WISH-a-WHIP: WebRTC ingest for broadcasting

Lorenzo Miniero
@elminiero

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

A few words about me

Lorenzo Miniero

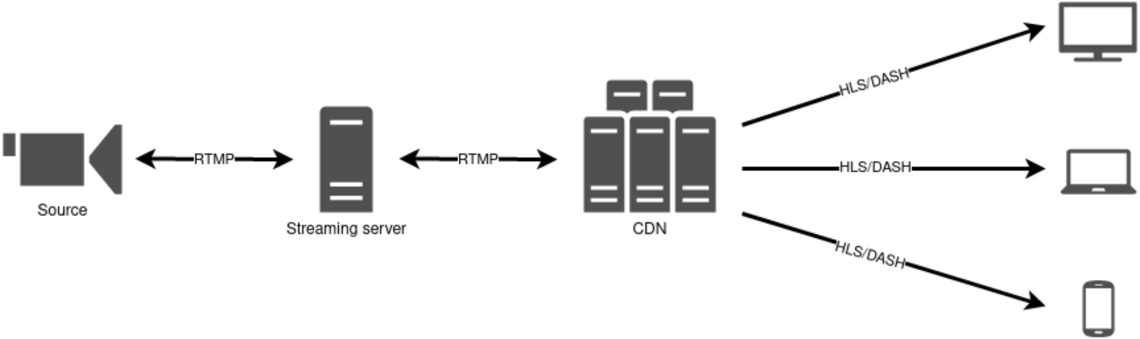
- Ph.D @ UniNA
- Chairman @ Meetecho
- Main author of Janus®

Contacts and info

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

There would be no WHIP without Dr. Alex ♥

Traditional broadcasting



Why not WebRTC?

- Traditional broadcasting efficient but higher latency
    - At best (live), delay will typically be in the range of a few seconds
- WebRTC natively conceived for very low latency, instead
    - Born for conversational audio/video/data
    - Can be (and often is) easily used for monodirectional streaming as well
- Strangely not really considered by the industry up until recently, though
    - Topic of my Ph.D years ago ("Streaming Of Large scale Events over Internet cLouds")
    - Clearing the industry FUD: https://webrtcbydralex.com/index.php/2020/04/14/
- Tooling an important aspect to foster WebRTC adoption, here
    - e.g., a standard way to send media, and tools à la OBS

Why not WebRTC?

- Traditional broadcasting efficient but higher latency
  - At best (live), delay will typically be in the range of a few seconds
- WebRTC natively conceived for very low latency, instead
  - Born for conversational audio/video/data
  - Can be (and often is) easily used for monodirectional streaming as well
- Strangely not really considered by the industry up until recently, though
  - Topic of my Ph.D years ago ("Streaming Of Large scale Events over Internet cLouds")
  - Clearing the industry FUD: https://webrtcbydralex.com/index.php/2020/04/14/
- Tooling an important aspect to foster WebRTC adoption, here
  - e.g., a standard way to send media, and tools à la OBS

Why not WebRTC?

- Traditional broadcasting efficient but higher latency
  - At best (live), delay will typically be in the range of a few seconds
- WebRTC natively conceived for very low latency, instead
  - Born for conversational audio/video/data
  - Can be (and often is) easily used for monodirectional streaming as well
- Strangely not really considered by the industry up until recently, though
  - Topic of my Ph.D years ago (“Streaming Of Large scale Events over Internet cLouds”)
  - Clearing the industry FUD: https://webrtcbydralex.com/index.php/2020/04/14/
- Tooling an important aspect to foster WebRTC adoption, here
  - e.g., a standard way to send media, and tools à la OBS

Why not WebRTC?

- Traditional broadcasting efficient but higher latency
  - At best (live), delay will typically be in the range of a few seconds
- WebRTC natively conceived for very low latency, instead
  - Born for conversational audio/video/data
  - Can be (and often is) easily used for monodirectional streaming as well
- Strangely not really considered by the industry up until recently, though
  - Topic of my Ph.D years ago (“Streaming Of Large scale Events over Internet cLouds”)
  - Clearing the industry FUD: https://webrtcbydralex.com/index.php/2020/04/14/
- Tooling an important aspect to foster WebRTC adoption, here
  - e.g., a standard way to send media, and tools à la OBS

Making WebRTC ingestion easy: WHIP!

https://www.meetecho.com/blog/whip-janus/ (September, 2020)

A new Working Group in the IETF...

IETF Datatracker Groups Documents Meetings Other User Document search

WebRTC Ingest Signaling over HTTPS (wish)

About | Documents | Meetings | History | Photos | Email expansions | List archive » | Tools »

WG	**Name**	WebRTC Ingest Signaling over HTTPS
	Acronym	wish
	Area	Applications and Real-Time Area (art)
	State	Active
	Charter	charter-ietf-wish-01 Approved
	Dependencies	Document dependency graph (SVG)
	Additional Resources	- Github
Personnel	**Chairs**	Nils Ohlmeier Sean Turner
	Area Director	Murray Kucherawy
Mailing list	**Address**	wish@ietf.org
	To subscribe	https://www.ietf.org/mailman/listinfo/wish
	Archive	https://mailarchive.ietf.org/arch/browse/wish/
Jabber chat	**Room address**	xmpp:wish@jabber.ietf.org?join
	Logs	https://jabber.ietf.org/logs/wish/

Charter for Working Group

The WISH working group is chartered to specify a simple, extensible, HTTPS-based signaling protocol to establish one-way WebRTC-based audiovisual sessions between broadcasting tools and real-time media broadcast networks.

https://datatracker.ietf.org/wg/wish/about/

... and a new draft for the WHIP specification!

Workgroup:	wish
Internet-Draft:	draft-ietf-wish-whip-00
Published:	22 August 2021
Intended Status:	Standards Track
Expires:	23 February 2022
Authors:	S. Murillo *CoSMo Software* A. Gouaillard *CoSMo Software*

WebRTC-HTTP ingestion protocol (WHIP)

Abstract

While WebRTC has been very successful in a wide range of scenarios, its adoption in the broadcasting/streaming industry is lagging behind. Currently there is no standard protocol (like SIP or RTSP) designed for ingesting media in a streaming service, and content providers still rely heavily on protocols like RTMP for it.

These protocols are much older than webrtc and lack by default some important security and resilience features provided by webrtc with minimal delay.

The media codecs used in older protocols do not always match those being used in WebRTC, mandating transcoding on the ingest node, introducing delay and degrading media quality. This transcoding step is always present in traditional streaming to support e.g. ABR, and comes at no cost. However webrtc implements client-side ABR, also called Network-Aware Encoding by e.g. Huavision, by means of simulcast and SVC codecs, which otherwise alleviate the need for server-side transcoding. Content protection and Privacy Enhancement can be achieved with End-to-End Encryption, which preclude any server-side media processing.

This document proposes a simple HTTP based protocol that will allow WebRTC endpoints to ingest content into streaming services and/or CDNs to fill this gap and facilitate deployment.

Table of Contents ▲

https://www.ietf.org/archive/id/draft-ietf-wish-whip-00.html

WebRTC-HTTP ingestion protocol (WHIP)

- HTTP-based signalling to create `sendonly` PeerConnections
  - HTTP POST to send SDP offer, and get an SDP answer in the response
  - Teardown of sessions using HTTP DELETE
- Authentication and authorization via Bearer tokens
  - https://www.rfc-editor.org/rfc/rfc6750.html
- Trickle and ICE restart via HTTP PATCH and SDP fragments
  - https://www.rfc-editor.org/rfc/rfc8840.html
- Everything else is your usual WebRTC!
  - ICE, DTLS, etc.

WebRTC-HTTP ingestion protocol (WHIP)

- HTTP-based signalling to create **`sendonly`** PeerConnections
  - HTTP POST to send SDP offer, and get an SDP answer in the response
  - Teardown of sessions using HTTP DELETE
- Authentication and authorization via Bearer tokens
  - https://www.rfc-editor.org/rfc/rfc6750.html
- Trickle and ICE restart via HTTP PATCH and SDP fragments
  - https://www.rfc-editor.org/rfc/rfc8840.html
- Everything else is your usual WebRTC!
  - ICE, DTLS, etc.

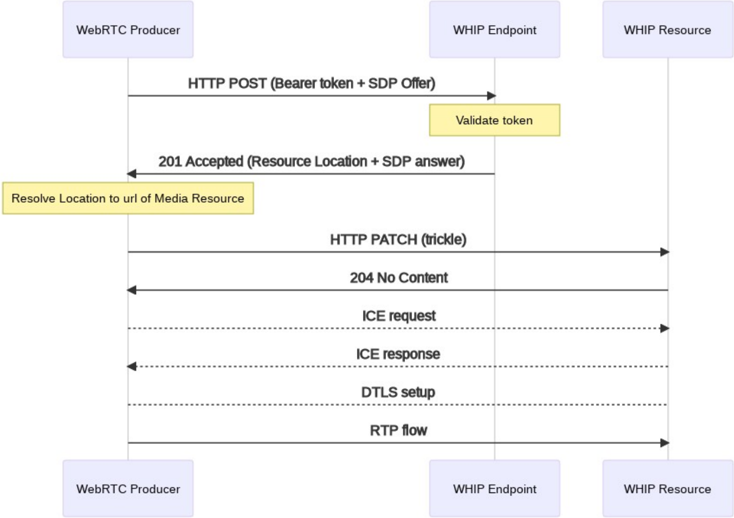
WebRTC-HTTP ingestion protocol (WHIP)

- HTTP-based signalling to create **`sendonly`** PeerConnections
  - HTTP POST to send SDP offer, and get an SDP answer in the response
  - Teardown of sessions using HTTP DELETE
- Authentication and authorization via Bearer tokens
  - https://www.rfc-editor.org/rfc/rfc6750.html
- Trickle and ICE restart via HTTP PATCH and SDP fragments
  - https://www.rfc-editor.org/rfc/rfc8840.html
- Everything else is your usual WebRTC!
  - ICE, DTLS, etc.

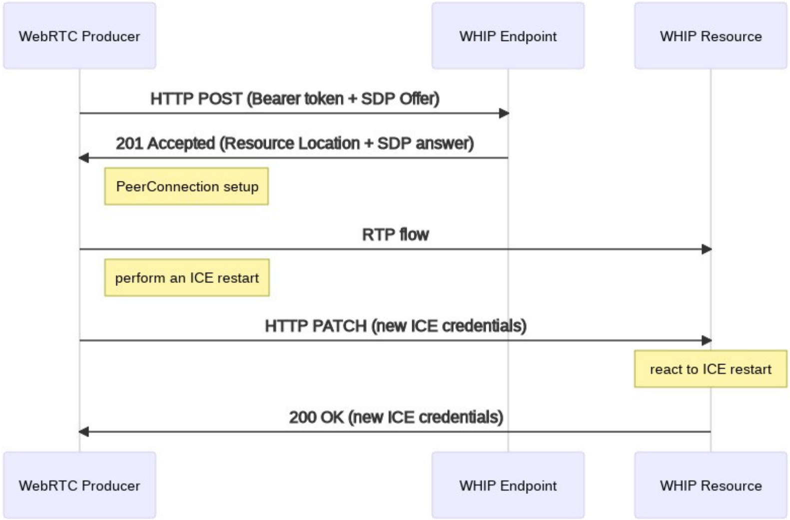
WebRTC-HTTP ingestion protocol (WHIP)

- HTTP-based signalling to create **`sendonly`** PeerConnections
  - HTTP POST to send SDP offer, and get an SDP answer in the response
  - Teardown of sessions using HTTP DELETE
- Authentication and authorization via Bearer tokens
  - https://www.rfc-editor.org/rfc/rfc6750.html
- Trickle and ICE restart via HTTP PATCH and SDP fragments
  - https://www.rfc-editor.org/rfc/rfc8840.html
- Everything else is your usual WebRTC!
  - ICE, DTLS, etc.

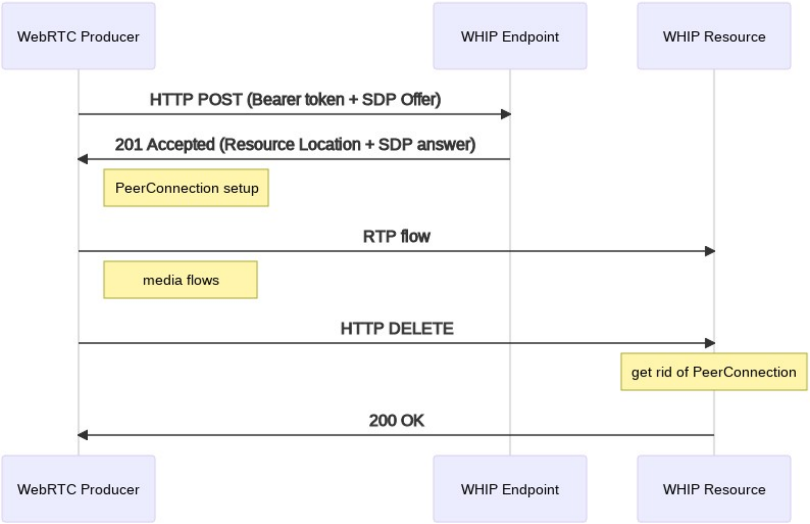
A few sequence diagrams



A few sequence diagrams



A few sequence diagrams



A WHIP server based on Janus

- Janus is a popular WebRTC server, so 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 (e.g., via SOLEIL)

Simple WHIP Server

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

A WHIP server based on Janus

- Janus is a popular WebRTC server, so 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 (e.g., via SOLEIL)

Simple WHIP Server

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

A WHIP server based on Janus

- Janus is a popular WebRTC server, so 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 (e.g., via SOLEIL)

Simple WHIP Server

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

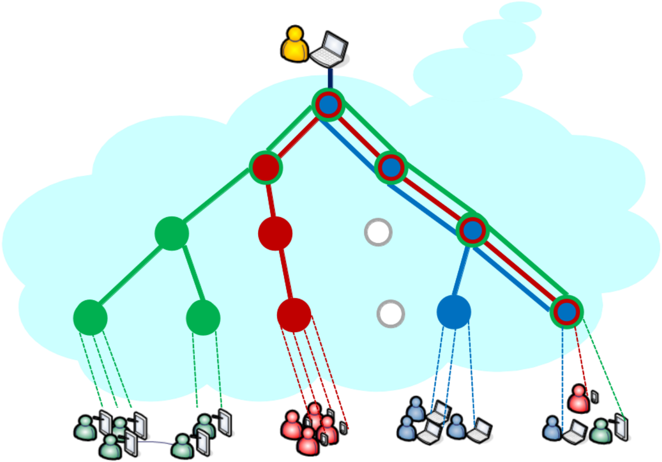
A WHIP server based on Janus

- Janus is a popular WebRTC server, so 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 (e.g., via SOLEIL)

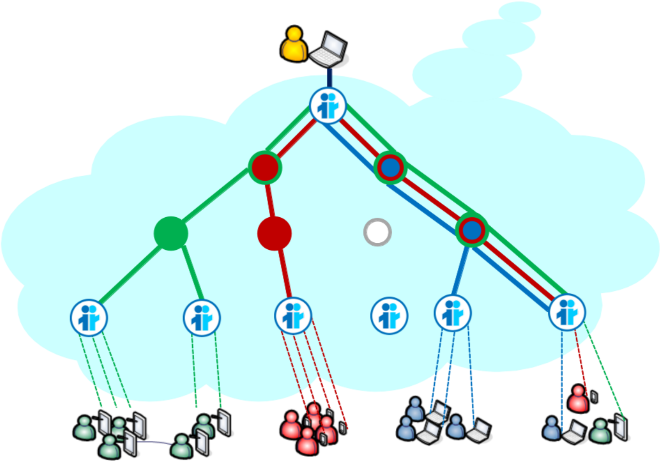
Simple WHIP Server

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

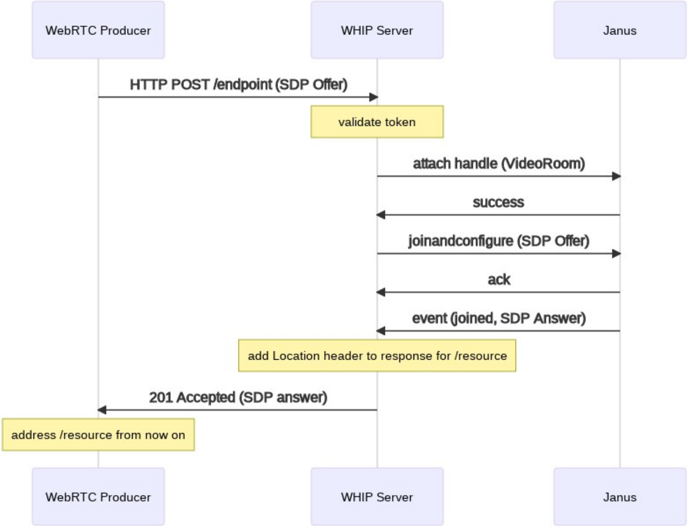
Distributing a WHIP Janus stream: SOLEIL



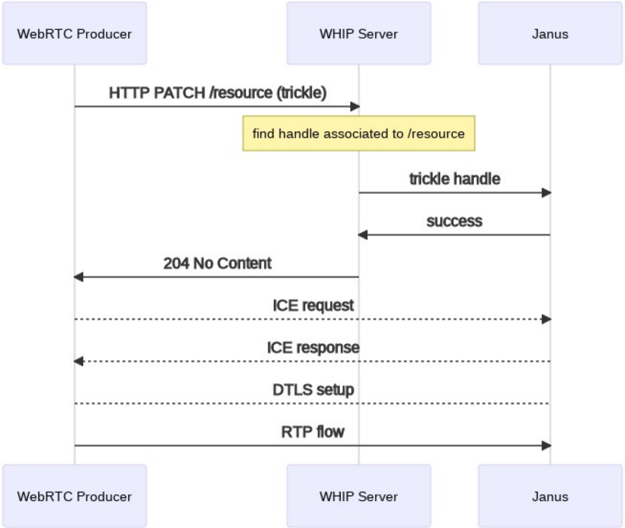
Distributing a WHIP Janus stream: SOLEIL



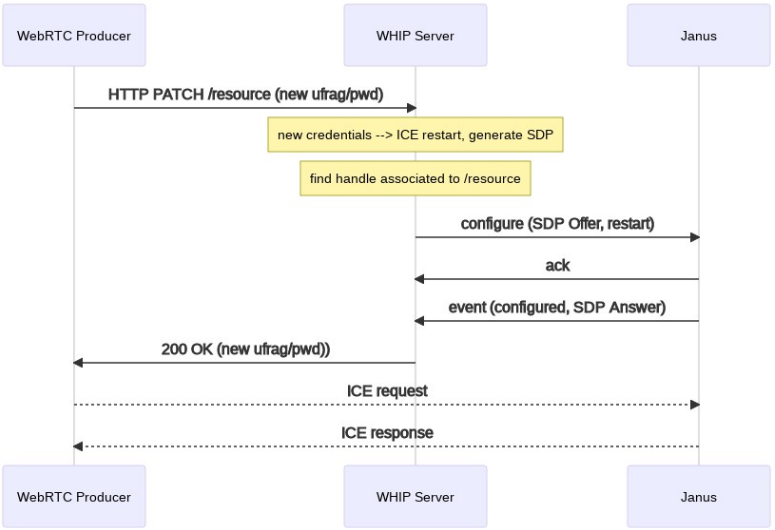
Mapping WHIP interactions to the Janus API



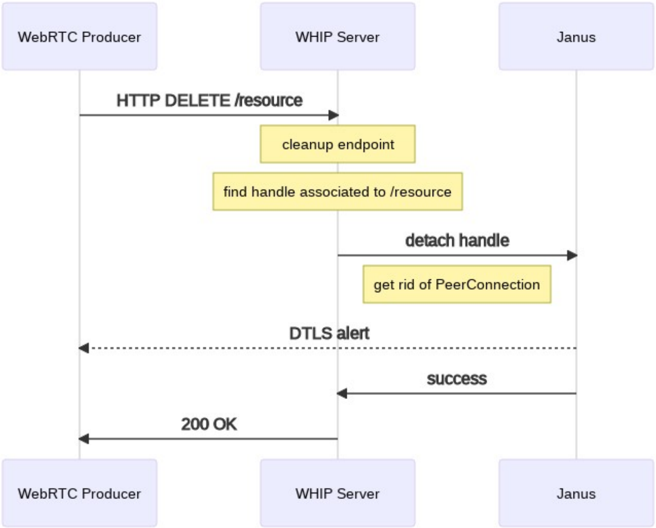
Mapping WHIP interactions to the Janus API



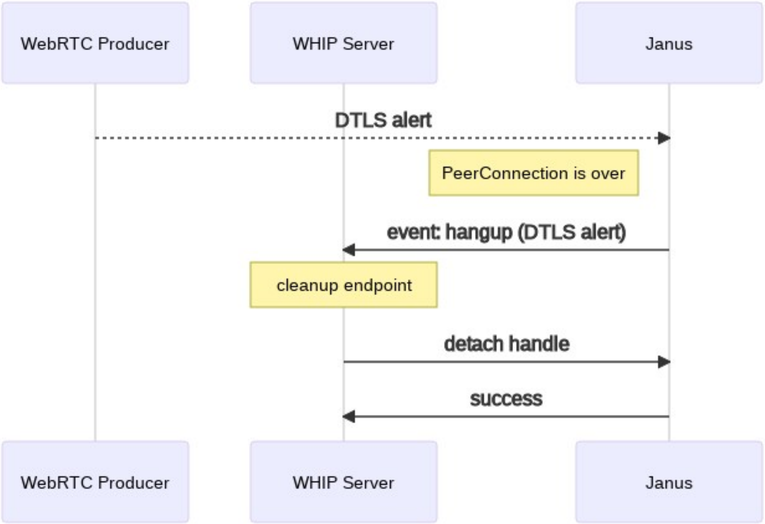
Mapping WHIP interactions to the Janus API



Mapping WHIP interactions to the Janus API



Mapping WHIP interactions to the Janus API



Simple WHIP Server in action 😎

WHIP server

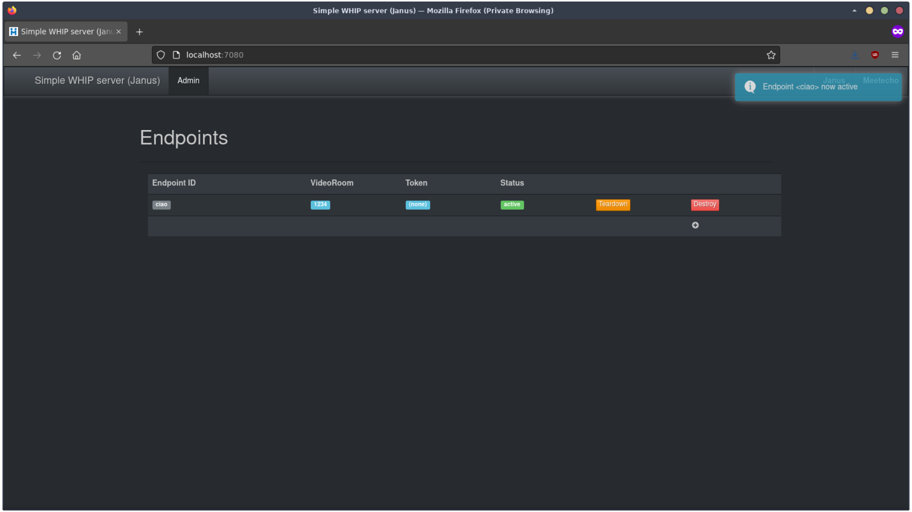
File Edit View Terminal Tabs Help

```
[lminiero@lminiero server]$ npm start

> janus-whip-server@0.0.1 start /home/lminiero/Work/code/services/whip/server
> DEBUG=whip:*,-whip:debug,janus:*,-janus:debug,-janus:vdebug node src/server.js

[1. Janus]
Connecting to Janus: { address: 'ws://127.0.0.1:8188' }
  janus:info Connecting to ws://127.0.0.1:8188 +0ms
  janus:info Janus WebSocket Client Connected +7ms
  janus:info Janus session ID is 1252536092417283 +2ms
  whip:info Connected to Janus: ws://127.0.0.1:8188 +0ms
[2. WHIP REST API]
WHIP REST API listening on *:7080
WHIP server prototype started!
[ 'Janus OK', 'WHIP REST API OK' ]
  whip:info [ciao] Created new WHIP endpoint +32s
  whip:info [ciao] Publishing to WHIP endpoint +5s
  whip:info [ciao] Terminating WHIP session +12s
  whip:info [ciao] Publishing to WHIP endpoint +23s
  whip:info [ciao] PeerConnection detected as closed +8s
```

Basic UI to create/manage endpoints)



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/lminiero/simple-whip-client/

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

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/lminiero/simple-whip-client/

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

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/lminiero/simple-whip-client/

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

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/lminiero/simple-whip-client/

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

Simple WHIP Client options

```
Usage:
  whip-client [OPTION?] -- Simple WHIP client

Help Options:
  -h, --help          Show help options

Application Options:
  -u, --url           Address of the WHIP endpoint (required)
  -t, --token         Authentication Bearer token to use (optional)
  -A, --audio         GStreamer pipeline to use for audio (optional, required if audio-only)
  -V, --video         GStreamer pipeline to use for video (optional, required if video-only)
  -S, --stun-server STUN server to use, if any (hostname:port)
  -T, --turn-server TURN server to use, if any (username:password@host:port)
  -l, --log-level  Logging level (0=disable logging, 7=maximum log level; default: 4)
```

Simple WHIP Client example

```
./whip-client -u http://localhost:7080/whip/endpoint/abc123 \
   -t verysecret \
   -A "audiotestsrc is-live=true wave=red-noise ! audioconvert !
      audioresample ! queue ! opusenc ! rtpopuspay pt=100 ssrc=1 !
      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 ssrc=2 ! queue !
      application/x-rtp,media=video,encoding-name=VP8,payload=96" \
   -S stun.l.google.com:19302
```

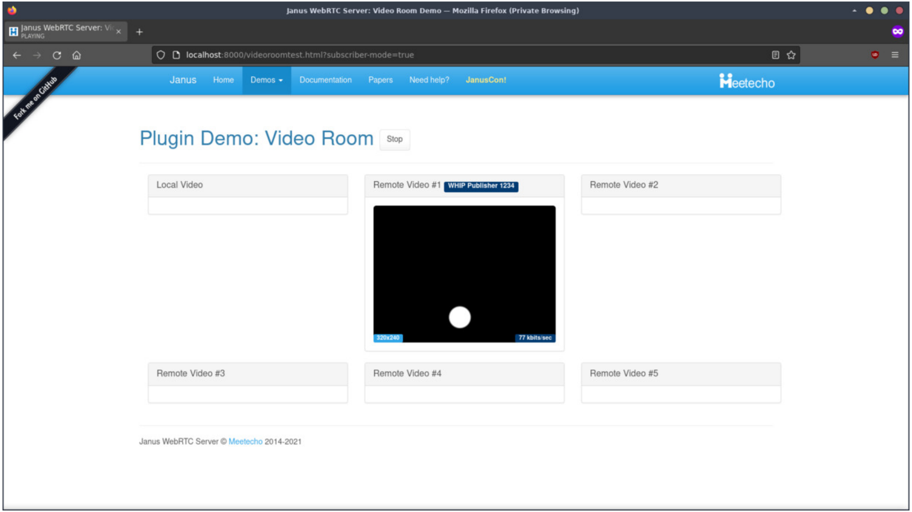
Simple WHIP Client in action 😎

WHIP client

File Edit View Terminal Tabs Help

```
[WHIP] Initializing the GStreamer pipeline:
webrtcbin name=sendonly bundle-policy=3   videotestsrc is-live=true pattern=ball
 ! videoconvert ! queue ! vp8enc deadline=1 ! rtpvp8pay pt=96 ssrc=2 ! queue ! a
pplication/x-rtp,media=video,encoding-name=VP8,payload=96 ! sendonly. audiotests
rc is-live=true wave=red-noise ! audioconvert ! audioresample ! queue ! opusenc
! rtpopuspay pt=100 ssrc=1 ! queue ! application/x-rtp,media=audio,encoding-name
=OPUS,payload=100 ! sendonly.
[WHIP] Starting the GStreamer pipeline
[WHIP] Creating offer
[WHIP] Offer created
[WHIP] Setting local description
[WHIP] Sending SDP offer (1167 bytes)
[WHIP] Resource URL: http://localhost:7080/whip/resource/ciao
[WHIP] Received SDP answer (1385 bytes)
[WHIP] Setting remote description
[WHIP] ICE gathering started...
[WHIP] PeerConnection connecting...
[WHIP] ICE connecting...
[WHIP] ICE completed
[WHIP] DTLS connecting...
[WHIP] DTLS connected
[WHIP] PeerConnection connected
[WHIP] ICE gathering completed
```

Testing my WHIP client with Janus



Other WHIP implementations

- A few other implementations are starting to appear already
  - Very useful for interoperability testing!
- Juliusz Chroboczek
  - WHIP server: https://github.com/jech/galene/tree/whip (Galene integration)
- Sergio Garcia Murillo
  - WHIP client: https://github.com/medooze/whip-js/ (web client)
  - WHIP server: Millicast integration
- Gustavo Garcia
  - WHIP client: https://github.com/ggarber/whip-go (command-line)
- More to come soon, hopefully!

Other WHIP implementations

- A few other implementations are starting to appear already
  - Very useful for interoperability testing!
- Juliusz Chroboczek
  - WHIP server: https://github.com/jech/galene/tree/whip (Galene integration)
- Sergio Garcia Murillo
  - WHIP client: https://github.com/medooze/whip-js/ (web client)
  - WHIP server: Millicast integration
- Gustavo Garcia
  - WHIP client: https://github.com/ggarber/whip-go (command-line)
- More to come soon, hopefully!

Other WHIP implementations

- A few other implementations are starting to appear already
  - Very useful for interoperability testing!
- Juliusz Chroboczek
  - WHIP server: https://github.com/jech/galene/tree/whip (Galene integration)
- Sergio Garcia Murillo
  - WHIP client: https://github.com/medooze/whip-js/ (web client)
  - WHIP server: Millicast integration
- Gustavo Garcia
  - WHIP client: https://github.com/ggarber/whip-go (command-line)
- More to come soon, hopefully!

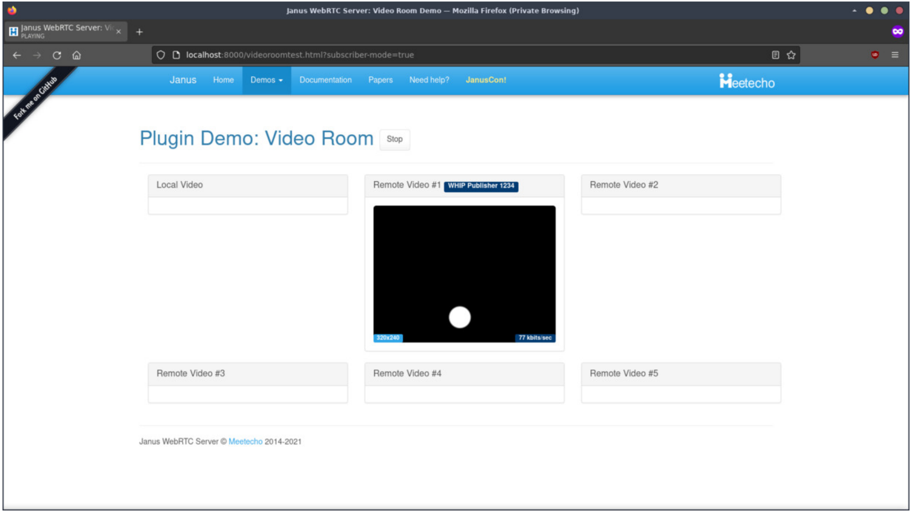
Other WHIP implementations

- A few other implementations are starting to appear already
  - Very useful for interoperability testing!
- Juliusz Chroboczek
  - WHIP server: https://github.com/jech/galene/tree/whip (Galene integration)
- Sergio Garcia Murillo
  - WHIP client: https://github.com/medooze/whip-js/ (web client)
  - WHIP server: Millicast integration
- Gustavo Garcia
  - WHIP client: https://github.com/ggarber/whip-go (command-line)
- More to come soon, hopefully!

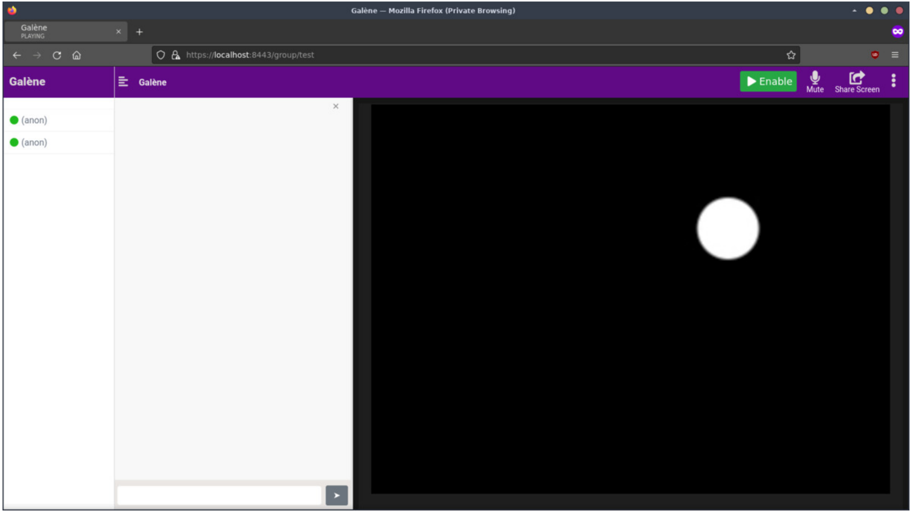
Other WHIP implementations

- A few other implementations are starting to appear already
  - Very useful for interoperability testing!
- Juliusz Chroboczek
  - WHIP server: https://github.com/jech/galene/tree/whip (Galene integration)
- Sergio Garcia Murillo
  - WHIP client: https://github.com/medooze/whip-js/ (web client)
  - WHIP server: Millicast integration
- Gustavo Garcia
  - WHIP client: https://github.com/ggarber/whip-go (command-line)
- More to come soon, hopefully!

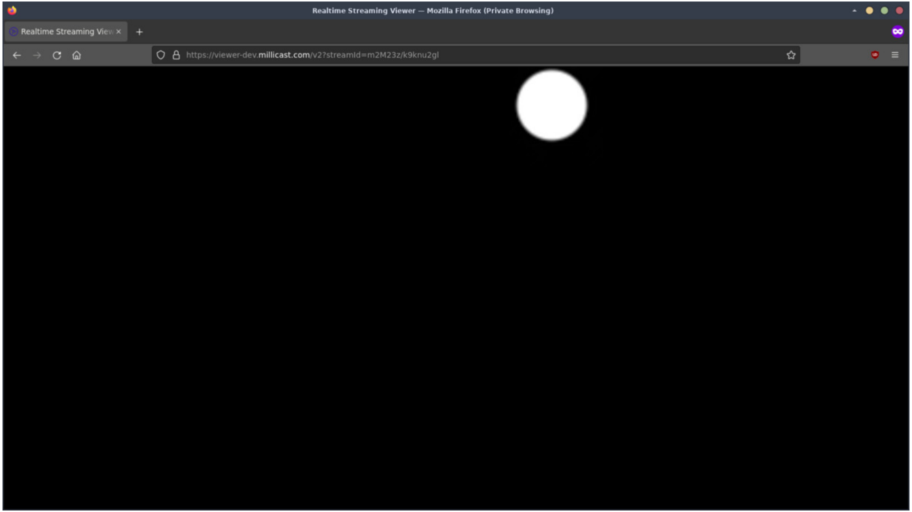
Testing my WHIP client with Janus



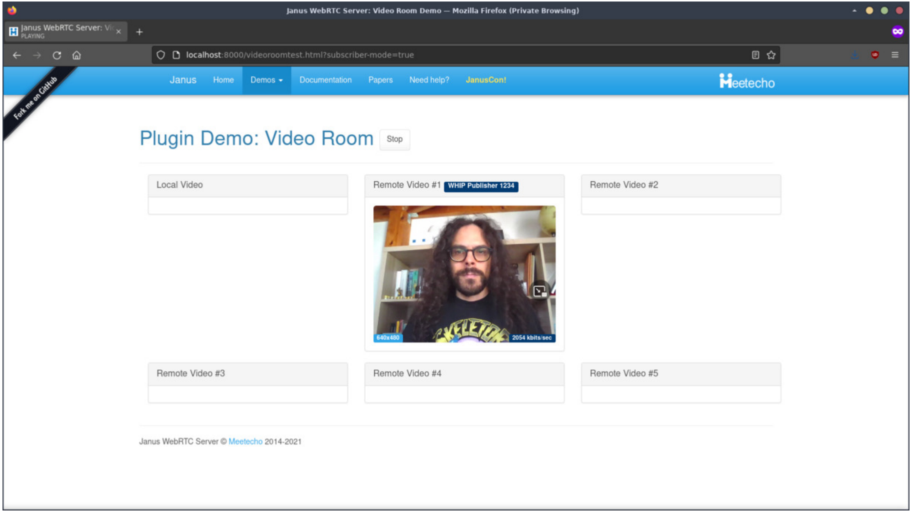
Testing my WHIP client with Galene



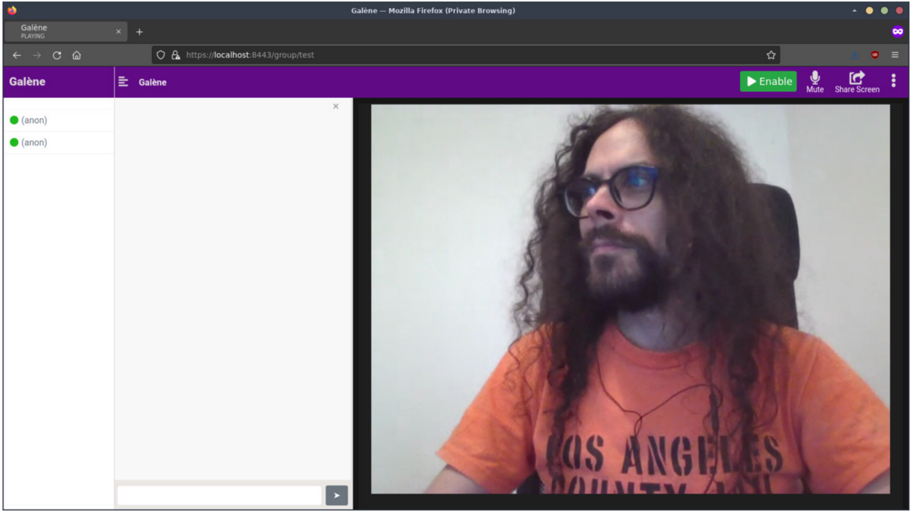
Testing my WHIP client with Millicast



Testing Sergio’s WHIP client with Janus

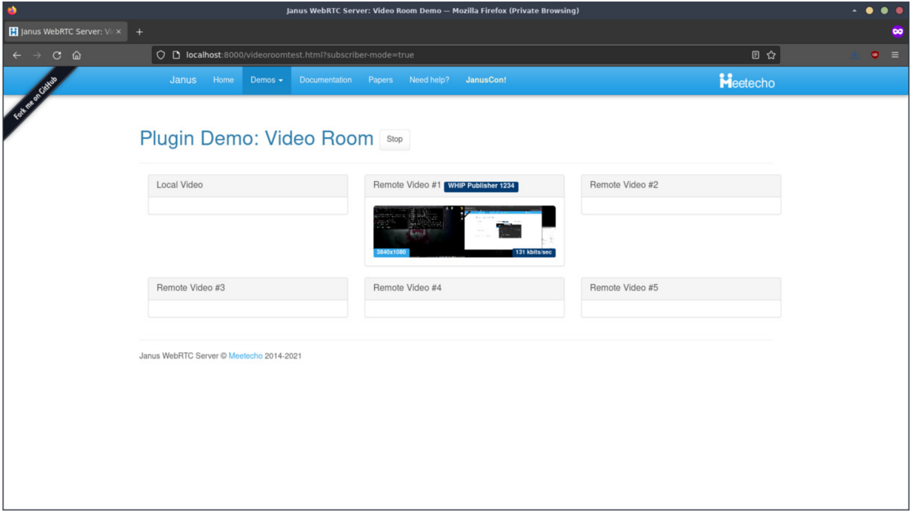


Testing Sergio’s WHIP client with Galene

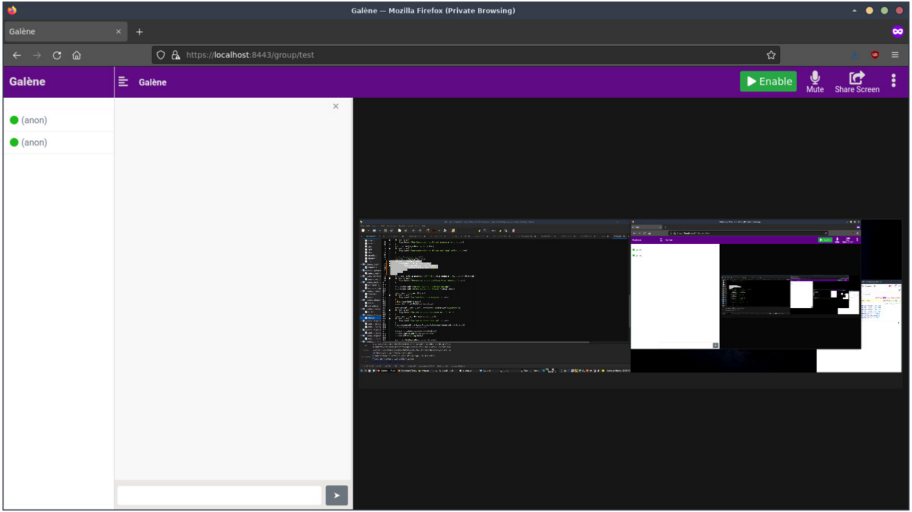


Testing Sergio's WHIP client with Millicast

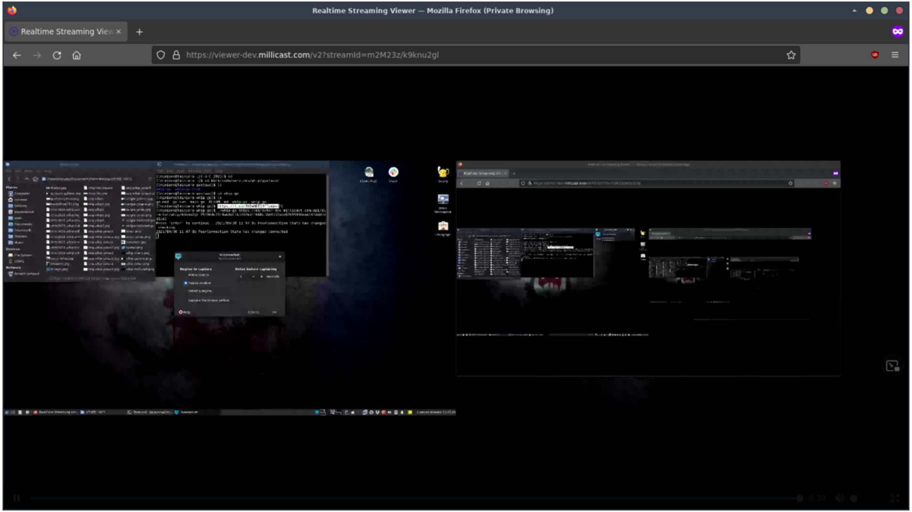
Testing Gustavo's WHIP client with Janus



Testing Gustavo’s WHIP client with Galene



Testing Gustavo's WHIP client with Millicast



Got some feedback to improve the spec

- Interoperability was surprisingly quite successful!
  - Even in early stage of specification, draft was easy to implement
  - All tests across different implementations got a PeerConnection working
- That said, some potential issues or challenges were identified
  - Unlike native clients, web-based WHIP clients are subject to CORS
  - RFC8840's format for candidates may be a bit too “convoluted”?
  - Document should expand on what to return in case of errors
  - How to respond to PATCH for trickle is unclear too (e.g., 204 vs. 200 vs. ??)
  - There may be race conditions between PATCH requests when doing an ICE restart

Got some feedback to improve the spec

- Interoperability was surprisingly quite successful!
  - Even in early stage of specification, draft was easy to implement
  - All tests across different implementations got a PeerConnection working
- That said, some potential issues or challenges were identified
  - Unlike native clients, web-based WHIP clients are subject to CORS
  - RFC8840's format for candidates may be a bit too "convoluted"?
  - Document should expand on what to return in case of errors
  - How to respond to PATCH for trickle is unclear too (e.g., 204 vs. 200 vs. ??)
  - There may be race conditions between PATCH requests when doing an ICE restart

Next stop: IETF 112 Hackathon!

I E T F

ABOUT · TOPICS OF INTEREST · HOW WE WORK · INTERNET STANDARDS · News & blog · Contact · Tools · Search

> How we work > Running code > IETF Hackathons

IETF 112 Hackathon Online

At IETF Hackathons, developers and subject matter experts discuss, collaborate, and develop utilities, ideas, sample code and solutions that show practical implementations of IETF standards.

When: Monday-Friday, November 01-05, 2021
Where: Online

The Hackathon is free to attend and open to everyone. It is a collaborative event, not a competition. Any competitiveness among participants is friendly and in the spirit of advancing the pace and relevance of new and evolving internet standards.

- Register for Hackathon- HERE!
- View the Hackathon attendees list- HERE!
- Subscribe to the email list to stay up to date
- Check out the Hackathon wiki to sign up for a project, or add your own.

Hackathon Co-Chairs:

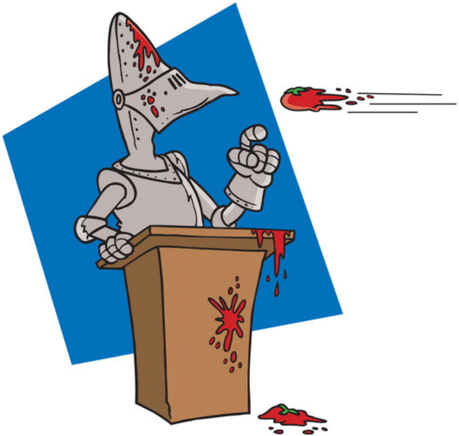
Charles Eckel, Cisco & Barry Leiba, Futurewei

IETF HACKATHONS

- IETF 110 Hackathon Online
- IETF 109 Hackathon Online
- IETF 108 Hackathon Online
- IETF 106 Hackathon Singapore
- IETF 105 Hackathon Montreal
- IETF 104 Hackathon Prague
- IETF Hackathon Bangkok
- IETF Hackathon Montreal
- IETF Hackathon Prague

https://www.ietf.org/how/runningcode/hackathons/112-hackathon/

Thanks! Questions? Comments?



Get in touch!

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