http://cnmat.berkeley.edu/publications/simple-synchronization-open-sound-control

Simple Synchronisation for Open Sound Control

Sebastian Madgwick
x-io Technologies
Bristol, UK
sebmadgwick@x-io.co.uk

Thomas Mitchell
University of the West of England
Bristol, UK
tom.mitchel@uwe.ac.uk

Carlos Barreto
x-io Technologies
Bristol UK
cbarreto61@unisalle.edu.co

Adrian Freed
CNMAT, UC Berkeley
CA, USA
adrian@cnmat.berkeley.edu

ABSTRACT

Clock synchronisation is a mature and important aspect of distributed computing systems. Despite the importance of accurate timing in music, there are relatively few widely applicable synchronisation solutions available to computer music practitioners. In this paper we present a simple OSC-based synchronisation method for wired and wireless applications, which is designed to be easy to apply and is shown to offer accuracy appropriate for fine-grained music applications. The proposed solution relies on a single master sending a synchronisation message to all slaves. Empirical studies with a heterogeneous network of 17 Wi-Fi slaves and 5 Ethernet slaves demonstrate that each homogeneous group is able to achieve a relative synchronisation accuracy of 166 us and 100 us respectively, offset from the master time by their respective network latencies. An acoustic localisation system is implemented to demonstrate an application that requires both accurate synchronisation and benefits from wireless connectivity. The system is shown to precisely locate a sound source with a standard deviation of 1.8 mm.

1. INTRODUCTION

In networks of distributed computational devices, individual network nodes are equipped with a local clock from which events may be accurately scheduled or timestamped. In practice this clock is derived from an on-board crystal or oscillator circuit, which, due to imperfections in the timing hardware, will tend drift with respect to the clocks on other network nodes [1]. Scenarios demanding precise temporal coordination have stimulated the development of many synchronisation procedures designed to establish an accurate, network-wide notion of time [2, 3]. Given the importance of timing and rhythm in music [4, 5], and the continually emerging applications of networked computational devices for musical purposes [6, 7, 8, 9, 10], there is a growing need for a widely applicable synchronisation solution for computer music applications.

This paper presents a simple Open Sound Control (OSC) synchronisation procedure for distributed systems, which has been designed to satisfy the following objectives:

- simple to implement on a wide range of platforms (heterogeneous network nodes)
- 'transport independent' operating entirely through the exchange of OSC messages
- meets the fine-grained synchronisation requirements of computer music applications (i.e. sub-ms error [11])

The paper begins with an introduction to network synchronisation with particular focus on computer music systems. The proposed synchronisation method is then presented, followed by a detailed empirical analysis and discussion of its performance. A practical application example of the synchronisation precision is then provided in the form of a wireless indoor acoustic localisation system. The paper closes with concluding remarks and proposals for future developments.

2. BACKGROUND

Clock synchronisation is a long and established component of distributed computing systems, underpinning the technology that enables GPS, mobile telecommunications, wireless data communications, networked file system integrity, sensor fusion, localisation, object and motion tracking [1, 12]. The Network Time Protocol (NTP) [13] has for decades kept internet clocks running to within a few milliseconds of Universal Coordinated Time and now forms one of a plethora of available synchronisation methods that have since been proposed. Usually, these methods operate by propagating a global or master time through the network from which slaves modify their local clocks to maintain synchronisation. The mechanisms of each synchronisation method differ based

upon the the relevant network topology and the application requirements in terms of accuracy, precision and energy. Comprehensive overviews can be found in the literature, see for example [14] and [15] for a more recent comparison of synchronisation methods for wireless sensor networks.

Synchronisation has also played an important role in distributed music systems to ensure audio and visual media alignment. As technology has pervaded the sonic and visual arts, a number of established synchronisation protocols and interfaces have been proposed and adopted that enable distributed devices to maintain a common time frame. For example, MIDI clock, MIDI timecode and SMPTE have provided reliable (although relatively imprecise) device synchronisation for professional media applications [16, 17]. Research into novel musical interfaces and laptop/mobile phone performances are making increasing use of interconnected networks of wired and wireless devices to sense and capture expressive control input and to schedule precisely timed output. For example, The Princeton Laptop Orchestra [18], the Carnegie Mellon Laptop Orchestra [19] and the LOLC laptop ensemble [20] implement their own synchronisation algorithms to ensure timely coordination amongst players. The former alludes to an OSC synchronisation procedure with synchronisation precision estimated in the order of 30-40 ms, while the latter two incorporate messaging processes based upon Christian's algorithm [21]. A third live coding synchronisation process developed by Ogborn [22] for the Cybernetic Orchestra encapsulates the complexities of the synchronisation process within a dedicated application called EspGrid, which exposes global beat resolution event messages to clients running OSC compatible music software. While this abstraction is elegant for synchronising to salient musical events, it is not designed to provide random access to the real-time clock, precluding high-precision synchronisation applications such as localisation and sensor fusion. In their 2014 live coding report, Blackwell et al [23] draw attention to a growing abundance of bespoke synchronisation methods noting that cooperation is currently hampered by a lack of agreement on a single protocol.

This work is concerned with cases in which many established synchronisation protocols may not be applicable because they depend upon specific physical interfaces or other specific OSI model technologies such as TCP/IP. The proposed synchronisation method is designed to operate in situations where nodes are able to communicate by OSC and have real-time access to an onboard physical clock. The OSC protocol has been chosen as it has now become a key technology within the computer music community [24]. The time tag [25] argument type was introduced to the OSC protocol as an enabling feature for device synchronisation. Despite several subsequent calls for an OSC synchronisation method [26, 27, 28], a standardised approach is still to be established.

3. PROPOSED SOLUTION

3.1 Algorithm derivation

The proposed synchronisation solution relies on a single synchronisation *master* periodically broadcasting a measurement of global time to all *slaves* on a communication network; a method resembling the 'simple' synchronisation solution proposed by Dannenberg [29].

The synchronisation messages sent by the master provide each slave with a precise observation of global time. However, this observation will incorporate a varying error due to the system latencies; the most significant being communication latency, and task scheduling within the slave. Figure 1 shows the error in the observed global clock for a single slave. This plot was obtained for a Wi-Fi node using the experimental setup described in Section 4.1 and a fixed synchronisation message rate of 5 messages per second.

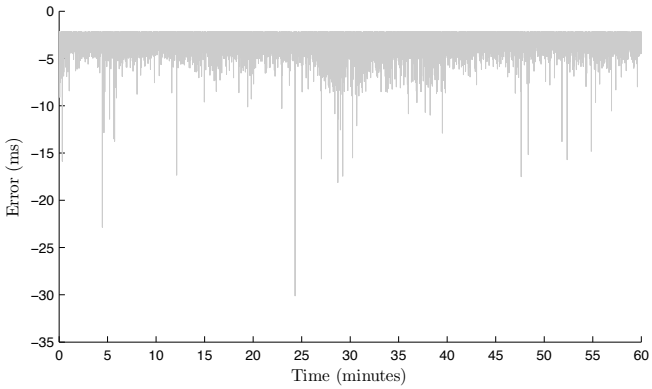

Figure 1. Error in the observed global clock for a single slave.

The visualisation of the error in the observed global clock shown in Figure 1 clearly indicates a fixed minimum error. Although errors may be as high as 30 ms, a significant number of observations have a precise error of 2.2 ms. This represents the minimum system latency, and corroborates Wi-Fi round-trip latency measurements made with similar devices in [30, 31].

The distribution of the error will vary between hardware and software platforms, and communication interfaces. However, for a network of homogeneous slaves, the minimum system latency will be equal for all slaves. A slave may therefore achieve relative synchronisation with other homogeneous types if it is able to identify synchronisation messages that are likely to represent the minimum system latency. The proposed solution achieves this through the following three heuristics:

1. The slave incorporates a fixed *slave clock drift* to ensure that the slave clock is slower than the global clock.
2. If the observed global clock is ahead of the current slave clock then update the slave clock to equal the observed global clock.

3. If the observed global clock differs from slave clock by more than a specified *threshold* then update the slave clock to equal the observed master clock.

The *slave clock drift* must be greater than the expected worst-case relative difference in speed between the global clock and slave physical clock. For example, if both the global clock and slave clock are derived from a ± 20 ppm crystal then drift should be at least -40 us per second. The *threshold* value should be greater than the expected maximum communication latency so that the difference between the slave clock and global clock only ever exceeds the threshold upon initialisation of the system.

Figure 1 demonstrates the slave clock maintaining synchronisation with a fixed offset from the global clock using the above heuristics. This plot was obtained for a Wi-Fi node using the experimental setup described in Section 4.1 and a fixed synchronisation message rate of 5 messages per second and *slave clock drift* of -10 us per second.

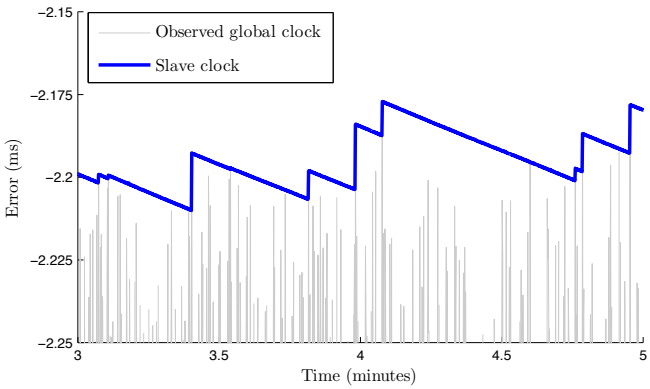

Figure 2. Slave clock corrections based on observations of master clock

An important characteristic of the slave synchronisation algorithm is that updates to the slave clock (post-initialisation) will only ever result in steps forward in time, an important condition that must be satisfied to ensure partial ordering of events [32, 15].

3.2 Algorithm implementation

The synchronisation message sent by the master is a 20-byte OSC message containing a single OSC time tag argument with the address pattern, "`/sync`". An OSC time tag is a 64-bit, unsigned fixed-point representation of time elapsed since January 1, 1900 with a resolution of 233 ps. This is also the representation used by NTP.

4. EMPIRICAL STUDIES

Empirical studies were conducted to demonstrate: the relationship between the synchronisation message rate and slave synchronisation error, the relationship between application throughput and slave synchronisation error, and the behaviour of a heterogeneous network of slaves.

4.1 Experimental setup

The experimental setup included 17 Wi-Fi slaves, 5 Ethernet slaves, and a single Ethernet master. The Wi-Fi slaves were x-OSCs [30] running customised firmware. The Ethernet master and slaves were Microchip Ethernet Start Kits. All devices communicated via a Netgear AC 1900 router. The Ethernet master had a direct connection to the router while the 5 Ethernet slaves were connected to the router via a Netgear GS308 Ethernet switch. The Wi-Fi slaves were connected to the router 2.4 GHz network. A laptop was connected to the router 5 GHz Wi-Fi network to log results without interfering with the 2.4 GHz traffic. This setup is illustrated in Figure 3.

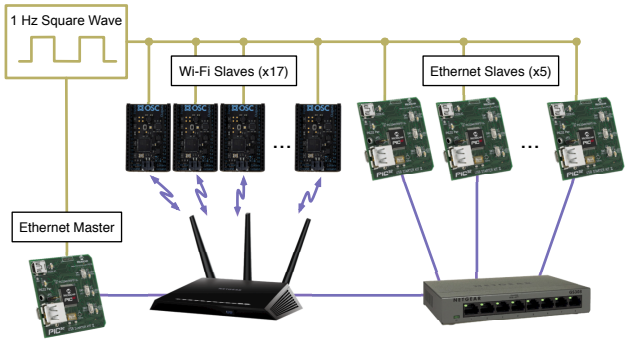

Figure 3. Experimental setup showing a network of 17 Wi-Fi and 5 Ethernet slaves with a single Ethernet master

A 1 Hz square wave signal was connected to a digital input on the master and each slave. The digital inputs were configured as external interrupts to send a time-stamped OSC message indicating its clock value on each edge of the square wave. The latency between the digital input edge and sampling of the clock is $<$1.5 us. This provides a means of sampling all clocks simultaneously so that the error of each slave clock relative to global time may be obtained.

The Wi-Fi slaves were measured to have a small variance in crystal tolerance and so used a slave clock drift value of -10 us per second. The Microchip Ethernet Start Kits used a slave clock drift value of -150 us per second

4.2 Results

4.2.1 Synchronisation message rate vs. synchronisation error

The box plots in Figure 4 shows the relationship between the synchronisation message rate and the synchronisation error of a single Wi-Fi slave. The synchronisation message rates range from 1 message every 10 seconds to 100 messages per second. Each box plot represents approximately 5300 samples over 45 minutes. The whiskers indicate 1.5$\times$ interquartile range.

The distributions in Figure 4 demonstrate the predicable relationship that higher synchronisation message rates yield a reduced variance in the synchronisation error and shift the mean towards to minimum system latency. Outliers

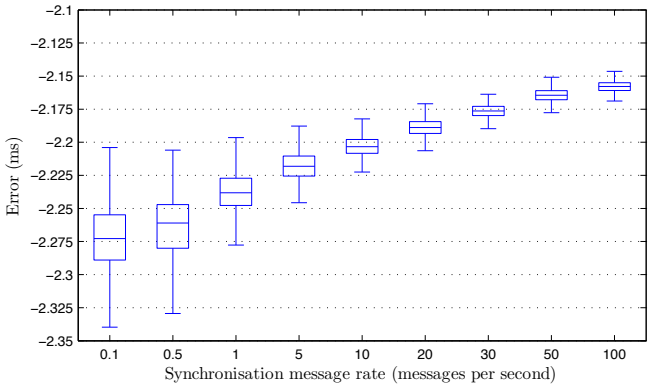

Figure 4. Relationship between the synchronisation message rate and the synchronisation error of a single Wi-Fi slave

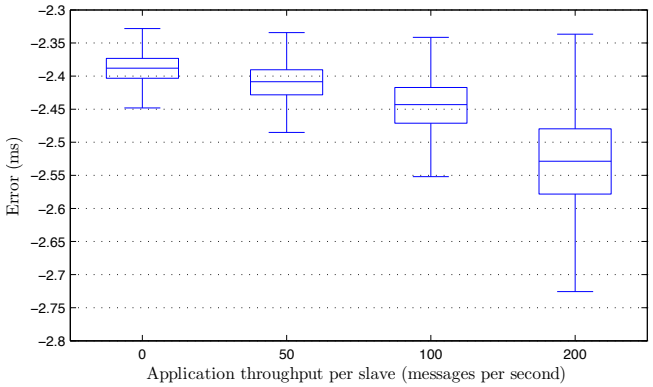

Figure 5. Relationship between application throughput and Wi-Fi slave synchronisation error

define the peak-to-peak variation and so indicate the potential worst-case synchronisation error. The peak-to-peak variation was measured as 149 us at 0.1 Hz, 50.6 us at 5 Hz, 26.5 us at 50 Hz and 26.2 us at 100 Hz.

The relationship between the synchronisation message rate and synchronisation error allows the designer to optimise for a given application. For high-bandwidth applications, a high message rate may achieve a synchronisation error (between homogeneous slaves) approaching *sample accurate* timing for CD quality audio (22.7 us sample period at 44.1 kHz). For low-bandwidth or low-power applications, message rates as low as once every 10 seconds may still achieve a low peak-to-peak variation of $<$150 us.

4.2.2 Application throughput vs. synchronisation error

Synchronisation messages place an additional demand on the bandwidth of a communication channel. The box plots in Figure 5 show the relationship between the application throughput and the synchronisation error of 17 Wi-Fi slaves. The application throughput was created by each slave sending an additional 0, 50, 100 or 200 UDP packets per second. Each packet contained a 104-byte OSC message containing 16 analogue input measurements represented as 32-bit floating point values. The synchronisation message rate was 5 messages per second. Each box plot represents approximately 30000 samples collected over 15 minutes.

The distributions in Figure 5 demonstrate that synchronisation error can be expected to increase for high application throughput. This is in part due to a reduction in the available bandwidth of the communication channel and in part due to the increased loading on the slaves' task scheduling. The peak-to-peak variation was measured as 117 us at 0 Hz, 159 us at 50 Hz, 218 us at 100 Hz and 376 us at 200 Hz.

4.2.3 Heterogeneous network of slaves

The proposed synchronisation solution is limited to only achieving accurate synchronisation of slaves of homogeneous types; each group of homogeneous types will be subject to different error offset approximating the minimum system latency for the type. The box plots in Figure 6 demonstrate this behaviour for two groups of homogeneous types: the 17 Wi-Fi slaves and the 5 Ethernet slaves. Each distribution represents approximately 53000 samples collected over 9 hours with a synchronisation message rate of 5 messages per second.

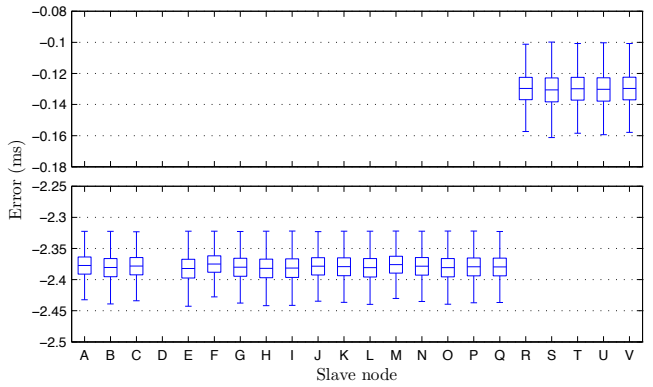

Figure 6. Synchronisation error of 22 heterogeneous slaves. Slaves A to Q are Wi-Fi nodes, slaves R to V are Ethernet nodes (slave D malfunctioned during the experiment)

A faster and more reliable connection enables the Ethernet slaves to achieve a lower variance with a mean synchronisation error of -125 us relative to the global clock. Although the 17 Wi-Fi nodes maintain relative synchronisation, they share a common mean synchronisation error of -2.37 ms relative to the global clock. The worst-case synchronisation error between any Wi-Fi slaves was 166 us, and 100 us for any two Ethernet slaves over the 9 hour period.

5. EXAMPLE APPLICATION: ACOUSTIC LOCALISATION

An acoustic localisation system was implemented to demonstrate an application that requires both accurate synchronisation and benefits from wireless connectivity. The system comprises eight wireless sensing nodes distributed

around the edge of a workspace. The location of a sound source within the workspace was calculated from the relative time-of-arrival of the sound at each sensing node using multilateration [33]. The accuracy of the location is therefore dependant of the accuracy of the synchronisation. Wireless connectivity simplifies deployment of the sensing nodes over a large workspace.

5.1 Wireless sensing node

Each wireless sensing node comprised an x-OSC, audio trigger circuit and battery. The x-OSC digital inputs provide a timestamped OSC message for each rising or falling edge. The audio trigger circuit would provide a digital pulse when a microphone amplitude exceeds a threshold so that the time of arrival of a sound would be obtained. The threshold was set by a x-OSC PWM output so that it could be adjusted wirelessly, via OSC messages, once the system had been deployed.

The audio trigger circuit, shown in Figure 7, was designed to minimise variations in latency between sensing nodes. The signal from an omni-directional electret microphone, multiplied by a gain of $\sim$1400, was compared with both a positive and negative threshold of equal magnitude to prevent sensitivity to the polarity of the wave form. The circuit does not incorporate any intentional low-pass filtering (e.g. an anti-aliasing filter) as resistor and capacitor value tolerances may risk a differing phase response between devices.

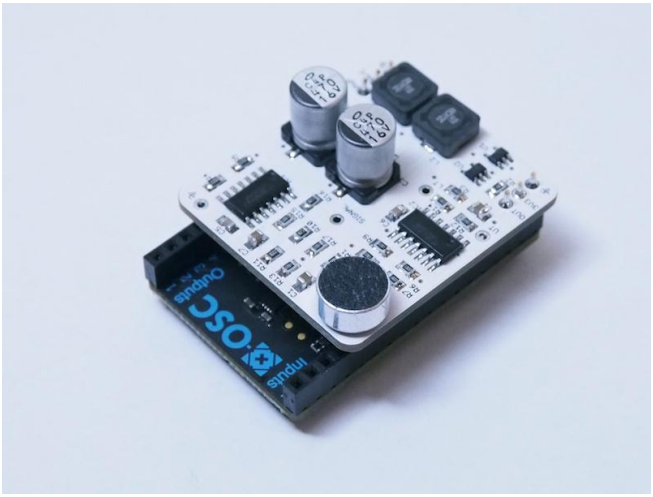

Figure 7. One of eight wireless senseor nodes with attached audio trigger circuitry

5.2 Deployment and results

The eight sensor nodes were equally spaced around the perimeter of a 6 m square. The synchronisation message rate was set to 50 messages per second to optimise for synchronisation accuracy. A loudspeaker was placed at 5 different locations within the square. The speaker would play a 15 ms noise burst 100 times at each location over 3 to 4 minutes. The microphone and loud speaker locations are illustrated in Figure 8. The locations of the speaker in this Figure were obtained using the acoustic localisation system.

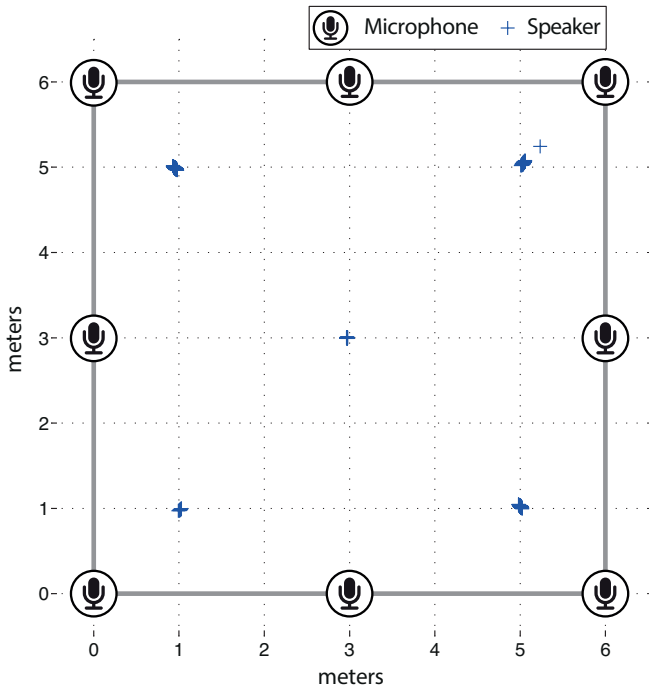

Figure 8. Eight sense nodes were equally spaced around the perimeter of a 6 m square to obtain measurements of 5 different speaker locations using multilateration

The standard deviation in the measured locations of the speaker was $<$8 mm. It was not possible to quantify the mean error of each measured location as the accuracy of the acoustic localisation measurement was greater than that of the measurement made when physically setting up the experiment. The precision was greatest when the speaker was located in the centre with a standard deviation of 1.8 mm, corresponding to a temporal error of 2.6 us. Figure 9 shows a detailed plot of the 100 measurements made at this location. These measurements suggest that the speaker was placed with 3 cm error in one dimension. This precision demonstrates the accuracy of the synchronisation.

6. CONCLUSIONS

In this paper we have presented a simple distributed computing synchronisation method for OSC. The method relies on a single master sending a synchronisation message to all slaves. The synchronisation algorithm has only three adjustable parameters: *slave clock drift*, *threshold*, and synchronisation message rate. The first two are straightforward to derive from the crystal tolerance and worst-case communication latency. The synchronisation message rate may be chosen to meet the requirements of a given application. For example, the empirical studies in Section 4 demonstrate a synchronisation precision of $<$150 us

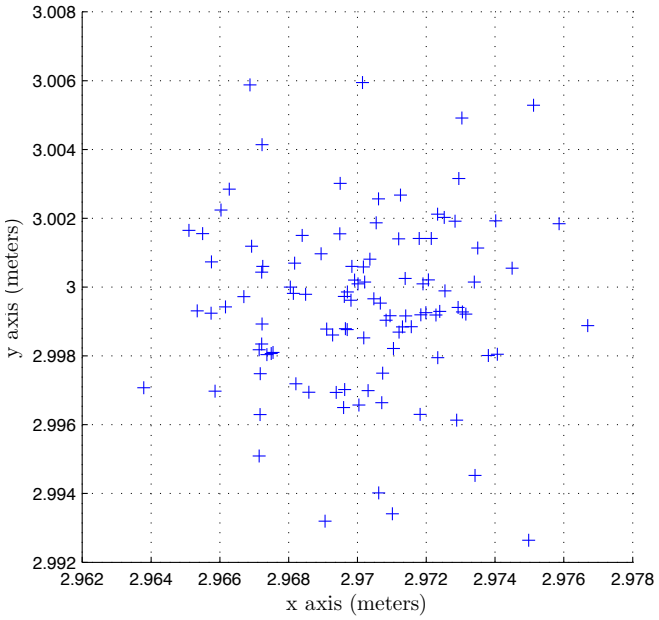

Figure 9. Detailed view of acoustic localisation measurements for the speaker located at the centre of the square

for a message rate of 0.1 Hz, and 26.5 us for a message rate of 50 Hz. Low message send rates are necessary for low-power and low-bandwidth applications. High message rates are appropriate for applications demanding greater precision, such as the acoustic localisation system demonstrated in Section 5.

The experimental work in this paper focused on Wi-Fi and Ethernet implementations. However, the proposed solution may be implemented on any communication network that supports OSC. Synchronisation accuracy will vary between different communication networks and types of slave as was shown in Section 4.2.3 for the heterogeneous network of Ethernet and Wi-Fi slaves, which showed a fixed offset between each homogeneous group. Many applications will use only homogeneous slave types and so can benefit from greater relative synchronisation accuracy as demonstrated by the 166 us error achieved for the 17 Wi-Fi nodes in Section 4.2.3.

Future work will explore methods of compensating for differing physical clock speeds and estimation of the fixed offset in synchronisation error from the global clock. Methods to address the susceptibility of the method to synchronisation loss if the clock master node fails are also under investigation.

7. REFERENCES

[1] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock Synchronization of Wireless Sensor Networks." *Signal Processing Magazine*, vol. 28.1, 2011.

[2] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214–226, 2006.

[3] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *Network, IEEE*, vol. 18, no. 4, pp. 45–50, 2004.

[4] N. Todd, "A Model of Expressive Timing in Tonal Music," *Music Perception*, vol. 3, no. 1, 1985.

[5] E. F. Clarke, "Rhythm and Timing in Music," in *The Psychology of Music*, 2nd ed., D. Deutch, Ed. London: Elsevier, 1999.

[6] N. Collins, A. McLean, J. Rohrhuber, and A. Ward, "Live coding in laptop performance," *Organised Sound*, vol. 8, pp. 321–330, 12 2003.

[7] C. Alexandraki and D. Akoumianakis, "Exploring new perspectives in network music performance," *Comput. Music Journal*, vol. 34, no. 2, 2010.

[8] R. A. Weinberg, Gil and K. Jennings, "The Beatbug Network: A Rhythmic System for Interdependent Group Collaboration," in *Proceedings of the 2002 conference on New interfaces for musical expression*, 2002.

[9] G. Weinberg, "Interconnected Musical Networks: Toward a Theoretical Framework," *Computer Music Journal*, 2005.

[10] A. Barbosa, "Displaced SoundScapes A Survey of Network Systems for Music and Sonic Art Creation," *Leonardo Music Journal*, vol. 13, 2003.

[11] A. Schmeder, A. Freed, and D. Wessel, "Best practices for open sound control," in *Proceedings of the Linux Audio Conference*, 2010.

[12] B. M. Sadler and A. Swami, "Synchronization in sensor networks: an overview," in *Military Communications Conference*, IEEE, Ed., 2006.

[13] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, 1991.

[14] S. Bregni, "A historical perspective on telecommunications network synchronization," *Communications Magazine (IEEE)*, vol. 36, no. 6, pp. 158–166, 1998.

[15] S. Rahamatkar, A. Agarwal, and N. Kumar, "Analysis and comparative study of clock synchronization schemes in wireless sensor networks," *Analysys*, vol. 2, no. 3, pp. 536–541, 2010.

[16] F. Rumsey, "Digital Audio Interfacing-A Brief Overview (Digital Audio Tutorial)," in *Proceedings of the Audio Engineering Society Conference: Images of Audio*, 1991.

[17] E. Brandt and R. Dannenberg., "Time in Distributed Real-Time Systems," in *Proceedings of the International Computer Music Conference*, 1999.

[18] D. Trueman, P. Cook, S. Smallwood, and G. Wang, "PLOrk: the Princeton laptop orchestra, year 1," in *Proceedings of the international computer music conference*, 2006.

[19] R. B. Dannenberg, S. Cavaco, E. Ang, I. Avramovic, B. Aygun, J. Baek, E. Barndollar, D. Duterte, J. Grafton, R. Hunter, C. Jackson, U. Kurokawa, D. Makuck, T. Mierzejewski, M. Rivera, D. Torres, and A. Yu., "The Carnegie Mellon Laptop Orchestra," in *Proceedings of the International Computer Music Conference*, 2007.

[20] J. Freeman and A. V. Troyer., "Collaborative Textual Improvisation in a Laptop Ensemble," *Computer Music Journal*, vol. 32, no. 2, 2011.

[21] F. Cristian, "Probabilistic clock synchronization," *Distributed Computing*, vol. 3, no. 3, 1989.

[22] D. Ogborn, "Live Coding in a Scalable, Participatory Laptop Orchestra," *Computer Music Journal*, vol. 38, no. 1, 2014.

[23] A. F. Blackwell, A. McLean, J. Noble, and J. Rohrhuber, "Collaboration and learning through live coding," *Dagstuhl Reports*, vol. 3, no. 9, 2014.

[24] A. Schmeder and A. Freed, "Implementation and Applications of Open Sound Control Timestamps." in *Proceedings of the international computer music conference*, 2008.

[25] M. Wright, A. Freed, and A. Momeni, "Opensound control: State of the art 2003," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2003.

[26] A. Freed, "Towards a More Effective OSC Time Tag Scheme." in *Proceedings of The Open Sound Control Conference*, 2004.

[27] M. Wright, "Open Sound Control: an enabling technology for musical networking," *Organised Sound*, vol. 10, no. 3, 2005.

[28] A. Freed and A. Schmeder, "Features and future of Open Sound Control version 1.1 for NIME," in *Proceedings of the Conference on New Interfaces for Musical Expression*, 2009.

[29] R. Dannenberg, "Clock Synchronization for Interactive Music Systems," in *Proceedings of The Open Sound Control Conference*, 2004.

[30] S. Madgwick and T. Mitchell, "x-OSC: A Versatile Wireless I/O Device For Creative/Music Applications," in *SMC Sound and Music Computing Conference*, 2013.

[31] T. Mitchell, S. Madgwick, S. Rankine, G. Hilton, A. Freed, and A. Nix, "Making the Most of Wi-Fi: Optimisations for Robust Wireless Live Music Performance." in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2014.

[32] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, 1978.

[33] B. T. Fang, "Simple Solutions for Hyperbolic and Related Position Fixes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 5, pp. 748–753, 1990.