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/
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

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
Mapping WHIP interactions to the Janus API

HTTP PATCH /resource (trickle)

find handle associated to /resource

trickle handle

204 No Content

success

ICE request

ICE response

DTLS setup

RTP flow
Mapping WHIP interactions to the Janus API

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
Mapping WHIP interactions to the Janus API

WebRTC Producer → WHIP Server

HTTP DELETE /resource

cleanup endpoint

find handle associated to /resource

detach handle

DTLS alert

get rid of PeerConnection

success

200 OK

WebRTC Producer → WHIP Server → Janus
Mapping WHIP interactions to the Janus API

Diagram:

- WebRTC Producer
- WHIP Server
- Janus

Flow:

1. WebRTC Producer
2. WHIP Server
3. Janus

- DTLS alert
  - PeerConnection is over
  - event: hangup (DTLS alert)
- cleanup endpoint
- detach handle
- success
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

```bash
./whip-client -u
https://mercury.conf.meetecho.com:8443/whip/endpoint/test \
-t hackathon \n-A "audiotests src 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 "videotests src 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
```
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://fosdem.org/2022/schedule/event/rtc_whip/
Having fun with WHIP 😊

https://fosdem.org/2022/schedule/event/rtc_whip/