Cureus Journal of Computer Science

Open Access Original Article

Integrating Artificial Intelligence and Encryption in Web Real-Time Communication: A Smart Video Conferencing Platform With Real-Time Transcription and Translation

Samarth K. Patil [1] , Vinayak Nigam [1] , Shriyash Olambe [1] , Anita Shinde [1]

1. Computer Engineering, Marathwada Mitra Mandal's College of Engineering, Pune, IND

Corresponding authors: Samarth K. Patil, samarthpatil2021.comp@mmcoe.edu.in, Vinayak Nigam, vinayaknigam2021.comp@mmcoe.edu.in, Shriyash Olambe, shriyasholambe2021.comp@mmcoe.edu.in, Anita Shinde, anitashinde@mmcoe.edu.in

Received 11/13/2024
Review began 11/17/2024
Review ended 04/03/2025
Published 04/12/2025

DOI: https://doi.org/10.7759/s44389-024-02669-z

Abstract

The rapid growth of remote work and global collaboration has underscored the need for advanced, accessible communication tools. Existing video conferencing platforms often fall into two categories: those with only basic functionality and those overly complex for users focused on boosting productivity. To bridge this gap, in this paper, we present BabelTalk, an artificial intelligence (AI)-driven, smart video conferencing platform built on web real-time communication with a unique quasi-peer architecture. Unlike traditional solutions, BabelTalk seamlessly integrates real-time AI processing for live transcription, multilingual translation, and automated meeting summarization, enhancing both accessibility and user productivity.

BabelTalk addresses several key limitations of current platforms. It offers real-time transcription in a peer-to-peer environment - a feature lacking in many mainstream tools - and provides a persistent chat system for contextual continuity, allowing newly joined users to understand prior discussions. Additionally, BabelTalk incorporates secure file-sharing directly within the video call, eliminating the need for external platforms. Rigorous testing and benchmarking demonstrate BabelTalk's scalability, security, and functionality, illustrating its potential to revolutionize remote collaboration. This research contributes to the evolution of communication technology by providing a scalable, secure, and feature-rich solution tailored to modern collaborative needs for small to medium teams.

Categories: AI applications, Remote Work and Collaboration Tools, Web development
Keywords: webrtc, speech-to-speech translation (s2st), text-to-speech synthesis (t2s), unit-to-unit translation (utut), synchronization mechanism, media synchronization

Introduction

In today's digital landscape, the need for effective real-time communication has surged, particularly with the expansion of remote work, global collaborations, and virtual events. While existing platforms like Google Meet and Microsoft Teams offer basic connectivity, they struggle to deliver advanced artificial intelligence (AI)-driven functionalities that enhance user experience, accessibility, and productivity. Integrating features such as live transcription, translation, and intelligent summarization poses challenges related to latency, security, and user experience on these platforms. To address these limitations, this paper introduces BabelTalk - a novel, quasi-peer-based communication platform designed to enhance real-time AI functionalities over web real-time communication (WebRTC), the foundational technology supporting peer-to-peer video, audio, and data transfer. BabelTalk's architecture is tailored to integrate AI-driven features seamlessly, while prioritizing security, scalability, and performance, positioning it as a significant advancement over current solutions.

Rise in demand for real-time communication platforms

The demand for seamless, real-time communication platforms has grown rapidly due to the increasing prevalence of remote work, global collaboration, and virtual events [1]. Current platforms like Google Meet and Microsoft Teams, while widely used, often present complex feature sets or limited functionality, especially when integrating advanced AI features like live transcription, translation, and meeting summarization [2].

Limitations of existing platforms in AI integration

Most existing platforms lack an AI-centric architecture, resulting in suboptimal performance and compromised user experience when retrofitting AI capabilities. These platforms often suffer from increased latency, reduced security, and a degraded user experience, impacting accessibility and

How to cite this article
Patil S K, Nigam V, Olambe S, et al. (April 12, 2025) Integrating Artificial Intelligence and Encryption in Web Real-Time Communication: A Smart Video Conferencing Platform With Real-Time Transcription and Translation. Cureus J Comput Sci 2 : es44389-024-02669-z. DOI https://doi.org/10.7759/s44389-024-02669-z

productivity [3].

WebRTC as a foundational technology

WebRTC has emerged as the core technology for direct, plugin-free audio, video, and data transmission in web browsers [4]. Although widely adopted, WebRTC still faces challenges with scaling for complex tasks like synchronization and live AI processing, especially in large or AI-augmented meetings [5].

Growing necessity of AI features in communication

Advanced AI features, including real-time transcription, multilingual translation, and intelligent summarization, have transitioned from optional to essential, significantly enhancing accessibility and engagement [6]. The absence of real-time AI processing architectures in most existing platforms often leads to high latency, security issues, and a diminished user experience [7].

BabelTalk: A quasi-peer architecture for enhanced AI integration

BabelTalk's unique quasi-peer architecture addresses these challenges by incorporating a media server for efficient synchronization, mixing, and processing of audio and video streams. This model enables real-time transcription, translation, and summarization while minimizing latency and ensuring security through Docker containerization, which securely isolates each AI service for handling sensitive data.

Paper outline and contributions

This paper outlines BabelTalk's design and methodology, explaining how WebRTC is augmented by a quasi-peer model to support advanced AI features. It discusses media synchronization, security, and scalability challenges, demonstrating how BabelTalk's architecture provides a robust, feature-rich communication solution.

Materials And Methods

Study selection process

The studies included in this review were selected based on a predefined set of inclusion and exclusion criteria to ensure relevance and methodological rigor. The search strategy involved multiple steps to identify and select the most appropriate sources for the analysis.

Search Strategy

A comprehensive literature search was conducted across multiple academic databases, including Google Scholar, IEEE Xplore, ACM Digital Library, and ScienceDirect, to identify studies related to WebRTC, AI in communication, and encryption for video conferencing platforms. The search terms used included WebRTC in video conferencing; AI integration in WebRTC; end-to-end encryption in WebRTC; real-time communication security; and artificial intelligence and encryption in communication systems.

The search covered articles published between 2010 and 2024 to capture the most recent advancements in these areas. Additionally, the reference lists of selected articles were screened for further relevant studies.

Inclusion Criteria

Studies were included if they met the following criteria: (1) published in peer-reviewed journals, conferences, or workshops; (2) focused on WebRTC or technologies closely related to real-time communication; (3) addressed AI integration in real-time communication or video conferencing systems; (4) discussed security mechanisms, such as encryption protocols (e.g., Secure Real-time Transport Protocol [SRTP], Datagram Transport Layer Security [DTLS], Interactive Connectivity Establishment [ICE]) used in WebRTC; (5) provided empirical results or theoretical insights that were applicable to the implementation or enhancement of real-time communication systems.

Exclusion Criteria

Studies were excluded if they (1) did not focus on WebRTC or related technologies; (2) did not provide sufficient technical detail or were lacking in methodological rigor; (3) were not peer-reviewed (e.g., blog posts, white papers, or non-academic sources); (4) did not provide a clear connection to AI, encryption, or real-time communication, even if WebRTC was mentioned.

Study Selection Process

The initial search yielded a large number of studies, which were first screened by title and abstract for

relevance. Full-text articles were then assessed for eligibility based on the inclusion and exclusion criteria outlined above. The final set of studies included in this review was selected through consensus among the authors to ensure comprehensive coverage of the topic while maintaining focus on key themes.

Methodology

System Architecture

WebRTC serves as the foundation for enabling real-time communication in NexTalk, allowing direct browser-to-browser (b2b) connections without requiring extra plugins [6]. This technology provides an application programming interface (API) necessary for sharing audio, video, and data, enabling encrypted, low-latency communication over the internet. BabelTalk leverages WebRTC for video conferencing, voice calls, and data transfer, incorporating key features for connection management and security. The system architecture is presented in Figure *1*.

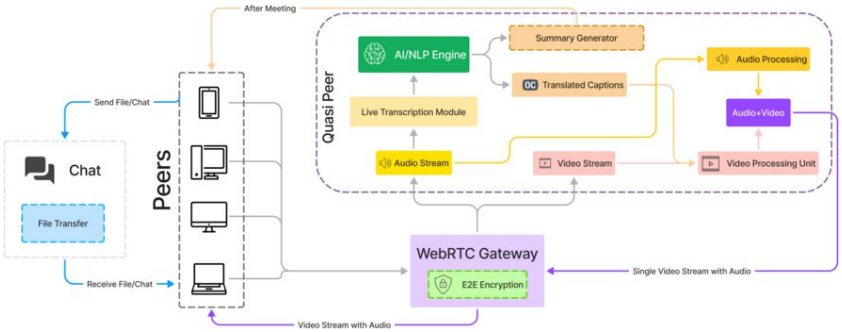

FIGURE 1: System Architecture

NLP, Natural Language Processing; AI, Artificial Intelligence; E2E, End to End

To establish secure P2P connections, WebRTC relies on several important components that are discussed as follows:

1) ICE (Figure *2*) [7] helps determine the best route for connection between peers by gathering possible candidates, such as IP (Internet Protocol) addresses and ports [8]. ICE may use STUN (Session Traversal Utilities for NAT) to assist peers in discovering their public internet protocol addresses when they are behind a NAT (Network Address Translation). If direct connections are not possible due to network restrictions, TURN (Traversal Using Relays around NAT) servers are used to relay the media streams [9].

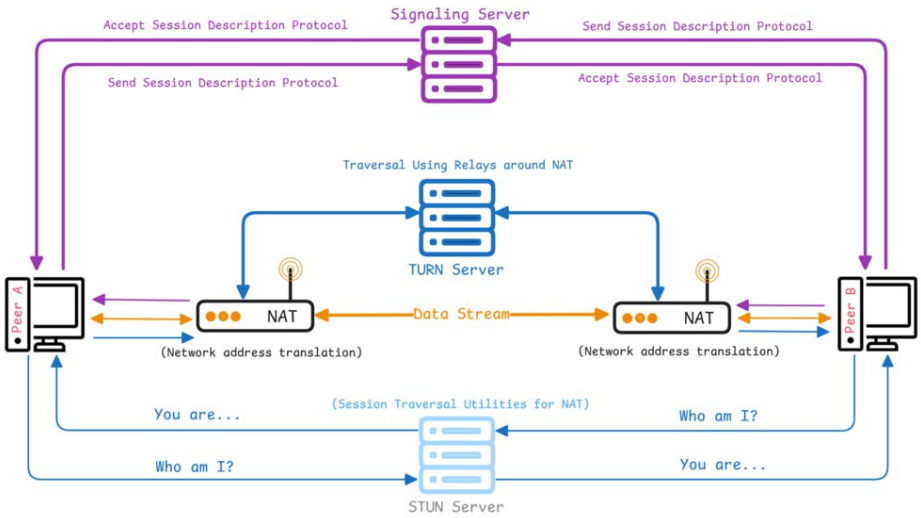

FIGURE 2: Vanilla Implementation of WebRTC

SDP, Session Description Protocol; TURN, Traversal Using Relays around NAT; NAT, Network Address Translation; STUN, Session Traversal Utilities for NAT

2) DTLS is employed to secure key exchange and signaling, ensuring that the process of setting up connections is protected from any unauthorized access [9].

3) SRTP is used to encrypt the actual media streams being shared, ensuring that all audio and video data remains private and secure during transmission [9].

Peer connection setup: WebRTC relies on RTCPeerConnection (Real-Time connection), a core API used to manage the connection between two peers. Before media can be transmitted, peers must establish a connection through a process involving signaling, session negotiation, and candidate gathering.

Signaling and Session Description Protocol

Signaling: WebRTC requires an external signaling mechanism to exchange connection details between peers (such as IP addresses and ports) [10]. While WebRTC does not specify how signaling is done, BabelTalk uses a signaling server to facilitate this process. The server only handles connection setup metadata and the exchanging of public keys between peers; it does not carry media streams (Figure *2*).

Session description protocol (SDP): Once signaling begins, peers exchange SDP messages that describe their connection parameters (audio codecs, video codecs, encryption methods, etc.). These messages help peers agree on how to establish the media channels. By default, Chrome sets a low transfer rate for WebRTC data channels, capping it at 30 kbps, which can make large file transfers painfully slow [11]. Fortunately, you can get around this limit by tweaking the SDP during the WebRTC connection setup. By simply changing the bandwidth setting in the SDP from b = AS:30 to a much higher value, like b = AS:1,638,400, you can raise the transfer speed to 1.6 Gbps. This small change makes a huge difference, speeding up large file transfers significantly [12].

Establishing Media Channels

Once the ICE candidate selection process is completed, WebRTC establishes encrypted media channels. The two main channels used are as follows:

1) Audio/Video Channels: These carry the real-time audio and video streams between peers.

2) Data Channels: WebRTC also supports data channels, which allow peers to exchange arbitrary data, such as text-based messages (for chat) or files. These data channels operate securely, just like the audio/video streams, and enable BabelTalk's persistent chat and file-sharing features.

In addition to audio and video streams, WebRTC supports data channels for sending non-media data between peers. In BabelTalk, RTCDataChannel is used to implement features such as persistent chat and file transfer.

RTCDataChannel API

The RTCDataChannel API allows peers to exchange any type of data-text, files, or binary data-securely over an encrypted connection. These data channels are also protected by DTLS and offer reliable or unreliable delivery, depending on the requirements (e.g., file transfer vs. live messaging).

Application in BabelTalk

Persistent chat: Data channels are leveraged to enable real-time messaging during meetings. All chat messages are sent securely via encrypted data channels, ensuring confidentiality.

File transfer: BabelTalk uses the data channels for file sharing between participants, ensuring that files are transferred securely and without relying on third-party cloud services. File transfer will be done by splitting them into smaller chunks for efficient P2P transmission [11]. The process starts with the user selecting a file, and the file's metadata (name and size) is sent to the receiver before the actual transfer begins. The file is then divided into chunks and transmitted sequentially using the WebRTC data channel. On the receiver side, the chunks are received, stored, and reassembled into the complete file. This method allows secure, real-time file sharing directly between users without relying on external servers [12].

Throughout the file transfer process, the file data is temporarily stored in memory on both the sender and receiver sides.

1) Sender Side: The file remains intact in memory as the sender slices and sends it in chunks.

2) Receiver Side: The file is reconstructed from the chunks and stored in the incoming File Data array. Each chunk is added sequentially to this array until the entire file is received. Once the complete file has been received, additional steps would be required to convert the data into a usable file format, such as creating a Blob object in the browser, which can then be downloaded.

Security

Signaling and key exchange: Diffie-Hellman key exchange generates a shared secret key between peers through a secure signaling server. To ensure secure media transmission, WebRTC relies on custom signaling methods for session control and authentication (Figure *3*) [13].

Key derivation: A key derivation unction is used to generate distinct keys: a media encryption key for end-to-end and a session-specific key for processing at the quasi-peer.

SRTP: Media streams are encrypted using SRTP with the media encryption key for secure communication between peers.

Session-specific key wrapping and sending to quasi-peer: The session-specific key is wrapped with a key-wrapping key derived from the shared secret key and securely transmitted to the quasi-peer.

Temporary decryption for AI processing: The quasi-peer unwraps the session-specific key, temporarily decrypts the media stream in an isolated Docker container, and processes it (e.g., transcription, translation). The stream is re-encrypted with the session-specific key after processing.

Final decryption by peers: The peers decrypt the re-encrypted stream using the session-specific key to access the final media.

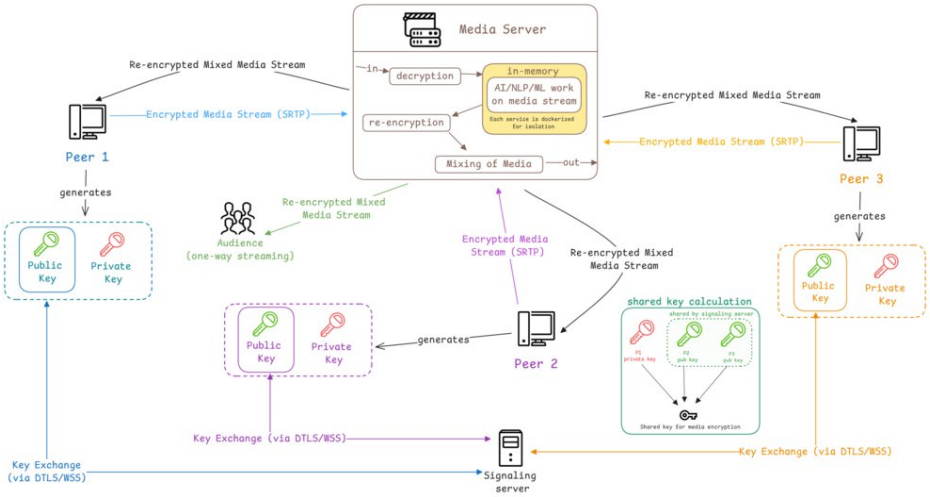

FIGURE 3: Encryption and Decryption Process

DTLS, Datagram Transport Layer Security; WSS, Web Socket Secure; SRTP, Secure Real-Time Transport Protocol; AI, Artificial Intelligence; NLP, Natural Language Processing; ML, Machine Learning

Additional WebRTC Features in BabelTalk

Network adaptation and quality monitoring: WebRTC automatically adapts to varying network conditions by adjusting the bitrate of the media streams. WebRTC's congestion control mechanisms ensure that media and data flows remain stable under varying network conditions, improving user experience during communication [14]. BabelTalk leverages WebRTC's built-in bandwidth estimation and congestion control algorithms to ensure that meetings run smoothly, even on low-bandwidth connections [15].

ICE restart mechanism: If a peer's connection is disrupted (due to network changes or connectivity issues), WebRTC supports ICE restarts, allowing BabelTalk to re-establish the connection seamlessly without interrupting the session.

Incorporation of Quasi-Peer in the Communication Architecture

To address the limitations of purely P2P WebRTC, BabelTalk introduces a quasi-peer (media server) to handle complex media processing tasks, such as real-time audio and video synchronization, mixing, and AI-driven features (e.g., transcription and translation). In typical P2P architectures, peers communicate directly, which limits scalability and flexibility when performing tasks like mixing streams or applying AI processing. The quasi-peer resolves these challenges by acting as an intermediary that processes media streams without generating its own data (Figure *1*).

While WebRTC enables encrypted, low-latency, P2P connections, certain tasks like real-time transcription, live translation, meeting summarization, and audio/video mixing require a centralized node that can process media streams. The quasi-peer performs these tasks by temporarily decrypting media streams, processing them, and re-encrypting them before transmission back to the peers or audience.

Media Reception and Synchronization

The quasi-peer receives media streams from multiple peers participating in a session. These media streams include audio and video, which are typically encoded using standard WebRTC codecs (e.g., Opus for audio and VP8/VP9 or H.264 for video). Each media stream is individually encrypted using SRTP before being sent to the quasi-peer.

Steps in media reception: The steps involved in media reception are as follows:

1) Receiving Media Streams: The quasi-peer receives encrypted media streams from all participating peers through WebRTC ICE connections.

2) Temporary Decryption for Processing: To perform tasks such as transcription or translation, the quasi-peer temporarily decrypts the media streams using a session-specific key. The media is decrypted only within Docker containers, ensuring that the raw data is isolated during processing.

3) Synchronization of Streams: Once decrypted, the quasi-peer synchronizes the incoming media streams. This is necessary because audio and video streams can arrive out of sync due to network latency or varying processing times at each peer. A global timeline is used to align the streams based on timestamps embedded in the media packets. Buffering and reordering mechanisms ensure that media streams are aligned before they are further processed or mixed [4].

AI Features in BabelTalk

Real-time transcription of spoken words: Transformers in automatic speech recognition (ASR): Created for transcription tasks, these models aim to change audio sequences into text. Their skill in managing sequence transduction, such as converting audio into text, is what makes them stand out [16].

1) Self-Attention Mechanism: Transformers utilize self-attention layers to concentrate on various aspects of the audio input at the same time instead of handling it sequentially. This enables more flexibility in data analysis and pinpointing the key aspects of the speech signal [17].

2) Speech Data Processing: Transformers, unlike traditional models, analyze the complete speech sequence simultaneously instead of in a sequential manner. This comprehensive perspective assists them in identifying important patterns and situations, which are essential for understanding complicated sentences, enhancing multilingual abilities, and boosting performance in noisy settings [18].

Real-time speech-to-speech translation: Transformer-based models, such as wav2vec 2.0 and S2U transformers, use self-attention mechanisms to analyze whole speech sequences together, capturing both local and global relationships [19]. This allows for direct translation of spoken language without the need for text in between, making it particularly beneficial for challenging tasks such as live translation [20].

Hybrid models like Conformer, which merge CNNs and transformers, capture both local and global features from speech signals. This mix enhances the system's capability to process intricate speech data in noisy settings or across various languages [20,21].

Many-to-many multilingual models in ASR outlined by Kim et al. [22] emphasize the benefits of employing many-to-many multilingual models in ASR. Their method centers on the direct conversion of spoken language into a target language without depending on intermediate text representations. This approach, commonly known as textless unit-to-unit mapping, utilizes transformers to analyze audio signals across languages by training on a substantial corpus of multilingual speech data. The model incorporates self-attention layers to dynamically identify which portions of the audio signal relate to meaningful speech units. Consequently, the system can proficiently recognize and transcribe speech from various languages, significantly improving its capability to facilitate real-time multilingual communication, making it particularly suitable for platforms such as BabelTalk.

Summarization of the transcripts: In BabelTalk's summarization feature, the goal is to pack spoken words into short overviews. They make sure to keep the key details and the overall meaning. Taking inspiration from the idea of Ding et al. [23], they rely on smart neural networks. These networks are great at picking out the most important bits from the words that have been written down.

1) Enhanced Semantic Attention in Summary Generation: The way of focusing on what meaning is, this works by looking closely at what each part of the text is saying. Rather than giving the same importance to every piece of text, the system scores different parts to highlight those bits that carry more useful information. By using a setup that includes several layers in its neural network, the system can analyze the transcript. This lets it get a deeper grasp of how words, phrases, and sentences connect.

2) Gain-Benefit Gate in Summary Generation: The gain-benefit gate is a new method in summary creation that helps decide which information should be included based on relevance and value. This method tackles the task of summarizing long transcripts by balancing the trade-off between acquiring key information and reducing unnecessary details or repetitions.

Layer of AI processing: Service for transcribing audio in real time.

Function: Produces live text transcriptions through use of models such as Whisper or comparable machine learning frameworks.

Output: Recorded text used as a foundation for additional processing. Service for providing translations.

The transcription in real time: Employs neural network models used in neural machine translation learn how to link input and output sequences, making them ideal for dealing with the complexities of human language and converting the text into the preferred language.

Live captions: Show translated text on screen as captions. Translating spoken words into another language using sound technology. Uses TTS technology to transform the translated text into synthesized speech for immediate audio output. Service that provides a brief overview or outline of information.

Transcription of the meeting in its entirety: Function: Analyzes the entire dialogue to produce a brief summary that showcases main points, choices, and tasks. A brief report distributed safely to all attendees following the event.

Function: Analyzes the complete dialogue from the event to generate a structured summary using AI-based natural language processing and dialog analysis techniques. This process includes audio-to-text transcription using ASR to convert spoken content into text with speaker attribution and timestamping. The text is then preprocessed by tokenizing sentences, removing noise (e.g., fillers, non-verbal cues), and ensuring each segment is attributed to the correct speaker for accurate representation of viewpoints.

Topic modeling methods, such as latent Dirichlet allocation or clustering algorithms based on BERT, identify and separate primary themes in the conversation, organizing the transcript by distinct topics. Key insights are captured using extractive summarization algorithms, which highlight central sentences from each segment, while abstractive models (e.g., GPT-4 or T5) rephrase content into a concise, coherent summary that represents the progression of ideas, decisions, and assigned tasks. This analysis also tracks consensus or dissent among participants to record any agreement or contrasting viewpoints.

Finally, the function generates a secure, accurate report distributed to all attendees, offering a reliable summary for follow-up, research, or action, enabling stakeholders to retain a comprehensive record of the event's key outcomes.

Final layer: During the meeting, participants can read real-time captions which are translated and displayed. Translated audio can be heard by participants in their native language via text to speech (TTS).

Distribution of meeting summaries: Participants receive the summary created at the end of the meeting through a secure channel.

Audio and Video Mixing

Once the streams are synchronized, the quasi-peer handles audio mixing and video composition. This allows BabelTalk to combine multiple peer streams into a single, unified output that can be sent back to peers or an audience.

Audio mixing: The quasi-peer mixes the audio streams from different participants into a single stream. The mixing process involves aligning the streams based on the previously synchronized timeline and combining them into a single audio track that can be played back to all peers.

1) Mixing Algorithm: Audio streams are mixed using a weighted sum algorithm where volume levels are adjusted dynamically to avoid overwhelming participants with multiple audio sources. Noise suppression and echo cancellation techniques are applied as part of the mixing process.

Video mixing: Video streams from different participants are combined into a single output stream. The quasi-peer can arrange the video frames in various layouts (e.g., grid, speaker view) depending on the requirements of the session.

1) Video Processing: The quasi-peer decodes the video streams into the planar YUV (Color Encoding in Video Processing) format, which is used for video manipulation. After the streams are combined and rendered, they are re-encoded using the same codec (e.g., VP8/VP9).

2) Hardware Acceleration: For efficient video processing, hardware acceleration techniques such as MMX2(MultiMedia eXtensions 2), SSE2 (Streaming SIMD Extensions 2) Fast, or SSSE3 (Supplemental Streaming SIMD Extensions 3) are employed, depending on the server hardware capabilities.

Broadcasting the Mixed Stream

After processing and mixing, the quasi-peer is responsible for broadcasting the mixed media stream back to the participants or a larger audience.

Broadcasting to peers: For small- to medium-sized meetings, the mixed stream is sent back to each peer over the established WebRTC connections. The stream is encrypted again using SRTP before transmission.

Broadcasting to audience: For larger meetings or webinars, the quasi-peer broadcasts the mixed stream to

a passive audience who can watch the video and listen to the mixed audio but does not participate in the meeting. This allows for scalability beyond standard P2P connections, which are limited by the number of peers.

Existing methodology and key findings

Reference	Methodology	Key Finding	Advantages	Disadvantages	Performance Matrix
Tang and Zhang [4]	The authors proposed a method to mix audio and video streams for real-time communications using WebRTC. The system uses a media server for processing and distribution of mixed media streams. The methodology focused on optimizing the audio-video synchronization and ensuring real-time communication.	The paper found that the audio and video mixing method enhanced the overall quality of communication, especially in multi-participant conferencing scenarios.	Improved quality of both audio and video, particularly in multi-participant settings. More scalable for large meetings.	Increased complexity in managing real-time mixing, requires more resources, which may be demanding for low-power devices.	Latency, synchronization accuracy, and resource usage were measured to gauge performance improvements.
Sredojev et al. [9]	This paper focuses on using WebRTC for real-time communication like video calls, chat, and file sharing without needing extra plugins. It covers key components like MediaStream for handling media (camera/microphone), RTCPeerConnection for direct connections, and RTCDataChannel for data transfer. It introduces a new signaling method with WebSocket since there is no set standard for signaling.	SecureLink, powered by WebRTC, enables fast, secure peer-to-peer communication without needing central servers. It uses encryption to protect data during transfers, though there's still some risk if a user is compromised. The system works well but depends on good network quality.	The main advantage of SecureLink is its ability to provide fast, real-time communication without relying on central servers, making it quicker and more reliable. It also uses encryption to keep data safe during transfers. Plus, it's flexible and can easily add new features like AI for future improvements.	The disadvantages of SecureLink include potential security risks if one of the peers is compromised, which could expose data despite encryption. Additionally, its performance heavily depends on network quality, so a bad connection can cause disruptions. Lastly, while it's promising, adding future AI features might take time and effort.	In SecureLink, performance is usually measured by factors like latency (how fast the data travels between peers), throughput (the amount of data transferred over time), and network stability (how well the system handles poor connections). Additionally, security performance is evaluated by how well encryption protects data during transfers. These metrics ensure the system runs smoothly, securely, and efficiently for users.
Duan and Liang [11]	The paper focuses on improving one-to-many file transfer efficiency, we divide the file into blocks. Each block is sent to a different receiver, and once a receiver gets a block, it forwards it to others. After receiving all blocks, the file is reassembled into a complete document. This strategy uses the network resources of all participants, reducing waiting time and speeding up the transfer process.	The strategy shows a reduction in total file transfer time as the number of workers increases, up to a certain point. When the number of workers exceeds 50, the efficiency starts to decline due to the overhead of managing multiple data channels.	This approach boosts file transfer speed by utilizing the bandwidth of receiving peers, improving efficiency as more peers join in. WebRTC ensures secure, encrypted peer-to-peer connections, enhancing privacy compared to server-based methods.	At the start of the transfer, speed may be slower due to fewer active senders, creating initial latency. The system's performance is also dependent on network stability, meaning slow or unreliable peers can affect overall transfer speed.	The main performance metric is the total transfer time, adjusted for the number of peers. The system's scalability is tracked by the number of peers, with different file sizes tested for impact. Bandwidth usage is also monitored to evaluate network efficiency.
Lopez-Fernandez et al. [13]	The paper highlights that WebRTC does not define signaling methods, allowing custom implementations. It discusses using WebSockets for better session control compared to SIP or XMPP. It explains MediaStream for handling audio/video and RTCDataChannel for data transfer. For authorization,	The authors found that WebSockets work well for flexible and efficient signaling in WebRTC. They compared two security models: ACLs, which offer better control, and	WebSockets enhance flexibility and efficiency in session control and message exchange compared to traditional protocols like SIP and XMPP. CAP performs faster due to in-memory token checks, making it ideal for real-time applications. ACLs offer fine-grained, user-	CAP, while faster, lacks user-specific control, making it less secure in multi-user environments where tokens can be shared or misused. CAP's lack of granularity limits its suitability for	The paper measures authorization speed in microseconds, comparing the efficiency of ACLs and CAP by testing the response times for user requests under different loads.

	ACLs provide detailed control but are slower, while CAP is faster with tokens but risks token sharing and lacks user-specific control.	CAP, which is faster but less secure in user-specific scenarios.	specific permission control, useful in scenarios requiring detailed access management.	applications needing revocable, user-specific permissions.	
Islam et al. [14]	The methodology involved implementing a coupled congestion control mechanism for both media and data flows in WebRTC. The authors used congestion control algorithms to regulate data traffic and conducted real-life evaluations under different network conditions.	The study showed that coupled congestion control improved the efficiency and stability of WebRTC media and data flows even under varying network conditions.	Stability in both media and data transmission, efficient use of bandwidth, applicable to real-world use cases.	Complex to implement and tune congestion algorithms, and performance is heavily dependent on network conditions.	Throughput, packet loss, and delay were measured to evaluate the performance of the congestion control mechanisms.
Kim et al. [22]	Introduces a method for multilingual speech-to-speech translation that skips the traditional step of converting speech to text. Instead, it directly maps speech sounds between languages, which improves accuracy, especially for less-studied languages.	The model showed substantial enhancements in multilingual translation compared to conventional methods by using text-based intermediate representations. It exhibited strong results in language pairs with limited resources, displaying flexibility and effectiveness.	The article introduces a new method that does not rely on text transcripts, allowing for direct multilingual speech-to-speech translation between multiple parties. It improves the scalability of low-resource languages, making them suitable for various linguistic contexts in real-world applications.	The lack of an intermediary textual representation can make it challenging for the method to maintain accurate semantic consistency in different languages. Performance may be negatively impacted by dealing with languages that are vastly different or tonal languages that heavily depend on subtle phonetic differences.	BLEU scores were employed to assess the precision of speech-to-speech translations among different language combinations. Perplexity and unit error rates were used to evaluate how well the model performs and generalizes in different languages.
García et al. [24]	The authors focused on measuring Quality of Experience (QoE) in WebRTC applications. They proposed a set of Key Performance Indicators (KPIs) to estimate QoE and analyzed the impact of different WebRTC topologies on QoE.	The paper identified key factors affecting QoE and provided a framework to measure it. The proposed KPIs give developers actionable insights to enhance user experience.	Emphasis on user satisfaction offers practical metrics for improving WebRTC applications. Comprehensive coverage of different topologies.	QoE measurement can be subjective and difficult to quantify precisely and requires a complex infrastructure to gather and analyze KPIs.	KPIs related to network conditions (latency, bandwidth, packet loss) and user satisfaction were used to measure QoE.

TABLE 1: Key Studies in WebRTC and AI

WebRTC, Web Real-Time Communication; ACLs, Access Control Lists; CAP, Capability-based Security; SIP, Session Initiation Protocol; XMPP, Extensible Messaging and Presence Protocol; SSL, Secure Sockets Layer; BLEU, Bilingual Evaluation Understudy; RTCPeerConnection, Real-Time Communication Peer Connection; RTCDataChannel, Real-Time Communication Data Channel; P2P, Peer-to-Peer; iDTT, Incremental Distributed Transfer Technique; AI, Artificial intelligence

This section evaluates each reference in terms of its relevance, strengths, and limitations, and how these findings influence the design of BabelTalk. By examining each study's applicability and challenges (Table *1*), we can identify gaps and highlight how BabelTalk addresses these through its unique quasi-peer architecture and AI-driven features.

Audio and Video Mixing Method for WebRTC

Tang and Zhang's [4] study on audio and video mixing for WebRTC proposes a media server for real-time synchronization and quality enhancement in multi-participant settings. While the proposed method successfully improves scalability and media quality, the high resource demand suggests potential challenges for low-power or resource-constrained environments. This insight informs BabelTalk's decision to use a quasi-peer approach for media processing but emphasizes the need for efficiency to

avoid excessive computational costs.

WebRTC Overview and Signaling Solution

Sredojev et al. [9] offer a foundational understanding of WebRTC's capabilities, with a focus on low-latency communication and signaling through WebSocket. While this study emphasizes direct, secure P2P connections, the performance sensitivity to network quality points to a need for robust fallback mechanisms in BabelTalk's design. This consideration is especially important given BabelTalk's integration of advanced AI features, which may require a more resilient signaling approach.

Block-Based File Transfer in WebRTC

Duan and Liang's [11] file transfer strategy, which leverages peer bandwidth for faster distribution, aligns with BabelTalk's need for efficient file sharing. However, reliance on network stability means that inconsistent performance may arise in environments with varying connection quality. BabelTalk may consider hybrid approaches, combining direct peer-to-peer (P2P) and server-based transfer to ensure consistent file-sharing experiences.

Authorization Models for WebRTC

Lopez-Fernandez et al. [13] explore authorization strategies in WebRTC, comparing access control lists (ACLs) and capability (CAP) models. While CAP is suitable for high-speed authorization, its lack of user-specific control raises security concerns in multi-user settings. BabelTalk's design may benefit from adopting ACLs for sensitive data handling, prioritizing user-specific control despite the added complexity to ensure privacy and data security.

Coupled Congestion Control for WebRTC

The congestion control mechanism in Islam et al.'s [14] work shows promise in improving stability under varying network conditions, making it ideal for real-world WebRTC applications. However, the implementation complexity and network-dependency of congestion algorithms present potential challenges. For BabelTalk, adopting a simplified congestion management approach may be more practical, allowing for balanced performance across diverse network conditions.

Multilingual Speech-to-Speech Translation

The direct speech-to-speech translation method by Kim et al. [22] introduces an innovative approach that bypasses text conversion, enhancing accuracy for low-resource languages. While this aligns well with BabelTalk's AI goals for real-time translation, the risk of semantic drift without text as an intermediary might affect translation quality for linguistically complex languages. BabelTalk could consider incorporating this approach selectively, using text intermediaries where higher accuracy is needed.

Quality of Experience in WebRTC

García et al. [24] provide valuable insights into measuring quality of experience (QoE) for WebRTC. Their proposed key performance indicators (KPIs) offer a structured framework for enhancing user satisfaction, which is critical for BabelTalk's goal of an optimized user experience. However, the high infrastructure demands of implementing QoE measurements suggest that BabelTalk may need a more streamlined approach, potentially focusing on essential KPIs to balance feasibility and performance.

Applications

BabelTalk is designed to address a variety of use cases in real-time communication by enhancing user engagement and productivity through AI-driven features. Here are some of the key applications:

Virtual team meetings and collaboration: Features such as persistent chat and secure file sharing further enhance team collaboration by allowing participants to share thoughts and resources during and after the meeting. To improve user experience, BabelTalk monitors KPIs like latency, bandwidth, and packet loss to ensure high-quality communication during meetings taking inspiration from García et al. [24].

Webinars and large-scale events: The summarization feature is especially useful in webinars, providing attendees with concise highlights of the key topics discussed.

Accessibility and language inclusivity: Meeting summaries generated by AI offer a convenient way for participants to quickly review key points and action items after a meeting, which is particularly useful for those who could not attend live.

Advantages of BabelTalk

BabelTalk's architecture provides multiple benefits, focusing on enhancing user experience, security, and scalability:

AI-driven and integrated by design: The inclusion of features such as live transcription, translation, and summarization helps increase productivity by allowing users to focus on discussions instead of taking notes.

Security and privacy: The use of Dockerized services for AI tasks (e.g., transcription, translation, summarization) ensures data isolation, providing an extra layer of security for sensitive information.

Scalability for different use cases: BabelTalk's quasi-peer architecture makes it scalable for various use cases-from small team meetings to larger webinars with many participants. This hybrid architecture allows it to handle both interactive sessions and passive audiences effectively.

Disadvantages of BabelTalk

While BabelTalk offers numerous advantages, there are some challenges and potential limitations that must be acknowledged:

Complexity of quasi-peer implementation: Introducing a quasi-peer into the architecture adds a layer of complexity compared to traditional P2P communication. Managing encryption keys, ensuring minimal latency, and securely handling temporary decryption require careful implementation, which could present challenges during deployment and scaling.

Increased resource requirements: Running multiple Dockerized AI services for tasks like transcription, translation, and summarization requires significant computing resources.

Dependency on AI model accuracy: The quality of AI-generated transcriptions, translations, and summaries heavily depends on the underlying models' accuracy. If the AI models fail to properly transcribe or translate due to poor audio quality or background noise, it may lead to misunderstandings and affect the user experience.

Latency concerns with real-time AI processing: While the quasi-peer is designed to minimize latency, performing tasks such as real-time transcription and translation introduces additional processing overhead.

Results

The initial phase of BabelTalk's development demonstrates core functionalities essential for real-time communication, forming the foundation for future AI-driven enhancements. This proof-of-concept (POC) implementation includes secure login capabilities, meeting creation and joining features, and a basic WebRTC architecture with Socket.IO for signaling. Below are the specific features showcased in the first phase.

Landing page

The BabelTalk landing page (Figure *4*) introduces users to a streamlined interface for easy access to the platform's services. Users can choose to log in via Google or with traditional username-password credentials, facilitated by Auth0 integration for secure authentication. This design prioritizes accessibility, setting the stage for a seamless entry into the meeting setup process.

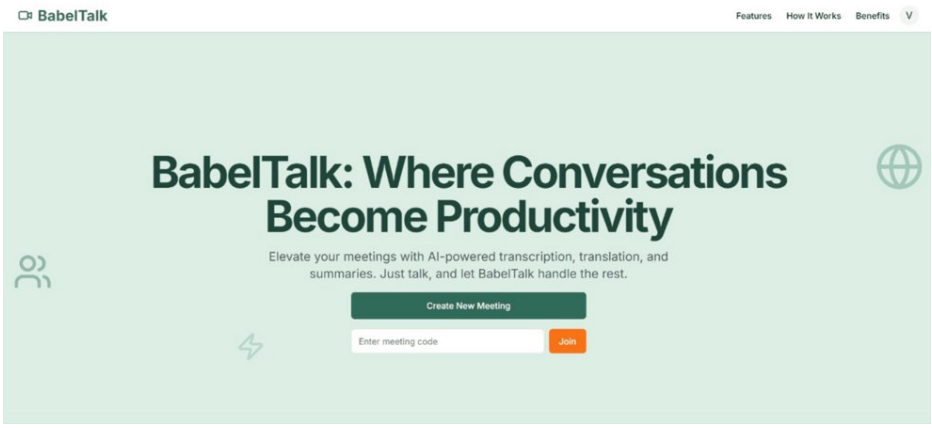

FIGURE 4: Landing Page

Login page with Auth0 integration

The login page (Figure 5) utilizes Auth0 to manage user authentication, supporting both Google SSO and username-password options. Auth0's secure and scalable authentication protocols ensure data privacy and protection, while simplifying access for users. This integration allows users to log in with confidence, knowing their credentials are managed by a trusted identity platform.

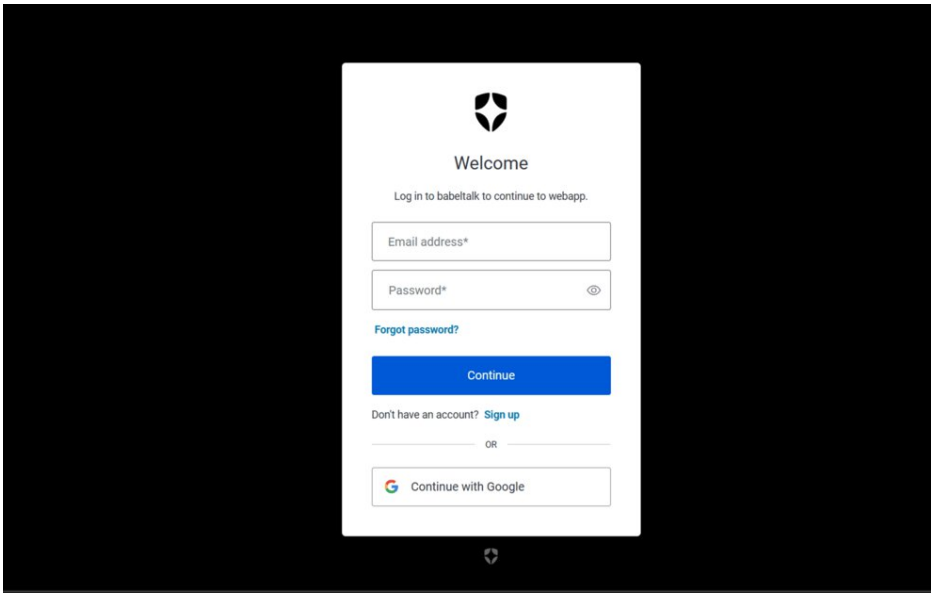

FIGURE 5: Login Page

Pre-meeting section for media configuration

The pre-meeting section (Figure 6) allows users to configure audio, video, and network settings before entering a meeting. This interface provides feedback on the status of connected devices, enabling users to test their microphone, camera, and network connection. By allowing for media configuration prior to the meeting, this section minimizes potential disruptions during sessions.

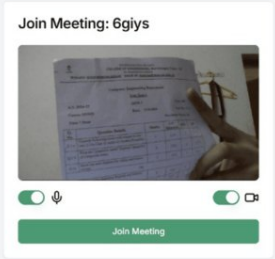

FIGURE 6: Pre-Meeting Page

In-meeting interface

The in-meeting interface (Figure 7) presents an intuitive layout with core controls for mute/unmute, video toggle, and screen sharing. The design mimics familiar video conferencing layouts, focusing on ease of use. At this POC stage, the architecture uses a basic WebRTC setup and Socket.IO for signaling, creating a reliable peer-to-peer communication environment. Future phases will expand upon this interface by adding AI-powered features like real-time transcription, translation, and advanced media synchronization.

FIGURE 7: In-Meet Interface

Discussion

The initial phase of BabelTalk demonstrates a functional proof-of-concept (POC) for a secure, real-time video conferencing platform built on WebRTC and integrated with Socket.IO for signaling. This phase successfully implements essential features such as user authentication through Auth0, Google login, and the ability to create and join meetings via shared links. The architecture, while foundational, highlights key considerations for scalability, usability, and security that will guide further development in the next phases. Here, we discuss the platform's current capabilities, the technical challenges addressed, and the anticipated enhancements to support AI-driven functionalities.

Platform usability and access

The implemented login and meeting interface provides an accessible, intuitive user experience. By integrating Auth0 for user authentication, BabelTalk minimizes the risks associated with handling user credentials directly, leveraging Auth0's security and identity management protocols [25]. Additionally, the inclusion of both Google SSO and traditional login options supports diverse user preferences and promotes ease of access. The pre-meeting configuration allows users to verify media devices before joining, thus improving the quality and reliability of in-meeting interactions.

Challenges with real-time communication and media synchronization

While WebRTC provides a robust foundation for peer-to-peer communication, integrating advanced AI functionalities in real-time environments poses specific challenges. Key considerations include maintaining low latency, achieving secure data handling, and synchronizing audio and video streams accurately. In this initial POC, Socket.IO is used to facilitate signaling, which offers reliable connection management and low-latency message delivery [26]. However, as AI features like live transcription, translation, and summarization are integrated, optimizing signaling and media synchronization will be essential to maintain high performance in large-scale applications.

Security and data privacy

Auth0's integration strengthens BabelTalk's security profile by externalizing authentication, which minimizes the risk of exposing sensitive user data. While this POC phase focuses on foundational security practices, future developments will expand security protocols to handle AI-driven features more securely. Planned Docker containerization of AI services will provide isolated environments for processing sensitive audio and video data, ensuring compliance with data privacy standards and enhancing user trust in the platform's handling of personal information.

Future directions for AI integration

The next stages of BabelTalk will introduce AI-driven features, including real-time transcription, multilingual translation, and automated meeting summaries. These functionalities require enhancements in media processing, data handling, and scalability beyond the capabilities of standard WebRTC implementations. The proposed quasi-peer architecture will support these AI features by adding a media server that performs tasks like stream synchronization, mixing, and real-time data processing. This architecture will enable BabelTalk to maintain low latency, high security, and seamless media integration, even as AI-driven functionalities are added.

Limitations and next steps

As this phase represents only the core functionality of BabelTalk, certain limitations remain. The POC currently operates on a basic WebRTC setup, which will require substantial upgrades to handle AI tasks at scale. In addition, the current signaling approach will be further optimized to support enhanced media synchronization and to reduce latency in AI-augmented sessions. Future iterations will address these areas by implementing the quasi-peer architecture, enabling BabelTalk to handle AI-driven features without compromising performance.

Conclusions

Incorporating advanced AI techniques alongside robust encryption methods holds immense potential to enhance both the functionality and security of WebRTC platforms like BabelTalk. This POC phase demonstrates foundational features such as secure user authentication with Auth0, Google login, and basic meeting capabilities, all built on a WebRTC architecture with Socket.IO signaling. As BabelTalk evolves, AI-powered functionalities - including real-time transcription, multilingual translation, and automated summarization - will play a vital role in fostering seamless, accessible communication across languages, contributing to a more inclusive user experience.

The planned quasi-peer architecture will further enable BabelTalk to overcome traditional WebRTC limitations by managing complex media synchronization, ensuring low latency, and providing a secure environment for processing sensitive data. By implementing Docker containerization for each AI service, BabelTalk will offer isolated, privacy-focused processing that upholds data security standards, thereby increasing user trust in the platform's handling of personal information. These advancements in AI and encryption not only provide BabelTalk users with valuable collaboration tools but also protect their privacy and sensitive information, reshaping the standards of secure, intelligent communication. As AI and encryption technologies continue to progress, we anticipate that WebRTC platforms will become increasingly efficient and secure, ultimately transforming how people connect and share information globally.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Samarth K. Patil, Vinayak Nigam, Shriyash Olambe

Acquisition, analysis, or interpretation of data: Samarth K. Patil, Vinayak Nigam, Shriyash Olambe, Anita Shinde

Drafting of the manuscript: Samarth K. Patil, Vinayak Nigam, Shriyash Olambe

Critical review of the manuscript for important intellectual content: Samarth K. Patil, Vinayak Nigam, Shriyash Olambe, Anita Shinde

Supervision: Anita Shinde

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Braesemann F, Stephany F, Teutloff O, Kässi O, Graham M, Lehdonvirta V: The global polarisation of remote work. PLoS ONE. 2022, 17:e0274630. 10.1371/journal.pone.0274630
2. Gunkel SNB, Hindriks R, El Assal KM, Stokking HM, Dijkstra-Soudarissanane S, ter Haar F, Niamut O: VRComm: An end-to-end web system for real-time photorealistic social VR communication. MMSys '21: Proceedings of the 12th ACM Multimedia Systems Conference. 2021, 65-79. 10.1145/3458305.3459595
3. Kasetwar A, Balani N, Damwani D, Pandey A, Sheikh A, Khadse A: A WebRTC-based video conferencing system with screen sharing. International Journal for Research in Applied Science and Engineering Technology. 2022, 10:5061-5066. 10.22214/ijraset.2022.43595
4. Tang D, Zhang L: Audio and video mixing method to enhance WebRTC. IEEE Access. 2020, 8:67228-67241. 10.1109/access.2020.2985412
5. Getchell KM, Carradini S, Cardon PW, Fleischmann C, Ma H, Aritz J, Stapp J: Artificial intelligence in business communication: The changing landscape of research and teaching. Business and Professional Communication Quarterly. 2022, 85:7-33. 10.1177/23294906221074311
6. W3C: WebRTC: Real-time communication in browsers. (2024). Accessed: October 8, 2024: https://www.w3.org/TR/webrtc/.
7. Raswa, Ghozali AL, Cahyanto KA: Interactive connectivity establishment protocol for WebRTC system with NAT traversal by manual signaling. Proceedings of the International Conference on Applied Science and Technology on Social Science 2022 (iCAST-SS 2022). 2022, 719:487-93. 10.2991/978-2-494069-83-1_87
8. Fakis A, Karopoulos G, Kambourakis G: Neither denied nor exposed: Fixing WebRTC privacy leaks. Future Internet. 2020, 12:92. 10.3390/fi12050092
9. Sredojev B, Samardzija D, Posarac D: WebRTC technology overview and signaling solution design and implementation. 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2015, 1006-1009. 10.1109/mipro.2015.7160422
10. Kalil Al-Khayyat AT, Mahmood SA: Peer-to-peer media streaming with HTML5. International Journal of Electrical and Computer Engineering (IJECE). 2023, 13:2356-2356. 10.11591/ijece.v13i2.pp2356-2362
11. Duan Q, Liang Z: File sharing strategy based on WebRTC. 2016 13th Web Information Systems and Applications Conference (WISA). 2016, 3-6. 10.1109/WISA.2016.11
12. Deepstream.io: WebRTC: File transfer. (2019). Accessed: August 28, 2024: https://deepstream.io/tutorials/webrtc/webrtc-file-transfer/.
13. López-Fernández L, Gallego M, García B, Fernández-López D, Lópe FJ: Authentication, authorization, and accounting in WebRTC PaaS infrastructures: The case of Kurento. IEEE Internet Computing. 2014, 18:34-40. 10.1109/mic.2014.102
14. Islam S, Welzl M, Fladby T: Real-life implementation and evaluation of coupled congestion control for WebRTC media and data flows. IEEE Access. 2022, 10:95046-95066. 10.1109/ACCESS.2022.3206041
15. Alahmadi M, Počta P, Melvin H: An adaptive bitrate switching algorithm for speech applications in context of WebRTC. ACM Transactions on Multimedia Computing Communications and Applications. 2021, 17:1-21. 10.1145/3458751
16. Zhu Q-s, Zhang J, Wu M-h, Fang X, Dai L-R: An improved Wav2Vec 2.0 pre-training approach using enhanced local dependency modeling for speech recognition. Interspeech. 2021, 4334-4338. 10.21437/interspeech.2021-67
17. Chen L, Asgari M, Dodge HH: Optimize Wav2vec2s architecture for small training set through analyzing its pre-trained models attention pattern. ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2022, 7112-7116. 10.1109/icassp43922.2022.9747831
18. Fu Y, Kang Y, Cao S, Ma L: DistillW2V2: A small and streaming Wav2vec 2.0 based ASR model [PREPRINT]. arXiv:2303.09278. 2023, 10.48550/arXiv.2303.09278
19. Wang Y, Mohamed A, Le D, Liu C, Xiao A, Mahadeokar J: Transformer-based acoustic modeling for hybrid speech recognition. ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2020, 6874-6878. 10.1109/ICASSP40776.2020.9054345
20. Gulati A, Qin J, Chiu C-C, et al.: Conformer: Convolution-augmented transformer for speech recognition. Interspeech. 2020, 5036-504. 10.21437/interspeech.2020-3015
21. Zhang Y, Lv Z, Wu H, et al.: MFA-Conformer: Multi-scale feature aggregation conformer for automatic speaker verification [PREPRINT]. arXiv:2203.15249. 2022, 10.48550/arXiv.2203.15249
22. Kim M, Choi J, Kim D, Ro YM: Textless unit-to-unit training for many-to-many multilingual speech-to-speech

translation. IEEE/ACM Transactions on Audio Speech and Language Processing. 2024, 32:3934-3946. 10.1109/taslp.2024.3444470

23. Ding Y, Li H, Ni H, Yang Z: Generative text summary based on enhanced semantic attention and gain-benefit gate. IEEE Access. 2020, 8:92659-92668. 10.1109/access.2020.2994092
24. García B, Gallego M, Gortázar F, Bertolino A: Understanding and estimating quality of experience in WebRTC applications. Computing. 2019, 101:1585-1607. 10.1007/s00607-018-0669-7
25. Auth0. (2024). Accessed: August 28, 2024: https://auth0.com/.
26. Socket.IO. (2024). Accessed: August 28, 2024: https://socket.io/.