Публикация и воспроизведение потока по WebRTC через TCP

Описание

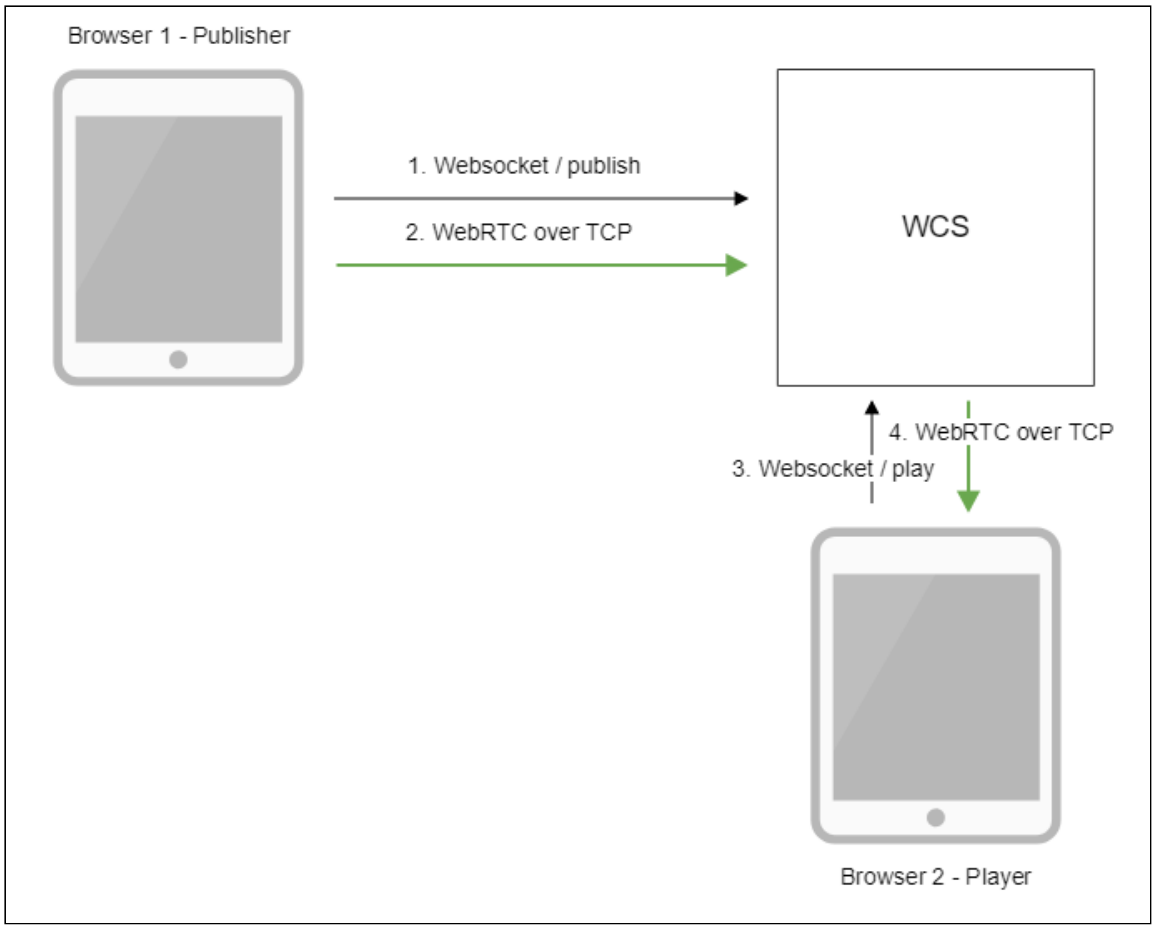
Как правило, для передачи WebRTC медиаданных на транспортном уровне используется UDP, это позволяет снизить задержки при передаче данных. С другой стороны, в результате потерь пакетов, качество высокобитрейтных FullHD и 4K трансляций снижается даже на относительно хороших каналах.

Если необходимо обеспечить качество WebRTC-трансляций, WCS позволяет использовать TCP на транспортном уровне в соответствии с RFC 4571 и 6544.

Поддерживаемые платформы и браузеры

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

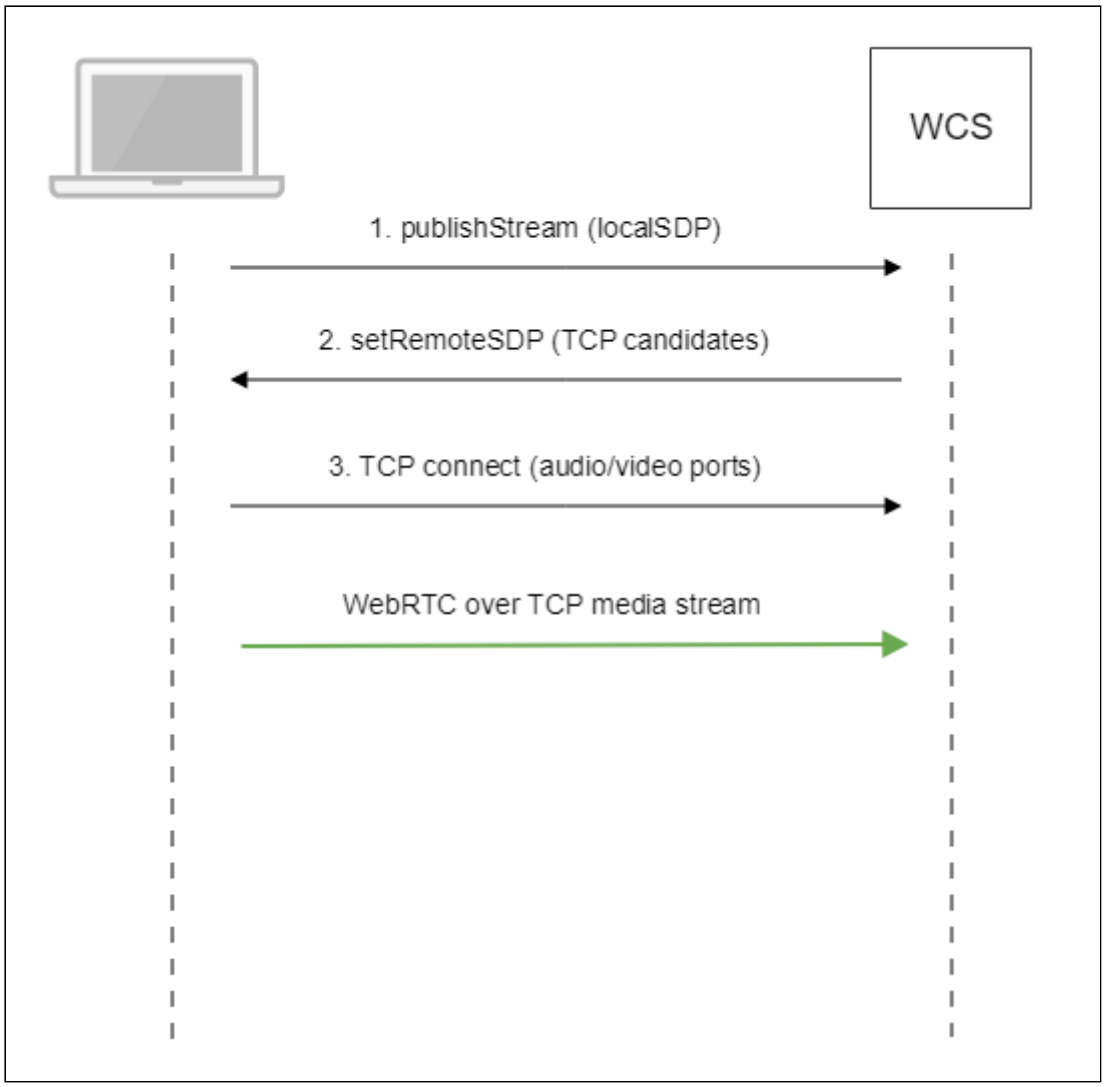
Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер через TCP.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду `playStream`.
4. Второй браузер получает WebRTC поток через TCP и воспроизводит этот поток на странице.

Последовательность выполнения операций

Последовательность выполнения операций немного отличается от публикации потока по WebRTC в части установки WebRTC соединения:



1. Клиент отправляет серверу предложение локального SDP по Websocket.
2. Клиент получает SDP от сервера. В SDP указываются TCP ICE-кандидаты:

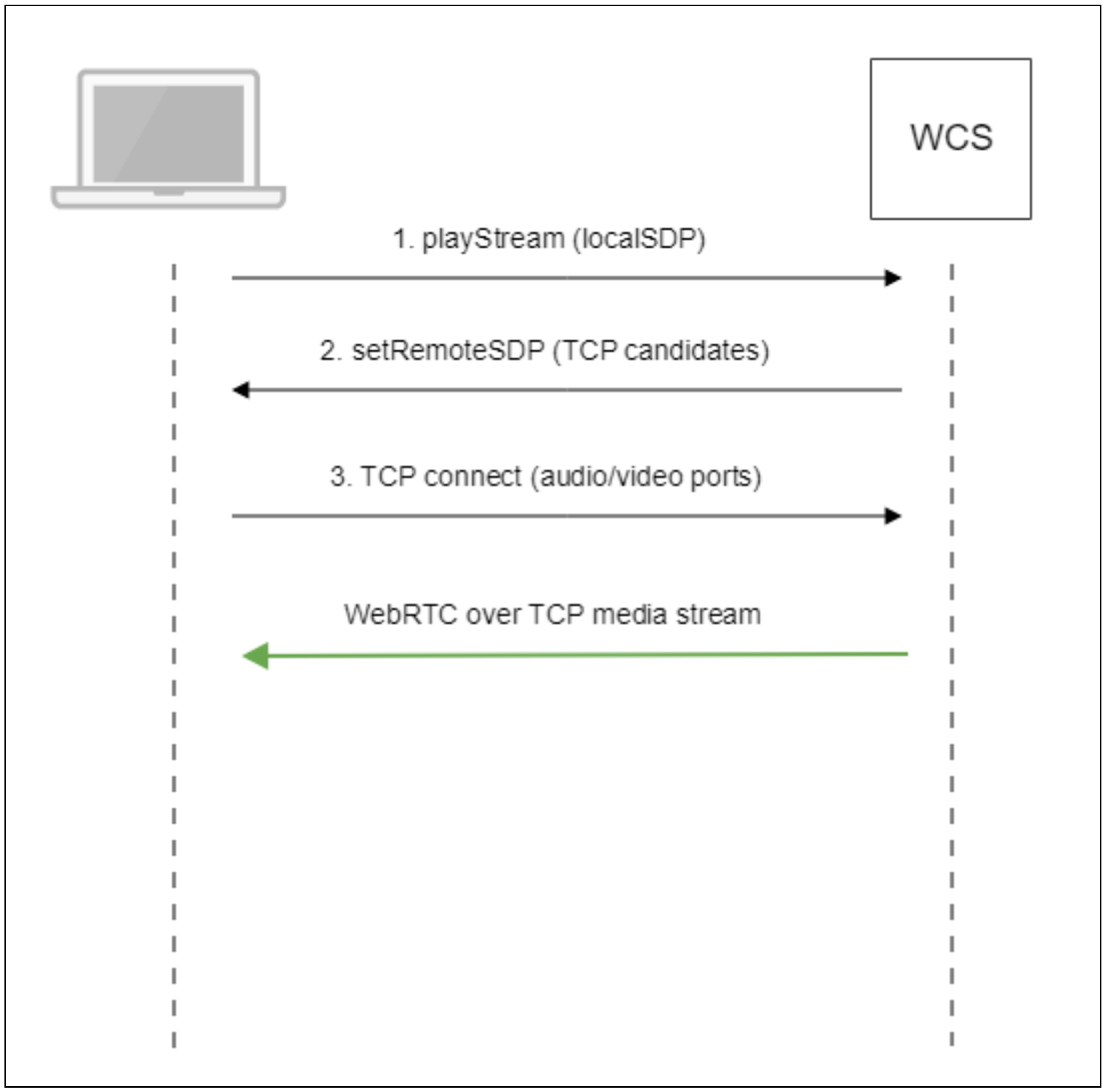
```
v=0
o=Flashphoner 0 1545364895231 IN IP4 192.168.1.5
s=Flashphoner/1.0
c=IN IP4 192.168.1.5
t=0 0
m=audio 31038 RTP/SAVPF 111 8 9
c=IN IP4 192.168.1.5
...
a=candidate:1 1 tcp 2130706431 192.168.1.5 31038 typ host tcptype passive
a=candidate:1 2 tcp 2130706431 192.168.1.5 31038 typ host tcptype passive
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
```

Здесь `192.168.1.5` - IP адрес WCS сервера

3. Клиент устанавливает TCP соединение на порты для аудио и видео данных, указанные в SDP, и начинает передачу медиаданных.

Аналогично, последовательность выполнения операций при воспроизведении потока будет следующей:



1. Клиент отправляет серверу предложение локального SDP по Websocket.
2. Клиент получает SDP от сервера. В SDP указываются TCP ICE-кандидаты.
3. Клиент устанавливает TCP соединение на порты для аудио и видео данных, указанные в SDP, и начинает прием медиаданных.

Настройка

Использование WebRTC через TCP включается настройкой в файле flashphoner.properties

```
ice_tcp_transport=true
```

Управление размерами буферов на прием и передачу

Размеры буферов для приема и отправки данных настраиваются при помощи следующих параметров:

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

По умолчанию, размеры буферов установлены в 1 Мб.

Управление размерами очередей

Размеры очередей TCP пакетов ограничиваются сверху и снизу настройками

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

По умолчанию, допустимый размер очередей находится между 5242880 и 52428800 байтами

Использование портов

Для WebRTC через TCP используются TCP порты с номерами из диапазона, выделенного для WebRTC медиа портов

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

Управление WebRTC транспортом на стороне клиента

Настройка на стороне сервера включает использование WebRTC через TCP по умолчанию для всех клиентов. При необходимости, использование TCP или UDP транспорта может быть выбрано на стороне клиента при помощи WebSDK. Для этого необходимо указать используемый транспорт в опциях потока при его создании для публикации (например, по UDP)

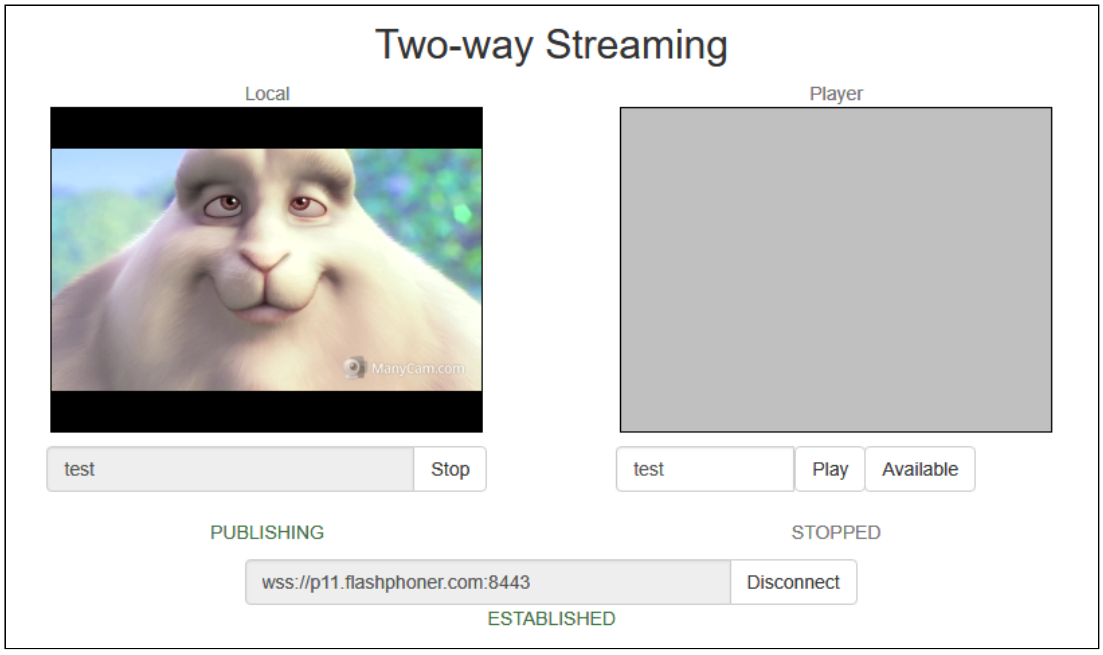
```
session.createStream({
  name: streamName,
  display: localVideo,
  cacheLocalResources: true,
  receiveVideo: false,
  receiveAudio: false,
  transport: "UDP"
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
...
}).publish();
```

или воспроизведения (например, по TCP)

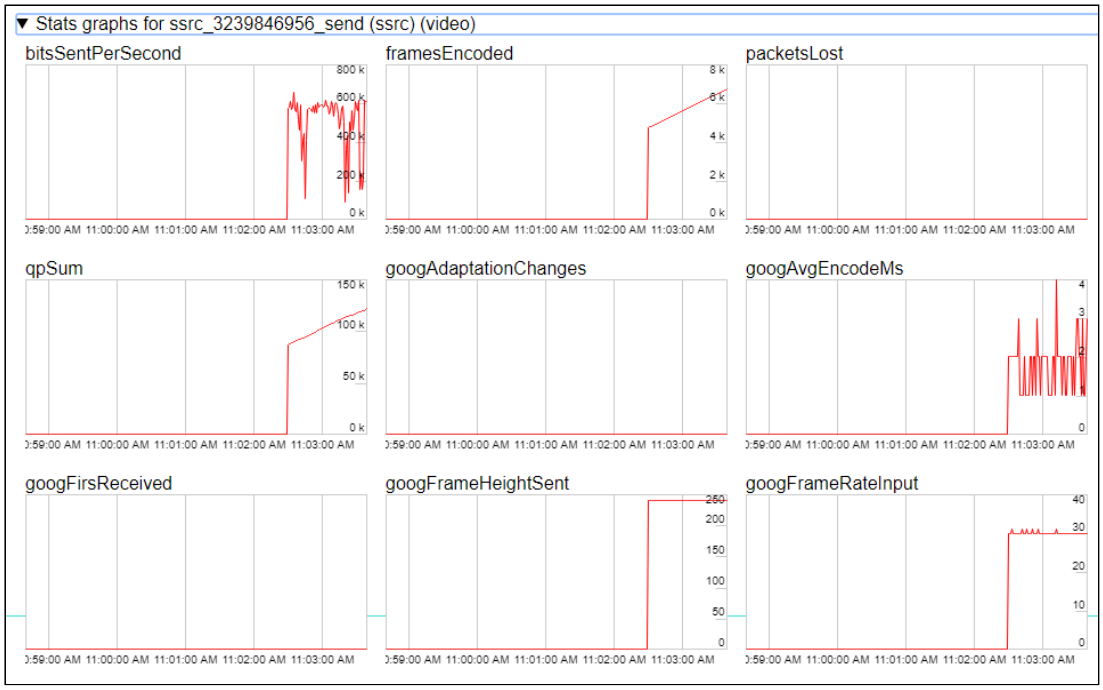
```
session.createStream({
  name: streamName,
  display: remoteVideo,
  transport: "TCP"
}).on(STREAM_STATUS.PENDING, function (stream) {
...
}).play();
```

Краткое руководство по тестированию

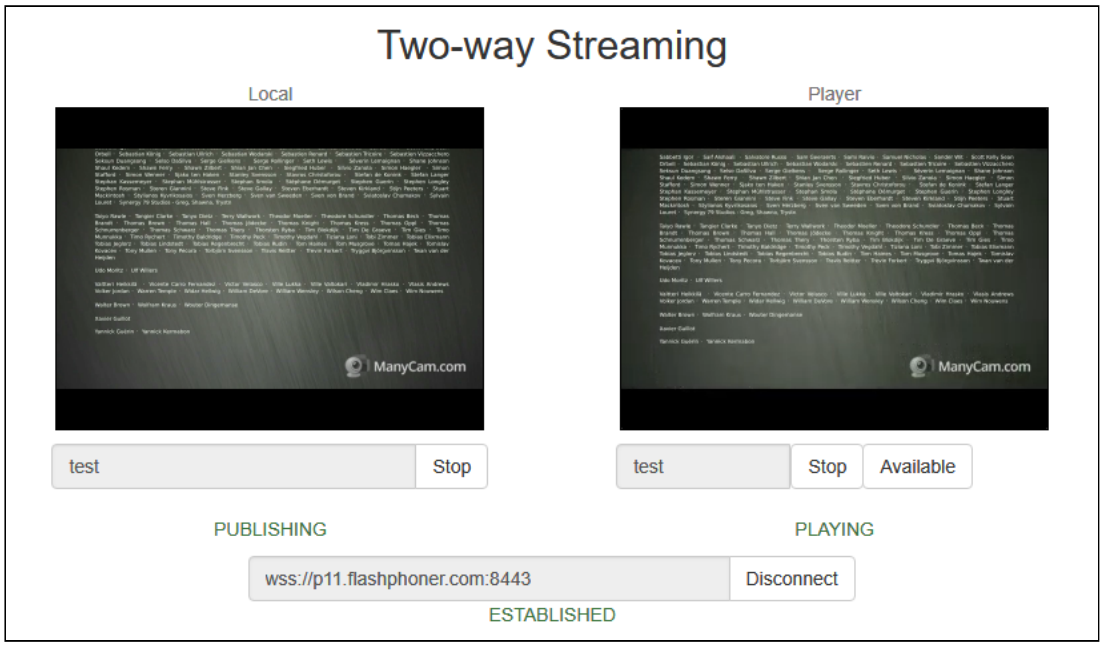
1. Для теста используем:
2. WCS сервер
3. веб-приложение Two Way Streaming для публикации и воспроизведения потока в браузере Chrome
4. Откройте веб приложение Two Way Streaming, нажмите `Connect`, введите имя потока test и нажмите `Publish`. Начнется публикация потока



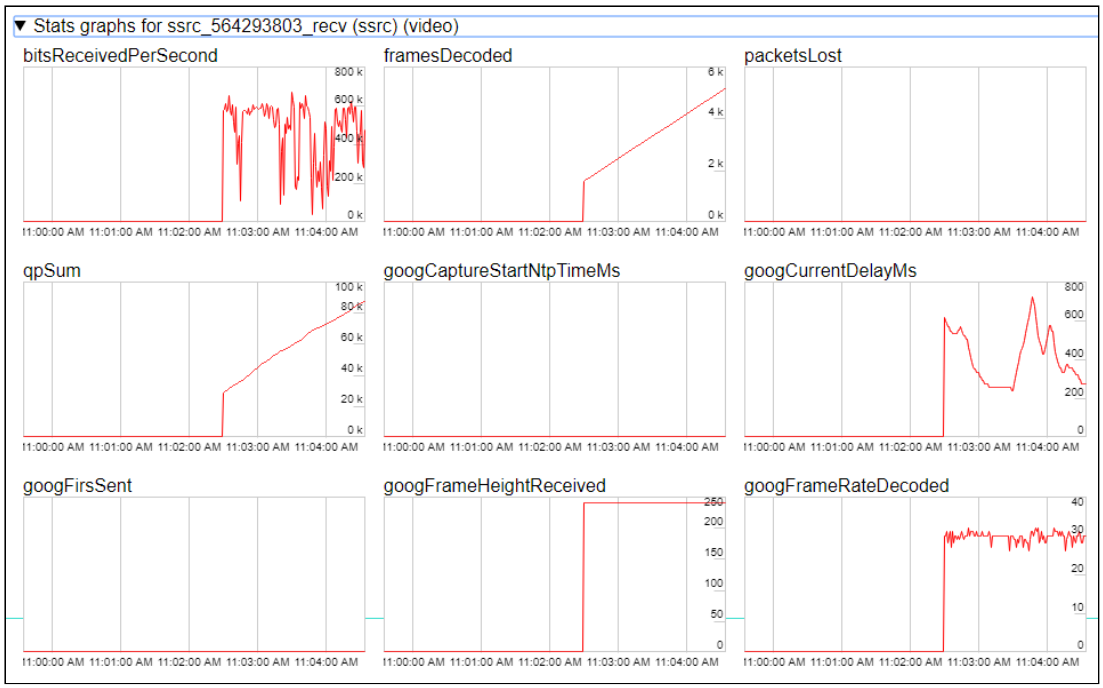
5. Чтобы убедиться, что поток отправляется на сервер, откройте `chrome://webrtc-internals`



6. В окне Player введите имя потока `test` и нажмите `Play`. Начнется воспроизведение публикуемого потока



7. Графики воспроизведения



8. Чтобы убедиться, что установлено TCP соединение, выполните на сервере команду

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

Результатом выполнения будут следующие строки

```
# 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
```

Здесь

9. 192.168.1.5 - IP адрес WCS сервера
10. 192.168.1.100 - IP адрес клиента

Известные проблемы

1. WebRTC соединение не устанавливается в некоторых браузерах, если на ПК используется дополнительный сетевой интерфейс (VPN)

Симптомы

Публикация и воспроизведение WebRTC через TCP не работают

Решение

Отключить все дополнительные сетевые интерфейсы, кроме имеющего доступ к WCS серверу.

2. WebRTC видео может не воспроизводиться по TCP у всех подписчиков потока, если у одного из подписчиков проблемы на канале

Симптомы

Не воспроизводится видео у всех подписчиков потока, если у одного из подписчиков проблемы на канале

Решение

Использовать non-blocking IO при помощи настройки

```
ice_tcp_nio=true
```

3. WebRTC по TCP требует больше оперативной памяти по сравнению с UDP при использовании non-blocking IO

Симптомы

При возрастании трафика на сервере, резко растет потребление оперативной памяти, вплоть до завершения работы сервера

Решение

Увеличить количество оперативной памяти на сервере, исходя из расчета

- 64 Gb RAM на 500 Мбит/с трафика
- 128 Gb RAM на 1 Гбит/с трафика