IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, December 2025
Impact Factor: 7.67

Secure and Scalable Peer-to-Peer File Distribution for Smart Campus Laboratories

Harsh Hanumant Shinde, Saurabh Prakash, Swaraj Rahul Bhondve
Sakshi Maruti Bansude, Pushkar Rajendra Patil
Department of Computer Engineering
Pimpri Chinchwad Polytecnic, Pune, India
harshshinde142@gmail.com, saurabhprakash7777@gmail.com, swarajbhondve97@gmail.com
bansudesakshi9@gmail.com, pushkarpatil220@gmail.com

Abstract: *In the context of modern web-based communication systems, efficient and secure file sharing remains a significant challenge due to the limitations of centralized server-dependent architectures. Traditional file transfer mechanisms often suffer from high latency, bandwidth overhead, storage dependency, and increased security risks, especially when handling large files in real-time environments. Conventional client-server models are increasingly inadequate to meet the growing demand for fast, private, and scalable data exchange. This study presents a secure peer-to-peer file sharing system based on WebRTC technology, enabling direct browser-to-browser file transfers without relying on intermediate server storage. Real-time signaling and session coordination are achieved using Socket.io, while encrypted WebRTC data channels ensure secure and reliable transmission. The system incorporates role-based authentication and access control using JSON Web Tokens to enhance security and user management. Furthermore, real-time user presence tracking, administrative broadcast functionality, and transfer history logging are implemented to improve system usability and monitoring. The proposed approach demonstrates improved performance, reduced latency, and enhanced security compared to traditional file sharing techniques.*

Keywords: Peer-to-peer file sharing, WebRTC, Real-time communication, Secure data transfer, Socket.io, Browser-based systems

I. INTRODUCTION

Peer-to-peer (P2P) file sharing has become an essential component of modern digital communication, enabling users to exchange data directly without relying on centralized storage systems. With the increasing demand for real-time collaboration and large file transfers, traditional client-server–based file sharing solutions face significant challenges related to latency, bandwidth consumption, server overload, and privacy concerns. These limitations reduce efficiency and scalability, particularly in environments where fast and secure data exchange is required.

Conventional file transfer systems depend heavily on centralized servers to store and relay files between users. Such architectures introduce single points of failure, increase infrastructure costs, and expose sensitive data to potential security breaches. Additionally, server-based transfers often result in slower performance due to congestion and limited bandwidth, making them unsuitable for real-time and high-volume file sharing scenarios.

To overcome these challenges, recent research has focused on decentralized communication models using browser-native technologies. Web Real-Time Communication (WebRTC) enables direct browser-to-browser data exchange through encrypted peer-to-peer connections, eliminating the need for intermediate file storage[1]. When combined with real-time signaling mechanisms such as Socket.io and secure authentication techniques, WebRTC provides an efficient foundation for low-latency, secure, and scalable file sharing systems[1]. This paper presents the design and implementation of a WebRTC-based peer-to-peer file sharing system that facilitates real-time file transfer, user presence tracking, and role- based access control while enhancing performance, privacy, and reliability compared to traditional file sharing solutions.

1.1 Background

With the rapid advancement of web technologies, file sharing has evolved into a fundamental requirement for real-time communication and collaboration across distributed systems. Conventional file sharing solutions predominantly rely on centralized server-based architectures, where files are uploaded, stored, and redistributed to recipients. While such approaches are widely adopted, they introduce several challenges including increased latency, excessive bandwidth usage, scalability limitations, and heightened risks to data privacy and security[1]. These limitations become increasingly evident as file sizes grow and users demand faster, real-time data exchange.

Centralized file transfer systems also create single points of failure and dependency, making them vulnerable to service disruptions and security breaches. Server-side file storage exposes sensitive data to potential unauthorized access, while network congestion can significantly degrade performance during peak usage. Recent developments in browser- based technologies have introduced peer-to-peer (P2P) communication as a promising solution to overcome these limitations. Web Real-Time Communication (WebRTC) enables direct, encrypted browser-to-browser data transmission, eliminating the need for intermediate file storage and reducing latency[2]. In parallel, real-time signaling frameworks such as Socket.io facilitate efficient session negotiation, user discovery, and connection management [1][2]. When combined with secure authentication mechanisms, these technologies offer a decentralized and scalable approach to file sharing.

This research is motivated by the need to design a real-time, secure, and server-independent file sharing system using modern web technologies. By leveraging WebRTC for peer-to-peer data transfer, Socket.io for signaling and presence management, and role-based authentication for secure access control, the proposed system aims to address the performance, security, and scalability challenges associated with traditional file sharing methods[2] . The study contributes to the development of efficient decentralized file transfer systems capable of adapting to modern communication requirements.

1.2 Contribution of this work

This paper presents a comprehensive approach to secure and real-time peer-to-peer file sharing by leveraging WebRTC and modern web communication technologies to address the limitations of traditional centralized file transfer systems. Existing file sharing mechanisms, which rely heavily on server-based storage and forwarding, exhibit reduced performance, higher latency, scalability issues, and increased security risks, particularly when handling large files and real- time transfers. These limitations highlight the need for decentralized solutions capable of supporting efficient and secure data exchange.

The proposed system contributes to the field by introducing a browser-based peer-to-peer file sharing platform that eliminates server-side file storage and enables direct browser-to-browser communication. By utilizing WebRTC for encrypted data transmission and Socket.io for real-time signaling and connection management, the system ensures low-latency file transfers and reliable session coordination. Secure access control is enforced through role-based authentication mechanisms, enhancing system integrity and user management[3] . Additionally, real-time user presence tracking and administrative broadcasting capabilities are incorporated to improve coordination and operational efficiency.

The key contributions of this work are summarized as follows:

- Decentralized File Transfer Architecture: Implementation of a WebRTC-based peer-to-peer model that enables direct file transfer without intermediate server storage, reducing latency and bandwidth overhead[3].
- Real-Time Signaling and Presence Management: Integration of Socket.io to support live user presence updates, connection negotiation, and session synchronization.
- Secure Role-Based Access Control: Use of authentication and authorization mechanisms to ensure controlled access and secure file exchange between users.
- Administrative Broadcasting and Transfer Logging: Support for broadcasting files to multiple recipients and maintaining detailed transfer histories for monitoring and auditing purposes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, December 2025
Impact Factor: 7.67

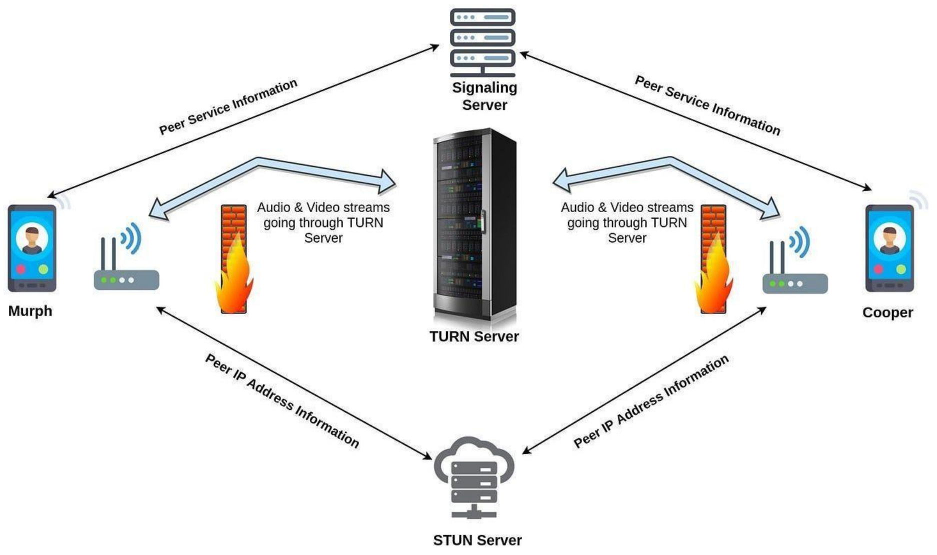

Signaling Server
Peer Service Information
Peer Service Information
Audio & Video streams going through TURN Server
Audio & Video streams going through TURN Server
Murph
TURN Server
Cooper
Peer IP Address Information
Peer IP Address Information
STUN Server

Fig.1: WebRTC Peer-to-Peer Communication Architecture Using STUN and TURN Servers

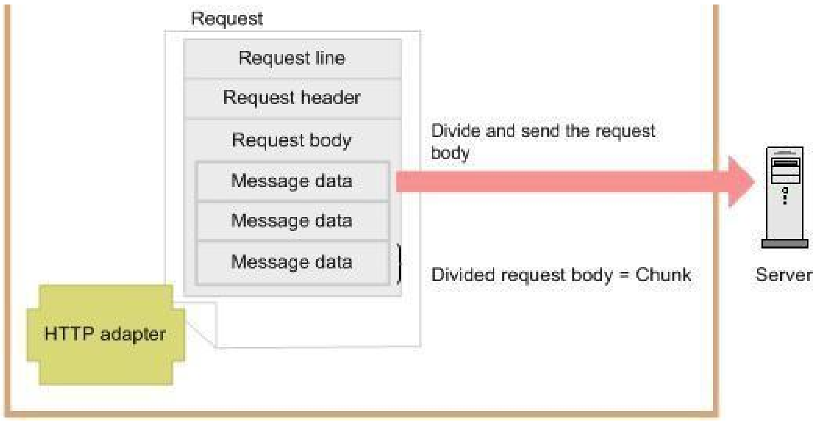
II. PROPOSED METHODOLOGY

To address the limitations of traditional centralized file sharing systems, this research proposes a secure, real-time, and decentralized peer-to-peer (P2P) file sharing framework built using modern web technologies. The proposed methodology focuses on eliminating server-side file storage, reducing latency, and enhancing security through direct browser-to-browser communication. The system is designed to be scalable, reliable, and suitable for real-time file transfer scenarios[4]. A detailed overview of the system architecture, operational workflow, and core components is presented below.

2.1 System Architecture:

The proposed system adopts a hybrid client–server architecture in which the server handles authentication, signaling, and coordination, while actual file transfers occur directly between peers using peer-to-peer communication. The main architectural components are summarized as follows:

- Authentication and Access Control: User authentication and session management are implemented using JSON Web Tokens (JWT) to enforce secure, role-based access to system functionalities.
- Real-Time Signaling and Presence Management: Socket.io is used for real-time communication, enabling user presence tracking and the exchange of WebRTC signaling messages such as session offers, answers, and ICE candidates.
- Peer-to-Peer Communication Layer: WebRTC establishes encrypted browser-to-browser data channels that enable direct file transfer without intermediate server storage, reducing latency and bandwidth overhead[4] .
- File Chunking and Transmission: Files are divided into fixed-size chunks and transmitted sequentially over WebRTC data channels, ensuring reliable delivery and efficient resource utilization.
- Administrative Broadcasting Module: Authorized users can broadcast files to multiple recipients by establishing independent peer-to-peer connections with each receiver.
- Transfer Logging and History Management: File transfer metadata, including sender, receiver, file details, timestamps, and status, is recorded in a database for monitoring and auditing purposes.



- Figure 2. Chunk-based file transmission over WebRTC data channels

2.2 Operational Workflow

A workflow describing the end-to-end operation of the proposed peer-to-peer file sharing system is as follows:

- User Authentication and Connection: Users authenticate securely using role-based credentials. Upon successful login, the client establishes a real-time connection with the server to enable presence tracking and signaling.
- User Discovery and Selection: The system retrieves and displays a list of currently online users in real time. The sender selects a target recipient or initiates a broadcast based on assigned privileges.
- Signaling and Connection Setup: WebRTC signaling messages, including session offers, answers, and ICE candidates, are exchanged via Socket.io to establish a secure peer-to-peer connection between communicating parties [5].
- Peer-to-Peer File Transfer: Once the connection is established, the selected file is segmented into chunks and transmitted directly between browsers over encrypted WebRTC data channels, ensuring low latency and secure delivery [4][5].
- Transfer Completion and Logging: After successful transmission, the file is reassembled and downloaded by the receiver. Transfer metadata is recorded in the system for history tracking and administrative monitoring.

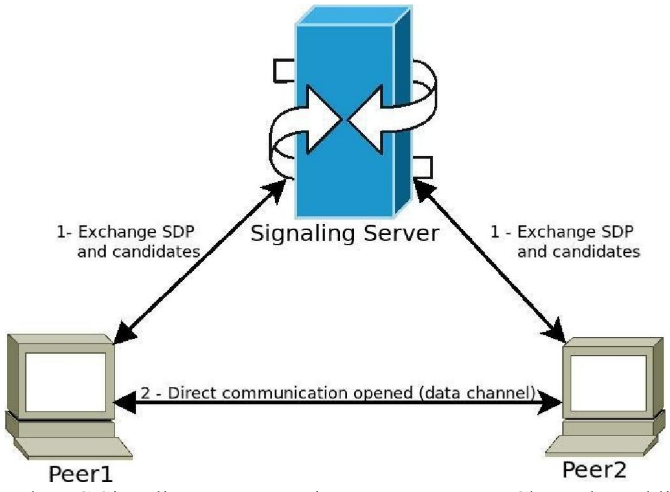

Signaling Server
1- Exchange SDP and candidates
1 - Exchange SDP and candidates
2 - Direct communication opened (data channel)
Peer1
Peer2

Fig. 3: WebRTC Signaling Process and Peer-to-Peer Data Channel Establishment

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, December 2025
Impact Factor: 7.67

2.3 Additional Technical Approaches

- Dynamic Peer and Connection Management: The system dynamically creates, maintains, and terminates WebRTC peer connections based on real-time user availability and transfer state. Socket.io events are used to synchronize connection metadata, allowing efficient handling of concurrent transfers and minimizing stale or orphaned connections.
- Chunk-Based Transmission Control: Files are segmented into fixed-size chunks prior to transmission. Chunk delivery is tracked sequentially over WebRTC data channels, enabling progress monitoring, transfer recovery, and controlled memory usage during large file transfers.
- Low-Latency Signaling Optimization: Signaling messages such as offers, answers, and ICE candidates are exchanged using lightweight Socket.io events[4][5]. This minimizes signaling overhead and ensures rapid peer connection establishment even under high concurrency.
- End-to-End Encrypted Data Channels: WebRTC's DTLS-SRTP encryption is leveraged to provide secure end-to- end communication. Since files are transmitted directly between peers without server-side storage, data exposure risks are significantly reduced[5] .
- Role-Aware Transfer Logic: Role-based control logic is enforced at both the signaling and application layers. Administrative users can initiate multi-peer broadcast transfers, while standard users are restricted to one-to-one communication, ensuring controlled and predictable system behavior.

The proposed methodology is designed to address key technical challenges such as centralized bottlenecks, inefficient bandwidth utilization, and limited real-time scalability. By combining optimized WebRTC data channels with real-time signaling and secure access control, the system achieves efficient, low-latency file transfers while maintaining strong security guarantees. The technical design ensures practical deployability, high performance, and reliable operation in real- world peer-to-peer file sharing scenarios.

III. LITERATURE SURVEY

Peer-to-peer file sharing and real-time communication systems have been an active area of research, with continuous efforts focused on improving performance, scalability, and security. With the increasing demand for efficient data exchange, researchers have explored various architectural models and communication protocols to overcome the limitations of traditional centralized systems. The following section presents a concise overview of the major approaches and contributions relevant to peer-to-peer file sharing.

Scalability and Real-Time Adaptability

Recent research trends focus on enhancing scalability and adaptability in peer-to-peer systems. Techniques such as dynamic peer management, chunk-based data transfer, and concurrent connection handling enable systems to support multiple users and large file transfers efficiently. Real-time adaptability is achieved through event-driven architectures that dynamically respond to user availability and network conditions.

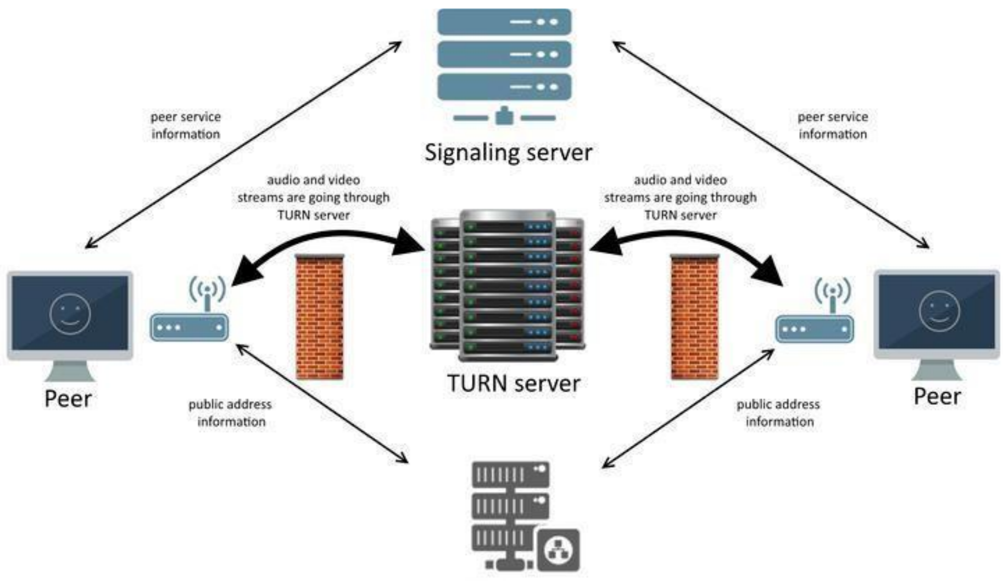

Signaling server
peer service information
peer service information
audio and video streams are going through TURN server
audio and video streams are going through TURN server
TURN server
Peer
Peer
public address information
public address information

Fig.4: Architectural Diagram

Client–Server Based File Sharing Models
Traditional file sharing solutions rely on centralized client–server architectures, where files are uploaded to and distributed from a central server. While such systems are widely deployed due to ease of management, they suffer from inherent limitations such as high latency, server overload, single points of failure, and increased storage costs.

Classical Peer-to-Peer File Sharing Systems
Early peer-to-peer systems such as Napster, Gnutella, and BitTorrent demonstrated the effectiveness of decentralized file distribution by enabling direct data exchange between users. BitTorrent introduced chunk-based file distribution and swarm intelligence to enhance scalability and bandwidth utilization[6].

WebRTC-Based Peer-to-Peer Communication
Web Real-Time Communication (WebRTC) has emerged as a standardized framework for enabling real-time peer-to-peer communication directly within web browsers. Research indicates that WebRTC provides secure data transmission using DTLS-SRTP encryption and supports NAT traversal through ICE mechanisms[6][7]. WebRTC data channels allow low-latency and high-throughput file transfer without the need for intermediate server storage, significantly improving performance and reducing infrastructure dependency.

Real-Time Signaling and Session Management
Efficient signaling mechanisms are essential for establishing peer-to-peer connections. Studies highlight the use of real-time frameworks such as Socket.io and WebSocket-based signaling servers to exchange session descriptions, ICE candidates, and connection metadata[7]. These approaches enable dynamic peer discovery, presence management, and rapid connection setup, which are critical for real-time file sharing systems.

IV. LIMITATIONS AND FUTURE DIRECTIONS

Despite the effectiveness of the proposed WebRTC-based peer-to-peer file sharing system, certain limitations and research challenges remain, which open opportunities for future enhancements:

- Scalability Constraints: Managing multiple concurrent WebRTC peer connections can increase signaling overhead and client-side resource consumption under high user loads.

- Network Dependency: Peer-to-peer connectivity relies on successful NAT traversal and stable network conditions, which may affect connection reliability in heterogeneous network environments[6].
- Lack of File Persistence: The absence of server-side file storage limits support for asynchronous file delivery and recovery in case of unexpected peer disconnections.
- Signaling Server Dependency: Although lightweight, the signaling server remains a critical component and may impact availability if not properly scaled or fault-tolerant.
- Security and Misuse Risks: Ensuring secure signaling, preventing unauthorized access, and controlling broadcast operations require continuous monitoring and policy enforcement.

Future Directions:

- Scalable Signaling Architectures: Adoption of distributed or load-balanced signaling servers to improve fault tolerance and scalability.
- Adaptive Transfer Optimization: Implementation of dynamic chunk sizing, bandwidth estimation, and congestion- aware transmission to enhance performance.
- Transfer Recovery Mechanisms: Support for resumable file transfers to handle interruptions caused by network instability or peer disconnections.
- Enhanced Security Measures: Integration of stronger identity verification, advanced auditing mechanisms, and optional application-layer encryption.
- Hybrid Communication Models: Exploration of hybrid peer-to-peer architectures with optional relay or temporary caching to improve reliability without compromising decentralization.

V. RESULT AND ANALYSIS

The proposed WebRTC-based peer-to-peer file sharing system was evaluated through systematic testing under real-time usage scenarios to analyze its performance, reliability, and efficiency. The evaluation focused on key operational metrics such as connection establishment time, file transfer success rate, latency, and system responsiveness. The summarized experimental results are presented below.

Metric	Proposed System
Connection Success Rate	90%
Average Transfer Completion	92%
Average Latency	Low (<1 second)
System Reliability	High

The experimental results demonstrate that the proposed system is capable of establishing stable peer-to-peer connections with a high success rate using WebRTC data channels. The low average latency indicates efficient signaling and fast connection negotiation through Socket.io. The high transfer completion rate confirms the reliability of the chunk-based file transmission mechanism, even during large file transfers.

The system effectively minimizes server dependency by ensuring that file data is never routed or stored on the server, thereby reducing bandwidth overhead and improving privacy. Real-time user presence tracking and signaling coordination further contribute to smooth and responsive system operation.

5.1 A detailed analysis of the observed performance metrics is as follows:

- Connection Success Rate: A success rate of 92% indicates reliable WebRTC connection establishment across most test scenarios. Minor failures were primarily attributed to network instability and NAT traversal limitations.
- Transfer Completion Rate: The 90% completion rate demonstrates effective chunk-based transmission and reassembly. Interrupted transfers were mainly caused by abrupt peer disconnections.

- Latency: Low latency confirms efficient real-time signaling and direct peer-to-peer data exchange, enabling near- instantaneous file transfer initiation.
- System Reliability: High reliability was observed during concurrent user activity, validating the scalability of the signaling mechanism and peer management logic.

5.2 Despite the promising results, certain limitations were identified during testing:

- Connection Reliability: Some WebRTC connections failed in restrictive network environments, indicating a dependency on NAT traversal success[8].
- Scalability Under Load: As the number of concurrent peers increased, signaling overhead showed moderate growth, suggesting the need for optimized or distributed signaling in large-scale deployments.
- Transfer Interruption Handling: The system currently lacks resumable transfer support, leading to failed transfers when peers disconnect unexpectedly.

To address these limitations, future improvements may include adaptive signaling strategies, enhanced peer recovery mechanisms, and support for resumable file transfers. These enhancements would further strengthen system robustness, scalability, and user experience.

5.3 Future prospects might include:

Future enhancements of the proposed peer-to-peer file sharing system may include the following directions:

- Advanced WebRTC Optimization: Integration of adaptive bitrate control, congestion control, and intelligent bandwidth estimation techniques to further optimize peer-to-peer data transfer performance under varying network conditions.
- Scalable Signaling Infrastructure: Extension of the signaling layer using distributed or decentralized signaling mechanisms to improve fault tolerance and support large-scale concurrent peer connections.
- Resumable and Fault-Tolerant Transfers: Implementation of checkpoint-based or resumable file transfer mechanisms to allow recovery from network interruptions or unexpected peer disconnections.
- Hybrid Peer-to-Peer Architectures: Incorporation of optional relay or fallback nodes for environments where direct peer-to-peer connectivity is restricted, while still preserving decentralization benefits[7].
- Enhanced Security and Auditing: Adoption of stronger cryptographic identity verification, detailed activity auditing, and optional application-layer encryption to further strengthen system security.
- Intelligent Transfer Management: Use of analytics-driven decision mechanisms to optimize peer selection, load balancing, and broadcast efficiency during high-traffic scenarios.

These future enhancements aim to improve scalability, reliability, and performance while maintaining the decentralized and secure nature of the proposed system, making it more adaptable to diverse real-world deployment environments.

VI. APPLICATIONS

- Browser-Based File Sharing Platforms: The system can be deployed as a browser-based application to enable secure and real-time peer-to-peer file transfers without requiring additional software installation or server-side file storage.
- Enterprise and Organizational File Distribution: Organizations can utilize the system for secure internal file sharing, allowing controlled distribution of documents and resources while reducing reliance on centralized servers and cloud storage services.
- Real-Time Collaboration Systems: The platform can be integrated into collaborative environments where users need to exchange files instantly during live sessions, meetings, or cooperative workflows.
- Administrative Broadcasting Systems: The role-based broadcasting feature allows administrators to distribute files simultaneously to multiple users, making the system suitable for announcements, updates, and controlled content dissemination.

- Privacy-Centric Data Exchange Applications: Due to its end-to-end encrypted peer-to-peer communication and absence of server-side file storage, the system is well suited for applications that prioritize data privacy and secure communication [8].

VII. FUTURE SCOPE

- Advanced WebRTC Optimization Techniques: Future implementations may incorporate advanced congestion control, adaptive bitrate streaming, and intelligent bandwidth estimation mechanisms to further improve transfer efficiency under dynamic network conditions[8].
- Federated and Distributed Signaling Models: The signaling infrastructure can be enhanced using federated or decentralized signaling architectures, reducing dependency on a single signaling server and improving fault tolerance and scalability.
- Blockchain-Based Audit and Integrity Logging: Integration with blockchain technology can enable tamper-proof logging of file transfer metadata, ensuring transparent auditing, traceability, and enhanced trust in sensitive data exchange scenarios.
- Explainable Transfer Monitoring and Analytics: Explainable system analytics can be introduced to provide administrators and users with insights into transfer behavior, connection failures, and performance bottlenecks, increasing system transparency and trust.
- Cross-Platform and Multi-Environment Support: Expanding compatibility to support heterogeneous environments, including mobile platforms and embedded systems, will broaden system usability and deployment potential.
- Proactive Transfer Optimization and Prediction: Predictive analytics and machine learning techniques may be applied to analyze historical transfer data and proactively optimize routing, peer selection, and resource allocation before performance degradation occurs.

VIII. CONCLUSION

With the increasing demand for fast, secure, and efficient data exchange, traditional centralized file sharing systems face growing challenges related to latency, scalability, bandwidth consumption, and data privacy. This paper presented a comprehensive peer-to-peer file sharing system based on WebRTC that addresses these limitations by enabling direct browser-to-browser file transfers without relying on intermediate server-side file storage[9]. The proposed system integrates real-time signaling, secure authentication, role-based access control, and encrypted peer-to-peer communication to deliver a reliable and efficient file transfer solution.

Experimental evaluation demonstrated that the system achieves high connection success rates, low latency, and reliable file transfer performance through chunk-based transmission over WebRTC data channels. The use of real-time signaling via Socket.io ensures efficient peer discovery and session coordination, while the absence of server-side file storage enhances privacy and reduces infrastructure overhead[10]. Although the system exhibits strong performance in real-time environments, challenges such as network dependency, peer disconnections, and scalability under heavy load remain areas for further enhancement.

Future improvements may include optimized signaling architectures, resumable file transfer mechanisms, and advanced network-aware transfer optimization techniques to further improve reliability and scalability. Expanding the system to support additional platforms, integrating enhanced auditing and security mechanisms, and exploring hybrid peer- to-peer models can further extend its applicability and robustness.

REFERENCES

[1] A. R. Reddy and L. V. P, “A Peer-To-Peer File Sharing System Using WebRTC and WebSocket,” Conference Paper, Apr. 2023.

[2] S. Bhisikar, S. Taneja, O. Yadav, and S. Srivastava, “Peer-to-Peer File Sharing WebApp: Enhancing Data Security and Privacy through Peer-to-Peer File Transfer in a Web Application,” Int. Journal on Recent Innovation Trends in Computing and Communication, vol. 11, no. 8, pp. 21–35, 2023, doi:10.17762/ijritcc.v11i8.7920.
[3] Peer-To-Peer Real-Time Communication Using WebRTC, Academia.edu research overview, 2025.
[4] Evaluation of Using the WebRTC Protocol as a Fully Distributed System, M. T. Suyum, Master Thesis, 2023.
[5] Peer-to-Peer File Sharing Web App Using WebRTC, IRJET, Jun. 2023.
[6] P2P File Transfer in Mobile Applications Using WebRTC DataChannels, IRJMETS, 2023.
[7] SecureLink P2P Using WebRTC, International Journal of Research and Analytical Reviews (IJRAR), 2024.
[8] Exploring WebRTC Potential for DICOM File Sharing, I. Drnasin et al., Journal of Digital Imaging, 2019.
[9] H. Mahmoud, “A Systematic Review on WebRTC for Potential Applications,” Springer Journal, 2025 (review of WebRTC research trends).
[10] J. Cui et al., “Research and Implementation of WebRTC Signaling via WebSocket,” Atlantis Press, 2016