Interactive Connectivity Establishment Protocol for WebRTC System with NAT Traversal by Manual Signaling

Raswa[1][a], Ahmad Lubis Ghozali[2][b] and Kurnia Adi Cahyanto[3][c]
[1,2,3] *Politeknik Negeri Indramayu, Jawa Barat, Indonesia*
Email: raswa@polindra.ac.id

Keywords: ICE Protocol, WebRTC, NAT Traversal, Signaling

Abstract: Nowadays, real-time communication (RTC) to become the trend for smart technology systems in some fields. The connection between hosts by RTC has a peer-to-peer (p2p) model, it's not the client-server model, so p2p needs connectivity establishment on every peer. In this paper, we will examine how the effect of the Interactive Connectivity Establishment (ICE) Protocol to build Web Real-time Communication (WebRTC) systems by NAT Traversal, such as Session Traversal Utility for Network Address Translation (STUN). In this study, we use manual signaling to interact and build connections with two peers, using the Whatsapp application to exchange session descriptions. The results of implementation in our method are that using RTC needs STUN as a Traversal of NAT for building p2p or real-time communication. Our recommendation is that actually establishing real-time communication using STUN is not always successful, so we need to test with other protocols such as Traversal Using Relay for NAT (TURN).

[a] https://orcid.org/0000-0000-0000-0000
[b] https://orcid.org/0000-0000-0000-0000
[c] https://orcid.org/0000-0000-0000-0000

1 INTRODUCTION

Web Real-Time Communications (WebRTC) is an open-source project allowing real-time communication via a web browser. The WebRTC protocol handles peer-to-peer (p2p) communication in real-time, the development of communication can be video, audio, and data via a browser (Altanai, 2014). Communication through the browser between the two peers is built starting with the generation of audio, video, and data from each peer, which is then exchanged through a signaling process with the Session Description Protocol (SDP), initial connection creation via the ICE protocol, which is then carried out by securing data by DTLS, to the point where two browsers communicate in real-time (Chandrappa, D.R., Guruprasad, H.S., 2018).

The problem is that p2p connections are more difficult to establish than client-server connections (Nayyef, Z.T., Amer, S.F., Hussain, Z., 2018). The p2p connection difficulty arises when the browsers on the hosts that will communicate with each other are between different IP versions, network IP addresses, and network protocol types, especially when finding protocol type restrictions and firewall rules set by network administrators ()Sergiienko, A., 2014). So, how do we establish a real-time connection between two browsers when different IP versions, IP addresses, and network protocol types have become a problem? The solution uses the correct Interactive Connectivity Establishment (ICE) protocol behavior.

The ICE protocol will determine the best transport address called a candidate for establishing a p2p connection (Ristic, D., 2015)). When the Host is behind a router with a strict NAT (Network Address Translation) configuration, in addition to the ICE protocol, a STUN (Session Traversal Utilities for NAT) protocol is also needed (Emmanuel, E. A., Dirting, B. D ., 2017)). ICE is a framework used by WebRTC systems to connect two peers, regardless of network topology, allowing two peers to discover and establish connections with each other even if they use NAT to share global IP addresses with other devices on their respective local networks (Feher, B., Sidi, L., Shabtai, A., Puzis, R. and Marozas, L., 2018)). In this study, we observe how the behavior of connections built through the ICE protocol when faced with creating a connection with the ice server configuration value on the RTCPeerConnection object

1. between two browsers in one Host is null,
2. between two browsers in one Host sourced from the STUN server,
3. between two browsers in different hosts is null, and
4. between two browsers in different hosts sourced from the STUN server.

Knowing about the behavior of the ICE protocol under various network conditions is important (Mahmood, S.A., Ercelebe, E., 2018)). The advantage is that developers of the real-time communication system, WebRTC, can ensure that certain configurations can connect WebRTC agents between two browsers.

2 RESEARCH METHODOLOGY

2.1 Procedure and Design

The method in this study has the following stages (1) using the mediaDevices.getUserMedia API WebRTC to generate a Media Stream with a track containing the requested media type, (2) adding a media stream to the RTCPeerConnection object to establish a connection between two WebRTC agents, and (3) using events and methods on the RTCPeerConnection object are exchanged using a manual signalling process. The design of the signalling environment is shown in figure 1.

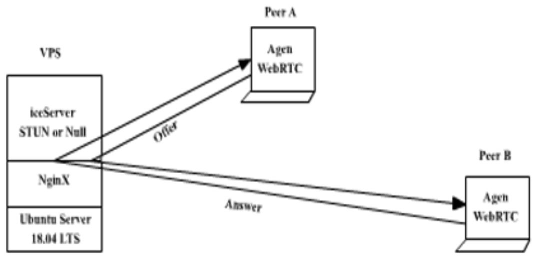


Figure 1: Signaling environment design.

.The actions createOffer, createAnswer, setLocalDescription, setRemoteDescription, addCandidate, and others, generate ICE candidates exchanged via SDP (Session Description Protocol). SDP contains information about any MediaStreamTrack that has been attached to a WebRTC session, codecs, and options supported by the browser, and candidates that have been collected by the ICE agent. The test to establish a connection between two agents in the WebRTC system is carried out in one host as shown in Figure 2.

Figure 2: Two browsers on a host in the WebRTC system.

Application of the ICE protocol to establish p2p connections, or connections without a central server. The trial whose design is shown in Figure 3 was carried out on the application of the ICE protocol supported by transverse NAT and STUN.

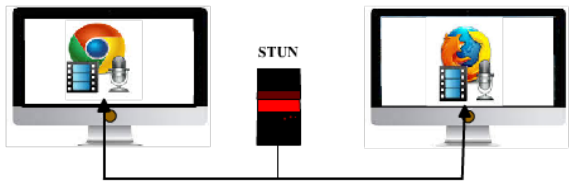


Figure 3: Two browsers between hosts in the WebRTC system.

2.2 Data Analysis

The behavior of the ICE protocol is studied by how the protocol interacts to establish connections between peers. The WebRTC system is initiated to establish a connection with the help of a signaling service to exchange Offer and Answer information containing SDP. In the endpoint configuration of the WebRTC connection system, we learn about session descriptions stored using SDP, including information about the type of media being sent, the format of the IP address and port of the endpoint, and the protocol used. Each peer is stored in two sessions: the local description describes itself, and the remote description describes the other end of the peer. SDP is analyzed concerning codecs, source addresses, and audio and video timing information. A codec (coder-decoder) is a program, algorithm, or device that encodes or decodes a data stream..

In the session description interaction through the bidding event, the receiver responds with an answer that learns the behavior of the ICE protocol for the two peers to establish a connection even though NAT separates the two devices. The RTCIceCandidate constructor is configured to represent one candidate ICE.

The WebRTC connectivity system, from sender to receiver and peer to peer, is analyzed with the getStats function of RTCPeerConnection. The primary statistical object used in the WebRTC statistical monitoring model is RTCStats..

Table 1: Parameter candidateInfo objek RTCIceCandidate.

Parameter	Nilai string
foundation	Are the two candidates the same
component	ICE transport (RTP=1 or RTCP=2)
protocol	udp or tcp
Priority	higher/lower or null
IP	Device source IP address candidate or null
port	Candidate device port number can be dialed or null
type	Host/srflx/prflx/relay

3 RESULTS AND DISCUSSION

The RTCPeerConnection interface represents the connection between two peers. Once a connection is established and opened, media streams (MediaStreams) and data channels (RTCDataChannels) can be added to the connection.RTCPeerConnection. Interfaces used to set up, open, and manage WebRTC connections: represent the peer media connections, data channels, and interfaces used when exchanging information about each peer's ability to select the best configuration for a two-way media connection.

We found a two-session return property during connection negotiations where the offer was rejected due to an incompatible format proposal, and each endpoint must have the ability to propose a new but incorrect format to the destination peer; the WebRTC agent used a delayed description, i.e., a description that was reset to null.

ICE candidates deal with exchanging information about network connections, either directly or via the STUN/TURN server. Candidates are faster UDP, and media streams can recover from disruptions relatively quickly, or candidates can also be TCP. The STUN server shows the address where the ICE agent sent the candidate.

The RTCIceCandidate object configures the candidate ICE used to create the RTCPeerConnection object. The ICE candidate reveals the routing required for WebRTC to communicate with remote devices. The candidate is represented as a string, the candidate IP address, which represents the candidate transport address used for connectivity checks. The string also indicates whether an RTP candidate or an RTCP candidate, the value is one or is derived from an a-line string (SDP attribute). RTCPeerConnection object configuration code:

```
componentDidMount=()=>{
   const configuration = null
   this.pc = new RTCPeerConnection(
   configuration)
   }
```

The ice server configuration value on the RTCPeerConnection object between two browsers on one host is null, represented by the RTCPeerConnection parameter:

{"candidate":"candidate:77142221 1 udp 2122260223 192.168.137.1 52920 typ host generation 0 ufrag zdOv network-id 1","sdpMid":"0","sdpMLineIndex":0}

Table 2: Parameter observation results RTCIceCandidate.

Parameter	Nilai string
foundation	77142221
component	RTP
protocol	UDP
Priority	2122260223
IP	192.168.137.1
port	52920
type	host

The ice server configuration value on the RTCPeerConnection object between two browsers in one host is sourced from the STUN server. RTCPeerConnection object configuration code:

```
componentDidMount=()=>{
   const configuration=
   "iceServers":[
      {
       urls:'stun:stun.l.google.com:
             19302'
   ]
   this.pc = new RTCPeerConnection(
   configuration)
   }
```

Parameters of RTCiceCandidate:

{"candidate":"candidate:0 1 UDP 2122187007 192.168.43.10 64384 typ host","sdpMid":"0", "sdpMLineIndex":0,"usernameFragment":"b0ef32ad "}

Table 3: RTCIceCandidate parameter observation results.

Parameter	Nilai string
foundation	0
component	RTP
protocol	UDP
Priority	2122187007
IP	192.168.43.10
port	64384
type	host

The ice server configuration value on the RTCPeerConnection object between two browsers in different hosts is null:

{"candidate":"candidate:77142221 1 udp 2122260223 192.168.137.1 61384 typ host generation 0 ufrag jNaW network-id 1","sdpMid":"0","sdpMLineIndex":0}

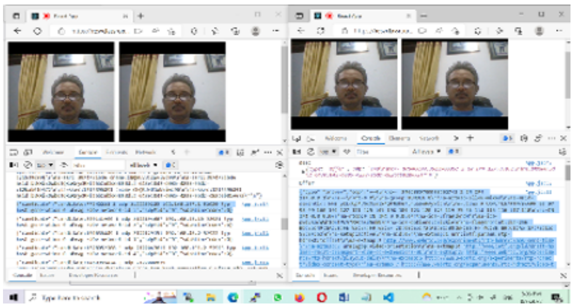


Figure 4: WebRTC agent connection between same hosts.

Table 4: The results of the observations of the RTCIceCandidate parameters on same hosts.

Parameter	Nilai string
foundation	77142221
component	RTP
protocol	udp
Priority	2122260223
IP	192.168.137.1
port	61384
type	host

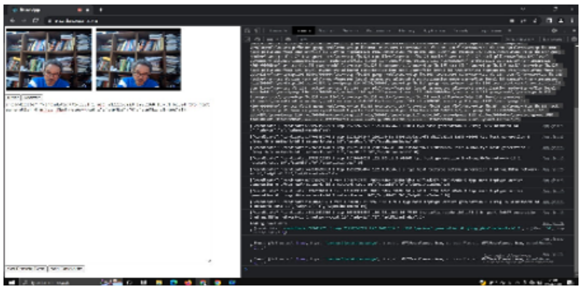
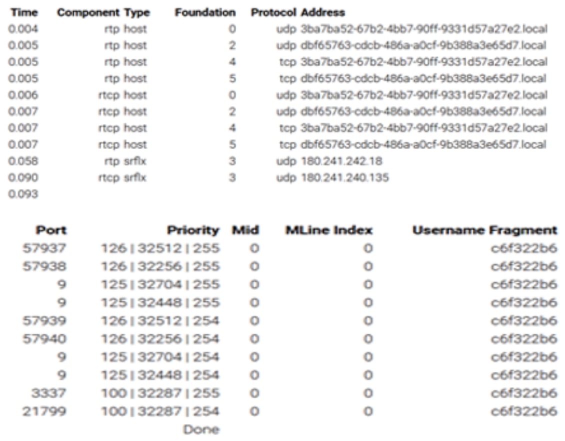


Figure 5: SDP and candidate list.

Ice server configuration value on RTCPeerConnection object: between two browsers on different hosts sourced from STUN server. Uji ice server urls: 'stun:stun.l.google.com:19302' menggunakan Trickle ICE diperoleh:

Table 5: STUN test results with Trickle ICE.



Time	Component Type	Foundation	Protocol	Address	Port	Priority	Mid	MLine Index	Username Fragment
0.004	rtp host	0	udp	3ba7ba52-67b2-4bb7-90ff-9331d57a27e2.local	57937	126 | 32512 | 255	0	0	c6f322b6
0.005	rtp host	2	udp	dbf65763-cdcb-486a-a0cf-9b388a3e65d7.local	57938	126 | 32256 | 255	0	0	c6f322b6
0.005	rtp host	4	tcp	3ba7ba52-67b2-4bb7-90ff-9331d57a27e2.local	9	125 | 32704 | 255	0	0	c6f322b6
0.005	rtp host	5	tcp	dbf65763-cdcb-486a-a0cf-9b388a3e65d7.local	9	125 | 32448 | 255	0	0	c6f322b6
0.006	rtcp host	0	udp	3ba7ba52-67b2-4bb7-90ff-9331d57a27e2.local	57939	126 | 32512 | 254	0	0	c6f322b6
0.007	rtcp host	2	udp	dbf65763-cdcb-486a-a0cf-9b388a3e65d7.local	57940	126 | 32256 | 254	0	0	c6f322b6
0.007	rtcp host	4	tcp	3ba7ba52-67b2-4bb7-90ff-9331d57a27e2.local	9	125 | 32704 | 254	0	0	c6f322b6
0.007	rtcp host	5	tcp	dbf65763-cdcb-486a-a0cf-9b388a3e65d7.local	9	125 | 32448 | 254	0	0	c6f322b6
0.058	rtp srflx	3	udp	180.241.242.18	3337	100 | 32287 | 255	0	0	c6f322b6
0.090	rtcp srflx	3	udp	180.241.240.135	21799	100 | 32287 | 254	0	0	c6f322b6
0.093						Done			

That STUN server test works if the TricleIce test can collect candidates of type "srflx". That TURN server test works if it can collect candidates of type "relay".

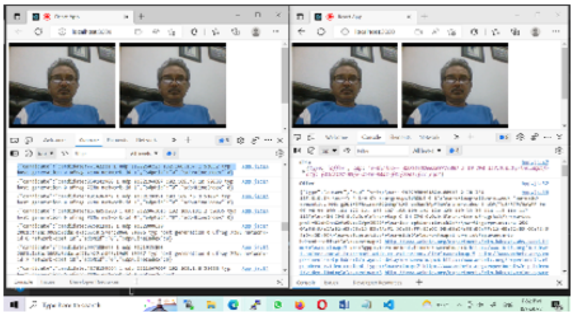


Figure 6: WebRTC agent connection between different hosts.

The ice server configuration value on the RTCPeerConnection object between two browsers in different hosts:

{"candidate":"candidate:2999745851 1 udp 2122260223 192.168.56.1 61386 typ host generation 0 ufrag /Wid network-id 4","sdpMid":"0","sdpMLineIndex":0}

Table 6: The results of the observations of the RTCIceCandidate parameters on different hosts.

Parameter	Nilai string
foundation	2999745851
compnent	RTP
protocol	udp
Priority	2122260223
IP	192.168.56.1
port	61386
type	host

The RTCIceTransport object provides access to and information about ICE transport, where data is sent and received, making it useful for accessing status information about connections, whether RTP or RTSP. The process of establishing a two peer connection in the WebRTC system is revealed through the process of checking, verifying, and selecting valid candidate pairs.

4 CONCLUSIONS

The ICE candidate agent determines how data travels from peer to peer. ICE allows candidates to represent connections over UDP or TCP, with UDP generally yielding the best candidate. Several types of UDP candidates can be indicated in the session description: candidate host (the IP address of the remote peer), prflx (the peer reflexive candidate that is the IP address that comes from symmetric NAT between two peers), srflx (the reflexive candidate server generated by STUN). The ICE protocol shows one of the two peers to function as the controlling agent, the decision maker as the controlling agent to decide which candidate pairs to use for connection, the other peer as the controlled agent. It is also indicated that the ICE agent gives a signal about the election to the controlled ICE agent using STUN and offers renewal. The controlling ICE agent in one ICE session selects more than one candidate pair. During unsuccessful connection, negotiation needs to renegotiate connection via ICE rollback. Failures are caused by the adaptation of hardware or network configurations and outdated WebRTC APIs. When the ICE negotiation session runs out of candidates to propose a specific RTCIceTransport, it has completed the collection for candidate generation. The remote partner, after receiving the candidate, will add the candidate to his candidate pool.

REFERENCES

Altanai.(2014). WebRTC Integrator's Guide: Successfully build your very own scalable WebRTC infrastructure quickly and efficiently. Birmingham: Packt Publishing Ltd.

Chandrappa, D.R.,Guruprasad, H.S. (2018). Real-Time Communication using WebRTC. SSRG International Journal of Mobile Computing & Application (SSRG-IJMCA) – Volume 5 Issue 2 May to August 2018.

Emmanuel, E. A., Dirting, B. D .(2017).A Peer-To-Peer Architecture For Real-Time Communication Using Webrtc. Journal of Multidisciplinary Engineering Science Studies (JMESS). ISSN: 2458-925X. Vol. 3 Issue 4, April – 2017.

Edan, N.M., Al-Sherbaz, A., Turner, S. (2017). WebNSM: A Novel Scalable WebRTC Signalling Mechanism for Many-to-Many Video Conferencing. IEEE 3rd International Conference on Collaboration and Internet Computing. DOI 10.1109/CIC.2017.00015.

Feher, B., Sidi, L., Shabtai, A., Puzis, R. and Marozas, L. (2018). WebRTC security measures and weaknesses. Int. J. Internet Technology and Secured Transactions. Vol. 8, No. 1, pp.78–102.

Loreto, S., dan Romano, S.P.(2014). Real-Time Communication with WebRTC. US: O'Reilly Media, Inc

Nayyef, Z.T., Amer, S.F., Hussain, Z.(2018). Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology. International Journal of Engineering & Technology, 7 (2.9) (2018) 125-130. Website: www.sciencepubco.com/index.php/IJET.

Mahmood, S.A., Ercelebe, E. (2018). Development of Video Conference Platform Based on WebRTC. International Journal of Electrical, Electronics and Data Communication, ISSN(p): 2320-2084, ISSN(e): 2321-2950, Volume-6, Issue-6, Jun.-2018. http://iraj.in

Pasha, M., Shahzad, F., Ahmad, A. (2016). Analysis of challenges faced by WebRTC videoconferencing and a remedial architecture. International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 10, October 2016.

Parmar, N., Virender Ranga, V.(2019).Performance Analysis of WebRTC and SIP for Video Conferencing. International Journal of Innovative Technology and Exploring Engineering (IJITEE). ISSN: 2278-3075, Volume-8, Issue-9S, July 2019.

Palloff, R. M., Pratt, K. (2013). Lessons from the Virtual Classroom: The Realities of Online Teaching. San Francisco: Jossey-Bass. Second Edition

Ristic, D. (2015). Learning WebRTC: Develop interactive real-time communication applications with WebRTC. Birmingham: Packt Publishing Ltd.

Sergiienko, A. (2014). WebRTC Blueprints: Develop your very own media applications and services using WebRTC. Birmingham: Packt Publishing Ltd.