H264 RTP包解析

1. 预备

视频：

由一副副连续的图像构成，由于数据量比较大，因此为了节省带宽以及存储，就需要进行必要的压缩与解压缩，也就是编解码。

h264裸码流：

对一个图像或者一个视频序列进行压缩，即产生码流，采用H264编码后形成的码流就是h264裸码流。

码流传输：

发送端将H264裸码流打包后进行网络传输，接收端接收后进行组包还原裸码流，然后可以再进行存储，转发，或者播放等等相关的处理。

存储转发可以直接使用裸码流，播放则需要进行解码和显示处理

解码显示：

一般会解成YUV数据，然后交给显示相关模块做处理，如使用openGL或者D3D等进行渲染（采用3d的方式来显示2d的图像称为渲染）

解码又分软件解码和硬件解码，

软解一般ffmpeg

硬解则各不同，由各硬件厂商开放sdk来处理，如：hisi，intel media sdk，nvida gpu，apple ios videotoolbox等。

2. h264码流传输

类似发送网页文本数据有http一样，视频数据在网络上传输也有专门的网络协议来支持，如：rtsp，rtmp等

由于码流是一帧一帧的图片数据，所以传输的时候也是一帧帧来传输的，因此这里就会涉及到各种类型的帧处理了

I 帧：参考帧或者关键帧，可以理解为是一帧完整画面，解码时只需要本帧数据就解码成一幅完整的图片，数据量比较大。

P帧：差别帧，只有与前面的帧或I帧的差别数据，需要依赖前面已编码图象作为参考图象，才能解码成一幅完整的图片，数据量小。

B帧：双向差别帧，也就是B帧记录的是本帧与前后帧的差别，需要依赖前后帧和I帧才能解码出一幅完整的图片，数据量小。

由于I帧比较大，已经超出mtu最大1500，所以需要拆包分片传输，这里说的拆包发送不是指发送超过1500的数据包时tcp的分段传输或者upd的ip分片传输，而是指rtp协议本身对264的拆包。

rtp打包后将数据交给tcp或者upd进行传输的时候就已经控制在1500以内，这样可以提高传输效率，避免一个I帧万一丢失就会造成花屏或者重传造成超时卡顿等等问题。

（顺便提一句，rtmp打包就比较简单，由于是基于tcp的协议，大包直接交给tcp去做分段传输，rtmp通过设置合适的trunk size去发送一帧帧数据）

既然要进行拆包发送与接收，就少不了需要相关的包结构以及打包组包了，继续。。。。。

3. H264在网络传输的单元：NALU

NALU结构：**NALU header**(1byte) + **NALU payload**

header 部分可以判断类型等信息，从右往左5个bit位

SPS：0x67　header & 0x1F = 7	I Frame: 0x65　header & 0x1F = 5
PPS：0x68　header & 0x1F = 8	P Frame: 0x41　header & 0x1F = 1
SEI：0x66　header & 0x1F = 6	

payload 部分可以简单理解为 编码后的264帧数据

详细可以去查阅 **h264 NALU 语法结构**

4. h264的RTP打包

1. 单NALU：　P帧或者B帧比较小的包，直接将NALU打包成RTP包进行传输　**RTP header**(12bytes) + **NALU header** (1byte) + **NALU payload**
2. 多NALU：　特别小的包几个NALU放在一个RTP包中
3. FUs(Fragment Units)：I帧长度超过MTU的，就必须要拆包组成RTP包了,有FU-A，FU-B

RTP header (12bytes)+ **FU Indicator** (1byte) + **FU header**(1 byte) + NALU payload

看到这里会不禁思考，

1. **NALU头不见了，如何判断类型？实际上NALU头被分散填充到FU indicator和FU header里面了**

bit位按照从左到右编号0-7来算，nalu头中0-2前三个bit放在FU indicator的0-2前三个bit中，后3-7五个bit放入FU header的后3-7五个中

```
//NALU header = (FU indicator & 0xe0) | (FU header & 0x1f)   取FU indicator前三和FU header后五
headerStart[1] = (headerStart[0]&0xE0)|(headerStart[1]&0x1F);
```

因此查看I帧p帧类型，遇到FU分片的，**直接看第二个字节，**即Fu header后五位，这个跟直接看NALU头并无差异，一般有：0x85，0x05，0x45等等

2. **多个RTP包如何还原组合成回一个完整的I帧？在FU header中有标记为判断**

照旧从左到右，**Fu header**前两个bit表示start和end标记，start为1表示一个I帧分片开始，end为1表示一个I帧分片结束

3. 如何查看是一个I帧分片开始？

看第一个字节**FU Indicator**，照旧从左到右，Fu Indicator前三个bit是NALU头的前三个bit，后五位为类型FU-A：28（11100），FU-B：29（11101）

RTP抓包看下来整个字节是0x7c开头

如果不是分片，第一字节就是NALU头，如：0x67,0x68,0x41等

5. 抓包分析如下图：

--->**RTP包中接收的264包是不含有0x00,0x00,0x00,0x01头的，这部分是rtp接收以后，另外再加上去的，解码的时候再做判断的。**

1. SPS

2. I帧分片开始，第一个字节**FU Indicator**，0x7c, 后五位11100，28，FU-A

第二个字节FU Header，0x85，**前两个bit start位1，end位0 表示 分片开始**，后五个bit值5，I帧

I帧分片，0x7c开头，**第二个字节0x05, FU Header start和end位 0**，后五个bit值5，I帧

I帧分片结束，7c开头，**第二个字节0x45,FU Headr，start 0,end1**，后五个bit值5，I帧， Mark标记一帧结束

3. p帧，第一个字节：0x41

4. P帧分片开始：第一个字节**FU Indicator**，0x7c, 后五位11100，28，FU-A

第二个字节FU Header，0x81, **前两个bit start位1，end位0 表示 分片开始**，后五个bit 值是1，p帧

P帧分片结束：0x5c开头，**第二个字节0x41,FU Headr，start 0,end1**，后五个bit值1，P帧， Mark标记一帧结束

6. 参考live555相关的源码如下：

```
Boolean H264VideoRTPSource
::processSpecialHeader(BufferedPacket* packet,
                       unsigned& resultSpecialHeaderSize) {
  unsigned char* headerStart = packet->data();
  unsigned packetSize = packet->dataSize();
  unsigned numBytesToSkip;

  // Check the 'nal_unit_type' for special 'aggregation' or 'fragmentation' packets:
  if (packetSize < 1) return False;
  fCurPacketNALUnitType = (headerStart[0]&0x1F);   //FU Indicator最五位即NALU类型 0x1F = 0001 1111
  switch (fCurPacketNALUnitType) {
  case 24: { // STAP-A
    numBytesToSkip = 1; // discard the type byte
    break;
  }
  case 25: case 26: case 27: { // STAP-B, MTAP16, or MTAP24
    numBytesToSkip = 3; // discard the type byte, and the initial DON
    break;
  }
  case 28: case 29: { // // FU-A or FU-B
    // For these NALUs, the first two bytes are the FU indicator and the FU header.
    // If the start bit is set, we reconstruct the original NAL header into byte 1:
    if (packetSize < 2) return False;
    unsigned char startBit = headerStart[1]&0x80;    //FU Header start标记位   0x80= 1000 0000
    unsigned char endBit = headerStart[1]&0x40;      //FU Header End标记位     0x40= 0100 0000
    if (startBit) {
      fCurrentPacketBeginsFrame = True;

      headerStart[1] = (headerStart[0]&0xE0)|(headerStart[1]&0x1F);  //还原NALU头
      numBytesToSkip = 1;
    } else {
      // The start bit is not set, so we skip both the FU indicator and header:
      fCurrentPacketBeginsFrame = False;
      numBytesToSkip = 2;
    }
    fCurrentPacketCompletesFrame = (endBit != 0);
    break;
  }
  default: {
    // This packet contains one complete NAL unit:
    fCurrentPacketBeginsFrame = fCurrentPacketCompletesFrame = True;   //默认没有分片，完整的NALU
    numBytesToSkip = 0;
    break;
  }
  }

  resultSpecialHeaderSize = numBytesToSkip;
  return True;
}
```

标签: h264 rtp

Leehm
粉丝 - 2 关注 - 1
+加关注

« 上一篇： c++ const 用法
» 下一篇： 使用git命令删除branch

posted on 2019-06-12 15:30 Leehm 阅读(8280) 评论(0) 编辑 收藏 举报