Building an Open-Source based audio streaming platform

Maxime Bugeia

2 February 2020
We stream audio!
French public broadcasting service

6 national channels
47 local channels
23 music webradios

2 Million listeners per day
200k simultaneous listeners
Before

Black Box
3rd party

Icecast
After

Our own cloud and open source based infrastructure

Icecast

HLS
Audio streaming

**ICECAST**
- Streaming server
- Master/Relay Architecture
- Single bitrate
- No cache

**HLS**
- Segmented audio/video files
- Sliding Playlist (m3u8)
- Adaptive bitrate
- Better mobile experience
- Cacheable content

Long persistent TCP connections

Just “static” files
Getting audio to the cloud
Audio to listeners

Getting audio to the cloud

Producing stream formats

AWS

Icecast

HLS
Audio to listeners

Getting audio to the cloud

Deliver content to listeners

AWS

CDN

Producing stream formats

Icecast

HLS
Audio to listeners

Getting audio to the cloud

Producing stream formats

Operating the platform

Deliver content to listeners
Getting audio to the cloud!
Transport: multicast problem

- NATIONAL CHANNELS
- LOCAL CHANNELS
- WEBRADIOS
- WAN
- MULTICAST
- DIRECT CONNECT
- AWS
Transport: multicast problem

NATIONAL CHANNELS

LOCAL CHANNELS

WEBRADIOS

WAN

MULTICAST

AWS

DIRECT CONNECT

?
Transport: multicast problem

NATIONAL CHANNELS

LOCAL CHANNELS

WEBRADIOS

WAN

MULTICAST

AWS

DIRECT CONNECT

No multicast in AWS!
Transport: SRT to the rescue

Secure | Reliable | Transport
Transport: to the cloud!
Transport: resiliency

- main
- backup

5 inputs per channel

SRT CALLER 1
SRT LISTENER 1
SRT CALLER 2
SRT LISTENER 2
backup sat
main backup
backup sat
Audio to cloud
Producing stream formats!
One program to rule them all

Liquidsoap

Open Source Audio & Video Streaming Language

Collaboration with Radio France

Receive
- SRT listener

Transcode
- Output AAC
- Output MP3
- Produce HLS
- Produce Icecast

Control
- Switch between sources
- Fallback logic
- Expose metrics
Producing the streams

- DIRECT CONNECT
- SAT BACKUP
- Icecast
- Liquidsoap
- Transcoder
- HLS
- SRT
- AWS
Define inputs

```python
main_caller1 = buffer(fallible=true, input.srt(port=10000))
main_caller2 = buffer(fallible=true, input.srt(port=10001))
backup_caller1 = buffer(fallible=true, input.srt(port=10002))
backup_caller2 = buffer(fallible=true, input.srt(port=10003))
override_caller1 = buffer(fallible=true, input.srt(port=10004))
override_caller2 = buffer(fallible=true, input.srt(port=10005))
sat_sat1 = buffer(fallible=true, input.srt(port=10006))
safe_blank = blank()
```
Radio as Code

Live: what we want to play

```javascript
live = switch(
    [(is_playing("main_caller1"),main_caller1),
     (is_playing("main_caller2"),main_caller2),
     (is_playing("backup_caller1"),backup_caller1),
     (is_playing("backup_caller2"),backup_caller2),
     (is_playing("override_caller1"),override_caller1),
     (is_playing("override_caller2"),override_caller2),
     (is_playing("sat_sat1"),sat_sat1) ])
```
Fallback logic

```python
radio_prod = fallback(
    [live,
     main_caller1,
     main_caller2,
     backup_caller1,
     backup_caller2,
     sat_sat1,
     safe_blank])
```
Radio as Code

Fallback logic

```plaintext
radio_prod = fallback([ live,
    main_caller1,
    main_caller2,
    backup_caller1,
    backup_caller2,
    sat_sat1,
    safe_blank ])
```

Never fail!
Radio as Code

Output HLS

```python
output.file.hls(on_file_change=on_file_change,
    playlist="fip.m3u8",
    segment_duration=2.0,
    segment_name=segment_name,
    "/var/hls/fip",
    ["fip_aac_midfi", aac_midfi_mpegs],
    ["fip_aac_hifi", aac_hifi_mpegs],
    radio_prod)
```

Source that never fails
Radio as Code

Output HLS

```
output.file.hls(on_file_change=on_file_change,
    playlist="fip.m3u8",
    segment_duration=2.0,
    segment_name=segment_name,
    "/var/hls/fip",
    ["fip_aac_midfi", aac_midfi_mpegts],
    ["fip_aac_hifi", aac_hifi_mpegts]),
radio_prod)
```

```
aac_midfi_mpegts = %ffmpeg(format="mpegts",
    ar=48000,
    b="96k",
    codec="libfdk_aac",
    profile="aac_low")
```

```
aac_hifi_mpegts = %ffmpeg(format="mpegts",
    ar=48000,
    b="192k",
    codec="libfdk_aac",
    profile="aac_low")
```
Radio as Code

Output HLS

```python
def on_file_change(state, fname):
    if state == "closed" and file.extension(fname) != '.m3u8' then
        upload_segment()
end

output.file.hls(on_file_change=on_file_change,
                playlist="fip.m3u8",
                segment_duration=2.0,
                segment_name=segment_name,
                "/var/hls/fip",
                ["fip_aac_midfi", aac_midfi_mpegtts],
                ["fip_aac_hifi", aac_hifi_mpegtts],
                radio_prod)

def segment_name(position, extname, stream_name) =
    timestamp = int_of_float(gettimeofday())
    "#{stream_name}_2_#{position}_#{timestamp}.ts"
end
```
Radio as Code

Output Icecast

```python
output.icecast(fallible=true,
    host="localhost",
    port=8000,
    password="icecastpassword",
    mount="fip-hifi.aac",
    aac_hifi,
    radio_prod)
```

Source that never fails

```bash
aac_hifi = %fdkaac(channels=2,
    samplerate=48000,
    bandwidth="auto",
    bitrate=192,
    afterburner=true,
    aot="mpeg4_aac_lc",
    transmux="adts",
    sbr_mode=false)
```
Scaling to 200k listeners!
Scale: Icecast

Icecast Master

Transcoder 1

Transcoder 2

Icecast Relay

Icecast Relay 1

Icecast Relay 2

Icecast Relay X

HTTPS :443

HTTP :80

https://icecast.radiofrance.fr/fip-hifi.aac

AWS NLB
Scale: Icecast

- **Icecast Master**
  - Transcoder 1
  - Transcoder 2

- **Icecast Relay**
  - Icecast Relay 1
  - Icecast Relay 2
  - Icecast Relay X

- **AWS NLB**

HTTP: 80
HTTPS: 443

https://icecast.radiofrance.fr/fip-hifi.aac

- No cache
- No autoscaling
- Icecast Relay automatic fallback
Scale: HLS

HLS base stream

Cache layer

Transcoder 1
Transcoder 2

Varnish 1
Varnish 2
Varnish X

AWS NLB

CDN

Playlists

https://stream.radiofrance.fr/fip/fip.m3u8
https://stream.radiofrance.fr/fip/fip_hifi.m3u8
fip_aac_hifi_2_301960_1579533023.ts

Segments
Scale: HLS

HLS base stream

Cache layer

Transcoder 1

Varnish 1

Transcoder 2

Varnish 2

Varnish X

AWS NLB

CDN

Playlists

https://stream.radiofrance.fr/fip/fip.m3u8
https://stream.radiofrance.fr/fip/fip_hifi.m3u8
fip_aac_hifi_2_301960_1579533023.ts

Segments

Autoscaling at cache layer

CDN level fallback possible
Scale: HLS

HLS base stream

Cache layer

- Transcoder 1
- Transcoder 2
- Varnish 1
- Varnish 2
- Varnish X

AWS NLB

CDN

Playlists
- https://stream.radiofrance.fr/fip/fip.m3u8
- https://stream.radiofrance.fr/fip/fip_hifi.m3u8
- fip_aac_hifi_2_301960_1579533023.ts

Segments

Just like web!
Transcoder Architecture

- Liquidsoap
- Icecast stream
- Icecast Master
- HLS files
- Nginx
- HLS stream
Transcoder Architecture

One Liquidsoap per channel
Automate

Declarative everything

Generating:

- SRT callers
- Liquidsoap scripts
- Icecast master & relay conf
Monitor

Export metrics

- Node Exporter
- Liquidsoap
- Icecast exporter

Collect

Prometheus

Visualize

Grafana
Monitor: real time listeners

HLS access logs
HTTP POST Json
Custom logs
Ingester

HLS

Icecast

Icecast prometheus Exporter

Prometheus
Learn more

https://github.com/mbugeia/srt2hls

Receive SRT stream
Produce HLS
Prometheus Metrics
HTTP API
Command and fallback logic

https://github.com/Haivision/srt

https://www.liquidsoap.info/
Special thanks to the Liquidsoap team!

https://www.liquidsoap.info/
Questions?
HLS / DASH Compatibility

Dynamic Adaptive Streaming over HTTP (MPEG-DASH)

HTTP-based media streaming communications protocol, an alternative to HTTP Live Streaming (HLS).

HTTP Live Streaming (HLS)

HTTP-based media streaming communications protocol.
mp3 / aac Compatibility

### MP3 audio format

**Popular lossy audio compression format**

<table>
<thead>
<tr>
<th>Browser</th>
<th>MP3 Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>6-8</td>
</tr>
<tr>
<td>Edge</td>
<td>2-3</td>
</tr>
<tr>
<td>Firefox</td>
<td>3.5-21</td>
</tr>
<tr>
<td>Chrome</td>
<td>3.1-3.2</td>
</tr>
<tr>
<td>Safari</td>
<td>10-12.1</td>
</tr>
<tr>
<td>Opera</td>
<td>3.2</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>2.1-2.2</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>3.2</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4-12.1</td>
</tr>
<tr>
<td>Opera Mobile</td>
<td>4-13.1</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>10.1</td>
</tr>
<tr>
<td>Firefox for Android</td>
<td>1.2</td>
</tr>
<tr>
<td>UC Browser for Android</td>
<td>12.12</td>
</tr>
<tr>
<td>Samsung Internet</td>
<td>1.2</td>
</tr>
<tr>
<td>QQ Browser</td>
<td>7.1</td>
</tr>
</tbody>
</table>

**Usage**

- **Global**: 96.74% (96.87%)

### AAC audio file format

**Advanced Audio Coding format, designed to be the successor format to MP3, with generally better sound quality.**

<table>
<thead>
<tr>
<th>Browser</th>
<th>AAC Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>6-8</td>
</tr>
<tr>
<td>Edge</td>
<td>2-21</td>
</tr>
<tr>
<td>Firefox</td>
<td>3.1-3.2</td>
</tr>
<tr>
<td>Chrome</td>
<td>10-12.1</td>
</tr>
<tr>
<td>Safari</td>
<td>3.2</td>
</tr>
<tr>
<td>Opera</td>
<td>2.1-2.2</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>3-4.4.4</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>12.12</td>
</tr>
<tr>
<td>Android Browser</td>
<td>4-9.2</td>
</tr>
<tr>
<td>Opera Mobile</td>
<td>10.1</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>1.2</td>
</tr>
<tr>
<td>Firefox for Android</td>
<td>12.12</td>
</tr>
<tr>
<td>UC Browser for Android</td>
<td>1.2</td>
</tr>
<tr>
<td>Samsung Internet</td>
<td>7.1</td>
</tr>
<tr>
<td>QQ Browser</td>
<td>1.2</td>
</tr>
</tbody>
</table>

**Usage**

- **Global**: 92.11% (96.7%)

---

radiofrance 51
### Ogg Vorbis audio format

Vorbis is a free and open source audio format, most commonly used with the Ogg container.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Usage Relative</th>
<th>Data Relative</th>
<th>Current Aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>12.16</td>
<td>2.3</td>
<td>6-10</td>
</tr>
<tr>
<td>Firefox</td>
<td>4.78</td>
<td>3.1-12.1</td>
<td>11.5</td>
</tr>
<tr>
<td>Chrome</td>
<td>10.1</td>
<td>2.3</td>
<td>6-10</td>
</tr>
<tr>
<td>Safari</td>
<td>79</td>
<td>79</td>
<td>73-74</td>
</tr>
<tr>
<td>Opera</td>
<td>13.2</td>
<td>all</td>
<td>13.3</td>
</tr>
</tbody>
</table>

### FLAC audio format

Popular lossless audio compression format

<table>
<thead>
<tr>
<th>Browser</th>
<th>Usage Relative</th>
<th>Data Relative</th>
<th>Current Aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>4.43</td>
<td>44-46</td>
<td>6-10</td>
</tr>
<tr>
<td>Firefox</td>
<td>17.55</td>
<td>3.1-10.1</td>
<td>10-41</td>
</tr>
<tr>
<td>Chrome</td>
<td>4.2-4.2</td>
<td>all</td>
<td>13.3</td>
</tr>
<tr>
<td>Safari</td>
<td>64</td>
<td>79</td>
<td>73-74</td>
</tr>
<tr>
<td>Opera</td>
<td>12.12</td>
<td>12.1</td>
<td>5-9.2</td>
</tr>
<tr>
<td>Safari Mini</td>
<td>79</td>
<td>79</td>
<td>73-74</td>
</tr>
</tbody>
</table>

Radio France
Real HLS diffusion

Transcoders

Segments HLS HTTP POST

NLB

Segment Forwarder

Bus

RabbitMQ

Storekeeper

Cluster Kubernetes

Segment .ts

Amazon S3

Playlists .m3u8

Player

PGM

PostgreSQL

Akamai

Radio France