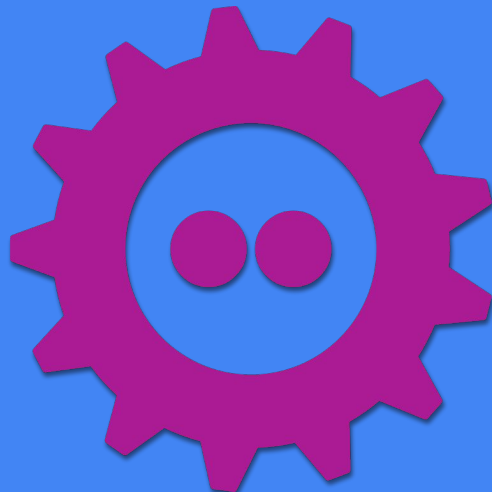
Beyond the WebRTC.org monoculture

FOSDEM 2019

Jeremy Lainé

Lennart Grahl



Jeremy Lainé

- CTO of Spacinov
- Free software since 2000
- Python since 2007
- Author of aiortc, co-author of QXmpp

Github: https://github.com/jlaine

Twitter: @JeremyLaine

Lennart Grahl

- Enjoys network programming
- Author of RAWRTC and SaltyRTC
- W3C WebRTC *Invited Expert*
- Working for Threema

Github: https://github.com/lgrahl

Twitter: @LennartGrahl

Website: https://lgrahl.de

WebRTC.org

What’s WebRTC.org?

- WebRTC : secure peer-to-peer exchange of data / audio / video
- WebRTC.org : the *de facto* WebRTC reference implementation
  https://webrtc.googlesource.com/
- Codebase widely used by browsers
  - Chrome
  - Firefox (only the A/V part)
  - Edge soon
- It’s awesome... but it’s gigantic and hard to integrate

What’s WebRTC.org?

Justin Uberti
@juberti

Follow

How many lines of code are in Google's WebRTC implementation (webrtc.googlesource.com/src)? As of the end of 2018, it consists of 1.21M lines of code (up from 1.08M in 2017); for a comparison, this is 3x as much code as the Space Shuttle software.

8:29 PM - 10 Jan 2019

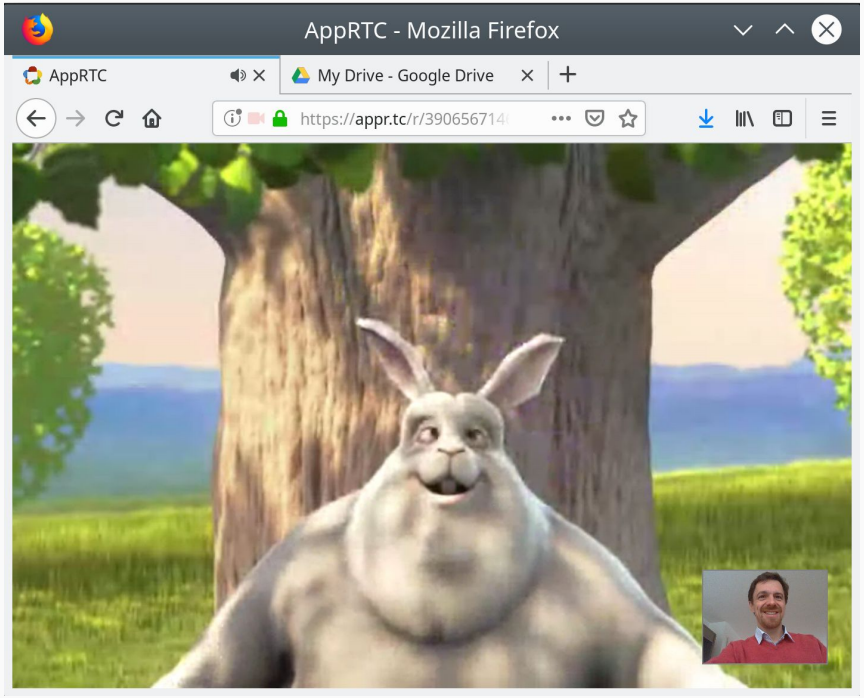
Enter aiortc and RAWRTC

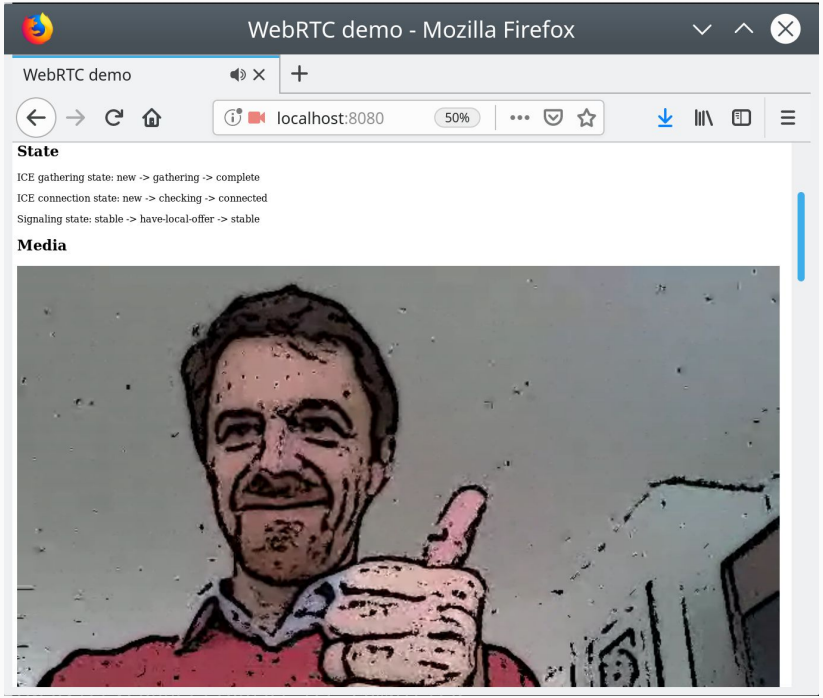
aiortc

- Written in Python, uses *asyncio*
- Supports audio, video and data channels
- Easy to hack on, around 6k lines of code, 100% test coverage
- Originally developed for automatically testing Spacivox WebRTC endpoint

- Taps into the broad Python ecosystem:
  - Uses *PyAV* for audio / video frames, leveraging FFmpeg for input and output
  - Lots of options for signaling using *aiohttp* and *websockets*
  - Easy to process media using *OpenCV* or even *Tensorflow*
  - Faster event loop? Use *uvloop*!

aiortc - ready to use examples





aiortc - use cases

- Data channels
  - Communicate with embedded devices using data channels
  - Run a VPN over data channels
- Media processing, machine learning
  - Perform real-time feature recognition on video streams
  - Central server to record images captured by mobile devices (city survey)
  - Secure server for remote access to your home surveillance cameras
- Rapid prototyping

aiortc - code snippet


```
# create peer connection
pc = RTCPeerConnection()

# add our media
player = MediaPlayer('big_buck_bunny.mp4')
pc.addTrack(player.audio)
pc.addTrack(player.video)

# create offer
offer = await pc.createOffer()
await pc.setLocalDescription(offer)

...

# handle answer
await pc.setRemoteDescription(answer)
```

RAWRTC

- Written in C
- Only supports data channels
- Intended to be resource friendly
- Uses re and usrsctp underneath

- Originally created in 2016 to test/improve data channel implementations
  - Test usrsctp
  - Forward SCTP to the FreeBSD kernel
  - First to address the *EOR problem*
  - Test throughput

RAWRTC - use cases

- Services & applications communicating with browsers
  - WebTorrent
  - Peer assisted CDNs
- All things embedded / IoT (with some power)
  - RC toys
  - Exterior/Interior illumination
  - Remote terminal behind a NAT
- Integrate it into your WebRTC stack
  - SFUs

RAWRTC - terminal demo

```
[002413514] helper-handler: (A) Data channel buffered amount low: terminal-1
[002413517] rawrtc-terminal: (A.terminal-1) Received 3 bytes
[002413518] rawrtc-terminal: (A.terminal-1) Piping 3 bytes into process...
[002413518] rawrtc-terminal: (A.terminal-1) ... completed!
[002413518] rawrtc-terminal: (A.terminal-1) Reading from process...
[002413518] rawrtc-terminal: (A.terminal-1) ... read 23 bytes
[002413518] rawrtc-terminal: (A.terminal-1) Sending 23 bytes
[002413651] helper-handler: (A) Data channel buffered amount low: terminal-1
[002413655] rawrtc-terminal: (A.terminal-1) Received 3 bytes
[002413655] rawrtc-terminal: (A.terminal-1) Piping 3 bytes into process...
[002413655] rawrtc-terminal: (A.terminal-1) ... completed!
[002413655] rawrtc-terminal: (A.terminal-1) Reading from process...
[002413655] rawrtc-terminal: (A.terminal-1) ... read 26 bytes
[002413656] rawrtc-terminal: (A.terminal-1) Sending 26 bytes
[002413819] helper-handler: (A) Data channel buffered amount low: terminal-1
[002413822] rawrtc-terminal: (A.terminal-1) Received 3 bytes
[002413822] rawrtc-terminal: (A.terminal-1) Piping 3 bytes into process...
[002413822] rawrtc-terminal: (A.terminal-1) ... completed!
[002413823] rawrtc-terminal: (A.terminal-1) Reading from process...
[002413823] rawrtc-terminal: (A.terminal-1) ... read 22 bytes
[002413823] rawrtc-terminal: (A.terminal-1) Sending 22 bytes
[002414019] helper-handler: (A) Data channel buffered amount low: terminal-1
[002414453] rawrtc-terminal: (A.terminal-1) Received 4 bytes
[002414453] rawrtc-terminal: (A.terminal-1) Piping 4 bytes into process...
[002414453] rawrtc-terminal: (A.terminal-1) ... completed!
[002414454] rawrtc-terminal: (A.terminal-1) Reading from process...
[002414454] rawrtc-terminal: (A.terminal-1) ... read 6 bytes
[002414454] rawrtc-terminal: (A.terminal-1) Sending 6 bytes
[002414457] rawrtc-terminal: (A.terminal-1) Reading from process...
[002414457] rawrtc-terminal: (A.terminal-1) ... read 16 bytes
[002414457] rawrtc-terminal: (A.terminal-1) Sending 16 bytes
[002414458] helper-handler: (A) Data channel buffered amount low: terminal-1
[002414853] rawrtc-terminal: (A.terminal-1) Received 4 bytes
[002414853] rawrtc-terminal: (A.terminal-1) Piping 4 bytes into process...
[002414853] rawrtc-terminal: (A.terminal-1) ... completed!
[002414853] rawrtc-terminal: (A.terminal-1) Reading from process...
[002414853] rawrtc-terminal: (A.terminal-1) ... read 6 bytes
[002414853] rawrtc-terminal: (A.terminal-1) Sending 6 bytes
[002414857] rawrtc-terminal: (A.terminal-1) Reading from process...
[002414857] rawrtc-terminal: (A.terminal-1) ... read 12 bytes
[002414857] rawrtc-terminal: (A.terminal-1) Sending 12 bytes
[002414858] helper-handler: (A) Data channel buffered amount low: terminal-1
[002432050] helper-handler: (A) New data channel instance: terminal-2
[002432050] helper-handler: (A) Data channel open: terminal-2
[002432050] rawrtc-terminal: (A) Starting process for data channel terminal-2
[002432176] helper-handler: (A) Data channel buffered amount low: terminal-1
[002432176] helper-handler: (A) Data channel buffered amount low: terminal-2
[002432177] rawrtc-terminal: (A.terminal-2) Received 5 bytes
[002432177] rawrtc-terminal: (A.terminal-2) Resizing terminal to 95 columns and 44 rows
[002432316] rawrtc-terminal: (A.terminal-2) Reading from process...
[002432316] rawrtc-terminal: (A.terminal-2) ... read 119 bytes
[002432316] rawrtc-terminal: (A.terminal-2) Sending 119 bytes
[002432316] rawrtc-terminal: (A.terminal-2) Reading from process...
[002432316] rawrtc-terminal: (A.terminal-2) ... read 82 bytes
[002432316] rawrtc-terminal: (A.terminal-2) Sending 82 bytes
[002432317] helper-handler: (A) Data channel buffered amount low: terminal-1
[002432317] helper-handler: (A) Data channel buffered amount low: terminal-2
[002432403] rawrtc-terminal: (A.terminal-2) Reading from process...
[002432403] rawrtc-terminal: (A.terminal-2) ... read 220 bytes
[002432403] rawrtc-terminal: (A.terminal-2) Sending 220 bytes
[002432404] rawrtc-terminal: (A.terminal-2) Reading from process...
[002432404] rawrtc-terminal: (A.terminal-2) ... read 15 bytes
[002432404] rawrtc-terminal: (A.terminal-2) Sending 15 bytes
[002432406] helper-handler: (A) Data channel buffered amount low: terminal-1
[002432406] helper-handler: (A) Data channel buffered amount low: terminal-2
```

RAWRTC Web Terminal Demo - Mozilla Firefox

RAWRTC Web Terminal

file:///home/lenny/Projects/anyrtc/rawrtc-terminal-demo/web

Connection | Terminal 2 | Terminal 3 | +

GNU nano 3.2 ../src/rawrtc-terminal.c

```
#include <string.h> // memcpy
#include <unistd.h> // STDIN_FILENO, STDOUT_FILENO, close, execvp, read, write
#include <limits.h> // USHRT_MAX
#include <signal.h> // SIGTERM, kill
#include <stdlib.h> // setenv
#include <termios.h> // ioctl, struct winsize
#include <pty.h> // forkpty
#include <rawrtc.h>
#include "helper/utils.h"
#include "helper/handler.h"
#include "helper/parameters.h"

#define DEBUG_MODULE "rawrtc-terminal"
#define DEBUG_LEVEL 7
#include <re_dbg.h>

enum {
    PIPE_READ_BUFFER = 4096
};

// Control message types
enum {
    CONTROL_MESSAGE_WINDOW_SIZE_TYPE = 0
};

// Control message lengths
enum {
    CONTROL_MESSAGE_WINDOW_SIZE_LENGTH = 5
};

static char const ws_uri_regex[] = "ws:[^]*";

struct parameters {
    struct rawrtc_ice_parameters* ice_parameters;
    struct rawrtc_ice_candidates* ice_candidates;
    struct rawrtc_dtls_parameters* dtls_parameters;
    struct sctp_parameters sctp_parameters;
};


    struct data_channel_helper* channel_helper;
```

[Read 892 lines]

Lessons learned

Challenges for alternative implementations

- Finding all the relevant documents is hard.
- The WebRTC stack is deep (ICE, DTLS, SRTP, SCTP). Try to produce re-usable building blocks.
- Structuring the code can be challenging. ORTC provides a good starting point as it breaks down the stack into discrete objects.
- Specification *politics*: “Substantial contributions to the W3C”

Benefits of alternative implementations

- A standard with a single implementor isn't really a standard
- Lean codebases are fun and easier to hack on for your custom projects
- Shake out bugs in browsers by triggering unusual code paths
- Inform standardization efforts by quickly prototyping new features
- Explore areas that weren’t originally considered by WebRTC

Where can we improve?

- Browsers are surprisingly far away from spec. Get involved!
- Data channel lobby is underrepresented
- How can we make the specification process more transparent, involve developers and users to provide direct feedback?
- WebRTC is still misunderstood by developers (peer-to-peer nature, purpose of signaling). Better docs?

Thanks for listening!

Further alternative WebRTC implementations:

- gstreamer (C)
- janus (C/JS/...)
- jitsi (Java)
- librtcdc (C)
- medooze/media-server (C++/JS/Go)
- mediasoup (C++/JS)
- pions/webrtc (Go)
- pi.pe (Java)
- ortclib (C++/C#)

0.5%

The lines of code of aiortc compared to webrtc.org