Network Music Performance in 5G Networks

Konstantinos Tsioutas and George Xylomenos

Mobile Multimedia Laboratory, Department of Informatics,
School of Information Sciences and Technology, Athens University of Economics and Business, Greece

***Abstract*—Network Music Performance (NMP), where musicians perform music together over the Internet, has been a pipedream for decades, due to the ultra-low latency required for effective musical interaction. The 5G networks being deployed around the world promise not only dramatically reduced latencies compared to 4G, but also sufficient bandwidth for both audio and video communication. In this paper we present audio and video latency measurements taken from two real 5G networks, a public 5G Non-Standalone network in Athens and a private 5G Standalone network in Berlin. Our measurements show that 5G in peer-to-peer mode can provide borderline acceptable latencies for audio, but not for video. When a server mediates between the participants, positioning the server in the 5G edge provides dramatically reduced latency, albeit not low enough for NMP.**

***Index Terms*—Network Music Performance, Ultra-Low Latency, 5G-NonSA, 5G-SA.**

I. Introduction

The proliferation of 5G networks has renewed interest in remote interaction applications, which require high bandwidth and low latency to create a sense of presence for the participants. A special case of remote interaction is *Network Music Performance* (NMP), where musicians collaborate via the Internet. The most critical requirement for NMP is that the end-to-end audio delay must be very low for the performance to be perceived as successful: some studies set this threshold to 25–30 ms [1]; our own study indicates that up to 40 ms can be acceptable in some cases [2].

This need for ultra-low latency makes NMP an ideal case for 5G networks which promise network latencies as low as 5 ms from an endpoint to a cell tower. Enhanced presence also requires video, preferably using low latency codecs; unfortunately, the need for low latency makes video coding less efficient, which in turn requires higher bandwidths, such as those promised by 5G. Finally, when more than two musicians participate in an NMP session, it is more practical to have a *Selective Forwarding Unit* (SFU) mediate between them. This SFU should ideally reside in 5G *Mobile Edge Computing* (MEC) servers, located inside the 5G cells, to avoid inflating the latency between the endpoints.

The *Telepresence-Enhanced Network Music Performance* (TENeMP) project[1] is investigating and developing solutions for an immersive NMP experience over 5G. TENeMP is a partner of the SPIRIT project[2], which explores the next generation of telepresence applications in multiple ways, including low-latency networking, scalability to large numbers of users and different forms of telepresence. Crucially, SPIRIT offers *5G Standalone* (5G-SA) testbeds, where the ultra low latency and high bandwidth offered by 5G radios are coupled with a low latency core network and the MEC servers needed to host SFUs. In contrast, many public 5G deployments are actually *5G Non-Standalone* (5G-NonSA), combining a 5G radio network with a 4G core network, thus not offering the full set of 5G capabilities.

In previous work [3] we presented our initial activities in the TENeMP project, including software selection, design of measurement methodologies and alternative setups for audio and video latency measurements, as well as baseline latency results from our local testbed in Athens, which offered LAN, 4G and 5G-NonSA connectivity. In this paper we present our final software configuration, our complete measurement methodology and our detailed test setup, as well as results from both the SPIRIT 5G-SA testbed in Berlin and our 5G-NonSA testbed in Athens. Our goal is to assess whether NMP with audio and video is feasible in a live private 5G-SA network and how this compares with a live public 5G-NonSA network, under a (nearly) identical testing configuration.

The remainder of this paper is structured as follows. In Section II we review related work. In Section III we describe our experimental testbeds. In Section IV we present the audio latency test setup and our results, while in Section V we do the same for video. Section VI summarizes our findings the paper and discusses future work.

II. Background and Related Work

Several papers have considered the feasibility of NMP over 5G. Baratè et al. [4] discuss the performance advantages offered by 5G networks to NMP. Centenaro et al. [5] outline a communication architecture for NMP in 5G networks, while Vignati et al. [6] compare NMP performance in 4G and 5G networks through simulations, suggesting that the latter can offer notable gains. Carôt et al. [7] describe a distributed 5G-based music event that took place in 2019 as a proof of concept. There is also some work experimentally evaluating NMP over 5G networks. Dürre et al. [8] conduct an in-depth performance analysis on NMP over a public 5G network in Finland, finding that 5G can support NMP, although performance variability raises concerns. Turchet and Casari [9] explore NMP performance in a private 5G network with four musicians located in the same radio cell, reporting that a continuous stream of reliable, low-latency communication is challenging, even in a private 5G network,

[1]https://mmlab-aueb.github.io/tenemp-site/

[2]https://www.spirit-project.eu/

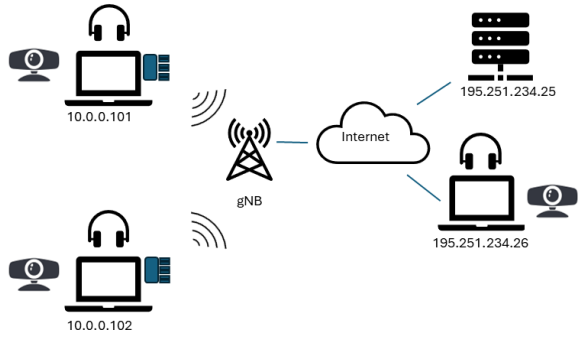

Fig. 1. MMLab testbed (5G-NonSA).

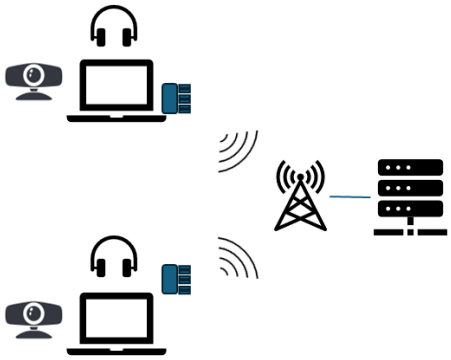

Fig. 2. Berlin testbed (5G-SA).

highlighting the need for edge computing. In a more recent work, Turchet and Casari [10] compared a private 5G-SA with a public 5G-NonSA network in terms of audio latency, using a simulated NMP setup; they found end-to-end packet delay to be 21.9 ms and 35.4, respectively. Finally, Turchet et and Casari [11] discuss the impact of 5G slicing on low latency audio transmission, weighing the advantages of having a custom slice for NMP against the overhead due to slicing. These studies focus solely on audio and do not consider video at all. Turchet et al. [12] combined an NMP-optimized audio streaming application with a generic volumetric video streaming application over a LAN, estimating the delay divergence between the audio and the volumetric video to be around 400 ms, which was more than noticeable.

In contrast to previous work, TENeMP focused on assessing the feasibility of immersive NMP over 5G, encompassing not only audio, but also 2D video and volumetric video [13]. Furthermore, since 5G deployments are quite diverse, our goal was to assess the performance differences for NMP between 4G, 5G-NonSA and 5G-SA networks, with or without an intermediate SFU, in a realistic and easily replicated setup. Unlike the work of Turchet and Casari [10], which also compared 5G-SA and 5G-NonSA networks by measuring the end-to-end audio packet delay, in our work we measure the complete mouth-to-ear and glass-to-glass delay, which is what the users perceive.

III. The 5G testbeds

The primary goal of the TENeMP project is to test NMP performance in the 5G-SA testbed offered by the SPIRIT project, located in Berlin. However, as this requires physical travel to the testbed's location, it is only economically feasible for short periods of time. Development and testing of software, measurement procedures and experimental setups, took place before visiting the testbed, allowing the visits to focus on final setup and experiment execution. Another complicating factor is that the Berlin testbed is isolated from the Internet for experimental purposes; that is, the actual endpoints of the 5G-SA network must be located in the testbed and they cannot communicate with the outside world. On the other hand, in addition to the endpoints, the Berlin testbed includes a MEC infrastructure located inside the 5G cell, where any deployed services are reachable from the endpoints with reduced latency.

In NMP there are two basic scenarios of communication, the peer-to-peer (P2P) scenario where each musician directly communicates with all others, and the Client-SFU scenario where each musician only communicates with the others via an SFU. The *Selective Forwarding Unit* (SFU), acts as a relay server between multiple peers by receiving only a single media stream from each endpoint and forwarding it to the others, without any processing, so as to avoid inflating delay; the alternative would be for each peer to send its media stream to each other participant, which is not scalable. As a result, the P2P topology is only used with two musicians; with more musicians, the Client-SFU topology is far more common. The SFU needs to reside at a server, ideally located close to the endpoints (e.g., a MEC server). In most current conferencing systems, however, the SFU is located in the cloud, introducing considerable latency between the participants.

To develop and test the required tools (endpoints and SFU), we created a local testbed at the MMLab in Athens. The hardware and network setup replicates as much as possible the SPIRIT testbed in Berlin, using similar depth cameras (Intel RealSense D435), AR glasses (XREAL Air 2 Pro) and 5G end devices (Samsung S24 5G smartphones and Teltonica RUTX50 5G routers). As shown in Figure 1, the testbed consists of two 5G endpoints, as well as a server in the MMlab, with an optional third endpoint reachable over the wired Internet. Each endpoint supports bidirectional ultra low delay audio and video streaming; the local endpoints are either 5G smartphones or laptops connected via Ethernet to a 5G router. The endpoints used for the measurements reported in this paper were Asus TUF Gaming A15 laptops, running Ubuntu 24.04.2 LTS, with embedded Realtek sound cards and 720P HD Logitech USB web cameras.

Figure 2 shows the configuration of the Berlin testbed. The endpoints (laptop, peripherals, 5G modems) are the same, but the network is quite different. First, at the MMLab we only had access to a 5G-NonSA network, while in Berlin the network operated in 5G-SA mode. Second, the Berlin testbed included MEC servers deployed inside the 5G cell,

unlike the MMLab testbed where they were deployed in our LAN and were reachable over the Internet. Third, the Berlin testbed was a closed network, not reachable from the Internet, using private IP addresses for the endpoints and MEC services, making for easy communication. On the other hand, the MMLab testbed used a public 5G network (COSMOTE/Telekom), which allowed communication with endpoints and servers on the Internet, but using NAT to translate between private and public IP addresses.

The use of NAT necessitated taking special measures to allow communication in the MMLab testbed, as getting two endpoints with private IPs to communicate with each other is not a trivial exercise. Initially, we relied on the WebRTC architecture and STUN and TURN servers, deploying a signaling server and the ICE and SDP protocols for peers to establish a bidirectional communication [3]. However, to reduce latency, we later used the UDP hole punching technique to establish bidirectional audio and video communication between the endpoints; of course, this is not an issue when an SFU is used, as our servers, located in the MMLab, used a public IP address. On the other hand, in the Berlin testbed there was no NAT, so both direct and indirect (via an SFU at the MEC) communication was possible without any extra steps.

IV. Audio Latency

A. Audio streaming tools

There are multiple tools available for streaming audio in real time; JackTrip[3], Sonobus[4], Jamulus[5], and SoundJack[6] are some of the state of the art open source tools for ultra low delay and high audio quality streaming. JackTrip and SoundJack support P2P communication when the endpoints can reach each other without NAT; they also supports Client-SFU communication, via a server which uses a public IP address. Sonobus operates in P2P mode only, using a public signaling server for peers to get to know each other and establish multiple connections between them. Jamulus only supports the Client-SFU architecture, providing a server application that must run on a machine with a public IP address.

In addition to these NMP-specific tools, real-time audio streaming is also feasible via simple Internet browsers. A peer-to-peer connection can be established using the Web APIs and a signalling server, with WebRTC providing access to the local audio and video devices and offering functions to establish bidirectional audio and video communication between two browsers. The Client-SFU mode is also an option, using frameworks such us Jitsi[7] and MediaSoup[8] to provide an SFU and a signaling server to help establish communication between the endpoints.

[3]https://www.jacktrip.com/
[4]https://sonobus.net/
[5]https://jamulus.io/
[6]https://www.soundjack.eu/
[7]https://desktop.jitsi.org/
[8]https://mediasoup.org/

A very flexible and customizable tool for both audio and video streaming is the Gstreamer framework[9]. Gstreamer provides plugins that construct pipelines connected in series. With Gstreamer it is very easy to capture raw audio from a sound card and create a UDP audio stream to send audio to a known IP address. This stream can be easily played back at the receiver side, with ultra low delay in LAN conditions. Similar pipelines can be created for video, with various codecs and options. In our previous work, we found Gstreamer pipelines to be faster and more flexible than using the Web APIs [3], so in this paper we focus exclusively on Gstreamer.

B. Measurement methodology and setup

Mouth to Ear (M2E) audio latency is the time between a user producing a sound and the sound reaching the ears of another user. We conducted M2E measurements using the *reflected pulse method*, shown in Figure 3, which we also used in the past to measure the delay of various conferencing tools [14]. In this approach, we produce audio pulses using the Audacity[10] software, which are sent to a receiver, played back there, captured again and returned to the sender; we record both the original and the returned signal as the left and right channels of a stereo signal, and then visually inspect the recorded waveforms to calculate the round trip delay of the signal. Assuming a symmetric connection, half of that delay is the M2E audio latency.

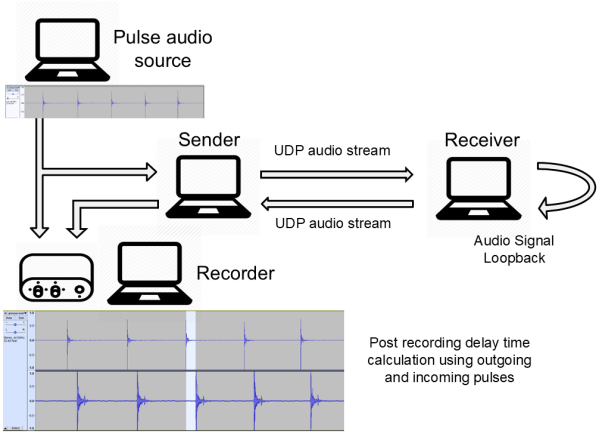

Fig. 3. Audio latency measurement setup.

For the actual measurements, we used Gstreamer pipelines in Linux to establish both P2P and Client-SFU audio connections between the two nodes. We sent an audio stream over UDP (with RTP encapsulation) consisting of a single channel of uncompressed audio, sampled at 44.1 KHz with 16 bits per sample, using the `pulsesrc` plugin for capture and the `pulsesink` plugin for playback. We set the `buffer-time` parameter to 3000, or 3 ms, the lowest possible value for our

[9]https://gstreamer.freedesktop.org/
[10]https://www.audacityteam.org/

hardware before corruption occurred, to reduce the packetization delay; this meant that each packet included 132 samples, or 264 bytes. We used the `rtpL16pay` plugin to packetize the data in RTP; the actual UDP packet payload size was 276 bytes, including the 12 byte RTP header. The Gstreamer pipeline for the sender side in the Berlin testbed was:

```
gst-launch-1.0 pulsesrc
latency-time=3000 buffer-time=3000 !
rtpL16pay ! application/x-rtp,media=audio,
clock-rate=44100,encoding-name=L16,
channels=1 ! udpsink host=10.24.2.24
port=10000 -v
```

A similar pipeline was used for audio playback at the receiver. For the Client-SFU measurements, we ran a Gstreamer pipeline in a Linux Docker container, running inside the Kubernetes cluster at the MEC server; in the Berlin testbed, this was inside the cell, in the MMLab testbed it was in our lab. The pipeline relayed incoming RTP/UDP packets from one client to the other, without additional processing. The Gstreamer pipeline for the SFU in the Berlin testbed was:

```
gst-launch-1.0 udpsrc port=10000
! application/x-rtp,media=audio,
clock-rate=44100,encoding-name=L16,
channels=1 ! udpsink host=10.24.2.4
port=10000
```

Slightly modified pipelines were used in the MMLab testbed, due to the need to perform the hole punching procedure to cross the NAT in P2P mode and the use of a server with a public IP addresss in the Client-SFU mode. For both the P2P and Client-SFU modes, we executed three (3) trials of one (1) minute duration each. During each trial, 30 audio pulses were transmitted and reflected back. With these parameters, we did not experience significant losses.

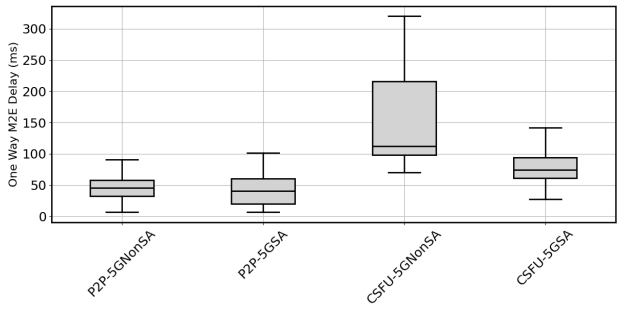

Fig. 4. Audio delay (M2E) for P2P and Client-SFU topologies in 5G-NonSA and 5G-SA networks.

C. Audio latency results

In Figure 4 we show boxplots for the end-to-end audio delay; from left to right, we show the results for the P2P topology in the MMLab (5G-NonSA) and in the Berlin testbed (5G-SA) and then for the Client-SFU topology, again first for the MMLab and then for the Berlin testbed. The boxplots show the median latency (the line inside the box), while the box edges show the 1st and 3rd quartile of the latencies; whiskers show the extent of the latencies that are no more than 1.5 times more than the IQR (the distance between the 1st and 3rd quartile) from the edges of the box. We also summarize the median values and standard deviations of the results in Table I.

The median value of the end-to-end audio delay for P2P mode was 40.55 ms in the 5G-SA network, lower than the 45.08 ms measured in the 5G-NonSA network, although with a slightly higher standard deviation. This is borderline acceptable for NMP based on our previous work which placed the delay tolerance limit to 30–40 ms [2], indicating that 5G-SA makes audio-based NMP feasible, at least for people located relatively close (within the same 5G cell or area).

In the Client-SFU mode on the other hand, we can see a large difference between the 5G-SA and 5G-NonSA networks (74.42 ms vs. 112.19 ms), which is to be expected, as in the 5G-NonSA case the SFU resides in our lab, outside the 5G cell, therefore packets need to exit the 5G network, travel to our lab, and return to the 5G network. Not only this inflates the delay, but it makes it quite unpredictable, as shown by the higher standard deviation. On the other hand, this is much lower than the delays we saw with commercial conferencing tools [14], as the SFU was latency-optimized and relatively close to the endpoints (in the same city), as opposed to a remote data center, possibly located in a different country. In the 5G-SA network, where the SFU is inside the cell and directly reachable by both endpoints, the latency is rather high for real-time musical performance, but probably acceptable for other NMP scenarios, such as music teaching.

V. Video tools, measurement setup, results

A. Video streaming tools

Most of the studies that investigate the feasibility of NMP, focus mainly on audio delay since it is the basic modality in musical performance. Furthermore, video requires so much bandwidth, especially when low-latency compression is used (to avoid inflating delay), that it has been impractical to add it to NMP tools targeting 4G or previous networks. In the absence of dedicated low latency video streaming tools, we tested both Gstreamer based and Web API/WebRTC based solutions, finding that Gstreamer exhibited lower delay [3]. Gstreamer is also far more flexible, as in addition to plugins for capturing, packetizing and streaming frames, it also allows different video codecs to be used and their parameters to be configured in detail, allowing latency-optimized pipelines.

B. Measurement methodology and setup

Measuring *Glass to glass* (G2G) video latency, which is to the time it takes for a single frame to be captured by a camera at one endpoint, and the corresponding frame to be presented at a screen at the other endpoint, is not straightforward. Both the camera and the screen operate at a few tens of Hz with different clocks, which makes video latency measurements exhibit high variances [15].

We conducted measurements to calculate G2G video latency by using two methods, a *reflected timer* [2] and a

flashing LED method [3] which was based on [15]. The reflected timer method relies on capturing a timer (running on a smartphone) at the sender and displaying it at the receiver, with a separate camera capuring both the smartphone and the receiver's screen. Although with this approach many frames are useless, due to tearing caused by the scan lines in the cameras and displays, the video can be analyzed manually to gather latency samples from the good frames.

The flashing LED method relies on the calculation of the time it takes for a flash of light produced by an LED and captured by the web-camera at the sender peer, to be streamed over the network and appear at the screen of the receiver peer. The exact setup, shown in Figure 5, used a LED pointing at the center of the camera, with both being enclosed in a dark plastic box. This meant that the camera was streaming black frames, until the LED went; the LED was lit once per second for 10 ms and then was dark for 990 ms. This resulted in a mostly black video stream which was easy to compress and transmit with low latency. The LED is easier to capture and detect with a *Light Dependent Resistor* (LDR) sensor, as we remove the smartphone and camera from the setup, thus providing more latency samples.

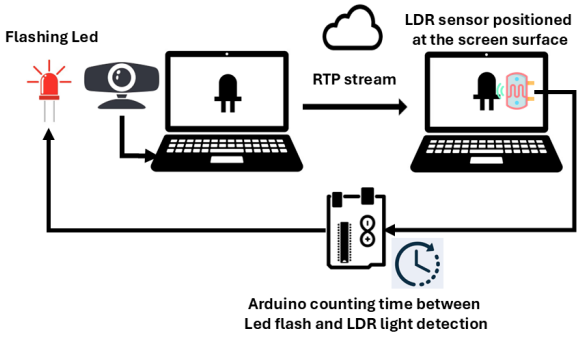

Fig. 5. Video latency measurement setup.

For the tests in this paper we used the `v4l2src` plugin to capture video frames and encode them with the H.264 codec using the `x264enc` plugin, which relies on a classic *Group of Frames* (GOP) structure that inflates delays [16]. We also experimented with the VP8 codec, which has a low latency option, using the `vp8enc` plugin, but it performed worse than the `x264enc` plugin, corrupting most frames. We used `rtph264pay` to create the RTP/UDP video stream. The Gstreamer pipeline at the sender for the Berlin testbed was:

```
gst-launch-1.0 v4l2src ! videoconvert
! x264enc ! rtph264pay ! udpsink
host=10.2.24.2 port=10000 -v
```

The pipeline created 1400 byte UDP packets (the default size). For the Client-SFU topology, we deployed again a simple Gstreamer pipeline at the Kubernetes cluster, which was nearly the same as the pipeline shown above for audio.

In the MMLab testbed, video streaming had to be configured with a smaller UDP packet size for the communication to be successful. Using the default UDP packet size (1400 bytes) led to packet drops and video corruption, therefore we set the `mtu` parameter in the GStreamer pipeline to 300 bytes; this meant a slightly higher overhead due to more RTP headers, but led to good video quality.

For video, we performed three (3) trials of one (1) minute each. During each minute 60 light pulses were transmitted via the LED, which were captured by the web camera at a rate of 30 frames per second. With these parameters, we did not experience significant losses.

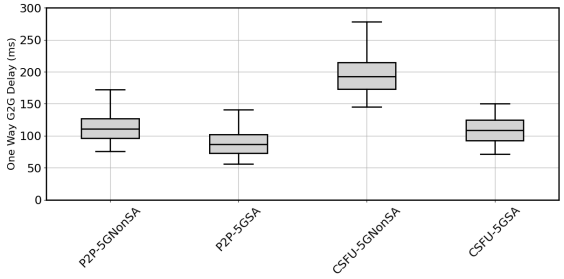

Fig. 6. Video delay for P2P and Client-SFU topologies in 5G-NonSA and 5G-SA networks.

C. Video latency results

Figure 6 shows the results from the video latency measurement tests in the form of boxplots; as for the audio results, we first show results from the P2P topology with the 5G-NonSA and the 5G-SA network, and then from the Client-SFU topology with the 5G-NonSA and the 5G-SA network. Detailed results are also shown in Table I.

As expected, the G2G median video latency measured in the Berlin 5G-SA testbed (87 ms) is reduced compared to the MMLab 5G-NonSA testbed (111 ms), far more than in the audio case. Again, these results are not sufficient for live NMP interaction, but in the 5G-SA case they may be acceptable for music teaching. In the Client-SFU mode, we have again a much higher difference between the two testbeds (108.5 ms vs. 193 ms), since in the MMLab testbed the SFU is located outside the 5G network. Interestingly, the addition of the SFU in the video case in the Berlin testbed adds less latency (around 20 ms) compared to audio (around 34 ms). In the Athens testbed on he other hand, the extra latency due to the SFU was more pronounced with video (around 82 ms) than with audio (around 67 ms).

VI. Conclusions and Future Work

5G networks promise ultra low delay, high bandwidth and processing close to the mobile edge. These aspects could finally make NMP a reality, supporting low latency and high quality audio and video. In this paper we described our local 5G-NonSA testbed and the Berlin 5G-SA testbed provided by the SPIRIT project, the NMP tools that we used, our measurement methodologies and an overview of our measurements of audio and video latency in both testbeds.

As we can see from the summary of results in Table I, only audio in P2P mode has latencies low enough for live

TABLE I
AUDIO AND VIDEO DELAY MEASUREMENTS (MS)

Audio				
Testbed	P2P		Client-SFU	
	Median	Std	Median	Std
MMLab (5G-NonSA)	45.08	18.24	112.19	64
SPIRIT (5G-SA)	40.55	25.73	74.42	25.85
Video				
Testbed	P2P		Client-SFU	
	Median	Std	Median	Std
MMLab (5G-NonSA)	111	25.53	193	53.68
SPIRIT (5G-SA)	87	18.93	108.5	19.12

NMP sessions; this is true for the 5G-SA Berlin network (40.55 ms), and partially true for the 5G-NonSA Athens network (45.08 ms). For video, the latencies are higher (87 ms in 5G-SA and 111 ms for 5G-NonSA), which are impressive compared to 4G, but insufficient for live NMP. With parallel transmission of audio and video, these numbers may also be inflated. Adding an SFU inside the network, in the MEC infrastructure of the 5G-SA testbed, adds 20 ms and 34 ms of delay for video and audio, respectively. When the SFU is located on the Internet, as is the rule today, where SFUs are located in data centers, the latency penalty is far higher. We can therefore conclude that 5G networks can support NMP audio sessions in P2P mode; adding an audio SFU in the cell and/or P2P video makes live NMP infeasible, but could be acceptable for scenarios like music teaching.

Another strand of work in the TENeMP project is volumetric video streaming; current volumetric video codecs do not use inter-frame compression, which makes for lower latency, but requires far higher bandwidths than what is available even at high speed 5G testbeds, necessitating bandwidth reduction measures, such as reducing the point cloud size by dropping samples. Furthermore, volumetric video is far more computationally expensive, requiring careful management of CPU resources and thread-level parallelization [13].

After the conclusion of the TENeMP project, we are continuing our NMP work in the *Adaptive Video Delivery for Network Music Performance* (AViD-NMP) project[11], supported by 6G-XR[12]. In this project we are focusing on making volumetric streaming faster and more efficient, by offloading computations to more powerful servers (like the SFU), improving volumetric video coding and composing different volumetric video sources to a single scene to allow for multiparty NMP sessions. These extensions will be tested in the 5TONIC testbed in Madrid, hosted by the 6G-XR project partners, offering more data points in our evaluation of NMP performance over 5G.

ACKNOWLEDGMENT

The work reported in this paper has been partly funded by the EU through the subgrant Telepresence-Enhanced Network Music Performance (TENeMP, SPIRIT OC1) of project SPIRIT (grant agreement No. 101070672). The SPIRIT project has also received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI). The work has also been partly funded by the EU through the cascading action Adaptive Video Delivery for Network Music Performance (AViD-NMP, 6G-XR OC3) of project 6G-XR (grant agreement No. 101096838). The 6G-XR project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU). The views and opinions expressed are however those of the authors only and do not necessarily reflect those of the EU. Neither the EU nor the granting authority can be held responsible for them.

[11] https://mmlab-aueb.github.io/avid-nmp/

[12] https://6g-xr.eu/

REFERENCES

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

[2] 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)*, 2021, pp. 1–5.

[3] K. Tsioutas, Y. Thomas, F. Bistas, I. Barous, G. Xylomenos, and G. C. Polyzos, "Network music performance beyond 4G," in *21st International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2025.

[4] A. Baratè, G. Haus, and L. Ludovico, "Advanced experience of music through 5G technologies," *IOP Conference Series: Materials Science and Engineering*, vol. 364, p. 012021, 06 2018.

[5] M. Centenaro, P. Casari, and L. Turchet, "Towards a 5G communication architecture for the Internet of musical things," in *27th Conference of Open Innovations Association (FRUCT)*, 2020, pp. 38–45.

[6] L. Vignati, G. Nardini, M. Centenaro, P. Casari, S. Lagén, B. Bojovic, S. Zambon, and L. Turchet, "Is music in the air? evaluating 4G and 5G support for the Internet of musical things," *IEEE Access*, 2024.

[7] A. Carôt, F. Sardis, M. Dohler, S. Saunders, N. Uniyal, and R. Cornock, "Creation of a hyper-realistic remote music session with professional musicians and public audiences using 5G commodity hardware," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE Computer Society, 2020, pp. 1–6.

[8] J. Dürre, N. Werner, S. Hämäläinen, O. Lindfors, J. Koistinen, M. Saarenmaa, and R. Hupke, "In-depth latency and reliability analysis of a networked music performance over public 5G infrastructure," in *Audio Engineering Society Convention 153*, 2022.

[9] L. Turchet and P. Casari, "Latency and reliability analysis of a 5G-enabled Internet of musical things system," *IEEE Internet of Things Journal*, vol. 11, no. 1, pp. 1228–1240, 2023.

[10] ——, "Assessing a private 5G SA and a public 5G NSA architecture for networked music performances," in *2023 4th International Symposium on the Internet of Sounds*, 2023, pp. 1–6.

[11] ——, "On the impact of 5G slicing on an internet of musical things system," *IEEE Internet of Things Journal*, vol. 11, no. 19, pp. 32 079–32 088, 2024.

[12] L. Turchet, N. Garau, and N. Conci, "Networked musical XR: where's the limit? a preliminary investigation on the joint use of point clouds and low-latency audio communication," in *Proceedings of the 17th International Audio Mostly Conference*, ser. AM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 226–230. [Online]. Available: https://doi.org/10.1145/3561212.3561237

[13] Y. Thomas and G. Xylomenos, "Ultra-low latency point cloud streaming in 5g," in *Proceedings of the 22nd EuroXR International Conference*, 2025.

[14] K. Tsioutas and G. Xylomenos, "Audio delay in web conference tools," in *Workshop on Web Engineering and Collaborative Music Learning (WECML)*, 2022, pp. 1–6.

[15] C. Bachhuber, E. Steinbach, M. Freundl, and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," *IEEE Transactions on Multimedia*, vol. 20, no. 1, pp. 238–252, 2018.

[16] A. Carôt, C. Hoene, H. Busse, and C. Kuhr, "Results of the Fast-Music project – five contributions to the domain of distributed music," *IEEE Access*, vol. 8, pp. 47 925–47 951, 03 2020.