WebRTC: Real-Time Communication for the Web

Timothy B. Terriberry
The Mozilla Corporation

What is WebRTC?

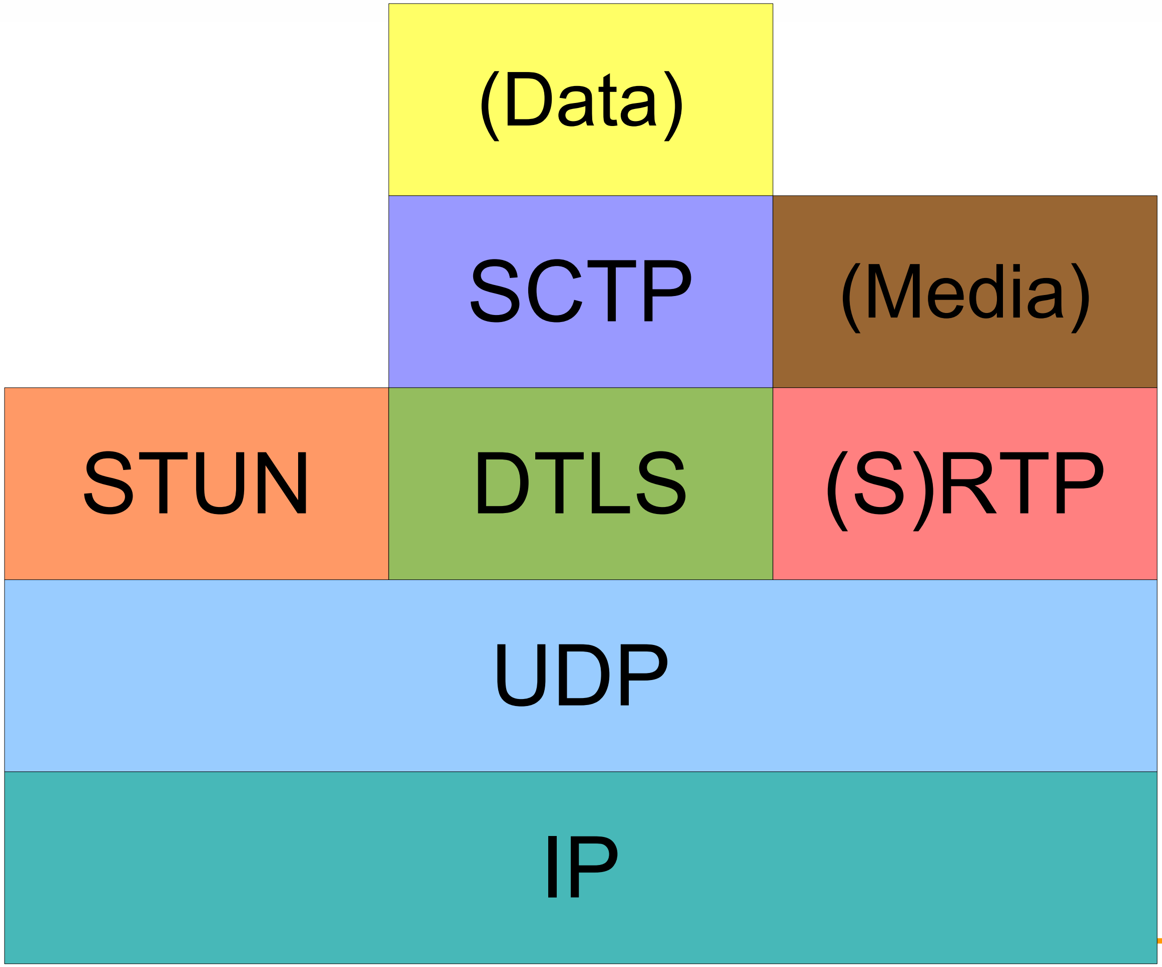
- Two-way audio and video chat, in a browser
  - Using open standards (under development)
    - W3C: WebRTC working group (APIs)
      - getUserMedia Task Force (camera input)
    - IETF: rtcweb working group (network protocols)
      - With other WGs as appropriate
- Media flows peer-to-peer
  - Allows browsers to exchange data directly with other browsers, without a web server in the middle
  - Also includes data channel (a p2p “websocket”)

How Will it Work?

- Firewall traversal: ICE (RFC 5245)
  - Uses STUN (RFC 5389) and TURN (RFC 5766)
- Session setup
  - Some form of SDP Offer/Answer (RFC 3264)
- Media: (S)RTP
  - With DTLS-SRTP (RFC 4347) for key exchange
  - Maybe also SDES (RFC 4568)
- Data channel: SCTP (RFC 4960) over DTLS
- All muxed on the same UDP port

Protocol Stack

- Ought to be enough layers for anybody



Firewall Traversal: ICE

- Complicated (117 page RFC), but basically
  - Try all the obvious things until something works
- STUN (RFC 5389)
  - Contact a server on public internet, it tells you your IP (from its perspective)
  - Includes short-term credential check
    - This is important for security!
- TURN (RFCs 5766, 6156)
  - Relay server to get around symmetric NATs

Session Setup

- Very controversial topic
  - Half the people want SIP (RFC 3261)
  - Half the people want Jingle (XEP-0166 draft)
  - Half the people want nothing defined at all
  - Some of the people are schizophrenic
- Likely based on SDP Offer/Answer
- Two competing proposals
  - ROAP: offer/answer state managed by browser
  - JSEP: offer/answer state managed by JS

Session Description Protocol (SDP)

- Describes everything needed to start talking
  - Media (types, codecs, initialization parameters, etc.)
  - Protocols (RTP profile, SRTP key exchange, etc.)
  - Transport (ICE candidates, muxing, etc.)
- Terrible agglomeration of years of crap
  - But we’re going to use it anyway
    - Would need years of standards work to replace
    - We’d have to map the result back to SDP anyway to interoperate with anything

RTCWeb Offer/Answer Protocol (ROAP)

- Minimal wrapper around SDP blobs
  - Gives context needed for offer/answer, e.g., “Is this an offer or an answer?”, etc.
  - Browser handles negotiation
- Defines interface between the browser and JS
  - *Not* a wire protocol, though with JSON it could be used to make one
- Maps to both SIP and Jingle relatively cleanly
  - Including forking, early media, etc.
- http://tools.ietf.org/html/draft-jennings-rtcweb-signaling-gateway-00

Javascript Session Establishment Protocol (JSEP)

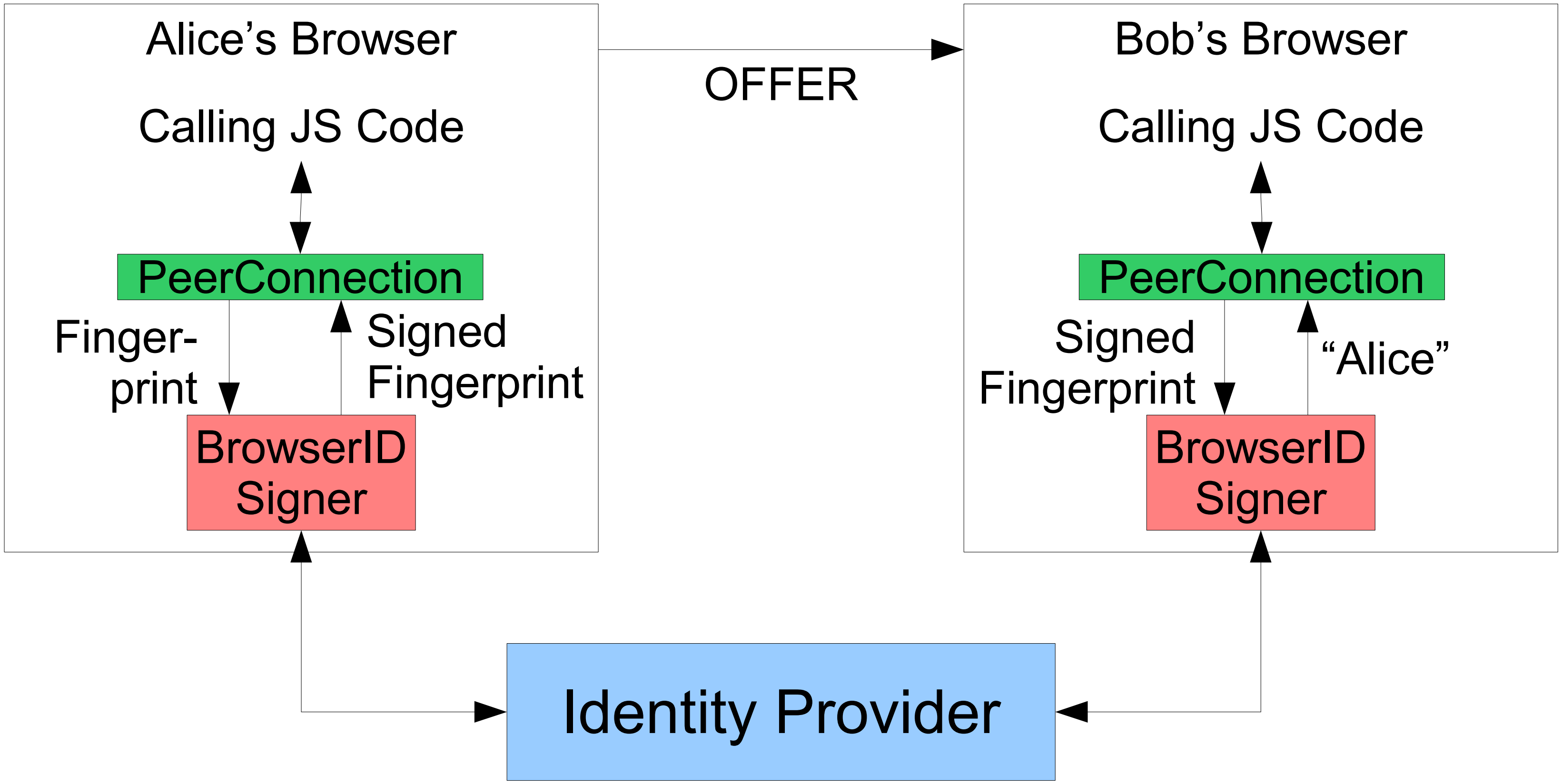
- Adds setLocalDescription() and setRemoteDescription() APIs
  - Informs the browser what SDP to use by fiat
  - JS handles the negotiation
  - Separates out ICE state machine from SDP
    - Must remain in browser for security
- Gives more flexibility to negotiation process
  - Example: Trickle ICE candidates
- http://lists.w3.org/Archives/Public/public-webrtc/2012Jan/0002.html

Media: (S)RTP

- We think media should be secure by default
  - We don’t trust the network, the web application, the signaling service... or anyone, really
- Current debates
  - Whether to allow plain RTP at all
    - Bid-down attacks possible if we do
  - What keying mechanisms to use for SRTP
    - SDES: keys exchanged in SDP for all to see, but (naturally) the most commonly implemented
    - DTLS-SRTP: “Datagram” version of TLS

Security: Identity

- Preventing MITM: sign DTLS keys with trusted identity provider (e.g., Mozilla's BrowserID)



Media: Codecs

- *Much* more important to have a mandatory to implement (MTI) codec than with <video> tag
  - No common format with <video> is a problem, but a solvable one
    - Server can host files in multiple formats
  - No common format with WebRTC means people can't communicate at all
- Everyone expects this to be contentious, but little real discussion so far

Chris Blizzard: “We’re fer serious about Royalty-Free”

- Video: VP8
  - Google has been enhancing libvpx for this
    - Better real-time encoding
    - Error resilience
    - Temporal scalability
  - No serious objections... yet
    - Talk of a royalty-free version of H.264 baseline (WebVC at MPEG), but unclear if field-of-use restrictions, compatibility, etc., will make this feasible
- Audio: G.711, Opus
  - See Jean-Marc Valin’s talk tomorrow at 16:40

Data Channel: Stream Control Transmission Protocol (SCTP)

- *Need* an unreliable p2p data protocol
  - It won't be real-time if it isn't
- *Want* a reliable one as well
  - If we don't provide one, everyone will make their own... badly
- SCTP provides both, and has an open-source user-space implementation
  - Other proposals (DCCP-based, etc.) exist, but SCTP is most promising

Congestion Control

- Loss-based (TCP-style) congestion control doesn’t work for real-time media
  - Buffers have to fill before you see any loss... meaning you’ll have lots of delay
  - Random Early Detection (RED), etc., rare
- Currently using a delay-based mechanism
  - Currently per-stream, want something per-peer
  - That may need new RTCP feedback messages
  - IETF “rmcat” WG forming: http://www.ietf.org/iesg/evaluation/rmcat-charter.txt

APIs: getUserMedia()

- Used to get camera/microphone access

```
interface NavigatorUserMedia {
    void getUserMedia(MediaStreamOptions? options,
        NavigatorUserMediaSuccessCallback? successCallback,
        optional NavigatorUserMediaErrorCallback? errorCallback);
};

interface NavigatorUserMediaSuccessCallback {
    void handleEvent(LocalMediaStream stream);
};
```

- Issues: camera/microphone selection, resolution, sampling rate, etc.
- Separated out from main WebRTC API, see http://dev.w3.org/2011/webrtc/editor/getusermedia.html

APIs: MediaStream

- Provides a means of routing around *synchronized* media data

```
interface MediaStream {
    readonly attribute DOMString            label;
    readonly attribute MediaStreamTrackList audioTracks;
    readonly attribute MediaStreamTrackList videoTracks;
             attribute boolean              ended;
             attribute Function?            onended;
    MediaStreamRecorder record();
};
```

- No access to data itself: feed to <video>, PeerConnection, etc.
  - ProcessedMediaStream proposed by Robert O'Callahan for this

APIs: PeerConnection

- Feeds MediaStreams, etc., to remote peer

```
interface PeerConnection {
    void processSignalingMessage(DOMString message);
    void addStream(MediaStream stream, MediaStreamHints hints);
    void removeStream(MediaStream stream);
    void close();
    readonly attribute MediaStream[] localStreams;
    readonly attribute MediaStream[] remoteStreams;
    /*... lots of state attributes/callbacks omitted ...*/
};
```

- Signaling messages (e.g., SDP/ROAP/etc.) exchanged with web server via XHR, WebSockets, etc.
- http://dev.w3.org/2011/webrtc/editor/webrtc.html

Status and Schedule

- Implemented in dev version of Chrome behind a flag this week (for real this time!)
- Test builds from Mozilla in Q1
  - Shipping in mozilla-central later this year
- This is experimental: things *will* change
- Initial implementations don't have ROAP/JSEP, SCTP data channel, Opus, per-peer congestion control many other things
  - But we'll get there

Get Involved

- IETF drafts: http://tools.ietf.org/wg/rtcweb/
- IETF mailing lists:
  https://www.ietf.org/mailman/listinfo/rtcweb
  http://www.alvestrand.no/mailman/listinfo/rtp-congestion
- W3C mailing lists:
  http://lists.w3.org/Archives/Public/public-webrtc/
  http://lists.w3.org/Archives/Public/public-media-capture/
- Open-source media backend:
  http://www.webrtc.org/
- #media on irc.mozilla.org

Questions?