Gstreamer WebRTC

Matthew Waters (ystreet00)

GStreamer conference 2017

21st October 2017

Centricular

Who Am I

- Australian
- Work - Centricular
- Graphics – OpenGL, Vulkan
- Multimedia
- WebRTC

Centricular

WebRTC Experience

- WebRTC.org
  - Integrating GStreamer-based hardware decoders
  - Wrapping WebRTC.org in GStreamer
- OpenWebRTC hardware acceleration
- GStreamer-based implementation

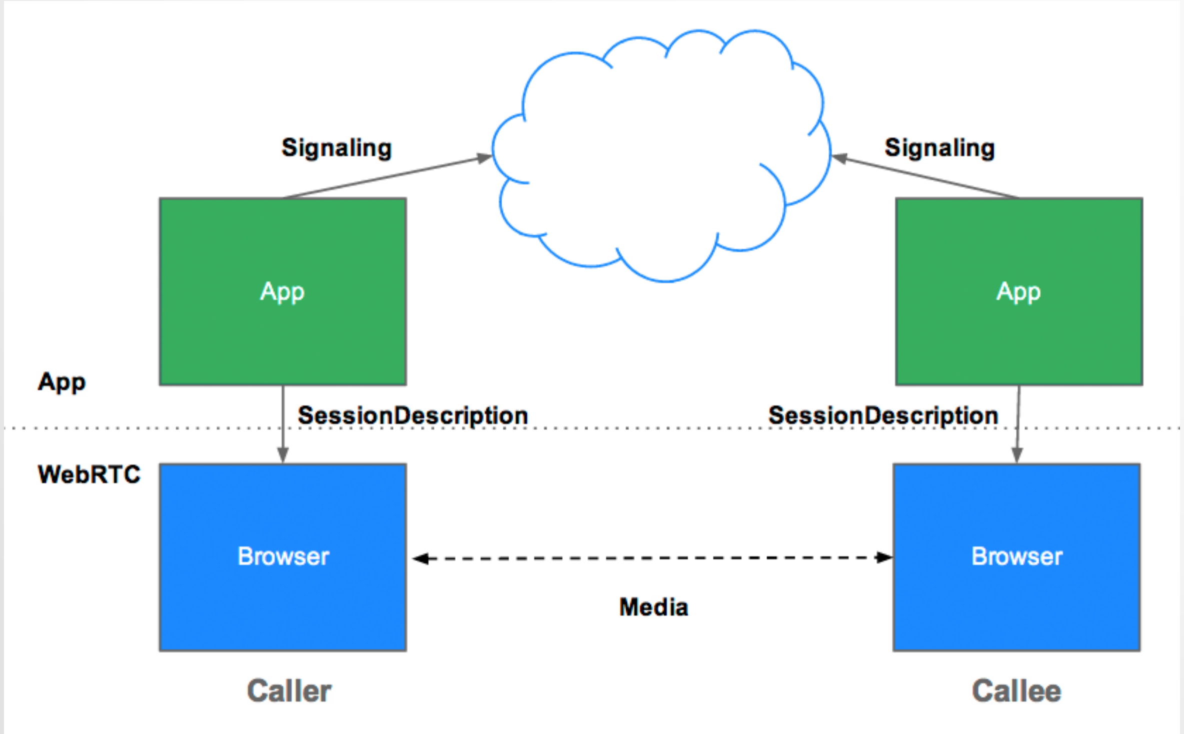
Background – WebRTC

- What are computers used for?
- Provide tools for developers to build web sites that meet these needs
- Without plugins/extensions
  - <video> html5 tag
  - <audio> html5 tag
  - Geolocation
  - WebGL
  - Canvas

Centricular

Enter WebRTC

- Real-time communication inside a web browser
- Tools for building the Skype's, Polycom's, Cisco Jabber's Google Chat/Hangouts/Duo



https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

Centricular

WebRTC limitations

- Draft specification
- 1:1 connection between two peers
- Some implementation required

https://www.w3.org/TR/webrtc/

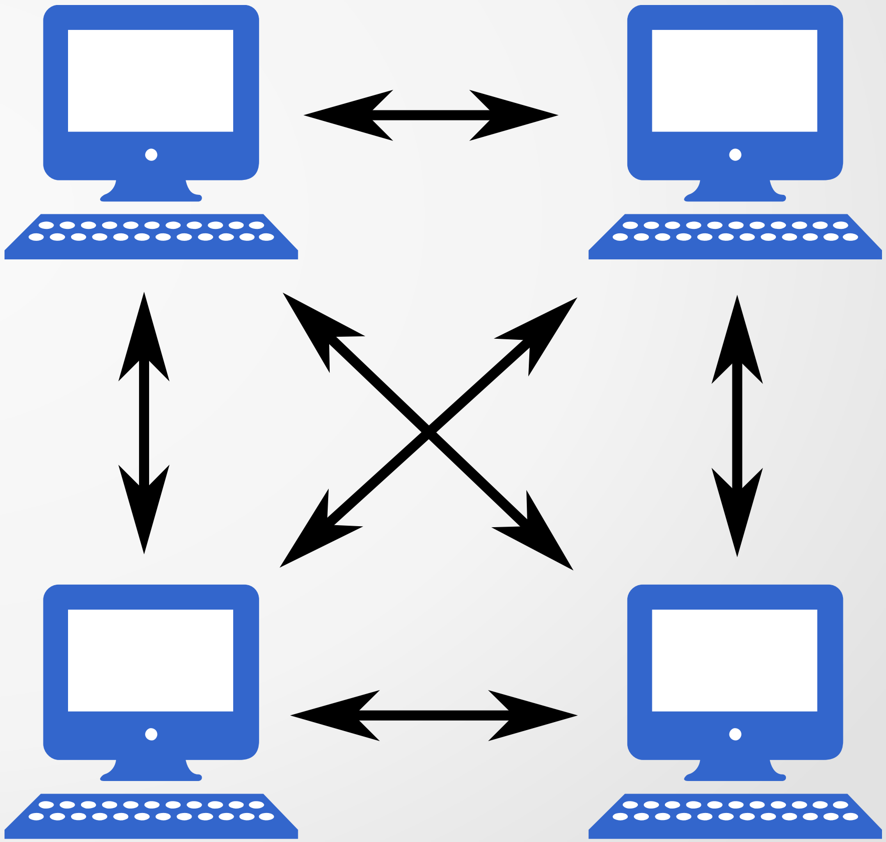
https://tools.ietf.org/wg/rtcweb/

WebRTC multi-party

- Three main models
  - Mesh – appear.in
  - SFU – Talky, SwitchRTC
  - MCU - BlueJeans

WebRTC - Mesh

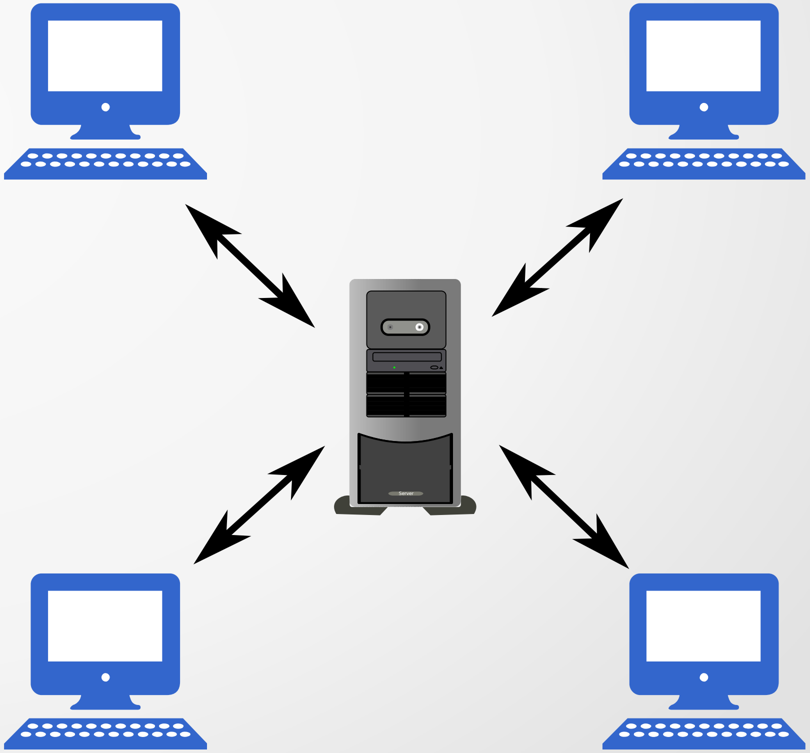
- Participants send/receive to each other participant
- Not scalable for many (5-10+) users
- Cheap for the provider
- Expensive for the user
- Mixed locally



Centricular

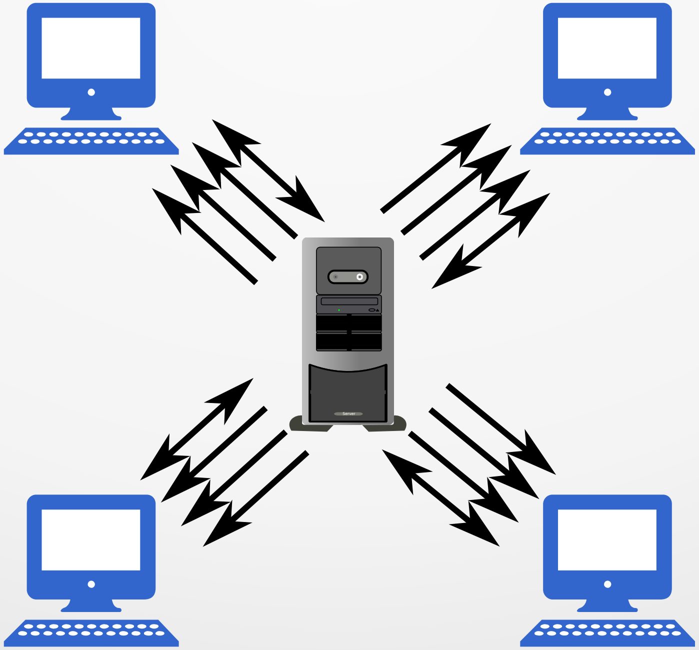
WebRTC – MCU – Multipoint Control Unit

- Central server mixes 1-n streams from the participants
- Participants send/receive a single stream
- High complexity for the provider
- Mixing is defined by the server
- Cheap for the user



WebRTC – SFU – Selective Forwarding Unit

- Central server routes data between multiple peers
- A Participant sends 1 stream, received n-1 streams
- Cheaper than MCU for the provider
- Semi-expensive for the user
- Mixed locally



Centricular

WebRTC – Complexity table

n = No of participants	**Mesh**	**SFU**	**MCU**
Provider Bandwidth	0	$O(n^2)$	O(n)
Single User Bandwidth	O(n)	O(n)	O(1)
All User Bandwidth	O(n!)	$O(n^2)$	O(n)
Provider processing	0	O(n)	O(n)-O(n!)
User Processing	O(n)	O(n)	O(1)

Centricular

Goals and Motivation

- Support the gateway/SFU and Mesh/MCU use case
- Hardware Acceleration
- Implement WebRTC API sanely
- Embedded devices
- Embedded devices/Harware Acceleration

Centricular

Existing GStreamer WebRTC Solutions

- OpenWebRTC - https://www.openwebrtc.org/
- Kurento - https://www.kurento.org/
- Custom Janus Plugin

OpenWebRTC

KURENTO

OpenWebRTC

- Ericsson Project
- Proof of Concept
- Does not support SFU
- Does have hardware accelerated encoder/decoder for Android/iOS
- Requires using a custom API

OpenWebRTC

Kurento

- Focus is on server applications
- Does not easily support hardware encoders/decoders
- Requires using a custom API

Other Noteable Implementations

- WebRTC.org
- Janus
- SIP gateways galore

WebRTC.org

- Build system woes (and changes)
- Does not support SFU
- No well-defined hardware fast-paths
- Integrating custom/hardware encoders/decoders is a pain
- Only zero-copy path are custom decoder-sink pairs
- Everything else requires raw media

Janus

- Generic WebRTC gateway server
- Core deals with signalling
- Pugins generate/consume media
- Requires a custom API

JANVS WEBRTC GATEWAY

Centricular

Janus – Streaming Plugin

- Only one way media communication into Janus
- No feedback on streaming bitrate/backlog
- Double the data through the kernel.
- Currently only static configuration.
- Example - https://planb.nicecupoftea.org/2015/07/28/hackspacehat-part-1-webrtc-janus-and-gstreamer/

JANVS
WEBRTC GATEWAY

Centricular

Hmm

- Nothing quite fits!
- Let's build something!

What Components Do We Need?

- RTP – rtpbin element
- ICE – libnice
- DTLS/SRTP/SCTP – dtlssrtpenc/dec elements
- An API – W3C PeerConnection API
- SDP Parsing Generation

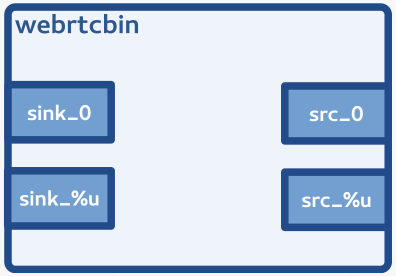
GstPromise - Promises/Futures

- Object for Promise/Future-like functionality
- Different states
  - PENDING
  - INTERRUPTED
  - REPLIED
  - EXPIRED
- Attach callback for when a reply is made
- https://github.com/ystreet/gstreamer/tree/promise

Centricular

Enter webrtcbin

- TADA!!
- https://github.com/ystreet/gst-plugins-bad/tree/webrtc



Centricular

webrtcbin – High Level Goals

- Stick as close as possible to the W3C PeerConnection API
- Gateway/full stack use case
- A dynamic number of streams
- Provide connection statistics

Centricular

webrtcbin – Low Level Goals

- RTCP muxing
- RTX - Retransmission
- FEC – Forward Error Correction
- RTP bundling
- LS groups
- Trickle ICE

Centricular

webrtcbin – Details

- SDP's are the signalling data exchange format
  - Constructed from connected sink pads and caps
- Trickle ICE candidates passed to application
- Request sink pads sending application/x-rtp
- Sometimes src pads receiving application/x-rtp

Centricular

webrtcbin – Examples

- one-way
- bidirectional
- A/V bidirectional

Centricular

webrtcbin – Demo!

- Localhost
- A/V bidirectional

Centricular

What Next? – Low Level

- FEC – Forward Error Correction
- RTX - Retransmissiion
- RTP bundling
- LS groups
- Statistics
- Fix bugs

What Next? – High Level

- Reconfiguration of streams
- Stream selection - GstStreamCollection
- Adaptive bitrate
- Full stack implementation/user

Centricular

Thanks!

- ystreet00 in #gstreamer on freenode
- @ystreet00 on twitter

Centricular