WHIP

WebRTC HTTP Ingestion Protocol

The Problem

- WebRTC is the best media transport protocol for real-time streaming.
- While other media transport could be used for ingest, webrtc for both ingest and delivery allows:
  - Working on browsers.
  - Avoiding protocol translation, which could add delay and adds implementation complexity.
  - Avoiding transcoding by sharing common codecs.
  - Using webrtc features end to end.
- However, there is no standard signalling protocol available to pair with it:
  - SIP or XMPP are not designed to be used in broadcasting/streaming services, and there also is no sign of adoption in that industry.
  - RTSP, which is based on RTP and maybe the closest in terms of features to webrtc, is not compatible with WebRTC SDP offer/answer model
- Consequences:
  - Each WebRTC streaming services requires implementing a custom ad-hoc protocol.

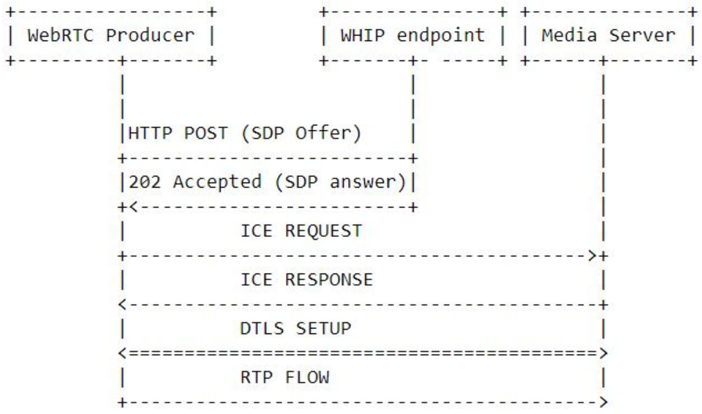
We need a reference signalling protocol.

Requirements

- Must be simple to implement, as easy to use as current RTMP URI.
- Support the specific ingest use case, which is a subset of webrtc possible use cases:
  - Only needs to support unidirectional flows.
  - Server is assumed to not be behind NAT (having a public IP or deployed in same private network as publisher)
  - No need to support renegotiations.
- Fully compliant with WebRTC and RTCWEB specs for the given use case.
- Must support authentication.
- Usable both in web browsers and in native encoders.
- Lower the requirements on both hardware encoders and broadcasting by reducing optionalities.
- Supports load balancing and redirections.

Proposed solution

- HTTP POST for exchanging and SDP O/A.
- Connection state is controlled by ICE/DTLS states
  - ICE consent freshness [RFC7675] will be used to detect abrupt disconnection
  - DTLS teardown for session termination by either side.
- Authentication and authorization is supported by the Authorization   HTTP header with a bearer token as per [RFC6750].
- Support HTTP redirections for LB.



WHIP session setup

Example implementation in JS

```
//Get user media
const stream = await
navigator.mediaDevices .getUserMedia ({audio:true, video:true});
//Create peer connection
const pc = new RTCPeerConnection ();
//Listen for state change events
pc.onconnectionstatechange  = (event)  =>{
        switch(pc.connectionState )  {
                case "connected" :
                        break;
                case "disconnected" :
                        break;
                case "failed" :
                        break;
                case "closed" :
                        break;
        }
}
```

```
//Send all tracks
for (const track of stream.getTracks ())
        //You could add simulcast too here
        pc.addTrack (track);
//Create SDP offer
const offer = await pc.createOffer ();
await pc.setLocalDescription (offer)
//Do the post request to the WHIP endpoint with the SDP offer
const fetched = await fetch(url, {
        method: "POST",
        body: offer.sdp,
        headers:{
                "Content-Type" : "application/sdp"
        }
});
//Get the SDP answer
const answer = await fetched.text();
await pc.setRemoteDescription ({type:"answer",sdp: answer});
```

Reducing implementation complexity

- Server may implement ICE lite, encoder must implement full ICE.
- SDP bundle and RTCP muxing must be supported by both sides.
- Encoder/media producer may use a setup attribute value of setup:active in the SDP offer, server must support acting as passive.