Can WebRTC help musicians?

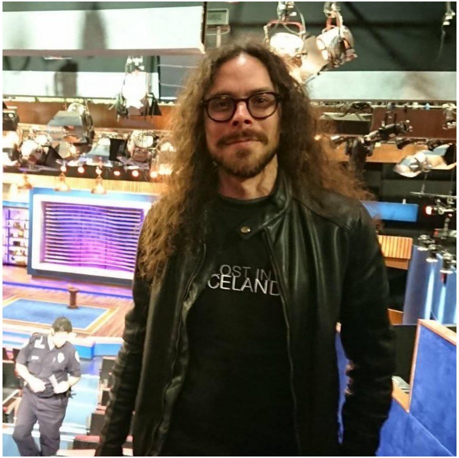
Going beyond traditional and boring use cases to support the arts

Lorenzo Miniero
@elminiero

FOSDEM 2021 Real Time devroom
6th February 2021, ~~Brussels~~ My couch

Who am I?

FOSDEM 21



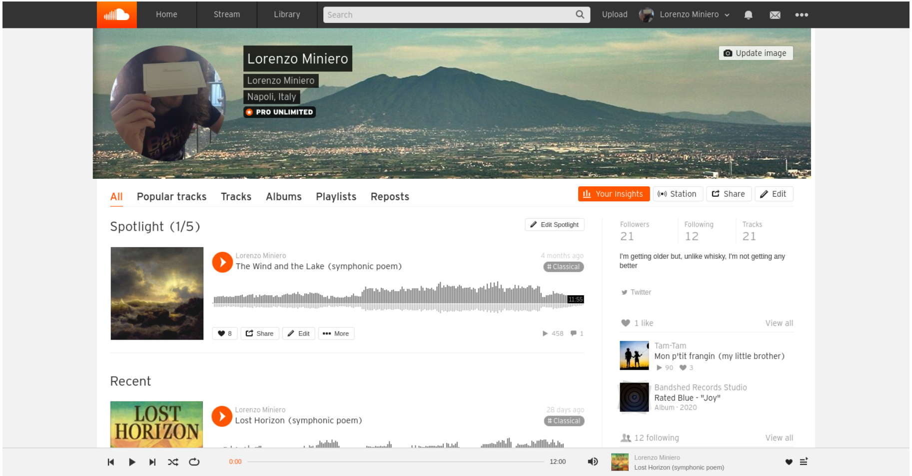
Lorenzo Miniero

- Ph.D @ UniNA
- Chairman @ Meetecho
- Main author of Janus

Contacts and info

- lorenzo@meetecho.com
- https://twitter.com/elminiero
- https://www.slideshare.net/LorenzoMiniero
- https://soundcloud.com/lminiero

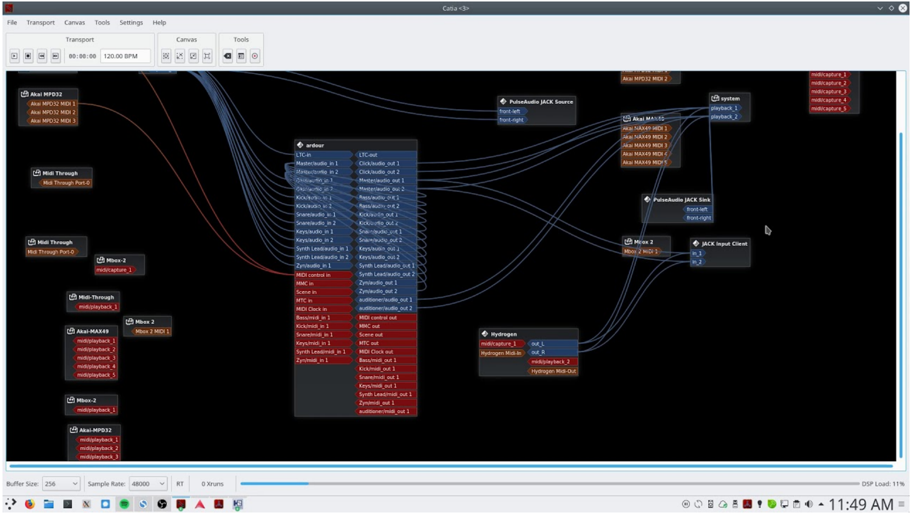
A middle age crisis! 😉



https://soundcloud.com/lminiero

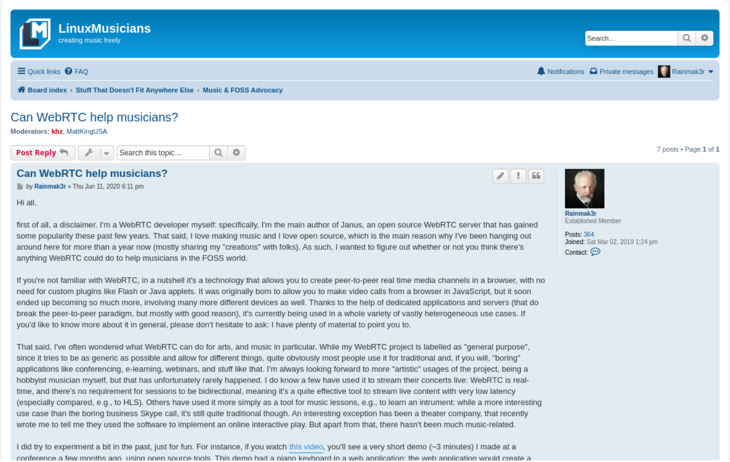
An amazing music ecosystem on Linux

FOSDEM 21



“Can WebRTC help musicians?”

FOSDEM 21



LinuxMusicians
creating music freely

Search…

Quick links FAQ

Notifications Private messages Rainmak3r

Board index ‹ Stuff That Doesn't Fit Anywhere Else ‹ Music & FOSS Advocacy

Can WebRTC help musicians?

Moderators: khz, MattKingUSA

Post Reply

Search this topic…

7 posts • Page 1 of 1

Can WebRTC help musicians?

by Rainmak3r » Thu Jun 11, 2020 6:11 pm

Hi all,

first of all, a disclaimer. I'm a WebRTC developer myself: specifically, I'm the main author of Janus, an open source WebRTC server that has gained some popularity these past few years. That said, I love making music and I love open source, which is the main reason why I've been hanging out around here for more than a year now (mostly sharing my "creations" with folks). As such, I wanted to figure out whether or not you think there's anything WebRTC could do to help musicians in the FOSS world.

If you're not familiar with WebRTC, in a nutshell it's a technology that allows you to create peer-to-peer real time media channels in a browser, with no need for custom plugins like Flash or Java applets. It was originally born to allow you to make video calls from a browser in JavaScript, but it soon ended up becoming so much more, involving many more different devices as well. Thanks to the help of dedicated applications and servers (that do break the peer-to-peer paradigm, but mostly with good reason), it's currently being used in a whole variety of vastly heterogeneous use cases. If you'd like to know more about it in general, please don't hesitate to ask: I have plenty of material to point you to.

That said, I've often wondered what WebRTC can do for arts, and music in particular. While my WebRTC project is labelled as "general purpose", since it tries to be as generic as possible and allow for different things, quite obviously most people use it for traditional and, if you will, "boring" applications like conferencing, e-learning, webinars, and stuff like that. I'm always looking forward to more "artistic" usages of the project, being a hobbyist musician myself, but that has unfortunately rarely happened. I do know a few have used it to stream their concerts live: WebRTC is real-time, and there's no requirement for sessions to be bidirectional, meaning it's a quite effective tool to stream live content with very low latency (especially compared, e.g., to HLS). Others have used it more simply as a tool for music lessons, e.g., to learn an intrument: while a more interesting use case than the boring business Skype call, it's still quite traditional though. An interesting exception has been a theater company, that recently wrote me to tell me they used the software to implement an online interactive play. But apart from that, there hasn't been much music-related.

I did try to experiment a bit in the past, just for fun. For instance, if you watch this video, you'll see a very short demo (~3 minutes) I made at a conference a few months ago, using open source tools. This demo had a piano keyboard in a web application: the web application would create a

Rainmak3r
Established Member

Posts: 364
Joined: Sat Mar 02, 2019 1:24 pm
Contact:

https://linuxmusicians.com/viewtopic.php?t=21617

Tsahi’s article back in 2017

FOSDEM 21

BlogGeek.Me

PRODUCTS SERVICES ABOUT BLOG

Sound Gurus Finding a Home in WebRTC

Market

31/07/2017

When it comes to different verticals and market niches, it seems like WebRTC can fit anywhere.

6 years in, and there are many who still question if WebRTC is the way to go with their use case. This is one of the reasons why I started the WebRTC Dataset. The idea behind it all was to

https://bloggeek.me/sound-guru-webrtc/ (2017)

Opus helps already!

- WebRTC mandates Opus, and it’s a good thing
  - High quality audio codec designed for the Internet
  - Very flexible in sampling rates, bitrates, etc.
- Different profiles for voice and **music**
  - Both encoding and decoding vary, in case
  - Can be mono and stereo (or more, as we’ll see in a minute!)
- With the right bitrate, it can sound quite good
  - ... and most importantly, with the help of WebRTC, real-time!

A “live” example: pre-recorded music via WebRTC

https://janus.conf.meetecho.com/streamingtest

Opus helps already!

FOSDEM 21

- WebRTC mandates Opus, and it's a good thing
  - High quality audio codec designed for the Internet
  - Very flexible in sampling rates, bitrates, etc.
- Different profiles for voice and **music**
  - Both encoding and decoding vary, in case
  - Can be mono and stereo (or more, as we'll see in a minute!)
- With the right bitrate, it can sound quite good
  - ... and most importantly, with the help of WebRTC, real-time!

A "live" example: pre-recorded music via WebRTC

https://janus.conf.meetecho.com/streamingtest

Opus helps already!

- WebRTC mandates Opus, and it's a good thing
  - High quality audio codec designed for the Internet
  - Very flexible in sampling rates, bitrates, etc.
- Different profiles for voice and **music**
  - Both encoding and decoding vary, in case
  - Can be mono and stereo (or more, as we'll see in a minute!)
- With the right bitrate, it can sound quite good
  - ... and most importantly, with the help of WebRTC, real-time!

A "live" example: pre-recorded music via WebRTC

https://janus.conf.meetecho.com/streamingtest

Opus helps already!

- WebRTC mandates Opus, and it's a good thing
  - High quality audio codec designed for the Internet
  - Very flexible in sampling rates, bitrates, etc.
- Different profiles for voice and **music**
  - Both encoding and decoding vary, in case
  - Can be mono and stereo (or more, as we'll see in a minute!)
- With the right bitrate, it can sound quite good
  - ... and most importantly, with the help of WebRTC, real-time!

A "live" example: pre-recorded music via WebRTC

https://janus.conf.meetecho.com/streamingtest

Multiopus: 5.1 and 7.1 surround audio

FOSDEM 21

- This is little known, but Chrome *does* support surround audio in WebRTC
  - Not really documented or standardized, though
  - Mostly just there because it's used by Stadia, today
- Multiopus (5.1 and 7.1)
  - Each packet is basically OGG with multiple stereo Opus streams
  - Number of streams determines number of channels
    - SDP munging needed on both offer and answer to specify the mapping

Some reading material if you're curious

- https://webrtcbydralex.com/index.php/2020/04/08/surround-sound-5-1-and-7-1-in-libwebrtc-and-chrome/
- https://github.com/meetecho/janus-gateway/pull/2059 (now supported in Janus)

Multiopus: 5.1 and 7.1 surround audio

FOSDEM 21

- This is little known, but Chrome *does* support surround audio in WebRTC
  - Not really documented or standardized, though
  - Mostly just there because it's used by Stadia, today
- Multiopus (5.1 and 7.1)
  - Each packet is basically OGG with multiple stereo Opus streams
  - Number of streams determines number of channels
    - SDP munging needed on both offer and answer to specify the mapping

Some reading material if you're curious

- https://webrtcbydralex.com/index.php/2020/04/08/surround-sound-5-1-and-7-1-in-libwebrtc-and-chrome/
- https://github.com/meetecho/janus-gateway/pull/2059 (now supported in Janus)

Multiopus: 5.1 and 7.1 surround audio

- This is little known, but Chrome *does* support surround audio in WebRTC
  - Not really documented or standardized, though
  - Mostly just there because it's used by Stadia, today
- Multiopus (5.1 and 7.1)
  - Each packet is basically OGG with multiple stereo Opus streams
  - Number of streams determines number of channels
    - SDP munging needed on both offer and answer to specify the mapping

Some reading material if you're curious

- https://webrtcbydralex.com/index.php/2020/04/08/surround-sound-5-1-and-7-1-in-libwebrtc-and-chrome/
- https://github.com/meetecho/janus-gateway/pull/2059 (now supported in Janus)

Tutoring and training

- A first simple use case: music lessons
  - Can be 1-to-1, or 1-to-many
  - In both cases, the closest to a “traditional” scenario
- It can basically be treated as a generic videocall or videoconference
  - Most of the interaction is conversational
  - No real need for plugging instruments directly in the call
    - A regular mic is more than enough in this context
- My sister uses Skype to teach her students...
  - ... so why not WebRTC? 🙂

Cool add-on: pitch detection?

- https://github.com/720kb/TeachMusicRTC (last updated 6 years ago, though)

Tutoring and training

- A first simple use case: music lessons
  - Can be 1-to-1, or 1-to-many
  - In both cases, the closest to a “traditional” scenario
- It can basically be treated as a generic videocall or videoconference
  - Most of the interaction is conversational
  - No real need for plugging instruments directly in the call
    - A regular mic is more than enough in this context
- My sister uses Skype to teach her students...
  - ... so why not WebRTC? 🙂

Cool add-on: pitch detection?

- https://github.com/720kb/TeachMusicRTC (last updated 6 years ago, though)

Tutoring and training

- A first simple use case: music lessons
  - Can be 1-to-1, or 1-to-many
  - In both cases, the closest to a "traditional" scenario
- It can basically be treated as a generic videocall or videoconference
  - Most of the interaction is conversational
  - No real need for plugging instruments directly in the call
    - A regular mic is more than enough in this context
- My sister uses Skype to teach her students...
  - ... so why not WebRTC? 🙂

Cool add-on: pitch detection?

- https://github.com/720kb/TeachMusicRTC (last updated 6 years ago, though)

Tutoring and training

- A first simple use case: music lessons
  - Can be 1-to-1, or 1-to-many
  - In both cases, the closest to a "traditional" scenario
- It can basically be treated as a generic videocall or videoconference
  - Most of the interaction is conversational
  - No real need for plugging instruments directly in the call
    - A regular mic is more than enough in this context
- My sister uses Skype to teach her students...
  - ... so why not WebRTC? 🙂

Cool add-on: pitch detection?

- https://github.com/720kb/TeachMusicRTC (last updated 6 years ago, though)

Broadcasting concerts

- A more interesting scenario: broadcasting concerts
  - Basically a 1-to-many (maybe few-to-many?) streaming session
  - Still traditional, if you will, but with a few caveats
- The audio source better not be the browser
  - Browsers mess with the captured audio a lot (e.g., AEC, AGC, etc.)
  - You want the broadcasted audio to be as close as possible to what was captured
    - OBS-WebRTC (via WHIP) or the Janus Streaming plugin (*wink wink*!) can help here
- This scenario is commonly done with HLS, today...
  - ... but you may want less delay and/or a way to interact with the audience!

Broadcasting concerts

- A more interesting scenario: broadcasting concerts
  - Basically a 1-to-many (maybe few-to-many?) streaming session
  - Still traditional, if you will, but with a few caveats
- The audio source better not be the browser
  - Browsers mess with the captured audio a lot (e.g., AEC, AGC, etc.)
  - You want the broadcasted audio to be as close as possible to what was captured
    - OBS-WebRTC (via WHIP) or the Janus Streaming plugin (*wink wink*!) can help here
- This scenario is commonly done with HLS, today...
  - ... but you may want less delay and/or a way to interact with the audience!

Broadcasting concerts

FOSDEM 21

- A more interesting scenario: broadcasting concerts
  - Basically a 1-to-many (maybe few-to-many?) streaming session
  - Still traditional, if you will, but with a few caveats
- The audio source better not be the browser
  - Browsers mess with the captured audio a lot (e.g., AEC, AGC, etc.)
  - You want the broadcasted audio to be as close as possible to what was captured
    - OBS-WebRTC (via WHIP) or the Janus Streaming plugin (*wink wink*!) can help here
- This scenario is commonly done with HLS, today...
  - ... but you may want less delay and/or a way to interact with the audience!

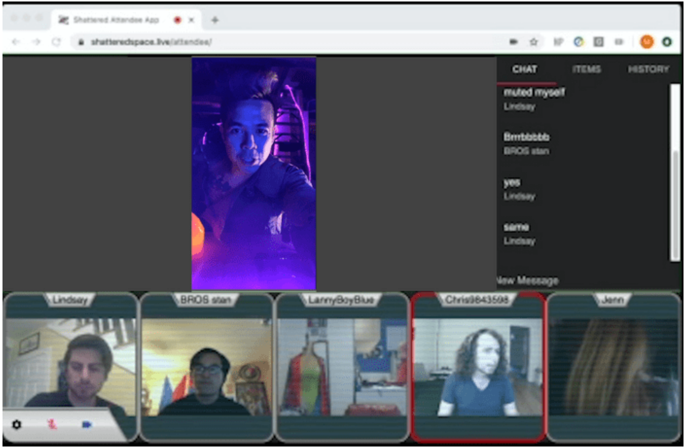
About interacting with the audience... ☺

About interacting with the audience... ☺

FOSDEM 21

Not really music, but close enough!

FOSDEM 21



https://chrisuehlinger.com/blog/2020/06/16/unshattering-the-audience-building-theatre-on-the-web-in-2020

Playing with music and WebRTC

- Many cool things that can be done with WebRTC
  - e.g., browser or native app as an UI to a remote music setup
- Several more or less basic use cases come to mind
  - Writing music in a browser
  - Interaction with (remote) instruments
  - Visual synchronization of music data
  - Integration in (remote) DAW
  - Distributed jam sessions
  - ...

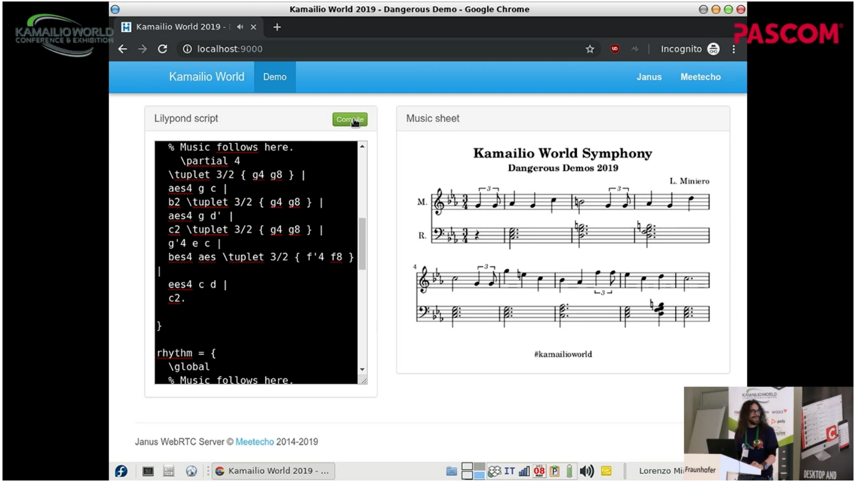
Playing with music and WebRTC

FOSDEM 21

- Many cool things that can be done with WebRTC
  - e.g., browser or native app as an UI to a remote music setup
- Several more or less basic use cases come to mind
  - Writing music in a browser
  - Interaction with (remote) instruments
  - Visual synchronization of music data
  - Integration in (remote) DAW
  - Distributed jam sessions
  - ...

A silly approach at online composition!

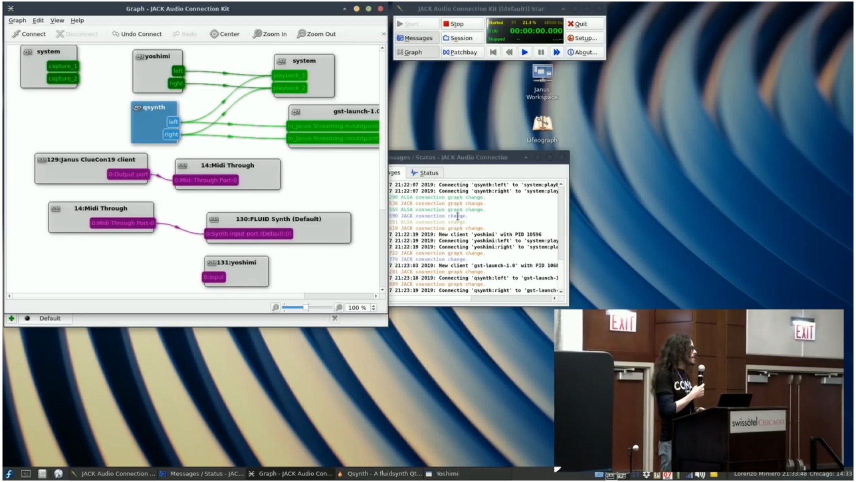
FOSDEM 21



https://youtu.be/d1hOR27r4uY?t=1158

Playing a keyboard with other people

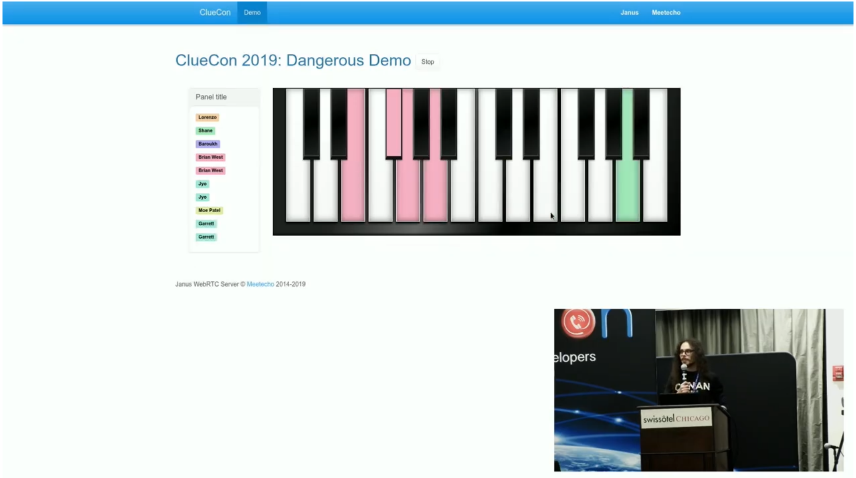
FOSDEM 21



https://youtu.be/8Hzg4hSJMsQ?t=790

Playing a keyboard with other people

FOSDEM 21



https://youtu.be/8Hzg4hSJMsQ?t=790

Web MIDI API + Insertable Streams

FOSDEM 21

https://twitter.com/komasshu/status/1329785808446836736

Jam sessions

- What about *really* playing with other people, though?
  - Harder to do because of this ugly pandemic...
- Only *apparently* a traditional use case
  - Yes, we can see it as a "conference" of sorts...
  - ... but we're not really talking, and latency is much more important
- Browsers are not a good option, here
  - Pipeline may be good for voice, but latency too high for live music
    - Unfortunately, on Linux they don't support Jack, only Pulseaudio
  - Hard to capture anything else than a microphone
    - Besides, as we said they'll mess with the source audio anyway

Jam sessions

- What about *really* playing with other people, though?
  - Harder to do because of this ugly pandemic...
- Only *apparently* a traditional use case
  - Yes, we can see it as a “conference” of sorts...
  - ... but we’re not really talking, and latency is much more important
- Browsers are not a good option, here
  - Pipeline may be good for voice, but latency too high for live music
    - Unfortunately, on Linux they don’t support Jack, only Pulseaudio
  - Hard to capture anything else than a microphone
    - Besides, as we said they’ll mess with the source audio anyway

Jam sessions

FOSDEM 21

- What about *really* playing with other people, though?
  - Harder to do because of this ugly pandemic...
- Only *apparently* a traditional use case
  - Yes, we can see it as a “conference” of sorts...
  - ... but we’re not really <u>talking</u>, and latency is much more important
- Browsers are not a good option, here
  - Pipeline may be good for voice, but latency too high for live music
    - Unfortunately, on Linux they don’t support Jack, only Pulseaudio 😵
  - Hard to capture anything else than a microphone
    - Besides, as we said they’ll mess with the source audio anyway

What about a native approach?

FOSDEM 21

- A few, non-WebRTC, native solutions exist already
  - e.g., Jamulus and NINJAM (both open source)
- It might be interesting to experiment with WebRTC as well
  - e.g., Native client that uses Jack for audio input/output
  - WebRTC exchange of live streams (P2P or via a server)

Idea for a personal fun/pet project of mine

- Native application based on GStreamer
- Ability to add local instruments, captured via Jack and encoded with Opus
- Janus as the reference WebRTC server for all the jam session “participants”
  - Publishing local instruments, subscribing to remote ones

What about a native approach?

- A few, non-WebRTC, native solutions exist already
  - e.g., Jamulus and NINJAM (both open source)
- It might be interesting to experiment with WebRTC as well
  - e.g., Native client that uses Jack for audio input/output
  - WebRTC exchange of live streams (P2P or via a server)

Idea for a personal fun/pet project of mine

- Native application based on GStreamer
- Ability to add local instruments, captured via Jack and encoded with Opus
- Janus as the reference WebRTC server for all the jam session “participants”
  - Publishing local instruments, subscribing to remote ones

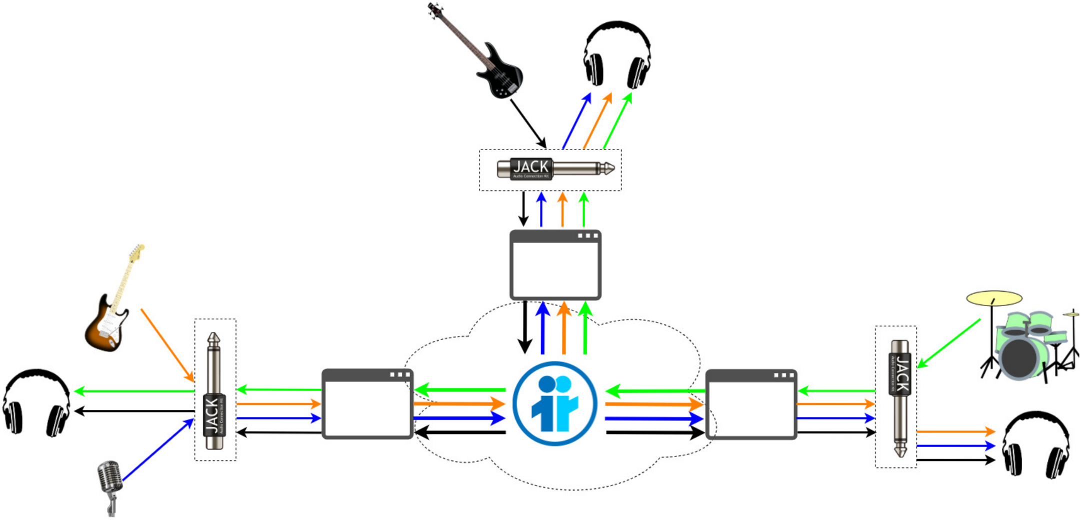
What about a native approach?

- A few, non-WebRTC, native solutions exist already
  - e.g., Jamulus and NINJAM (both open source)
- It might be interesting to experiment with WebRTC as well
  - e.g., Native client that uses Jack for audio input/output
  - WebRTC exchange of live streams (P2P or via a server)

Idea for a personal fun/pet project of mine

- Native application based on GStreamer
- Ability to add local instruments, captured via Jack and encoded with Opus
- Janus as the reference WebRTC server for all the jam session “participants”
  - Publishing local instruments, subscribing to remote ones

The ugliest diagram you'll see today



A distributed/remote concert

FOSDEM 21

- Whatever the approach, it might be useful to stream this remote session
  - A truly distributed concert!
- If the session can be captured, it can be broadcast
  - If an SFU is used, streams can be relayed as-is (they're already there)
    - Basically a few-to-many conferencing session
  - Audio can also be mixed, though, either on the server or client side
    - Many already use OBS for that, so OBS-WebRTC (WHIP) may be a simple option
    - Server-side mixing may be more "balanced" in terms of delay, though?
- In general, same considerations made before on broadcasting apply here too

A distributed/remote concert

- Whatever the approach, it might be useful to stream this remote session
  - A truly distributed concert!
- If the session can be captured, it can be broadcast
  - If an SFU is used, streams can be relayed as-is (they're already there)
    - Basically a few-to-many conferencing session
  - Audio can also be mixed, though, either on the server or client side
    - Many already use OBS for that, so OBS-WebRTC (WHIP) may be a simple option
    - Server-side mixing may be more "balanced" in terms of delay, though?
- In general, same considerations made before on broadcasting apply here too

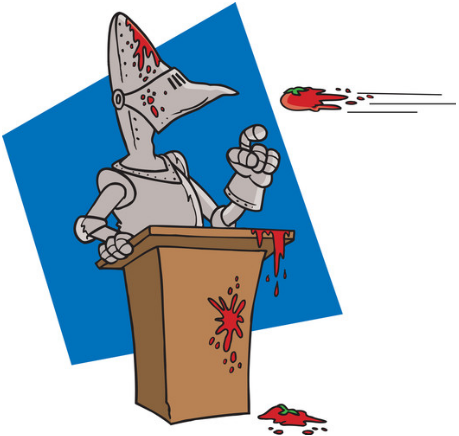
A distributed/remote concert

FOSDEM 21

- Whatever the approach, it might be useful to stream this remote session
  - A truly distributed concert!
- If the session can be captured, it can be broadcast
  - If an SFU is used, streams can be relayed as-is (they're already there)
    - Basically a few-to-many conferencing session
  - Audio can also be mixed, though, either on the server or client side
    - Many already use OBS for that, so OBS-WebRTC (WHIP) may be a simple option
    - Server-side mixing may be more “balanced” in terms of delay, though?
- In general, same considerations made before on broadcasting apply here too

Thanks! Questions? Comments?

FOSDEM 21



Get in touch!

- https://twitter.com/elminiero
- https://twitter.com/meetecho
- https://www.meetecho.com