frozen mountain

Anton Venema
Chief Technology Officer

Anton is a leading expert on Real Time Communications solutions, and the visionary lead architect behind IceLink, WebSync and LiveSwitch.

Audio/Video Mixing

Scaling large conferences on the server

Introduction

While peer connections and forwarded (SFU) connections cover a vast majority of use cases, and are very efficient on the server while they do it, there is a fundamental limitation on the client-side if a large number of the participants in a session are sending audio and/or video. At some point, the number of peer connections or SFU downstream connections involved will exceed what the client can handle, both in terms of physical device processing resources and available downstream bandwidth.

Enter mixing.

Mixing moves the load from the client to the server, commonly referred to as a multi-point control unit, or MCU. By mixing the contents of all participants into a single outbound stream, clients need only create a single connection to the server to hear everyone's audio and see everyone's video. There is a cost, though, since each inbound stream must be decoded into its raw format before it can be mixed and then encoded, all of which are CPU/GPU-intensive operations.

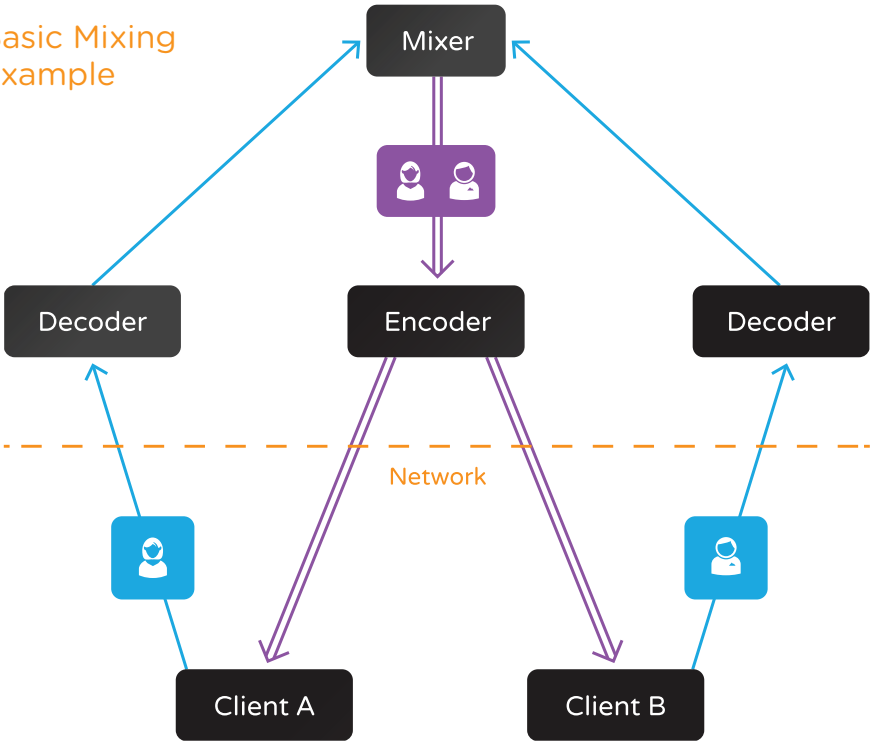
Mixing also allows clients with incompatible codecs to interoperate, as long as the MCU supports at least one of the codecs offered by the various participants and is configured to produce multiple encodings from the mixed output. Again, this benefits comes at the cost of additional CPU/GPU drain.

Despite the cost, mixing is often the best (and often only) option for large-scale conferences where many of the participants are sharing audio and/or video.

Basic Mixing

On the surface, mixing is quite simple. An MCU acts as a termination point for the media stream, turning the incoming stream of RTP packets into a set of encoded frames which are processed by a decoder. Each incoming stream owns its own decoder, which feeds the decoded output into a session-level mixer. The mixer takes the audio and video frames from all the participants as input and produces a sequence of mixed audio and video frames as output. These mixed frames are then fed into one or more encoders to be compressed. There could be just a single encoder, if all the participants support the same codec, or multiple encoders, if there isn't a common codec among the group. In any case, the compressed output from each encoder is then transformed into a stream of RTP packets to be sent out over the network.

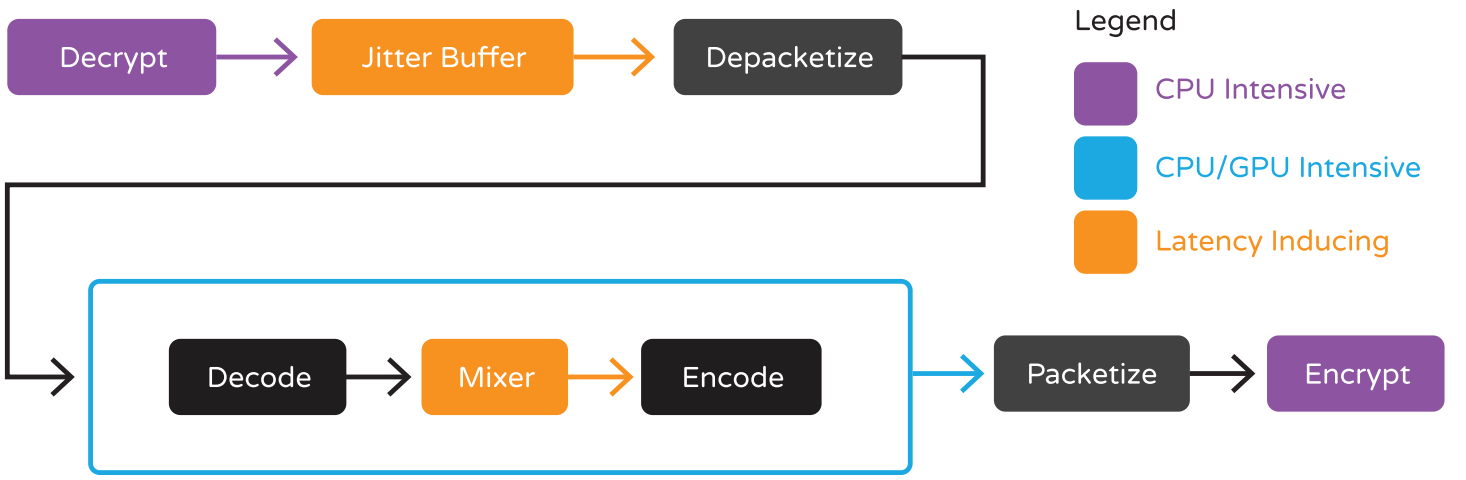
Basic Mixing Example



Mixing is fundamentally higher-latency than forwarding. Unlike an SFU, an MCU must depacketize and decode each inbound stream. Depacketizing, the process of assembling the contents of individual RTP packets into an encoded frame, generally adds the most latency, since it must run every RTP packet through a variable length jitter buffer to account for varying latency on the network path between the sending client and the server. Decoding adds latency simply due to the time spent by the CPU or GPU decompressing the encoded frame into raw media.

The mixer itself adds latency as well, as it must use buffering to synchronize the various inbound participant streams. The mixed output must then be encoded and packetized as well, each of which takes additional CPU/GPU time.

Media Pipeline



Audio Mixing

Audio mixing is performed at the session level by taking all the audio samples from each of the inbound streams and combining them continuously using buffering as needed to handle jitter and synchronization. The mixed audio is then encoded and sent out. If a participant is both sending and receiving audio (a typical use case), then that participant's audio is first removed from the mix before encoding. Otherwise, the client would hear their own audio echoed back in the mix, which is very undesirable.

Video Mixing

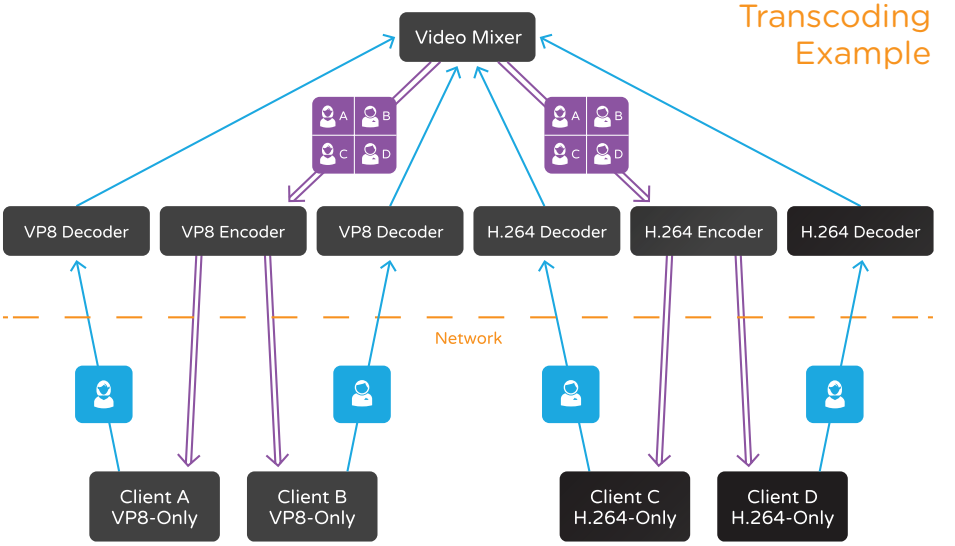
Video mixing is also performed at the session level. Each inbound stream decoder writes the decoded image to a shared canvas that is in turn, encoded on a regular interval (frames per second) and then sent out.

Because video processing is so much more CPU/GPU-intensive than audio processing, and because the "echo" effect isn't there, a unique video stream is not sent to each participant. This means that each participant that sends and receives video will see their own video feed in the mix (with the extra latency that comes with going to and from the server). To avoid this distraction, client code can position the video preview precisely such that it completely covers that portion of the video feed. So long as the MCU signals changes to the mixed video size and layout, the client can be assured of a fantastic user experience.

Transcoding

An MCU is, essentially, a transcoder, or at least capable of being one.

Consider the VP8 and H.264 video codecs. While both VP8 and H.264 are "mandatory-to-implement" for browsers that claim WebRTC compatibility, as of this writing, Apple has opted to disable VP8 in WebKit for Safari. Add to this the fact that Chrome on Android only enables H.264 if the device supports hardware acceleration, and you have a fundamentally incompatible pair of peers. Whereas a peer or forwarded connection would fail, an MCU allows the peers to talk to each other since it knows how to "speak" both VP8 and H.264.

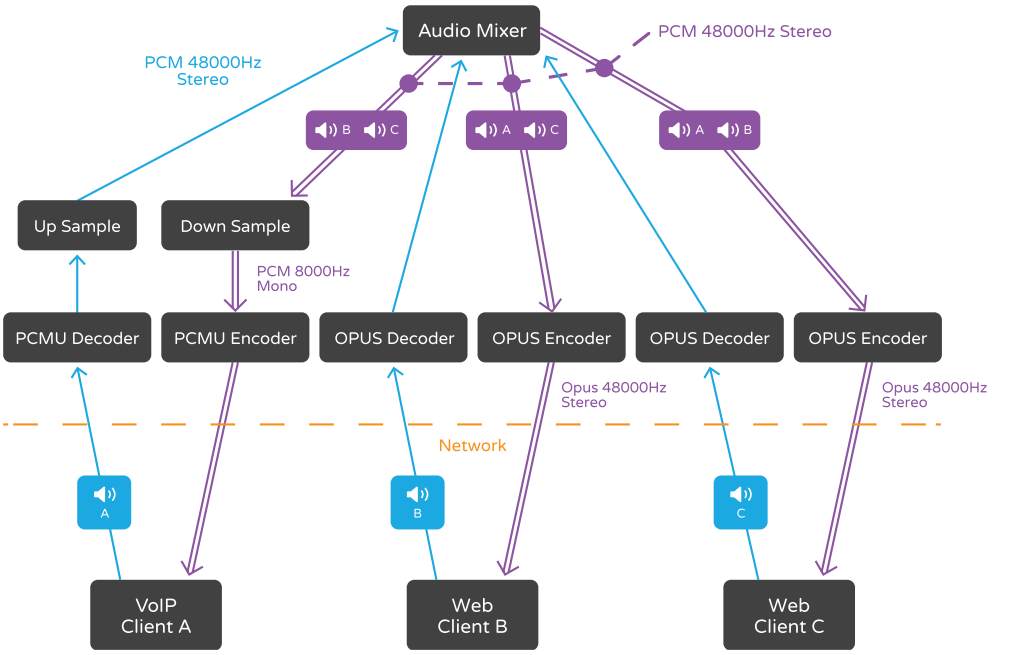


As you might expect, the landscape gets progressively more complicated as you add more platforms and server integrations. Instead of working with each individual vendor to add support for a new codec, an MCU (like the one included with LiveSwitch) allows you to simply add support once and be done. As new codecs are popularized or standardized, they are added to LiveSwitch to make interoperability a breeze.

VoIP Integration

An MCU is almost always required when VoIP integration is desired. Under very specific conditions, it is possible for a VoIP device to talk directly to something else (another VoIP device or even a WebRTC client) using a peer connection, but it is limited to a single, bidirectional, one-to-one call, and the codecs and encryption standards have to align. For example, while WebRTC standards mandate encryption on all connections, many SIP clients simply don't support encryption at all. These and most other use cases require forwarding, transcoding and/or mixing.

VoIP Integration Example



The SIP connector included with LiveSwitch ensures that both inbound and outbound SIP connections are MCU-based to avoid any potential compatibility issues with WebRTC clients. This allows LiveSwitch-based applications on desktop, mobile, and web to use the latest technology and highest-quality codecs available while allowing any number of legacy VoIP devices to join in.

Mixing Limitations

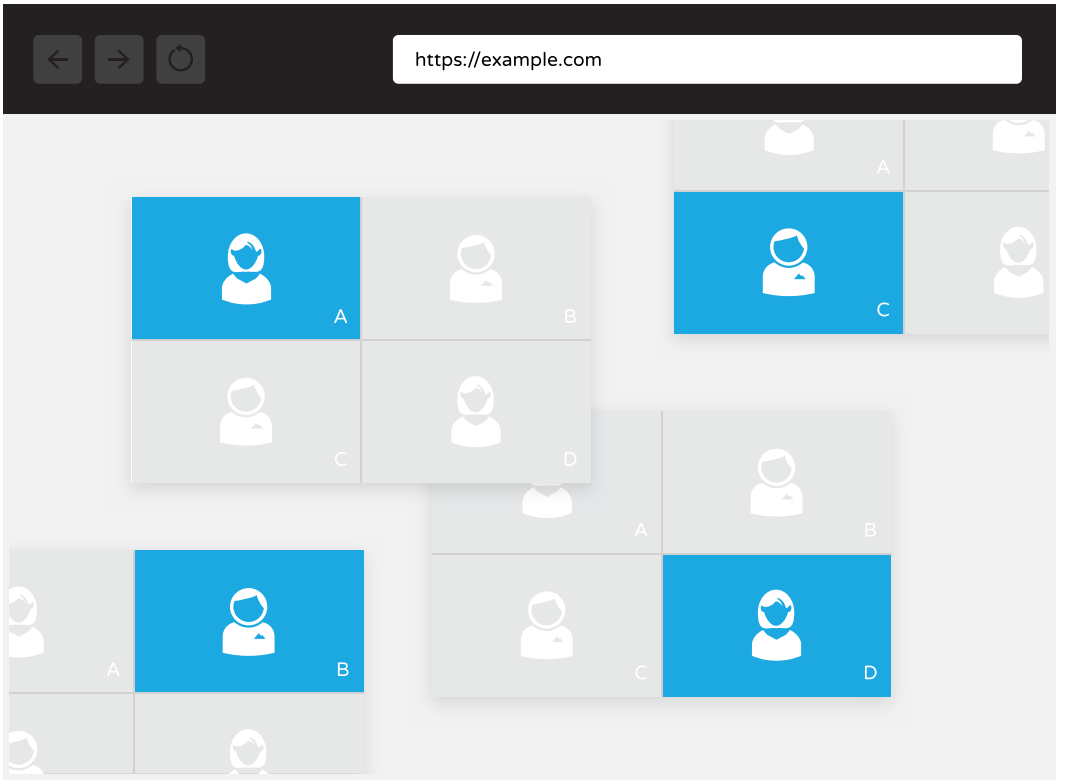
There are two primary limitations of an MCU:

1. Servers bear the burden of decoding, mixing, and encoding multiple streams.
2. Clients have limited flexibility in customizing their particular downstream mix.

The first limitation becomes more obvious as the number of senders increases. The difference in CPU/GPU resources consumed by a conference where each participant uses MCU connections will be noticeably higher than a conference where each participant uses SFU connections. The reverse will be true when looking at client resources. An MCU-based conference will consume far fewer resources on the client when compared to an SFU-based conference.

The second limitation can be alleviated in part through server controls on the audio mix and client-side layout cues. Since the audio mix is customized for each participant, it is also possible to selectively mute/unmute peers on a per-client basis. The video mix is received as a single stream, but it doesn't have to be presented that way. As long as the MCU signals the position and size of each image within the mixed video stream, it's possible for client-side logic to display the video feed multiple times in multiple views, with each view clipping to the boundaries of individual images within the feed.

Client-side Layout Options



Wrap-Up

Mixing is an excellent way to scale audio/video conferences beyond the limits of what peer- and forwarding-based conferences can handle. While the server demands are higher, there is no other approach as broadly compatible and client-side scalable as an MCU-based solution.

If you are interested in learning more about how Frozen Mountain can help you scale your video communications, please don't hesitate to contact us. Our product suite is custom-designed from the ground up to be flexible enough to work in every scenario for every customer, regardless of how unique or constrained your needs are, and yet powerful enough to serve massive customer bases. We look forward to hearing from you!