LiveKit Support for GStreamer WebRTC Elements

Jordan Yelloz
Senior Software Engineer

COLLABORA

Open First

About Me

- Working on GStreamer projects at Collabora since 2022
- Previously worked at Amazon Video and a few much smaller companies
  - Projects ranging from digital print automation, GStreamer, Linux audio drivers, web services
- Based in Fort Collins, Colorado, USA

Agenda

- GStreamer WebRTC Elements Introduction
- LiveKit Server Introduction
- Basics: Source/Sink Elements
- Extra: Video Simulcast
- Extra: End-to-End Encryption
- Development Challenges

COLLABORA

GStreamer WebRTC Elements
Introduction

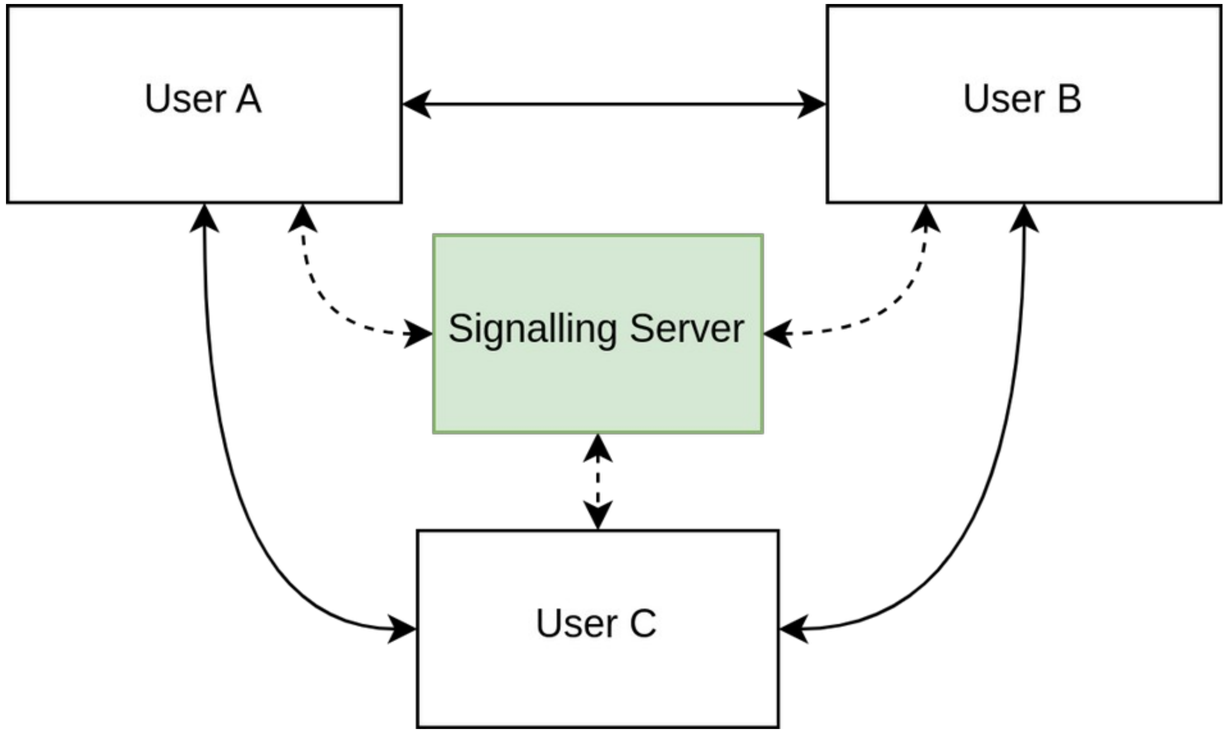
WebRTC Elements - Intro

- Source and Sink elements in gst-plugins-rs
- Single Function: Either Sender or Receiver
- Reduced user-facing complexity compared to webrtcbin
  - Can get started with `gst-launch`
- Extensible to multiple WebRTC systems, currently:
  - Built-in Reference Server
  - WHIP/WHEP
  - Janus
  - AWS KVS
  - LiveKit

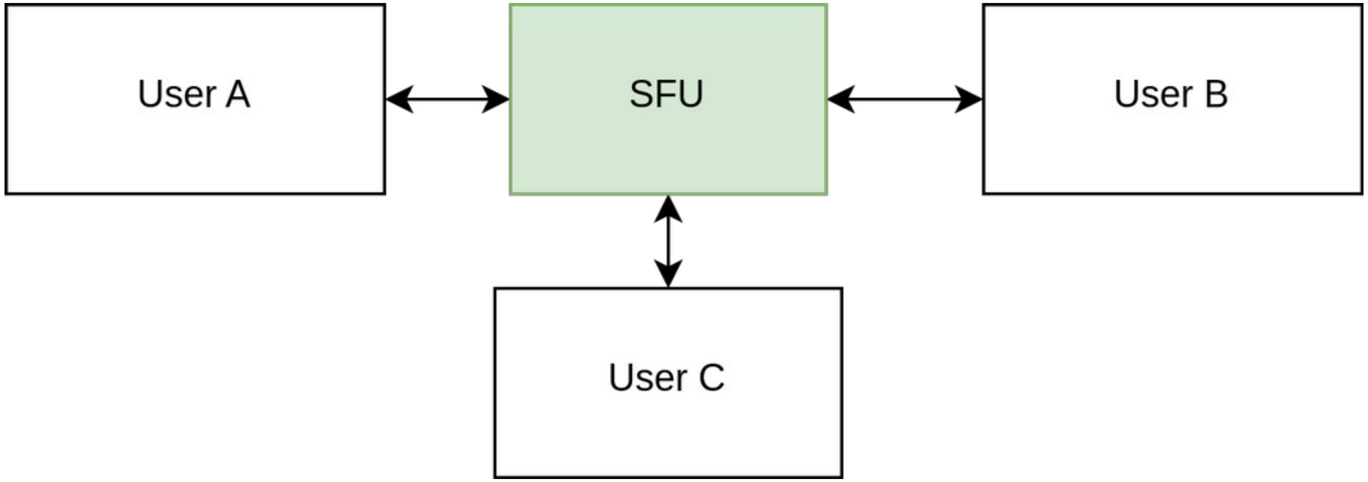
LiveKit Server - Intro

- Open Source WebRTC System
- Signalling Server: Negotiates SDP between clients and server
- Selective Forwarding Unit (SFU)
  - Not a Peer-to-Peer architecture, analogous to a router
  - Clients transfer media to and from SFU, do not communicate directly
  - Limits number of connections per client
- LiveKit distributes client libraries in many languages, including Rust

WebRTC Peer-to-Peer



WebRTC - SFU



COLLABORA

LiveKit Source/Sink Elements

Basic Audio/Video Support

LiveKit Sink Element

- Sends media to LiveKit room
- Same code as all other webrtcsink elements
- All there is to do is add a new signalling client
  - Implement the `Signallable` GObject interface
  - Integrate with LiveKit client library
    - Propagate ICE candidates
    - Send SDP Offer

LiveKit Source Element

- Receives media from LiveKit room
- Same code as all other webrtcsrc elements
- Builds on existing LiveKit signalling client
  - Implement Consumer mode
  - Answer SDP offer from server
- Signalling client controls subscription to tracks with 2 possible modes:
  - Subscribe to one sender
  - Subscribe to all senders
    - Optionally, filter out fixed set of senders
- Pad created for each track

COLLABORA

Video Simulcast

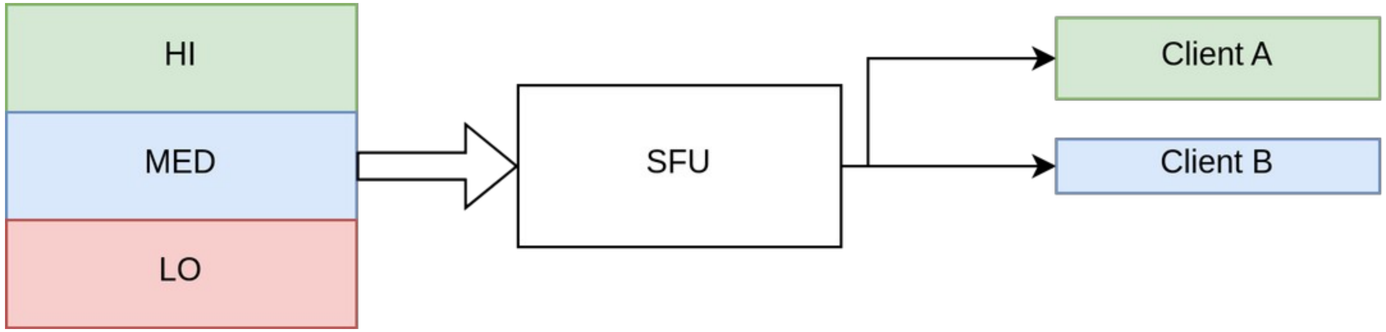
Video Simulcast - Intro

- Adaptive Technique
- Sender provides multiple versions of the same stream at different qualities
- LiveKit server forwards optimal version to each receiver
- Prevents any individual receiver’s poor connection quality from downgrading other receiver’s video quality

Video Simulcast - Implementation

- Only applies to webrtcsink
  - Simulcast is mostly transparent to receivers
- Limitations of webrtcsink
  - No mechanism allowing user to group and rank tracks
  - No capability to generate Simulcast SDP attributes

Video Simulcast



COLLABORA

End-to-End Encryption (E2EE)

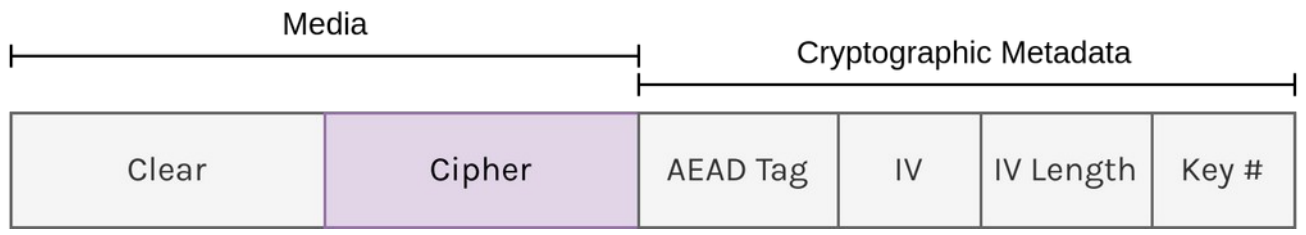
E2EE - Introduction

- Why? Isn't WebRTC already encrypted?
  - Yes, but this encryption has limited utility primarily for two reasons:
    - Key Exchange is performed over signalling protocol
    - Encryption is point-to-point. In the SFU configuration of WebRTC, encryption is between each client and the LiveKit server.
  - If the signalling server or SFU is not trusted, the encryption does not provide security.

E2EE – LiveKit’s Scheme

- Implemented on top of WebRTC Insertable Streams
  - Cryptographic filters are applied to the receiver and sender
- Part of each buffer is unencrypted, how much depends on codec
- Cipher is AES-GCM, which verifies the unencrypted portion

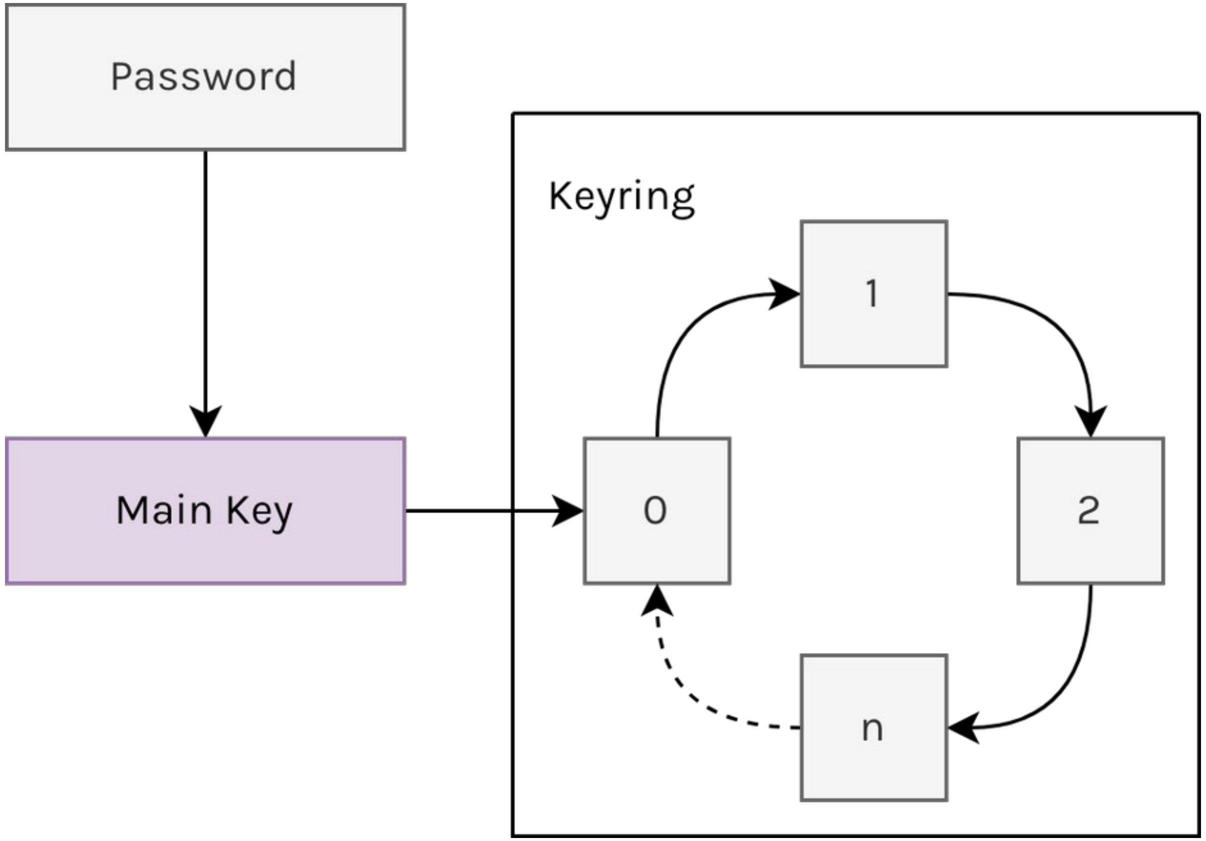
E2EE – Message Structure



E2EE – LiveKit’s Scheme

- Most typical configuration is a Pre-shared Key (PSK) derived from a password
- A ring of sub-keys are generated from key and used to encrypt data

E2EE – Key Management



E2EE - Implementation

- Encryption and Decryption are implemented as `GstBaseTransform`
  - Application inserts elements using request-encoded-filter signal
- Opus, VP8 are straightforward
  - Clear portion ends at fixed position
- H.264 needs special treatment
  - Clear portion ends at first picture NAL unit
  - Stream must be RBSP-escaped

COLLABORA

Development Challenges

Development Challenges - Simulcast

- Multiple Versions of Specification
- Webrtcbin support for simulcast is not well-documented
- Requires Design Changes to Webrtcsink
- Coordination of RTP Header extensions is required

Development Challenges - E2EE

- Codec Parsing
  - Filter elements need to parse and repack H.264 streams in order to partition data
- GStreamer RTP Payloading
  - VP8 payloader relies on information inside each incoming buffer and encrypted data can confuse it
  - H.264 payloader trims trailing zero bytes
- Libwebrtc interoperability
  - Has an unexpected behavior with H.264 streams, it expands absent optional values in the SPS unit with the defaults when receiving data, which tampers with encrypted stream. Custom SPS expander element to imitate libwebrtc was implemented

Development Status

- Livekit Elements - Merged:
  - Sink: https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/merge_requests/1262
  - Source: https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/merge_requests/1461
- Simulcast – Draft:
  - https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/merge_requests/1450
- E2EE - Soon

Thank you!