Decentralized Communication Application for Local Network

Sanjeev Kumar Sharma[1], **M.L. Sharma**[2]

[1]*Student, Dept. of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India*
[2]*Professor, Dept. of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India*

---***---

Abstract – *Now a days there are bundles of mobile chatting applications for communication purpose. Many provides services over an internet connection. This paper emphasizes on making an application that is decentralized and do not require any internet connection for making audio and video calls i.e. it is a local network solution for connecting them with each other and able to communicate to each other by transmitting audio as well as video data.*

Key Words: webRTC, Decentralized, peer-to-peer, STUN, TURN, wifi, Ipv6

1. INTRODUCTION

Web real-time communication(webRTC) is a communication technology launched by google in 2011 and is open-sourced[1]. It enables peer-to-peer communication between the browsers in real time allowing exchange of audio and video data without any third-party plugin. Main objective of this project is to implement the webRTC for android applications and making an application which is decentralized by eliminating STUN server which is generally required in building WebRTC applications running over internet.

This project will empower local networks or community networks and make communication convenient at a lower cost.

1.1 Environment

Signaling server: we need some third party service that will help the peers share the service information and start direct streams transmission. That's a job for a **signaling server**.

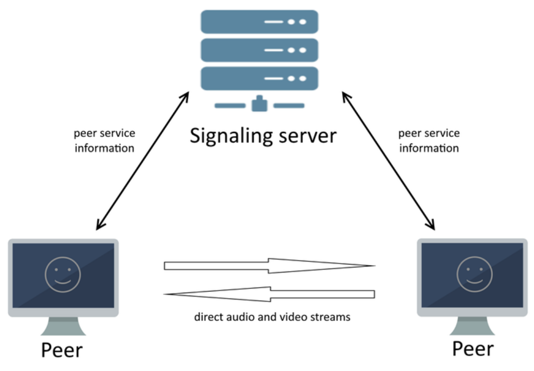


Fig 1: Signaling meachanism

STUN server: When peers are on other networks then they are connected to each other via internet and are hidden behind NAT routers. If they are on the same network then NAT will be eliminated and thus the STUN server.

TURN server: It is used to let peers transmit streams to each other behind a firewall. There is no peer-to-peer streams transmission in this case. All media traffic goes through a TURN server.

2. Proposed System

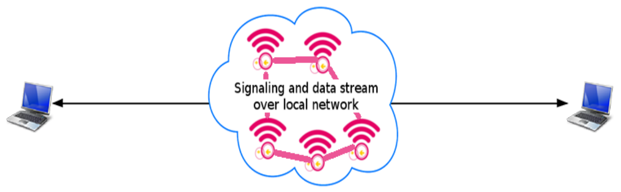


Fig 2: signalling between peers

2.1 Application flow:

1) User A issues a call request to user B, knowing destination address.

2) User B may decline or receive a call which can be typically done by JSON-based protocol.

3) If a successful in its process then it creates a session description also known as SDP which is also transmitted to B.

4) User B receives the request of SDP and replies with the same protocol.

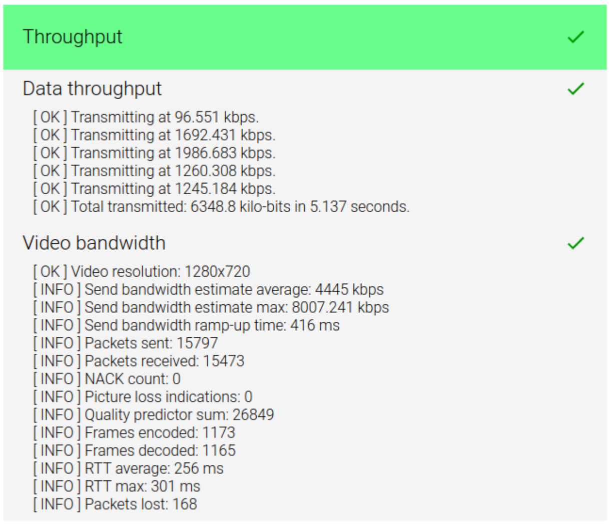
5) Now after successful handshake RTCPeerconnection is enabled and all data is transmitted through separate channel.

6) Both the user exchange audio and video through RTCPeerconnection.

2.2 why webRTC:

WebRTC fits best choice since it is a well-build, tested and mostly documented standard for video- and audio communication, which is exactly needed for the project. The best advantage of webRTC implementation is due to its low latency, it handles reading the camera and microphone input, encodes that data and tries to find the way with the lowest possible latency to send the data. Since, in most environments, due to NATs a direct connection is not possible, WebRTC handles NAT traversal, or even completely redirects the traffic over a TURN server. Here both the servers are turned off , since the application is designed to work P2P in local networks.

Normally, to find the best possible route between two nodes a STUN server is used, since that needs an external server. Since WebRTC is used in many different mobile and web applications we have room for scalability, giving us the possibility to adapt to other projects and applications. The signalling, e.g. the exchange of relevant networking information has to be done by the application. In the case of this application, an 'offer' is created by the WebRTC API of the initiating application. This offer is then encoded in a JSON object and transmitted to the receiver using IPv6 as explained above, which then creates an 'answer'. That answer is then transmitted back to the initiator, where it is then fed back to the WebRTC API. Upon this point, WebRTC takes over, creates a connection and starts transmitting audio and video.


```
Throughput                                                   ✓

Data throughput                                              ✓
  [ OK ] Transmitting at 96.551 kbps.
  [ OK ] Transmitting at 1692.431 kbps.
  [ OK ] Transmitting at 1986.683 kbps.
  [ OK ] Transmitting at 1260.308 kbps.
  [ OK ] Transmitting at 1245.184 kbps.
  [ OK ] Total transmitted: 6348.8 kilo-bits in 5.137 seconds.

Video bandwidth                                              ✓
  [ OK ] Video resolution: 1280x720
  [ INFO ] Send bandwidth estimate average: 4445 kbps
  [ INFO ] Send bandwidth estimate max: 8007.241 kbps
  [ INFO ] Send bandwidth ramp-up time: 416 ms
  [ INFO ] Packets sent: 15797
  [ INFO ] Packets received: 15473
  [ INFO ] NACK count: 0
  [ INFO ] Picture loss indications: 0
  [ INFO ] Quality predictor sum: 26849
  [ INFO ] Frames encoded: 1173
  [ INFO ] Frames decoded: 1165
  [ INFO ] RTT average: 256 ms
  [ INFO ] RTT max: 301 ms
  [ INFO ] Packets lost: 168
```

Fig -3: Latency test of webRTC

4. SYSTEM REQUIREMENTS

4.1 Hardware Requirements

mobile phones or tablets
a hotspot

4.2 Software Requirements

Android v4.0 or above

4.3 Development Requirements

Android Studio version 1.5
JDK version 1.8
IDE

5. FINAL PRODUCT

The developed application shows how the peers are connected to each other and communicate to each other when they are connected to common local network.

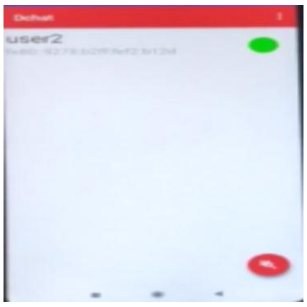


Fig -4 Online status of peer

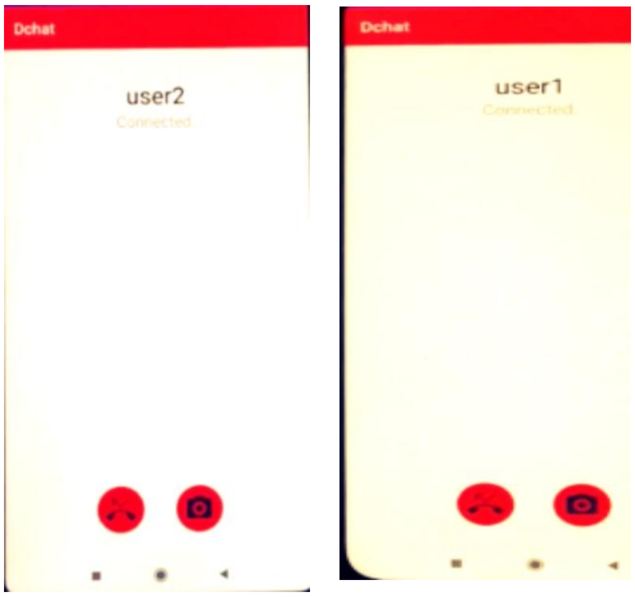


Fig -5 peers connected

6. CONCLUSION

peer-to-peer decentralized android application has been developed using webRTC. The users need to be connected to a common network or community network to call other user. The users can be contact if their status is online. The transmission of audio is fair enough for a proper communication but still there need an improvement in the quality. Each user is uniquely idendified by a link local address created and shared amongst other users.

REFERENCES

[1] "A WebRTC Video Chat Implementation Within the Yioop Search Engine"(2019). By Ho, Yangcha, Master's projects. 726. DOI:https://doi.org/10.31979/etd.dtz2-hstt

[2] "Communication App works without cellular Network" by Rachel Metz, in MIT Technology Review.

[3] "Instant Messaging over LAN using Android Application", by Avinash, Chirag Patel A R, Chiranth H N, Karthik Prasad K, Shabana Sultana in vol 4 issue:4, Apr 2017, IRJET.

[4] "A secure Chat Application Based on Peer-to-Peer Architecture", Mohamad Afendee Mohamed, Abdullah Muhammed, and Mustafa Man on 28-05-2015 in Journal of Computer Science.