JSEP Update

Justin Uberti
IETF 83.5

Topics

- Activity since IETF 83
- Implementation Status
- Issues Raised

Recap from IETF 83

- Offer/answer state machine specified
- PRANSWER's only meaning is "allow additional future answers"
- Media transmission controlled by SDP direction attribute
- Consensus on Serial forking
- Description of SDP attributes changeable by the application

Activity since IETF 83

- JSEP support landed in Chrome 20
  - Based on draft-ietf-rtcweb-jsep-00
  - Seems to be well understood
- Long discussion about JSEP in API spec
  - Mostly syntactical/ease-of-use
- Several issues raised on list
  - Parallel forking
  - "Special" offer creation
- draft-ietf-rtcweb-jsep-01 published
  - API spec moved to Appendix
  - Document will focus on semantics and SDP layer

Implementation Status

- Chrome 20 has JSEP
  - Suffixed API: webkitPeerConnection00
- Concepts mostly understood by developers
  - Many developers using JSEP API, features like trickle candidates
  - JSEP->ROAP, JSEP->SIP, JSEP->Jingle JS libs all created
- Some confusion on timing of startIce vs setLocalDescription

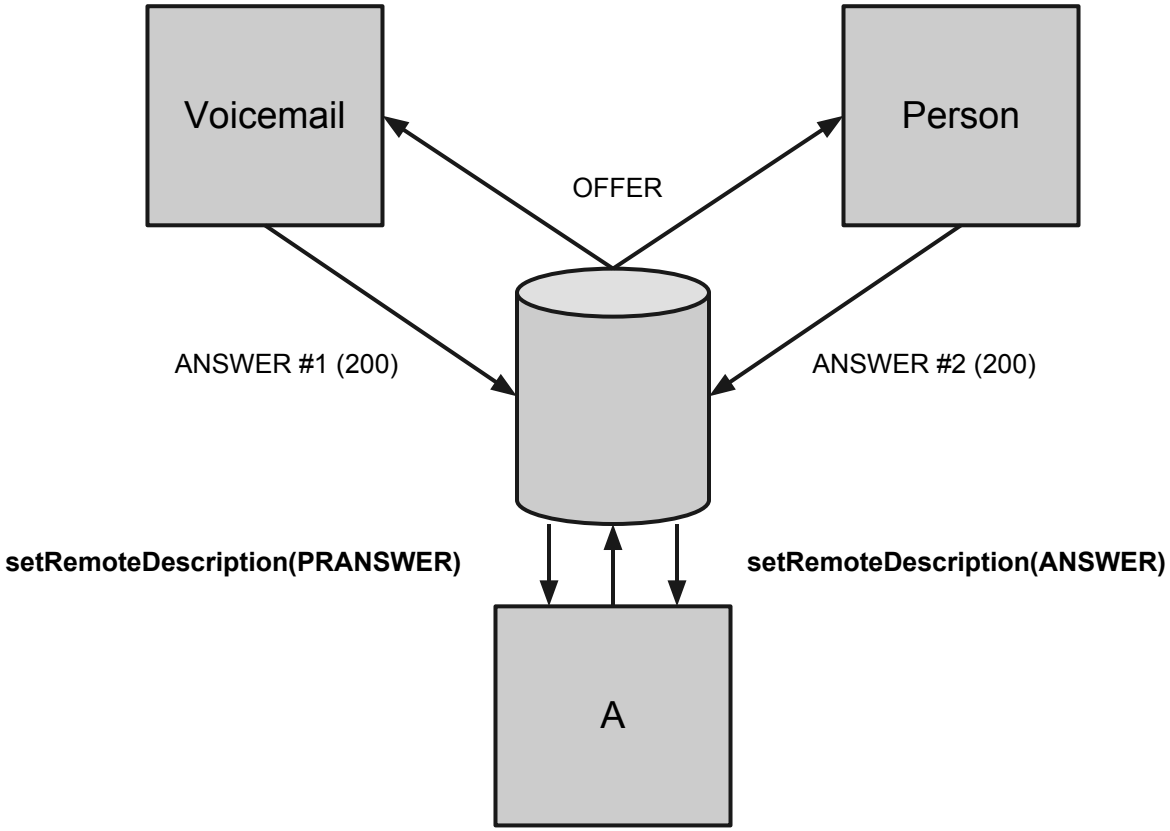
W3C Status

- Resolved issues
  - Async offer/answer generation
  - Semantics of startIce
- TBD issues
  - SessionDescription.type field
  - SDP auto-stringify/unstringify
  - Flags for PRANSWER/restartIce
  - OnRenegotiateNeeded

Issues Raised

- Serial vs Parallel forking
  - PeerConnection Cloning
- Special offer creation
- Rehydration, revisited

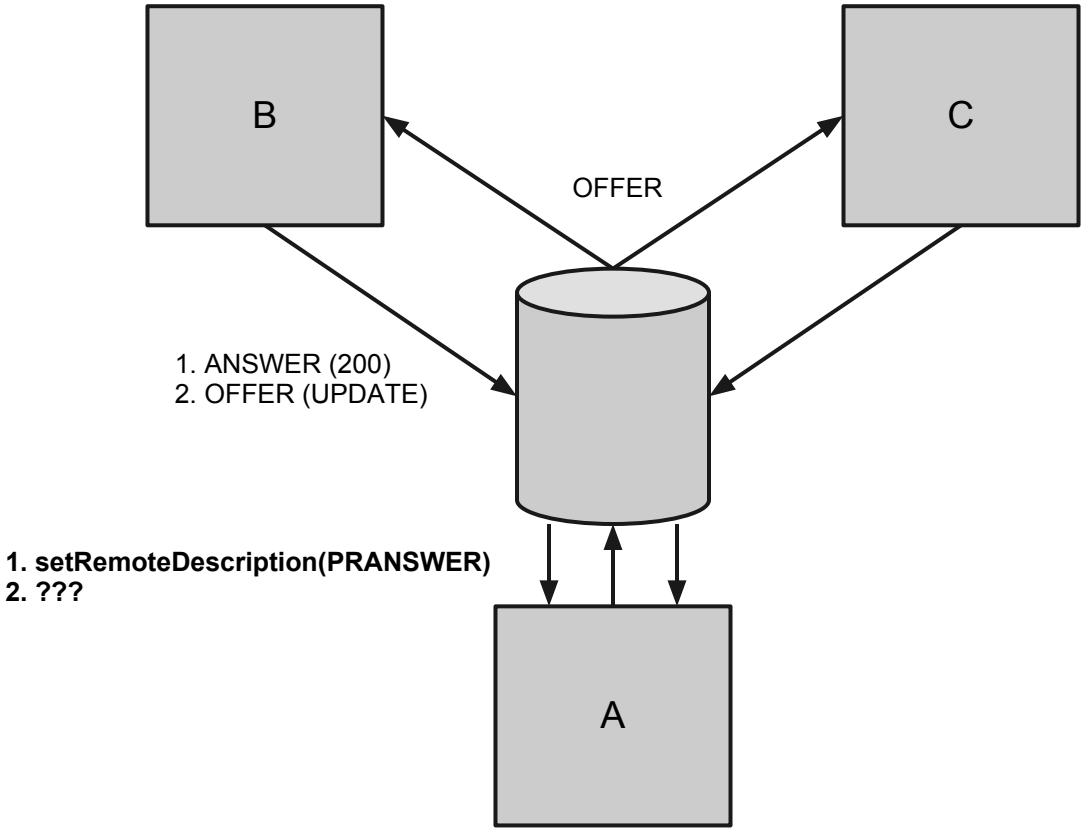
Serial Forking



Serial Forking + Trickle

- Callee's trickle candidates are no longer valid when a new PRANSWER arrives
- Browser needs to discard any existing candidates when PRANSWER/ANSWER is set with different ICE credentials
- Unlikely to be a real-world problem
  - Only SIP forks
  - Only Jingle trickles

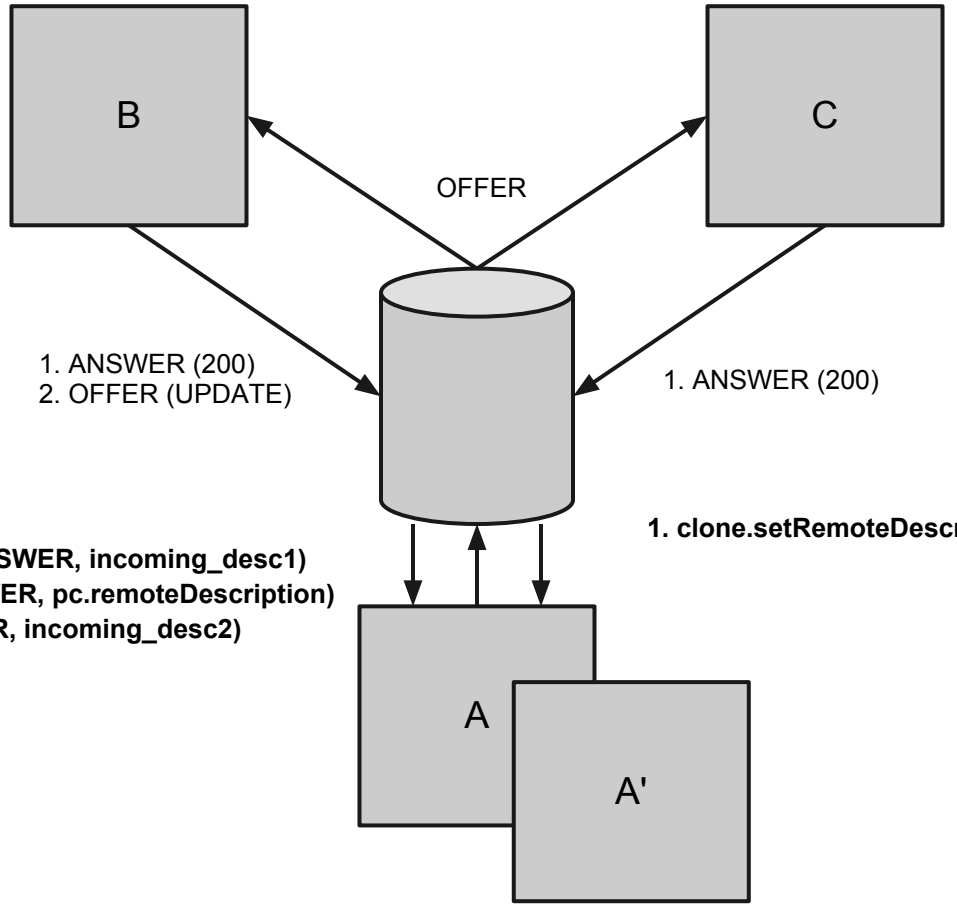
Problematic Call Flow



Solutions

- Do nothing
  - App applies previous PRANSWER as final ANSWER, then handles new OFFER as usual
  - Any future ANSWERs from other remote endpoints must be discarded
- Clone PeerConnection
  - Original PeerConnection follows steps above
  - Cloned PeerConnection stays open for ANSWERs from other remote endpoints
  - Everyone is happy (except implementors, perhaps)

Cloning



1. clone = pc.clone()
2. pc.setRemoteDescription(PRANSWER, incoming_desc1)
3. pc.setRemoteDescription(ANSWER, pc.remoteDescription)
4. pc.setRemoteDescription(OFFER, incoming_desc2)

1. clone.setRemoteDescription(ANSWER, incoming_desc3)

Cloning, in-depth

(Thanks to Richard Ejzak)

1. PeerConnection.clone creates a new PeerConnection from an existing PeerConnection
2. The parent PC must have a valid localDescription, but cannot have a final remoteDescription (i.e. must be in OFFER or PRANSWER state)
3. Cloned PC object will inherit the local streams, local ICE candidates, and local description of the PC, but not the remoteDescription.
4. The state of the cloned PC will be OFFER.
5. Cloning will fail if the resources needed by the second localDescription cannot be allocated.

Questions

We already support serial forking, and multiple independent PeerConnections.

How important is cloning/parallel forking?

- In scope for v1?
- In scope at all?

Special Offer Creation

- Get capabilities
  - SIP OPTIONS, essentially
  - Return SDP of everything supported, even if not typically used
- "Full" offer
  - Want to generate a new complete offer in an existing session
- ICE restart
  - Want to generate an offer with a new ufrag/pwd
- Can we do this with offer constraints?

Rehydration

- Added section in draft to discuss this in more detail
- Basically this becomes
  - save localDescription
  - kill PeerConnection
  - create new PeerConnection
  - change ICE ufrag/pwd in saved description
  - setLocalDescription(OFFER, munged_desc)
  - ICE restarts, new remote desc received
  - setRemoteDescription(ANSWER, remote_desc)
  - call continues

Next steps

- Decide on whether we want to proceed with cloning
- Start referencing draft-ietf-rtcweb-rtp-usage-03
  - Specify what SDP created by createOffer/Answer should look like

Questions