WebRTC

The Web Way to Communicate

Marcel Großmann

Professorship for Computer Science,
Communication Services, Telecommunication Systems
and Computer Networks

4. February 2014

Content

Outline

- NVP (Network Voice Protocol) - 1977
- RTP (Real-time Transport Protocol) -1992
  - First published as IETF RFC in 1996
  - Still used today for VoIP and with SIP
- ITU H.323 video telephony standard - 1996
- IETF SIP (Session Initiation Protocol) - 1999
  - Unleashed VoIP revolution on telphony
  - Video and room conferencing
  - Protocol widely used by service providers and in enterprises
- Real-Time Communcation on the Web
  - Voice and video on the Internet using browser plugins
  - 2006 with GoogleTalk inside Gmail
  - WebRTC standardizes and eliminates need for plugin or download

Why not just use Flash?

Pros

- Most browser already have Flash plugin
- Streaming audio and video uses Flash today
- Flash supports real-time audio and video
- Web developers familiar with Flash

Cons

- Flash is single-vendor proprietary and closed
- Losing market share and not available on iOS
- Limited codec and echo cancellation options
- Proprietary development tools

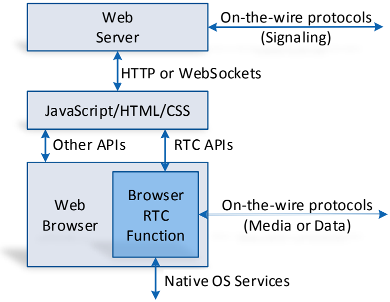
WebRTC is “Skype in the browser”

- Access to camera and microphone without a plugin ($\rightarrow$ No more Flash!)
- Audio/video direct from browser to browser
- Why does it matter?
  - Media can stay local
  - Mobile devices eventually dropping voice channel anyway
  - Games

Outline

The Browser RTC Function

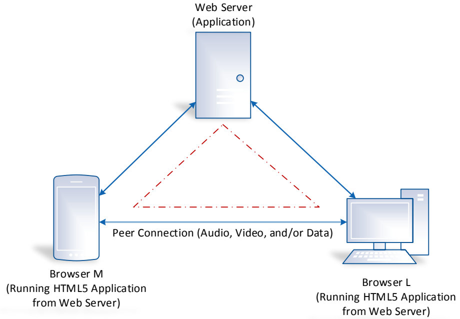
- New Browser Real-Time Communication (RTC) Function built-in to browsers
- Contains
  - Audio and video codecs
  - Ability to negotiate peer-to-peer connections
  - Echo cancellation, packet loss concealment
- In Chrome and Mozilla today



So What's the Big Deal

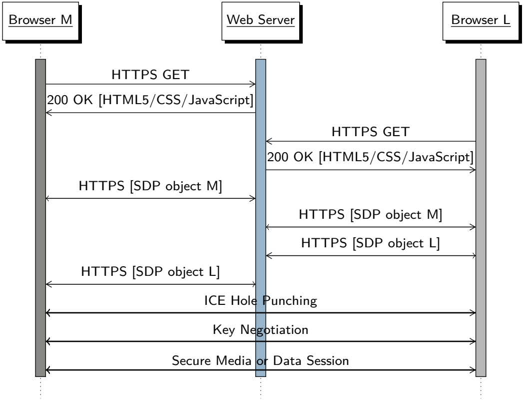
- The web is now a platform for real-time communications development
- Communication will be secure (encrypted) by default
- Latest audio and video codecs means superior quality to anything else
- WebRTC provides peer-to-peer media, even through NATs
- Standard that can interoperate with existing VoIP, video conferencing, and even PSTN

WebRTC Triangle

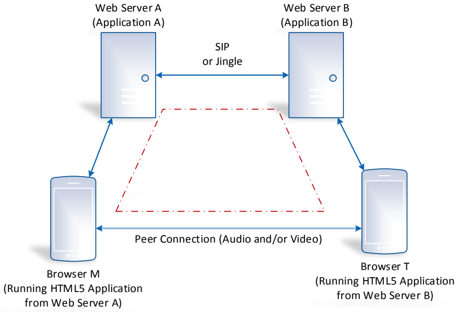


- Both browsers running the smae web application from web server
- Peer Connection media session is establised between them
- Signaling is not standardized, could be SIP, Jingle, proprietary. Uses HTTP or WebSockets for transport

WebRTC Triangle Call Flow

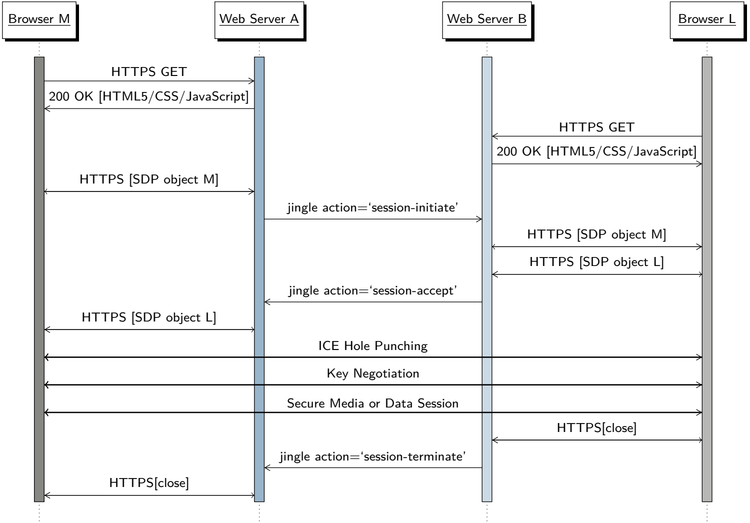


WebRTC Trapezoid

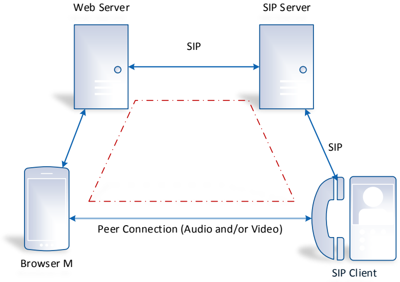


- Similar to SIP Trapezoid
- Web Servers communicatie using SIP or Jingle
- Useful for building conventional telephony apps, but unclear how this works in the web world

WebRTC Tapezoid Call Flow with Jingle

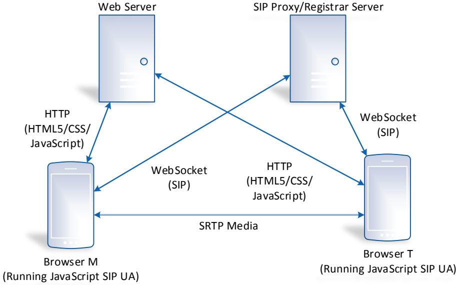


WebRTC and SIP



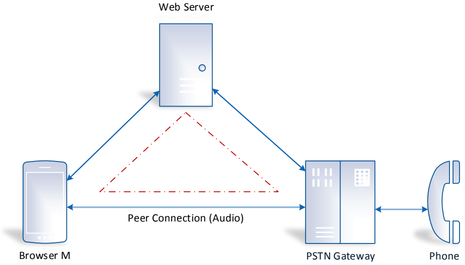
- Peer Connection appears as a standard RTP media session, described by SDP
- Web Server implements a JSEP (JavaScript Session Establishment Protocol) to SIP signaling gateway
- SIP Endpoint must support RTCWEB Media extensions (ICE NAT Traversal, Secure RTP, etc.)

WebRTC with SIP



- Browser runs a SIP User Agent by running JavaScript form Web Server
- Secure RTP media connection uses WebRTC APIs
- Details in [draft-ietf-sipcore-websocket] that defines SIP transport over WebSockets

WebRTC and PSTN



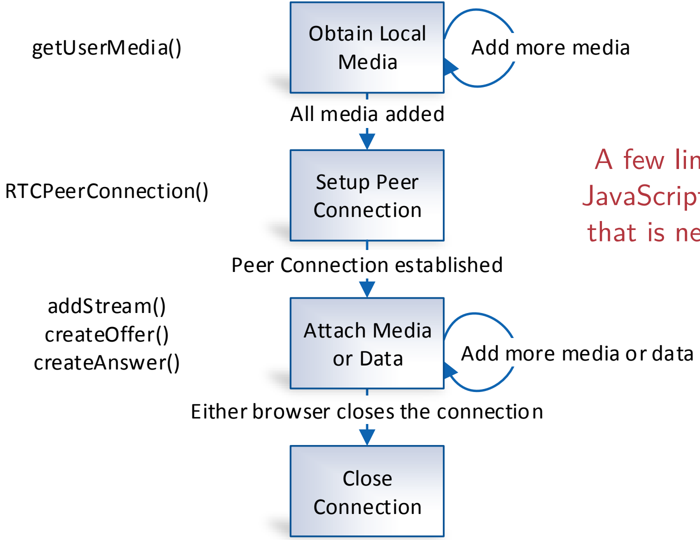
- Peer Connection terminates ona PSTN Gateway
- Audio only
- Could also use SIP trunking such as SIPconnect 1.1 recommendation

WebRTC Support fo Multiple Media

- Multiple sources of audio and video are assumed and supported
- All RTP media, voice and video, and RTCP feedback messages are multiplexed over the same UDP port and address

Outline

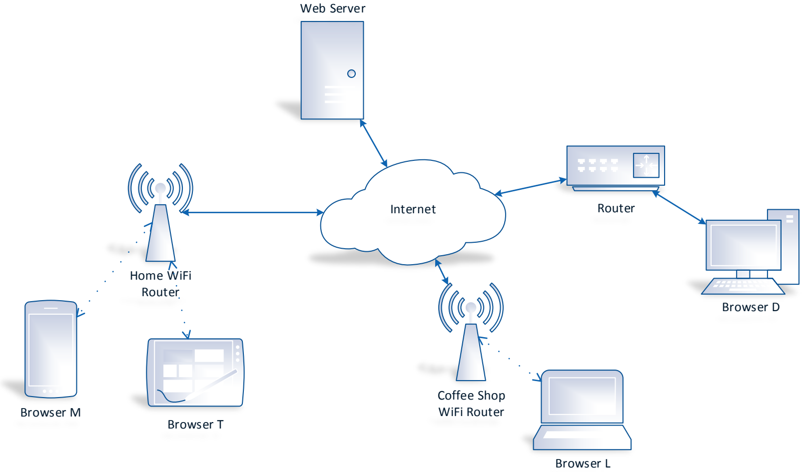
How to use WebRTC



A few lines of JavaScript is all that is needed!

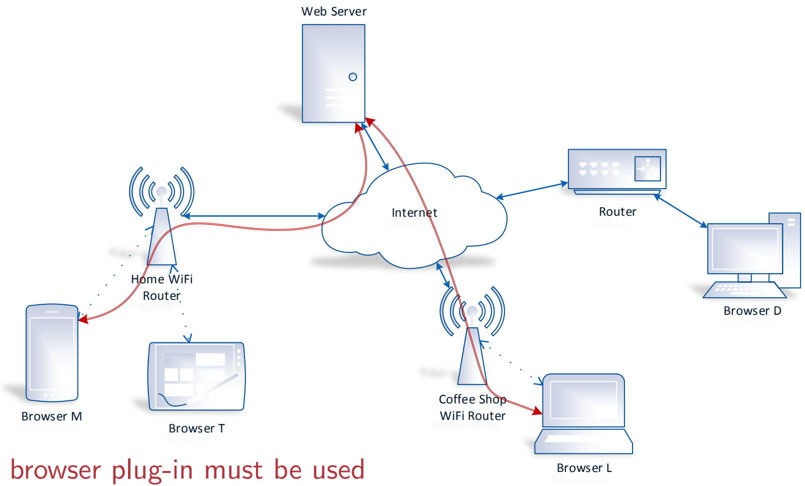
WebRTC Peer-to-Peer Media

Media Flows in WebRTC



WebRTC Peer-to-Peer Media

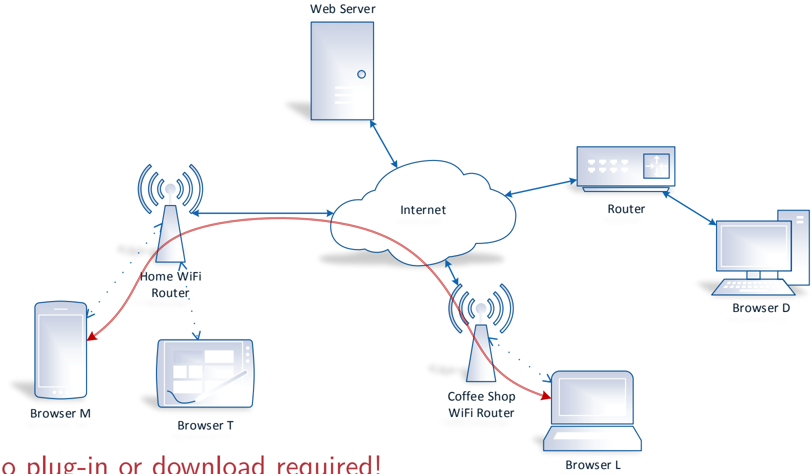
Media without WebRTC



A browser plug-in must be used

WebRTC Peer-to-Peer Media

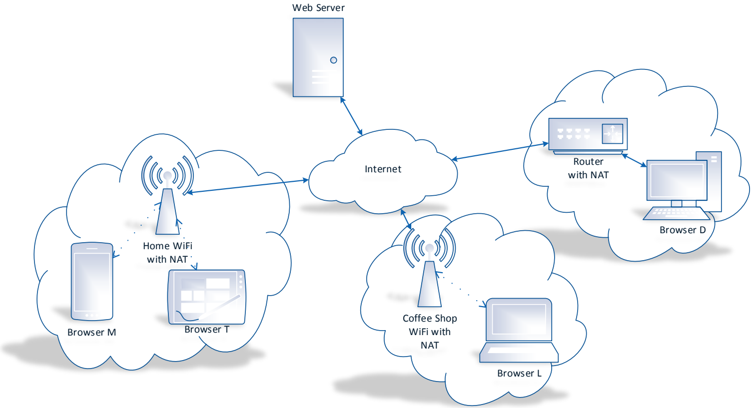
Peer-to-Peer Media with WebRTC



No plug-in or download required!

WebRTC Peer-to-Peer Media

NAT Complicates Peer-to-Peer Media

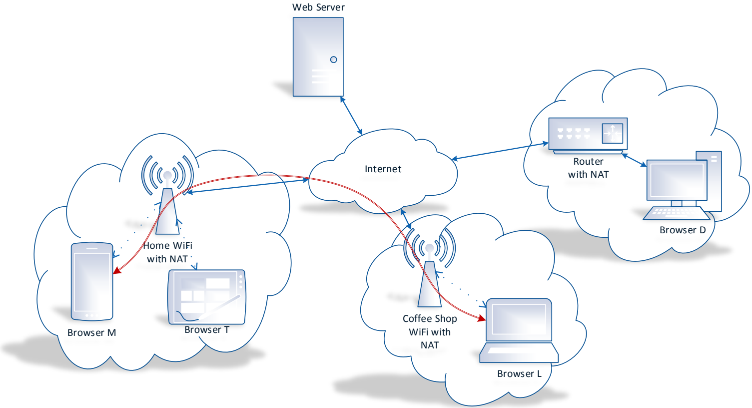


Network Address Translator

Most browsers are behind NATs on the Internet, which complicates the establishment of peer-to-peer media sessions

WebRTC Peer-to-Peer Media

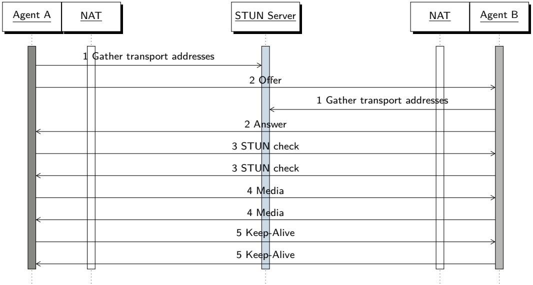
NAT Media Through NAT



Interactive Communications Establishment, RFC 5245

ICE hole punching can often establish a direct peer-to-peer session between browsers behind different NATs

High Level ICE Call Flow



Transport address is IP address and port number

1. Gather candidate transport addresses
2. Exchange candidates over signaling channel
3. Perform connectivity checks
4. Choose selected pair and begin media transport
5. Send keep-alives

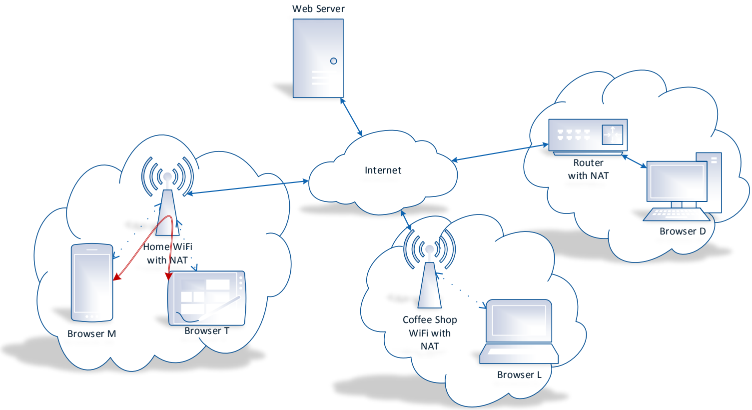
If either side detects a change iin

IP address in use, ICE is restarted

(back to step 1)

WebRTC Peer-to-Peer Media

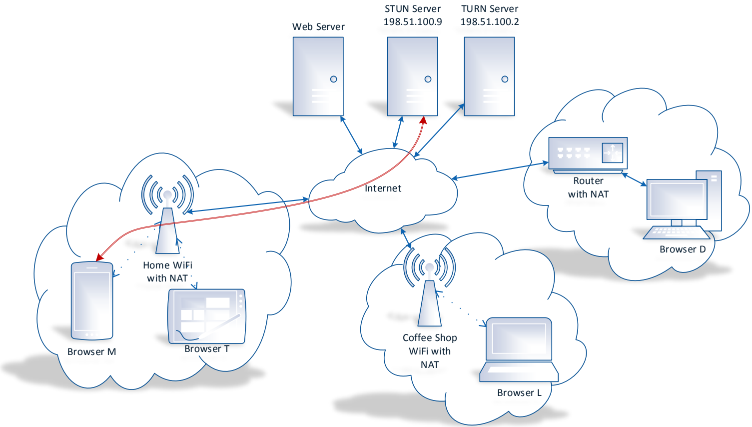
P2P Media Can Stay Local to NAT



If both browsers are behind the same NAT, hole punching can often establish a connection that never leaves the NAT

WebRTC Peer-to-Peer Media

Browser Queries STUN Server



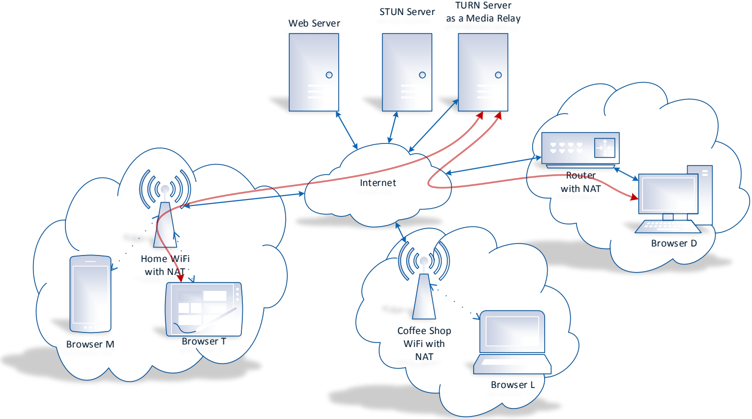
Session Traversal Utilities for NAT, RFC 5389

Browser sends STUN test packet to STUN server to learn its public IP address (address of the NAT)

WebRTC Peer-to-Peer Media

University of Bamberg

TURN Server Can Relay Media



Traversal of UDP
aRound NAT, RFC 5766

In some cases, hole punching fails, and a TURN Media Relay on the public Internet must be used.

Outline

Media Stream

aka getUserMedia

- Represents synchronized streams of media
  - taken from camera and microphone input
  - with synchronized video and audio tracks
- Each MediaStream has an input generated by `navigator.getUserMedia()`
- and an output, which might be passed to a video element or a RTCPeerConnection.

The `getUserMedia()` method takes three parameters:

- A constraints object.
- A success callback which, if called, is passed a MediaStream.
- A failure callback which, if called, is passed an error object.

Each MediaStream has a label. An array of MediaStreamTracks is returned by the `getAudioTracks()` and `getVideoTracks()` methods.

Media Stream

An Audio Example

```
function gotStream(stream) {
    window.AudioContext = window.AudioContext || window.webkitAudioContext;
    var audioContext = new AudioContext();

    // Create an AudioNode from the stream
    var mediaStreamSource = audioContext.createMediaStreamSource(stream);

    // Connect it to destination to hear yourself
    // or any other node for processing!
    mediaStreamSource.connect(audioContext.destination);
}

navigator.getUserMedia({audio:true}, gotStream);
```

RTCPeerConnection

Signaling is used to exchange three types of information

- Session control messages: to initialize or close communication and report errors.
- Network configuration: to the outside world, what's my computer's IP address and port?
- Media capabilities: what codecs and resolutions can be handled by my browser and the browser it wants to communicate with?

The exchange of information via signaling must have completed successfully before peer-to-peer streaming can begin.

RTCPeerConnection I


```
var signalingChannel = createSignalingChannel();
var pc;
var configuration = ...;
// run start(true) to initiate a call
function start(isCaller) {
    pc = new RTCPeerConnection(configuration);
    // send any ice candidates to the other peer
    pc.onicecandidate = function (evt) {
        signalingChannel.send(JSON.stringify({ "candidate": evt.candidate }));
    };
    // once remote stream arrives, show it in the remote video element
    pc.onaddstream = function (evt) {
        remoteView.src = URL.createObjectURL(evt.stream);
    };
```

RTCPeerConnection II


```
    // get the local stream, show it in the local video element and send it
    navigator.getUserMedia({ "audio": true, "video": true }, function (stream) {
        selfView.src = URL.createObjectURL(stream);
        pc.addStream(stream);

        if (isCaller)
            pc.createOffer(gotDescription);
        else
            pc.createAnswer(pc.remoteDescription, gotDescription);

        function gotDescription(desc) {
            pc.setLocalDescription(desc);
            signalingChannel.send(JSON.stringify({ "sdp": desc }));
        }
```

RTCPeerConnection III


```
    });
}

signalingChannel.onmessage = function (evt) {
    if (!pc)
        start(false);

    var signal = JSON.parse(evt.data);
    if (signal.sdp)
        pc.setRemoteDescription(new
            RTCSessionDescription(signal.sdp));
    else
        pc.addIceCandidate(new RTCIceCandidate(signal.
            candidate));
};
```

RTCDataChannel

There are many potential use cases for the API, including

- Gaming
- Remote desktop applications
- Real-time text chat
- File transfer
- Decentralized networks

The API has several features to make the most of RTCPeerConnection and enable powerful and flexible peer-to-peer communication:

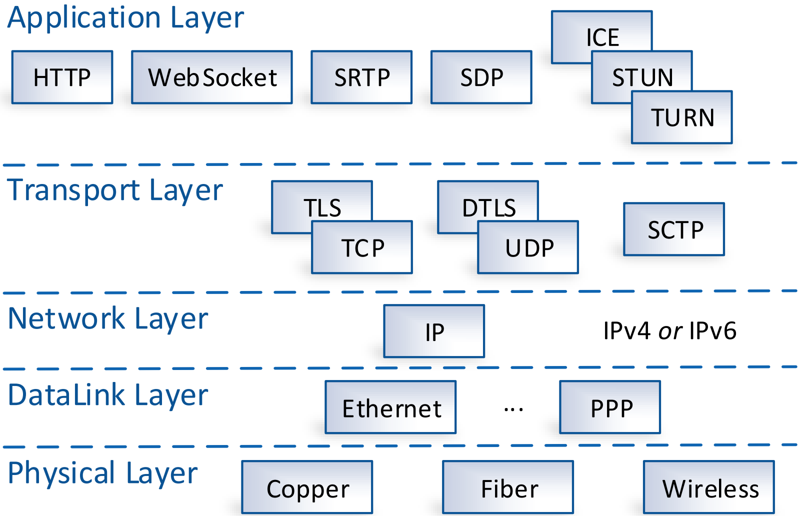
- Leveraging of RTCPeerConnection session setup.
- Multiple simultaneous channels, with prioritization.
- Reliable and unreliable delivery semantics.
- Built-in security (DTLS) and congestion control.
- Ability to use with or without audio or video.

RTCDataChannel


```
var pc = new webkitRTCPeerConnection(servers,
  {optional: [{RtpDataChannels: true}]});

pc.ondatachannel = function(event) {
  receiveChannel = event.channel;
  receiveChannel.onmessage = function(event){
    document.querySelector("div#receive").innerHTML =
        event.data; };};

sendChannel = pc.createDataChannel("sendDataChannel",
    {reliable: false});

document.querySelector("button#send").onclick =
    function (){
  var data = document.querySelector("textarea#send").
      value;
  sendChannel.send(data);};
```

Outline

WebRTC Protocols

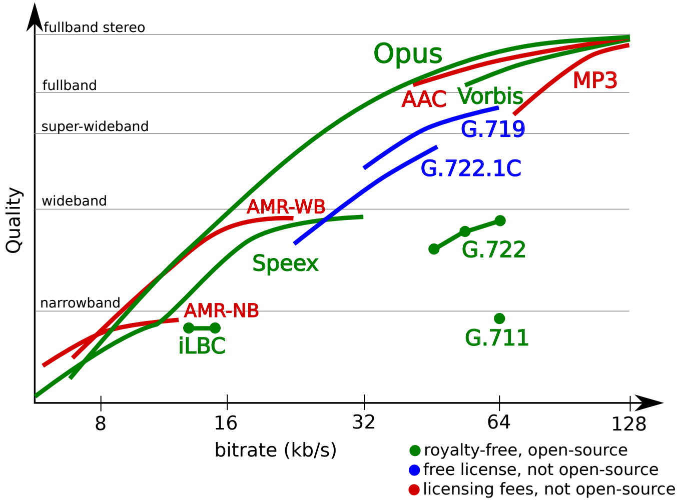


A Joint Standards Effort

- World Wide Web Consortium (W3C)
    - Standardizing APIs
    - Most work in WEBRTC Working Group
    - Used by JavaScript to access RTC function
- Internet Engineering Task Force (IETF)
    - Standardizing protocols (bit on wire)
    - Codecs
    - Peer Connection will use RTP, SDP, and extensions
    - Some work in RTCWEB Working Group
    - Lots of related work in MMUSIC, AVTCORE, etc.

OPUS Audio Codec

Comparison



Standard Codecs in WebRTC

Codec	Use	Specification
Opus	Narrowband to wideband Internet audio codec for speech and music	RFC 6716
G.711	PCM audio encoding for PSTN interworking and backwards compatibility with VoIP systems	RFC 3551
Telephone Events	Transport of Dual Tone Multi Frequency (DTMF) tones	RFC 4733
H.264	Video codec requiring licensing	RFC 6184
VP8	Open source video codec	RFC 6386

- Mandatory to Implement (MTI) audio codecs settled on Opus and G.711 (finally!)
- Video is not yet settled (H.264 vs. VP8 fight!)

WebRTC and the Enterprise

- Enterprise has unique requirements on WebRTC
- Security
  - Firewall traversal
  - Access control
  - Peer-to-peer data flows
- Compliance
  - Recording & logging
  - Policy compliance
- Integration with existing infrastructure
- New element proposed
  - "Secure Edge" located in enterprise DMZ

What's Next?

- W3C and IETF standards still need to be finalized
- Browsers need to add support
  - Chrome and Firefox browser have much of this functionality now!
  - Their mobile derivatives are almost on the same level.
- Mandatory to Implement video codec needs to be decided
- Enterprise use of WebRTC need to be worked out

References

[1] A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Digital Codex LLC, 2013. [Online]. Available: http://www.amazon.com/WebRTC-RTCWEB-Protocols-HTML5-Real-Time/dp/098597883X%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D098597883X

[2] R. Manson, *Getting Started with WebRTC*. Packt Publishing, 2013. [Online]. Available: http://www.amazon.com/Getting-Started-WebRTC-Rob-Manson/dp/1782166300%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1782166300

Questions ?

Marcel Großmann
marcel.grossmann@uni-bamberg.de