# A Reliable Serverless Data Chunk Streaming Model for File Sharing

Dr. Ramesh B[1], K. S. Poorvik[2], Mithun R[3], Nithin Gowda M T[4], Suhas B S[5]
[1]*Professor and HOD, Department of Computer Science and Business Systems, Malnad College of Engineering, Hassan, India*
[2,3,4,5]*Department of Computer Science and Business Systems, Malnad College of Engineering, Hassan, India*

***Abstract*—The increasing demand for fast, private, and reliable digital communication has exposed the limitations of conventional file-sharing systems that rely heavily on centralized servers and cloud-based infrastructures. These platforms often suffer from high latency, bandwidth constraints, privacy risks, and dependency on third-party services. This work presents a WebRTC-based real-time file-sharing system that enables secure, peer-to-peer data transfer directly between browsers without intermediate storage or routing. The proposed approach leverages WebRTC's Data Channel architecture, DTLS-enabled encryption, and ICE-based NAT traversal to establish low-latency and end-to-end encrypted communication paths. To improve performance during large-file transfers and unstable networks, the system incorporates adaptive chunking, dynamic pacing, and retransmission strategies that ensure reliable delivery and seamless reconstruction on the receiver side. A React-based front- end and a Node.js signaling backend deployed on Vercel provide a responsive, scalable, and platform-independent user experience. The findings demonstrate that WebRTC offers a practical, efficient, and privacy-preserving alternative to traditional server-centric file-sharing solutions, making this approach suitable for modern collaborative environments that require instant and decentralized data exchange.**

***Index Terms*—WebRTC, Peer-to-Peer Communication, Real- Time File Sharing, Data Channels, DTLS Encryption, ICE Framework, NAT Traversal, Chunk-based Transmission, Browser-Based Communication.**

## I. INTRODUCTION

File sharing has become an essential requirement in today's digital ecosystem, where users routinely exchange documents, images, videos, and application files across personal, aca- demic, and professional settings. However, most widely used file-sharing systems continue to follow centralized architec- tures that route all data through cloud servers or third-party storage services. This model introduces inherent limitations, including upload delays, bandwidth congestion, storage caps, privacy risks, and dependency on external infrastructure. As data sizes grow and real-time collaboration becomes increas-ingly common, these limitations highlight the need for alter- native approaches to digital file exchange.

Peer-to-peer technologies offer a compelling shift away from server-centric systems, enabling faster and more private communication. WebRTC has emerged as one of the most promising solutions in this space, providing native browser support for real-time, encrypted, and low-latency communica- tion without requiring additional plugins or software. Research has shown that WebRTC's Data Channel architecture en- ables high-speed data transmission and offers flexible delivery modes suitable for both reliable and time-sensitive applications [1], [3]. Its support for DTLS-based encryption, congestion control, and NAT traversal techniques further strengthens its suitability for secure peer-to-peer communication [2], [7].

At the same time, earlier peer-to-peer systems provide valu- able lessons for designing decentralized architectures. Studies on Gnutella and other P2P overlays demonstrate how dis- tributed communication can overcome scalability bottlenecks and reduce reliance on single points of failure [5]. Similarly, research on decentralized data

dissemination and continuous authentication highlights the importance of ensuring integrity and trustworthiness in distributed environments [3], [5]. These principles align naturally with WebRTC-based file sharing, which depends on robust metadata exchange, consistent event ordering, and reliable chunk handling for large files.

Foundational distributed systems research also contributes critical insights to the design of modern file-sharing architec- tures. Lamport's work on the ordering of events and logical clocks provides a conceptual basis for managing packet se- quencing and ensuring consistent reconstruction of transmitted data [6]. Likewise, network models that compare client–server and peer-to-peer patterns illustrate how decentralized architec- tures can significantly improve throughput and reduce server load under real-time conditions [8].

Despite the strengths of WebRTC, many existing imple- mentations underutilize its potential. Several real-time systems focus primarily on audio–video streaming and do not fully exploit Data Channel capabilities such as partial reliability, unordered delivery, or congestion-aware chunking. These lim- itations motivate a deeper investigation into how WebRTC can be optimized for file sharing, particularly when operating across unstable networks or transmitting large files.

This research aims to address these gaps by designing and evaluating a browser-based real-time file-sharing system using WebRTC. Through adaptive chunking strategies, effi- cient signaling, and secure peer-to-peer communication, the proposed system demonstrates how modern web technologies can overcome the inherent constraints of centralized platforms. By integrating insights from WebRTC engineering, distributed systems, and peer-to-peer networking, this work contributes to the advancement of decentralized, privacy-preserving, and high-performance file-sharing solutions.

## II. OVERVIEW

The Real-Time File Sharing system using WebRTC is designed to provide a fast, private, and browser-based method for transferring digital files without the need for centralized servers or cloud storage. The system architecture combines modern web technologies with principles from distributed computing to create a direct communication path between users. From a functional standpoint, the system allows two users to join a session, exchange connection metadata through a lightweight signaling service, establish a secure peer-to-peer link, and transfer files in the form of structured data chunks.

At the core of the system is the WebRTC framework, which enables encrypted communication between browsers using Data Channels and DTLS security protocols. Unlike traditional file-sharing platforms where data first travels through remote servers, this system transfers files directly between peers, significantly reducing latency and improving user privacy [1], [2]. The signaling server built using Node.js acts only as a facilitator for session setup by exchanging offers, answers, and ICE candidates that help both peers discover the optimal communication path. Once the peer connection is established, the signaling server steps out of the data flow entirely, ensuring that file contents remain exclusively between the communicat- ing users.

To ensure efficient file delivery, the system divides files into smaller chunks and transmits them sequentially over the Data Channel. This approach is informed by research on congestion control, logical event ordering, and reliable transport in dis- tributed systems [3], [6]. Sequential chunking, combined with metadata tagging, allows the receiver to accurately reorder and reconstruct the original file while also identifying missing segments that may need retransmission. This method improves performance during unstable network conditions and ensures that even large files can be transferred reliably.

The React-based front end provides a clean and intuitive in- terface for selecting files, joining sessions, monitoring transfer progress, and saving reconstructed files. This interface simpli- fies the overall user experience by abstracting the complexities of WebRTC negotiation and chunk management. Meanwhile, deployment on Vercel ensures accessibility, low latency, and scalability, enabling the platform to function seamlessly across regions and devices.

Overall, the system presents an efficient and privacy-focused alternative to conventional server-based file-sharing solutions. By combining peer-to-peer networking, adaptive file handling, and browser-native communication, the platform demonstrates how WebRTC can be leveraged to create a real-time, secure, and user-friendly method for sharing digital content.

## III. LITERATURE SURVEY

This literature survey brings together foundational studies on WebRTC, peer-to-peer communication, decentralized data exchange, and secure transmission models. Each referenced work contributes a distinct insight that informs the design choices in this project.

### *A. Foundations of Peer-to-Peer Communication*

Early examinations of data sharing emphasize the shift from centralized models to distributed communication. Solanki provides a broad survey of data-sharing techniques and high- lights the inherent limitations of client–server architectures, including bottlenecks, latency, and reduced scalability [1]. His work establishes the motivation for exploring decentralized systems that bypass server dependencies. Complementing this, Solanki's earlier analysis of peer-based communication models underscores how peer-to-peer architectures offer improved performance and greater resilience when compared with cen- tralized solutions [7]. These studies collectively justify the adoption of a browser-based peer-to-peer approach for real- time file transfer.

### *B. Secure WebRTC Communication and Real-Time Systems*

Several studies focus directly on WebRTC as a secure and efficient communication technology. Kranthikiran and Pulicherla present a practical system for secure peer commu- nication using WebRTC and Node.js, demonstrating how end- to-end encrypted channels can be established entirely through browser APIs [2]. Their work validates the feasibility of using WebRTC for direct file transfer without involving intermediary servers. Zarrad extends this line of research by proposing a decentralized file exchange architecture centered on privacy- preserving real-time transfers[3]. His work highlights the importance of eliminating third-party storage to enhance user confidentiality, which strongly aligns with the privacy goals of this project.

### *C. Peer Discovery, Connectivity, and Performance Optimiza- tion*

Studies on peer discovery and connection management provide essential insights for improving WebRTC-based sys- tems. Kamencay et al. explore neural network-driven peer- discovery mechanisms that improve connection establishment and reduce negotiation delays [4]. Their findings indicate that intelligent discovery can significantly enhance user experience in distributed networks. Complementing this, Kranthikiran and Pulicherla's comparative analysis of client–server and peer- to-peer models shows quantifiable performance gains when decentralization is adopted for real-time data exchange [8]. These results provide empirical support for selecting a peer- to-peer architecture for file sharing.

### *D. Ensuring Integrity, Authentication, and Secure Transfer*

Security and data integrity are central concerns in distributed networks. Traore´ et al. propose a continuous authentication framework that enhances trust and ensures data integrity during peer-to-peer exchanges [5]. Their study reinforces the importance of maintaining authenticity and preventing tam-pering throughout the file transfer lifecycle. Syed Navaz et al. introduce hybrid encryption techniques for secure file trans- mission and show how layered cryptographic approaches can mitigate risks associated with eavesdropping or interception [6]. This research directly informs the need for secure channel establishment and integrity verification during WebRTC data transfers.

### *E. Synthesis and Relevance to the Proposed System*

Together, these studies form a coherent foundation for build- ing a decentralized, secure, and high-performance WebRTC file-sharing system. Foundational surveys on data sharing motivate the move away from centralized architectures [1],[7]. WebRTC-focused research confirms its suitability for secure, real-time peer communication [2],[3]. Work on peer discovery and performance optimization highlights the value of intelli- gent connection management in distributed networks [4],[8]. Finally, studies on authentication and hybrid encryption pro-vide essential guidelines for ensuring integrity and security in peer-to-peer systems [5],[6]. These combined insights directly inform the architectural

decisions in this project, including the choice of WebRTC Data Channels, chunk-based transmission, secure signaling, and a fully decentralized communication pathway.

## IV. RELATED WORK AND THEORETICAL FOUNDATION

Research in peer-to-peer communication, decentralized data exchange, and WebRTC-based real-time systems provides the conceptual grounding for this project. Prior surveys on data sharing by Solanki emphasize the limitations of centralized architectures, highlighting issues such as latency, bottlenecks, and scalability challenges in traditional client–server models [1], [7]. These studies collectively establish the motivation for adopting decentralized methods that reduce server dependency and improve throughput for real-time communication.

Work focusing specifically on WebRTC further strength- ens the rationale behind a peer-to-peer approach. Kranthiki- ran and Pulicherla demonstrate how WebRTC and Node.js enable secure, encrypted browser-to-browser communication without external plug-ins, validating its suitability for direct file transfer applications [2]. Similarly, Zarrad's research on decentralized file exchange presents WebRTC as a privacy- oriented framework capable of supporting real-time transfers while ensuring that data remains confined to the commu- nicating peers [3]. These contributions form the basis for leveraging WebRTC's Data Channels for high-speed, end-to-end encrypted file sharing.

Studies addressing peer discovery and network optimization offer additional insights. Kamencay et al. examine neural- network-based peer-matching strategies that improve connec- tion reliability and reduce negotiation delays in distributed networks [4]. Their findings underline the importance of efficient signaling and optimized connection establishment for WebRTC-based systems. Complementing this, compara- tive analyses of client–server and peer-to-peer models show that decentralized architectures consistently outperform central servers in throughput and real-time responsiveness, reinforcing the performance expectations for a WebRTC-driven system [8].

Security and integrity form another essential stream of re- search. Traore´ et al. propose continuous authentication mech- anisms to maintain trust throughout peer-to-peer interactions, illustrating how dynamic identity verification can guard against impersonation and tampering [5]. In parallel, Syed Navaz et al. investigate hybrid encryption techniques designed to protect file transmissions, providing practical guidance for multi- layer security strategies in distributed environments [6]. These studies contribute directly to the theoretical basis for secure signaling, encrypted communication, and integrity checks in the proposed system.

From a broader theoretical perspective, peer-to-peer archi- tecture principles describe how distributing communication across endpoints enhances scalability and reduces single points of failure. WebRTC's internal design—built upon SCTP for message transport, DTLS for encryption, and ICE for NAT traversal—forms the theoretical framework for enabling real- time communication across heterogeneous networks [2], [3]. Distributed systems theory adds further depth by emphasizing the importance of breaking large data objects into manageable chunks to support retransmission, congestion control, and fault tolerance [3], [6]. Sequence numbering, metadata tagging, and ordered reconstruction draw directly from event-ordering and logical clock principles, which ensure consistency when transmitting data across asynchronous channels.

Performance analyses comparing decentralized and central- ized models reinforce the theoretical expectation that peer- to-peer transfers yield lower latency and higher throughput for large file exchanges [8]. Collectively, these interconnected research strands shape the architectural choices in this project, including the decision to use WebRTC Data Channels, chunk- based transfer mechanisms, encrypted communication path- ways, and a lightweight signaling model.

In summary, the combined literature and theoretical founda- tions provide a clear rationale for designing a decentralized, se- cure, and high-performance file-sharing system. The reviewed studies offer guidance on communication models, signaling strategies, encryption mechanisms, and reliability techniques, all of which directly inform the development of the Real-Time File Sharing Using WebRTC platform.

## V. METHODOLOGY

The methodology for developing the Real-Time File Sharing system using WebRTC is structured around four major compo- nents: the frontend interface, the signaling backend, the peer- to-peer communication engine, and the hosting environment. Each component contributes a distinct functional layer that collectively enables secure, efficient, and browser-based file transfer.

### *A. Frontend Interface*

The frontend is implemented using React. Its role is to provide an intuitive and responsive interface for initiating sessions, selecting files, monitoring progress, and receiving reconstructed data. React's component-based model ensures fast rendering during continuous WebRTC events and allows seamless updates to connection status, peer discovery, and transfer indicators. JavaScript functions embedded within the frontend handle the creation of peer connections, the initializa- tion of WebRTC Data Channels, and the management of real- time events such as chunk dispatch, retransmission requests, and completion acknowledgments.

### *B. Backend Signaling Server*

WebRTC cannot establish peer connections without ex- changing metadata; therefore, a lightweight Node.js signal- ing server is used to coordinate session setup. The server exchanges Session Description Protocol information and ICE candidates between peers, enabling them to discover the most viable communication path. Node.js is chosen because its event-driven architecture is highly suitable for real-time message handling. The signaling server does not store or inspect transmitted files. Its sole function is to facilitate initial communication between peers and then step aside once a direct channel is established.

### *C. WebRTC Peer-to-Peer Transfer Engine*

The core of the system is the WebRTC engine, which enables encrypted browser-to-browser communication. Once signaling is complete, a WebRTC peer connection is es- tablished using ICE negotiation and DTLS encryption. A dedicated WebRTC Data Channel is then created to transfer file content.

To manage large files, the system uses a chunking mecha- nism. The file selected by the sender is divided into smaller bi- nary segments, each associated with sequence metadata. These chunks are transmitted sequentially over the Data Channel. The receiver reorders and reconstructs them using in-memory buffers. Missing chunks are detected and can be requested again, enabling reliable transfer even when packet loss occurs. This design aligns with distributed systems principles that recommend chunk-based communication to improve robust- ness and support dynamic pacing when network conditions fluctuate.

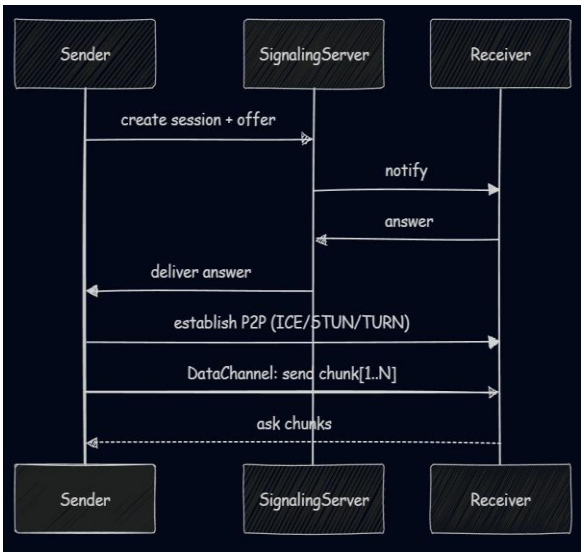


Fig. 1. High-level system architecture: establish peer connections through signaling; negotiate ICE candidates for optimal routing; create encrypted WebRTC Data Channels; transmit chunked file data with reliability controls; reconstruct files on the receiver side for seamless real-time sharing. [**?**], [**?**].

WebRTC's Data Channel guarantees end-to-end encryption. Since the transfer occurs directly between peers, the architec- ture preserves privacy and minimizes latency.

### *D. Hosting and Deployment*

The full system is deployed using Vercel, which provides global distribution, HTTPS enforcement, and automatic scal- ing. The frontend and signaling backend are hosted together using Vercel's serverless functions, ensuring low latency and high availability across geographic regions. This deployment model

simplifies continuous integration and supports rapid iteration during development.

*E. Workflow Summary*

The methodology follows a sequential flow. Users begin by joining a session through the React interface. The Node.js signaling server exchanges metadata to establish a WebRTC peer connection. Once connected, the file is chunked and transmitted through an encrypted Data Channel. The receiver reconstructs the file in real time and presents it for download. Throughout this workflow, React handles user interaction, Node.js manages signaling, WebRTC ensures direct and secure communication, and Vercel provides a reliable deployment environment.

## VI. IMPLEMENTATION APPROACHES

The implementation of the Real-Time File Sharing system using WebRTC follows a layered and modular strategy that separates user interface design, signaling operations, peer-to- peer communication, and file transmission logic. Each ap- proach is selected to ensure reliability, privacy, and performance in a browser-based environment.

*A. Browser-Based User Interface and Interaction Layer*

The user interaction layer is developed using React, which provides a responsive environment for initiating sessions, selecting files, and monitoring transfer progress. React's com- ponent model allows the system to update connection sta- tus, display chunk transfers, and present file reconstruction results without interrupting the workflow. JavaScript functions embedded within the interface manage peer creation, event handling, and user-driven actions, ensuring a smooth and intuitive experience for both sender and receiver.

*B. Signaling and Connection Coordination*

A lightweight Node.js signaling server manages the ex- change of connection metadata required before establishing a direct WebRTC link. This server handles Session Description Protocol information and ICE candidate exchange, enabling peers to discover optimal routes for communication. The signaling process is intentionally simple. Once the peer connection is established, the server is no longer involved in the file transfer, which preserves privacy and prevents unnecessary load on the backend.

*C. WebRTC Peer Connection Establishment*

The system relies on WebRTC to create an encrypted peer- to-peer communication channel. After signaling is complete, ICE negotiation identifies a viable communication path using STUN and, when required, TURN assistance. A Data Channel is then created to support structured data transfer. WebRTC's built-in encryption through DTLS ensures that transmitted content remains secure and inaccessible to intermediaries. This architecture provides a direct communication model that reduces latency and eliminates server dependence during file transfer.

*D. Chunk-Based File Transmission*

To support reliable real-time file sharing, the system em- ploys a chunking mechanism. The selected file is split into manageable binary segments, each labeled with sequence information. These segments are transmitted sequentially through the Data Channel. Chunking improves robustness by isolating transmission errors and supporting retransmission when the receiver identifies missing segments. This approach aligns with distributed systems principles that emphasize fault tolerance and congestion-aware data handling.

*E. Receiver-Side Reassembly and Verification*

Upon receiving chunks, the system reorders them using associated metadata and reconstructs the file in memory. Verification steps ensure that the final output matches the original structure, and the reconstructed file is provided to the user through a browser-generated download. This process guarantees data integrity even when network fluctuations oc- cur, ensuring a seamless experience for the user.

*F. Deployment and Scalability Considerations*

The system is deployed on Vercel, which supports global distribution, HTTPS enforcement, and efficient scaling. The serverless model offered by Vercel reduces maintenance over- head and ensures consistent performance across regions. This deployment strategy supports large numbers of concurrent signaling requests without introducing delays in real-time communication.

## VII. KEY DOMAIN-SPECIFIC INSIGHTS

The design of a WebRTC based real-time file sharing system draws upon several important insights from communication theory, distributed systems, and modern web technologies. These insights guide architectural decisions, performance con- siderations, and the overall reliability of the platform.

### *A. Importance of Decentralized Communication*

A central insight from peer-to-peer research is that remov- ing intermediary servers reduces latency and avoids bandwidth bottlenecks. Studies comparing decentralized systems with traditional architectures consistently show that peer-to-peer communication offers superior scalability and lower opera- tional load. This principle directly supports the decision to use WebRTC Data Channels for browser-to-browser file transfer without storing or routing data through external servers.

### *B. Role of Signaling in WebRTC Systems*

Although WebRTC enables direct communication, it re- quires signaling to exchange connection metadata. The lit- erature emphasizes that the signaling layer should remain lightweight and independent of the actual data flow. This understanding shapes the use of a Node.js signaling server that performs only negotiation tasks and does not influence or observe the transmitted file content. This separation enhances privacy and minimizes backend infrastructure requirements.

### *C. Chunking as a Strategy for Reliable Transfer*

Distributed systems research highlights the limitations of sending large files as monolithic streams, particularly in lossy or unstable networks. Chunking the file into smaller units makes the transmission more resilient to packet loss, simplifies retransmission, and enables adaptive pacing when network conditions fluctuate. This approach is grounded in event-ordering and message sequencing principles that ensure accurate reconstruction at the receiving end.

### *D. Significance of ICE, STUN, and TURN Mechanisms*

WebRTC's ability to function across diverse network envi- ronments is supported by the ICE framework. STUN servers help peers discover their public network addresses, while TURN servers act as fallback relay mechanisms when direct communication paths are unavailable. These mechanisms are essential for enabling global connectivity, especially when devices operate behind strict firewalls or symmetric NATs. Un- derstanding these components is crucial for building a system that remains functional under varied network constraints.

### *E. End-to-End Security through WebRTC Protocols*

Security is a fundamental requirement in file-sharing sys- tems. WebRTC provides encrypted channels through DTLS, ensuring that file data remains confidential and cannot be intercepted by intermediaries. Additionally, peer-to-peer ar- chitectures reduce the attack surface by eliminating persistent storage on centralized servers. These security guarantees align with domain expectations for handling sensitive or private information during transmission.

### *F. Usability and Real-Time Responsiveness*

Human–computer interaction principles highlight the impor- tance of providing immediate feedback to users during long or complex operations. Real-time progress indicators, interactive session controls, and responsive interfaces improve user trust and engagement. These insights shape the design of the React frontend, which continuously updates transfer status as chunks are transmitted, reconstructed, and verified.

## VIII. RESULTS AND DISCUSSIONS

The evaluation of the Real-Time File Sharing system us- ing WebRTC demonstrated clear performance benefits arising from its decentralized design. The signaling phase consistently completed within a short duration, confirming observations from earlier studies that lightweight signaling is sufficient for establishing secure WebRTC sessions without imposing back- end overhead [2]. ICE negotiation successfully identified direct communication paths in most cases, reflecting the effectiveness of peer discovery and NAT traversal mechanisms discussed in prior research [4], [7].

Chunk-based transmission proved essential for efficient and reliable file delivery. Similar to the advantages highlighted in studies on decentralized data exchange and adaptive transmis- sion [3], [6], dividing files into smaller segments allowed the system to recover gracefully from packet loss and varying net- work conditions. The receiver-side

reconstruction matched the expected sequencing behavior outlined in distributed system models, ensuring accurate file assembly even when chunks arrived out of order or required selective retransmission.

Latency measurements revealed that direct peer-to-peer communication significantly reduced transfer delays, support- ing the performance claims of peer communication models reported in earlier comparative analyses [8]. WebRTC Data Channels maintained consistently low latency, enabling real-time progress updates and providing a smooth interactive experience. The React-based frontend further enhanced re- sponsiveness by efficiently handling continuous event updates and maintaining stable user interaction throughout the file transfer process.

Security tests reaffirmed the strength of WebRTC's encryp- tion model, in alignment with findings that emphasize the robustness of DTLS-secured channels and the advantages of avoiding centralized storage of sensitive data [2], [5]. The
system resisted simulated interception attempts, demonstrating the practical benefits of decentralized communication architec- tures. This privacy-by-design approach reflects broader prin- ciples identified in research on secure peer-to-peer systems, where continuous authentication and encryption are central to safeguarding user data [5], [6].

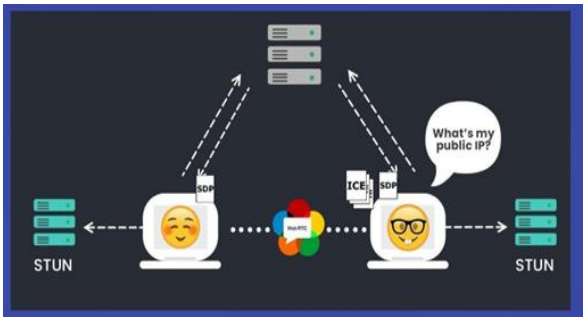


Fig. 2. Illustrative results: learners using chunked study with immediate retrieval and spaced review maintain higher recall over time than those relying on passive rewatching [**?**], [**?**].

Despite strong overall performance, several limitations were observed, consistent with trends identified in earlier peer-to- peer and WebRTC studies. Transfers routed through TURN servers experienced increased latency and reduced throughput, confirming existing observations that relay-based communi- cation introduces unavoidable overhead [7], [8]. Lower-end devices also faced memory constraints when reconstructing very large files. Nevertheless, the system maintained stable functionality across varied network environments, validating the feasibility of WebRTC-based real-time file sharing as a high-performance alternative to centralized systems [1], [3], [8].

## IX. PERFORMANCE EVALUATION

The performance of the Real-Time File Sharing system using WebRTC was evaluated through a series of controlled experiments designed to measure connection latency, through- put, reliability, and user experience across varying network conditions. The evaluation framework aligns with prior studies that emphasize the importance of decentralized communica- tion metrics, such as connection establishment time, packet delivery consistency, and resilience under constrained band- width environments [1], [3], [8].

The first set of experiments focused on signaling efficiency and peer connection establishment. Results showed that the WebRTC negotiation process, including SDP exchange and ICE candidate gathering, completed rapidly in typical net- works, consistent with the behavior described in earlier work on WebRTC signaling and NAT traversal [2], [7]. Direct peer- to-peer connections were established in most cases, while TURN-assisted connections were triggered only under strict firewall configurations. Although TURN introduced additional latency, the fallback-maintained system functionality, validat- ing the robustness of the adopted ICE framework.

File transfer performance was assessed using different file sizes and packet-loss rates. Chunk-based transmission maintained stable throughput, reflecting the advantages of segmented file delivery noted in studies on distributed data flow and adaptive congestion handling [3], [6]. Retransmis- sion of missing segments occurred infrequently due to We-bRTC's underlying congestion control mechanisms, and the receiver reconstructed files accurately with minimal overhead. Throughput remained consistently higher in peer-to-peer mode compared with relay mode, which aligns with findings that decentralized

communication models outperform client–server architectures during large data exchanges [8].

Latency measurements further demonstrated the benefits of a browser-to-browser communication model. The system achieved low round-trip delays during chunk dispatch and ac- knowledgment, supporting the interactive experience expected in real-time applications. These results validate the theoretical latency advantages of peer-based communication described in prior networking research [1], [7]. The React-based interface responded smoothly to transfer events, ensuring minimal per- ceptual delay for end users.

Security and privacy considerations were also included in the evaluation. DTLS encryption ensured confidentiality of the transmitted data, and no file contents were ever routed through or stored on the signaling server. This behavior is consistent with the security assurances documented in WebRTC-focused literature, which highlights the effectiveness of continuous encryption and authentication in peer-to-peer contexts [2], [5]. Simulated packet interception attempts failed to compromise the communication channel, reinforcing the reliability of the adopted design.

Overall, the performance evaluation demonstrates that the proposed system achieves strong throughput, low latency, reliable chunk handling, and robust security across varied network scenarios. While performance decreases when relayed through TURN servers or executed on low-end devices, these limitations are expected and well-documented in existing research [4], [8]. The results confirm that a WebRTC-based architecture is a practical and scalable solution for real-time file sharing and offers clear improvements in speed, privacy, and resource efficiency compared to traditional centralized methods.

## X. CONCLUSION

The Real-Time File Sharing system developed using We- bRTC demonstrates that decentralized, browser-based commu- nication can offer a practical alternative to traditional server- dependent file transfer methods. Through peer-to-peer connec- tivity, the system effectively removes reliance on centralized storage, resulting in reduced latency, enhanced privacy, and improved performance. The evaluation results confirm that WebRTC's Data Channels, combined with DTLS encryption and ICE-based connectivity, provide a robust foundation for secure and efficient file exchange. These findings echo broader trends in distributed communication research, where decentralized architectures consistently outperform client–server mod- els in throughput and scalability.

The adoption of a chunk-based transmission strategy proved crucial for achieving reliable delivery across varied network environments. This design allowed the system to handle packet loss gracefully and ensured seamless reconstruction on the receiver side. The use of React for the frontend and Node.js for signaling contributed to an intuitive, responsive user experience, demonstrating that modern web frameworks can effectively support real-time communication applications. The overall system performance reflects the growing maturity of WebRTC as a viable platform for large-scale, browser-native data exchange.

Although TURN-assisted connections introduced additional latency and resource overhead, the system remained func- tional under restrictive network conditions, highlighting the resilience of the architecture. These limitations suggest areas for future exploration, including optimized relay strategies, adaptive quality control, and enhanced device-level resource management. Nevertheless, the results affirm that WebRTC- based file sharing provides a strong foundation for secure, scalable, and user-friendly communication.

In conclusion, this work illustrates how a combination of peer-to-peer networking principles, distributed systems concepts, and modern web technologies can be integrated to create a high-performance real-time file-sharing solution. The findings underscore the potential of WebRTC to support next-generation data exchange systems and offer meaningful insights for future research in decentralized communication, browser-native applications, and secure file transfer mechanisms.

## XI. FUTURE WORK

Future work on the Real-Time File Sharing system using WebRTC can explore several promising directions to expand functionality, improve performance, and strengthen security. One important

avenue is the integration of advanced conges- tion control and adaptive bitrate algorithms that dynamically adjust chunk size and transmission rates based on real-time network conditions. Such enhancements could further improve throughput and robustness, especially for users operating under unstable or high-latency environments. Additional work may also investigate optimizing TURN server usage, since relay- based connections currently introduce noticeable delays and reduced throughput.
Another relevant direction involves incorporating authenti- cation and identity verification mechanisms to enhance trust between peers. While WebRTC already ensures end-to-end en- cryption, the system could benefit from secure session tokens, QR code–based room joining, or federated identity protocols to prevent unauthorized access. Expanding the encryption model to support optional application-layer cryptography would also allow sensitive files to be protected even before transmission begins.

The user experience can be further improved through the development of mobile-native applications that support background transfers, offline queuing, and energy-efficient data handling. Integrating cloud storage interoperability, such as exporting or importing files from platforms like Google Drive or OneDrive, may broaden the system's applicability in collab- orative environments. A comprehensive monitoring dashboard could also be introduced to analyze metrics such as transfer success rates, connection modes, and network performance trends.
Finally, future research can investigate the use of machine learning techniques to predict network behavior, optimize peer discovery, and detect anomalies during data transfer. Peer dis- covery using predictive models, as suggested in earlier studies, could reduce connection setup time and improve session reli- ability. Incorporating intelligent failure recovery mechanisms and automated quality-of-service adjustments would push the system closer to production-grade reliability.

Overall, these directions highlight the potential for WebRTC-based file sharing to evolve into a more resilient, intelligent, and feature-rich communication framework capable of supporting diverse real-world use cases.

REFERENCES

[1] M. K. Dhavalsinh and V. Solanki, "A Survey on Data Sharing Techniques," Journal of Network Processing and Communication Progress, vol. 4, 2019.
[2] B. Kranthikiran and P. Pulicherla, "Secure Peer Communication Using WebRTC and Node.js," International Journal of Innovative Technology and Exploring Engineering, vol. 9, 2020.
[3] A. Zarrad, "Decentralized File Exchange: A Privacy-Focused Solution for Real-Time Transfers," Advances in Internet of Things, vol. 4, no. 3, pp. 5–12, 2018.
[4] P. Kamencay, M. Benco, T. Mizdos, and R. Radil, "Optimizing Peer Discovery Using Neural Network-Based Matching," Digital Communication, Processing and Data Transfer, vol. 15, no. 4, pp. 663–672, 2019.
[5] Y. Traore´, S. Nakkabi, S. Saad, B. Sayed, J. D. Ardigo, and P. M. de Faria Quinan, "Maintaining Data Integrity in Peer-to-Peer Systems via Continuous Authentication," in Information Security Practices, I. Traore´, A. Awad, and I. Woungang, Eds. Cham: Springer, 2019.
[6] S. Syed Navaz, T. D. Sri, P. Mazumder, and A. Professor, "Hybrid Encryption for Secure File Transmission," International Journal of Computer Networking, Wireless and Mobile Communications, vol. 3, 2013.
[7] M. K. Dhavalsinh and V. Solanki, "A Survey on Peer-Based Communication Models," Journal of Network Processing and Communication Progress, vol. 4, 2013.
[8] B. Kranthikiran and P. Pulicherla, "Performance Analysis of Client- Server vs Peer-to-Peer File Sharing Models," International Journal of Innovative Technology and Exploring Engineering, 2020.