WebRTC support in WebKitGTK and WPEWebKit with GStreamer

Philippe Normand

FOSDEM 2025

igalia

Outline

- Intro
- Current status
- Practical use-cases
- On-going work & plans

About me

- Multimedia engineer at Igalia since ~2009
- Working on GStreamer and WebKit Linux ports
- Was involved in the OpenWebRTC project and its integration in WebKitGTK

Intro - WebKit

- FOSS Web engine, maintained by Apple, Igalia, Sony
- WebView API, generally allowing native apps to render the web
  - Web browsers
  - Set-top-box UIs
  - News readers
- Platform-specific (upstream) ports:
  - Apple products
  - Sony Playstation
  - Linux Desktop (GTK port)
  - Linux Embedded (WPE port)
  - Windows & JSCOnly

Intro - Why not use LibWebRTC? (1/2)

Intro - Why not use LibWebRTC? (2/2)

WPE and WebKitGTK support LibWebRTC but:

- BoringSSL license preventing use in GPL apps
- Footprint in tarballs
- Integration with GStreamer video decoders very fragile
- Lack of hardware-accelerated video encoding
- Why do we need another Multimedia framework anyway?

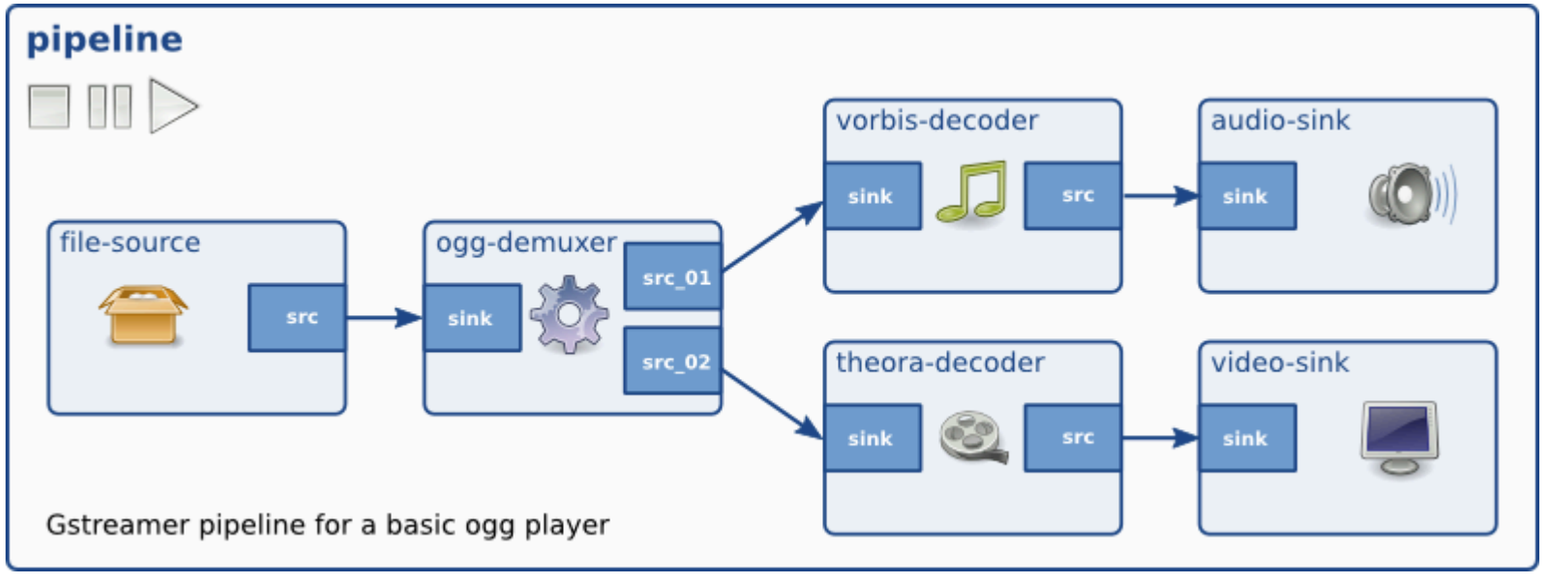
Let’s now talk about GStreamer and GstWebRTC ;)

Intro - GStreamer (1/2)

- Cross-platform open-source multimedia framework
- Widely deployed on embedded platforms (even in the ISS space station!)
- Custom plugins available, depending on target Set-top-box platforms

Checkout the **GStreamer: State of the Union 2025** talk tomorrow in the Open Media devroom (K.3.401) at 3pm!

Intro - GStreamer (2/2)



Intro - GstWebRTC

- one GStreamer element (`webrtcbin`) for integration in pipelines
  - Can be thought of as a PeerConnection object
  - Signals and properties to perform Offer/Answer
- one GStreamer library shipping public API (`libgstwebrtc.so`)
  - Defines classes for WebRTC-related objects (Transceivers, …)
- Pluggable ICE and Transport backend support (only libnice currently)

Current status

WebRTC in WebKit, using GStreamer

- Dedicated pipeline for audio/video capture
- Another pipeline for encoding and streaming over the network
  - Seamless integration with hardware-accelerated encoders
  - Zero-copy encoding from webcam to RTP payloader possible
- Incoming streams routed to separate pipelines for playback
  - Seamless integration with hardware-accelerated decoders

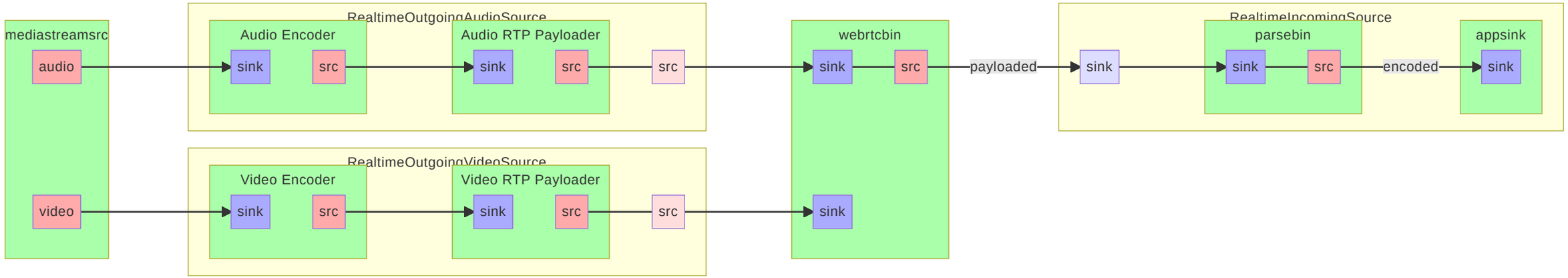
Stream capture

- One pipeline per capture device (Camera, Microphone, Desktop)
- Permission request handling
- For `getDisplayMedia()`:
  - Desktop portal (PipeWire) mandatory
  - Capture pipeline connects to PipeWire, gets frames as DMABufs

`MediaStream` handling

- `webkitmediastreamsrc` GStreamer source element acting as Observer for:
  - Capturer pipelines
  - Canvas / `<video>` elements
  - Incoming WebRTC `MediaStreams`
- Each `MediaStreamTrack` maps to a `GstPad` in `webrtcbin`

`PeerConnection` handling (1/2)



`PeerConnection` handling (2/2)

- `webrtcbin` implements the JavaScript Session Establishment Protocol (JSEP)
- Almost seamless integration with WebKit's PeerConnection infrastructure
- `GstPromise` integration with WebKit's `Promise<T>`

Incoming `MediaStream` playback

- In JS: `videoElt.src = mediaStream`
- => In `WebCore::MediaPlayerPrivateGStreamer`: `playbin3 uri=mediastream://uuid`
- ==> Internally hooks the `MediaStream` to `webkitmediastreamsrc`

Additional WebRTC-related APIs (1/2)

- Statistics
  - Queried to `webrtcbin` using a GObject action signal
  - Some informations can’t be filled by `webrtcbin`
  - `frames-decoded`, `frames-encoded`, `bitrate`, … filled in by WebKit
- DataChannel, `RTCDataChannel` maps fairly well with `GstWebRTCDataChannel`

Additional WebRTC-related APIs (2/2)

- `replaceTrack()`
- `pc.createOffer({offerToReceiveAudio: true, offerToReceiveVideo: true})`

Practical use-cases

The obvious ones: Video calls

- Jitsi: Very early stage bring-up
- Livekit: We can connect and receive A/V. Sending not working yet

Amazon Luna

- Game streaming, gamepad events sent using DataChannel
- Optional support for live WebCam/Mic overlay to Twitch (WIP)
- **Demo**

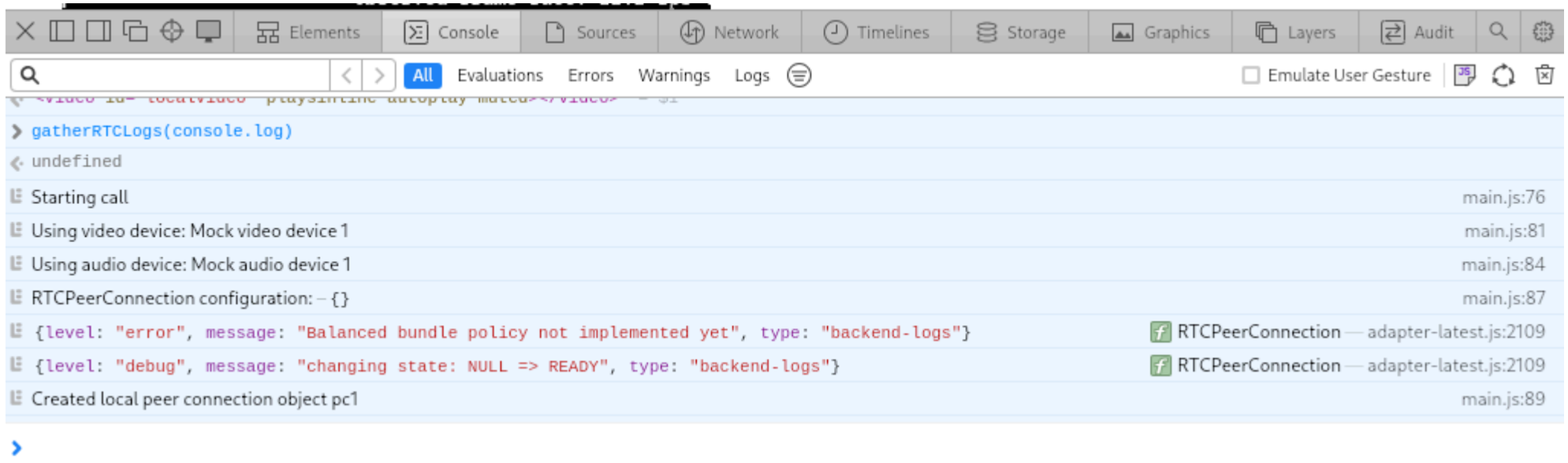
Zoo, Industrial modeling application

- CAD model rendered server-side with WebRTC
- DataChannel for sending pointer events
- **Demo**

On-going work

WebRTC Devtools in WebKit (1/3)

- Currently a well-hidden WebInspector feature: `gatherWebRTCLogs(func)`



WebRTC Devtools in WebKit (2/3)

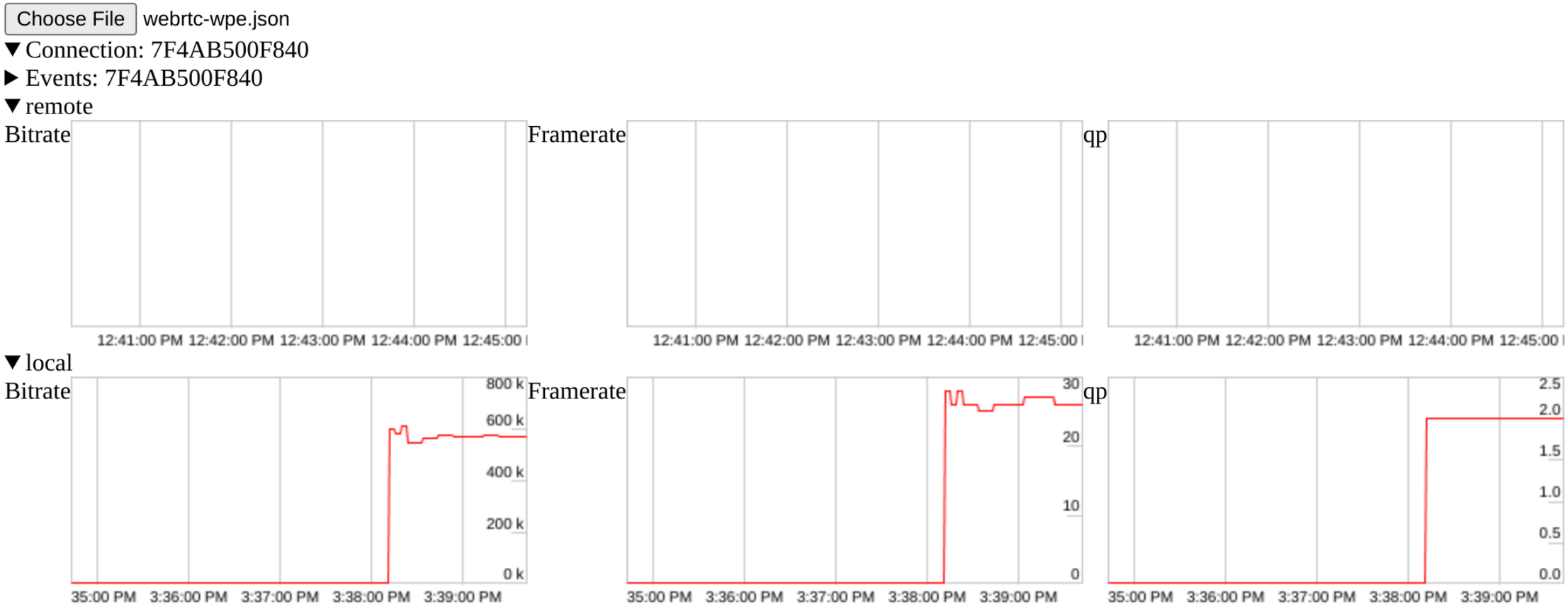
- End goal: Events and stats gathering for live graphing and post-mortem analysis
- Support for LibWebRTC and GstWebRTC WebKit builds
- JSON stream emitted by WebKit's PeerConnection backends, including timestamped events, example:

```
{"peer-connection":"7F1C6E013520","timestamp":1725960727633365.8,"type":"event","event":{"message":"PeerConnection creat
{"peer-connection":"7F1C6E013680","timestamp":1725960735855362.8,"type":"stats","event":{"type":"inbound-rtp","id":"rtp-
```

- Backend enabled using an env variable: `WEBKIT_WEBRTC_JSON_EVENTS_FILE`
- Basic web frontend able to read such JSON file and render graphs

igalia

WebRTC Devtools in WebKit (3/3)



Graphs showing basic stats about inbound and outbound RTP streams

More features

- Basic support for network conditions simulation (packet loss, …) ✅
- Simulcast: ✅-ish
- SVC (VP8 ✅, VP9/AV1 🚧)
- Hardware-accelerated encoding 🚧 (promising results on RPi/OMX and desktop/VAAPI)
- E2EE with SFrame 🚧 (pending implementation details and potential interop issues)

Access to capture devices from sandboxed WebKit

- The current capture device pipeline runs in WebProcess (BAD!)
- Ideally the WebProcess should be sandboxed, hence no direct access to capture devices
- Currently we allow-list v4l devices in the sandbox
- Plan (2025): Integration with the desktop Camera portal, giving us access to PipeWire nodes
- WPE on Embedded platforms will still need v4l devices allow-listing

igalia

UDP streaming from sandboxed WebKit

- The current streaming pipeline runs in WebProcess (BAD!)
- Ideally the network usage should be restricted to the NetworkProcess
- `libgstwebrtc` now supports custom ICE implementations
- `librice` provides a Sans-IO implementation for ICE handling
- Plan (2025): Implement a librice-based ICE backend in the WebProcess, handling IPC I/O with NetworkProcess

Thanks!

Any question?