WebRTC :

WebRTC provides Real Time Communications (RTC) capabilities to browsers and mobile applications and to be more precise about WebRTC it a set of JavaScript APIs in the browser to enable peer-to-peer, real time media and data exchange. This technology has a vast range of implementation from Video Conferencing to Mesh Networking. This technology is designed to be used without having a server so now the data need not to go through the servers rather it can be exchanged directly among the users as it establishes a peer-to-peer connection to other person's browser. The WebRTC API contains media capture, encoding and decoding audio and video, transportation layer, and session management. This approach is relatively safer than other methods. Let's say you want to send a file so that file is directly delivered to the person without the need for uploading it anywhere. So it minimizes the risk of the file falling in the wrong hands. WebRTC technology is facilitating a move towards browser-based VOIP telephony. These applications allow calls to be made and received from within a internet search browser, replacing the requirement to download and install a app for VOIP calling.

The JavaScript APIs in WebRTC are :-

MediaStream :

This API helps to capture the Audio and Video. This allows a web browser to access the microphone and the camera. Each MediaStream object can contain a number of different MediaStreamTrack objects that each represents different input media, such as video or audio from different input sources.Each MediaStreamTrack can then contain multiple channels (for example, the left and right audio channels). These channels are the smallest units that are defined by the MediaStream API.

RTCPeerConnection :

This API helps in streaming Audio and Video between the user's browsers.The RTCPeerConnection API is the heart of the peer-to-peer connection between each of the WebRTC enabled browsers or peers.The RTCPeerConnection interface represents a WebRTC connection between the local computer and a remote peer. It provides methods to connect to a remote peer, maintain and monitor the connection, and close the connection once it's no longer needed.

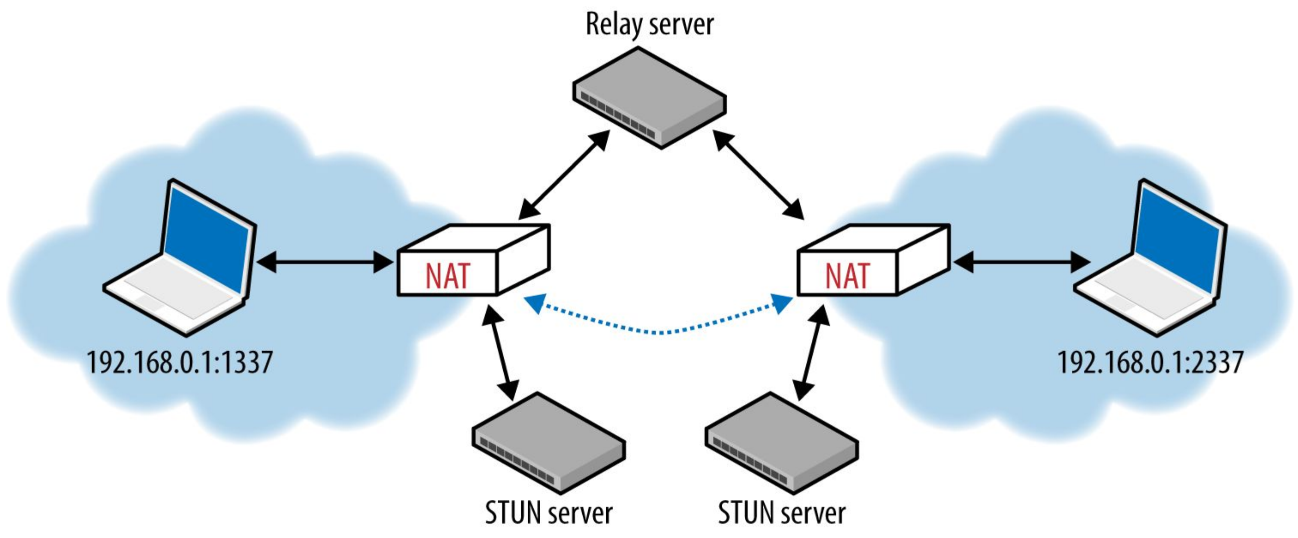
`RTCDataChannel :`

This API helps in streaming data between users. As well as sending media streams between peers using WebRTC, it is also possible to use the DataChannel API to send arbitrary streams of data. Although many people commonly refer to this as the RTCDataChannel API, it is more accurately defined as the WebRTC DataChannel API.The RTCDataChannel interface represents a network channel which can be used for bidirectional peer-to-peer transfers of arbitrary data. Every data channel is associated with an RTCPeerConnection and each peer connection can have up to a theoretical maximum of 65,534 data channels (the actual limit may vary from browser to browser)

`getStats :`

This API helps in retrieving the set of statistics about WebRTC sessions.The RTCStatsReport interface is used to provide statistics data about WebRTC connections. Calling getStats() on an RTCPeerConnection lets you specify whether you wish to obtain statistics for outbound, inbound, or all streams on the connection. The RTCRtpReceiver and RTCRtpSender versions of getStats() specifically only return the incoming and outgoing streams, respectively.

This is diagram explaining the working of WebRTC and the implementation of ICE Technology in WebRTC.



The WebRTC is optimized for Low-Latency. It was designed to use the UDP/IP protocols for transport. This protocol, mixed together with WebRTC’s video and audio codec means WebRTC is designed for low-latency and maximum throughput.

ICE tries to find the best path to connect peers. It tries all possibilities in parallel and chooses the most efficient option that works. ICE first tries to make a connection using the host address obtained from a device's operating system and network card; if that fails (which it will for devices behind NATs) ICE obtains an external address using a STUN server, and if that fails, traffic is routed via a TURN relay server.

NATs provide a device with an IP address for use within a private local network, but this address can't be used externally. Without a public address, there's no way for WebRTC peers to communicate. To get around this problem WebRTC uses STUN. RTCPeerConnection tries to set up direct communication between peers over UDP. If that fails, RTCPeerConnection resorts to TCP. If that fails, TURN servers can be used as a fallback, relaying data between endpoints. TURN is used to relay audio/video/data streaming between peers but it

STUN servers live on the public internet and have one simple task: check the IP:port address of an incoming request (from an application running behind a NAT) and send that address back as a response. In other words, the application uses a STUN server to discover its IP:port from a public perspective. This process enables a WebRTC peer to get a publicly accessible address for itself, and then pass that onto another peer via a signaling mechanism, in order to set up a direct link. (In practice, different NATs work in different ways, and there may be multiple NAT layers, but the principle is still the same.)

In particular assuming the IP address of the STUN server is known, the WebRTC application first sends a binding request to the STUN server. The STUN server replies with a response that contains the public IP address and port of the client as seen from the public network.

Now the application discovers its public IP and port tuple which can send to the other peer through SDP. With this mechanism in place, whenever two peers want to talk to each other over UDP, they can then use the established public IP and port tuples to exchange data.

Unfortunately, in some cases UDP may be blocked by a firewall. To address this issue, whenever STUN fails, we can use the Traversal Using Relays around NAT (TURN) protocol as a fallback, which can run over UDP and switch to TCP if all else fails.

All peer to peer data and stream transport is encrypted using DTLS-SRTP. The Datagram Transport Layer Security (DTLS) is a security protocol designed to secure stream based web communication. To make full use of WebRTC's features we must use HTTPS with TLS.

Currently WebRTC is supported on Microsoft Edge, Google Chrome, Mozilla Firefox, Safari and also on Opera Browser.

Applications of WebRTC :

1. File sharing:

Say you're working on a project and want to send massive files to your colleagues. You can directly send it through web browser using WebRTC instead of emailing and uploading it to third-party cloud storage. This will save your quota on the cloud and has lower risk of getting your files into the wrong hands.

2. Multiparty video conferencing:

You can create a multi-user video conference using WebRTC and establish a direct peer to peer connection with one another. Every platform provides free trial in starting, up to limited people. But scalability depends on the hardware and bandwidth, and you can also scale using multipoint control units and selective forwarding units. This feature is not just limited to the Web Browser but it is also implemented in Android and iOS. There are many websites and apps who are already using this technology in their apps to provide better performance.

3. Screen sharing:

You can share your share with your colleagues during meetings and discussions. It helps you to get better and clear ideas for the work in progress and its outcomes.

4. Broadcasting:

You may not broadcast with WebRTC, but it enables one-way media transmissions like concerts, speeches, podcasts, and live videos. Few WebRTC platforms also allow you with real-time access to participants attendance.

5. Embedded endpoints:

You can connect with live agents when you're on-the-go using WebRTC. You can embed vending machines, ATMs, bus stops, and retail store kiosks with WebRTC engines.

6. Sales enablement:

You can help your sales representatives for using this technology. You can provide ongoing assistance during the purchasing process by integrating your website or application with a WebRTC channel.

7. Emergency response:

WebRTC is playing major role in public safety. You can also use location based services and safeguard interactions enabling text communications, audio, and video communications. WebRTC data channel also allows you to have a deeper insight to previously existing communications, when responding to emergency calls.

8. Patient management:

You can also reduce patient visits and queue by using WebRTC based solutions for ongoing treatments. WebRTC platform helps you to give more time to higher priority patients. You may even provide treatment remotely by suggesting methods to an emergency case.

9. Storage conservation :

By using WebRTC you won’t need to download those bulky plugins and softwares to attend the conferences and live events. You don’t need to worry about the System Requirements anymore. This cross-platform availability is a huge advantage and develop the project once, use it anywhere.

10. e-Learning :

The teachers can teach their students remotely and ease up the teaching process. Due to availability of features like sharing presentations and conferencing. The learning and teaching process can be made more interactive.

Uses of WebRTC on the Liquid Galaxy:

Liquid Galaxy has a vast range of Implementation if we talk about the WebRTC technology. WebRTC has various utilities and Liquid Galaxy can enhance them for a better user experience. As shown in the Demo, Carlos De Dios showed the Liquid Galaxy using the WebRTC to initiate a video conference. Different displays of LG show different windows. So on the utility of Liquid Galaxy using WebRTC, after spending sometime on researching about the uses and where it can fit perfectly I came up with some uses that are as follows :

1. Cinema:

Liquid Galaxy can be the next big thing. LG when implemented with 3D screens can produce an amazing effect on streaming movies and this 3D effect can be extended to games as well. Imagine you are playing multiplayer Fifa 18 (just an example) and you just feel like you are on the ground with players. This will be a huge dent in the history of gaming and getting the 3D view without losing anything on the immersive quality display and the amazingly low latency of WebRTC.

2. Medical Treatments:

Liquid Galaxy can be implemented to carry out various remote treatments where the person can just start chatting and the doctor can have the full view of the bruises and wounds of the patient and that's not it LG can also be used to remotely train doctors by showing them the procedure. LG can be setup in remote areas where no qualified doctors are available. This will act as a guide for them and improve the medical conditions around the world.

3. Remote Support

The Support Team members will find this tool very useful as they have refer to various sources to provide assistance and they can even assist more clients at the same time. Let’s say that a support team member has to solve an issue which only his superiors can solve he can start a call from one of the screens without disconnecting the customer and complete the task with more ease.