WebRTC
API Design Questions

July 23, 2011

Cullen Jennings
Cisco Systems

Design Principles

- A simple app does not need to know a lot about assembling and manipulating the data to do ICE and media negotiations
- A simple app does not require the server to do a lot of work other than relay the signaling messages
- There are things that are widely done by IP phones and soft clients and users like and often use. We should look at existing uses of these to help drive use cases. Programs like Skype, Facetime, and Google Chat , Microsoft Office Communicator, Webex, telepresence are particularly interesting to consider.

Connecting to media issues

- Do we need to be able enumerate microphone / camera devices and select one? How to get the right one, how to name them, how to get default one? It is very common to have multiple cameras and hard to automatically choose.
- Deal with errors including:, failed, no permission, in use , partial success (audio but no video)
- Should we be able to switch devices mid call ?
- Does there have to be a video preview window when sending video?

ICE / DTLS

- Does the JS need to understand where in the ICE process the browser is, what errors it may be encountering, and be able to render progress to the user?
- Passing array of stun/turn other servers. How to deal with credentials? Pass in password on URL?
- Do we want JS to be notified of errors/events such as:
  - Can't gather address from one of servers
  - Fail to connect to turn server
  - Other side disconnects
  - Got new address, which address you got,
  - Connectivity checks
  - Got a check that works
  - Finished doing all check
  - Success of various connectivity checks, what path got used
- Connecting to an enterprise TURN or STUN sever (locally configured)

Signaling Issues

- When do you send codec signaling messages to the other side? How long to wait? Do we want an explicit API call to indicate things are ready to go? Renegotiation during the calls?
- Dealing with glare conditions?
- Is it SDP offer / answer based?
- Tracks: what are they?
  - Is video + audio, 1 or 2 tracks. Stereo audio: 1 or 2 tracks ?
  - Do we need to understand what codecs are in use and data rates?
  - Do we need to understand states: signaling not stared yet, we might receive media, we can send media, we can send and receive media. Basically when are we "connected"? Do we want the JS to be able to display to the user indication of connected state?
  - Can you renegotiate to add video mid call?
  - Do both sides understand when the renegotiation is completed?
  - Do we signaling an orderly close or just disappear and stop sending data ?

Signaling Issues #1

- Do we want to understand when the bit rate, quality, etc goes up or down for a track?
- When gateways to a SIP gateway to the PSTN, will the JS get information about ringing to connected state change ?
- When the JS gets a stream, how do you know if it is audio or video, or where to connect it to?
- When A calls B, does B needs the option to reject incoming call based on the fact that it is coming from A? Is B's IP address and location revealed to A if B rejects the call?
- How much do we implicitly rely on run to completion model vs explicitly API calls to tell the peerConnection object to move to next state?
- If both sides decide to add video at same time, how to to deal with glare and mark sure we reuse existing connections?

Msg Blob (thing passed via web signaling) issues

- Note: this is not the WhatWG recording blob, this is just the bag of bits that the JS app needs to pass back and forth to the other side
- We think we need the following in this blob: ice candidates, crypto context, what AV channels are bound to the crypto and who they are encrypted to
- Do we have to allow a design where server does not need to do any more than proxy msg to tother side ? i.e. message up is same format as message down
- How large can these blobs be? how large is SDP? Magnus?

Media issues

- How much do we want to separate the "media flow" from the "ICE connection"? - many flows can reuses a connection. How do we map all this into JS objects? Note: the lifetime of an RTP session is NOT the same as the lifetime of the ICE connection.
- Codec negotiation hints: screen size, spoken voice vs non spoken voice, spacial vs temporal quality (send QR code vs, vs arcade game) , max portion of bandwidth? min bandwidth? codec preference order ? (to avoid transcoding)
- Issues of representing hints, string, json object ,etc. Extensibility,
- Can we pause sending RTP packets and restart?
- Do we want to know the resolution of video we are receiving?
- Can we say send "blank" media?

Media Issues #2

- Packet loss stats? RTCP stats? Extensibility?
- Basic PLC ?
- PLC plug in architecture ?
- Plug in codec implemented in JS ?
- So this is media is encrypted. How to find who it is encrypted to and relevant information about crypto ?
- DTMF sending and receiving ?
- Can a peer send and receive audio and video from different IP address (all this still in single SDP offer / answer)?

Acknowledgments

- Many people contributed to these slides. Thanks to Cary Bran, Eric Rescorla, Harald Alvestrand, Matthew Kaufman, Stefan Håkansson, Timothy Terriberry, and Dzonatras Sol