All Together Now: A Synchronous Platform for Distributed Spatial Audio

Thomas Rushton(thomas.rushton@inria.fr)[1], **Romain Michon**[1], and **Tanguy Risset**[1]

[1] **Inria, INSA Lyon**, CITI, EA3720, 69621 Villeurbanne, France

ABSTRACT

State-of-the-art systems for spatial and immersive audio lie beyond the reach of many potential users due to the financial and technical investments that they demand. The risks associated with buying into a closed, centralised system of inflexible hardware and specialist software mean that, although areas of considerable research interest in recent years, significant barriers-to-entry exist with regard to immersive audio and virtual acoustics via practical application of sound field synthesis techniques. It is possible, however, to parallelise sound field synthesis algorithms and distribute them, subject to guarantees regarding the timing of distributed signal processors. Exploiting one of a number of low-cost embedded hardware platforms to have emerged over the past decade, and open-source software implementing the network Precision Time Protocol, we show that low-cost, networked audio devices can be synchronised to a degree sufficient to support time-sensitive audio applications.

1. INTRODUCTION

Whether operating on a budget of millions, or being an individual or institution of more modest means, the desire to establish a large-scale multichannel audio system is one inevitably confronted by difficult decisions, and one that is at the mercy of factors both technological and commercial. Beyond the obvious technical concern, that of *"will the system that I am designing serve the purpose I intend?"* lie less comfortable questions such as *"will it still work in a decade's time?" "what options do I have if I want to extend it later?"* and perhaps *"who really owns this?"*

Politics aside, the centralised nature of conventional systems for spatial and immersive audio renders them inflexible, with limited scalability and modularity, and costly, being dependent on high channel-count commercial audio interfaces and machines capable of performing demanding, real-time signal processing computations. In short, such systems have an accessibility problem — one that poses a significant barrier to research and innovation.

Other, less conventional approaches are possible though, and have, in recent years, and in various forms, been attempted. From networked GPUs [1] and microcontrollers [2] to DIY ambisonics domes [3] and frugal wave field synthesis arrays [4], efforts are underway to offer alternatives to the established order, bringing potential benefits, in terms of cost and scalability, with them. Work of this sort aims to democratise access to scholarly and creative practice in the sphere of spatial audio; commercial interests, too, could benefit from broader adoption of systems with improved accessibility.

In this paper, we present recent advancements to our work on distributed spatial audio, the latest steps on the road toward an accessible, scalable multichannel audio system with the strength to stand as a viable alternative to the commercial state-of-the-art. In Section 2 we discuss prior work in distributed and frugal spatial audio and outline the key challenges associated with audio in a distributed context; in Section 3 we present our approach to establishing an all-important *authoritative source of time* in that context; in Section 4 we present some preliminary results based on the method outlined in Section 3; we conclude, in Section 5, by describing some outstanding challenges and sharing our aims for future research.

2. BACKGROUND

The importance of sound in the immersive experience is well established [5]; if, however, the auditory aspect of virtual and extended reality remains comparatively neglected when considered alongside other sensory channels [6], this may be ascribed, in part, to the complexity and cost associated with immersive audio installations. Large-scale, multichannel spatial audio systems are typically very expensive, centralised installations, reliant on proprietary hardware and specialist software. Such systems scale poorly and demand significant computing power to serve DSP needs [2]. Particularly in the case where one wishes for numerous individuals to share in a participatory auditory experience, unencumbered by headsets, an immersive acoustic environment is desirable, and yet may lie out of reach, for practical purposes. The exclusivity of multichannel audio systems has a chilling effect on research and creativity.

In simple terms, to serve a large listening area with the greatest possible fidelity, sound field synthesis algorithms [7] such as wave field synthesis (WFS) and higher order ambisonics (HOA) call for large numbers of loudspeakers. To minimise spatial aliasing, WFS demands a dense array of secondary point sources [8]; to achieve the same end, ambisonics requires a number of speakers sufficient to reproduce spherical harmonics of higher order [9]. Conventional systems for meeting these needs range from

Platform	MCU/SoC	Memory	Ethernet	PTP	Unit Cost
Raspberry Pi CM5	Broadcom BCM2712	2 GB SDRAM	Yes	Yes	€ 89
Bela	Texas Instruments Sitara AM3358	512 MB SDRAM	Yes	Yes	€ 190
ESP32-LyraT	ESP32-WROVER-B	4 MB PSRAM	Yes	No	€ 21
Teensy 4.1	NXP iMXRT1062	1024 KB SRAM	Yes	Yes	€ 48
Daisy Seed	STM32H750IB	65 MB SDRAM	No	Yes	€ 25

Table 1: Selected embedded computing platforms listed with their corresponding microcontroller (MCU) or system on chip (SoC), availability of networking functionality, and approximate cost per unit (rounded to the nearest euro) as of March 2025.

square arrays of 64-channels backed up by a single Mac Pro and multichannel audio digital interface (MADI) hardware [10], to arrays of hundreds of speakers requiring clusters of powerful computers to provide DSP [11, 12]; [1] in either case, conservative cost estimates on the order of hundreds of euros per channel may be reached [2].

2.1 Frugal Approaches to Spatial Audio

Progress in embedded computing, devices of the *internet of things* variety, and initiatives in the DIY and maker communities, have encouraged alternative approaches. The past decade or so has seen the emergence of a raft of low-cost, embedded hardware platforms (a selection of which are gathered in Table 1), either providing audio support, or having audio as a key part of their *raison d'être*. Amongst these platforms are a selection based on Embedded Linux Systems (ELS) such as *Bela* [13] and *Elk* [14], companion boards for the BeagleBone Black [2] and Raspeberry Pi [3] respectively. These examples profit from the power of the underlying ELS, with a system-on-chip (SoC) providing a multicore CPU and peripherals such as memory and connectivity via, e.g., USB and ethernet. A separate subset based on microcontroller (MCU) systems, [4] prominent amongst these being the *Teensy* [15] (with an MCU from manufacturer NXP [5]), *Daisy* [16] (STMicroelectronics [6]), and various *ESP32* [17] development boards. Beyond matters of economy, connectivity, and modest power requirements, the attraction of such devices lies in the relative ease with which they can be programmed, each providing either an SDK, a selection of utility classes, or, in the case of ELS, compatibility with existing audio applications and plugins. Furthermore, it may be possible to produce compatible code in a higher level audio language such as Faust, in which case signal processing or audio synthesis code can be written once and compiled to run on a variety of platforms [18].

With frugality in mind, a modular, FPGA-based WFS system has been demonstrated, programmed in Faust and using high-quality, low-cost audio codecs [4]; a fully-costed DIY ambisonics dome, with 3D-printed speaker enclosures, is another recent contribution [3]. Both WFS and ambisonics have been attempted in distributed form, on GPUs [1] and MCUs [19], with ambisonics further enhanced by the use of an MCU supporting the Audio Video Bridging (AVB) protocol suite for audio processor synchronisation [20].

2.2 The Rise of Networked Audio

Also pivotal in facilitating more economical methods of achieving immersive audio outcomes has been the emergence of ethernet as the de facto standard in the transmission of multichannel digital audio [21, 22]. As high speed ethernet became more widely available at the beginning of the 2000's, audio over ethernet replaced prior analogue point-to-point systems, massively reducing complexity in broadcast and concert-hall scenarios, and driving development in networked music performance platforms such as JackTrip [23] and NetJack [24]. Serving the time-sensitive networking needs of a range of industries, 2002 saw the release of the IEEE 1588 Precision Time Protocol (PTP) [25]. Successor to the millisecond-order Network Time Protocol, PTP enables networked device synchronisation to the sub-microsecond level [26], and serves as the time-exchange mechanism for the industry standard media protocols, IEEE's AVB [27] and the Audio Engineering Society's AES67 [28]. PTP also underpins device synchronisation in proprietary systems such as Audinate's *Dante*. [7]

If improvements in networking technology and infrastructure — ubiquitous high-speed internet and gigabit ethernet switches — democratised and simplified multichannel audio, PTP once again raised the barrier to entry. For best accuracy, PTP requires that ethernet packets are timestamped at the hardware level, functionality that, to this day, is typically only available on high-end networking equipment. Rare exceptions to this rule do exist; the CRS326-24G-2S+ series [8] of switches from manufacturer MikroTik chief amongst them.

Perhaps surprisingly then, support for PTP is offered by a number of low-cost MCUs and SoCs, including the Broadcom BCM2712, as found on the Raspberry Pi Compute Module 5 [29], STM32H750IB [30] (Daisy Seed), and NXP iMXRT1060 series [31], the microcontroller utilised by the Teensy 4.1 development board. The difficulty comes in finding a platform that satisfies all of the particular re-

[1] For the logical conclusion to this technological story, look no further than HOLOPLOT's behemoth installation at *The Sphere*, Las Vegas: `https://holoplot.com/insights/case-studies/msg-sphere-case-study`.

[2] `https://www.beagleboard.org/boards/beaglebone-black`

[3] `https://www.raspberrypi.com/`

[4] The distinction between an SoC and a MCU is not without its subtlety. The latter tend to be more spartan, with a single processor core, limited memory, and basic peripherals; the former may have multiple cores and can encompass MCUs to provide specific functionality. SoCs tend to run an operating system and be more expensive than MCUs which typically run *bare-metal*.

[5] `https://www.nxp.com/`

[6] `https://www.st.com/`

[7] `https://www.audinate.com/`

[8] `https://mikrotik.com/product/crs326_24g_2s_in`

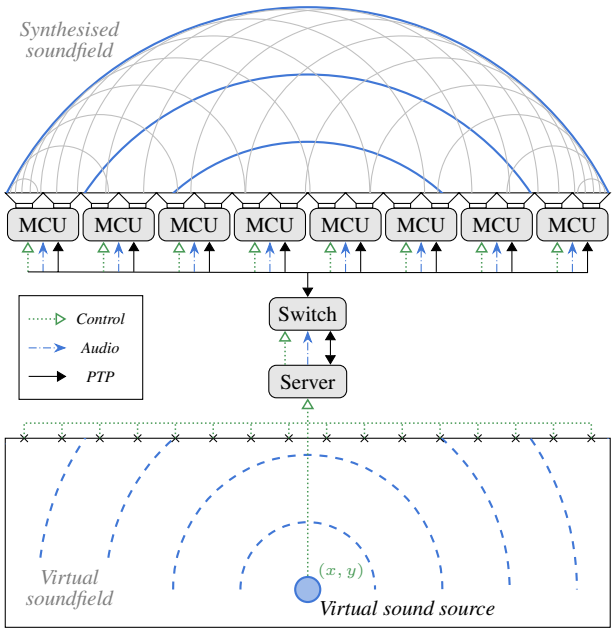


Figure 1: The holophonic effect of primary-source WFS is created by applying appropriate per-loudspeaker delays to a virtual sound source; delays are independent and can be computed in distributed fashion. A server delivers audio and control data to a collection of MCU-based signal processors via an ethernet switch; each MCU is informed of its position, and the position of the virtual sound source; MCU audio clocks are conditioned via PTP to match that of the server. Synchronicity amongst the group of MCUs is necessary to ensure the integrity of the synthesised wavefront.

quirements presented by a distributed spatial audio system. One may find, for example, that a sub-€30 development board such as the *Daisy Seed*, well-known to the DIY/maker audio community, well-appointed with memory, easily programmable in Faust, and *even possessing an MCU with PTP support*, has no Ethernet PHY,[9] and thus cannot be connected to a network. The Raspberry Pi CM5, coupled with associated I/O board, offers ethernet connectivity as standard, plus PTP, and is memory rich, but its audio clock is not configurable in fine enough increments to effect adjustments on the order of nanoseconds per second (ns/s), a level of precision that time-sensitive audio applications such as WFS demand (see Figure 1).

3. IMPLEMENTATION

Exact synchronisation in a distributed audio scenario is of paramount importance [32]; what, then, does it take to synchronise audio reproduction on physically separate audio devices? There are two factors to consider: 1) the rate of reproduction, i.e. the audio sampling frequency, and 2) simultaneity of reproduction, i.e. certainty to within reasonable limits that a given sample is reproduced at a predictable time.

[9] An ethernet PHY, or *physical layer interface*, typically takes the form of a dedicated chip, and facilitates the connection between a SoC and the physical LAN bus, a twisted-pair cable or fiberoptic connection for example [30].

3.1 Sampling Frequency Conditioning

For the first concern, two approaches are possible; one is to take incoming time-exchange information, over PTP for example, and use this information to condition the primary clock source — most likely a crystal oscillator (XO) — on a given device. This can be achieved by replacing the static XO with one that is voltage controlled (a VCXO), or with a direct digital synthesiser (DDS); in either case, a component that is capable of providing a variable clock frequency [33]. Such an approach requires that physical modifications are made to the device, an obstacle to accessibility that will be given no further consideration here. The second approach is to exploit the functionality of the audio subsystem of the SoC or MCU, which typically provides a hardware timer, derived from the XO, and adjustable via a phase locked loop (PLL), to govern the audio sampling rate [34]. The PLL is configurable via hardware registers representing dividers to be applied to the XO frequency to achieve the desired sampling frequency.

Whereas the Raspberry Pi's SoC provides only 12-bits of clock divider resolution, perhaps owing to the fact that it is an all-purpose device whose audio subsystem is governed by a general purpose I/O (GPIO) clock, the Teensy's NXP iMXRT1062 MCU, having a dedicated audio PLL, offers 30-bits. For this reason, the MCU's PTP support, and Bela's high price tag, Teensy 4.1 is the device upon which our implementation is based.

In the core Teensy codebase, the iMXRT1062's PTP functionality lies unimplemented. Fortunately, thanks to recent research efforts [26], PTP functions are now available and open-source. This functionality is only directly relevant to the MCU's ethernet subsystem, however; using it to condition the system's audio clock is a separate matter. A full description of the PTP algorithm lies beyond the scope of this paper, but essentially, following per-second exchange of timestamped ethernet packets between a clock authority and a clock subscriber, the latter, assuming a symmetrical transmission path, has enough information to calculate its offset from the former, producing a figure in nanoseconds by which, for a given second, it must adjust its ethernet timer. Taking this ns/s adjustment, and assuming that ethernet and audio clocks are derived from the same clock source, i.e. the device's 24 MHz XO (a safe assumption in the case of the iMXRT1062; see Figure 2), achieving sample rate parity is a relatively simple case of applying the adjustment, a, expressed in seconds, to the nominal sampling frequency, f_s, to produce a target sampling frequency, $\hat{f}_s$, such that

$$\hat{f}_s = f_s\,(1 + a)\ . \tag{1}$$

For example, should a clock subscriber, running at a nominal 48 kHz, drift fast over a given second by 1750 ns (1.75×10^{-6} s), the new sampling frequency to be applied should be

$$\begin{aligned}\hat{f}_s &= 48\,000\left(1 - 1.75 \times 10^{-6}\right)\\ &= 48\,000 \times 0.999\,998\,925\\ &= 47\,999.916\,\text{Hz}\ ,\end{aligned}$$

or about $\frac{1}{12}$Hz slower. An adjustment of 1 ns/s at a nominal 48 kHz corresponds to approximately one twenty-one-thousandth of a cycle of the audio clock. Whereas the Broadcom BCM2712 offers around $\frac{1}{14}$Hz resolution at this frequency, the iMXRT1062 provides better than one in six-hundred-thousand.

Note that the XO of the clock authority is manufactured to the same tolerance and stability specifications as those of the clock subscribers; none of these components produces a *true* 24 MHz clock frequency. It would be entirely possible to synchronise the clock authority to a PPS signal registered by a GPS receiver [26], but this of course implies the availability of a GPS signal, which, in an indoor context for instance, is far from guaranteed [34]. We consider the device designated as the clock authority as being a sufficiently reliable source of time for our purposes.

3.2 Reproduction Simultaneity

Sampling frequency conditioning alone will not suffice in achieving synchronous output; if clients start their respective audio subsystems on boot and simply apply sampling frequency adjustments once PTP time exchange begins, it is vanishingly unlikely that I^{2}S interrupts — moments at which samples are requested by the audio subsystem — will correspond. The solution is to wait until a PTP lock is achieved before starting the audio subsystem clock at the client side; on device startup, we configure the audio subsystem, but leave the audio clock and audio interrupts disabled, waiting until a PTP offset of less than 100 ns is achieved; at that point the audio clock is enabled, hardware registers are set that activate I^{2}S processing, and power delivered to the device's audio codec. [10]

4. EVALUATION

4.1 Measurement Setup

Four MCUs, one running as clock authority, three as clock subscribers, were connected via 50 cm CAT5E ethernet cables to a MikroTik CRS326-24G-2S+IN ethernet switch, which was configured to act as a PTP boundary clock. The clock authority also acted as an audio server, generating a 1 Hz impulse train and sending packets of audio data, timestamped 1 ms later than the measured instant of transmission, to the network on a multicast UDP address; additionally, the clock authority stored the outgoing packets in a ring buffer for reproduction by its own audio subsystem. Clock subscribers joined the multicast group, acting as audio clients, receiving audio packets from the server and writing these to their own respective ring buffers. On audio system startup, server and clients selected the audio packet timestamped closest to the instantaneous time reported by their own PTP reference clock; subsequently, all MCUs reproduced packets sequentially.

A digital logic analyser (Digilent Analog Discovery 2, with a sampling frequency of 100 MHz [11]) was connected to the I^{2}S data pin of each MCU, and accompanying software (Digilent *WaveForms* v. 3.23.4) used for data capture; the software's *Logic* mode was used, and set up to trigger a capture when the server/clock-authority reproduced an impulse. At each capture, the reproduction offset for each client/clock-subscriber relative to the server was measured and written to a CSV file.

Two test conditions were employed. For the first test, clock-subscribers did not apply PTP-derived audio sampling frequency correction, simply running at a nominal 48 kHz as derived from the frequency of their internal crystal oscillator. For the second, sampling frequency adjustments were made according to the method described in Section 3.

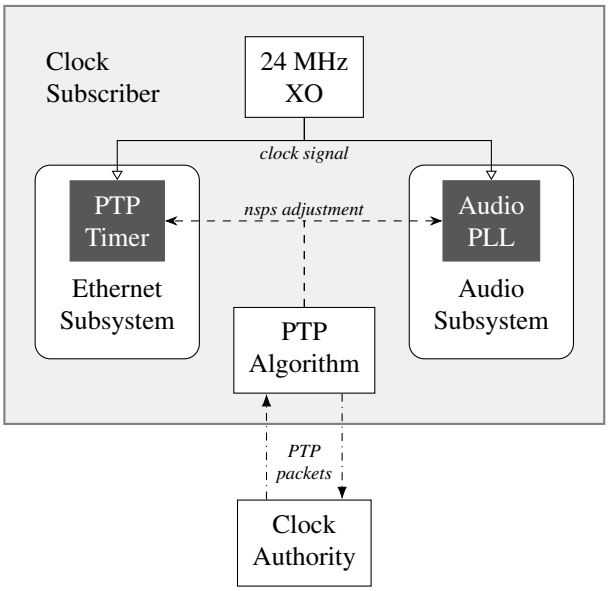


Figure 2: Overview of the audio clock conditioning system. The timing of a clock subscriber's subsystems is governed by a crystal oscillator (XO). Clock subscribers exchange PTP packets with a clock authority, and run an algorithm that determines the necessary adjustments to be made to a dedicated PTP timer, part of the ethernet subsystem. Those same adjustments are used to condition the timing of the audio subsystem by modifying divider registers that control the output frequency produced by the audio phase locked loop (PLL).

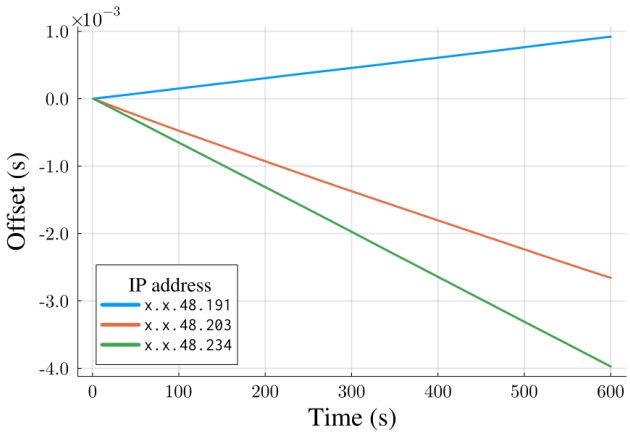
4.2 Results

Measurements for the two test conditions are visualised in Figure 3. In the absence of any manner of sampling frequency correction, Figure 3a essentially reports raw, quasi-linear clock drift for the three clients relative to the server, with two running faster than the server, the other slightly slower. The client with IP address `x.x.48.234` exhibits the most significant drift, ∼4 ms over the course of ten minutes equating to around 6.7 parts per million. [12] Assuming a sound propagation speed of 343 m/s in air at room temperature, in a practical application the inter-client spread of

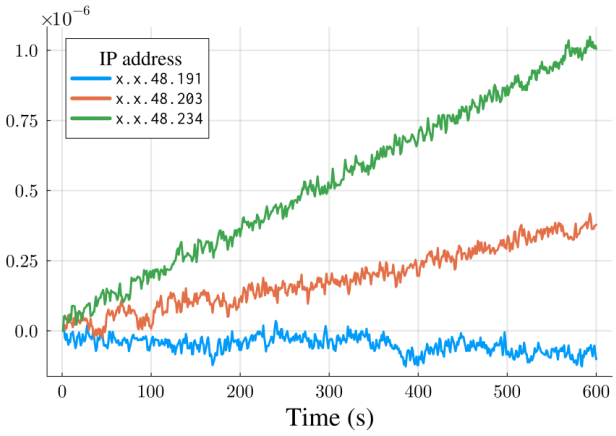
[10] Code for the proposed implementation can be found at `https://github.com/hatchjaw/teensy-audiosync/`.

[11] `https://digilent.com/reference/test-and-measurement/analog-discovery-2/specifications`

[12] Which suggests that, for this collection of devices at least, the Teensy 4.1 uses a reasonably high-quality crystal oscillator.



(a) Without sampling frequency correction.



(b) With PTP-derived sampling frequency correction.

Figure 3: Audio reproduction offsets for three MCU audio clients with respect to an MCU audio server. Server and clients reproducing a 1 Hz impulse train. Measurements triggered at the onset of the impulse produced by the server MCU, i.e. once per second for a total of ten minutes. For details of the test procedure, see Section 4.1.

approaching 5 ms at the end of the ten minutes would correspond with a propagation discrepancy of roughly 1.72 m.

Figure 3b is the equivalent plot for clients with sampling frequency correction applied. While some drift remains, note that it is three orders of magnitude less severe than in the correction-free case. Once again, `x.x.48.234` displays the greatest asynchronicity; now, however, its 1 µs over ten minutes represents less than 1.7 parts per *billion*. At the end of the test, the three clients are separated by an interval of just over 1 µs (marginally outside the AES guideline of 5% of the reference frame) corresponding to a propagation discrepancy of less than half a millimetre; we have improved from a scale comparable with the height of an adult human — sufficient to obliterate the integrity of any attempt at sound field synthesis — to an order of magnitude below that of the diameter of the eardrum.

Interestingly, and lending encouragement to the hope that further improvements are within reach, the remaining audio drift appears to be inversely proportional to the raw drift, with client `x.x.48.191` now tending very slightly fast, whereas previously it lagged behind.

5. CONCLUSION AND FUTURE WORK

We have demonstrated the feasibility of using the precision time protocol to synchronise reproduction across a distributed array of audio processors using open-source software and low-cost hardware. With distributed audio devices now producing output *all together*, this represents a key technological finding and will serve as impetus for future research on this topic.

If one were to identify a *weak link* in the proposed system, that might be the ethernet switch. Its manufacturer, Latvian enterprise MikroTik, is not what one might call a household name, [13] and it is striking that their 24-port, PTP-compliant, managed switch is offered at the comparatively low price of around €200 at the time of writing — Dante-enabled switches tend to be up to an order of magnitude more expensive. To support more than a notional 48 output channels (two per client) it would be necessary to daisy-chain switches; from a technical perspective, it is not known at the present time whether the PTP implementation on the MikroTik device can function effectively in such a configuration. We lie at the mercy of the market in this regard; an switch running on open-source software, and perhaps an FPGA, represents a long-term objective.

Besides establishing the cause of the remaining parts-per-billion drift and mitigating for it accordingly, future work will be concerned with extending to a system in which a general purpose computer can be used as the audio server, an MCU being unsuited to running software associated with established audio workflows. In prior work, we implemented an audio server and WFS controller in a digital audio workstation plugin; this approach should be adapted to the new model. A computer acting as an audio server in this way must also be a participant in PTP time exchange, so a cost-effective means to add PTP support, typically lacking in consumer-grade network interface controllers (such as one might find in a laptop computer), must be sought.

One key area of research that we wish to explore in the future is that of virtual acoustics and auralisation. The creation of virtual acoustic environments demands the execution of real-time impulse-response convolutions, which in turn imposes the requirement that a node in a distributed DSP system possess the computational power to perform such convolutions, the memory required to store the impulse responses, and sufficiently performant access to that memory for real-time processing. It remains to be seen whether the current hardware platform can rise to these challenges; we will endeavour to find out.

[13] Boosting hopes of the continued availability of their very useful ethernet switch, MikroTik has (according to its own reports) existed as a company for over twenty years — `https://mikrotik.com/aboutus`.

6. REFERENCES

[1] J. A. Belloch, J. M. Badía, D. F. Larios, E. Personal, M. Ferrer, L. Fuster, M. Lupoiu, A. Gonzalez, C. León,

A. M. Vidal, and E. S. Quintana-Ortí, "On the performance of a GPU-based SoC in a distributed spatial audio system," *The Journal of Supercomputing*, vol. 77, no. 7, pp. 6920–6935, Apr. 2021.

[2] T. A. Rushton, R. Michon, S. Serafin, T. Risset, and S. Letz, "Networked microcontrollers for accessible, distributed spatial audio," *Frontiers in Virtual Reality*, vol. 5, Nov. 2024.

[3] M. Mitterhuber, R. Sharafi, and E. Tomás, "Ottosonics," 2022. [Online]. Available: https://tamlab.kunstuni-linz.at/projects/ottosonics/

[4] R. Michon, J. Bizien, M. Popoff, and T. Risset, "Making Frugal Spatial Audio Systems Using Field-Programmable Gate Arrays," in *Proceedings of the 2023 New Interfaces for Musical Expression Conference (NIME-23)*, May 2023. [Online]. Available: https://inria.hal.science/hal-04169228

[5] M. Geronazzo and S. Serafin, Eds., *Sonic Interactions in Virtual Environments*, ser. Human–Computer Interaction Series. Springer Cham, 2023.

[6] S. Serafin, F. Avanzini, A. De Goetzen, C. Erkut, M. Geronazzo, F. Grani, N. C. Nilsson, and R. Nordahl, "Reflections from five years of Sonic Interactions in Virtual Environments workshops," *Journal of New Music Research*, vol. 49, no. 1, pp. 24–34, Jan. 2020.

[7] J. Ahrens, *Analytic Methods of Sound Field Synthesis*, ser. T-Labs Series in Telecommunication Services. Springer Science & Business Media, 2012.

[8] F. Winter, J. Ahrens, and S. Spors, "A Geometric Model for Spatial Aliasing in Wave Field Synthesis," in *Proceedings of the German Annual Conference on Acoustics (DAGA)*, Munich, Germany, Mar. 2018.

[9] R. Nicol, "Sound Field," in *Immersive Sound*. New York, USA: Routledge, 2017, pp. 276–310.

[10] F. Grani, D. Di Carlo, J. Madrid Portillo, M. Girardi, R. Paisa, J. S. Banas, I. Vogiatzoglou, D. Overholt, and S. Serafin, "Gestural Control Of Wavefield synthesis," in *Proceedings of the Sound and Music Computing Conference (SMC-16)*, Hamburg, Germany, Aug. 2016.

[11] M. Noisternig, T. Carpentier, and O. Warusfel, "Dispositif de spatialisation sonore 3Da l'Espace de Projection de l'IRCAM: Un Réseau de 345 haut-parleurs pour une restitution par WFS et HOA," *Acoustique et Techniques*, vol. 71, pp. 30–39, 2012.

[12] M. A. J. Baalman, T. Hohn, S. Schampijer, and T. Koch, "Renewed architecture of the sWONDER software for Wave Field Synthesis on large scale systems," in *Proceedings of the 5th Int. Linux Audio Conference*, Berlin, Germany, Apr. 2007.

[13] A. McPherson, "Bela: An embedded platform for low-latency feedback control of sound," *The Journal of the Acoustical Society of America*, vol. 141, no. 5 (supplement), p. 3618, May 2017.

[14] L. Turchet and C. Fischione, "Elk Audio OS: An Open Source Operating System for the Internet of Musical Things," *ACM Transactions on Internet of Things*, vol. 2, no. 2, pp. 12:1–12:18, Mar. 2021.

[15] R. Michon, Y. Orlarey, S. Letz, and D. Fober, "Real Time Audio Digital Signal Processing With Faust and the Teensy," in *Proceedings of the Sound and Music Computing Conference (SMC-19)*, Malaga, Spain, May 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03153709

[16] B. R. Gaster and M. Cole, "Audio Anywhere with Faust," in *Proceedings of the 2nd International Faust Conference (IFC-20)*, Paris, France, 2020.

[17] R. Michon, D. Overholt, S. Letz, Y. Orlarey, D. Fober, and C. Dumitrascu, "A Faust Architecture for the ESP32 Microcontroller," in *Proceedings of the Sound and Music Computing Conference (SMC-20)*, Turin, Italy, Jun. 2020. [Online]. Available: https://hal.science/hal-02988312

[18] R. Michon, Y. Orlarey, S. Letz, D. Fober, and D. Roosenburg, "Embedded Real-Time Audio Signal Processing With Faust," in *Proceedings of the 2nd International Faust Conference (IFC-20)*, Paris, France, Dec. 2020.

[19] T. A. Rushton, R. Michon, and S. Letz, "A Microcontroller-based Network Client Towards Distributed Spatial Audio," in *Proceedings of the Sound and Music Computing Conference (SMC-23)*, Stockholm, Sweden, 2023.

[20] S. Devonport and R. Foss, "The Distribution of Ambisonic and Point Source Rendering to Ethernet AVB Speakers," in *Proceedings of ICSA 2019*, Ilmenau, Germany, Sep. 2019.

[21] R. Bakker, A. Cooper, and A. Kitagawa, "An introduction to networked audio," Yamaha Commercial Audio Team, Rellingen, Germany, White Paper, 2014.

[22] P. Cochard, J. Weber, R. Michon, T. Risset, and S. Letz, "Ethernet Real-Time Audio Transmission to FPGA," in *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, Sep. 2024, pp. 1–7.

[23] J.-P. Cáceres and C. Chafe, "JackTrip/SoundWIRE Meets Server Farm," in *Proceedings of the 6th Sound and Music Computing Conference*. Porto, Portugal: The MIT Press, Jul. 2009, pp. 29–34. [Online]. Available: https://www.jstor.org/stable/40963030

[24] A. Carôt, T. Hohn, and C. Werner, "Netjack – Remote music collaboration with electronic sequencers on the Internet," in *Proceedings of the 7th Linux Audio Conference*, Parma, Italy, Apr. 2009.

[25] J. C. Edison, M. Fischer, and J. White, "IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, Reston, VA, USA, Dec. 2002.

[26] J. Schleusner, C. Fahnemann, R. Pfleiderer, and H. Blume, “Sub-Microsecond Time Synchronization for Network-Connected Microcontrollers,” in *2024 IEEE International Conference on Consumer Electronics (ICCE)*. Las Vegas, NV, USA: IEEE, Jan. 2024, pp. 1–6.

[27] IEEE, “IEEE Std 802.1BA-2011, IEEE Standard for Local and metropolitan area networks—Audio Video Bridging (AVB) Systems,” IEEE, Tech. Rep., 2011.

[28] A. Hildebrand, “AES67-2013: AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability,” in *Proceedings of the NAB Broadcast Engineering Conference*, Las Vegas, NV, USA, Apr. 2014.

[29] Raspberry Pi Ltd., “Raspberry Pi Compute Module 5: A Raspberry Pi for deeply embedded applications.” Nov. 2024. [Online]. Available: https://datasheets.raspberrypi.com/cm5/cm5-datasheet.pdf

[30] STMicroelectronics, “STM32H750VB STM32H750ZB STM32H750IB STM32H750XB - 32-bit Arm® Cortex®-M7 480MHz MCUs, 128 Kbyte flash, 1 Mbyte RAM, 46 com. and analog interfaces, crypto - Datasheet DS12556 Rev 7,” Mar. 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32h750ib.pdf

[31] NXP Semiconductors, “I.MX RT1060 Processor Reference Manual Rev 3,” Jul. 2021.

[32] R. Lienhart, I. Kozintsev, S. Wehr, and M. Yeung, “On the importance of exact synchronization for distributed audio signal processing,” in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP ’03).*, vol. 4, Apr. 2003, pp. IV–840.

[33] J.-M. Friedt, “Network synchronization of computers for timestamping under GNU/Linux : NTP, PTP and GPS on Raspberry Pi Compute Module 4,” *Hackable Magazine*, vol. 51, pp. 50–95, 2023.

[34] A. Carôt, H. Mahmood, and C. Hoene, “GNSS-based Sound Card Synchronization,” in *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*. IEEE, Sep. 2019, pp. 309–312.