Design and Implementation of a Latency-tolerant RTC Audio Framework for DAW.

SHAN JIANG, Georgia Institute of Technology, USA

CCS Concepts: • **Networked Music**;

ACM Reference Format:
Shan Jiang. 2021. Design and Implementation of a Latency-tolerant RTC Audio Framework for DAW.. 1, 1 (December 2021), 7 pages. https://doi.org/10.1145/1122445.1122456

The source code for this project is available at VIBEIO.

1 INTRODUCTION

This paper addresses the pronounced divide between professional music production and amateur music enthusiasts, a gap that has evolved with the advent of digital audio processing and the internet around the year 2000. This divide is evident in the stark contrast between the sophisticated audio engineering capabilities used in professional studios, such as 5.1 and 7.1 channels with 192kHz sampling rates, and the predominantly mono channel speakers through which most audiences experience music. Despite billions of aspiring singers worldwide, only a few of them have access to professional recording studios, highlighting a significant underutilization of advanced audio technologies by the general populace.

Historically, high-quality musical production was contingent upon professional singers possessing advanced musical performance skills. However, contemporary digital signal processing technology now allows a variety of post-production techniques accessible even to amateur singers. Yet, these advancements remain largely confined to professional studios, creating an accessibility gap in modern audio engineering services. Networking emerges as a key solution, democratizing access to mixing services. Several enterprises have ventured into offering mixing services tailored for non-professional singers[Alexander 1994], typically involving users recording a vocal clip and uploading it for processing [Rottondi et al. 2016], a model that meets the high demand but often lacks the sophistication required for high-quality music mixing.

In response to these challenges, this paper proposes a novel network-based, bidirectional, real-time audio mixing interface. This system is designed to bridge the gap between amateur singers and professional sound engineers in a digital recording environment, facilitating collaboration and synchronization between the participants. The paper explores the related work in this field, including systems like Jacktrip and Listento, and web-based platforms such as Soundtrap and Smule, each with their unique approaches and limitations.

The paper then delves into the design and implementation of the proposed system, detailing a user-centric approach that caters to the distinct needs of singers and sound engineers. The technical architecture encompasses mobile web applications, PC-based Electron applications, and VST/AU plugins, each playing a pivotal role in the system's functionality. A critical comparison of WebRTC and RTMP protocols highlights the choice of WebRTC for its low latency and peer-to-peer capabilities, essential for real-time audio communication. The integration of both lossless and lossy audio encoding is also explored, ensuring both audio fidelity and operational efficiency.

Furthermore, the paper analyzes the comparative advantages of native application development versus web-based development, guiding the project's strategic choices. Finally, the transition from Firebase to WebRTC Datachannel for remote effect control is discussed, underscoring the efficiency and suitability of WebRTC Datachannel for real-time updates in the system.

Organized into four main sections, the paper reviews existing network-based audio processing platforms, details the implementation of our bidirectional, real-time music transmission interface, evaluates the limitations of this implementation, and discusses potential future enhancements. This comprehensive approach aims to contribute valuable insights and innovations to the field of digital audio production and networking, bridging the gap between professional and amateur realms in the music industry.

2 RELATED WORK

2.1 Jacktrip

Jacktrip is an innovative solution designed to enhance the stability and integrity of audio signals in fluctuating network environments. One of its key strengths lies in its use of a buffer for jitter, which temporarily stores data to smooth out inconsistencies. Additionally, Jacktrip employs interpolation to address data drop issues, ensuring continuous audio transmission even in unstable conditions. A significant advantage of Jacktrip is its reliance on the UDP protocol. This choice is strategic, as UDP minimizes the high latency typically associated with packet retransmissions and acknowledgement responses, common in other protocols like TCP. As a result, Jacktrip offers a robust platform for reliable audio transmission in various network scenarios.

However, Jacktrip's dependency on the UDP protocol also introduces certain limitations. Unlike protocols such as RTSP[Schulzrinne et al. 1998], UDP does not incorporate flow control mechanisms, leading to a higher risk of packet loss. This aspect can be particularly problematic in environments where network reliability is not guaranteed. Another drawback of Jacktrip is its lack of consideration for NAT traversal issues. The system primarily operates in environments where public IP point-to-point connections are

Author's address: Shan Jiang, sjiang340@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA, 30332.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
XXXX-XXXX/2021/12-ART $15.00
https://doi.org/10.1145/1122445.1122456

available, limiting its applicability in scenarios where such network configurations are not feasible or available.

2.2 Listento

Listento represents a state-of-the-art, commercial-grade studio remote monitoring software, tailored for the professional audio industry. It offers a comprehensive tool chain that includes various critical features such as account management, connection analysis, and flexible audio stream encoding format and quality switching. These features make Listento an all-encompassing solution for studio professionals, providing them with the necessary tools to manage and monitor audio streams efficiently and effectively in a remote setting.

Despite its advanced capabilities, Listento poses certain challenges and limitations. Its design and functionalities target professional studio users, making it a more complex and specialized platform that may not be accessible or user-friendly for amateurs or non-professionals. Furthermore, Listento does not support mobile sound recording, a significant limitation in today's mobile-centric world. Users are required to purchase specific hardware and software to utilize Listento effectively, which can lead to a high cost of usage. This requirement for additional investments can be a barrier for smaller studios or individual professionals with limited budgets.

2.3 Soundtrap

Soundtrap emerges as a pioneering, web-based Digital Audio Workstation (DAW) that is designed to facilitate collaborative music creation. As Lind (2017) [Lind and MacPherson 2017] notes, its primary aim is to serve as a versatile platform for multiple creators, while also offering an accessible entry point for beginners in digital audio production. The tool is noteworthy for its ease of use, allowing for seamless operation within a web browser, and initially presents an image of high accessibility. However, empirical observations during testing revealed certain limitations in its advertised synchronous collaborative features. Notably, cross-platform collaborations frequently necessitated manual page refreshes to reflect changes on a collaborator's interface. Such inconsistencies can pose significant challenges for amateur musicians, potentially requiring multiple recording takes and causing considerable frustration. Additionally, despite its sleek and minimalist interface, Soundtrap retains the core complexities of a traditional DAW, which may not align with the expectations of novice users seeking simplicity in their digital audio explorations.

2.4 Smule

Smule represents an innovative approach to music production by simplifying the user's role to primarily singing, while it automates the mixing and provision of backing tracks. Wang (2009)[Wang et al. 2009] describes Smule as a platform predominantly utilized by amateur musicians who prefer straightforward musical performance without the complexities of advanced recording techniques. Initially, the concept of fully automated music production held significant appeal. However, it became apparent that Smule's focus is more on delivering an enjoyable, informal performance experience, akin to karaoke, rather than on facilitating professional-level music recording. The platform's collaborative features predominantly engage users in interactions with other amateurs, rather than with individuals aiming for high-level music production. This focus, while aligning with a recreational approach to music, ultimately limits Smule's applicability for scenarios requiring more sophisticated music production capabilities.

3 DESIGN AND IMPLEMENTATION

3.1 User Portraits and Product Requirements

In optimizing the interaction between amateur singers and professional sound engineers within a digital recording environment, the proposed software system is designed with stringent user-case limitations to maximize efficiency and effectiveness. The typical usage scenario envisages an amateur singer recording performances using a smartphone, while a sound engineer, proficient in sound processing and Digital Audio Workstation (DAW) operations, conducts real-time mixing remotely. This asymmetry in expertise necessitates a tailored approach to the software tools provided to each user.

For the singer, who possesses limited knowledge of sound processing and seeks to minimize engagement with technical aspects of recording, it is imperative to provide a user-friendly mobile application. This application must be capable of transmitting high-fidelity audio streams to a remote location while offering basic local sound effects to enhance the recording experience. The primary objective is to simplify the recording process for the singer, eliminating the need for complex interactions with sound effect selection and adjustments.

Conversely, for the sound engineer, the software needs to be robust and versatile. The engineer's setup must facilitate the reception of high-quality audio streams from the singer's location and ensure seamless integration with mainstream DAWs. This integration is crucial for enabling both real-time and post-production mixing, aligning with the engineer's professional workflow requirements.

Furthermore, effective collaboration between the singer and the engineer is a critical component of this system. To this end, an intercom channel is essential for bidirectional communication. This channel serves a dual purpose: it facilitates verbal communication and also transmits the mixed audio back to the singer. Such feedback is vital for the singer to listen to the completed tracks in real-time.

Lastly, considering the singer's limited expertise in sound effect adjustments, the system incorporates a remote sound control feature. This functionality enables the sound engineer to remotely manipulate sound effects within the singer's monitor channel. By transferring control to the engineer, the system ensures that the singer's recording benefits from professional sound processing without requiring direct involvement in the technical complexities of sound effect manipulation.

3.2 Program Utilization: Detailed Workflow

The operational procedure for the proposed program involves a structured sequence of steps designed to facilitate the collaboration between a singer and a mixer in a digital audio production environment. The standard workflow can be delineated as follows:

3.2.1 Initialization by Singer. The singer initiates the process by accessing the application's webpage. Here, they generate a unique Token, which serves as a secure key for establishing a connection.

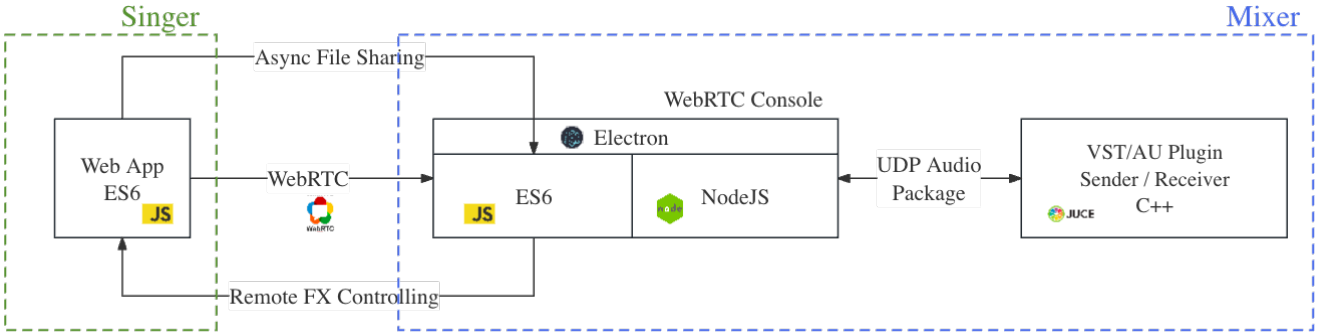

Fig. 1. Tech Stack Overview

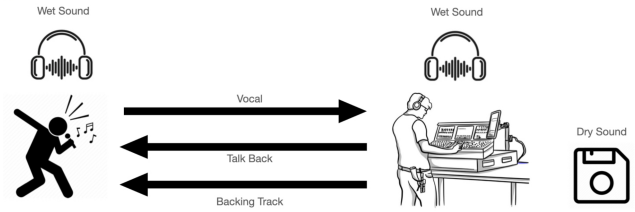

Fig. 2. A Typical Case for Music Recording

This Token is then shared with the Mixer to enable the subsequent linkage.

3.2.2 Connection Establishment by Mixer. Upon receiving the Token, the Mixer inputs it into the system to form a stable and secure connection with the Singer's setup.

3.2.3 Mixer's DAW Configuration. The Mixer proceeds to set up the Digital Audio Workstation (DAW). This involves creating specific Playback Tracks and integrating the Sender Virtual Studio Technology (VST)/Audio Unit (AU) plugin, a critical component for audio signal processing and transmission.

3.2.4 Importing Backing Tracks. Subsequently, the Mixer imports the necessary backing music into the Playback Tracks and initiates playback. This step is crucial for providing the musical foundation for the singer's performance.

3.2.5 Singer's Performance Commencement. As the backing track plays, the Sender transmits the audio signal to the Singer, who then begins their singing performance, synchronizing with the backing track.

3.2.6 Recording the Singer's Vocals. The audio of the Singer's vocals is transmitted back to the DAW at the Mixer's end. Here, the Mixer sets up new Recording Tracks and mounts the Receiver VST/AU plugin, which is essential for capturing and processing the incoming vocal audio.

3.2.7 Completion of Recording. The recording process is concluded once the Mixer stops the recording, signifying the completion of the audio capture phase.

Each step in this workflow is designed to optimize the collaboration between the singer and the mixer, ensuring a seamless integration of performance and production elements. The process underscores the importance of technological synchronization in modern digital audio workflows.

3.3 Technical Stack Overview

The described workflow integrates three primary modules, each encompassing a range of functionalities essential to the system's operation. These modules are:

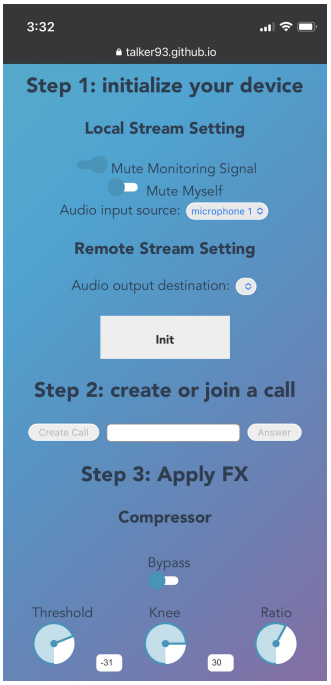

Fig. 3. The Mobile App

3.3.1 Mobile Web Application.

- Development Framework
  The application is developed as an ECMAScript 6 (ES6) web application, ensuring compatibility with contemporary web standards.
- Core Responsibilities
  Its primary functions include interfacing with Web Real-Time Communication (WebRTC)[Sredojev et al. 2015] on the PC Console, managing database communications, and handling Session Description Protocol (SDP) and Interactive Connectivity Establishment (ICE) information.
- Platform Compatibility
  Designed for universal browser compatibility, the application negates the need for platform-specific installations or rebuilds. Audio Processing Features: Leveraging Web Audio, it offers various native sound effects like Gain, Pan, Compressor, Convolver, and Infinite Impulse Response (IIR) Filter.
- Real-time Feedback
  The Web Audio framework also enables signal feedback routing, facilitating real-time detection of signal delays.
- User Interface Components
  Incorporation of NexusUI provides graphical representations of common audio controls (e.g., Knob, Slider, Switch) and audio analysis tools like Oscilloscope and Spectrogram.

3.3.2 *PC - Electron Application.*

- DAW Independence
  This application operates independently of a Digital Audio Workstation (DAW).
- Communication and Management: It handles communication with the Mobile Web App through WebRTC, manages database interactions, and oversees SDP and ICE information.
- Plugin Communication
  The app communicates with the VST/Audio Unit (AU) plugin using the User Datagram Protocol (UDP).
- Thread Management
  It simultaneously manages two distinct threads: the Server thread and the Browser thread.
  Server Thread Functions running on the Node.js compiler, the Server thread is tasked with UDP data transmission and reception.
  The Browser thread, utilizing the JavaScript compiler, is responsible for Web Audio rendering and maintaining the WebRTC instance.

3.3.3 *PC - VST/AU Plugin.*

- Plugin Variants
  The system includes both Sender and Receiver VST/AU plugins.
- Framework Utilization
  These plugins are developed using the JUCE C++ framework, a standard in high-performance audio software development.
- Advanced Audio Processing
  They implement hidden embedding and de-embedding of clock watermarks, alongside delay compensation for the Playback listening channel, ensuring synchronization and audio fidelity.

Each of these modules plays a pivotal role in the overall functionality of the system, providing a cohesive and efficient framework for real-time audio recording and processing. The integration of these technologies facilitates a seamless and intuitive user experience, catering to both amateur singers and professional sound engineers.

4 DISCUSSION

4.1 Comparison of WebRTC and RTMP Protocols

WebRTC and RTMP are both pivotal in the realm of audio and video streaming, yet they cater to distinct requirements. WebRTC, an open-source project, excels in enabling real-time communication directly in web browsers and mobile applications. Its standout feature is low latency, making it ideal for applications demanding real-time interactions, such as video conferencing or collaborative platforms. Additionally, WebRTC supports peer-to-peer connections, allowing direct streaming between users, which is crucial for applications where immediate interaction is necessary. This protocol is also known for its adaptability to fluctuating network conditions and wide support across modern web browsers, facilitating ease of access without additional plugins.

In contrast, RTMP, developed by Adobe, is traditionally used for high-quality streaming of audio and video content, particularly in scenarios like live broadcasting to large audiences. While it is known for its reliability and high-quality streaming, RTMP generally experiences higher latency compared to WebRTC and relies on media servers for streaming. This dependence can introduce additional complexity and cost.

4.1.1 *Why WebRTC is Preferred in This Program.*

- Real-Time Interaction The program requires real-time audio communication between singers and sound engineers. WebRTC's low latency is crucial for ensuring that audio streams are synchronized and interactive without perceptible delays.
- Peer-to-Peer Streaming WebRTC enables direct peer-to-peer connections, which is essential for the program's need for a direct link between the singer and the sound engineer without the intermediation of servers.
- Flexibility in Network Conditions Given that users might have varying network qualities, WebRTC's ability to adapt to different network conditions is beneficial. It ensures a consistent streaming experience even in less-than-ideal connectivity scenarios.
- Ease of Access and Use The program's design focuses on accessibility and ease of use. WebRTC's compatibility with modern web browsers allows users to participate without needing to download additional software or plugins, making it more user-friendly and accessible.

In summary, while RTMP offers robust streaming capabilities, especially for broadcasting to large audiences, WebRTC's advantages in terms of low latency, peer-to-peer connections, and browser compatibility make it more suitable for a program that requires real-time, interactive audio communication between individual users.

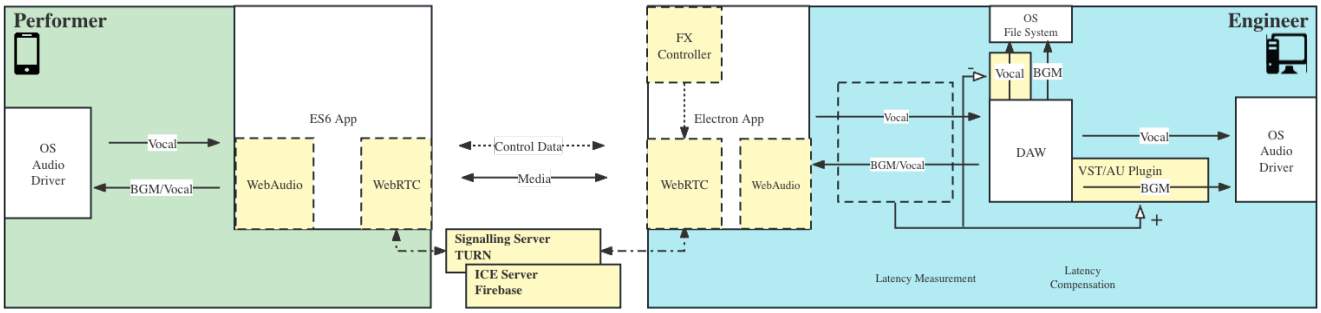

Fig. 4. Implementation Details

4.2 Integration of Lossless and Lossy Audio Encoding in the Program

In our program, a dual encoding strategy is implemented, utilizing both lossless and lossy audio encoding to cater to different aspects of the audio processing workflow.

4.2.1 *Lossless Audio Encoding.*

- Format and Transmission: The lossless audio is captured and stored in the WAV format. For transmission, we use the WebRTC Datachannel, which allows for asynchronous transfer of these high-fidelity audio files.
- Usage in Recording and Mixing: This lossless audio is crucial for the recording and mixing phases. At the conclusion of each recording session, the lossless audio file is automatically saved on the mobile device. Subsequently, it is synchronized to a designated folder on the PC side.
- Manual Integration by Mixer: The Mixer is then required to manually import these lossless audio files. This process ensures that the recordings retained in their pristine quality are accurately utilized during the mixing process.

4.2.2 *Lossy Audio Encoding.*

- Encoding and Storage: For the lossy audio component, we employ the Opus codec. This codec is renowned for its efficiency and is well-suited for real-time applications.
- Synchronization and Real-Time Access: The lossy encoded audio is synchronized using the standard WebRTC audio channel. It is then accessed in real-time within the monitoring channel of the Recording Tracks in the Digital Audio Workstation (DAW).

By implementing this combination of lossless and lossy audio encoding, the program achieves a comprehensive audio processing capability. The lossless encoding ensures the preservation of audio quality for critical tasks like recording and mixing. In contrast, the lossy encoding, with its reduced data rate, facilitates low-latency monitoring, crucial for real-time audio feedback during recording sessions. This dual approach not only maintains the integrity of the audio files but also ensures the operational efficiency required for effective monitoring and processing in a digital audio production environment.

4.3 Comparative Analysis of Development Approaches

In evaluating the technological development approach for both Peer A (singer's end) and Peer B (mixer's end), it is crucial to assess the implications of choosing between native application development and web-based development. This analysis will consider factors such as performance, stability, compatibility, and developmental complexity, which are pivotal in determining the efficacy and usability of the software for both user groups.

4.3.1 *Native Application Development.*

- Performance and Stability: Native applications typically offer enhanced software performance and stability. This is attributable to their direct interaction with the underlying operating system and hardware.
- Platform-Specific Development: A significant limitation is the platform-specific nature of native applications. Developing for iOS and Android requires distinct environments, leading to essentially two divergent development processes, as indicated by Devoe (2011)[DeVoe 2011] in their exploration of Objective-C. This bifurcation necessitates parallel development strategies, effectively doubling the workload.
- Developmental Challenges: Native development is inherently more complex, involving extensive use of system-specific APIs and low-level memory management commands. These aspects substantially escalate the development and debugging duration.

4.3.2 *Web-Based Development.*

- Cross-Platform Compatibility: In contrast, web-based development boasts superior compatibility across various platforms. A single web application can function across different operating systems and devices, eliminating the need for platform-specific development.
- Stability and API Concerns: However, this approach is not without drawbacks. The stability of web applications can be inconsistent, influenced by variables such as browser type, version, and underlying operating system constraints. API inconsistencies across different browsers or browser versions can lead to functional discrepancies.
- Memory and OS Limitations: Web applications often consume significant memory resources and face restrictions imposed

by operating systems, particularly in file access permissions. These factors can limit the functional capabilities and performance of the software.

4.3.3 Decision Rationale. Considering the project's objectives to rapidly develop a usable version within a strict usage scenario, web-based development emerges as the more pragmatic choice. Although it presents certain limitations in stability and performance, it aligns well with the need for rapid development and broad compatibility. Consequently, the decision to prioritize web-based development is a strategic choice, aimed at balancing the trade-offs between developmental speed, software functionality, and cross-platform usability. This approach effectively addresses the project's immediate needs while acknowledging the inherent constraints of each development methodology.

4.4 Comparative Analysis of WebRTC Datachannel and Firebase for Remote Effect Control

The development of a remote effect controller presented a choice between using WebRTC Datachannel and Firebase for control signal transmission.

4.4.1 Initial Use of Firebase.

- Method: Initially, Firebase's store-and-forward mechanism was employed, with peers monitoring and updating sound effect parameters in a shared database.
- Challenges: Despite the ease of implementation, this approach led to excessive database access, particularly with high-frequency updates (e.g., adjusting Gain values). This high volume of access risked triggering database protection mechanisms, even with attempts at implementing delayed averaging.

4.4.2 Shift to WebRTC Datachannel.

- Decision: Due to these limitations, the project transitioned to WebRTC Datachannel
- Advantages: This technology offered direct peer-to-peer communication, eliminating the need for third-party data forwarding and reducing database access concerns. Its real-time data transmission capability streamlined the update process for effect parameters.

In conclusion, while Firebase offered simplicity in setup, WebRTC Datachannel's efficiency in handling real-time, high-frequency updates made it the more suitable choice for the remote effect controller's needs.

5 CONCLUSION AND FUTURE WORK

5.1 Conslution

Based on the comprehensive examination of the issues and innovations in the realm of digital audio production, this paper has presented a novel network-based, bidirectional, real-time audio mixing interface. This interface is designed to bridge the significant divide between amateur singers and professional sound engineers in the digital recording environment. By offering a user-centric approach and integrating advanced technologies like WebRTC for low-latency communication and dual audio encoding strategies, the proposed system addresses the accessibility gap in modern audio engineering services.

The analysis of existing platforms like Jacktrip, Listento, Soundtrap, and Smule highlighted their unique strengths and limitations, providing a clear context for the necessity of this new system. While each of these platforms offers valuable contributions to the field of digital audio production, they also exhibit specific constraints that limit their effectiveness in bridging the gap between amateurs and professionals. This gap is particularly evident in aspects such as ease of access, real-time interaction, and the complexity of use.

In response, the proposed system leverages a combination of mobile web applications, PC-based Electron applications, and VST/AU plugins to create a seamless and intuitive user experience. The system emphasizes real-time collaboration, allowing for efficient and effective interaction between singers and sound engineers. The detailed workflow and technical stack overview illustrate a well-considered approach that caters to the distinct needs of both user groups.

The comparative analysis of WebRTC and RTMP[Aloman et al. 2015] protocols, as well as the integration of lossless and lossy audio encoding, underscore the technical sophistication of the proposed solution. These choices reflect a deep understanding of the requirements for real-time, high-quality audio communication in various network conditions. Additionally, the decision to focus on web-based development, considering the trade-offs between performance, stability, and cross-platform compatibility, demonstrates a strategic approach to software design and implementation.

In conclusion, this paper contributes significantly to the field of digital audio production by presenting a novel solution that effectively bridges the gap between amateur singers and professional sound engineers. The proposed system not only addresses the current limitations in the field but also opens new avenues for collaboration and innovation in digital music production. Future enhancements and iterations of this system could further democratize high-quality music production, making it more accessible and inclusive for a broader range of users.

5.2 Future Work

(1) Comprehensive Evaluation of the System
Future work should involve a more extensive and systematic evaluation of the proposed audio mixing interface. This would include user testing with a larger, more diverse group of participants, encompassing both amateur singers and professional sound engineers. The evaluation should focus on usability, quality of audio transmission, latency, and overall user satisfaction. Gathering qualitative and quantitative data through surveys, interviews, and real-time usage metrics would provide valuable insights into the system's performance and areas for improvement. Additionally, comparative studies with existing platforms could be conducted to benchmark the system's effectiveness against industry standards.
(2) Development of Advanced User Account Management
To enhance user experience and security, the development of a sophisticated user account management system is crucial.

This system would facilitate user profile creation, management, and authentication. Features like personalized settings, history of sessions, and preferences for audio mixing could be incorporated. A more robust account management system would also improve data security and privacy, ensuring safe and secure storage of user data and recordings. Integration with social media platforms for easy login and sharing capabilities could also be explored.

(3) Implementation of a Video Call Function
Incorporating a video call function into the platform would significantly enhance the collaborative aspect of the system. This addition would allow singers and sound engineers to communicate visually in real-time, further bridging the gap in remote music production. The video call feature should be optimized for low latency and synchronized with the audio stream to ensure a seamless experience. Additionally, features like screen sharing, session recording, and annotation tools could be added to facilitate a more interactive and productive collaboration.

(4) Development of a Separate TURN/STUN Server Program
To improve network reliability and connectivity, the development of a dedicated TURN/STUN server program is recommended. This server would facilitate better handling of NAT traversal and improve connection stability between peers, especially in complex network environments. Developing a custom TURN/STUN[Rosenberg et al. 2008] server would allow for greater control over network traffic and optimization for audio data transmission. It would also reduce dependencies on third-party services, potentially enhancing the overall reliability and scalability of the system.

REFERENCES

Peter J Alexander. 1994. New technology and market structure: Evidence from the music recording industry. *Journal of Cultural Economics* 18, 2 (1994), 113–123.

Alexandru Aloman, AI Ispas, Petrica Ciotirnae, Ramon Sanchez-Iborra, and Maria-Dolores Cano. 2015. Performance evaluation of video streaming using MPEG DASH, RTSP, and RTMP in mobile networks. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 144–151.

Jiva DeVoe. 2011. *Objective-c*. Vol. 29. John Wiley and Sons.

Fredrik Lind and Andrew MacPherson. 2017. Soundtrap: A collaborative music studio with Web Audio. (2017).

Jonathan Rosenberg, Rohan Mahy, Philip Matthews, and Dan Wing. 2008. *Session traversal utilities for NAT (STUN)*. Technical Report.

Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. 2016. An overview on networked music performance technologies. *IEEE Access* 4 (2016), 8823–8843.

Henning Schulzrinne, Anup Rao, and Robert Lanphier. 1998. *Real time streaming protocol (RTSP)*. Technical Report.

Branislav Sredojev, Dragan Samardzija, and Dragan Posarac. 2015. WebRTC technology overview and signaling solution design and implementation. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 1006–1009.

Ge Wang, Georg Essl, Jeff Smith, Spencer Salazar, Perry R Cook, Rob Hamilton, Rebecca Fiebrink, Jonathan Berger, David Zhu, Mattias Ljungstrom, et al. 2009. Smule= Sonic Media: An Intersection of the Mobile, Musical, and Social.. In *ICMC*.