A WebRTC Video Chat Implementation Within the Yioop Search Engine

Yangcha K. Ho

5/20/2019

"You Raise me up"

- When I am down, and, oh, my soul, so weary …
- You raise me up, so I can stand on mountains
  You raise me up to walk on stormy seas
  I am strong when I am on your shoulders
- …
  You raise me up to more than I can be
- You raise me up, so I can stand on mountains
  You raise me up to walk on stormy seas
- This Lyric sums up my words. Thank you for providing me a solid foundation and being patient with me.
-

Table of contents:

A WebRTC Video Chat Implementation Within the Yioop Search Engine

- Suppose both Bob and Alice has logged into Yioop.com, and they want a video chat with each other. Bob is in New York and Alice is San Jose.
- Bob selects Alice from the drop down list.
- Both Bob and Alice are connected to the signaling server which sends messages to each one.
- When Bob clicks “call” button to call Alice, the signaling server informs that Bob wants to call Alice.
- Alice must accept the call from Bob.

What is WebRTC ?

- Not a single piece of software

  It utilizes a collection of technologies such as encryption algorithm, HTML5, JavaScript APIs, and several network protocols
- The World Wide Web (W3C) standardizes APIs
- IETF – standardizes various protocols

Self Help Sites vs This Research Project

- May 2011, Google released WebRTC as open source project.
- Popular since then and already commercialized: tokbox.com
- Many self-help sites to teach WebRTC technology.
- They run on one desktop with two browsers sharing the same memory. They neither use signaling server nor encryption algorithm.
- This project runs an encryption algorithm on two different desktops, with a home made signaling server.
- Built in an academic setting, it has all the components comparable to commercial WebRTC sites.

Before HTML5

- To develop an application with video/audio feature on browser you had to use Flash, or Silverlight, or A Java applet.
- Also need a third party plug in software to make it work.
- Also need to implement codecs; divide the frames in smaller chunks, compress them and do it reverse order in the other end.

Example of HTML5 tag: video tag

- <html>
- …
- <body>
- **<video id**="localVideo" playsinline autoplay muted></video>
- **<video id**="remoteVideo" playsinline autoplay>**</video>**
- …
- </body>
- </html>

WebRTC comes with Three APIs and Other components

- **The three JavaScript APIs:**
  - a) getUserMedia() handles video and audio streams
  - b) RTCPeerConnection() handles major communication
  - c) RTCDataChannel() handles data transfer

- **Other Components:**
- Encryption framework, STUN/TURN servers, signaling server, ICE, SDP, NAT, UDP, TCP.

getUserMedia() API

```
<video autoplay></video>
<script>
var constraints = { video: true, audio: true, };
```

- **if(navigator.getUserMedia) {**

```
    navigator.getUserMedia(constraints, mediaOK,
    mediaError); }
    else { alert('Your browser does not support
            getUserMedia API'); }
</script>
```

Main Components of WebRTC RTCPeerConnection

function start(isCaller) {

- pc = new RTCPeerConnection(STUN/TURN Parameter);
- pc.onicecandidate = gotIceCandidate;
- pc.onaddstream = gotRemoteStream;
- pc.addStream(localStream); if(isCaller) {
- pc.createOffer(gotDescription, createOfferError); } }

RTCPeerConnection API – cont.

- pc = new RTCPeerConnection(STUN/TURN Parameter);
- The parameter lists array of STUN and TURN servers for locating the ICE candidates.
- Google's free public STUN servers at code.google.com
- Not many free TURN server and commercial sites available
- This project uses a public TURN server at: https://github.com/pions/turn.

RTCPeerConnection API

Responsible for connecting two peers.

1. pc.onicecandidate = gotIceCandidate;

- It initializes a connection, gathers ICE candidates - browser's public IP number and port
- Three different kinds of information must be exchanged between them: a) when to start/end, b) IP address, Port number and 3) codecs, and media types used.

onaddstream() & addStream()

- The pc object obtains local and remote media stream using the getUserMedia() method.
- This media steam must be attached to pc object via **onaddstream()** method for remote media stream. ***addStream()*** method for local video/audio stream.

pc.createOffer()

- The caller pc creates an offer using **createOffer().**
- It creates an <u>**SDP**</u> offer for a new connection to a remote peer.
- It contains the codecs, encryption methods, and ICE. Wraps inside RTCSessionDescription(offer) object.
- It attaches to pc.setLocalDescription() to send to its target peer through a signaling server.

createAnswer()

- The callee pc receives an offer in SDP format from the caller.
- The callee pc creates a SDP using createAnswer() method.

- This SDP is wrapped inside pc.setRemoteDescription().

- This process relies on several protocols and supporting architecture to make the connection takes place.

- Good place to describe supporting technologies.

What is a signaling Server ?

- Each browser might be behind some network, but each needs to find other peer to be connected between them.
- Each peer needs to figure out other peer’s codecs, settings, bandwidth, IP address, and its port accessed by outsider.
- They cannot do by themselves, they need a broker which can connect them.
- A signaling server does this broker role to establish and coordinate connection between these two peers.

Signaling Server – cont.

- But the connection must be secure.
- The original packets in transit not be modifiable if either peer is attacked. This is one of mandatory WebRTC requirements.

- But signaling process is not defined by the WebRTC Spec.
- One of the reason: to allow an interoperability among different protocols.
- Application developer has freedom to build a signaling server

Signaling Server – cont.

- Use any language or protocols to build a signaling server
- This project implemented two signaling servers:
- One is written in Node.js using WebSocket for a WebRTC video chat application.
- The other one which is written in PHP with WebSocket runs inside Yioop.com.
- The basic signaling process is the same in both cases - to exchange messages between two browsers.

Signaling Server – cont.

- Commercial signaling servers: Asterisk and OnSip.
  Skype_uses its own proprietary signaling server.
  Google "Hangouts", is free, but you must download its software.

- When starting the signaling process, the two browsers do not know each other's codecs, and media types that are used.

- Interactive Connectivity Establishment (ICE) comes to solve this problem.

ICE Candidate with signaling server

- Each ICE candidate contains its IP address and its port number.
- As soon as the two peers agree upon ICE candidates, they exchange the video stream and data.

- They continue exchanging ICE candidates, hoping to find better options until the current session ends.

- This project codes each ICE candidate with a JSON string message of type "candidate,”.

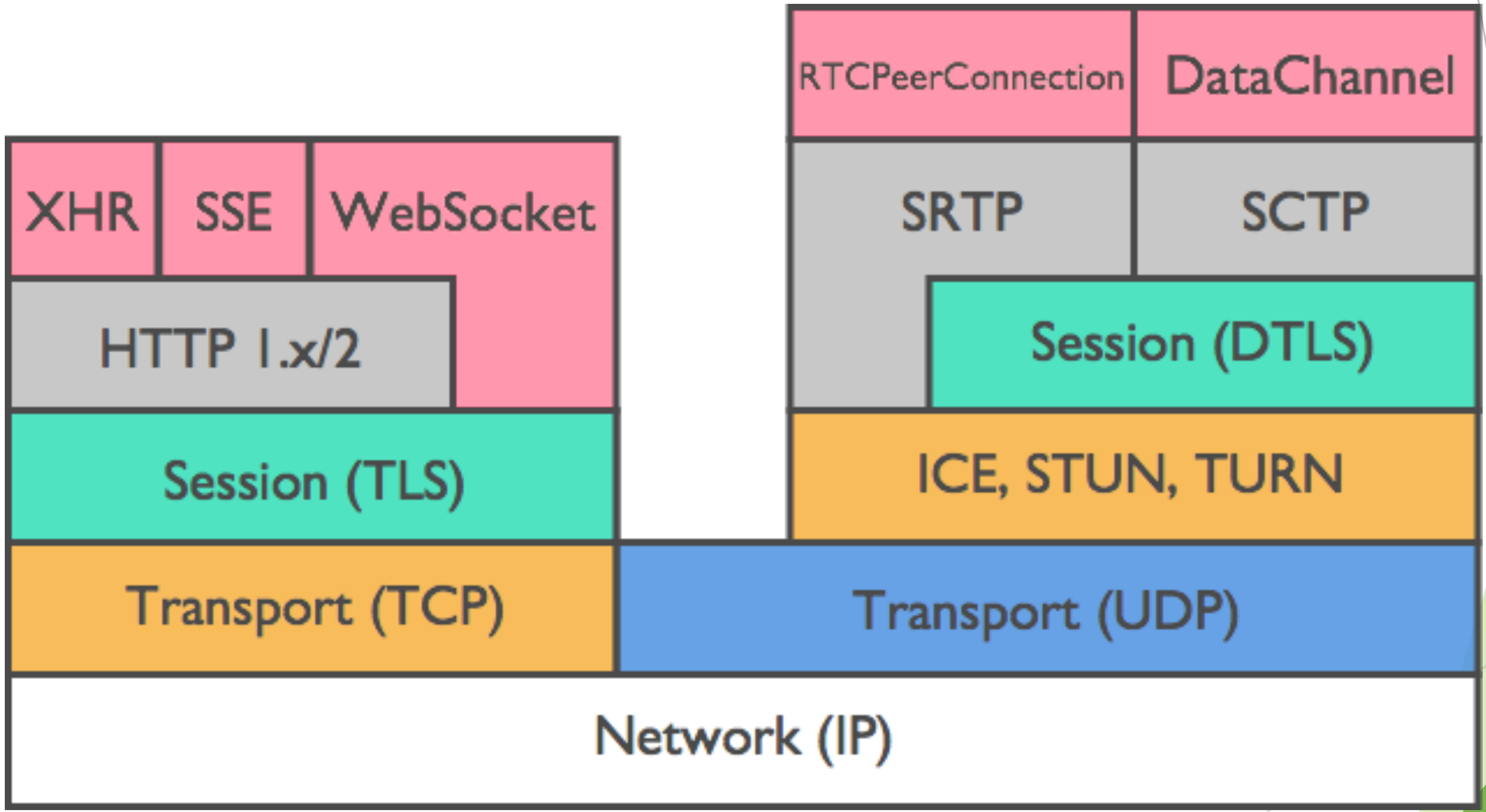
ICE and SDP

- The caller, (Alice), finishes gathering ICE candidates and creates an offer in Session Description Protocol (SDP) format, to initiate the call to the other party.

- Bob creates an answer in an SDP format in response to the offer from Alice.

- This paper writes signaling server using WebSocket to transmit offer messages with the type " webrtcmessage."

WebRTC protocol stack



Signal Server needs SDP

- A signal server plays a very important role in exchanging audio and video streams between two peers, but it cannot work alone.

- It needs support from several other underlying protocols and SDP is one of them.

SDP

- To share media-based data with the other peers over a network.
- SDP includes the name, purpose of the session, the media type, protocols, codec and its settings, timing, etc. and it is a kind of name card in a business world.
- When the pc object starts collecting ICE candidates, an SDP is created.

SDP – cont.

- The SDP has been around since the late 1990s for media-based connections such as phones before it is used in WebRTC.

- The SDP has a text-based format.

- It has a set of key-value pairs.
- Example: “<key>=<value>\n”.

- It uses mnemonic names such as shown below.

Example of SDP

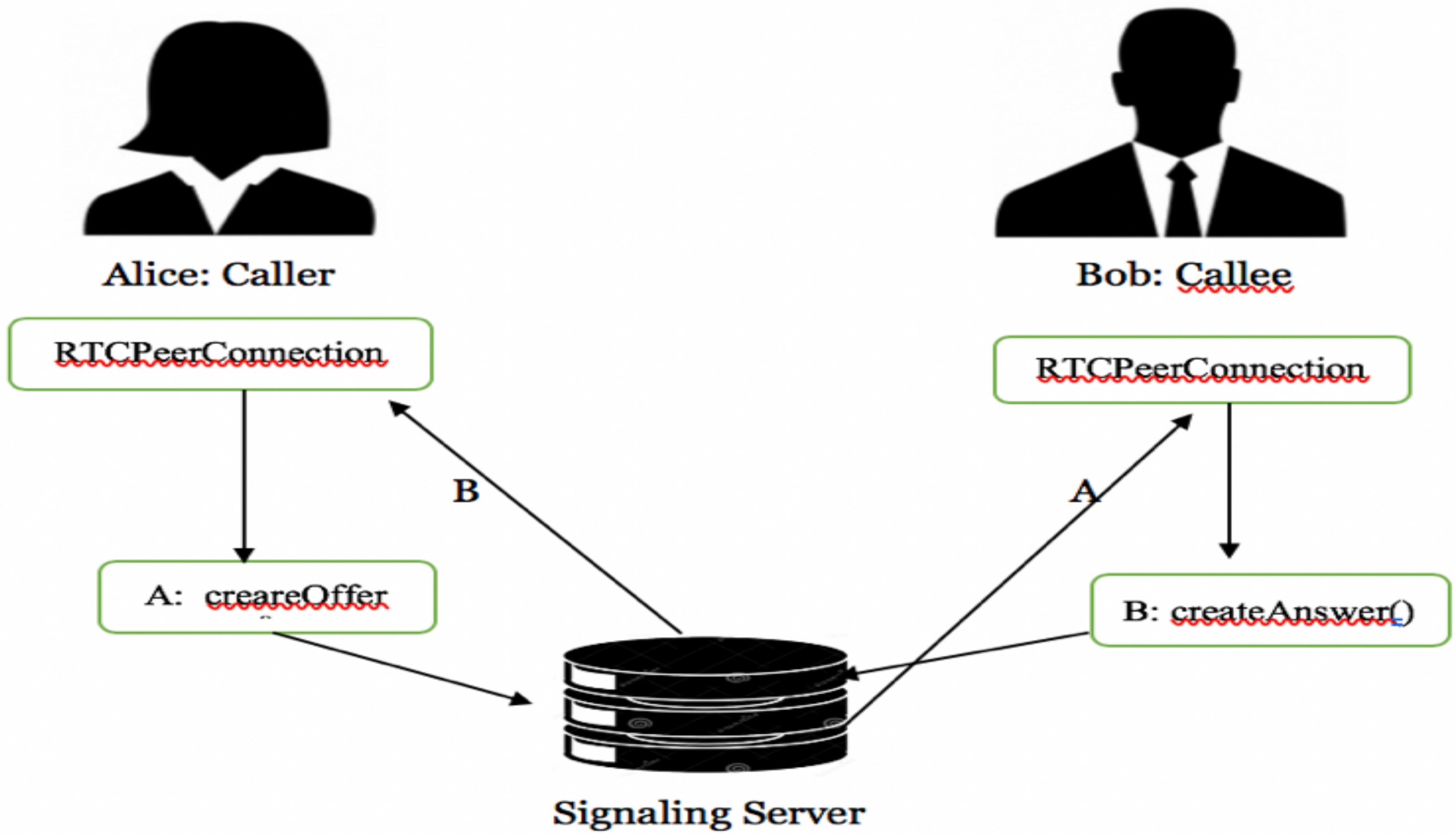
v = 0
o = mhandley2890844526 2890842807 IN IP4 126.16.64.4
s = SDP Seminar
i = A Seminar on the session description protocol
u = http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e = mjh@isi.edu(Mark Handley)
c = IN IP4 224.2.17.12/127
t = 2873397496 2873404696
a = recvonly
m = audio 49170 RTP/AVP 0
m = video 51372 RTP/AVP 31
m = application 32416udp wb
a = orient:portrait

WebRTC video chat application with the SDP

- Alice creates new objects from Signaling_Server and RTCPeerConnection.
- Call them signalServer and pc respectfully.
- Alice attaches getUserMedia() method pc.
- Alice creates SDP (offer) and attaches it to a local description() method.
- Alice sends this SDP offer to remote peer via signalServer.
- Bob, the recipient, returns an answer in an SDP format wrapped in a remote description method using signalServer.
- Both peers have established connection – see the figure

Bob calls Alice for Video Chat inside Yioop

Yioop.com – Alice selects Bob from drop drown list next to “CALL” button



A: Caller sends RTCPeerConnection.setLocalDescription() to callee.

B: Callee sends RTCPeerConnection.setRemoteDescription() to caller.

UDP (User Datagram Protocol)

- To deliver real-time communication in WebRTC.

- UDP uses timeliness, offers no promises on reliability, no guarantee, no orderliness of the data, and delivers each packet to the application the moment it arrives.

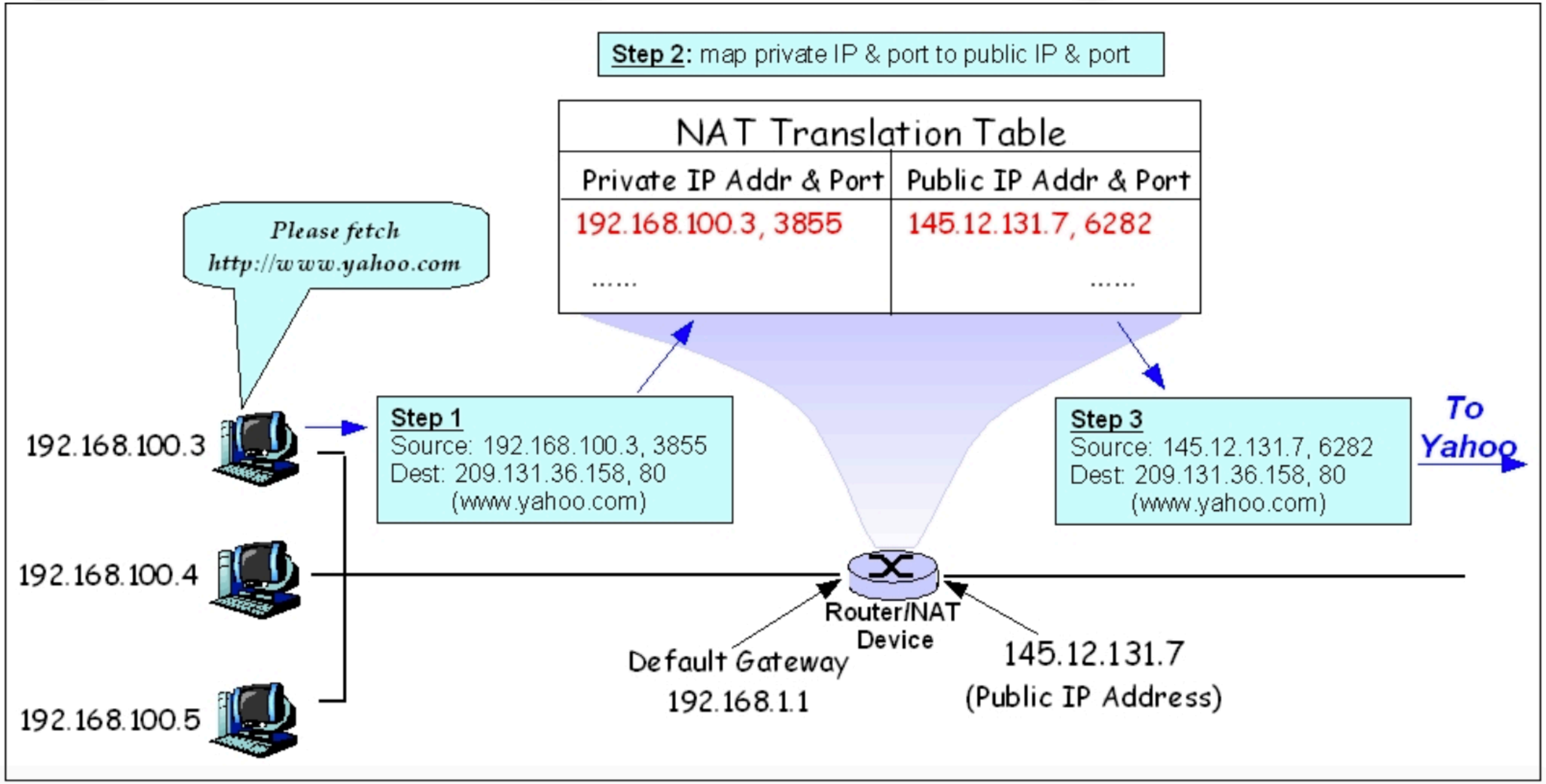
- If audio and video streams occasionally lose a few packets, the audio
- and video codecs makes up to fill in small data gaps, and users
- do not notice any a difference.

- WebRTC uses UDP at the transport layer, but UDP does not work alone. It needs support from other layers of NATs and firewalls.

Network Address Translation (NAT)

- A firewall or a router, maps one external IP address to a computer inside a private network.
- Allows several local devices can be connected to one public IP address to conserve the IPv4 address.
- When a device on the local network tries to send packets to outside, the NAT translates the IP address to match the external address.
- NAT devices also screens outside calls coming inside for security.

NAT example

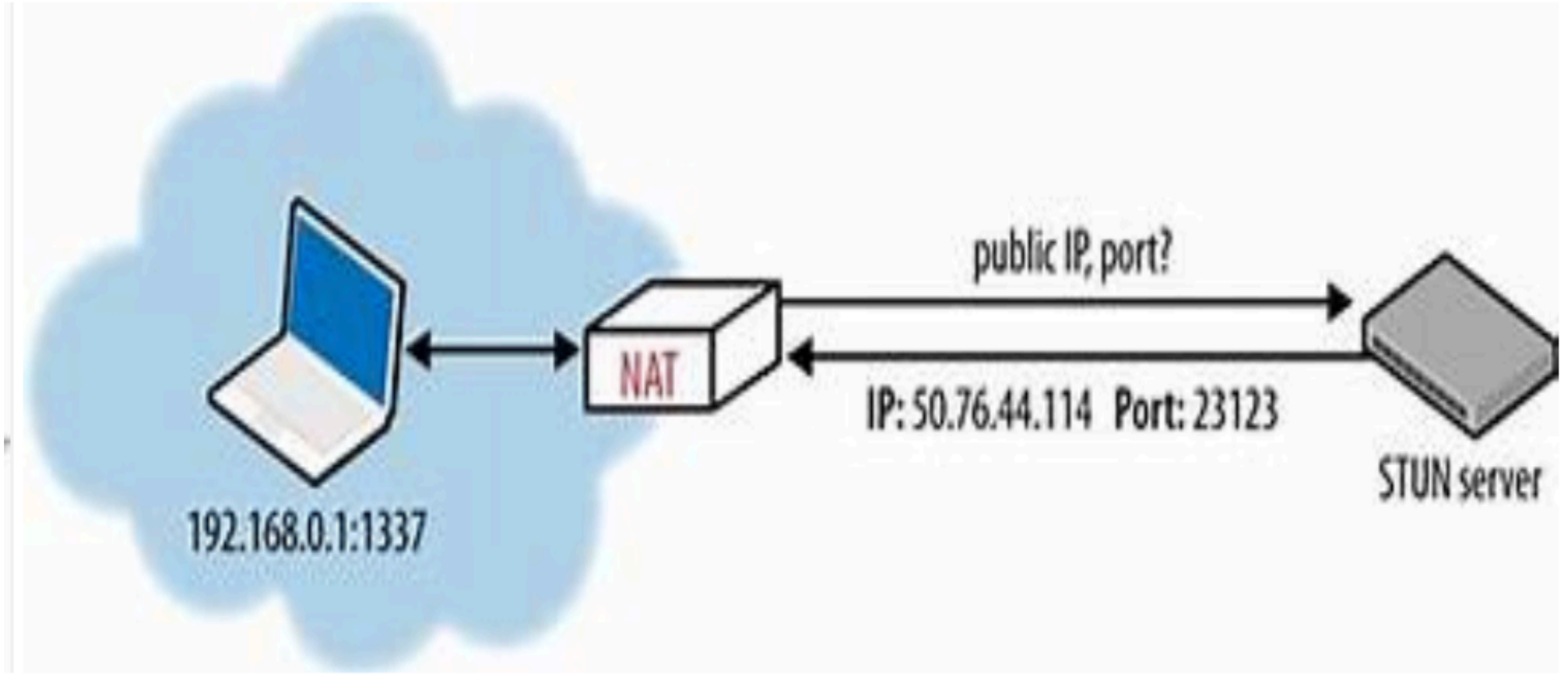


Step 2: map private IP & port to public IP & port
NAT Translation Table
Private IP Addr & Port
Public IP Addr & Port
192.168.100.3, 3855
145.12.131.7, 6282
……
……
Please fetch http://www.yahoo.com
192.168.100.3
192.168.100.4
192.168.100.5
Step 1
Source: 192.168.100.3, 3855
Dest: 209.131.36.158, 80
(www.yahoo.com)
Step 3
Source: 145.12.131.7, 6282
Dest: 209.131.36.158, 80
(www.yahoo.com)
To Yahoo
Router/NAT Device
Default Gateway
192.168.1.1
145.12.131.7
(Public IP Address)

Session Traversal Utilities for NAT (STUN)

- STUN server is to find out what your a public IP address is.
- WebRTC uses a STUN server to determine its public IP address, and the ICE framework which finds a suitable STUN server during connection establishment.
- STUN servers are free: https://gist.github.com/zziuni/3741933.
- STUN servers may not work in some cases due to network security or NAT device types.
- Then it relies on TURN server.

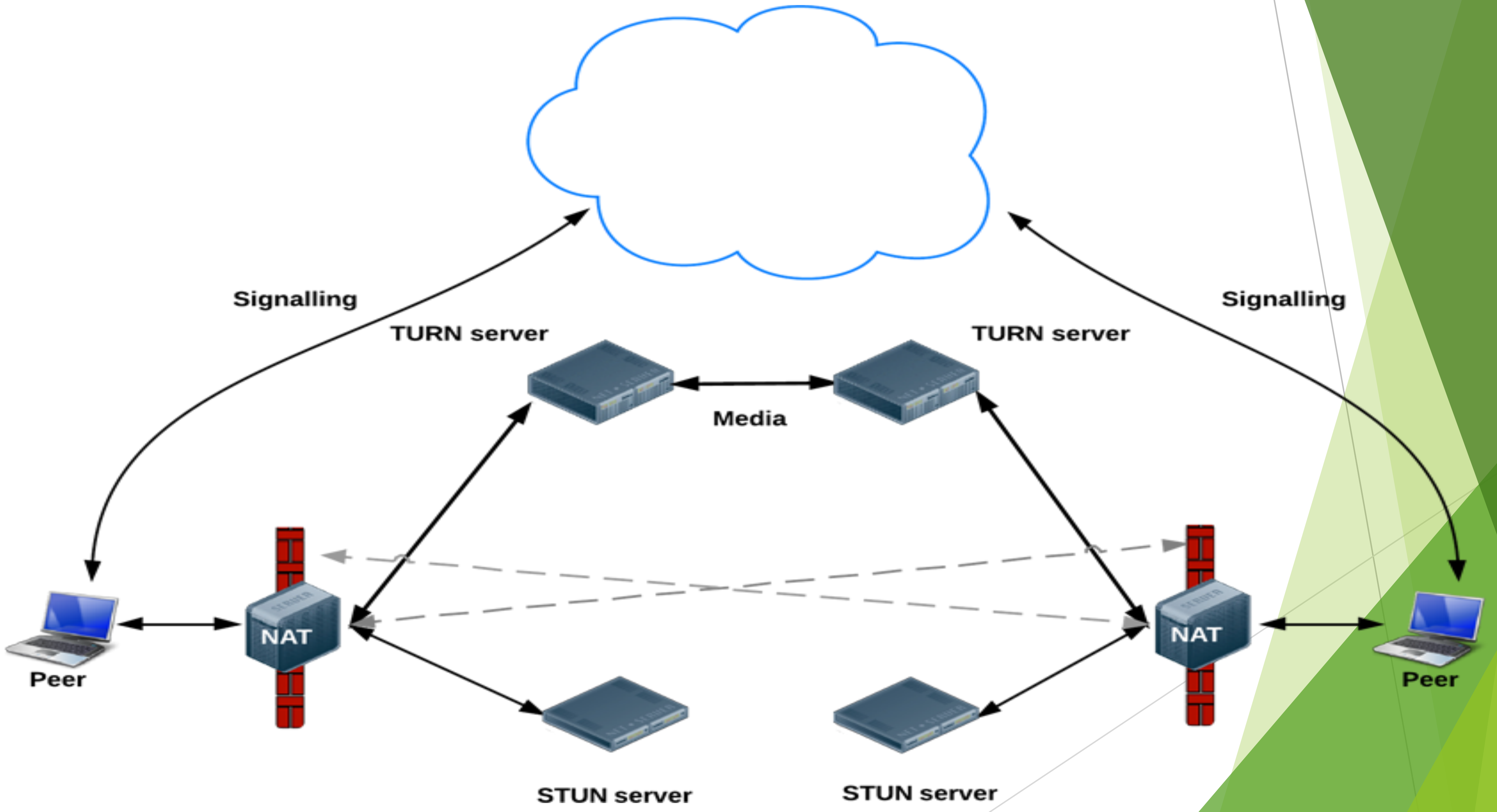
Example of STUN server



Traversal Using Relays around NAT (TURN) server

- Responsible for transmitting audio/video/data streaming.
- Most of the time, the STUN server is good enough,
- If it fails, a TURN server comes in to relay the media data.
- RTCPeerConnection.icecandidate() method establishes a connection between peers over STUN/TURN servers.
- It requires a high bandwidth, not free, and incurs cost.

An Example of STUN/TURN servers



Interactive Connectivity Establishment (ICE)

- Signaling server has been set up, then it uses ICE to get around with NATs and try to find the best option to connect peers.
- ICE tries to find the host address by querying its operating system.
- If this search does not work due to NAT device, then ICE relies on a STUN server to obtains its target external address.
- If this still fails, it resorts to a TURN server as a last solution.

WebRTC Security

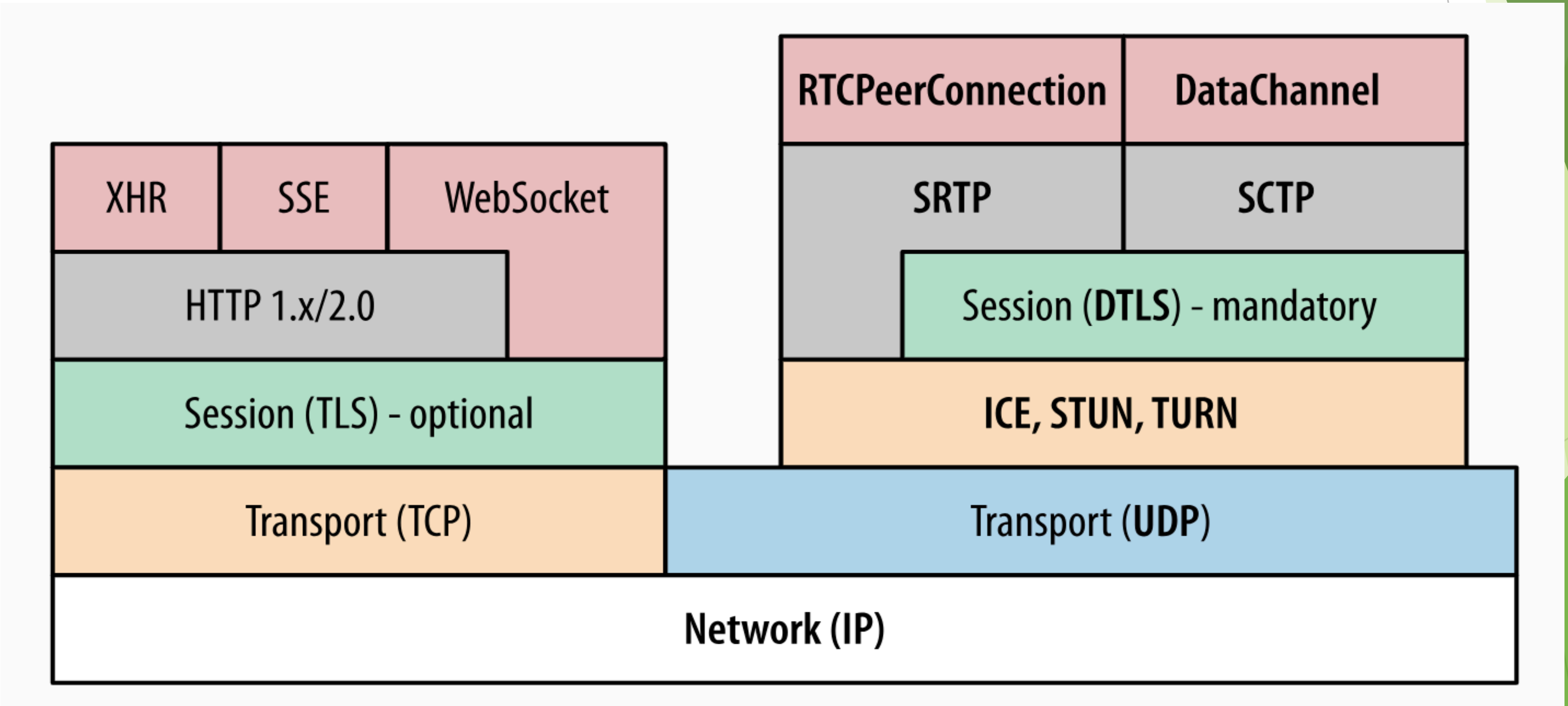
- There are many opportunities that media streams in transit could to leak .
- It can happen during peer-to-peer communication or peer-to-server communication, with a third party acting as a MiM.
- Encryption is a mandatory in WebRTC.
- The encryption technology must satisfy these requirements :
  - If messages are stolen in transit, they must not be readable.
  - Must utilize the highest bandwidth possible between the clients;
  - and the Datagram Transport Layer Security (DTLS) fits the bill.

Datagram Transport Layer Security(DTLS)

- A simple and easy-to-use protocol.
- It works with the UDP transport layer.
- It is modeled after the TLS protocol.

- Encryption protocols are based on datatype:
  - Data sent over RTCDataChannel is secured using DTLS.
  - Media streams are encrypted using the Secure Real-Time Transport Protocol (SRTP).

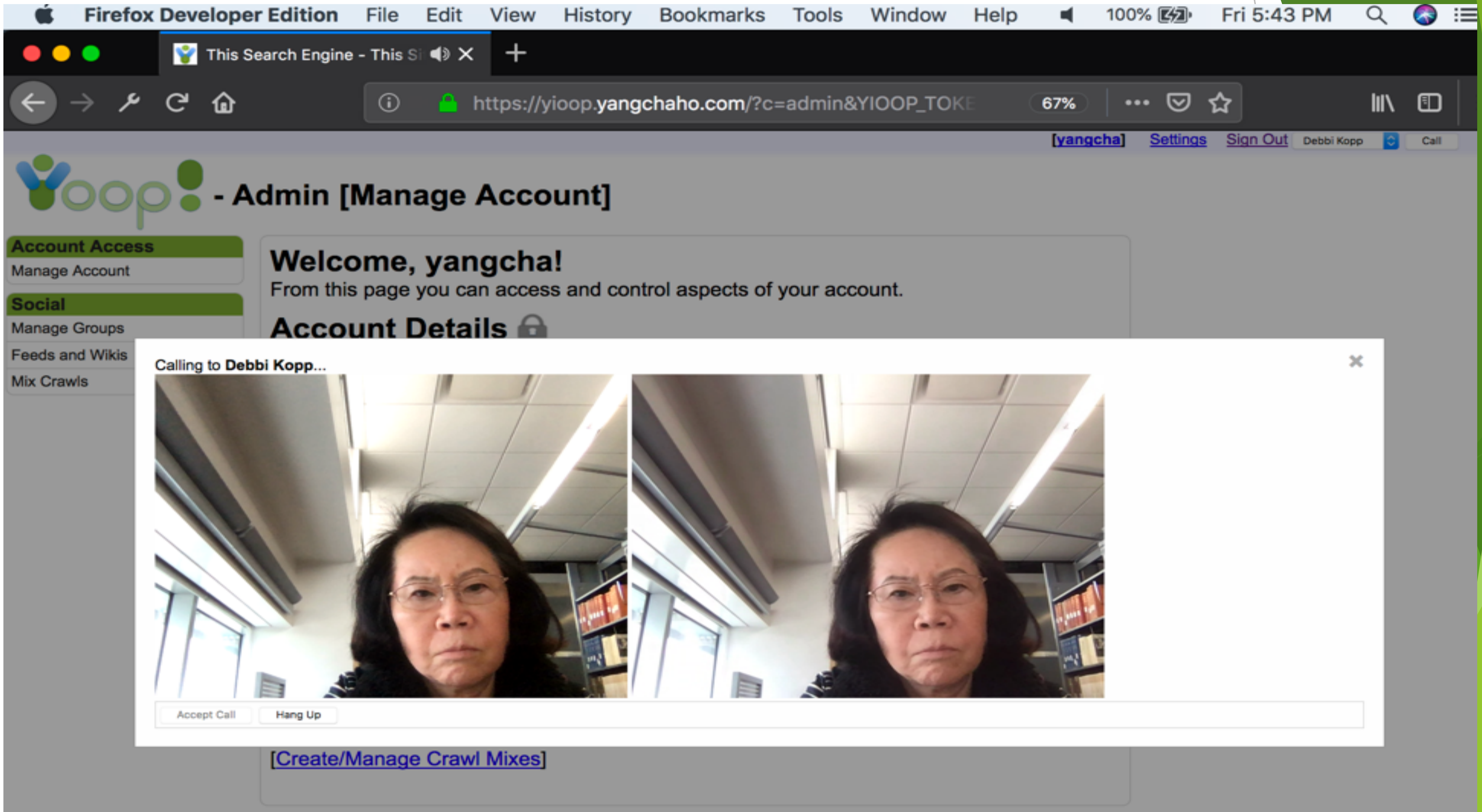
WebRTC protocol stack



RTCDataChannel

- Transfers data directly from one peer to another.
  It supports strings, binary types, Blob, and ArrayBuffer.
- Resembles to the WebSocket API, users use the same programming model.
- This paper does not use the RTCDataChannel, so discussion on RTCDataChannel is limited to this much.

WebRTC video Chat inside Yioop.com



WebRTC video Chat inside Yioop.com - cont.

- When a user logs on to the Yioop, which opens the WebSocket connection to the signaling server.

- Once the connection has been made, the signaling server sends Yioop a list of all users who is are currently online and keeps the list until the web page closes.

- When user logs off, the signaling server knows that the user becomes offline.

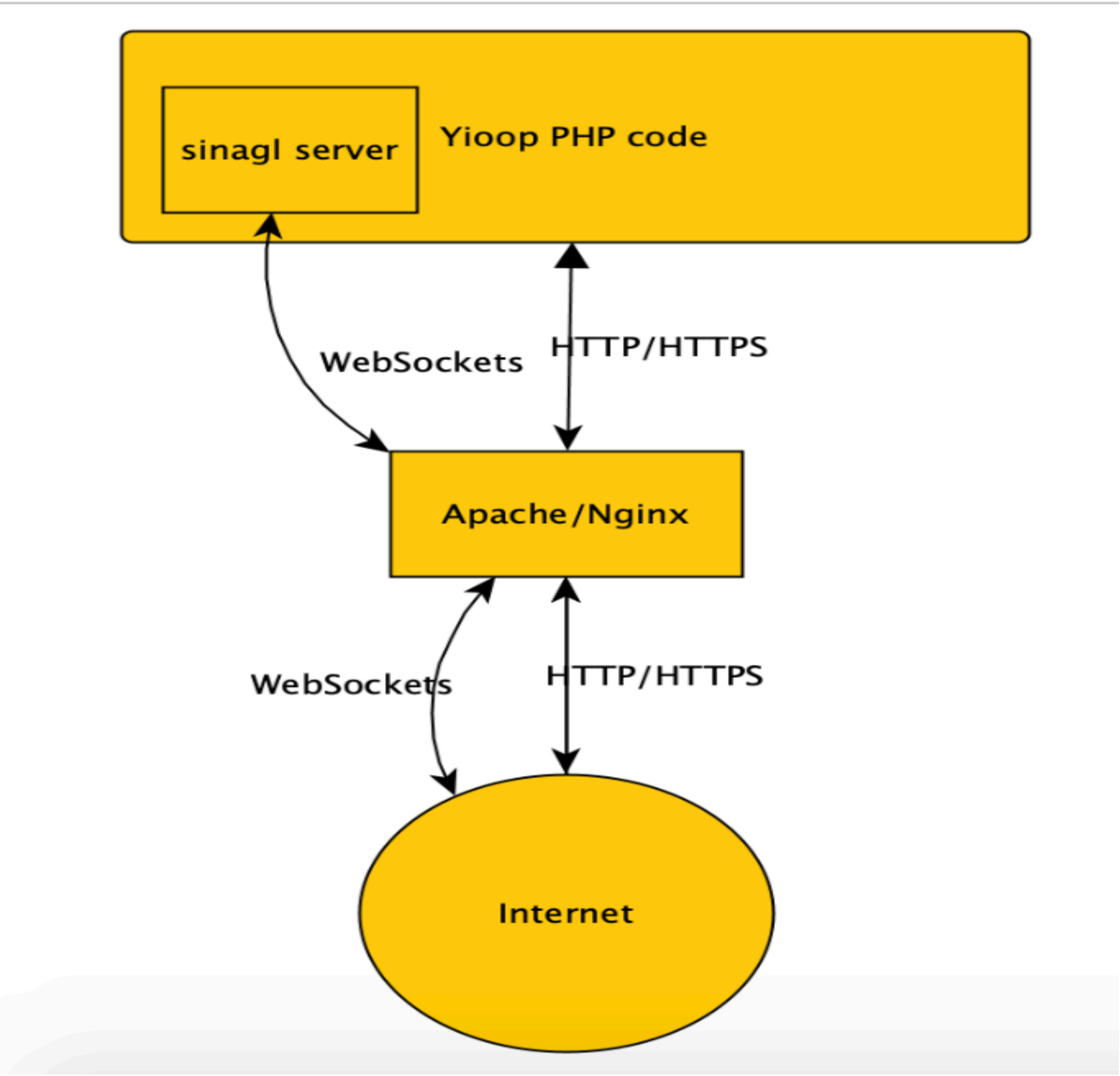
WebRTC video Chat inside Yioop.com - cont.

- Bob logs on to Yioop, and selects Alice from the drop down list provided that Alice has already logged in.
- Both Bob and Alice are connected to the signaling server via the WebSocket protocol, and the signaling server sends messages to each one.
- When Bob selects Alice, Yioop sends a message to the signaling server and informs that Bob wants to call Alice.
- At the same time, the signaling server sends the message to Bob.

WebRTC video Chat inside Yioop.com - cont.

- Yioop shows the green circle, indicating another user is calling him; callers exchange WebRTC data and establish the call.
- We put a WebSocket server into the signaling server and WebSocket client part into the Yioop page.
- This application is written in PHP, runs on the server, and listens to WebSocket connections on the TCP port 2002.

A snap shot of relationship with Yioop and WebSockets



Conclusion

- WebRTC inside Yioop.com can be used in lieu of your cell phone as long as you log on Yioop.com.

- This technology can be utilized for online class.

- We could extend this feature to make a conference call as a next step.