**田野2018/11/19**

实时传送协议（Real-time Transport Protocol或简写RTP，也可以写成RTTP）是一个网络传输协议，它是由[IETF](http://baike.baidu.com/view/155093.htm)的多媒体传输工作小组1996年在RFC 1889中公布的。

RTP协议详细说明了在互联网上传递音频和视频的标准数据包格式。它一开始被设计为一个多播协议，但后来被用在很多单播应用中。RTP协议常用于流媒体系统（配合RTCP协议或者[RTSP协议](http://www.cnblogs.com/qingquan/archive/2011/07/14/2106834.html)）。因为RTP自身具有Time stamp所以在ffmpeg 中被用做一种formate.

**RTP协议格式：**

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|V=2|P|X| CC |M| PT | sequence number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| timestamp |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| synchronization source (SSRC) identifier |

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

| contributing source (CSRC) identifiers |

| .... |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

上图引自[rfc3550](http://www.ietf.org/rfc/rfc3550.txt)，由上图中可知道RTP报文由两个部分构成--RTP报头和RTP的负载：

RTP报文由两部分组成：报头和有效载荷。RTP报头格式如图6.7所示，其中：

1.V：RTP协议的版本号，占2位，当前协议版本号为2。

2. P：填充标志，占1位，如果P=1，则在该报文的尾部填充一个或多个额外的八位组，它们不是有效载荷的一部分。

3. X：扩展标志，占1位，如果X=1，则在RTP报头后跟有一个扩展报头。

4.  CC：CSRC计数器，占4位，指示CSRC 标识符的个数。

5. M: 标记，占1位，不同的有效载荷有不同的含义，对于视频，标记一帧的结束；对于音频，标记会话的开始。

6. PT: 有效载荷类型，占7位，用于说明RTP报文中有效载荷的类型，如GSM音频、JPEM图像等,在流媒体中大部分是用来区分音频流和视频流的，这样便于客户端进行解析。

7. 序列号：占16位，用于标识发送者所发送的RTP报文的序列号，每发送一个报文，序列号增1。这个字段当下层的承载协议用UDP的时候，网络状况不好的时候可以用来检查丢包。同时出现网络抖动的情况可以用来对数据进行重新排序，在helix服务器中这个字段是从0开始的，同时音频包和视频包的sequence是分别记数的。

8. 时戳(Timestamp)：占32位，时戳反映了该RTP报文的第一个八位组的采样时刻。接收者使用时戳来计算延迟和延迟抖动，并进行同步控制。

9. 同步信源(SSRC)标识符：占32位，用于标识同步信源。该标识符是随机选择的，参加同一视频会议的两个同步信源不能有相同的SSRC。

10. 特约信源(CSRC)标识符：每个CSRC标识符占32位，可以有0～15个。每个CSRC标识了包含在该RTP报文有效载荷中的所有特约信源。

如果扩展标志被置位则说明紧跟在报头后面是一个头扩展，其格式如下：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| defined by profile | length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

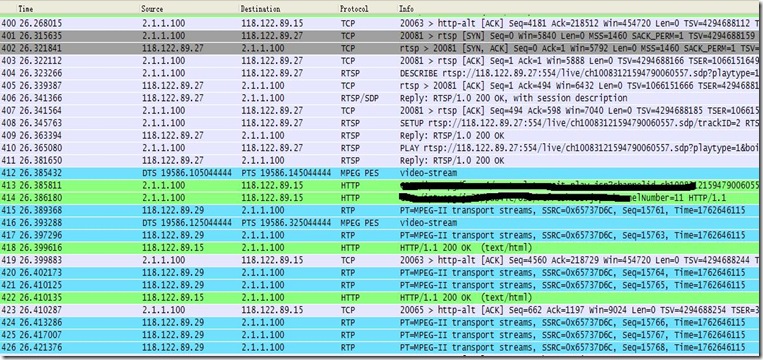
| header extension |

| .... |

**RTP协议的用途：**

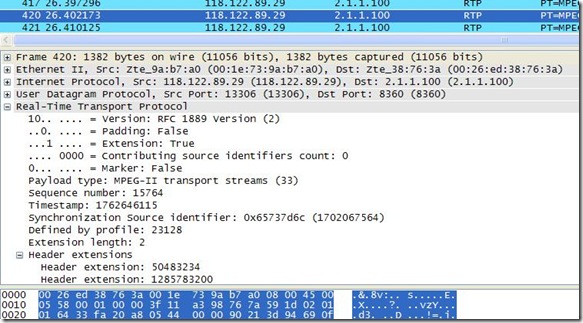
概述中已经基本阐述了ＲＴＰ协议的用途了，其主要用于在互联网上传递音频和视频的标准数据包。在当前三网融合中ＲＴＰ可以用来承载TS流，进行电视媒体数据的传播。ＲＴＰ可以用来传送像TS流这种自身已经具有ｆｏｒｍａｔｅ的媒体流，同时也可以用来承载ＡＶＣ，ＡＡＣ等去除了fromate的媒体流，这时rtp协议可被看做为一种formate，这种形式最少常见于helix 流媒体服务器的rtp流。其控制流由[RTSP协议](http://www.cnblogs.com/qingquan/archive/2011/07/14/2106834.html)来提供。

**RTP协议的使用：**

[](http://images.cnblogs.com/cnblogs_com/qingquan/201107/201107282259116288.jpg)

ＲＴＰ的使用实例之一如上图：

上面是某省IPTV２.０早期的一个数据包的情况。从包中可以看出RTP是怎么和RTSP配合一起使用的。从包402到411为RTSP的协商过程，RTSP在PLAYer命令后数据包就到来。紧跟其后412包就是一个mpeg 的PES包，它是有由rtp来承载的TS来形成。从在420包中就可以更加清析的看出这个RTP流的情况。其PT即payload type为mpeg2 transport streams 也就是ts流，其SSRC为：0x65737D6c，其Seq号为15764，从中也可以看出对于一个RTP流其SEQ号可以开始于一个随机的数值，但是肯定是逐包递增的。下图为420包的展开图：

[](http://images.cnblogs.com/cnblogs_com/qingquan/201107/20110728225922738.jpg)

从中可以看出承载RTP的为UDP的数据流这个包中有x标志位为1则说明其有 header extensions.其header extensions为最下面。extension 的 profile为23128,长度为：2内容如上图最后两部分。

**自己动手写RTP服务器——关于RTP协议**

本文会带领着你一步步动手实现一个简单的RTP传输服务器，旨在了解RTP流媒体传输协议以及一些关于多媒体编解码的知识。

**关于RTP协议的必备知识**

要动手实现一个协议，当然首先需要阅读该协议的文档。RTP协议的文档，有rfc1889、rfc1890、rfc3550，其中rfc3550是现在的版本，另外两个是过期版。这个协议可以在ietf的官网找到：http://tools.ietf.org/html/rfc3550

**RTP packet**

RTP是基于UDP协议的，RTP服务器会通过UDP协议，通常每次会发送一个RTP packet。客户端通过解析RTP packet，读取其中的数据然后进行播放了。

RTP packet的结构如下：

1. RTP Header：RTP 包的头部
2. contributing sources：个数为0-n个，所以可以为空。具体定义参考rfc3550
3. RTP payload：即RTP要传输的数据

**RTP Header**

这是RTP流的头部，在网上搜索RTP格式，就会搜到很多文章介绍这个头部的定义。我们这里参考rfc3550的定义，在5.1节(http://tools.ietf.org/html/rfc3550#section-5.1)。

    0                   1                   2                   3  
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
   |V=2|P|X|  CC   |M|     PT      |       sequence number         |  
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
   |                           timestamp                           |  
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  
   |           synchronization source (SSRC) identifier            |  
   +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+  
   |            contributing source (CSRC) identifiers             |  
   |                             ....                              |  
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

每行是32 bits，由此可以直观看到每个表示部分所占的位数。简单介绍一下：

V(version)：2 bits，RTP的版本，这里统一为2

P(padding)：1 bit，如果置1，在packet的末尾被填充，填充有时是方便一些针对固定长度的算法的封装

X(extension)：1 bit，如果置1，在RTP Header会跟着一个header extension

CC(CSRC count): 4 bits，表示头部后contributing sources的个数

M(marker): 1 bit，具体这位的定义会在一个profile里

PT(playload type): 7 bits，表示所传输的多媒体的类型，对应的编号在另一份文档rfc3551中有列出(http://tools.ietf.org/html/rfc3551)

sequence number: 16 bits，每个RTP packet的sequence number会自动加一，以便接收端检测丢包情况

timestamp: 32 bits，时间戳

SSRC: 32 bits，同步源的id，没两个同步源的id不能相同

CSRC: 上文说到，个数由CC指定，范围是0-15

以上的一些概念是一些要实现RTP服务器所必备的知识。介绍的非常简略，详细的定义还是要参考rfc3550原文。

**动手实践**

我们既然已经知道了RTP packet的结构，那么我们以前用到的RTP流是否也是这样的结构呢？如何验证呢？接下来，我们就一步步验证RTP流的结构。

我们知道RTP是基于UDP协议的，那么我们就先做一个简单的UDP接受端，看看我们可以从RTP服务器接受到什么信息。要实现这个接受端，你需要有一定的网络编程经验，至于具体到操作系统、编程环境、开发语言等都不限制。为了简单，我这里用python给出一个小小的例子程序。

**[python]** [view plaincopy](http://blog.csdn.net/baby313/article/details/7353605)

1. **import** socket
3. # Build a socket to receive data from RTP server.
4. # Here we use SOCK\_DGRAM, because RTP is on UDP.
5. sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)
6. sock.bind(("localhost", 6666))
8. **for** i **in** range(5):
9. # We just get 16 bytes to analyze the RTP Header.
10. buf = sock.recv(16)
12. # Output the result in octal.
13. **for** c **in** buf:
14. **print** "%x" % ord(c),
15. **print**
17. sock.close()

这就是接受程序啦，非常短小，而且有简单注释，这里就不解释了。

接受端已经做好了，那么去哪里找RTP服务器作发送端呢？你可以用一些搭建流媒体服务器的工具，我这里选用的是强大的VLC。关于VLC搭建流媒体服务器的方法，请参考我前面的文章[基于移动平台的多媒体框架——用VLC搭建简单的流媒体服务器](http://blog.csdn.net/baby313/article/details/6966466)。这里需要注意几个配置的地方，一是选择Destination的时候要选择RTP而不要选择RTSP，然后地址可以填写本机ip地址或直接写localhost，端口号填写的要和接受端一致，这里是6666。配置好之后的string应该类似于：

:sout=#rtp{dst=localhost,port=6666,mux=ts} :no-sout-rtp-sap :no-sout-standard-sap :ttl=1

服务端配置完成之后，开始Stream。这时打开接受端，就会接受到一些数据，我接收到的数据开头是：

80 a1 20 43 8c cf 76 3c 93 59 d 74 47 0 44 10  
80 a1 20 44 8c cf 79 4b 93 59 d 74 47 40 42 36  
80 a1 20 45 8c cf 7d 36 93 59 d 74 47 0 44 1a  
80 a1 20 46 8c cf 81 21 93 59 d 74 47 40 45 1a  
80 a1 20 47 8c cf 85 c 93 59 d 74 47 0 45 1b  
这是十六进制的表示。我们依照上面的Header的格式对其进行解读：  
第一个byte 80 表示：

V(version)=2

P(padding)=0

X(extension)=0

CC(CSRC count)=0

第二个byte a1 表示：

M(marker)=1

PT(playload type)=33(对照rfc3551可以发现，33表示MP2T AV，正是我们用VLC Stream的格式类型)

后面的2bytes的sequence number我们可以直观的看出是在加一，4bytes的timestamp也是在不断递增的。再之后的93 59 d 74就是SSRC id了，由于CC为0，所以没有CCRC。再之后的几位都是RTP所要传输的数据了。

**总结**

对RTP协议的熟悉是实现它的基础。这里我只是做一个简单的介绍，需要详细了解，读官方的文档是必不可少的步骤。

通过写一个小程序打印出RTP流中具体的数据，并没有对实现RTP服务器有直接帮助。但是可以让你对协议本身以及编程环境更加熟悉，也方便了以后实现过程中进行调试。不论你在什么环境用什么语言实现，都强烈建议写一个这样的小程序。

**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 (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。

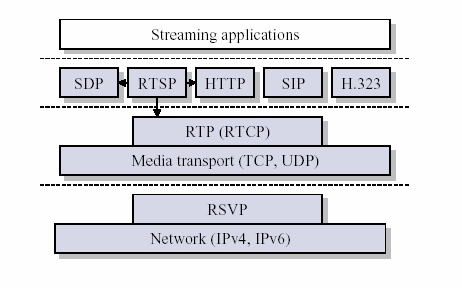


图 1 流媒体体系结构

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。

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。

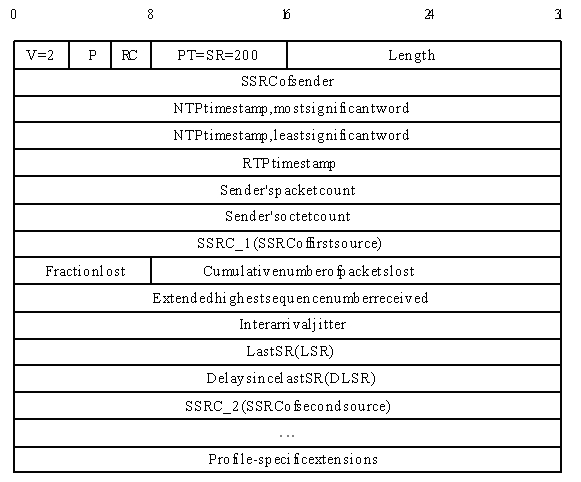


图 3 RTCP头部的格式

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.cnpaf.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。

实时传输协议RTP（Real-time Transport Protocol）是一个[网络传输协议](http://baike.baidu.com/view/16807.htm)，它是由IETF的多媒体传输工作小组1996年在RFC 1889中公布的，后在RFC3550中进行更新。

[**国际电信联盟**](http://baike.baidu.com/view/26606.htm)ITU-T也发布了自己的RTP文档，作为H.225.0，但是后来当IETF发布了关于它的稳定的标准RFC后就被取消了。它作为因特网标准在RFC 3550（该文档的旧版本是RFC 1889）有详细说明。RFC 3551（STD 65，旧版本是RFC 1890）详细描述了使用最小控制的音频和视频会议。

RTP协议详细说明了在互联网上传递音频和视频的标准[数据包](http://baike.baidu.com/view/25880.htm)格式。它一开始被设计为一个[多播](http://baike.baidu.com/view/378050.htm)协议，但后来被用在很多[单播](http://baike.baidu.com/view/492261.htm)应用中。RTP协议常用于[流媒体](http://baike.baidu.com/view/794.htm)系统（配合RTSP协议），视频会议和一键通（Push to Talk）系统（配合[H.323](http://baike.baidu.com/view/76998.htm)或SIP），使它成为IP电话产业的技术基础。RTP协议和RTP控制协议RTCP一起使用，而且它是建立在[用户数据报协议](http://baike.baidu.com/view/468464.htm)上的。

**中文名**

实时传输协议

**外文名**

Real-time Transport Protocol

**简    称**

RTP

**类    型**

概念

目录

1[特征](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin" \l "1)

2[报文格式](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin#2)

3[RTCP概要](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin#3)

4[封包结构](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin#4)

1特征[编辑](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin)

实时传输协议（RTP）为数据提供了具有实时特征的端对端传送服务，如在[组播](http://baike.baidu.com/view/492256.htm)或[单播](http://baike.baidu.com/view/492261.htm)网络服务下的交互式视频音频或模拟数据。应用程序通常在 UDP 上运行 RTP 以便使用其多路结点和校验服务；这两种协议都提供了[传输层](http://baike.baidu.com/view/239605.htm)协议的功能。但是 RTP 可以与其它适合的底层网络或传输协议一起使用。如果底层网络提供[组播](http://baike.baidu.com/view/492256.htm)方式，那么 RTP 可以使用该组播表传输数据到多个目的地。

RTP 本身并没有提供按时发送机制或其它服务质量（QoS）保证，它依赖于低层服务去实现这一过程。 RTP 并不保证传送或防止无序传送，也不确定底层网络的可靠性。 RTP 实行有序传送， RTP 中的[序列号](http://baike.baidu.com/view/53324.htm)允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如：在视频解码中，就不需要顺序解码。

RTP 由两个紧密链接部分组成：

RTP ― 传送具有实时属性的数据；

2报文格式[编辑](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin)

RTP报文由两部分组成：报头和有效载荷。RTP报头格式如图所示，其中：

l V：RTP协议的版本号，占2位，当前协议版本号为2。

l P：填充标志，占1位，如果P=1，则在该报文的尾部填充一个或多个额外的八位组，它们不是有效载荷的一部分。

l X：扩展标志，占1位，如果X=1，则在RTP报头后跟有一个扩展报头。

l CC：CSRC计数器，占4位，指示CSRC 标识符的个数。

l M: 标记，占1位，不同的有效载荷有不同的含义，对于视频，标记一帧的结束；对于音频，标记会话的开始。

l 同步信源(SSRC)标识符：占32位，用于标识同步信源。该标识符是随机选择的，参加同一视频会议的两个同步信源不能有相同的SSRC。

l 特约信源(CSRC)标识符：每个CSRC标识符占32位，可以有0～15个。每个CSRC标识了包含在该RTP报文有效载荷中的所有特约信源。

l PT: 有效载荷类型，占7位，用于说明RTP报文中有效载荷的类型，如GSM音频、JPEM图像等。

l 序列号：占16位，用于标识发送者所发送的RTP报文的序列号，每发送一个报文，序列号增1。接收者通过序列号来检测报文丢失情况，重新排序报文，恢复数据。

l 时戳(Timestamp)：占32位，时戳反映了该RTP报文的第一个八位组的采样时刻。接收者使用时戳来计算延迟和延迟抖动，并进行同步控制。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| V | P | X | CC | M | PT | 序列号 |
| 时戳 | | | | | | |
| 同步信源(SSRC)标识符 | | | | | | |
| 特约信源(CSRC)标识符 | | | | | | |
| ··· | | | | | | |

RTP报头格式

这里的同步信源是指产生媒体流的信源，它通过RTP报头中的一个32位数字SSRC标识符来标识，而不依赖于网络地址，接收者将根据SSRC标识符来区分不同的信源，进行RTP报文的分组。特约信源是指当混合器接收到一个或多个同步信源的RTP报文后，经过混合处理产生一个新的组合RTP报文，并把混合器作为组合RTP报文的SSRC，而将原来所有的SSRC都作为CSRC传送给接收者，使接收者知道组成组合报文的各个SSRC。

3RTCP概要[编辑](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin)

RTCP 控制协议（RTCP） ― 监控服务质量并传送正在进行的会话参与者的相关信息。RTCP 第二方面的功能对于“松散受控”会话是足够的，也就是说，在没有明确的成员控制和组织的情况下，它并不非得用来支持一个应用程序的所有控制通信请求。

4封包结构[编辑](http://baike.baidu.com/view/1268656.htm?from_id=8974125&type=syn&fromtitle=RTP&fr=aladdin)

Ver.（2位元）是目前协定的版本号码，目前版号是2。P（1位元）是用于RTP封包（packet）结束点的预留空间，视封包是否需要多余的填塞空间。X（1位元）是否在使用延伸空间于封包之中。。CC（4位元）包含了CSRC数目用于修正标头（fixedheader）。M（onebit）是用于应用等级以及其原型（profile）的定义。如果不为零表示目前的资料有特别的程式解译。PT（7bits）是指payload的格式并决定将如何去由应用程式加以解译。SSRC是同步化来源。