原 最简单的基于Flash的流媒体示例：RTMP推送和接收（ActionScript）

2015年02月25日 12:40:49

===

Flash流媒体文章列表：

最简单的基于Flash的流媒体示例：RTMP推送和接收（ActionScript）

最简单的基于Flash的流媒体示例：网页播放器（HTTP，RTMP，HLS）

===

本文记录一些基于Flash的流媒体处理的例子。Flash平台最常见的流媒体协议是RTMP。此前记录的一些基于C/C++的RTMP播放器/推流器，但是没有记录过基于Flash中的ActionScript的RTMP播放器/推流器。其实基于Flash的RTMP播放器/推流器才能算得上是RTMP技术中的“正规军”。RTMP本身设计出来就是用于Flash平台之间通信的，而且RTMP最大的优势——“无插件直播”，也是得益于广泛安装在客户端的Flash Player。因此本文分别记录一个基于ActionScript的RTMP播放器和基于ActionScript的RTMP推流器。

基于C/C++的RTMP流媒体处理的例子可以参考下面几个。

发布

最简单的基于librtmp的示例：发布H.264（H.264通过RTMP发布）

最简单的基于librtmp的示例：发布（FLV通过RTMP发布）

最简单的基于FFmpeg的推流器（以推送RTMP为例）

接收

最简单的基于librtmp的示例：接收（RTMP保存为FLV）

最简单的基于FFMPEG+SDL的视频播放器 ver2 （采用SDL2.0）

简介

相比于使用C/C++处理RTMP而言，使用ActionScript处理RTMP非常的简单。RTMP建立连接的方法都已经封装好了，只需要调用现成的接口函数就可以了。但是使用ActionScript处理RTMP的劣势也十分明显——可供自己开发的地方很少。由于Flash本身不开源，所以我们无法得到它的底层代码，因而也不能对编解码底层的参数进行调整。总而言之，ActionScript处理RTMP可以概括为几个字：“简单但是不灵活”。

ActionScript播放RTMP

ActionScript播放RTMP流媒体的流程如下图所示。

从图中可以看出，流程可以分成两部分：播放和显示。

播放

播放分成3步：

（1）建立NetConnection
（2）建立NetStream
（3）调用NetStream的play()方法

前2步分别建立了RTMP规范中的两个逻辑结构：NetConnection和NetStream。NetConnection代表服务器端应用程序和客户端之间基础的连通关系。NetStream代表了发送多媒体数据的通道。服务器和客户端之间只能建立一个NetConnection，但是基于该连接可以创建很多NetStream。这两个结构的结构如下图所示。

显示

显示部分将播放的视频显示在“舞台”上。这一部分通过创建一个Video对象实现。

ActionScript推送RTMP

ActionScript推送RTMP流媒体的流程如下图所示。

从图中可以看出，推送RTMP的流程和播放有些类似，最主要的不同在于推送最后调用的是NetStream的publish()方法，而播放最后调用的是NetStream的play()方法。
推流分成4步：
（1）建立NetConnection
（2）建立NetStream
（3）绑定摄像头和麦克风
（4）调用NetStream的play()方法
推流程序开始运行后，可以通过ffplay，VLC或者Flash应用程序访问相应的RTMP URL查看流媒体。

代码

本文附件中包含以下2个ActionScript工程：

simplest as3 rtmp player，最简单的RTMP播放器，其中包含3个独立的子工程：

simplest_as3_rtmp_player：最简单的RTMP播放器。

simplest_as3_local_player：最简单的本地文件播放器。

simplest_as3_rtmp_player_multiscreen：最简单的RTMP多屏播放器。

simplest_as3_rtmp_streamer，最简单的RTMP推流器

下面看一下上述几个工程的源代码。

simplest_as3_rtmp_player

simplest_as3_rtmp_player是最简单的RTMP播放器，代码如下所示。

[plain]

```
/**
 * 最简单的基于ActionScript的RTMP播放器
 * Simplest AS3 RTMP Player
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序使用ActionScript3语言完成，播放RTMP服务器上的流媒体
 * 是最简单的基于ActionScript3的播放器。
 *
 * This software is written in Actionscript3, it plays stream
 * on RTMP server
 * It's the simplest RTMP player based on ActionScript3.
 *
 */
package {
    import flash.display.Sprite;
    import flash.net.NetConnection;
    import flash.events.NetStatusEvent;
    import flash.events.AsyncErrorEvent;
    import flash.net.NetStream;
    import flash.media.Video;


    public class simplest_as3_rtmp_player extends Sprite
    {
        var nc:NetConnection;
        var ns:NetStream;
        var video:Video;


        public function simplest_as3_rtmp_player()
        {
            nc = new NetConnection();
            nc.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
            nc.connect("rtmp://localhost/live");
        }






        private function netStatusHandler(event:NetStatusEvent):void
        {
            trace("event.info.level: " + event.info.level + "\n", "event.info.code: " + event.info.code);
```

```
            switch (event.info.code)
            {
                case "NetConnection.Connect.Success":
                    doVideo(nc);
                    break;
                case "NetConnection.Connect.Failed":
                    break;
                case "NetConnection.Connect.Rejected":
                    break;
                case "NetStream.Play.Stop":
                    break;
                case "NetStream.Play.StreamNotFound":
                    break;
            }
        }



        // play a recorded stream on the server
        private function doVideo(nc:NetConnection):void {
            ns = new NetStream(nc);
            ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);


            video = new Video(640,480);
            video.attachNetStream(ns);


            ns.play("myCamera");
            addChild(video);
        }


        // create a playlist on the server
        /*
        private function doPlaylist(nc:NetConnection):void {
            ns = new NetStream(nc);
            ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);


            video = new Video();
            video.attachNetStream(ns);


            // Play the first 3 seconds of the video
            ns.play( "bikes", 0, 3, true );
            // Play from 20 seconds on
            ns.play( "bikes", 20, -1, false);
            // End on frame 5
            ns.play( "bikes", 5, 0, false );
            addChild(video);
        }
        */
    }
}
```

simplest_as3_local_player

simplest_as3_local_player用于播放本地FLV文件。ActionScript中播放本地视频（*.flv）和播放RTMP流程是一样的：先创建NetConnection，再创建NetStream。它们最大的不同在于，播放本地文件建立NetConnection的时候，是不传地址的。例如播放RTMP的时候代码如下：

```
nc.connect("rtmp://localhost/live");
```

播放本地文件的时候代码如下：

```
nc.connect(null);
```

调用play()的时候，RTMP传递服务器上的路径，如下所示。

```
ns.play("myCamera");
```

本地文件直接传递本地路径，如下所示。

```
ns.play("sintel.flv");
```

simplest_as3_rtmp_player_multiscreen

simplest_as3_rtmp_player_multiscreen是一个多屏播放的简单例子。实现了2x2网格播放4路视频。不再过多记录。

simplest_as3_rtmp_streamer

simplest_as3_rtmp_player是最简单的RTMP推流器，代码如下所示。

[plain]

```
/**
 * 最简单的基于ActionScript的RTMP推流器
 * Simplest AS3 RTMP Streamer
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序使用ActionScript3语言完成，推送本地摄像头的数据至RTMP流媒体服务器，
 * 是最简单的基于ActionScript3的推流器。
 *
 * This software is written in Actionscript3, it streams camera's video to
 * RTMP server.
 * It's the simplest RTMP streamer based on ActionScript3.
 *
 */




package {
    import flash.display.MovieClip;
    import flash.net.NetConnection;
    import flash.events.NetStatusEvent;
    import flash.net.NetStream;
    import flash.media.Video;
    import flash.media.Camera;
    import flash.media.Microphone;
    //import flash.media.H264Profile;
    //import flash.media.H264VideoStreamSettings;


    public class simplest_as3_rtmp_streamer extends MovieClip
    {
        var nc:NetConnection;
        var ns:NetStream;
        var nsPlayer:NetStream;
        var vid:Video;
        var vidPlayer:Video;
        var cam:Camera;
        var mic:Microphone;

        var screen_w:int=320;
        var screen_h:int=240;


        public function simplest_as3_rtmp_streamer()
        {
            nc = new NetConnection();
            nc.addEventListener(NetStatusEvent.NET_STATUS, onNetStatus);
            nc.connect("rtmp://localhost/live");
        }


        private function onNetStatus(event:NetStatusEvent):void{
            trace(event.info.code);
            if(event.info.code == "NetConnection.Connect.Success"){
                publishCamera();
                displayPublishingVideo();
                displayPlaybackVideo();
            }
        }


        private function publishCamera() {

            //Cam

            cam = Camera.getCamera();


            /**
             * public function setMode(width:int, height:int, fps:Number, favorArea:Boolean = true):void
             *  width:int — The requested capture width, in pixels. The default value is 160.
             *  height:int — The requested capture height, in pixels. The default value is 120.
             *  fps:Number — The requested capture frame rate, in frames per second. The default value is 15.
             */
```

```
            cam.setMode(640, 480, 15);

            /**
             * public function setKeyFrameInterval(keyFrameInterval:int):void
             * The number of video frames transmitted in full (called keyframes) instead of being interpolated by the video compressi
algorithm.
             * The default value is 15, which means that every 15th frame is a keyframe. A value of 1 means that every frame is a key
me.
             * The allowed values are 1 through 300.
             */
            cam.setKeyFrameInterval(25);

            /**
             * public function setQuality(bandwidth:int, quality:int):void
             * bandwidth:int — Specifies the maximum amount of bandwidth that the current outgoing video feed can use, in bytes per s
nd (bps).
             *    To specify that the video can use as much bandwidth as needed to maintain the value of quality, pass 0 for bandwidt

             *    The default value is 16384.
             * quality:int — An integer that specifies the required level of picture quality, as determined by the amount of compress

             *     being applied to each video frame. Acceptable values range from 1 (lowest quality, maximum compression) to 100
             *    (highest quality, no compression). To specify that picture quality can vary as needed to avoid exceeding bandwidth,

             *    pass 0 for quality.
             */
            cam.setQuality(200000, 90);

            /**
             * public function setProfileLevel(profile:String, level:String):void
             * Set profile and level for video encoding.
             * Possible values for profile are H264Profile.BASELINE and H264Profile.MAIN. Default value is H264Profile.BASELINE.
             * Other values are ignored and results in an error.
             * Supported levels are 1, 1b, 1.1, 1.2, 1.3, 2, 2.1, 2.2, 3, 3.1, 3.2, 4, 4.1, 4.2, 5, and 5.1.
             * Level may be increased if required by resolution and frame rate.
             */
            //var h264setting:H264VideoStreamSettings = new H264VideoStreamSettings();
            // h264setting.setProfileLevel(H264Profile.MAIN, 4);


            //Mic

            mic = Microphone.getMicrophone();

            /*
             * The encoded speech quality when using the Speex codec. Possible values are from 0 to 10. The default value is 6.
             * Higher numbers represent higher quality but require more bandwidth, as shown in the following table.
             * The bit rate values that are listed represent net bit rates and do not include packetization overhead.
             * ------------------------------------------
             * Quality value | Required bit rate (kbps)
             *-------------------------------------------
             *      0        |       3.95
             *      1        |       5.75
             *      2        |       7.75
             *      3        |       9.80
             *      4        |       12.8
             *      5        |       16.8
             *      6        |       20.6
             *      7        |       23.8
             *      8        |       27.8
             *      9        |       34.2
             *      10       |       42.2
             *-------------------------------------------
             */
            mic.encodeQuality = 9;

            /* The rate at which the microphone is capturing sound, in kHz. Acceptable values are 5, 8, 11, 22, and 44. The default v
e is 8 kHz
             * if your sound capture device supports this value. Otherwise, the default value is the next available capture level abo
8 kHz that
             * your sound capture device supports, usually 11 kHz.
             *
             */
            mic.rate = 44;


            ns = new NetStream(nc);
            //H.264 Setting
            //ns.videoStreamSettings = h264setting;
            ns.attachCamera(cam);
            ns.attachAudio(mic);
            ns.publish("myCamera", "live");
        }


        private function displayPublishingVideo():void {
            vid = new Video(screen_w, screen_h);
            vid.x = 10;
            vid.y = 10;
            vid.attachCamera(cam);
            addChild(vid);
        }
```

```
        }


        private function displayPlaybackVideo():void{
            nsPlayer = new NetStream(nc);
            nsPlayer.play("myCamera");
            vidPlayer = new Video(screen_w, screen_h);
            vidPlayer.x = screen_w + 20;
            vidPlayer.y = 10;
            vidPlayer.attachNetStream(nsPlayer);
            addChild(vidPlayer);
        }
    }
}
```

结果

simplest as3 rtmp player运行后会自动连接RTMP URL：rtmp://localhost/live/myCamera。
程序运行后的结果如下图所示。

simplest_as3_local_player运行会播放sintel.flv文件。
运行结果如下图所示。

simplest_as3_rtmp_player_multiscreen运行后会连接4个RTMP URL。
运行结果如下图所示。

simplest_as3_rtmp_streamer运行结果后会推送本机的摄像头的视频和麦克风的音频到指定的RTMP URL（在这里是rtmp://localhost/live/myCamera）。左侧的视频是从摄像头读取的视频，右侧的视频是推流后从RTMP URL读取的视频（一般会有一定延时）。
运行结果如下图所示。

下载

Simplest flashmedia example

SourceForge： https://sourceforge.net/projects/simplestflashmediaexample/

Github： https://github.com/leixiaohua1020/simplest_flashmedia_example

开源中国： http://git.oschina.net/leixiaohua1020/simplest_flashmedia_example

CSDN下载： http://download.csdn.net/detail/leixiaohua1020/8456441

本工程包含如下基于Flash技术的流媒体的例子：
simplest_as3_rtmp_player: 最简单的RTMP播放器（基于ActionScript）
simplest_as3_rtmp_streamer: 最简单的RTMP推流器（基于ActionScript）
rtmp_sample_player_adobe: 从Adobe Flash Media Sever提取出来的测试播放器
rtmp_sample_player_wowza: 从Wowza服务器中提取出来的测试播放器
rtmp_sample_player_flowplayer: 基于FlowPlayer的RTMP/HTTP播放器（添加RTMP plugin）
rtmp_sample_player_videojs: 基于VideoJS的RTMP/HTTP播放器
rtmp_sample_player_jwplayer: 基于JWplayer的RTMP/HTTP播放器
hls_sample_player_flowplayer: 基于FlowPlayer的HLS播放器（添加HLS plugin）
hls_video_player_html5: 基于HTML5的HLS/HTTP播放器
activex_vlc_player: 基于VLC的ActiveX控件的播放器

注意：某些播放器直接打开html页面是不能工作的，需要把播放器放到Web服务器上。
（例如Apache或者Nginx）

文章标签： flash actionscript 播放 推流 RTMP

个人分类： Flash相关 我的开源项目