

32 | HLS：实现一对多直播系统的必备协议

2019-09-26 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 16:08 大小 14.79M



在[上一篇文章](#)中，我们对 RTMP 协议和 HLS 协议的优势与劣势进行了比较。从比较的结果我们可以看出，RTMP 作为传统的直播传输技术在实时性方面要比 HLS 好很多，所以它还是有一定优势的。

不过，随着 Chrome 浏览器宣布不再对 Flash 插件提供支持、Adobe 公司停止对 RTMP 协议更新以及苹果公司声称 iOS 上不允许使用 RTMP 协议等一系列事件的发生，我们可以断定 RTMP 协议已失去了未来。

而 HLS 协议则恰恰相反，它在未来会有更广阔的应用前景。我们可以通过以下几点来得到这个结论：

HLS 是苹果开发的协议，苹果产品原生支持此协议；

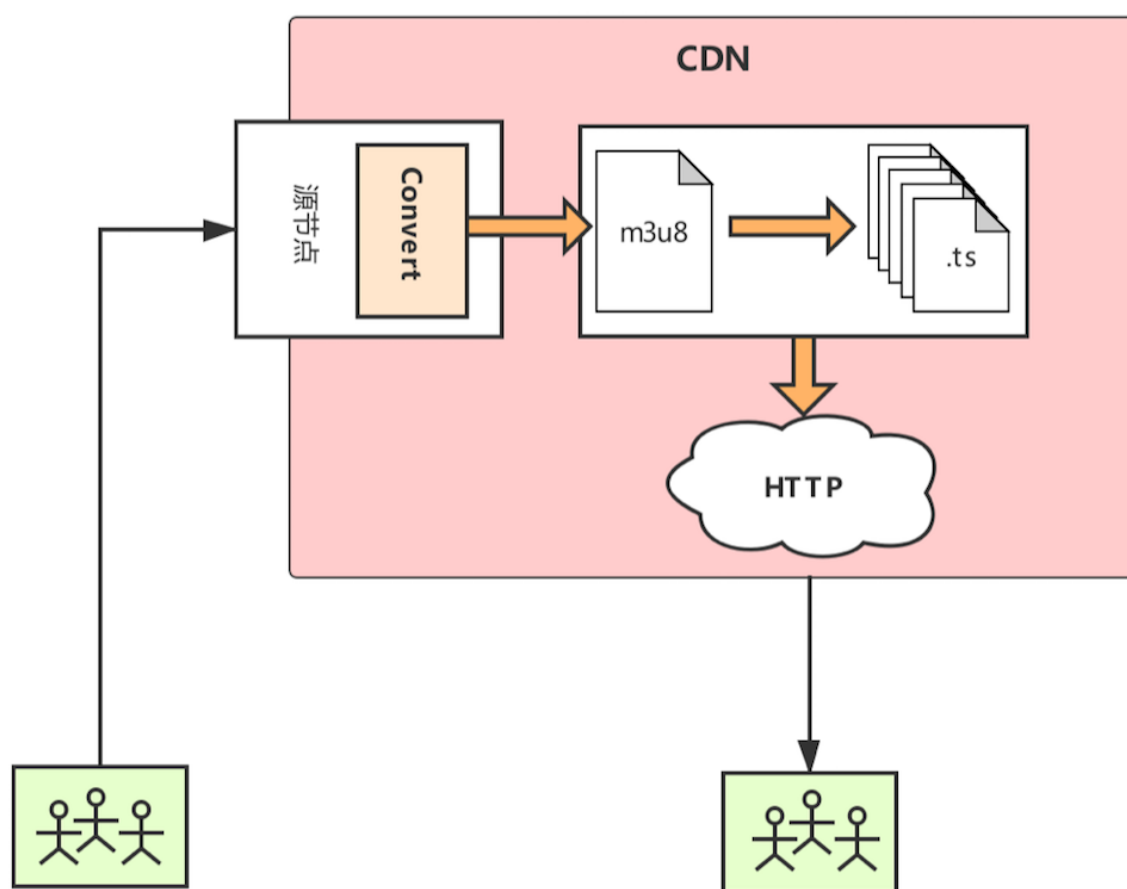
HLS 是基于 HTTP 的，可以不受防火墙限制，所以它的连通性会更好；

HLS 还能根据客户的网络带宽情况进行自适应码率的调整，这对于很多用户来说是非常有吸引力的。

基于以上原因，我们有必要从 HLS 直播架构、FFmpeg 生成 HLS 切片、HLS m3u8 格式和 HLS TS 格式这四个方面对 HLS 协议的细节做一下介绍。

HLS 直播架构

下面我们来看一下 HLS 直播系统的架构图，如下所示：



HLS 直播架构图

我们在[上一篇文章](#)中讲过，传统直播系统大致分为三部分：直播客户端、信令服务和 CDN 网络，使用 HLS 协议也是如此。只不过在我们这里为了简化流程，去掉了信令服务系统。

如上图所示，客户端采集媒体数据后，通过 RTMP 协议将音视频流推送给 CDN 网络的源节点（接入节点）。源节点收到音视频流后，再通过 Convert 服务器将 RTMP 流切割为

HLS 切片文件，即 .ts 文件。同时生成与之对应的 m3u8 文件，即 HLS 播放列表文件。

切割后的 HLS 分片文件 (.ts 文件) 和 HLS 列表文件 (.m3u8 文件) 经 CDN 网络转发后，客户端就可以从离自己最近的 CDN 边缘节点拉取 HLS 媒体流了。

在拉取 HLS 媒体流时，客户端首先通过 HLS 协议将 m3u8 索引文件下载下来，然后按索引文件中的顺序，将 .ts 文件一片一片下载下来，然后一边播放一边缓冲。此时，你就可以在 PC、手机、平板等设备上观看直播节目了。


对于使用 HLS 协议的直播系统来说，最重要的一步就是**切片**。源节点服务器收到音视频流后，先要数据缓冲起来，保证到达帧的所有分片都已收到之后，才会将它们切片成 TS 流。

为了便于分析，本文是通过 FFmpeg 工具将 MP4 文件切割成 HLS 格式的文件切片。但不管选择使用哪一种切割文件的方法或工具，生成的切片和索引文件的格式都是一致的。

勿在浮沙筑高台，为了让你在工作中做到得心应手、心中有数，接下来就让我们一起探索 HLS 协议的一些具体细节吧。

FFmpeg 生成 HLS 切片

这里我们是通过 FFmpeg 工具将一个 MP4 文件转换为 HLS 切片和索引文件的。所以，你需要预先准备一个 MP4 文件，并且下载好 FFmpeg 工具。你可以从[FFmpeg 官网](#)下载二进制包，也可以通过下载源码自行编译出 FFmpeg 工具。FFmpeg 用于将 MP4 切片成 HLS 的命令如下：

 复制代码

```
1 ffmpeg -i test.mp4 -c copy -start_number 0 -hls_time 10 -hls_list_size 0 -hls_segment_f:
```

该命令参数说明如下：

-i，输入文件选项，可以是磁盘文件，也可以是媒体设备。

-c copy，表示只是进行封装格式的转换。不需要将多媒体文件中的音视频数据重新进行编码。

-start_number, 表示 .ts 文件的起始编号, 这里设置从 0 开始。当然, 你也可以设置其他数字。

-hls_time, 表示每个 .ts 文件的最大时长, 单位是秒。这里设置的是 10s, 表示每个切片文件的时长, 为 10 秒。当然, 由于没有进行重新编码, 所以这个时长并不准确。

-hls_list_size, 表示播放列表文件的长度, 0 表示不对播放列表文件的大小进行限制。


-hls_segment_filename, 表示指定 TS 文件的名称。

index.m3u8, 表示索引文件名称。

执行完这条命令后, 在当前路径下会生成一系列 .ts 文件和 index.m3u8 文件。下面, 我们再分别分析一下 .m3u8 文件格式和 .ts 文件格式。

m3u8 格式分析

正如前面讲到, HLS 必须要有一个 .m3u8 的索引文件。它是一个播放列表文件, 文件的编码必须是 UTF-8 格式。这里我们将前面生成的 .m3u8 文件内容展示一下, 以便让你有个感观的认识。内容如下:

 复制代码

```
1 #EXTM3U
2 #EXT-X-VERSION:3           // 版本信息
3 #EXT-X-TARGETDURATION:11    // 每个分片的目标时长
4 #EXT-X-MEDIA-SEQUENCE:0     // 分片起始编号
5 #EXTINF:10.922578,          // 分片实际时长
6 test000.ts                  // 分片文件
7 #EXTINF:9.929578,           // 第二个分片实际时长
8 test001.ts                  // 第二个分片文件
9 ...
```

这里截取了分片列表文件开头部分的内容, 可以看出文件内容要么是以#字母开头, 要么就是没有#字母。关于文件格式规范, [RFC8216 草案](#)第四节有详细的说明, 你可以到那里查看详细的内容。

RFC8216 规定, .m3u8 文件内容以#字母开头的行是注释和 TAG, 其中 TAG 必须是#EXT 开头, 如上面示例中的内容所示。

接下来，我们对这几个 TAG 做一下说明：

EXTM3U 表示文件是第一个扩展的 M3U8 文件，此 TAG 必须放在索引文件的第一行。

EXT-X-VERSION: *n* 表示索引文件支持的版本号，后面的数字 *n* 是版本号数字。需要注意的是，一个索引文件只能有一行版本号 TAG，否则播放器会解析报错。

EXT-X-TARGETDURATION: *s* 表示 .ts 切片的最大时长，单位是秒 (s) 。

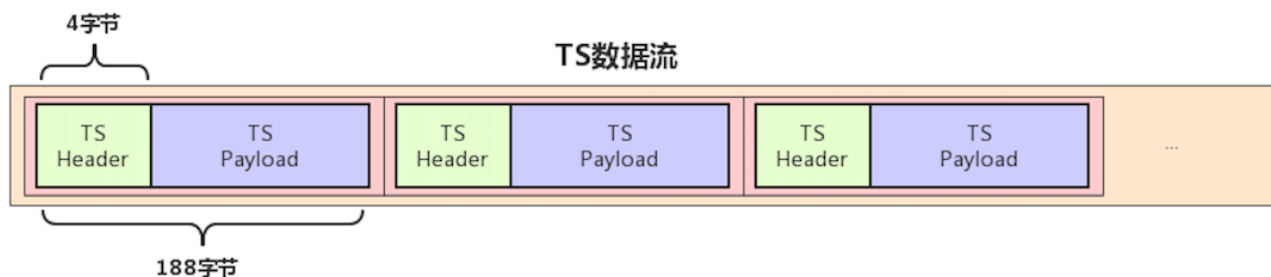
EXT-X-MEDIA-SEQUENCE: *number* 表示第一个 .ts 切片文件的编号。若不设置此项，就是默认从 0 开始的。

EXTINF: *duration*, *title* 表示 .ts 文件的时长和文件名称。文件时长不能超过#EXT-X-TARGETDURATION中设置的最大时长，并且时长的单位应该采用浮点数来提高精度。

TS 格式分析

TS 流最早应用于数字电视领域，其格式非常复杂，包含的配置信息表多达十几个。TS 流中的视频格式是 MPEG2 TS，格式标准是在 ISO-IEC 13818-1 中定义的。

苹果推出的 HLS 协议对 MPEG2 规范中的 TS 流做了精减，只保留了两个最基本的配置表 PAT 和 PMT，再加上音视频数据流就形成了现在的 HLS 协议。也就是说，HLS 协议是由 PAT + PMT + TS 数据流组成的。其中，TS 数据中的视频数据采用 H264 编码，而音频数据采用 AAC/MP3 编码。TS 数据流示意图如下所示：



TS 数据流示意图

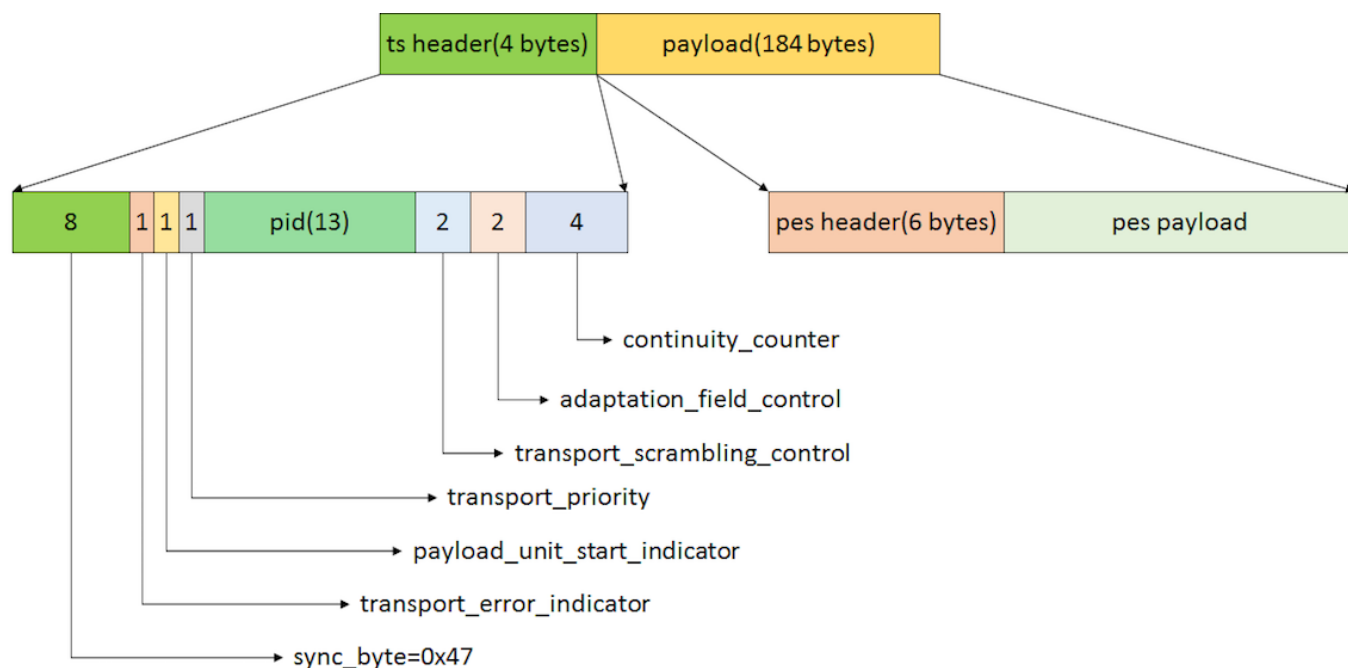
我们进一步细化，TS 数据流由 TS Header 和 TS Payload 组成。其中，TS Header 占 4 字节，TS Payload 占 184 字节，即 TS 数据流总长度是 188 字节。

TS Payload 又由 PES Header 和 PES Payload 组成。其中，PES Payload 是真正的音视频流，也称为 ES 流。

PES (Packet Elementary Stream) 是将 ES 流增加 PES Header 后形成的数据包。

ES (Elementary Stream) ，中文可以翻译成**基流**，是编码后的音视频数据。

下面我们就来分析一下 TS 数据流的格式，如下图所示：



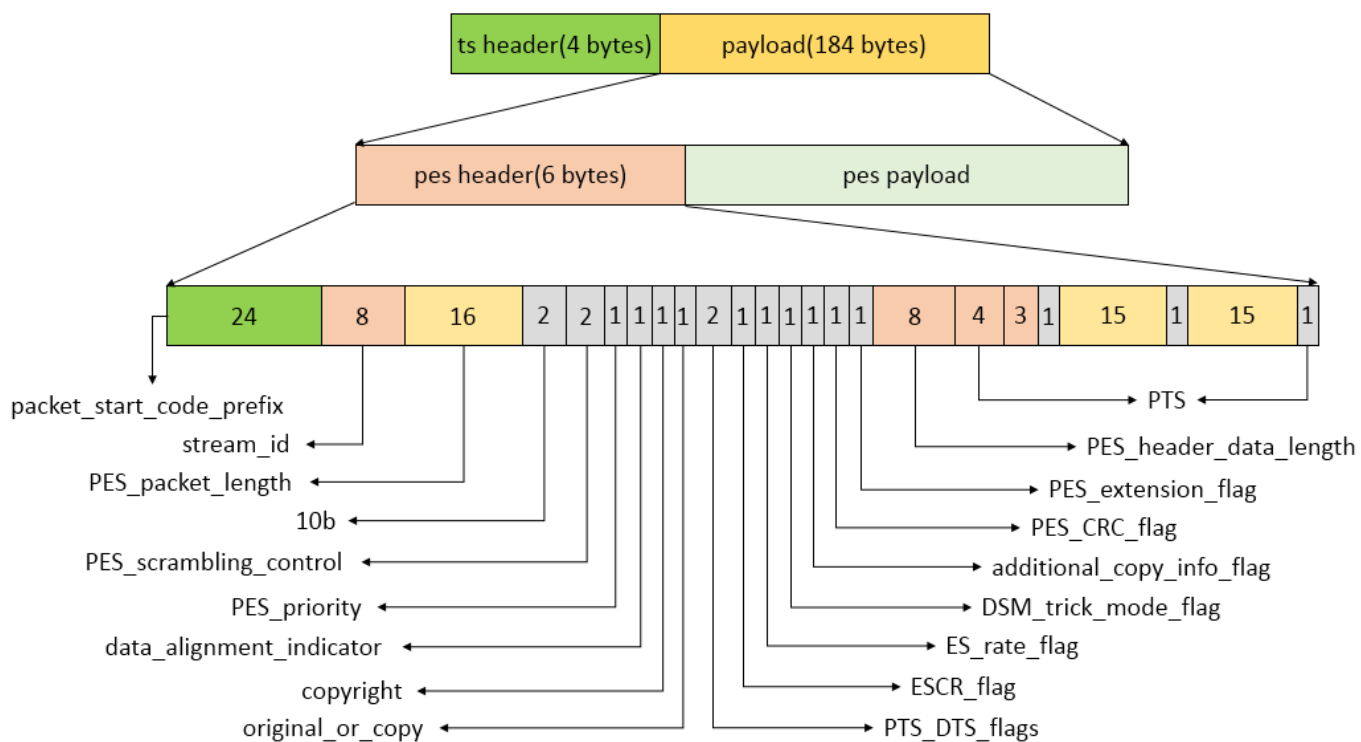
TS 文件格式图

这是 TS Header 各个字段的详细说明，图中数字表示长度，如果数字后面带有 bytes ，单位就是 bytes；否则，单位都是 bit。

TS Header 分为 8 个字段，下面我们分别解释一下：

| 字段名称 | 单位 (bit) | 字段说明 |
|------------------------------|-------------|---|
| sync_byte | 8 | 固定值是 0x47，算是 magic 字段，用它可以判断是否是 TS 文件。 |
| transport_error_indicator | 1 | 传输错误指示标志，如果此标志为 1，表示发生了不可纠正的错误。 |
| payload_unit_start_indicator | 1 | 标识 TS 数据流的 Payload 域中包含的是否是 PES 数据。 为1，表示 TS Header 后面第一个字节是 PES Header 的内容； 为0，表示后面直接是 ES 数据。 |
| transport_priority | 1 | 包优先级标志，设置为 1 表示高优先级。 |
| PID | 13 | Packet Identifier 是每个 ES 的唯一标识。PID 的部分值是保留的，用来区分 Payload 的类型。 PID 的分配表，请参考 ISO-IEC-13818-1 2.4.3.3 节。 |
| transport_scrambling_control | 2 | 扰码控制模式，有四种模式：00 表示不加扰码；01、10、11 由用户自定义。 |
| adaptation_field_control | 2 | 表示在 TS Header 后面是否有适配控制字段。 00：属于保留值； 01：没有适配字段，只有 Payload； 10：只有适配字段，没有 Payload； 11：适配字段后面是 Payload。 |
| continuity_counter | 4 | TS 数据包累加计数器，对于同一个 PID 的 TS 流，此字段每次都会增加，最大值是 15，溢出后，从 0 再开始计数。 此值受 adaptation_field_control 字段控制，当取值 00 或 10 时，计数不累加。这也说明累加器只对包含 Payload 的 TS 数据包起作用。 |

PES Packet 作为 TS 数据流的 Payload，也有自己的 Header，如下图所示：



PES 结构图

下面我们就对这些常用的字段——做下解释，当然也还有很多不常用的字段，我们这里就不列出来了，如有需求，可参考 ISO-IEC 13818-1 2.4.3.7 节。

PES Header 长度是 6 字节，字段说明如下：

| 字段名称 | 长度 (bit) | 字段说明 |
|---------------------------|-------------|---|
| packet_start_code_prefix | 24 | PES 包前缀，固定值是 0x000001。 |
| stream_id | 8 | 节目流 ID，用于标识 ES 流。 音频取值（0xc0-0xdf），通常为 0xc0； 视频取值（0xe0-0xef），通常为 0xe0。 |
| PES_packet_length | 16 | 表示 PES 包长度，从 PES_packet_length 字段后的第一个字节开始计算。 如果此值为 0，表示长度不受限制。 |
| PES_scrambling_control | 2 | PES 包扰码模式。 |
| PES_priority | 1 | PES 包优先级，取值 1 优先级更高。 |
| data_alignment_indicator | 1 | 数据对齐标志，一般为 0。 |
| copyright | 1 | 表示 PES Payload 是否受版权保护，一般为 0。 |
| original_or_copy | 1 | 表示 PES Payload 是否原创。 1：原创； 0：copy。 |
| PTS_DTS_flags | 2 | 表示 PTS 和 DTS 是否包含在 PES 包中。 10：只是包含 PTS； 11：同时包含 PTS 和 DTS。 |
| ESCR_flag | 1 | 表示 PES Packet 是否包含 ESCR，一般为 0。 |
| ES_rate_flag | 1 | 表示 PES Packet 是否包含 ES_rate 字段，一般为 0。 |
| DSM_trick_mode_flag | 1 | 表示 PES Packet 是否包含 8 bit trick mode 字段，一般为 0。 |
| additional_copy_info_flag | 1 | 表示 PES Packet 是否包含 additional_copy_info 字段，一般为 0。 |
| PES_CRC_flag | 1 | 表示 PES Packet 是否包含 CRC 字段，一般为 0。 |
| PES_extension_flag | 1 | 表示 PES Packet 是否包含 extension 字段，一般为 0。 |
| PES_header_data_length | 8 | 表示 PES Packet 可选字段和填充位的长度。 |

另外，PTS（Presentation Timestamp）字段总共包含了 40 bit，高 4 个 bit 固定取值是 0010；剩下的 36 个 bit 分三部分，分别是：3 bit+1 bit 标记位；15 bit+1 bit 标记位；15 bit+1 bit 标记位。

通过以上的描述我们就将 HLS 协议中最重要的 TS 数据流向你介绍清楚了。

小结

本文我们首先讲述了采用 HLS 协议的直播架构。实际上该直播架构与我们上文中介绍的直播架构是一致的，只不过我们这里为了强调 HLS 协议对之前的直播架构做了简化。同时，我们还通过该直播架构模型向你介绍了传统直播系统中，从用户推流到服务端切片、再到用户拉流的基本过程。

随后，我们借助 FFmpeg 工具向你讲解了如何将 MP4 文件转换成 HLS 文件，并向你展示了 .m3u8 文件的基本结构。最后还重点介绍了 TS 数据流的格式。

通过本文的学习，我相信你应该已经对 HLS 协议有了非常清楚的认知了。实际上，作为应用级的开发人员来说，你并不需要了解到文中所介绍的那么细，只需要对 HLS 协议有个基本的理解就可以了。因为目前有很多不错的开源库已经完成了大部分的工作，你只需要将这些开源库使用好即可。

比如在播放 HLS 流时，就有很多开源库可用。在移动端可以使用ijkplayer，在浏览器上可以使用video.js，在PC端可以使用VLC。而服务端的HLS切片则是由CDN网络完成的，你只需要向CDN网络推流就可以了，CDN网络会直接将上传的流进行HLS切片。而在CDN网络内部，它就是使用我们上面所介绍的FFmpeg开源库编译好的工具来完成切片工作的。

思考时间

每个TS格式数据包是188字节长，不够188字节就需要用Padding填充，那为什么要限制成188字节呢？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

从 0 打造音视频直播系统

手把手教你打造实时互动音视频直播系统

李超

新东方音视频直播技术专家
前沪江音视频架构师



新版升级：点击「👤 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 31 | 一对多直播系统RTMP/HLS，你该选哪个？

下一篇 33 | FLV：适合录制的多媒体格式

精选留言 (1)

写留言



tech2ipo

2019-09-26

思考题回答：因为高清TS数据包为204字节，标清TS数据包为188字节。当一个媒体数据流小于5个TS包时，这个媒体数据流的长度不可能同时被204和188同时整除。

不太确定，是这样吗？

展开

