

Linking Music Students and Music Instructors Via The Internet

Roy Vanegas
Hunter College
City University of New York
rvanegas@hunter.cuny.edu

November 14, 2005

Abstract

There are two established methods by which one can learn to play a musical instrument: through self-study or a personal instructor. Employing the former is convenient, but more challenging. One studies at home, but forfeits the personal attention that an instructor provides. Conversely, the latter approach, while less convenient, makes the pedagogical experience more rewarding: by working closely with the instructor, the student is given the important personal attention he or she needs. I propose a third alternative, encompassing the advantages of each of the previously mentioned methods in a near-real time didactic method: teaching and learning to play a musical instrument via the Internet using MIDI, Musical Instrument Digital Interface. By combining MIDI-equipped instruments, specially-designed software, and a computer connected to the Internet, a student and an instructor can simulate a typical student-teacher environment, regardless of distance. This new approach to musical instruction, which is a vast departure from commonplace educational techniques, is perhaps the next logical progression in musical pedagogy.

1 Introduction

1.1 New Pedagogical Approach

There are many software packages on the market that make use of MIDI for single-person instruction, e.g., Practica Musica, Music Study, eMedia's Guitar Method, etc., and my review of the literature revealed many instances in

which the Internet had been used for collaborative purposes [Hajdu, Georg] [Chew and Sawchuck] [Young and Fujinaga], but I was unable to locate research projects or market-based software programs in which the Internet *and* MIDI were being used in unison for one-on-one musical instruction.

Potential results will demonstrate the benefits of distance music education using MIDI and the Internet. For the academic community, from grade school to the university level, “virtual classes” could be offered to students located as nearby as the next building, or as far off as a different continent. On an individual level, private instructors could make their services available to a much wider student pool. With the constraints imposed by distance removed, I theorize that many people will either supplement their current course of music study, or simply start. And because I intend to prove favorable results using a slow Internet connection, like that provided by a 56kbps modem, users without high-end bandwidth capabilities will also be able to use this new medium for musical instruction.

1.2 MIDI

Established in 1982 by two major synthesizer manufacturers, the now-defunct American company Sequential Circuits and Japan’s Roland Corporation, MIDI was designed as a simple protocol for transferring control messages to and from hardware-based synthesizers and musical instruments. Today, MIDI’s presence is ubiquitous: it is used in virtually every recording and movie studio throughout the world, and has even ventured out into areas for which it was never designed, such as in theater lighting applications and in video rendering effects, to name a few.

2 MIDI vs. Audio

Sound files are quite large. A mere five seconds of sound converted to Audio Interchange File Format (*aiff*), Apple Computer’s proprietary lossless audio format, is equivalent to 1MB in size. On the other hand, a 6-minute 36-second MIDI file is only 36.9 kilobytes in size. In addition, ordinary sound files are simply too cumbersome to transmit over the Internet in near-real time without the use of high-end equipment [Chew and Sawchuck] or specially-designed software [Cooperstock and Spackman]. Without these devices, latency becomes a serious problem.

MIDI, on the other hand, proves to be an ideal tool for a research undertaking in which latency must remain at zero or near-zero levels. It is small: eight bits per MIDI data byte, and three bytes make up a MIDI message. For

example, a “Note-on” message consists of one byte to indicate the note to play, another byte to denote how loud to play it, and one last byte to specify when to release it. A message like this roughly equals the size of a text file containing three characters. Because it is so compact, MIDI data can travel faster than audio across the Internet, and research has shown that a file 6656 bytes in length and about 1 minute and 30 seconds in duration may be transmitted in about “2 seconds” using a 28.8 modem [Faber, Orlarey, and Letz].

MIDI’s ability to capture a musician’s natural expression with good precision is another reason why it was chosen for this project. And that expression will be translated into visual and audio representations of each person’s performance. On one end, the instructor will be able to analyze the student’s overall “feel” and technical ability, then provide feedback on the student’s performance. And on the other end, the student will have the ability to respond to the instructor’s corrective action by either playing along or simply archiving the lesson for later use. Both the student and the instructor will reap the benefits of a two-fold lecture experience without being in the same room.

Clearly, MIDI is the better choice for use in this research.

3 The Transport Layer

There are numerous ways in which to transmit data between computers using the Internet, but the two primary transport layers are Transmission Control Protocol and User Datagram Protocol, TCP and UDP, respectively.

3.1 TCP vs UDP

Regarded as “reliable,” TCP is the primary protocol for Internet traffic [Stevens]. It ensures the arrival of data, within a timeframe that could span minutes, at its destination in the order in which it was sent from the source. Regrettably, its strength for most Internet purposes is its weakness for this one: we simply cannot wait more than fifty or sixty ms for data to arrive at its intended source. (However, during the early developmental stage of this project, Java’s TCP networking functionality will be used to test and debug the interface.)

UDP, on the other hand, may behave faster, and previous research has proven this [Young and Fujinaga]. Cooperstock and Spackman, Young and Fujinaga, and Foss and Mosala have all favored the flexibility of the UDP protocol in their experiments transmitting MIDI either over the Internet or over a local Ethernet connection.

4 Software

It is now time to explain the software interface, or front-end, of the application.

4.1 Java

Java contains a rich set of methods for MIDI manipulation by way of its `javax.sound.midi` package, and provides graphics and GUI packages for the creation of front-end interfaces. Together, with its relatively easy cross-platform capabilities, Java, like MIDI, proves to be well-suited for the purpose of this project.

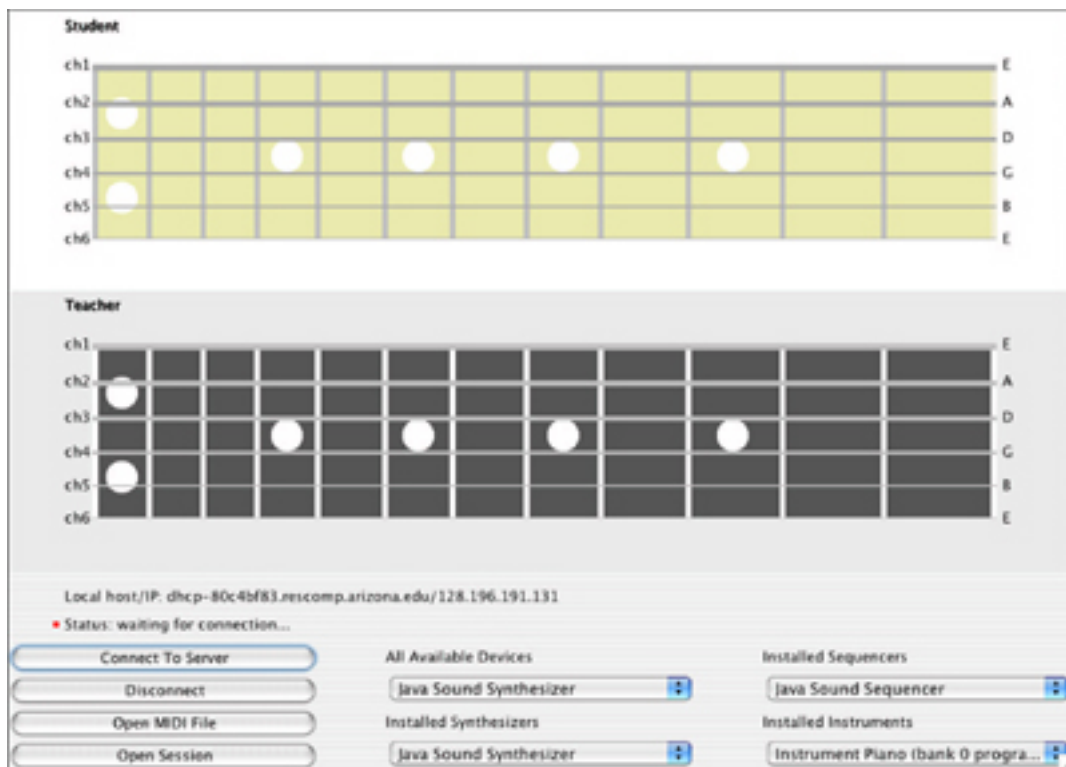
4.2 The Software's Current State

In a two-month span, I probed the complexities of network engineering and the intricacies of graphics programming using Java's aforementioned capabilities, and thus began a software implementation of the program, focusing on the graphical user interface, or GUI, of the program. As of this writing, the state of the software implementation is as such:

- **Fretboard:** A graphics-implemented simulation of a guitar's fretboard in two octaves from frets 1 to 12 appears at the top-center of the screen. Each note on the fretboard of the guitar has had its coordinates mapped to specific MIDI note numbers. As MIDI data arrive, each audible note will trigger its mapped region on the fretboard (MIDI triggering has yet to be implemented).
- **Host Name And IP Address Of Local Machine:** The server-teacher version of the program waits for a connection on its local host port, providing the host-IP data in the interface for the student-client version of the software to use for binding the Internet connection. (The student-client version of the software will not provide this information in its interface, since a client merely binds, while a server waits to bind.)
- **Multiple Buttons:** There are currently four buttons that appear in the interface, and not one is attached to a `listener`, a device in the Java language that triggers mouse clicks and perform other actions. In the final product, however, each button will behave as such:
 - **Connect To Server:** This button will only be present in the client version of the program, and upon a click event, will seek out the host machine and bind the client to the host.

- **Disconnect:** This button will be present on both the client and server versions of the program, and will simply break the connection between the client and its host. Once the connection is broken, any MIDI data that was bound via the network connection to the auxiliary guitar neck image will be remapped to MIDI files on the user’s local machine for playback or analysis (in the instructor’s case). For example, a student can load a MIDI file into the program and that file will trigger audio and any note mappings on the disconnected guitar in the interface. This is analogous to a player piano in which a round toothed device triggers the piano keys.
 - **Open MIDI File:** This button will only be active when a connection with the student or teacher is not active.
 - **Open Session:** A session will be comprised of MIDI data, in the form of a lesson, original tempo in BPM, or beats per minute, and the student’s or instructor’s original MIDI performance. This function is analogous to playing back a previously recorded videotape of a lesson, but without the performance editing capabilities.
- **Status Indicator:** A small floating box located in the westward area of the panel window will appear in red if no connection is made with a server or client, or green upon a successful connection. Text to its immediate right will also indicate a bound or unbound connection.
 - **Available Devices:** The program queries the local system for available MIDI devices, then displays their availability in a drop-down box for the user to choose.
 - **All Available Devices:** Java is capable of probing the local machine for all present MIDI devices. The program employs this capability, locating all hardware items, including MIDI IN and MIDI OUT ports, and reports the array within this drop-down menu.
 - **Installed Synthesizers and Installed Sequencers:** These report, in different components, the same data that the “All Available Devices” button reports, and are currently present for debugging purposes.
 - **Installed Instruments:** A default soundbank is acquired by the program and bound to the `Synthesizer` object presently active, and that soundbank is tied to an array of `Instrument` objects. All of those instruments are displayed in a drop-down menu from which the user may choose.

The project was developed using version 1.4.0.01 of the Java Software Development Kit on the Apple Macintosh's 10.2.8 operating system platform.



4.3 Next Phase

Many aspects of the software are lacking. First, MIDI transmission, using TCP, needs to be established, either by using raw MIDI data or by extracting `String` objects from MIDI messages that will be parsed at the local level then re-assembled at its destination. This will be implemented in the beginning stages of testing and debugging, and once the program is sufficiently robust, TCP will most likely be abandoned as the transport protocol in favor of an alternate protocol.

The protocols considered for this project's long-term are UDP and Open-Sound Control, a protocol "optimized" for "multimedia devices" that has proven to perform well in projects of this nature [Freed, Momeni, and Wright].

Once the program is ready, the next phase of the research will be carried out with two MIDI-capable instruments: a guitar and electronic synthesizer. Each instrument will be connected to a computer that is linked to the Internet, and both computers will share the same IP address via a network hub. I will be the main beta tester, running the computers and instruments in the

same room. Other than a standard MIDI interface, no *special* devices will be needed to carry out the research, and no hardware devices will need to be altered; software will be the driving force behind this project.

In the next and final stage, the software will be put to use in the real world by one or two users over the course of perhaps six months to a year, at which point I will make an analysis of both the student's progress and the software's evolution.

5 Limitations

It is not until the latter stages of the project, in which an alternate protocol has been established and fully-implemented, that the greatest limitation of this project will reveal itself: that of latency. The MIDI protocol is notorious for being slow in a local environment, (31.25 Kbits/sec) [MIDI Manufacturers Association] and the Internet is bound to heighten this problem, creating a substandard working environment if latency exceeds 50 ms [Chew and Sawchuck]. To the beginner, however, latency would not play such a major deterrent, because the novice learns through slow, repeated examples. As the need to learn closer to real-time approaches, that is, as a student becomes more proficient, latency on the order of about 60 ms could pose a surmountable annoyance. Although Young and Fujinaga have shown that a classical composition comprised of over 40,000 MIDI events produces "satisfactory" results, fluctuating Internet conditions will present varying latency-related obstacles along the development route [Young and Fujinaga]. In addition, it is common knowledge in the world of computer music that any virtuoso could push the limits of MIDI, causing events to back up, or bring about the "smearing" of MIDI messages, which is a phenomenon within the protocol leading to sporadic behavior. Internet distances would heighten the aforementioned [Foss and Mosala].

6 Conclusion

From the initial idea of this project, I was aware that writing a software project of this type would take a considerable amount of time. The summer months of 2005 allowed me to establish a simple, yet strong, foundation for the work ahead. And although substantial work and thought have gone into the present state of the software, which has yet to serve its purpose, it will be months, perhaps even a year or more, before a fully working program is in place.

Updates will be posted at <http://roy.vanegas.org/research>.

7 Future Directions

Platforms other than Mac OS X will be employed in the initial and future beta tests, starting with the Windows operating system. In addition, other versions of the Java compiler, ranging from version 1.2 to 1.5, also known as 5.0, will be used to test for backward compatibility.

Over the long-term, perhaps twelve to twenty-four months, the software has the potential of providing a high level of detail in terms of data. For example, the GUI may provide the instructor with precise information regarding performances by analyzing the pitchbend¹ of each executed MIDI note. The instructor will then be able to use this information to precisely determine the accuracy of the student's fingering.

For a more accurate depiction of string playing, the software program could analyze a mix of the audio streaming in from the electro-magnetic pickups and the data streaming in from the MIDI pickups. Combining the information from these two streams would help "paint" a better picture of the student's performance, yielding an even stronger depiction of fingering. For example, string buzz, which is caused by insufficient pressure on the strings, would be captured by the MIDI pickups and reported as a weakly-struck note. But, if an audio signal is also provided in tandem (only within the local environment), then the program could notify both the student and instructor about the presence of buzzing, more accurately analyzing performance. In other words, the program would loosely be reporting finger angle on the fretboard.

Future developments could also include an editing environment in which both the student and instructor can alter MIDI data freely.

The results of this project may extend to all stringed instruments, which can easily be laced with devices that "pickup" the elliptical pattern of a vibrating string, transforming those vibrations into MIDI messages. In fact, since microphones are used to trigger MIDI messages in some percussive instruments, using a pitch-to-MIDI converter, any musical instrument capable of triggering MIDI data could potentially take advantage of the results of this research project.

¹Pitchbend applies to instruments like the guitar or electronic synthesizer in which notes may be raised or lowered in terms of cents or fractions. A piano is unable to execute pitchbend messages.

8 Acknowledgements

Many thanks to Dr John Hartman for advising, Dr Kendra Gaines for proof-reading, Donna Treloar, Dean Maria Vélez, Dr Louis Ray, and C Maxene Summey for support.

References

- [Chew and Sawchuck] Chew, Elaine, and Alexander Sawchuk. *Distributed Immersive Performance*. 27 Jun. 2005 <<http://www-rcf.usc.edu/~echew/papers/NASM2004/DIP-2004NASM-proceedings.pdf>>
- [Cooperstock and Spackman] Cooperstock, Jeremy R., and Stephen P. Spackman. "The Recording Studio that Spanned a Continent." *First International Conference on WEB Delivering of Music*. 161-67. California: IEEE Computer Society, 2001.
- [Faber, Orlarey, and Letz] Fober, Dominique, and Yann Orlarey, Stephane Letz. "Real Time Musical Events Streaming over Internet." *First International Conference on WEB Delivering of Music*. 147-54. California: IEEE Computer Society, 2001.
- [Foss and Mosala] Foss, R., and Thabo Mosala. "Routing MIDI Messages Over Ethernet." *Journal of the Audio Engineering Society* 44.5 (1996): 406-15.
- [Freed, Momeni, and Wright] Freed, Adrian, Ali Momeni, and Matthew Wright. "OpenSound Control: State of the Art 2003." *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03), Montreal, Canada*.
- [Hajdu, Georg] Hajdu, Georg. "Quintet.net: An Environment for Composing and Performing Music on the Internet." *Leonardo Music Journal* 38.1 (2005): 23-30.
- [MIDI Manufacturers Association] MIDI Manufacturers Association. *The Complete MIDI 1.0 Detailed Specification*. Los Angeles: The MIDI Manufacturers Association, 2001.
- [Stevens] Stevens, W. Richard. *UNIX Network Programming*. New Jersey: Prentice-Hall, 1998.
- [Young and Fujinaga] Young, John P., and Ichiro Fujinaga. "Piano Master Classes Via The Internet." <<http://www.netmuse.org/jpy-netmidi99.pdf>>