# 33 | FLV：适合录制的多媒体格式

2019-09-28 李超



**讲述：李超**

时长 14:19　大小 13.12M

虽然苹果拒绝使用 RTMP 协议并推出了自己的 HLS 技术，但大多数用户仍然还是使用 RTMP 协议作为传统直播系统的传输协议。在 Adobe 宣布不再对 RTMP 技术进行支持的情况下，仍然还有这么多用户在使用它，说明 RTMP 协议具有其他协议不可比拟的优势。

这里我们做个对比，你就知道 RTMP 协议的优势在哪里了。

首先，与 HLS 技术相比，RTMP 协议在传输时延上要比 HLS 小得多。主要原因在于 HLS 是基于切片（几秒钟视频的小文件）、然后缓存的技术，这种技术从原理上就比直接进行数据传输慢很多，事实也证明了这一点。

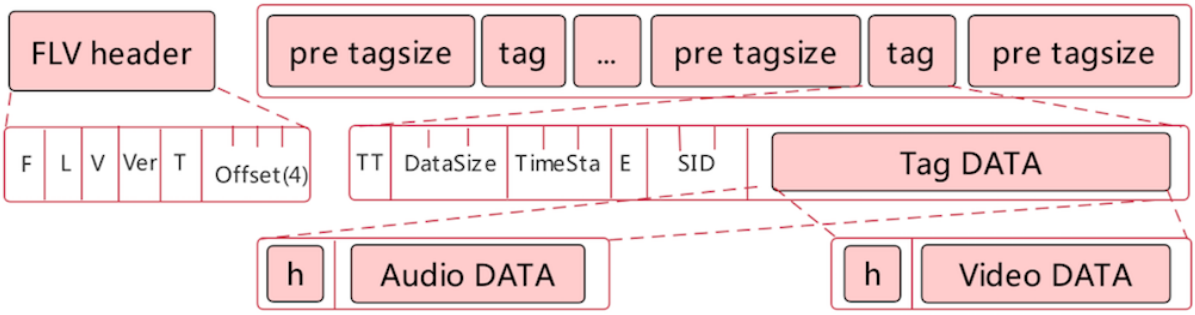
其次，相对于 RTP 协议，RTMP 底层是基于 TCP 协议的，所以它不用考虑数据丢包、乱序、网络抖动等问题，极大地减少了开发人员的工作量；而使用 RTP 协议，网络质量的保障都需要自己来完成。

最后，与现在越来越火的 WebRTC 技术相比，RTMP 也有它自己的优势。虽然在实时传输方面 WebRTC 甩 RTMP 技术几条街，但对于实时性要求并没有那么高的传统直播来说，RTMP 协议在音视频的服务质量上普遍要好于 WebRTC 的音视频服务质量。

这下你知道 RTMP 协议为什么会存活这么久了吧。那说了这么多，RTMP 协议与 FLV 又有什么关系呢？实际上，FLV 文件与 RTMP 之间是“近亲”关系，甚至比“近亲”还要近，亲得就好像是“一个人”似的。

## FLV 文件格式

我们先来看一下 FLV 的文件格式，如下图所示：




FLV 文件格式结构图

这是我阅读FLV 格式规范文档后，总结出的 FLV 文件格式结构图。从图中我们可以看出，FLV 文件格式由 FLV Header 和 FLV Body 两部分组成。其中，FLV Header 由 9 个字节组成，Body 由 Pre TagSize 和 Tag 组成。

为使你对它们有一个更清楚的认知，下面我们就来详细介绍一下 FLV Header 和 FLV Body。

### 1. FLV Header

它由 9 个字节组成：3 个字节的 “F” “L” “V”字母，用于标记该文件是 FLV 文件；1 个字节的 Version，指明使用的 FLV 文件格式的版本；1 个字节的 Type 标识，用于表明该 FLV 文件中是否包括音频或视频；4 个字节的 FLV Header 长度，由于 FLV 文件格式头是固定 9 个字节，所以这个字段设置得有点多余。

Type 标识（TypeFlag）又可以细分为： 1bit 用于标识 FLV 文件中是否有音频数据；1bit 标识 FLV 文件中是否有视频数据；如果两个 bit 位同时置 1，说明该 FLV 文件中既有音频数据又有视频数据，这也是通常情况下 FLV Header 的设置；除了两个 bit 的音视频数据标识外，其他位都是预留位，必须全部置 0。详细的含义可以参考下面张图表：

| | | |
|---|---|---|
| Signature | UI8 | Signature byte always 'F' (0x46) |
| Signature | UI8 | Signature byte always 'L' (0x4C) |
| Signature | UI8 | Signature byte always 'V' (0x56) |
| Version | UI8 | File version (for example, 0x01 for FLV version 1) |
| TypeFlagsReserved | UB[5] | Must be 0 |
| TypeFlagsAudio | UB[1] | Audio tags are present |
| TypeFlagsReserved | UB[1] | Must be 0 |
| TypeFlagsVideo | UB[1] | Video tags are present |
| DataOffset | UI32 | Offset in bytes from start of file to start of body (that is, size of header) |

这张图表清晰地表达了 FLV Header 中每个域所占的字节以及该域的具体含义。另外，如果你使用的是 Windows 系统，就可以安装 **FlvAnalyzer** 工具，该工具是一款功能非常强大的 FLV 文件分析工具。使用它打开任何一个 FLV 文件，就可以看到该文件的 FLV 格式。

## 2. FLV Body

从“FLV 文件格式结构图”我们可以看出，FLV Body 是由多个 Previous TagSize 和 Tag 组成的。其含义如下图表所示，其中 PreviousTagSize 占 4 个字节，表示前一个 Tag 的大小。这里需要注意的是，第一个 Previous TagSize 比较特殊，由于它前面没有 Tag 数据，所以它的值必须为 0。

| **FLV file body** | | |
|---|---|---|
| **Field** | **Type** | **Comment** |
| PreviousTagSizeN-1 | UI32 | Size of second-to-last tag |
| TagN | FLVTAG | Last tag |
| PreviousTagSizeN | UI32 | Size of last tag |

接下来我们再来看一下 FLV 中的 Tag，从 FLV 文件格式结构图中我们可以看到 Tag 由两部分组成，即 Tag Header 和 Tag Data 。

Tag Header 各字段的含义如下图所示：

| FLVTAG | | |
|---|---|---|
| **Field** | **Type** | **Comment** |
| TagType | UI8 | Type of this tag. Values are:<br>8: audio<br>9: video<br>18: script data<br>all others: reserved |
| DataSize | UI24 | Length of the data in the Data field |
| Timestamp | UI24 | Time in milliseconds at which the data in this tag applies. This value is relative to the first tag in the FLV file, which always has a timestamp of 0. |
| TimestampExtended | UI8 | Extension of the Timestamp field to form a SI32 value. This field represents the upper 8 bits, while the previous Timestamp field represents the lower 24 bits of the time in milliseconds. |
| StreamID | UI24 | Always 0 |
| Data | If TagType == 8<br>AUDIODATA<br>If TagType == 9<br>VIDEODATA<br>If TagType == 18<br>SCRIPTDATAOBJECT | Body of the tag |

TagType，占 1 个字节，表示该 Tag 的类型，可以是音频、视频和脚本。如果类型为音频，说明这个 Tag 存放的是音频数据；如果类型是视频，说明存放的是视频数据。

DataSize，占 3 个字节，表示音频 / 视频数据的长度。

Timestamp 和扩展 Timestamp，一共占 4 个字节，表示数据生成时的时间戳。

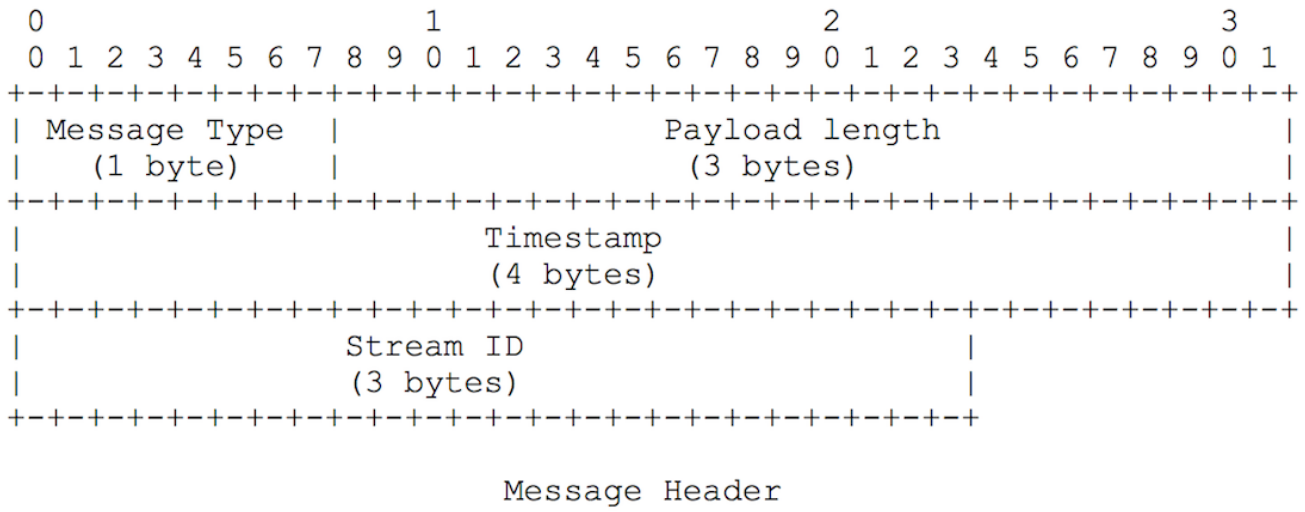
StreamID，占 3 个字节，总是为 0。

而 Tag Data 中存放的数据，是根据 TagType 中的类型不同而有所区别的。也就是说，假如 TagType 指定的是音频，那么 Tag Data 中存放的就是音频数据；如果 TagType 指定

的是视频，则 Tag Data 中存放的就是视频数据。

另外，无论 TagData 中存放的是音频数据还是视频数据，它们都是由 Header 和 Data 组成。也就是说，如果该 Tag 是一个音频 Tag ，那么它的数据就是由“AudioHeader + AudioData”组成；如果是一个视频 Tag，则它的数据是由“VideoHeader + VideoData”组成。

特别有意思的一点是，如果你翻看RTMP 协议，查看它的 6.1.1 小节，你会发现它定义的 RTMP Message Header 与 Tag Header 是一模一样的。下图是我从 RTMP 协议中截取的协议头：




Message Header

因此，我们可以说**FLV 文件就是由“FLV Header + RTMP 数据”构成的。**这也揭开了 FLV 与 RTMP 之间的关系秘密，即 **FLV 是在 RTMP 数据之上加了一层“马甲”。**

## 为什么 FLV 适合录制

通过上面的描述你会发现，**FLV 文件是一个流式的文件格式。**该文件中的数据部分是由多个 “PreviousTagSize + Tag”组成的。这样的文件结构有一个天然的好处，就是你可以将音视频数据随时添加到 FLV 文件的末尾，而不会破坏文件的整体结构。

在众多的媒体文件格式中，只有 FLV 具有这样的特点。像 MP4、MOV 等媒体文件格式都是结构化的，也就是说音频数据与视频数据是单独存放的。当服务端接收到音视频数据后，如果不通过 MP4 的文件头，你根本就找不到音频或视频数据存放的位置。

正是由于 FLV 是流式的文件格式，所以它特别适合在音视频录制中使用。这里我们举个例子，在一个在线教育的场景中，老师进入教室后开启了录制功能，服务端收到信令后将接收到的音视数据写入到 FLV 文件。在上课期间，老师是可以随时将录制关掉的，老师关闭录制时，FLV 文件也就结束了。当老师再次开启录制时，一个新的 FLV 文件被创建，然后将收到的音视频数据录制到新的 FLV 文件中。 这样当一节课结束后，可能会产生多个 FLV 文件，然后在收到课结束的消息后，录制程序需要将这几个 FLV 文件进行合并，由于 FLV 文件是基于流的格式，所以合并起来也特别方便，只需要按时间段将后面的 FLV 直接写到前面 FLV 文件的末尾即可。

使用 FLV 进行视频回放也特别方便，将生成好的 FLV 直接推送到 CDN 云服务，在 CDN 云服务会将 FLV 文件转成 HLS 切片，这样用户就可以根据自己的终端选择使用 FLV 或 HLS 协议回放录制好的视频。

而对于回放实时性要求比较高的业务，还可以将 FLV 按 3～5 分钟进行切片，这样就可以在直播几分钟后看到录制好的内容了。

另外，FLV 相较 MP4 等多媒体文件，它的文件头是固定的，音视频数据可以随着时间的推移随时写入到文件的末尾；而 MP4 之类的文件，文件头是随着数据的增长而增长的，并且体积大，处理时间长。因此， **FLV 文件相较于其他多媒体文件特别适合于在录制中使用。**

## 小结

本文首先向你详细介绍了 FLV 文件格式，通过对该文件格式的讲解，你可以发现 FLV 与 RTMP 协议的关系十分密切，在 RTMP 数据之上加个 FLV 头就成了 FLV 文件。

在本文的后半段，我们还讲解了 FLV 用在服务端录制上的好处。由于它是基于流式的文件存储结构，可以随时将音视频数据写入到 FLV 文件的末尾，且文件头不会因文件数据的大小而发生变化，所以不管是在录制时，还是进行回放时，相较于 MP4 之类的多媒体格式，FLV 都更有优势。

虽然 FLV 协议比较简单，但如果我们自己去实现 FLV 格式分析的话，还是要做很多的工作。庆幸的是，目前已经有很多非常不错的处理 FLV 文件格式的开源库了，如 FLVParser。我们只需要拿来直接使用即可。同时也有很多不错的 FLV 分析工具，比如前面提及的 FlvAnalyzer 就是一款功能非常强大的 FLV 文件格式分析工具，通过这些工具可以让我们更快速地理解 FLV 协议。

RTMP 与 FLV 就像是同一个人，只不过穿的“衣服”不同而已，所以在 RTMP 与 FLV 相互转换时也很容易。比如，我们想将 FLV 以 RTMP 方式进行推流就非常简单，而且还有一个专门的 RTMP 库，即 librtmp，可以帮我们完成底层转换的处理逻辑。

## 思考时间

你知道有哪些有名的能够播放 FLV 格式文件的开源播放器吗？

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。



上一篇　32 | HLS：实现一对多直播系统的必备协议

## 精选留言 (5)

写留言

**XiaoJun**
2019-09-29

bilibili 的 flv.js，可以用 h5 播放 flv

展开

作者回复: 不错！非常好的播放器

**Jason**
2019-09-29

potplayer

展开

作者回复: 这个以前还真没用过，回头试试！

**Jason**
2019-09-29

讲的很清楚，赞

展开

作者回复: 谢谢！

**刘丹**
2019-09-28

FLV Body里为什么把TagSize放在Tag后面呢？如果TagSize在Tag前面，解析时就知道需要读入多少字节了。

作者回复: 这是咱们正常的思维。但FLV标准的制定者不是这么考虑的。他们认为通过 Tag中的音频头或视频头就可以知道 Tag 的长度，然后后面再用 TagSize校验一下长度是否OK比较方便。

**刘丹**
2019-09-28

请问合并2个FLV文件的时候，是否要把第2个FLV文件的FLV头这几个字节删除掉?

作者回复: 是要去掉的