HS18

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

WebRTC

T. Bocek

thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the *News*

- **21.09.2018: Opening corporate accounts for blockchain companies – Swiss Bankers Association publishes guidelines for its members**
  - Blockchain startup & ICO: bank account?
  - Some banks not even talk (personal experience)
  - Bank Frick accepts but is expensive
  - Bank Zarattini, Hypothekarbank Lenzburg, open for discussion, partially accepting
  - Two types:
    - Blockchain companies without an ICO: works
    - Blockchain companies with ICOs: very difficult
      - “The guidelines recommend that the ICO organiser should apply the relevant Swiss standards on the origin of funds (KYC) and money laundering (AML) when accepting cryptocurrencies under an ICO.”
- **Bitmain:**
  - 13.09.2018: A New Line of Powerful ASIC Miners Is Coming to Ethereum
  - 26.09.2018: Bitmain By the Numbers: An Inside Look at a Bitcoin Mining Empire
    - 880m$ in crytocurr. 900m$ profit, IPO
- **28.09.2018: Ethereum Security Lead Joins Effort to Oust Blockchain's Big Miners**
  - Seeking to block powerful ASIC miners
  - Discussion in an Ethereum dev meeting
- **28.09.2018: BitTorrent to Integrate Tron Tokens in New Incentive Model**
  - Earn TRX tokens for remaining online for longer periods of time
  - Users can pay seeders TRX

Distributed Systems - In the *News*

- **26.09.2018: [BitTorrent Traffic is Not Dead, It's Making a Comeback]**
  - Globally BT: 3% downstream, 22% upstream
  - EMA 32% upstream, rising
  - Why rise?
    - Increased fragmentation in the streaming service
    - Exclusive content available on a single streaming or broadcast service – think Game of Thrones for HBO, House of Cards for Netflix, The Handmaid's Tale for Hulu, or Jack Ryan for Amazon

Filesharing is not dead: BitTorrent is #1 in Europe and APAC, and #2 in Americas for upstream bandwidth consumption

Source: Sandvine Global Internet Phenomena Report, October 2018

URL: b.socrative.com/student/

Room: HSR

WebRTC – Introduction

- **WebRTC for browser to browser communication**
  - P2P, no server involved (~mostly)
  - Main goal: eliminate plugins or native apps
  - Supported by Google, Microsoft, Mozilla, Opera
- **Google bought in 2010 GIPS and open sourced WebRTC in 2011**
- **Protocol standardized by IETF (codec requirements, media protocol), JavaScript API by W3C**
- **Supported by Chrome, Firefox (now many others)**

- **Compatibility**
  - Microsoft Edge 15~
  - Google Chrome 56+
  - Mozilla Firefox 44+
  - Safari 11+
  - Opera 43+
- **Workarounds**

WebRTC – Introduction

- **Filling gap in the Web-Experience**
  - Video Chat → Google Hangouts Plugin, Flash, Java
  - Multimedia / Confernces → 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, appear.in**
- **Standard not finalized**
  - W3C Candidate Recommendation 27 September 2018
  - «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 a page?
- **WebRTC API could be simplified?**
- **Security Concerns**
  - Private IP / IP behind VPN, Tor?
- **But: WebRTC forbids unencrypted communication**
  - DTLS, SRTP
- **Data Streams**
  - SCTP over DTLS over UDP

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

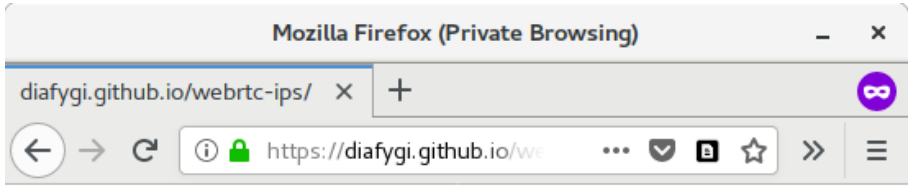
This demo secretly makes requests to STUN servers that can log your request. These requests by browser plugins (AdBlock, Ghostery, etc.).

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:

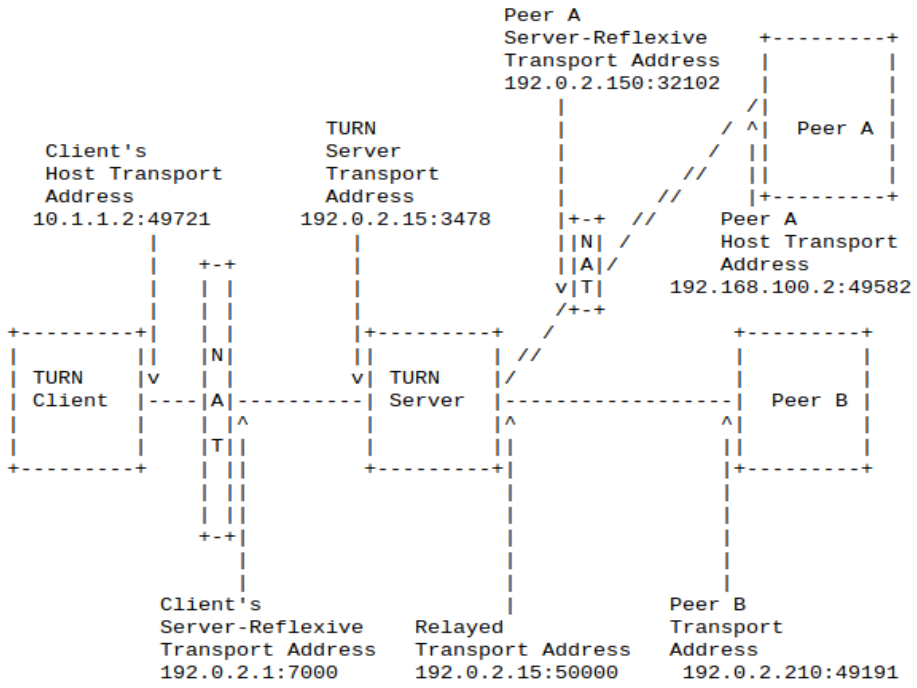
- 192.168.1.139

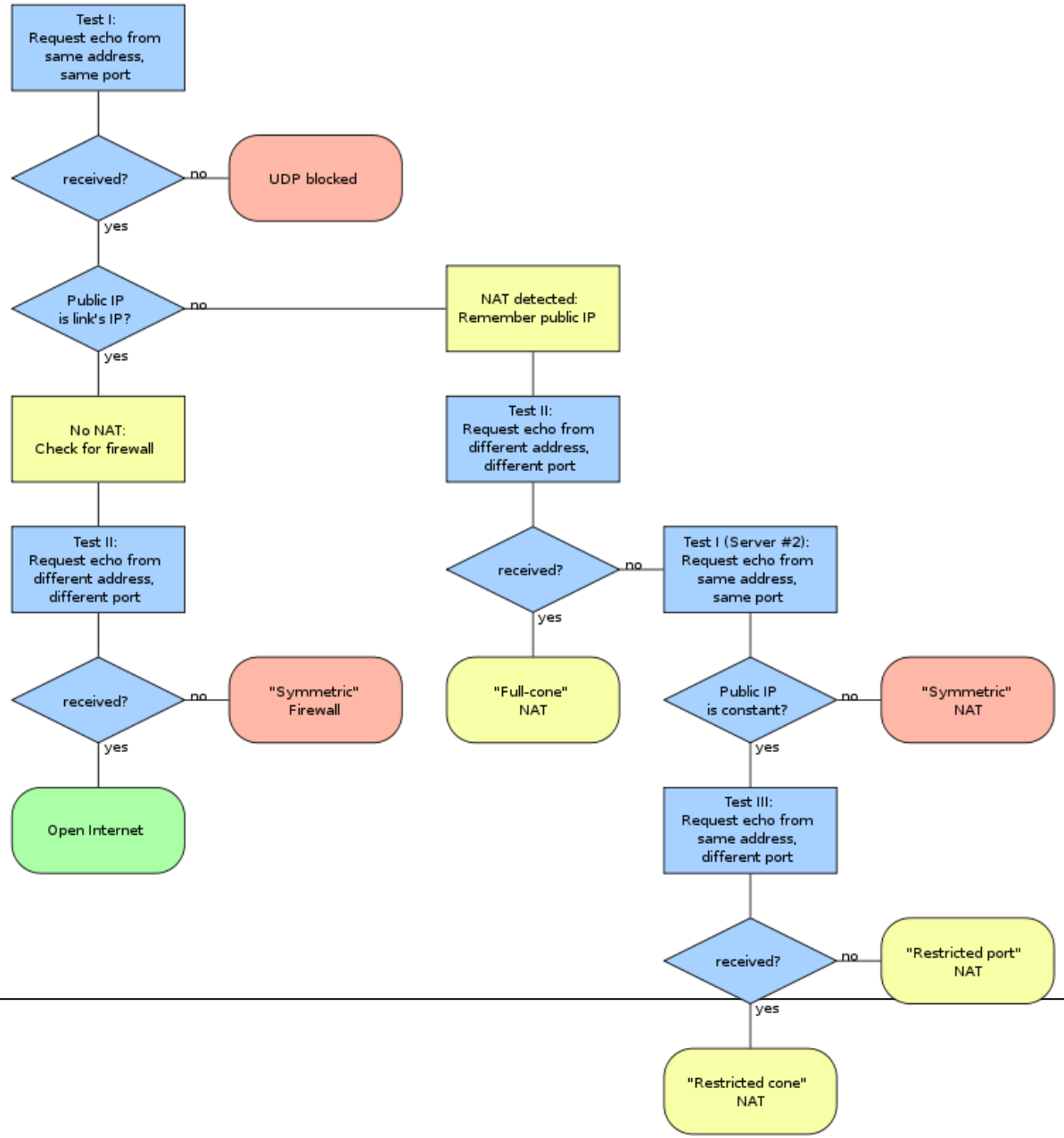
Your public IP addresses:

Your IPv6 addresses:

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)
    - “STUN is not a NAT traversal solution by itself”
  - TURN: **t**raversal **u**sing **r**elays around **N**AT

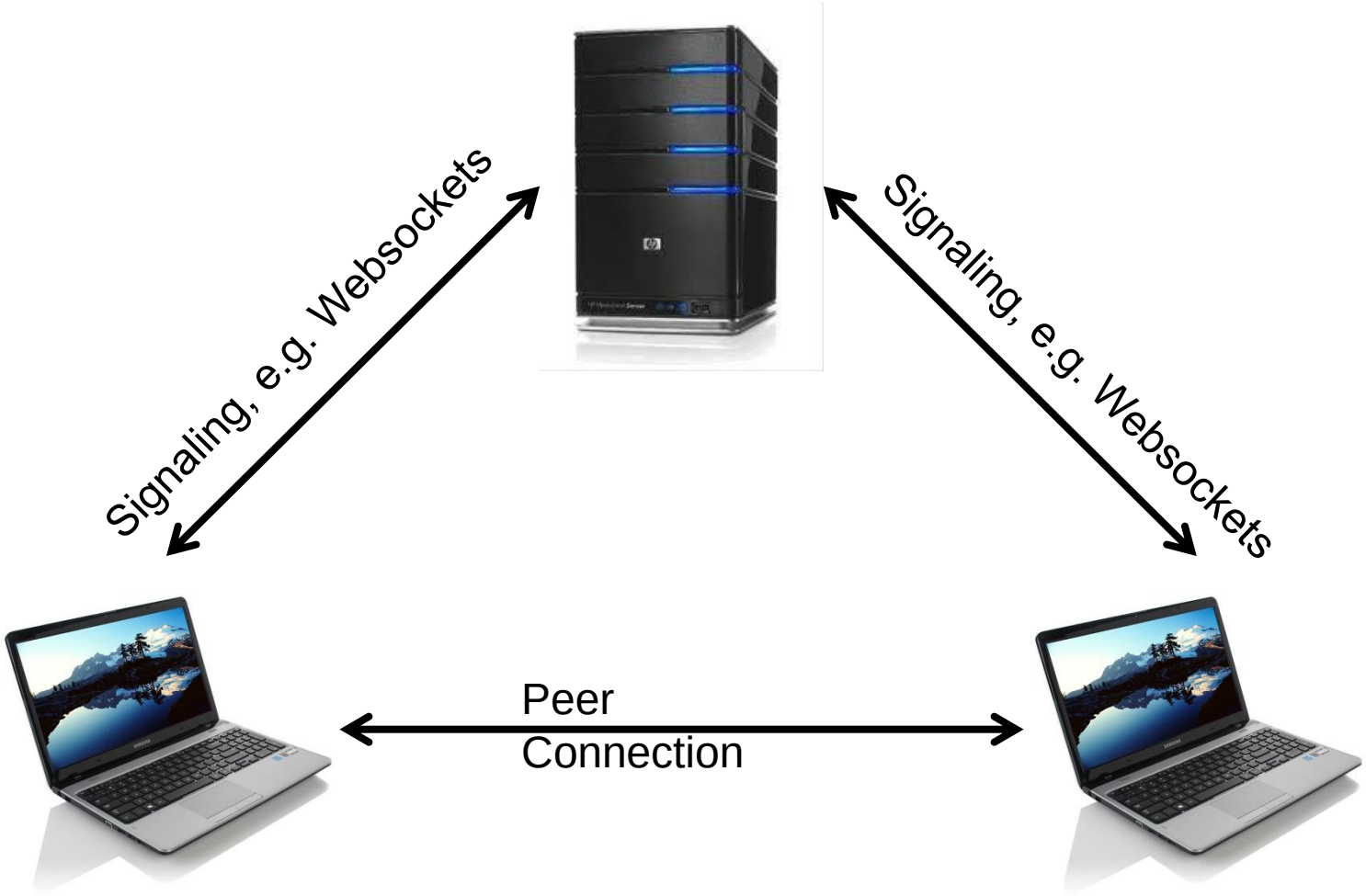




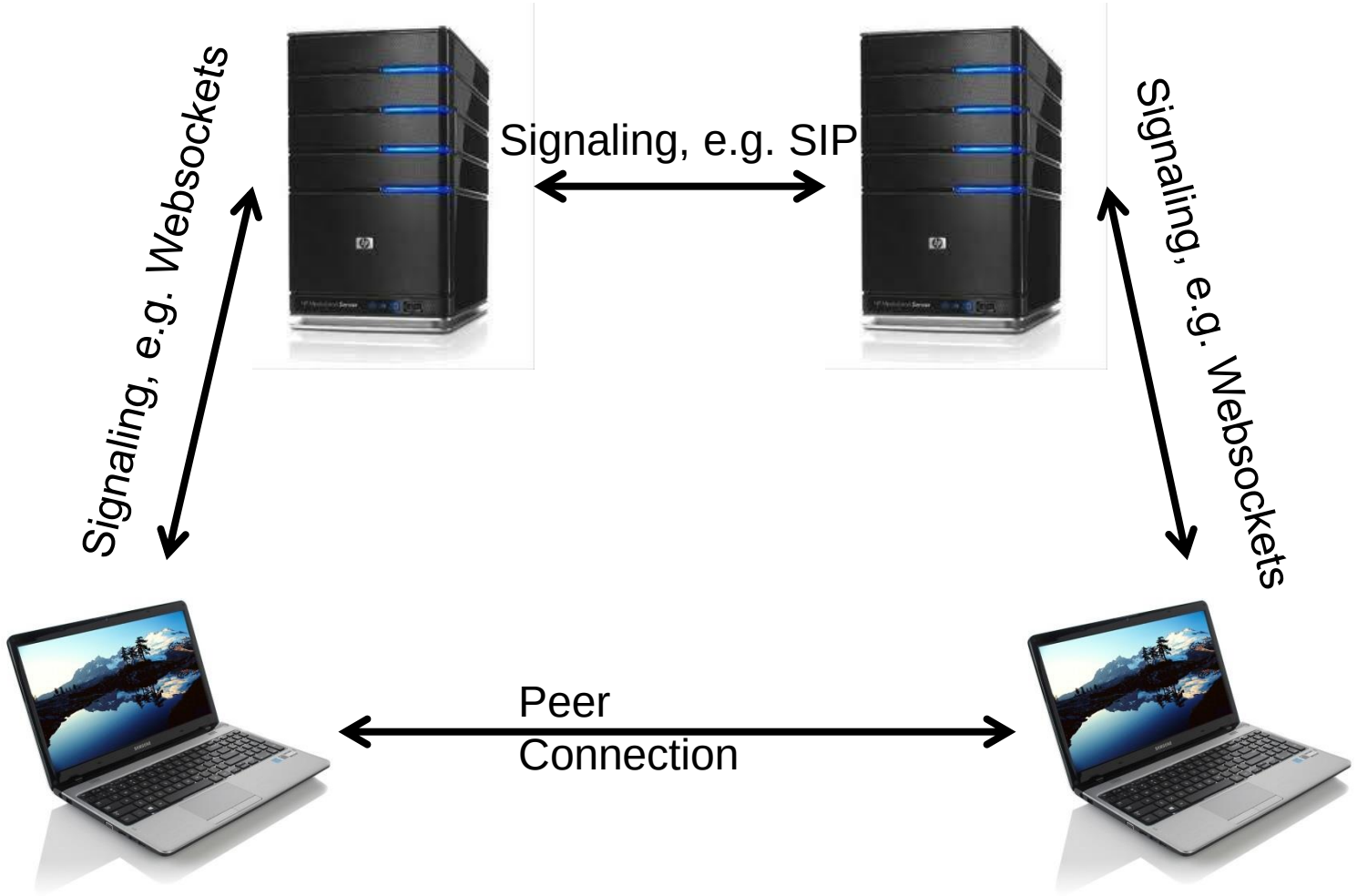
WebRTC – Introduction

- **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
- **ICE - Interactive Connectivity Establishment**
  - RFC 8445 NAT traversal for UDP
  - “ICE works by exchanging a multiplicity of IP addresses and ports, which are then tested for connectivity by peer-to-peer connectivity checks.”
  - Connectivity checks: STUN, TURN
- **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 …`

WebRTC Architecture - Triangle



WebRTC Architecture - Trapezoid



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
  - 22 loc
- **Run**
  - yarn install
  - node server.js
- **Minimal example!**

Example: https://github.com/tbocek/DSA-webrtc

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

// App setup
var app = express();
var server = app.listen(4000, function () {
    console.log('listening for requests on port 4000.');
});

// Static files
app.use(express.static('.'));
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.broadcast(message);
    });
});

wss.broadcast = function(data) {
    this.clients.forEach(function(client) {
        if(client.readyState === WebSocket.OPEN) {
            client.send(data);
        }
    });
};
console.log('Server running.');
```

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");
}
```

Example: https://github.com/tbocek/DSA-webrtc

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

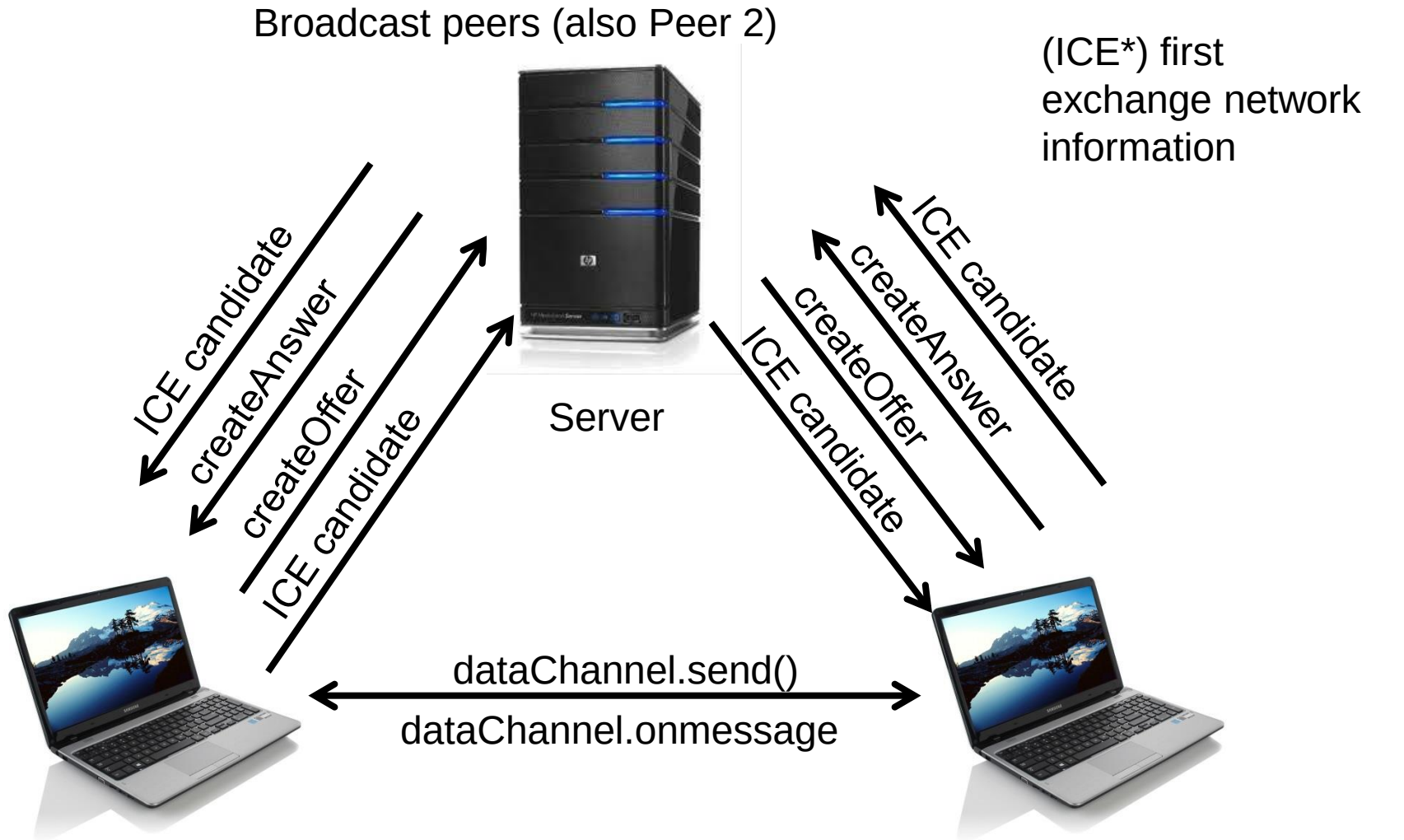
Example: https://github.com/tbocek/DSA-webrtc

```
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



URL: b.socrative.com/student/

Room: HSR

WebRTC – Outlook

- **Strong focus on VoIP [demo]**
  - Skype competitor?
  - Microsoft / IE 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
- **New types of applications – Conferencing, gaming, P2P file sharing**
  - PeerCDN, serve static content from browsers of other visitors
- **ORTC, WebRTC 1.1?**
  - Object Real-Time Communications (ORTC) instead Session Description Protocol (SDP)

URL: b.socrative.com/student/

Room: HSR

Questions?

Prof. Dr. Thomas Bocek thomas.bocek@hsr.ch
Roman Blum roman.blum@hsr.ch