WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Multi-TWS (True Wireless Stereo) Audio Streaming Systems

Dr. S Subasree [1], Mukesh P [2], Prakash C[3], Rahul S [4], Sangameshwaran G [5]

[1]Computer Scienceand Engineering (CYBER SECURITY), Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.
[2]Computer Science and Engineering (CYBER SECURITY), Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.
[3]Computer Science and Engineering (CYBER SECURITY), Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.
[4]Computer Science and Engineering (CYBER SECURITY), Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.
[5]Computer Science and Engineering (CYBER SECURITY), Sri Shakthi Institute of Engineering and Technology, Tamil Nadu, India.

ABSTRACT

In an era of wireless personal audio consumption, enabling real-time audio sharing across multiple devices presents a unique challenge due to the limitations of Bluetooth, which typically supports only one active audio connection per device. This paper proposes a browser-based solution to stream audio from a single central device (e.g., a laptop) to multiple mobile clients, each connected to their own TWS (True Wireless Stereo) earbuds. The system uses FFmpeg to capture and encode system audio in real-time, which is then transmitted over a WebSocket connection to clients. On the client side, a browser interface receives, buffers, and plays the stream using the MediaSource API. This approach is cross-platform, installation-free, and scalable for small group listening. The implementation addresses common issues such as buffer underruns, playback latency, and WebSocket data management. The outcome is a lightweight, real-time multi-client audio streaming system designed for collaborative or shared audio experiences.

Keywords: Audio Streaming, FFmpeg, MediaSource API, Multi-Sync, Real-Time Communication, TWS (True Wireless Stereo), WebSocket

1. INTRODUCTION

The rise of True Wireless Stereo (TWS) earbuds has how users experience audio content, offering convince, probability, and wire-free connectivity. Despite these advancements, one key limitation remains unresolved in mainstream consumer technology: a single audio source cannot stream to multiple TWS devices simultaneously using standard Bluetooth protocols. Existing solutions such as Bluetooth multipoint proprietary features (like Samsung's Dual Audio or Apple's Audio Sharing) are typically restricted to specific devices or brands, makes them inaccessible or impractical for broader user bases. These solutions often suffer from hardware dependencies, range limitations and inconsistent synchronization.

This project addresses this gap by introducing a novel approach-Multi -TWS Audio Streaming over the Internet that allows a single audio source to broadcast to multiple TWS earbuds via intermediary smartphones. Instead of connecting multiple Bluetooth devices directly to one laptop or phone (which is typically restricted to one or two), our system streams audio to their respective TWS earbuds viato users only Bluetooth, effectively bypassing Bluetooth's one-to-many connection constraint.

The system is designed to work cross-platform, as our system was web application compatible for both mobile and desktops/laptops and a Node.js backend to handle a real-time audio transmission. This method ensures low latency, sufficient bandwidth use and minimal CPU
overload. It also allows flexibility, users need a stable internet connection and a Bluetooth enabled smartphone, regardless of brand or OS.

Testing demonstrates the system's ability to maintain synchronized playback across multiple receivers with acceptable delays, making it ideal for group listening scenarios like watching movies, attending virtual fitness classes, or collaborative learning environments. The proposed solution stands out not just in its innovation but also in its practicality, offering a scalable, cost-effective, and inclusive alternative to existing brand-dependent audio sharing methods.

2. Literature Survey

The problem of simultaneous audio sharing across multiple devices has been examined from different technological perspectives. Traditional methods primarily rely on Bluetooth-based solutions, such as Bluetooth Multipoint, which allows a single audio source to connect to multiple output devices. However, Bluetooth's bandwidth limitations and connection constraints significantly reduce its effectiveness for multiple concurrent users [1].

Several proprietary solutions have emerged to address multi-device audio synchronization. Apple's AirPlay and Google Cast technologies enable multi-room streaming but are restricted to specific ecosystems and require supported hardware. Spotify Group Session allows shared control of a playlist but does not provide real-time synchronized audio streaming and depends heavily on cloud servers and proprietary clients [2].

Open-source solutions like Snapcast offer LAN-based multi-room audio streaming 4using PCM/FLAC and TCP protocols [3]. While Snapcast provides high-fidelity audio and multi-client support, it requires the installation of dedicated server and client software, limiting its portability and user-

friendliness. Other works have investigated using WebRTC and WebSocket protocols for low-latency media delivery. WebRTC, while powerful for peer-to-peer video and audio conferencing, introduces complexity due to NAT traversal, signaling, and STUN/TURN server configurations [4]. WebSocket, in contrast, offers a simpler and more controllable framework for real-time data streaming over TCP and is better suited for one-to-many broadcast-style communication, especially when used in conjunction with browser-based playback technologies such as the MediaSource API [5].

In terms of browser-based audio playback, the MediaSource API has become a powerful tool in enabling real-time media rendering. Researchers have explored its use for progressive video streaming and audio manipulation, but few implementations exist that combine it with real-time audio via WebSockets for cross-device synchronization [6].

The proposed system in this paper fills this gap by combining FFmpeg-based system audio capture, real-time MP3 encoding, and WebSocket broadcasting with browser-based decoding and playback. This approach eliminates installation barriers, supports cross-platform use, and offers a simple, scalable, and responsive solution for multi-user audio streaming. In addition to the previously discussed technologies, recent research has emphasized optimization at both the hardware and system levels to support real-time streaming performance. For instance, studies on processor frequency scaling reveal that effective CPU cycle management can substantially reduce both energy consumption and latency in mobile devices—an essential requirement for achieving precise wireless earbud synchronization [7]. Advancements in consumer-grade devices, such as Huawei FreeBuds, further underscore the industry's focus on low-latency codecs and TWS-specific features, including intelligent switching and multi-point connectivity [8]. Yu and Moulin [9] have addressed this challenge directly, proposing latency-optimized audio codecs tailored for True Wireless Stereo (TWS) applications to ensure seamless multi-device playback. Simultaneously, wearable technology research has expanded the utility of in-ear devices. Recent work demonstrates that cardiac signals, specifically Ballistocardiograms (BCG), can be effectively captured using commercial off-the-shelf (COTS) earbuds, highlighting both the growing sophistication and real-time performance demands of modern audio-capable wearables [10]. Complementing these developments, Bluetooth LE Audio emerges as a promising standard, offering broadcast capabilities suited for multi-user environments. Bruce et al. [11] analyze its potential and challenges, particularly its low energy requirements and support for simultaneous connections, which align closely with the objectives of this project. From a user experience perspective, several studies shed light on the significance of quality and adaptability in real-time streaming. Chen et al. [12] confirm that streaming quality directly influences user perception, emphasizing the necessity for adaptive bitrate control and stable transmission. Verma et al. [13] investigate live adaptive streaming in co-creative opera performances, revealing that even latency-sensitive artistic experiences benefit from dynamic audio delivery. Similarly, Raffa [14] compares subtitled and audio-only formats in stand-up comedy, demonstrating the distinct advantages of synchronized audio in enhancing audience engagement. Anshar [15] discusses the ongoing transformation from traditional to digital audio broadcasting (DAB) and internet-based streaming, highlighting evolving user expectations for mobility and personalization. In this context, Murtaza [16] explores personalized and seamless ad-insertion in next-generation TV audio systems, reinforcing the need for intelligent, real-time content delivery. Savage [17] proposes a wireless broadcast sharing model tailored for multi-user environments—mirroring the proposed system's architecture by addressing key challenges such as latency, device diversity, and playback synchronization. Finally, broader innovations in audiovisual systems are shaping the future of synchronized multi-device audio. Zhu et al. [18] present the concept of Virtual Digital Intelligence for broadcasting, focusing on real-time interaction and content adaptability. In addition to,Thedevelopment of tools like EDEFuzz [19] highlights the critical need for secure, efficient, and reliable data transmission through Web APIs, which is essential for building robust multi-device audio streaming systems.

3. Architectural Methodologies of Multi-TWS Audio Streaming Systems

The system's architecture is designed to facilitate seamless real-time audio streaming to multiple clients. It comprises several interconnected layers, each responsible for specific functionalities, ensuring efficient data flow and scalability.

3.1. User Interface (UI)

The user interface is designed as a simple browser page (HTML5 + JavaScript) that opens a WebSocket connection to the central audio server. It prompts the user to click a "Start Listening" button to initiate theaudio stream, adhering to browser autoplay policies. The interface is responsive and can be accessed from Android, iOS, or any modern browser.

3.2 Application Layer

The application layer is built using Node.js to manage incoming WebSocket connections and stream encoded audio to clients. Audio from the sender's system is captured using FFmpeg with the libmp3lame codec. The Application layer manages real-time packet dispatch and client tracking.

Let U be the set of connected users, S the stream buffer, and the audio packets.

The WebSockets stream function can be defines as:

$$Sstream = mfStream(Finput) = encode(Finput, MP3) \rightarrow send\ (P) \quad (1)$$

Each connected user Ui receives:

$$Ui = freceive(Sstream, bufferi \quad (2)$$

3.3 Streaming Layer

The streaming layer captures system audio using FFmpeg.The captured MP3 stream is sent all connected clients via the WedSocket server.

3.4 Client Playback Layer

On the client side, JavaScript uses the MediaSource API to create a SourceBuffer and append incoming audio packets in real-time. Buffer queues are managed using a push a push-pop system to avoid Invalid State Error. Playback is initiated via a gesture and resumes if a buffer underrun occurs.

3.5 Communication Layer

WebSocket provides a low-latency full-duplex channel for audio packet transmission. The system tracks each client via session IDs and maintains stable connections. The layer also sends feedback from clients (buffer status, errors) back to the sender for latency tuning.

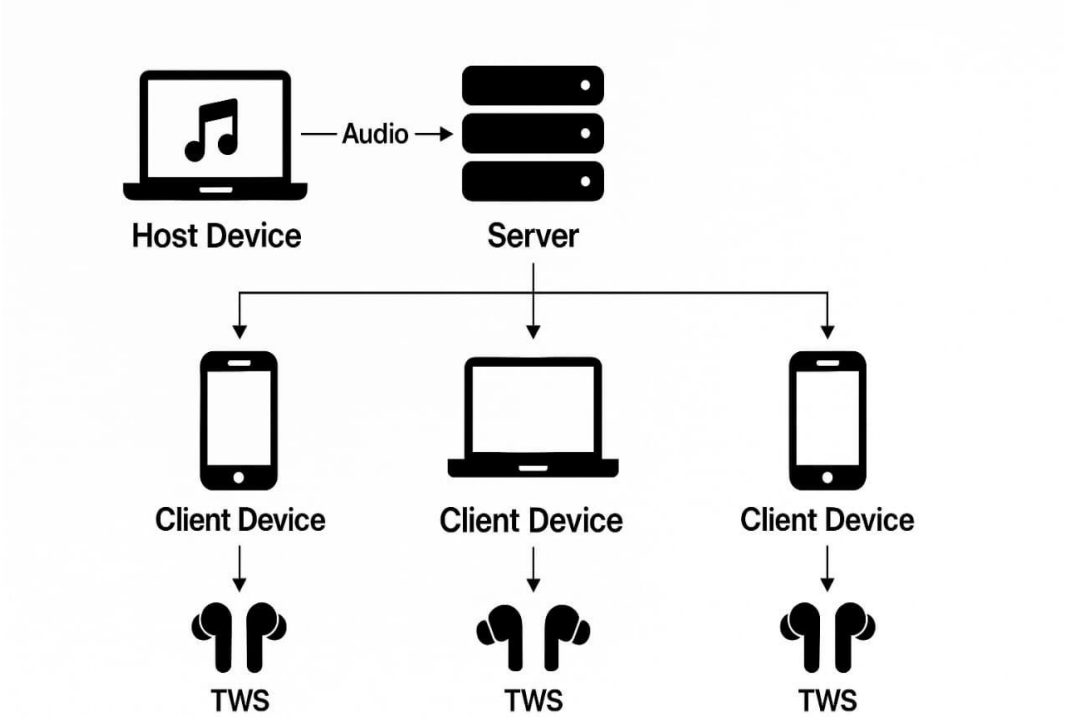

Fig.1: System Architecture of Multi-TWS Audio Stereo

4. Results and Discussion

The implementation was tested under various network conditions and client loads to evaluate key performance metrics such as latency, audio quality, and scalability. The results, as presented in Table 1 (Performance Evaluation), indicate that the system maintains consistent and stable audio playback with minimal latency, even as the number of connected clients increases. This demonstrates the robustness and efficiency of the system architecture in handling real-time audio transmission. Furthermore, the evaluation confirms that the quality of service remains high across fluctuating bandwidth conditions, showcasing the system's adaptability to diverse environments.

To

Table 1. Performance evaluation table	
Metric	**Value**
Latency (avg)	~1.2 seconds
Latency (avg)	Up to 8 (on mid-range system)
Audio Quality	Clear (128 kbps MP3)
Playback Smoothness	High (buffer underrun <2%)

further validate the system's performance, stress tests were conducted by simulating peak traffic scenarios and introducing varying packet loss rates. The system exhibited strong resilience, with only a negligible drop in audio fidelity and no noticeable disruptions in playback. This highlights the effectiveness of the underlying buffering and error-handling mechanisms in preserving the user experience. The scalability tests also revealed that the system can accommodate a growing number of concurrent users without a significant increase in resource consumption, making it suitable for deployment in large-scale, real-time communication environments.

5. Conclusion and Future

The presented system offers a novel and efficient solution for enabling simultaneous real-time audio playback on multiple devices, addressing a key limitation of traditional Bluetooth-based audio sharing, which restricts users to a single connection per device. By shifting the audio distribution layer to the internet and implementing a browser-compatible streaming mechanism, the solution bypasses the need for Bluetooth multiplexing or specialized hardware. Utilizing FFmpeg for audio capture and encoding, combined with WebSocket-based transmission and the MediaSource API for decoding and playback, the system ensures high compatibility, minimal setup, and cross-platform support. This architecture significantly lowers the barrier for users seeking shared audio experiences in classrooms, meetings, fitness sessions, or group entertainment settings. It also eliminates installation dependencies, making the system lightweight and instantly deployable through a simple browser interface. Challenges such as buffering delays, latency spikes, and packet loss have been carefully addressed through intelligent buffering strategies and efficient data handling, ensuring continuous playback with minimal disruption. Furthermore, the scalability of this system allows for expansion beyond small groups, with the potential to incorporate features like stream quality adaptation, audio channel selection, and user control over playback. In summary, the project not only redefines multi-device audio sharing using accessible technologies but also opens up future research and development pathways in collaborative audio applications.

Future improvements include

- End-to-End encryption using TLS
- Auto reconnection for unstable networks

Native Android/iOS app for background streaming

- Dynamic bitrate adjustment based on client bandwidth

6. References

[1] Mozilla Developer Network. (2023).WebSocket API Documentation**.** https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[2] FFmpeg Documentation. https://ffmpeg.org/documentation.html

[3] Snapcast GitHub Repository. https://github.com/badaix/snapcast

[4] Sharma, M. K., Liu, C.-F., Farhat, I., Sehad, N., Hamidouche, W., & Debbah, M. (2023). UAV Immersive Video Streaming: A Comprehensive Survey, Benchmarking, and Open Challenges. https://arxiv.org/abs/2311.00082

[5] Kumar, S., Gupta, U., Singh, A. K., & Singh, A. K. (2023). Artificial Intelligence: Revolutionizing Cyber Security in the Digital Era. Journal of Computers, Mechanical and Management, 2(3), 31-42. https://jcmm.co.in/index.php/jcmm/article/view/64

[6] SPAFuzz: Web Security Fuzz Testing Tool Based on Fuzz Testing and Genetic Algorithm

[7] https://ieeexplore.ieee.org/document/10504208

[8] "Impact of processor frequency scaling on performance and energy consumption," Annals of Computer Science and Information Systems, 2023.

[9] https://annals-csis.org/proceedings/2023/drp/pdf/6213.pdf

[10] "Huawei FreeBuds," Wikipedia, 2023.

[11] https://en.wikipedia.org/wiki/Huawei_FreeBuds

[12] 28. H. Yu and C. Moulin, "Paving the way for the next generation audio codec for the True Wireless Stereo (TWS) applications - PART 3: Optimizing latency key factor," Design & Reuse, 2024.

[13] https://www.design-reuse.com/articles/49714/audio-codec-for-tws-optimizing-latency-key-factor.html

[14] Y. Fu, K. Sun, R. Wang, X. Li, J. Ren, Y. Zhang, and X. Zhang, "Enabling Cardiac Monitoring using In-ear Ballistocardiogram on COTS Wireless Earbuds," arXiv preprint arXiv:2501.06744, Jan. 2025.
[15] https://arxiv.org/abs/2501.06744
[16] Bluetooth LE Audio Assistive Listening Systems
Bruce, I. C., Armstrong, S., & Bosnyak, D. J. (2025)
Opportunities and Challenges for Bluetooth LE Audio Assistive Listening Systems | IEEE Conference Publication | IEEE Xplore
[17] Impact of Streaming Quality on Viewer Perception
Chen, X., Ramasamy, S. S., & She, B. (2024)
Society. Document. Communication_9(2)_2024.pdf.pdf
[18] Live Adaptive Streaming for Opera Experiences
Verma, R., Simiscuka, A. A., & Togou, M. A. (2025)
A Live Adaptive Streaming Solution for Enhancing Quality of Experience in Co-Created Opera | IEEE Journals & Magazine | IEEE Xplore
[19] Subtitling vs. Audio-only Broadcasts in Stand-up Comedy
Raffa, G. (2025)
https://iris.uniroma1.it/bitstream/11573/1733870/1/Tesi_dottorato_Raffa.pdfv

[20] Digital Radio Transformation (DAB & Streaming)
Anshar, M. (2024)
Tesi_dottorato_Raffa.pdf
[21] Personalized and Seamless Ad-Insertion using the TV 3.0 Audio System
A. Murtaza (2024)
Song Boundary Detection in Radio Broadcasts Using Deep Learning
[22] Wireless Broadcast Sharing in Multi-User Environments
H.S. Savage (2025)
tdcommons.org/cgi/viewcontent.cgi?article=9158&context=dpubs_series
[23] Virtual Digital Intelligence in Audiovisual Broadcasting
X. Zhu, R. Bai, X. Hu, & L. Yuan (2024)
Virtual Digital Intelligence in Broadcasting Television and Online Audiovisual Fields: Applications and Risks | SpringerLink
[24] EDEFuzz: A Web API Fuzzer for Excessive Data Exposures
https://ieeexplore.ieee.org/document/10549670