Integrating AI and Encryption in WebRTC: A Smart Video Conferencing Platform with Real-Time Transcription and Translation

Samarth Patil[1], Vinayak Nigam[2], Shriyash Olambe[3] and Ms. Trupti G.Ghongade[4]

[1]Samarth Patil, Computer Engineering, Marathwada Mitra Mandal's College of Engineering
[2]Vinayak Nigam,Computer Engineering, Marathwada Mitra Mandal's College of Engineering
[3]Shriyash Olambe, Computer Engineering, Marathwada Mitra Mandal's College of Engineering
[4]Ms. Trupti G.Ghongade,Computer Engineering, Marathwada Mitra Mandal's College of Engineering

---***---

Abstract - As the demand for seamless, real-time communication tools grows, traditional video conferencing platforms need help to balance simplicity with advanced features. BabelTalk aims to address this challenge by integrating AI-powered capabilities into a scalable, secure, real-time communication platform built on WebRTC. WebRTC relies on basic peer-to-peer architectures, which limit scalability and also lack pre-processing of media streams, BabelTalk takes inspiration from a quasi-peer architecture to enable advanced features such as real-time transcription, translation into captions or speech, and post-meeting summaries while ensuring secure media transmission. The platform prioritizes data privacy and offers a feature-rich communication experience tailored for small to medium-sized teams by utilizing isolated Dockerized services for temporary decryption during AI processing.

***Key Words*:** WebRTC, synchronization mechanism, media synchronization, congestion control, Speech-to-speech translation (S2ST), Text-to-speech synthesis (T2S), Unit-to-unit translation (UTUT)

INTRODUCTION

The demand for seamless, real-time communication platforms has surged in recent years, driven by the rise of remote work, global collaboration, and virtual events [1]. Platforms such as Google Meet and Microsoft Teams have become essential tools for online meetings, yet these services have either complex feature sets for general meetings or limited functionality when it comes to integrating advanced features, particularly those leveraging artificial intelligence (AI) for tasks like transcription, translation, and meeting summarization [2]. Typically, these platforms were not originally designed to be AI-driven, leading to suboptimal integration of such features, which can degrade performance or compromise user experience.

WebRTC (Web Real-Time Communication) has emerged as the foundational technology that enables real-time audio, video, and data transmission between peers directly in browsers without plugins [3]. Its ability to establish secure, low-latency connections has made it the backbone of many modern conferencing solutions. However, while WebRTC offers secure and efficient peer-to-peer communication through protocols such as DTLS (Datagram Transport Layer Security) and SRTP (Secure Real-Time Transport Protocol), it faces challenges when scaling to handle tasks such as audio/video synchronization, AI-powered live transcription, and stream mixing [4].

In recent years, the integration of AI in communication platforms has shifted from an optional feature to a core necessity[9]. Live transcription, translation into multiple languages, and intelligent summarization are no longer just enhancements—they are critical features that can dramatically improve accessibility, user engagement, and productivity during meetings. However, most platforms today face the challenge of retrofitting AI into an architecture that was not built to handle such tasks natively. This often leads to high latency, compromised security, and subpar user experiences.

BabelTalk addresses these issues by introducing a quasi-peer architecture. In contrast to purely peer-to-peer (P2P) models, the quasi-peer serves as a media server responsible for synchronizing, mixing, and processing audio and video streams. This architecture enables real-time AI-driven features such as transcription and translation, all while maintaining high security and minimizing latency. The quasi-peer temporarily decrypts media streams to perform tasks like generating meeting transcriptions, which are subsequently used for live translations and post-meeting summaries. Each AI service—whether transcription, translation, or summarization—is dockerized to provide secure isolation, ensuring that sensitive media data is handled in a controlled environment.

In this paper, we present the methodology and design of BabelTalk, highlighting how WebRTC's real-time communication features are extended through the quasi-peer to support AI-based transcription, translation, and summarization services. We will also discuss the challenges of media synchronization, security, and scaling, and how BabelTalk's architecture overcomes these to deliver a seamless, feature-rich communication platform.

1.1 Objectives of the study

The primary objective of this study is to develop and evaluate an AI-integrated WebRTC-based video conferencing platform (BabelTalk) with enhanced security, real-time processing, and scalability. The key objectives are as follows:
To implement a secure, low-latency WebRTC architecture
Utilize ICE, STUN, and TURN servers for optimal connectivity. Ensure end-to-end encryption (E2EE) using DTLS-SRTP to protect audio, video, and data streams.
To integrate AI-driven transcription, translation, and summarization
Implement Automatic Speech Recognition (ASR) using transformer-based models (e.g., Whisper, wav2vec 2.0).
Enable real-time multilingual speech-to-text translation for global accessibility. Develop AI-generated meeting summaries for post-call documentation.
To evaluate the performance and efficiency of WebRTC's RTCDataChannel Measure latency, throughput, and reliability for real-time file transfers. Optimize SDP bandwidth constraints to enhance large file transmission.

To implement a quasi-peer model for scalable AI processing Introduce a dedicated quasi-peer node to handle AI computations without exposing sensitive data.
Ensure containerized execution of AI models for privacy and efficiency.
To analyze the scalability, network adaptation, and security of BabelTalk Assess WebRTC's adaptive bitrate control and congestion management in varying network conditions.Compare BabelTalk's performance and security with traditional centralized video conferencing platforms.

By achieving these objectives, this study aims to demonstrate an innovative, AI-powered, and secure WebRTC framework that enhances real-time communication while maintaining privacy, efficiency, and scalability.

Related Work (Literature Survey)

The development of BabelTalk, an AI-powered video conferencing platform leveraging WebRTC, builds upon a rich body of research in real-time communication, media processing, AI-driven transcription and translation, and secure data transfer. Tang and Zhang [4] explored the challenge of mixing audio and video streams in WebRTC-based real-time communication systems. Their methodology employed a media server to process and distribute mixed media streams, with a particular emphasis on optimizing audio-video synchronization to ensure seamless interactions. The study found that this approach significantly enhanced communication quality, particularly in multi-participant conferencing scenarios, by delivering clear and synchronized audio-visual outputs. The advantages of their method include improved audio and video quality and scalability for large meetings, making it suitable for enterprise-level applications. However, the increased complexity of managing real-time mixing and the resource-intensive nature of the media server pose challenges, especially for low-power devices such as mobile phones or budget laptops. Performance was evaluated through metrics like latency, synchronization accuracy, and resource usage, providing a benchmark for assessing media processing efficiency.

In the realm of network optimization, another study [23] investigated coupled congestion control for WebRTC media and data flows. The authors implemented congestion control algorithms to regulate traffic, conducting real-life evaluations under diverse network conditions, such as high packet loss or fluctuating bandwidth. Their findings demonstrated improved efficiency and stability in WebRTC flows, ensuring consistent performance even in suboptimal network environments. This approach offers the advantage of stable media and data transmission, with efficient bandwidth utilization that is applicable to real-world scenarios like remote collaboration. However, the complexity of implementing and tuning these algorithms, combined with their dependence on network conditions, presents significant drawbacks. Performance was measured through throughput, packet loss, and delay, offering insights into the trade-offs between stability and implementation overhead.

Focusing on user experience, a study [24] addressed the measurement of Quality of Experience (QoE) in WebRTC applications. The researchers proposed a set of Key Performance Indicators (KPIs) to estimate QoE and analyzed the impact of various WebRTC topologies, such as peer-to-peer and server-mediated architectures. Their findings highlighted key factors affecting QoE, such as latency and packet loss, and provided a framework for developers to enhance user satisfaction. The emphasis on user-centric metrics and the comprehensive analysis of different topologies are notable strengths, enabling practical improvements in application design. However, QoE measurement remains subjective and challenging to quantify precisely, and the infrastructure required to collect and analyze KPIs can be complex and costly. The study used KPIs related to network conditions (latency, bandwidth, packet loss) and user satisfaction, establishing a foundation for evaluating user experience in real-time communication platforms.

In the domain of AI-driven communication, Kim et al. [7] proposed a groundbreaking method for multilingual speech-to-speech translation that bypasses text conversion, directly mapping speech sounds between languages. Targeting low-resource languages, their approach utilized transformer-based models with self-attention mechanisms to process audio signals, trained on a large corpus of multilingual speech data. increases security risks, and ACLs' detailed checks slow.

Reference [4]: Audio and Video Stream Mixing with WebRTC

Methodology: The authors developed a method to mix audio and video streams for real-time communication using WebRTC. A media server processes and distributes mixed media streams, with a focus on optimizing audio-video synchronization to ensure seamless real-time interaction.

Key Findings: The approach significantly improved communication quality, particularly in multi-participant conferencing scenarios, by enhancing audio and video synchronization.

Advantages:

Enhanced audio and video quality in multi-participant settings.

Scalable for large meetings, accommodating more participants effectively.

Disadvantages:

Increased complexity in managing real-time mixing processes.

Resource-intensive, posing challenges for low-power devices.

Performance Metrics Used: Latency, synchronization accuracy, and resource usage were measured to evaluate performance improvements.

Reference [23]: Coupled Congestion Control in WebRTC

Methodology: The study implemented a coupled congestion control mechanism for both media and data flows in WebRTC. Congestion control algorithms were used to regulate traffic, with real-life evaluations conducted under varying network conditions.

Key Findings: The coupled congestion control improved the efficiency and stability of WebRTC media and data flows, even in fluctuating network environments.

Advantages:

Stable transmission for both media and data.

Efficient bandwidth utilization, suitable for real-world applications.

Disadvantages:

Complex implementation and tuning of congestion algorithms.

Performance heavily reliant on network conditions.

Performance Metrics Used: Throughput, packet loss, and delay were measured to assess the congestion control mechanisms' effectiveness.

Reference [24]: Quality of Experience (QoE) in WebRTC

Methodology: The authors focused on measuring Quality of Experience (QoE) in WebRTC applications by proposing a set of Key Performance Indicators (KPIs). They analyzed the impact of different WebRTC topologies on QoE.

Key Findings: The study identified key factors influencing QoE and provided a framework for measurement, offering developers actionable insights to improve user experience.

Advantages:

Emphasis on user satisfaction with practical metrics.

Comprehensive analysis of various WebRTC topologies.

Disadvantages:

QoE measurement is subjective and difficult to quantify precisely.

Requires complex infrastructure to collect and analyze KPIs.

Performance Metrics Used: KPIs related to network conditions (latency, bandwidth, packet loss) and user satisfaction were used to evaluate QoE.

Reference [7]: Multilingual Speech-to-Speech Translation

Methodology: Introduced a novel method for multilingual speech-to-speech translation that skips text conversion, directly mapping speech sounds between languages. The approach targets low-resource languages for improved accuracy.

Key Findings: The model demonstrated significant improvements in multilingual translation, particularly for low-resource language pairs, showing flexibility and effectiveness.

Advantages:

Scalable for low-resource languages, supporting diverse linguistic contexts.

Eliminates reliance on intermediate text transcripts.

Disadvantages:

Challenges in maintaining semantic consistency across languages.

Performance may degrade with tonal or vastly different languages.

Performance Metrics Used: BLEU scores, perplexity, and unit error rates were used to assess translation precision and model generalization.

Reference [11]: One-to-Many File Transfer Efficiency

Methodology: Proposed a method to improve one-to-many file transfers by dividing files into blocks, distributing them to receivers, and reassembling them. Receivers forward blocks to others, leveraging participants' network resources.

Key Findings: File transfer time decreased with more workers (up to 50), but efficiency declined beyond this due to overhead from managing multiple data channels.

Advantages:

Faster transfers by utilizing peer bandwidth.

Secure, encrypted peer-to-peer connections enhance privacy.

Eliminates server storage, reducing server load.

Disadvantages:

Slower initial transfer due to fewer active senders.

Performance depends on network stability; slow peers impact speed.

Overhead in large networks with many peers.

Performance Metrics Used: Total transfer time (adjusted for peer count), scalability (number of peers), and bandwidth usage were measured.

Reference [14]: SecureLink with WebRTC

Methodology: Focused on using WebRTC for real-time communication (video calls, chat, file sharing) without plugins. Introduced a WebSocket-based signaling method and discussed future AI-driven features like transcription and translation.

Key Findings: SecureLink enables fast, secure peer-to-peer communication without central servers, with potential for AI enhancements to improve functionality.

Advantages:

Fast, serverless communication with encrypted data.

Flexible architecture for integrating future AI features.

Disadvantages:

Security risks if peers are compromised, despite encryption.

Performance depends on network quality; poor connections cause disruptions.

AI feature integration may require significant time and effort.

Performance Metrics Used: Latency, throughput, network stability, and encryption security were evaluated.

Reference [13]: WebRTC Signaling and Authorization

Methodology: Highlighted WebRTC's lack of defined signaling methods, proposing WebSockets for improved session control over SIP or XMPP. Compared ACLs (detailed control) and CAP (token-based) for authorization.

Key Findings: WebSockets provide flexible, efficient signaling; ACLs offer better control, while CAP is faster but less secure in user-specific scenarios.

Advantages:

WebSockets enhance flexibility and efficiency in session control.

CAP is fast for real-time applications.

ACLs provide fine-grained, user-specific permission control.

Disadvantages:

CAP lacks user-specific control, reducing security in multi-user settings.

ACLs are slower due to detailed permission checks.

Performance Metrics Used: Authorization speed (in microseconds) and response times under different loads were measured.

Reference [8]: Text Summarization with Semantic Attention

Methodology: Introduced a text summarization method combining enhanced semantic attention with a gain-benefit gate mechanism to emphasize important details and reduce repetition.

Key Findings: The model outperformed existing summarization methods in semantic relevance and informativeness, particularly for longer, complex documents.

Advantages:

Improved attention allocation enhances summary quality.

Balances key information capture and readability.

Disadvantages:

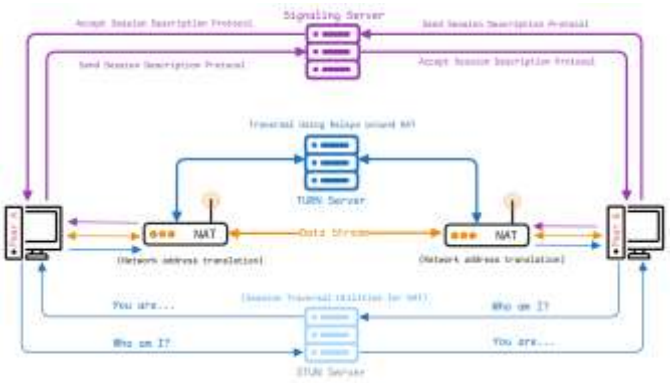
High computational costs limit real-time applicability.

Struggles with highly abstract or domain-specific texts.

Performance Metrics Used: ROUGE scores (precision, recall, F1-measure) and semantic similarity metrics assessed summary quality and coherence.

Proposed Methodology

(3.1) WebRTC (Web Real-Time Communication) serves as the foundation for enabling real-time communication in NexTalk, allowing direct browser-to-browser connections without requiring extra plugins [21]. This technology provides the APIs necessary for sharing audio, video, and data, enabling encrypted, low-latency communication over the internet. NexTalk leverages WebRTC for video conferencing, voice calls, and data transfer, incorporating key features for connection management and security.


Signaling Server
Accept Session Description Protocol
Send Session Description Protocol
Send Session Description Protocol
Accept Session Description Protocol
Traversal Using Relays around NAT
TURN Server
NAT
(Network address translation)
Data Stream
NAT
(Network address translation)
(Session Traversal Utilities for NAT)
You are...
Who am I?
Who am I?
You are...
STUN Server

To establish secure peer-to-peer (P2P) connections, WebRTC relies on several important components:

(3.1.1) ICE (Interactive Connectivity Establishment) helps determine the best route for connection between peers by gathering possible candidates, such as IP addresses and ports [22]. ICE may use **STUN (Session Traversal Utilities for NAT)** to assist peers in discovering their public IP addresses when they are behind a NAT. If direct connections aren't possible due to network restrictions, **TURN (Traversal Using Relays around NAT)** servers are used to relay the media streams [14].

(3.1.2) DTLS (Datagram Transport Layer Security) is employed to secure key exchange and signaling, ensuring that the process of setting up connections is protected from any unauthorized access [14].

(3.1.3) SRTP (Secure Real-time Transport Protocol) is used to encrypt the actual media streams being shared, ensuring that all audio and video data remains private and secure during transmission [14].

(3.2) Peer Connection Setup:

WebRTC relies on **RTCPeerConnection**, 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.

(3.3) Signaling and SDP (Session Description Protocol):

- **Signaling:** WebRTC requires an external signaling mechanism to exchange connection details between peers (such as IP addresses and ports)[5]. 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.
- **SDP (Session Description Protocol):** 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 (Session Description Protocol) 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:1638400, you can raise the transfer speed to 1.6 Gbps. This small change makes a huge difference, speeding up large file transfers significantly.[12]

(3.4) Establishing Media Channels:

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

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

(3.5) 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 peer-to-peer 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: **Sender-Side**: The file remains intact in memory as the sender slices and sends it in chunks.**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.

(3.6) Security:

1. **Signaling and Key Exchange**: **Diffie-Hellman key exchange** generates a **shared secret key** between peers through a secure signaling server.
2. **Key Derivation**: A **Key Derivation Function (KDF)** is used to generate distinct keys: a **media encryption key** for E2EE and a **session-specific key** for processing at the quasi-peer.
3. **Media Encryption (SRTP)**: Media streams are encrypted using **SRTP** with the **media encryption key** for secure communication between peers.
4. **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.
5. **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.
6. **Final Decryption by Peers**: The peers **decrypt the re-encrypted stream** using the **session-specific key** to access the final media.

4 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. BabelTalk leverages WebRTC's built-in bandwidth estimation and congestion control algorithms to ensure that meetings run smoothly, even on low-bandwidth connections.[10]
- **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.

5 Incorporation of Quasi-Peer in the Communication Architecture

(5.1) Overview:

To address the limitations of purely peer-to-peer (P2P) WebRTC communication, 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.

While WebRTC enables encrypted, low-latency, peer-to-peer 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.

(5.2) 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.

(5.2.1) Steps in Media Reception:

1. **Receiving Media Streams:**
   - The quasi-peer receives encrypted media streams from all participating peers through **WebRTC ICE connections**.
   - To process the media for AI tasks (like transcription or translation), the quasi-peer temporarily decrypts the streams using a session-specific key.
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]

6 AI features in BabelTalk:-

(6.1) Real-Time Transcription of Spoken Words:

Transformers in 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 [18].

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 [19].

Speech Data Processing: Transformes, 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 [20].

(6.2) 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. 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 [16].
- 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 [15] and [17].
- **Many-to-Many Multilingual Models in ASR:** The methodology outlined by [7] emphasizes the benefits of employing many-to-many multilingual models in automatic speech recognition (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.

(6.3) 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 in [8], 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.

- **Enhanced Semantic Attention in Summary Generation:** The way of focusing on what meaning is, 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.
- **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.

(6.4) 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.

(6.4.1) The transcription in real time.

Purpose: Employs Neural network models used in NMT learn how to link input and output sequences, making them ideal for dealing with the complexities of human language. to convert the text into the preferred language.

(6.4.2) 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.

(6.4.3) 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.

(6.4.4) 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 TTS.

Distribution of Meeting Summaries: Participants receive the summary created at the end of the meeting through a secure channel

(6.4.5) Workflow Example

- Attendees join the BabelTalk session, and their audio is recorded.
- The quasi-peer node decrypts the media streams and forwards the audio to the transcription service.
- The transcription service converts the audio into text live, which is then forwarded to the translation service.
- The transcriber service changes the recorded material into another language.

7 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.

(7.1) 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.
- **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.

(7.2) 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.
- **Video Processing:** The quasi-peer decodes the video streams into the **planar YUV 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).
- **Hardware Acceleration:** For efficient video processing, hardware acceleration techniques such as **MMX2**, **SSE2 Fast**, or **SSSE3** are employed, depending on the server hardware capabilities.

(7.3) Broadcasting the Mixed Stream

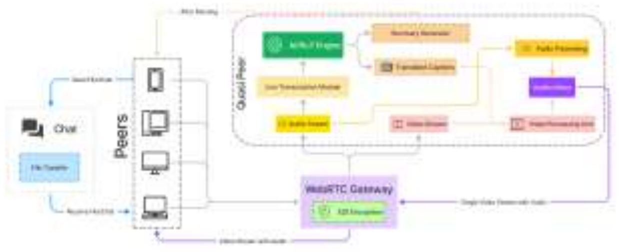
After processing and mixing, the quasi-peer is responsible for broadcasting the mixed media stream back to the participants or a larger audience.

(7.4) 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.

(7.5) 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.



8 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:

(8.1) 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.

(8.2) Webinars and Large-Scale Events: The summarization feature is especially useful in webinars, providing attendees with concise highlights of the key topics discussed [Reference needed].

(8.3) 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 [Reference needed].

9 Advantages of BabelTalk

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

(9.1) 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 [Reference needed].

(9.2) 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 [Reference needed].

(9.3) 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 [Reference needed].

10 Disadvantages of BabelTalk

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

(10.1) 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 [Reference needed].

(10.2) Increased Resource Requirements: Running multiple **Dockerized AI services** for tasks like transcription, translation, and summarization requires significant computing resources.

(10.3) 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 [Reference needed].

(10.4) 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.

CONCLUSIONS

To sum up, incorporating new AI techniques with advanced encryption methods can greatly improve both the functionality and security of WebRTC platforms. AI-powered functions like instant transcription, interpretation, and summarization facilitate smooth interaction among different languages, enhancing user satisfaction and inclusivity. At the same time, strong encryption methods guarantee that confidential information sent during communication stays protected from unauthorized entry. These advancements provide users with useful collaboration tools while also protecting their privacy and information. With the advancement of AI and encryption technologies, we anticipate that WebRTC platforms will become more efficient and secure, changing how we connect and share information worldwide.

ACKNOWLEDGEMENT

We express our heartfelt gratitude to all those who have contributed to the successful completion of this research on BabelTalk, an AI-powered video conferencing platform. We are deeply indebted to our academic supervisor, Ms. Trupti G.Ghongade, from Marathwada Mitra Mandal's College of Engineering, for their invaluable guidance, insightful feedback, and unwavering support throughout the research process. Their expertise in real-time communication systems and artificial intelligence was instrumental in shaping the direction of this project. We also extend our sincere appreciation to the faculty and staff of Marathwada Mitra Mandal's College of Engineering for providing access to essential resources, including computing facilities and research laboratories, which enabled us to develop and test the BabelTalk platform. Special thanks go to our technical collaborators, for their expertise in WebRTC implementation and AI model integration, which significantly enhanced the platform's functionality. We are grateful to our peers and colleagues for their constructive feedback during the design and evaluation phases, which helped refine our methodology and improve the manuscript. Finally, we thank our families and friends for their encouragement and patience, which motivated us to pursue this ambitious project. This work would not have been possible without the collective contributions of all these individuals and entities.

REFERENCES

[1] Braesemann, F., Stephany, F., Teutloff, O., Kässi, O., Graham, M., & Lehdonvirta, V. (2021). The global polarisation of remote work. PLoS ONE, 17. https://doi.org/10.1371/journal.pone.0274630.
[2] Gunkel, S., Hindriks, R., Assal, K., Stokking, H., Dijkstra-Soudarissanane, S., Haar, F., & Niamut, O. (2021). VRComm: an

end-to-end web system for real-time photorealistic social VR communication. Proceedings of the 12th ACM Multimedia Systems Conference. https://doi.org/10.1145/3458305.3459595.

[3] Kasetwar, A., Balani, N., Damwani, D., Pandey, A., Sheikh, A., & Khadse, A. (2022). A WebRTC-Based Video Conferencing System with Screen Sharing. International Journal for Research in Applied Science and Engineering Technology. https://doi.org/10.22214/ijraset.2022.43595.

[4] Tang, D., & Zhang, L. (2020). Audio and Video Mixing Method to Enhance WebRTC. IEEE Access, 8, 67228-67241. https://doi.org/10.1109/ACCESS.2020.2985412.

[5] Al-Khayyat, A., & Mahmood, S. (2023). Peer-to-peer media streaming with HTML5. International Journal of Electrical and Computer Engineering (IJECE). https://doi.org/10.11591/ijece.v13i2.pp2356-2362.

[6] Ghozali, A. L., & Cahyanto, K. A. (2022, December). Interactive Connectivity Establishment Protocol for WebRTC System with NAT Traversal by Manual Signaling. In International Conference on Applied Science and Technology on Social Science 2022 (iCAST-SS 2022) (pp. 487-493). Atlantis Press.

[7] M. Kim, J. Choi, D. Kim, and Y. M. Ro, "Textless Unit-to-Unit Training for Many-to-Many Multilingual Speech-to-Speech Translation," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 32, pp. 3934-3946, 2024, doi: 10.1109/TASLP.2024.3444470.

[8] Ding, Y. Li, H. Ni and Z. Yang, "Generative Text Summary Based on Enhanced Semantic Attention and Gain-Benefit Gate," in IEEE Access, vol. 8, pp. 92659-92668, 2020, doi: 10.1109/ACCESS.2020.2994092.

[9] Getchell, K. M., Carradini, S., Cardon, P. W., Fleischmann, C., Ma, H., Aritz, J., & Stapp, J. (2022). Artificial Intelligence in Business Communication: The Changing Landscape of Research and Teaching. Business and Professional Communication Quarterly, 85(1), 7-33. https://doi.org/10.1177/23294906221074311

[10] Alahmadi, M., Počta, P., & Melvin, H. (2021). An Adaptive Bitrate Switching Algorithm for Speech Applications in Context of WebRTC. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 17, 1 - 21. https://doi.org/10.1145/3458751.

[11] Q. Duan and Z. Liang, "File Sharing Strategy Based on WebRTC," 2016 13th Web Information Systems and Applications Conference (WISA), Wuhan, China, 2016, pp. 3-6, doi: 10.1109/WISA.2016.11.

[12]WebRTC:FileTransfer(deepstream.io)- https://deepstream.io/tutorials/webrtc/webrtc-file-transfer/

[13] Lopez-Fernandez, L., Gallego, M., Garcia, B., Fernandez-Lopez, D., & Lopez, F. J. (2014). Authentication, Authorization, and Accounting in WebRTC PaaS Infrastructures: The Case of Kurento. IEEE Internet Computing, 18(6), 34–40. doi:10.1109/mic.2014.102

[14] Sredojev, B., Samardzija, D., & Posarac, D. (2015). “WebRTC technology overview and signaling solution design and implementation.” 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). doi:10.1109/mipro.2015.7160422

[15]Gulati, A., Qin, J., Chiu, C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., & Pang, R. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition. ArXiv, abs/2005.08100. https://doi.org/10.21437/interspeech.2020-3015.

[16]Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., Fuegen, C., Zweig, G., & Seltzer, M. (2020). Transformer-Based Acoustic Modeling for Hybrid Speech Recognition. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/ICASSP40776.2020.9054345.

[17]Zhang, Y., Lv, Z., Wu, H., Zhang, S., Hu, P., Wu, Z., Lee, H., & Meng, H. (2022). MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification. , 306-310. https://doi.org/10.48550/arXiv.2203.15249.

[18]Zhu, Q., Zhang, J., Wu, M., Fang, X., & Dai, L. (2021). An Improved Wav2Vec 2.0 Pre-Training Approach Using Enhanced Local Dependency Modeling for Speech Recognition. , 4334-4338. https://doi.org/10.21437/interspeech.2021-67.

[19]Chen, L., Asgari, M., & Dodge, H. (2022). 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), 7112-7116. https://doi.org/10.1109/icassp43922.2022.9747831.

[20]Fu, Y., Kang, Y., Cao, S., & Ma, L. (2023). DistillW2V2: A Small and Streaming Wav2vec 2.0 Based ASR Model. ArXiv, abs/2303.09278. https://doi.org/10.48550/arXiv.2303.09278.

[21] WebRTC: Real-Time Communication in Browsers. Accessed: Sep. 17, 2024. [Online]. Available: https://www.w3.org/TR/webrtc/

[22] Fakis, A., Karopoulos, G., & Kambourakis, G. (2020). Neither Denied nor Exposed: Fixing WebRTC Privacy Leaks. Future Internet, 12, 92. https://doi.org/10.3390/fi12050092.

[23]S. Islam, M. Welzl, and T. Fladby, “Real-Life Implementation andEvaluation of Coupled Congestion Control for WebRTC Media and DataFlows,” IEEE Access, vol. 10, pp. 95 046–95 066, 2022

[24]García, B., Gallego, M., Gortázar, F. et al. Understanding and estimating the quality of experience in WebRTC applications. Computing 101, 1585–1607 (2019). https://doi.org/10.1007/s00607-018-0669-7