



OST

Eastern Switzerland
University of Applied Sciences

Distributed Systems Advanced & Blockchain

WebRTC

Thomas Bocek

27 September 2020

WebRTC – Introduction

- [WebRTC](#) for browser to browser communication
 - P2P, no server involved (~mostly)
 - Real time communication (RTC) via API
 - Main goal: eliminate plugins or native apps
 - Supported by Google, Microsoft, Mozilla, Opera, Apple
- Google bought in 2010 Global IP Solutions (GIPS) and open sourced WebRTC in 2011
- Protocol standardized by IETF (codec requirements, media protocol), JavaScript API by [W3C](#)
- Supported initially by Chrome, Firefox (now many others)
 - First cross-browser call in February 2013

- [Compatibility](#), 93%
 - Microsoft Edge 15~
 - Google Chrome 56+
 - Mozilla Firefox 44+
 - Safari 11+
 - Opera 43+
- [PeerJS](#)
 - Not all browser work 100% compatible with each other → PeerJS
 - «PeerJS provides a complete, configurable, and easy-to-use peer-to-peer API built on top of WebRTC»
- [SimpleWebRTC](#) – archived
 - Use webrtc directly?



WebRTC

WebRTC – Introduction

- Filling gap in the Web-Experience
 - Video Chat → Google Hangouts Plugin, Flash, Java
 - Multimedia / Conferences → Expensive, proprietary 3rd party apps
 - Customer Service → Chat only, 3rd party plugins/apps
 - Online Games → Flash
 - Real-time Feeds → Proprietary software
 - File Sharing → Requires Server / BitTorrent
- WebRTC widely deployed, no client necessary!
- Used in WhatsApp, Facebook Messenger, whereby.com
- Standard still not finalized
 - [W3C Candidate Recommendation 03 September 2020](#) ([W3C process flow](#))
 - «The RTCPeerConnection API has endured three design iterations on this topic over the years. As a result, each browser today implements a snapshot from a different point in the timeline of an evolving spec.» ([source](#)) ([other updates](#)) ([large messages](#), [fixed](#))



WebRTC – Concerns

- Outdated documentation ([source](#)):

```
var dc = pc.createDataChannel("namedChannel", {reliable: false});
```

- «reliable» is [obsolete](#)
- HTML browsers get [bloated](#)
 - Several GB RAM to open couple of tabs?
 - Hint: [adblocker](#)
- WebRTC API could be simplified
- Security Concerns
 - [Private IP](#) / IP behind VPN, Tor? (<https://dsl.hsr.ch/lect/webrtc>)
- But: WebRTC forbids unencrypted communication
 - [DTLS, SRTP](#)
 - [Complexity](#) - SCTP over DTLS over UDP

Demo for: <https://github.com/diafygi/webrtc-ips>

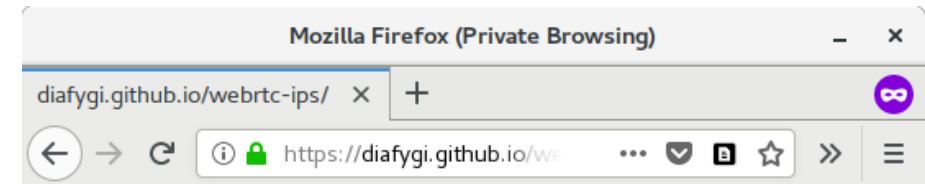
This demo secretly makes requests to STUN servers that can log your request. These requests are blocked by browser plugins (AdBlock, Ghostery, etc.).

Your local IP addresses:

- 10.8.0.18

Your public IP addresses:

Your IPv6 addresses:



Demo for: <https://github.com/diafygi/webrtc-ips>

This demo secretly makes requests to STUN servers that can log your request. These requests do not show up in developer consoles and cannot be blocked by browser plugins (AdBlock, Ghostery, etc.).

Your local IP addresses:

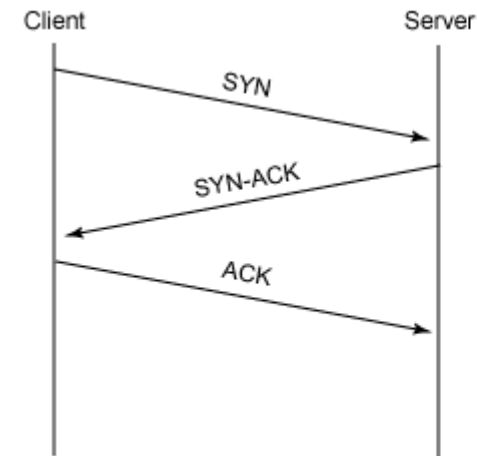
- 192.168.1.139

Your public IP addresses:

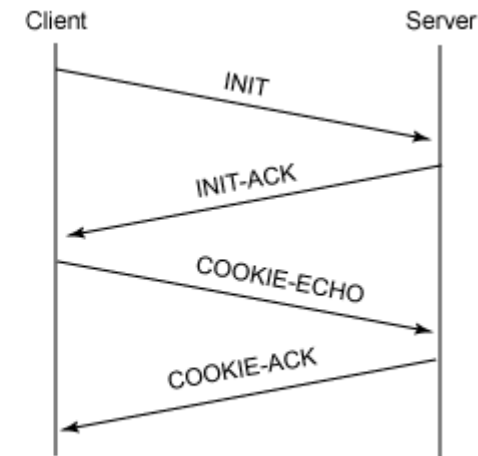
Your IPv6 addresses:

SCTP (Stream Control Transmission Protocol)

- SCTP Message-based (reliable) [[rfc](#)]
- Allows data to be divided into multiple streams
- Syn cookies - SCTP uses a four-way handshake with a signed cookie
- 32-bit end-to-end checksum (CRC32C)
- Multi-homing multiple IP addresses of endpoints
- Not widely used: “We have been deploying SCTP in several applications now, and encountered significant problem with SCTP support in various home routers.”
 - E.g., OpenWRT – not enabled by default
 - E.g., UFW - Uncomplicated Firewall – not supported
- SCTP used by WebRTC



TCP 3-way handshake

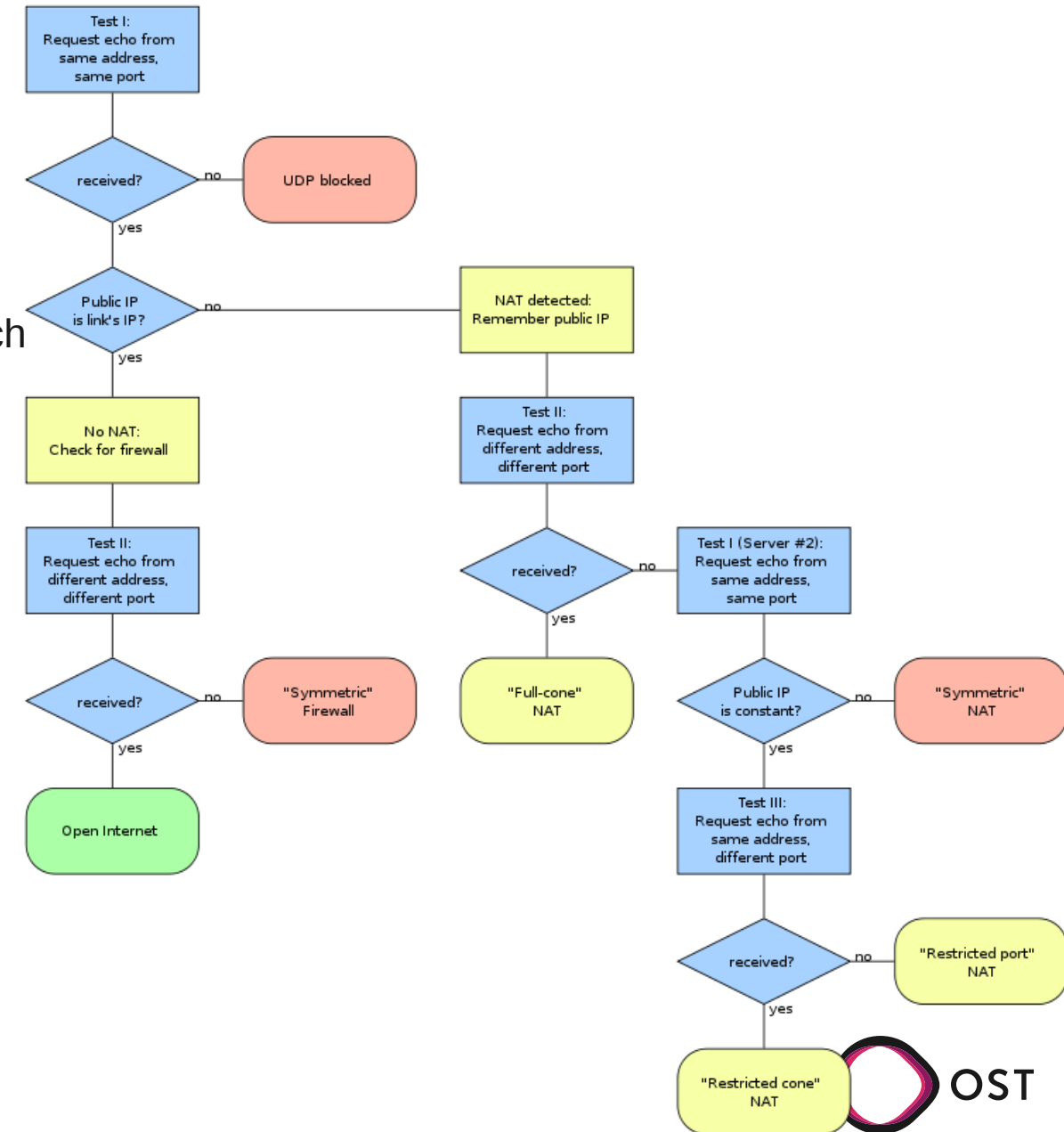
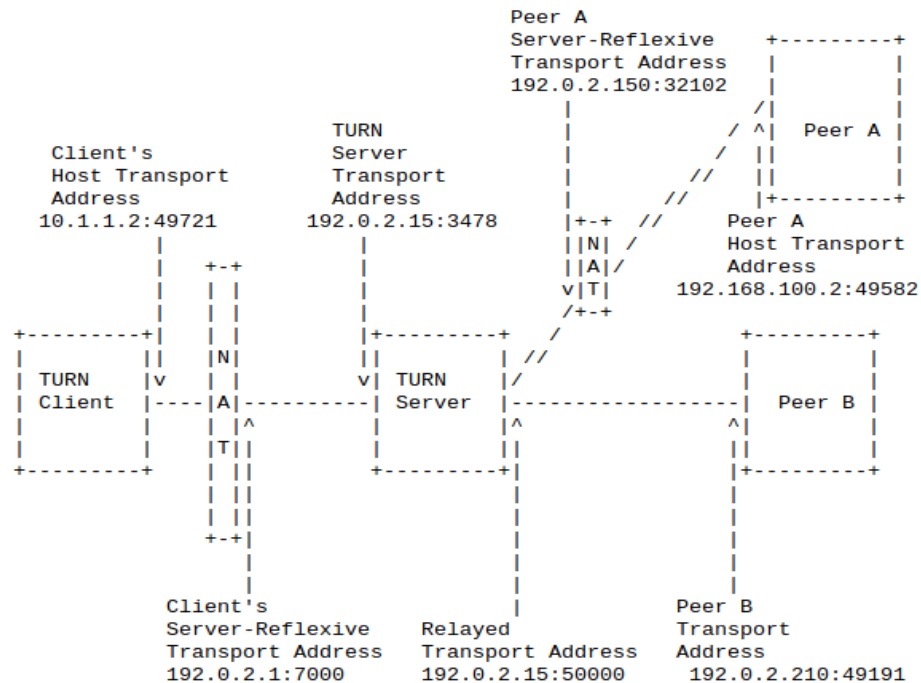


SCTP 4-way handshake

<https://www.ibm.com/developerworks/library/l-sctp/>

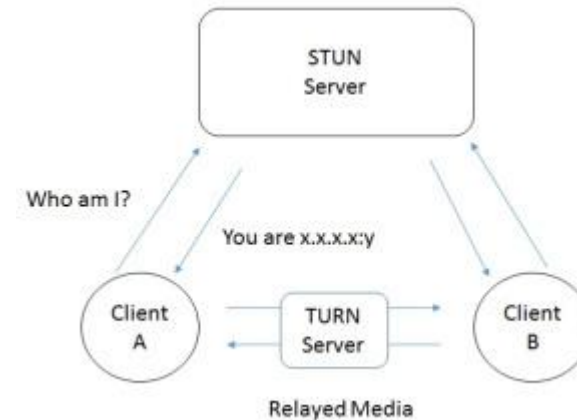
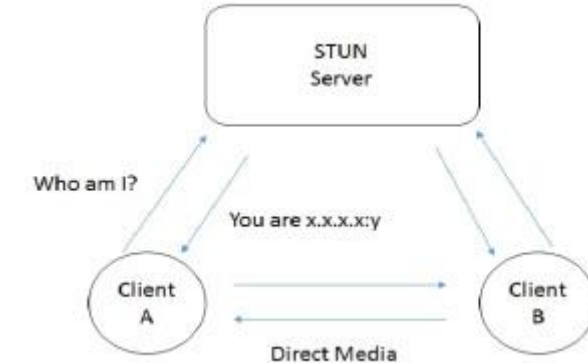
WebRTC – Introduction

- On the bright side: developer does not need to care about NAT
 - Abstraction using STUN, ICE, TURN
 - STUN: session traversal utilities for NAT (detect which kind of NAT, [rfc5389](https://tools.ietf.org/html/rfc5389))
 - “STUN is not a NAT traversal solution by itself”
 - TURN**: traversal using relays around NAT



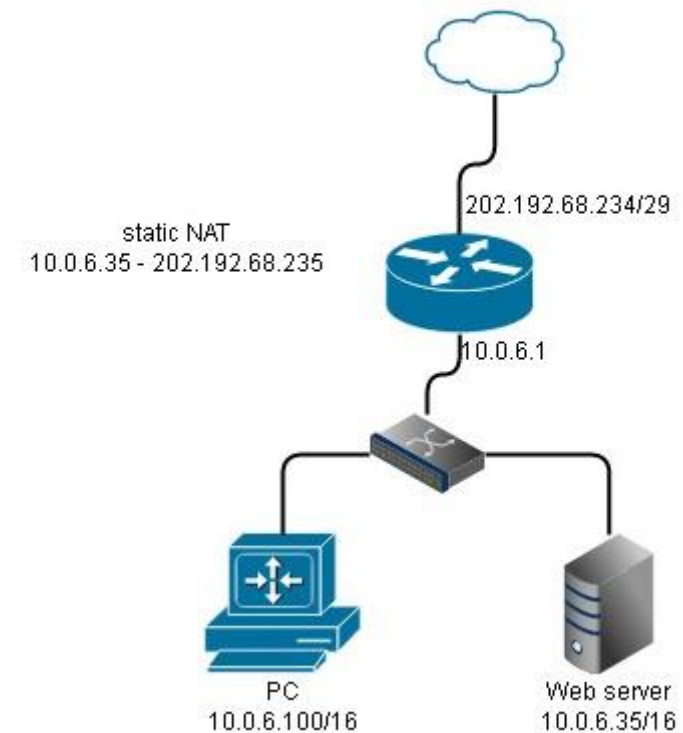
WebRTC – Introduction

- TURN
 - TURN always UDP server and the peer
 - TURN client/server UDP, TCP/TLS
 - Some firewalls block UDP entirely
 - UPnP / NAT-PMP setup by the browser optional?
 - Bugzilla@Mozilla – [Bug 860045](#)
 - TURN is a relay extension for STUN
- ICE - Interactive Connectivity Establishment
 - [RFC 8445](#) a protocol for NAT traversal
 - “ICE works by exchanging a multiplicity of IP addresses and ports, which are then tested for connectivity by peer-to-peer connectivity checks.”



WebRTC – Connectivity

- NAT loopback: 2 devices behind same NAT
 - (peer-reflexive candidate)
- Encryption is mandatory for all WebRTC components
 - SRTP for Media, DTLS for Data, HTTPS for signaling (optional)
- WebRTC is not a plugin
 - Camera and microphone access must be granted explicitly
- Troubleshoot: test.webrtc.org
- Once connection is established – easy API
 - `RTCPeerConnection.send("hallo")`
 - `RTCPeerConnection.onmessage = function ...`



Connectivity, Security, and Robustness

- NAT
 - Network Address Translation – breaks end-to-end
 - “If nothing else, [NAT] can serve to provide temporarily relief while other, more complex and far-reaching solutions are worked out”
(RFC 1631 - The IP Network Address Translator (NAT))
- How to deal with NAT: “Easy solution”:
 - Manual port forwarding: e.g., setup on your router

The screenshot shows the OpenWrt web interface. At the top is a navigation bar with 'OpenWrt' and links for 'Status', 'System', 'Services', 'Network', and 'Logout'. A blue badge on the right indicates 'UNSAVED CHANGES: 10'. Below the navigation bar are four tabs: 'General Settings', 'Port Forwards' (which is active), 'Traffic Rules', and 'Custom Rules'. The main heading is 'Firewall - Port Forwards', followed by a descriptive text: 'Port forwarding allows remote computers on the Internet to connect to a specific computer or service within the private LAN.' Below this is a section titled 'Port Forwards' containing a table with the following columns: 'Name', 'Match', 'Forward to', 'Enable', and 'Sort'.

Name	Match	Forward to	Enable	Sort
TomP2P	IPv4-TCP, UDP	IP 192.168.1.200, port 4000 in lan	☒	↑ ↓ Edit Delete

Connectivity, Security, and Robustness

- Easy solution: UPNP / NAT-PMP
 - Both configure port forwarding, but UPNP is more: discover devices - uses broadcasting to find router (Simple Service Discovery Protocol)
 - UPNP: configure devices - uses HTTP and XML to configure port forwarding (Internet Gateway Device Protocol)
 - NAT-PMP: protocol made for configuring port-forwarding, but no discover (how to find router?)

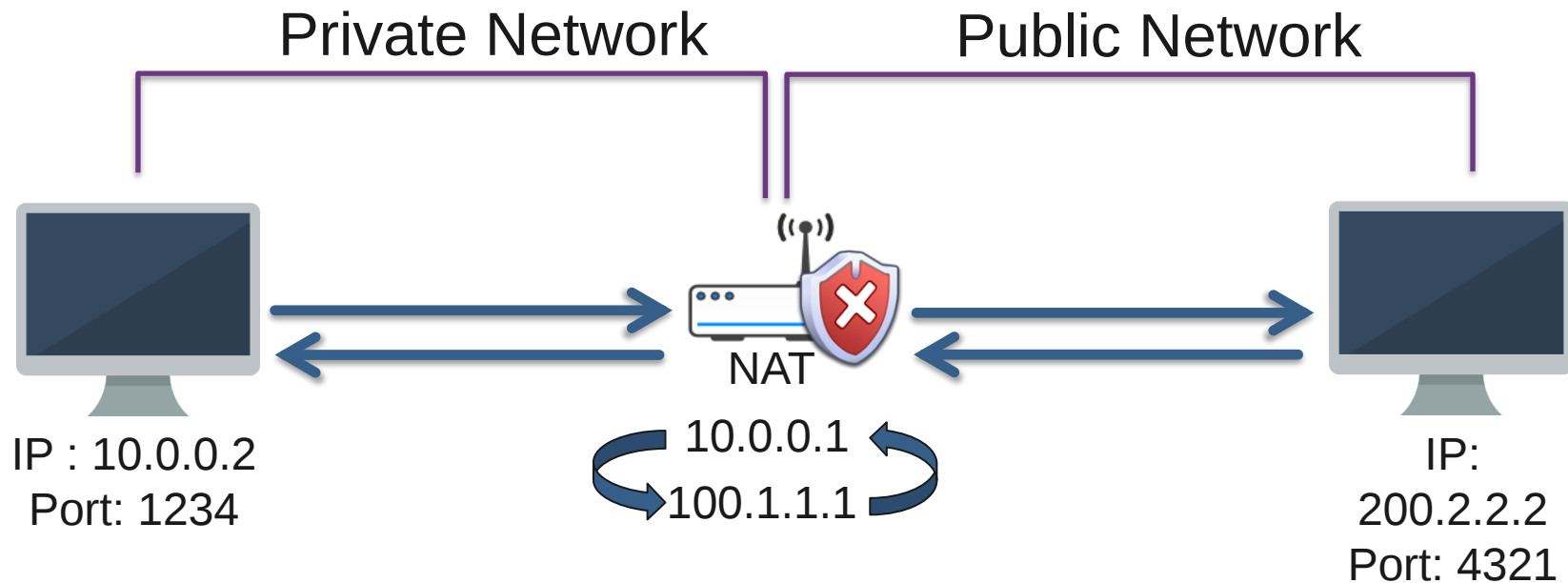
Active UPnP Redirects

Protocol	External Port	Client Address	Client Port	
UDP	60011	192.168.1.200	60011	 Delete Redirect
TCP	60011	192.168.1.200	60011	 Delete Redirect

Powered by [LuCI Trunk \(0.12+svn-r10530\)](#) OpenWrt Barrier Breaker 14.07

Connectivity, Security, and Robustness

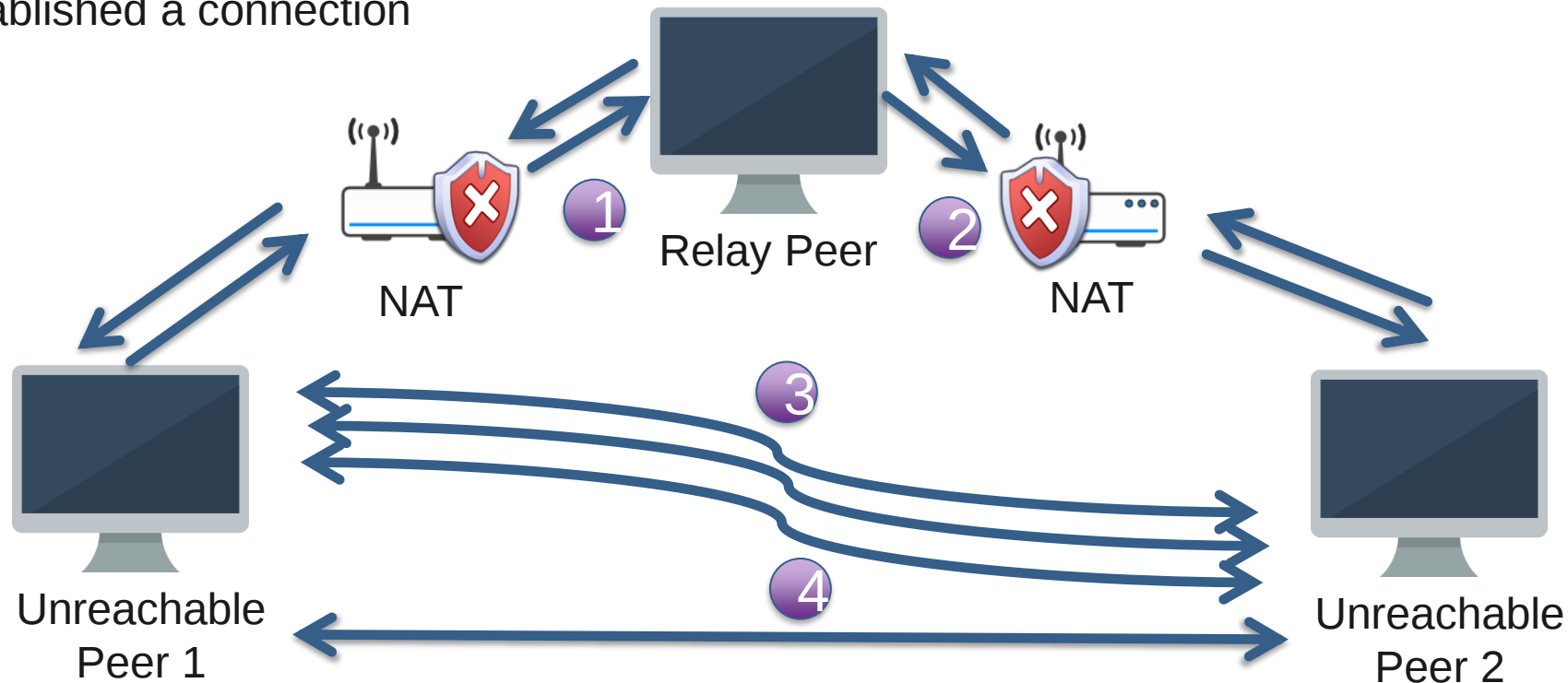
- Difficult solution: hole punching
 - rendezvous / relay peer which does “hole punching”, in worst case relay traffic.
- NAT: translation table for private / public network



NAT Table Entry: (10.0.0.2:1234, 200.2.2.2:4321; 200.2.2.2:4321, 100.1.1.1:1234)

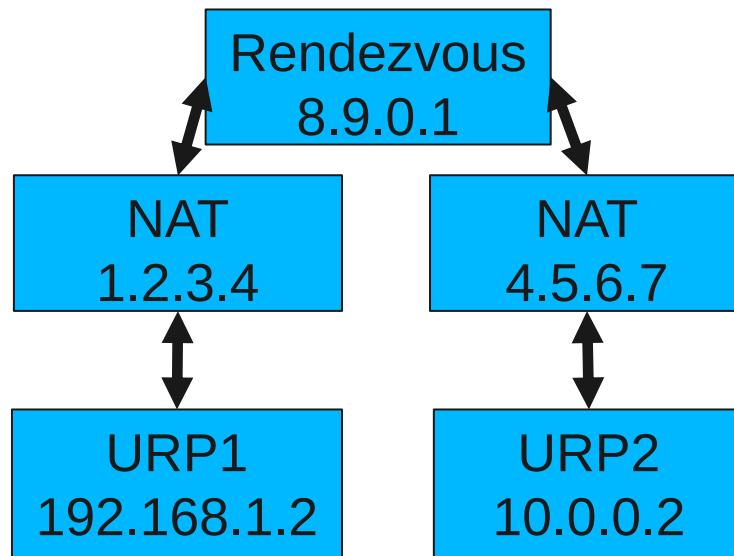
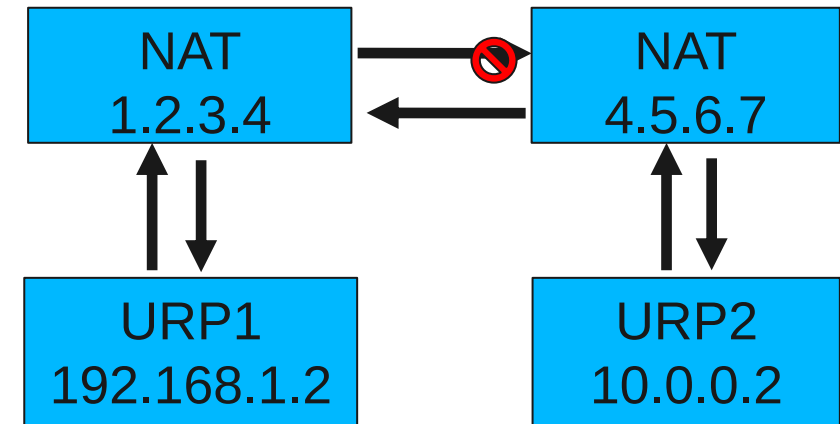
Connectivity: Hole punching in P2P Networks

- 1) Peer 1 initiates a new connection trial to peer 2 via relay and signals its source ports and IP (relay/rendez-vous peer has connection to Unreachable Peer 2)
- 2) Peer 2 answers back with its source ports and IP
- 3) Both of the peers punch holes into their firewall/NAT
- 4) Established a connection



Connectivity, Security, and Robustness

- Hole punching
 - URP1 got 4.5.6.7:5000, URP2 got 1.2.3.4:4000
 - Unreachable peer 1 request to NAT 4.5.6.7, will fail – no mapping, however, unreachable peer 1 creates mapping with that request
 - Unreachable peer 2 sends request to unreachable peer 1 (1.2.3.4:4000) success!



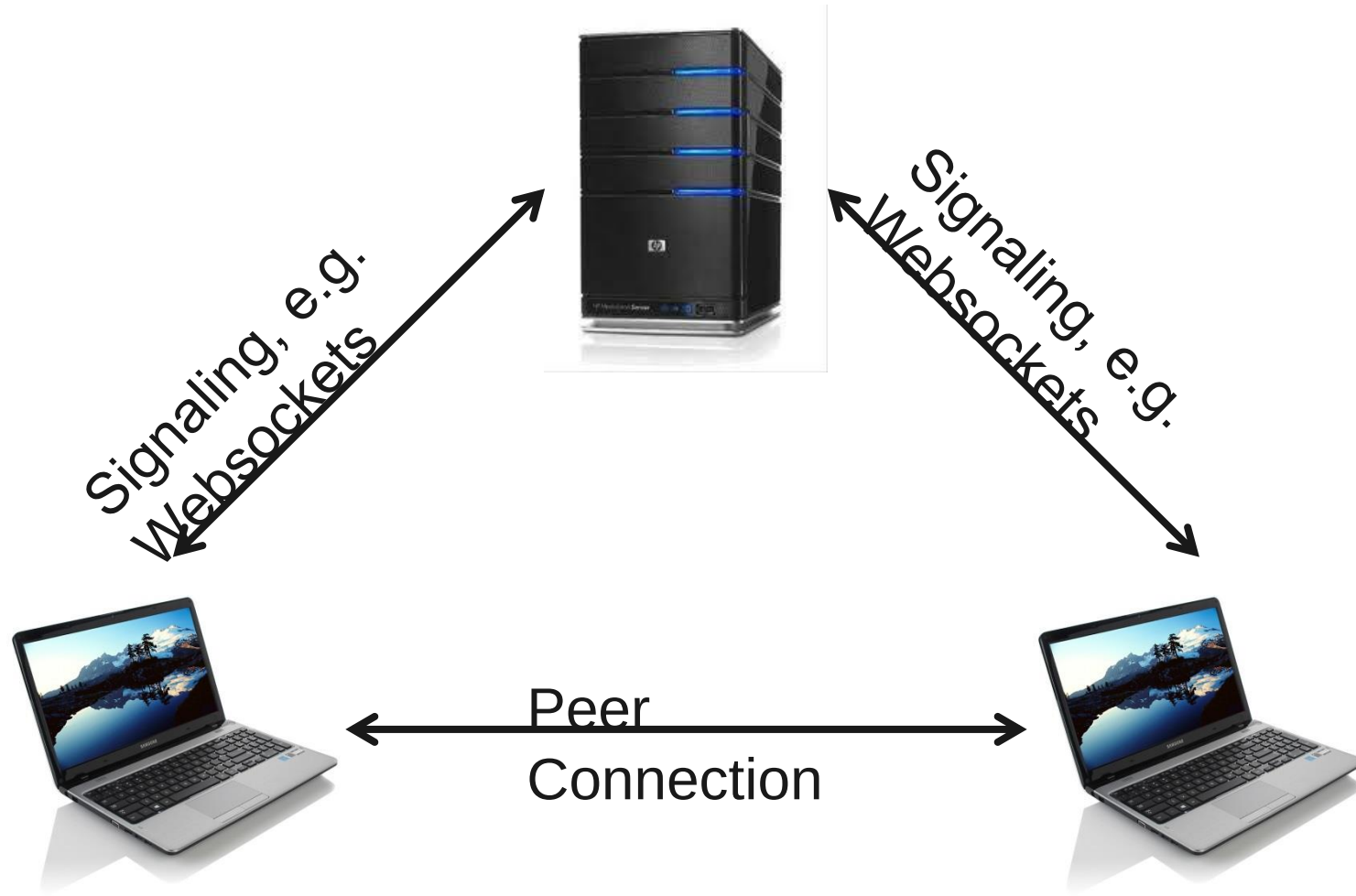
Mapping for NAT 1.2.3.4 (Unreachable peer 1)

192.168.1.2:4000	4.5.6.7:5000	4.5.6.7:5000	1.2.3.4:4000
------------------	--------------	--------------	--------------

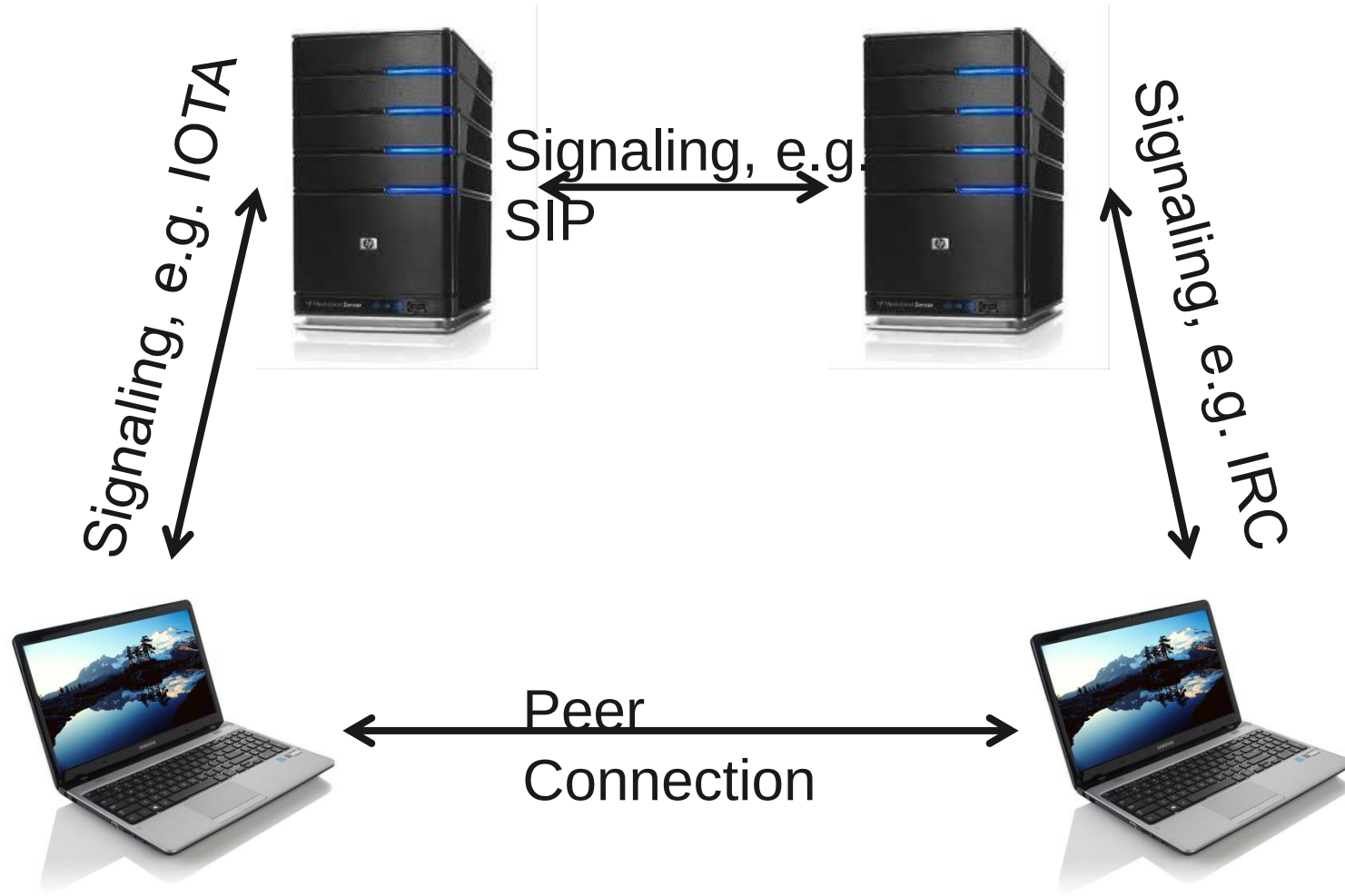
Mapping for NAT 4.5.6.7 (Unreachable peer 2)

10.0.0.2:5000	1.2.3.4:4000	1.2.3.4:4000	4.5.6.7:5000
---------------	--------------	--------------	--------------

WebRTC Architecture - Triangle

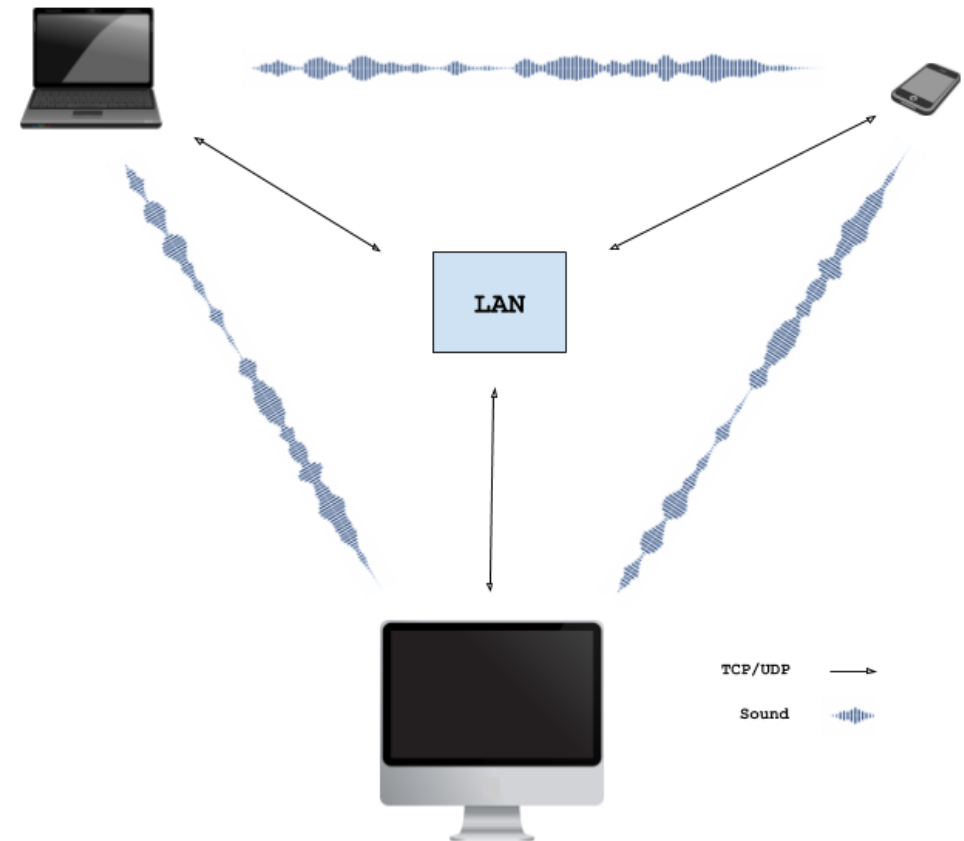


WebRTC Architecture - Trapezoid



WebRTC Signaling

- A proof-of-concept for WebRTC signaling using sound. Works with all devices that have microphone + speakers. Runs in the browser:
 - <https://github.com/ggerganov/wave-share>



WebRTC - Demo

- Server - server.js
 - Node.js server
 - Serves files (index.html / [express](#))
 - Listens to incoming WS messages and broadcast messages to all connected clients
 - 19 loc
- Run
 - yarn install
 - node server.js
- Minimal example!

```
const express = require('express');
const WebSocket = require('ws');

let app = express();
// setup static files
app.use(express.static('.'));
// setup listening
let server = app.listen(4000, function () {
  console.log("listening on " + server.address().address + ":" +
server.address().port);
});

//WebSocket broadcast setup
const wss = new WebSocket.Server({ server });
wss.on('connection', function(ws) {
  ws.on('message', function(message) {
    // Broadcast any received message to all clients
    console.log('received: %s', message);
    wss.clients.forEach(function(client) {
      if(client.readyState === WebSocket.OPEN) {
        client.send(message);
      }
    });
  });
});

console.log('Server running.');
```

Example: <https://github.com/tbocek/ADSB-webrtc>



WebRTC - Demo

- Client – index.html
 - 2 buttons, 2 text fields
- Client JavaScript
 - peerConnection, here we could add STUN/TURN
 - createDataChannel, create named channel. This needs to be done before offer/answer
 - onicecandidate, once offer was sent ICE candidates are sent

```
<body>

<button id="connectButton">Connect</button>
<label for="message_send">Enter a message:
    <input type="text" id="message_send" placeholder="Message
send" size=30 maxlength=30>
</label>
<button id="sendButton">Send</button>
<label for="message_rcv">Messages received:
    <input type="text" id="message_rcv" placeholder="Message
received" size=30 maxlength=30 disabled>
</label>
```

```
</body>
```

```
function startup() {
document.getElementById('connectButton').addEventListener('click',
connectPeers, false);
document.getElementById('sendButton').addEventListener('click',
sendMessage, false);
peerConnection = new RTCPeerConnection();
peerConnection.onicecandidate = gotIceCandidate;
dataChannel = peerConnection.createDataChannel("test");
}
```

WebRTC - Demo

- Client 1 JavaScript
 - setLocalDescription()
 - serverConnection.send()
- Server broadcasts local description to all
- Client 2
 - If SDP, set remote description
 - If offer, send answer

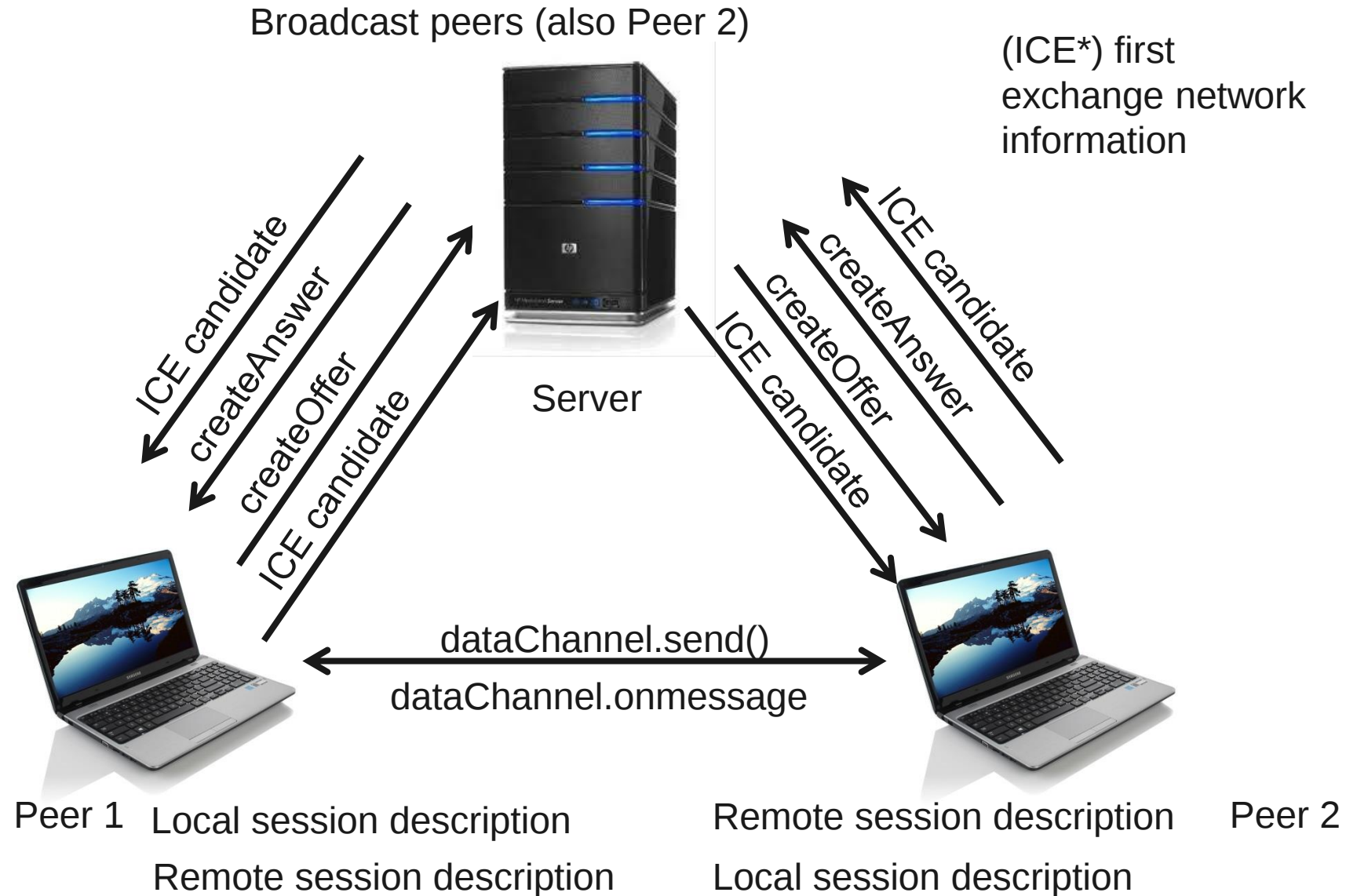
Example: <https://github.com/tbocek/ADSB-webrtc>

```
function connectPeers() {  
  peerConnection.createOffer().then(createdDescription).catch(errorHandler)  
}  
  
function createdDescription(description) {  
  console.log('Got description');  
  peerConnection.ondatachannel = gotDataChannel;  
  peerConnection.setLocalDescription(description).then(function() {  
    serverConnection.send(JSON.stringify({'sdp':peerConnection.localDescription,  
    'uid': uid}));}).catch(errorHandler);  
}
```

```
function gotMessageFromServer(message) {  
  const signal = JSON.parse(message.data);  
  if(signal.uid == uid) return; // Ignore messages from ourself  
  console.log('Got message from signaling server: '+message.data);  
  if(signal.sdp) {  
    peerConnection.setRemoteDescription(new RTCSessionDescription(signal.sdp))  
      .then(function() {  
        // Only create answers in response to offers  
        if(signal.sdp.type == 'offer') {  
          peerConnection.createAnswer().then(createdDescription)  
            .catch(errorHandler);  
        }  
      }).catch(errorHandler);  
  } else if(signal.ice) {  
    peerConnection.addIceCandidate(new RTCIceCandidate(signal.ice))  
      .catch(errorHandler);  
  }  
}
```



WebRTC Architecture - Demo



WebRTC – Outlook



- Strong focus on VoIP [[demo](#)]
 - Skype competitor? MS Teams?
 - Microsoft / IE / Edge and WebRTC?
 - SDP / signaling overhead if using with raw P2P data
- Fewer plugins (flash, java), fewer registrations
- Mandatory Codecs: [VP8, H264](#)
 - [Video codec in JavaScript](#)
- Web-based P2P frameworks
 - <http://peerjs.com> - make the API simpler, is it complex?
- New types of applications – Conferencing, gaming, P2P file sharing
 - [PeerCDN](#), serve static content from browsers of other visitors (broken)
- [ORTC](#), WebRTC 1.1?
 - Object Real-Time Communications (ORTC) instead Session Description Protocol (SDP)