REAL-TIME SIGNALING AND MEDIA COMMUNICATION PROTOCOLS AT SCALE

http://horatiulazu.ca/blog

Prepared by
Horatiu Stefan Lazu
horatiulazu@gmail.com
May 1, 2020

Table of Contents

Executive Summary

This report discusses and analyzes different protocols and technologies used in real-time communication systems by social media platforms. Real-time communication typically involves both a signaling and media stack. During signaling, actions are performed, such as ringing and hanging up, which often need to be relayed to users in an efficient manner. Similarly, media involves the encoding, relay and encryption of a stream of data, which includes audio and/or video. This report primarily focuses on a high-level overview of the different stacks, along with open-source protocols defined for use in this context. In addition, this report discusses different algorithms that are applied for media streaming, in the context of codecs and the perspective of performance.

The objective of this report it to raise familiarity with the challenges associated with real-time communications, and how different companies tackled problems related to it. In particular, this report focuses on the signaling and media streaming aspects, and not on underlying infrastructure details such as ensuring durability or reliability of hardware.

1.0 Introduction

The objective of real-time communication in the context of social media is to connect people from different avenues of the world together in a lifelike fashion. By allowing peer-to-peer or multiway communication through audio and/or video, users can connect in a personable way despite being geographically disperced. Billions of calls are performed everyday over the internet, including those from popular sites such as WhatsApp which support over 1 billion daily users (Mansoor Iqbal, 2020). During crisis and major world-events (such as COVID-19), real-time communication and social media can truly bring the world together in a safe manner.

Numerous technical challenges arise from designing scalable systems that handle users from different regions. First includes how to initiate the call and inform users that another user wishes to connect. Afterwards, there needs to be a direct connection established to allow for a reliable stream of data to be sent to the recipient(s). Moreover, there are variable network conditions that need to be accounted for such as low bandwith, unreliable connections or firewalls. In addition, there is the consideration of encrypting data sent over the network, and ensuring privacy of users by designing robust systems. This is just a small sample of the set of challenges faced by software engineers, architects and site reliability engineers in designing the world-class systems powering the large-scale social media websites of today.

2.0 Analysis

2.1 Overview of Signaling and Media Streaming

The high-level goal of real-time communication is to connect a party of people together such that they can communicate with low latency. To achieve this, two connections are established: signaling and media. Signaling is the process of exchange information between different parties, typically associated with events or actions. Media is the actual media being sent, which includes audio and/or video.

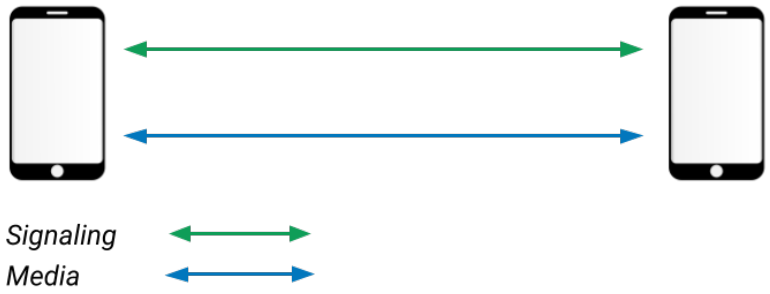

Figure 1: The simplest form of peer-to-peer communication.

Signaling is primarily used for session control, network data, media metadata and key material. Session control involves using messages to open, modify and close communication channels. Network data consists of any information for connecting the endpoints, such as obstructions in access. Media metadata is used to transmit metadata such as media types supported, codecs, along with encryption schemes used. Lastly, key material is used for sending encryption keys used to decode sensitive data sent over the network.

Media streams are used to send the data that users want to experience, such as audio and/or video. Audio and video media is typically encoded using different codecs and can be fine tuned depending on network conditions. Moreover,

audio and video media is often susceptible to packet loss, as performance is more critical than losing individual packets of information. Various techniques have been researched and developed over the years to allow for more effective lossy/loseless compression, improved sound quality (echo cancellation, jitter avoidance, etc.) and remidiation of intermittent packet loss.

2.2 History of Signaling and Media Streaming

Signaling is not a modern concept - in fact, it was first introduced in the 1800's as a major component of telephone systems. Electric pulses and audible tones are used to transmit information such as busy signalings, addressing, dial tones and to request service. The rotary dial in 1896, which was later formalized by 1910, introduced the first design of performing such requests which relayed electrical pulses in current flow ("Development of long-distance transmission", n.d.). Over time, the standards incrementally envolved to take into consideration overseas transmission, newer transmission mediums such as coaxial cabling and applied techniques such as FDM (frequency division multiplexing) to best utilize existing infrastructure.

In contrast to telephone communication, real-time communication used by social media websites typically uses VoIP (Voice over Internet Protocol) versus PSTN (public switched telephone network). While general concepts remain consistent (such as signaling, connection setup, digitization of analog signals via quantization), data is transmitted over a packet-switched network versus circuit-switched network.

The first providers of VoIP mirrored solutions of the legacy telephone systems. Second generation providers including Skype adopted large-scale closed networks providing free calling capability with the ability to tap into PSTN

networks if preferred at cost. The latest generation leverages federated VoIP, which is a form of voice telephony using packets across autonomous domains without requiring switching centers or centralized virtual exchange points. An example would include Google Talk which uses ENUM (E.164 Electronic Number to URI Mapping standard), acting similarly to DNS record types where a telephone number would map to a URI or IP address.

VoIP calling is implemented using a combination of propriertary and open-source protocols. The foundation was formed in 1928 by Homer Dudley at Bells Labs creating the first electronic voice synthesizer (known as Vocoder). Later, ARPANET (Advanced Research Project Agency) built the first packet-switched network in 1969. By 1973 the first voice data packet was transmitted by MIT, followingly the first audio codec was approved by 1988 (G.722), and then finally by 1991 Autodesk released the first VoIP application (known as Netfone) to the public domain. Development increased substantially after that point, leading to Free World Dialup in 1994 and the first-for-profit VoIP application VocalTec in 1995 (Robert Pepper, 2014).

The H.323 system specification, instated in 1996, is recommended by the ITU Telecommunication Standardization Sector (ITU-T) as a stack of protocols used to provide audio/video communication on packet-switched networks (Margaret Rouse, n.d.). The standard addresses call signaling, control, multimedia transport and control, along with bandwith control for peer to peer and multiway conferences. As part of the H.323 family of telecommunication protocols includes H.225.0 RAS (Registration, Admission, Status), H.225.0 Calling Signaling, H.245 control protocol and Real-Time Transport Protocol (RTP) for delivering media.

SIP (Session Initiation Protocol) was originally standardized in 1999, and is

designed to dictate signaling and call setup protocols for IP-based communications (Tien-Thinh Nguyen, Christian Bonnet, 2016). Unlike H.323, which is was standardized by the International Telecommunication Union (ITU), SIP was formalized by the Internet Engineering Task Force (IETF), hence distinguishing its roots in the internet community rather than telecommunications industry. SIP works in conjunction with other protocols including SDP (Session Description Protocol) to negotiate and establish call state.

With respect to streaming media, MPEG-DASH (Dynamic Adaptive Streaming under HTTP, also known simply as DASH) was published in 2012 as the only adaptive bit-rate HTTP streaming solution recognized internationally for delivering variable bitrate content. This codec agnostic solution is used by YouTube, Netflix and more. Popular codecs used today for encoding video streams in real-time communication includes H.264 (MPEG-4 AVC) introduced in 2003 constituting the majority of traffic, and H.265 (MPEG-H HEVC) introduced in 2013.

2.3 ITU-T H.323 Protocols

H.323 is a standard that defines a series of protocols to provide audio/video communication sessions across packet-switched networks. The standard, created in 1996 by ITU-T (ITU Telecommunication Standardization Sector) addresses call control and signaling, multimedia control and transport, along with bandwith control for both peer-to-peer and multiway conferences (Margaret Rouse, n.d.). While originally created for use in LAN networks, it was quickly adopted for a variety of IP networks including WANs and the greater internet. The standard has since been iterated on, last updated in 2009, while remaining entirely backwards compatible since its first version. It resides on

TCP port 1720.

Firstly, several definitions will be addressed.

Definition. 2.1. Terminal The fundamental element in a H.323 system representing a device users would typically use (such as a phone).

Definition. 2.2. Multipoint Control Unit (MCU) Device responsible to acting as a conference bridge and allows mixing both video/audio.

Definition. 2.3. Gateway Device responsible for enabling communication between H.323 networks and other network types (such as PSTN). An example of using gateways includes allowing enterprise IP phones to communicate with external users via PSTN.

Definition. 2.4. Gatekeeper An optional component across the H.323 network which provides a series of services to terminals and gateways. Examples include endpoint registration, admission control, and user authentication.

Gatekeepers use one of two signaling modes: direct and gatekeeper routed. When direct, endpoints use RAS protocol to learn the endpoint of the remote endpoint and a call is established directly with the remote device. In contrast, when using gatekeeper routed the gatekeeper is used as the intermediary and all signaling passes through it.

Definition. 2.5. Zone The set of endpoints registered to a single Gatekeeper in a H.323 system.

After the address of the remote terminal is determined the initiating endpoint will leverage H.225.0 call signaling to establish communication with the remote terminal. Fast-connect procedures are defined in H.323 to establish calls using only 2-3 messages, reducing latency.

Endpoints generally use RAS protocol to communite with a gatekeeper, while gatekeepers also use RAS protocol to connect with other gatekeepers.

A terminal would typically invoke a GRQ (Gatekeeper Request) message for discovering gatekeepers that are willing to accept messages. After, applicable gatekeepers would confirm (GCF) and the terminal node would pick its most desirable gatekeeper.

Upon sending a call initiation signal, the terminal node would perform an admission request (ARQ) to its preferred gatekeeper. In response, the resolves address is returned in the form of an admission confirm message (ACF). Now, the initiating terminal node will send the ARQ to the remote endpoint. The remote endpoint will then likewise send an ARQ and receive an ACF from its preferred gatekeeper. This step is required for ensuring the device is properly authenticated to call the other recipient, and network conditions are capable of sustaining a call.

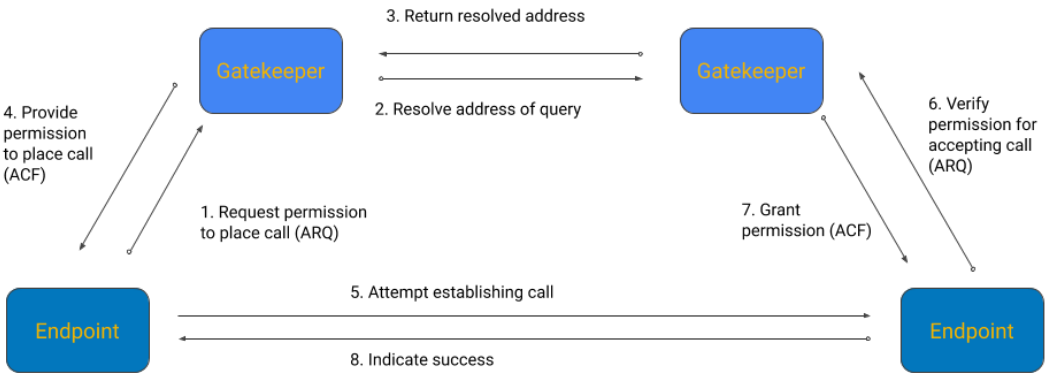

Figure 2: Signaling between two endpoints with gatekeepers using H.225.0.

After the call was initiated, the endpoints may further invoke H.245 call control signaling to provide more control. H.245 is capable of sending information required such as encryption, jitter management and preferences (disabling audio

and/or video). However, one disadvantage of H.245 is the lengthy four-way protocol handshake required when opening logical channels of a conference. This was alleviated using fastStart in a H.225.0 message ("H.323 Fast Start", 2015). Later, H.460.6 introduced Extended Fast Connect Feature, providing a one-way handshake.

2.4 Session Initiation Protocol (SIP)

Session Initiation Protocol, standardized in 1999, is a signaling protocol used for the initiation, maintenance and termination of real-time voice/video applications (Tien-Thinh Nguyen, Christian Bonnet, 2016). The protocol is text-based, and borrows its structure from Simple Mail Transfer Protocol (SMTP) and Hypertext Transfer Protocol (HTTP). SIP is used alongside Session Description Protocol (SDP) as payload containing media metadata, and was designed to be independent of the transport layer. Therefore, SIP can be transfered over Transmission Control Protocol (TCP), User Datagram Protocol (UDP) or Stream Control Transmission Protocol (SCTP). In addition, SIP transmission is accompanied by encryption using Transport Layer Security (TLS). In contrast with H.323, SIP has been standardized by the IETF rather than the ITU, hence has a stronger connection to the internet community than telecommunications.

First, several definitions will be addressed.

Definition. 2.6. User Agent Client (UAC) User Agent capable of receiving responses and sending requests.

Definition. 2.7. User Agent Server (UAS) User Agent capable of receiving requests and sending responses.

Definition. 2.8. User Agent Network endpoint that receives SIP messages

and is capable of managing SIP sessions. Unlike other protocols, SIP requires all user agents to implement both client and server roles, which includes ability to handle both requests and responses.

Definition. 2.9. Proxy Server Network endpoint that receives SIP messages and is capable of managing SIP sessions. Unlike other protocols, SIP requires all user agents to implement both client and server roles, which includes ability to handle both requests and responses.

Definition. 2.10. Registrar SIP endpoint that accepts REGISTER requests, and records the address from the requesting user agent. Registrars are often used to locate other user agents.

SIP can be used for both peer-to-peer and multiway conferences. Typically, both user agents will register with the registrar by providing REGISTER requests along with any authentication. Afterwards, the UAC will signal requests (such as an INVITE) to the UAS. A selection of common requests/responses is included in Table 1 and 2.

Table 1: Session Initiation Protocol Requests

Request Name	Description
REGISTER	Registers the caller's URI to the registar, used by "To" field
ACK	Confirms user agent receiving final response to INVITE
BYE	Signal that the call should be terminated
INVITE	Initiate dialog from UAC to UAS to begin call
CANCEL	Cancel any pending requests (such as pre-existing rings)
REFER	Ask recipient to takeover existing call (and issue request)
INFO	Send any conference metadata while not modifying session state
MESSAGE	Deliver a text message, typically used by IM clients
PUBLISH	Deliver an event to a notification server
OPTIONS	Retrieve the capabilities of an endpoint/user agent

Lastly, an extension of SIP is the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE), which can be used

Table 2: Session Initiation Protocol Responses

Reponse Code	Description
1xx	Request was valid and is being processed (asynchronously)
2xx	Successful completion (for INVITE, it indicates an accept)
3xx	Call redirection required (INVITE to another user agent)
4xx	Invalid request (potentialy due to incorrect request syntax)
5xx	Server failure (potentially internal errors)
6xx	Global failure (not limited to one server, can be invalid destination)

for features such as the "..." typing indicator, and adds more specialized request/response headers for instant messaging. The first line of requests containst the method along with the request URI. On the other hand, the first line of responses is the response code.

2.5 Signal Protocol (formerly TextSecure Protocol)

The Signal Protocol, started in 2013 by Open Whisper Systems, is a non-federated cryptographic protocol allowing for end-to-end encryption of voice/video calls and instant messaging conversations. Originally it was used in the popular messaging app Signal, but later was adopted by other apps including WhatsApp and Messenger/Skype in their optional private conversation modes.

By implementing end-to-end encryption, the contents of messages and calling signals/media are only available to the sender and recipient(s), and not the intermittent servers and devices. First, a set of long-term identity key pairs, medium-term signed prekey pair and several ephemeral prekey pairs are generated locally and stored securely. After, all the public keys and registration ids are stored in a "key bundle" and registered with the Key Distribution Center. For the sender to send messages, the sender has to be able to get the registration ID and public keys from the receiver, which can be found in the publically accessible key bundle.

To start a session, the sender uses their identity and medium-term private keys along with the recipient's set of private keys to determine the master shared secret. The recipient can then receive the master shared secret, decipher and validate it. Afterwards, the two users can send messages to each other.

While the session is alive, the sender uses the master shared key and the recipient's ephemeral keys to create a root key, chain key and message chain. This results in a new set of single-use epehemeral keys to encrypt/decrypt future messages. In total, the protocol combines the Double Ratchet algorithm, prekeys, an extended triple Elliptic-curve Diffie-Hellman (X3DH) key agreement protocol, along with using Curve25519, AES-256 and HMAC-SHA256 as cryptographic primitives (Ksenia Kozhukhovskaya, 2017).

X3DH is used for generating all the required keys for the sender/receiver to communicate, including the shared secret key between the two callers, by registering identity and prekeys to a server. Using this architecture, the caller can retrieve the "prekey bundle" of the callee even when the callee is offline.

Double Ratchet Algorithm is used to provide end-to-end encryption based on the shared secret key retrieved by X3DH. From the shared secret key produced, a "root key" and "sending chain key" are generated. Using a key derivation chain (KDF) from the sending chain key, each subsequent message has a different epehemeral key by advancing down the chain, and it makes it impossible to decrypt out-of-order messages. In short, when encrypting messages, the sender always forwards the sending chain by one, and generates a new sending chain key and messaging encryption key.

Finally, AES-256 and HMAC-SHA256 are both 256-bit length encryption functions/block ciphers to protect and encrypt sensitive data. Using the master private keys that are shared across caller/callee, the schemes are used to ensure

that the data is essentially non-recoverable without the proper credentials.

2.6 WebRTC Framework

WebRTC (Web Real-Time Communication) is an open-source project providing mobile applications and web-browsers with RTC capabilities via simple APIs. The mission of WebRTC is to "enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices, and allow them all to communicate via a common set of protocols".

First, several definitions will be addressed.

Definition. 2.11. Session Description Protocol (SDP) Format used in describing and negotiating a session's profile (which includes properties and parameters such as media types and encryption keys). SDP can use attributes extending the protocol's capabilitiesas key/value pairs, and is otherwise a text-based format with one field per line.

Definition. 2.12. NAT (Network Address Translation) Process for translating local IP addresses to public IP addresses, often used for security reasons and conserves legally registered IP addresses (due to limitations with IPv4 bit count).

NATs typically work in four ways: full cone (internal address maps to external address and any external host can send requests), address-restricted cone (any host can send to an externally mapped host/port if and only if that host/port previously sent to the host), port-restricted cone (similar to address-restricted cone, except not only did the receiver need to have previously sent to the same host, but also the matching sender port), and lastly symmetric (where a new mapping is used for each request).

Definition. 2.13. STUN (Session Traversal Utilities for NAT) Lightweight and simple service which provides the public IP address of its caller, used to reply back its IP address to the original caller to establish an IP address; typically used in non-symmetric NATs (full-cone, address-restricted, port-restricted).

Definition. 2.14. TURN (Traversal using Relays around NAT) Computationally expensive relay for sending messages/media to a destination, typically used in symmetric NATs where the public IP cannot be discoverable and instead requires a statically defined external service to send information to the requesting caller.

Definition. 2.15. ICE (Interactive Connectivity Establishment) Standard used for performing NAT (Network Address Translation) traversals. ICE deals with returning candidate agent addresses (local, reflexive such as STUN and relayed such as TURN).

Definition. 2.16. Trickle ICE Optimization of ICE specification providing parallelized connectivity checks across the candidate addresses, reducing the overall initiation time.

There are several core components of the WebRTC framework that is exposed through the JavaScript APIs. Some examples include "getUserMedia" that is used to acquire the audio/video media. In addition, "RTCPeerConnection" is used for performing signal processing, security, peer-to-peer communication and bandwith management. Another core capability is the "RTCDataChannel" for allowing bidirectional communication among peers for sending media using a system based on WebSockets (MDN Contributors, 2020).

The first task in establishing a WebRTC connection is to use a signaling server for resolving the connection. The role of the signaling server is to act as an

intermediary and allow the peers to establish a connection with minimizing private information exposure. WebRTC does not mandate any specific transport mechanism. Instead, WebRTC leverages SDP payloads and ICE to determine candidates for establishing the connection.

For exchanging media WebRTC uses SDP to execute the offer and answer mechanism across peers. Unfortunately, firewalls and NATs are often used in the real-world to both protect private IPs and limit the number of registered IP addreses due to limited IPv4 addressing. However, NAT mappings would occur at the network layer, and hence will change the TCP/UDP packet headers but leave SDP payloads unchanged hence leaving them unaware of handling external NAT IP addresses and port restrictions.

Figure 3: Example of simple signaling and media transfer without NATs.

Depending on the type of NAT (full cone, address-restricted/port-restricted cone or symmetric), a different configuration of services is required. STUN is used in the case of asymmetric NATs, where it is simply invoked to determine the public IP address and then that can be used for media transfer. This is a lightweight solution and is common in practice. Unfortunately, for symmetric cones a relay solution is required where TURN servers will be used as the intermediary endpoints. The disadvantage of this approach is that TURN servers are expensive, add additional latency and will be another source of failure among the media streaming path.

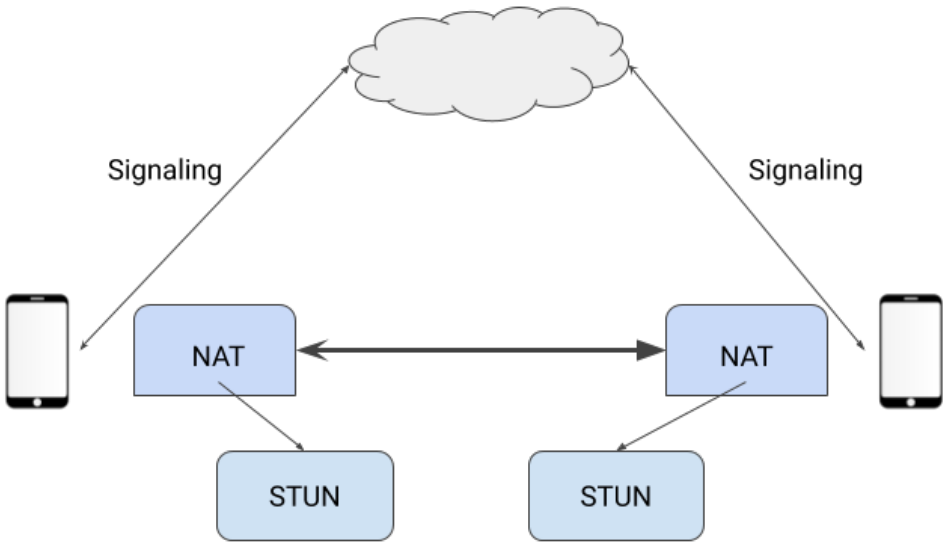

Figure 4: Example of signaling and media transfer between coned/reflexive NATs.

Popular users of the WebRTC users includes Facebook with over 300 million monthly active users, Discord and Skype (Chad Hart, 2017).

2.7 Real-Time Streaming Protocol (RTSP)

Real-Time Streaming Protocol (RTSP), standardized in 1998 by the IETF, is a network control protocol for controlling streaming media servers (Ivn Santos-Gonzlez, Alexandra Rivero-Garca, Jezabel Molina-Gil, Pino Caballero-Gil, 2017). RTSP is not responsible for the streamed data itself, which is often handled by the Real-time Transport Protocol (RTP). Instead, it facilitates capabilities such as play and pause. The protocol is similar to HTTP, with several exceptions such as RTSP being stateful and resides on both TCP/UDP ports 554 (although UDP is rarely ever used).

RTSP supports several types of requests, including several overlapping with HTTP. “OPTIONS”, similar to HTTP, returns the accepted request types by

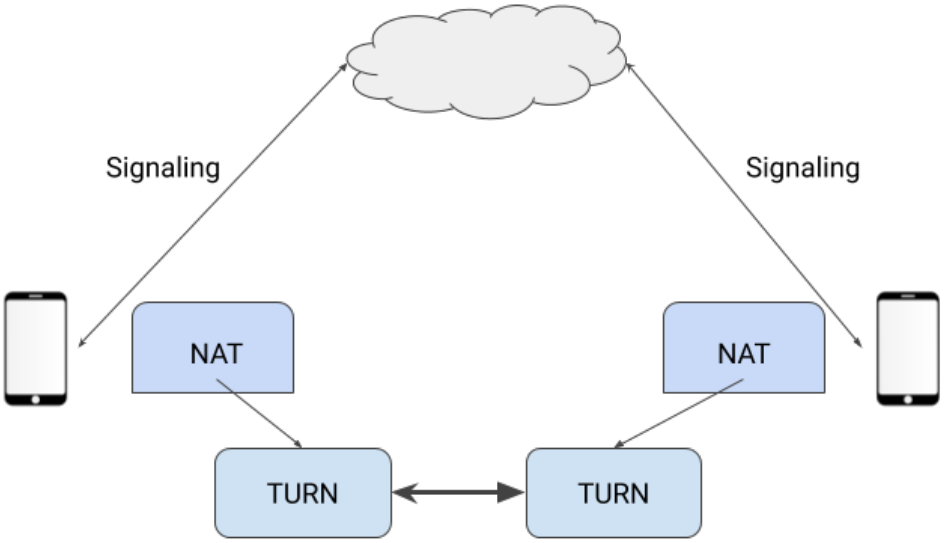

Figure 5: Example of signaling and media transfer between symmetric NATs.

the server. "DESCRIBE" requests are used for obtaining SDP presentation descriptions, which contain metadta such as the list of media streams that are controlled by the aggregate URL, bitrate, mime type, etc. Lastly, there are "PLAY", "PAUSE" and "TEARDOWN" requests.

Popular client implementations of this protocol include VLC media player, Skype, Spotify, QuickTime and Windows Media Player. In addition, YouTube uses RTSP as an available streaming option when the mobile HTTPS site is viewed on desktop.

2.8 Real-Time Control Protocol (RTCP)

Real-Time Control Protocol (RTCP) is often used in conjunction with RTP to provide out-of-band (signaling running on a dedicated channel away from media channels in PTSN) control information and statistics for an RTP session. RTCP provides feedback on the quality of service including round-

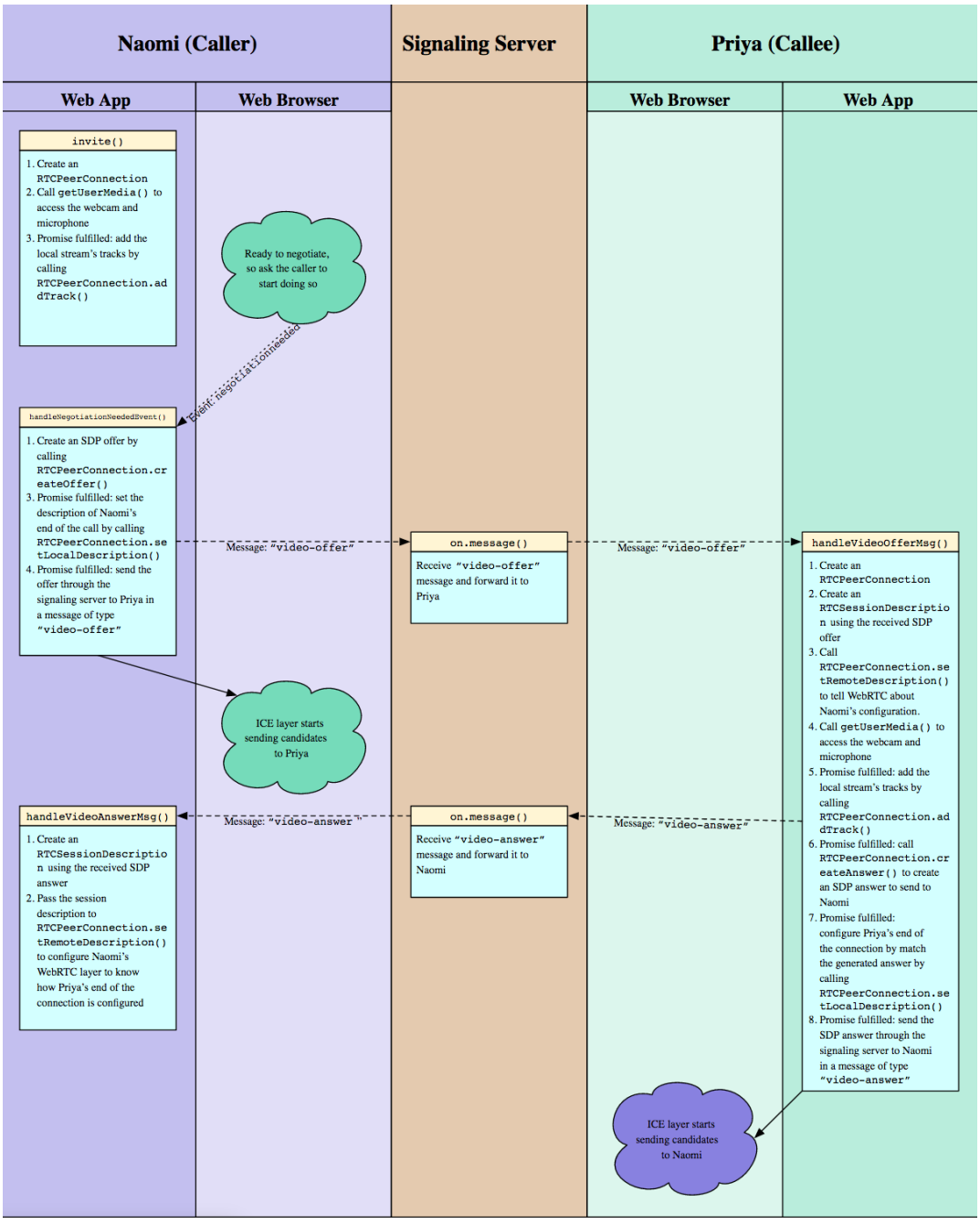

Figure 6: Example of signaling between two peers in WebRTC.

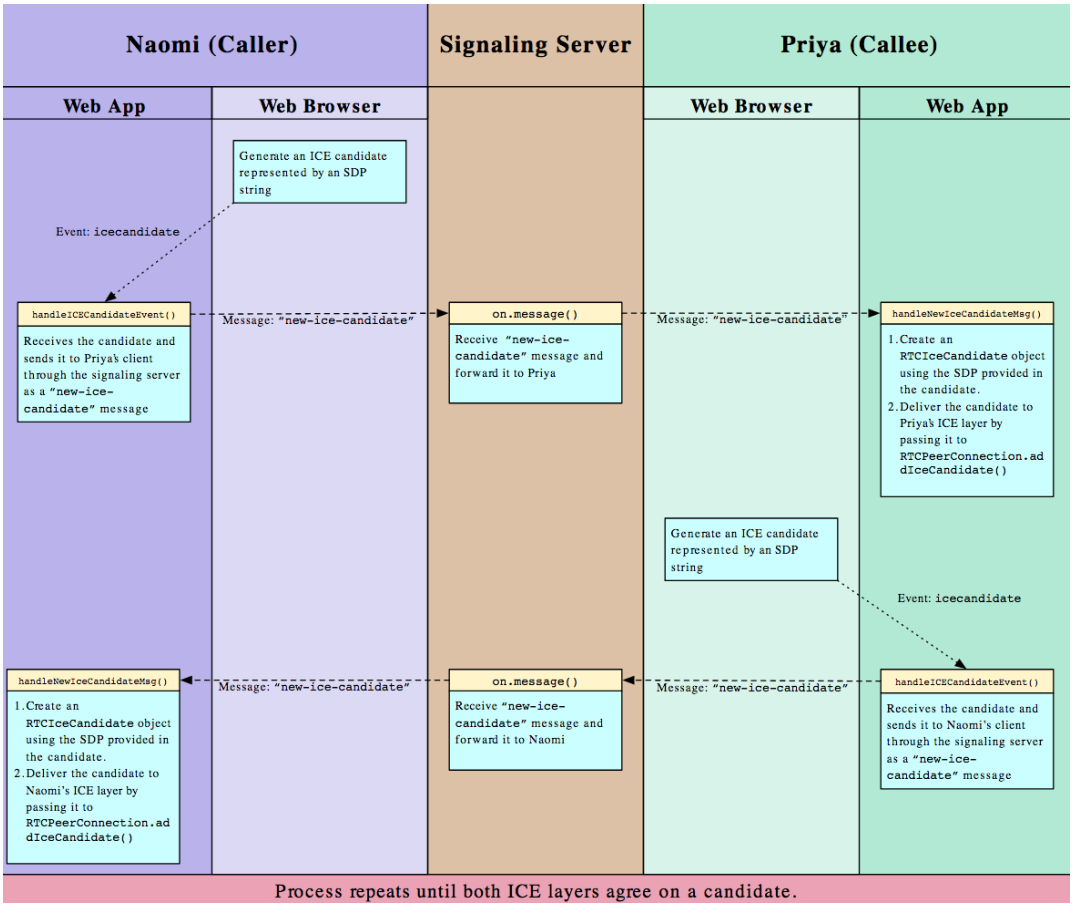

Figure 7: Example of ICE negotiation in WebRTC.

Table 3: Browsers with WebRTC Support

Web-browser Name	Supported Versions
Microsoft Edge	12+
Google Chrome (Desktop)	28+
Mozilla Firefox	22+
Opera	18+
Safari	11+
Google Chrome (Android)	28+
Mozilla Firefox (Android)	24+
Opera (Mobile)	12+
iOS MobileSafari/WebKit	iOS11+

trip delay time, packet delay variation, packet loss and packet counts. This allows for adaptive capabilities such as Dynamic Adaptive Streaming over HTTP (DASH) to be implemented ("RTP Control Protocol (RTCP)", 2017). RTP is generally executed on an even-numbered UDP port, and RTCP on the next odd-numbered port. While RTCP does not provide any encryption/authentication methods, Secure Real-time Transport Protocol (SRTP) provides such capabilities.

RTCP provides three functionalities. First, RTCP provides statistics on quality of service during media distribution. This can be used, as mentioned earlier, for the implementation of DASH alorithms and paging Engineers to look into potential network disruptions. Secondly, RTCP provides Canonical Name Records (CNAME) to conference participants and allows for effective third-party monitoring. Lastly, RTCP is often used for reaching all conference participants, as RTP only transmits via media source. RTCP reporting is applied to all participants of a conference, and is randomized in reporting time to avoid unintended synchronization of reporting (typically intervals around five seconds are used). As a best practice to avoid network congestion, RTCP bandwith usage should not be higher than 5% of overall session bandwith to prevent causing degraded performance.

Several types of messages are supported by RTCP, and can be extended further to include custom packets. Sender report (SR) is periodically sent by active senders in conferences to report reception and transmission statistics. Receiver report (RR) is for receivers of RTP packets, and likewise sends quality reports to senders. Source description (SDES) provides CNAMEs to session participants for use in third-party monitoring solutions. Goodbye (BYE) is used to shut down a tream of data.

2.9 Real-Time Transport Protocol (RTP)

Real-Time Transport Protocol (RTP), standardized by the IETF in 1996, is a networking protocol for audio and video delivery over IP networks. It is often used in conjunction with SIP and RTCP to create and ensure quality transmission of media. RTP is designed for real-time transfer of media streams and provides capabilities such as detection of packet loss, out-of-order delivery, jitter compensation and IP multicast support. RTP is based on application-layer framing, where the protocol functions are implemented in the application rather than operating system.

RTP supports a range of multimedia formats and is extensible by design. By providing profiles and one or more payload formats for each class of application (such as audio or video), RTP supports definining mapping codecs to payload format codes inside profiles, which then describes the transport method for the encoded data. For example, Secure Real-Time Transport Protocol (SRTP) is an RTP profile for providing cryptographic services for transferring payload data. RTP senders would capture the media, encode it, then transmits it as an RTP packet with the appropriate timestamps and increasing sequence numbers. RTP receivers then detect missing packets and may reorder packets.

Afterwards, the stream can be decoded and presented to the user.

Packet headers for RTP support various mandatory fields, as shown in the figure below. Version is used to indicate the current version of the protocol used (2 bits, where the current version is 10_2). P (Padding, 1 bit) may indicate extra padded bytes at the end of a header, typically used due to size requirements imposed by encryption algorithms. X (Extension, 1 bit) is used to indicate presence of an extension header between header and the payload. CC (CSRC count, 4 bits) indicates the number of CSRC identifiers fllowing SSRC. Sequence numbers are incremented for each RTP data packet sent, used to implement out-of-order delivery and detect lost packets. SSRC (synchronization source identifier, 32 bits) is used to identify the source of the stream, while CSRC (contributing source ids, 32 bits each) are used to enumerate contributing sources to the stream. Lastly, the optional header extension would include a 16 bit profile-specific identifier and a 16 bit length specifier, with the custom extension header data following.

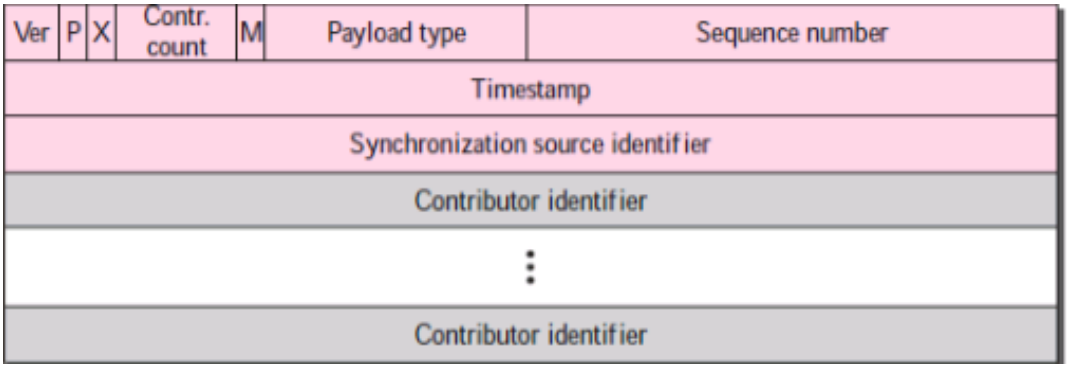

Figure 8: Example of an RTP packet header.

2.10 Audio Streaming Codecs

The process of encoding and decoding audio data is handled by an audio codec (**co**der/**dec**oder). In real-time communication, the balance between having high-quality audio and low-bandwith requirements is a power struggle that often requires compromises from either side. The two general categories of factors affecting encoded audio is the codec chosen and the details about the source audio's contents and format.

Different codecs provide a variety of parameters to tune allowing for variable bit rate, audio frequency bandwith and many other custom fields. Audio codecs are based on advanced numerical computational algorithms and can provide either lossless or lossy compression. AAC (Advanced Audio Coding) is a common codec defined by the MPEG-4 (H.264) standard, used by Blue-ray disks, HDTV, and songs purchased from iTunes (MDN contributors, 2019). However, the format is protected by numerous patents making distribution less predictable. On the other hand, G.722 codec is built with voice compression in mind, is low-latency and uses Adaptive Differential Pulse Code Modulation (ADPCM) to reduce recording size. G.722 is mandated by the WebRTC specification and is typically used for WebRTC connections. Lastly, MP3 (MPEG-1 Audio Layer III) is one of the most common codecs, where MPEG-1 MP3 is generally used for music and MPEG-2 MP3 is used for simpler sounds and requires less space. Moreover, MP3 patents expired as of 2017 in the United States making it supported by all popular web-browsers.

The WebRTC API does not mandate if a particular can be used in a track, but instead requires support VP8 and H.264's constrained baseline profiles for video and G.711 PCM (A-law) and G.711 PCM (u-Law) audio codecs for Chrome, Firefox and Safari browsers.

Table 4: Audio Encoder Configuraion Effects

Feature Name	Size Effects	Quality Effects
Lossless Compression	Less than 40-50% compression	None
Lossy Compression	Up to 80-95% compression	Always some
Bit Rate	Positively correlated	Positively correlated
Audio Frequency Bandwith	Positively correlated	Positively correlated
Stereo Coding	Joint stero may reduce size	None

Table 5: Audio Encoder Source Format Effects

Feature Name	Size Effects	Quality Effects
Channel Count	Positively correlated	Higher directionality perception
Sample Rate	Positively correlated	Positively correlated
Sample Size	Positively correlated	Positively correlated
Noises	Larger due to increased complexity	Lower

2.11 Video Streaming Codecs

Video encoding is the process of turning raw video into a digital format to be viewed on different devices. Similar to audio codecs, video codecs vary in behavior and specialization depending on use-case, and are often either lossless or lossy in nature. Encoded videos are wrapped into a "video container" (such a .mp4, mov), which contains the video codec, audio codec and associated metadata. The key difference between a container format and a codec is that the codec is at the source and playback to compress and decompress respectively, while the container can be used to determine which programs accept the stream and holds the different components (audio, video, closed captioning) together.

The most common encoding for streaming is MPEG-4 H.264/AVC (Advanced Video Coding), developed by the International Telecommunications Union (ITU) and International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Experts Group. H.264/AVC

has capabilities of being packaged into a variety of container types including .mov, .mp4, .3GP and more.

MPEG-5 H.265/HVEC is a newer codec, which has improved compression efficiency and supports 8k resolution. However, the market-share is only around 10% due to royalty complications (Traci Ruether, 2019). To address this, AV1 was created in a partnership with several leading tech companies (Google, Microsoft, Amazon, Netflix, and more) to create a performant royalty-free codec. Google created the VP9 codec as a royalty-free and more performant version of HVEC, however it is not supported on Apple devices and is often considered to be "AV0".

2.12 Dynamic Adaptive Streaming over HTTP (DASH)

Dynamic Adaptive Streaming over HTTP (MPEG-DASH, otherwise known simply as DASH) was standardized in 2012 to provide adaptive bitrate streaming for media over HTTP web servers. DASH works by taking content and breaking it down into smaller HTTP-based file segments, where each segment contains a small interval of content. Each individual segment is then broken down into different bit rates. While the media is streamed, the client applies a bit rate adaption (ABR) algorithm that selects the appropriate segment such that the content can be downloaded in time without causing rebuffering/stalls and maximizes quality.

DASH uses TCP as the chosen transport protocol, and remains agnostic to codecs (meaning it supports various formats including H.264, H.265, VP9), ABR logic and the underlying application layer protocol. Alongside the individual segments containing intervals of streamed data, DASH uses a media presentation description (MPD) to describe segment information (URL, tim-

ing, bit rates, etc.), and is presented using a variety of data-structures such as timelines or lists. While there is no restrictions on the type of media data, the specification provides recommendations on using two types of containers: MPEG-2 Transport stream or an ISO base media file format (such as MP4).

Adoption is widespread for DASH, including support by YouTube and Netflix and VLC. While not directly supported in HTML5, there are open-source JavaScript implementations for DASH adding the functionality. Moreover, when combined with WebGL, HTML5-based DASH allows for streaming of 360 degrees content. Alongside developer and product support, content distribution networks (CDN) support for DASH is vast, including Akamai, Amazon CloudFront, CloudFlare and Azure Media Services.

HTTP Live Streaming (HLS), released in 2009, is an alternative HTTP-based adaptive bitrate streaming protocol developed by Apple (Max Wilbert, 2020). Similar to DASH, HLS breaks the stream into a sequence of smaller HTTP-based file downloads in a variety of different bitrates. After, the list of available streams encoded at different bitrates is sent as an extended M3U playlist to the client. Unlike RTP, HLS allows for traversing firewalls and NATs that would otherwise allow regular HTTP traffic to travel through. Later versions of the protocol also introduced subtitle support.

The architecture of HLS is comprised of three parts: server, distributor and client. The server is responsible for preparing the video for delivery, encoding encoding the video files in H.264 format in variable bitrates as MP3, AAC, AC-3 or EC-3, and then encapsulated in MPEG-2 Transport Stream. After, the segmenter will divide the MPEG-2 TS file into equally sized segments, and then create the index file for fragmented files as an m3u8 file. The distributor will then act as a regular HTTP web server, and deliver the required m3u8

playlist file and ts segment files required to stream the content. Lastly, the client will retrieve the m3u8 file containing the index of segments, and then retrieve the necessary segments from the distributor.

2.13 Multiway Calling Architectures

WebRTC media streaming natively supports communication across two different peers, but often real-time communication applications support multiway calling involving numerous peers. Multiway conferences for voice and video can be supported using three architectures: mesh, mixing and routing (Tsahi Levent-Levi, 2019). In addition, signaling often uses an intermediary centralized server acting as an anonymous "peer", and relays signaling using ICE candidates (such as STUN or TURN).

Mesh routing involves an n^2 amount of linkages, where n is the number of peers, which is not scalable to many users and requires substantial amounts of bandwith. Mixing leverages an MCU (Multipoint Conferencing Unit) which acts as a centralized point where multiple streams from individual peers are combined into a single unified stream. However, despite the simplicity in design, it comes without flexibility such as client-specific processing because the streams are combined at the relay source.

Routing leverages an SFU (Selective Forwarding Unit), which instead acts as the router of the media (Tsahi Levent-Levi, 2019). In contrast to mixing, routing will send individual streams directly to other peers allowing for more client-sided flexibility in terms of processing. Three approaches are applied for routing: multi-unicast, simulcast and SVC. Multi-unicast is the trivial approach, where users would send streams to the SFU and the SFU would decide where to route, not performing any bit rate adaption. On the other

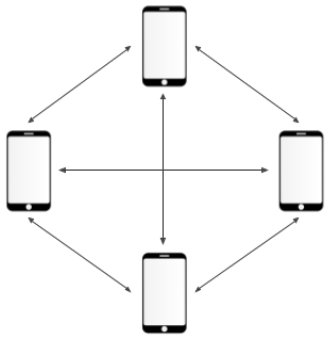

Figure 9: Example of multiway mesh streams.

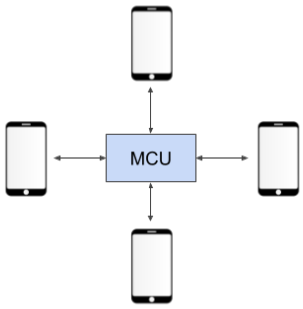

Figure 10: Example of multiway mixing streams.

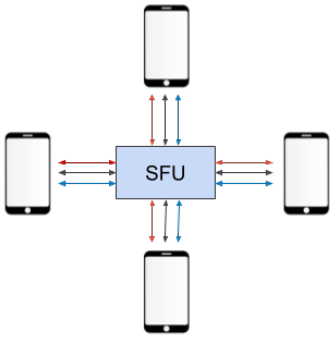

Figure 11: Example of multiway routing streams.

hand, simulcast will accept multiple streams of varying qualities and then will send the supported ones (based on network conditions and device capabilities) to the other peers. Lastly, SVC (scalable video coding) uses a similar methodology as simulcast, but instead of sending disjoint streams of varying bitrates, SVC sends a layered stream of increasing quality where particular layers can be "peeled" off the top to reduce quality. This is an enhancement over simulcast because it reduces computation time and allows error corrections to occur only at base levels; it was introduced to WebRTC in the VP9 video codec.

3.0 Case Studies

3.1 Skype's P2P Signaling Protocol (2003)

Skype is a VoIP client developed by Microsoft allowing users to place audio and video calls over the internet. Skype uses an overlayed peer-to-peer network, similar to its file sharing predecessor Kazaa (Salman A. Baset, Henning G. Schulzrinne, 2004). There are two types of nodes in the architecture: ordinary hosts and super nodes (SN). An ordinary node is any Skype client that is used for issuing voice and video calls. Super nodes are also Skype clients, however they are promoted to SN once it is identified that they have a public IP address, performant hardware (RAM, CPU) and network bandwith. In addition, the login server is a critical piece of Skype infrastructure as it handles the login credentials of users and in later Skype versions the friends lists of users. Moreover, SkypeOut and SkypeIn are servers used to bridge VoIP with PSTN, however it is not used in pure VoIP calls.

Every Skype node uses a variant of the STUN/TURN protocol to identify NAT and firewall restrictions of users. When the SC is loading, it first sends

an HTTP request to determine if there should be a software update. After, a connection to a SN is established. A critical component of Skype clients is the host cache (HC), which is built and refreshed regularly to contain a list of SN capped at a length of 200. If none of the SN inside the HC are reachable, Skype resorts to establishing a TCP connection to a bootstrapped list of 8 hardcoded SN addresses - if that does not work either then Skype fails to login. After SC is connect to a SN, the client applies the username/password to authenticate with the Skype login server.

Skype user search leverages a Global Index (GI) technology (Salman A. Baset, Henning G. Schulzrinne, 2004). First, the SC will ask the SN over UDP if it knows any users matching the regular expression. If an SN does not know, then it will provide 8 addresses of SN nodes over TCP to the SC to further query. This process repeats with 16, 32, and exponentially many more nodes. After an abritrary cutoff, if none are found, the login server is requested (which is always invoked in the case of no matching usernames). Repeated queries are cached locally at the client.

The Skype protocol has no silence suppression, meaning it still sends UDP packets containing null noise even when muted. There are two benefits to this design choice, namely preventing reapplying UDP bindings and in the case of TCP being used it prevents drops in the TCP congestion window which would reduce preliminary RTT time until it ramps back up. The codec of choice used by Skype is iCodec, with a minimum bandwith requirement of 2Kb/second (Salman A. Baset, Henning G. Schulzrinne, 2004). Lastly, it was observed that in conferences, the media is not fully meshed, meaning the most powerful machine is elected to be the host and collect and later distribute the streams to the remaining hosts, as show in the figure below.

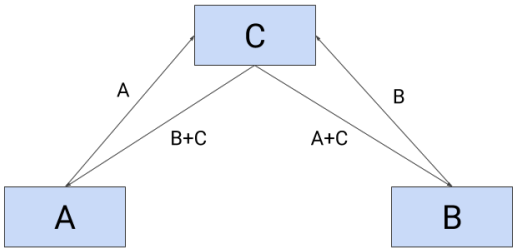

Figure 12: Example conference signaling without full mesh.

4.0 Conclusions

4.1 Resulting architecture of RTC systems

This report provides a high-level overview of different protcols, standards and codecs used to implement scalable real-time communication services such as video calling. Various protocols and technologies are involved in the signaling, encoding, and delivery of media over networks. Over time, numerous optimizations have been applied to reduce the number of handshakes required to establish connections, save bandwith and improve call quality. By applying the techniques applied in this report, it becomes possible to get a high-level view of designing a large-scale system supporting numerous concurrent users across ranging network setups such as private IP addresses guarded by firewalls. In the future, frameworks such as WebRTC will become more mature, codecs such as H.265 will have widespread adoption and protocols will be added making it easier than ever to keep the world connected and together by using real-time communication.

References

[1] Mansoor Iqbal (2020). WhatsApp Revenue and Usage Statistics. *Business of Apps (2020)*. Retrieved April 12, 2020, from https://www.businessofapps.com/data/whatsapp-statistics/

[2] (n.d.) Development of long-distance transmission. *ENCYCLOPDIA BRITANNICA*. Retrieved April 12, 2020 from https://www.britannica.com/technology/telephone/Transmission

[3] Robert Pepper (2014). The History of VoIP and Internet Telephones *GetVoIP*. Retrieved April 12, 2020, from https://getvoip.com/blog/2014/01/27/history-of-voip-and-internet-telephones/

[4] Jan Ozer (2011). What Is MPEG DASH? *Streaming Media*. Retrieved April 12, 2020, from https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79041

[5] Margaret Rouse (n.d.). H.323 *Search Networking*. Retrieved April 12, 2020, from https://searchnetworking.techtarget.com/definition/H323

[6] Tien-Thinh Nguyen, Christian Bonnet (2016). IP Mobility Management for Future Public Safety Networks *Wireless Public Safety Networks 2*. Retrieved April 12, 2020, from https://www.sciencedirect.com/topics/computer-science/session-initiation-protocol

[7] (2015). H.323 Fast Start. *Dialogic.*. Retrieved April 12, 2020, from https://www.dialogic.com/webhelp/BorderNet2020/1.1.0/WebHelp/h323_faststart.htm

[8] Ksenia Kozhukhovskaya (2017). Demystifying the Signal Protocol for End-to-End Encryption (E2EE) *Cloudboost*. Retrieved April 12, 2020, from https://blog.cloudboost.io/demystifying-the-signal-protocol-for-end-to-end-encryption-e2ee-3e31830c456f

[9] MDN Contributors (2020). WebRTC API. *Mozilla Developer Network*. Retrieved April 12, 2020, from https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

[10] Chad Hart (2017). WebRTC: One of 2016's Biggest Technologies No One Has Heard Of *Web RTC World*. Retrieved April 12, 2020, from http://www.webrtcworld.com/topics/webrtc-world/articles/428444-webrtc-one-2016s-biggest-technologies-no-one-has.htm

[11] Ivn Santos-Gonzlez, Alexandra Rivero-Garca, Jezabel Molina-Gil, Pino Caballero-Gil (2017). Implementation and Analysis of Real-Time Streaming Protocols *US National Library of Medicine*. Retrieved April 12, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5424723/

[12] (2017). RTP Control Protocol (RTCP) *SDX Central*. Retrieved April 12, 2020, from https://www.sdxcentral.com/resources/glossary/rtp-control-protocol-rtcp/

[13] MDN contributors (2019). Web audio codec guide *Mozilla Developer Network*. Retrieved April 12, 2020, from https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_codecs

[14] Traci Ruether (2019). Video Codecs and Encoding: Everything You Should Know (Update) *Wowza Media Streams*. Retrieved April 12, 2020, from https://www.wowza.com/blog/video-codecs-encoding

[15] Max Wilbert (2020). What is HLS streaming and when should you use it? *dacast*. Retrieved April 12, 2020, from https://www.dacast.com/blog/hls-streaming-protocol/

[16] Tsahi Levent-Levi (2019). WebRTC Multiparty Architectures *BlogGeek*. Retrieved April 12, 2020, from https://bloggeek.me/webrtc-multiparty-architectures/

[17] Salman A. Baset and Henning G. Schulzrinne (2004). An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol *Columbia University*. Retrieved April 12, 2020, from http://www1.cs.columbia.edu/ salman/publications/skype1_4.pdf