Implementation Experience
WHIP in Janus (and GStreamer)

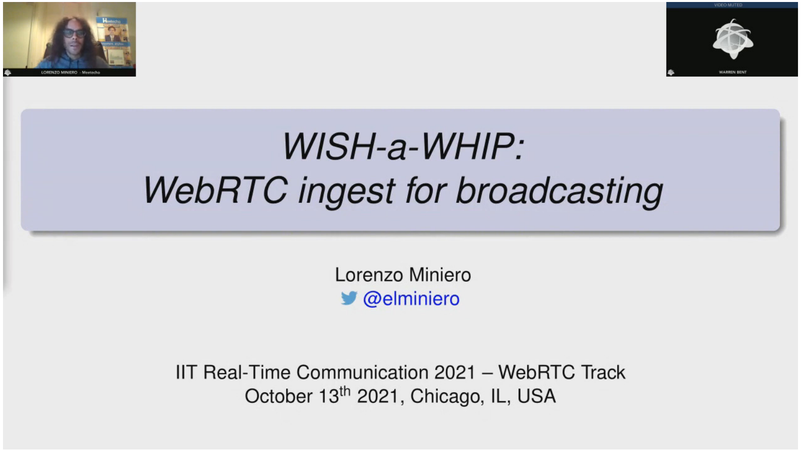
Lorenzo Miniero
lorenzo@meetecho.com

IETF 113, WISH WG
March 21st, 2022

Detailed blog post on integration

https://www.meetecho.com/blog/whip-janus-part-ii/

IIT RTC 2021 Presentation



WISH-a-WHIP:
WebRTC ingest for broadcasting

Lorenzo Miniero
@elminiero

IIT Real-Time Communication 2021 – WebRTC Track
October 13th 2021, Chicago, IL, USA

https://www.youtube.com/watch?v=b_QBd3WnGgY

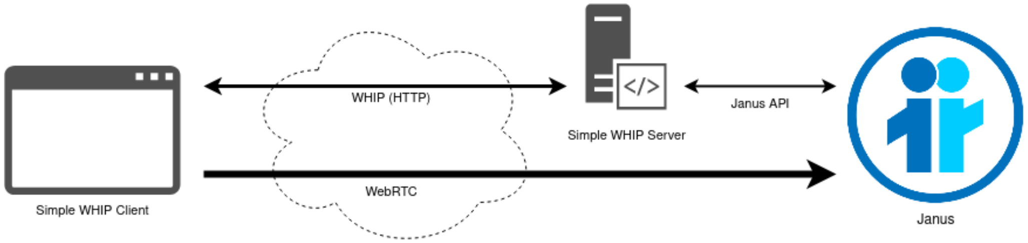
A WHIP server based on Janus

- Janus is a popular WebRTC server, so a good option for WHIP
  - It implements its own JSON-based API, though (Janus API)
- Simple (and transparent) solution: basic API translator in front of Janus
  - WHIP API maps quite simply to set of Janus API primitives
  - No need to change anything in the WebRTC stack
- Implemented simple prototype using node.js and Express
  - REST server that implements the WHIP API, and talks to Janus accordingly
  - Only takes care of ingest: distribution out of scope

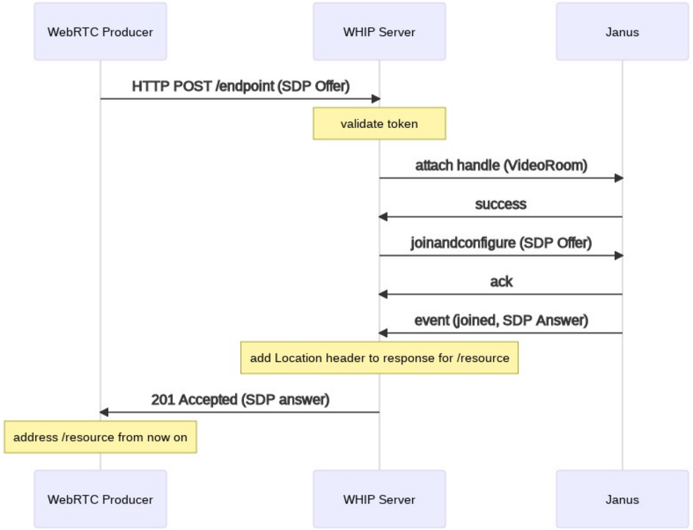
Simple WHIP Server

https://github.com/meetecho/simple-whip-server/

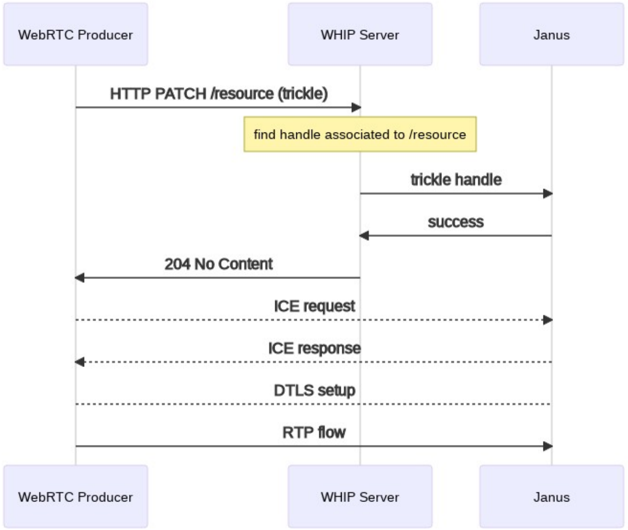
WHIP client and server + Janus



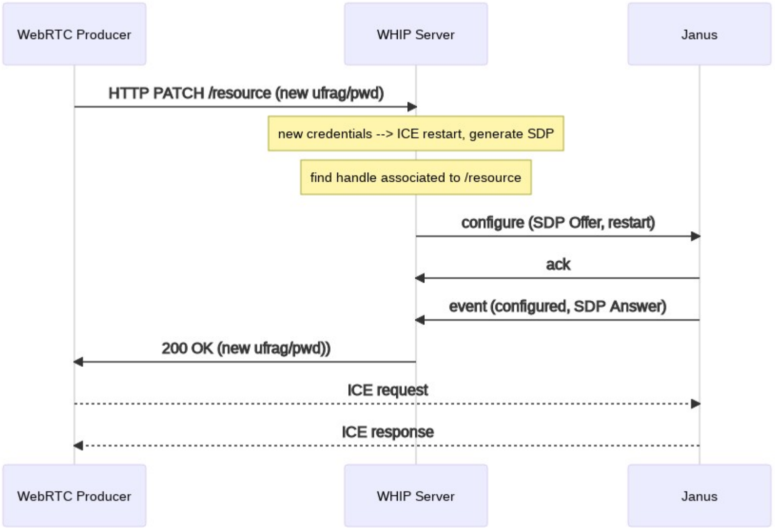
Mapping WHIP interactions to the Janus API


WebRTC Producer
WHIP Server
Janus
HTTP POST /endpoint (SDP Offer)
validate token
attach handle (VideoRoom)
success
joinandconfigure (SDP Offer)
ack
event (joined, SDP Answer)
add Location header to response for /resource
201 Accepted (SDP answer)
address /resource from now on
WebRTC Producer
WHIP Server
Janus

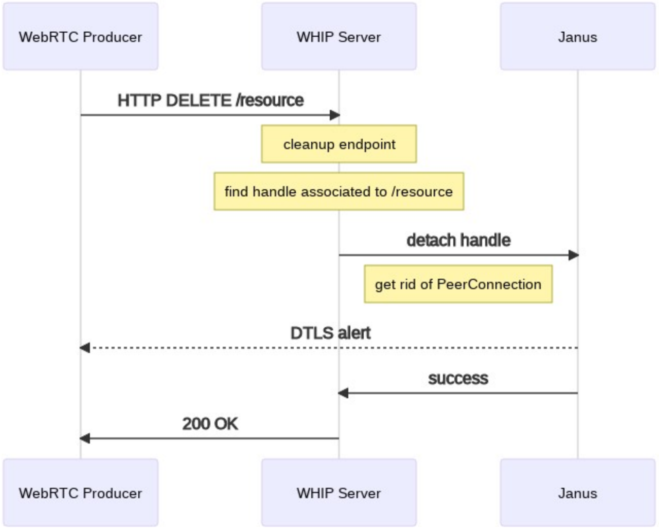
Mapping WHIP interactions to the Janus API



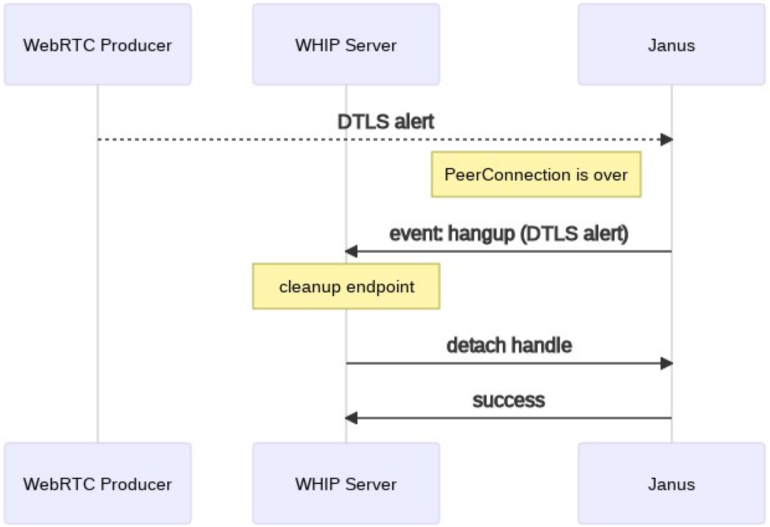
Mapping WHIP interactions to the Janus API


WebRTC Producer
WHIP Server
Janus
HTTP PATCH /resource (new ufrag/pwd)
new credentials --> ICE restart, generate SDP
find handle associated to /resource
configure (SDP Offer, restart)
ack
event (configured, SDP Answer)
200 OK (new ufrag/pwd))
ICE request
ICE response
WebRTC Producer
WHIP Server
Janus

Mapping WHIP interactions to the Janus API



Mapping WHIP interactions to the Janus API



Writing a WHIP client for testing

- Needs to support HTTP (WHIP API) and have a WebRTC stack
  - Browsers are the obvious choice, but what about a native solution?
  - Many broadcasters today use custom tools (e.g., OBS)
- Unfortunately OBS-WebRTC is not currently an option
  - Used legacy WHIP API, and currently only supports Millicast ingestion
- Chose GStreamer's **`webrtcbin`**[1] for the purpose
  - Used it already with success in other applications (e.g., JamRTC)
  - Modular and very powerful, so easy to feed with external sources

Simple WHIP Client

https://github.com/meetecho/simple-whip-client/

[1] https://gstreamer.freedesktop.org/documentation/webrtc/

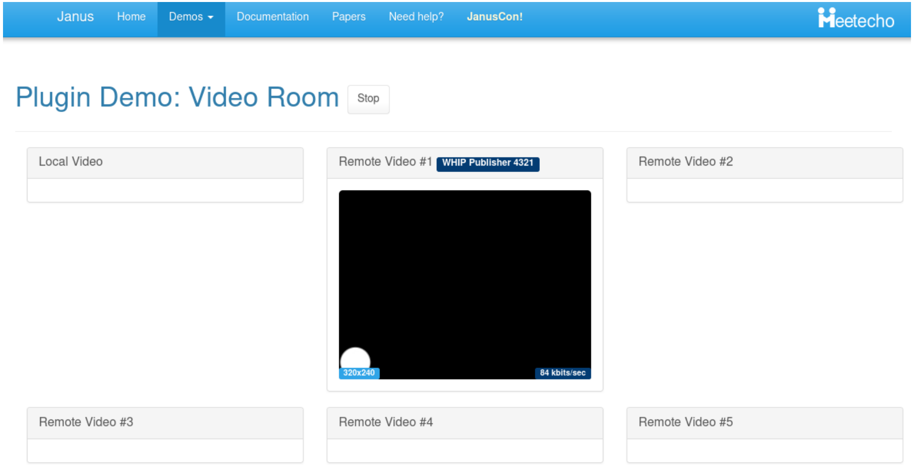
WHIP client features (and limitations)

- Almost everything supported
  - Trickle (PATCH), STUN/TURN (via `Link` too), tokens, DELETE, etc.
  - Added support for non-trickle too (manual addition of candidates to SDP)
  - Option to force TURN (`iceTransportPolicy:"relay"` equivalent)
- Customizable audio/video pipelines
  - Easy to experiment with different sources and codecs
- A couple of things not supported in `webrtcbin` yet, though
  - ICE restarts (there seems to be a PR, though)
  - `Link` support in POST (we currently only do it via OPTIONS in the client)
- Only supports Linux at the moment (help to port to Windows/MacOS welcome!)

Simple WHIP Client example

```
./whip-client -u
   https://mercury.conf.meetecho.com:8443/whip/endpoint/test \
   -t hackathon \
   -A "audiotestsrc is-live=true wave=red-noise ! audioconvert !
      audioresample ! queue ! opusenc ! rtpopuspay pt=100 ! queue !
      application/x-rtp,media=audio,encoding-name=OPUS,payload=100" \
   -V "videotestsrc is-live=true pattern=ball ! videoconvert ! queue !
      vp8enc deadline=1 ! rtpvp8pay pt=96 ! queue !
      application/x-rtp,media=video,encoding-name=VP8,payload=96" \
   -S stun.l.google.com:19302
```

Simple WHIP Client example

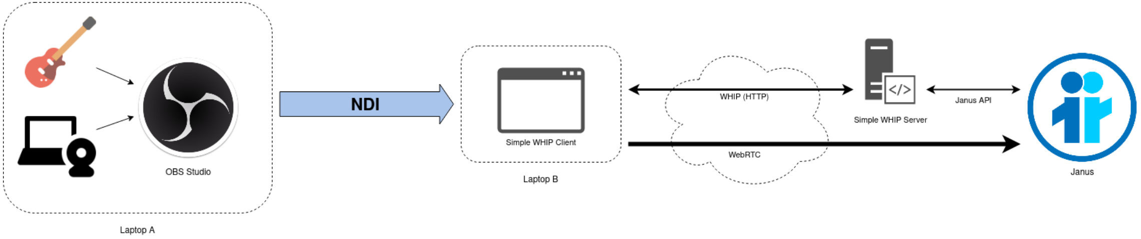


https://janus-legacy.conf.meetecho.com/videoroomtest.html?room=4321&subscriber-mode=true

Experimenting with ETags

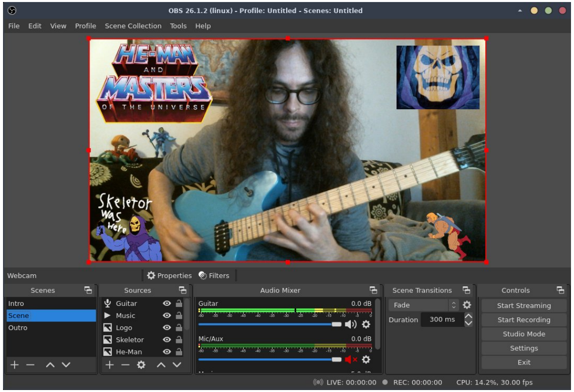
- Implemented in both server and client as pull requests
  - https://github.com/meetecho/simple-whip-server/pull/4
  - https://github.com/meetecho/simple-whip-client/pull/9
- **Server**
  - Generates (and returns) random ETag in response to POST and PATCH restarts
  - Expects ETag, and right one, when receiving trickle candidates (412 otherwise)
  - Expects `"*"` when receiving restarts, returns error otherwise (is this wrong?)
- **Client**
  - If an ETag is found in response to POST, it's used for PATCH requests
  - Client doesn't support restarts yet, so related ETag part is missing

Having fun with WHIP ☺



https://2021.commcon.xyz/talks/whip-ndi-and-janus-genesis-of-a-broadcasting-demo
https://fosdem.org/2022/schedule/event/rtc_whip/

Having fun with WHIP ☺



https://2021.commcon.xyz/talks/whip-ndi-and-janus-genesis-of-a-broadcasting-demo
https://fosdem.org/2022/schedule/event/rtc_whip/