GStreamer for audio distribution at Sveriges Radio

Karl Johannes Jondell

Christofer Bustad

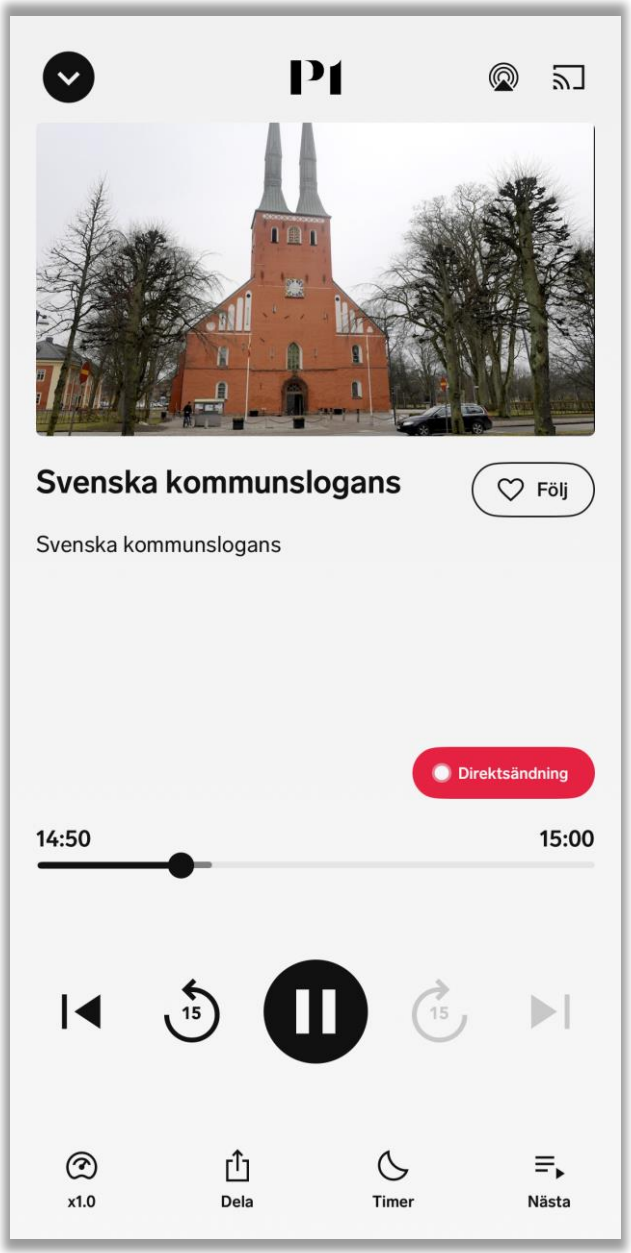
sverigeSRradio

About Sveriges Radio (SR)

- SR is one of three public broadcasting companies in Sweden: SR, SVT, UR.
  - This year celebrating 100 years on the air.
- Our primary medium is radio: terrestrial FM & DAB, IP based-distribution.
- 7 million listeners every week. Sweden’s population is 10.5 million.
- Radio channels: 4 national, 25 local, 7 special/niched channels (primarily digital-only).
- Public trust in SR is foundational and consistently measured high.
- Commercial and political independence.
- Mission to be innovative.
- Emergency preparedness mission.

Our IP offerings

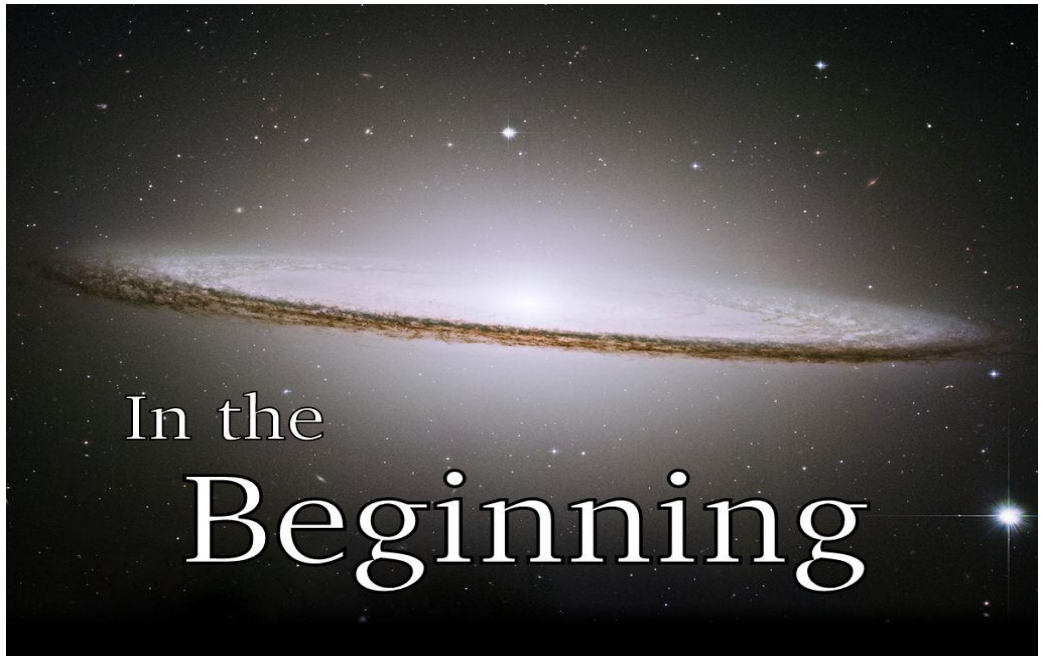
- Our platforms
  - iOS/Android app
  - sverigesradio.se
  - 3rd party platforms
- Linear
  - Using HLS for our own platforms & some 3rd party platforms.
  - Using ICY for backwards compatibility and most 3rd party platforms.
  - **HLS:** 52 live streams with 24/7 content encoded in: AAC
  - **ICY**: 37 live streams encoded in: AAC, mp3, FLAC, Opus
- On-demand
  - Podcasts, audio clips, catch up radio
  - 3rd party platforms
  - Formats used: AAC, mp3



sverigesradio

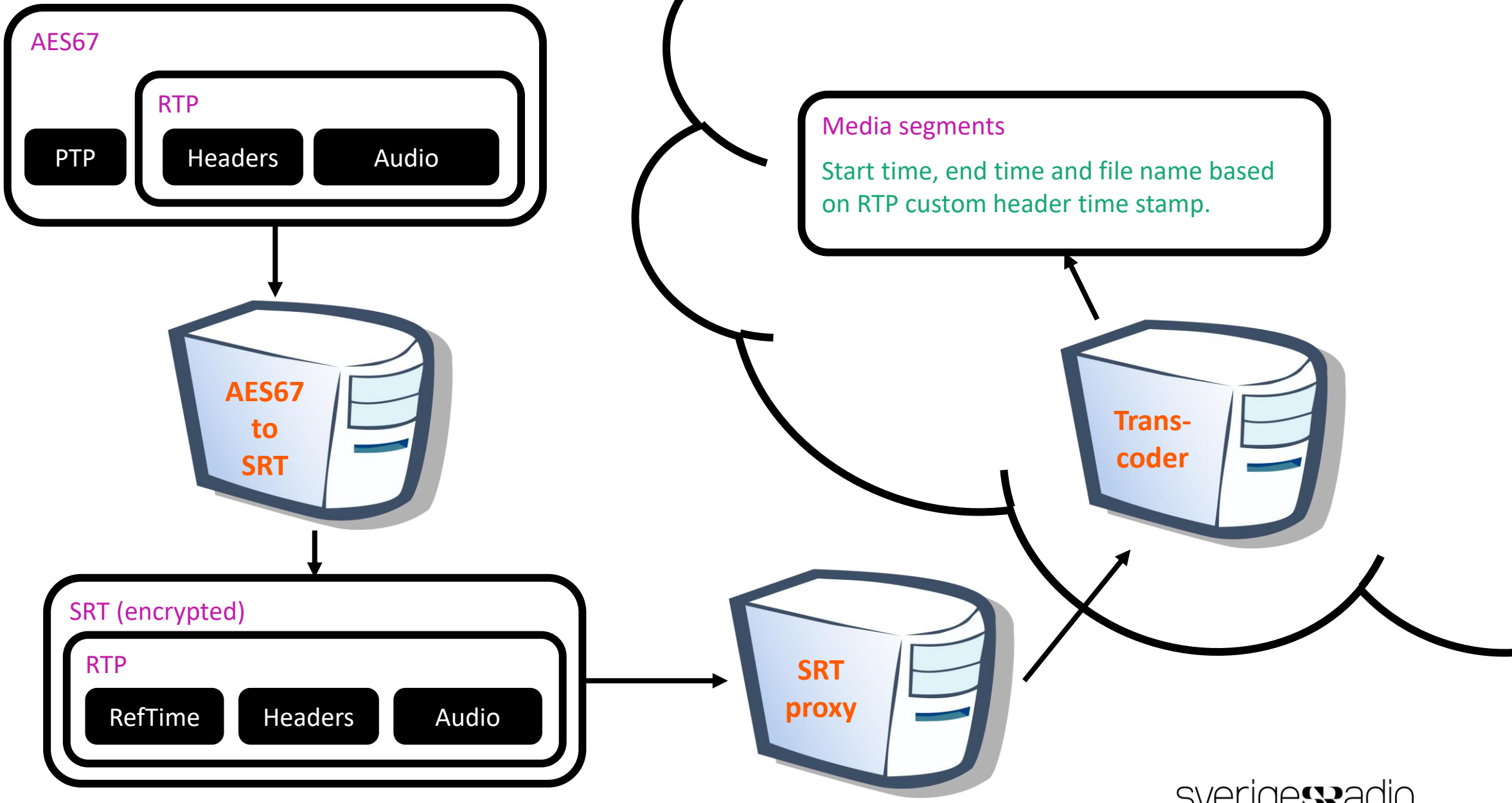
In the beginning...

- There was a need for a renewed IP based distribution system.
- At the crossroads: modernising existing solution or building new from open software.
- Requirements
  - Containerised
  - Hybrid cloud
  - Redundancy
  - Reusability
  - Lower latency
  - Audio watermarking
  - Broadcast processing
- We decided to use GStreamer (and Rust)!



Pilot: HLS

- Containerised setup
- Three pilot channels
  - Knattekanalen, P4 Plus, P2
- HLS is a segmented media-based protocol.
- To achieve optimal redundancy, identical media segments produced independently by several transcoders are required.
- We decided to use GStreamer for transport, encoding, packaging and metadata, but not for HLS manifests, playlists, and peripheral services.
  - GStreamer application built in Rust for stability. Peripheral services built in Perl and Python.
- The foundation of the pilot was built by Centricular and is available at: https://github.com/centricular/aes67-relay-chunker



sverigesradio

Pilot: Lessons learnt

- Initially we had a gradual drift between our redundant pathways. The media segments created were identical, but the creation time diverged slowly.
- Large overhead in low bit rates. Large number of PES.
- LATM framing caused client playback issues for HE-AAC v2. We now ADTS use instead.
  - Bug in Chrome playing HE-AAC v2 with ADTS framing.
  - We only serve HE-AAC v2 to our iOS and Android apps.
- Other issues and shortcomings not directly related to media.

Pilot: ICY

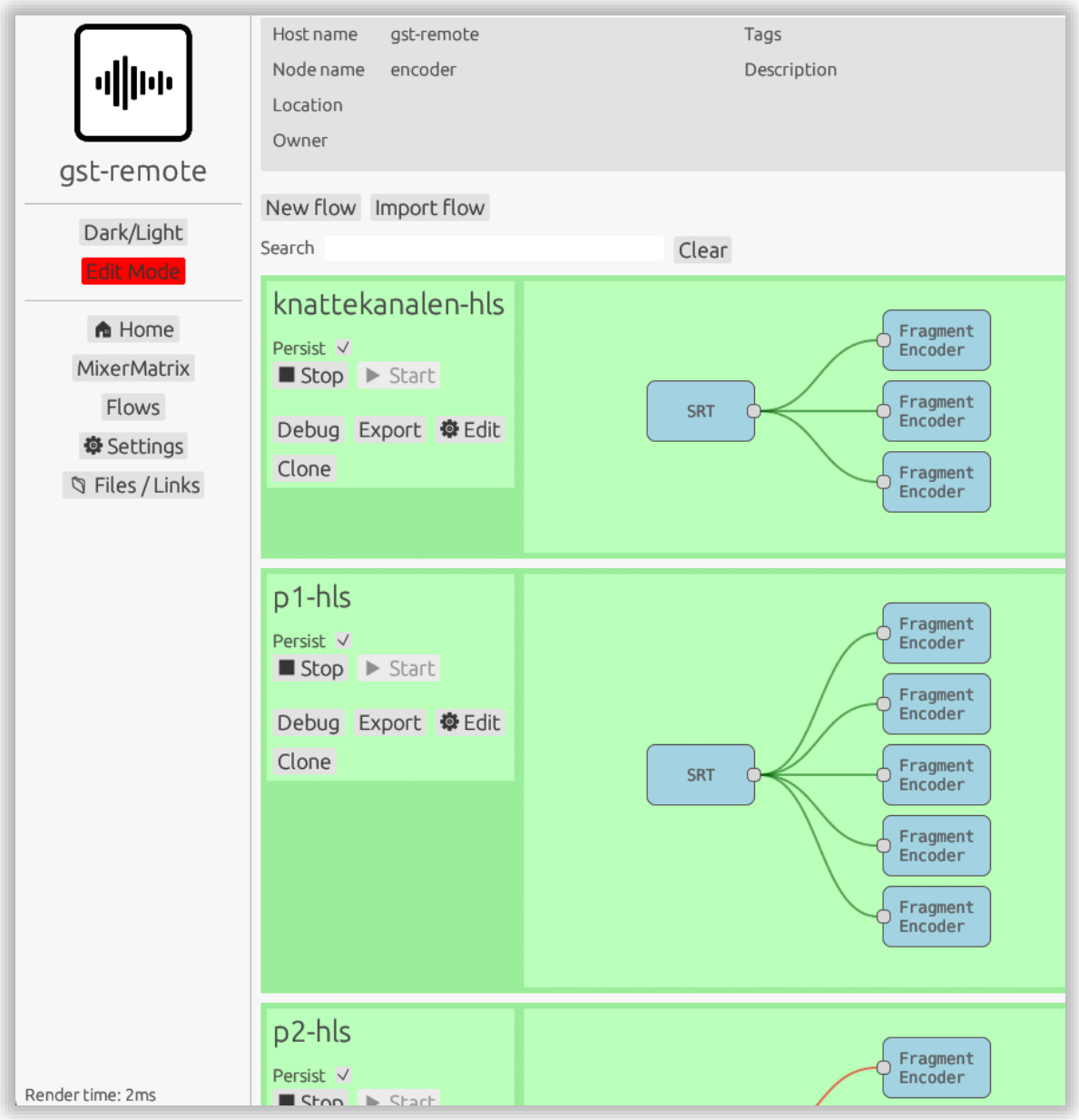
- ICY streaming protocol: Continuously growing file. Origins from WinAmp's SHOUTcast.
- We wanted to offer more formats than previously, more specifically the open formats FLAC and Opus.
- We have a long history and reputation of focusing on audio quality.
  - We both take pride in and enjoy fulfilling the high demands of our audiophilic audience to provide high quality distribution formats.
- For our music channel P2, we provide an ICY FLAC stream, that has been well received.
- Initially we used the shout2send plug-in, but now we use icecastsink (can be found in a branch on Tim-Philipp Müller's gst-plugin-rs fork: https://gitlab.freedesktop.org/tpm/gst-plugins-rs/-/tree/icecastsink).
- An ICY shortcoming: libvlc at 48 kHz sample rate.

Disaster Recovery

- A refinement of the pilot. Still hybrid-cloud. Redundancy completely independent of our main HLS distribution. Used if needed, for all linear channels.
- Used alongside both our old streaming platform and our current platform.
- No longer any need for an SRT proxy.

Our production systems

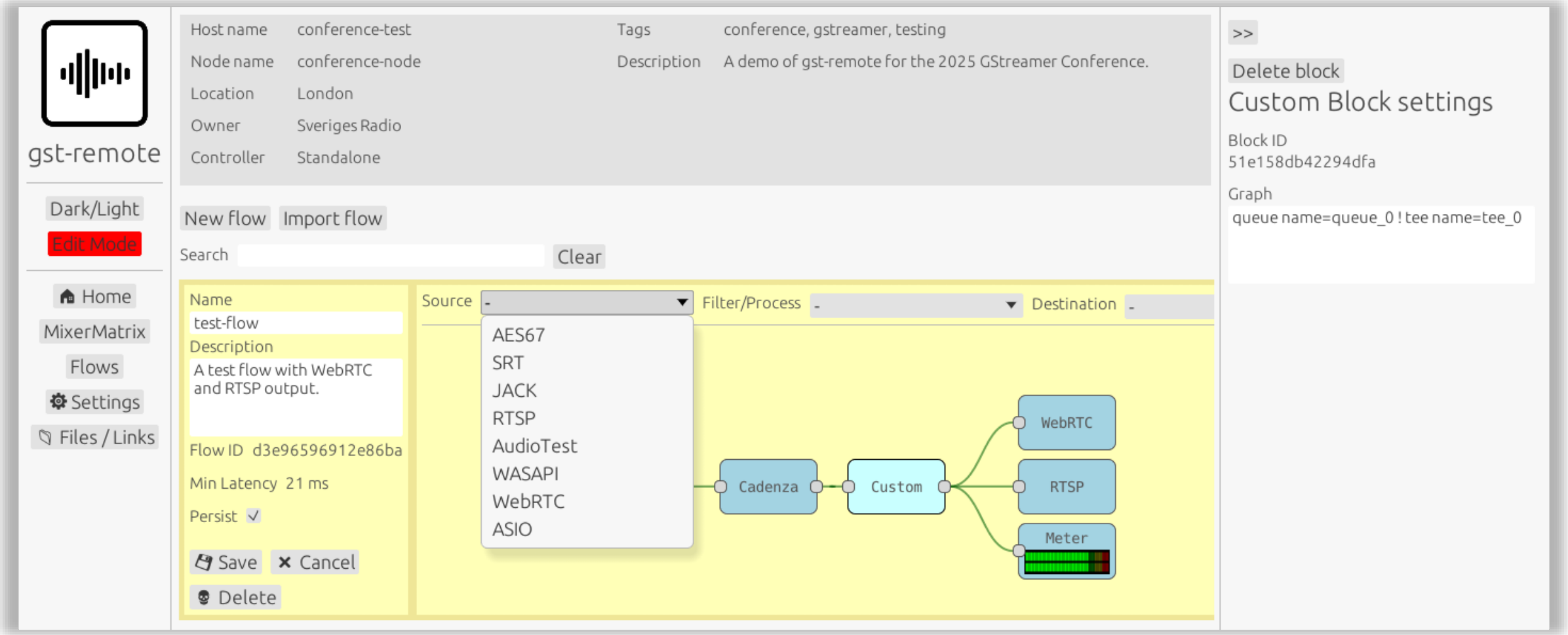
- A lot refinements based on the pilot, but the structure is basically the same.
- Completely on-prem.
- Refined pipelines
  - no dependency on PTP for media segment synchronisation
  - watermarking in custom GStreamer plug-in
- Running 52 channels on HLS and 37 channels on ICY (no pop-up channels)



gst-remote

- We needed an application to manage our GStreamer pipelines.
- Consists of a WebSocket API, reusable modular GStreamer components (bins), and a GUI. Written in Rust.
- We also use GStreamer for other applications at SR.
- gst-remote gives us a unified way of developing, managing and operating different products at the organisation.

gst-remote



sverigesRadio

GStreamer plug-ins

- Our model/starting point for developing plug-ins was: https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/tree/main/audio/csound
- Our first plug-in was audio watermarking
  - All our linear audio is watermarked regardless of distribution.
  - Channel specific code & time stamp
  - For audience measurements
  - Perceptually transparent
  - Should be late in the signal chain, but might cause extra peaks in the waveform

GStreamer plug-ins

- Currently working on a plug-in for broadcast audio processing.
- What is broadcast audio processing?
  - Audio processing, after the radio production, but before the audio reaches listeners.
  - Similar to mastering.
  - EQ, AGC, compression, limiting, stereo widening, multi-/single band, etc
- Why do broadcast audio processing?
  - To adapt the audio for a technical platform.
    - Loudness, peak levels, SNR, audio bandwidth, etc
    - Regulatory requirements & best listener experience within the technical platform.
  - To adapt the audio for the typical listener & listening environment.
  - To achieve a certain “sound”: Listeners will recognise the radio channel, and for aesthetics.
  - To adjust any mistakes from earlier in the signal chain: levels, frequency balance, etc

GStreamer plug-ins

- Audio quality has always been a top priority at SR: Listening tests, broadcast processing, etc
- 25 years ago, our colleague Torbjörn Wallentinus developed the ideas of an advanced one-band broadcast processor in collaboration with the company Factum (now Factum Radioscape)
  → The Cadenza was born!
- The hardware units has been working great for many years.
- SR recently bought a software version of the Cadenza, including the source code.
- Working towards a pilot for FM in the near future.

Wrapping up...

- Working towards getting gst-remote and our plug-ins open source to the public.
- Streaming platform foundation: https://github.com/centricular/aes67-relay-chunker

- Questions?