they can execute a program immediately. Compilers require some time before an executable program emerges. However, programs produced by compilers run much faster than the same programs executed by an interpreter. |
| Compression | Storing data in a format that requires less space than usual. Data compression is particularly useful in communications because it enables devices to transmit the same amount of data in fewer bits. There are a variety of data compression techniques, but only a few have been standardized. |
| Digital | Describes any system based on discontinuous data or events. Computers are digital machines because at their most basic level they can distinguish between just two values, 0 and 1, or off and on. There is no simple way to represent all the values in between, such as 0.25. All data that a computer processes must be encoded digitally, as a series of zeroes and ones. |
| Duplex | Refers to the transmission of data in two directions simultaneously. For example, a telephone is a full-duplex device because both parties can talk at once. In contrast, a walkie-talkie is a *half-duplex* device because only one party can transmit at a time. |
| FFT | Fast Fourier Transform. Once the time response of a circuit is obtained, the continuous spectrum of non-periodic signals can be calculated using Fourier analysis. The spectrum is displayed in a dual panel diagram either as amplitude and phase or as the amplitude of co sinusoidal and sinusoidal components. One can define the number of FFT sampling points, the start and end times of the sampling, and the maximum and minimum frequencies to be displayed. |

| | |
|---|---|
| GSM | Global System for Mobile Communications. Digital c`ellular standard which uses time division (TDMA) to carry numerous simultaneous calls on the same frequency. Wireless Network. |
| Karaoke | A device that plays instrumental accompaniments for a selection of songs to which the user sings along. |
| Kilobit | 1,024 bits for technical purposes, such as data storage. 1,000 bits for general purposes. Data transfer rates are measured in kilobits per second, abbreviated as *Kbps*, and count a kilo as 1,000 bits. |
| Latency | Delay in transmitting data between two devices—the time it takes to send a packet of information from one device to the other. The delay inherent in wireless systems. |
| Linux | Open source implementation of UNIX that runs on a number of hardware platforms. |
| Megabit | (1) When used to describe data storage, 1,024 kilobits. (2) When used to described data transfer rates, it refers to one million bits. Networks are often measured in megabits per second, abbreviated as *Mbps*. |
| MIDI | Musical Instrument Digital Interface. Pronounced *middy*, an acronym for ***M**usical **I**nstrument **D**igital **I**nterface*, a standard adopted by the electronic music industry for controlling devices, such as synthesizers and sound cards, that emit music. At minimum, a MIDI representation of a sound includes values for the note's pitch, length, and volume. It can also include additional characteristics, such as attack and delay time. |
| Minstrel | Device which connects PDA's to the cellular network. |
| Moore's Law | Established in 1975 by Gordon Moore describing the rapid increase in microprocessors. |
| Napster | Internet based system for distributing music in MP3 format. |
| NIC | Network Interface Card. |
| Node | A data entry point in a database management system. |
| Packet | A piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in |

| | |
|---|---|
| | addition to the data. In IP networks, packets are often called *datagrams* . |
| PAN | Personal Area Networks. |
| PDA | Personal Digital Assistant. Can function as cellular phone, organizer and fax sender. |
| Remote Processing | Refers to files, devices, and other resources not connected directly to the workstation. |
| Scalability | The ability of a technique or algorithm to work when applied to larger problems or data sets. |
| Skiffcluster | A server which runs on an Arm processor. |
| Streaming | A technique for transferring data which can be processed as a steady and continuous stream. Streaming technologies are becoming increasingly important with the growth of the Internet because most users do not have fast enough access to download large multimedia files quickly. With streaming, the client browser or plug-in can start displaying the data before the entire file has been transmitted. |
| TCP | Abbreviation of *Transmission Control Protocol,* and pronounced as separate letters. TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. |
| Telephony | The science of translating sound into electrical signals, transmitting them, and then converting them back to sound; that is, the science of telephones. The term is used frequently to refer to computer hardware and software that performs functions traditionally performed by telephone equipment. For example, telephony software can combine with the modem to turn a computer into a sophisticated answering service. Voice mail is a popular telephony application. |
| TDMA | Time Division Multiple Access. A wireless network which divides the frequency into time slots. Each frequency may carry multiple transmissions. Wireless network. |
| UDP | Short for User Datagram Protocol, a connectionless protocol that is used primarily for broadcasting messages over a network. |

| | |
|---|---|
| WAI-KNOT: | Wireless Audio Interactive Knot. |
| WAN | Wireless Area Network. |
| Vector | In computer programming, a one-dimensional array. A vector may also mean a pointer. |

# 9 References


[Berreby 96] Berreby, David. "Sound of Mind Tod Machover's Brain Opera." http://www.slate.com:80/Opera/96-08-13/Opera.asp

[Blackburn 98] Blackburn, S. and DeRoure, D. (1998) A tool for content based navigation of music. ACM Multimedia, Briston. **http://www.audio.ecs.soton.ac.uk.sbh**.

[Bluetooth 00] Boosting Product Development. Bluetooth Academy (2000) Seminar Workbook. Copenhagen, Denmark.

[Brain Opera] "Brain Opera Singing and Speaking Trees" **http://park.org/Events/BrainOpera/onsite/forest/page2.html**.

[Browning 97] Browning, Paul. "Audio Digital Signal Processing in Real Time" Masters Thesis MIT. 1997. **http://slate.msn.com/Opera/96-08-13/Opera.asp**.

[Chai 00] Chai, Wei. 2000. "Melody Retrieval On The Web" MS Thesis proposal, MIT Media Laboratory. **http://www.media.mit.edu/~chaiwei/papers/proposal.pdf**.

[Champness] Champness, Angela "The Path to High Speed Wireless Data Networking.

[Cheshire 96] Cheshire, Stuart. 1996. "Latency and the Quest for Interactivity." **www.stuartcheshire.org/papers/LtencyQuest.html**.

[Daniel] Daniel, S. Hypertext, Collaboration, and Interactive Art: "The Brain Opera." **http://arts.ucsc.edu/sdaniel/new/brain2.html**.

[Handhelds] **http://www.handhelds.org**.

[Jehan] Jehan, Tristan. 2001. "Perceptual Synthesis Engine: An Audio-Driven Timbre Generator" Masters Thesis, MIT Media Laboratory.

[Machover 92] Machover, Tod 1986-1992 "Classic Hyperinstruments." http://brainop.media.mit.edu/Archive/Hyperinstruments/classichyper.html

[Machover 99] Machover, Tod (1999) "Interview with Tod Machover", LA Times. Years End Calendar Section.

[Machover 94] Machover, Tod et al. (1994) Vox-Cubed, Performance at the MIT Media Laboratory.

[Puckette] **http://www-crca/ucsd.edu/~msp**

[Minsky 81] Minsky, Marvin. Music, Mind, and Meaning. Computer Music Journal. Fall 1981, Vol 5. Number 3.

[Oliver 97] Oliver, William 1997. "The Singing Tree: A Novel Interactive Musical Interface." Masters Thesis, MIT Media Laboratory.

[Resnick 98] Resnick, Michel. "Technologies for Lifetime Kindergarten". "Educational Technology Research & Development" Vol. 46. No 4. 1998.

[Rigopolous 94] Rigopolous, A. 1994. Growing Music from Seeds: Parametric Generation and Control of Seed-Based Music for Interactive Composition and Performance. MS Thesis MIT Media Laboratory.

[Stevenson 01] Stevenson, Reed 2001. "A wWeek On, DoCoMo's 3G Users Report Minor Glitches." http://www.xtra.co.nz/technology/0,,827-521966.00.html.

[Theremin] "Theremin". **http://www.thereminworld.com/learn.asp**

[Weinberg 99] Weinberg. Gil. "Expressive Digital Musical Instruments for Children. Masters Thesis MIT. September 1999.

[Witowsky 98] Witowsky, William E. 1998. IP Telephone Design andImplementation Issues. www.telogy.com/our_products/golden_gateway/pdf/IP_Telephone.pdf.