WEBRTC(Web Real-Time Communication)

- for sharing video conferencing, file transfering and desktop sharing
- GetUserMedia() - access of webcam
- RTCPeerConnection() - establishes peer connection between two clients securely
- RTCDataChannel() - share data over channel
- GetStats() - helps to get webrtc sessions

ADVANTAGES

- No need of servers Inbetween clients, we can directly send data through wires Inbetween.
- Real-Time Audio and Video Streaming:
- Secure Communication:
- Easy Implementation:

WEBSOCKET VS WEBRTC:

- Webrtc is designed for high performance, high quality, it is browser to browser communication and faster than websocket.
- Also webrtc uses udp connection whereas websocket uses tcp protocol.

WEBRTC Uses websockets for signaling mechanism.

SIGNALING:

- Signalling server transfers and shares 3 meta data information between peers they are:
  - Session control information
  - IP address and port related information
  - codec and media type information

  So, these meta data is exchanged via Session Description Protocol (SDP) protocol

- After signalling is done we start sharing media between peers using RTCPeerConnection()

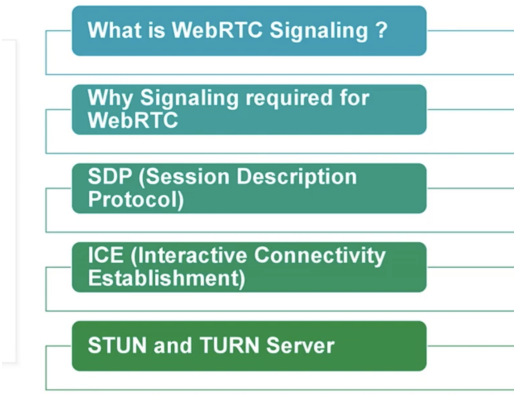
CONNECTING PEERS DIRECTLY IF YOU KNOW IP ADDRESS:

- usually peers directly cannot establish their connection with IP address with each other, because they are usually not allowed to.
- THIS IS MADE POSSIBLE BY ICE(INTERACTIVITY CONNECTION ESTABLISHMENT).
- But sometimes this connection will be Interrepted by NATS(network address translators) and firewalls.
- So we use STUN(SESSION TRAVERSAL UTILITIES FOR NAT) AND TURNS(TRAVERSAL USING RELAYS AROUND NATS) p
- So ICE USES STUN AND TURN PROTOCOLS to establish a connection.

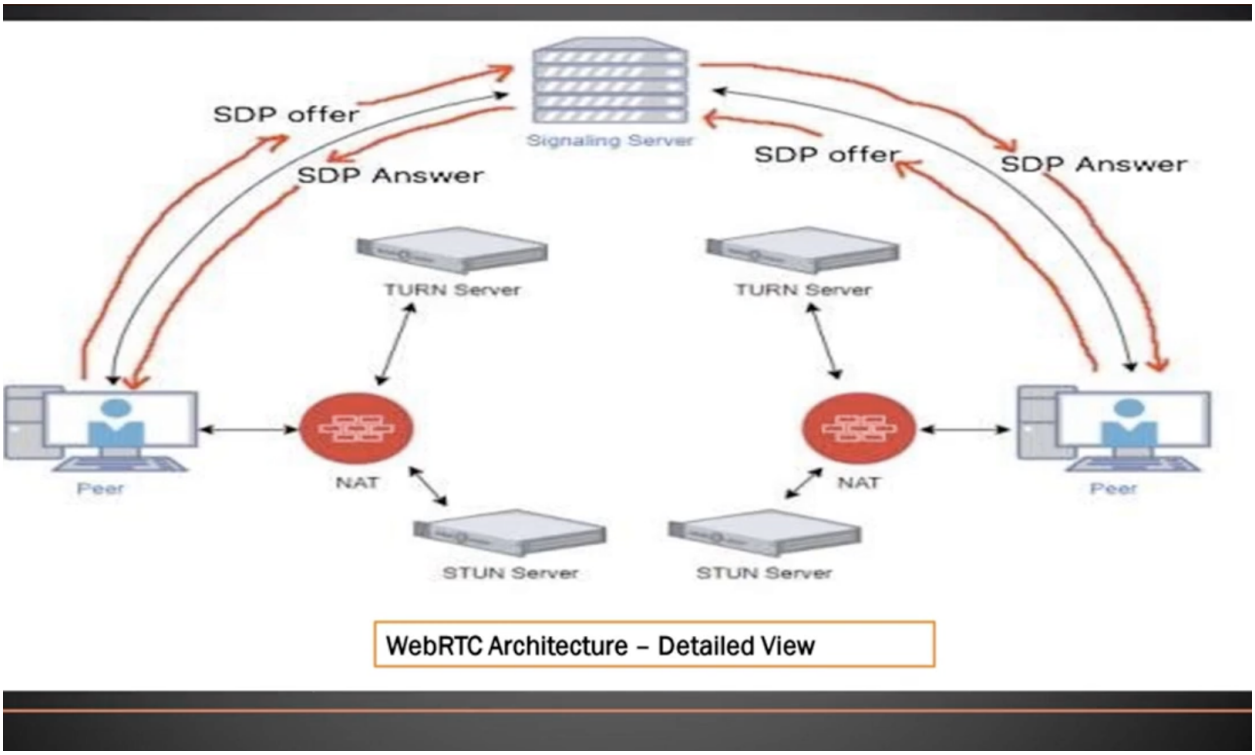
WHAT IS STUN AND TURN:

- IF THERE IS A USER IS UNDER NAT WHICH HAS LOCAL IP ADDRESS then PUBLIC IP ADDRESS CANT BE REACHED TO
- BUT IF THE NAT IS SYMMETRIC THEN WE USE TURN SERVER PROTOCOL.
- NAT IS A LOCAL IP ADDRESS
- WE CAN DISCOVER PUBLIC IP ADDRESS USING STUN OR TURN SERVERS.

```
//ICE
var configuration = {
    "iceServers": [
        {
          "url": "stun:stun.1.google.com:19302"
        },
        {
          url: 'turn:192.158.29.39:3478?transport=tcp',
          credential: 'JZEOEt2V3Qb0y27GRntt2u2PAYA=',
          username: '28224511:1379330808'
        }
  ]
};
//RTCPeerConnection
m_my_Connection = new RTCPeerConnection(configuration, {
    optional: []
```



- You don't strictly required a signaling server with WebRTC, as long as you have SDP offer and SDP answer
- So, you don’t need a signaling server if you have SDP offer and SDP answer and dont want to connect any extra/different/new p



CONNECTION FLOW CHART:

SIGNALING SERVER.

1. USER A SENDS SDP OFFER TO SIGNALING SERVER AND THIS INTERN SENDS TO USER B.
2. USER B RESPONDS WITH SDP ANSWER TO SIGNALING SERVER AND THIS SENDS TO USER A.
3. NOW USER A HAS CLIENTS DETAILS AND SESSION DETAILS.

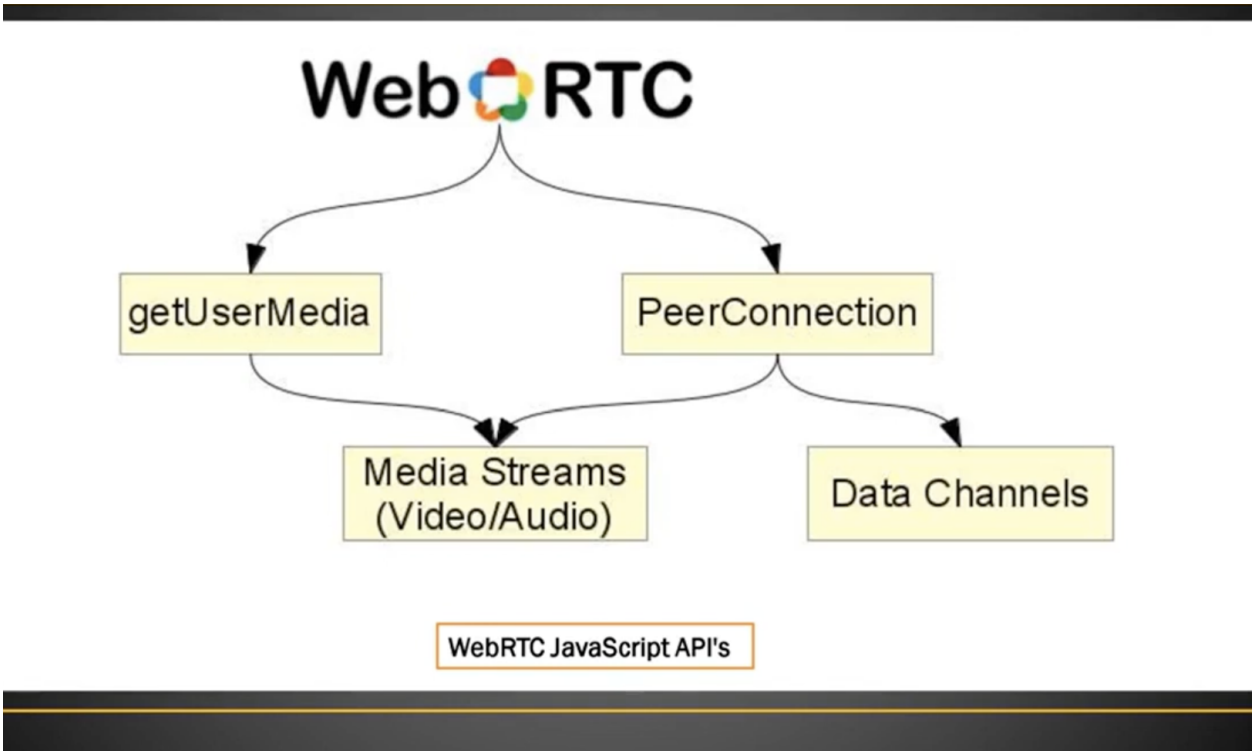
NOW SIGNALIGN SERVER PART IS DONE.

PEER TO PEER CONNECTION:

- AS USERS MAY USE NAT PROTOCOL AND IT WILL HAVE PRIVATE IP ADDRESSES UNDISCOVERABLE, THEN IT USES I(
- TURN IS USED IN SYMMETRIC NAT SERVERS.

THEN CHANNEL IS ESTABLISHED AND STREAMING HAPPENS.

WEBRTC APIS:



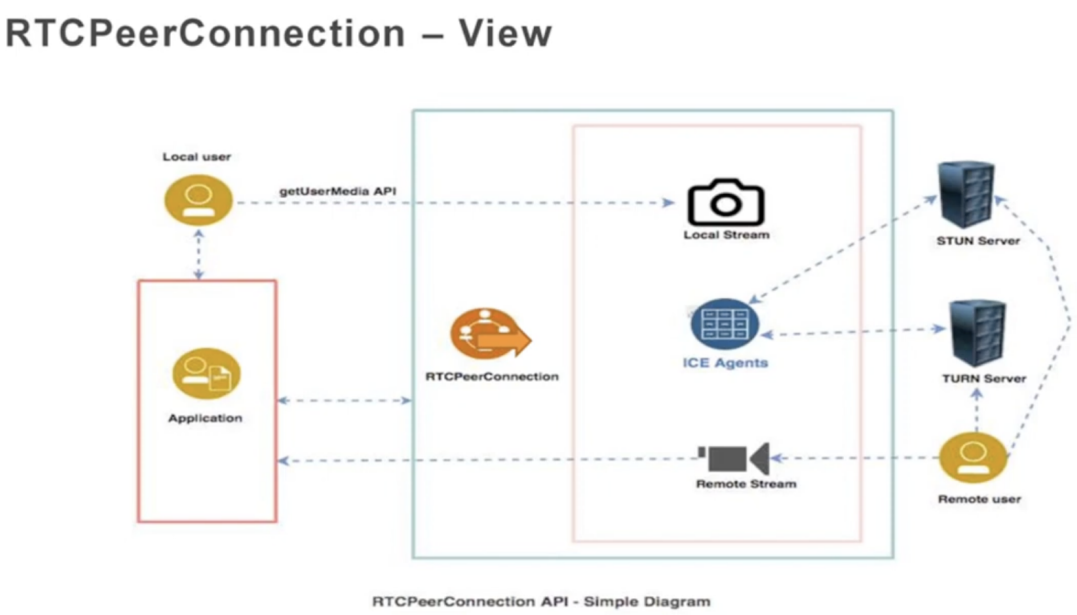
GETUSERMEDIA:

- IT JUST GETS THE ACCESS FROM CAMERAS AND MEDIA DEVICES AND ENABLES THEM AND GIVES US ACCESS TO S

PEERCONNECTION:

```
const configuration = {'iceServers': [{'urls': 'stun:stun.l.google.com:19302'
const peerConnection = new RTCPeerConnection(configuration);
```

-

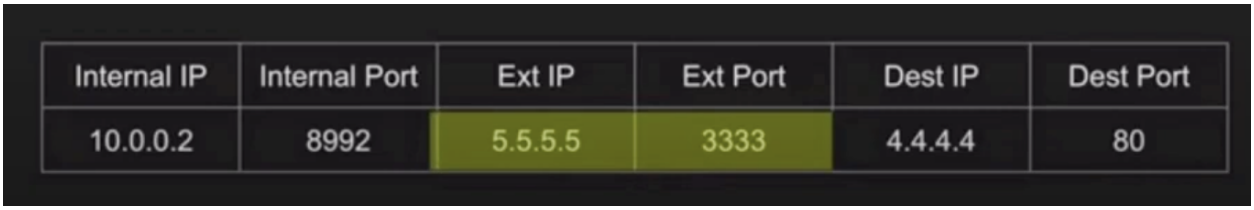


- RTCPEERCONNECTION ESTABLISHES CONNECTION BETWEEN USERS USING ICE PROTOCOLS.

DATA CHANNELS:

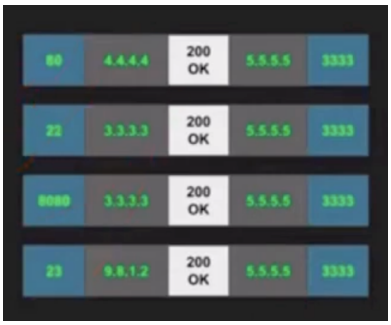
- THROUGH THIS TRANSFER OF DATA HAPPENS.

STUN AND TURN:



Internal IP	Internal Port	Ext IP	Ext Port	Dest IP	Dest Port
10.0.0.2	8992	5.5.5.5	3333	4.4.4.4	80

- ONE TO ONE NAT
  - Just public ip address and port of external is enough



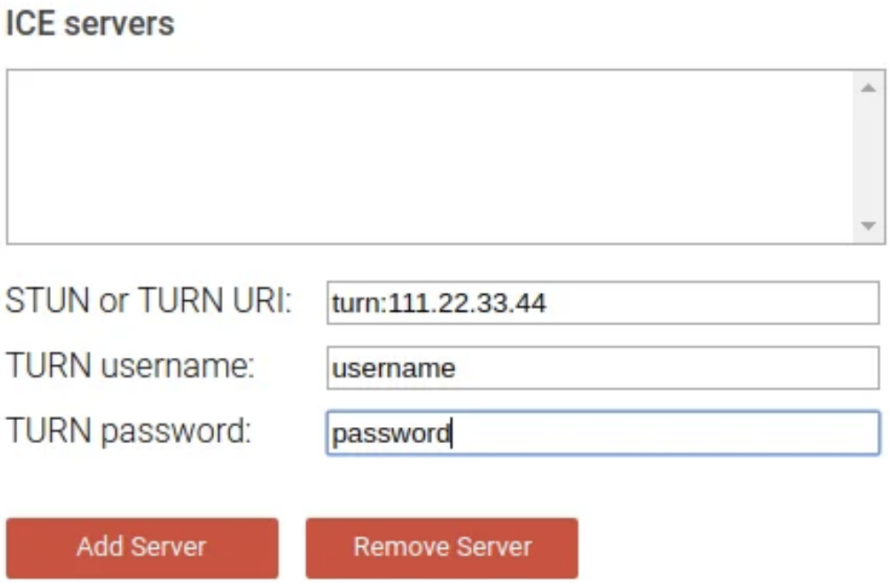
- ADDRESS RESTRICTED NAT
- PORT RESTRICTED NAT
- SYMMETRIC NAT
    - Along with public ip address and port of external, you also need

Internal IP	Internal Port	Ext IP	Ext Port	Dest IP	Dest Port
10.0.0.2	8992	5.5.5.5	3333	4.4.4.4	80
10.0.0.2	9999	5.5.5.5	4444	3.3.3.3	80
10.0.0.2	8888	5.5.5.5	2222	3.3.3.3	8080

TURN REGISTERS THE SENDERS PUBLIC IP AND PORT NUMBERS, By sending an hi message to the receiver.

SETTING UP TURN SERVER:

USER COTURN OPEN SOURCE


ICE servers
STUN or TURN URI:
turn:111.22.33.44
TURN username:
username
TURN password:
password
Add Server
Remove Server