4 RTP之H264封包和解包

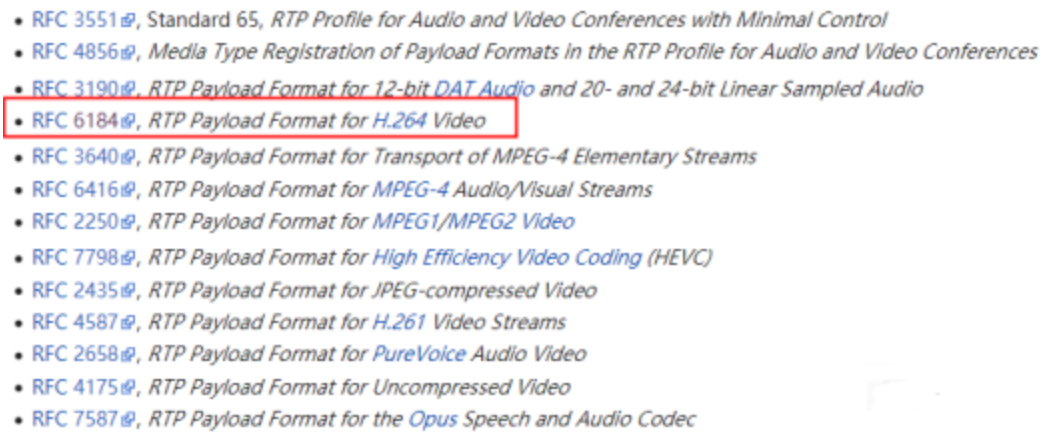
腾讯课堂 零声学院

FFmpeg/WebRTC/RTMP 音视频流媒体高级开发 https://ke.qq.com/course/468797?tuin=137bb271

1 H264打包RTP的方法

RTP的特点不仅仅支持承载在UDP上，这样利于低延迟音视频数据的传输，另外一个特点是它允许通过其它协议接收端和发送端协商音视频数据的封装和编解码格式，这样固定头的playload type字段就比较灵活。截止目前为止，RTP是我见过传输音视频数据类型最多的，具体参考：https://en.wikipedia.org/wiki/RTP_payload_formats。

今天我以H264裸码流NALU为例，给大家讲述下如何进行H264的打包，这也是我上面几篇封装格式讲解的固定套路，其中H264打包的详细方法要参考RFC6184文档。



H.264标准协议定义了两种不同的类型：一种是VCL即Video Coding Layer，一种是NAL即Network Abstraction Layer。其中前者就是编码器吐出来的原始编码数据，没有考虑传输和存储

问题。后面这种就是为了展现H.264的网络亲和性，**对VCL输出的slice片数据进行了封装为NALUs(NAL Units)**，然后再封装为RTP包进行传输，这些都是H.264的基础，见后续文章。

NALU的基本格式是：NALU Header + NALU Data,其中NALU的头由一个字节组成如下所示：

```
+---------------+
|0|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+-+
|F|NRI|   Type  |
+---------------+
```

F **(1 bit)**	如果是坏帧，则置1，其余H.264固定为0。
NRI **(2 bit)**	用来指示该NALU 的重要性等级。值越大，表示当前NALU越重要。具体大于0 时取何值，没有具体规定。 例如：如果是00，则表示此帧即使丢失了，也不影响解码；其他值则表示此帧如果丢失了，会影响解码，这个字段指明了该NALU的重要性，但是**实际我们不太关心这个字段**。
Nalu_Type **(5 bit)**	NAL Unit的类型，这个值指明了NALU的类型，其中NALU的类型见下表

Nalu_Type	NALU内容	备注
0	未指定	
1	非IDR图像编码的slice	比如普通I、P、B帧
2	编码slice数据划分A	2类型时，只传递片中最重要的信息，如片头，片中宏块的预测模式等；一般不会用到；
3	编码slice数据划分B	3类型是只传输残差；一般不会用到；
4	编码slice数据划分C	4时则只可以传输残差中的AC系数；一般不会用到；
5	IDR图像中的编码slice	IDR帧，IDR一定是I帧但是I帧不一定是IDR帧。
6	SEI补充增强信息单元	可以存一些私有数据等；
7	SPS 序列参数集	编码的参数配置
8	PPS 图像参数集	编码的参数配置

9	接入单元定界符	
10	序列结束	
11	码流结束	
12	填充数据	
24	STAP-A Single-time aggregation packet	单一时间聚合包模式，意味着一个RTP包可以传输多个NALU，但是这些NALU的编码时间要一样才能聚合到一个RTP。
25	STAP-B Single-time aggregation packet	单一时间聚合包模式，比STAP-B多一个DON
26	MTAP 16 Muti-time aggregation packet	多个时间聚合包模式：意味着一个RTP包可以传输多个NALU，但是这些NALU的编码时间有可能不一样。
27	MTAP 24 Muti-time aggregation packet	多个时间的聚合包模式
28	FU-A Fragmentation unit	分包模式：当一个RTP容纳不下一个NALU时，就需要FUs这种格式。
29	FU-B Fragmention unit	分包模式
30-31	未指定，保留	

参考文档：https://tools.ietf.org/html/rfc3984 过时的
https://tools.ietf.org/html/rfc6184 最新的

我们看到1-11就是NALU的单个包类型，但是一个NALU的大小是不一样的，如果是非视频数据的SPS PPS才十几个字节，对于IDR帧，则有可能几十KB。
这样把NALU打包到RTP方式就很多，分为：

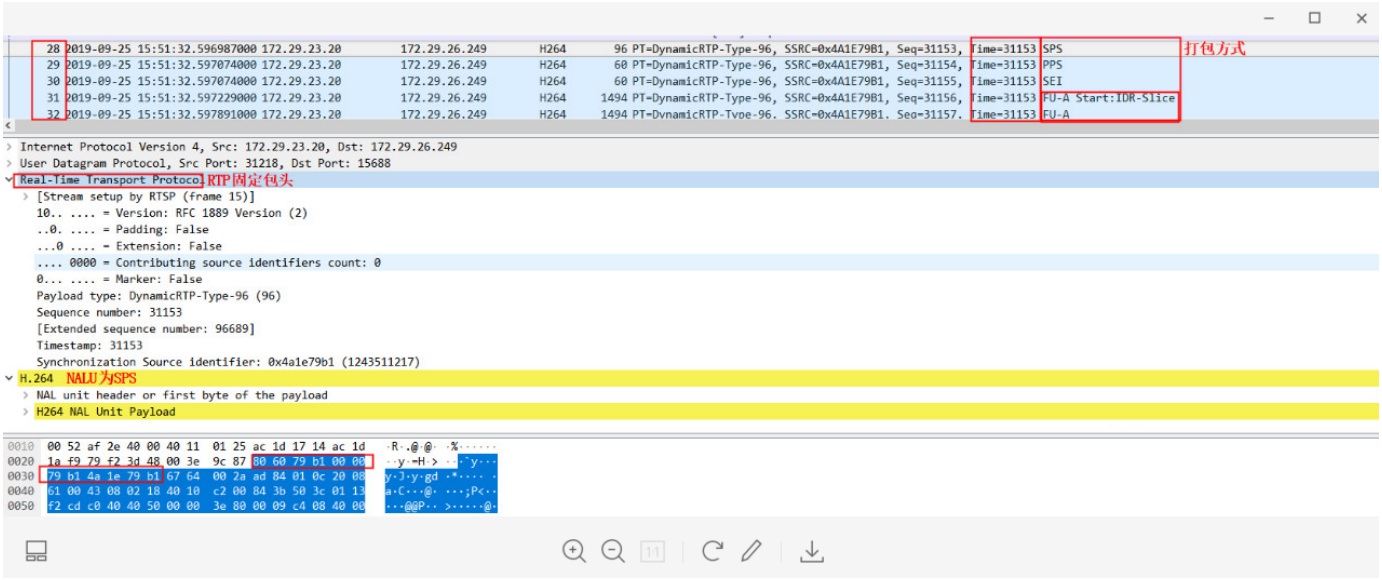
- 一个RTP包承载一个NALU；
- 多个NALU合并到一个RTP；
- 一个大的NALU切分成多个RTP。

同时由于时间戳的问题，就有了**24-29**几种类型。
但是对于发送端组RTP包的一方来说，尽可能找简单的打包方式。对于接受端则需要适配各种发送端的打包方式，因为无法决定输入源的打包方式。这里先分享下我们的打包方式，比较简单（**打包**

的时候不要搞太复杂的模式）：

1. 我们对于NALU的长度<=1400（rtp payload size）的则采用的是单一NALU打包到单一的RTP包中；
2. 我们对于NALU的长度>1400的则采用了FU-A的方式进行了打包，这种就是把一个大的NALU进行了切分，最后接收方则进行了合并，把多个RTP包合并成一个完整的NALU即可；
3. 为什么NALU的长度大于1400字节就要进行FU-A切片，是因为底层MTU大小值固定为1500，从传输效率讲，这里用1400作为切分条件。

同时我们发现现在视频监控领域摄像头通过RTP 传输码流的打包方式都是基本这种，这种打包方案简单容易实现，又满足需要。如下图所示：



① 28、29、30三个RTP分别传输的SPS、PPS、SEI这三种NALU，其中一个NALU分到一个RTP包，这是单一打包方式；

② 31、32三个RTP包分别传输了IDR帧的NALU单元，由于比较大，发送方采用了FU-A的打包方式；

③ 上图还显示了SPS、PPS、SEI的RTP包固定头，Seq初始值不为0，为随机值，并且一个RTP包就顺序+1，这跟上面分析的一致；

④ 上面SPS、PPS、SEI本身不涉及时间戳，但是这里头填充为和自己后面的第一个RTP保持一致，同样IDR的NALU切分的不同RTP包时间戳也是一样的；

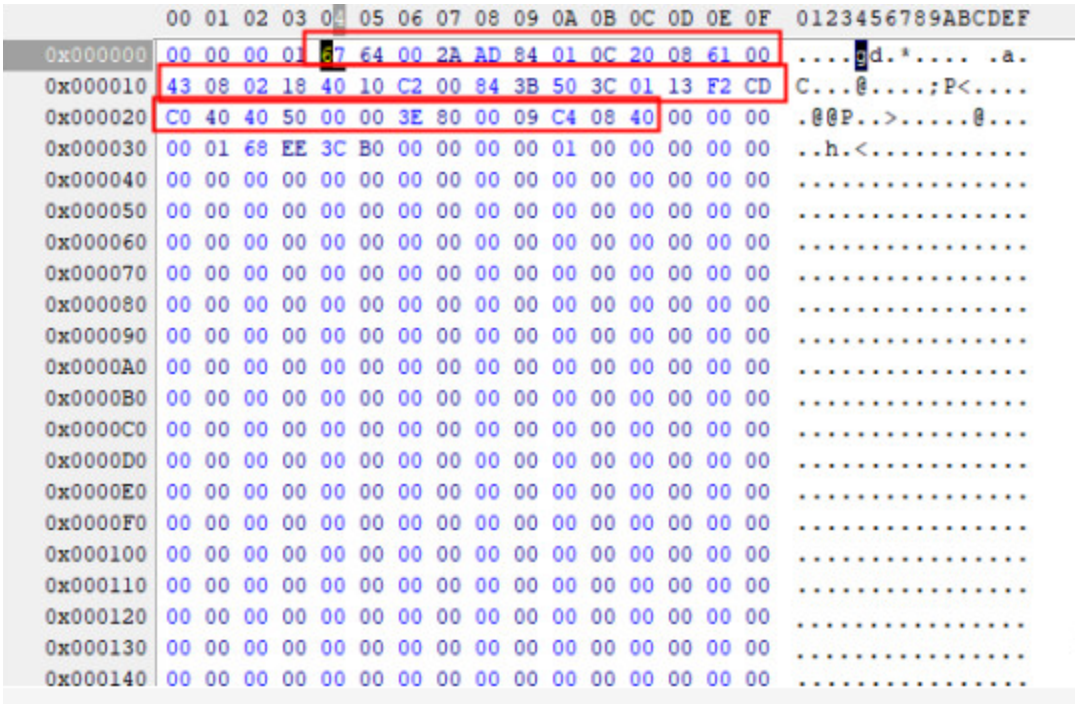
那么到底将单一的NALU打包到RTP或者把比较大的NLAU打包到多个RTP即FU-A方式是怎么操作的呢?

1.1 打包方式之Single NAL Unit

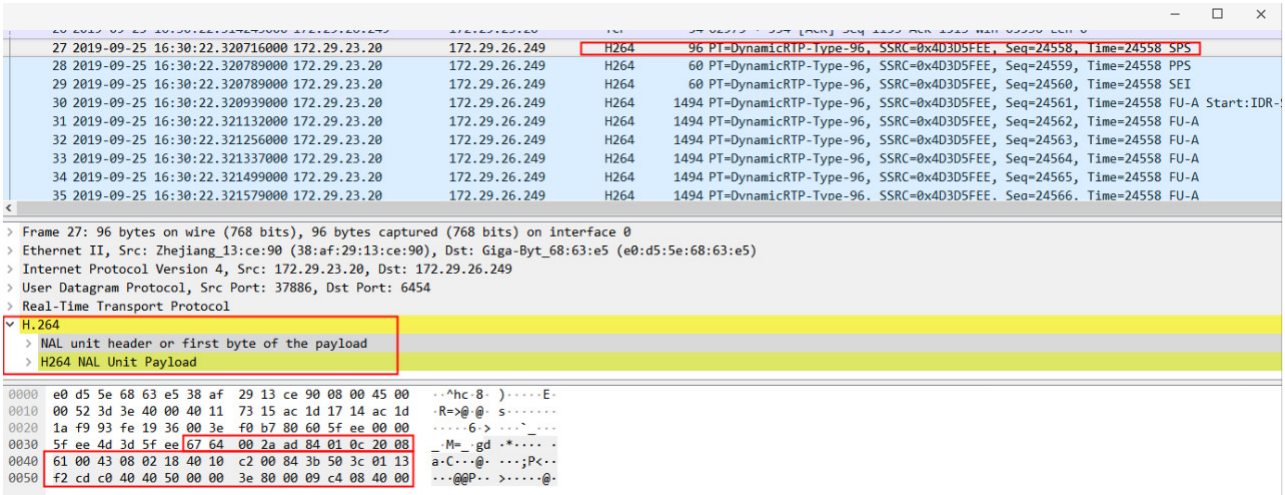
就是一个RTP包打包一个单独的NALU方式，其实最好理解，就是在RTP固定头后面**直接填充NLAU单元数据即可**，即：

RTP Header + NALU Header + NALU Data; （不包括startcode）

我了验证猜想我进行了抓包和写文件，我们发现写文件的SPS和抓包RTP包固定头后面的负载完全是一致的，写文件中的SPS:



抓包中的RTP固定头后面的SPS：



1.2 打包方式之FU-A

这种打包方式也不复杂，为了解释清楚，需要了解下面两个数据包头即FU indicator和Fu header。

FU indication

```
+---------------+
|0|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+-+
|F|NRI|  Type   |
+---------------+
```

这里面的的F和NRI已经在NALU的Header解释清楚了，就是NALU头的前面三个bit位，**后面的TYPE就是NALU的FU-A类型28**，这样在RTP固定头后面第一字节的后面5bit提取出来就确认了该RTP包承载的不是一个完整的NALU，是其一部分。
那么问题来了，一个NALU切分成多个RTP包传输，那么到底从哪儿开始哪儿结束呢？可能有人说RTP包固定头不是有mark标记么，注意区分那个是**以帧图像的结束标记**，这里要确定是NALU结束的标记，其次NALU的类型呢？那么就需要RTP固定12字节后面的Fu Header来进行区分。

FU header

```
+---------------+
|0|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+-+
|S|E|R|  Type   |
+---------------+
```

字段解释：
S: 1 bit 当设置成1,**开始位指示分片NAL单元的开始**。当跟随的FU荷载不是分片NAL单元荷载的开始，开始位设为0。
E: 1 bit 当设置成1, **结束位指示分片NAL单元的结束**，即，荷载的最后字节也是分片NAL单元的最后一个字节，当跟随的FU荷载不是分片NAL单元的最后分片,结束位设置为0。
也就是说一个NALU切片时，第一个切片的SE是10，然后中间的切片是00，最后一个切片时11。
R: 1 bit
保留位必须设置为0，接收者必须忽略该位。
Type: 5 bits
此处的Type就是**NALU头中的Type,取1-23的那个值**，表示 NAL单元荷载类型定义，

综上所述：

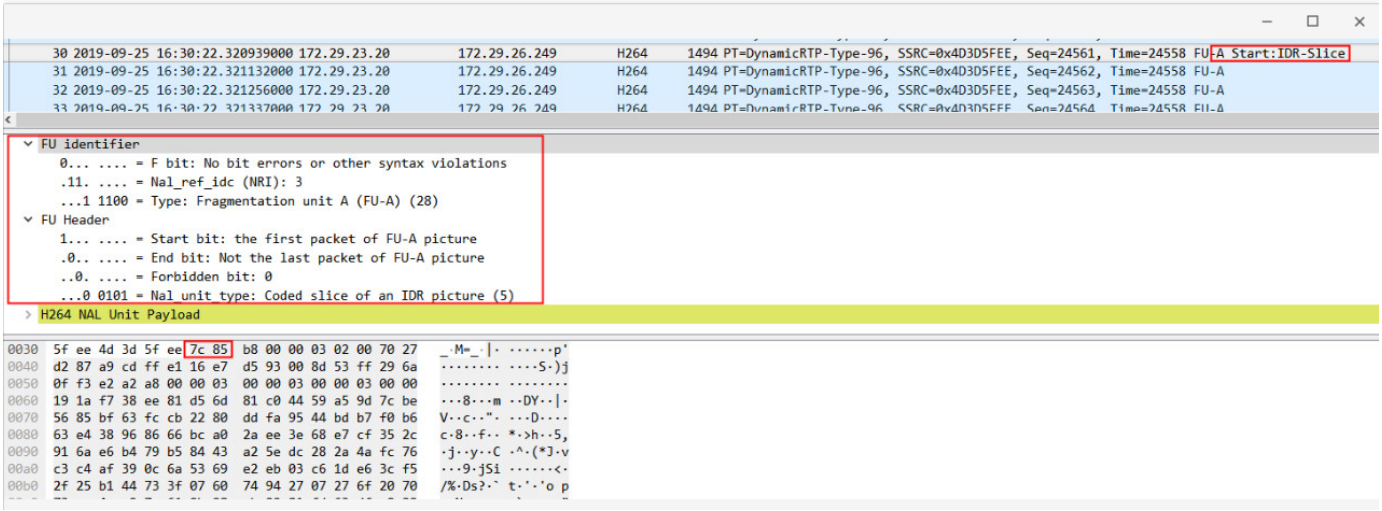
对于比较大的NLAU进行FU-A切片时，其中NALU的Header字段在RTP打包时划分为两个字节

1、NALU header的前3bit为RTP固定头后面第一个字节FU-indication的前3bit，后面5bit后面跟了FU-A打包这种类型28;

2、NALU header的后面5bit变成了RTP固定头第二字节的后面5bit,其中前3bit标识了分片的开始和结束。

为了验证这种打包方式，我们同样进行了写文件和抓包，对第一个IDR帧的NLAU采取的这种分片进行了研究。

第一个IDR帧的NALU第一个切片：



FU indication

十六机制：0x7C

二进制：0111 1100

FU header

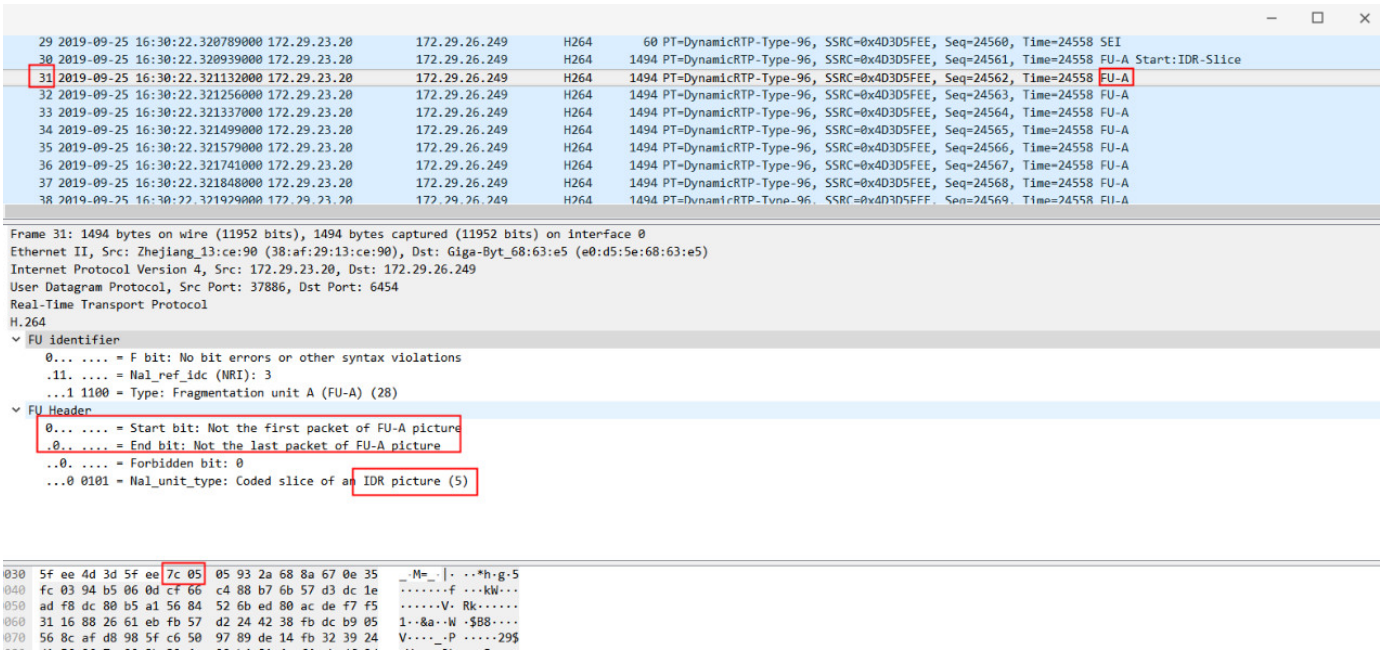
十六进制：0x85

二进制：1000 0101

这里的SE是10，则说明该RTP包承载的NALU的第一个切片。

这样我们提取FU indication字节的前3bit位和Fu header字节后5bit位则为**0110 0101即0x65**这刚好符合IDR帧的NALU Header定义,后面的b8 00 00 03等二进制就是NALU的DATA字段。

第一个IDR帧的NALU第二个切片：



FU indication

十六机制：0x7C

二进制：0111 1100

FU header

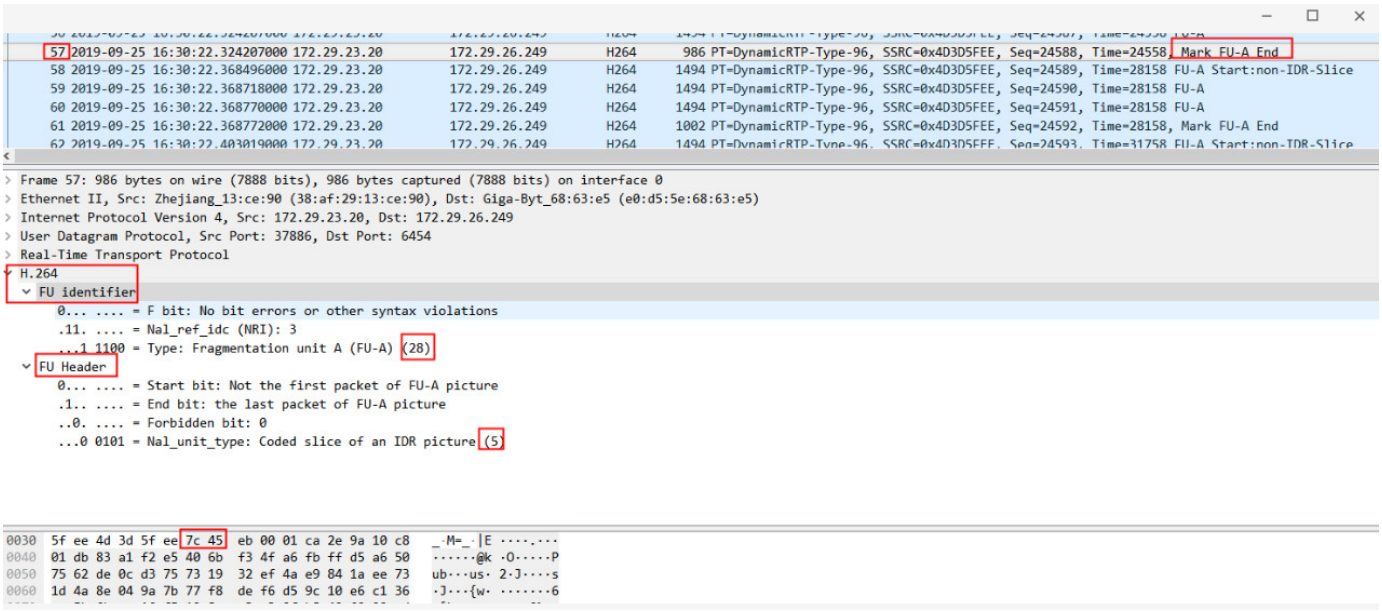
十六进制：0x05

二进制：0000 0101

这里的SE是00，则说明该RTP包承载的NALU的中间切片。

按照同样方法提取，则NALU Header的二进制为0110 0101即**0x65**，同样说明是IDR帧类型，类似这种的中间切片有很多，从31-56RTP包都是中间切片，直到最后一个NALU的切片。

第一个IDR帧的NALU最后一个切片：



FU indication

十六机制：0x7C

二进制：0111 1100

FU header

十六进制：0x45

二进制：0100 0101

这里的SE是01，则说明该RTP包承载的NALU的最后一个切片。

当然通过抓包你还可以到IDR帧时比较大的，而后面的P帧就相对比较小，因为一个NALU切片4次就完了，同样能看到时间戳的变化值等信息。

我们可以看到发送端一般采用Single NAL Unit和FU-A打包方式就基本可以将H264数据发送到接收端了，对于AAC音频来说，直接将ADTS头部去掉以1024字节组成一帧直接塞到RTP即可，打包并不难。至于其他的封装格式如PS、TS或者H265,VPx等数据如何打包RTP，以后再给大家进行分享，完善这个传输系列。