Tools and Techniques for Rhythmic Synchronization in Networked Musical Performance

Reid Kei Oda

a dissertation
presented to the faculty
of princeton university
in candidacy for the degree
of Doctor of Philosophy

recommended for acceptance
by the Department of
Computer Science

Advisers:
Rebecca Fiebrink
Adam Finkelstein

June 2017

Abstract

This thesis examines the Internet as a space for playing collaborative, rhythmically synchronized music among artists who are in different physical locations. The Internet has revolutionized many aspects of life, but has yet to become a frequently-used venue for music performance. We take the position that in order for this to occur, we must develop a new class of musical instruments that are specifically designed for the web. In particular, they should compensate for the delay that exists in networked communication. We explore two different approaches to this problem. The first is a method for tempo synchronization across large distances. The second is a method for reducing delay by predicting notes before they are played, sending the note information across the Internet, and scheduling the note to be synthesized into audio at the same time in multiple locations.

Our method for tempo synchronization, the Global Metronome, delivers timing error comparable to or less than the industry standard method. However, our method can synchronize tempo between devices located anywhere on earth, while the industry standard method requires devices to be connected via a physical cable. Our approach depends on synchronized clocks. Therefore this thesis presents an inexpensive, portable and accurate time server device to grant users the ability to use the Global Metronome. This device, called PIGMI, is a software suite that runs on a popular single-board computing platform. The software is published as a freely-available, open source software project.

This thesis also presents a musical instrument that uses prediction to reduce network latency, called MalLo. We present a series of studies that explore the sensor requirements, viability, and usability of the system. We also use MalLo in concert performance and study musicians as they learn to use the instrument over 3 rehearsals and the concert.

The primary contributions of this work include (1) a method to synchronize tempo between devices that are separated by great distance; (2) a small, inexpensive timeserver that facilitates this approach; (3) a better understanding of the advantages and disadvantages of tempo synchronization via this approach; (4) a method for latency reduction via per-note prediction including simulations, a usability study, and a study of live performance exploring its viability. This work enables new methods of remote musical collaboration and moves us towards realizing the dream of a global community of collaborating musicians.

Acknowledgments

I am deeply grateful to the many people who have helped me throughout my academic career. First and foremost I would like to thank my advisor, Rebecca Fiebrink, for her guidance throughout my graduate study at Princeton. Your expertise, broad vision, and skillful mentoring has been truly inspiring. There is no way to thank you enough for all of the time and support you've given to me.

I am also deeply indebted to Adam Finkelstein who provided much needed in-person support and guidance. You helped me to feel at home at Princeton. I learned a lot from you and I had fun doing it.

I would like to humbly thank Lynn Shostack and the Princeton Project X grant for choosing to fund this work. I am privileged to be one of the many "out-of-left-field" lines of research you have supported.

My favorite part of Princeton has been working with the faculty of Department of Computer Science. In particular I would like to thank Szymon Rusinkiewicz for helping to build the initial prototypes of the MalLo system. I would like to thank Thomas Funkhouser for thinking deeply about my work and giving honest, insightful feedback. I'd like to thank the graphics group for allowing us audio people to tag along. I am very impressed by the well-oiled research machine you have created, and I am grateful to have benefited from it. I would also like to thank Jennifer Rexford who, despite her multitude of obligations, always made time to answer my questions.

Thank you to my committee members. Marshini Chetty, your skillful advice on human-computer interaction topics have been a breath of fresh air. Chris Chafe, I could not have done this research without your foundational work on networked music performance. I feel fortunate that when I needed guidance in my thesis work I could go straight to the source. Dan Trueman, your seminar during my first semester at Princeton helped establish my belief that the relationship between music and technology can be symbiotic, but we have to put some effort into it. Now, six years later, I'm honored to have you on my committee.

I am extremely grateful for my time with Jeff Snyder spent exploring a vast range of topics. Through our conversations I gained a broader understanding of music and technology than I thought possible. I'd also like to thank Jeff for fostering a thriving computer music community at Princeton. You are the beating heart of the beast.

I would have been lost without our graduate coordinators, Melissa Lawson and Nicki Gotsis. Thank you for helping me to navigate the complex world of Princeton academics.

This work would not have been possible without the many volunteers who helped to test and develop the systems described in this thesis. I'd like to thank my fellow Internet music adventurers, Olivier Ozoux, Mike O'Connor, and Mason Malvini. Your eagerness made the experience fun, and your patience made it possible. To the musicians who performed in the Numerical Experience concert I send my heartfelt thanks. Chris Douthitt, Nikitas Tampakis, Joshua Becker, and Josh Collins, you brought your enthusiasm and made the event a success.

My Sound Lab colleagues made my experience at Princeton fun. To Katie Wolf, thank you for the many conversations and insights. Nearly everything that I produced in graduate school has your mark on it. I am proud to call you a friend and colleague. I genuinely could not have made it to this point without your help. I also could not have made it without Zeyu Jin. Your help and expertise made MalLo a reality. Your humor and zest for life made the Sound Lab a fun and silly place. I'm eager to see what you two do next.

I would like to thank my past mentors and colleagues. My time with you formed the basis for this thesis. Jim Hollan, Gert Lanckriet, Luke Barrington, Brian McFee, Ery Arias-Castro, Dr. Willie Brown; I am grateful for you. In particular, I'd like to thank Dr. Willie Brown for believing in me.

Finally I'd like to thank those that are closest and dearest to my heart. Thank you to my friends; you all are the best. Thanks to my family: Dad, Mom, Nalani, Kim. And of course thank you to Erin. Your loving support, conviction, and technical mastery helped me through many challenges to the finish line.

To my parents, Francis and Caroline Oda.
Thank you for your never-ending love and support.

Contents

Listing of figures

Why would anyone care to perform music between distant locations? If you are on the East Coast and the musician you want to perform with is on the West Coast then there is a reason. If it is possible to do such a thing then there is more reason. As the technology improves exponentially and ubiquitously then eventually there will be no reason not to perform music at a distance. Globalization gives us more reason. Making music together makes friends.

Pauline Oliveros

1 Introduction

1.1 Internet as a Performance Space

Just as the Internet has come to mediate so many aspects of our culture, it also offers huge potential as a shared musical performance space. It affords new opportunities such as discovering collaborators among a worldwide community of musicians, holding impromptu improvisation sessions, and playing with others who share similar artistic goals and vision.

However, the Internet has yet to become a popular venue for musical collaboration and performance. Part of the reason is that traditional music and traditional instruments are difficult to play over the Internet.

The Internet introduces a time lag between the moment a musician plays a note and when a collaborator in a distant location hears it. This is partly due to the routing architecture of the Internet, and partly due to the fact that network data can travel no faster than the speed of light. This lag makes it difficult for musicians to rhythmically synchronize with one another, which in turn makes it challenging to play any type of rhythmic music.

This thesis asks: How can we overcome the challenge of rhythmic synchronization in presence of network delay? Do we need a new class of musical instrument designed specifically for the Internet? What characteristics would this new type of instrument have? Are there genres of popular music that are well-suited for playing over Internet? Are there ways that we can get around the speed-of-light barrier? If so, what are the trade-offs? Beyond delay, what are other factors that get in the way of playing music over the Internet?

1.2 Goals and Contributions

The goals of this thesis are (1) to create new instruments and tools that enable rhythmically synchronized Internet music performance, (2) to deploy these tools to a community of users, (3) to evaluate their effectiveness via user studies and musical performance, and (4) to make recommendations for future research. In this work we present two projects, each of which represents a different approach to rhythmic synchronization.

The first project, the Global Metronome, is a method for synchronizing tempo between any number of devices, anywhere on earth, with no communication between them. The

Global Metronome project was motivated by the fact that computer system clocks are becoming increasingly easy to synchronize to a degree suitable for music timekeeping. Therefore, we asked the question: In this world of highly synchronized system clocks, how should we go about managing tempo? To simulate this world we create the PIGMI (Pi Global Metronome Implementation), which is an affordable timeserver that can be built for less than $100. It enables any device to use the Global Metronome approach. We show that the PIGMI and Global Metronome approach is capable of synchronizing devices that are thousands of miles apart with an accuracy that is on par with the industry standard, MIDI clock. The difference is that MIDI clock requires devices to be physically connected via a cable with a maximum length of 50 feet.

Before the PIGMI there was no reliable method for MIDI synchronization across long distances. Therefore, we deploy PIGMIs to participants located in various geographic locations and study the process of playing rhythmically synchronized sequencer-based music. We record detailed notes regarding setup and performance with the intention of gaining a deeper understanding of the barriers to playing live in such a manner. Finally, we distill these notes into a series of observations about the viability of this method and lay out a series of questions that remain unanswered, offering hypotheses about potential solutions.

The second project, MalLo, is a system that uses prediction to combat latency. It is a new type of instrument that predicts notes *before* they have occurred, so that information about the predicted note can be sent over a network, and the note can be synthesized at a receiver's location at approximately the *same moment* the note occurs at the sender's location. MalLo is inspired by percussion instruments and uses a striking gesture. This gesture was chosen for a number of reasons. First, striking gestures are almost universally understood by people. Second, the striking motion is surprisingly easy to predict. Third, despite its predictability, it is highly rhythmically accurate. Fourth, this simple gesture can be used a building block for complex, pitched instruments.

We evaluate MalLo through a series of studies. First we determine the sensor requirements for such a system. Next, we run a simulation of MalLo performance under varying degrees of latency by sending message between computers on the Internet. We perform a lab-based usability study. We use MalLo in rehearsal and in concert to study how performers might learn and play such a system. Finally, we perform a pilot study of MalLo as an instrument for use over the Internet.

We have previously published some of the work in this thesis. Specifically, we have described an early version of MalLo in Oda et al. (2013); described the MalLo end-to-end system in Jin et al. (2015); presented the Global Metronome in Oda & Fiebrink (2016).

1.3 Outline

In the next chapter, we provide an overview of networked music performance, a detailed explanation of the challenge of latency, and a background of the tools and techniques that we and others have used to combat it. We also describe the motivating factors around our proposed solutions.

In Chapter 3 we present the Global Metronome, a method for tempo synchronization between unconnected devices. We also describe the PIGMI, our inexpensive hardware device that uses GPS to enable high accuracy clock synchronization[*].

In Chapter 4 we deploy PIGMIs to three participants and investigate the process of us-

[*]Joint work with Zeyu Jin. See page 82 for a detailed description of the division of labor.

ing them to play rhythmically synchronized music. In this chapter the participants play partial roles as subjects, and partial roles as collaborators as we seek to determine the advantages, problems and challenges of this style of performance.

In Chapter 5 we present MalLo, an instrument that uses prediction to reduce network latency. We begin with an initial feasibility study to establish the hardware sensor requirements for such a system. Next we present and evaluate an end-to-end system that uses MalLo prediction to reduce latency and an overlay network (Andersen et al., 2001) to reduce latency spikes.

In Chapter 6 we run a lab-based usability study to determine how a variety of users respond to the MalLo system. Users perform a simple "drumming" task where they use the system to synchronize to a metronome beat. We collect timing data to test the rhythmic accuracy of the system. We also collect observation, interview and survey data to evaluate the usability of the system.

In Chapter 7, we deploy MalLo to performers in a concert situation. We record their usage over 3 rehearsals and a performance to study how their technique and accuracy changes over time. Finally, we run a preliminary test of MalLo use over the Internet to gauge its viability over longer distances.

In Chapter 8 we conclude with a review of our findings and contributions. We describe the next steps in research regarding these tools, and enumerate important open questions.

2
Background and Motivation

2.1 Summary

The goal of this work is to develop new tools and techniques for playing rhythmically synchronized music over the Internet. It is inspired by and draws from a large body of research in fields such as networking, networked musical performance, musical instrument design, and clock synchronization. In this chapter we present an overview of each of these topics and define key terms that will be used in subsequent chapters. First we discuss the Internet, its architecture, and its characteristics that make playing over it promising, but challenging. Next we discuss networked musical performance, its short history, and the greatest challenge for playing rhythmically synchronized music: latency. Next we describe clock synchronization and discuss the current state of the art. Finally, we discuss digital musical instrument design. We close the chapter with a summary of the key findings of this previous research, and describe our motivations for this current work. In this chapter we define terms that will be used in later chapters. Important terms that will be used later will be indicated in bold.

2.2 The Internet

In order to understand some of the challenges of playing rhythmically synchronized music via the Internet it is useful to understand what aspects of the Internet's architecture contribute to these challenges.

2.2.1 Data Transmission

The Internet is made up of a series of interconnected networks. It is robust and decentralized, meaning if some networks fail or are cut off, the individual parts can function on their own. When we send data (e.g., audio data) via the Internet we have little-to-no control over the physical path it takes while traversing these networks from its source to its destination (Peterson & Davie, 2007). This means that any time delays due to length of path or congestion are largely unavoidable.

Data is divided into small chunks and sent as packets, which have a destination address.

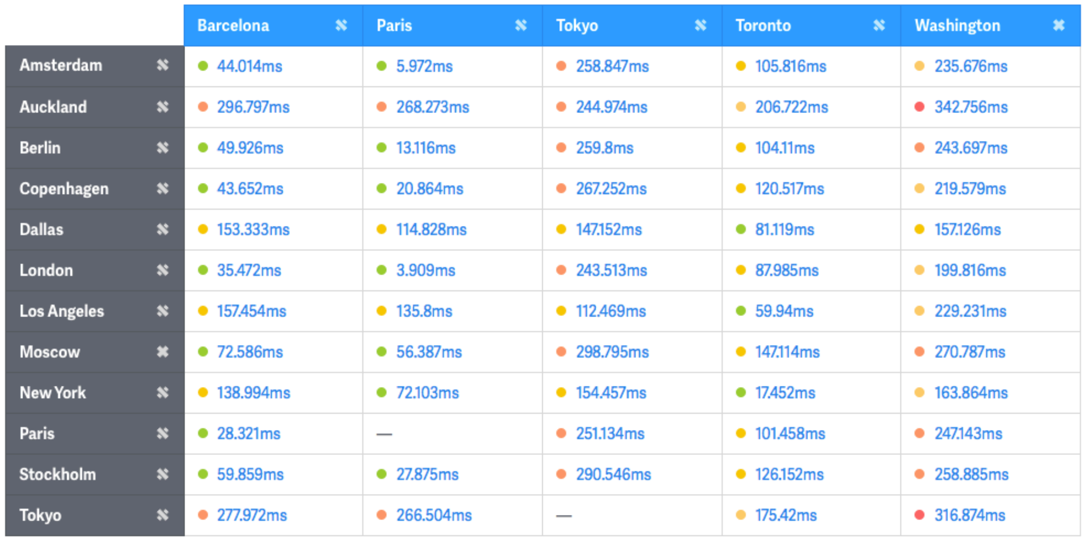
As the packets travel from source to destination they are sent from one router to another. Routers are specialized computers that sit at the borders of networks and within them and for each destination address, they forward the packet on to the next router that claims to be closer to the destination. The Internet is constantly changing due to computers joining and leaving networks, or problems with communication between computers, so routers are in constant communication with their neighbors, updating routing information. A packet's path is completely dependent on the routes that have been computed by the routers that it passes through. Usually this path is fairly efficient, but if changes in network conditions occur, such as congestion or breakdowns in communication routers can be slow to respond, causing periodic or sustained delays in packet delivery. For a more detailed description of how data is transmitted, see *Computer networks: A Systems Approach* by Peterson & Davie (2007).

2.2.2 Latency and Bandwidth

Network latency is the delay that exists in data transmission. It is the length of time it takes a single packet to get from its source to its destination. If we use the metaphor of a river, with the water atoms as packets, latency is how long it takes a single atom to flow from one location to the other; a low-latency connection is like a fast moving river. Latency and bandwidth are the two metrics by which we judge the data transmission performance of a network. Bandwidth can be thought of as the width of the river (i.e., if we slice across the river, it is the amount of water in that slice). Typically the speed of a network connection is advertised by its bandwidth and is measured in bits per second. The best performance comes from high bandwidth and low latency. Network bandwidth is increasing all the time while latency remains more or less the same year after year. In other words, the river is getting wider, but the water is moving at the same rate. What this means for musical applications is that as time goes on we will be able to send more data. This might take the form of higher quality audio, more streams (tracks) of audio, and more control information. However, as long as the basic Internet architecture remains unchanged, and the speed of light barrier remains unbreakable, there is a limit to how quickly that data will arrive at its destination.

Network latency depends on three main factors (Peterson & Davie, 2007). The first is the amount of time it takes for data to be "packetized", or put into data packets. The current crop of networked music tools are able to do this very quickly so that the time increase is negligible. The second factor is propagation latency, which is how fast a packet moves through the transmission medium, and is upper-bounded by the speed of light. Light (electromagnetic radiation) travels through a vacuum at 3.0×10^8 m/s, through copper at 2.3×10^8, and through optical fiber at 2.0×10^8 (Peterson & Davie, 2007). For longer distances, the propagation delay can be significant. This means that, generally, latency is correlated with distance; locations that are closer together generally experience less network latency. Figure 2.1 shows a recent measure of one-way latencies between well-known cities. Note that the latencies between Paris and Tokyo are not the same in each direction.

The third factor of network latency is queuing latency. This is the time that a packet spends waiting in a router queue before it is sent to the next destination. The amount of delay increases when there is heavy traffic through a router, and traffic can change rapidly and unexpectedly. Queuing delay can be an explanation for why sometimes short distances have greater latency.



	Barcelona	Paris	Tokyo	Toronto	Washington
Amsterdam	44.014ms	5.972ms	258.847ms	105.816ms	235.676ms
Auckland	296.797ms	268.273ms	244.974ms	206.722ms	342.756ms
Berlin	49.926ms	13.116ms	259.8ms	104.11ms	243.697ms
Copenhagen	43.652ms	20.864ms	267.252ms	120.517ms	219.579ms
Dallas	153.333ms	114.828ms	147.152ms	81.119ms	157.126ms
London	35.472ms	3.909ms	243.513ms	87.985ms	199.816ms
Los Angeles	157.454ms	135.8ms	112.469ms	59.94ms	229.231ms
Moscow	72.586ms	56.387ms	298.795ms	147.114ms	270.787ms
New York	138.994ms	72.103ms	154.457ms	17.452ms	163.864ms
Paris	28.321ms	—	251.134ms	101.458ms	247.143ms
Stockholm	59.859ms	27.875ms	290.546ms	126.152ms	258.885ms
Tokyo	277.972ms	266.504ms	—	175.42ms	316.874ms

Figure 2.1: The one-way latencies between cities (Wondernetwork, 2017).

2.3 Networked Musical Performance

Networked musical performance (NMP) is the practice of using communication networks to facilitate music collaboration between artists in remote physical locations. Networked performance began as early as the 1890s when telephone operators entertained themselves with impromptu musical jams over the phone lines (Marvin, 1990). John Cage's "Imaginary Landscape No. 4 for Twelve Radios" used radios that were interconnected. The practice began more popular in the 1990's when music pioneers such as The Hub (Brown, 2002) and Pauline Oliveros (Oliveros et al., 2009) began experimenting with using the Internet as a performance space. Since then researchers such as Chris Chafe (Chafe et al., 2010), Juan-Pablo Câceres (Cáceres & Chafe, 2010), Alexander Carôt (Carôt et al., 2009), Elaine Chew (Chew et al., 2005), Ajay Kapur (Kapur et al., 2005), Mihir Sarkar (Sarkar & Vercoe, 2007), and many others have contributed to a growing body of work on the subject.

2.3.1 Latency in Networked Musical Performance

Overcoming the latency in data transmission is one of the most difficult challenges when performing rhythmically synchronized networked music via a network. Latency causes a phenomenon called tempo drag (i.e. tempo deceleration). The delay causes players to slow down in order to stay in time with the audio they hear, which causes their collaborators to also slow down to stay in time with their delayed audio, which in turn causes the first player to slow down, etc (Cáceres & Chafe, 2010; Chew et al., 2005; Chafe et al., 2004). The players slow down and might reach an equilibrium at a slower tempo. In other cases the tempo remains erratic. At high levels of latency, it can be impossible for players to stay in time with one another.

Rottondi et al. (2016) explains that amount of tempo drag depends on a number of factors such as player skill, familiarity with the music, the instruments played, visual feedback, and rhythmic complexity. They also state that a generally accepted rule is that 30 ms or less of latency is invisible to the player. 30-60 ms is perceivable and causes some tempo deceleration. Above 60 ms it becomes difficult to play in time and at a constant speed. For an in-depth analysis of the NMP field, please read "An Overview on Networked Music Performance Technologies" by Rottondi et al. (2016).

Rottondi et al. list the sources of latency as:

1. in-air sound propagation from source to microphone;
2. transduction from the acoustic wave to electric signal in the microphone (negligible);
3. signal transmission through the microphone's connector (negligible);
4. analog to digital conversion (possibly with encoding) and internal data buffering of

the sender's sound card driver;

5. processing time of the sender's machine to packetize the audio data prior to transmission;
6. network propagation, transmission and routing delay; processing time of the receiver's PC to depacketize (and possibly decode) the received audio data;
7. application buffering delay;
8. driver buffering of the receiver's sound card and digital to analog conversion;
9. transmission of the signal through the headphone or audio monitor (loudspeaker) connector (negligible);
10. transduction from electric signal to acoustic wave in the headphones or loudspeaker (negligible);
11. for loudspeakers, in-air sound propagation from loudspeaker to ear.

In NMP and in other applications, such as clock synchronization, we often think of latency in terms of round trip time (RTT), which is the time it takes for audio to travel from one player to a second, and from the second back to the first. Often times this latency is nearly the same as two times the one-way latency, though at times the one-way latencies can be unequal. We discuss this further in Section 2.5.

2.3.2 Compensating for Latency, An Incomplete History

One of the earliest methods for dealing with network latency is to simply embrace it as part of the performance medium. Pauline Oliveros was a pioneer of NMP, and performed her first piece over the telephone with artists in Oakland, CA; Kingston, NY; New York, NY; Houston, TX; San Diego, CA; and Los Angeles, CA (1991). This performance purposefully used the latency—in the 100s of milliseconds—and the sonic characteristics of the telephone as features of the "telematic" piece (Oliveros et al., 2009).

When the Internet gained in popularity in the mid-1990s, bandwidth was limited, so to perform artists used MP3 compression (Brandenburg, 1999) in order to reduce the number of packets that needed to be transmitted. The encoding process, however, added a large amount of latency, which ballooned up to 8 seconds (Oliveros et al., 2009). The bandwidth of the Internet has grown since and can now transmit uncompressed audio without suffering from queuing delays. Although a sender has no control over the path that a packet takes on the Internet, we can control certain aspects of latency by minimizing the time it takes for audio to be packetized, and by using UDP as the packet format. UDP (User Datagram Protocol) is a network packet protocol that prioritizes speed over error correction (Postel, 1980). If a UDP packet is lost in transmission, no attempt is made to confirm receipt of the packet or resend the packet if it is lost.

The current state of the art for minimizing aspects of latency that are under the players' control is JackTrip (Cáceres & Chafe, 2010), developed by the SoundWire group at Stanford, led by Chris Chafe. JackTrip uses a non-blocking approach to packetizing and de-packetizing audio data. "Every time a buffer is available [...] the sender thread immediately sends it as a UDP packet" (Cáceres & Chafe, 2010). On the receiving side, as soon as a packet containing audio is received it is put into a buffer to be played. It uses UDP as its network packet protocol. In addition to providing extremely low latency, JackTrip provides *constant* latency, by buffering incoming audio. This eliminates latency jitter, thereby making the listening experience more akin to when players are co-located.

In order to reduce latency beyond that made possible by JackTrip, we could attempt to predict the notes that an artist is going to play before they are played. Recent work by Sarkar and Vercoe does just this, by creating a system what allows two tabla players to play together with zero latency (Sarkar & Vercoe, 2007; Sarkar et al., 2007). Their system, TablaNet, works by listening to each tabla player and classifying the pattern being played into one of a known set of canonical tabla patterns. This classification is sent to the remote system, and that system synthesizes a version of the classified pattern into audio. Note that with this system many of the nuances of the playing are lost. The receiver does not hear the exact notes that the sender is playing, but they hear the general intention.

Our work is heavily inspired by Sarkar and Vercoe's work in that we present an instrument that uses prediction to reduce network latency, but ours predicts each individual note instead of the pattern. The motivation for this is that we wanted to create an instrument that responds in a manner that is typical of traditional instruments, where one gesture results in one note. The intention is to allow for higher in-the-moment expressiveness by being able to vary intensity, timing, or other attributes of the note.

Another promising approach to rhythmically synchronized music on the web is to let computers manage the timing of notes. Sequencers are hardware or software systems that play back a series of notes at a specified time. In performance they are often used to layer loops of phrases on top of each other to build compositions. In presenting their system Netjack, Carôt et al. (2009) uses sequencers to illustrate the effectiveness of their network audio framework.

In our work we use sequencers to test the timing accuracy of our tempo synchronization device, the PIGMI. We also explore sequenced music as a paradigm for Internet musical performance via a study where we play sequenced music over the Internet with participants and study their reaction to the process.

2.4 Gestural Control of Sound Synthesis

The study of gesture as it relates to musical interfaces sits at the intersection of music and human-computer interaction. With the advent of audio synthesis methods we are able to decouple the physical interface from the sound generation method, and therefore can make arbitrary mappings between player inputs and the resulting audio. This opens up vast possibilities for expressive use of technology.

Musical interfaces—in particular, gestural control of audio synthesis—represent a very specific type of use of gesture. According to Wanderley (2001): "Gestural control of computer generated sound can be seen as a highly specialized branch of human-computer interaction (HCI) involving the simultaneous control of multiple parameters, timing, rhythm, and user training".

One approach to gestural control is to capitalize on existing expertise of the musician by sensing the gestures they already make, such as sensing the bell movement of a clarinet (Wanderley et al., 2005), or adding sensors to the keys of a flute (Palacio-Quintin, 2008). These augmented instruments do not add any more keys or switches, yet they add new control dimensions which require practice to master. Other instruments, such as the Radio Drum (Boie, 1989) use a familiar gesture, such as striking a drum, to control other musical processes such as synthesizers.

Our approach builds off these previous works in that we also use the motion of well-known gesture—hitting a drum surface—as the basis for our instrument. But it is different in that we capitalize the on the *predictability* of the gesture in order to predict the timing and intensity of the notes before they occur.

2.5 Clock Synchronization

Computer system clocks are becoming increasingly more accurately synchronized. In this thesis we look ahead to a world where all clocks are synchronized a high degree (less than 1 ms error) and explore the musical interactions that result. In Chapters 3 and 4 we use synchronized clocks to enable tempo synchronization over long distances. In chapters 5, 6, and 7 we use synchronized clocks to enable events to be scheduled to occur in multiple locations at the same time.

All clocks, including the system clock in computers, advance at slightly different rates, which means that as time advances they will become out of sync with one another. This is due to a number of factors including construction materials, imperfections in those materials, age, and temperature. In order to keep clocks in sync with one another we must periodically test and correct them.

2.5.1 Summary of Terms

There is some disagreement on how to define clock synchronization terminology, but we will use terms as they are defined in Section 10 of RFC 2330 (Paxson et al., 1998).

A clock's offset is "the difference between the time reported by the clock and the 'true' time as defined by UTC. If the clock reports a time T_c and the true time is T_t, then the clock's offset is $T_c - T_t$." (pg. 14)

A clock's skew "at a particular moment is the frequency difference (first derivative of its offset with respect to true time) between the clock and true time." (pg. 14)

Clock skew is not constant, especially under changing temperature conditions such as those created by computer processor heat waste. Drift is "the second derivative of the clock's offset with respect to true time" (pg. 14).

There are three aspects of clock synchronization (Laird, 2012) that can also be generalized to tempo synchronization. The first is frequency synchronization, which is getting clocks to run at the same rate. The second is phase synchronization which is getting frequency synchronized clocks to "tick" in unison. The final is time synchronization which is making them all agree on the same time of day.

This thesis explores tempo synchronization in Chapters 3 and 4, which is related to clock synchronization in that music and clocks can be conceptualized as an oscillator and a counter that counts the oscillations. Whereas a clock might count seconds (or in most cases, small fractions of a second) in music we count beats. The musical equivalent of frequency synchronization is synchronizing tempo beats per minute. The equivalent of phase synchronization is note or measure phase, (i.e., getting the tempo-synchronized scores to start playing notes or measures at the same time). The equivalent of time synchronization is to agree on the position within a score (i.e., the number of notes or measures that have been played so far).

2.5.2 Clock Synchronization Methods

There are two common approaches for computer system clock synchronization. The first is packet-based clock synchronization, which is used for most computers, including cellular phones. This approach uses computer networks (both wired and wireless) to send packets back and forth between clocks in order to keep them synchronized with one another.

The second is GPS-based time synchronization, which is used for a variety of purposes including synchronizing cell-tower clocks, servers for distributed computation, scientific instruments, etc. This method senses radio signal sent from orbiting satellites which contain ultra-high accuracy atomic clocks. The GPS clock synchronization signal is available to anyone who has a clear view of the sky. It has a theoretical accuracy of $\pm$10 nanoseconds (Allan & Weiss, 1980).

Of packet-based approaches, the most common are Network Time Protocol (NTP) (Mills, 1991) and Precision Time Protocol (PTP) (Eidson & Lee, 2002). Although there are differences between the two, both use a similar method to poll the time offset between the two clocks which is used as input to the synchronization algorithm. Specifically, both assume that the network latency is the same in each direction. Figure 2.2-A shows how the messaging process works in an ideal network situation. Clock 2 sends Clock 1 a message that contains time T_1. Clock 1 then responds with a messages that contains times T_1 and T_2. The response message is received by Clock 2 at time T_3. Clock 2 assumes that the timestamp T_2 was applied at the midpoint between T_1 and T_3 and computes the offset between the two clocks as $\mathit{offset} = T_2 - \frac{T_3 - T_1}{2}$.

In the case that latency is the same in each direction, this method can be arbitrarily accurate, no matter how large the latency. However, if the latency is asymmetric it introduces error, as seen in Figure 2.2-B. For large latencies, this can translate to large amounts of offset error. If asymmetry is accompanied by latency jitter, it can cause both skew and drift error.

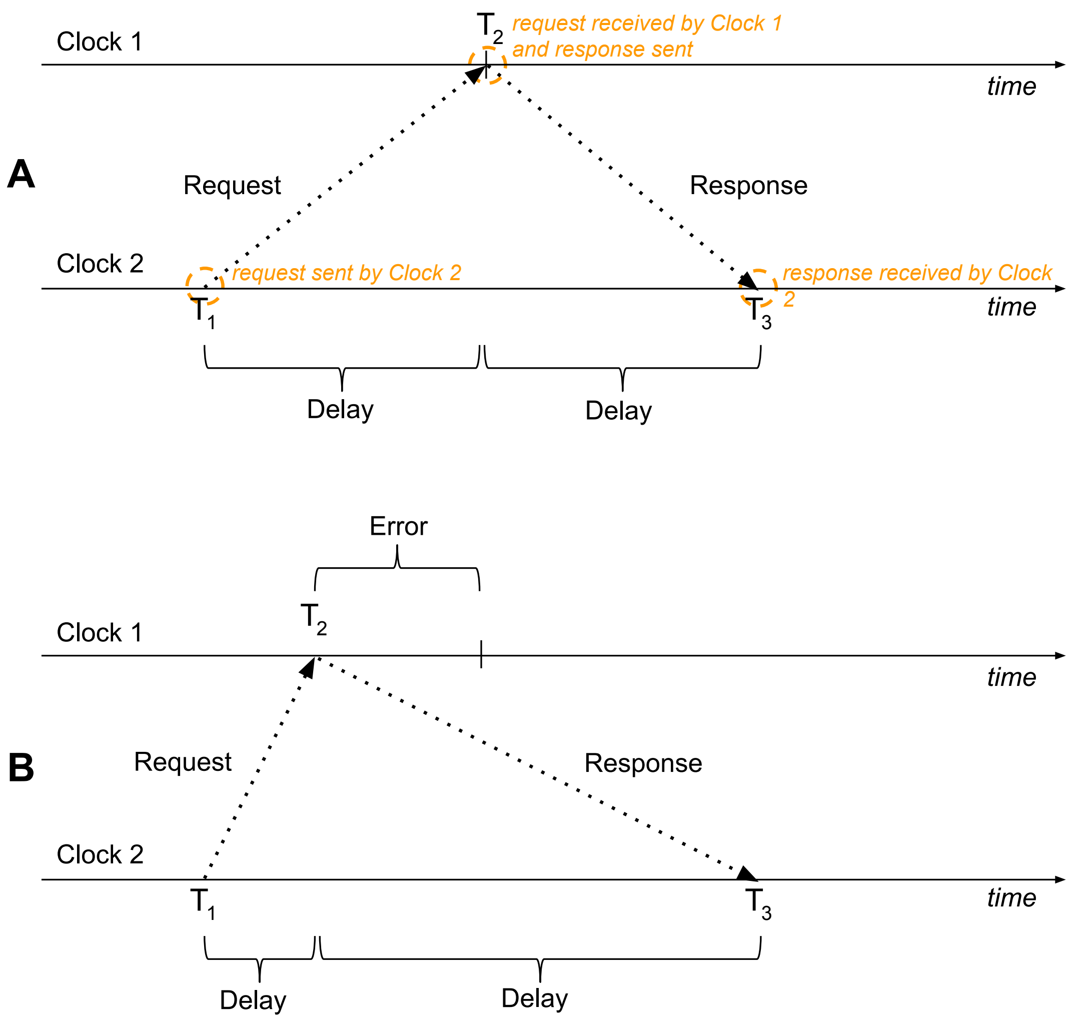


Figure 2.2: Packet-based clock synchronization methods make the assumption that network delay is the same in both directions. If this is not the case then error is introduced.

2.6 Conclusions

The Internet has had an unprecedented impact on our society, and enabled many new forms of data sharing and social interaction, but has yet to become a popular venue for music performance. This is largely because of latency in networked communication which is caused by two main factors (1) the distance that the data must travel, and (2) the architecture of Internet. Latency makes it difficult to perform rhythmically synchronized music.

This thesis, at its core, seeks to answer the question: What are the ways in which we could play rhythmically synchronized music over Internet? We take two different approaches. One is to explore how highly synchronized clocks can allow us to tightly coordinate events on systems that are separated by great distance. The other is to use prediction to reduce latency beyond what latency optimization techniques can achieve.

We build upon a body of work of artists such as John Cage, Pauline Oliveros, the Hub, and Chris Chafe who have explored various ways to play music over networks in spite of this latency. In the beginning, artists such as Pauline Oliveros incorporated latency into their works as part of the medium. As network throughput progressed artists like Chafe et al. (2000) developed tools to reduce latency as far as possible. Later, Sarkar and Vercoe used prediction to create TablaNet (Sarkar et al., 2007) which enabled drummers to play as if there were no latency. We seek to take this concept to the next logical step. While TablaNet predicts with a pattern granularity, we predict on note granularity, to give our instrument a feel more akin to traditional instruments where one gesture equals one note. We also drew from and contribute back to research about musical gestures. Selecting gestures for their *predictability* and using them to reduce the latency of networked communication represents a new type of gesture use within this body of research.

In the creation of our solutions we lean heavily on research into GPS clock synchronization to simulate a world where clocks are highly synchronized in order to explore how tempo synchronization should be done in such a situation. We build upon the work of Carôt et al. (2009) and his system NetJack, deploying musical sequencers to explore their use as a possible paradigm for Internet musical performance. We study musicians as they play with both systems and evaluate their effectiveness at enabling rhythmically synchronized music performance over the Internet.

3 The Global Metronome

3.1 Introduction

As described in the previous chapter, one of the biggest challenges of Internet music performance is synchronizing tempos between players who are not in the same physical location. To address this problem we created the Global Metronome, which is a simple, computationally inexpensive approach to obtaining *absolute tempo synchronization* between ensembles or processes. Absolute synchronization means that each Global Metronome synchronizes and aligns to a central authoritative tempo source and (if the tempo is constant) all Global Metronome implementations beat in time with one another, automatically, with no communication between them. The Global Metronome assumes that the system clocks of each device are synchronized to a high degree, whereas current approaches such as MIDI clock (Moog, 1986), Link (abl, 2016) depend on constant communication between devices in order to stay in time with one another. This approach can be implemented on a variety of devices, and as long as their clocks are synchronized it greatly simplifies the process of tempo synchronization.

This work was done in anticipation of a time in the near future when many computer system clocks will be synchronized to a high degree (less than 1 ms of error). These synchronized clocks will open up a new set of artistic interactions, allowing us to coordinate events across unconnected computers to a degree that is currently impossible. These events may include media playback (e.g., audio and video), mechanical manipulation of the world, and other computational processes. In our particular case, highly synchronized system clocks allow us to provide tight tempo synchronization over long distance, with very low computational overhead.

The work presented in this chapter makes three contributions. The first contribution is the concept of the Global Metronome, which can be used on a variety of devices. The second contribution is evidence demonstrating the effectiveness of this method on currently available hardware. The third is the Pi Global Metronome Implementation (PIGMI), which is an inexpensive, open-source implementation that grants people access to a highly-synchronized clock source, and can be built for under $100.

3.2 Background

The Global Metronome approach takes inspiration from a tempo management system used for centuries—the conductor. Sound travels roughly one foot per millisecond, so a physically distributed ensemble (e.g., a marching band) needs a mechanism to stay synchronized in spite of audio latency. To help, they use a central timing source: the conductor. This approach capitalizes on the fact that light travels faster than sound, which allows each performer to see a globally synchronized timing source. In cases where the ensemble is very large, multiple conductors are used, and each of these conductors synchronize to a main conductor as shown in Figure 3.1. This is the approach that the Global Metronome uses. Multiple sub-conductors (Global Metronome implementations) synchronize to a virtual "master" tempo, and players synchronize to the metronomes.

The Global Metronome relies on a high degree of system clock synchronization, which can be difficult to achieve. The main challenges are: (1) clocks in electronic devices suffer from varying degrees of inaccuracy and (2) high accuracy clock synchronization over networks is vulnerable to error. All clocks advance at different rates (Moon et al., 1999). This difference in rate is called clock drift, and it is affected by factors such as heat and imperfection of construction materials. As an example, two of our test computers have a drift of 0.7 ms per second, so after 1 minute they are 42 ms out of sync with one another—an audible difference (note that this is an extreme example). In order to keep clocks in better time we can use network clock synchronization algorithms to estimate drift, and adjust the time of the clocks. The most popular algorithm is the Network Time Protocol (NTP) (Mills, 1991). However, NTP is accurate on the order of tens or hundreds of milliseconds, which is sufficient error to cause tempo drag effect (Chafe et al., 2010). The accuracy of NTP depends on a number of factors such as network latency and accuracy of timeservers, but the most important one is network route asymmetry (see Section 2.5.2 for an in-depth discussion). Standard (unmodified) NTP assumes that the transit time to and from a timeserver is symmetric, but in reality it may not be. The synchronization messages (and all Internet messages) take different paths in each direction, and routers have different levels of buffer congestion depending on path. This asymmetry is the main source of error in NTP, and any other network-based clock synchronization scheme (e.g. Precision Time Protocol). If we could estimate the route symmetry between two hosts, we could accurately synchronize their clocks, but the symmetry estimate procedure depends on the accuracy of the host clocks, resulting in a chicken and egg problem.

Two recent developments are poised to make clock synchronization far more accurate. These are GPS as standard equipment in computers, and the clock synchronization requirements of LTE (a.k.a. 4G) cellular networks (Sesia et al., 2009). Neither of these technologies suffer from the route asymmetry problem. GPS sends a time signal that has a theoretical accuracy of 14 nanoseconds (Dana, 1997), which is a few orders of magnitude more accurate than is needed for our tempo synchronization. The signal for GPS is freely available to anyone with a receiver and a view of the sky. LTE time is transmitted from cell base stations to LTE enabled devices. The LTE standard requires that clocks be synchronized to within 5-1.5 microseconds (depending on their distance from the transmitter) (Weiss, 2012). While GPS is globally accurate, the accuracy of LTE time synchronization depends on the accuracy of the LTE transmitter tower clock. This can vary depending on the operator but many major telecommunications companies and equipment vendors are pushing for higher degrees of clock synchronization (Neil, 2012; Ericsson, 2015; Nokia, 2017).

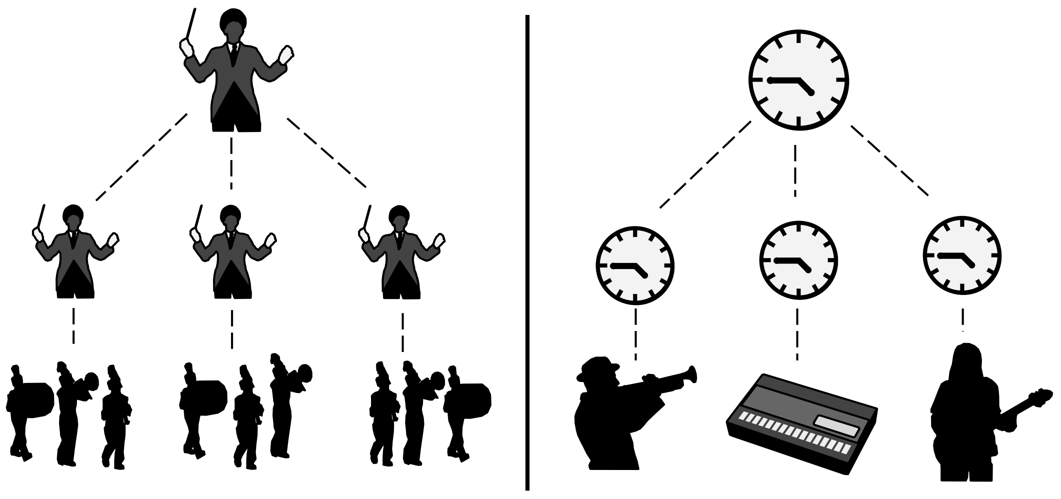


Figure 3.1: The Global Metronome approach is similar to using multiple conductors in a large marching band. Sub-conductors follow a head conductor to help all parts of the band stay synchronized. This approach relies on highly synchronized system clocks, which will become more common in the future. The "head conductor" is a virtual metronome that is imagined to have been ticking at tempo since the Unix epoch.

There already exist a number of effective tempo synchronization methods such as MIDI time clock (Moog, 1986), and new offerings such as LANdini (Narveson & Trueman, 2013) and Ableton Link (abl, 2016). These methods have been designed for synchronization of co-located players and instruments. While the Global Metronome could be used locally to provide wireless synchronization but it is intended to complement them by acting as a bridge to connect ensembles across long distances. For instance, in our user studies in Chapter 4 we use the Global Metronome for remote synchronization and MIDI clock for local synchronization.

3.3 The Global Metronome Phase

The Global Metronome approach uses an agreed-upon method to compute the next "tick", or phase, of the metronome. To accomplish this we think of the relationship between a metronome and time t as a rotating phasor with phase $0 \leq \varphi(t) < 2\pi$, as shown in Figure 3.2.

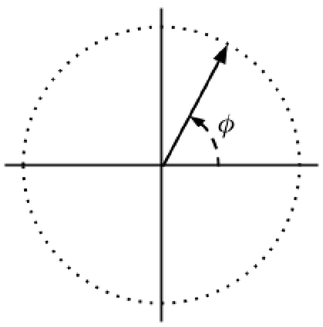


Figure 3.2: A metronome ticking at f bpm is represented as a phasor which rotates 2π radians once every $T = \frac{60}{f}$ seconds. For any time t, the current phase $\varphi(t)$ is the fractional portion of $\frac{t}{T}$.

For a given tempo f (in beats per minute), the phasor makes f rotations every minute, and the metronome "tick" (if desired) is played when $\varphi(t) = 0$. When two metronomes have $\varphi_1(t) \equiv \varphi_2(t)$ for all t they are synchronized with one another, rotating at the same rate and with the same phase. Given synchronized system clocks, the concept of the Global Metronome is, at its core, a standardized way to compute $\varphi(t)$ so that we get tempo synchronization for free.

Computing the current phase of the Global Metronome is straightforward and computationally inexpensive. Imagine that a "virtual" metronome has been ticking at a constant tempo since the Unix epoch (January 1, 1970). For a Unix time t in seconds, we divide by the period of the metronome, and the fractional part determines the current phase of the master metronome.

$$T = \frac{60}{f} \tag{3.1}$$

$$\varphi_{global}(t, T) = 2\pi\left(\frac{t}{T} - \left\lfloor\frac{t}{T}\right\rfloor\right) \tag{3.2}$$

where f is the tempo in beats per minute, T is the period, and φ_{global} is the phase of the Global Metronome.

Sometimes we might want the phase of a local metronome to be purposefully offset from the Global Metronome (e.g. after a tempo change). Therefore we allow a local client to specify a local phase offset term $0 < \theta < 2\pi$. The phase of a client at time t, given the desired tempo f and local phase offset θ, can be precisely computed in terms of the Global Metronome as

$$\varphi_{local}(t, T, \theta) = \left[2\pi\left(\frac{t}{T} - \left\lfloor\frac{t}{T}\right\rfloor\right) + \theta\right] \mod 2\pi \tag{3.3}$$

where T is the period as computed in Equation 3.1.

A basic example of when we might want to use a phase offset is if we wanted to begin playing immediately, without waiting to align with the Global Metronome. In this case when we starting playing we would almost certainly be out of phase with the Global Metronome, but we could transmit our current phase offset to our collaborators. Another common case would be after a smooth tempo change, which we will discuss next.

3.4 Tempo Changes

The Global Metronome can help to make tempo changes easier by guaranteeing that once a change is complete, all players are still synchronized with each other. All that is needed is that they use the same f and θ. However, changing from tempo f_1 to f_2 will cause a discontinuous jump in metronome phase because $\varphi(t, T_1, \theta\star) \neq \varphi(t, T_2, \theta\star)$ for nearly all $f_1 \neq f_2$. In this section we explain how to properly execute perceptually smooth tempo

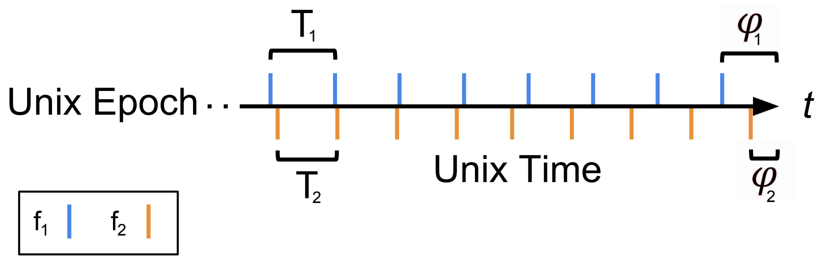


Figure 3.3: For tempos f_1 and f_2 the orange lines represent the times when $\varphi(f_1) = 0$, and the blue lines represent when $\varphi(f_2) = 0$. An instantaneous switch from f_1 to f_2 will almost certainly cause a discontinuous phase jump, no matter how small the tempo change is. Therefore for each tempo change, we must compute a new phase offset.

changes by (1) dividing the full tempo change into smaller quantized steps and (2) adjusting the phase offset θ to maintain smooth progress through the current beat.

The most important detail to address when making tempo changes is making sure the metronome phase progresses smoothly, without any jumps. Tempo changes using the Global Metronome are different than other tempo systems because of how we divide Unix time into phasor rotations, as shown in Figure 3.3. When we change tempo, the number of completed phasor rotations changes, as does the current φ. The overwhelming majority of changes in tempo (no matter how small) will cause a discontinuous jump in φ. To maintain smooth progress through the current beat we need to recompute θ for each new tempo we pass through.

To execute a perceptually smooth tempo change, we want to quantize our tempo changes into a series of discrete tempo change "steps" that are large enough to have minimal impact on processor usage, but small enough so that the change is perceptually smooth. For example, if we want to transition from 60 bpm to 72 bpm over 3 beats, we divide those beats into 12 equal steps of 1 bpm tempo increases. Implemented in terms of metronome phase, we specify that this tempo change happens over $3 \times 2\pi$ radians, spending $\pi/2$ radians of the phasor in each intermediate tempo. (We recommend 1 bpm for each step value because—for tempos greater than 50 bpm—it is below the 2% just noticeable difference recommended by Vos et al (Vos et al., 1997). However, this quantization amount can be changed to suit the situation.)

During a tempo change it is useful to be aware that there are two different "time" worlds. The first is *Unix* time, and the other is *beat* time. As we perform our acceleration, we spend the same amount of beat time in each tempo (e.g. in case above $\pi/2$ radians, or 1/4 of a beat). If our tempo increases, however, we spend a decreasing amount of Unix time in each successive tempo step, and if our tempo decreases we spend more time in each step. When we change to a new tempo, we recompute our phase offset as

$$\theta_{new} = \left[\varphi_{local} - 2\pi \left(\frac{t}{T_{new}} - \left\lfloor \frac{t}{T_{new}} \right\rfloor \right) \right] \mod 2\pi \tag{3.4}$$

where t is the current Unix time, T_{new} is the period of our new tempo, and φ_{local} is the current phase. Now, if we use θ_{new} in the formula $metro(t, f, \theta_{new})$, there will be no phase discontinuity between the two tempos. Note that this method is just one example of how to make a tempo change. Other methods might apply a curve to the rate change, or spend equal Unix time in each intermediate tempo. The important thing to remember is that (1) we must recompute θ with every tempo change and (2) we should quantize our tempo changes into steps that are large enough to be easy on our processor, but small enough to be perceptually smooth (we recommend 1 bpm steps).

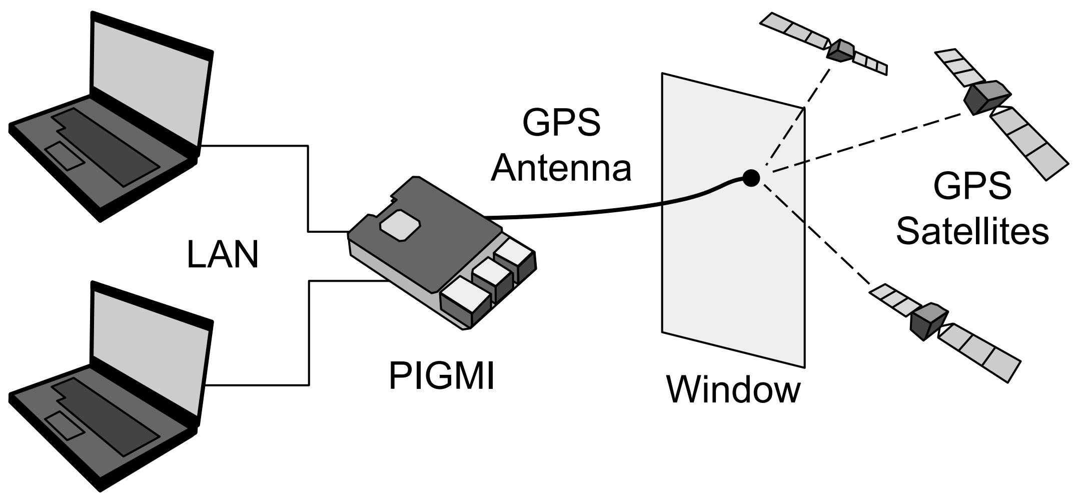


Figure 3.4: The Pi Global Metronome Implementation (PIGMI) is inexpensive and portable, and greatly simplifies the process of synchronizing to the Global Metronome. It requires a view of the sky in order to sense GPS satellites. Its antenna is placed in a window and is connected to an Ethernet network.

3.5 PIGMI: The $99 Global Metronome

Using the Global Metronome approach requires access to a local high-accuracy time source. For this purpose we have created the Raspberry Pi Global Metronome Implementation (PIGMI), a tiny, inexpensive timeserver that can be used to synchronize computers on a LAN. The hardware consists of a Raspberry Pi 2, a GPS expansion board, an active GPS antenna, and a USB wifi dongle. We have created instructions for building a PIGMI on our project website [*]. Included are hardware recommendations and a Raspbian Linux image with the software preinstalled.

A PIGMI must be placed near a window that has a view of the sky and (if desired) connected it to a LAN, as shown in Figure 3.4. The PIGMI currently uses LANdini (Narveson & Trueman, 2013) as its primary method to transmit clock synchronization information

[*]http://pigmi.cs.princeton.edu

to local devices. LANdini is a software suite that simplifies time synchronization and OSC message transmission between host computers on a LAN. In the future we plan to support more local synchronization methods (e.g., Ableton Link and MIDI clock). Note that each PIGMI is autonomous, and there is no limit to how many different geographic locations can use this method. Because each PIGMI gets its timing information from GPS, there is no additional computational overhead to adding more PIGMIs.

The PIGMI is designed to be used in a number of different configurations. First, it can be used as a metronome, generating audio internally. The included metronome software uses a GPIO pin to generate sharp (but pleasant), audible clicks with low latency. A user can connect the GPIO pin to an audio mixer to hear the result. Second, the PIGMI can act as an NTP server. NTP over the Internet can be inaccurate for tempo sync purposes, but NTP over a LAN can yield extremely accurate time synchronization, as we will show in our evaluation. Third, the most convenient way to use the PIGMI is via LANdini. When a PIGMI is connected to a LAN, other hosts running LANdini will automatically discover it and begin to synchronize their timing to it.

3.6 Evaluation

In order to gauge the accuracy of the Global Metronome given current technology, we conducted a series of studies. GPS research shows that as long as a receiver can sense three GPS satellites, it will be able to sense the time synchronization signal with a theoretical accuracy of 14 nanoseconds (Elson et al., 2002). However, reading the sync signal and then conveying the synchronization information to another computer incurs error. The goal of the following evaluation is to gain an intuition into the magnitude of error that different configurations of PIGMIs incur.

We tested three different useful PIGMI configurations, shown in Figure 3.5, and described in detail below. In every case, there are two audible metronomes ticking at $f = 120$ with $\theta = 0$ for 30 minutes, with the audio generated from either a PIGMI or a computer. We recorded each audio stream, and for each tick we computed the timing offset (in ms) from the corresponding tick in the other audio stream. We also recorded, for comparison, a configuration where one PIGMI was not synchronized to the Global Metronome, and another configuration with two commercial drum machines synchronized via MIDI clock (Figure 3.6). In the descriptions below, PIGMI-1 and PIGMI-2 are the names of specific PIGMI systems that are re-used throughout the experiment. Note that although these devices were next to one another in our lab, GPS receivers that are physically far apart are expected to have the same synchronization accuracy.

- Configuration 1 (both GPS): In this configuration two independent PIGMIs (PIGMI-1, PIGMI-2), each with a GPS sensor, generate audio ticks by switching a GPIO pin from high to low and back again. The GPIO pins were connected to a mixer, and the audio was recorded from the master out of that mixer. This configuration is meant to investigate the error incurred for devices that have direct access to GPS.

- Configuration 2 (GPS and NTP): Both PIGMIs are synchronized to GPS. PIGMI-1 generates ticks directly via GPIO pin. A PC host (a 2011 Macbook Pro running OS X Yosemite) synchronizes its system clock via NTP to PIGMI-2. On the host, metronome tick times are computed by referencing its system clock, and the audio is generated via a Python program. This configuration is meant to test the scenario where systems connect via NTP to a low-latency, accurate timeserver (in this case, a PIGMI).

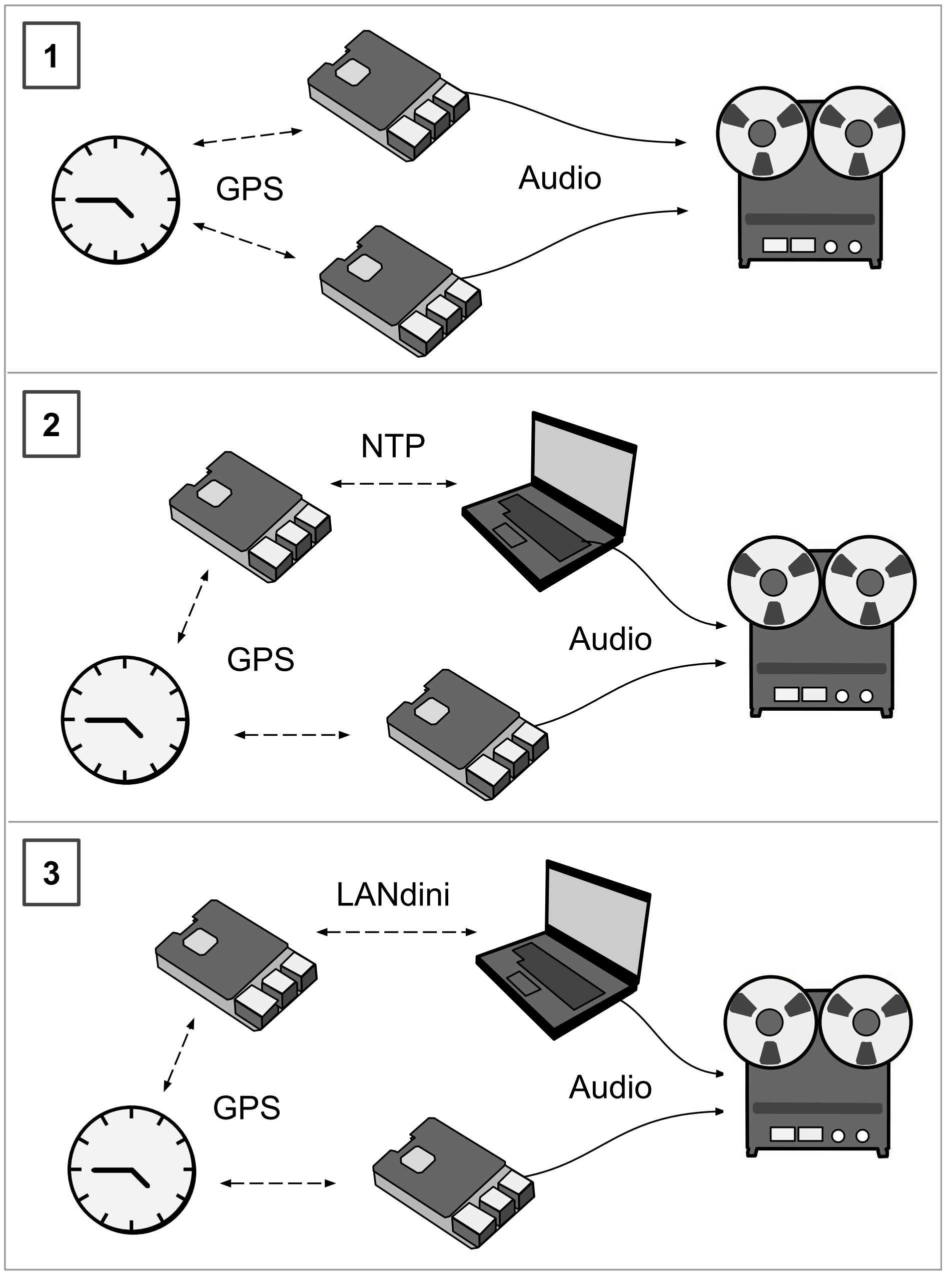


Figure 3.5: We tested 3 configurations of PIGMIs along with two "control" configurations (shown in Figure 3.6) in order to learn the magnitude of synchronization error. In each of the displayed configurations, we recorded 30 minutes of audio ticks at $f = 120$ beats per minute. Then, we analyzed the per-tick timing differences (offsets) of one recording compared to the other.

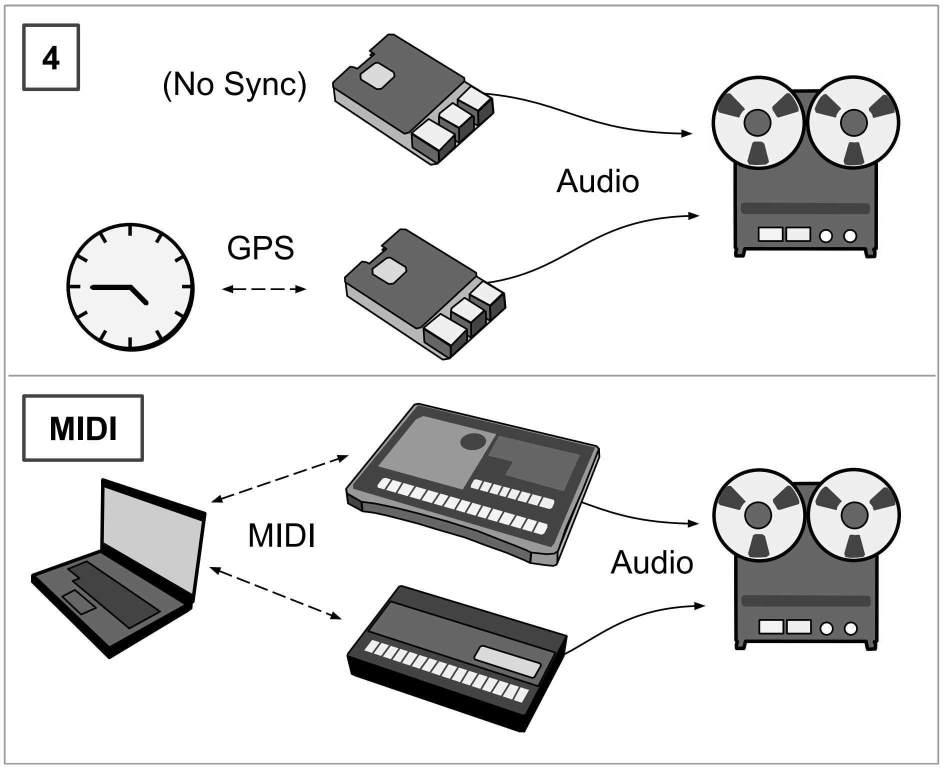


Figure 3.6: These configurations serve as informal synchronization benchmarks. Configuration 4 tests the scenario when one PIGMI is not synchronized to the Global Metronome, while "MIDI" tests two commercial drum machines, connected via a physical MIDI cable. Just as for the configurations in Figure 3.5, we recorded 30 minutes of audio ticks at 120 beats per minute. Then, we analyzed the offsets of these ticks.

- Configuration 3 (GPS and LANdini): Both PIGMIs are synchronized to GPS. PIGMI-1 generates ticks via GPIO. The PC host connects via LANdini to PIGMI-2. In this case it is not the system clock that is synchronized to PIGMI-2, but a software clock written in SuperCollider (McCartney, 2002). The Global Metronome tick times are computed by referencing the software clock and the audio is generated in SuperCollider. This configuration is meant to test the amount of error when using LANdini to connect to a timeserver.

- Configuration 4 (GPS and no sync): PIGMI-1 is synchronized to GPS. PIGMI-2 is unsynchronized and uses its free-running internal clock for timekeeping. This is to illustrate drift between two clocks.

- MIDI clock: A Korg Volca Beats and a Korg Electribe ER-1 are connected via physical MIDI cable to a host running Ableton Live which sends MIDI clock to both. This configuration is to give intuition into the amount of jitter that exists in commercial MIDI devices.

Figure 3.7 shows the tick offsets over each 30 minute recording. First, note the high degree of accuracy for the GPS-connected and NTP-synchronized configurations when compared to the MIDI connected devices. While they exhibit occasional spikes, these spikes are generally less than 3 ms. Also, note that the "No Sync" scenario (Configuration 4) has an average offset of -2 ms. We fit a linear regression to each set of data. Of the 5 configurations, the "No Sync" scenario is the only regression with a non-zero slope of -0.006 ms/min with $p < 0.0001$. All other configurations had $p > 0.33$.

Figure 3.8 shows the histogram of offsets for all configurations, along with their standard deviations. Note that the x-axis ranges vary from plot to plot. Both GPS and NTP exhibit extremely small offsets. The vast majority are less than 1 ms. LANdini exhibits performance on par with the directly connected drum machines. The "No Sync" configuration has an offset of -2 ms, and this offset will become more negative with time, due to the drift between the clocks. Our MIDI devices exhibit a bimodal distribution in offsets, and have a standard deviation that is five times that of the GPS and NTP synchronized devices.

3.7 Discussion

Our data shows that a directly-connected GPS sensor, NTP to a GPS-connected device, and LANDdini to a GPS-connected device are all extremely accurate ways to achieve accurate time synchronization. Their performance is on par with a physical MIDI clock connection despite the fact that they could be hundreds of miles apart from one another. (Note, again, that GPS receivers that are far apart are expected to have the same accuracy as those that are side by side (Elson et al., 2002).

The mechanism used in Configuration 1 is a "best-case" scenario for latency, since nearly all real-world mechanisms for generating audio will involve more computational and/or communication overhead than switching a GPIO pin. However, Configurations 2 and 3 suggest that our approach supports very accurate synchronization even under more realistic circumstances, where audio is generated by a high-level programming language (Python in Configuration 2, SuperCollider in Configuration 3) running on a standard laptop computer (running OS X Yosemite in both cases).

In practice, the Global Metronome approach is most immediately applicable to supporting musicians who play genres such as Hip Hop or Electronic Dance Music. The Global

Metronome (along with an audio streaming technology such as JackTrip (Cáceres & Chafe, 2010)) can allow an unlimited number of artists to play in sync with one another. The Global Metronome could also be used to synchronize drum tracks for rock and jazz musicians, and other genres where players are accustomed to playing to click tracks. For ensembles who want the freedom to change tempos organically, the Global Metronome could be one component of a more sophisticated system, incorporating (for instance) virtual tempo controls exposed in a convenient manner to a human conductor, or automated beat tracking, or conductor tracking applied to designated musical leaders.

3.8 Conclusion

In this chapter we have described the Global Metronome, an *absolute* tempo synchronization scheme that capitalizes on the fact that computer system clocks are becoming increasingly easier to synchronize to a high degree. We have presented evidence to show that it is possible to synchronize unconnected devices to a degree suitable for music performance. Additionally we have presented the PIGMI (Pi Global Metronome Implementation), a tiny, affordable timeserver that greatly simplifies the process of using the Global Metronome approach. In the following chapters we will present a number of projects which use the Global Metronome approach and/or the PIGMI in order to simplify musical collaboration and enable new musical interactions.

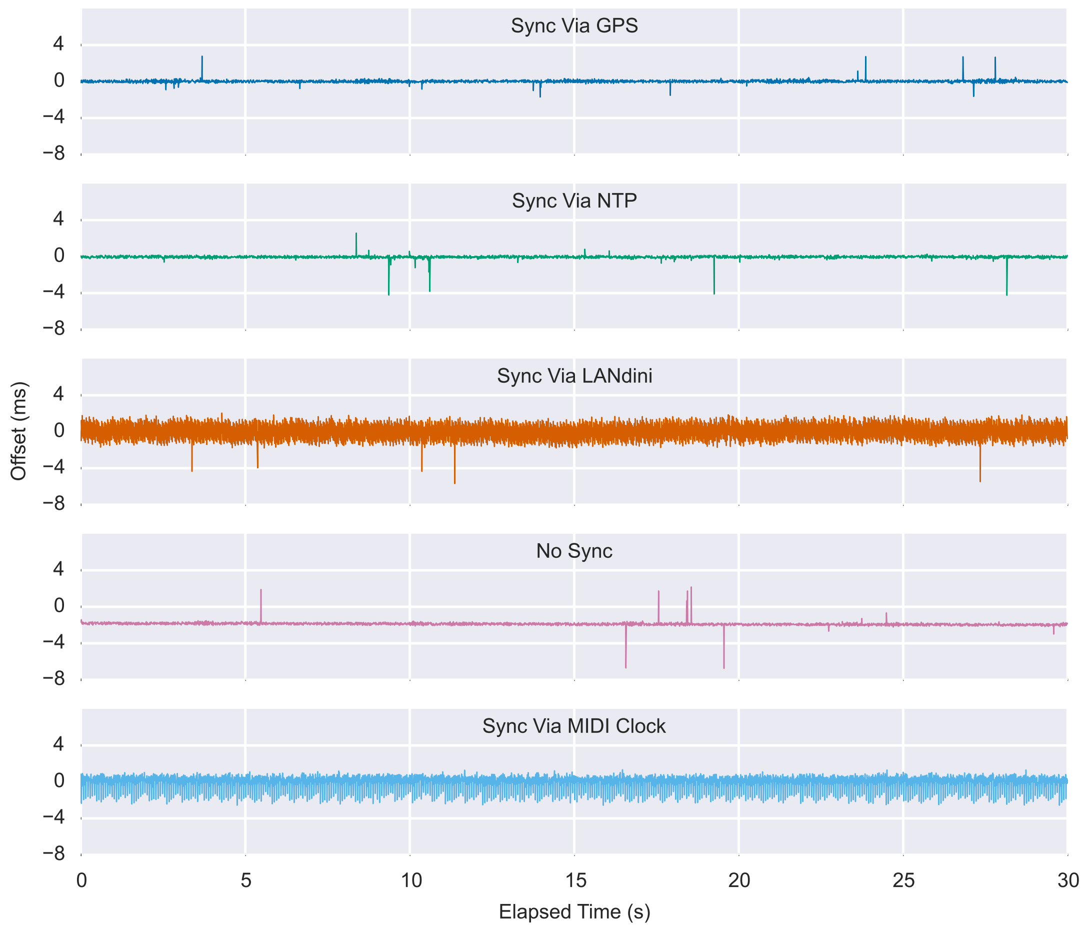


Figure 3.7: The offsets for each experiment configuration for $f = 120$, $\varphi = 0$ over a period of 30 minutes. See Figures 3.5 and 3.6 for an illustration of the configurations. Note that GPS and NTP offer high accuracy synchronization, superior to the physically connected MIDI drum machines. LANDindi exhibits performance on par with MIDI. In all three PIGMI use cases, the devices could be hundreds of miles apart from one another. Note the -2 ms offset for the “No Sync” scenario.

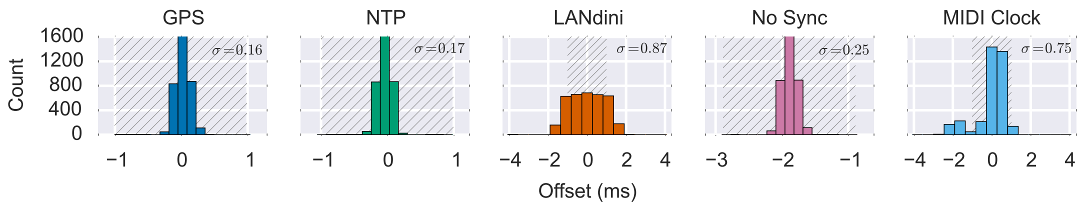


Figure 3.8: Histograms of the offset data, along with σ, the standard deviation for each. Note that the x-axis range is not the same for each plot. The hatch-shaded area indicates a 2 ms range centered on each mean. GPS offers the most accurate synchronization, followed closely by NTP, while MIDI clock and LANdini exhibit similar performance. “No Sync” has low standard deviation, but is offset by -2 ms.

4
The Global Metronome in Use

4.1 Introduction

In this chapter we explore the types of musical interactions that are made possible by the Global Metronome and the PIGMI. Our initial goal is to test the effectiveness of the Global Metronome as a tempo synchronization device for Internet musical performance, but beyond that, we also want to understand some of the barriers that stand in the way of online music collaboration. These include details such as problems when setting up the PIGMI, establishing audio connections, and communicating with one another while playing. Therefore, we catalog all technical hurdles, including those caused by devices other than the PIGMI. Finally, we explore a musical paradigm which we feel is promising for playing collaborative music over the web—sequenced music. The PIGMI is the first device that we know of which can allow for MIDI clock synchronization for networked performance. Therefore it is now possible to play collaborative sequencer-based music over the web. Our hypothesis, partially explored here, is that this style of music-making is particularly well-suited for Internet performance.

Sequenced music was previously used by Carôt et al. (2009) as a means for testing Netjack, their networked music performance audio framework. Sequenced music is a relatively new performance method that emerged in the 1970's alongside synthesizers, and offers its own methods of expression. Instead of musicians playing the individual notes, phrases are programmed into a device that plays them back, often in loops. These loops are layered on top of one another in order to build compositions. Expressiveness is achieved by starting and stopping loops, changing synthesis parameters, adjusting volume and other methods that are often associated with studio mixing techniques. The performer's role is similar to one of a conductor, but one who can change the composition at will.

Sequenced music is an ideal paradigm for testing the Global Metronome. Because timing is controlled by a machine, the timing is very accurate. By removing human timing error it allows us to gather data on the accuracy of tempo-keeping of the Global Metronome. We seek to answer two key questions. First, does the Global Metronome allow for tempo synchronization over the Internet? If so, there is the problem of audio latency. Although the audio generation might be synchronized in really time, it takes time for network packets containing audio data to travel from one player to another. So, second, does this delay negatively affect the quality of the playing experience, and if so, how can we ameliorate this problem?

4.1.1 Hardware Setup

We deployed three remote PIGMIs, one in Santa Monica, California, one in Denver, Colorado and one in Alma, Wisconsin. During a series of sessions these were used to play music along with a PIGMI based in Culver City, California operated by the author, who has been playing sequenced music for 20 years. All collaborations happened over residential Internet connections and no special routing was used. We performed between 3-5 sessions per participant each lasting 1-2 hours, for a total of 13 sessions.

Remote tempo synchronization between the PIGMIs was accomplished using the Global Metronome method described in the previous chapter. Attached to each PIGMI was a commercial sequencer of some sort. These included Ableton Live (Ableton, 2017), Korg Volca Beats, Korg ER-1 (Korg, 2017), Squarp Pyramid (Squarp, 2017), and Novation Circuit (Novation, 2017).

Local tempo synchronization was accomplished via MIDI clock. As shown in Figure 4.1, attached to each PIGMI was a MIDI interface. The MIDI interface sent a MIDI clock signal in beats per minute to the attached sequencer. In order to start all sequencers at the same time we waited until the beginning of the next measure to start the attached sequencer. We compute the start time t_{start} of the next measure with the following equation

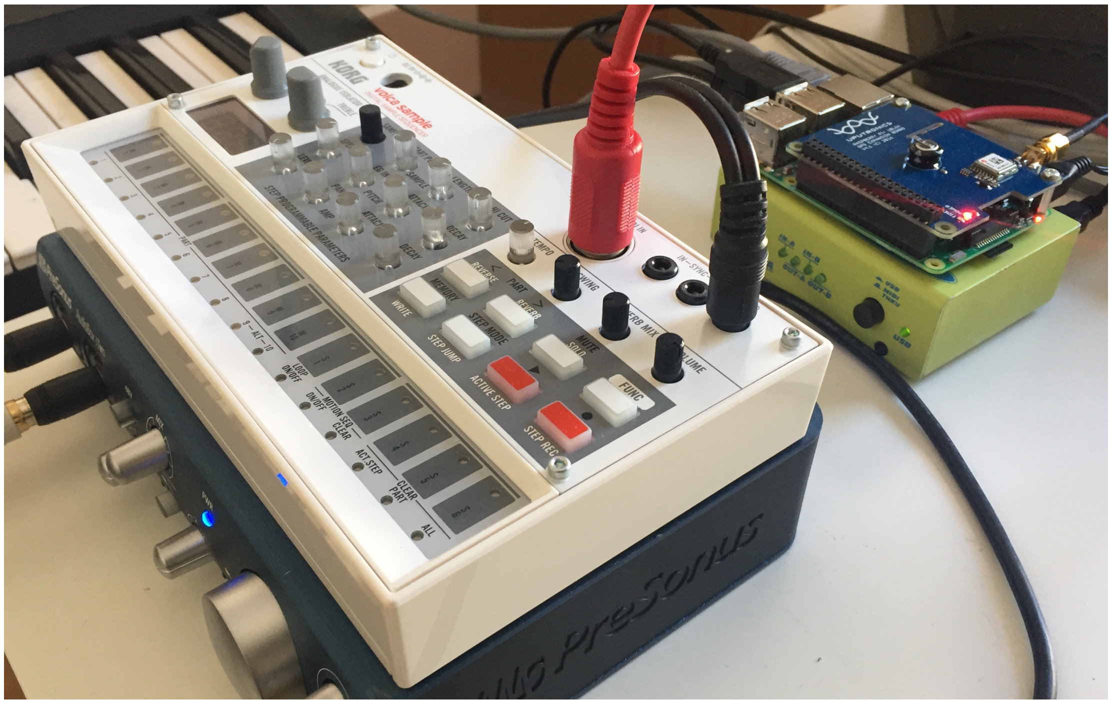


Figure 4.1: An example of a PIGMI set up for working with a MIDI sequenced instrument. Attached to the Raspberry Pi is a Midiman MIDI interface which is connected to a Korg Volca Sample sampling drum machine.

$$T = \frac{60}{f} \tag{4.1}$$

$$t_{start}(t, T, k) = Tk \left\lceil \frac{t}{Tk} \right\rceil \tag{4.2}$$

where f is the tempo in beats per minute, T is the beat period, t is the current time, and k is the number of beats in a measure.

This delay until the beginning of the next measure also afforded us the ability to start and stop tempo processes at will while remaining phase synchronized on a beat and measure level.

Audio transmission was accomplished using two different free software tools. Some test sessions used JamKazam (Jamkazam, 2017), and later tests were done using JackTrip (Cáceres & Chafe, 2010). Each of these software tools have strengths and weaknesses. JamKazam comes with a set of useful diagnostic tools for setting up the service. It provides the lowest possible instantaneous latency, meaning audio is played as soon as it is received. A consequence of the lowest latency approach is that the latency changes based on network conditions. JackTrip provides constant latency, and the amount of fixed latency is determined by the buffer size. As our studies show, the constant latency is important for sequenced music over the web. One drawback to JackTrip is that it is considerably more difficult to configure due to the complexity of routing audio using the Jack Audio Connection Kit (Davis & Hohn, 2003).

When using JackTrip we employed a *delayed feedback approach* as Carôt et al. (2009) used in their Netjack study. In this approach (shown in Figure 4.2) the player sends their local audio through a delay and manually adjusts it to to be phase synchronized with the incoming audio stream. A typical delay amount was between 40 ms and 100 ms.

4.1.2 Recruitment

Participants were recruited to set up PIGMIs in remote locations, and engage in musical jamming withe the author. Volunteers were found by posting to websites including livepa.org, craigslist.org community Los Angeles, and the Facebook Groovebox Society Group. The message was similar to the following, with slight adjustments depending on the website:

"Hi all. I've developed a technology to allow MIDI clock sync for jamming/playing over the Internet. It is as accurate as MIDI sync via a cable. I'm looking for volunteers to test it

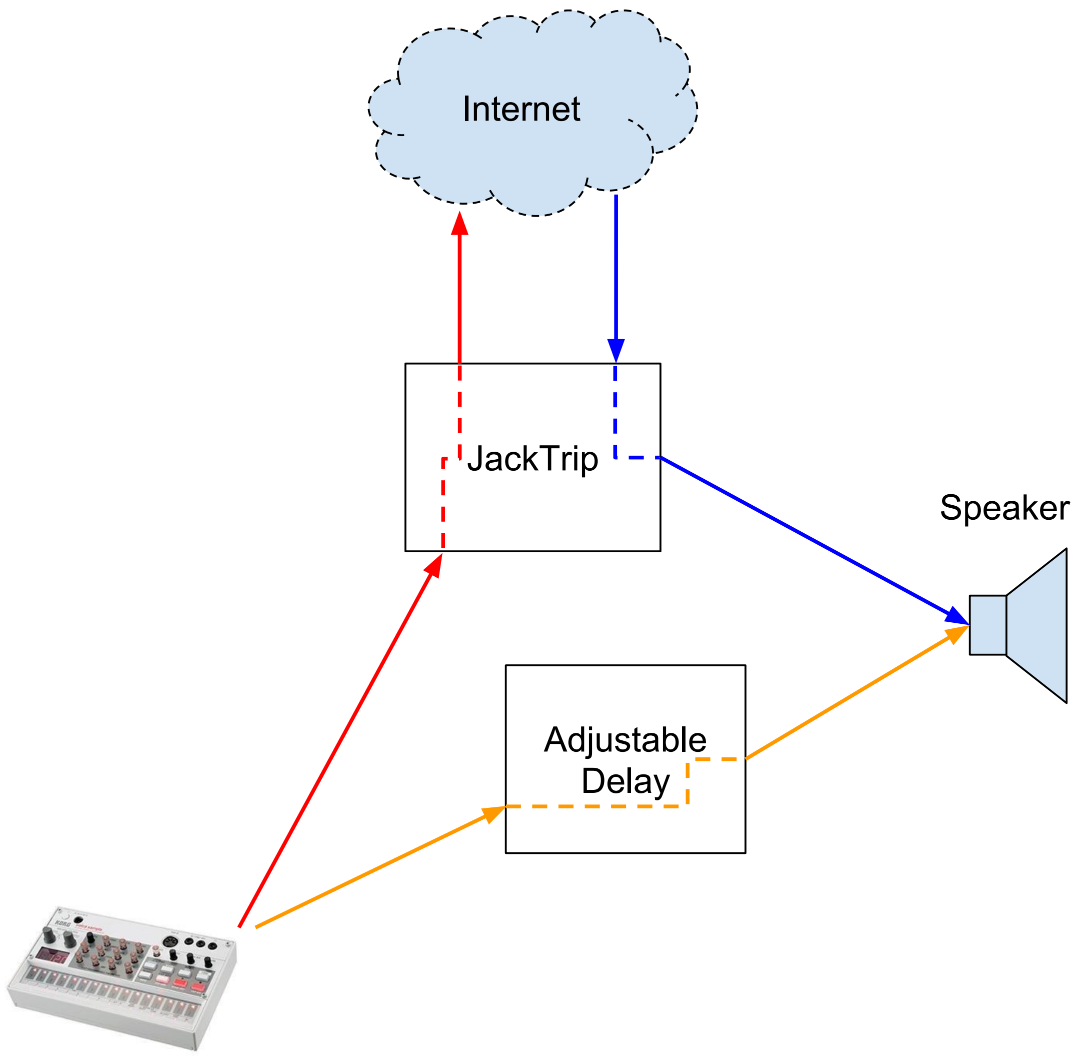


Figure 4.2: The audio routing for our JackTrip setup. In order to adjust the phase synchronization between an incoming audio stream and the local stream, the participant manually adjusts the delay of their local audio.

out. If you're interested, please contact me. Here's a link to the scientific paper (link)."

When users responded they were told:

"I will mail you a fully-functioning PIGMI device. However, you'll have to supply the MIDI interface. I'd like to have at least four sessions with you in the next two weeks. The first two will be about an hour and a half and will be dedicated to setting up the system. The second two will be dedicated to playing. I'll be taking notes everything that we do, including any problems that we have during the setup process. I'll also be collecting ping times between the PIGMI devices. For that to work you'll have create a port-forwarding rule in your router. You can close it up after we are done and it won't affect the functionality of the PIGMI."

4.1.3 Study Procedure

The studies with each participant followed a semi-structured format, but because of the exploratory nature and differing skill sets of the participants the actual format varied. In this section we describe the intended format of the studies. Later we will describe the specifics of each individual study.

Setup Sessions (typically sessions 1 and 2): Once the PIGMI has arrived at the participant's location, call them, walk them through the process of setting up the PIGMI and the audio software (JamKazam or JackTrip). Have the participant talk about the individual problems that occur during the setup. Note breakdowns and frustrations. Do not offer too much explanation, let the user struggle and ask for help. When prompted by the user, freely offer the help they ask for.

Playing Session: Play a metronome-like beat on the sequencers. Check that everything stays synchronized. Then, play in an unstructured way for at least 2 minutes at a time. Between playing conduct short semi-structured interviews. Note all breakdowns and challenges. Take note if a participant does not want to stop playing or seems eager to stop. At the end of the session have the participant fill out a short survey about that day's activities.

The semi-structured interview questions were taken from the following pool:

- How did that feel?
- Were there any problems during that jam?
- I noticed you did _____. Why did you do that?
- Did our parts sound synchronized?

The end of session survey was purposefully short in order to make it easy for the participants to complete repeatedly and included the following questions:

- For approximately how many minutes did you play?
- Did you run into any technical problems? If so what were they?
- How rhythmically tight did the music feel when both people were playing? (Response is a 10-point scale)
- How musically or socially connected did you feel with the other player(s)? (Response is a 10-point scale)

At the end of the entire study, participants were given the following exit survey:

- What is your age?
- How long have you been playing music?
- Do you have formal training?
- Have you played music professionally?

- What types of music have you played, and what types have you performed?
- What drew you to online music?
- How long have you been working with JackTrip?
- What are your goals, musically?
- Roughly how many minutes of music-related activities do you do each week?
- What is your primary instrument?

4.1.4 Data Collected

Data collected during *every* session includes:

1. Ping times between the PIGMIs every 5 seconds.
2. Detailed notes about the technical problems encountered.
3. Detailed notes from semi-structured interviews interspersed throughout the setup and playing process

During each session where music was played, we collected survey data. During only one of our sessions we collected audio ticks in the same manner as the previous chapter, replicating our in-lab study.

4.1.5 Data Analysis Methods

To answer the question of whether the Global Metronome is a reliable method for tempo synchronization over the Internet we used a combination of quantitative and qualitative analysis. Quantitatively, we performed the same study as in the previous chapter, analyzing the offset of audio ticks, but on both ends of the connection. Next, we analyzed the one-way ping times to determine how much advantage our system has over traditional network clock synchronization methods (e.g. NTP, PTP). The results of this analysis are described in Section 4.3.2 Our qualitative evaluation involved asking our collaborators about their subjective option about the degree of tempo synchronization in our semi-structured interviews, as well as our own observations about the quality of the tempo sync.

To determine the barriers to playing online as well as whether sequenced music is a viable paradigm for Internet music playing, we gathered observation data and analyzed it topically using an affinity diagram. We reviewed our notes turning each session into a series of statements such as, "spending a lot of time on audio routing", or "the drums sound like drunk drummers playing together". We put these statements onto sticky notes, and arranged them into thematic groups, shown in Figures 4.1 and 4.2. Once the groups were established we added a summary topical sentence to form the topics discussed in Section 4.4. The survey data was used to inform the topics, as well.

Because "fun" is challenging to quantify, we chose to evaluate the enjoyability of this system by analyzing the responses of the participants for valence data. In particular, we looked for expressions of frustration and expressions of satisfaction. Additionally, the post-session survey question of "How musically or socially connected did you feel with the other player(s)?" was aimed at capturing one aspect of fun.

4.2 Study Sessions, in Practice

In this section we describe in more detail the individual participants of the study as well as the details of their particular studies. It is worth mentioning again that in these informal studies the participants acted as part volunteer and part collaborator, as we worked together to overcome the technical challenges of using the system. The data from these sessions are

Measure / beat system is different using this system	Using the system is easier for those with a good sense of tempo sync	GPS is not necessary, but it makes things easier
Can't shift the downbeat	Aligning the beats isn't trivial for some people	GPS is great ground truth
Restarting on the measure works	Local click track makes it easier to play	Latency is not always symmetric
	Helps to listen in one ear (when adjusting offset)	It's possible to do this without GPS

Table 4.1: To analyze our notes we extracted statements from the notes from the various sessions and arranged them into columns. The first row is a topic sentence that summarizes the column. This is the first half of the collection.

evaluated in Section 4.4. A brief summary of each session, in chronological order can be seen in Table 4.3.

4.2.1 Participant One: Skilled Groovebox User

Participant number one (referred to as P1 from here on) can be characterized as a tech worker and hobbyist musician who plays for fun and has some desire to play shows for others. P1 lives in Santa Monica, CA and works in information technology in a creative and technical position. They are an experienced programmer who has their own Github repository. They are familiar with navigating Linux and understand Internet networking, in that they are familiar with ports and IP addresses.

P1 is in their 40s, played music from 15-25, stopped for 15 years and is just now getting back into it. They have not played music professionally. They play primarily electronic dance music. P1's goal musical goal is to "make music", and is currently piloting a setup

Constant latency is better than lowest latency	It's possible to play a traditional instrument, and they don't need to be in full "follow" mode	Audio routing can be very time consuming
High, constant latency is tolerable	"I can play with 30 ms latency, but not 80 ms—or at least, it's not fun."	Can't tell who can hear what in audio routing
Latency jitter kills the vibe	Can dictate who gets the latency	Many installation problems (Jack)
Sounds like drunk drummers from latency jitter		Audio routing and dealing with Jack is time consuming
Delay the local audio		Need Jack interface that displays audio levels, for troubleshooting
30 ms jitter is too much		

Table 4.2: To analyze our notes we extracted statements from the notes from the various sessions and arranged them into columns. The first row is a topic sentence that summarizes the column. This is the second half of the collection.

Index	Participant #	Remote Location	Activity
1	P1	Santa Monica	PIGMI Setup
2	P1	Santa Monica	Set up JamKazam; Played music
3	P1	Santa Monica	Checked for MIDI drift; Played music
4	P2	Alma	Set up PIGMI and JackTrip
5	P2	Alma	Audio routing; P2 played live keyboard
6	P2	Alma	Played music
7	P2	Alma	Played music
8	P1	Santa Monica	Set up JackTrip; Played music
9	P3	Denver	Set up Jack
10	P1	Santa Monica	Played Music
11	P3	Denver	Problems with Jack
12	P3	Denver	Problems with Jack
13	P2	Alma	Audio Tick test

Table 4.3: A chronological list of the individual sessions and the activities performed. Each session way approximately 1-2 hours in length.

that allows them to live stream themselves doing it. They make music using grooveboxes, which involves tempo synchronizing drum machines and sequenced synthesizers together to combine and layering their patterns (as described earlier). They have a large setup in the corner of their main living space, about the size of an ice cream cart where all their devices sit. They spend about 6-12 hours per week doing music-related activities. They have been using JackTrip only since the beginning of this study, and are interested in networked music performance because "everything is online". In summary, P1 is a fair representation of the target user for the PIGMI as an enabler of online sequenced music.

P1 was the first study participant. Much of our work together centered around testing the timing stability of the PIGMI as well as investigating the effects of audio latency on enjoyment. We had 6 sessions with P1, each ranging from 1.5-2 hours in length. The first three sessions were done using JamKazam as the audio communication software.

4.2.2 Participant Two: Enthusiastic Musician

Participant number two (P2) is a retiree with an enthusiastic obsession for music. P2 describes themselves as a "retired geek, broadcaster, early-ISP type". They are familiar with Linux and networking, and have been using JackTrip for about a year. They have taken Chris Chafe's Kadenze Online Jamming course (Kadenze, 2017) which provides a broad overview of networked music performance. They are in their 60s, have been playing music for 62 years and spend roughly 8-16 hours of time per week on musical activities. They play a variety of music styles including boogie woogie, folk, blues, rock, electronic music, funk, dance and machine-learning-created and generative music. They are drawn to networked music to "attain the dream of collaborating with others across the globe". They are located in rural Wisconsin, and are close (7 ms) to a branch of the USA Internet backbone.

P2 was the second study participant and their sessions overlapped with the final 3 of P1. All sessions with P1 were conducted using JackTrip as the audio software.

4.2.3 Participant Three: Aspiring Musician and Developer

Participant number three (P3) is in their 20s and works for a company in Denver, CO that develops musical synthesizers. They are technically skilled and understand networks and programming. They are moderately skilled in Linux. They play sequenced music genres of techno, drum and bass, pop and noise. Their main instrument is a modular synthesizer, and they spend about 4 hours per week making music. They have been streaming themselves via Twitch (Twitch, 2017) for about a year, and were introduced to JackTrip through this study. They have never played music professionally, but their interest in networked musical performance is to "connect to a broader audience".

Our 3 sessions with P3 were full of technical problems and we were never able to play music. However, we were able to gather data about the types of roadblocks a hobbyist might encounter when attempting to play music with others over the Internet.

4.3 Results

4.3.1 GPS vs Network-based Clock Synchronization

In this section we show that in certain cases packet-based clock synchronization are sufficient for using the Global Metronome approach, but if latency round trip times are highly asymmetric and have high jitter, synchronization can suffer to the point where it would be audible.

The highly synchronized clocks of the PIGMI enable us to determine the one-way latency between two collaborators. In this subsection we will use these one-way ping times to estimate how much clock synchronization error would have occurred using traditional network-based clock synchronization methods. As described in detail on page 18, traditional clock synchronization methods such as NTP and PTP assume that network transit times are the same in each direction of a connection, which potentially introduces error into the synchronization algorithm.

In the case that transit time is asymmetric, the error in each individual measurement can be expressed as the difference between half the round trip time $\frac{t_1+t_2}{2}$ and the actual time of the first leg t_1.

$$\begin{aligned} error &= t_1 - \frac{t_1 + t_2}{2} \\ &= \frac{t_1 - t_2}{2} \end{aligned}$$

NTP uses a custom Kalman filter to smooth noisy measurements and estimate clock offsets and drift. Here we use a simplified 1D Kalman filter (no drift estimate) to estimate clock error. The input to the filter is the error measurements and the output is a smoothed estimate of the clock synchronization error.

Figure 4.3 shows the ping times between our experimenter's (referred to as E) PIGMI in Culver City, CA and a PIGMI in Alma, WI. Note that the ping times are very consistent with the E to P2 ping times hovering around 45 ms and the P2 to E times hovering at around 40 ms. Figure 4.4 shows the per-ping errors along with the smoothed clock error estimates. The clock error is quite low with a median of 2.8 ms and a standard deviation of 0.24 ms. In this case, we could easily rely on traditional clock synchronization methods because timing error of 3-5 ms is barely audible, and is not likely to affect the perceived quality of the synchronization.

Figure 4.5 shows the ping times between our experimenter's (E) PIGMI in Culver City, CA and a PIGMI in Santa Monica, CA. Note that although the two locations are much closer in physical distance the ping times are quite erratic in the direction E to P1. In this direction ping times regularly spike about 100 ms, while in the other direction ping times are consistent and hover around 15 ms. Figure 4.6 shows the per-ping errors along with the smoothed clock error estimates. The clock error takes a while to settle, has a median of 13.7

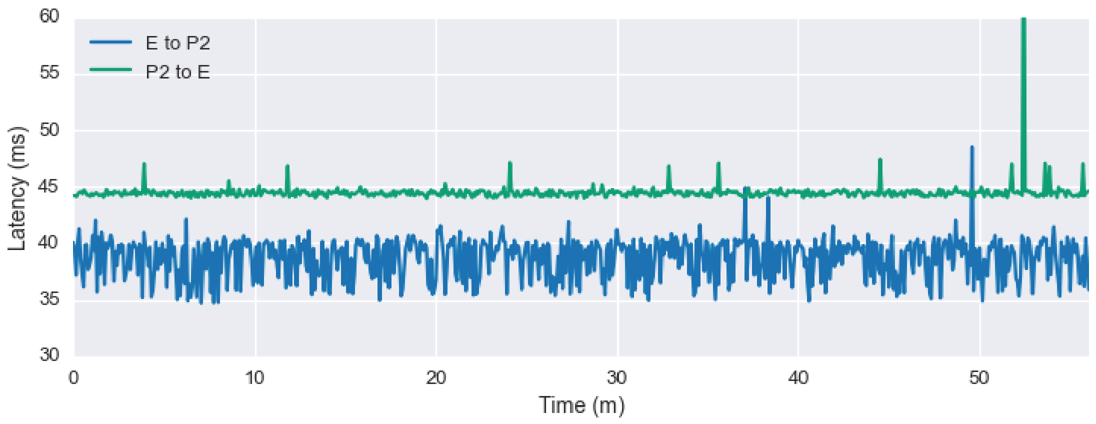


Figure 4.3: Ping times between the experimenter (E) in Culver City and P2 in Alma Wisconsin. Note how consistent they are, especially in the direction of P2 to E.

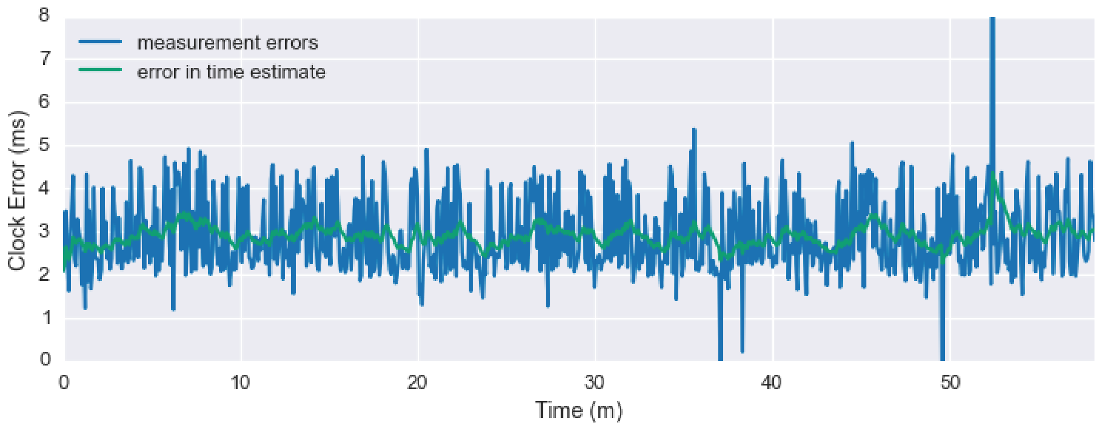


Figure 4.4: The estimated error when using a packet based clock sync approach is low, around 3 ms, and would be unlikely to cause musical problems.

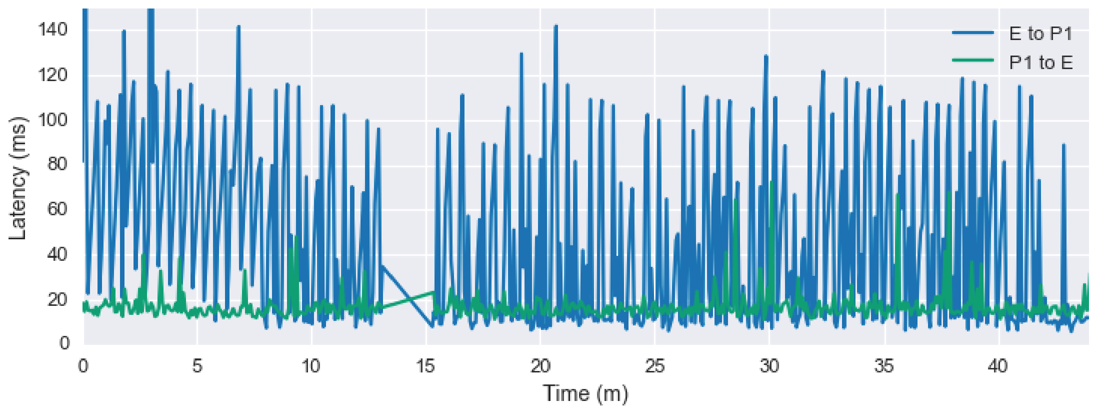


Figure 4.5: Ping times between the experimenter (E) in Culver City, CA and P1 in Santa Monica, CA. The E to P1 direction exhibit large latency jitter, and the ping times are quite asymmetric.

ms and a standard deviation of 13.6 ms. On the plot we have marked a line at 34 ms, which represents 20 ms away from the median. At the beginning of the plot, the clock synchronization begins more than 20 ms away from the median, which means at some point we would begin to hear the two streams out of sync with one another.

These two examples are representative of the ping data from each location. The communication between Culver City, CA and Santa Monica, CA was typically asymmetric in transit times, while the communication between Culver City and Alma, WI was stable and nearly symmetric.

In summary, when transit times are highly symmetric, we can use the Global Metronome approach and expect stable tempo sync. In the case where transit exhibits asymmetry but low jitter, this method could still work if we manually adjust the phase between the local metronome and remote metronome. The worst case scenario is where there are slow or fast changes to the symmetry of the transit times. It will be difficult to model the relationship between the local and remote clock and the tempo will drift. However, without highly

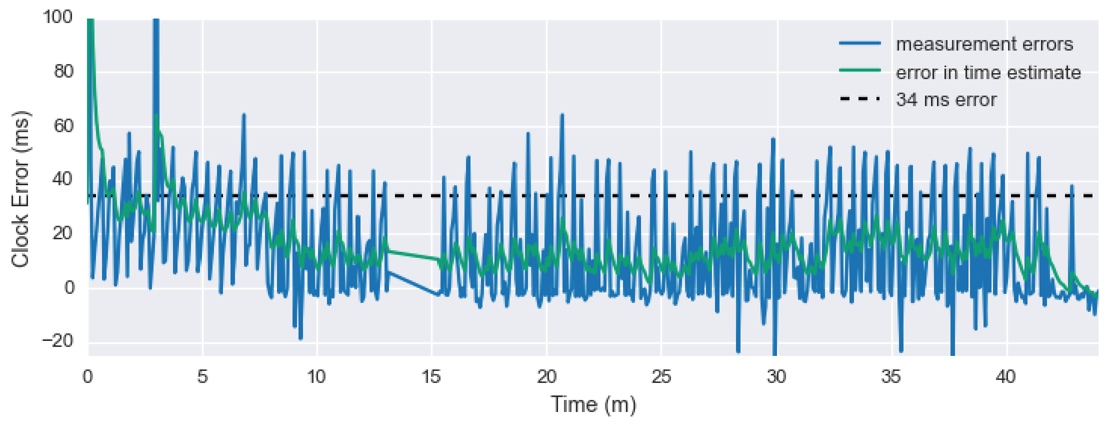


Figure 4.6: The estimated error when using a packet based clock sync approach is high, and changes with time. This level of synchronization error, greater than 20 ms, would be audible.

synchronized clocks such as are enabled by GPS, it is difficult or impossible to estimate the route asymmetry. Therefore, GPS may not be strictly necessary for this application, but it is useful.

4.3.2 Tempo Synchronization

In this section we will explore how well a GPS-connected PIGMI can synchronize tempo in practice. As shown in the previous chapter, the PIGMI worked extremely well as a tempo synchronization device, with performance on par with a physical MIDI connection. Here we present our results of using the PIGMI over the open Internet, testing its ability to keep tempo tempos in sync with one another. We had two PIGMI setups, one in Alma, WI and one in Culver City, CA each with sequencers synchronized via MIDI clock, as was shown in Figure 4.1. These generate sharp quarter note ticks at 90 bpm. We recorded the ticks in both Alma and in Culver city. We then analyzed the onset time of each tick and computed the timing error between each corresponding pair of ticks as shown in Figure 4.7. Note

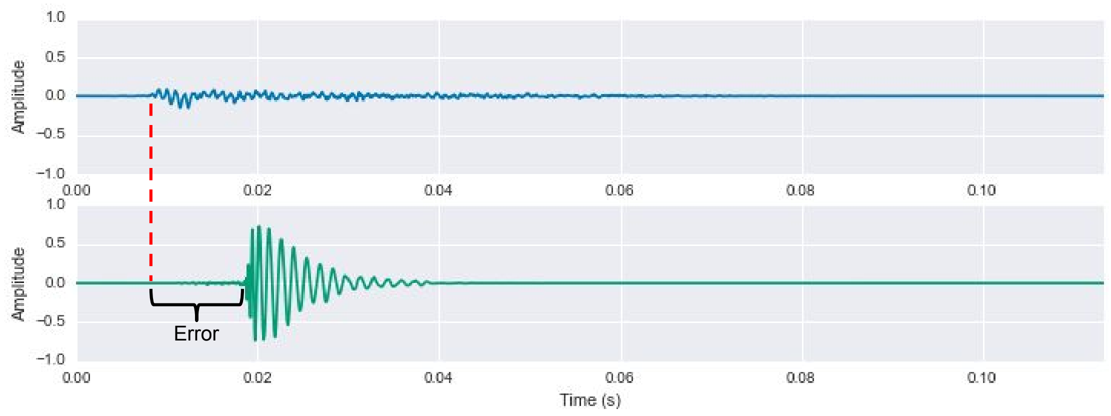


Figure 4.7: Records of two corresponding ticks, as recorded in Culver City. The top tick was sent from Wisconsin. The bottom is the *delayed* tick generated in Culver City. The error is the difference in time of the tick onset.

that the local tick sound is as it is recorded *after* after it is sent through the delayed feedback setup shown in Figure 4.2.

In interpreting the results keep in mind that there are three potential sources of error in these recordings. The first is the transmitting computer's audio interface clock error, the second is MIDI clock error, and the third is network transmission delays. We have not determined which combination of sources is responsible for the error.

Figures 4.8 and 4.9 show the timing offset over 50 minutes. The timing error of the recordings in Wisconsin are within the range of ±5 ms. Of interest is the low-frequency decreasing sawtooth wave that has a period of roughly 5 minutes. A probable source of the sawtooth wave are clock skew of audio interfaces of the computers that are used to send the audio via JackTrip.

The error in the Culver City recordings are considerably higher and exhibit a fair amount of jitter. Although the most error is in the ±5 ms range, there are a large number of spikes, some as high as 50 ms (beyond the plot range). Our hypothesis is that network congestion

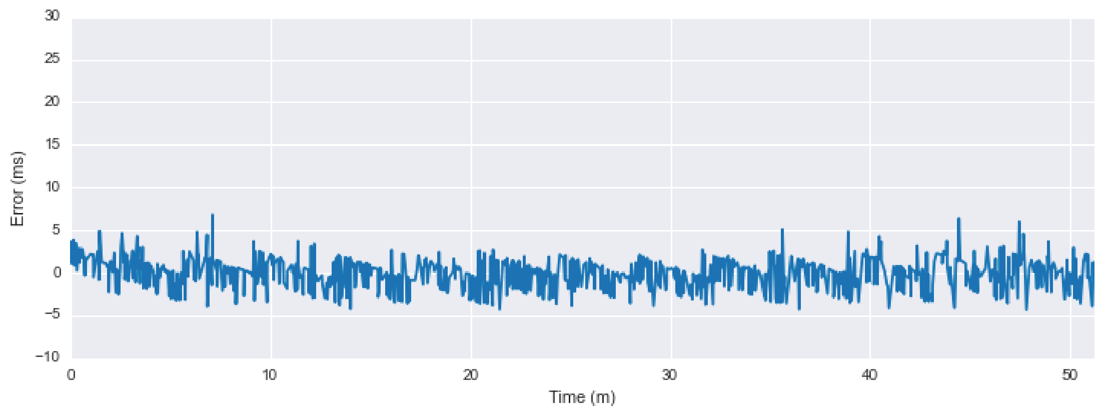


Figure 4.8: The timing error in the audio recordings made in Alma, WI.

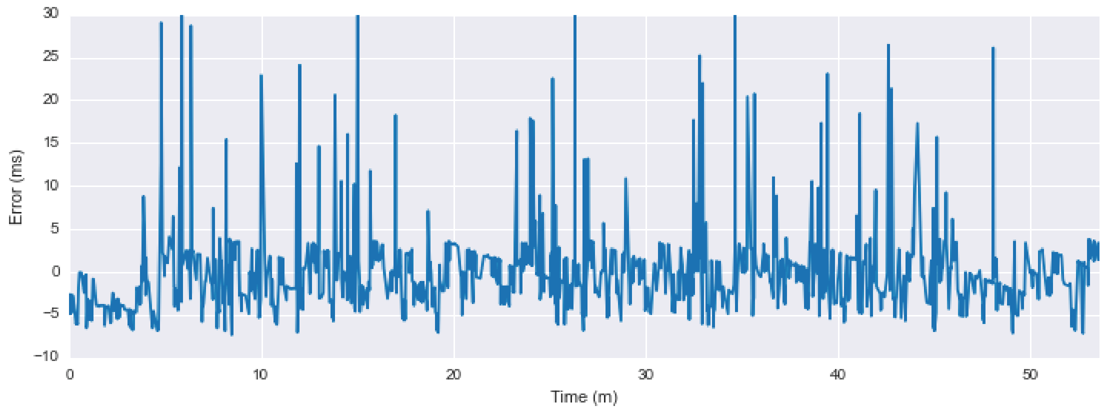


Figure 4.9: The timing error in the audio recordings made in Culver City, CA.

caused these spikes. The sawtooth wave is visible in this plot as well, though obscured by the spikes. We plan to investigate the sources of the error in future work.

In summary, the PIGMI and global metronome are effective at enabling tempo synchronization between MIDI devices that are over 1000 miles apart. Although certain factors caused timing spikes, there was no drift between the two tempos over the 50 minutes period.

4.4 Lessons from Participant Experiences

In this section we present a series of topics distilled from our notes as well as other data taken during the informal case studies. In the cases where problems are identified we offer suggestions for possible solutions or further study.

4.4.1 Constant Latency is Better than Lowest Latency

We found that for sequenced music constant latency, even high latency, is better than lowest but variable latency. Sessions 1-3 were performed using JamKazam, which provides the lowest latency possible, but this latency varies in length. The variation in latency caused our percussive sequences, when played together, to sound like two "drunk drummers" were performing together. P1 responded in one post-session survey "audio sync drifts in and out". During these sessions there was agreement between the experimenter and P1 that the tempo synchronization felt unsteady, and needed troubleshooting. The experimenter then checked the timing of the sequencer audio against the timing of metronome ticks generated using the PGIMI's GPIO pins, making sure that the system clock was being properly disciplined via GPS. Over 20 minutes there was no audible drift, and it was assumed the same for P2's setup. This is what motivated the switch to JackTrip and its constant latency time. Our impression is that greater than 20 ms timing jitter is too much, and produces the "drunk drummer" phenomenon. This contradicts our earlier assumption that timing offsets of up to 30 ms would be tolerable to users. We have not done a formal study of this threshold, which we leave to future research.

Once the sessions switched to JackTrip, using the delayed feedback approach, there were no incidents recorded in the notes where participants complained of tempo drift. The survey results reflected this as well. When using JamKazam the question of "How rhythmically tight did the music feel when both people were playing?" received an average score of 6.5 out of 10. When using JackTrip, the average was 10. Note, however, that only two responses were given using JamKazam and both were from P1.

We found that as long as the timing of the two audio streams are aligned with the timing of the remote audio (within 20 ms), users can tolerate a surprisingly high amount of latency—up to 200 ms. This was observed during sessions with P2, when the experimenter assumed the entire round trip latency burden. We have not done a formal study to this assumption.

4.4.2 Latency Can Be Flexibly Shared Among Players

We found that a benefit to using the Global Metronome along with a local audio delay is that it is possible to "distribute" the total amount of latency between two players. This means that we could determine how much of the total round-trip latency was experienced by each participant, shown in Figure 4.10. Figure 4.10-A shows a typical Global Metronome scenario where the tempo processes are phase synchronized, with their beats occurring at the same time. However, the latency times are different which means that Player 1 hears Player 2's audio later, and therefore must use more local delay on their own signal. This can be corrected by adjusting Player 2's phase to play earlier. Figure 4.10-B shows such a scenario, with both players experiencing the same amount of latency. In another potential scenario, say Player 1 might want to play a live instrument along with the music. Rather than assume the full leader-follower scenario shown in Figure 4.10-D, Player 1 can absorb some of the latency and still play comfortably, as shown in Figure 4.10-C.

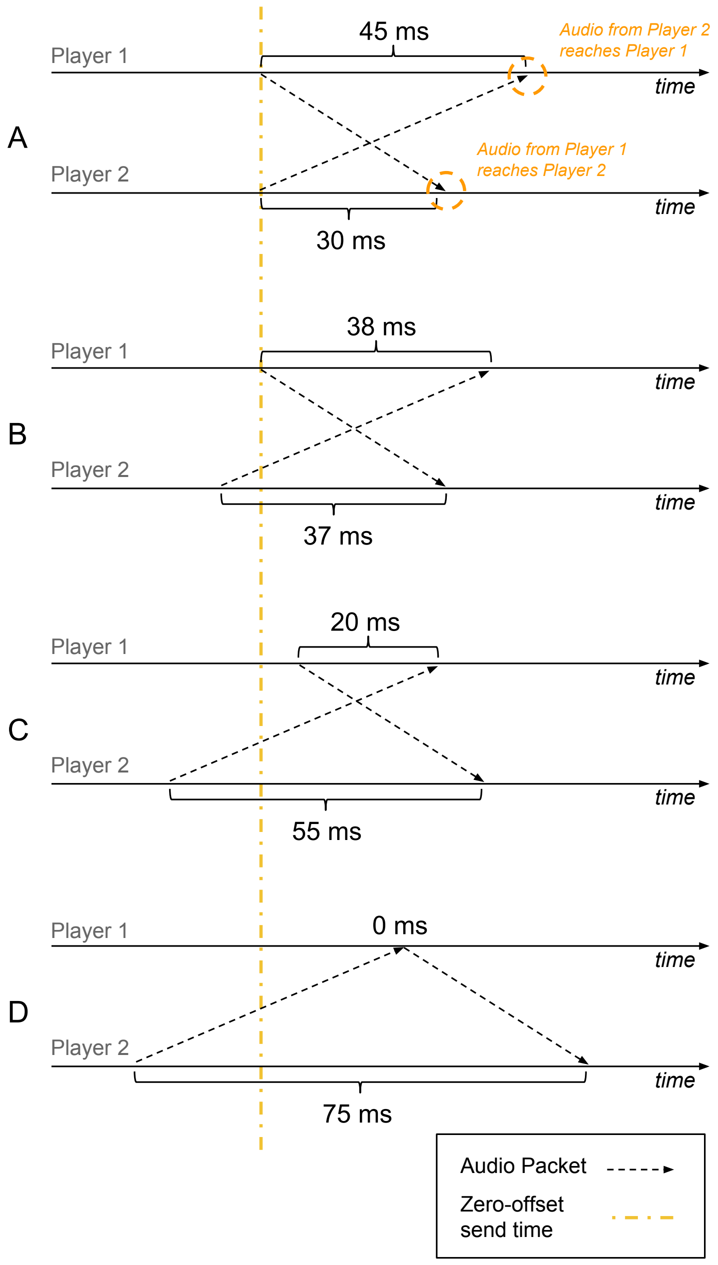


Figure 4.10: By adjusting the start time offset and the local audio delay, it is possible to distribute the latency between two participants. Figure A shows latencies when their start times are phase synchronized. B shows an even distribution of latency by starting Player 2 early. C shows a scenario where Player 1 has a short enough latency to play a live instrument in time with the incoming audio, yet still absorbs 20 ms of latency. D shows the classic leader-follower scenario where Player 1 hears no latency, and Player 2 hears the full latency.

4.4.3 Sometimes We Fall Out of Measure Phase

Measure phase synchronization alignment is less flexible when using sequencers via the Global Metronome. We assumed that players would always base their phrasing on the start and stop time of the sequencers. However, 2 times when the system was left running for a while, players lost track of the downbeat and began their phrases in the on the third beat instead of the first. In this scenario the experimenter had no way of re-orienting their sequencer to be phase aligned with their phrasing because they only had the option of starting and stopping on the "official" Global Metronome measures, which were described in Equation 4.2. This particular problem could be remedied by allowing the Global Metronome to start and stop at a finer granularity, but it highlights a limitation of the Global Metronome approach.

4.4.4 It Is Fun, But More Research is Needed

Our study revealed little valence data about the experience of the participants. We counted 23 utterances that we interpreted as frustrated during the equipment setup process. While playing music, during the 2 sessions that used JamKazam there were 8 expressions of frustration over 45 minutes of playing. After switching to JackTrip for 4 session, there were 8 over 1.5 hours of playing. For the 2 JamKazam sessions, the average answer to the survey question "How musically or socially connected did you feel with the other player(s)?" was 8 out of 10, when using JackTrip, over 4 sessions the average was 10 out of 10. Our assessment of the playing paradigm is that it is enjoyable, but more research is needed to completely answer the question. In particular, the questions that can be asked are: What aspects of this method of playing are enjoyable? In what ways must a musician alter their playing style to playing this manner? How irritating are those adjustments?

4.4.5 Technical Barriers and Problems Remain

There are two main categories of challenges when using a setup like the one we have described. The first is troubleshooting timing problems and the second is troubleshooting audio routing problems.

Timing problems can be frustrating to solve under latency conditions, especially if it is unclear how the one-way latency relates to the round trip latency. When using the variable latency JamKazam it was extremely difficult to determine whether it was the audio that was drifting or the PIGMI that was drifting. To troubleshoot this we used the PIGMI's ability to output audio ticks, and check their alignment with the attached drum machines.

Audio routing was the most time consuming barrier to setting up this system. Even a relatively simple routing—such as sending an audio stream to multiple destinations—was difficult to troubleshoot because it was often unclear whether audio was in fact being sent where we intended to it be sent. The routing problem was exacerbated by the vague routing interfaces provided by the most popular Jack routing programs JackPilot and qjackctl which are bundled with the Jack Audio Connection Kit. The routing would be much easier if there were audio level meters for each of the audio paths, which are represented by solid lines in Figure 4.2.

Installing Jack and audio quality problems got in the way of P1 and P3. P1 had problems with randomly changing playback frequency. P3 never got Jack working.

4.4.6 GPS Helps Troubleshoot

The main advantage of GPS is ground truth. Troubleshooting timing problems on the Internet can be very challenging because of the multiple sources of error. These potentially include, clock synchronization error, changing network congestion, sequencer error, operating system slowdown, software. Using GPS removes one of the most likely sources of error, and provides a rock solid timer against which to check other processes. The ticking sound produced via PIGMI GPIO pin was a useful tool when trying to troubleshoot timing error.

4.5 Conclusion

In this chapter we have presented findings from studies with three musicians performed as an exploration into the use of the Global Metronome for live performance. We have shown that it is a useful tool for tempo synchronization over long distances. We have also shown that in certain cases where latency is relatively constant and symmetric, it is possible to use traditional clock synchronization methods instead of GPS. In our experience, though, the highly synchronized clocks of the PIGMIs greatly simplify the process of troubleshooting network problems. We have shown that playing sequenced music over the Internet is possible and enjoyable, but that more research is needed to determine just how viable it is as a performance paradigm.

5 MalLO: A Distributed, Synchronized Musical Instrument

5.1 Introduction

An alternate approach to enabling rhythmically synchronized playing in spite of latency is via prediction. If we can predict the notes that a musician will play, we can send that information over the Internet before they are played, and schedule them to by synthesized in all locations at same time. In this chapter we present MalLo (short for *Mal*let *Loc*ator), a new musical instrument that uses prediction to send note data across a network *before* the note is played so that it can be scheduled and synthesized at both the sending and receiving location at the *same time*.

MalLo is an example of a new category of musical instrument that is specifically designed to enable natural, latency-free collaborations between musicians. These instruments are distributed, in the sense that they both exist and generate audio in many locations, as well as synchronized, in the sense that audio is generated and heard in nearly the same moment in all locations. We demonstrate in this chapter that building such *distributed, synchronized musical instruments* (DSMIs) is achievable, and we present the implementation details of our approach alongside an evaluation of its effectiveness.

To enable this type of prediction, we capitalize on the relatively long physical travel of malleted percussion instruments (e.g. vibraphone, drum kit) gestures which gives us time to predict when a note will occur. We chose this gesture because it has a number of useful characteristics. First, people easily understand the act of striking a surface with a stick; the motion feels nearly innate. Second, striking gestures are surprisingly predictable; it is easy to build a model of the gesture and predict the time of impact and impact velocity. Third, despite the length and predictability of a striking gesture, they are rhythmically accurate. Finally, they can be used for pitched music as well as rhythmic music. All we have to do is expand the number of striking surfaces and we can play melodies and chords.

In this chapter we

1. Present a study to determine the sensing requirements for MalLo

2. Describe the design of MalLo
3. Show that MalLo is capable of playing a note that is nearly synchronized (within 30 ms) with an actual mallet strike

Our work in this chapter was inspired by Sarkar and Vercoe who created a system that uses pattern recognition to eliminate network latency (Sarkar & Vercoe, 2007). In their approach, two remote tabla players perform together. Each end of the connection employs a software system that has learned a collection of canonical tabla beats. The system classifies the pattern that the local musician is playing and sends this information to the remote host, which then synthesizes a version of the pattern to the remote player. Because tabla patterns are highly standardized, it is likely that the receiver hears a close approximation to what is actually being performed. The spirit of the players' intention is communicated and the artists experience playing with one another in synchrony. Our method builds on this idea of prediction. However, instead of predicting with pattern granularity, we predict each individual note.

MalLo also builds on the work of Dahl (Dahl, 2011), who showed that the path of a percussion mallet head is sufficiently predictable that a DSMI could be built around it. The swing gesture of a mallet is long (on the scale of 40-110 ms) and smooth, as shown in Figure 5.3. This smoothness stems from the purpose of the gesture: to efficiently build kinetic energy to convert into audio, via a vibrating surface.

Note that DSMIs are made possible by the fact that a musician no longer needs to manually excite a surface in order to generate sound. Instead, we use synthesis algorithms generate musically useful audio. This decoupling is essential, because it allows us to begin the transmission process without waiting for sound to be generated locally. Thus, when con-

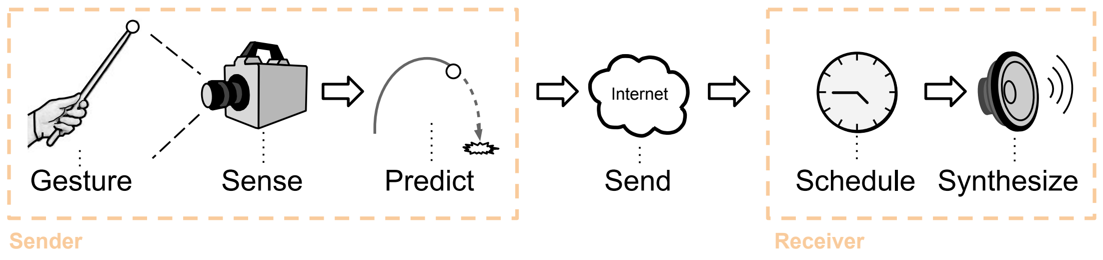


Figure 5.1: MalLo is an example of a *distributed, synchronized musical instrument* (DSMI). The instrument is composed of various parts that exist in different locations. It uses prediction in order to synthesize the same audio in both a sender's and receiver's location at the same moment. It capitalizes on the predictable path of a percussion mallet in order to accomplish this goal.

structing a DSMI, we seek to design an instrument with *predictable* properties rather than particular acoustic properties. Two crucial aspects of predictability are the *anticipation time* (i.e., how far in advance it can be made) and the prediction accuracy (i.e., how different is the predicted time from the time of the actual note event). Making predictions earlier in advance can mask longer network delays (which generally correspond to longer distances), and making more accurate predictions allows more authentic collaboration.

5.2 Initial Study

In order to gauge the viability of this approach we performed an initial study using an ultra-high-speed camera to capture image of a percussion mallet in motion. Our goal was to gain insight into how fast a sensor we might need, and how the amount of prediction error relates to sensor speed.

5.2.1 Recording Setup

To capture mallet motion, we used a Photron Fastcam SA3 grayscale camera running at 500 frames per second. The resolution is 512×512 pixels. A wooden striking surface was

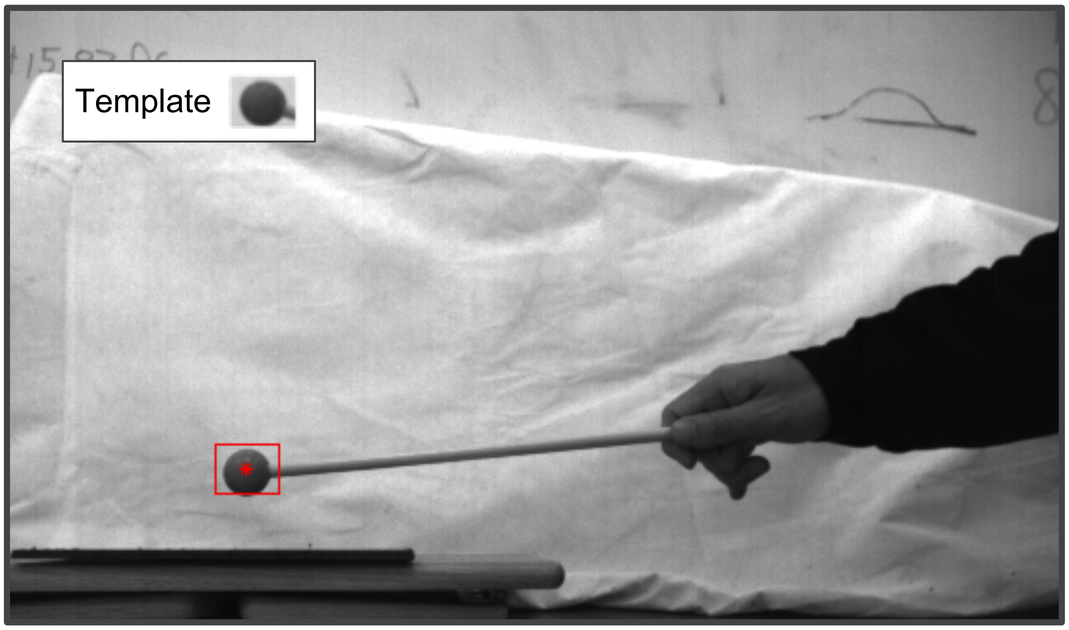


Figure 5.2: A single image from the high speed camera. Inset is the template used to track the mallet head. Best match is indicated in red.

oriented such that mallet contact with the surface could be easily identified by examining mallet height, as seen in Figure 5.2.

The mallet head was tracked using a template-matching algorithm, which takes as input a *template* image of a mallet head, and a *source* image within which to locate the mallet head (Szeliski, 2010). The algorithm computes the normalized cross-correlation distance between the template and each position on the source image.

One of the researchers played and recorded recorded two patterns, a 4/4 quarter note pattern at 120 bpm with an accent on the first beat of each measure (consisting of 24 notes), and a syncopated rhythm at 180 bpm (consisting of 34 notes). The syncopated rhythm included quarter and eighth notes. These two rhythms were chosen to loosely represent two rhythmic extremes, one relatively slow and simple, the other fast and more complicated.

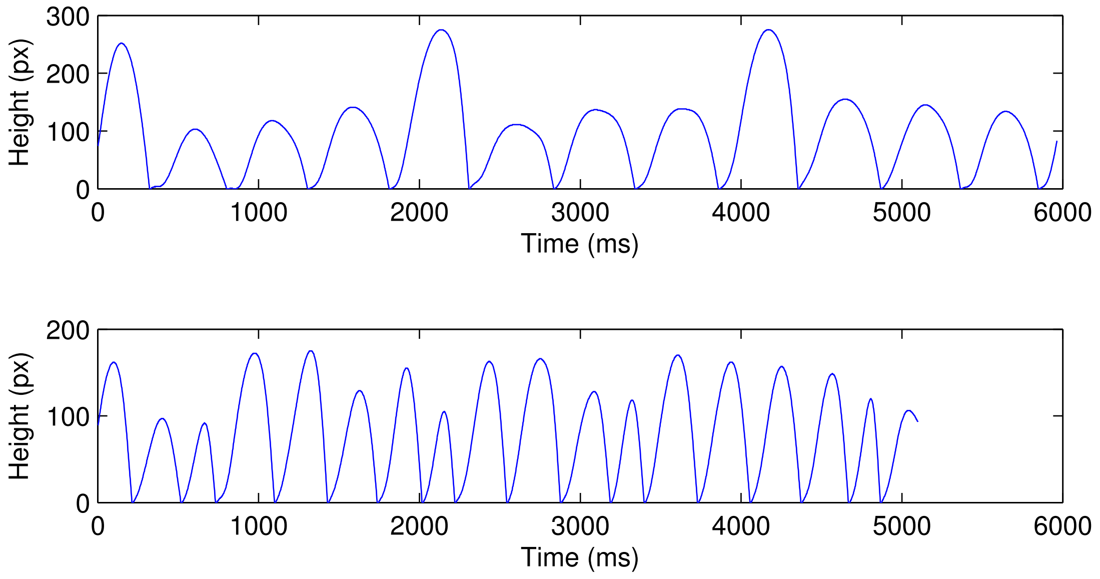


Figure 5.3: Mallet height over time, as output by our computer vision tracking algorithm, for quarter notes at 120 bpm (top) and syncopated notes at 180 bpm (bottom).

This yielded 11329 images. Examples of the tracking algorithm's raw, unsmoothed height output appear in Figure 5.3.

5.2.2 Evaluation

We divided the recording into 58 individual strikes, called *cycles* in this chapter. We chose a simple prediction algorithm, described shortly, that predicts strike time by fitting a polynomial to a cycle during the descent. Figure 5.4 illustrates one such single cycle in blue: the dashed red line is the predicted mallet path, and yellow circles represent the data fed to the polynomial-fitting algorithm. Here, the algorithm predicts the strike time to be 8 ms before the strike actually occurred.

We conducted a sequence of tests evaluating the accuracy of predicted *strike time* and *strike velocity*, as well as the number of missed notes, as we varied (1) the *anticipation time*

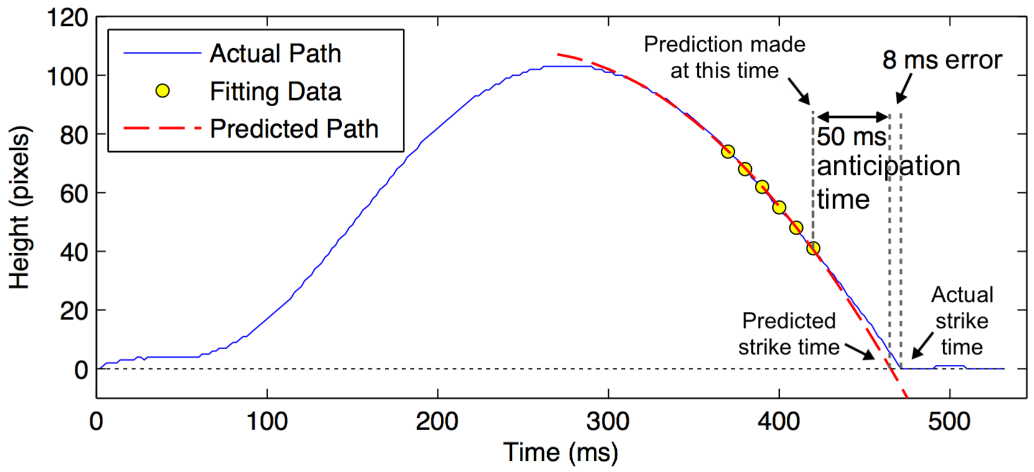


Figure 5.4: Mallet cycle fitted with path prediction. The final prediction is computed 50 ms before the predicted strike, based on the data (yellow, 100 Hz). Prediction error is 8 ms.

(i.e., how far in advance of the strike to commit to a prediction, and send the prediction over the network to a receiver), and (2) the *sample rate* of the camera data. Tests employed anticipation times of 10, 30, 40, and 50 ms. The sampling rates tested were 60, 100, 250, and 500 Hz. We chose these sampling rates because they correspond roughly to different cost levels of commercial cameras. The dataset for each sample rate was constructed by downsampling from the original 500 Hz dataset.

Our prediction algorithm works as follows: For each new mallet height sample, test to see whether the peak of the cycle has passed and the mallet is descending. If so, fit a quadratic using the samples from the previous 30 ms. The predicted strike time is simply the largest root (zero-crossing) of this polynomial. Next, we determine if there is sufficient time between the current time and the predicted strike time to take another sample (i.e., wait for another frame of video to arrive and be processed), to produce an improved prediction. If so, repeat the above process. If not, we “send” the event over the network to the receiver. (In our prototype used for this evaluation, the system does not actually send the event; the prediction is simply recorded for later analysis.)

In our work that quadratic polynomials perform better in this context than either linear or cubic predictors. To produce a ground-truth estimate of the velocity at the time of contact, we also fit a quadratic curve to samples from the 40 ms preceding the strike up to the strike time. To estimate the ground-truth velocity we compute the first derivative of the quadratic curve at the strike time.

Desired Accuracy

For this initial test, we wanted to gain at least 30 ms of *anticipation time* (time between the sending of the prediction and the actual strike). This is roughly equal to the average network latency between New York City and Chicago, or Denver and San Diego (Ano, 2013).

We aim for the prediction error to be within 20 ms of the *actual* strike, because perceptual studies have shown that humans cannot perceive ordering in sounds that occur within 20 ms of each other (Broadbent & Ladefoged, 1959). (Broadbent and Ladefoged show that while a listener may be able to perceive the existence of two sound onsets, they cannot tell which one occurred first.)

Likewise, we aim for the sound pressure level (SPL) of the synthesized strike to be within 3 dB of the actual strike. This amount of error is perceivable, but loudness is not as vital a characteristic as timing for rhythmic synchronization. We use a simplified model for estimating the discrepancy between loudness of the synthesized and actual strikes, based on the Hertz Contact Model (Hertz, 1881), and on empirical testing of how commercial synthesizers respond to MIDI velocity changes (Dannenberg, 2006). Specifically, as shown by Dannenberg, we assume peak RMS varies with the square of impact velocity. We can then express a perceptually-relevant, if overly simple, estimate of the error in decibels:

$$L_{error} = 20\log_{10}\left[\left(\frac{V_p}{V_m}\right)^2\right] \tag{5.1}$$

V_p is the predicted velocity, and V_m is the measured velocity. L_{error} is the difference in SPL between the predicted and actual note, in decibels.

Missed Notes

Occasionally, the prediction algorithm determines that there is enough time to collect another sample before making its final prediction, only to find that when the next prediction is made, the desired anticipation time cannot be satisfied. The cutoff time has passed, and were a note sent it would arrive too late or too with too much error. For our evaluation, we record these as *missed notes*.

5.2.3 Results and Discussion

This section evaluates the effectiveness of this approach under varying conditions, using the dataset of mallet heights described earlier.

Figure 5.5 shows the empirical cumulative distribution of the errors in strike time prediction for various levels of anticipation times, at a sampling rate of 500 Hz. For 10, 30, and 40 ms, 100% of the measured errors fall within the desired 20 ms. Additionally, over 80% of the errors for these times are under 5 ms. The SPL errors also fall within the desired range of error. For anticipation times of 10, 30, and 40 ms, 100% of the error is less than the target of 3 dB. Additionally, over 90% of the errors for these times are under 2 dB.

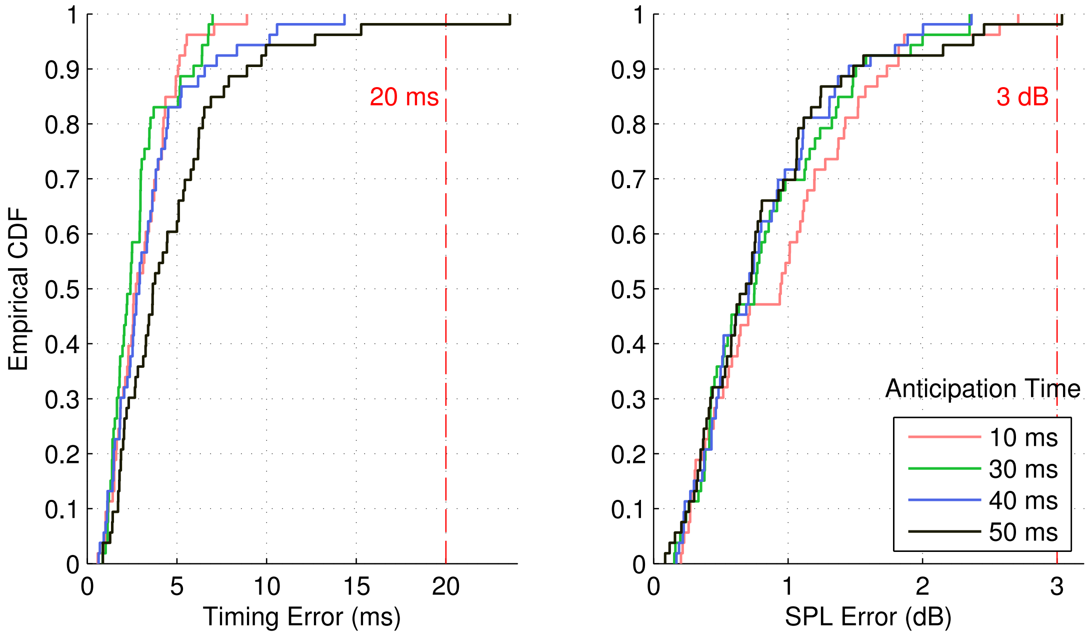


Figure 5.5: Varying Anticipation Time. Empirical cumulative distribution function of timing and sound pressure level errors while varying anticipation time. Sampling: 500 Hz.

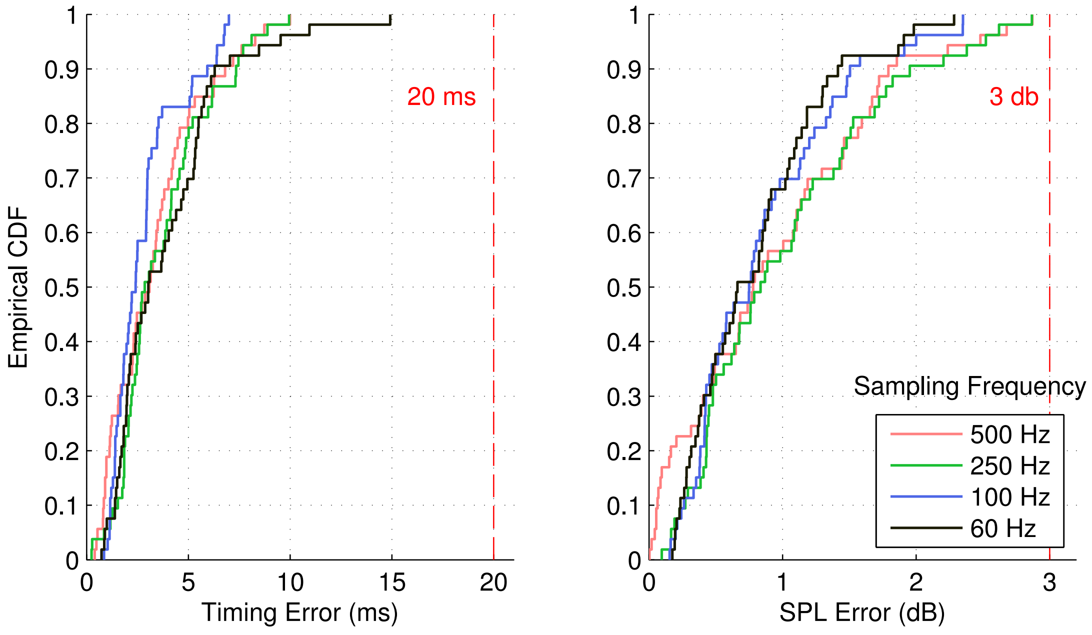


Figure 5.6: Varying Camera Sampling Rate. Empirical cumulative distribution function of timing and SPL errors while varying sampling frequency. Anticipation time: 30 ms.

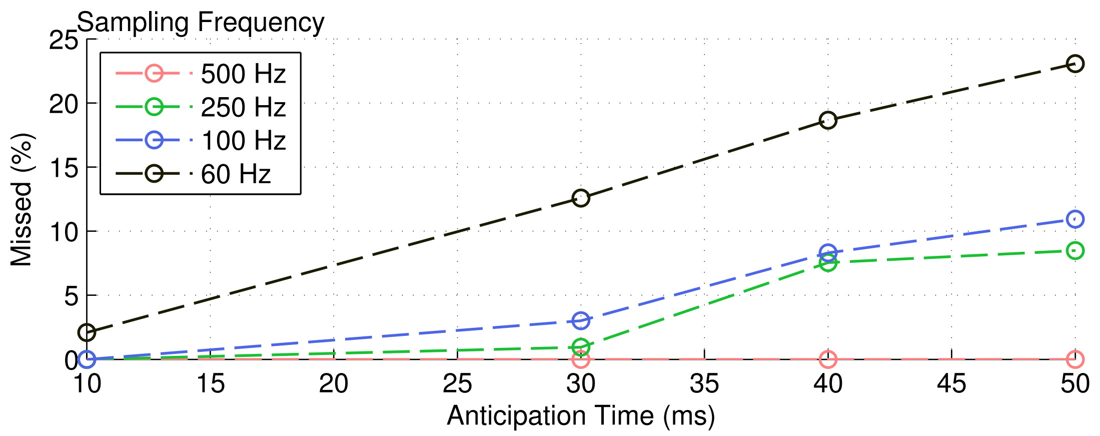


Figure 5.7: Missed notes. Fraction of notes missed for various required anticipation times and camera sampling rates.

Varying Camera Sampling Rate

Figure 5.6 shows the effect of sampling rate on prediction accuracy using an anticipation time of 30 ms. All frequencies performed roughly equally well. Occasionally the lower frequencies exhibit lower error than the higher frequencies. This is because troublesome cycles that are difficult to predict at the higher frequency ranges become impossible to predict at lower frequencies. They are recorded as missed notes, and are not included in the CDF.

Missed Notes

Figure 5.7 shows the result of varying the sampling rate and the anticipation time as they impact number of missed notes. At a sampling rate of 500 Hz, there are no missed notes at any anticipation time. Lower sampling rates increase the percentage of missed notes, as do higher anticipation times. We see a sweet spot at 250 Hz (shown in green) and 30 ms where the error rate is 0.97%. This means that for roughly every 100 notes played, 1 note must either be dropped or transmitted knowing it is possible that it will arrive late enough to be perceived as late.

Limits and Timing

Faster tempos, shorter note lengths, and lower sampling rates all have a similar effect of reducing the number of samples in a cycle. Our experiments with the number of sample points used in curve fitting suggested that at least 5 points were necessary to produce an accurate prediction. With these limitations in mind, at a tempo of 120 bpm, using a sampling rate of 250 Hz, the fastest note that can be played is a 16th note. Additionally, the time from the vertical peak of the swing to the strike time must be less than the sum of the anticipation time and the prediction computation time. If it is longer, the algorithm can't possibly predict a strike. Furthermore, if an artist stops a downward swing after the predicted note has been sent to the receiver, then a false note will sound.

Computation Time

In order to further determine the plausibility of our proposed approach, we analyzed the computation time taken by the various steps of the prediction process. The computationally significant operations that must be performed on each image are (1) locating the mallet head via template matching and (2) making a new path prediction via least squares regression.

The complexity of the template matching algorithm is $O(mn)$ where m and n are the number of pixels in the template and source image, respectively. Using the 512×512 pixel source image, the operation takes roughly 112 ms (in Matlab, using a 2.3 GHz Core i5 Duo processor), which is too long. Therefore, we constrained the search area to a smaller region around the last known location of the mallet head. Searching a 64×64 area takes roughly 1.4 ms. This approach works extremely well for higher sampling rates where the mallet head moves no more than 8 pixels per sample. However, at 60 Hz, the mallet head in our dataset moved a maximum of 75 pixels. The size of the template image is 24×31 pixels. This implies that at 60 Hz, we need a search area of at least 160×160 pixels, which takes roughly 9 ms to search.

The least-squares algorithm used to predict the path algorithm is quite fast, taking roughly 1 ms. We varied the number of example points between 5 and 100 and found the fitting time to be empirically constant with regards to the number of input points. To this we add a 7 ms estimate of latency for a high performance motion-capture camera.

In summary, for each image captured, our Matlab-based motion tracking and prediction system required 17 ms of computation time. We have included this estimate in our prediction model. This means that in figures 5.7, 5.5, and 5.6, when the anticipation time of 10, 30, 40, or 50 ms is listed, the prediction was made 27, 47, 57 and 67 ms before the predicted strike time, to allow 17 ms before "sending" the prediction across the network.

5.2.4 Conclusion of the Initial Study

Our initial study implies that with a sensor frame rate of 250 Hz we could predict notes before they occur with low enough error (< 20 milliseconds timing error and < 3 decibels amplitude error) to be useful for players to perform rhythmically synchronized music with one another over the Internet. The error rate increases with higher latencies and lower frame rates, and the minimum recommended frame rate was determined to be 250 Hz.

5.3 MalLo: Proof of Concept

In this section we describe the fully-functioning MalLo system[*] MalLo attacks the problem of network latency in two ways. First, it uses prediction to reduce the overall latency. Prediction is performed by a *Sender*, a process that senses the percussion gesture, and sends the prediction (e.g., across the Internet) to one or more *Receivers*. A Receiver is a process that receives the messages and schedules audio to be played. Second, MalLo uses an overlay multipath forwarding network to reduce the amount of latency variation. In this section we will describe the design of MalLo, then we will test MalLo under various degrees of real-world latency. Finally, we will test the effectiveness of the forwarding network at reducing latency variation.

5.3.1 MalLo Construction

MalLo is made up of 3 parts: (1) the sensing mechanism, (2) the prediction algorithm, which determines how the mallet path is interpreted by both the Sender and Receiver, and (3) a multipath overlay network (Andersen et al., 2001). The implementation of each component is described here, and the components are evaluated in Section 5.3.6.

We built two versions of MalLo, one using a high-speed RGB camera and one using the Leap Motion sensor (Figure 5.8). The high-speed RGB version is more accurate and was used for all tests in Section 5.3.6. The Leap Motion version was built as an underpowered-yet-usable version of Mallo, which we used for our usability tests and performances. We

[*]The fully-functioning MalLo system was developed in collaboration with Zeyu Jin. All text in this chapter was written by the thesis author. The concept of the distributed, synchronized musical instrument described in Section 5.1 is the author's. The prediction algorithms on pages 98 and 99, and the multipath forwarding strategy were conceived of jointly. The forwarding server software and Leap Motion version of MalLo were implemented by Zeyu Jin. The data analysis was done jointly.

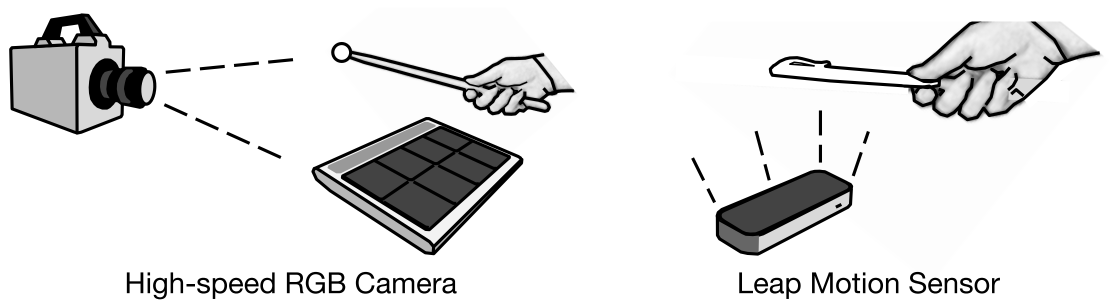


Figure 5.8: Two versions of MalLo. The high-speed RBG version uses a 200 fps camera, OpenCV to track the mallet head, and a MIDI drum pad to collect timing ground truth. The Leap Motion version version has no striking surface, and uses a yellow highlighter pen as a mallet. It is affordable and portable, but it has lower accuracy and higher latency.

describe both here to emphasize that MalLo can be implemented using a variety of hardware.

5.3.2 High-Speed RGB Camera Version

The RGB camera version of MalLo has low latency and tight timing synchronization. The host for this Sender system—used in our tests—is a Linux host running Ubuntu 12.04, with a dual-core 2.40 GHz Core 2 Duo processor. The camera is a Point Grey GRAS-03K2C-C FireWire 800 camera. It was chosen because it can isochronously transfer images with low latency (6 ms) from the camera to computer. Furthermore, the 1394 bus clock allows us to estimate shutter times with high accuracy (< 1 ms). It has a frame rate of 200 fps, a resolution of 640×480 pixels (turned sideways), and a global shutter to reduce motion blur. These specifications roughly meet the minimum requirements for tracking percussion mallets shown in our previous study.

The mallet head is tracked using OpenCV 2 libraries (Bradski & Kaehler, 2008), via the template matching algorithm (Szeliski, 2010). This search algorithm is the most computationally expensive operation in our pipeline. A typical search time is 5–15 ms (if the mallet is found to be within 50 pixels of the last known location) bringing the total processing time to 11–21 ms per frame.

5.3.3 Leap Motion Version

The Leap Motion is an inexpensive and portable sensor that collects images using IR illumination and multiple cameras at a rate of 120 fps, and it has a transfer latency of roughly 20 ms (Weichert et al., 2013). The device ships with the ability to track the tip of a stick, but the algorithm is proprietary. In general, the Leap Motion delivered noisier data than the RGB camera. From this point on in this chapter, all references to MalLo refer to the high-speed RGB version.

5.3.4 Prediction, Transmission, and Playback

In this section we describe the prediction and scheduling algorithms that MalLo uses. For each new mallet height h_i (one per frame) sample, a Sender performs a new prediction. If the mallet is descending, the Sender fits a second degree polynomial (quadratic regression) to the last k heights (7 in this section) and solves for the zero crossing, which is the predicted impact time, p_i. The velocity v_i at impact time is computed as the derivative of the polynomial at p_i, in pixels per second. This curve-fitting approach was shown in the preliminary study section in Figure 5.4. Finally, p_i and v_i are sent in a message to the Receiver via the multipath forwarding network described in Section 5.3.5. The time predictions are sent in absolute time, which means that Sender and Receiver must have system clocks that are synchronized to a high degree ($<$ 1 millisecond clock drift). The sending algorithm is shown in

Algorithm 1.

The new algorithm improves on the prediction method used in the initial tests in the way it deals with network latency, and is made possible by synchronizing the clocks of Sender and Receiver (e.g., via a PIGMI). In the original method, the Sender transmits a *single* message per strike, sent once the Sender has determined that a strike is imminent and the cannot wait any longer due to its estimate of the network latency. In our preliminary tests we assumed constant latency. However in reality latency is not constant. If the Sender estimates the network latency incorrectly, this can cause a prediction to be sent too early or too late, which can increase error or cause a prediction to arrive too late to be played. The new method allows the Receiver to determine which of the predictions to act on. This, in essence, gives us a perfect knowledge of the per-packet latency. To see a performance comparison between the old and new algorithms see Section 5.3.10.

For each new message received, the Receiver determines if the message is newer than the last message received, and whether a user-specified inter-note wait time T_{min} has passed. If so, the message is processed. If it is an "unschedule" message, then the currently scheduled note from that Sender is removed from the scheduling queue. This is a standard scheduling method for realtime systems. If it is a prediction message, then a note event is scheduled at time p_i with velocity v_i. The Receiver continues to replace older predictions with newer predictions until one of the predictions is played. This is based on the assumption that as the strike approaches, newer predictions have lower error than older ones. In practice we found this to be true; each new prediction is almost always better than the previous. The details of the receiving/scheduling algorithm are shown in Algorithm 2.

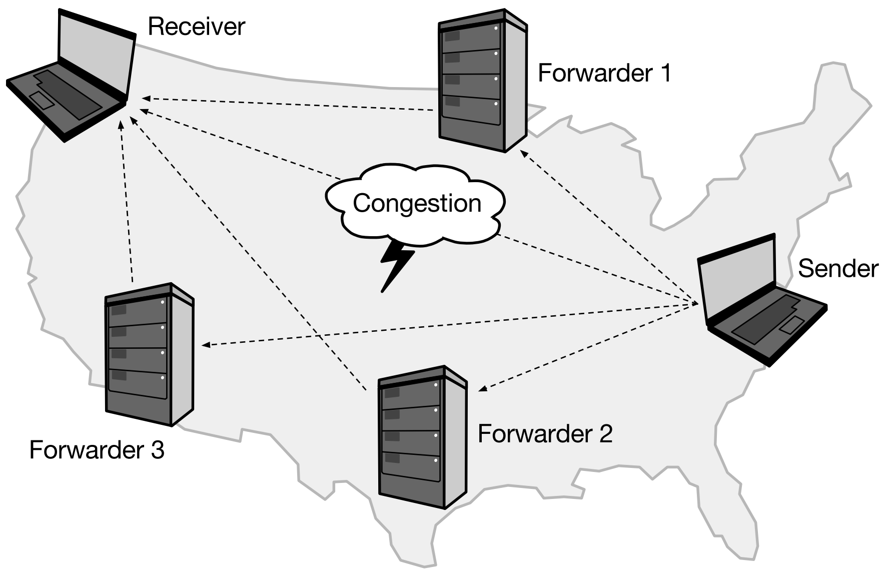


Figure 5.9: To reduce latency peaks, messages are sent along multiple simultaneous paths by sending both to the Receiver and to forwarding servers, in an attempt to route around local network congestion.

5.3.5 Multipath Forwarding Strategy

MalLo sends predictions from one location to another using a *multipath forwarding* approach, which is essentially an overlay network (Andersen et al., 2001). This aspect of the instrument addresses the problem that message latency on the Internet varies based on network traffic. By sending duplicate messages over many different network paths at once, MalLo can avoid moments of high latency that are associated with bursts of Internet congestion. Figure 5.9 illustrates these servers, placed in multiple strategic geographic location so as to create diverse routes through the Internet. We show that our method reduces peak latencies, thereby increasing overall prediction accuracy and robustness.

The overlay network consists of a series of servers throughout the Internet, hosted on Amazon EC2 and other commercial hosting companies. For each prediction to be sent, the Sender sends one copy directly to the Receiver, as well as one copy to each of the forwarding servers. They, in turn, forward the message to the Receiver. In this manner, the redundancy helps to route around congested areas of the Internet. Ideally the servers should be located so that their paths are different than the direct route, in an arrangement similar to Figure 5.9.

5.3.6 Experiments and Results

The tests in this section are aimed at determining how the MalLo approach would perform in over the Internet, under various latencies. Our goals are to determine if MalLo is effective at reducing (1) overall latency and (2) latency variation. To relate these tests to original study, the purpose of the initial study was determine the frame rate requirements for such a system (as well as the general viability of the approach). This series of test is closer to a real-world usage scenario and represents a true end-to-end system. The initial study assumed a fixed latency, while the latency in these tests varies based on the network conditions at the time of the test. In addition, while the initial test used only 2 tempos (120 and 180 bpm) here we use five. Finally, because of the success of the initial study under latencies up to 50 ms, we test up through a latency of 90 ms.

We evaluate the effectiveness of the prediction and multipath forwarding components of MalLo using a series of experiments over the Internet. Our first experiment investigates the timing accuracy of our prediction method under different latency conditions. The second experiment investigates the accuracy of our note velocity predictions. Our third examines the effectiveness of the multipath forwarding strategy in reducing latency spikes arising from congestion. Finally, we compare our current prediction system against the version used in Section 5.2.

5.3.7 Timing Accuracy

Our main goal is to show that prediction can allow a performer to play the same sound, synchronized, in multiple locations, in spite of real-world latency. Faster tempos and higher latency make percussive strikes more difficult to accurately predict, so we tested the system under different latency conditions and at different playing tempos.

We recorded a musician (one of the researchers) playing a syncopated, repeating pattern made up of quarter and eighth notes, at 60, 80, 100, 120, and 140 bpm for 20 seconds. The recordings resulted in a sequence of tuples containing a mallet height and a corresponding timestamp. We also used the MIDI drum pad to collect the corresponding timing and (but not velocity) ground truth. Each sequence has approximately 30 notes, with faster tempos having more.

To test the accuracy of our note timing predictions under different latency conditions, we used the height/timestamp tuples to generate timing predictions that were each sent to different servers around the world. Each server immediately redirected the message packets back to our lab, where they were received by a Receiver process running on the same computer as the Sender. (This allowed perfect clock synchronization between Sender and Receiver for our tests.) We consider the entire round trip time, to the server and back, as the tested latency. There were 6 servers in total corresponding with 21, 40, 48, 61, 78, and 90 ms of average latency. In general the latencies were stable, varying by ± 5 ms for each. However, at times we observed large momentary spikes in latency. (See Section 5.3.9 for more information about latency behavior.)

We compare the predicted note onset time with the timing of the ground truth. Based on Chafe's analysis of the musical consequences of different latencies between performers

Latency \ BPM	60	80	100	120	140
21.0 ms	100.0	100.0	100.0	100.0	100.0
40.0 ms	100.0	100.0	100.0	100.0	100.0
48.0 ms	100.0	100.0	100.0	100.0	100.0
61.0 ms	100.0	100.0	100.0	100.0	100.0
78.0 ms	99.8	99.6	99.9	99.6	99.6
90.0 ms	100.0	100.0	97.9	94.7	94.7

Table 5.1: The percentage of notes predicted within 30 ms of the ground truth when playing the pattern described in Section 5.3.6

(Chafe, 2012), we consider notes played by the Receiver within 30 ms of the ground truth as "ideal", notes within 30–50 ms of ground truth as "tolerable", and those beyond 50 ms as "missed". Table 5.1 shows the percentage of notes that were "ideal". The table also includes false positives in the count of "missed" notes. (False positives occur when the musician moves the mallet downwards but does not complete the motion with a strike and the Receiver does not receive an "unschedule" message in time.) Note that the system begins to break down with 88 ms of Internet latency, dropping 6 out of every 100 notes at 120 bpm. However, at 73.0 ms, the system accurately predicts at least 99.6% of notes over all tempi. 73 ms is roughly the time it takes for a network packet to travel from Berlin to Cairo, New York to Lima, Tel Aviv to Halifax, or Tokyo to Los Angeles (Reinheimer & Will, 2015).

For a closer look at the behavior of MalLo, Figure 5.10 shows the distributions of predictions as average latency is varied (by using different servers), for a pattern played at 120 bpm. Note that as latency increases, so does the variance of the predictions. Also note that for each average latency level, the extreme predictions are never more than half the amount of latency, which means that in its worst case MalLo effectively halved the amount of latency.

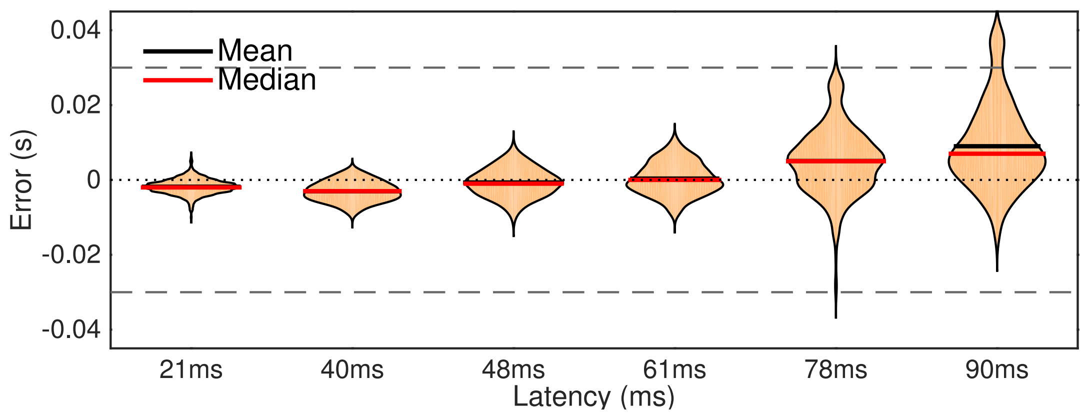


Figure 5.10: A pattern (described on page 88) played at 120 bpm, with varying average network latency, accomplished by sending note messages to various locations around the world. Higher latencies created more variance in prediction accuracy. At 80 ms latency 99.6% of predictions are within 30 ms of the ground truth.

As latency increases, the mean prediction becomes later, and a noticeable protrusion begins to form at the top of the distribution. This is due to the eighth notes in the pattern occurring too quickly to be predicted on time. However, although they are predicted late, they still fall within the "tolerable" range of 30–50 ms error

Figure 5.11 shows the accuracy of predictions at a latency of 80 ms at various tempos. We see the same behavior as Figure 5.10, with higher tempos showing higher variance. This is because higher tempos and higher latencies are roughly equivalent as far as the algorithm is concerned. A faster tempo means there is less time between notes. Higher latency requires predictions to be made earlier in order to send over the Internet in time. Both require that a prediction be made earlier in the stroke of the mallet, and if the prediction is made too early (e.g. before the mallet has descended long enough) there will not be enough data to predict accurately.

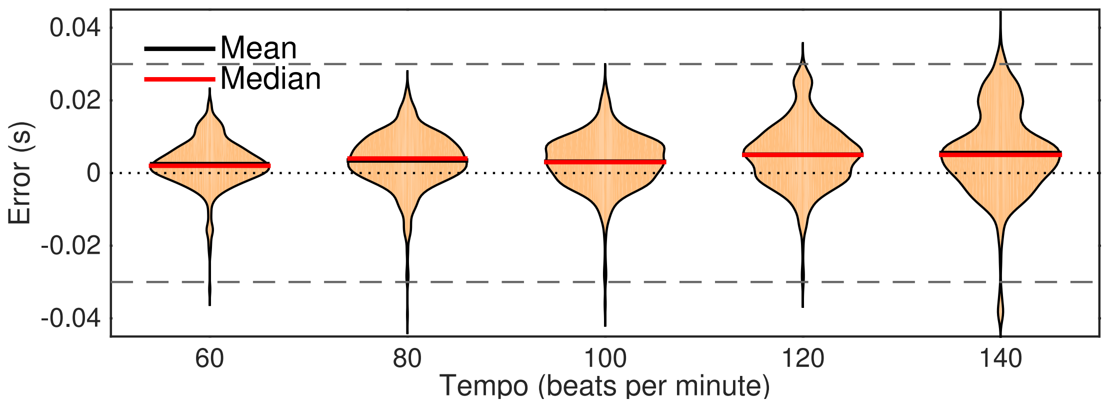


Figure 5.11: A pattern (described on page 88) played at varying tempos, over 77.5 ms of average latency. Higher tempo causes more variance, but in the worst case of 140 bpm, 99.8% of predictions are within 30 ms of ground truth.

5.3.8 Velocity Accuracy

To test velocity accuracy, we compare the predicted velocity of the mallet with the velocity of the mallet at the time of impact (as derived from camera tracking data). We did not compare the predicted velocity to the drum pad's MIDI velocity because we found the velocity sensitivity of the drum pad to be quite noisy, producing widely different MIDI velocities for similar strike velocities. Instead we used the same approach as in the initial study. The ground truth is computed using the final 7 samples before the actual strike, fitting a quadratic regression, and computing the derivative of the regression at the time of the strike. In contrast, the prediction's quadratic was fit using 7 samples taken at the time of the prediction (roughly 21, 40, 48, 61, 78 or 90 ms early depending on the real-time latency of the experiment). We used the same velocity error measure used in the preliminary study.

Figure 5.12 displays the results. Compared to the preliminary study, we found there was more variance in the velocity error. For latencies between 20 and 50 milliseconds, the error is centered around 0 dB, but where the preliminary study showed that the majority of ve-

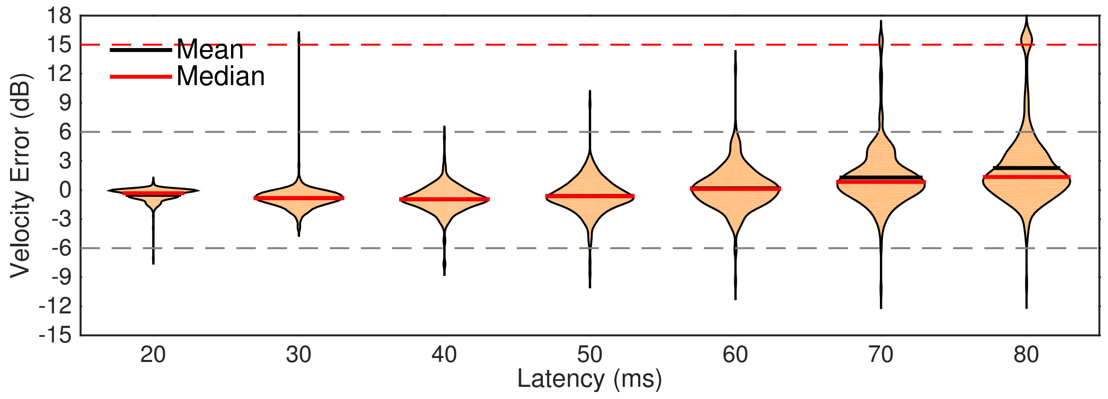


Figure 5.12: The difference between ground truth and predicted velocities (converted to probable sound pressure level when striking a rigid body). Performance is within an acceptable range up through 50 ms, but begins to degrade. At 80 ms, 5% of estimates are at least 15 dB louder than ground truth.

locity error was less than 3 dB of magnitude, the new study shows a fair number of error between 3-6 dB and outliers as large as 9 dB. For latencies between 60 to 80 milliseconds, the velocity error mean and median begin to become more positive. Performance degrades until 80 ms latency, where 5% of predictions are at least 15dB louder than the ground truth. Using our current prediction algorithm, velocity prediction is not as robust as timing prediction, but it is also not as important for synchronization.

5.3.9 Latency Peak Reduction

To test latency variation with and without the multipath overlay network, we sent a stream of packets from Sender to Receiver via the network of 4 forwarding servers located in Virginia, Texas, Georgia, and London. The Sender machine was located in New Jersey, USA, and the Receiver machine was located in Oregon.

On the Receiver's end, we measured and compared the one-way transmission time from Sender to Receiver for each path. The results of the latency measurements, with and with-

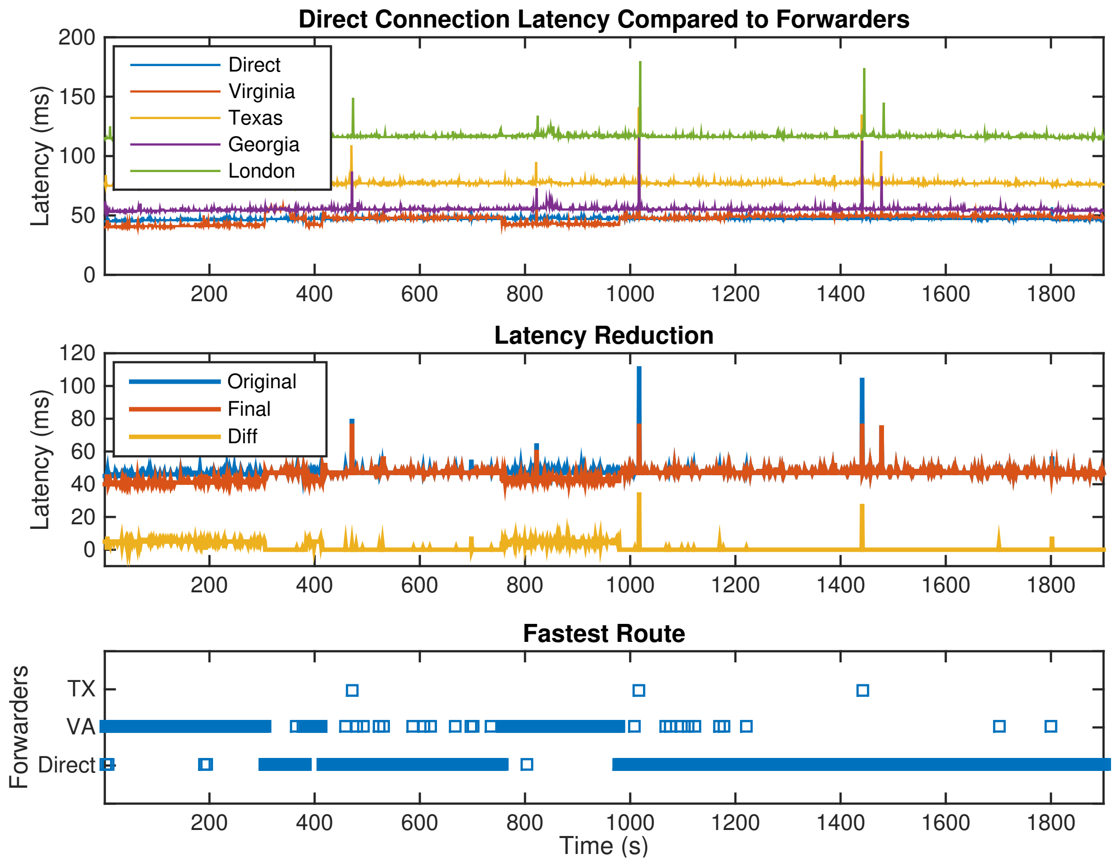


Figure 5.13: The multipath network helped to reduce latency peaks. The top plot shows measured latencies for different paths. The middle plot shows the original latency, the final latency, and the difference between them. Note that two large spikes around 1000 and 1500 seconds were reduced by roughly half. The lower plot shows the shortest route at each time point. The previously mentioned latency spikes were reduced by routing through Texas.

out the multipath network, are shown in Figure 5.13. (For this experiment we synchronized the sending and receiving servers' clocks using Network Time Protocol, which is only accurate within tens of milliseconds (Mills, 1991). But it is not crucial that the clocks be synchronized exactly, because we are interested in the relative latency between the different paths, not the absolute latency.)

This experiment shows the multipath system works to reduce latency peaks. Latency peaks may be bursts (which last on the scale of seconds or milliseconds), or they may be sustained (which can last several minutes, or even hours). Figure 5.13 shows that the multipath network is effective in reducing both types. At times the route through Virginia was faster than the direct route, but when congestion caused this path to slow down, the direct route became the preferred path. When in use, the Virginia route reduced latency by 5 ms, which can be enough time to allow an "unschedule" note event to arrive in time to cancel a false positive. The multipath system was also effective at preventing some burst latency. At around 1000 and 1500 seconds there are two major bursts that are reduced by almost half by routing through Texas. These two reductions kept all peaks under 80 ms, as opposed to greater than 100 ms. Finally, some bursts were not possible to alleviate. These occurred when the congestion was local to Sender or Receiver, or simply included all the paths.

5.3.10 Old Method vs. New Method

To compare the current algorithm to the previous method used in the initial study, we implemented a slightly-improved version of previous algorithm. It estimates the latency every 0.5 seconds by sending a timestamped packet from Receiver to Sender to Receiver and estimating the one-way packet latency. We then modeled the latency as a Gaussian distribution with the packet latency as the mean, and used the 95th percentile in the algorithm latency estimate. We used this conservative estimate in order to avoid sending messages too late.

We tested the relative effectiveness of both methods by running them simultaneously in the same manner as the experiment in Section 5.3.7. The results of this comparison are shown in Figure 5.14. At all levels of latency, the new algorithm did as well as or better than the old approach.

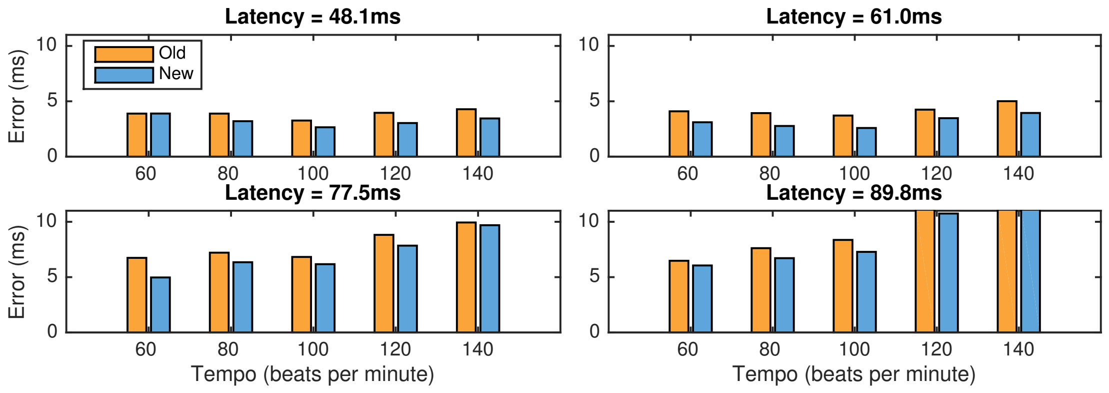


Figure 5.14: Our new prediction algorithm has a lower mean error in nearly every case than the method used in the initial study.

5.3.11 Discussion

Widening the Radius of Collaboration

The advantage of using a predictive instrument like MalLo is that it allows players to play with collaborators who are physically further away from them than is possible using traditional instruments. These experiments have shown that MalLo is capabable of nearly 80 ms of prediction, in spite of using a relatively primitive prediction algorithm. This means a considerable increase in players' radius of collaboration. For reference, in our experiments we found that the round trip latency from New York to San Francisco is roughly 80 ms. This means that this DSMI could easily synchronize audio between artists in each city by making up for 40 ms of one-way latency on each side. It also means that if there are latency spikes of up to 80 ms, MalLo can gracefully absorb them. Synchronized performances between major cities such as Amsterdam and Detroit (51 ms), London and Fez (47 ms), Hangzhou and Hyderabad (41 ms), and Tel Aviv and Varna (57 ms) are all theoretically possible (Reinheimer & Will, 2015) without reaching the limits of prediction. In our companion video we demonstrate a jam played over a latency of 90 ms, equivalent to the one-way latency between New York and Buenos Aires.

Tradeoffs and Challenges

MalLo is a new type of instrument, and one that requires a musician to practice it to gain proficiency. The basic movement is easy to understand, because it uses the metaphor of striking an object, but it is slightly different from a typical percussion instrument. This is likely to be the case with all DSMIs. In order to have sufficient time to predict reliably, it is useful to have longer, slightly slower movements than traditional instruments.

There are limits on how fast a musician can play using MalLo. The fastest playable note sequence depends on the amount of latency. In some cases the multipath forwarding approach will not protect against congestion. If the congestion is near the Sender or Receiver then it may not be possible to route around it.

5.3.12 Conclusion

In this chapter, we have shown how it is possible to use prediction in order to create a truly distributed, synchronized, musical instrument (DSMI) despite the latency inherent in the Internet. We described the implementation of one such instrument, MalLo, and showed that its timing prediction is within the perceptual bounds of synchrony for network one-way latencies up to 80 ms, and that its velocity prediction performs well for latencies below 50 ms. We created straightforward algorithms for both the sending and receiving parts of the DSMI. Finally, we presented a method for latency peak reduction, using a network of servers to achieve multipath transmission, based on overlay networks. These proved effective at reducing the effects long term network fluctuations on the order of 5ms, and some burst latency on the order of 40 ms.

Algorithm 1 Sender Prediction Algorithm

```
function SEND(command, timestamp, $p_i$, $v_i$)
    Combine arguments into message
    Send message to the Receiver and all forwarders
end function

// Let $H = \{h_{j-k+1}, ..., h_j\}$ and $T = \{t_{j-k+1}, ..., t_i\}$
// $H$ is a sequence of mallet heights
// $T$ is the sequence of corresponding timestamps
// $t_i$ is the time of the most recent height
// $k$ is the number of samples used in the linear model
// this function is called when a mallet sample is captured
function PREDICT($H$, $T$)
    if mallet head is ascending, (e.g. $h_j > h_{j-1}$) then
        SEND("unschedule", current_time, "",“");
    else
        Fit linear model $h(t) = \beta_0 + \beta_1 t + \beta_2 t^2$ on $(H,T)$;
        Solve for zero crossing time $h(p_i) = 0$
        // $p_i$ is the current predicted time
        $v_i \leftarrow h'(p_i)$
        // $v_i$ is the current predicted velocity
        if $p_i$ exists AND $p_i > t_i$ then
            SEND("schedule", current_time, $p_i$, $v_i$);
        else
            SEND("unschedule", current_time, "",“");
        end if
    end if
end function
```

Algorithm 2 Receiver Scheduling Algorithm

```
Global Variables:
    $t_1$, timestamp of the most recently received message
    $t_0$, time of the most recently played note event

// $T_{min}$ is the minimum time to wait between two notes
// This function is called when a message is received
function RECEIVE(path, timestamp, $p_i$, $v_i$)
    if timestamp > $t_1$ AND timestamp > $t_0 + T_{min}$ then
        $t_1$ = timestamp;
        if path == "unschedule" then
            Unschedule all events;
        else
            Schedule note event at time $p_i$ with velocity $v_i$;
            Schedule ($t_0 \leftarrow p_i$) to occur at time $p_i$;
        end if
    end if
end function
```

6 Usability of MalLo

6.1 Introduction

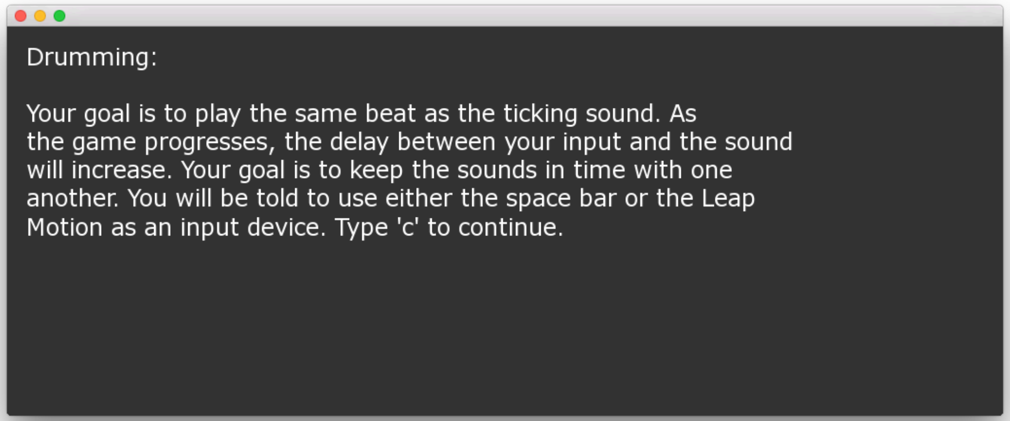
In this chapter we discuss our usability study for the physical controller part of the MalLo system. In the previous chapter we studied the sensor requirements and the general viability. In those studies gestures were played by a single player, recorded, and used as input data for the studies. In this chapter we investigate how a variety of participants interact with the system. Our goals in this study were to (1) determine whether MalLo is effective in enabling rhythmic synchronization when used by a variety of people (2) uncover problems, breakdowns, and shortcomings of the interface, (3) gather qualitative feedback on how enjoyable the interface is to work with.

To accomplish these goals we developed an experiment application in which the participant performs a drumming task (described in detail in Section 6.2) which involved playing along to a metronome under varying amounts of simulated latency. During the task we collected the timing of their inputs, which was analyzed against the the timing of the metronome in order to evaluate their rhythmic accuracy. We also collected observation notes during the task, noting the participant's questions, problems and actions. The noted observation data was coded and analyzed, along with a topical analysis in order to identify problems and gauge valence reactions to the interface. Next we will describe the experiment in detail.

6.2 Experiment Design

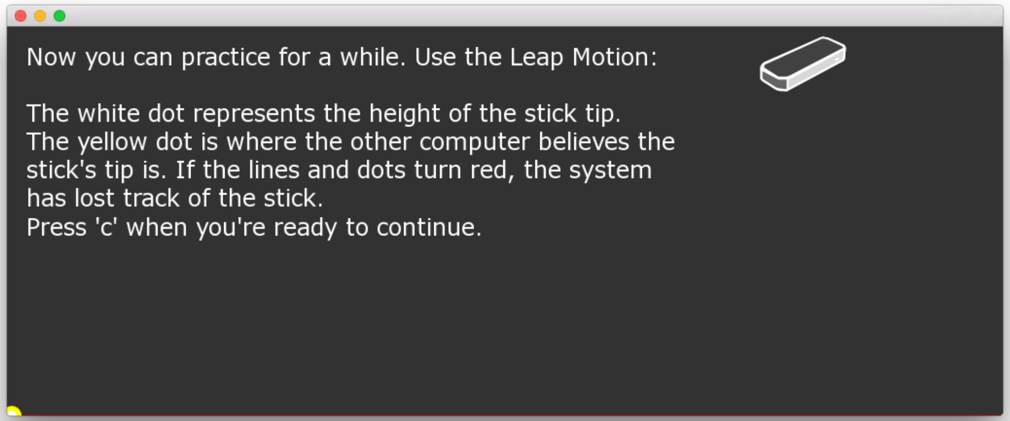
The study involves 8 trials of a "drumming" task, which was performed using the Leap Motion and the space bar (as a "control interface"), followed by a semi-structured interview and a survey. The various screens of the application are shown in Figures 6.1, 6.2, 6.3, and 6.4.

In each trial, the goal is for the participant to play a constant rhythm along with a metronome. During each trial, the user hears a ticking metronome sound at 100 beats per minute, and is instructed to play along, trying to match their playing to the metronome, exactly. They perform the task first with the space bar and then with the MalLo interface. The applica-


Drumming:

Your goal is to play the same beat as the ticking sound. As
the game progresses, the delay between your input and the sound
will increase. Your goal is to keep the sounds in time with one
another. You will be told to use either the space bar or the Leap
Motion as an input device. Type 'c' to continue.

Figure 6.1: The welcome screen of the drumming application.


Now you can practice for a while. Use the Leap Motion:

The white dot represents the height of the stick tip.
The yellow dot is where the other computer believes the
stick's tip is. If the lines and dots turn red, the system
has lost track of the stick.
Press 'c' when you're ready to continue.

Figure 6.2: Each trial first has a practice period. The interface to use is both stated in text and displayed as an image.

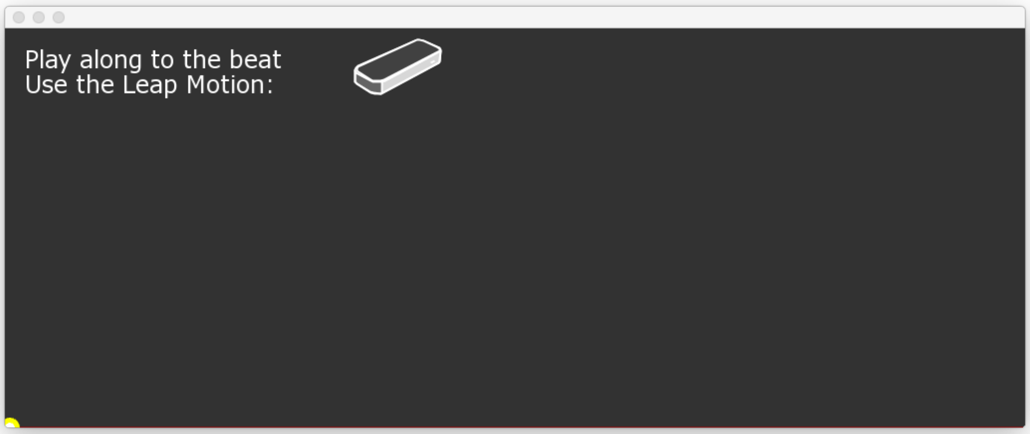


Figure 6.3: The test part of the task where the user is prompted to play along.

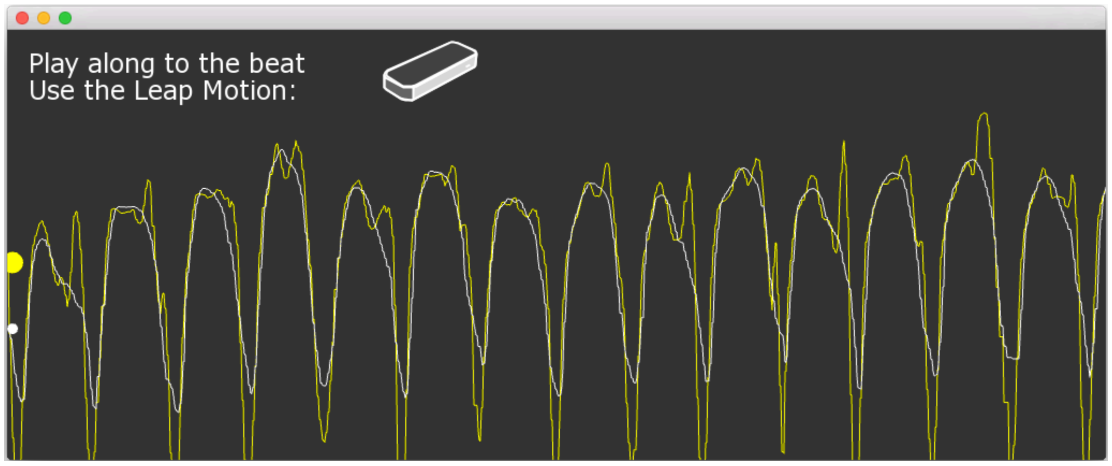


Figure 6.4: The test section after the user has played a number of notes. This visualization is also present in each practice portion.

tion simulates latency by introducing a delay between the time that their input is registered and when their drum sound is rendered into audio. We tested 4 latency levels: 30, 60, 90 and 120 ms. This resulted in a total of 8 trials per person. The order and combinations are shown in Table 6.1.

We deliberately opted against randomizing the order of interfaces. The motivation for this is that the space bar is a very well known and often used interface, while MalLo is not. By allowing the person to first use the space bar for each latency level it allowed them to acclimate to the latency level using a familiar interface before moving on to one that is more foreign. Additionally, we opted against randomizing the latency levels. This is largely because we are interested in how the choice of interface affects performance, not the amount of latency. Additionally, in the real world the user is not presented with a random latency level. In practice we assume that have time (probably lots of time) to acclimate to it before performing. Therefore we felt the more ecologically valid approach would be to present the latency levels in order.

Each trial begins with a practice period of unlimited time when the participant can "get used to" the current latency level and interface. During the practice session they hear a metronome ticking at 100 bpm and try to synchronize the audio of their drum to it. At will, they then proceeded to the "official" playing period which consists of 30 ticks at 100 bpm. Our software logs the timing of each strike during this official playing period as well as the practice period.

Each MalLo trial provides a visualization to aid the participant in their understanding of the system, shown in Figure 6.4. At the far left of the screen are two circles that move up and down with the drumstick. The small white circle indicates the current height of the tip of the drumstick, and the larger yellow circle indicates the location of the tip as seen by a hypothetical receiving computer (i.e., the predicted height for the current time, given the current amount of simulated delay). The participants were informed that when the yellow circle is at or below the bottom of the display, the drum sound plays. Previous heights of the mallet and the mallet prediction trail away to the right, forming a moving history of the heights.

The four levels of latency correspond roughly to notable levels found in the literature. 30 ms of latency is near the threshold when two percussive sounds can be perceived as separate (Broadbent & Ladefoged, 1959). At 60 ms latency is obvious, but can be compensated for by most people (Chew et al., 2005; Chafe et al., 2004). 90 ms of latency has been shown to cause tempo drag during rhythmic synchronization, while 120 ms of latency is beyond the threshold which causes tempo drag (Chew et al., 2005; Chafe et al., 2004).

The drum sound used was a recording of a conga, played via the built-in audio playback library in openFrameworks (openFrameworks, 2017), a C++ framework with which the experiment application was built. The metronome sound was a recording of a sidestick (i.e., a drum stick hitting the rim of a snare). To simulate latency in the drum sound while using MalLo we used the prediction algorithms described on page 98, placing a fixed delay between the Sender and Receiver processes. To simulate latency in the space bar interface, we added a delay between the time that the space bar was pressed down and the the time that the drum audio was played.

Trial #	Latency	Interface
1	30 ms	MalLo
2	30 ms	space bar
3	60 ms	MalLo
4	60 ms	space bar
5	90 ms	MalLo
6	90 ms	space bar
7	120 ms	MalLo
8	120 ms	space bar

Table 6.1: The drumming task employed 8 trials, with delay and interface combinations above.

	Weekly Gaming	Weekly Music	Music Years	Age
Median	10 m	0 m	15 yr	40 yr
SD	25.7 m	22.9 m	22.9 yr	3.9 yr

Table 6.2: Participants were asked how many minutes they spent gaming, playing music weekly, and how many years of music experience they had.

6.2.1 Participants

The twelve participants range in age from 21-43 and reported a variety of levels of both music and gaming experience. There were no highly competent percussionists used in this experiment, but one participant had percussion experience in grade school. There were 6 male and 6 female participants. Table 6.2 shows median and standard deviations for age, the number of minutes spent gaming each week, the number of minutes spent playing music each week, and the number of years experience playing music.

6.2.2 Experimental Protocol

Below is a description of the experimental protocol—the script used to perform the study for each of the participants. Quotes denote instructions from the experimenter.

Entrance

The participant was greeted and asked to sign the consent form. We sat them at a table in front of the computer which has the experiment interface on it. We then gave them an explanation of the experiment based very closely on the following script:

"We are testing alternative interfaces for music over the web. You will be will be playing a drumbeat along with another beat. You will be using two different input devices, one will be the space bar on your computer, and the other will be a leap motion. First, let me show you the input methods. The space bar is simply pressed, as you normally use it. To use the Leap Motion sensor you hold this stick and strike downwards, as if you are hitting a surface, but you don't make contact with it. To get the drum to sound, (referring to the visualization) make the yellow dot dip below the bottom of the screen. Here, you try it."

"The task is called drum playing. You will hear a simple rhythm, a constant tick that sounds like this: *tick, tick, tick*. Your goal is to play exactly the same pattern, so that the tick and drum sound occur at the same time. This will last for 30 seconds each. For each test we will vary some aspect of delay in the interface, meaning there will be a pause between when you hit the space bar (or swing the stick) and when the sound is played. Just to make this clear, we aren't testing your skill. We are testing the interface. Do the best that you can."

Running the Study

"OK, now we're running the experiment. You're going to see a practice round, followed by a 'real' round, then a practice round, then a real round, etc. The input device will always stay the same between the practice and real rounds, so you'll do a practice with the Leap, then a real round with the Leap, then a practice with the space bar, then a real round with the space bar. The instructions and picture on screen will always tell you what device to use. Do you have any questions? I'm going to sit back and watch and take notes. If you get confused about anything, you can stop and tell me, and we can re-start the round."

The participant then executed the task, while the experimenter noted the following details. (1) When using Leap Motion or space bar, what were the errors, breakdowns, confusions, frustrations, exclamations? (2) What questions were asked? (3) What emotive gestures did the participant make? What body language did they exhibit?

Post-Task Interview and Exit Survey

After the task was complete, the experimenter conducted a semi-structured interview based off the following set of questions.

- Did you have any problems with the spacebar?
- Did you have any problems with the mallet?
- At any point did you become frustrated?
- What was challenging about playing as the delay increased?
- Were there times when the delay was unnoticeable?
- I see that you had trouble getting _____ to activate at times. What was the problem? What approach did you use to resolve it? Did you a particular strategy?
- I noticed that you had trouble with _____ at one point, what was the problem there?
- Did you prefer the Leap Motion or the space bar? Why?

Finally, the participant was given an exit survey with the following questions.

- How often do you play video games, and for how many minutes?

- How many years of experience do you have playing music?
- Do you play musical instruments regularly? If so how often and for how many minutes?
- Do you play percussion instruments? If so, which?
- What is your age?
- What is your gender identification? (you may decline to answer)

6.2.3 Data Collected

The data collected during the task and interview includes:

1. Timing data when the metronome ticks were played (ground truth).
2. Timing data of when the drum sounds were played, these include the simulated latency.
3. The survey and observation data described in the previous section

6.2.4 Analysis Method

To analyze whether MalLo or the space bar led to differences in performance, our null hypothesis is that the median timing errors for each interface are equal. To compute the timing errors, we look at the 30 metronome ticks during the non-practice part of the experiment and compute the time difference between the tick and the moment the application receives user input (via the space bar or MalLo, with the simulated delay). The difference between them is the timing error. If the time to the nearest input is greater than 400 ms, we label that tick as "missed" and do not include it in the timing error analysis. We use two different error measures, the signed error and the unsigned error magnitudes. The median of the absolute error tells us about the overall amount of timing error, while the unsigned error lets us know how early or late the hits tend to be, with positive values corresponding to late hits and negative to early hits. We want the signed error to be near zero and the unsigned error to be as low as possible.

To evaluate the 8 conditions (listed in Table 6.1) we used a Mann-Whitney test (a non-parametric ANOVA alternative) chosen because the errors are not normally distributed (using the Kolmogorov–Smirnov test) and have different variances. It discards the ordered nature of the error data.

Here we list our goals once again and describe how we plan to accomplish them via our analysis.

1. Determine whether MalLo is effective in enabling rhythmic synchronization across many participants.

   Within the context of this task we define "enabling rhythmic synchronization" as having two parts. First, we want the timing of the participants' playing to more closely match the timing of the metronome ticks. Second, we want the median timing error to be centered around zero error (an equal number of early and late hits). The motivation behind this second error measure is that we hypothesize that it is the trend towards later hits that causes tempo drag.

2. Uncover problems, breakdowns, and shortcomings of the interface.

   To investigate the problems and breakdowns with MalLo we analyzed the observational data during the task noting participants' questions, confusions, and gesture technique. Our goal was to build a qualitative picture of the usability of MalLo. The observations were coded, and similar categories of observation events were grouped together and counted. Additionally, participants filled out a survey and participated in a semi-structured interview once all tasks were complete.

3. Gather qualitative feedback on how enjoyable the interface is to work with.

   To evaluate how enjoyable the interface was to use, we analyzed the gathered body language and utterances for valence information. Additionally, during the semi-structured exit interview where we discussed with the participants their feelings and experience.

6.3 Results

The timing errors are shown in Figure 6.5 for each interface and latency amount. The figure also shows both the smoothed error for all participants, computed via LOESS (Jacoby, 2000) which is a locally weighted polynomial regression method. Additionally, the absolute error LOESS curve is shown in order give an idea of the non-signed magnitude of the error. The grey box indicates an area of ± 30 ms error, which is the preferred range of error. The plots all exhibit higher error at the beginning of each trial, followed by a period of "settling" down in the direction of zero error. Higher latencies have much higher variance.

The absolute error indicates that for both the space bar and MalLo the participants were able to maintain a lower median error than the given latency once the performance settled. At 30 ms latency participants using both interfaces maintained a near zero or negative median error, meaning notes registered early. At 60 ms those using MalLo maintained a median error of around 30 ms, and the space bar 50 ms. At 90 ms those using both maintained a median error of around 50 ms. At 120 ms (an extreme amount of latency) MalLo maintained a median error of 50 ms, and the space bar between 50-100 ms. Note the latency are 60 and 120 ms, where while using MalLo, the participants maintained the same median magnitude as the previous level. We are unsure about the cause of this, and would like to explore it further. It is possible that there are latency ranges within which technique can remain constant (e.g., less than 90 ms and more than 90 ms) and this allowed for more practice and therefore better performance. However, because we did not gather stick height data, this is conjecture and warrants further investigation.

The signed error gives us a sense of how early or late the participants were playing. Note that for all latency levels, while using the MalLo interface participants succeeded in keeping the median signed error within the ± 30 ms zone. Using the space bar they were also successful for 30 and 90 ms, but failed at 60 and 120 ms. This means that on the whole, when people used MalLo they played less problematically late than when they used the space bar. However, the difference between the two is not large, and at any particular time no greater than 30 ms. Finally, note that qualitatively, the plots for each interface are quite similar. To help to quantify any differences we therefore used a significance test.

The results of the Mann-Whitney for the errors and absolute errors at each latency level are shown in Tables 6.3 & 6.4.

For latencies of 30 and 90 ms, the median differences of signed and absolute error are not significant, while for 60 and 120 ms they are. Note that for latencies larger than 30 ms, MalLo produces median errors less than or equal to the space bar. The magnitudes are similar for both error and absolute error, but each test tells us something a bit different. The results from absolute error give an intuition into just how error prone a participant's per-

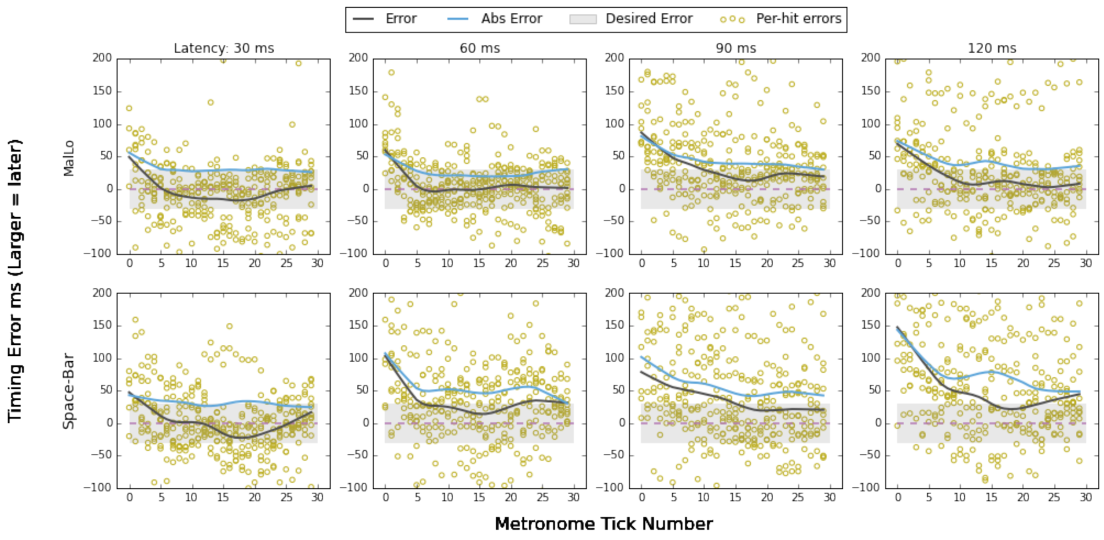


Figure 6.5: For each interface and each latency level, we computed the timing error offsets (how late or early the input was received) for each metronome tick. We used LOESS Jacoby (2000) to plot both a smoothed error curve as well as a smooth absolute error curve. Values below the dotted zero line represent inputs that were received before the desired time. Closer to the dotted zero line is better. Note that at the highest latency level the space bar exhibits a strong tendency to register inputs late while the smoothed MalLo inputs remain near zero.

	Median (ms)		Errors	
Latency	MalLo	space bar	U	p-value
30 ms	-2.0	-3.0	54233.5	0.06
60 ms	6.0	27.5	45667.5	< 0.001
90 ms	26.0	27.0	60286.0	0.34
120 ms	19.0	39.0	49465.0	< 0.001

Table 6.3: The results of the Mann-Whitney test on the *signed* timing errors. The median values here give us a sense of how late or early hits tended to be. Larger numbers mean the events occurred later in comparison to the metronome tick. Note that for latency levels of 60 and 120 ms, the test results are significant. The null hypothesis is that each condition has the same median.

	Median (ms)		Abs. Errors	
Latency	MalLo	space bar	U	p-value
30 ms	26.0	28.0	57984.5	0.426
60 ms	23.0	49.5	37641.5	< 0.001
90 ms	41.0	48.0	58117.6	0.11
120 ms	36.0	59.5	47903.5	< 0.001

Table 6.4: The results of the Mann-Whitney test on error *magnitudes*. The median values here give us a sense of the overall magnitude of errors for each interface and latency level. Note that for latency levels of 60 and 120 ms, the test results are significant. The null hypothesis is that each condition has the same median.

formance was, as well as how much the error varied. The results from the signed error tells us whether one interface had more late inputs (which contribute to tempo drag) than the other. Note that at 120 ms of latency, the space bar shows a notable shift towards later input times while the MalLo median remains within the desired < 30 ms region of error.

6.4 Discussion

The result of whether MalLo enables rhythmic synchronization is promising, but not definitive. Overall the best that we can claim is that in this study MalLo performed at least as well as the space bar in terms of errors. The significance tests imply a different median, but further study is needed. One confounding factor is that the interface is one that probably needs to be learned in order to gain proficiency. In the next chapter we study three musicians who learn the MalLo interface for a performance and analyze how their accuracy changes with time. Another possible confound is that practice with the space bar may have helped with MalLo for the current latency level. If we were to re-do this study we would randomize the interface order to avoid possible confounds and to make statistical analysis more sound.

6.4.1 Problems and Breakdowns

The main problem that was uncovered in this study was how participants react to the Leap Motion's loss of tracking of the tip of the striking stick. Often, when the system did not register, participants compensated by using a more jerky and forceful gesture, making it more difficult for the sensor to track. That they adopted a more forceful gesture is not surprising given the drum-striking metaphor that MalLo uses—it makes intuitive sense that a more forceful strike might activate a non-responsive sensor. Future work should be done into how to guide participants towards keeping their gestures within the proper velocity range.

6.4.2 Enjoyability and User Preference

During the exit interview, we asked participants which interface they preferred. 50% of participants (six participants) preferred MalLo and 50% preferred the space bar. The reasons for preferring the space bar included being familiar with it, and feeling secure in knowing that their input actions would register properly. Based on our visual observations, participants who performed well on using the space bar for the drumming task used a physical heuristic to help them to focus on the delayed sound of the drum as opposed to the press of the bar. For instance, four out of the six (who preferred the space bar) bobbed their head vigorously in time with the metronome ticks. Three out of the six modified their space bar-pressing action as latency increased. These three began to time the release of the space bar with the delayed drum sound, allowing their finger to rest for a short amount of time. One user slid their finger along the space bar, with the distance of the path correlating to the magnitude of the latency. As latency increased, they would slide their finger for a longer distance across the space bar, timing their release with the delayed drum sound.

Participants who did not like the space bar felt that it was awkward due to the knowing that they needed to "hit early", but being drawn to press the space bar at the time of the metronome tick. Four out of the six expressed verbal frustration as latency levels increased akin to, "I hate this!" or "This is terrible!" In contrast, increases in latency did not elicit such reactions while using MalLo. Four out of the six of those who preferred MalLo did not alter their technique until the 120 ms level of latency. This has potentially good implications for real-world use of this system, since Internet latency is constantly changing. Four out of six of participants who chose MalLo said that drumming was "more fun" than the space bar and that they played using "feel" and "intuition".

We measured frustration by counting the number of negative valence utterances and body language gestures. For instance, shaking their head with a frown on their face or exclaiming, "'This sucks!" were counted as a frustrated "event". We found that MalLo had a total of 24 such events and the space bar had 33. We did not find any correlation between participants' music or gaming experience levels and their number of frustrated events. We note that it is possible that the frustration counts could be confounded by the lack of randomization in the interface ordering.

6.4.3 Further Discussion

The purpose of the added visualization was to help the participants know when their gestures are properly within sensing range, and we now consider it a key part of MalLo. In our pilot studies, we found that people often gestured in such a way that the Leap Motion lost track of the tip of the drumstick. This was largely due to people moving too quickly or letting the stick go out of sensor range. Therefore, we created the visualization to help them understand how their actions were sensed by the system.

Despite the visualization tracking was a problem at times during this study, possibly confounding the results. This is likely due to a combination of our primitive prediction algorithm, and the slow frame rate of the Leap Motion sensor. Our initial study recommends at least 250 fps, while the Leap Motion over a USB 2.0 connection is 120 fps. We chose the Leap Motion for this project because of ease of setup and development. A remaining question is: Would a faster sensor significantly change the results of the study?

Another avenue that remains to be explored is how to improve the prediction algorithm, which is very primitive. In this study we did not collect height data, just timing data. Therefore it was not possible to analyze the gestures closely with the intention of improving the prediction algorithm. In the next chapter we present a case study where gesture data was collected and make recommendations about how to improve the prediction algorithm. Because tracking and reliability were one of the prime concerns of users, it is likely that improving these aspects will improve the enjoyment of such a system.

6.5 Conclusion

In this chapter we have presented a user study of the MalLo interface, comparing it to the space bar. We showed that for our drumming task, MalLo performed as well as or better than the space bar and was preferred for the task 50% of the time. Our observations indicate that the space bar provides reliability and familiarity. MalLo suffered from some poor tracking problems and was difficult to troubleshoot, and frustrated participants at times. Both interfaces were equally well-liked for the task, with stability given as the reason for favoring the space bar, and fun generally given as the reason for preferring MalLo.

7
MalLo in Use

7.1 Introduction

In this chapter we describe two studies designed to investigate how MalLo is used in performance situations by musicians. The first was a concert performance where musicians learned to play MalLo over a two week period. The second was a pilot usage of MalLo over the Internet.

7.2 The Numerical Experience Performance

The Numerical Experience (hereafter referred to as NumX) was a concert piece performed at Princeton University's House of Sound on January 11, 2017 and was created in collaboration with KatieAnna Wolf. The concept was a musical performance where audience members could customize their own audio experience. During the piece the concert hall was silent. Each of the audience members listened through a personal computing device such as a phone, tablet, or laptop. The piece, composed by Joshua Collins, had two movements. During the first movement the audience listened passively as they might in any concert. During the second movement the audience was able, through a browser interface on their computing device, to customize the audio they heard. The purpose of this performance, beyond the artistic value, was to gather data for this dissertation and that of Wolf. We held three rehearsals and one performance and recorded the musicians' interactions with MalLo, as well as collected observation notes.

This concert represents another application for MalLo and its predictive abilities—to enable expensive computational processes, such as complex audio synthesis. In the context of this concert piece MalLo allowed messages to be sent across a wifi network to a large number of computing devices that each synthesized audio within their web browsers.

The NumX concert attempted to answer three questions about MalLo:

1. How does technique change over time?
2. How does technique compare between musicians?
3. What characterizes good gesture technique?

7.2.1 Performance Description

Figures 7.1 and 7.2 show the NumX piece included three performers each playing a single MalLo. Their predictions were sent to a local webserver which forwarded the predictions on to the audience, each of whom had a computing device running a browser application of our creation. The goal of this performance, from the perspective of this thesis, was to collect data on how the musicians interacted with MalLo over 3 practice sessions and the performance. During these practice sessions the performers learned how to play MalLo, learned how to play the music piece, and then rehearsed the piece a number of times.

The piece was devised to help gather data for testing the accuracy of MalLo. Users played to a metronome, which we use later as a timing ground truth. On their screen they see a digit between 1 and 8 displayed, which indicated the number of times they needed to play during the next 8 beats. These digits are the sequence of decimal digits of the irrational number *e*, with 9 rounded down to 8. When the performers play a note, a synthesized percussion sound played in their headphones, as well as the headphones of any listeners. Velocity of the drumming stick was mapped to the volume of the percussive sound. Instead of predicting velocity, we use the velocity as reported by the Leap Motion the time of the prediction. This is because we found that final velocity is highly correlated with velocity at the time of prediction.

MalLo makes predictions in absolute time, which requires the clocks of MalLo, the server, and the web browsers to be synchronized to within a millisecond of each other. The MalLo computers were connected via Ethernet to the webserver, and synchronized a software clock via a custom version of NTP. The browser apps also synchronized their clocks to the webserver using the same method, but via wifi. Our custom NTP used a 1-D Kalman filter which modeled clock offset, but not clock skew. The performance was roughly 15 minutes in length. The first and second movements were each 6 minutes long with a short intermission.

During the second movement, the audience was given the option to change the synthesis parameters via our web interface. The interface was created by KatieAnna Wolf as part of her thesis research. It was designed to record the parameter changes of the audience. The audio was synthesized using the Web Audio API (Smus, 2013). The synthesized notes were percussive, and metallic sounding, generated via FM synthesis. The description and analysis of this interface and the audience's interactions with this are described in detail in (Wolf, 2017).

7.2.2 Rehearsal and Performance

Each rehearsal was approximately 1.5 hours in length. During the first rehearsal we introduced the musicians to the MalLo instrument and to the web interface for hearing the synthesized audio. We began by playing quarter notes at 60 bpm to a metronome to allow the performers to get a feel for the system. We played at 60 bpm for 22 minutes and played at 80 bpm for the rest of the session. During this session we did not rehearse the composition. During the next two rehearsals we played the composition three times at each. Between each rehearsal of the composition we conducted semi-structured interview and discussions.

The concert performance took place in the large performance space in the Princeton Music Department building. There were approximately 100 attendees. During the performance the players played the piece as described above.

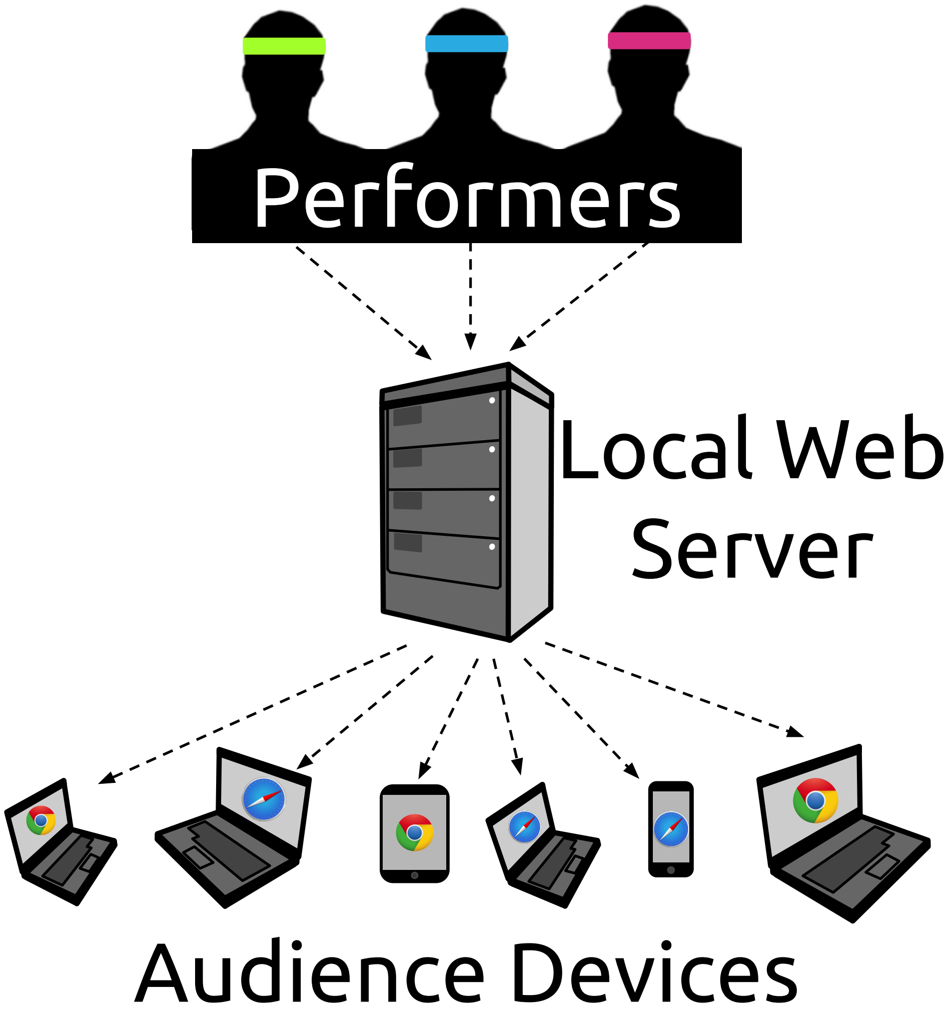


Figure 7.1: The NumX performance setup. Three performers each using a MalLo setup send predictions to a local web server which forwards the predictions to the audience where they are scheduled and synthesized into audio. During the second movement of the piece the audience can control the synthesis parameters to personalize their performance experience.

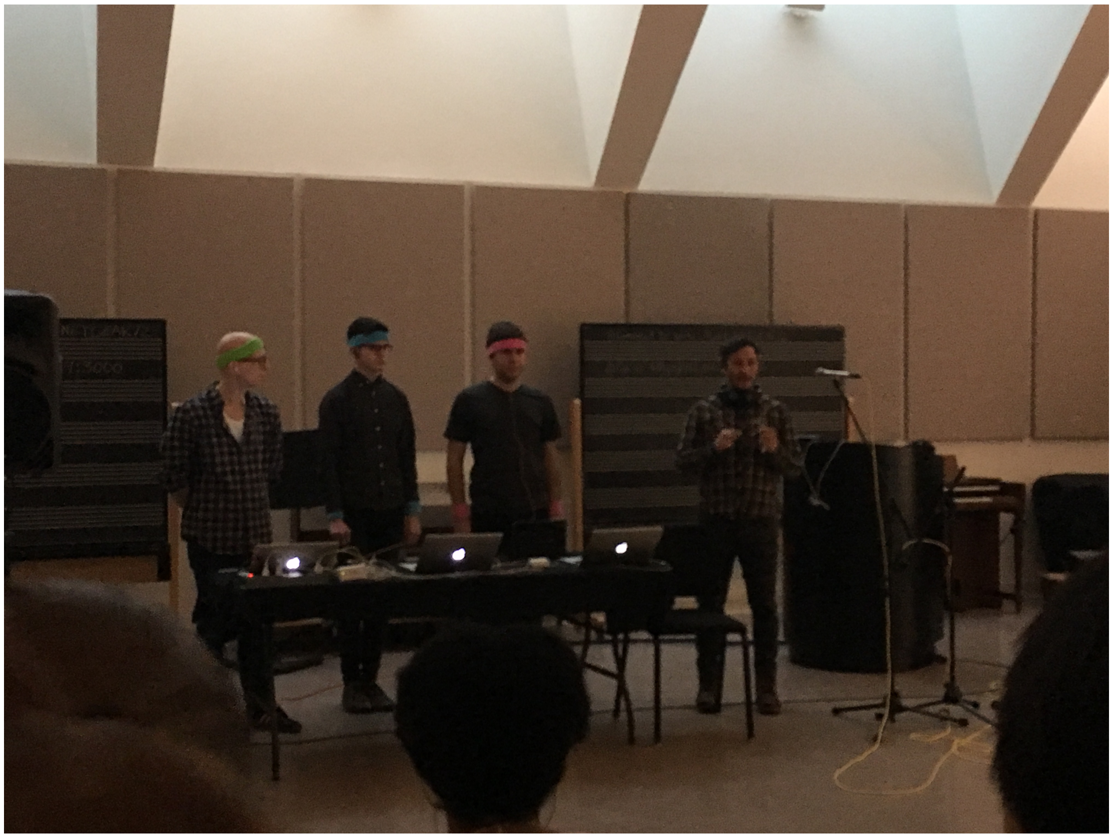


Figure 7.2: A photograph of the NumX performance being introduced by the author.

7.2.3 Data Gathered

During the 3 practice sessions and the performance we gathered the following data:

- Height and velocity of the tip of drum stick as seen by the Leap Motion sensor.
- When each prediction was played in each browser.
- The time of metronome ticks—our ground truth for when notes should be played.
- Written notes observing technique, performers' questions, and any problems that arose.

During the semi-structured interviews we asked questions such as:

- How did that feel?
- Did you have any problems? What were they?
- I noticed that you had a problem with _____. What was going on there?

7.2.4 Data Analysis

To measure the accuracy of the drumming over time, we compute the error between when each note was heard and when it should have been heard according to the metronome. For each performer, the predictions that are made by a gesture are sent to two other listening browsers including their own. We compute the error for all of the notes heard in all three browsers. Note that we do not account for notes that were intended to be played but were not. Neither do we discard false positives. Because of the improvisational nature of the composition, we have no score to determine whether or not a note was supposed to have been played, only whether it is rhythmically accurate. These errors (both signed and unsigned) were computed and then plotted for visual analysis in Section 7.2.6.

To analyze the gesture style we plot the stick tip heights for each performer during each practice session and concert performance. We compare them visually across the performers and across the session to characterize how each changed with time and how they were similar or different from the other performers. We analyze the plots for the causes of false positive notes.

7.2.5 Participants

Participants P1, P2, and P3 were recruited for this performance by Jeff Snyder, the director of the Princeton Laptop Orchestra. All three have a history of playing computer music and experimenting with new interfaces for musical expression. All three are affiliated in some way with the Princeton Laptop Orchestra (Trueman et al., 2006).

7.2.6 Results

Figure 7.3 shows plots of the stick heights for the performers. All three plots are taken from the same period of time of the rehearsal. Note the clear difference in curves between the two performers. P1 tends to keep the stick at a low altitude and then before a note quickly raise and lower it. When observing P1 in person, the gesture looks similar to the motion a conductor makes with their baton—controlled but forceful. P2 kept the stick tip high, rarely dipping below 100 mm of height. P2 held the stick gently between the thumb and forefinger, allowing it to pivot on the forefinger. P3 gripped the stick tightly, keeping it parallel with the ground and moved in a slow, deliberate manner, as is evidenced by the

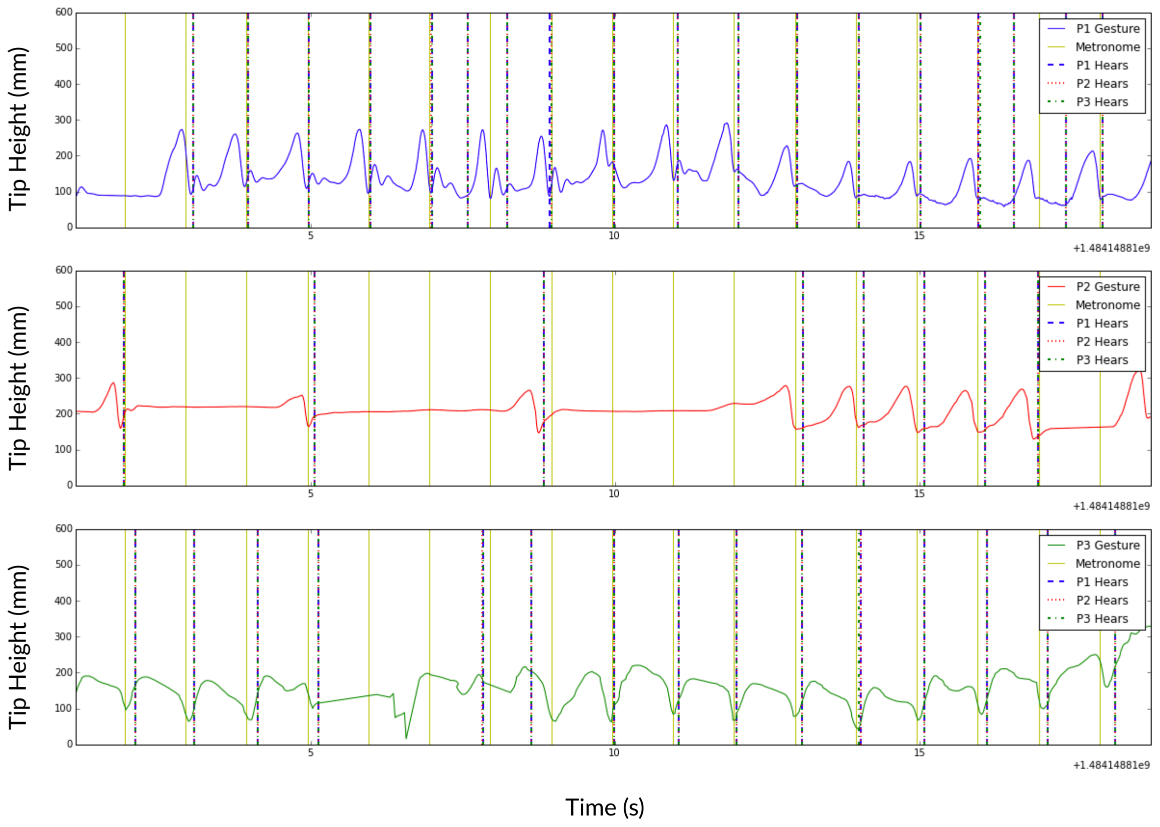


Figure 7.3: Representative plots of the performers' stick tip heights over time during the first performance. Each performer is color coded. The yellow solid lines represent the desired note time, and the dotted lines represent when each performer hears the note that was synthesized from the plotted gesture. Note the distinct curves of each performer. This session was performed at 60 bpm.

dome-like curves in the plot.

Figure 7.4 shows representative plots of the stick heights for each performer at the final performance. The technique of each performer is similar in shape to their original technique, but the style and accuracy has become refined. This sample was taken from the beginning of the concert when all of the performers were very accurate, as is shown later the accuracy P2 decreased as the performance progressed.

Figure 7.5 shows excerpts from P1's playing over the three practice sessions and the con-

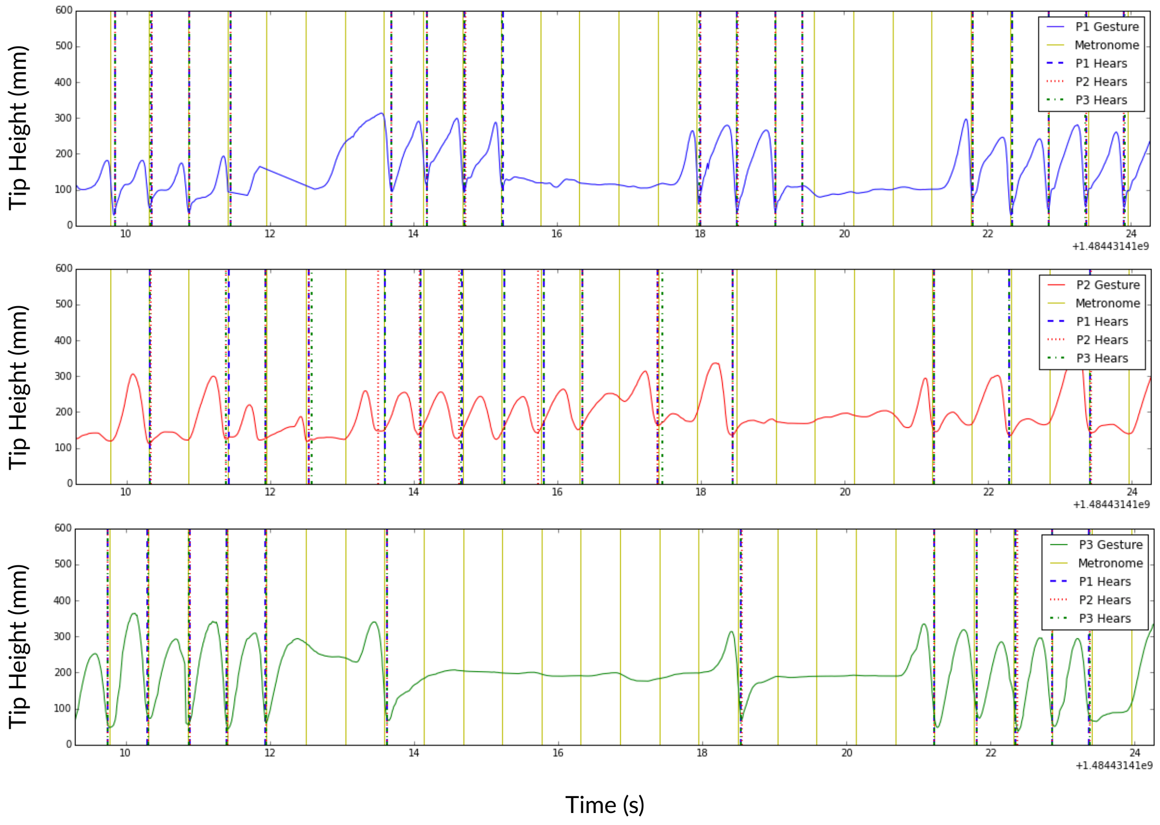


Figure 7.4: Representative plots of the performers' stick tip heights over time during the concert performance. Each performer is color coded. The yellow solid lines represent the desired note time, and the dotted lines represent when each performer hears the note that was synthesized from the plotted gesture. Note the distinct curves of each performer. This session was performed at 110 bpm.

cert. The excerpts were chosen to accurately represent their technique during the sessions. Here we will use each of the four plots to discuss aspects of technique and the prediction algorithm that we observed across the performers and across time.

Practice 1

This plot has high frequency content that can cause false positives. When P1 pulls up at the bottom of the drumming gesture there is a wobble in the stick tip which on 2 occasions causes a false positive. In general their accuracy is high, but this plot was made at 60 bpm.

Practice 2

The style is similar to Practice 1. The tempo has increased to 110 bpm which is the likely cause of the drop in accuracy. There is still a lot of high frequency content in the curve. A common problem in practice session 2 was that performers would get into their own "groove" that was influenced by the physics of the stick. When held lightly between the fingers it bounced at its own rhythm and sometimes pulled the performer away from the metronome beat. This can be seen in the middle half of the plot.

Practice 3

The technique has become more refined. This plot was taken near the end of a long practice session. There is much less high frequency content. The accuracy of the player is increased. There is one false positive which was caused by a very slight dip in the stick height.

7.2.7 Concert Performance

The technique has become very controlled and precise. The movements are larger and very confident. Of note is the consistently controlled recovery before the next stroke. Sequential notes exhibit very similar shapes. There are no false positives in this plot, but there were a some during the performance.

7.2.8 Error Over Time

Figure 7.6 shows the signed error of each performer across all the practices and performances. The error is normalized and given as the fraction of the beat length. This because the tempo changed and therefore an error in milliseconds would find its upper bound decreasing as the tempo increased because of the shorter beat period. This plot gives an intuition into the accuracy and whether their playing tended to be early or late. All the performers showed a high degree of error variance in the first practice session, and a higher accuracy in final performance.

Figure 7.7 shows the unsigned error of each performer. This plot helps us to understand the overall error magnitude of each performer. Again we see a high degree of variation in the first practice session. P2 had higher error during practice session 2. All of the performers had periods of very low error during the first part of the performance. The per-session mean error and standard deviation for each player is shown in Figure 7.8.

7.2.9 Discussion

To address the question of the what makes a technique reliable, our opinion is that a technique that has a high degree of control and fewer high frequency wobbles helps avoid false

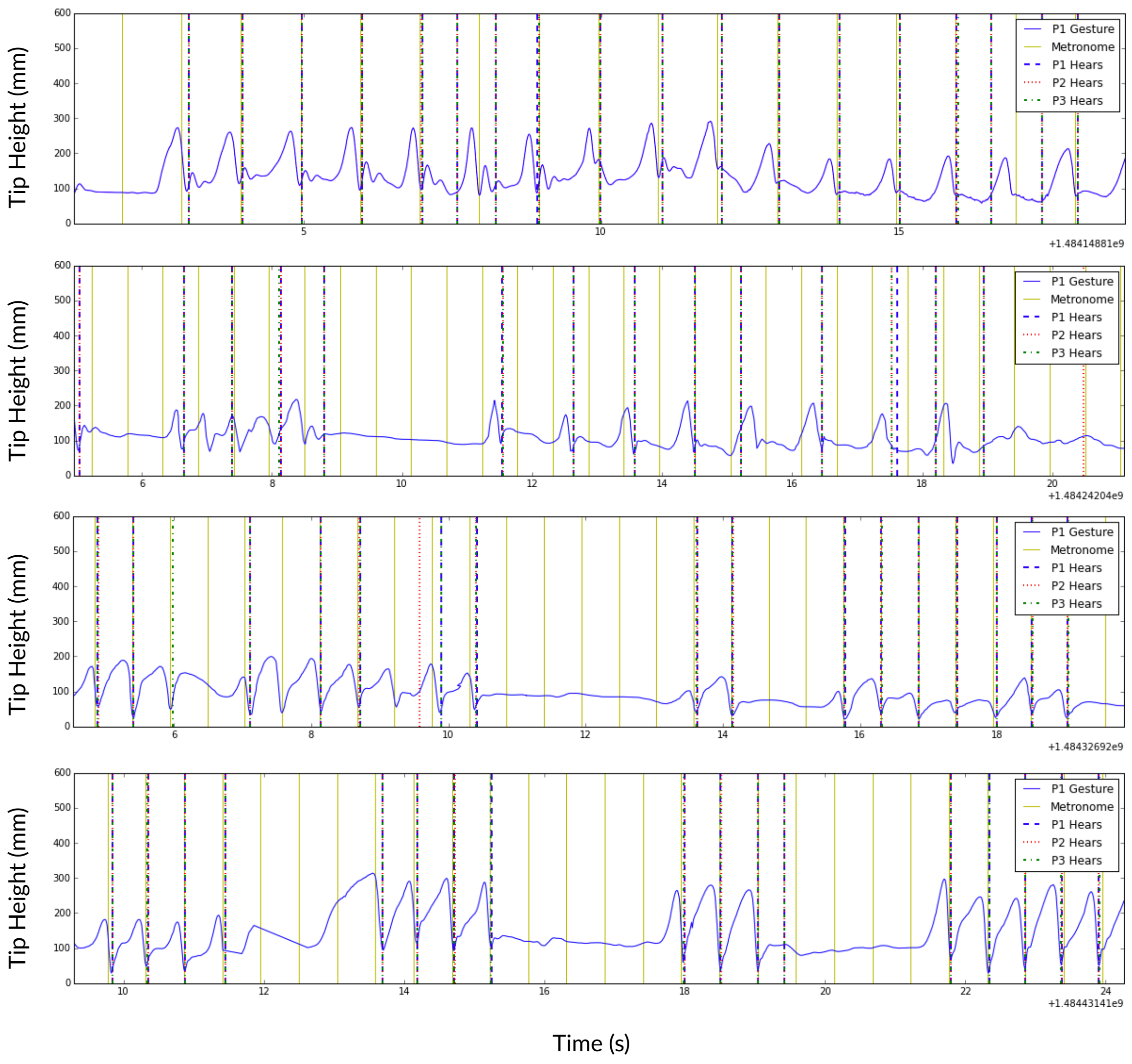


Figure 7.5: Plots of P1's stick height over time. The top plot 3 plots are from practice sessions 1-3, respectively, and the bottom is from the concert performance. The top plot was performed at 60 bpm, and the other three at 110 bpm. Note the evolution of technique towards efficient, confident movements with less random movement and therefore less possibility of spurious predictions.

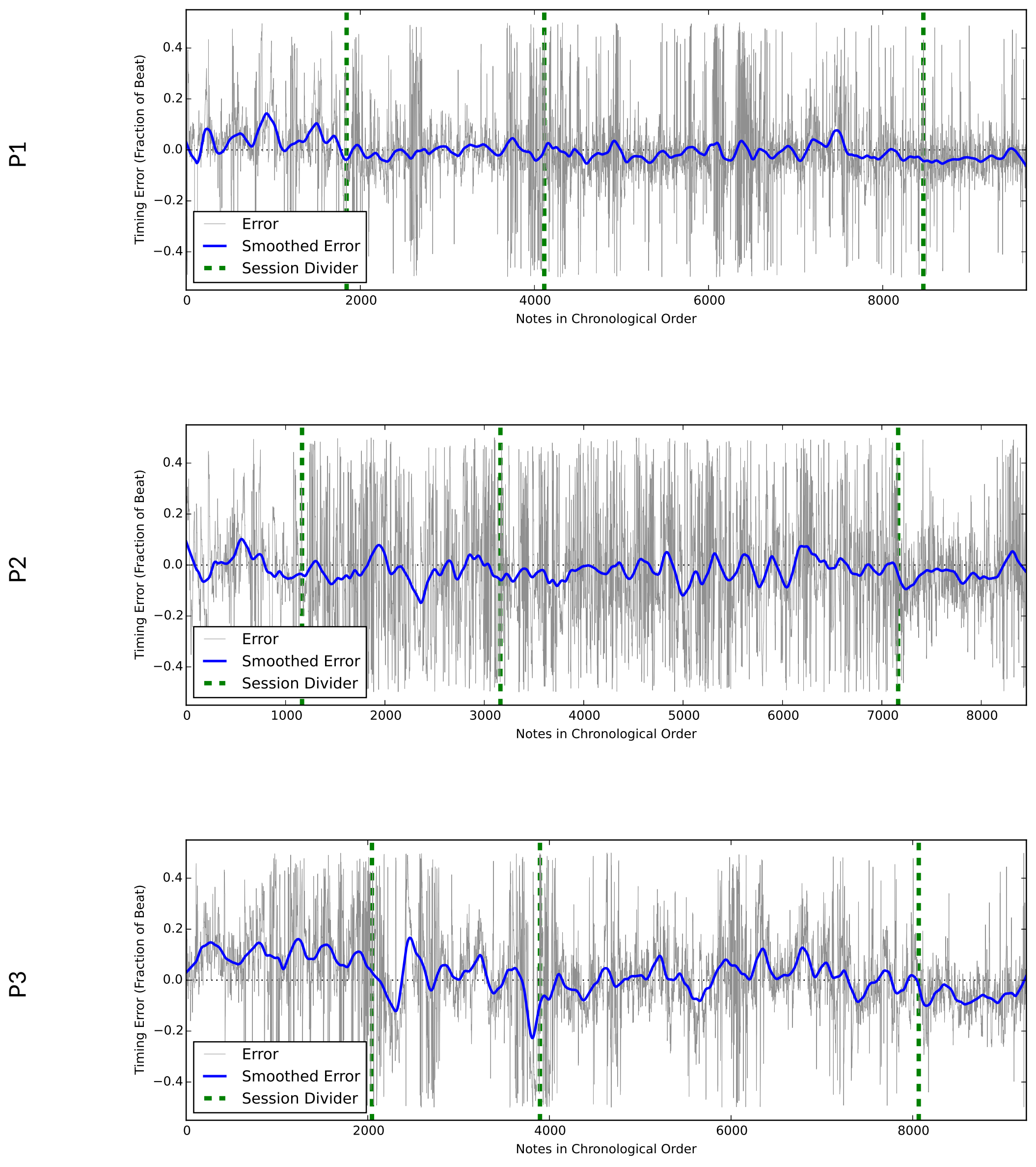


Figure 7.6: Each player's signed error over time, over all the practice and performances. Session 1 (leftmost division) was played at of 60 and then 80 bpm. All other sessions were performed at 110 bpm. Higher is later. Note that players tended to play late in the early sessions.

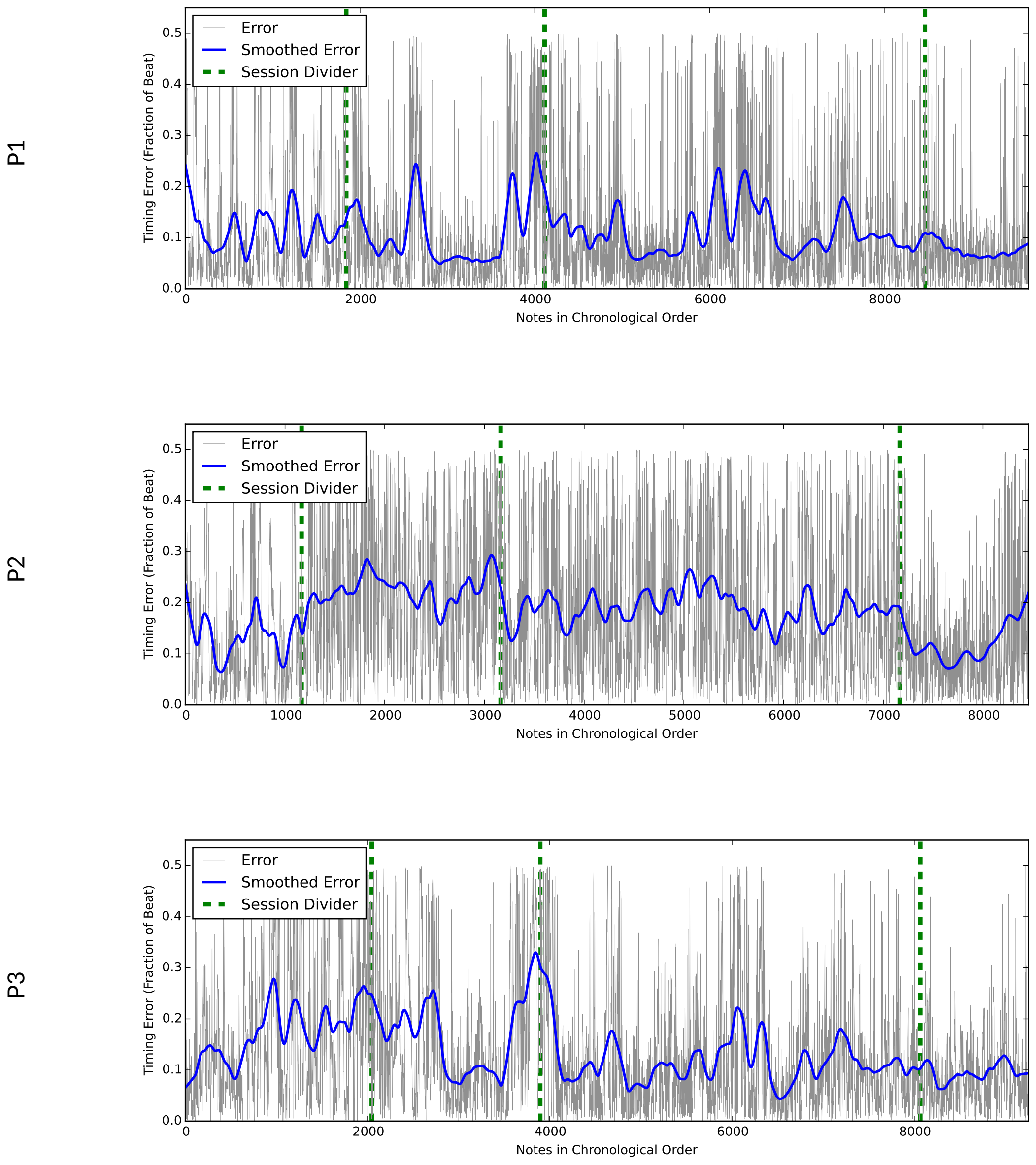


Figure 7.7: Each player's absolute error over time, over all the practice and performances. Session 1 (leftmost division) was played at of 60 and then 80 bpm. All other sessions were performed at 110 bpm. Lower means less overall error.

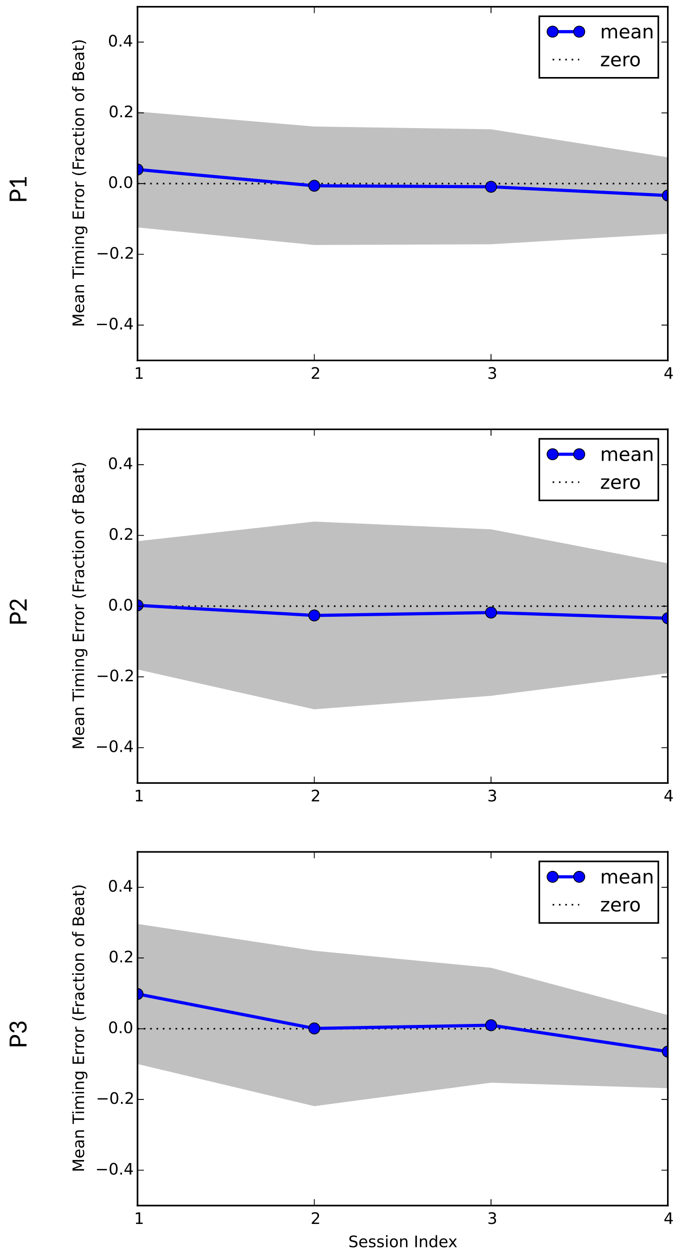


Figure 7.8: Each player's mean absolute error over the 3 practice sessions and performance. Session 1 was played at of 60 and then 80 bpm. All other sessions were performed at 110 bpm. The shaded boundary indicates one standard deviation.

positives. These wobbles typically occur during the recovery after the downward striking motion. A quick but smooth recovery before the next stroke helps to avoid these. P2, who used a technique that kept their stick high experienced more error over time. P1 and P3 both converged on a similar quick and smooth downward motion. Their recovery was different, but in both cases was smooth and controlled. To summarize the ideal MalLo technique as we currently understand it:

- Strike firmly and dip down as far as possible.
- Recover smoothly and don't wobble.
- Hold the stick firmly.

We have seen that technique using MalLo changes with time, and accuracy improves. While improved accuracy reduces the number of timing errors, the plots give us insight into the drawbacks of the prediction method. The system is susceptible to false positives that can be caused by small dips in the tip height. It is our opinion that detecting and preventing these false positives is the key to a more usable generation of this style of predictive interface. Probably a great number can be fixed using simple tuning approaches. However, a better approach might be to use a mixture of two models, one to model downward stroke and one to model the recovery. Another approach would be to use machine learning or another data-driven method.

7.3 Internet Pilot Study

We ran a pilot study in order to test the feasibility of MalLo for use over the Internet. The study was performed with a collaborator in Alma, Wisconsin, the participant known as P2 in Chapter 4. Here they will be referred to here as M1 (to avoid confusion with the P2 in the NumX performance). M1's demographic details are discussed in detail on page 55.

7.3.1 Study Design

The study used the same software as NumX. In this case, however, each user ran their own webserver hosted on a PIGMI. The PIGMI enabled high-accuracy clock synchronization between servers. Both MalLo devices sent predictions to the local PIGMI, as well as the remote PIGMI.

During the study, E and M1 played a quarter-eight-quarter-eight-quarter note pattern without a metronome, twice. We then computed the timing offsets between each of E's notes and the corresponding note in M1's playing.

The latency between E and M1 was not measured, but it was assumed to be similar to the levels recorded for previous session, which was roughly 40 ms in each direction.

7.3.2 Results

The first of the two playing sessions is shown in Figure 7.9. Negative error means that M1 played early. In this session the median error is -60 ms, the standard deviation is 115 ms, and the percentage of notes that have less than $\pm$30 ms error is 22%. Figure 7.10 shows the second session. Here the median error is -32 ms, the standard deviation is 59 ms, and the percentage of notes within $\pm$30 ms is 36%.

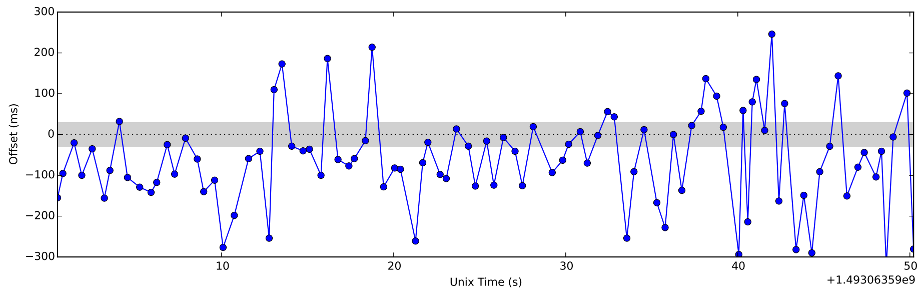


Figure 7.9: The timing offsets of M1 and E playing the pattern described on page 136. The shaded area shows the range of $\pm$30 ms.

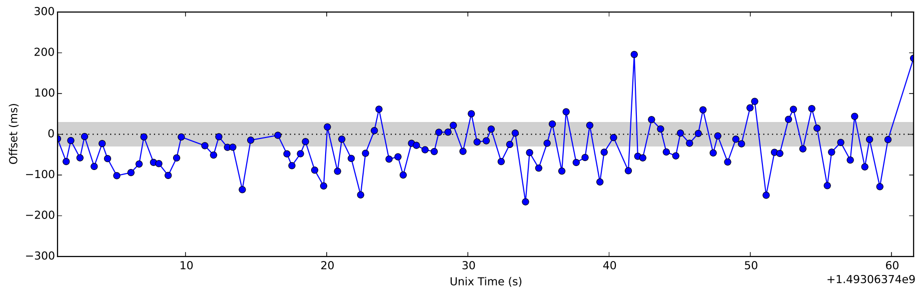


Figure 7.10: The timing offsets of M1 and E playing the pattern described on page 136. The shaded area shows the range of $\pm$30 ms.

7.3.3 Discussion

Note that the timing errors improved between sessions 1 and 2, but the amount of error is still considerably higher than we would like. We would prefer the vast majority of timing errors to be under 30 ms. Still, despite this level of error M1 and E reported that they were able to synchronize with one another, easily, with no perceivable tempo drag.

This result is encouraging, but is not drastically different than the degree of synchronization that two trained musicians might enjoy when playing their instruments of choice. There are confounding factors in this pilot study including the fact that M1 does is not proficient in playing MalLo, and synchronizing between two humans is a different task than we used our previous studies (i.e., synchronizing to a metronome).

This study was rudimentary and is meant as a precursor to more rigorous investigation into MalLo as an instrument for the Internet. Some of the open questions that remain are: How does MalLo perform under higher latency conditions, and latency that varies? What is the limit to how much latency can be tolerated?

7.4 Conclusion

In this chapter we have presented two studies meant to give us insight into MalLo as a performance instrument. We found that musicians' technique does improve with time, improving accuracy. We also identified what it means for technique to improve. Specifically, it involves controlled movements with less high frequency shaking of the mallet tip. We uncovered ways in which the MalLo prediction algorithm might be improved. In particular, MalLo would benefit from improved mechanisms for filtering out false positives. Finally, we performed a pilot study using MalLo on the open Internet. The results were promising but more investigation is needed to understand its strengths and weaknesses.

8 Conclusion

8.1 Summary and Contributions

In this thesis we have investigated new ways to enable rhythmically synchronized music performance over the Internet. The Internet offers a tantalizing space for music performance, and the ability to reach huge audiences. But this dream remains just out of reach due to network latency.

In this section, we summarize our work and the contributions of this thesis. These contributions include:

1. The Global Metronome: A method to synchronize tempo between devices that are separated by great distance.
2. PIGMI: A small, inexpensive timeserver that facilitates the Global Metronome approach.
3. A better understanding of the advantages and disadvantages of MIDI tempo synchronization via the Global Metronome.
4. The distributed, synchronized music instrument, MalLo. An approach to latency reduction via per-note prediction.
5. A simulation, usability study and a study of live performance exploring the viability of the MalLo approach.

8.1.1 The Global Metronome

The Global Metronome is an *absolute* tempo synchronization scheme that capitalizes on the fact that computer system clocks are becoming increasingly easy to synchronize with high accuracy. We presented evidence to show that it is possible to synchronize unconnected devices to a degree suitable for music performance. Additionally we presented the PIGMI (Pi Global Metronome Implementation), a tiny, affordable timeserver that greatly simplifies the process of using the Global Metronome approach.

8.1.2 The Global Metronome in Use

We presented findings from studies with three musicians as an exploration into the use of the Global Metronome for live performance. By setting up an Internet jamming system using sequencers and the Global metronome, we showed that it is effective for tempo synchronization over long distances. We have also showed that in certain cases where latency is relatively constant and symmetric, it is possible to use traditional clock synchronization methods instead of GPS. However we note that the highly synchronized clocks of the PIGMIs greatly simplify the process of troubleshooting network problems.

8.1.3 MalLo: A Distributed, Synchronized Musical Instrument

We showed how it is possible to use prediction in order to create a truly distributed, synchronized, musical instrument (DSMI) despite the latency inherent in the Internet. We described the implementation of our instrument, MalLo, and our tests showed that its timing prediction is likely within the perceptual bounds of synchrony for network one-way latencies up to 80 ms, and that its velocity prediction performs well for latencies below 50 ms. We created straightforward algorithms for both the sending and receiving parts of the DSMI. Finally, we presented a method for latency peak reduction, using an overlay network for message transmission. These proved effective at reducing the effects of long term network fluctuations on the order of 5ms, and some burst latency on the order of 40 ms.

8.1.4 MalLo Usability

We presented a user study of the MalLo interface, comparing it to a common interface—the space bar. We showed that for a drumming task, users using MalLo performed as well as or better than the space bar and was preferred 50% of the time. MalLo suffered from some poor tracking problems and was difficult to troubleshoot, and frustrated participants at times. Both interfaces were equally well-liked for the task, with stability given as the reason for favoring the space bar, and fun generally given as the reason for preferring MalLo.

8.1.5 MalLo in Use

We performed two studies meant to give us insight into MalLo as a performance instrument. The first, a concert performance, enabled us to study three musicians over three rehearsals and the performance. We found that musicians' technique does improve with time, improving accuracy. We identified that controlled movements with less high frequency shaking of the mallet tip are import to playing accuracy. We also uncovered ways in which the MalLo prediction algorithm might be improved. The second, a pilot study of MalLo over the Internet, shows promise, but more in-depth study is needed.

8.1.6 Future Work

This work explores two avenues for playing rhythmically synchronized music over the web. It is our belief that to develop the Internet as a performance space, we need a new class of instruments and new types of music interaction that are fun, yet specifically designed for the Internet. We chose sequenced music as a possible candidate for Internet performance because it is a common practice, and many artists enjoy doing it. Sequenced music stands out because an artist can use their existing tools in nearly the same manner that they normally use them. This allows artists to capitalize on the skills they already have. By identifying other instruments and paradigms that would work well on the web we can build a new set of tools for a new performance space. In particular, we can seek out musical instruments that emphasize predictability as a desirable trait.

8.2 Conclusion

Since the emergence of the Internet in the 1990's we have made remarkable progress towards developing it as a rewarding musical space. There has been significant work into studying perception, reducing latency, and exploring new ways to control audio processes via networks. However, there is still progress to be made before we see widespread musical collaboration over the web. We look forward to exploring new instruments and methods for playing live, collaborative music over the Internet.

References

(2013). Global IP Network Latency. http://ipnetwork.bgtmo.ip.att.net/pws/network_delay.html.

(2016). Ableton link. `https://www.ableton.com/en/link/`. Accessed: 2016-01-01.

Ableton (2017). Live is software for creating musical ideas, turning them into finished songs, and even taking them onto the stage. `https://www.ableton.com/en/live/`. Accessed: 2017-05-06.

Allan, D. W. & Weiss, M. A. (1980). *Accurate time and frequency transfer during common-view of a GPS satellite*. Electronic Industries Association.

Andersen, D., Balakrishnan, H., Kaashoek, F., & Morris, R. (2001). Resilient overlay networks. In *Proc. ACM Symposium on Operating Systems Principles*.

Boie, B. (1989). *The radio drum as a synthesizer controller*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library.

Bradski, G. & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV library*. O'Reilly.

Brandenburg, K. (1999). MP3 and AAC explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*: Audio Engineering Society.

Broadbent, D. E. & Ladefoged, P. (1959). Auditory perception of temporal order. *The Journal of the Acoustical Society of America*, 31(11), 1539–1539.

Brown, C. (2002). Indigenous to the Net: Early Network Music Bands in the San Francisco Bay Area. `http://crossfade.walkerart.org/brownbischoff/index.html`. Accessed: 2017-05-07.

Cáceres, J. & Chafe, C. (2010). JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research*.

Carôt, A., Hohn, T., & Werner, C. (2009). *Netjack-Remote music collaboration with electronic sequencers on the Internet*. Proceedings of the Linux Audio Conference.

Chafe, C. (2012). Living with net lag. In *Proceedings of AES 43rd International Conference*.

Chafe, C., Cáceres, J., & Gurevich, M. (2010). Effect of temporal separation on synchronization in rhythmic performance. *Perception*, 39.

Chafe, C., Gurevich, M., Leslie, G., & Tyan, S. (2004). Effect of time delay on ensemble accuracy. In *Proceedings of the International Symposium on Musical Acoustics*.

Chafe, C., Wilson, S., Leistikow, R., Chisholm, D., & Scavone, G. (2000). A simplified approach to high quality music and sound over ip. In *COST-G6 Conference on Digital Audio Effects* (pp. 159–164).

Chew, E., Sawchuk, A., Tanoue, C., & Zimmermann, R. (2005). Segmental Tempo Analysis of Performances in User-Centered Experiments in the Distributed Immersive Performance Project. In *Proceedings of Sound and Music Computing Conference*.

Dahl, S. (2011). Striking movements: A survey of motion analysis of percussionists. *Acoustical science and technology*, 32(5), 168–173.

Dana, P. H. (1997). Global positioning system (gps) time dissemination for real-time applications. *Real-Time Systems*, 12(1), 9–40.

Dannenberg, R. (2006). The Interpretation of MIDI Velocity. In *Proceedings of the International Computer Music Conference*.

Davis, P. & Hohn, T. (2003). Jack audio connection kit. In *Proc. Linux Audio Conference, LAC*, volume 3 (pp. 245–256).

Eidson, J. & Lee, K. (2002). Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In *Sensors for Industry Conference, 2002. 2nd ISA/IEEE* (pp. 98–105).: IEEE.

Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36.

Ericsson (2015). Network Synchronization A key component of Ericsson's Evolved IP Network solution. `http://archive.ericsson.net/service/internet/picov/get?DocNo=5/28701-FGB101686&Lang=EN&HighestFree=Y`. Accessed: 2017-04-07.

Hertz, H. (1881). On the contact of elastic solids. *Math*, 92, 156–171.

Jacoby, W. G. (2000). Loess:: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies*, 19(4), 577–613.

Jamkazam (2017). The jamkazam platform. `https://www.jamkazam.com/products/platform`. Accessed: 2017-05-05.

Jin, Z., Oda, R., Finkelstein, A., & Fiebrink, R. (2015). MalLo: A Distributed, Synchronized Instrument for Internet Music Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

Kadenze (2017). Online Arts and Technology Courses | Kadenze. `https://www.kadenze.com`. Accessed: 2017-05-06.

Kapur, A., Wang, G., Davidson, P., & Cook, P. (2005). Interactive Network Performance: A dream worth dreaming? *Organised Sound*, 10.

Korg (2017). Today, and into the future. inspiring instruments that make music fun. `http://www.korg.com/us/`. Accessed: 2017-05-06.

Laird, J. (2012). Clock synchronization terminology. `https://www.iol.unh.edu/sites/default/files/knowledgebase/1588/clock_synchronization_terminology.pdf`. Accessed: 2017-05-04.

Marvin, C. (1990). *When old technologies were new*. Oxford University Press.

McCartney, J. (2002). Rethinking the computer music language: SuperCollider. *Computer Music Journal*, 26(4), 61–68.

Mills, D. L. (1991). Internet time synchronization: The network time protocol. *Communications, IEEE Transactions on*, 39.

Moog, R. A. (1986). Midi: Musical instrument digital interface. *Journal of the Audio Engineering Society*.

Moon, S. B., Skelly, P., & Towsley, D. (1999). Estimation and removal of clock skew from network delay measurements. In *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*.

Narveson, J. & Trueman, D. (2013). LANdini: a networking utility for wireless LAN-based laptop ensembles. In *Proceedings of SMC Sound, Music and Computing Conference*.

Neil, J. (2012). Synchronization distribution architectures for lte networks. `https://www.microsemi.com/document-portal/doc_view/133469-synchronization-distribution-architectures-for-lte-networks`. Accessed: 2017-04-07.

Nokia (2017). Synchronization as a service. `https://tools.ext.nokia.com/asset/200644`. Accessed: 2017-04-07.

Novation (2017). The inspirational grid-based groove box. `https://us.novationmusic.com/circuit/circuit`. Accessed: 2017-05-06.

Oda, R. & Fiebrink, R. (2016). The Global Metronome: Absolute tempo sync for networked musical performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

Oda, R., Finkelstein, A., & Fiebrink, R. (2013). Towards Note-Level Prediction for Networked Music Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

Oliveros, P., Weaver, S., Dresser, M., Pitcher, J., Braasch, J., & Chafe, C. (2009). Telematic Music: Six Perspectives. *Leonardo Music Journal*, 19.

openFrameworks (2017). openframeworks is an open source c++ toolkit for creative coding. `http://openframeworks.cc`. Accessed: 2017-04-26.

Palacio-Quintin, C. (2008). Eight years of practice on the hyper-flute: Technological and musical perspectives. In *NIME* (pp. 293–298).

Paxson, V., Almes, G., Mahdavi, J., & Mathis, M. (1998). *Framework for IP Performance Metrics*. RFC 2330.

Peterson, L. L. & Davie, B. S. (2007). *Computer Networks: A Systems Approach*. Elsevier.

Postel, J. (1980). *User Datagram Protocol*. RFC 768.

Reinheimer, P. & Will, R. (2015). WonderNetwork: Global Ping Statistics. https://wondernetwork.com/pings.

Rottondi, C., Chafe, C., Allocchio, C., & Sarti, A. (2016). An overview on networked music performance technologies. *IEEE Access*.

Sarkar, M. et al. (2007). Tablanet: A real-time online musical collaboration system for indian percussion. Master's thesis, Massachusetts Institute of Technology.

Sarkar, M. & Vercoe, B. (2007). Recognition and prediction in a network music performance system for Indian percussion. *Proceedings of the International Conference on New Interfaces for Musical Expression*.

Sesia, S., Toufik, I., & Baker, M. (2009). *LTE: the UMTS long term evolution*. Wiley Online Library.

Smus, B. (2013). *Web Audio API: Advanced Sound for Games and Interactive Apps*. " O'Reilly Media, Inc.".

Squarp (2017). Modern, easy-to-use and versatile hardware sequencer. `http://www.squarp.net`. Accessed: 2017-05-06.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*. Springer.

Trueman, D., Cook, P. R., Smallwood, S., & Wang, G. (2006). PLOrk: The Princeton Laptop Orchestra, Year 1. In *ICMC*.

Twitch (2017). Social video for gamers. `https://www.twitch.tv/p/about`. Accessed: 2017-05-06.

Vos, P. G., van Assen, M., & Fraňek, M. (1997). Perceived tempo change is dependent on base tempo and direction of change: Evidence for a generalized version of Schulze's (1978) internal beat model. *Psychological Research*, 59(4), 240–247.

Wanderley, M. M. (2001). Gestural control of music. In *International Workshop Human Supervision and Control in Engineering and Music* (pp. 632–644).

Wanderley, M. M., Vines, B. W., Middleton, N., McKay, C., & Hatch, W. (2005). The musical significance of clarinetists' ancillary gestures: An exploration of the field. *Journal of New Music Research*.

Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5), 6380–6393.

Weiss, M. (2012). Telecom requirements for time and frequency synchronization. In *National Institute of Standards and Technology (NIST), USA,[Online]: www. gps. gov/cgsic/meetings/2012/weiss1. pdf*.

Wolf, K. (2017). *Designing Techniques and Tools For End-User Sonification Design and Personalized Performance*. PhD thesis, Princeton University.

Wondernetwork (2017). Global ping statistics. `https://wondernetwork.com/pings`. Accessed: 2017-05-04.