Zoom video calling System Design

Facebook video calling

Whatsapp video calling

Tango video calling ☺

Functional Requirements:

(1) One to one calls

(2) Group calls

(3) Audio / Video / Screen Share

(4) Recording

Non-Functional Requirements:

(1) Super Fast (Even low latency is unacceptable)

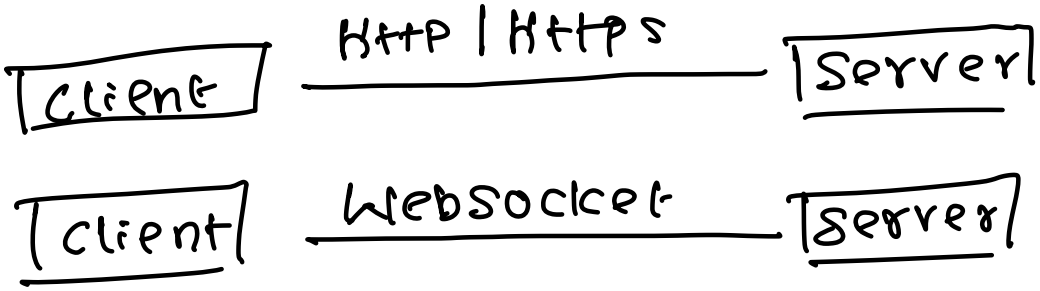
(2) Highly Available

(3) Data loss is okay (Consistency compromise)

Architecture:

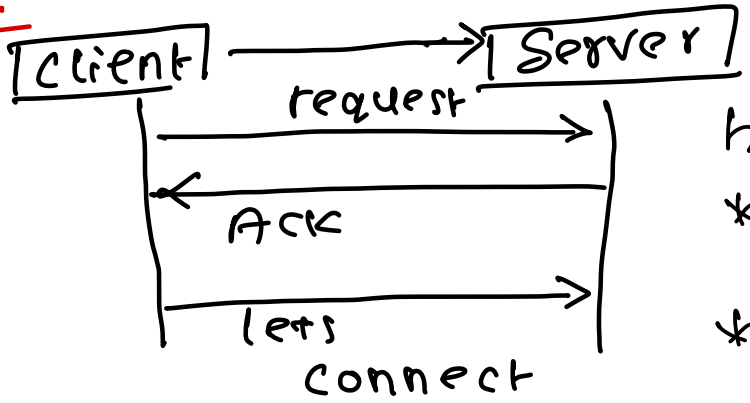
Client-Server Architecture

Communication channels between clients & servers:



All three i.e., HTTP/ HTTPS / Websockets are built on TCP Transport Layer Protocol

TCP:



*TCP is a three way hand shake.

*packets will be reshared if they are lost

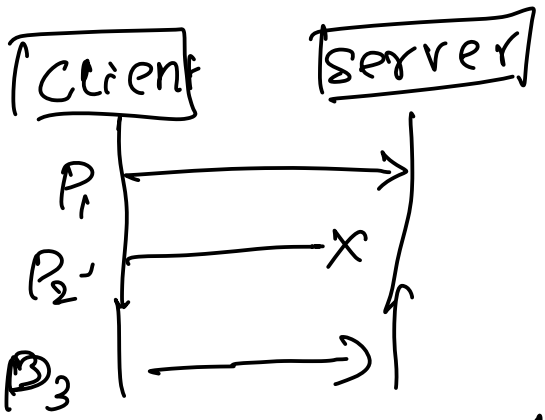
*Order of packets shared is maintained.

Why not TCP for Video Calling??

Video calling does not expect any delays but if we opt for TCP -- Three way handshake This may lead to low latency. Therefore, No to TCP.

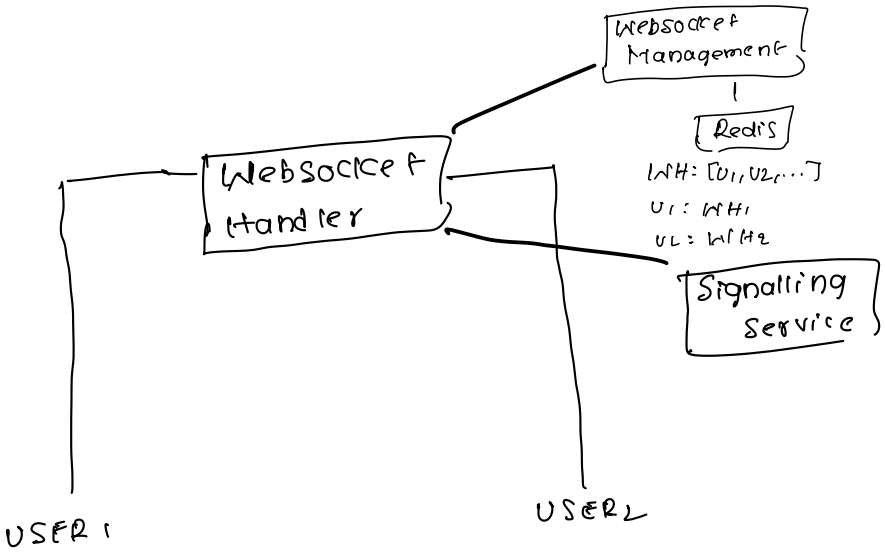
But we can opt for WebSocket communication between servers and users other than video calling.

UDP Transport Layer Protocol -- Video calling:



* There is no three way hand shake
* Header has only 8 bytes unlike TCP 20 byte header
* There is data loss but it is okay to loose few frames of data in a live call.

ONE TO ONE VIDEO CALLING:



Dry Run:

* USER 1 communicates with Websocket Handler that it want to make a video call to USER 2
* WH gets the WH data of USER 2 from Websocket Management service
* WM Management service keep track of all users and respective WHs information in Redis
* Upon receiving information of corresponding WH of USER 2, WH of USER 1 will request it to communicate the video call request from USER 1
* USER 2 gets the information related to the video call request with the help of Signalling service

* Possibility one:

(1) USER2 is OFFline. If offline Video call cannot be completed

* Possibility Two:

(1) USER2 is ONLINE but rejected the call,

=> Here, Klit of USER2 updates the information to Signaling service

=> Klit of USER1 updates the user, with either error message or rejected message

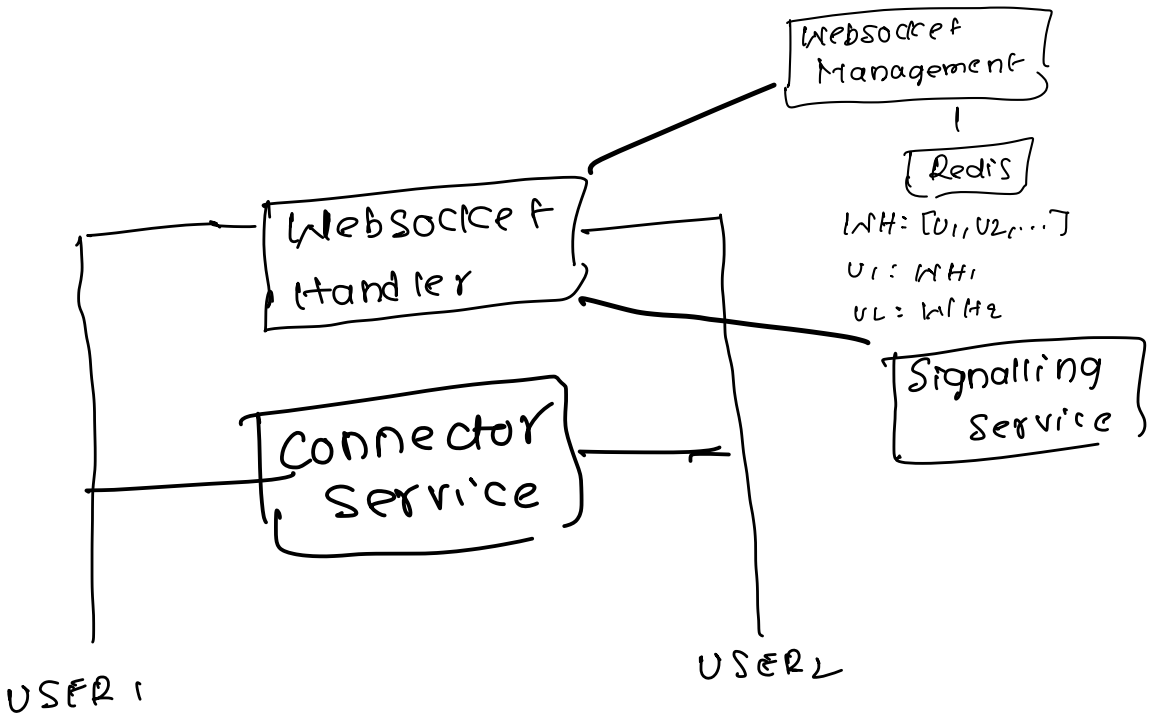
* Possibility Three:

(1) USER2 accepts the call.

In this case, We cannot communicate via webSockets

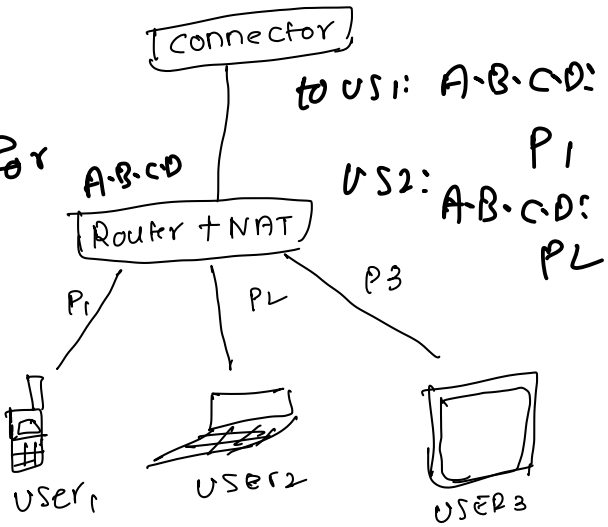
Therefore, we introduce connector service to the existing design

Addition of Connector Service:



(1) Connector service is introduced to eliminate TCP Connection.

(2) Instead of TCP, If we opt for peer 2 peer communication we can eliminate latency issues



(3) To obtain P2P communication, connector service is introduced

(4) connector service helps in identifying the public IPs of users requesting for video call

(5) Most of the devices we use are IPV4 which dont have separate public IPs.

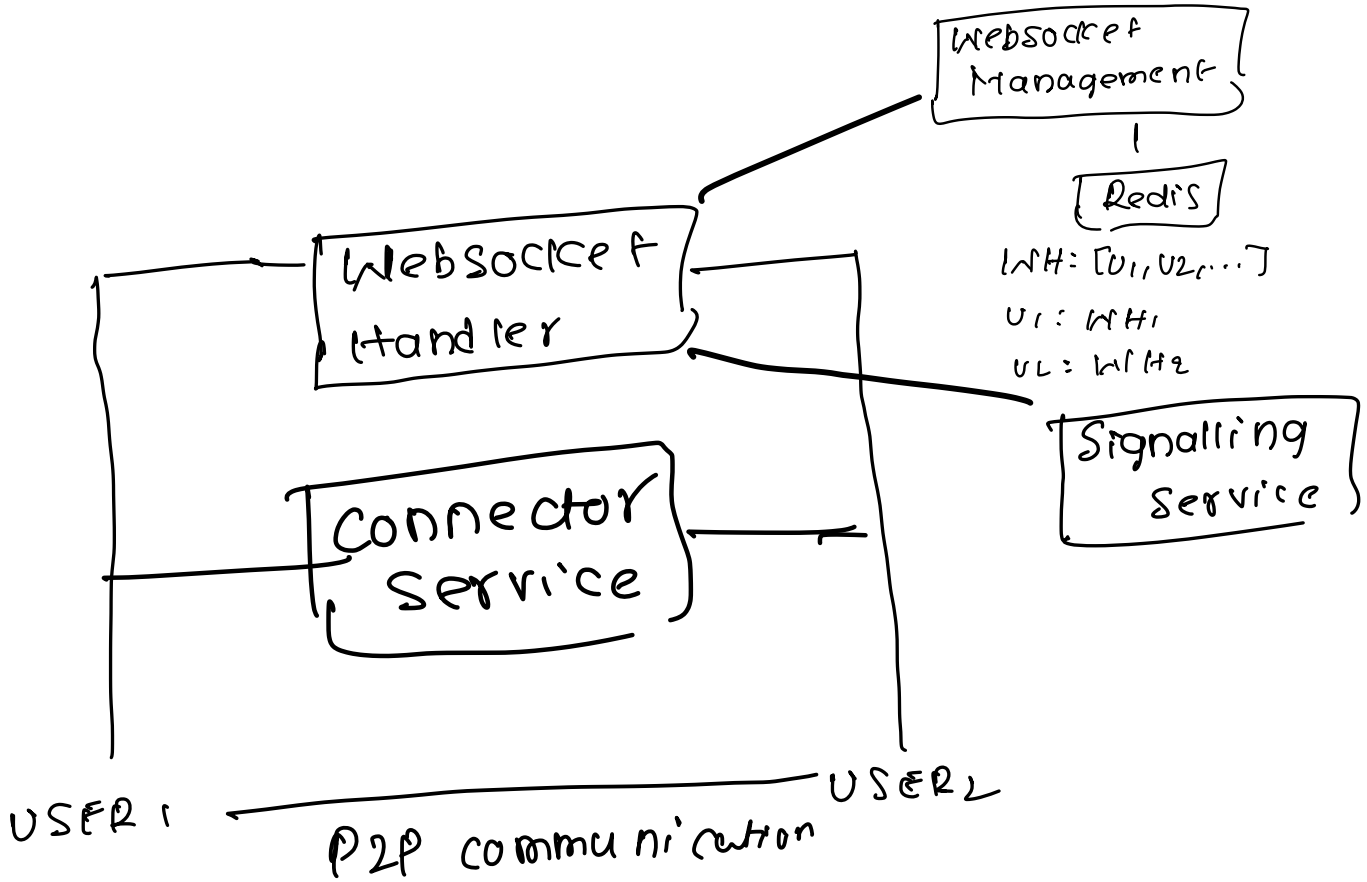
(6) To attain public IPs. At first, user1 requests connector service for it's public ip connected via routers and ISC

(7) Now, connector will return IP of ISC and it's port as public IP since IPv4

A.B.C.D : P1

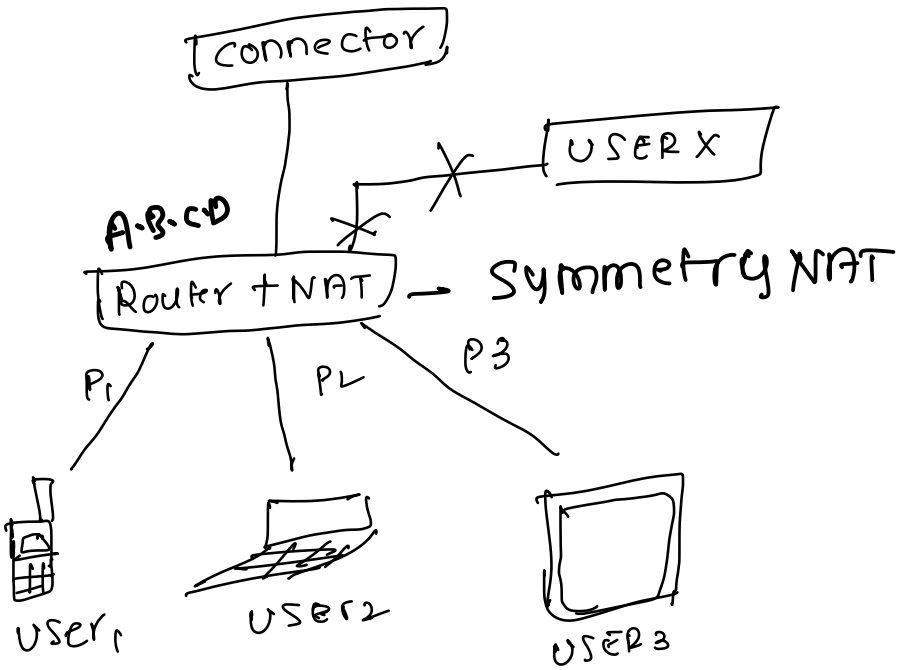
(8) once users receive their IP addresses, they will exchange with each other via websocket handlers.

(9) upon receiveing each others public IPs they communicate peer to peer



ADDITION OF CALL SERVICE:

1) In reality, most of the ISPs will have firewalls and symmetry-NAT which dont allow more than one communication at a time

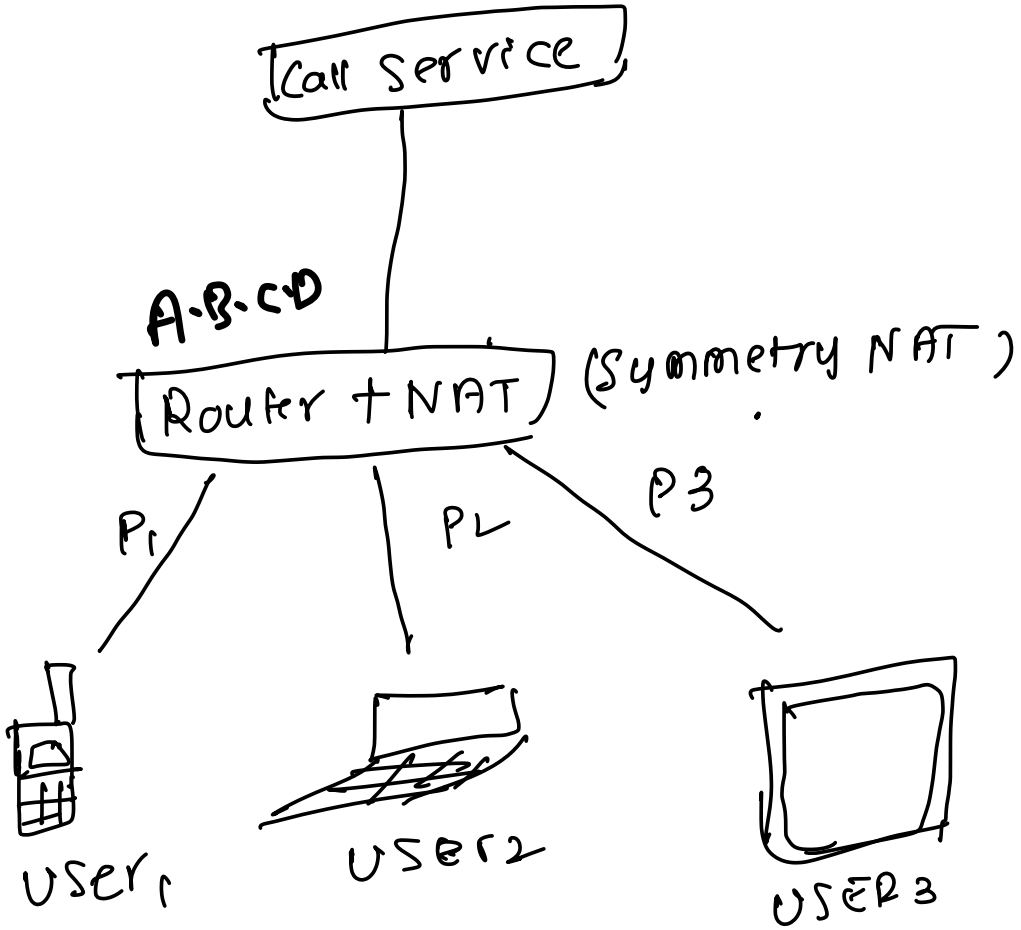


2) Say User X wants to have a video call with USER1. If symmetry NAT is present at ISP then it will reject the connection

3) To fix this issue, A new service called call service is introduced.

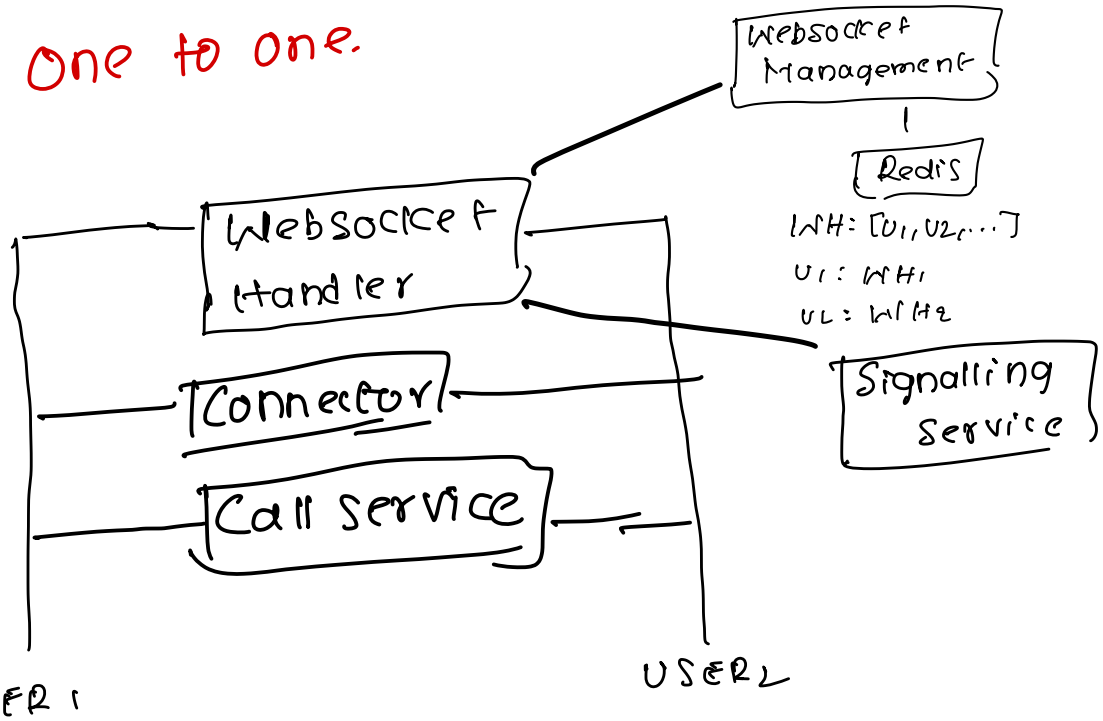
4) The functionality of call service is similar to connector service - i.e, it will get public IPs of each other. Along with that, It will act as an intermediate between the users for communication via video call.

5] USER1 pushes stream of bytes to call service and USER2 pulls bytes from call service and vice-versa

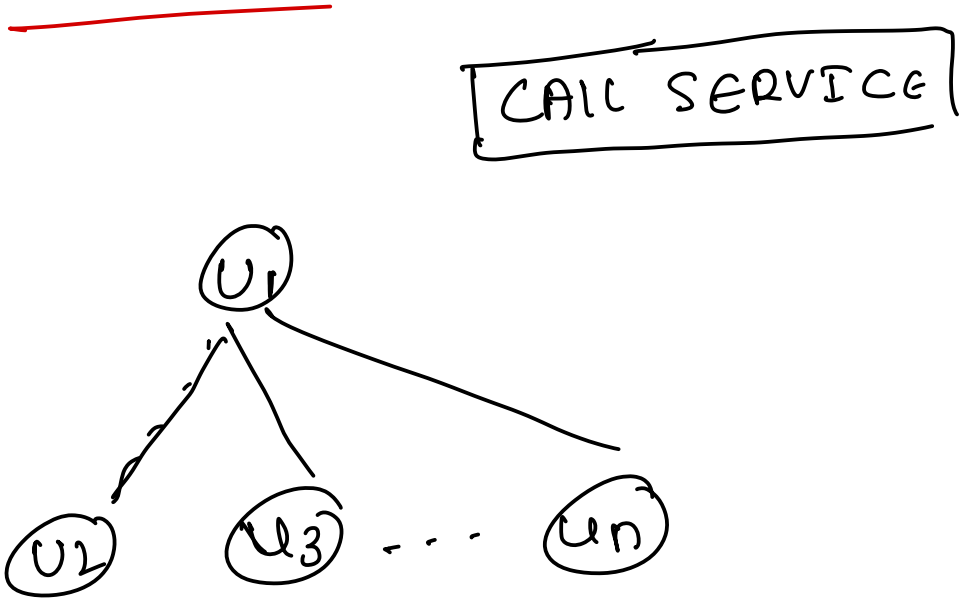


Overall architecture:

One to one.



GROUP CALL:



Assumption of having P2P Communication:

If we opt for P2P, then every user has to experience (N-1) latency and bandwidth.

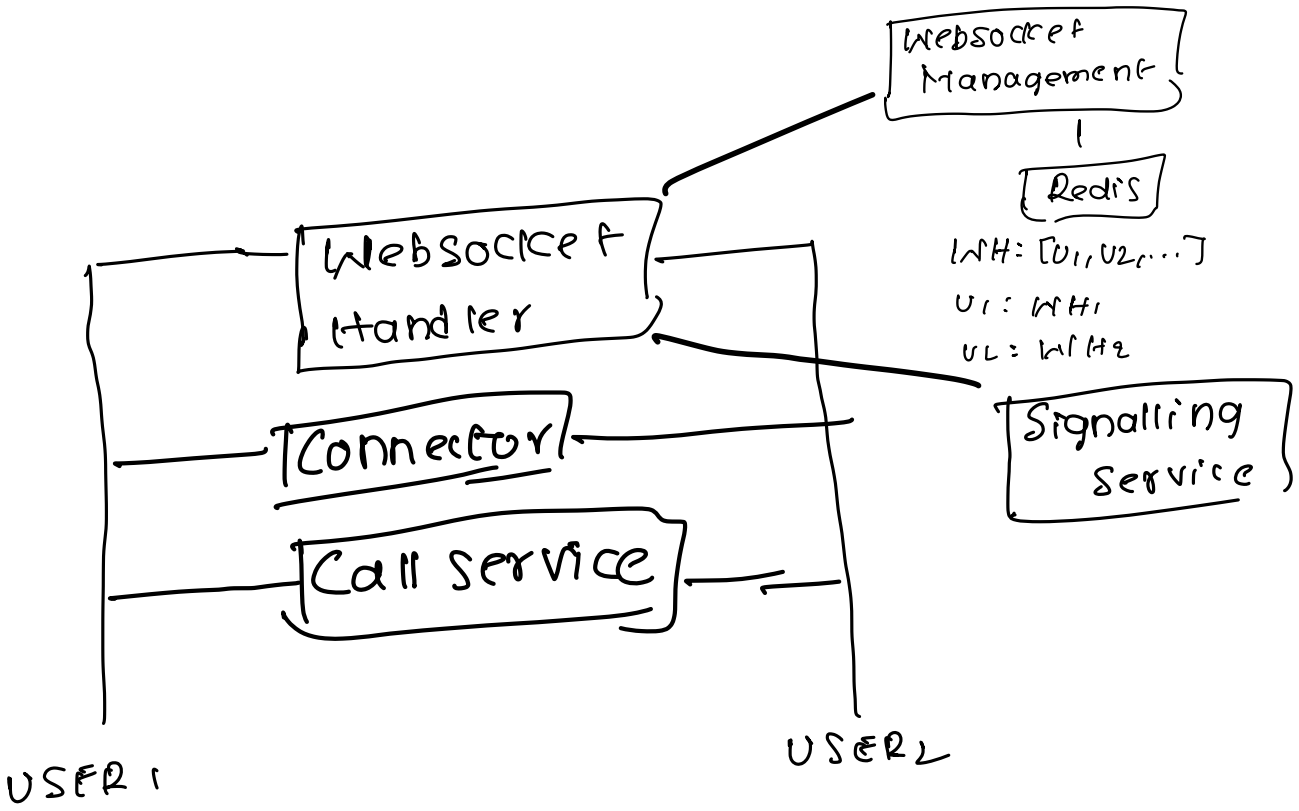
We are unsure of every devices latency and bandwidths.

Therefore, P2P is impossible when it comes to group call.

Instead we can push all the data to Call service and pull all the data from individual devices.

By doing this we can eliminate dependency on individual devices latency and bandwidth

Web Real Time communication (WebRTC)



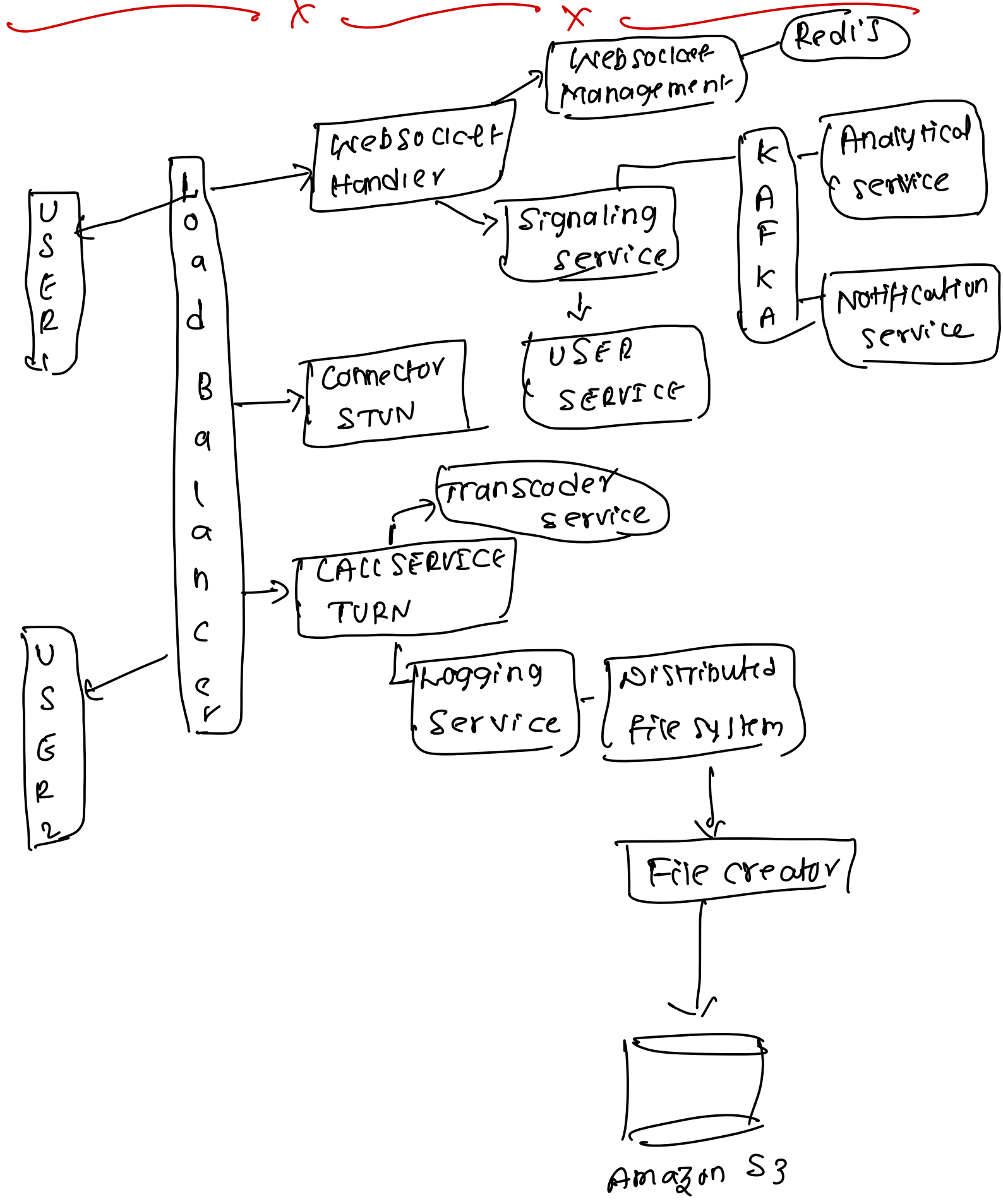
1) Connector: Connector is called STUN in web RTC and the connector is inspired from the same

2) CALL SERVICE: Call service is known as TURN in webRTC.

3) Websocket Handler:

Websocket Handler is known as signaling.

OVERALL ARCHITECTURE OF VIDEO CALLING:



DRYRUN. 1 to 1

(1) User1 wants to make video call to User2. Therefore, request WebSocket Handler

(2) WebSocket Handler gets the relavent data of WebSocket Handler of User2

(3) parallely, WebSocket Handler1 of User1 communicates with signaling service that it wants to communicate with user2

(4) Now, WebSocket Handler of user2 notifies that user1 wants to connect after receiving event from signaling service

(5) If user 2 is offline do nothing

(6) If user 2 reseCts → Notify to UI1 of user1 via signaling service

(7) If User2 accepts → Call connector service to get public IPs to establish P2P connection

(8) If established, continue with the call

(9) If not due to Symmetric NAT, call call service

(10) Now, data is transfered via call service.

DRY RUN: Group

(1) User1 wants to communicate with a group

(2) Gets the WHS information all the members in the group via WebSocket Management Service.

(3) Call call service. Now, Call service communicates to all but with the help of transcoding service

(4) Every device cannot have same bandwidth

Therefore, transcoding service after receiving the information of every device latency and bandwidth from signaling service : It will transfer data in that format

Recording Service:

All the information is communicating via call service. Therefore, a logger service is introduced at call service to collate all the data that is being streamed.

The logger service now transfers all the aggregated data in chanks to file distributed system.

Once video ends, the signaling service triggers file creation service. which creates a file

and aploads to Amazon S3.

Notification service:

Edge cases:

While having P2P communication, there is a possibility of one user to change IP Therefore, in this case, Websocket handler should inform this to signaling service and change to call service.

Group call Transcoding service Edge case:

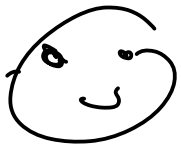
Every device latency and bandwidth changes every sec. Therefore, Transcoding service should update the resolution as per the need.

Analytical service:

To reduce issues in video calling, we can share all the signaling service information to Analytical service via kafka topic. We can analyse the data and reduce issu.

Metrics:

We can log -- video call timings between users for auditing purpose.

for auditing purpose 

Hahahaha---

Data center:

Region based approach.