webRTC Workshop

Callstats.io

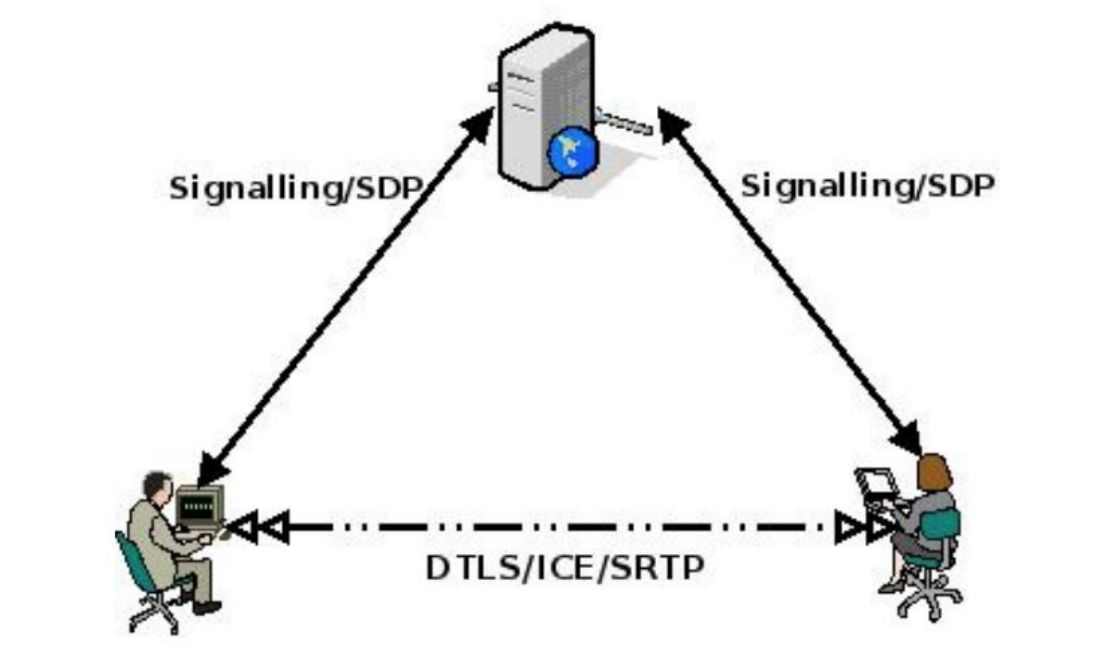
webRTC - Web Real-Time Communication

- Real-time media communication in a browser
- In the past, no standard solution!
  - No interoperability
  - Plugins needed to be installed anyway

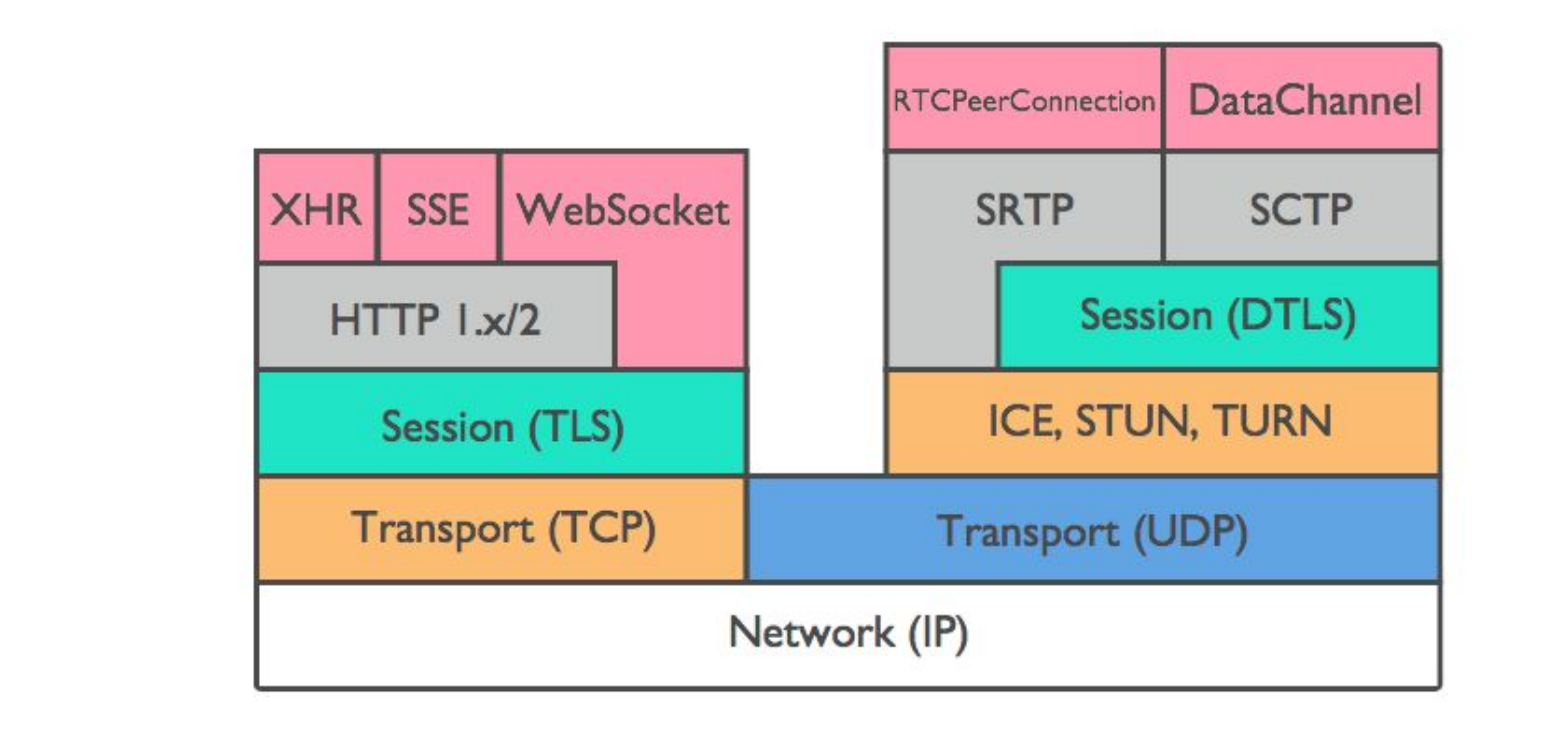
webRTC - Joint standardization efforts

- IETF - Internet Engineering Task Force (protocols and formats)
- W3c - World Wide Web consortium (UI and API access)

webRTC reference architecture.



webRTC protocol stack



webRTC protocol stack

- Signalling and Negotiation
  - Javascript Session Establishment Protocol (JSEP)
  - Session Description Protocol (SDP) adaptation
- Connection Establishment and NAT Traversal
  - Session Traversal Utilities for NAT (STUN)
  - Traversal Using Relay NAT (TURN)
  - Interactive Connectivity Establishment (ICE)
- Media Transport and Control
  - Real-time Transport (and Control) Protocol (RTP/RTCP)
  - Secure Extensions to RTP (SRTP)
  - Datagram Transport Layer Security (DTLS)
- Multimedia codecs
  - Opus audio codec (MTI, Mandatory-to-implement)
  - VP8 and H.264 video codecs (MTI, Mandatory-to-implement) •
- Generic Data
  - WebRTC Data Channels (SCTP)

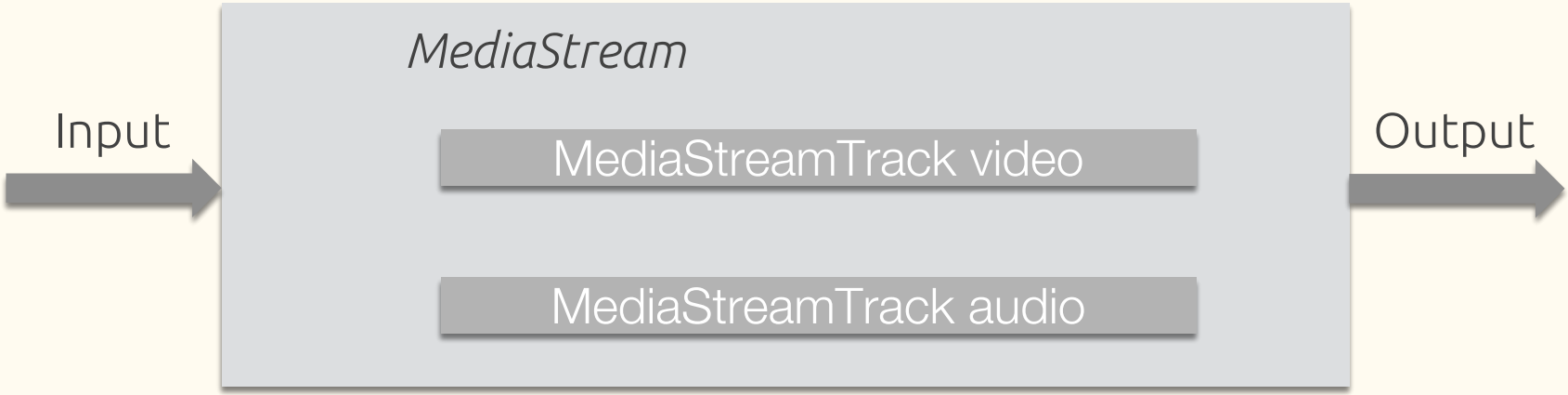



webRTC APIs

- MediaStream (to audio and video source)
- RTCPeerConnection (audio and video communication)
- RTCDataChannel (message communication)

MediaStream - getUserMedia API

- Represents stream of audio/video content
- Consists of media stream tracks



getUserMedia Usage

```
var constraints = {
   audio: true,
   video: true,
}
navigator.mediaDevices.getUserMedia(constraints)
  .then(stream => {
        // stream obtained
  })
  .catch(err => {
    console.log('getUserMedia error', err);
  })
```

https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia

RTCPeerConnection

Mechanism for communication of media streams

It provides:

- Signal processing
- Codec handling
- Peer to peer communication
- Communication security mechanisms
- Congestion control mechanisms

RTCPeerConnection Caller sample

```
var pc = new RTCPeerConnection();
pc.onaddstream = gotRemoteStream;
pc.addStream(localStream);
pc.createOffer(gotOffer);

function gotOffer(desc) {
  pc.setLocalDescription(desc);
  sendOffer(desc);
}

function gotAnswer(desc) {
  pc.setRemoteDescription(desc);
}

function gotRemoteStream(e) {
  attachMediaStream(remoteVideo, e.stream);
}
```

RTCDataChannel

- Mechanism for bidirectional exchange of data between peers
- Two modes of operation: reliable and unreliable
- Low latency
- Secure communication

RTCDataChannel Sample

```
var pc = new RTCPeerConnection();

// offer-answer exchange

sendChannel = pc.createDataChannel("sendDataChannel”);


sendChannel.onmessage = function(event){
    console.log("received: " + event.data);
};
```

webRTC Resources

webRTC 1.0 specification - https://www.w3.org/TR/webrtc/

webRTC Samples - https://github.com/webrtc/samples

webRTC Stats - https://www.w3.org/TR/webrtc-stats/

webRTC News and blogs - https://bloggeek.me/posts/

webRTC Landscape - https://bloggeek.me/introducing-webrtc-developer-tools-landscape/

https://github.com/callstats-io/webrtc-workshop