ROAP Design Rationale

Cullen Jennings
January 2012

ROAP Design Rationale

- Be the simplest protocol possible that can do RFC 3264 Offer/Answer with SDP

- Concerns about an SDP-based interface:
  1. Forces browser to be conscious of SDP offer/answer state
  2. Speed of call setup
  3. State resurrection on page reload

About SDP...

- The SDP Offer/Answer pair define what media engine is sending and capable of receiving
- Offer/Answer pair changes over time. RFC 3264 provides the rules to keep consistent state between the two sides
- Allows the state to evolve in a linear versioned line, but does not allow branching or merging

SDP Streams and Formats

- RFC 3264 - Sec 5

If multiple formats are listed, it means that the offerer is capable of making use of any of those formats during the session. In other words, the answerer MAY change formats in the middle of the session, making use of any of the formats listed, without sending a new offer.

If multiple media streams of the same type are present in an offer, it means that the offerer wishes to send (and/or receive) multiple streams of that type at the same time.

Offer with two video streams each offering two codecs

```
v=0
o=bob 123 1 IN IP4 192.0.2.1
s=-
c=IN IP4 192.0.2.1
t=0 0
m=video 5134 RTP/AVP 31 34
  a=rtpmap:31 H261/90000
  a=rtpmap:34 H263/90000
m=video 5132 RTP/AVP 31 34
  a=rtpmap:31 H261/90000
  a=rtpmap:34 H263/90000
```

Managing codec resources - From RFC 3264

Allocating

Once the offerer has sent the offer, it MUST be prepared to receive media for any recvonly streams described by that offer. It MUST be prepared to send and receive media for any sendrecv streams in the offer, and send media for any sendonly streams in the offer

Sending Media

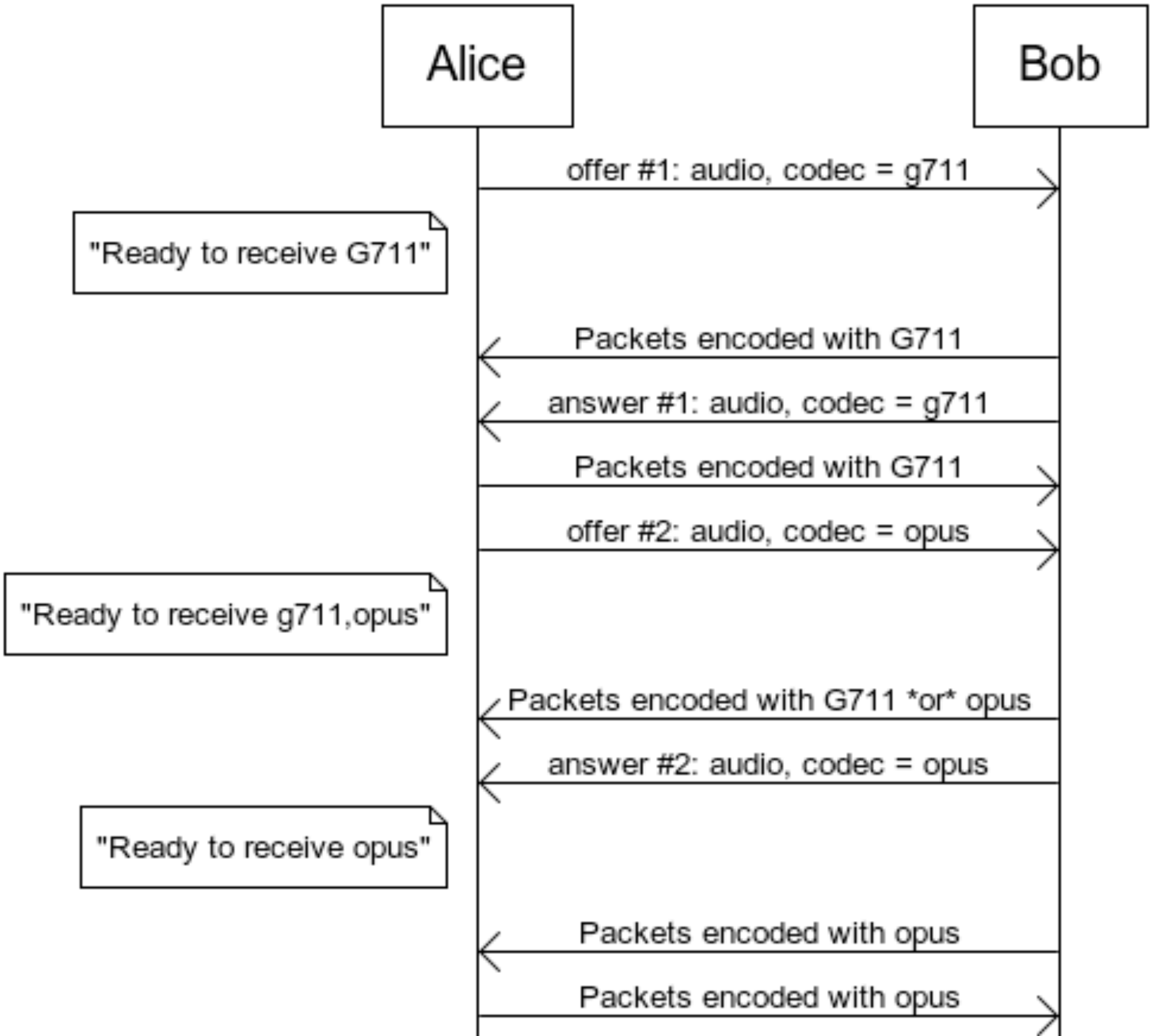
When the offerer receives the answer, it MAY send media on the accepted stream(s) (assuming it is listed as sendrecv or recvonly in the answer).

Once the answerer has sent the answer ... The answerer MUST be prepared to receive media for recvonly or sendrecv streams using any media formats listed for those streams in the answer, and it MAY send media immediately.

Freeing

The offerer MAY immediately cease listening for media formats that were listed in the initial offer, but not present in the answer.

Resource allocation / deallocation flow



Modifying the SDP offer/answer pair state

- RFC 3264

At any time, either agent MAY generate a new offer that updates the session. However, it MUST NOT generate a new offer if it has received an offer which it has not yet answered or rejected. Furthermore, it MUST NOT generate a new offer if it has generated a prior offer for which it has not yet received an answer or a rejection.

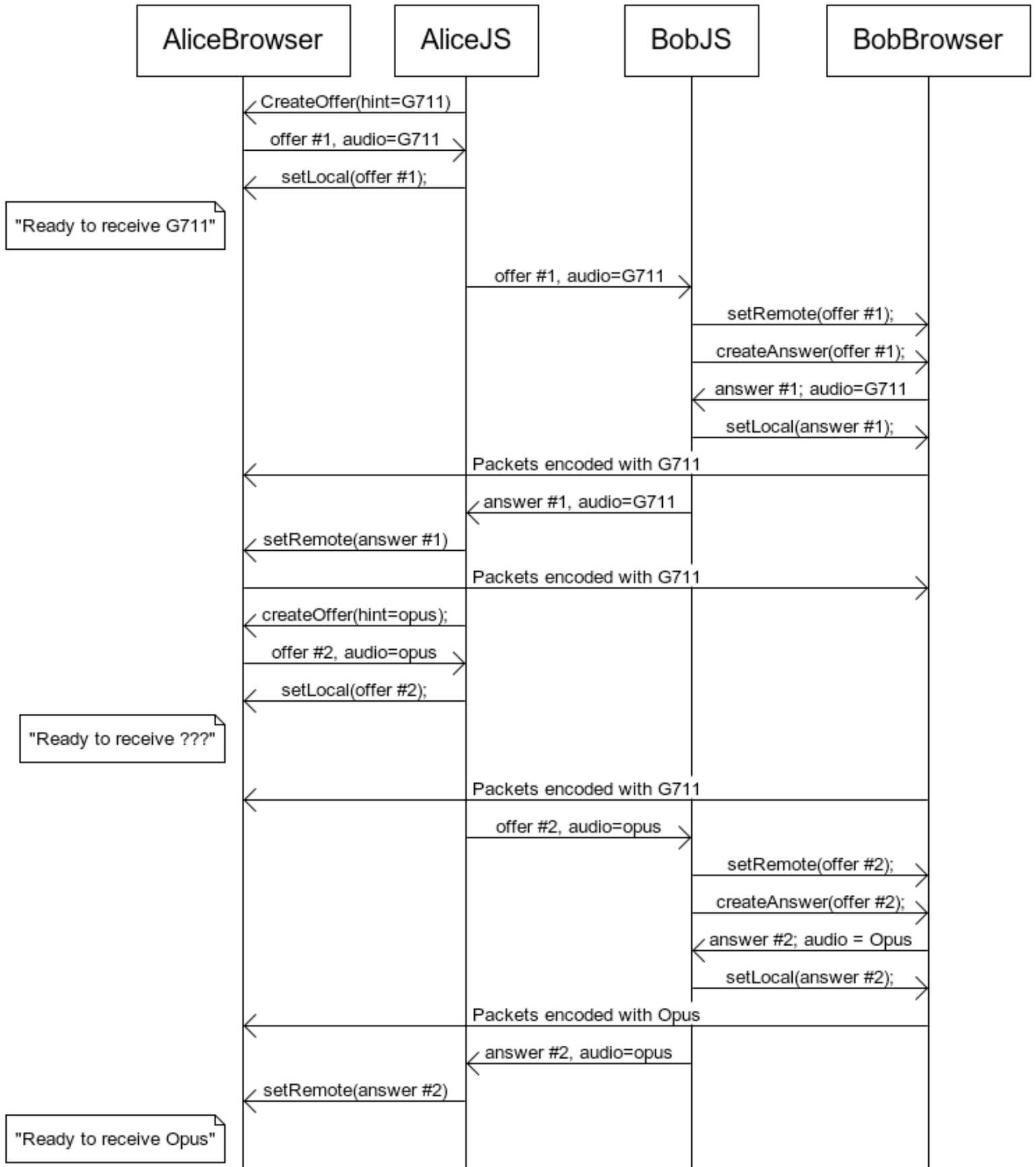
When issuing an offer that modifies the session ... the version in the origin field MUST increment by one from the previous SDP.

If an SDP is offered, which is different from the previous SDP, the new SDP MUST have a matching media stream for each media stream in the previous SDP.

The offerer MUST be prepared to receive media with both the old and new types until the answer is received, and media with the new type is received and reaches the top of the playout buffer.

Of course, if the offered stream is rejected, the offerer can cease being prepared to receive using the new port as soon as the rejection is received.

How the media engine ends up needing outstanding offer state



What is SDP State

- Resources allocated for Old and New SDP
- When to free Old resources, when to free New

```
Offer
   v=0
   o=bob 123 3 IN IP4 172.168.1.1
   s=-
   c=IN IP4 172.168.1.1
   t=0 0
   b=TIAS:50780
   b=maxprate:10.0
   a=group:LS ls1 ls2
   m=audio 65422 RTP/AVP 97 0
     a=mid:ls1
     a=rtpmap:0 PCMU/8000
     a=rtpmap:97 AMR/8000
     a=rtcp:53020
     a=fmtp:97 octet-align;
   m=video 0 RTP/AVP 31
   m=video 53000 RTP/AVP 34
     a=mis:ls2
     a=rtpmap:32 H263/90000
     a=rtcp-mux
   m=audio 51434 RTP/AVP 110
     a=rtpmap:110 telephone-events/8000
     a=recvonly
```

```
Answer
   v=0
   o=alice 123 3 IN IP4 172.168.2.2
   s=-
   c=IN IP4 172.168.2.2
   t=0 0
   m=audio 49170 RTP/AVP 0
     a=rtpmap:0 PCMU/8000
   m=video 0 RTP/AVP 31
     a=rtpmap:31 H261/90000
   m=video 53000 RTP/AVP 34
     a=rtpmap:32 H263/90000
   m=audio 53122 RTP/AVP 110
     a=rtpmap:110 telephone-events/8000
     a=sendonly
```

SDP Hold

- RFC 3264

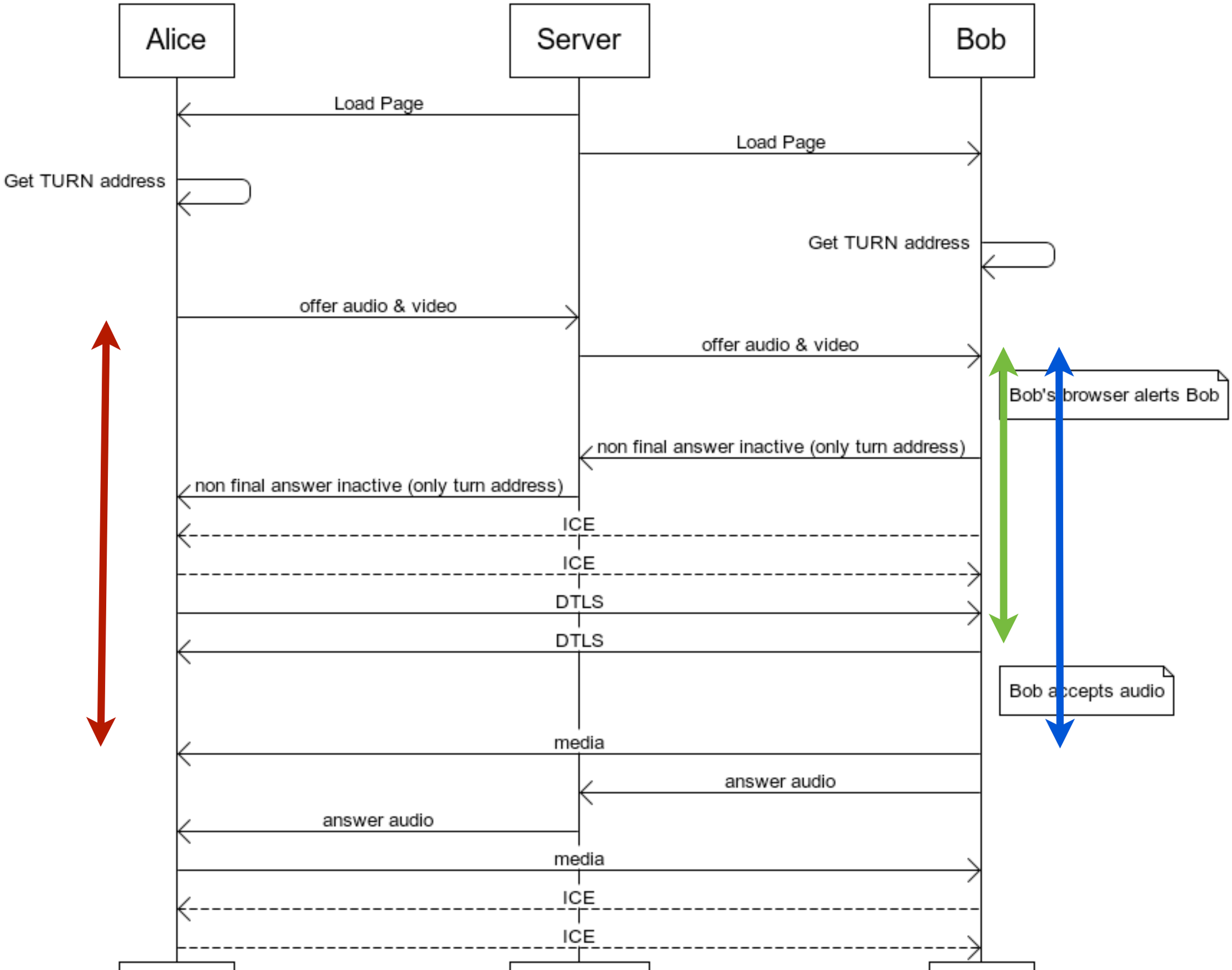
  ```
  If the stream to be placed on hold was previously a sendrecv
  media stream, it is placed on hold by marking it as sendonly.
  If the stream to be placed on hold was previously a recvonly
  media stream, it is placed on hold by marking it inactive.
  ```

- Hold inherently involves the media engine no longer sending certain media packets and the SDP offer/answer pair being updated to reflect that

- PeerConnection needs to be able to tell JS Application when this sort of SDP change is needed. May also need this for changing media for congestion control reasons

  (Note to webrtc folks: you want an API for both hold and mute)

Timing for quick call setup



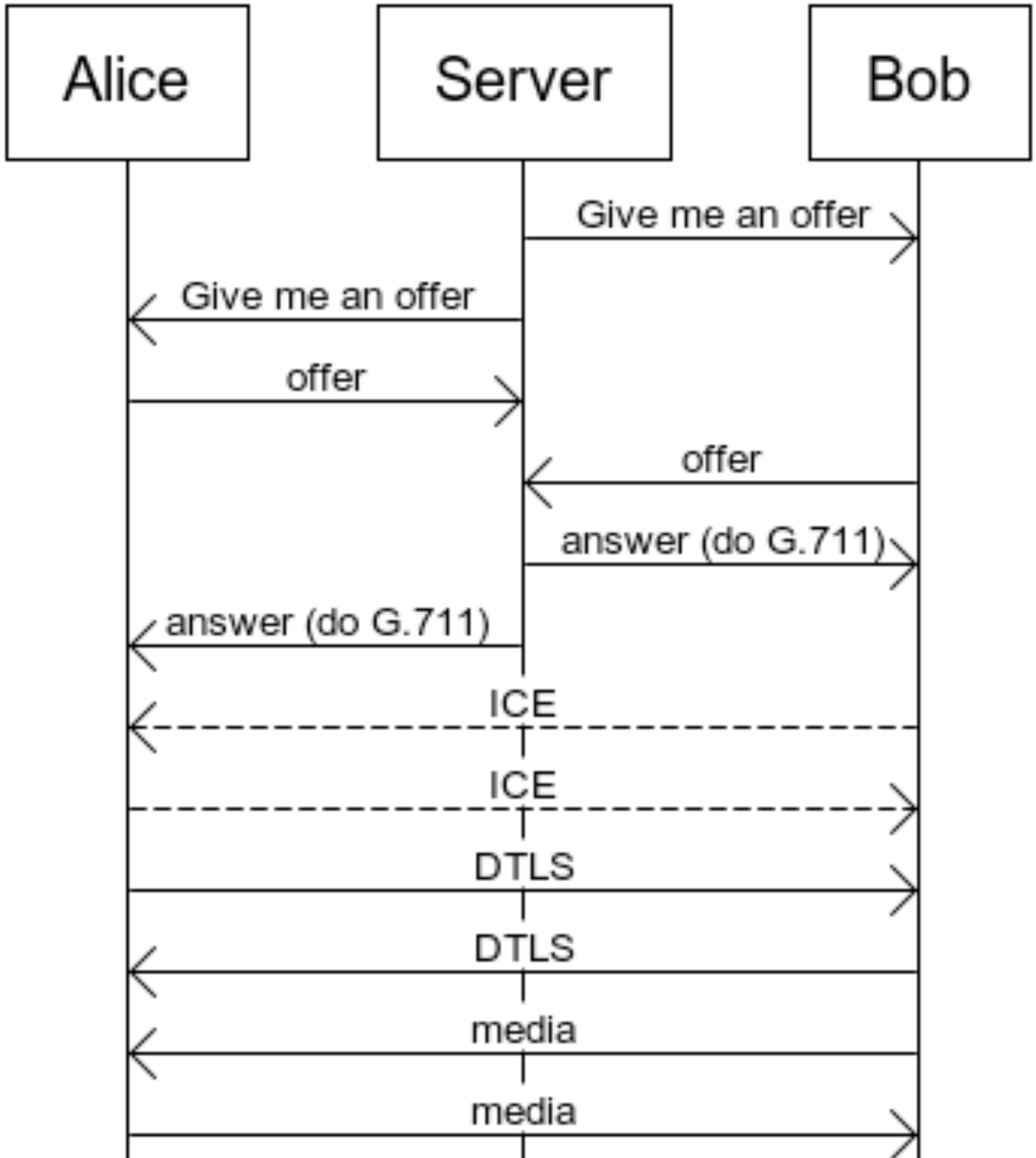
Proposal to add Candidates message to ROAP

- Hard to separate the transport information out of SDP but there is one thing that looks like it could be done to allow "trickle ICE"
- Make a new ROAP message called CANDIDATE with new candidates. This message can be sent at any time and each element in the array augments the candidates in the previous SDP offer/answer
- CANDIATE message structure: array of sets of candidate lines
  - The n'th element in the array is added to the candidates for the n'th "m=" line
- Believe (hope) this will not break ICE because ICE already allows new candidates in form of peer reflexive at any point in time
- Need cost/benefit analysis of performance improvement versus potential problems
- Thoughts? Experiment with this try out?

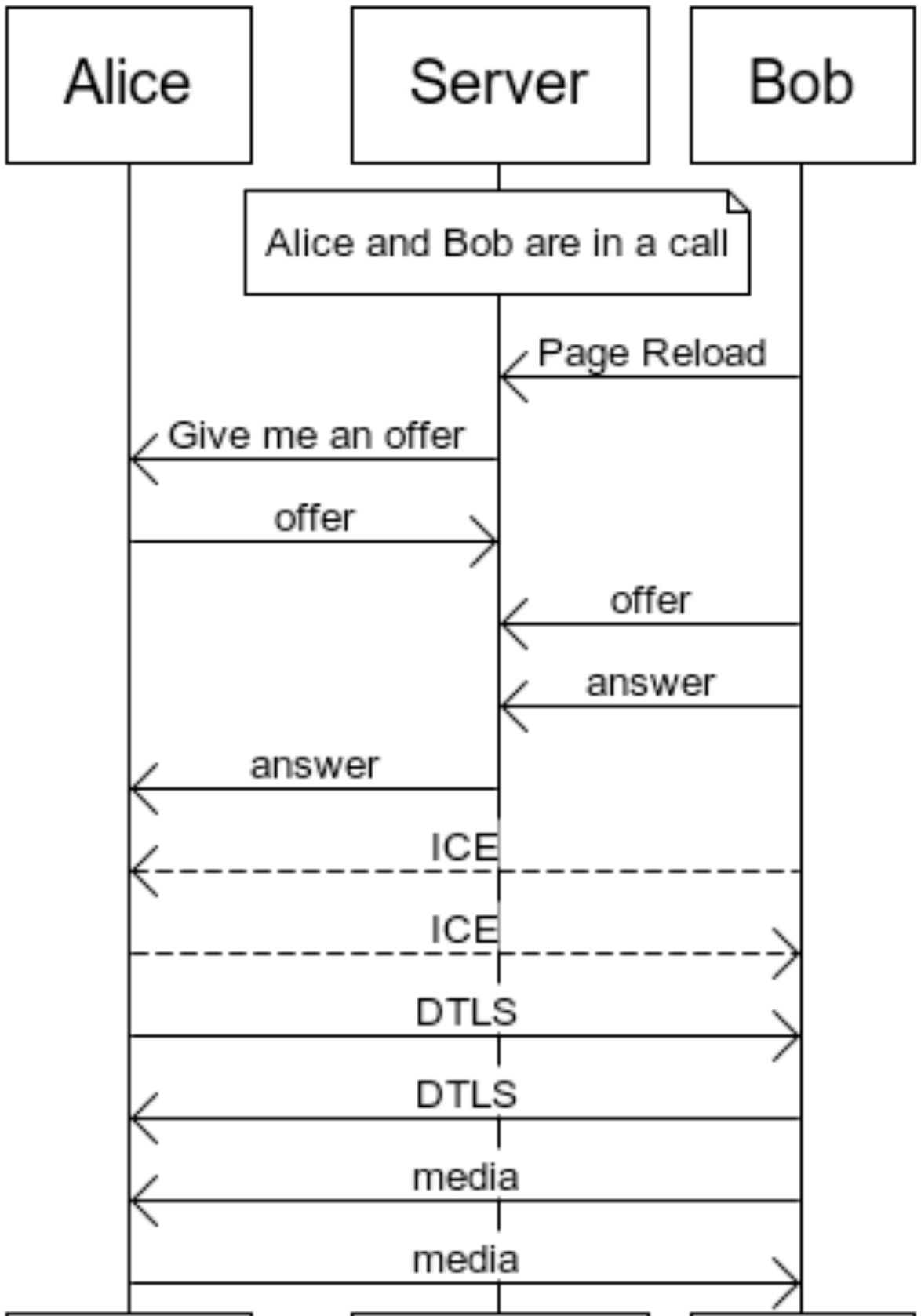
What parts of SDP involve the transport?

```
Offer
   v=0
   o=bob 123 3 IN IP4 172.168.1.1
   s=-
   c=IN IP4 172.168.1.1
   a=setup:actpass
   a=fingerprint: SHA-1 4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
   t=0 0
   b=TIAS:50780
   a=ice-pwd:asd88fgpdd777uzjYhagZg
   a=ice-ufrag:8hhY
   m=audio 65422 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
   a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
   a=rtcp:53020
   a=rtcp-mux
   a=sendrecv
   a=tcap:1 UDP/TLS/RTP/SAVP RTP/AVP
   a=pcfg:1 t=1
```

Web Server Media Engine Control



Resurrection on page reload



Proposal: JS to control update timing

- Current ROAP API calls a callback with new SDP anytime something changes
- JS expected to send SDP immediately
- Instead it could call a callback indicating "conditions changed; new SDP required". The JS Application would then be able to ask for the new SDP and send it

- Not clear where this flexibility helps but it would not be hard to add and would match JSEP in this regard
- Thoughts? Should we do this?

Design Choices

Path	Pros	Cons	Time Line
Use SDP Offer Answer	Finished and deployed	Ugly	4 months
Separate transport out of SDP	architectural purity (Others ?)	Hard to figure out implications of what would break	12 months
Replace SDP	SDP is hideous (Others ???)	Very hard to deploy	many years