WebRTC

- What? Why? How?
- Signaling
- Overview
- Connecting
- Security
- Communication
- Media Servers
- Challenges

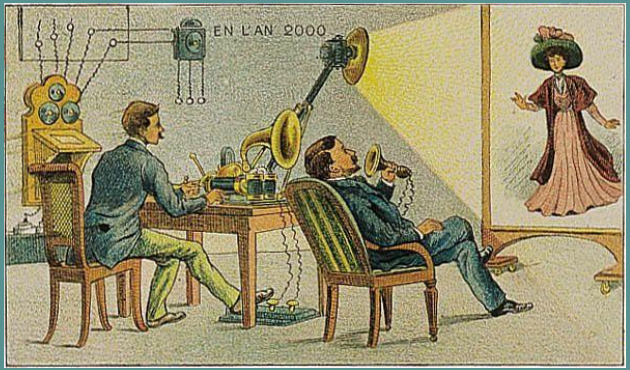
Mohammad Nasr

Software engineer @ snapp!
B.Sc. @ Amirkabir

What? Why? How?

Reasons:

- Need for video conferences and live communication.
- Open Standard with multiple APIs.
- Without any plugins like using telephone and etc .
- Secure and with low latency.
- Mandatory encryption
- Control over dynamic network
- Created by Google initially in 2011 (stable in 2018).

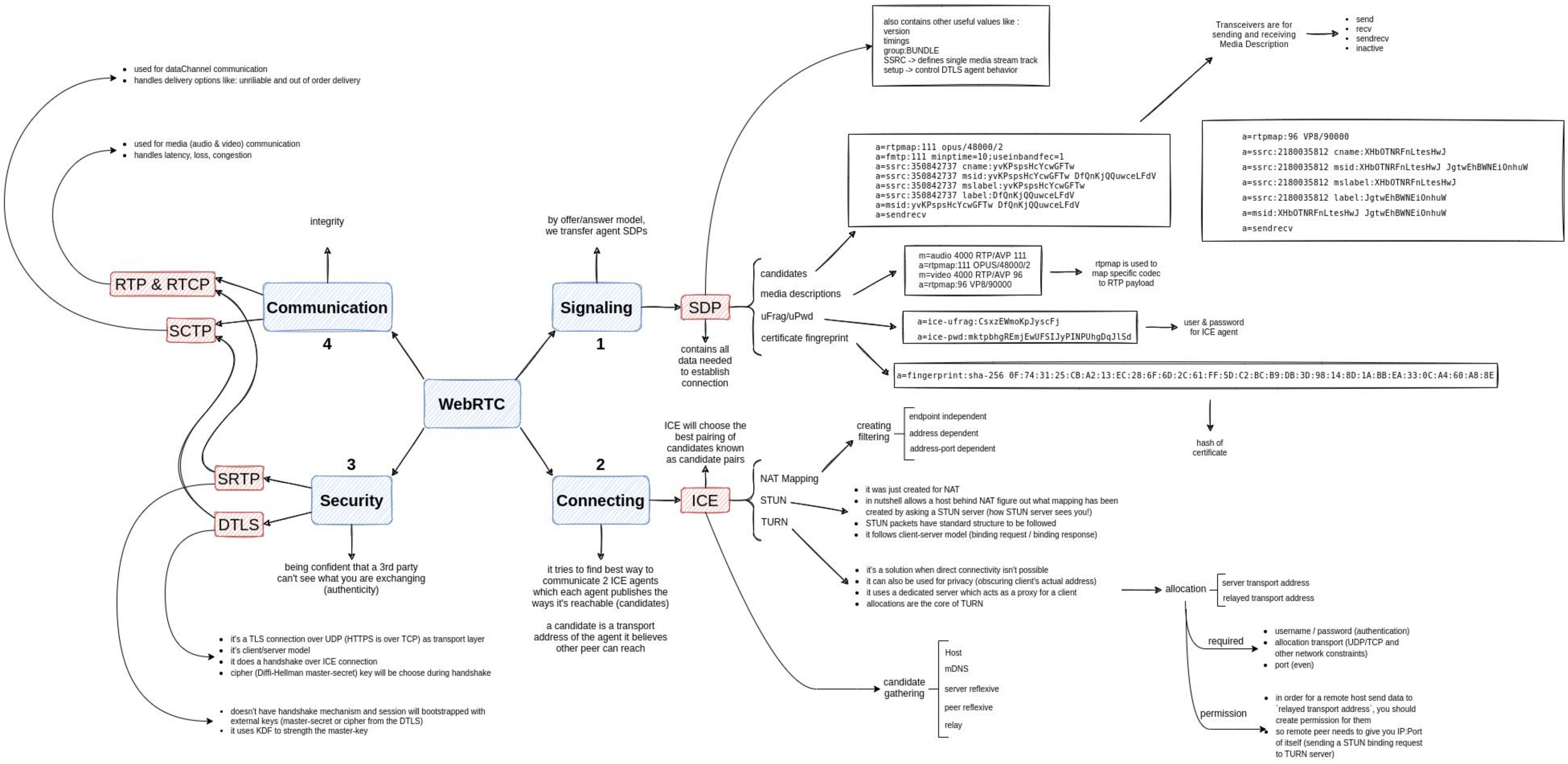


Introduction:

- Short for Web Real Time Communication, is both API and Protocol.
- Protocols are some set of rules to negotiate bi-directional secure real-time communication
- API allows developers to use webRTC
- Collection of other protocols like in era of networking, cryptography, Codecs and ... .
- Each step is made up of many other protocols! To make webRTC, so it only take advantage of subset of protocols.
- It’s a technology, neither a protocol or an application (skype, zoom and etc).
- There are tons of provided APIs for WebRTC but the general API is documented in https://www.w3.org/TR/webrtc/ as described in RFC8825 and RFC8826.
- We will gain some benefits by webRTC like reduced bandwidth, low latency, secure communication and etc .

Cont:

- Breaking down topics into 4 categories:
- Signaling: how peers find each other.
- Connecting: connecting to each other.
- Communication: having bi-directional communication of media and data.
- Security: having a secure communication.


WebRTC
Signaling
1
by offer/answer model,
we transfer agent SDPs
SDP
contains all
data needed
to establish
connection
candidates
media descriptions
uFrag/uPwd
certificate fingerprint
also contains other useful values like :
version
timings
group:BUNDLE
SSRC -> defines single media stream track
setup -> control DTLS agent behavior
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
a=ssrc:350842737 cname:yvKPspsHcYcwGFTw
a=ssrc:350842737 msid:yvKPspsHcYcwGFTw DfQnKjQQuwceLFdV
a=ssrc:350842737 mslabel:yvKPspsHcYcwGFTw
a=ssrc:350842737 label:DfQnKjQQuwceLFdV
a=msid:yvKPspsHcYcwGFTw DfQnKjQQuwceLFdV
a=sendrecv
Transceivers are for
sending and receiving
Media Description
send
recv
sendrecv
inactive
a=rtpmap:96 VP8/90000
a=ssrc:2180035812 cname:XHbOTNRFnLtesHwJ
a=ssrc:2180035812 msid:XHbOTNRFnLtesHwJ JgtwEhBWNEiOnhuW
a=ssrc:2180035812 mslabel:XHbOTNRFnLtesHwJ
a=ssrc:2180035812 label:JgtwEhBWNEiOnhuW
a=msid:XHbOTNRFnLtesHwJ JgtwEhBWNEiOnhuW
a=sendrecv
m=audio 4000 RTP/AVP 111
a=rtpmap:111 OPUS/48000/2
m=video 4000 RTP/AVP 96
a=rtpmap:96 VP8/90000
rtpmap is used to
map specific codec
to RTP payload
a=ice-ufrag:CsxzEWmoKpJyscFj
a=ice-pwd:mktpbhgREmjEwUFSIJyPINPUhgDqJlSd
user & password
for ICE agent
a=fingerprint:sha-256 0F:74:31:25:CB:A2:13:EC:28:6F:6D:2C:61:FF:5D:C2:BC:B9:DB:3D:98:14:8D:1A:BB:EA:33:0C:A4:60:A8:8E
hash of
certificate
Connecting
2
it tries to find best way to
communicate 2 ICE agents
which each agent publishes the
ways it's reachable (candidates)
a candidate is a transport
address of the agent it believes
other peer can reach
ICE
ICE will choose the
best pairing of
candidates known
as candidate pairs
NAT Mapping
STUN
TURN
creating
filtering
endpoint independent
address dependent
address-port dependent
it was just created for NAT
in nutshell allows a host behind NAT figure out what mapping has been created by asking a STUN server (how STUN server sees you!)
STUN packets have standard structure to be followed
it follows client-server model (binding request / binding response)
it's a solution when direct connectivity isn't possible
it can also be used for privacy (obscuring client's actual address)
it uses a dedicated server which acts as a proxy for a client
allocations are the core of TURN
allocation
server transport address
relayed transport address
required
username / password (authentication)
allocation transport (UDP/TCP and other network constraints)
port (even)
permission
in order for a remote host send data to `relayed transport address`, you should create permission for them
so remote peer needs to give you IP:Port of itself (sending a STUN binding request to TURN server)
candidate
gathering
Host
mDNS
server reflexive
peer reflexive
relay
Security
3
being confident that a 3rd party
can't see what you are exchanging
(authenticity)
SRTP
DTLS
it's a TLS connection over UDP (HTTPS is over TCP) as transport layer
it's client/server model
it does a handshake over ICE connection
cipher (Diffi-Hellman master-secret) key will be choose during handshake
doesn't have handshake mechanism and session will bootstrapped with external keys (master-secret or cipher from the DTLS)
it uses KDF to strength the master-key
Communication
4
integrity
RTP & RTCP
SCTP
used for dataChannel communication
handles delivery options like: unriliable and out of order delivery
used for media (audio & video) communication
handles latency, loss, congestion

Signaling

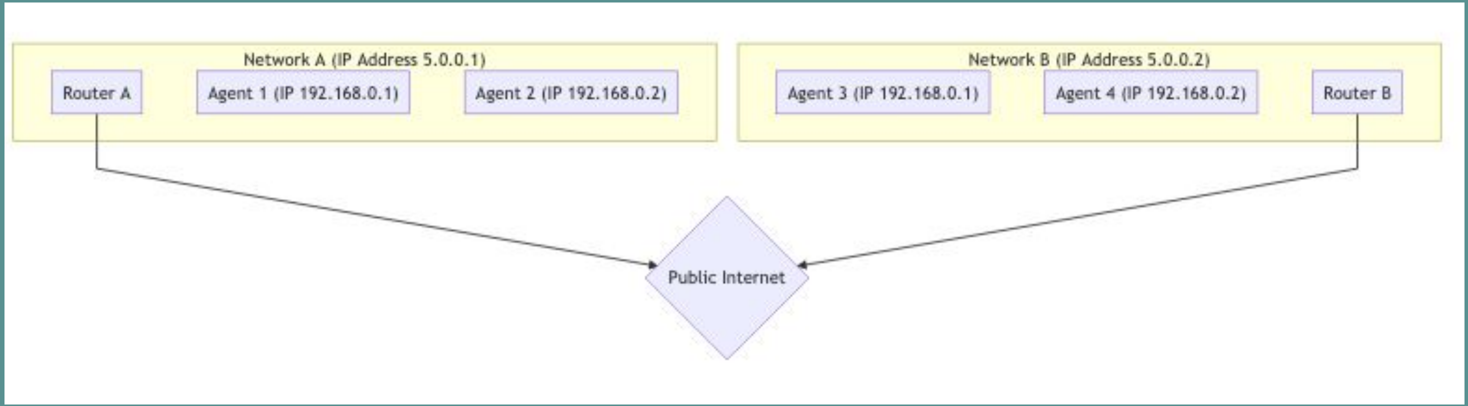
- Mostly about how peers find each other and used to bootstrap the call.
- WebRTC doesn't specify anything about signaling transport, it can be a HTTP call, websocket, whatsapp barcode and ... .
- Uses SDP protocol (key-value text protocol) to share all the required state to establish connection, it describes the Media-Descriptors (with respective direction like send, recv, sendrecv, inactive), required codecs (like OPUS and VP8), fingerprints (hash), ice-ufrag (for ICE traffic username), ice-pwd (for ICE traffic user authentication) and etc .

```
...

a=fingerprint:sha-256 0F:74:31:25:CB:A2:13:EC:28:6F:6D:2C:61:FF:5D:C2:BC:B9:DB:3D:98:14:8D:1A:BB:EA:33:0C:A4:60:A8:8E
m=audio 4000 RTP/AVP 111
a=rtpmap:111 OPUS/48000/2
a=sendrecv
m=video 4002 RTP/AVP 96
a=rtpmap:96 VP8/90000
a=sendrecv
a=candidate:foundation 1 udp 2130706431 192.168.1.1 53165 typ host generation 0
a=candidate:foundation 2 udp 2130706431 192.168.1.1 53165 typ host generation 0
a=ice-ufrag:CsxzEWmoKpJyscFj
a=ice-pwd:mktpbhgREmjEwUFSIJyPINPUhgDqJlSd

...
```

Connecting

Real-world networking:

- Not in the same network (no direct connectivity).
- Protocol restrictions like ban UDP or TCP traffic
- Firewall rules: filter and drop webRTC packets



Some networking attributes:

- Bandwidth: maximum rate of data that can be transferred.
- Transmission Time: how long it takes for a packet to arrive to destination.
- Jitter: is the fact that transmission time vary for each packet.
- Packet Loss: when packets are lost in transient.
- Maximum Transmission Unit: the limit on how large a single packet can be.
- Congestion: when we reached the limit of network and will drop the packets.
- Dynamic: networks are incredibly dynamic and conditions can be changed rapidly.

- When 2 agents know enough details of each other then they try to connect
- We use ICE protocol (predates webRTC) to establish bi-directional connection
- In webRTC instead of client-server we use P2P model, both peers are equally in a same level, task of creating connection is distributed to peers
- The procedure of gathering all transports and establishing a reliable connection is called ICE.
- ICE tries to find best way of communication between agents and the ICE is made up of candidates which are generally transport addresses of agents
- NAT is the magic that makes webRTC connectivity possible, it’s about behavior of network and how to overcome the limitations (endpoint independent, address dependent, address-port dependent)

NAT:

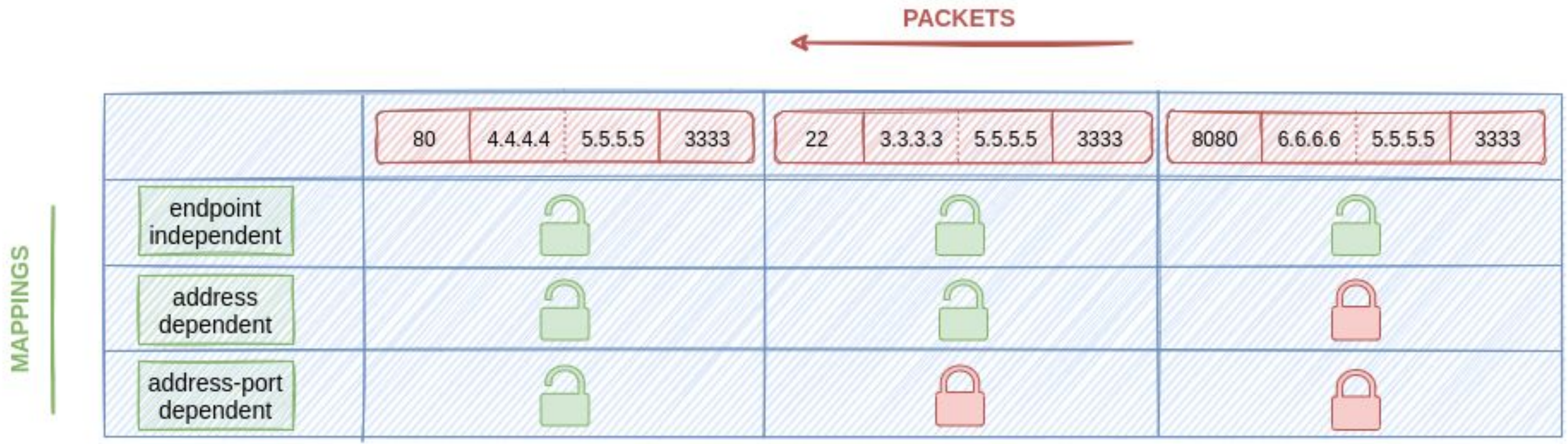
- is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

- Creation (sending a packet to outside network)
    - Endpoint-independent
    - Address dependent
    - Address-Port dependent

- Filtering (who is allowed to use the mapping)
    - Endpoint-independent
    - Address dependent
    - Address-Port dependent

IP TABLE

Internal IP	Internal Port	External IP	External Port	Destination IP	Destination Port
10.0.0.2	8992	5.5.5.5	3333	4.4.4.4	80
10.0.0.2	8995	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



STUN:

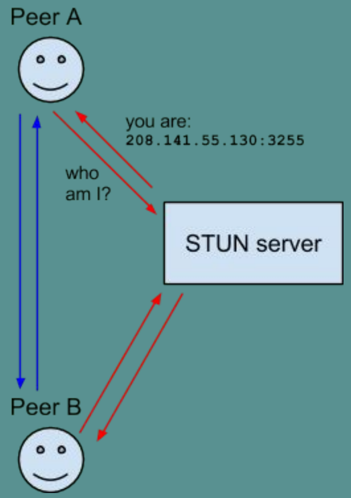
- Stands for Session Traversal Utilities for NAT that predates ICE and was created just for working with NATs.

- In a nutshell. STUN helps an endpoint behind a NAT figure out what mapping was created by asking a STUN server outside NAT to report what it observe.

- Creating a NAT mapping using STUN, just takes to send a STUN request and a response from a STUN server (like google STUN server: stun:stun.1.google.com:19302)

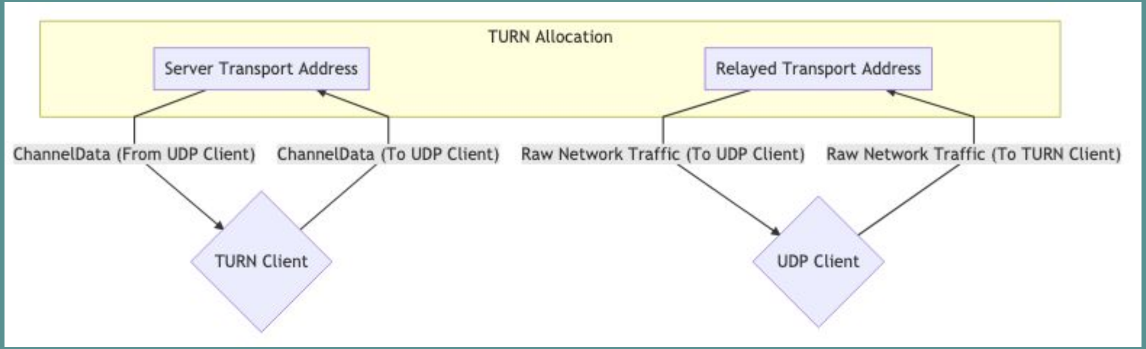
- STUN packet structure:

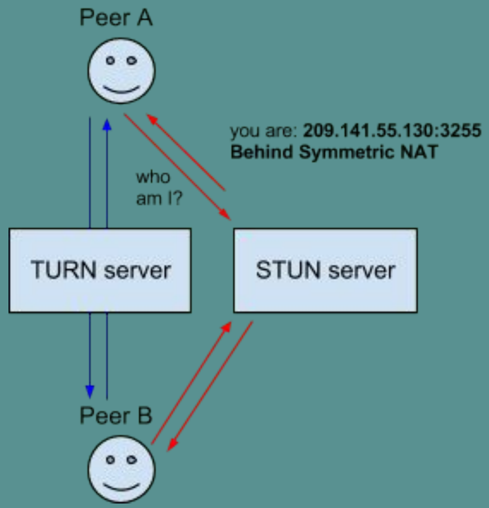
```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0|     STUN Message Type     |         Message Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic Cookie                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Transaction ID (96 bits)                  |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

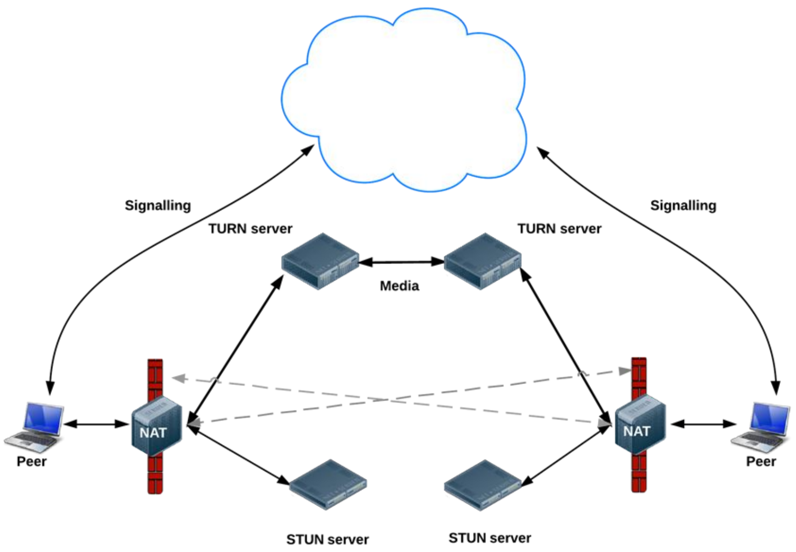


TURN:

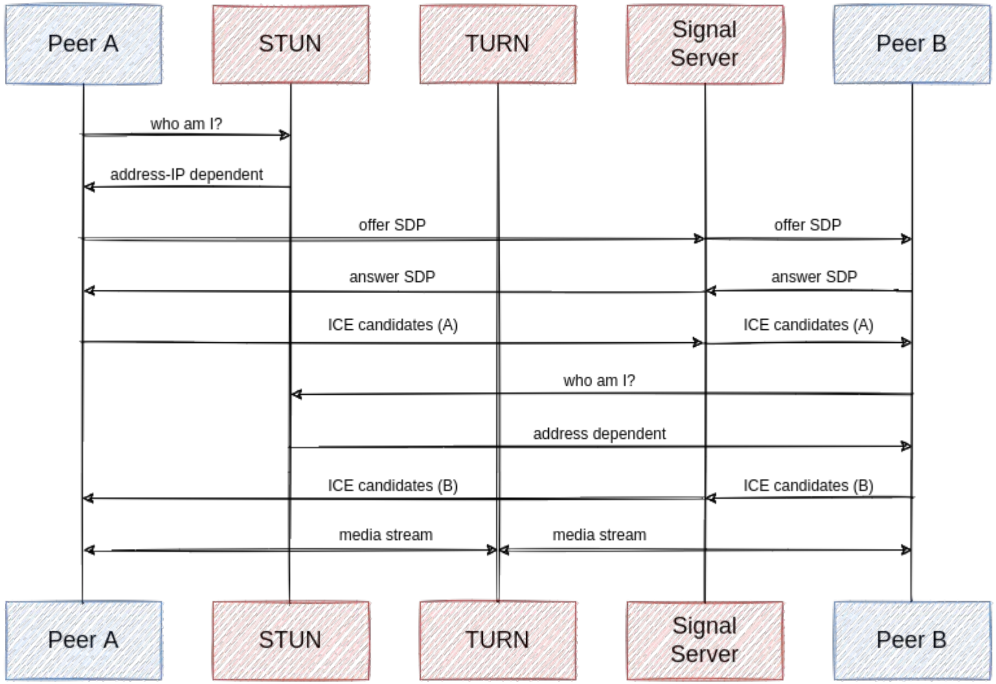
- Stands for Session Traversal Using Relays around Nat.
- is a solution when direct connectivity isn’t possible and it will uses a dedicated server which acts as a proxy server (relay server) to forward webRTC traffic throw that.







Signalling
Signalling
TURN server
TURN server
Media
NAT
NAT
Peer
Peer
STUN server
STUN server



Peer A
STUN
TURN
Signal Server
Peer B
who am I?
address-IP dependent
offer SDP
offer SDP
answer SDP
answer SDP
ICE candidates (A)
ICE candidates (A)
who am I?
address dependent
ICE candidates (B)
ICE candidates (B)
media stream
media stream
Peer A
STUN
TURN
Signal Server
Peer B

- Solving packet loss: with acknowledgments variations.
- Solving Jitter: using some buffer mechanism.
- Solving Congestion: first detecting we have congestion (bandwidth, packet-loss and etc) and then resolve it by sending slower or sending less

Communication

Media (audio/video):

- We can have unlimited amount of audio and video streams (independent or bundled)
- Storing 30 mins uncompressed 720 8-bit video takes 110 GB space!
- Compressions can be lossy or lossless.
- Compressions are Inter(bits in a single frame) or Intra frame(frames themselves and can be I-FRAME(complete frame), P-FRAME(changes from previous), B-FRAME(pre and after changes)).
- We use RTP (Real-time Transport Protocol) and RTCP (RTP Control Protocol) protocols for this purpose to carry the media (predate webRTC)
- Every RTP packet contains SSRC, timestamp and encoded payload
- RTCP is RTP control protocol which carries RTP metadata and statistics which will be used to handle packet loss, latency and Jitter (out-of-order)
- Real-Time is about trade-offs between latency and quality (more latency -> higher quality)

DataChannel (data):

- We use SCTP (Stream Control Transmission Control) transport layer protocol for this purpose to carry the data which is an alternative to TCP and UDP, in webRTC it runs over DTLS.

- By design of this protocol, it will handle delivery order and unreliability of communication.

- You can have unlimited steams and each one can be configured (durability and ordering) separately, dataChannels are just abstractions for SCTP.

- Why not websocket? Because it’s configurable to behave like UDP with unordered and lossy delivery (low latency and high performance).

Security

Introduction:

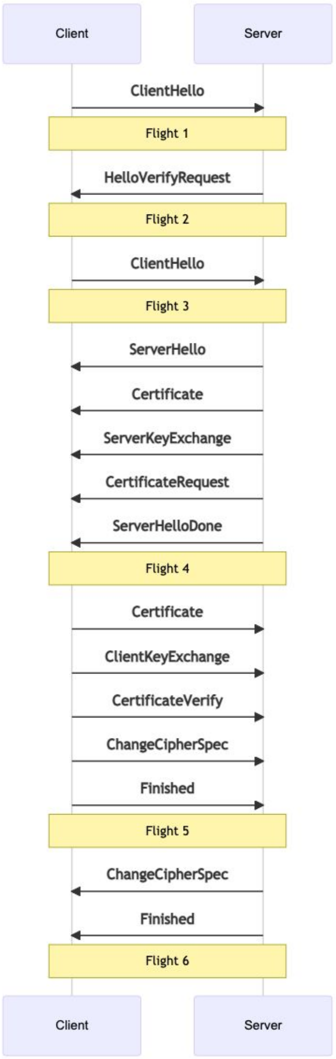
- Every webRTC connection is authenticated and encrypted and you can be confident that a 3rd party can’t see what you are transferring.
- It’s very important no-one can tempers those data
- Securing transport layer with DTLS and SRTP and it’s all about .

Cryptography:

- Cipher: series of steps that takes plaintext to ciphertext
- Hash Functions: one-way process to generate a digest
- Public-Private key cryptography: having 2 keys, one for encryption and another one for decryption.
- Diffie-Hellman exchange: sharing a secret key (discrete logarithm problem)
- Pseudorandom Function: predefined functions to generate some value from other value that appears random.
- Key Derivation Function: type of Pseudorandom Function that is used to make a key stronger

DTLS (Datagram Transport Layer Security):

- DTLS is just a TLS (Transport Layer Security) over UDP (User Datagram Protocol) to exchange data securely.
- WebRTC connection will start with DTLS handshake over ICE connection.
- Like HTTPS we don't use CAs and we just use the certificates exchange during DTLS to match the fingerprint we sent via signaling.
- During handshake, a cipher key (via Diffi-Hellman) will be chosen.



SRTP (Secure Real-time Transport Protocol):

- We use it to exchange media securely.

- It uses existing DTLS cipher key and then by applying KDF (Key Derivation Function) will generate a key for SRTP session .

Media Servers

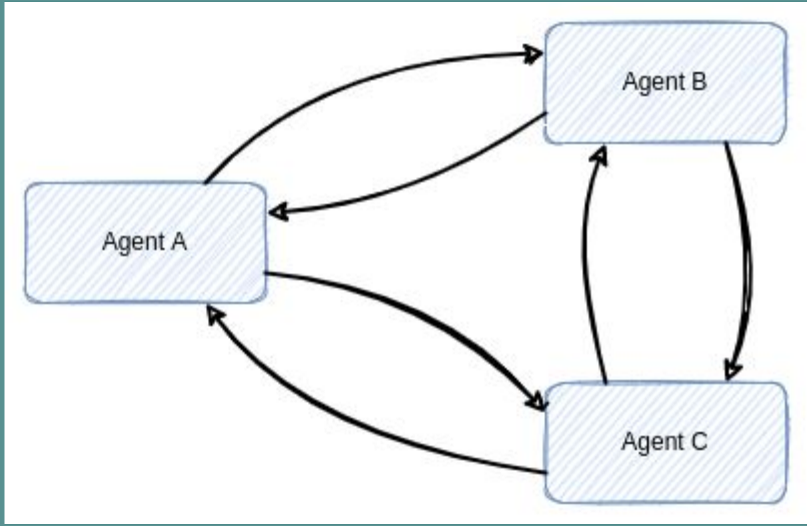
Introduction:

- Normally in a group call with pure webRTC we have to provide tons of mutual P2P connections!, they can have interruptions and … .

- It’s a central server to process webRTC media, filtering, enhancements, facilitate group calls, provide calling features, recording and … .

Full Mesh:

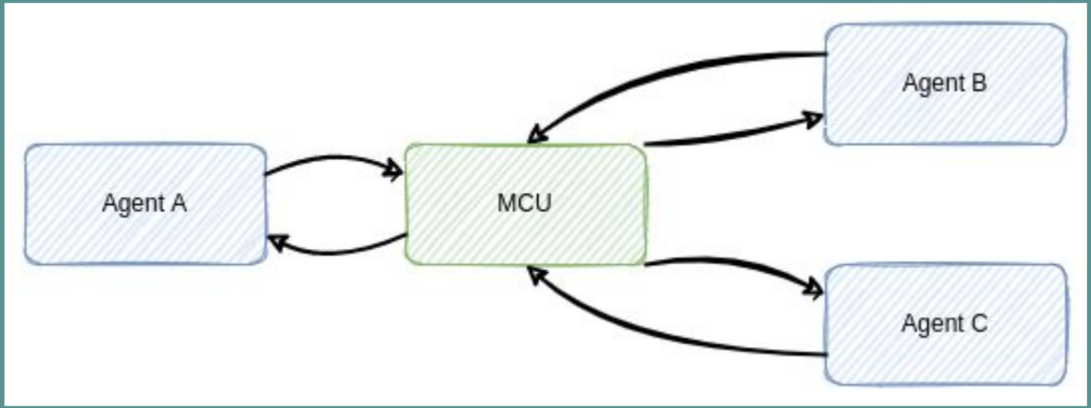
- In this topology each user establishes a connection with every user directly.
- You have to encode and upload media for each member of the call separately.
- Good for small groups.



MCUs (Multipoint Control Units):

- Each peer establishes a connection to the server and send up its media, the MCU in turn will makes a composite media of the group media and forward it to each peer

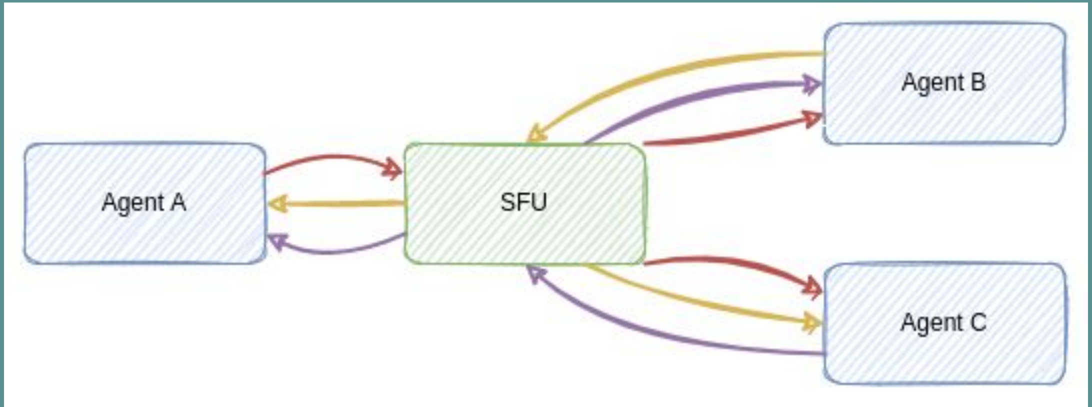
- Better bandwidth for peers, more CPU usage



SFUs (Selective Forwarding Units):

- Like MCU here each peer establishes a connection to the server but SFU doesn’t make any composite stream, rather it sends different received input streams to each user (except itself media stream)

- Reduced CPU, can be E2E encrypted



Challenges

Signaling:

- Choosing appropriate protocol.
- NAT

Scaling:

- Easy deployment and maintenance.
- Easy replication process.

Security:

- Security of snapp and users.
- Forbid DDos (denial of service attack), MITMA (man in the middle attack) and etc.

Resources:

- https://github.com/snapp-incubator/ghodrat/
- https://github.com/pion/webrtc
- https://github.com/pion/ion-sfu
- https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
- https://gabrieltanner.org/blog/broadcasting-ion-sfu
- https://webrtc.ventures/2020/12/webrtc-media-servers-sfus-vs-mcus/

Thank you for listening: