JSEP Update

The Sequel

Justin Uberti & Cullen Jennings

IETF 85

For Today

- Media stream taxonomy
- Mapping of MediaStreams to m-lines
- Trickle ICE
- Rehydration

Media Stream Taxonomy

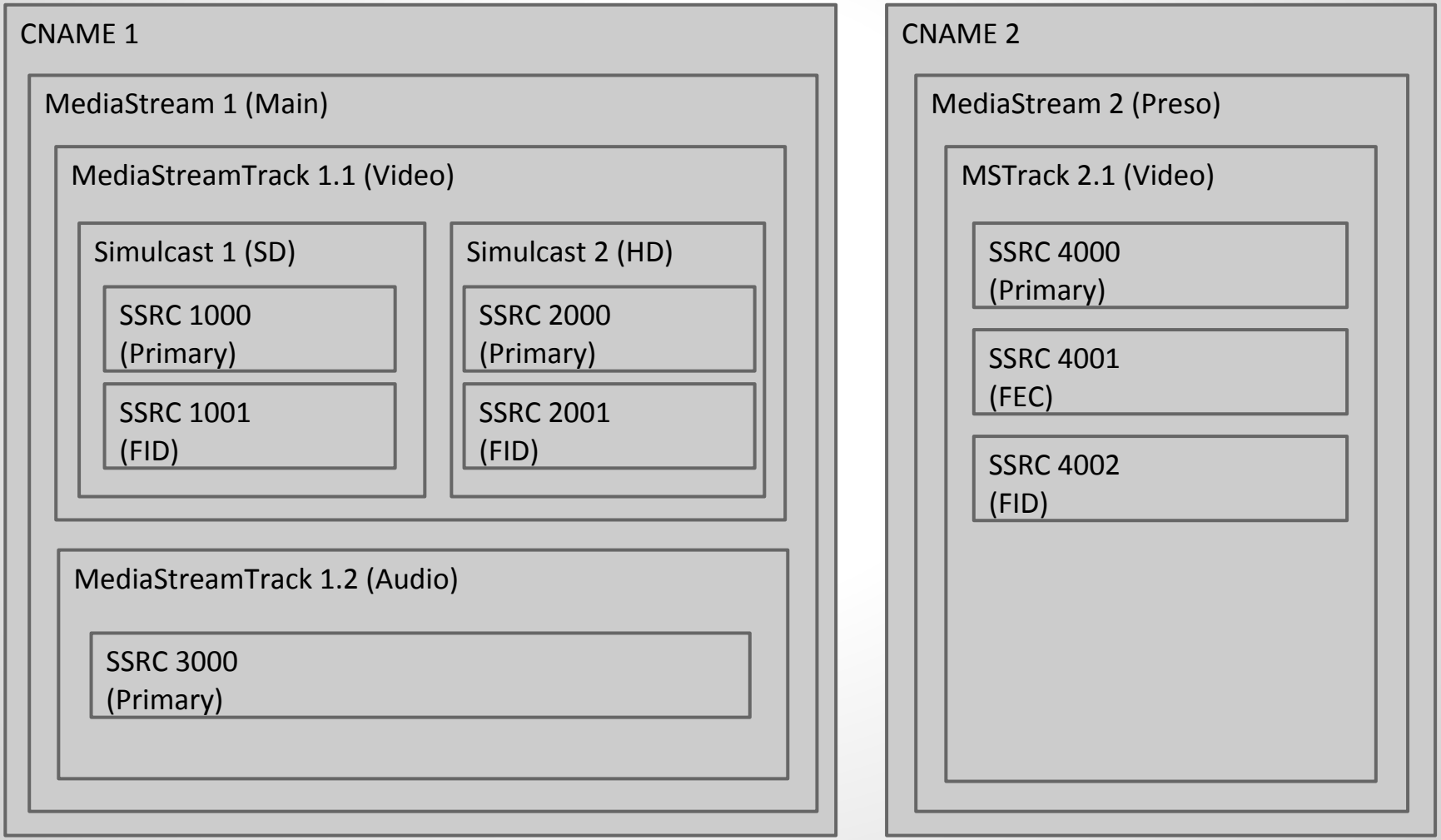
The term "media stream" means different things to different people. Depending on whom you ask, it could refer to a:

- W3C MediaStream
- W3C MediaStreamTrack
- SDP m= line
- RTP CNAME
- RTP SSRC
- set of RTP SSRCs

And, depending on the case, they could all be right!

No surprise we have drafts in rtcweb, avtcore, clue discussing how to identify "streams"

An Example


CNAME 1
MediaStream 1 (Main)
MediaStreamTrack 1.1 (Video)
Simulcast 1 (SD)
SSRC 1000 (Primary)
SSRC 1001 (FID)
Simulcast 2 (HD)
SSRC 2000 (Primary)
SSRC 2001 (FID)
MediaStreamTrack 1.2 (Audio)
SSRC 3000 (Primary)
CNAME 2
MediaStream 2 (Preso)
MSTrack 2.1 (Video)
SSRC 4000 (Primary)
SSRC 4001 (FEC)
SSRC 4002 (FID)

Example SDP

```
m=audio
a=ssrc:3000 msid:MediaStream1 a0
a=ssrc:3000 cname:CNAME1
m=video
a=ssrc:1000 cname:CNAME1
a=ssrc:1000 msid:MediaStream1 v0
a=ssrc:1001 msid:MediaStream1 v0
a=ssrc:2000 cname:CNAME1
a=ssrc:2000 msid:MediaStream1 v0
a=ssrc:2001 msid:MediaStream1 v0
a=ssrc:4000 cname:CNAME2
a=ssrc:4000 msid:MediaStream2 v0
a=ssrc:4001 msid:MediaStream2 v0
a=ssrc:4002 msid:MediaStream2 v0
a=ssrc-group:FID 1000 1001
a=ssrc-group:FID 2000 2001
a=ssrc-group:FEC 4000 4001
a=ssrc-group:FID 4000 4002
```

Example Hierarchy

Here's the hierarchy that I think results from this, along with some possible names:

- "Synchronization Context" (CNAME)
  - "Multimedia Source" (MediaStream/partial MSID)
    - "Media Source" (MediaStreamTrack/full MSID)
      - "Media Encoding" (CLUE Capture ID?)
        - "SSRC Group" (ssrc-group)
          - "SSRC" (ssrc)

Jonathan Lennox is writing up a -00 draft to come up with official terminology for these constructs, as well as a way to identify each of them; ideally, will unify MSID, srcname, and capture ID

m= line Mapping

How do MediaStreamTracks map onto m-lines?

For 1 audio + 2 video tracks, we could do either:

- **m=audio 1000**
  a=ssrc:1 msid:Track1
  m=video 1001
  a=ssrc:2 msid:Track2
  a=ssrc:3 msid:Track3

- **m=audio 1000**
  a=ssrc:1 msid:Track1
  m=video 1001
  a=ssrc:2 msid:Track2
  m=video 1002
  a=content:slides

m= line Mapping API

To allow application to control which of these it wants, we add a new MediaStreamTrack.content property, defaulting to empty string. createOffer then generates a m= line for each { media, content } tuple.

- 1 audio + 2 video tracks, no content property
  { "audio", "" } + { "video", "" }
  2 m-lines
- 1 audio + 2 video tracks, video2.content = "slides"
  { "audio", "" } + { "video", "" } + {"video", "slides" }
  3 m-lines

Trickle ICE

New draft, draft-rescorla-mmusic-ice-trickle, specifies the details on how trickle ICE can be used with new and legacy endpoints

JSEP is almost fully compliant with this draft:

- createOffer with no candidates generates a RFC 3264-compliant offer (0.0.0.0:1 address)
- Indication of end-of-candidates provided by API
- Can support "No Trickle", "Half Trickle", and "Full Trickle"
- Just need to add ICE option to indicate trickle support

Rehydration

Previous approaches to rehydration have focused on persisting and restoring local call state, but SDP does not provide enough info to do this reliably:

- ICE ports and credentials
- RTP seqnum
- SRTP ROC
- DTLS key material

While we could find ways to persist and restore this information, we favor a simpler approach called Session Restart.

Session Restart

Session restart is an ICE restart plus restart of crypto, resulting in a new offer/answer exchange (re-INVITE) to the remote peer.

For rehydration:

- Store old local description
- Refresh page and create new PeerConnection
- Apply old local description as offer
- createOffer("RestartSession") to get new local description with new ICE and crypto
- Apply new local description as offer and send it
- Apply received answer as remote description

Could also be used to fix non-rehydrated calls...

Questions