原创 猿诗人 于 2019-04-18 19:59:46 发布 4728 收藏 35 版权
分类专栏： h264传输-编解码 流媒体 RTSP/RTP/RTCP/RTMP H264 文章标签： H264



H264帧需要通过RTP协议进行传输，这其中就涉及到H264数据帧的封包、拆包和解包等过程。

**RTP协议格式**

下面是 RFC 3550 中规定的 RTP 头的结构：

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                               ....                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RTP协议头部共12字节，在每一个RTP packet中都会存在，头部之后就是RTP payload部分，也即是H264数据；这里只关注RTP payload部分。由于H264帧数据打包进入RTP packet时会把start code（00 00 00 01 或者00 00 01）去掉，所以RTP payload部分第一字节就是H264 NALU Header部分；

**H264 NALU**

H264帧是由一系列的NALU组成，NALU Header由一个字节组成,它的语法如下:

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

**F:** 1个比特. forbidden_*zero*_bit. 在 H.264 规范中规定了这一位必须为 0.禁止位，0表示正常，1表示错误，一般都是0；

**NRI:** 2个比特. nal_*ref*_idc. 取00~11,表示这个NALU的重要性,如00的NALU解码器可以丢弃它而不影响图像的回放.

**Type:** 5个比特，重要 nal_*unit*_type. 这个NALU单元的类型.简述如下:

| Type | Packet | Type name |
|---|---|---|
| 0 | undefined | - |
| 1-23 | NAL unit | Single NAL unit packet per H.264 |
| 24 | STAP-A | Single-time aggregation packet |
| 25 | STAP-B | Single-time aggregation packet |
| 26 | MTAP16 | Multi-time aggregation packet |
| 27 | MTAP24 | Multi-time aggregation packet |
| 28 | FU-A | Fragmentation unit |
| 29 | FU-B | Fragmentation unit |
| 30-31 | undefined | |

之前提到过，单个H264仅用1-23类型，24以及以后的用在RTP协议中 H264数据的头部，用于标识传输过程中H264的类型。

**H264打包过程**

对于一个原始的 H.264 NALU 单元常由 [Start Code] [NALU Header][NALU Payload] 三部分组成, 其中 StartCode 用于标示这是一个NALU 单元的开始, 必须是 "00 00 00 01" 或 "00 0001", NALU 头仅一个字节, 其后都是NALU 单元内容.打包时去除 "00 0001" 或 "00 00 00 01" 的开始码, 把其他数据封包的 RTP 包即可.

根据H264 NALU的数据长度和MTU进行比较的结果，RTP有三种打包模式：单一NALU模式、组合封包模式、分片封装模式。

**1、单一NALU模式**

对于NALU的长度小于MTU大小的包，一般采用单一NALU模式：

下面就是相应的格式示意：

```
0           1           2           3           4
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|NRI|  type |                                               |
+-+-+-+-+-+-+-+                                               |
|                                                             |
|             Bytes 2..n of a Single NAL unit                 |
|                                                             |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               :...OPTIONAL RTP padding       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

如有一个 H.264 的 NALU 是这样的:

[00 0000 01 67 42 A0 1E 23 56 0E 2F ... ]

这是一个序列参数集 NAL 单元. [00 00 00 01] 是四个字节的开始码, 67 是 NALU 头, 42 开始的数据是 NALU 内容.封装成 RTP 包将如下:

[ RTPHeader ] [ 67 42 A0 1E 23 56 0E 2F ]

**2、组合封包模式**

当多个NALU长度特别小时，可以将多个NALU一起封装在一个RTP包中。这里就对应着NALU Type为24 、 25 、26 、27.下面就是相应的格式示意：

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          RTP Header                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|STAP-A NAL HDR |         NALU 1 Size           | NALU 1 HDR    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          NALU 1 Data                          |
:                                                               :
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               | NALU 2 Size                   | NALU 2 HDR    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          NALU 2 Data                          |
:                                                               :
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
16      |                               :...OPTIONAL RTP padding        |
17      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

例如：如有一个 H.264 的 NALU 是这样的:

[00 00 00 01 67 42 A0 1E 23 56 0E 2F...]

[00 00 00 01 68 42 12 2B 6A D6 FF E4...]

封装成RTP包，结果如下：

[RTP Header] [78(STAP-A头，一个字节)] [第一个NALU长度，两个字节] [67 42 A0 1E 23 56 0E 2F...] [第二个NALU长度，两个字节] [68 42 12 2B 6A D6 FF E4...]

**3、分片封包模式**

当 NALU 的长度超过 MTU 时, 就必须对 NALU 单元进行分片封包. 也称为Fragmentation Units (FUs).

```
1    0            1              2              3              4
2    0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
3   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
4   | FU indicator  |   FU header   |                               |
5   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
6   |                                                               |
7   |                         FU payload                            |
8   |                                                               |
9   |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
10  |                               :...OPTIONAL RTP padding        |
11  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

FU indicator有以下格式：

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

FU指示字节的类型域Type=28表示FU-A。。NRI域的值必须根据原分片NAL单元的NRI域的值设置。

FU header的格式如下：

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

**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。

**R:** 1 bit 保留位必须设置为0，接收者必须忽略该位。

**Type:** 5 bits 取1-23的那个值，也就是原被分片的H264帧的原始NALU Type
NAL单元荷载类型定义见下表:

| nal_unit_type | NAL类型 | C |
|---|---|---|
| 0 | 未使用 | |
| 1 | 不分区、非IDR的片 | 2,3,4 |
| 2 | 片分区A | 2 |
| 3 | 片分区B | 3 |
| 4 | 版分区C | 4 |
| 5 | IDR图像中的片 | 2,3 |
| 6 | 补充增强信息单元（SEI） | 5 |
| 7 | 序列参数集 | 0 |
| 8 | 图像参数集 | 1 |
| 9 | 分界符 | 6 |
| 10 | 序列结束 | 7 |
| 11 | 码流结束 | 8 |
| 12 | 填充 | 9 |
| 13..23 | 保留 | |
| 24..31 | 不保留 | |



分片模式举例：一帧H264为：

00 00 00 01 21 42 0A DE DF FF AD 5D ...(需要第一个RTP Packet)...A5 00 0F 08 56 78 DA ...（需要第二个RTP Packet）...DE D6 D8 89 09 00 0F...(需要第三个RTP Packet)... 则按照分片封包模式，分为三个包，依次为：

第一个包：[RTP Header] [FU Indicator =3C ] [FU Header = 81] [21(去掉) 42 0A DE DF FF AD 5D ...][...]

第二个包：[RTP Header] [FU Indicator =3C ] [FU Header = 01] [A5 00 0F 08 56 78 DA ...][...]

第三个包：[RTP Header] [FU Indicator =3C ] [FU Header = 41] [DE D6 D8 89 09 00 0F...][...]

然后，总共三个RTP Packet发出去。

**解包过程：**

当接收端接收到一个一个的RTP Packet，就需要将其中的H264数据取出，并还原原始的H264帧，交由解码器解码。

单一NALU模式和组合封包模式解包过程其实比较简单，主要看一下分片封包模式：

1、当接收到接收到FU-A类型的RTP Packet时，需要看FU Header中的S / E两位：

- 如果时10，表示时一帧数据的开头部分；
- 如果是00，表示一帧数据的中间部分，有可能多个00的分片；
- 如果是01，表示一帧数据的结尾。

2、把之前所有的数据都暂时保存、拼接成完成的一帧H264数据，然后加上start code：00 00 00 01即可，然后就可以交由解码器解码；

3、还原后的NAL头的八位是由FU indicator的前三位加FU header的后五位组成，即：

nal_unit_type = (fu_indicator & 0xe0) | (fu_header & 0x1f)

**接下来，看代码**

```
//This method is used to get the NAL unit from the RTP packet
    //Then translate the NAL unit to the decoder
    public void H264PacketRecombine(byte[] data,int length) {
        nalType = data[0] & 0x1F;

        switch (nalType) {
            //Single-timeaggregation packet
            case NAL_UNIT_TYPE_STAP_A:
                break;

            //Single-timeaggregation packet
            case NAL_UNIT_TYPE_STAP_B:
                break;

            //Multi-time aggregationpacket
            case NAL_UNIT_TYPE_MTAP16:
                break;

            //Multi-time aggregationpacket
            case NAL_UNIT_TYPE_MTAP24:
                break;

            //Fragmentationunit
            case NAL_UNIT_TYPE_FU_A:
                packFlag = data[1] & 0xC0;
                switch (packFlag) {
                    //NAL Unit start packet
                    case 0x80://一帧的开头
                        NALUnit[4] = (byte) ((data[0] & 0xE0) | (data[1] & 0x1F));
                        System.arraycopy(data, 2, NALUnit, 5, length - 2);
                        tmpLen = length + 5 - 2;
                        break;
                    //NAL Unit middle packet
                    case 0x00:
                        System.arraycopy(data, 2, NALUnit, tmpLen, length - 2);
                        tmpLen +=length - 2;
                        break;
                    //NAL Unit end packet
                    case 0x40:
                        System.arraycopy(data, 2, NALUnit, tmpLen, length - 2);
                        tmpLen +=length - 2;

                        if (SquareApplication.getInstance().getH264Stream()!=null) {
                            SquareApplication.getInstance().getH264Stream().decodeH264Stream(Arrays.copyOfRange( I
                        }
                        tmpLen =0;

                        break;
                }
                break;

            //Fragmentationunit
            case NAL_UNIT_TYPE_FU_B:
                break;

            //Single NAL unit per packet
            default:
                if (SquareApplication.getInstance().getH264Stream()!=null) {

                    System.arraycopy(data, 0, BaseNALUnit, 4, length);
                    SquareApplication.getInstance().getH264Stream().decodeH264Stream(Arrays.copyOfRange( BaseNALU

                    tmpLen =0;
                }
                break;
        }

    }
```

接下来就是使用MediaCodec进行硬解码。

使用RTP包荷载H264码流数据 — 梳茶的博客 — 2554
在音视频通话中我们通常使用 RTP 协议包荷载音视频码流数据，例如用户摄像头采集图像后进行编码成帧，再将帧数据拆分到 RTP 协议包后发送到流媒…

基于RTP协议的H.264视频传输系统：实现 — 草上爬的博客 — 1万+
实现的原理：基于RTP协议的H.264视频传输系统：原理 相关文章： 【1】RTP协议分析 【2】jrtplib简介 【3】Qt调用jrtplib实现单播、多播和广播 【4】…

h264流，拆出来帧与其余帧，并能重新组装，完整工程 — 11-03
可以提取h264流中的帧和其他类型帧，并可以重新组装起来，还原出原来的h264流，使用了x264中代码，参考了网络上一位仁兄的代码

H264裸数据打包RTP方法和实验.pdf — 06-28
主要讲了H264数据格式，RTP数据格式，以及如何将H264的NALU打包为RTP。 本文的实验在VLC播放器进行了验证，能够将本地文件推送到VLC进行顺…

H264-encod-and-decod.rar_H264实时编码_h264传输 — 07-15
\bin 目录下是已编译成功的编解码器程序以及相应说明 \Idecod目录下是H.264视频解码器程序码。 \Iencod目录下是H.264视频编码器程序码。...\rtpdump …

RFC3984协议-H.264(RTP荷载格式).pdf — 06-23
RFC3984协议-H.264(RTP荷载格式).pdf

音视频传输：RTP协议详解和H.264打包方案 — weixin_41422027的博客 — 2598
问题背景： 前面讲解了PS、TS、FLV这三种媒体封装格式，现在新开一个系列讲解下传输协议，这里面会包含RTP、RTSP、HLS、RTMP等。当然最复…

H264码流分析和打包RTP过程 — 小孟来码的博客 — 1万+
对摄像头采集的每一帧视频需要进行编码，由于视频中存在空间和时间的冗余，需要用算法来去除这些冗余。H264是专门去除这些冗余的算法，我们把这…

网络摄像机（IPC）开发（7）：RTP协议解析（H264码流） — ProYuan的博客 — 2742
一、RTP（实时传输协议） RTP全名是Real-time Transport Protocol（实时传输协议），它是IETF提出的一个标准，对应的RFC文档为RFC3550（RFC1…

H264 RTP包解析 — l15902020106的博客 — 646
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…

ffmpeg5.0解码rtp协议传输的h264码流 — 06-07
要使用FFmpeg 5.0解码RTP协议传输的H.264码流，需要使用以下命令： ``` ffmpeg -i rtp://address:port -vcodec copy -f h264 - ``` 其中，address是RTP…

“相关推荐”对你有帮助么?

非常没帮助 没帮助 一般 有帮助 非常有帮助