- Peer to peer
- However it is very difficult to scale it in the p2p environment
- A group call in P2P translates into a mesh network, where every WebRTC client has a peer connection opened to all other clients directly.

Benefits

- **It is cheaper to operate**. Since there are no media servers, the media flows directly between the users. With WebRTC, oftentimes, the biggest cost is bandwidth. By not routing media through servers as much as possible the cost of running the service reduces drastically
- **It is more private**. Yap. As the service provider you don’t have any access to the media, since it doesn’t flow through your servers, so you can market your service as one that offers a higher degree of privacy for the end users

Cons

- Because it brings with it a lot of challenges and headaches when it comes to bandwidth and CPU requirements. So much so that it fails miserably in many cases.
- In WebRTC mesh, we put a bigger strain on the uplink in terms of bandwidth
- In an office setting, where people need to use the network in parallel, giving every user in a remote meeting 15Mbps uplink won’t be possible.
- Video coding is a CPU (and memory) hog. Encoding is a lot worse than decoding when it comes to CPU resources. Having 10 decoders is hard enough. 10 encoders is brutal. Each video stream from our speaker to the viewers has its own dedicated video encoder. With our 10 viewers, that means 10 video encoders.

Components of WEB_RTC

STUN

- STUN protocol enables devices that are behind a NAT to discover their own public IP address and port number.
- .The devices communicate with a STUN server that is on the internet, and the STUN server provides this information to the client device. The device can then share this information with another device or devices on the Internet with which it wants to communicate. This allows external devices to communicate with each other directly, effectively traversing the NAT.

Understanding the NAT Problem

A router will have a public IP address and every device connected to the router will have a private IP address. Requests will be translated from the device's private IP to the router's public IP with a unique port. That way you don't need a unique public IP for each device but can still be discovered on the Internet.

- Routers map multiple private IP addresses to a single IP public IP address using a technique called NAT or Network Address Translation. This allows multiple devices that are on the local network to connect to the internet using a single public IP.
- NAT blocks direct communication between devices by blocking inbound traffic, altering the port mappings and hiding the public IP address from the devices that are behind the NAT.

The core functionalities of STUN include:

- Public IP address discovery: STUN allows a client device to learn its public IP address
- Port Mapping: STUN helps the client device know what port number it has been assigned by the NAT device

How STUN Works

- When a client device sends a request to the STUN server, which is on the internet. The STUN server can see the public IP address and port number from which the request is coming from.
- The STUN server then sends this information back to the client. This is how the STUN server helps devices discover their own public IP and port number that is assigned to them by the NAT router.

TURN

- Some routers using NAT employ a restriction called 'Symmetric NAT'. This means the router will only accept connections from peers you've previously connected to.
- While TURN is more robust than STUN in that it assists in traversal of more types of NATs, a TURN communication relays the entire communication through the server requiring far more server bandwidth than the STUN protocol, which typically only resolves the public facing IP address and relays the information to client and peer for them to use in direct communication. For this reason, the ICE protocol mandates STUN usage as a first resort, and TURN usage only when dealing with symmetric NATs or other situations where STUN cannot be used.

Multi-party video conferencing

- Video conferencing between multiple parties is more complex because the peers may have different capabilities and network conditions: one particular video stream resolution, rate,

and quality, may not suit all recipients, and at the same time it is not efficient or scalable for a sender to generate and send multiple streams to many recipients.
- The most common approach to address these issues is to use an intermediary server known as a *Selective Forwarding Unit* (SFU) or *Selective Forwarding Middlebox* (SFM). Senders output video encoded such that the SFM can selectively forward an appropriate video stream for each recipient. There are two main technologies used by WebRTC for encoding video in this case: simulcast and scalable video coding.

SFU VS MCU

- In an MCU((Multipoint Control Unit)) topology, each of the client devices is connected to a centralized MCU server, which decodes, rescales, and mixes all incoming streams into a single new stream and then encodes and sends it to all clients. Although bandwidth-friendly and less CPU-intensive on the client side—instead of processing multiple streams, devices have to decode and render only one stream—an MCU solution is rather expensive on the server side. Transcoding multiple audio and video streams into a single media stream and then encoding it at multiple resolutions in real time is very CPU intensive, and the more clients connected to the server, the higher its CPU requirements.
- One of the greatest benefits of an MCU, however, is its ease of integration with external (legacy) business systems because it combines all incoming streams into a single, easy-to-consume outgoing stream. While WebRTC-based web applications are increasingly ubiquitous, having the ability to integrate with other already written systems is essential.
- **Advantages of MCU Architecture**
  - Bandwidth friendly
  - Composite output simplifies integration with external services (beyond WebRTC)
  - Your only option when you need to combine many streams (unless you use an XDN approach, which we'll discuss next)
- **Drawbacks**
  - CPU-intensive; a large number of video and audio streams requires a beefier server
  - Single point-of-failure risk because of centralized processing
  - High operational costs due to computational load on the server

What are the benefits of using an SFU?

SFUs have a number of benefits over other types of media servers:

1. SFUs can support a large number of parties.
2. SFUs can provide transcoding capabilities.
3. SFUs can provide bandwidth management capabilities.
4. SFUs can provide security features.

5. SFUs can provide quality of service features.

ICE

- ICE helps WebRTC traverse NATs and firewalls by allowing devices to find and use the most efficient network path between them, regardless of how the network is configured.
- ICE achieves this through a process called "ICE gathering." During this process, devices exchange ICE candidates, which are potential network addresses. These candidates describe the different paths that a device can take to establish a connection. The ICE protocol then evaluates these candidates and selects the most suitable one for establishing a connection, even when NATs and firewalls hinder Peer-to-peer communication.
- ICE candidates serve as addresses that devices use to connect with each other over the internet. A device can have multiple ICE candidates, each containing information such as IP addresses, port numbers, and transport protocols. These candidates are instrumental in establishing the most optimal connection between devices, regardless of their network setups.
- ICE candidates use a process called connectivity checks to determine which network path to use. This process involves sending and receiving messages between the two devices to test the connection quality and latency.
- Once the best network path is determined, the devices establish a direct peer-to-peer connection over the Internet. STUN and TURN servers, which we'll discuss later, are often used to facilitate the exchange of ICE candidates and establish connections when direct peer-to-peer communication is not possible.

SDP Protocol

Session Description Protocol (SDP) is a text-based protocol used to set up multimedia sessions between devices across the internet. It includes details of the media streams, such as the type of codec, transport protocol, and other related information.

The Role of SDP in WebRTC

SDP plays a critical role in WebRTC by enabling devices to negotiate media formats, transport protocols, and other details required for a successful connection. When two devices attempt to establish a WebRTC connection, they exchange SDP messages to negotiate the details of the media streams they want to send and receive.

How SDP works

When devices are establishing a connection over the internet, they need to let each other know the type of media streams they want to send and receive, the transport protocols that should be used, the type of codec and any additional information needed to establish a connection. All this information is sent and received as an SDP.

If the devices come to an agreement on the connection requirements, a peer-to-peer connection is established and the devices can send and receive media streams in a direct and standardised way. Additionally, this ensures compatibility between devices from different manufacturers.