В браузере по Websocket + Canvas, WSPlayer

Сочетание технологий WebSocket + HTML5 Canvas целесообразно использовать для воспроизведения видеопотока в случаях, когда браузер клиента не поддерживает WebRTC, и при этом необходимо обеспечить минимальные задержки. Подробное описание плеера WSPlayer, построенного на данном сочетании технологий, можно найти в этой статье.

Описание

Не все браузеры поддерживают технологию WebRTC. Например, основной способ доставки Live-видеопотока в браузер Safari под iOS 9 и iOS 10 — это HLS (HTTP Live Streaming). Данный протокол может давать дает задержку более 15 секунд.

Web Call Server отдает видеопоток на браузер по протоколу Websocket, что позволяет сократить задержку до 1-3 секунд и дает видео реального времени по сравнению с HLS. Поток воспроизводится в браузере при помощи элемента HTML5 Canvas.

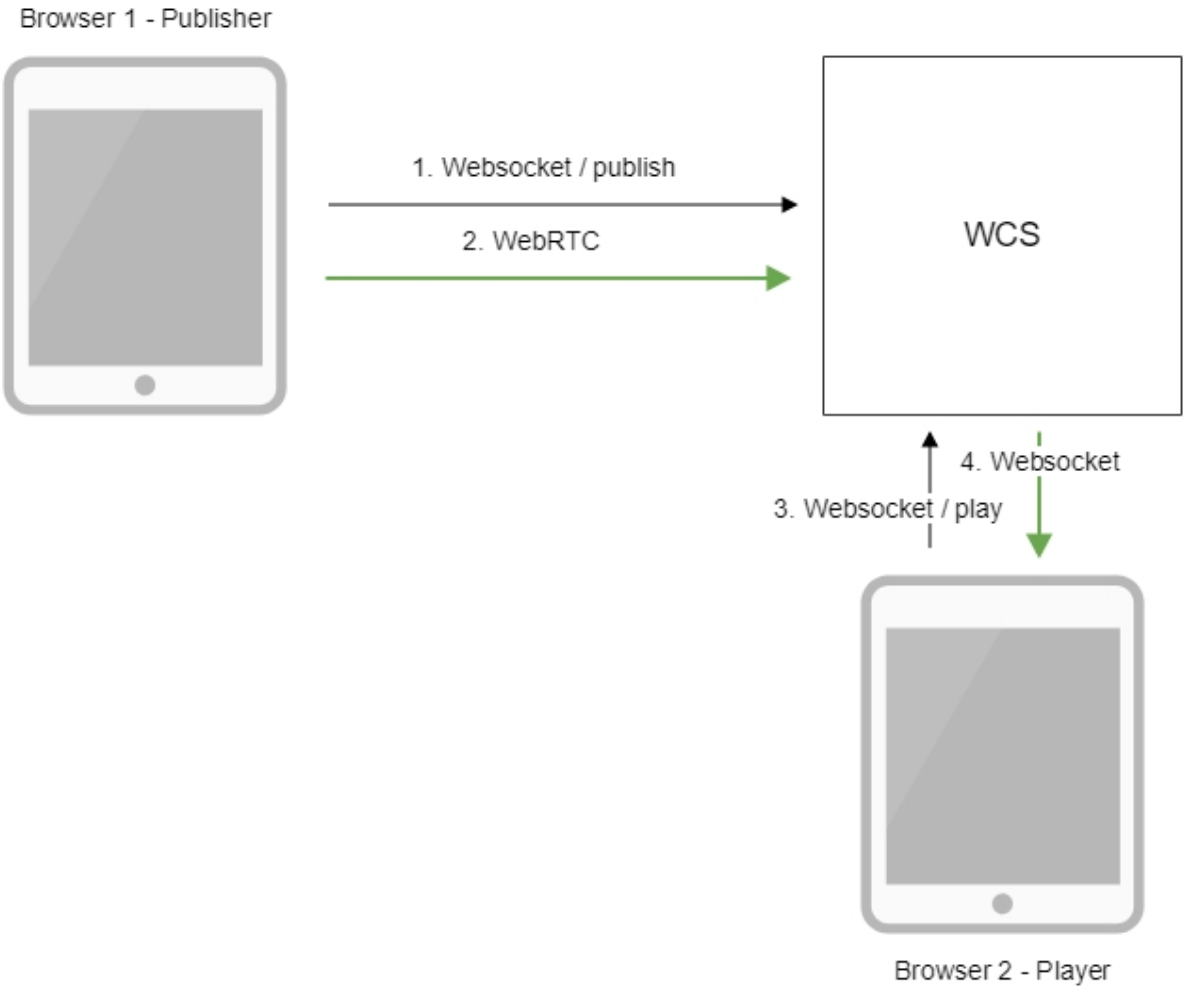
Поддерживаемые платформы и браузеры

	Chrome	Firefox	Safari 11	Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		
iOS	-	-	+	

Поддерживаемые кодеки

- Видео: MPEG
- Аудио: G.711

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду play.
4. Второй браузер получает MPEG + G.711 поток по Websocket и воспроизводит этот поток при помощи HTML5 Canvas на странице.

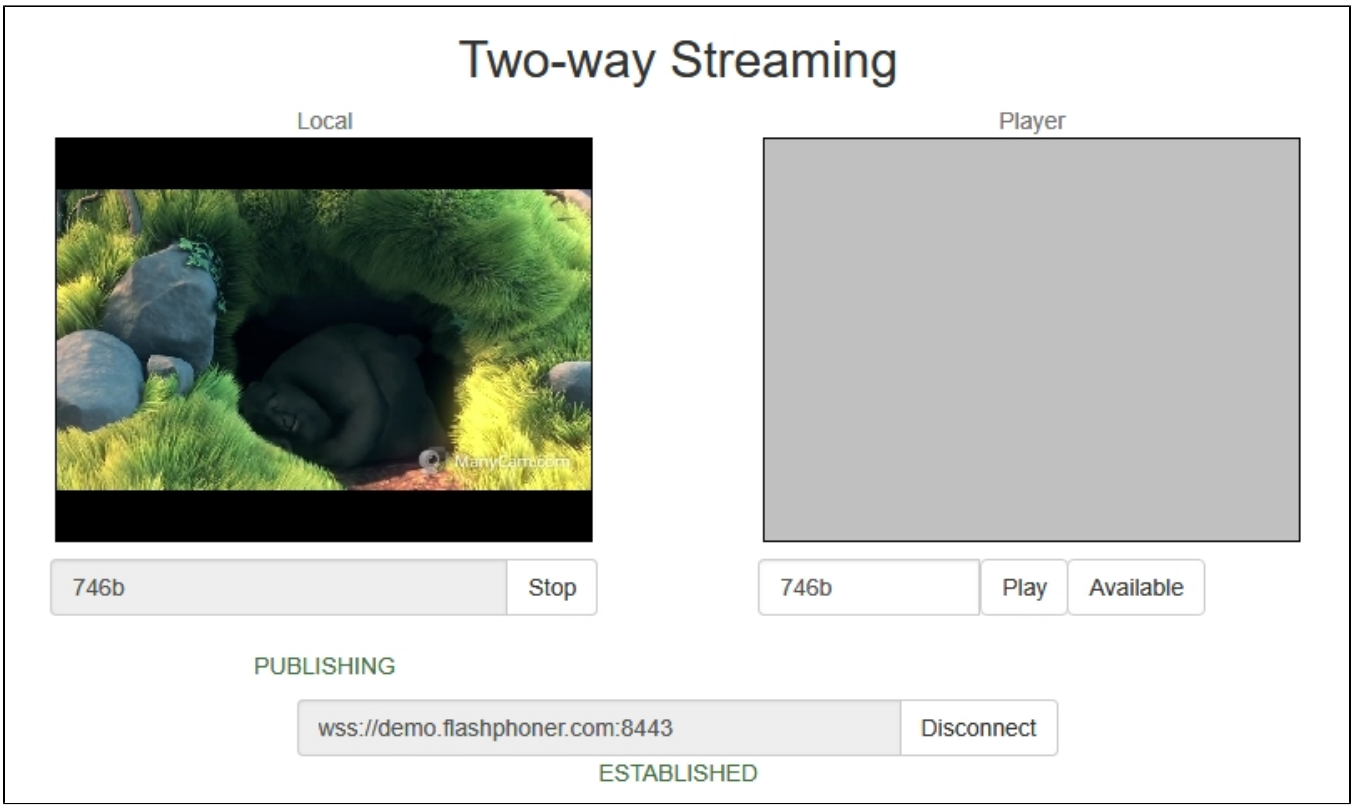
Краткое руководство по тестированию

Трансляция видеопотока на сервер и воспроизведение его по HTML5 + Canvas (WSPlayer) в браузере

1. Для теста используем:

- демо-сервер demo.flashphoner.com;
- веб-приложениеTwo Way Streamingдля публикации потока
- веб-приложениеPlayerдля воспроизведения потока

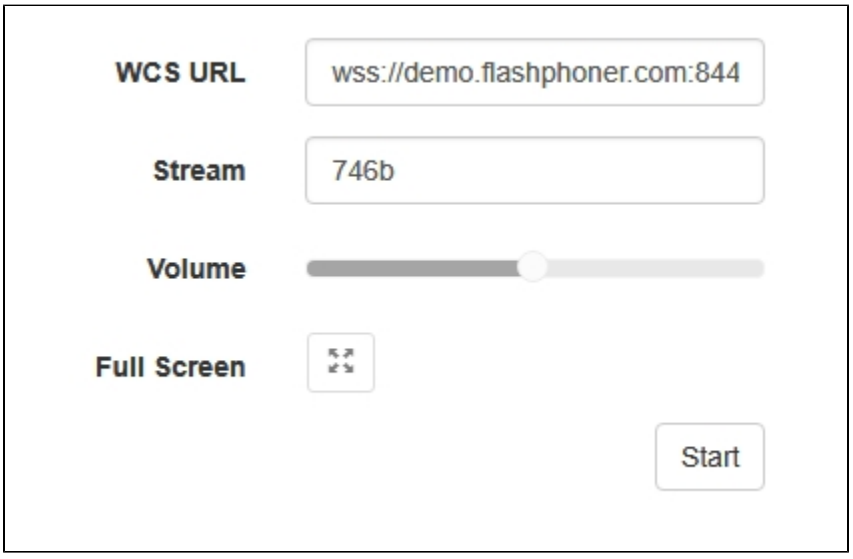
2. Откройте веб-приложение Two Way Streaming. Нажмите Connect, затем Publish. Скопируйте идентификатор потока:



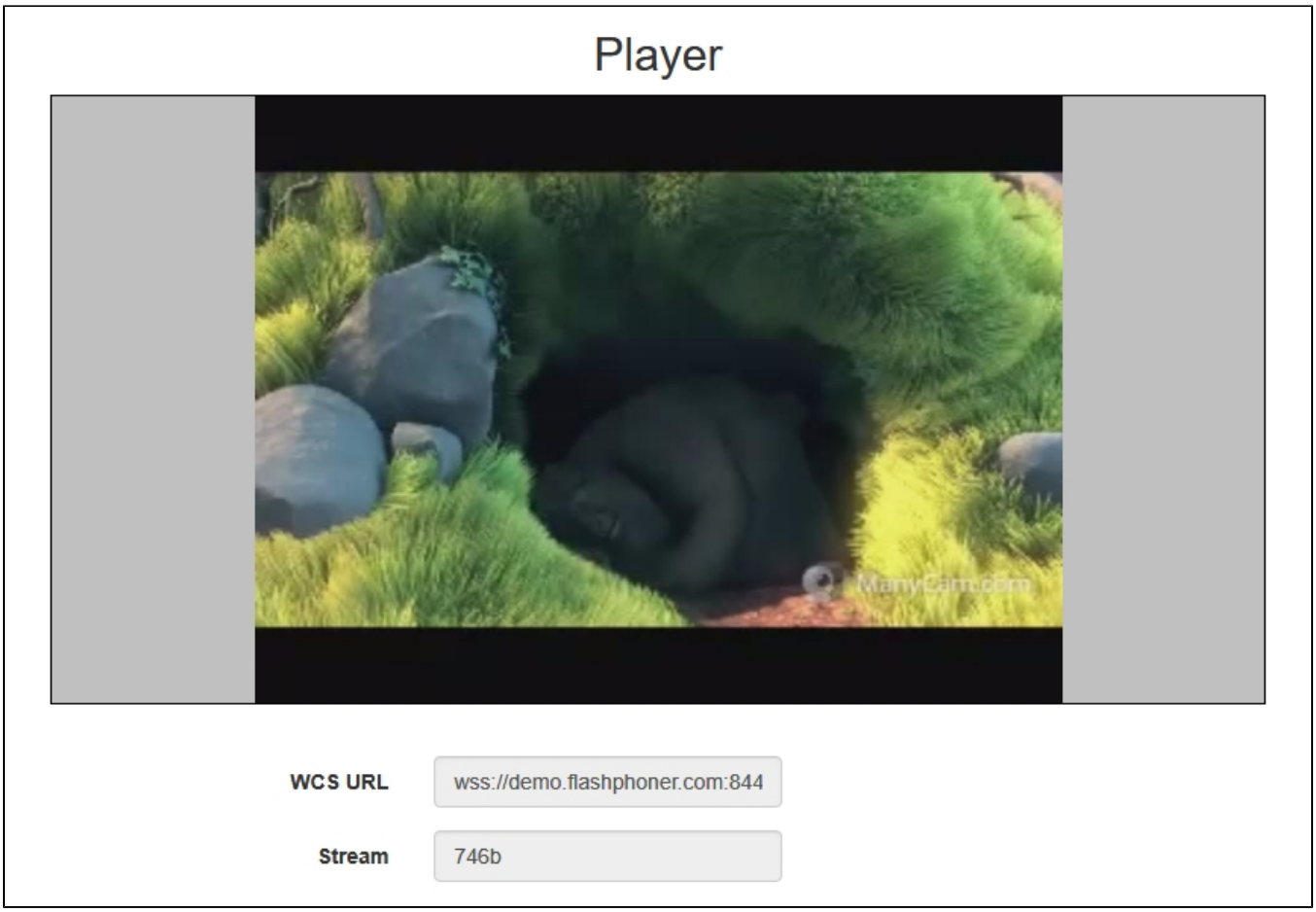
3. Откройте веб-приложение Player, указав в параметрах URL WSPlayer

https://demo.flashphoner.com/client2/examples/demo/streaming/player/player.html?mediaProvider=WSPlayer

4. Укажите в поле Stream идентификатор потока:



5. Нажмите кнопку Start. Начнется воспроизведение потока:

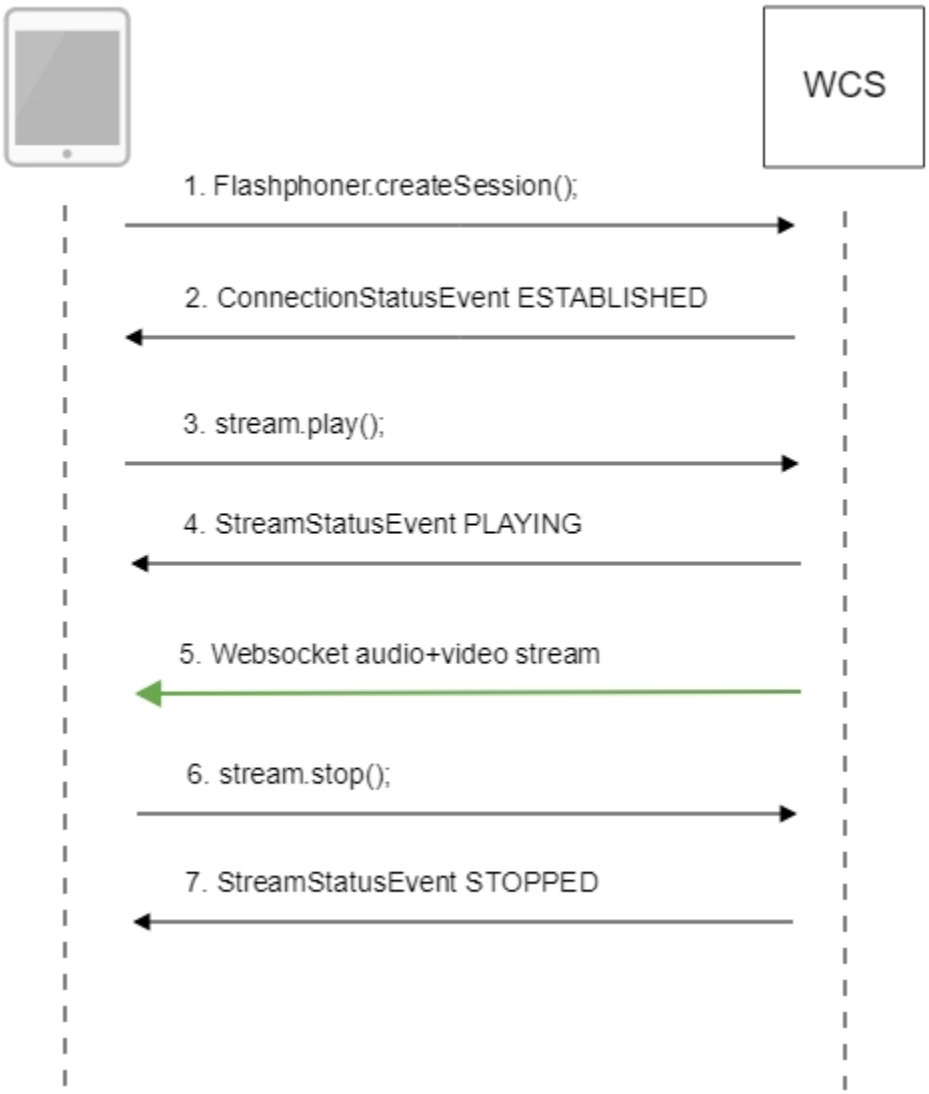


Последовательность выполнения операций (Call flow)

Ниже описана последовательность вызовов при использовании примера Player для воспроизведения потока при помощи WSPLayer

player.html

player.js



1. Установка соединения с сервером.

init():code

```
    if (Flashphoner.getMediaProviders()[0] == "WSPlayer") {
        resolution_for_wsplayer = {playWidth:640,playHeight:480};
    }
```

Flashphoner.createSession();code

```
    Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
        setStatus(session.status());
        //session connected, start playback
        playStream(session);
    }).on(SESSION_STATUS.DISCONNECTED, function(){
        setStatus(SESSION_STATUS.DISCONNECTED);
        onStopped();
    }).on(SESSION_STATUS.FAILED, function(){
        setStatus(SESSION_STATUS.FAILED);
        onStopped();
    });
```

2. , .

ConnectionStatusEvent ESTABLISHED code

```
    Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
        setStatus(session.status());
        //session connected, start playback
        playStream(session);
    }).on(SESSION_STATUS.DISCONNECTED, function(){
        ...
    }).on(SESSION_STATUS.FAILED, function(){
        ...
    });
```

3. Воспроизведение потока.

stream.play();code

```
    if (Flashphoner.getMediaProviders()[0] === "MSE" && mseCutByIFrameOnly) {
        ...
    }
    if (resolution_for_wsplayer) {
        options.playWidth = resolution_for_wsplayer.playWidth;
        options.playHeight = resolution_for_wsplayer.playHeight;
    } else if (resolution) {
        ...
    }
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    });
    stream.play();
```

4. Получение от сервера события, подтверждающего успешное воспроизведение потока.

StreamStatusEvent, статус PLAYINGcode

```
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    }).on(STREAM_STATUS.PLAYING, function(stream) {
        $("#preloader").show();
        setStatus(stream.status());
        onStarted(stream);
    }).on(STREAM_STATUS.STOPPED, function() {
        ...
    }).on(STREAM_STATUS.FAILED, function(stream) {
        ...
    }).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
        ...
    });
    stream.play();
```

5. Прием аудио-видео потока по Websocket и воспроизведение по MSE

6. Остановка воспроизведения потока.

stream.stop();code

```
function onStarted(stream) {
    $("#playBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        stream.stop();
    }).prop('disabled', false);
    ...
}
```

7. Получение от сервера события, подтверждающего остановку воспроизведения потока.

StreamStatusEvent, статус STOPPEDcode

```
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    }).on(STREAM_STATUS.PLAYING, function(stream) {
        ...
    }).on(STREAM_STATUS.STOPPED, function() {
        setStatus(STREAM_STATUS.STOPPED);
        onStopped();
    }).on(STREAM_STATUS.FAILED, function(stream) {
        ...
    }).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
        ...
    });
    stream.play();
```

Известные проблемы

1. В iOS при воспроизведении по WSPlayer не поддерживается переключение в полноэкранный режим

Симптомы: вызов функции Stream.fullScreen() не приводит к переходу в полноэкранный режим при использовании WSPlayer

Решение: при возможности, обновить устройство до последней версии iOS и использовать WebRTC в браузере Safari

2. WSPlayer не поддерживает воспроизведение потоков без видео составляющей

Симптомы: поток только с аудио не играет, нет звука, при этом поток в статусе PLAYING

Решение:использовать потоки видео+аудио, при необходимости заглушать видео (будет отображаться черный экран)

3. Нельзя воспроизвести два потока по WSPlayer через одно Websocket соединение на одной странице

Симптомы: в примере 2Players не играют два потока при подключении по HTTP в основных браузерах (Chrome, Firefox, Safari)

Решение: использовать отдельное Websocket соединение для каждого потока на одной странице при воспроизведении по WSPlayer