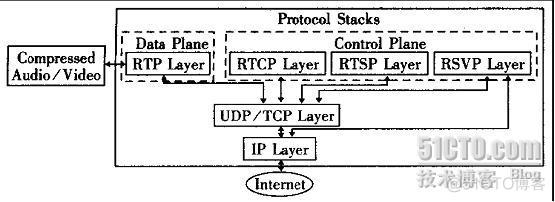
## 一、流媒体

流媒体技术是网络技术和多媒体技术发展到一定阶段的产物。术语流媒体既可以指在网上传输连续时基媒体的流式技术，也可以指使用流式技术的连续时基媒体本身。在网上传输音频、视频等多媒体信息目前主要有两种方式：下载和流式传输。采用下载方式，用户需要先下载整个媒体文件，然后才能进行播放。由于网络带宽的限制，下载常常要花很长时间,所以这种处理方式延迟很大。而流媒体实现的关键技术是流式传输。传输之前首先对多媒体进行预处理(降低质量和高效压缩) ,然后使用缓存系统来保证数据连续正确地进行传输。使用流式传输方式，用户不必像采用下载方式那样要等到整个文件全部下载完毕，而是只需经过几秒到几十秒的启动延时即可在客户端进行播放和观看。此时媒体文件的剩余部分将在后台继续下载。与单纯的下载方式相比，这种对多媒体文件边下载边播放的流式传输方式不仅使启动延时大幅度地缩短，而且对系统缓存容量的需求也大大降低。使用流式传输的另一个好处是使传输那些事先不知道或无法知道大小的媒体数据(如网上直播、视频会议等) 成为可能。

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 的QuickTime 以及Microsoft 的Advanced Streaming Format (ASF) 。

多媒体应用的一个显著特点是数据量大,并且许多应用对实时性要求比较高。传统的TCP 协议是一个面向连接的协议，它的重传机制和拥塞控制机制都是不适用于实时多媒体传输的。RTP 是一个应用型的传输层协议,它并不提供任何传输可靠性的保证和流量的拥塞控制机制。RTP 位于UDP(User Datagram Protocol) 之上。UDP 虽然没有TCP 那么可靠,并且无法保证实时业务的服务质量,需要RTCP 实时监控数据传输和服务质量。但是，由于UDP 的传输时延低于TCP ，能与音频和视频很好地配合。因此，在实际应用中，RTP/ RTCP/ UDP 用于音频/ 视频媒体，而TCP 用于数据和控制信令的传输。目前，支持流媒体传输的协议主要有实时传输协议RTP( Real-Time Transport Protocol) 、实时传输控制协议RTCP(Real-Time Transport Control Protocol) 和实时流协议RTSP(Real-Time Streaming Protocol) 等。



## 二、RTP概述

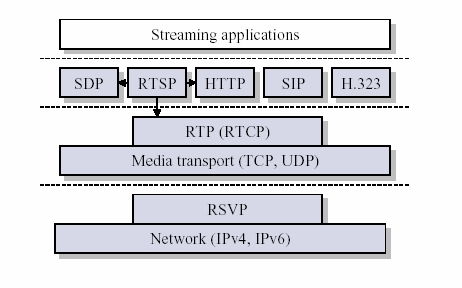
RTP全名是Real-time Transport Protocol（实时传输协议）。它是IETF提出的一个标准，对应的RFC文档为RFC3550。RFC3550不仅定义了RTP，而且定义了配套的相关协议RTCP（Real-time Transport Control Protocol，即实时传输控制协议）。RTP用来为IP网上的语音、图像、传真等多种需要实时传输的多媒体数据提供端到端的实时传输服务。RTP为Internet上端到端的实时传输提供时间信息和流同步，但并不保证服务质量，服务质量由RTCP来提供。

RTP是一种运行在传输层的协议，通常基于 UDP 协议，但也支持 TCP 协议。

RTP定义了两种报文：RTP报文和RTCP报文，RTP报文用于传送媒体数据（如音频和视频），它由 RTP报头和数据两部分组成，RTP数据部分称为有效载荷(payload)；RTCP报文用于传送控制信息，以实现协议控制功能。RTP报文和RTCP 报文将作为下层协议的数据单元进行传输。

如果使用UDP，则RTP报文和RTCP报文分别使用两个相邻的UDP端口，RTP报文使用低端口，RTCP报文使用高端口（偶数端口是分配给RTP进行裸码流数据传输的，奇数端口则是分配给RTCP进行传输控制的）。

如果使用其它的下层协议，RTP报文和RTCP报文可以合并，放在一个数据单元中一起传送，控制信息在前，媒体数据在后。通常，RTP是由应用程序实现的。



在发送端，上层应用程序以分组形式将编码后的媒体数据传给RTP通信模块，作为RTP报文的有效载荷，RTP通信模块将根据上层应用提供的参数在有效载荷前添加RTP报头，形成 RTP报文，通过Socket接口选择UDP协议发送出去。

在接收端，RTP通信模块通过Socket接口接收到RTP报文后，将RTP报头分离出来作相应处理，再将RTP报文的有效载荷作为数据分组传递给上层应用。

考虑到在Internet这种复杂的环境中举行视频会议，RTP定义了两种中间系统：混合器(Mixer)和转换器(Translator)。

在Internet上举行视频会议时，可能有少数参加者通过低速链路与使用高速网络的多数参加者相连接。为了不强制所有会议参加者都使用低带宽和低质量的数据编码，RTP允许在低带宽区域附近使用混合器作为RTP级中继器。混合器从一个或多个信源接收RTP 报文，对到达的数据报文进行重新同步和重新组合，这些重组的数据流被混合成一个数据流，将数据编码转化为在低带宽上可用的类型，并通过低速链路向低带宽区域转发。为了对多个输入信源进行统一的同步，混合器在多个媒体流之间进行定时调整，产生它自己的定时同步，因此所有从混合器输出的报文都把混合器作为同步信源。为了保证接收者能够正确识别混合器处理前的原始报文发送者，混合器在RTP报头中设置了CSRC标识符队列，以标识那些产生混和报文的原始同步信源。

在Internet环境中，一些会议的参加者可能被隔离在应用级防火墙的外面，这些参加者被禁止直接使用 IP组播地址进行访问，虽然他们可能是通过高速链路连接的。在这些情况下，RTP允许使用转换器作为RTP级中继器。在防火墙两端分别安装一个转换器，防火墙之外的转换器过滤所有接收到的组播报文，并通过一条安全的连接传送给防火墙之内的转换器，内部转换器将这些组播报文再转发送给内部网络中的组播组成员。

威胁多媒体数据传输的一个尖锐的问题就是不可预料数据到达时间。但是流媒体的传输是需要数据的适时的到达用以播放和回放。rtp协议就是提供了时间标签，序列号以及其它的结构用于控制适时数据的流放。在流的概念中时间标签是最重要的信息。发送端依照即时的采样在数据包里隐蔽的设置了时间标签。在接受端收到数据包后,就依照时间标签按照正确的速率恢复成原始的适时的数据。不同的媒体格式调时属性是不一样的。但是rtp本身并不负责同步，rtp只是传输层协议，为了简化运输层处理，提高该层的效率。将部分运输层协议功能（比如流量控制）上移到应用层完成。同步就是属于应用层协议完成的。它没有运输层协议的完整功能，不提供任何机制来保证实时地传输数据，不支持资源预留，也不保证服务质量。rtp报文甚至不包括长度和报文边界的描述。同时rtp协议的数据报文和控制报文的使用相邻的不同端口，这样大大提高了协议的灵活性和处理的简单性。

rtp协议和udp二者共同完成运输层协议功能。udp协议只是传输数据包，不管数据包传输的时间顺序。 rtp的协议数据单元是用udp分组来承载的。在承载rtp数据包的时候，有时候一帧数据被分割成几个包具有相同的时间标签，则可以知道时间标签并不是必须的。而udp的多路复用让rtp协议利用支持显式的多点投递，可以满足多媒体会话的需求。

rtp协议虽然是传输层协议但是它没有作为osi体系结构中单独的一层来实现。rtp协议通常根据一个具体的应用来提供服务，rtp只提供协议框架，开发者可以根据应用的具体要求对协议进行充分的扩展。

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、音频和视频会议。

为什么要使用RTP？

一提到流媒体传输、一谈到什么视频监控、视频会议、语音电话（VOIP），都离不开RTP协议的应用，但当大家都根据经验或者别人的应用而选择RTP协议的时候，你可曾想过，为什么我们要使用RTP来进行流媒体的传输呢？为什么我们一定要用RTP？难道TCP、UDP或者其他的网络协议不能达到我们的要求么？

像TCP这样的可靠传输协议，通过超时和重传机制来保证传输数据流中的每一个bit的正确性，但这样会使得无论从协议的实现还是传输的过程都变得非常的复杂。而且，当传输过程中有数据丢失的时候，由于对数据丢失的检测（超时检测）和重传，会数据流的传输被迫暂停和延时。或许你会说，我们可以利用客户端构造一个足够大的缓冲区来保证显示的正常，这种方法对于从网络播放音视频来说是可以接受的，但是对于一些需要实时交互的场合（如视频聊天、视频会议等），如果这种缓冲超过了200ms，将会产生难以接受的实时性体验。

RTP协议是一种基于UDP的传输协议，RTP本身并不能为按顺序传送数据包提供可靠的传送机制，也不提供流量控制或拥塞控制，它依靠RTCP提供这些服务。这样，对于那些丢失的数据包，不存在由于超时检测而带来的延时，同时，对于那些丢弃的包，也可以由上层根据其重要性来选择性的重传。比如，对于I帧、P帧、B帧数据，由于其重要性依次降低，故在网络状况不好的情况下，可以考虑在B帧丢失甚至P帧丢失的情况下不进行重传，这样，在客户端方面，虽然可能会有短暂的不清晰画面，但却保证了实时性的体验和要求。

多播在网络视频会议方面有着很广泛的应用。如果 服务器为每一路客户端单独建立连接进行数据的传输，这样明显不太合理浪费带宽，因此，多播技术可以很好地解决这种问题，即同一份数据，由服务器发送到公共 的多播地址，各个客户端均监听同一个多播地址来获取数据，这样，既节省了带宽，同时也保证了各个客户端所观看的视频的同步。这样的多播应用TCP协议是不支持的，而RTP协议在最初就是为了实现类似的视频会议的应用而诞生的，对其有非常好的支持。

RTP的profile机制：

RTP为具体的应用提供了非常大的灵活性，它将传输协议与具体的应用环境、具体的控制策略分开，传输协议本身只提供完成实时传输的机制，开发者可以根据不同的应用环境，自主选择合适的配置环境、以及合适的控制策略。

这里所说的控制策略指的是你可以根据自己特定的应用需求，来实现特定的一些RTCP控制算法，比如前面提到的丢包的检测算法、丢包的重传策略、一些视频会议应用中的控制方案等等（这些策略我可能将在后续的文章中进行描述）。

对于上面说的合适的配置环境，主要是指RTP的相关配置和负载格式的定义。RTP协议为了广泛地支持各种多媒体格式（如 H.264, MPEG-4, MJPEG, MPEG），没有在协议中体现出具体的应用配置，而是通过profile配置文件以及负载类型格式说明文件的形式来提供。对于任何一种特定的应用，RTP定义了一个profile文件以及相关的负载格式说明，相关的文件如下所示：

《RTP Profile for Audio and Video Conferences with Minimal Control》（RFC3551）

《RTP Payload Format for H.264 Video》（RFC3984）

《RTP Payload Format for MPEG-4 Audio/Visual Streams》（RFC3016）

如果应用程序不使用专有的方案来提供有效载荷类型(payload type)、顺序号或者时间戳，而是使用标准的RTP协议，应用程序就更容易与其他的网络应用程序配合运行，这是大家都希望的事情。例如，如果有两个不同的公司都在开发因特网电话软件，他们都把RTP合并到他们的产品中，这样就有希望：使用不同公司电话软件的用户之间能够进行通信。

RTP其他的一些良好特性：

RTP协议在设计上考虑到安全功能，支持加密数据和身份验证功能。

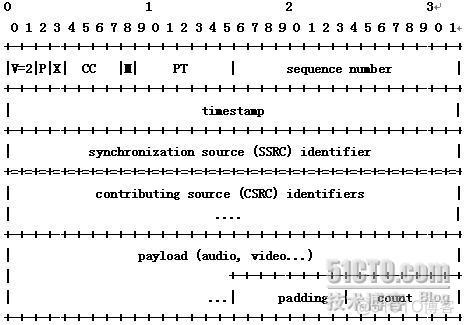
有较少的首部开销，TCP和XTP相对RTP来说具有过多的首部开销(TCP和XTP3.6是40字节，XTP4.0是32字节，而RTP只有12字节)。

## 三、RTP报文格式

RTP报文由两部分组成：报头（RTP Header）和有效载荷（RTP Payload）。

RTP Header占用最少12个字节，最多72个字节。前12字节是固定的，CSRC可以有多个或者0个；

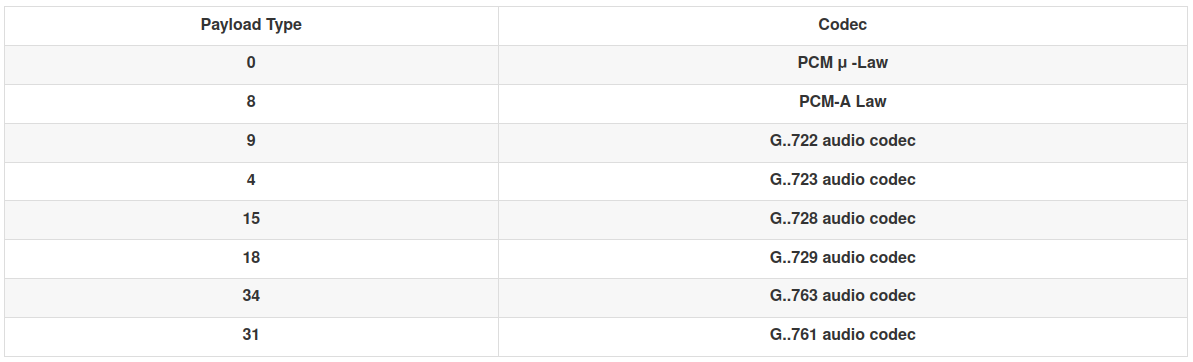
另一部分是RTP Payload，用来封装实际的数据负载，例如h264的裸码流数据。



1. RTP Header：
   * V：RTP协议的版本号，占2位，当前协议版本号为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。

填充的最后一个八进制包含应该忽略的八进制计数。某些加密算法需要固定大小的填充字，或为在底层协议数据单元中携带几个RTP包。

* + X：扩展标志，占1位，如果X=1，RTP固定头部后面就跟有一个扩展头部。
  + CC：CSRC计数器，占4位，指示固定头部后面跟着的CSRC的数目。
  + M：标志，占1位，目的在于允许重要事件在包流中标记出来。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。该位的解释由配置文档（Profile）来承担。
  + PT（payload type）：有效荷载类型，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，这样便于客户端进行解析。

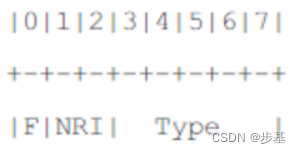


* + 序列号：占16位，用于标识发送者所发送的RTP报文的序列号，每发送一个报文，序列号增1。这个字段当下层的承载协议用UDP的时候，网络状况不好的时候可以用来检查丢包。当出现网络抖动的情况可以用来对数据进行重新排序。序列号的初始值是随机的，使对加密的文本攻击更加困难，同时音频包和视频包的sequence是分别计数的。
  + 时戳（Timestamp）：占32位，必须使用90kHZ时钟频率（程序中的90000）。时戳反映了该RTP报文的第一个八位组的采样时刻。接受者使用时戳来计算延迟和延迟抖动，并进行同步控制。可以根据RTP包的时间戳来获得数据包的时序。同一个帧的不同分片的时间戳是相同的。这样就省去了起始标志和结束标志。一定要记住，不同帧的时间戳肯定是不一样的。
  + 同步信源（SSRC）标识符：占32位，用于标识同步信源。同步信源是指产生媒体流的信源，他通过RTP报头中的一个32为数字SSRC标识符来标识，而不依赖网络地址，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。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，是全局唯一的。
  + 提供信源（CSRC）标识符：0～15项，每项32比特。如超出15个作用源，也仅标识15个。每个CSRC标识了包含在RTP报文有效载荷中的所有提供信源。

提供信源用来标识对一个RTP混合器产生的新包有贡献的所有RTP包的源。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，将原来所有的SSRC都作为CSRC传送给接收者，是接受者知道组成组合报文的各个SSRC。

1. RTP Payload：

rtp头后是rtp载荷，rtp载荷第一个字节格式和NALU头一样。



SODB(String Of Data Bits)：最原始的编码数据, 长度不一定是8的倍数，此时需要对齐。

RBSP：在SODB后添加结尾比特（RBSP trailing bits 一个bit“1”）若干比特“0”,以便字节对齐。

EBSP：在 RBSP 的基础上增加了防止伪起始码字节（0X03）。

NALU是对RBSP的封装，而RTP是对NALU的封装。



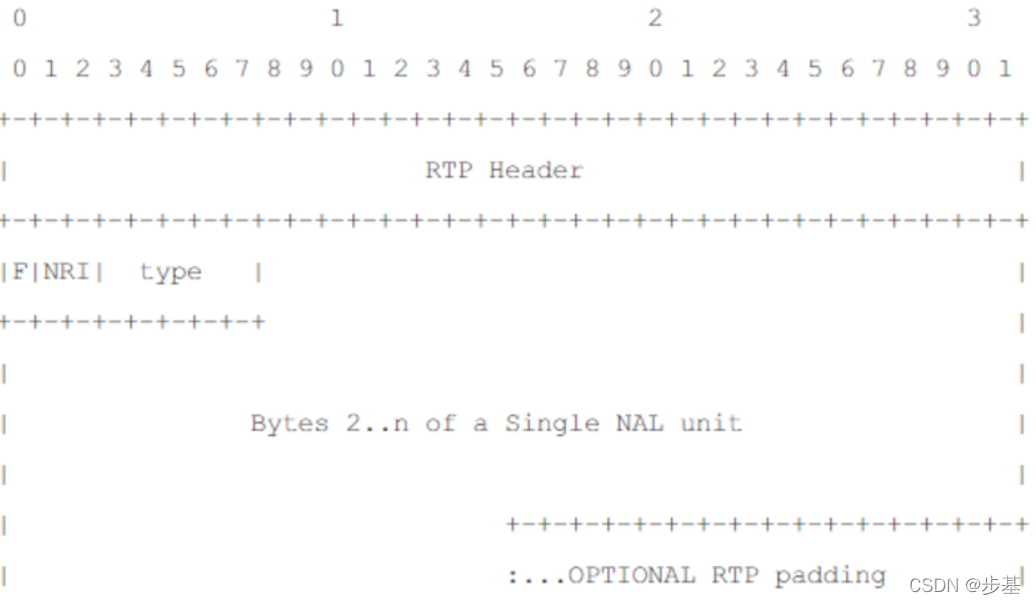
F和NRI也跟NALU头一样，只有Type有些不一样：拓展24 – 31

* + - 0没有定义；
    - 1-23 NAL单元 ，单个 NAL 单元包；
    - 24 STAP-A 单一时间的组合包；
    - 25 STAP-B 单一时间的组合包；
    - 26 MTAP16 多个时间的组合包；
    - 27 MTAP24 多个时间的组合包；
    - 28 FU-A 分片的单元；
    - 29 FU-B 分片的单元；
    - 30-31 没有定义
  1. 单个NAL单元包：

对于NALU（NAL单元）的长度小于MTU大小的包，一般采用单一NAL单元模式。定义在此的NAL单元包必须只包含一个。RTP序号必须符合NAL单元的解码顺序。这种情况下，NAL单元的第一字节和RTP荷载头第一个字节重合。

对于一个原始H264的NALU单元常由[start code] [NALU Header] [NALU Payload]三部分组成，其中start code用于标识这是一个NALU单元的开始，必须是“00 00 00 01”或“00 00 01”,NALU头仅一个字节，其后都是NALU单元载荷。

打包时去除“00 00 01”或“00 00 00 01”的开始码，把其他数据封装成RTP包即可。

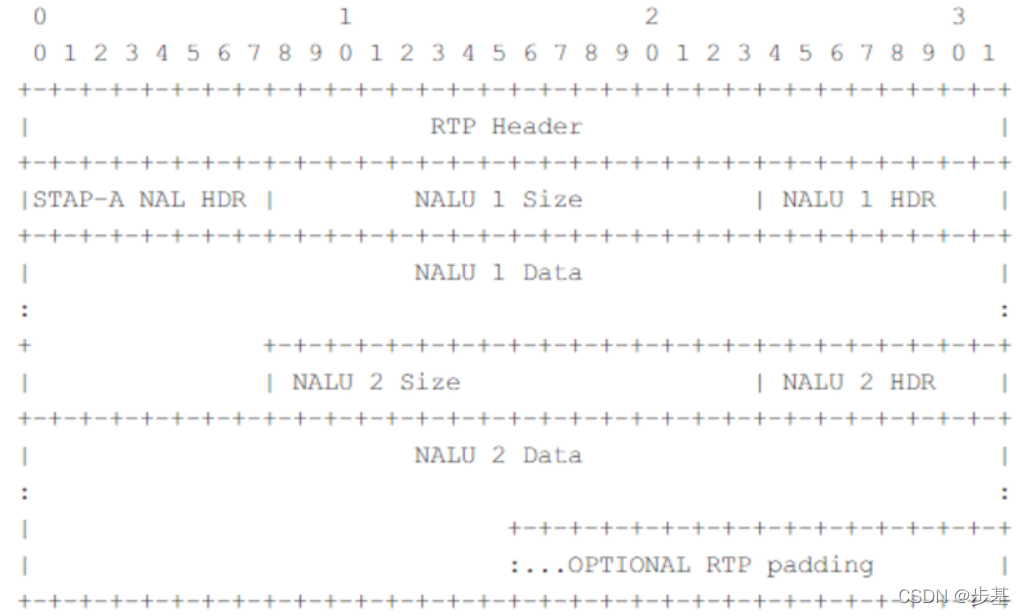


* 1. 组合包：

本类型用于聚合多个NAL单元到单个RTP荷载中。本包有四种版本，单时间聚合包类型A（STAP-A）单时间聚合包类型B（STAP-B）,多时间聚合包类型（MTAP）16位位移（MTAP16）,多时间聚合包类型（MTAP）24位位移（MTAP24）。

赋予STAP-A，STAP-B，MTAP16，MTAP24的NAL单元类型号分别是24 25 26 27。

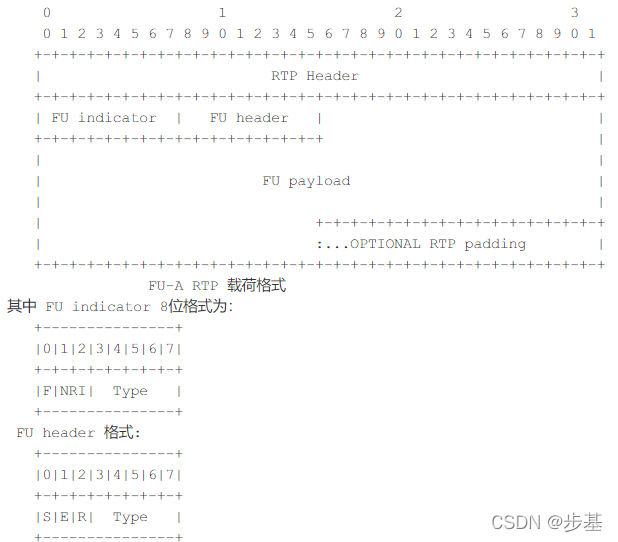
当NALU的长度特别小时，可以把几个NALU单元封在一个RTP包中。这种模式下，有多个NALU头和NALU载荷。



* 1. 分片包：

用于分片单个NAL单元到多个RTP包。现存两个版本FU-A，FU-B，用NAL单元类型（Type）28 29标识。

常用的打包时的分包规则：如果小于MTU采用单个NAL单元包，如果大于MTU就采用FUs分片方式。

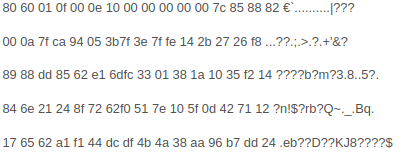


当NALU的长度超过MTU时，就必须对NALU单元进行分片封包，也称为Fragmentation Units (FUs)NAL单元的一个分片由整数个连续NAL单元字节组成。每个NAL单元字节必须正好是该NAL单元一个分片的一部分。相同NAL单元的分片必须使用递增的RTP序号连续顺序发送(第一和最后分片之间没有其他的RTP包）。同时，NAL单元必须按照RTP顺序号的顺序装配。STAPs，MTAPs不可以被分片。 FUs不可以嵌套。 即, 一个FU 不可以包含另一个FU。

* + - S（开始位）: 1 bit， 当设置成1，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（结束位）: 1 bit， 当设置成1，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（保留位）: 1 bit， 保留位必须设置为0，接收者必须忽略该位。
    - Type（类型）：5 bit， 是NAL Header中的Type。

打包时，原始的NAL头的前三位为FU indicator的前三位，原始的NAL头的后五位（Type）为FU header的后五位（Type）。

取一段rtp码流分析如下：



前12字节是RTP Header，7c是FU indicator，85是FU Header，FU indicator（0x7C）和FU Header（0x85）二进制0111 1100 1000 0101。

* + - 0 是F
    - 11 是NRI
    - 11100 是FU Type，这里是28，即FU-A
    - 1 是S，Start，说明是分片的第一包
    - 0 是E，End，如果是分片的最后一包，设置为1，这里不是
    - 0 是R，Remain，保留位，总是0
    - 00101 是NAL Type，这里是5，说明是关键帧

打包时，FU indicator的F、NRI是NAL Header中的F、NRI，Type是28（FU-A）；FU Header的S、E、R分别按照分片起始位置设置，Type是NAL Header中的Type。

解包时，取FU indicator的前三位和FU Header的后五位，即0110 0101（0x65）为NAL类型。

* 1. SPS/PPS：

一个frame分割成多个slice，一个slice分割成多个宏块。一个Slice编码之后被打包进一个NAL单元。不过NAL单元除了容纳Slice编码的码流外，还可以容纳其他数据，比如序列参数集SPS。

H.264码流第一个 NALU是 SPS（序列参数集Sequence Parameter Set）

H.264码流第二个 NALU是 PPS（图像参数集Picture Parameter Set）

SPS和PPS都是特殊的NALU。一个MP4文件只有一个SPS，但是有很多PPS，SPS必须在所有NALU的最开头。

H.264码流第三个 NALU 是 IDR（即时解码器刷新），第一个I帧称为IDR帧，IDR帧的作用是立刻刷新,使错误不致传播,从IDR帧开始,重新算一个新的序列开始编码。解码器收到IDR帧时，将所有的参考帧队列丢弃，把所有的PPS和SPS参数进行更新。

总的来说，H264的码流的打包方式有两种,一种为annex-b byte stream format 的格式，这个是绝大部分编码器的默认输出格式，就是每个帧的开头的3~4个字节是H264的start\_code,0x00000001或者0x000001；另一种是原始的NAL打包格式，就是开始的若干字节（1，2，4字节）是NAL的长度，而不是start\_code,此时必须借助某个全局的数据来获得编 码器的profile，level，PPS，SPS等信息才可以解码。

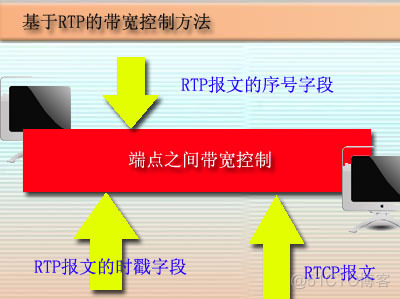
## 四、基于RTP的带宽控制方法

为了实时传输数据，RTP利用了简单而快捷的UDP协议实现网络传输。由于UDP协议是一种无连接传输协议，不保证报文传输的正确性和有序性，也不提供流量控制功能。另一方面，在多媒体通信中，由于多媒体数据的特殊性，不宜采用通常的重传纠错法来提供正确性，而是采用控制传送带宽方式来减少报文丢失，以满足多媒体应用所需的QoS。在RTP协议中，通过 RTCP报文提供了基于无连接传输协议的端到端控制机制，这是一种基于接收者反馈的网络传输QoS监测机制，在RTCP的接收报告中包含了当前网络传输 QoS有关信息，如报文丢失率、报文丢失累计、接收到的最高序列号、平均延迟抖动以及用于计算发布接收报告往返所需时间的时间标签等。发送者可通过这些信息监测和评价网络传输QoS状况，并可采取适当的策略实施同步控制。

RTP协议规定，每个RTP 系统必须实现RTCP的控制功能，由内部功能模块定期自动执行。RTCP报文是轻载信息，其信息量与最低的数据通信量相平衡，它所产生的通信量只是数据通信量的5%左右。

要实施端到端的强制同步控制，其前提条件是发送端要能够获取网络失调状态信息。一种可行的同步控制策略是：各个接收端将一种轻载的网络失调状态信息（如 QoS参数状态）反馈给发送端，发送端据此进行强制性同步控制，以满足接收端演示质量的要求。

基于RTP的带宽控制算法正是利用这种控制策略来实施强制性同步控制的，其基本思想是在RTP协议机制支持下，发送端通过接收端周期反馈的接收报告来评价当前网络传输的QoS，并以此对数据发送速率进行适当调整。端点之间利用RTP报文和RTCP报文来实现带宽控制：



* + RTP报文的序号字段可用于排序RTP报文分组，以消除重复分组，保持视频或音频流内同步和连续地播放。
  + RTP报文的时戳字段可作为流间同步标识，以保持视频和音频流间同步和连续地播放。
  + 发送者可利用接收者反馈的RTCP报文来制实施端到端的强制性同步控制，以改善当前网络传输的QoS。

1. 接收端的控制策略：



* 1. SSRC字段用于标识不同的信源，以支持多对一或多对多的多媒体通信。
  2. 时戳字段作为流间同步标识，用于媒体流间的流间控制，以保持视频和音频流间同步和连续地播放，并作为时间量用于计算报文分组的传输延时、延时抖动以及数据更新周期等，滤除严重延时的RTP报文分组。
  3. 序号字段作为流内同步标识，用于排序RTP报文分组，消除重复报文分组，保持视频或音频流内同步和连续地播放。
  4. 将接收端检测到的当前网络 QoS状况通过RTCP的接收报告周期地反馈给发送端。

1. 发送端的控制策略：



* 1. 设bs为发送端当前的带宽，bmin和bmax分别为应用所设置的最小带宽和最大带宽，且bs?[bmin，bmax]。
  2. 在每个发送带宽级上保持一个时间片，超时后将根据网络QoS状况提高或降低一个带宽级，以避免带宽频繁波动。这里使用报文丢失率作为QoS指示器，并设置一个阈值。如果QoS指示器超阈，说明网络发生阻塞，通过改变发送速率来调整传送带宽，疏导网络交通。
  3. 初始时按最大带宽发送报文分组，即bs?bmax，以提高网络通道的利用率。
  4. 如果在规定的时间片内 QoS指示器超阈，说明网络发生阻塞，则在超时后需要降低一个带宽级，即bs? max { bs-μ, bmin }，其中μ为比例因子。
  5. 如果在规定的时间片内 QoS指示器未超阈，说明网络交通状况良好，则在超时后应当提高一个带宽级，即bs? min { bs+μ, bmax }。
  6. 在点到多点通信场合中，发送者将面对多个不同网段上的接收者，而每个网段的交通状况又不尽相同。因此，在改变带宽时可采用多数表决法，即当报文丢失率超阈的接收者超过一定比例时再改变带宽。

这种方法的特点是：利用 RTP协议机制来传送网络状态信息，不需要另外构造网络检测机构，易于实现；RTCP报文是一种轻载报文，占用较少的通信带宽。

## 五、RTCP协议

RTCP负责管理传输质量在当前应用进程之间交换控制信息。在RTP会话期间，各参与者周期性地传送RTCP包，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，服务器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，能以有效的反馈和最小的开销使传输效率最佳化，故特别适合传送网上的实时数据。

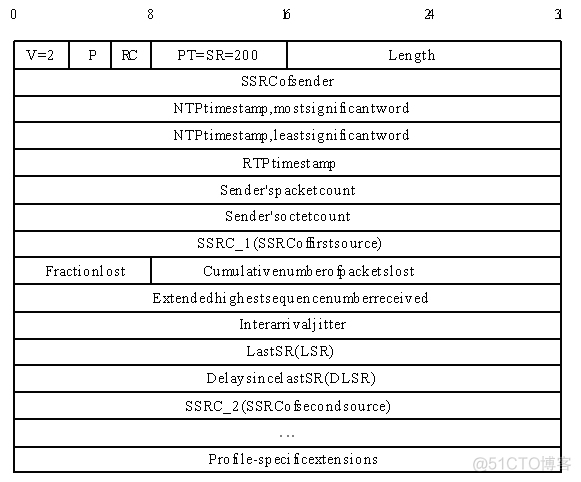
当应用程序开始一个rtp会话时将使用两个端口：一个给rtp，一个给rtcp。rtp本身并不能为按顺序传送数据包提供可靠的传送机制，也不提供流量控制或拥塞控制，它依靠rtcp提供这些服务。在rtp的会话之间周期的发放一些rtcp包以用来传监听服务质量和交换会话用户信息等功能。rtcp包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料。因此，服务器可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型。rtp和rtcp配合使用，它们能以有效的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。根据用户间的数据传输反馈信息，可以制定流量控制的策略，而会话用户信息的交互，可以制定会话控制的策略。

在RTCP通信控制中，RTCP协议的功能是通过不同的RTCP数据报来实现的，主要有如下几种类型：

1. SR:发送端报告，所谓发送端是指发出RTP数据报的应用程序或者终端，发送端同时也可以是接收端。
2. RR:接收端报告，所谓接收端是指仅接收但不发送RTP数据报的应用程序或者终端。
3. SDES:源描述，主要功能是作为会话成员有关标识信息的载体，如用户名、邮件地址、电话号码等，此外还具有向会话成员传达会话控制信息的功能。
4. BYE:通知离开，主要功能是指示某一个或者几个源不再有效，即通知会话中的其他成员自己将退出会话。
5. APP:由应用程序自己定义，解决了RTCP的扩展性问题，并且为协议的实现者提供了很大的灵活性。



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。



* + 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。

## 六、 RTP传输中的时间戳

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 。

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分组中的数据的第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采样的话音信号，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（0.02×8000=160）。因此每发送一个RTP分 组，其时间戳的值就增加160。

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。

## 七、资源预订协议RSVP (Resorce Reservation Protocol)

由于音频和视频数据流比传统数据对网络的延时更敏感，要在网络中传输高质量的音频、视频信息，除带宽要求之外，还需其他更多的条件。RSVP是Internet上的资源预订协议，使用RSVP预留部分网络资源(即带宽)，能在一定程度上为流媒体的传输提供QoS。

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。

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。

## 八、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。

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。

## 九、常见的疑问

* 怎样重组乱序的数据包？

可以根据RTP包的序列号来排序。

* 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。

* 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。

* 接收缓冲和播放缓冲的作用？

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。