Research and Implementation of WebRTC via WebSocket-based for Real-time Voice Communications

[1]Ankit Chander, [2]Sanat Gupta, [3]Yash Giri, [4]Yashdeep Verma

Department of Information Technology, IIMT College of Engineering, Greater Noida, Uttar Pradesh , India Department

Abstract. WebRTC (Web real-time communication) achieves a peer-to-peer real- time multimedia communication on web. This paper researched and analyzed the core architecture and relatedtechnologies of WebRTC, including video input and output, multimedia transmission, the process ofpeer- to-peer connection establishment and signaling mechanism. This paper presented a signaling exchange mechanism via WebSocket-based and researched the WebRTC peer-to-peer connection based WebSocket in detail. A real-time multimedia communication system has been built and running well on the mobile internet. The research provides an academic and practical foundation forWebRTC signaling work.

Keywords: WebRTC; WebSocket; Signaling Mechanism; Real-time Multimedia Communication; Mobile Internet

Introduction

With the rapid development of mobile Internet and more and more intelligent mobile devices, the research of multimedia communication systems which focus on mobile user terminal (such as smartphones) has become a focus of people's concern. Web multimedia applications that based on WebRTC have got more and more attention because of higher development productivity and ease of use. Web Real-Time Communication (WebRTC) is a collection of standards, protocols, and JavaScript APIs, the combination of which enables peer-to-peer audio, video, and data sharing between browsers (peers). Instead of relying on third- party plug-ins or proprietary software, WebRTC turns real-time communication into a standard feature that any web application can leverage via a simple JavaScript API [1]. WebRTC greatly reduce hardware costs and technology costs of Web real-time multimedia interactive systems development. Currently, the latest version of the PC Chrome, Firefox browser and the Android Chrome have already support WebRTC. The official standard of WebRTC that include browser API, data transmission protocol etc. is still in process of formulating.

WebSocket enable bidirectional, message- oriented streaming of text and binary data between client and server. Any side can send data to another side any time

In this paper, we analyzed the core framework and related core technologies of WebRTC, and put forward a WebRTC signaling exchanging mechanism based on WebSocket as the signaling exchanging data carrier. It realized the initial negotiation of multimedia session between browsers and the establishment interactive connections. Finally, using this signaling exchanging mechanism implemented and WebRTC API, designed and implemented a peer-to-peer multimedia real-time interactive system in the mobile internet

WebRTC Core Architecture and Related Technical Analysis

The core function of WebRTC is to support the network multimedia communication. Overall, establishing multimedia connections between browsers needs audio technology, video technology and network transmission technology.Audio input and output devices: They are used to collect and play multimedia information.Network connections: In the online video communication, there is a large number of data between peers need to be transmitted. It requires a stable and reliable network connection as the guarantee of data transmission.Data encoding, decoding, transmission and display: After collected video by input devices, the data need to be encoded, transmitted. The other peer of connection need to accept data, decode and display.

The multimedia audio and video communication process is shown in Figure 1.

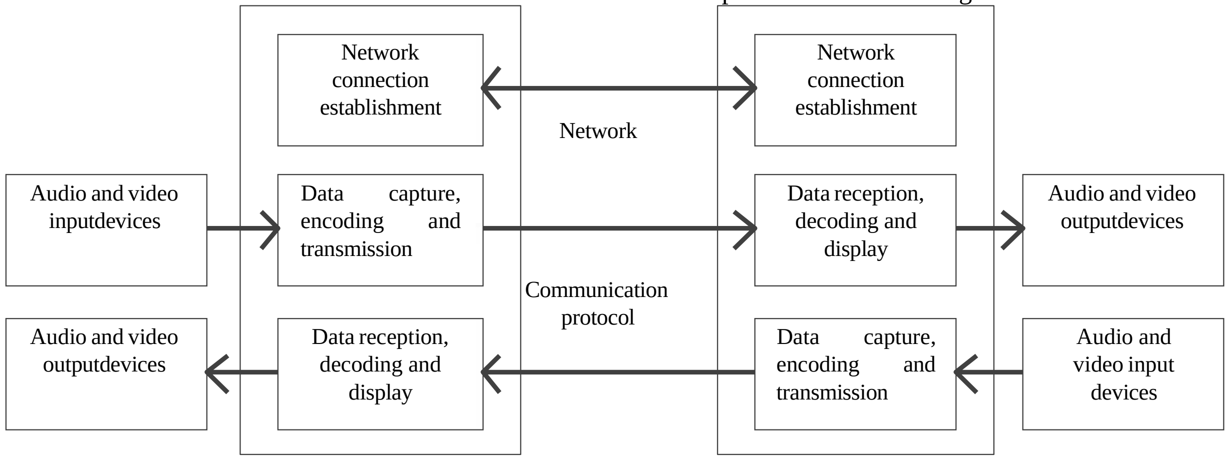

Network connection establishment
Network connection establishment
Network
Audio and video inputdevices
Data capture, encoding and transmission
Data reception, decoding and display
Audio and video outputdevices
Communication protocol
Audio and video outputdevices
Data reception, decoding and display
Data capture, encoding and transmission
Audio and video input devices

Fig.1. Multimedia Communication Process

WebRTC Architecture and Analysis. WebRTC core architecture is based on multimedia communication process, it's mainly include voice module, video module and transmission module . As shown in Figure 2.

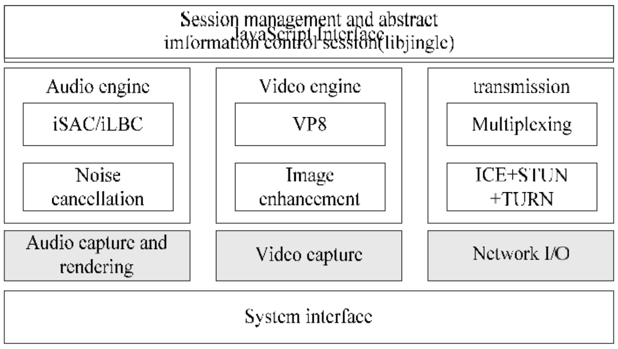


Fig. 2. WebRTC Architecture

The top of the architecture is the JavaScript API that can be called by developer directly; Session management relies on open-source project libjingle; Voice and video engine depend on appropriate architecture and technology. Next, this paper will focuses on analysis of the data transmission and signaling management of WebRTC.

Real Time Data Transmission Technology Based on WebRTC. The data transmission of thetraditional B/S systems is carry out between the browser and the server. The browser sends arequest to the server, and then the server respond corresponding data according to the request parameters. WebRTC achieves peer-to-peer real-time data transmission between browsers. In the delivery of real-time data, timeliness and low latency can be more important than reliability. Therefore, WebRTC uses UDP at the transport layer.

In the both peers of session or data transmission, one of the fundamental requirement is the ability to locate and identify each other on the network. In the trivial case, where both peers are located on the same internal network without any firewalls or NATs between them. To establish the connection, each peer can query its operating system for its IP address. But usually, there are many firewalls or NATs between peers, so ICE agent which colligates STUN, TURN, etc. is used to penetrate firewalls and NATs in WebRTC [4, 5].

Each WebRTC connections object contains an ICE agent. ICE agent is responsible for gatheringlocal IP, port tuples and queries an external STUN server to retrieve the public IP and port tuple of the peer. Furthermore, it is responsible for performing connectivity checks between peers and sending connection keepalives to STUN server. If configured, ICE agent appends the TURN server.If the peer-to-peer connection fails, the data will be relayed through the specified intermediary [6]. This complete process is shown in Figure 3.

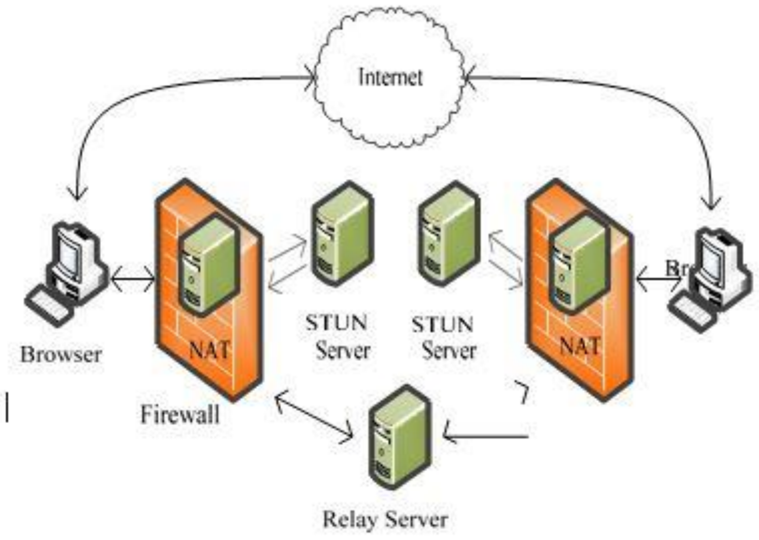


Fig.3 NAT and FireWall Pentration policy of ICE

WebRTC Signaling Mechanism. On the basis of reliable data channel, what is required is session negotiation before stablishment a connection between browsers. This part of work is done by the WebRTC signaling mechanism.Before session negotiation, it must determine whether the data can be successfully transmitted to the other peer and whether the other peer ready to establish a connection. Thus, the initial peer of session need to send an Offer signal, the other peer need to respond an Answer signal. WebRTC does not define the standard of transmission channel and protocol, this allow interoperability with a variety of other signaling protocols powering existing communications infrastructure, such as SIP, Jingle, ISUP and so on [6]. In this paper, the transmission channel is implemented using WebSocket.After transmission channel be implemented, next, it is supposed to exchange the session description information. WebRTC uses Session Description Protocol (SDP) to describe the parameters of the peer-to-peer connection. SDP describe the session profile, which represents a list of properties of the connection: types of media to be exchanged (audio, video, and application data), network transports, used codecs and their settings, bandwidth information, and other metadata [7]. The process of SDP exchange between peers is as follow.

- The initiator (User A) creates an offer, and set it as his local description of the session. Then, he sends the generated session offer to the other peer (User B)

- Once the offer is received by User B, he sets User A's description as the remote description of the session, generates the answer SDP description, and sets it as the local description of the session. Then he User B sends the generated session answer back to User A.
- Once User B's SDP answer is received by User A, User A sets User B's answer as the remote description of his original session.

integration WebSocket Service in HTTP server, opening the WebSocket port and listening to the request. In the browser, a WebSocket object need to be created according to the server address and port. The WebSocket connection will be created after handshake. Finally, the browser and the server are able to push signal data to each other.

RTCPeer Connection API Introduction. In WebRTC, the above mentioned data transmission, session mechanism and other functions are encapsulated in RTCPeerConnection API. It is responsible

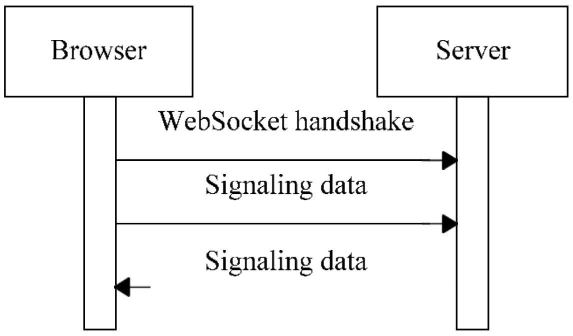


Fig.4. WebSocket Connection Establishment Process

for the management of full ICE workflow for NAT traversal, sending STUN automatic keepalives between peers, keeping track of local streams and remote streams. Developer can use itto generate offer, receive answer, manage ICE status and so on.

Real-time Multimedia Interactive System Design

Studies above shown that almost all of the peer-to-peer multimedia transmission technologies are encapsulated in WebRTC API. But the signaling mechanism and transmission channel that are not defined in WebRTC need to research and implement by developer. Therefore, this paper will presents a complete WebRTC signaling mechanism via WebSocket, and uses this mechanism and WebRTC API to achieve a peer-to-peer real- time multimedia interactive system.

WebSocket Service Design. As the signal transmission channel, WebSocket is the foundation of signaling. First, it is necessary to build a WebSocket service and then establish a WebSocket bidirectional connection between a browser and a server. This connection establishment process is shown in Figure 4. The specific method is

Identification among Connections Peers. In consideration of browsers need to identify each other by server after building the formal WebRTC peer-to-peer connection. Since a 'Chat Room Pattern' (multiple users visit a same website URL) was designed to achieve identification among connections peers. The process of identification is as follow.

- User A visit the appointed chat room address (e.g. http://webrtc/chat1), the page will establish a WebSocket connection with server after loaded the page. Browser sends a WebSocket message to server. The message name is 'join_in' and data is the chat room identification (chat1). Then, User A keeps a wait state.
- Once the 'join_in' message is received by server, it identifies whether there is a chat room with the same names (chat1). It will create a new chat room if not exist.
- User B visit the same chat room address with User A, and sends the same WebSocket message to server.
- Once the message from User B is received by server, the same name chat room and existed userwill be identified. Next, server broadcast all of the users socketID (every WebSocket client has his own unique socketID) to users.
- Once all users received the socketID that is broadcast by server, they save it in local environment.

Now, the identification work among users has been completed. If User A want to negotiate WebRTC connection with User B, he just need to send negotiate message and User B's socketID to server. The server will select the User B by socketID and forward the message from User A. The answer from User B to User A has the same procedure.

Complete SDP and ICE Information Exchange. After ensuring that multiple communication peers can be identified each other, the next step is exchanging signaling information. The signaling information mainly includes the SDP information and ICE information of each peer. These signaling information is serialized using JSON and transmitted by WebSocket.

For SDP information, the initiator of communication sends the WebSocket Offer signal to others peers by server's forwarding. The socketID and the SDP data of initiator are carried in signaling. The other peers will respond the Answer signal according to received information.

For ICE information, ICE agent of each peer automatically begins the process of discovering all the possible candidate after multimedia communication, whenever a new candidate is discovered, the information of this candidate will be sent to others peers by server's forwarding. The other peers configure its remote ICE candidate once received information.

The process of SDP and ICE information exchanging is shown in Figure 5, Figure 6.

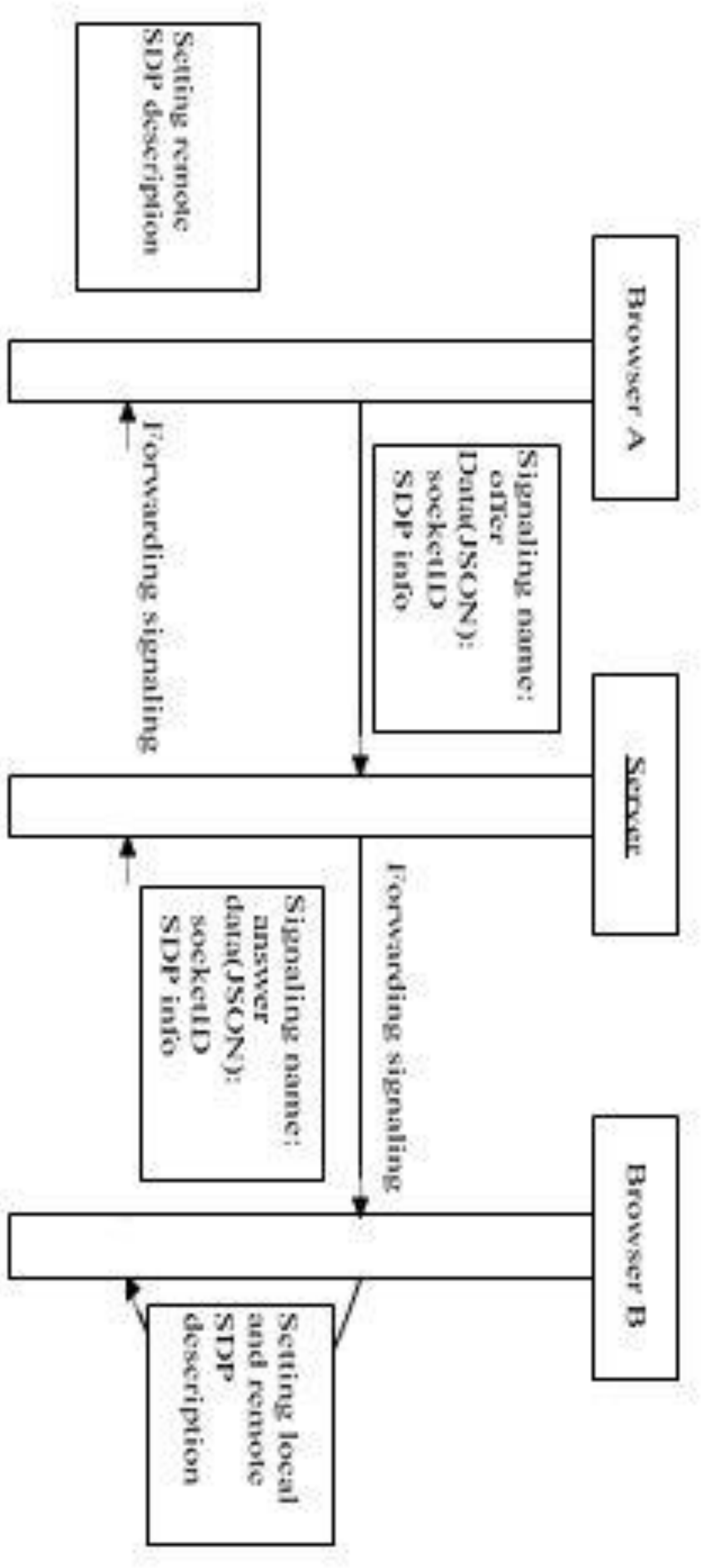


Fig.5. SDP Information Negotiation

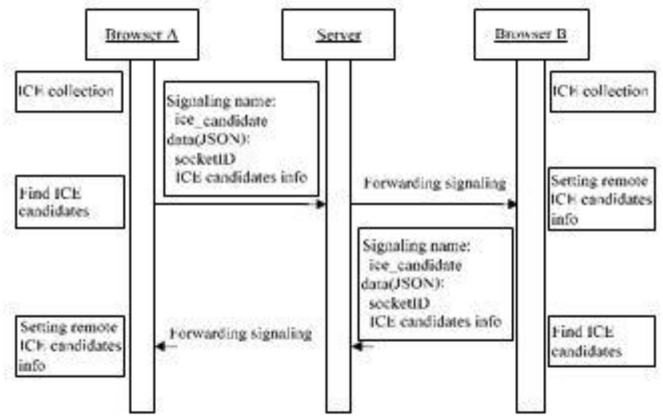

Browser A
Server
Browser B
ICE collection
Signaling name: ice_candidate data(JSON): socketID ICE candidates info
ICE collection
Find ICE candidates
Forwarding signaling
Setting remote ICE candidates info
Signaling name: ice_candidate data(JSON): socketID ICE candidates info
Setting remote ICE candidates info
Forwarding signaling
Find ICE candidates

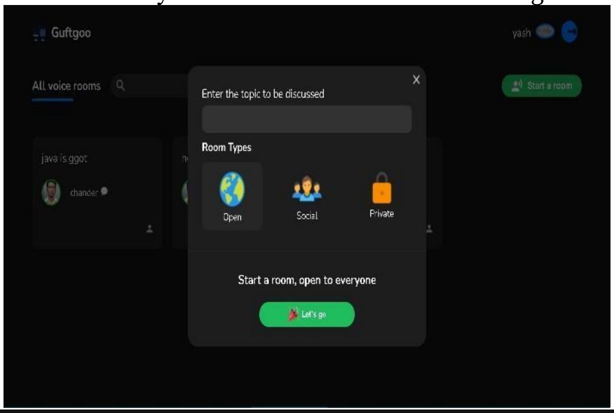
Fig.6. ICE Information Negotiation

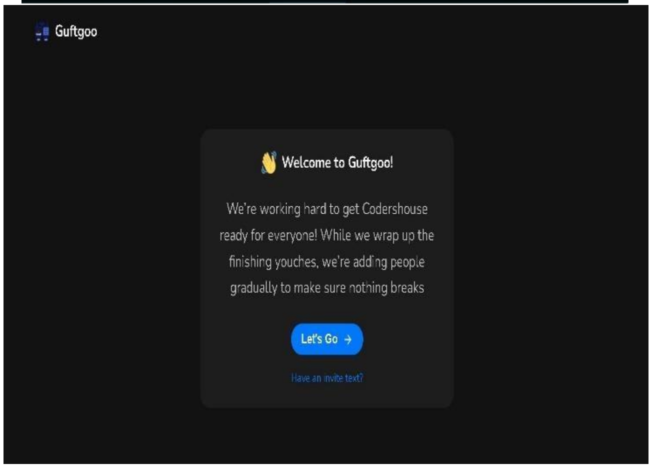
So far, the peer-to-peer connection between browsers has been built after SDP and ICE information negotiation. The preparation of the whole WebRTC communication has been completed.

Audio and Video Transmission and Output. After all of above preparations have been completed, browsers use WebRTC API to get local audio and video input, then transmit these data by the built peer-to-peer connection. The HTML tag <video> is used to output the video data.

System Implementation and Analysis

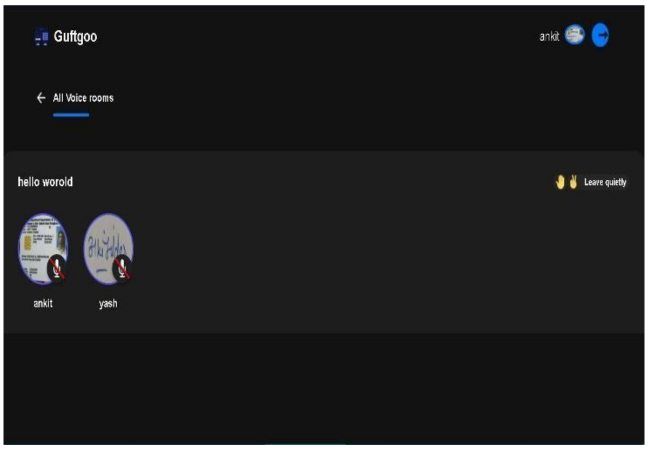
On the basis of the above research, we implemented a simple real-time multimedia communication system. This system supports peer-to-peer connection between browsers after signaling negotiation. Multimedia data can be real-time transmitted. Users can perform a real-time video communication. Furthermore, this system is mobile internet- based, and supports multimedia real-time interaction between the smart mobile devices. The system's screenshots is shown in Figure 7.





Welcome to Guftgoo!

We're working hard to get Codershouse ready for everyone! While we wrap up the finishing youches, we're adding people gradually to make sure nothing breaks



System Implementation Method. In the browser side, the system is built mainly based on the HTML5 and JavaScript and call the WebRTC and WebSocket interface. In the server, the HTTP server is built based on Node.js Express module, the WebSocket service is supposed by ws module. For STUN server, configured to use Google's public test server. As space is limited, the system building process cannot be explained detailedly here.
Analysis. This system, which is running on the normal HTTP server, has low-cost hardware costs. In the whole communication process, the server's function is negotiated the signaling before communication, the multimedia data transmit from one peer to another peer directly, server load is reduced greatly. WebSocket which suppose the bidirectional data communication used for signalingtransmission channel, it is also well suited to WebRTC usage scenario

Conclusion

First, this paper researched WebRTC architecture and analyzed the core technology. Next, for the part of signaling management that is not been defined in WebRTC, we proposed a solution that used WebSocket as signaling transmission channel and built a signaling server to forwarding signaling information. Finally, we built a real- time multimedia communication system based on the mechanism. The system is running well. The research provides an academic and practical foundation for WebRTC signaling work. What is most important is the system can running in mobile internet, the mobile smart devices can 2006).
communicate with each other. In the next step of the work, multiple browsers communication framework will be researched due to the low communication efficiency because of too many connection.

References

1. Jennings, Cullen, Ted Hardie, and Magnus Westerlund. "Real-time communications for the web." Communications Magazine, IEEE 51.4 (2013): 20-26.
2. Fette, Ian, and Alexey Melnikov. "The websocket protocol." (2011).
3. Bergkvist, Adam, D. Burnett, and Cullen Jennings. "A. Narayanan," WebRTC 1.0: Real-time Communication Between Browsers." World Wide Web Consortium WD WD-webrtc-20120821 (2012).
4. Rosenberg, Jonathan. Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols. No. RFC 5245. 2010.
5. Mahy, Rohan, Philip Matthews, and Jonathan Rosenberg. Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun). No. RFC 5766. 2010.
6. Grigorik, Ilya. High Performance Browser Networking: What every web developer shouldknow about networking and web performance. " O'Reilly Media, Inc.", 2013.
7. Handley, Mark, Colin Perkins, and Van Jacobson. "SDP: session description protocol." (