Acta Acustica 2024, **8**, 18

https://doi.org/10.1051/aacus/2024011

Available online at:
https://acta-acustica.edpsciences.org

TECHNICAL & APPLIED ARTICLE OPEN ACCESS

Interactive low delay music and speech communication via network connections (OVBOX)

Giso Grimm*

Carl von Ossietzky Universität Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany

Received 11 January 2024, Accepted 15 March 2024

Abstract – The "OVBOX" is a tool for low-delay network audio communication and generic data transmission between multiple clients. Acoustic end-to-end delays of about 30 ms can be achieved, assuming a good internet connection and a distance between clients of less than about 1500 km. For optimal spatial perception, e.g., when using multiple instruments of similar timbre, an individual 3-dimensional room acoustic simulation based on physical modelling is applied in each client. The system is optimised for headless operation using a dedicated single-board computer (Raspberry Pi 4B), but desktop clients are also available for better integration with other audio software. A client-server system allows remote configuration and automatic traversal of network address translation routers and firewalls. With the low latency that can be achieved, the "OVBOX" is used for music applications such as distributed rehearsals or concerts. Other applications include hearing research to achieve interactive speech communication with low delay transmission of head movements for real-time control of virtual reality, and transmission of other biophysical data for online analysis or central data logging. The tool is fully open source.

Keywords: Network audio, Virtual acoustics, Remote music rehearsals, Communication research, Interactive distributed binaural rendering

1 Introduction

The 2020–2022 pandemic, with its frequent lockdowns and social distancing, has had a huge impact on musical performance, but also on research with human participants. This has led to increased use of network-based collaboration tools for music and telepresence communication. A number of tools are currently available for remote music collaboration (e.g., Jamulus [1], Soundjack [2], JamKazam [3], Jacktrip [4]). For telepresence communication, tools such as Zoom, Microsoft Teams, or Webex are commonly used.

Low end-to-end delay is required for remote music collaboration, especially when playing rhythmic music such as classical or jazz [5, 6]. On general-purpose desktop computers, low-latency audio processing may require manual optimisation of the operating system, which requires expert knowledge. This means that without such optimisation, desktop applications may not be able to achieve sufficiently low latency. Pre-configured operating systems that include such optimisations can be provided if the hardware is known, e.g. for the Raspberry Pi.

When ensemble music is played in a real room, the musicians have a fixed relative position. In rehearsals, circular arrangements of musicians are often used, as was common in 17th-century consort music [7]. In a jazz trio, for example, the bassist may have the drums on the left and the piano on the right, then the drummer will have the bassist on the right and the piano on the left, and so on. Especially when the timbre of the instruments involved is similar, a physically matched spatial reproduction with an individual three-dimensional acoustic rendering of the virtual sound sources is beneficial for a realistic perception.

Individual mixing of relative sound levels is another feature of remote music collaboration tools that is not available in all of them, but is desired by many musicians. Despite basic physical modelling, there may still be differences in the perception of sound levels due to complex radiation patterns or personal preferences.

Research applications typically require more control over the session than videoconferencing tools offer, such as access to individual audio streams or the transmission of arbitrary additional data for distributed sensors. For example, hearing research applications sometimes require the transmission of head movement data from remote participants [8, 9].

To overcome these problems, a custom solution was developed based on the network audio tool "zita-njbridge" [10], which proposes one solution to the problem of adaptive

*Corresponding author: g.grimm@uol.de

resampling in network audio applications [11]. Remote sound sources are rendered in a 3D audio engine, the Toolbox for Acoustic Scene Creation and Rendering (TASCAR) [12]. Room acoustics and spatially distributed source positions are simulated, optionally including head tracking. This system was originally developed to enable rehearsals of the ensemble "ORLANDOviols" during the Covid19 pandemic; the name OVBOX stands for "ORLANDOviols consort box".

This paper describes the structure of the system with the interaction of its different components, the implementation and typical applications. Delay constraints and computational performance data are presented and discussed.

2 System architecture

The general structure of the OVBOX includes several components, see Figure 1 for an overview: the *client device*, which is the actual OVBOX, typically a combination of a Raspberry Pi single board computer and an audio interface; a *configuration server*, which provides the user interface; and a *session server*, which is responsible for the actual exchange of data between clients. The central part is the configuration server, which consists of a web interface for users and a Representational State Transfer Application Programming Interface (REST API) for client devices. Each client device, typically identified by its Media Access Control (MAC) address, is associated with a user account. A session is established when two or more devices enter a "meeting room". Device configuration and selection of a "meeting room" can be managed by the user. When a client device is started, it connects to the configuration server's REST API and obtains its configuration, including the IP address, port number and session key of the selected room. The session server is a hybrid of a Selective Forwarding Unit (SFU) and a management server for a peer-to-peer (P2P) architecture. An experimental mode is also available where the session server acts as a Multipoint Control Unit (MCU) with server-side mixing, in which case a dedicated client running on the session server acts as a down-mixer. A typical network topology is shown in Figures 2 (server mode, SFU architecture) and 3 (P2P architecture). Sessions with a mixture of SFU and P2P network topologies are also possible.

The OVBOX client software consists of four main components: A REST API client to receive configuration and optional session server information; a generic User Datagram Protocol (UDP) forwarding tool that communicates with the session server and, in peer-to-peer mode, with all other clients; a network audio client with adaptive resampling that can receive audio from UDP packets as well as a corresponding sender (zita-njbridge, [10, 11]); and an acoustic simulation tool (TASCAR, [12]), for individual three-dimensional acoustic rendering of the sound sources in a virtual acoustic environment. The network audio client communicates with the generic UDP forwarding tool, which handles all communication with the session server and other clients, and includes a minimal access control and error

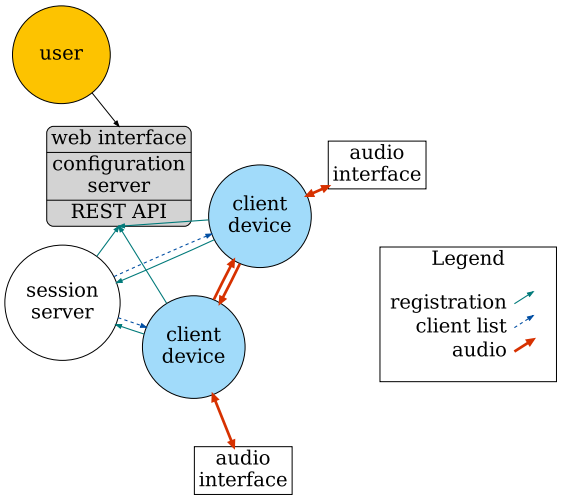

Figure 1. Schematic overview of the system architecture. Each *client device* connects to the Representational State Transfer Application Programming Interface (REST API) of the *configuration server* to obtain its configuration and session settings, which can be controlled by users via a web interface. When a session is to be established, one or more client devices connect to a *session server*.

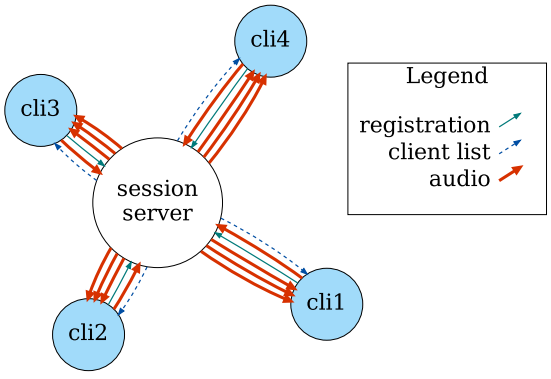

Figure 2. Example of an OVBOX session with four clients, all operating in server mode. The session server acts as a Selective Forwarding Unit (SFU). Client registration with the session server is repeated every few seconds. In response, the session server returns a list of all active clients. In this example, each client is sending 0.9 MBit/s and receiving 2.7 MBit/s.

correction mechanism. In SFU mode, data can optionally be transmitted to the session server via a Transmission Control Protocol (TCP) tunnel to prevent packet loss at the cost of increased network delay and jitter. By default, audio is sent as uncompressed 16-bit audio at a sampling rate of 48 kHz. Other options are 24-bit or 32-bit floating point resolution per sample, with any sampling rate supported by the audio interface. The data is sent uncompressed because data compression typically requires re-buffering and therefore increases delay. Together with some packaging overhead, this results in a bandwidth requirement of approximately 0.9 MBit/s per audio channel in standard settings. The total bandwidth requirement depends on the number

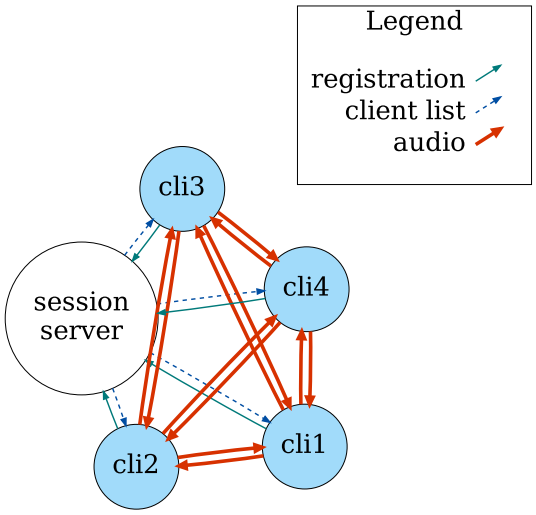

Figure 3. Example of an OVBOX session with four clients, all operating in peer-to-peer mode. Here the session server manages the peer-to-peer (P2P) architecture, but does not receive or send any client audio data. Each client is sending to every other client. In this example, each client is sending and receiving 2.7 MBit/s.

of clients in a session and the client configuration, see Figures 2 and 3 for examples; the downstream bandwidth requirement $B_{d,k}$, and upstream bandwidth requirement in peer-to-peer mode $B_{u,\mathrm{p2p},k}$ and in server mode $B_{u,\mathrm{srv},k}$ in MBit/s at the client k for a session with N clients, sending C channels with R bits per sample at a sampling rate f_s, and with a packaging overhead O (typically 0.2 at block sizes of 2 ms) is:

$$B_{d,k} = \sum_{i\neq k, i=1}^{N} R_i C_i f_{s,i}(1+O)\cdot 10^{-6} \quad (1)$$

$$B_{u,\mathrm{p2p},k} = (N-1) R_k C_k f_{s,k}(1+O)\cdot 10^{-6} \quad (2)$$

$$B_{u,\mathrm{srv},k} = R_k C_k f_{s,k}(1+O)\cdot 10^{-6} \quad (3)$$

Example bandwidth requirements for a number of typical settings are provided in Table 1.

The generic UDP forwarding tool also handles the traversal of incoming packets through network address translation (NAT) routers and firewalls. A typical "UDP hole punching" method is used to achieve this: Control packets are continuously sent from each client to the session server. This server determines the publicly visible IP address and outgoing port number, and shares this information with all other clients in the same session. Now all the other clients can send messages to the first client, which are treated by the NAT router as replies to the original message, and are therefore passed through.

A session is rendered as a virtual acoustic environment. In a session, each remote source and the local microphone signals are rendered as a virtual sound source. The local signals are mixed with the remote sources to provide acoustic feedback of the user's own voice or instrument.

Table 1. Example bandwidth requirements of each client for combinations of number of clients N, sample resolution R, and sampling frequency f_s. For simplicity, it is assumed that all clients use identical settings.

N	R Bits	f_s kHz	$B_{d,k}$ MBit/s	$B_{u,\mathrm{p2p},k}$ MBit/s	$B_{u,\mathrm{srv},k}$ MBit/s
3	16	48	1.84	1.84	0.92
5	16	48	3.68	3.68	0.92
7	16	48	5.53	5.53	0.92
3	16	32	1.23	1.23	0.61
5	16	32	2.46	2.46	0.61
7	16	32	3.68	3.68	0.61

By default, the sound sources are arranged in a circle with equal angular spacing. This arrangement of sources is surrounded by a shoebox room acoustic model simulated with a feedback delay network (FDN) based on [13], but operating in the first-order Ambisonics domain, with rotation operations at each reflection filter. The approach of using an FDN was chosen to minimise the computational cost, see [12] for discussion. FDN-generated reverberation often has a "metallic" character [14]. In the proposed implementation, the first order early reflections can be omitted to reduce this "metallic" character. Optionally, early reflections can be simulated using a geometric image source model at the cost of higher computational load. The use of this geometric image source model reduces the problem of front-back confusion and improves externalisation when combined with head tracking.

Several methods can be used to render the output of the spatial audio model to physical reproduction systems. Since in typical applications headphones are used for listening, the default method is to use a parametric binaural head model [15] for binaural synthesis. The advantage of the parametric model over HRIR convolution is a low computational cost compared to convolution with HRIRs. Alternatively, a simple physical model of an ORTF microphone technique, i.e. a pair of cardioid microphones with a distance of 17 cm and an opening angle of 110°, is available to generate stereo signals. A single channel omnidirectional renderer or a renderer for an ITU 5.1 playback system [16] can also be selected. Optionally, the output signal can be processed by an additional software component to provide hearing support [17].

3 Implementation

The OVBOX client software is implemented in C++. Some components, such as the jack audio connection kit, the zita-njbridge network audio client and the optional hearing support, are executed as separate processes. The other components of the client software are linked as libraries, either at compile time (e.g. room acoustic modelling) or as dynamic libraries (e.g. rendering methods or additional components such as DMX512 interfaces).

The session server is also implemented in C++ and shares some code with the client, which is linked as a library.

The configuration server is written in the scripting languages PHP and JavaScript. All components are publicly available as open source under the GNU General Public License (GPL) version 2 or later, see [18] for the source code. A publicly available deployment of the configuration server and a set of session servers is provided by the author [19].

Binary packages of the OVBOX client software are available for Ubuntu Linux (x86_64), Debian (32 and 64 bit arm, e.g. for RaspiOS) and for MacOS. An installer image for Raspberry Pis version 4B and 3B+ optimised for headless operation is provided.

4 Operating system optimisation

The primary requirement for real-time low-delay audio processing is the reliable processing of a short chunk of audio in a fixed amount of time that is less than the length of the audio chunk. Typical general purpose operating systems (GPOS) are not optimised for low-delay audio processing. A GPOS can be optimised for low-delay audio processing by, for example, disabling desktop features and unnecessary background tasks, and installing a low-latency optimised kernel. As these optimisations are difficult to achieve on a desktop system and for untrained users, the Raspberry Pi computer with a pre-configured installation image and disabled desktop functionality is an easy way to provide a delay-optimised system at low cost. The Raspberry Pi 4B's CPU power is sufficient for real-time low-delay signal processing, including virtual acoustics with up to 10 primary sound sources. Audio block sizes down to 0.25 ms can be achieved, which is not possible on most desktop systems. Another limitation of GPOS is that, on newer systems, the hardware's power-saving features, in particular the automatic switching of the CPU's clock frequency, often interfere with low-delay audio processing due to brief CPU stalls.

5 Delay performance

The primary objective of the OVBOX is to provide acoustic communication over network connections with low latency. The latency requirements depend on the application. In hearing research, the system is used to achieve telepresence. In typical network audio speech communication, delays of up to 200 ms can be accepted when using high quality audio coding as in the OVBOX [20]. However, to simulate face-to-face speech communication, much lower delays are desirable. For typical triadic communication at a table, the acoustic end-to-end delay is in the order of 4 ms (head-to-head distance about 1.4 m). In hearing aids, delays of less than 20 ms are acceptable [21], and commercial devices typically achieve delays of less than 10 ms. When used to simulate head-tracked binaural synthesis, the delay between head movement and corresponding acoustic changes should also not exceed 200 ms [22]. Altogether, this means that the delay requirements for typical research applications are in the range of 10–200 ms.

In music applications, the delay constraints are stricter than in the hearing research context of speech communication. Here, an acoustic end-to-end delay of more than 20 ms can be perceived, and delays of more than about 50 ms are not acceptable [5]. An increased end-to-end delay results in a continuous slowing of the tempo. A slower musical pulse results in greater robustness to delay. Delay is particularly critical in music with fast pulses and micro-rhythmic interactions. Not only the end-to-end delay, but also the self-monitoring delay can be critical: if the direct acoustic signal and the monitoring signal are added at similar amplitudes, the delay will result in comb filter artefacts, which can be particularly critical for singers. This effect can be avoided when adding only reverberation, but no direct path of the self-monitoring signal.

In the OVBOX system, the acoustic end-to-end delay is composed of several stages, see Table 2 for an overview. Some of these delays depend only on the spatial configuration and device settings, e.g. the distance between the sound source and the microphone, or the block size of the audio device. The delay due to network transmission cannot be controlled by the user. It depends mainly on the number of hubs to be traversed and, for longer distances, on the speed of light.

Jitter is the variation in the arrival time of network packets. Ideally, each network packet sent at one end will take a certain amount of time to reach the receiver. In practice, however, the travel time will sometimes be longer and sometimes shorter. To compensate for this effect, every network audio receiver needs some buffering to wait for all (or at least most) packets to arrive. The length of the jitter compensation buffers in the OVBOX system can be controlled by the user. The default settings are optimised for typical internet technologies. They can be adjusted to achieve a reasonable trade-off between low delay and packet loss. For example, when used in local network setups for monitoring, much lower values can be used.

A typical source of jitter is the link from the internet backbone to the user's router, often referred to as the "last mile", especially if this last mile is based on a network technology that relies heavily on error correction, such as Digital Subscriber Line (DSL) or mobile networks. The minimum network round trip time (RTT) and jitter, defined here as the difference between the 99% percentile of RTT and the minimum RTT, of some example network technologies are shown in Table 3. The OVBOX client software performs RTT measurements by sending a message containing the local time of sending and a serial number; these packets are immediately returned to the sending client by other clients and the session server, allowing the RTT to be determined from the difference between the time of reception and the time of sending. Data collection was performed on a Linux desktop PC connected to the Internet via a fibre optic (FO) connection. All devices tested were a Raspberry Pi 4B connected to the Internet via Internet Service Providers (ISPs) different from the ISP of the data collector. RTT measurements were taken every 50 ms for a total recording time of approximately 20 min for each technology, resulting in approximately 24,000 samples.

Table 2. Sources of delay in network audio communication, with typical values for Digital Subscriber Line (DSL) network, and ranges which can typically occur.

Delay source	Delay	Configuration	Range
Source to mic.	1 ms	0.3 m dist.	1–5 ms
A/D conv., USB	1.5 ms	USB sound card	0.5–2 ms
Block processing	2 ms	96 samples/block $f_s = 48$ kHz	0.25–5 ms
Network	10 ms	<1000 km	1–20 ms
Jitter comp.	10 ms	Typical for DSL	5–20 ms
Block processing	2 ms	See above	0.25–5 ms
D/A conv., USB	1.5 ms	USB sound card	0.5–2 ms
Speaker to ear	0.1 ms	Headphones	0.1 ms
Total delay	28.1 ms		10–50 ms

Table 3. Round trip times (RTT) and jitter of different network technologies. From the last-mile technologies, fibre optic internet (FO) performed best, followed by Digital Subscriber Line (DSL), cable internet (CI) and 4G Long Term Evolution (4G LTE). Peer-to-peer mode is not possible with 4G LTE. With local Gigabit Ethernet, minimum RTT of 0.1 ms and jitter of 0.2 ms can be reached (lower part). Delay performance of Wi-Fi depends on the distance. An increase of distance from approximately 4–20 m considerably increased the jitter.

Last mile technology	Peer-to-peer mode		Server mode	
	RTT_{min}	Jitter	RTT_{min}	Jitter
FO	12.6	1.0	13.5	1.1
DSL	18.4	3.0	29.7	3.1
CI	20.2	12.4	27.3	12.4
4G LTE	NaN	NaN	30.8	26.7
Local technology	RTT_{min}	Jitter		
Ethernet	0.1	0.2		
Wi-Fi, 4 m	0.8	2.7		
Wi-Fi, 20 m	1.0	317.9		

The data shows that FO connections perform best in terms of minimum RTT and jitter, followed by DSL, Cable Internet (CI) and 4G Long Term Evolution (4G LTE) mobile Internet. With 4G LTE, peer-to-peer data transmission is typically not possible.

Using server mode instead of peer-to-peer mode increases jitter only slightly, indicating that the last mile is the main source of jitter. The increase in minimum RTT when switching from peer-to-peer to server mode depends on the ISP. The RTT and jitter of the local Gigabit Ethernet and Wi-Fi 802.11g network technologies were also measured. The Wi-Fi connection was measured at a distance of approximately 4 m, separated from the access point by thin internal walls, and at a distance of approximately 20 m, shielded by concrete walls. Several competing networks on the same channel were active in the area. In all test conditions, the minimum RTT did not exceed 1 ms, but the jitter increased considerably with increasing distance in the Wi-Fi connection.

6 Computational performance

The performance requirements for rendering a session with five devices were analysed using various combinations of hardware and software configurations, see Table 4 for details. CPU usage was reported by the jack audio connection kit. It is the time taken to process one audio block divided by the duration of this block. The number of underruns and overruns of the audio interface was counted over a measurement period of approximately 900 s. In addition, the acoustic RTT of the monitoring, i.e. the time from an analogue electrical input signal to the local analogue audio playback signal without network transmission, was measured. Measurements were taken for block sizes of 2 ms and 1 ms respectively. The devices tested were a Raspberry Pi model 3B+, a Raspberry Pi model 4B, a desktop PC with an Intel CPU and a Mac mini with an Apple M1 CPU; a fifth device was part of the session but not analysed. The Raspberry Pi operating system was optimised for audio processing using the methods described in Section 4, based on Raspbian 10 "buster" with a 32-bit kernel version 5.10. The Linux desktop PC was manually tuned for low-latency audio performance using a 5.15 low-latency kernel and real-time scheduling of system interrupts. The Mac Mini ran a standard desktop system (12.7.3 "monterey") as supplied by the manufacturer, with no additional optimisations.

The best CPU performance was achieved by the Mac Mini (4.6% for 2 ms, 5.7% for 1 ms block size), followed by the desktop PC (8.2% for 2 ms, 15% for 1 ms) and the Raspberry Pis. The Raspberry Pi 3B+ was only able to render a session with five devices at 2 ms block sizes. At 2 ms block sizes, all systems except the Raspberry Pi 3B+ reported zero or one xrun throughout the measurement period. The Raspberry Pi 3B+ reported 80 xruns, corresponding to an xrun rate of 5.3 per minute. The best monitoring latency was achieved with the Linux desktop PC: 6.04 ms electrical input to electrical output was achieved with a block size of 2 ms and 3.9 ms monitoring latency with a block size of 1 ms. All monitoring latencies were less than 10 ms for the 2 ms block size and 7 ms or less for the 1 ms block size.

The standard acoustic model was used in this measurement, without simulation of early reflections. A detailed

Table 4. Analysis of computational performance and local self-monitoring latency for different operating systems and configurations. All devices were running in the same session, connected to the same local network, using ovbox version 0.21.2, with a Focusrite Scarlett solo USB audio interface and one audio input channel. The column "CPU" denotes the CPU usage as reported by the audio server, which is the average time used to process one audio block divided by the duration of one audio block. Overruns and underruns of the audio device ("xruns") were counted over a period of 900 s. The monitoring round-trip time ("RTT") was measured from analogue input to analogue output.

Hardware	Operating system	2 ms block size			1 ms block size		
		CPU	xruns	mon. RTT	CPU	xruns	mon. RTT
		%	1/min	ms	%	1/min	ms
Raspberry Pi 3B+ (BCM2837B0)	Raspbian 10 (32 Bit) 5.10.103-v7l+	62	5.3	9.25	100	>9000	5.4
Raspberry Pi 4B (BCM2711)	Raspbian 10 (32 Bit) 5.10.103-v7l+	24	<0.05	9.125	34	1	5.125
Linux desktop (Intel i5-8500)	Ubuntu 20.04 5.15.0-97-lowlatency	8.2	0	6.04	15	0	3.9
mac mini M1 2020 (Apple M1)	macOS Monterey 12.7.3	4.6	<0.05	9.04	5.7	<0.05	7.04

evaluation of the computational performance of the underlying room acoustic modelling toolbox TASCAR can be found in [12].

7 Applications

The primary application of the OVBOX is remote musical collaboration, e.g. for distributed rehearsals, but also to perform distributed concerts, either as a live stream or as an extended concert where one or more musicians are represented by speakers that reproduce the streamed audio signal. For rehearsals, a metronome with delay compensation is available, as well as a browser-controllable multichannel audio recorder and mixer. Extensions allow not only audio but also DMX512 to be distributed for synchronised control of stage effects such as lighting for distributed concert performances. When used in a local network, it can be used as an in-ear monitoring system, with the option of using head trackers to create an individual head-tracked binaural synthesis with physical modelling of the acoustic space.

An example of a research application of the OVBOX is shown in Figure 4. In general, interactive communication plays an increasingly important role in hearing research [9, 23, 24], especially when hearing devices are involved that interact with the user's behaviour [25–28]. The use of telepresence tools has allowed paradigms based on interactive communication to continue during the pandemic [24], but more importantly, when used in combination with virtual reality, it can provide access to all signals separately, allowing the highest level of experimental control while maintaining free conversation.

Depending on the application, the requirements for telepresence tools may differ. However, when comparing music and hearing research applications, there is a large overlap in requirements. Both applications require high audio quality of the transmitted signals, and benefit from separate transmission of each audio channel, either to provide access to direct signals in research applications, or to allow acoustically transparent spatial separation of sources with individual spatial representations and mix settings. Head tracking of all session participants can be important in music applications for dynamic binaural synthesis with simulation of source and receiver movement. In hearing

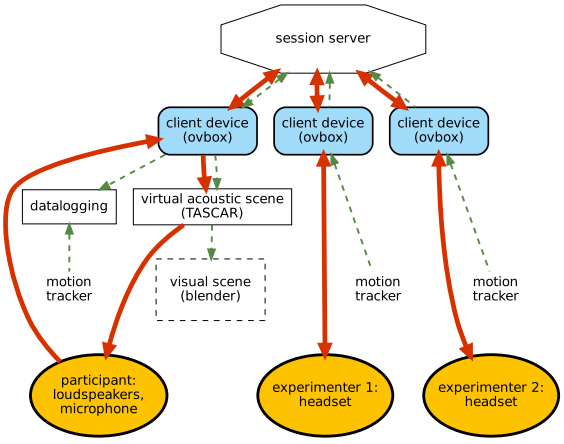

Figure 4. Example application of the OVBOX system in a hearing research application based on [9]. The OVBOX is used to transmit movement and gaze data recorded by head-mounted sensors, as well as audio, to a central laboratory for real-time rendering of audiovisual virtual environments and data logging.

research applications, head tracking can be used for analysing the movement behaviour of all participants. The low-latency transmission of arbitrary data has applications in both music and hearing research: in music applications to control stage effects, e.g. to transmit lighting settings, while in research applications it is used for experimental control and simultaneous recording of various biophysical sensor signals, e.g. gaze direction with electro-oculography. When used as a desktop client, all source signals can be accessed separately via the jack audio connection kit. Again, this can be used in either application: for music, to route signals to digital audio workstations, for example to create a dedicated mix for streaming, or for research, to integrate with other laboratory applications.

The configuration server user interface has two levels: For typical music applications, users have access to the most commonly controlled parameters, such as gains, jitter buffer settings, and selection of the audio device and its parameters. For more complex applications, such as research or specific music applications, access to almost all parameters is provided.

For reproducible open science it is important that the tools used in the experimental setup are also published as open source [29]. This is the case for the OVBOX, where all components, from the signal processing in the acoustic simulation to the web interface for configuration, are published with open access under open source licences.

8 Conclusions

The presented tool for network-based low-delay audio communication, the OVBOX system, is used for remote music rehearsals and distributed concerts, but can also facilitate controlled experimental paradigms with interactive communication for applications in hearing research. Using low-delay signal processing for interactive acoustic simulation, very low end-to-end delays can be achieved as long as the underlying Internet connection provides low delay and jitter. The generic UDP forwarding tool allows flexible low-delay transmission of arbitrary additional data. The OVBOX system is publicly available as open source software with pre-compiled packages, making it a useful tool for both music applications and hearing research.

Acknowledgments

Thanks to the members of the ensemble ORLANDOviols for early testing, comments and use of the OVBOX. Thanks to Angelika Kothe, Volker Hohmann, Júlia Vető and two anonymous reviewers for their helpful comments on the manuscript.

Funding

Funded by the Deutsche Forschungsgemeinschaft (DFG) – Project-ID 352015383 – SFB 1330 B1.

Conflicts of Interest

The author declares that he has no conflicts of interest in relation to this article.

Data availability statement

The research data associated with this article are available in zenodo, under the reference [30].

References

1. V. Fischer: Jamulus. Available at https://jamulus.io/ (accessed 2023).
2. A. Carôt: Soundjack. Available at https://www.soundjack.eu/ (accessed 2023).
3. Jamkazam. Available at https://jamkazam.com/ (accessed 2023).
4. C. Chafe: Jacktrip. Available at https://www.jacktrip.com/ (accessed 2023).
5. C. Rottondi, M. Buccoli, M. Zanoni: Feature-based analysis of the effects of packet delay on networked musical interactions, Journal of the Audio Engineering Society 63, 11 (2015) 864–875.
6. C. Rottondi, C. Chafe, C. Allocchio, A. Sarti: An overview on networked music performance technologies, IEEE Access 4 (2016) 8823–8843.
7. T. Mace: Musick's Monument, John Carr, 1676.
8. S.M. Boker, J.F. Cohn, B.-J. Theobald, I. Matthews, T.R. Brick, J.R. Spies: Effects of damping head movement and facial expression in dyadic conversation using real-time facial expression tracking and synthesized avatars, Philosophical Transactions of the Royal Society B: Biological Sciences 364 (2009) 3485–3495.
9. G. Grimm, H. Kayser, A. Kothe, V. Hohmann: Evaluation of behavior-controlled hearing devices in the lab using interactive turn-taking conversations, in 10th Convention of the European Acoustics Association, Turin, Italy, 11–15 September, 2023.
10. F. Adriaensen: Zita-njbridge. 2023. Available at: https://kokkinizita.linuxaudio.org/linuxaudio/.
11. F. Adriaensen: Controlling adaptive resampling, in Linux Audio Conference, Stanford, USA, 2012.
12. G. Grimm, J. Luberadzka, V. Hohmann: A toolbox for rendering virtual acoustic environments in the context of audiology, *Acta Acustica United with Acustica* 105 (2019) 566–578.
13. D. Rocchesso, J.O. Smith: Circulant and elliptic feedback delay networks for artificial reverberation, *IEEE Transactions on Speech and Audio Processing* 5 (1997) 51–63.
14. J.-M. Jot, A. Chaigne: Digital delay networks for designing artificial reverberators, in 90th Audio Engineering Society Convention, Paris, France, 19–22 February, 1991.
15. F. Schwark, M.R. Schädler, G. Grimm: Data-driven optimization of parametric filters for simulating head-related transfer functions in real-time rendering systems, in EUROREGIO BNAM2022 Joint Acoustics Conference, Aalborg, Denmark, 2022, pp. 1–10.
16. B.S. Series: Report itu-r bs.2159-7: Multichannel sound technology in homeand broadcasting applications. Technical report, International Telecommunication Union, 2015.
17. H. Kayser, T. Herzke, P. Maanen, M. Zimmermann, G. Grimm, V. Hohmann: Open community platform for hearing aid algorithm research: open Master Hearing Aid (openMHA), *SoftwareX* 17 (2022) 100953.
18. G. Grimm: ovbox client software (ov-client), 2023. Available at: https://github.com/gisogrimm/ov-client.
19. G. Grimm: ORLANDOviols Consort box (ovbox), 2021. Available at: https://ovbox.de/.
20. S. Na, S. Yoo: Allowable Propagation Delay for VoIP Calls of Acceptable Quality, Springer, Berlin Heidelberg, 2002, pp. 47–55.
21. M.A. Stone, B.C.J. Moore: Tolerable hearing aid delays. I. Estimation of limits imposed by the auditory path alone using simulated hearing losses, Ear and Hearing 20, 3 (1999) 182–192.
22. C. Rappin, J. Palacino, P. Rueff, L. Feichter, M. Paquier: Latency detection threshold of head-tracking for different head rotation speeds in binaural rendering, in 10th Convention of the European Acoustics Association, Turin, Italy, 11–15 September, 2023.
23. L.V. Hadley, W.M. Whitmer, W. Owen Brimijoin, G.M. Naylor: Conversation in small groups: Speaking and listening strategies depend on the complexities of the environment and group, Psychonomic Bulletin & Review 28 (2021) 632–640.
24. M. Hartwig, V. Hohmann, G. Grimm: Speaking with avatars-influence of social interaction on movement behavior in interactive hearing experiments, in IEEE VR 2021 Workshop: Sonic interactions in Virtual Environments (SIVE), IEEE, 2021, pp. 94–98.
25. V. Best, E.J. Ozmeral, B.G. Shinn-Cunningham: Visually-guided attention enhances target identification in a complex auditory scene, *Journal for the Association for Research Otolaryngology* 8 (2007) 294–304.
26. G. Kidd Jr., S. Favrot, J.G. Desloge, T.M. Streeter, C.R. Mason: Design and preliminary testing of a visually guided hearing aid, Journal of the Acoustical Society of America 133 (2013) EL202–EL207.
27. G. Grimm, H. Kayser, M.M.E. Hendrikse, V. Hohmann: A Gaze-based Attention Model for Spatially-Aware Hearing Aids, VDE Verlag GmbH, Berlin, Offenbach, 2018, pp. 231–235.
28. A. Favre-Felix, C. Graversen, R.K. Hietkamp, T. Dau, T. Lunner: Improving speech intelligibility by hearing aid eye-gaze steering: Conditions with head fixated in a multitalker environment, Trends in Hearing 22 (2018) 1–11.
29. S. Spors, M. Geier, H. Wierstorf: Towards open science in acoustics: Foundations and best practices, Tagungsband der DAGA 17 (2017) 218–221.
30. G. Grimm: Round-trip time distribution for network based low-delay audio communication, 2024. Available at https://doi.org/10.5281/zenodo.10491647.

Cite this article as: Grimm G. 2024. Interactive low delay music and speech communication via network connections (OVBOX). Acta Acustica, **8**, 18.