Audio Delay in Web Conference Tools

Konstantinos Tsioutas and George Xylomenos
Mobile Multimedia Laboratory, Department of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Patision 76, Athens 10434, Greece
E-mail: {ktsioutas, xgeorge}@aueb.gr

***Abstract*—Music collaboration over the Internet, known as Network Music Performance (NMP), remains a challenge for researchers and engineers, since transmission, switching and audio processing delays hinder the synchronization of the participating musicians. Although widely available Web-based voice and video communication tools are not designed for real-time remote musical performances, during the pandemic many musicians worldwide had to use them, due to the lack of widespread NMP-oriented tools. In this paper we provide measurements for the end-to-end audio delay of a number of Web conference tools, using real network conditions, either in a LAN or in a WAN setting, and compare them to the corresponding delays exhibited by an NMP-specific tool.**

I. Introduction

During the CoVid-19 pandemic, methods of real-time remote collaboration have come to the forefront of computing. Google Meet, Microsoft Teams, Zoom, Facebook, Skype and other commercial or open-source tools were widely used for voice and video communication due to the physical isolation imposed by the pandemic. *Network Music Performance* (NMP) though, that is, the performance of music when musicians are connected over a network, is hampered by its need for ultra low delay communication. Even though all human-to-human communications have strict delay requirements, so as to allow the participants to maintain synchronization, NMP is an extreme case. While regular video conferencing can tolerate up to 150 ms of one way audio delay[1], in NMP delays of more than 25-30 ms are considered problematic [1]. Some studies have reported that actual musicians managed to cope with higher delays during real performances [2], indicating that the aesthetic experience of a music performance is a more complex phenomenon. Still, delays of more than 50 ms have been found to be problematic for NMP in all previous studies.

Keeping delay low is hard as, even when the two endpoints are directly connected, the speed at which electrical or optical signals travel is finite. In practical networks, the endpoints are connected via a set of links and switches, therefore the distances are increased, and the switches add delays due to queueing and processing packets. In many cases, a server is needed between the endpoints, to facilitate their connection and the extension of a call to more than two parties. Even the process of gathering enough samples and encapsulating them in a packet for transmission, as well as the reverse process of receiving a packet and playing back the samples it contains, contributes to delay. Finally, since all Web-based communication tools are designed for vocal communication, they apply many audio processing steps like automatic gain control, echo cancellation, noise suppression and others, which may reduce the overall audio quality and increase the delay experienced by the participants, making them problematic for NMP.

Despite all these issues, during the pandemic, many musical activities, such as musical education and rehearsals, were conducted via such tools, even though the resulting delays were large and real time performance was infeasible or very hard to achieve. To clarify how far such tools are from the ideal for NMP, this study is an attempt to measure the real audio delay a user experiences when using some common web conferencing tools and test if this delay exceeds the limits reported in previous studies for NMP. For our experiments we employed real Internet conditions, including DSL links, to assess the delay in a realistic setting, considering both LAN and WAN connections. We also tested an NMP-specific tool, which was optimized specifically to reduce delays, so as to provide a delay baseline.

The outline of the rest of the paper is as follows. In Section II, we briefly present related work on assessing the effects of delay on NMP. In Section III we present the communication tools tested. Section IV describes the setup of our experimental scenarios. In Section V we present the results from this analysis. We summarize our findings and discuss future work in Section VI.

II. Related Work

The *Mouth to Ear* (M2E) delay is used in many studies, either for the evaluation of telecommunication systems in the case of oral communication, or to investigate the delay impact in tempo and musical performance in NMP [3]. It is often referred to as *the one way delay*, since it covers one direction of communication. M2E delay is the time from the moment a sound is generated from the speaker to the moment it will be heard by the listener. Since the speaker and listener may be separated by large physical distances, M2E delay is difficult to measure with high accuracy, as it is difficult to accurately synchronize the clocks of two widely separated endpoints. Thus, in many cases, M2E is either estimated, based on approximate clock synchronization, or the *two way delay* is measured by looping back the audio signal in the one end point. The two-way delay is often referred to as the *My Mouth to My Ear* (MM2ME) [4] delay. Although MM2ME delay can be measured very accurately, as it is measured at a single

[1]https://www.itu.int/rec/T-REC-G.114

point, it obscures any asymmetries between the two directions of communication.

In [5], the authors conducted an extended study to measure the delay experienced in a voice conversation with Skype, Google Talk, Yahoo Messenger and Windows Live Messenger. They used PlanetLab[2] to route audio signals through six Internet paths with known traffic conditions. Instead of using impulse audio signals, they developed a human-response-simulator software which runs at the two endpoints. The authors report that the performance of the various systems is comparable under ideal network conditions. However, performance started to deviate as they introduced more delay, jitter, and loss. These differences indicate that different trade-offs are made by each system to overcome network impairments. In terms of audio delay, they observed that Skype consistently exhibited larger M2E delays, and that Google Talk generally exhibited shorter M2E delays.

In [6], the authors calculated the M2E delay for different web conferencing clients, including Skype, MSN, Google Talk, Yahoo Messenger and others, using the Windows and Linux operating systems, multiple speech codecs and two different Linux kernels, as well as telephone devices. They made multiple observations regarding the M2E latency. Regarding the audio codecs tested, the authors reported that codecs performing well on one client seemed to perform well across all clients (e.g. G.711 A-law) and codecs performing badly on one client seemed to perform badly on all clients (e.g. iLBC). The effect of the codec seemed to be a constant across all the clients. The two kernels examined were a Fedora Core 9 stock kernel (2.6.27.5-37.fc9) and a real-time kernel developed in CCRMA (kernel-rt-2.6.24.7-1.rt3.2.fc9). The real-time kernel from CCRMA performed better in terms of audio latency. Regarding the web conferencing clients, the authors reported that Windows Messenger exposed the smallest latency. Other popular clients like Google Talk, Skype and Yahoo were measured with higher latency values.

The two way, MM2ME delay, can be measured by reflecting the signal back to the sender, and comparing the two waveforms; then, the MM2ME delay is halved to approximate the M2E, one way, delay. In most cases, impulse audio signals are used to determine these delays, as their regularity allows them to be compared easily with a reference signal, so as to determine delay. MATLAB provides the audio system toolbox[3] to determine delay between two audio signals locally captured by a single audio interface in one machine.

In [7] authors proposed a tool for *Distributed Audio/Video Delay Estimation* (DAViDE). DAViDE is a portable embedded device that exploits the clock precision of the *Global Positioning System* (GPS). This allows actual M2E (one way) delays to be measured with high accuracy. They estimated end to end audio and video delays for Skype, Discord, Google Hangouts and ICQ[4] and they reported that Skype, Discord and Google Hangouts achieved delays on the order of 300 ms, while ICQ delay was estimated to be higher than 400 ms.

[2]https://planetlab.cs.princeton.edu/
[3]https://www.mathworks.com/help/audio/ug/measure-audio-latency.html
[4]https://www.icq.com/

It should be noted that the first two studies mentioned above took place in 2009, where the networking landscape was different (even PlanetLab, used in the first study, has been decommissioned), the available applications were different and application capabilities were not the same as today. The third study mentioned above is more recent (from 2018), but it involves custom equipment for measurements. Our goal in this study was to provide an updated set of measurements, but only using easily available software and equipment.

TABLE I
AUDIO CODEC AND QUALITY CONFIGURATION.

Tool	Audio Codec	Audio Quality
Sonobus	PCM	configurable
Jitsi	Opus	configurable
Teams	SILK, G.722	High Fidelity Opt
Zoom	Opus	Original Audio Opt
Facebook	AAC	non-conf
Skype	SILK	non-conf
Google Meet	Proprietary	non-conf

III. AUDIO EXCHANGE TOOLS

In this section, we present the tools examined in this study. Multiple commercial platforms offer audio and/or video calls between two or more users, operating either via a web browser, or as standalone applications. Most of them support multiple operating systems, including Windows, Linux, iOS, and Android. In Web conferencing audio delay is important, but the acceptable delay limit is considered to be 150 ms (M2E). We tested every commercial tool that could be used without charge, but only report results for tools that we could configure in such a way so as to avoid echo cancellation, as echo cancellation prevented us from making delay measurements (see Section 3).

In addition to the commercial tools, we also tested an open source Web conferencing tool, Jitsi, as it is more flexible to configure and setup, free and purely web-based. Finally, we tested an NMP-specific tool, Sonobus, to provide a reference for the delay achievable in a tool specifically engineered to minimize latency. We summarize the tools tested, the audio codecs used and the settings used (if available) in Table I.

NMP-specific tools differ from Web conferencing ones in two ways. First, NMP tools try to minimize their internal delay (by using faster audio drivers, smaller sample buffers and smaller packets), while Web conferencing tools try to avoid making any changes to a user's system. Second, NMP tools accept the need to make tradeoffs due to their very stringent delay budget, therefore they offer multiple configuration options to their users. In contrast, Web conferencing tools use the higher delay budget of voice communication to optimize their behavior, without asking the user to make any choices. Essentially, Web conferencing tools strive for simplicity, while NMP tools strive for performance.

A. Sonobus

Multiple NMP tools exist, providing the ability of real-time musical collaboration, as they were designed for ultra low audio delivery. Soundjack[5] JackTrip[6], LOLA[7] and Sonobus[8] are some of them; an extended overview can be found in [8]. Even though all these tools strive to minimize delays, no tool has yet managed to offer acceptable delays for NMP in a wide variety of networking scenarios.

Sonobus is an open source multi-platform, standalone application for NMP purposes. It achieves ultra low delays by using the ASIO4ALL audio drivers in a Windows environment. It streams uncompressed audio, and uses a peer to peer architecture. It employs a mechanism for the peers to get to know each other, obviating the need to configure port forwarding to bypass firewalls. Sonobus is highly configurable, to allow different audio setups, as opposed to the extremely simple, one size-fits-all, web conferencing tools. We used Sonobus to compare audio delays against the other, more general, Web conference tools and extract conclusions about the sources of delay in real network conditions. We configured Sonobus to use a 48 kHz sampling rate and an audio buffer size of 64 samples (which is only 1.3 ms worth of samples).

B. Jitsi

WebRTC[9] is an open standard architecture for Web Conferencing, offering mechanisms for real-time audio and video delivery, file-sharing, instant messaging, screen sharing and other features, via a web browser. WebRTC allows building conferencing clients directly inside a web browser, without requiring browser plugins or external applications to be downloaded and installed by the user. Jitsi[10] is an open source web conferencing client, based on WebRTC.

Jitsi is easy to set up either locally or in a public server and provides all the features of Web conferencing. In terms of audio delivery, Jitsi employs the Opus codec to reduce bandwidth consumption. Jitsi employs peer to peer communication when only two peers are connected[11]; with more than two peers, it routes audio and video through the *VideoBridge*, an open source *Selective Forwarding Unit* (SFU), that is, a server that only forwards (or drops) packets, without processing them[12]. According to its documentation, Jitsi has a complicated mechanism for handling users, multimedia streams and file exchanges, and it is not clear how it deals with audio streams in terms of ultra low delay and real-time configurations.

For our experiments, we used a Jitsi server, which was set up for the needs of the MusiCoLab[13] project, situated at the Hellenic Mediterranean University (HMU) in the city of Heraklion in Crete, Greece. Since our endpoints were located in Athens, Greece, which is around 300 km away from Rethymnon (on a straight line), this introduced noticeable delays; on the other hand, we knew where the server was located, unlike in commercial tools, where the location of the server is hidden from the user.

Jitsi provides to the user the ability to configure multiple functions, such as enabling or disabling audio processes that might increase audio delay. This configuration is feasible through the URL of the service by adding the *#config...* script. An example of such a configuration is shown below:

*a) https://147.95.32.219/testroom
#config.analytics.disabled=true
&config.prejoinPageEnabled=true
&config.p2p.enabled=false
&config.disableAP=true
&config.disableAEC=true
&config.disableNS=true
&config.disableAGC=true
&config.disableHPF=true
&config.stereo=false
&config.opusMaxAverageBitrate=255000
&config.enableOpusRed=false
&config.enableNoAudioDetection=false
&config.enableNoisyMicDetection=false
&config.disableAudioLevels=true
&config.disableSimulcast=true
&config.enableLayerSuspension=true:* As shown in the script, we disabled all functionality that could be a source of delay. Additionally, we set the *opusMaxAverageBitrate* parameter to the value of 255 Kbps according to the documentation. As tested, the lower the value we set, the bigger the delay that was inserted.

C. Teams

Teams[14] is Microsoft's group collaboration tool, which includes audio and video conferencing, among many other features. Two of our lab members used their Teams accounts to communicate using the Microsoft Teams desktop application (a web version is also available). We enabled the high fidelity audio setting, which allowed us to send and receive impulse audio signals and measure delay; without this setting, reflecting audio so as to measure its delay, was impossible, due to echo cancellation. Teams uses either the SILK codec, a closed source precursor of the Opus codec, or the G.722 codec, a wideband codec adopted by the ITU to support 7 kHz audio.

D. Zoom

Zoom[15] is one of the most popular conferencing tools, as it is freely available with some limitations; commercial licenses also exist. Zoom has the *Enable Original Audio* feature. With this feature, all audio processes, like noise suppression, auto gain control and others are disabled and looping back the audio signal is feasible. Zoom uses the Opus codec for the audio stream.

[5]https://www.soundjack.eu/
[6]https://www.jacktrip.org/
[7]https://conts.it/art/lola-project/old-lola-project-web-site/lola-low-latency-audio-visual-streaming-system
[8]https://www.sonobus.net/
[9]https://webrtc.org/
[10]https://jitsi.org/
[11]https://jitsi.org/security/
[12]https://jitsi.org/jitsi-videobridge/
[13]https://musicolab.hmu.gr/
[14]https://www.microsoft.com/el-gr/microsoft-teams/
[15]https://zoom.us/

E. Facebook

Facebook[16] has a voice and video calling feature which has been widely used during the pandemic especially through smartphones and 4G Networks for instant calls between users. It has been reported by individual musicians that they could conduct a real time musical performance using Facebook during the pandemic. We ran Facebook in two different web browsers on each endpoint machine, and we logged in using four different accounts; we then connected the input and output of the two accounts at one endpoint to achieve audio loopback. In this way we could avoid echo cancellation and were able to send and receive impulse audio signals so to measure delay. Facebook uses the AAC audio codec, popularized by MPEG.

F. Skype

Skype[17] is a very common tool for audio and video calls between two or more users. It has been widely used for a long time; after it was acquired by Microsoft, it was positioned as an independent web conferencing tool (as opposed to Teams, which offers extensive group collaboration functionality). It runs as a desktop application in all operating systems. In terms of audio delivery, Skype uses the SILK codec and a noise cancellation process. Skype provides the feature of hosting a room where a visitor can connect through an Internet browser without logging in to an account. Using this feature, we signed in to one Skype account at the first computer, and we used two browsers in incognito mode for connecting to two different rooms hosted in the first computer. In this way, we could send and receive audio without any audio distortion.

G. Google Meet

Google Meet[18] is Google's web browser based conferencing tool, which provides a room hosting feature. Similar to Teams, it is part of a group collaboration suite. Google also offers the Hangouts conferencing tool, which is an independent tool, similar to Skype, but it has been discontinued, hence we did not test it. In terms of audio quality configuration, Google Meet's documentation states that only Google administrator accounts are allowed to use the option of audio device configuration to enable or disable audio processes. We used four different accounts to login to four corresponding different browsers, two running at each machine, with the two accounts connected at one machine to achieve audio loopback. Google Meet uses a proprietary codec for audio.

IV. Experimental Setup

A. Measuring Delay

The easiest way to measure audio delay in a two-way configuration between two end points is to send audio from one end and loop it back at the other end. Loopback can be configured in multiple ways, from the physical layer to the application layer. One can use the Windows 10 audio feature, *listen to the microphone*, which loops back audio as it captures audio samples and directly feeds them to the audio output using a circular buffer. The smaller the audio buffer, the lower the delay the user experiences to listen to his voice. But if we want to take into account the audio processing delay, each of the examined Web tools inserts to the audio chain, only external audio loo back can be used.

External loopback can be achieved by connecting an audio cable from the audio output to the audio input at the receiving end, allowing the sender to hear a time-delayed version of the signal sent. The problem with this configuration is that many conferencing applications use echo cancellation to eliminate echoes. The echo phenomenon occurs when a speaker hear his or her own voice - while he or she speaks - delayed by a critical amount of time, above which talking is impossible because the speaker is being confused. Most conferencing tools cancel echo by comparing the mic-in stream with the speaker-out stream. If the two signals are similar, then cancellation is enabled and the user cannot listen to the speaker-out stream. This is exactly what we want to achieve with the external loop back, though. Most of the tools examined in this study have strong mechanisms for echo cancellation, due to their voice orientation. As a result, we needed to modify the settings to disable echo cancellation or, if no such setting was available, use different connections in the forward and reverse direction in order to loop back the audio and measure its delay.

B. Network Topology

We used two network configurations, shown in Figures 1 and 2. In both of them, the one endpoint is a computer running Windows 10 placed in the *Mobile Multimedia Laboratory*[19] of the Athens University of Economics and Business, as the one end point. In the WAN setup, shown in Figure 1, the second endpoint was configured behind an ADSL line owned by one of the MMLab members (also in Athens). A Dell latitude laptop, running Windows 10, was connected via an Ethernet cable to the ADSL router and audio was looped back using an audio cable to connect the headphone output to the microphone input. The laptop could be controlled remotely using the Anydesk[20] application. The WAN setup was configured using the ADSL line, so as to experiment with real conditions and not simulated ones. In the LAN setup, shown in Figure 2, the second endpoint was another computer running Windows 10, also connected to the MMLab LAN, again using audio loopback; essentially, the only difference between the two configurations was the intervening network.

To measure the delays induced, two laptops were connected to the endpoint on the left side of the figure: the first one played impulse audio signals, which were fed to the computer where the conferencing client was running, serving as the audio source, and also to the other laptop, which was configured to record two audio signals. The first signal came from the playback laptop (direct audio, blue arrow), fed to the left channel of the external sound interface (not shown in the figures) of the recording laptop. The second signal was the

[16]https://www.facebook.com/
[17]https://www.skype.com/
[18]https://meet.google.com/
[19]https://mm.aueb.gr/
[20]https://anydesk.com/

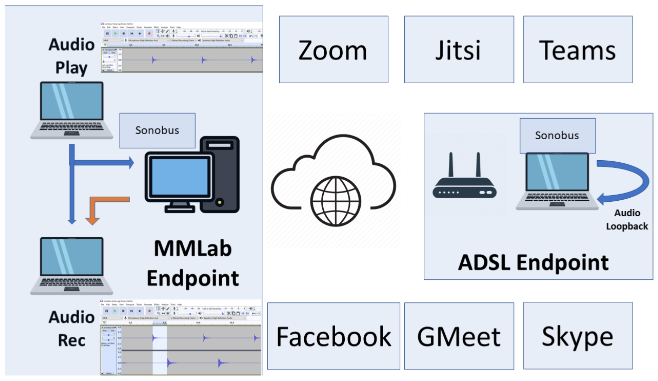

Fig. 1. WAN experimental setup.

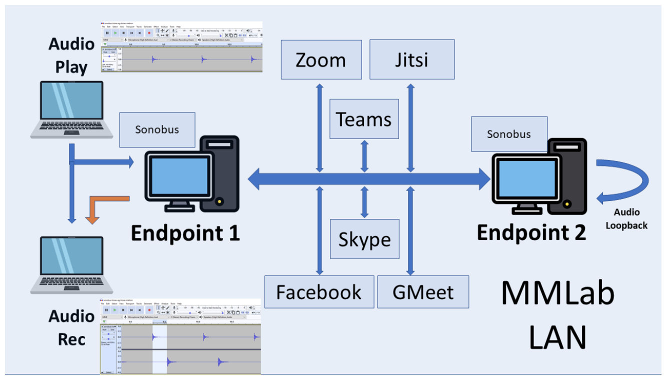

Fig. 2. LAN experimental setup.

looped back one, coming from the output of the conferencing client and fed to the right channel of the external sound interface (looped back audio, orange arrow). Essentially, the recorded signal showed the original signal on the left channel, and the looped back signal on the right channel. By looking at the waveforms of the two channels, we could easily detect the time shift of the impulses.

C. Delay Calculation

All recordings were made with Audacity[21]. The audio routing was configured using an eight channel Mackie mixing console. The recording laptop used an external USB audio interface, the Focusrite Scarlett 2i2, to record the audio from the sender (left channel) and receiver (right channel).

In each experiment, we recorded 60 sec of audio, and then used the MIRToolbox [9] to process the audio. Specifically, for each of the 50 audio pulses sent during the testing period, we first calculate the delay for each pulse, by comparing the outgoing and incoming signal, and then calculate the average delay and its standard deviation.

V. Analysis

Figure 3 shows the average MM2ME (two way) delays for each tool, in seconds, with the blue and red bars showing the delay in the LAN and WAN settings, respectively. Table II shows the detailed results of our measurements: for each tool, we show the average MM2ME, two way, delay measured in milliseconds, and its standard deviation, first for the WAN and then for the LAN setup. The last column shows the difference between the two (WAN delay minus LAN delay).

[21]https://www.audacityteam.org/

From the results shown in Table II, we can observe that Sonobus achieves (by far) the lowest delay values for both configurations, as well as the lowest difference between the WAN and LAN metrics (nearly tied with Zoom). The highest delays in the WAN case are exhibited by Google Meet, as well as the highest divergence between WAN and LAN; in the LAN case, the worst delay is exhibited by Skype, which also has the second highest difference between WAN and LAN, and the highest variance in both scenarios.

Standard deviation is generally higher in the WAN setting, which is as expected, since the network links are far more unpredictable. An unexpected result is that Facebook exhibited less latency in the WAN setup than in the LAN setup. This is likely due to the choice of intervening server, which the endpoints have no control over; if the server chosen is far from the endpoints or the links to it are loaded, delays will suffer. Since in each experiment Facebook can choose a different server, a bad choice can make the LAN setting worse then the WAN one.

TABLE II
Average and Standard Deviation of MM2ME (two way) delay (milliseconds).

	WAN		LAN		Diff
Tool	Avg	Std	Avg	Std	Avg
Sonobus	67	7	46	6	11
Jitsi	259	0.04	235	0.4	24
Teams	363	18	326	0.2	36
Zoom	400	0.9	388	0.3	12
Facebook	545	18	586	0.9	-40
Skype	672	61	613	10	59
Google Meet	1025	15	560	2	465

Based on our previous work [10] which showed that actual musicians found NMP acceptable with delays of up to 80 ms (MM2ME), it is clear that none of the Web conferencing tools is up to the task of providing acceptable delays for NMP sessions, in a realistic setting; even when both endpoints are in the same room, the use of intervening servers leads to unacceptable delays with most tools. Among the Web conferencing tools tested, Jitsi has the lowest delay; Teams and Zoom are one level up in delay; Facebook, Skype and Google Meet have delays that are problematic not only for NMP, but even for voice communications (recall that the delay tolerance for voice calls is generally believed to be 150 ms M2E, or 300 ms MM2ME).

On the other hand, Sonobus offered acceptable delays in the LAN scenario (23 ms M2E) and passable in the WAN scenario (33.5 ms M2E). Even with the limitations of our study (small data set, experiments at a single date/time), the gap between an NMP-specific tool and the Web conferencing tools is more than apparent.

VI. Conclusions

We provided a set of measurements of a number of popular Web conferencing tools, as well as a dedicated NMP tool, in

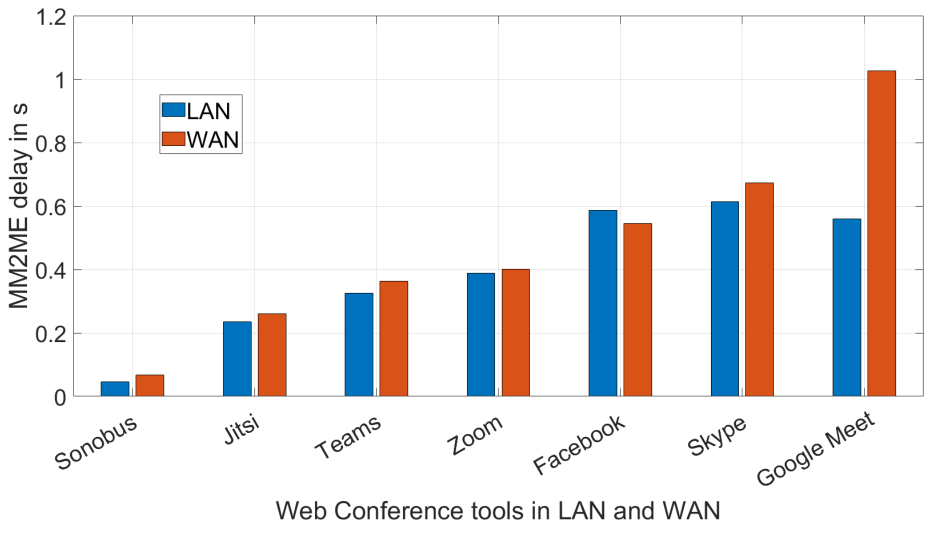

Fig. 3. Average MM2ME (two way) delay (seconds).

order to assess their delay in a realistic setup, either in a LAN or in a WAN environment. Our measurements indicate that none of the Web conferencing tools is capable of offering the ultra low delays needed for effective NMP sessions, unlike the NMP-specific Sonobus which works fine over a LAN and is borderline acceptable over a WAN. Jitsi, Teams and Zoom were found to be acceptable for voice communication, while Facebook, Skype and Google Meet were not even capable of handling the requirements of good quality voice conferencing.

These results are preliminary, in that they do not reflect a large number of test runs at different times and days, and we did not safely determine whether each tool used an intervening server and its approximate geographical position, although the delay metrics, especially for the LAN setup, imply that a server was used in most cases. We are working on gathering additional measurements and determining the path that the audio signals are taking in each test, so as to distinguish the intrinsic delay of the tools from the delay induced by the choice of servers.

Acknowledgments

We would like to thank Prof. Chrisoula Aleandraki for providing us with access the the Jitsi server maintained by the MusiCoLab project.

References

[1] N. Schuett, "The effects of latency on ensemble performance," Bachelor Thesis, CCRMA Department of Music, Stanford University, 2002.

[2] K. Tsioutas, G. Xylomenos, I. Doumanis, and C. Angelou, "Quality of musicians experience in network music performance: A subjective evaluation," in *Audio Engineering Society Convention 148*, May 2020.

[3] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[4] K. Tsioutas and G. Xylomenos, "On the impact of audio characteristics to the quality of musicians' experience in network music performance," *Journal of the Audio Engineering Society*, vol. 69, no. 12, pp. 914–923, 2021.

[5] B. Sat and B. W. Wah, "Analyzing voice quality in popular VoIP applications," *IEEE MultiMedia*, vol. 16, no. 1, pp. 46–59, 2009.

[6] C. Agastya, D. Mechanic, and N. S. Kothari, "Mouth-to-ear latency in popular VoIP clients," Department of Computer Science, Columbia University, Tech. Rep., 2009, CUCS-035-09.

[7] A. Beifuß and B. E. Wolfinger, "Measuring user-perceived end-to-end delays in geographically distributed multimedia systems," in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2018, pp. 1–8.

[8] C. Rottondi, M. Buccoli, M. Zanoni, D. Garao, G. Verticale, and A. Sarti, "Feature-based analysis of the effects of packet delay on networked musical interactions," *Journal of the Audio Engineering Society*, vol. 63, pp. 864–875, November 2015.

[9] O. Lartillot, D. Cereghetti, K. Eliard, W. J. Trost, M.-A. Rappaz, and D. Grandjean, "Estimating tempo and metrical features by tracking the whole metrical hierarchy," in *3rd International Conference on Music & Emotion*, 2013.

[10] K. Tsioutas, G. Xylomenos, and I. Doumanis, "An empirical evaluation of QoME for NMP," in *IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, April 2021.