Rtmp数据流转h264的详细转码过程

一、 目的：

这段时间，因为工作上的需要，在RTMP上做了flv流到标准h264、AAC的转换。

二、 RTMP介绍：

RTMP(Real Time Messaging Protocol)实时消息传送协议是Adobe Systems公司为Flash播放器和服务器之间音频、视频和数据传输开发的私有协议。

RTMP协议就像一个用来装数据包的容器,这些数据可以是AMF格式的数据,也可以是FLV中的视/音频数据.

它有三种变种：

1)工作在TCP之上的明文协议，使用端口1935；

2)RTMPT封装在HTTP请求之中，可穿越防火墙；

3)RTMPS类似RTMPT，但使用的是HTTPS连接；

Rtmp详细介绍请参照网上文档（一搜一大把）

三、 Rtmp流转h264

基于rtmp client端收取的数据包packet进行解码。

1、 在rtmp传输数据流的时候，不论是在点播情况下或者直播情况下，rtmpserver会在流开始的时候添加File Header和 MedadataTag。这些不需要分析。

2、 数据头解析

2-1、Tagtype，一字节

每个packet的第一个字节代表了当前packet的类型。0x04表示Ping包，0x08为audio，0x09为video。0x12为script data。0x16比较特殊，这个是为了实现H.264数据的直播而增加了一个数据类型。这个之后特别讨论。

2-2、Datasize，24bit

这三个字节表述了tag中数据段的大小。

2-3、Timestamp，24bit

记录了每一个tag相对于第一个tag（File Header）的相对时间。以毫秒（milliseconds）为单位。而File Header的timestamp永远为0。

2-4、TimestampExtended，8bit

扩展时间字段，此字段与timestamp字段共同组成完整的时间戳字段，只不过Timestamp是这个字段的低24位，TimestampExtended为这个字段的高8位。

2-5、StreamID，24bit

永远都是0。 Tag body。存储音视频信息等。 如果， Tagtype==0x08，Data为audiodata。 Tagtype==0x09，Data为videodata。

Tagtype==0x12，Data为scriptdataobject。

Tagtype==0x16，Data为H264直播流数据包。

2-6、Data，大小是Datasize

3、 对rtmp音频数据的解析（audio tag）

不难看出rtmp中flv音频流就是一个接着一个的Audio tag。每次传输流的第一个audio tag标示了音频使用的adts header信息，即AAC header（audiodata），占据前3个字节。之后根据标志位的不同，分AAC sequence header和AAC raw信息。具体解析如下：

3-1、AAC header结构（2个字节）

3-1-1、SoundFormat，4bit

0 = Linear PCM, platform endian

1 = ADPCM

2 = MP3

3 = Linear PCM, little endian

4 = Nellymoser 16 kHz mono

5 = Nellymoser 8 kHz mono

6 = Nellymoser

7 = G.711 A-law logarithmic PCM

8 = G.711 mu-law logarithmic PCM

9 = reserved

10 = AAC

11 = Speex

14 = MP3 8 kHz

15 = Device-specific sound

3-1-2、SoundRate，2bit，抽样频率

0 = 5.5 kHz

1 = 11 kHz

2 = 22 kHz

3 = 44 kHz

对于aac音频来说，总是0x11，即44khz.

3-1-3、SoundSize，1bit，音频的位数。

0 = 8-bit samples

1 = 16-bit samples

AAC总是为0x01,16位。

3-1-4、SoundType，1bit，声道

0 = Mono sound

1 = Stereo sound

3-1-5、AACPacketType，8bit。

这个字段来表示AACAUDIODATA的类型：0 = AAC sequence header，1 = AAC raw。第一个音频包用0，后面的都用1

3-2、AAC sequence header，2字节：

3-2-1、audioObjectType,5bit。结构编码类型

0=AAC main

1=AAC lc

2=AAC ssr

3=AAC LTP

一般AAC使用2，即AAC lc

3-2-2、SamplingFrequencyIndex，4bit，音频采样率索引值

0: 96000 Hz

1: 88200 Hz

2: 64000 Hz

3: 48000 Hz

4: 44100 Hz

5: 32000 Hz

6: 24000 Hz

7: 22050 Hz

8: 16000 Hz

9: 12000 Hz

10: 11025 Hz

11: 8000 Hz

12: 7350 Hz

13: Reserved

14: Reserved

15: frequency is written explicitly

通常aac固定选中44100，即应该对应为4，但是试验结果表明，当音频采样率小于等于44100时，应该选择3，而当音频采样率为48000时，应该选择2.但是也有例外。

3-2-3、ChannelConfiguration，4bit，音频输出声道。

对应的是音频的频道数目。单声道对应1，双声道对应2，依次类推。

0: Defined in AOT Specifc Config

1: 1 channel: front-center

2: 2 channels: front-left, front-right

3: 3 channels: front-center, front-left, front-right

4: 4 channels: front-center, front-left, front-right, back-center

5: 5 channels: front-center, front-left, front-right, back-left, back-right

6: 6 channels: front-center, front-left, front-right, back-left, back-right, LFE-channel 7: 8 channels: front-center, front-left, front-right, side-left, side-right, back-left, back-right, LFE-channel

8-15: Reserved

3-2-4、frameLengthConfig，1bit，标志位，用于表明IMDCT窗口长度

始终为0。

3-2-5、dependsOnCoreCoder，1bit，标志位，表明是否依赖于corecoder

始终为0。

3-2-7、extensionFlag，1bit

如果是AAC-LC,这里必须为0.

3-3、AAC raw ,大小不定

3-3-7、AAC payload

大小由packet的大小减去头的大小，再减去AAC header的2个字节的大小。一般形式是，0xAF 0x01 +PAY LOAD.

4、 对rtmp视频数据的解析（video tag）

如果packet头中的TagType==9时，就表示这个TAG是video。那么StreamID之后的数据就表示是VideoTagHeader。VideoTagHeader只有一个字节，也就是接跟着StreamID的1个字节包含着视频帧类型及视频CodecID最基本信息VideoTagHeader结构如下：

4-1、VideoTagHeader(8bit)结构：

4-1-1、FrameType，4bit，帧类型

1 = key frame (for AVC, a seekable frame)

2 = inter frame (for AVC, a non-seekable frame)

3 = disposable inter frame (H.263 only)

4 = generated key frame (reserved for server use only)

5 = video info/command frame

H264的一般为1或者2.

4-1-2、CodecID ，4bit，编码类型

1 = JPEG(currently unused)

2 = Sorenson H.263

3 = Screen video

4 = On2 VP6

5 = On2 VP6 with alpha channel

6 = Screen video version 2

7 = AVC

4-3、VideoData (根据CodecID 判断之后videodata的类型)

If CodecID = 2, H263videopacket;

If CodecID = 3, SCREENvideopacket;

If CodecID = 4, VP6FLVvideopacket;

If CodecID = 5, VP6FLVALPHAvideopacket;

If CodecID = 6, SCREENV2videopacket;

If CodecID = 7, AVCvideopacket;

一般用AVCvideopacket。

VideoTagHeader之后跟着的就是VIDEODATA数据了，也就是video payload.当然就像音频AAC一样，这里也有特例就是如果视频的格式是AVC（H.264）的话，VideoTagHeader会多出4个字节的信息。AVCPacketType 和 CompositionTime。AVCPacketType表示接下来 VIDEODATA（AVCVIDEOPACKET）的内容：

IF AVCPacketType == 0 AVCDecoderConfigurationRecord（AVC sequence header） IF AVCPacketType == 1 One or more NALUs (Full frames are required)

AVCDecoderConfigurationRecord.包含着是H.264解码相关比较重要的sps和pps信息，再给AVC解码器送数据流之前一定要把sps和pps信息送出，否则的话解码器不能正常解码。而且在解码器stop之后再次start之前，如seek、快进快退状态切换等，都需要重新送一遍sps和pps的信息.AVCDecoderConfigurationRecord在FLV文件中一般情况也是出现1次，也就是第一个 video tag.

4-4、VIDEODATA(CodecID = 7)

4-4-1、AVCPacketType,8bit

IF AVCPacketType == 0 AVCDecoderConfigurationRecord（AVC sequence

header）(此时FrameType必为1)

IF AVCPacketType == 1 One or more NALUs (Full frames are required)

IF AVCPacketType == 2 AVC end of sequence (lower level NALU sequence

ender is not required or supported)

4-4-2、CompositionTime，24bit

IF AVCPacketType == 1

Composition time offset,（没解析。）

ELSE

4-4-3、AVCDecoderConfigurationRecord（AVCPacketType == 0，FrameType==1） 4-4-3-1、configurationVersion，8bit

4-4-3-2、AVCProfileIndication，8bit

4-4-3-3、profile\_compatibility，8bit

4-4-3-4、AVCLevelIndication，8bit

4-4-3-5、lengthSizeMinusOne，8bit

H.264 视频中 NALU 的长度，计算方法是 1 + (lengthSizeMinusOne

3),实际测试时发现总为ff，计算结果为4.

4-4-3-6、numOfSequenceParameterSets，8bit

SPS 的个数，计算方法是 numOfSequenceParameterSets 0x1F,实际测

试时发现总为E1，计算结果为1

4-4-3-7、sequenceParameterSetLength，16bit

SPS 的长度

4-4-3-8、sequenceParameterSetNALUnits ，sps。

长度为sequenceParameterSetLength。

4-4-3-9、numOfPictureParameterSets，8bit

PPS 的个数,计算方法是 numOfPictureParameterSets 0x1F,实际测试时发现总为E1，计算结果为1。

4-4-3-10、pictureParameterSetLength，16bit。

PPS的长度。

4-4-3-11、PPS

长度为pictureParameterSetLength。

4-4-4、NALUs（AVCPacketType == 1，FrameType==1或者2）

一个packet里面可能包含多个NALUs，每个NALUs前的4个字节都是标示这个NALU长度的4个字节，然后是这个NALU包。一个NALU包结束之后，接着时下一个NALU包的长度4字节，然后是下一个NALU包。以此类推，直到这个packet结束。

4-4-4-1、nal\_length，32bit。每个nal包长度

每个NALU包前面都有（lengthSizeMinusOne 3）+1个字节的NAL包长度描述（前文提到的，还记得吗），前面计算结果为4个字节。 4-4-4-2、nal包

这里插入一点NALU的小知识，每个NALU第一个字节的前5位标明的是该NAL包的类型，即NAL nal\_unit\_type

#define NALU\_TYPE\_SLICE 1

#define NALU\_TYPE\_DPA 2

#define NALU\_TYPE\_DPB 3

#define NALU\_TYPE\_DPC 4

#define NALU\_TYPE\_IDR 5 //I帧

#define NALU\_TYPE\_SEI 6

#define NALU\_TYPE\_SPS 7

#define NALU\_TYPE\_PPS 8

#define NALU\_TYPE\_AUD 9 //访问分隔符

#define NALU\_TYPE\_EOSEQ 10

#define NALU\_TYPE\_EOSTREAM 11

#define NALU\_TYPE\_FILL 12

四、 参考文档：

1、

2、

3、

4、

本文档下载自文档之家，www.doczj.com-免费文档分享平台，众多试卷、习题答案、公务员考试、英语学习、法语学习、人力资源管理、电脑基础知识、学习计划、工作计划、工作总结、活动策划、企业管理等文档分类免费下载；乐于分享，共同进步，转载请保留出处:http://www.doczj.com/doc/b57c1b1aa21614791711284f.html