Implementation Report: WHIP in Galene

Juliusz Chroboczek
IRIF, University of Paris

12 November 2021

Galene

Galene is a WebRTC SFU.

`https://galene.org`

Implemented during first French lockdown,
to meet my teaching needs.

Goals:

- easy to install (15 minutes);
- easy to administer (no specialist needed);
- moderate server-side resources (400 flow/core);
- scales up to 16 cores.

(Development supported by `https://nexedi.com`.)

WHIP in Galene: goals

In September 2021, I implemented WHIP in Galene.

First-class implementation, not a translation into Galene's native signalling protocol.

Goals :

- verify that Galene's internals are flexible enough to implement multiple signalling protocols;
- get access to third-party clients (notably for mobile);
- play with a cool new protocol.

Good news

Implementing WHIP in Galene was easy:

- one night of work;
- 427 lines specific to WHIP;
- 50 lines of patches to Galene's core (mostly exporting internal data structures).

The need to export internal data structures indicates that Galene's package structure is not granular enough. Useful guidance for future development.

Thanks to Lorenzo Miniero for `simple-whip-client`.

Try it at home

Just five shell commands:

```
git clone -b whip \
    https://github.com/jech/galene
cd galene
mkdir groups
echo '{"op":[{}],"allow-anonymous":true}' \
     > groups/test.json
go run galene.go -public-server
```

Then point your browser and your WHIP client at

```
https://localhost:8443/group/test/
```

Difficulties

Minor difficulties. Two main reasons:

- I am incompetent;
- WHIP and Galene use different models.

If WHIP is successful, I will no longer be the only incompetent WHIP implementer.
The draft must be aimed at incompetents.

Difficulties:

- CORS;
- URL multiplexing;
- Authentication;
- distinguishing candidates from restarts.

CORS Preflight

WHIP in Galene was developed against Lorenzo's `simple-whip-client`, which is a native client.

Web clients perform a CORS preflight, which apparently mitigates some web security issues. They fail if the server doesn't handle CORS preflight.

This was explained to me by Lorenzo. Not all implementers have access to Lorenzo.

Suggestion: CORS requirements should be explained in the draft.

URL multiplexing

Galene uses the same URL for the native protocol and WHIP:

- `GET` request ⟶ native JS client;
- `POST` request ⟶ WHIP request.

Is this too fragile? Not extensible enough?

Suggestion: add `?protocol=whip` to initial WHIP requests.

(Or use a custom HTTP header?)

Authorisation

Galene's native client has two ways to login:

- simple deployments use username/password;
- more complex deployments use a cryptographic token provided by a third-party server (signed JWT).

WHIP only supports token authorisation, with no standard way to map user/password to token.

Suggestion: require that all WHIP clients implement HTTP Basic auth. (Or define a mapping from username/password to token.)

ICE restarts

In WHIP, trickled ICE candidates and ICE restarts use the same MIME type.

Must be distinguished by ufrag. Issues:

- explicit is better than implicit;
- fragile in the presence of reordering.

Suggestion: add `?type=restart` to ICE restarts, or use a custom HTTP header.

Conclusion

WHIP is easy to implement on the server side (Assuming you already have a WebRTC stack) :

- one night;
- < 500 lines of code.

Some minor nits remain, should be easy to fix.

There are now multiple WHIP servers:

- Janus;
- Millicast;
- Galene.

It is time to write useful WHIP clients.

(Screensharing on mobile, please.)