HS19

DISTRIBUTED SYSTEMS ADVANCED & BLOCKCHAIN

WebRTC

T. Bocek

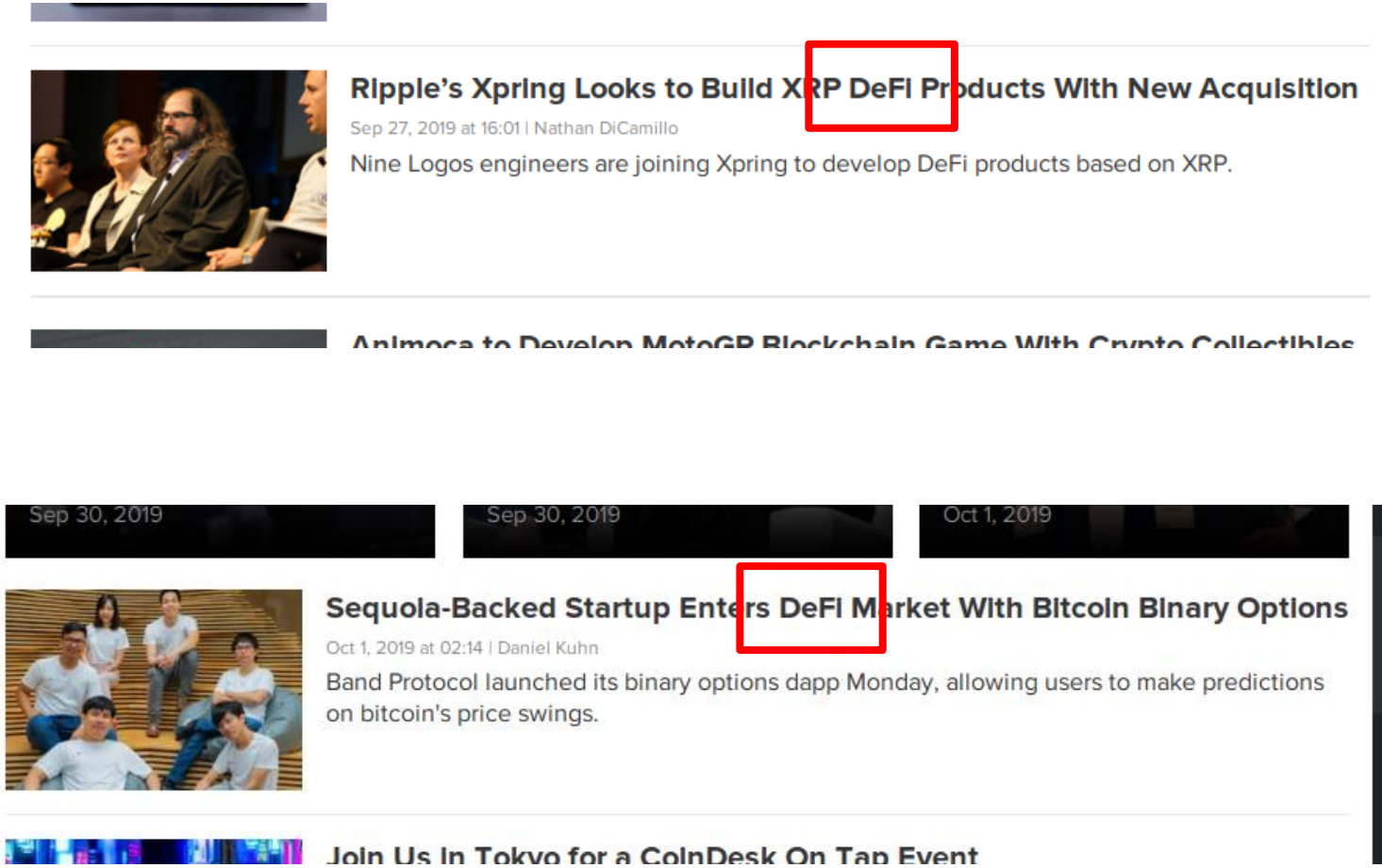
thomas.bocek@hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Distributed Systems - In the News

- **DeFi?**
  - No this one
- **Definition 1**
  - “DeFi is essentially just conventional financial tools built on a blockchain — specifically Ethereum.”
- **Definition 2**
  - “Decentralized Finance (DeFi) is the movement that leverages decentralized networks to transform old financial products into trustless and transparent protocols that run without intermediaries. ”
- **Definition 3**
  - “Everything that you know in today’s centralized financial world if put on decentralized infrastructure through public blockchains like Ethereum or Bitcoin, is decentralized finance.”



Distributed Systems - In the *News*

- **27.09.2019: Full Disclosure: CVE-2019-12998 / CVE-2019-12999 / CVE-2019-13000**
  - A lightning node accepting a channel must check that the funding transaction output does indeed open the channel proposed.
  - "Three months is not a long time. It's a pretty short time because you have to give users the amount of time needed to update. … A lot of users don't do it."
- **28.09.2019: Google Achieves Quantum Supremacy. Is Encryption Safe?**
  - 53 qubits, breaking RSA requires few 1000 qubits
  - "apply a random (but known) sequence of quantum operations" – hard to simulate, proof that quantum computer are faster

- **2019-06-27: Bug discovered, LND and Eclair notified.**
- **2019-06-28: CVEs assigned.**
- **2019-07-02: lnd v0.7.0-beta released.**
- **2019-07-03: Eclair 0.3.1 released.**
- **2019-07-04: c-lightning 0.7.1 released.**
- **2019-07-06: disclosure to other projects begins (rust-lightning, ptarmigan, BLW).**
- **2019-07-30: lnd v0.7.1-beta released.**
- **2019-08-17: [Review next dates based on deployment stats/problems]**
- **2019-08-30: Reveal existence of CVEs, encourage laggards to upgrade.**
- **2019-09-07: First conclusive evidence of exploit attempt in the wild.**
- **2019-09-27: Full disclosure of CVEs.**
- **2019-09-27: Submit PR to spec to require this.**

Distributed Systems - In the News

- **30.9.2019: [Ethereum's Istanbul Upgrade Will Break 680 Smart Contracts on Aragon]**
  - Istanbul hard fork on the way, released on Monday on Ropsten testnet.
  - SLOAD now 800 gas instead 200 gas. For contracts with predefined gas: issue!
  - Slide 20, lecture 2:

Appendix G. Fee Schedule

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{suicide}$	24000	Refund given (added into refund counter) for suiciding an account.
$G_{suicide}$	5000	Amount of gas to pay for a SUICIDE operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.

URL: b.socrative.com/student/

Room: HSR

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 intially by Chrome, Firefox (now many others)**

WebRTC

- **Compatibility, 88%**
  - Microsoft Edge 15~
  - Google Chrome 56+
  - Mozilla Firefox 44+
  - Safari 11+
  - Opera 43+
- **PeerJS**
  - Not all browser work 100% compatbile 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 – 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, whereby.com**
- **Standard still 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

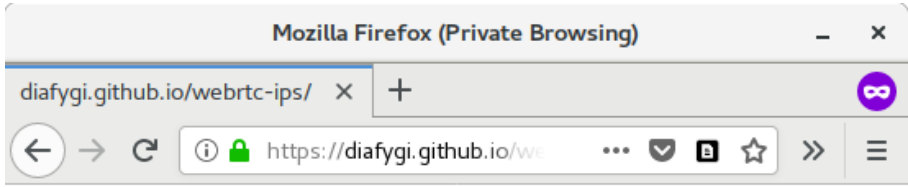
This demo secretly makes requests to STUN servers that can log your request. These requests by browser plugins (AdBlock, Ghostery, etc.).

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:

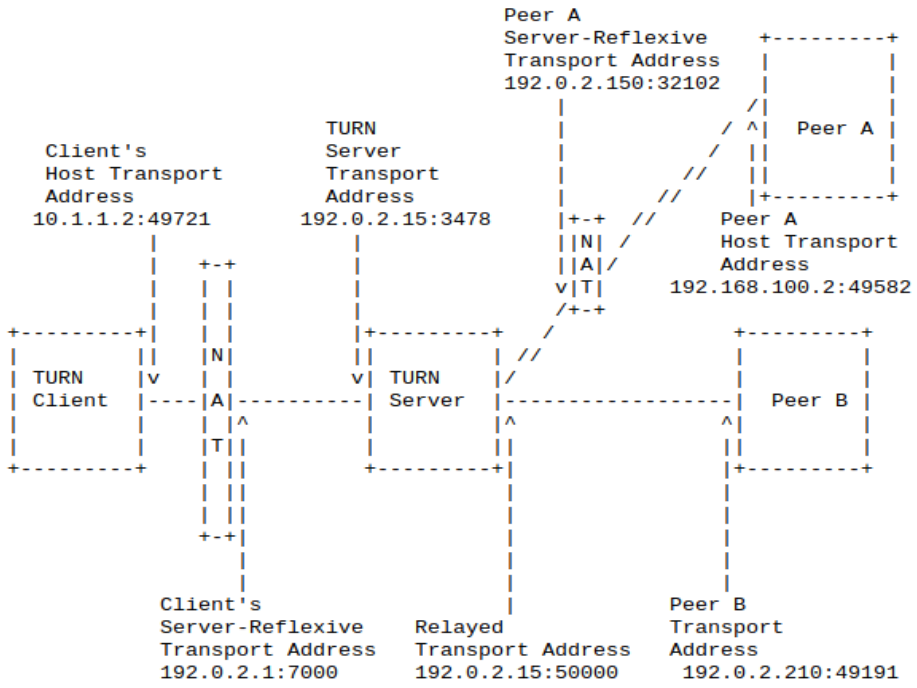
Your IPv6 addresses:

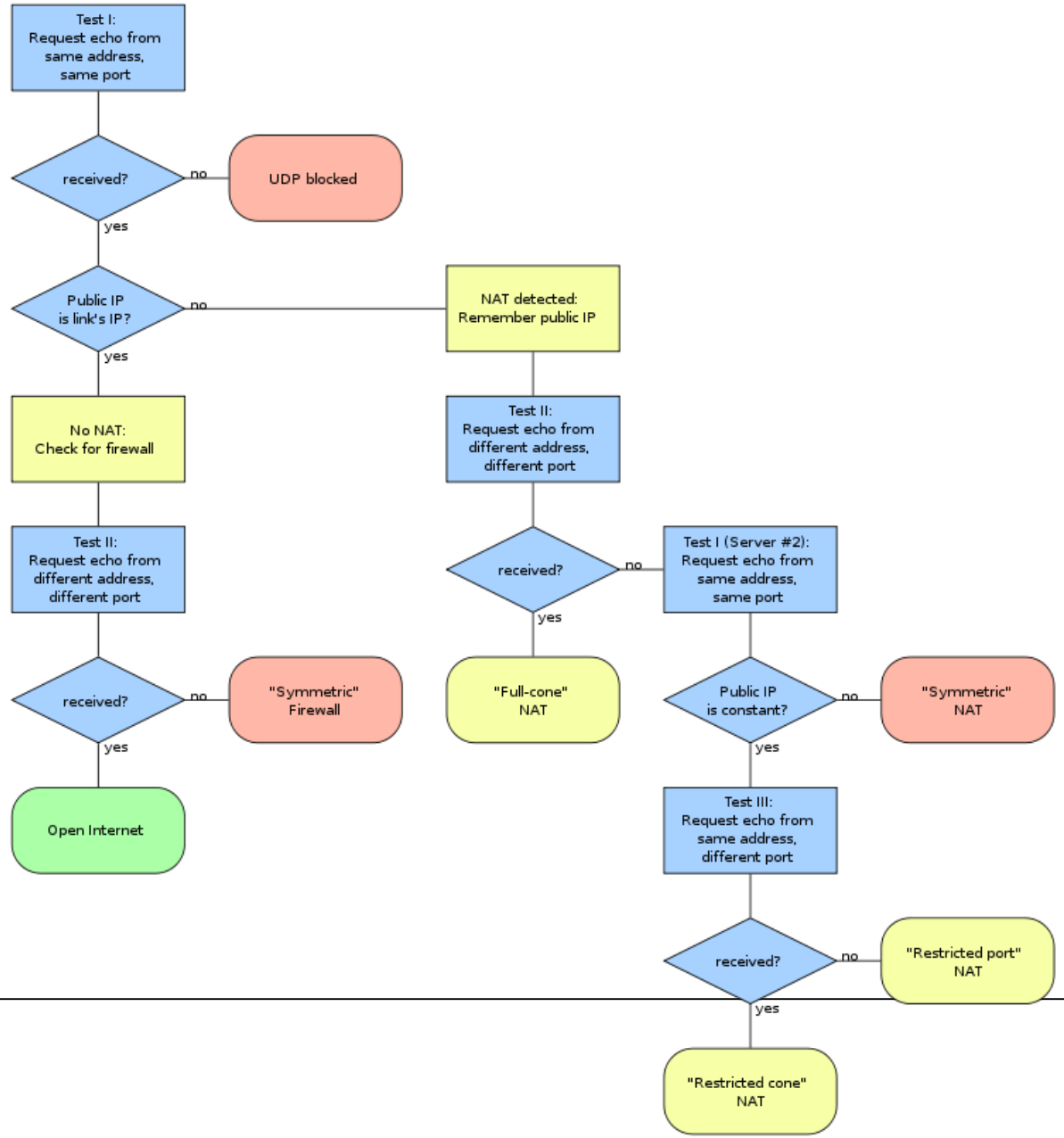
URL: b.socrative.com/student/

Room: HSR

WebRTC – Introduction

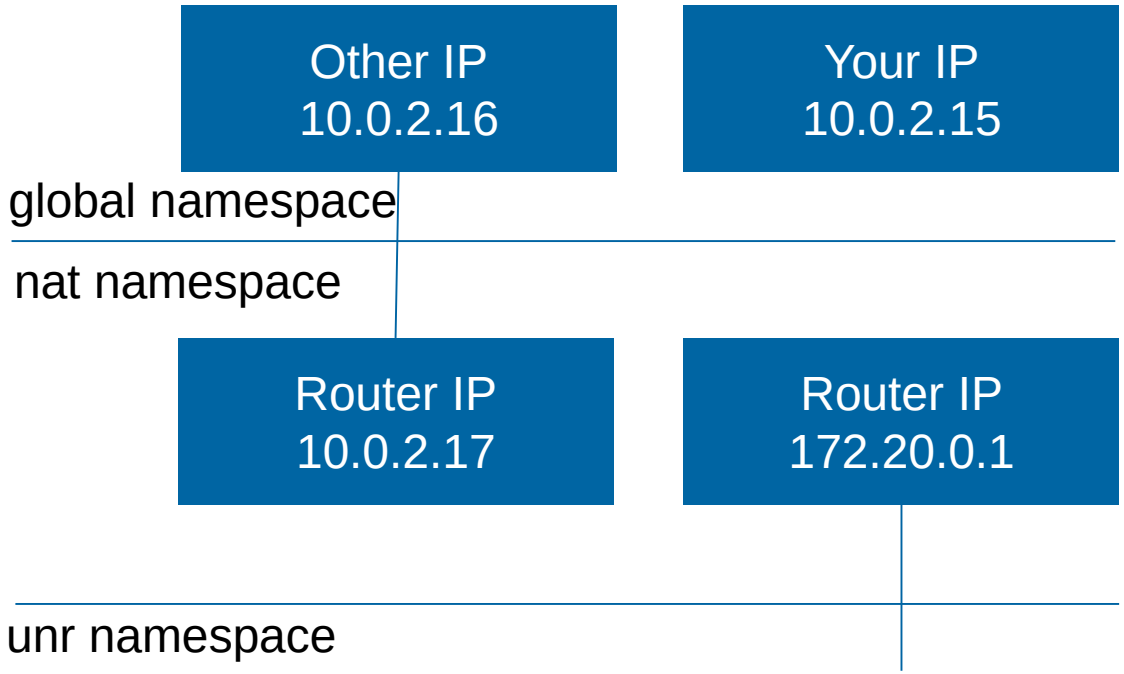
- **On the bright side: developer does not need to care about NAT**
  - Abstraction using STUN, ICE, TURN
  - STUN: session traversal utilities for NAT (detect which kind of NAT, rfc5389)
    - “STUN is not a NAT traversal solution by itself”
  - TURN: **t**raversal **u**sing **r**elays around **N**AT





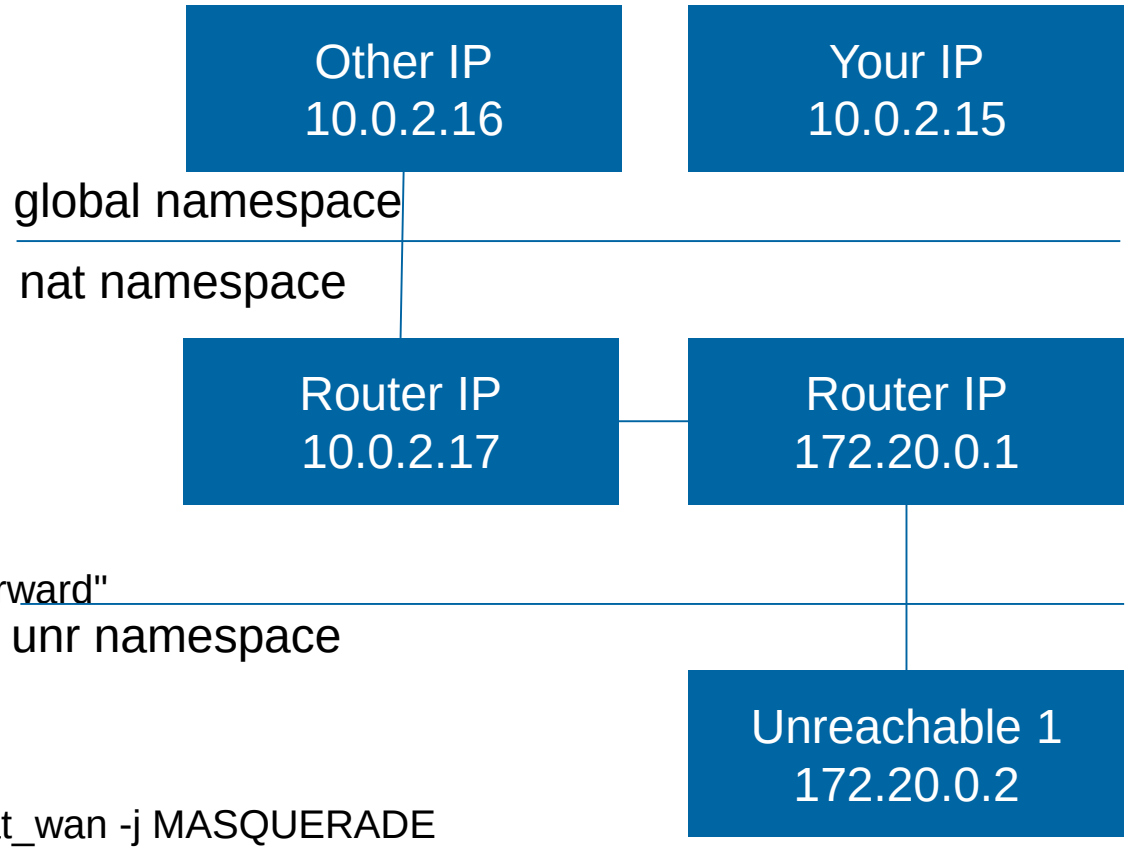
VSS, FS19: Make your own Testbed for P2P System

- **Network namespaces – also used in docker**
  - We do it manually
- **veth - Virtual Ethernet Device**
  - Tunnels between network namespaces
    - ip netns add unr (nat) / ip netns list
    - ip link add nat_wan type veth peer name nat_real
    - ip link add nat_lan type veth peer name unr_lan
    - ip link set nat_wan netns nat
    - ip link set nat_lan netns nat
    - ip link set unr_lan netns unr
    - ip address add 10.0.2.16/24 dev nat_real
    - ip link set nat_real up
    - ip netns exec nat ip address add 10.0.2.17/24 dev nat_wan
    - ip netns exec nat ip link set nat_wan up
    - ip netns exec nat ip address add 172.20.0.1/24 dev nat_lan
    - ip netns exec nat ip link set nat_lan up



VSS, FS19: Make your own Testbed for P2P System

- **veth - Virtual Ethernet Device**
  - Tunnels between network namespaces
    - ip netns exec unr ip address add 172.20.0.2/24 dev unr_lan
    - ip netns exec unr ip link set unr_lan up
    - ip route add 10.0.2.17 dev nat_real
    - ip netns exec unr route add default gw 172.20.0.1
    - ip netns exec nat route add default gw 10.0.2.16
    - ip netns exec nat sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
    - echo 1 > /proc/sys/net/ipv4/ip_forward
    - echo 1 > /proc/sys/net/ipv4/conf/all/proxy_arp
    - ip netns exec nat iptables -t nat -A POSTROUTING -o nat_wan -j MASQUERADE
    - ip netns exec nat iptables -A FORWARD -i nat_wan -o nat_lan -m state --state REL
    - ATED,ESTABLISHED -j ACCEPT
    - ip netns exec nat iptables -A FORWARD -i nat_lan -o nat_wan -j ACCEPT



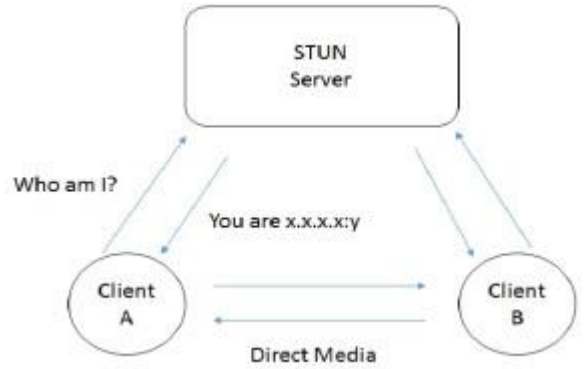
VSS, FS19: Exercises

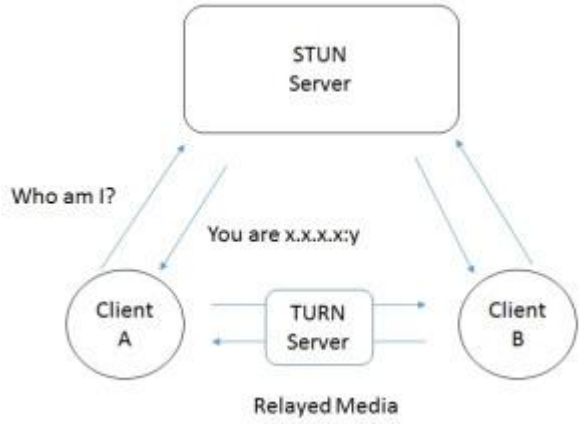
- Download VirtualBox
- Download alpine image (36.9MB)
- SSH forwarding enabled on port 2222
- Login: ssh root@localhost –p 2222
- Password: hsr12345
- ./nw.sh is the networking setup – the two previous slides
- ./nat.sh run shell in nat namespace
- ./unr.sh run shell in unr namespace

- nc -u -p 3000 10.0.2.15 4000
- nc -u -l -p 3000
- nc -u -s 10.0.2.15 -p 4000 10.0.2.17 3000

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.”

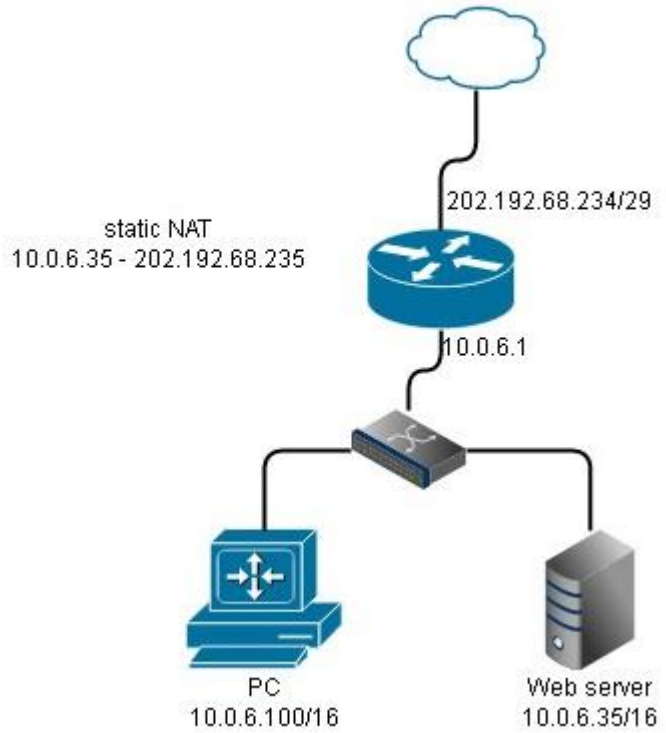




source

WebRTC – Introduction

- **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 …`

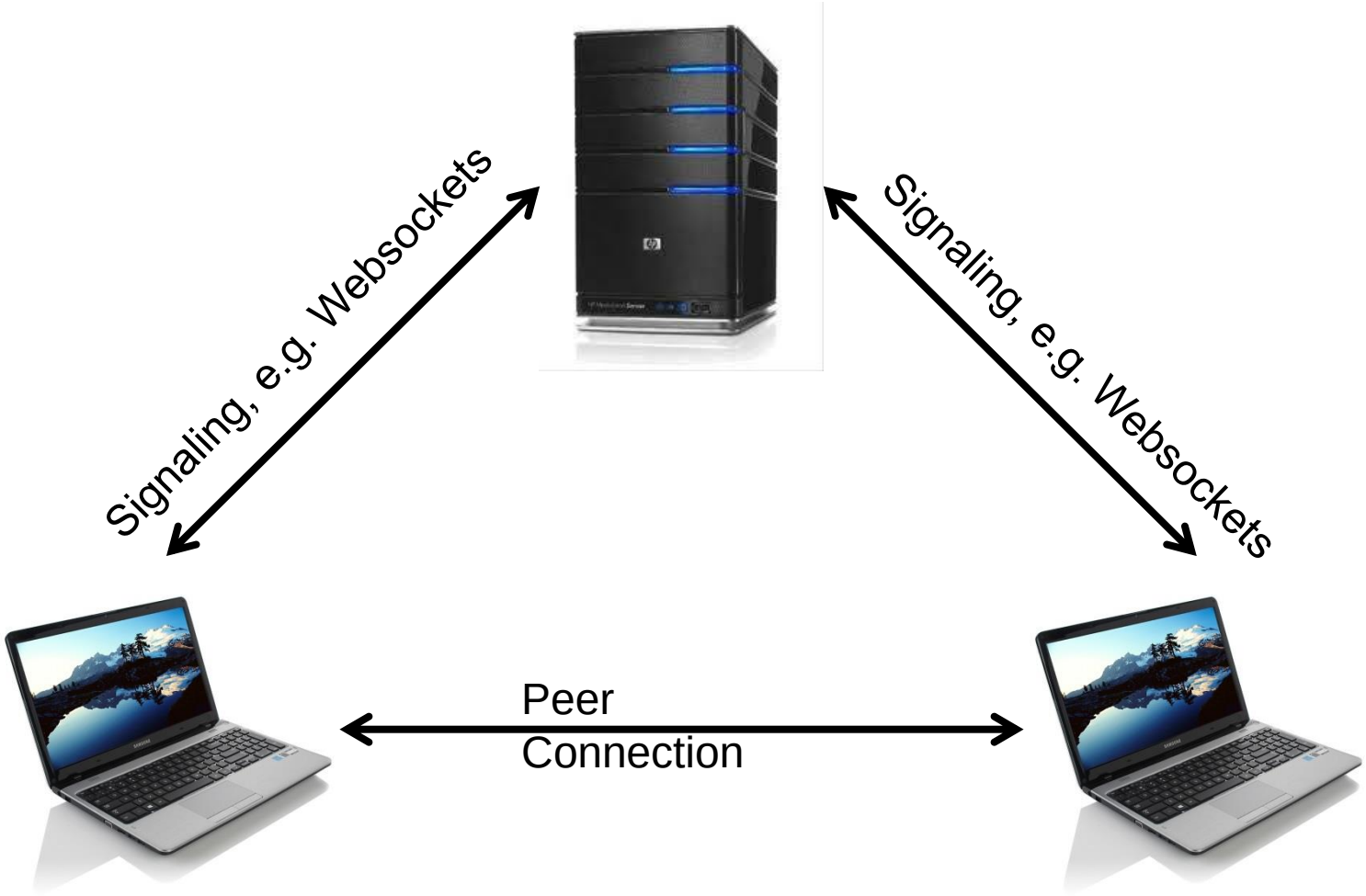


https://networkengineering.stackexchange.com/questions/25383/cisco-nat-hairpinning

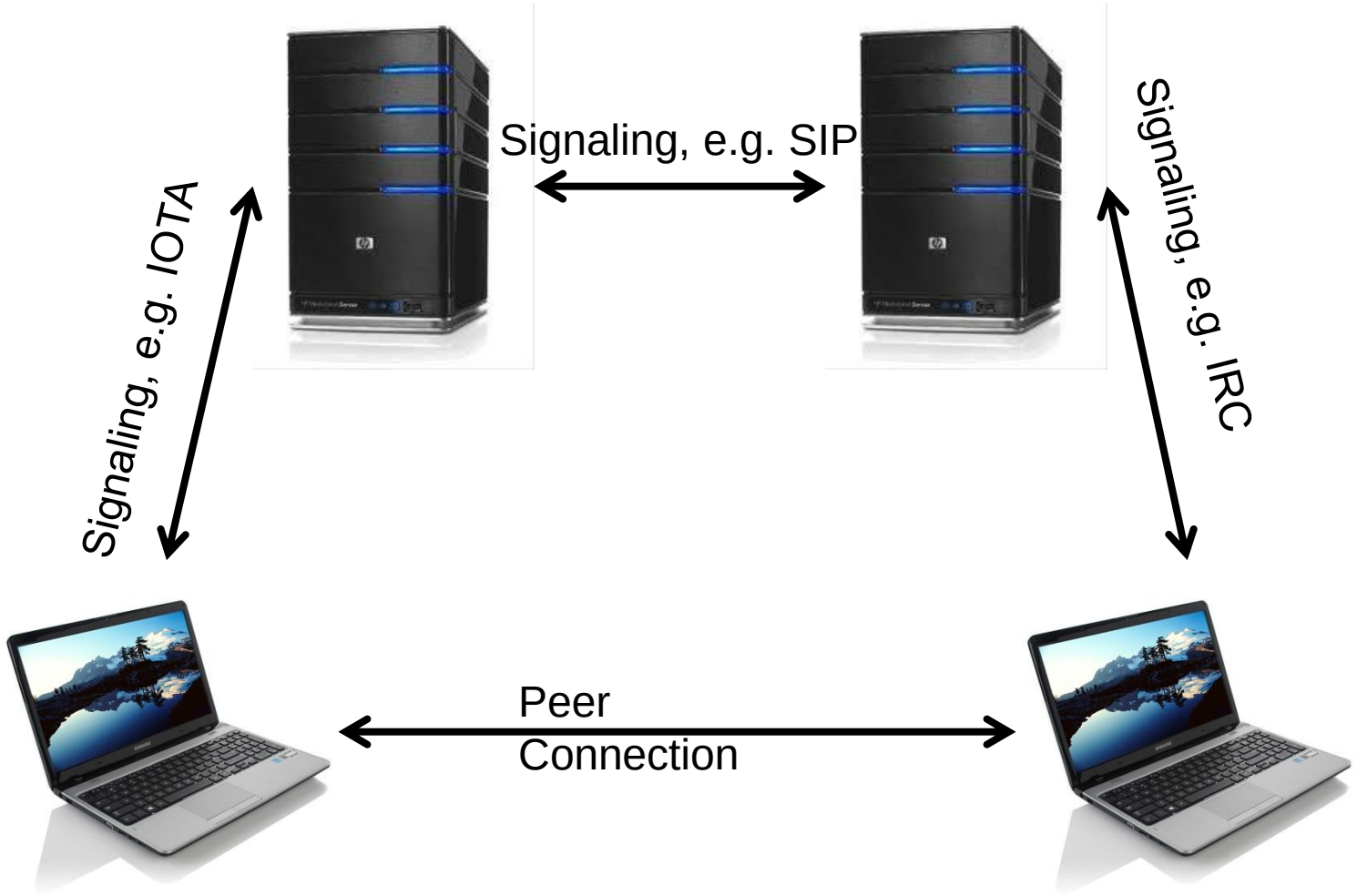
URL: b.socrative.com/student/

Room: HSR

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-HS18/tree/master/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

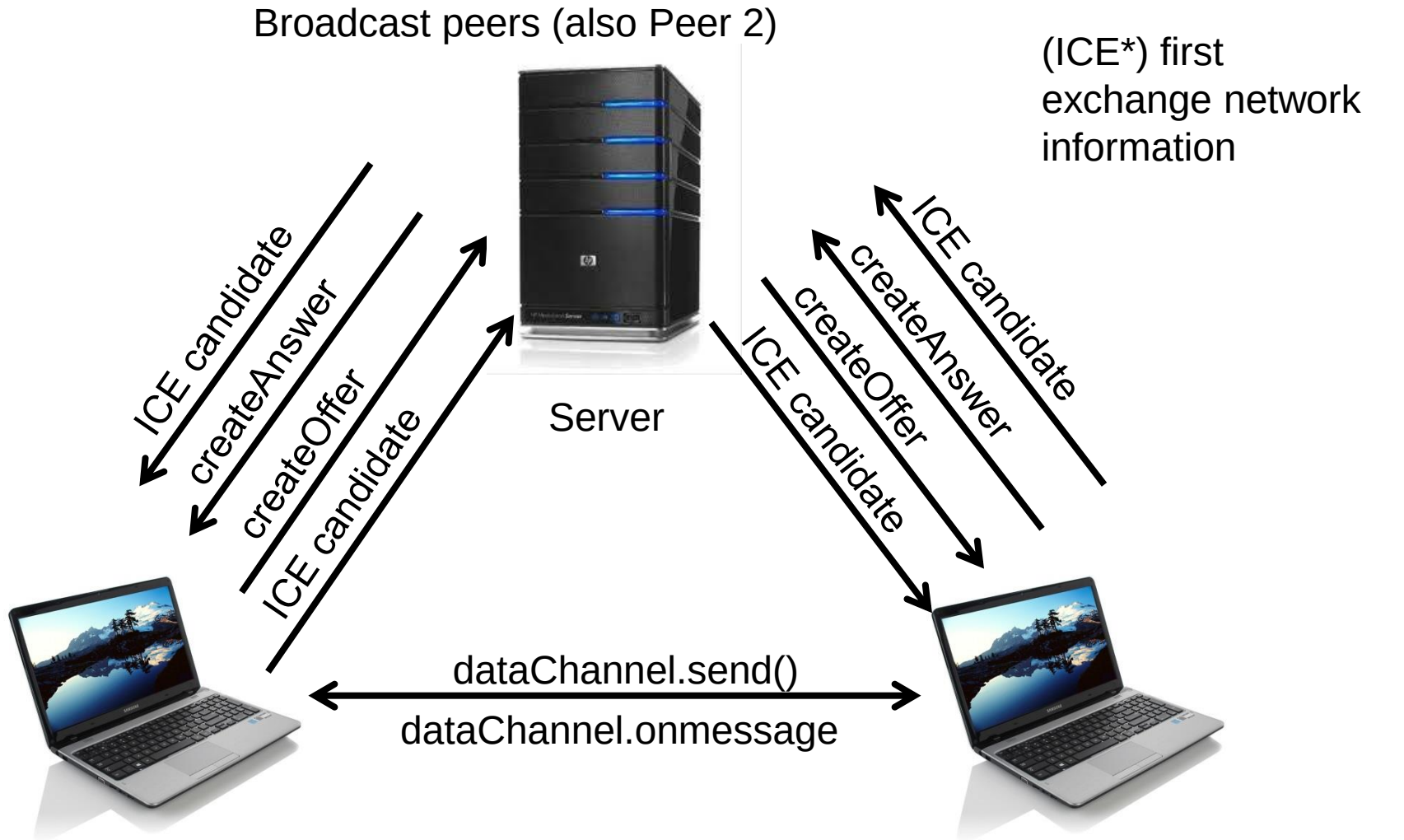
Example: https://github.com/tbocek/DSA-webrtc

```
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



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, 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)

URL: b.socrative.com/student/

Room: HSR

Dr. Thomas Bocek thomas.bocek@hsr.ch