Flash Streaming

Пример стримера и плеера в native Flash / Flex приложении

Данный пример показывает как воспроизводить видеопоток с одновременной публикацией другого потока, используя клиентское Flash приложение, которое может быть запущено простым swf-файлом. Стриминг в данном примере может работать по двум протоколам `rtmp://` и `rtmfp://`

На скриншоте показан пример Flash приложения у которого видеопоток отправляется на сервер и воспроизводится с сервера по протоколу RTMFP.

Поле `Server` содержит RTMFP адрес сервера для установки соединения. Поле `Publish` содержит имя отправляемого на сервер видеопотока с веб-камеры. Поле `Play` содержит

имя видеопотока для воспроизведения с сервера. Ниже можно задать дополнительные параметры видеозахвата и отправки видеопотока:

- ширина и высота кадра
- частота кадров
- качество
- частота ключевых кадров
- наличие аудио или видео составляющей в отправляемом потоке

Файлы примера

Пример представляет собой скомпилированный SWF-файл на HTML-странице, с использованием Flex / ActionScript3 и MXML и находится по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/flash_client/streaming.html

- streaming.html - страница примера
- streaming/bin/streaming.swf - файл приложения

Работа с исходным кодом примера

Для разбора кода возьмем версию файла `streaming.mxml` с хешем `90eb5073687bbe63bbb7467de3f3be4f3fe33802`, который находится здесь. Результатом сборки `streaming.mxml` файла является приложение примера `streaming.swf`. Скомпилированный swf и исходный код доступен для скачивания в соответствующей сборке 0.5.3.1894.

1. Доступ к камере и микрофону

Сразу после загрузки приложения получаем доступ к веб-камере и микрофону

line 42

line 44

```
cam = Camera.getCamera();
videoMy.attachCamera(cam);
mic = Microphone.getEnhancedMicrophone();
```

2. Настройка камеры и микрофона

Применяем настройки камеры и микрофона

line 76

line 84

Рекомендуемая настройка для веб-камеры:

- Порог чувствительности движения в кадре для отправки видео: cam.setMotionLevel(0,2000);

Рекомендуемые настройки для микрофона:

- Аудио кодек Speex: mic.codec = SoundCodec.SPEEX;
- Количество фреймов на один пакет: mic.framesPerPacket=1;
- Порог чувствительности звука для отправки аудио: mic.setSilenceLevel(0,2000);

```
private function initCam():void{
    cam.setMode(int(camWidth.text),int(camHeight.text),int(camFPS.text),true);
    cam.setQuality(0,int(camQuality.text));
    cam.setKeyFrameInterval(int(camKeyFrame.text));
    cam.setMotionLevel(0,2000);
    Logger.info("Cam initizlized "+cam.width+"x"+cam.height);
}

private function initMic():void{
    var options:MicrophoneEnhancedOptions = new MicrophoneEnhancedOptions();
    options.mode = MicrophoneEnhancedMode.FULL_DUPLEX;
    options.echoPath = 128;
    options.nonLinearProcessing = true;
    mic.codec = SoundCodec.SPEEX;
    mic.encodeQuality = 5;
    mic.framesPerPacket=1;
    mic.gain=50;
    mic.setSilenceLevel(0,2000);
    mic.enhancedOptions = options;
    Logger.info("Mic initialized");
}
```

3. Установка соединения с сервером

line 103

Здесь мы устанавливаем соединение с сервером и передаем `obj.appKey = "flashStreamingApp"`; Этот appKey дает серверу понять, что он имеет дело с standalone Flash приложением, а не с Websocket/WebRTC клиентом

```
private function connect():void{
    trace("connect");
    var url:String = connectUrl.text;
    nc = new NetConnection();
    //if (url.indexOf("rtmp") == 0){
    nc.objectEncoding = ObjectEncoding.AMF0;
    //}
    nc.client = this;
    nc.addEventListener(NetStatusEvent.NET_STATUS, handleConnectionStatus);
    var obj:Object = new Object();
    obj.login = generateRandomString(20);
    obj.appKey  = "flashStreamingApp";
    nc.connect(url,obj);
}
```

4. Публикация потока

Отправка потока на сервер происходит в методе примера `publish()`

line 165

```
if (publishAudio.selected){
    initMic();
    publishStream.attachAudio(mic);
    Logger.info("Init audio stream")
}
if (publishVideo.selected){
    initCam();
    publishStream.attachCamera(cam);
    addH264();
    Logger.info("Init video stream");
}
addListenerAndPublish
```

Непосредственно перед отправкой потока, для него задаются дополнительные параметры буферизации и кодека H.264 в методах `addH264()` и `addListenerAndPublish()`

line 199

line 208

```
private function addListenerAndPublish():void{
    publishStream.videoReliable=true;
    publishStream.audioReliable=false;
    publishStream.useHardwareDecoder=true;
    publishStream.addEventListener(NetStatusEvent.NET_STATUS, handleStreamStatus);
    publishStream.bufferTime=0;
    publishStream.publish(publishStreamName.text);
}

public function addH264():void{
    var videoStreamSettings:H264VideoStreamSettings = new
H264VideoStreamSettings();
    videoStreamSettings.setProfileLevel(H264Profile.MAIN,H264Level.LEVEL_3_1);
    publishStream.videoStreamSettings = videoStreamSettings;
}
```

5. Воспроизведение потока

Воспроизведение потока начинается с вызова метода `play()`

line 223

```
private function play():void{
    if (playStreamName.text == "") {
        playStatus.text = "Empty stream name";
        playStatus.setStyle("color","#ff0000");
        return;
    }
    playStatus.setStyle("color","#000000");
    Logger.info("play");
```

```
    subscribeStream = new NetStream(nc);
    addListenerAndPlay();
}
```

Все настройки и размеры буферов проставляются непосредственно перед воспроизведением в методе `addListenerAndPlay()`

line 244

```
private function addListenerAndPlay():void{
    subscribeStream.videoReliable=true;
    subscribeStream.audioReliable=false;
    subscribeStream.useHardwareDecoder=true;
    subscribeStream.addEventListener(NetStatusEvent.NET_STATUS,
handleSubscribeStreamStatus);
    subscribeStream.bufferTime=0;
    var soundTransform:SoundTransform = new SoundTransform();
    soundTransform.volume=0.7;
    subscribeStream.soundTransform = soundTransform;
    subscribeStreamObject = createStreamObject();
    subscribeStream.play(playStreamName.text);
    videoFarEnd.attachNetStream(subscribeStream);
    videoFarEnd.width = 320;
    videoFarEnd.height = 240;
    videoFarEnd.visible = true;
}
```