MalLo: A Distributed, Synchronized Instrument for Internet Music Performance

Zeyu Jin*
Department of Computer Science
Princeton University
zjin@cs.princeton.edu

Reid Oda*
Department of Computer Science
Princeton University
roda@cs.princeton.edu

Adam Finkelstein
Department of Computer Science
Princeton University
af@cs.princeton.edu

Rebecca Fiebrink
Department of Computing
Goldsmiths, University of London
r.fiebrink@gold.ac.uk

ABSTRACT

The Internet holds a lot of potential as a music listening, collaboration, and performance space. It has become commonplace to stream music and video of musical performance over the web. However, the goal of playing rhythmically synchronized music over long distances has remained elusive due to the latency inherent in networked communication. The farther apart two artists are from one another, the greater the delay. Furthermore, latency times can change abruptly with no warning.

In this paper, we demonstrate that it is possible to create a *distributed, synchronized musical instrument* that allows performers to play together over long distances, despite latency. We describe one such instrument, MalLo, which combats latency by predicting a musician's action before it is completed. MalLo sends information about a predicted musical note over the Internet before it is played, and synthesizes this note at a collaborator's location at nearly the same moment it is played by the performer. MalLo also protects against latency spikes by sending the prediction data across multiple network paths, with the intention of routing around latency.

Author Keywords

Networked Performance, Novel Musical Interface, Object Tracking, Network Latency Reduction

1. INTRODUCTION

Just as the Internet has come to mediate so many other aspects of our culture, it also offers huge potential as a shared musical performance space. It affords new opportunities like discovering collaborators among a worldwide community of musicians, holding impromptu improvisation sessions, and playing with others who share similar artistic goals and vision. Unfortunately, this potential remains largely unrealized because communication latency inhibits rhythmic synchronization among remote performers using traditional instruments.

Depending on musical context, latency greater than some threshold causes *tempo drag*: each musician repeatedly slows down in response to apparent delay from their remote partner [6, 7]. As latency increases it becomes increasingly difficult to stay on beat, until eventually the performance breaks down. In practice, this means that artists have a limited *radius of collaboration*, meaning they can only play with artists who are within a certain distance (because network latency roughly correlates with physical distance). Players can adjust to a certain amount of lag, so if we assume that people can tolerate 45 ms of latency [6, 7], it is possible for artists in New York and San Francisco to play with one another, but they will have to train themselves to ignore the fact that the music is audibly out of sync, and be conscious of not slowing down. Furthermore, they will be at the mercy of network traffic, where bursts of congestion are the norm and cause unpredictable delays. This approach has been the basis of many telematic performances played in spite of lag [14]. Another approach to working with latency has been to allow one side of the connection to lead while the other follows [11]. However, these approaches do not attempt to remove latency, and so they cannot truly allow for distributed ensembles in which the audio is synchronized among multiple locations. We envision 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 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 paper 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.

The synchronization in a DSMI is accomplished by predicting a note before it is played. These note predictions can then be transmitted over the network early enough that they can be played remotely at the same time as they occur locally (Figure 1). DSMIs capitalize on the fact that the musician no longer needs to manually excite a physical instrument in order to generate sound. Instead, we can use synthesis algorithms along with a sound system to generate audio. This decoupling is essential, because it allows us to begin the transmission process without waiting for sound to be generated locally. Thus, when constructing a DSMI, we seek to design an instrument with *predictable* properties

*These authors contributed equally.

NIME'15, May 31-June 3, 2015, Louisiana State Univ., Baton Rouge, LA.

rather than particular acoustic properties. Two crucial aspects of predictability are the *anticipation time* (i.e., how far in advance can it 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.

In this paper, we present our DSMI named "MalLo" (short for *Mal*let *Lo*cator), which combines prediction of a musician's gesture with a network forwarding strategy in order to support synchronization between distant performers. We show that MalLo is capable of playing a note that is nearly synchronized (within 30 ms) with an actual mallet strike thousands of miles away. MalLo builds on the work of Dahl [8] and Oda [13], 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 40-80 ms) and smooth. This smoothness stems from the purpose of the gesture: to efficiently build kinetic energy to convert into audio, via a vibrating surface. We describe MalLo's sensing and prediction mechanisms in Section 3.

MalLo sends predictions from one location to another using a *multipath forwarding* approach, similar to an overlay network [1]. 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 2 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.

2. RELATED WORK

There have been a number of musical instruments built specifically for the Internet. In particular, Sarkar and Vercoe [16] built a predictive percussion instrument, which determines which pattern a sending performer is playing, and synthesizes it for the Receiver. Our system is inspired by theirs, but ours plays the same musical content on each end of the connection, while theirs might differ if the sender plays any material not in the learned repertoire. Somewhat similarly, Derbinsky uses reinforcement learning to allow the computer to learn drum patterns from a human performer, in order to act as a stand-in [9].

Barbosa provides an in-depth survey of Internet music collaborative systems that outlines the challenges of playing over the Internet, and it discusses previous uses of note messages as a means of communicating between distributed ensemble members [2] . Other works explore using the network itself as a resonant instrument [5, 4], while Tanaka created a hybrid instrument that uses both the Internet *and* physical space [18].

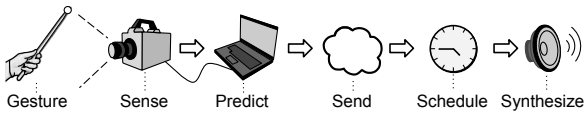

Figure 1: An example of a *distributed, synchronized musical instrument* (DSMI). The instrument is composed of various parts that exist in different locations. It uses prediction in order to synthesize the same audio in each location at the same moment. It capitalizes on the predictable path of a percussion mallet in order to accomplish this goal.

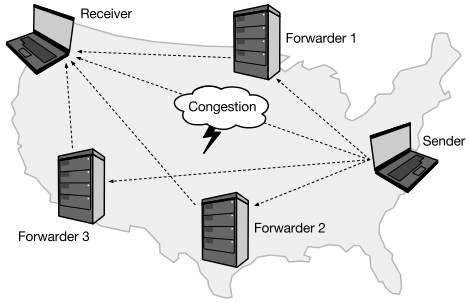

Figure 2: To reduce latency peaks, messages are sent along multiple simultaneous paths by sending both to the Receiver and to forwarding servers, in an attempt to route around local network congestion.

3. MALLO CONSTRUCTION

The key components of MalLo are (1) the sensing mechanism, (2) the prediction algorithm, which determines how the mallet path is interpreted by both the Sender and Receiver, and (3) the multipath forwarding network. The implementation of each component is described here, and the components are evaluated in Section 4.

We have built two versions of the MalLo sensing mechanism, one using a high-speed RGB camera and one using the Leap Motion sensor (Figure 3). The high-speed RGB version is more accurate and was used for all tests in Section 4. The Leap Motion version was built as a proof of concept, to show that it is possible to construct MalLo with inexpensive components.

3.1 High-Speed RGB Camera Version

The RGB camera version of MalLo has low latency and tight timing synchronization. The host for this system 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

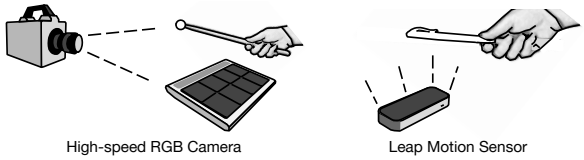

Figure 3: Two versions of MalLo. The high-speed RBG version uses a 200 fps camera, OpenCV to track the mallet head, and a MIDI drum pad to collect timing ground truth. The Leap Motion version version has no striking surface, and uses a yellow highlighter pen as a mallet. It is affordable and portable, but it has lower accuracy and higher latency.

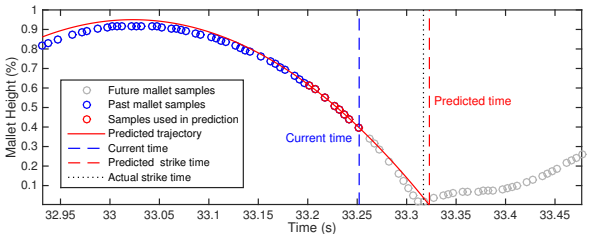

Figure 4: To predict the strike time of a mallet, we fit a second-degree polynomial to the previous 7 mallet locations. We compute the zero crossing to obtain the predicted time p_i of the strike, and the derivative of the polynomial at time p_i to obtain the velocity v_i.

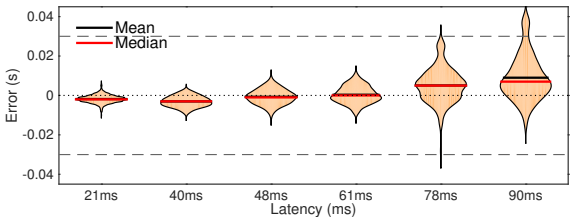

Figure 5: A pattern played at 120 bpm, with varying average network latency, accomplished by sending note messages to various locations around the world. Higher latencies created more variance in prediction accuracy. At 78 ms latency 99.6% of predictions are within 30 ms of the ground truth.

the minimum requirements for tracking percussion mallets shown in [13]. Our striking surface is a Roland SPD11 MIDI percussion pad, which is used to collect timing ground truth data for the experiments presented in Section 4.

The mallet head is tracked using OpenCV 2 libraries [3], via the template matching algorithm [17]. This search algorithm is the most computationally expensive operation in our pipeline. A typical search time is 5–15 ms, bringing the total processing time to 11–21 ms per frame.

3.2 Leap Motion Version

We built an inexpensive version that uses the Leap Motion as input. The Leap Motion is a small, portable commercial 3D sensor. It collects images using IR illumination and multiple cameras at a rate of 200 fps, and it has a transfer latency of 20 ms. 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. A companion video[1] submitted with this paper demonstrates the Leap Motion and RGB camera versions of MalLo in action. From this point on in the paper, all references to MalLo refer to the high-speed RGB version.

3.3 Prediction, Transmission, and Playback

We perform a new prediction for each new mallet height h_i (one per frame). If the mallet is descending, the Sender fits a second degree polynomial (quadratic regression) to the last k heights (7 in our tests) 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 is shown in Figure 4. Finally, p_i and v_i are sent in a message to the Receiver via the multipath forwarding network described in Section 3.4.

For each new message received, the Receiver determines if the message is newer than the last message received, and a user-specified inter-note wait time T_{min} has passed; if so, the message is processed. If it is an "unschedule" message, then all events from that Sender are removed from the note 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. When the note is played, t_0 is updated to p_i. The Receiver continues to replace older predictions with newer predictions until one is finally played. In doing this the Receiver assumes that prediction error monotonically decreases as the next strike approaches. In practice we found this to be true; each new prediction is almost always better than the previous.

[1]http://gfx.cs.princeton.edu/pubs/Jin_2015_MAD/

3.4 Multipath Forwarding Strategy

Naively sending the note prediction packet directly from the Sender to the Receiver over the Internet leaves our packet at the mercy of temporary congestion that may lie in the path chosen for it by the series of routers it encounters. To reduce the risk of sudden, severe congestion delaying receipt of the prediction, we have constructed a multipath forwarding system that uses 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 2.

4. EXPERIMENTS AND RESULTS

We have evaluated 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 a previously proposed version [13].

4.1 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 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 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 ground truth. Each sequence has approximately 30 notes.

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 20, 40, 47, 61, 73, and 88 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 4.3 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 [6], 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 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 99.6% of all notes. 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 [15].

For a closer look at the behavior of MalLo, Figure 5 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. 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.

Latency \ BPM	60	80	100	120	140
20.0 ms	100.0	100.0	100.0	100.0	100.0
40.0 ms	100.0	100.0	100.0	100.0	100.0
47.0 ms	100.0	100.0	100.0	100.0	100.0
61.0 ms	100.0	100.0	100.0	100.0	100.0
73.0 ms	99.8	99.6	99.9	99.6	99.6
88.0 ms	100.0	100.0	97.9	94.7	94.7

Table 1: A pattern of eighth and sixteenth notes was played back with various latencies and tempos. Shown are the percentage of notes predicted within 30 ms of the ground truth. 30 ms is only barely perceptible as non-synchronous.

Figure 6 shows the accuracy of predictions at a latency of 77.5 ms at various tempos. We see the same behavior as Figure 5, 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.

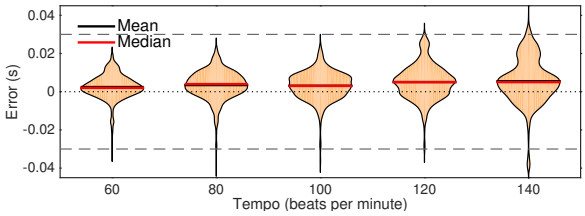

Figure 6: A pattern played at varying tempos, over 77.5 ms of average latency. Higher tempo causes more variance, but in the worst case of 140 bpm, 99.8% of predictions are within 30 ms of ground truth.

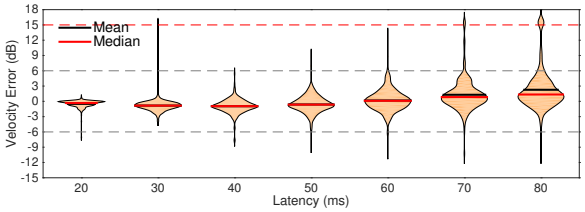

Figure 7: The difference between ground truth and predicted velocities (converted to probable sound pressure level when striking a rigid body). Performance is within an acceptable range up through 50 ms, but begins to degrade. At 80 ms, 5% of estimates are at least 15 dB louder than ground truth.

4.2 Velocity Accuracy

To test velocity accuracy, we compare the predicted velocity of the mallet with the measured velocity of the mallet at the time of impact (as derived from camera tracking data). Each measure of velocity is accomplished by fitting a quadratic to the mallet path. The difference is that the prediction is made before the actual impact, and the ground truth is measured at the exact time of impact. (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.) We use the same velocity error measure used by Oda et al. [13] which compares the relative sound pressure levels of the two strikes using an equation derived from Hertz's original study of rigid body vibrations [10].

Figure 7 displays the results. We found that velocity prediction was quite precise up through 50 ms of average latency. 60 ms of latency brings an audible difference in velocity, but not an extreme one. Performance then starts to degrade 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 not as important for synchronization.

4.3 Latency Peak Reduction

To test latency variation with and without the multipath 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 without the multipath network, are shown in Figure 8. (For this experiment we synchronized the sending and receiving servers' clocks using Network Time Protocol, which is only accurate within tens of milliseconds [12]. But it is not crucial that the clocks be synchronized exactly, because we are interested in the change of latency over time, not the absolute latency.)

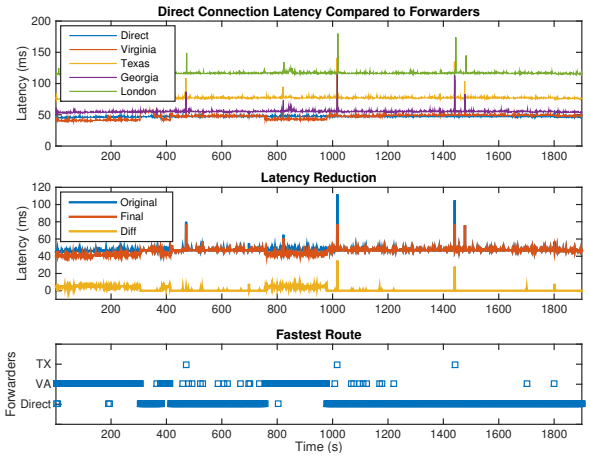

Figure 8: The multipath network helped to reduce latency peaks. The top plot shows measured latencies for different paths. The middle plot shows the original latency, the final latency, and the difference between them. Note that two large spikes around 1000 and 1500 seconds were reduced by roughly half. The lower plot shows the shortest route at each time point. The previously mentioned latency spikes were reduced by routing through Texas.

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 8 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.

4.4 Comparison To Previous Work

Our current algorithm improves on the previous prediction method used by Oda et al. [13] in the way it deals with network latency, even without considering the multipath forwarding strategy. 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.

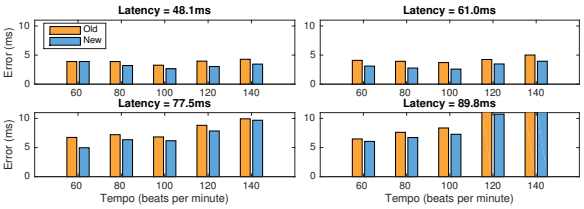

Figure 9: Our new prediction algorithm has a lower mean error in nearly every case than the method used in [13].

However, if the Sender estimates the network latency incorrectly, this can cause a note to be missed. Our current method sends *all* predictions (on a per-frame basis) as long as the mallet is descending at a certain speed. This 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 compare our current algorithm to Oda et al.'s method, we implemented a slightly-improved version of their algorithm that estimates the latency every 0.5 seconds, by sending a timestamped packet from Receiver to Sender and computing the one-way packet latency. We then model the latency as a Gaussian distribution with the packet latency as the mean, and use the 95th percentile in the algorithm latency estimate. We use 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 4.1. The results of this comparison are shown in Figure 9. At all levels of latency, the new algorithm did as well as or better than the old approach.

5. DISCUSSION

5.1 Widening the Radius of Collaboration

Oda et al. previously hypothesized that 40 ms of prediction at 120 beats per minute might be possible with an instrument such as MalLo, but our experiments have shown that nearly twice that is possible, in spite of using a relatively primitive prediction algorithm. This means a much larger radius of collaboration than expected. 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 can 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 [15] 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.

Our current prediction algorithm, which uses quadratic regression, is fairly primitive. We chose this approach to highlight that in this case it is relatively simple to predict these notes before they are played. We anticipate that more advanced algorithms would add to the system's range and stability.

5.2 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. At a latency of 100 ms, a musician is roughly limited to playing eighth notes at 150 bpm before prediction accuracy fails.

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 Design Philosophy

Internet latency will not significantly improve without fundamental changes to the architecture of the Internet. If we want to take full advantage of the Internet as a performance space we must design instruments that can be played despite its latency characteristics. In this work, we demonstrate a new approach to designing Internet-based instruments that preserves certain characteristics of most existing instruments—namely, the ability to be played in note-by-note synchrony by ensembles of musicians.

What properties of an instrument make it well suited to distributed, synchronous performance? What might be the shared design strategies for new DSMIs? These questions are wide open to future exploration. So far, it seems clear that an instrument can be made to be more predictable by slowing down the movements of a musician, or by restricting paths of movement in particular ways. One might be able to predict the fingering of a flute-like controller by placing pressure sensors on the keys, while also making them require a fair amount of force to press. This would give the system time to infer the final finger position. A kick drum pedal might also be a good candidate for pressure sensing and tracking, because of its long travel and constant contact with a foot. However, we do not yet know if this slowing of instruments would be enjoyable for musicians, or irritating, and we intend to explore this question in future work.

6. CONCLUSIONS AND FUTURE WORK

In this paper, 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 73 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. 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.

There are a number of new directions to explore using MalLo and DSMIs. While synchronization can be achieved with purely audio cues, it helps to have visual cues as well. In the future we would like to explore using the same prediction scheme to drive a visualization of the Sender's movements. Furthermore, the prediction strategies described here could be used not only for combating latency, but for local purposes, such as time correcting a note that is played late. They could also be used to enable expensive synthesis algorithms, such as simulating the sympathetically vibrating strings of a piano. Finally, of course, there is ample space for future work exploring how to construct new DSMI interfaces that best enable accurate prediction of musically expressive actions.

7. ACKNOWLEDGMENTS

This work was supported by the Project X fund at Princeton University. Much thanks to Huiwen Chang and Shuran Song for help in developing the MalLo application, and to Katie Wolf for valuable advice.

8. REFERENCES

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM Symposium on Operating Systems Principles*, 2001.

[2] Á. Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13, 2003.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV library*. O'Reilly, 2008.

[4] J. P. Cáceres and A. B. Renaud. Playing the network: the use of time delays as musical devices. In *Proc. International Computer Music Conference*, 2008.

[5] C. Chafe. Distributed internet reverberation for audio collaboration. In *Proc. AES 24th International Conference*. Audio Engineering Society, 2003.

[6] C. Chafe. Living with net lag. In *Proc. AES 43rd International Conference*, 2012.

[7] E. Chew, A. Sawchuk, C. Tanoue, and R. Zimmermann. Segmental tempo analysis of performances in user-centered experiments in the distributed immersive performance project. In *Proc. Sound and Music Computing Conference*, 2005.

[8] S. Dahl. Striking movements: A survey of motion analysis of percussionists. *Acoustical Science and Technology*, 32, 2011.

[9] N. Derbinsky and G. Essl. Exploring reinforcement learning for mobile percussive collaboration. *Proc. NIME*, 2012.

[10] H. Hertz. On the contact of elastic solids. *Math*, 92, 1881.

[11] A. Kapur, G. Wang, P. Davidson, and P. Cook. Interactive Network Performance: A dream worth dreaming? *Organised Sound*, 10, 2005.

[12] D. L. Mills. Internet time synchronization: The network time protocol. *Communications, IEEE Transactions on*, 39, 1991.

[13] R. Oda, A. Finkelstein, and R. Fiebrink. Towards note-level prediction for networked music performance. In *Proc. NIME*, 2013.

[14] P. Oliveros, S. Weaver, M. Dresser, J. Pitcher, J. Braasch, and C. Chafe. Telematic music: Six perspectives. *Leonardo Music Journal*, 19, 2009.

[15] P. Reinheimer and R. Will. WonderNetwork: Global Ping Statistics, Jan. 2015. https://wondernetwork.com/pings.

[16] M. Sarkar and B. Vercoe. Recognition and prediction in a network music performance system for Indian percussion. *Proc. NIME*, 2007.

[17] R. Szeliski. *Computer vision: Algorithms and applications*. Springer, 2010.

[18] A. Tanaka and B. Bongers. Global string: A musical instrument for hybrid space. In *Proc. Cast01//Living in Mixed Realities*, 2001.