WebRTC

Web Real Time Communication

20 Aug 2016
Đinh Quang Trung
Geeky Weekend

SILICON STRAITS SAIGON

Google Developers Experts

Đinh Quang Trung
Web Engineer

fb.com/trungdq88

@trungdq88

github.com/trungdq88

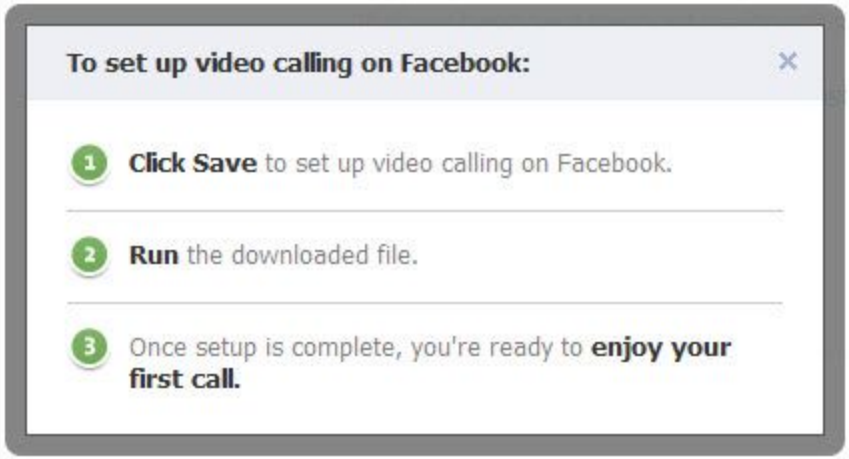
- 12/1993 - Born
- 12/2009 - Wrote very first lines of code
- 01/2013 - Started learning **Web Development**
- 09/2015 - Graduated **FPT University**
- 12/2015 - Web Engineer at **Silicon Straits Saigon**
- 04/2016 - Web-tech Expert from **Google Developer Experts Program**

Overview

1. What is WebRTC and why WebRTC
2. Setup a WebRTC connection
    a. getUserMedia()
    b. RTCPeerConnection
    c. RTCDataChannel
3. WebRTC in action
    a. Signaling
    b. STUN/TURN servers
    c. Architecture
4. Start your WebRTC app

WebRTC

Bringing realtime communication to the web


To set up video calling on Facebook:
1 Click Save to set up video calling on Facebook.
2 Run the downloaded file.
3 Once setup is complete, you're ready to enjoy your first call.

WebRTC

Bringing realtime communication to the web

WebRTC

Bringing realtime communication to the web

- Enable peer-to-peer connection for faster connection
- It’s fast enough that you can stream video/audio with low latency
- You can transfer data too

Three main APIs

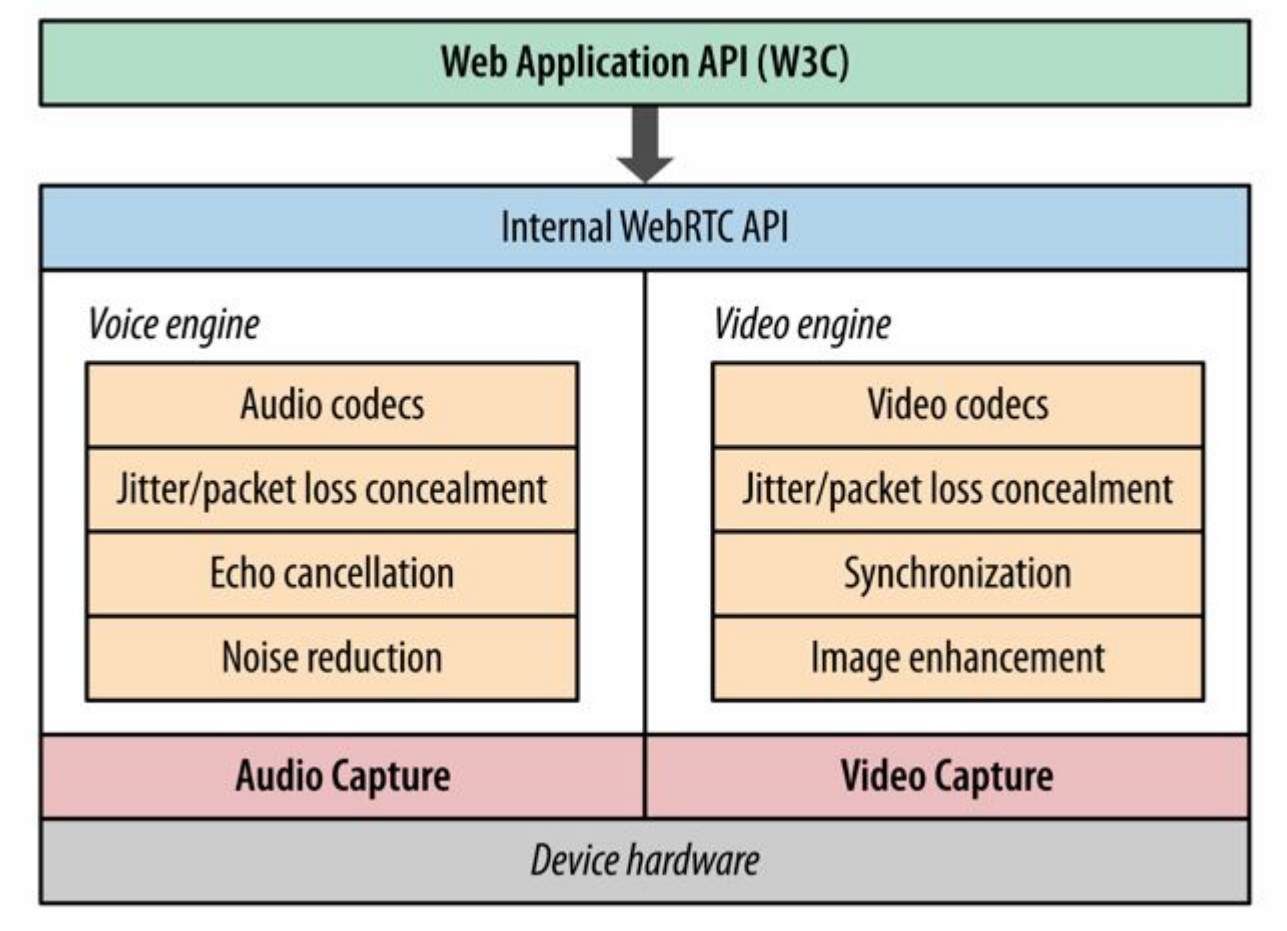
1. getUserMedia - Acquiring video and audio
2. RTCPeerConnection - Transfering video and audio peer to peer
3. RTCDataChannel - Transfering data peer to peer

getUserMedia

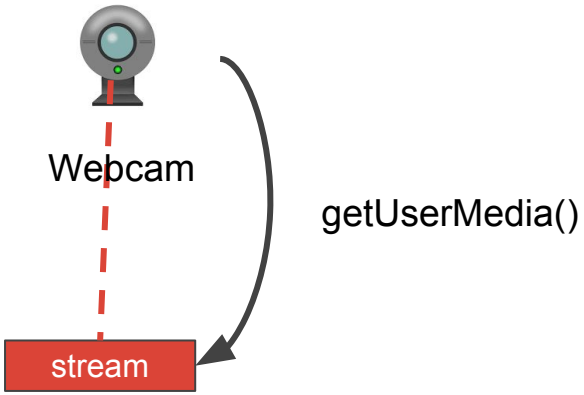
Acquiring video and audio

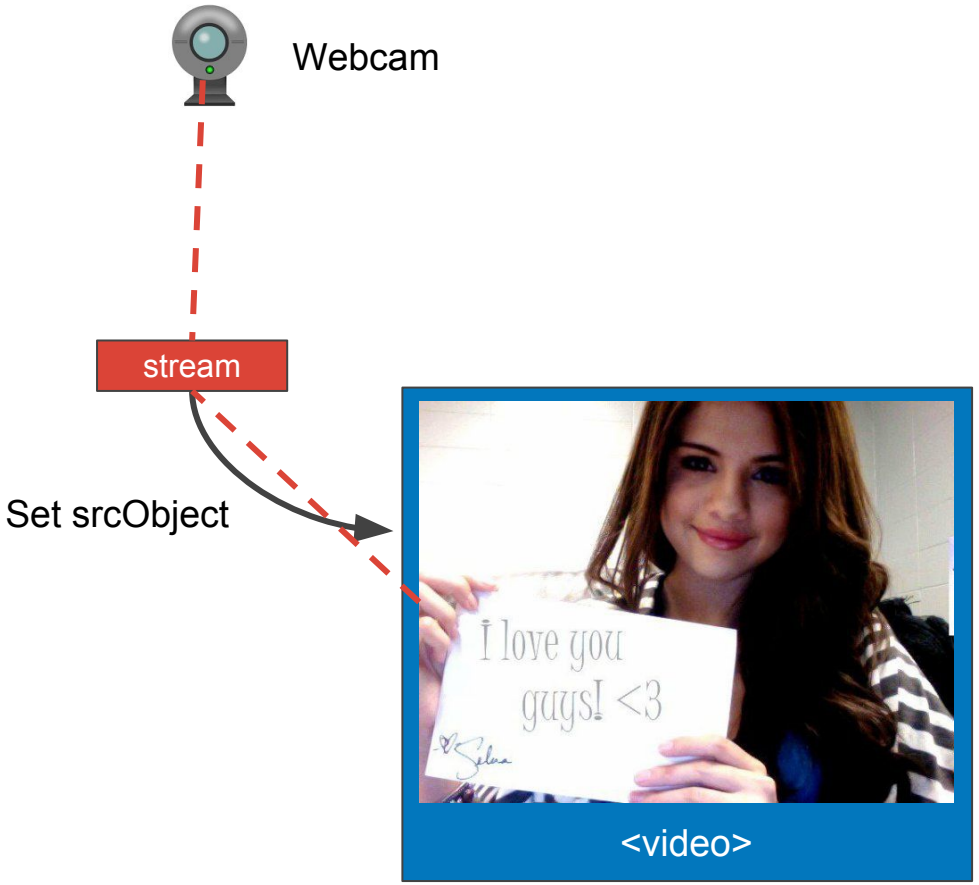
MediaStream

```
navigator.getUserMedia()
```



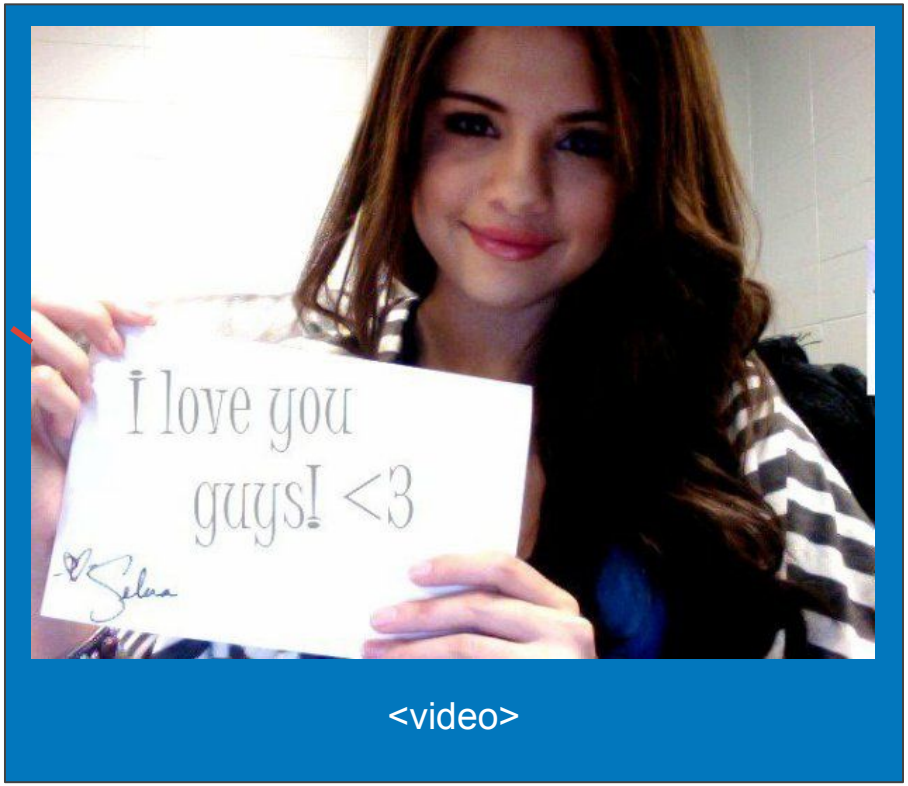
Webcam


Webcam
getUserMedia()
stream


Webcam
stream
Set srcObject
I love you guys! <3
<video>

Webcam

stream



https://jsfiddle.net/trungdq88/2c9g4L0r/

JAVASCRIPT

```javascript
var constraints = {
  audio: false,
  video: true
};
var video = document.querySelector('video');

// Show to <video> tag
function successCallback(stream) {
    video.srcObject = stream;
}

// User might not have a camera?
function errorCallback(error) {
  console.log('navigator.getUserMedia error: ', error);
}

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

Stream object

Vendor prefix

https://idevelop.ro/ascii-camera/

https://webcamtoy.com/

Screen capture

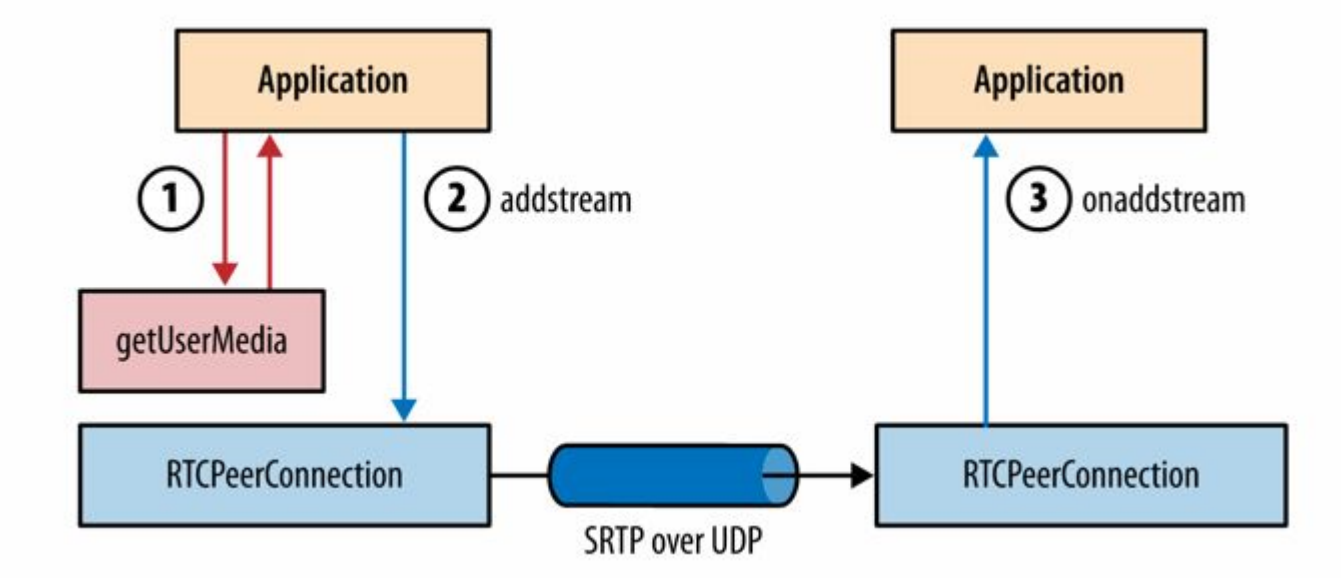
```
4
5  var constraint = {
6    video: {
7      mediaSource: "screen"
8    },
9  };
10
```

https://mozilla.github.io/webrtc-landing/gum_test.html
(See in Firefox)

RTCPeerConnection

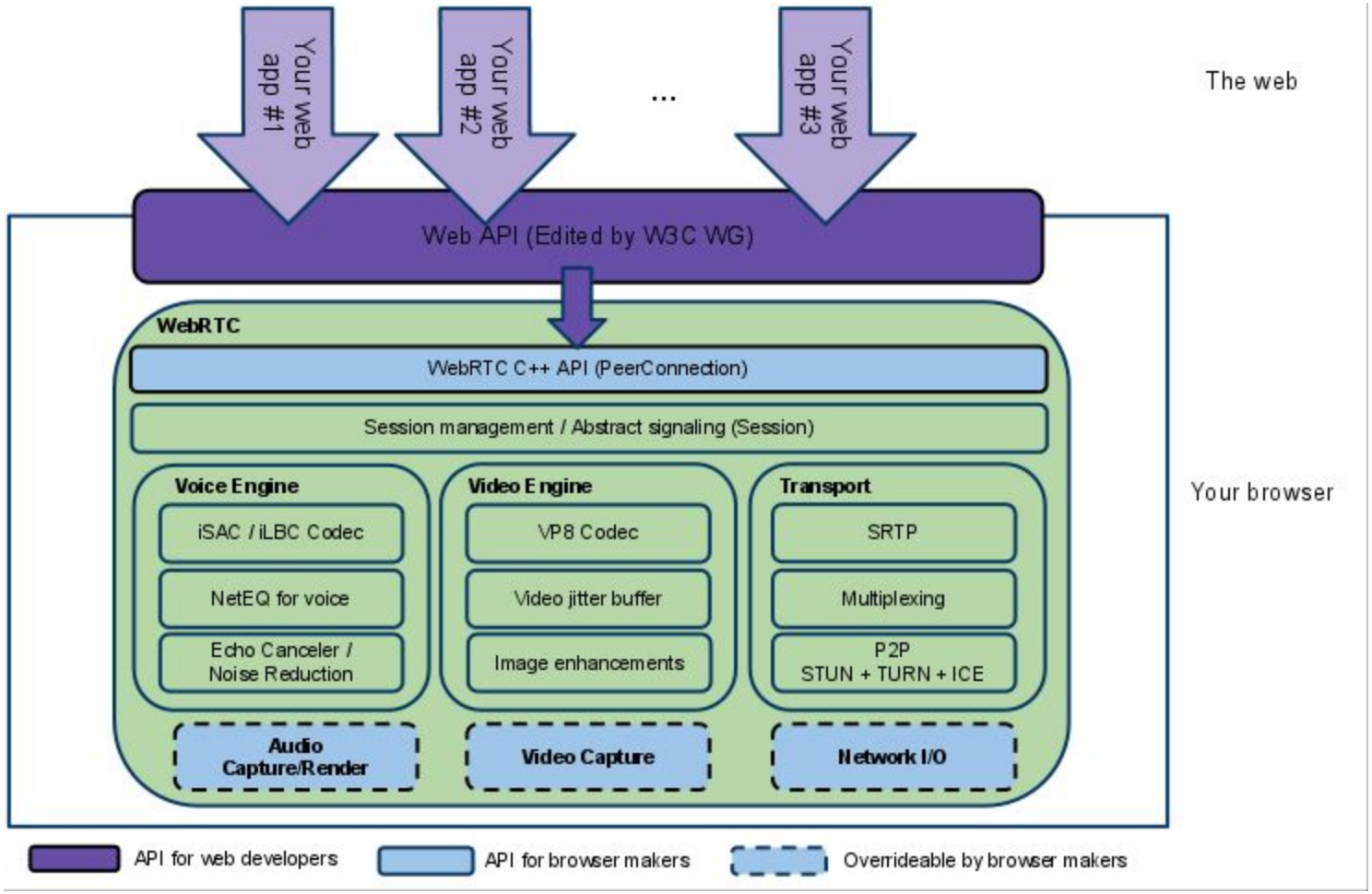
Transfering video and audio peer to peer


I love you
guys! <3
I love you
guys! <3


Application
Application
1
2
addstream
3
onaddstream
getUserMedia
RTCPeerConnection
RTCPeerConnection
SRTP over UDP

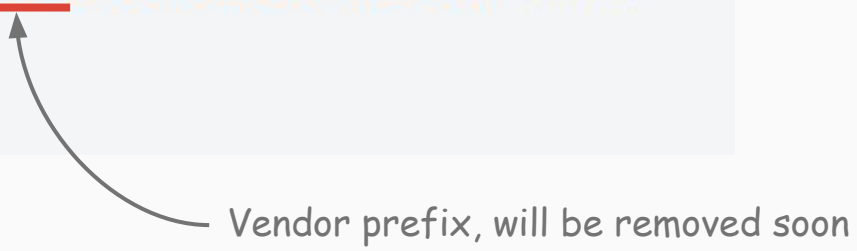
RTCPeerConnection does a lot

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


Your web app #1
Your web app #2
...
Your web app #3
The web
Web API (Edited by W3C WG)
WebRTC
WebRTC C++ API (PeerConnection)
Session management / Abstract signaling (Session)
Voice Engine
iSAC / iLBC Codec
NetEQ for voice
Echo Canceler / Noise Reduction
Video Engine
VP8 Codec
Video jitter buffer
Image enhancements
Transport
SRTP
Multiplexing
P2P STUN + TURN + ICE
Audio Capture/Render
Video Capture
Network I/O
Your browser
API for web developers
API for browser makers
Overrideable by browser makers

Create new RTCPeerConnection instance

```
3
4
5  var pc = new webkitRTCPeerConnection(null);
```



Set up a RTCPeerConnection

Set up a RTCPeerConnection

Prepare yourself, this is not as simple as sending an ajax request or web socket...

How RTCPeerConnection works

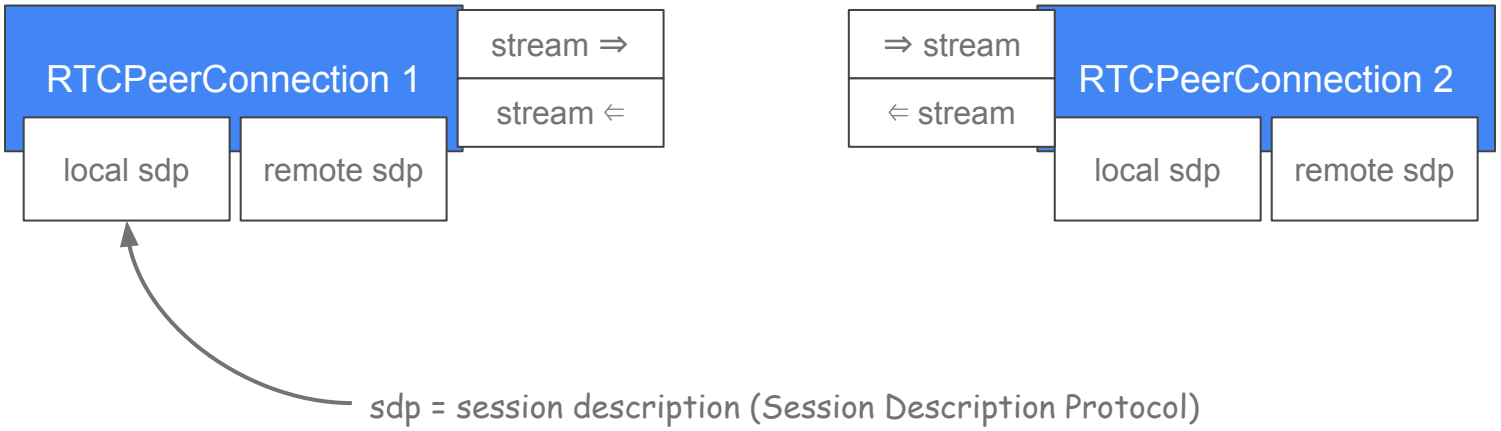
Webcam

RTCPeerConnection 1

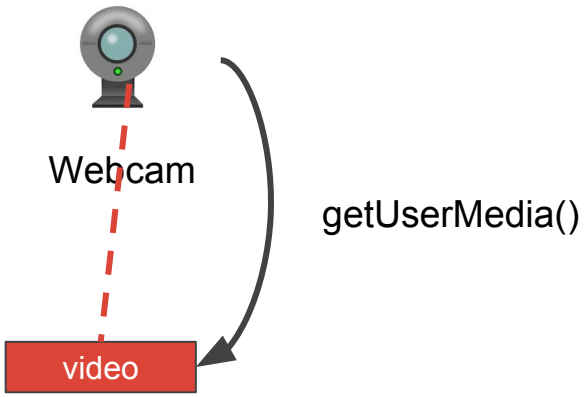
RTCPeerConnection 2

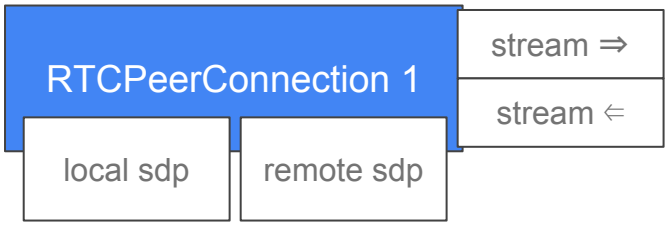
How RTCPeerConnection works

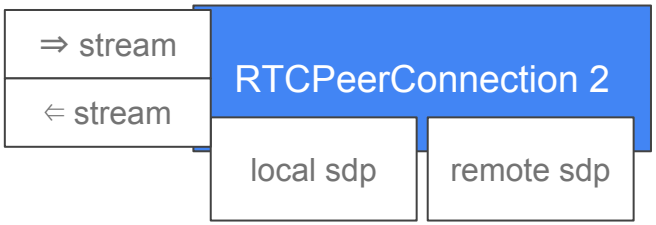
Webcam



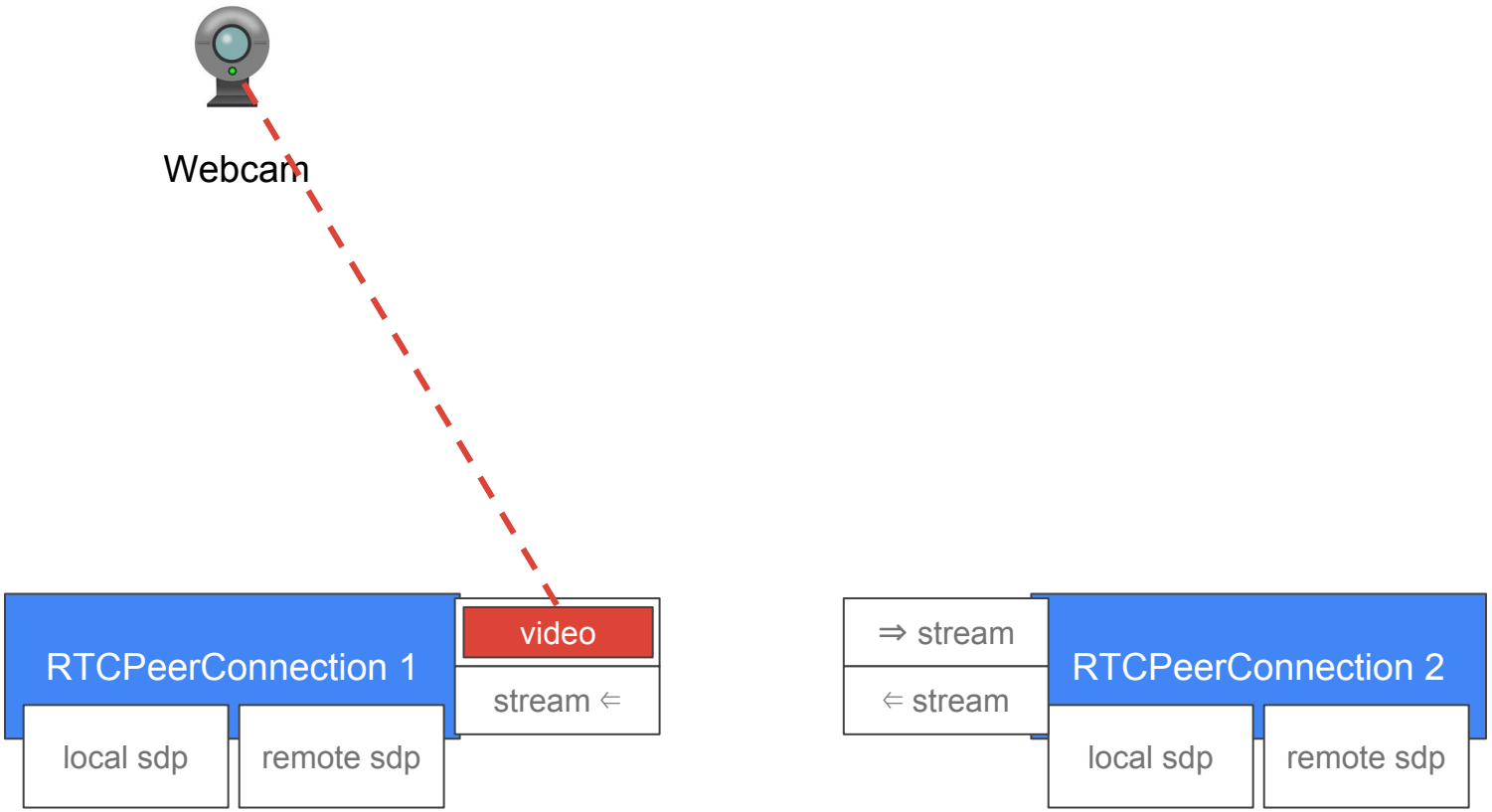
How RTCPeerConnection works



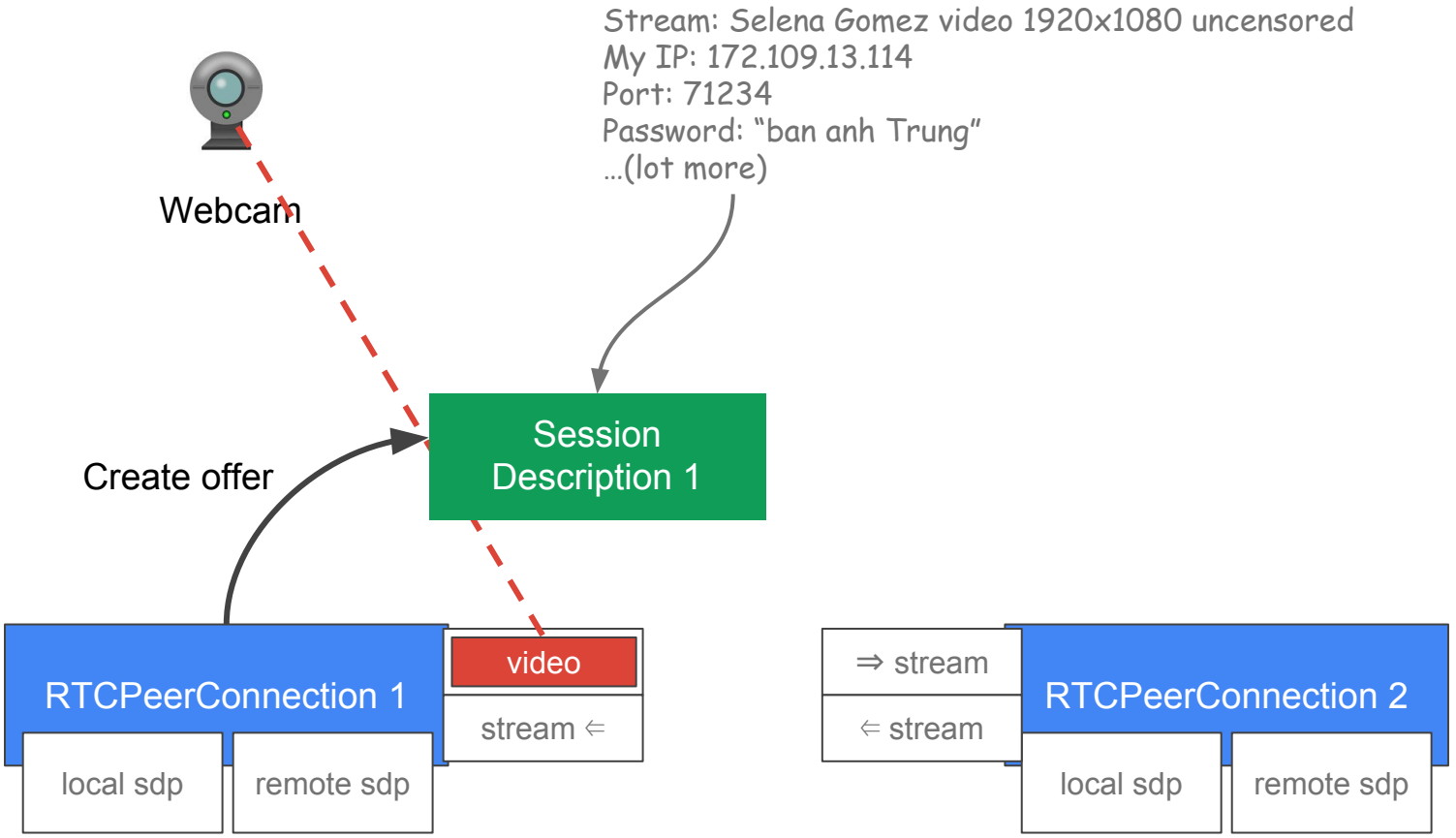




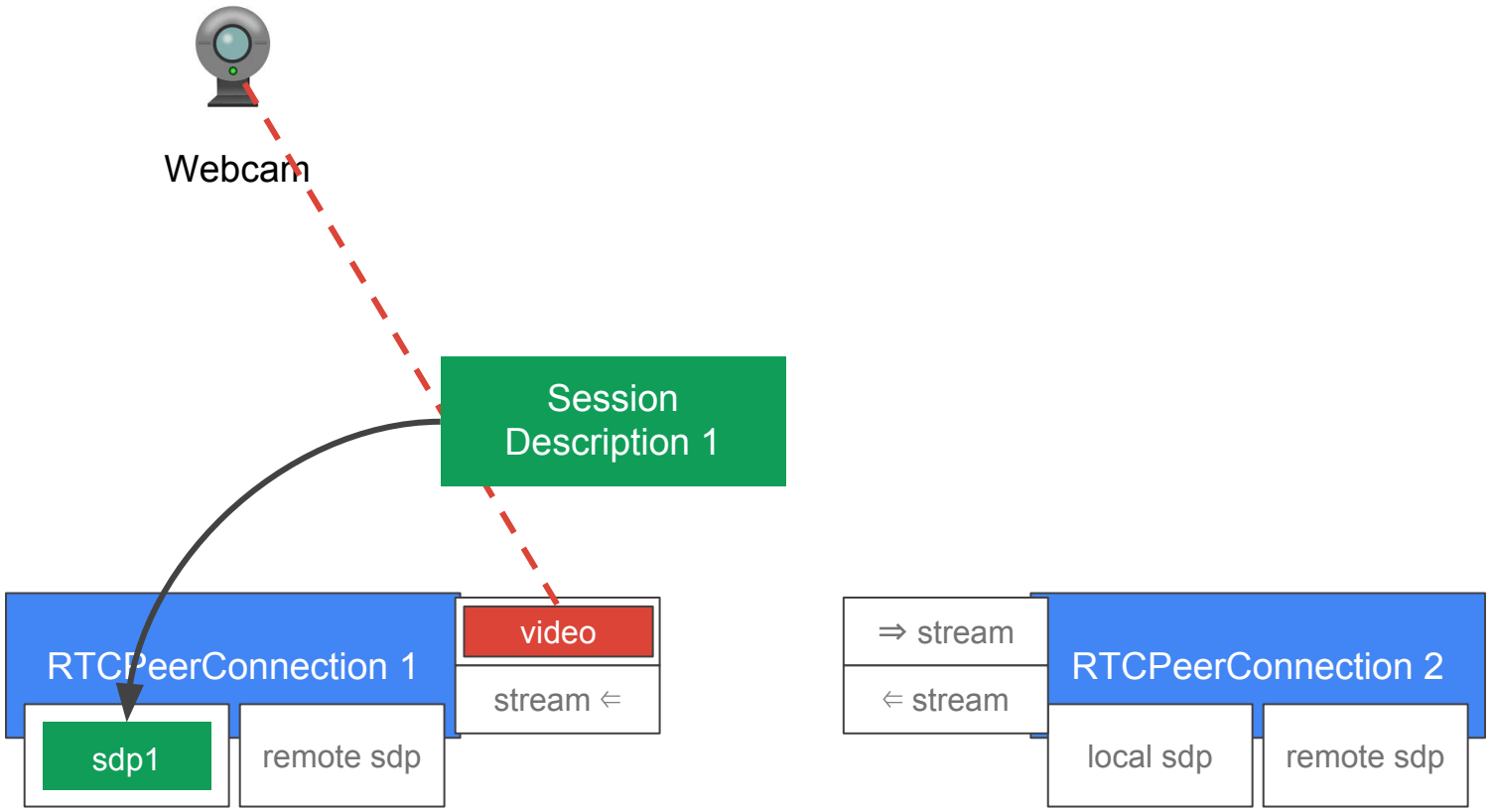
How RTCPeerConnection works



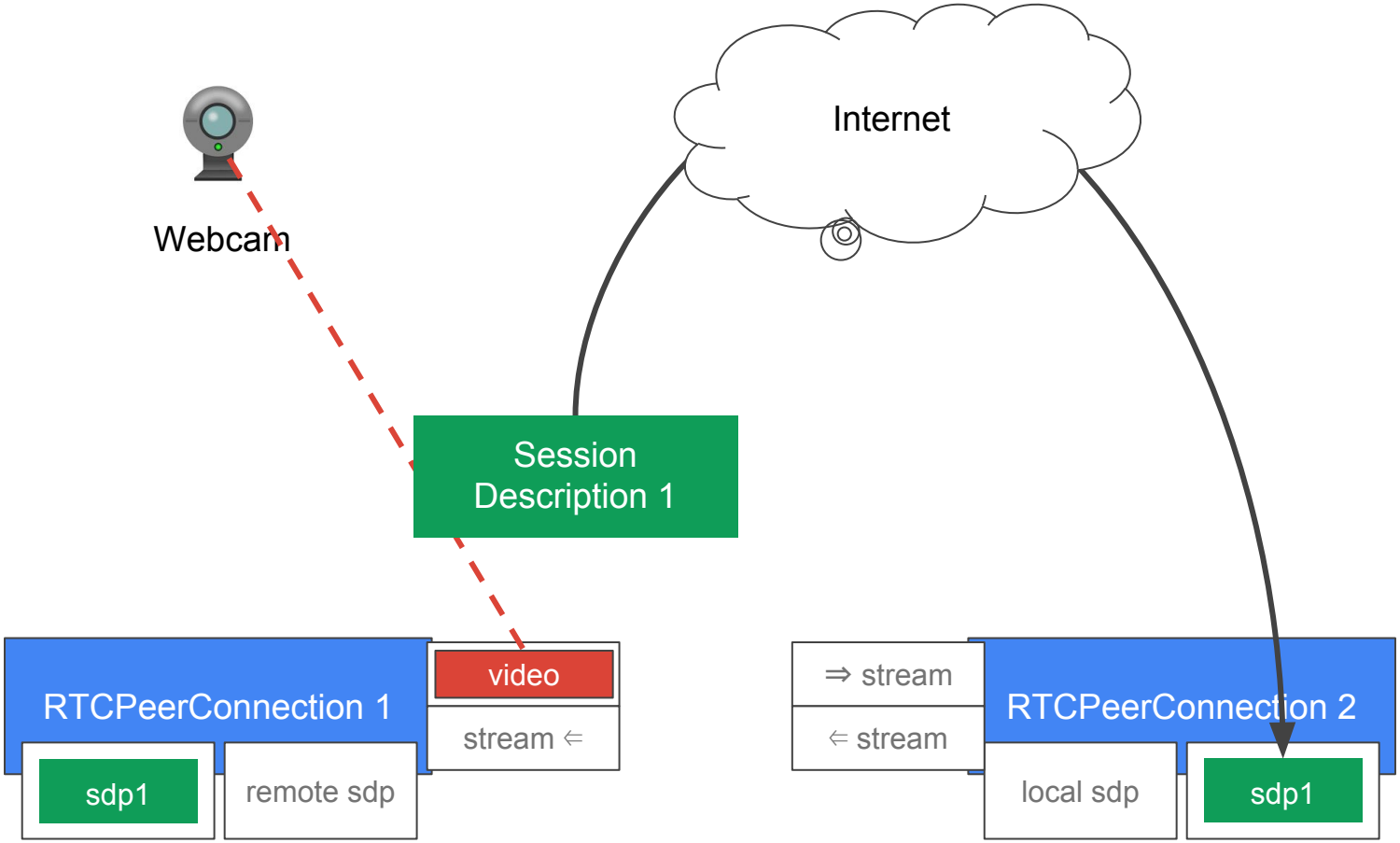
How RTCPeerConnection works



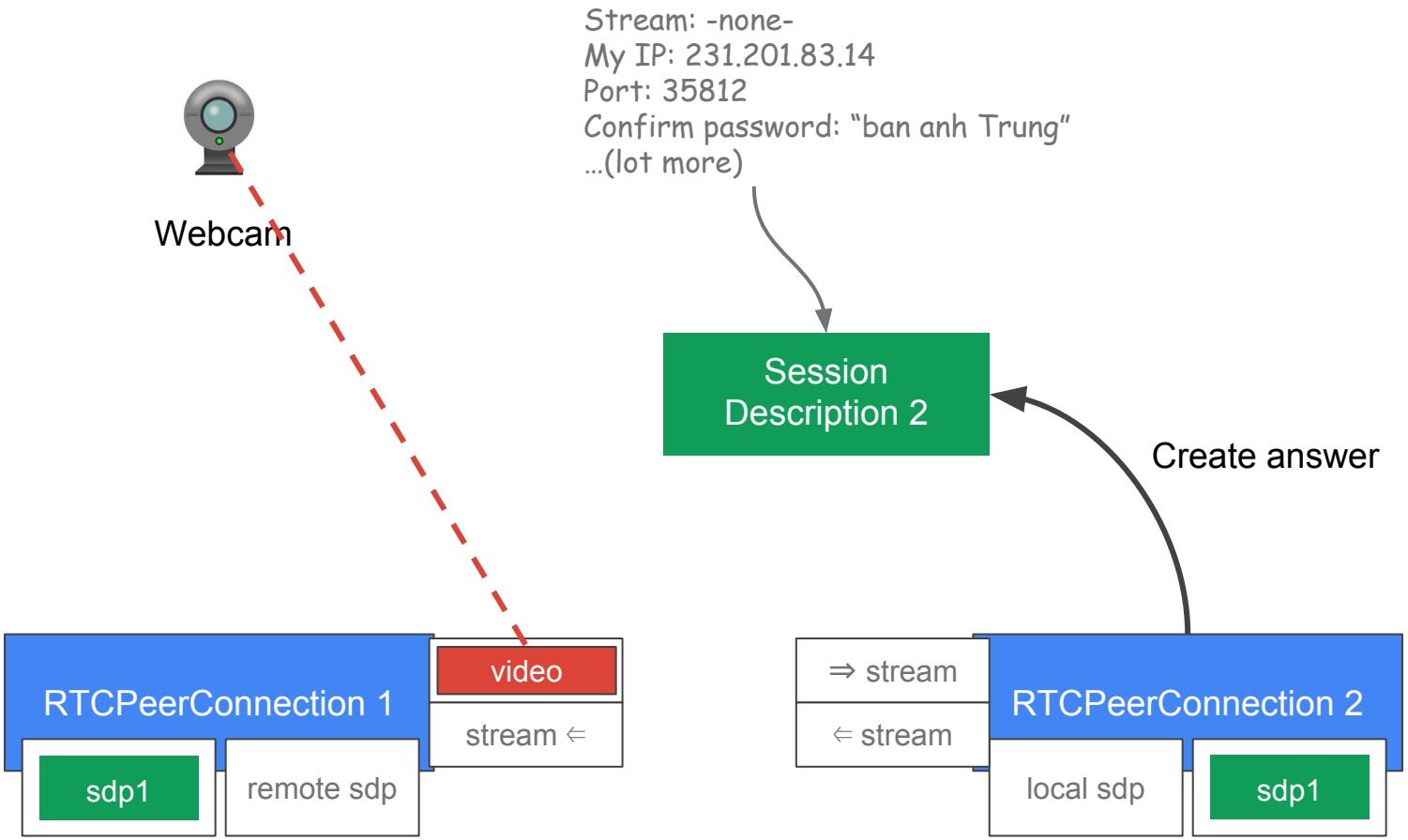
How RTCPeerConnection works



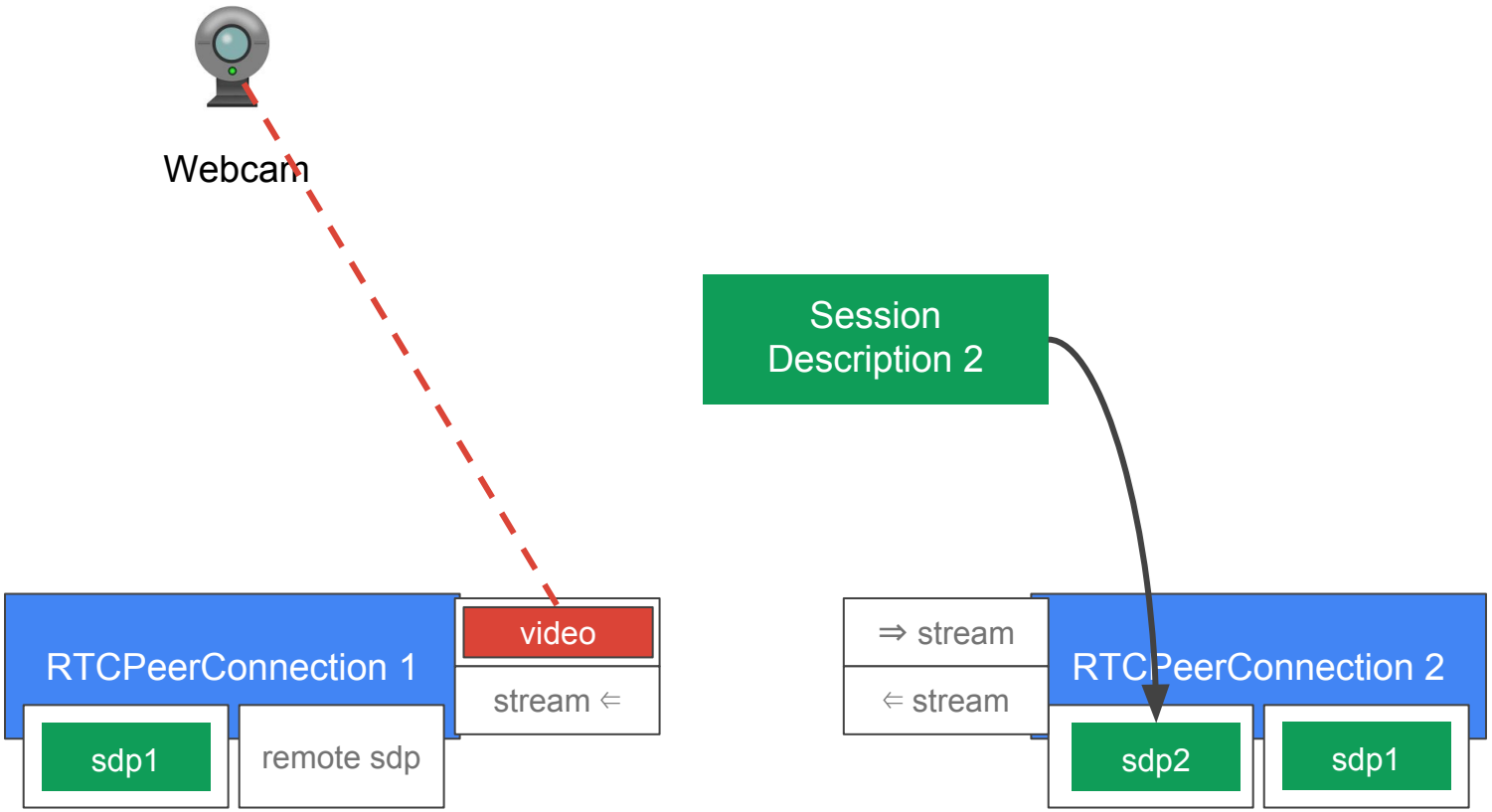
How RTCPeerConnection works



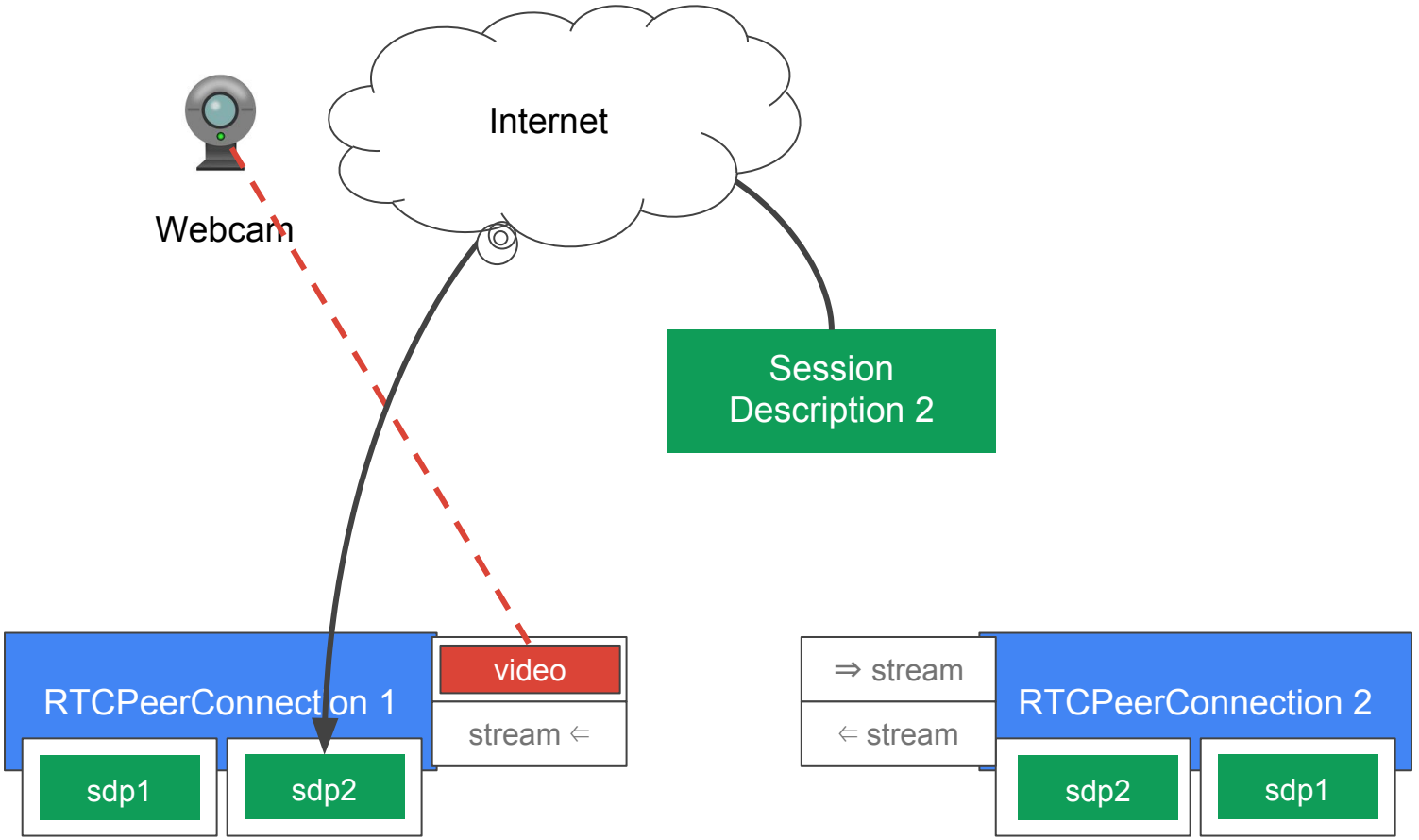
How RTCPeerConnection works



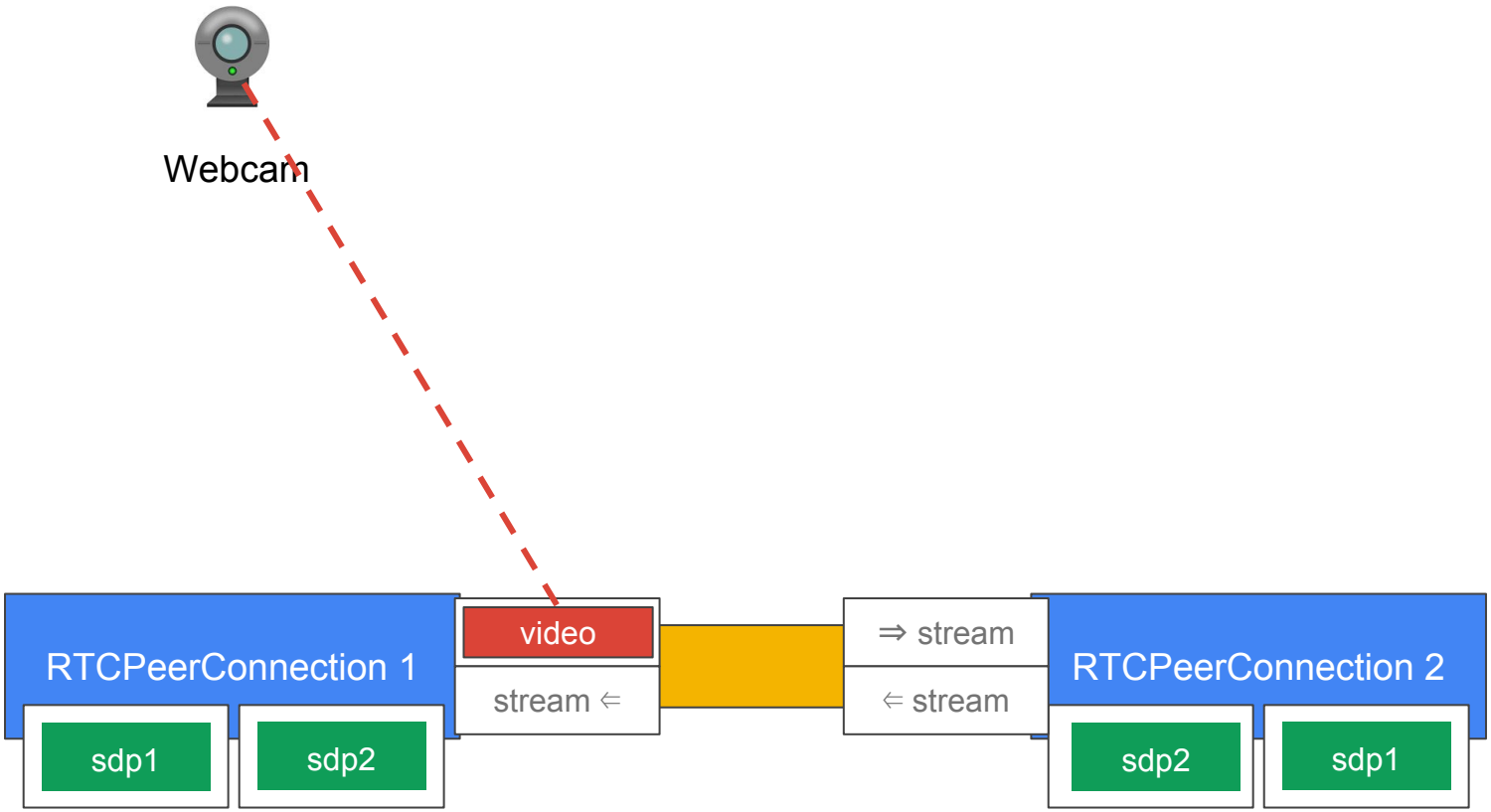
How RTCPeerConnection works



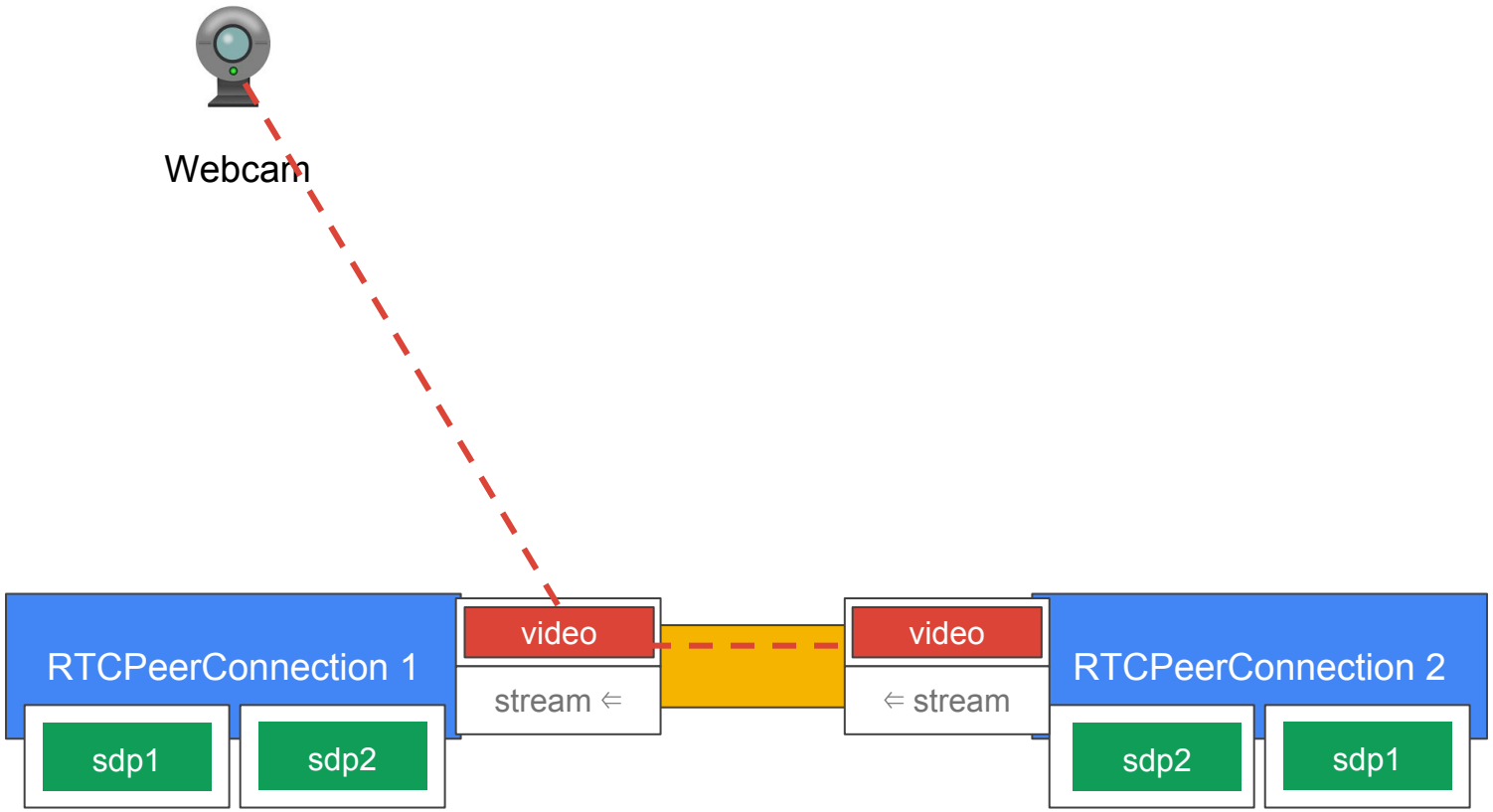
How RTCPeerConnection works



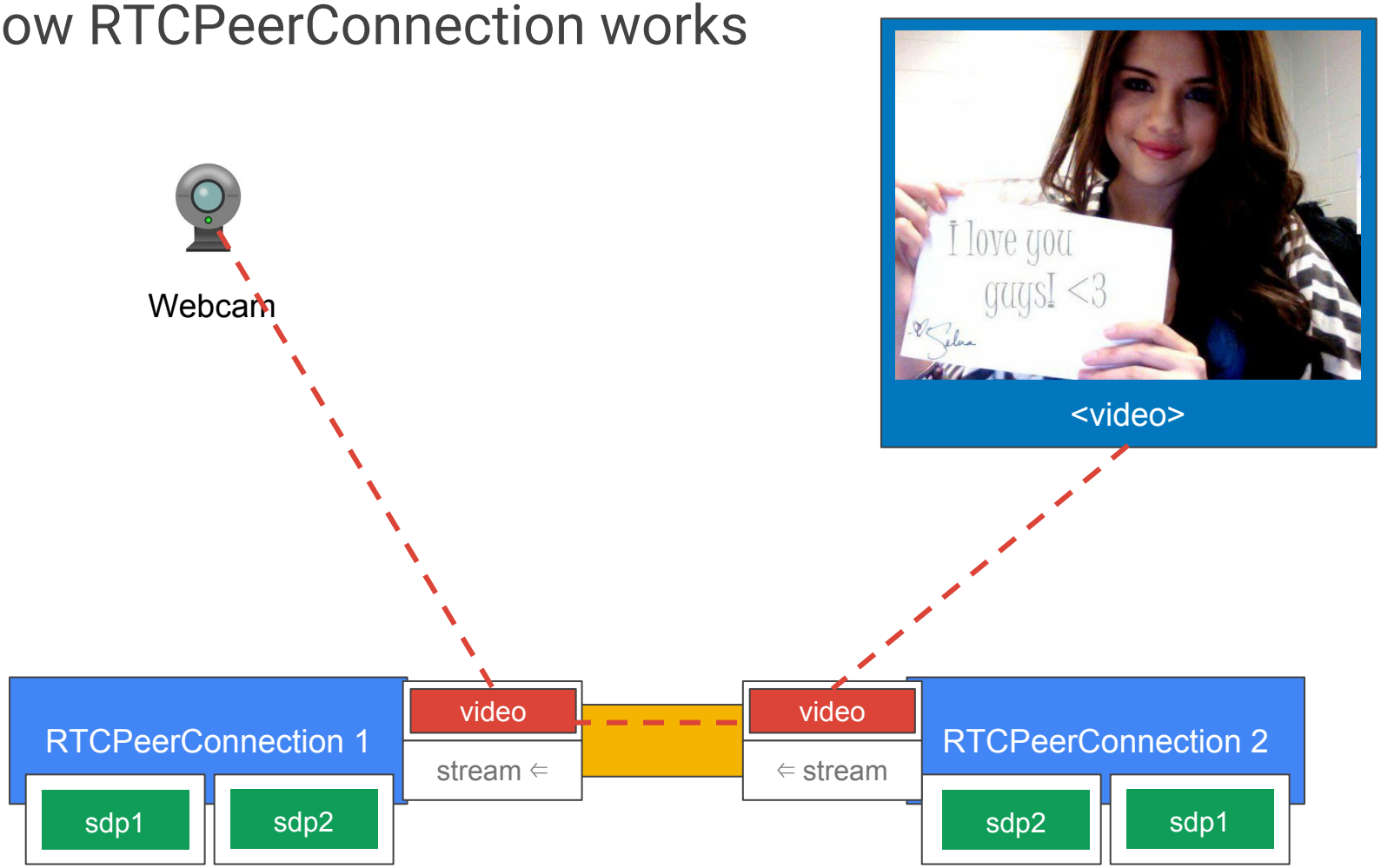
How RTCPeerConnection works



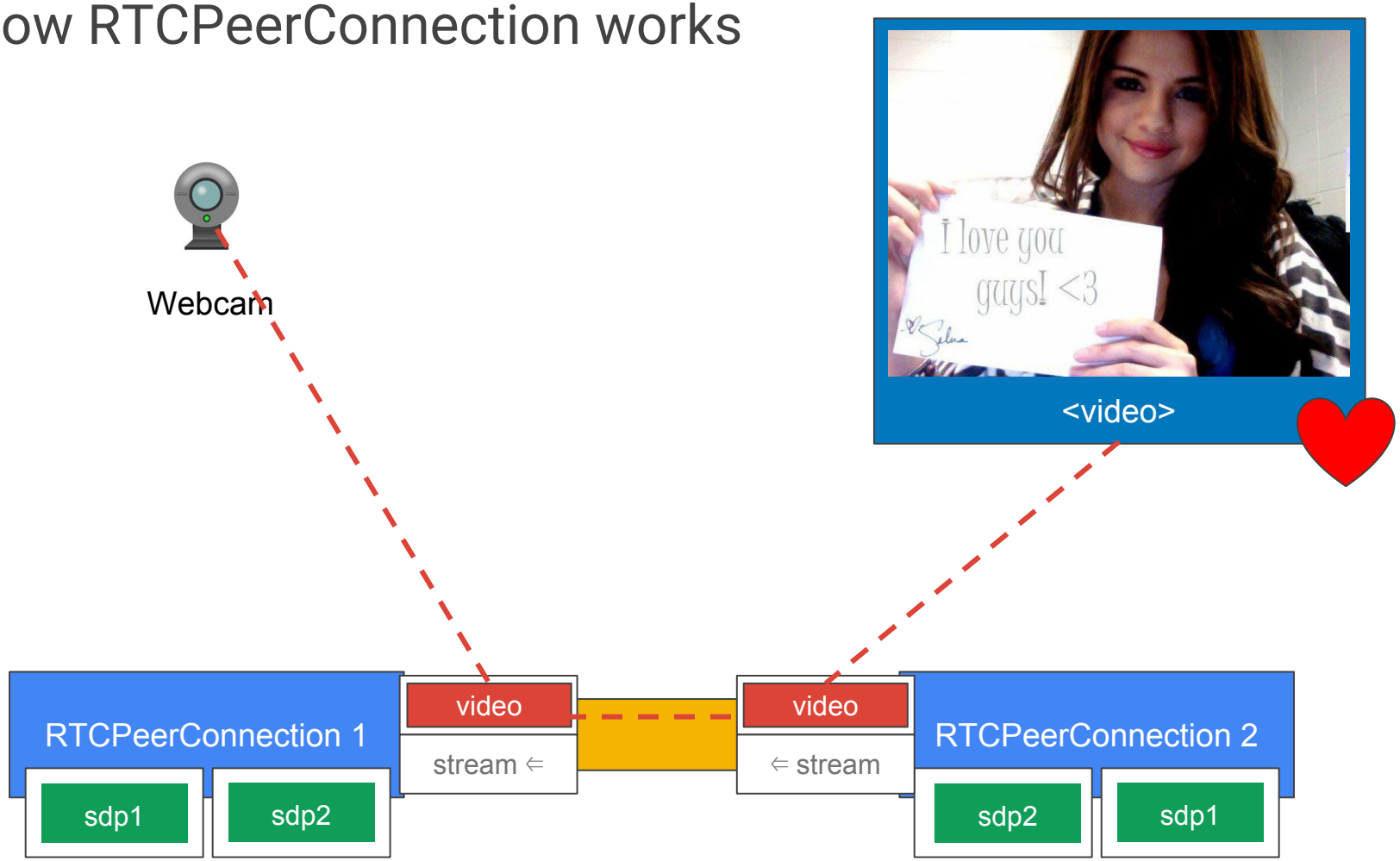
How RTCPeerConnection works



How RTCPeerConnection works



How RTCPeerConnection works

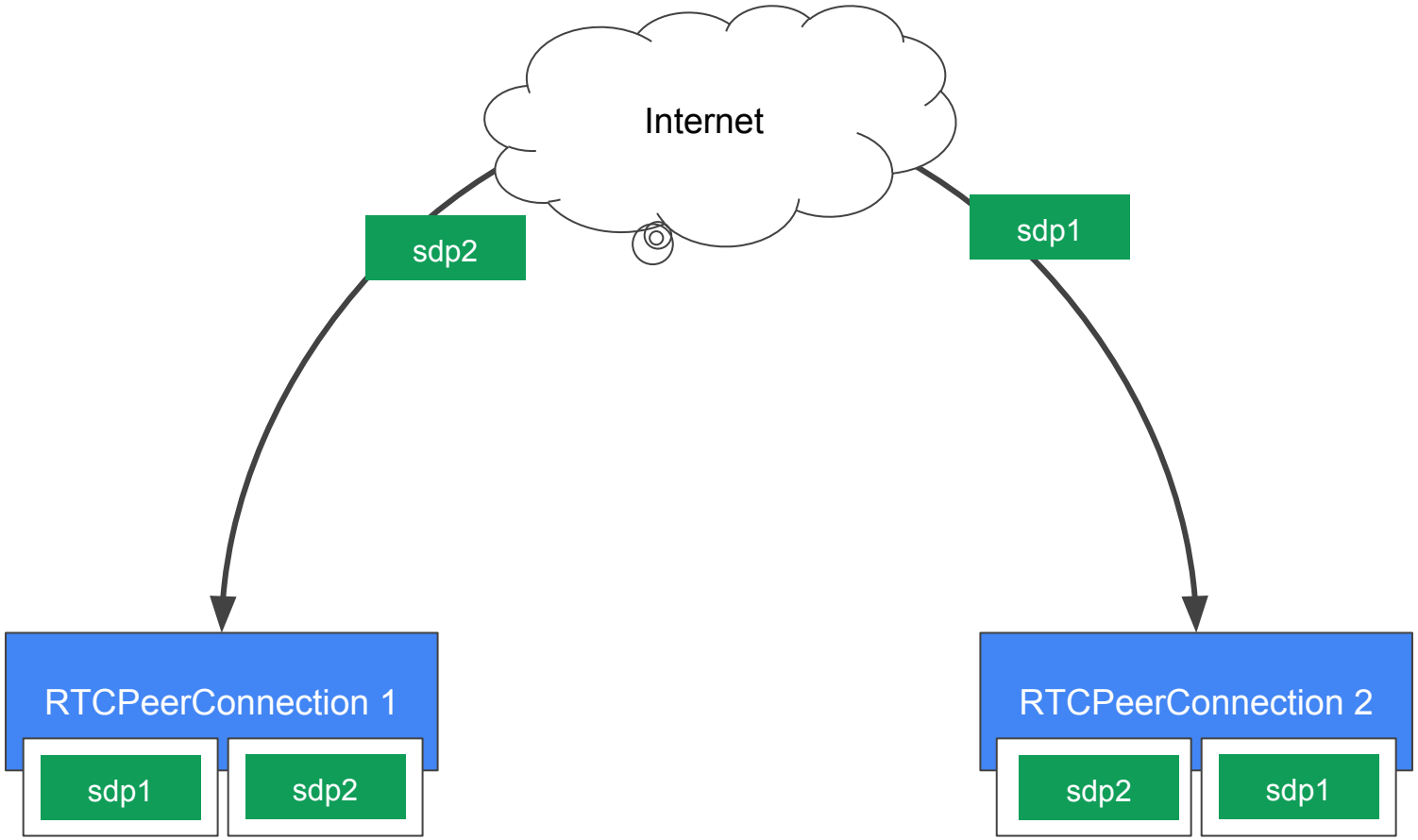


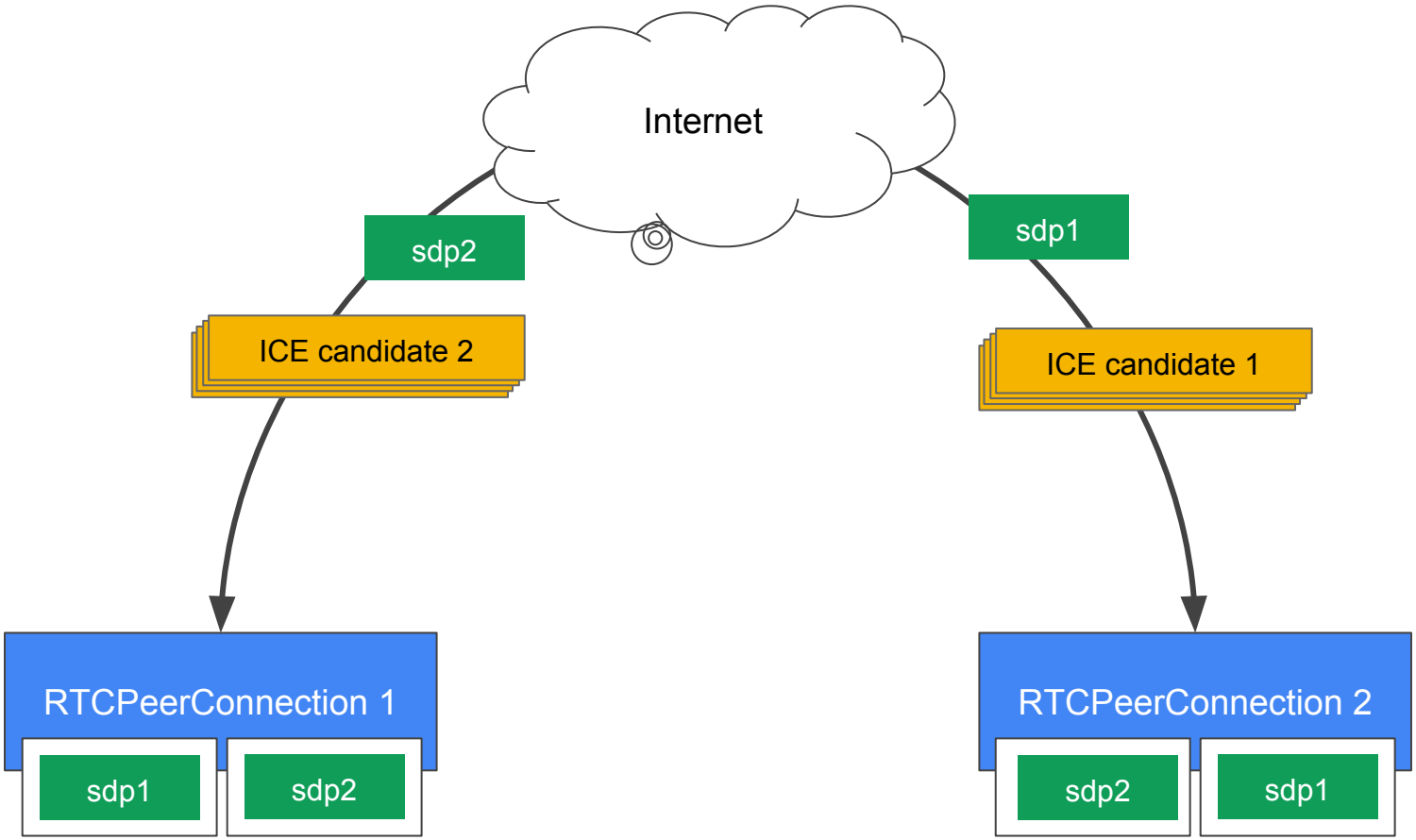
How RTCPeerConnection works

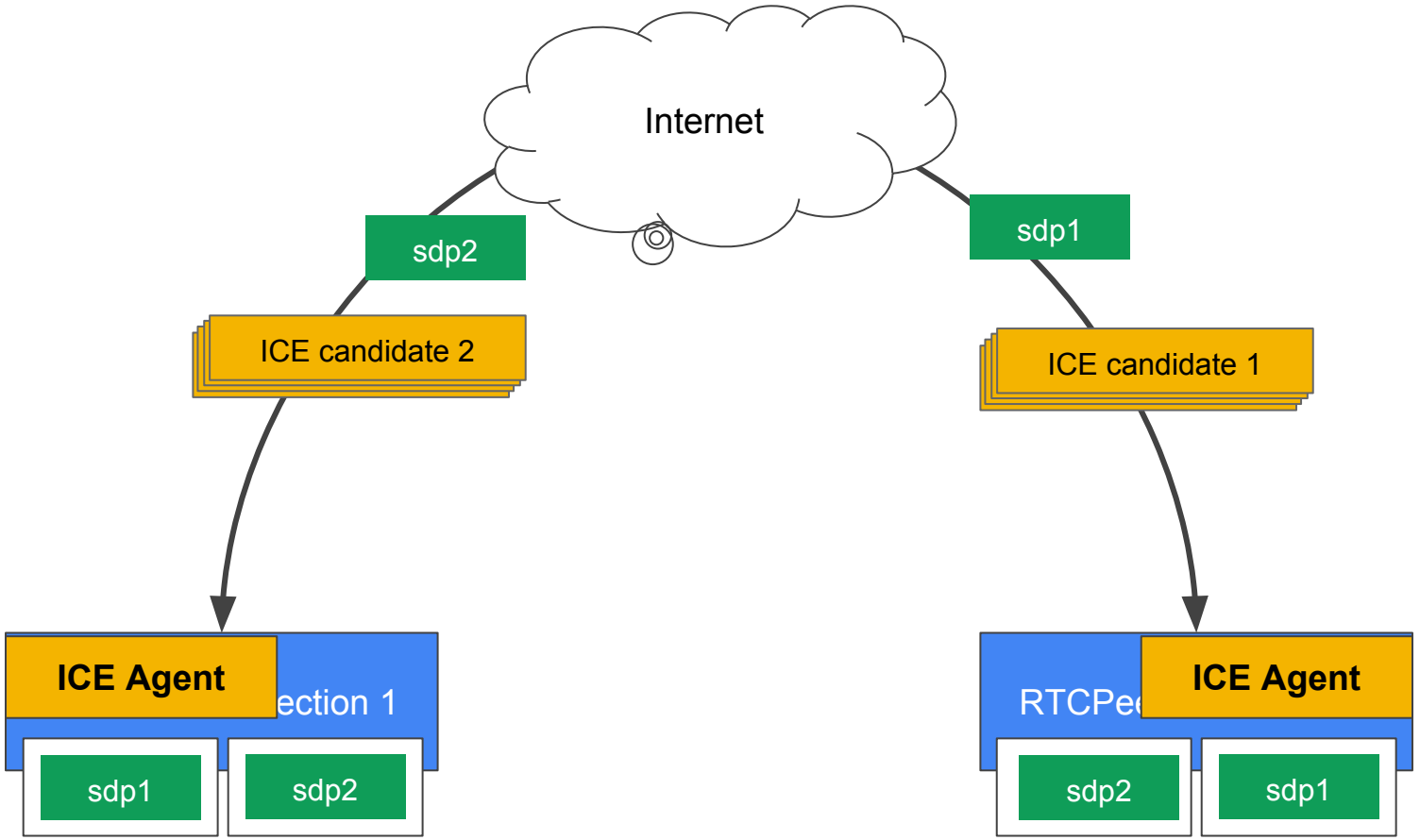
Webcam

<video>

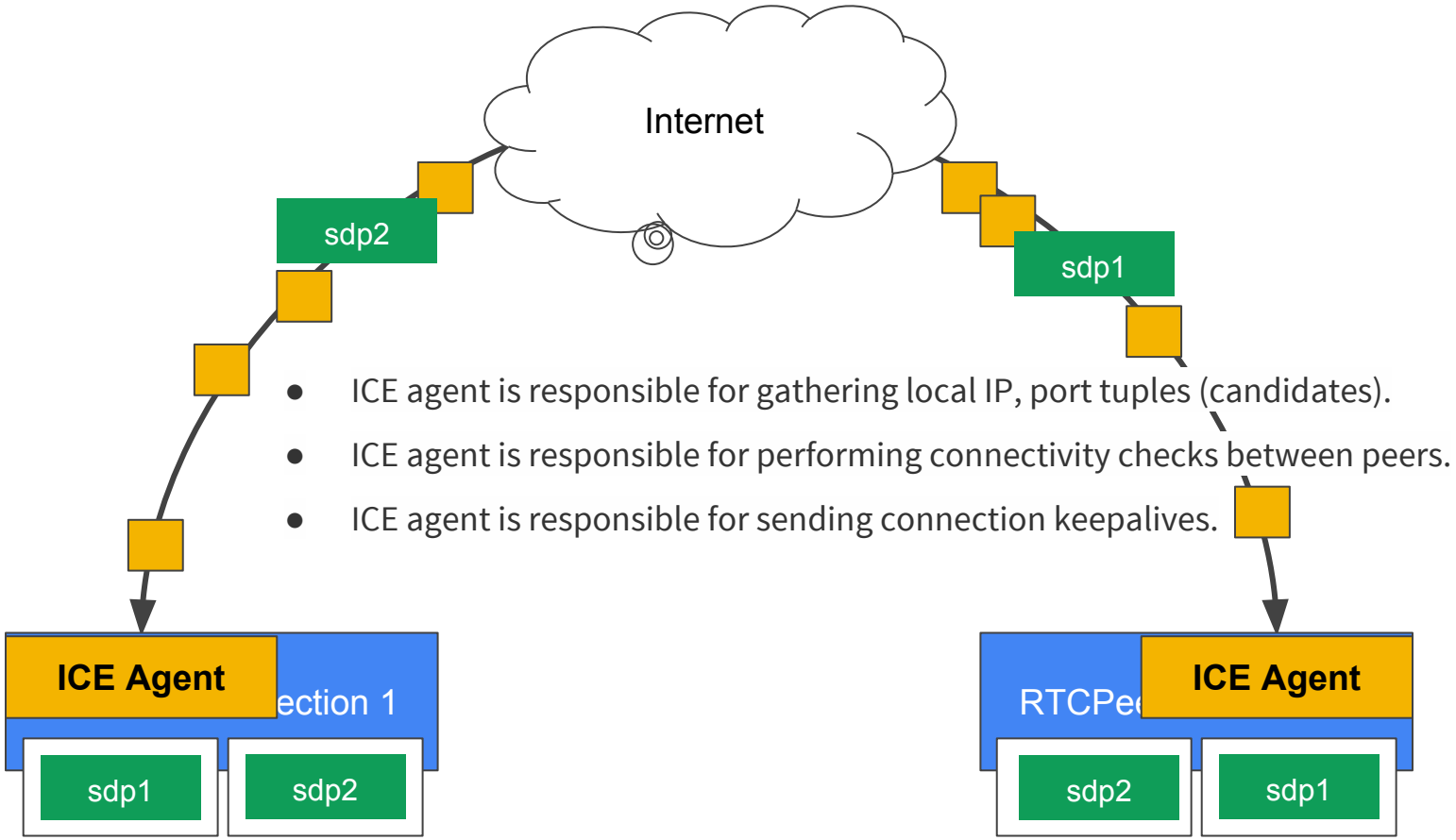
But wait, there is more...


Internet
sdp2
sdp1
RTCPeerConnection 1
sdp1
sdp2
RTCPeerConnection 2
sdp2
sdp1


Internet
sdp2
sdp1
ICE candidate 2
ICE candidate 1
RTCPeerConnection 1
sdp1
sdp2
RTCPeerConnection 2
sdp2
sdp1



ICE: Interactive Connectivity Establishment (RFC 5245)



In short, ICE agent finds a way to connect two peers

The whole process

We are trying to connect two RTCPeerConnection: from pc1 to pc2.

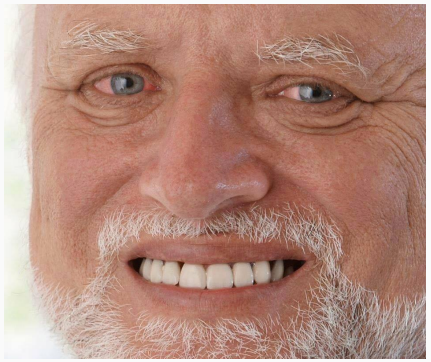
1. Add video stream to PC1
2. PC1 create an offer and get a session description (sessionDescription1)
3. PC1 set sessionDescription1 as local description
4. PC1 send sessionDescription1 to PC2
5. PC2 set sessionDescription1 as remote description
6. PC2 create an answer and get another session description (sessionDescription2)
7. PC2 set sessionDescription2 as local descrition
8. PC2 send sessionDescription2 to PC1
9. PC1 set sessionDescription2 as remote description
10. PC1 and PC2 exchange ICE candidate
11. Ping!

(ICE candidate can be exchanged right after set local description)

The whole process

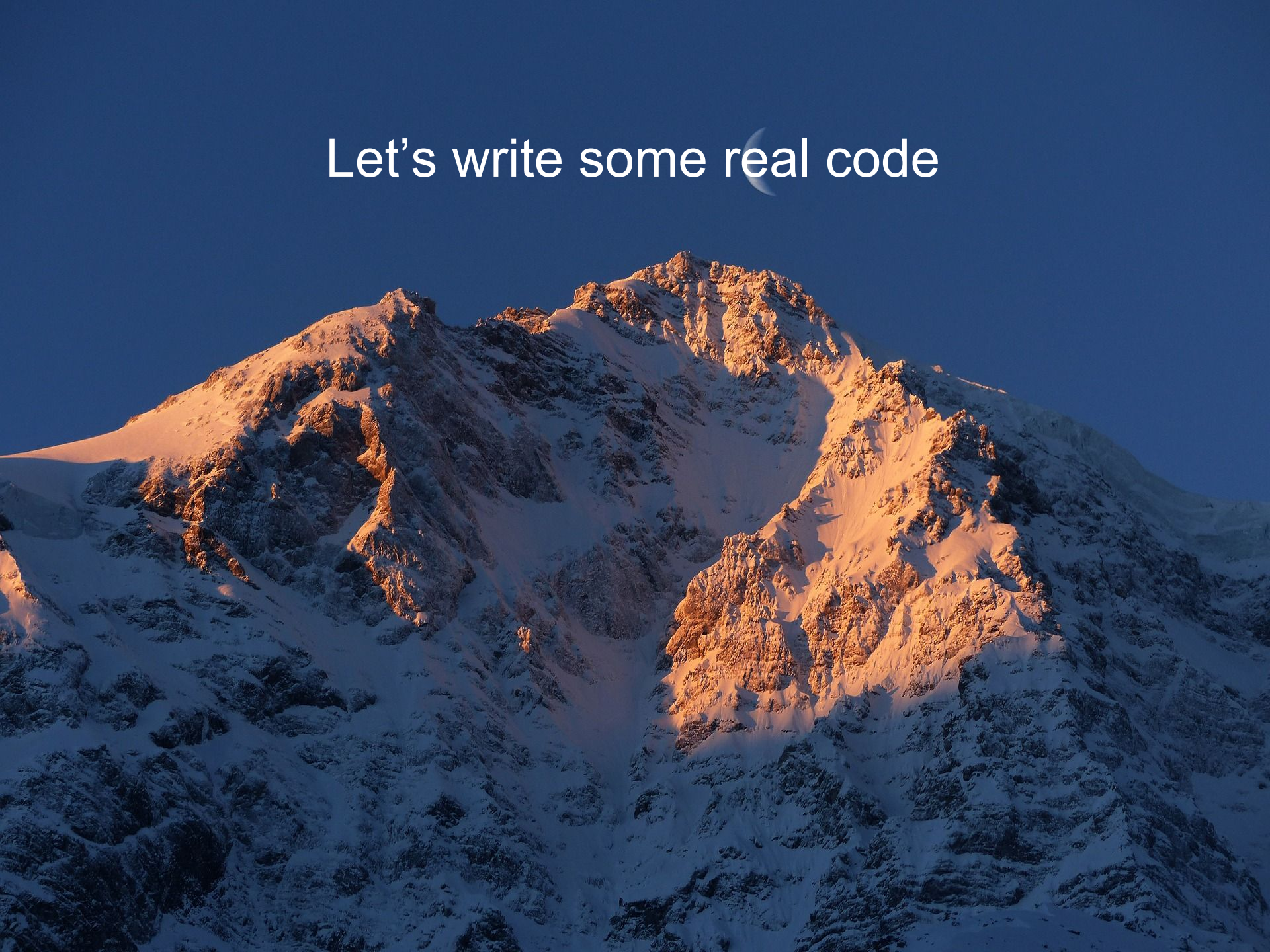
We are trying to connect two RTCPeerConnection: from pc1 to pc2.

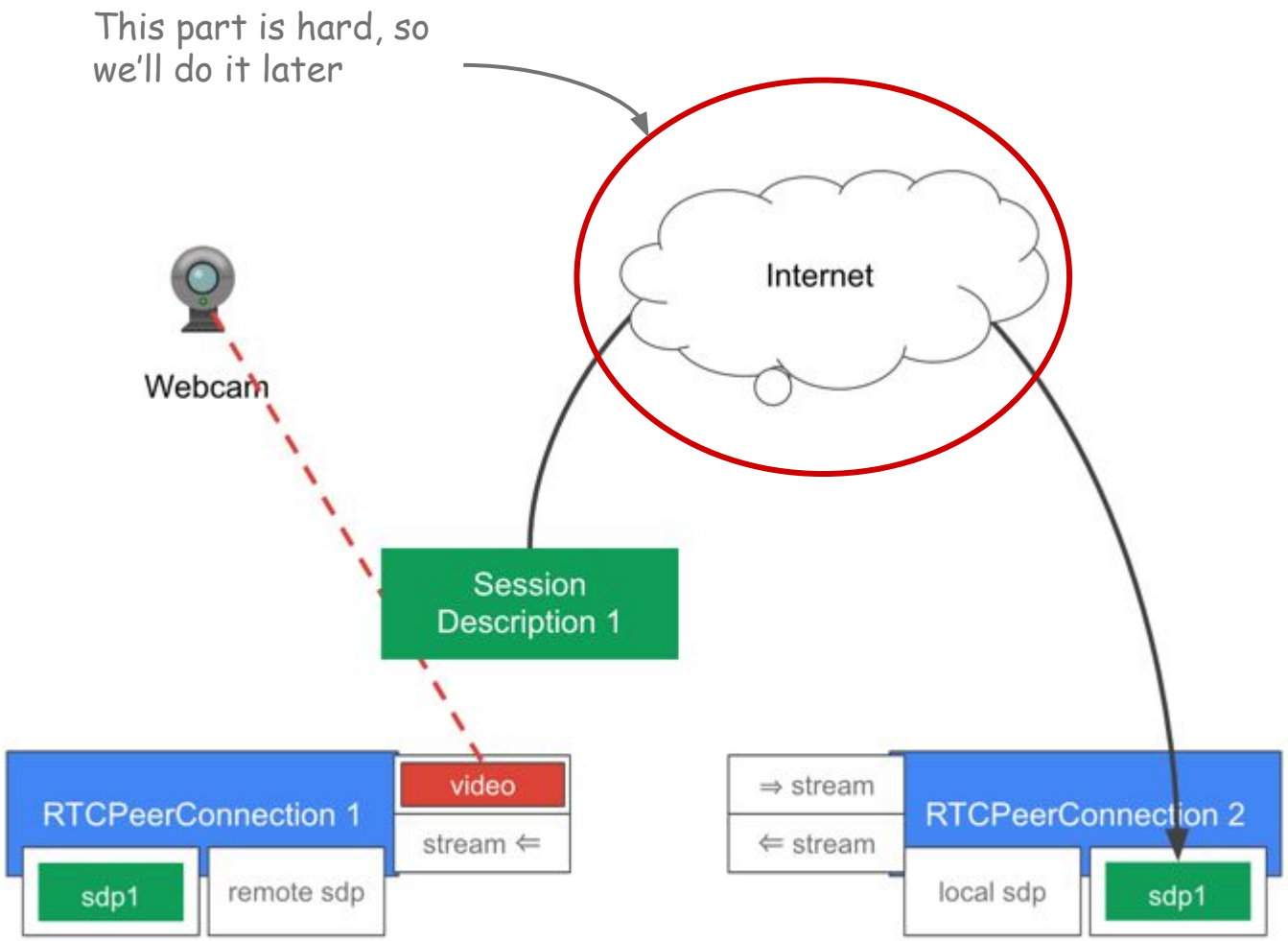
1. Add video stream to PC1
2. PC1 create an offer and get a session description (sessionDescription1)
3. PC1 set sessionDescription1 as local description
4. PC1 send sessionDescription1 to PC2
5. PC2 set sessionDescription1 as remote description
6. PC2 create an answer and get another session description (sessionDescription2)
7. PC2 set sessionDescription2 as local descrition
8. PC2 send sessionDescription2 to PC1
9. PC1 set sessionDescription2 as remote description
10. PC1 and PC2 exchange ICE candidate
11. Ping!



:dieinside:

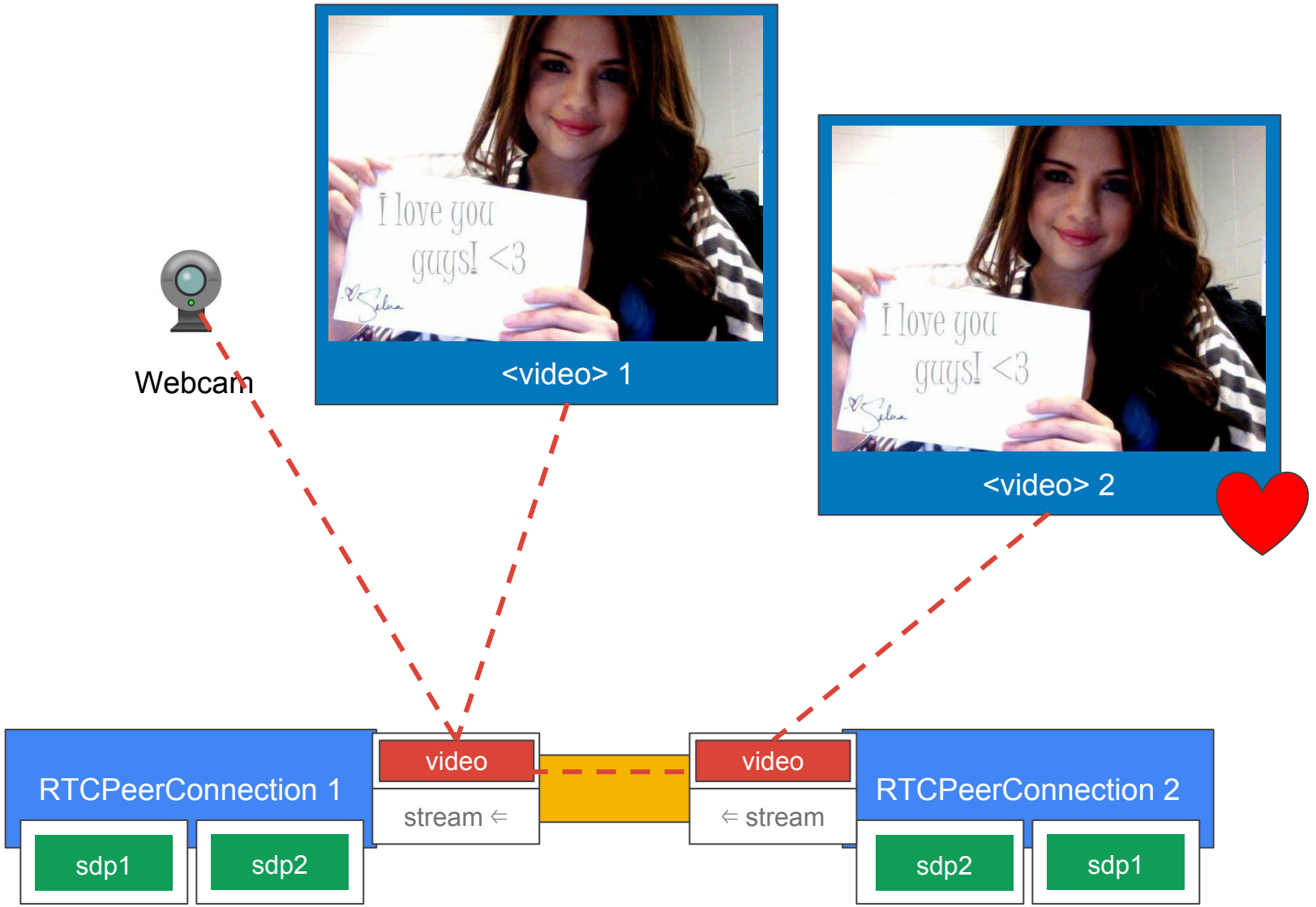
Let’s write some real code



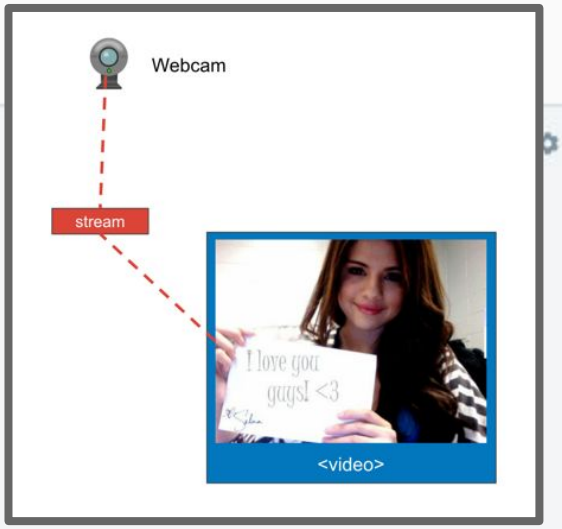


We can have 2 peer connection in the **same** page.

Same page demo



```
var constraints = {
  audio: false,
  video: true
};

var video = document.querySelector('video');

var videoStream;

function successCallback(stream) {
    // Show to <video> tag
    videoStream = stream;
    video.srcObject = videoStream;
}

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

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



https://jsfiddle.net/trungdq88/y69v2yzn/1/

HTML

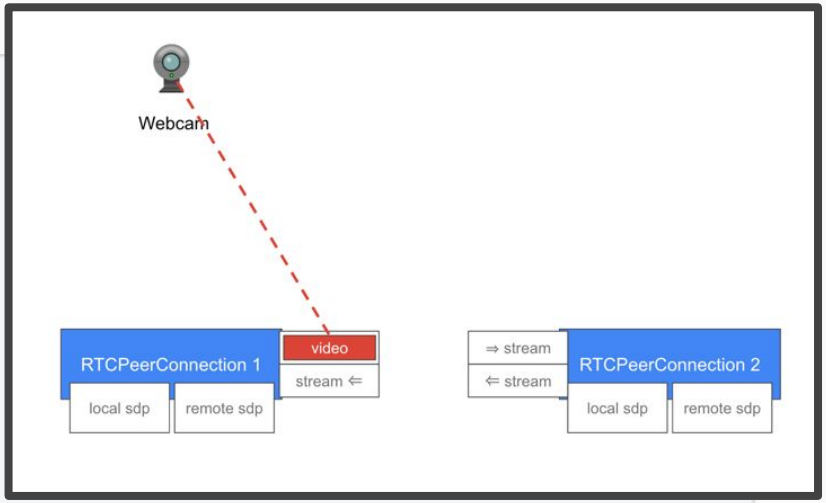
```html
<video id="video1" width="200px" height="200px" autoplay></video>
<video id="video2" width="200px" height="200px" autoplay></video>
```

JAVASCRIPT

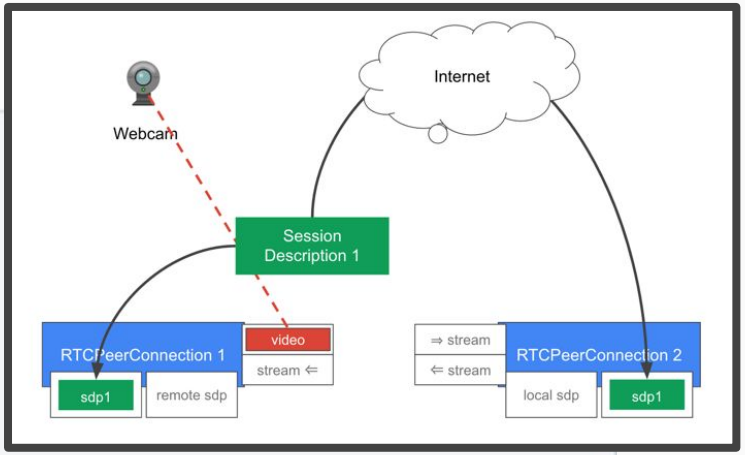
```javascript
var constraints = {
  audio: false,
  video: true
};

var video1 = document.querySelector('#video1');
var video2 = document.querySelector('#video2');

var videoStream;

function successCallback(stream) {
    // Show to <video> tag
    videoStream = stream;
    video1.srcObject = videoStream;
}

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

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

https://jsfiddle.net/trungdq88/y69v2yzn/2/

```
var constraints = {
  audio: false,
  video: true
};

var video1 = document.querySelector('#video1');
var video2 = document.querySelector('#video2');

var videoStream;

function successCallback(stream) {
    // Show to <video> tag
    videoStream = stream;
    video1.srcObject = videoStream;

    startCalling();
}

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

navigator.webkitGetUserMedia(constraints, successCallback, errorCallback);

function startCalling() {
  var pc1 = new webkitRTCPeerConnection(null);
  var pc2 = new webkitRTCPeerConnection(null);

  pc1.addStream(videoStream);
}
```

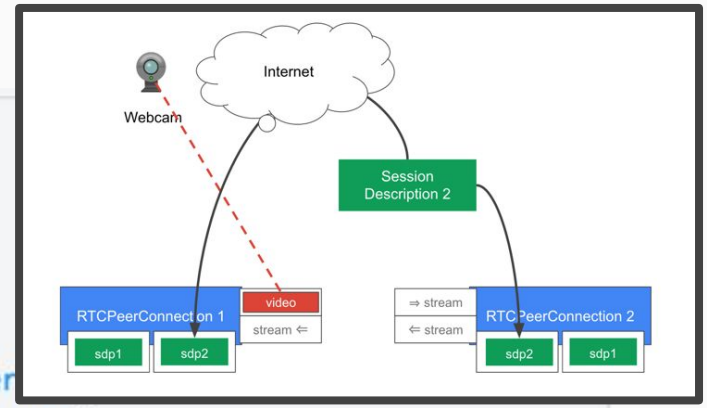


https://jsfiddle.net/trungdq88/y69v2yzn/3/



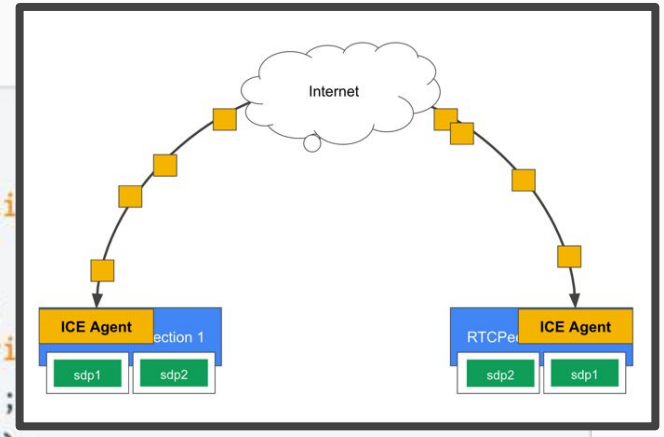

```
11 function successCallback(stream) {
12     // Show to <video> tag
13     videoStream = stream;
14     video1.srcObject = videoStream;
15
16     startCalling();
17 }
18
19 function errorCallback(error) {
20   console.log('navigator.getUserMedia error: ', error);
21 }
22
23 navigator.webkitGetUserMedia(constraints, successCallback, errorCallback);
24
25 function startCalling() {
26   var pc1 = new webkitRTCPeerConnection(null);
27   var pc2 = new webkitRTCPeerConnection(null);
28
29   pc1.addStream(videoStream);
30   pc1.createOffer().then(function (sessionDescription) {
31     pc1.setLocalDescription(sessionDescription);
32     pc2.setRemoteDescription(sessionDescription);
33   });
34 }
```

https://jsfiddle.net/trungdq88/y69v2yzn/4/




```
    startCalling();
}

function errorCallback(error) {
  console.log('navigator.getUserMedia error: ', er
}

navigator.webkitGetUserMedia(constraints, successCallback, errorCallback);

function startCalling() {
  var pc1 = new webkitRTCPeerConnection(null);
  var pc2 = new webkitRTCPeerConnection(null);

  pc1.addStream(videoStream);
  pc1.createOffer().then(function (sessionDescription1) {
    pc1.setLocalDescription(sessionDescription1);
    pc2.setRemoteDescription(sessionDescription1);
    pc2.createAnswer().then(function (sessionDescription2) {
      pc2.setLocalDescription(sessionDescription2);
      pc1.setRemoteDescription(sessionDescription2);
    })
  });
```

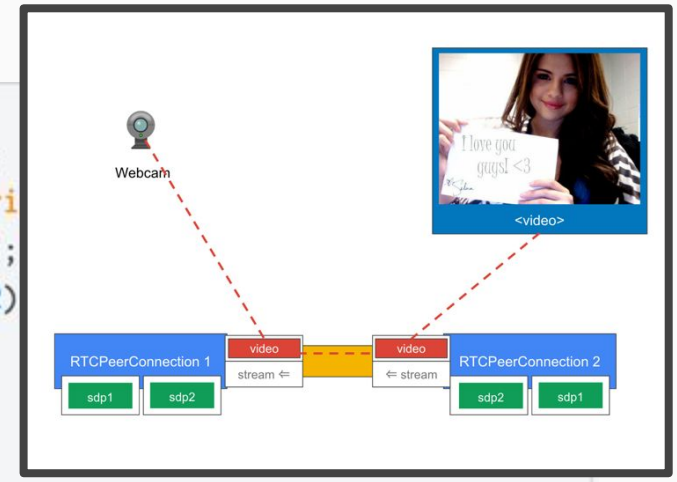
https://jsfiddle.net/trungdq88/y69v2yzn/6/




```
28
29    pc1.addStream(videoStream);
30    pc1.createOffer().then(function (sessionDescripti
31      pc1.setLocalDescription(sessionDescription1);
32      pc2.setRemoteDescription(sessionDescription1);
33      pc2.createAnswer().then(function (sessionDescri
34        pc2.setLocalDescription(sessionDescription2);
35        pc1.setRemoteDescription(sessionDescription2);
36      })
37    });
38
39    pc1.onicecandidate = function (event) {
40      if (event.candidate) {
41        pc2.addIceCandidate(new RTCIceCandidate(event.candidate));
42      }
43    }
44
45    pc2.onicecandidate = function (iceCandidate) {
46      if (event.candidate) {
47        pc1.addIceCandidate(new RTCIceCandidate(event.candidate));
48      }
49    }
50
51  }
```

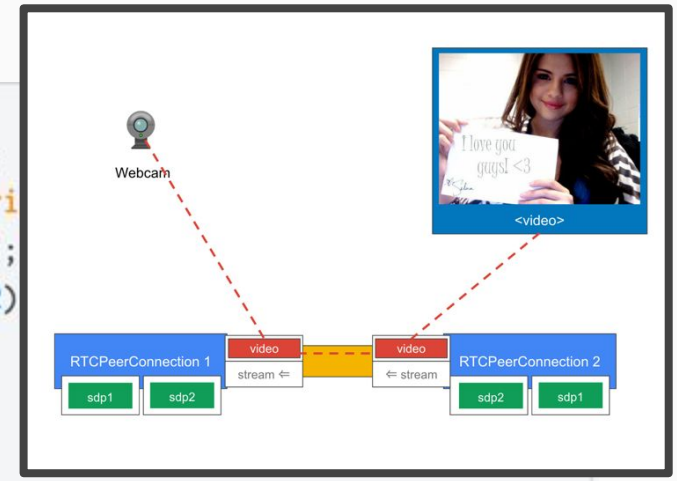
https://jsfiddle.net/trungdq88/y69v2yzn/6/

```
31      pc1.setLocalDescription(sessionDescription1);
32      pc2.setRemoteDescription(sessionDescription1);
33      pc2.createAnswer().then(function (sessionDescri
34        pc2.setLocalDescription(sessionDescription2);
35        pc1.setRemoteDescription(sessionDescription2)
36      })
37    });
38
39    pc1.onicecandidate = function (event) {
40      if (event.candidate) {
41        pc2.addIceCandidate(new RTCIceCandidate(event.candidate));
42      }
43    }
44
45    pc2.onicecandidate = function (iceCandidate) {
46      if (event.candidate) {
47        pc1.addIceCandidate(new RTCIceCandidate(event.candidate));
48      }
49    }
50
51    pc2.onaddstream = function (e) {
52      video2.srcObject = videoStream
53    }
54  }
```



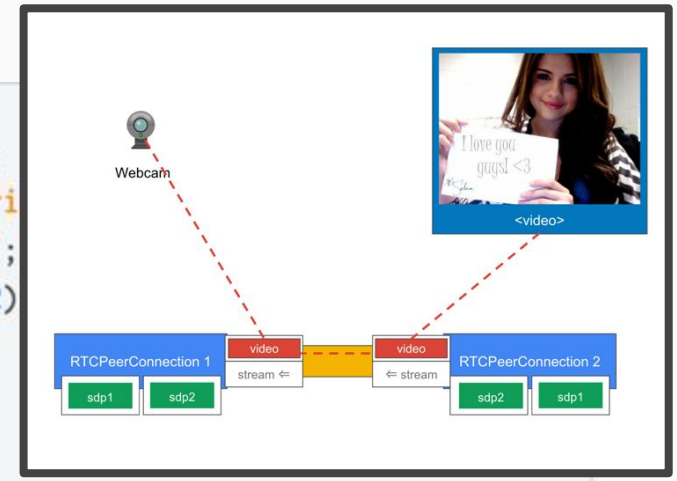
https://jsfiddle.net/trungdq88/y69v2yzn/7/

```
31      pc1.setLocalDescription(sessionDescription1);
32      pc2.setRemoteDescription(sessionDescription1);
33      pc2.createAnswer().then(function (sessionDescri
34        pc2.setLocalDescription(sessionDescription2);
35        pc1.setRemoteDescription(sessionDescription2)
36      })
37    });
38
39    pc1.onicecandidate = function (event) {
40      if (event.candidate) {
41        pc2.addIceCandidate(new RTCIceCandidate(event.candidate));
42      }
43    }
44
45    pc2.onicecandidate = function (iceCandidate) {
46      if (event.candidate) {
47        pc1.addIceCandidate(new RTCIceCandidate(event.candidate));
48      }
49    }
50
51    pc2.onaddstream = function (e) {
52      video2.srcObject = videoStream
53    }
54  }
```



https://jsfiddle.net/trungdq88/y69v2yzn/7/

```
31      pc1.setLocalDescription(sessionDescription1);
32      pc2.setRemoteDescription(sessionDescription1);
33      pc2.createAnswer().then(function (sessionDescri
34        pc2.setLocalDescription(sessionDescription2);
35        pc1.setRemoteDescription(sessionDescription2)
36      })
37    });
38
39    pc1.onicecandidate = function (event) {
40      if (event.candidate) {
41        pc2.addIceCandidate(new RTCIceCandidate(event.candidate));
42      }
43    }
44
45    pc2.onicecandidate = function (iceCandidate) {
46      if (event.candidate) {
47        pc1.addIceCandidate(new RTCIceCandidate(event.candidate));
48      }
49    }
50
51    pc2.onaddstream = function (e) {
52      video2.srcObject = e.stream;
53    }
54  }
```



This is the video stream we added from pc1

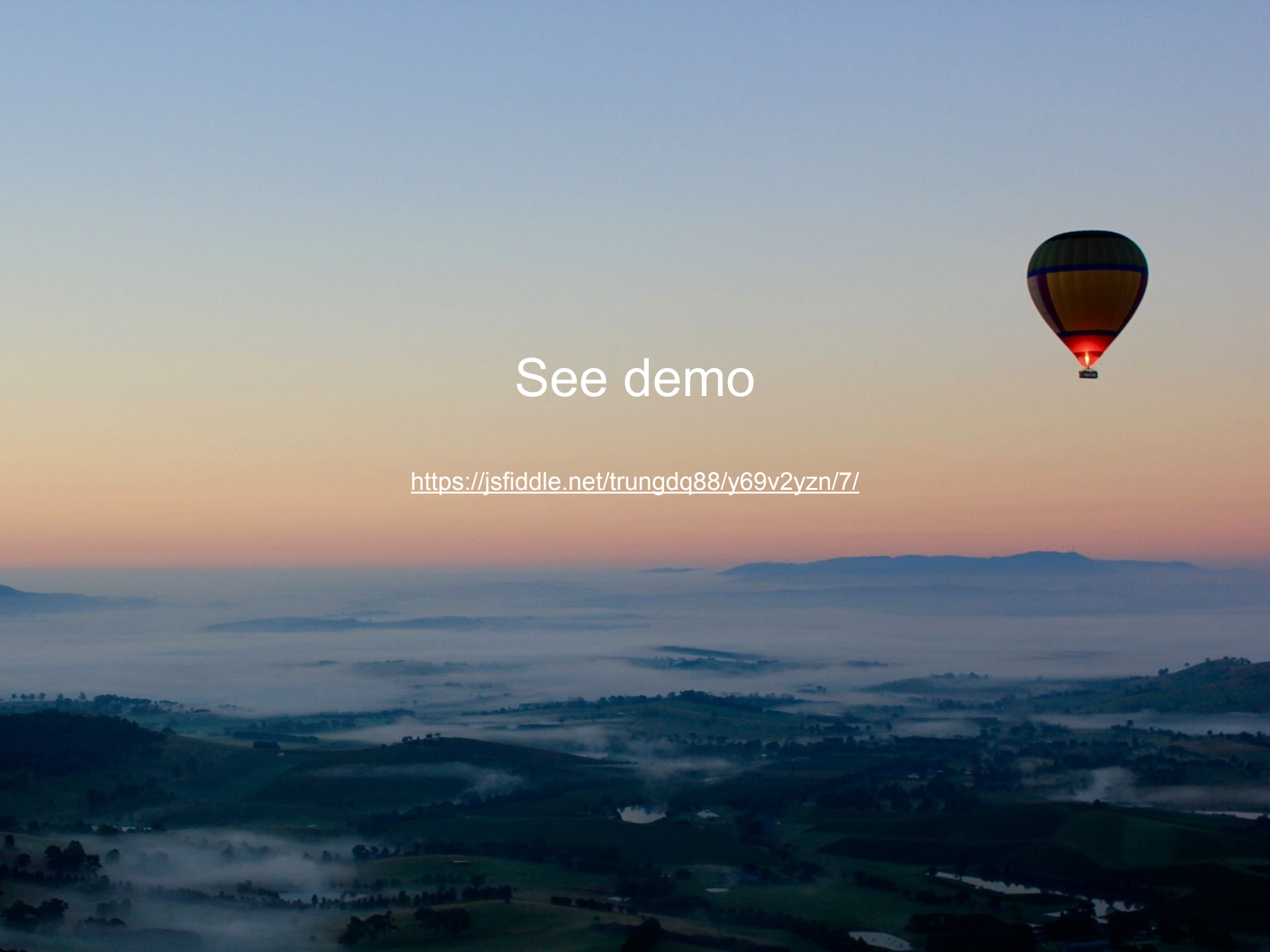
https://jsfiddle.net/trungdq88/y69v2yzn/7/

JAVASCRIPT

```
    pc1.setLocalDescription(sessionDescription1);
    pc2.setRemoteDescription(sessionDescription1);
    pc2.createAnswer().then(function (sessionDescription2) {
      pc2.setLocalDescription(sessionDescription2);
      pc1.setRemoteDescription(sessionDescription2);
    })
  });

  pc1.onicecandidate = function (event) {
    if (event.candidate) {
      pc2.addIceCandidate(new RTCIceCandidate(event.candidate));
    }
  }

  pc2.onicecandidate = function (iceCandidate) {
    if (event.candidate) {
      pc1.addIceCandidate(new RTCIceCandidate(event.candidate));
    }
  }

  pc2.onaddstream = function (e) {
    video2.srcObject = e.stream;
  }
}
```

Ping!

https://jsfiddle.net/trungdq88/y69v2yzn/7/

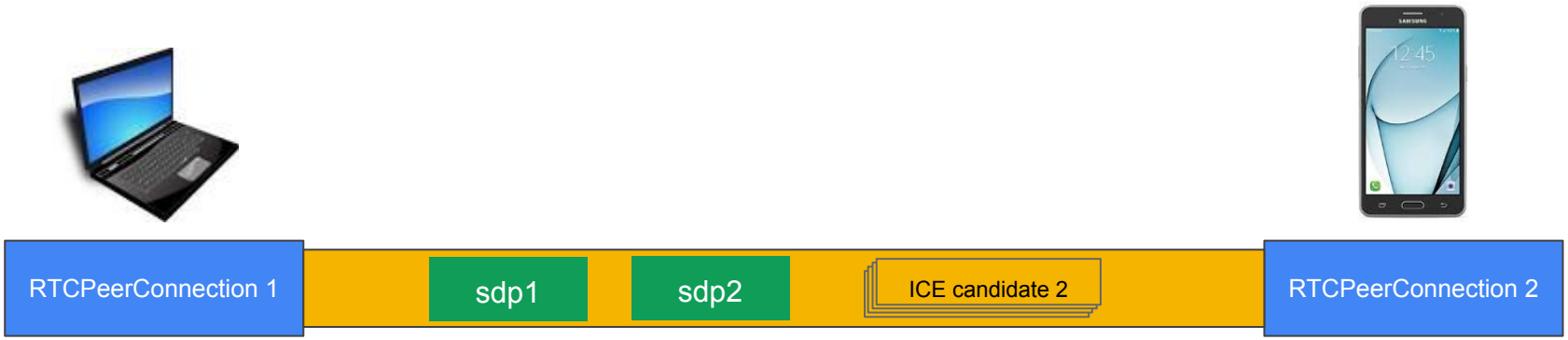


See demo

https://jsfiddle.net/trungdq88/y69v2yzn/7/

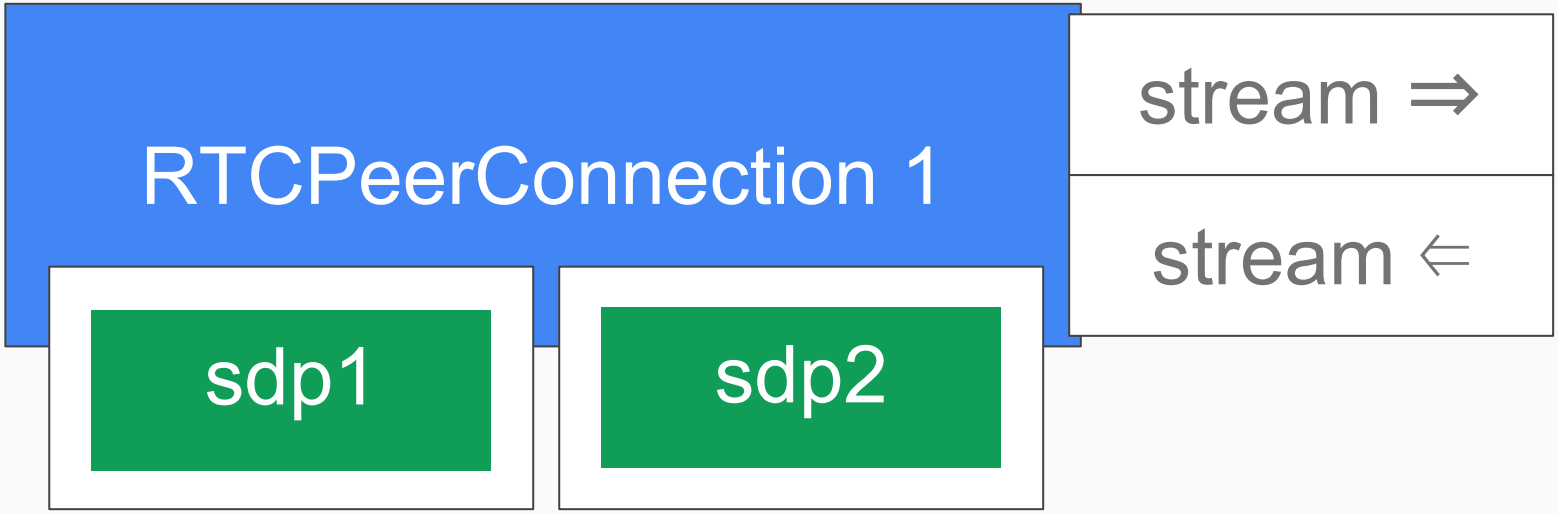
In reality...

pc1 and **pc2** can be in **different machine/device**, as long as you can **exchange** the **session descriptions** and **ICE candidates** between them.

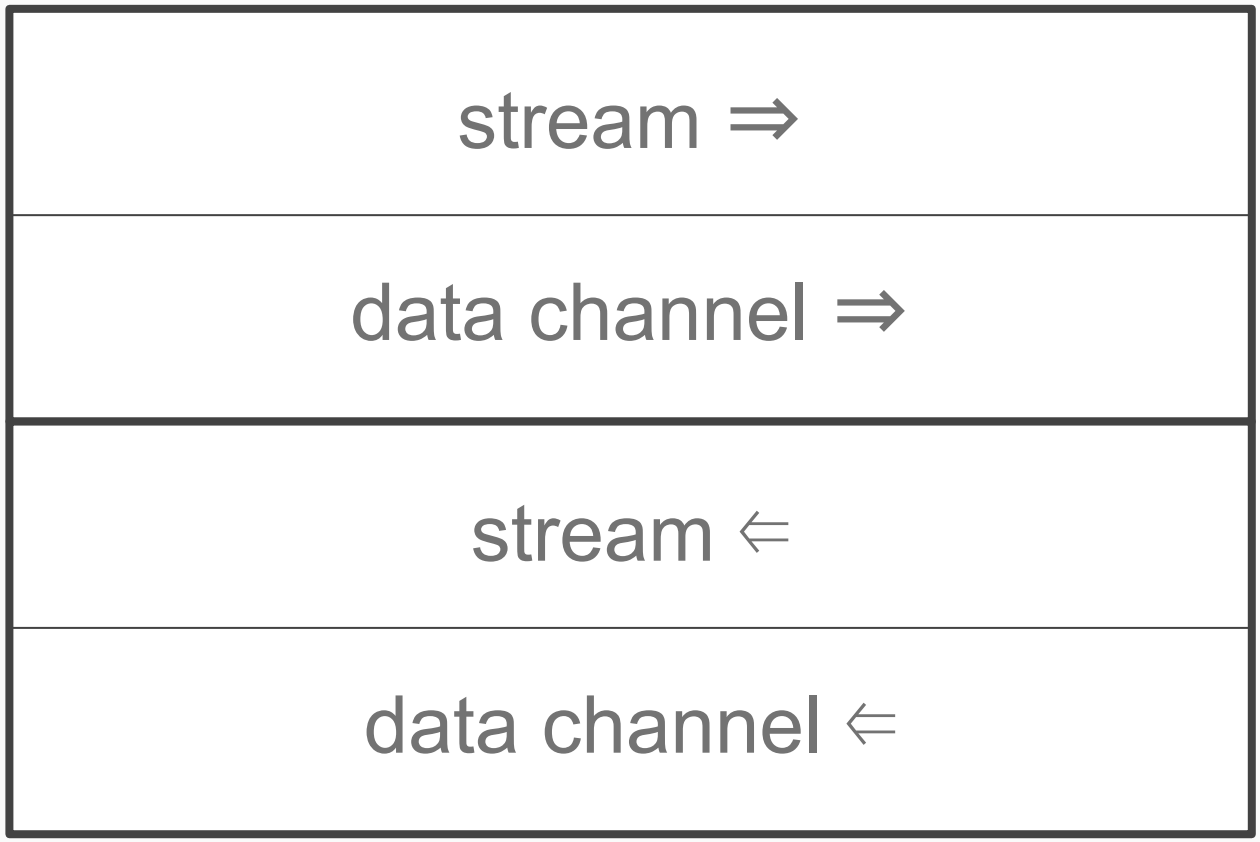


RTCDataChannel

Transfering data peer to peer


RTCPeerConnection 1
stream ⇒
stream ⇐
sdp1
sdp2


stream ⇒
stream ⇐

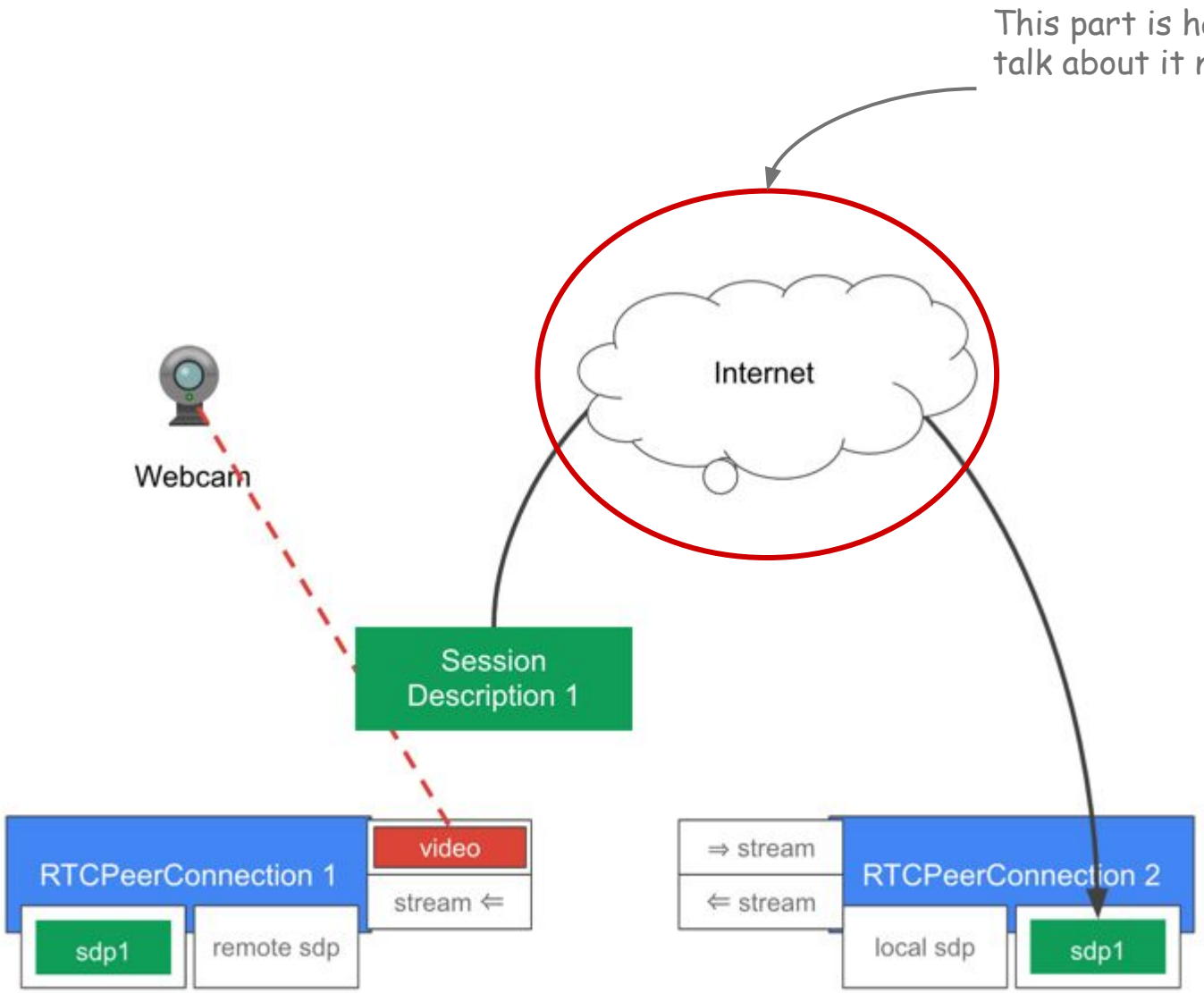

stream ⇒
data channel ⇒
stream ⇐
data channel ⇐

JAVASCRIPT

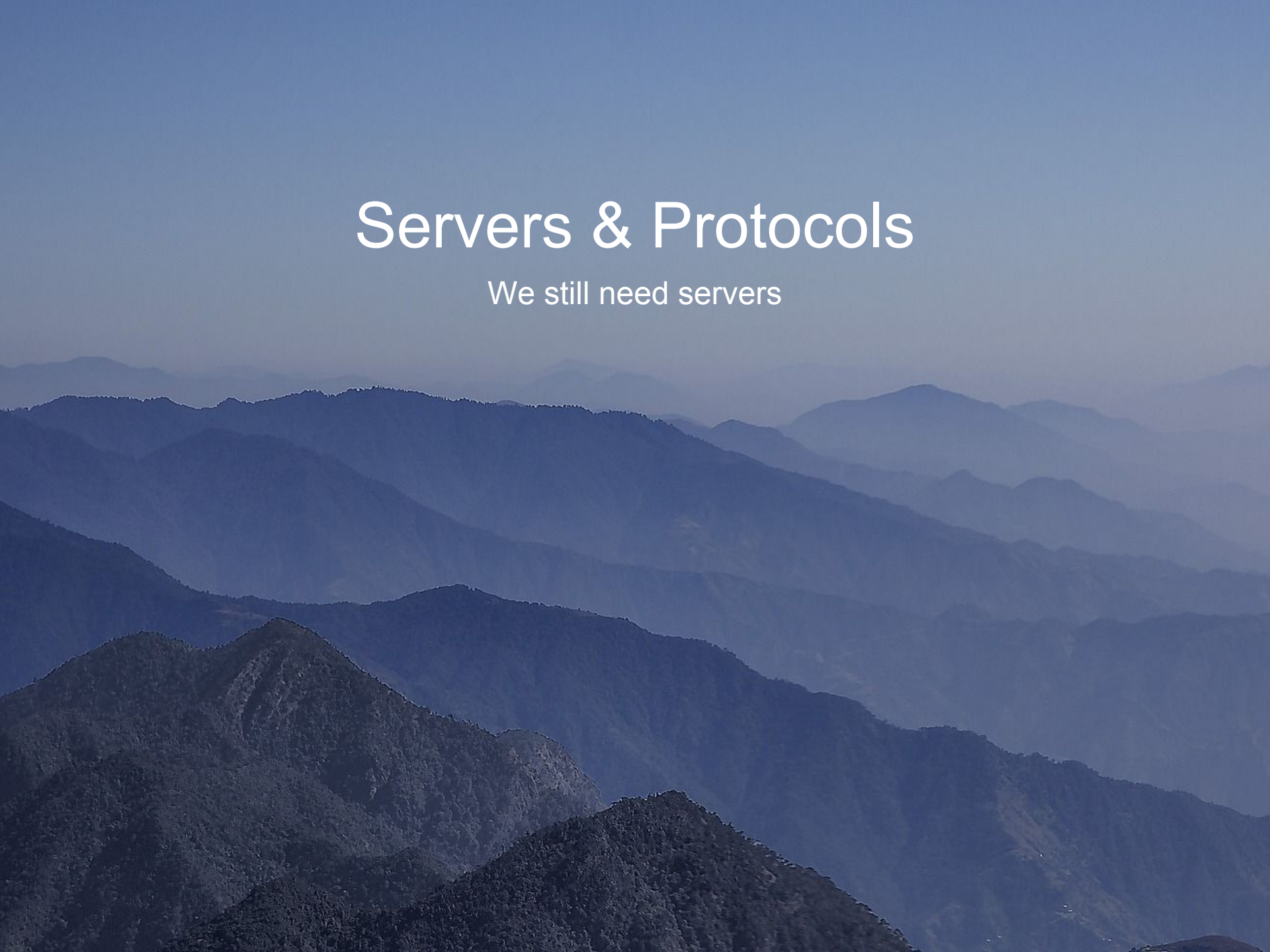
```javascript
sendChannel = pc1.createDataChannel("sendDataChannel");
sendChannel.send('Hello world!');

pc2.ondatachannel = function(event) {
  receiveChannel = event.channel;
  receiveChannel.onmessage = function(event){
    console.log(event.data); // --> 'Hello world!'
  };
};
```

WebRTC with RTCDataChannel

https://www.sharefest.me

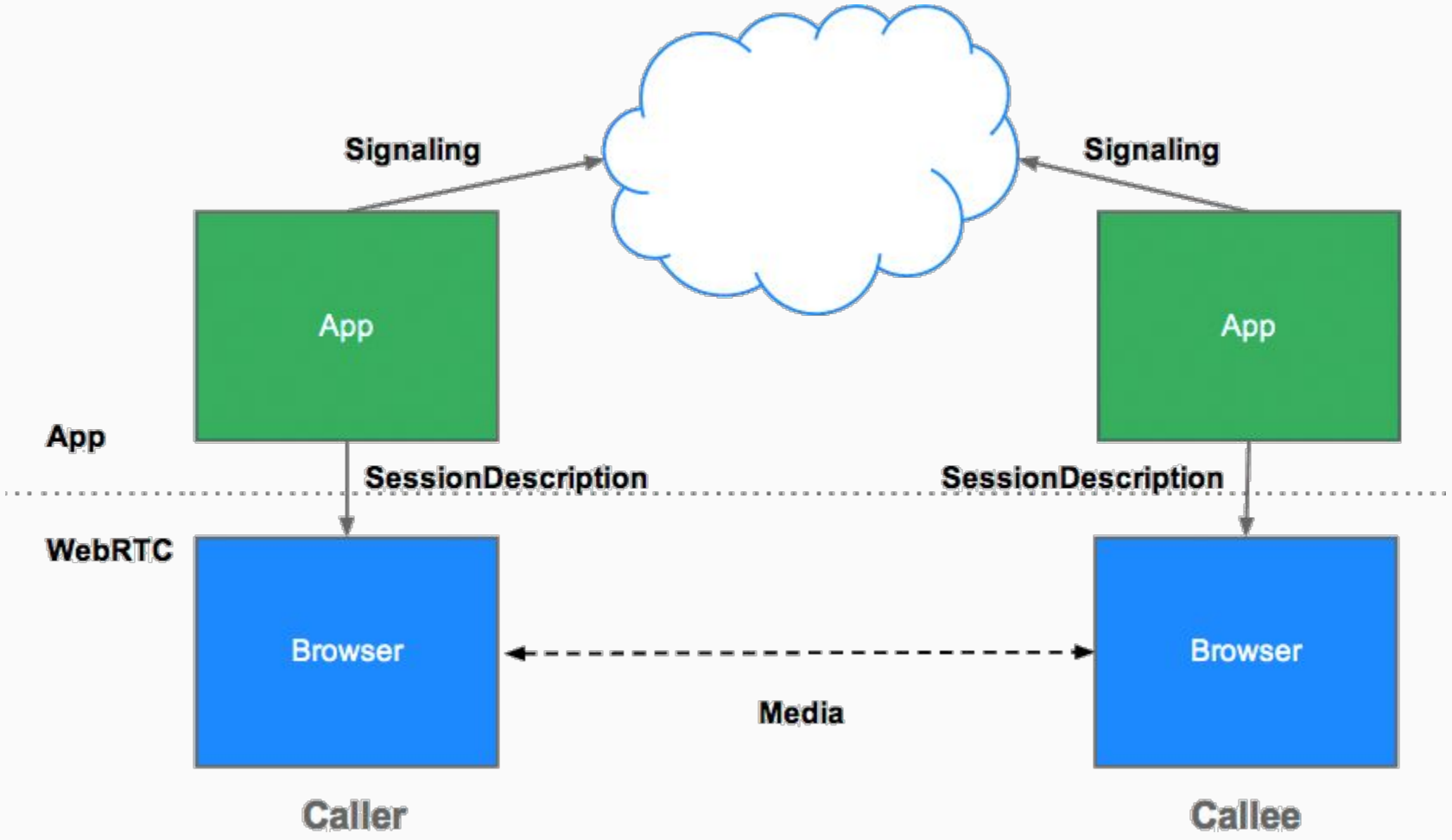


WebRTC don't care how you exchange the session descriptions and ICE candidates from peer to peer


Servers & Protocols
We still need servers

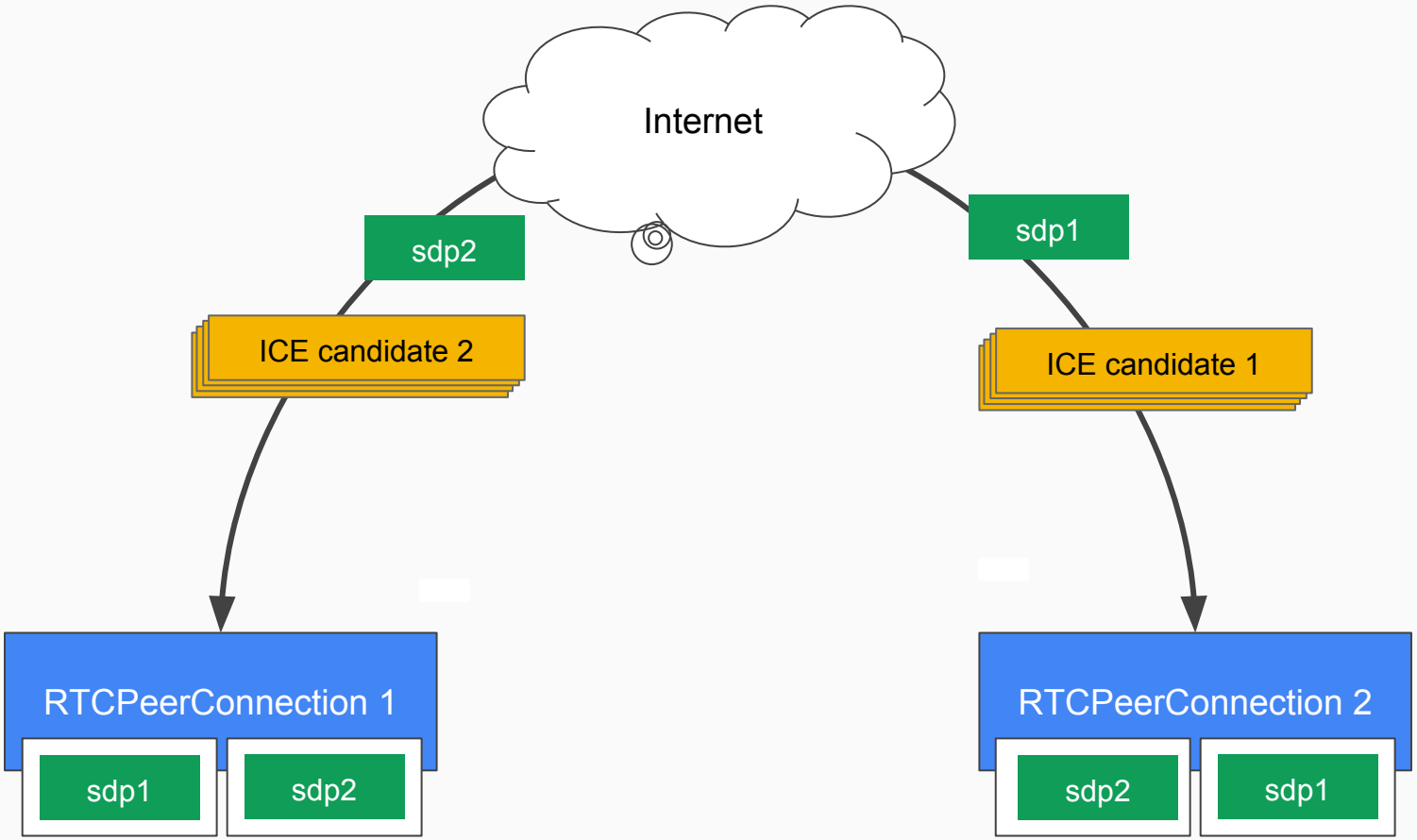
Server is used to:

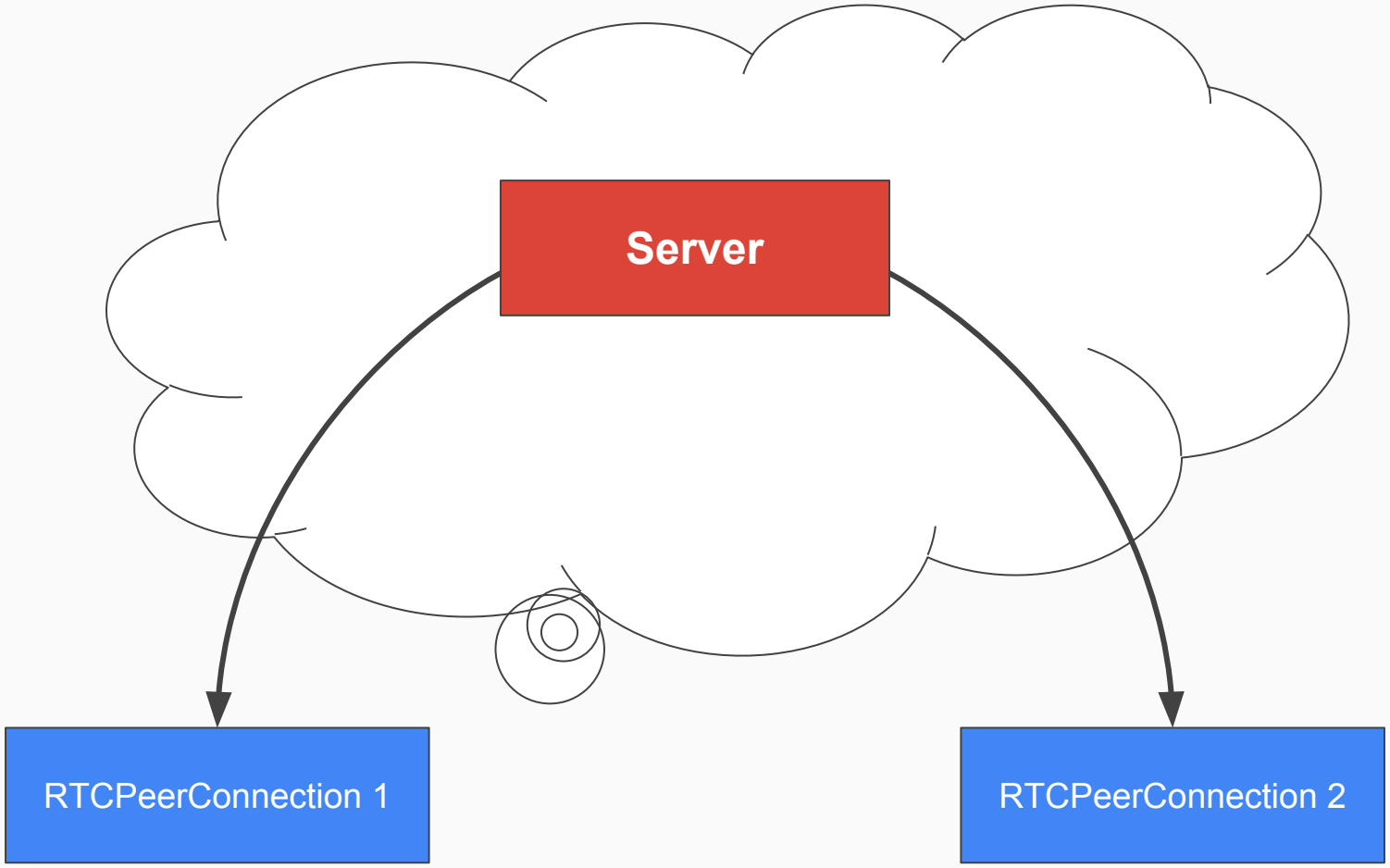
- Explore other peers, so you'll know who you want to connect to.
- Exchange "session description" object between peers
- Exchange ICE candidate information between peers

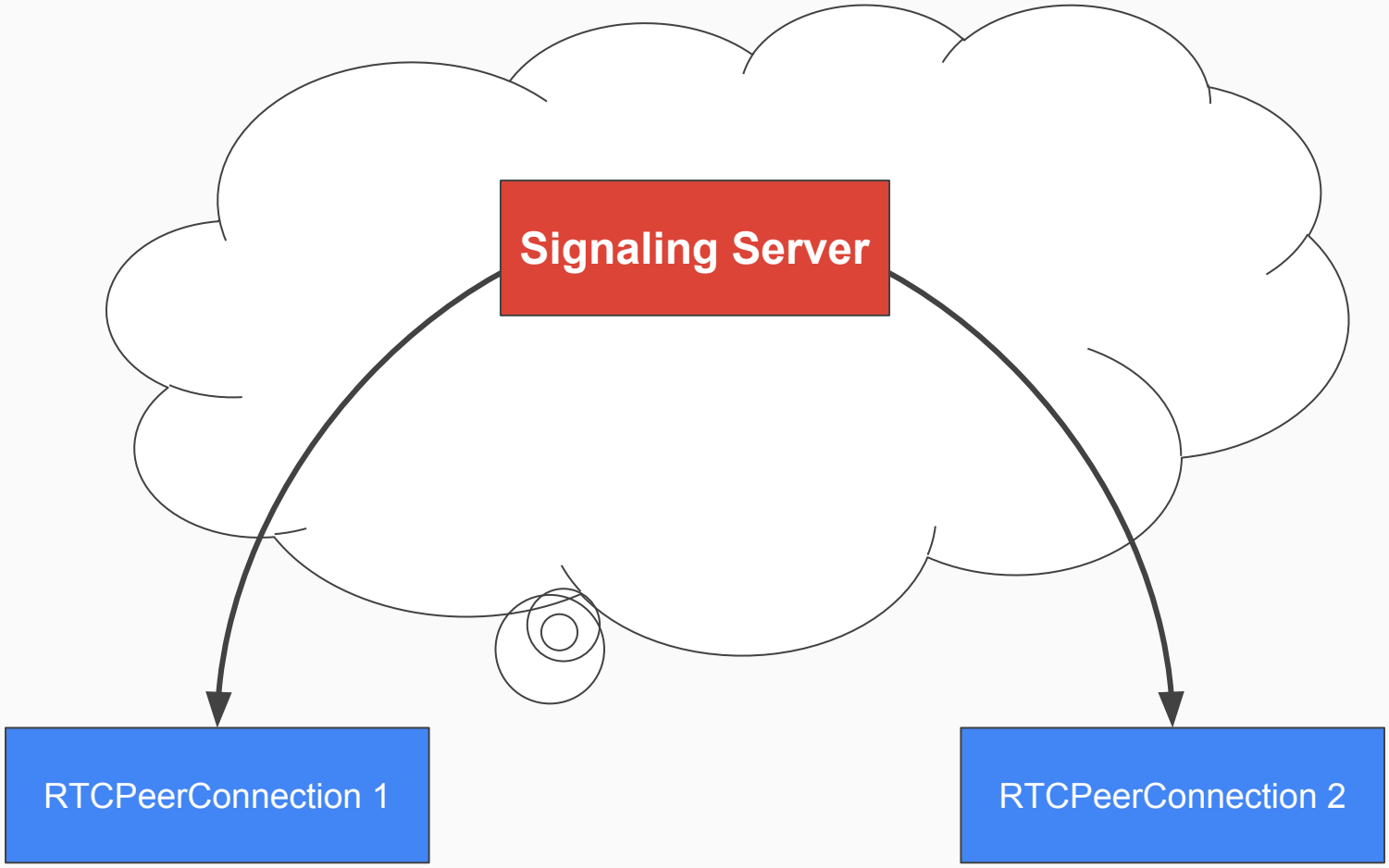

Signaling
Signaling
App
App
App
SessionDescription
SessionDescription
WebRTC
Browser
Browser
Media
Caller
Callee

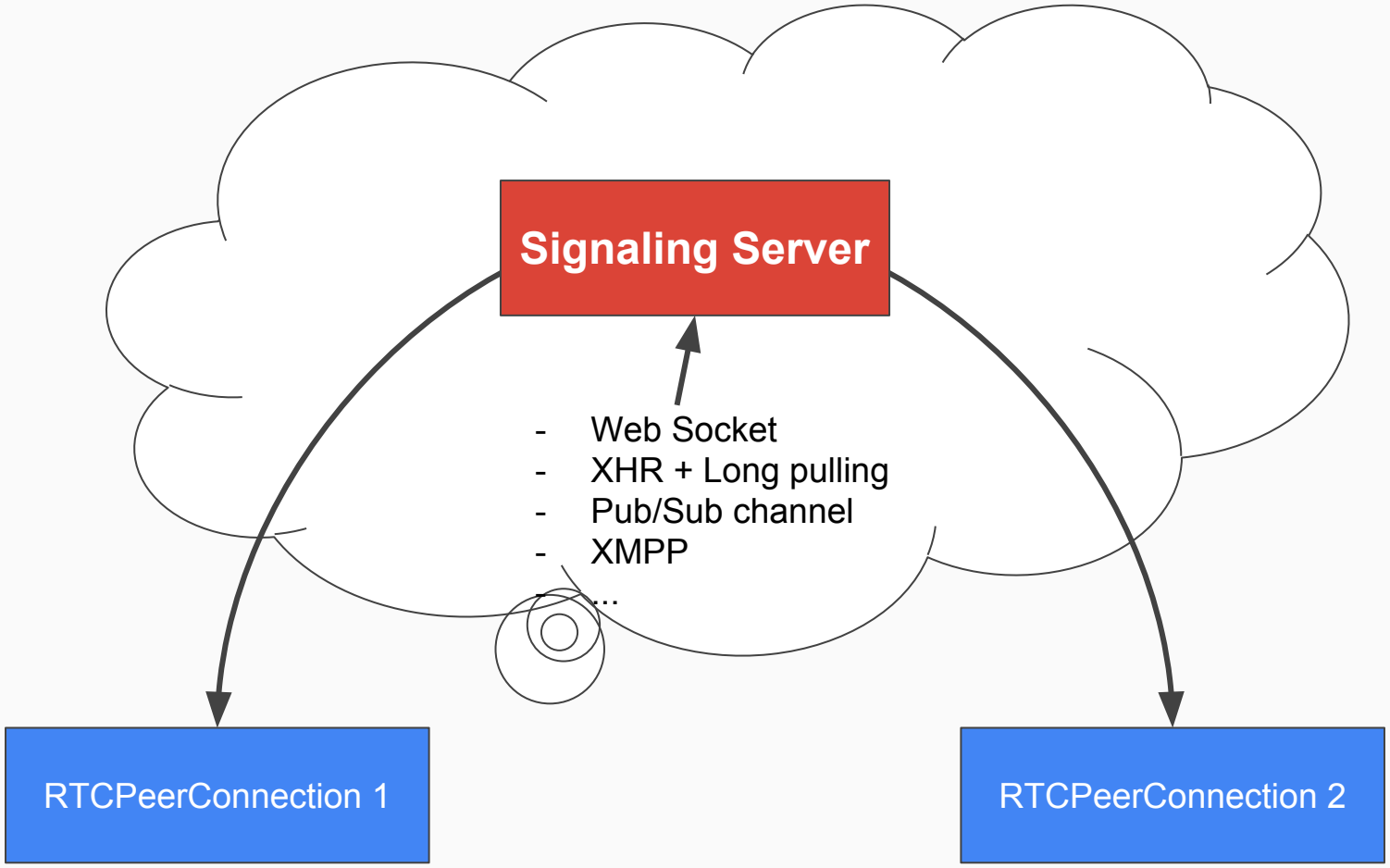
What is inside a Session Description

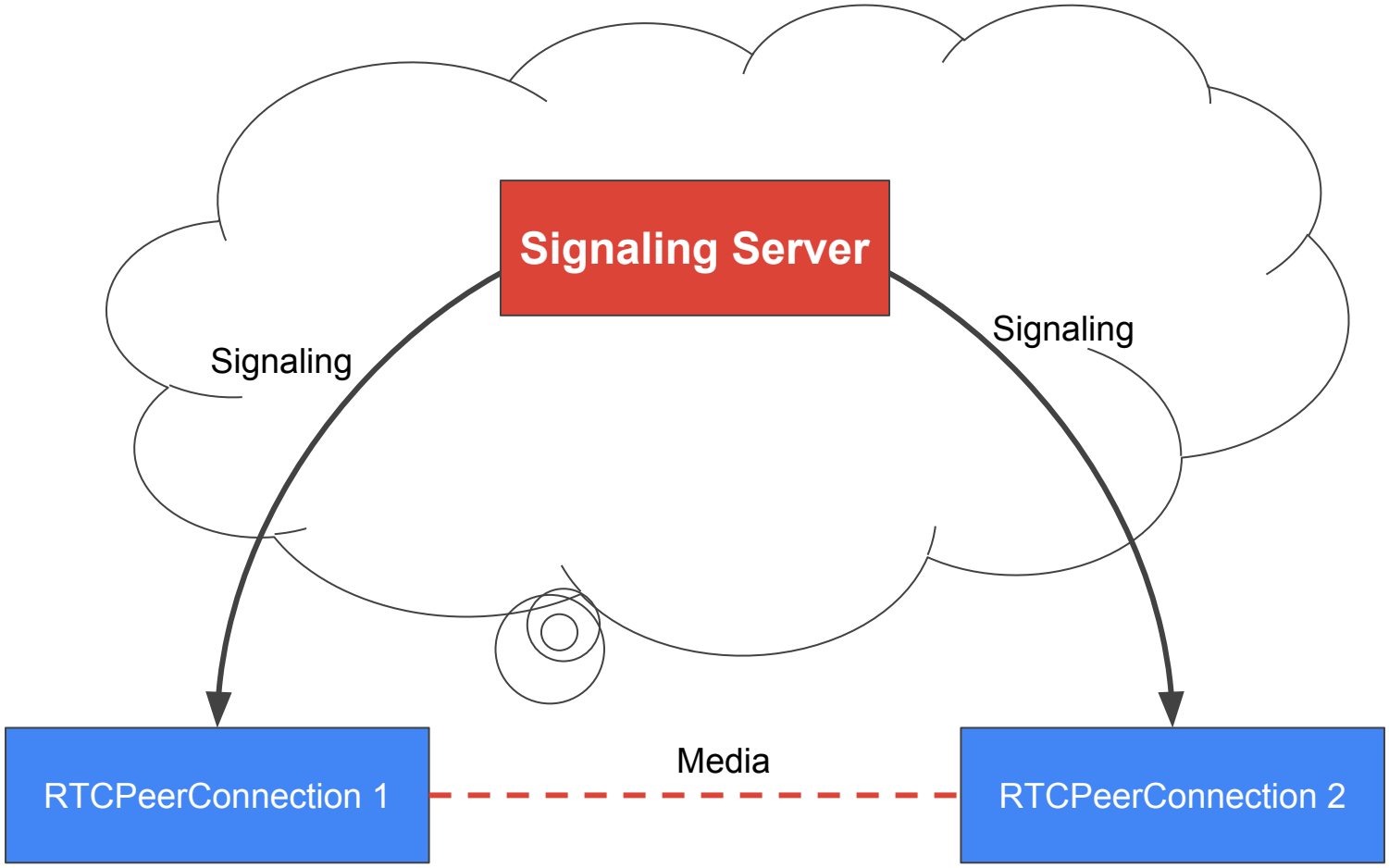
```
v=0
o=- 7614219274584779017 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS
m=audio 1 RTP/SAVPF 111 103 104 0 8 107 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:1 IN IP4 0.0.0.0
a=ice-ufrag:W2TGCZw2NZHuwlnf
a=ice-pwd:xdQEccP40E+P0L5qTyzDgfmW
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=mid:audio
a=rtcp-mux
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:9c1AHz27dZ9xPI91YNfSlI67/EMkjHHIHORiClQe
a=rtpmap:111 opus/48000/2
...
```

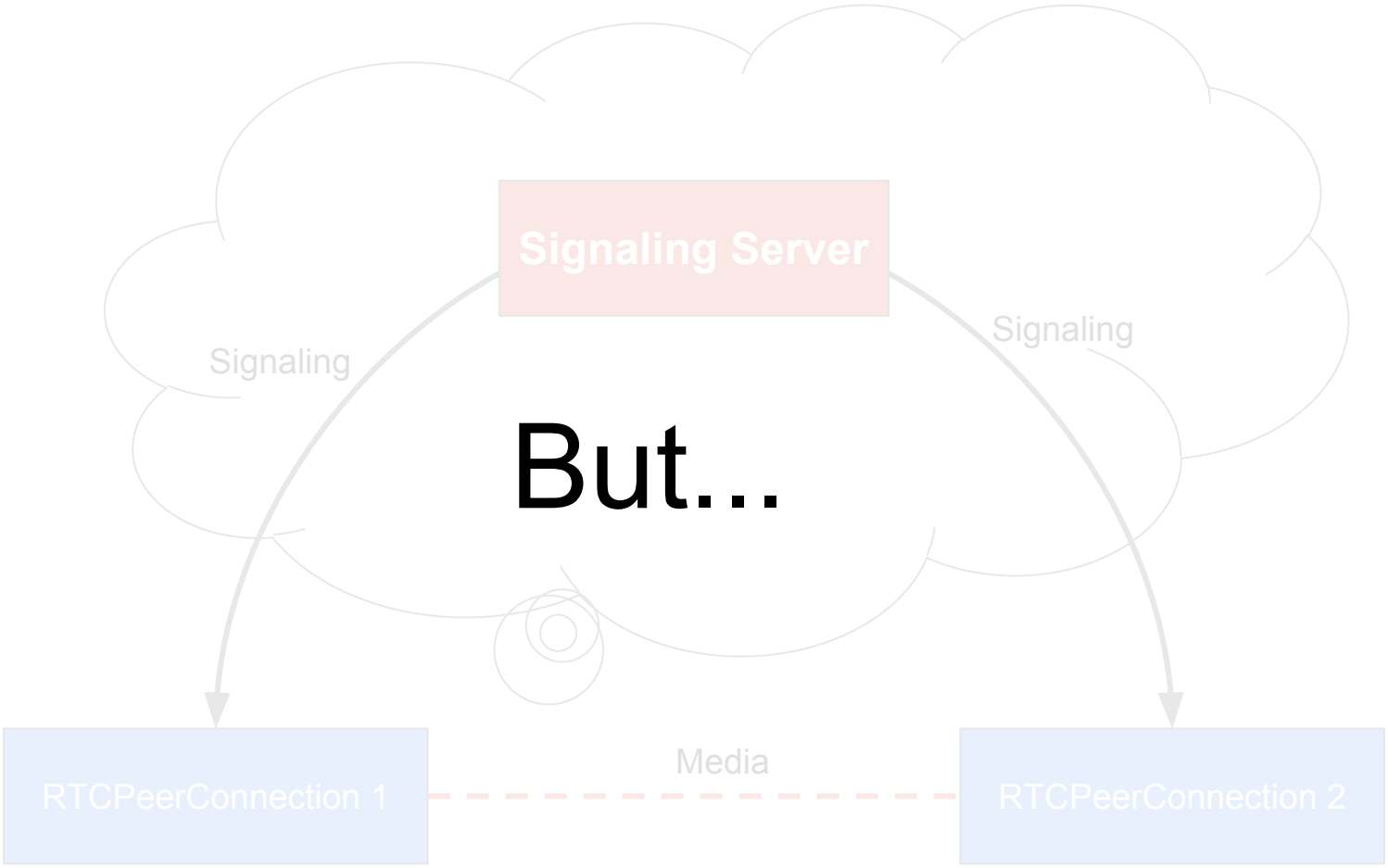

Internet
sdp2
sdp1
ICE candidate 2
ICE candidate 1
RTCPeerConnection 1
sdp1
sdp2
RTCPeerConnection 2
sdp2
sdp1

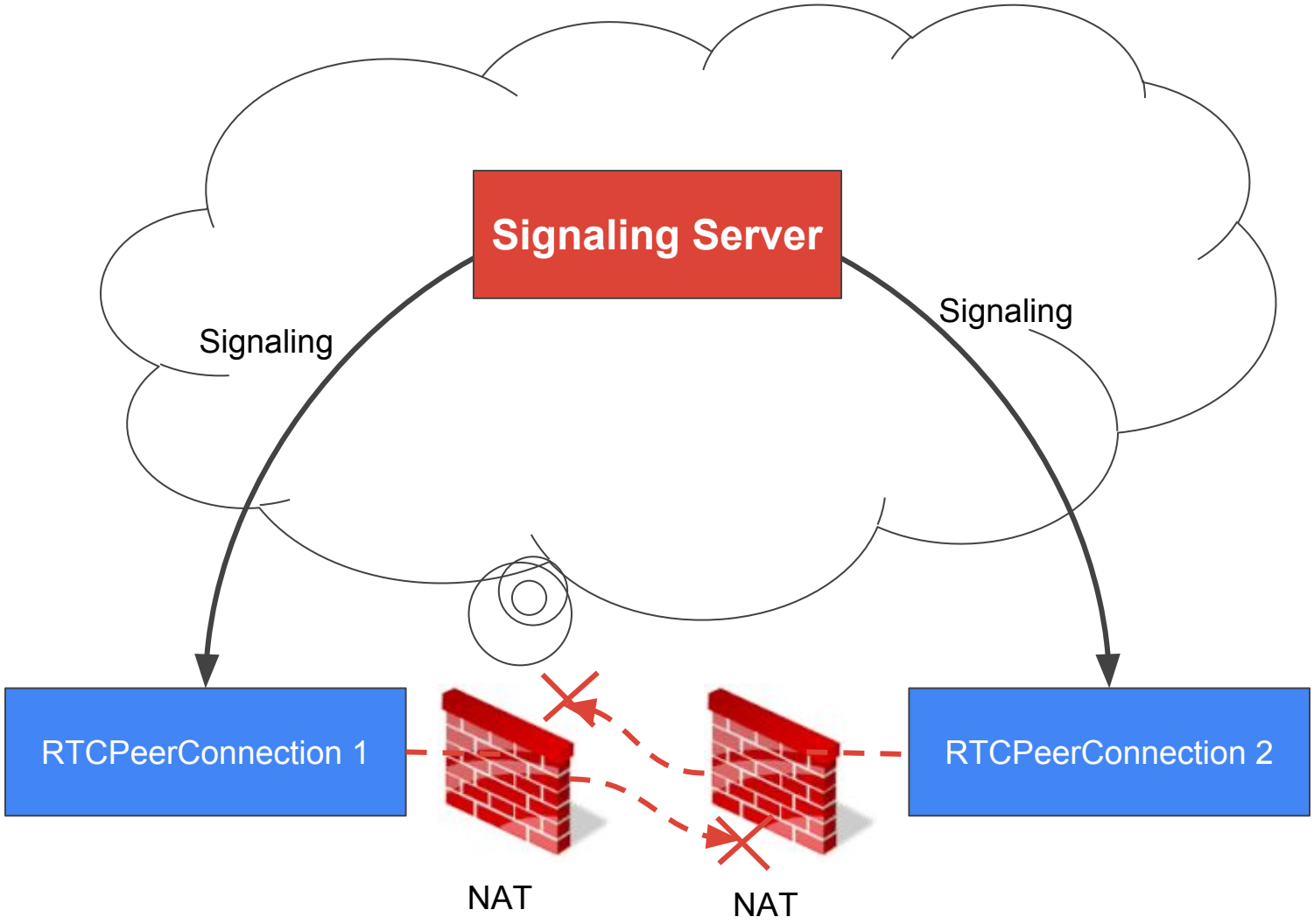

Server
RTCPeerConnection 1
RTCPeerConnection 2

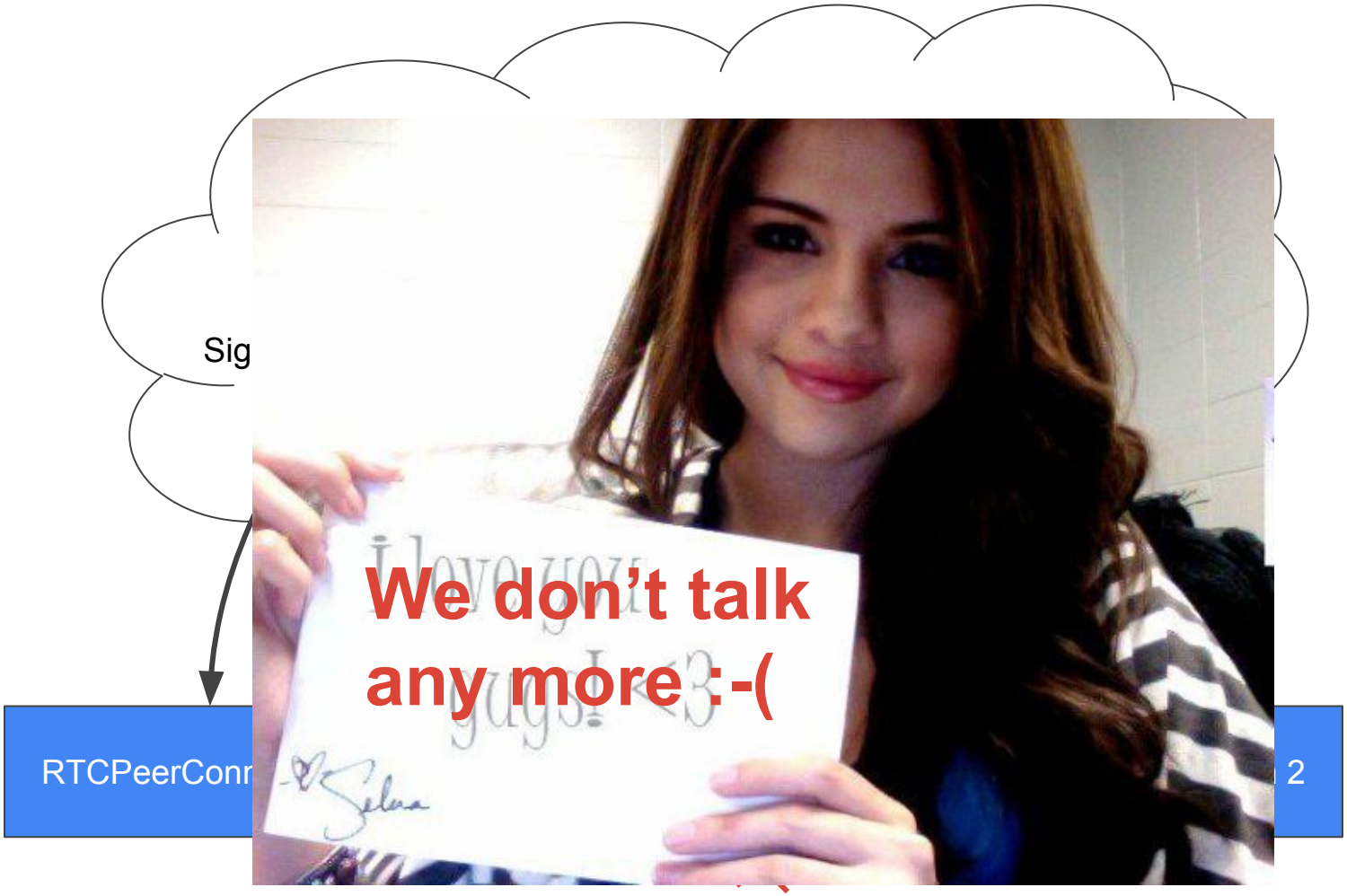

Signaling Server
RTCPeerConnection 1
RTCPeerConnection 2


Signaling Server
- Web Socket
- XHR + Long pulling
- Pub/Sub channel
- XMPP
- ...
RTCPeerConnection 1
RTCPeerConnection 2

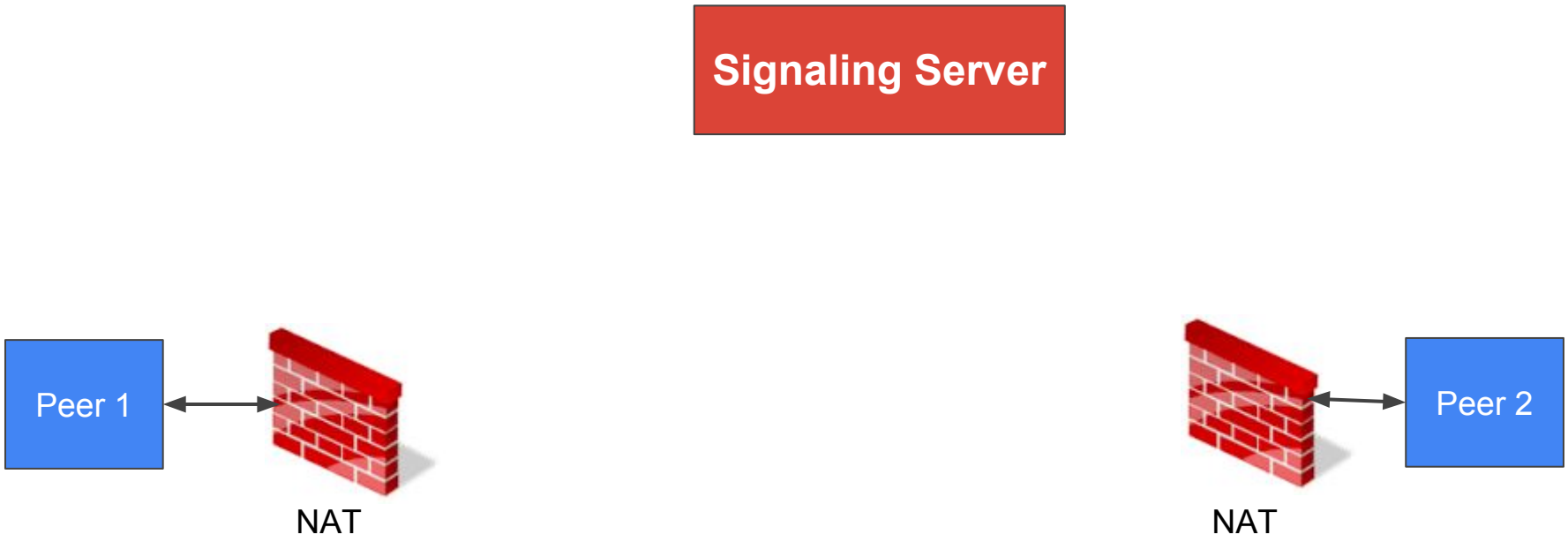

Signaling Server
Signaling
Signaling
RTCPeerConnection 1
Media
RTCPeerConnection 2


Signaling Server
Signaling
Signaling
But...
RTCPeerConnection 1
Media
RTCPeerConnection 2

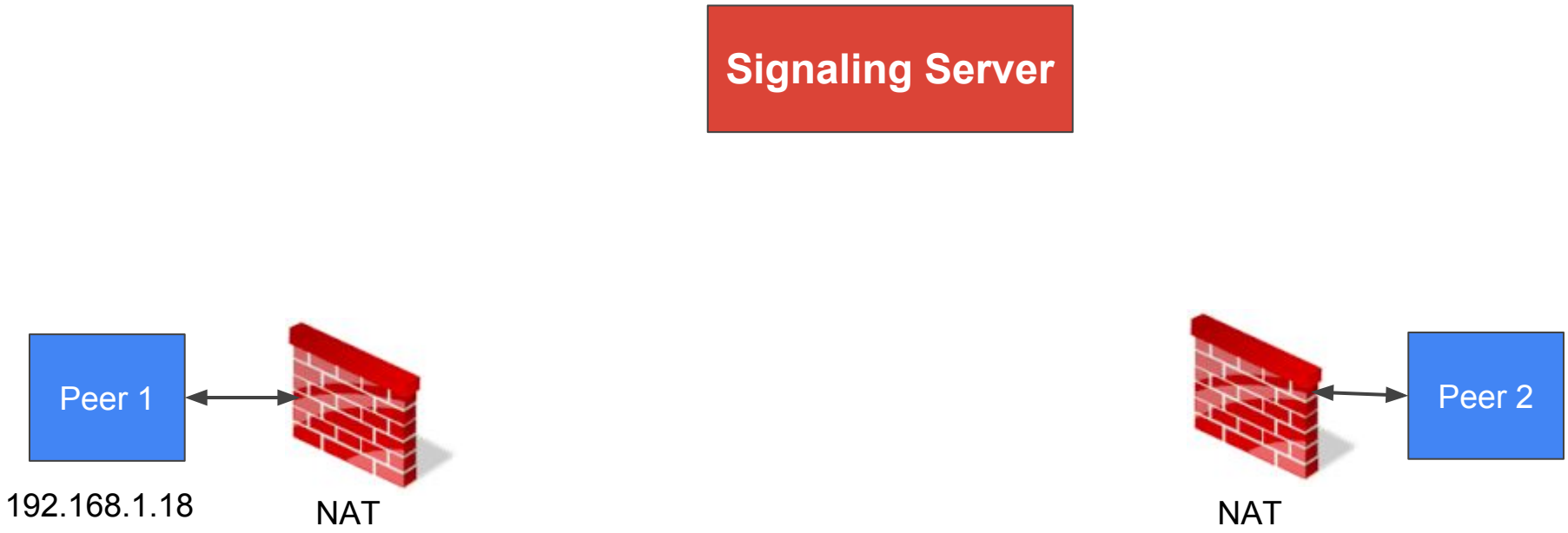

Signaling Server
Signaling
Signaling
RTCPeerConnection 1
RTCPeerConnection 2
NAT
NAT


Sig
RTCPeerConn
2
We don't talk
any more :-(

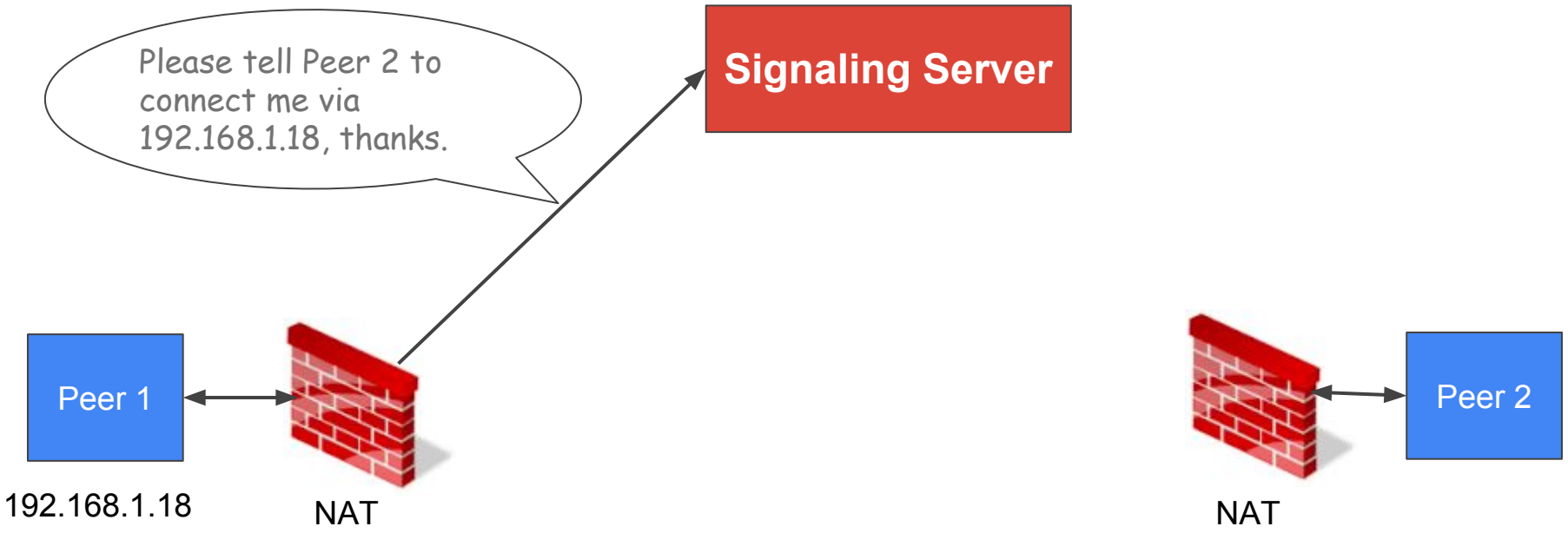
Why does it fail?



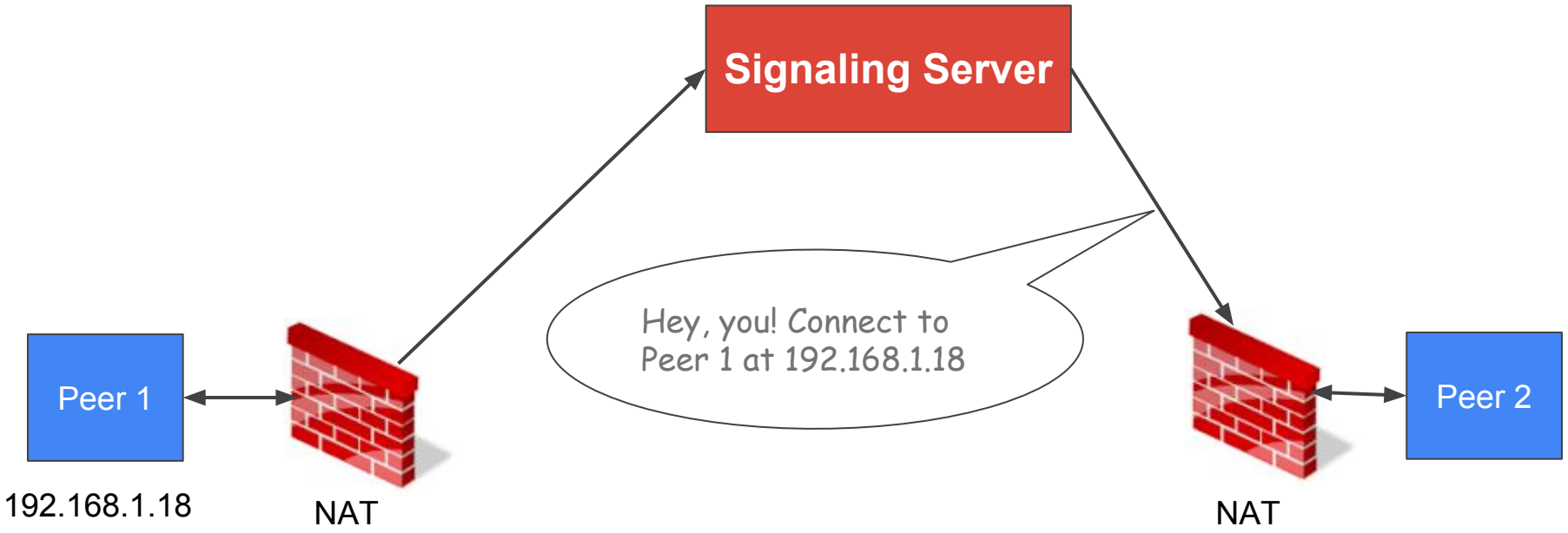
Why does it fail?



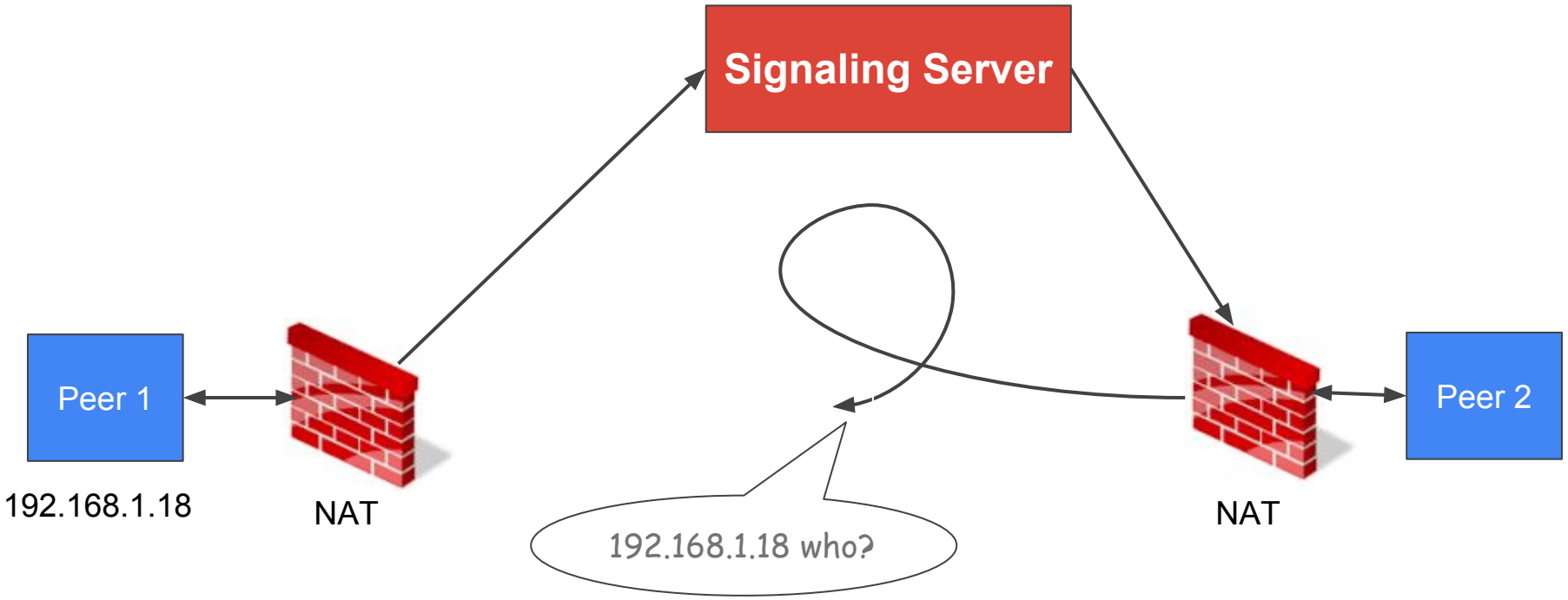
Why does it fail?



Why does it fail?

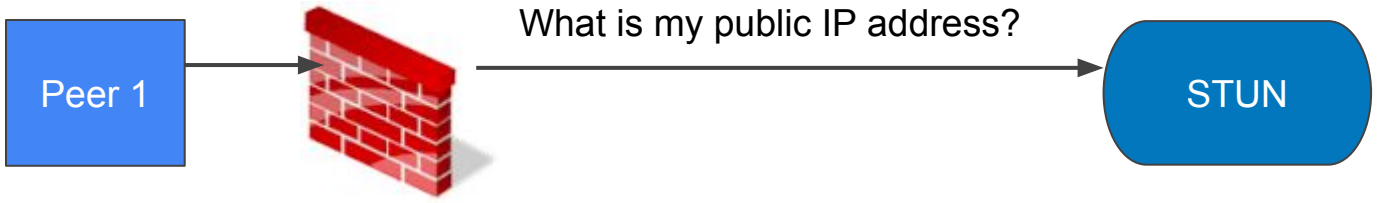


Why does it fail?

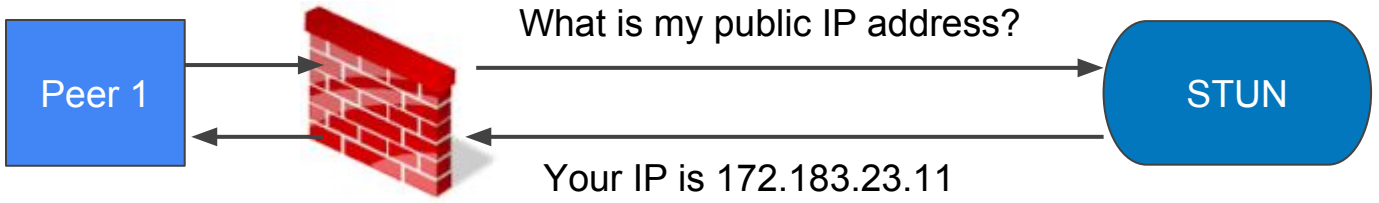



STUN server
Session Traversal Utilities for NAT (RFC 5389)

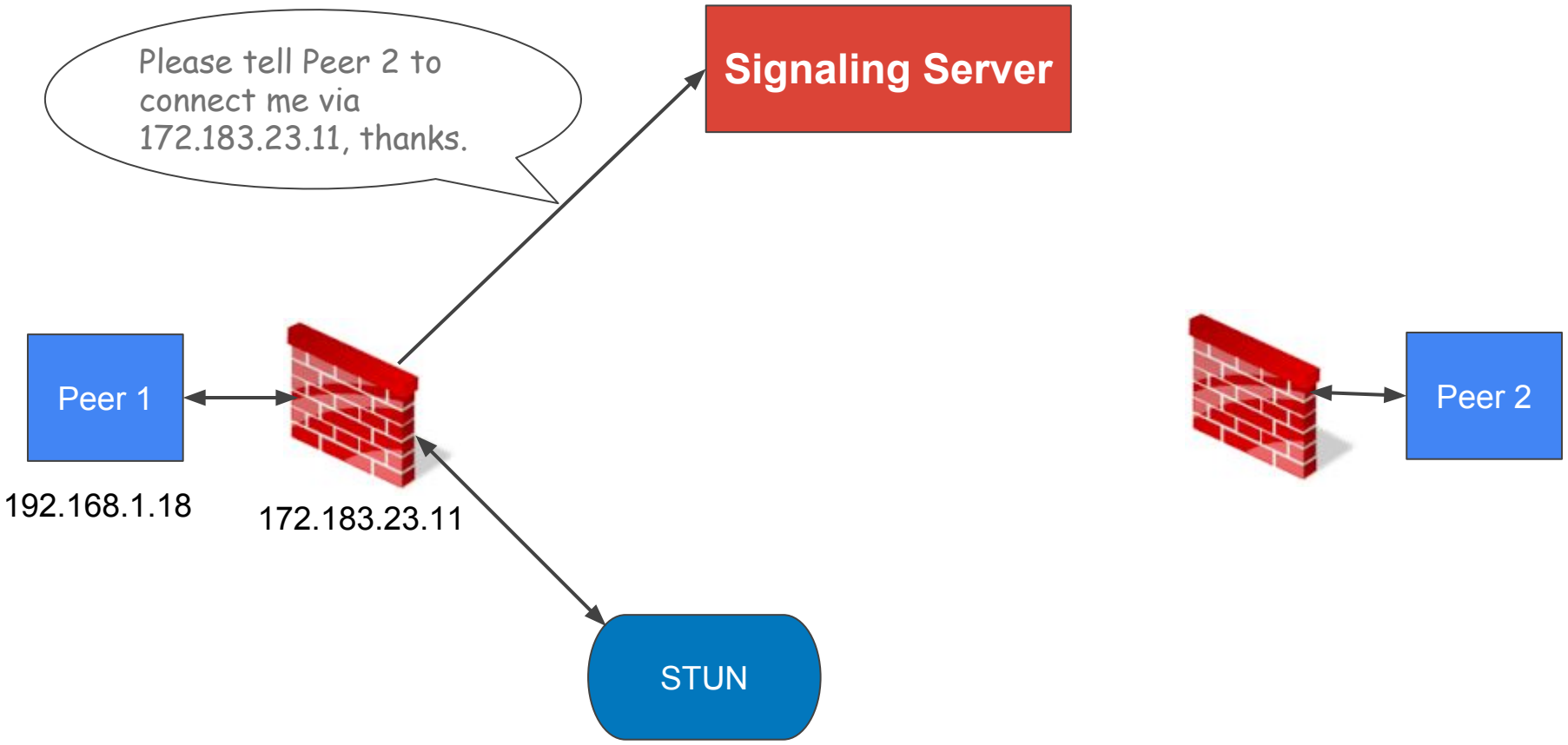
STUN server have one job:



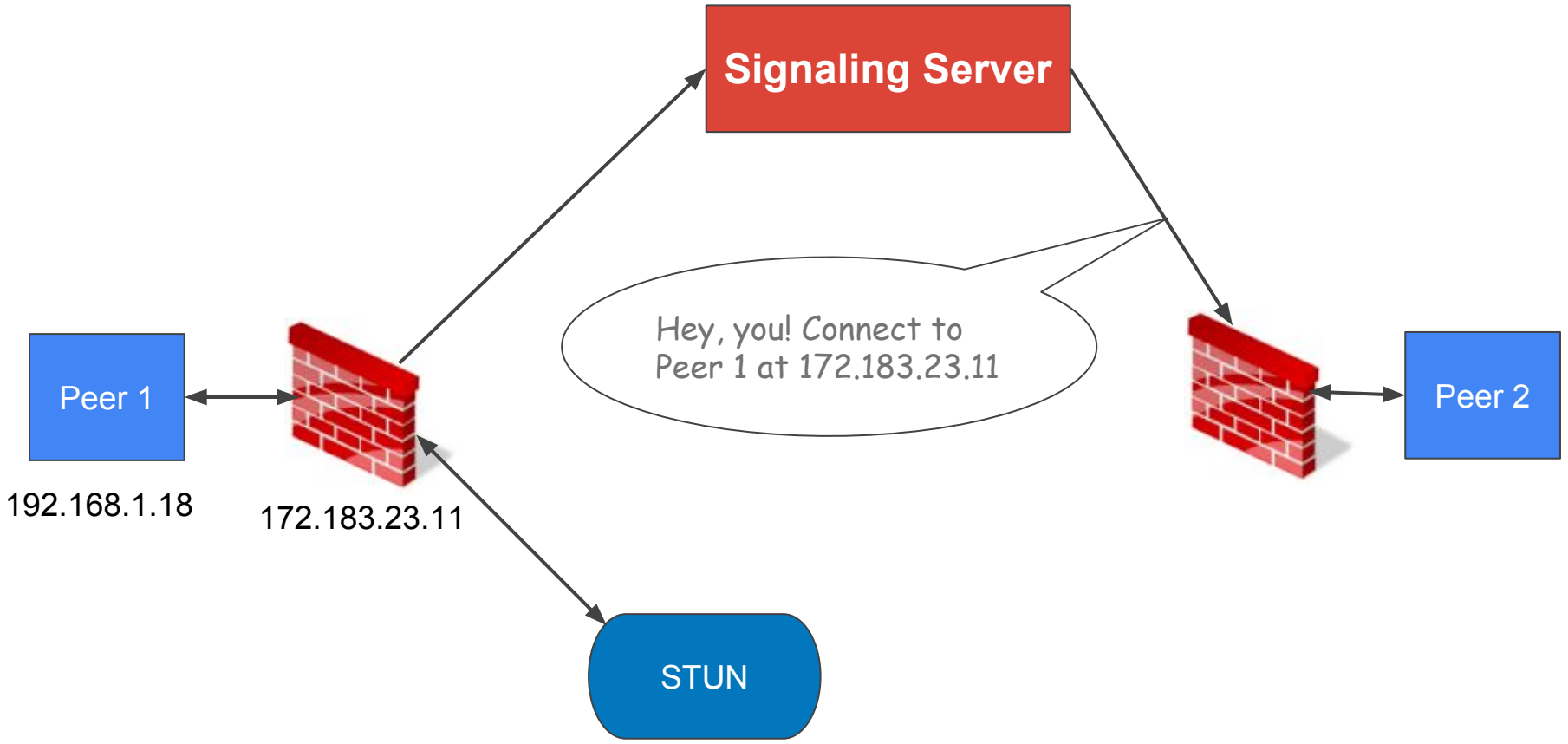
STUN server have one job:



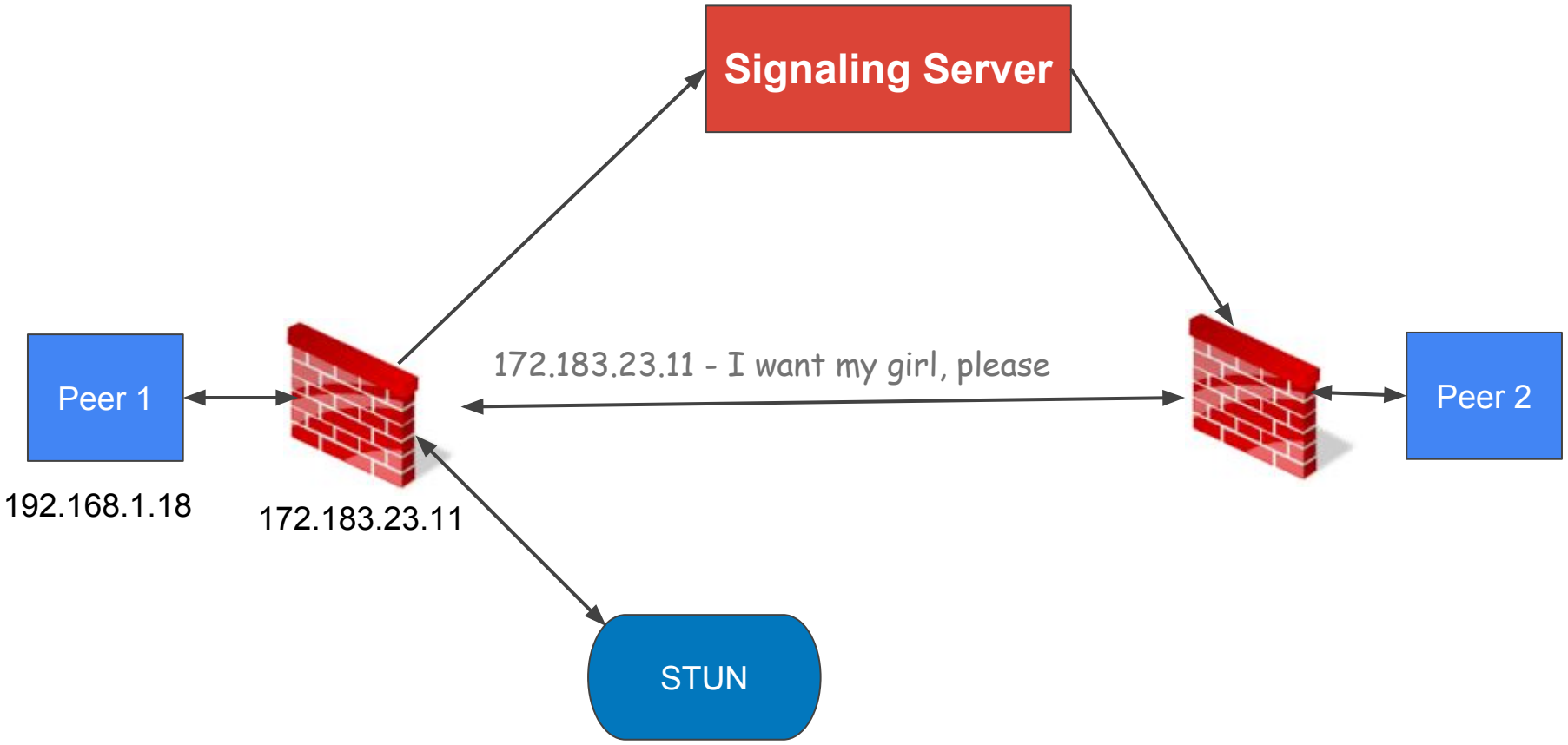
Use STUN servers to connect peers behind NATs



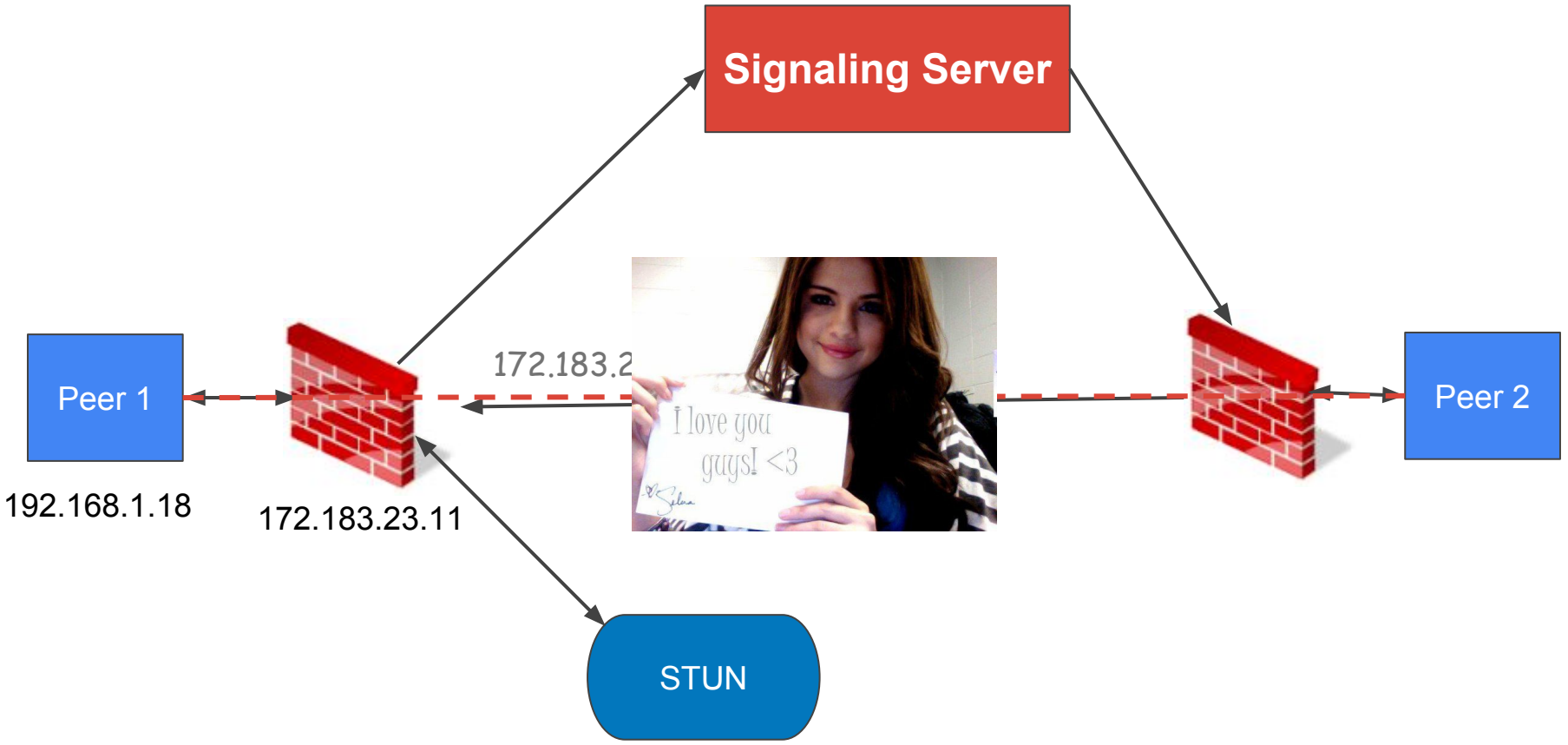
Use STUN servers to connect peers behind NATs



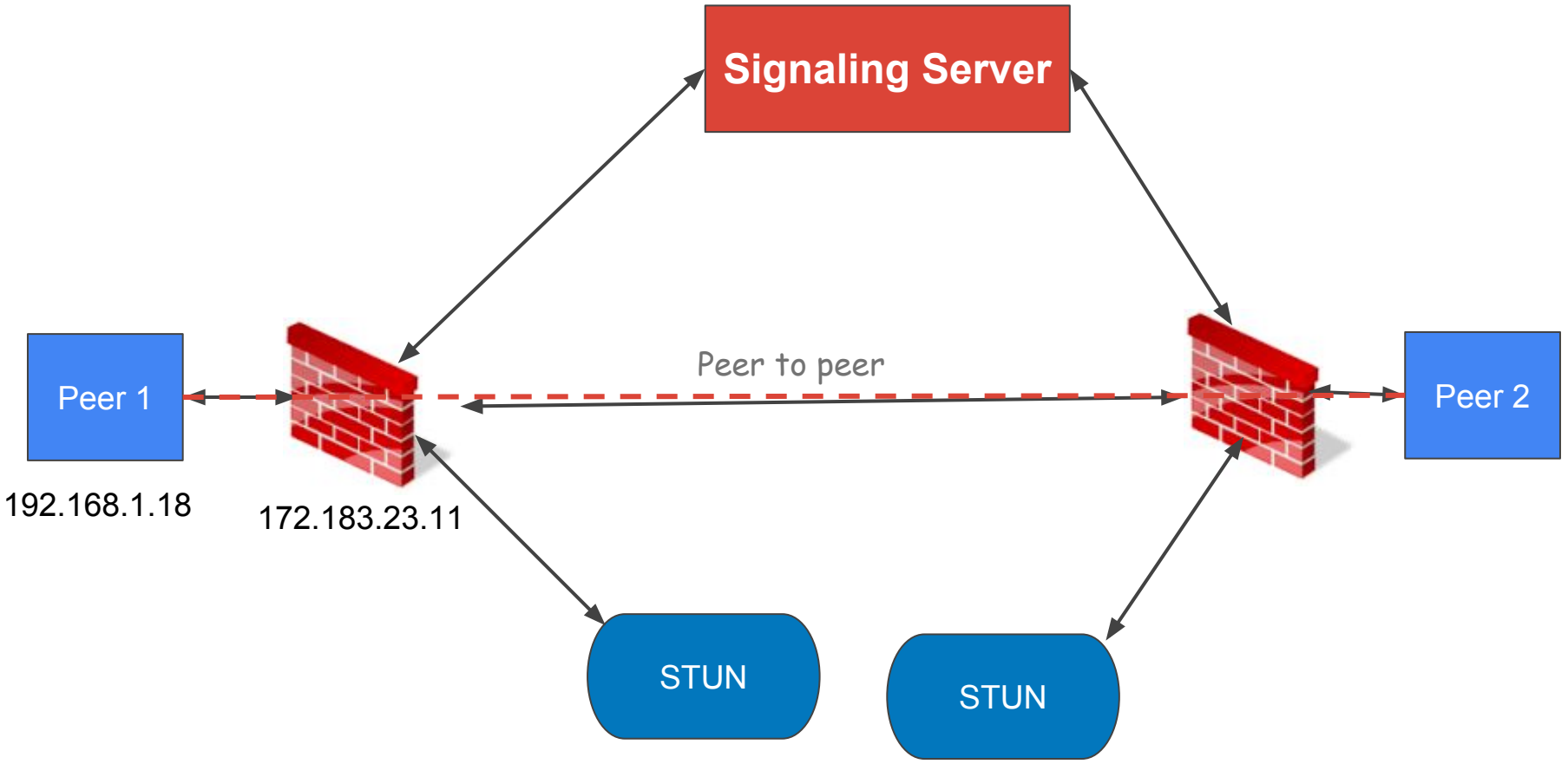
Use STUN servers to connect peers behind NATs



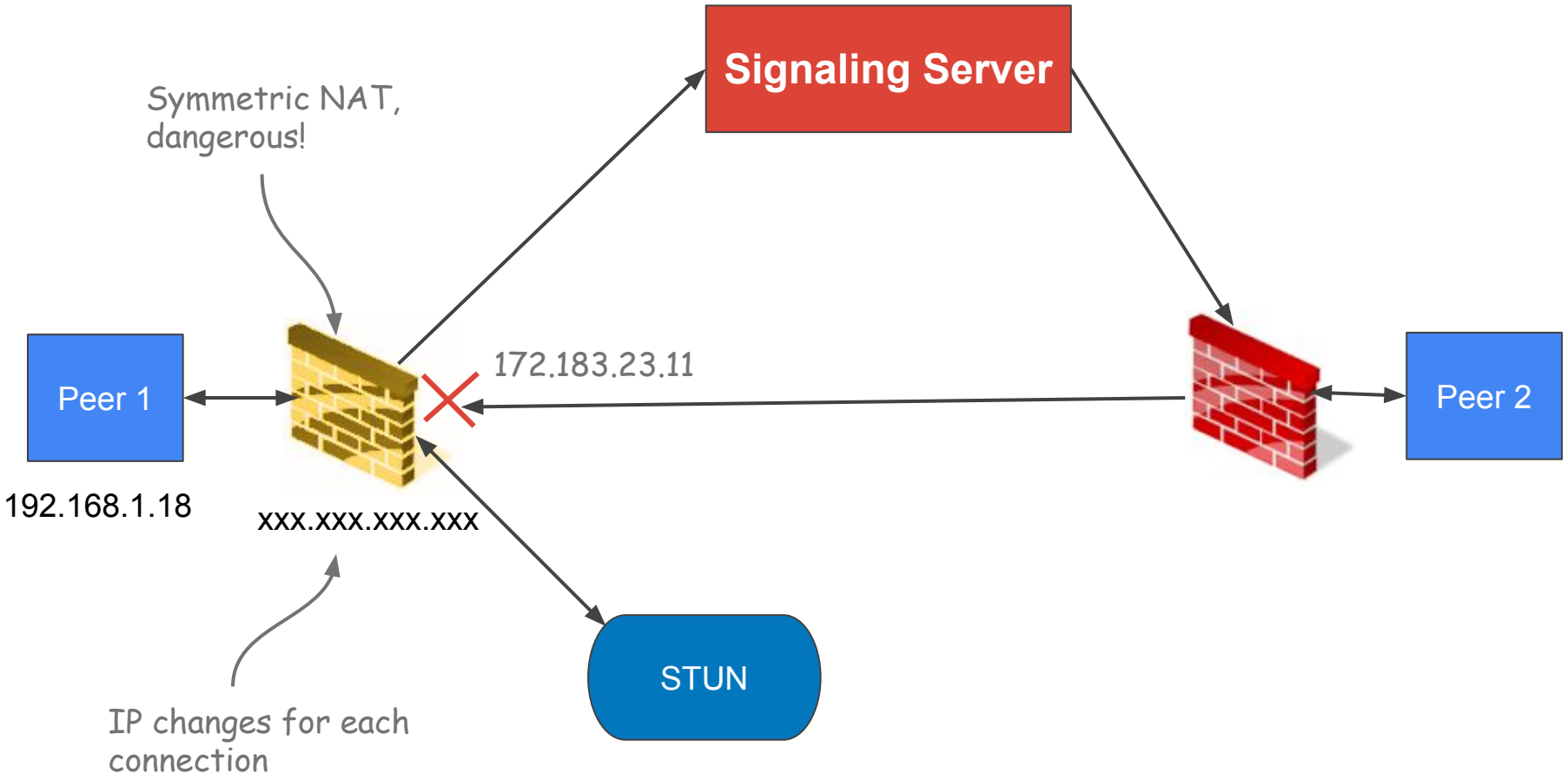
Use STUN servers to connect peers behind NATs



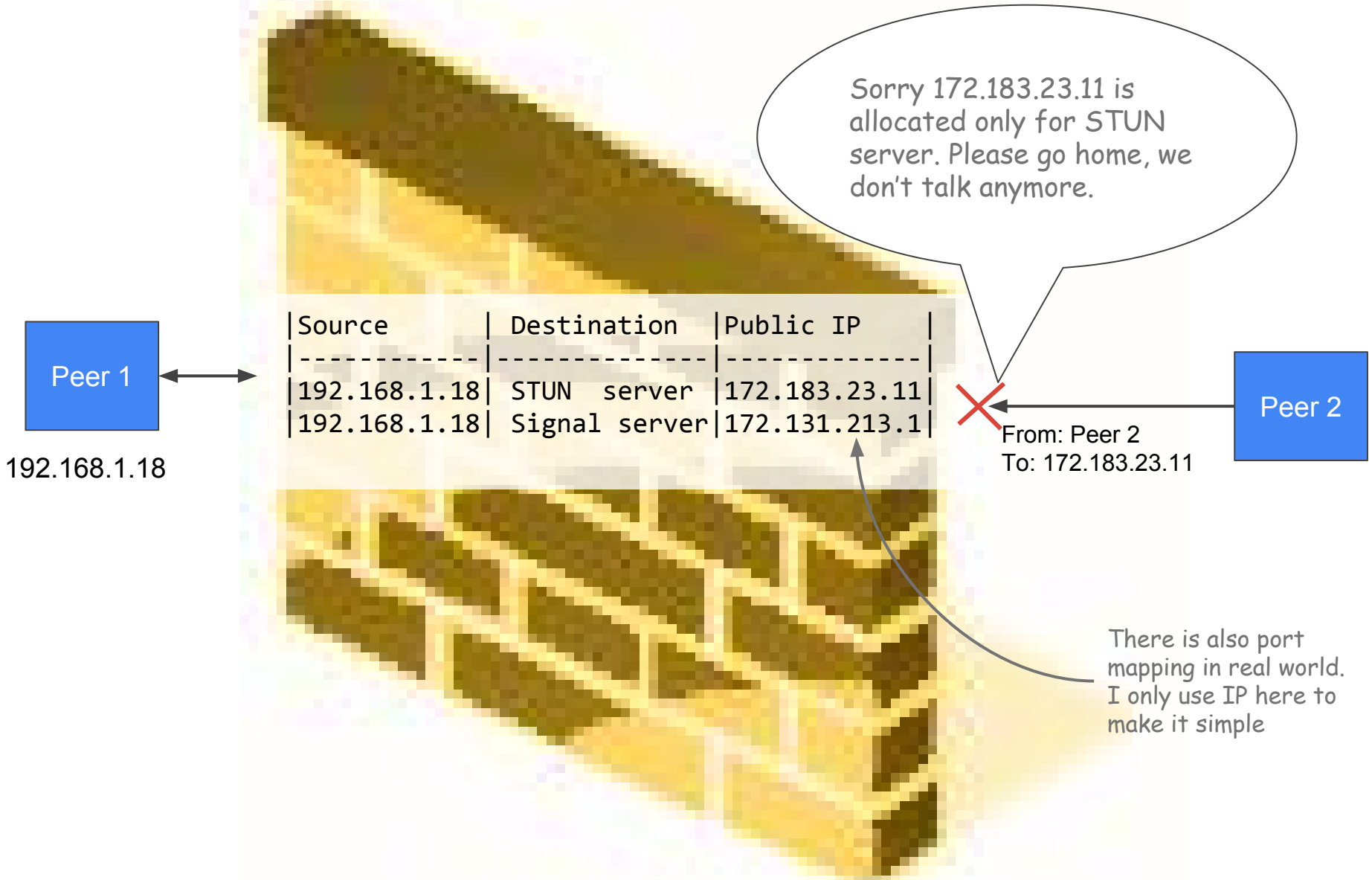
Use STUN servers to connect peers behind NATs



But sometimes it's still fail

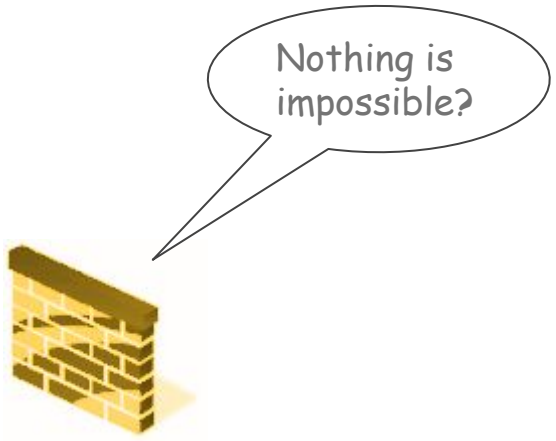

Signaling Server
Symmetric NAT,
dangerous!
Peer 1
192.168.1.18
172.183.23.11
xxx.xxx.xxx.xxx
Peer 2
STUN
IP changes for each
connection

Symmetric NAT

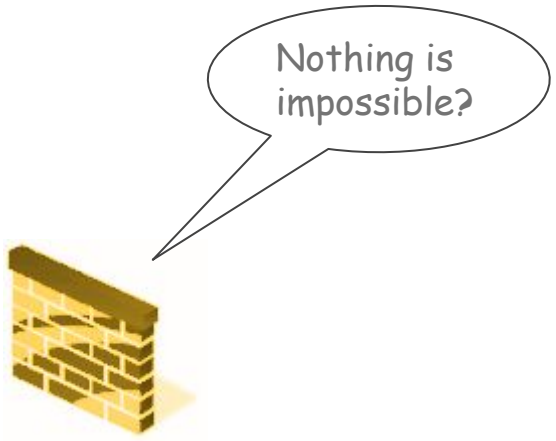


Don't believe me? → https://en.wikipedia.org/wiki/Network_address_translation#Methods_of_translation

In this case, it is impossible to establish a peer to peer connection



In this case, it is impossible to establish a peer to peer connection



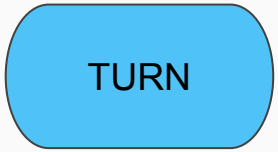
We have plan B

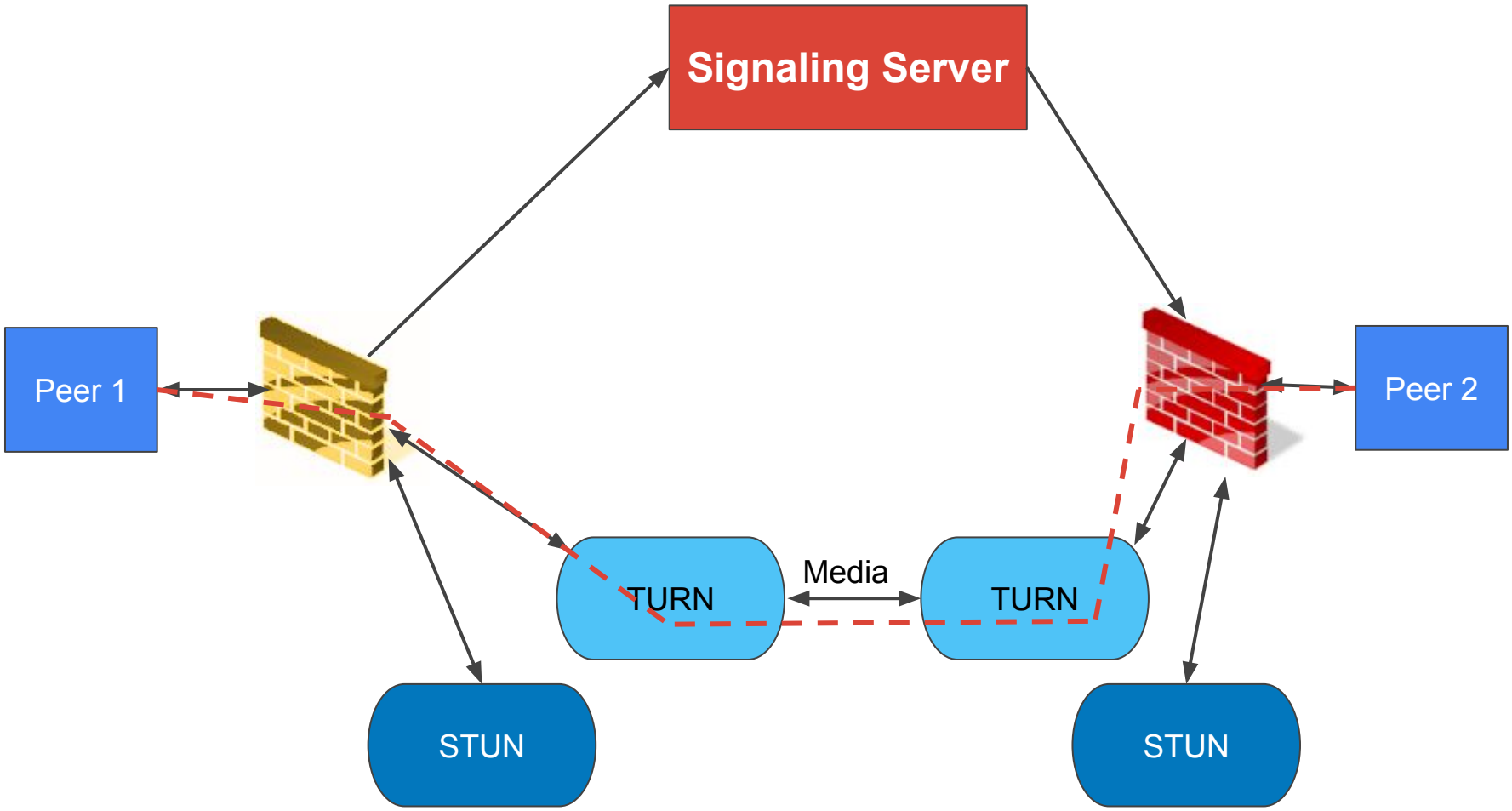
TURN server

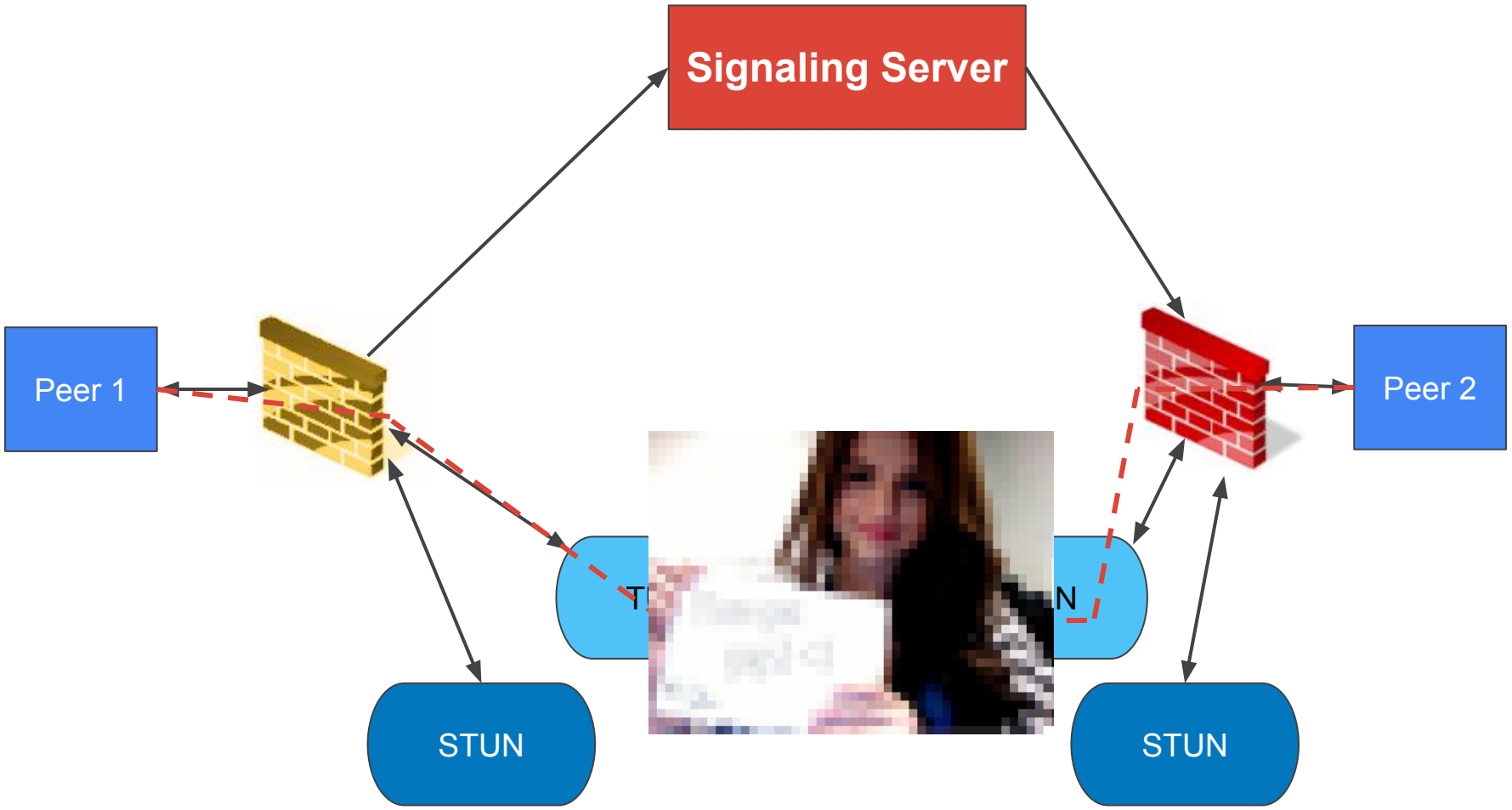
TURN: Traversal Using Relays around NAT (RFC 5766)

TURN server

TURN servers are used to relay traffic if direct (peer to peer) connection fails.




Signaling Server
Peer 1
Peer 2
TURN
Media
TURN
STUN
STUN


Signaling Server
Peer 1
Peer 2
STUN
STUN

Well…

That’s a lot of WebRTC!

Ready to see all the code?

JAVASCRIPT

```javascript
var constraints = {
  audio: false,
  video: true
};

var video1 = document.querySelector('#video1');
var video2 = document.querySelector('#video2');

var videoStream;

function successCallback(stream) {
    // Show to <video> tag
    videoStream = stream;
    video1.srcObject = videoStream;

    startCalling();
}

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

navigator.webkitGetUserMedia(constraints, successCallback, errorCallback);

function startCalling() {
  var pc1 = new webkitRTCPeerConnection(null);
  var pc2 = new webkitRTCPeerConnection(null);

  pc1.addStream(videoStream);
}
```

```
function startCalling() {
  var pc1 = new webkitRTCPeerConnection(null);
  var pc2 = new webkitRTCPeerConnection(null);
```

JAVASCRIPT

```javascript
var servers = {
   "iceServers":[
      {
         "url":"stun:turn02.uswest.xirsys.com"
      },
      {
         "username":"7d4288ec-6578-11e6-9134-544d095e7044",
         "url":"turn:turn02.uswest.xirsys.com:443?transport=udp",
         "credential":"7d4289b4-6578-11e6-8007-f9fac44b5c9c"
      }
   ]
}

var pc1 = new webkitRTCPeerConnection(servers);
```

JAVASCRIPT

```
var servers = {
   "iceServers":[
      {
         "url":"stun:turn02.uswest.xirsys.com"
      },
      {
         "username":"7d4288ec-6578-11e6-9134-544d095e7044",
         "url":"turn:turn02.uswest.xirsys.com:443?transport=udp",
         "credential":"7d4289b4-6578-11e6-8007-f9fac44b5c9c"
      }
   ]
}

var pc1 = new webkitRTCPeerConnection(servers);
```

That's it 

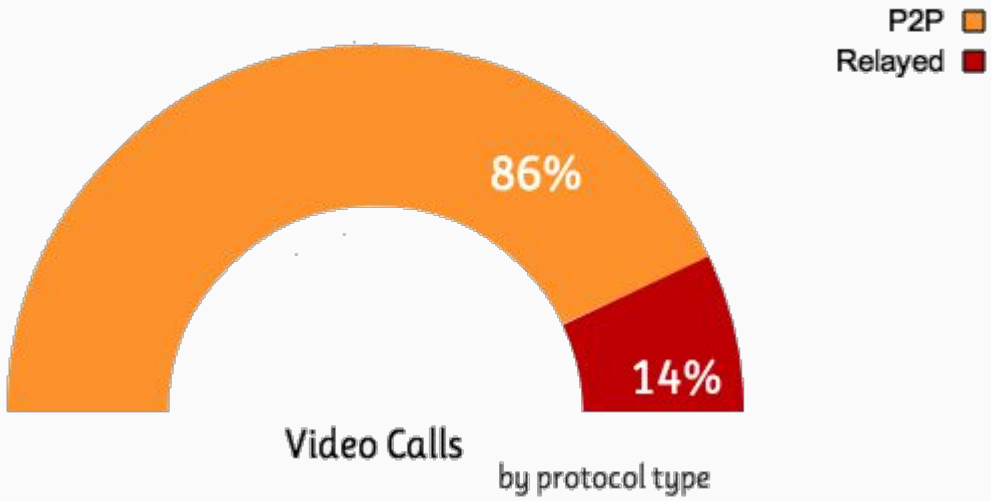
JAVASCRIPT

```javascript
var servers = {
   "iceServers":[
      {
         "url":"stun:turn02.uswest.xirsys.com"
      },
      {
         "username":"7d4288ec-6578-11e6-9134-544d095e7044",
         "url":"turn:turn02.uswest.xirsys.com:443?transport=udp",
         "credential":"7d4289b4-6578-11e6-8007-f9fac44b5c9c"
      }
   ]
}

var pc1 = new webkitRTCPeerConnection(servers);
```

ICE Framework finds the best way to create a peer-to-peer connection.

ICE Framework:

- Try to create a peer-to-peer connection via UDP
- If that fail, try TCP: first HTTP, then HTTPS
- If that still fail, use TURN server



Don’t worry, most of the time there will be a peer-to-peer connection created.

Where can I get my STUN/TURN server?

Do it by yourself: deploy an open-sourced STUN/TURN server

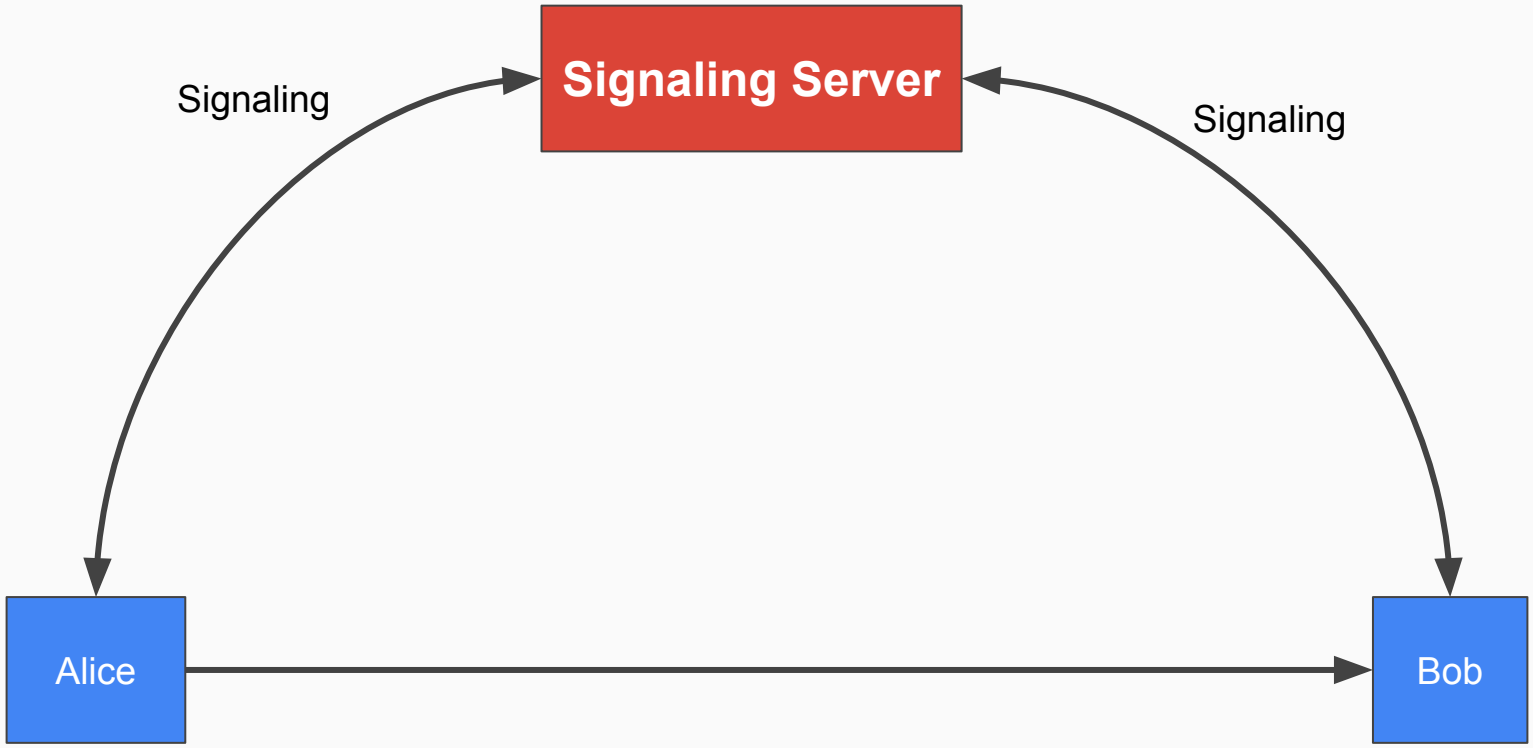
https://github.com/coturn/coturn
(maintained by Google)

Or use a service:

- https://www.twilio.com/stun-turn
- http://numb.viagenie.ca/
- https://xirsys.com/

Can we run the demo yet?

Alice want to share her webcam to Bob



Signaling Server with NodeJS and SocketIO

```
84   // 3. Alice send Bob an offer
85   socket.on('alice-offer-a-session', function (data) {
86     console.log('Alice sent an offer to ', data.bobName);
87     bobList[data.bobName].emit('alice-offer-a-session', data.sessionDescription);
88   });
89
90   // 4. Bob answer the offer
91   socket.on('bob-answer-the-offer', function (sessionDescription) {
92     // Deliver the answer to Alice
93     var bob = 'Bob-' + socket.id;
94     alice.emit('bob-answer-the-offer', {
95       bobName: bob,
96       sessionDescription: sessionDescription,
97     });
98     console.log(bob, ' answered the offer. Told Alice about that.');
99   });
100
101  // 5. Alice send ICE candidate info
102  socket.on('alice-sending-ice-candidate', function (data) {
103    // Deliver the candidate to all Bobs
104    console.log('Alice sent an ICE candidate to ', data.bobName);
105    bobList[data.bobName].emit('alice-sending-ice-candidate', data.candidate);
106  });
```

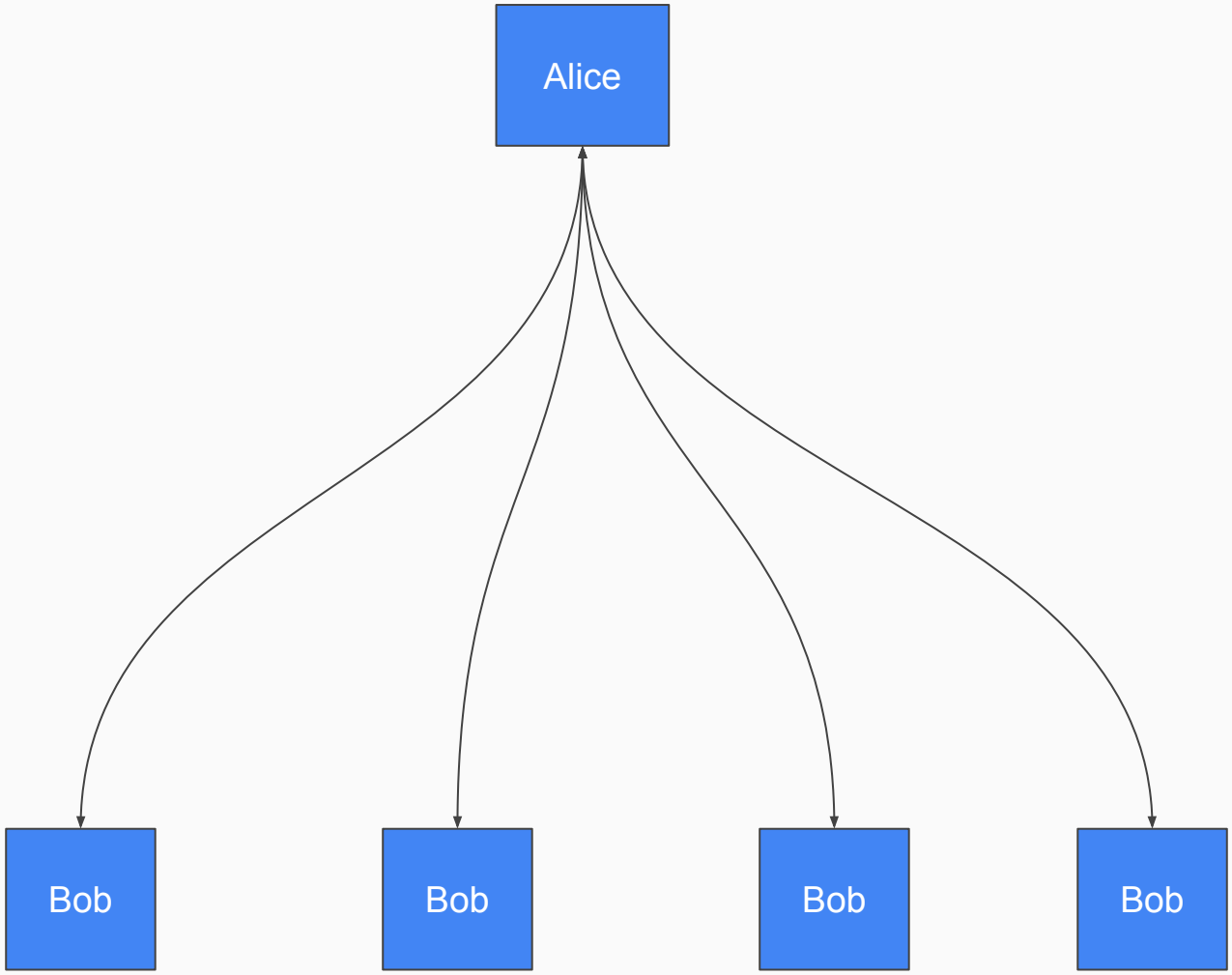
https://github.com/trungdq88/webrtc-demo/blob/master/signalling-server.js

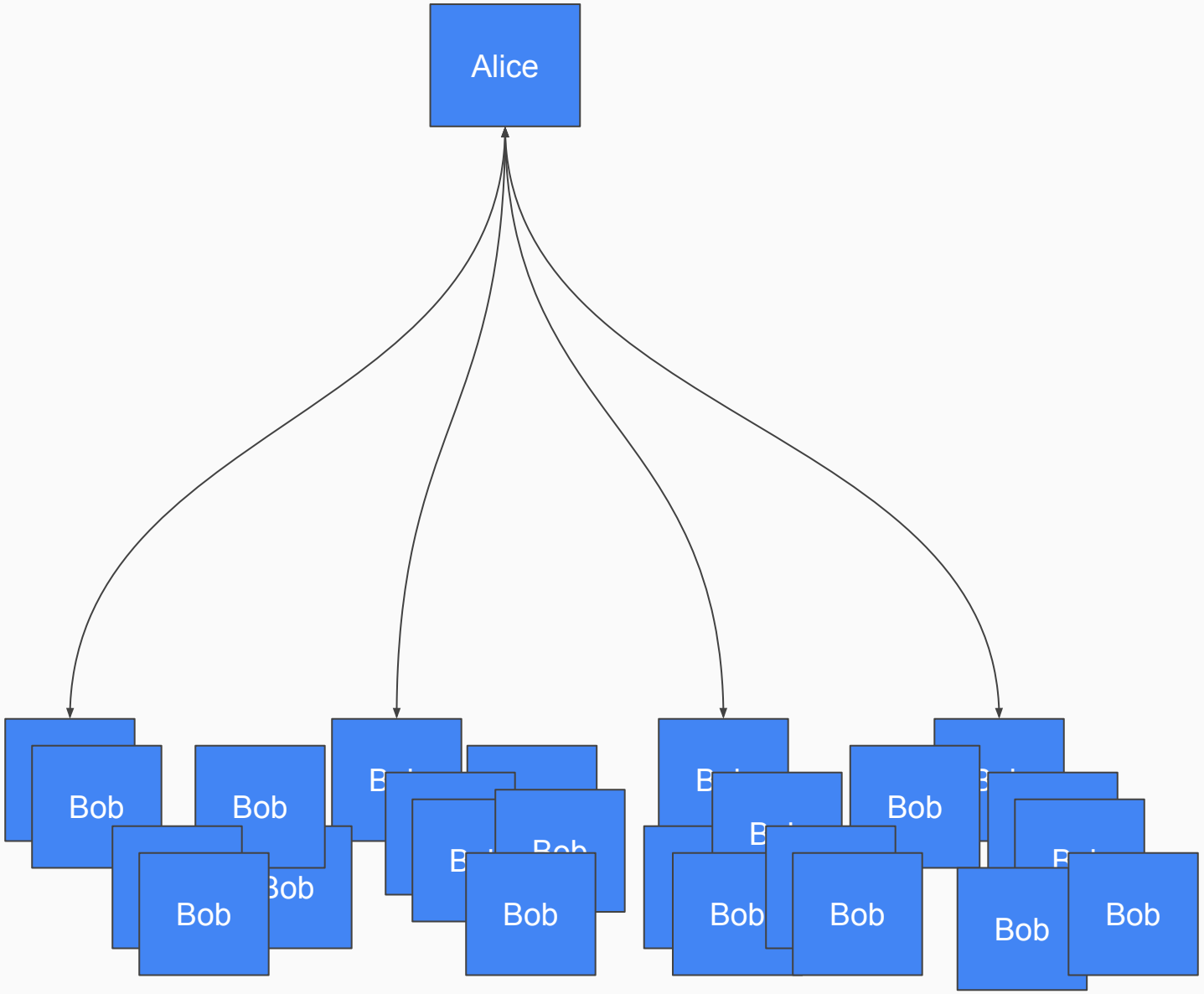
http://bit.ly/rtc-demo

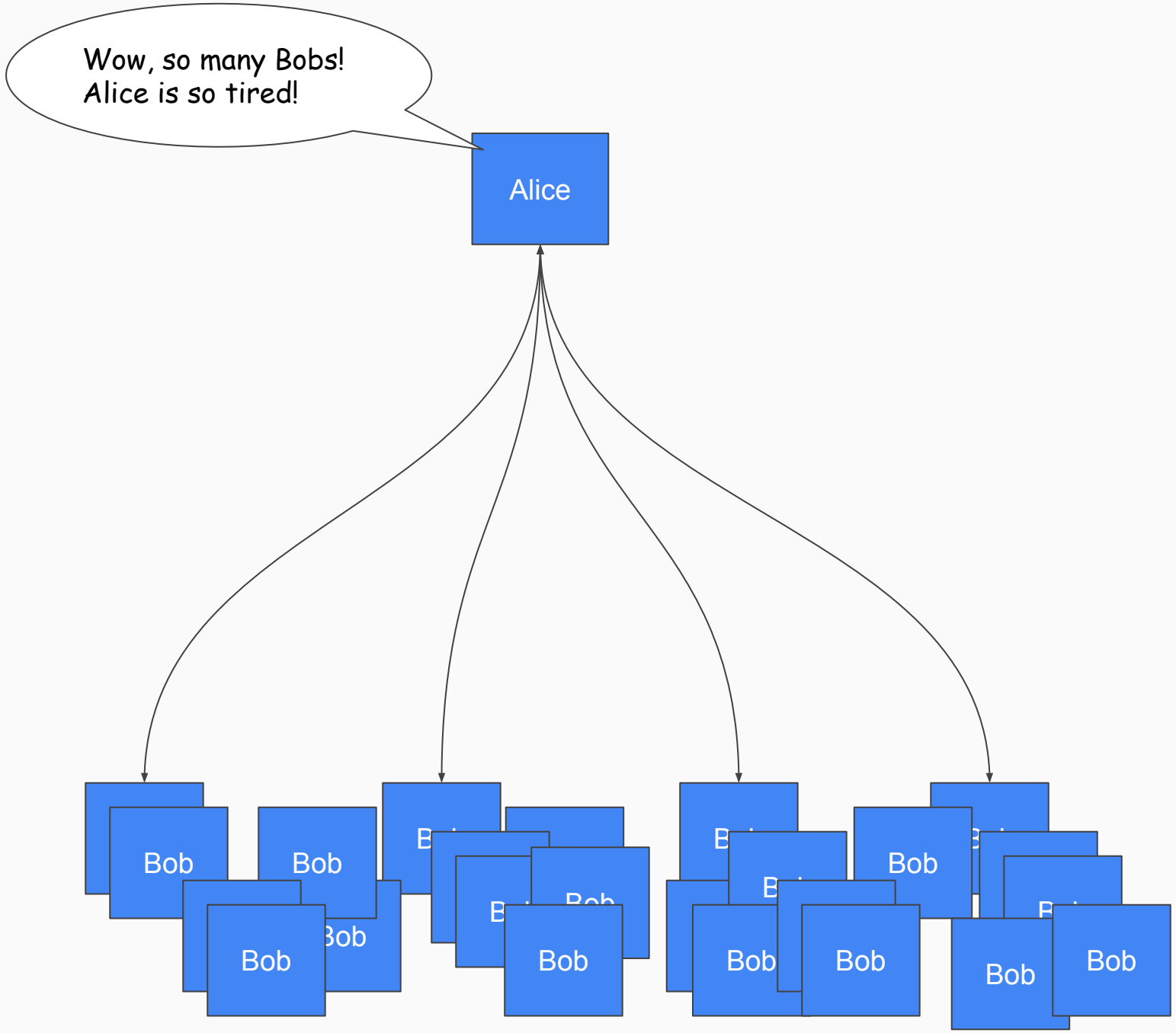
http://bit.ly/rtc-demo

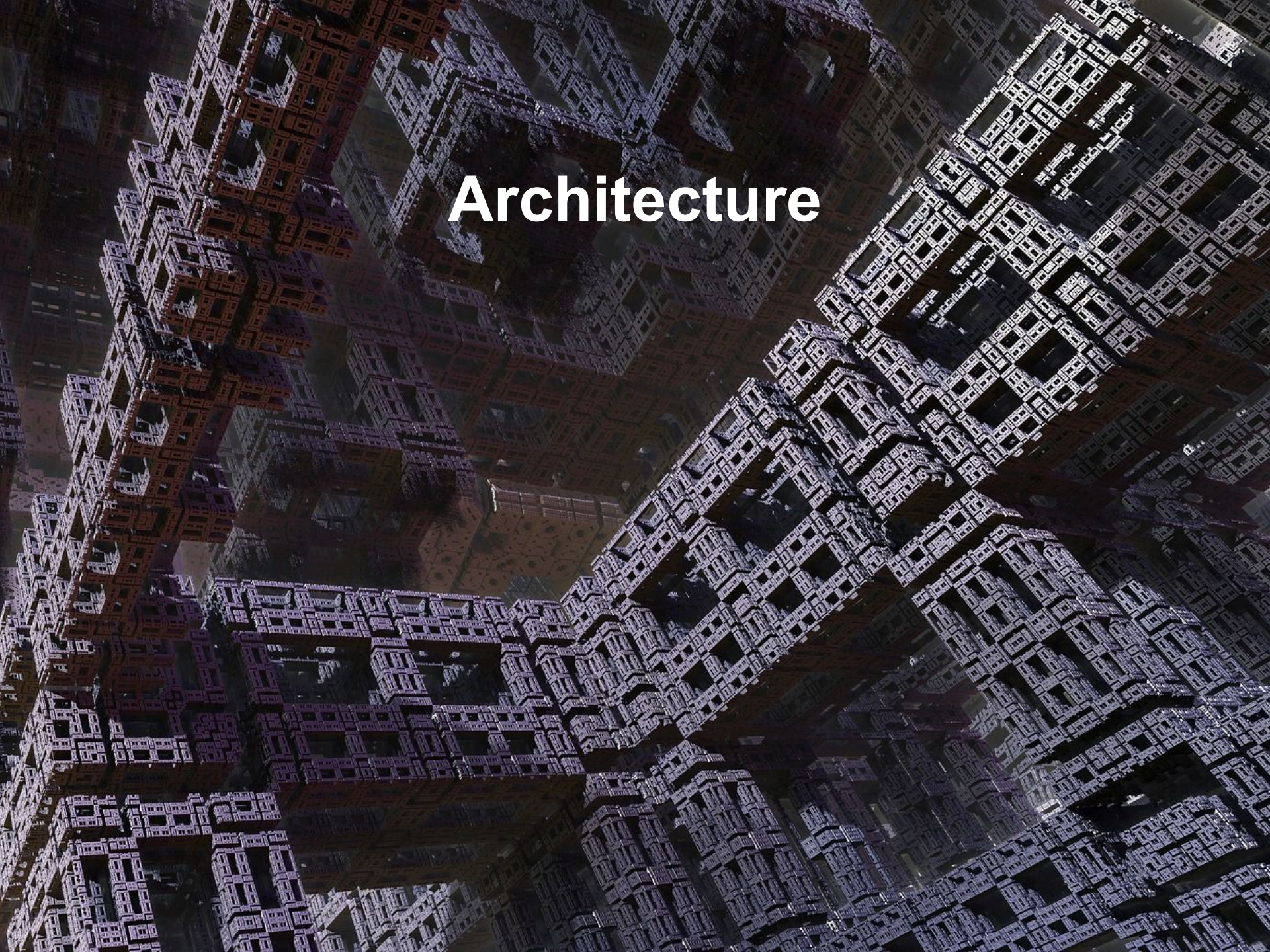
Homework:

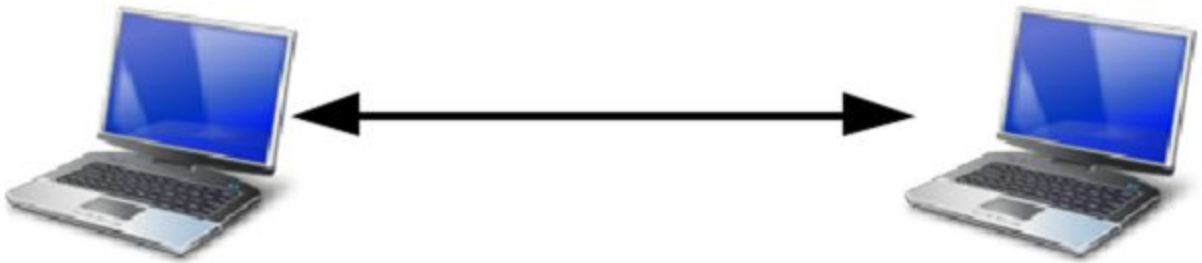
Alice want to see Bob too!


Alice
Bob
Bob
Bob
Bob

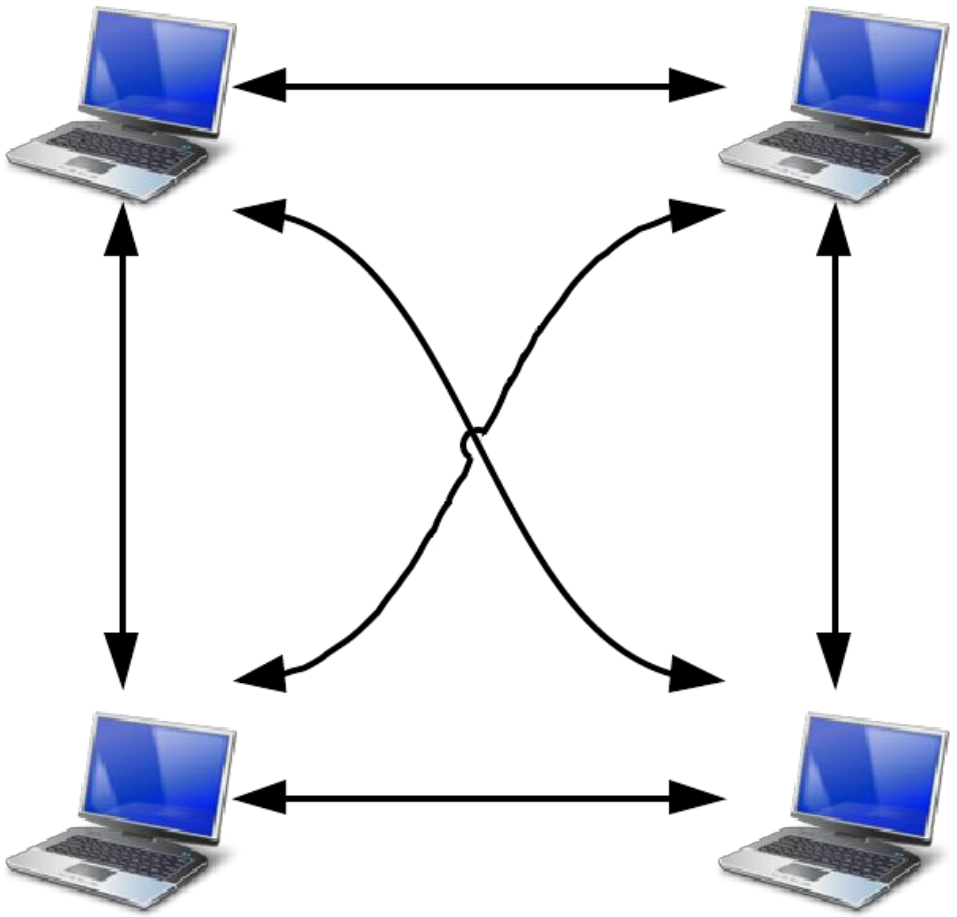

Alice
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob


Wow, so many Bobs!
Alice is so tired!
Alice
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob
Bob

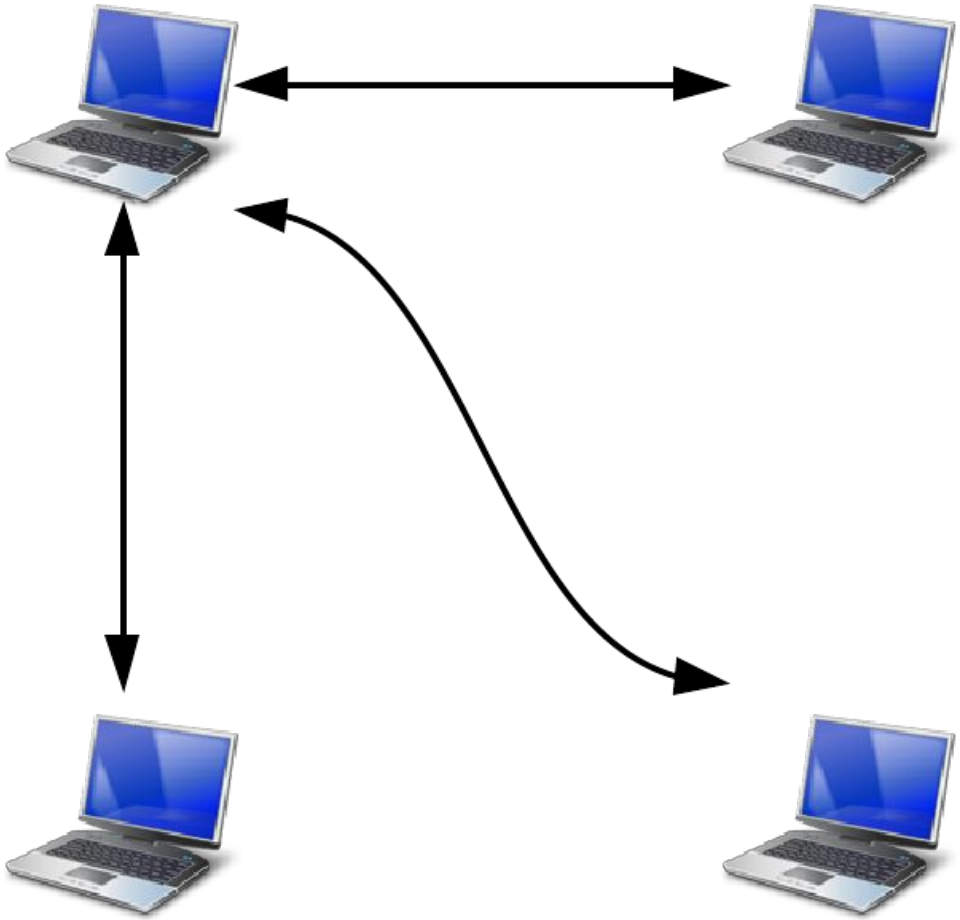

Architecture



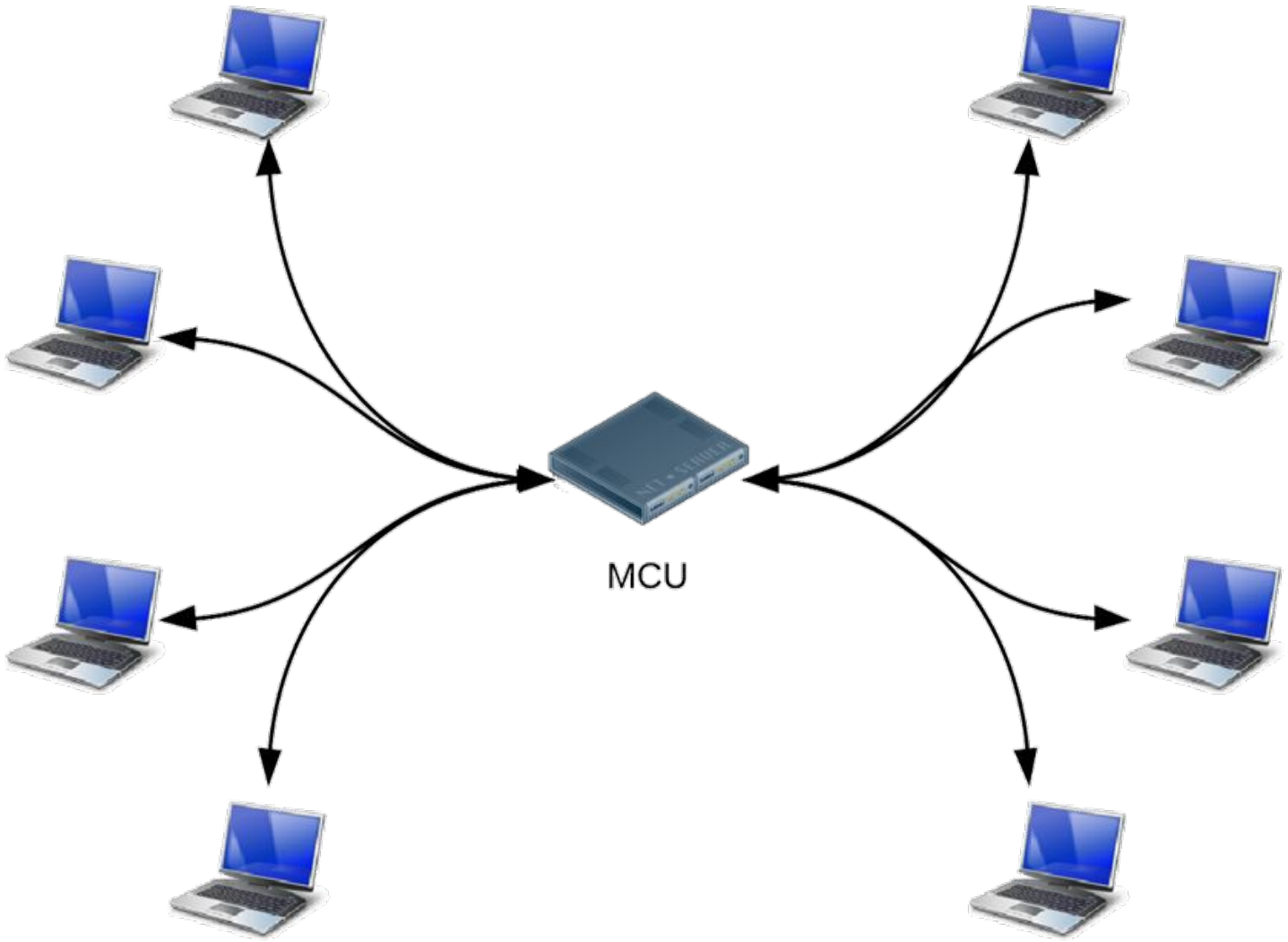
Peer to peer: one-to-one call



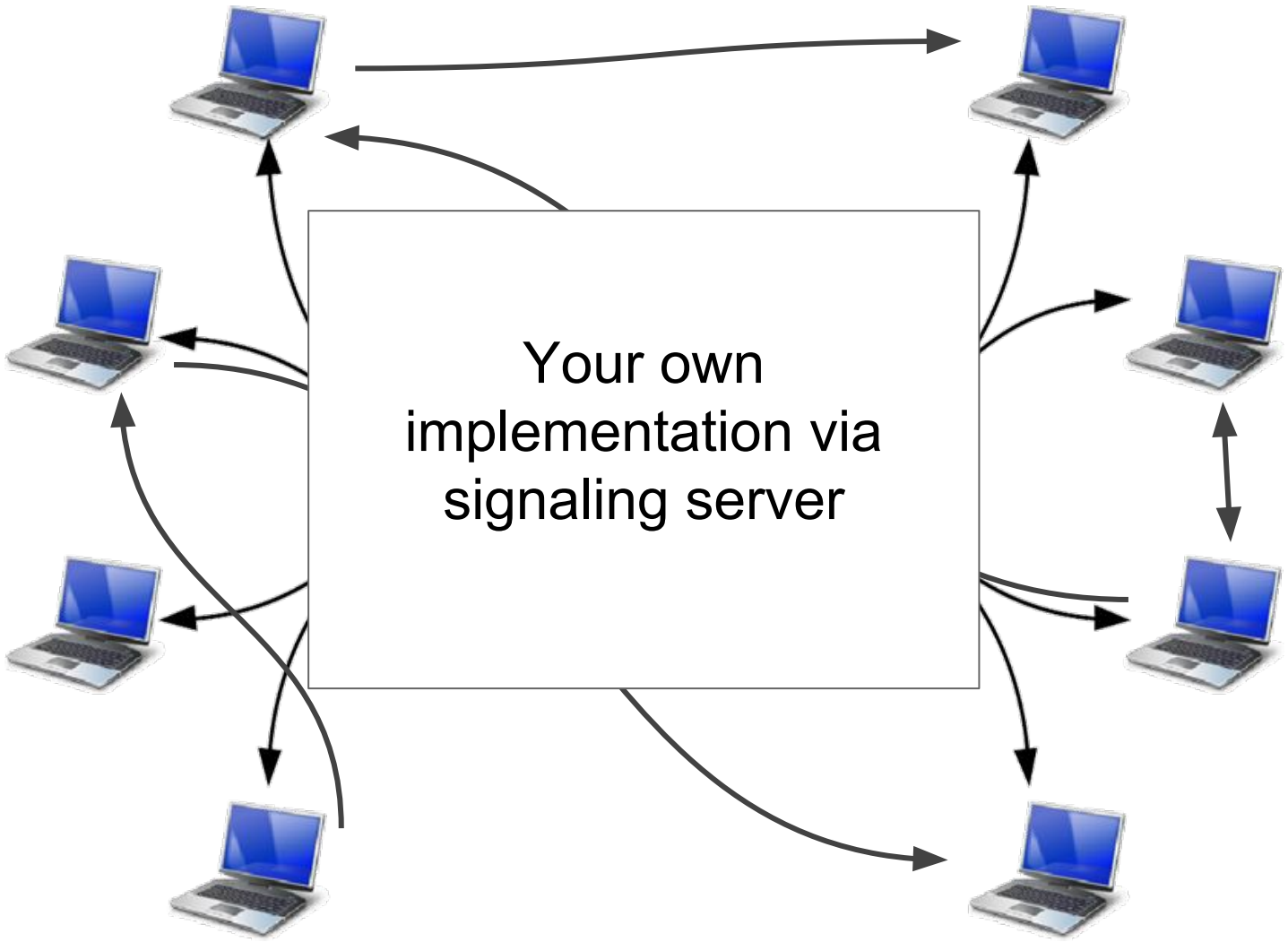
Mesh: small N-way call



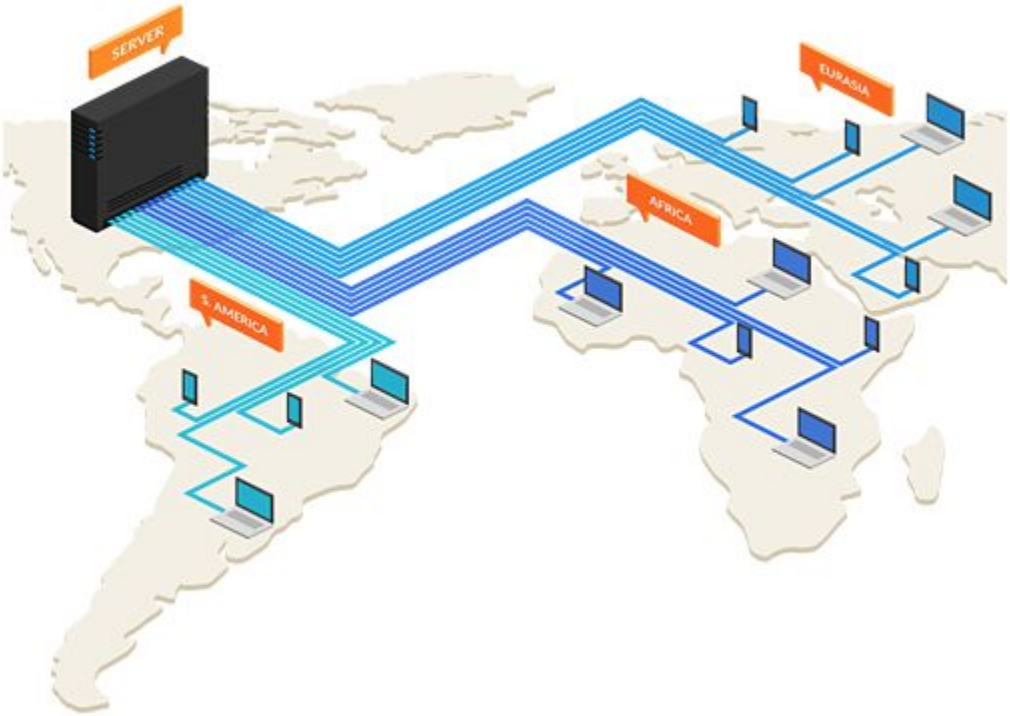
Star: medium N-way call



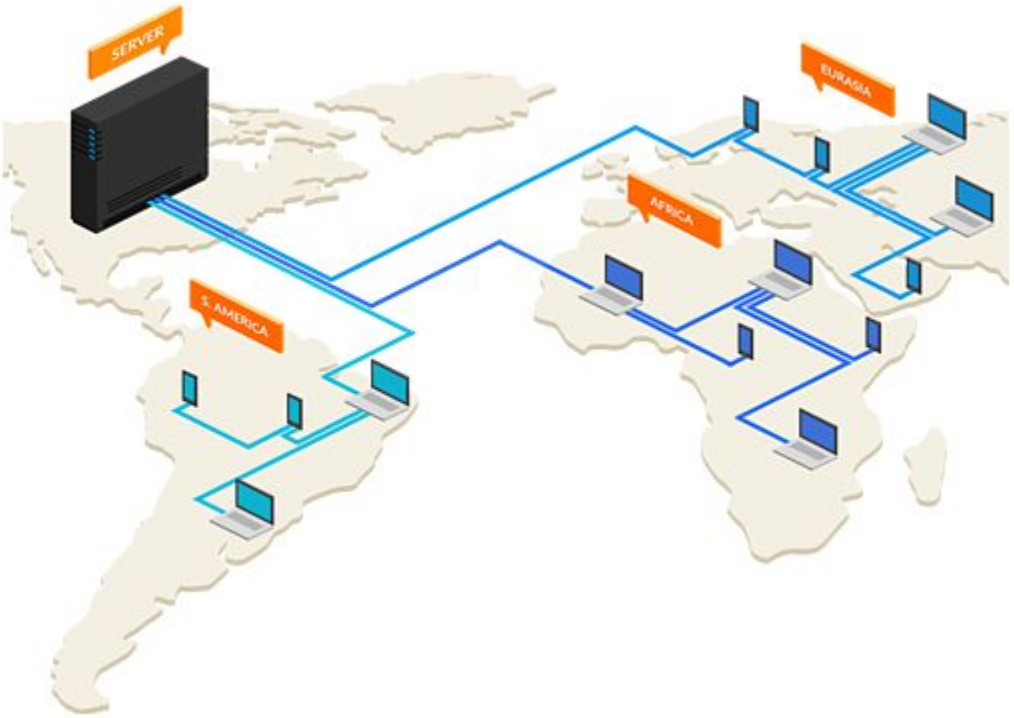
MCU: large N-way call


Your own
implementation via
signaling server

VIRAL.JS



https://github.com/PixelsCommander/ViralJS

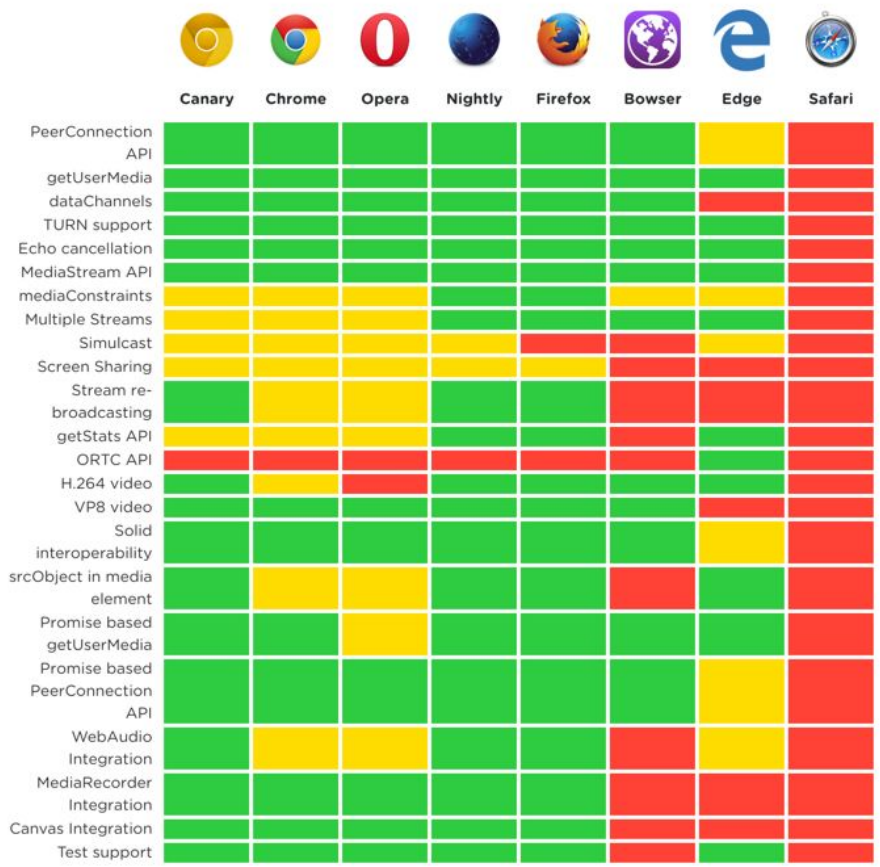


https://github.com/PixelsCommander/ViralJS

Start developing your WebRTC application today!

Give me the frameworks, libraries, tools…!

Browser support scorecard



Completion Score: 66.0%

http://iswebrtcreadyyet.com/

Library

adapter.js

Lets you use the same code in all browsers:

https://github.com/webrtc/adapter

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

JavaScript frameworks

Video chat:

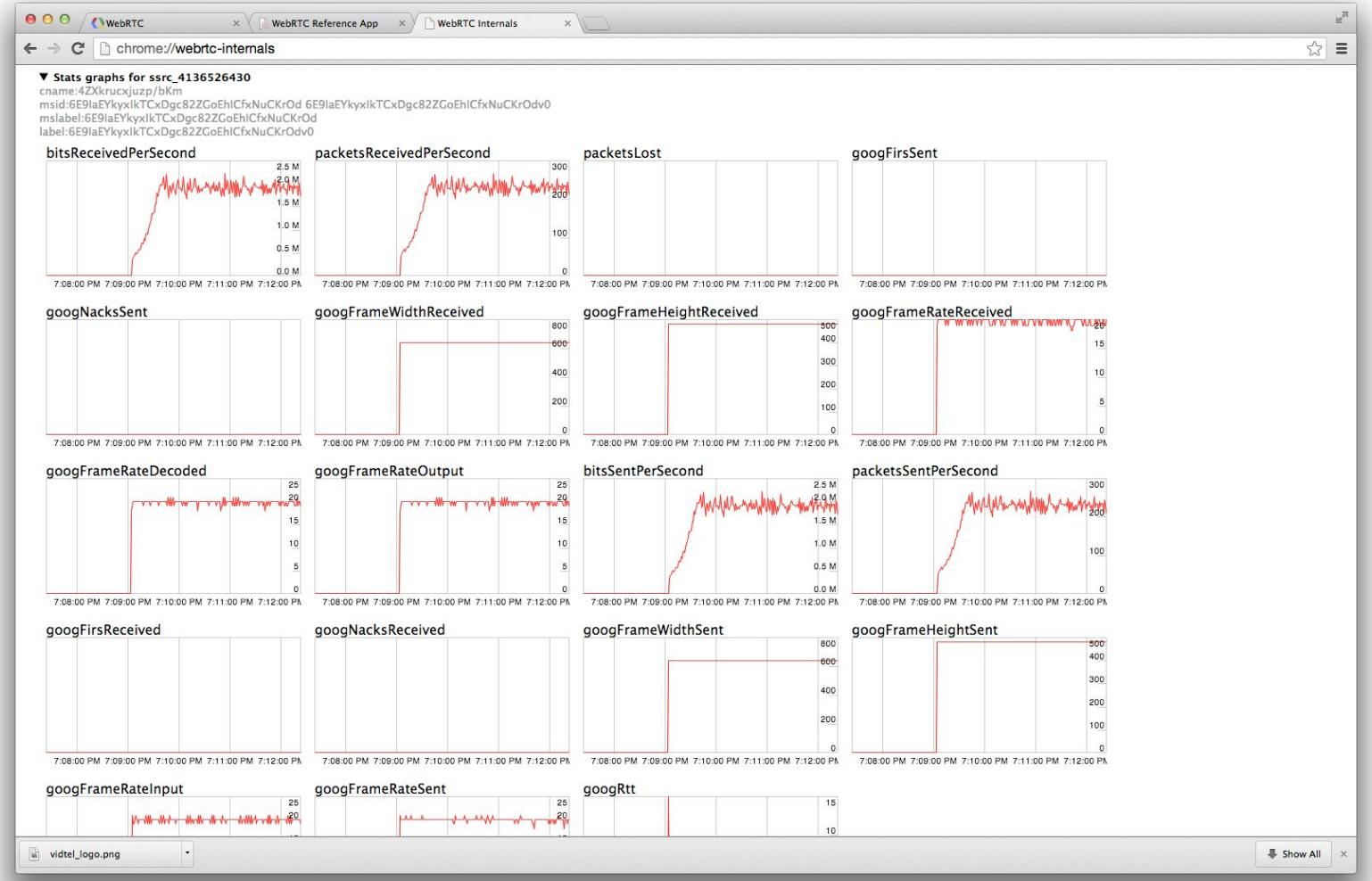
- SimpleWebRTC - https://github.com/andyet/SimpleWebRTC
- easyRTC - https://github.com/priologic/easyrtc
- webRTC.io - https://github.com/webRTC-io/webRTC.io

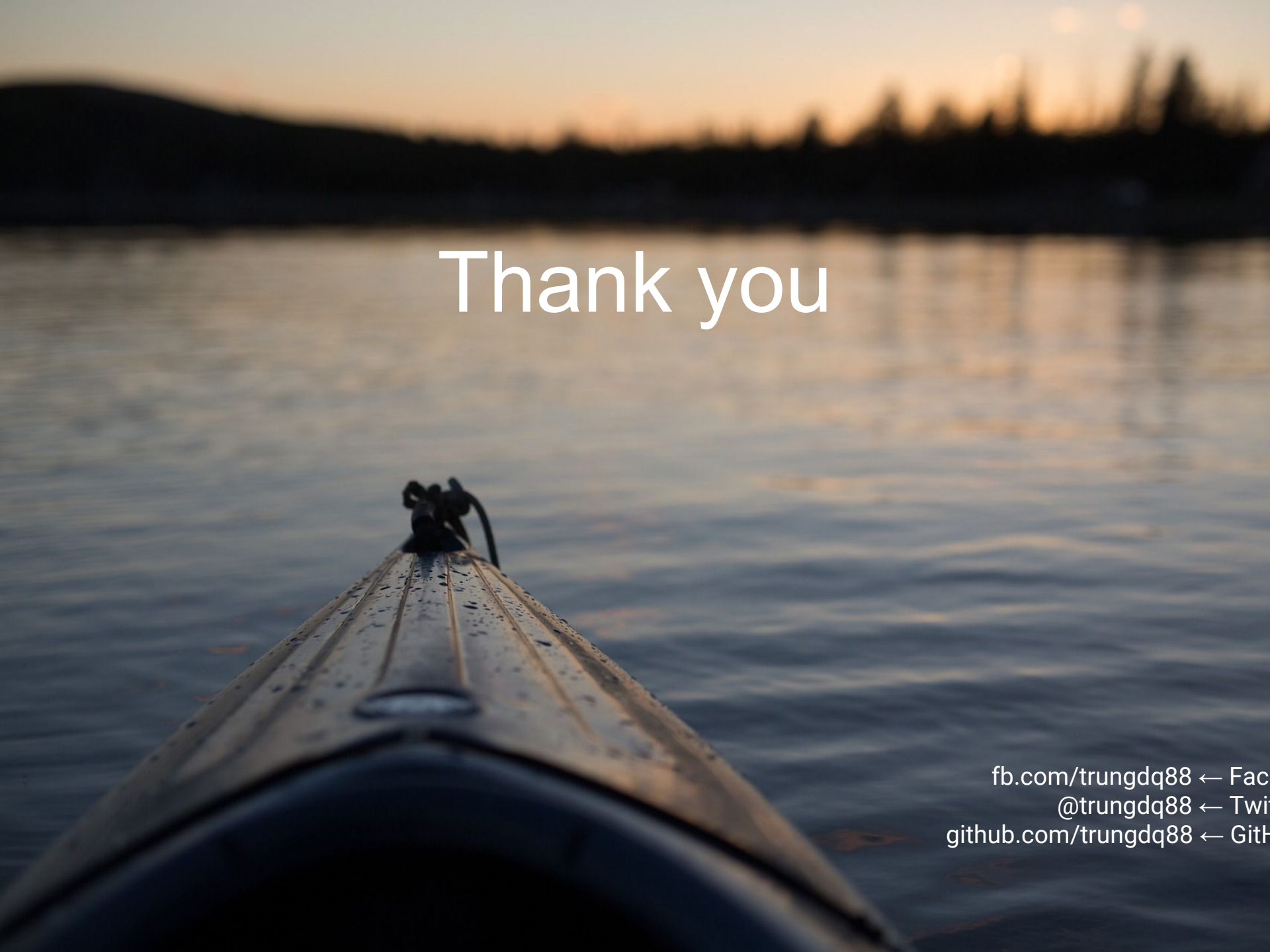
Peer-to-peer data:

- PeerJS - http://peerjs.com/
- Sharefest - https://github.com/peer5/sharefest

Tools

chrome://webrtc-internals




Thank you
fb.com/trungdq88 ← Fac
@trungdq88 ← Twi
github.com/trungdq88 ← GitH

Answer some questions:

quangtrung 10:49 AM

@channel:

Mọi người có ai từng thắc mắc đâu là công nghệ đằng sau web app http://appear.in mà mọi người dùng để họp team khi remote? Đâu là kỹ thuật truyền gửi dữ liệu nhanh nhất trên công nghệ Web hiện tại? Tại sao khi video call thì thỉnh thoảng lại bị lag? — Em chỉ hỏi cho vui vậy thôi, có hay không thì mọi người cũng tham gia buổi sharing về Web Real Time Communication (WebRTC) vào 4h chiều mai tại War Room nhé.

WebRTC là công nghệ không mới lắm (từ 2013), tuy nhiên cho đến nay vẫn ít có developer nào dám đụng đến vì độ phức tạp trong cách viết và sự ổn định của bộ spec đi kèm. Trong buổi này em sẽ giới thiệu ở mức cơ bản và demo một web app nhỏ sử dụng WebRTC để mọi người nắm được cách hoạt động của nó.

Ai tham gia thì react vào bên dưới để em gửi Calendar nha 🙂 Thanks mọi người.

Thời gian: 4h-5h, thứ 6 (19 tháng tám)
Địa điểm: War Room (edited)

👍 20

References

This slide: http://bit.ly/webrtc-slide

https://webrtc.org/start/
http://www.html5rocks.com/en/tutorials/webrtc/basics/
http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/
http://io13webrtc.appspot.com/
https://hacks.mozilla.org/2013/03/webrtc-data-channels-for-great-multiplayer/
https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity
https://webrtchacks.com/an-intro-to-webrtcs-natfirewall-problem/
https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/WebRTC_basics
https://xirsys.com/webrtc-connectivity-woes-and-you/
https://webrtc.github.io/samples/src/content/peerconnection/trickle-ice/
https://hpbn.co/webrtc/ ← it's a book, highly recommend, lot of images copied from there
https://webrtc-conference.com/interview-svein-willassen-appear-in/

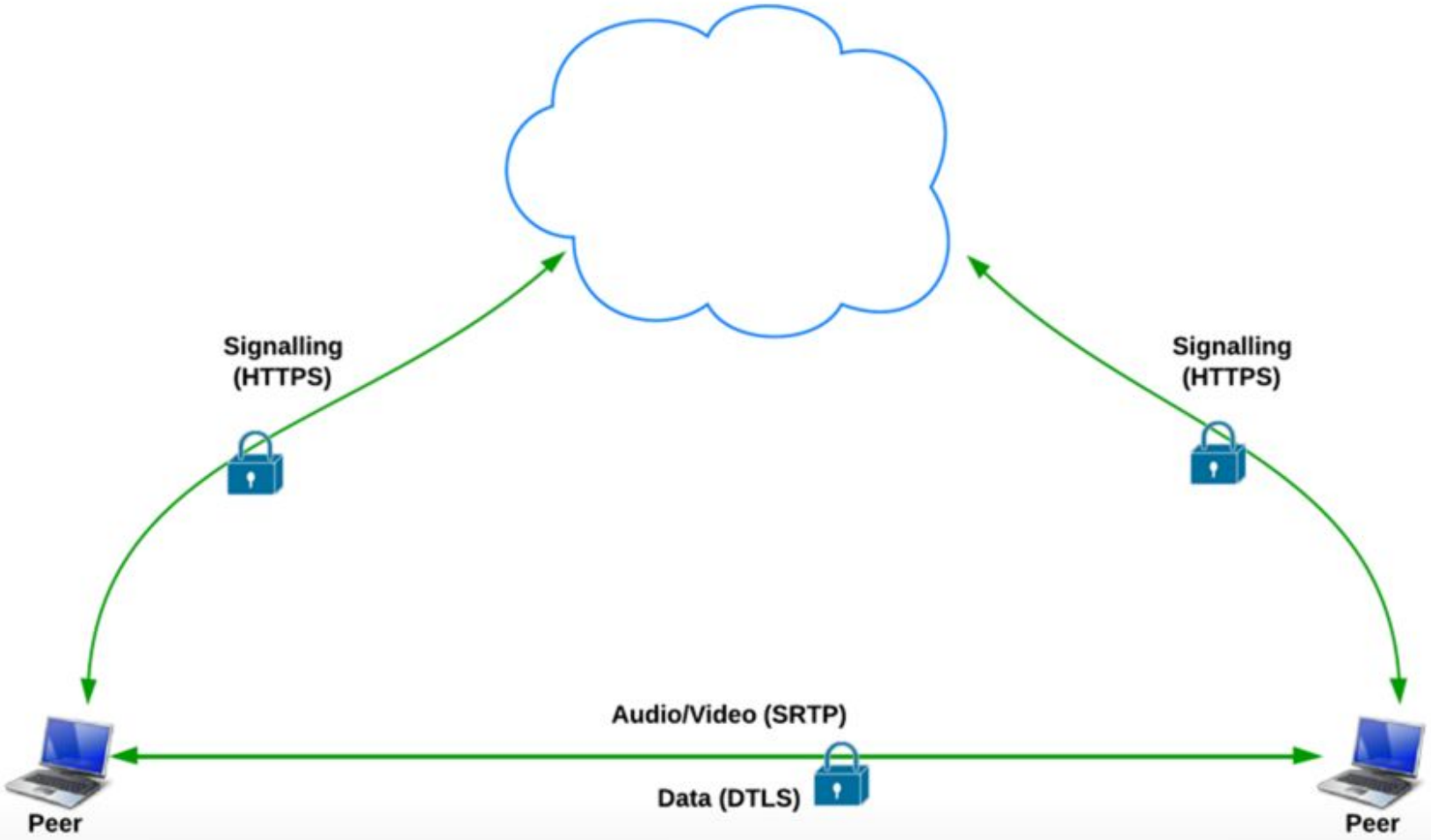
Lot of images from https://pixabay.com
Thanks Selena Gomez for a nice picture, contact me to if having any copyright issue.

My info:

fb.com/trungdq88 ← Facebook.
@trungdq88 ← Twitter .
github.com/trungdq88 ← GitHub .

Conditional slides

Secure pathways



protocols and services above it.

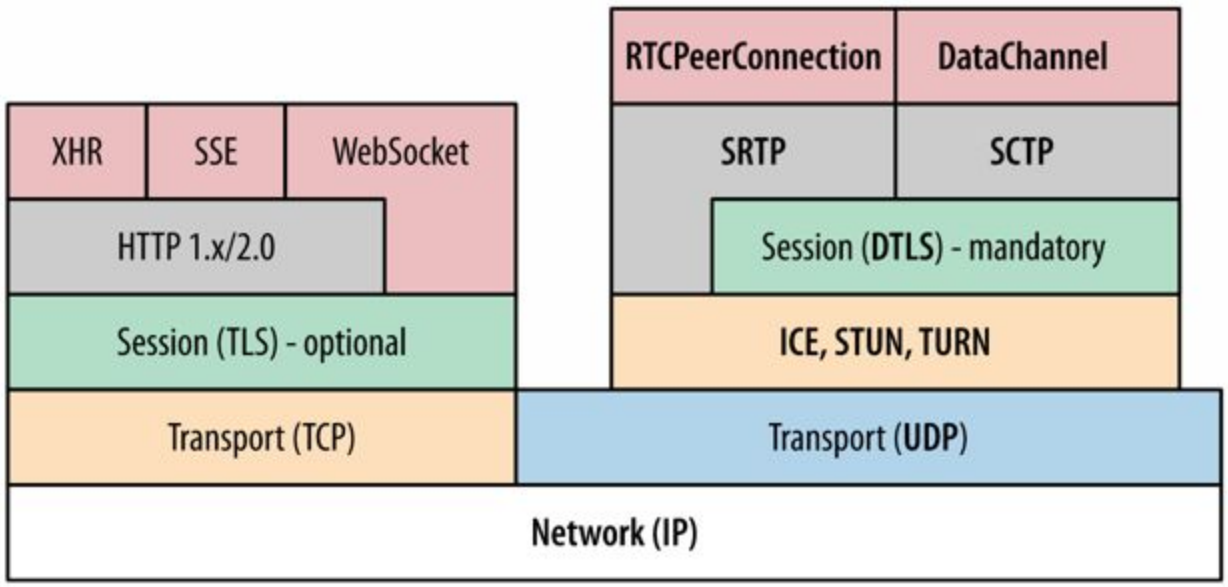


Figure 18-3. WebRTC protocol stack

- ICE: Interactive Connectivity Establishment (RFC 5245)
  - STUN: Session Traversal Utilities for NAT (RFC 5389)
  - TURN: Traversal Using Relays around NAT (RFC 5766)
- SDP: Session Description Protocol (RFC 4566)
- DTLS: Datagram Transport Layer Security (RFC 6347)
- SCTP: Stream Control Transport Protocol (RFC 4960)
- SRTP: Secure Real-Time Transport Protocol (RFC 3711)

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

http://demos.zingaya.com/webrtc-pstn/