Design and Implementation of WebRTC Video Conference System Structure Compatible with GB/T28181 Devices*

Shuming Jiang*, Nengwu Liu, Guoli Yang

Qilu University of Technology, JiNan ShanDong 250000, China

jsm@sdas.org

Abstract. In order to be able to better understand the scenarios in the conference, this paper proposes an architecture of WebRTC (Web Real-Time Communication) video conferencing system compatible with GB/T28181(Technical Requirements for Information Transmission, Exchange and Control of Public Security Preventive Video Surveillance Networking System proposed by the Chinese government). The WebRTC videoconferencing system in this paper consists of four layers, which are service layer, STUN/TURN server, signaling server and streaming media server. In order to be able to access the GB/T28181 equipment, this paper proposes a method to transcode the PS stream of GB/T28181 into a WebRTC stream. Meanwhile, GCC (Google Congest Control) congestion control algorithm is added in the stream transmission process to get better viewing effect.

Keywords: GB/T28181; WebRTC; video conferencing system; transcode

1 Introduction

Video conference system [1][2] is a comprehensive application of various technologies such as network, communication and multimedia. With the increasingly strong demand for real-time information, the video conference system has received more and more attention due to its intuitive, real, cross-regional and low-cost communication characteristics, and is widely used in various fields of various organizational activities. Such as distance education, distance negotiation, telemedicine, telediagnosis, etc.

Especially with the continuous construction and development of new smart city infrastructure, command and dispatch applications in city management require command halls to go to command and dispatch personnel, field operators, business and technical experts to work collaboratively through multi-terminal video conferencing. It provides intuitive, realistic and cross-regional real-time information exchange for command and dispatch personnel, field operators, business and technical experts at any time and in any region. However, traditional video conferencing systems are not conducive to

* Fund Project: National Key R&D Program of China (2019YFB1404700)

H. Wu et al. (Eds.): CSIEDE 2022, ACSR 103, pp. 411–420, 2022.
https://doi.org/10.2991/978-94-6463-108-1_47

promotion due to their high cost and long development time. So the short development time and low cost of WebRTC (Web Real Time Communication) is often used as the first choice for video conferencing [3].

2 Related Technology Introduction

2.1 WebRTC

The purpose of WebRTC [2] technology is to provide a wide range of developers with a real-time communication platform that can run on the web. This platform will be mainly used for real-time transmission of audio and video signals, which is a major challenge for current Web applications. Previous mainstream approaches all required clients to load additional plugins. One of the principles of WebRTC is that all the interfaces it provides are open source, free, standardized, efficient, and integrated within the browser without plug-ins.

The WebRTC model includes three modules [4]: front-end Web application, browser and Web API. In essence, the multimedia modules and protocols required for real-time communication (network transmission and session management and signaling abstraction, etc.) are embedded in the Web browser, thus eliminating the differences between the underlying hardware and the operating system. The overall architecture is shown in Figure 1.

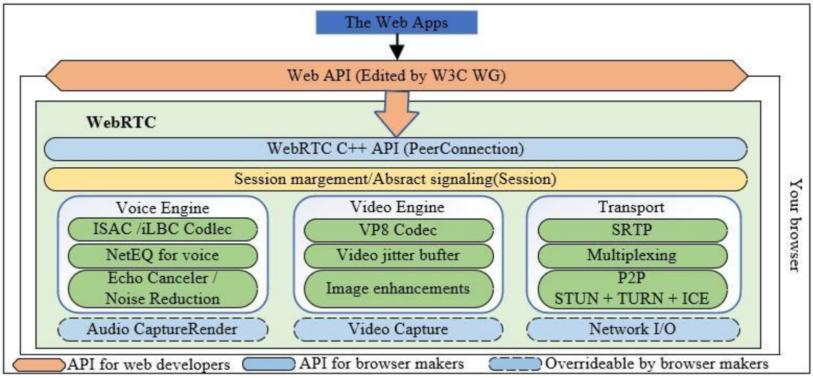

Fig. 1. WebRTC General Architecture

2.2 GB/T 28181

GB/T 28181 to SIP protocol as a benchmark, specifying the interconnection structure and communication protocol structure between video surveillance networking systems. Among them, the video surveillance networking system in real-time video and audio on-demand, historical video and audio playback and other media session services, the establishment of two transmission channels: session channel (SIP) and media flow channel (RTP/RTCP); in the heartbeat, historical video and audio playback control and other non-session services, the use of the relevant extension protocols to achieve SIP. In terms of NAT penetration, GB/T 28181 does not give relevant regulations [6][8].

2.3 PS Encapsulation of Video and Audio Data Based on RTP

RTP-based PS encapsulation first encapsulates the video and audio streams into PS packets, and then encapsulates the PS packets into RTP packets in a load-based manner. For PS encapsulation, each video frame should be encapsulated into a PS package, and the PS package for each key frame should contain the System Header and PSM (Program Stream Map), with the System Header and PSM placed after the PS package header and before the first PES package. A typical video keyframe PS packet structure is shown in Table 1, where PESV is the video PES packet, PESA is the audio PES packet, and the video non-keyframe PS packet structure generally does not contain the system header and PSM.

Table 1. PS Package Structure of Video Keyframes

PS Header	System Header	PSM	PESV	PESA

ES (Elementary Streams) are streams of data that come directly from the encoder, also known as net load data, ES is a collective term for encoded video streams (such as H.264), audio streams (such as AAC), and other encoded data streams. ES are data streams that contain only one type of content (e.g. pure video or audio). Each ES is composed of several access units (AUs). Each video AU or audio AU is composed of header and encoded data. Partially composed, 1 AU is equivalent to 1 encoded video image or 1 audio frame. It can also be said that each AU is actually a display unit of an encoded data stream, that is, a sample equivalent to a decoded video image or an audio frame.

Packetized Elementary Streams (PES), a data structure used to deliver ES. It is the data stream formed by the ES stream after PES packing, that is, the ES stream is grouped, packed and added with packet header information, which is the first packing of the ES stream.

A PS (Program Stream) packet consists of several PES packets with synchronization information and clock recovery information in the PS packet header. A PS packet can contain up to 16 video PES packets and 32 audio PES packets with the same clock reference.

2.4 GCC (Google Congest Control)

The Google Congestion Control (GCC) algorithm is used for congestion control in WebRTC. The GCC algorithm is mainly divided into two parts, one is packet loss-based congestion control and the other is delay-based congestion control. In recent WebRTC implementations, GCC has moved both of its congestion control algorithms to the sender side for implementation. The two algorithms themselves have not changed, except that some additional feedback information is needed on the sending side to calculate the delay, for which WebRTC extends the RTCP protocol. The main thing is to increase the Transport-CC Feedback, which carries the arrival time of each media packet received by the receiver [9][10].

Part of the code control structure in webrtc is shown in Figure 2. After receiving data from the socket layer, the rtcp packet is processed by transport to get the feedback, and the feedback is forwarded to the rtcp processing module on the corresponding send-stream through call, and finally the feedback is forwarded to the GoogCcNetworkController through RtpTransportControllerSend forwards the feedback to GoogCcNetworkController for code rate estimation, and then forwards the estimated code rate (target bitrate), probing strategy (probe config), congestion windows to pacer, pacer forwards to pacingContrller to use for sending bitrate control.

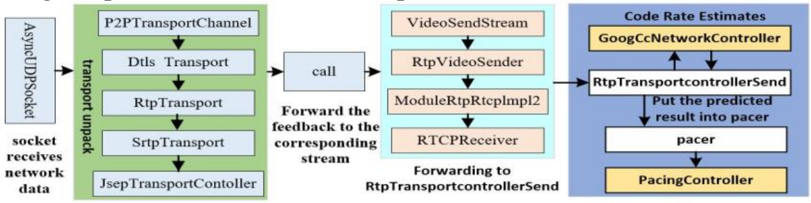

Fig. 2. Overall Code Control Structure in Webrtc

3 Implementation of WebRTC Video Conference System

3.1 Architecture of WebRTC Video Conferencing System

Two conditions are required to access GB/T 28181 devices in the conference system: GB/T 28181 transcoding into WebRTC streams; GB/T 28181 signaling and WebRTC signaling interoperability. The overall structure of the conference system based on GB/T28181 and WebRTC technology is shown in Figure 3. The system adopts a layered architecture system, which mainly includes service layer, signaling layer and media layer, and adopts the modular design idea of low coupling and high cohesion within each layer.

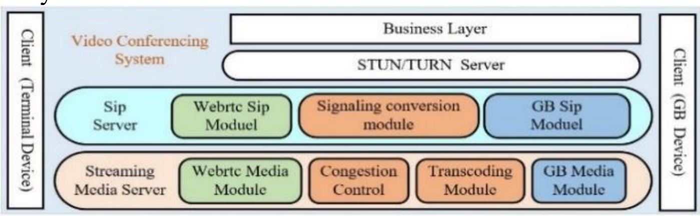

Fig. 3. Overall Structure of Video Conference System

Business Layer: mainly responsible for providing business interfaces for WebRTC clients and storing business data.

STUN/TURN Server: It is used to implement net traversal. This article uses the open source coturn server.

Signaling Layer (Signaling Server): it contains WebRTC signaling module, signaling protocol conversion module and GB/T 28181 signaling module. Among them, WebRTC signaling module to achieve the signaling interaction between the video conferencing system and the client; signaling protocol conversion module to achieve the conversion between WebRTC signaling and GB/T28181 signaling; GB/T28181 signaling module to achieve the session establishment with GB/T28181 terminal acquisition equipment.

Media Layer (Streaming Media Server): it contains WebRTC media module, media transcoding module, GB/T 28181 media module and congestion control module. Among them, WebRTC media module realizes media transmission between video surveillance equipment and WebRTC client; media transcoding module realizes conversion between WebRTC media stream and GB/T28181 media stream; GB/T28181 media module realizes media transmission with terminal acquisition equipment. At the same time, a congestion control module based on IGCC-A congestion control algorithm is implemented between the transcoding module and the WebRTC media module to reduce the packet loss rate of streaming media packets in the network and to reduce the network delay.

3.2 Streaming media server compatible with GB/T28181

In this paper, the streaming media server is ported from the open source SRS server to the WebRTC part, and further developed to implement GB/T28181 media stream to WebRTC code stream.

Implementation of GB/T28181 Media Stream to WebRTC Stream. In the field of video conferencing, it is now more about WebRTC, which can be used directly in the browser, without transcoding and with lower latency. In today's public security field, equipment cascade is mainly carried out through GB/T 28181, and the media stream in GB/T 28181 is RTP-loaded PS stream, and the loaded video format in PS stream is H.264. Since the WEB browser cannot directly support PS stream on-demand, users need to install plug-ins to play PS stream. Plug-ins need to be adapted to various browsers and various operating systems, plug-in compatibility is poor, and the user experience is not good. Moreover, browser companies need to develop the corresponding plug-ins, which requires additional development workload and wastes human and material resources. Therefore, we need a system to convert GB/T28181 media streams to WebRTC streams, and transfer the video in the field of public safety to the field of Internet video conferencing.

As shown in Figure 4, the real-time conversion steps of the realized GB/T28181 media stream to WebRTC stream are as follows [1]:

Step 1: The receive port receives RTP packets loaded with PS streams, first looks for the first RTP packet loaded with a PS packet header, and then puts the loaded RTP packet into the PS packet cache;

Step 2: Check if the Sequence of RTP packets is contiguous, if so, go to step 3. If not, clear the PS cache and also clear the main H.264 cache and frame cache while skipping to step 5.

Step 3: Parse the RTP packets in the PS packet cache, while continuing to accept RTP packets into the PS packet cache continuously until a complete PS packet is received. Parse the video data loaded in the complete PS packet into ES streams and store them in the primary and secondary H.264 video caches.

Step 4: Un-framing the ES stream in the main H.264 cache and storing each complete H.264 frame that is unframed into the frame cache.

Step 5: Un-framing the last ES stream cache of the sub-H.264 cache and storing each complete H.264 frame into the frame cache to ensure that the frame is always available on the playback side.

Step 6: packaging the whole H.264 frames into WebRTC streams by the packaging module.

Step 7: The packaged stream is pushed to the WebRTC streaming module via the URL push stream module based on the sending code rate feedback from the GCC algorithm in the congestion control module.

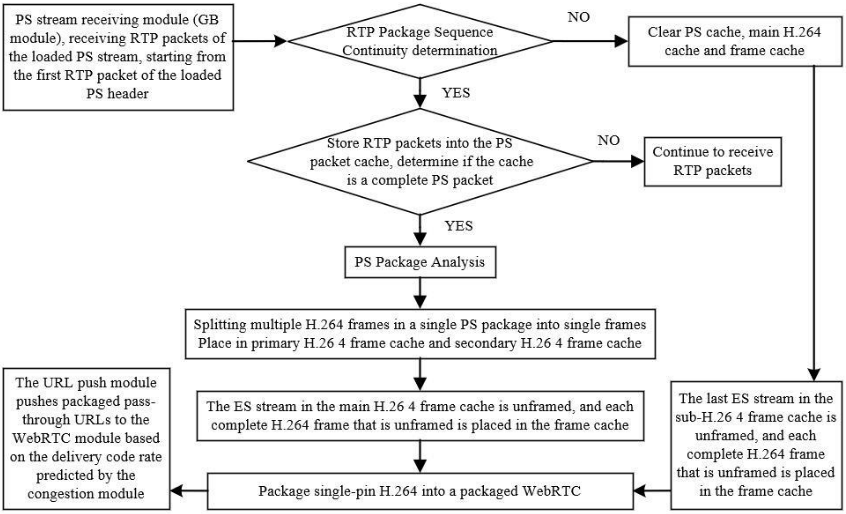

Fig. 4. Stream Conversion Process

As shown in Figure 5, the modular components of the conversion system built by the GB/T28181 media stream to WebRTC live stream conversion method include the RTP-loaded PS packet cache module, PS packet parsing module, media stream cache module, H.264 frame packing module and URL push stream module.

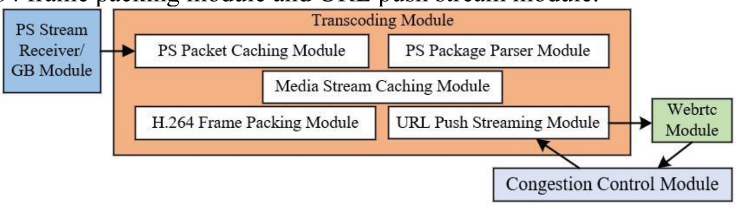

Fig. 5. Stream Transformation Structure

The PS stream receiving module is responsible for receiving RTP packets loaded with PS streams. The RTP loaded PS packet cache module is responsible for parsing the

RTP packets loaded with PS streams. The first RTP packet loaded with a PS packet header is found, and then the RTP packet loaded with a PS packet is stored in the PS packet cache. Check whether the Sequence of RTP packets is consecutive, if it is, then the ES streams in the main H.264 cache are unframed in the media stream cache module. If not, the PS cache is cleared, the main H.264 cache and frame cache are cleared, and the last ES stream in the secondary H.264 cache is unframed to ensure that there is always a picture on the playback side. The PS packet parsing module is responsible for parsing the RTP packets in the PS packet cache, while continuing to accept RTP packets into the PS packet cache until a complete PS packet is received, parsing the loaded ES streams in a complete PS packet into the primary and secondary H.264 caches. The media stream cache module is responsible for unframing the H.264 data (ES streams) in the primary and secondary H.264 caches, and putting each complete H.264 frame from the unframing into the frame cache. The H.264 frame packing module is responsible for packing complete H.264 frames into WebRTC. The congestion control module estimates the sending code rate with GCC algorithm by feeding the Transport Feedback message from the WebRTC streaming module, and gives the sending code rate to the URL pushing module. The URL push module pushes the packaged stream to the WebRTC streaming module via URL address according to the estimated sending code rate of the congestion module.

Implementation of Some Main Interface.

Local Media Stream Acquisition Interface(getUserMedia). The getUserMedia interface is used to get the media stream captured from the local camera and microphone. The format used for this interface is getUserMedia(constraints, successCallback, errorCallback). Where constraints indicates whether audio and video are supported, the format is a JSON string; successCallback is a function that will be executed if the acquisition is successful; errorCallback is a function that will be executed if the acquisition fails.

Establish a Real-time Communication Connection Interface (RTCPeerConnection). The RTCPeerConnction interface is used to establish a connection for transporting data streams between two nodes. RTCPeerConnection can be used as a parameter of the browser event target interface. The event (Event) is the HTML default class, and the target will return the element itself that triggered the event.

Real-time Communication Data Channel Interface (RTCDataChannel). RTCDataChannel is an interface developed for the reliable and efficient transfer of data in any format between two users, allowing instant messaging, file transfer and other functions.

Create GB/T28181 RTP Packet Receiving Port (openRtpServer). Create the RTP receive port of the GB/T28181 device, and if the port times out receiving data, it will be reclaimed automatically (without calling the closeRtpServer interface).

At the same time, it also implements interfaces such as closing the GB/T28181 RTP receiving port (closeRtpServer), obtaining the RTP push stream information interface (getRtpInfo), and RTP packet processing interface (ProcessInterface).

3.3 Client

Based on the current market prospect and functional requirements of video conferencing system, the client of the whole system realizes the following functions: registration, login, room creation, whether to turn on video equipment, whether to turn on voice equipment, free scaling of video screen ratio, switching video screen clarity, hanging up video conference connection, inviting friends to join video conference, private chat with other users, sharing files, removing conference personnel.

The effect of the video conferencing system is shown in Figure 6 and Figure 7.

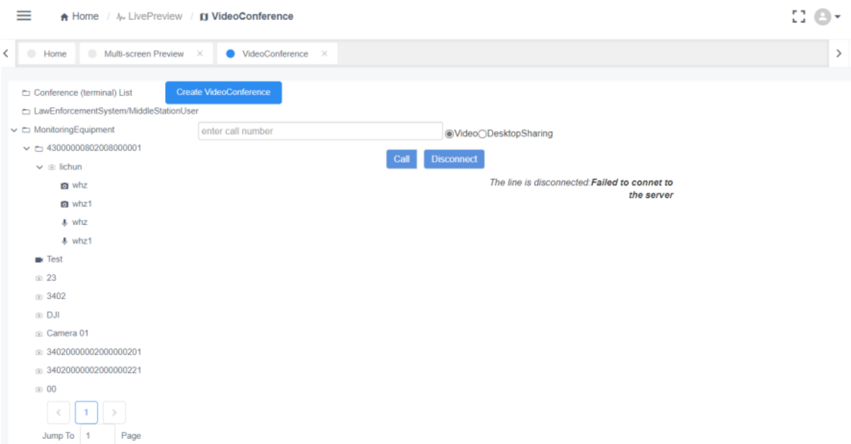

Fig. 6. Video conference system initial interface

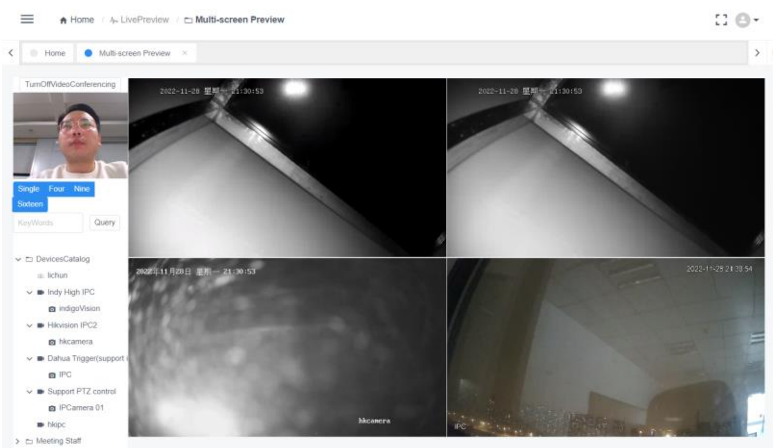

Fig. 7. Video conference system meeting opening interface

4 Conclusion

This paper designs a WebRTC video conferencing system with access to GB28181 devices, which is conducive to a real-time intuitive understanding of the scene in the meeting, and makes timely feedback in scenarios requiring on-site command. And a method of converting PS streams of GB/T28181 to WebRTC streams is proposed, which improves the conversion efficiency and client experience. At the same time, the GCC algorithm is optimized, and the IGCC-A algorithm is proposed to avoid bandwidth degradation due to burst delays and reduce frequent jitter in the sending rate caused by small fluctuations in network delay, which improves the user's viewing experience.

References

1. Bi Kaiqiang, Wansen, Liu Zilong, Chen Xiaoqi, Liu Qiong, Zhang Jingfeng. A real-time conversion method of national standard PS stream to RTMP live stream [P]. Anhui: CN110753234A,2020-02-04.
2. Wang Xin. Design and implementation of WebRTC-based video conferencing system[D]. Shenyang University of Technology,2019.
3. Wang Bingyang, Liu Yumei, Sun Qiongfang, Meng Qingqing. Design of mobile video surveillance system based on GB/T28181 and WebRTC technology[J]. Electronic Measurement Technology,2020,43(18):112-116.
4. Yuan Guanfu, Huang Zhichao, Wang Juhui. A streaming media transmission optimization method and system [P]. Fujian Province: CN112954414A, 2021-06-11.
5. Wang Yida. Research and implementation of real-time streaming media distribution network [D]. Beijing University of Posts and Telecommunications, 2021.
6. Li Yuxuan. Design and implementation of WebRTC-based instant messaging video system [D]. Beijing Jiaotong University, 2016.
7. Zeng Zhaocheng. Design and implementation of WebRTC-based video conferencing system [D]. Zhongnan University of Forestry Science and Technology, 2016.
8. G. Carlucci, L. De Cicco, C. Ilharco, S. Mascolo, "Congestion control for real-time communications: A comparison between NADA and GCC," 2016 24th Mediterranean Conference on Control and Automation (MED), Athens, Greece, 2016, pp. 575-580.
9. L. Wu, A. Zhou, X. Chen, L. Liu, H. Ma, "GCC-beta: Improving Interactive Live Video Streaming via an Adaptive Low-Latency Congestion Control," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.
10. Wang Bingyang, Liu Yumei, Sun Qiongfang, Meng Qingqing. Design of mobile video surveillance system based on GB/T28181 and WebRTC technology[J]. Electronic Measurement Technology,2020,43(18):112-116.