Aretousa: A competitive audio streaming software for Network Music Performance

Konstantinos Tsioutas*, George Xylomenos* and Ioannis Doumanis†
*Athens University of Economics and Business, Department of Informatics, Greece
†University of Central Lancashire, School of Physical Science and Computing, United Kingdom

Abstract—**Many existing open source systems provide support for Network Music Performance (NMP), with each one catering to a specific system and usage scenario. As our research in evaluating the Quality of Experience (QoE) of NMP systems as perceived by musicians involves widely different scenarios and requires extensive instrumentation of the platform, we built a new NMP system, Aretousa. Our system offers a large number of configuration and monitoring options, without sacrificing latency, the most critical factor for NMP. To show that Aretousa provides flexibility while being competitive with the state of the art in terms of latency, we present measurements comparing it against JackTrip in multiple setups over a high speed research network.**

I. INTRODUCTION

Fast and reliable Internet services are being used on an everyday basis for commercial an recreational purposes. Services like teleconferencing and tele-presence provide the ability for companies and individuals to communicate in real time via fixed and mobile networks, offering audio, video and collaboration facilities. Proprietary systems such as Skype, Vidyo and WebEx provide professional solutions for high quality telepresence and teleconferencing. The WebRTC[1] framework is used in many open source platforms, such as BigBlueButton[2] which is widely used in e-learning environments. The *Quality of Service* (QoS) of such real time services over the Internet has improved so much over the past few years, that even commercial versions of some proprietary systems exist.

Network Music Performance (NMP), where two or more musicians perform music through their Internet connections, ideally as if they were placed in the same room, is an "extreme" case of teleconferencing. The critical differentiation of NMP from teleconferencing is the need for very low audio latency between the musicians, which places strict limits on the underlying network and coding latencies. The *Mouth to Ear* (M2E) delay, that is, the delay between a microphone at one end and a speaker at the other end, must be kept below 24 ms [1][2][3] for musicians to be able to synchronize. In contrast, teleconferencing works well even with delays of over 100 ms.

As a result, NMP is not currently feasible for plain Internet users located behind residential (ADSL) links, requiring instead the very fast connections of high speed research networks. Even in such networks, keeping latency low requires optimizing the network, avoiding servers and using low delay codecs or, even, no coding at all. Of course, assessing the suitability of a specific setup for NMP goes beyond QoS metrics such as latency: we actually need to measure the *Quality of Experience* (QoE) of the participating musicians, which depends on many additional factors.

In order to measure QoE with different network setups, codec configurations, collaboration setups and musical styles, we need a flexible NMP platform that can be easily instrumented for data gathering and adapted to different network and collaboration configurations. Aretousa is a simple but flexible new NMP system, built on top of open source frameworks for portability and maintanability. Unlike previous NMP systems that can operate only in specific setups (e.g., two directly connected musicians for JackTrip), Aretousa supports a wide range of setups and options. Using test measurements in a high speed research network, we show that its latency is competitive with JackTrip, the most responsive open source NMP system.

This paper is organized as follows. In Section II we present related work and existing NMP frameworks. The Aretousa software is described in Section III, while the experiments conducted with Aretousa are described in Section IV and the results are provided in Section V. We summarize our work in Section VI.

II. RELATED WORK ON NMP

Interactive networked music performance begins with John Gage, who designed the *Imaginary Landascape No. 4* performance in 1951, where twelve users on a stage tuned their portable radio receivers to random radio stations and changed the volume levels in a random way. Although audio conferencing has been used on the Internet since the mid 1990s, NMP became practical, even in research networks, only in the 2000s [4]. The SoundWire group at Stanford University has conducted extended research on the field leading to JackTrip [5][6], an open source service for real time uncompressed point-to-point audio streaming; JackTrip avoids servers and coders/decoders to keep delay low. Other frameworks include DIAMOUSES [6], LOLA [7] and MusiNET [8]. In MusiNet our laboratory, the *Mobile Multimedia Laboratory* (MMLAB)[3] explored the use of multicast [9], ultra-low delay servers [10] and multipath routing [11] to reduce the delay in NMP sessions with more than two endpoints.

[1]https://webrtc.org/
[2]https://bigbluebutton.org/
[3]http://mm.aueb.gr

Some experiments have compared server-based (many clients connected via a server) and peer-to-peer (two or more clients directly communicating with each other) topologies in terms of latency and QoS; using a server simplifies sessions with three or more participants, as each endpoint needs to communicate only with the server, but introduces delays on the network path [9]. Other experiments compared uncompressed audio to audio compressed with low delay audio codecs such as Opus; common music codecs such as MP3 introduce delays too high to be useful for NMP. To evaluate M2E delay, some experiments used claps[2], while in others musicians played actual instruments. Delays are easier to measure with simple sounds, which can even be pre-recorded, obviating the need for musicians. In most cases, very fast LANs or research WANs were used and the focus was on QoS metrics like delay.

Although QoS is important, it cannot accurately predict the QoE of musicians, or the effects that NMP has on musician behavior. Research on how network latency affects musicians has concluded that as the latency increases, musicians slow down their tempo [1][3]. Chafe [2] reached the same conclusion experimenting with musicians who clapped their hands, adding that when latency is below 11 ms, musicians accelerate their tempo. Olmos [12] experimented with two opera singers and a conductor over a network, evaluating two bio-metric measures, the Galvanic Skin Response (GSR) and the number of Skin Conductance Responses (SCR), using software for behavior recording along with questionnaires. These tests can reveal far more about QoE in NMP than simple QoS measurements, and they are the focus of our current research.

III. The Aretousa software

Among the frameworks used for NMP, JackTrip is the most common choice for end users. It is free to download and simple to use, but it only supports direct connections between two musicians. JackTrip uses the Linux ALSA drivers and provides ultra low latency with (uncompressed) PCM audio. The user can listen to his own sound, along with the sound from his peer, and configure the audio buffer size and a few other driver properties. Most importantly, JackTrip is fast, as it was built to reduce latency as far as possible.

While JackTrip is an excellent NMP system, it aims for simplicity and speed in operation. Our desire to evaluate the musicians' QoE during NMP as a function of several parameters, including the audio format, network topology, audio buffer size, Ethernet packet size, audio compression parameters, requires a far more flexible system, that could sacrifice simplicity, but should strive to keep latency low. This is the motivation for our new NMP system, Aretousa.

To avoid low level coding that would make the prototype difficult to implement and maintain, Aretousa is based on the GStreamer[4] and GTK[5] open frameworks. It supports the initialization, configuration, control and mix of multiple outgoing and incoming audio streams. GStreamer provides tools to build audio pipelines that capture audio from the sound card's input, make all the necessary transformations to the audio format, segment audio to packets, add necessary protocol headers like RTP and UDP and, finally, send it to the network using the `udpsink` plugin. At the other end, the `udpsrc` plugin, receives incoming UDP packets at a UDP port chosen by the user, strips the protocol headers and sends audio samples to the sound card's output. These pipelines are constructed by plugins connected serially via sources and sinks. The element that captures audio is `pulsesrc`, while the element that plays out the audio is `pulsesink`; these use the Pulse audio API which in turns uses the ALSA drivers to capture audio. GTK is used for the *User Interface* (UI).

[4]https://gstreamer.freedesktop.org/

[5]https://www.gtk.org/

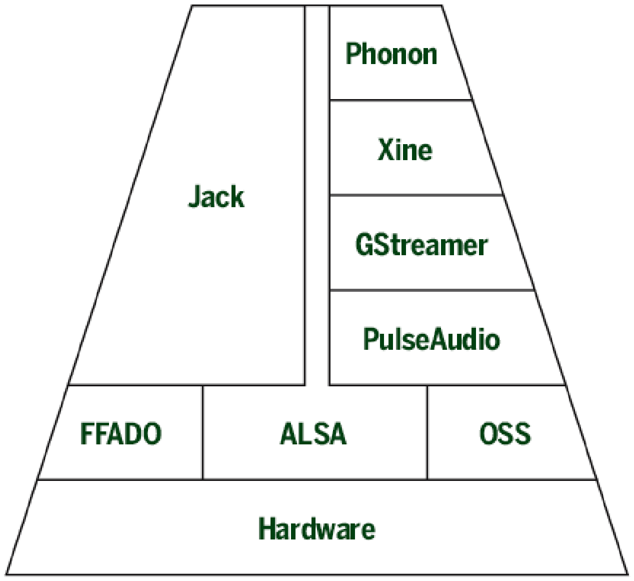
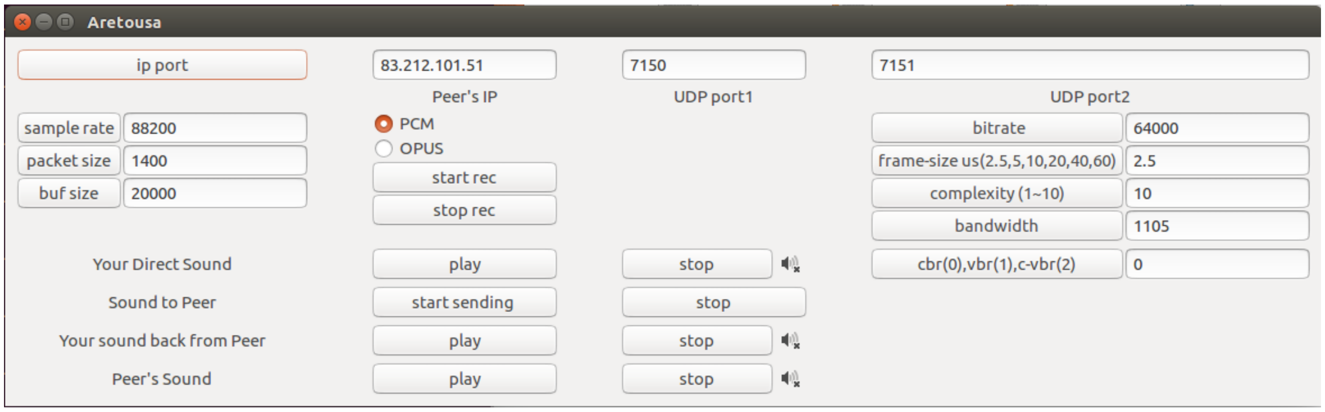


Fig. 1: Gstreamer and JackTrip API's stack.

As shown in Figure 1, while JackTrip uses directly the ALSA layer, the GStreamer framework sits on top of PulseAudio and ALSA. One can construct a simple pipeline using GStreamer command line tools to listen to his own audio, for example, `gst-launch-1.0 pulsesrc ! pulsesink`. Using this script, the user will experience an audio delay of over 200 ms. Each of these plugins is followed by parameters which can be configured for experimental purposes. By constructing a pipeline which streams audio to a certain host, we can implement a topology to evaluate latency for the path that the stream will follow.

Aretousa supports both peer-to-peer and server-based architectures, as well as both uncompressed PCM audio and the Opus codec for audio compression. Furthermore, the user can configure parameters such as the IP address of the NMP server in server-based mode, or the other peer's IP address in peer-to-peer mode, as well as the necessary UDP ports, as shown in Figure 2. Aretousa also allows configuring the audio buffer size, the Ethernet packet size, the Opus bit-rate and the Opus audio bandwidth, among other settings. The musician can listen to his own sound directly, his sound coming back from the peer (echo) and/or the peer's sound, controlling the sound level for stean via volume sliders.

Aretousa provides the option of recording incoming and


Aretousa

ip port: 83.212.101.51 (Peer's IP), 7150 (UDP port1), 7151 (UDP port2)

sample rate 88200
packet size 1400
buf size 20000

[✓ PCM] OPUS

bitrate 64000
frame-size us(2.5,5,10,20,40,60) 2.5
complexity (1~10) 10
bandwidth 1105
cbr(0),vbr(1),c-vbr(2) 0

Fig. 2: Aretousa's client UI.

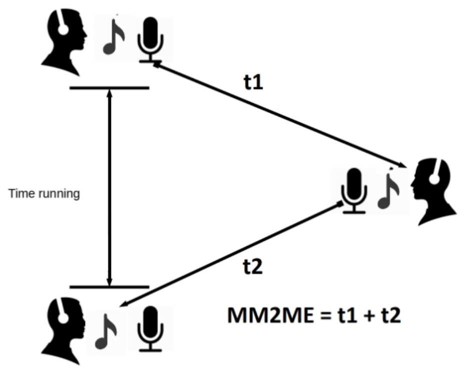


Fig. 3: My Mouth to My Ear delay.

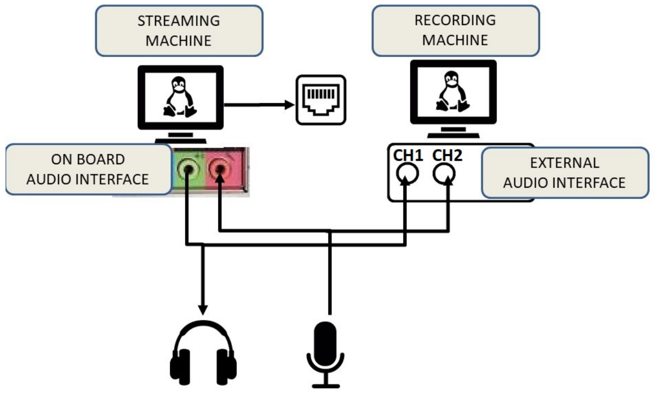


Fig. 4: NMP testing configuration.

outgoing streams to separate wave files, allowing the user to mix and/or analyze them offline, using audio processing software. Although Aretousa can be used by ordinary musicians, its target group is experimenters wanting to test different parameters and setups. To test server-based configurations, we have built a simple GStreamer-based server that operates as a *Selective Forwarding Unit* (SFU), that is, it selectively forwards packets, but it does not decode and re-encode them, to avoid adding delay. An SFU is very useful when more than two musicians want to collaborate, since it can replicate a single stream coming from each musician to all other participants in the session [10].

IV. Test Configuration

To illustrate the capabilities of Aretousa and demonstrate that, despite its flexibility, it is competitive with JackTrip, we tested both systems and measured their round-trip audio delay, which we define as *My Mouth to My Ear* (MM2ME) delay. As shown in Figure 3, MM2ME is a more appropriate metric than M2E for NMP, as when musicians play together, each musician plays one note and unconsciously expects to listen to the other musicians' note to play his next one and so on. In addition, measuring MM2ME delay accurately is much easier than measuring the M2E delay, as it can be done at one endpoint, by simply reflecting the transmitted sound at the other endpoint; M2E needs to be measured at both endpoints, thus requiring perfectly synchronized clocks. During these tests, we also evaluated the degree of synchronization musicians could achieve with various configuration parameters.

Network delay can be measured by using the RTP time stamps in the packets. The M2E delay also includes the time needed to capture, encode (optionally), packetize and then depacketize, decode (optionally) and playout samples; MM2ME requires repeating this process for each direction of communication. To assess such delays, we can send an audio stream to the peer, get it back, and compare the outgoing and incoming streams.

We implemented sessions using both Aretousa and JackTrip with PCM audio. With Aretousa, we also tested compressed audio using the Opus codec, with varying parameters. We experimented with various Ethernet packet sizes and audio buffer sizes. We used the GRNET[6] infrastructure, to which MMLAB is connected through fiber optic links. The computers used for the experiments ran Ubuntu 16.04 with i7 processors and 12 GB of RAM; we used the onboard sound card of each machine for audio capture and playback.

As shown in Figure 4, at each endpoint the computer used for streaming was complemented by a separate computer for recording, which used an external audio interface, to avoid

[6]https://grnet.gr/en/

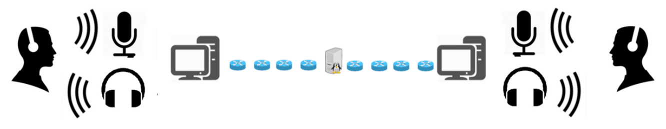


Fig. 5: Topology through NMP server.

delaying the audio capture and playout operations due to recording. A little mixing console with an auxiliary output, a condenser microphone and closed type headphones were used by each of the two musicians participating. The microphone, which captured physical audio, was routed to the audio input of the streaming machine for transmission using the Aretousa software. In parallel, it was routed via the console to the recording computer, using channel 1 of the external audio interface. The audio output of the streaming machine was directed to the musician's headphone, and was also routed to the recording computer, where it was recorded using channel 2 of the external audio interface. As a result, the recording combined what the musician produced (channel 1) and what the musician heard (channel2).

As mentioned above, using Aretousa, the user can monitor or mute his own sound, the other peer's sound and his audio echoed back from the peer. By monitoring his own sound, the user experiences the audio delay introduced by the streaming machine's audio buffers used to capture and playback sound. By monitoring his audio echoed back from the other peer, he also experiences the delays due to packetization, network transmission and reception, depacketization at the peer, and then repacketization, transmission, reception and depacketization at his end.

In our first scenario, we played back the captured sound directly, so as to assess the delays due to audio buffering: the audio was captured and then played out directly in the same machine. In our second scenario, we connected the two peers directly: the audio stream was sent from one client to the other, played out, captured again, and reflected back. This scenario captures the full MM2ME delay. The third scenario was the same as the second, but using an NMP server between the endpoints, as shown in Figure 5; the server did not perform any processing. This scenario was only tested with Aretousa, as JackTrip does not support NMP servers.

To calculate MM2ME delay, we used hand claps, as they are easy to spot in audio processing programs: we simply needed to measure the distance between the peaks (the claps) among the two channels representing the sound sent and received. We also asked musicians to perform using the setup of Figure 5, but without reflecting the audio of each musician. After each session, they were asked to answer a survey with questions about sound quality, clicks, audio interrupts, delay perceived, ability to synchronize, ability to express feelings and the overall procedure quality.

V. Results and Discussion

Figure 6 shows the MM2ME delay measured in the first scenario (direct audio in a single machine) as the audio buffer capture size grows from 4 ms to 20 ms worth of audio, with uncompressed sound (PCM); orange boxes represent the mean/min/max and variance with Aretousa and green boxes the same metrics for JackTrip. Since in this scenario audio is captured and played out once, these numbers are the minimum possible one way delays due to the audio system (without networking). It is clear that beyond 10 ms of audio buffering, it is impossible to achieve the latencies required for NMP (24 ms). The delays between JackTrip and Aretousa are nearly the same in all aspects.

Figure 5 shows the MM2ME delay measured in the second scenario (direct connection between two peers); this is the real-world MM2ME delay in a configuration without servers. If we consider as an upper limit of MM2ME twice the limit for M2E, that is, 2x24 ms, the audio capure buffer needs to be kept to no more than 10 ms. Again, Aretousa and JackTrip are virtually the same in each case.

Figure 8 shows the MM2ME delay measured in the third scenario (two peers with a server in between), as the capture buffer size grows. This time, we only use Aretousa, but in addition to uncompressed audio (PCM), we show the MM2ME delay when using the Opus codec with a frame size of 2 us and 20 us. We also show for reference the network delay, as measured by the RTP timestamps; anything above this line, is due to the audio system. Assuming again that the MM2ME limit is around 2x24 ms, PCM audio can handle up to 10 ms of audio buffering, while Opus cannot handle more than 5 ms, due to its additional coding and decoding delay, which is 20 ms or more. We can also note that the server has added around 7 ms of delay, by comparing the PCM delays at 10 ms of audio buffering between Figure 7 and Figure 8.

The MM2ME delay calculations show that over a high speed research network, such as GRNET, and with a server acting as a simple forwarder for each stream, the biggest percentage of the audio delay comes from the audio system, which defines the total MM2ME delay. The network delay when the server was included was very low, around 7 ms as evaluated with Wireshark. Aretousa did not add any perceivable delay when using uncompressed PCM audio, but coding with Opus added at least 20 ms of delay, requiring a reduced audio buffer size to make NMP possible.

VI. Summary

We have created a new software platform for NMP, Aretousa, which offers numerous options for experimenters desiring to assess either the QoS or the QoE of NMP sessions in widely different setups. To show that Aretousa is competitive with other NMP options in terms of latency, the main limiting factor for NMP, we compared it with JackTrip and found that it provides virtually identical M2E and MM2ME delays. In addition, we provided results with an intervening NMP server and with Opus encoded audio, to showcase the flexibility of Aretousa for testing. We are planning to use Aretousa for a comprehensive assessment of QoE with actual musicians, by designing and carrying out a proper experimental study, that will encompass network metrics, QoE questionnaires and biometric data from electroencephalography (EEG) headsets, to reveal the full range or human responses to NMP.

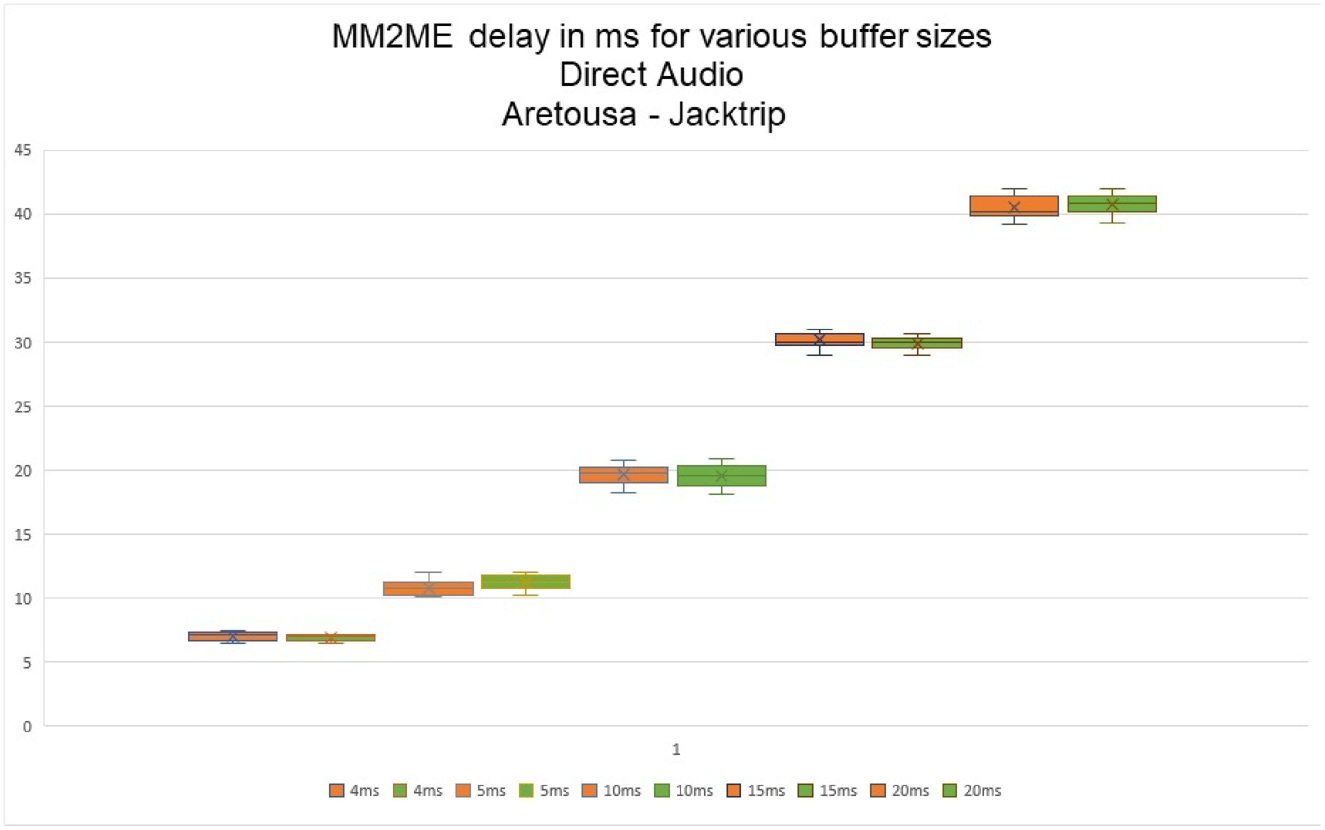


Fig. 6: MM2ME delay in ms, direct audio, using Aretousa (orange) and JackTrip (green) for various buffer sizes.

VII. Acknowledgments

We would like to thank the musicians Nikos Sariyannis and Christos Panagiotakopoulos for their patience during the experiments.

References

[1] C. Bartlette, D. Headlam, M. Bocko, and G. Velikic, "Effect of network latency on interactive musical performance," *Music Perception: An Interdisciplinary Journal*, vol. 24, no. 1, pp. 49–62, 2006. [Online]. Available: http://www.jstor.org/stable/10.1525/mp.2006.24.1.49

[2] C. Chafe and M. Gurevich, "Network time delay and ensemble accuracy: Effects of latency, asymmetry," in *Audio Engineering Society Convention 117*, Oct 2004. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=12865

[3] P. F. Driessen, T. E. Darcie, and B. Pillay, "The effects of network delay on tempo in musical performance," *Computer Music Journal*, vol. 35, no. 1, pp. 76–89, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1162/COMJ_a_00041

[4] A. Barbosa, "Displaced soundscapes: A survey of network systems for music and sonic art creation," *Leonardo Music Journal*, vol. 13, pp. 53–59, 2003.

[5] J.-P. Cáceres and C. Chafe, "JackTrip: Under the hood of an engine for network audio," in *Proceedings of International Computer Music Conference*, 2009, p. 509–512.

[6] C. Alexandraki and D. Akoumianakis, "Exploring new perspectives in network music performance: The diamouses framework," *Computer Music Journal*, vol. 34, no. 2, pp. 66–83, 2010. [Online]. Available: https://doi.org/10.1162/comj.2010.34.2.66

[7] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via lola: Low latency high quality a/v streaming system," in *Information Technologies for Performing Arts, Media Access, and Entertainment*, P. Nesi and R. Santucci, Eds. Berlin, Heidelberg: Springer, 2013, pp. 240–250.

[8] D. Akoumianakis, C. Alexandraki, V. Alexiou, C. Anagnostopoulou, A. Eleftheriadis, V. Lalioti, Y. Mastorakis, A. Modas, A. Mouchtaris, D. Pavlidi, G. C. Polyzos, P. Tsakalides, G. Xylomenos, and P. Zervas, "The musinet project: Addressing the challenges in networked music performance systems," in *International Conference on Information, Intelligence, Systems and Applications (IISA)*, July 2015, pp. 1–6.

[9] C. Stais, Y. Thomas, G. Xylomenos, and C. Tsilopoulos, "Networked music performance over information-centric networks," in *IEEE International Conference on Communications Workshops (ICC)*, June 2013, pp. 647–651.

[10] G. Baltas and G. Xylomenos, "Evaluating the impact of network i/o on ultra-low delay packet switching," in *IEEE Symposium on Computers and Communication (ISCC)*, July 2015, pp. 397–402.

[11] Y. Thomas, G. Xylomenos, and G. C. Polyzos, "Exploiting path diversity for networked music performance in the publish subscribe internet," in *International Conference on Information, Intelligence, Systems and Applications (IISA)*, July 2015, pp. 1–6.

[12] A. Olmos, M. Brulé, N. Bouillot, M. Benovoy, J. Blum, H. Sun, N. W. Lund, and J. R. Cooperstock, "Exploring the role of latency and orchestra placement on the networked performance of a distributed opera," in *12th Annual International Workshop on Presence*, 2009.

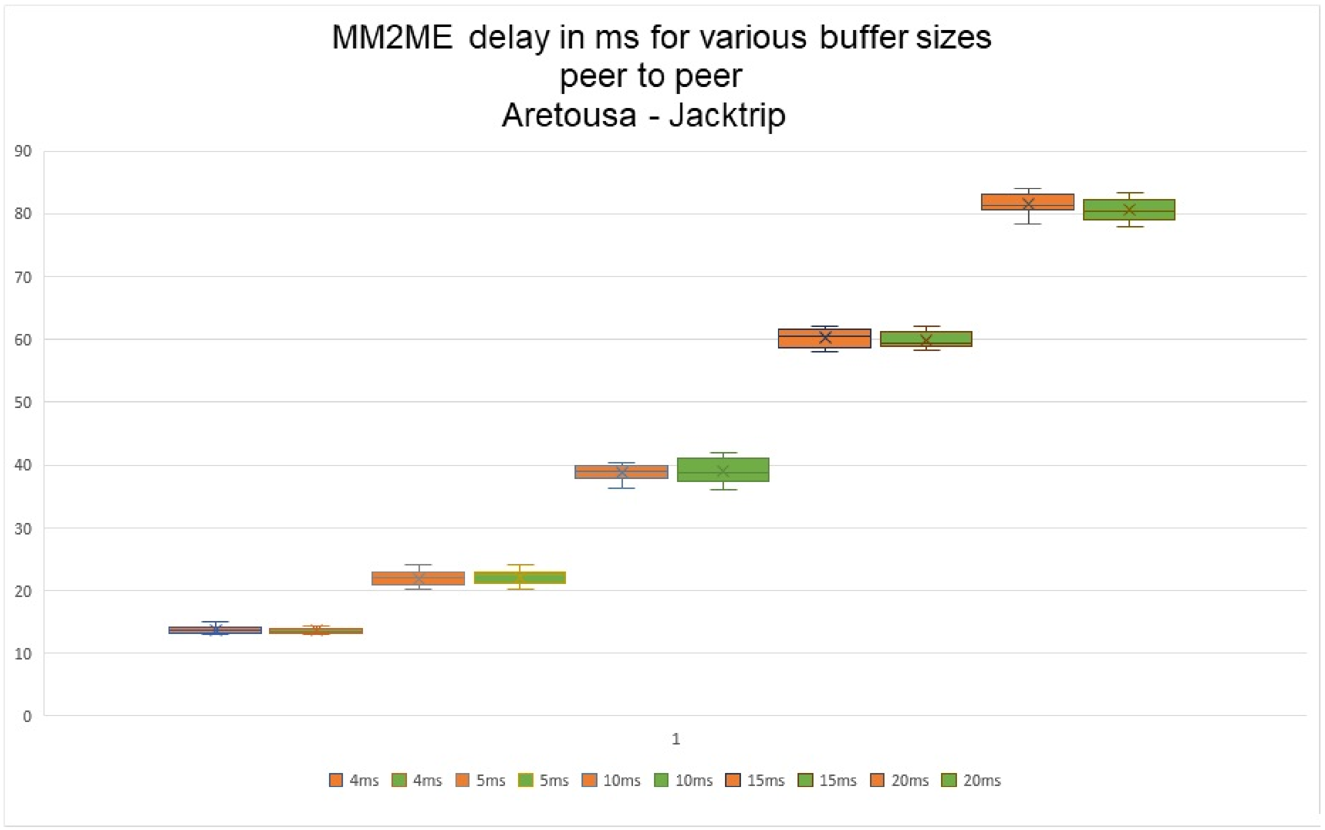


Fig. 7: MM2ME delay in ms, peer to peer, using Aretousa (orange) and JackTrip (green) for various buffer sizes.

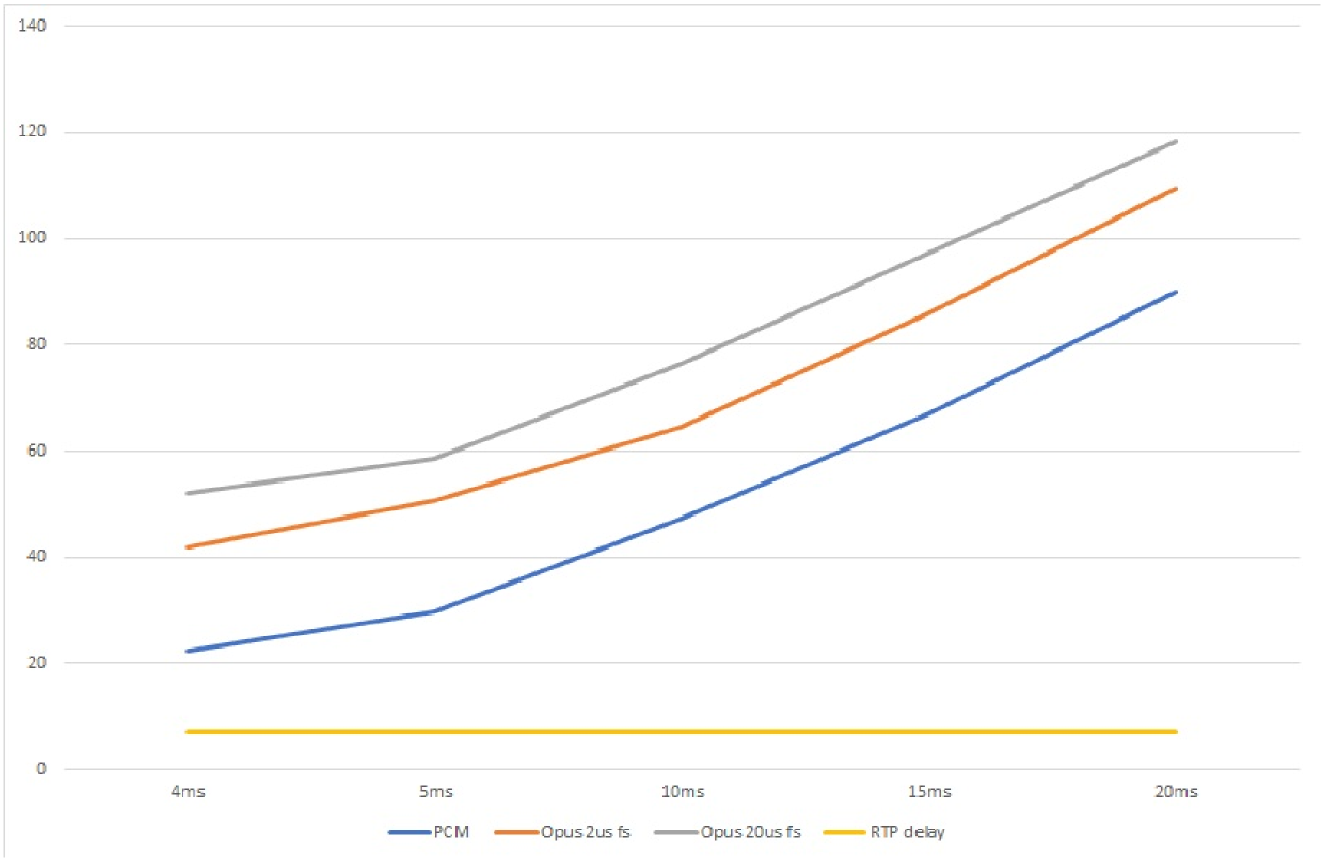


Fig. 8: MM2ME delay in ms, peer to peer via a server, using Aretousa for various buffer sizes.