GstWebRTC in WebKit, current status & plans

Philippe Normand

GStreamer conference 2025

Outline

- Intro
- What’s new?
- Network-related on-going work
- Plans

igalia

Intro

igalia

WebKit

- Cross-platform FOSS Web engine, started by Apple as a fork from KHTML
- Powering Safari, GNOME Web, and widely deployed on embedded products (WPE port)
- Multi-process architecture:
  - UIProcess: WebView API consumer
  - WebProcess: DOM, JavaScript, Media (Linux ports), …
  - NetworkProcess: Network handling
  - GPUProcess: WebGL, WebGPU, Media (Apple ports)

WebRTC JavaScript APIs

- `getUserMedia()`: Access to camera/microphone
- `getDisplayMedia()`: Access to screen contents
- `MediaStream`: Playback of streams, integration with `<video>`, canvas, WebGL, WebAudio
- `RTCPeerConnection`: Peer-to-peer connection establishment and streaming

igalia

WebRTC WebKit backends

- LibWebRTC:
  - Basic GStreamer integrations (video decoding)
  - Not shipped in releases
- GstWebRTC:
  - Relying on `webrtcbin` and `libgstwebrtc`
  - Hopefully will ship in releases ;)

igalia

What’s new?

igalia

Coordination of Video Orientation (CVO) (1/2)

- Spec: 3GPP TS 26.114
- `urn:3gpp:video-orientation` custom RTP header extension, reads/writes one byte:

Bits (LSB)	7	6	5	4	3	2	1	0
Definition	0	0	0	0	C	F	R1	R0

- `C`: Camera direction, which we should handle eventually!
- `F`: `1` (horizontal flip), `0` (no flip).`R1` and `R0`: Rotation signaling
- 6 bits variant (`urn:3gpp:video-orientation6`): Not supported

Coordination of Video Orientation (CVO) (2/2)

- Sending direction workflow:

  1. WebKit's `MediaStream` source adds `WebKitVideoFrameMetadata` on its video buffers
  2. Extension `write` function: reads orientation/flip value from the buffer's metadata

- Receiving direction workflow:

  1. Extension `read` function: stores orientation/flip values to the buffer as a new `WebKitVideoFrameMetadata`
  2. WebKit's media player handles the metadata and signals the compositor

Dual-tone Multi-frequency (DTMF) tones (1/3)

- RTP format and tones specified in RFC 4733
- Signalled in SDP using `audio/telephone-event` media type
- IDL and JS:

```
partial interface RTCRtpSender {
  readonly attribute RTCDTMFSender? dtmf;
};
```

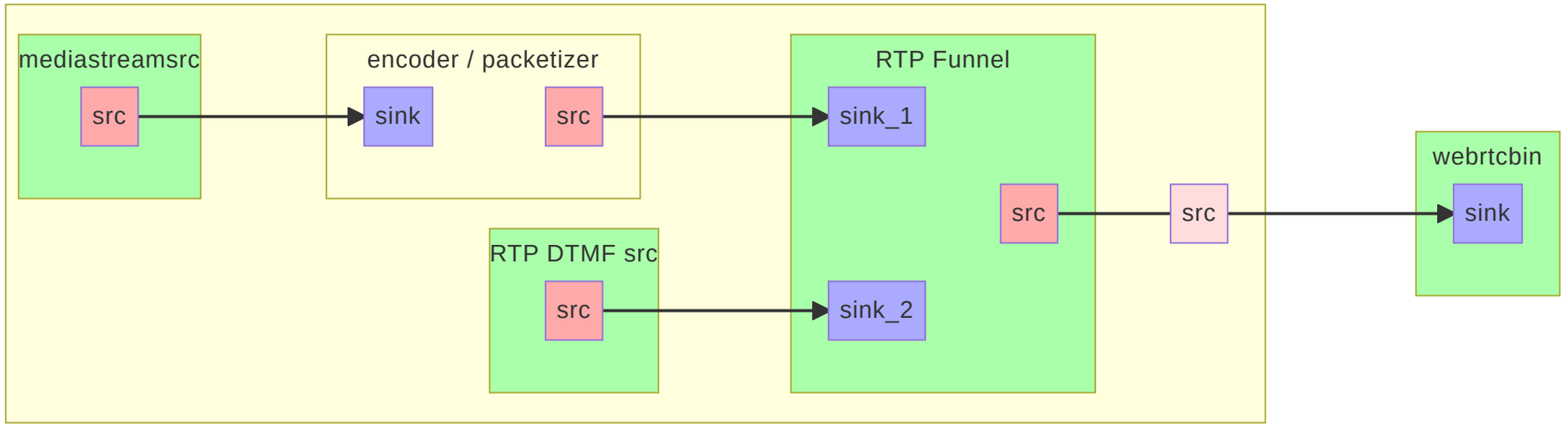
```
let dtmfSender = sender.dtmf;
let tones = "1199##9,6633221";
dtmfSender.insertDTMF(tones);
```

Dual-tone Multi-frequency (DTMF) tones (2/3)

- Outgoing audio sources now include an `rtpdtmfsrc` element
- Sender backend:
  - Sends `dtmf-events` to `rtpdtmfsrc`
  - Local playback with a dedicated audio pipeline leveraging `dtmfsrc`
- Incoming DTMF events are ignored (playback would be possible but isn't mandated by spec)

igalia

Dual-tone Multi-frequency (DTMF) tones (3/3)

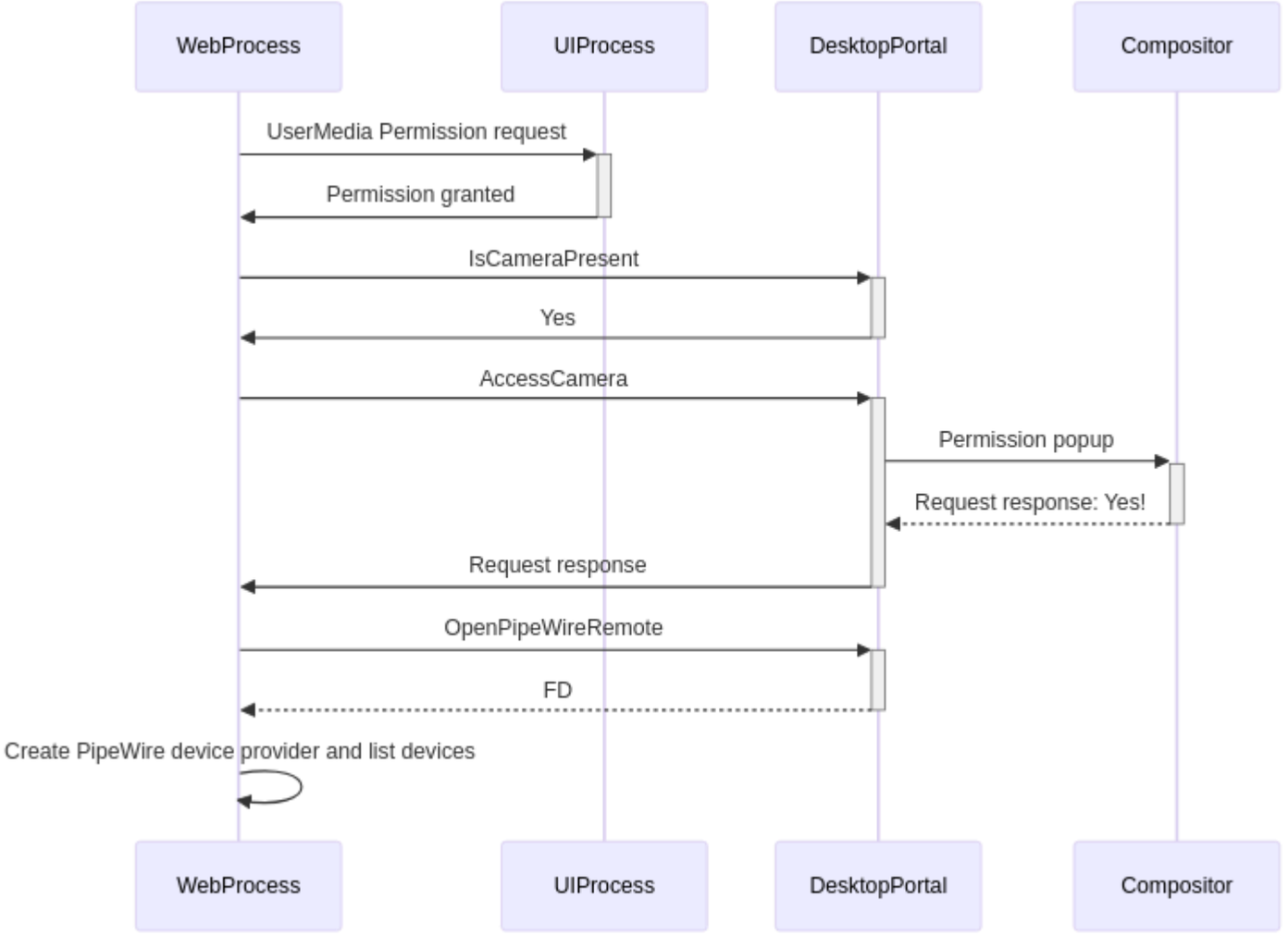


Media capture using Camera portal (1/3)

- Sandboxed WebProcess doesn’t have access to PipeWire
- Screen-sharing (`getDisplayMedia()`) already hooked-up to XDG Desktop Portal
- We now also use the Camera portal, when it’s available!

igalia

Media capture using Camera portal (2/3)



Media capture using Camera portal (3/3)

- Portal related open topics:
  - Audio capture
  - Multiple cameras
  - Screen-sharing of apps including their audio streams
- On-going discussions:
  https://github.com/flatpak/xdg-desktop-portal/discussions/1142

igalia

Misc improvements (1/2)

- Network conditions simulation using `netsim`, examples:
  - `WEBKIT_WEBRTC_NETSIM_SRC_OPTIONS=drop-probability=0.05` drops buffers in receiving direction
  - `WEBKIT_WEBRTC_NETSIM_SINK_OPTIONS=delay-probability=0.10` delays buffers in sending direction
- Rapid Synchronization (RFC 6051)
  - Already supported in `rtpbin` using `ntp-time-source=clock-time` and `rtcp-sync-send-time=1`
  - RTP header extension: `urn:ietf:params:rtp-hdrext:ntp-64`

Misc improvements (2/2)

- `getStats()` had a significant impact on CPU perf
- Stats rate limiting:
  - LibWebRTC caches stat reports for 50 ms
  - We settled on a larger default timeout, 300 ms
  - Can be overridden using `WEBKIT_GST_WEBRTC_STATS_CACHE_EXPIRATION_TIME_MS`

igalia

Network-related on-going work

Network access handling in WebKit

- Currently all our GStreamer pipelines run in the WebProcess, sandboxed
- By default no filesystem access, no network access
- NetworkProcess ⟺ WebProcess IPC for local and remote (over HTTP) files playback
- Big historical problem for our WebRTC backend!

igalia

Network handling in GstWebRTC: GstWebRTCICE

- GObject base class abstracting access to an ICE agent
- Provides various virtual methods for:
  - STUN / TURN servers configuration
  - ICE candidates handling (gathering, credentials, stats)
  - Transport / Stream handling
- `ice-agent` construct-only property in `webrtcbin`
- Default implementation based on `libnice` used by `webrtcbin`

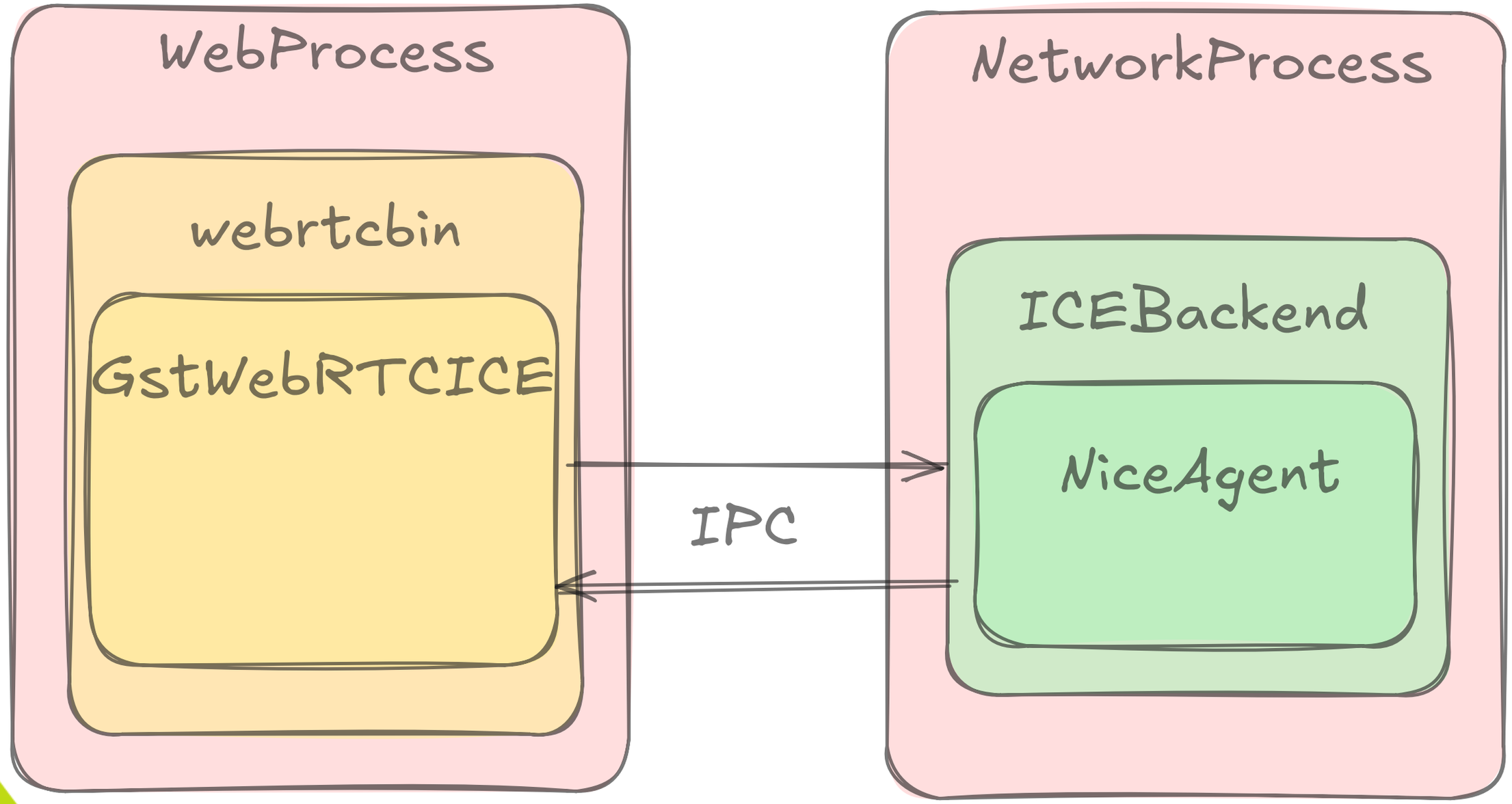
igalia

Sandboxing the ICE agent, scenarios

- Sockets abstraction in libnice
- Move the entire libnice agent to NetworkProcess
- LibRice, leveraging its Sans-IO design ;)

igalia

Experimenting with libnice (1/2)


WebProcess
webrtcbin
GstWebRTCICE
IPC
NetworkProcess
ICEBackend
NiceAgent

Experimenting with libnice (2/2)

- Large IPC surface, had to cover the entire GstWebRTCICE API
- Significant amount of IPC synchronous calls
- Conclusion: Interesting experiment but complex and likely hard to maintain

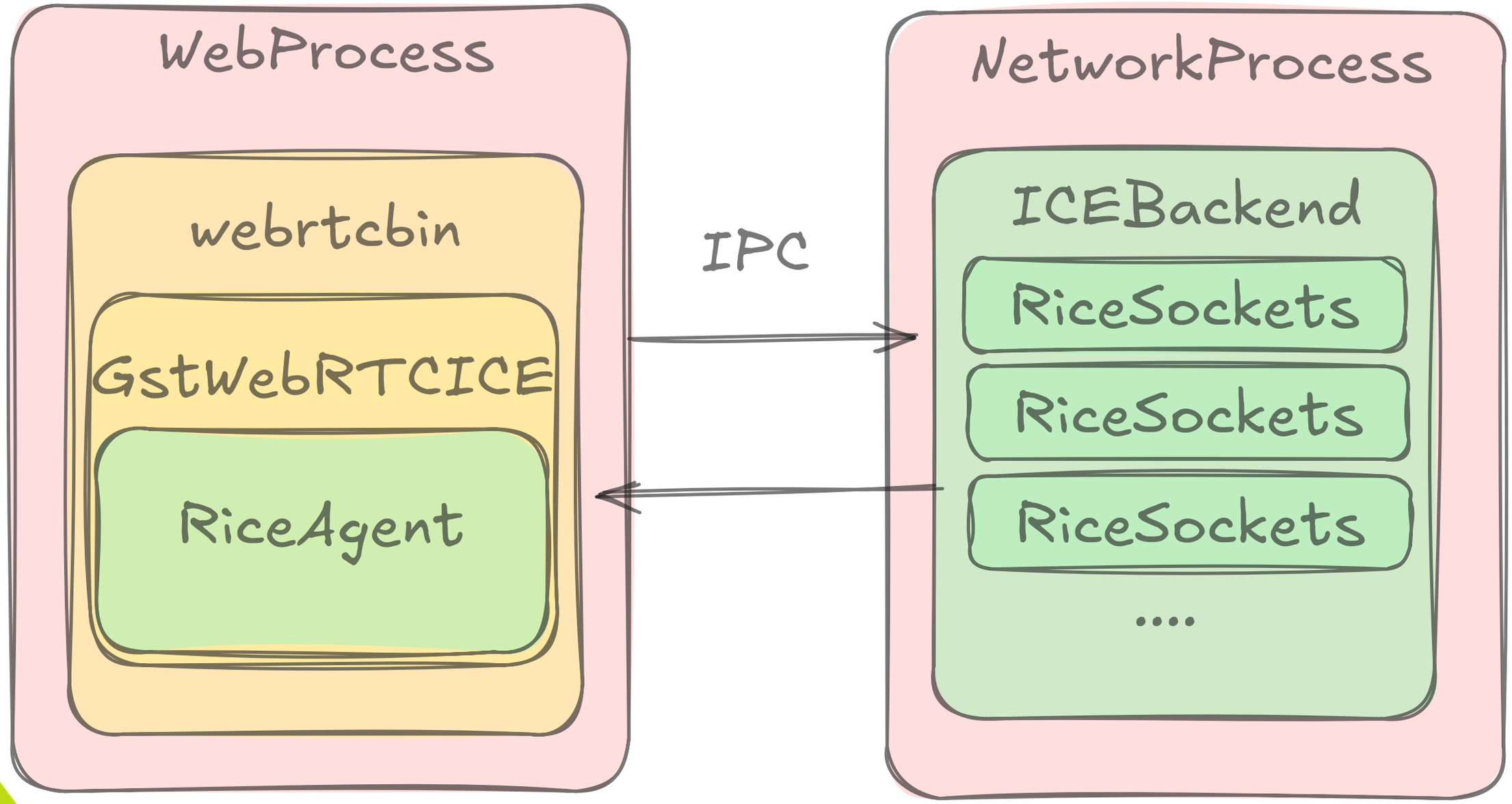
igalia

LibRice

- Sans-IO implementation of ICE (RFC 8445)
- **https://github.com/ystreet/librice**
- Written in Rust, C API provided
- Two crates:
  - `rice-proto`: Provides the ICE agent API
  - `rice-io` (optional): UDP/TCP sockets handling
- Agent integration in apps using a poll events loop

igalia

Integrating LibRice in WebKit (1/3)


WebProcess
webrtcbin
GstWebRTCICE
RiceAgent
IPC
NetworkProcess
ICEBackend
RiceSockets
RiceSockets
RiceSockets
....

Integrating LibRice in WebKit (2/3)

WebProcess ⟶ NetworkProcess

```
ResolveAddress(String address) -> (Expected<String, WebCore::ExceptionData> result)
GatherSocketAddresses(unsigned streamId) -> (Vector<String> addresses) Synchronous
SendData(unsigned streamId, enum:uint8_t WebCore::RTCIceProtocol protocol,
  String from, String to, std::span<const uint8_t> data);
FinalizeStream(unsigned streamId);
```

NetworkProcess ⟶ WebProcess

```
NotifyIncomingData(unsigned streamId, enum:uint8_t WebCore::RTCIceProtocol protocol,
   String from, String to, std::span<const uint8_t> data);
```

Integrating LibRice in WebKit (3/3)

- Pull-request: **https://github.com/WebKit/WebKit/pull/50740**
- TODO:
  - Improve `RiceAddress` cross-process sharing
  - ICE candidate stats
  - TCP sockets support
  - Proxy handling
- Planned to ship in 2.52 (March 2026)

igalia

Multicast DNS (mDNS 1/3)

- Hostname resolution within small networks
- Uses IP multicast UDP packets
- Implementations
  - macOS: Bonjour
  - Linux: Avahi

igalia

Privacy issues WebRTC (mDNS 2/3)

Example SDP candidate: `a=candidate:1467250027 2 udp 2122260222 192.168.0.196 56280 typ host`

Local IP address ex-filtration:

```
const pc = new RTCPeerConnection();
pc.onicecandidate = e => {
    if (e.candidate) {
        console.log(e.candidate.candidate.split(" ")[4]);
    }
}
pc.createOffer()
    .then(offer => pc.setLocalDescription(offer));
```

This is bad! We should prevent local IP addresses leaking in SDP/JS

MDNS in WebRTC (mDNS 3/3)

- IETF draft-ietf-mmusic-mdns-ice-candidates-03
- Open topics for the ICE/IO implementation:
  - Internally deal with local IP addresses
  - Ability to resolve FQDNs exposed in 3rd party ICE candidates
  - Expose mDNS FQDN for local IPs in host ICE candidates

Plans

Pending features

- SVC video encoding
- RTP header encryption
- ICE candidates filtering
- SFrame encryption
- Increased stats coverage
- Transceiver direction changes?

igalia

Thanks!

Any question?