WEBRTC: REAL-TIME COMMUNICATION BETWEEN WEB BROWSERS

Safiqul Islam

IN5150 - Recent Advancements in Internet Protocols

DISCLAIMER

- Many slides and pictures from various sources :
- webrtc.org
- L. Cicco's GCC paper (GCC image)
- RFC8698 (NADA images)

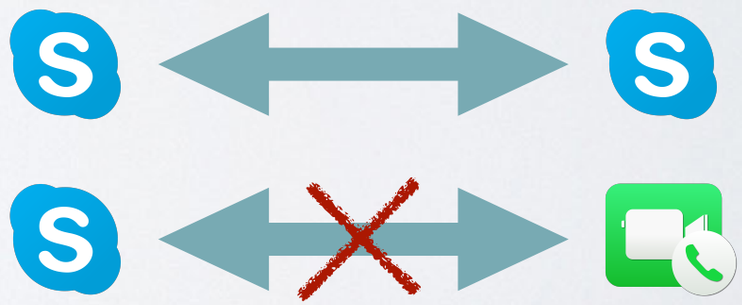
Overview

Real Time Communication

- Lots of RTC applications are available on the web

  ➡ Smart devices + high speed Internet Connections

  ➡ proprietary set of algorithms



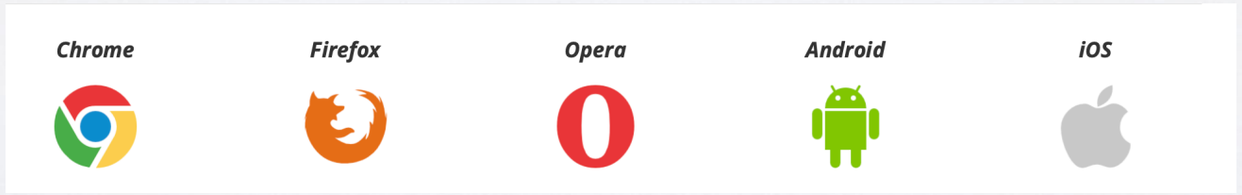
Interoperability?

Why not use web browsers for RTC?

Web Real Time Communication

- Enables peer-to-peer video, audio and data sharing between browsers

- WebRTC (W3C) and RTCWEB (IETF) working groups are currently working on the standardization.



Google Acquired Global IP solutions in 2010 who developed RTC based solutions

WebRTC is a free, open project that enables web browsers with Real-Time Communications (RTC) capabilities via simple JavaScript APIs

webrtc.org

Open Source Release by Google in 2011

W3C home > Mailing lists > Public > public-webrtc@w3.org > May 2011

Google release of WebRTC source code

This message: [Message body] [Respond] [More options]
Related messages: [Previous message]

From: Harald Alvestrand <hta@google.com>
Date: Wed, 1 Jun 2011 01:04:28 +0200
Message-ID: <BANLkTinXEJypAc2tVvAySzUGOtG9kLZNXA@mail.gmail.com>
To: public-webrtc@w3.org

Today, Google made available WebRTC, an open source software package for real-time voice and video on the web that we will be integrating in Chrome. This is our initial contribution to achieve the mission of making audio and video available in all browsers, through a uniform standard set of APIs. This initial release will provide the functionality we envision for WebRTC/RTCWEB, as detailed at https://sites.google.com/site/webrtc/. Working with the browser community and working groups like this, our goal is to expand the available APIs over the next few months for developers to create voice and video applications on the web.

The underlying components we're releasing are stable and the interfaces for this initial release are consistent with the discussions in this working group. We will continue to provide working implementations for consideration and feedback to collectively ensure stable standards are finalized. Google is committed to fully supporting these standards and we look forward to your input in the coming months.

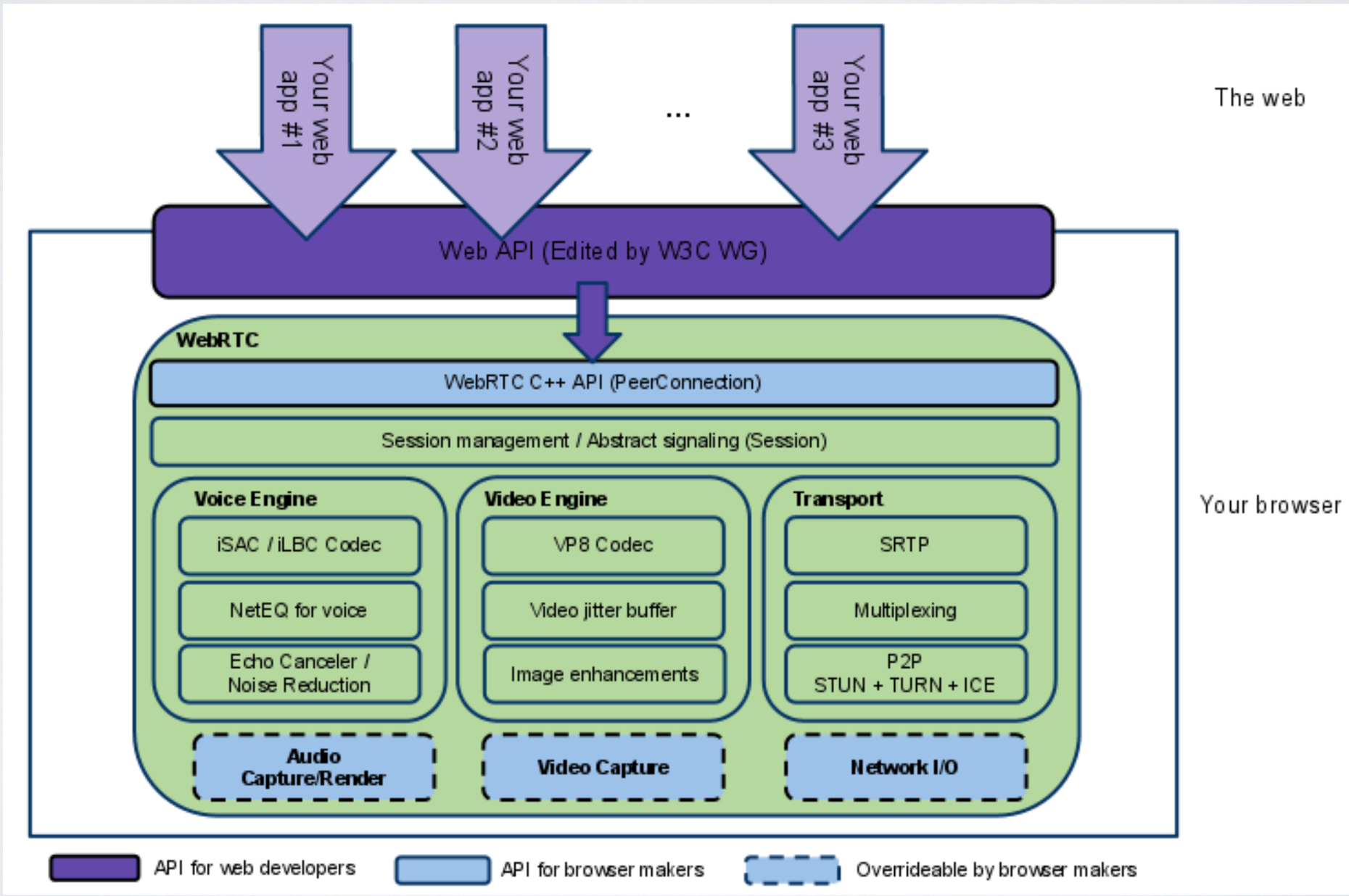
Harald, speaking for Google.

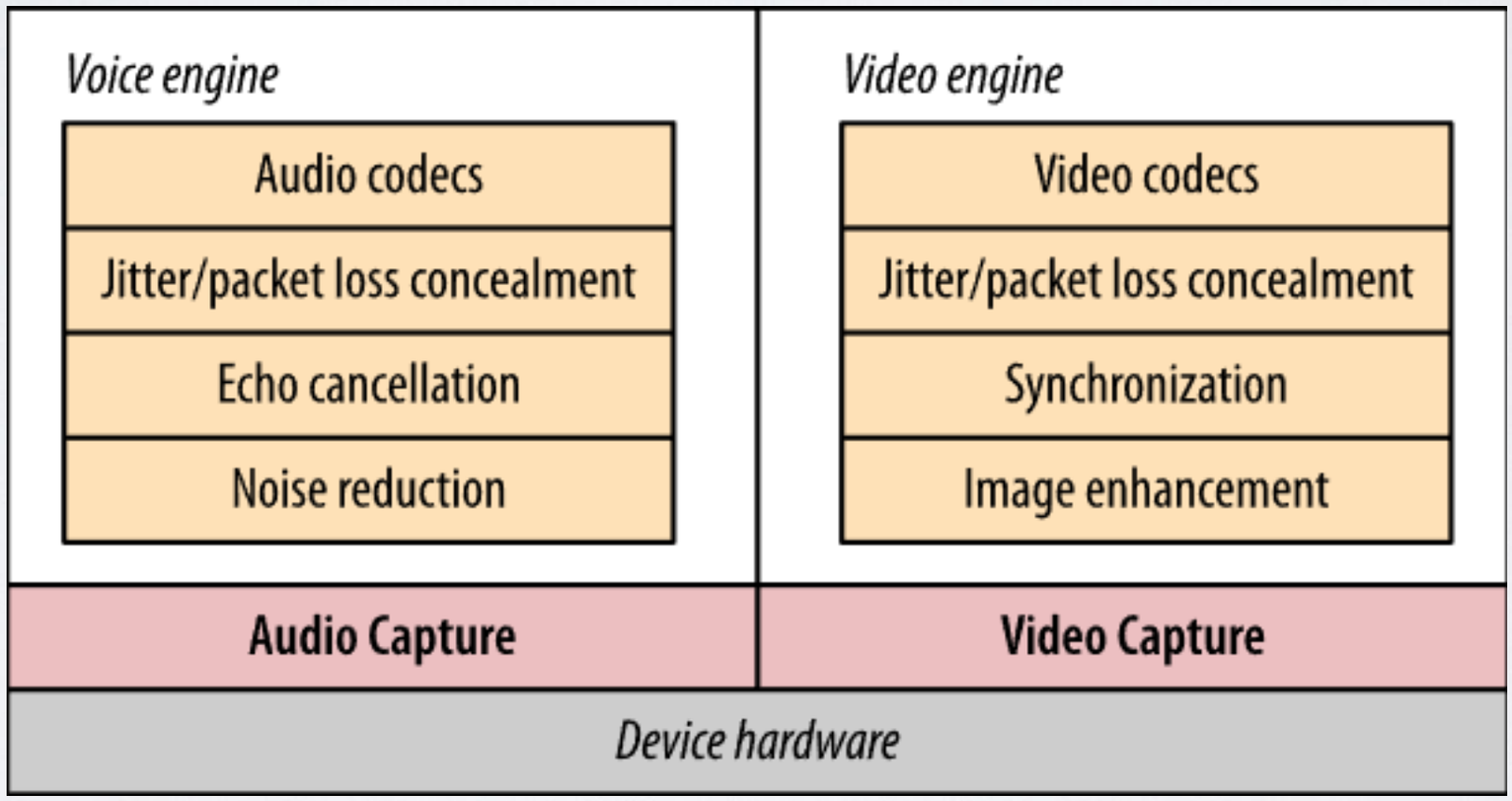
STANDARDS

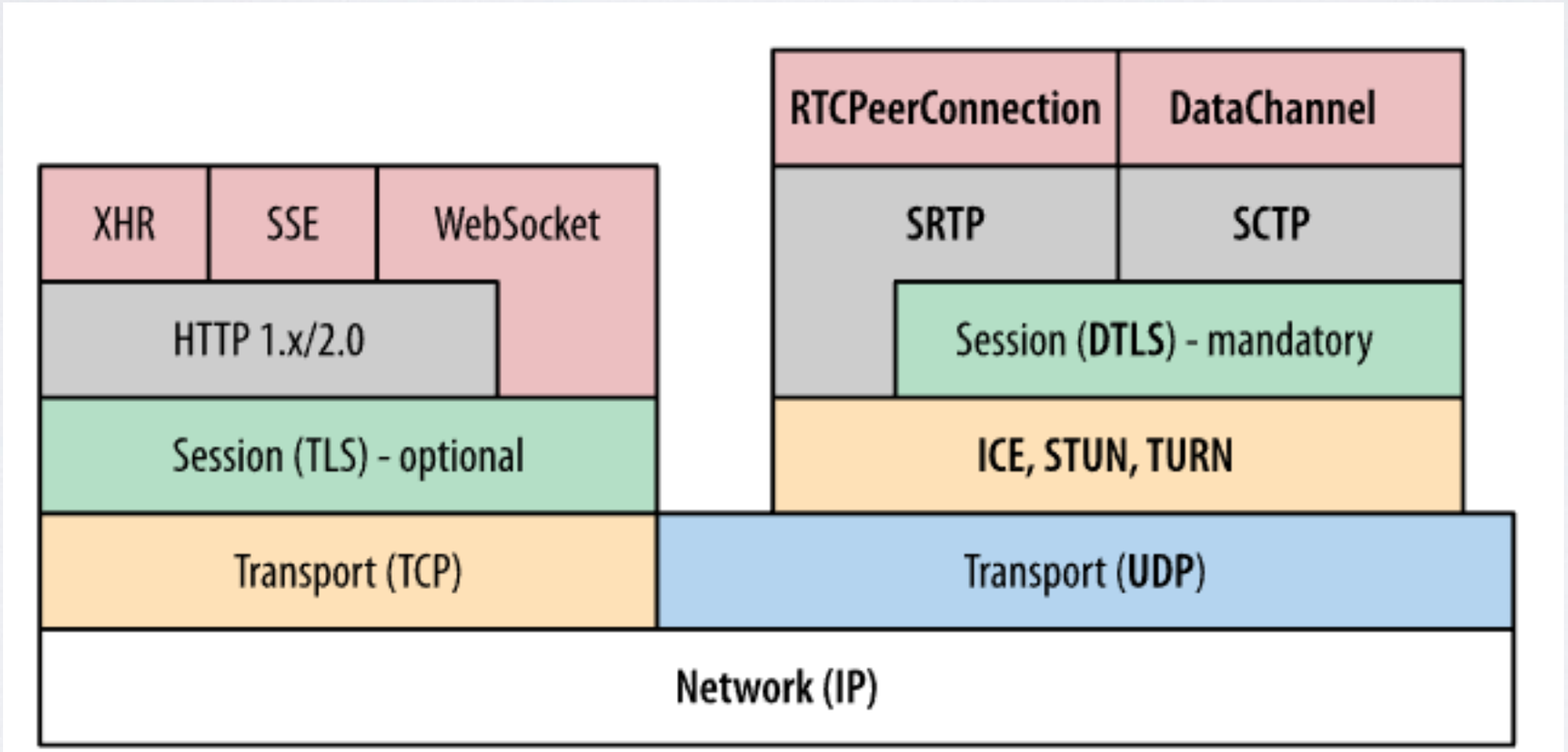
- W3C defines the APIs that a Web application can use to control this communication (Javascript APIS for HTML)

- IETF defines the formats and protocols used to communicate between browsers (signaling, NAT traversal, security, congestion control)

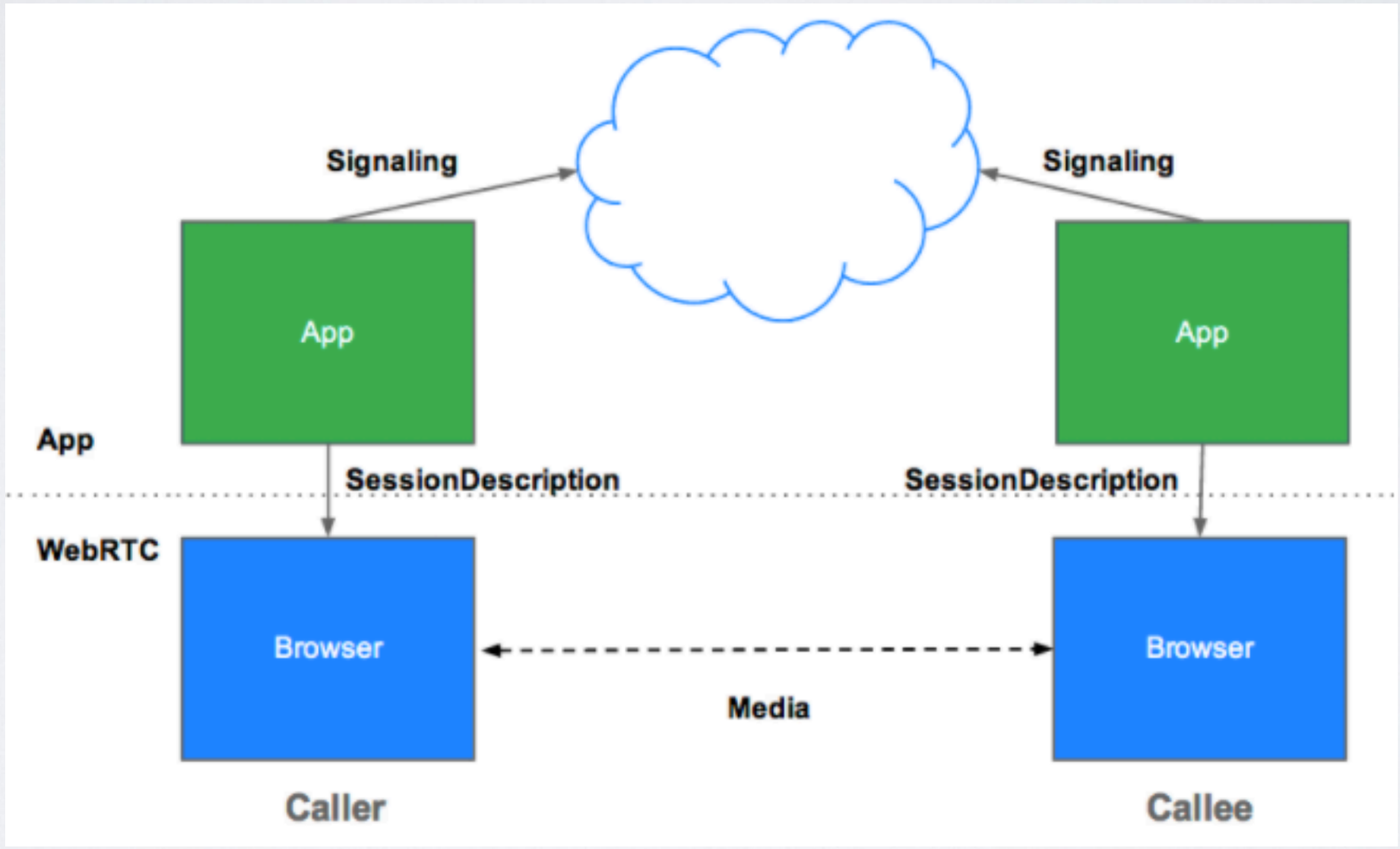
WebRTC Architecture


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


Voice engine
Audio codecs
Jitter/packet loss concealment
Echo cancellation
Noise reduction
Video engine
Video codecs
Jitter/packet loss concealment
Synchronization
Image enhancement
Audio Capture
Video Capture
Device hardware


RTCPeerConnection
DataChannel
XHR
SSE
WebSocket
SRTP
SCTP
HTTP 1.x/2.0
Session (DTLS) - mandatory
Session (TLS) - optional
ICE, STUN, TURN
Transport (TCP)
Transport (UDP)
Network (IP)

WebRTC enables peer to peer communication. But, WebRTC still needs servers!

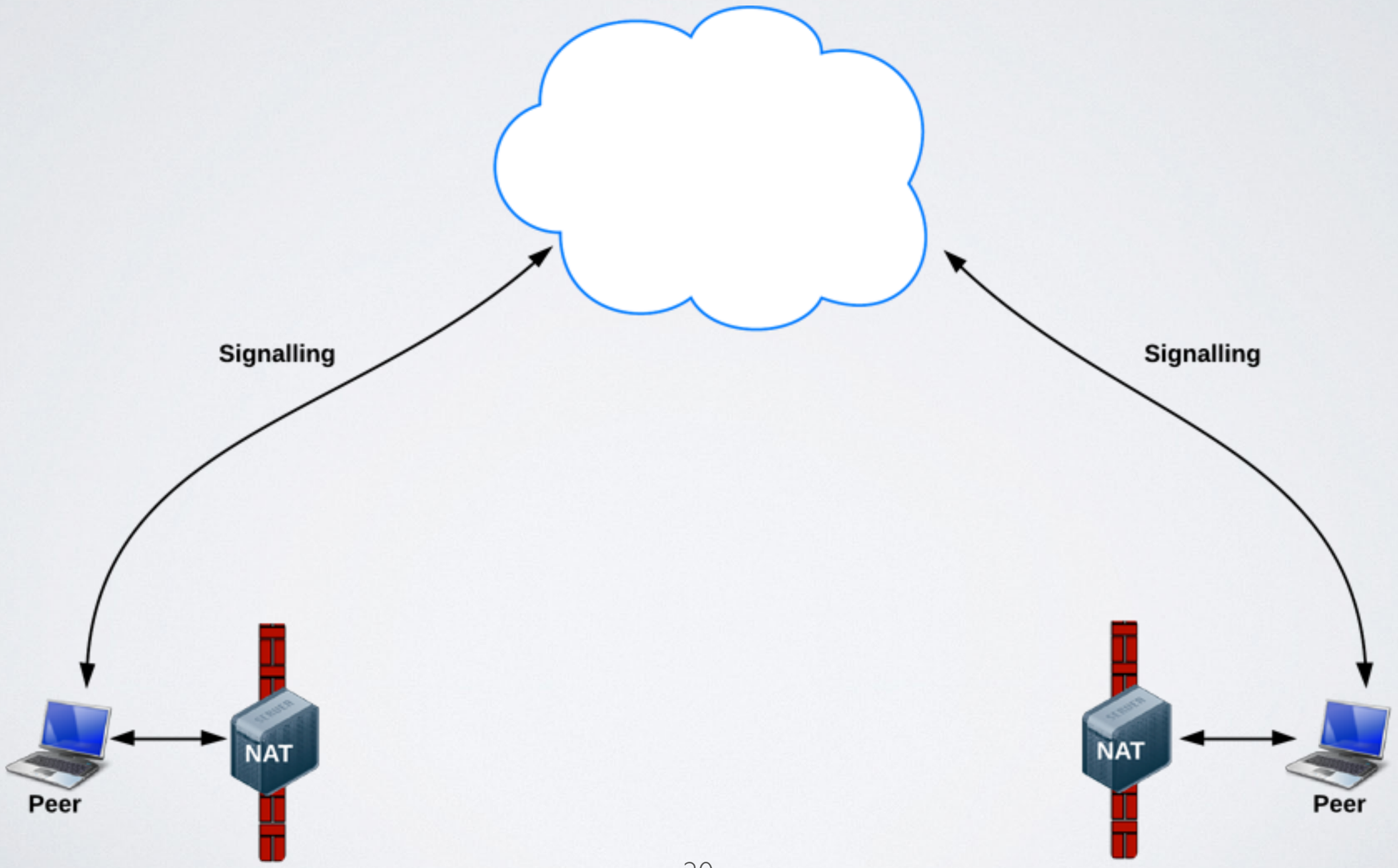

Signaling
Signaling
App
App
App
SessionDescription
SessionDescription
WebRTC
Browser
Browser
Media
Caller
Callee

Signalling

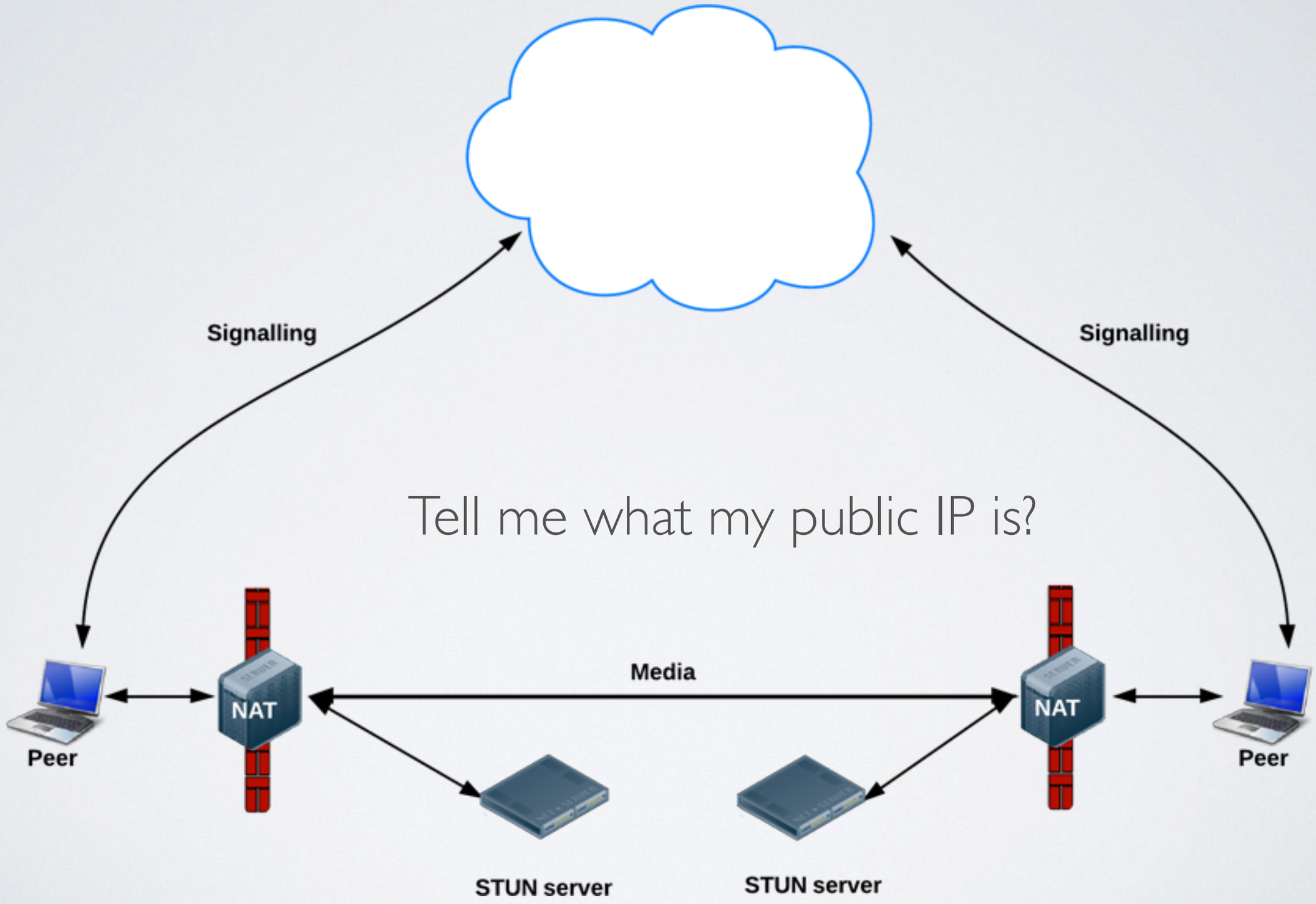
- Exchange session description object
- messaging mechanisms, protocol
- what formats I support, what should i send
- network information for p2p connection setup

Ideal Scenario

Reality!

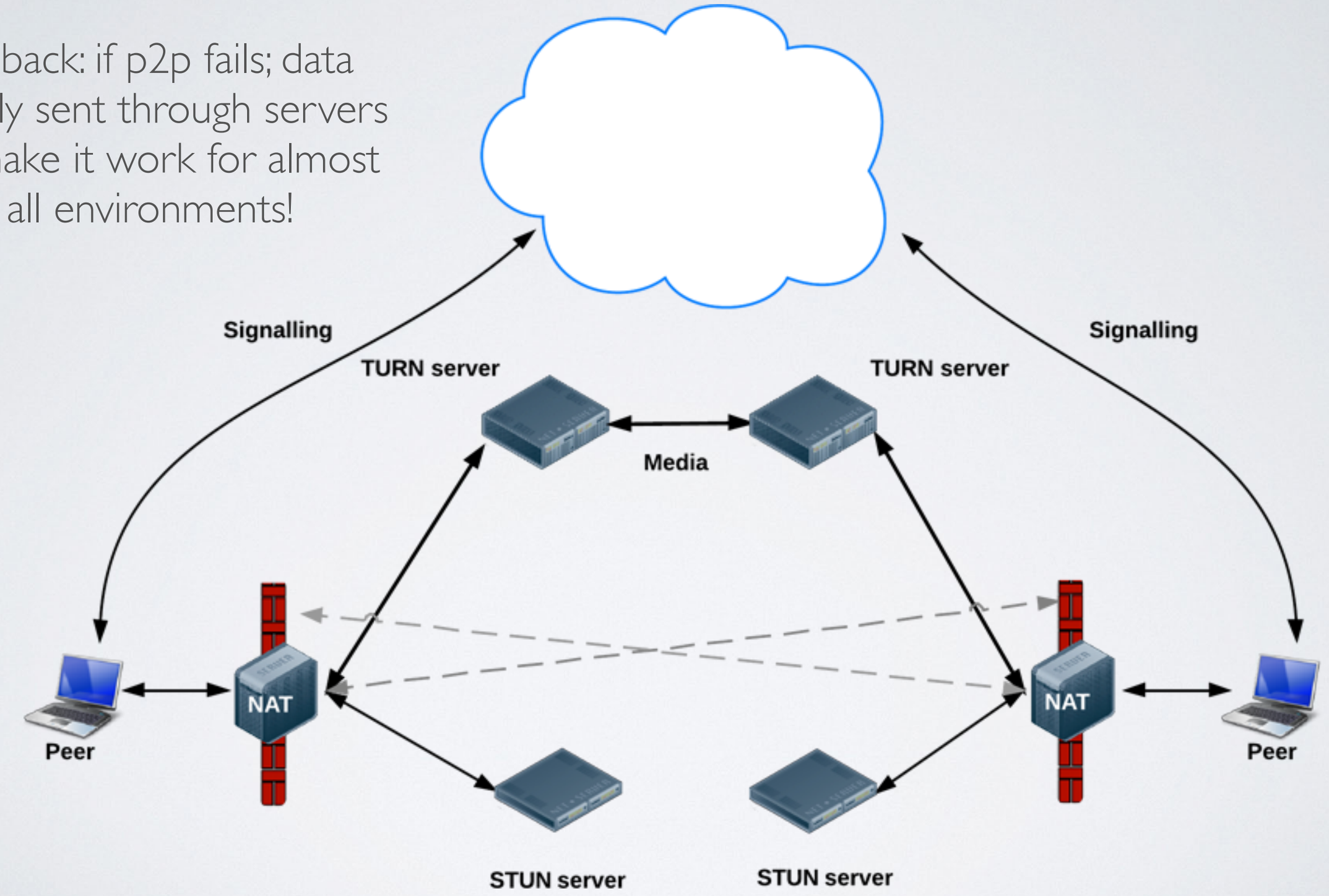


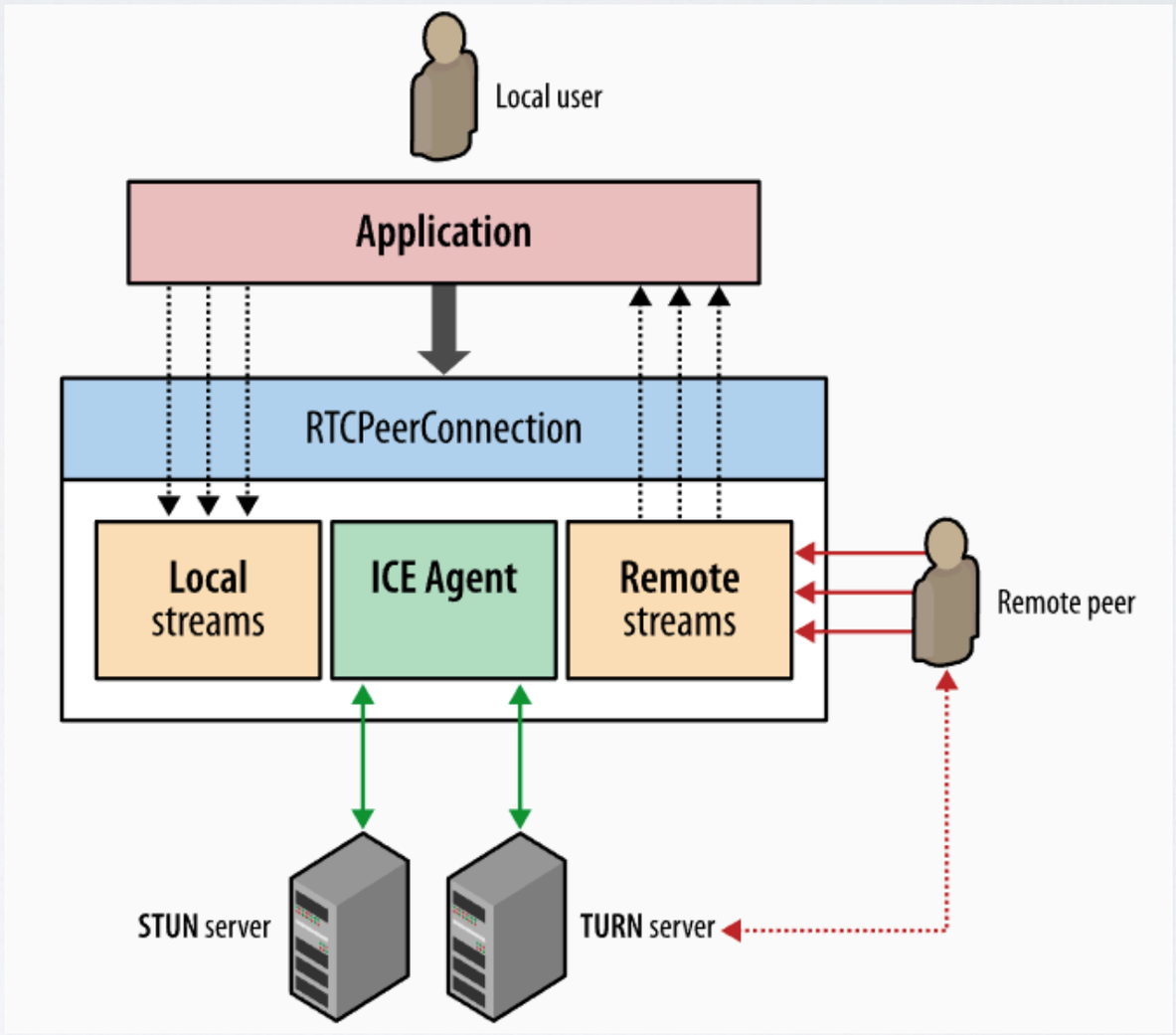
STUN


Signalling
Signalling
Tell me what my public IP is?
Media
NAT
NAT
Peer
Peer
STUN server
STUN server

TURN

Fallback: if p2p fails; data usually sent through servers to make it work for almost all environments!



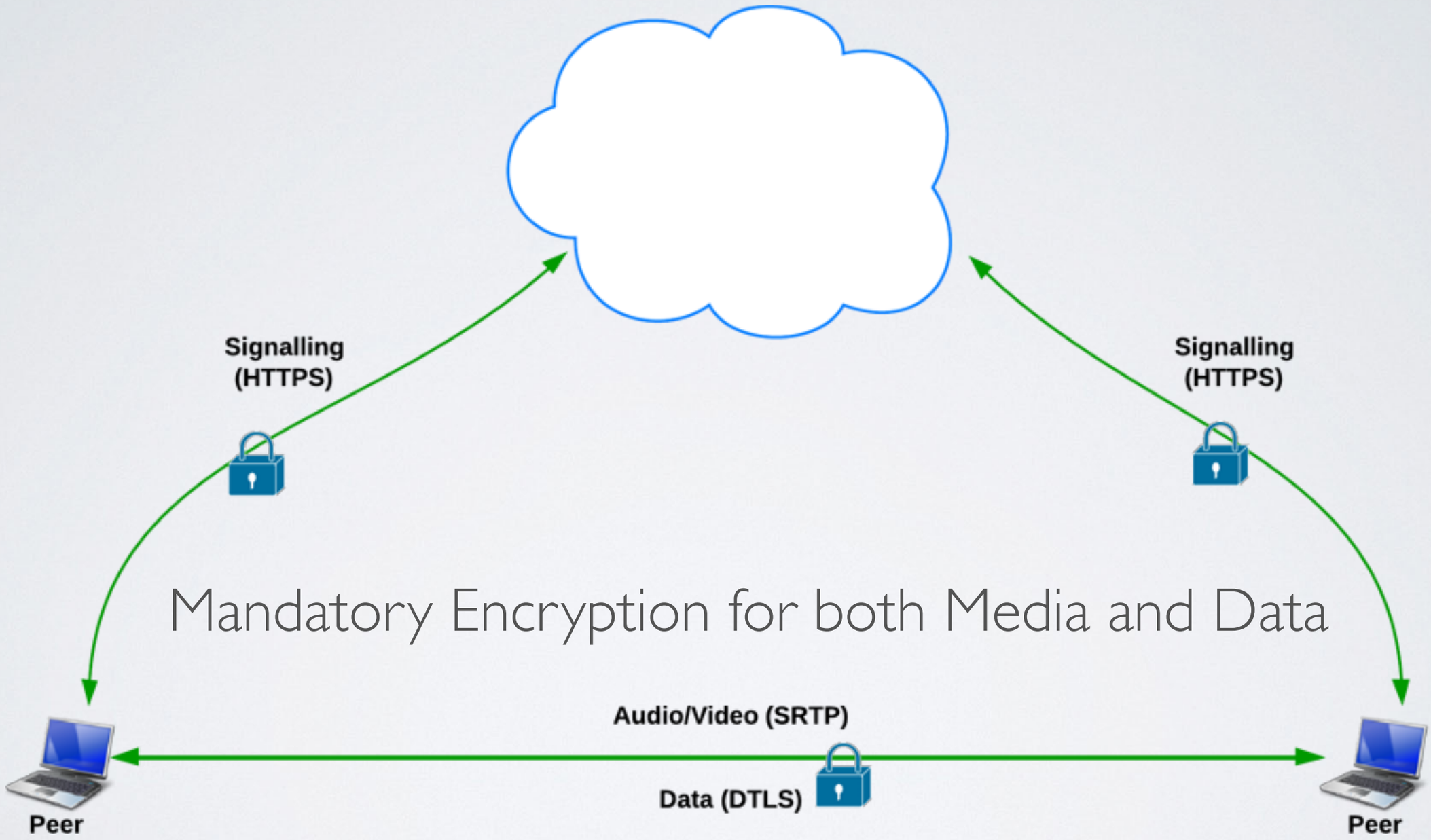

Local user
Application
RTCPeerConnection
Local streams
ICE Agent
Remote streams
Remote peer
STUN server
TURN server

ICE

- A framework for connecting peers
- Tries to find an optimal path for the peers
- Most of the peers can use STUN

Security

Signaling



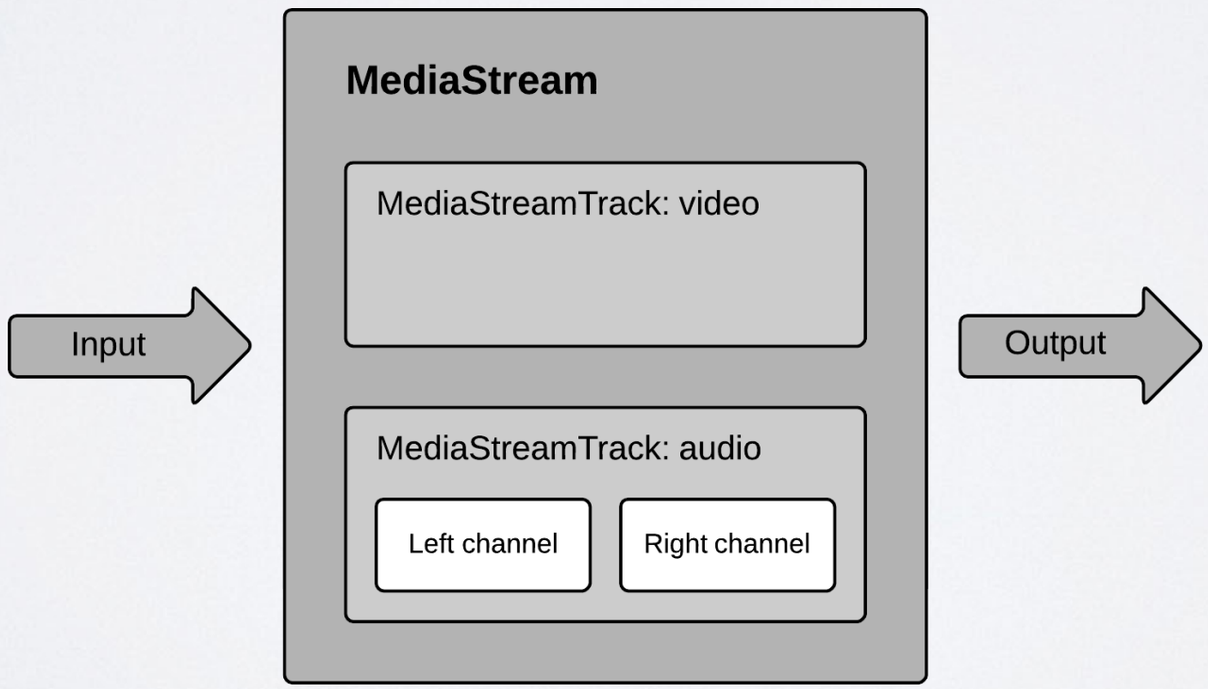
Mandatory Encryption for both Media and Data

JAVASCRIPT API

1. MediaStream (aka getUserMedia)
2. RTCPeerConnection
3. RTCDataChannel

MediaStream

- MediaStream (aka getUserMedia)
  - Represents audio and video
  - Can be obtained by navigator.getUserMedia()



```
<!DOCTYPE html>
<html>
<head>
  <title>Realtime communication with WebRTC</title>
  <link rel="stylesheet" href="css/main.css" />
</head>
<body>
  <h1>Realtime communication with WebRTC</h1>

  <video autoplay playsinline></video>
  <script src="js/main.js"></script>
</body>
</html>
```

```
'use strict';

// On this codelab, you will be streaming only video (video: true).
const mediaStreamConstraints = {
  video: true,
};

// Video element where stream will be placed.
const localVideo = document.querySelector('video');

// Local stream that will be reproduced on the video.
let localStream;

// Handles success by adding the MediaStream to the video element.
function gotLocalMediaStream(mediaStream) {
  localStream = mediaStream;
  localVideo.srcObject = mediaStream;
}

// Handles error by logging a message to the console with the error message.
function handleLocalMediaStreamError(error) {
  console.log('navigator.getUserMedia error: ', error);
}

// Initializes media stream.
navigator.mediaDevices.getUserMedia(mediaStreamConstraints)
  .then(gotLocalMediaStream).catch(handleLocalMediaStreamError);
```

RTCPEERCONNECTION

- Signal processing
- codec handling
- p2p connection
- security

RTCDATACHANNEL

- same API as websockets
- ultra-low latency
- unreliable or reliable
- secure


AppRTC
https://appr.tc/r/454516835

CHROME (WEBRTC INTERNALS)

Congestion Control Mechanisms

CONGESTION CONTROL

- IETF RMCAT WORKING GROUP is formed to develop standards for RTP based Media
    - cc requirements, designing evaluation test cases, developing congestion control mechanisms, identifying shared bottleneck and develop congestion control coupling to fairly allocate the rates for the flows sharing the same bottleneck
- Three congestion control mechanisms proposed so far : Cisco´s NADA (network-assisted dynamic adaptation) (RFC8698), Google's Google Congestion Control (GCC) (draft-ietf-rmcat-gcc-02) and Ericsson's ScreAm (RFC8298)
    - GCC is deployed in chrome, Firefox. opera …

NADA

- Rate based congestion control mechanism

- Sender reacts to the collection of network congestion indicators.

- Two modes: accelerated ramp-up and gradual rate update

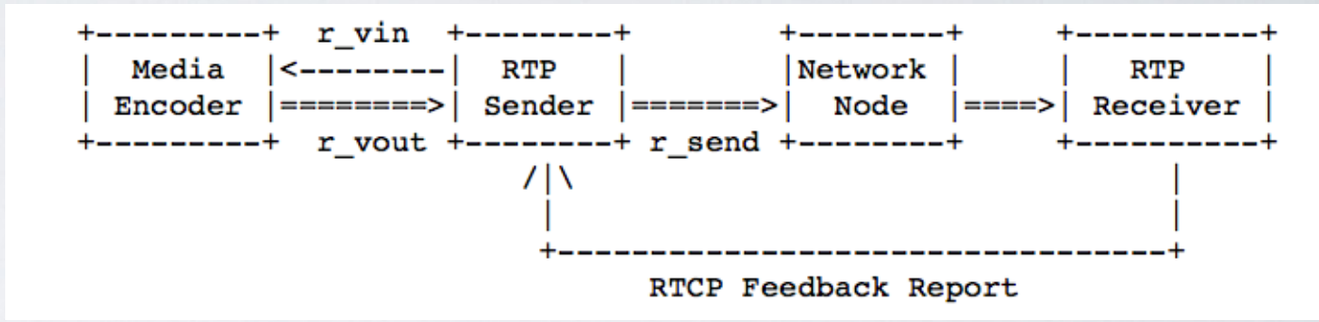


Fig: System overview

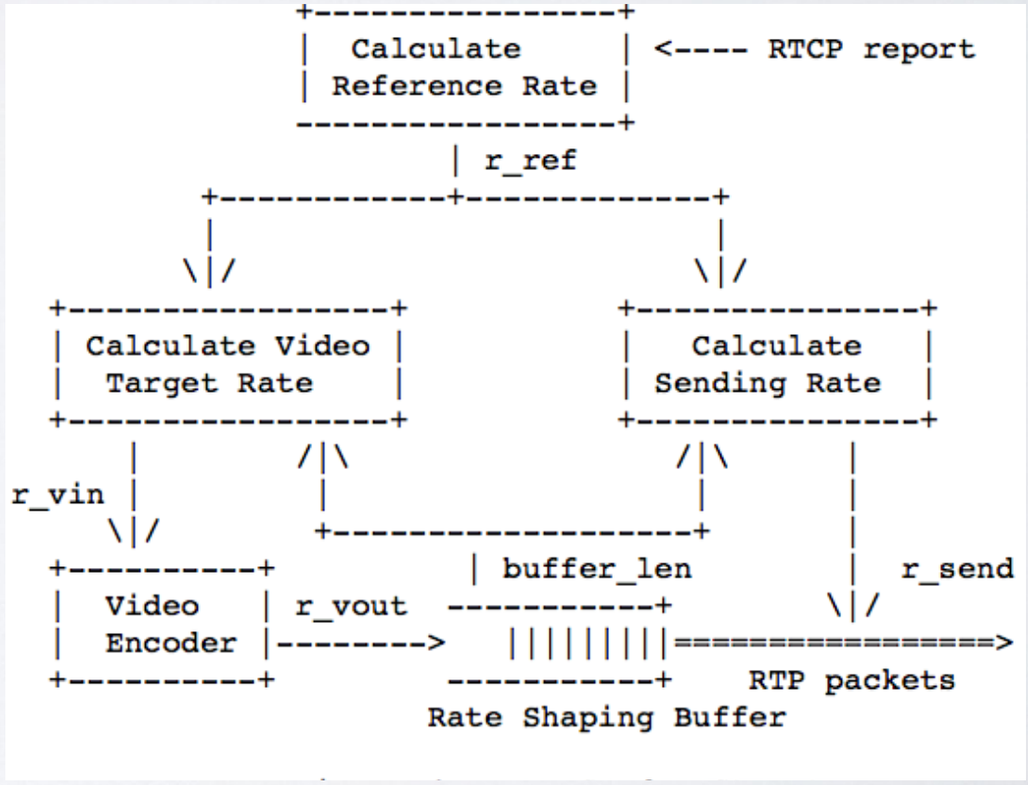
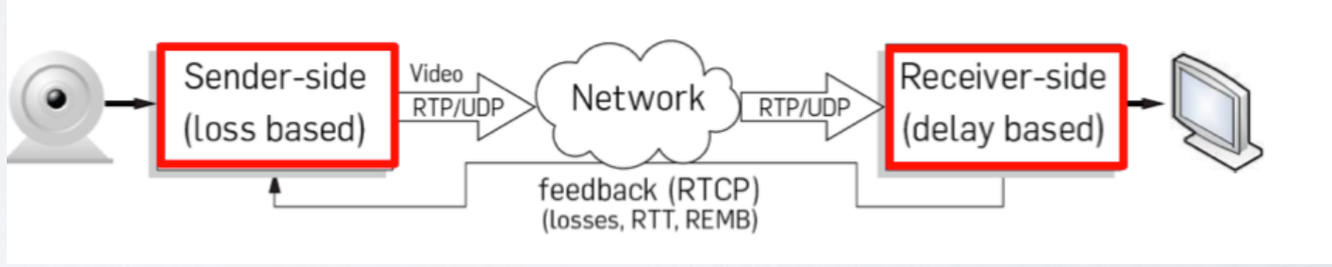


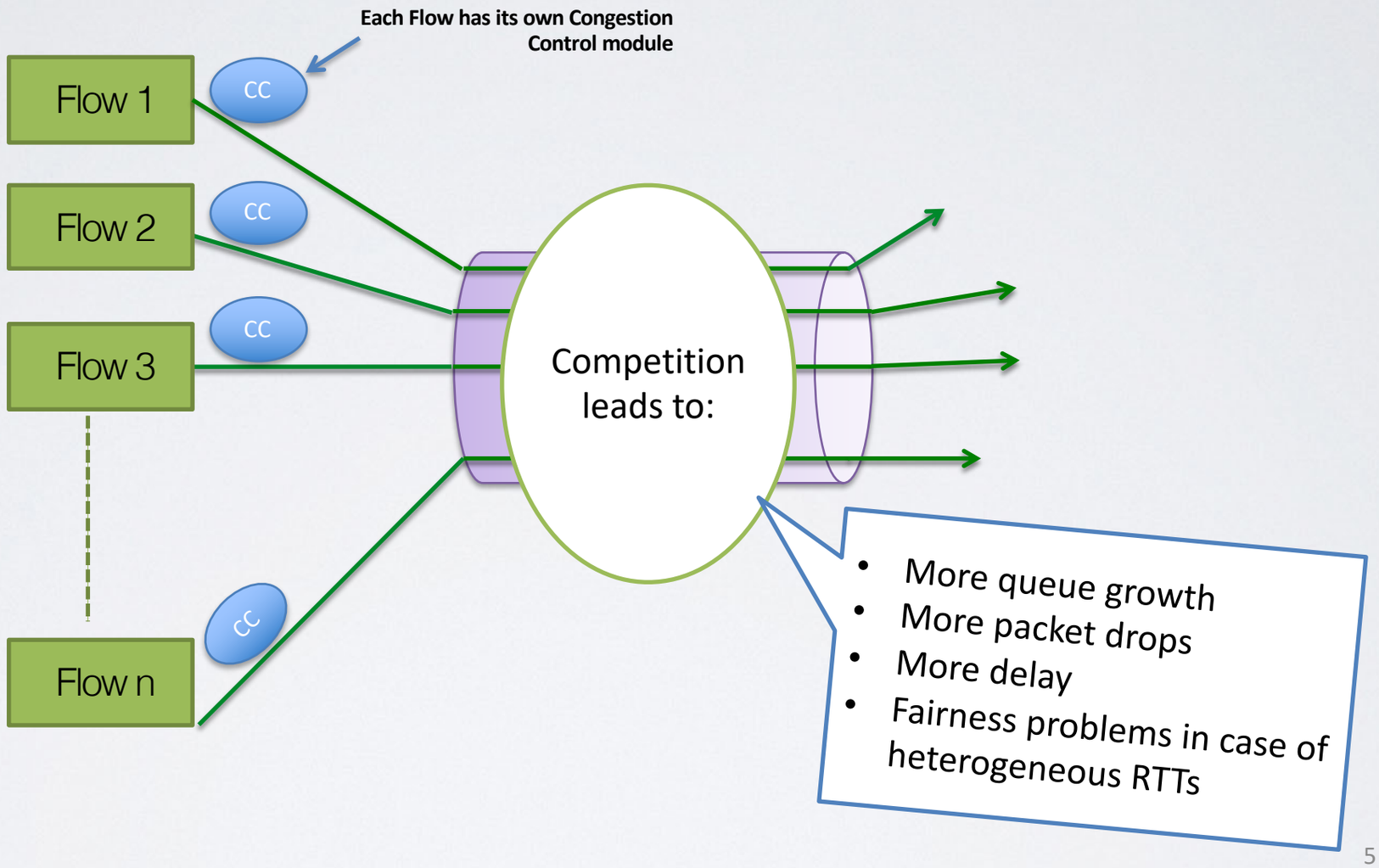
Fig: NADA Sender structure

GCC

- Another congestion control mechanism proposed for WebRTC
- Employs two controller:
  - Loss-based controller
  - Delay-based controller
- Two ways to implement this: either both controller at the sender or loss-based at the sender and delay-based controller at the receiver.

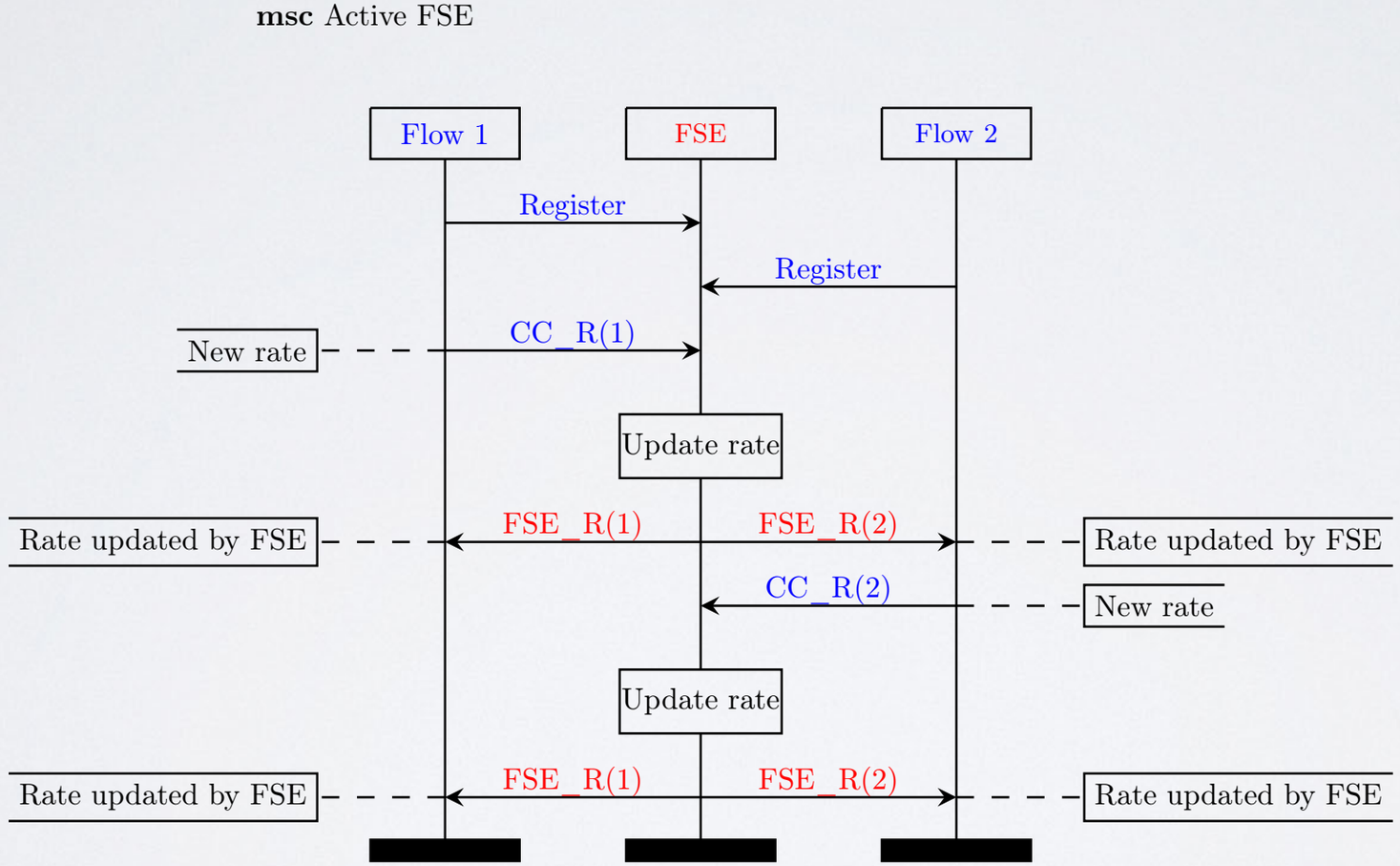


COUPLED CONGESTION CONTROL



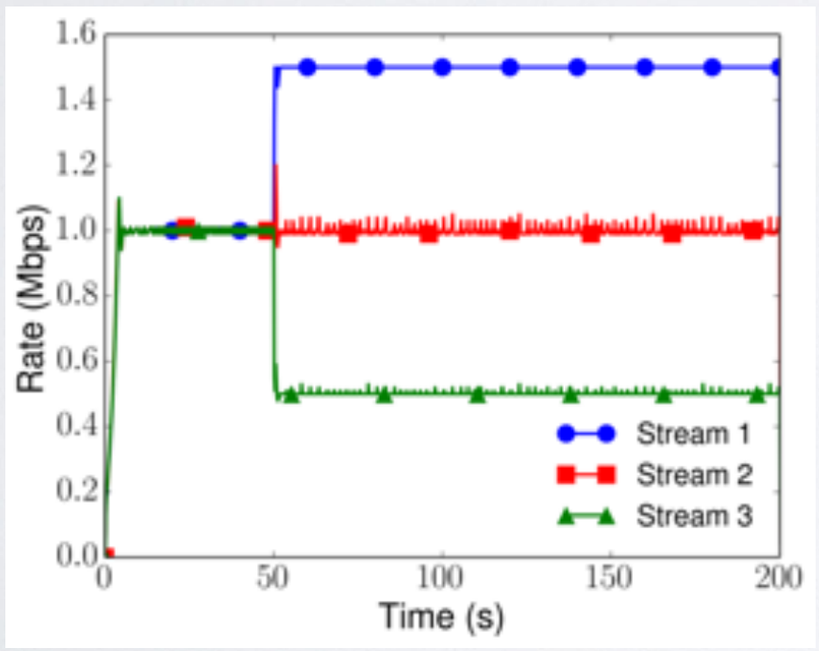
5

FLOW STATE EXCHANGE (FSE)

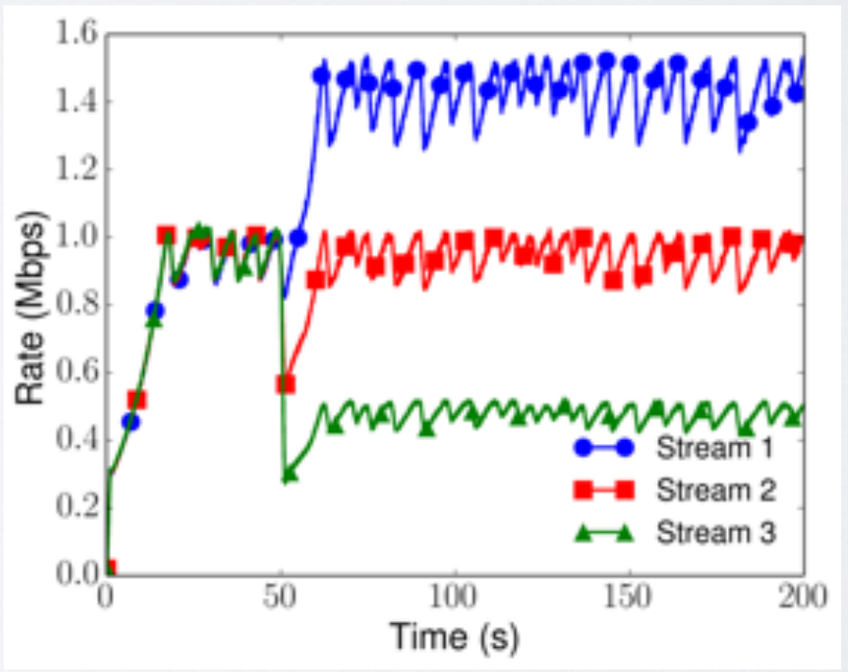


FSE (PRIORITAZATION)

- Assigned rates based on the priorities assigned from the users (RFC8699)

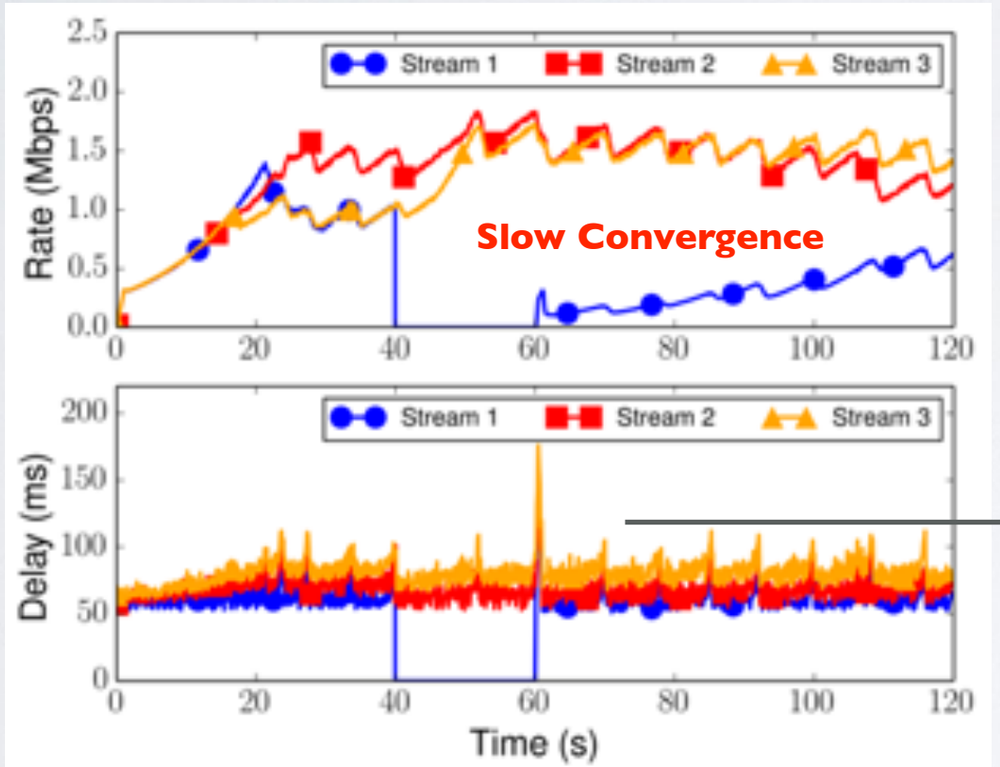


Three NADA flows with FSE

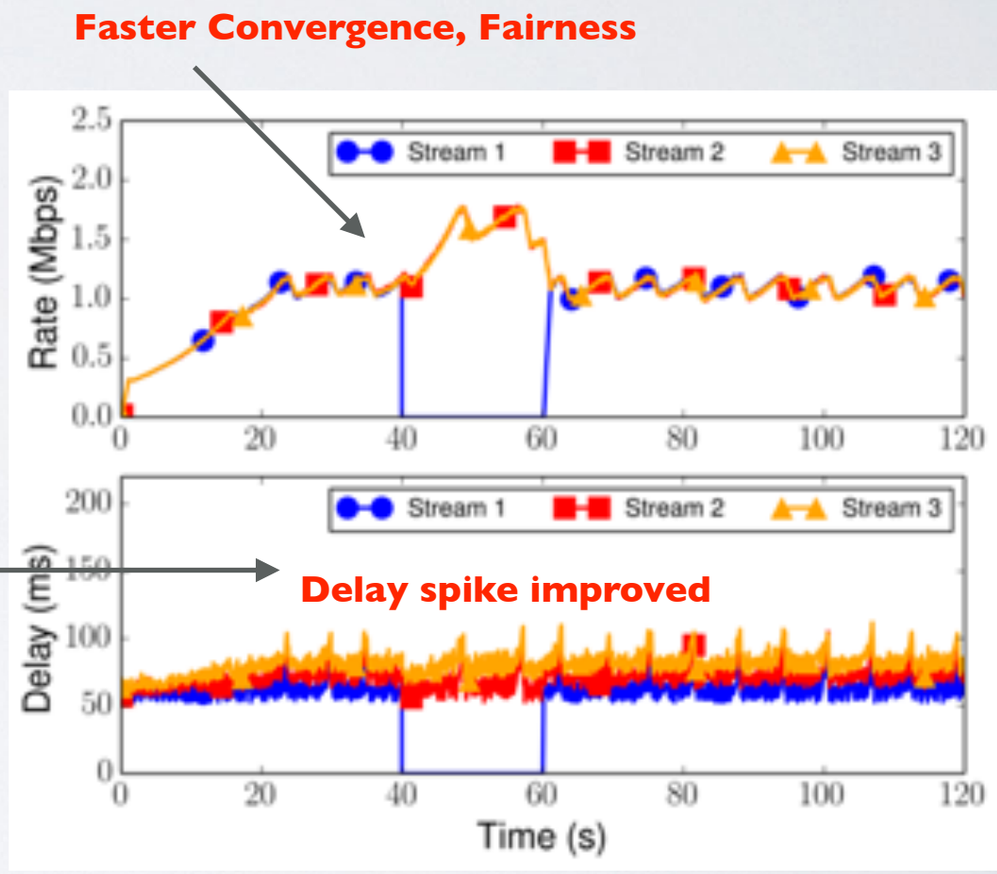


Three GCC flows with FSE

FSE (MEDIA PAUSE AND RESUME)



Three GCC flows without FSE



Three GCC flows with FSE

SHARED BOTTLENECK DETECTION (SBD)

- Multiplexing (same 5 tuple) - fits WebRTC
- Configuration (common wireless uplink)
- Measurement based SBD (RFC8382)

"Questions?"