The Global Metronome: Absolute Tempo Sync For Networked Musical Performance

Reid Oda
Department of Computer Science
Princeton University
35 Olden Street 08544
Princeton, New Jersey, USA
roda@cs.princeton.edu

Rebecca Fiebrink
Department of Computing
Goldsmiths, University of London
London, SE14 6NW
United Kingdom
r.fiebrink@gold.ac.uk

ABSTRACT

At a time in the near future, many computers (including devices such as smart-phones) will have system clocks that are synchronized to a high degree (less than 1 ms of error). This will enable us to coordinate events across unconnected devices with a degree of accuracy that was previously impossible. In particular, high clock synchronization means that we can use these clocks to synchronize tempo between humans or sequencers with little-to-no communication between the devices. To facilitate this low-overhead tempo synchronization, we propose the Global Metronome, which is a simple, computationally cheap method to obtain absolute tempo synchronization. We present experimental results demonstrating the effectiveness of using the Global Metronome and compare the performance to MIDI clock sync, a common synchronization method. Finally, we present an open source implementation of a Global Metronome server using a GPS-connected Raspberry Pi that can be built for under $100.

Author Keywords

Tempo Synchronization, Networked Music, Clock Synchronization, GPS, Global Metronome

ACM Classification

H.5.5 [Information Interfaces and Presentation] Sound and Music Computing—Systems, H.5.3 [Information Interfaces and Presentation] Group and Organization Interfaces

1. INTRODUCTION

The Internet has dramatically expanded the reach of modern humans, allowing us to speak face-to-face with people who are far away, access remote documents, and purchase items from across globe. However, music performance using the Internet has yet to become commonplace. A main reason for this is because tempo synchronization via the Internet is difficult, due to the latency inherent in networked communication. This delay causes *tempo drag* (i.e., musicians slow down in order to stay in time with one another), which makes rhythmic synchronization difficult, and at high latencies, impossible [2, 4, 5, 6]. In this work we explore a straightforward solution to synchronizing tempos over the Internet. Using this technique, players follow a local "metronome" that is synchronized to every other metronome, thereby gaining rock-solid timing, despite audio delays between performers. However, synchronizing metronomes requires synchronizing computer system clocks, which via this Internet is non-trivial. Asymmetries in networked communication, which are common, can cause audible errors when using modern synchronization algorithms.

In this work we propose a twofold solution to tempo synchronization. One is the concept of the Global Metronome, an absolute tempo synchronization method. All Global Metronome implementations beat in time with one another, automatically, with no communication between them. This is possible because they use the same (simple) method for computing the time of the next beat, and because their clocks are synchronized. Part two of our solution is to use GPS receivers to synchronize system clocks. This could also be implemented using LTE (4G) cellular phone technology, making mobile devices prime candidates for using the Global Metronome. Synchronizing via GPS or LTE circumvents the error inherent in Internet-based time synchronization methods. Any device with low-latency access to an accurate clock source can use the Global Metronome approach. Therefore, to provide this access to anyone who wants it, we present the PIGMI, a (Raspberry) Pi Global Metronome Implementation. A PIGMI can grant any local area network (LAN) access to a highly accurate clock synchronization source.

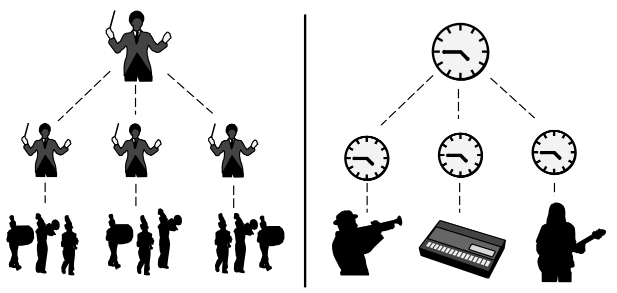

Figure 1: The Global Metronome approach is similar to using multiple conductors in a large marching band. Sub-conductors follow a head conductor to help all parts of the band stay synchronized. This approach relies on highly synchronized system clocks, which will become more common in the future. The "head conductor" is a virtual metronome that is imagined to have been ticking at tempo since the Unix epoch.

NIME'16, July 11-15, 2016, Griffith University, Brisbane, Australia.
.

2. BACKGROUND

Much of popular music is performed with rhythmic synchronization between musicians. We would like to perform this style of music via the Internet but it is difficult because of network latency. Network packets, containing encoded audio, take time to travel from sender to receiver causing a delay between when a note is played and when it is heard. The listener instinctively waits in order to play in time with the music they hear, causing further delay. This, in turn, causes a cascade effect (called *tempo drag*) where players on each side of a connection wait longer and longer [2, 4, 5, 6]. If there is enough latency, timing falls apart.

Audio latency occurs in the physical world, as well. Sound travels roughly one foot per ms, so a large ensemble (e.g. a marching band) has to deal with synchronization in spite of audio latency. A common solution is to use a central timing source, the conductor. The conductor 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 1. This is the approach that the Global Metronome uses. Multiple subconductors (Global Metronome implementations) synchronize to a virtual "master" tempo, and players synchronize to the metronomes.

The Global Metronome takes advantage of highly synchronized system clocks, which has been historically difficult to achieve. This is because (1) clocks in electronic devices are inaccurate and (2) high accuracy clock synchronization over networks is difficult. All clocks advance at different rates [10]. 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. In order to keep clocks in better time we can use network clock synchronization algorithms to estimate drift, and adjust the time of clocks. The most popular algorithm is the Network Time Protocol (NTP) [8]. However, NTP is accurate on the order of tens or hundreds of milliseconds. This is not accurate enough for music timekeeping, which requires millisecond (or ideally sub-ms) accuracy. 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. Standard (unmodified) NTP assumes that the transit time to and from a timeserver is symmetric, but in reality it is not. 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 [12]. Neither of these technologies suffer from the route asymmetry problem. GPS sends a time signal that has a theoretical accuracy of 14 nanoseconds, 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 microseconds. 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 anecdotally, towers are trending towards more accurate time synchronization.

There already exist a number of effective synchronization methods such as MIDI time clock [9], and new offerings such as LANdini [11] and Ableton Link [1]. While these methods have been designed for local synchronization, the Global Metronome can complement these methods by connecting ensembles across long distances.

The contributions of this paper are:

- the *concept* of the Global Metronome, which includes a simple and cheap way to compute the phase of the Metronome for any time t
- *experimental results* which deomonstrate the magnitude of error for various applications of the Global Metronome
- the *PIGMI* (Pi Global Metronome Implementation) an open source implementation of a Global Metronome, using the Raspberry Pi, which can be built for less than $100

3. THE GLOBAL METRONOME

We can think of the relationship between a metronome and time as a rotating phasor with phase $0 \leq \phi < 2\pi$, as shown in Figure 2.

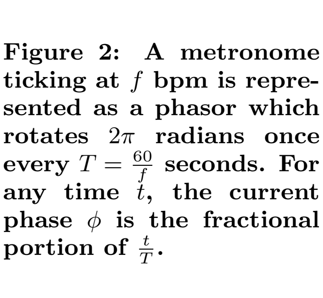

Figure 2: A metronome ticking at f bpm is represented as a phasor which rotates 2π radians once every $T = \frac{60}{f}$ seconds. For any time t, the current phase ϕ is the fractional portion of $\frac{t}{T}$.

For a given tempo f (in beats per minute), the phasor makes f rotations every minute. When two metronomes have $\phi_1 \equiv \phi_2$ 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 ϕ so that we get tempo synchronization for free.

Computing the current phase of the Global Metronome is straightforward and computationally cheap. 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.

$$T = \frac{60}{f} \tag{1}$$

$$\phi_{global} = 2\pi\Big(\frac{t}{T} - \Big\lfloor\frac{t}{T}\Big\rfloor\Big) \tag{2}$$

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

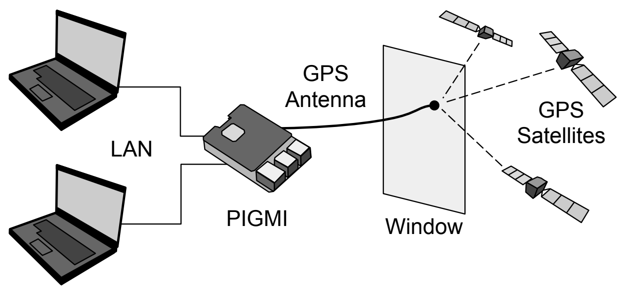

Figure 3: The Pi Global Metronome Implementation (PIGMI) is inexpensive and portable, and greatly simplifies the process of synchronizing to the Global Metronome. It requires a view of the sky in order to sense GPS satellites. To use it, place its antenna in a window, and connect the PIGMI to a network.

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 any 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

$$\begin{aligned}\phi_{local} &= metro(t, f, \theta) && (3)\\ &= \left[2\pi\left(\frac{t}{T} - \left\lfloor\frac{t}{T}\right\rfloor\right) + \theta\right] \mod 2\pi && (4)\end{aligned}$$

where T is the period as computed in Equation 1.

For many types of music, tempo changes are an important part of performance. However, tempo changes in the presence of latency can be challenging. Because it is an *absolute* metronome, the Global Metronome takes the ambiguity out of tempo changes. As tempos change, a desired change in the current tempo f or offset θ can be communicated among players, or the changes can be pre-planned. As long as all metronomes used by performers have the same f and θ they will be in sync with one another. How these changes are implemented is up to the developers of their system.

3.1 PIGMI: The $99 Global Metronome

Using the Global Metronome approach requires access to a local high-accuracy time source. The Raspberry Pi Global Metronome Implementation (PIGMI) is a tiny, inexpensive server 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 [1]. Included are hardware recommendations and a Raspbian Linux image with the software preinstalled.

To use a PIGMI, place it near a window that has a view of the sky and (if desired) connect it to a LAN, as shown in Figure 3. The PIGMI currently uses LANdini [11] as it's primary method to transmit clock synchronization information 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 that use this method. Each location is automatically in time with the others.

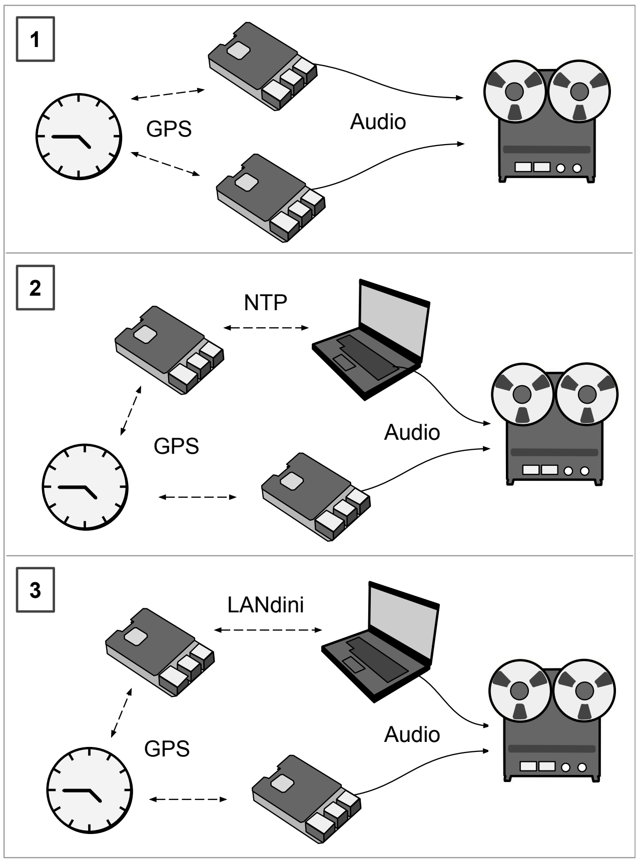

Figure 4: We tested 3 configurations of PIGMIs along with two "control" configurations (shown in Figure 5) in order to learn the magnitude of synchronization error. In each of the displayed configurations, we recorded 30 minutes of audio ticks at $f = 120$ beats per minute. Then, we analyzed the per-tick timing differences (offsets) of one recording compared to the other.

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 audio, 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 promote it to master timeserver.

4. EVALUATION

In order to gauge the accuracy of the Global Metronome given current technology, we conducted a series of experiments. GPS research shows that as long as a receiver can sense three GPS satellites, it will be synchronized to a high degree [7]. However, reading the sync signal and then conveying the synchronization information to another computer incurs error. Our goal is to gain an intuition into the magnitude of error that different configurations of PIGMIs incur.

We tested three different useful PIGMI configurations,

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

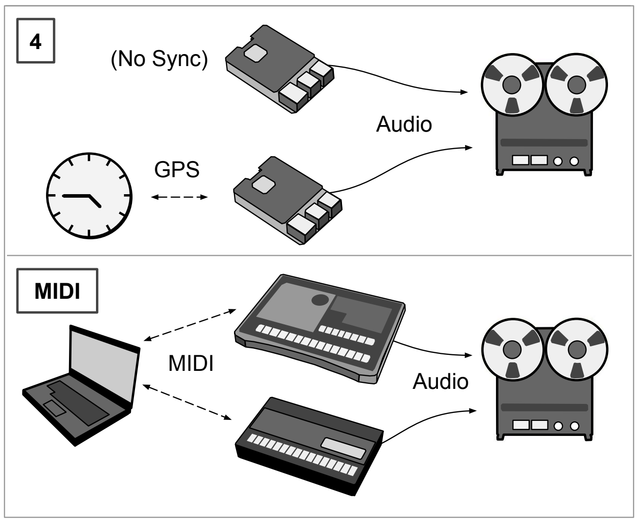

Figure 5: These configurations serve as informal synchronization benchmarks. Configuration 4 tests the scenario when one PIGMI is not synchronized to the Global Metronome, while "MIDI" tests two commercial drum machines, connected via a physical MIDI cable. As before, we recorded 30 minutes of audio ticks at 120 beats per minute. Then, we analyzed the offsets of these ticks.

shown in Figure 4, 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. For comparison, we also recorded 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 5). In the descriptions below, PIGMI-1 an PIGMI-2 are the names of specific PIGMI systems that are re-used throughout the experiment. Note that GPS receivers that are physically far apart are expected to have the same synchronization accuracy as those that are right next to one another.

- 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 via GPS. PIGMI-1 generates ticks directly via GPIO pin, and 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. This configuration is meant to test the scenario where systems connect via NTP to a low-latency, accurate timeserver (in this case, a PIGMI).

- Configuration 3 (GPS and LANdini): Both PIGMIs are synchronized via GPS. PIGMI-1 generates ticks directly, and the PC host connects via LANdini to PIGMI-2. In this case the host system clock is not synchronized to PIGMI-2, and the Global Metronome tick times are computed by referencing LANdini's representation of network time. 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 via 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 a physical MIDI cable, synchronized with MIDI clock to a host running Ableton Live. This configuration is to give intuition into the amount of jitter that exists in commercial MIDI devices.

Figure 6 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, and of the 5 configurations, the "No Sync" scenario is the one only whose regression has a non-zero slope of -0.006 ms/min with $p < 0.0001$. All other configurations had $p > 0.33$.

Figure 7 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 σ that is five times that of the GPS and NTP synchronized devices. While this offset is relatively large compared to the PIGMIs, they still sound synchronized (see demonstration video), illustrating the fact that the PIGMI tempo synchronization is exceptionally accurate.

5. DISCUSSION

Our data shows that GPS, NTP (to a low latency accurate time source), and LANDdini are all extremely accurate time synchronization methods, 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 [7].)

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. In these genres tempos rarely change, and the Global Metronome (along with an audio streaming technology such as JackTrip [3]) 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.

6. CONCLUSIONS

We have presented 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 experimental 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 future we plan to explore the tightly synchronized, physically distributed musical performances that the Global Metronome enables, as well as human interactions that result.

7. ACKNOWLEDGMENTS

We would like to thank Ableton Inc for giving us early access to the Link repository, thus allowing us to glimpse into the future of tempo synchronizing. We would also like the thank KatieAnna Wolf and Zeyu Jin for their insightful input. This work is supported in part by the Princeton University Project X grant.

8. REFERENCES

[1] Ableton link. `https://www.ableton.com/en/link/`. Accessed: 2016-01-01.

[2] C. Bartlette and M. Bocko. Effect of Network Latency on Interactive Musical Performance. *Music Perception*, 24, 2006.

[3] J. Cáceres and C. Chafe. JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research*, 2010.

[4] C. Chafe, J. Cáceres, and M. Gurevich. Effect of temporal separation on synchronization in rhythmic performance. *Perception*, 39, 2010.

[5] C. Chafe, M. Gurevich, G. Leslie, and S. Tyan. Effect of time delay on ensemble accuracy. In *Proc. the International Symposium on Musical Acoustics*, 2004.

[6] 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.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36, 2002.

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

[9] R. A. Moog. Midi: Musical instrument digital interface. *Journal of the Audio Engineering Society*, 1986.

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

[11] J. Narveson and D. Trueman. Landini: a networking utility for wireless lan-based laptop ensembles. In *Proc. SMC Sound, Music and Computing Conference*, 2013.

[12] S. Sesia, I. Toufik, and M. Baker. *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

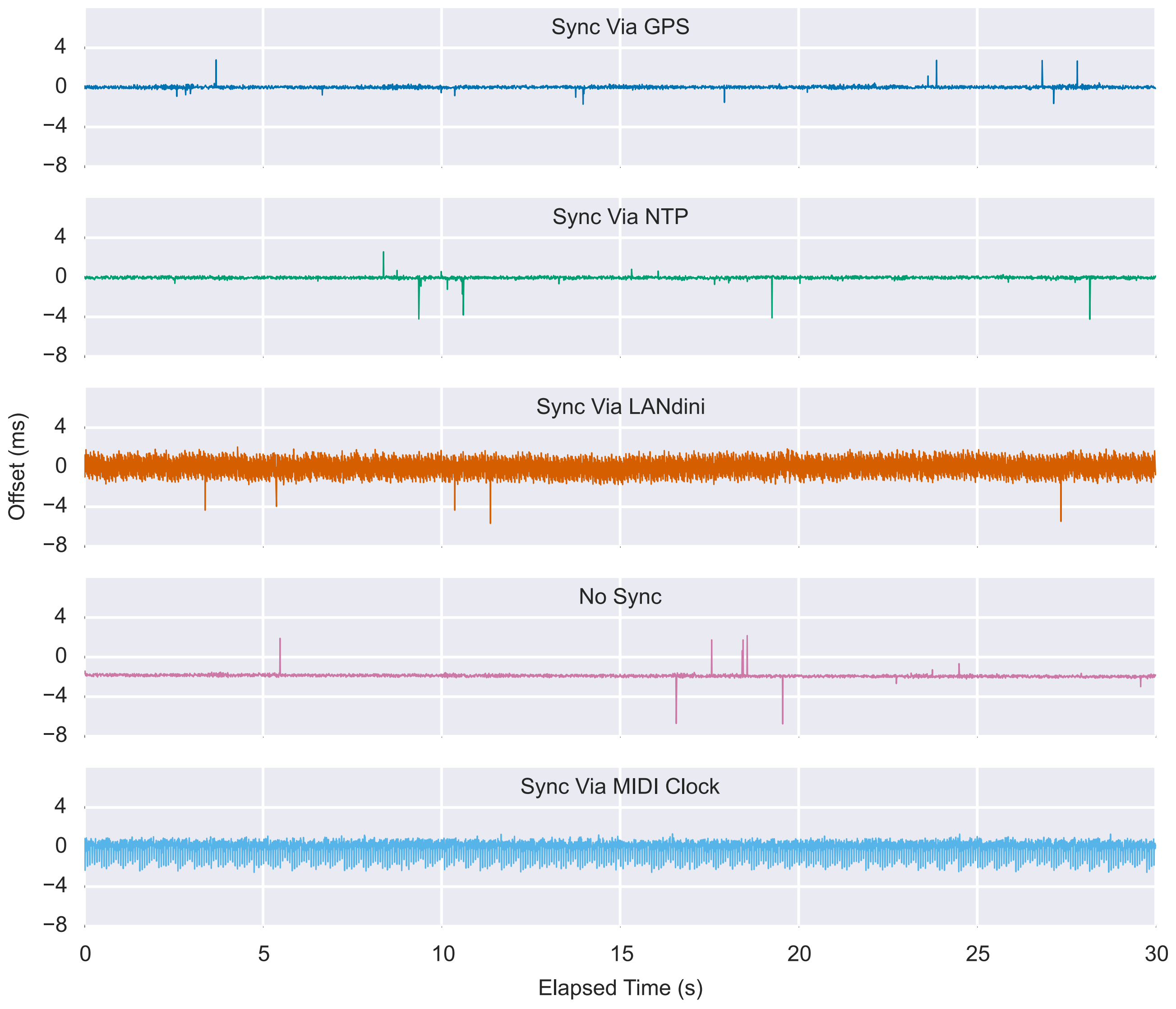

Figure 6: The offsets for each experiment configuration for $f = 120$, $\phi = 0$ over a period of 30 minutes. In the cases of NTP and LANdini synchronization, the devices connect to a PIGMI on a network, which is itself synchronized via GPS (see Figures 4 and 5 for a detailed illustration). Note that 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.

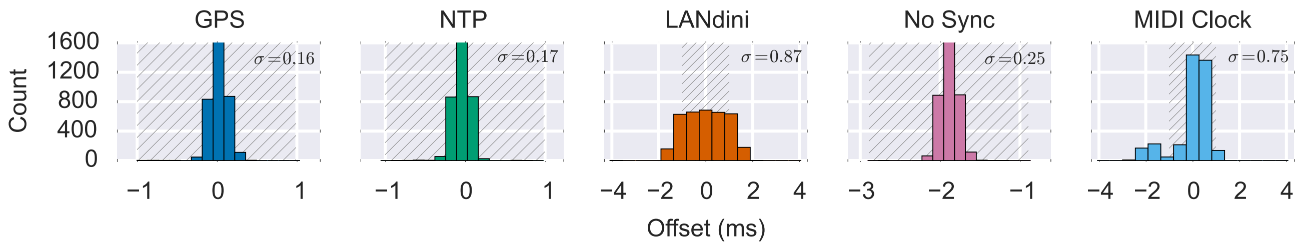

Figure 7: Histograms of the offset data, along with σ, the standard deviation for each. Note that the x-axis range is not the same for each plot. The hatch-shaded area indicates a 2 ms range centered on each mean. GPS offers the most accurate synchronization, followed closely by NTP, while MIDI clock and LANdini exhibit similar performance. "No Sync" has low standard deviation, but is offset by -2 ms.