Publishing and playing stream via WebRTC over TCP

Overview

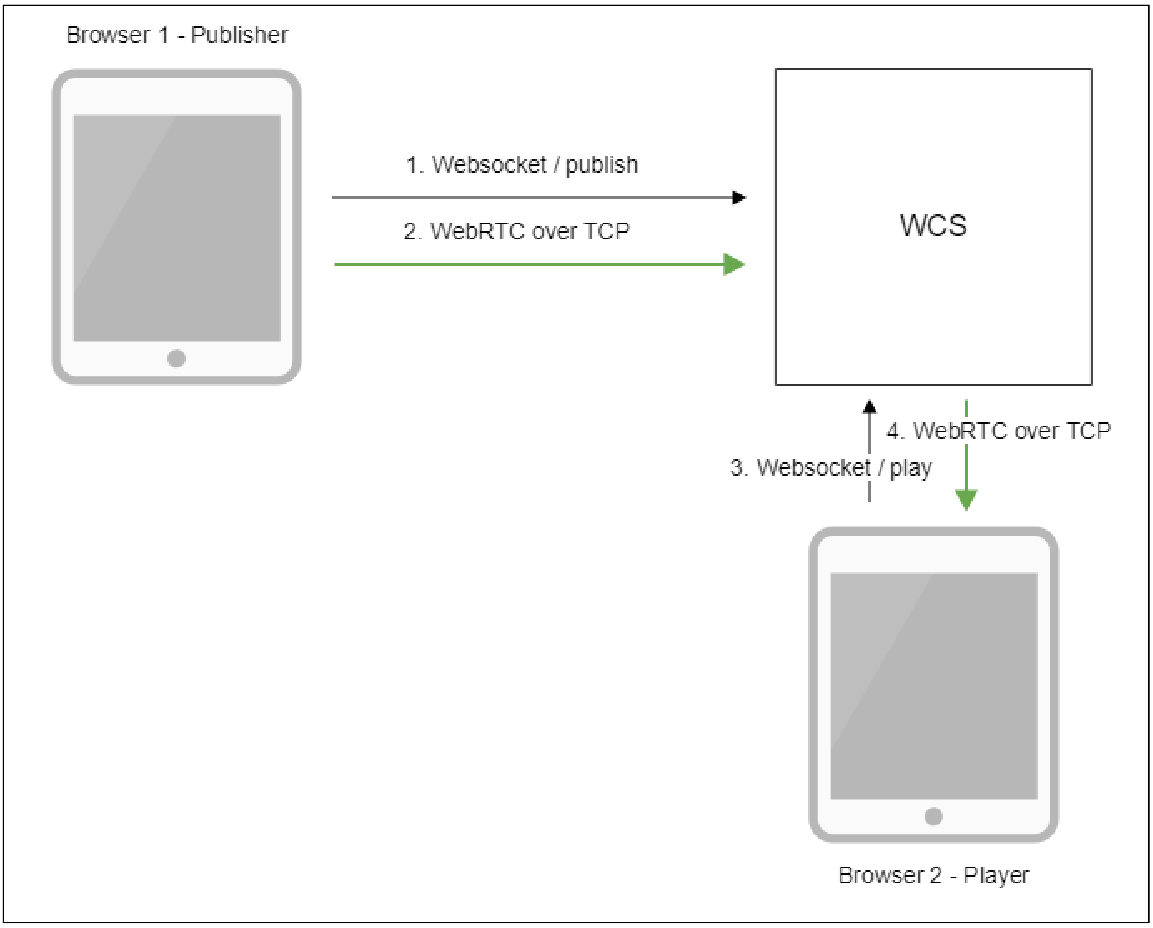
Usually, UDP is used on transport level to transfer WebRTC mediadata, this allows to reduce data transfer latency. On the other hand, high bitrate FullHD and 4K translations quality decreases even on relatively good channels due to packets loss.

If WebRTC translations quality is a must, WCS allows to use TCP on transport level according to RFC 4571 and 6544.

Supported platforms and browsers

	Chrome	Firefox	Safari	Edge
Windows	✅	✅	❌	✅
Mac OS	✅	✅	✅	✅
Android	✅	✅	❌	✅
iOS	✅	✅	✅	✅

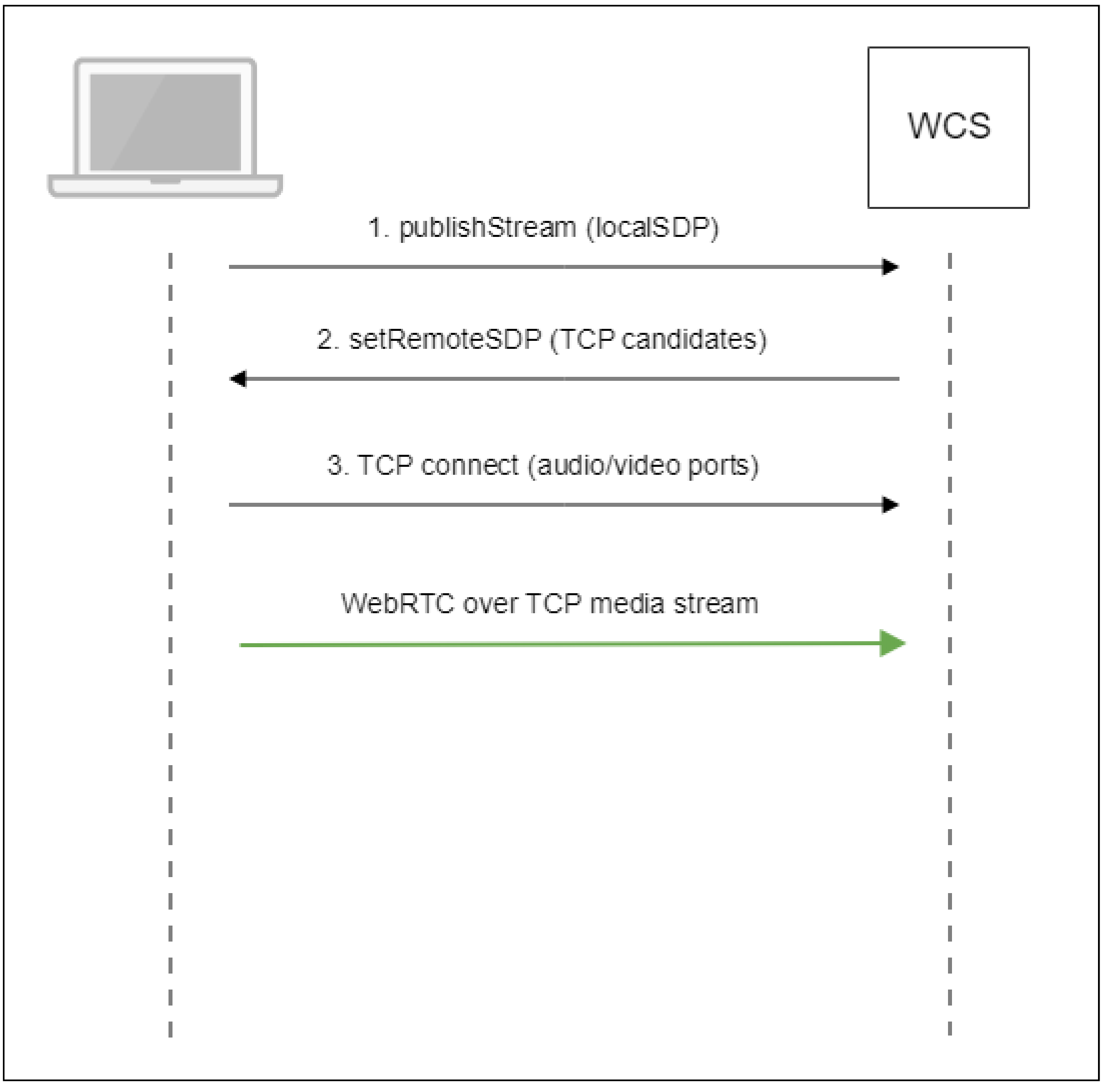
Operation flowchart



1. The browser connects to the server via the Websocket protocol and sends the `publishStream` command.
2. The browser captures the microphone and the camera and sends a WebRTC stream to the server over TCP.
3. The second browser establishes a connection also via Websocket and sends the `playStream` command.
4. The second browser receives the WebRTC stream over TCP and plays that stream on the page.

Call flow

When WebRTC stream is publishing over TCP, call flow slightly differs from steam publishing via UDP:



1. Client sends SDP offer to server via Websocket.
2. Client receives SDP with TCP ICE candidates from server:

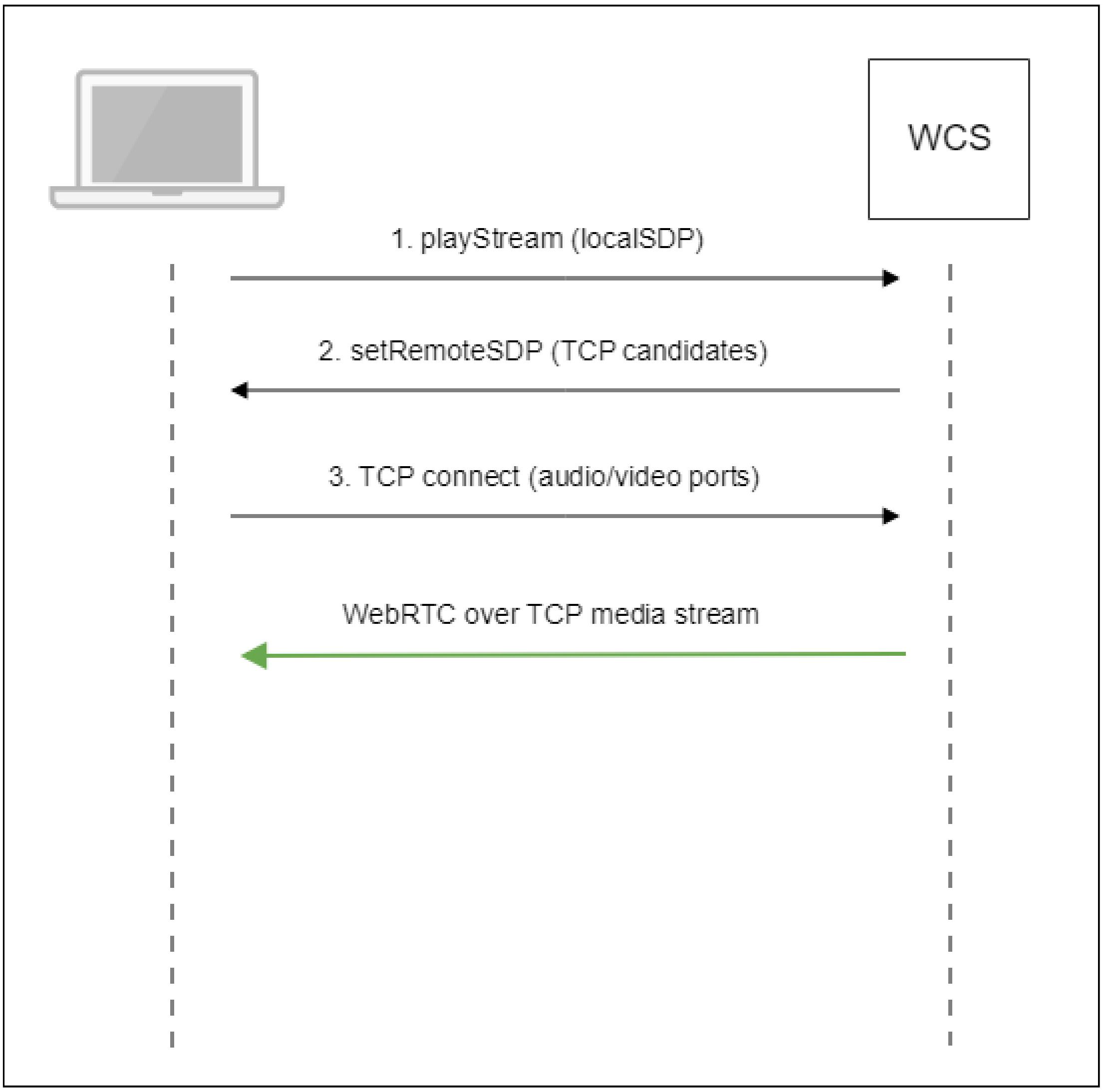
```
v=0
o=Flashphoner 0 1545364895231 IN IP4 192.168.1.5
s=Flashphoner/1.0
c=IN IP4 192.168.1.5
t=0 0
m=audio 31038 RTP/SAVPF 111 8 9
c=IN IP4 192.168.1.5
...
a=candidate:1 1 tcp 2130706431 192.168.1.5 31038 typ host tcptype passive
a=candidate:1 2 tcp 2130706431 192.168.1.5 31038 typ host tcptype passive
a=end-of-candidates
...
m=video 31040 RTP/SAVPF 100 127 102 125 96
c=IN IP4 192.168.1.5
...
a=candidate:1 1 tcp 2130706431 192.168.1.5 31040 typ host tcptype passive
a=candidate:1 2 tcp 2130706431 192.168.1.5 31040 typ host tcptype passive
a=end-of-candidates
```

```
a=rtcp-mux
a=rtcp:31040 IN IP4 192.168.1.5
a=sendonly
a=ssrc:564293803 cname:rtp/video/1b951110-04d5-11e9-a8b5-19c6b1a7cdbb
```

Where `192.168.1.5` is WCS server IP address

3. Client establishes TCP connection to audio and video data ports from SDP and starts to send mediadata.

Similarly, playback call flow goes as follows:



1. Client sends SDP offer to server via Websocket.
2. Client receives SDP with TCP ICE candidates from server.
3. Client establishes TCP connection to audio and video data ports from SDP and starts to receive mediadata.

Configuration

WebRTC over TCP usage is enabled with the following parameter in flashphoner.properties file

```
ice_tcp_transport=true
```

Adjusting send and receive buffers

Send and receive buffers sizes are set with the following parameters:

```
ice_tcp_send_buffer_size=1048576
ice_tcp_receive_buffer_size=1048576
```

By default, buffers sizes are set to 1 M.

Adjusting TCP queues

TCP queues high and low watermarks are set with the following parameters

```
ice_tcp_channel_high_water_mark=52428800
ice_tcp_channel_low_water_mark=5242880
```

By default, TCP queues size is between 5242880 and 52428800 bytes.

Ports used

TCP ports from WebRTC media ports data range are used for WebRTC connection over TCP

```
media_port_from         =31001
media_port_to           =32000
```

WebRTC transport management on client side

Server side settings enable WebRTC over TCP for all the clients by default. If necessary, TCP or UDP transoprt usage may be chosen on client side with WebSDK. To do this, transport to be used should be set in stream options when stream is created for publishing (via UDP for example)

```
session.createStream({
  name: streamName,
  display: localVideo,
  cacheLocalResources: true,
  receiveVideo: false,
  receiveAudio: false,
```

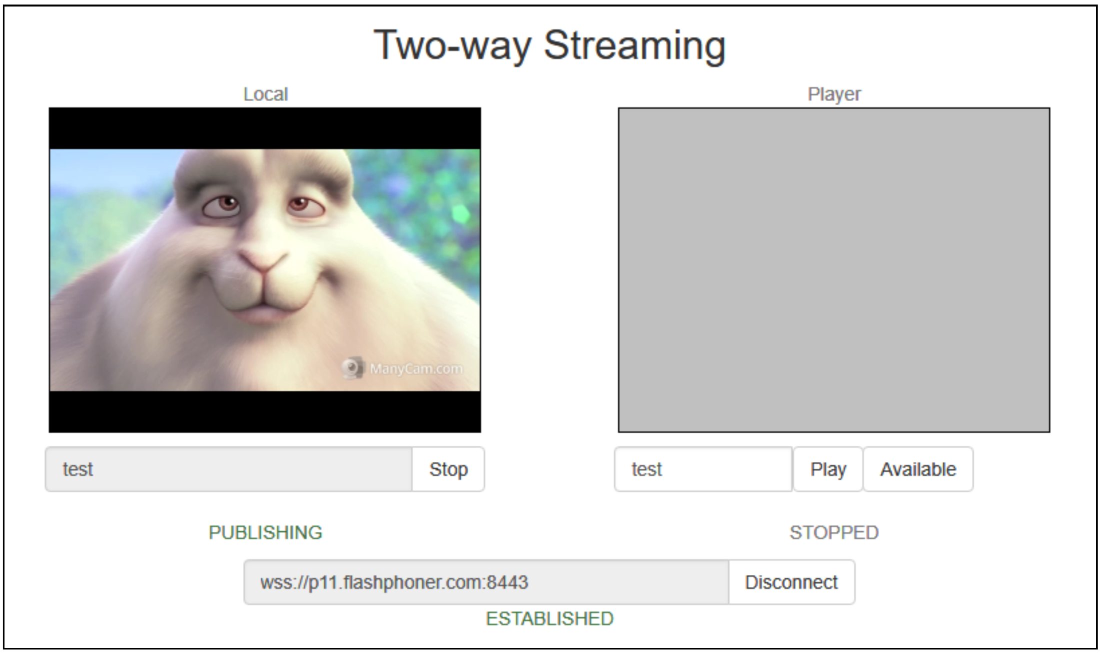
```
  transport: "UDP"
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
...
}).publish();
```

or playing (via TCP for example)

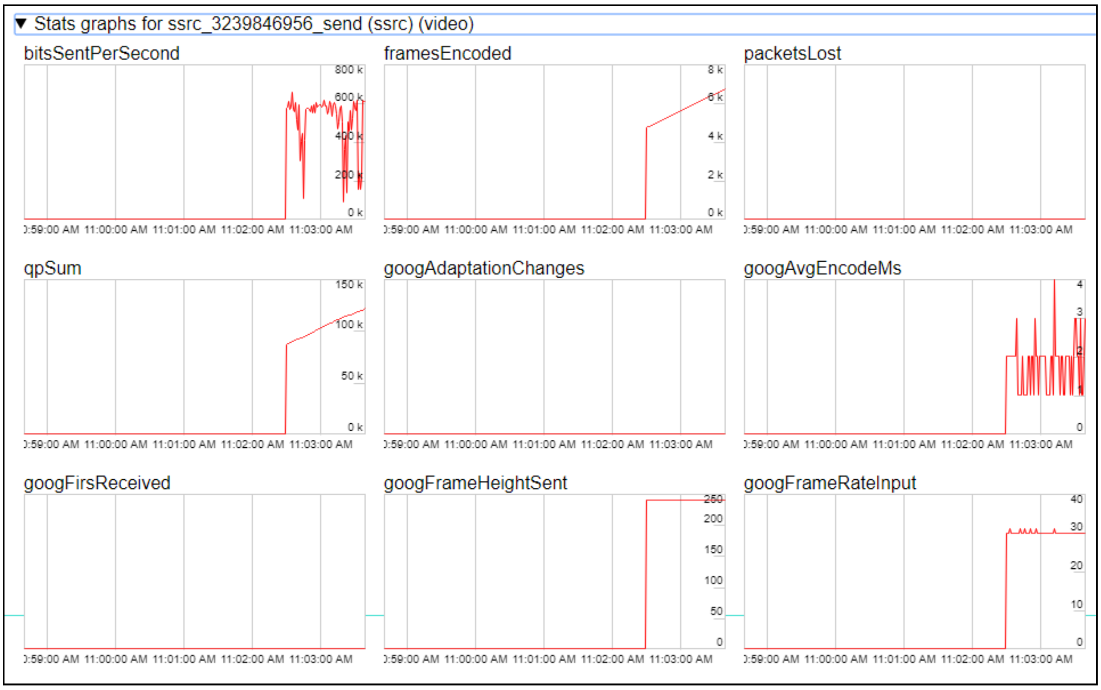
```
session.createStream({
  name: streamName,
  display: remoteVideo,
  transport: "TCP"
}).on(STREAM_STATUS.PENDING, function (stream) {
...
}).play();
```

Quick manual for testing

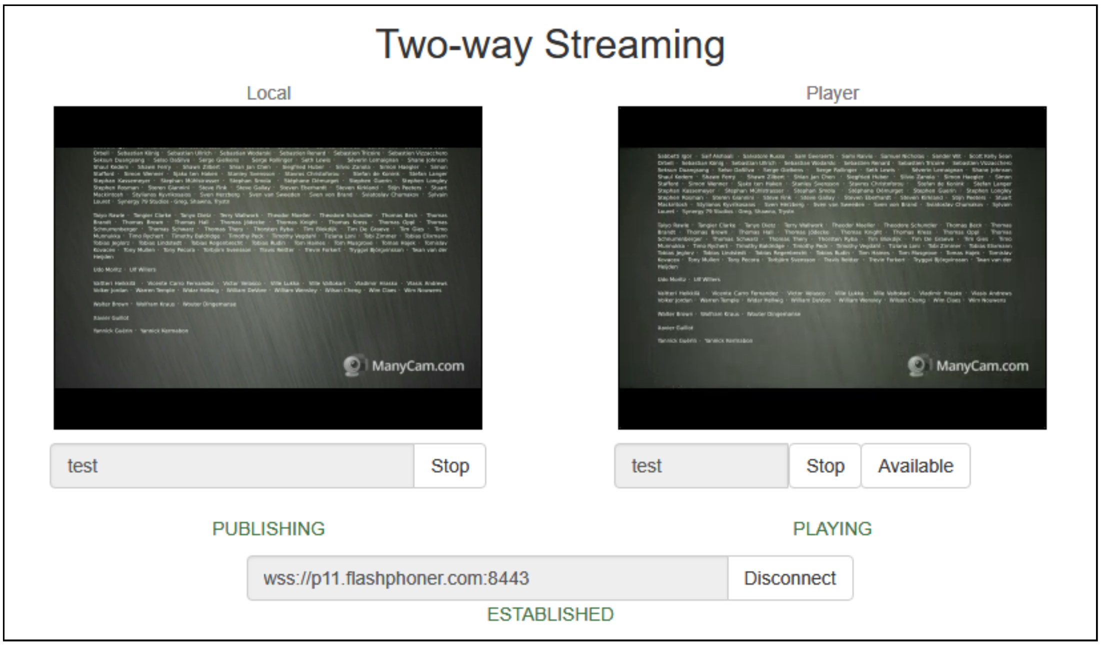
1. For test we use:
2. WCS server
3. Two Way Streaming web application to publish and play WebRTC stream in Chrome browser
4. Open Two Way Streaming web application, press `Connect`, enter stream name test and press `Publish`. Stream publishing will start



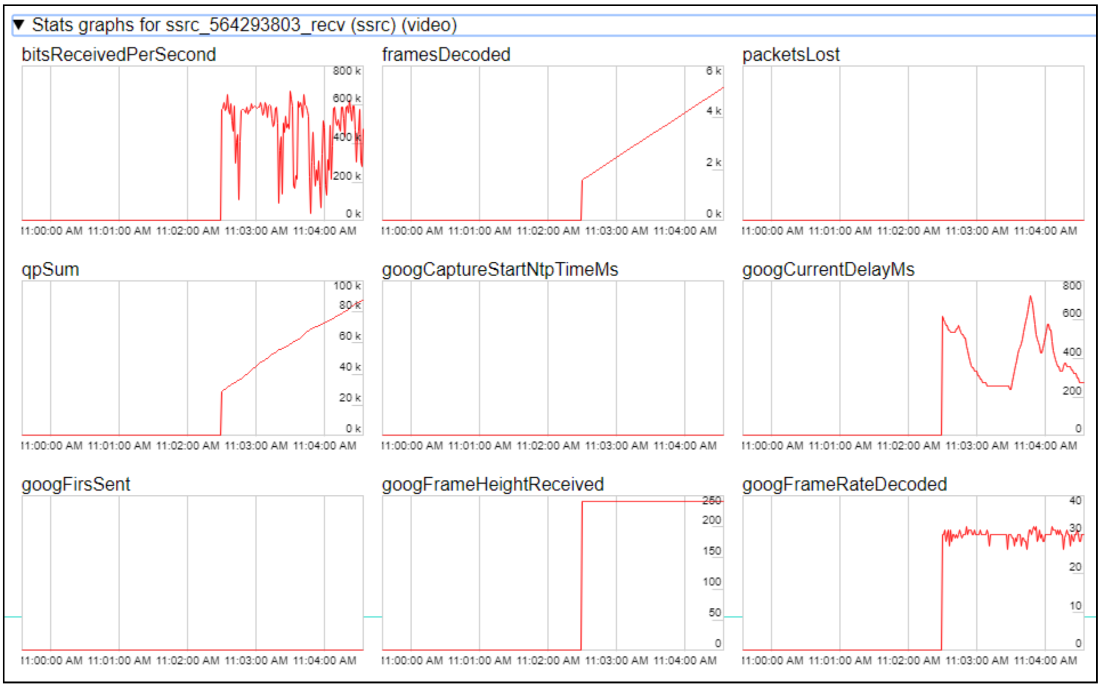
5. To make sure stream goes to server open `chrome://webrtc-internals`



6. In Player window enter stream name `test` and press `Play`. Stream playback will start



7. Playback graphs



8. To make sure TCP connection is established, launch this command on server

```
netstat -np | grep ESTABLISHED
```

The following lines will be in command output

```
# Websocket session
tcp        0      0 192.168.1.5:8443      192.168.1.100:60289
ESTABLISHED 7459/java
# publishing stream
tcp        0      0 192.168.1.5:31030     192.168.1.100:60305
ESTABLISHED 7459/java
tcp        0      0 192.168.1.5:31032     192.168.1.100:60307
ESTABLISHED 7459/java
# playing stream
tcp        0    112 192.168.1.5:31038     192.168.1.100:60515
ESTABLISHED 7459/java
tcp        0    817 192.168.1.5:31040     192.168.1.100:60517
ESTABLISHED 7459/java
```

Where

9. `192.168.1.5` is WCS server IP address
10. `192.168.1.100` is client IP address

Known issues

1. Some browsers do not establish WebRTC connection if additional network interfaces are enabled (VPN)

Symptoms

WebRTC stream publishing and playback do not work over TCP

Solution

Disable any additional network interfaces except this one used to access WSC server

2. WebRTC video cannot be played for all subscribers if one of the subscribers has a channel problems

Symptoms

Video cannot be played for all subscribers if one of the subscribers has a channel problems

Solution

Enable non-blocking IO with the following parameter

```
ice_tcp_nio=true
```

3. WebRTC over TCP requires more RAM comparing with UDP when non-blocking IO is used

Symptoms

With increasing traffic on the server, the RAM iconsumption increases strongly, up to the server shutdown

Solution

Add more RAM to the server according to the following recommendations

- 64 Gb RAM for 500 Mbps of traffic
- 128 Gb RAM for 1 Gbps of traffic