

Ananas: A Networked Audio System for Distributed Spatial Audio Applications

Thomas Rushton

Inria, INSA Lyon

CITI, EA3720

69621 Villeurbanne, France

thomas.rushton@inria.fr

Romain Michon

Inria, INSA Lyon

CITI, EA3720

69621 Villeurbanne, France

romain.michon@inria.fr

Tanguy Risset

Inria, INSA Lyon

CITI, EA3720

69621 Villeurbanne, France

tanguy.risset@insa-lyon.fr

Abstract—Spatial audio has an accessibility problem. Large-scale spatial and immersive audio installations remain costly and exclusive, with high barriers to entry. We propose “Ananas,” a networked audio system consisting of network protocols, control software and microcontroller firmware, with the aim of rendering spatial audio applications accessible to a broad cohort of potential users. Based on a network of low-cost microcontroller development boards programmed in the Faust programming language, synchronised to nanosecond order via IEEE 1588 Precision Time Protocol, and controlled by approachable, digital audio workstation-based tooling, Ananas represents an opportunity to disrupt and democratise the spatial audio landscape. In this paper we demonstrate Ananas’s protocols for audio and control data transmission and device discovery, its methods for media clock and presentation time recovery, and its applications to wave field synthesis and higher order ambisonics in a modular, scalable form.

I. INTRODUCTION

Despite significant interest in the research and commercial spheres, access to large-scale spatial audio systems remains stubbornly out of reach for many interested parties. Highly multichannel audio interfaces are prohibitively expensive for all but the institutions with the deepest pockets; implementing a spatial audio system represents an investment in terms of money and expertise that places the endeavour beyond the reach of many artists, researchers and other potential users. In this paper we present Ananas — “*Ananas necessitates another networked audio system*” — a networked audio system which, being based on a low-cost microcontroller development board, open source software, and familiar digital audio workstation (DAW) tools, seeks to lower the barrier to entry to spatial audio techniques such as wave field synthesis (WFS) [1] and higher order ambisonics (HOA) [2].

The Ananas protocol suite describes a system of protocols for audio and control data transmission, device discovery and synchronisation. Taking advantage of multicast user datagram protocol (UDP) groups for audio transmission and certain elements of control data, Ananas minimises bandwidth, and, being based on IEEE 1588 Precision Time Protocol (PTP), ensures synchronisation amongst distributed hardware modules. To our knowledge, Ananas is the first open source system providing nanosecond order inter-module synchronisation, following millisecond order synchronised systems such as by Belloch et al. [3].

II. BACKGROUND

A wide array of networked audio protocols exist; why, then, create a new one? The answer to this question lies in the time-sensitive nature of spatial audio reproduction. In a distributed setting, synchronisation of hardware nodes is critical, and existing protocols are either proprietary, or do not cater to time-sensitive applications.

Chief amongst proprietary systems is Dante [4] (Digital Audio Network Through Ethernet), which uses PTP as its time-exchange mechanism. Dante-enabled hardware is typically very expensive, and, being a closed-source initiative, it is not suitable for further scholarly enquiry. Another closed-source platform can be found in the shape of the networking component of Elk Audio OS [5].

A variety of open source protocols and systems are available, such as JackTrip [6], [7], NetJack [8], part of the JACK audio connection kit, message-based systems such as Audio Over OSC [9] (the audio engine behind the audio streaming application Sonobus¹), jamming-focused platforms such as Jamulus [10] and Soundjack [11], and the audiovisual performance streaming platform LOLA [12]. What these systems have in common, however, is a focus on networked music performance and concert streaming, applications where low latency, and not synchronicity, is of greatest concern. For time-sensitive applications, this points to the need for a new networked audio system, one that ensures synchronicity between distributed signal processing modules.

Unlike Devonport and Foss [13], and Schultz et al. [14] we opted not to use Audio Video Bridging (AVB) or AES67 due to the high complexity associated with implementing these protocols in a microcontroller context.

A. The Hardware Platform

Effective deployment of a distributed spatial audio system is dependent on the identification of a suitable hardware platform for the distributed nodes. Such a platform should be low-cost and powerful, with support for audio and ethernet. The Teensy 4.1 microcontroller development board [15] was identified as satisfying these requirements, costing less than € 50 with its audio and ethernet add-ons, and sporting a 600 MHz ARM

¹<https://sonobus.net>

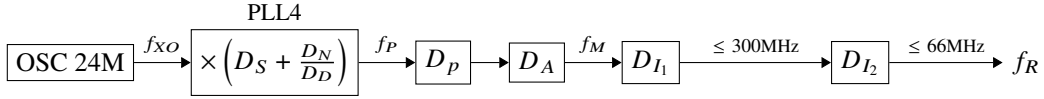


Fig. 1. iMXRT1062 audio PLL and SAI1 clock divider registers.

Core M7 CPU. Unlike other comparable devices, such as the Raspberry Pi, its microcontroller, an NXP iMXRT1062 MCU, features a dedicated audio subsystem, with high-resolution clock divider registers for its audio phase locked loop (PLL); 30 bits are available for the PLL's numerator and denominator registers, meaning that nanosecond-order modifications of the audio sampling rate are possible [16].

III. IMPLEMENTATION

By convention, the output frequency produced by the iMXRT1062's synchronous audio interface SAI1_CLK_ROOT, f_R , must be 256 times oversampled with respect to the target audio sampling rate, f_s :

$$f_R = 2^8 f_s. \quad (1)$$

The NXP MCUXpresso Config Tools software [17] indicates that f_R must be lower than or equal to 66 MHz, therefore

$$f_s \leq 257\,812.5 \text{ Hz}. \quad (2)$$

f_R is found by dividing the frequency of PLL4_MAIN_CLK, f_M , by the SAI1_CLK_PRED, D_{I1} , and SAI1_CLK_PODF, D_{I2} fields of the CCM_CS1CDR register:

$$f_R = \frac{f_M}{D_{I1} D_{I2}}. \quad (3)$$

f_M is found by dividing f_P , the frequency of PLL4, by the POST_DIV_SELECT, D_P , and AUDIO_DIV, D_A , registers:

$$f_M = \frac{f_P}{D_P D_A}. \quad (4)$$

f_P is the result of multiplying the iMXRT1062's 24 MHz clock source, f_{XO} , by the PLL4 divider, D_P :

$$f_P = f_{XO} D_P, \quad (5)$$

and D_P is, in turn, the sum of the *PLL loop divider*, the DIV_SELECT field of the CCM_ANALOG_PLL_AUDIO register, D_S , with the quotient of CCM_ANALOG_PLL_AUDIO_NUM, D_N , the *PLL fractional loop divider* numerator register and CCM_ANALOG_PLL_AUDIO_DENOM, D_D , the denominator register:

$$D_P = D_S + \frac{D_N}{D_D}. \quad (6)$$

See figure 1 for a diagrammatic representation of the effect of the audio PLL and clock divider registers on the 24 MHz clock source.

The DIV_SELECT field has a valid range of $27 \leq D_S \leq 54$, and $|D_N| < D_D$, so D_P describes (in theory) a number in the following range:

$$\left(26 + \frac{1}{D_D}\right) \leq D_P \leq \left(54 + \frac{D_D - 1}{D_D}\right), \quad (7)$$

with $1 \leq D_D \leq 2^{30}$; effectively, $26 < D_P < 55$. MCUXpresso Config Tools, and the iMXRT1062 reference manual [16, §13.3.2.2], indicate, however, that the output frequency range for PLL4 is:

$$650 \text{ MHz} \leq f_P \leq 1\,300 \text{ MHz}, \quad (8)$$

plus Config Tools contradicts the reference manual and states that D_N must be a *positive* 30 bit integer, so:

$$\left(27 + \frac{1}{12}\right) \leq D_P \leq \left(54 + \frac{1}{6}\right). \quad (9)$$

Setting extreme (but valid) values for the divisors, we can find the minimal valid f_s :

$$309.9 \text{ Hz} \leq f_s \leq 257\,812.5 \text{ Hz}. \quad (10)$$

A. A General Expression for the Sampling Rate

Bringing all of the above together, the general expression for achieving a target sampling rate $\hat{f}_s$ is

$$\begin{aligned} \hat{f}_s &= \frac{f_R}{2^8} \\ &= \frac{f_M}{2^8 D_{I1} D_{I2}} \\ &= \frac{f_{XO} D_P}{2^8 D_P D_A D_{I1} D_{I2}} \\ &= \frac{24 \times 10^6}{2^8} \frac{D_S + \frac{D_N}{D_D}}{D_P D_A D_{I1} D_{I2}}. \end{aligned} \quad (11)$$

This reckoning alone does not take into account certain constraints on the system, however, such as equation (8) and the frequency at the input of D_{I2} being less than 300 MHz (see figure 1). There are too many free variables to solve for any one of them, so an iterative approach to finding appropriate register and field values is necessary.

B. Setting the Sampling Rate

Say we wish to set a sampling rate of $f_s = 48 \text{ kHz}$; our target f_R , $\hat{f}_R$ equals, as per equation (1)

$$\hat{f}_R = 48\,000 \times 2^8 = 12\,288\,000 \text{ Hz}. \quad (12)$$

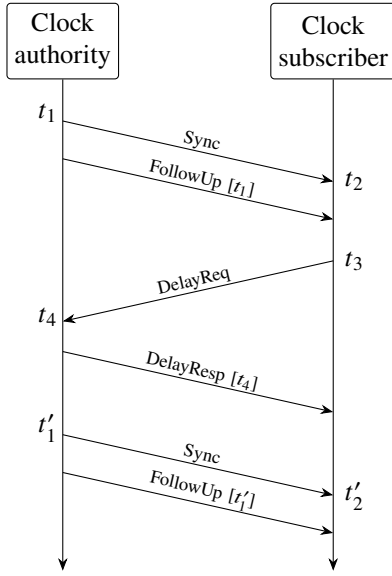


Fig. 2. PTP packet flow.

Strategically leaving D_P and D_A set to their default values of 1, values for D_S , D_N , D_D , D_{I_1} and D_{I_2} can be found:

- $D_S = 29$
- $D_N = 184\,000\,000$
- $D_D = 1\,000\,000\,000$
- $D_{I_1} = 3$
- $D_{I_2} = 19$

C. Fine-Tuning the Sampling Rate

Say a set of divider register values has been established and some manner of fine adjustment is desired, e.g., to compensate for clock drift. Equation (11) can be rewritten to find an expression for an unknown D_N :

$$D_N = D_D \left(\frac{2^8 \hat{f}_s D_P D_A D_{I_1} D_{I_2}}{24 \times 10^6} - D_S \right). \quad (13)$$

For very large D_D it may be advantageous to divide by the main oscillator's frequency in MHz instead of Hz; to do so, multiply the contents of the parenthesis in equation (13) by one million and divide the outside by one million to compensate:

$$D_N = \frac{D_D}{1 \times 10^6} \left(\frac{2^8 \hat{f}_s D_P D_A D_{I_1} D_{I_2}}{24} - 1 \times 10^6 D_S \right). \quad (14)$$

D. Media Clock Recovery

The Teensy's NXP iMXRT1062 MCU also features a dedicated hardware PTP timer [18] thus allowing for effective media clock recovery via an appropriate algorithm.

PTP is a time exchange mechanism consisting of the exchange of multicast network packets between a clock authority and a clock subscriber [18], [19]. PTP packet flow is illustrated in figure 2. Flow begins with the clock authority sending a *sync* packet; the subscriber notes the time, t_2 , at which this packet

arrives. The authority then sends a *follow-up* packet which is timestamped, t_1 , with the time of transmission. With t_1 and t_2 in hand, the subscriber sends a *delay request* packet and notes the time, t_3 of transmission; the authority notes the time, t_4 of reception of this packet and sends it to the subscriber in a final *delay response* packet. The above process then repeats for the next time period, typically a second, beginning with the receipt time t'_2 of the following sync packet and the receipt of the next follow-up packet containing timestamp t'_1 . Once the subscriber has the above timestamp information it can begin to compute its drift, α , delay, δ , and offset, θ relative to the authority:

$$\alpha = 1 - \frac{t'_2 - t_2}{t'_1 - t_1}, \quad (15)$$

$$\delta = \frac{(t_4 - t_1) - (t_3 - t_2)}{2}, \quad (16)$$

$$\theta = t_2 - t_1 - \delta. \quad (17)$$

Armed with the above figures, the subscriber can feed them into a proportional integral (PI) controller to yield a number of nanoseconds for second n by which it should adjust its internal clock,

$$a_n = \alpha - K_P \theta_n + K_I \sum_{t=0}^n \theta, \quad (18)$$

where K_P and K_I are co-efficients to be applied to the proportional and integral terms respectively; small co-efficients, typically $K_I < K_P < 1$. The adjustment value a_n can be used to set, for example, the per-cycle increment of a hardware ethernet timer register.

Time exchange at the level of the ethernet subsystems of distributed modules does not directly help us achieve audio sampling rate parity. The time exchange information derived from PTP must be used to condition the audio clocks of the modules such that sampling rate parity is achieved; this is a process referred to as *media clock recovery* [20].

In the case of the iMXRT1062 MCU this process is achieved by using the nanoseconds-per-second PTP-derived adjustment value at second n , a_n , to adjust the sampling rate. For a nominal sampling rate f_s (e.g., 48 kHz), the sampling rate, $\hat{f}_{s,n}$ that “matches” that of the time authority is found simply as

$$\hat{f}_{s,n} = f_s (1 + a_n). \quad (19)$$

For example, if $a_n = -1750$ ns, we have

$$\begin{aligned} \hat{f}_{s,n} &= f_s (1 - 1750 \times 10^{-9}) \\ &= 48\,000 \times 0.999\,998\,25 \\ &= 47\,999.916 \text{ Hz}. \end{aligned}$$

Inserting this value into equation (11), the expression for the audio sampling rate of the iMXRT1062, yields

$$47\,999.916 = \frac{24 \times 10^6}{2^8} \frac{D_S + \frac{D_{N,n}}{D_D}}{D_P D_A D_{I_1} D_{I_2}}, \quad (20)$$

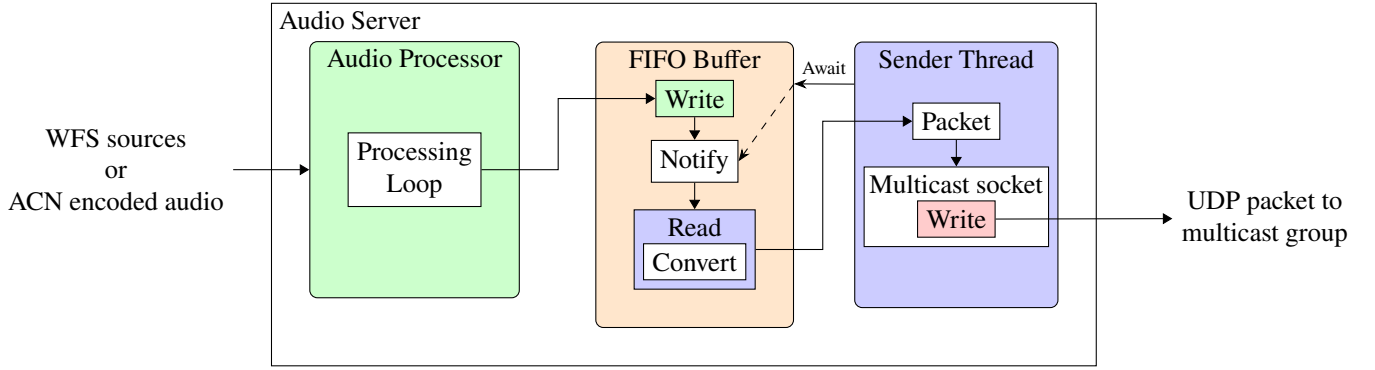


Fig. 3. Structural overview of the Ananas audio server.

and rearranging, using equation (14) to find the appropriate value for the PLL4 numerator register

$$D_{N,n} = \frac{D_D}{1 \times 10^6} \left(\frac{2^8 \times 47\,999.916 \times D_p D_A D_{I_1} D_{I_2}}{24} - 1 \times 10^6 D_S \right) \quad (21)$$

Using the values derived in section III-B yields

$$D_{N,n} = 1 \times 10^3 \left(\frac{2^8 \times 47\,999.916 \times 3 \times 19}{24} - 29 \times 10^6 \right) = 183\,948\,928,$$

which can be used as a new value for the CCM_ANALOG_PLL_AUDIO_NUM register of the iMXRT1062 MCU.

In practice equation (19) does not suffice, however, and some residual parts per billion drift remains. The proposed solution is as follows: first, observing that at a nominal sampling rate of 48 kHz for power of two audio buffer sizes less than or equal to 128 there will be an integer number, I , of audio interrupts per second, take a representative nanoseconds figure, t_S , on audio system startup; second, every I interrupts, reading the system's PTP timer, taking the nanoseconds part, t_{PTP} and comparing with t_S :

$$\Delta_n = t_{PTP} - t_S; \quad (22)$$

finally, applying another PI controller:

$$\hat{f}_{s,n} = f_s \left(1 + a_n + K_{P_\Delta} \Delta_n + K_{I_\Delta} \sum_{t=0}^n \Delta \right). \quad (23)$$

These additional correction factors ensure reliable media clock recovery indefinitely.

IV. THE ANANAS NETWORKED AUDIO SYSTEM

Ananas is a system for time-sensitive networked audio for supporting distributed approaches to sound field synthesis. In the sections that follow we describe the various components of the Ananas system.

A. The Audio Server

The Ananas audio server is a C++ library written using the JUCE framework.² On startup, the library initialises a series of threads for handling audio transmission, reading PTP timestamps and listening for connected clients, amongst other tasks. The audio transmission part of the server is illustrated in figure 3. The server receives a collection of input channels, either monophonic WFS sources or ambisonics channel numbering (ACN) encoded audio channels, and writes these to an intermediate first-in-first-out (FIFO) buffer. When the FIFO contains enough samples to fill an outgoing audio packet, the sender thread is notified and reads from the FIFO to fill a packet for transmission. The sender thread delivers the packet to a UDP multicast socket and transmits it to the network.

B. The Ananas Audio Protocol

Being first and foremost concerned with the transmission of audio data, at the heart of the Ananas system is the Ananas Audio Protocol (AAP). A UDP multicast-based protocol, using address 224.4.224.4, taken from the second ad-hoc block of IANA's multicast address space [21], AAP consists of a stream of sequence numbered, timestamped packets. The structure of an AAP packet is given in figure 4. It was found that, particularly under high-bandwidth conditions, packets sporadically arrive out of order, hence the inclusion of the sequence number, which is used at the receiving side to ensure the integrity of the packet stream. For timestamping, the server reads a PTP follow-up packet, adds a small reproduction offset (for packets of 16 frames it was found that an offset of 1/70 s was most appropriate) and writes the timestamp to the outgoing packet; successive packets are then updated with the expected inter-packet interval, e.g., for 16 frames at 48 kHz this is 333 333 $\frac{1}{3}$ ns.

Completing the packet header are the number of channels and number of frames in the packet, used at the receiving side for decoding the byte-encoded audio data that follows. At 16 frames per packet it is possible to transmit up to 44 channels of 16-bit audio while respecting ethernet's 1500 byte *maximum transmissible unit*.

²JUCE v8.0.6 <https://juce.com>.

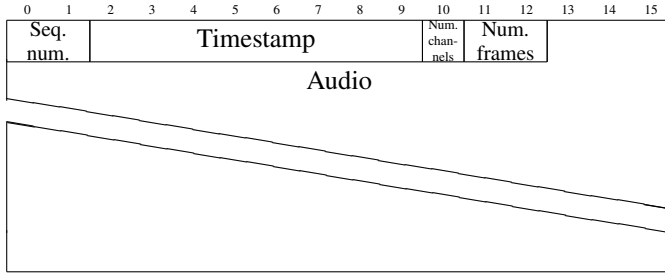


Fig. 4. Structure of an Ananas audio packet.

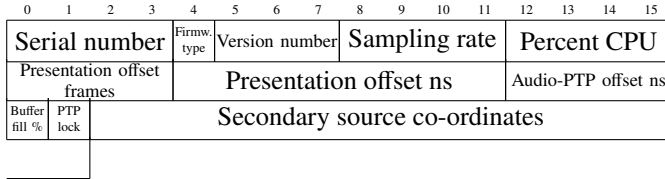


Fig. 5. Structure of an Ananas Client Description packet.

C. The Ananas Client Description Protocol

Clients announce their presence on the network with periodic UDP multicast packets on address 224.4.224.6. A client is deemed to be connected if the server has received an Ananas Client Description Protocol (ACDP) packet within the last second. Each packet contains a variety of information about a given client, such as its serial number, firmware type and version, reported sampling rate, buffer fill percentage and PTP lock status, etc., information which can be displayed in a digital audio workstation plugin interface. The structure of an ACDP packet is given in figure 5.

D. The Time Authority

One Teensy 4.1 is designated as an *authoritative source of time* for the entire system. It is used as a USB audio device for the server machine, a general purpose laptop computer, and as a PTP clock authority for all other devices on the network. Since PTP time and USB audio timing on the time authority are governed by the same crystal oscillator, in serving this hybrid role the time authority ensures synchronicity between the server machine, the ethernet switch, which acts as a PTP boundary clock, and, by extension, all other Teensy 4.1-based modules in the system.

E. Control Systems

Control data is sent as OSC messages over UDP. For WFS, hardware modules are assigned their positions in the speaker array, i.e. given their secondary source (loudspeaker) positions over unicast UDP. E.g., a given module may receive its secondary source x-position via an OSC message of the form `/ss/0/x 1.5`, indicating that its first secondary source is at x-coordinate 1.5 m. WFS virtual sound source positions are delivered as multicast OSC messages on address 224.4.224.5 and are of the form `/vs/0/y -0.5`, indicating that the first virtual sound source's y-coordinate should be -0.5 m.

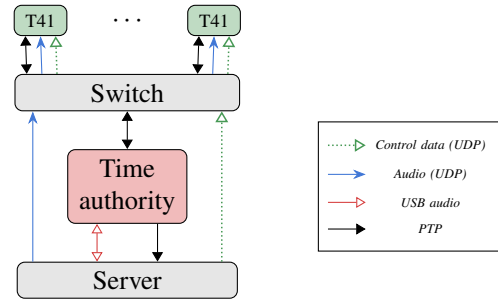


Fig. 6. Overview of the Ananas networked audio system.

For ambisonics, modules instead receive their decoder coefficients over unicast UDP, again as OSC messages. A module may for example receive a message of the form `/dm/0/0 0.2385`, indicating that it should use a coefficient of 0.2385 for the first element in the first row of the decoder matrix.

F. System Overview

An overview of the Ananas networked audio system is given in figure 6. The time authority governs USB audio timing on the server machine and PTP time for the switch, a MikroTik CRS326-24G-2S+IN³ chosen for its high port count, PTP support and low cost (~€ 200), and other connected modules. The server sends audio and control data to the switch, which forwards this data on to the connected modules, which perform their part of a distributed spatial audio algorithm.

Code for the Ananas server can be found at <https://github.com/hatchjaw/ananas-server>; code for the Ananas client and time authority can be found at <https://github.com/hatchjaw/ananas-client>.

V. DEPLOYMENT



Fig. 7. Ananas driving a 16-channel WFS system.

Deployed as per figure 6, Ananas can operate either as a WFS or a HOA system. DSP, programmed in the Faust language, is distributed amongst the hardware modules; for WFS, modules

³https://mikrotik.com/product/crs326_24g_2s_in

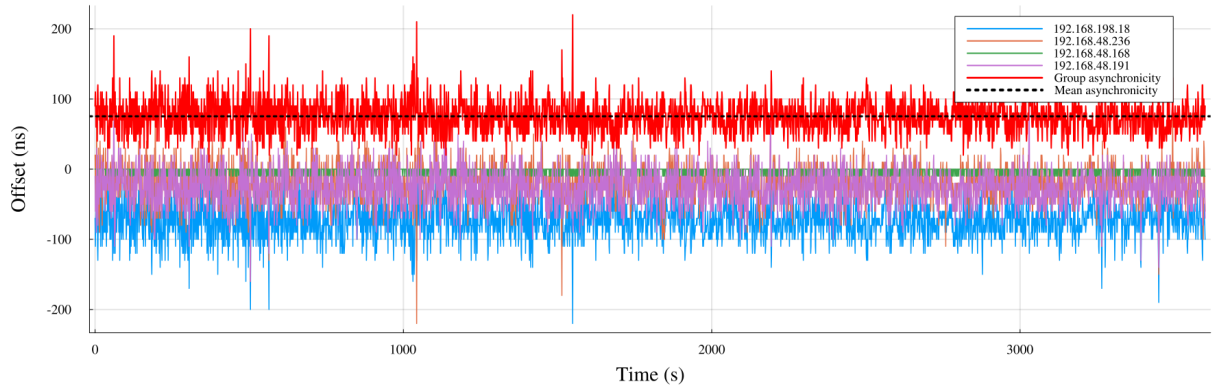


Fig. 8. Synchronisation measurements for a group of four Teensy modules reproducing a pulse-per-second audio signal. Group asynchronicity represents the instantaneous range of offsets for all four modules.

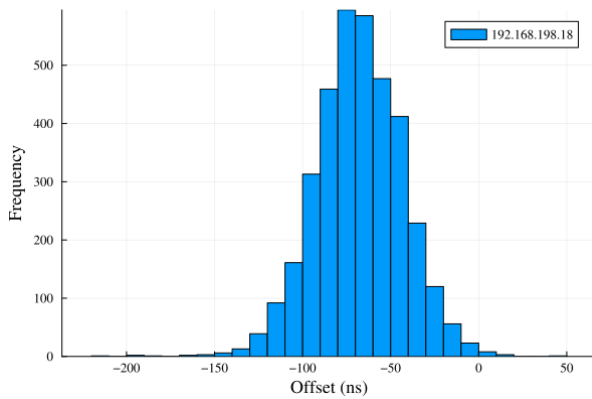


Fig. 9. Histogram of synchronisation measurements for one of the clients in figure 8.

need only know the spatial coordinates of their loudspeakers and the coordinates of the virtual sound sources to produce their part of the synthesised sound field; for HOA, modules are provided with the rows of the decoder matrix for which they are responsible and delivered ACN-encoded audio streams.

An example WFS setup is pictured in figure 7. Timing in the system is governed by the time authority Teensy described in section IV-D. The server machine, a general purpose laptop computer, runs the Reaper⁴ digital audio workstation with a CLAP format WFS control plugin assigned to a 16-channel track, and sends monophonic audio streams to a UDP multicast group managed by a single switch, which forwards the data on to the hardware modules. Eight modules are used in this setup for a total of sixteen output channels. The server sending packets containing sixteen frames of audio data, it is possible to transmit up to 44 channels, each representing either a WFS virtual source or an ACN-format channel; up to fifth order HOA is supported.

Thanks to the media clock recovery algorithm described in section III-D, modules remain synchronised to within around 200 ns at all times. Figure 8 shows an hour's worth of syn-

chronicity measurements for a group of four hardware modules. The modules were connected to a Digilent Analog Discovery 2 logic analyser and their I²S output measured in Digilent Waveforms software. The server delivered a pulse-per-second audio signal and offsets were measured at the rising edge of the pulse. The mean asynchronicity of ~ 80 ns falls well within the AES guideline for time-sensitive applications of 5% of the reference frame ($\sim 1 \mu\text{s}$ at 48 kHz) and corresponds with a propagation discrepancy in air of roughly $27.4 \mu\text{m}$. A histogram of synchronisation data for client 192.168.198.18 is presented in figure 9.

VI. CONCLUSION

In this paper we have presented *Ananas*, a new networked audio system for time-sensitive applications. With its low-cost, microcontroller-based hardware platform, it is modular and scalable and familiar digital audio workstation tools make it approachable to non-technical users. Modules are synchronised to extremely high precision thanks to media clock and presentation time recovery algorithms. *Ananas* represents an opportunity to democratise spatial audio, rendering WFS and HOA accessible to a broad potential userbase.

In terms of future work, we aim to increase the channel count available on each hardware module via time-division multiplexing, further reducing the cost per output channel associated with the *Ananas* system.

REFERENCES

- [1] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, May 1993. [Online]. Available: <https://asa.scitation.org/doi/abs/10.1121/1.405852>
- [2] F. Zotter and M. Frank, *Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*, ser. Springer Topics in Signal Processing. Cham: Springer International Publishing, 2019, vol. 19.
- [3] 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.
- [4] Dante, "What is Dante? | Audinate | Dante Pro AV Networking," 2022. [Online]. Available: <https://www.audinate.com/meet-dante/what-is-dante>

⁴<https://reaper.fm>

- [5] 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.
- [6] J.-P. Cáceres and C. Chafe, “JackTrip/SoundWIRE Meets Server Farm,” *Computer Music Journal*, vol. 34, no. 3, pp. 29–34, Sep. 2010.
- [7] —, “JackTrip: Under the Hood of an Engine for Network Audio,” *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, Sep. 2010.
- [8] 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.
- [9] W. Ritsch, “Towards Message Based Audio Systems,” in *Proceedings of the Linux Audio Conference*, Karlsruhe, Germany, 2014.
- [10] V. Fischer, “Case Study: Performing Band Rehearsals on the Internet With Jamulus,” 2015.
- [11] A. Renaud, A. Carôt, and P. Rebelo, “Networked Music Performance : State of the Art,” in *30th AES International Conference on Intelligent Audio Environments*, Saariselkä, Finland, Mar. 2007.
- [12] C. Drioli, C. Allocchio, and N. Buso, “Networked Performances and Natural Interaction via LOLA: Low Latency High Quality A/V Streaming System,” in *Conference Proceedings of the Second International Conference on Information Technologies for Performing Arts, Media Access and Entertainment, ECLAP*. Porto, Portugal: Springer, Apr. 2013, pp. 240–250.
- [13] S. Devonport and R. Foss, “The Distribution of Ambisonic and Point Source Rendering to Ethernet AVB Speakers,” in *Proceedings of ICISA 2019*, Ilmenau, Germany, Sep. 2019.
- [14] F. Schulz, M. Weidauer, S. Weinzierl, and H. von Coler, “SeamLess: Distributed Spatial Audio Rendering on the Linux Audio Stack,” in *Proceedings of the 20th Linux Audio Conference (LAC-26)*, Maynooth, Ireland, 2026.
- [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] NXP Semiconductors, “i.MX RT1060 Processor Reference Manual Rev 3,” Jul. 2021.
- [17] —, “MCUXpresso Config Tools User’s Guide (Desktop),” 2018.
- [18] J. Schleusner, C. Fahnenmann, R. Pfeleiderer, 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.
- [19] K. B. Lee, J. C. Eidson, H. Weibel, and D. Mohl, “Proceedings of the 2005 Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” Oct. 2005.
- [20] K. Crankshaw and S. Winberg, “Media Clock Recovery Over Ethernet Using IEEE 1722 on an FPGA,” *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2021.
- [21] D. Meyer, M. Cotton, and L. Vegoda, “IANA Guidelines for IPv4 Multicast Address Assignments,” Internet Engineering Task Force, Request for Comments RFC 5771, Mar. 2010.