CSRCs/AudioLevels in WebRTC 1.0

at 2015 f2f

Open Issues

#4: Need API to read CSRC on received tracks

#6 Need API to receive Mixer-to-Client audio level information for a track

#276 Control to turn on Client-to-Mixer Audio Level in Offers

Issue #276: Control to turn on Client-to-Mixer Audio Level in Offers

Support for RFC 6464 is mandatory in RTP-Usage.

Use cases:

- PERC (SFU can’t decrypt the payload)
- Forwarding vs. Mixing (forwarding avoids Opus encoding/decoding).

Proposal from Harald:

- Implementations MUST offer Client-to-Mixer audio level and use it if negotiated.
- No new API surface needed

Issue #4: Need API to read CSRC on received tracks

Usage scenario (from Cullen): Conference systems that use an MCU style mixer use less bandwidth than SFO style systems that forward unmixed audio and have the client do the mixing. We need support for both types. Many deployed MCU systems support this (but not all).

The Javascript app gets the SSRC and name of each user out of band from the conferences system as each user joins.

Using the CSRCs, the app can have a UI that indicate which users are generating audio. It can poll the list of CSRCs most recently received by the browser. (A poll rate of 50ms should work fine)

Issue #4 (cont'd)

Proposal discussed in ORTC CG:

```
partial interface RTCRtpReceiver {
    sequence<RTCRtpContributingSource> getContributingSources ();
};

dictionary RTCRtpContributingSource {
        DOMHiResTimeStamp timestamp;  //When an RTP packet containing the contributing source was last received
        unsigned long     csrc;   // the contributing source
};
```

Pros

Implementation experience: seems to work, not complicated to implement.

Likely to be used in conferencing scenarios.

Cons

Not useful outside of conferencing scenarios.

Issue #6: Need Mixer-Client Audio Level Info

Mixer-Client Audio Level (RFC 6465) optional in RTP-Usage.

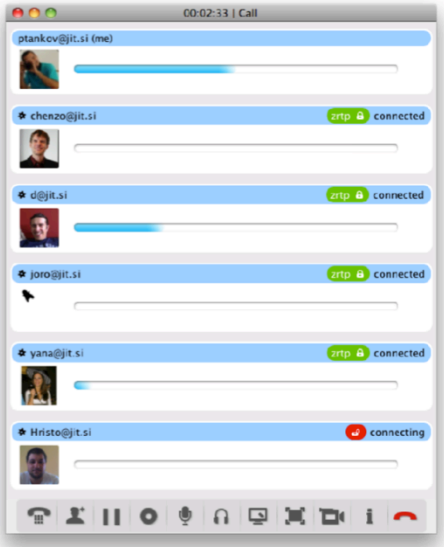
Proposal by Harald: Don’t add API support in WebRTC 1.0.

Cullen: Don’t care if we have this in WebRTC 1.0.

Emil: Main use case is implementing GUI for mixed

https://www.dropbox.com/s/j9rv8jdczjnmzxr/audiolevels.png?dl=0

Can you do it by sending the levels some other way? Yes. CSRC audio levels would potentially be more in-sync but does that matter? To some it probably would. To many others probably not.



Issue #6 (cont'd)

Potential extension to RTCRtpContributingSource dictionary:

```
partial dictionary RTCRtpContributingSource {
        byte        audioLevel;
//audio level of the contributing source. Value is between 0 and -127 representing the contributing source dBov value [RFC6465]
};
```

Pros

- Could be used to implement Emil's UI example.

Cons

- If goal is just to identify sources contributing energy, CSRCs are sufficient.
- Alternatives available: Audio analysis can be done on the mixer, with results sent to browser via data channel.