Reconstructing the Sound at the Edge: A Low-Latency Framework for NMP with Autotuned Local Synthesis

Luca Borgianni[1], Cristian Bua[1], Davide Adami[2], Stefano Giordano[1]

[1]*University of Pisa, Dept. of Information Engineering, Via G. Caruso 16, 56122 Pisa, Italy*
[2]*CNIT - University of Pisa, Dept. of Information Engineering, Via G. Caruso 16, 56122 Pisa, Italy*

***Abstract*—Networked Music Performance (NMP) systems aim to enable real-time collaboration between geographically distributed musicians. Traditional approaches transmit raw audio over IP networks, suffering from delay, jitter, and packet loss, which negatively impact the quality of experience (QoE). In this paper, we propose a novel architecture for NMP that departs from audio streaming: instead of transmitting the raw audio, we transmit symbolic musical control data (e.g., MIDI or pitch with dynamics), and synthesize the audio at the destination with potential benefits in terms of latency, bandwidth, and robustness. In addition, the new framework goes in the direction of a cloud continuum NMP, allowing the use of edge processing, such as the application of real-time effects like autotune, to improve the subjective experience. In this work, we provide the open-source code of the proposed system, and we evaluate the Quality of Experience (QoE) to assess the feasibility of the system. The proposed architecture provides a solid foundation for low-latency and scalable NMP systems with potential real-world deployment.**

***Index Terms*—Networked Music Performance (NMP), Low-latency systems, Edge computing, Audio synthesis.**

I. Introduction

Music, as a universal language, has always found technology to be a powerful ally in overcoming physical boundaries. In recent years, the growing interest in Networked Music Performance (NMP) has led to the emergence of systems that allow musicians located in geographically distant locations to play together in real time, attempting to reproduce the conditions of a live acoustic performance. However, the technical challenges associated with the transmission of uncompressed audio in real time, such as latency, jitter, and packet loss, still represent a major obstacle to the deployment and reliability of these systems.

The currently popular NMP systems, such as JackTrip, LOLA, and SoundJack, rely predominantly on the transmission of raw or compressed audio streams via UDP, providing high-level performance in certain contexts, but often requiring low-latency, high-capacity network connections, such as dedicated fiber networks. This limits their use outside academic or professional environments, effectively excluding a large segment of potential users, such as musicians in rural or mobile settings.

In this scenario, it becomes urgent to rethink the very architecture of the music transmission system: Is it really necessary to transmit audio? Or can we design a lighter, adaptable, and resilient system that transmits only the minimal information required to reconstruct the sound realistically at the receiving node?

In this paper, we propose a radically different approach that combines the advantages of real-time audio synthesis technologies, edge-based architectures, and the use of symbolic data (such as MIDI or pitch + envelope events). The central idea is to reconstruct the sound locally at the receiving node, avoiding the transmission of raw audio and instead exploiting a lightweight stream of control data. This dramatically reduces the required bandwidth, increases robustness against adverse network conditions, and enables edge processing.

In addition, the introduction of intelligent local effects (such as dynamic autotune, adaptive reverbs, or envelope shaping) allows for enriching musical perception and compensating for any network or source imperfections, improving the quality of perceived experience (QoE) by users. In this work, autotune is introduced only as an illustrative feature of the architecture, while the present evaluation focuses on symbolic transmission and synthesis.

The main contributions of this paper are as follows:

- A new distributed architecture for NMP, which abandons the audio streaming paradigm;
- An open source code related to the implemented framework [1].
- A real-world implementation of the system, tested under varying network conditions;
- A qualitative evaluation of the perceptual impact, through subjective tests on musicians and non-musicians.

The rest of the paper is organized as follows. Section II presents the most common approaches used for NMP. Section III introduces the implemented framework. Section IV presents the evaluation of the new approach with a Mean Opinion Score (MOS) survey, and finally V concludes the work.

II. Related Works

In this section, we describe the methods used in the literature to transmit audio in the field of NMP and their

[1]https://github.com/Borgianni/Recostruction_Sound_Edge

performance and effectiveness during transmission over communication networks. NMP is an area that investigates the possibility of real-time collaborative music performances over networks, aiming to reproduce the same conditions as acoustic instrumental on-site performances.

Very low communication latency, low and constant jitter (i.e., variation of latency), and high audio quality (i.e., low packet losses resulting in unperceivable dropouts) are fundamental parameters in NMP for ensuring Quality of Service (QoS), as analyzed in [1]. The effects of latency in NMP are analyzed in [2], while [3] highlights that the choice of jitter buffer size is non-trivial, requiring trade-offs between latency and audio quality. Moreover, it should be acknowledged that the Perceptual Evaluation of Audio Quality (PEAQ) [4] is designed to predict basic audio quality (BAQ) in the presence of codec artifacts, and is not specifically intended to model the degradations introduced by network impairments in NMP scenarios. For this reason, MOS-based subjective evaluation remains the primary indicator of perceptual quality in our study.

Audio transmission over communication channels must be reliable and fast, despite potential connectivity interruptions. In this context, Packet Loss Concealment (PLC) mechanisms play a fundamental role in NMP to mitigate the effects of packet erasures and preserve audio quality during transmission interruptions. In particular, redundancy-based methods such as Forward Error Correction (FEC) are widely adopted to enhance robustness against packet loss [5]. Beyond redundancy, codec-specific PLC strategies are crucial. The Opus codec, already acknowledged for its widespread adoption in NMP applications, incorporates advanced PLC techniques, including waveform extrapolation and data-driven approaches such as DRED (Deep Redundancy) [6]. These methods significantly improve perceived continuity during dropouts and are especially relevant in constrained or unstable network environments typical of home or mobile broadband connections. Furthermore, [7] proposes a low-complexity error correction scheme employing a period extraction and alignment module based on zero-crossings and matched pre-processing, while [8] utilizes an autoregressive model to compensate for such interruptions.

Several solutions have been proposed both in academia and commercially to address NMP challenges. Among academic solutions, JackTrip, developed at Stanford CCRMA, transmits uncompressed PCM audio over UDP with minimal signal processing limited to buffer, jitter, and delay management [9]. Although it provides high fidelity and low latency performance, it demands stable, high-bandwidth connections. Similarly, the Low Latency audio-visual streaming system (LOLA), developed by Italian conservatories in collaboration with GARR, enables ultra-low latency performances by transmitting uncompressed audio and MJPEG-compressed video via UDP, but its deployment is limited to academic-grade fiber networks [10].

Soundjack, a hybrid research-commercial system from Carôt Alexande [11], offers compressed transmission via the Opus codec over UDP in peer-to-peer or client-server configurations, incorporating compression, echo cancellation, and monitoring, making it suitable for home networks while maintaining acceptable latency [12].

In the commercial domain, Jamulus compresses audio using Opus and implements centralized server mixing. While it incurs slightly higher latency compared to JackTrip, its accessibility has driven widespread adoption [13]. Other commercial services, such as Audiomovers ListenTo [14] support both compressed and lossless audio over TCP/UDP, facilitating high-quality remote mixing, though not optimized for synchronized real-time performances. Source-Connect, mainly used in broadcasting and dubbing, provides high-quality compressed audio but with latency levels unsuitable for ensemble playing [15]. Lastly, Elk Audio OS and Elk LIVE combine hardware and software to achieve ultra-low latency, reportedly under 5 ms between devices, by optimizing compression and networking at the system level [16].

Table I summarizes the audio codecs commonly used in NMP systems.

To reduce network load and enhance communication robustness, synthesizing audio during transmission and regenerating it upon reception has been proposed as part of the Internet of Musical Things (IoMusT) vision. For this reason, it is relevant to discuss audio synthesis and generation techniques from the literature. In sound synthesis, conditioning the Generator on mel-spectrogram inputs is a well-established approach, as described in [17], with mel-spectrogram conditioned Generators also used as vocoders [18]. Beyond mel-spectrograms, conditioning has been implemented using raw audio [19], one-hot vectors encoding musical pitch [20], linguistic features [21], or latent representations identifying speakers [22].

Regarding audio generation methods, MelGAN introduces a Multi-Scale Discriminator operating on multiple scales of waveforms modulated by Average Pooling [18]. WaveGAN proposes a multi-resolution spectrogram loss that stabilizes adversarial training and improves generation quality [23], while HiFi-GAN uses a Multi-Period Discriminator to identify periodic audio patterns for synthesizing high-fidelity outputs [24]. Finally, Fre-GAN2 synthesizes frequency-consistent audio comparable to ground-truth recordings [25]. These GAN-based methods outperform older autoregressive [26] and flow-based vocoder approaches [27].

Recently, innovative approaches have been explored to overcome NMP limitations, including adaptive buffering (e.g., JackTrip 2.0, QuackTrip), QoS-aware routing via SDN, OpenFlow, or MPLS, and AI/ML-based prediction models to proactively adapt to network conditions [28]. Additionally, the use of 5G and low-earth-orbit satellite networks (e.g., Starlink) has been investigated to reduce unpredictable jitter, while WebRTC has been tested for music performance despite challenges in achieving ultra-low latency [29].

An emerging paradigm is edge-based audio synthesis, transmitting control parameters (e.g., MIDI notes, envelopes) to edge devices for local synthesis, thus drastically reducing bandwidth. Neural audio synthesis techniques such as DDSP

TABLE I
AUDIO CODECS COMMONLY USED IN NMP SYSTEMS.

Audio Codec	Compression	Protocol	Added Latency	Audio Quality	Notes
PCM (Raw)	No	UDP	$\approx$0 ms	Transparent	Requires high, stable bandwidth (>1 Mbps per channel)
Opus	Yes	UDP/TCP	<1–5 ms (configurable)	Transparent at $\geq$64 kbps	Widely adopted; supports PLC and VBR
MP3/AAC	Yes	TCP	$\geq$100 ms	Good at higher bitrates	Not suitable for real-time NMP
MJPEG (video)	Yes	UDP	10–50 ms	-	Used for video in systems like LOLA

and NSynth reconstruct audio from latent embeddings at the edge, enabling low-bandwidth transmission with high fidelity [30]. Hybrid solutions combining compressed audio with control data (e.g., OSC messages) are also explored to enable real-time adaptive processing.

Table II compares traditional transmission methods with edge-based approaches in terms of latency and bandwidth.

Despite the wide range of techniques and architectures developed for Networked Music Performance, current solutions still rely primarily on the transmission of raw or compressed audio. These approaches, while effective under ideal network conditions, suffer from limitations in scalability, accessibility, and robustness, especially in constrained environments such as mobile or rural networks.

Moreover, although edge computing and neural synthesis have been recently explored in experimental settings, a practical, low-latency framework combining symbolic control transmission, real-time synthesis, and perceptual enhancement (e.g., autotune) at the edge is still missing in the literature. No open-source implementation or evaluation under realistic network conditions has yet demonstrated the feasibility and effectiveness of such an architecture for live collaborative music.

To fill this gap, we propose a novel end-to-end framework for NMP that abandons audio transmission entirely, leveraging symbolic control data and local edge-based synthesis enhanced with real-time audio effects.

III. RECONSTRUCTING THE SOUND AT THE EDGE: SYSTEM IMPLEMENTATION

In this section, we present the proposed new paradigm for NMP with real implementation, with the concept of exploiting the audio synthesis at the edge.

The proposed system leverages symbolic transmission for NMP, consisting of a lightweight client that transmits control data (MIDI or pitch) and a synthesis server that reconstructs the sound at the edge. The architecture enables low latency and robust interaction over IP networks, even under variable network conditions. Real-time pitch extraction from audio typically requires short buffering windows to achieve stable estimates, which may introduce additional latency that needs to be carefully evaluated in the context of NMP. For this reason, the present study focuses exclusively on the MIDI-capture approach, with audio-based symbolic extraction identified as a promising avenue for future research rather than as part of the current framework.

The system (see Fig. 1) comprises three main components:

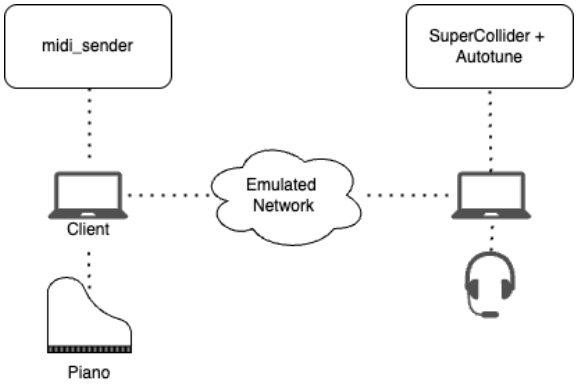

Fig. 1. General architecture of the proposed networked music performance system with symbolic data transmission and local autotune-enhanced synthesis.

1) **Client (Performer Node)**: Captures control data (MIDI or real-time pitch).
2) **Network Link**: Transmits symbolic data via UDP.
3) **Server (Synthesis Node)**: Receives symbolic data, maps it to audio synthesis, and plays the sound using a local sound engine.

Let us describe each component in detail, with its specific functionalities.

A. Client (Performer Node)

In our implementation, the client acts as a bridge between standard MIDI hardware and OSC-based synthesis at the edge.

Upon opening the MIDI port, the client enters a tight polling loop (with a 1 ms sleep interval) to continuously read MIDI message tuples, each comprising a 3-byte MIDI packet and its inter-arrival delta time. This polling ensures minimal latency at the expense of increased CPU usage.

Depending on the instrument type, the client captures either:

- MIDI messages from a USB keyboard using Python's `rtmidi` library.
- Pitch and amplitude values extracted from a microphone input using `aubio` or `CREPE`.

In our architecture, the client module is responsible for capturing incoming MIDI events from a local hardware controller and translating them into Open Sound Control (OSC) messages suitable for remote synthesis in SuperCollider [31]. The implementation is written in Python and leverages two main libraries: `rtmidi`for low-level MIDI I/O, and `python-osc` for UDP-based OSC transport. We chose `rtmidi` due to its stable, cross-platform bindings and low

TABLE II
COMPARISON BETWEEN TRADITIONAL AND EDGE-BASED TRANSMISSION.

Approach	Transmission	Edge Processing	Added Latency	Typical Bandwidth (per channel)
PCM Audio	Raw audio	No	<5 ms	≈1.4 Mbps (48 kHz, 16-bit mono)
Opus Audio	Compressed audio	No	<1–5 ms	64–128 kbps
MIDI/OSC + Synth	Control messages	Yes	<10 ms	<1 kbps
DDSP Embeddings	Latent vectors	Yes	<20–30 ms	≈20–50 kbps
Hybrid	Audio + control	Partial	10–30 ms	100–300 kbps

latency, and `python-osc` for its simplicity in creating and sending datagram packets without additional protocol overhead.

All outgoing OSC messages are sent to the address `/midi` and carry a two-element payload: a binary "note on" flag (1 for on, 0 for off), and the MIDI note number. The control data is serialized and sent via UDP packets to the synthesis server. We opted for UDP as the transport protocol due to its low overhead and lack of head-of-line blocking, which is especially beneficial in real-time contexts. However, this comes at the cost of reliability, packet loss may result in missed note events. Our implementation serves as the basis for designing simple logic on the server to infer likely state transitions in case of missed messages. Although the current implementation forwards raw MIDI note numbers, we also provide a utility function that can convert them into absolute frequencies (in Hz), simplifying server-side synthesis logic.

Through this client-side configuration, we establish a modular and extensible low-latency interface between MIDI controllers and networked synthesis engines. The architecture supports further enhancements in real-time reliability, dynamic peer discovery, and adaptive behavior under fluctuating network conditions.

B. Server-Side Configuration and Real-Time Sound Synthesis

On the server side, we deploy a real-time synthesis engine based on SuperCollider, a domain-specific language and platform optimized for low-latency audio rendering. A custom script listens for OSC messages on the `/midi` address and dynamically spawns and controls synth instances based on incoming events. The server maintains internal dictionaries to manage currently active notes and scheduled note termination tasks. The server operates as a reactive node that receives OSC messages and transforms symbolic note events into continuous-time audio signals.

Let us define the incoming OSC message as a discrete event tuple:

$$m(t) = (\text{on}_t, n_t) \in \{0, 1\} \times \mathbb{N}$$

where t denotes the reception timestamp, on_t indicates whether the event is a *note-on* (1) or *note-off* (0), and n_t is the MIDI note number. Upon receiving a note-on message, the system maps n_t to a fundamental frequency $f_t \in \mathbb{R}^+$ via the standard 12-TET mapping:

$$f_t = 440 \cdot 2^{(n_t - 69)/12}$$

This frequency f_t parametrizes a monophonic oscillator node governed by an amplitude envelope with controllable attack, decay, sustain, and release (ADSR) characteristics.

To ensure robustness against packet loss and missing note-off messages (common in UDP-based communication), each note-on event is associated with a timeout handler $\tau(n)$ that releases the envelope after a fixed interval Δt_{max}, e.g., 2 seconds:

$$\tau(n_t) = \begin{cases} \text{release } x(t), & \text{if no note-off within } \Delta t_{\text{max}} \\ \emptyset, & \text{otherwise} \end{cases}$$

This timeout mechanism ensures bounded resource allocation and prevents hanging notes in performance scenarios. If a note-off message is received before the timeout, both the synth and the scheduled task are terminated cleanly.

The server maintains two finite dictionaries:

- S: active synth instances, indexed by MIDI note.
- T: timeout tasks associated with each active note.

For each incoming $m(t)$, the server performs conditional updates on (S, T) according to a state-transition diagram driven by on_t and prior state. This structure avoids redundant synth creation and ensures graceful concurrent note handling.

The system is designed to be modular, with a clear separation between control logic (OSC handling and scheduling) and synthesis logic (signal generation). The current configuration supports basic monophonic synthesis with a sine oscillator, but it can be extended to:

- Multitimbral or polyphonic synthesis.
- Parameter mapping from continuous controller (CC) MIDI data.
- Timbre variation via more complex sound engines (FM, granular, sample-based).

This architecture enables rapid prototyping and composability, making it suitable for live NMP scenarios where responsiveness and adaptability are critical.

IV. EVALUATION AND CONSIDERATIONS

In this section, we present the implementation of the proposed framework and the preliminary results.

In particular, the real implementation is presented in a short demo available at this GitHub repository.

In the video, we can see the server that is receiving the notes sent by the keyboard via the client. The server (see Fig. 2) is able to reconstruct the sound and play the original song.

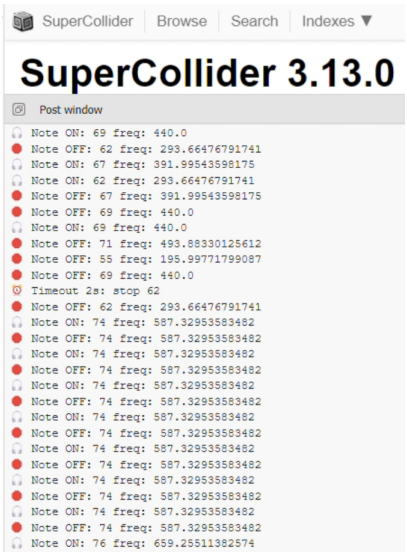

Fig. 2. SuperCollider at the edge

A. Quality of Experience Evaluation

In order to understand the feasibility of this new paradigm, we decided to give an initial baseline understanding of the user-perceived QoE under different network conditions by altering the network parameters between the client and server using the *Network Link Conditioner* tool. In particular, the packet loss configuration of Network Link Conditioner introduces independent random drops across packets (i.i.d. losses), without burst correlation. Specifically, we considered four distinct scenarios:

- Scenario 1: 0% packet loss, 0 ms additional delay.
- Scenario 2: 5% packet loss, 20 ms additional delay.
- Scenario 3: 15% packet loss, 40 ms additional delay.
- Scenario 4: Limited uplink bandwidth (200 kbps), 0% packet loss, 0 ms additional delay.

The network conditions in this study were emulated using the Network Link Conditioner tool, which allowed for reproducible scenarios with controlled levels of packet loss, delay, and bandwidth constraints. While this approach is valuable for preliminary testing, it is acknowledged that it cannot capture all characteristics of real-world networks (e.g., correlated loss bursts, cross-traffic, or jitter variability). Consequently, the presented results should be interpreted as indicative of the relative behavior of the systems under controlled impairments, rather than as a definitive measure of performance in live Internet conditions.

For each scenario, we conducted a Mean Opinion Score (MOS) survey involving 15 participants, including both musicians (8) and non-musicians (7), to assess their experience in using our proposed framework.

Each participant experienced both systems under each scenario: JackTrip implementation and our system with Sound Reconstruction at the Edge.

To ensure a fair comparison, the JackTrip benchmark was configured to transmit audio generated from the same symbolic source employed in our framework. Specifically, MIDI events produced by the USB keyboard were rendered into audio at the transmitter side using a software synthesizer, and the resulting PCM signal was streamed via JackTrip. In this way, both systems were evaluated under identical musical content, avoiding bias due to the use of different sound sources. In our evaluation, a single performer generated musical sequences using a USB MIDI keyboard (electronic keys), which were transmitted over the network through the proposed framework. The JackTrip configuration employed for all scenarios was: 48 kHz sampling rate, 16-bit resolution, mono channel, frame size of 128 samples, default jitter buffer length, and without Forward Error Correction. Packet losses were handled by the standard JackTrip implementation. The evaluation task consisted of reproducing short musical phrases under the different network conditions.

After playing for a short time, the participants were asked to rate their experience using a standardized 5-point MOS scale (1 = Bad, 5 = Excellent) across the following questions:

1) **Audio Quality:** "How clear and natural was the audio you heard?"
2) **Delay:** "Did the delay of the network impact on your experience ?"
3) **Overall Experience:** "How satisfied were you with the session overall?"

The final MOS score per system and scenario was calculated as the average over all participant responses and all three question rates:

$$\text{MOS} = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{1}$$

Table III presents the results comparing JackTrip and our system across the four network conditions.

TABLE III
DETAILED MOS SCORES BY USER CATEGORY AND NETWORK SCENARIO

Scenario	Musicians		Non-Musicians		Avg. Diff
	JackTrip	Ours	JackTrip	Ours	
1 (Ideal)	4.9	4.7	4.7	4.5	-0.1
2 (5% loss, 20ms)	3.2	3.4	4.0	3.6	+0.1
3 (15% loss, 40ms)	2.6	2.8	3.4	3.0	+0.1
4 (200 kbps limit)	1.1	3.9	1.5	3.7	+2.6
Average	2.95	3.7	3.4	3.7	+0.425

In ideal conditions, both systems performed well, with JackTrip slightly outperforming our solution due to its use of raw, uncompressed audio. However, in network-degraded conditions, our system maintained significantly higher QoE thanks to edge-based audio reconstruction and adaptive bitrate compression. Notably, participants reported fewer perceived glitches and improved temporal stability in scenarios with constrained upload bandwidth.

Musicians were generally more sensitive to timing and synchronization issues, particularly in Scenarios 2 and 3.

These results demonstrate that edge intelligence can provide a meaningful advantage in maintaining performance quality in constrained or lossy environments.

Musicians were found to be more sensitive to latency and packet loss than non-musicians, especially in terms of synchronization and rhythmic precision. These results highlight the advantage of using edge-based audio reconstruction and bitrate adaptation in degraded network environments, such as low-upload connections or satellite internet links.

Finally, it should be emphasized that the evaluation was conducted with a specific class of instrument, namely electronic keys. As such, the reported results are directly related to this type of symbolic transmission and sound reconstruction. Results may differ for other instruments, particularly those producing continuous pitch (e.g., violin, voice) or complex timbral dynamics (e.g., drums, guitar). In these cases, the perceptual impact of packet loss and synthesis artifacts may be more pronounced. Future work will extend the evaluation to a wider set of instruments to better capture the generalizability of the proposed approach.

B. Average Missing Notes Analysis

To conclude the evaluation of the proposed framework, we sent MIDI note events over the client-server UDP connection in the four distinct scenarios presented earlier.

A total of 500 MIDI notes were sent per trial, with each note lasting 100 ms. The receiving server logged the incoming notes and compared them against the expected sequence to identify missing events. Each scenario was repeated 10 times to ensure statistical significance.

In scenario 1, the system achieved near-perfect delivery, with an average of 0.2 missing notes per trial (standard deviation $\sigma = 0.1$).

In scenario 2 (5% packet loss and 20 ms delay), the mean number of missing notes increased to 1.8 ($\sigma = 0.5$). Losses were attributed primarily to packet drops, as expected.

Scenario 3 introduced severe conditions with 15% packet loss and 40 ms delay. Here, reliability degraded substantially, with an average of 4.6 missing notes per trial ($\sigma = 1.2$). The wider variance suggests that high jitter and compounded losses significantly affect delivery consistency.

The small mean differences observed in Scenarios 1–3 (on the order of 0.1 on a 5-point scale) should therefore not be interpreted as evidence of superiority of one system over the other.

In Scenario 4, bandwidth was artificially constrained to 200 kbps, with no explicit packet loss or delay. It is crucial to note that JackTrip transmits uncompressed PCM audio at approximately 1.4 Mbps per monophonic channel (48 kHz, 16-bit), whereas our symbolic transmission requires less than 1 kbps on average for MIDI note events, with occasional bursts during fast passages. Consequently, the bandwidth-constrained scenario at 200 kbps represents a severe bottleneck for JackTrip, but it has a negligible impact on our proposed solution. While JackTrip represents a reference-quality baseline in Scenarios 1–3, in Scenario 4, it effectively behaves as a low anchor. For this reason, the difference in MOS ratings between the two systems in Scenario 4 cannot be directly compared to the differences in Scenarios 1–3. Rather, the key insight is that our proposed framework continues to operate acceptably under conditions in which PCM-based NMP streaming is no longer feasible. This explains the large gap observed in the MOS results under Scenario 4, highlighting one of the key advantages of symbolic transmission combined with local synthesis at the edge. Despite the absence of artificial impairments, the system experienced a mean of 2.1 missing notes ($\sigma = 0.7$), likely due to congestion-induced losses during data bursts. This highlights the sensitivity of symbolic transmission to available throughput.

Fig. 3. Distribution of Missing Note Events per Scenario

Fig. 4. Average Missing Notes Per Scenario with Std Dev

Overall, the results indicate that symbolic musical data transmitted via UDP remains robust in bandwidth-limited environments but becomes increasingly unreliable in high-loss scenarios.

Fig. 3 and Fig. 4 summarize the results obtained in this part of the evaluation.

V. Conclusion and Future Work

The framework proposed in this study represents a new approach in the field of NMP, aiming to overcome the limita-

tions of traditional systems based on the transmission of raw audio. By transmitting symbolic data (such as MIDI or pitch) and synthesizing the audio locally at the destination node, the system allows for the relaxation of required bandwidth requirements and sensitivity to packet losses, while ensuring a high-quality music experience even under suboptimal network conditions.

The edge-based architecture, combined with real-time audio effects such as autotune, demonstrates that it can improve users' perceived QoE, as evidenced by the MOS survey results. In particular, the system maintains superior performance compared to solutions such as JackTrip in scenarios with packet loss or limited bandwidth, making it suitable for real-world settings, including rural or mobile.

The availability of open-source code and the modularity of the framework pave the way for future optimizations, such as the integration of more advanced synthesizers, polyphonic support, and dynamic adaptation to network conditions. This innovative approach not only addresses current NMP challenges but also lays the foundation for an evolution toward a musical cloud continuum, where distributed processing and artificial intelligence can further enhance remote artistic collaboration.

Future work will extend the proposed framework to additional instruments, incorporate real-time effects such as autotune, and validate performance over real-world Internet conditions. While pitch tracking remains promising, it requires audio buffering for accurate frequency estimation, which may increase end-to-end latency beyond typical NMP requirements. Finally, future research will address this trade-off by testing low-latency pitch extractors and evaluating their perceptual impact in real-time performance.

ACKNOWLEDGMENT

The work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 — program "RESTART") and by the Italian Ministry of Education and Research (MUR) in the framework of the FoReLab and CrossLab projects (Departments of Excellence).

REFERENCES

[1] B. Li, X. Yuan, Y. Hu, and A. Schmeink, "Transmission latency minimization in full-duplex relaying network operating with finite blocklength codes," *IEEE J.Sel. A. Commun.*, vol. 43, no. 4, p. 1168–1182, Jan. 2025. [Online]. Available: https://doi.org/10.1109/JSAC.2025.3531543

[2] K. Tsioutas and G. Xylomenos, "Assessing the effects of delay to nmp via audio analysis," *SN Computer Science*, vol. 4, no. 2, p. 126, 2022.

[3] 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," 10 2022.

[4] A. F. Khalifeh, A.-K. Al-Tamimi, and K. A. Darabkh, "Perceptual evaluation of audio quality under lossy networks," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2017, pp. 939–943.

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

[6] J.-M. Valin, J. Büthe, A. Mustafa, and M. Klingbeil, "Dred: Deep redundancy coding of speech using a rate-distortion-optimized variational autoencoder," 2024. [Online]. Available: https://arxiv.org/abs/2212.04453

[7] M. Fink and U. Zölzer, "Low-delay error concealment with low computational overhead for audio over ip applications." in *DAFx*, 2014, pp. 309–316.

[8] M. Sacchetto, Y. Huang, A. Bianco, and C. Rottondi, "Using autoregressive models for real-time packet loss concealment in networked music performance applications," in *Proceedings of the 17th International Audio Mostly Conference*, ser. AM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 203–210. [Online]. Available: https://doi.org/10.1145/3561212.3561226

[9] M. Bosi, A. Servetti, C. Chafe, and C. Rottondi, "Experiencing remote classical music performance over long distance: a jacktrip concert between two continents during the pandemic," *Journal of the Audio Engineering Society*, vol. 69, no. 12, pp. 934–945, 2021.

[10] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via lola: Low latency high quality a/v streaming system," in *International conference on information technologies for performing arts, media access, and entertainment*. Springer, 2013, pp. 240–250.

[11] 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, 2020.

[12] A. Carôt and C. Werner, "Distributed network music workshop with soundjack," *Proceedings of the 25th Tonmeistertagung, Leipzig, Germany*, p. 5, 2008.

[13] A. MONTALI and L. GHIȘA, "Digital innovations for distance music education: The imsv project and the application of jamulus software." *ICT in Muzical Field/Tehnologii Informatice si de Comunicatie in Domeniul Muzical*, vol. 15, no. 2, 2024.

[14] N. Harrison and G. Parton, "New technology and approaches for audio education as a response to covid-19," *The Australasian Sound Archive*, no. 45, pp. 82–107, 2022.

[15] P. Ferguson, "Real-time long-distance music collaboration using the internet." Future Technology Press, 2016.

[16] L. Turchet and C. Fischione, "Elk audio os: an open source operating system for the internet of musical things," *ACM Transactions on Internet of Things*, vol. 2, no. 2, pp. 1–18, 2021.

[17] P. Neekhara, C. Donahue, M. Puckette, S. Dubnov, and J. McAuley, "Expediting tts synthesis with adversarial vocoding," 2019.

[18] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," 2019.

[19] L. Li, Wudamu, L. Kürzinger, T. Watzel, and G. Rigoll, "Lightweight end-to-end speech enhancement generative adversarial network using sinc convolutions," *Applied Sciences*, vol. 11, no. 16, 2021.

[20] C. Hernandez-Mejia, X. Ren, A. Thabuis, J. Chavanne, P. Germano, and Y. Perriard, "Generative adversarial networks for localized vibrotactile feedback in haptic surfaces," in *2021 24th International Conference on Electrical Machines and Systems (ICEMS)*. IEEE Press, 2021, p. 105–110.

[21] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," 2019.

[22] L. Orynbay, B. Razakhova, P. Peer, B. Meden, and Emeršič, "Recent advances in synthesis and interaction of speech, text, and vision," *Electronics*, vol. 13, no. 9, 2024.

[23] D. Wagner, I. Baumann, and T. Bocklet, "Generative adversarial networks for whispered to voiced speech conversion: a comparative study," *International Journal of Speech Technology*, vol. 27, no. 4, pp. 1093–1110, 2024.

[24] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 022–17 033.

[25] S.-H. Lee, J.-H. Kim, K.-E. Lee, and S.-W. Lee, "Fre-gan 2: Fast and efficient frequency-consistent audio synthesis," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6192–6196.

[26] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet:

A generative model for raw audio," in *9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[27] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3617–3621.

[28] M. Bagaa, D. L. C. Dutra, T. Taleb, and K. Samdanis, "On sdn-driven network optimization and qos aware routing using multiple paths," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4700–4714, 2020.

[29] B. Sredojev, D. Samardzija, and D. Posarac, "Webrtc technology overview and signaling solution design and implementation," in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2015, pp. 1006–1009.

[30] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A review of differentiable digital signal processing for music and speech synthesis," *Frontiers in Signal Processing*, vol. 3, p. 1284100, 2024.

[31] J. McCartney, "Rethinking the computer music language: Super collider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.