JSEP Update

Justin Uberti & Cullen Jennings

IETF 85

Topics

- Recap
- Implementation Status
- Report from W3C meeting
- Changes since last draft
- Pending Issues

Recap from last IETF

- Resolved
    - Serial forking/Trickle ICE interaction
    - Special offer creation
        - Capabilities, ICE restart, etc
- Deferred
    - Cloning
    - Rehydration

Implementation Status

- Current WebRTC implementations
  - Public: Chrome 23 Beta, Chrome 24 Dev
  - Behind flag: Mozilla 18 Alpha
- Using W3C RTCPeerConnection API
  - createOffer/Answer, setLocal/Remote now async
  - startIce removed, ICE starts automatically
- Getting mileage on key WebRTC functionality
  - Opus
  - ICE, STUN, TURN
  - SRTP
  - DTLS
  - BUNDLE
  - MSID

W3C Status

- Issues recently resolved
  - Format of SessionDescription, Candidate objects
  - State machines for signaling and ICE
  - Timings for MediaStream-related callbacks
  - Mapping of MediaStreams to m= lines
  - Error handling
- TBD issues
  - getUserMedia timing
  - Handling of multiple streams/tracks
  - Exact format of stats API
  - DTMF API

Changes to JSEP -02

Removed old cruft including ROAP stuff

Removed stuff now covered by W3C API spec

Clarified applicability of PRANSWER

Provided explanation around forking

Added a list of SDP that MUST be implemented

Open Issues

- Document review notes and feedback
- What SDP must be supported
- Cloning
- Quirks Mode
- MSID
- Handling of multiple streams
- Trickle ICE
- Rehydration

Document review notes

Richard Ejzak and Andrew Hutton, as well as several others, provided detailed feedback:

- JSEP state machine vs RFC 3264 state machine is unclear
  - Reached agreement on state machine at W3C meeting, need to update draft accordingly
- Exact SDP generated by createOffer is not defined, especially in a re-INVITE scenario
- Exact handling of SDP by setLocal/setRemote not defined
  - Starting to nail these down in -02
- Relationship between MediaStreams and m= lines is unclear
  - Sent proposal, general agreement on this at W3C

Proposal for SDP that MUST be implemented

RFC 4566 base SDP spec

RFC 5124 to signaling RTP/SAVPF profile

RFC 5761 to signal multiplexing of RTP and RTCP

RFC 5245 to signaling the ICE candidate

RFC 3264 to signal media direction

RFC 5888 grouping framework

RFC 5576 to signal RTP SSRC values

RFC 4572 for DTLS/SRTP Fingerprint

RFC 4733 for DTMF

draft-spittka-payload-rtp-opus for Opus

RFC 5245 for ICE UDP

RFC 6544 for ICE TCP

Proposal for SDP that MAY be implemented

RFC 5506 to signal Reduced-Size RTCP

RFC 3556 with bandwidth modifiers

Should we add?

MUST:

RFC 6236 Image resolution negotiation

RFC 4796 for a=content

draft-rescorla-mmusic-ice-trickle for Trickle ICE

draft-ietf-mmusic-sdp-bundle-negotiation

draft-alvestrand-mmusic-msid for MSID in SDP

draft-ietf-mmusic-sctp-sdp for data channel

MAY:

RFC 3890 TIAS

draft-westerlund-mmusic-sdp-bw-attribute

Forking with PRANSWER

A ---- offer --------------> PSTN

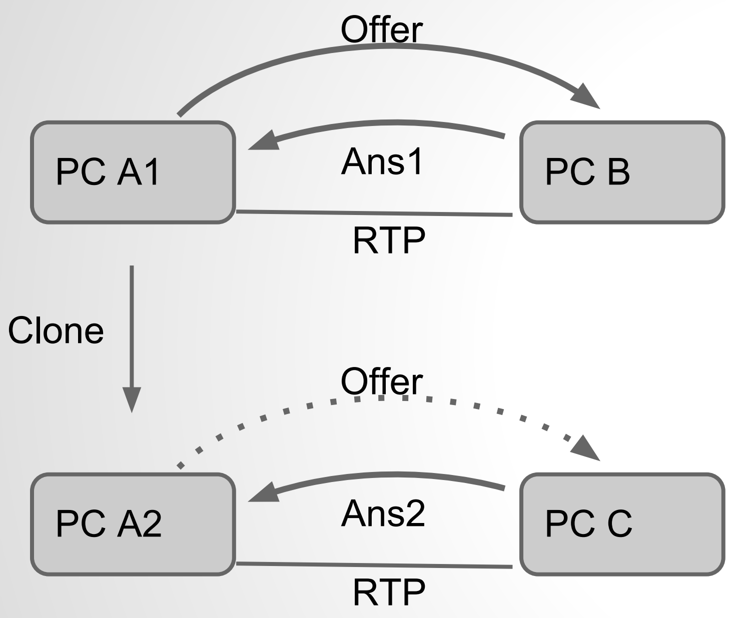
A <-- pr answer --------- PSTN

A ---- ringing media ---- PSTN

A <---- answer ----- VoiceMail

A ------ media ------ VoiceMail

Forking with Cloning



1. PC A1 sends offer to B
2. B sends answer to A1
3. JS clones A1 to form A2
4. A2 is in the offer state, just like A1
5. A1 installs B's answer
6. Media between A1 and B
7. C sends answer to A2
8. A2 installs C's answer
9. Media flows between A2 and C
10. JS application figures out what to do with the media from B and C

Compare with PRANSWER:

3. A1 installs B's answer as PRANSWER
4. Media between A1 and B
5. A1 installs C's answer
6. Media between A1 and C (only)

Cloning - Details

The cloned Peer Connection would share same ports and ICE ufrag/pwd on local side, but have an independent state machine. To work this needs:

- ○ Use of ICE and ability to demux by 5-tuple
- ○ Ability to allocate 2x encode/decode resources
- ○ Ability to allocate 2x bandwidth
- ○ Ability of app to handle simultaneous incoming media from multiple locations

More complicated than PRANSWER, and more flexible. But do we need it? Don't want to support both.

- Can constrained devices handle the 2x requirements above?

- Do we need to support 200 after 180 with different SDP?

Quirks Mode

A recurring question: should JSEP generate "correct" SDP, or "compatible" SDP?

e.g. SRTP/AVPF vs RTP/AVP for profile, for devices that croak when they see SRTP/AVPF

Our proposal: generate correct SDP by default, but allow app to generate quirky SDP by use of a "QuirksMode" constraint that can be passed to CreateOffer/Answer.

MediaStream ID

MediaStream ID (MSID) is the glue that binds MediaStreamTracks (from the JS API) to SSRCs in SDP:

```
m=video
a=ssrc:1 msid:<MediaStreamTrack.label>
```

- Necessary to support multiple MediaStreamTracks and SSRCs.
- Not the same as SSRC; MediaStreamTrack can map to multiple SSRCs (e.g. SSRC-muxed FEC)
- Not the same as CNAME, CNAME can be shared between tracks (i.e. they are synced).
- Need to understand distinction between MSID and recent CLUE "srcname" and "capture ID" proposals.

Stream Selection

General problem:

- A has M streams (e.g. 2 cameras), B has N streams (e.g. MCU with 5 participants)
- How are streams described in SDP?
- How does B select some or all of the streams from A, and vice versa?
- How does this work when B is a legacy device that can only receive one stream?

Stream SDP

A's SDP:

- `m=video`
  a=ssrc:1 msid:<Cam 1 MediaStreamTrack>
  a=ssrc:2 msid:<Cam 2 MediaStreamTrack>

B's SDP:

- m=video
  a=ssrc:11 msid:<User 1 MediaStreamTrack>
  a=ssrc:12 msid:<User 2 MediaStreamTrack>
  a=ssrc:13 msid:<User 3 MediaStreamTrack>
  a=ssrc:14 msid:<User 4 MediaStreamTrack>
  a=ssrc:15 msid:<User 5 MediaStreamTrack>

Stream Selection SDP

To get a subset of streams, we can use **draft-lennox-mmusic-sdp-source-selection** to request streams, and specify hints on how they should be sent.

A sends to B, give me User1 in HD, User2/3 in SD:

-
  ```
  m=video
  a=remote-ssrc:11 recv:on
  a=remote-ssrc:11 imageattr:* [x=1280,y=720]
  a=remote-ssrc:12 recv:on
  a=remote-ssrc:12 imageattr:* [x=640,y=360]
  a=remote-ssrc:13 recv:on
  a=remote-ssrc:13 imageattr:* [x=640,y=360]
  a=remote-ssrc:14 recv:off
  a=remote-ssrc:15 recv:off
  ```

Stream Selection O/A

The ideal case (1.5 RTT)

- A sends list of sources to B as offer
- B sends the set of A's sources it wants to receive as answer
- B sends its list of sources to A as offer
- A replies with the set of B's sources it wants to receive as answer

Fallback case (B is legacy)

- A sends list of sources to B as offer
- B replies with answer, no remote-ssrc attribute
- A sends its "default" stream to B

This mostly lines up with direction of CLUE WG, need to nail down details

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

(Backup) Other approaches

Plan for handling multiple streams of the same media type is to combine them on the same m= line, and demux based on a=ssrc.

However, need some way to indicate that endpoint supports this kind of demuxing.

- a=ssrc not sufficient; doesn't have right semantics, and endpoint may not send any SSRCs (i.e. no a=ssrc lines)
- New a=max-recv-ssrc proposal covers this, but has IPR constraints
- We only need a single bit, i.e. a=i-can-demux-ssrcs; suggest we roll this into source selection draft or new