





# WebRTC

Plugin-free realtime communication

Sam Dutton  
Developer Advocate, Google Chrome

[@sw12](https://twitter.com/sw12)

[gowebrtc.appspot.com](https://gowebrtc.appspot.com)

---

[github.com/sambutton/gowebRTC](https://github.com/sambutton/gowebRTC)

“WebRTC is a new front in the long war  
for an open and unencumbered web”

Brendan Eich  
– Mozilla CTO and inventor of JavaScript



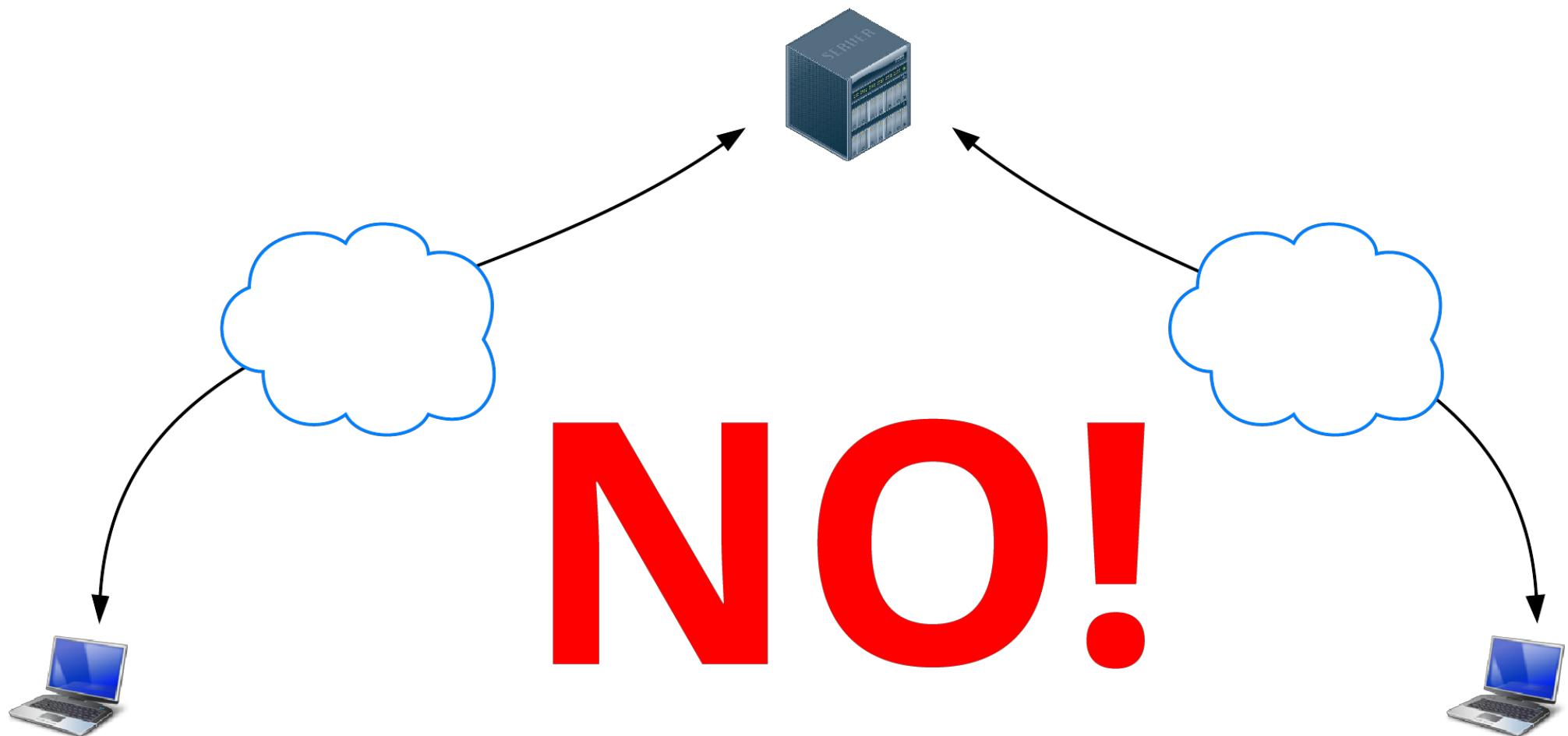
Real-time communication built into the browser

Open source, no plugins, free

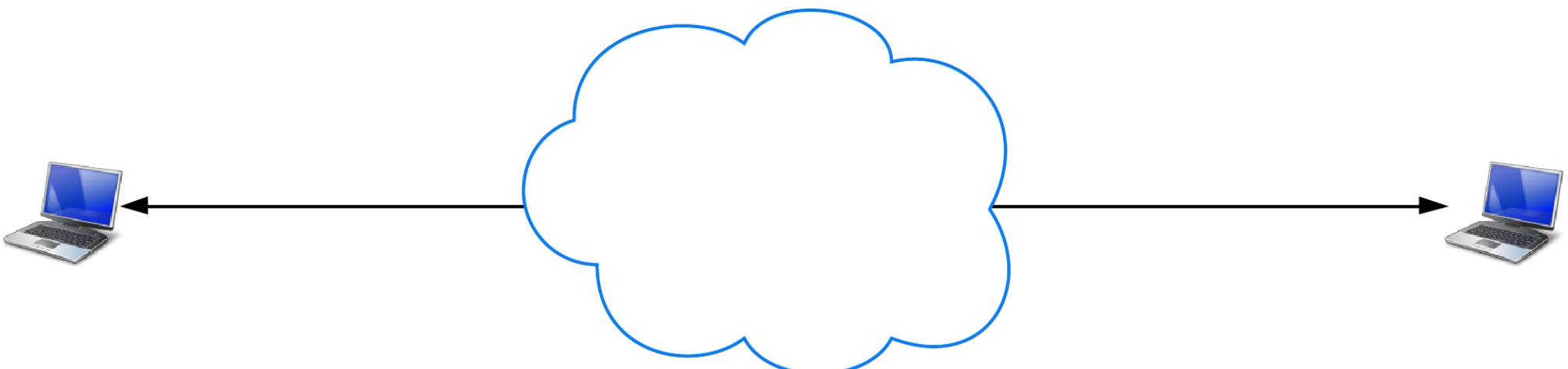
Video, audio, data

High quality, low cost

# Peer to peer



# JA!

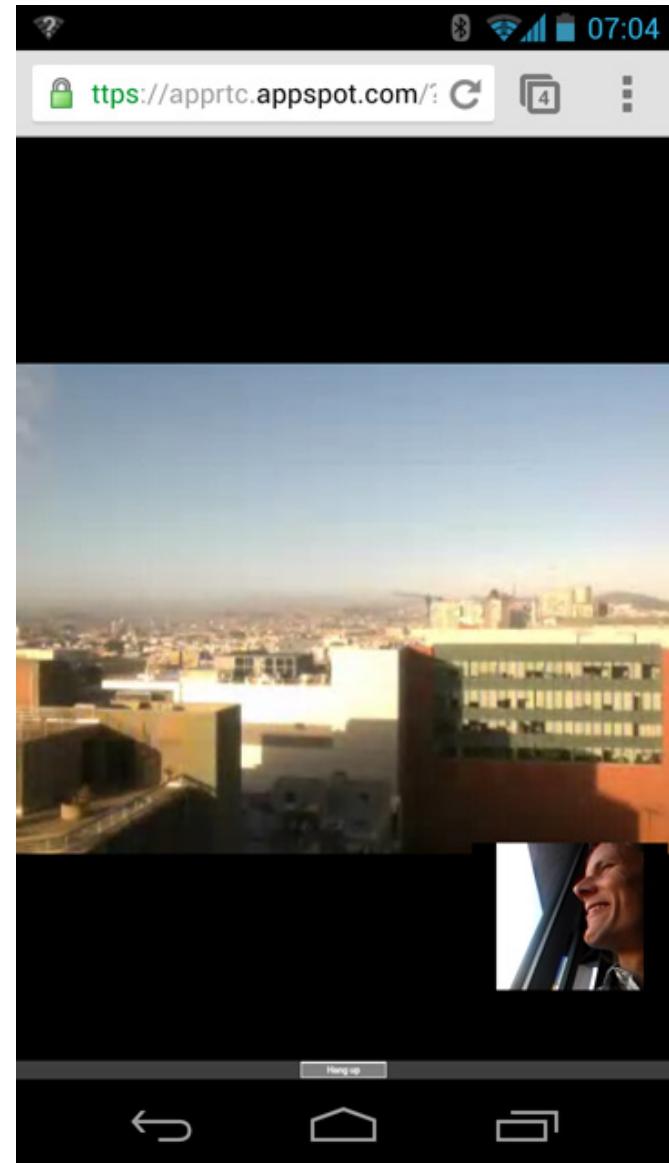


# Shopping list

- IETF communication protocols: [RTCWEB](#), [JSEP](#)...
- W3C API standards: [WebRTC](#), [Media Capture and Streams](#)...
- Media and communication stack: [libjingle](#), [VP8](#), [Opus](#)...
- Design for a new communications ecosystem

# WebRTC across platforms

- Chrome and Chrome for Android
- Firefox and Firefox for Android
- Opera
- Native Java and Objective-C bindings



# Qt moving to Chromium

- Framework for cross-platform/device native and embedded apps
- Qt WebKit => Qt WebEngine
- Multimedia and new HTML5 features such as WebRTC working out-of the-box

**1,000,000,000+**

WebRTC endpoints

“Voice is just another JS application”

Henning Schulzrinne  
– CTO, US FCC



sky NEWS FTSE 6457.78

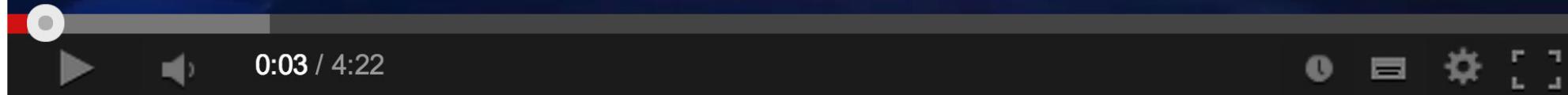


SKY NEWS CENTRE | LIVE | SOUTH LONDON

15:37 | TAKEN TO A MILITARY HOSPITAL

| BREAKING NEWS

| AP: POLICE SAY TWO ROCKETS FIRED FRO









What do we need for RTC?

# Three main tasks

- Acquiring audio and video
- Communicating audio and video
- Communicating arbitrary data

# Three main JavaScript APIs

- MediaStream (aka getUserMedia)
- RTCPeerConnection
- RTCDataChannel



# MediaStream

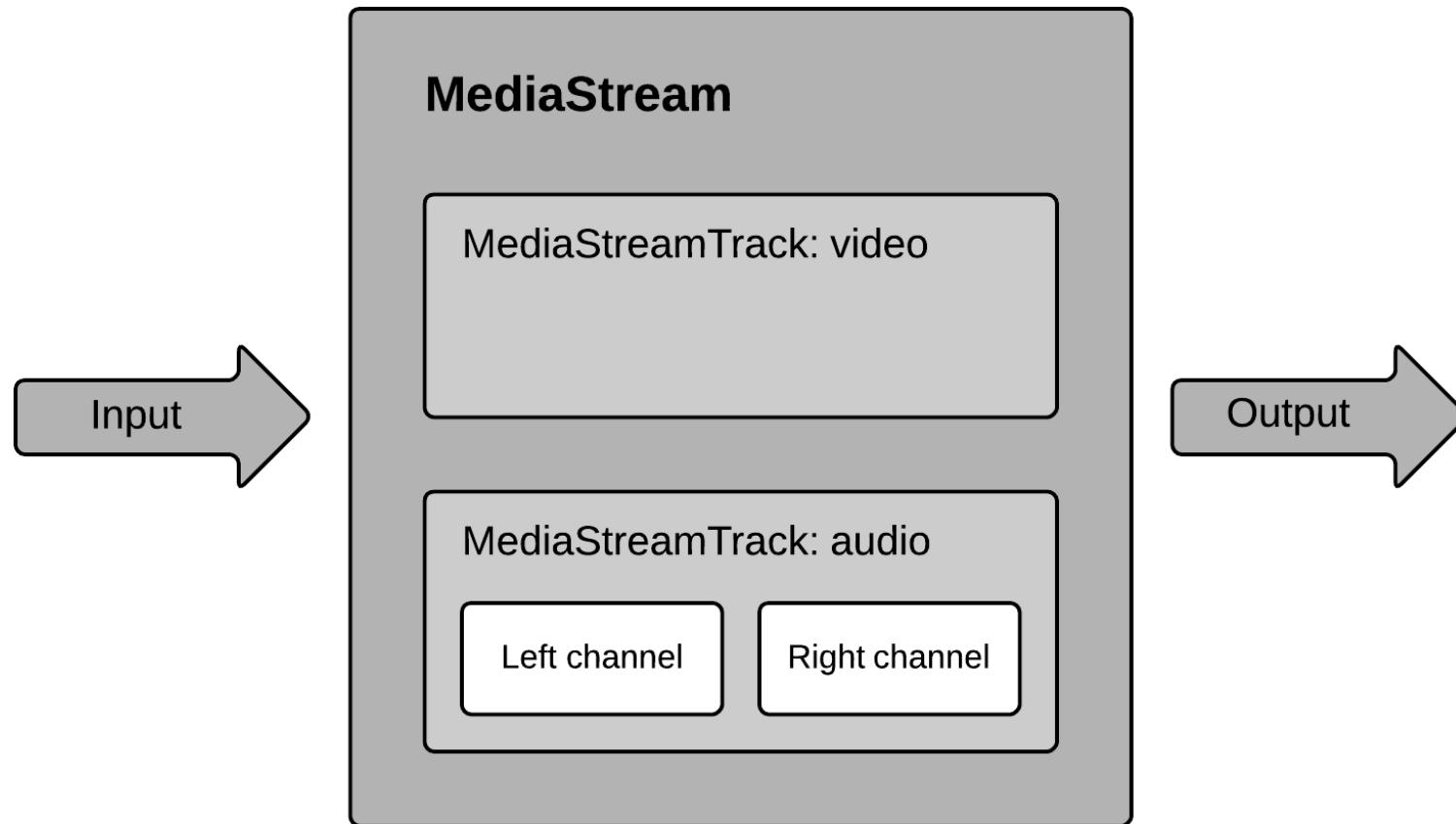
Acquiring audio and video

# What do we need?

- Streaming media: `MediaStream`
- Get local media: `navigator.getUserMedia()`

# MediaStream

- Represents a stream of synchronised media
- Can contain multiple audio and/or video MediaStreamTracks
- Obtain a MediaStream with `navigator.getUserMedia()`



# gUM

It's pretty simple.

```
var constraints = {video: true};
```

JAVASCRIPT

```
function successCallback(stream) {
  var video = document.querySelector("video");
  video.src = window.URL.createObjectURL(stream);
}
```

```
function errorCallback(error) {
  console.log("navigator.getUserMedia error: ", error);
}
```

```
navigator.getUserMedia(constraints, successCallback, errorCallback);
```

[simpl.info/gum](http://simpl.info/gum)

---

# gUM permissions

- HTTPS
- Chrome apps: `audioCapture` and `videoCapture` permissions
- Chrome flag: `--use-fake-ui-for-media-stream`
- UI settings can be changed afterwards.



[idevelop.github.com/ascii-camera](https://idevelop.github.com/ascii-camera)

FaceKat

[html5-demos.appspot.com/static/getusermedia/photobooth.html](http://html5-demos.appspot.com/static/getusermedia/photobooth.html)

[webcamtoy.com](http://webcamtoy.com)

# Constraints

- Mandatory or optional
- Resolution: width and height
  - from a fixed list
  - no cropping or scaling (yet)
- Frame rate
- Facing mode: front or back camera
- Source type: video camera, screen capture...
- Source id
- Volume

[simpl.info/getusermedia/constraints](https://simpl.info/getusermedia/constraints)

[simpl.info/getusermedia/sources](https://simpl.info/getusermedia/sources)

# Facing mode, applyConstraints()

- Choose source: [spec](#)
- Apply constraints from JavaScript: [spec](#)

# getUserMedia + Web Audio

```
// Success callback when requesting audio input stream
function gotStream(stream) {
    var audioContext = new webkitAudioContext();

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

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

}

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

JAVASCRIPT

Make sure to enable Web Audio Input in about:flags!

[webaudiodeemos.appspot.com/](http://webaudiodeemos.appspot.com/AudioRecorder)[AudioRecorder](http://webaudiodeemos.appspot.com/AudioRecorder)

# gUM screencapture

```
var constraints = {  
  video: {  
    mandatory: {  
      chromeMediaSource: 'screen'  
    }  
  }  
};  
  
navigator.webkitGetUserMedia(constraints, gotStream);
```

JAVASCRIPT

Screen sharing

Tab capture: chrome.tabCapture

# Media Stream Recording API

- Demo: [simpl.info/mediarecorder](http://simpl.info/mediarecorder)
- Spec [Spec](#)
- Chrome [Intent to Implement](#)
- [Streams API](#)

# Media Stream Image Capture API

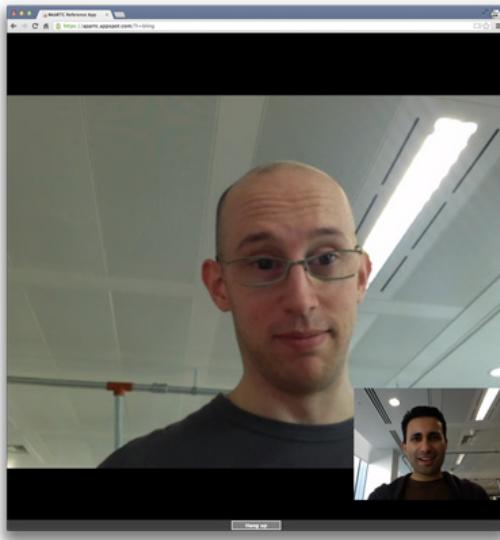
- Demo
- [Spec](#)
- `getFrame()` creates an `ImageData` object available in `onframegrab`
- `takePhoto()` creates a Blob available in `onphoto`



# RTCPeerConnection

Audio and video communication between peers

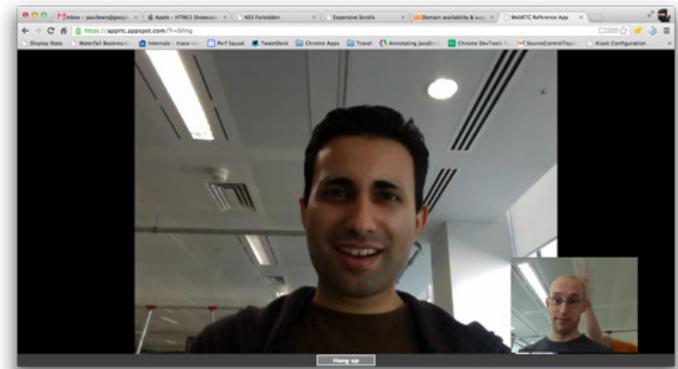
# Communicate Media Streams



getUserMedia

+

RTCPeerConnection

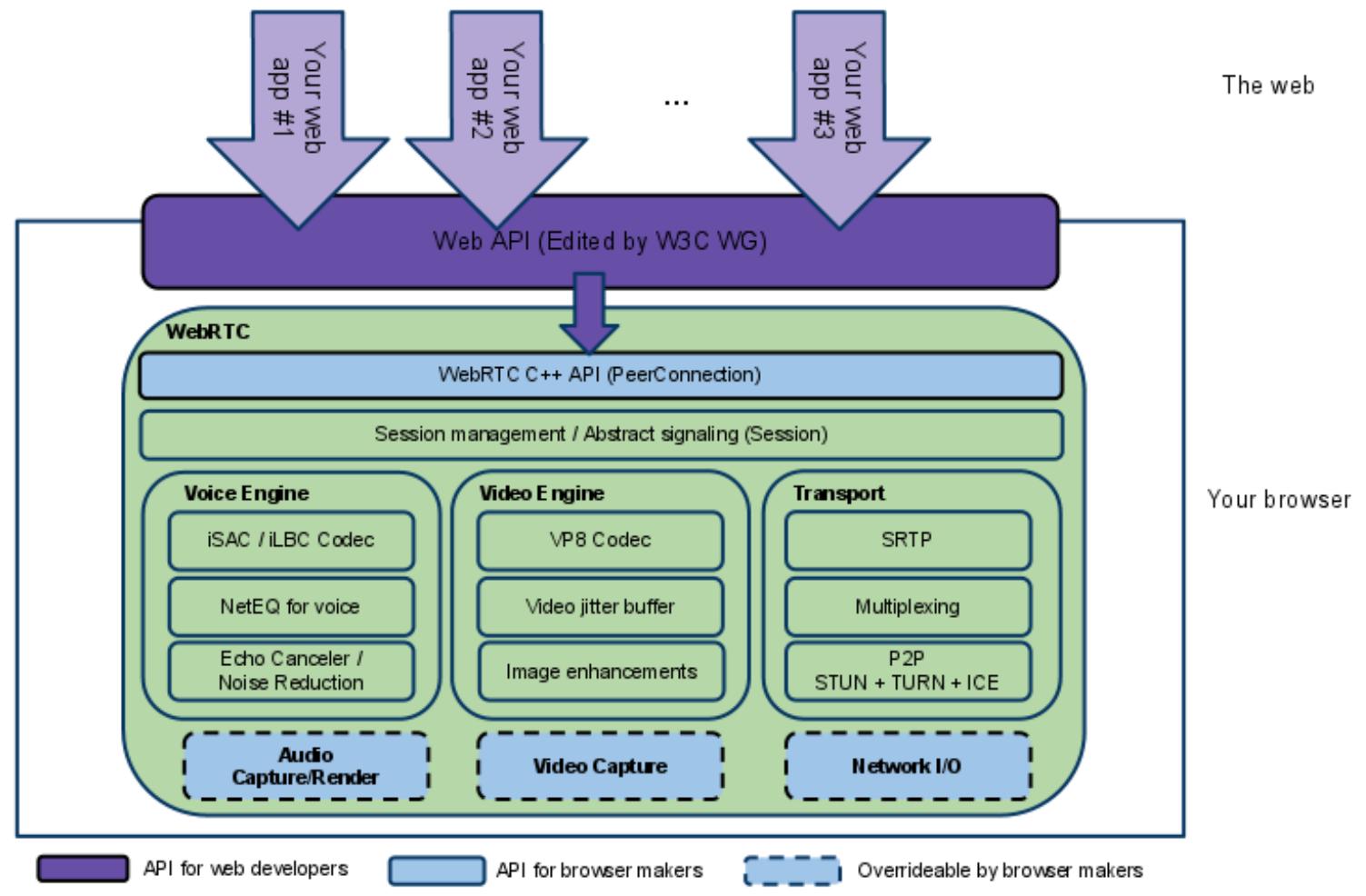


# RTCPeerConnection does a lot

- Signal processing
- Codec handling
- Peer to peer communication
- Security
- Bandwidth management

...

# WebRTC architecture

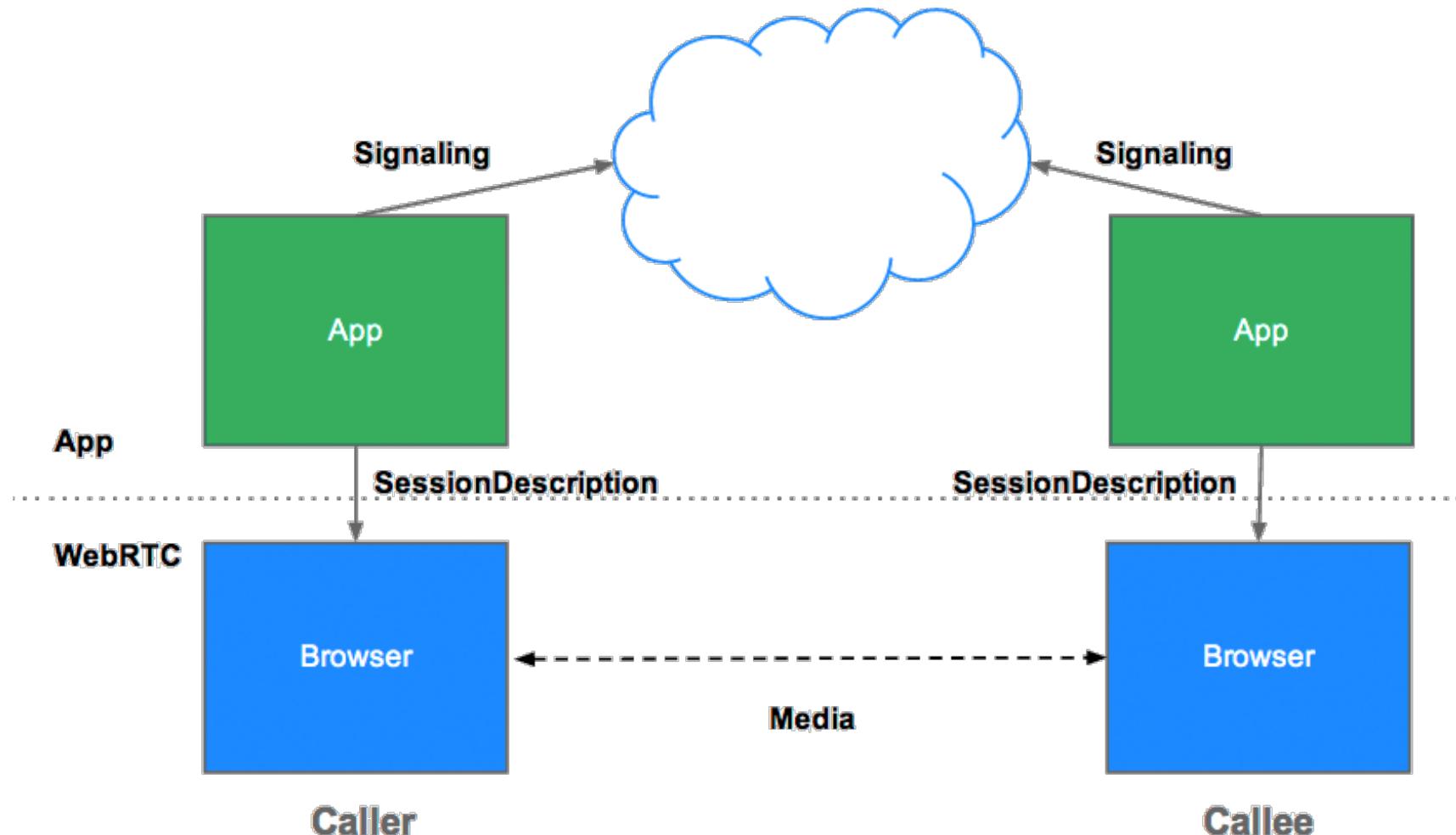


Peer to peer — but we need servers :^\\

# What does WebRTC need servers for?

- Exchange metadata to coordinate communication: signaling
- Cope with NATs and firewalls: STUN and TURN

# JSEP architecture



# Signaling

- Need to exchange 'session description' objects:
  - What media formats I support, what I want to send
  - Network information for peer-to-peer setup
- Can use any messaging mechanism
- Can use any messaging protocol
- ...and can be used for application data

# Signaling: how?

- Needs to be bidirectional
- Repeated polling: inefficient and not scalable
- Long polling / Comet (as used by Google App Engine)
- XHR + EventSource (aka Server-sent events): demo
- WebSocket:
  - more natural solution — it's bidirectional!
  - supported by all browsers that support WebRTC, desktop and mobile
  - use TLS: for security and to avoid proxy problems
  - for more information: see Ilya Grigorik's forthcoming O'Reilly chapter
  - Peter Lubber's WebSocket Cheat Sheet

# Signaling with Node and Socket.io

- Socket.io uses WebSocket with fallbacks
- Simple to exchange messages
- Built-in concept of 'rooms'

[Codelab](#)

[Live example](#)

# Signaling gotchas

- RTCPeerConnection won't start gathering candidates until `setLocalDescription()` is called: mandated in [JSEP IETF draft](#)
- Take advantage of [Trickle ICE](#): call `addIceCandidate()` as soon as candidates arrive
- Not bandwidth or CPU hungry, but could still be heavy load

# RTCPeerConnection initialisation

- Ascertain local media conditions: resolution, codec capabilities...
- Get potential network addresses for the application's host: candidates

# Make me an offer

1. Fred calls `createOffer()`
2. In the callback, Fred calls `setLocalDescription()`
3. Fred serialises the offer and sends it to Wilma
4. Wilma calls `setRemoteDescription()` with the offer
5. Wilma calls `createAnswer()`
6. Wilma calls `setLocalDescription()` with the answer and sends it to Fred
7. Fred receives answer and calls `setRemoteDescription()`

# Find me a candidate

1. Fred and Wilma create RTCPeerConnection objects
2. If success, `icecandidate` events start propagating
3. Fred starts serialises IceCandidates and sends them to Wilma
4. Wilma gets Fred's IceCandidates, calls `addIceCandidate()`
5. Wilma serialises IceCandidates and sends them to Fred
6. Fred gets Wilma's IceCandidates, calls `addIceCandidate()`
7. Ping!

# An RTCSessionDescription

```
v=0                                         UGH
o=- 7614219274584779017 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS
m=audio 1 RTP/SAVPF 111 103 104 0 8 107 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:1 IN IP4 0.0.0.0
a=ice-ufrag:W2TGCZw2NZHuwlNF
a=ice-pwd:xdQEccP40E+P0L5qTyzDgfmW
...
...
```

Want to know what all this SDP gobbledegook actually means?

Take a look at the [IETF examples](#).

# Candidate objects

```
{ NICE
  'type': 'candidate',
  'label': 1,
  'id': 'video',
  'candidate': 'a=candidate:1274936569 1 udp 1845501695 2.96.35.15 63579
                typ srflx raddr 192.168.0.3 rport 63579 generation 0\r\n'
}
...
{
  'type': 'candidate',
  'label': 0,
  'id': 'audio',
  'candidate': 'a=candidate:3802297132 1 udp 2113937151 192.168.0.3 63579
                typ host generation 0\r\n'
}
```

# RTCPeerConnection + signaling

[w3.org/TR/webrtc/#simple-peer-to-peer-example](https://w3.org/TR/webrtc/#simple-peer-to-peer-example)

[simpl.info/rtcpeerconnection/munge](http://simpl.info/rtcpeerconnection/munge)

[simpl.info/rtcpeerconnection/multi](http://simpl.info/rtcpeerconnection/multi)

[simpl.info/pc](http://simpl.info/pc)

---

[apprtc.appspot.com](http://apprtc.appspot.com)

---



# RTCDATAChannel

Bidirectional communication of arbitrary data between peers

# Communicate arbitrary data



```
onreceivemessage = handle(data);  
...  
var myData = [  
  {  
    id: "ship1";  
    x: 24,  
    y: 11,  
    velocity: 7  
  },  
  ...  
]  
send(myData);
```

→  
RTCDataChannel

+

RTCPeerConnection

←

```
onreceivemessage = handle(data);  
...  
var myData = [  
  {  
    id: "ship7";  
    x: 19,  
    y: 4,  
    velocity: 18  
  },  
  ...  
]  
send(myData);
```



# RTCDataChannel

- Same API as WebSockets
- Ultra-low latency
- Optionally unreliable or reliable:
  - Firefox and Chrome 31, Chrome 30 behind a flag
- Secure

# RTCDataChannel API

```
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);  
};
```

[simpl.info/dc](http://simpl.info/dc)

---

# Sharefest

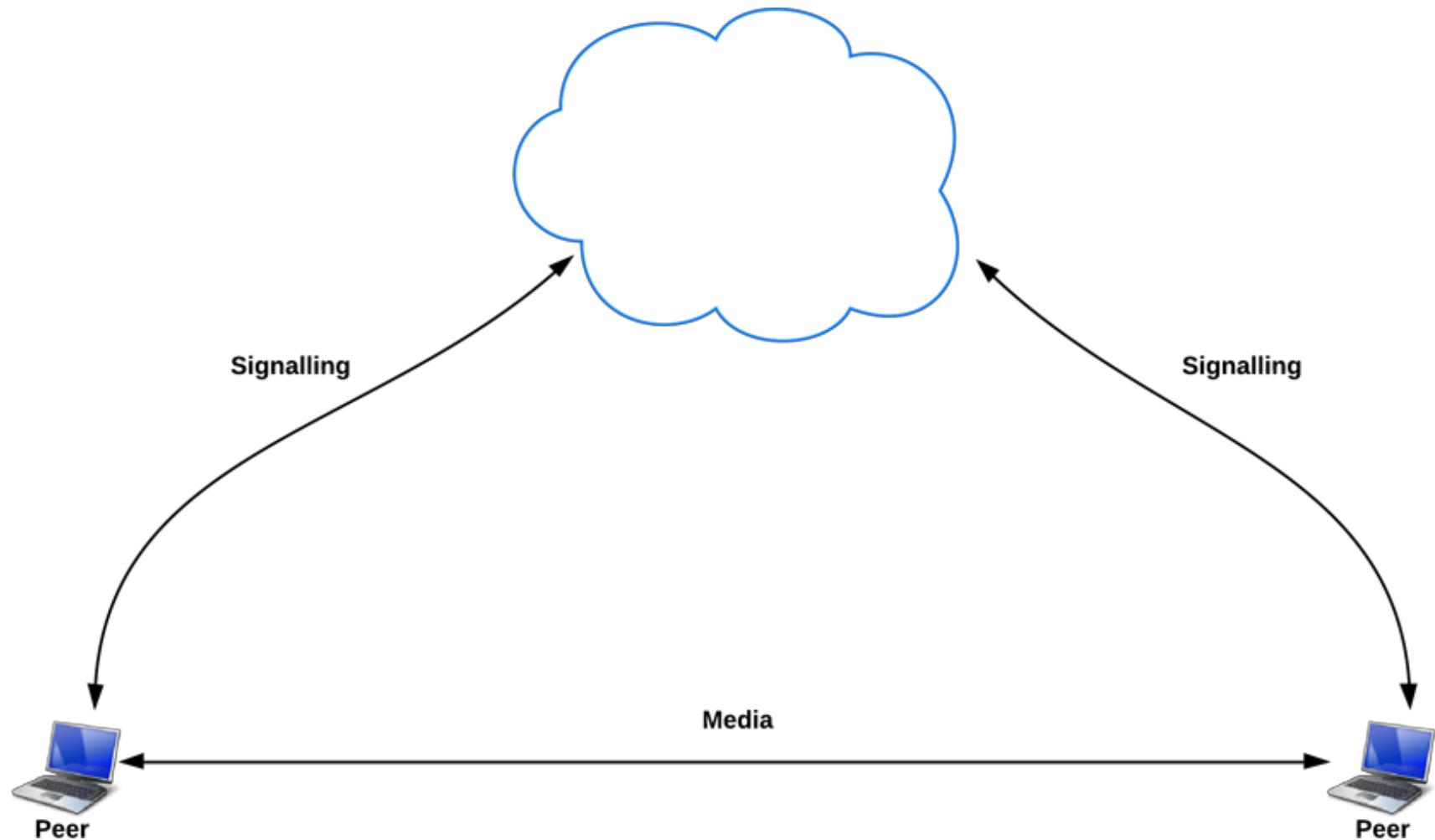
---



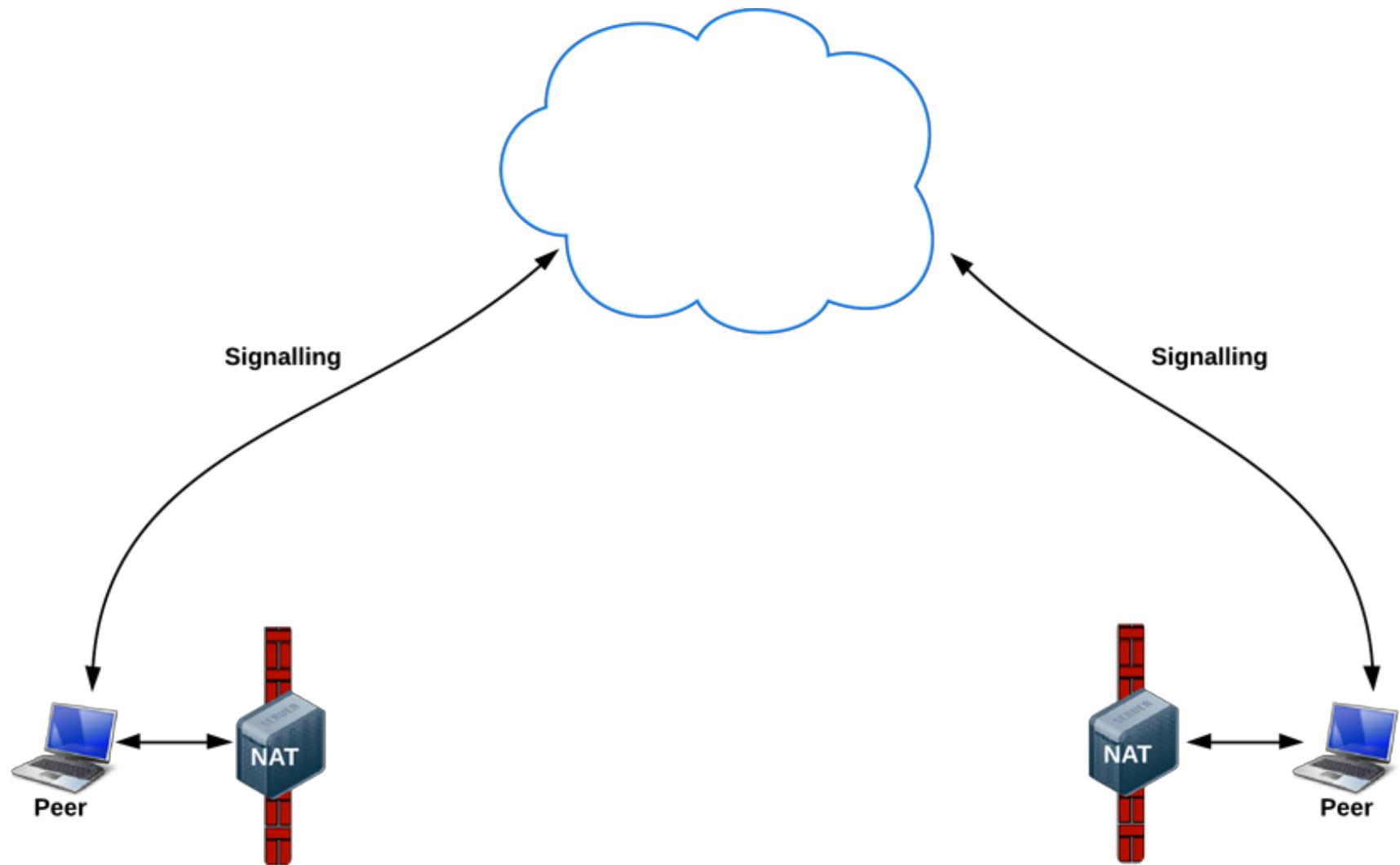
# STUN and TURN

P2P in the age of firewalls and NATs

# An ideal world



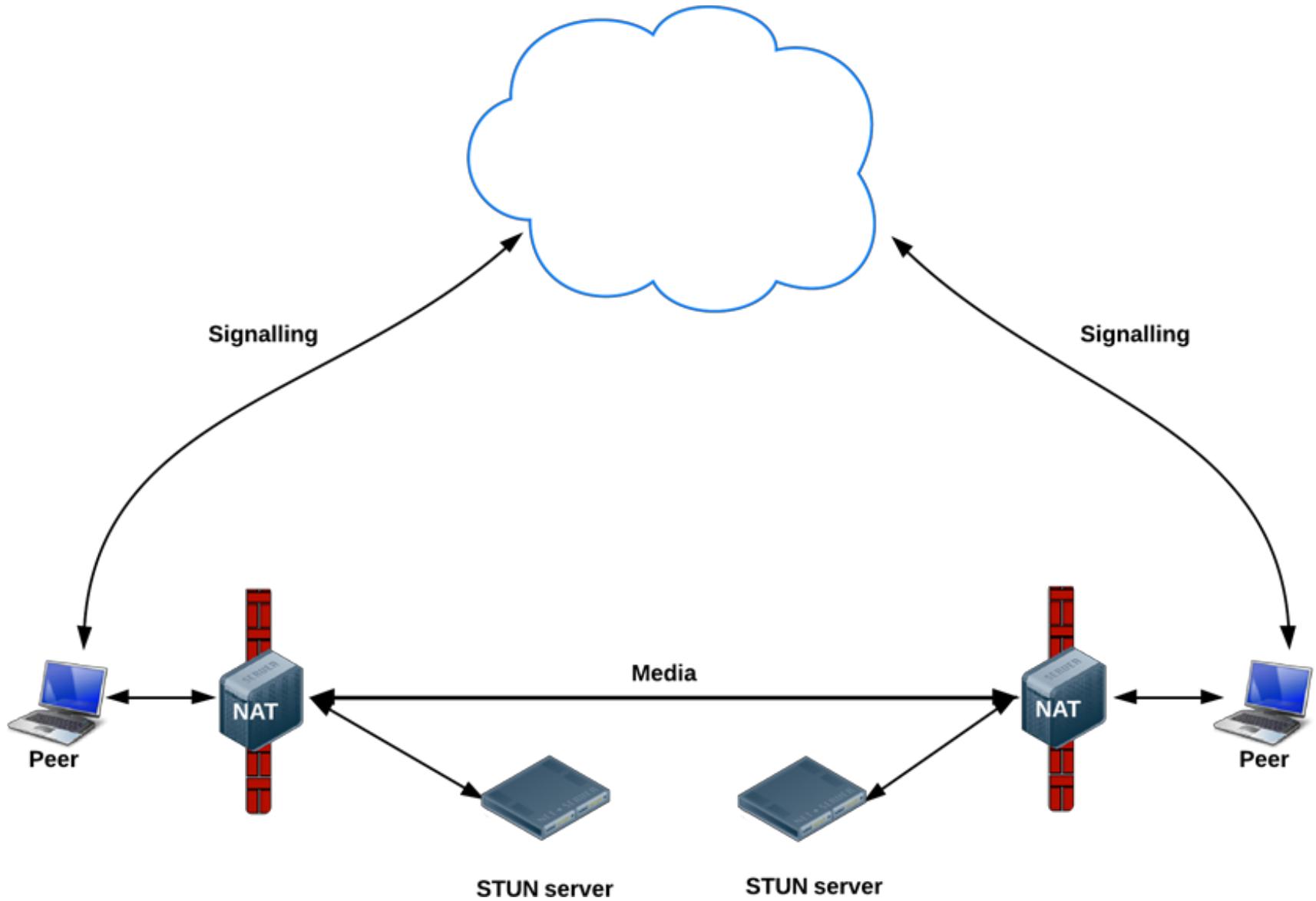
# The real world



# STUN

- Tell me what my public IP address is
- Simple server, cheap to run
- Data flows peer-to-peer

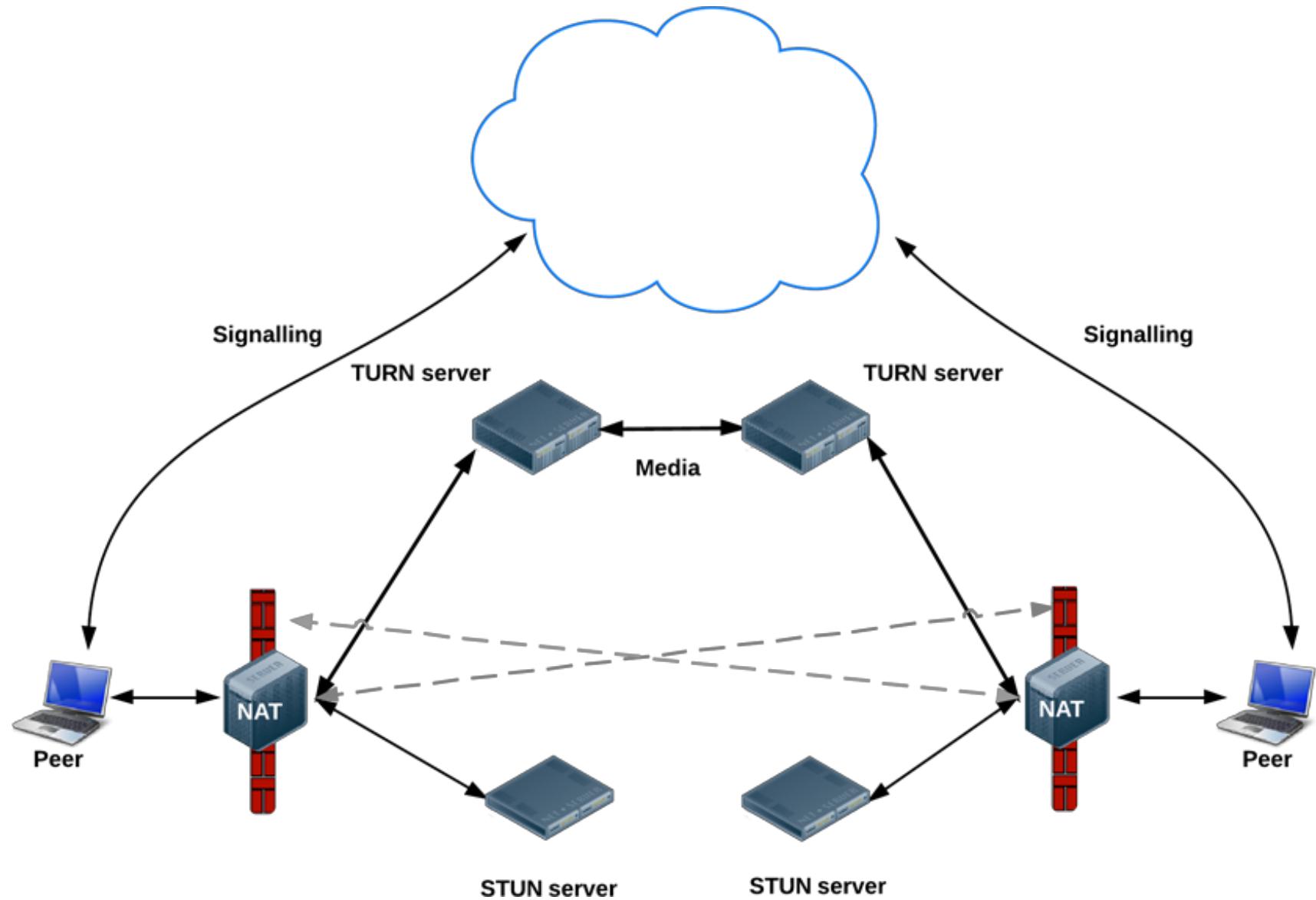
# STUN



# TURN

- Provide a cloud fallback if peer-to-peer communication fails
- Data is sent through server, uses server bandwidth
- Ensures the call works in almost all environments

# TURN



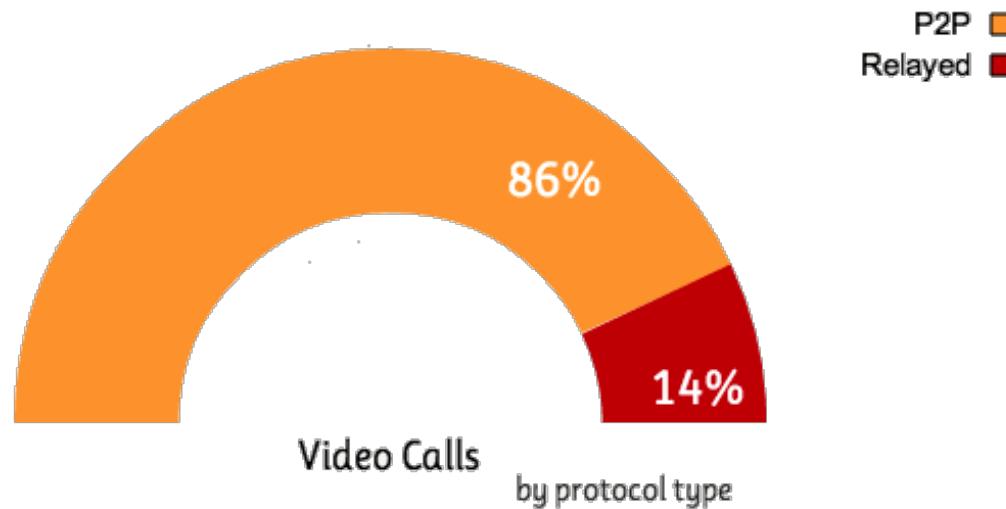
# Selecting STUN and TURN servers

```
{  
  "iceServers": [  
    {  
      "url": "stun:stun.l.google.com:19302"  
    },  
    {  
      "url": "turn:192.158.29.39:3478?transport=udp",  
      "credential": "JZEOEt2V3Qb0y27GRntt2u2PAYA=",  
      "username": "28224511:1379330808"  
    },  
    {  
      "url": "turn:192.158.29.39:3478?transport=tcp",  
      "credential": "JZEOEt2V3Qb0y27GRntt2u2PAYA=",  
      "username": "28224511:1379330808"  
    }  
  ]  
}
```

JAVASCRIPT

# ICE

- ICE: a framework for connecting peers
- Tries to find the best path for each call
- Vast majority of calls can use STUN ([webrtcstats.com](http://webrtcstats.com)):



# Deploying STUN/TURN

- stun.l.google.com:19302
- WebRTC stunserver, turnserver
- [rfc5766-turn-server](#)
- [VM image for Amazon Web Services](#)
- restund

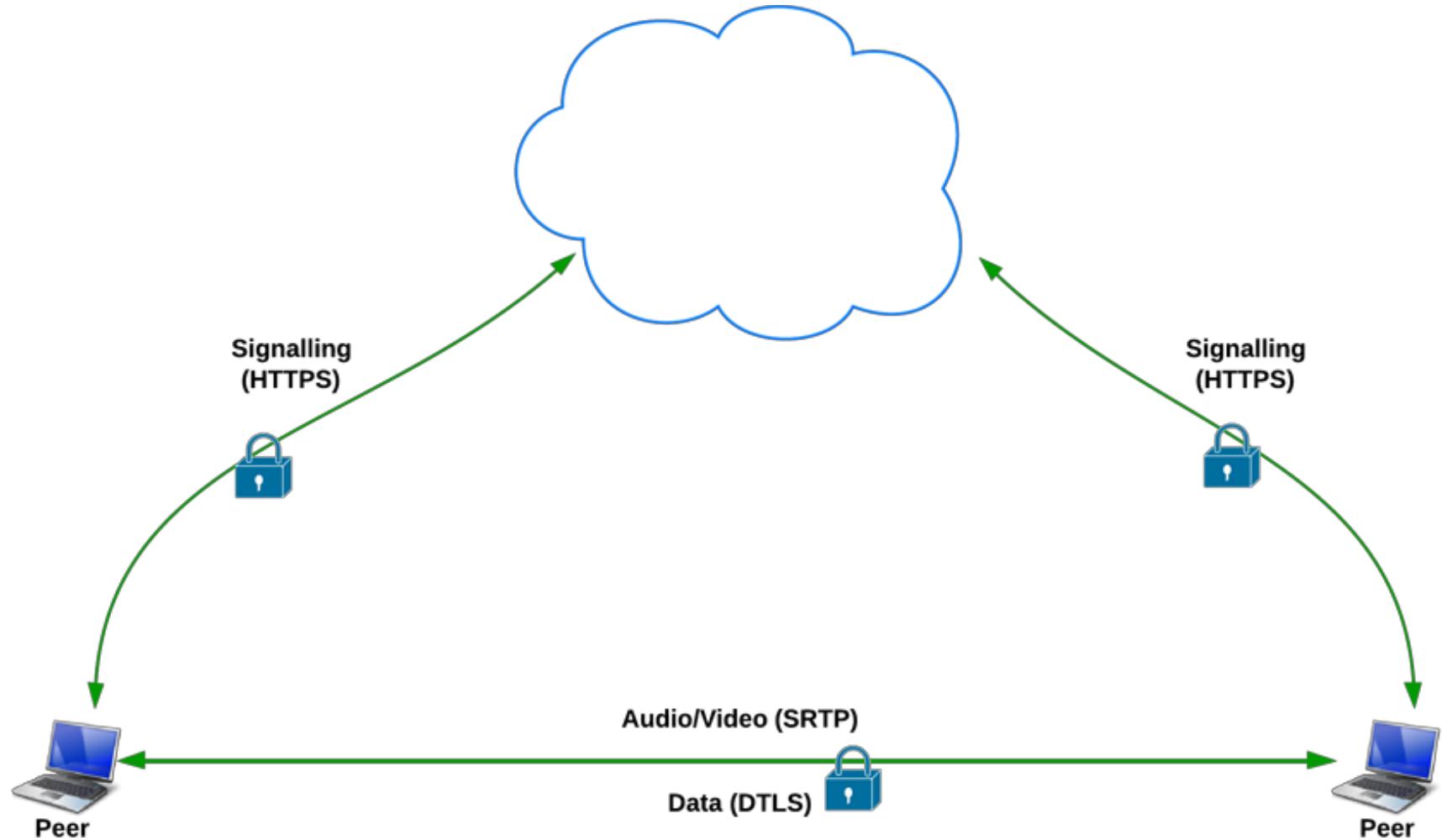


# Security

# Security throughout WebRTC

- Mandatory encryption for media and data
- Secure UI, explicit opt-in
- Sandboxed, no plugins
- [WebRTC Security Architecture](#)

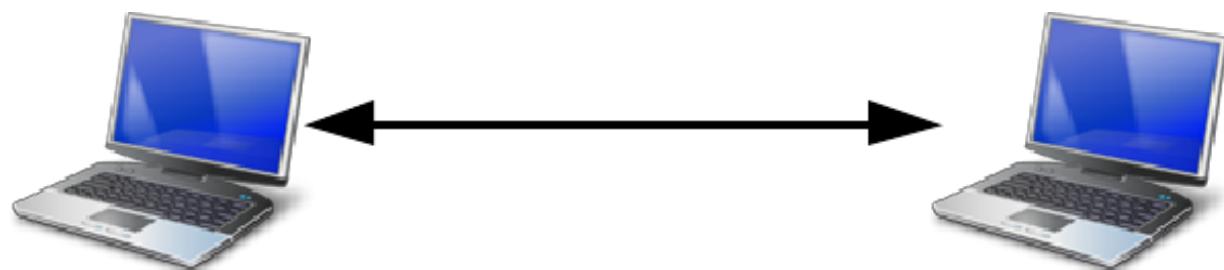
# Secure pathways



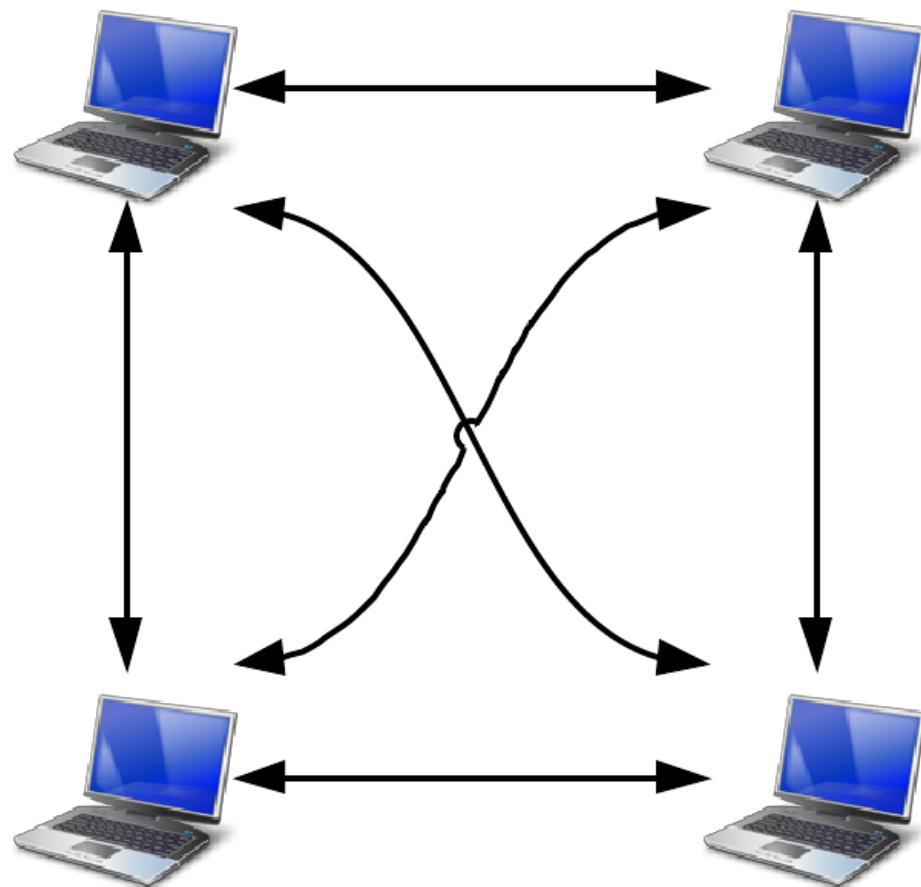


# Architectures

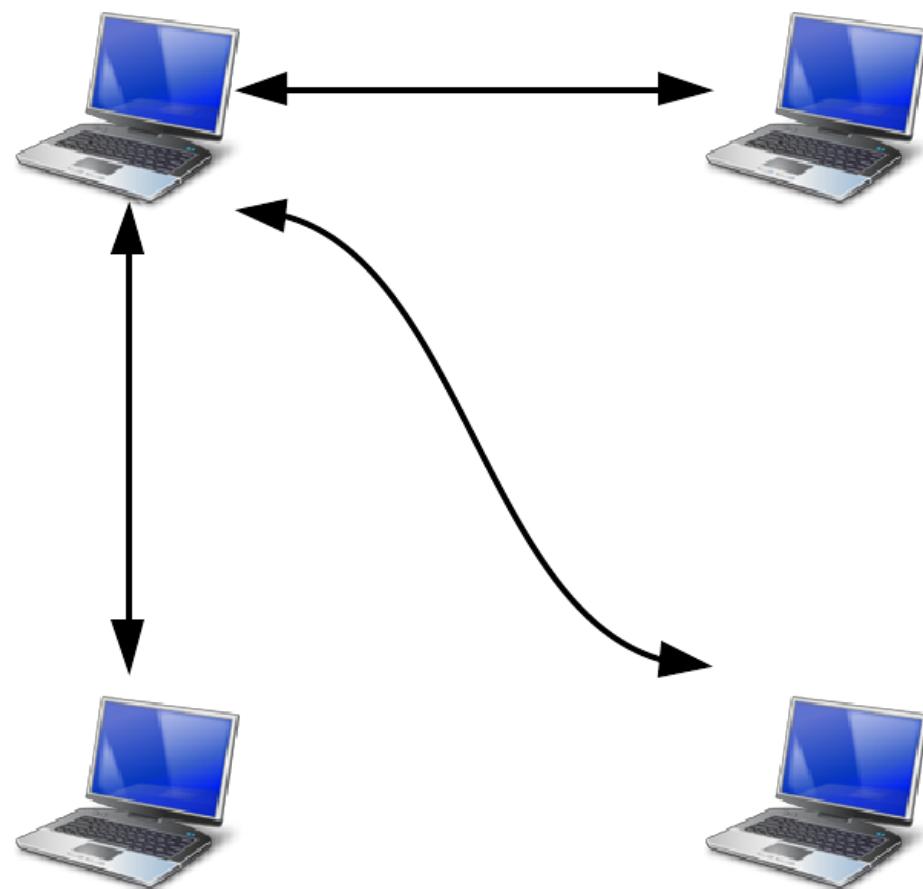
# Peer to peer: one-to-one call



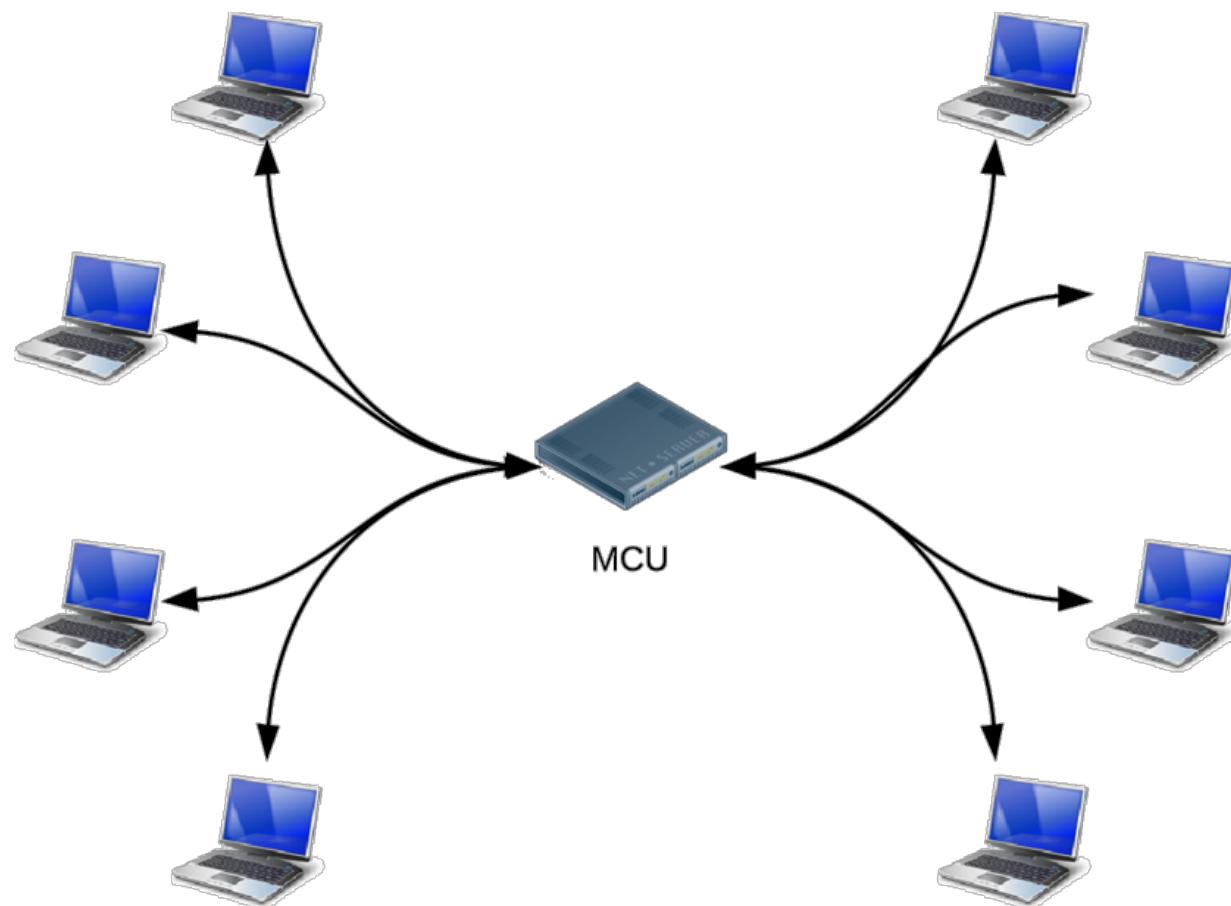
# Mesh: small N-way call



# Star: medium N-way call



# MCU: large N-way call





# Beyond browsers

# Phones and more

- Easy to interoperate with non-browser devices
  - [sipML5](#) open source JavaScript SIP client
  - [Phono](#) open source JavaScript phone API
  - [Zingaya](#) embeddable phone widget

# Telephony

Zingaya PSTN

---

# Tethr

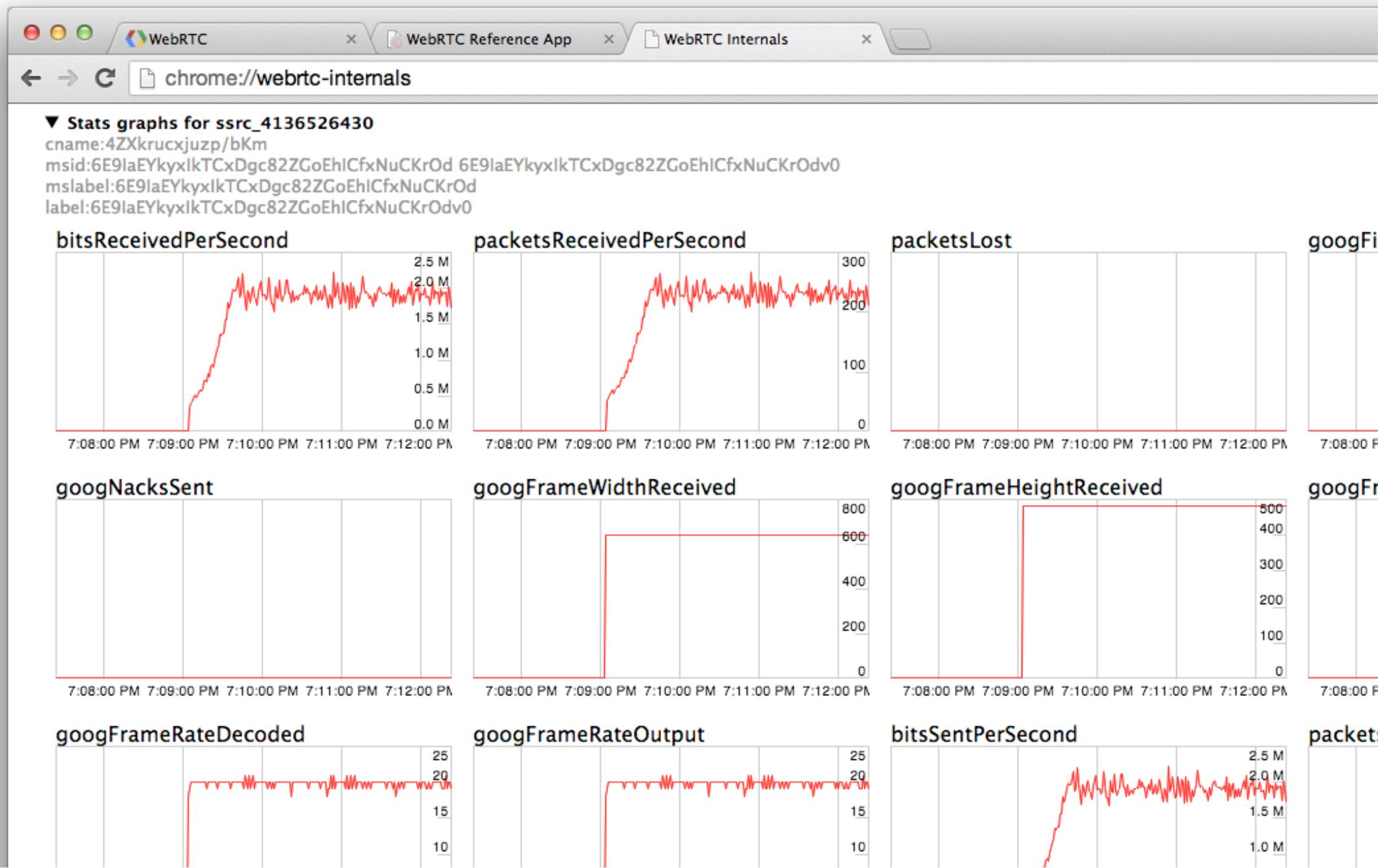


@sw12



# Building a WebRTC app

# chrome://webrtc-internals



# adapter.js

Lets you use the same code in all browsers:

- Removes vendor prefixes
- Abstracts Chrome/Firefox differences
- Minimizes effects of spec churn

**This is doing my head in.**

# JavaScript frameworks

- Video chat:
  - [SimpleWebRTC](#)
  - [easyRTC](#)
  - [webRTC.io](#)
- Peer-to-peer data:
  - [PeerJS](#)
  - [Sharefest](#)

# SimpleWebRTC

Easy peer-to-peer video and audio

```
var webrtc = new WebRTC({
  localVideoEl: 'localVideo',
  remoteVideosEl: 'remoteVideos',
  autoRequestMedia: true
});

webrtc.on('readyToCall', function () {
  webrtc.joinRoom('My room name');
});
```

JAVASCRIPT

# PeerJS

Easy peer-to-peer data

```
var peer = new Peer('someid', {key: 'apikey'});
peer.on('connection', function(conn) {
  conn.on('data', function(data){
    // Will print 'hi!'
    console.log(data);
  });
});

// Connecting peer
var peer = new Peer('anotherid', {key: 'apikey'});
var conn = peer.connect('someid');
conn.on('open', function(){
  conn.send('hi!');
});
```

JAVASCRIPT

# Complete services

- OpenTok (acquired by Telefonica Digital)
- vLine



# Don't forget the C++!

[webrtc.org/webrtc-native-code-package](https://webrtc.org/webrtc-native-code-package)

# More Information

- WebRTC and Web Audio resources list: [bit.ly/webrtcwebaudio](http://bit.ly/webrtcwebaudio)
- [Google I/O 2013 WebRTC presentation](#)
- Justin Uberti: [Google I/O 2012 presentation video](#)
- Cullen Jennings video: [HTML5 WebRTC](#)
- HTML5 Rocks:
  - [Capturing audio and video in HTML5](#)
  - [Getting Started With WebRTC](#)
  - [Updates](#)
- ...and a book: [webrtcbook.com](http://webrtcbook.com)

# Contact Us

- [webrtc.org](http://webrtc.org)
- [discuss-webrtc](http://discuss-webrtc.com)
- [+webrtc](http://plus.google.com/+webrtc)
- [@webrtc](http://twitter.com/webrtc)
- [crbug.com/new](http://crbug.com/new)

“WebRTC and HTML5 could enable the same transformation for real-time communications that the original browser did for information.”

Phil Edholm  
— NoJitter



[talky.io/gowebbrtc](https://talky.io/gowebbrtc)

---

[gowebrtc.appspot.com](https://gowebrtc.appspot.com)

---



# <Thank You!>

g+ [plus.sambutton.com](https://plus.google.com/u/0/+SamDutton)

twitter @sw12

www [www.sambutton.com](http://www.sambutton.com)

github [github.com/sambutton](https://github.com/sambutton)

