Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(37^{th}cycle)

Towards Accessible and Inclusive Remote Musical Interactions

Challenges and Advancements in Networked Music Performances

By

Matteo Sacchetto

******

Supervisor(s):
Prof. Cristina Rottondi, Supervisor
Prof. Andrea Bianco, Co-Supervisor

Doctoral Examination Committee:
Prof. Luca Turchet, Università degli studi di Trento, *Referee*
Prof. Chrisoula Alexandraki, Hellenic Mediterranean University, *Referee*
Prof. Alexander Carôt, Hochschule Anhalt
Prof. Augusto Sarti, Politecnico di Milano
Prof. Paolo Giaccone, Politecnico di Torino

Politecnico di Torino
2025

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Matteo Sacchetto
2025

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To my beloved nonni

"Music is the universal language of mankind"

— Henry Wadsworth Longfellow

Acknowledgements

First and foremost, I would like to thank Cristina for giving me the opportunity to undertake this Ph.D. journey. Her availability and guidance throughout this Ph.D. have been invaluable, and her support and assistance were particularly appreciated, especially during my occasional mini "panic" moments.

Secondly, I would like to thank Andrea for his support and for his extensive knowledge. His constructive feedback was extremely valuable throughout this research and consistently provided me with a different perspective on challenges.

Together, Cristina and Andrea formed an exceptional supervisory team, and I am truly grateful for their combined expertise and dedication. I could not have asked for better mentors to guide me through this research.

Then I would like to thank Leonardo Severi, a friend since our undergraduate days at Politecnico di Torino. Sharing the Ph.D. experience with him and working together on Networked Music Performance in the same office made this journey much more memorable. I also fondly remember our creative singing sessions and our ongoing ping pong rivalry (the final tally remains to be determined!). Moreover, I am deeply grateful to him for co-authoring most of the work presented in this thesis. This thesis would not have been possible without his invaluable collaboration.

I would like then to thank Antonio Servetti. He was my Master's degree supervisor, and he was the person who made me discover the field of Networked Music Performance. I would probably not be here if it wasn't for him.

Then there are my parents, who I want to thank for their constant love, support, and encouragement throughout my studies. Thank you for always encouraging me and for believing in me, especially when things got tough.

Many thanks to my brother, Marco, for his constant support and encouragement throughout this journey. I also appreciate his interest in my research and his will-

ingness to always lend an ear (even if he didn't always understand the technical details!).

Finally, I want to thank all the wonderful people who accompanied me throughout this Ph.D. journey. First off, German Sviridov and Riccardo Peloso, who I had the pleasure of collaborating with before starting this adventure. They were helpful in convincing me to start this Ph.D. journey (and yes, you could say they were a bit crazy, but in the best way possible!). Then there are Franco Galante and Alessandro Cornacchia, the first two Ph.D. students I met. Our chats, idea-sharing, and endless Ph.D.-related discussions were a constant source of sanity (or maybe shared insanity?). I then want to thank Pietro Buccellato, another colleague with whom I had the pleasure of working with during this Ph.D. He also works on Networked Music Performance, but came from an electronic background. While working together, we had the opportunity of sharing our knowledge and learning from each other (even though I was unable to convince him to use git). I also want to thank Paolo Gastaldi, Alessandro de Vitis, Luigi Pirisi, and all the research fellows who joined me for a part of this journey! All of you were incredibly helpful with all the development aspects. To all the thesis students who I had the distinct pleasure of working with (sorry if you had to put up with me and my sense of humor!). The researchers I had the opportunity to meet at various conferences also deserve a big thank you for making each event genuinely enjoyable. A big thanks to all the other professors of this and other universities I had the chance to meet and who shared their knowledge with me during this Ph.D. And last but definitely not least, a huge thank you to all my friends who kept me (somewhat) sane and supported me through these years (thanks for bearing with my endless complaints!).

Abstract

Networked Music Performance (NMP) refers to a real-time interaction between geographically distant musicians, enabled by solutions for audio streaming over a telecommunication network, that allows them to play together as if they were physically located in the same room. While the concept has roots in early technological experiments, its importance has intensified in recent years, driven by advancements in the Internet infrastructure. These advancements include the widespread availability of higher bandwidth, reduced latency, and more reliable network connectivity, which are crucial for supporting the demands of real-time audio streaming. Furthermore, the recent global pandemic and its associated restrictions on physical gatherings have dramatically highlighted the importance and potential of NMP technologies.

Commonly used communication platforms often fall short on replicating the experience of a co-located performance. This is partly due to the reliance on general-purpose tools, such as videoconferencing applications, which, while user-friendly, typically suffer from network latency and limitations in audio quality, detrimental to musical interactions. Conversely, dedicated NMP solutions, while addressing the performance requirements of low-latency and high audio fidelity, are often perceived as complex to use, particularly for individuals with diverse technical backgrounds or specific accessibility needs.

This thesis attempts to address these limitations by presenting a comprehensive investigation on how to enhance NMP systems, exploring innovative solutions and strategies designed to promote more accessible, high-quality, and user-friendly remote musical interactions. This research involves the development and evaluation of web-based NMP systems, specifically by investigating an alternative, unconventional use of WebRTC technologies to achieve low-latency, high-fidelity (uncompressed) audio streaming. Furthermore, it explores the design of hybrid web-native architectures, aiming to integrate user-friendly web interfaces with the improved performance

of dedicated native applications. Moreover, the thesis details the investigation of dedicated hardware acceleration for NMP through the design and implementation of an FPGA-based system intended to minimize audio processing delays. Advanced techniques for packet loss concealment are also investigated, focusing on the application of autoregressive models to enhance the robustness of audio transmission over the network. To support the broader NMP research community, the thesis also presents the collection process of a specialized dataset, named DUST, of real-world NMP network traffic, carefully designed to facilitate the development and evaluation of NMP-specific algorithms. The potential of immersive technologies is also explored through the investigation of a Virtual Reality system for remote orchestral conduction, as a new approach to provide visual feedback to musicians. Finally, the thesis places a strong emphasis on inclusivity in music education, detailing the investigation of cross-modal associations to design accessible interfaces and providing a comprehensive analysis of technological solutions for users with diverse abilities.

Contents

List of Figures **xv**

List of Tables **xix**

Nomenclature **xx**

1 Introduction **1**

1.1 Networked Music Performance 1

1.2 Motivation and Aims 6

1.3 Outcomes 9

1.4 Thesis Structure 12

2 Background and Related Work **16**

2.1 Background 16

2.1.1 Networking Fundamentals 16

2.1.2 Digital Audio Fundamentals 27

2.1.3 WebRTC Fundamentals 31

2.1.4 Time Series Fundamentals 34

2.1.5 Accessibility Fundamentals 42

2.2 Related Work 43

2.2.1 Networked Music Performance Systems 43

2.2.2 WebRTC 48
2.2.3 Packet Loss Concealment 49
2.2.4 Visual Feedback in Networked Music Performance 51
2.2.5 Haptic and Visual Methods for Deaf and Hard of Hearing Subjects 52

3 Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution 54

3.1 Introduction 55
3.2 Limitations of Videoconferencing Tools in an NMP Scenario 57
3.3 The Proposed Low-Latency WebRTC Application 60
3.4 Latency Measurements 63
3.4.1 Experimental Settings 63
3.4.2 Overall Latency 65
3.5 Audio Chain Latency Assessment 66
3.6 Application Robustness 69
3.6.1 Jitter Measurements 69
3.6.2 Scalability Evaluation 70
3.7 Summary 71

4 Networked Music Performance: A Unified Native and Web Approach 73

4.1 Introduction 74
4.2 State of the Art 74
4.3 Architecture 76
4.3.1 Server 77
4.3.2 Client 78
4.4 Results 81
4.4.1 Audio Quality 81

4.4.2 M2E Latency 85
4.5 Summary 86

5 FPGA-Based Low-Latency Audio Coprocessor for Networked Music Performance 87
5.1 Introduction 88
5.2 Background 89
5.3 System Description 90
5.3.1 Audio Board 91
5.3.2 Audio Processor 93
5.3.3 Raspberry Pi 99
5.4 Results 99
5.4.1 FPGA Utilization 99
5.4.2 FPGA Power Consumption 101
5.4.3 SPI Bandwidth and Protocol Overhead 102
5.5 Summary 103

6 Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications 104
6.1 Introduction 105
6.2 Background 107
6.2.1 Autoregressive Models 107
6.2.2 Stationarity Test 108
6.2.3 Burg's Method 109
6.3 Algorithmic Framework 111
6.3.1 Notation 111
6.3.2 AR Model Implementations 112
6.3.2.1 Traditional Burg's Method 112

6.3.2.2 Denominator Optimization 114
6.3.2.3 Hybrid Denominator 116
6.3.2.4 Error Compensation 117
6.4 Experimental Validation 118
6.4.1 Experimental Setup 118
6.4.2 Implementation Details 119
6.4.3 Dataset 121
6.4.4 Performance Metrics 121
6.4.5 Stationarity Test 122
6.4.6 Stationarity Test Results 123
6.4.7 Fitting and Validating the AR Models 124
6.4.8 Evaluation of the Prediction Error 126
6.4.9 Evaluation of Computational Requirements 132
6.4.10 Evaluation of the Prediction Error on Long Predictions . . . 134
6.4.11 Qualitative Analysis 137
6.4.12 Qualitative Analysis Results 137
6.5 Limitations 140
6.6 Summary 143

7 Introducing DUST: a Dataset of Real-Time UDP Sound Packet Traces 144
7.1 Introduction 144
7.2 The Dataset 147
7.2.1 Data Collection Method 147
7.2.2 Dataset Structure and Content 148
7.3 Information Extraction and Utilization 151
7.3.1 Clock Drift and Jitter Characterization 151
7.3.2 Sample Loss Simulation 152

7.4 Summary 155

8 Remote Orchestral Conduction via a Virtual Reality System 156

8.1 Introduction 156

8.2 System Overview 158

8.2.1 Transmission Setup 159

8.2.2 Framework Description 160

8.3 Evaluation Methodology 162

8.4 Results 163

8.4.1 Testbed Description 163

8.4.2 Transmission Jitter 163

8.4.3 Messages RTT Measurements 164

8.4.4 M2P Latency of Hand Movement 165

8.4.5 Bit Rate Requirements 165

8.5 Limitations 166

8.6 Summary 166

9 Collection of Design Directions for a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students 168

9.1 Introduction 169

9.2 Background 170

9.2.1 Basic Notions on DHH Impairments 170

9.2.2 Attributes for Music Characterization 171

9.2.3 Programmable Graphical Interface with Haptic Feedback . . 173

9.3 Exploratory Study 174

9.3.1 Choice of Cross-Modal Associations 174

9.3.2 Visual Presentation 178

9.4 Implementation Details 178

9.5 Summary 181

10 Towards an Inclusive Framework for Remote Musical Education and Practices 183

10.1 Introduction 184

10.2 Accessible Human-Machine Interfaces for Musical Education 186

10.2.1 Educational and Performative Issues Experienced by Disabled Users in Music-Related Fields 186

10.2.2 Haptic Feedback Techniques 187

10.2.3 Motion Tracking Techniques 188

10.2.4 Immersive Audio Rendering 189

10.2.5 Exemplary Field Trials 190

10.3 NMP Applications and Backend Infrastructures 190

10.3.1 Main Functionalities of NMP Systems 190

10.3.2 Peer-to-Peer VS Client-Server Architecture 191

10.4 Towards an Inclusive Framework for NMP and Remote Musical Education 192

10.4.1 Camera Feedback and Motion Sensor Interface Control . . . 193

10.4.2 Sensors/Actuators for Remote Haptic Signals 194

10.4.3 Accessible GUI 195

10.4.4 Remote Mixing Capabilities 195

10.4.5 AR/VR Environments and Spatialized Audio Rendering . . 195

10.4.6 5G Connectivity 195

10.4.7 Summary 196

10.5 Use Cases 196

10.5.1 Accessible GUI 197

10.5.2 Accessible Metronome 199

10.5.3 Tactile Feedback for Timbre Discrimination 201

10.6 Technical Challenges 202
10.6.1 Latency Management and Packet Loss Concealment for Cross-Modal Data Streams 202
10.6.2 Interoperability and Standardization 203
10.6.3 Minimizing Network Contributions to Mouth-to-Ear Latency 204
10.7 Towards an Inclusive Musical Metaverse 205
10.8 Summary 207

11 Conclusions **208**

11.1 Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution 208
11.2 Networked Music Performance: A Unified Native and Web Approach 210
11.3 FPGA-based Low-Latency Audio Coprocessor for Networked Music Performance 211
11.4 Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications 212
11.5 Introducing DUST: a Dataset of real-time UDP Sound packet Traces 213
11.6 Remote Orchestral Conduction via a Virtual Reality System 214
11.7 Collection of Design Directions for a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students . . 215
11.8 Towards an Inclusive Framework for Remote Musical Education and Practices 215
11.9 Conclusive Remarks 216
11.10 Future Works 217

References **220**

List of Figures

2.1 Traditional WebRTC application structure. 33

3.1 Distortions due to AGC (top) and perceptual codec compression (bottom) in Google Meet and Jitsi, running in Chrome v.98. The blue line represents the original signal, while the red line represents the processed signal, in both the time and frequency domains. The audio track is a sample of a bowed violin playing quietly (-12 dB) at the B4 pitch. 58

3.2 M2E latency of common videoconferencing software, both native and web-based (Chrome v.98), between two MacBooks connected to the same Ethernet switch. When available, the LL software configuration without audio processing was selected. For each configuration, ten sessions of the duration of one minute have been measured. . . . 59

3.3 Custom WebRTC application structure with the `RTCDataChannel` and the `Shared Array Buffer` (SAB). 60

3.4 M2E measurement process. The measurement device computes the delay as the time difference between the time it emits the test signal and the time it receives it back from the full audio chain, which includes the audio interfaces (here represented as the rectangular objects), the peers and the network transmission between them. . . . 64

3.5 M2E latency measurements with different browsers and OSes for the two approaches based on the `RTCPeerConnection` and the `RTCDataChannel`. 66

3.6 M2E latency measurements for different setups of the audio chain with Chrome and Firefox on different operating systems. Labels correspond to the setups (A, B, C) described in Sec. 3.5 either with the MC (0) or the SAB (1). 67

3.7 Inter-Frame Delay Variation (IFDV) with Chrome and Firefox on macOS at the sender, after acquisition from the input device, and at the receiver, after reception from the network layer. Both sender and receiver are on the same device in order to have the same time reference. 70

3.8 Percentage of lost audio frames, i.e., discarded because too late, as a function of the number of concurrent peers, when MC or SAB are used. 71

4.1 Architecture with server components and the two types of clients (web and native) unified through the use of the same web-based GUI. 77

4.2 Demonstration of the system usage, showcasing the unified web-based GUI (purple window) and the native solution (*box*) based on the Raspberry Pi 4B (bottom left). 79

4.3 GUI 80

4.4 The first row shows the time and frequency domain representation of an audio segment (bowed violin). The following rows show the distortions introduced by the proposed audio transmission framework and by three state-of-the-art videoconferencing solutions, mainly due to perceptual codec compression and automatic digital signal processing applied to the original signal. The plots also indicate the SNR and the LSD computed between the original and the processed signal for the proposed framework and then for Zoom (with the original audio feature turned on), Jitsi (with audio processing disabled) and Google Meet. 83

4.5 Results of the subjective audio quality test. Four stimuli: female soprano (Female NV), female soprano with vibrato (Female V), cello (Cello), and bowed violin (Violin), processed by four different systems: the proposed one (P), Zoom (Z), Jitsi (J), and Google Meet (M), have been presented to 26 subjects. The statistical significance threshold is 17. 84

4.6 M2E of the various analyzed solutions, including native implementations running on a Raspberry Pi 4B (left) and web implementations running on macOS (right) using different web browsers: Mozilla Firefox v.97 (FF) and Google Chrome v.98 (CH). 85

5.1 Simple TTA architecture 89

5.2 System block diagram 91

5.3 Custom audio board 92

5.4 Audio processor top-level block diagram 94

5.5 TTA core: FUs and interconnections 96

5.6 Flow diagram of the firmware routine 98

5.7 FPGA resource utilization 100

5.8 FPGA power consumption contributions 101

6.1 Stationarity 125

6.2 Performance of AR models in terms of MAE 129

6.3 Performance of AR models in terms of RMSE 130

6.4 Visual comparison of the reconstructed waveform by the three PLC techniques 131

6.5 AR models - computational time trend. 133

6.6 AR models - long prediction MAE 136

6.7 Qualitative analysis results 139

6.8 Resulting waveform for AR-based predictions of 2560 samples (20 packets of 128 samples) of a 2 kHz sine wave 141

7.1 Sender process diagram 148

7.2 Receiver process diagram 148

7.3 Clock drift and packet delay variation in the `sanri_12` capture . . . 152

7.4 Example flow-chart for a sample loss simulator, given a file of DUST and parameters B (buffer size) and C (samples per packet). An output value of zero marks a lost sample. 154

7.5 Example of the evolution of $W_i - R_i$ over time, assuming $B = 128$. Negative values indicate the presence of lost samples. 155

8.1 The conductor's avatar pointing at *James*, seen from the conductor's perspective. 158

8.2 The same scenario of Fig. 8.1 seen from the perspective of the four musicians 159

8.3 System overview 161

8.4 Distributions of message inter-arrival times for the three transmission stacks. The ideal value should be ~ 14 ms $= 1000/72$ fps. 163

8.5 M2P latency of the whole chain. 165

9.1 Black and white image rendering for haptic texture design 175

9.2 Listening Test Results 177

9.3 Screenshots of the developed interface 179

10.1 The proposed inclusive NMP framework for remote music education and practice 193

10.2 MEVO's Accessible GUI - Rooms page 199

10.3 Metronome GUI example 200

10.4 Example of the layout of the custom HW metronome bracelet 201

List of Tables

6.1 Time (ms) required to fit the specified algorithm on a training set of size n, with p set to 64 and data type set to `double` 134

6.2 Time (ms) required to fit the specified algorithm on a training set of size n, with p set to 128 and data type set to `double` 134

6.3 Time (ms) required to predict the next n_{test} (128) samples, with p set to 64 and 128 and data type set to `double` 135

9.1 Listening Test Results - Summary - For each instrument and category, we listed the two or three most significant associations. The associations are sorted by significance. When two options received the same number of responses, they are presented using the format A/B to indicate equal ranking. 176

10.1 Overview of Educational Issues of Disabled Music Students 186

10.2 Inclusive Features and Benefits 197

Nomenclature

Acronyms / Abbreviations

ACF	Autocorrelation Function
ADC	Analog-to-Digital Converter
ADF	Augmented Dickey-Fuller
AGC	Automatic Gain Control
AI	Artificial Inteligence
AIC	Akaike Information Criterion
ALU	Arithmetic Logic Unit
ANS	Adaptive Noise Suppression
API	Application Programming Interface
AR	Autoregressive
AR/VR	Augmented Reality/Virtual Reality
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
ASIP	Application Specific Instruction set Processor
BRAM	Block RAM
BVI	Blind and Visually Impaired

C/S	Client-Server
CRC	Cyclic Redundancy Check
CVD	Color Vision Deficiency
DAC	Digital-to-Analog Converter
DAW	Digital Audio Workstation
DHH	Deaf and Hard-of-Hearing
DL	Deel Learning
DSP	Digital Signal Processing
EC	Echo Cancellation
ECC	Error-Correcting Code
FPGA	Field Programmable Gate Array
FU	Functional Unit
GCU	Global Control Unit
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HDL	Hardware Description Language
HRTF	Head-Related Transfer Function
I2S	Inter-IC Sound
ICT	Information and Communication Technologies
IoMusT	Internet of Musical Things
ISP	Internet Service Provider
LAN	Local Area Network
LED	Light-Emitting Diode

LSD	Logarithmic Spectral Distance
LSU	Load-Store Unit
LUT	Lookup Table
LUTRAM	Lookup Table RAM
M2E	Mouth-to-Ear
M2P	Motion-to-Photon
MA	Moving Average
MEC	Multi-Access Edge Computing
MIDI	Musical Instrument Digital Interface
ML	Machine Learning
MM	Musical Metaverse
MusT	Musical Things
NFT	Non-Fungible Token
NFV	Network Function Virtualization
NMP	Networked Music Performance
P2P	Peer-to-Peer
PACF	Partial Autocorrelation Function
PLC	Packet Loss Concealment
PLL	Phase Locked Loop
QoE	Quality of Experience
QoS	Quality of Service
RTL	Register Transfer Level
RTP	Real-time Transport Protocol

RTT	Round-Trip Time
SDK	Software Development Kit
SDN	Software Defined Networking
SMO	Service Management and Orchestration
SNR	Signal-to-Noise Ratio
SPI	Serial Peripheral Interface
SPI	Universal Serial Bus
TCP	Transmission Control Protocol
TTA	Transport Triggered Architecture
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
VoIP	Voice over IP
VR	Virtual Reality
WAN	Wide Area Network
WPF	Windows Presentation Foundation
WSS	Wide-sense Stationarity
XAML	Extensible Application Markup Language
XR	Extended Reality

Chapter 1

Introduction

This thesis focuses on Networked Music Performances (NMPs), i.e., remote musical collaborations where performers are placed in different physical locations connected via the Internet or other telecommunication networks, allowing them to play music together in real-time [1]. This introductory chapter begins with an overview of the topic, its historical evolution, and the essential system requirements for NMP systems in Sec. 1.1. Following this, the motivation behind the research work and its objectives are presented in Sec. 1.2. Next, the main contributions and outcomes, including associated publications and demonstrations, are summarized in Sec. 1.3. Finally, the overall organization of the thesis is outlined in Sec. 1.4.

1.1 Networked Music Performance

According to [2], one of the earliest experiments that can be considered a precursor of NMP is John Cage's composition "Imaginary Landscape No. 4 for twelve radios". This composition "used radio transistors as a musical instrument. The transistors were interconnected, thus influencing each other" [3]. Only with the development and commercialization of personal computers (1970s), more realistic networked interactions were experimented. In particular, in 1977-78, Jim Horton and Rich Gold experimented with creating music through telephone lines [4]. This early exploration highlighted the artistic potential of connecting performers through existing communication networks.

While there were other experiments which tried to exchange real-time data, in the form of MIDI information [5], over the network [1, 6], it is only with the development of high-speed overprovisioned backbones, such as the Internet2, at the beginning of the year 2000s that NMP became feasible in the sense that we know it today. Previously to the introduction of such infrastructures, the World Wide Web was already used to exchange audio signals in real-time in Voice over IP (VoIP) communication systems. Though, it was not yet powerful enough to send high-quality full-band real-time audio signals. With the introduction of such infrastructure, it was now finally possible to create real-time high-quality audio connections [2]. One of the first experimenters, which took advantage of this powerful network backbone, was carried out by the SoundWIRE group [7], at Stanford's CCRMA led by Chris Chafe. They are the group that developed JackTrip [8], a powerful software which allows the streaming of high-quality uncompressed bidirectional audio. Throughout the first decade of the 2000s, other solutions began to be developed. Some notable ones are Soundjack (2006) [9, 10], Jamulus (2006) [11] and DIAMOUSES (2008) [12]. During the same years, some research groups also started experimenting with the transmission of low latency uncompressed video alongside audio. This resulted in the development of the McGill Ultra-Videoconferencing System (2001) [13] by the McGill Ultra-Videoconferencing Research Group and LOLA (2005) [14] created by the "Conservatorio di Musica Giuseppe Tartini" from Trieste (Italy) in collaboration with the GARR Consortium.

While, during these years, NMP was gaining increasing attention, it was still a niche field, primarily explored within academic and experimental music circles. In contrast, videoconferencing/VoIP technologies have already achieved widespread adoption for general communication purposes, ranging from business meetings to social interactions. In 2020, due to lockdown measures imposed globally to contain the widespread SARS-CoV-2 pandemic [15], videoconferencing platforms became the primary means of communication for work, education, and social interaction. However, these platforms showed significant limitations when it came to real-time musical collaboration, such as high latency and low audio fidelity, which are critical for musical performances [16–19]. As a result, NMP saw increasing interest, with musicians and music schools exploring dedicated tools and platforms like Jamulus [20] and JackTrip [8] to overcome these challenges. During that time, a plethora of NMP systems were developed, each trying to explore different approaches to NMP to provide musicians with a tool that would fit their use case [21–23]. Nonetheless,

often even in the musical context, especially music education, the videoconferencing platforms were still preferred over NMP specific solutions due to the simplicity in their setup, the familiarity in their use, the widespread diffusion and higher perceived reliability [24, 25]. For a comprehensive overview of the history of NMP, the reader is referred to [2, 26, 27].

Nowadays, NMP has become one of the building blocks of a broader ecosystem: the Internet of Musical Things (IoMusT) [28]. IoMusT is an emerging research field that extends the Internet of Things (IoT) specifically to the domain of music. It involves the integration of various technological components, such as sensors, actuators, connectivity solutions, and software, to create a networked ecosystem where musical devices, instruments, and environments can interact and communicate [29]. IoMusT enables the connection of musical devices, instruments, and technologies (e.g., smart instruments, wearable devices), defined as Musical Things, into an interconnected system. This allows for enhanced music production, performance, and consumption by facilitating real-time interactions between musicians and audiences. The ecosystem supports various use cases, including live performances, music creation, and immersive experiences, leveraging technologies like web audio, cloud computing, and edge computing to ensure low latency and high reliability in communication [30]. In this context, NMP systems represent one of the main elements for exchanging musical data in a timely manner.

Requirements

The primary objective of an NMP system is to allow musicians to collaborate, through the network, as if they were physically located in the same room. To meet this condition, one of the most important parameters to consider is latency. In the field of NMP, latency typically refers to the Mouth-to-Ear (M2E) latency, which is the total time delay from when an audio signal is captured by the microphone of an input device, to when it is played back by the speaker or headphones of an output device. In a typical NMP setup, the input and output devices are separate and located at different physical locations. This latency is influenced by multiple factors, including audio acquisition/playback, network transmission, buffering, and processing delays. To guarantee a satisfactory experience for musicians, the M2E latency should be kept below 30 ms [22, 27], which corresponds to the delay between

two musicians roughly 10 m apart[1] (for an overview on how factors such as genre, tempo and instrumental characterization of the musical piece influence that threshold, the reader is referred to [31, 27]). This target latency is much lower than that of traditional videoconferencing solutions, where typical recommended values range from 100 ms to 150 ms [32].

While M2E latency is a crucial factor that determines the Quality of Experience (QoE) in using an NMP system, it is not the sole determinant. In scenarios such as live performances and music rehearsals, extremely low latency is essential to maintain real-time responsiveness. However, in the context of music education, where often instruction, demonstration, and feedback occur in an alternating manner rather than through synchronous playing, higher latencies can be tolerated without significantly impairing the educational process. In these cases, the emphasis shifts toward ensuring high-quality audio, which is critical for accurately conveying musical nuances and providing effective instruction. Thus, beyond latency, an equally important factor is audio quality. High-fidelity audio acquisition, streaming, and playback are essential for ensuring an immersive and accurate listening experience. This is an area where videoconferencing applications often fall short when used for music related activities, as they are primarily designed and optimized for speech. These systems typically employ lossy codecs and heavy processing techniques, such as echo cancellation and noise suppression, which, while beneficial for speech intelligibility, can significantly degrade the quality and fidelity of musical audio, thereby compromising the listening experience in a musical context [33].

To meet the strict latency requirements specified earlier, NMP systems typically leverage unreliable network protocols, such as the Real-time Transport Protocol (RTP) [34], with the User Datagram Protocol (UDP) [35] at the transport layer, or even directly use UDP. These protocols are favored for their low overhead and ability to minimize transmission delays. Unlike the Transmission Control Protocol (TCP) [36], which incorporates reliability mechanisms like retransmission and congestion control, that introduce latency penalties, UDP-based protocols prioritize speed over guaranteed delivery. Moreover, to ensure high audio quality, as previously discussed, these systems often opt for the transmission of uncompressed audio. This has the dual effect of maintaining full fidelity, preserving all nuances and details

[1] Considering the speed of sound as 343 m/s

present in the original recording, and avoiding encoding/decoding latencies, at the expense of higher bandwidth usage.

In a real-time audio streaming context, where network conditions are inherently unpredictable, packet loss is an unavoidable challenge. Unlike traditional streaming or stored media playback, where large buffers can help smooth out disruptions, real-time audio cannot afford such delays without impacting user experience. Since packet delivery is not guaranteed, codec-based error recovery mechanisms often cannot be leveraged, since no codec is used, and since retransmissions are not possible due to the strict latency requirements, implementing a Packet Loss Concealment (PLC) strategy becomes essential [37, 38]. Packet loss in audio streams introduces discontinuities and artifacts, which perceptually manifest as glitches and degrade the QoE. PLC techniques are employed to mitigate this degradation by computationally reconstructing the missing audio segments. Various PLC techniques have been developed, especially in the VoIP field. These methods range from simple approaches, such as inserting silence or repeating the previous audio frame [39], which are computationally efficient but often result in audible artifacts, to more complex statistical and deep-learning based techniques that aim to create less perceptible replacements for lost data segments [40–43].

While many PLC methods are available, most are specifically tailored to speech and cannot be directly applied in a musical context. For example, WSOLA [44] is widely used for speech PLC, as it compensates for packet losses by extending or shortening segments of audio, without changing pitch, based on time-domain waveform similarity. This approach works well for speech since speech signals exhibit relatively predictable periodicity in voiced segments, allowing WSOLA to adjust duration while preserving intelligibility. However, since WSOLA alters the timing of the audio signal to compensate for missing data, it may introduce unintended distortions when applied in a musical context. In particular, stretching or compressing musical passages can affect the perceived tempo of musicians, which is critical for maintaining rhythmic coherence in ensemble performances. Even small fluctuations in timing can disrupt synchronization between instruments, leading to an unnatural or inconsistent listening and playing experience [31].

In addition to packet loss, other aspects that NMP systems must mitigate are network jitter and clock drift. Jitter refers to the variation in the arrival times of consecutive data packets. This variability disrupts the intended constant rate of

audio data, potentially leading to audible artifacts like gaps or glitches in the audio stream if the packets arrive too late to be played. To counteract jitter, NMP systems typically employ a de-jitter buffer at the receiving end [45]. This buffer temporarily stores incoming packets, absorbing the arrival time variations and allowing audio frames to be played out smoothly at a constant rate. However, the introduction of a de-jitter buffer inherently adds to the overall M2E latency, requiring thus a careful trade-off between jitter resilience and the latency requirements. Furthermore, clock drift poses another significant challenge. Since the audio clocks of the sending and receiving audio devices operate independently, slight differences in their frequencies can accumulate over time, causing the receiver's buffer to either gradually empty (underflow) or fill up (overflow), leading to audio discontinuities/silence or increasing delays/synchronization issues [46]. Typically, the clock of the two audio cards cannot be easily synchronized, and thus alternative approaches are required. Managing both jitter and mitigating clock drift is therefore essential for maintaining both the temporal accuracy and the continuous audio stream.

The last aspect that is extremely important to consider is the ease of use and accessibility of the NMP system. In particular, it should accommodate users of all abilities, including those with disabilities, by providing features such as screen reader compatibility, keyboard navigation, and adjustable text sizes, etc. Additionally, it should be intuitive and user-friendly, allowing individuals with varying levels of technical proficiency to navigate and operate it efficiently. This aspect became particularly noticeable during the pandemic, when people preferred relying on videoconferencing applications rather than NMP systems due to their ease of use.

1.2 Motivation and Aims

Despite the increased interest and significant advancements in NMP, several challenges remain to be addressed before these technologies can achieve mainstream adoption.

One primary concern is ease of use. As mentioned in the previous Sec. 1.1, during the recent pandemic, videoconferencing applications became widely popular largely due to their intuitive interfaces, and even in a musical context they were preferred over NMP specific applications thanks to their user-friendliness. This observation motivates an exploration of NMP implementations that leverage familiar,

widely adopted technologies, such as those available in modern web browsers, to deliver high-quality audio streaming with minimal user friction.

A second challenge relates to performance, particularly regarding the strict latency requirements essential for real-time musical interaction. To reconcile the low-latency demands with the user-friendliness characteristic of conventional video-conferencing applications, this work investigates a hybrid architecture. In this approach, audio processing and streaming are managed by dedicated hardware, while user control is facilitated via a web-based interface. Devices such as the Raspberry Pi offer a promising balance between performance and functionality, making them ideal candidates for experimental validation.

A third critical aspect is PLC. Given the strict latency requirements of NMP systems, implementing PLC on the end-user device is essential to ensure smooth audio playback. While many PLC methods have been developed for VoIP, these techniques are typically optimized for voice signals and may not translate effectively to full-band musical audio. The diverse nature of musical content, from percussive elements to sustained tones, requires a PLC method that can accommodate a wide range of signal characteristics while operating within tight latency constraints. Furthermore, the variability in end-user devices, which can range from personal computers to embedded systems, adds another layer of complexity to the problem. Thus, it is worth exploring PLC techniques to establish a baseline method which can work in many scenarios while providing satisfactory recovery capabilities.

The final aspect to consider is the accessibility of the proposed solution. In recent years, accessibility has become a fundamental requirement for any technological tool, ensuring inclusivity for individuals with diverse abilities. In the context of NMP, it is particularly important to explore alternative approaches to music interfaces and music visualization that can accommodate users with various impairments. By integrating accessible design principles, such solutions can enhance the engagement and participation of both musicians and audiences, fostering a more inclusive and equitable musical experience. This is especially crucial in music education, where accessibility can significantly impact learning opportunities and the development of musical skills for all individuals.

This Ph.D. thesis aims to address these open problems in the NMP field through the following key objectives:

- Investigate and optimize a web-based NMP system using WebRTC `RTCDataChannel` and the Web Audio API, focusing on achieving low M2E latency and uncompressed audio streaming. Finally, evaluate its performance compared to conventional `MediaStream`-based solutions.
- Investigate and design a hybrid web-native NMP system architecture aimed at improving user accessibility through a user-friendly web interface while simultaneously satisfying professional requirements for high-fidelity and low-latency audio streaming via a dedicated native component.
- Design and propose a dedicated hardware-accelerated system for NMP leveraging an FPGA to minimize audio processing delays, overcoming limitations of general-purpose architectures.
- Address the problem of packet loss in NMP systems by proposing and evaluating a PLC method based on Autorgressive (AR) models. The objective is to investigate the effectiveness of AR models in improving audio quality compared to traditional PLC techniques, and to analyze their feasibility for real-time implementation on embedded devices like the Raspberry Pi 4B, considering the trade-off between prediction accuracy and computational cost.
- Assemble a dataset of real NMP communication traces, collected between different geographic locations and using different audio interfaces, to facilitate the development and evaluation of NMP-specific algorithms for automatic de-jitter buffer estimation, clock drift correction, and PLC techniques.
- Investigate and assess an alternative method for conveying visual cues in a remote orchestra conducting scenario, by leveraging a Virtual Reality (VR) headset to capture and transmit the conductor's hand and head gestures. This approach aims to reduce bandwidth requirements and latency compared to traditional video streaming methods. Moreover, it could be integrated into a more accessible NMP framework, where the conductor gestures are rendered through haptic feedback.
- Investigate cross-modal associations between auditory experiences of musical instruments and visual (color, shape) and tactile (texture) stimuli. Based on the findings, propose design directions for a programmable visual interface with customizable haptic feedback to support the musical education of Deaf and Hard of Hearing (DHH) students.

- Provide a comprehensive analysis of the educational and performative needs of disabled users in remote musical teaching and practice. Identify relevant technological solutions to address these needs and explore their potential integration into a proposed NMP framework to enhance inclusivity. Additionally, highlight current open challenges and outline future research directions in this domain.

1.3 Outcomes

Publications

The doctoral research presented in this thesis has resulted in the following scientific publications:

[i] Matteo Sacchetto, Yuen Huang, Andrea Bianco, and Cristina Rottondi. Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications. In *Proceedings of the 17th International Audio Mostly Conference*, AM '22, pages 203–210, New York, NY, USA, October 2022. Association for Computing Machinery

[ii] Matteo Sacchetto, Cristina Rottondi, Antonio Servetti, Louena Shtrepi, Marco Masoero, and Andrea Valle. HiFiReM: un approccio unificato, web e nativo, per la didattica musicale remota. In *Proceedings of the XXIII CIM*, Ancona, October 2022. CIM

[iii] Matteo Sacchetto, Paolo Gastaldi, Chris Chafe, Cristina Rottondi, and Antonio Servetti. Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution. *Journal of the Audio Engineering Society*, 70(11):926–937, November 2022[2]

[iv] Matteo Sacchetto, Leonardo Severi, Cristina Rottondi, Louena Shtrepi, Marco Masoero, Carlo Barbagallo, Andrea Valle, and Antonio Servetti. Pristine Quality Networked Music Performance System for the Web. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum*

[2]This submission won the *2022 JAES Best Paper Award*

2023, pages 2957–2964, Turin, Italy, September 2023. European Acoustics Association

[v] Diego Bert, Nicola Domini, Riccardo Peloso, Leonardo Severi, Matteo Sacchetto, Andrea Bianco, and Cristina Rottondi. FPGA-based Low-Latency Audio Coprocessor for Networked Music Performance. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–8, October 2023

[vi] Matteo Sacchetto, Maria Sangüesa, Piera Bagnus, Chiara Nicora, and Cristina Rottondi. Collection of Design Directions for the Realization of a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–7, October 2023

[vii] Matteo Sacchetto, Cristina Rottondi, and Andrea Bianco. Implementation and optimization of Burg's method for real-time packet loss concealment in networked music performance applications. *Personal and Ubiquitous Computing*, 28(5):727–743, May 2024

[viii] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, and Cristina Rottondi. Introducing DUST: A Dataset of Real-Time UDP Sound Packet Traces. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–4, September 2024

[ix] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Gabriele Abbate, Antonio Paolillo, and Alessandro Giusti. Remote Orchestral Conduction via a Virtual Reality System. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–6, September 2024

[x] Cristina Rottondi, Matteo Sacchetto, Leonardo Severi, and Andrea Bianco. Toward an Inclusive Framework for Remote Musical Education and Practices. *IEEE Access*, 12:173836–173849, November 2024

Demonstrations

The research was also shared through the following demonstrations:

1. Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Aleksandra Knapinska, and Piotr Lechowicz. Demonstration of a Networked Music Performance Experience with MEVO, April 2024. In *2023 4th International Symposium on the Internet of Sounds*

2. Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Gabriele Abbate, Antonio Paolillo, and Alessandro Giusti. Demonstration of remote orchestral conduction via a virtual reality system. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, September 2024

Open-Source Software and Datasets

Part of the research presented in this thesis has been made available as open-source software and publicly available datasets:

1. **JackTrip-WebRTC**: this GitHub repository contains all the code developed for the JackTrip-WebRTC web application. The `main` branch contains the first version of the application, while the `experimental` branch contains the updated version presented in this thesis. Available at https://github.com/jacktrip-webrtc/jacktrip-webrtc/

2. **PLC with AR models - Burg implementation**: this GitHub repository contains all code used for the implementation and the performance evaluation of the AR models, as a PLC strategy, using Burg's method. Additionally, the repository also contains the audio dataset used for the quantitative and qualitative analysis, as well as the recovered files. Available at https://github.com/matteosacchetto/burg-implementation-experiments/

3. **DUST**: this Zenodo repository contains the dataset of UDP packet traces collected using MEVO. Available at https://doi.org/10.5281/zenodo.12726670

4. **Sonic Grain Haptic and Visual Interface**: this GitHub repository contains the code, the audio files and the images used for the realization of a visual and haptic interface prototype based on the TanvasTouch device. Available at https://github.com/mariasanguesa/TanvasTouch_interface

1.4 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2 Provides the necessary background and reviews the related works relevant to the topic presented in this thesis. It offers the reader background knowledge on networking fundamentals, digital audio fundamentals, WebRTC fundamentals, time series principles, and accessibility guidelines. Subsequently, the chapter reviews the related literature on the key aspects presented in this thesis, providing details on available NMP solutions, PLC methods, visual feedback in NMP, and visual/haptic representations.

Chapter 3 Investigates the feasibility of implementing an NMP system based solely on web technologies. It proposes an alternative approach to low-latency P2P audio streaming, diverging from the traditional WebRTC `MediaStream` method. It leverages the `RTCDataChannel`, a component of the WebRTC standard, to send arbitrary data and the `AudioWorklet`, part of the Web Audio API, for custom audio processing, enabling the transmission of uncompressed audio between peers and offering greater control over other aspects, such as buffering strategies. The chapter details the development of JackTrip-WebRTC as a practical implementation of this approach and presents a comparative analysis of its latency and audio quality against traditional videoconferencing solutions. Furthermore, it provides insights and guidelines on how to improve the scalability of the proposed system.

Chapter 4 Presents a hybrid web-based and native NMP application designed to overcome the limited adoption of existing NMP software, often perceived as complex, and the performance constraints of user-friendly videoconferencing tools. This approach seeks to offer a user-friendly experience without compromising on both audio quality and low-latency, essential for real-time musical interaction. This is achieved by providing a user-friendly web application functioning as the system's graphical user interface and enabling P2P connections via the technology detailed in the preceding chapter. Furthermore, a native application running on a dedicated device overcomes the inherent limitations of web applications to support professional real-time collaborations that demand both high-fidelity and minimal latency.

Chapter 5 Presents a dedicated hardware system for NMP that combines software and FPGA approaches to achieve extremely low latency. This system centers around the design of an Application Specific Instruction set Processor implemented using a Transport Triggered Architecture, optimized for minimal latency audio processing within the NMP domain. Leveraging a dedicated FPGA offers the potential to achieve lower latencies compared to purely software-based solutions. Furthermore, this hardware-based approach allows for the implementation of additional functionalities directly on the FPGA, such as audio processing or PLC methods.

Chapter 6 Presents a PLC technique for NMP applications, based on AR models. To address the challenge of packet loss, inherent to low latency real-time audio streaming, this approach leverages AR models to reconstruct lost audio data. The effectiveness of the proposed method is evaluated through quantitative metrics and subjective listening tests, with a comparative analysis against conventional PLC methods such as silence substitution and pattern replication, to assess their ability to balance prediction accuracy with computational efficiency. Subsequently, the chapter investigates the practical real-time implementation of this technique on a Raspberry Pi 4B, analyzing performance constraints and exploring optimization strategies to ensure its ability to run on different NMP environments.

Chapter 7 Introduces DUST, a novel dataset of network packet traces specifically designed for the analysis of real-time audio streaming applications. The motivation for designing DUST arises from the recognized limitations of existing publicly available datasets, which typically examine network factors such as jitter, packet loss, and clock drift in isolation, without considering their combined effects on audio playout. A key feature of DUST is its integration of detailed information about the audio playout process alongside the precise timestamps of packet arrivals, captured in successive snapshots. This enables researchers to simulate scenarios where only a portion of the audio samples within a late-arriving packet might be considered lost, allowing for a more accurate representation of real-world conditions. Additionally, DUST facilitates the precise quantification of clock drift between the sender and receiver audio interfaces. By providing these detailed features, DUST aims to allow the NMP research community to conduct more accurate simulations of real-time audio streaming behavior to foster the development of NMP-specific PLC techniques, buffering strategies, etc.

Chapter 8 Introduces a VR-based approach for providing real-time visual feedback of a conductor to remote musicians in an NMP scenario, aiming to offer a lower-latency and lower bandwidth alternative to traditional video streaming. The proposed system uses a VR headset to track the conductor's gestures, including hand movements and head orientation, and transmits this motion data to be visualized as an avatar to the performers. This real-time visual representation aims to convey the conductor's expressive intentions more efficiently and with better responsiveness, enhancing the overall experience of remote musical collaboration.

Chapter 9 Focuses on gathering design directions for a visual interface with haptic feedback aimed at facilitating the understanding of the concept of "sonic grain" for DHH music students. It explores a cross-modal association between images, colors, sounds and textures to offer a multisensorial perceptual experience of sound. To establish a foundation for this interface, the chapter details a survey campaign conducted with 56 respondents to collect empirical data on sensory associations between musical instruments, colors, shapes, and textures. The results of this survey were then used to develop a prototype that associates musical excerpts with visual and tactile representations. This work aims to provide design guidelines for creating tools for music education that can be standalone or integrated into NMP e-learning environments.

Chapter 10 Provides a comprehensive overview of inclusivity in remote music education and NMP, specifically addressing the challenges faced by users with visual, auditory, or motor impairments. It analyzes and classifies various audiovisual tools, motion tracking systems, immersive audio, and haptic feedback technologies that can be leveraged to enhance the accessibility of web-based platforms for online music teaching and remote performances. Discusses the potential integration of these technological solutions within an envisioned NMP framework, highlighting their benefits through exemplary use-cases for different categories of users with disabilities. Furthermore, it identifies open technical challenges that need to be addressed for the practical realization and deployment of such an inclusive NMP framework, and outlines future research directions towards its incorporation within the broader context of the Musical Metaverse.

Chapter 11 Concludes the thesis, summarizing the key findings and contributions of this research, discussing its limitations, and outlining potential directions for future works in the field.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Networking Fundamentals

The widespread adoption of computer networks and the Internet has profoundly reshaped modern society, facilitating global connectivity, knowledge dissemination, and the efficient delivery of services. These networks interconnect computing devices, enabling resource sharing and data exchange. They can be classified based on various criteria, such as geographic scope, operational purpose, or underlying architectural principles. A common classification distinguishes between Local Area Network (LAN), which operates within a limited space such as a home or office, and Wide Area Network (WAN), which connects multiple LANs and other networks across extensive geographical distances. While these classifications define the scale and scope of a network, its operation depends on a combination of physical and logical components. The physical infrastructure, consisting of transmission media (wired or wireless) and networking hardware such as switches and routers, establishes the foundation for connectivity. At the same time, logical elements, including communication protocols and network services, govern data exchange, ensuring efficient and reliable communication between devices. Together, these components enable networks to function seamlessly across different environments and applications.

This section provides a foundational overview of networking concepts, with particular emphasis on their relevance to real-time audio streaming applications. While the discussion remains broadly applicable to networking principles, special attention is given to the unique challenges and requirements of real-time audio streaming applications, using NMP as a central case study.

The Internet Protocol (IP)

The Internet Protocol (IP) is the fundamental communication protocol that governs the transmission of data packets across interconnected networks, enabling global communication. To understand IP's role, it's helpful to consider the layered architecture of network communication. Network communication models, such as the OSI model and the TCP/IP model, organize network functionalities into distinct layers, each responsible for specific tasks. This layering simplifies network design and implementation by dividing the complex process of data transmission into manageable modules. Each layer adds its own specific information, building upon the information provided by the layers above it. This layered approach simplifies network design and allows different protocols at different layers to work together seamlessly. The OSI model comprises seven layers: Application, Presentation, Session, Transport, Network, Data Link, and Physical. The TCP/IP model, while conceptually similar, condenses some of these layers. Notably absent in the TCP/IP model, when compared to the OSI model, are the Presentation and Session layers. While their functions are still performed in TCP/IP networks, they are often integrated into the Application layer or handled by other protocols.

These layers aren't isolated entities; they interact and exchange information in a well-defined manner. The process of sending data begins at the application layer (Layer 5 in the TCP/IP model), where the application generates the data to be transmitted. This data is then passed down to the transport layer (Layer 4), where protocols like TCP [36] or UDP [35] add headers containing additional information about the connection (TCP), port numbers, and other relevant details. This encapsulated data unit, often called a segment, in TCP, or datagram, in UDP, is then passed down to the network layer (Layer 3), where IP resides. This layer is concerned with the logical addressing and routing of data packets across networks.

At the network layer, IP adds a header, containing the source and destination IP addresses. This header enables routers to properly route the packet across the

network. The resulting unit, now called an IP packet, is then passed down to the data link layer (Layer 2). The data link layer is responsible for framing the IP packet into a format suitable for transmission over the specific physical medium (e.g., Ethernet, Wi-Fi). Finally, the data link layer passes the framed data down to the physical layer (Layer 1), which handles the actual transmission of bits over the physical medium.

On the receiving end, the process is reversed. The physical layer receives the bits and passes them up to the data link layer, which removes its header and trailer and passes the IP packet up to the network layer. IP examines the destination IP address and, if it matches the receiving device's address, removes the IP header and passes the data up to the transport layer. The transport layer then removes its header and delivers the data to the correct application at the application layer. This process of encapsulation and decapsulation at each layer ensures that data is properly addressed, routed, and delivered to its intended destination.

IP's core function is to facilitate the delivery of data packets from a source device to a destination device, regardless of the specific underlying network technologies in use. This fundamental task is accomplished through the use of IP addresses, unique numerical labels assigned to each device connected to an IP network. These addresses act as distinct identifiers, analogous to postal addresses, enabling data packets to be directed to their intended recipients.

Two primary versions of IP are currently in use: IPv4 [59] and IPv6 [60]. IPv4, the earlier iteration, employs 32-bit addresses, providing a theoretical address space of 2^{32} (approximately 4.3 billion) unique addresses. However, due to the exponential growth of the Internet and the proliferation of connected devices, IPv4 addresses have become a scarce resource. IPv6, the more recent version, addresses this limitation by utilizing 128-bit addresses, offering a vastly expanded address space of 2^{128} unique addresses, capable of accommodating the ever-increasing number of Internet-connected devices. Beyond the sheer expansion of the address space, IPv6 also introduces other enhancements, including a simplified header format, improved support for mobile devices, and mechanisms for automatic address configuration. Despite the advantages of IPv6, IPv4 remains prevalently used across many existing networks and systems, largely due to the widespread implementation of Network Address Translation (NAT) and the reuse of private IP addresses [61]. These mechanisms allow multiple devices within a private network to share a single public IPv4 address, effectively mitigating the address exhaustion problem.

IP itself is characterized as a connectionless and best-effort protocol. The "connectionless" nature of IP signifies that it does not establish a dedicated, end-to-end connection between the sender and receiver before transmitting data. Instead, each data packet is treated independently and routed based on its destination IP address. This approach offers flexibility and efficiency, but also means that IP does not inherently guarantee the delivery, order, or reliability of packets. The "best-effort" designation further emphasizes this point, indicating that while IP strives to deliver packets to their intended destinations, it does not provide any guarantees of success. Packet loss, duplication, and out-of-order delivery are all possible occurrences at the IP layer. Higher-level protocols, such as TCP, are then relied upon to provide the necessary reliability, and other essential features, required by many applications.

Transmission Control Protocol vs. User Datagram Protocol

While IP provides the fundamental mechanism for addressing and routing data packets across networks, it doesn't itself guarantee reliable delivery or specific delivery order. That responsibility falls to the transport layer protocols. The two main transport layer protocols are TCP and UDP. They determine how data is packaged, transmitted, and received, offering different trade-offs in terms of reliability, speed, and overhead.

TCP is a connection-oriented, reliable, and ordered delivery protocol. Before any data exchange can occur, TCP establishes a connection between the sender and receiver through a process known as the *three-way handshake*. This handshake ensures that both parties are ready to communicate and agree on the parameters of the connection. Once the connection is established, TCP guarantees that data packets will be delivered in the correct sequence, retransmitting any lost or corrupted packets along the way. It achieves this through mechanisms like sequence numbers, acknowledgements, and timeouts. TCP also provides flow control, preventing the sender from overwhelming the receiver with data, by dynamically adjusting the sending rate based on network conditions and receiver capacity. This reliability and flow control come at the cost of some overhead, as connection establishment, error recovery, and flow control mechanisms require additional data exchange. Consequently, TCP is well-suited for applications where data integrity is required, even at the expense of some additional latency. Examples include file transfer (FTP), email (SMTP), remote login (SSH), and web browsing (HTTP).

In contrast, UDP is a connectionless protocol. It does not establish a connection before transmitting data; instead, it sends packets directly to the destination. UDP is a simpler and faster protocol than TCP, as it avoids the overhead of connection management, error recovery, and flow control. However, this simplicity comes at the cost of reliability. UDP does not guarantee delivery, order, or integrity of packets. Packet loss, duplication, and out-of-order delivery are all possible. Despite this lack of reliability, UDP is well-suited for applications where speed and low latency are more critical than guaranteed delivery. Examples include real-time audio/video streaming (where some packet loss is tolerable), online gaming (where low latency is crucial for real-time interaction), VoIP, and DNS lookups. In these scenarios, the application may implement its mechanisms for handling packet loss or may simply tolerate some level of data loss.

The choice between TCP and UDP depends entirely on the specific requirements of the application. If data integrity and ordered delivery are critical, TCP is the preferred choice. If speed and low latency are critical, UDP is more appropriate. For real-time audio streaming applications like NMP, some data loss is tolerable, and thus UDP is often chosen due to its lower latency.

Latency

Latency, quantifies the total time required for a data packet to traverse a network from source to destination. Latency is influenced by several factors, including propagation delays, transmission delays, and processing delays of the various network components involved in the transmission. Typically measured in milliseconds (ms), latency is a key indicator of network performance and responsiveness, playing a critical role in real-time streaming applications such as NMP. A widely used metric to assess latency is the Round-Trip Time (RTT), which measures the time it takes for a packet to travel to the destination and back to the sender. This measurement inherently captures the transmission and propagations times taken along the forward and backward paths, as well as the processing delays of the various network components, including the destination. A significant advantage of measuring RTT is that it relies on a single clock, at the sender, eliminating the need for synchronized clocks between the sender and receiver

In some contexts, evaluating the one-way delay, i.e., the time for a packet to travel solely from the sender to the receiver, can provide additional insights. Unlike

RTT, which includes both the forward and return paths, one-way delay provides a more granular understanding of network performance, particularly in asymmetric networks where the two paths may have different characteristics. Measuring one-way delay can be especially valuable in applications requiring precise synchronization. However, accurately measuring one-way delay often requires synchronized clocks at both the sender and receiver, which can introduce additional complexity and may not be feasible. Thus, typically, the RTT is measured, and the one-way delay is approximated as $\mathrm{RTT}/2$.

In the context of VoIP and NMP applications, the relevant latency measurement is the M2E latency. M2E latency represents the total time elapsed from when an audio signal is generated at the sender's end (e.g., a person speaking or a musician playing an instrument) to when it is heard at the receiver's end (e.g., through speakers or headphones). This measurement encompasses multiple stages, including audio capture, encoding, packetization, network transmission, buffering, decoding, and audio playback.

M2E latency is a critical metric in real-time audio streaming applications, as excessive delays can disrupt the natural flow of interaction. In VoIP, high latency can lead to unnatural conversation dynamics, causing participants to inadvertently speak over each other or experience confusing pauses. In NMP, even minor latency variations can disrupt synchronization between musicians, making ensemble performance challenging.

Similarly to network delay, measuring M2E latency can be challenging when the sender and receiver devices are in different locations, as precise clock synchronization between the two endpoints is required for accurate measurement. To address this, a common approach involves creating a loopback at the receiver end by short-circuiting the output of the audio device with its input. This allows the measurement to be performed exclusively on the sender side, eliminating the dependency on synchronized clocks. In this scenario, M2E latency is then approximated as half of the measured loopback delay.

Jitter

Jitter denotes the temporal variation in latency. Although a certain amount of latency is inevitable, significant fluctuations can be particularly disruptive for real-

time applications. Even if the average latency remains within acceptable bounds, substantial variations in packet arrival times may lead to performance degradation. For instance, in VoIP and NMP applications, jitter can result in audio dropouts, distortions, or synchronization problems. It is important to note that if a packet arrives later than its expected time, it is effectively equivalent to a *lost* packet.

A common mitigation strategy for jitter involves the use of a de-jitter buffer on the receiver side. The de-jitter buffer temporarily stores incoming packets for a brief amount of time before they are processed by the playback system. However, properly sizing this buffer is a nontrivial challenge. The buffer must be large enough to accommodate fluctuations in packet arrival times, preventing the loss of packets that arrive slightly late, yet not too large, or it introduces excessive additional latency. Furthermore, the level of jitter typically varies throughout the duration of a communication session, meaning that a fixed-size de-jitter buffer is suboptimal. Instead, adaptive buffering strategies should be employed, where the size of the buffer evolves in real time based on current network conditions. Typically, the de-jitter buffer is designed to mitigate jitter while allowing some packet loss in case of high latency spikes. For an overview on jitter and de-jitter buffer strategies, refer to [45].

Packet Loss

Packet loss occurs when data packets either do not reach the destination at all or arrive after their intended timeframe. As previously mentioned, in VoIP and NMP application scenarios, packets arriving past their scheduled playout time are typically regarded as lost and are consequently discarded by the receiver. Thus, during playout the output audio buffer will have “holes”, corresponding to the missing audio packets, which result in perceivable audio glitches or distortion. Moreover, it’s essential to distinguish between random and burst packet loss. Random packet loss occurs sporadically, while burst packet loss involves the consecutive loss of multiple packets, which is generally more disruptive. The impact of packet loss depends heavily on the application.

Generic applications usually address packet loss by retransmitting the lost packets. However, retransmission introduces additional latency, as the sender must wait for either a negative acknowledgement (NAK) from the receiver or a timeout before retransmitting. This added latency makes retransmission generally unsuitable for

real-time applications, where timely delivery is crucial. Waiting for a retransmitted packet to arrive would generally result in the packet arriving too late to be useful.

Forward Error Correction (FEC) offers an alternative approach to handling packet loss, particularly valuable in real-time scenarios. Instead of relying on retransmission, FEC proactively adds redundant information to the data stream at the sender. This added redundancy allows the receiver to reconstruct lost packets, up to a certain extent, without needing to request their retransmission. Various FEC schemes exist, each with different trade-offs in computational complexity and error correction capability. A simple example is sending multiple copies of the same packet; if one copy is lost, others may still arrive. More sophisticated schemes employ encoding techniques, to generate parity data. This parity data allows the receiver to reconstruct lost packets within a block of data. FEC is well-suited for real-time applications, as it avoids the latency introduced by retransmission. However, it does increase bandwidth consumption, as the sender transmits extra redundant data. The level of redundancy is a balance between error correction strength and bandwidth overhead.

For VoIP and NMP applications, a more suitable approach is Packet Loss Concealment (PLC). PLC methods aim to mitigate the perceptual impact of packet loss by attempting to reconstruct or "conceal" the missing data at the receiver. A common and effective PLC technique involves generating missing samples from the previously received ones. This might involve simple repetition of the previous samples, linear interpolation between adjacent samples, or more sophisticated techniques that consider the characteristics of the audio signal. While the exact reconstruction of the original signal is typically not possible, PLC allows the application to continue playing audio with minimal interruption, masking the effect of the lost packet. The specific PLC method employed depends on factors like the application's characteristics, available processing power, the required computational time, and the desired quality of the concealed audio. More advanced PLC techniques may involve analyzing the surrounding audio to predict the missing samples more accurately, or using a Machine Learning/Deep Learning (ML/DL) model to accurately recreate missing sections. However, even sophisticated PLC methods can struggle with burst packet loss, where multiple consecutive packets are lost, as there may be insufficient information to accurately reconstruct the missing data. In such cases, PLC techniques might resort to muting or introducing a brief silence to avoid more disruptive artifacts. For an overview on PLC techniques, refer to [38].

Peer-to-Peer vs. Client-Server

Networked applications can be broadly categorized based on their underlying architectural model: Client-Server (C/S) or Peer-to-Peer (P2P). These paradigms determine how resources and services are distributed among network participants and have significant implications for scalability, reliability, and system complexity.

In a C/S architecture, one or more dedicated servers provide services or resources to a multitude of clients. Clients send requests to the server, which processes these requests and replies with the corresponding responses. This centralized model offers several advantages. Centralized management simplifies administration by enabling updates and maintenance to be performed exclusively on the server. Additionally, security measures can be effectively implemented at the server level through rigorous access controls and policies, while resource allocation can be centrally optimized to ensure equitable distribution among clients. However, C/S architectures also have limitations. The server can become a bottleneck, particularly when handling many clients, and its failure can disrupt service for all connected clients. Furthermore, the server needs to have sufficient processing power and bandwidth to handle all client requests.

In contrast, a P2P architecture distributes resources and responsibilities among all participating devices, referred to as "peers." Each peer can act as both a client and a server, directly sharing resources and services with other peers. This decentralized approach offers several benefits. Scalability is often improved, as the load is distributed across multiple peers. Resilience is enhanced, as the failure of a single peer does not necessarily disrupt the entire network. P2P networks can also be more efficient for certain types of applications, such as file sharing, as data can be exchanged directly between peers without passing through a central server. Often, P2P networks use a so-called "signaling server" to facilitate discovery and exchange connection details between peers. This signaling server can effectively become the "control plane" of the P2P network. P2P communication typically establishes a full-mesh network, where each peer connects directly to every other peer. This minimizes latency between individual pairs of participants, which is highly desirable for real-time interactions. However, this comes at the cost of increased bandwidth consumption, as each peer must send and receive data from every other peer. The bandwidth requirement of each peer increases linearly with the number of participants. Establishing direct P2P connections can also be challenging due to NAT,

firewalls, and other network security measures that can hinder direct communication between devices. Thus, specific techniques, are often necessary to traverse these network obstacles, which may not be feasible in some scenarios.

In the context of NMP, typically a central server is present and only used for session setup and participant management, while the actual audio streams usually are exchanged directly between performers in a P2P fashion to minimize latency. Though, this only works for a few participants. As the number of participants increases, a C/S architecture might become more suitable. In a C/S setup, the server could perform additional actions like pre-aggregating (mixing) the audio streams, thus reducing the bandwidth required by each client. With a C/S architecture, the server handles audio distribution, requiring each client to send only once. Instead, in a P2P setup, each client is responsible for sending its audio to every other client individually. Traditional NMP systems often provide a hybrid approach, where the decision to use a C/S architecture or a P2P one is decided by the user based on their requirements. Refer to [62] for an overview on P2P and C/S architectures in a NMP context.

Network Address Translation

NAT is a technique that allows multiple devices on a private network to share a single public IP address when connecting to the Internet. This was introduced to deal with the scarcity of public IPv4 addresses. Without NATs, each device on a private network would require a public IP address, quickly exhausting the available pool. NATs effectively act as a gateway, translating private IP addresses [61], which can be reused in private networks, and port numbers into a single public IP address and a unique port number for each connection, drastically reducing the number of public IP addresses needed.

NAT works by intercepting outgoing packets from devices on the private network. The NAT (typically integrated into a router) modifies each packet by replacing the source IP address and source port number with the router's public IP address and a selected port number. This mapping is stored in a NAT table. When a response arrives from the Internet, the NAT device uses the destination port number in the incoming packet to look up the corresponding private IP address and port number in its NAT table. It then forwards the packet to the correct device on the private

network. This process effectively hides the internal network topology and private IP addresses from the Internet, providing in addition a basic level of security.

NAT devices can exhibit a range of behaviors regarding how they manage address and port mappings. An initial attempt to categorize NAT behaviors was made in RFC 3489 [63]. However, this early classification scheme was later deemed insufficient to capture the full spectrum of NAT behaviors and was subsequently obsoleted by RFC 5389 [64] and later superseded by RFC 5780 [65], which extended RFC 4787 [66]. RFC 5780 provides a more granular and accurate characterization of NAT types, focusing on the specific ways in which external IP addresses and ports are associated with internal IP address and port combinations. While terminology from RFC 3489 is sometimes still used colloquially, RFC 5780 represents the current standard for describing NAT behavior. RFC 5780 departs from the simplistic cone-based classification by focusing on two key aspects of NAT operation: the mapping of internal endpoints to external IP address and port pairs, and the filtering of incoming packets based on their source characteristics.

In the context of mapping behavior, NAT devices may employ what is termed endpoint-independent mapping, where the same external mapping is consistently assigned to all connections initiated from a particular internal IP and port, regardless of the remote destination. Alternatively, some devices implement address-dependent mapping, where the external mapping varies according to the remote IP address, or even address and port-dependent mapping, where both the remote IP and port influence the resulting external mapping. The latter behavior is typically more restrictive and is often associated with what was formerly known as symmetric NAT.

In parallel, the filtering behavior of NAT devices also varies. With filtering, we refer to the manner in which they permit or block incoming packets. Devices with endpoint-independent filtering will accept incoming packets from any external host once a mapping is established, thereby facilitating communication. In contrast, address-dependent filtering restricts acceptance to packets from the same remote IP address to which the internal host has previously sent packets, while address and port-dependent filtering imposes an even stricter limitation by requiring an exact match of both the remote IP and port. These differences in filtering behavior further delineate the challenges in achieving reliable NAT traversal.

Additionally, NAT devices may or may not support hairpinning, which is the ability of internal hosts to communicate with each other via their external mappings.

Support for hairpinning can be a critical factor in environments where internal P2P communication is necessary, but it is not universally available across all NAT implementations.

P2P applications require direct communication between peers, but NAT can hinder this by masking the private IP addresses and ports of the peers. NAT traversal techniques are employed to overcome these obstacles. These techniques, such as STUN (Session Traversal Utilities for NAT), defined in RFC 5389 [64], TURN (Traversal Using Relays around NAT) defined in RFC 5766 [67] and updated in RFC 8155 [68], and ICE (Interactive Connectivity Establishment), defined in RFC 8445 [69], help peers discover their public IP addresses and ports and establish connections even when behind NAT. STUN enables peers to discover their public IP addresses and port mappings by communicating with a STUN server. In contrast, TURN acts as a relay, forwarding traffic between peers when a direct connection cannot be established. ICE combines the functionalities of both STUN and TURN to determine the optimal path for communication, ensuring that peers can establish connections even when faced with restrictive NAT behaviors or stringent firewall configurations.

With the advent of IPv6, the network architecture has evolved to mitigate the issues that led to the widespread deployment of NAT in IPv4 networks. IPv6 offers a vastly expanded address space, allowing every device to possess a unique, globally routable address and thereby enabling true end-to-end connectivity. In native IPv6 environments, the necessity for NAT is greatly diminished, and many of the traversal challenges inherent to IPv4 are alleviated. Nevertheless, NAT or NAT-like mechanisms may still be employed in IPv6 networks for reasons such as policy enforcement, security, or facilitating communication between IPv6-only and IPv4-only hosts. Thus, while IPv6 is designed to obviate many of the complications introduced by NAT in IPv4, the principles underlying NAT traversal continue to be significant in scenarios where NAT-like behavior persists.

2.1.2 Digital Audio Fundamentals

This section provides a brief overview of digital audio concepts, providing all the basic knowledge needed to understand the principles behind digital audio and its use in applications like NMP.

The Nature of Sound

Sound is a form of energy that propagates as waves through a medium (typically air). These waves are generated by vibrations, which create alternating regions of compression and rarefaction in the surrounding molecules. The human auditory system perceives these variations as sound. Fundamental properties of sound include:

- **Frequency**: The number of vibrational cycles per second, measured in Hertz (Hz). Frequency corresponds to the perceived pitch. Frequency is inversely related to wavelength, i.e., the distance between two consecutive points in phase on a sound wave.
- **Amplitude**: The magnitude of the pressure variations, directly related to the perceived loudness. Greater amplitudes are perceived as louder sounds.
- **Timbre**: The quality or color of a sound that distinguishes different sound sources producing the same pitch and loudness. Timbre is determined by the waveform's shape and the presence of harmonic content or overtones, allowing us to differentiate, for example, a violin from a flute playing the same note.

The Digital Representation of Audio

To work with audio in digital systems, the audio signal must be converted from its continuous *analog* form into a discrete *digital* representation. This conversion involves two fundamental processes: sampling and quantization.

Sampling It involves measuring the amplitude of the analog audio signal at regular intervals. The sampling rate, expressed in hertz (Hz) or kilohertz (kHz), determines how many samples are taken per second. According to the Nyquist-Shannon sampling theorem, the sampling rate must be at least twice the highest frequency we want to capture to avoid aliasing, an effect where higher frequencies are misinterpreted as lower frequencies. Standard sampling rates commonly employed include 44.1 kHz (used in CDs) and 48 kHz (frequently used in audio/video applications). These rates ensure that the full range of audible frequencies (20 Hz to 20 kHz) is accurately captured. In certain specialized contexts, higher sampling rates, typically multiples of the aforementioned values, such as 88.2 kHz, 96 kHz and 192 kHz are used to

capture finer details from the audio signal, as they can represent a wider range of frequencies (well above 20 kHz). E.g., these higher sampling rates find application in high-resolution audio/orchestral recordings, where it is important to preserve all the nuances.

Quantization The sampled signal, while discrete in time, is still continuous in amplitude. Quantization involves approximating these amplitude values to the nearest available digital representation. The precision of this conversion is determined by the bit depth, which specifies the number of bits used to represent each sample. For example, a bit depth of 8 bit allows for $2^8 = 256$ quantization levels, a bit depth of 16 bit allows for $2^{16} = 65,536$ quantization levels and so on. A higher bit depth provides a more precise representation of the original signal, resulting in a higher dynamic range and a better Signal-to-Noise Ratio (SNR). The dynamic range is the range between the quietest and loudest sounds that can be accurately represented. The SNR is the ratio of the power of the desired signal to the power of the background noise. In digital audio, the quantization error primarily determines the noise floor. A higher bit depth reduces quantization error, which in turn lowers the noise floor and increases the SNR.

Digital audio quality is a function of both the sampling rate and the bit depth. Although higher sampling rates and bit depths typically ensure superior audio quality, this enhancement comes at the cost of greater storage capacity requirements and higher data rates for transmission or processing.

Pulse Code Modulation

The direct output of the sampling and quantization process is a stream of digital values representing the amplitude of the audio signal at each time point. This uncompressed digital representation is known as Pulse Code Modulation (PCM) [70]. PCM is the simplest digital audio representation and serves as the foundation for digital audio formats and codecs. It's the raw, uncompressed data before any compression (lossless or lossy) is applied. PCM is also the standard digital audio format used by sound cards and audio interfaces to exchange audio data with applications. The component that performs the conversion from analog sound to this digital PCM representation is the Analog-to-Digital Converter (ADC). Conversely, the compo-

nent that converts PCM data back into the analog domain is the Digital-to-Analog Converter (DAC).

Digital Audio Codecs and Formats

Although the uncompressed nature of PCM data, makes it extremely valuable for its audio fidelity, its intrinsic size presents significant challenges for storage and transmission, especially in scenarios with limited bandwidth or storage capacity. To illustrate this, consider a single monophonic audio signal sampled at 44.1 kHz with a bit depth of 16 bit. This configuration requires a bandwidth of 705.6 kbit/s. This bandwidth requirement doubles for a stereophonic signal, and similarly, doubling either the sample rate or the bit depth also doubles the necessary bandwidth. Consequently, the bandwidth requirement can easily reach the order of some Mbit/s.

Therefore, to mitigate these bandwidth and storage constraints, various strategies have been developed to reduce the bitrate while, to varying degrees, preserving the perceived audio quality. This need for efficient handling of digital audio gives rise to the importance of digital audio codecs and formats.

Audio Codecs Codecs (encoders/decoders) are algorithms used to compress and decompress digital audio data. They serve as fundamental components in the efficient storage and transmission of audio. Codecs can be generally classified into two categories:

- **Lossless Codecs**: These codecs compress audio data without discarding any information, allowing the original PCM data to be perfectly reconstructed from the compressed form. A well-known example of a lossless codec is FLAC (Free Lossless Audio Codec) [71]. While lossless compression preserves the original audio quality, it typically achieves lower compression ratios compared to lossy codecs. Nevertheless, it is ideal for archiving and scenarios where maintaining the original audio fidelity is necessary.
- **Lossy Codecs**: These codecs achieve significantly higher compression ratios by discarding audio information considered perceptually not relevant, based on psychoacoustic models. While this approach greatly reduces file size and bandwidth requirements, it inevitably introduces some degree of quality degradation. Common examples of lossy codecs include MP3 (MPEG Layer 3) [72],

AAC (Advanced Audio Coding) [73], and Opus [74]. Lossy compression is widely employed in audio streaming, online music distribution, and other scenarios where minimizing bandwidth is a primary concern. Many lossy codecs allow the compression level to be adjusted, enabling a trade-off between file size and perceived audio quality.

Audio codecs exhibit a key asymmetry in computational complexity: encoding, the compression process, is typically far more intensive than decoding, the reconstruction. This arises from the encoder's complex analysis of the audio signal to optimize data compression, while the decoder primarily reverses this through a simpler process. Consequently, the encoding stage contributes the majority of processing latency. This, along with any buffering, contributes to the overall latency introduced by codecs. Different codecs balance compression efficiency and latency. Low-latency codecs like Opus are prioritized for real-time applications such as live communication and NMP, even at the cost of slightly reduced compression. Codecs for storage, where real-time interaction is not critical, typically prioritize compression.

Digital Audio Formats While codecs handle compression and decompression, digital audio formats define how the encoded audio data, along with any metadata (information about the audio), is organized and stored. These formats act as containers for the encoded audio streams. For instance, a WAV file can contain uncompressed PCM data. Similarly, an MP4 file might contain audio encoded with the AAC codec. The format specifies how the audio data is structured, including headers, metadata tags, and the actual encoded audio stream. For real-time streaming, the encoded audio data is often further processed into smaller packets for efficient transmission over networks.

2.1.3 WebRTC Fundamentals

Web Real-Time Communication (WebRTC) [75] is an open-source project that provides web browsers with real-time communication (RTC) capabilities via simple application programming interfaces (APIs). It enables P2P audio, video, and data communication without the need for any plugin or third-party software.

Origins and Standardization

The WebRTC project was initiated by Google in 2011 with the acquisition of GIPS (Global IP Solutions), a company specializing in RTC technologies. Recognizing the need for standardization, Google, along with other industry leaders, submitted WebRTC to the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) for standardization.

The IETF focused on standardizing the underlying protocols, such as the Session Description Protocol (SDP) [76] for session negotiation, the Interactive Connectivity Establishment (ICE) [69] for NAT traversal, and the Secure Real-time Transport Protocol (SRTP) [77] for secure media transport. The W3C, on the other hand, worked on defining the JavaScript APIs that developers use to access WebRTC functionalities within web browsers.

Over the years, WebRTC has become an essential component of modern web applications, with major browsers such as Chrome, Firefox, Safari, and Edge providing native support.

WebRTC Architecture

WebRTC operates through a decentralized P2P network and comprises three main components:

- `MediaStream`: Handles the capture and management of media (audio and video) streams from user devices.
- `RTCPeerConnection`: Establishes and maintains P2P connections, handling codec negotiation, encryption, and data transmission.
- `RTCDataChannel`: Enables direct, low-latency data exchange between peers, useful for gaming, file sharing, text-based communication and for sending arbitrary data.

`MediaStreams`, together with the `getUserMedia()` method, are used to request access to input devices, such as the camera or the microphone, and obtain an audio/video stream. The `RTCDataChannel` is a transport mechanism used for sending arbitrary data, e.g., to transfer files between peers. It is implemented through the

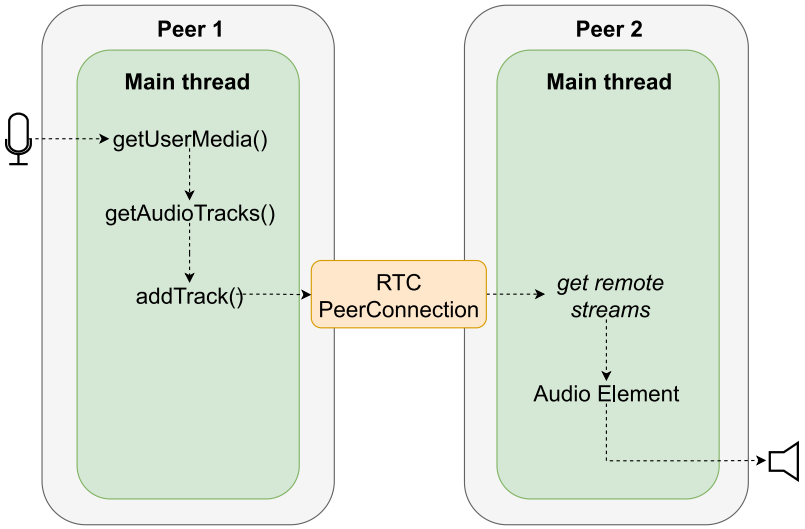


Fig. 2.1 Traditional WebRTC application structure.

Stream Control Transmission Protocol (SCTP) [78], a low-latency, message-oriented, multi-streaming protocol based on UDP and capable of providing also reliable, in-sequence transport of messages with congestion control and packet retransmission similarly to TCP. `RTCPeerConnection` is considered the main part of the WebRTC specification, as it enables audio and video communication between the peers. It is built on top of the RTP/SRTP protocol, but it also provides multimedia codecs, bandwidth management, and includes several signal processing algorithms.

In a WebRTC application, `MediaStreams` play a key role, since they are used both for media acquisition/playback and for exchanging multimedia data through the network. Fig. 2.1 depicts the typical WebRTC application structure.

At the sender side, the web application uses the `mediaDevices.getUserMedia()` method to capture data from a media input, i.e., the audio feed from the internal microphone or the input of a dedicated audio device, as a `MediaStream`. Note that the `getUserMedia()` method takes a `MediaStreamConstraints` object, whose properties are essential to set up the media acquisition process. The `getAudioTracks()` method is then used to retrieve an audio track from the `MediaStream` and the `RTCPeerConnection.addTrack()` method is ultimately used to add the new media track to the set of tracks that will be transmitted to the other peers. The `RTCPeerConnection` component contains all the logic to manage the real-time media stream delivery over the network. At the receiver side, a symmetrical struc-

ture is implemented. When a new track connects, a callback retrieves the remote `MediaStream` and attaches it to an HTML element for playback.

To facilitate P2P connections, WebRTC relies on STUN and TURN servers to bypass NAT restrictions and firewalls. However, WebRTC does not include a built-in signaling mechanism to establish connections. Signaling, which is required for exchanging session descriptions and network information, must be implemented separately by the application developer. This is often achieved using WebSockets, or other communication protocols that allow the exchange of SDP messages and ICE candidates between peers.

For a thorough overview on WebRTC, with implementation details, refer to the Mozilla Developer Network documentation [79].

2.1.4 Time Series Fundamentals

Definition and Types of Time Series

A time series is a sequence of data points, y_t (or $y(t)$), collected or recorded at specific time intervals, t. The defining characteristic of a time series is the inherent temporal dependence between observations; the order of data points is crucial and reflects the evolution of the observed phenomenon. This distinguishes time series analysis from other statistical analyses where observation order may not be relevant. While the underlying phenomenon being measured may be continuous (e.g., temperature, sound pressure), the recorded data is always discrete due to the act of measurement and sampling. The time index t reflects these discrete points in time.

Time series data arises in diverse fields, including economics, finance, environmental science, medicine, and, importantly, audio signal processing. In the context of audio, a digital audio signal is itself a time series, where the distance between each data point (sample) is defined by the sampling rate. Each sample of the digitized audio waveform represents the amplitude of the sound wave at a specific point in time. The sequence of these samples, indexed by their time of occurrence, forms a time series that captures the dynamic variations of the audio signal. Analyzing this time series allows us to understand the characteristics of the sound, process it (e.g., for noise reduction or equalization), and even synthesize new sounds.

Time series can be classified based on several criteria, including the number of variables considered and the regularity of time intervals. Two important classifications are:

- **Univariate Time Series**: A single variable is measured and tracked over time. Examples include the daily closing price of a stock, the temperature recorded hourly at a specific location, or, in the context of audio, the audio samples from a single audio channel.
- **Multivariate Time Series**: Multiple related variables are tracked over time. These variables are often interconnected, and their joint behavior is of interest. Examples include weather forecasting (temperature, humidity, wind speed, etc.), econometrics (GDP, inflation, unemployment), or, in audio, a multichannel audio recording where each channel's audio samples are a time series.

Another important distinction relates to the time intervals between observations:

- **Regular Time Series**: Observations are collected at consistent, equally spaced intervals (e.g., hourly temperature readings, daily stock prices, audio samples taken at a fixed sampling rate). Most time series analysis techniques assume regular intervals.
- **Irregular Time Series**: Observations are collected at irregular or unevenly spaced intervals (e.g., earthquake occurrences, event logs, some types of sensor data). Analyzing irregular time series requires specialized techniques that can handle the varying time gaps between observations. In audio, while digital audio itself is typically sampled at regular intervals, events within the audio (like note onsets in music) can form an irregular time series.

A time series can often be decomposed into several underlying components, which contribute to its overall behavior. These components are often assumed to be additive, meaning that they combine linearly to form the observed time series. The primary components are:

- **Trend**: The long-term direction of the data, representing a general increase or decrease. Trends can be linear (constant rate of change) or nonlinear (varying rate of change).

- **Seasonality**: Regular, repeating patterns within the data, occurring at fixed intervals. These patterns are often associated with specific time periods (e.g., daily, weekly, yearly).
- **Cyclic Patterns**: Fluctuations that occur over longer, non-fixed periods, often related to broader economic or business cycles. These cycles are more irregular than seasonal patterns and can span several years.
- **Irregular (Residual) Variations**: Unpredictable, random fluctuations due to external or unforeseen factors. These variations, also known as noise or residuals, do not follow any discernible pattern. In audio, irregular variations could be caused by sudden bursts of noise or random environmental sounds.

Stationarity

An important concept in time series analysis is *stationarity*. A time series is considered stationary if its statistical properties, such as the mean, variance, and autocorrelation structure, remain constant over time. More formally, a time series $\{y_t\}$ is said to be *strictly stationary* if the joint probability distribution of $(y_{t_1}, y_{t_2}, \ldots, y_{t_k})$ is identical to that of $(y_{t_1+h}, y_{t_2+h}, \ldots, y_{t_k+h})$ for all time shifts h and for any collection of time points $t_1, t_2, \ldots, t_k$.

In many practical applications, the assumption of *wide-sense stationarity* (WSS) (or weak-sense stationarity) is adopted. A time series is considered wide-sense stationary if it satisfies the following less restrictive conditions:

- **Constant Mean**: The expected value of the time series is constant for every time point t, i.e., $\mathrm{E}[y_t] = \mu, \quad \forall t$.
- **Constant Variance**: The variance of the time series is invariant with respect to time, i.e., $\mathrm{Var}(y_t) = \sigma^2, \quad \forall t$.
- **Constant Autocovariance**: The autocovariance between y_t and y_{t+h} solely on the lag h (the time difference between the two points), and not on the specific time t, i.e., $\mathrm{Cov}(y_t, y_{t+h}) = g_h, \quad \forall t, h$.

WSS requires only that its first moment, i.e., the mean, and its autocovariance function remain invariant over time, and that the second moment, i.e., the variance,

is finite for all t. This makes it a much weaker and more practical condition to check in real-world time series. Most practical applications of time series analysis rely on the assumption of WSS, as strict stationarity is difficult to verify. From here on out, unless otherwise stated, when we refer to "stationarity," we are generally referring to WSS. It's crucial to understand that WSS does not mean the data is constant. A stationary time series will still exhibit fluctuations. What remains constant are the statistical properties of these fluctuations, the spread of the fluctuations, and how the fluctuations at different times are related.

Temporal Dependencies

The concepts of autocorrelation and lagging are fundamental to understanding the inherent temporal dependencies within time series data. These concepts are essential for modeling and predicting future values based on past observations.

Autocorrelation Quantifies the linear relationship between a time series and a *lagged* version of itself. Specifically, it measures the degree to which past values of a time series are correlated with its current values. Formally, the autocorrelation at lag k, denoted as ρ_k, is defined as:

$$\rho_k = \frac{\text{Cov}(y_t, y_{t-k})}{\sigma_y \cdot \sigma_{y-k}} \tag{2.1}$$

where $\text{Cov}(y_t, y_{t-k})$ is the covariance between y_t and y_{t-k}, and σ_y and σ_{y-k} are the standard deviations of y_t and y_{t-k}, respectively. For a stationary time series, where the variance remains constant over time (i.e., $\sigma_y = \sigma_{y-k}$), the formula simplifies to:

$$\rho_k = \frac{\text{Cov}(y_t, y_{t-k})}{\text{Var}(y_t)} \tag{2.2}$$

Autocorrelation values range from -1 to 1. A positive autocorrelation at lag k indicates that values k time periods apart tend to move in the same direction. A negative autocorrelation indicates that they tend to move in opposite directions. A zero autocorrelation implies no linear relationship.

The Autocorrelation Function (ACF) is a plot of the autocorrelation coefficients ρ_k against the lag k. The ACF provides a visual representation of the autocorrelation

structure of the time series and is a crucial tool for identifying patterns and selecting appropriate time series models. For example, a slowly decaying ACF indicates that past values have a prolonged influence, which is characteristic of a trend component, while a periodic ACF suggests seasonality.

Lagging Is the process of shifting a time series backward in time. A lagged version of a time series, y_{t-k}, consists of the original time series values shifted by k time periods. Lagging is essential for analyzing the temporal dependencies captured by autocorrelation. For example, to calculate the autocorrelation at lag 1, we compare the current value y_t with the previous value y_{t-1}.

In audio signal processing, autocorrelation and lagging are fundamental techniques. Autocorrelation is widely used in pitch detection algorithms, where it helps identify the fundamental frequency of speech or musical notes by measuring periodicities in the waveform. Lagging is used in various audio processing applications, including echo cancellation, time stretching, and pitch shifting. For example, in echo cancellation, a delayed version of the audio signal is subtracted from the current signal to remove unwanted echoes.

Lagged variables are also used as input features in time series forecasting models. For instance, in an autoregressive (AR) model, past values of the time series, $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$, are used as predictors of the current value y_t. The selection of the appropriate number of lags is crucial for building accurate models. Techniques like the Partial Autocorrelation Function (PACF) help determine the model order.

Model and Forecasting

Accurately modeling and forecasting time series data is essential for extracting meaningful predictive insights from temporally ordered observations. Given the dependencies between past and future values, a range of modeling strategies has been developed. These strategies can be broadly classified into *statistical models* and *machine learning models*, each offering unique advantages based on data characteristics and forecasting objectives.

Statistical Models Statistical models rely on probabilistic frameworks to capture the underlying patterns and dependencies within time series data. They often assume

specific structural properties, such as stationarity or linearity, and use explicit mathematical formulations to model relationships between past and future values. Among these, some relevant models are:

- **Autoregressive (AR) Model**: The AR model assumes that future values can be predicted as a linear combination of past observations. An AR model of order p, denoted as AR(p), is defined as follows:

$$y_t = c + \sum_{i=1}^{p} a_i y_{t-i} + \varepsilon_t \tag{2.3}$$

  where c is a constant, a_i are the autoregressive coefficients, and ε represents white noise. The order p can be either fixed in advanced or computed through techniques such as the analysis of the PACF.

- **Moving Average (MA) Model**: Instead of past observations, an MA model relies on past error terms to forecast future values. An MA model of order q, denoted as MA(q), is defined as follows:

$$y_t = \mu + \sum_{i=1}^{q} b_i \varepsilon_{t-i} + \varepsilon_t \tag{2.4}$$

  where μ is the mean of the series, b_i are the moving average coefficients, and $\varepsilon_t, \varepsilon_{t-i}, \ldots, \varepsilon_{t-q}$ are the error terms.

- **Autoregressive Moving Average (ARMA) Model**: A combination of AR and MA models. Denoted as ARMA(p, q), it captures both autoregressive and moving average components.

- **Autoregressive Integrated Moving Average (ARIMA) Model**: Extends the ARMA model by introducing a differencing component (d) to transform non-stationary series into stationary ones. It is denoted as ARIMA(p, q, d).

Machine Learning Models Machine learning approaches offer powerful alternatives to statistical models, particularly in cases where time series exhibit complex, nonlinear patterns.

- **Regression Models**: Both linear and nonlinear regression models can be applied to time series forecasting by using lagged values as input features.

- **Decision Trees and Random Forests**: These models capture intricate relationships and interactions within time series data. Random forests, in particular, offer robustness by averaging multiple decision trees.
- **Long Short-Term Memory (LSTM) Networks**: A type of recurrent neural network (RNN) designed to learn long-range dependencies, making them well-suited for sequential forecasting tasks, including speech recognition and financial forecasting.
- **Deep Learning Models**: Advanced deep learning architectures, such as Transformer-based models and Convolutional Neural Networks (CNNs), have been increasingly used for time series forecasting. Transformers capture long-range dependencies through self-attention mechanisms, while CNNs can extract hierarchical temporal features from time series data.

Selecting the appropriate forecasting model requires balancing interpretability, computational efficiency, and predictive accuracy. While statistical models provide robust and interpretable solutions for structured time series data, machine learning and deep learning models offer greater flexibility and adaptability for capturing complex, nonlinear dependencies. Though, regardless of the chosen model, rigorous model evaluation is essential to ensure accuracy and reliability.

Model Evaluation

Evaluating the performance of a time series model is crucial to ensure its reliability and accuracy in forecasting future values. Several metrics are used to quantify the difference between predicted and actual values, providing insights into the model's strengths and weaknesses. Among these, some commonly used metrics are:

- **Mean Absolute Error** (MAE): The MAE measures the average magnitude of the errors in the forecast values, without considering their direction. It is calculated as:

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{2.5}$$

  where n is the number of forecasted elements, y_t is the actual value at time t and $\hat{y}_t$ is the forecasted value at time t. MAE is easy to interpret and is expressed in the same units as the original data. A lower MAE indicates better

forecast accuracy. It is also less sensitive to outliers than MSE or RMSE, as it doesn't involve squaring the errors.

- **Mean Squared Error** (MSE): The MSE calculates the average of the squared differences between the forecasted and actual values:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 \tag{2.6}$$

where n is the number of forecasted elements, y_t is the actual value at time t and $\hat{y}_t$ is the forecasted value at time t. MSE penalizes larger errors more heavily than MAE due to the squaring operation. This can be both a strength and a weakness. It emphasizes the importance of avoiding large errors, which might be critical in certain applications, but is more sensitive to outliers. The units of MSE are squared, making it less directly interpretable than MAE.

- **Root Mean Squared Error** (RMSE): The RSME is the square root of the MSE:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{2.7}$$

where n is the number of forecasted elements, y_t is the actual value at time t and $\hat{y}_t$ is the forecasted value at time t. RMSE expresses the error in the same units as the original data. Like MSE, RMSE is sensitive to outliers. It provides a measure of the typical "size" of the forecast errors. RMSE is often preferred over MSE since it is more interpretable

The choice between MAE, MSE, and RMSE depends on the specific application and the priorities of the analysis. It's often useful to consider multiple metrics to get a more comprehensive picture of model performance. Although these metrics provide valuable quantitative measures of forecast accuracy, they should be considered in conjunction with other factors, such as the interpretability of the model and its computational cost. Moreover, the context in which the time series data is generated can significantly influence metric selection. For example, although MAE, MSE, and RMSE are widely utilized in many forecasting applications, they may not fully capture performance nuances in specialized fields like audio or speech processing. In these domains, alternative evaluation measures, e.g., PEAQ, potentially incorporating perceptual quality or domain-specific error tolerances, may be more effective in reflecting the true performance of the model.

2.1.5 Accessibility Fundamentals

Inclusive design is a fundamental aspect in the creation of accessible interfaces, ensuring that digital products and services serve a diverse population regardless of individual abilities. This section explores the core principles and considerations that guide the design of accessible interfaces, encompassing the crucial aspects of visual, auditory, and haptic modalities.

At its core, accessible interface design aims to make information perceivable across various sensory modalities, thereby accommodating a wide spectrum of user needs. To achieve this, one must first ensure that all non-textual content is accompanied by appropriate text alternatives. These alternatives enable assistive technologies, such as screen readers, to effectively communicate information to users with visual impairments. Furthermore, time-based media, such as video and audio recordings, must be accompanied by comprehensive support, including captions, subtitles, and transcripts, with audio descriptions provided for visual elements within video content. This ensures that users with auditory or visual impairments can fully engage with multimedia content. Content should be adaptable, presented in various formats without compromising information or structural integrity. This includes responsive design that seamlessly accommodates diverse display sizes and orientations. Moreover, content must be easily distinguishable, thus requiring sufficient color contrast between text and background, avoidance of reliance on color as the sole means of conveying information, and provision of user-adjustable text sizes, audio and volume controls. The foundation of these principles lies in the well-established framework provided by the Web Content Accessibility Guidelines (WCAG) [80], which offer a robust framework for creating accessible web content.

Operability constitutes another critical dimension of accessible interface design. All interface functionality must be accessible via keyboard navigation, accommodating users who cannot utilize mouse or trackpad input, e.g., blind or visually impaired (BVI) users relying on a screen reader for content identification. Additionally, the interface must provide flexible interaction timelines to accommodate all users, including those with cognitive impairments. Interfaces should be designed to prevent content that could induce seizures or adverse physical reactions, such as rapidly flashing or changing elements. Clear and consistent navigation mechanisms, employing headings, labels, and landmarks, are essential for user orientation. Moreover, content should be presented in clear, simple language, avoiding complex terms and

sentence structures. Additional input assistance, including clear error messages, form labels, and robust input validation, serves to minimize user errors and enhance overall usability.

Adhering to established web standards and accessibility guidelines not only ensures compatibility with a wide range of user agents and assistive technologies but also reinforces the commitment to inclusivity. Screen reader compatibility, for example, is a critical component in guaranteeing that digital content is accessible to all users.

Finally, while modalities such as haptic feedback are less standardized compared to visual and auditory interfaces, they nonetheless play a vital role in accessible design. Haptic feedback can confirm user actions, provide navigational cues, and signal critical alerts, especially for users with visual or auditory impairments. To maintain consistency and user control, it is preferable to offer options to adjust the intensity or disable haptic feedback altogether.

2.2 Related Work

2.2.1 Networked Music Performance Systems

NMP systems can be characterized by four key elements that define their architecture and functionality. The first one is related to the type of connectivity that they support or recommend. This includes both the physical connection, i.e., Ethernet, Wi-Fi, etc. and the logical connection, i.e., P2P or C/S. The second element is related to the platform that they run on. This can be either a dedicated device which only runs the NMP application, or a general purpose device, i.e., a common computer, which may run additional software. The third aspect involves the type of specialized software that they run. This can either be an application running directly on the dedicated device, an application that users install on their computer and use in the form of either a standalone application or as a plug-in (VST) of a Digital Audio Workstation (DAW), or an application that can be used through a web browser. The last aspect involves the type of interaction the application is designed for. This can be either real-time interaction among musicians, a beat-synchronous approach, where musicians perform in sync with the previous beat of the song, etc.

Over the years, many NMP systems have been developed, and the following discussion presents some of the most influential, which are still maintained, classified using a slight variant of the framework introduced in [2]. For a comprehensive, up-to-date list of the available NMP systems, refer to [81].

Realistic Jam Approach (RJA)

This category includes NMP systems designed to allow musicians to interact remotely as if they were physically located in the same room. These software programs aim to replicate the experience of live, in-person jamming for remote musicians by minimizing latency and ensuring high audio quality. When we generically refer to an NMP system, we are typically considering this category of solutions.

Typically, these systems tend to prefer P2P over C/S, and all of them recommend a cabled connection (Ethernet) over a wireless one, to minimize network latency and jitter and to satisfy the strict latency requirements of NMP. Though, throughout the years, many evolved to support both P2P and C/S. Moreover, to satisfy constraints in terms of audio quality, all of these systems typically support uncompressed audio (PCM) and some provide additional codecs, to reduce bandwidth usage at the expense of a potential slight reduction in audio quality. Some well-known solutions belonging to this category are JackTrip [8], SoundJack [9], Jamulus [20], JamKazam [82], Sonobus [83], Elk Live [84] and LOLA [14].

JackTrip [8] is an open-source NMP system, initially developed in the early 2000s, distributed as a standalone application targeted to general-purpose computers. It can also be deployed on dedicated hardware platforms, such as a Raspberry Pi [85], and managed via a web application interface. At its core, JackTrip streams uncompressed audio over UDP, operating in either a P2P or a C/S configuration. In the C/S setup, the server additionally performs real-time audio mixing and can handle various audio processing tasks. Nowadays, JackTrip is available in two forms: the open-source version and a paid service one known as Virtual Studio [86]. Virtual Studio expands on the base functionality by offering enhanced features such as integrated video streaming, live-streaming, audio recording, Digital Audio Workstations (DAWs) integration, in the form of a VST, and other advanced capabilities. It also provides a managed, cloud-based/edge-based server backbone and an improved Graphical User Interface (GUI), which simplifies configuration and improves overall usability and performance for remote musical collaboration.

SoundJack is a proprietary R&D NMP system, provided free of charge, and designed to deliver ultra-low latency audio communication. It is distributed as a standalone application for general-purpose computers and has been continuously maintained since its initial release around 2006 [9]. During the pandemic, a Raspberry Pi-backed version, referred to as the Fastmusic Box, was developed to extend its reach into more affordable, dedicated hardware environments [22]. SoundJack transmits either uncompressed or Opus encoded audio over UDP, and provides advanced controls that allow users to fine-tune buffering and processing settings to adapt to varying network conditions. Its user interface is designed to serve both beginners, by offering simple controls, and advanced users who wish to optimize performance parameters for live, real-time remote jamming. Furthermore, SoundJack is versatile in its network architecture, supporting both P2P and C/S configurations, and providing C/S functionalities similar to JackTrip. Alongside JackTrip, SoundJack is recognized as one of the earliest developed and still actively maintained NMP systems, offering comparable functionalities that continue to support the evolving needs of remote musical collaboration.

Jamulus [20] is an open-source NMP system renowned for its simplicity and performance. It uses a C/S model, and uses Opus as the audio codec[1]. Musicians stream their audio to a central server, which then mixes the incoming streams in real time. Designed to run on general-purpose computers (Windows, macOS, Linux) with minimal configuration requirements, Jamulus has become a popular tool for live rehearsals and jam sessions, especially during the pandemic.

JamKazam [82] is a proprietary NMP platform that supports both P2P and C/S modes, offering musicians flexibility in their desired setup. It is designed for real-time jamming, rehearsals, and live-streaming, integrating video and social features to create a comprehensive virtual collaboration environment. With a free version available for basic sessions and a premium tier unlocking advanced functionalities such as improved latency management and recording options, JamKazam targets a wide range of users, from educators to professional ensembles.

SonoBus [83] is a cross-platform NMP solution that enables real-time audio collaboration on Windows, macOS, Linux, iOS, and Android. It is one of the newest NMP systems, with its first release in 2020. It stands out for its dual approach: it can function as a standalone application or as a VST plugin, allowing seamless

[1] https://github.com/jamulussoftware/jamulus

integration with DAWs. SonoBus supports both uncompressed PCM audio and low-latency compressed audio (using the Opus codec), providing users the flexibility to optimize performance based on their available bandwidth. Its fully featured interface and broad compatibility make it a versatile choice for remote jamming and recording. Though, beginners may find the feature-full GUI and the connection setup more complex than other NMP systems.

Elk Live was a specialized NMP system that leveraged the Elk Audio OS [87] to deliver professional-grade remote collaboration. It was primarily designed to be run on a dedicated HW device (a hardware audio interface called Elk Bridge), though in recent developments a macOS application was also introduced. Elk Live focused on minimizing latency through the use of uncompressed audio and dedicated hardware integration. Its managed service infrastructure simplified setup and ensured reliable, studio-quality performance, making it especially well-suited for professional ensembles and educational institutions that demand the audio fidelity combined with ease-of-use [84]. Though this solution was extremely promising from a performance and ease-of-use perspective, the Elk Live service shut down in June 2024.

LOLA is a proprietary NMP system originally developed for high-end educational and professional environments such as conservatories. Operating in a C/S configuration, LOLA streams both uncompressed audio and uncompressed video with extremely low latency (one of the few NMP systems to support that), ensuring that remote performances closely emulate live, in-person rehearsals. Due to the transmission of uncompressed video, LOLA imposes significant bandwidth requirements (a minimum network bandwidth of 1 Gbit/s is required[2]) and relies on a robust network backbone to maintain minimal latency and packet loss, making high-performance network infrastructure a critical component for optimal operation. In Italy, this is achieved through the GARR Consortium network backbone, which connects all universities and conservatories throughout Italy. Although it demands high-performance hardware, professional-grade audio equipment, and a reliable high-bandwidth network, LOLA is available free for academic and non-profit educational uses (with commercial usage offered under a shareware model), positioning it as a robust solution for institutions committed to low-latency remote rehearsals [14].

[2] https://lola.conts.it/downloads/LolaManual2.0.0rev001.pdf

Semi-Realistic Jam Approach (RJA)

Although this category is absent in [2], it is introduced here to distinguish systems that are conceived as RJA systems, where the underlying technology limits the latency figures these systems can achieve. In this category fall all the browser-based NMP systems. These systems rely on WebRTC and tend to have a M2E latency in the order of 100 ms to 150 ms (latency figures recommended for VoIP applications [32]). Moreover, these usually only support the audio codecs available in WebRTC[3]. Refer to Chap. 3 for a detailed overview. The two well-know systems that fall in this category are 4D Jam Connect [88] and Jitsi [89]. One thing to note is that even though Jitsi is a videoconferencing solution and not specifically designed for NMP, thanks to the high-level of customization of the WebRTC-specific settings it exposes, it became one of the preferred web-based solutions for musical purposes as it was able to achieve lower latencies w.r.t. other web-based videoconferencing solutions [19], and provided a way to disable algorithms such as echo cancellation, automatic gain control and noise suppression, not desirable in an NMP scenario.

While these systems do not deliver the same performance of RJA systems, they are usually perceived as simpler to use since they are often designed similarly to traditional videoconferencing applications, and provide very similar GUI and interaction modalities. Moreover, they do not require any installation, and can run on any device with a "recent" web-browser, thus simplifying the initial setup.

Latency Accepting Approach (LAA)

This last category includes systems that embraced the inherent characteristics of the Internet as a globally interconnected, spatially independent medium. Thus, instead of trying to minimize latency, these systems are designed considering that the network can introduce latencies exceeding 200 ms, acknowledging it as an acceptable operational parameter. This approach explores novel methodologies for musical interaction that accommodate, rather than eliminate, network-induced latency. A well-know solution belonging to this category is NINJAM [90] (together with any NINJAM-based or NINJAM-compatible solution).

[3] WebRTC supported audio codecs

Unlike RJA systems, which prioritize ultra-low latency to enable synchronous interaction, NINJAM adopts a unique approach by leveraging latency as a structural component of the musical experience. It operates on a looping buffer principle, where musicians transmit and receive audio in fixed-length intervals based on the tempo and time signature of the session. This method allows performers to play in sync with one another despite network delays, as each musician hears the others with a predetermined musical delay. Citing directly from NINJAM's website [90]: "The NINJAM client records and streams synchronized intervals of music between participants. Just as the interval finishes recording, it begins playing on everyone else's client. So when you play through an interval, you're playing along with the previous interval of everybody else, and they're playing along with your previous interval. If this sounds pretty bizarre, it sort of is, until you get used to it, then it becomes pretty natural. In many ways, it can be more forgiving than a normal jam, because mistakes propagate differently."

2.2.2 WebRTC

The recent increased need for interacting and working remotely has fostered the adoption of simple and easy-to-use software tools for videoconferencing. In almost every modern web browser, those solutions leverage the WebRTC standard. WebRTC provides an open and royalty-free standard for real-time audio/video acquisition and P2P multimedia and data transmission. Nowadays, most of the services that were originally developed with proprietary protocols, such as Cisco's WebEx or Zoom, have a WebRTC alternative implementation that allows users to join directly from their web browser without downloading any software [91].

In [17] the authors provide an extensive comparison of the performance of four videoconferencing applications (Zoom, Microsoft Teams, VoiceLessonsApp, and FaceTime) in terms of audio fidelity for the real-time transmission of musical content. By analyzing the introduced distortions in both time and frequency domain, they conclude that, though all the considered platforms introduce artifacts or noise, Zoom provides the highest fidelity to dynamics and spectral characterization of the streamed signals. Nevertheless, we should note that the web-based Zoom alternative that leverages the WebRTC implementation does not provide the same audio quality level as the native application.

To overcome the scalability limits of the P2P structure of WebRTC, in [92] the authors propose a system that efficiently extends such structure with synchronized mixing and broadcasting of the audio/video streams to large audiences with adequate real-time performance. [93] proposes an alternative to improve bandwidth efficiency and audio quality for speech communications with an adaptive bitrate switching algorithm that selects the most suitable bitrate and operating mode of the Opus codec [94], thus lowering the impact of bursty and random packet loss conditions.

The same authors in [95] highlight the limits of the WebRTC de-jitter buffer behavior in enabling low-latency audio communications, even in the presence of negligible network delay: measurements show that WebRTC default applications may lead to latency levels that approach the ITU-T Rec. G.114 [32] thresholds. In order to significantly reduce the WebRTC communication latency, the authors in [96] envisioned bypassing the WebRTC standard algorithm for the de-jitter buffer and all the overhead introduced by the `RTCPeerConnection` channel. The alternative investigated in that study is to rely on the `RTCDataChannel` that, although designed for data (and not media) communications, can be configured to exclude reliable and ordered delivery algorithms and to allow a UDP-like unreliable and unordered transmission. While that implementation did not achieve the expected results due to the support for efficient and timely management of audio data in web browsers still being in its infancy [97], and new improvements yet to be introduced [98, 99], it laid the foundations for future explorations.

2.2.3 Packet Loss Concealment

In NMP, the strict latency requirements and the use of UDP as the transport layer provide no guarantees on packet delivery or their timing. Moreover, the common preference for transmitting uncompressed PCM audio, rather than using audio codecs, prevents leveraging PLC techniques integrated into streaming-oriented codecs such as Opus [74]. As a result, implementing PLC is essential to maintain audio quality and continuity in NMP systems. While in the VoIP field, many PLC methods have been investigated [100], these often do not translate well when applied to generic audio, especially in a musical context. Musical signals have a wider frequency range, intricate harmonic structures, and transient events that are more challenging to reconstruct than speech. While some more advanced techniques are available, such as waveform similarity overlap-add (WSOLA) [40], or even deep learning based

approaches [101], in the NMP field, some of the most common PLC methods are still based on filling the gaps with silence (silence substitution) or with the repetition of the previous content (pattern replication), due to their wide applicability and extremely low computational cost.

Among the PLC solutions explored in VoIP, AR models have emerged as a promising approach. Their adoption has been extensively studied and proven effective for PLC in speech signal restoration (see, e.g., [42]). However, their application in NMP scenarios remains unexplored. In music-related contexts, AR models have been applied for various tasks. For example, they have been leveraged in combination with N-gram models to predict beat-synchronous Mel-frequency Cepstral coefficients and chroma features in musical audio streams [102], and for online prediction of the tempo curve [103], where the parameters of the AR model are determined by a deep convolutional neural network fed with the history of the tempo curve and the music score of the piece being performed. An AR self-attention based model for music score infilling is described in [104], to generate a polyphonic music sequence that fills in the gap between given past and future excerpts. In [105], the authors discuss how to apply Burg's method to estimate the short-term coefficients of the linear-predictive coding analysis of the Opus codec for speech and audio. The optimized Burg's method implementation used in the Opus codec is described in [106]. Many NMP solutions are available today (Sec. 2.2.1), most also running on embedded devices such as Raspberry Pi Single Board Computer, due to its easiness of portability and deployment. Therefore, in this thesis, we focus on PLC methods that can be executed also on a Raspberry Pi processor within the real-time constraints of NMP applications. The majority of the studies on PLC in NMP applications focus on recovery methods for the transmission of audio signals in MIDI format. The authors of [1] proposed a PLC method leveraging a recovery journal section within RTP packets, which encodes the most recent variations of the system status so that a receiver can recover from a wrong status induced by the non-reproduction of MIDI events or commands contained in previously lost packets. Other recovery approaches rely on auxiliary reliably connected channels to transmit critical MIDI events [107] or on acknowledgment mechanisms to permit the re-transmission of lost music data [108]. Differently, in [109], a deep convolutional neural network is adopted to conceal the missing audio fragments in PCM format. The proposed ML-based framework is shown to outperform the predictive capabilities of AR models, but would require GPU-accelerated hardware to ensure acceptable execution

delays, which is unlikely to be deployable in embedded devices like the Raspberry Pi. Finally, it is worth mentioning that there is rich literature on ML-based generative models for raw audio, though not necessarily focused on NMP applications. The interested reader may refer to [110] for a comprehensive survey of the most relevant studies on such a topic.

2.2.4 Visual Feedback in Networked Music Performance

Several studies tackled the assessment of the maximum acceptable asynchrony between audio and video in real-time streaming. In [111], the authors show that the perception of misalignment is deeply linked to the type of content considered. As a general outcome, the study highlights that when the delay between visual and audio streams is greater than 100 ms, more than a half of the subjects involved in the test campaign therein reported were able to perceive it, thus suggesting that the delay offset should not exceed 100 ms.

In [112], the authors explored how different delay times between audio and video impacted a musical performance involving remote musicians and a conductor. One interesting outcome of that study is that visualization of the conductor's gestural cues was considered extremely important by the musicians, as it helped mitigate the negative impact of audio latency on the performance quality. Furthermore, high audio latency was found to be much more impactful on the musicians' capability of maintaining a stable tempo w.r.t. video latency, thus suggesting that higher video delays may be tolerated, provided that they do not exceed 100 ms to 200 ms.

A preliminary attempt to bypass video streaming in NMP practices can be found in [113], where a conventional computer mouse was proposed as a replacement for the conductor's baton in an NMP scenario, with the aim of providing low bit rate and low-latency visual feedback while preserving network resources for audio streaming. Results show that the proposed mechanism, though not user-friendly, proved viable for conducted music, with conductor and orchestra tolerating one-way transmission delays in the range from 35 to 75 ms.

More recently, several studies investigated real-time streaming of control or gestural data for music-related applications in the Metaverse. In [114, 115], extended reality (XR) platforms are proposed to support bidirectional polyrhythmic interactions between players, embodied by avatars, exploiting a virtual drum circle.

In [116], an XR Environment for immersive NMP is presented. The framework implements a position-dependent visual and auditory representation of the users within a shared virtual space, characterized by early reflections and diffuse reverberation. In [117, 118], upper body tracking is exploited for avatar representation of four remote musicians, coupled with immersive audio rendering of the virtual space.

2.2.5 Haptic and Visual Methods for Deaf and Hard of Hearing Subjects

Haptic Devices

Several studies have investigated the conveyance of musical information through the sense of touch (see [119] for a thorough overview). Tactile rendering setups normally focus on capturing a single music feature, e.g., rhythm, pitch, melody, timbre, and loudness. Up to now, conveyance of "sonic grain" (for an explanation of this concept, refer to [120]) through haptic feedback mechanisms has never been attempted.

The usage of haptic feedback to enrich the emotive experience while listening to music has been experimented in [121]. In music teaching, vibrotactile feedback has been already demonstrated to convey the concept of rhythm and of the spectrum of sound frequencies to visually impaired and Deaf and Hard of Hearing (DHH) students [122, 123]. Moreover, systems that transpose gestural cues into haptic feedback have proven to be useful to enrich musical perception and interplay for such categories of performers, as reported in [124].

Notably, an emerging sector where haptic feedback has been proven to have a disruptive impact is that of the Internet of Sounds [125], where networked environments are leveraged to offer various types of music-related experiences to their users. In [126], the authors experimented with the use of different kinds of haptic wearable devices to enhance the experience of the audience during such performances, enabling listeners to perceive the performed musical pieces thorough vibrotactile feedback devices. In the study, the devices were responding to the performed music, but the authors envisioned the use of such types of devices as well for facilitating the creative participation of the audience in the live performance.

Music Visualization Techniques

Visualization can also be exploited to present music content to subjects with auditory impairments: within the rich corpus of scientific literature dedicated to such a topic (see [127] for a comprehensive survey), a few studies have specifically targeted the DHH community. Among those, in [128] sounds generated by a violin are digitized and associated to movements of objects such as flowers and plants, with the goal of achieving interactive sound visualization to support music education for children with hearing impairments. [129] presents ViTune, a music visualization prototype which enhances the musical experiences of the DHH community thanks to the usage of an on-screen visualizer generating effects alongside music. Additionally, [130] introduces BufferBeats, a toolkit for creating multimodal experiences for streaming services, aimed at improving accessibility to online music streaming by DHH subjects. The study is motivated by the presence of Digital Rights Management anti-piracy encryption, which often restricts the access to audio data which is required to create multimodal experiences for music streaming. Focusing on songs, opera and theatrical pieces, displaying criteria of captions and lyrics specifically tailored for DHH audiences have also been investigated in [131] (such as font and size, number of lines and positioning on the screen).

Currently, the combined use of visual, auditory, and tactile feedback for multimodal interfaces in music education of DHH students remains unexplored in the scientific literature.

Chapter 3

Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution

Nowadays, widely used videoconferencing software has been diffused even further by the social distancing measures adopted during the SARS-CoV-2 pandemic. However, none of the web-based solutions currently available support high-fidelity stereo audio streaming, which is a fundamental prerequisite for NMP applications. This is mainly due to the fact that the WebRTC `RTCPeerConnection` standard for web-based audio streaming does not handle uncompressed audio formats. To overcome that limitation, this thesis discusses an implementation of 16-bit PCM stereo audio transmission on top of the WebRTC `RTCDataChannel`, leveraging the Web Audio API and the `AudioWorklet` components. Results obtained with multiple configurations, browsers, and operating systems show that the proposed approach outperforms the WebRTC `RTCPeerConnection` standard in terms of audio quality and latency, which in the best configuration has been reduced to 40 ms between two MacBooks on a LAN.

The work presented in this chapter is part of a contribution published in the Journal of the Audio Engineering Society in November 2022 [49].

3.1 Introduction

Web-based audio/video (A/V) streaming platforms for videoconferencing services have become ubiquitous, in part thanks to their easy integration within the web environment. The majority of web-based videoconferencing solutions leverage Web Real-Time Communication (WebRTC) `MediaStreams` [75], which represents the standard approach to P2P low-latency A/V streaming. The social distancing countermeasures imposed to mitigate the spreading of the SARS-CoV-2 pandemic have further fostered the extension of such platforms to NMP applications, to support real-time musical interaction between musicians performing together from multiple geographical locations as if they were in the same room.

Unfortunately, `MediaStreams` do not allow for fine-grained control over the latency introduced by A/V acquisition, processing and buffering, which is of pivotal importance for the perceived QoE in NMP applications (see Sec. 1.1). For this reason, NMP platforms are typically conceived as standalone, native applications that run directly on an operating system of choice to benefit from fast access to system calls and customized software implementation. Moreover, they can directly exploit UDP transmission of uncompressed audio streams to minimize the M2E latency. Furthermore, these software programs often provide access to the manual configuration of various audio-streaming related settings, such as packet size and de-jitter buffering size, audio codec selection, etc., to strike the best trade-off between quality of the audio streaming and perceived latency. For an overview on NMP systems, refer to Sec. 2.2.1.

Conversely, real-time communication based on WebRTC media streams offers limited configuration options and adopts compressed audio formats to limit the bitrate, at the price of introducing additional latency during the audio encoding/decoding process: such delay may reach 20 ms or more, as measured in experiments with the Aretusa NMP software [132].

To overcome the lack of configurability of WebRTC media streams, i.e., the `MediaStream`, an alternative option is to avoid their usage and to exploit instead the `RTCDataChannel`, which is usually adopted for non-multimedia data. The feasibility of such an approach was first explored in 2017 by researchers at Uninett, Otto J. Wittner et al. [96], who took advantage of the now deprecated `ScriptProcessorNode` [133] to extract data from the `MediaStream` and send it over WebRTC's `RTCDataChannel`.

However, the limited audio processing capabilities of the `ScriptProcessorNode` introduced a significant amount of latency. The new `AudioWorklet` API [134] promises to reduce such latency contribution consistently, as it provides independent real-time threads and enables more efficient audio processing.

Inspired by such possibilities, in this study an alternative approach to P2P high quality and low-latency audio communication is presented, which exploits both the WebRTC `RTCDataChannel` and the `AudioWorklet` API. To demonstrate its capabilities, we developed a WebRTC application named JackTrip-WebRTC as a sibling to the popular JackTrip software[1] for NMP over the Internet. The application is released as open-source software on GitHub[2]. The *main* branch contains the code of the early version of JackTrip-WebRTC, presented in the Web Audio Conference 2021 [135], while the *experimental* branch contains the code of the enhanced version discussed in this chapter.

A preliminary version of this study appears in [135], which validated the idea. The novel contributions presented in this chapter are related to the adoption of the *SharedArrayBuffer* (SAB) instead of the `Message Channel` (MC) for exchanging data between the main thread and the audio thread. The reduction of the number of AudioWorklets used, which previously increased linearly with the number of connected peers, leading to CPU bottlenecks that restricted the number of connected peers to a small number, thus limiting practical usability. In the updated version, this linear dependency has been removed, thus improving the scalability of the system. Lastly, we performed an extensive performance assessment of the proposed implementation, including the comparison of its performance to that of widely available web-based audio communication platforms such as Jitsi [89] and Google Meet [136].

The remainder of this chapter is organized as follows. Sec. 3.2 discusses limitation of videoconferencing tools when used in an NMP context. Sec. 3.3 describes our proposed solution for low-latency WebRTC communications. Experimental results are discussed in Sec. 3.4 and Sec. 3.5, respectively, presenting the overall M2E latency and the latency introduced by each layer of the audio chain. Sec. 3.6 reports the assessment of the application jitter and scalability for multi-peer communications.

[1] https://www.jacktrip.org/
[2] https://github.com/jacktrip-webrtc/jacktrip-webrtc/

In conclusion, Sec. 3.7 provides a summary of the key findings and discussions presented in this chapter.

3.2 Limitations of Videoconferencing Tools in an NMP Scenario

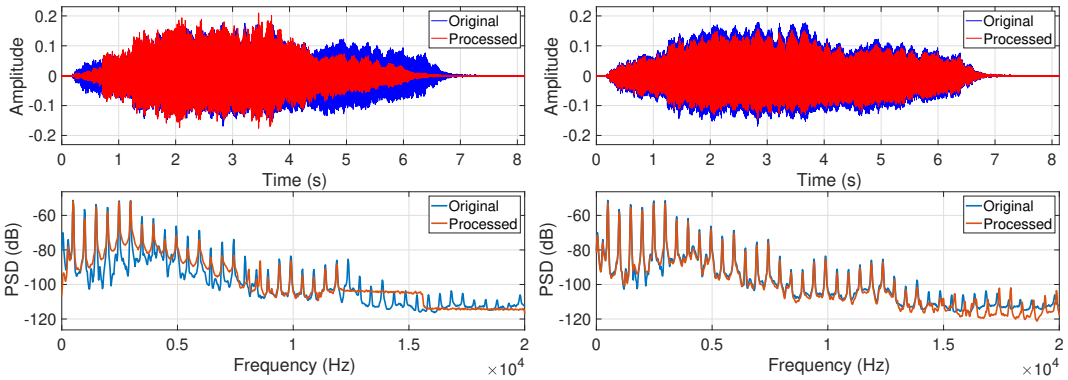
Designed before the social distancing experienced during the pandemic, the WebRTC protocol is mainly optimized for speech-centered, turn-taking scenarios. The criteria that govern the selection of multimedia codecs and of the audio processing algorithms focus on preserving network bandwidth at the expenses of higher M2E latency and on preserving intelligibility at the expenses of audio fidelity.

In fact, although the WebRTC protocol could support any multimedia codec, the standardization process defined a limited set of formats to ensure compatibility. Among them, the frame-based perceptual codecs Opus [94] and G.711 [70] are mandatory. Conversely, the use of uncompressed PCM, which would ensure high-fidelity quality and introduce no algorithmic delay, is not supported. Moreover, as reported later in Sec. 3.4, even with all the settings optimized to achieve the lowest latency, a WebRTC application based on the `RTCPeerConnection` and `MediaStream` still introduces too much delay (60 ms in the best scenario), to be considered a suitable tool for NMP applications.

Another main drawback exhibited by the vast majority of videoconferencing applications, when adopted in NMP scenarios, is that adaptive noise suppression (ANS), echo cancellation (EC) and automatic gain control (AGC) are enabled by default. Such features are useful to increase the intelligibility of speech signals, but alter the naturalness of the sounds produced by musical instruments or by singers' voices, and introduce distortions in sound transients and intentional loudness dynamic variations. Furthermore, all such functionalities introduce non-negligible processing delays. For some videoconferencing software products, it is difficult or even impossible to disable such features. As an example, Google Meet and Microsoft Teams do not offer the possibility to disable AGC. In Jitsi, disabling AGC is possible only by tinkering with the URL parameters, since there is no setting option directly available in the GUI. Skype introduced such a possibility only very

recently (previously, a modification in the Windows system registry was necessary to disable AGC).

To substantiate the above claims, using the same audio material provided by [17], available online[3], we analyzed the audio transmitted by several videoconferencing applications. The obtained results are comparable to those reported in [17].



(a) Google Meet (default with AGC, NS, AEC) (b) Jitsi (with AGC, NS, AEC disabled)

Fig. 3.1 Distortions due to AGC (top) and perceptual codec compression (bottom) in Google Meet and Jitsi, running in Chrome v.98. The blue line represents the original signal, while the red line represents the processed signal, in both the time and frequency domains. The audio track is a sample of a bowed violin playing quietly (-12 dB) at the B4 pitch.

Fig. 3.1 shows an example of the alteration of the amplitude envelope of a bowed violin, where the intensity of various segments of the excerpt has been completely reshaped by the AGC, thus altering the original signal. In addition, in several occurrences, sustained notes by cello or violin were suppressed since misinterpreted as noise by the ANS algorithm.

Even when ANS, EC, and AGC are disabled, videoconferencing applications still adopt some form of audio compression to reduce the media bitrate, thereby reducing the audio quality. Fig. 3.1 also shows the comparison of the frequency content of the original signal with the compressed one obtained with Google Meet or Jitsi. Google Meet seems to preserve less accurately the quality of the signal, especially in the high-frequency range. The same happens with all the other videoconferencing applications except for Zoom, which provides an "original audio" option that can transmit the audio with minimal alterations of its spectral content. However, we must note that this option is available only in the native Zoom application, whereas

[3]The media files used in the article are available at https://www.ianhowellcountertenor.com/preliminary-report-testing-video-conferencing-platforms

it is not provided in the web-based client application[4]. Ultimately, the logic that automatically controls the bandwidth usage and the latency of the de-jitter buffer, (required to mitigate the variability in packet inter-arrival times) is designed to maximize speech quality at the expense of introducing additional latency (up to 150 ms). Such latency can be tolerated in interactive speech communications, but substantially hinders remote music performances.

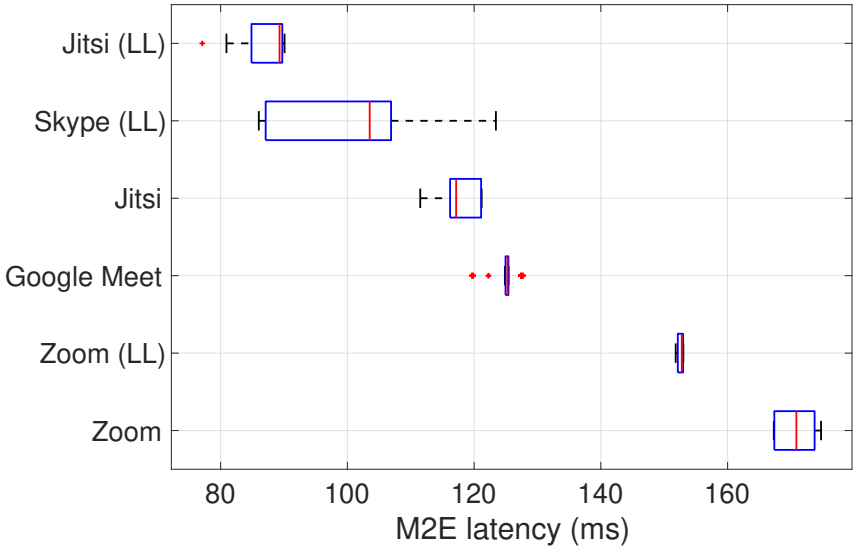


Fig. 3.2 M2E latency of common videoconferencing software, both native and web-based (Chrome v.98), between two MacBooks connected to the same Ethernet switch. When available, the LL software configuration without audio processing was selected. For each configuration, ten sessions of the duration of one minute have been measured.

Fig. 3.2 shows the results of several experiments with popular videoconferencing software: the measured M2E delays are in the range of 80 ms to 180 ms between two computers in the same local area network. Note that, when allowed by the software, we tested both the default configuration and a low-latency (LL) configuration where most of the audio processing features that increase the M2E latency were disabled. Jitsi allows disabling audio processing, automatic EC, ANS, AGC, high pass filtering by providing such information in the URL[5]. Skype allows disabling ANS and AGC from the *audio & video* preference panel. Zoom has the "original audio" setting that

[4]Comparison of Zoom Desktop with Zoom Web Client: https://support.zoom.us/hc/en-us/articles/360027397692

[5]https://www.homepages.ucl.ac.uk/~rtnvrmp/JitsiStereo.html

disables all audio processing and improves audio coding[6]. In contrast, Google Meet lacks customization options, thus only the default configuration has been used.

Being based on WebRTC, Jitsi, and Google Meet show similar latency figures, on the order of 120 ms, when the default configuration is used. In its low-latency configuration, Jitsi latency is reduced by about 30 ms, and almost the same reduction is obtained with Zoom. We note that Zoom presents the highest latency among all the applications because it always works in a C/S mode, thus relying on the transmission through a server on the Internet, instead than in a P2P mode with a direct connection between the devices.

3.3 The Proposed Low-Latency WebRTC Application

The structure of the proposed low-latency WebRTC application is depicted in Fig. 3.3. It is derived from the classical structure of a WebRTC application, reported in Fig. 2.1.

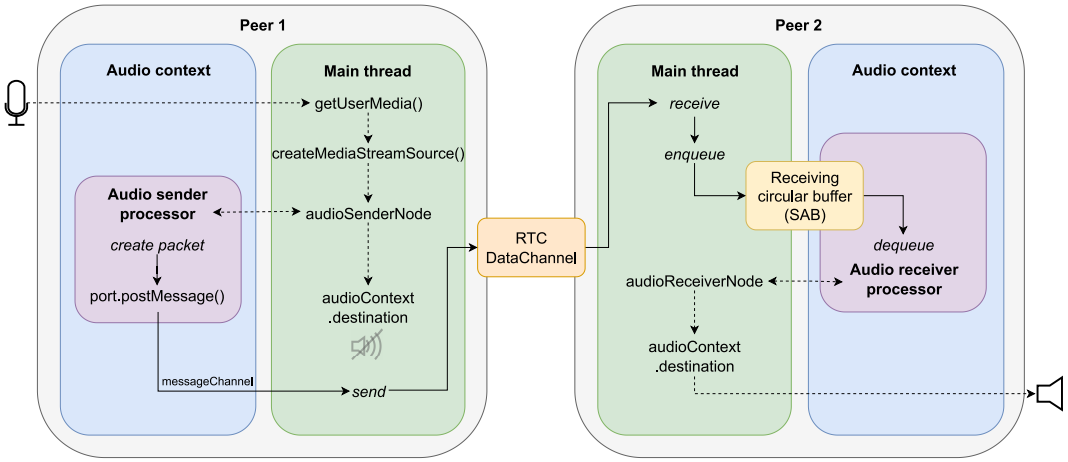


Fig. 3.3 Custom WebRTC application structure with the `RTCDataChannel` and the `Shared Array Buffer` (SAB).

Note that, in contrast to native NMP applications (see Sec. 2.2.1) that have direct access to the low-level OS API, allowing them to implement custom functionalities, a web application is forced to use only the JavaScript API made available by the web browser. It follows that, in our attempt to identify alternative WebRTC configurations

[6] https://blog.zoom.us/high-fidelity-music-mode-professional-audio-on-zoom

with the aim of reducing latency, we can modify the application structure only if the JavaScript API provides us with alternative implementations for a given task.

To switch from an architecture based on the `RTCPeerConnection` channel to an architecture based on the `RTCDataChannel`, we have to resort to the Web Audio API [79] and the `AudioWorklet` interface. It is, in fact, necessary to tamper with the WebRTC `MediaStream` object to access the raw audio content and route it to and from the `RTCDataChannel`. To achieve that, two custom `AudioNodes` need to be implemented: one on the sender side, to extract the raw audio from the `MediaStream` object obtained through the `getUserMedia` method, and one on the receiver side to reverse the process.

More in detail, at the transmitter the `MediaStream` is fed as an audio source into the audio processing graph using the `createMediaStreamSource` method of the `AudioContext`, which creates a `MediaStreamAudioSourceNode` (Fig. 3.3), i.e., an audio node whose media is retrieved from the specified source media stream. Such a node can then be chained with other nodes of the Web Audio API for further processing.

In this case, the required audio processing is limited to accessing the raw audio data and packetizing it for transmission. This task is performed by an AudioWorklet that consists of two objects: an `AudioWorkletNode` that allows the `AudioWorklet` to be connected with other nodes of the Web Audio graph, and an `AudioWorkletProcessor` that is responsible for executing the actual audio processing. We name the custom `AudioWorklet` in the transmitter `AudioSender` (Fig. 3.3). It extracts uncompressed audio from the source `MediaStream` and creates the packets to be delivered on the `RTCDataChannel`. Low-latency requirements can be satisfied by the `AudioWorkletProcessor` since it is executed in a separate thread and is called each 128 samples, i.e., every 2.6 ms at 48 kHz.

At the receiver, the custom `AudioWorklet` is named `AudioReceiver`. Here, the audio data retrieved from the `RTCDataChannel` needs to be managed by a proper de-jitter buffer before playout. The `AudioWorkletProcessor` is then in charge of transferring the audio frames from the de-jitter buffer to its outputs using its *process* method. Finally, the associated `AudioWorkletNode` is connected to the `AudioContext.destination` to enable audio playback on the selected output device. In this architecture, the `RTCDataChannel` has been configured to behave as a UDP-like channel, so not implementing any retransmission or reordering logic [137].

Since the `AudioWorkletNode` and the `AudioWorkletProcessor` are executed in different threads, the browser has to handle the overhead of exchanging audio data between the two of them. Two different methods can be used to transfer data between the main thread and the audio thread:

- `MessageChannel`[7]: interface which allows the two threads to exchange data using messages between one another through pipes with a *port* on each end.
- `SharedArrayBuffer`[8]: object which represents a pre-allocated memory block that can be accessed by both threads for fast data transfer.

In our preliminary implementation [135], we described an approach based on the asynchronous `MessageChannel`, already available in the web browsers for a long time. This channel was conceived to allow two separate scripts running in different browsing context of the same document to communicate between each other, passing messages through a two-way pipe named port. This approach showed to be suboptimal for real-time audio as it suffers from high latency, and it requires repeated memory allocation to copy the data, thus strongly limiting the reliability and scalability of the system.

In the updated implementation presented in this chapter, we replaced it with a `SharedArrayBuffer`, a buffer shared between the page main thread and the AudioWorklet thread, which enables more efficient data communication between them. Here, a single buffer is allocated once, and reused for each data exchange, since it can be accessed from both threads with proper coordination, as illustrated in [99]. For multi-thread synchronization, `Atomics`[9] methods are used to ensure concurrent access to the shared buffer.

On the receiver side, in the main thread, the *enqueue* process consists of *mixing* every audio frame received from each peer into the buffer [138]. This operation is implemented with a simple addition in the appropriate position, and it is performed with the `Atomics.add` method. The position is given by the current frame index and the peer offset. The peer offset is computed when the first frame from that peer is received, and it is equal to the current read position minus the de-jitter offset.

[7] https://developer.mozilla.org/en-US/docs/Web/API/MessageChannel

[8] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/SharedArrayBuffer

[9] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Atomics

Since the de-jitter buffer is a circular buffer, every position is computed *modulo* its size. See [139] for a description of the de-jitter buffer logic. Note though that, as explained in Sec. 2.1.1, in a real application scenario, this strategy is suboptimal, as the offset in the buffer for each peer should dynamically adapt to the jitter. Though, in the context of this prototype, this simpler solution is acceptable.

Instead, in the AudioWorklet thread, the *dequeue* procedure consists of *extracting* successive audio frames from the shared buffer for playout. The reading position is always incremented by one and the values of the current frame are substituted by zeros so that, in case of lost frames, silence is played out. These operations can be performed in a single instruction by the `Atomics.exchange` method.

3.4 Latency Measurements

3.4.1 Experimental Settings

To measure the M2E latency, a testing environment is set up where the delay between the sound acquired by the sending input device and the sound emitted by the receiving output device can be accurately quantified. To monitor and better control the network delay, we perform the measurements between two different devices (peers) connected throughout a wired connection to the same switch of a LAN, as shown in Fig. 3.4.

Moreover, to exclude behavioral differences due to the use of different audio devices, all measurements use the same USB audio interface: the Behringer UMC404HD. This choice is also motivated by the fact that this platform is targeted at professional musicians, and so we consider the use of an external audio interface mandatory. Additionally, measurements performed in a non-professional setting, using the internal audio card of the devices, have already been presented in [135]. Measurements are collected using custom developed software running on an external device, which is calibrated every time to avoid the introduction of any processing latency. This allows to accurately measure the whole audio chain delay, including the delay introduced by the audio interfaces themselves. The measurement device emits an impulsive sound and records the time elapsed between the peak value of the emitted sound and the corresponding peak of the received one.

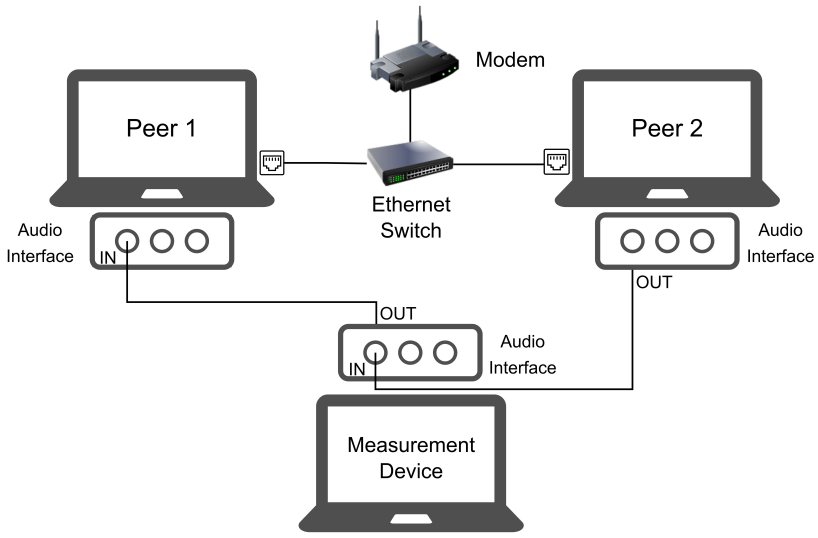


Fig. 3.4 M2E measurement process. The measurement device computes the delay as the time difference between the time it emits the test signal and the time it receives it back from the full audio chain, which includes the audio interfaces (here represented as the rectangular objects), the peers and the network transmission between them.

As reported in [135], the first crucial step to reduce the overall latency is to appropriately configure the `getUserMedia` call. By disabling all the additional processing performed on `MediaStreams`, i.e., AGC, ANS and EC, it is possible to reduce the latency by tens of milliseconds. Both the `RTCPeerConnection` and `RTCDataChannel` based solutions may benefit from this latency reduction. For this reason, all the measurements of this work are performed with the settings as in Listing 3.1, i.e., `autoGainControl` (AGC), `echoCancellation` (EC), `noiseSuppression` (NS) set to `false`, and `latency` set to `0`.

All measurements are performed on three different operating systems: macOS (v10.14.6 and v12.2.1) with Core Audio, Windows 10 (v20H2) with Windows Audio, and Ubuntu Linux (20.04, kernel 5.15) with PulseAudio[10]. Windows and Linux are running on two different machines: an HP ProBook 430 G8 with an i7-1165G7 (4 cores / 8 threads) processor and 16 GB LPDDR4 3200 MHz RAM and a custom-built PC with an i9-10900 (10 cores / 20 threads) processor with 16 GB of DDR4 2133 MHz RAM. Two different MacBooks are running macOS: a MacBook Pro 2018 with an i7-8559U (4 cores / 8 threads) processor and 16 GB LPDDR3 2133 MHz RAM, and a MacBook Pro 2020 with an i5-8257U (4 cores / 8 threads) processor

[10]All three OSes have been used with the default system setup. However, further reduction in latency may be achieved with a custom setup, in particular with Linux.

and 8 GB LPDDR3 2133 MHz RAM. For each OS, we tested the latest version of the two major web browsers that currently support AudioWorklets: Chrome v.98 and Firefox v.97.

Listing 3.1 Low-latency constraints for WebRTC `getUserMedia`.

```
const constraints = {
  video: false,
  audio: {
    autoGainControl: false,
    echoCancellation: false,
    latency: 0,
    noiseSuppression: false,
    sampleRate: 48000,
  }
};
const mediaStream =
  navigator.mediaDevices.getUserMedia(constraints);
```

3.4.2 Overall Latency

Measurements in Fig. 3.5 show the comparison between the `RTCPeerConnection` and the `RTCDataChannel` implementations. Each row summarizes the results of ten sessions of the duration of one minute and the box plot presents the maximum, 75th percentile, median, 25th percentile, and minimum values.

Interestingly, the approach with the `RTCDataChannel` shows a significantly lower M2E delay with respect to the architecture with the `RTCPeerConnection`, with a M2E latency as low as 40 ms on Firefox on macOS. This result, which already includes a de-jitter buffer of about 20 ms, is very close to the ones that could be achieved by native NMP tools. On the contrary, the performance on other browsers and OSes is not so appealing, mostly since, currently, no browser can be easily configured to directly exploit the audio card low-latency drivers (ASIO, ALSA), as can be done by native NMP applications. Nevertheless, the measurements reveal that the adoption of the `AudioWorklet` and the `RTCDataChannel` for low-latency audio communication may be more effective than the actual implementation of the WebRTC `MediaStream` that suffers from the algorithmic delay of audio coding and,

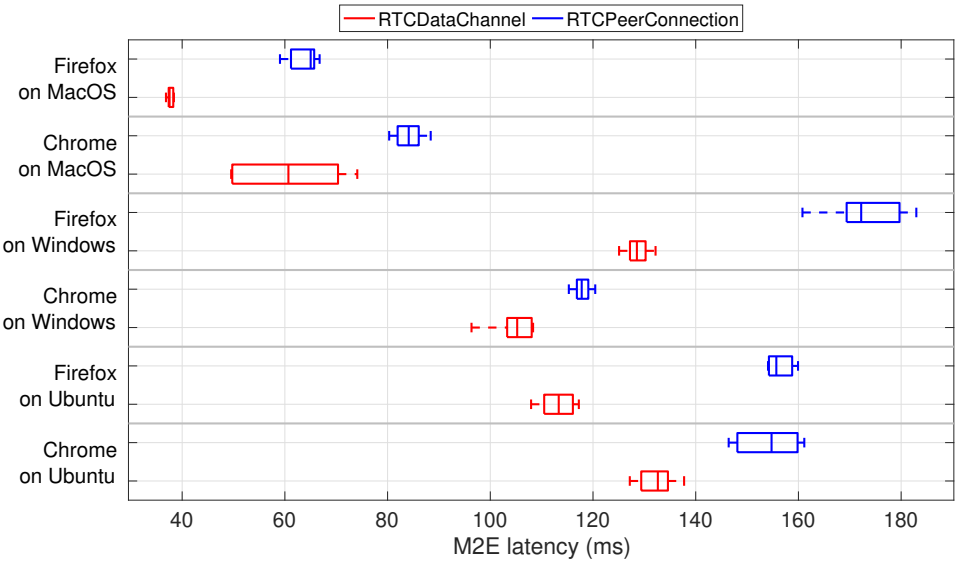


Fig. 3.5 M2E latency measurements with different browsers and OSes for the two approaches based on the `RTCPeerConnection` and the `RTCDataChannel`.

mostly, from its propensity for using large delays in the de-jitter buffer, that, in the reported experiments, have been measured in the order of 20 ms to 60 ms [95].

3.5 Audio Chain Latency Assessment

In this section, we report further measurements to investigate how each layer of the audio chain contributes to the overall delay.

From the architecture depicted in Fig. 3.3 we can decompose the application into three main layers:

A. The **input/output layer** that corresponds to the elements that are directly responsible for exchanging data with the audio device, i.e., *getUserMedia* and *audioContext.destination*.

B. The **processing layer** that includes the *AudioWorklets* and that transfers audio data from the *MediaStream* to the main thread and vice versa. Here, in the receiver, a de-jitter buffer (that introduces additional latency) is implemented to compensate for network delay variance.

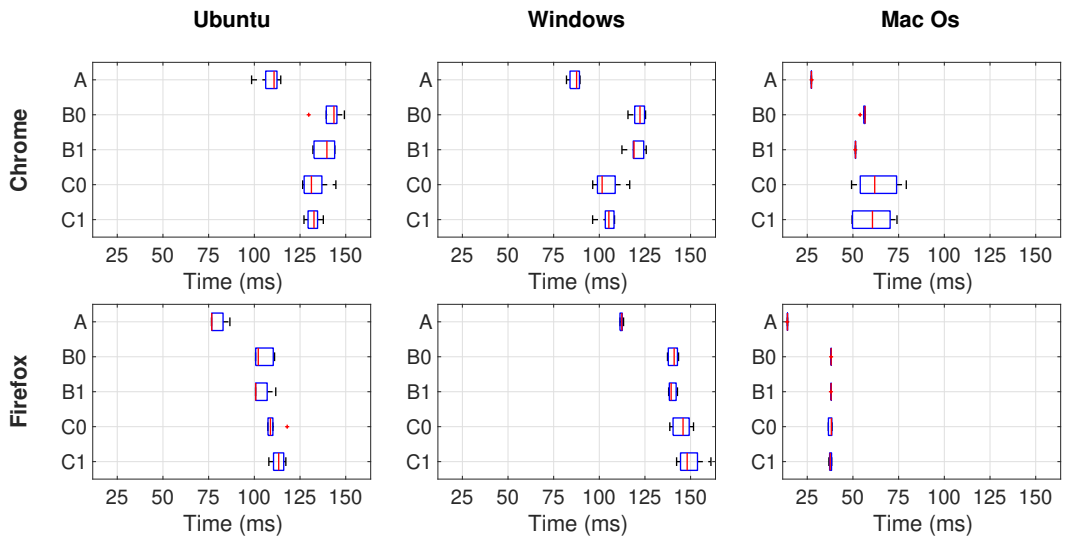


Fig. 3.6 M2E latency measurements for different setups of the audio chain with Chrome and Firefox on different operating systems. Labels correspond to the setups (A, B, C) described in Sec. 3.5 either with the MC (0) or the SAB (1).

C. The **network layer** that corresponds to the *send* and *receive* functions which are responsible for exchanging data with the *RTCDataChannel*, thus managing the transmission and reception of each audio frame.

We tested five *ad hoc* setups of the application. Fig. 3.6 presents the results of the M2E latency assessments for different combinations of web browsers and operating systems.

As described in Sec. 3.4, to simulate a professional setup, we used an external audio card instead of the internal microphone and speakers. With respect to the results presented in [135] we noticed that the adoption of an external audio card reduces by about 20 ms the latency on macOS. On the contrary, we record a worse performance in the other OSes, likely due to the fact that, currently, other operating systems do not optimize web browser access to the audio interfaces.

Setup A represents the simplest I/O chain that is possible to implement in a browser. Here, the input layer has been directly connected to the output layer to test the baseline latency that any audio application implemented in a web browser may suffer from for just capturing and playing out the unprocessed audio, i.e., avoiding processing and networking. In this setup, both browsers achieve the lowest latency on macOS, a value that is well below the one achieved on the other operating systems:

about 30 ms for Chrome and close to 15 ms for Firefox. Firefox is faster than Chrome on macOS and Ubuntu, but not on Windows.

Setups B0 and B1 use both the I/O layer and the processing layer, but not the network layer: audio is captured, processed by the `AudioSender` and `AudioReceiver` `AudioWorklets`, and then played out within the same web browser tab. This allows us to evaluate the impact of the processing layer as well as of the data transfer between the two contexts: main thread and audio thread. In this setup, we must consider two implementations, one with the MC (B0) and one with the SAB (B1). As shown in Fig. 3.6, both implementations suffer from about the same latency. We should note, that the increase in latency with respect to the setup A is almost completely due to the presence of a de-jitter buffer in the `AudioReceiver`, which was kept for consistency. A buffer of eight audio frames is used, that accounts for a delay of 21.34 ms at 48 kHz.

Finally, setup C0 and C1 use the complete audio chain and connect two applications on two different devices, as in a real scenario. Here, the devices are connected on the same 1 Gbit/s LAN with an Ethernet cable to limit the effect of network delay (and its variation, i.e., jitter). For this setup, in the majority of the scenarios we note a small increase in the delay, but in two scenarios it happens that the delay is instead lower than the delay in setups B0 and B1.

Given the high variability of the latency measurements and the negligible impact of the network (way below one millisecond), from these results we can assume that in this scenario also the overhead of the processing and network layer is very modest. The measurements confirm that the `RTCDataChannel` can provide timely audio delivery even if designed for data, and not audio, transfer. At the same time, the artifice of using `AudioWorklet` to divert audio from the `MediaStream` to the main thread and vice versa does not significantly impact the performance of the system.

From the analysis of the contribution of each single layer of the software application to the overall M2E latency, it is clear that the main drawback in implementing an NMP tool in a web browser is due to the latency in the real-time acquisition and reproduction of the audio data, that accounts for the vast majority of the overall M2E delay. This result represents a great limitation for every low-latency real-time audio processing application that would be implemented in a web browser, and not only for NMP applications. However, the performance of Firefox on macOS, that has

a latency close to 15 ms, is not significantly far from the one achievable by native NMP software applications, that is in typically in the order of 5 ms to 10 ms.

3.6 Application Robustness

3.6.1 Jitter Measurements

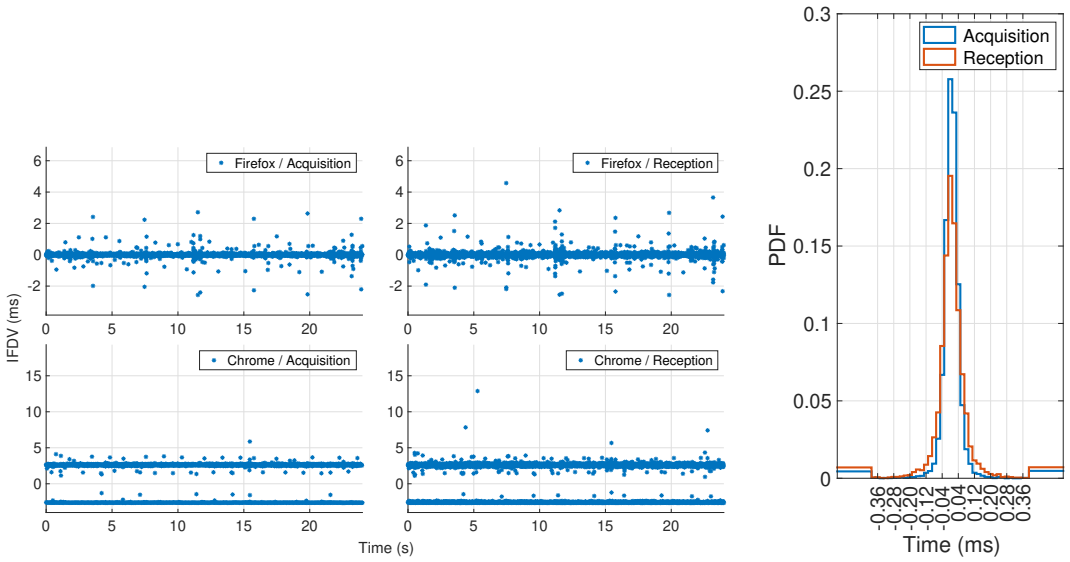
All the experiments discussed so far have been performed with a de-jitter buffer of eight audio frames (21.34 ms). However, the length (and delay) of the buffer can be set to different values, depending on the CPU load and the network conditions. Since varying delays can be introduced at different stages of the audio transmission chain, in the following, we report additional measurements of both the acquisition and the transmission stages that illustrate the real-time characteristics of the system. Note that, in order to have the same time reference at both transmitter and receiver sides, the two peers are running on the same device, in two tabs of the same web browser.

Using the high-precision timestamp API available in the web browser[11], we traced the arrival time of each audio frame at the `AudioWorkletNode`, both in the transmitter and in the receiver peer (those are the first places at both ends where we can trace the frame timestamps). The acquisition chain measurement includes only the jitter introduced by the `getUserMedia` method on the input device. The transmission chain measurement also includes the jitter due to the `RTCDataChannel` that is supposed to be slightly higher (even if the measurement is performed on the same computer as the loopback device).

Fig. 3.7a shows traces of the Inter-Frame Delay Variation (IFDV) both after frame acquisition at the sender and after frame reception at the receiver. The two peers are on the same browser, running Chrome or Firefox under low CPU load (only the application and some system and user background services). In a low-jitter setup the IFDV value should be close to zero.

Comparing the acquisition and reception traces, as expected, we see larger variation in the IFDV of the latter. This is also confirmed by Fig. 3.7b, that reports the histogram of the probability density function for the two IFDV traces measured using Firefox on macOS. It is important to notice that with Chrome there are two

[11] https://developer.mozilla.org/en-US/docs/Web/API/Performance/now



(a) IFDV (Firefox and Chrome on macOS)

(b) IFDV Probability Density Function (Firefox on macOS)

Fig. 3.7 Inter-Frame Delay Variation (IFDV) with Chrome and Firefox on macOS at the sender, after acquisition from the input device, and at the receiver, after reception from the network layer. Both sender and receiver are on the same device in order to have the same time reference.

most frequent series of values, 2.67 ms and -2.67 ms, which means that the frames are handled in pairs. This behavior can, however, appear in both browsers when the CPU load increases.

3.6.2 Scalability Evaluation

The proposed solution has been tested to verify its scalability, both with the MC and with the SAB implementation, when multiple peers are interconnected. Even though the performance with two peers is the same in terms of latency, as shown in Fig. 3.6, and no losses are present, we experience a rapid reduction in performance for the MC implementation as soon as the number of connected peers increases. Contrary to this, the SAB implementation does not report any loss in any of the audio streams, even with as many as twenty concurrent peers.

Fig. 3.8 presents the results of the measurements in terms of audio frame losses versus the number of concurrent peers (two sessions of one minute have been

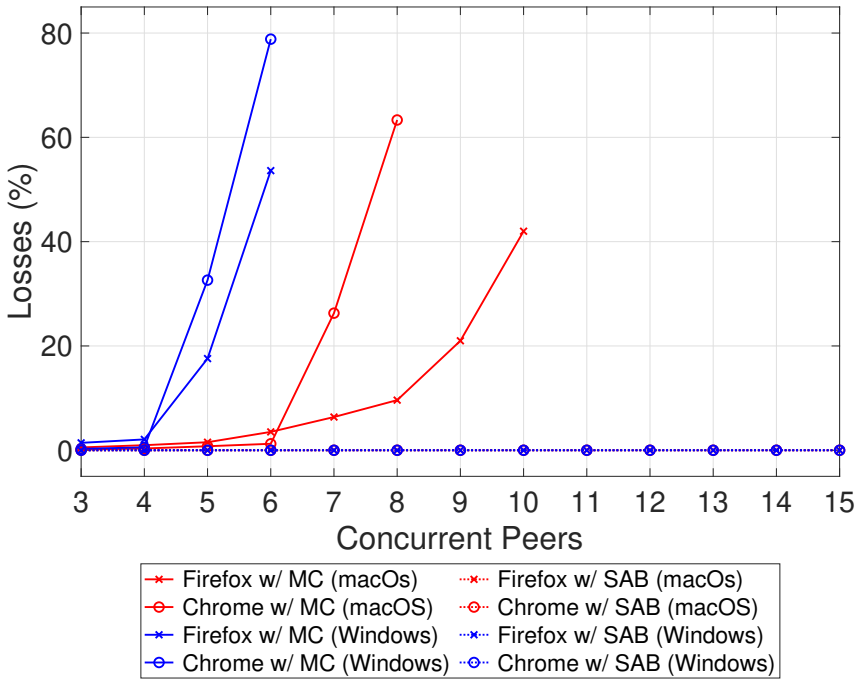


Fig. 3.8 Percentage of lost audio frames, i.e., discarded because too late, as a function of the number of concurrent peers, when MC or SAB are used.

performed for each scenario and the average loss is reported). Here the lost frames are indeed *late losses*, that is, frames that have been received too late for playback (despite the 21.34 ms de-jitter delay). Also in this scenario macOS shows better results than Windows with both the browsers, and Firefox outperforms Chrome in both the OSes.

The combination of `AudioWorklet`, SAB and `RTCDataChannel`, has proved to efficiently handle both the high bit rate (1.536 kbit/s per peer) required for the transmission of two uncompressed 16-bit PCM audio channels at 48 kHz and the high frame rate (375 frames per second) given by using small audio frames of 128 samples each.

3.7 Summary

In the time of SARS-CoV-2 restrictions, we have seen a widespread adoption of software for remote music performance by conservatories, music schools, and even professional musicians. Some of these users adopted proper NMP tools, but many

of them relied on videoconferencing applications, that are far simpler to use, in particular if web-based. However, all the current web-based audio communication solutions suffer from high latencies and constrained audio quality, due to the fact that the WebRTC's `MediaStream` (sent through the `RTCPeerConnection`) is primarily designed for speech interaction and its API does not provide ways to customize its behavior.

To the extent of our knowledge, this is the first work that presents a web-based solution (i.e., based on Web Audio and WebRTC) for NMP with the quality of uncompressed stereo PCM audio and with far reduced M2E latency with respect to the state-of-the-art. On one side, the results presented in this chapter show that it is possible to achieve the desired audio quality, robustness, and scalability with such an implementation. On the other side, the achieved M2E latency appears to be just above the requirements of NMP applications only on macOS with the Firefox web browser; other scenarios still present too high latency figures. Nevertheless, the current implementation may be used in some scenarios where high audio quality is pivotal and low-latency is not particularly important, such as remote individual music lessons.

Investigation of the different layers of the audio communication chain reveals that quite a high portion of the overall latency is due to the audio acquisition and reproduction layers that are implemented in the native code of the web browser and are not under control of the JavaScript programmer. Thus, we advocate for the reduction of browser latency in audio I/O, potentially by allowing web browsers to access the low-latency audio drivers provided by the various OSes, and inclusion of uncompressed audio in the `RTCPeerConnection` API with the possibility to customize media streams for music applications.

This chapter focused mainly on the exploration of an alternative approach to capture, transmit, and reproduce uncompressed audio within a web browser, achieving lower latencies compared to traditional WebRTC applications. Though, some challenges are still to be addressed to transform the presented prototype into an actual application. Future work could explore the implementation of algorithms specifically designed for the web context to tackle remaining issues. These could include the development of a dynamic de-jitter buffer to adapt to the varying network conditions, techniques for mitigating clock drift between clients, and packet loss concealment strategies optimized for the constraints and characteristics of web applications.

Chapter 4

Networked Music Performance: A Unified Native and Web Approach

The urge for high-speed Internet connectivity, that we have witnessed since the origins of the web, has now have paved the way for high-quality uncompressed audio communications, and, whenever possible, with low acquisition and processing latency. Professional low-latency audio transmission applications aimed at supporting networked music sessions have traditionally been developed as native software (see Sec. 2.2.1), specific to each operating system, with consequent difficulties in installation, configuration, and use. This has limited their use to a niche of computer-savvy musicians. On the other hand, web-based solutions are still lacking in terms of performance when used in a musical context (see Chap. 3).

To address this, in this chapter, we present a software application that provides a unified environment, both native and web-based, for high-fidelity interactive multi-user musical performances across the Internet. The web-based system is shown to outperform, in terms of perceived audio quality, popular video-conferencing applications that adopt state-of-the-art audio compression techniques. Subjective quality assessments confirm that, given adequate network conditions, users involved in remote audio communications experience performative conditions close to those of in-presence musical interactions, even if at several miles of distance.

The work reported in this chapter is based on two contributions: one presented in the XXIII CIM [48], and the other presented in the 10th Convention of the European Acoustics Association (Forum Acusticum) [50].

4.1 Introduction

The growing interest in NMP systems, driven by the needs that arose from the confinement imposed during the pandemic, has highlighted the existence of numerous software solutions for playing online. However, despite the existence of several experimental and commercial NMP solutions, their adoption outside highly specialized user communities has been, up to now, quite limited (see Sec. 1.1). The main reason is that these solutions often require a high level of technical-IT knowledge, making them inaccessible to non-expert users. In contrast, widely used videoconferencing tools (e.g., Skype, Google Meet, Zoom) have become the go-to platforms for remote music teaching and collaboration [24, 25], despite their apparent limitations in terms of performance and audio quality. The preference for easy to use solutions, at the expense of audio quality and latency, highlights the need to improve the ease of use of existing NMP systems.

With the development of a hybrid web-based and native application, the research described in this chapter aims to illustrate that the user's preference for less sophisticated solutions can be addressed without compromising on the overall QoE in terms of audio quality and latency, thus catering to the high-fidelity audio demands of professionals while still being intuitive to use. The objective is to design: i) a web application, that can be used directly from a web browser, without having to install anything on the user device, and which can provide high-quality audio to support mainly music teaching applications; ii) a native application, running on a dedicated hardware device ("box") aimed at supporting real-time interactions, where both high-quality audio and low latency are required. The native application is controlled through the same web-based GUI.

After a brief overview of the state of the art in Sec. 4.2, the description of the architecture of the proposed solution follows in Sec. 4.3, while the audio quality and latency results are reported in Sec. 4.4. Sec. 4.5 provides a summary of this chapter.

4.2 State of the Art

The vast majority of NMP applications (see Sec. 2.2.1) are native applications, i.e., software developed for a specific target platform (e.g., Windows, macOS, Linux) in a compiled programming language supported by that platform (e.g, C/C++). On

the other hand, web applications are written in a more "generic" language that is interpreted and executed in the web browser context, allowing them to be used on any platform without the need for adaptation or installation. Even though specific NMP applications with dedicated functionalities are far better in terms of audio quality and M2E latency, during the SARS-CoV-2 pandemic many users opted for well-known videoconferencing software, preferring ease of use [24, 25] over superior audio performance [17] (Sec. 1.1). This was particularly true for remote music teaching purposes, where Zoom [140], Skype [141] and Google Meet [136] were often used. Moreover, this was also due to their availability, as often the university/conservatory already paid a business license to use that videoconferencing tool.

As the following sections will demonstrate, Zoom and Skype generally offer superior audio quality compared to Google Meet. However, Google Meet has still attracted a substantial user base, including audiophiles. One possible reason for this is the fundamental difference between these platforms: while Zoom and Skype are native applications requiring installation and configuration, Google Meet is a web-based tool that runs directly in a web browser, eliminating the need for any setup.

The trend towards web-based software solutions for remote musical interactions has also emerged in the aftermath of the pandemic. In fact, new solutions that appeared on the market in the past years adopted some form of web-based services accessible directly through a web browser, making them more user-friendly, even for those with limited computer technical abilities. However, the actual improvement of these new services focuses on the implementation of new features, such as additional audio controls and mixing options, to meet the requirements of musicians. On the contrary, in terms of audio quality, their implemented improvements are marginal and do not go beyond the ones achievable with a fine-tuning of the options provided by Skype or other web-based applications. Additionally, the communication latency remains within the range of interactive speech communications, which is far beyond the ultra-low delay necessary for remote musical interactions (see Sec. 1.1).

Web applications trade ease of use for greater limitations in performance optimization, particularly in reducing latency of audio data acquisition, processing, and transmission, as they are constrained to the functionalities provided by the web browser. However, the implementation of the Web Audio [133] and WebRTC [75] standards in all major browsers has allowed complex audio processing and transmis-

sion applications to migrate to the web: initially, video conferencing programs, and later, even software for full digital audio workstations (e.g., Audiotool [142]). However, as of currently, we are not aware of any web solutions capable of transmitting low-latency, uncompressed audio, since the basic implementation of WebRTC does not support this.

As seen in Chap. 3, even with high bit rates, audio compression hinders the audio quality and adds algorithmic delay during the encoding process. In addition, audio processing algorithms designed for voice communications can introduce audio artifacts due to automatic gain adjustment mechanisms that change audio levels. Furthermore, noise-canceling algorithms tend to dampen sustained stationary sounds, while algorithms for adapting playback speed to channel bandwidth fluctuations can alter the pitch and timbre of sounds.

This project aims to overcome the aforementioned limitations by providing a unified solution, both web and native, for the transmission of high-quality uncompressed audio. This approach attempts to find a balance between technical knowledge and ease-of-use in designing NMP solutions that adapt to a broad range of users, without compromising on performance. The prototype presented in this chapter is primarly designed around a music school context, though it can be easily adapted to other contexts.

4.3 Architecture

The software architecture consists of two main components: a *server* and a *client*. The *server* component has two primary functions: a *signaling* function, which enables both clients to exchange the necessary information for establishing P2P connections, and a *management* function, which oversees the administration of musical sessions and classes. The *client* component is available in two implementations: a *web-based* version and a *native* version. Regardless of the implementation, both versions share the same GUI, which is accessible through a web browser. Fig. 4.1 provides a high-level overview of the architecture.

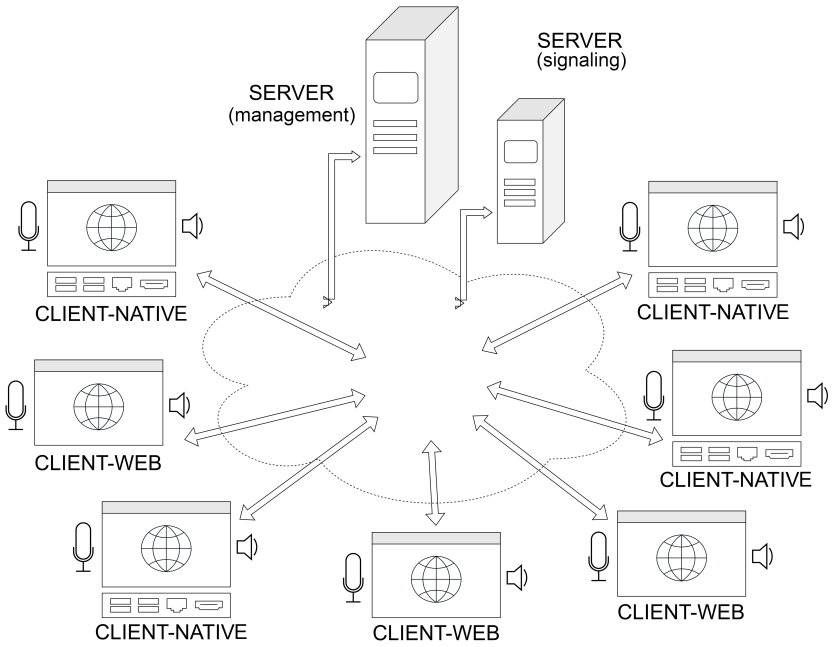


Fig. 4.1 Architecture with server components and the two types of clients (web and native) unified through the use of the same web-based GUI.

4.3.1 Server

The *management* server component provides all the functionalities necessary to manage the different roles (student, teacher, technician, secretary, and admin) and courses. It allows the secretary to register students and teachers to the platform, and to create the various courses. Upon creation, courses are then populated with the students and teachers involved in the specific course. Both students and teachers will see the courses they are enrolled in through the web UI on their personal page. Moreover, teachers are also allowed to manage the students in the course. Courses are configured to support various scenarios: a 1-to-1 lesson, a group lesson, a music rehearsal and lessons with external teachers. Users in a course can then meet in a virtual “room” to remotely play together. A virtual room reflects the same concept as a physical room (class), and corresponds to an ongoing lesson within a course. From a course, either teachers, or students, only in music rehearsals case, can create the associated virtual room, deciding the type of client device they want to use for that virtual room: browser (i.e., the web client) or box (i.e., the native client). Once the room is created, students will see on their personal page the room as active and will be able to join it by clicking on the "Join" button. The joining process is handled

transparently from the user perspective, as the *management* server communicates all the necessary information to the *signaling* server to handle P2P connectivity between the clients. Note that, while joining the room, before being connected to the other peers, the user is informed of the type of client that will be used in that room.

The *signaling* server component instead takes care of supporting the creation of connections between the various users. It keeps tracks of active rooms and takes care of selecting the appropriate signaling protocol based on the room configuration. This component is the one that abstracts away the fact that two different technological stacks are used (native and web). Note that because of significant differences in M2E latency, as will be shown later in Sec. 4.4, it is currently not possible, and in general not recommended, to connect web and native clients to the same room, particularly for the rooms that require a strict real-time interaction between musicians, like music rehearsals, as web users would experience a temporal misalignment compared to native users. The signaling is managed through a web socket connection, over which the clients can exchange all the necessary the information to establish the P2P connection. Moreover, the web socket connection is also used to provide additional functionalities, such as allowing teachers or technicians to remote control another user device for configuration or mixing related tasks. Note that, to prevent unwanted changes, this is only possible upon user approval.

4.3.2 Client

The *web-based* client is based on the solution presented in Chap. 3. This implementation is based on the Web Audio [133] and WebRTC [75] standards. The audio stream is "rerouted", by extracting audio samples from `MediaStream` through an `AudioWorklet`, and sending them over an `RTCDataChannel`. In this case, each packet sent over the RTCDataChannel contains 128 samples per channel. In addition, a `SharedArrayBuffer` provides an efficient way of exchanging data between the audio thread and the main thread, with negligible latency and processing overhead. For additional details on the implementation and its performance evaluation, refer to Chap. 3.

The *native* client is designed for more demanding scenarios that require lower interaction latencies. These extend beyond teacher-student interactions and typically involve multiple musicians performing together in real time. The implementation of

the native client, also referred to as *box*, is based on a Raspberry Pi 4B. The native client takes care of directly managing audio acquisition, playback, and transmission, while the interaction between the user and the device (configuration, control, joining/leaving rooms) happens through the web application, using the same GUI as for the *web-based* client. Native clients communicate in a P2P fashion, using UDP as the transport protocol. The packet sent by the native client does not have a fixed, predefined number of samples per channel, but adapts based on other conditions (e.g., the audio interface used, the number of channels, etc.). Though, in our tests on average we observed 112 samples per channel per packet. The rationale behind opting for a hardware component over a personal computer is that it eliminates the need for software installation and the possibility of conflicts with other software running on the same device. Additionally, it allows for the implementation of the application to suit the specific hardware, thereby optimizing computational performance and audio input/output chain. In fact, direct access and configuration of the audio card (something not available in the web context) allows it to reduce the audio latency of the NMP system to its minimum. Fig. 4.2 presents an illustrative example of the system operation, depicting the web-based GUI (purple window) and the Raspberry Pi 4B unit (bottom left). Moreover, in a music school context, the device could be managed and provided directly by the music school, thus eliminating the need for students and teachers to buy and setup up the Raspberry Pi.

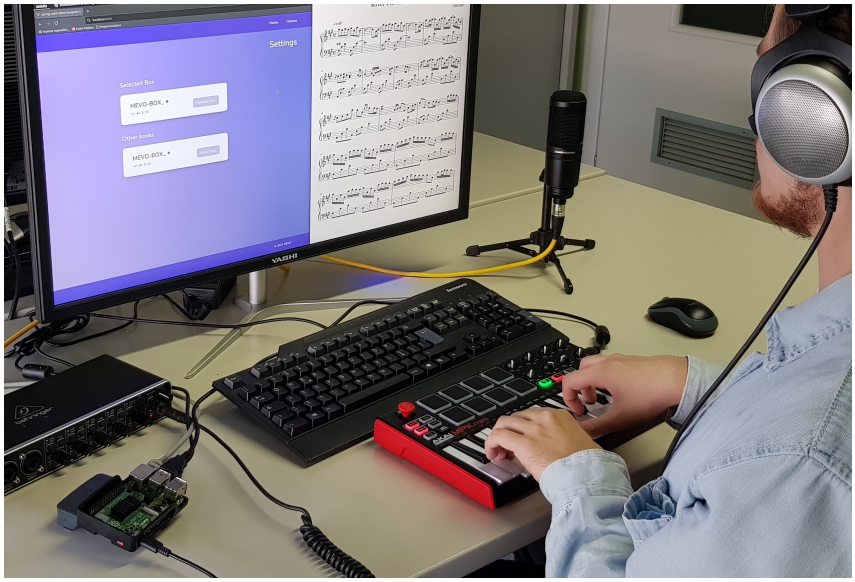
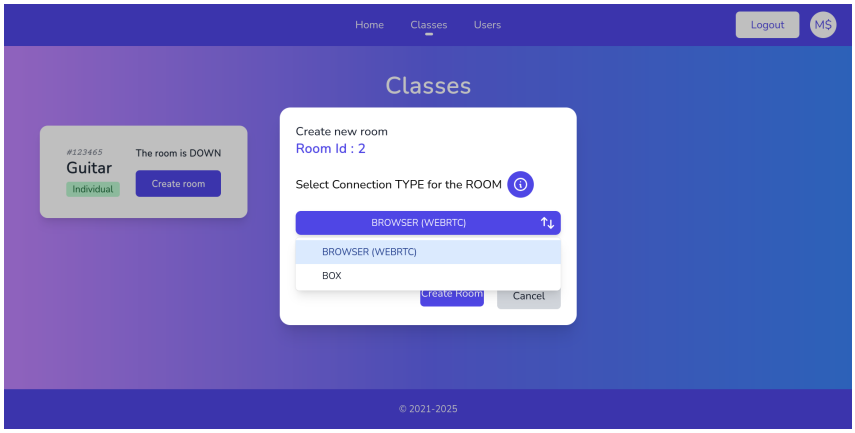
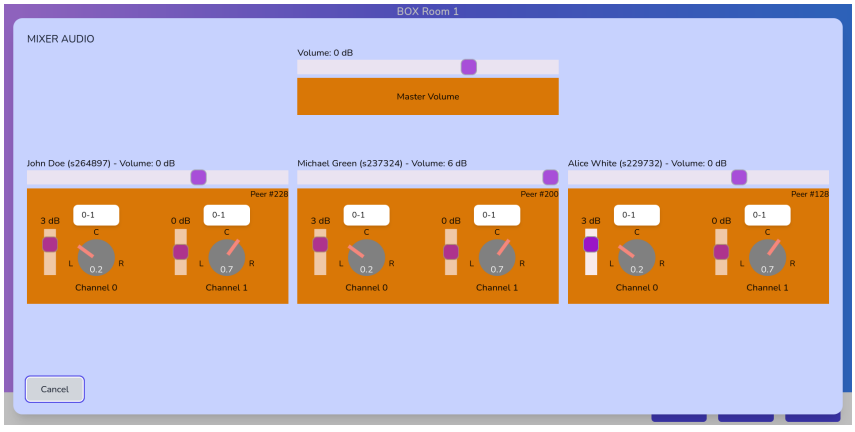


Fig. 4.2 Demonstration of the system usage, showcasing the unified web-based GUI (purple window) and the native solution (*box*) based on the Raspberry Pi 4B (bottom left).

Both types of clients are unified through the web GUI, which takes care of providing user's access to the functionalities provided by the servers. Moreover, it allows users to interact with the native client in order to configure it, by selecting, for example, the audio interface to use, the audio channels to use, etc. Finally, when the user joins a room, it provides them with all the additional audio related functionalities, e.g., audio mixing, mute/unmute, etc. Fig. 4.3 provides an overview of the prototype GUI, showcasing the selection, during the room creation, of the type of client that will be used in that room (Fig. 4.3a), and showcasing the mixing functionality (Fig. 4.3b).



(a) Selection of the type of client during the room creation



(b) When connected to a room, a user can control the audio mixing of the incoming audio streams received from the other peers

Fig. 4.3 GUI

Despite the differences in communication protocols, both communication channels, i.e., the one created by the web application and the one created by the native

application, carry the same audio format (16-bit PCM [70, 143] at 44.1 kHz). Thus, they are identical from an audio quality perspective.

4.4 Results

From a system functionality perspective, the defining feature of the proposed solution, specifically for remote music education, is the conceptualization of performance rooms as educational environments. Unlike traditional NMP solutions, where musicians interact as peers, this approach defines a hierarchy, granting the teacher (or a third-party audio technician) the ability to control users' device settings and customize the audio mix for both the instructor and the students.

From a performance standpoint, the software implementation was compared with other systems to validate the two key performance factors in this context: (a) the quality of the received audio signal and (b) the overall M2E latency. To accurately assess the best possible performance of each software, independent of network characteristics, tests were conducted in a wired 1 Gbit/s local network where bandwidth limitations, delay, and packet loss were negligible. For the same reason, the de-jitter buffer was set to the minimum possible value to ensure that: (i) it compensated for local network jitter, making packet loss at the receiver negligible, and (ii) it remained (approximately) consistent across the tested applications, ranging from 6 ms to 9 ms for native solutions and around 20 ms for web-based solutions[1].

4.4.1 Audio Quality

The choice to transmit uncompressed audio, as adopted by the proposed solution and many native NMP software applications, inherently guarantees the highest audio quality, provided there is no packet loss, thus eliminating the necessity for comparative evaluations between different applications. For the web-based solution, an indicative listening test was conducted to compare the proposed approach, which utilizes uncompressed audio (PCM), with a traditional WebRTC-based solution incorporating audio compression via Opus [74], AGC, ANS, and EC, as this rep-

[1] Note that for some applications it was not possible to set a precise value for this parameter, as the size of this buffer was computed directly by the application (as a multiple of the audio frame size) and may have been designed to dynamically adapt based on network conditions

resents one of the systems frequently used for remote music education during the SARS-CoV-2 pandemic.

The test was carried out in an anechoic chamber with a background noise level in accordance with the recommendations of ITU-R BS.1116-1 [144] by 26 subjects who self-declared to have non-impaired hearing conditions and who evaluated four stimuli corresponding to different types of audio selected from the material already used in a previous study [17]. For each stimulus, the uncompressed audio transmitted by the proposed solution was compared to the compressed audio transmitted by the benchmark applications through an *ABX*-type test [145] presented to each participant through headphones (Sennheiser 600HD).

In this procedure, three stimuli are presented to the listener: stimulus A and stimulus B, which have a known difference, and stimulus X. The listener's task is to identify whether X equals A or B. If there is no audible difference between the two signals, the listener's responses should be binomially distributed, such that the probability of responding $X = A$ equals the probability of responding $X = B$, i.e., 50%. This score is interpreted as an indication of the absence of perceptual differences between A and B.

The minimum number of correct responses needed to indicate a perceptual difference can be determined using the inverse cumulative probability of a binomial distribution based on the number of trials, confidence level and probability of a correct answer. Under the test conditions, statistical significance is achieved if at least 17 out of 26 participants correctly identify perceptual differences in the presented signals.

The four stimuli presented to the subjects contained different audio material: *i*) a bowed violin playing pitch B6 for 6.7 s (Violin), *ii*) a cello playing pitch C6 for 6.9 s (Cello), *iii*) a female soprano singing a C5 for 2.8 s (Female NV), and *iv*) a female soprano singing a G4 with operatic vibrato production (Female V). Each stimulus was then recorded after the processing and transmission with each one of the four considered applications: *i*) the proposed solution, *ii*) Zoom with the "original sound" option set, *iii*) Jitsi with AGC, ANS, and EC disabled, and *iv*) Google Meet with its default configuration, since it was not possible to customize its audio processing features.

In the first row of Fig. 4.4, the time and frequency representation of an audio track is presented, consisting of a bowed violin playing the pitch B6 quietly (-12 dB). The

subsequent rows display the audio impairments produced by various applications, including, from top to bottom, the proposed solution, Zoom, Jitsi, and Google Meet.

For each application, the SNR [146] and the Logarithmic Spectral Distance (LSD) [147] were computed and shown in the figure respectively in the time and frequency domain plot. It is acknowledged that the time domain difference and SNR computation are not significant for audio distortion. In fact, due to the involvement of an analog section in the communication path, a slight misalignment of even a portion of a sample period leads to low SNR, not justified by the listening and by the spectral difference evaluation. The spectral difference representation yields instead significant results, as further demonstrated by subjective listening tests, indicating that the proposed implementation can achieve almost transparent quality even if implemented as a web application. A quality that is clearly superior to the one offered by similar software such as Jitsi and Google Meet.

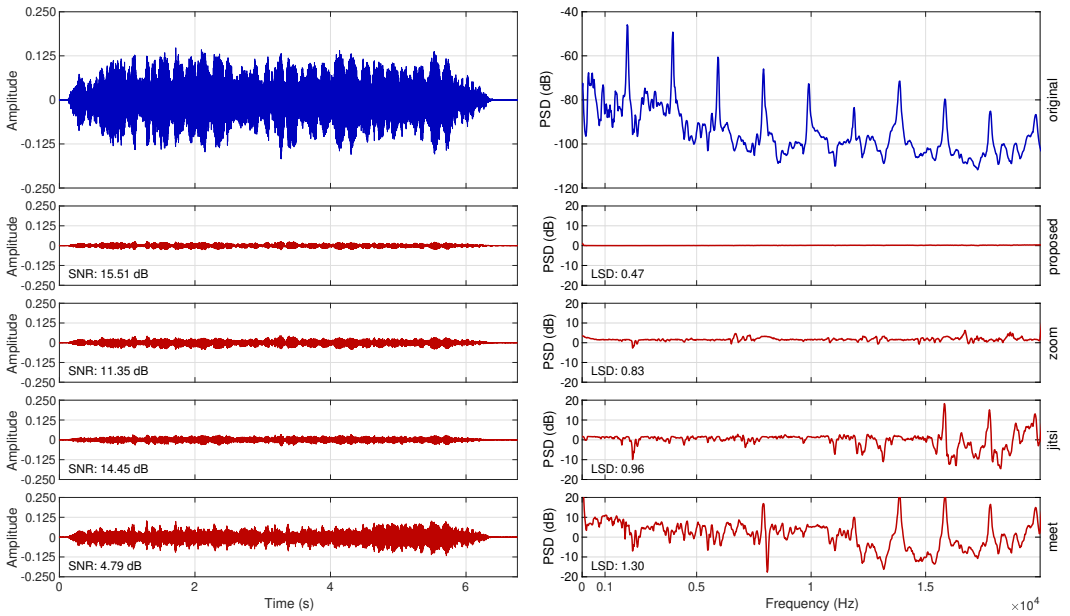


Fig. 4.4 The first row shows the time and frequency domain representation of an audio segment (bowed violin). The following rows show the distortions introduced by the proposed audio transmission framework and by three state-of-the-art videoconferencing solutions, mainly due to perceptual codec compression and automatic digital signal processing applied to the original signal. The plots also indicate the SNR and the LSD computed between the original and the processed signal for the proposed framework and then for Zoom (with the original audio feature turned on), Jitsi (with audio processing disabled) and Google Meet.

Although there is some distortion at low frequencies in the Zoom plot, the subjective tests evaluated its audio as nearly transparent. It is highly likely that the

spectral difference is due to components of the sound that are masked by the original signal and, therefore, imperceptible.

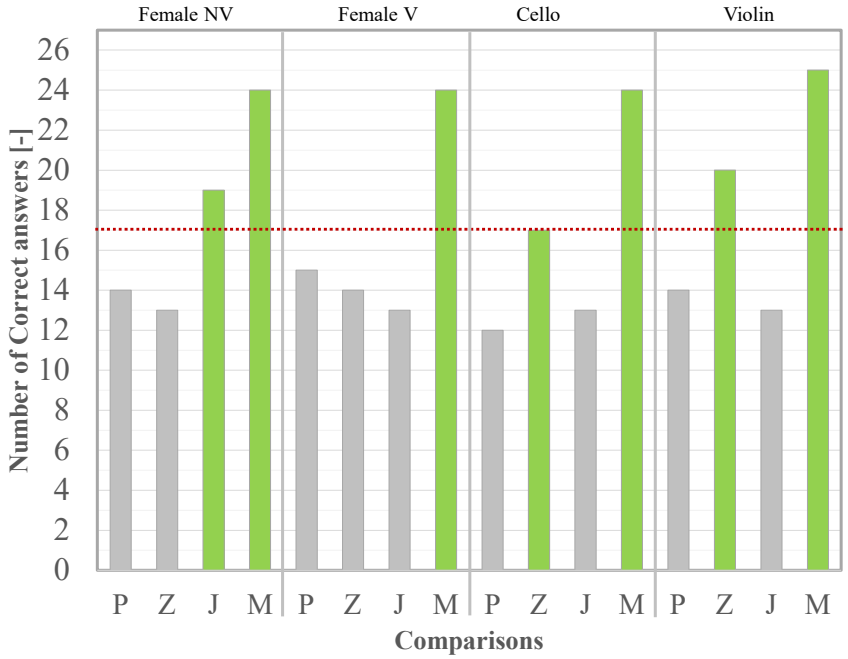


Fig. 4.5 Results of the subjective audio quality test. Four stimuli: female soprano (Female NV), female soprano with vibrato (Female V), cello (Cello), and bowed violin (Violin), processed by four different systems: the proposed one (P), Zoom (Z), Jitsi (J), and Google Meet (M), have been presented to 26 subjects. The statistical significance threshold is 17.

Fig. 4.5 shows the overall results of the subjective experiment. In the case of audio transmitted by the proposed system, only between 12 and 14 subjects out of 26 correctly identified the non-original audio in the various listening tests, while in the case of audio transmitted by a third-party web application, between 13 and 25 out of 26 people correctly identified the non-original audio. As mentioned in the previous paragraph, the results above 17 have been considered as statistically significant in order to confirm the perceived differences. The results indicate that the audio of the proposed system does not show any significant difference for all the stimuli, conversely, Google Meet presents significant differences independently of the type of stimuli. On the other hand, Zoom shows significant differences for the instrumental stimuli, i.e., Cello and Violin, while for Jitsi, a significant difference emerges for the Female singing a C5 (Female NV) only.

4.4.2 M2E Latency

The analysis of the M2E latency, whose results are reported in Fig. 4.6, revealed significantly different outcomes between the native component and the web component. The native component turned out to be on par with, or even slightly better than, other implementations used for comparison, namely SonoBus [83] and JackTrip [8]. Conversely, the web component exhibited a significantly higher M2E latency, ranging between 100 ms and 150 ms when executed on Windows or Ubuntu. However, it achieved considerably better performance on macOS, where the measured latency was approximately 30 ms to 40 ms using the Firefox web browser, and 50 ms to 60 ms when using the Chrome web browser.

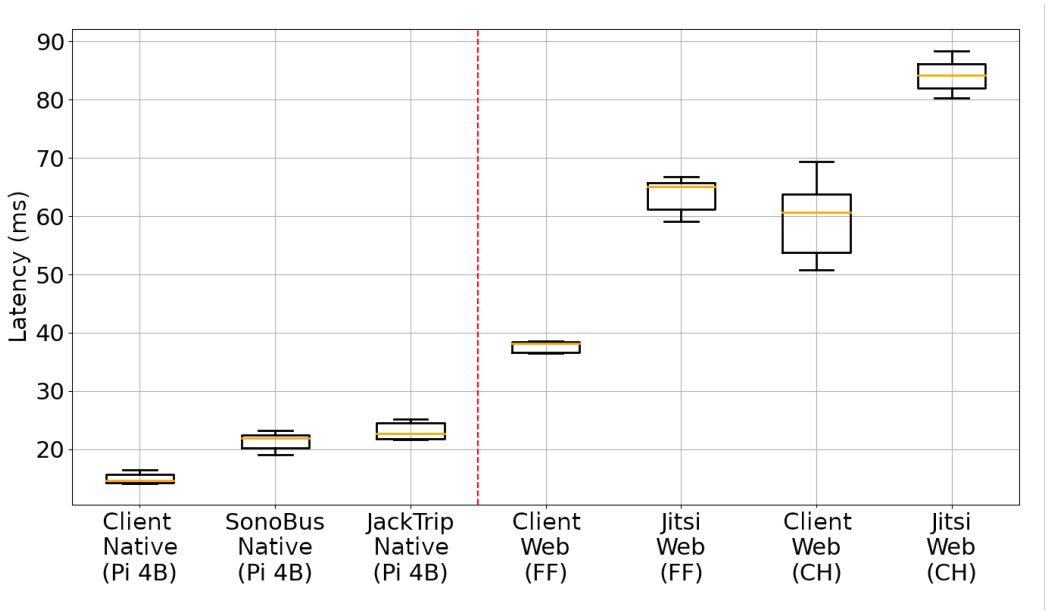


Fig. 4.6 M2E of the various analyzed solutions, including native implementations running on a Raspberry Pi 4B (left) and web implementations running on macOS (right) using different web browsers: Mozilla Firefox v.97 (FF) and Google Chrome v.98 (CH).

Although not directly comparable to the delay of native solutions, these preliminary results indicate a significant improvement over the latency observed in other web-based implementations that employ audio compression. Under the same conditions, latency for Firefox typically measures around 60 ms, while for Chrome it reaches approximately 80 ms on macOS. On Windows and Ubuntu, delays exceed 150 ms. These values align with the results presented in Chap. 3.

The improved performance on macOS is likely attributable to the better integration of the web browser within the system's audio server/API. To the best of our knowledge, on Windows it is not possible to configure the browser to operate directly with ASIO audio drivers, which would otherwise ensure lower latency. On Linux, although low-latency audio servers such as JACK are available, web browsers, in most common distributions, cannot connect to them directly. Instead, they have to interface with the system-wide audio server (e.g., PulseAudio), which introduces additional latency.

4.5 Summary

The development of a unified web and native system for NMP still presents certain technical challenges to overcome. While migrating most applications to a web environment does not pose significant difficulties, in the case of real-time audio applications, the browser's limited integration with the operating system audio APIs remains an obstacle to meeting low-latency requirements. Nevertheless, the current implementation has successfully achieved the goal of providing a web-based environment for real-time transmission of uncompressed high-fidelity audio with reduced latency compared to existing solutions, making it suitable for music education.

The integration of both the web and native components within an application framework also designed for music education lays the foundation for future field experimentation, benefiting music schools and conservatories. Enabling the remote execution of these activities facilitates student participation regardless of their place of residence, reduces travel needs, thus saving time and money, and simultaneously helps mitigate pollution caused by transportation-related emissions.

Chapter 5

FPGA-Based Low-Latency Audio Coprocessor for Networked Music Performance

NMP applications are particularly challenging to implement due to their strict latency requirements and their demand for high-quality audio. As described is Sec. 2.2.1, many NMP solutions developed in the last decades tried to overcome these obstacles by leveraging software approaches. Though, that can introduce additional time delays due to the general-purpose nature of the architectures on which they are implemented. Alternatively, a dedicated audio processor can be employed to minimize the mouth-to-ear latency.

This chapter presents the ongoing development of a hardware system that exploits an Application-Specific Instruction set Processor (ASIP) implemented on a Field-Programmable Gate Array (FPGA) to accelerate audio sample management. Specifically, a Transport Triggered Architecture (TTA) is being investigated as a processor design that aligns well with the required application domains. Preliminary empirical results indicate that the proposed solution has the potential to achieve an extremely low latency, compatible with NMP requirements. Further optimizations and enhancements are actively being pursued to address the yet open challenges posed by NMP applications.

The work presented in this chapter is part of a contribution presented in the 4th International Symposium on the Internet of Sounds in October 2023 [51].

5.1 Introduction

The main purpose of NMP is to link different performers who are not physically in the same room, making them feel like they are playing close to each other. This is usually done by leveraging an ultra-low latency audio stream over a telecommunication network. However, in this scenario, a key challenge is maintaining stable real-time communication between users, which means keeping the M2E latency within acceptable values (see Sec. 1.1). Since there are several bottlenecks in the communication, it is not straightforward to meet this requirement. To reduce latency, different software and hardware solutions have been proposed. However, since latency requirements are very strict, a purely software-based approach may not be the best option. An alternative solution, described also in [27], is the use of a system based on an FPGA, where audio packets are completely handled by dedicated hardware. In this way, the sources of additional latency present in a general-purpose architecture can be bypassed.

The solution presented in this chapter is based on the preliminary work presented in [148] and [149]. It consists of a dedicated hardware system, which combines a software approach and an FPGA-based one, designed to ensure extremely low latency. In particular, it proposes the design of an Application Specific Instruction set Processor (ASIP) implemented by taking advantage of a Transport Triggered Architecture (TTA). This architecture is particularly suitable for the NMP domain, as it can be exploited to process audio samples with minimal added latency.

Several approaches to exploit FPGAs for real-time audio processing have already been investigated in [150–152]. Though, to the best of our knowledge, apart from some preliminary attempts appeared in 2013 [153, 154], this study proposes for the first time an implementation of an ultra-low latency audio streaming coprocessor based on FPGA, specifically designed for NMP applications.

In Sec. 5.2, an introduction to the TTA processor architecture and the toolset exploited for its design is provided. The proposed system is described in Sec. 5.3, providing an overview of its components and the interfaces between them, with a particular focus on the FPGA-based audio coprocessor and its firmware. Sec. 5.4 presents the results obtained so far from a theoretical and an empirical perspective, focusing in particular on the performance of the FPGA. Finally, Sec. 5.5 summarizes this chapter.

5.2 Background

As mentioned in Sec. 5.1, the proposed system makes use of a custom TTA processor. The datapath of this architecture comprises several Functional Units (FUs), register files and interconnecting buses.

The instruction set philosophy for the TTA processor revolves around a single type of instruction, namely the "move" operation. In this system, more complex operations can be accomplished by orchestrating a sequence of "move" instructions between different FUs, which are responsible for actually processing the data. As a consequence, at compile-time, each user program is mapped to a discrete set of operations, moving data from one FU to another via the interconnected buses. This compilation process ensures the efficient execution of higher-level tasks using the fundamental "move" operation as a building block. The length of each instruction is usually very long, since it is divided into several "move" slots, each dedicated to one of the available transport buses. Therefore, a TTA processor provides a natural parallelism in the execution of instructions.

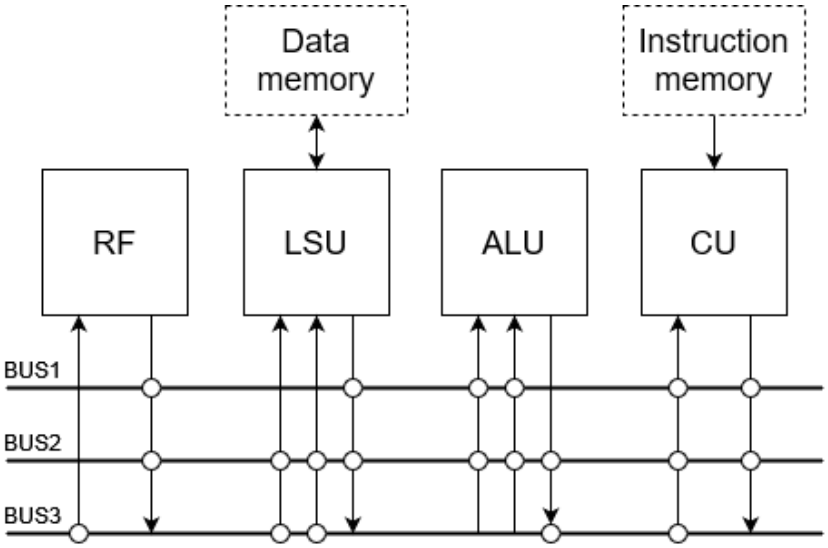


Fig. 5.1 Simple TTA architecture

A picture of a simple TTA architecture is shown in Fig. 5.1. Due to its characteristics, TTA processors are particularly suitable to implement ASIPs, enabling designers to create custom units to accelerate specific tasks. Given the focus on NMP, a TTA proves to be a valuable choice to reduce the latency of audio data handling. Specifically, tasks like audio playback, audio source mixing, and audio

stream processing (e.g., effects, equalization) can be efficiently sped up using the TTA architecture. To achieve this acceleration, dedicated FUs can be designed for each of these functions. The TTA's inherent parallelism allows these specialized FUs to work simultaneously, enabling efficient and concurrent execution of multiple NMP tasks. This approach harnesses the power of parallel processing to meet the stringent latency requirements and demanding audio processing needs of NMP applications.

To facilitate the creation of a custom TTA processor, TTA-based Co-design Environment (TCE), now known as OpenAsip [155], was adopted in this project. Developed by the Customized Parallel Computing group at Tampere University, this environment offers a comprehensive set of tools to aid users in co-designing the Register Transfer Level (RTL) architecture of a TTA processor and the high-level software running within it. Thanks to TCE, users can seamlessly design and interconnect transport buses, register files, and basic FUs, forming the foundation of the TTA processor. Furthermore, custom FUs described through a Hardware Description Language (HDL), can be integrated into the system. To enable high-level programming for the generated architecture, a custom LLVM/CLANG compiler is employed. Users can write their high-level code in the C/C++ programming languages, which the compiler then translates into instructions compatible with the custom processor. This cohesive ecosystem ensures efficient co-design and implementation of custom TTA processors, empowering developers to effectively tailor their solutions for specific tasks.

5.3 System Description

The developed system is a combination of a software and a custom digital hardware approach. In particular, as shown in Fig. 5.2, it consists of three main components, each of them implemented on a different hardware support: an audio board, an audio processor and a Raspberry Pi 4B.

The three main components of the system exploit different communication protocols to interface with each other:

- *Inter-IC Sound* (I2S): between the audio board and the audio processor.
- *Musical Instrument Digital Interface* (MIDI): between the audio board and the audio processor.

- *Serial Peripheral Interface* (SPI): between the audio processor and the Raspberry Pi.

Internally, the audio processor and the audio board can handle up to 24-bit samples, even though currently the Raspberry Pi host machine works with 16-bit samples.

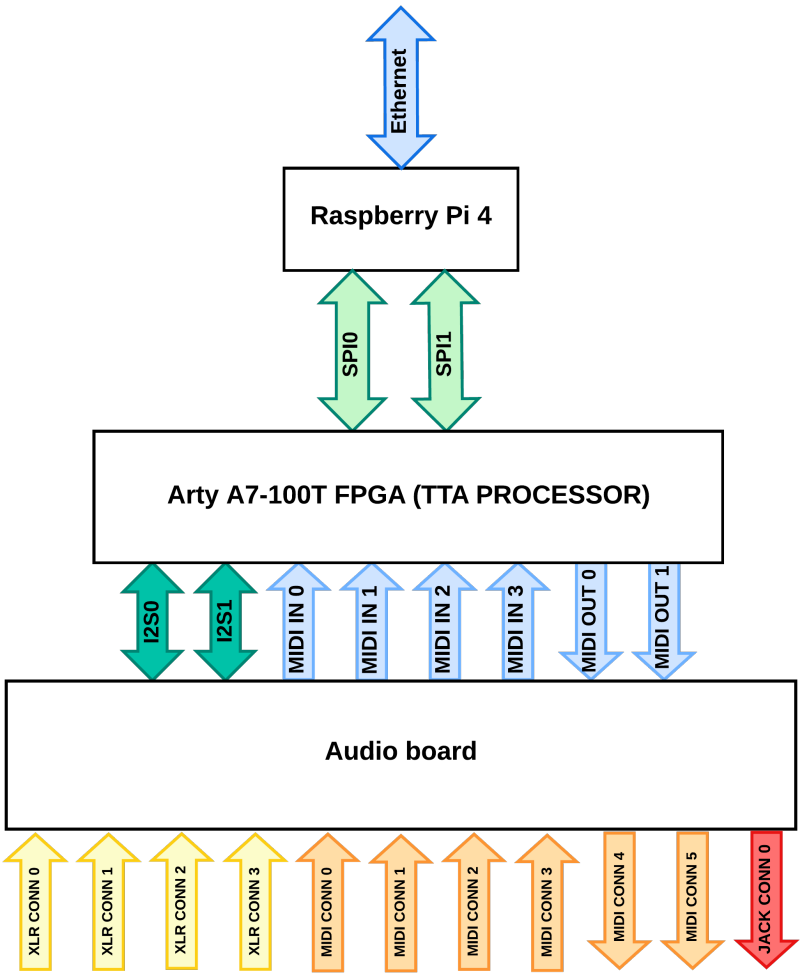


Fig. 5.2 System block diagram

5.3.1 Audio Board

The custom audio board (Fig. 5.3) serves as a hardware circuit designed specifically to function as a sound card within the system. Its conception revolves around meeting

strict low-latency requirements and addressing issues commonly associated with traditional consumer-grade sound cards based on USB 2.0.

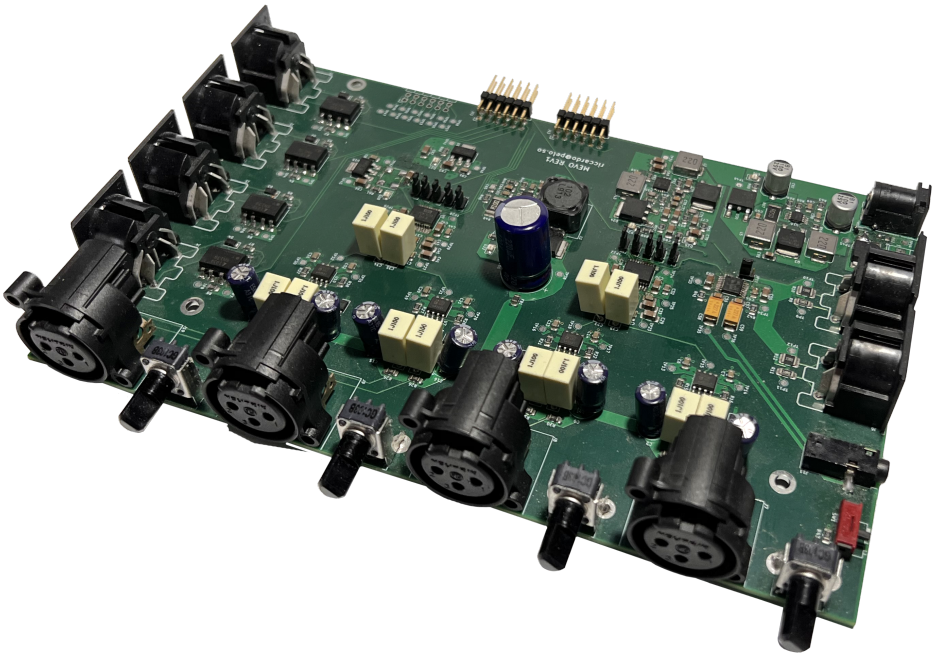


Fig. 5.3 Custom audio board

The USB 2.0 is a widely adopted half-duplex serial interface known for its versatility and ease of use. It is commonly used to connect various peripherals and devices to computers and other host systems. USB is designed to be flexible and support a wide range of functionalities, making it suitable for tasks such as data transfer, power supply and device control. However, USB is not inherently optimized for low-latency applications, such as real-time audio processing in the context of NMP. The USB protocol was originally designed to cater to various devices with different data transfer requirements, including keyboards, mice, printers, and storage devices. As a result, the USB protocol includes a complex control software stack that introduces some inherent overhead. This software stack, while necessary for managing the various devices and their functionalities, can introduce additional processing delays, which may impact real-time audio processing. The overhead introduced by the USB software stack can result in unpredictable variations in data transfer times and latency, making it challenging to achieve the precise and consistent timing required for real-time audio processing.

To ensure optimal performance, the developed audio board is engineered to efficiently acquire and play sound samples using multiple audio interfaces. It seamlessly exchanges data with the processor via the I2S protocol, employing minimal buffering and filtering. This reduction in buffering and filtering helps minimize added delays and optimize latencies. The board is equipped with readily available commercial audio Analog-to-Digital Converters (ADCs), specifically two PCM1803 [156], and a Digital-to-Analog Converters (DACs), the PCM1771 [157], which can be directly employed as a headphone amplifier. MIDI [5] interfaces are derived by converting Universal Asynchronous Receiver-Transmitter (UART) messages through commonly used optically isolated circuits.

The custom audio board is designed to accept input DC voltages ranging from 3 V to 18 V. It efficiently derives the necessary power supplies for both the digital and analog sections through a combination of switching and low-noise linear regulators. Additionally, the board has the capability to generate a 48 V Phantom power supply to facilitate the operation of condenser microphones directly on board. Furthermore, the audio board assumes the critical role of directly interfacing with users for control purposes. It provides user controls, such as volume adjustment for each input audio channel, enabling smooth and intuitive interaction for performers and participants in an NMP session.

5.3.2 Audio Processor

The second and most crucial component of the developed system is the audio processor, implemented as a custom TTA ASIP. Its goal is to create a communication channel between the audio board and the Raspberry Pi device, introducing the lowest possible amount of latency, as well as providing custom audio processing and manipulation, thus removing the computational load from the Raspberry Pi and accelerating these tasks. To host the processor, an FPGA-based board was chosen due to its flexibility in implementing digital circuits and its possibility to be completely reconfigurable. The choice for the development board fell on the Arty A7-100T produced by Digilent, featuring the Artix-7 XC7A100TCSG324-1 FPGA chip from Xilinx [158]. This FPGA model boasts an abundance of digital resources that can be harnessed for future advancements in the audio processor's capabilities.

Fig. 5.4 depicts a block diagram of the top module of the processor. Several components are shown:

- *TTA core*: the actual processor, hosting the different FUs.
- *Instruction memory*: containing the instructions obtained by compiling the user code.
- *Data memory*: used to store data and global variables.
- *I2S clock generator*: needed since the developed architecture acts as the master device for the I2S protocol. It generates the required clocks, namely the bit clock (bclk) and the left-right clock (lrclk).
- *Phase-locked loop (PLL)*: generating the master clock, running at 11 MHz.

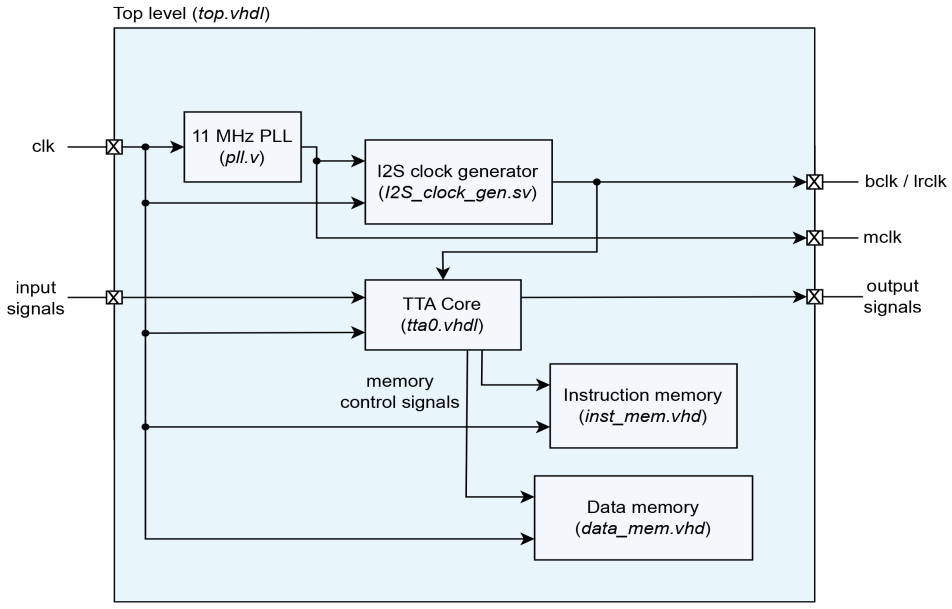


Fig. 5.4 Audio processor top-level block diagram

It is possible to observe in Fig. 5.5 that the developed TTA processor is composed of several FUs connected through four 32-bit transport buses. Some of these units are common to most architectures, such as Arithmetic Logic Unit (ALU), Global Control Unit (GCU), Load-Store Unit (LSU), two register files (“bool” and “REG_FILE_0” in the figure) and a timer (“TIMER_0” in the figure). Then, some custom units were developed in order to deal with the Light-Emitting Diodes (LEDs) and switches of

the FPGA board but, most importantly, to create peripherals that interface the system components and provide some processing on the audio data:

- *I2S_LJ_master_TX_x*: I2S transmitter with Left-Justified format and 24-bit audio samples on 32-bit frames.
- *I2S_LJ_master_RX_x*: I2S receiver. Its behavior is symmetrical to that of the I2S transmitter.
- *UART_TX_x*: UART transmitter, working at 31 250 Hz rate. It sends 8-bit MIDI messages.
- *UART_RX_x*: UART receiver, working at 31 250 Hz rate. It is used to receive 8-bit MIDI messages.
- *SPI_slave_x*: an SPI transmitter/receiver, working with 8-bits frames.
- *MIXER_0*: performs the mixing operation on multiple audio sources. It features gain and left-right panning controls for each source.
- *REVERB_0*: applies a reverb effect on an audio source. It features a dry/wet control.

The last two units of the list, as well as other possible future ones, can be exploited to accelerate time-consuming audio processing tasks typically required by musicians. Moreover, the introduction of a reverb unit can have a positive impact on the way users interact, leading to more synchronized performances [159].

All the functional units are linked together with a fully connected structure, i.e., each of them has access to all the available buses. In this way, the compiler is free to choose how to route the data and how to organize the move operations, depending on the required tasks.

For what concerns the uploaded firmware routine, the FPGA processor's task is to collect data coming from the audio board peripherals (I2S and MIDI) and route them to the SPI peripheral connected to the Raspberry Pi, and vice versa.

SPI was selected as the interconnection protocol between the FPGA and the Raspberry Pi due to its widespread availability as a standard interface for prototyping purposes. This synchronous and dependable interface is well-suited for transmitting

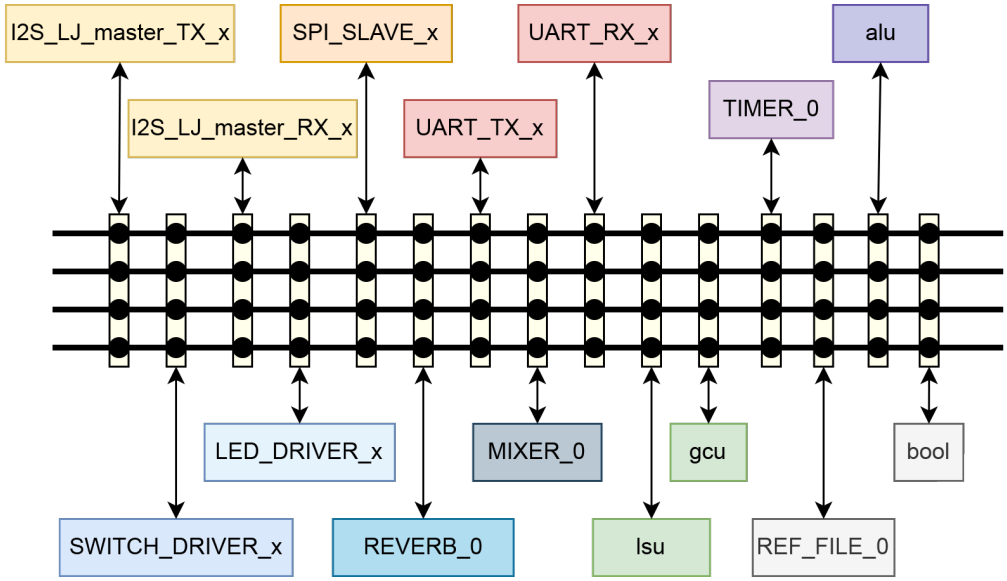


Fig. 5.5 TTA core: FUs and interconnections

versatile data payloads, encompassing both audio data and command signals. For this task, I2S has not been employed, as it is primarily designed for the transmission of stereo audio streams and incorporates an embedded clock signal. One advantage of the custom-designed TTA I2S interface is its internal buffering, which eliminates the need for a master clock. Sample re-synchronization occurs only when audio data is sent to the DAC, ensuring precise timing and synchronization. DSP tasks are executed asynchronously at the FPGA's system clock frequency. This design approach is geared towards minimizing latency, thereby ensuring that audio processing tasks are carried out with the lowest possible latency.

Support for two different operating modes has been considered, depending on whether the mixing is done at the Raspberry Pi or FPGA level. The latter option accelerates the mixing task and so speeds up the system, but will likely experience limitations for what concerns the maximum number of sources it can handle, due to the SPI link bandwidth (more details in Sec. 5.4.3).

During operation, the FPGA processor waits for a command coming from the Raspberry Pi, that is acting as an SPI master device. Three possible commands can be issued:

Command 1 is used to change the environment configuration variables, such as the audio samples bit depth, the number of active input channels on the audio board, the number of sources coming from the Raspberry Pi and others.

Command 2 starts the streaming of a packet of samples and MIDI messages from the audio board to the Raspberry Pi and vice versa. The mixing operation is performed by the Raspberry Pi.

Command 3 starts the streaming of a packet of audio samples and MIDI messages from the audio board to the Raspberry Pi and vice versa. In this case, the mixing operation is done at the FPGA level.

Furthermore, an analysis has been carried out regarding the effects of errors that can potentially occur in the SPI channel. While the impact of sporadic errors in random audio samples, resulting in occasional audio glitches, can be deemed tolerable, keeping the error rate as low as possible is crucial when issuing commands, in particular *Command 1*. In fact, if one or more of the configuration variables are not correctly received by the FPGA side, the whole communication could fail. To address this problem, a custom communication protocol has been developed. Firstly, each command is preceded by a preamble sequence informing that the Raspberry Pi is asking to start the communication. Then, the transmission continues with an ACK/NACK (Acknowledgment/Negative Acknowledgment) mechanism. Moreover, redundancy in the preamble and commands is intentionally introduced, to guarantee that the information is correctly exchanged.

In the final implementation, SPI communication packets will incorporate Error-Correcting Codes (ECCs), such as Cyclic Redundancy Checks (CRCs), to enhance the reliability of the communication channel. The ECC scheme will be supported by firmware or hardware during the packet encoding and decoding processes, thereby providing an additional layer of assurance for data transmission integrity.

Fig. 5.6 represents a flow diagram of the firmware routine, devised as a Finite State Machine (FSM) consisting of the following states:

- **MEVO_WAIT_PREAMBLE**: during this state, the processor waits for the preamble sequence coming from the SPI side.

- **MEVO_WAIT_COMMAND**: when the preamble sequence has been identified, the processor enters this state expecting a command to be issued.
- **MEVO_CONFIG**: in this state, *Command 1* is executed.
- **MEVO_STREAM**: in this state, *Command 2* is executed.
- **MEVO_STREAM_MIX**: in this state, *Command 3* is executed.

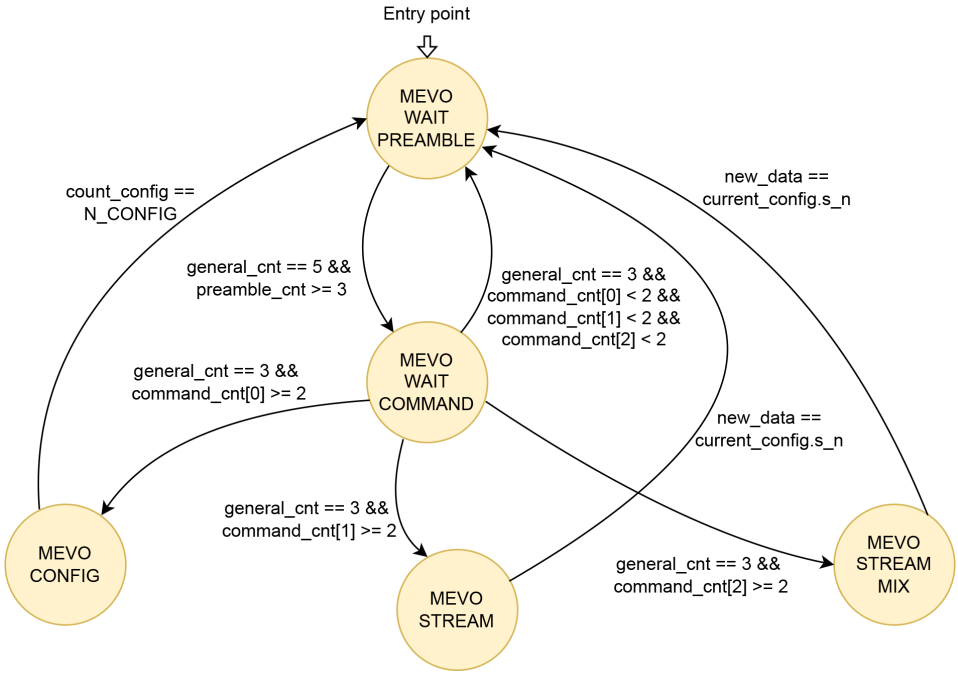


Fig. 5.6 Flow diagram of the firmware routine

Currently, the processor does not perform many operations and is clearly underused. Its real potential is still to be exploited in enhanced versions of this architecture, that are foreseen as future work. Transferring tasks from the Raspberry Pi to the FPGA offers several advantages, particularly in terms of speed. Consider, for instance, a reverb effect. In software, processed samples must continually shuttle back and forth between the processor and memory. Conversely, hardware implementation allows for efficient utilization of parallelism and pipelines, resulting in a notable reduction in computational overhead. While developing hardware solutions involves higher upfront costs, the investment may be justified in latency-critical contexts such as NMP.

5.3.3 Raspberry Pi

In the developed system, the Raspberry Pi 4B [160] plays a crucial role in managing the transmission and reception of audio packets over the network through its full gigabit Ethernet port. For an NMP context, the choice of an appropriate device was crucial, and the Raspberry Pi 4B's specifications proved it to be a fitting option. It features a 1.5 GHz quad-core 64-bit ARM Cortex-A72 processor and 8 GB of RAM, providing enough processing power and memory capacity for real-time audio processing tasks.

Moreover, the Raspberry Pi 4B offers 28 General Purpose Input/Output (GPIO) pins, providing various interface options and including support for multiple concurrent SPI communication lines. These capabilities make it a versatile platform for handling audio data efficiently. To ensure a robust and flexible system, the Linux operating system was deployed on the Raspberry Pi, benefiting from its stability and adaptability within the Raspberry Pi framework.

From a networking perspective, two crucial design choices were made. Firstly, UDP was chosen as transport protocol for communication. Secondly, a P2P architecture was adopted. Refer to Sec. 1.1 and Sec. 2.1.1 for more details on these aspects.

5.4 Results

The following sections present the results obtained so far for the proposed system. It is worth mentioning that, since the architecture design is still ongoing, most of the experimental evaluations cannot yet be assessed, in particular the ones related to the Raspberry Pi 4B component. For the same reason, a thorough evaluation of the impact of network delay and of the overall system performance will be possible only once the design phase is completed.

5.4.1 FPGA Utilization

The proposed TTA processor was implemented on an FPGA leveraging the Vivado IDE by Xilinx to evaluate the efficiency of the design and the resource consumption. The reported utilization results, shown in Fig. 5.7, indicate how the ASIP makes use

of the FPGA resources. Specifically, approximately 21% (13479 out of 63400) of the available logic elements, i.e., Lookup Tables (LUTs), were employed, while only 8% (20 out of 240) of the DSP cells were used. These outcomes emphasize that many signal processing operations on audio samples can still be handled using specific units, performing tasks which are recognized to be particularly time-consuming in a general-purpose environment.

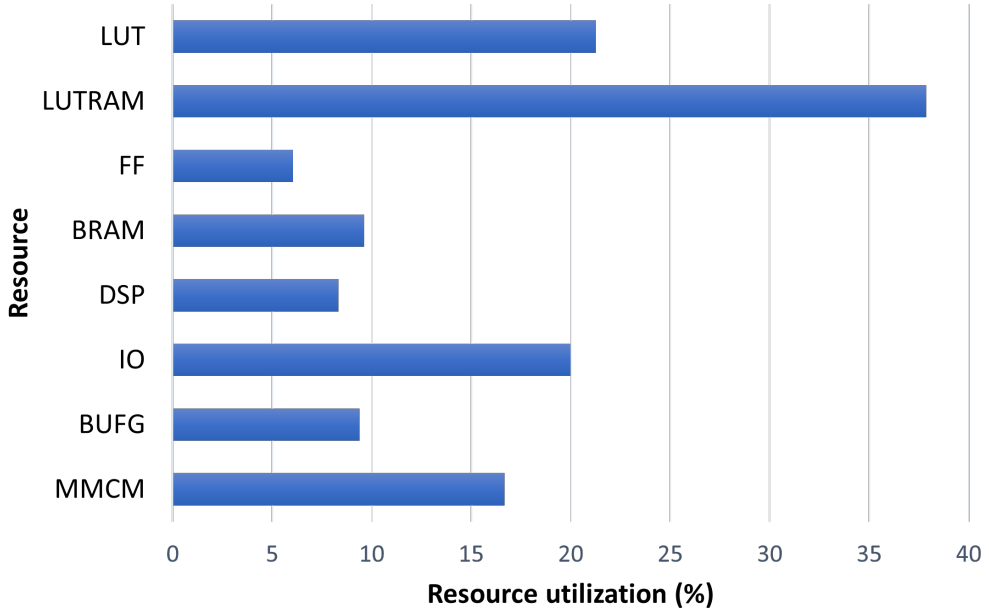


Fig. 5.7 FPGA resource utilization

Regarding memory blocks, it can be noticed that 10% (13 out of 135) of the available Block RAMs (BRAMs) and 38% (7193 out of 19000) of the Lookup Table RAMs (LUTRAMs) were used, mainly to store instruction and data memories of the TTA processor and to address the need for long delay lines in the reverb unit. This result indicates that the design made efficient use of the FPGA, leaving room for potential inclusion of additional functionalities which require to store data. Thanks to this available programmable logic, several features can be considered for future developments as custom FUs, such as new audio effects and hardware PLC techniques to cope with missing data packets.

5.4.2 FPGA Power Consumption

Power consumption is a critical factor, especially in portable or energy-constrained systems. To assess the power efficiency of the proposed FPGA-based processor, the Vivado static power consumption estimator has been used. The power consumption of the developed TTA architecture was found to be significantly lower compared to traditional software-based approaches, with a total estimated power of approximately 350 mW. In particular, in Fig. 5.8, the different power contributions are reported. The main ones are those related to the PLL circuit (indicated as "MMCM" in the figure) and to the static power consumption of the ASIP (indicated as "PL static" in the figure).

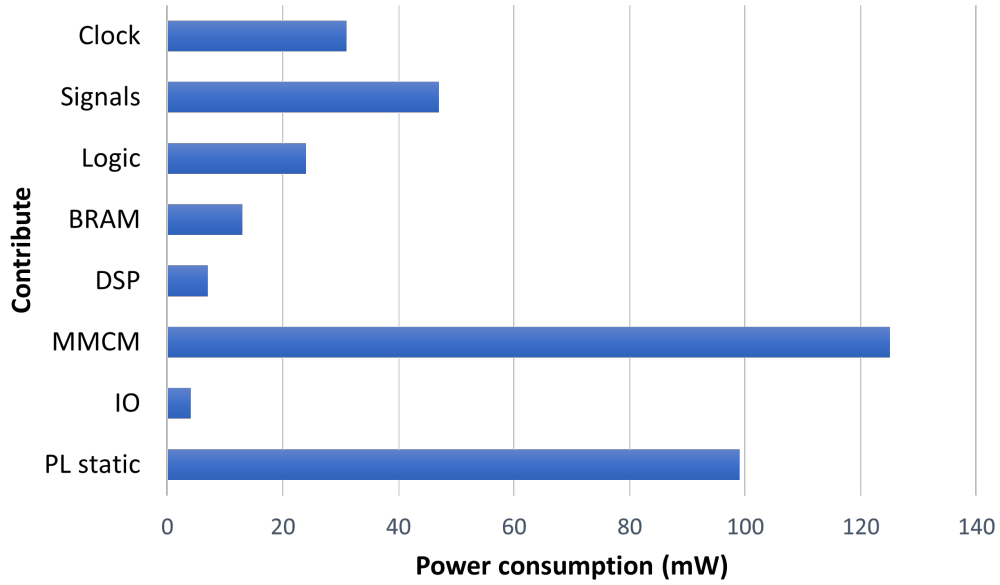


Fig. 5.8 FPGA power consumption contributions

The power consumption results obtained so far are acceptable estimations provided by Vivado. However, to understand the real power dissipation of the processor, future empirical evaluations will be considered by exploiting FPGA power monitoring capabilities.

5.4.3 SPI Bandwidth and Protocol Overhead

The required bandwidth to link the FPGA to the Raspberry Pi through the SPI interface is primarily determined by the number of audio channels being used. With a fixed audio sample rate of $f_s = 44.1\,\text{kHz}$ and a bit depth of $w_s = 16$ bits per sample, each channel requires the following bandwidth:

$$\text{BW}_{\text{chann}} = f_s \cdot w_s = 705.6\,\text{kbit/s}$$

However, it is essential to consider the overhead introduced by the communication protocol described in Sec. 5.3.2, as it affects the effective data rate. For instance, when dealing with *Command 2* (Raspberry Pi mixing mode), the protocol introduces redundancy that needs to be accounted for in the bandwidth calculation.

To quantify the overhead, a formula can be used, taking into account the sample bit depth (w_s), the number of channels sent by the FPGA through the communication link (n_{CHF}), and the number of samples inside an audio data packet (s_n). For example, with $w_s = 16$ bits, $n_{\text{CHF}} = 4$, and $s_n = 112$, the total overhead introduced in a packet of samples can be computed as:

$$\text{OH} = \frac{8}{(\frac{w_s}{8} \cdot n_{\text{CHF}} \cdot s_n) + 8} \approx 0.88\%$$

where the number 8 at the numerator represents the number of bytes required by the preamble+command sequence, while the denominator term represents the total number of bytes transferred during one occurrence of *Command 2*.

Another interesting figure is the number of audio sources that can be streamed through the SPI link, especially when the mixing operation is performed at the FPGA level. To be sure to work in “safe" conditions, BW_{chann} can be increased to a higher value, for instance $\text{BW}'_{\text{chann}} = 750\,\text{kbit/s}$. According to the datasheet of the chip BCM2711 [161], which implements the SPI communication on the Raspberry Pi device, the maximum speed the link can reach is $\text{BW}_{\text{RPI}} = 125\,\text{MHz}$. Thus, supposing that the electrical connection between the Raspberry Pi and the FPGA board permits to reach this speed, the theoretical maximum number of sources that can be streamed through the SPI link is:

$$N_{\text{SRC}_{\text{MAX}}} = \frac{\text{BW}_{\text{RPI}}}{\text{BW}'_{\text{chann}}} = \frac{125\,\text{MHz}}{750\,\text{kbit/s}} \approx 166$$

Moreover, Raspberry Pi features two SPI peripherals, that can potentially be used in parallel to increase the throughput or to evenly distribute the bandwidth requirement on both of them.

5.5 Summary

This chapter presented the ongoing design and implementation of a hardware audio streaming system designed for NMP applications. This system employs an ASIP implemented on an FPGA. The focus of this research is the exploration and design of a TTA coprocessor, as its characteristics align well with the specific requirements of NMP applications.

The results obtained so far demonstrated that the proposed ASIP-based solution achieved remarkably low latencies in local loopback tests, meeting optimal requirements for NMP communication. Additionally, lossless audio processing leads to high-quality input and output sound streams, ensuring a seamless musical interaction among remote performers. Even if the results achieved are mostly focusing on the FPGA performances up to now, the overall system can be deemed as a novel and promising approach to NMP.

The combination of software and FPGA-based hardware solutions offers an alternative approach to tackle the challenges of NMP applications, providing a significant advancement in the field of NMP. Future work could explore further optimizations and refinements in the hardware design to enhance its efficiency and versatility, opening up possibilities for wider adoption in various real-time audio applications, even beyond NMP.

Chapter 6

Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications

As explained in Sec. 1.1, NMP applications have strict latency requirements to guarantee realistic interactions between musicians. To satisfy such constraints, NMP applications typically leverage uncompressed audio and an unreliable transport protocol (i.e., UDP) to avoid unnecessary processing and re-transmission delays. Given that no guarantee on packet delivery is offered, NMP applications must deal with late/lost audio packets to mitigate the impact of the resulting audio artifacts on the quality of the playback audio stream.

This chapter explores an audio PLC technique based on AR models. In particular, it analyzes the effectiveness of AR models as a PLC method, with both quantitative metrics and subjective tests. Additionally, it investigates the algorithmic implementation of AR models, using Burg's method, and the parameter's configuration that offers the best trade-off between prediction error and computational time requirements. We analyze the computational time required to fit the model and predict future samples by considering six implementations and various compilation flags. The purpose is to find the most suitable solution capable of running on a Raspberry Pi 4B within the real-time audio boundaries imposed by NMP applications. Results

confirm that AR models can predict future audio samples more accurately than traditional PLC approaches used in NMP, which consist of filling audio gaps with silence or repeating the last received audio segment. Furthermore, results clearly show the effectiveness of the proposed solution in meeting the strict latency requirements when deployed on a Raspberry Pi 4B.

The work presented in this chapter is part of two contributions: the first one presented at the 17th International Audio Mostly Conference in October 2022 [47], and the second one, consisting in an extended version of the previous work, published in the Personal and Ubiquitous Computing Journal in May 2024 [53].

6.1 Introduction

NMP systems enable musicians in different locations to interact musically in real-time over the internet. To ensure that musicians can seamlessly interact, NMP systems must satisfy strict latency requirements and deliver high-quality audio. To satisfy latency and audio quality requirements, a common approach is to stream uncompressed PCM audio over the UDP transport protocol, which prioritizes timely delivery over data-transfer reliability. Therefore, no guarantees on the actual delivery of a packet or on its delivery time are provided. Lost packets cause gaps in the playout buffer, which generate audio artifacts/glitches in the reproduced sound. Since packet loss is unavoidable and retransmissions are incompatible with delay requirements, alternative PLC methods are essential. These methods aim to mitigate the perception of audio artifacts by filling the audio gaps in the playout buffer with reconstructed audio segments. See Sec. 1.1 for an overview of NMP and its requirements.

In this chapter, we propose a PLC method, designed to run on a broad spectrum of devices, that leverages autoregressive (AR) models. AR models are a special case of multilinear regressive models, in which a forecast variable is obtained as a linear combination of its past values. The idea is to fit an AR model on the past samples of a given audio signal and then predict the missing samples in the playout buffer before playback. This process is repeated every time the soon-to-be reproduced audio section does not contain any data, due to one or multiple missing audio packets. Thanks to their low computational requirements, AR models represent a lightweight alternative to more complex prediction approaches (e.g., those based on Machine

Learning/Deep Learning techniques [109]) and can also be leveraged as a benchmark to evaluate their performance.

First we investigate whether using AR models could improve audio quality over traditional PLC solutions used in NMP, such as silence substitution, i.e., filling the audio gaps with silence, and pattern replication, i.e., filling the audio by replicating the audio samples contained in the last correctly received packet [40]. To determine the effectiveness of AR models, we perform both quantitative measurements and subjective tests to compare their overall quality with that of traditional PLC solutions in NMP. Then, we focus on the study of the algorithmic implementation of AR models and the configuration that provides the best trade-off in terms of prediction error and computational time requirements while running the predictive model on a Raspberry Pi 4B, within the NMP real-time audio boundaries, i.e., in the order of few milliseconds. We perform such analysis by considering six different algorithmic implementations and different compilation flags.

Results confirm that AR models, even with low orders and a relatively short history of audio samples, can predict future audio samples more accurately than traditional PLC solutions. Moreover, results also indicate that running AR models in real-time on a device such as the Raspberry Pi 4B is feasible with a wide enough range of configurations, which gives the system designer freedom of choice in either providing higher audio quality with higher computational cost or favoring scalability by reducing computational costs at the expense of slightly lower audio quality.

The remainder of this chapter is structured as follows: Sec. 6.2 provides background information on AR models, with a specific focus on Burg's method. The algorithms under consideration are detailed in Sec. 6.3, followed by a description of the experimental validation in Sec. 6.4. and an analysis of the potential limitations of the proposed PLC method in Sec. 6.5. Finally, Sec. 6.6 provides a summary of the chapter.

6.2 Background

6.2.1 Autoregressive Models

The basic assumption behind time series regression models is the existence of a linear relationship between the time series of the variable of interest y_t and another time series x_t . The y_t variable is usually called a forecast variable, regressand, dependent, or explained variable. The x_t variable is usually called a predictor variable, regressor, independent, or explanatory variable. A linear regression model can have multiple regressor variables $x_{i,t}$. In such cases, it is called a multiple linear regression model. The general form of a multiple linear regression model with k regressor variables is expressed as follows:

$$y_t = \beta_0 + \sum_{i=1}^{k} \beta_i x_{i,t} + \varepsilon_t = \beta_0 + \beta_1 x_{1,t} + ... + \beta_k x_{k,t} + \varepsilon_t \tag{6.1}$$

where ε_t is the error term. The coefficients $\beta_0 ... \beta_k$ define the linear relationship between each regressor $x_{i,t}$ and the forecast variable y_t. These coefficients are estimated by fitting the model over the observed data. AR models are a special case of multiple linear regression models, where the forecast variable y_t can be expressed as a linear combination of its past values. The past values of the forecast variable are also called *lags* or *lagged values*. The number of lags, usually denoted as p, defines the order of the AR model, i.e., the number of past values of the time series that need to be used to predict the next sample. An AR model of order p, denoted as AR(p) is defined as follows:

$$y_t = c + \sum_{i=1}^{p} a_{p,i} y_{t-i} + \varepsilon_t = c + a_{p,1} y_{t-1} + ... + a_{p,p} y_{t-p} + \varepsilon_t \tag{6.2}$$

where ε_t is typically modelled as white noise and c is a constant. The order of the model can be either chosen as a fixed value, estimated while fitting the model using a given criterion (such as the Akaike Information Criterion (AIC) [162]), or selected, for example, by extracting the number of most relevant terms from the PACF of time series used to fit the model. For an overview on time series, refer to Sec. 2.1.4.

6.2.2 Stationarity Test

An AR model is stable if and only if the considered time series is stationary. Indeed, since AR models predict future values based on a linear combination of the past values, the model could diverge or have a biased trend for non-stationary time series. Since for audio traces neither of these scenarios are desirable, to use AR models for audio PLC, we first check if the considered time series is stationary. If the time series is not stationary, we must rely either on simple, traditional methods or on more complex alternative strategies, not covered in this chapter. Instead, if the time series is stationary, we can safely use AR models.

To test the stationarity of the time series, we adopt the Augmented Dickey-Fuller (ADF) test [163] as our proof. The ADF test is a statistical significance test which tests the null hypothesis that a unit root is present in a time series. Assuming the null hypothesis is true, we calculate the p-value, defined as the probability of obtaining test results at least as extreme as those observed. We then compare it against the significance level α, which can be defined as the probability of rejecting the null hypothesis, and based on the obtained result we can conclude whether the null hypothesis is rejected, thus confirming the alternate hypothesis, or not. In particular, we consider the performed test as statistically significant when p-value $\leq \alpha$. The significance level α is traditionally set to 5% or lower, depending on the study. In our case, α is set to 5%, i.e., 0.05.

The p-value is a practical tool to evaluate the "strength of evidence" against the null hypothesis, but it can also be considered as the "strength of evidence" of confirming the alternative hypothesis, which in the ADF test is the stationarity of the time series. Let us consider a simple AR(1) model defined as:

$$y_t = a_1 y_{t-1} + \varepsilon_t + c \tag{6.3}$$

where t is the time index, ε is white noise and c is a constant. We can write the regression model as:

$$\Delta y_t = (a_1 - 1) y_{t-1} + \varepsilon_t + c \tag{6.4}$$

where Δ is the first difference operator. We can conclude that the model has a unit root if $(a_1 - 1) = 0$, thus if $a_1 = 1$. The idea of the Dickey-Fuller test is to test if

$a_1 = 1$ in Eq. 6.3, which can be rewritten in a more general form as:

$$y_t = a_1 y_{t-1} + \Phi_1 \Delta y_{t-1} + \varepsilon_t + c \tag{6.5}$$

The ADF test is an augmented version of the Dickey-Fuller test, and it expands Eq. 6.5 to include higher order regressions:

$$y_t = a_1 y_{t-1} + \sum_{i=1}^{p} \Phi_i \Delta y_{t-i} + \varepsilon_t + c \tag{6.6}$$

where p is the number of lags we are considering during the test. The unit root test is carried out under the null hypothesis $a_1 = 1$ against the alternate hypothesis $a_1 < 1$. If the p-value obtained through the ADF test, is < 0.05, then we can reject the null hypothesis concluding that the time series we are considering is stationary; if instead p-value ≥ 0.05, the null hypothesis cannot be rejected and the time series is considered to be non-stationary.

6.2.3 Burg's Method

The main idea behind Burg's method for maximum spectral analysis [164] is that we can estimate a set of autoregressive parameters $(k_1, ..., k_p)$, also referred to as *reflection coefficients*, that minimize the sum of the squared forward and backward prediction errors over a given section of the time series. These coefficients are proven to satisfy the inequality $|k_i| \leq 1$, which guarantees the stability of the model. In [164], Burg also demonstrated that there is a one-to-one correspondence between these reflection coefficients and the coefficients of the autocorrelation function, which allows us to specify the second-order statistics of the time series. So, by directly estimating the reflection coefficients through Burg's method, we obtain a representation of the second-order statistic of the time series. Then, by using Levinson's recursion [165] we can compute the coefficients of the AR model.

We can define the i-th order forward and backward prediction errors at time n (respectively indicated as $f_{i,n}$ and $b_{i,n}$) for a time series y_n of length N as follows:

$$
\begin{aligned}
f_{i,n} &= a_{i,0}y_n + a_{i,1}y_{n-1} + ... + a_{i,i-1}y_{n-i+1} + a_{i,i}y_{n-i} \\
&= \sum_{j=0}^{i} a_{i,j}y_{n-j} \\
b_{i,n} &= a_{i,0}y_{n-i} + a_{i,1}y_{n-i+1} + ... + a_{i,i-1}y_{n-1} + a_{i,i}y_n \\
&= \sum_{j=0}^{i} a_{i,j}y_{n-i+j}
\end{aligned} \tag{6.7}
$$

with $a_{i,0} = 1, i \leq n < N$. The coefficients $a_{i,j}$ are used for the calculation of both forward and backward predictions errors, but in reversed order. Then we can define the sum of the forward and backward prediction errors energies as:

$$
E_i = \frac{1}{2}\sum_{n=i}^{N-1}\{f_{i,n}^2 + b_{i,n}^2\} \tag{6.8}
$$

This sum of errors is then minimized with respect to the AR coefficients, under the constraint that the coefficients satisfy Levinson's recursion [165]:

$$
a_{i,j} = a_{i-1,j} + a_{i,i}a_{i-1,i-j} \quad 1 \leq j < i \tag{6.9}
$$

The minimizing value of Eq. 6.8 is obtained by setting the derivative of E_i w.r.t. $a_{i,i}$ equal to zero, obtaining Burg's relation for the i-th reflection coefficient k_i:

$$
k_i = a_{i,i} = \frac{\text{num}_i}{\text{den}_i} = \frac{-2\sum_{n=i}^{N-1} f_{i-1,n}b_{i-1,n-1}}{\sum_{n=i}^{N-1}\{f_{i-1,n}^2 + b_{i-1,n-1}^2\}} \tag{6.10}
$$

By computing the reflection coefficients k_i as shown in Eq. 6.10, we can derive the AR coefficients using Levinson's recursion [165] as follows:

$$
\begin{aligned}
[a_{i,0}, a_{i,1}, ..., a_{i,i}] = [a_{i-1,0}, a_{i-1,1}, ..., a_{i-1,i-1}, 0] \\
+ k_i[0, a_{i-1,i-1}, a_{i-1,i-2}, ..., a_{i-1,0}]
\end{aligned} \tag{6.11}
$$

with $a_{i,0} = 1$. Let p be the model order, we compute the p reflection coefficients (k_1, ..., k_p), and for each reflection coefficient $k_i, 1 \leq i \leq p$, we iterate Levinson's recursion, ultimately obtaining the AR coefficients $(a_{p,1}, ..., a_{p,p})$. The AR model coefficients are then used to predict the next element of the time series through the AR model definition:

$$y_t = \sum_{i=1}^{p} -a_{p,i} y_{t-i} \tag{6.12}$$

It is worth noting that the original Burg's method [164] is defined for complex-valued signals. However, since audio signals are real-valued, the above formalization has been adapted accordingly.

6.3 Algorithmic Framework

6.3.1 Notation

We introduce some definitions and symbols that will be used in the description of the algorithms that implement Burg's method:

- *training set* ($\mathscr{D}$): the section of the time series used for fitting the AR model. It contains the past data samples, based on which the model predicts the missing ones. The training set size is defined by $n = |\mathscr{D}|$, expressed in number of samples.
- *test set* ($\mathscr{D}_{\text{test}}$): the section of the time series containing the actual future values, which will be used as ground truth to compute the prediction accuracy metrics. The size of the test set is denoted by $n_{\text{test}} = |\mathscr{D}_{\text{test}}|$.
- γ: the index of the first value (sample) in the time series that we want to predict. Starting from this index, we define the training set ($\mathscr{D}$) as the samples in the interval $[\gamma - n, \gamma)$, and the test set ($\mathscr{D}_{\text{test}}$) as the samples in the interval $[\gamma, \gamma + n_{\text{test}})$.

6.3.2 AR Model Implementations

In this section, we detail the implementation of the AR models considered in this study. We first evaluate the suitability of AR models for audio PLC, then focus on quantitative measurements and on identifying the optimal configuration capable of running in real-time on a Raspberry Pi 4B device. To this end, we implemented in C++ the AR models based on Burg's method. The following sections focus on the implementation of the fit and predict functions of the Burg-based algorithms.

Sec. 6.3.2.1 presents a general pseudocode for an AR model based on the traditional Burg's method [164], used for parameter estimation and future sample prediction. Sec. 6.3.2.2 introduces an optimized version of Burg's method, leveraging the order recursive formula for the denominator of Eq. 6.10, as proposed by L. J. Faber in [166]. Sec. 6.3.2.3 presents a novel implementation that combines elements of both the traditional Burg's method and the recursive denominator approach. Finally, Sec. 6.3.2.4, illustrates an alternative variant of these algorithms that incorporates error-free floating-point transformations to enhance numerical precision.

6.3.2.1 Traditional Burg's Method

The implementation of Burg's method for estimating the AR model parameters, introduced in Sec. 6.2.3, is illustrated in Alg. 1. The algorithm begins by initializing the forward and backward prediction errors with the samples in the training set $\mathscr{D}$ (lines 1-2). Then, it initializes the array containing the AR model parameters (a) and sets its first element to 1 (lines 3-4). This is needed to compute the various model parameters through Levinson's recursion. Then, the main loop (lines 5-31) repeats until the desired order (p) is reached. In each iteration (i) of the loop, the reflection coefficient for the current iteration (k_i) is computed. k_i is then used to update both forward and backward prediction errors (lines 11-18), as well as the AR model parameters (lines 20-27).

Algorithm 1 AR model - fit - Traditional Burg's method

$f \leftarrow$ training_set$[0:n]$
$b \leftarrow$ training_set$[0:n]$
$a \leftarrow []$
$a[0] \leftarrow 1$
$i \leftarrow 1$
while $i \leq p$ **do**
    num $\leftarrow -2f[i:n] \cdot b[0:n-i]$
    den $\leftarrow f[i:n] \cdot f[i:n] + b[0:n-i] \cdot b[0:n-i]$
    $k_i \leftarrow \frac{\text{num}}{\text{den}}$ ▷ Calculate the reflection coefficient k_i

    $j \leftarrow i$
    while $j < n$ **do** ▷ Update b and f
        tmp_b $\leftarrow b[j-i]$
        tmp_f $\leftarrow f[j]$
        $b[j-i] \leftarrow$ tmp_b $+ k_i$tmp_f
        $f[j] \leftarrow$ tmp_f $+ k_i$tmp_b
        $j \leftarrow j+1$
    end while

    $j \leftarrow 1$
    while $j \leq \lfloor \frac{i}{2} \rfloor$ **do** ▷ Levinson's recursion
        tmp_a $\leftarrow a[j]$
        tmp_an $\leftarrow a[i-j]$
        $a[j] \leftarrow$ tmp_a $+ k_i$tmp_an
        $a[i-j] \leftarrow$ tmp_an $+ k_i$tmp_a
        $j \leftarrow j+1$
    end while
    $a[i] \leftarrow k_i$

    $i \leftarrow i+1$
end while
return a

The coefficient computed at the current iteration ($k_i = a_{i,i}$) is then added to the array of the AR model coefficients (line 28).

Algorithm 2 AR model - prediction - Traditional Burg's method

1: $a \leftarrow a[1 : p+1]$ ▷ Remove $a[0] = 1$
2: history $\leftarrow$ train_set$[0 : n]$
3: pred $\leftarrow$ []
4: $n_h \leftarrow n$
5: $i \leftarrow 0$
6: **while** $i \leq n_{\text{test}}$ **do** ▷ Predict the n_{test} samples
7: $s \leftarrow$ history$[n_h - p : n_h]$
8: pred$[i] \leftarrow s \cdot -a$
9: history$[n_h + i] \leftarrow$ pred$[i]$
10: $n_h \leftarrow n_h + 1$
11: $i \leftarrow i + 1$
12: **end while**
13: **return** pred

Once the AR model coefficients ($[a_{p,0}, ..., a_{p,p}]$) are computed, they are used to predict the future values of the time series through Alg. 2. Since the sole purpose of the coefficient $a_{p,0} = a[0] = 1$ is to serve as a starting point for Levinson's recursion, it is removed from the array of coefficients (line 1). The prediction loop (lines 5-12) is a simple application of the AR model definition, shown in Eq. 6.12, where the past samples are extracted either from samples in the training set or from previously predicted samples.

6.3.2.2 Denominator Optimization

Though Burg's method is relatively easy to implement, it requires computing three dot products for every iteration of the main loop of the fit phase. In [166], L. J. Faber demonstrated that it is possible to rewrite the denominator of Burg's method as an order recursive formula, thus drastically reducing the complexity of the algorithm. So, according to [166], the denominator of Eq. 6.10 can be rewritten as:

$$\begin{aligned}\text{den}_i &= \sum_{n=i}^{N-1} f_{i-1,n}^2 + b_{i-1,n-1}^2 \\ &= \left[\sum_{n=i-1}^{N-1} f_{i-1,n}^2 + b_{i-1,n-1}^2\right] - f_{i-1,i-1}^2 - b_{i-1,N-1}^2 \\ &= (1-k_{i-1}^2)\text{den}_{i-1} - f_{i-1,i-1}^2 - b_{i-1,N-1}^2 \end{aligned} \tag{6.13}$$

This reduces the number of dot products to one product in the initialization step (line 5 of Alg. 3) to compute the initial denominator and one product per iteration (line 7 of Alg. 1), to compute the numerator of k_i. The initial denominator is computed as:

$$den_0 = \sum_{n=0}^{N-1} f_{0,n}^2 + b_{0,n}^2 = \sum_{n=0}^{N-1} D_n^2 + D_n^2 = 2\sum_{n=0}^{N-1} D_n^2 \tag{6.14}$$

So, the resulting algorithm is similar to Alg. 1 with the changes shown in Alg. 3. Note that lines 7-33 of Alg. 3 are identical to lines 5-12 of Alg. 1, with the only difference that line 8 of Alg. 1 is replaced with line 10 of Alg. 3.

This optimization only affects the fit phase. The prediction phase is identical to the one of traditional Burg's algorithm (Alg. 2).

Algorithm 3 AR model - fit - Denominator optimization

```
1:  f ← training_set[0 : n]
2:  b ← training_set[0 : n]
3:  a ← []
4:  a[0] ← 1
5:  den ← 2(training_set[0 : n] · training_set[0 : n])        ▷ f = b = training_set
6:  k_0 ← 0
7:  i ← 1
8:  while i ≤ p do
9:      ...
10:     den ← (1 − k_{i−1}^2)den − |f[i − 1]|^2 − |b[n − i]|^2
11:     ...
33: end while
34: return a
```

6.3.2.3 Hybrid Denominator

One limitation of Burg's method is that, due to the recursive nature of the algorithm, an error in the computation of the reflection coefficients (k_i) during the initial iterations propagates as progressively larger errors in later reflection coefficients. So, since the traditional implementation (Alg. 1) using the dot product for the denominator led to smaller numerical errors on the value of k_i w.r.t. the implementation with the denominator optimization, we propose a hybrid approach where for the first m iterations, with $1 \leq m \leq p$, we compute the denominator with the traditional algorithm (Alg. 1) and then during the next $p-m$ iterations we use the denominator optimized algorithm (Alg. 3). In our proposed version of the algorithm, by tuning the value of m it is possible to achieve a trade-off between accuracy and computational complexity. After extensive testing, we found that with $m = \lfloor \sqrt{p} \rfloor$ the errors on the coefficients computed with the hybrid approach were similar to the ones computed with the traditional method. However, a minimum number of 8 iterations was found to be necessary to ensure consistent results, as well as for lower model orders. So, in the final implementation, we set $m = \max(\lfloor \sqrt{p} \rfloor, 8)$.

Algorithm 4 AR model - fit - Hybrid denominator

```
1:  f ← training_set[0 : n]
2:  b ← training_set[0 : n]
3:  a ← []
4:  a[0] ← 1
5:  m ← max(⌊√p⌋, 8)
6:  i ← 1
7:  while i ≤ p do
8:      ...
9:      if i ≤ m then
10:         den ← f[i : n] · f[i : n] + b[0 : n − i] · b[0 : n − i]
11:     else
12:         den ← (1 − k_{i−1}^2)den − |f[i − 1]|^2 − |b[n − i]|^2
13:     end if
14:     ...
36: end while
37: return a
```

The resulting algorithm is shown in Alg. 4. Lines 6-36 of Alg. 4 are identical to lines 5-31 of Alg. 1, with the only difference that line 8 of Alg. 1 is replaced with

lines 9-13 of Alg. 4. Also in this case, the prediction phase is identical to the one of traditional Burg's algorithm (Alg. 2).

6.3.2.4 Error Compensation

In the implementation of Burg's method in a computer program, floating-point arithmetic is used. Due to the representation of a real number on a limited number of bits, all operations introduce a negligible approximation error. However, when combined in a recursive way, they can result in a cumulative error which may become no longer negligible. Thus, to reduce the magnitude of the error, we tested an implementation of all of the above described variants of Burg's algorithm using error-free floating-point transformations.

Error-free floating-point transformation is a concept that makes it possible to compute accurate results within a floating-point arithmetic. [167] describes several error-free transformations for sum and multiplication of floating-point elements, whereas [168] recalls well-known error-free transformations for real and complex numbers. Among those, we opted for the usage of TwoSum, TwoProductFMA and the Dot2, which allows performing sum, product and dot-product with doubled precision. Thus, we implemented all the variants of the Burg's method, replacing sums, products and dot products with the transformations mentioned above. The operations performed by the three chosen error-free transformations are reported In Alg. 5-7 for completeness. The interested reader is referred to the aforementioned references for a thorough discussion.

The rationale behind this choice is to provide more precise implementation of the various algorithms using the `double` data type, as the Raspberry Pi lacks hardware-level support for the `long double` data type.

Algorithm 5 TwoSum

1: **function** TWOSUM(a, b)
2: $x \leftarrow \mathrm{fl}(a+b)$
3: $z \leftarrow \mathrm{fl}(x-a)$
4: $y \leftarrow \mathrm{fl}((a-(x-z))+(b-z))$
5: **return** x, y
6: **end function**

Algorithm 6 TwoProductFMA

```
1: function TwoProductFMA(a, b)
2:     x ← fl(a · b)
3:     y ← FMA(a, b, −x)
4:     return x, y
5: end function
```

Algorithm 7 Dot2

```
1: function Dot2(v1, v2, N)
2:     [p, s] ← TwoProductFMA(v1[0], v2[0])
3:     for i ← 1 to N − 1 do
4:         [h, r] ← TwoProductFMA(v1[i], v2[i])
5:         [p, q] ← TwoSum(p, h)
6:         s ← fl(s + (q + r))
7:     end for
8:     return fl(r + s)
9: end function
```

6.4 Experimental Validation

In this section, after a brief description of the experimental setup, implementation details and dataset, we present the metrics used for validating the AR models. Then, each of the subsequent subsections follows a similar structure, introducing first the specific methods used to evaluate AR models, followed by the associated results and analysis.

6.4.1 Experimental Setup

As mentioned in Sec. 6.1, the goal of this study is to implement a PLC solution for NMP based on AR models able to run in real-time on embedded devices, specifically on a Raspberry Pi 4B.

First, we evaluate the effectiveness of using AR models as a PLC solution. In this scenario, the first step is the evaluation of the stationarity of time series. Once the stationarity evaluation is done, we proceed to tune the parameters of the AR model. We then evaluate its performance using the two error metrics: MAE and RMSE, and compare the results against the ones obtained using the two benchmark solutions used for PLC in an NMP context (silence substitution and pattern replication). This

test is carried out with various audio files which belong to six different categories. Once we find the parameters which provide the best results across all the categories, we use them to generate short excerpts by artificially inserting audio gaps, which are later used to perform a qualitative analysis via listening tests. Then, we measure the time required to fit an AR model and predict the missing section, to find the best configuration in terms of prediction quality while being able to meet the real-time requirements of NMP. In our setup, the Raspberry Pi 4B is running the Raspberry Pi OS lite version 64-bit edition and is configured to turbo the four cores ARM cortex-a72 CPU to 2.1GHz, by setting the following parameters in the /boot/config.txt

```
over_voltage=6
arm_freq=2100
```

On the Raspberry Pi 4B, we run the three implementations of AR models based on Burg's method described in Sec. 6.3.2. For each implementation, we consider multiple parameter configurations. The considered AR models are compared to the two benchmark solutions. Furthermore, we measure the computational time that each model requires for both fitting and predicting the missing samples. Tests are run considering several audio files, grouped into six different categories based on the musical instrument(s) involved. Finally, an analysis of the performance of the various models on longer prediction periods is presented, to show the limitation of the model and how the error propagates in the different implementations.

6.4.2 Implementation Details

The stationarity test (Alg. 8) and the qualitative analysis (Alg. 10) are implemented in Python using the `adfuller` function and the `AutoReg` class of the `statsmodels` Python module [169].

Instead, in all other cases, we used the implementation described in Sec. 6.3.2. The implementation of all the algorithms described in Sec. 6.3.2 is done in C++, using the C++17 standard. As already noted in [166], double-precision floating-point was found to be necessary. Thus, we run all the algorithms with the double-precision floating-point (`double`) data type and teh quadruple-precision floating-point (`long double`) data type. Note that the error-compensation techniques explained in

Sec. 6.3.2.4, are applied only when using the double-precision floating-point data type. For the quadruple-precision floating-point data type, since it already works with twice the working precision of the double-precision floating-point data type, additional error compensation is not necessary.

The C++ code was compiled directly on the Raspberry Pi 4B using g++ 10.2.1 on aarch64, the ARM 64 bit architecture. Compilation was performed using the gcc compilation flags `-O3` and `-mtune=cortex-a72`. The `-O3` flag enables the highest compiler optimization level (3), which allows the compiler to perform operations that reduce code execution time. The `-mtune=cortex-a72` hints to the compiler the specific processor architecture for which the code has to be built.

During the testing, we experimented with another compiler flag: `-ffast-math`. This flag is useful in programs using floating-point mathematical operations, to allow the compiler to perform additional optimizations, by skipping some checks and making assumptions on the code. We report an extract of some of the optimizations it enables, taken from the documentation[1]: "This mode enables optimizations that allow arbitrary reassociations and transformations with no accuracy guarantees. It also does not try to preserve the sign of zeros.", thus breaking strict IEEE compliance on floating-point representation. It follows that its use may not be appropriate for all applications. One thing to note about `-ffast-math` is that, since it allows for "arbitrary reassociations and transformations with no accuracy guarantees", it cannot be applied to the error-compensated versions of the algorithms, since the error-free floating-point transformations shown in 6.3.2.4 are based on a specific execution order of the provided instructions.

For details on the optimization flags, please refer to section "Optimize Options" of the online documentation[2]. Instead, for details on compilation flags specific to the ARM processor architecture, please refer to section "ARM Options"of the online documentation[3].

The experimental datasets and the code developed and used for the purpose of this study is publicly available on GitHub[4].

[1] https://gcc.gnu.org/wiki/FloatingPointMath
[2] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html
[3] https://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html
[4] https://github.com/matteosacchetto/burg-implementation-experiments

6.4.3 Dataset

The dataset we used is a collection of 29 audio files. Among those, 23 audio files contain individual instruments or human voice, while 6 audio files contain full songs. We grouped the audio files in six categories, based on the audio type they contain. The first four categories are: Violin, Drums, Piano and Guitar, each containing 4 - 5 audio files. These four categories contain only audio files associated with a single instrument. The fifth category, named Generic, contains audio files relative to single audio sources which were randomly chosen and did not fit in any of the other five categories. In this category, audio files contain either a spoken/singing voice or a horn. The last category is named Songs and encompasses six audio files with full songs, containing a mixture of different instrumental sources. This division in categories allows us to have a more general representation of the behavior of the PLC techniques when applied in different scenarios.

Since in NMP the most common scenario involves sending one or multiple mono audio channels, each one related to a different instrument (e.g., guitar and voice), to reduce the required bit rate and the amount of equipment needed, we decided to perform a pre-processing step to convert all audio files from stereo to mono. This allows us to evaluate the performance of AR models when used to recover individual mono audio channels, and makes the proposed approach generalizable to any audio channel configuration (mono, stereo, multichannel), since we can run an AR model individually for each audio channel. Alternative approaches based on vector autoregressive (VAR) models [170, 171] could be considered to jointly work on multiple channels. The inclusion of such approaches in our implementation is left for future work. Moreover, to standardize the audio file format and simplify the loading of audio files, we converted these files to WAV files with a sample rate of 44.1 kHz, commonly used in NMP applications. The preprocessing was performed using `FFmpeg`[5].

6.4.4 Performance Metrics

We evaluate the performance using two common metrics used in time series analysis: MAE and RMSE. In the context of time series analysis, MAE is a model evaluation metric which measures the mean absolute distance between predicted values and

[5] https://ffmpeg.org/

target values of a time series y. Given a section of the time series, which we define as *test set* ($\mathscr{D}_{\text{test}}$), the MAE is defined as the sum of per-sample absolute errors divided by the number of samples in the test set. Defined as y_t the true target value of the time series at time t, $\hat{y}_t$ the predicted value at the same time instant, and n_{test} the size of the test set, we can define the MAE as:

$$MAE(\hat{y}, y) = \frac{\sum_{t=1}^{n_{\text{test}}} |\hat{y}_t - y_t|}{n_{\text{test}}} = \frac{\sum_{t=1}^{n_{\text{test}}} |e_t|}{n_{\text{test}}} \tag{6.15}$$

The RMSE instead is defined as the square root of the MSE, which measures the quadratic distance between predicted values and target values of the time series. The MSE is defined as the sum of per-sample quadratic errors divided by the number of values in the test set. The RMSE is simply the square root of that ratio. So, given the same notation previously introduced, we can define the RMSE as:

$$RMSE(\hat{y}, y) = \sqrt{MSE(\hat{y}, y)} = \sqrt{\frac{\sum_{t=1}^{n_{\text{test}}} (\hat{y}_t - y_t)^2}{n_{\text{test}}}} = \sqrt{\frac{\sum_{t=1}^{n_{\text{test}}} (e_t)^2}{n_{\text{test}}}} \tag{6.16}$$

Refer to Sec. 2.1.4 for an overview of these metrics.

6.4.5 Stationarity Test

To implement a PLC solution based on AR models, several steps must be executed. As already mentioned in Sec. 6.2.2, the first step is the evaluation of the stationarity of time series. Since we are fitting the AR model on the section $\mathscr{D}$ of the considered time series, the stationarity check does not need to be performed on the whole time series, but only on $\mathscr{D}$. So, depending on its size n we could obtain different results from the ADF test. Additionally, we know that the ADF test also depends on the number of lags (p) we are considering.

A typical UDP packet size for NMP contains 128 audio samples per audio channel. Smaller sizes may result in excessive overhead. Instead, larger sizes would increase the M2E latency due to the increase in the time needed to initially fill

the playout buffer before starting the audio playback. Since our PLC method is designed specifically for NMP scenarios, we set $n_{\text{test}} = 128$. Notably, this value allows for potential future generalization of the proposed PLC solution to NMP web applications. Those applications take advantage of the Web Audio API[6], in particular the AudioWorklet component[7], for audio processing, which operates on buffers of 128 samples. The interested reader is referred to Chap. 3 for technical details regarding NMP with web technologies (WebRTC, WebAudioAPI). Regarding n, i.e., the size of the train set ($\mathscr{D}$), since we are working with packets of 128 audio samples per audio channel, we set it to an integer multiple of that size, i.e., $n = m \cdot 128, m \in \mathbb{N}, m > 0$. The actual values we used for n are: 512, 1024, 2048, 4096 and 8192. Finally, since we aimed at predicting 128 samples, for the model order p we explored as values all powers of 2 in the range [1, 128], i.e., $p = 2^e, e \in \mathbb{N}, 0 \leq e \leq 7$.

To obtain a general understanding of the stationarity properties across audio categories, we applied the stationarity test to all files in each category and computed the average results for each category. For each audio file of a given category, we selected 100 random positions. At each position (i), we evaluated the stationarity of the time series for all combinations of n and p. The implementation of this test is described with the pseudocode in Alg. 8, where the ADF function is implemented using the `adfuller` function of the `statsmodels` Python module [169]. To verify the stationarity of the time series, we use the p-value from the ADF test. Values below 0.05 indicate that the time series is stationary, while values above 0.05 suggest non-stationarity.

6.4.6 Stationarity Test Results

As reported in Fig. 6.1, using $n = 512$ samples or $n = 1024$, resulted in a non-stationary time series for most of the 6 categories. In general, the bigger the n, the lower the average p-value. Therefore, stationarity of the time series can be typically achieved by increasing n. Conversely, by increasing p, stationarity characteristics do not become so evident, and for each category a preferential value of p appears to minimize the p-value. It also emerges that 128 lags are often excessive and, especially with low values of n, cause the p-value to increase. Thus, in that configuration we

[6] https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
[7] https://developer.mozilla.org/en-US/docs/Web/API/AudioWorkletProcessor

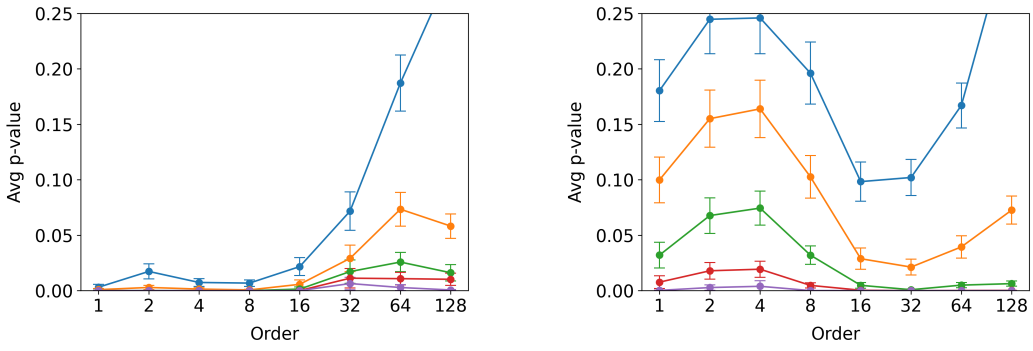
Algorithm 8 Stationarity test

```
train_sizes ← [512, 1024, 2048, 4096, 8192]
lags ← [1, 2, 4, 8, 16, 32, 64, 128]

for category in categories do
    p_res ← []

    for file in audio_files do
        samples ← load_samples(file)
        positions ← generate_100_random_indexes(samples)

        for train_size in train_sizes do
            for lag in lags do
                for pos in positions do
                    train_set ← samples[pos − train_size, pos]
                    p_value ← ADF(train_set, lag)
                    p_res[train_size][lag].append(p_value)
                end for
            end for
        end for
    end for

    m ← mean(p_res)
    interval ← confidence_interval(p_res)
    plot(m, interval)
end for
```

have no guarantee that the time series will be stationary, and in turn the resulting model may diverge.

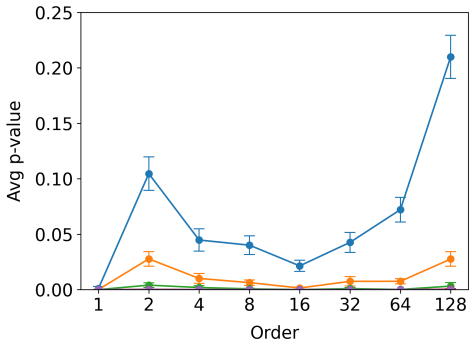
6.4.7 Fitting and Validating the AR Models

Firstly, we evaluate the performance of every algorithm across all six dataset categories in terms of MAE and RMSE. For each audio file, after loading all samples, we extract 100 distinct random positions γ that are multiples of n_{test}, $\gamma = m \cdot n_{\text{test}}, m \in \mathbb{N}, m > 0$. This approach ensures a consistent number of predictions per audio file while simulating a packetized audio stream with random packet losses. For each γ, we fit the AR model on the n previous samples, i.e., on set $\mathscr{D}$, and then we use the resulting AR model to predict the next n_{test} samples. Next, we compute

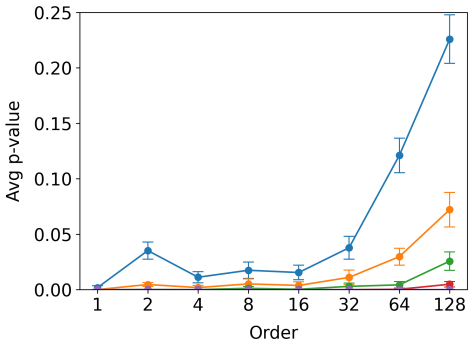


(a) Violin

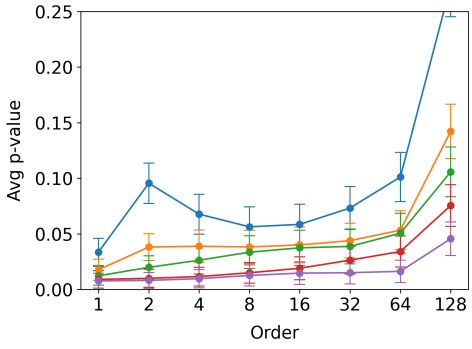
(b) Drums



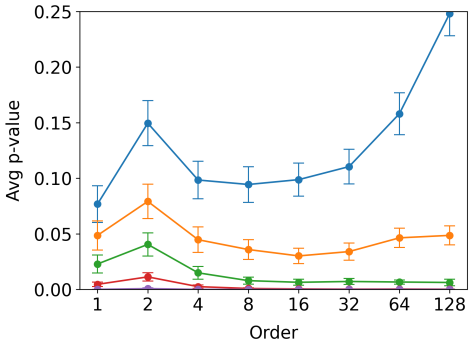
(c) Piano



(d) Guitar



(e) Generic



(f) Songs

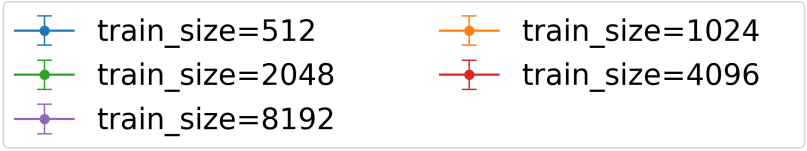


Fig. 6.1 Stationarity

the two metrics over the predicted samples and the actual samples, i.e., $\mathscr{D}_{\text{test}}$. For comparison, we also compute MAE and RMSE for the two benchmark PLC methods. Alg. 9 provides the pseudocode that implements the above-described procedure. This algorithm needs to be run for every AR model implementation. For n, n_{test} and p, we used the same values specified in Sec. 6.4.5.

Secondly, we evaluate the computational requirements of the different algorithms presented in Sec. 6.3.2. To do so, we measure the time required to run the fit and predict methods for each algorithm, for all combinations of p and n. Alg. 9 already includes instructions for the measurement of the computational time of the fit and predict phases.

Finally, we investigate how the different models perform on longer predictions. This is helpful to understand how error propagates in the three considered implementations. This test is carried out by considering a sine wave with a fixed frequency of 2 kHz, and an amplitude covering the whole digital audio range ([-1, 1]). For each model and every value of p and n, we train the model on $\mathscr{D}$ and then predict the next 20 packets, i.e., $n_{\text{test}} = 20 \cdot 128$. Every run of this algorithm uses the same value of γ, set to the max value of n, i.e., 8192. We then compute the MAE to compare the predicted samples to the actual samples, i.e., $\mathscr{D}_{\text{test}}$.

To evaluate the prediction capabilities of AR models, we initially tried to run both the `AutoReg` class of the `statsmodel` Python module, and the algorithms implemented in C++. As no significant difference was observed in the results obtained with the `AutoReg` class of the `statsmodel` Python module and the C++ implementations, we proceeded with the latter. The C++ version was selected specifically since it permitted the combined evaluation of the two chosen metrics and the assessment of computational time required. Thus, for all the steps reported in this section, we concentrated solely on the C++ version.

6.4.8 Evaluation of the Prediction Error

Figs. 6.2 and 6.3 show, for each audio category, the MAE and RMSE trend of the AR models compared to that of the two benchmarks, for all combinations of n and p. The plots shown in Figs. 6.2 and 6.3 are obtained using the traditional Burg's method (Sec. 6.3.2.1), but identical results were observed for the other two

Algorithm 9 Fitting and predicting

$n_s \leftarrow [512, 1024, 2048, 4096, 8192]$
$p_s \leftarrow [1, 2, 4, 8, 16, 32, 64, 128]$
$n_{\text{test}} \leftarrow 128$

for category **in** categories **do**
    for file **in** audio_files **do**
        samples $\leftarrow$ load_samples(file)
        $i_s \leftarrow$ generate_100_random_indexes(samples)

        for i **in** i_s **do**
            silence $\leftarrow$ zeros(n_{test})
            previous $\leftarrow$ samples$[i - n_{\text{test}} : i]$
            $\mathscr{D}_{\text{test}} \leftarrow$ samples$[i : i + n_{\text{test}}]$

            store(MAE($\mathscr{D}_{\text{test}}$, silence)) ▷ Store the benchmark results
            store(RMSE($\mathscr{D}_{\text{test}}$, silence))

            store(MAE($\mathscr{D}_{\text{test}}$, previous))
            store(RMSE($\mathscr{D}_{\text{test}}$, previous))
        end for

        for n **in** n_s **do**
            for p **in** p_s **do**
                for i **in** i_s **do**
                    $\mathscr{D} \leftarrow$ samples$[i - n, i]$
                    $\mathscr{D}_{\text{test}} \leftarrow$ samples$[i, i + n_{\text{test}}]$

                    AR_model $\leftarrow$ AR(n)

                    start $\leftarrow$ start_timer()
                    $a \leftarrow$ AR_model.fit($\mathscr{D}, p$)
                    store(stop_timer() $-$ start) ▷ Store fit time

                    start $\leftarrow$ start_timer()
                    pred $\leftarrow$ AR_model.predict($\mathscr{D}, a, n_{\text{test}}$)
                    store(stop_timer() $-$ start) ▷ Store predict time

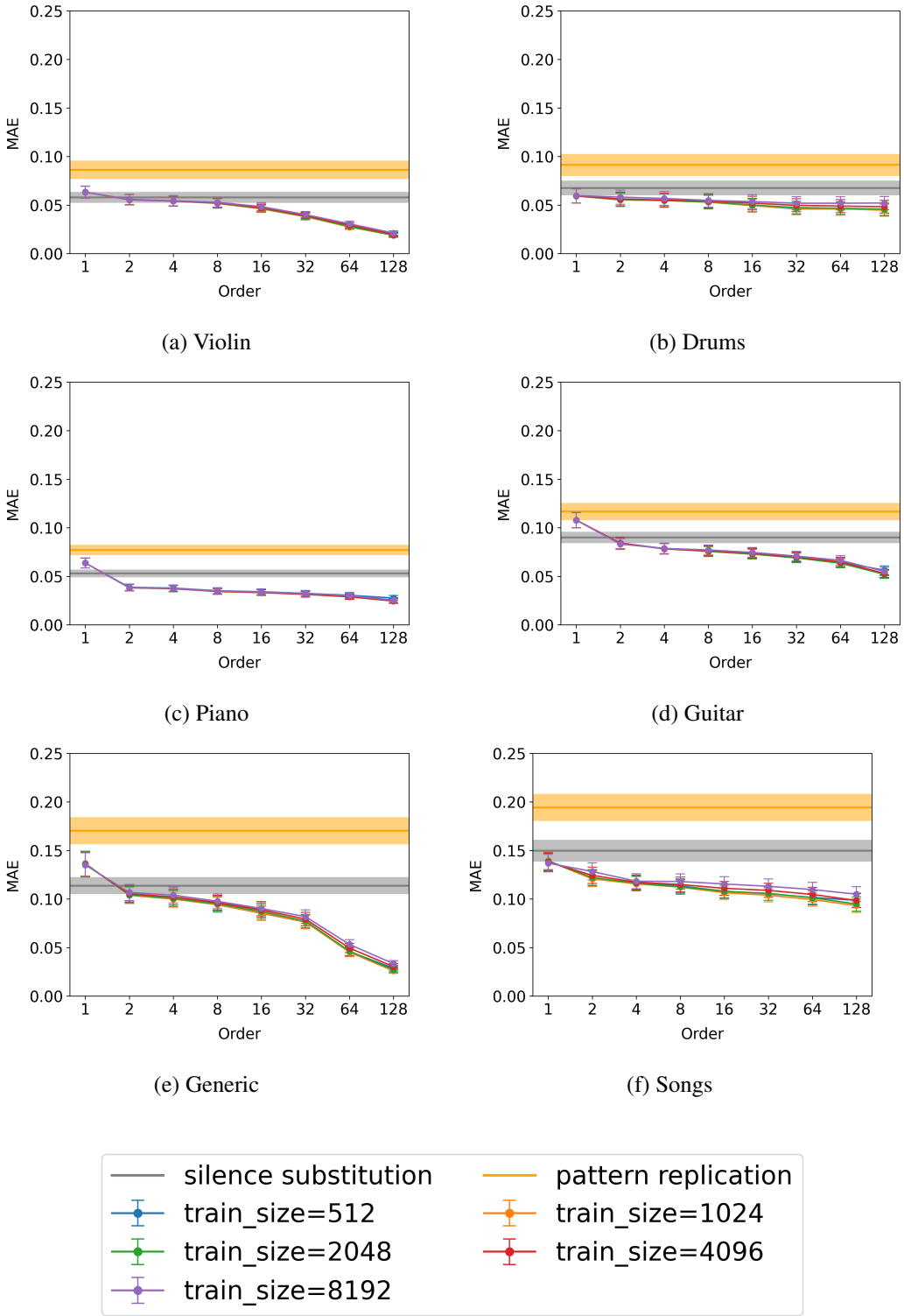
                    store(MAE($\mathscr{D}_{\text{test}}$, pred)) ▷ Store the AR results
                    store(RMSE($\mathscr{D}_{\text{test}}$, pred))
                end for
            end for
        end for
    end for
end for

algorithms (Sec. 6.3.2.2-6.3.2.3). Thus, in this subsection, we will adopt the general term "AR models" instead of referring to each individual algorithm.

Since the two benchmarks do not have any dependency on n and p, their MAE/RMSE values are displayed as horizontal lines, with shaded bands to indicate confidence intervals. The gray line represents the error obtained using the silence substitution technique, whereas the yellow line represents the error obtained with the pattern replication technique. The other lines, instead, display the error measured with AR models for different values of n and p. Results show that, in general, the variation of n only mildly affects the prediction error, with lower values of n resulting in a slightly lower error. This has two main effects. First, the possibility of using lower values of n limits the number of past packets to be memorized, which reduces memory consumption. Second, in Burg's method, to calculate the numerator and denominator of Eq. 6.10 we compute the dot-products of f and b. Since f and b depend on n, smaller n means smaller size vectors to multiply, hence lower computational cost.

Moreover, results show that p is the most impacting parameter to tune: generally, the higher p, the lower the error. However, increasing p means increasing also the computational cost of both fitting the model and predicting future samples. Thus, in a real scenario, a trade-off emerges between prediction quality and computational cost. Note that, depending on the input signal, it is not always necessary to select high values of p, since in some cases the improvement in prediction quality is negligible, while the increase in computational cost is significant. Thus, to use AR models for PLC, we first need to find the best configuration of n and p, according to the specific runtime context (i.e., computational resources, timing constraints, etc.). Then, we could use a criterion such as AIC [162] to allow for a potential early stop of the fitting phase, with the aim of reducing computational cost for easy-to-predict signals.

One apparently counterintuitive outcome is that, by replicating the previous packet, we obtain higher errors than those obtained with silence substitution. The reason appears to be that the replicated audio segment, while similar in shape, is often phase shifted w.r.t. the lost audio segment. This phase shift creates a push-pull effect in the audio waveform. Consequently, the difference between actual and replicated samples becomes larger than the difference between actual samples and zero, resulting in higher errors than those obtained with silence substitution. An example of the phase shift between the waveform generated by the pattern replication



(a) Violin

(b) Drums

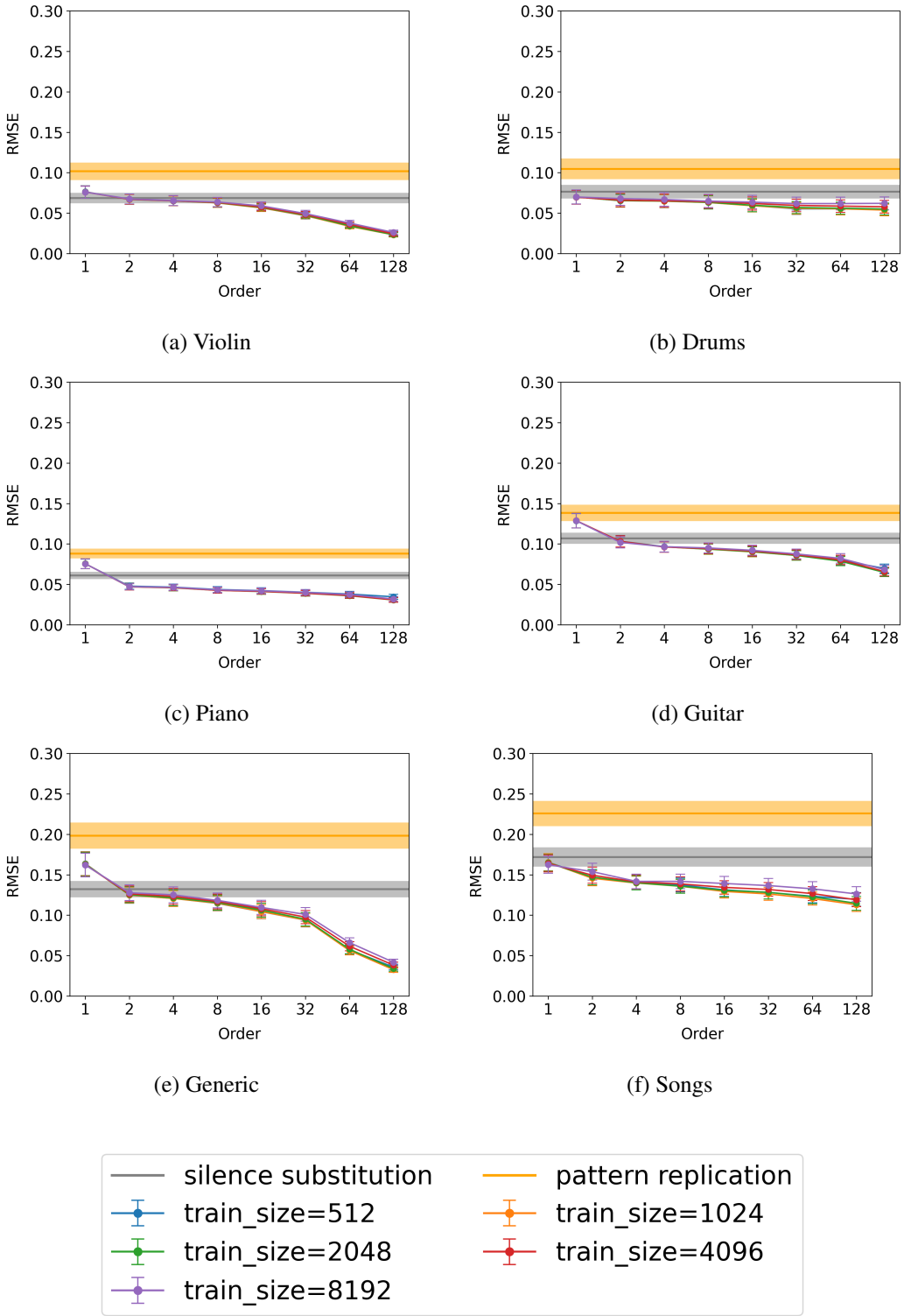
(c) Piano

(d) Guitar

(e) Generic

(f) Songs

Fig. 6.2 Performance of AR models in terms of MAE



(a) Violin

(b) Drums

(c) Piano

(d) Guitar

(e) Generic

(f) Songs

Fig. 6.3 Performance of AR models in terms of RMSE

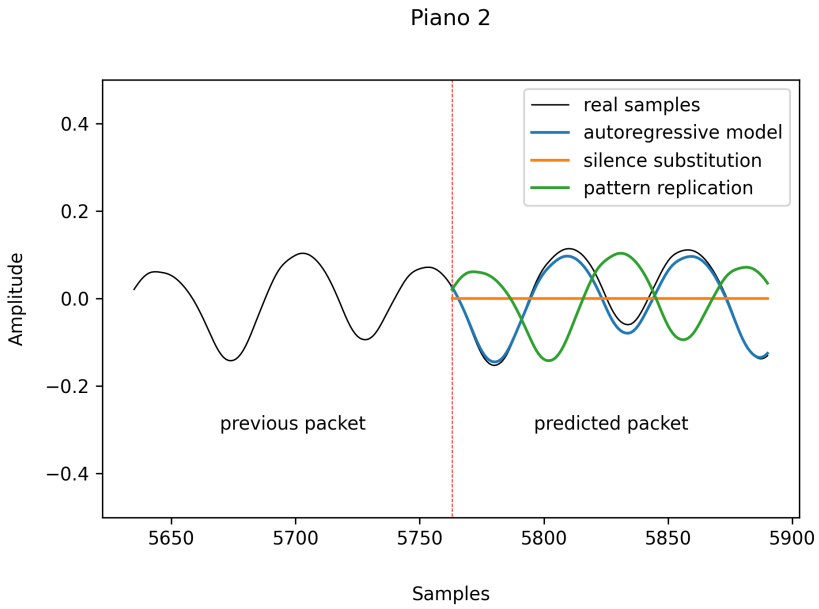


Fig. 6.4 Visual comparison of the reconstructed waveform by the three PLC techniques

PLC technique and the actual waveform is visible in Fig. 6.4, which displays the resulting waveform created by the different PLC solution when reconstructing the same packet.

In general, we can see that, the more sustained the sound is, the better AR models can reconstruct missing samples. Conversely, the more transient-oriented the sound is, the less effective AR models are, showing a trend very similar to the one obtained by silence substitution. This can be clearly observed in Figs. 6.2 and 6.3, where in the categories Violin, Piano and Generic, all of which contain more sustain-oriented sounds with less abrupt transitions, AR models show a greater improvement w.r.t. silence substitution and pattern replication. Instead, transient-oriented sounds like those in the categories Guitar (which contains some strumming tracks), Songs (which features multiple instruments, among which drums), and, most notably, Drums (i.e., the most transient-oriented sources), the improvement is far less noticeable. Especially in Drums, we can see that AR models have a trend very close to that of silence substitution. This is due to the inability of AR models to learn any significant pattern, and the resulting predictions often quickly decaying to zero.

Notably, Figs. 6.2 and 6.3 show that, while the improvement in prediction quality varies from category to category, AR models tend to outperform the two benchmark

approaches already with $p = 4$. This means that, in a real scenario, there is a fairly wide range of configurations we can choose from; in other words, we can adapt in real-time the AR model parameters to the current runtime conditions.

6.4.9 Evaluation of Computational Requirements

Given a number of samples n_s and a sampling rate f, to compute the time interval in which the n_s samples are generated/consumed we can simply divide n_s by f:

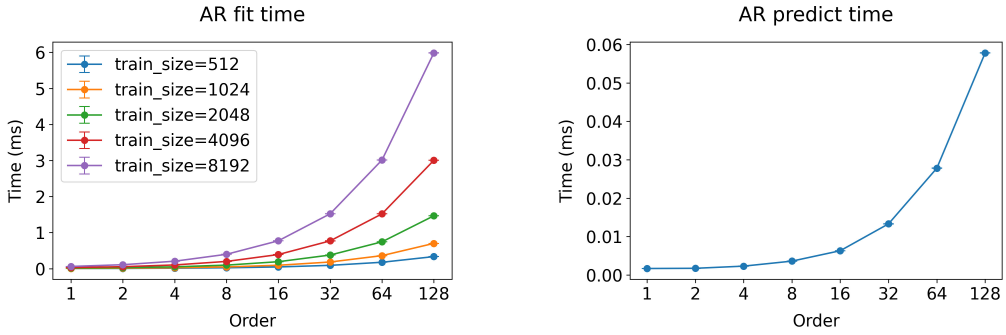
$$t_s = \frac{n_s}{f} \tag{6.17}$$

In our specific case, $n_s = n_{\text{test}} = 128$ and $f = 44.1$ kHz. Thus, in our scenario, the NMP application generates and consumes packets every $\frac{128}{44.1}$ ms ≈ 2.90 ms. This means that, to apply our PLC solution, we need to fit the AR model and predict the next n_{test} samples within 2.90 ms.

The general trend is shown in Fig. 6.5a, in terms of computational time for fitting the model, and in Fig. 6.5b in terms of computational time for predicting the next n_{test} samples. Both plots were generated using the traditional Burg's method (Sec. 6.3.2.1) with the double-precision floating-point data type (`double`) and no `-ffast-math` compiler flag. It is worth nothing that similar trends were observed for all other algorithm variants (Sec. 6.3.2.2-6.3.2.3) and configurations, differing only in the scale of computational times. While the absolute values vary between algorithms, the relative effect of parameters n and p on execution times remains consistent.

In general, replacing the traditional Burg's method (Alg. 1) with Burg's method with denominator optimization (Alg. 3) reduces the fitting time of a factor between 1.5 and 2. Instead, substituting the traditional Burg's method (Alg. 1) with our proposed hybrid method (Alg. 4) reduces the fitting time by a factor between 1.3 and 1.8 for $p \geq 16$, and has the same fitting time for $p \leq 8$. As expected, the prediction trend instead does not change among the three variants.

The error-compensated versions of the algorithm are 4 to 5 times slower during the fitting phase than their corresponding non error-compensated version, and 2 times slower during the prediction phase. The quadruple-precision floating-point (`long double`) version of the algorithms instead are between 80 and 100 times slower



(a) Fitting time trend for the traditional Burg's method (Sec. 6.3.2.1) with `double` and no additional compilation flags

(b) Prediction time trend for the traditional Burg's method (Sec. 6.3.2.3) with `double` and no additional compilation flags

Fig. 6.5 AR models - computational time trend.

during the fit phase and about 40 times slower during the predict phase than their double-precision floating-point (`double`) equivalent.

Regarding compilation flags, `-ffast-math` helped reduce the computation cost by a factor between 1.2 and 1.4. Note that this holds only for the double-precision floating-point version of the algorithms, whereas for the quadruple-precision floating-point version it had no impact. Since on the Raspberry Pi 4B the quadruple-precision floating-point data type is library provided, and it is not directly backed by hardware, it is very expensive to operate on such a device, and additional compiler optimizations provided by `-ffast-math` may not be possible. Thus, even though using quadruple-precision floating-point parameters may help the AR models to reduce the prediction error, running them in real-time on the selected target device (i.e., the Raspberry Pi 4B) is not feasible. In fact, with $n = 512$ and $p = 16$, using the quadruple-precision floating-point data type we already reached the 2.90ms constraint, with only the Burg's denominator optimized algorithm (Alg. 3) being able to stay within our deadline, with a timing of 2.47ms for the fitting phase and 0.21ms for the prediction phase. The same algorithm, with the most demanding configuration ($n = 8192$ ans $p = 128$), required 305ms for the fitting phase and 1.65ms for the prediction phase. So, from now on, we will focus only on the double-precision floating-point (`double`) version of the algorithms.

Tab. 6.1 and 6.2 show the computational time required to fit the various models for different values of n, whereas p is set to 64 and 128 respectively. Tab. 6.3 shows instead the computational time required by the various models to predict the next n_{test}

Table 6.1 Time (ms) required to fit the specified algorithm on a training set of size n, with p set to 64 and data type set to `double`

	Fit time (ms)				
Algorithm	n=512	n=1024	n=2048	n=4096	n=8192
Burg's method	0.178	0.362	0.746	1.520	3.024
Denominator optimization	0.091	0.182	0.378	0.771	1.528
Hybrid denominator	0.102	0.205	0.424	0.866	1.720
Burg's method (`-ffast-math`)	0.114	0.228	0.477	1.007	1.985
Den. opt. (`-ffast-math`)	0.069	0.136	0.292	0.624	1.227
Hybrid den. (`-ffast-math`)	0.075	0.148	0.315	0.672	1.320
Burg's method (compensated)	0.795	1.628	3.327	6.706	13.385
Den. opt. (compensated)	0.473	0.962	1.967	3.945	7.878
Hybrid den. (compensated)	0.515	1.049	2.139	4.292	8.567

Table 6.2 Time (ms) required to fit the specified algorithm on a training set of size n, with p set to 128 and data type set to `double`

	Fit time (ms)				
Algorithm	n=512	n=1024	n=2048	n=4096	n=8192
Burg's method	0.335	0.703	1.466	3.009	5.996
Denominator optimization	0.172	0.352	0.743	1.519	3.018
Hybrid denominator	0.188	0.385	0.807	1.649	3.280
Burg's method (`-ffast-math`)	0.216	0.444	0.939	1.994	3.933
Den. opt. (`-ffast-math`)	0.132	0.267	0.573	1.230	2.423
Hybrid den. (`-ffast-math`)	0.141	0.283	0.606	1.296	2.552
Burg's method (compensated)	1.505	3.170	6.563	13.305	26.637
Den. opt. (compensated)	0.905	1.882	3.886	7.826	15.665
Hybrid den (compensated)	0.962	2.000	4.122	8.300	16.611

(128) samples, with p set to either 64 or 128. All results shown in the aforementioned tables have an absolute error $\leq \pm 3\%$.

6.4.10 Evaluation of the Prediction Error on Long Predictions

Fig. 6.6 shows, for every AR model algorithm and various values of n, the MAE over a long prediction of 20 packets of 128 samples each (2560 samples in total),

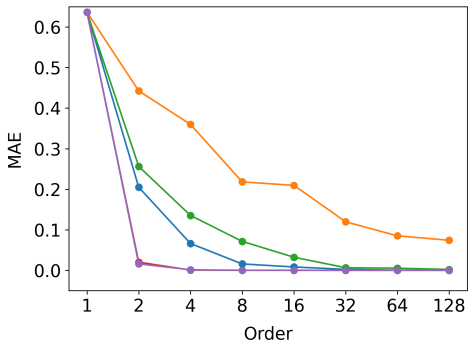
Table 6.3 Time (ms) required to predict the next n_{test} (128) samples, with p set to 64 and 128 and data type set to `double`

	Prediction time (ms)	
Algorithm	p=64	p=128
Burg's method Denominator optimization Hybrid denominator	0.028	0.058
Burg's method (`-ffast-math`) Den. opt. (`-ffast-math`) Hybrid den. (`-ffast-math`)	0.022	0.050
Burg's method (compensated) Den. opt. (compensated) Hybrid den. (compensated)	0.060	0.124

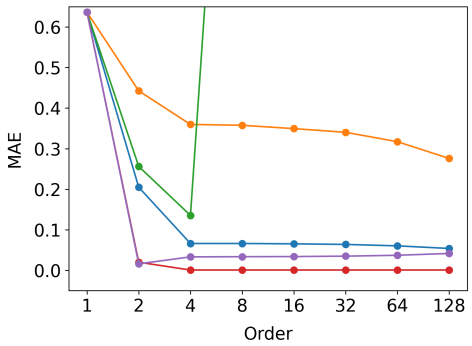
for each value of p. In the plots, where the MAE goes above the y-axis range, the model diverged with that configuration, drastically exceeding the [-1, 1] interval, which represents the standard range used in digital audio floating-point representations. Based on these plots, it emerges that, while error compensation may help on longer predictions, this effect is not deterministic. For example, in Fig. 6.6e, error compensation helped to avoid model divergence with $n = 2048$, which was instead present in Fig. 6.6b, but the error compensated model ended up diverging for $n = 1024$. Thus, in general, the benefits that error compensation may bring are negligible when compared to the additional computational cost. Moreover, depending on the implementation, the parameters, and the training data, numerical stability may not be granted.

Figs. 6.6b and 6.6e show another interesting effect. In the Burg's denominator optimized algorithm (Alg. 3), for every value of n, the model does not improve the prediction error beyond a certain value of p. This is a consequence of the errors made in estimating the initial k_i reflection coefficients, which lead to progressively smaller improvements with the increase in the model order. Interestingly, this does not happen for the Burg's hybrid denominator version (Figs. 6.6c and 6.6f), which follows more closely the trend of the traditional Burg's method (Figs. 6.6a and 6.6d).

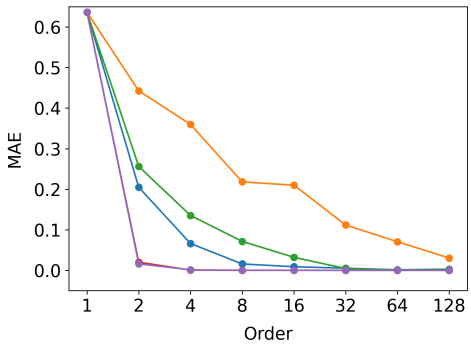
Thus, the proposed Burg's hybrid algorithm (Alg. 4) has the advantage of following more closely the error propagation trend of the traditional Burg's method, for



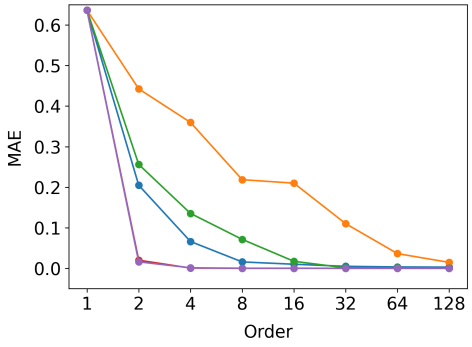
(a) Burg's method



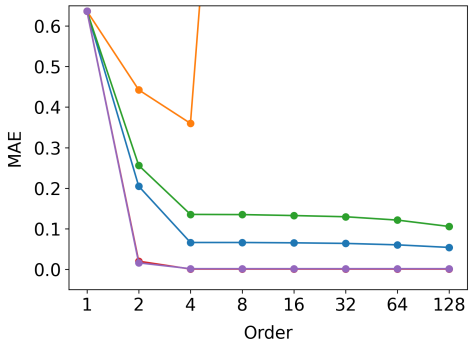
(b) Optimized den.



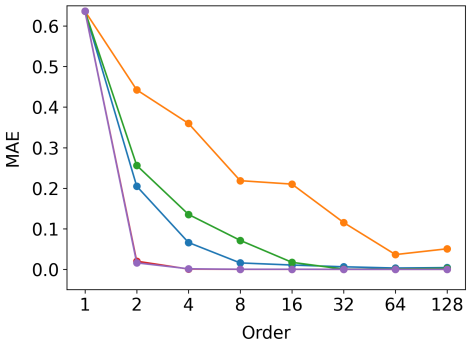
(c) Hybrid denominator



(d) Burg's method (compensated)



(e) Optimized den. (compensated)



(f) Hybrid den. (compensated)

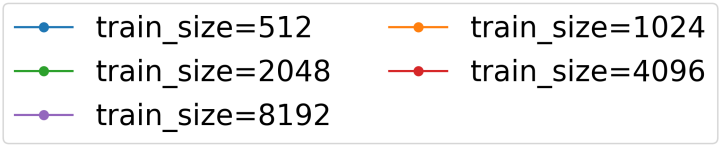


Fig. 6.6 AR models - long prediction MAE

longer predictions, while incurring in a computational cost only slightly higher than that of Burg's denominator optimized algorithm (Alg. 3), allowing for significant savings in comparison to the traditional method.

6.4.11 Qualitative Analysis

After identifying the combination of n and p that provided the best trade-off in terms of prediction performance versus required computational resources, we generated a series of samples for a qualitative analysis. For each category, we selected two audio files, for a total amount of 12 files. We manually trimmed all the audio files to be 5 s long, by selecting the portion of the file which was most relevant from the musical perspective (e.g., sections with silence were avoided). For each audio file, we then simulated the loss of 20 packets, by selecting 20 random positions. For each position (i) we removed the n_{test} samples in the interval $[i, i+n_{\text{test}})$. We then recovered the n_{test} samples by using the three PLC techniques: silence substitution, pattern replication and AR models. The three different audio files with recovered audio gaps were then stored on disk. Finally, we set up a quality test which involved human listeners: volunteers were recruited and asked to listen to the original excerpt and to the three generated audio files, and then to select the one which sounded more similar to the original. Alg. 10 reports the pseudocode for the preparation of the excerpts. For convenience, this step was implemented in Python with the `AutoReg` class, with $n = 2048$ and $p = 64$.

6.4.12 Qualitative Analysis Results

The quantitative analysis reported in the previous subsections does not consider that one of the most important requirements for a PLC technique in NMP is not to alter the perceived tempo. Indeed, when musicians play together, they derive an internal tempo reference from the perceived one and can continue playing following this internal tempo even if occasional silence periods are inserted in the audio playback they are listening to. If such silence periods are short enough, musicians can continue playing in sync, whereas longer silence periods may cause slight desynchronizations, that can be easily recovered once the audio play out is restored. On the other hand, replication of past audio samples may introduce alterations that, especially for percussive instruments, could be interpreted as double-beats, thus degrading the

Algorithm 10 Quality assessment

```
train_size ← 2048
lag ← 64
test_size ← 128

for category in categories do
    for file in audio_files do
        samples ← load_samples(file)
        b0_samples ← samples.copy()
        b1_samples ← samples.copy()
        ar_samples ← samples.copy()
        positions ← generate_20_random_indexes(samples)

        for pos in positions do
            train_set ← samples[pos − train_size, pos]
            test_set ← samples[pos, pos + test_size]
            start ← train_size
            end ← train_size + test_size − 1

            silence ← [0] ∗ test_size
            prev ← samples[pos − test_size, pos]

            b0_samples[pos, pos + test_size] ← silence
            b1_samples[pos, pos + test_size] ← prev

            AR_model ← AR(train_set, lag).fit()
            pred ← AR_model.predict(start, end)
            ar_samples[pos, pos + test_size] ← pred
        end for

        store_samples(b0_samples, b0_file)
        store_samples(b1_samples, b1_file)
        store_samples(ar_samples, ar_file)
    end for
end for
```

musicians' perception of the tempo curve, making their interplay more difficult. This is the main reason why the most widely adopted PLC solution in NMP is still silence substitution. Being based on the history of the time series, AR models have the property of likely not altering the musical tempo, while they should help to mitigate the impact of packet losses on the quality of the audio playback. To understand if

such quality improvement was perceivable, we administered the test described in Sec. 6.4.11 to a group of 25 volunteers.

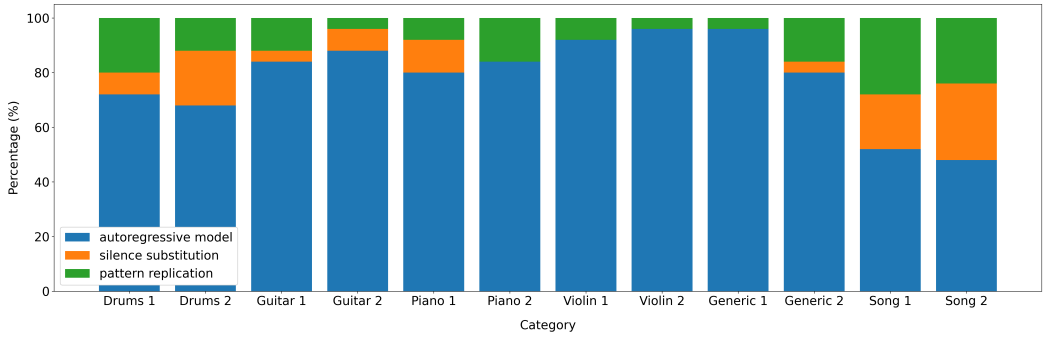


Fig. 6.7 Qualitative analysis results

Results of this test can be found in Fig. 6.7, which shows that, in most cases, the recovered audio obtained with the PLC solution based on AR models was perceived to be the most similar to the original audio. The main difference people heard, from comments we received after the test, was in terms of "noise". Testers selected the track which was perceived as "less noisy". The noise they refer to is actually caused by signal discontinuities, resulting in audible audio glitches.

It is worth noticing that, whenever we are replacing missing audio samples with new ones, we are likely to create two discontinuities in the audio signal, at the two extremes of the audio gap. The perceived intensity of the glitches depends on the distance between the two consecutive samples that cause the discontinuities. Since AR models predict values based on the history of the time series, they ensure a continuity on the left side of the predicted section, thus generating (on average) only half the number of glitches w.r.t. the other two solutions. This is the main reason why, in most of the cases, the PLC method based on AR was preferred.

From this analysis, it is again confirmed that in general AR models work better with certain categories of instruments: in particular, the more the signal contains sustained sounds, the better the AR models work as a PLC technique, while the more the signal contains transient sounds with abrupt variations, the less noticeable is the benefit of AR models over the two benchmarks. This difference in behavior also holds for the pattern replication PLC technique, which provides a better experience w.r.t. silence substitution for more sustained-oriented sounds, while it provides worse results for more transient-oriented sounds, as it may create the double-beats effect or

may even create a flanging effect if the lost audio section extends for more than one audio packet.

Results of the subjective test showed that for all the categories that include single instrumental sources, the AR models solution was preferred, while for the category "Songs" that includes polyphonic music, no clear preference emerges. Indeed, polyphonic music masks better audio glitches, thus making the differences between the three solutions less noticeable. The interested reader can find the reconstructed audio files using the three PLC techniques (silence substitution, pattern replication and AR models with the Hybrid denominator implementation, n=2048 and p=[64, 128]) in the publicly available GitHub repository[8]

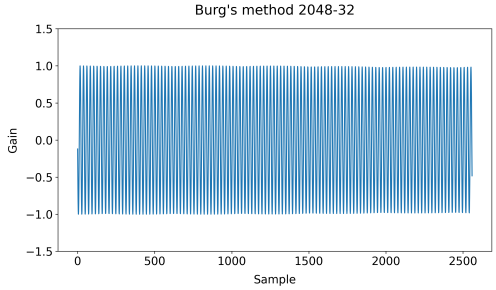
6.5 Limitations

While AR models using Burg's method proved to outperform the two benchmark solutions, they have some limitations. The first one is in terms of computational requirements: though AR models are reasonably light in terms of memory, they are more demanding in terms of CPU, especially during the fitting phase (the prediction phase is about one to two orders of magnitude lighter). Through algorithm optimizations, like the denominator optimization illustrated in Sec. 6.3.2.2, and the usage of the appropriate compilation flags provided by the specific language compiler in use, CPU usage can be tamed. However, even if optimized, they are still non-negligible, which means that the scalability of this solution is not as good as one of the two benchmark PLC solutions. Thus, if used in the context of an NMP application, some trade-offs emerge. In a real deployment, a possible approach could be to set an upper bound on the maximum number of AR models to be executed in parallel, associated with the network channels that are more likely to experience packet losses (this choice could be done based on packet loss statistics collected over the history and updated in real-time).

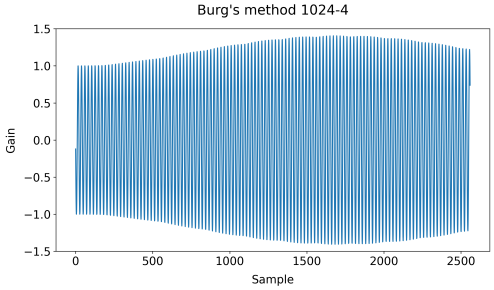
The second limitation, which is specific to Burg's method implementation, is that every time a new packet is received, the whole model needs to be fitted again on the new training set. This can be mitigated, e.g., by fitting the model upon detection of a lost packet and then using it to predict the next M packets (without re-fitting the

[8] https://github.com/matteosacchetto/burg-implementation-experiments

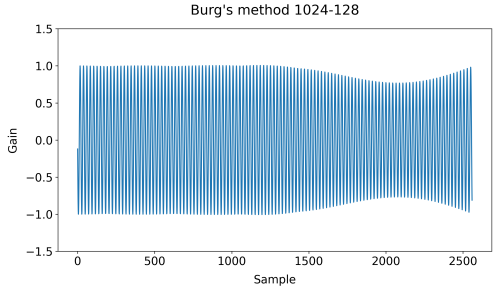
model on a per-packet basis), with M carefully chosen so that the signal stationarity is likely to be maintained within such time window. The investigation of solutions that allow for an incremental incorporation of new audio samples without requiring to fit again the model from scratch on the whole training set is left for future work.



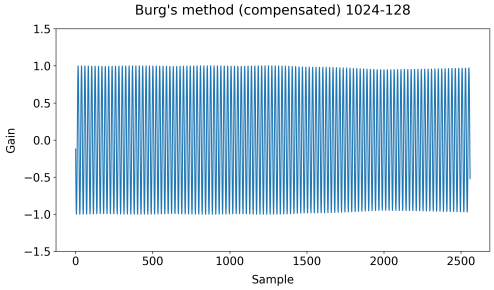
(a) Burg's method with n=2048 and p=32



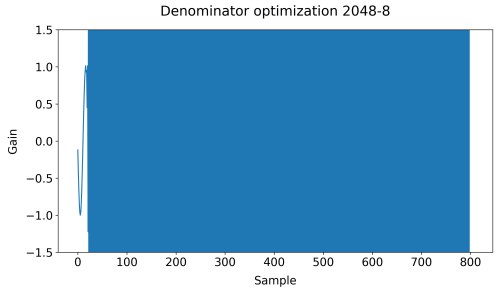
(b) Burg's method with n=1024 and p=4



(c) Burg's method with n=1024 and p=128



(d) Burg's method (compensated) with n=1024 and p=128



(e) Optimized denominator with n=2048 and p=8

Fig. 6.8 Resulting waveform for AR-based predictions of 2560 samples (20 packets of 128 samples) of a 2 kHz sine wave

The third limitation of AR models is that numerical stability may not be granted, as discussed in Sec. 6.4.10, as it depends on the implementation, the parameters, the training data and the number of samples to predict. To show this effect, we fit different implementations and configurations on a sine wave with a fixed frequency

of 2 kHz. We then used the various models to predict the next 2560 samples (20 packets of 128 samples). In Fig. 6.8 we report some examples of how the different implementations and configurations behaved. Sometimes, error-compensation may help, as reported in Figs. 6.8c and 6.8d. These two plots were obtained with the same algorithm and parameters, with the only difference that in Fig. 6.8d error-compensated floating-point transformations were adopted. Nevertheless, as shown in Figs. 6.8a, 6.8b and 6.8e, numerical divergence heavily depends on the AR model algorithm and parameters used.

In Fig. 6.8a we obtained an almost perfect reconstruction of the input signal, while Fig. 6.8b reports a sine wave which exceeds the range [-1, 1], which, once played back, would result in auditory distortions due to audio converters clipping. Fig. 6.8e, instead, shows that after 20 samples amplitude values ultimately explode to $[-\infty, +\infty]$, and after 800 samples the algorithm aborted. Countermeasures to avoid the numerical stability problems need to be applied. A possible approach could be to pre-check the generated samples and, if they exceed the range [-1, 1], but are still limited (like in Fig. 6.8b) an amplitude rescaling could be applied. Conversely, if they diverge to $[-\infty, +\infty]$, a fallback solution could be to use silence substitution, to avoid generating loud distortions which could possibly damage the equipment or the ears of the listeners. Alternative approaches could include introducing a regularization term or forcing the prediction to fade to silence when the number of contiguous samples to predict exceeds a specified threshold.

The last consideration about AR models is that, while they have the benefit of creating a prediction which ensures continuity with past samples, they have no guarantees of continuity with future samples. Thus, to avoid creating discontinuities between the predicted sections and the next actual section, which would result in audible glitches, some strategies need to be implemented. A well-known strategy to smooth out the discontinuities is to predict a number of samples greater than the size of the missing section, and then use a simple cross-fade on the overlapping samples to fade-out the predicted samples and fade-in the actual samples of the next section [172–174].

6.6 Summary

This chapter proposes a PLC solution based on AR models and targeted towards NMP applications. In particular, the focus is on exploring different implementations of AR models based on the Burg's method for AR model fitting and proposing a novel variant that mitigates error propagation during the recursive execution of the prediction algorithm, while preserving computational scalability. Since many NMP solutions are also built to run on a Raspberry Pi, the focus of this chapter is to propose a PLC solution which can run on the Raspberry Pi 4B. Experiments were performed considering various categories of audio signals, exploring various choices of parameters and different implementations.

Results showed that, even with small orders, AR models outperform the prediction performance of the two benchmark PLC techniques in terms of MAE and RMSE. These results were also confirmed by the qualitative analysis which showed that, especially for solo instruments, AR models are perceived to recreate an audio track more similar to the original one in comparison to the two benchmarks. Concerning computational timings, results indicated that a training set of 2048 samples is the highest value which guarantees the completion of the model fitting and prediction of the next 128 samples within 2.90 ms, for model orders up to 128. Moreover, using either the denominator optimization or the hybrid denominator versions of the Burg's method reduces the computation cost by a factor of 1.3 to 2, thus increasing the scalability of the proposed solution. Finally, we want to acknowledge that, while this PLC method and its implementation were optimized and tested with NMP as the target application, it could be employed in any other scenario where low-latency streaming of music/audio material is involved.

Chapter 7

Introducing DUST: a Dataset of Real-Time UDP Sound Packet Traces

This chapter introduces DUST, a dataset composed of UDP audio packet traces generated by a real-time NMP system. The dataset includes various files collected under different network and device configurations. Each file combines traces of incoming audio packets with local playback status information. The aim of this dataset is to support offline simulations of real-time audio streaming systems, to facilitate the development of algorithms to address challenges such as packet loss concealment, network jitter resilience, and clock drift compensation.

The work presented in this chapter was part of a contribution to the 5th International Symposium on the Internet of Sounds [54], held in September 2024.

7.1 Introduction

Real-time audio streaming over the internet is employed by a wide range of applications, such as videoconferencing and NMP tools. While deferred audio streaming, as adopted in music on-demand services, primarily focuses on preserving the quality of the received data, real-time audio streaming imposes the additional constraint of maintaining a low latency. This constraint arises from the necessity to ensure a seamless and responsive user experience (see Sec. 1.1). Therefore, designing a real-time

audio streaming system poses additional challenges compared to implementing a deferred streaming system.

The basic setup of a real-time audio streaming application involves an audio source device at one location and an audio destination device at another (remote) location, interconnected via a telecommunication network, such as the Internet. The best-effort design of IP does not permit to concurrently achieve timely and error-free data delivery, thus introducing a trade-off between these two objectives. More in detail, factors such as network configuration, link layer connection type (e.g., Ethernet, Wi-Fi), network congestion, and routing changes influence packet delivery timing, creating observable *network jitter*. Network jitter, can significantly impact the performance of real-time applications (see Sec. 2.1.1). To mitigate such impact, the receiver end of a real-time audio streaming system usually features a de-jitter buffer to smooth out the jitter effect.

Moreover, real-time audio streaming applications typically rely on UDP as a transport layer protocol. Therefore, no guarantees are provided regarding the actual packet delivery, leading to potential late or lost packets. Such *packet losses* deteriorate the quality of the audio playback by introducing audible artifacts. Therefore, PLC techniques aimed at mitigating the perceptual impact of missing data portions in a multimedia stream, due to either delayed or lost packets, need to be adopted, as discussed in Chap. 6.

Finally, since communication occurs between two distinct machines, two separate audio interfaces are involved for audio capture and playback. The clocks of the two audio interfaces are never identical. Thus, it becomes necessary to tackle the problem of *clock drift* (see Sec. 1.1).

Even though these three factors can be modelled independently, in a real scenario they *influence each other*. Indeed, a high jitter has the potential to cause local buffer underruns. Additionally, drift must be corrected to prevent unrecoverable overruns and underruns. Finally, packet loss can only be handled if promptly detected, i.e., if the buffer is large enough to permit the application of any recovery strategy in due time.

To the best of our knowledge, currently publicly available datasets consider multimedia streaming over HTTP of prerecorded material (e.g., [175]) or videoconferencing applications (e.g., [176]), instead of NMP frameworks, and predominantly focus on the analysis of one of the three above-mentioned aspects (e.g., [101]),

but do not permit to evaluate the intrinsic interconnection among them. Moreover, packet traces are typically constructed under the hypothesis that every packet can be either lost or correctly received, under the common assumption that a late packet is treated as a lost packet. However, this assumption does not necessarily hold true in a real-time audio streaming scenario. In fact, a packet that arrives too late w.r.t. the due playout time of the first audio sample in its payload may still contain a portion of samples whose playout time has not yet expired. In such scenarios, the packet content should not be entirely discarded, as part of the carried audio signal could still be reproduced within the required timeframe.

To address such shortcomings, this chapter presents DUST: a dataset of network packet traces transmitted by an NMP application (i.e., MEVO [57]). MEVO leverages UDP as a transport layer protocol, on top of which it implements a simple custom application layer protocol to store all details related to the included audio samples, such as the capture buffer index of the first sample in the packet, the audio format, the number of samples in the packet, etc. This information allows, for example, to quantify the exact samples lost in the case of actual packet loss, even in scenarios where the number of samples per packet changes throughout the communication. Moreover, in the case of late packets, it also allows computing the number of samples that are still within their intended timeframe, which depends on the buffering strategy employed.

The traces involve the communication between two Raspberry Pi 4B devices running MEVO, and were collected under varying network conditions and using different audio interfaces. Data collection and storage occurred on each device at the receiver side of the communication. DUST has been created with the purpose of enabling simulating the behavior of a real-time audio streaming application in realistic network conditions. The dataset is designed to integrate information about the playout process with audio packet arrivals captured in successive snapshots. This allows the simultaneous simulation of the reception or loss of a number of samples smaller than the packet size and an accurate quantification of the clock drift. In turn, such features facilitate a more thorough evaluation of the performance of state-of-the-art PLC and buffering techniques.

The remainder of this chapter is organized as follows: Sec. 7.2 provides details about the dataset, the data collection method, and a description of its structure and

content. Sec. 7.3 provides some details on how to process the dataset to quantify drift and sample losses. Finally, Sec. 7.4 concludes this chapter.

7.2 The Dataset

7.2.1 Data Collection Method

Data has been collected leveraging a modified version of the MEVO system [57]. The system features a Raspberry Pi 4B as computing device and an off-the-shelf professional audio interface. The Raspberry Pi is connected to the Internet using a wired Ethernet connection. The Raspberry Pi runs the Raspberry Pi OS with the Linux PREEMPT_RT kernel. The MEVO software running on it controls the audio interface through the in-kernel ALSA driver and sets the hardware parameters of both the capture and playback interface to the lowest possible hardware buffer size, and the sampling rate to 44.1 kHz. Since sender and receiver features can be independently set, in the following we will describe the behavior of the two processes separately, from a software perspective.

Sender process As soon as a chunk of audio is ready to be captured, it is read and stored in a small memory buffer by the capture thread (`TC`). According to a given policy, after a sufficient number of samples (usually between 30 and 132) have been collected, a second thread, the sender thread (`TS`), stores them in a UDP packet, using a simple custom application layer protocol, and sends it to the remote receiver process (see Fig. 7.1).

Receiver process Upon reception of a packet from the network, the network thread (`TN`) extracts audio samples from it and enqueues them in the playout buffer. The playout thread (`TP`) constantly reads audio samples from the playout buffer and outputs them through the ALSA driver. For the sake of the data collection, the behavior of `TN` has been modified to collect a snapshot of the playout buffer state and of the incoming packet when enqueueing it. The snapshot collection process has been designed to be computationally lightweight, to minimize its impact on the measured quantities. `TN` writes the snapshot to memory, and the logger thread (`TL`) is responsible for batch writing the snapshots to mass storage as lines of a CSV file

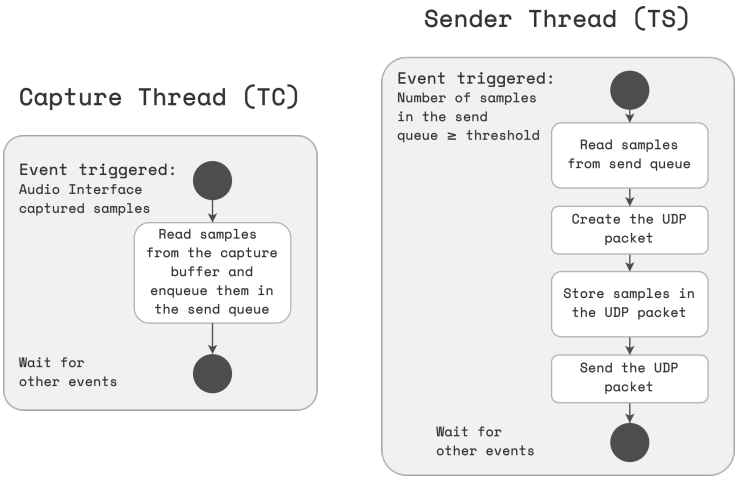


Fig. 7.1 Sender process diagram

(see Fig. 7.2). During the data collection process, the audio has been monitored at regular intervals to guarantee it was correctly audible on both sides.

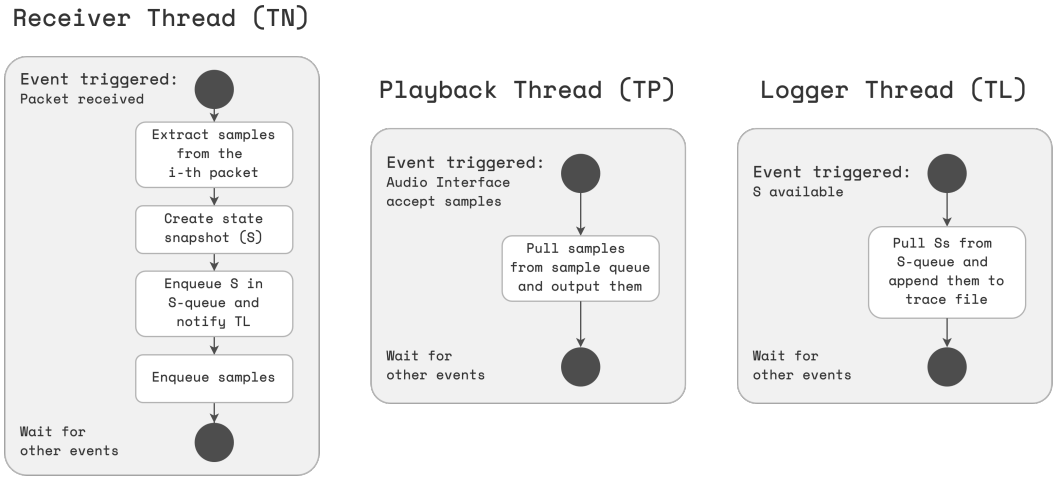


Fig. 7.2 Receiver process diagram

7.2.2 Dataset Structure and Content

The dataset is a collection of CSV files containing one record for each received packet. Each line consists of three columns:

- `read_index`: this index corresponds to the number of samples read from the playback audio buffer (i.e., consumed) by `TP`.

- `send_index`: this index corresponds to the position, in the playback buffer, where to write the first sample of the packet being enqueued by `TN`. This information is directly obtained from the received packet's header.
- `tstamp_ns`: a local timestamp, measured in nanoseconds, according to the Raspberry Pi real-time-clock, relative to the first reception event by `TN`.

All initial values are set to 0, and no data cleaning has been performed. For each experiment, three files are generated: a pair of CSV files (one for each device), adhering to the aforementioned format, produced by the two communicating devices, where each device acts both as sender and receiver, plus a third file, in JSON format, that offers a high-level description of the experiment, providing the following details:

- `card`: the models of the audio interfaces used.
- `location`: the physical locations.
- `duration`: the duration of the experiment, expressed in hours (ranging from 1 to 9).
- `conn_type`: the type of the *last mile* Internet connection (e.g., Fiber To The Home (FTTH)).
- `comments`: additional comments related to the experiment.

At the time of writing, three different audio interfaces were used:

- Behringer UMC404HD.
- Scarlett 2i2 (1st generation).
- Audient ID14 MKII.

Experiments have been performed through non-exhaustive combinations of network conditions. We combined a Fiber To The Home (FTTH), Fiber To The Cabinet (FTTC) and an enterprise network. Additionally, we collected data on a direct Gigabit-Ethernet connection between the two Raspberry Pis, using the same model of audio interface on both sides to provide a baseline of the expected behavior. Some

traces present behaviors which are strongly influenced by a third agent (e.g., another device concurrently sharing the same network access link).

For reference, below are reported the first few lines of a CSV file from the dataset (loc_11A.csv) along with its associated JSON file.

Listing 7.1 Initial content of the CSV file loc_11A.csv

```
read_index,send_index,tstamp_ns
0.0,0.0,0.0
60.0,66.0,1459422.0
126.0,132.0,2960308.0
192.0,198.0,4450305.0
258.0,264.0,5948580.0
324.0,330.0,7453891.0
390.0,396.0,8949073.0
456.0,462.0,10448792.0
...
```

Listing 7.2 Content of the JSON file loc_11.json

```
{
    "files": {
        "loc_11A": {
            "card": "umc404hd",
            "conn_type": "eth",
            "location": "none"
        },
        "loc_11B": {
            "card": "umc404hd",
            "conn_type": "eth",
            "location": "none"
        }
    },
    "duration": 1.0,
    "date": "20/03/2024",
    "comments": "direct connection"
}
```

The dataset is publicly available on Zenodo[1]. For efficient storage and distribution, each CSV file was compressed using Gzip compression via the `gzip`[2]

[1] https://doi.org/10.5281/zenodo.12726670
[2] https://www.gnu.org/software/gzip/

command-line utility, resulting in files with the `.csv.gz` extension. To access the raw CSV data, the `.csv.gz` files must first be decompressed using either `gunzip` (or `gzip -d`) on Linux and macOS, or 7-Zip[3] on Windows. At the time of writing, we experienced an issue with Firefox, where downloading from Zenodo an individual `.csv.gz` file led to a double Gzip compression of it. Thus, to avoid this issue, it is recommended to either download the entire dataset as a ZIP archive, or retrieve individual files using a Chromium-based web browser or a command-line tool, e.g., `curl`[4].

7.3 Information Extraction and Utilization

7.3.1 Clock Drift and Jitter Characterization

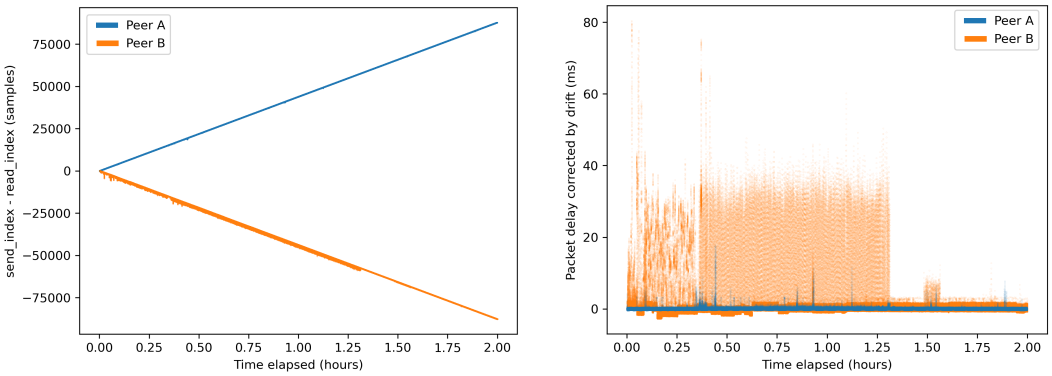
Each file enables the extraction of the drift between the devices' clocks and the characterization of jitter. The drift d is generally expressed in parts-per-million (ppm). For audio devices, drift can be measured using the number of samples as a time indicator. Given a file of length N with remote device D_R and local device D_L, a possible simple definition of the average clock drift of D_R w.r.t. the clock of D_L can be given by the following formula:

$$d = 10^6 \left(\frac{S_{N-1}}{R_{N-1}} - 1 \right) \tag{7.1}$$

with S and R being respectively the `send_index` and the `read_index`, and their subscripts indicating the packet number (starting from 0). Assuming a certain behavior of d (e.g., supposing that it remains constant during the experiment), it is possible to define a function f such that $\overline{S_i} = f(R_i)$ with $\overline{S_i}$ being the estimated value of S_i considering the drift. If d and f are representative of the behavior of the two devices, then it is possible to use the value $\varepsilon_i = S_i - \overline{S_i}$ to characterize the jitter observed by the receiver. An example of clock drift and jitter characterization is presented in Fig. 7.3. Specifically, Fig. 7.3a shows, for each peer, the difference between the number of received samples (`send_index`) and the number of consumed samples (`read_index`) for the `sanri_12` capture. A positive value indicates that

[3] https://www.7-zip.org/
[4] https://curl.se/

the device is consuming samples more slowly than it is receiving them, i.e., its clock is slightly slower than the sender's, while a negative value indicates faster consumption. The closer the difference is to zero, the smaller the drift between the two clocks. Instead, Fig. 7.3b depicts the difference between the actual and expected packet arrival times, after compensating for clock drift. In this example, the drift was estimated using a simple linear regression.



(a) Clock drift between the two peers

(b) Delay between expected packet reception time and actual reception time corrected by the clock drift

Fig. 7.3 Clock drift and packet delay variation in the `sanri_12` capture

7.3.2 Sample Loss Simulation

Identifying lost samples in the audio playout is pivotal for the assessment of PLC methods. When referring to audio streams, sample losses can be due to three possible causes:

- *Hardware/Software (machine) faults*: for instance caused by a buffer overrun/underrun due to processor overload (at the sender side).
- *Packet drops*: introduced by intermediate routers in the telecommunication infrastructure, when using a non-reliable transport protocol, i.e., UDP.
- *Latency spikes*: one or more consecutive packets, i.e., a chunk of audio samples, arrives at the destination significantly later than their expected timeframe.

Publicly available datasets used for packet loss simulation typically identify packet loss by assigning a boolean value to each packet, to indicate whether it

was correctly received. Differently, the information collected in DUST allows for the simulation of more refined scenarios, e.g., where a portion of the data carried by a late packet can still be reproduced. For instance, suppose a packet carrying 128 samples being 1 ms late w.r.t. its expected playout time. In this scenario, assuming that the sampling rate is 44.1 kHz, the application could still reproduce $128 - \lceil 1 \cdot 44.1 \rceil = 128 - 45 = 83$ samples contained in that packet. This may have a non-negligible impact on the performance of a PLC technique, since in one case the algorithm must regenerate the whole packet, whereas in the other case it needs to only reconstruct one-third of the packet content.

To use DUST for sample loss simulation, a dedicated simulator must be created. The simulator would produce a bit-stream (a stream of boolean values) associated to a given file of DUST. Each value of the bit-stream would mark the correct/wrong reception of the corresponding sample. Fig. 7.4 provides an illustrative example of how the simulator could produce the bit-stream. We highlight that the simulation output is not unique, as it is strongly dependent on the buffering and drift-correction policies implemented in the simulator. We can safely assume that there exists a controller that applies a correction to the packet's *send index* S_i, incorporating the buffering policy and the drift correction.

A simple approach to implement such controller may assume a fixed user-defined target value for the buffer size (in number of samples) and the presence of an oracle that provides a bias B to be applied to S_i based on the elapsed time and the prior knowledge of the clock drift d between the two audio interfaces. In Eq. 7.2, we show how to obtain a value W_i, defined as the *write index* of the incoming i-th packet, from S_i with the parameters B and d (expressed in ppm).

$$W_i = B + \frac{S_i}{(1 + \frac{d}{10^6})} \tag{7.2}$$

W_i is directly comparable to R_i such that

$$L_i = \begin{cases} 0, & \text{if } W_i \geq R_i \\ \min(R_i - W_i, C), & \text{otherwise} \end{cases} \tag{7.3}$$

with C being the number of samples per packet and L_i the number of initial lost samples of i-th packet. Fig. 7.5 shows an example of the behavior of $W_i - R_i$ assuming $B = 128$. We also assume that there are no out-of-order packets. Otherwise, the file

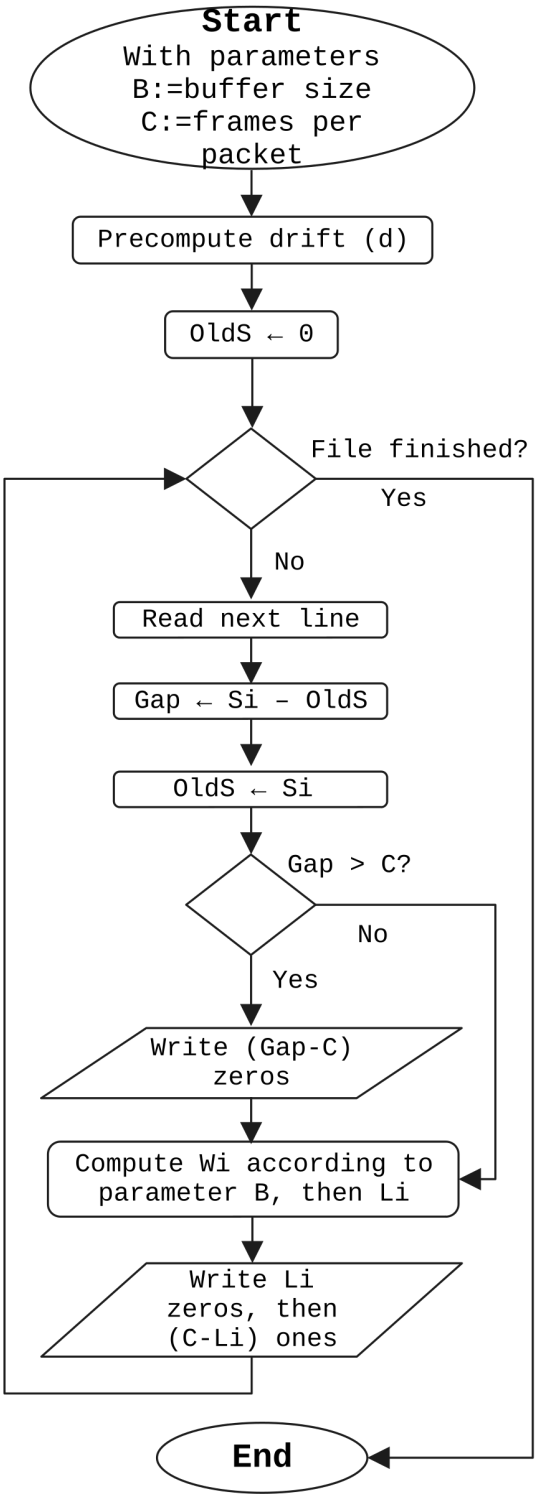


Fig. 7.4 Example flow-chart for a sample loss simulator, given a file of DUST and parameters B (buffer size) and C (samples per packet). An output value of zero marks a lost sample.

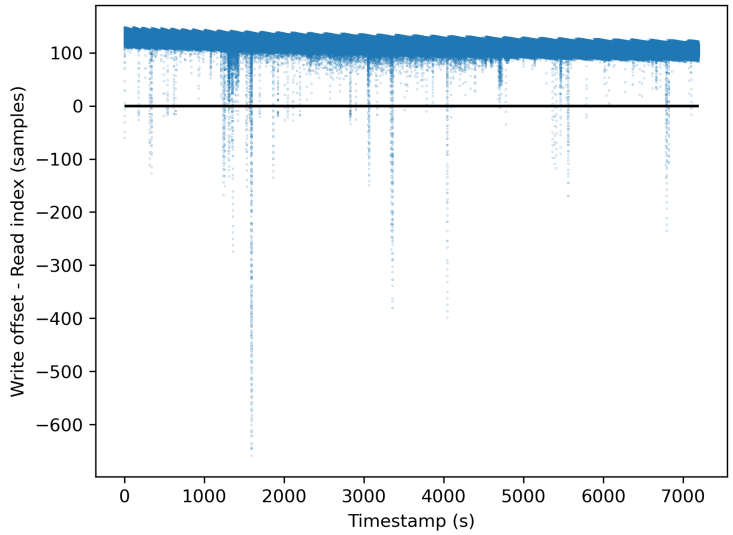


Fig. 7.5 Example of the evolution of $W_i - R_i$ over time, assuming $B = 128$. Negative values indicate the presence of lost samples.

must be sorted in ascending order by the `send_index` column before being fed into the simulator.

It should be noted that the algorithms described in Sec. 7.3 serve as illustrative examples of how to extract meaningful information from this dataset. However, both represent simplified implementations that do not fully capture the complexity of real-world scenarios.

7.4 Summary

This chapter presented DUST, a dataset containing UDP audio traces collected from the communication between two devices running an NMP system over an IP network. The dataset encompasses different scenarios, considering various network configurations and audio devices. DUST aims to be a support tool for research on packet loss concealment, drift estimation and jitter characterization. Even though the dataset has been created with NMP as reference application, we acknowledge its potential for broader utilization, even under less restrictive latency requirements (as, e.g., in the case of videoconferencing systems). In the future, we aim to further expand DUST by incorporating new traces collected in additional network scenarios, such as Wi-Fi and 5G networks.

Chapter 8

Remote Orchestral Conduction via a Virtual Reality System

This chapter introduces a Virtual Reality (VR)-based approach designed to provide remote musicians with real-time visual feedback of the conductor without relying on traditional video acquisition and transmission. In the context of NMP, where latency and bandwidth constraints pose significant challenges, the proposed system leverages a VR headset to track the conductor's gestures, including hand movements and head orientation. These motion data are then transmitted to remote performers and visualized through an avatar representation of the conductor. Quantitative results suggest that the Motion-to-Photon (M2P) latency introduced by the system is tolerable for NMP applications, while the proposed system achieves a substantial reduction of bit rate requirements in comparison to traditional video streaming.

The content of this chapter is part of a contribution presented in the 5th International Symposium on the Internet of Sounds [55] in September 2024.

8.1 Introduction

In NMP, several research efforts have focused on conveying high-fidelity audio while minimizing the M2E latency. However, an in presence musical interaction is not solely based on audio cues, but also relies heavily on visual cues, which musicians use to synchronize their performances and convey expressive intentions [177].

The most common form of visual feedback that an NMP system could exploit is a video stream, as typically done in videoconferencing platforms. Unfortunately, including video streaming in an NMP system increases the network bandwidth required, thus introducing scalability limitations. To reduce bandwidth requirements, the video needs to be encoded before streaming, which introduces additional latency. Since NMP systems have strict M2E latency requirements to ensure a synchronous musical interplay (see Sec. 1.1), audio and video streams are generally transmitted and reproduced out-of-sync. To address this synchronization issue, some attempts have been made to eliminate encoding/decoding latency by streaming uncompressed video [13, 14]. However, this approach resulted in a resulted in a minimum network bandwidth requirement of 1 Gb/s (and even higher if the HD video configuration is used[1]).

In this study, we explore an alternative to video transmission in the context of a remote orchestra conducting scenario: we exploit a VR headset to extract and transmit the conductor's gestures to a geographically distributed orchestra interconnected via an NMP system. Our proposed approach has the twofold goal of reducing *i*) the required transmission bitrate, since we need to transmit only data related to the pose of hands and head of the conductor, and *ii*) the M2P latency, i.e., the time taken for a user gesture to result in a corresponding visual change on a display [178], as pose tracking can operate at a higher frame rate than what is commonly supported by cameras.

Numerical results show that the average M2P latency achieved by our experimental setup is on average 140 ms and never exceeds 210 ms, which constitutes a promising starting point for future potential integration in a Musical Metaverse (MM) framework [179]. Indeed, the MM concept foresees the creation of a digital universe where users can interact musically via avatars, blending the physical and digital worlds through Musical Extended Reality (Musical XR) and IoMusT technologies [29]. In particular, we envision the adoption of our proposed system as a building block for MM-mediated orchestra rehearsals and performances, as already occurred in [180].

The remainder of this chapter is organized as follows: Sec. 8.2 provides an overview of the system with technical details on its implementation, Sec. 8.3 describes the methodology used to evaluate the performance of the system, Sec. 8.4

[1] https://lola.conts.it/downloads/LolaManual2.0.0rev001.pdf

presents the obtained results, Sec. 8.5 outlines the identified limitations. Finally, Sec. 8.6 concludes this chapter.

8.2 System Overview

In this section, we present our proposed framework for transmitting visual cues for remote orchestra conduction and explore the potential approaches for connecting a VR headset worn by the conductor to a web application used by the musicians in the NMP session.

The application running on the VR headset captures head and hand movements of the wearer and displays them in an immersive scenario. This scenario is shown from the conductor's perspective, with the musicians in front of them. Musicians are represented as avatars, with their name reported on top of them, and are positioned in the virtual space in a way that facilitates the conductor to identify and point to each musician individually. An example of the conductor's perspective is shown in Fig. 8.1.

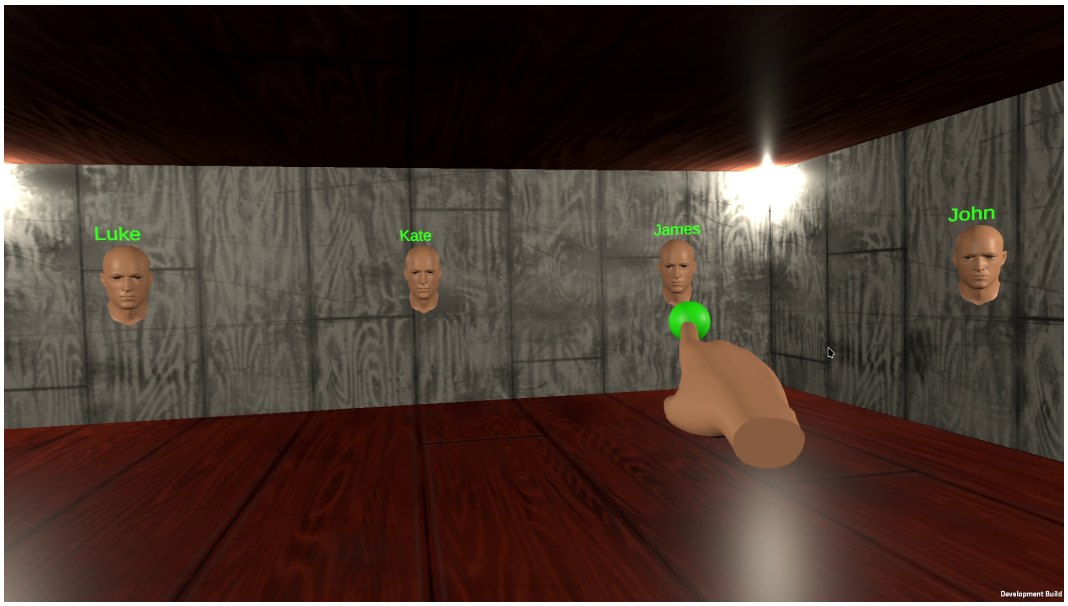


Fig. 8.1 The conductor's avatar pointing at *James*, seen from the conductor's perspective.

On the musician's side, a receiver device obtains the pose data from the conductor and renders it on a screen. It displays a virtual scenario depicting the avatars of the

conductor and of the other musicians in the same field of view. An example of the conductor's avatar seen from the perspective of four different musicians is shown in Fig. 8.2. The positioning of each individual musician in the scene visualized by the conductor through the VR headset is coherent with the positioning of the conductor in the personalized visualization offered to each musician.

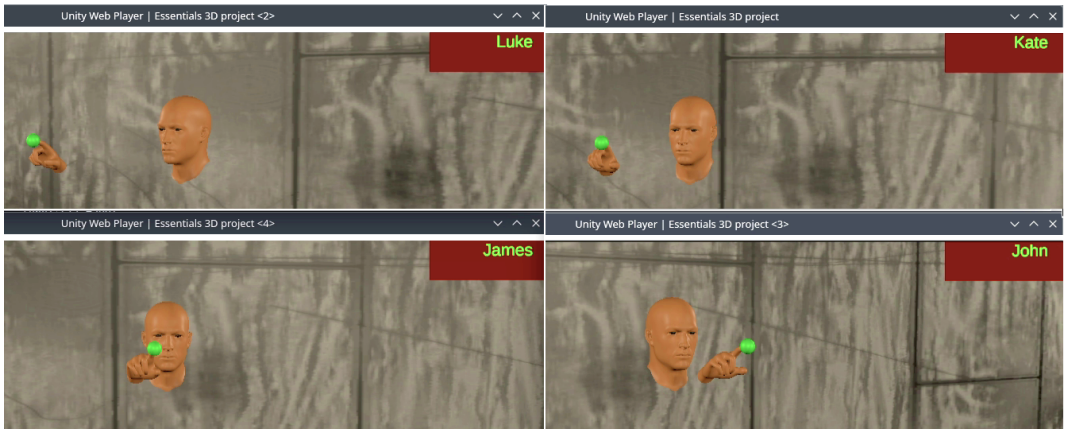


Fig. 8.2 The same scenario of Fig. 8.1 seen from the perspective of the four musicians

8.2.1 Transmission Setup

The main challenge of this system is the streaming of real-time pose data from the VR headset to the remote musicians' devices. First, even though any wired or wireless transmission medium could be leveraged to convey pose data to the musicians' devices (equipped with a web browser to visualize the conductor's gestures), it is highly recommended to rely on a wired Internet connection, since latency and jitter figures achievable by Wi-Fi and 4G technologies are hardly compatible with NMP requirements [181]. Second, web applications cannot directly handle TCP connections or UDP packets. However, they can leverage WebRTC [75], that supports the exchange of messages in a P2P fashion, as well as the use of unreliable (UDP-like) data channels [137], which is the preferred choice when dealing with real-time data. Thus, our application uses WebRTC for the conductor's pose data streaming.

The application running on the VR headset is not strictly bound to any specific transport protocol. However, the VR headset has limited options when it comes to physical connections. It supports *i*) Wi-Fi, through an integrated Wi-Fi adapter; *ii*) a

Wired Ethernet adapter over USB-C; *iii*) Android Debug Bridge (ADB)[2] on top of USB.

During testing, we discovered that Ethernet support varies among VR headsets. In that phase, we discovered that the Meta Quest 2 did not support it, while the Meta Quest 3 did. However, the graphical interface of the Meta Quest 3 did not provide consistent information about the connectivity status when using Ethernet, indicating that its integration is not yet stable. For such reasons, we opted for not leveraging an Ethernet connection, and relied instead on the ADB option since, as shown later in Sec. 8.4, it yields comparable latency results while being more stable software-wise. The ADB option has the main drawback of requiring an additional device to enable the connectivity from the VR headset to the Internet. However, since in a realistic scenario the conductor needs to receive the audio streams produced by the remote musicians, we reckon on the presence of a device, which can also act as a streaming device for pose data, that may additionally run an NMP system for audio transmission (e.g., one of the systems listed in Sec. 2.2.1). Note that our system is agnostic to the number of audio channels streamed by the NMP system, i.e., both stereo and immersive (3D) audio streaming could be used.

In summary, in our setup, the VR headset is connected to a streaming device via USB. The application on the VR headset communicates with other peers and the server through a TCP socket forwarded by ADB reverse forwarding. The streaming device, i.e., a generic computer running the ADB tool, handles the connections. Messages in the TCP socket are synchronously produced and consumed by the application immediately after each graphic frame generation.

8.2.2 Framework Description

Our VR-based system prototype for remote orchestra conduction consists of four main components, as shown in Fig. 8.3:

- **VR headset**: worn by the orchestra conductor to track their hands and head movements and to visualize the remote performers via avatars. In our experiments, the VR headset used was the Meta Quest 3.

[2] https://developer.android.com/tools/adb

- **Streaming device**: a PC connected to the VR headset via USB and to the Internet through a wired connection. It takes care of receiving pose data from the VR headset through ADB and of streaming it to the remote musicians through WebRTC.
- **Signaling server**: a server that facilitates the setup of WebRTC P2P connections.
- **Receiver(s) device**: a PC connected to the Internet with a wired connection. It takes care of receiving and displaying the conductor's movements to the musicians through a web application.

Notably, the adoption of WebRTC is in line with recently proposed implementations of multi-user applications for the MM, e.g, in [182–184].

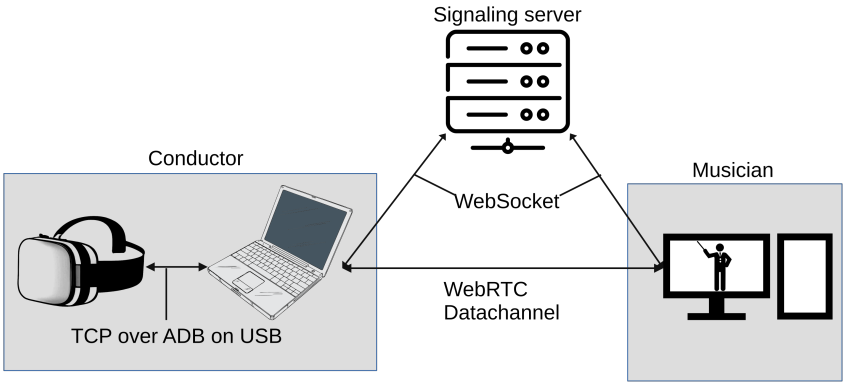


Fig. 8.3 System overview

The application running on the VR headset is built with the Unity game engine[3] and the aid of the Oculus SDK[4]. The web application running on each receiver's device is also built with Unity and compiled as a WebGL target. It reuses the same scenario and layout of the application running on the VR headset, making the interaction and the positioning coherent among the two entities. The streaming device application and signaling server are Node.js applications.

[3] https://unity.com/
[4] https://developer.oculus.com/downloads/unity/

8.3 Evaluation Methodology

The M2P latency can be decomposed in:

a. *Acquisition/Tracking latency*: time required by software and hardware components of the VR headset to track head movements and hand gestures.

b. *Processing latency*: time taken by the application to incorporate the collected pose data in the message to be sent.

c. *Network transmission latency*: time needed to convey messages from the VR headset to the musicians' application.

d. *Rendering latency* (receiver device): time required to render the received data graphically.

e. *Video output latency* (receiver device): time required by the driver and hardware to display the rendered frames.

We consider three different latency-related measurements:

- *Network packet jitter*: message generation happens at fixed intervals (once per frame generation, i.e., $d = 1000/f$, where d indicates the delay in milliseconds and f indicates the frame rate, measured in frames per second (fps), of the VR headset). At the receiver side, we measure the inter-arrival time between two consecutive messages.

- *Messages RTT*: we measure the time difference between the dispatch of an "echo request" to the streaming device (or musician application), to the corresponding reception of an "echo reply" message. It is measured in terms of d, which represents our time granularity for every measure performed in the VR headset.

- *M2P latency of hand movement*: using a camera with a frame rate of 119 fps, we record the movement of a single hand executing an upward conducting gesture, close to the screen where the musicians' application is being displayed. We then manually annotate the video frames corresponding to the beginning and the end of a movement, for both the real hand and the rendered one. The

time difference between the frame annotated as the start of a given movement of the real hand and the corresponding frame for the rendered hand represents the M2P latency. It is measured in number of frames and then converted in milliseconds.

8.4 Results

8.4.1 Testbed Description

The test environment consists of a single machine concurrently operating as a streaming device, signaling server and receiver device. The machine is a laptop with an Intel i5-7300U, 12 GB of RAM and a Linux distribution (kernel 6.8.5 PREEMPT_DYNAMIC). The musicians' web-application runs in Chromium (version 123). The laptop is connected to an external monitor (HP e24m G4) through DisplayPort over USB-C. The monitor is used to display the video output visualized in the browser, based on which all M2P measurements were collected.

8.4.2 Transmission Jitter

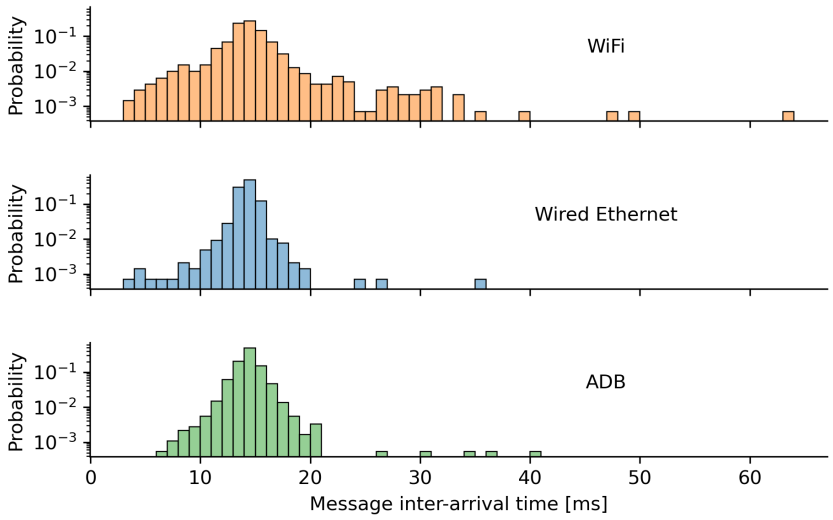


Fig. 8.4 Distributions of message inter-arrival times for the three transmission stacks. The ideal value should be ~ 14 ms $= 1000/72$ fps.

Fig. 8.4 compares the jitter values achieved by the three different connections methods available on the VR headset. The wired-Ethernet connection (through the Ethernet to USB-C adapter) results to be the most stable one in terms of jitter, followed by ADB, whereas the Wi-Fi connection yields the worst jitter figures. While these results suggest that the best solution would be the use of the Ethernet connection through a USB-to-Ethernet adapter, we preferred the ADB option for the reasons described in Sec. 8.2.1.

8.4.3 Messages RTT Measurements

Regarding RTT, we performed two measurements:

VR headset to streaming device (short path) The VR headset application sends an "echo request" at time t_1, and the application in the streaming device replies with an "echo reply". The VR headset receives the answer at time t_2. A single measure of the RTT (R) is equal to $t_2 - t_1$.

VR headset to musician application (long path) The application in the VR headset runs in the same way as above. The application in the streaming device forwards the "echo request" to a single connected musician's application, which replies to it. The streaming device forwards back the "echo reply" to the VR headset through ADB.

The short path RTT measurements yielded 99.7% of the measures exhibiting a value of $R = 1d$ and the remaining 0.3% exhibiting a value of $R = 2d$. Instead, the long path RTT experiment yielded 83.4% of the measures exhibiting a value of $R = 1d$, 13% with value $R = 2d$, 2.5% with value $R = 3d$, and the remaining 1.1% with values greater than $3d$. The RTT measured in terms of d can be converted in milliseconds by knowing the frame rate of the VR headset (f). The RTT expressed in ms (R_{ms}) is bound within the range $(R-1) <= \frac{R_{\text{ms}} \cdot f}{1000} <= R$. The total chain RTT (i.e., from the VR headset application to the musician's application) deserves further investigation but, assuming a symmetric RTT, we can infer that the one-way transmission latency is less than 21 ms in 99% of the cases (note that $f = 72$ fps on our VR headset). Pose data is rendered immediately upon arrival, since no buffering

is performed at the receiving device. This has the effect of minimizing the latency at the cost of occasional graphical frame stuttering.

8.4.4 M2P Latency of Hand Movement

Fig. 8.5 displays the measured M2P latency of hand movements. The camera used for recording had a frame rate of 119 fps. The median latency value is 17 frames, i.e., $\sim$ 143 ms. Based on the previously discussed RTT measurements, we know that latency due to transmission is lower than 21 ms. According to Abdlkarim et al. [185], the hand tracking latency for the Meta Quest 2 VR headset should be in the order of 40 ms. Other sources [186] state that the Meta Quest 3 should have a slightly lower hand tracking latency than the of Meta Quest 2, i.e., around $\sim$ 30 ms. This would imply that at most $\sim$ 90 ms are introduced by the processing time on board of the VR headset plus the rendering and output on the musician's application side.

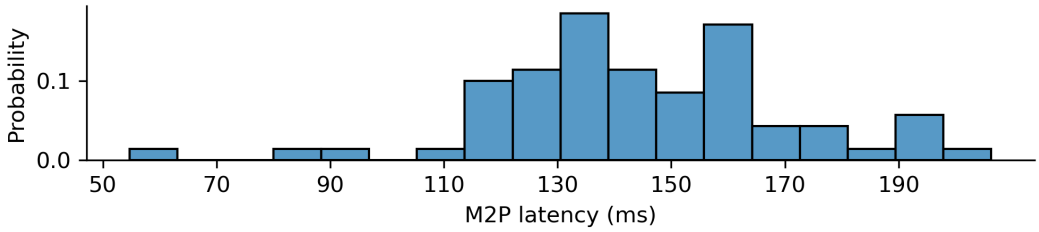


Fig. 8.5 M2P latency of the whole chain.

8.4.5 Bit Rate Requirements

Bit rate requirements can be computed deterministically, assuming that both hands are tracked and that messages are sent once per frame generation. For each message, we track the 6D pose of 24 bones per hand, for both hands, plus one pose representing the head. Overall, 49 6D poses are tracked. Each 6D pose consists of the Cartesian position and rotation data, both represented as triplets of 4 bytes floating-point values, totaling $2 \cdot 3 \cdot 4 = 24$ bytes per bone. Therefore, each message is 1176 bytes long, with an overhead of 24 bytes for our protocol implementation, thus yielding 1200 bytes. It follows that the bit rate required for each communication channel director-musician amounts to 86.4 kB/s, i.e., 691.2 kbit/s. It is worth noting that the communication is stateless, i.e., every message conveys the current state of the

system. Thus, a single loss only affects the receiver's state until the next message is received, avoiding any error accumulation issues.

It should be noted that further reductions of the bit rate requirements could be achieved by diminishing the bit-depth of position and rotation data representations, at the price of a small degradation of the accuracy of the rendered pose. However, such degradation would not worsen over time, due to the absence of error accumulation.

8.5 Limitations

Although the results obtained are promising, the proposed prototype has some limitations in its setup. Firstly, the current prototype is not designed to accommodate more than one musician per device. Indeed, a single device is represented as one avatar from the conductor's perspective, making it impossible to distinguish between multiple musicians. Secondly, the prototype relies on embedded motion tracking features, which are based on a camera paired with deep-learning algorithms, both of which introduce substantial latency. Sensor-based alternatives could be investigated. Lastly, to the best of our knowledge, commercially available VR headsets offer limited customization options from both hardware and software standpoints. For instance, state-of-the-art NMP applications often rely on custom scheduling, as seen in JackTrip [8], and on the use of wired Ethernet connections, which was not feasible with the selected VR headset.

8.6 Summary

This chapter proposed a remote orchestra conduction setup for NMPs, consisting of a VR headset that tracks the conductor's hands and head movements and streams them to remote musicians, who visualize pose data via a web-based application. We assessed the feasibility of the proposed system by measuring the achievable M2P latency and bit rate requirements, based on the connectivity options available on the VR headset.

Results showed that latency values lie in the same order of magnitude as the human visual reaction time (i.e., around 600 ms as reported in [187]) and are comparable to those reported in [115], where the minimum noticeable delay is estimated

to range between 160 ms and 320 ms for remote drum players[5]). Moreover, bit rate requirements are at least one order of magnitude lower than those of state-of-the-art video streaming.

Future research efforts will be devoted to define and perform subjective tests able to assess the usability of the setup (e.g., to evaluate the impact of latency on the quality of experience perceived by the performers). As a future development, we also plan to integrate facial tracking as a mean to convey intentions and emotions to orchestra members via the conductor's facial expression. Moreover, a system enabling remote musical conduction based on motion tracking also paves the way for forthcoming research efforts aimed at conveying the conductor's gestures to visually impaired musicians through haptic feedback mechanisms, coupled with a visual representation [188]. Furthermore, we foresee a robot conductor that leads an orchestra by actuating in real time the gestures performed by a remote conductor. In this context, we plan to study the interaction between human musicians and the robotic conductor, and how the audience perceives the proposed technology.

[5]It is worth mentioning that it is common practice for conductors to anticipate their gestural cues w.r.t. the desired timing of the orchestra reaction, in order to leave time to the musicians to process and react to the visual hints. Thus, we expect conductors to be able to compensate for the M2P delay introduced by our proposed system by adapting the anticipation time of their gestures.

Chapter 9

Collection of Design Directions for a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students

This chapter presents the results of a survey campaign aimed at gathering design directions for the realization of a visual interface with haptic feedback. The aim of the interface is to ease the conveyance of the concept of "sonic grain" to Deaf and Hard of Hearing (DHH) music students. Results from the questionnaire were leveraged for the realization of a prototype which exploits cross-modal associations among images, colors, sounds and textures to render different types of sonic grains and offer a multisensorial perceptual experience to the users. This prototype represents a promising starting point for further investigation on how to jointly exploit visual, auditory and haptic feedback to support more inclusive pedagogical approaches to music teaching.

The content of this chapter is part of a contribution presented in the 4th International Symposium on the Internet of Sounds in October 2023 [52].

9.1 Introduction

Musical education of DHH students is particularly challenging due to the essential involvement of auditory perception [189]. For such categories of subjects, the adoption of extra-auditory cues is crucial to ease music perception. Indeed, the process of hearing involves a multiplicity of perceptual modes [190], even beyond merely aural functions.

In particular, the notion of *sonic grain* conceptualizes music as a form of *tactilization* of sound that enables a cross-modal mapping of texture between the microstructures of sound and material surfaces [191], thus bridging tactile and sonic experiences. According to [120], high granularity sounds have rapid sonic impacts, which may sound like a sequence of discrete attacks, a cluster of smaller sounds, or something "noisy" with sonic interference. In contrast, low granularity "soft" sounds have little surface fluctuation or have regular pitch fluctuations with smooth transitions. Most soft sounds have a clear tone emanating from a sustained vibration, while coarse-grained sounds tend to have an indeterminate pitch and typically arise from percussive impacts. The similarities in microstructure lead to an understanding of grain as a perceptual phenomenon that works across sensory modalities, suggesting that tactile, auditory, and visual experiences of texture are not only comparable, but also intertwined and concretely interconnected. Indeed, the grain of sound can provide information to the listener about the texture and physical state of the objects used to produce it.

Therefore, exploiting sonic grain for the multisensory conveyance of music to DHH subjects appears as a promising direction to support their musical education and its related practices. In particular, our goal is to help DHH students learn how to discriminate among sounds generated by different musical instruments via the cross-modal presentation of their corresponding sonic grain. However, while cross-modal correspondences between auditory, visual, and haptic stimuli have been widely documented [192], their relationship to more complex and highly subjective synaesthetic mappings remains an active area of research, with only a handful of studies examining the nature of such phenomena (see e.g., [193–195]). Thus, the need to empirically create a "dictionary" of associations between tactile and sonic experiences arises.

To fill this gap, we present the results of a questionnaire offered to 56 respondents, to collect statistics about sensory associations between musical instruments and colors, shapes, and textures, inspired by listening to musical excerpts. We then proceed to provide some details on how we leveraged those results for the realization of an interface prototype. The interface associates to recorded musical excerpts, generated by different instruments, a color, an image of a shape and a texture, according to the "grain" of the sound emitted by the instruments. The aim of this study is to provide design directions for the realization of a programmable visual interface with customizable haptic feedback to convey the concept of sonic grain to DHH music students.

We envision this interface being used either in a stand-alone scenario, or as building block of an NMP framework for e-learning purposes, where music excerpts could be retrieved on-demand by a remote instructor from a cloud-located repository and offered to the student, or even generated in real-time and automatically associated to cross-modal features through a dedicated mapping algorithm, based on the spectral characterization of the sound. This would pave the way for novel e-learning opportunities specifically tailored for DHH music students, following recent online teaching trials in other educational subjects [196–199].

The remainder of the chapter is organized as follows: Sec. 9.2 provides some background notions on DHH impairments and the programmable haptic interface adopted for the envisioned prototype. The results of the survey and the derived interface prototype are presented in Sec. 9.3. Some practical considerations about the interface implementation are offered in Sec. 9.4. Finally, Sec. 9.5 summarizes this chapter.

9.2 Background

9.2.1 Basic Notions on DHH Impairments

According to [200], hearing loss stands out as one of the most frequently encountered disabilities on a global scale, with over 430 million people experiencing more than 35 dB of hearing loss in their least impaired ear. Moreover, hearing loss can manifest at various stages throughout the lifespan, attributable to diverse factors, including

but not limited to genetic predisposition, infections, accumulation of fluids in the ear, chronic medical conditions, and exposure to high-intensity sounds.

Hearing loss can be divided into several categories, depending on the anatomical location of the impairment within the auditory system. The principal categories are the following:

- *Conductive Hearing Loss*: this type of impairment arises from pathological conditions affecting the conduction of sound waves from the outer/middle-ear to the inner ear. Typical causes are excessive earwax, disorders affecting the eardrum, etc.
- *Sensorineural Hearing Loss*: it ensues from damage to the inner ear (cochlea) or the auditory nerve, representing the most prevalent form of permanent hearing loss. Usually, it is caused by age-related degeneration, prolonged exposure to loud sounds, hereditary conditions, etc.
- *Mixed Hearing Loss*: this is characterized by a combination of both conductive and sensorineural hearing losses.

Hearing loss usually affects the high frequencies, but it is possible to experience hearing loss of some specific frequencies. Due to these issues, music is perceived by DHH subjects in different ways.

9.2.2 Attributes for Music Characterization

Though the definition of music varies across cultures, we can describe it objectively by considering its properties. In [201], Levitin identifies nine main attributes:

- *Tone*: a discrete musical sound (what you hear). *Note* is also used, but it refers to something written on a score (what you see).
- *Pitch*: a psychological construct related to both the frequency of a particular tone and its position on the scale
- *Timbre*: the tonal color which allows distinguishing two different musical instruments when playing the same note.

- *Contour*: describes the shape of a musical melody, considering only the patterns of "up" and "down".
- *Rhythm*: pertains to the time duration of a sequence of musical notes and how these notes are organized into distinct groupings or patterns.
- *Tempo*: the overall speed or pace of a piece.
- *Loudness*: a psychological construct related to how the amplitude of a given tone is perceived.
- *Spatial location*: position of the source of a given tone in a three-dimensional space.
- *Reverberation*: perception of the distance of a given sound source in combination with the size of the room/hall in which the sound source is placed.

These attributes are separable, according to Levitin [201], which means that varying one will not affect the others. Though, the difference between *music* and a *random set of sounds* has to do with how these attributes combine. In [201], Levitin explains that the combination of such fundamental attributes creates higher-order concepts:

- *Metre*: combination of rhythm and loudness cues that refers to the way in which tones are grouped together across time.
- *Key*: a hierarchy of importance between tones.
- *Melody*: the primary theme of a musical composition. A melody is created by arranging a sequence of notes from its designated key, each with distinct rhythm, contour, and other musical characteristics.
- *Harmony*: the relationship between the pitches of different notes. It can refer to a chord progression or refer to parallel melodies to the primary one.

While the attributes and higher-order concepts can be understood by non-hearing-impaired subjects, when it comes to DHH subjects those concepts are challenging to describe. Since audio by itself cannot be fully perceived, alternative representations need to be used. Among them, visual feedback or vibration of haptic surfaces can be

exploited to represent some of these attributes and concepts. For example, pitch can be associated with color, loudness with its intensity, timbre with surface roughness, rhythm with surface vibration, etc.

9.2.3 Programmable Graphical Interface with Haptic Feedback

As described in Sec. 2.2.5, previous studies already experimented with visual, auditory and vibrational/haptic feedback to enrich the representation of musical content. However, one of the challenges of designing a solution for multimodal music conveyance is the reproducibility of the setup, which in some cases may not be feasible. For such reason, for the realization of our prototype we opted for an off-the-shelf haptic device which can be used to provide a multimodal experience, through auditory, haptic and visual feedback, with the goal of conveying to DHH students the concept of "sonic grain". Another reason that led us to opt for such a kind of device is its versatility. The ability to expand its features beyond the mere haptic feedback opens up possibilities for its integration into NMP frameworks, which would unleash novel opportunities for remote music education, particularly for DHH students.

In this study, we used the TanvasTouch device[1], which enables a customized design of tactile feedback localized to the user's fingertip without the need for surface vibration, by exploiting electroadhesion-based haptic technology. This way, a new software-defined interaction layer can be overlapped to traditional visual representation [202]. Haptic textures are designed using a simple image-based metaphor: black and white images are leveraged, where white corresponds to rough texture and black to smooth texture. These are integrated via APIs, and then rendered in real-time thanks to the TanvasTouch Engine, which is a software driver executed on a general purpose machine that generates friction control commands to be sent via a USB cable to the TanvasTouch Controller, that in turn modifies the friction surface accordingly. In parallel, the images displayed on the TanvasTouch screen can be updated in real-time via an HDMI cable connecting it to the machine.

[1]Tanvas. Surface Haptics. https://tanvas.co/technology

9.3 Exploratory Study

In this section, we discuss the outcomes of the survey and present the derived cross-modal associations to be displayed via the interface. We then briefly describe the graphical presentation of our interface prototype.

9.3.1 Choice of Cross-Modal Associations

To investigate meaningful cross-modal associations among the timbral characteristics of musical instruments and colors, shapes and textures, we administered a questionnaire to a pool of 56 participants, composed of conservatory/university students and staff (including professors, researchers, and PhD students) of both genders and various ages. The goal of such collection was to identify the sonic grains most typically associated with different types of instruments, to render them via the graphical interface. Given the inherent requirement to listen to the various audio excerpts in order to find associations between the sound and visual or tactile characteristics, the questionnaire was only administered to participants without hearing impairments.

The questionnaire included 12 questions, one for each of the chosen musical instruments. The chosen musical instruments were: Harp, Piano, Tenor Sax, Violin, Harpsichord, Guitar, Accordion, Flute, Horn, Hurdy-gurdy, Eardrum, and Trumpet. In each question, the respondent was first asked to listen twice to an audio excerpt of a duration of 15 s to 20 s. Once the listening phase was completed, four questions appeared, asking the participants to associate the audio excerpt with one to three options from each of the following categories:

- **Color**: Black, Blue, Brown, Gray, Green, Pink, Purple, Red, White, Yellow, Light Blue, Orange
- **Light**: Cold, Dazzling, Hot, Lunar, Solar, Weak
- **Shape**: Big, Conic, Cubic, Curved, Filiform, Pyramidal, Rounded, Sharp, Small, Spherical, Flat, Thick, Irregular
- **Texture**: Coarse, Cold, Hard, Hot, Iron, Smooth, Soft, Squashy, Woody, Elastic, Pungent

For each category, an 'Other' option was also provided, allowing participants to specify additional entries not included among the predefined options. For the sake of limiting the time required for completing the questionnaire, a reduced version was also produced, containing a subset of 6 questions obtained by random choice among two predefined splits of the initial set. Students were offered the reduced version of the questionnaire, whereas staff members answered the full version. On average, around 30 sets of answers per instrument were collected. A summarized version of the results is reported in Tab. 9.1, where for each instrument and category we listed, in descending order, the two/three most selected attributes. The complete set of associations of attributes to the different instruments and categories is shown in Fig. 9.2.

The above-reported results were then used for the selection and adjustment of the various images to display on the interface prototype. The final attributes were selected as follows:

- *Color*: the pair of most voted options were selected and blended (for example, if yellow and green were the two most voted colors, the resulting color adopted for visualization was lime). Minor adjustments of the blends were introduced to avoid associations of the same color to multiple instruments.
- *Light*: the most voted option was selected and the brightness of the displayed color was adjusted accordingly.
- *Shape and Texture*: the most voted option per category was selected, and the image of an object with characteristics matching the identified pair was chosen. For example, a combination of rounded shape and smooth texture was associated with the image of marbles.

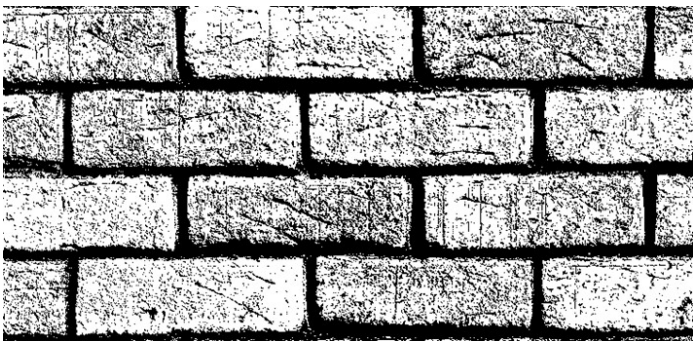
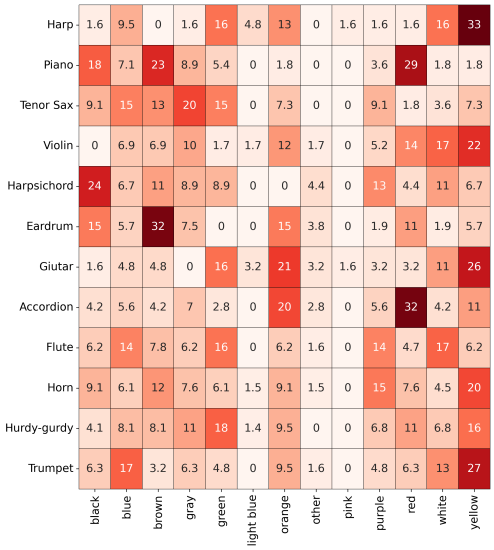


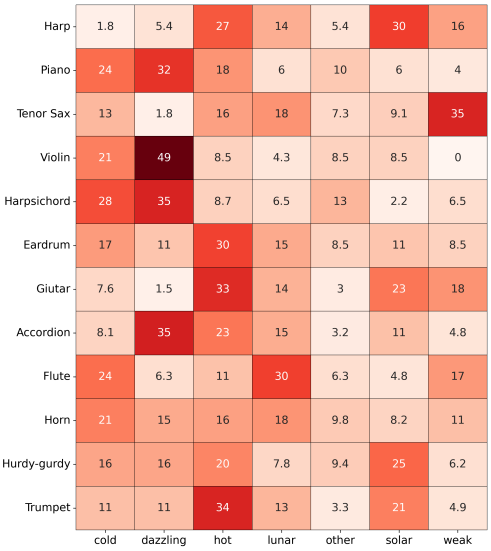
Fig. 9.1 Black and white image rendering for haptic texture design

Table 9.1 Listening Test Results - Summary - For each instrument and category, we listed the two or three most significant associations. The associations are sorted by significance. When two options received the same number of responses, they are presented using the format A/B to indicate equal ranking.

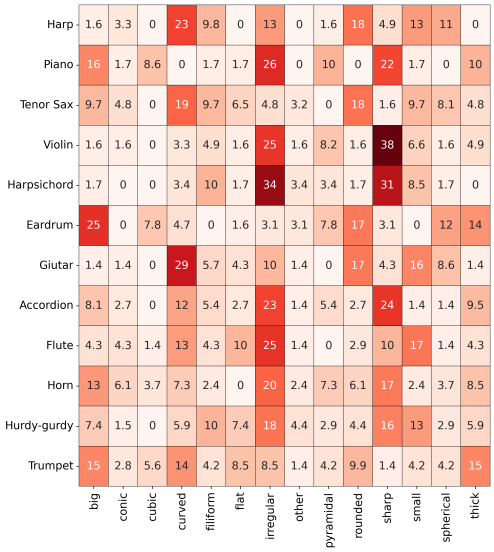
Instrument	**Color**	**Light**	**Shape**	**Texture**
Harp	Yellow White Green	Solar Hot	Curved Rounded Small	Soft Smooth
Piano	Red Brown Black	Dazzling Cold	Irregular Sharp Big	Hard Iron
Tenor Sax	Gray Green/Blue	Weak Lunar	Curved Rounded	Soft Smooth
Violin	Yellow White Red	Dazzling Cold	Sharp Irregular	Pungent Iron
Harpsichord	Black Purple Brown/White	Dazzling Cold	Irregular Sharp	Pungent Iron
Eardrum	Brown Black Orange	Hot Cold	Big Rounded	Hard Hot
Guitar	Yellow Green Orange	Hot Solar	Curved Rounded Small	Soft Smooth/Hot
Accordion	Red Orange Yellow	Dazzling Hot	Irregular Sharp Small	Pungent Coarse/Hard
Flute	White Green Purple	Lunar Cold	Irregular Small	Cold Woody Soft/Smooth
Horn	Yellow Purple Gray	Cold Lunar	Irregular Sharp Big	Iron Coarse/Pungent
Hurdy-Gurdy	Green Yellow Red/Gray	Solar Hot	Irregular Sharp Small	Woody Pungent Coarse
Trumpet	Yellow Blue White	Hot Solar	Big Thick Curved	Smooth Soft/Hard Hot/Iron



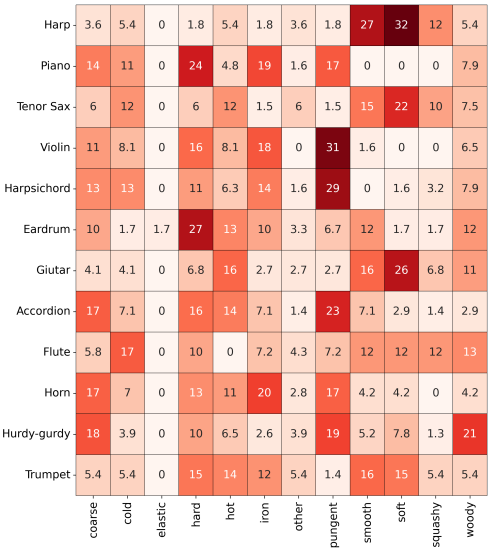
(a) Matrix of associations of colors to instruments



(b) Matrix of associations of light options to instruments



(c) Matrix of associations of shapes to instruments



(d) Matrix of associations of texture options to instruments

Fig. 9.2 Listening Test Results

The selected images were then recolored according to the associations found through the questionnaire, and their b/w version was used to generate the superimposed haptic texture (see Fig. 9.1 for an example). In some cases, some post-processing was necessary, e.g., to invert black and white to better match the coarse/smooth areas of the image.

9.3.2 Visual Presentation

Our envisioned interface is divided in two parts: a sidebar on the left, used for navigating among the different pages, and a main section on the right, which contains, for each page, up to 4 subsections. Each subsection includes 4 main elements: a button on the top left, to play and pause the associated audio track, a picture of the instrument audible in the audio track, a background image, with a chosen background color, giving a visual representation of the "texture" associated to the audio track, and a haptic image which provides the haptic feedback of the "texture" associated to the audio track. Once a given page is selected, the associated images and elements will appear, as shown in Fig. 9.3. For each of them, when pressing the play button, the associated audio track will be played and the texture file will be loaded. The haptic feedback is available only while the audio is playing. This is done to help focus the attention of the user on one sound at a time, while avoiding being distracted by other textures when a given sound is reproduced. The final prototype is shown in Fig. 9.3.

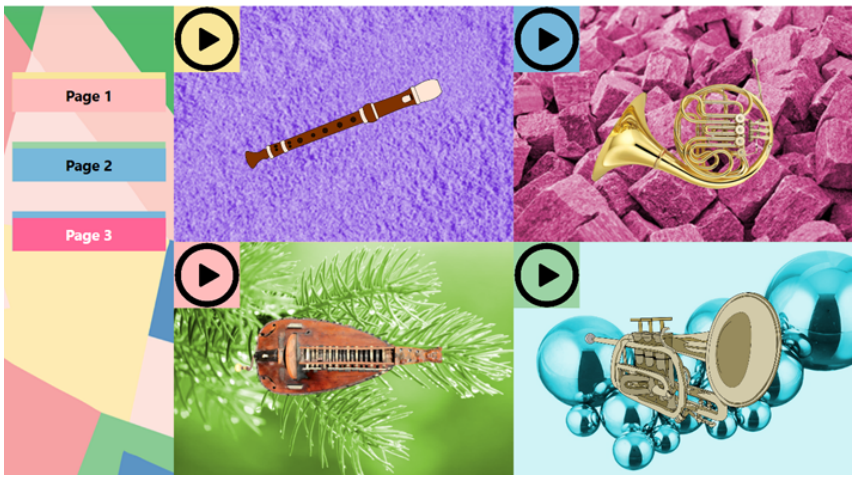
9.4 Implementation Details

In this section, we provide some details on the implementation of our TanvasTouch-based prototype, and we offer suggestions on how to tackle practical issues that may occur when developing a GUI for such a device. Our preliminary version is developed in C# using the Windows Presentation Foundation (WPF) UI framework of the .NET framework[2]. To load the haptic images on the TanvasTouch device, we

[2]The code, audio excerpts and images that compose the interface are publicly available at: https://github.com/mariasanguesa/TanvasTouch_interface.

(a) First page

(b) Second page



(c) Third page

Fig. 9.3 Screenshots of the developed interface

used the TanvasTouch API[3] provided by the TanvasTouch Software Development Kit (SDK).

The WPF framework uses XAML (Extensible Application Markup Language) to declaratively define the GUI elements, and uses C# to implement the associated functions. In our case, the XAML code defines the various GUI elements on the page, like buttons, images, the layout etc. The C# code instead controls the visibility of the GUI elements by showing and hiding the images, for enabling navigation between pages, for playback of audio tracks and for loading and removing the haptic feedback images. Images are implemented in the XAML file using the `Image` element, while the haptic feedback is implemented using the `TSprite` element of the TanvasTouch API.

A common issue that arises when designing textures with the TanvasTouch device is making the visualized image overlap with the haptic textures, due to the coordinate system that the device uses to place images on the screen. By default, visual coordinates are used. However, the dimension and position of the `TSprite`, which is the on-screen haptic object, must be defined in the screen coordinates system. To achieve the above, the size, and position of `TSprite` must match the size and position of the graphic image multiplied by the `dpiScale`. The `dpiScale` represents the screen scaling factor of the Tanvas device. For example, if the scaling factor is 125% the `dpiScale` value will be 1.25.

Another aspect to note is that the coordinates in the screen coordinate system are relative to the main screen. That is, the point (0,0) will be the top-left corner of the main screen. If a second screen is added, as it happens when the TanvasTouch device is connected, the coordinates of what is presented on the second screen will be relative to the main screen. For example, if the second screen is placed to the left of the main screen, the coordinates on the horizontal axis will be negative for every object displayed on the second screen.

[3]TanvasTouch .NET API documentation is available at: https://api-docs.tanvas.co/tanvastouch/dotnet/5.0.1/api/index.html

9.5 Summary

This chapter presents the results of a survey campaign involving 56 participants, aimed at defining design directions for the realization of a visual interface with haptic feedback to convey the concept of "sonic grain" to DHH music students. These results were then used for the realization of an interface prototype which exploits cross-modal associations among images, colors, sounds and textures to render different sonic grains and offers a multisensorial perceptual experience to the users.

Such results represent a promising starting point for further investigation on how to jointly exploit visual, auditory and haptic feedback to support more inclusive pedagogical approaches to music teaching. Reaching consensus on cross-modal associations between sounds and visual/tactile experiences to create a common "dictionary" is an open research direction, which should be promptly addressed to guide hardware/software designers in the realization of innovative interfaces for accessible fruition of musical content and multisensory music learning.

It is important to note that, throughout the research presented in this chapter, we actively communicated with two music educators specialized in teaching children with hearing impairments. The development of the questionnaire, including the selection of musical excerpts, the categorization, and the provided response options, was guided by the feedback provided by these experts. However, due to practical limitations, DHH individuals were not directly involved in the design or implementation of the prototype. Consequently, the validity of the reported findings has yet to be established, and future research will aim to assess this through subjective evaluations with actual DHH participants.

In future works, we also plan to expand the capabilities of the Tanvas device by finalizing the prototype and integrating functionalities to support cross-modal display of an audio track either acquired in real-time, or retrieved on-demand from a remote repository. Moreover, we plan to investigate the adoption of dynamic visual and haptic representations, where the time dimension comes into play when matching and tailoring visual and haptic content to the audio stream being reproduced. Finally, we intend to experiment with remote interaction capabilities, integrating our prototype in an NMP framework to enable the conveyance of "sonic grain" also in the context of remote music teaching: this would unleash novel e-learning opportunities for DHH

subjects and help to overcome a number of barriers currently being experienced in online education of DHH students.

Chapter 10

Towards an Inclusive Framework for Remote Musical Education and Practices

This chapter provides an overview of inclusivity in remote music education and NMPs, highlighting issues and difficulties encountered by users with visual, auditory, or motor impairments when approaching such practices. Audiovisual tools, motion tracking systems, immersive audio and haptic feedback technologies that can be leveraged to enhance the accessibility of web-based platforms for online music teaching and remote performances are analyzed and classified. Moreover, it discusses the integration of such technological solutions into an NMP framework and their potential benefits for various categories of users with disabilities, describing some exemplary use-cases. Finally, open technical challenges for the practical realization and deployment of such a framework are identified, as well as future research directions towards its incorporation within the MM framework.

The work presented in this chapter is part of a contribution published in the IEEE Access Journal in November 2024 [56].

10.1 Introduction

Information and Communication Technologies (ICT) and Artificial Intelligence (AI) are introducing dramatic changes in the way pedagogical processes are conceived: distance education and computer-based learning have started proliferating in the last decade and are expected to further expand [203], fostered by novel, interactive and immersive approaches enabled by technologies such as IoT, Extended Reality/Virtual Reality (XR/VR), and the Metaverse [204]. This implies a global rethinking of the fruition modalities for educational services.

There are several categories of subjects who would benefit from the long-term adoption of distance learning frameworks. Among those, students with visual/auditory/motor impairments or special needs, for whom traditional in-presence access to education is challenging or even precluded, represent a non-negligible share. According to [205], in 2020/21 15% of US public school students received special education services, for a total of 7.2 millions. Such amount comprises well over 1 million students diagnosed with health impairments resulting in limited strength, vitality, or alertness, around 700 thousand students with hearing impairments, and around 350 thousand students with orthopedic/visual impairments or deaf-blindness. The same source reports that 1% of students with special educational needs are homebound or confined in hospitals or separate residential facilities.

As of today, remote teaching heavily relies on videoconferencing technologies, which are mainly designed to maximize speech intelligibility while reducing the required transmission bit rate. This results in the adoption of low sampling frequencies and highly efficient compression codecs, at the price of causing voice signal distortions. Moreover, the encoding/decoding processes add consistent latency to increase resiliency to packet losses [17]. Differently from videoconferencing, in the application field of remote music teaching and NMP scenarios, where musicians play together in real-time from distance by exploiting audio streaming over wide area networks, specific requirements emerge, such as:

- *Strict latency guarantees*: in the order of at most 30 ms [27], to ensure realistic performative conditions.
- *High reliability*: to ensure high-fidelity quality of the audio playback.

- *Appropriate computational capabilities for audio processing, mixing, and rendering*: to ensure scalability to large deployments (e.g., for music classes, ensembles, choirs, or orchestras).

Refer to Sec 1.1 for a more detailed overview on NMP system requirements. As a consequence, the use of off-the-shelf videoconferencing technologies is not satisfactory in remote musical education and NMP scenarios. In such applications, audio codecs worsen the perceived end-to-end delay and reduce the quality of the streamed audio signal due to their compressive effects. Moreover, an adequate backend infrastructure to support low-latency and high-fidelity audio mixing is needed. The aforementioned limitations clearly emerge in [206], which highlights the difficulties encountered by music schools in delivering group lessons and supporting rehearsal sessions using commercial e-learning technologies during the first lockdown enforced to mitigate the spreading of the SARS-CoV-2 pandemic.

The previously identified limitations significantly restrict remote music education possibilities, particularly for individuals with disabilities. Therefore, this chapter aims to accomplish three primary objectives:

- offer a comprehensive overview of the educational and performative needs experienced by disabled users in the field of remote musical teaching and practices;
- identify the technological solutions that could address such needs;
- discuss their integration in existing NMP frameworks to enhance inclusivity, highlighting open challenges and future research directions.

The remainder of this chapter is organized as follows: Sec. 10.2 reviews accessibility and interaction requirements to enable fruition of remote musical education and practices by several categories of users with disabilities. It also discusses the main technological solutions for the realization of accessible human-machine interfaces tailored to musical performances. Sec. 10.3 overviews existing NMP applications and compares their design choices in terms of implementation and infrastructure. Sec. 10.4 proposes an inclusive NMP framework and details its main building blocks. Some illustrative use cases are presented in Sec. 10.5. Open technical challenges and future research directions involved in its realization are discussed in Sec. 10.6 and 10.7, respectively. Finally, 10.8 summarizes this chapter.

10.2 Accessible Human-Machine Interfaces for Musical Education

10.2.1 Educational and Performative Issues Experienced by Disabled Users in Music-Related Fields

Table 10.1 Overview of Educational Issues of Disabled Music Students

Type of Impairment	**Issues**	**Countermeasures**
Blind and Visually Impaired (BVI) students	Being music notation predominantly visual, it cannot be written, read or directly reproduced by the majority of BVI musicians, which heavily hinders their capability of conveying their creative intentions [207–209];	Availability of braille scores [210], modified musical instruments [211] and accessible music production interfaces [212].
DHH students	DHH subjects experience difficulties in fine pitch discrimination, which often precludes practicing with instruments such as violin, cello or brass instruments, all of which require very precise intonation. Moreover, DHH musicians rely on visual cues to compensate for their reduced hearing capabilities, which may constrain their physical placement when playing in ensembles, bands, and orchestras [213].	Choice of instruments which easily convey vibrations to DHH players (e.g., guitar, electric bass). Strategic pairing of non-DHH to DHH performers to ensure dedicated guidance/cues [213]. Availability of modified musical instruments [211].
Mobility impaired students	Subjects with motor impairments often cannot play traditional musical instruments or use music production software due to lack of accessibility [214].	Wide offer of adapted musical repertoire (e.g., one-handed piano pieces) [210]. Availability of modified musical instruments [210].

Research on how to accommodate the needs of students with disabilities in the field of music education started more than four decades ago, as testified by two surveys [215] and [216]. Such studies triggered further investigations on how music teaching can embrace inclusivity and diversity [217]. In Tab. 10.1, we report some

of the most widely recognized barriers that preclude or strongly limit the inclusion in musical education of subjects with various types of disabilities, as well as a number of resources available to educators to make musical practices accessible to students with special needs.

In the following sections, we review the main technological solutions already adopted for in-presence inclusive musical education. In particular, in the past two decades, the research areas referred to as Musical XR [218] and IoMusT [29] have started being investigated in the scientific community. The former introduced a paradigm shift by disrupting traditional notions of musical interaction and enabling performers and audiences to interact musically with virtual/augmented objects, agents, and environments [219], also in the context of music education [220]. Conversely, the latter identifies computational devices integrated into physical objects (e.g., smart musical instruments or wearables with music-related purposes) devoted to generating or receiving musical content, interconnected via a telecommunication infrastructure. Such smart musical objects can also be leveraged for educational purposes. Finally, we complete the section by reviewing some literature-reported use-cases that testify the adoption of the presented technologies in inclusive music education trials.

10.2.2 Haptic Feedback Techniques

A number of studies have investigated the conveyance of musical information through the sense of touch (see [119, 221] for a thorough overview). Tactile rendering setups normally focus on capturing a single music feature, e.g., rhythm, pitch, melody, timbre, and loudness.

The usage of musical haptic wearables to improve communication among performers sharing the same physical environment has been envisioned in [222]. The adoption of haptic feedbacks to enrich the emotive experience while listening to music has also been experimented in [121, 223]. Furthermore, systems that transpose gestural cues into haptic feedbacks have proven to be useful to enrich musical perception and interplay for Blind and Visually Impaired (BVI) and Deaf and Hard of Hearing (DHH) performers [124].

Notably, the sensitivity of the tactile system is quantified in [224] in terms of frequency, intensity and temporal features. The study reports that, through

haptic stimulation at the arm, wrist, or hand, approximately 40 different frequency steps can be discriminated. Moreover, in [224] amplitude modulation sensitivity was found to be higher when using a carrier tone at 250 Hz. The perception of simultaneity between audio and haptic signals depends on several factors, including the stimulated body part, the characteristics of the auditory stimulus, and the type of haptic device used. Reported temporal integration windows vary accordingly: some studies estimate the threshold for perceived simultaneity to be around 25 ms [225], while others suggest broader windows of up to 50 ms [226] or even 100 ms [227]. In terms of bit rate requirements, tactile feedback ranges from tens to hundreds of kbit/s, depending on the type of stimuli being involved.

10.2.3 Motion Tracking Techniques

In the context of music education, several studies have already demonstrated the benefits of using computer technologies for BVI people [228]. However, no scientific study has yet tackled the integration of motion tracking or haptic feedback functionalities in NMP systems to accommodate the special needs of disabled users. The exploitation of Kinect cameras for telerehabilitation has been investigated in [229] using a WebRTC-based communication protocol, though without integration of audio streaming functionalities. Kinect cameras for motion data acquisition have already been adopted to analyze gestures of music performers [230, 231].

Notably, from a networking standpoint, streaming pose data for avatar rendering rather than video frames significantly reduces transmission bit rate requirements, as it only needs to convey the Cartesian position and rotation data of the represented joints. This results in savings of at least one order of magnitude compared to encoded video streaming, and potentially up to three orders of magnitude for uncompressed video. Regarding M2P latency, experimental values are below 40 ms [232], which is comparable to the delays introduced by a camera streaming compressed video frames. This would foster scalability in large deployments involving a high number of participants, (e.g., choirs or orchestras), bypassing the need to convey a dedicated video stream to each of them.

10.2.4 Immersive Audio Rendering

Immersive audio is a sound rendering methodology that produces a three-dimensional (3D) acoustic space, facilitating the listener's perception of sound sources from distinct spatial directions. For the specific use-case of NMP applications, the usage of headphones is predominant compared to surround sound systems, due to their portability, ease of configuration, and lower cost. In such a scenario, binaural audio rendering is typically the selected immersive audio solution. It relies on the rendering of acoustic cues such as interaural time/level differences and acoustic filtering (i.e., the spectral information that depends on the characteristics of the user's physical attributes such as the shape of ears, head, shoulders, torso). This spatial rendering is accomplished using Head-Related Transfer Functions (HRTFs), which model the directional transfer of sound to the listener's tympanic membranes [233].

Spatial audio technology has already proved to enhance the sense of virtual imagery and spatialization for BVI users [234–237]. In the context of musical practices, 3D audio rendering can be leveraged to recreate a shared acoustical scene at the premises of every remote participant. Therefore, spatialized sound source placement will likely facilitate the participation to ensemble performances of BVI subjects, who could take advantage of the spatialized placement of sound sources to enhance their perception of the auditory cues necessary to synchronize their playing with that of the other performers. Similarly, DHH users could benefit of a personalized design of the acoustic scene rendered at their premises, tailored to their specific auditory impairment.

Though a few examples of immersive and multimodal environments specifically designed to permit fruition also by subjects with disabilities exist (e.g., [238]), no attempt to integrate immersive audio rendering systems in inclusive NMP frameworks has yet been reported in the literature. However, immersive audio settings for NMPs have started being investigated and assessed [116, 239–241].

Concerning bit rate requirements, they depend on the desired sampling rate and number of bits per sample, as well as on the number of audio channels to be supported. For example, an uncompressed mono audio stream at 48 kHz using 16-bits resolution accounts for 768 kbit/s. Such amount scales linearly with the number of audio channels. It follows that scenarios involving multiple participants and complex spatialized audio renderings may have significant requirements in terms

of data volumes to be transferred, potentially non-compatible with up/downstream bit rates supported by typical residential connectivity.

10.2.5 Exemplary Field Trials

In in-presence music teaching, vibrotactile feedbacks have been already demonstrated to convey the concept of rhythm and of the spectrum of sound frequencies to BVI and DHH students [122, 123]. Moreover, motion tracking technologies in connection with sound and visual production have been experimented to ease musical expressions of subjects with mobility or cognitive impairments [242]. The potentialities connected to the adoption of virtual and augmented reality technologies for music teaching have been investigated in several studies, especially for primary education [243], focusing on mobile settings [244], and even including disabled students [220]. Notably, the combination of passive haptics with VR has proved useful for music conduction education [245].

Focusing instead on on-line music education, several pilot studies conducted with traditional videoconferencing systems have been documented (e.g., [246]), especially during the SARS-CoV-2 pandemic [247, 248]. However, as emerged in [249], sustainable technology integration in music classrooms is still far from being achieved at a large scale, mainly due to barriers such as lack of digital literacy, non-affordability of equipment costs and scarce alignment with the instructors' pedagogical praxis. This outcome is in line with the general trend observed in e-learning-based education, as testified by numerous studies on its technological acceptance in various geographical areas (e.g., [250–253]). Another issue is the economical sustainability of ICT-based educational solutions, for which dedicated pricing models and cost-benefits analyses must be devised, as recently investigated in [254–256].

10.3 NMP Applications and Backend Infrastructures

10.3.1 Main Functionalities of NMP Systems

The core functionality of NMP systems is the support of professional-quality low-latency audio streaming, i.e., with M2E latency figures comparable to those achiev-

able in in-presence settings (see Sec. 1.1). Furthermore, NMP systems may provide additional features, such as:

- local audio manipulation capabilities (e.g., recording, reproduction of pre-recorded streams, adjustment of volumes, equalization levels and panning).
- local digital audio workstations.
- communication and interaction interfaces such as chats, user status tracking etc.
- collaborative music composition/editing/production tools.

For what concerns video support, only a few NMP solutions integrate video streaming capabilities. Unfortunately, typical video acquisition and encoding/decoding delays significantly exceed the target NMP M2E latency threshold. Therefore, the offered video-stream is usually desynchronized w.r.t. the corresponding audio-stream, thus heavily hindering reliance on visual cues during real-time remote musical interactions [257]. Alternatively, video may be streamed without being encoded (as in [258, 14]), though, this requires an adequate telecommunication infrastructure capable of supporting an uplink/downlink bit rate of a few Gbit/s. For an overview of NMP requirements and available NMP systems, refer to Sec. 1.1 and Sec. 2.2.1.

10.3.2 Peer-to-Peer VS Client-Server Architecture

Depending on the implementation choices and on the scalability needs of the users, either C/S or P2P architectural solutions are adopted. Typically, P2P architectures offer lower M2E latency than C/S ones, but allow only a few musicians in the same session. Conversely, for larger deployments C/S approaches are preferred, at the price of added latency caused by the server operating as an intermediate relay node. The scalability limitations intrinsic to P2P architectures lie in the fact that, in a session with N participants, each one sends and receives $N-1$ audio streams. If the audio stream consists of uncompressed PCM stereo audio signals sampled at either 44.1 kHz or 48 kHz and a bit depth of 16 bits, the required uplink/downlink bitrate is around 1.4 Mbit/s or 1.5 Mbit/s per participant. More sophisticated configurations, involving more than two channels per user, further increase such figures. It follows that the uplink bitrate offered to residential customers by Internet Service Providers

(ISPs) may be inadequate for P2P NMP sessions with many participants. In such cases, a C/S configuration with a server node equipped with audio synchronization and mixing capabilities is typically leveraged. The server collects the audio streams generated by every musician in upstream, mixes them at the server's premises and produces a single stereo (or multichannel) signal to be delivered in downstream, thus drastically reducing bit rate requirements per participant. In this scenario, each participant only sends a single audio stream to the server and receives back a single audio stream from the server, thus decoupling the required uplink/downlink bandwidth of the end-user from the number of participants. However, this second scenario introduces the server as a single point of failure.

10.4 Towards an Inclusive Framework for NMP and Remote Musical Education

Though a number of NMP solutions already exist (see Sec. 2.2.1 for further details), they show several limitations:

- Lack of accessibility for users with visual/auditory/mobility impairments.
- Lack of user-friendliness for subjects without technical background in communication technology (e.g., children): for instance, users may be requested to manually input IP addresses or UDP port numbers to enable multimedia streaming, or to manually configure some settings.
- Lack of portability and deployment flexibility: NMP solutions leverage wired networking infrastructures for the network access segment, due to the unacceptable latency introduced by the medium access control mechanisms of Wi-Fi and 4G technologies. The adoption of 5G connectivity for NMP practices has started being investigated very recently (e.g., [259]) but not yet fully integrated in NMP frameworks.
- Lack of a dedicated backend to support low-latency audio streaming: they often treat latency contributions introduced by the telecommunication network as a "black box", with no effort to minimize it to improve the perceived QoE.

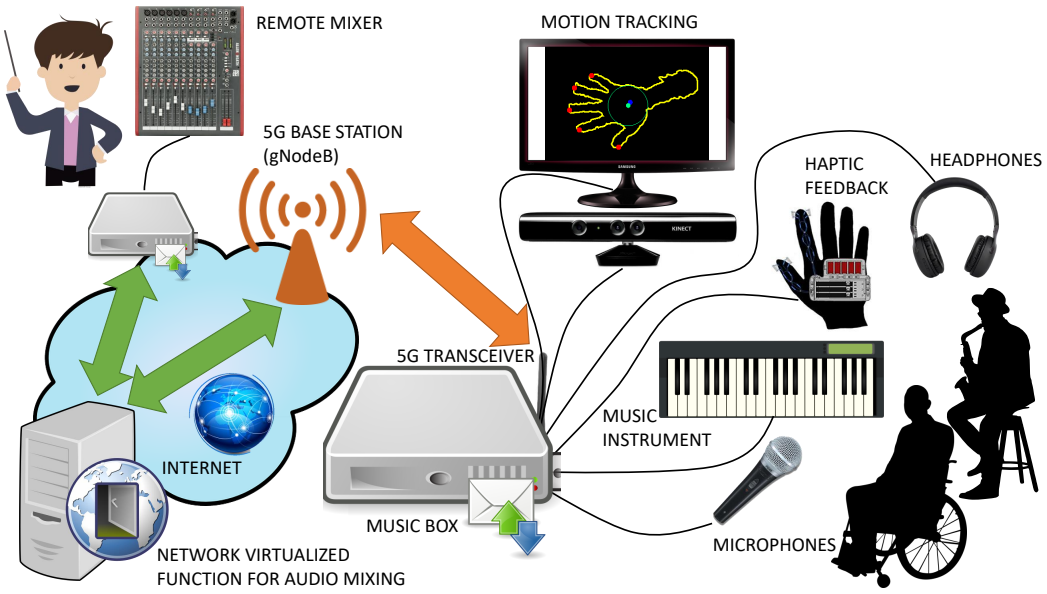


Fig. 10.1 The proposed inclusive NMP framework for remote music education and practice

To address the aforementioned shortcomings, our envisioned inclusive NMP system for remote music teaching and practices (shown in Fig. 10.1) adopts a C/S architectural paradigm. It envisions the possibility of leveraging an easy-to-use dedicated hardware device (referred to as "music box"), with the goal of overcoming the real-time processing limitations of desktop computers and operating systems. The music box can leverage a single-board computer (e.g., a Raspberry Pi, as in [85, 22]), or even customizable hardware (e.g., an FPGA board) to support the technologies described in the following.

10.4.1 Camera Feedback and Motion Sensor Interface Control

Enabling mobility-impaired users to provide a stream of control information to interact with the interfaces of the NMP platform and with remote performers requires wearable sensors or hand-tracking technologies. This represents a unique way for those users to participate in music performance and collaboration, overcoming barriers that prevent their involvement in traditional music-making settings.

User inputs can be captured by sensors tailored to the specific mobility abilities of the subject, such as:

- video-based hand pose reconstruction methods [260], or specialized unobtrusive sensors [261] to reconstruct small-scale hand and finger movements.
- specialized joysticks or buttons that can be actuated with minimal effort and range of motion using a hand or the chin.
- gaze tracking devices [262].
- sip-and-puff devices [263].

This would enable a mobility-impaired user to participate in music-making as a conductor or instructor, by providing a real-time stream of information to an ensemble of performers. The same input methodology can be used to allow the user to act as a performer [264–266], by controlling a synthetic instrument either by directly determining the notes being played, or by shaping some aspects of the sound of an instrument that operates, at least in part, automatically (e.g., a sequencer playing a predetermined melody or an arpeggiator).

Gestural data streams acquired through the previously described motion tracking methods can be rendered at the remote counterparts either via realistic avatars or through stylized, simplified representations that still convey the user's intention. The avatar could, for example, reproduce the pose of a conductor/instructor whose movements match the inputs of the user. In addition to enhancing usability, this brings a significant advantage in terms of bit rate requirements, since gestural data can be encoded much more efficiently in terms of data volume and could achieve lower latencies than video data [267].

10.4.2 Sensors/Actuators for Remote Haptic Signals

Haptic feedback mechanisms (possibly in combination with a touchscreen interface) can be adopted to ease perception for DHH and BVI users. Indeed, visually/auditory-impaired subjects heavily rely on haptic sensations to interact with the surrounding environment [268, 269]. Tactile wearable actuators [270] can also be used in combination with motion tracking systems (see previous section) to convey haptic information regarding the gestures performed by the remote counterpart(s) to BVI/DHH users, e.g., those of a remote instructor or conductor.

10.4.3 Accessible GUI

The GUI of the NMP system should be adherent to the accessibility standards for web applications [271] (see Sec. 2.1.5), possibly enhanced with tactile feedback [272] (e.g., through a touchscreen interface with customizable electro-adhesive characteristics to render various textures in different areas of the screen, such as that described in Chap. 9).

10.4.4 Remote Mixing Capabilities

Sound technicians should be allowed to remotely synchronize and mix the audio tracks received by all the participants to an NMP session, adjust their listening volumes and optionally add audio effects such as equalization and reverb, to create personalized mixes for each musician. This would foster inclusion of subjects with reduced control capabilities of local interfaces due, e.g., to physical impairments, and user with less technical knowledge.

10.4.5 AR/VR Environments and Spatialized Audio Rendering

The integration of spatialized audio rendering in the envisioned NMP framework is fundamental for conveying to the performance the sense of sharing a single acoustical scene [116]. Moreover, the usage of personalized HRTFs can help DHH users to compensate for partial auditory deficiencies, with a moderate increase of bit rate requirements. AR/VR environments can also be coupled with immersive audio rendering to further enhance the perception of being part of a shared musical environment [273].

10.4.6 5G Connectivity

To interconnect the user's front-end to a dedicated backend infrastructure, 5G offers an unprecedented level of flexibility to fulfill service-specific requirements in terms of bitrate, latency, and reliability. 5G (and beyond-5G) wireless transmission technologies promise to guarantee access delay to the backbone telecommunication

infrastructure below 10 ms, and even as low as 1 ms for mission-critical applications [274], which matches the latency requirements of NMP [275].

Additionally, flexibility of NMP deployments would be greatly enhanced by the adoption of 5G technologies. Indeed, the usage of a cabled connection (Ethernet) for LAN connectivity heavily restricts portability of the system. While Wi-Fi would inherently offer greater portability, these wireless connections are typically avoided in NMP setups, as they introduce excessively high jitter [276], making them unsuitable for real-time, synchronized musical collaboration. Jitter can severely disrupt the rhythmic accuracy and overall coherence of a musical performance. The high jitter mandates the use of a larger de-jitter buffer at the receiving end. While this buffer helps to smooth out the inconsistencies in packet arrival, it inevitably increases the overall M2E latency. This increased latency can lead to noticeable timing discrepancies between musicians, making it challenging to play together in sync.

Finally, 5G networks are considered enablers of the so-called "tactile Internet" [274], which entails ultra-low latency transmission of haptic signals. Therefore, 5G/beyond-5G technologies constitute a promising candidate to accommodate the specific requirements of NMP systems. However, to date only a few designs of 5G infrastructures for NMPs have been investigated [277] along with a scarcity of testbed deployments and in-depth statistical analysis on their latency and reliability performances [275]. Usage of 6G technologies is even more embryonal, though speculated by recent studies [278, 279].

10.4.7 Summary

At present, the features described in the above sections have never been jointly designed and implemented in an NMP framework. Their potential usage and expected benefits are summarized in Tab. 10.2.

10.5 Use Cases

In this section, we discuss three illustrative use-cases for the framework proposed in this chapter. The inclusive features described below include both existing im-

Table 10.2 Inclusive Features and Benefits

Feature	Usage	Benefits
Accessible GUI and haptic feedback	Speech/gesture navigation, haptic feedback	Improved autonomy for DVI/mobility-impaired; sensory complement for DHH; haptic score for DVI.
Motion tracking	Gesture acquisition	Local control of virtual instruments; remote gesture conveyance/avatar visualization.
Haptic actuators	Remote auditory/gestural cues	Non-verbal remote interaction; tactile metronome/conductor cues.
Remote mixing control	Remote audio engineers/technicians	Remote assistance for users with limited control or technical skills.
3D audio or AR/VR	Virtual space navigation	3D audio/instrument visualization; customizable AR/VR for user-specific needs.
5G connectivity	Low-latency wireless access	Increased deployment flexibility; overcomes physical accessibility barriers.

plementations and conceptual designs for future development in MEVO, an NMP system currently being developed at Politecnico di Torino and first demonstrated in June 2023 in a distributed concert involving musicians in Turin (Italy) and Wrocław (Poland) [57].

10.5.1 Accessible GUI

An accessible GUI is the first step towards an inclusive NMP system. Though several guidelines regarding how to implement accessible Web GUIs are available (see Sec. 2.1.5), creating a fully accessible GUI requires considering several aspects. First, it should be easily navigable by a broad spectrum of users, including BVI and DHH subjects. To facilitate navigation with assistive technologies such as screen readers, the GUI should be organized in a hierarchical structure that is logically coherent. This allows screen readers to parse and present the content in a well-defined structure, with the right level of importance to the various graphical elements, and allows the user to skip recurring elements, such as navigation elements, to go directly to the main content of a page. Additionally, it must support keyboard-based interactions and provide meaningful descriptions for each element. Moreover, the GUI should be designed considering font sizes and colors contrast, to be distinguishable by subjects with color blindness or Color Vision Deficiency (CVD). Furthermore, it should adopt a terminology and implement/design interactions that ensure that any user can navigate and understand it, independently of their technological and educational

background. All the above-mentioned aspects have been taken into account for the development of the GUI prototype of MEVO.

More specifically, MEVO's GUI allows the user to perform all the necessary actions to configure the system, create virtual rooms where users meet to play together, and control additional functionalities, such as remote mixing. All the GUI components on the web page are organized in a hierarchical structure, and each component was created following the web accessibility guidelines for GUI components[1]. Following those guidelines ensures seamless integration of the GUI with assistive technologies, such as screen readers. We also made sure that color contrast was appropriate[2] for the various font sizes, to ensure readability by subjects with CVD.

The GUI, shown in Fig. 10.2, presents a left sidebar, used for navigating between the various pages. The two main pages are: "Rooms" (to create virtual rooms) and "Settings" (to configure the various system settings). Furthermore, the room creation page allows for creating rooms with either selected participants or with a shareable link that anyone can use to join. The creation of virtual rooms and the connection setup is handled and presented by the platform in a similar way to traditional videoconferencing applications. Moreover, any additional functionality to manipulate the audio stream, such as audio mixing, is presented with a layout that recalls the one adopted by traditional audio editing applications.

Once the users are connected to the same virtual room, each of them can individually customize the audio mixing of the various audio streams received from the other peers. If users are not able (or willing) to perform this action by themselves, it is possible to allow another participant (e.g., an assistant/audio engineer) to remotely control the audio mixing of each user. From a technical standpoint, this is possible thanks to the presence of the central server, which enables the communication between a user and a music box. Thus, the control messages a given box receives are independent of the user who generated them. This allows us to seamlessly handle switching from a local control to a remote control of a music box by another participant in the session, as long as they have the required permissions to do so.

[1] https://www.w3.org/WAI/ARIA/apg/

[2] https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum

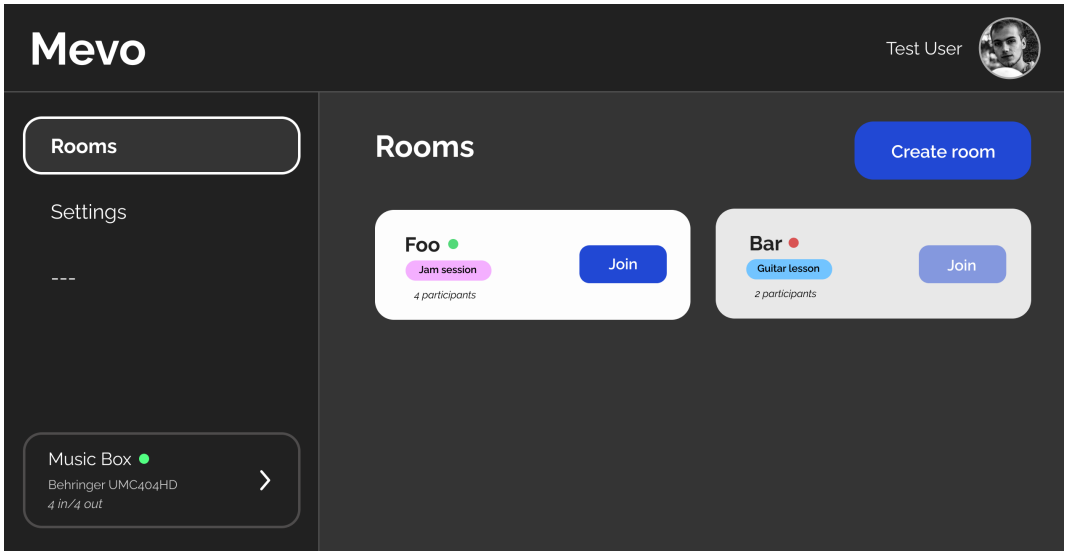


Fig. 10.2 MEVO's Accessible GUI - Rooms page

10.5.2 Accessible Metronome

Metronomes can be easily integrated in NMP systems, as demonstrated in [280], even including adaptive capabilities to time-varying latency conditions. In turn, audio cues generated by the metronome can be complemented with visual and tactile feedback to offer a multisensory experience that caters to a broad spectrum of users, including BVI and DHH subjects.

More in detail, our envisioned prototype of an accessible metronome is designed to operate in a multi-user scenario where each user is provided with a software component running on a PC, that interacts with a wearable device (a bracelet in our case). Such a system can offer three distinct methods for delivering metronome cues:

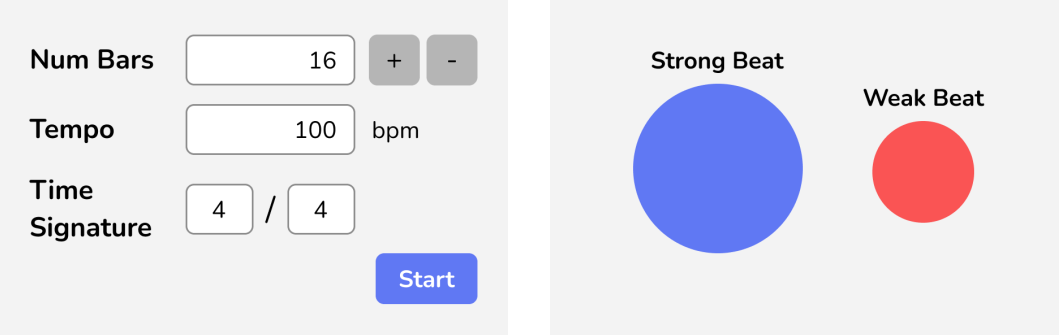
- *Graphical*: a user-friendly GUI that facilitates the interaction and provides a clear visualization of the metronome's beats in real-time for DHH users.
- *Tactile*: a bracelet equipped with vibrating motors activated synchronously with the metronome beats for BVI or DHH users.
- *Auditory*: both the bracelet (through buzzers), and the PC (through speakers/-headset) can emit sounds associated to metronome clicks.

For this envisioned prototype, two different GUIs are needed. One that acts as an orchestrator and is meant to be run by only one user at a time to set up the settings

shared among all the participants in the session. The available settings could be expressed in forms of single values (or lists of values), each describing a group of adjacent bars with common characteristics. The possible settings are:

- *Number of bars*: it indicates the number of bars in each group (leaving this field empty configures the metronome to continue operating indefinitely).
- *Bpm per bar*: it indicates the beats per minute value for bars belonging to each group.
- *Tempo signature per bar*: it indicates the numerator of the tempo signature (i.e., the number of beats per bar) for each group.

Conversely, a second GUI should display two circles, each one mapping respectively a strong and a weak beat. E.g., a blue circle could indicate the strong beats of a bar, while a red circle could be used for the weak beats. This visual representation aims to assist users in identifying the timing of the metronome's beats and the bar subdivision. A simplified example of how the two GUIs could appear is shown in Fig. 10.3, with Fig. 10.3a showing the GUI used to configure the metronome and Fig. 10.3b showing the GUI used to visualize the strong and weak beats of the metronome.



(a) GUI for configuring the metronome

(b) GUI for the metronome visualization

Fig. 10.3 Metronome GUI example

The custom hardware component of the metronome is envisioned to be a bracelet, which utilizes a microcontroller with wireless communication capabilities (e.g., an Arduino Nano with ESP32), to control two vibrating motors and two buzzers, with two potentiometers to independently control their volume. This configuration

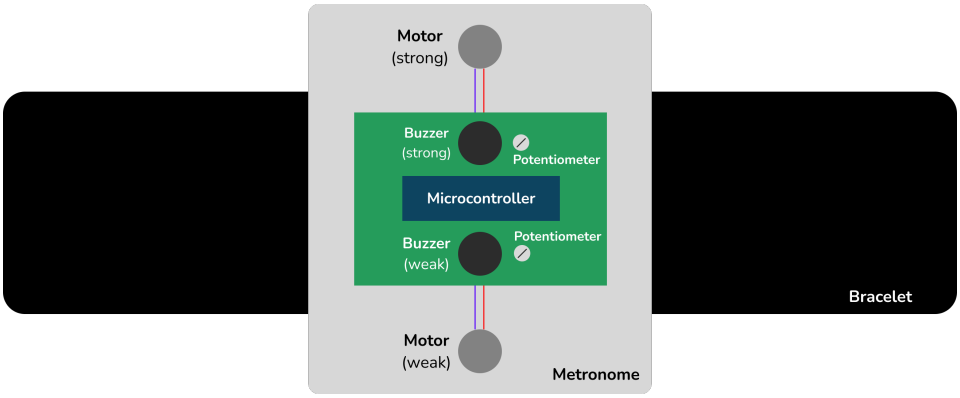


Fig. 10.4 Example of the layout of the custom HW metronome bracelet

allows one motor and one buzzer to be associated with strong beats of a measure, in accordance with the chosen time signature, while the second motor and buzzer can be assigned to signal the remaining weak beats of the measure. The software not only has to take care of preserving the synchronization between the graphical beats and the bracelet through periodic feedback from the latter, but also needs to exchange real-time data with the participants to the online music session with the aim of preserving synchronization with all the participants' devices, minimizing the effects of network jitter and clock drifts. A simplified example of the bracelet layout is provided in Fig. 10.4.

10.5.3 Tactile Feedback for Timbre Discrimination

MEVO is envisioned to also integrate a visual representation of pre-recorded audio samples enhanced with tactile feedback, with the aim of facilitating discrimination of timbre and sonic grain by DHH users. As thoroughly described in Chap. 9, audio excerpts generated by different instruments are associated with:

- a visual representation of the instrument.
- a background image and a background color, which should be evocative of the emotions experienced by the listener.
- a texture pattern superimposed on the background, to render the tactile sensation associated with the background image.

Cross-modal associations among images, colors, sounds, and textures should thus be exploited to offer a multisensorial perceptual experience to the users.

10.6 Technical Challenges

Based on the technology overview provided in Sec. 10.2, it emerges that accessible web-mediated musical interactions in real-time are a rather unexplored area, with only a few studies investigating collaborative music making [281], mainly due to the extremely low latency requirements necessary to maintain a synchronous musical interplay [27]. Therefore, how to successfully incorporate such inclusive technological solutions within NMP frameworks for remote music teaching and practices is a yet almost completely unexplored research direction, which urges for dedicated efforts by the scientific community. A roadmap to guide such efforts is provided in the remainder of this section, which reviews the most significant technical challenges involved in the realization of an inclusive NMP system, discussing open research directions.

10.6.1 Latency Management and Packet Loss Concealment for Cross-Modal Data Streams

When heterogeneous data types (e.g., audio, gestural and haptic signals) need to be jointly transmitted (i.e., they cannot be processed and streamed separately due to low-latency and high-reliability application requirements), cross-modal transmission strategies must be adopted [282].

After acquisition and digitization, data that need to be streamed over a network are packetized before being transmitted, to adhere to the packet-oriented switching paradigm adopted in telecommunication networks. Then, every packet is routed individually through the telecommunication network infrastructure to be conveyed to its intended destination. Consecutive packets may experience different delays along their end-to-end route, since some latency components (e.g., the queueing delays introduced by routers traversed by the packet along its path) may vary over time. Variations of the inter-arrival times between consecutive packets (“jitter”) and packet losses due to transmission errors or drops (actuated by routers to prevent excessive

traffic congestion) may cause data portions to arrive too late to be reproduced in due time at the receiver side. In turn, this generates glitches (in the case of audio streams), deterioration of the rendered gestures (for motion data) or of the perceived haptic feedback (for tactile data). For a deeper overview of this topic, refer to Sec. 1.1, Sec. 2.1.1 and Chap. 6.

State-of-the-art packet retransmission techniques, such as those implemented by the TCP protocol [283], cannot be exploited by NMP applications, as they would further increase the M2E latency. Therefore, PLC techniques specifically tailored for cross-modal streams should be devised [284, 285], while ensuring compatibility with the low-latency requirements of NMP. Additionally, dedicated solutions must be developed to effectively handle cross-modal coding, transmission and latency when synchronizing audio, visual, and haptic data streams in real-time. A recent survey on such techniques can be found in [286].

Focusing on audio streaming, a further cause of artifacts is the "drifting" effect caused by the imperfect alignment of local clock oscillators [46]. Indeed, even when the clocks' nominal frequency is identical, small deviations may still occur due, e.g., to temperature variations or to fabrication imperfections. This leads to buffer over/underruns caused by the difference between the amount of audio samples generated by the sender and the amount of those reproduced by the receiver during a reference time window [287, 288]. Eventually, such differences cause either occasional drops of audio blocks or generate gaps in the audio stream. This side effect further motivates the need for adequate PLC mechanisms or for dedicated clock synchronization approaches (see [287, 288]).

10.6.2 Interoperability and Standardization

The need for interoperability among the heterogeneous components that are envisioned to be part of an inclusive NMP system has already emerged in [289], which addresses IoMusT environments. The lack of standardization of interfaces and communication protocols further hinders from seamless integration of Musical Things in NMP frameworks. Though some preliminary steps towards standardization for IoMusT have recently been done [290], the specific use-case of NMP has been neglected so far.

10.6.3 Minimizing Network Contributions to Mouth-to-Ear Latency

The overall M2E latency experienced by the users of NMP systems includes multiple contributions introduced by different stages of the audio acquisition, processing, transmission and reproduction chain. While acquisition, processing, buffering and playout delays can be controlled by NMP hardware/software designers, the latency component introduced by the network infrastructure is typically unknown and difficult to estimate, since it is influenced by several factors such as the physical distance among performers, the network traffic congestion level (which exhibits variations over time), the traffic priority policies applied by network operators and ISPs, etc.

The exploitation of Software Defined Networking (SDN) [291], Network Function Virtualization (NFV) [292], and Multi-Access Edge Computing (MEC) [293] technologies is envisioned as a game-changer to ensure compliance to the Quality of Service/Experience (QoS/QoE) requirements of NMP systems. Such technologies should be incorporated in a backend infrastructure dedicated to NMP applications, which should include tailored functionalities capable of dynamically adapting the routing of audio streams and the positioning of NVFs hosting NMP servers by jointly considering:

- the physical location of the involved users and of the candidate sites for NVFs placement, possibly exploiting MEC to ensure user proximity.
- the current network congestion level.
- predictions on the future evolution of traffic patterns.

Moreover, the involvement of ISPs and network operators in the design of commercial NMP services would enable the adoption of prioritization criteria for NMP-generated data (e.g., with a dedicated SDN slice), thus offering adequate QoS guarantees to the users in terms of latency and jitter.

To implement such functionalities, the backend portion of the NMP system should also include a Service Management and Orchestration (SMO) engine, responsible for instantiating NVFs hosting NMP servers within MEC nodes and to coherently allocate resources in the involved network segments. This would enable

latency optimization by locating NVFs in strategic positions within the network infrastructure, depending on the users' physical location, bit rate requirements, and on the traffic congestion level along the paths interconnecting users to servers. The SMO should also collect monitoring information from the users currently participating in the musical session (e.g., experienced latency and packet losses) to dynamically update the routing of audio flows as traffic conditions evolve. To this aim, the SMO could also exploit traffic prediction methods [294, 295] operating at various time scales. In turn, dedicated optimization algorithms implemented at the SMO premises will permit to reduce network latency by exploiting the outputs of such traffic predictors, which will enable proactive operations and dynamic adjustments of NVFs' locations depending on the current and forecasted network congestion conditions, thus ultimately improving the QoS/QoE perceived by the users.

Though a detailed discussion on such topics is out of the scope of this chapter, the interested reader is referred to [240] for a proposal of an NMP infrastructure supporting immersive audio rendering, which leverages NFVs and MEC technologies. The adoption of SDN technologies for NMP systems has been first envisioned in [296], which provides examples of possible interactions between a real-time network latency monitoring module and an NMP system. Despite such preliminary attempts, the potential of SDN integration in NMP contexts remains largely unexpressed.

10.7 Towards an Inclusive Musical Metaverse

As of today, the concept of MM starts to be explored in the research community. According to [179], the MM is "an interoperable persistent network of multiuser environments merging physical reality with digital reality, which serve a musical purpose. It is based on the convergence of Musical XR [218] and IoMusT technologies [30] that enable multisensory, networked musical interactions between musical stakeholders, as well as between such stakeholders and Musical XR environments and objects." The range of musical activities that could be performed in the MM encompasses composition, fruition of virtual performances, teaching, and collaborative content production.

In addition to IoMusT and Musical XR, the MM also builds upon a large pool of emerging technological advancements [179], including:

- *Blockchain* and *Non-Fungible Tokens* (NFTs): to control, regulate and ensure transparency in copyright ownership of music-related material, as well as in royalties distribution.
- *Digital twins*: to create virtual musical entities replicating physical ones, such as music instruments, acoustic scenes etc.
- *AI and ML*: to enhance user-experience by improving proactivity and context-awareness of musical services. To this aim, ML-based predictive algorithms leveraging user-gathered sensing data (e.g., musical gestures), can be exploited, as well as recognition/classification methods to categorize musical content.
- *5G-and-beyond communication infrastructure*: to ensure seamless fruition of the MM by adhering to strict latency, bit rate and reliability requirements while providing widespread access and supporting user mobility.

Among the opportunities that are envisioned to be unlocked by a widespread adoption of the MM, it is worth mentioning [179]:

- the growth of unprecedented forms of musical expressions and activities, entailing novel ways of engagement of the audience and fostering collaborative creation among performers.
- the constitution of new social communities and of novel music-based social interactions, with their management and governmental rules.
- novel monetization opportunities for the music industry, related to music streaming and distribution in the MM.

It is also envisioned that inclusivity will be one important characteristic of the MM, which will pave the way for novel opportunities for disabled musicians and audiences to access musical experiences [179]. For example, XR-based musical instruments could ease control and manipulation for visual/mobility-impaired users. Similarly, fruition and participation in music events in the MM could be extended to disabled subjects for whom physical participation in in-presence performances is hindered or precluded.

We believe that inclusive NMP systems will constitute a pivotal building block for the realization of a disability-friendly MM, by providing the technological substrate

upon which such a virtual world will be built. Indeed, based on the MM framework proposed in [179], NMP systems will act as middleman to bridge the physical layer (constituted by users, Musical Things, etc.) and the link layer, which encompasses the networking and communication substrate, as well storage and computational nodes in the cloud. However, current NMP systems will need to be significantly evolved and adapted for seamless incorporation in MM ecosystems, thus calling for dedicated research efforts by the scientific community. In particular, 5G/beyond-5G communication technologies combined with MEC functionalities are envisioned for future integration in NMP infrastructures, as they are capable of supporting real-time streaming of high volumes of cross-modal data, thus finally enabling VR delivery for fully immersive experiences.

10.8 Summary

This chapter offered a thorough overview of the educational and performative needs of visually/auditory/mobility-impaired users in the context of remote music education and online musical practice. Technological solutions that could address such needs, which include spatialized audio rendering, music visualization, haptic feedback and motion tracking techniques. Moreover, the chapter envisioned the integration of such technologies into a proposed NMP framework to enhance its inclusiveness, highlighting the technical challenges to be tackled and discussing future research directions towards the realization of an inclusive musical metaverse.

Chapter 11

Conclusions

This thesis has explored several aspects related to improving the overall experience and inclusivity of NMP systems. It has investigated the development of web-based solutions for NMP, including both standalone and hybrid web-native applications, alongside the development of a dedicated hardware system designed for ultra-low latency audio. Furthermore, it has focused on enhancing the audio quality of NMP systems through the study and implementation of a PLC method based on AR models. Beyond audio, it has explored innovative approaches to visual feedback in NMP, such as a VR-based system for transmitting conductor gestures to a remote orchestra, and has addressed the critical area of accessibility in music education, proposing and investigating interfaces tailored for DHH students, and ultimately providing comprehensive overview of accessibility considerations and a vision for an inclusive framework in remote music learning and performance. This concluding chapter synthesizes the key findings and contributions made across these diverse areas of research. It provides an overview of the main outcomes, and finally, outlines promising future research directions.

11.1 Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution

Chapter 3 investigated the feasibility of implementing an NMP system using only web technologies, specifically addressing the limitations of standard WebRTC

`MediaStreams` for high-fidelity, low-latency audio streaming. The work involved developing a web application that utilizes the WebRTC `RTCDataChannel` for transmitting uncompressed 16-bit PCM stereo audio data in a P2P fashion. This approach leveraged the Web Audio API and its `AudioWorklet` element for custom, low-latency audio processing in an independent real-time thread. The chapter detailed the architecture of the application, describing key improvements made to the preliminary version [135], including the adoption of `SharedArrayBuffer` for more efficient data exchange between the main thread and the audio thread, as well as scalability improvements achieved by removing the linear dependency between the number of `AudioWorklets` and the number of connected peers. The performance of the proposed solution was assessed across various web browsers, operating systems, and network configurations, and it was benchmarked against the traditional WebRTC `MediaStream` architecture and other popular web-based videoconferencing platforms like Jitsi and Google Meet.

The results of the investigation demonstrated that the proposed approach using WebRTC `RTCDataChannel` and `AudioWorklets` outperforms the standard WebRTC `MediaStream` in terms of audio quality and latency. Specifically, the best achieved one-way latency was 40 ms obtained between two MacBooks on a LAN, with the buffering strategy contributing approximately 20 ms to this total. In contrast, videoconferencing applications tested on the same devices showed latencies ranging from 90 ms to 160 ms. Furthermore, the application showcased a considerable improvement in scalability w.r.t. the preliminary version, allowing it to support tens of connected peers, limited primarily by the available bandwidth and CPU resources. Lastly, this work also highlighted that the main source of latency in a web-based NMP system is introduced by the audio input/output system, thus advocating for further advancements or integrations of web browsers with the low-latency audio drivers provided by the various OSes.

The work presented in this chapter is part of a contribution published in the Journal of the Audio Engineering Society in November 2022 [49].

11.2 Networked Music Performance: A Unified Native and Web Approach

Chapter 4 detailed the development and evaluation of an NMP system providing both a web-based client and a native client, controllable through a unified web-based GUI. The work aimed to bridge the gap between the high performance of traditional native NMP software and the ease of use of traditional web-based videoconferencing tools. Moreover, this approach aimed to provide a low-latency audio streaming alternative, still controlled through a web-based GUI, for those scenarios requiring a strict low-latency musical interaction, to overcome the limitations of web-based solutions, shown in Chap. 3. The chapter described the architecture of the system, which includes a server for signaling and session management, and the two client implementations: a web-based client (using the same approach described in Chap. 3), and a native client based on a Raspberry Pi 4B.

A part of the work involved a comprehensive performance evaluation. Subjective audio quality tests were conducted in an anechoic chamber with 26 participants using an ABX methodology to compare the proposed system's uncompressed audio with the audio of commonly used videoconferencing solutions (Zoom, Jitsi, and Google Meet) across various musical stimuli (violin, cello, soprano singing). Additionally, objective M2E latency measurements were performed for both the web client, across different operating systems (macOS, Windows, Ubuntu) and web browsers (Firefox, Chrome) in comparison to other web-based solutions, as well as for the native client, which was compared to other native NMP solutions like SonoBus and JackTrip.

The subjective audio quality tests revealed that the audio transmitted by the proposed system was practically indistinguishable from the original uncompressed audio for all tested musical excerpts. In contrast, the audio from Google Meet showed statistically significant differences across all stimuli. Zoom and Jitsi also exhibited significant differences for certain instruments or vocal types. This indicates that the proposed solution offers superior audio quality compared to the tested videoconferencing platforms.

The M2E latency analysis indicated that the native client, running on a Raspberry Pi 4B, achieved latency on par with or even slightly better than other established native NMP applications, i.e., SonoBus and JackTrip. Instead, the web client on

macOS achieved a latency around 40 ms with Firefox and 60 ms with Chrome. On Windows and Ubuntu, it achieved higher latencies, ranging from 100 ms to 150 ms.

While challenges remain in achieving ultra-low latency in web-based environments comparable to native solutions, the proposed system successfully provides a user-friendly web interface for high-quality uncompressed audio transmission with reduced latency compared to existing web-based options, making it suitable for applications like music education. This hybrid approach also offers an alternative for professional musicians, who can rely on the dedicated native client to achieve lower latency figures, allowing them to play together in real-time.

The work reported in this chapter is based on two contributions: one presented in the XXIII CIM [48], and the other presented in the 10th Convention of the European Acoustics Association (Forum Acusticum) [50].

11.3 FPGA-based Low-Latency Audio Coprocessor for Networked Music Performance

Chapter 5 presented the ongoing development of a dedicated hardware system for NMP aimed at achieving extremely low-latency audio acquisition and processing. The core of the work was the design and implementation of an Application-Specific Instruction set Processor (ASIP) on an FPGA. This ASIP utilized a Transport Triggered Architecture (TTA) optimized for minimal-latency audio processing. The system comprised a custom audio board for audio input/output, the FPGA-based audio processor, and a Raspberry Pi 4B for network communication. This chapter described the architecture of the TTA processor, including its functional units (ALU, GCU, LSU, register files, timer, and custom units for I2S, MIDI, SPI, mixing, and reverb), and the communication protocols used between the different hardware components (I2S, MIDI, SPI). Moreover, it also described the firmware developed for the FPGA processor, outlining its role in routing audio and MIDI data between the audio board and the Raspberry Pi via SPI, with considerations for different mixing modes (Raspberry Pi vs. FPGA). Finally, it presented a theoretical analysis of the SPI bandwidth and the overhead introduced by the custom communication protocol between the FPGA and the Raspberry Pi.

The preliminary results on the resource usage of the FPGA-based audio coprocessor showed that the implemented TTA processor used approximately 21% of the available logic elements and 8% of the DSP cells. The power consumption of the FPGA-based processor was estimated to be around 350 mW, suggesting good power efficiency. Theoretical calculations regarding the SPI bandwidth between the FPGA and the Raspberry Pi indicated the potential to stream a significant number of audio sources.

The work presented in this chapter is part of a contribution presented in the 4th International Symposium on the Internet of Sounds in October 2023 [51].

11.4 Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications

Chapter 6 investigated the use of AR models as a PLC method for NMP applications. The study involved a comparative investigation of AR models against traditional NMP PLC techniques like silence substitution and pattern replication, using both quantitative measurements and subjective listening tests to assess the potential improvements in audio quality. Furthermore, the research explored into the algorithmic implementation of AR models using Burg's method, and aimed to identify the optimal parameter configurations that provide the best balance between prediction accuracy and computational time, suitable for real-time operation on a Raspberry Pi 4B within the strict latency constraints of NMP. The chapter analyzed the computational time required for model fitting and prediction across six different algorithm implementations and with various compiler optimization flags to determine the most efficient approaches for the target platform.

The results of the research confirmed that AR models, even with relatively low orders and a short history of past audio samples, can predict future audio samples with greater accuracy than traditional PLC solutions commonly used in NMP. The study also demonstrated the feasibility of running AR models in real-time on a device like the Raspberry Pi 4B across a wide range of configurations. These findings provide system designers with the flexibility to choose between prioritizing higher audio quality, which may come with a higher computational cost, or favoring

scalability by reducing computational costs, potentially at the expense of a slight reduction in audio quality. These results suggest that AR models offer a promising and adaptable PLC solution for NMP applications, which can be used as a baseline for more advanced PLC methods.

The work presented in this chapter is part of two contributions: the first one presented at the 17th International Audio Mostly Conference in October 2022 [47], the second one, consisting in an extended version of the previous work, published in the Personal and Ubiquitous Computing Journal in May 2024 [53].

11.5 Introducing DUST: a Dataset of real-time UDP Sound packet Traces

Chapter 7 focused on the presentation of DUST, a novel dataset of UDP audio packet traces specifically generated by a real-time NMP system called MEVO [57]. The work involved collecting various traces under different network and audio device configurations, and different physical locations to capture the variability inherent to real-world scenarios. A key aspect of the dataset design was the inclusion of local playback status information alongside the traces of incoming audio packets at the receiver. This was done to enable a more comprehensive overview of the challenges faced by real-time audio streaming systems. The dataset was designed to support offline simulations of real-time audio streaming systems, with the specific aim of aiding the development and evaluation of algorithms that address critical problems in NMP systems, such as PLC, resilience to network jitter, and compensation for clock drift between the sender and receiver devices. Furthermore, the dataset was designed to allow the simulation of scenarios where late packets might still contain usable audio data, enabling a more nuanced evaluation of PLC techniques compared to simply treating all late packets as lost.

The work presented in this chapter was part of a contribution to the 5th International Symposium on the Internet of Sounds [54], held in September 2024.

11.6 Remote Orchestral Conduction via a Virtual Reality System

Chapter 8 introduced a VR-based system as an alternative to traditional video streaming to provide visual feedback from a conductor to remote musicians, in an NMP setting. The work involved the use of a VR headset to track the conductor's gestures, specifically hand movements and head orientation. The captured motion data is then transmitted over a network to the remote musicians, where it is displayed in a web browser through an avatar representation of the conductor within a virtual environment. This approach aimed to address the limitations of video streaming in NMP, such as high bandwidth requirements and additional encoding/decoding latencies, and provided personalized visualizations for the conductor and each musician, allowing the musicians to easily understand if and when the conductor was pointing at them. The study included quantitative measurements of the M2P latency of the system. Additionally, the research involved an analysis of the bitrate required by the transmission of motion data.

The quantitative results indicated that the experimental VR-based system achieves an average M2P latency of approximately 140 ms, with the maximum latency not exceeding 210 ms. Notably, these latency values lie in the same order of magnitude as the human visual reaction time, which is reported to be around 600 ms. Moreover, conductors usually anticipate their gestural cues w.r.t. the desired timing of the musicians' reaction, to leave them enough time to process and react to the visual hints. Thus, conductors should be able to compensate for the M2P delay introduced by the proposed system. In addition to latency performance, the proposed VR-based approach demonstrated a reduction in the required transmission bitrate, achieving requirements that are at least one order of magnitude lower than those of state-of-the-art video streaming.

The content of this chapter is part of a contribution presented in the 5th International Symposium on the Internet of Sounds [55] in September 2024.

11.7 Collection of Design Directions for a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students

Chapter 9 detailed the process of collecting design directions for a visual interface with electro-adhesive haptic feedback, with the goal of facilitating the understanding of the concept of "sonic grain" for DHH music students. The work involved conducting a survey campaign targeting 56 respondents to gather information on cross-modal associations between musical instruments and visual (images, colors, shapes) and tactile (textures) stimuli, inspired by listening to musical excerpts. This survey aimed to empirically establish a set of associations that could be leveraged in the design of the interface.

These results were then directly used in the realization of a prototype interface. The prototype demonstrated the potential of utilizing cross-modal associations to represent different types of feedbacks through images, colors, and textures, offering a multisensorial experience. This prototype represents a promising initial step towards developing more inclusive music education tools for DHH students by integrating visual and haptic feedback.

The content of this chapter is part of a contribution presented in the 4th International Symposium on the Internet of Sounds in October 2023 [52].

11.8 Towards an Inclusive Framework for Remote Musical Education and Practices

Chapter 10 addressed the crucial topic of inclusivity in remote music education and NMP, focusing on the challenges faced by users with visual, auditory, or motor impairments. This chapter presented a comprehensive overview of the educational and performative needs of visually, auditory, and mobility-impaired users in remote music education and online musical practice. Then, it provided a classification and analysis of technological solutions, such as spatialized audio rendering, music visualization, haptic feedback, and motion tracking techniques, that can be used to enhance accessibility in NMP and remote musical education. These technologies

are envisioned to be integrated into a proposed NMP framework to create a more inclusive environment. Furthermore, it provided illustrative use cases demonstrating the potential benefits for various categories of users with disabilities. Finally, it identified key technical challenges that need to be tackled for the practical realization of such an inclusive framework and discussed future research directions, specifically towards its incorporation within the Musical Metaverse.

The work presented in this chapter is part of a contribution published in the IEEE Access Journal in November 2024 [56].

11.9 Conclusive Remarks

The research presented in this work has explored various facets of enhancing NMP applications. Chap. 3 demonstrated the feasibility of leveraging web technologies, specifically WebRTC `RTCDataChannels` and `AudioWorklets`, to create a web-based NMP system. Building upon this, Chap. 4 detailed the development of a hybrid NMP system, offering both a user-friendly web client and a high-performance native client, providing flexibility for different user needs and technical capabilities, and overcoming latency limitations of web only solutions.

To further minimize latency and offload audio processing from general-purpose computers, Chap. 5 presented the ongoing development of a dedicated FPGA-based audio coprocessor. Preliminary results indicated promising resource utilization, power efficiency, and the potential for achieving ultra-low latency audio processing and input/output. Addressing the inherent challenges of packet loss in network communication, Chap. 6 investigated the use of AR models for PLC, demonstrating their superior prediction accuracy compared to traditional methods and the flexibility in their deployment.

To support further research and development in the field, Chap. 7 introduced DUST, a novel dataset of real-time UDP audio packet traces collected through an NMP system. This dataset aims to enable realistic offline simulations to address issues like packet loss, jitter, and clock drift. Expanding beyond audio, Chap. 8 explored the use of a VR headset for remote orchestral conduction, offering a lower bandwidth alternative to video, with tolerable latency for visual cues.

Finally, this thesis addressed the critical aspect of inclusivity in remote music education and NMP. Chap. 9 detailed the initial steps in designing a visual and haptic interface to convey the concept of "sonic grain" to DHH music students, through the collection of cross-modal associations between visual and tactile stimuli. Finally, Chap. 10 provided a comprehensive overview of the needs of users with various disabilities in remote music contexts, analyzed enabling technologies, and proposed a vision for an inclusive NMP framework.

Collectively, this thesis has contributed to the advancement of more robust, user-friendly, and inclusive NMP applications, paving the way for richer and more accessible remote musical interactions.

11.10 Future Works

Building upon the research presented in this thesis, several directions for future investigation and development can be identified. In the realm of web-based NMP systems, future efforts should focus on finding effective strategies to minimize the browser latency in audio input/output, which remains a bottleneck for real-time musical interaction. Simultaneously, advocating for the inclusion of uncompressed audio support and greater customization options within the WebRTC RTCPeerConnection API would significantly enhance the capabilities of web-based NMP solutions. Moreover, another important future work is to investigate and implement crucial functionalities such as dynamic de-jitter buffering strategies, drift correction algorithms, and PLC algorithms specifically designed to operate within the web environment constraints. The successful integration of these elements would be essential in turning the current prototype into a practical NMP application.

The ongoing development of the FPGA-based hardware system for ultra-low latency NMP will continue to explore further optimizations and refinements in the hardware design. This includes enhancing its efficiency in terms of resource utilization and power consumption and increasing its versatility to support a wider array of audio processing tasks, e.g., by implementing other audio effects. To ensure the practical relevance and usability of both the hybrid NMP system and the FPGA-based solution, extensive user testing will be a crucial next step. This will involve gathering feedback from musicians in realistic performance scenarios to validate the practical utility of the developed hybrid system, assess the effectiveness of the

functionalities implemented in both clients, and rigorously evaluate the end-to-end latency performance of the FPGA-based solution in real-world conditions.

Regarding PLC, future work could explore the actual impact of the proposed AR model-based solution in a real-performance setting. This would involve implementing the solution within a complete NMP system and conducting extensive user testing with musicians to gather subjective feedback on its effectiveness in concealing packet losses without introducing noticeable artifacts. Furthermore, exploring various deployment options for the proposed solution is also envisioned. Investigation into alternative or complementary PLC implementations and strategies, such as Forward Error Correction (FEC) mechanisms or a dynamic PLC method that adapts its behavior based on the characteristics of the received audio signal, could also yield significant improvements towards finding a widely agreed-upon PLC solution for NMP systems. Moreover, another area of exploration could be in terms of smoothing out discontinuities on the right side of a prediction, by investigating more advanced techniques than a simple cross-fade.

The VR-based system for remote orchestral conduction opens several future directions. These include conducting comprehensive subjective tests with conductors and musicians to evaluate its usability, intuitiveness, and the impact of the measured latency on the perceived QoE of the musical interaction. Integrating facial tracking to convey a wider range of conductor expressions, including intentions and emotions, is another planned development that could significantly enhance the sense of presence and communication. Furthermore, another important aspect would be to explore the use of haptic feedback mechanisms, to effectively convey the conductor's gestures to visually impaired musicians, opening up new possibilities for inclusive orchestral participation. Another promising future work could be the investigation of alternative and potentially more precise ways to capture and transmit the conductor's gestures, perhaps through the use of sensors applied to a traditional conductor's baton, effectively creating a "smart baton", or by utilizing sensors integrated into a wearable device. These alternative tracking methods could potentially help to reduce the intrinsic tracking latency associated with VR headsets, and could also be leveraged to provide more direct and nuanced forms of haptic feedback to visually impaired orchestra musicians.

In the domain of inclusive music education, future efforts will concentrate on finalizing the developed visual and haptic interface to convey the concept of "sonic

grain" to DHH students. This includes integrating robust functionalities for the cross-modal display of audio tracks, whether acquired in real-time from a musical instrument or retrieved on-demand from a remote repository of musical samples. Moreover, future works could try exploring dynamic visual and haptic representations synchronized with the audio stream, and experimenting with remote interaction capabilities by integrating the prototype within an NMP framework for remote music teaching. More broadly, future research will continue to address the challenges of inclusivity in remote music education and NMP for users with visual, auditory, or motor impairments. This involves developing accessible human-machine interfaces tailored to musical performances, designing robust backend infrastructure for large-scale, low-latency audio mixing, and ultimately contributing towards the realization of a truly inclusive Musical Metaverse.

References

[1] John Lazzaro and John Wawrzynek. A case for network musical performance. In *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '01, pages 157–166, New York, NY, USA, January 2001. Association for Computing Machinery.

[2] Alain B Renaud, Alexander Carôt, and Pedro Rebelo. Networked music performance: State of the art. In *AES 30th International Conference*, March 2007.

[3] James Pritchett. *The Music of John Cage*. Music in the Twentieth Century. Cambridge University Press, Cambridge [England] ; New York, 1993.

[4] John Bischoff, Rich Gold, and Jim Horton. Music for an Interactive Network of Microcomputers. *Computer Music Journal*, 2(3):24, December 1978.

[5] The MIDI Association (TMA). MIDI Specifications. https://midi.org/specs, December 2023.

[6] Scot Gresham-Lancaster. The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music. *Leonardo Music Journal*, 8:39, 1998.

[7] Chris Chafe, Scott Wilson, Randal Leistikow, Dave Chisholm, and Gary Scavone. A simplified approach to high quality music and sound over IP. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, December 2000.

[8] Juan-Pablo Cáceres and Chris Chafe. JackTrip: Under the Hood of an Engine for Network Audio. *Journal of New Music Research*, 39(3):183–187, September 2010.

[9] Alexander Carôt, Alain B. Renaud, and Bruno Verbrugghe. Network music performance (NMP) with soundjack. In *International Conference on New Interfaces for Musical Expression*, Paris, France, June 2006.

[10] Alexander Carot and Christian Werner. Distributed Network Music Workshop with Soundjack. In *Proceedings of the 25th Tonmeistertagung*, Leipzig, German, September 2008.

[11] Volker Fischer. Case Study: Performing Band Rehearsals on the Internet With Jamulus, January 2015.

[12] Chrisoula Alexandraki, Panayotis Koutlemanis, Petros Gasteratos, Nikolas Valsamakis, Demosthenes Akoumianakis, Giannis Milolidakis, G Vellis, and D Kotsalis. Towards the implementation of a generic platform for networked music performance: The DIAMOUSES approach. In *ICMC*, pages 251–258, August 2008.

[13] McGill Ultra-Videoconferencing Research Group. Ultra-videoconferencing Overview. https://srl.mcgill.ca/projects/rtnm/.

[14] Carlo Drioli, Claudio Allocchio, and Nicola Buso. Networked Performances and Natural Interaction via LOLA: Low Latency High Quality A/V Streaming System. In *Information Technologies for Performing Arts, Media Access, and Entertainment*, volume 7990, pages 240–250. Springer Berlin Heidelberg, Berlin, Heidelberg, January 2013.

[15] WHO. Archived: Who timeline - covid-19. https://www.who.int/news/item/27-04-2020-who-timeline—covid-19.

[16] Yoong Chong Yew. The Issues and Challenges of Conducting Chamber Music Classes Online: A Case Study, March 2020.

[17] Ian L Howell, Kayla Jane Gautereaux, Joshua Glasner, Nicholas Perna, Chadley Ballantyne, and Theodora Nestorova. Preliminary Report: Comparing The Audio Quality Of Classical Music Lessons Over Zoom, Microsoft Teams, Voicelessonsapp, And Apple Facetime, March 2020.

[18] Rachel Carlson and Scot Hanna-Weir. Conducting During COVID. *The Choral Journal*, 61(9), April 2021.

[19] Konstantinos Tsioutas and George Xylomenos. Audio Delay in Web Conference Tools. In *Web Audio Conference 2022 (WAC 2022)*. Zenodo, June 2022.

[20] Jamulus. https://jamulus.io/.

[21] Rebekah Wilson. Aesthetic and technical strategies for networked music performance. *AI & SOCIETY*, 38(5):1871–1884, October 2023.

[22] Christian Hoene, Ian Howell, and Alexander Carôt. Networked Music Performance: Developing Soundjack and the Fastmusic Box During the Coronavirus Pandemic. *Journal of the Audio Engineering Society*, 26, July 2021.

[23] Patrick Müller, Roman Haefeli, Johannes Schütt, and Matthias Ziegler. Telemersive Audio Systems for Online Jamming. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–15, September 2024.

[24] Kelsey E. Onderdijk, Freya Acar, and Edith Van Dyck. Impact of Lockdown Measures on Joint Music Making: Playing Online and Physically Together. *Frontiers in Psychology*, 12:642713, May 2021.

[25] Benjamin Paul Redman. *The Use of Videoconferencing and Low-Latency Technologies for Instrumental Music Teaching*. Thesis, The University of St Andrews, June 2022.

[26] Álvaro Barbosa. Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation. *Leonardo Music Journal*, 13:53–59, December 2003.

[27] Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. An Overview on Networked Music Performance Technologies. *IEEE Access*, 4:8823–8843, December 2016.

[28] Luca Turchet and Cristina Rottondi. On the relation between the fields of Networked Music Performances, Ubiquitous Music, and Internet of Musical Things. *Personal and Ubiquitous Computing*, 27(5):1783–1792, October 2023.

[29] Luca Turchet, Carlo Fischione, Georg Essl, Damian Keller, and Mathieu Barthet. Internet of Musical Things: Vision and Challenges. *IEEE Access*, 6:61994–62017, September 2018.

[30] Luca Turchet, Francesco Antoniazzi, Fabio Viola, Fausto Giunchiglia, and György Fazekas. The Internet of Musical Things Ontology. *Journal of Web Semantics*, 60:100548, January 2020.

[31] Cristina Rottondi, Michele Buccoli, and Massimilliano Zanoni. Feature-Based Analysis of the Effects of Packet Delay on Networked Musical Interactions. *Journal of the Audio Engineering Society*, 63(11):864–875, December 2015.

[32] ITU. One-way transmission time. Technical report, International Telecommunication Union (ITU), May 2003.

[33] Dana Kemack Goot, Harshit Chaubey, Timothy Y. Hsu, and William Scott Deal. A Perceptual Evaluation of Music Real-Time Communication Applications. *IEEE Access*, 11:46860–46870, April 2023.

[34] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. Request for Comments RFC 3550, Internet Engineering Task Force, July 2003.

[35] Jon Postel. User Datagram Protocol. Request for Comments RFC 768, Internet Engineering Task Force, August 1980.

[36] Wesley Eddy. Transmission Control Protocol (TCP). Request for Comments RFC 9293, Internet Engineering Task Force, August 2022.

[37] Jérémie Lecomte and Tom Bäckström. *Packet Loss and Concealment*, pages 161–184. Springer International Publishing, Cham, 2017.

[38] N Tatlas, A Floros, T Zarouchas, and J Mourjopoulos. Perceptually-Optimized Error Concealment for Audio Over WLANs. *The Mediterranean Journal of Electronics and Communications*, 3:77–86, 2007.

[39] C. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12(5):40–48, September 1998.

[40] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod. A new technique for audio packet loss concealment. In *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference*, volume MiniConfInternet, pages 48–52, November 1996.

[41] Naofumi Aoki. A VoIP packet loss concealment technique taking account of pitch variation in pitch waveform replication. *Electronics and Communications in Japan (Part I: Communications)*, 89(3):1–9, 2006.

[42] Guoqiang Zhang and W. Bastiaan Kleijn. Autoregressive model-based speech packet-loss concealment. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4797–4800, April 2008.

[43] Florian Stimberg, Alex Narest, Alessio Bazzica, Lennart Kolmodin, Pablo Barrera González, Olga Sharonova, Henrik Lundin, and Thomas C. Walters. WaveNetEQ — Packet Loss Concealment with WaveRNN. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 672–676, November 2020.

[44] W. Verhelst and M. Roelands. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 554–557 vol.2, April 1993.

[45] Gagan L. Choudhury and Robert G. Cole. Design and analysis of optimal adaptive de-jitter buffers. *Computer Communications*, 27(6):529–537, April 2004.

[46] Christoffer Lauri and Johan Malmgren. Synchronization of streamed audio between multiple playback devices over an unmanaged IP network. Master's thesis, Lund University, October 2015.

[47] Matteo Sacchetto, Yuen Huang, Andrea Bianco, and Cristina Rottondi. Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications. In *Proceedings of the 17th International Audio Mostly Conference*, AM '22, pages 203–210, New York, NY, USA, October 2022. Association for Computing Machinery.

[48] Matteo Sacchetto, Cristina Rottondi, Antonio Servetti, Louena Shtrepi, Marco Masoero, and Andrea Valle. HiFiReM: un approccio unificato, web e nativo, per la didattica musicale remota. In *Proceedings of the XXIII CIM*, Ancona, October 2022. CIM.

[49] Matteo Sacchetto, Paolo Gastaldi, Chris Chafe, Cristina Rottondi, and Antonio Servetti. Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution. *Journal of the Audio Engineering Society*, 70(11):926–937, November 2022.

[50] Matteo Sacchetto, Leonardo Severi, Cristina Rottondi, Louena Shtrepi, Marco Masoero, Carlo Barbagallo, Andrea Valle, and Antonio Servetti. Pristine Quality Networked Music Performance System for the Web. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum 2023*, pages 2957–2964, Turin, Italy, September 2023. European Acoustics Association.

[51] Diego Bert, Nicola Domini, Riccardo Peloso, Leonardo Severi, Matteo Sacchetto, Andrea Bianco, and Cristina Rottondi. FPGA-based Low-Latency Audio Coprocessor for Networked Music Performance. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–8, October 2023.

[52] Matteo Sacchetto, Maria Sangüesa, Piera Bagnus, Chiara Nicora, and Cristina Rottondi. Collection of Design Directions for the Realization of a Visual Interface with Haptic Feedback to Convey the Notion of Sonic Grain to DHH Students. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–7, October 2023.

[53] Matteo Sacchetto, Cristina Rottondi, and Andrea Bianco. Implementation and optimization of Burg's method for real-time packet loss concealment in networked music performance applications. *Personal and Ubiquitous Computing*, 28(5):727–743, May 2024.

[54] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, and Cristina Rottondi. Introducing DUST: A Dataset of Real-Time UDP Sound Packet Traces. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–4, September 2024.

[55] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Gabriele Abbate, Antonio Paolillo, and Alessandro Giusti. Remote Orchestral Conduction via a Virtual Reality System. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–6, September 2024.

[56] Cristina Rottondi, Matteo Sacchetto, Leonardo Severi, and Andrea Bianco. Toward an Inclusive Framework for Remote Musical Education and Practices. *IEEE Access*, 12:173836–173849, November 2024.

[57] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Aleksandra Knapinska, and Piotr Lechowicz. Demonstration of a Networked Music

Performance Experience with MEVO, April 2024. In *2023 4th International Symposium on the Internet of Sounds*.

[58] Leonardo Severi, Matteo Sacchetto, Andrea Bianco, Cristina Rottondi, Gabriele Abbate, Antonio Paolillo, and Alessandro Giusti. Demonstration of remote orchestral conduction via a virtual reality system. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, September 2024.

[59] Jon Postel. Internet Protocol. Request for Comments RFC 791, Internet Engineering Task Force, September 1981.

[60] Steve E. Deering and Bob Hinden. Internet Protocol, Version 6 (IPv6) Specification. Request for Comments RFC 8200, Internet Engineering Task Force, July 2017.

[61] Robert Moskowitz, Daniel Karrenberg, Yakov Rekhter, Eliot Lear, and Geert Jan de Groot. Address Allocation for Private Internets. Request for Comments RFC 1918, Internet Engineering Task Force, February 1996.

[62] Chrisoula Alexandraki and Demosthenes Akoumianakis. Exploring New Perspectives in Network Music Performance: The DIAMOUSES Framework. *Computer Music Journal*, 34(2):66–83, June 2010.

[63] Jonathan Rosenberg, Christian Huitema, Rohan Mahy, and Joel Weinberger. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). Request for Comments RFC 3489, Internet Engineering Task Force, March 2003.

[64] Philip Matthews, Jonathan Rosenberg, Dan Wing, and Rohan Mahy. Session Traversal Utilities for NAT (STUN). Request for Comments RFC 5389, Internet Engineering Task Force, October 2008.

[65] Derek MacDonald and Bruce Lowekamp. NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN). Request for Comments RFC 5780, Internet Engineering Task Force, May 2010.

[66] Cullen Fluffy Jennings and Francois Audet. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. Request for Comments RFC 4787, Internet Engineering Task Force, January 2007.

[67] Philip Matthews, Jonathan Rosenberg, and Rohan Mahy. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Request for Comments RFC 5766, Internet Engineering Task Force, April 2010.

[68] Prashanth Patil, Tirumaleswar Reddy.K, and Dan Wing. Traversal Using Relays around NAT (TURN) Server Auto Discovery. Request for Comments RFC 8155, Internet Engineering Task Force, April 2017.

[69] Ari Keränen, Christer Holmberg, and Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal. Request for Comments RFC 8445, Internet Engineering Task Force, July 2018.

[70] ITU. Pulse code modulation (PCM) of voice frequencies. Technical report, International Telecommunication Union (ITU), November 1988.

[71] Martijn van Beurden and Anew Weaver. Free Lossless Audio Codec (FLAC). Request for Comments RFC 9639, Internet Engineering Task Force, December 2024.

[72] ISO. Information technology – coding of moving pictures and associated audio for digital storage media at up to 1,5 mbit/s – part 3: Audio. Technical Report ISO/IEC 11172-3, International Organization for Standardization, 1993.

[73] ISO. Information technology – coding of audio-visual objects – part 3: Audio. Technical Report ISO/IEC 14496-3, International Organization for Standardization, 2019.

[74] Jean-Marc Valin, Koen Vos, and Timothy B. Terriberry. Definition of the Opus Audio Codec. Request for Comments RFC 6716, Internet Engineering Task Force, September 2012.

[75] Cullen Jennings, Henrik Boström, and Jan-Ivar Bruaroey. WebRTC 1.0: Real-Time Communication Between Browsers. W3C Recommendation, World Wide Web Consortium (W3C), January 2021. https://www.w3.org/TR/2021/REC-webrtc-20210126/.

[76] Ali C. Begen, Paul Kyzivat, Colin Perkins, and Mark J. Handley. SDP: Session Description Protocol. Request for Comments RFC 8866, Internet Engineering Task Force, January 2021.

[77] Elisabetta Carrara, Karl Norrman, David McGrew, Mats Naslund, and Mark Baugher. The Secure Real-time Transport Protocol (SRTP). Request for Comments RFC 3711, Internet Engineering Task Force, March 2004.

[78] Randall R. Stewart, Michael Tüxen, and karen Nielsen. Stream Control Transmission Protocol. Request for Comments RFC 9260, Internet Engineering Task Force, June 2022.

[79] WebRTC API - Web APIs | MDN. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API.

[80] Joshue O'Connor, Michael Cooper, Alastair Campbell, and Andrew Kirkpatrick. Web content accessibility guidelines (WCAG) 2.1. W3C Recommendation, World Wide Web Consortium (W3C), December 2024. https://www.w3.org/TR/2024/REC-WCAG21-20241212/.

[81] Comparison of remote music performance software. https://en.wikipedia.org/wiki/Comparison_of_remote_music_performance_software.

[82] JamKazam. https://jamkazam.com/.

[83] SonoBus. https://sonobus.net/.

[84] Elk Live. https://elk.live/.

[85] Chris Chafe and Scott Oshiro. Jacktrip on Raspberry Pi. *Stanford University*, March 2019.

[86] JackTrip Virtual Studio. https://www.jacktrip.com/.

[87] Luca Turchet and Carlo Fischione. Elk Audio OS: An Open Source Operating System for the Internet of Musical Things. *ACM Trans. Internet Things*, 2(2):12:1–12:18, March 2021.

[88] 4D Jam Connect. https://www.4dcreatives.ca/jam.

[89] Jitsi. https://jitsi.org/.

[90] NINJAM. https://www.cockos.com/ninjam/.

[91] Niklas Blum, Serge Lachapelle, and Harald Alvestrand. WebRTC: Real-time communication for the open web platform. *Communications of the ACM*, 64(8):50–54, August 2021.

[92] Dongming Tang and Liqun Zhang. Audio and Video Mixing Method to Enhance WebRTC. *IEEE Access*, 8:67228–67241, April 2020.

[93] Mohannad Alahmadi, Peter Pocta, and Hugh Melvin. An Adaptive Bitrate Switching Algorithm for Speech Applications in Context of WebRTC. *ACM Trans. Multimedia Comput. Commun. Appl.*, 17(4):133:1–133:21, November 2021.

[94] Jean-Marc Valin, Gregory Maxwell, Timothy B. Terriberry, and Koen Vos. High-Quality, Low-Delay Music Coding in the Opus Codec. *Journal of the Audio Engineering Society*, October 2013.

[95] Oliwia Komperda, Hugh Melvin, and Peter Pocta. A Black Box Analysis of Webrtc Mouth-To-Ear Delays. *Communications - Scientific Letters of the University of Zilina*, 18(1):11–16, February 2016.

[96] Otto J. Wittner. Live audio and video over WebRTC's datachannel. http://in-legacy.uninett.no:8080/live-audio-and-video-over-webrtcs-datachannel/, Accessed Dec. 18, 2020.

[97] Paul Adenot. Web audio api vs. native: Closing the gap. In *1st International Web Audio Conference*, Paris, France, Jan 2015. https://medias.ircam.fr/x2af2f6.

[98] Paul Adenot. Web audio api vs. native, closing the gap, take 2. In *Proceedings of the 5th International Web Audio Conference (WAC)*, pages 178–179, Trondheim, Norway, Dec. 2019.

[99] Paul Adenot. Introduction to lock-free programming on the web with audioworklet. https://www.youtube.com/watch?v=T2Yonjy7-g4, accessed Apr. 5, 2020.

[100] Lorenz Diener, Sten Sootla, Solomiya Branets, Ando Saabas, Robert Aichner, and Ross Cutler. INTERSPEECH 2022 Audio Deep Packet Loss Concealment Challenge. http://arxiv.org/abs/2204.05222, April 2022.

[101] Lorenz Diener, Solomiya Branets, Ando Saabas, and Ross Cutler. The Icassp 2024 Audio Deep Packet Loss Concealment Grand Challenge. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 41–42, Seoul, Korea, Republic of, April 2024. IEEE.

[102] Peter Foster, Anssi Klapuri, and Mark D Plumbley. CAUSAL PREDICTION OF CONTINUOUS-VALUED MUSIC FEATURES. In *ISMIR 2011*, Miami, Florida, USA, October 2011.

[103] Akira Maezawa. Deep Linear Autoregressive Model for Interpretable Prediction of Expressive. In *SMC 2019*, Málaga, Spain, May 2019.

[104] Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang. Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding. http://arxiv.org/abs/2108.05064, August 2021.

[105] Koen Vos, Karsten Vandborg Sørensen, Søren Skak Jensen, and Jean-Marc Valin. Voice coding with Opus. In *Audio Engineering Society Convention 135*, October 2013.

[106] Koen Vos. A Fast Implementation of Burg's Method. *OPUS codec*, August 2013.

[107] Jussi Virolainen and Pauli Laine. Methods and apparatus for transmitting MIDI data over a lossy communications channel. US Patent No. US6898729B2, May 2005. Assignee: Nokia Oyj.

[108] Jason Robert NELSON, Allen James HEIDORN, and Robert John COX. Reliable real-time transmission of musical sound control data over wireless networks. US Patent No. US9601097B2, March 2017. Assignee: Zivix LLC.

[109] Prateek Verma, Alessandro Ilic Mezza, Chris Chafe, and Cristina Rottondi. A Deep Learning Approach for Low-Latency Packet Loss Concealment of Audio Signals in Networked Music Performance Applications. In *2020 27th Conference of Open Innovations Association (FRUCT)*, pages 268–275, October 2020.

[110] Jean-Pierre Briot. From artificial neural networks to deep learning for music generation: History, concepts and trends. *Neural Computing and Applications*, 33(1):39–65, January 2021.

[111] R. Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, January 1996.

[112] Adriana Olmos, Mathieu Brulé, Nicolas Bouillot, Mitchel Benovoy, Jeff Blum, Haijian Sun, Windfeld Lund, and Jeremy R Cooperstock. Exploring the role of latency and orchestra placement on the networked performance of a distributed opera. In *Proceedings of the 12th Annual International Workshop on Presence*, Los Angeles, November 2009.

[113] Alexander Carôt and Gerald Schuller. Towards a Telematic Visual-Conducting System. *Journal of the Audio Engineering Society*, 16, November 2011.

[114] Bavo Van Kerrebroeck, Kristel Crombé, Stéphane M. de Leymarie, Marc Leman, and Pieter-Jan Maes. The virtual drum circle: Polyrhythmic music interactions in mixed reality. *Journal of New Music Research*, 0(0):1–21, April 2024.

[115] Torin Hopkins, Suibi Che Chuan Weng, Rishi Vanukuru, Emma A. Wenzel, Amy Banic, Mark D. Gross, and Ellen Yi-Luen Do. AR Drum Circle: Real-Time Collaborative Drumming in AR. *Frontiers in Virtual Reality*, 3, August 2022.

[116] Hupke Robert, Preihs Stephan, and Peissig Juergen. Immersive room extension environment for networked music performance. *Journal of the Audio Engineering Society*, October 2022.

[117] Hunt Anthony, Daffern Helena, and Kearney Gavin. Avatar representation in extended reality for immersive networked music performance. *Journal of the Audio Engineering Society*, August 2023.

[118] Patrick Cairns, Anthony Hunt, Daniel Johnston, Jacob Cooper, Ben Lee, Helena Daffern, and Gavin Kearney. Evaluation of Metaverse Music Performance With BBC Maida Vale Recording Studios. *Journal of the Audio Engineering Society*, 71(6):313–325, June 2023.

[119] Byron Remache-Vinueza, Andrés Trujillo-León, Mireya Zapata, Fabián Sarmiento-Ortiz, and Fernando Vidal-Verdú. Audio-Tactile Rendering: A Review on Technology and Methods to Convey Musical Information through the Sense of Touch. *Sensors*, 21(19):6575, September 2021.

[120] Luis-Manuel Garcia. Beats, flesh, and grain: Sonic tactility and affect in electronic dance music. *Sound Studies*, 1(1):59–76, January 2015.

[121] Alice Haynes, Jonathan Lawry, Christopher Kent, and Jonathan Rossiter. FeelMusic: Enriching Our Emotive Experience of Music through Audio-Tactile Mappings. *Multimodal Technologies and Interaction*, 5(6):29, May 2021.

[122] Marcello Giordano, John Sullivan, and Marcelo M. Wanderley. Design of Vibrotactile Feedback and Stimulation for Music Performance. In Stefano Papetti and Charalampos Saitis, editors, *Musical Haptics*, pages 193–214. Springer International Publishing, Cham, May 2018.

[123] Alice-Ann Darrow. Music and the hearing impaired: A review of the research with implications for music educators. *Update (Loma Linda University. Ethics Center)*, 7(2):10–12, 1989.

[124] Tobias Grosshauser and Thomas Hermann. Augmented Haptics – An Interactive Feedback System for Musicians. In M. Ercan Altinsoy, Ute Jekosch, and Stephen Brewster, editors, *Haptic and Audio Interaction Design*, pages 100–108, Berlin, Heidelberg, January 2009. Springer.

[125] Luca Turchet, Mathieu Lagrange, Cristina Rottondi, György Fazekas, Nils Peters, Jan Østergaard, Frederic Font, Tom Bäckström, and Carlo Fischione. The Internet of Sounds: Convergent Trends, Insights, and Future Directions. *IEEE Internet of Things Journal*, 10(13):11264–11292, July 2023.

[126] Luca Turchet, Travis West, and Marcelo M. Wanderley. Touching the audience: Musical haptic wearables for augmented and participatory live music performances. *Personal and Ubiquitous Computing*, 25(4):749–769, August 2021.

[127] Hugo B. Lima, Carlos G. R. Dos Santos, and Bianchi S. Meiguins. A Survey of Music Visualization Techniques. *ACM Comput. Surv.*, 54(7):143:1–143:29, July 2021.

[128] Jeeeun Kim, Swamy Ananthanarayan, and Tom Yeh. Seen music: Ambient music data visualization for children with hearing impairments. In *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, pages 426–429, New York, NY, USA, June 2015. Association for Computing Machinery.

[129] Jordan Aiko Deja, Alexczar Dela Torre, Hans Joshua Lee, Jose Florencio Ciriaco, and Carlo Miguel Eroles. ViTune: A Visualizer Tool to Allow the Deaf and Hard of Hearing to See Music With Their Eyes. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, pages 1–8, New York, NY, USA, April 2020. Association for Computing Machinery.

[130] Thomas Barlow McHugh, Abir Saha, David Bar-El, Marcelo Worsley, and Anne Marie Piper. Towards Inclusive Streaming: Building Multimodal Music Experiences for the Deaf and Hard of Hearing. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, pages 1–7, New York, NY, USA, May 2021. Association for Computing Machinery.

[131] Soledad Zárate. *Captioning and Subtitling for d/Deaf and Hard of Hearing Audiences*. UCL Press, January 2021.

[132] Konstantinos Tsioutas, George Xylomenos, and Ioannis Doumanis. Aretousa: A competitive audio streaming software for Network Music Performance. *Journal of the Audio Engineering Society*, March 2019. paper 10201.

[133] Paul Adenot and Hongchan Choi. Web Audio API. W3C Recommendation, World Wide Web Consortium (W3C), June 2020. https://www.w3.org/TR/2020/CR-webaudio-20200611/.

[134] Hongchan Choi. Audioworklet: the Future of Web Audio. In *Proceedings of the 44th International Computer Music Conference*, Daegu, South Korea, Aug. 2018.

[135] Matteo Sacchetto, Antonio Servetti, and Chris Chafe. JackTrip-WebRTC: Networked music experiments with PCM stereo audio in a Web browser. In *Web Audio Conference*, July 2021.

[136] Google Meet. https://meet.google.com/.

[137] Randell Jesup, Salvatore Loreto, and Michael Tüxen. WebRTC Data Channels. Request for Comments RFC 8831, Internet Engineering Task Force, January 2021.

[138] Sekhar Chandra, Senthil Kumar, and Bala Prasad. Audio mixer for multi-party conferencing in voip. In *Proceedings of the 3rd IEEE Intl. Conf. on Internet Multimedia Services Architecture and Applications (IMSAA)*, pages 1–6, Bangalore, India, Jan. 2010.

[139] ITU. Buffer models for development of client performance metric. Technical report, International Telecommunication Union (ITU), July 2012.

[140] Zoom. https://www.zoom.us/.

[141] Skype. https://www.skype.com/.

[142] Audiotool. https://www.audiotool.com/.

[143] Stanley P. Lipshitz and John Vanderkooy. Pulse-Code Modulation–An Overview. *Journal of the Audio Engineering Society*, 52(3):200–215, March 2004.

[144] ITU. Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems. Technical report, International Telecommunication Union (ITU), April 1997.

[145] W. A. Munson and Mark B. Gardner. Standardizing Auditory Tests. *The Journal of the Acoustical Society of America*, 22(5_Supplement):675, September 1950.

[146] Nuggehally S. Jayant and Peter Noll. *Digital coding of waveforms. Principles and applications to speech and video*. Elsevier, 1985.

[147] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., USA, July 1993.

[148] Nicola Domini. Ultralow latency audio processing via FPGA for Networked Music Performance applications. Master's thesis, Politecnico di Torino, October 2022.

[149] Diego Bert. FPGA-based implementation of audio effects for ultralow-latency Networked Music Performance applications. Master's thesis, Politecnico di Torino, April 2023.

[150] Clemens Wegener, Sebastian Stang, and Max Neupert. FPGA-accelerated Real-Time Audio in Pure Data. In *Proceedings of the 19th Sound and Music Computing Conference*, June 2022.

[151] Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stéphane Letz. Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing. In *Proceedings of the 19th Sound and Music Computing Conference*, June 2022.

[152] Tanguy Risset, Romain Michon, Yann Orlarey, Stéphane Letz, Gero Müller, and Adeyemi Gbadamosi. Faust2FPGA for Ultra-Low Audio Latency: Preliminary work in the Syfala project. In *Proceedings of the 2nd International Faust Conference (IFC-20)*, pages 1–9, December 2020.

[153] George Xylomenos, Christos Tsilopoulos, Yannis Thomas, and George C Polyzos. Reduced switching delay for networked music performance. In *Packet Video Workshop (Poster Session)*. Citeseer, December 2013.

[154] George Baltas and George Xylomenos. Ultra low delay switching for networked music performance. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, pages 70–74, August 2014.

[155] Pekka Jääskeläinen, Timo Viitanen, Jarmo Takala, and Heikki Berg. HW/SW Co-design Toolset for Customization of Exposed Datapath Processors. In Waqar Hussain, Jari Nurmi, Jouni Isoaho, and Fabio Garzia, editors, *Computing Platforms for Software-Defined Radio*, pages 147–164. Springer International Publishing, Cham, December 2017.

[156] PCM1803. https://www.ti.com/product/PCM1803.

[157] PCM1771. https://www.ti.com/product/PCM1771.

[158] Arty A7 Reference Manual. https://digilent.com/reference/programmable-logic/arty-a7/reference-manual.

[159] Snorre Farner, Audun Solvang, Asbjørn Sæbo, and U Peter Svensson. Ensemble hand-clapping experiments under the influence of delay and various acoustic environments. *Journal of the Audio Engineering Society*, 57(12):1028–1041, October 2006.

[160] Datasheet Raspberry Pi 4B. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

[161] Datasheet BCM2711 ARM Peripherals. https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf.

[162] Hirotugu Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21(1):243–247, December 1969.

[163] David A. Dickey and Wayne A. Fuller. Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.

[164] John Parker Burg. *Maximum Entropy Spectral Analysis*. PhD thesis, Stanford University, May 1975.

[165] Norman Levinson. The Wiener (Root Mean Square) Error Criterion in Filter Design and Prediction. *Journal of Mathematics and Physics*, 25(1-4):261–278, April 1946.

[166] L.J. Faber. Commentary on the denominator recursion for Burg's block algorithm. *Proceedings of the IEEE*, 74(7):1046–1047, July 1986.

[167] Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi. Accurate Sum and Dot Product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, January 2005.

[168] Stef Graillat and Valérie Ménissier-Morain. Error-free transformations in real and complex floating point arithmetic. In *Proceedings of the International Symposium on Nonlinear Theory and Its Applications*, April 2007.

[169] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with Python. In *Proc. of the 9th Python in Science Conference*, 2010.

[170] Christopher A. Sims. Macroeconomics and Reality. *Econometrica*, 48(1):1–48, January 1980.

[171] Eric Zivot and Jiahui Wang. Vector Autoregressive Models for Multivariate Time Series. In *Modeling Financial Time Series with S-PLUS®*, pages 385–429. Springer, New York, NY, January 2006.

[172] O.J. Wasem, D.J. Goodman, C.A. Dvorak, and H.G. Page. The effect of waveform substitution on the quality of PCM packet communications. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(3):342–348, March 1988.

[173] Marco Fink, Martin Holters, and Udo Zölzer. Comparison of various predictors for audio extrapolation. In *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, pages 1–7, September 2013.

[174] Christian Sejer Pedersen, Mo Zhou, Martin Bo Møller, Niels de Koeijer, Niels Evert Marius, and Jan Østergaard. AR model for low latency packet loss concealment for wireless sound zones at low frequencies. *Audio Engineering Society 154th Convention*, pages 1–10, May 2023.

[175] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94, New York, NY, USA, February 2012. Association for Computing Machinery.

[176] Babak Naderi, Ross Cutler, Nabakumar Singh Khongbantabam, Yasaman Hosseinkashi, Henrik Turbell, Albert Sadovnikov, and Quan Zou. VCD: A Video Conferencing Dataset for Video Compression. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3970–3974, March 2024.

[177] Frederick A. Seddon. Modes of communication during jazz improvisation. *British Journal of Music Education*, 22(1):47–61, March 2005.

[178] B Iribe. Virtual reality-a new frontier in computing. Oculus VR, 2013.

[179] Luca Turchet. Musical Metaverse: Vision, opportunities, and challenges. *Personal and Ubiquitous Computing*, 27(5):1811–1827, October 2023.

[180] Gema F.B. Martín. Social and psychological impact of musical collective creative processes in virtual environments; The Avatar Orchestra Metaverse in Second Life. *Musica/Tecnologia*, pages 75–87, August 2018.

[181] Luca Vignati, Giovanni Nardini, Marco Centenaro, Paolo Casari, Sandra Lagén, Biljana Bojovic, Stefano Zambon, and Luca Turchet. Is Music in the Air? Evaluating 4G and 5G Support for the Internet of Musical Things. *IEEE Access*, 12:38081–38101, 2024.

[182] Alberto Boem and Luca Turchet. Musical Metaverse Playgrounds: Exploring the design of shared virtual sonic experiences on web browsers. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–9, October 2023.

[183] Damian Dziwis, Henrik Von Coler, and Christoph Pörschmann. Orchestra: A Toolbox for Live Music Performances in a Web-Based Metaverse. *Journal of the Audio Engineering Society*, 71(11):802–812, November 2023.

[184] Damian Dziwis and Henrik von Coler. The Entanglement: Volumetric Music Performances in a Virtual Metaverse Environment. *Journal of Network Music and Arts*, 5(1), May 2023.

[185] Diar Abdlkarim, Massimiliano Di Luca, Poppy Aves, Mohamed Maaroufi, Sang-Hoon Yeo, R. Chris Miall, Peter Holland, and Joeseph M. Galea. A methodological framework to assess the accuracy of virtual reality hand-tracking systems: A case study with the Meta Quest 2. *Behavior Research Methods*, 56(2):1052–1063, February 2023.

[186] Vision pro and quest 3 hand-tracking latency compared. https://www.roadtovr.com/apple-vision-pro-meta-quest-3-hand-tracking-latency-comparison/. Accessed on May 01, 2024.

[187] Cindy Orosy-Fildes and Robert W. Allan. Psychology of computer use: XII. Videogame play: Human reaction time to visual stimuli. *Perceptual and Motor Skills*, 69(1):243–247, 1989.

[188] Luca Turchet, David Baker, and Tony Stockman. Musical Haptic Wearables for Synchronisation of Visually-impaired Performers: A Co-design Approach. In *Proceedings of the 2021 ACM International Conference on Interactive Media Experiences*, IMX '21, pages 20–27, New York, NY, USA, June 2021. Association for Computing Machinery.

[189] Marina Christine McLerran. *Instrumental Music Education for Middle School Students with Hearing Impairment*. PhD thesis, University of Miami, May 2021.

[190] Roshanak Kheshti. Touching Listening: The Aural Imaginary in the World Music Culture Industry. *American Quarterly*, 63(3):711–731, September 2011.

[191] Jeremy Wallach. The Poetics of Electrosonic Presence: Recorded Music and the Materiality of Sound. *Journal of Popular Music Studies*, 15(1):34–64, June 2003.

[192] Charles Spence. Crossmodal correspondences: A tutorial review. *Attention, Perception, & Psychophysics*, 73(4):971–995, May 2011.

[193] Rachel Guetta and Psyche Loui. When music is salty: The crossmodal associations between sound and taste. *PLOS ONE*, 12(3):e0173366, March 2017.

[194] Anja Moos, David Simmons, Julia Simner, and Rachel Smith. Color and texture associations in voice-induced synesthesia. *Frontiers in Psychology*, 4, September 2013.

[195] Caroline Curwen. Music-colour synaesthesia: Concept, context and qualia. *Consciousness and Cognition*, 61:94–106, May 2018.

[196] Ni Luh Putu Sri Adnyani, Ni Made Rai Wisudariani, Gede Aditra Pradnyana, I Made Ardwi Pradnyana, and Ni Komang Arie Suwastini. Multimedia English Learning Materials for Deaf or Hard of Hearing (DHH) Children. *Journal of Education Technology*, 5(4):571, November 2021.

[197] Wajdi Aljedaani, Mona Aljedaani, Eman Abdullah AlOmar, Mohamed Wiem Mkaouer, Stephanie Ludi, and Yousef Bani Khalaf. I Cannot See You—The Perspectives of Deaf Students to Online Learning during COVID-19 Pandemic: Saudi Arabia Case Study. *Education Sciences*, 11(11):712, November 2021.

[198] Sama'a Al Hashimi and Dalal Alsindi. Optimizing Online Learning Experiences and Outcomes for Hearing-Impaired Art and Design Students. *International Journal of Learning, Teaching and Educational Research*, 20(7):1–22, July 2021.

[199] Faisl M. Alqraini and Khalid N. Alasim. Distance Education for d/Deaf and Hard of Hearing Students during the COVID-19 Pandemic in Saudi Arabia: Challenges and Support. *Research in Developmental Disabilities*, 117:104059, October 2021.

[200] Deafness and hearing loss. https://www.who.int/en/news-room/fact-sheets/detail/deafness-and-hearing-loss, 2023. Last accessed: 19-07-2023.

[201] Daniel J Levitin. This is your brain on music: The science of a human obsession, plume, 2007.

[202] Afonso Castiço and Paulo Cardoso. Usability Tests for Texture Comparison in an Electroadhesion-Based Haptic Device. *Multimodal Technologies and Interaction*, 6(12):108, December 2022.

[203] Allan Collins and Richard Halverson. *Rethinking Education in the Age of Technology: The Digital Revolution and Schooling in America*. Teachers College Press, January 2018.

[204] Agariadne Dwinggo Samala, Soha Rawas, Tianchong Wang, Janet Marie Reed, Jinhee Kim, Natalie-Jane Howard, and Myriam Ertz. Unveiling the landscape of generative artificial intelligence in education: A comprehensive taxonomy of applications, challenges, and future prospects. *Education and Information Technologies*, August 2024.

[205] National Center for Education Statistics, U.S. Department of Education, Institute of Education Sciences. Students with disabilities. condition of education., 2022. https://nces.ed.gov/programs/coe/indicator/cgg.

[206] European Music School Union. European music schools in times of coronavirus, 2020. http://www.musicschoolunion.eu/wp-content/uploads/2020/05/EMU-Survey-Coronavirus.pdf".

[207] William Christopher Payne, Fabiha Ahmed, Michael Zachor, Michael Gardell, Isabel Huey, Amy Hurst, and R. Luke Dubois. Empowering Blind Musicians to Compose and Notate Music with SoundCells. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '22, pages 1–14, New York, NY, USA, October 2022. Association for Computing Machinery.

[208] Jens Vetter. WELLE - a web-based music environment for the blind. In *IME'20*, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom, July 2020.

[209] Abir Saha, Thomas Barlow McHugh, and Anne Marie Piper. Tutoria11y: Enhancing Accessible Interactive Tutorial Creation by Blind Audio Producers. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–14, New York, NY, USA, April 2023. Association for Computing Machinery.

[210] Joseph Abramo. Disability in the Classroom: Current Trends and Impacts on Music Education. *Music Educators Journal*, 99(1):39–45, September 2012.

[211] Emma Frid and Alon Ilsar. Reimagining (Accessible) Digital Musical Instruments: A Survey on Electronic Music-Making Tools. In *NIME 2021*, Shanghai, China, June 2021. PubPub.

[212] Emma Frid. Accessible Digital Musical Instruments—A Review of Musical Interfaces in Inclusive Music Practice. *Multimodal Technologies and Interaction*, 3(3):57, July 2019.

[213] Phillip M. Hash. Teaching Instrumental Music to Deaf and Hard of Hearing Students. *Research and Issues in Music Education*, 1(1), September 2003.

[214] Fabio Catania, Pietro Crovari, Eleonora Beccaluva, Giorgio De Luca, Erica Colombo, Nicola Bombaci, and Franca Garzotto. Boris: A Spoken Conversational Agent for Music Production for People with Motor Disabilities. In *Proceedings of the 14th Biannual Conference of the Italian SIGCHI Chapter*, CHItaly '21, pages 1–5, New York, NY, USA, July 2021. Association for Computing Machinery.

[215] Judith A. Jellison and Donald M. Taylor. Attitudes toward Inclusion and Students with Disabilities: A Review of Three Decades of Music Research. *Bulletin of the Council for Research in Music Education*, pages 9–23, April 2007.

[216] Sara K. Jones. Teaching students with disabilities: A review of music education research as it relates to the individuals with disabilities education act. *Update: Applications of Research in Music Education*, 34(1):13–23, October 2014.

[217] Tuulikki Laes and Heidi Westerlund. Performing disability in music teacher education: Moving beyond inclusion through expanded professionalism. *International Journal of Music Education*, 36(1):34–46, February 2018.

[218] Luca Turchet, Rob Hamilton, and Anil Çamci. Music in Extended Realities. *IEEE Access*, 9:15810–15832, January 2021.

[219] Florent Berthaut. 3D interaction techniques for musical expression. *Journal of New Music Research*, 49(1):60–72, January 2020.

[220] Stefania Serafin, Ali Adjorlu, Niels Nilsson, Lui Thomsen, and Rolf Nordahl. Considerations on the use of virtual and augmented reality technologies in music education. In *2017 IEEE Virtual Reality Workshop on K-12 Embodied*

Learning through Virtual & Augmented Reality (KELVAR), pages 1–4, June 2017.

[221] Alejandro Flores Ramones and Marta Sylvia del-Rio-Guerra. Recent Developments in Haptic Devices Designed for Hearing-Impaired People: A Literature Review. *Sensors*, 23(6):2968, March 2023.

[222] Luca Turchet and Mathieu Barthet. Co-Design of Musical Haptic Wearables for Electronic Music Performer's Communication. *IEEE Transactions on Human-Machine Systems*, 49(2):183–193, April 2019.

[223] Diana Carvalho, João Barroso, and Tânia Rocha. Multisensory Experience for People with Hearing Loss: A Preliminary Study Using Haptic Interfaces to Sense Music. In Vincent G. Duffy, Qin Gao, Jia Zhou, Margherita Antona, and Constantine Stephanidis, editors, *HCI International 2022 – Late Breaking Papers: HCI for Health, Well-being, Universal Access and Healthy Aging*, pages 292–306, Cham, October 2022. Springer Nature Switzerland.

[224] Mark D. Fletcher. Can Haptic Stimulation Enhance Music Perception in Hearing-Impaired Listeners? *Frontiers in Neuroscience*, 15, August 2021.

[225] Ira J. Hirsh and Carl E. Sherrick Jr. Perceived order in different sense modalities. *Journal of Experimental Psychology*, 62(5):423–432, 1961.

[226] M Ercan Altinsoy. Perceptual aspects of auditory-tactile asynchrony. In *Proceedings of the Tenth International Congress on Sound and Vibration*, pages 3831–3838, 2003.

[227] Bryan Gick, Yoko Ikegami, and Donald Derrick. The temporal window of audio-tactile integration in speech perception. *The Journal of the Acoustical Society of America*, 128(5):EL342–EL346, November 2010.

[228] Irina B. Gorbunova and Anasyasia A. Govorova. Music Computer Technologies as a Means of Teaching the Musical Art for Visually-Impaired People. In *TSETWM-18,CAAES-18,LLHSS-18,LEBDM-18 Oct. 2-4, 2018 Budapest (Hungary)*. Universal Researchers, October 2018.

[229] David Antón, Gregorij Kurillo, Alfredo Goñi, Arantza Illarramendi, and Ruzena Bajcsy. Real-time communication for Kinect-based telerehabilitation. *Future Generation Computer Systems*, 75:72–81, October 2017.

[230] Aristotelis Hadjakos. Pianist Motion Capture with the Kinect Depth Camera. In *Proceedings of the 9th Sound and Music Computing Conference*, Copenhagen, Denmark, July 2012.

[231] Aristotelis Hadjakos, Tobias Großhauser, and Werner Goebl. Motion Analysis of Music Ensembles with the Kinect. In *Proceedings of the 13th International Conference on New Interfaces for Musical Expression*, Daejeon + Seoul, Korea Republic, May 2013.

[232] Matthew Warburton, Mark Mon-Williams, Faisal Mushtaq, and J. Ryan Morehead. Measuring motion-to-photon latency for sensorimotor experiments with virtual reality systems. *Behavior Research Methods*, 55(7):3658–3678, October 2023.

[233] Robert Pelzer, Manoj Dinakaran, Fabian Brinkmann, Steffen Lepa, Peter Grosche, and Stefan Weinzierl. Head-related transfer function recommendation based on perceptual similarities and anthropometric features. *The Journal of the Acoustical Society of America*, 148(6):3809–3817, December 2020.

[234] Christopher Frauenberger and Markus Noisternig. 3D Audio Interfaces for the Blind. In *Proceedings of the 2003 International Conference on Auditory Display*, July 2003.

[235] Brian F. G. Katz and Lorenzo Picinali. Spatial Audio Applied to Research with the Blind. In Pawel Strumillo, editor, *Advances in Sound Localization*. InTech, April 2011.

[236] Amandine Afonso-Jaco and Brian F. G. Katz. Spatial Knowledge via Auditory Information for Blind Individuals: Spatial Cognition Studies and the Use of Audio-VR. *Sensors*, 22(13):4794, June 2022.

[237] Jeffrey R. Blum, Mathieu Bouchard, and Jeremy R. Cooperstock. What's around Me? Spatialized Audio Augmented Reality for Blind Users with a Smartphone. In Alessandro Puiatti and Tao Gu, editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 49–62, Berlin, Heidelberg, January 2012. Springer.

[238] Afnen Arfaoui, Geoffrey Edwards, Ernesto Morales, Patrick Fougeyrollas, Afnen Arfaoui, Geoffrey Edwards, Ernesto Morales, and Patrick Fougeyrollas. Designing Interactive and Immersive Multimodal Installations for People with Disability. In *Virtual Reality and Its Application in Education*. IntechOpen, January 2020.

[239] Patrick Cairns, Helena Daffern, and Gavin Kearney. Parametric Evaluation of Ensemble Vocal Performance Using an Immersive Network Music Performance Audio System. *Journal of the Audio Engineering Society*, 69(12):924–933, December 2021.

[240] Luca Turchet, Claudia Rinaldi, Carlo Centofanti, Luca Vignati, and Cristina Rottondi. 5G-Enabled Internet of Musical Things Architectures for Remote Immersive Musical Practices. *IEEE Open Journal of the Communications Society*, 5:4691–4709, May 2024.

[241] Luca Turchet and Matteo Tomasetti. Immersive networked music performance systems: Identifying latency factors. In *2023 Immersive and 3D Audio: From Architecture to Automotive (I3DA)*, pages 1–6, October 2023.

[242] Marcella Mandanici, Roberto Di Filippo, and Stefano Delle Monache. The Discovery of Interactive Spaces: Learning by Design in High School Music Technology Classes. *Technology, Knowledge and Learning*, 26(4):1131–1151, December 2021.

[243] Valéria Farinazzo Martins, Letícia Gomez, and Ana Grasielle Dionísio Corrêa. Teaching Children Musical Perception with MUSIC-AR. *EAI Endorsed Transactions on e-Learning*, 2(5):e3–e3, March 2015.

[244] Edoardo Degli Innocenti, Michele Geronazzo, Diego Vescovi, Rolf Nordahl, Stefania Serafin, Luca Andrea Ludovico, and Federico Avanzini. Mobile virtual reality for musical genre learning in primary education. *Computers & Education*, 139:102–117, October 2019.

[245] Angelos Barmpoutis, Randi Faris, Luis Garcia, Luis Gruber, Jingyao Li, Fray Peralta, and Menghan Zhang. Assessing the Role of Virtual Reality with Passive Haptics in Music Conductor Education: A Pilot Study. In Jessie Y. C. Chen and Gino Fragomeni, editors, *Virtual, Augmented and Mixed Reality. Design and Interaction*, pages 275–285, Cham, July 2020. Springer International Publishing.

[246] Andrew King, Helen Prior, and Caroline Waddington-Jones. Connect Resound: Using online technology to deliver music education to remote communities. *Journal of Music, Technology & Education*, 12(2):201–217, September 2019.

[247] Phillip M. Hash. Remote Learning in School Bands During the COVID-19 Shutdown. *Journal of Research in Music Education*, 68(4):381–397, December 2020.

[248] Diego Calderón-Garrido and Josep Gustems-Carnicer. Adaptations of music education in primary and secondary school due to COVID-19: The experience in Spain. *Music Education Research*, 23(2):139–150, 15/13/2021.

[249] Emmett J. O'Leary and Julie K. Bannerman. Technology Integration in Music Education and the COVID-19 Pandemic. *Bulletin of the Council for Research in Music Education*, pages 23–40, October 2023.

[250] Nnamdi C. Onuora-Oguno and David Sopuluchukwu Umeojiaka. THE DOMINANCE OF E-LEARNING AND INFORMATION COMMUNICATION TECHNOLOGY (ICT) IN NIGERIAN MUSIC EDUCATION: PROBLEMS AND PROSPECTS. *Journal of African Studies and Sustainable Development*, 3(1), April 2020.

[251] P. Paul, P. S. Aithal, S. Sharma, and R. Saavedra. Infrastructure-Technology, Socio-Economical Issues and Challenges in ICT-based Education and Digital Education with Possible Solutions-A Scientific Observation, May 2023.

[252] Muhamad Suhaimi Taat and Agatha Francis. Factors Influencing the Students' Acceptance of E-Learning at Teacher Education Institute: An Exploratory Study in Malaysia. *International Journal of Higher Education*, 9(1):133–141, 2020.

[253] Maruf Gbadebo Salimon, Sany Mohd Mokhtar Sanuri, Olayemi Abdullateef Aliyu, Selvan Perumal, and Maha Mohammed Yusr. E-learning satisfaction and retention: A concurrent perspective of cognitive absorption, perceived social presence and technology acceptance model. *Journal of Systems and Information Technology*, 23(1):109–129, June 2021.

[254] Dayong Nie, Elena Panfilova, Vadim Samusenkov, and Alexey Mikhaylov. E-Learning Financing Models in Russia for Sustainable Development. *Sustainability*, 12(11):4412, May 2020.

[255] Niklas Hanes and Sofia Lundberg. E-learning as a Regional Policy Tool: Principles for a Cost-benefit Analysis. *RUSC. Universities and Knowledge Society Journal*, 5(1), April 2008.

[256] Reina Setiawan, Agus Putranto, Wihendro, Elfindah Princes, Ira Geraldina, Eka Julianti, Julia Safitri, and Paulina Pannen. Tech-Driven Transformation: Innovative Pricing Strategies for E-Learning. *IEEE Access*, 12:59063–59078, May 2024.

[257] Juan-Pablo Cáceres, Robert Hamilton, Deepak Iyer, Chris Chafe, and Ge Wang. To the edge with China: Explorations in network performance. In *ARTECH 2008: Proceedings of the 4th International Conference on Digital Arts*, pages 61–66. ARTECH US, November 2008.

[258] Benjamin Redman and Jonathan Impett. The potential of videoconferencing and low-latency (LoLa) technology for instrumental music teaching. *Music & Practice*, 6:0, July 2020.

[259] Luca Turchet and Paolo Casari. Latency and Reliability Analysis of a 5G-Enabled Internet of Musical Things System. *IEEE Internet of Things Journal*, 11(1):1228–1240, January 2024.

[260] Weiya Chen, Chenchen Yu, Chenyu Tu, Zehua Lyu, Jing Tang, Shiqi Ou, Yan Fu, and Zhidong Xue. A Survey on Hand Pose Estimation with Wearable Sensors and Computer-Vision-Based Methods. *Sensors*, 20(4):1074, January 2020.

[261] Oliver Glauser, Shihao Wu, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. Interactive hand pose estimation using a stretch-sensing soft glove. *ACM Trans. Graph.*, 38(4):41:1–41:15, July 2019.

[262] Matteo Cognolato, Manfredo Atzori, and Henning Müller. Head-mounted eye gaze tracking devices: An overview of modern devices and recent advances. *Journal of Rehabilitation and Assistive Technologies Engineering*, 5:2055668318773991, January 2018.

[263] José Cláudio da Silva Junior and Walter Germanovix. Development of a Sip-and-Puff Interface for Communication and Control of Devices. In César A. González Díaz, Christian Chapa González, Eric Laciar Leber, Hugo A. Vélez, Norma P. Puente, Dora-Luz Flores, Adriano O. Andrade, Héctor A. Galván, Fabiola Martínez, Renato García, Citlalli J. Trujillo, and Aldo R. Mejía, editors, *VIII Latin American Conference on Biomedical Engineering and XLII National Conference on Biomedical Engineering*, pages 1137–1146, Cham, October 2019. Springer International Publishing.

[264] Quinn Jarvis-Holland, Crystal Cortez, Nathan, Gamill, and Francisco Botello. Expanding Access to Music Technology – Rapid Prototyping Accessible Instrument Solutions For Musicians With Intellectual Disabilities. http://arxiv.org/abs/2011.09143, November 2020.

[265] Amble Skuse and Shelly Knotts. Creating an Online Ensemble for Home Based Disabled Musicians: Why disabled people must be at the heart of developing technology. In *NIME '20*, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom, July 2020.

[266] Seth Thorn. Telematic Wearable Music: Remote Ensembles and Inclusive Embodied Education. In *Proceedings of the 16th International Audio Mostly Conference*, AM '21, pages 188–195, New York, NY, USA, October 2021. Association for Computing Machinery.

[267] V. J. Kartsch Morinigo, Simone Benatti, and Luca Benini. A High SNR, Low-latency Dry EMG Acquisition System for Unobtrusive HMI Devices. In *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 544–548, November 2022.

[268] Julian Kreimeier and Timo Götzelmann. Two Decades of Touchable and Walkable Virtual Reality for Blind and Visually Impaired People: A High-Level Taxonomy. *Multimodal Technologies and Interaction*, 4(4):79, November 2020.

[269] Mark D. Fletcher. Using haptic stimulation to enhance auditory perception in hearing-impaired listeners. *Expert Review of Medical Devices*, 18(1):63–74, January 2021.

[270] Yu Chen, Yiduo Yang, Mengjiao Li, Erdong Chen, Weilei Mu, Rosie Fisher, and Rong Yin. Wearable Actuators: An Overview. *Textiles*, 1(2):283–321, August 2021.

[271] Rubén Gonzalez Crespo, Jordán Pascual Espada, and Daniel Burgos. Social4all: Definition of specific adaptations in Web applications to improve accessibility. *Computer Standards & Interfaces*, 48:1–9, November 2016.

[272] Waseem Safi, Fabrice Maurel, Jean-Marc Routoure, Pierre Beust, Michèle Molina, Coralie Sann, and Jessica Guilbert. Blind Navigation of Web Pages through Vibro-tactile Feedbacks. In *25th ACM Symposium on Virtual Reality*

Software and Technology, pages 1–1, Parramatta NSW Australia, November 2019. ACM.

[273] Ben Loveridge. Networked Music Performance in Virtual Reality: Current Perspectives. *Journal of Network Music and Arts*, 2, January 2020.

[274] Seyed Salar Sefati and Simona Halunga. Ultra-reliability and low-latency communications on the internet of things based on 5G network: Literature review, classification, and future research view. *Transactions on Emerging Telecommunications Technologies*, 34(6):e4770, April 2023.

[275] Jan Dürre, Norbert Werner, Seppo Hämäläinen, Oscar Lindfors, Janne Koistinen, Miro Saarenmaa, and Robert Hupke. In-Depth Latency and Reliability Analysis of a Networked Music Performance over Public 5G Infrastructure. *Journal of the Audio Engineering Society*, October 2022.

[276] Luca Turchet and Edoardo Rinaldo. Technical Performance Assessment of the Ableton Link Protocol Over Wi-Fi. *Journal of the Audio Engineering Society*, 69(10):748–756, October 2021.

[277] Marco Centenaro, Paolo Casari, and Luca Turchet. Towards a 5G Communication Architecture for the Internet of Musical Things. In *2020 27th Conference of Open Innovations Association (FRUCT)*, pages 38–45, October 2020.

[278] Lina Mohjazi, Bassant Selim, Mallik Tatipamula, and Muhammad Ali Imran. The Journey Toward 6G: A Digital and Societal Revolution in the Making. *IEEE Internet of Things Magazine*, 7(2):119–128, March 2024.

[279] Xuelinzi Bai. Remote Music Learning Based on Wireless Sensors Supporting 6G and CPS. *Wireless Personal Communications*, May 2024.

[280] Riccardo Battello, Luca Comanducci, Fabio Antonacci, Giovanni Cospito, and Augusto Sarti. Experimenting With Adaptive Metronomes in Networked Music Performances! *Journal of the Audio Engineering Society*, 69(10):737–747, October 2021.

[281] Rob Hamilton and Chris Platz. Gesture-based collaborative virtual reality performance in carillon. In *Proceedings of the 2016 International Computer Music Conference*, pages 337–340, September 2016.

[282] Xin Wei, Meng Zhang, and Liang Zhou. Cross-Modal Transmission Strategy. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(6):3991–4003, August 2021.

[283] Ethan Blanton, Vern Paxson, and Mark Allman. TCP Congestion Control. Request for Comments RFC 5681, Internet Engineering Task Force, September 2009.

[284] Xin Wei, Yuyuan Yao, Haoyu Wang, and Liang Zhou. Perception-Aware Cross-Modal Signal Reconstruction: From Audio-Haptic to Visual. *IEEE Transactions on Multimedia*, 25:5527–5538, July 2022.

[285] Yanan Chen, Peilin Li, Ang Li, Dan Wu, Liang Zhou, and Yi Qian. Toward Low-Latency Cross-Modal Communication: A Flexible Prediction Scheme. *IEEE Transactions on Mobile Computing*, 23(12):13310–13324, December 2024.

[286] Xin Wei, Dan Wu, Liang Zhou, and Mohsen Guizani. Cross-modal communication technology: A survey. *Fundamental Research*, September 2023.

[287] Christian Werner and Robert Kraneis. UNISON: A Novel System for Ultra-Low Latency Audio Streaming Over the Internet. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4, March 2021.

[288] Paul Ferguson, Chris Chafe, and Simon Gapp. Trans-Europe Express Audio: Testing 1000 mile low-latency uncompressed audio between Edinburgh and Berlin using GPS-derived word clock, first with jacktrip then with Dante. *Journal of the Audio Engineering Society*, May 2020.

[289] Luca Turchet and Francesco Antoniazzi. Semantic Web of Musical Things: Achieving interoperability in the Internet of Musical Things. *Journal of Web Semantics*, 75:100758, January 2023.

[290] Rômulo Vieira and Flávio Schiavoni. Sunflower: An environment for standardized communication of IoMusT. In *Proceedings of the 16th International Audio Mostly Conference*, AM '21, pages 175–181, New York, NY, USA, October 2021. Association for Computing Machinery.

[291] Kashif Nisar, Emilia Rosa Jimson, Mohd Hanafi Ahmad Hijazi, Ian Welch, Rosilah Hassan, Azana Hafizah Mohd Aman, Ali Hassan Sodhro, Sandeep Pirbhulal, and Sohrab Khan. A survey on the architecture, application, and security of software defined networking: Challenges and open issues. *Internet of Things*, 12:100289, December 2020.

[292] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, February 2015.

[293] Abderrahime Filali, Amine Abouaomar, Soumaya Cherkaoui, Abdellatif Kobbane, and Mohsen Guizani. Multi-Access Edge Computing: A Survey. *IEEE Access*, 8:197017–197046, October 2020.

[294] Kyungeun Lee, Moonjung Eo, Euna Jung, Yoonjin Yoon, and Wonjong Rhee. Short-Term Traffic Prediction With Deep Neural Networks: A Survey. *IEEE Access*, 9:54739–54756, April 2021.

[295] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Murtaza Choudhury, and A. K. Qin. A Survey on Modern Deep Neural Network for Traffic Prediction: Trends, Methods and Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 34(4):1544–1561, April 2022.

[296] Emmanouil Lakiotakis, Christos Liaskos, and Xenofontas Dimitropoulos. Improving networked music performance systems using application-network collaboration. *Concurrency and Computation: Practice and Experience*, 31(24):e4730, May 2018.