Advanced Video Conferencing Application

Nikita Orlov, Radoslav Vargic
Slovak University of Technology , Faculty of Electrical Engineering and Information Technology, Ilkovičova 3
841 04 Bratislava
xorlovn@stuba.sk

Abstract - **This document presents the development of an advanced video conferencing application that integrates unique functionalities not commonly available in current mainstream systems such as Zoom or Google Meet. The application enables multi-camera support, dynamic camera manipulation, role-based access control, real-time chat features, and conference customization. Built with Jitsi and WebRTC protocols, the system ensures secure and scalable video communication. The backend and frontend are combined in a single repository, and the application is containerized using Docker for easy deployment.(Abstract)**

Keywords - Video conferencing; Codec; WebRTC; Jitsi; Real-time communication; Docker; Multi-camera support; Role-based access

I. INTRODUCTION

This Video conferencing systems are essential for work, study, and many other purposes. Currently, there are many such systems on the market (Zoom, Microsoft Teams, Google Meet). I decided to create a video conferencing system with additional unique features that are not yet available in existing systems.

II. ANALYSIS

A. How video conferencing systems work

- Capturing video from the user's camera
- Frame processing
- Compression
- Transmission of video and audio to other users over the network
- Decoding and display on the other users' side

B. The principle of how a codec works

The Codec – is a device that compresses (on the sender's side) and decompresses (on the receiver's side) media files (video or audio).

Reasons for use: Efficient data transmission , reduced storage costs , improved transmission quality , reduced file size

The main function of a codec is video encoding (encoding includes compression). Video encoding often also involves audio encoding (a video file usually contains not only image frames but also sound).

Types of compression based on frame processing:

- Intra-frame: Compresses each video frame individually, searching for redundant data within the frame to reduce the data volume.
- Inter-frame: It encodes only the differences between successive frames, offering stronger compression than intra-frame compression.

Types of compression based on data storage:

• Lossy: Saves space - quality is reduced to maximize file size efficiency.

• Lossless: Preserves as much detail of the original as possible, so the resulting file size is usually larger

C. Comparison of Popular Video Codecs

1) H.264 / AVC (Advanced Video Coding):

The most widely used encoding format, offering high video quality with low data rates. It provides a good balance bet ween streaming quality and compression but is subject to licensing fees and requires more computing power.
Used by: Netflix, Hulu, YouTube, Zoom, Microsoft Teams, Cisco Webex, Jitsi (a single video call system can use multiple codecs) [1].

2) H.266 / VVC (Versatile Video Coding):

Requires about half the data to transmit video at the same quality. More efficient for high-resolution video streaming but demands significant computational resources. Not yet used in video conferencing systems due to being relatively new and requiring high processing power [2].

3) VP8 (Open-source):

Free to use with no licensing fees, supported by many browsers and platforms. However, encoding is more CPU-intensive compared to some other codecs.
Used by: Google Meet, WebRTC, Daily.co, 8x8 [3].

III. DETAILED ANALYSIS OF EXISTING VIDEO CONFERENCING SYSTEMS

Zoom

Zoom is one of the most widely used video conferencing systems. It uses a hybrid architecture that combines the advantages of both client-server and distributed network models [4]:

Client Side:
Can run on various devices such as Windows, macOS, iOS, Android, and web browsers. Captures video and audio using hardware or software encoding:

1) Hardware encoding – if the device has a built-in codec (e.g., GPU).

2) Software encoding – if the system relies on the CPU.

After encoding, the data is sent to the server.

Server Side:

Utilizes a hybrid cloud infrastructure that enables easy scaling for different workloads. Video and /audio are transmitted between participants via a distributed network of data centers around the world. Companies can also deploy on-premise Zoom servers using Zoom Meeting Connector, which offers an extra level of security.

Video Stream Processing Methods:

1) SFU (Selective Forwarding Unit): Forwards video streams between users without processing.

2) MCU (Multipoint Control Unit): Merges multiple video streams and sends a single stream to less powerful clients.

Codec Selection: Zoom selects the codec based on several factors: Device capabilities, network conditions, type of content ,compatibility between participants , administrative settings

After connecting, the client sends its technical capabilities to the server. The server then analyzes the working conditions of all participants and selects the optimal strategy.

Most commonly used codec: H.265

Dynamic Adjustment with SVC (Scalable Video Coding):

1) During a call, the system constantly monitors the situation and can adjust encoding parameters such as: resolution , frame rate , bitrate , audio quality

2)This flexibility ensures stable communication, even under varying network conditions or device limitations.

IV. External API Used for Video Transmission

Jitsi Api

There are three types of Jitsi API interfaces: Global, Local, and JaaS [5]

1) Global: uses the public server https://meet.jit.si
Main advantage: easy connection to the application

Main disadvantage: server-side configuration cannot be changed, scalability and customization limitations

2) Local: can be run using Docker containers (as I did), direct deployment on a server
Main advantage: control over configuration

Main disadvantages: complex setup, documentation is not always up to date, requires resources

3) JaaS: paid model (unlike the global and local versions) provided by 8x8
Main advantage: easy integration into applications

Main disadvantage: paid subscription

WebRTC protocols are used for video and audio transmission, with SRTP and DTLS encryption
Used video codecs [6]:

- VP8 – primary
- H.264 – if required by devices
- VP9 – in newer versions

Used audio codec:

- Opus – primary

V. Proposed Solution

A. Application Functionality

- Support for communication with multiple cameras
- Camera manipulation (cameras can be moved, removed, and hidden)
- Access to the conference (and other pages) page from various devices
- Selection of displayed participants
- User sections (where their cameras are shown) can be moved, displayed in fullscreen, and zoomed in
- Registration and authorization option → after authorization, the user can see their previous conferences
- Option to log in as a guest (only a name is required) → the account is deleted after leaving the conference
- Chats are available: personal (between two users) and conference chats
- Screen sharing
- Role-based access (User, Creator, Admin): The Creator and Admin have access to the admin page, where they can create/delete/modify users and change global settings (e.g., time for deleting unused conferences and temporary users).

B. Application Structure

The project is a single-module: the backend and frontend are placed in one repository.

C. Server-side of the Application:

1) AuthController: Controller for User Registration Management

2) ChatController: Controller for Managing Chats
WebSocket configuration is used for real-time chat functionality. A configuration class WebSocketConfig implements the interface WebSocketMessageBrokerConfigurer and is annotated with @Configuration and @EnableWebSocketMessageBroker (to include WebSocket and STOMP support in the project).

3) ControlController: Controller for the Admin Page

4) InitialController: Controller for Managing the Home Page, User Search, Settings Management, and Storing Initial Video Conference Devices (audio and video)

5) ConferenceController: Controller for Managing Video Conferences

D. *Functionality of the pages*

1) *Registration Page:*

The user sets the following details: first name, last name, email, password, city, country, and address. Server-side validation is performed for the first name, last name, password, and email (they cannot be empty or contain only spaces).

2) *Chat Page*

Messages are added dynamically using the showMessage method from chatLogic.js. The main chat configuration operates in JavaScript.

Chat between users functionality: send messages in real-time, delete their own messages in real-time , delete the chat in real-time , clear the chat in real-time in real-time

Conference chat functionality: Send messages in real-time delete messages in real-time

3) *Initial Page*

Functionality:

If the user is not logged in, they can: log in by opening a modal window with fields for the username and password to access their account, register by being redirected to the registration page, or start a conference by entering their name and proceeding to the initial device selection page. If the user is logged in, they can see: their chats and join them, a list of users with search functionality to start a chat, a list of conferences where they can leave a conference, delete a conference if their account is not in it, join a conference directly if their account is in it, join via the initial device settings page with the conference identifier predefined, or use search by conference ID. They can also view their account information (name, surname, email, city, country, address), and they can edit their details by clicking the "Edit" button (including changing their password), or delete their account by clicking the "Delete" button, with a confirmation window appearing.

4) *Device Settings Page*

Functionality:

It is possible to select a previously saved device configuration (which is unique). Users can also join the conference without selecting a microphone or video devices. Additionally, more than one camera can be selected.

5) *Control Page*

Functionality:

Users with the Admin or Creator role can enter the page, where a list of conferences is displayed. The user can leave a conference, delete a conference if their account is not in it, join the conference directly if their account is in it, or join via the initial device settings page, where the conference identifier will be predefined. A list of users with their information is also displayed. The page allows for removing or updating user information or adding a new user. Additionally, a list of global settings is shown, such as the time after which unused conferences will be deleted. The user can modify the values of settings (e.g., time interval) and add new settings.

6) *Conference page*

Functionality:

Communicate with other users, add/remove own cameras, hide cameras of other users. It is possible to enlarge and move cameras. You can enlarge, display in full screen, and move user sections across the screen. You can use the conference chat. It is possible to choose which users will be visible on the page and which to hide. A password can be set for the conference. Users can be invited to the conference.

E. **Docker containers of the application**

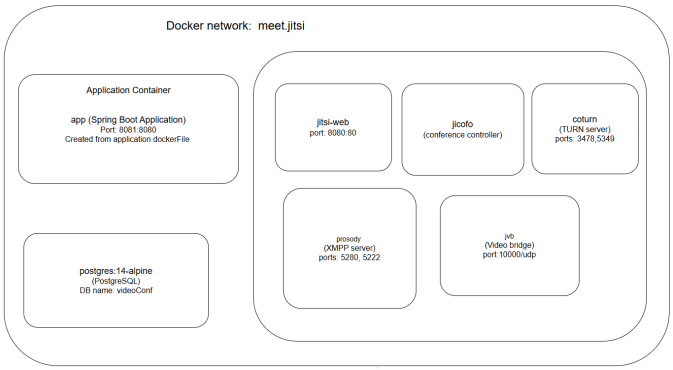

Figure 1. Docker containers of the application

F. *Conference page interface*

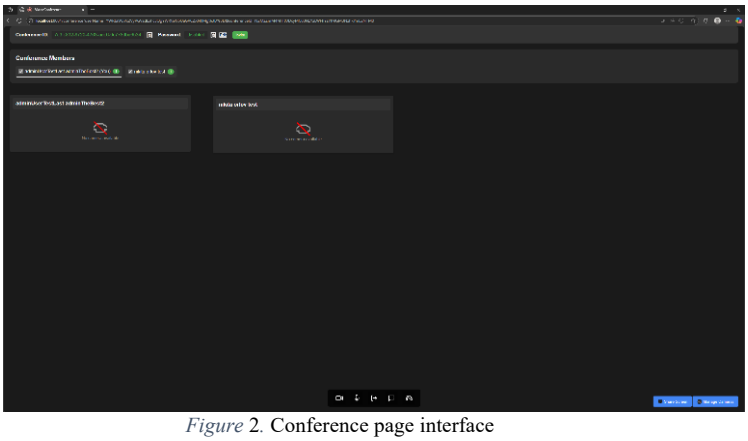

Figure 2. Conference page interface

VI. CONCLUSION

In this project, I analyzed the principles and technologies behind modern video conferencing systems and implemented my own application with extended functionality. The system

supports multiple cameras, flexible user interface interactions, chat functionality, and role-based access control. By using Jitsi as the core video conferencing engine and integrating WebRTC protocols, the application ensures real-time communication with security and scalability in mind.

Compared to existing platforms like Zoom and Google Meet, my solution introduces more advanced camera management and customization options, while also providing flexibility through both guest and registered user access. The deployment using Docker containers enhances portability and manageability across environments.

Overall, this work demonstrates not only an understanding of how modern video conferencing technology's function but also the ability to design and implement a full-featured, extensible conferencing system from the ground up.

ACKNOWLEDGEMENT

This paper was supported by DISIC (09I05-03-V2), NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2), and EULiST (ERASMUS).

REFERENCES

[1] Cisco, "Known issues and limitations for Webex video," Webex Help Center. [Online]. Available: https://help.webex.com/en-us/article/gw8u4j/Known-issues-and-limitations-for-Webex-video

[2] Nokia, "The future of video compression: Is VVC ready for prime time?" Nokia Blog, Oct. 4, 2022. [Online]. Available: https://www.nokia.com/blog/the-future-of-video-compression-is-vvc-ready-for-prime-time

[3] Coconut, "What is VP8 Codec?" Coconut Blog. [Online]. Available: https://www.coconut.co/articles/what-is-vp8-codec

[4] CometChat, "Zoom Video Technology Architecture," CometChat Blog, Jan. 18, 2022. [Online]. Available: https://www.cometchat.com/blog/zoom-video-technology-architecture

[5] 8x8, "JaaS FAQ," 8x8 Developer Documentation. [Online]. Available: https://developer.8x8.com/jaas/docs/faq

[6] Jitsi, "News," Jitsi Desktop, [Online]. Available: https://desktop.jitsi.org/Main/News.html