第3章　流媒体传输协议

3.1　流媒体网络传输协议

正如前面章节所讲，流媒体传输与其他网络应用数据传输，对网络传输的要求存在很大的不同，流媒体传输更加注意实时性而不是可靠性。因此，类似 TCP 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比特的正确性，使得无论从协议的实现还是传输的过程都变得非常复杂，并且由于超时检测和重传，会导致后续数据流传输的暂停或延时。

当然，可以在客户端通过建立足够大的缓冲区来保证用户的体验，但是其代价是硬件成本的提高。随着高清视频图像的出现，实时视频流的数据会越来越多，而缓冲区的规模将越来越大。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），缓冲区过大，又会产生过大的延时，出现卡顿等现象。因此，适合流媒体传输的网络协议成为解决问题的关键。因此本章着重介绍 RTP、RTCP、RTSP、RSVP 四种协议。

RTP（实时传输协议）是因特网上针对多媒体数据流的一种传输协议。RTP 工作在一对一或一对多的传输情况下，它提供时间标志、序列号以及其他能够保证实时数据传输的标志，其目的是提供时间信息和实现流同步。

RTCP（实时传输控制协议）和 RTP 一起提供流量控制和拥塞控制服务。在 RTP 会话期间，各参与者周期性地传送 RTCP 包。RTCP 包中含有已发送数据报的数量、丢失数据报的数量等统计资料。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，能有效地反馈和最小化开销使传输效率最佳化，因此特别适合传送流媒体实时数据。

RTSP（实时流协议）是由 Real Networks 和 Netscape 共同提出的，该协议定义了一对多应用程序如何有效地通过 IP 网络传送多媒体数据。RTSP 在体系结构上位于 RTP 和 RTCP 之上，它使用 TCP 或 RTP 完成数据传输。

RSVP（资源预留协议）是因特网上的资源预留协议，能在一定程度上为流媒体的传输提供服务质量标准，它不负责传输数据。

3.2　实时传输协议

实时传输协议（Real-time Transport Protocol，RTP）是一个网络传输协议，是由 IETF 的多媒体传输工作小组于 1996 年在 RFC 1889 中公布的，后在 RFC3550 中进行更新。国际电信联盟 ITU-T 也发布了相应的 RTP 文档，命名为 H.225.0，但是当 IETF 发布关于它

的稳定标准 RFC 后，H.225.0 就被取消了。因此它作为互联网标准，在 RFC 3550 中有详细的说明。

RTP 是在互联网上传递音频和视频的标准数据报格式。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 常用于流媒体系统（配合 RTSP）、视频会议和一键通（Push to Talk）系统（配合 H.323 或 SIP），使它成为 IP 电话产业的技术基础。RTP 和 RTCP 一起使用，是建立在 UDP 上的。RTP 标准定义了两个子协议：RTP 和 RTCP。其中，RTP 用于传输实时数据，协议提供的信息包括时间戳（用于同步）、序列号（用于丢包和重排序检测），以及负载格式（用于说明数据的编码格式）；而 RTCP 用于 QoS 反馈和同步媒体流。相对于 RTP 来说，RTCP 所占的带宽非常小，通常只有 5%。

RTP 虽然是传输层协议，但是它不是作为 OSI 体系结构中单独的一层来实现的。RTP 通常根据一个具体的应用来提供服务，这就是说，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。目前，RTP 的设计和研究主要是用来满足多用户的多媒体会议的需要，另外，它也适用于连续数据的存储、交互式分布仿真和一些控制、测量的应用中。

如图 3.1 所示，给出了流媒体应用中的一个典型的协议层次模型。从图 3.1 中可以看出，RTP 被划分在传输层，它建立在 UDP 之上。RTP 和 UDP 二者共同完成传输层协议功能。RTP 数据报作为数据载荷被封装在 UDP 数据报中，UDP 只是传输数据报，不管数据报传输的时间顺序。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，这一点可以从 RTP 的帧结构中看出。

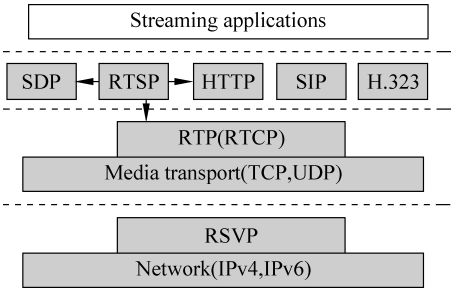

图 3.1　流媒体协议层次模型

RTP 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报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。因此，对于丢失的数据报，不存在由于超时检测而带来的延时。对于丢弃的数据报，可以由上层根据其重要性来选择性地重传。例如，对于视频传输数据的 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。从而在客户端方面，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。

RTP 广泛应用于流媒体传输应用场景，根据 RFC3550 协议，RTP 应用场景有以下四种。

（1）简单多播音频会议（Simple Multicast Audio Conference）。

（2）音频和视频会议（Audio and Video Conference）。

（3）混频器和转换器（Mixers and Translators）。

（4）分层编码（Layered Encodings）。

3.2.1　RTP 帧结构

帧结构是 RTP 实现的基础，如图 3.2 所示。RTP 的帧头包括 12B 的固定长度头部和

15b 的可选长度参与源标识符。RTP 数据报由帧头和数据载荷两部分构成。在 TCP/IP 网络上传输时，将进行 UDP 封装和 IP 封装。而 RTP 帧作为 UDP 的数据载荷，UDP 帧作为 IP 的数据载荷。

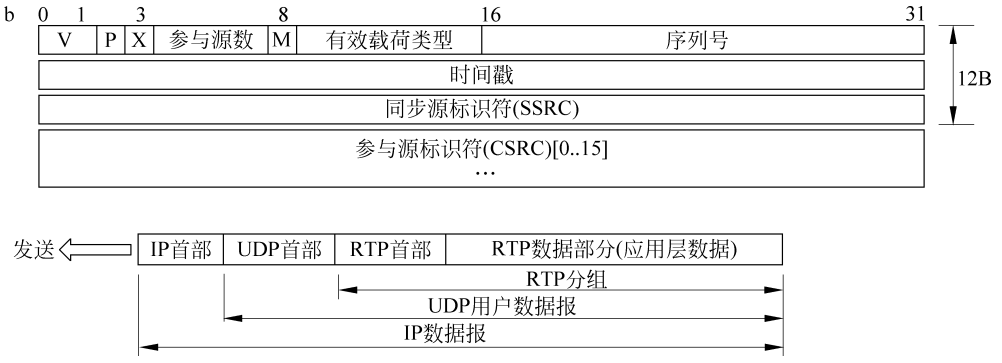

图 3.2　RTP 帧结构

如图 3.2 所示，RTP 帧头的标志位如下。

V(Version，版本)：2b，表明 RTP 版本号。协议初始版本为 0。

P(Padding，填充位)：1b。如果 P 被设置为 1，则表明 RTP 数据报末尾包含额外的附加字节，用于凑齐一个整数倍字节，如 32b。附加信息的最后一个字节表示额外附加信息的长度(包括该字节本身)。填充位是为了保证一些加密机制需要的固定长度的数据块，或者为了在一个底层协议数据单元中传输多个 RTP 数据报。

X(Extension，扩展位)：1b。如果 X 被设置为 1，则在固定的头部后存在一个扩展头部，格式定义在 RFC3550 协议中。几乎所有互联网通信协议都被收录在 RFC 文件中。

CC(CSRC Count，参与源数)：4b。标明 12B 的帧头后存在的 CSRC 标识符数目，CSRC 表示参与源。

M(Marker，标记)：1b。标记的解释由 Profile 配置文件定义，允许重要事件如帧边界在数据报流中进行标记。Profile 可以改变该位的长度，但是要保持 Marker 和 Payload Type 总长度不变(共 8b)。

PT(Payload Type，有效载荷类型)：7b，表明 RTP 包所携带信息的类型，并通过应用程序决定其解释。标准类型列出在 RFC3551 中。如果接收方不能识别该类型，必须忽略此 RTP 包。

Sequence Number(序列号)：16b。每发送一个 RTP 数据报，序列号增加 1。接收方可以根据序列号重新排列数据报顺序。

Timestamp(时间戳)：32b。反映 RTP 包所携带信息中第一个字节的采样时间。时间戳反映了 RTP 数据报中第一个字节的采样时间。采样时钟必须来源于一个单调、线性递增时钟，用来同步和去除网络引起的数据报抖动。该时钟的分辨率必须满足理想的同步精度和测量数据报到来时的抖动需要。时钟分辨率不满足的一种典型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 specication)进行静态描述。也可以通过非 RTP 方法(non-RTPmeans)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同一个包传递或是被丢弃。

时间戳的初始值应当是随机的。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）。选取采样时间作为 RTP 时间戳的参考点是因为它可以被传输的终节点获知，而且对所有媒体内容有一个相同的定义，再者是它不受编码延迟或其他数据处理的约束。目的是对所有在同一时刻采样的媒体进行同步。

SSRC（同步源标识符）：32b。标识数据源，其标识符随机选择，旨在确保在同一个 RTP 会话中不存在两个同步源具有相同的 SSRC 标识符。

CSRC 列表（参与源标识符）：0～15 个参与源的标识符，每个 32b。CSRC 列表指出了对此包中负载内容的所有参与源。识别符的数目在 CC 域中给定。若有参与源多于 15 个，仅识别 15 个。只有存在 Mixer 时才有效，如一个将多声道的语音流合并成一个单声道的语音流，在这里就可以列出原来每个声道的 SSRC 标识符。

3.2.2 RTP 数据报传输过程

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，两个端口中的一个端口给 RTP 包，另一个端口给 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 从上层接收流媒体信息码流（如 H.264），封装成 RTP 数据报；RTCP 从上层接收控制信息，封装成 RTCP 控制包。

（2）RTP 将 RTP 数据报发往 UDP 端口对中的偶数端口；RTCP 将 RTCP 控制包发往 UDP 端口对中的奇数端口。

如图 3.3 所示，RTP 数据报中只包含 RTP 数据，而控制是由 RTCP 提供的。RTP 在 1025～65 535 中选择一个未使用的偶数 UDP 端口号，而在同一次会话中的 RTCP 则使用下一个奇数 UDP 端口号。端口号 5004 和 5005 分别用作 RTP 和 RTCP 的默认端口号。

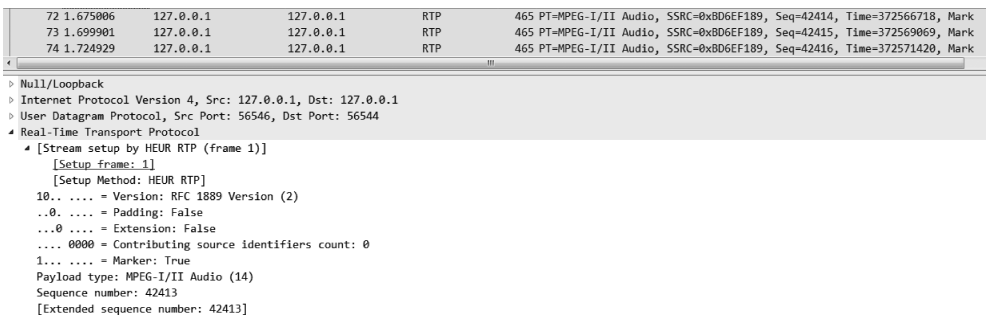

图 3.3　RTP 数据报软件分析

3.2.3 RTP 应用实例

本节将对 MPEG-TS 数据报封装在 RTP 帧的过程，及描述 RTP 相应的数据结构进行详解。其过程如图 3.4 所示。

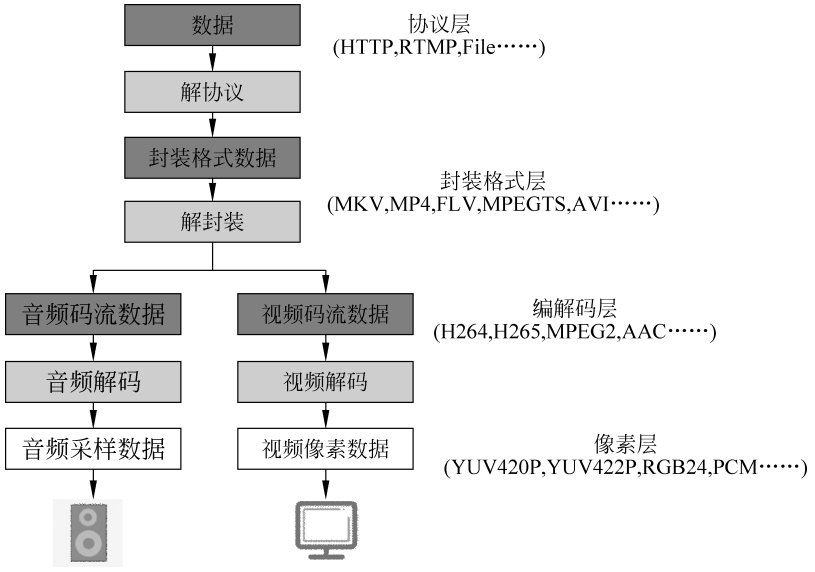

图 3.4　音视频数据接收过程

从网络上接收的数据报，需要通过解 IP 帧封装，解 UDP 帧封装，以及解 RTP 帧封装，把数据报中的音频或者视频数据从数据负载中提取出来，进行音频或者视频解码，产生音频采样数据或视频像素数据，并把数据发送到对应的扬声器或者显示器等设备。

MPEG2-TS 的帧格式中数据大小固定为 188B 的 TS 数据报。如图 3.5 所示，首先将 7 个 MPEG-TS 数据报封装为一个 RTP 数据报，然后将每个 RTP 数据报封装为一个 UDP 数据报。RTP 数据报封装方法就是在 MPEG-TS 数据前面加上 RTP 帧头，而 UDP 数据报封装方法就是在 RTP 数据报前面加上 UDP 帧头。

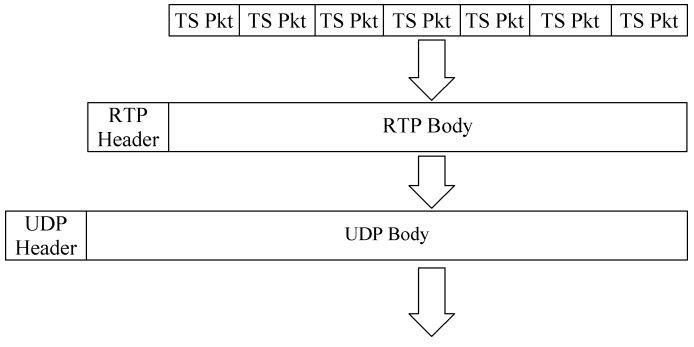

图 3.5　MPEG2-TS 数据报封装过程

用 C 语言定义 RTP 数据报的帧头，如图 3.6 所示，RTP 帧的固定帧头为 12B，每个字节的定义和协议的描述相同，采用了无符号的字符类型、短整型和长整型数据结构。

```
typedef struct RTP_FIXED_HEADER{
   /* byte 0*/
   unsigned char csrc_len:4;                    /* expect 0 */
   unsigned char extension:1;                   /* expect 1 */
   unsigned char padding:1;                     /* expect 0 */
   unsigned char version:2;                     /* expect 2 */
    /* byte 1 */
   unsigned char payload:7;
   unsigned char marker:1;                      /* expect 1 */
   /* bytes 2,3  */
   unsigned short seq_no;
   /* bytes 4-7  */
   unsigned long timestamp;
   /* bytes 8-11 */
   unsigned long ssrc;                          /* stream number is used here. */
} RTP_FIXED_HEADER;
```

图 3.6　C 语言定义 RTP 帧头数据结构

如图 3.7 所示是用 C 语言编写的 RTP 帧解封装程序，simplest_udp_parser()函数采用 while(1)循环语句完成接收数据报的任务；利用 Socket 接口把数据报接收过来，并存放在 recvData[10000]字符数组中；之后通过设置 parse_rtp 标志位为 1，实现 rtp 数据报帧头的解析工作，包括时间戳 timestamp、序列号 seq_no 等；通过设置 parse_mpegts 标志为 1，实现 mpegts 数据报的解析工作，将解析结果存入 myout 文件指针所指向的文件。程序的最后进行了文件和 Socket 接口的关闭等操作。

```
int simplest_udp_parser(int port)
{
    WSASATA wsaData;
    WORD sockVersion = MAKEWORD(2,2);
    int cnt=0;

    //FILE *myout=fopen("output_log.txt","wb+");
    FILE *myout=stdout;

    FILE *fp1=fopen("output_dump.ts","wb+");

    if (WSAStartup(sockVersion,&wsaData) !=0){
        return 0;
    }


if(serSocket == INVALID_SOCKET){
    printf("socket error !");
    return 0;
}
sockaddr_in serAddr;
serAddr.sin_family = AF_INET;
serAddr.sin_port = htons(port);
serAddr.sin_addr.S_un.S_addr = INADDR_ANY;
if(bind(serSocket,(sockaddr *)&serAddr,sizeof(serAddr)) == SOCKET_ERROR){
    printf("bind error !");
    closersocket(serSocket);
    return 0;
}
sockaddr_in remoteAddr;
int nAddrLen = sizeof(remoteAddr);

//How to parse?
int parse_rtp=1;
int parse_mpegts=1;

printf("Listening on port %d\n",port);

char recvData[10000];

while(1){
    int pktsize = recvfrom(serSocket,recvData,10000,0,(sockaddr *)&remoteAddr,&nAddrLen);
    if (pktsize > 0){
        //printf("Addr:%s\r\n",inet_ntoa(remoteAddr.sin_addr));
        //printf("packet size:%d\r\n",pktsize);
        //parse RTP
        //
        if(parse_rtp!=0){
            char payload_str[10]=[0];
            RTP_FIXEO_HEADER rtp_header;
            int rtp_header_size=sizeof(RTP_FIXEO_HEADER);
            //RTP Header
            memcpy((void *)&rtp_header,recvData,rtp_header_size);

            //RFC3551
            char payload=rtp_header.payload;
```

图 3.7　C 语言程序——RTP 帧解封装

```
            switch(payload){
                case 0 :
                case 1 :
                case 2 :
                case 3 :
                case 4 :
                case 5 :
                case 6 :
                case 7 :
                case 8 :
                case 9 :
                case 10 :
                case 11 :
                case 12 :
                case 13 :
                case 14 :
                case 15 :
                case 16 :
                case 17 :
                case 18 :sprintf(payload_str,"Audio");break;
                case 31 :sprintf(payload_str,"H.261");break;
                case 32 :sprintf(payload_str,"MPV");break;
                case 33 :sprintf(payload_str,"MP2T");break;
                case 34 :sprintf(payload_str,"H.263");break;
                case 96 :sprintf(payload_str,"H.264");break;
                default:sprintf(payload_str,"other");break;
            }


            unsigned int timestamp=ntohl(rtp_header.timestamp);
            unsigned int seq_no=ntohs(rtp_header.seq_no);

            fprintf(myout,"[RTP Pkt] %5d| %5s| %10u| %5d| %5d|\n",cnt,payload_str,timestamp,seq_no,pktsize);

            //RTP Data
            char *rtp_data=recvData+rtp_header_size;
            int rtp_data_size=pktsize-rtp_header_size;
            fwrite(rtp_data,rtp_data_size,1,fp1);

            //Parse MPEGTS
            if(parse_mpegts!=0&&payload==33){
                MPEGTS_FIXED_HEADER mpegts_header;
                for (int i=0;i<rtp_data_size;i=i+188){
                    if(rtp_data[i]!=0x47)
                        break;
                    //MPEGTS Header
                    //memcpy((void *)&mpegts_header,rtp_data+i,sizeof(MPEGTS_FIXED_HEADER));
                    fprintf(myout, " [MPEGTS Pkt]\n");
                }
            }
        }else{
            fprintf(myout,"[UDP Pkt]%5d| %5d|\n",cnt,pktsize);
            fwrite(recvData,pktsize,1,fp1);
        }
        cnt++;
    }
}


    closesocket(serSocket);
    WSACleanup();
    fclose(fp1);

    return 0 ;
}
```

图 3.7 （续）

3.3 实时传输控制协议

RTCP(Real-time Transport Control Protocol,实时传输控制协议)被设计为和 RTP 一起使用的,进行流量控制和拥塞控制的服务控制协议。

RTCP 和 RTP 一起提供流量控制和拥塞控制服务。在 RTP 会话期间,各参与者周期性地传送 RTCP 包,如图 3.8 所示。RTCP 包中含有已发送的数据报的数量、丢失的数据

报数量等统计资料。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。

RTCP 就是用于监控服务质量和传达关于在一个正在进行的会议中的参与者的信息，包括对抗卡顿、网络拥塞控制扩展功能的实现，均利用 RTCP 报文实现，RTCP 的 NACK 报文就是实现抗丢包的策略(详见 RFC4585)。

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报提供可靠的传输机制，也不提供流量控制或拥塞控制，RTP 依靠 RTCP 提供这些服务。RTCP 在 RTP 的会话周期里发放一些 RTCP 包用以传输监听服务质量和交换会话用户信息等功能。

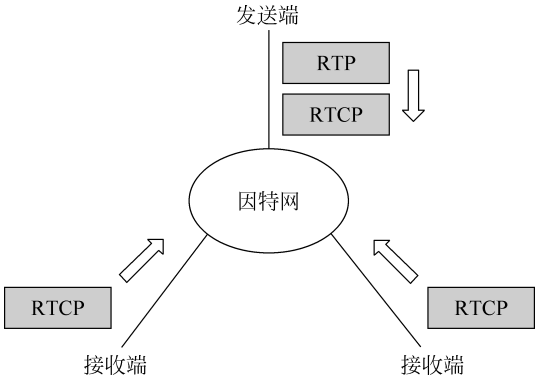

图 3.8　RTCP 控制数据报传输过程

RTCP 原理是向会话中的所有成员周期性地发送控制数据报来实现的，应用程序通过接收这些控制数据报，从中获取会话参与者的相关资料，以及网络状况、分组丢失概率等反馈信息，从而能够对服务质量进行控制或者对网络状况进行诊断。RTCP 处理机制根据需要定义了五种类型的报文(RTCP 包)，它们完成接收、分析、产生和发送控制报文的功能。

RTCP 的功能是通过不同的 RTCP 数据报文来实现的，如表 3.1 所示。

表 3.1　RTCP 的五种类型报文

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

SR(Sender Report)：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。

RR(Receiver Report)：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。

SDES(Source Description Items)：源点描述，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。

BYE 报文：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。

APP 报文：由应用程序定义，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。

RTCP 数据报携带有服务质量监控的必要信息，能够对服务质量进行动态调整，并能够对网络拥塞进行有效控制。由于 RTCP 数据报采用的是组播方式，因此会话中的所有成员都可以通过 RTCP 数据报返回的控制信息，来了解其他参与者的当前情况。

3.3.1 SR 帧结构

发送端 SR(Sender Report)报文的作用是使发送端以多播方式向所有接收端报告发送情况。SR 报文的主要内容有：相应的 RTP 流的 SSRC，RTP 流中最新产生 RTP 报文的时间戳和 NTP，RTP 流包含的分组数，RTP 流包含的字节数。SR 数据报分为三部分：头部(header)，发送者信息(Sender Info)和反馈块(Report Block)。如果发送端也作为接收端，那么才会存在反馈块(Report Block)，当存在多个码流时就会反馈多个 Report Block。SR 报文的负载类型是 200，SR 数据报的封装如图 3.9 所示。

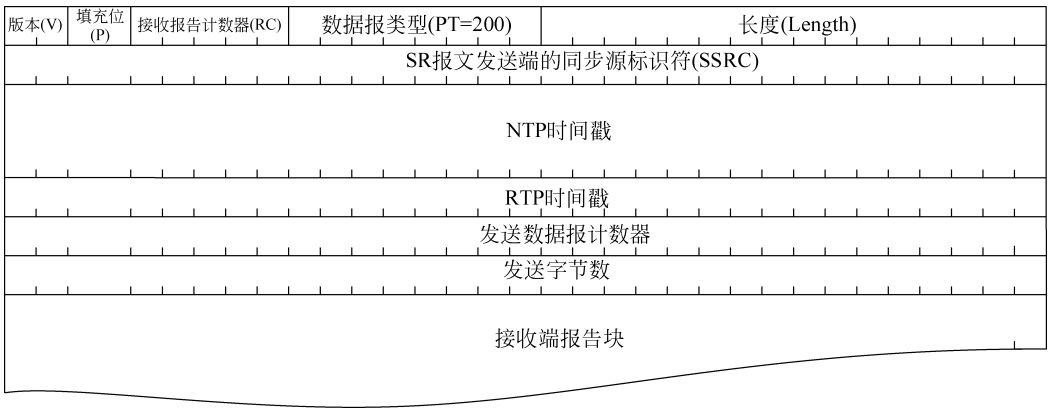

图 3.9　发送端报告分组 SR 帧结构

版本(V)：同 RTP 帧结构的定义相同。

填充位(P)：同 RTP 帧结构的定义相同。

接收报告计数器(RC)：5b，此 SR 数据报中的接收报告块的数目，可以为零。

数据报类型(PT)：8b，SR 数据报的标识是 200。

长度(Length)：16b，其中存放的是此 SR 数据报以 32b 为单位的总长度减 1。

同步源标识符(SSRC)：32b，SR 数据报发送者的同步源标识符。与对应 RTP 中的 SSRC 一样。

NTP(Network Time Protocol)时间戳：32b，SR 数据报发送时的绝对时间值。NTP 的作用是同步不同的 RTP 媒体流。

RTP 时间戳：32b，与 NTP 时间戳对应，与 RTP 数据报中的 RTP 时间戳具有相同的单位和随机初始值。

发送数据报计数器：32b，从开始发送数据报到产生 SR 数据报这段时间里，发送者发送的 RTP 数据报的总数，SSRC 改变时，此计数器清零。

发送字节数：32b，从开始发送数据报到产生 SR 数据报的时间里，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，发送字节数域清零。

同步源 n 的 SSRC 标识符：32b，此报告块中包含的是从该源接收到的报文的统计信息。

SR 报文的接收端报告块中包含的信息，与 RR 报文中的接收端报告块定义相同。

3.3.2 RR 帧结构

RR 报文为接收端反馈的 RTCP 数据报，向服务器端反馈当前接收到的 RTP 数据报情况。RR 报文针对每个信源都提供 RTP 数据报丢失数、已收 RTP 数据报最大序列号、到达时间抖动、接收最后一个 SR 报文的时间、接收最后一个 SR 报文的延迟等信息。

<table>
<tr><td>版本(V)</td><td>填充位(P)</td><td>接收报告计数器(RC)</td><td>数据报类型(PT)</td><td>长度(Length)</td></tr>
<tr><td colspan="5">RR报文发送端的同步源(SSRC)</td></tr>
<tr><td colspan="5">RR报文接收端的同步源(SSRC)</td></tr>
<tr><td>丢失率</td><td colspan="4">累积丢弃数据报的数目</td></tr>
<tr><td colspan="5">收到的扩展最大序列号</td></tr>
<tr><td colspan="5">接收抖动</td></tr>
<tr><td colspan="5">上次接收到的发送端报告的时间戳</td></tr>
<tr><td colspan="5">上次接收到的发送端报告以来的延时</td></tr>
<tr><td colspan="5">接收端报告块</td></tr>
<tr><td colspan="5">…</td></tr>
</table>

图 3.10　发送端报告分组 RR 帧结构

版本（V）：2b，同 RTP 帧结构的定义相同。

填充位（P）：1b，同 RTP 帧结构的定义相同。

接收报告计数器（RC）：5b，此 RR 数据报中的接收报告块的数目，可以为零。

数据报类型（PT）：8b，RR 数据报的标识是 201。

长度（Length）：16b，其中存放的是 RR 数据报，以 32b 为单位的总长度减 1。

RR 报文发送端的同步源（SSRC）：32b，是 RTP 报文接收端的 SSRC。

RR 报文接收端的同步源（SSRC）：32b，是 RTP 报文发送端的 SSRC。

丢失率（Fraction Lost）：8b，表明一个 RR 发送间隔中 RTP 报文的丢失率。计算方法为 loss fraction＝lost rate×256。如果丢包率为 25%，该字段为 25%×256＝64。

累计丢弃数据报的数目：24b，从 SSRC_n 传过来的 RTP 数据报的丢失总数。理论计算方式为：packet lost ＝期待得到报文数量－实际收到报文的数量；实际计算方式为：packet lost＝期待收到最新序列号－第一次收到报文的序列号。

收到的扩展最大序列号：32b，从 SSRC_n 收到的 RTP 数据报中最大的序列号。

接收抖动（Interarrival Jitter）：32b，RTP 数据报接收时间的统计方差的估计值。这里

的接收抖动指的是 RTP 报文发送方的网络传输时间的估计值。计算单位是基于时间戳的单位，也是 32 位无符号整数。因为 RTP 的发送和接收方没有时间同步系统，所以不大可能准确地测量网络传送时间。传输时间＝|RTP 的时间戳－RTP 接收者本地时间|，因为没有发送和接收方的时间同步机制，所以这里关心的不是传输时间，而是两次接收到 RTP 报文传输时间的对比。

上次接收到的发送端报告的时间戳(Last SR，LSR)：32b，取最近从 SSRC_n 收到的 SR 数据报中的 NTP 时间戳的中间 32b。如果目前还没收到 SR 数据报，则该域清零。

上次接收到的发送端报告以来的延时(Delay Since Last SR，DLSR)：32b，上次从 SSRC_n 收到 SR 数据报到发送 RR 报告的延时。

SR 发送端报告和 RR 接收端报告用于提供接收质量反馈，在流媒体应用场合下，发送媒体流的应用程序将周期性地产生发送端报告 SR，此 RTCP 数据报含有不同媒体流间的同步信息，以及已经发送的数据报和字节的计数，接收端根据这些信息可以估计出实际的数据传输速率。另外，接收端会向所有已知的发送端发送接收端报告 RR，此 RTCP 数据报含有已接收数据报的最大序列号、丢失的数据报数目、延时抖动和时间戳等重要信息，发送端应用根据这些信息可以估计出往返时延，并且可以根据数据报丢失概率和时延抖动情况动态调整发送速率，以改善网络拥塞状况，或者根据网络状况平滑地调整应用程序的服务质量。

3.3.3 SDES 帧结构

SDES 是发送源信息描述报文，可以用于描述发送端的名字、邮箱、电话等信息，SDES 的负载类型是 202，如图 3.11 所示。SDES 分为两部分：头部 header 和描述信息组块(chunk)。信息组块(chunk)内需要包含一个 SSRC 和至少一个 SEDS 条目，每个条目用于描述不同的信息。条目中的 length 字段是用于表示后面描述信息的长度。

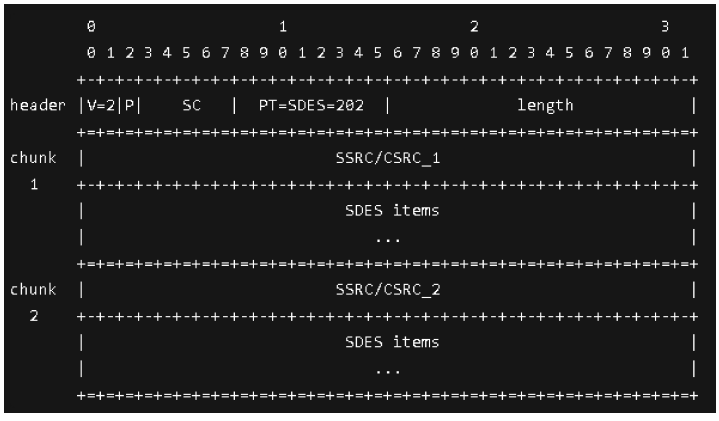

图 3.11　SDES 帧结构

SDES 源描述包提供了直观的文本信息来描述会话的参加者，包括 CNAME、NAME、EMAIL、PHONE、LOC 等源描述项，这些为接收方获取发送方的有关信息提供了方便。SDES 数据报由帧头与数据块组成，数据块可以没有，也可以有多个。帧头由版本(V)、填充位(P)、长度指示、数据报类型(PT)和源计数(SC)组成。PT 占 8b，用于识别 RTCP 的 SDES 数据报，SC 占 5b，表示包含在 SDES 包中的 SSRC/CSRC 块数量。数据块由源描述

项组成，源描述项的内容如下。

CNAME：规范终端标识 SDES 项。类似 SSRC 标识，RTCP 为 RTP 连接中每一个参加者赋予唯一一个 CNAME 标识。在发生冲突或重启程序时，由于随机分配的 SSRC 标识可能发生变化，CNAME 项可以提供从 SSRC 标识到仍为常量的源标识。

如图 3.12 所示，CNAME：规范终端标识 SDES 项。类似 SSRC 标识，RTCP 为 RTP 连接中每一个参加者赋予唯一一个 CNAME 标识。在发生冲突或重启程序时，由于随机分配的 SSRC 标识可能发生变化，CNAME 项可以提供从 SSRC 标识到仍为常量的源标识的绑定。为方便第三方监控，CNAME 应适合程序或人员定位源。

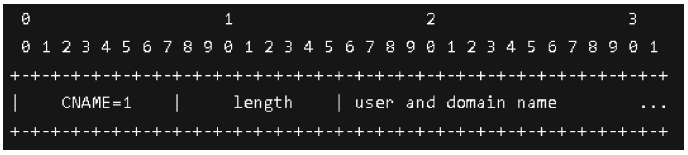

图 3.12　类型 1 的组块

如图 3.13 所示，NAME 用于描述源的真正的名称，如“John Doe，Bit Recycler，Megacorp”，可以是用户想要的任意形式。由于采用文本信息来描述，对诸如会议应用，可以对参加者直接列表显示。NAME 项是除 CNAME 项以外发送最频繁的项目。NAME 值在一次 RTP 会话期间应该保持为常数，但它不该成为连接的所有参加者中的唯一依赖。

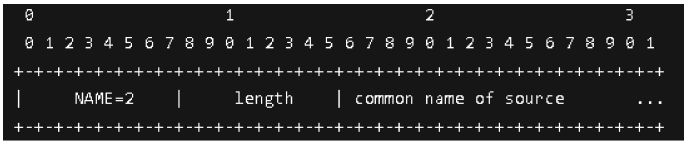

图 3.13　类型 2 的组块

如图 3.14 所示，EMAIL：电子邮件地址 SDES 项。邮件地址格式由 RFC822 规定，如“John. Doe@megacorp. com”。一次 RTP 会话期间，EMAIL 项的内容希望保持不变。

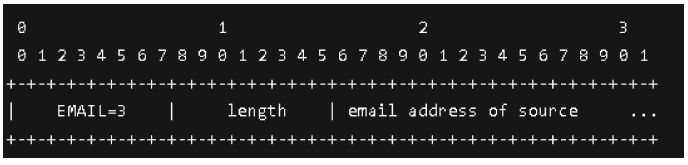

图 3.14　类型 3 的组块

如图 3.15 所示，PHONE：电话号码 SDES 项。电话号码应带有加号，代替国际接入代码，如“+1 908 555 1212”即为美国电话号码。

图 3.15　类型 4 的组块

如图 3.16 所示，LOC：用户地理位置 SDES 项。根据应用，此项具有不同程度的细节。对会议应用，字符串如“Murray Hill，New Jersey”就足够了。然而，对活动标记系统，字符串如“Room 2A244，AT&T BL MH”也许就适用。细节留给实施或用户，但格式和内容可用设置指示。在一次 RTP 会话期间，除移动主机外，LOC 值期望保持不变。

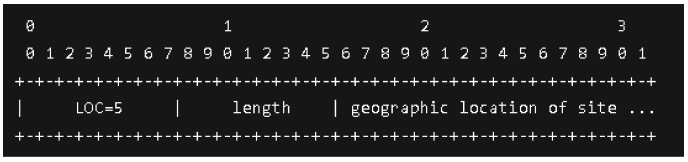

图 3.16　类型 5 的组块

如图 3.17 所示，TOOL：应用或工具名称 SDES 项。TOOL 项包含一个字符串，表示产生流的应用的名称与版本，如“videotool 1.2”。这部分信息对调试很有用，类似于邮件或邮件系统版本 SMTP 头。TOOL 值在一次 RTP 会话期间保持不变。

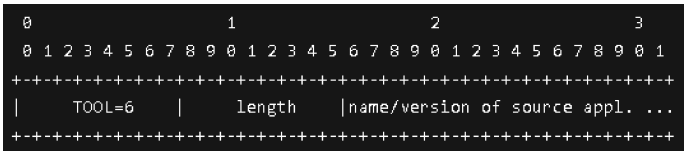

图 3.17　类型 6 的组块

如图 3.18 所示，NOTE：通知/状态 SDES 项。NOTE 项旨在描述源当前状态的过渡信息，如“on the phone，can't talk”，或在讲座期间用于传送谈话的题目，它的语法可在设置中显式定义。NOTE 项一般只用于携带例外信息，而不应包含在全部参加者中，因为这将降低接收报告和 CNAME 发送的速度，损害协议的性能。一般 NOTE 项不作为用户设置文件的项目，也不会自动产生。

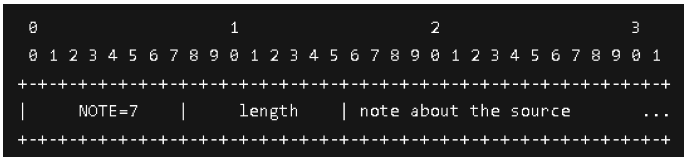

图 3.18　类型 7 的组块

由于 NOTE 项对显示很重要，当会话的参加者处于活动状态时，其他非 CNAME 项（如 NAME）传输速率将会降低，结果使 NOTE 项占用 RTCP 部分带宽。若过渡信息不活跃，NOTE 项继续以同样的速度重复发送几次，并以一个串长为零的字符串通知接收者。

PRIV：专用扩展 SDES 项。

如图 3.19 所示，PRIV 项用于定义实验或应用特定的 SDES 扩展，它由长字符串对组成的前缀，后跟填充该项其他部分和携带所需信息的字符串值组成。前缀长度段为 8b。前缀字符串是定义 PRIV 项人员选择的名称，唯一对应应用接收到的其他 PRIV 项。应用实现者可选择使用应用名称，如有必要，外加附加子类型标识。另外，推荐其他人根据其代表的实体选择名称，然后在实体内部协调名称的使用。

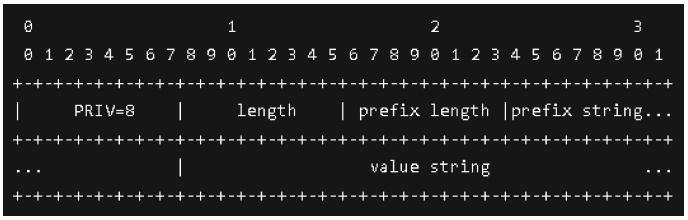

图 3.19　类型 8 的组块

注意，前缀应尽可能短。SDES 的 PRIV 项前缀没在 IANA 处注册。如证实某些形式的 PRIV 项具有通用性，IANA 应给它分配一个正式的 SDES 项类型，这样就不再需要前缀，从而简化应用，并提高传输的效率。

3.3.4　BYE 帧结构

BYE 帧结构的主要功能是表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。发送端主动停止发送的情况，最后会发送一个 BYE 数据报。例如，如果混频器关闭，应该发出一个 BYE 数据报，列出它所处理的所有源，而不只是自己的 SSRC 标识。其他的帧头位和前面 RTP 定义的类似，在这里就不赘述了。BYE 数据报包括一个可选项(24b)，表示离开的原因，如“camera malfunction”或“RTP loopdetected”等，如图 3.20 所示。

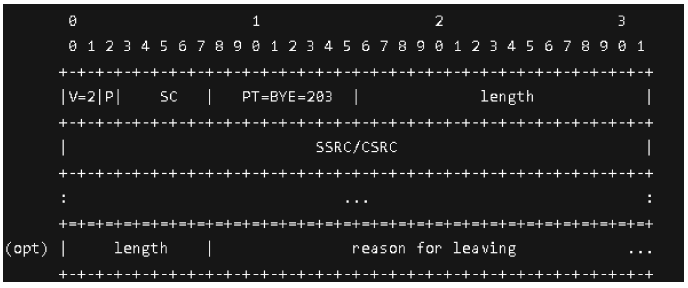

图 3.20　BYE 帧结构

3.3.5　APP 帧结构

APP 帧结构如图 3.21 所示，包括长度、版本号等协议信息，同时包括应用相关的数据。

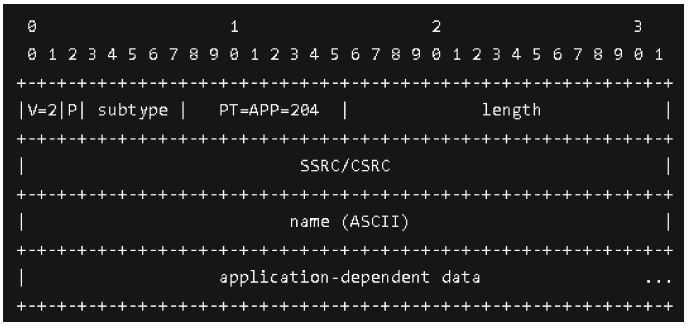

图 3.21　APP 帧结构

3.3.6 RTCP 传输间隔

由于 RTP 允许应用自动扩展，因此可以从几个人的小规模系统扩展成上千人的大规模系统。而每个会话参与者周期性地向所有其他参与者发送 RTCP 控制数据报，如果每个参与者以固定速率发送接收报告，控制流量将随参与者数量线性增长。例如，在音频会议中，数据流量是内在限制的，因为同一时刻只有一两个人说话；对多播，给定连接数据率仍是常数，独立于连接数。但控制流量不是内在限制的。如每个参加者以固定速率发送接收端报告，控制流量将随参加者数量线性增长，因此，速率必须按比例下降。

计算 RTCP 控制数据报间隔，依赖对连接中地址加入数量的估计。当新地址被监听到，就加到此计数中，并在以 SSRC 或 CSRC 标识索引的表中为之建立一个条目，用来跟踪它们。直到收到带有多个新 SSRC 的数据报，新建条目才有效。当收到具有相应 SSRC 标识的 RTCP 的 BYE 数据报时，条目可从表中删除。如果在 RTCP 报告间隔内没有接收到 RTP 或 RTCP 包，参加者可能将另外一个地址标记成不活动，如还未生效就删除掉。这为防止包丢失提供了强大支持，为了“超时”正常工作，所有地址必须对 RTCP 报告间隔记入大致相同的数值。

一旦确认地址有效，如后来标记成不活动，地址的状态应仍保留，地址应继续计入共享 RTCP 带宽地址的总数中，时间要保证能扫描典型网络分区，建议为 30min。注意，这仍大于 RTCP 报告间隔最大值的 5 倍。

3.4 实时流协议

RTSP 定义了一对多应用程序如何有效地通过 IP 网络传送多媒体数据。RTSP 在体系结构上位于 RTP 和 RTCP 之上，它使用 TCP 或 RTP 完成数据传输。HTTP 与 RTSP 相比，HTTP 传送 HTML 超链接文档，而 RTSP 传送的是多媒体数据。HTTP 请求由客户机发出，服务器做出响应；使用 RTSP 时，客户机和服务器都可以发出请求，即 RTSP 可以是双向的。点对点的手机可视通话，必须在手机终端实现 RTSP。

RTSP 最早由 Real Networks 和 Netscape 以及美国哥伦比亚大学共同提出，它是由 Real Networks 的 RealAudio 和 Netscape 的 LiveMedia 的实践和经验发展而来。第一个 RTSP 是由 IETF(Internet Engineering Task Force，因特网工程任务组)在 1996 年 8 月 9 日正式提交为因特网标准，在此后该协议经过了很多明显的变化。RTSP 目前的应用非常广泛，Apple、Vxtreme、Sun 等诸多公司都宣称它们的在线播放器支持 RTSP。

RTSP 是一种客户端到服务器端的多媒体描述协议。RTSP 提供了一个可扩展框架，使实时数据(如音频与视频)的受控、点播成为可能，每个发布和媒体文件被定义为 RTSP UPL，而媒体文件的发布信息则被写进一个被称为媒体发布的文件里，在这个文件中说明了编码器、语言、RTSP ULS、地址、端口号等参数，这个发布文件可以在客户端通过 E - mail 形式或者 HTTP 形式获得。RTSP 以客户端方式工作，对流媒体提供播放、暂停、后退、前进等操作。该标准由 IETF 指定，对应的协议是 RFC2326。

3.4.1 RTSP 的工作原理

RTSP 建立并控制一个或几个时间同步的连续媒体流。尽管连续媒体流与控制流交叉是可能的，通常 RTSP 本身并不发送连续流。换言之，RTSP 充当多媒体服务器的网络远程控制。

目前还没有具体的 RTSP 连接概念，取而代之的是由服务器维持携带标识符的会话。RTSP 会话没有绑定到传输层连接，如 TCP。在 RTSP 会话期间，RTSP 用户可打开或关闭多个对服务器的可靠传输连接以发出 RTSP 请求。此外，可使用无连接传输协议，如 UDP。RTSP 控制的连续媒体流可能用到 RTP，但 RTSP 操作并不依赖用于携带连续媒体的传输机制。RTSP 在语法和操作上与 HTTP 1.1 类似，因此 HTTP 的扩展机制大都可加入 RTSP。尽管如此，在很多重要方面 RTSP 仍不同于 HTTP。

RTSP 支持的操作如下。

(1) 从媒体服务器上检索媒体。

用户可通过 HTTP 或其他方法提交一个演示描述。如果演示是组播，演示描述就包含用于连续媒体的组播地址和端口。如果演示仅通过单播发送给用户，用户为了安全应提供目的地址。

(2) 媒体服务器邀请进入会议。

媒体服务器可被邀请参加正在进行的会议，或回放媒体，或记录其中一部分或全部。这种模式在分布式教育应用上很有用，会议中几方可轮流单击远程控制按钮。

(3) 将媒体加入到现有讲座。

如果服务器告诉用户可获得附加媒体内容，对现场讲座就显得尤其有用。

3.4.2 RTSP 的特点

(1) 可扩展性：新方法和参数很容易加入 RTSP。

(2) 易解析：RTSP 可由标准 HTTP 或 MIME 解析器解析。

(3) 安全：RTSP 使用网页安全机制。

(4) 独立于传输：RTSP 可使用不可靠数据报协议(如 UDP)或者可靠数据报协议(如 TCP)，如要实现应用级可靠，应使用 TCP。

(5) 多服务器支持：每个流可放在不同服务器上，用户端自动同不同服务器建立几个并发控制连接，媒体同步在传输层执行。

(6) 记录设备控制：RTSP 可控制记录和回放设备。

(7) 适合专业应用：通过 SMPT 时间标签，RTSP 支持帧级精度，允许远程数字编辑。

(8) 代理与防火墙友好：协议可由应用和传输层防火墙处理。

(9) HTTP 友好：RTSP 明智地采用了 HTTP，使得结构可以重用。结构包括因特网内容选择平台。

(10) 传输协调：实际处理连续媒体流前，用户可协调传输方法。

(11) 性能协调：如果基本特征无效，必须有一些清理机制让用户决定哪种方法未生效。

3.4.3 RTSP 的结构

如图 3.22 所示，RTSP 是一个基于文本的多媒体播放控制协议，属于应用层。RTSP 以客户端方式工作，对流媒体提供播放、暂停、后退、前进等操作。该标准由 IETF 指定，对应的协议是 RFC2326。

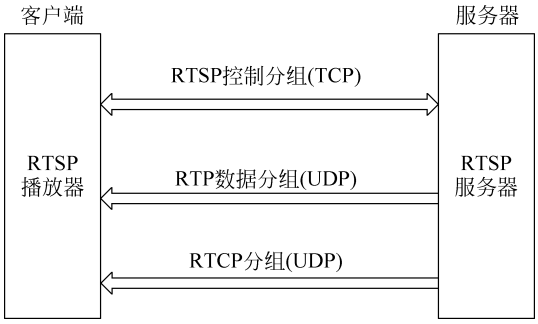

图 3.22 RTSP 传输模型

RTSP 作为一个应用层协议，提供了一个可供扩展的框架，使得流媒体的受控和点播变得可能，它主要用来控制具有实时特性的数据的发送，但其本身并不用于传送流媒体数据，而必须依赖下层传输协议（如 RTP/RTCP）所提供的服务来完成流媒体数据的传送。RTSP 负责定义具体的控制信息、操作方法、状态码，以及描述与 RTP 之间的交互操作。

RTSP 中所有的操作都是通过服务器和客户端的消息应答机制完成的，其中消息包括请求和应答两种。RTSP 是双向的协议，客户端和服务器都可以发送和回应请求。RTSP 是一个基于文本的协议，它使用 UTF-8 编码（RFC2279）和 ISO10646 字符序列，采用 RFC882 定义的通用消息格式，每个语句行由 CRLF 结束。请求消息常用的方法如表 3.2 所示。

表 3.2 请求消息的常用方法

方　法	作　用
OPTIONS	获得服务器提供的可用方法
DESCRIBE	得到会话描述信息
SETUP	客户端请求建立会话，并确立传输模式
TEARDOWN	客户端发起关闭会话请求
PLAY	客户端发起播放请求

如表 3.3 所示，一次基本的 RTSP 交互过程如下，C 表示客户端，S 表示服务器端。

表 3.3 RTSP 客户端与服务器端交互的基本过程

序号	方向	消　息	描　述
①	C→S	OPTION request	Client 询问 Server 有哪些方法可用
	S→C	OPTION response	Server 回应信息中包含所有可用方法
②	C→S	DESCRIBE request	Client 请求得到 Server 提供的媒体初始化描述信息
	S→C	DESCRIBE response	Server 回应媒体初始化信息，主要是 SDP（会话描述协议）

续表

序号	方向	消　息	描　述
③	C→S	SETUP request	设置会话属性以及传输模式，请求建立会话
	S→C	SETUP response	Server 建立会话，返回会话标识符以及会话相关信息
④	C→S	PLAY request	Client 请求播放
	S→C	PLAY response	Server 回应播放请求信息
⑤	S→C	Media Data Transfer	发送流媒体数据
⑥	C→S	TEARDOWN request	Client 请求关闭会话
	S→C	TEARDOWN response	Server 回应关闭会话请求

首先客户端连接到流媒体服务器并发送一个 RTSP 描述请求(DESCRIBE request)，服务器通过一个 SDP(Session Description Protocol)描述来进行反馈(DESCRIBE response)，反馈信息包括流数量、媒体类型等信息。客户端分析该 SDP 描述，并为会话中的每一个流发送一个 RTSP 连接建立请求(SETUP request)，该命令会告诉服务器用于接收媒体数据的端口，服务器响应该请求(SETUP response)并建立连接之后，就开始传送媒体流(RTP 包)到客户端。在播放过程中客户端还可以向服务器发送请求来控制快进、快退和暂停等。最后，客户端可发送一个终止请求(TEARDOWN request)来结束流媒体会话。

请求消息和响应消息如图 3.23 和图 3.24 所示。请求消息由请求行、标题行中的各种标题域和主体实体组成，请求行和标题行由 ASCII 字符组成。

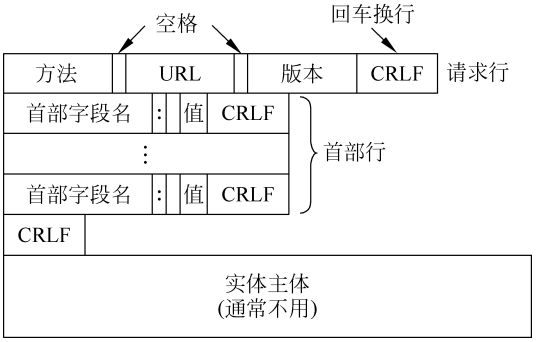

图 3.23　请求消息

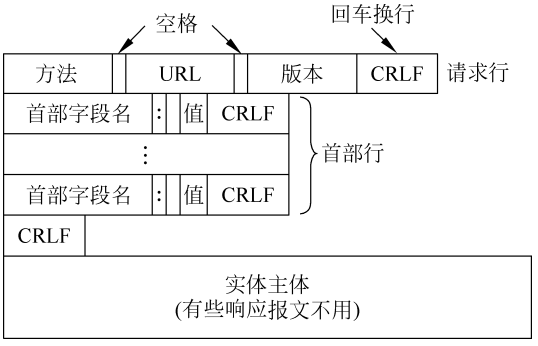

图 3.24　响应消息

请求消息和响应消息中的参数定义如下。

RTSP 版本：RTSP 版本号。

RTSP URL：用于指 RTSP 使用的网络资源。

会议标识：会议标识对 RTSP 来说是模糊的，采用标准 URL 编码方法编码，可包含任何 8 位组数值，会议标识必须全局唯一。

连接标识：连接标识是长度不确定的字符串，必须随机选择，至少要 8 个 8 位组长，使其很难被猜出。

SMPTE 相关时标：SMPTE 相关时标表示相对剪辑开始的时间，相关时标表示成 SMPTE 时间代码，精确到帧级。时间代码格式为小时：分钟：秒：帧。默认 smpte 格式是“SMPTE 30”，帧速率为每秒帧 29.97 帧。其他 SMPTE 代码可通过选择使用 smpte 时间获得支持(如“SMPTE 25”)。时间数值中帧段值可从 0 到 29。每秒 30 帧与每秒 29.97 帧的差别可通过将每分钟的头两帧丢掉来实现。如帧值为零，就可删除。

正常播放时间：正常播放时间表示相对演示开始的流绝对位置。时标由十进制分数组成。左边部分用秒或小时、分钟、秒表示；小数点右边部分表示秒的部分。演示的开始对应 0.0 秒，负数没有定义。特殊常数定义成现场事件的当前时刻，这也许只用于现场事件。直观上，是联系观看者与程序的时钟，通常以数字式显示在 VCR 上。

绝对时间：绝对时间表示成 ISO 8601 时标，采用 UTC(GMT)。

可选标签：可选标签是用于指定 RTSP 新可选项的唯一标记。这些标记用在请求和代理-请求头段。

3.4.4 RTSP 中的方法与实例

客户端向服务器请求媒体资源描述，服务器端通过 SDP(Session Description Protocol)格式回应客户端的请求。资源描述中会列出所请求媒体的媒体流及其相关信息，典型情况下，音频和视频分别作为一个媒体流传输，如图 3.25 和图 3.26 所示。

```
C->S:  OPTIONS rtsp://example.com/media.mp4 RTSP/1.0
       CSeq: 1
       Require: implicit-play
       Proxy-Require: gzipped-messages

S->C:  RTSP/1.0 200 OK
       CSeq: 1
       Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

图 3.25　OPTIONS 方法请求与应答的实例

SETUP 请求确定了具体的媒体流如何传输，该请求必须在 PLAY 请求之前发送。SETUP 请求包含媒体流的 URL 和客户端用于接收 RTP 数据的端口以及接收 RTCP 数据的端口。服务器端的回复通常包含客户端请求参数的确认，并会补充缺失的部分，如服务器选择的发送端口。每一个媒体流在发送 PLAY 请求之前，都要首先通过 SETUP 请求来进行相应的配置，如图 3.27 所示。

```
C->S: DESCRIBE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 2

S->C: RTSP/1.0 200 OK
      CSeq: 2
      Content-Base: rtsp://example.com/media.mp4
      Content-Type: application/sdp
      Content-Length: 460

      m=video 0 RTP/AVP 96
      a=control:streamid=0
      a=range:npt=0-7.741000
      a=length:npt=7.741000
      a=rtpmap:96 MP4V-ES/5544
      a=mimetype:string;"video/MP4V-ES"
      a=AvgBitRate:integer;304018
      a=StreamName:string;"hinted video track"
      m=audio 0 RTP/AVP 97
      a=control:streamid=1
      a=range:npt=0-7.712000
      a=length:npt=7.712000
      a=rtpmap:97 mpeg4-generic/32000/2
      a=mimetype:string;"audio/mpeg4-generic"
      a=AvgBitRate:integer;65790
      a=StreamName:string;"hinted audio track"
```

图 3.26　DESCRIBE 方法请求与应答的实例

```
C->S: SETUP rtsp://example.com/media.mp4/streamid=0 RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001

S->C: RTSP/1.0 200 OK
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=8000-8001;server_port=9000-9001;ssrc=1234ABCD
      Session: 12345678
```

图 3.27　SETUP 方法请求与应答的实例

客户端通过 PLAY 请求来播放一个或全部媒体流，PLAY 请求可以发送一次或多次，发送一次时，URL 为包含所有媒体流的地址，发送多次时，每一次请求携带的 URL 只包含一个相应的媒体流。PLAY 请求中可指定播放的 Range，若未指定，则从媒体流的开始播放到结束，如果媒体流在播放过程中被暂停，则可在暂停处重新启动流的播放，如图 3.28 所示。

```
C->S: PLAY rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 4
      Range: npt=5-20
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 4
      Session: 12345678
      RTP-Info: url=rtsp://example.com/media.mp4/streamid=0;seq=9810092;rtptime=3450012
```

图 3.28　PLAY 方法请求与应答的实例

PAUSE 请求会暂停一个或所有媒体流，后续可通过 PLAY 请求恢复播放。PAUSE 请求中携带所请求媒体流的 URL，若参数 Range 存在，则指明在何处暂停，若该参数不存在，则暂停立即生效，且暂停时长不确定，如图 3.29 所示。

```
C->S: PAUSE rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 5
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 5
      Session: 12345678
```

图 3.29　PAUSE 方法请求与应答的实例

结束会话请求，该请求会停止所有媒体流，并释放服务器上的相关会话数据，如图 3.30 所示。

```
C->S: TEARDOWN rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 8
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 8
```

图 3.30　TEARDOWN 方法请求与应答的实例

检索指定 URL 数据中的参数值。不携带消息体的 GET_PARAMETER 可用来测试服务器端或客户端是否可通(类似 ping 的功能)，如图 3.31 所示。

```
S->C: GET_PARAMETER rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 9
      Content-Type: text/parameters
      Session: 12345678
      Content-Length: 15

      packets_received
      jitter

C->S: RTSP/1.0 200 OK
      CSeq: 9
      Content-Length: 46
      Content-Type: text/parameters

      packets_received: 10
      jitter: 0.3838
```

图 3.31　GET_PARAMETER 方法请求的实例

重定向请求，用于服务器通知客户端新的服务地址，客户端需要向这个新地址重新发起请求。重定向请求中可能包含 Range 参数，指明重定向生效的时间。客户端若需向新服务地址发起请求，必须先 TEARDOWN 当前会话，再向指定的新主机 SETUP 一个新的会话，如图 3.32 所示。

```
S->C: REDIRECT rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 11
      Location: rtsp://bigserver.com:8001
      Range: clock=19960213T143205Z-
```

图 3.32　REDIRECT 方法请求的实例

除了上面所述的 RTSP 支持的基本方法，表 3.4 中是 RTSP 支持的所有方法。

表 3.4 RTSP 支持的方法

方　法	指向	对象	要求	含　义
DESCRIBE	C→S	P,S	推荐	检查演示或媒体对象的描述,也允许使用接收头指定用户理解的描述格式。DESCRIBE 的答复-响应组成媒体 RTSP 初始阶段
ANNOUNCE	C→S S→C	P,S	可选	当从用户发往服务器时,ANNOUNCE 将请求 URL 识别的演示或媒体对象描述发送给服务器;反之,ANNOUNCE 实时更新连接描述。如新媒体流加入演示,整个演示描述再次发送,而不仅仅是附加组件,使组件能被删除
GET_PARAMETER	C→S S→C	P,S	可选	GET_PARAMETER 请求检查 URL 指定的演示与媒体的参数值。没有实体时,GET_PARAMETER 也许能用来测试用户与服务器的连通情况
OPTIONS	C→S S→C	P,S	要求	可在任意时刻发出 OPTIONS 请求,如用户打算尝试非标准请求,并不影响服务器状态
PAUSE	C→S	P,S	推荐	PAUSE 请求引起流发送临时中断。如请求 URL 命名一个流,仅回放和记录被停止;如请求 URL 命名一个演示或流组,演示或组中所有当前活动的流发送都停止。恢复回放或记录后,必须维持同步。在 SETUP 消息中连接头超时参数所指定时段期间被暂停后,尽管服务器可能关闭连接并释放资源,但服务器资源会被预订
PLAY	C→S	P,S	要求	PLAY 告诉服务器以 SETUP 指定的机制开始发送数据;直到一些 SETUP 请求被成功响应,客户端才可发布 PLAY 请求。PLAY 请求将正常播放时间设置在所指定范围的起始处,发送流数据直到范围的结束处。PLAY 请求可排成队列,服务器将 PLAY 请求排成队列,顺序执行
RECORD	C→S	P,S	可选	该方法根据演示描述初始化媒体数据记录范围,时标反映开始和结束时间;如没有给出时间范围,使用演示描述提供的开始和结束时间。如连接已经启动,立即开始记录,服务器数据请求 URL 或其他 URL 决定是否存储记录的数据;如服务器没有使用 URL 请求,响应应为 201(创建),并包含描述请求状态和参考新资源的实体与位置头。支持现场演示记录的媒体服务器必须支持时钟范围格式,smpte 格式没有意义
REDIRECT	S→C	P,S	可选	重定向请求通知客户端连接到另一服务器地址。它包含强制头地址,指示客户端发布 URL 请求;也可能包括参数范围,以指明重定向何时生效。若客户端要继续发送或接收 URL 媒体,客户端必须对当前连接发送 TEARDOWN 请求,而对指定的新连接发送 SETUP 请求

续表

方　法	指向	对象	要求	含　义
SETUP	C→S	S	要求	对 URL 的 SETUP 请求指定用于流媒体的传输机制。客户端对正播放的流发布一个 SETUP 请求，以改变服务器允许的传输参数。如不允许这样做，响应错误为“455 Method Not Valid In This State”。为了通过防火墙，客户端必须指明传输参数，即使对这些参数没有影响
SET_PARAMETER	C→S S→C	P,S	可选	这个方法请求设置演示或 URL 指定流的参数值。请求仅应包含单个参数，允许客户端决定某个特殊请求为何失败。如请求包含多个参数，所有参数可成功设置，服务器必须只对该请求起作用。服务器必须允许参数可重复设置成同一值，但不让改变参数值。注意：媒体流传输参数必须用 SETUP 命令设置。将设置传输参数限制为 SETUP 有利于防火墙。将参数划分成规则排列形式，结果有更多有意义的错误指示
TEARDOWN	C→S	P,S	要求	TEARDOWN 请求停止给定 URL 流发送，释放相关资源。如 URL 是此演示 URL，任何 RTSP 连接标识不再有效。除非全部传输参数是连接描述定义的，SETUP 请求必须在连接可再次播放前发布

注：P—演示，S—流，C—客户端，S—服务器端

响应的状态与意义如表 3.5 所示。

表 3.5　响应的状态与意义

响应状态值	意　义
100	继续(Continue (all 100 range))
200	可以(OK)
201	创建(Created)
250	存储空间低(Low on Storage Space)
300	多重选择(Multiple Choices)
301	永久被移除(Moved Permanently)
302	暂时被移除(Moved Temporarily)
303	看其他的(See Other)
304	不被修改(Not Modified)
305	使用代理(Use Proxy)
350	消失(Going Away)
351	负载平衡(Load Balancing)
400	错误的请求(Bad Request)
401	没被授权(Unauthorized)
402	需要付费(Payment Required)
403	禁止(Forbidden)
404	没被发现(Not Found)
405	不被允许的方法(Method Not Allowed)

续表

响应状态值	意　义
406	不被接受(Not Acceptable)
407	需要代理授权(Proxy Authentication Required)
408	请求超时(Request Time-out)
410	消失(Gone)
411	需要的长度(Length Required)
412	前提条件失败(Precondition Failed)
413	请求实体太大(Request Entity Too Large)
414	请求 URL 太大(Request-URL Too Large)
415	不支持的媒体类型(Unsupported Media Type)
451	不理解的参数(Parameter Not Understood)
452	保留(reserved)
453	带宽不足(Not Enough Bandwidth)
454	没找到会话(Session Not Found)
455	此状态下方法无效(Method Not Valid in This State)
456	资源的帧头无效(Header Field Not Valid for Resource)
457	无效范围(Invalid Range)
458	参数只读(Parameter Is Read-Only)
459	不被允许的聚合操作(Aggregate operation not allowed)
460	只允许聚合操作(Only aggregate operation allowed)
461	不被支持的传输(Unsupported Transport)
462	目的地不可达(Destination unreachable)
500	内部服务器错误(Internal Server Error)
501	不被实现(Not Implemented)
502	坏的网关(Bad Gateway)
503	不可用的服务(Service Unavailable)
504	网关超时(Gateway Time-out)
505	RTSP 版本不被支持(RTSP Version not supported)
551	不支持选项(Option not supported)

3.5　资源预留协议

资源预留协议(Resource Reservation Protocol,RSVP)最初是 IETF 为 QoS 的综合服务模型定义的一个信令协议,用于在流(Flow)所经路径上为该流进行资源预留,从而满足该流的 QoS 要求。

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和视频信息等,除带宽要求之外,还需要其他更多的条件。RSVP 是一种工作于因特网上的资源预留协议,利用 RSVP 预留一部分网络资源(即带宽),能在一定程度上为流媒体的传输提供 QoS(服务质量)。它允许路由器网络任何一端上终端系统或主机在彼此之间建立保留带宽路径,为网络上的数据传输预订和保证 QoS。它对于需要保证带宽和时延的业务,如语音传输、视频会议等具有十分重要的作用,在一些网络视频会议工具中就集成了 RSVP。

3.5.1 RSVP 的架构与工作原理

RSVP 的基本架构包含决策控制(Policy)、接纳控制(Admission)、分类控制器(Classifier)、分组调度器(Scheduler)与 RSVP 处理模块等几个主要成分。决策控制用来判断用户是否拥有资源预留的许可权；接纳控制则用来判断可用资源是否满足应用的需求，主要用来减少网络负荷；分类控制器用来决定数据分组的通信服务等级，主要用来实现分组过滤；分组调度器则根据服务等级进行优先级排序，主要用来实现资源配置以满足特定的 QoS。当决策控制或接纳控制未能获得许可时，RSVP 处理模块将产生预留错误消息并传送给收发端点；否则将由 RSVP 处理模块设定分类与调度控制器所需的通信服务质量参数。其基本架构如图 3.33 所示，各个模块相互协作，完成资源的预留。

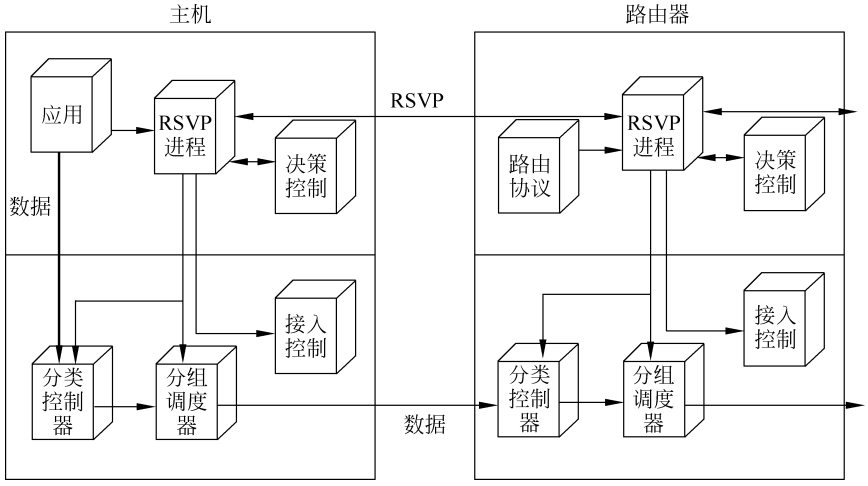

图 3.33 RSVP 基本架构

其协议工作模型如图 3.34 所示。资源预留的过程从应用程序流的源节点发送 Path 消息开始，该消息会沿着流所经路径传到流的目的节点，并沿途建立路径状态；目的节点收到该 Path 消息后，会向源节点回送 Resv 消息，沿途建立预留状态，如果源节点成功收到预期的 Resv 消息，则认为在整条路径上资源预留成功。

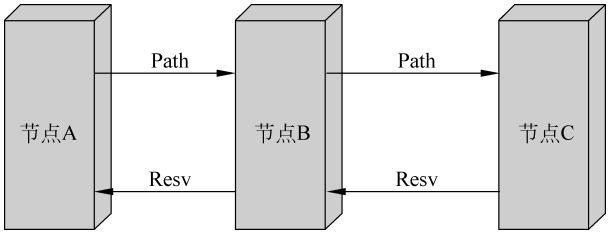

图 3.34 RSVP 工作模型

RSVP 是由接收者提出资源预留申请的，这种申请是单向的，也就是说，为从节点 A 到节点 C 的数据流预留资源，对于从节点 C 到节点 A 的数据流是不起作用的。因为在当前的因特网中，双向的路由是不对称的：从节点 A 到节点 C 的路径并不一定是从节点 C 到节点

A 的路径的反向；另外，两个方向的数据传输特征和对应申请预留的资源也未必相同。因此，RSVP 只在单方向上进行资源请求。即使是相同的应用程序，也可能既是发送者也是接收者。但 RSVP 对发送者与接收者在逻辑上是有区别的。RSVP 运行在 IPv4 或 IPv6 上层，位于七层协议栈中传输层协议的第四层。

RSVP 不传输应用数据，但支持路由控制协议，如 ICMP、IGMP 或者路由选择协议。RSVP 本质上并不属于路由选择协议，RSVP 的设计目标是与当前和未来的单播和组播路由选择协议同时运行。RSVP 参照本地路由选择数据库以获得传送路径。以组播为例，主机发送 IGMP 信息以加入组播组，然后沿着组播组传送路径，发送 RSVP 信息以预留资源。路由选择协议决定数据报转发到哪里，RSVP 只考虑根据路由选择所转发的数据报的 QoS。为了有效适应大型组、动态组成员以及不同机种的接收端需求，通过 RSVP，接收端可以请求一个特定的 QoS。

RSVP 的两个重要概念是“流”与“预留”。流是从发送者到一个或多个接收者的连接特征，通过报文中的“流标记”来体现。发送一个流前，发送者传输一个路径信息到目的接收方，这个报文包括源 IP 地址、目的 IP 地址和一个流规格。这个流规格是由流的速率和延迟组成的，这些都是流媒体的 QoS 所需要的。

3.5.2 RSVP 的数据流

在 RSVP 中，数据流是一系列信息，有着相同的源、目的（可有多个）和 QoS。QoS 要求通过网络以流规格进行通信。RSVP 支持三种传输类型：尽最大可能（Best-effort）、速率敏感（Rate-sensitive）与延迟敏感（Delay-sensitive）。尽最大可能传输为传统 IP 传输，应用包括文件传输（如邮件传输）、磁盘映像、交互登录和事务传输，支持最好性能传输的服务称为最好性能服务；速率敏感传输放弃及时性，而确保速率，RSVP 服务支持速率敏感传输，称为速率保证服务；延迟敏感传输要求传输及时，并因而改变其速率，RSVP 服务支持延迟敏感传输，被称为实时服务。

RSVP 数据流的基本特征是连接，报文在其上流通。连接是具有相同单播或组播目的的数据流，RSVP 分别处理每个连接。RSVP 支持单播和组播连接，而流总是从发送者开始的。特定连接的报文被导向同一个 IP 目的地址或公开的目的端口。IP 目的地址可能是组播发送的组地址，也可能是单个接收者的单播地址。

RSVP 数据发布是通过组播或单播实现的。组播传输将某个发送者的每个数据报复制转发给多个目的端口。单播传输的特征是只有一个接收者。即使目的地址是单播，也可能有多个接收者，以不同端口区分。多个发送者也可能存在单播地址，在这种情况下，RSVP 可建立多对一传输的资源预留。每个 RSVP 发送者和接收者对应唯一的因特网主机。然而，单个主机可包括多个发送者和接收者，以不同端口进行区分。

RSVP 支持两种主要资源预留：独占资源预留和共享资源预留。独占资源预留所要求的预留资源只用于一个发送者。即在同一会话（session）中的不同发送者分别占用不同的预留资源。而共享资源预留所要求的预留资源用于一个或多个发送者。即在同一会话（session）中的多个发送者共享预留资源。

3.5.3 RSVP 隧道

在整个因特网上同时配置 RSVP 或任意其他协议都是不可能的。实际上，RSVP 绝不可能在每个地方都被配置。因此，RSVP 必须提供正确的协议操作，即使只有两个支持 RSVP 的路由器与一群不支持 RSVP 的路由器相连。一个中等规模不支持 RSVP 的网络不能执行资源预留，因而服务保证也就不能实现。然而，如该网络有充足的额外容量，也可以提供可接受的实时服务。

为了允许 RSVP 网络连接通过不支持 RSVP 的网络，RSVP 支持隧道技术。隧道技术要求 RSVP 和非 RSVP 路由器用本地路由表转发到目的地址的路径信息。当路径信息通过非 RSVP 网络时，路径信息复制携带最后一个支持 RSVP 的路由器的 IP 地址。预留请求信息转发给下一个支持 RSVP 的路由器。

3.5.4 RSVP 帧格式

RSVP 支持四种基本的消息：资源预留请求消息、路径消息、错误和确认消息、拆链消息。

资源预留请求消息(Reservation-Request Message)：一个资源预留请求消息由接收方主机向发送方主机发送。资源预留请求消息使用同数据报路由方向相反的方向传送，直至到达发送方主机。一个资源预留请求消息必须到达发送方主机，只有这样，发送方才能为传输的第一跳设置合适的控制参数。

路径消息(Path Message)：一个路径消息由发送方通过单播或组播路由向外发送。路径消息用于存储每个节点的路径状态。资源预留请求正是通过这些路径状态才能从相反方向回到发送方的。

错误和确认消息(Error and Confirmation Message)：错误消息有两种类型，即 PathErr 和 ResvErr。PathErr 由路径消息引起，并传送到发送者。ResvErr 消息由预留消息引起，并传送到相关的接收者。

拆链消息(Teardown Message)：RSVP 拆链用于超时之前删除路径和预留状态。拆链消息有两种类型：PathTear 和 ResvTear。PathTear 删除从消息发出的节点到所有的接收者路径上的预约状态，PathTear 的路由和路径消息的路由严格一致。ResvTear 删除从消息发出的节点到所有发送者路径上的预约状态，ResvTear 的路由和预留消息的路由严格一致。ResvTear 消息可以由一个接收者，或一个状态超时或预约被剥夺的节点产生。节点上状态的删除可能会引起本节点相关预约状态的更新。

RSVP 帧格式由公共帧头(见表 3.6)和不同报文组成。

表 3.6 RSVP 帧公共帧头

4b	4b	8b	16b	16b	8b	8b	32b	1b	16b
Version	Flag	Type	Checksum	Length	Reserved	Send TTL	Message ID	MF	Fragment Offset

Version：4b，协议版本号。

Flag：4b，标志位。

Type：8b，表示消息类型，1 表示 Path 路径消息，2 表示 Resv 资源预订请求消息，3 表示 PathErr 路径错误消息，4 表示 ResvErr 资源预订错误消息，5 表示 PathTear 路径断开消息，6 表示 ResvTear 资源预订断开消息，7 表示 Resv-Conf 资源预订请求确认消息。

Checksum：16b，表示基于 RSVP 消息的标准 TCP/UDP 校验和。

Length：16b，表示 RSVP 帧的字节长度，包括公共帧头和随后携带的信息。

Send TTL：8b，表示发送报文所使用的 IP 生存时间值。

Message ID：32b，提供下一 RSVP 跳/前一 RSVP 跳消息中所有片段共享标签。

MF：1b，片段标志，一个字节的最低位，其他 7b 用于预订。除消息的最后一个片段外，都将设置 MF。

Fragment Offset：16b，表示报文中片段的字节偏移量。

1. 发送者的 Path 报文

RSVP 规定，发送者在发送数据前首先要发送 Path 报文与接收者建立一个传输路径，并协商 QoS 级。一个 Path 报文包含如表 3.7 所示的信息。

表 3.7　Path 报文

Phop	Sender Template	Sender Tspec	Adspec

Phop：后续节点地址，指出转发该 Path 消息的下一个支持 RSVP 节点（路由器或接收端）的 IP 地址。该路径上每个支持 RSVP 的路由器都要更新这个地址。

Sender Template：发送者模板，包括发送者的 IP 地址和可选择的发送者端口。

Sender Tspec：发送者传输说明，其传输说明是用一种漏桶流量模型描述的，其中有数据流峰值速率 p、桶深 b、标记桶速率 r、最小管理单元 m 以及最大数据报长度 M 等参数。

Adspec：通告说明，可选项，含有 OPWA（One Pass With Advertising）信息，使得接收者能计算出应预留的资源级，以获得指定的端到端 QoS。该路径上每个支持 RSVP 的路由器都要更新这些信息。

尽管 Adspec 是一个可选项，但它含有接收者用来计算 QoS 的重要信息。因此，这里对 Adspec 进行详细的讨论。Adspec 由一个报文头、一个默认通用参数（Default General Parameters，DGP）段以及至少一个 QoS 段组成。目前，RSVP 支持 GS 和 CLS 两个基本的 QoS 类，省略 QoS 段的 Adspec 是无效的。

表 3.8　DGP 段

最小路径等待时间
路径带宽
全局中止位
综合服务（IS）网段（hop）计数
路径最大传输单元（PathMTU）

最小路径等待时间：是指在路径上单个连接等待时间的累加和，表示无任何排队延迟的端到端等待时间。在 GS（见表 3.9）中，接收者可以使用该值计算端到端排队延迟限制，以及所有端到端延迟限制。

路径带宽：是指在路径上单个连接带宽的最小值。

全局中止位：是一个标志位。发送者创建 Adspec 时，该位置 0。路径上任何不支持 RSVP 的路由器都可将该位置 1，以通知接收者 Adspec 是无效的。

综合服务(IS)网段(hop)计数：在路径上每个支持 RSVP/IS 的路由器都将该值加 1。

路径最大传输单元(PathMTU)：是指在路径上单个连接最大传输单元(MTU)的最小值。

在路径上每个支持 RSVP 的路由器都要更新这些参数，最后将端到端的值提供给接收者。

表 3.9　保证服务(GS)字段

Ctot	Dtot	Csum	Dsum

Ctot：端到端偏差项 C 的总和。

Dtot：端到端偏差项 D 的总和。

Csum：自上次刷新点开始 C 的总和。

Dsum：自上次刷新点开始 D 的总和。

偏差项 C 和 D 是由漏桶流量模型引入的，表示路由器的近似模型与理想模型之间所允许的偏差。在分布树的某些点上，Csum 和 Dsum 被用于刷新处理。

GS 中止位：是一个标志位。发送者创建 Adspec 时，该位置 0。路径上任何 RSVP/IS 但不支持 GS 的路由器都可将该位置 1，以通知接收者 Adspec 是无效的，服务得不到保证。

GS 通用参数头/值：是一个选项。就接收者所希望的 GS 预留而言，如果选择了其中的任何一个，都会忽略 DGP 段所给定的相应值。

Adspec 的 CLS 段包含如下字段。

CLS 中止位：是一个标志位。发送者创建 Adspec 时，该位置 0。路径上任何支持 RSVP/IS 但不支持 CLS 的路由器都可将该位置 1，以通知接收者 Adspec 是无效的，服务得不到保证。

CLS 通用参数头/值：是一个选项。与 GS 段一样，它忽略 DGP 段所给定的特殊服务通用参数。

发送者通过 Path 报文中的 Adspec OPWA 定义端到端服务的预留模式。如果在 Path 报文中没有定义 Adspec 选项，则预留模式只是简单地提交“One Pass”，接收者也就无法确定端到端的服务。Path 报文通过多个中间支持 RSVP 的路由器传输给一个或多个接收者，并形成一个传输路径。在各个路由器上，Path 报文建立起相应的路径状态，并等待接收者 Resv 报文的协商确认，最终将按所需 QoS 确定该路径上的预留资源。

2. 接收者的 Resv 报文

接收者接收到 Path 报文后，从 Sender Tspec 和 Adspec 字段中提取传输特性参数和 QoS 参数，利用这些参数建立起接收者预留说明 Rspec。Rspec 由如下参数组成。

带宽 R：根据 Sender Tspec 参数计算而成。如果得到的 R 值大于 Adspec 中的路径带宽值，则 R 值必须相应地减小。R 值将保存在各个路由器上。

时隙 S：表示端到端延迟限制与应用所需端到端延迟的差值，初始为 0。通过设置 S 值，将为各个路由器在确定局部预留上提供更多的伸缩性，提高端到端预留的成功率。

利用 Rspec 可以创建 Resv 报文。一个 Resv 报文包含如表 3.10 所示的内容。

表 3.10　Resv 报文

预留模式
过滤器说明(Filterspec)
数据流说明(Flowspec)

预留模式：RSVP 的资源预留是针对路由器端口的，路由器使用 Filterspec 和 Flowspec 为相应的端口定义其预留模式，并实施对资源预留的控制。RSVP 可用的预留模式主要有下面三种。

Fixed Filter(FF)：为一个特定发送者建立资源预留状态，由 Filterspec 指定一个特定发送者，合并后的 Flowspec 为该发送者所有预留请求中最大的 Flowspec 值。重新生成的 Resv 报文传送给该发送者的上游节点。

Shared Explicit(SE)：为一个特定的发送者集合建立共享的资源预留状态，由 Filterspec 指定一个特定的发送者集合，合并后的 Flowspec 为这个发送者集合所有预留请求中最大的 Flowspec 值。重新生成的 Resv 报文传送给这些发送者集合的上游节点。

Wildcard Filter(WF)：为所有发送者建立共享的资源预留状态，Filterspec 是通配符，表示可以和任何发送者相匹配，合并后的 Flowspec 为所有预留请求中最大的 Flowspec 值。重新生成的 Resv 报文传送给它的上游节点。

在这些预留模式中，FF 用于单播(点到点通信)、SE 用于组播(点到多点通信)、WF 用于广播(点到所有点通信)场合，其中，SE 和 WF 适合于会议应用。因为在这类应用中，某一时刻只有一个发送者是主动的，应当为发送者的音频和视频流建立资源预留状态，并预留发送带宽。

过滤器说明(Filterspec)：用来标识期望接收的发送者集合，采用与一个 Path 报文中 Sender Template 完全相同的格式。对于 WF 模式，将被忽略。

数据流说明(Flowspec)：用来说明一个期望的服务质量(QoS)，由预留说明 Rspec 和流量说明 TRspec 组合而成。通常，将 TRspec 设置成与 Sender Tspec 相等。

预留确认对象(ResvConf)：可选项，含有接收者的 IP 地址，用于指示接收该预留请求的节点。ResvConf 报文在分布树上向上传播，最终到达该消息接收者，表明端到端预留的成功。

Resv 报文按指定的路径逆向传送给发送者。在每个路由器节点上，Resv 报文对发送者的预留请求给予确认，并且可以和达到同一端口的其他 Resv 报文合并，再传送给由 Phop 指示的上游路由器，直至到达发送者。

3.5.5　路由器对 RSVP 帧的处理

1. Path 报文的处理

在点到多点的传输路径上，中间要通过多个支持 RSVP 的路由器，形成一个分布树。这些路由器都要截获 Path 报文，并检查其有效性。如果发现错误，则要卸下 Path 报文，并用 PathErr 报文通告给上游的发送者，以便让发送者采取适当的动作。

如果 Path 报文是有效的，则路由器将执行下列处理。

(1) 更新发送者路径状态登记项。发送者是用 Sender Template 标识的，如果当前尚无路径状态，则要建立该状态。路径状态包含 Phop、Sender Tspec 以及任意一个 Adspec。Phop 必须存储，以便在分布树上逆向查找转发 Resv 报文的路由。Sender Tspec 提供一个阈值，用于对 Resv 报文中的 Tspec 进行限制。

(2) 设置清除计时器。每个路径状态登记项采用软状态机制，必须使用 Path 报文进行周期性更新。如果在清除计时器规定的时间间隔内没有收到 Path 报文，则会自动删除相应的路径状态登记项，以免死亡的路径状态登记项长期残留在路径状态登记表中。每当收到 Path 报文，要重新设置清除计时器，路径状态信息就不会因超时而被删除。

(3) 生成和转发 Path 报文。根据所存储的路径状态信息生成新的 Path 报文，并沿着分布树向下转发，以刷新下游路由器的路径状态。在下列情况下将创建并发送 Path 报文：一是每当所存储的路径状态发生改变时，将立即创建 Path 报文并发送给下游节点；二是每当更新周期计时器发生超时，将周期地创建 Path 报文并发送给下游节点。

为了维护路径状态信息，路由器的 RSVP 设有两个计时器：清除计时器和更新周期计时器。后者的时间间隔比前者要小若干倍，这样偶尔发生的 Path 报文丢失不会引起不必要的路径状态信息删除。但最好是用最小网络带宽来配置 RSVP 报文，以免因拥挤而丢失数据。如果一直没有 Path 报文的刷新，路径状态信息最终会因超时而被删除。当路径状态登记项被删除时，将产生 PathTear 报文，它沿着和 Path 报文相同的路径传送，沿途各个路由器都要删除该发送者的路径状态信息，并且相关的预留状态也随之被删除。

2. Resv 报文处理

当路由器接收到 Resv 报文后，按其预留模式和流规格(Flowspec)进行如下处理。

(1) 将有效的流规格(Flowspec)提交给路由器的传输控制模块，由传输控制模块实施许可控制和策略控制，以确定是否接受预留。许可控制将单独确定是否有足够容量来满足预留请求，策略控制采用某种策略实施控制，例如，采取某种策略来限制用户的预留的带宽等。

(2) 如果该预留请求被拒绝，则路由器将保持已有的预留状态，并向下游节点发送一个 ResvErr 报文。

(3) 如果该预留请求被接受，则路由器用有效的流规格(Flowspec)和 Filterspec 设置其预留状态。这时，可采用某种规则来改变与该预留请求相关联的预留说明(Rspec)，还可以采用某种规则将该预留请求和其他预留请求相合并，产生新的 Resv 报文。路由器将从所存储的路径状态中获得上游路由器，将 Resv 报文转发给它。

习　　题

1. 现在利用 RTSP/RTP/RTCP 搭建一个流媒体服务器，请详述一下整个流媒体传输的过程。

2. 在 RTP 帧结构中，哪些字段可以表示流媒体的特性？

3. 请详述一下 RTCP 中 SR 报文的每个字段的含义。