http://cnmat.berkeley.edu/publications/audio-and-gesture-latency-measurements-linux-and-osx

Audio and Gesture Latency Measurements on Linux and OSX

Matthew Wright[*†], Ryan J. Cassidy[*], Michael F. Zbyszyński[†]

[*]Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
[†]Center for New Music and Audio Technologies (CNMAT), University of California, Berkeley
matt@{ccrma.stanford,cnmat.berkeley}.edu, ryanc@ieee.org, mzed@cnmat.berkeley.edu
http://ccrma.stanford.edu/~matt/latencytest

Abstract

We have measured the total system latencies of MacOS 10.2.8, Red Hat Linux (2.4.25 kernel with low-latency patches), and Windows XP from stimulus in to audio out, with stimuli including analog and digital audio, and the QWERTY keyboard. We tested with a variety of audio hardware interfaces, audio drivers, buffering and related configuration settings, and scheduling modes. All measured audio latencies tracked expectedly with buffer sizes but with a consistent amount of unexplained additional latency. With analog I/O there was also a consistent additional bandwidth-dependent latency seemingly caused by hardware. Gesture tests with the QWERTY keyboard indicate discouragingly large amounts of latency and jitter, but large improvements on Linux when real-time priorities are set.

1 Introduction and Prior Work

Modern personal computers have more than enough CPU throughput for sophisticated real-time sound synthesis and processing. The rise in popularity of digital audio recording onto personal computers has led to the commercial availability of affordable high-quality multi-channel audio interfaces. In our view the limiting factor on making computer-based instruments out of commercial hardware and standard operating systems is latency, defined as the total time from input stimulus to output response, and jitter, the variability of latency. A handful of milliseconds of latency and jitter can make the difference between a responsive, expressive, satisfying real-time computer music instrument and a rhythm-impeding frustration.

Characterizing the acceptable ranges of latency and jitter for various forms of music is beyond the scope of this paper. Psychoacoustic experiments on temporal acuity indicate that changes in the inter-onset times of pairs of events played at nearly the same time produce audible timbral differences at around 1ms (Henning and Gaskell 1981, Ronken 1970); that would be one criterion for determining an acceptable amount of jitter. Chafe et al. measured the effect on ensemble tempo for a simple rhythmic task when performers were separated by various latencies (as in wide-area networking); they found significant deterioration around 20 ms one-way latency (Chafe, et al. 2004). One author's personal experience (Wright 2002) indicates that a latency of 10 ms with 1ms of jitter is a good goal.

Although latency measurements of specific hardware/software combinations become irrelevant rather rapidly, the history of this work is important both for methodology and to see if things are getting better or worse over time. Dannenberg et al. (Dannenberg, et al. 1993) measured the jitter in the MIDI output from a multithreaded application trying to produce a MIDI event exactly every 10 ms under Mach 3.0; they found it to be within the range of error of their measurement system (< 0.7 ms) when not connected to the network but 2.6ms with the network connection. This shows the importance of measuring latency under different system load conditions. Brandt and Dannenberg later repeated this kind of experiment for Windows NT4, Windows 95, and Irix 6.4 (Brandt and Dannenberg 1998), finding significant fractions of events delayed by tens of milliseconds, especially with increased CPU load.

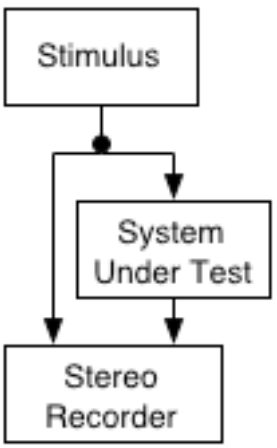

Figure 1: The Stereo-Digital-Recorder Paradigm

The principal contribution of Freed, Chaudhary, and Davila (Freed, et al. 1997), was a simple circuit implementing a near-zero-latency transcoder from Ethernet or MIDI to a short audio "blip." This hardware allows a

standard stereo digital sound recorder (which could of course be another computer) to be used to record the stimulus in one channel and the response in the other for easy measurement of latency. We will call this the "stereo-digital-recorder paradigm," illustrated in Figure 1. There are many advantages to this paradigm:

- It tests the system under normal conditions (no added profiling software, etc.)
- All latencies in the stimulus, recording, and measurement systems are applied equally to the two channels, so they will cancel differentially.
- It provides an indisputable measure of total system latency not based on any software's opinion of the current time.

MacMillan et al. used the same paradigm to perform a comprehensive set of experiments of audio latency, passing a single impulse via an analog interface through a trivial "audio through" program on each system (MacMillan, et al. 2001). They measured Linux 2.2 and 2.4, MacOS 8, 9, and X, and Windows 98, ME, and 2000 on various CPUs, audio hardware, and sound I/O APIs, with results ranging from 2.72 to 935 ms.

Recent work by Nelson and Thom (Nelson and Thom 2004) measured MIDI latency under Linux, Mac OS X, and Windows with their own MIDI-to-audio transcoder using the stereo-digital-recorder paradigm. Because their work is so recent, thorough, and high quality, we have decided not to measure MIDI performance in the present work. They conclude that MIDI is capable of performing with nearly imperceptible timing errors, but that performance is highly contingent on the specific MIDI data and the configuration of the system. CPU load and rapid MIDI "bursts" yielded a marked increase in latency and jitter, with worst-case results of 36.4 ms and 34.4 ms, respectively.

2 Components of Systems to be Tested

A system's total audio latency (from audio input to audio output) should be the sum of the latencies of these components:

- Phase delay of analog anti-aliasing and DC-blocking filters
- Digital buffering internal to audio input hardware
- Buffering of input samples in audio driver and API
- Buffering inside the application
- Latency of the application itself (e.g., to get in and out of the frequency domain)
- Buffering of output samples in audio driver and API
- Digital buffering internal to audio output hardware
- Phase delay of reconstruction (and possible DC-blocking) filters in D/A converters

For gestures, these are the components of the latency until the application receives the input:

- Latency and jitter of the sensors themselves
- Latency induced by smoothing and other signal conditioning
- Transmission delay of the sensor measurements
- Operating system latency to process sensor measurements (possibly including context switching, inter-thread communication, etc.)

Ideally, we would like to quantify each of these latencies individually. Unfortunately that is not practical for two reasons. First, some of these data paths cannot be accessed externally (e.g., the buffering internal to audio input hardware). Second, profiling and measuring times within the system being tested often changes the overall performance of the system.

3 The Systems We Tested

- Macintosh 1.25 GHz Powerbook G4, MacOS 10.2.8, Max/MSP 4.3, built-in audio hardware via CoreAudio.
- 3 GHz Dell Pentium 4, Windows XP home edition version 5.1 (Build 260.xpsp2.030618-0119: Service Pack 1), Max 4.3.2, Mark of the Unicorn ("MOTU") 828 Firewire audio interface (driver version 3.0) via a PCI Firewire card.
- 2.7 GHz Pentium 4, Red Hat Linux 9, 2.4.25 kernel with Andrew Morton's low-latency patches and Robert Love's preemptible kernel patches. (Phillips has an excellent description of how to tune Linux for real-time performance (Phillips 2003).), M-Audio Omni I/O.

4 Audio Latency

4.1 Audio Latency Test Method

We measured total latency from audio input to audio output with the stereo-digital-recorder paradigm. In our terminology the *stimulus* channel is the one that went straight from the sound source to the recorder and the *response* channel is the one that went through the system under test.

For digital audio tests we generated a test signal of ideal impulses (single-sample maximum-amplitude values embedded in a stream of zero-amplitude samples) and sent it out over S/PDIF. We used a Z-Systems z-8.8a Lightpipe Digital Detangler to output two copies of the S/PDIF signal. One went through the system under test back into the Detangler; the second copy went through a short S/PDIF cable back into the Detangler. The two incoming S/PDIF streams were merged into a single 8-channel ADAT Lightpipe stream, which went to an 8-channel recorder. We carefully synchronized the entire system to a single sample rate clock.

For analog audio tests the splitting and merging of the separate channels was, of course, much easier. We felt that

ideal impulses would be likely to excite undesirable behavior in the analog parts of the system, particularly the anti-aliasing filters in the A/D converters and the reconstruction filters in the D/A converters. So instead we synthesized band-limited impulse trains ("BLITs") consisting of a sum of equal-amplitude, zero-phase cosines at harmonics of 2.0 Hertz. For early prototyping we synthesized BLITs with only 50 harmonics, i.e., with a bandwidth of 100 Hertz. Later we synthesized wider-bandwidth BLITs and were surprised to find that the BLIT bandwidth had a large effect on audio latency; these results are discussed in the next section.

We wrote matlab programs to analyze the resulting stereo sound files and produce a latency measurement per impulse. For digitally transferred ideal impulses this is trivial; just see how many samples each impulse is delayed in the response versus the stimulus.[1]

For the analog tests we had to contend with noise and imperfect transfer of the test waveform. We passed both channels through a matched filter whose impulse response was one impulse of the synthesized BLIT from zero-crossing to zero-crossing around an amplitude maximum. This filter concentrates the signal's energy at the instant of the impulse so that each local amplitude maximum will be at the "right" time even in the presence of noise. Although this filter introduces a delay, it is the same for both channels, so the measured latency is unaffected.

The analysis software then finds each local maximum of amplitude and fits a parabola to it and the two adjacent points to find the "instant" of the peak with sub-sample accuracy. The two lists of time instants for each peak are then just subtracted pointwise to get the list of measured latencies. For each series of band-limited impulses, the variation of measured latency among all impulses was always within one sample and often under 0.1 samples; in other words, jitter seems to be negligible for audio through (as we expected).

4.2 Bandwidth-Dependent Analog Delay

When the transmission path of the test signal included the A-to-D and D-to-A converters of the machine under test, we observed a significant increase in overall latency with increasing impulse train bandwidth. For example, Figure 2 shows two time-domain waveform plots: the upper plot shows a stimulus and response impulse band-limited to 100 Hz, and the lower plot shows a stimulus and response impulse band-limited to 125 Hz. Figure 3 shows plots for impulse train stimuli band-limited to 100 Hz and 22050 Hz (respectively). In both figures, the latency clearly increases with higher impulse train bandwidth. The audio device was set to the "hardware-through" test configuration, which relays the audio data from its input directly to its output, bypassing any transmission along the peripheral bus of the machine to/from the CPU or main memory.

To verify that our matched-filter latency measurement technique was correctly showing that the A-to-D and D-to-A converters were responsible for this increase in latency with increasing BLIT bandwidth, we repeated the experiment using digital audio connections and the same matched filter technique. Because the matched filter technique yielded identical latency regardless of impulse-train bandwidth used for each run of the test, we concluded that there was no bandwidth-dependence in the all-digital test configuration. This increases our confidence that the measured bandwidth-dependent latency is a result of the conversion to/from the analog domain, as opposed to a flaw in our latency measurement method.

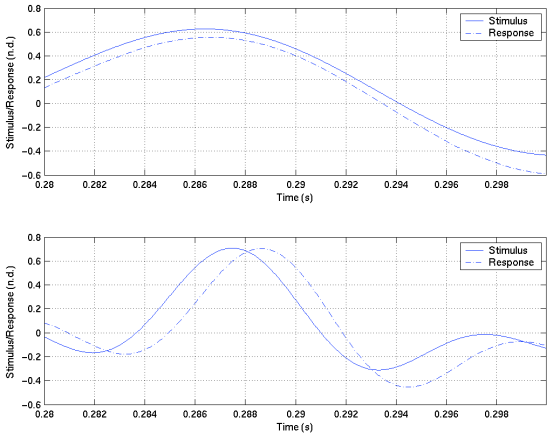

Figure 2: Plots of stimulus and response impulses band-limited to 100 Hz (upper plot) and 125 Hz (lower plot). (Linux PC using an M-Audio Omni I/O, HW through)

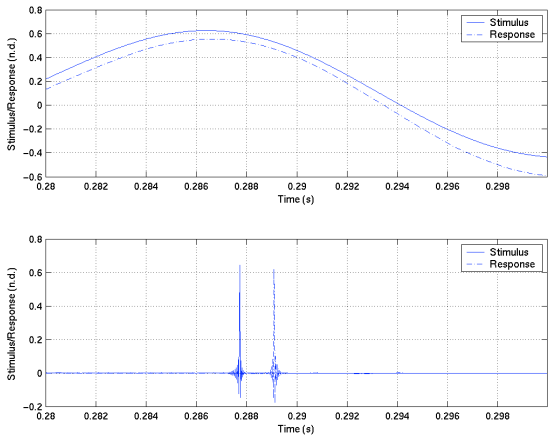

Figure 3: Plots of stimulus and response impulses band-limited to 100 Hz (upper plot) and 22050 Hz (lower plot). The test configuration and procedure is identical to that of Figure 2. The increase in latency from the 100 Hz case to the 22050 Hz case is readily apparent.

[1] We assume that no bit or sample errors occur in the all-digital transmission and did not, in fact, find any such errors.

We next wanted to determine if any software component was introducing bandwidth-dependent latency in addition to that introduced by the converter hardware. We ran exhaustive tests for a given audio device (in this case, the M-Audio Omni I/O running on a Linux computer) using a variety of combinations of audio applications (e.g. Pd, Qjackctl), audio APIs (e.g. ALSA, JACK), and audio buffer settings. The results (for a total of 19 application-API-buffer combinations) are shown in Figure 4. For each line plotted, the latency measured with the 100 Hz BLIT was subtracted from the corresponding latencies for the higher-bandwidth BLITs to ensure all plots would be vertically aligned for fair comparison. The figure shows that the effect of all software on the bandwidth-dependence is negligible. (It also shows the amount of bandwidth-dependent extra latency as a function of bandwidth.) We concluded that the frequency- or bandwidth-dependent latency observed has been introduced solely by the converter hardware in the test set-up.

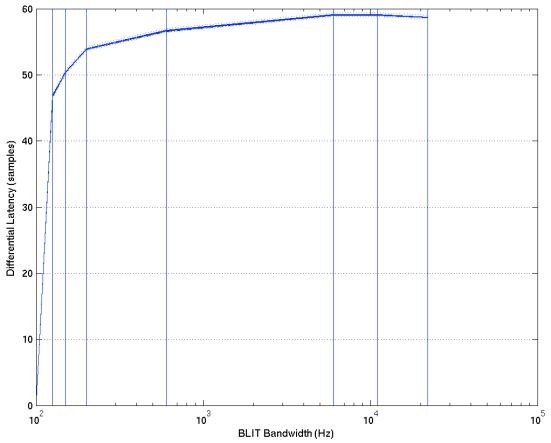

Figure 4: Plot showing differential latency (relative to latency measured for the 100 Hz BLIT) versus BLIT bandwidth for 19 different audio software configurations on a Linux PC using a M-Audio Delta 1010 audio device. The BLIT bandwidths used are denoted by the vertical lines on the plot: 100 Hz, 125 Hz, 150 Hz, 200 Hz, 600 Hz, 6000 Hz, 11025 Hz, and 22050 Hz. The tight packing of the lines on the plot indicates the independence of bandwidth-dependent latency and audio software configuration, implicating the audio hardware as the sole source of bandwidth-dependent latency.

We re-ran these tests with two other audio interfaces, a MOTU 828 and the built-in audio on a Macintosh Powerbook G4. We found similar curves but with different overall amounts of bandwidth-dependent latency, as shown in Figure 5.

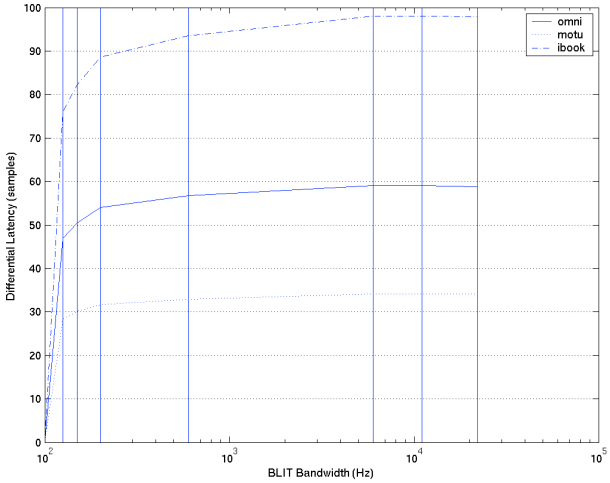

Figure 5: Same as Figure 4, but comparing three analog audio interfaces: M-Audio Omni I/O (solid line), MOTU 828 (dotted line), and Powerbook G4 built-in audio (dashed line).

4.3 JACK Latency

On the Linux platform, we measured the latency introduced by the JACK API for various buffer sizes. Table 1 shows the results, with the expected buffer-size-related latency subtracted out. The non-negligible delay introduced by the converter hardware (present in the analog case but not in the all-digital case), as discussed above, is clearly apparent from the results. Otherwise, the latency tracks quite consistently with buffer size.

JACK Buffer Size	Expected latency from buffers	Measured extra latency: digital	Measured extra latency: analog
64	128	2.4	59.4
128	256	2.2	58.8
256	512	2.4	58.3
512	1024	2.4	59.3
1024	2048	3	58.8
2048	4096	2.6	58.8

Table 1 - Latency in samples for JACK audio API running on Linux platform with various buffer sizes. We expect that there will be latency equal to twice the buffer size; this table shows "extra" latency not accounted for by the buffers. The all-digital pass-through test was repeated 5 times, with the average of results shown. The analog pass-through test was conducted 2 times for each buffer setting. For the digital tests, an ideal impulse train was used, and for the analog tests, an impulse train band-limited to half the sampling rate (near ideal) was used.

Using the hardware-through configuration discussed in the previous section (i.e., with the Linux audio device configured to directly relay its audio input to its audio

output), the measured latency for the all-digital pass-through was either 4 or 5 samples in each run of our tests. Clearly even with all-digital I/O there are still a handful of samples of extra latency.

4.4 Mac/Windows Latency Measurements

We also applied our tests to Max/MSP running on Windows (using the ASIO audio API) and Macintosh (using the Core Audio API), with results shown in table 2. Again, there is a consistent extra latency beyond that expected by the buffer size.

Buffer Size	Expected latency from buffers	Measured extra latency: OSX, built-in audio	Measured extra latency: Windows, MOTU828
32	64	95.4	
64	128	95.4	
96	192		86.9
128	256	95.4	86.9
192	384		86.8
256	512	95.5	86.8
512	1024	95.6	86.6
768	1536		86.4
1024	2048		86.2
1536	3072		85.8
2048	4096		85.4

Table 2 – Analog audio-through latency measurements for Max/MSP under OSX 10.2.8 using the built-in audio device of a Macintosh PowerBook and under Windows using the MOTU 828 firewire audio interface. All numbers are in samples; "extra" latency means measured total latency minus the expected latency.

5 Gesture to Audio Latency

5.1 Gesture to Audio Test Method

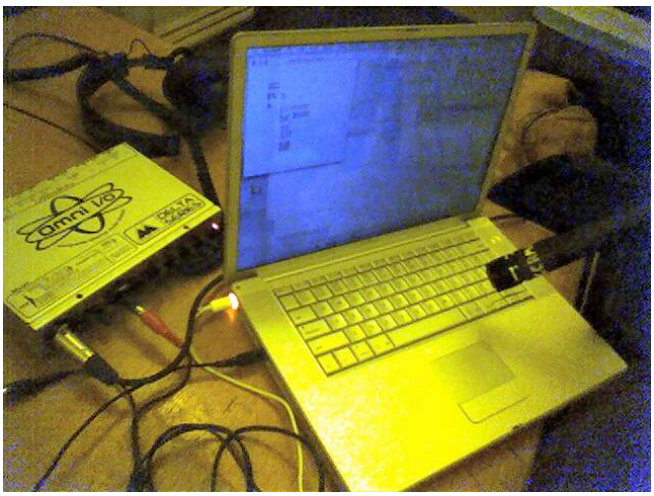

Figure 6: Gesture-to-Audio Latency Test Setup

We measured gesture-to-sound latency with input from QWERTY keyboards. As Figure 6 shows for the case of testing an Apple Powerbook G4, we pointed a microphone directly at the keyboard to record the stimulus: the acoustic sound of physically hitting the key. We tried to strike each key quickly and sharply with a fingernail so as to produce the most impulsive possible sound. The microphone was placed just a few centimeters from the location of the striking of the keys, so as to be able to ignore the speed of sound propagation.

For these tests, the computer was running a trivial program that produces a 1K sinusoid with an amplitude envelope that goes instantly to full volume at the start of each note and then decays quickly. Since the frequency is 1K, the period is 1ms, so any error from not knowing the initial phase will be less than a millisecond. Figure 7 shows the Max/MSP version of this program. The resulting stereo sound files look like the one shown in Figure 8.

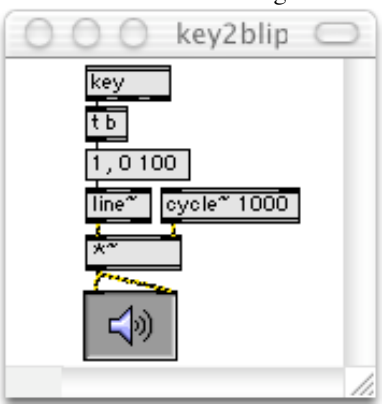

Figure 7: "Keypress to Blip" program in Max/MSP

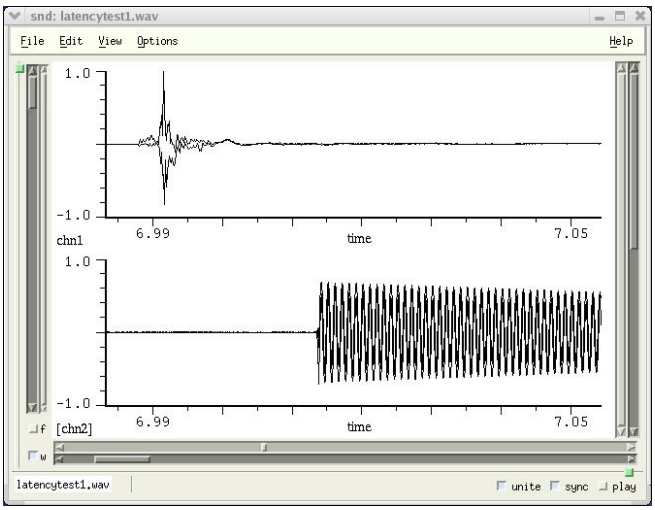

Figure 8: Gesture stimulus and response viewed in a sound-file editor

As one can see from Figure 8, although it is quite obvious where the response begins (namely, the first sample above the noise floor), it's not so easy to pick a single

instant as the time of the stimulus. We used the local amplitude maximum, which is not necessarily the moment that the finger touched the key, but seems close enough. In every case, the total duration of the stimulus signal was only about 14 ms per event, so that gives one upper bound on the jitter that this measurement method could be adding to our results. We tightened our upper bound on measurement error by testing a commercial drum machine (Kawai R-100) that plays a sample (with presumably very low latency and jitter) when each "pad" is pressed. Since these latency results ranged only from 4 to 8ms, even if we assume that this device has zero jitter, we know that our measurement error cannot be more than 4ms.

5.2 Gesture to Audio Results

Because there was substantial jitter in our gesture-to-audio measurements, we display the results as histograms rather than in tables summarizing average latency. (Underneath each histogram we also give the mean and standard deviation.)

Figure 9 compares the QWERTY keyboard to sound output latency for Max/MSP's three scheduler modes: no overdrive, overdrive, and "Scheduler in Audio Interrupt." This was Max/MSP 4.3 under MacOS 10.2.8 on a 1.25 GHz Powerbook G4 running with 64-sample I/O (and signal) buffers, on an unloaded system. We see that all are about the same except that with overdrive there are occasional outliers.

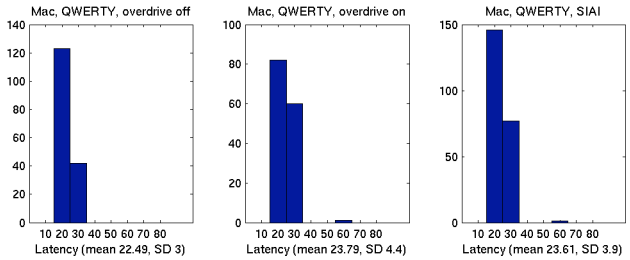

Figure 9: Histograms of latency of QWERTY keyboard to audio out latency for Max/MSP 4.3 under MacOS 10.2.8 on an unloaded machine with 64-sample IO and signal buffers and various scheduling modes.

Figure 10 shows the results of Pd with ALSA. We see the expected decrease in average latency as the stated latency goes from 50 to 8. The "-rt" flag has an enormous improvement on latency and especially on jitter. Figure 11 shows the results of Pd with Jack; we see that reducing Jack's buffer size improves the latency more than should be expected and also makes a large improvement on the jitter.

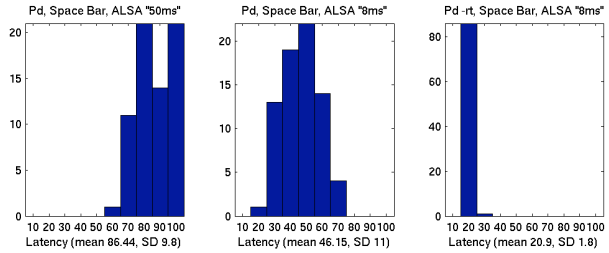

Figure 10: Histograms of latency of QWERTY keyboard to audio out latency for Pd under Linux (configured as described in section 3), using ALSA. Pd provides control of ALSA's buffer size in units of milliseconds; the left histogram shows the behavior with Pd set for 50 ms of ALSA latency (Pd's default), the middle histogram with Pd set for 8 ms of ALSA latency (the minimum reliable value), and the right histogram with Pd set for 8 ms ALSA latency but invoked with the "-rt" flag to use "real-time priority."

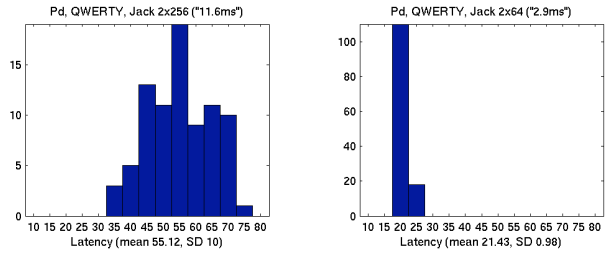

Figure 11: Histograms of latency of QWERTY keyboard to audio out latency for Pd under Linux (configured as described in section 3), using Jack. The left histogram is for Jack using 256-sample buffers; the right is for 64-sample buffers.

6 Future Work

The frequency-dependent latency of the converter hardware discovered in the course of this work is especially noteworthy. Further investigation is needed to more precisely characterize the magnitude and phase of the converter frequency responses. For this task, system identification using pseudo-noise/pseudo-random sequences provides an alternative to direct impulse-based tests that offers increased noise-robustness (Rife and Vanderkooy 1989). Yet another pseudo-noise-based technique makes use of Golay codes (Foster 1986). Accurate characterization of converter transfer-functions should help to explain not only the frequency-dependent latency we studied in this work, but also future frequency-dependent effects introduced by converter hardware and filters.

We would like to measure and analyze many more situations:

- Variable CPU loads
- Event stimuli via Ethernet and MIDI (using an audio transcoder)
- Other Macintosh audio applications including Pd, Supercollider, a custom CoreAudio application, etc.

- ♦ Firewire audio devices on the Macintosh
- ♦ Built-in sound cards on Windows
- ♦ Other computers (slower, SCSI, etc.)
- ♦ QWERTY via an external USB keyboard
- ♦ Other gestural inputs such as mice, Wacom tablets, joysticks, etc.
- ♦ Tapping into the electrical output of a QWERTY keyboard rather than using the microphone to detect the stimulus.

There is a website for this latency testing project.[2] As we continue to make these measurements we will continue to post the results there. Also, all of our matlab software for measuring, analyzing, and plotting the latency from stimulus/response stereo sound files is available there; we encourage other people to make the same kinds of measurements and analyze them with our methods.

7 Conclusion

We have presented latency measurements for the transmission of audio and gesture data using desktop, or general-purpose, computer platforms. Regarding audio latency, it was first reassuring to note that latency was approximately equal to twice the buffer size of the audio hardware. With the exception of a few excess samples, the audio latency was found to be equal to twice the buffer size (as expected) when the digital interface of any audio device we tested was used. When using analog interfaces, however, significant, frequency-dependent latencies were found, in excess of the nominal (i.e. twice-buffer-size) value. This is hypothesized to be a result of the converter hardware (e.g., DC-blocking circuitry) involved (as discussed, further characterization is possible). For QWERTY keyboard input, latencies were generally much larger than what one would expect from the audio buffer sizes. Proper configuration of Linux real-time priorities was found to make a large improvement in both latency and jitter.

8 Acknowledgments

Adrian Freed, Jay Kadis, Fernando Lopez-Lezcano, Max Mathews, Julius Smith, William Verplank.

[2] http://ccrma.stanford.edu/~matt/latencytest

References

Brandt, E. and R. Dannenberg (1998). "Low-Latency Music Software Using Off-the-Shelf Operating Systems." In *Proceedings of the International Computer Music Conference*, pp. 137-141. Ann Arbor, Michigan: International Computer Music Association. (http://www-2.cs.cmu.edu/~rbd/papers/latency98/latency98.htm)

Chafe, C., M. Gurevich, G. Leslie and S. Tyan (2004). "Effect of Time Delay on Ensemble Accuracy." In *Proceedings of the International Symposium on Musical Acoustics*. Nara, Japan.

Foster, S. (1986). "Impulse response measurements using Golay codes." In *Proceedings of the IEEE ICASSP-86*, pp. 929-932. Tokyo: IEEE.

Freed, A., A. Chaudhary and B. Davila (1997). "Operating Systems Latency Measurement and Analysis for Sound Synthesis and Processing Applications." In *Proceedings of the International Computer Music Conference*. Thessaloniki, Hellas: ICMA. (http://cnmat.CNMAT.Berkeley.EDU/ICMC97/papers-html/Latency.html)

Henning, G. B. and H. Gaskell (1981). "Monaural phase sensitivity measured with Ronken's paradigm." *J. Acoust. Soc. Am.* 70, 1669-1673.

MacMillan, K., M. Droettboom and I. Fujinaga (2001). "Audio Latency Measurements of Desktop Operating Systems." In *Proceedings of the International Computer Music Conference*, pp. 259-262. Habana, Cuba: International Computer Music Association.

Nelson, M. and B. Thom (2004). "Interactive MIDI: Real-time Performance Evaluation." In *Proceedings of the New Interfaces for Musical Expression*. Hamamatsu, Japan.

Phillips, D. (2003). "Computer Music and the Linux Operating System: A Report from the Front." *Computer Music Journal* 27(4), 27-42.

Rife, D. D. and J. Vanderkooy (1989). "Transfer-Function Measurement with Maximum-Length Sequences." *J. Audio Eng. Soc.* 37(6), 419-444.

Ronken, D. (1970). "Monaural detection of a phase difference between clicks." *J. Acoust. Soc. Am.* 47, 1091-1099.

Wright, M. (2002). "Problems and Prospects for intimate and satisfying sensor-based control of computer sound." In *Proceedings of the Symposium on Sensing and Input for Media-Centric Systems (SIMS)*, pp. 1-6. Santa Barbara, CA. (http://cnmat.CNMAT.Berkeley.EDU/Research/SIMS2002/Wright-SIMS2002.pdf)

Wright, M., D. Wessel and A. Freed (1997). "New Musical Control Structures from Standard Gestural Controllers." In *Proceedings of the International Computer Music Conference*, pp. 387-390. Thessaloniki, Hellas: ICMA. (http://cnmat.CNMAT.Berkeley.EDU/ICMC97/papers-html/Tablet.html)