



min2011

博客园
新随笔
订阅

随笔 - 133 文章 - 3 评论 - 0

昵称: min2011
园龄: 3年3个月
粉丝: 12
关注: 14
+加关注

< 2016年6月						
日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

常用

我的

- iOS开发解
- ANE打
- iOS
- php服务器端
- Unity3D
- xcode制作静
- iOS开发
- iOS学习流程or
- iOS应用安
- iOS应用企业证书

随笔分

- C/C++/OC知识点拾
- FFmp
- iOS AudioQu
- iOS Ope
- iOS OpenG

RTP时间戳

一、RTP协议分析

第1章. RTP概述

1.1. RTP是什么

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（RFC1889为其过期版本）。RFC3550不仅定义了RTP，而且定义了配套的相关协议RTCP（Real-time Transport Control Protocol，即实时传输控制协议）。RTP用来为IP网上的语音、图像、传真等多种需要实时传输的多媒体数据提供端到端的实时传输服务。RTP为Internet上端到端的实时传输提供时间信息和流同步，但并不保证服务质量，服务质量由RTCP来提供。

1.2. RTP的应用环境

RTP用于在单播或多播网络中传送实时数据。它们典型的应用场合有如下几个。

简单的多播音频会议。语音通信通过一个多播地址和一对端口来实现。一个用于音频数据（RTP），另一个用于控制包（RTCP）。

音频和视频会议。如果在一次会议中同时使用了音频和视频会议，这两种媒体将分别在不同的RTP会话中传送，每一个会话使用不同的传输地址（IP地址+端口）。如果一个用户同时使用了两个会话，则每个会话对应的RTCP包都使用规范化名字CNAME（Canonical Name）。与会者可以根据RTCP包中的CNAME来获取相关联的音频和视频，然后根据RTCP包中的计时信息(Network time protocol)来实现音频和视频的同步。

翻译器和混合器。翻译器和混合器都是RTP级的中继系统。翻译器用在通过IP多播不能直接到达的用户区，例如发送者和接收者之间存在防火墙。当与会者能接收的音频编码格式不一样，比如有一个与会者通过一条低速链路接入到高速会议，这时就要使用混合器。在进入音频数据格式需要变化的网络前，混合器将来自一个源或多个源的音频包进行重构，并把重构后的多个音频合并，采用另一种音频编码进行编码后，再转发这个新的RTP包。从一个混合器出来的所有数据包要用混合器作为它们的同步源（SSRC，见RTP的封装）来识别，可以通过贡献源列表（CSRC表，见RTP的封装）可以确认谈话者。

1.3. 相关概念

1.3.1. 流媒体

流媒体是指Internet上使用流式传输技术的连续时基媒体。当前在Internet上传输音频和视频等信息主要有两种方式：下载和流式传输两种方式。

下载情况下，用户需要先下载整个媒体文件到本地，然后才能播放媒体文件。在视频直播等应用场合，由于生成整个媒体文件要等直播结束，也就是用户至少要在直播结束后才能看到直播节目，所以用下载方式不能实现直播。

流式传输是实现流媒体的关键技术。使用流式传输可以边下载边观看流媒体节目。由于Internet是基于分组传输的，所以接收端收到的数据包往往有延迟和乱序（流式传输构建在UDP上）。要实现流式传输，就是要从降低延迟和恢复数据包时序入手。在发送端，为降低延迟，往往对传输数据进行预处理（降低质量和高效压缩）。在接收端为了恢复时序，采用了接收缓冲；而为了实现媒体的流畅播放，则采用了播放缓冲。

使用接收缓冲，可以将接收到的数据包缓存起来，然后根据数据包的封装信息（如包序号和时戳等），将乱序的包重新排序，最后将重新排序了的数据包放入播放缓冲播放。

为什么需要播放缓冲呢？容易想到，由于网络不可能很理想，并且对数据包排序需要处理时耗，我们得到排序好的数据包的时间间隔是不等的。如果不用播放缓冲，那么播放节目会很卡，这叫时延抖动。相反，使用播放缓冲，在开始播放时，花费几十秒钟先将播放缓冲填满（例如PPLIVE），可以有效地消除时延抖动，从而在不太损失实时性的前提下实现流媒体的顺畅播放。

到目前为止,Internet 上使用较多的流式视频格式主要有以下三种:RealNetworks 公司的RealMedia ,Apple 公司的QuickTime 以及Microsoft 公司的Advanced Streaming Format

iOS开发
笔谈与实战(原
工作与人生的思考(原
音视频编解码 (转载

随笔档案

- 2015年11月
- 2015年
- 2015年
- 2015年
- 2015年5月
- 2015年4月
- 2015年
- 2015年
- 2014年11月
- 2014年11月
- 2014年
- 2014年
- 2014年7月
- 2014年6月
- 2014年5月

积分与排名

积分 -
排名 -

最新评论

- 1. Re: 实战FFmpeg -- 编译iOS平台使用的FFmpeg库 (支持arm64的FFmpeg) -- 好详细 -- songxing
- 2. Re: 谈一谈做iOS播放器库开发所涉及的知识点 -- 一个好人
- 3. Re: 笔谈OpenGL ES 实战FFmpeg + OpenGL ES -- iOS平台上的视频解码和播放该文本被密码保护。 posted @ 2015-11-18 18:33 min2011 阅读(26) | 评论 (0) 编辑
- 4. Re: 实战FFmpeg -- 编译iOS平台使用的FFmpeg库 (支持arm64的FFmpeg) @sanarara可以, 你看下FFmpeg的libavcodec.henum AVCodecID { AV_CODEC_ID_NONE, /* video codecs */ -- mi
- 5. Re: 实战FFmpeg -- 编译iOS平台使用的FFmpeg库 (支持arm64的FFmpeg) 兄弟, 这个能用来解码吗? -- san

阅读排行

- 1. iOS硬解H.264: -VideoToolboxDemo 析[草稿]
- 2. iOS学习笔记
- 3. OpenGL ES教程系列 (经典合集)
- 4. [破解版] Unity3d引擎最新稳定版本4.5.1 (官方最新稳定版本)
- 5. xcode 手动管理内存 的相关知识点总结
- 6. 实战FFmpeg -- 编译iOS平台使用的FFmpeg库 (支持arm64的FFmpeg2.6.2)
- 7. Unity3D如何接入第三方的SDK - iOS篇
- 8. Mac OS X上搭建Apache、PHP、MySQL

(ASF)。

上面在谈接收缓冲时，说到了流媒体数据包的封装信息（包序号和时戳等），这在后面的RTP封装中会有体现。另外，RealMedia这些流式媒体格式只是编解码有不同，但对于RTP来说，它们都是待封装传输的流媒体数据而没有什么不同。

第2章. RTP详解

2.1. RTP的协议层次

2.1.1. 传输层的子层

RTP（实时传输协议），顾名思义它是用来提供实时传输的，因而可以看成是传输层的一个子层。图 1给出了流媒体应用中的一个典型的协议体系结构。

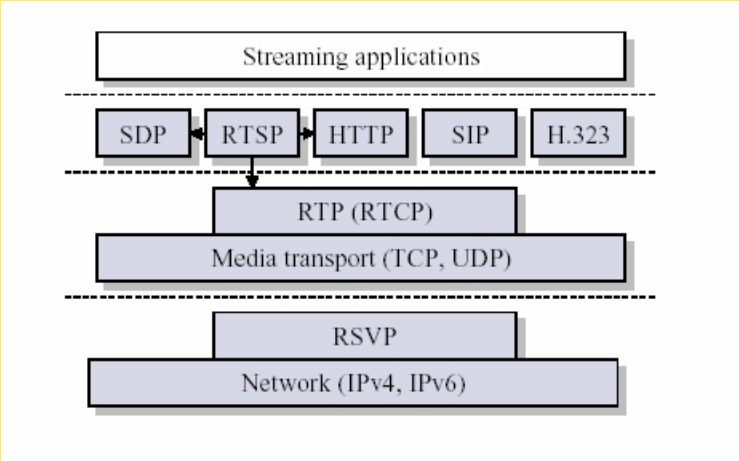


图 1 流媒体体系结构

从图中可以看出，RTP被划分在传输层，它建立在UDP上。同UDP协议一样，为了实现其实时传输功能，RTP也有固定的封装形式。RTP用来为端到端的实时传输提供时间信息和流同步，但并不保证服务质量。服务质量由RTCP来提供。这些特点，在第4章可以看到。

2.1.2. 应用层的一部分

不少人也把RTP归为应用层的一部分，这是从应用开发者的角度来说的。操作系统中的TCP/IP等协议栈所提供的是我们最常用的服务，而RTP的实现还是要靠开发者自己。因此从开发的角度来说，RTP的实现和应用层协议的实现没不同，所以可将RTP看成应用层协议。

RTP实现者在发送RTP数据时，需先将数据封装成RTP包，而在接收到RTP数据包，需要将数据从RTP包中提取出来。

2.2. RTP的封装

一个协议的封装是为了满足协议的功能需求的。从前面提出的功能需求，可以推测出RTP封装中应该有同步源和时戳等字段，但更为完整的封装是什么样子呢？请看图 2。

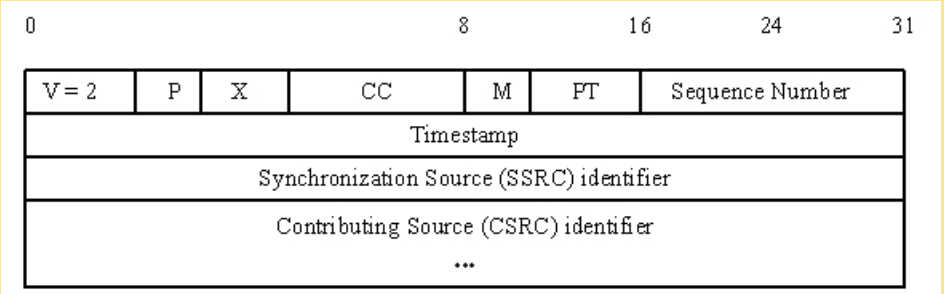


图 2 RTP的头部格式

版本号 (V)：2比特，用来标志使用的RTP版本。

填充位 (P)：1比特，如果该位置位，则该RTP包的尾部就包含附加的填充字节。

- b服务
9. 整合91平台接入的AN
10. How to decode a H.264 frame on hardware decoding

评论排

1. 实战FFmpeg – 编译iOS平台使用的FFr
- (支持arm64的FFmpeg2.6
2. 笔谈OpenGL ES (
3. 谈一谈做iOS播放器库开发所涉及的知

推荐排

1. iOS学习资源

扩展位 (X)：1比特，如果该位置位的话，RTP固定头部后面就跟有一个扩展头部。

CSRC计数器 (CC)：4比特，含有固定头部后面跟着的CSRC的数目。

标记位 (M)：1比特,该位的解释由配置文档 (Profile) 来承担。

载荷类型 (PT)：7比特，标识了RTP载荷的类型。

序列号 (SN)：16比特，发送方在每发送完一个RTP包后就将该域的值增加1，接收方可以由该域检测包的丢失及恢复包序列。序列号的初始值是随机的。

时间戳：32比特，记录了该包中数据的第一个字节的采样时刻。在一次会话开始时，时间戳初始化成一个初始值。即使在没有信号发送时，时间戳的数值也要随时间而不断地增加（时间在流逝嘛）。时间戳是去除抖动和实现同步不可缺少的。

同步源标识符(SSRC)：32比特，同步源就是指RTP包流的来源。在同一个RTP会话中不能有两个相同的SSRC值。该标识符是随机选取的 RFC1889推荐了MD5随机算法。

贡献源列表 (CSRC List)：0~15项，每项32比特，用来标志对一个RTP混合器产生的新包有贡献的所有RTP包的源。由混合器将这些有贡献的SSRC标识符插入表中。SSRC标识符都被列出来，以便接收端能正确指出交谈双方的身份。

2.3. RTCP的封装

RTP需要RTCP为其服务质量提供保证，因此下面介绍一下RTCP的相关知识。

RTCP的主要功能是：服务质量的监视与反馈、媒体间的同步，以及多播组中成员的标识。在RTP会话期间，各参与者周期性地传送RTCP包。RTCP包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料，因此，各参与者可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型。RTP和RTCP配合使用，它们能以有效的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。

从图 1可以看到，RTCP也是用UDP来传送的，但RTCP封装的仅仅是一些控制信息，因而分组很短，所以可以将多个RTCP分组封装在一个UDP包中。RTCP有如下五种分组类型。

类型	缩写表示	用途
200	SR (Sender Report)	发送端报告
201	RR (Receiver Report)	接收端报告
202	SDES (Source Description Items)	源点描述
203	BYE	结束传输
204	APP	特定应用

表 1 RTCP的5种分组类型

上述五种分组的封装大同小异，下面只讲述SR类型，而其它类型请参考RFC3550。

发送端报告分组SR (Sender Report) 用来使发送端以多播方式向所有接收端报告发送情况。SR分组的主要内容有：相应的RTP流的SSRC，RTP流中最新产生的RTP分组的时间戳和NTP，RTP流包含的分组数，RTP流包含的字节数。SR包的封装如图3所示。

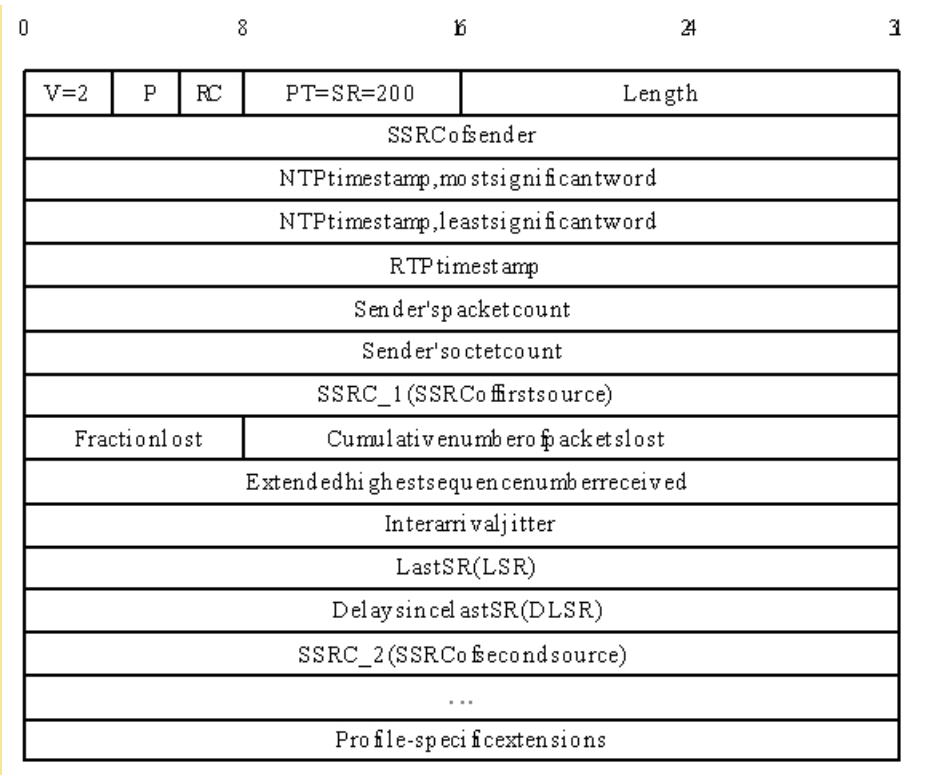


图 3 RTCP头部的格式

- 版本（V）：同RTP包头域。
- 填充（P）：同RTP包头域。
- 接收报告计数器（RC）：5比特，该SR包中的接收报告块的数目，可以为零。
- 包类型（PT）：8比特，SR包是200。
- 长度域（Length）：16比特，其中存放的是该SR包以32比特为单位的总长度减一。
- 同步源（SSRC）：SR包发送者的同步源标识符。与对应RTP包中的SSRC一样。
- NTP Timestamp（Network time protocol）SR包发送时的绝对时间值。NTP的作用是同步不同的RTP媒体流。
- RTP Timestamp：与NTP时间戳对应，与RTP数据包中的RTP时间戳具有相同的单位和随机初始值。
- Sender's packet count：从开始发送包到产生这个SR包这段时间里，发送者发送的RTP数据包的总数。SSRC改变时，这个域清零。
- Sender's octet count：从开始发送包到产生这个SR包这段时间里，发送者发送的净荷数据的总字节数（不包括头部和填充）。发送者改变其SSRC时，这个域要清零。
- 同步源n的SSRC标识符：该报告块中包含的是从该源接收到的包的统计信息。
- 丢失率（Fraction Lost）：表明从上一个SR或RR包发出以来从同步源n(SSRC_n)来的RTP数据包的丢失率。
- 累计的包丢失数目：从开始接收到SSRC_n的包到发送SR,从SSRC_n传过来的RTP数据包的丢失总数。
- 收到的扩展最大序列号：从SSRC_n收到的RTP数据包中最大的序列号，
- 接收抖动（Interarrival jitter）：RTP数据包接受时间的统计方差估计
- 上次SR时间戳（Last SR,LSR）：取最近从SSRC_n收到的SR包中的NTP时间戳的中间32比特。如果目前还没收到SR包，则该域清零。
- 上次SR以来的延时（Delay since last SR,DLSR）：上次从SSRC_n收到SR包到发送本报告的延时。

2.4. RTP的会话过程

当应用程序建立一个RTP会话时，应用程序将确定一对目的传输地址。目的传输地址由一个网络地址和一对端口组成，有两个端口：一个给RTP包，一个给RTCP包，使

得RTP/RTCP数据能够正确发送。RTP数据发向偶数的UDP端口，而对应的控制信号RTCP数据发向相邻的奇数UDP端口（偶数的UDP端口 + 1），这样就构成一个UDP端口对。RTP的发送过程如下，接收过程则相反。

- 1) RTP协议从上层接收流媒体信息码流（如H.263），封装成RTP数据包；RTCP从上层接收控制信息，封装成RTCP控制包。
- 2) RTP将RTP 数据包发往UDP端口对中偶数端口；RTCP将RTCP控制包发往UDP端口对中的接收端口。

第3章. 相关的协议

3.1. 实时流协议RTSP

实时流协议RTSP（Real-Time Streaming Protocol）是IETF提出的协议，对应的RFC文档为RFC2362。

从图 1可以看出，RTSP是一个应用层协议（TCP/IP网络体系中）。它以C/S模式工作，它是一个多媒体播放控制协议，主要用来使用户在播放流媒体时可以像操作本地的影碟机一样进行控制，即可以对流媒体进行暂停/继续、后退和前进等控制。

3.2. 资源预定协议RSVP

资源预定协议RSVP(Resource Reservation Protocol)是IETF提出的协议，对应的RFC文档为RFC2208。

从图 1可以看出，RSVP工作在IP层之上传输层之下，是一个网络控制协议。RSVP通过在路由器上预留一定的带宽，能在一定程度上为流媒体的传输提供服务质量。在某些试验性的系统如网络视频会议工具vic中就集成了RSVP。

第4章. 常见的疑问

4.1. 怎样重组乱序的数据包

可以根据RTP包的序列号来排序。

4.2. 怎样获得数据包的时序

可以根据RTP包的时间戳来获得数据包的时序。

4.3. 声音和图像怎么同步

根据声音流和图像流的相对时间（即RTP包的时间戳），以及它们的绝对时间（即对应的RTCP包中的RTCP），可以实现声音和图像的同步。

4.4. 接收缓冲和播放缓冲的作用

如1.3.1所述，接收缓冲用来排序乱序了的数据包；播放缓冲用来消除播放的抖动，实现等时播放。

第5章. 实现方案

ID	Protocol	Captured contents						
		Account	password	Local telephone number	Opponents Telephone Number	audio	login	logout
36	Rtp					√		

表 2 协议分析要求

表 2给出了协议分析要求。容易看出要获取RTP音频包中的音频信息很容易，直接将RTP包的包头去掉即可。当然，要成功地播放解码获取到的音频流，需要知道其编码，这可从RTP包包头的有效载荷类型字段（PT）获得。

第6章. 参考资料

- [1] RFC文档：RFC3550对应RTP/RTCP，RFC2362对应RTSP，RFC2208对应RSVP
- [2] <http://www.faqs.org/rfcs/>，上面有全面的英文RFC文档

[3] <http://www.cnpat.net/>，有不少协议分析文档，也有中文RFC文档，但质量不是特别高。

二、RTP与RTCP解释.含同步时间戳

RTP协议是real-time transport protocol的缩写，被设计来传输流媒体数据，有着广泛的应用，其它相关介绍自己去看RFC，我不打算讨论这些无聊的概念性的东西。

(1) 了解RTP

可以说，RTP协议不依赖于底层协议，也就是说，它是独立的协议。而一般的，由于UDP包的快速、实时性高的特点，它通常和UDP结合在一起，作为UDP的上层载体数据的形式传播。

```
typedef struct {  
    IN OUT  UINT32  timestamp;  
    IN OUT  BOOL    marker;  
    IN OUT  BYTE    payload;  
  
    OUT  UINT32  sSrc;  
    OUT  UINT16  sequenceNumber;  
    OUT  int     sByte;  
    OUT  int     len;  
} rtpParam;
```

这是一个RTP头，很简单，并没有你想象的那么复杂，对不对？我们来看几个主要的参数，他们也是RTP的灵魂：

(1) payload。payload表示了此RTP包的数据是那种类型的数据，不同的数值表示不同的类型。如0是PCMU，8是723，24是视频263等等。

(2) SSRC，这个东西并不常用，实际上它是一个随即生成的ID，表示了一个RTP连接。在应用的时候，确保这个ID唯一就可以了。

(3) sequence number。也就是**序列号**，它表示了当前包是第几个包。**发送方每发送一个包，就把这个数值加一。接受方可以根据这个数值来重新组合包顺序，判断包是否丢失等操作。**注意：它只是表示了包的先后顺序，它不能表示时间上的任何其它信息。这个请和后面的时间戳比较。

(4) timestamp。时间戳，它的概念稍微有点复杂，我用稍微通俗点的理解去解释它，虽然这样有点不太正确。时间戳顾名思义，它表示了一个数据产生的时间，和我们邮递的邮戳一样，它是个时间标记（至于这个时间干什么用，我后面会详细的说），通常表示RTP数据包中，第一个字节数据产生的时间（至于你是不是这么用就是你写程序的问题了）。

如果你上面理解了，那么我们更进一步：**实际上，时间戳增加一并不是我们通常意义上的过了一个微秒，而是增加了一个采样间隔那么长的时间。举个例子来说。不同的采集有不同的采样频率，比如一般的音频是8K的采样频率，也就是一毫秒采集8次数据，也就是每次采样间隔是1/8MS，而timestamp增加1也就意味着增加了一个采样间隔。也就是过了1/8MS。换个例子，如果令一种编码的采样频率是16K，那么timestamp增加1也就意味着系统过了1/16MS。也就是说，再同一个系统中，对不同编码，虽然使用同一个时钟，但timestamp的增长速度是不同的，在这个例子中，采样频率是16K的编码要比8K的快两倍，请记住这个区别。**

(2) 了解RTCP

RTCP协议是real-time transport control protocol的缩写，被设计来做RTP的控制，这个相对来说大家不怎么关心，我只介绍下它基本的东西。

RTCP实际上是RTP传输情况的反馈，通俗的说，它告诉另外一方，在一端时间内（5秒），它发送多少数据包给对方，接收到了多少对方的包。

另外，在RTCP中，还有两个比较重要的东西，**一个64位的绝对时间戳和一个32位的相对时间戳。**64 位时间戳也叫NTP时间戳，它的前32位是从1900 年1 月1 日0 时开始到现在的以秒为单位的整数部分，后32 位是此时间的小数部，因此，它可以肯定的表示了数据发送出去的绝对时间。32位的时间戳和RTP中的时间戳是一样的，没有任何区别。

(3) 大家感兴趣的时间戳的使用和同步的一些话题。

1、SSRC的作用。

SSRC相当于一个RTP传输session的ID，就象每个人都有个名字一样，每一个RTP传输也都有一个名字。这个数字是随机产生，并且要保证唯一。当RTP session改变（如IP等）时，这个ID也要改变。

2、序列号字段是否可以作为流内的同步标志？

我在上面已经说过，**序列号**只表示了包发出的先后顺序，它表示不了任何时间上的其它概念，所有严格的说，序列号并不能作为流内的同步标志。但是，由于一般来说，包的发送时间都会有严格限制，比如音频包是每秒种发送30个数据包，也就是说，每个包间隔1000/30MS，而这个时间就可以作为一个同步时间来播放。也就是说，按照序列号，每1000/30MS间隔播放一个数据包，这样也能保证同步，但是这时候请考虑丢包问题。

3、绝对时间戳和相对时间戳在进行同步处理时有什么不同

当我们取得绝对时间后，我们就可以根据这个绝对时间来播放这些数据包。这个绝对时间，加上我们需要的延时（用于数据传输，解码等等的时间）就是我们的播放时间，这样我们可以保证时间的严格同步（相当于把对方的动作延时一段时间后原原本本的再现出来）。**目前，在RTP中，能得到这个绝对时间的办法只有RTCP。**

对于相对时间戳，我们更关心的是两个时间戳之间的时间间隔，依靠这个时间间隔，来确定两个包的播放时间间隔。

4、单个媒体内的同步和不同媒体流之间的同步在处理方式上有什么不同

应该说，不同媒体之间同步比单媒体同步要复杂得多，除了要保证本身的播放要和时间同步外，还要保证两个或多个媒体间同步（比如音视频的同步）。这种不同更关心的两个时间戳时间的换算统一，前面我已经说过，不同编码有不同的采样频率，那么时间戳的增长速度就不同。**另外，两个时间戳也需要有一个标准时间来表示时间戳的同步。最简单的方法是两个媒体的第一个时间戳相同，表示两个流的采集开始时间统一。**另外还可以通过RTCP来做不同流之间的同步，这在下个问题中会提到。

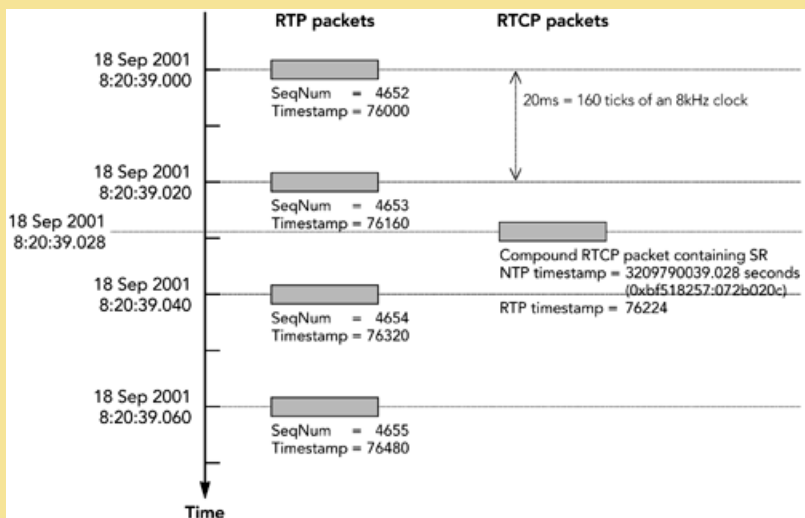
5、时间戳字段如何用于作为流间同步标识

在RTP协议中，我们取得时间戳的方法有两个：一个是RTP包中的时间戳，另外一个为RTCP包中的绝对时间戳和相对时间戳。绝对时间戳的概念上面我已经说了，它可以表示系统的绝对时间。而RTCP包中的相对时间就是RTP包中的时间。根据这两个时间，不同流都可以纠正自己播放时间和真正时间的偏差以达到和绝对时间同步的目的。**反过来说，如果我们没有办法拿到这个绝对时间，只有RTP包中的相对时间，那我们则需要确定两个流在某一时间点的时间戳的数值。通俗的说，就是在某个时间点，流A的timestamp是多少，B是多少，然后根据这个时间两个流播放的延时时间，以达到同步的目的。实现这个目的最简单的办法是在两个流开始的时候，使用相同的stamp，拿音视频来说，在某一绝对时刻，采集相应的数据，并打上相同的时间戳，以后的播放，都以这个时间做基准时间，以保证同步**

三、RTP时间戳相关

通过RTSP建立好会话之后，就可以开始传输RTP数据和RTCP SR包了（用来同步音视频）。

这两者涉及到很重要的问题：时间戳。下面是《rtp_audio_and_video_for_the_internet》上的一个时间图。



TimeStamp的初始值是随即生成的，然后每一帧数据固定增加一个增量，客户端在接收到数据时，根据这个时间戳就能以正确的时间恢复（其中被分包的视频帧是没有时有一个NTP时间，这是距1900年1月1日的秒数，允许每个系统存在差异，只要同一个系统不同流的该值频为例，若其帧率为30帧，则每一帧的时间戳增量为90000/30=3000；RTCP的SR包的时间戳也可以以RTP包发送时间）*单位时间增量，其中单位时间增量=90000*1000000/(2^32)，因为SR包中的微秒要做"/(2^32)"这样一个转化。

RFC中说时间戳增量需要满足线性增长，实际上没必要严格按照诸如3000增量来增长，我是按照实际的帧的时间间隔来打的这个时间戳：

时间戳 = 上一次时间戳 + 采样频率（典型值为90000）* 0.000001 * 两帧时间差（单位毫秒）来计算

时间戳（timestamp） 32比特 时间戳反映了RTP数据包中第一个字节的采样时间。（采样时钟必须来源于一个及时的单调、线性递增时钟，以便允许同步和去除网络引起的数据包抖动。该时钟的分辨率必须满足理想的同步精度和测量数据包到来时的抖动的需要（一种典型的时钟分辨率不满足情况是每个视频帧仅一个时钟周期）时钟 频率依赖于负载数据的格式，并在描述文件（profile）中或者是在负载格式描述中（payload format specification）进行静态描述。也可以通过非RTP方法（non-RTP means）对负载格式动态描述。

如果RTP包是周期性产生的，那么将使用由采样时钟决定的名义上的采样时刻，而不是读取系统时间。例如，对一个固定速率的音频，采样时钟（时间戳时钟）将 在每个周期内增加1。如果一个音频从输入设备中读取含有160个采样周期的块，那么对每个块，时间戳的值增加160，而不考虑该块是否用一个包传递或是被 丢弃。

时间戳的初始值应当是随机的，就像序号一样。几个连续的RTP包如果（逻辑上）是同时产生的，如：属于同一个视频帧的RTP包，将有相同的序列号。如果数据并不是以它采样的顺序进行传输，那么连续的RTP包可以包含不是单调递增（或递减）的时间戳（RTP包的序列号仍然是单调变化的）。

根据一些文章我自己推敲了一下几个概念如下：

时间戳单位：时间戳计算的单位不为秒之类的单位，而是由采样频率所代替的单位，这样做的目的就是为了让时间戳单位更为精准。比如说一个音频的采样频率为8000HZ，那么我们可以把时间戳单位设为1/8000。

时间戳增量：相邻两个RTP包之间的时间差（以时间戳单位为基准）。

如何设定时间戳之间的增量呢？

按照刚才时间戳单位来看，1秒钟按照时间戳单位就是8000，那么一秒钟如果可以播放20帧，也就是发送30帧（帧率），那么可以求出相邻两帧之间的时间差，也就是时间戳增量，那么显而易见是用8000/20，那么这个时间戳增量就为400。

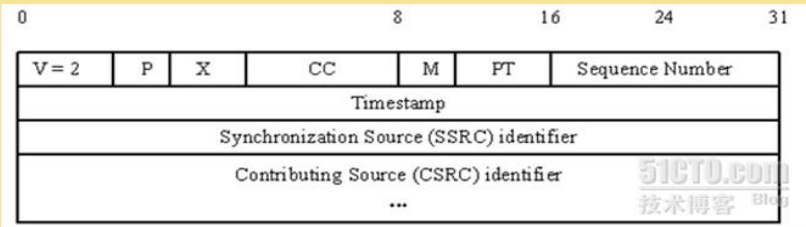
网上大多数列举的一个例子是： 例如MPEG，每帧20ms，采样频率8000Hz，设定时间戳单位1/8000，而每个包之间就是160的增量

这里又该如何理解呢？可以轻易地看出增量是直接8000与20ms相乘的结果，我们可以知道这里两帧之间的时间为20ms，也就是0.02s，这个单位是以秒来衡量的，那么我们要用时间戳单位来表示那么就是8000*0.02=160.所以时间戳增量为160。

还有一点为什么一般都用90000作为视频采样频率呢？

90k是用于视频同步的时间尺度(TimeScale),就是每秒90k个时钟tick。为什么采用90k呢？目前视频的帧速率主要有25fps、29.97fps、30fps等，而90k刚好是它们的倍数，所以就采用了90k。

RTP协议包头的格式：



10~16 Bit为PT域，指的就是负载类型（PayLoad），负载类型定义了RTP负载的格式，协议原文说该域由具体应用决定其解释。

目前，负载类型主要用来告诉接收端（或者播放器）传输的是哪种类型的媒体（例如G.729, H.264, MPEG-4等），这样接收端（或者播放器）才知道了数据流的格式，才会调用适当的编解码器去解码或者播放，这就是负载类型的主要作用。

时间戳单位：时间戳计算的单位不是秒之类的单位，而是由采样频率所代替的单位，这样做的目的就是为了是时间戳单位更为精准。比如说一个音频的采样频率为8000Hz，那么我们可以把时间戳单位设为 $1 / 8000$ 。

时间戳增量：相邻两个RTP包之间的时间差（以时间戳单位为基准）。

采样频率：每秒钟抽取样本的次数，例如音频的采样率一般为8000Hz

帧率：每秒传输或者显示帧数，例如25f/s

再看看RTP时间戳课本中的定义：

RTP包头的第2个32Bit即为RTP包的时间戳，Time Stamp，占32位。

时间戳反映了RTP分组中的数据的一个字节的采样时刻。在一次会话开始时的时间戳初值也是随机选择的。即使是没有信号发送时，时间戳的数值也要随时间不断的增加。接收端使用时间戳可准确知道应当在什么时间还原哪一个数据块，从而消除传输中的抖动。时间戳还可用来使视频应用中声音和图像同步。

在RTP协议中并没有规定时间戳的粒度，这取决于有效载荷的类型。因此RTP的时间戳又称为媒体时间戳，以强调这种时间戳的粒度取决于信号的类型。例如，对于8kHz采样的语音信号，若每隔20ms构成一个数据块，则一个数据块中包含有160个样本（ $0.02 \times 8000 = 160$ ）。因此每发送一个RTP分组，其时间戳的值就增加160。

官方的解释看懂没？没看懂？没关系，我刚开始也没看懂，那就听我的解释吧。

首先，时间戳就是一个值，用来反映某个数据块的产生（采集）时间点的，后采集的数据块的时间戳肯定是大于先采集的数据块的。有了这样一个时间戳，就可以标记数据块的先后顺序。

第二，在实时流传输中，数据采集后立刻传递到RTP模块进行发送，那么，其实，数据块的采集时间戳就直接作为RTP包的时间戳。

第三，如果用RTP来传输固定的文件，则这个时间戳就是读文件的时间点，依次递增。这个不再我们当前的讨论范围内，暂时不考虑。

第四，时间戳的单位采用的是采样频率的倒数，例如采样频率为8000Hz时，时间戳的单位为 $1 / 8000$ ，在JrtpLib库中，有设置时间戳单位的函数接口，而ORTP库中根据负载类型直接给定了时间戳的单位（音频负载 $1/8000$ ，视频负载 $1/90000$ ）

第五，时间戳增量是指两个RTP包之间的时间间隔，详细点说，就是发送第二个RTP包相距发送第一个RTP包时的时间间隔（单位是时间戳单位）。

如果采样频率为90000Hz，则由上面讨论可知，时间戳单位为 $1/90000$ ，我们就假设1s钟被划分了90000个时间块，那么，如果每秒发送25帧，那么，每一个帧的发送占多少个时间块呢？当然是 $90000/25 = 3600$ 。因此，我们根据定义“时间戳增量是发送第二个RTP包相距发送第一个RTP包时的时间间隔”，故时间戳增量应该为3600。

【补充】：最近思考了一下，又有了新的体会和解释，可能对大家更容易地去理解这个时间戳增量会有所帮助，补充在下面吧：

其实，网络发送重点关注的是流量的平衡，即均匀地利用网络带宽，为了实现这一点，需要满足：数据采集的速率与数据网络传输的速率尽量保持一致。时间戳增量的设置影响的是RTP包的网络传输的速率，时间戳增量越小，发送速度越快。

下面再进一步解释一下时间戳增量是怎么计算出来的：

对于PAL制式的视频而言，每秒摄像头会采集 25 帧 数据，那么，每采集到 1 帧 耗时 $1/25$ s，如果我们设计为1个RTP包只包含1帧数据，并且一次发送1帧，那么，要想网络流量均匀，则时间戳增量应该设计为 $1/25$ s。而在一般的RTP协议的实现中，时间戳单位不是 秒（s），而约定为采样频率的倒数，由于一般视频的采样频率是 90000，故时间戳单位为 $1/90000$ s，因此，实际的时间戳增量 = 时间戳增量（ $1/25$ s）/ 时间戳单位（ $1/90000$ s）= 3600

分类: [FFmpeg](#), [iOS开发生涯](#)

好文要顶

关注我

收藏该文





[min2011](#)
[关注 - 14](#)
[粉丝 - 12](#)

[+加关注](#)

0

推荐

0


反对

(请您对文章做出评价)

« 上一篇: [iPhone屏幕尺寸、分辨率及适配](#)
» 下一篇: [ffmpeg 如何音视频同步](#)

posted @ 2015-05-18 14:21 min2011 阅读(172) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】融云即时通讯云 – 豆果美食、Faceu等亿级APP都在用
- 【推荐】报表开发别头大！类Excel 复杂报表开发实例，即学即用
- 【推荐】阿里云万网域名：.xin .com将推出重磅优惠



- 最新IT新闻：
- 微软宣布开源Edge浏览器的WebGL GLSL转编译器
 - 百度与21家银行签署20亿美元贷款协议
 - jQuery 3.0正式发布：近期JavaScript新闻汇总
 - 面向学校的微软Minecraft教育版进入beta阶段
 - 51Talk敲响开市钟 开盘价19.50美元市值57亿人民币
- » 更多新闻...

90%的开发者都在用极光推送

不只是稳定

- 最新知识库文章：
- 高效编程之道：好好休息
 - 快速学习者的高效学习策略
 - 一个前端的自我修养
 - 架构漫谈（九）：理清技术、业务和架构的关系
 - 架构漫谈（八）：从架构的角度看如何写好代码
- » 更多知识库文章...