视频会议中视频实时传输系统的研究与实现

摘　要

随着网络和流媒体技术的发展及相关协议标准的成熟，基于 IP 的软件视频会议系统以低廉的成本和快捷优质的服务成为目前研究和开发的热点之一，成为视频会议系统的主要发展趋势。然而，IP 网络只能提供尽力而为的服务，不适合视频数据的实时传输。视频数据常常会因为网络拥塞出现马赛克甚至暂停，网络状态的波动也会影响视频的回放效果，视频服务质量很难保证。如何实现视频数据在 IP 网上的实时传输是基于 IP 的视频会议系统要解决的关键问题之一。本文的研究目标就是在现有的网络设施和计算机设备的基础上建立一个纯软件系统来解决这一问题。

本文内容包括三个方面：

1. 方案选取：根据视频数据在 IP 网络上传输的特点，选取 RTP/RTCP 作为传输协议；提出了混合式组播并把它作为多点对多点的传输方式；选取 MPEG-4 作为压缩标准；数据的采集和回放用 DirectShow 技术实现。

2. 具体实现：重点给出了 Video Capture Filter, Video Send Filter, Video Receive Filter 等的设计与实现。Video Send Filter 负责对视频数据进行 RTP 封装并发送到网络；Video Receive Filter 负责一边从网络接收数据，一边将得到的 RTP 包进行重组，并使用异步接收方式、双重缓冲和多线程技术来提高系统的性能。

3. 采用了应用级别的端系统 QoS 控制策略来提高视频的实时传输质量：提出并实现了接收端的拥塞控制、发送端的拥塞控制和差错控制的重传策略。

本人的工作在于：

1. 对基于 IP 网络的视频传输策略进行了研究分析，设计并实现了视频传输过程中所用到的各个 Filter。

2 提出并实现了基于分层排列图的混合式组播，提高了系统的可扩展性。

3. 设计并实现了端系统 QoS 控制策略以保证视频传输的服务质量，它包括接收端的拥塞控制、发送端的拥塞控制和差错控制的重传策略等。

关键词：视频会议系统；混合式组播；拥塞控制；差错控制

Research and Implementation of Video Real-time Transmission System in Conferencing

Abstract

With the development of network technology, streaming media technology and maturity of interrelated protocol standards, software conferencing system based on IP network becomes a focus of research in conferencing system field, due to its inexpensive cost and high quality service. However, the IP network can only provides "best off" services, so it is not appropriate for video real-time transmission. It often appears mosaic even pause when network congestion appears, volatile network states can also reduce the video renderer effect, so the IP network can hardly ensure quality of services for video real-time transmission. Therefore, how to implement video real-time transmission system becomes one of important issues of conferencing system based on IP network. This paper aims to build such a software system on existing network and computer equipment to find a solution to it.

The contents are as follows:

1. Scheme choices: According to the characteristics of video transmission on IP network, RTP/RTCP is chosen to be the transmission protocol; HAG-based mixed multicast is introduced and is used to transmit video data among users; MPEG-4 is chosen to be the code/decode standard; DirectShow technology is used to capture and render the video data.

2. Detailed implementations: The primary task is to design and finish video capture filter, video send filter and video receive filter. Video send filter can encapsulate video data into RTP packets and then transmit them to IP network; video receive filter can regroup the received RTP packets when it receiving data. To improve its performances, the receive filter adopts asynchronous receiving mode, double-cache and multi-thread technology.

3. Control strategies of application layer for video real-time transmission services: The paper introduces congestion control strategy on both receiving sides and sending sides and data packets repeating strategy of error control.

Following jobs is done in this paper:

1. After studying of transmission strategies, the filters in the process of video transmission on IP network are designed and implemented.

2. HAG-based mixed multicast manner is introduced and applied, which improved the expansibility of the system.

3 To ensure video QoS, control strategies on end system are designed in this

paper. It includes: congestion control on receiving sides, congestion control on sending sides and data packets repeating strategy of error control. By virtue of these strategies, QoS of the system is improved to a certian extent.

Keywords: conferencing system; mixed multicast; congestion control; error control

独 创 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含未获得_____（注：如没有其他需要特别声明的，本栏可空）或其他教育机构的学位或证书使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：宋柱芹　　签字日期：　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权学校可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。（保密的学位论文在解密后适用本授权书）

学位论文作者签名：宋柱芹　　导师签字：

签字日期：　年　月　日　　签字日期：2006年 5月20日

学位论文作者毕业后去向：

工作单位：　　电话：

通讯地址：　　邮编

0 前言

视频会议系统(Video Conferencing System)是通过网络通信技术来实现的虚拟会议，使在地理上分散的用户可以共聚一处，通过图像和声音等多种方式交流信息，支持人们远距离进行实时信息交流与共享，开展协同工作的应用系统[43]。视频会议极大地方便了协作成员之间真实和直观的交流，对于远程教学等应用也有着举足轻重的作用。目前，整个视频会议市场规模随着用户群的不断增加而日益扩大。IDC(Internet Data Center，互联网数据公司)预测，视频会议市场销售额在2007年将达到2.74亿美元。

随着网络通信和流媒体技术的发展及相关国标协议标准的成熟，视频会议系统已从高价位专用市场向低价位普通用户市场转移，从硬件视频会议转向软件视频会议，从基于N-ISDN或PSTN的视频会议系统向基于分组交换网的IP视频会议系统过渡。相对于传统的会议模式来说，基于IP的软件视频会议系统以低廉的成本快捷优质的服务成为目前研究和开发的热点之一，这也是视频会议系统的主要发展趋势。

视频实时传输是IP视频会议系统要实现的关键功能之一。视频信号数据量大，需要占用较多的带宽。为保证视频会议的正常进行，需要研究并选择合适的采集、压缩编解码和网络传输方案，还要为之提供一定的QoS。本文的研究目标是在现有的网络设施和计算机设备的基础上建立一个纯软件系统来完成视频数据在IP网上的实时传输。在实施方案的选择上，用DirectShow技术实现发送端的视频数据采集；用MPEG-4对其压缩后以混合式组播的方式传输到接收端；传输协议选择RTP/RTCP；接收端对数据解码之后再用DirectShow技术回放；在QoS方面，在端系统上从拥塞控制和差错控制两方面提供保证。

本软件系统的基本组件是DirectShow的Filter，所以系统的可重用性和扩展性强。在本软件系统上建立的IP视频会议系统在普通的PC机上安装之后，用户就可以通过标准的视音频采集设备实现视音频信号的采集和传输，进行基于IP网络的虚拟视频会议，并取得了良好的效果。

1 绪论

1.1 视频数据在 IP 网络上的实时传输

视频会议的发展趋势是从基于 N-ISDN 或 PSTN 的视频会议系统转向基于分组交换网的 IP 视频会议系统。视频实时传输是视频会议系统关键部分之一，本部分简要介绍视频数据在 IP 网络上传输时的问题、特点、传输方式及 QoS。

1.1.1 视频传输中容易出现的问题

视频数据在 IP 网上传输时会出现以下四个方面的问题：

1. 延时：数据包从发送端到接收端所需要的传输时间。延时越大，对视频效果损害越严重；
2. 延时抖动：即不同数据包延时之间的差别。延时的抖动会造成视频画面不连续，从而导致视频效果下降；
3. 错序：数据包先发后到或者后发先到；
4. 丢包：数据包在网上传输的过程中，因网络拥塞而造成数据包的丢失。

1.1.2 视频实时传输的特点

1. 占较大的网络带宽

以 640*480 像素的视频帧为例，如果以 12 比特的二进制数(亮度 8 比特，色度 4 比特)来表示，按照 30 帧/秒的传输速率，则在一秒内传输的数据量是：(((640*480*30*12)/8bits/byte)/1024bytes/kB)/1024kB/MB = 13.18MB。也就是说，未经压缩的一路视频每秒的数据量超过 13MB。为减轻对通信网络带宽的压力，必须对视频数据进行压缩。

2. 具有突发性

采集到的视频数据经过压缩通过网络传输到接收端。由于数据本身和压缩算法的影响，输出的比特流速率不恒定，即所谓变比特率传输，具有突发性的特点。

3. 允许一定的传输误码，但对延时和延时抖动要求较高。

传统的文件传输不允许数据丢失，但可以有一定的延时和延时抖动。视频传输则相反，能够容忍一定的数据丢失，但必须有较严格的延时和延时抖动才能保证实时性。在视频会议系统中，视频数据的延时应控制在 150ms 以内。

4. 应保持视频与音频的同步

视频数据和音频数据往往分开传输，接收端回放的时候，应采用一些措施保证视音频数据流的同步[36]。

1.1.3 流式传输[5,9]

流式传输的基本原理是把视频数据分割成小块进行连续传输，接收端可以对接收到的部分进行解码，传输和回放可以同时进行。流式传输减少了等待时间，由于在任何一个时间点上只需要保存很少的一部分的视频数据，所以降低了对存储空间的要求。

流式传输技术可分两种：顺序流式传输和实时流式传输。顺序流式传输是顺序下载，在下载文件的同时用户可以观看。但是，用户的观看与服务器上的传输并不是同步进行的，用户必须在一段延时后才能看到服务器上传来的信息。视频会议系统中的视频传输属于实时流式传输方式，在网络的不同用户之间传送的连续视频数据需要实时处理生成低延迟低抖动的连续位流。

1.1.4 传输质量保证

视频传输质量保证可分为空间域和时间域两个方面。空间域质量保证是指要求接收端能接收到一定的数据流量，特别要保证接收到一些重要解码信息，它受到网络传输带宽的影响；时间域质量保证要求接收端解码器能够及时接收到重要的解码信息，从而保证一定的视频恢复质量，它需要考虑不同延时对视频质量的影响。视频会议属于实时应用，要求有较小的延时，可以通过回放缓冲区来平滑延时抖动。

从应用系统的角度来看，视频传输质量保证可以分为基于应用系统(发送端或接收端)的质量保证和基于传输信道的质量保证。前者主要在发送端和接收端进行，发送端通过在对视频源的编码过程中使用一定的抗误码技术来保证接收端的数据纠错与恢复能力，同时通过控制发送速率来保证在信道传输过程中的最小数据丢失。接收端通过使用一定后处理技术(错误隐藏)来保证视频质量恢复；基于传输信道的质量保证主要是通过改进网络的传输技术来保证视频流的传输质量。

1.2 本文的研究目标和主要工作

在基于 IP 的软件视频会议系统项目的研究开发中，本人主要负责视频传输相关技术的研究及相关功能模块的开发设计。本文的目标就是根据国际标准对采集的视频数据进行压缩编解码，并保证视频数据传输的实时性，实现全软件的 IP 视频会议系统的良好的实时性和交互性。本文采用 MPEG-4 视频压缩编码、DirectShow 技术和 WinSock 编程技术完成视频数据在 IP 网上的采集和实时传输。

系统具有投资少并且使用方便，在此基础上可根据需要构建成各种专用系统。完成的主要工作如下：

1. 对基于 IP 网络的视频传输策略进行了研究分析。
2. 提出并实现了基于分层排列图的混合式组播，提高了系统的可扩展性。
3. 设计并实现了视频传输过程中用的各个 Filter。
4. 提出并实现了应用层的端系统 QoS 控制机制，包括接收端的拥塞控制、发送端的拥塞控制和差错控制重传策略。

1.3 本文的内容安排

本文的内容安排如下：

第一章为概述部分，主要介绍了视频数据在 IP 网上实时传输的特点和本文的主要目标和内容；

第二章介绍了 RTP/RTCP 协议，根据 IP 视频会议的特点选取了 UDP 之上的 RTP/RTCP 作为系统采用的传输协议；

第三章介绍了视频压缩编解码标准，选取 MPEG-4 为系统中采用的编解码标准，给出了 MPEG-4 视频流的 RTP 组包方式；

第四章介绍了 IP 组播与应用层组播技术，提出了本系统采用的基于分层排列图的混合式组播；

第五章是系统的实现部分，采用 DirectShow 技术完成视频数据的采集和回放，设计并实现了各个 Filter；在网络上的传输方式采用混合式组播；

第六章在端系统上从拥塞控制和差错控制两方面为视频传输提供了一定的 QoS 保证，给出了实验测试结果。

第七章对本文进行了总结，指出了将来需要深入研究的内容。

2 RTP/RTCP 协议

TCP 通过一系列的机制(重传和滑动窗口等)有效的保证了数据的可靠传输。但是，如果用 TCP 传输海量的实时视频数据则会带来不可预测的延迟，难以解决网络带宽限制问题。目前的 IP 组播技术也不可以使用 TCP 方式实现。RTP/RTCP 就是为解决多媒体数据的实时传输问题而产生的[52]。

包含 SIP 协议和 H.323 协议在内，目前的视频会议系统传输视频音频信息的方式都是采用 RTP/RTCP[53,55]方式。

2.1 RTP/RTCP 协议简介

2.1.1 协议简介

RTP/RTCP 协议由两个紧密相关的部分组成：RTP(Real-time Transport Protocol，实时传输协议)和 RTCP (Real-time Transport Control Protocol，实时传输控制协议)。RTP 是一种提供端对端传输服务的协议，用来支持在单播和多播网络服务中传输实时数据。实时数据的传输由 RTCP 来监视和控制，RTCP 负责接收端和发送端交换控制信息以实现 QoS 监测、会话中成员信息的交换等功能[30]。

RTP/RTCP 协议一般建立在 UDP 协议之上。由应用程序生成的声音和视频数据块被封装在 RTP 信息包中，每个 RTP 信息包被封装在 UDP 消息段中，然后再封装在 IP 数据包中。如图 2-1 所示：

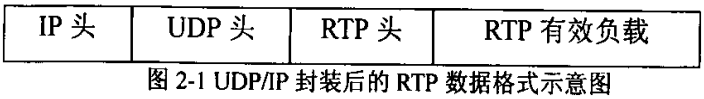

图 2-1 UDP/IP 封装后的 RTP 数据格式示意图

RTP 会话过程要用到两个端口：一个给 RTP，只负责传递有效数据；另一个给 RTCP，负责控制流的传递，用来帮助监视网络流量和阻塞情况，为有效数据传递提供可靠保障。RTP 和 RTCP 两者配合，能以有效的反馈和最小的开销使传输效率达到最佳。所以，RTP/RTCP 特别适合于在 IP 网络上传输实时数据。在视频会议中，要求两组 RTP 和 RTCP，视频数据和音频数据分开传输，使用不同的一组 UDP 端口和组播地址，这样可以允许与会成员按自身需求有选择地接受数据。例如，与会成员带宽不足时，可以选择只接收音频流而不接收视频流。包头中时间信息可以用来保证视频和音频同步。RTP 提供端对端网络传输功能，适合通过组播传送视频音频等实时数据，RTP 没有涉及资源预订和质量保证等实时服务，RTCP 扩充数据传输以允许监控数据传送，提供最小的控制和识别功能。

RTP 协议完成的主要功能如下：

1. 源数据到 RTP 包的转换：在收到的数字信息流的前面加上一个 RTP 数据包头，形成 RTP 包后送交下层。

2. 包的传送：RTP 包放入变长的数据报中，给每个数据报附带地址和控制信息，网络的中间结点检查每个 IP 数据包所带的地址信息，为其选择到目的结点的下一跳路径。

3. RTP 包到数据的转换：数据包到达接收端之后，由 UDP 直接提交 RTP 解包。RTP 对 RTP 包的版本和长度等信息进行验证，解封装合法包，提交给上层进行解码等其它的相关处理。

2.1.2 协议术语

RTP 包：它是一个数据包，由一个固定的 RTP 头、一个贡献源(该贡献源可能为空)以及有效负载的数据共同组成。

RTP 有效负载：它是 RTP 传输时数据包中要传输的实际数据，例如压缩后的音频或者视频数据。

RTP 会话：它是在使用 RTP 的通信者之间建立的一组关联。对每一个通信者来说，会话是由一对特殊的目的传输地址(一个网络地址加上一个 RTP 和 RTCP 的端口对)所定义的。

RTCP 包：它是一个控制包，由一个固定的头以及变长结构化元素组成。

非 RTP 方式：它是为提供某种服务而采用的除 RTP 以外的协议和机制。对视频会议来说，一个会议控制应用程序可能会发布组播地址和密钥，协商加密算法，并且定义在 RTP 有效负载值和有效负载格式之间的动态映射。

端口：传输协议用端口在给定的主机中区分多个目的地址。在 TCP/IP 协议中使用小的正整数来识别端口。RTP 往往靠端口来对一个会话当中的 RTP 和 RTCP 包进行复用。

传输地址：它是一个网络地址和端口的组合，用来识别一个传输层的端点，例如：一个 IP 地址和一个 UDP 端口。包是从源传输地址传到目的传输地址的。

同步源(SSRC)：它是 RTP 包的源，由 32 位的 SSRC 数字标识符来识别，由 RTP 的头文件所携带。从一个同步源出来的所有包构成相同的时间和空间序列。在接收端可以用同步源为包分组，从而进行回放。SSRC 标识符是一个随机选择的值，在一个特定的 RTP 会话中它是唯一的。

混合器：它是一个中间系统，从一个或多个源接收 RTP 包，将这些包以某种方式组合成一个新 RTP 包。从一个混合器出来的所有数据包都要用混合器作为它们的同步源来识别。混合器在保留原有格式的基础上将不同源负载组合成一个流。

贡献源(CSRC)：它是对 RTP 混合器产生的组合流有贡献的 RTP 包的源。

终端系统：它是一个应用程序，产生能在 RTP 包中传送的内容，或者是可以“消耗”接收到的 RTP 包的内容。终端系统的作用相当于一个特定的 RTP 会话中的一个或多个同步源，通常的情况是一个同步源。

转换器(翻译器)：它是一个中间系统，用完整的同步源标识符来转发 RTP 包。转换器可以将负载从一种语法转换(编码)为另一种语法，比如说将一个高带宽需求的压缩格式转换成一个低带宽格式，这样可以保证转换器两边的终端可以正常通信。

监视器：它是一个应用程序，在一个 RTP 会话中接收由发送端发送的 RTCP 包，尤其是接收报文、估算服务质量、进行错误诊断以及长期的静态统计。监视器函数可能被创建到参加会话的应用程序中，也可以作为独立的应用程序，既不参与也不发送或接收 RTP 数据包。

2.2 RTP/RTCP 数据包格式

2.2.1 RTP 数据协议包头

RTP 数据协议包头如图 2-2 所示：

版本号	填充位	扩展位	CSRC数	标记	载荷类型	序列号	时间戳	SSRC标志	CSRC标志	数据
2位	1位	1位	1位	1位	7位	16位	32位	32位	变长	变长

图 2-2 RTP 协议包头格式图

1. 版本(V)：2 位，标识 RTP 版本。

2. 填充位(P)：1 位，指定在有效载荷后是否补零填充。如果设置了填充位，那么在包的结尾包含一个或者多个附加填充字节，这些字节不是 RTP 包的一部分，填充字节的最后一个八位字节的值指示要求被忽略的字节数。

3. 扩展位(X)：1 位，如果设置了扩展位，固定的头(前 12 字节)后面要跟一个扩展的头。

4. CSRC 数(CC)：4 位，CSRC 数包含出现在固定的头后面的 CSRC 数目。

5. 标记位(M)：1 位，标记位由一个序(profile)来解释。对于不同的 RTP 负载类型将有不同的解释。

6. 负载类型(PT)：7 位，定义标志 RTP 负载类型，标志 RTP 载荷格式并决定其解释，设置指定载荷类型代码对载荷格式的静态映射，决定应用程序解析包的方法。

7. 序列号：16 位，其值由 RTP 数据包自动增加，用于接受方诊断包丢失和恢复包序列，序列的初始值随机生成。每发送一个 RTP 信息包序列号就加 1，接

收端可以用它来检查信息包是否有丢失以及按序列号来处理信息包。

8. 时间戳(Timestamp)：32 位，随机生成，可用于视音频流的同步。它反映 RTP 数据信息包中数据块(音频、视频等)第一个字节的采样时刻。该采样时间必须取自随时间单向线性增长的时钟，以便支持同步和抖动计算。其时钟频率取决于 RTP 帧的荷载类型，时钟的精度必须满足同步精度和测量分组的到达时延抖动的需要。接收端利用时间戳可以实现数据流的同步回放，包括同一数据流的流内同步和不同数据流的流间同步，完成对数据包的重组，并按照正常的速率回放数据。对于一些大的数据块，一个数据块被分成多个 RTP 包，它们的时间戳相同。仅靠时间戳不足以恢复数据包的顺序，RTP 利用提供的序列号(Sequence Number)来恢复数据包的顺序，实现包丢失检测，为网络的实时传输提供网络拥塞等信息。

9. 同步源标识(SSRC)：32 位。此字段随机选择，用来标识 RTP 包流的起源，其目标是保证同一个 RTP 会话在任两个同步源的 SSRC 标志不相同。SSRC 提供了对实时传输交换性的支持，可以帮助接收端利用发送端生成的唯一的数值来区分多个同时的数据流，得到数据的发送源。如：在 IP 网络视频会议中，通过源鉴定可以得知哪一个用户在发言，并且可以得到发言人的信息。

10. 贡献源标志(CSRC)：0 到 15 项，每项 32 位，表示此分组的负载字段中包含的贡献源。贡献源的个数由 CC 字段指定。如果有多于 15 个贡献源，则只能表示 15 个贡献源。它由混合器插入，其值等于贡献源的 SSRC 标志。

2.2.2 RTCP 包格式

RTCP 数据包分为 5 种：

1. SR：发送端报文，由当前活动的发送端产生，它包含发送端信息、媒体间的同步信息、数据包累计和发送的字节数等。

2. RR：接收端报文，由非活动的发送端产生，主要是数据传输的接收质量反馈。它包括接收的最大数据包数、丢失的数据包数、发送端和接收端的往返延迟等。

3. SDES：源描述，包含源描述的信息，它包括 CNAME 等。

4. BYE：表示结束参与。组播组的成员是动态的，成员可以随时加入和离开组。因此，在任何时候都需要了解组成员的情况以及他们接受数据的情况。为了实现这个目的，每个接受者定期广播一个接收报文并加上自己的名称表示自己的存在；当成员离开时，发送一个 BYE 类型的 RTCP 包。

5. APP：具有特定功能的应用函数。

现在对它们分别详细介绍：

1. SR 和 RR

RTP 报文的接收端可以利用 SR 和 RR 提供的有关数据接收质量的统计信息，具体选用 SR 还是 RR 要看该接收端是否同时也是 RTP 报文的发送端。SR 报文与 RR 报文的区别在于前者包含 20 字节的发送端信息。SR 和 RR 都可包括零至多个接收报文块，最大可有 31 个接收报文块嵌入在 SR 或 RR 包中。

发送端报文的结构定义如图 2-3 所示：

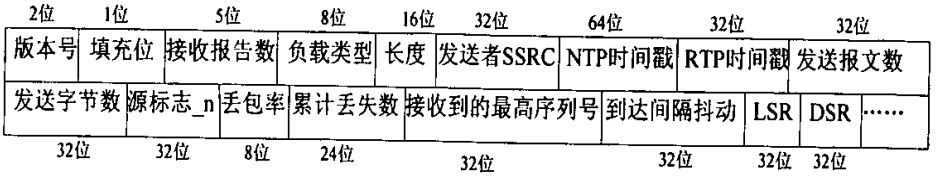

图 2-3　发送端报文的结构图

版本(V)：2 bit，用于协议鉴别。

填充(P)：1 bit，如果设置填充位，则此 RTCP 包结尾处包含一些附加的填充字节，这不属于控制信息的内容。填充位在某些固定块大小的加密算法中使用。在组合包中，填充位只出现在最后一个 RTCP 包中，因为组合包是作为一个整体来压缩的。

接收报文数(RC)：5 bit，在 SR 中包含的 RR 数目。

负载类型(PT)：8 bit，报文类型，以二进制表示。用十进制的 200 代表 SR。

长度(length)：16 bit，报文长度，包括包头和任何填充位的长度。

发送端同步源标志(SSRC of sender)：32 bit，源同步码，用以标识本次会话。

以上是固定头部分。下面是发送端信息，为 20 字节长度。

NTP 时间戳(NTP timestamp)：64 bit，绝对时间戳。在测量环路时延时可在对方的 RR 报文中带回；如果发送端不具有绝对时钟的能力，则可以用会话开始时间作为时钟 0 点或将此域置 0。

RTP 时间戳(RTP timestamp)：32 bit，以 RTP 时间戳为基准。

发送的报文数(sender's packet count)：32 bit，从会话开始后发送端总共发送的 RTP 报文数目。

发送的字节数(sender's octet count)：32 bit，从通话开始后发送端总共发送的有效负载的数目。随后描述的是一个或多个 RR 报文块。

源标志_n (SSRC_n)：32 bit，源同步码，用以标识此 RR 块所从属的会话，这里一般 n 等于 1。

丢包率(fraction lost)：8 bit，从上一个 SR 或 RR 报文发送后的丢包率，其值为接收端在这段时间内期待的 RTP 报文与所收到的 RTP 包数目的差值再除以它所期待的 RTP 报文的数目。

累计的包丢失数(cumulative number of packets lost)：24 bit，存在的累计的包丢失数。

接收到的扩展的最高序列号(extended highest sequence number received)：32

bit，低 16 位是收到的 RTP 包中的 sequence number 的最新值，高 16 位标识收到的 RTP 报文的 sequence number 的循环次数。

到达间隔抖动(interarrival jittery)：32bit，时延抖动。

上一 SR 报文时间戳(LSR)：32 bit，收到的最近一个 SR 报文的 NTP 时间戳的中间 32 位。

自上一 SR 的时间(DLSR)：32bit，收到上一 SR 报文与此次已送报文之间的时间，以 1/65536s 记。如果还没有收到任何 SR 报文，此值置 0。

2. SDES：源描述符 RTCP 包。

SDES 的基本格式如图 2-4 所示：

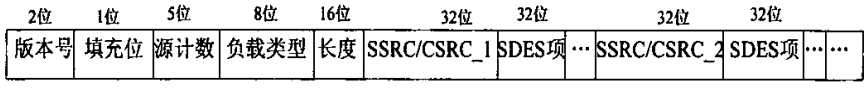

图 2-4　SDES 的基本格式图

版本(V)、填充(P)、长度：相同于 SR 包中的描述。

包类型(PT)：8 位，识别 RTCP SDES 包。

源计数(SC)：5 位，包含在 SDES 包中的 SSRC/CSRC 块数量。

各种类型源描述项内容描述：

CNAME：规范终端标识 SDES 项。发生冲突或重启程序时，随机分配的 SSRC 标识可能发生变化，需要 CNAME 项提供从 SSRC 标识到仍为常量的源标识的绑定。

NAME：用户名称 SDES 项，这是用于描述源的真正的名称，可以是用户想要的任意形式。

EMAIL：电子邮件地址 SDES 项，邮件地址格式由 RFC822 规定。

PHONE：电话号码 SDES 项。

LOC：用户地理位置 SDES 项，根据应用，此项具有不同程度的细节，但格式和内容可用设置指示。

TOOL：应用或工具名称 SDES 项，是一个字符串，表示产生流的应用的名称与版本。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 值在连接期间仍保持常数。

NOTE：通知/状态 SDES 项。

PRIV：专用扩展 SDES 项。该项用于定义实验或应用特定的 SDES 扩展，它包括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。

3. BYE：断开 RTCP 包

混合器接收到 BYE 包就转发该 BYE 包，但不改变 SSRC/CSRC 标识。如混合器关闭，它也应该发出一个 BYE 包，列出它所处理的所有源，而不只是自己

的 SSRC 标识。作为可选项，BYE 包可包括一个 8 位八进制计数，后跟很多八进制文本，表示离开原因。

4. APP：定义应用的 RTCP 包

APP 包用于开发新应用和新特征的实验，不要求注册包的类型值。带有不可识别名称的 APP 包应被忽略掉。

RTCP 协议的设计目的是与 RTP 协议共同合作，为顺序传输数据包提供可靠的传送机制，并对网络流量和阻塞进行控制。RTCP 的基本做法是周期性地和所有参与者进行通信，采用和 RTP 数据包分配传送的相同机制来发送控制包。每个 RTCP 包的前一部分是固定的，类似于 RTP 的数据包；后面的结构根据包的类型不同长度而不同，但总是 32 位的整数倍。其长度在固定部分的长度域中标明。

多个 RTCP 包不需要任何分隔符而连接起来形成一个 RTCP 组合包，在低层协议用单一包发送出去。由于需要低层协议提供整体长度来决定组合包的结尾，在组合包中没有单个 RTCP 包的显式计数。组合包中每个 RTCP 包可独立处理，不需要根据包组合顺序。但为了执行协议功能，规定如下约束条件：

1. 只要带宽允许，接收统计(在 SR 或 RR 中)应该经常发送，因此每个周期发送的组合 RTCP 包应包含报文包。

2. 新接收端需要接收 CNAME，并尽快识别源，开始与联系媒介进行同步，因此每个包应该包含 SDES CNAME。

3. 出现在组合包前面的是包类型数量，其增长应该受到限制，以提高常数位数量，提高 RTCP 包对错误地址 RTP 数据包或其他无关包成功确认的概率。

因此，所有 RTCP 包必须至少以两个包组合形式发送。推荐格式如下：

加密前缀(Encryption prefix)：仅当组合包被加密，才加上一个 32 位随机数用于每个组合包发送。

SR 或 RR：组合包中第一个 RTCP 包必须是一个报文包以方便头的确认。即使没有数据发送，也没有接收到数据，也要发送一个空 RR。

附加 RR：如果报文统计源数目超过 31，在初始报文包后应该有附加 RR 包。

SDES：包含 CNAME 项的 SDES 包必须包含在每个组合 RTCP 包中。

BYE 或 APP：除了 BYE 应作为最后一个包发送，其它的 RTCP 包类型可以以任意顺序排列，包类型出现可不止一次。如果组合包整体长度超过网络路径最大传输单元，则可分成多个较短组合包用低层协议以单个包形式发送。每个组合包必须以 SR 或 RR 包开始。

通过上述控制信息数据包，RTCP 可以完成下列控制功能：

1. 提供数据发布的质量反馈

RTCP 包提供监控 QoS 所必须的信息，向应用程序提供数据发布质量的反馈。这些参数包括：包丢失率、抖动、延迟、接收到的最大顺序号等。这个控制信息

对发送端、接收端和第三方监视都很有用。发送端可以根据接收端的反馈信息调整数据的发送，而接收端可以得到网络阻塞的情况。另外，在 IP 组播机制的支持下，RTCP 的这一功能允许监视者通过接收反馈报文来推断网络的服务质量。提供反馈的功能由 SR 和 RR 共同完成。

2. 源标识

RTCP 为每一个 RTP 源分配一个永久的传输层标志 CNAME(规范名字)。虽然 RTP 数据包中的 SSRC 标志可以区分同一会话中的不同码流，但当 RTP 发现不同码流的 SSRC 标志发生冲突或应用重新启动时，SSRC 的值就被更改，接收端需利用 CNAME 而非 SSRC 来跟踪每一个会话参与者。另外借助于 CNAME，接收端可以关联来自同一用户但经由不同 RTP 会话传来的码流以完成特定的任务，如视频流和音频流的同步。CNAME 由 RTCP 的 SDES 报文所携带。

3. 调节报文的发送速率

RTCP 给出了关于调整控制报文发送速率的方法。RTP 用户能够通过接收来自其它用户的 RTCP 报文了解会话参与者的数目。RTCP 利用该数值参数为每个会话参与者计算控制报文的时间间隔，以达到 RTCP 所占的通信带宽不超过会话带宽 5%的目的。由于这些报文通过组播到达各方，所以每个用户可以记录已经分发的报文数目，从而适当调整自己的报文时间间隔，避免出现所有用户同时发送报文而产生高峰负载。

4. 可靠性和安全性

RTP 的设计目的是传输实时数据流，它不提供有关错误检测和包顺序监控的机制。RTCP 协议在设计上考虑到安全功能，支持对数据加密和身份鉴别认证功能。RTCP 协议将控制包周期地发送给所有连接者。

5. 传送最小连接控制信息，如参加者标志(可选功能)。

2.3 本文传输协议的选取

与 UDP 相比，TCP 协议不适合视频实时传输。其原因主要表现在它的重传机制和报文头构成上：在 TCP/IP 协议中，当发送端发现数据丢失时，它将要求重传丢失的数据包。根据 TCP/IP 的快速重传机制，这将需要三个额外的帧延迟。接收端等待重传数据的到来会造成视频延迟、断点和凝固等问题；TCP 的报文头为 40 个字节，而 UDP 的报文头仅为 12 个字节，并且 TCP 的报文头不能提供时间戳和编解码信息，而这些信息恰恰是接收端的应用程序所需要的。从这点来说 TCP 也不适合传输实时的视频数据。

视频流在时间轴上的连续性要求网络的实时传输及高带宽，同时又允许传输中存在一定的数据错误率及数据丢失率。RTP 本身没有一种独立传输能力，它必须与低层网络协议结合才能完成数据的传输服务。视频在时间轴上的相关性不

强，而数据的实时性要高于其可靠性。所以，在 UDP 之上利用 RTP/RTCP 协议对视频流进行封装，打包和同步，可以使视频数据的网络传输延时达到最小。UDP 是一种无连接的数据报投递服务，没有 TCP 那么可靠，并且无法保证实时视频传输业务的服务质量，需要 RTCP 实时监控数据的传输和服务质量。但是，UDP 的传输延时低于 TCP，它能与音频和视频流很好地匹配。

由于以上原因，本文采用 RTP/RTCP/UDP 作为视频数据的传输协议。

2.4 本章小结

本章介绍了 RTP 和 RTCP 协议，根据视频传输的特点，选定 RTP/RTCP/UDP 作为本系统的传输协议。

3 视频压缩编码

数字视频信号的数据大，必须对其进行压缩才能节省存储设备，才能有效利用信道带宽。视频信号的压缩理论主要依靠两点：视频信号本身具有时间、空间和统计相关性；人眼视觉系统对视频的主观感知允许一定的失真。视频压缩算法利用这两点可大幅度地压缩数据量，同时保证较好的主观图像质量。

3.1 视频压缩编码标准的发展

多个国际组织制订了一系列视频编码国际标准。目前主要的视频编码标准有国际电信联盟(ITU, International Telecommunication Union)制订的H.26x系列和国际标准化组织的运动图像专家组(MPEG, Moving Picture Experts Group)制定的MPEG系列。

H.261是规范ISDN网上的视频会议和可视电话应用中的视频编码技术。它结合了可减少时间冗余的帧间预测和可减少空间冗余的DCT变换的混合编码方法，在帧间编码时采用基于16×16的宏块和整像素精度的运动估计，而在帧内编码时采用8×8数据块的DCT运算。由于有效地压缩了视频序列在时间和空间上的冗余度，H.261具有较高的压缩比。H.263是低码率图像压缩标准，通过利用帧间预测去除时间上的冗余度，利用变换编码减少预测余量信号空间冗余度。解码器具有运动补偿能力，最后采用变长编码形成传输码流。H.263+和H.263++都是对H.263的扩展。H.263针对的带宽主要为甚低码率(小于64Kbps)，高带宽和高质量的视频压缩不在该标准覆盖范围内。H.264在保留运动补偿和变换编码技术的基础上，加入了如类DCT整数变换、CAVLC、CABAC等新技术，进一步提高了编码算法的压缩效率和图像回放质量。

MPEC-1标准用于数字存储媒体的活动图像和相应的音频编码，该标准的制定使得基于CD-ROM的数字视频以及MP3等成为可能。MPEG-2标准用于数字视频广播、高清晰度视频、数字视盘等运动图像及其伴音的编码。MPEG-4的编码系统是开放的，可随时加入新的有效算法模块。该标准适用于多媒体Internet、视频会议和视频电话等个人通信。MPEG-4支持误码信道传输下的鲁棒性，提供了更好的同步和误码恢复机制。

其中，H.264和MPEG-4是当前视频压缩领域的主流编码算法。

3.2 MPEG-4 标准简介[53]

1999 年 1 月 MPEG-4 第 1 版正式公布，1999 年 12 月第 2 版公布。标准中规定适应的三段比特率范围分别是低于 64kbit/s、64kbit/s～384kbit/s 和 384kbit/s～4Mbit/s。

MPEG-4 标准与 MPEG-1 和 MPEG-2 标准最根本的区别在于 MPEG-4 采用基于对象的方法，可以支持基于对象的互操作性。为适应通用访问，MPEG-4 标准加入了面向功能的传送机制，其中的错误鲁棒性、错误恢复的处理和速率控制等功能使编码能适应不同信道的带宽要求。MPEG-4 编码系统是开放性质的，可随时加入新的编码算法模块，并可根据不同的应用需求配置解码器。

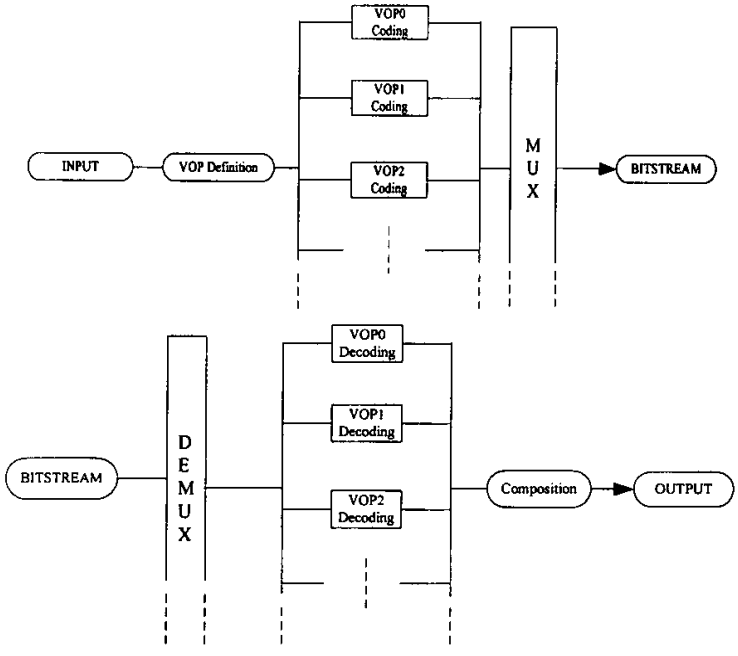

图 3-1 MPEG-4 编解码原理图

如图 3-1 所示，MPEG-4 编解码原理如下：

1. 基本思想

MPEG-4 是基于图像内容的第二代视频编解码方案，并将基于合成的编码方案也结合在标准中。它根据图像的内容将图像分割成不同的视频对象 VO(Video Object)，在编码过程中对前景对象和后景对象采用不同的编码策略，对于人们所关心的前景对象，则尽可能的保持对象的细节及平滑，而对人们不大关心的后景对象采用大压缩比的编码策略。

2. 编解码的数据结构

MPEG-4 按照五个层次组织要编码的图像，从上至下依次为：视频段 VS(Video Session)、视频对象、视频对象层 VOL(Video Object Layer)、视频对象

组层 GOV(Group of Video Object Plane)和视频对象平面 VOP(Video Object Plane)。

在 MPEG-4 中，VO 主要被定义为画面中分割出来的不同物体，每个 VO 由三类信息来描述：运动信息、形状信息和纹理信息。VO 的构成依赖于具体应用和系统所处环境。在要求超低比特率的情况下，VO 可以是一个矩形帧(即传统 MPEG-1 中的矩形帧)，从而与原来的标准兼容；对于基于内容的表示要求较高的应用来说，VO 可能是场景中的某一物体或某一层面；VO 也可能是计算机产生的二维或三维图形。

3. VOP 编码器结构

编码器主要由两部分组成：形状编码和传统的运动纹理编码，其中形状编码是 MPEG-4 在编码任意形状的 VOP 时所必须的。

4. MPEG-4 的编解码流程及框架

MPEG-4 的编码流程：第一步是 VO 的形成，先要从原始视频流中分割出 VO，之后由编码控制机制为不同的 VO 以及各个 VO 的三类信息分配码率，之后各个 VO 分别独立编码，最后将各个 VO 的码流复合成一个位流。其中，在编码控制和复合阶段可以加入用户的交互控制或由智能化的算法进行控制。现在的 MPEG-4 包含了基于网格模型的编码和 Sprite 技术。解码流程正好相反，在此不作赘述。

3.3 H.264 标准简介[56]

2001 年底，联合视频小组(Joint Video Team, JVT)。JVT 在 2003 年 5 月推出了 ITU H.264 标准(也称 MPEG4 Part10 AVC 标准)。H.264 继承了 H.263 和 MPEG1/2/4 视频编码协议的优点，在保留运动补偿和变换编码技术的基础上，加入了如类 DCT 整数变换、CAVLC 和 CABAC 等新技术，进一步提高了编码算法的压缩效率和图像回放质量。在人眼主观感受相同的情况下，H.264 较之 H.263 的编码效率提高了 50%左右。

H.264 标准是 H.263 的进一步改进和升级，同时又是 MPEG-4 的第十部分，它很好的解决了不同视频压缩编码的系统之间的兼容性问题。和以前的标准相比，H264/AVC 支持从低比特率要求的移动或拨号设备到有娱乐质量标准要求的电视服务和 HDTV 等，标准中还包含了用于消除差错的工具，便于压缩视频在误码和丢包多发环境中传输。同时，H.264 的码流结构网络适应性强，能够很好地适应 IP 和无线网络的应用。

H.264 标准的主要技术特点为：

1. 分层设计

H.264 在算法上分为两层：视频编码层(Video Coding Layer, VCL)和网络提取层(Network Abstraction Layer, NAL)。VCL 负责高效的视频内容表述，包括基于块的运动补偿混合编码和一些新特性，它通过时域、空域预测和变换编码来完成对视频信息的压缩。NAL 负责根据下层网络的分段特性来封装数据。包括组帧、发送逻辑信道的信令、利用同步定时信息或序列结束信号等将与网络相关的信息从视频压缩系统中抽象出来，使网络层对于 VCL 是透明的。VCL 和 NAL 之间定义了基于分组方式的接口，高效编码和良好的网络适应性分别由 VCL 和 NAL 完成。

2. 运动估计与补偿

H.264 支持 1/4 或 1/8 像素精度的运动矢量。在 1/4 像素精度时可使用 6 抽头滤波器来减少高频噪声；对于 1/8 像素精度的运动矢量，可使用更为复杂的 8 抽头的滤波器。在进行运动估计时，编码器还可选择“增强”内插滤波器来提高预测的效果。

在 H.264 的运动预测中，一个宏块可以分为不同的子块，形成 7 种不同模式的块尺寸。这种灵活的多模式和细致的划分，更切合图像中实际运动物体的形状，大大提高了运动估计的精确度。

3. 整数 DCT 变换

H.264 与先前的标准相似，对残差数据采用基于块的变换编码，变换编码可以去除原始图像的空间冗余，使图像能量集中在一小部分系数上，这样可以提高压缩比、增强抗干扰能力。H.264 中的变换是整数操作而不是实数运算，其过程和 DCT 基本相似。这种方法的优点在于：在编码器中和解码器中允许精度相同的变换和反变换，便于使用简单的定点运算方式。这样，不但变换计算量比较小，而且在运动物体边缘处的衔接误差也大为减小。

H.264 为了提高码率控制的能力，量化步长变化的幅度控制在 12.5%左右，而不是以不变的增幅变化。变换系数幅度的归一化被放在反量化过程中处理以减少计算的复杂性。为了强调色彩的逼真性，对色度系数采用了较小量化步长。

4. 熵编码

H. 264 中熵编码有两种方法：UVLC 和 CABAC。UVLC 使用一个长度无限的码字集，设计结构非常有规则，用相同的码表可以对不同的对象进行编码。这种方法很容易产生一个码字，而解码器也很容易地识别码字的前缀，UVLC 在发生比特错误时能快速获得重同步。CABAC 是可选项，其编码性能比 UVLC 稍好，但计算复杂度也高。

5. 帧内预测

在 H.264 中，当编码 Intra 图像时可用帧内预测。对于亮度块可分为两种方式：Intra-4×4 和 Intra-16×16。对于图像中比较平坦的部分采用 Intra-16×16，对于需要进行细化的图像部分采用 Intra-4×4。在进行预测前对与当前宏块相邻的

左、右重构块进行分类，然后根据不同的分类，以已编码并重建的块作为参考，选择不同的预测模式对当前块进行预测。这种帧内预测是在空间域上进行的，可以除去相邻块之间的空间冗余度，取得更有效的压缩。

3.4 本文视频压缩方案的选取

当前视频会议系统一般可以采用 MPEG-4 和 H.264 视频编解码方式。H.264 是最新的编解码方式，可以在 1Mbps 左右的带宽下实现高清晰的图像视频。H.264 获得优越性能的代价是运算复杂度的大幅增加，与 MPEG-4 相比，H.264 对硬件的要求很高。MPEG-4 编解码方式可以在 1.5Mbps 带宽下实现 4CIF 的高清晰视频会议，可以满足当前的主流带宽要求。基于软件的视频会议系统，基本上都是采用 MPEG-4 作为视频编解码标准。

根据目前大多数用户的硬件条件和各种编码标准特点，本文的视频压缩编解码选用 MPEG-4 标准。MPEG-4 具有以下特点：

压缩率高：MPEG-4 算法对视频数据的压缩率占有明显优势，能有效节省视频会议中宝贵的带宽资源。

对不同带宽的适应能力：MPEG-4 算法具有良好的带宽适应性和视频压缩率，适应从 56Kbps 拨号上网到宽带的网络环境。这就保证可以满足了不同用户的需求。用户可以根据自己的实际情况，设置好相应的带宽，这样就可以最大限度的利用带宽资源，从而达到最好的视频会议效果。

视频压缩算法的发展方向：MPEG-4 应用已经越来越普遍，它已经成为了视频产品的主要发展方向，并成为互联网实时通信和流媒体压缩的事实标准。

3.5 本章小结

本章介绍了 H.264 和 MPEG-4 压缩标准，结合目前大多数用户的硬件条件和视频传输的特点，选定 MPEG-4 为系统的视频编解码标准。

4 组播技术研究

在 IP 网络中，当音频和视频等数据传送的地址是网络中的多个用户时，可以采用的传输方式有以下四种：

1. 单播(Unicast)：单播方式为每个用户单独建立一条数据传送通路。

2. 广播(Broadcast)：广播方式把数据传送给网络中的所有用户(不管它们是否需要)。

3. IP 组播(IP Multicast)：在路由器上实现组播功能。发送的同一数据在物理链路中只传输一次，数据在网络三层交换结点(路由器)处进行复制和分发，从而减少数据在网络传输中的冗余以节约带宽。

4. 应用层组播(Application Level Multicast，ALM)：在应用层实现组播功能。应用层组播将组播功能从路由器转移到端系统，由端系统完成如组播成员管理、数据包复制和分发等功能。它需要将参与组播的成员组织成叠加网[21](overlay network)。叠加网包括数据拓扑和控制拓扑：数据拓扑也称叠加树(overlay tree)，由结点与结点间建立的单播链路形成，主要用来传输数据；控制拓扑主要用来发布控制信息，用于树分割恢复、失效成员检测和信息交换等方面。

视频会议传输数据量大，对带宽要求高。视频数据的传输若用单播传输机制实现，需要给与会成员发送大量内容相同的数据包，资源消耗大、效率低、浪费大量带宽资源，并且这种情况会随着接收端的增多而加剧。而广播存在资源无谓占用，对网络可能意味着传输风暴。若用广播传输机制实现，不仅极大浪费带宽，而且不利于会议信息的安全和保密。在网络资源的有效利用方面，组播技术可以在现有网络带宽条件下支持更大规模的应用，能够较好的解决骨干网络及局域网出口带宽的瓶颈问题，能够有效地解决多点对多点数据通信的网络带宽问题。因此，相对于单播和广播方式来说，组播是最适合于视频会议系统的一种传输方式。

目前人们提出两种组播技术：网络层的 IP 组播和应用层的应用层组播。

4.1 IP 组播

1988 年 Deering 提出了 IP 组播(IP multicast)。IETF RFC 1112 对 IP 组播业务提供的方式和形式进行了描述和定义，被看成是 IP 组播的标准业务模型定义。IP 组播具有网络利用率高、传输速度快和实时性强等优点。

4.1.1 IP 组播技术的基础知识概述

1. IP 组播技术的概念

IP 组播是指在 TCP/IP 网络上实现的组播)，是对标准 IP 网络层协议的扩展。它的基本方法是：当某一个主机向一组主机发送数据时，它不必将数据向每一个主机都发送，只需将数据发送到一个特定的预约的组地址，所有加入该组的人均可以收到这份数据。这样对发送者而言，数据只需发送一次就可以发送到所有接收者，避免了广播风暴的产生，大大减轻了网络的负载和发送者的负担。IP 组播是多点对多点通信中节省网络带宽的一种有效方法。

2. IP 组播地址和组播组

IP 组播必须依赖于 IP 组播地址，在 IPv4 中，从 224.0.0.0 到 239.255.255.255 的地址被划分为局部链接组播地址、预留组播地址和管理权限组播地址三组。

1) 局部链接组播地址：从 224.0.0.0 到 224.0.0.255，为路由协议和其它用途保留的地址，路由器并不转发属于此范围的 IP 包；
2) 预留组播地址：从 224.0.1.0 到 238.255.255.255，可用于全球范围或网络协议；
3) 管理权限组播地址：从 239.0.0.0 到 239.255.255.255，可供组织内部使用，不能用于 Internet，可限制组播范围。

使用同一个 IP 组播地址接收组播数据包的所有主机构成一个主机组，称为组播组。组播组的成员是随时变动的，一台主机可以随时加入或离开组播组，组播组成员的数目和所在的地理位置也不受限制，一台主机也可以同时属于不同的组播组。

3. 组播分布树

为了在组播网中组播数据，用组播分布树来描述 IP 组播在网络中传输的路径。组播分布树连接了对应组播组中所有主机，可分为有源树和共享树。

有源树是以组播源作为有源树的根，有源树的分支形成通过网络到达接收主机的分布树。由于有源树以最短的路径贯穿网络，所以也常称为最短路径树。

共享树以组播网中某些可选择的组播路由中的一个结点作为共享树的公共根。共享树又可分为单向共享树和双向共享树。

4.1.2 IP 组播路由及其协议

历经 20 多年的研究和发展，IP 组播已经形成了较为完整的组播协议体系，包括组播主机和网络的交互协议、组播路由协议和组播的地址管理协议等。

1. IP 组播路由的基本类型

根据组播组成员在网络中的分布，IP 组播路由协议可分为两种基本类型。第一种类型假设组播组成员在网络中是稀疏分散的并且网络不能提供足够的传输带宽，这种模式被称为稀疏模式。稀疏模式组播路由协议必须依赖于具有路由选择能力的技术来建立和维持组播树。第二种类型假设组播组成员密集地分布在网络中，也就是说，网络大多数的子网都至少包含一个组播组成员，而且网络带宽足够大。这种被称作“密集模式”的组播路由协议依赖于广播技术来将数据“推”向网络中所有的路由器。密集模式路由协议包括距离向量组播路由协议、组播开放最短路径优先协议和密集模式独立组播协议等。

2. 稀疏模式组播路由协议

1) 基于核心树的组播协议(CBT)

CBT 协议只构建一棵共享树给组中所有成员共享。整个组播组的组播数据都在这个共享树上进行收发而不论发送源有多少或者在什么位置。这种共享树的使用能够极大的减少路由器中的组播状态信息。

2) 独立组播稀疏模式协议(PIM-SM)

独立组播协议(PIM)是一种标准的组播路由协议，并能够在 Internet 上提供可扩展的域间组播路由而不依赖于任何单播协议。PIM 有两种运行模式，一种是密集分布组播组模式，另一种是稀疏分布组播组模式，前者被称为独立组播密集模式协议(PIM-DM)，后者被称为独立组播稀疏模式协议(PIM-SM)。

PIM-SM 围绕一个被称为集中点(RP)的路由器构建组播树，接收端在集中点能查找到新的发送源。PIM-SM 中的独立的接收端可以选择构建共享树还是最短路径树。

PIM-SM 协议先为组播组构建一个组共享树，它由连接到集中点的发送端和接收端共同构建。共享树建立以后，一个接受者(实际上是最接近这个接收端的路由器)可以选择通过最短路径树改变到发送源的连接。

3. 密集模式协议

1) 距离向量组播路由协议 (DVMRP)

DVMRP 是第一个支持组播功能的路由协议。DVMRP 为每个发送源和目标主机组构建不同的组播树，它是以组播发送源作为根，以组播接受目的主机作为叶的最小扩展分布树。对子网中密集分布的组播组来说 DVMRP 能够很好的运作，但 DVMRP 不能支持大型网络中稀疏分散的组播组。

2) 组播开放最短路径优先(MOSPF)

MOSPF 依赖于单播路由协议 OSPF，在一个 OSPF/MOSPF 网络中每个路由器都维持一个最新的全网络拓扑结构图。这个“链路状态”信息被用来构建组播树。它将数据包在最小开销路径上进行传送。

3) 独立组播密集模式协议(PIM-DM)

PIM-DM 有点类似于 DVMRP，这两个协议都使用了反向路径组播机制来构建组播树。其不同在于：PIM 不依赖网络中的任何单播路由协议而 DVMRP 依赖于某个相关的单播路由协议机制；PIM-DM 比 DVMRP 简单，PIM-DM 倾向于简单性和独立性，有时不惜增加数据包复制而引起额外开销。

4.1.3 IP 组播技术的优势与局限

IP 组播技术在多点视频数据传输方面具有很大的优势。比如：可以减轻系统和网络的负担；提高 CPU 资源和网络带宽的利用率[25]；提高视频数据传输的实时性等。目前绝大多数面向内部网络的路由器都实现了 PIM-SM 组播路由协议和 IGMP 协议，具有支持组播业务的能力。PIM-DM 和 PIM-SM 是在视频会议中普遍采用的两种重要 IP 组播传送路径形式，当前最流行的基于 IP 网络的视频会议标准 H.323 就采用了 IP 组播技术。

但是从应用现状看，没有能在 Internet 上得到广泛应用[27]。一方面，因特网中的网络极少开放 IP 组播业务，至今还没有全因特网范围的组播业务，使得 IP 组播只能在小范围的 IP 组播岛上使用；另一方面，基于 IP 组播的上层应用也屈指可数，IP 组播的发展非常缓慢。从因特网发展的过程和 IP 网络的体系结构看，阻碍 IP 组播业务发展的主要因素为：

1. IP 组播体系结构缺乏可扩展性。路由器需要为每个组播组单独保存路由状态，而且这些组播地址不能聚合。组成员的动态性和网络中大量的组播组将需要路由器巨大的存储和处理开销。
2. 要求所有参加组播的端系统之间的路由器都必须支持组播功能，这给 IP 组播的推广使用带来很大的困难。
3. IP 组播模型在开放的因特网环境中难以支持有效的组管理和控制机制。当存在大量规模很小的组播组或者组播成员在空间上的分布很稀疏时，组播组管理上的开销将超过组播在带宽方面的优势。标准的 IP 组播业务模型中的接入控制、组管理和组地址的协调机制目前还没有找到有效的解决方案。
4. 在经济方面，组播打破了传统的流量计费机制。目前网络运营商以带宽使用获得收益，对承载的是单播还是 IP 组播没有必要区分，也就没有动力在路由器中增加对 IP 组播的支持。

4.2 应用层组播

4.2.1 应用层组播简介

面对 IP 组播业务在 Internet 中的困境，一些研究者提出将复杂的组播功能放在端系统实现的思想，将组播作为一种叠加的业务，实现为应用层的服务。因此，端系统组播又称为应用层组播。应用层组播网的结点是组播成员主机，数据路由、复制和转发功能都由成员主机完成。成员主机之间建立一个叠加在 IP 网络上的实现组播业务逻辑的功能性网络，这个网络称为叠加网(overlay network)，也称为控制拓扑，主机基于自组织算法建立和维护叠加网。IP 组播的数据沿着物理链路复制和转发，而应用层组播的数据则在主机实现复制和转发，数据报沿着逻辑链路即数据拓扑转发，多条逻辑链路可能经过同一条物理链路。应用层组播与 IP 组播的区别如图 4-1 所示：

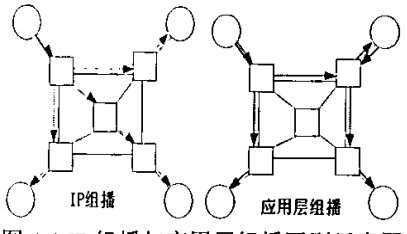

图 4-1 IP 组播与应用层组播区别示意图

4.2.2 应用层组播的研究现状和进展

应用层组播思想提出后的短短几年内，多个研究机构开展了应用层组播体系结构的研究项目：

1. ESM[20]

ESM 是目前为止最成功的一个项目。2000 年，ESM 的研究表明在端系统中实现组播功能的体系结构是可行的。2001 年，通过在 Internet 中实际运行基于 ESM 的视频会议，验证了 ESM 采用的自组织协议可以在动态的异构的因特网中支持较小规模的视频应用。Narada 是 ESM 的组网协议。Narada 协议首先在组播成员之间建立一个网状的叠加网，然后在叠加网上运行组播路由协议，建立一棵组播树。通过动态探测网络状态，Narada 动态地对叠加网进行维护和改良。

2. YOID

YOID 是在 2000 年提出的基于应用层组播的一整套内容分发解决方案，它包括应用层组播之上的可靠、安全和拥塞控制等机制。YMTP(YOID multicast tree protocol) 是 YOID 体系的核心，它是一种自组织的拓扑管理协议，将主机组织成网状网和共享的组播转发树。

3. Scattercast

Scattercast 是 2000 年提出的一种基于应用层组播实现因特网大规模广播业务的体系结构。其思想是在因特网中部署支持广播业务的称为 Scattercast Proxy 结点的服务器，将这些服务器组织成一个支撑网，支持用户规模巨大的 Internet 广播和软件分发应用。

4. Overcast

Overcast 是解决因特网内容分布的一个体系结构，实现可靠的组播业务。通过在网络中战略性地部署 Overcast 结点，然后由应用层组播机制将该结点组成一个骨干转发网络，能够以可靠的方式实现内容的分发。与 Scattercast 相比，Overcast 侧重于实现组播数据的可靠分发。

5. ALMI[16]

ALMI 是从 2000 年开始进行的研究项目，提出了将应用层组播作为端系统基础服务功能的体系结构。ALMI 设计了在操作系统的套接字之上，以中间件的形式向上层应用提供组播服务的结构，中间件实现自组织组网、组播复制和转发功能，在组播成员结点之间组成一个应用层组播网。

当前，应用层组播的研究还处于起步阶段。目前的各种应用层组播体系结构都是某种应用的解决方案，这些体系结构支持的应用不同，组网协议的性能侧重点不同。以上各项目的比较如表 4-1 所示：

表 4-1 应用层组播各项目比较表

项目	面向上层应用	支持组规模	选取链路时考虑的参数
ESM	实时和非实时	较小	带宽和时延，可以调整
ALMI	实时和非实时	较小	带宽和时延，可以调整
Scattercast	非实时，可靠	大	带宽
YOID	非实时，可靠	巨大	带宽，结点的逻辑连接数
Overcast	非实时，不可靠	巨大	结点的度

4.2.3 应用层组播技术的优势与局限

应用层组播可以避开网络层实现组播功能所遇到的难题：一是应用层组播的状态在主机系统中维护，不需要路由器保持组的状态，故不需要对路由器进行任何修改；二是组播应用可以随时部署，不需要网络设备的升级和功能扩展；三是可以简化组播的控制等功能的实现，建立在网络连接之上的应用层组播可以使用 TCP 和 UDP 服务，如可以利用 TCP 的可靠和拥塞控制简化组播的可靠和拥塞控制；四是便于针对特定应用进行优化，可针对不同的应用使用不同的实现方案，

而不必象 IP 组播那样必须统一到一个模型中。

应用层组播的局限主要有以下几方面：

1. 一般会比 IP 组播使用更多的网络资源。
2. 由于参与转发的端系统可能不稳定，导致组播转发的可靠性受到影响。
3. 由于参与转发的端系统的性能无法保证，可能导致延迟和转发速率等性能的下降。
4. 端系统对 IP 网络的了解有限，结点参与组网时，只能通过探测获得一些网络性能参数，选取的逻辑链路难以优化。
5. 主机不了解 IP 网络的拓扑结构，只能通过带宽和时延等外在的特性参数，以启发式的方式建立叠加网，控制拓扑不能较好地利用质量较好的底层网络资源，叠加网的多条链路可能经过同一条物理拓扑。

4．3 混合式组播

4.3.1 混合式组播的提出

在 OSI 模型的网络层或者应用层实现组播是目前组播业务的两种实现体制。通过前面对网络层的 IP 组播和应用层的应用层组播的简单分析，我们不难得知：基于 IP 组播的视频会议只能在相互孤立的组播岛上进行；在应用层组播中适合于视频会议的有 ESM 和 ALMI。ESM 的 NARADA 规定每个成员都与所有其它成员建立单播连接，只能应用于与会成员很少的情况；ALMI 规定多个数据源通过一个共享树来传输数据，传输延时比基于源的组播树要大，树中的结点个数(即与会成员)不能太多。同时，由于 ALMI 要求由会议控制者(Session Controller)指定新成员在叠加树中的父成员，所以当多个用户发言时就会出现多条耗带宽的组播链路，这样一来会议控制者就可能成为整个网络的瓶颈，这在一定程度上也限制了视频会议的规模。

为了充分利用应用层组播和 IP 组播的优势，扩大 IP 视频会议规模，在此把两者接合起来，称为混合式组播。

4.3.2 混合式组播的拓扑结构

在采用混合式组播的 IP 视频会议中，与会成员组成如图 4-2 所示的拓扑结构。在每个 IP 组播岛内，与会成员之间的数据传输使用 IP 组播，并选定一个与会成员作 DM(Designate Member，指定成员)，把岛内除 DM 外的与会成员称为 PM (Pertinent Member，其它成员)，DM 作为岛内成员与岛外成员之间通信的桥

梁。岛外成员与 DM 参加应用层组播叠加网(包括一个控制拓扑和一个数据拓扑)的构建，把它们称为 AM (ALM Member，应用层组播成员)，AM 之间的数据传输使用应用层组播。IP 组播岛中 PM 通过 DM 与叠加网相连，DM 与岛外成员之间的数据传输采用了应用层组播的方式，与岛内的成员之间的数据传输使用 IP 组播的方式。DM 负责将来自叠加网的数据以IP组播方式传送给组播岛中 PM，PM 的数据也通过 IP 组播方式传送到 DM，由 DM 以应用层组播的方式发送到叠加网上的其它与会成员。

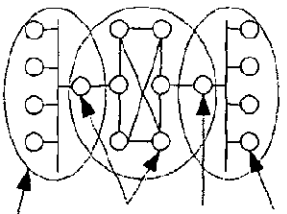

图 4-2 混合式组播拓扑结构图

DM 的选取原则：岛中第一个参加会议的成员被设为 DM，以后可以根据某种策略(如：参加会议的时间最长者或物理位置“中心”点成员，或剩余带宽资源最多者等等)选定某个成员作为 DM。如果 DM 离开，根据某种策略从岛内的 PM 中选择一个作 DM。

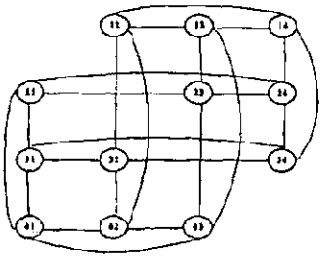
图 4-3 排列图 $A_{4,2}$ 的拓扑结构图

4.3.3 基于分层排列图的混合式组播

4.3.3.1 控制拓扑构建

在数学上，(n, k)维排列图用 $A_{n,k}$ 表示，其中 n 和 k 是自然数，$1\le k\le n-1$。令 $P\binom{n}{k}$ 表示集合{1,2,...,n}中 k 个符号的排列，$A_{n,k}$ 定义为无向图(V,E)，其中：

$$V=\left\{\left\{X=x_1x_2\ldots x_k \mid 1\le x_i\le n, x_i\ne x_j, 1\le i\ne j\le k\right\}=P\binom{n}{k}\right\},$$

$$E=\left\{(x,y)\mid x\in V, y\in V, \exists i, 1\le i\le k, x_i\ne y_i, \text{且 } x_i=y_j, \text{对于 } 1\le j\ne i\le k\right\}$$

图 4-3 是一个排列图实例 $A_{4,2}$。一个排列图 $A_{n,2}$ 仅能容纳 n(n-1)个主机，为使系统可容纳任意多主机，可将参与主机组合成多个 $A_{n,2}$ 结构，这些 $A_{n,2}$ 按照树形结构连接在一起，称为分层排列图结构。

混合式组播的核心任务是为 AM 完成拓扑(包括数据拓扑和控制拓扑)的构建。由于网络状况的动态性，以及客户端数量与客户端加入系统时间的不确定性，当前的应用层组播方案一般需要每个结点定期与其它全部或一定比例的结点交换控制信息，以了解其它结点的状态，来形成控制拓扑以保障数据拓扑的健壮性。在结点个数逐渐增加时，网络上的控制信息数量会占用大量的带宽资源。应用层

组播的数据拓扑可以分为两大类：信源树和共享树。信源树的缺点必须为每个组播源保存路由信息，这样会占用大量的系统资源。共享树传输延时较大。视频会议系统属于实时交互应用，视频数据占据很大的带宽，并且对端到端传输延迟和网络抖动的要求很高。需要寻找一种既可以减少交换控制信息又可以保证视频数据实时传输的拓扑构建方式，为利用排列图的优越特性，如结点和边的对称性、故障恢复能力和容错能力，可以嵌入独立的组播树等，这里采用分层排列图的结构来组织混合式组播中的 AM。

本文先按照文献[21]中的协议设计方法把参与 IP 视频会议所有主机中的 AM 组织成分层排列图结构作为混合式组播的控制拓扑，并在主机申请加入会议时根据其带宽处理能力和一定的算法规定它的入度最大值和出度最大值，它们分别也是剩余入度和剩余出度的初始值。每个分层排列图都对应一棵以排列图为结点单位的树。如图 4-4 所示的以与会主机为单位的拓扑结构是一个由三个 $A_{4,2}$ 结构(AG_1、AG_2 和 AG_3)所组成的分层排列图。图 4-4 也可以看作如图 4-5 所示的以排列图为单位的树结构，即以排列图 AG_2 为根，以 AG_1 和 AG_3 为叶子的一棵树状拓扑。设在分层排列图中，父排列图 AG_i 中的结点 i_1 和 i_2 分别是子排列图 AG_j 的数据源结点和备用数据源结点，AG_j 中的结点 j_1 和 j_2 分别是 i_1 和 i_2 对应的根结点，则在树拓扑中父排列图 AG_i 与子排列图 AG_j 的连线实际上包括两条：一是 i_1 与 j_1 之间的连线，一是 i_2 与 j_2 之间的连线。图 4-5 中父排列图 AG_2 与其子排列图 AG_1 之间的连线有两条：一是图 4-4 中 AG_2 的 a_{14} 与 AG_1 的 a_{31} 之间的连线，一是图 4-4 中 AG_2 的 a_{24} 与 AG_1 的 a_{41} 之间的连线。各排列图之间的连线在混合式组播中并不一定作为数据传输路径。

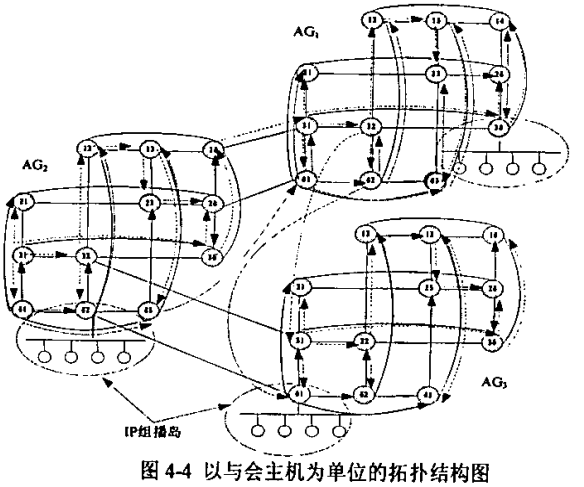

图 4-4 以与会主机为单位的拓扑结构图

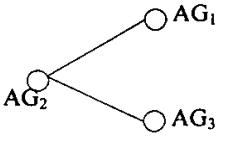

图 4-5 以排列图为单位的树结构图

4.3.3.2 数据拓扑构建

基于分层排列图的混合式组播的控制拓扑是分层排列图，其数据拓扑在此基

础上构建，流程图如图 4-6 所示。具体算法如下：

设某排列图中的与会成员主机 $a_{i,j}$ 发言，首先把分层排列图对应的树结构转换为以 $a_{i,j}$ 所在的排列图为根的树结构。记结点剩余入度值和剩余出度值分别用 *rd* 和 *cd* 表示。执行 1－3：

1. 把 $a_{i,j}$ 所处的排列图当作当前排列图，记作 AG。在 AG 中用文献[21]中的算法生成以 $a_{i,j}$ 为根的组播树，并修改 AG 中所有结点的 *rd* 和 *cd*；
2. 为 AG 的所有子排列图选择数据源、备用数据源和根结点(为描述方便起见，这里分别把它们记为 s_1，s_2 和 *r*)。为某个子排列图 AGi 选择数据源 s_1、备用数据源 s_2 和根结点 *r* 的算法如下：
   1) 令与会成员主机结点子集 $A = \{a_{m,n} | a_{m,n}$ 属于 AG，$a_{i,j}$ 与 $a_{m,n}$ 相邻接并且 $a_{m,n}$ 的 *cd* 大于 0$\}$。
   - 如果$|A|>1$，在 A 中选择 cd 最大者作为数据源 s_1，次大者作为备用数据源 s_2，此时一定可以选取合适的根结点使得组播树独立；
   - 如果$|A| = 0$，则选择在 AG 中选择最大剩余出度结点作为数据源 s_1，次大者作为备用数据源 s_2；
   - 如果$|A| = 1$，则选择 A 中的结点作为数据源 s_1，选择 AG 中其它成员中最大剩余出度结点作为备用数据源 s_2；
   2) s1 和 s2 的 *cd* 分别减 1；
   3) 在 AGi 中选取数据源 s1 对应的根结点 r，并把选出的根结点的 rd 减 1。当 s1 不能正常工作或失效时，s2 作为 r 的数据源结点。选取根结点时即要考虑结点的 *cd* 和 *rd*，又要注意尽力保证组播树的独立性，如果能保证组播树的独立性的结点的剩余入度值全为零，则选取子排列图中的 *rd* 最大者；
   4) 并在 AGi 内根据文献[21]中的算法生成以 r 为根的组播树，并修改相应结点的 *cd* 和 *rd*；
3. 如果 AGi 还有再下一级的子排列图 AGj，则令 $a_{i,j}$ = AGj 的根结点的 r，转 1。否则，结束。

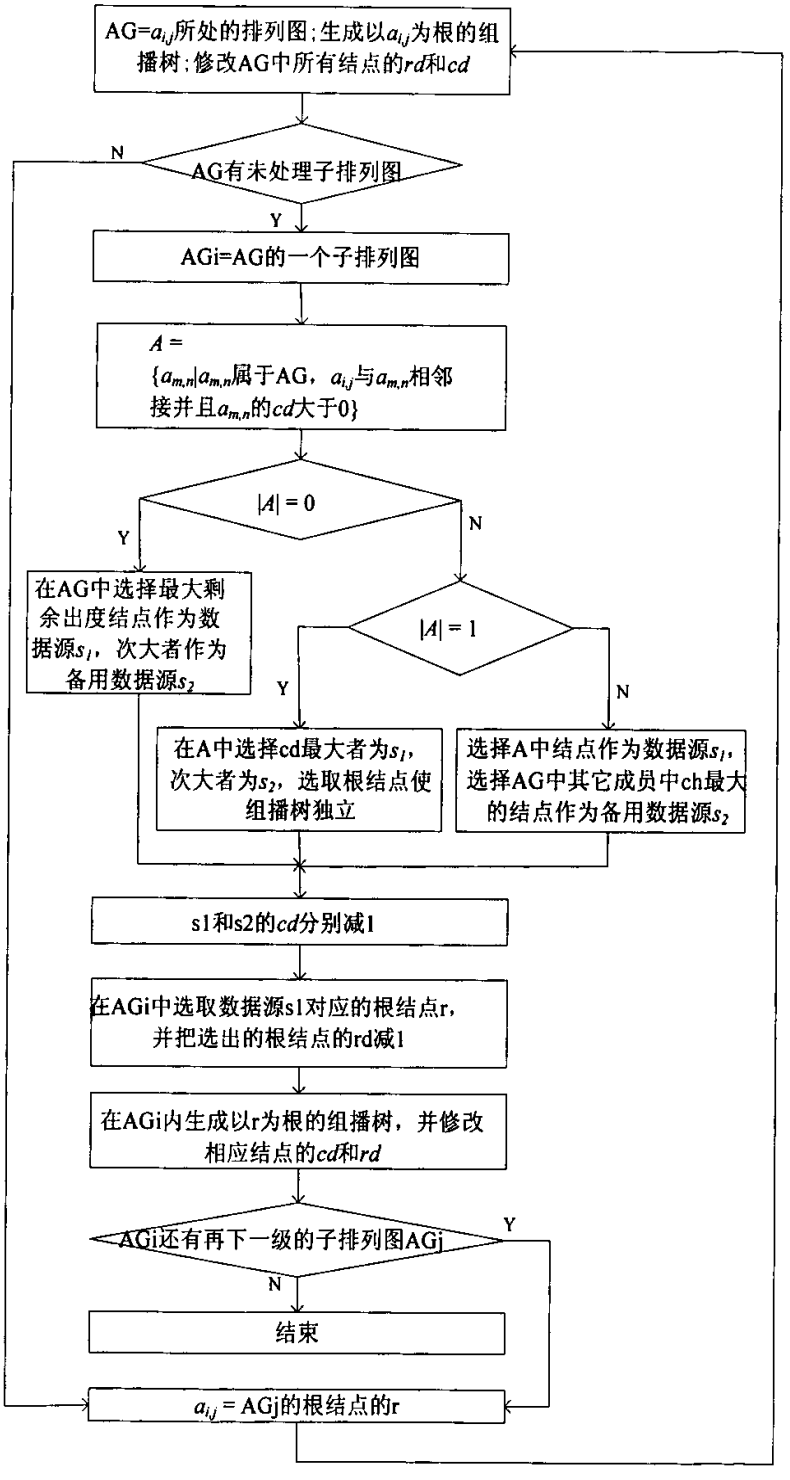

图 4-6 数据拓扑构建流程图

图 4-4 给出了 AG_1 中的 a_{31} 和 AG_2 中的 a_{41} 发言时根据以上算法为实现应用层组播建立的组播树，分别以红色虚线和绿色实线表示。在 AG_1、AG_2 和 AG_3 中分别由三个 DM(a_{34}, a_{42}, a_{41})作为 IP 组播岛的数据源结点，组播岛内的 IP 组播树

分别以它们为根建立，由它们传输到 IP 组播岛中的 PM，IP 组播岛中对应的 IP 组播树在图 4-4 中没有画出。

4.3.3.3 性能分析

与当前其它的应用层组播相比，在 IP 视频会议中采用基于分层排列图的混合式组播技术，有以下优点：

1. 增强了系统的可扩展性。系统主要维护树形式的分层排列图即可，它就是应用层组播的控制拓扑，由它生成数据拓扑。IP 组播岛内的数据传输由 IP 组播技术实现，系统只需选定 DM，由 DM 来作为岛内成员与岛外成员之间通信的桥梁。控制信息数量大大减少，节约了网络带宽，使更多的用户可以加入会议。

2. 提高了系统的容错能力，减少了网络抖动。由于在 $A_{n,2}$ 中可以建立 n-2 个独立的组播树，在用户带宽资源足够丰富的情况下只要发言用户数小于等于 n-2 就可以保证组播树的独立，使得每个用户只在最多一棵组播树上作为中间结点，这样一个用户出现故障至多影响一个组播树中的数据传输。分层排列图的这一性质能使基于分层排列图的混合式组播能提高视频会议系统的容错能力，减少数据传输过程中的网络抖动。

3. 缩短了用户加入会议的时间。由于 AM 组织成分层排列图结构，IP 组播岛外的用户加入会议的复杂度为 O(log(N))，其中 N 表示 IP 视频会议系统中 AM 数目；组播岛内的用户加入更简单，它不需要了解岛外 AM 的任何信息，就可以加入它所属的 IP 组播岛，通过这个岛上的 DM 与岛外与会成员传输数据，因而一个申请加入会议的主机可迅速地完成加入过程。

4.3.3.4 在 TCP/IP 协议栈中的位置

基于分层排列图的混合式组播处于用户空间，它在 TCP/IP 协议栈中的位置如图 4-7 所示：

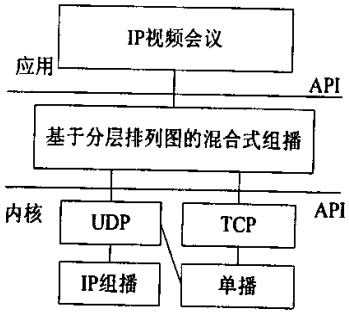

图 4-7 混合式组播在 TCP/IP 协议栈中的位置示意图

可以使用内核提供的 API 将基于分层排列图的混合式组播设计成一个中间

件或者守护程序，并向其上层应用基于 IP 的视频会议系统提供一个 API 接口。

4.4 本章小结

本章对 IP 组播和应用层组播进行了较为详细的介绍，分析对比了两者的优势与局限，提出了基于分层排列图的混合式组播，给出了它的控制拓扑与数据拓扑的构建过程，并把它作为本系统视频组播方式。

混合式组播机制解决了互联网对 IP 组播的屏蔽，是一种智能化高效率的处理技术。混合式组播的处理方式是岛外发言成员通过应用层组播的方式将视频数据传输到组播岛中 DM，也就是 DM 与岛外发言成员之间只建立一条数据传输通道，而组播岛中其余各与会成员均接收通过 DM 转发的 IP 组播数据，这样就大大减轻了组播岛出口带宽的压力。同样道理，当该组播岛内的与会成员的视频数据通过 IP 组播的方式传输到 DM，然后由 DM 通过应用层组播的方式传输到岛外的与会成员。这样将的应用层组播与 IP 组播的优势完美的统一在一起，即保证了功能上的完善，更提供了性能上的高效，为 IP 视频会议系统提供了一种可靠的混合式组播机制，整个过程非常简单，无需复杂配置。混合式组播机制能充分利用参与会议的主机资源，使得系统能支持更多的用户参加会议。

5 系统设计

本系统的硬件要求非常简单，只需要在计算机上配有标准的视频采集卡或其它视频设备。本部分研究视频实时传输系统的软件实现，软件开发平台为Windows 2000，主要开发工具为 Microsoft Visual C++6.0[41,42]，并采用 DirectShow 技术[47,48](安装版本为 DirectX 9.0 SDK Update)。

5.1 DirectShow 概述

微软推出的 DirectShow 技术，可以对来自本地或网络的各种视频音频压缩格式的媒体文件的进行高品质的解码和回放，可以从设备上捕捉多媒体流，也可以处理各种压缩算法处理的流媒体。解决视音频数据处理的高效性以及时刻保持视音频同步是 DirectShow 的重要设计目标。DirectShow 通过 DirectDraw 和 DirectSound 将多媒体流进行解码并快速地传送到系统的声卡或视频卡中进行回放；通过将多媒体流封装成带时间戳的流实例(Stream Samples)来实现同步问题；通过过滤器(Filter)来处理不同的数据源、数据格式以及硬件设备等问题。DirectShow 简化了媒体的回放，格式转化和捕捉工作。DirectShow 支持基于 WDM(Windows Driver Model)驱动程序的设备的过滤器，也支持早期的 VFW(Video for Windows)视频捕捉卡的过滤器以及为 ACM(Audio Compression Manager)和 VCM (Video Compression Manager)接口编写的编解码器。DirectShow 基于 COM，提供了一个开放式的开发环境，使得在 Windows 平台上开发基于 DirectShow 框架的应用程序变得非常简单。

DirectShow 系统结构如图 5-1 所示：

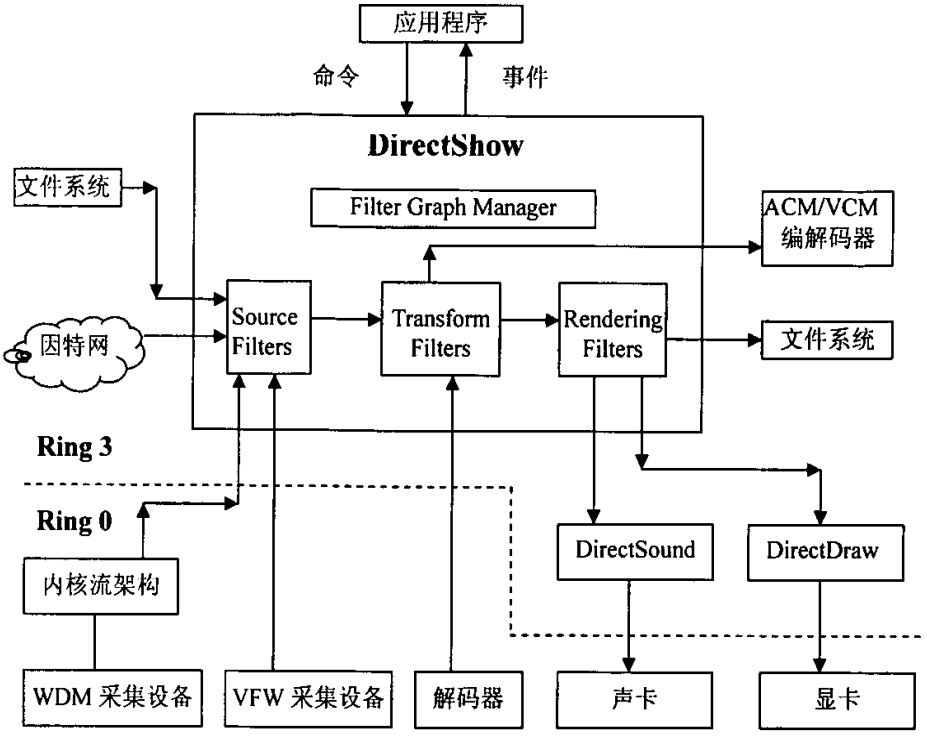

图 5-1 DirectShow 系统结构示意图

DirectShow Filter 工作在用户模式(User Mode，操作系统特权级别为 Ring 3)，而硬件工作在内核模式(Kernel Mode，操作系统特权级别为 Ring 0)，这些 Filter 封装了硬件设计，Filter 内部完成了与系统的硬件驱动程序进行交互，当 Filter 上的接口方法被应用程序或其他 Filter 调用时，它会将调用方法以及参数传递给硬件驱动程序，由驱动程序最终完成指定功能。这个特性使得开发人员从为支持硬件的特殊处理中解脱出来。

过滤器(Filter)是执行特定任务的 COM 对象，是可插入标准组件，它带有输入或输出针(Pin)，或二者兼而有之。过滤器按功能可分为源过滤器(Source Filter)、变换过滤器(Transform Filter)和表现过滤器(Renderer Filter)三种。

DirectShow 使用过滤器图(Filter Graph，如图 5-2 所示)模型管理整个数据流的处理过程；各个过滤器在 Filter Graph 中按一定的顺序连接成一条“流水线”协同工作。为了完成整个任务，必须要将所有的过滤器 Filter 连接起来构成过滤器图，它仅能有一个 Source Filter 和一个 Renderer Filter，但可以有零到多个 Transform Filter。Filter Graph 是通过 Filter 的输入输出针(Pin)顺序连接而成的，这些 Pin 经过协商来决定它们将支持的多媒体格式，前级 Filter 的输出成为下级 Filter 的输入。在 Filter Graph 中，Source Filters 主要负责获取数据，数据源可以是本地文件、DVD、视频捕捉卡和电视接收卡等，然后将数据透明地传输到下一级 Filter；Transform Filters 用来获取、转换和传送媒体数据，它包括分离视频和音频的分解变换过滤器(Splitter Transform Filter)、解压视频数据的视频转换过

滤器(Video Transform Filter)、解压音频数据的音频转换过滤器(Audio Transform Filter)；Rendering Filters 主要负责数据的最终去向，用来在硬件上表现媒体数据，比如将把视频数据显示在屏幕上，音频数据播放到声卡上，或将它们存储在磁盘里。

图 5-2 Filter Graph 示意图

针(Pin)是和一个 Filter 相连的对象，它能在两个 Filter 中间传送和操纵多媒体数据流。Pin 有两种类型：输入(Input)针和输出(Output)针。媒体样本从 Output Pin 流向 Input Pin，这种方向为流下游(downstream)，反之为流上游(upstream)。一个 Filter 的 Input Pin 连接到流上游 Filter 的 Output Pin，Filter 的 Output Pin 连接到流下游 Filter 的 Input Pin。当 Filter 请求互相连接时，处在流下游的 Filter 的 Input Pin 请求处在流上游的 Output Pin 是否支持该种媒体格式，这种处理是通过 Filter Graph Manager 协调来完成的。连接 Filter 必须互相协商直到它们发现两个 Pin 能互相连接。Filter Graph 中从一个 Filter 流向另一个 Filter 的数据(例如一个视频帧)称为媒体样本(Media Sample)。除了多媒体数据外，Media Sample 对象还包含特定媒体样本的类型、大小和时间戳等信息。

Filter Graph 由一个称为过滤器图管理器(Filter Graph Manager)的 COM 对象管理。Filter Graph Manager 是一个高层的组件，它控制 Filter Graph 中的数据流动和状态变化，为所有的 Filter 建立参考时钟，在 Filter 和应用程序之间传递信息，它能够管理 Filter Graph 的建立，实例化一个 Filter 并将其加入到 Graph 中。Filter Graph Manager 可以向应用程序传递事件通知，应用程序对事件响应。此外，Filter Graph Manager 还简化了 Filter Graph 的创建。应用程序可以通过 Filter Graph Manager 提供的接口来访问 Filter Graph、控制流媒体或者接收 Filter 事件。Filter，Filter Graph，Filter Graph Manager，应用程序之间的关系如图 5-3 所示：

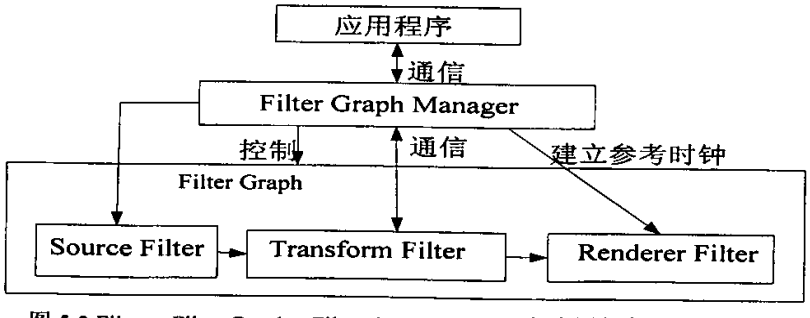

图 5-3 Filter，Filter Graph，Filter Graph Manager 与应用程序关系示意图

由于 DirectShow 支持可重构的 Filter Graph 结构，所以使用相同的软件组件可以播放多种类型的媒体。开发人员可以通过定义自己的 Filter 来扩展

DirectShow 对媒体的支持功能。

5.2 开发环境的配置

在进行 DirectShow 开发之前，需要下载并安装相应版本的 DirectX SDK，这个可以从 Microsoft 的官方站点 http://msdn.microsoft.com/directx 上免费获得，最新的 DirectX 9.0 SDK Update(Summer 2003)。

1. 使用 VC 向导生成一个具体项目，如 Win32 Dynamic-Link；

2. 包含头文件 streams.h；

3. 在 VC 的菜单中选择 Project→Settings→C/C++，在弹出的对话框中的 Category 中选择 Code generation，然后在 Calling convention 中选择_stdcall；

4. 使用多线程语言运行时库，即在 VC 的菜单中选择 Project→Settings→C/C++，在弹出的对话框中的 Category 中选择 Code generation，然后在 Use run-time library 中，Debug 版选择 Debug Multithreaded，Release 版选择 Multithreaded。

5. 配置必要的链接库文件，即在 VC 的菜单中选择 Project→Settings→Link，在弹出的对话框中的 Category 中选择 General，然后在 Object/library modules 中输入如下代码：

Debug 版本：strmbasd.lib, msvcrtd.lib, winmm.lib

Release 版本：strmbase.lib, msvcrt.lib, winmm.lib

并且选中 Ignore all default libraries。

DirectShow 应用程序应该至少连接库文件 strmiids.lib 和 quartz.lib。前者定义了 DirectShow 标准的 CLSID 和 IID，后者定义了导出函数 AMGetErrorText(如果应用程序中没有使用到这个函数，也可以不连接这个库)。如果程序里包含了头文件 streams.h，库文件一般还要连接 strmbasd.lib、uuid.lib 和 winmm.lib。

6. 将 DirectX SDK 的 Include 和 Lib 目录配置到 VC 的系统目录中去，并且放在标准的 VC 目录之前，以保证编译器能够得到最新版本的源文件。选择 Tools→Options→Directories，在弹出的对话框中的 Show directories for 中选择 Include files，配置如下(假设 DirectX SDK 安装在 D:\DXSDK 目录下，VC 安装在 C:\Program Files 下)：

D:\DXSDK\Include

D:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES

D:\DXSDK\SAMPLES\C++\COMMON\INCLUDE

C:\Program Files\Microsoft Visual Studio\VC98\INCLUDE

C:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE

C:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE

再在 Show directories for 中选择 Library files，配置如下：

D:\DXSDK\Lib

D:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES\DEBUG

D:\DXSDK\SAMPLES\C++\DIRECTSHOW\BASECLASSES\RELEASE

C:\PROGRAM FILES\MICROSOFT SDK\LIB

C:\Program Files\Microsoft Visual Studio\VC98\LIB

C:\Program Files\Microsoft Visual Studio\VC98\MFC\LIB

7. 因为 DirectShow 应用程序是一种 COM 客户程序，因此应在调用任何 COM 函数之前调用 CoInitialize( )(或 CoInitializeEx( ))函数进行 COM 库的初始化(一般是在应用程序启动的时候调用一次)，在结束 COM 库使用时调用 CoUninitialize( )函数进行反初始化(一般是在应用程序退出前调用一次)。

5.3 系统总体设计

视频实时传输主要实现视频数据的采集、压缩编解码和回放功能。这里用 Directshow 技术[37,47,48]实现视频数据的采集和回放，用 MPEG-4 标准对视频数据进行压缩/解压缩[10]。

本文视频实时传输系统流程如图 5-4 所示：

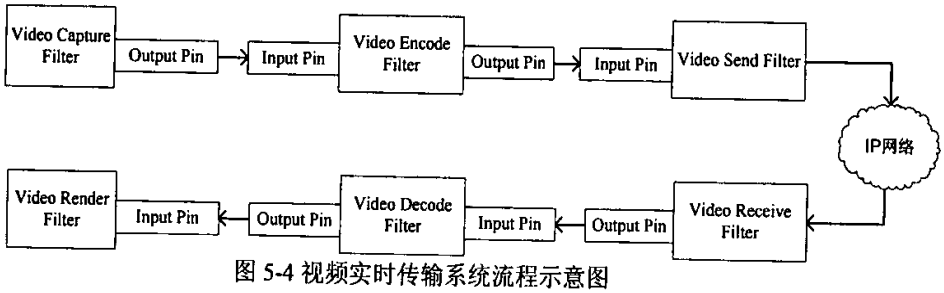

图 5-4 视频实时传输系统流程示意图

图 5-4 上半部分为发送端模块，下半部分为接收端模块，双方通过 IP 网络交换实时视频数据。其中，Video Capture Filter 是视频采集过滤器，Video Send Filter 是视频发送过滤器，Video Receive Filter 是视频接收过滤器，Video Render Filter 是视频回放过滤器，Video Encode/Decode Filter 是视频压缩/解压过滤器。数据传输的质量控制消息封装在 RTCP 协议的报文中。

视频会议往往要求对实时采集到的视频数据在本地进行预览。为节约带宽，若采集设备不是 AGP 接口时，不直接从采集设备 Filter 的 Preview 输出 Pin 接出直接预览，而是在采集 Filter 的 Output Pin 后面接一个 Smart Tee。这样，实现采集时预览占用的系统资源要少很多。预览和采集前端 Filter 链路的构建是相同的，仅在后端有些差异：预览时只接 Video Render Filter，在采集时则接 Video Encode Filter。限于篇幅，预览功能在本文中没有体现。

5.4 Video Capture Filter 设计

5.4.1 系统枚举

不同的计算机上硬件配置情况不同，我们使用 DirectShow 中的系统枚举来使用这些不预知的采集设备。采集设备必须以 Filter 的形式参与到 DirectShow 中与其他 Filter 协同工作。采集设备 Filter 创建的实质上也是一个系统枚举过程。当用户选定了一个视频采集设备后可以得到它的友好名字，并以这个名字为参数去创建 Filter。视频捕捉设备注册在 CLSID_VideoInputDeviceCategory 目录之下。因此，系统可以通过枚举视频类型目录，来得到当前的视频采集设备。枚举的步骤如下：

1. 使用 CoCreateInstance( )函数创建一个系统枚举组件对象，并获得 IcreateDevEnum 接口。

2. 使用 IcreateDevEnum::CreateClassEnumerator( )为指定的类型目录创建一个枚举器，并获得 IenumMoniker( )接口。

1) 使用 IenumMoniker::Next 枚举指定类型目录下所有的设备标识(Deviece Moniker)，每个设备标识对象上都实现了 Imoniker 接口。

2) 调用 Imoniker::BinToStorage( )后就可以访问设备标识的属性集，如设备名(Display Name)和设备友好名(Friendly Name)等。

3) 调用 Imoniker::BindToObject 可以将设备标识绑定成一个 DirectShow Filter，通过调用 IFilterGraph::AddFilter 加入到 Filter Graph 中即可。

5.4.2 视频采集流程

视频采集流程图如图 5-5 所示：

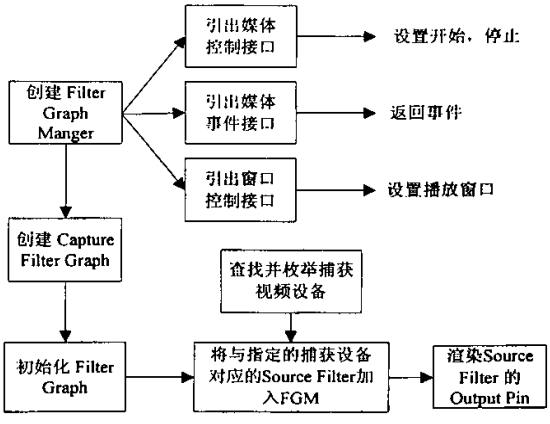

图 5-5 视频采集流程图

5.4.3 Video Capture Filter 的具体实现

Video Capture Filter 的具体实现：

```
BOOL QueryVideoCaptureDevices(DEVICE_LIST&outDevices)
/*通过枚举 CLSID_VideoInputDevieceCategory 目录得到系统中所有视频采集设备。*/
{ return QueryDevieceCategory(
CLSID_VideoInputDevieceCategory, outDevices);}
BOOL QueryDevieceCategory (GUID nCategory, EVICE_LIST&outDevices)
{
HERSULT hr = NOERROR;
IcreateDevEnum * enumHardware = NULL;
hr = CocreateInstance(CLSID_SystemDeviceEnum, NULL, CLSCTX_ALL,
IID_IcreateDevEnum, (void**) & enumHardware);
//创建一个系统枚举组件对象
IEnumMoniker * enumMoniker = NULL;
hr = enumHardware->CreateClassEnumerator(inCategory, &enumMoniker, 0);
//为指定的类型目录创建一个枚举器
……
//通过循环逐一查询类型目录下所有设备
while(SUCCEEDED(enumMoniker->Next(1, &moniker, &friend))&&fetched)
if(moniker)
{
WCHAR *wzDisplayName = NULL;
IpropertyBag *propertyBag = NULL;
IbaseFilter *filter = NULL;
VARIANT name;
//获取当前设备的显示名字
hr = moniker->GetDisplayName(NULL, NULL, &wzDisplayName);
if(SUCCEEDED(hr))
hr = moniker->BindToStorage(0, 0, IID_IpropertyBag,(void**)&propertyBag;)
}
//获取当前设备的友好名字，用于用户界面上显示
hr = propertyBag->Read("FriendlyName", &name, NULL);
//将设备标识绑定为 Filter 形式
hr = moniker->BindToObject(0, 0, IID_IbaseFilter, (void**)&filter);
if(SUCCEEDED(hr))
{
//保存当前设备信息，类型，并将当前设备加入到列表框中供用户选择
……
```

```
}
}
```

5.5 Video Encode/Decode Filter 设计

视频数据容量很大，在网络传输前都需要进行压缩，接收端接收后同样需要进行解压然后送入播放缓冲区进行回放。这里我们采用的视频编解码方案是MPEG-4。DivX[51]是与MPEG-4完全兼容的视频压缩技术，其视频编码采用的就是MPEG-4压缩标准，是目前互联网上普遍采用的MPEG-4编码器之一。在本软件的实际开发中，选择了DivX 5.0.2编码解码器用于视频图像的压缩解压缩编码处理。

Video Encode/Decode Filter属于Transform Filter类型。

5.5.1 压缩编码实现

创建视频压缩的实现：

通过系统枚举 Video Compressors 目录(CLSID-VideoCompressorsCategory)得到系统中已安装的视频压缩编码器，显示给用户，用户选择后反馈给系统用于创建压缩Filter。

实现代码如下：

```
//其他 Filter 创建
……
//如果用户选择 DivX 压缩，创建该压缩 Filter
if(mLiveCapture->GetEncodingType( )==ET_DIVX)
{
//通过枚举 Video Compressors 目录来创建 DivX 压缩 Filter
IBaseFilter *pFlt = UDsUtils::CreateCompressor(TRUE, "DivX 5.0.2 Codec");
//如果创建成功，将压缩 Filter 加入到 Filter Graph 中
mVideoEncoder = new CDXFilter(mGraph->GetGraph( ));
mVideoEncoder->Attach(pFlt, "DivX, Encoder");
…
}
//系统枚举函数
IBaseFilter * UDsUtils::CreateCompressor(BOOL inIsVideo,
                const char * inName )
{ //Video Compressors 目录(CLSID-VideoCompressorsCategory)
GUID guid = inIsVideo?CLSID-VideoCompressorsCategory:
```

```
                CLSID-AudioCompressorsCategory;
Return CreateHardwareFilter(guid, inName);
}
```

5.5.2 解码实现

解码实现过程的关键部分：

1. 包含用到的头文件，确定解码器端需求数据结构，解码器存储结构定义，解码器结构定义，解码器端视频帧结构定义，变量宏的定义。

2. 初始化相关信息，如视频帧信息等。

3. 内存空间分配。可以跳过内存分配失败检测。

```
yuv_buffer = (unsigned char*) malloc(XDIM *YDIM);
bmp_buffer = (unsigned char*) malloc(3*XDIM*YDIM);
divx_buffer = (unsigned char*) malloc(BUFFERSIZE);
bmp_head = (unsigned char*) malloc(54);
filehandle = fopen("bmphead.dat", "rb");
divx_size = fread(bmp_head, sizeof(unsigned char), 54, filehandle);
fclose(filehandle);
```

4. 解码过程

```
//初始化解码器
.....
//视频解码的主循环
while(解码标志)
//读入 MPEG-4 编码视频帧数据
filehandle = fopen(filename, "rb");
if(!filehandle)
fprintf(stderr, "Error reading file\n");
divx_size = fread(divx_buffer, 1, BUFFERSIZE, filehandle);
fclose(filehandle);
//帧解码
 dec_frame.length = divx_size;
 dec_frame.bitstream = divx_buffer;
 dec_frame.bmp = (void *)bmp_buffer;
 dec_frame.render_flag = 1;//0 代表跳过该帧
 dec_frame.stride = XDIM;
 status = decore(MY_APP_ID, 0, &dec_frame, NULL);
//写入编码后的帧
 filehandle = fopen(filename, "wb");
```

```
fwrite(bmp_head, 1, 54, filehandle);
fwrite(bmp_buffer, XDIM, YDIM*3, filehandle);
fclose(filehandle);
```

5. 解码器资源释放

```
.....
free(divx_buffer);
free(yuv_buffer)
free(bmp_buffer);
```

5.5.3 DivX 编码解码器的源代码

DivX 的编码解码器的源代码如下：

```
#ifdef _cplusplus
Extern "C"{
#endif
#ifndef _CODEC_H
#define _CODEC_H
class cadec{
private:
      float framerate
public:
     codec( );
    ~codec( );
     long about( LPARAM lParam1, LPARAM lParam2 );
     long config(LPARAM lParam1, LPARAM lParam2);
     long getInfo(LPARAM lParam1, LPARAM lParam2);
     long encGetFormat(LPARAM lParam1, LPARAM lParam2 );
     long encGetSize(LPARAM lParam1, LPARAM lParam2 );
     long encQuery(LPARAM lParam1, LPARAM lParam2);
     long encFramesInfo(LPARAM lParam1, LPARAM lParam2 );
     long encBegin(LPARAM lParam1, LPARAM lParam2 );
     long encEncode( LPARAM lParam1, LPARAM lParam2 );
     long encEnd( LPARAM lParam1, LPARAM lParam2 );
     long decGetFormat( LPARAM lParam1, LPARAM LParam2 );
     long decQuery(LPARAM lParam1, LPARAM lParam2 );
     long decBegin( LPARAM lParam1, LPARAM lParam2);
     long decDecode( LPARAM lParam1, LPARAM lParam2 );
     long decEnd( LPARAM lParam1, LPARAM lParam2 );
     //解码参数
```

```
        long decSetPostProcessing(LPARAM lParam1, LPARAM lParam2 );
    private:
        int getImageType(BTIMAPV4HEADER *bmpinfohdr);
        void convertImage ( void *imageIn, void *imageOut, long type, long
                    x_dim , long y_dim );
        void InitLookupTable( );
        int convertYUV422(void *yuv422, void *yuv,  long type, long x_dim,
                    long y_dim);
        int convertYUV420 ( void *yuv420, void *yuv,  long type, long x_dim,
                    long y_dim );
        int convertRGB(void *bmp, void *yuv, long type, long x_dim, long
                    y_im );
    };
#endif//_CODEC_H
#include<vfw. h>
static const DWORD FOURCC_DIVX =
            mmioFOURCC('D', 'I', 'V', 'X');//DivX 格式
static const DWORD FOURCC_UYVY =
            mmioFOURCC('U', 'Y', 'V', 'Y');//未压缩的 UYVY
static const DWORD FOURCC_YUY2 =
            mmioFOURCC('U', 'Y', 'V', '2');//未压缩的 YUY2
static const DWORD FOURCC_YUYV =
            mmioFOURCC('Y', 'U', 'Y', 'V');//未压缩的 YUYV
static const DWORD FOURCC_V422 =
            mmioFOURCC('V', '4', '2', '2');//未压缩的 V422
static const DWORD FOURCC_YVYU =
            mmioFOURCC('Y', 'V', 'Y', 'U');//未压缩的 YVYU
static const DWORD FOURCC_YV12 =
            mmioFOURCC('Y', 'V', '1', '2');//未压缩的 YV 12
static const DWORD FOURCC_I420 =
            mmioFOURCC('I', '4', '2', '0');//未压缩的 I420
static const DWORD FOURCC_IYUV =
            mmioFOURCC('I', 'Y', 'U', 'V');//未压缩的 IYUV
extern long rc_period, bitrate;
extern int search_range;
extern int max_quantizer;
extern int min_quantizer;
#ifdef cplusplus
}
```

#endif

5.6 Video Send Filter 设计

Video Send Filter 位于发送端，用于处理 Video Encode Filter 编码后的视频流，将数据流分块打包后发送出去。在 DirectShow 中，Filter 之间的数据传送以 Media Sample 的形式来完成，对于连接入链路的任一个 Filter 都有机会获得数据流中的每一个 Media Sample。因此在需要开发一个 Video Send Filter 接到 Video Capture Filter 的后面，以获取采集输出的视频数据，然后向网络发送。它是发送端 Filter Graph 的终点，属于 Renderer Filter 类型。

5.6.1 Video Send Filter 的数据处理流程与数据流动

Video Send Filter 属于 Renderer Filter，它只有一个 Input Pin，没有 Output Pin。Video Send Filter 用于发送视频流，过滤器的输入针(pin)接收经编码压缩处理后的视频媒体实例(Media Sample)，然后进行媒体类型检查，根据不同的媒体类型确定 RTP 数据报的负载类型(PT)以及 RTP 时间戳增量。封装好 RTP 数据分组后将数据发出。

Video Send Filter 的数据处理流程和数据流动如图 5-6 和 5-7 所示：

图 5-6 Video Send Filter 的数据处理流程图

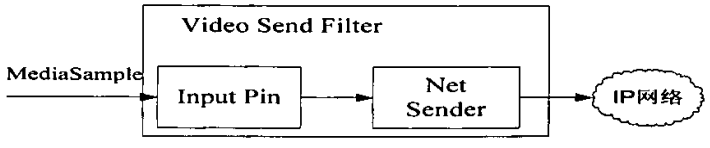

图 5-7 Video Send Filter 的数据流动示意图

5.6.2 Video Send Filter 的结构

Video Send Filter 的结构如图 5-8 所示：

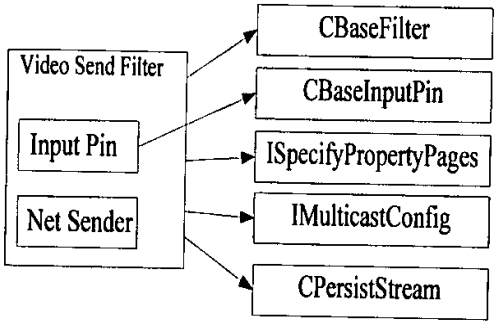

图 5-8 Video Send Filter 的结构示意图

Video Send Filter 有两个主要的成员对象：Input Pin 和 Net Sender。图中的箭头表示类的继承关系，Video Send Filter 对象继承了 4 个父类：

1. CBaseFilter 基类：由 DirectShow SDK 提供，是为自行开发 Filter 而提供的 DirectShow 基类。CBaseFilter 是所有 Filter 的基类。
2. ISpecifyPropertyPages 接口：主要功能是提供属性页的设置。
3. IMuIticastConfig：主要功能是进行网络配置，该接口由本方案来实现。
4. CPersiststream 类：这个类主要为 Filter 提供持久的属性，可以将属性设置记录在存贮的 Graph 中。

Video Send Filter 的成员 Input Pin 继承了 CBaseInputPin 基类。Input Pin 需要完成的主要功能是 Filter 之间所传输的媒体类型的协商和从上方的 Filter 接收 Media Sample。Video Send Filter 的成员 Net Sender 用于完成基本的网络发送功能，包括加入或退出组播，以及发送一定长度的数据块。

5.6.3 具体实现

5.6.3.1 Net Sender 的实现

Net Sender 类用于完成基本的网络传输功能。它还提供了接口，供外部应用执行组播通信的基本配置，这些配置必须在过滤器的停止状态下完成，当过滤器由停止转换到暂停状态时，输入 Pin 的 Active( )函数被激活，过滤器开始接收第一帧数据。方案中，它的主要功能有三个：

1.JoinMulticast( )：加入组播组。

该函数有 4 个参数：组播地址、端口号、使用的 IP 地址(函数中被忽略)以及 TTL。它创建 Socket，绑定端口和地址，设置 TTL，并加入多播组。

2. LeaveMultcast( )：退出多播组。

该函数没有参数，它关闭当前使用的 Socket。

3.SetTTL(unsigned char ttl)：设置组播传送的生命周期。

4.SetOwnPacket(bool accept)：设定是否接收环回数据。

5. Send( )：负责发送数据。

该函数有 2 个参数：一个数据块地址，一个是要发送的数据块大小。使用同步传输方式传输指定大小的数据块。

5.6.3.2 Input Pin 的实现

Input Pin 有两个主要功能，媒体类型的协商以及 Video Sample 的接收。方案需要重载三个函数，来完成这两个功能。其中 CheckMediaType( )和 GetMediaType( )用于媒体类型的协商；Receive( )用于接收 Media Sample。

1. CheckMediaType( )：检查媒体类型。

该函数用于检查该 Pin 是否支持某种媒体类型。

2. GetMediaType( )：获取媒体类型。

该函数返回详细的媒体类型。

3. Receive( )：接收 Media Sample。

该函数用于从上方的 Filter 接收 Media Sample 数据。它有一个 Media Sample 的指针类型的参数。它的基本流程是：首先，锁住 Filter 的 Media Sample 接收，因为涉及的处理都是多线程的，所以在处理完一个 Media Sample 之前，需要挂起其他的 Media Sample 接收；然后，调用 CBaseInputPin 基类的方法 Receive( )，获取一个 Media Sample，保存到参数 MediaSamle 指针指向的缓冲；之后，调用 Video Send Filter 的方法 Send( )，将该 Media Sample 包含的媒体数据发送到网络上；最后，对 Filter 的 Media Sample 接收进行解锁。

5.6.3.3 Video Send Filter 的实现

1. Video Send Filter 中的 Send( )函数

这个函数是 Video Send Filter 中最主要的一个函数。视频数据在传输前应分割成适当大小的 ADU(Application Data Unit，应用数据单元)，分割的标准是使每个 ADU 成为一个独立的解码单元，并能封装于一个网络数据单元中，而且 ADU 在 RTP 包中所占的有效信息比要尽可能大。首先，MPEG-4 视频数据打包时要使不同的 VOP 应该分片为不同的 RTP 包，一个 RTP 包只包括与唯一 VOP 的时间相关的数据；其次是取值大小的问题，包过大或过小都会使网络性能变坏。一方面，太大的 ADU 在传输前要拆分成若干个网络数据单元，接收端又需要对其进行重组。这样会加大时延，且丢包对视频回放质量的影响较大。组成 ADU 的所有网络数据单元必须正确到达才能解码，而丢失 ADU 的概率又总是存在的。因此，若报文长度过大，就会降低数据报成功到达的概率，即降低网络的吞吐量，使网络性能变坏；另一方面，过小的 ADU 会产生很多有效信息比较少的包，浪费网络带宽。为了不造成 IP 碎片，减少错误传播，包长不能超过该网络路径的 MTU(Maximum Transmission Unit，最大传输单元)。因此怎样对 MPEG-4 视频数据进行组包对获得良好视频质量并有效利用网络资源是十分重要的。我们取包长为当前 VOP 大小与路径 MTU 值的较小值。最大的报文长度可以根据用户的需求和网络环境自己定义，但是不能超过网络的 MTU 限制。该函数接收从 Input Pin

传来的一个 Media Sample，对其中的媒体数据进行分片和 RTP 封装之后，利用 Net Sender 对象的 Send( )函数发送到网络。

2. CBaseFilter 的重载函数

需要重载的 CBaseFilter 类的主要函数有：

a. GetPin( )

该函数根据传入的索引值，获得 Filter 指定 Pin 并返回。由于 Video Send Filter 只有一个 Pin，所以直接返回它的 Input Pin。

b. Pause( )

Graph 和 Filter 开始运行，在进入 run 状态前，必须先进入 pause 状态。

该函数没有参数。它的基本流程是：首先，锁住 Filter 操作；然后，判断当前状态，如果当前状态是 run，则只需改写状态字段为 pause，如果是 Stop，则表示是要从停止状态进入运行状态，这时先调用 Net Sender 的 JoinMulticast( )函数准备好网络发送，再改写状态字段为 pause，然后判断 Input Pin 是否连接，如果已连接，则激活它(调用基类的 Active( )函数)。最后，将 Filter 操作解锁。

c. Stop( )

该函数终止 Filter 的运行，没有参数。它的基本流程是：首先，依次将 Media Sample 接收和 Filter 操作加锁；然后调用 Net Sender 的 LeaveMulticast( )函数推出网络发送；再改写状态字段为 stop；最后依次将 Filter 操作和 Media Sample 接收解锁。

3. ISpecifyPropertyPages 的重载函数

该接口用于确定属性页，在 Video Send Filter 中要重载的函数只有一个，就是 GetPages( )函数。它获得对象可用的属性页。函数在返回的结构体中填入可用的属性页的 Clsid。

4. CPesistStream 的重载函数

CPesistSteam 类的作用是为 Filter 提供可持久存在的属性，可以记录在存储的 Graph 中。它有三个函数需要重载。

a. WriteToStream( )和 ReadFromStream( )

这两个函数都只有 1 个 IStream 的接口指针参数。前者向该 Stream 依次写入网络发送的四个参数：组播地址、端口号、使用的 IP 地址和 TTL。后者从该 Stream 依次读出以上 4 个参数。

b. SizeMax( )

该函数获得当前 Stream 的最大字节数，返回网络发送 4 个参数所占用的字节数。

在网络发送的时候，如果需要添加新支持的媒体类型，只需要在 CheckMediaType( )和 GetMediaType( )函数中添加相应媒体类型就可以了。而对于原有编码器版本升级，甚至不需要修改任何代码就可以直接应用。

5.6.3.4 将 Net Send Filter 连接到 Filter Graph 的代码实现

```
……
//new 一个 Filter 类的实例
mVideoSender = new CVideoSenderFilter (NULL, &hr)
//获得 IbaseFilter 接口以增加一个引用计数
IbaseFilter *pFilter = NULL;
MviderSender->QueryInterface(IID_IbaseFilter, (void**)&pFilter);
//将 Net Send Filter 加入到 Filter Graph
pass = mServerVideoGraph->AddFilter(pFilter, L"Video Sender")
……
```

5.7 网络传输实现

5.7.1 组播传输实现

发送端和接收端通信是通过 Windows Sockets 编程实现的。Sockets 同时支持数据流 Sockets 和数据报 Sockets。数据流 Sockets 是 TCP 传输协议的接口，它定义了一种可靠的面向连接的服务，实现无差错无重复的顺序数据传输。数据报 Sockets 是 UDP 数据报服务的接口，它定义了一个无连接的服务。数据通过相互独立的包进行传输，包的传输是无序的并且不保证是否出错、丢失和重复。UDP 通信过程如图 5-9 所示：

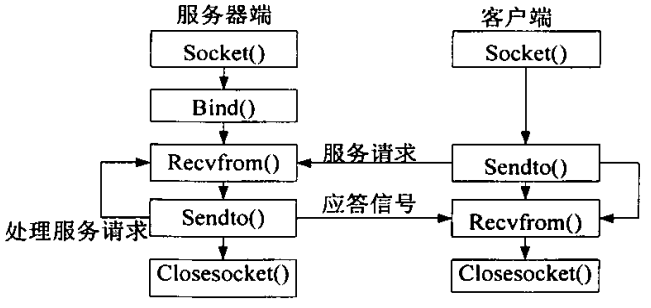

图 5-9 UDP 通信过程示意图

组播传输的实现

```
class CMulticast
{
private:
    SOCKET    mSckReceiver;//接收用的 Socket
    SOCKET    mMulticaster;//组播发送用的 Socket
    DWORD     mMulticastIP;    // 使用 Host byte order 的 IP 地址
```

```
    WORD      mMulticastPort;//端口号
    BOOL      mIsReceiving;//正在接收数据标记
    HANDLE    mRcvThread;//接收线程的句柄
public:
    CMulticastAdmin( );
    ~CMulticastAdmin( );
//设置/获取地址
    void SetMulticastIP(DWORD inIP);
    DWORD GetMulticastIP(void);
    void SetMulticastIP(const char * inIP);
    void GetMulticastIP(char * outIP);
//设置/获取地址端口号
    void SetMulticastPort(WORD inPort);
    WORD GetMulticastPort(void);
//创建/销毁组播 Socket
    BOOL CreateMulticaster(void);
    void DeleteMulticaster(void);
//创建/销毁接收用的 Socket
    BOOL CreateReceiver(void);
    void DeleteReceiver(void);
    // 数据通信
    BOOL Multicast(const char * inBuffer, long inLength);
    BOOL StartReceiving(void);//启动数据接收线程
    void StopReceiving(void);
private:
    void ReceivingLoop(void);//数据接收
    static DWORD WINAPI ReceivingThread(void * pParam);//接收线程
};
```

组播传输实现中采用了基于 WinSock2 的多线程技术通信方式。在 Video Send 和 Video Receiver Filter 中分别封装了 Socket 类 CNetSender 和 CNetReceiver。其中部分成员函数如下：

JoinMulticast(ULONG ulIP, USHORT usPort, ULONG ulNIC, ULONG ulTTL)：加入组播组；

LeaveMulticast( )：脱离组播组；

Send(BYTE * pbBuffer, DWORD dwLength)：送数据包；

CNetReceiver::ReadCompletion(CBuffer pBuffer, DWORD dwError)：读取数据完成；

CNetReceiver::ThreadProc( )：工作线程控制。

5.7.2 穿越防火墙方案

IP 视频会议要在 Internet 上传输视频音频数据，不可避免地要解决如果穿越防火墙和 NAT 的问题。

标准的 H.323 或 SIP 视频会议系统在有防火墙/NAT 的环境下实施时，一般采取以下几种方法：

1. 说服客户不使用防火墙/NAT，因为各种原因有很多客户最终不得不接受这种建议。
2. DMZ MCU：把 MCU 放在没有防火墙/NAT 等保护的 DMZ (Demilitarized Zone)区域，这样可以很好地保护他们的私有网络。放在 DMZ 区域的 MCU 要装两块网卡，一块提供访问私有网络的入口，另一块提供访问公网 Internet 的入口。这个方案的一个最大缺点是即使是进行点对点的呼叫也得需要使用 MCU。另外，如果在呼叫路径上有多个 NAT 设备，那么在每个 NAT 设备的位置都需要放置一个 MCU。
3. H.323 代理：H.323 代理其实是一种特殊类型的网关，通过中转来解决 NAT 问题。它一般实现标准的网守功能和 RTP/RTCP 多媒体流的代理功能。这种方案要在防火墙后放一个 H.323 代理并分配公有 IP 地址。配置防火墙允许代理和外部进行多媒体通信。如果网络路径有多个 NAT 设备，则需要在每一个使用 NAT 的地方放置代理。
4. 应用层网关：应用层网关(Application layer gateways, ALG)是能识别指定 IP 协议(如：H.323 和 SIP 协议)的防火墙。它更深层的分析应用层的数据，根据负载中的控制信息，分析哪一个端口需要打开，防火墙动态地打开那些被应用的端口，而另外所有的端口依然安全地保持关闭状态。如果有一个 NAT 被用来屏蔽内部 IP 地址，ALG 就需要一个代理。
5. 虚拟专用网(VPN)：VPN 技术在同一个 VPN 网内可以解决防火墙穿越问题，但它无法与位于公众网的终端用户进行通信。
6. 隧道穿透方案：隧道穿透方案由 Server 和 Client 两个组件构成。Client 放在防火墙内的私有网，它同时具有网守功能和代理功能，私有网内的终端注册到 Client 上，它和防火墙外的 Server 创建一个信令和控制通道，可以把视音频数据转发到 Server。Server 放在防火墙外的公众空间或者 DMZ 区域，相当于网守代理的角色。当一个终端呼叫防火墙外的另一个终端时，所有的数据包都通过 Client 路由到 Server，返回的数据也从 Server 通过 Client 路由回到终端。当呼叫被建立后，Client 确保所有必需经过防火墙的视音频通道保持开放，这样视音频数据可以通过这些防火墙上开放的通道进行传输。

以上方案，有的以牺牲系统安全性为代价，有的以增加系统复杂度和投资为代价。一般企业网都不想升级或者改动他们的防火墙和 NAT 设备的配置，故采用隧道穿透方案是相对合适的，隧道穿透方案的最大缺点是所有经过防火墙的通信都必须由 Server 中转，这会引起潜在的瓶颈。

还有一种简单的实现方案：RTSP-OVER-TCP。当采用 RTSP-OVER-TCP 传输数据时，流媒体数据的接收、回放和控制信息的发送都采用 TCP 通道。为了区分 RTP/RTCP 数据报文与 RTSP 控制报文，流媒体服务器在发送 RTP/RTCP 数据报文时，在 RTP/RTCP 头部添加了一个扩展头，以区别 RTSP 控制报文，RTSP 报文的内容在发送时并不改变。虽然这种方式能够成功穿越防火墙，但由于采用 TCP 协议传输数据会占用更多的网络资源，并会影响到网络中其它用户对网络资源的请求，严重影响会议规模。

由于任何防火墙都会将超文本协议 HTTP 作为一个基本的通信协议而打开，本文考虑设计一个组件来支持用标准的 HTTP 协议实现多媒体流传输。当存在防火墙时，系统自动采用 HTTP 协议进行数据传输。以这种方式穿越防火墙，使用户可以在不增加投资条件下安全地使用 IP 视频会议系统。

防火墙增强协议(FEP)允许任何应用程序通过防火墙，并且实现了在防火墙的安全性和防火墙的透明传输通道两个方面的统一。这里借签 FEP 的思想。实现的方法是，将 IP 视频会议的 UDP 包重新编码并置于 HTTP 上，按文献[40]所指定的 MIME 格式编码把视频数据包隐含在 HTTP 的消息体内，头信息隐含到 HTTP 的头内，包装成 HTTP 的格式。由于防火墙的目标是阻止外部攻击，故该方案并不违反实际防火墙的安全有效性。

5.8 Video Receive Filter 设计

Video Receive Filter 位于接收端，是一个 Source Filter，用于从网络接收视频流。经过处理后，将数据传送到下一级 Video Decode Filter，实时视频数据量比较大，为实现流式传输处理，保证平滑回放，Video Decode Filter 中建立了一个数据缓冲区，接收到的数据经 RTP 数据分组分解，提取 Media Sample，并将其置入缓冲区中，准备传送至 Video Decode Filter 的 Input Pin。

5.8.1 Video Receive Filter 的数据处理流程与数据流动

Filter Graph 中有两种基本的数据流模式：推模式(push Mode)和拉模式(Pull Mode)。推模式情况下，Source Filter 主动把数据往下传送；而在拉模式下，需要后面的 Filter 向 Source Filter 请求数据。在本文中，因为对于后面连接的是 Video Decode Filter，故采用推模式。

Video Receive Filter 的数据处理流程如图 5-10 所示：

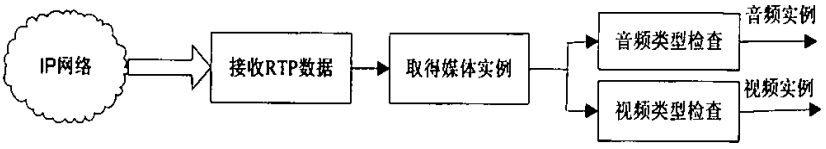

图 5-10 Video Receive Filter 的数据处理流程图

Video Receive Filter 在接收到 RTP 包后分析 RTP 包头，判断其版本、长度和负载类型等信息的有效性，然后按照 RTP 时间戳和包序列号等进行同步，整理 RTP 包顺序，重构视频帧。也就是说，它需要动态地一边从网络接收数据，一边将得到的 RTP 包进行重组。

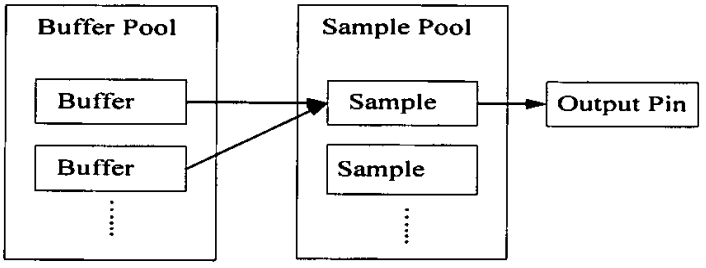

图 5-11 Net Receive Filter 的数据流动示意图

Net Receive Filter 的数据流动如图 5-11 所示。

Video Receive Filter 采用双缓冲池机制：Buffer Pool 和 Sample Pool。Buffer Pool 用作网络数据接收的缓冲池。每次接收前，先要向 Buffer Pool 申请一个 Buffer，如果暂时没有可用的 Buffer，则挂起，当有其他线程释放使用的 Buffer 时，Buffer Pool 就会通知该网络接收进程，并分配给它一个空余的 Buffer。通常一个 Buffer 对应一个视频数据报。由于接收到的数据报可能是经过分片的，因此必须将它重新组合成未分片前的状态，才可以恢复一个可以回放的 Media Sample。Video Receive Filter 通过分析每个数据报的 RTP 头来进行重组。在一定的时间阈值内如果重组完成，则将重组后的数据传递给 Sample Pool，并把 Buffer Pool 中对应的缓冲清空；如果超时，则丢弃这些数据报。Sample Pool 当发现没有可用的 Media Sample 时，也将挂起处理，直到有别的线程释放 Media Sample 为止。

RTP 包重组函数中有一个结构体链表，结构体有两个成员，一个是指向 Buffer 的指针，一个是该 Buffer 的序列号。首先当 RTP 报头中的 Marker 位为 0 或者序列号前 12 位与链表中 Buffer 的前 12 位不同(序列号前 12 位表示 Media Sample 的序列号，后 4 位表示该分片在 Media Sample 中的位置)时，表示一个新的 Media Sample 的开始。这时，要将链表中的 Buffer 的数据组合起来，传递给 Sample Pool。将 Buffer 插入链表时，根据序列号的低 4 位决定插入的位置。同时将前一个 Media Sample 链表中包含的媒体数据块拷贝到一个共同数据块中，

并赋予 RTP 包报头中的媒体属性信息。按照其中 RTP 数据报头中的序列号进行排列，重组完毕或者超时之后，交给 Media Sample 缓冲区。如果 Media Sample 头信息的首个分片丢失，丢失了头信息的 Media Sample 将无法工作，直接清空链表，丢弃链表中的 Buffer。

Net Receive Filter 与 Net Send Filter 的之间最大的区别就是 Net Receive Filter 需要自己实现 Media Sample 的管理。这是由于这两个 Filter 采用的是 PUSH 的数据流动方式。在这种数据流动方式中，都是由上方的 Filter 提供 Allocator，并进行 Media Sample 的管理。而 Net Send Filter 属于 Renderer Filter，它的下方已经没有 Filter 了，因此它不需要提供 Allocator，也不需要管理 Media Sample。

5.8.2 Video Receive Filter 的实现

5.8.2.1 Video Receive Filter 的结构

Video Receive Filter 的结构如图 5-12 所示：

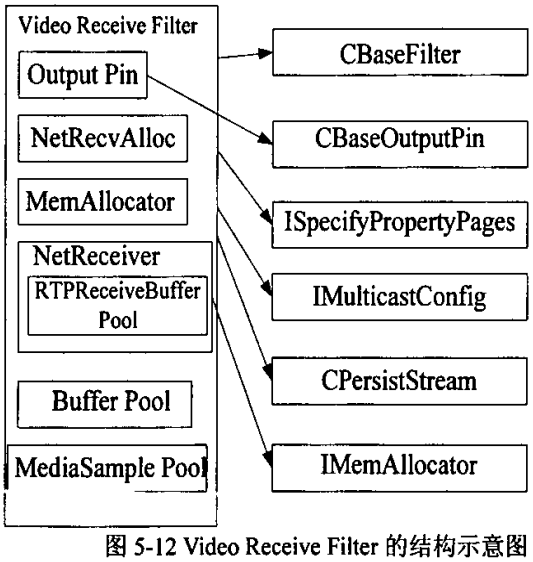

图 5-12 Video Receive Filter 的结构示意图

Video Receive Filter 的继承结构与 Video Send Filter 相同。都有 4 个相同的父类，分别是：CBaseFilter, ISpecifyPropertyPages, ImulticastConfig 和 CPersistStream。Video Receive Filter 与 Video Send Filter 的成员不同。Video Receive Filter 有 6 个成员：

1. Output Pin

Video Receive Filter 与 Video Send Filter 相反，它属于 Source Filter，没有 Input Pin，只有 Output Pin。本方案中的 Video Receive Filter 只有一个 Output Pin。它负责将 Media Sample 传递给下方的 Filter 的 Input Pin。

2. Net Receiver

Net Receiver 负责从网络接收数据。为提高效率，Net Receiver 采用的是异步

接收的方式。

3. MemAllocator

MemAllocator，内存分配器，它是一个 IMemAllocator 接口指针。这个成员用于接收 Filter 所使用的内存分配器。由于 Video Receive Filter 使用的是 PUSH 的数据流动方式(由上方的 Filter 主动向下方 Filter 传递数据的方式)，内存分配器是由 Video Receive Filter 自己提供的。

4. NetRecvAlloc

这个对象继承了 IMemAllocator 接口，它是 Video Receive Filter 自己的内存分配器。它的主要功能是进行 Media Sample 的上层管理。当 Graph 中下方的 Filter 与 Video Receive Filter 连接时，两者都将使用 NetRecvAlloc 进行 Media Sample 的管理。

5. Buffer Pool

接收缓冲池 Buffer Pool，这个缓冲池用于存放网络上接收的数据，直到数据被转化为 Media Sample 为止。位于 Buffer Pool 内的是 Buffer 对象，每个 Buffer 对象存放一个网络数据报。Buffer Pool 对池内的 Buffer 进行管理。由于网络接收是异步接收方式，Buffer Pool 管理 Buffer 的时候要考虑互斥情况。

6. Sample Pool

Sample 缓冲池 Media Sample Pool，这个缓冲池用于管理 Media Sample。它主要负责 Media Sample 的申请、分配和创建等工作。与 Buffer Pool 相同，它的实现也要注意资源互斥问题。

5.8.2.2 Net Receiver 的实现

Net Receiver 采用的是异步接收方式，在实现 Net Receiver 的过程中，将较多的用到资源互斥。涉及的函数有：

1. Activate( )：用于启动 Net Receiver 的网络接收。

该函数有 3 个参数，分别是组播地址、端口和使用的 IP 地址。

基本流程：首先加锁并调用 JoinMulticast( )初始化网络；然后启动线程 ThreadProc，如果线程启动失败，则调用 LeaveMulticast( )；最后，解锁。

2. ThreadProc( )：Net Receiver 网络接收过程的主体函数。

该函数没有参数和返回值，是一个无限循环。

基本流程：不断调用 WaitForMultipleObjectsEx 等待事件发生；如果接收到 Stop 事件，则跳出循环；如果超时，则调用过程进行一次网络接收，之后继续循环等待事件。

3. Stop( )：中止网络接收。

该函数无参数。

基本的流程是：首先触发 Stop 事件，之后调用 LeaveMulticast( )，最后等待 ThreadProc 线程的终止。

4. PendReads( )：执行一次网络异步接收

该函数有一个双字节型的参数，表示它在等待获取一个 Buffer 时所愿意等待的时间。该函数作为异步接收的操作实体，可能同时有多个线程正在运行该函数，一次函数调用可能会调用多次异步接收。为了避免出现过多的异步接收过程，需要控制异步接收调用的数量。为此设置了一个变量 m_ IReadsPended 来指示当前有多少异步接收过程尚未完成，对改变量的操作都是原子操作。

5. AsyncCompletionCallBack( )：WSARecv 异步接收完毕时的回调过程。

该函数有 4 个参数，3 个是双字节型变量，分别表示错误码、接收的字节数和标志位；另一个是指向 WSAOVERLAPPED 结构体的指针，该结构体是在 PendReads( )中调用 WSARecv( )时填写的，用于恢复原始的环境数据(如上下文)。

基本流程：首先获取 WSAOVERLAPPED 结构体中的上下文(这里是 Net 龙去脉 Receiver 对象实例)和接收数据的 Buffer；然后设定 Buffer 的实际数据长度为接收到的字节数；最后调用 ReadCompletion( )函数。

6. ReadCompletion( )：完成异步接收完毕的后处理。

该函数有两个参数：一个是指向 Buffer 的指针，另一个是双字节型的错误码。

函数的基本流程是：首先递减 m_IReadsPended；调用 PendRead_( )以尽快接收网络数据：如果没有错误的话，将 Buffer 的数据进行重组；最后，释放 Buffer。

Net Receiver 中加入和离开多播组函数(JoinMulticas( )和 LeaveMulticast( ))的实现过程与 Net Sender 中相似。

5.8.2.3 Buffer Pool 的实现

数据接收缓冲池 Buffer Pool 用于进行数据接收缓冲区的管理。数据接收缓冲的实体是 Buffer，每一个 Buffer 存放接收到的一个数据报文。

Buffer Pool 负责 Buffer 的创建、请求和销毁等功能。由于可能同时有多个线程向 Buffer Pool 申请获取 Buffer 而 Buffer 的数量有限，为了更高效的利用资源，Buffer Pool 通过建立一套申请机制来处理对 Buffer 的请求。

Buffer Pool 的 Buffer 管理机制主要通过三个链表来实现：第一个链表是可用 Buffer 链表，它存放现在可用的 Buffer；第二个链表是可用 RequestBlock 链表，当暂时没有 Buffer 可用时，就需要申请预定 Buffer。用来填写预定申请的结构体数据块(即 RequestBlock)需要从 RequestBlock 链表中获取；第三个链表是等待中 Request 链表，它用来存放目前已填写完毕的预定申请结构体，即处于等待状态中的申请。

5.8.2.4 Sample Pool 的实现

Sample Pool 用于进行 Media Sample 的管理。它负责 Media Sample 的创建，请求和销毁等功能。由于同时可能有多个线程向 Media Sample Pool 申请获取 Media Sample，与 Buffer Pool 类似，Media Sample Pool 也建立了一套申请机制来处理对 Media Sample 的请求。

5.8.2.5 NetRecvAlloc 的实现

NetRecvAlloc 用于 Media Sample 的上层管理，它位于 Media Sample Pool 之上，负责 Filter 与 Filter 之间对 Media Sample 的获取与访问。

NetRecvAlloc 继承了 IMemAllocator 接口。其中大部分功能已由基类实现。只有在获取 Allocator 属性的函数(GetProperties( ))中，需要将属性成员中的 Buffer 数量设为 Media Sample Pool 中 Media Sample 的总量，以及属性成员中的 Buffer 大小设为 RTPReceiverBuffer Pool 中的 Buffer 大小(即一个 Media Sample 数据量的最大值)。

5.8.2.6 OutputPin 的实现

Video Receive Filter 的 Output Pin 继承了 CBaseOutputPin 基类，主要功能是与下方的 Filter 进行媒体类型的协商，以及向下方 Filter 传递 Media Sample。

5.8.2.7 Video Receive Filter 的实现

Video Receive Filter 与 Video Send Filter 相同，它们都继承了 CBaseFilter 基类、ISpecifyPropertyPages 接口、IMulticastConfig 接口和 CPersistStream 类。Video ReceiveFilter 所需要重载的函数也与 Video Send Filter 相同，这里不再重述。

Video Receive Filter 另外需要实现的一个重要成员函数是 ProcessBuffer( )。该函数将从 RTPReceiveBuffer Pool 传递过来的数据块封装进一个 Media Sample，然后传递给下方的 Filter。ProcessBuffer( )函数有三个参数：第一个是指向传入的数据块的 BYTE 型指针；第二个是数据块长度；第三个是指向媒体数据的属性结构体的指针。属性结构体的成员主要有时间戳和标志位等。

Video Receive Filter 同样也定义了外部接口，声明了三个函数：

SetPort( )，设置接收端的通信端口；

JoinMuticast( )，加入组播组；

SetTTL( )，设置组播数据生命周期。

5.9 Video Renderer Filter 设计

为减少视频数据处理时延、减轻数据的抖动和保证视频的流畅，在 Video Render Filter 中也采用双缓冲队列技术。这里建立两个队列，一个是空闲的缓冲队列 FreeList，用以接收存放数据；另一个是尚未处理的数据缓冲队列 DataList，根据 RTP 包头的“包序号”字段顺序回放。当接收到数据后，从 FreeList 队列头上取出一个缓冲块，存放数据，然后将这个缓冲块加入到 DataList 队列的尾部。回放时，从 DataList 队列的头上取出一个缓冲块，读取数据，再将读完的缓冲块加到 PoolList 的尾部，等待再一次的数据接收。具体实现中只需要对 DirectShow SDK 中的标准 Renderer Filter 稍加改写即可。如果有音频信息存在，还要考虑为视频音频设置公共参考时钟来实现流间同步，在此不作赘述。

5.10 本章小结

本章给出了系统的总体设计和详细设计。利用 DirectShow 技术实现了视频数据在网络上的实时传输，给出了 Video Capture Filter、Video Send Filter 和 Video Receive Filter 的具体实现。

6 视频 QoS 控制策略

目前的 IP 网络只能提供“尽力而为”(best-effort)的服务，不能为视频的流式传输提供服务质量(QoS，Quality of Service)保证。主要表现在：IP 网络没有预留机制来满足视频传输的带宽要求；传统的路由器不能参与拥塞控制，过大的流量可能会导致实时视频吞吐量的下降，甚至拥塞崩溃；视频传输有严格的端到端时延要求，如果传输时延超过回放时刻，该包无效，等同于丢失，IP 网络无时延保证；常常会因网络拥塞导致大量丢包而严重影响视频质量；带宽、丢包率、时延和时延抖动等网络状况和网络特性都随时间而不断变化。

视频的控制策略可以从网络层和应用层两方面考虑。因为视频业务是端到端的服务，要求网络上的结点都支持 QoS 通常是不可能的。所以，网络层的解决方法虽然在技术上简单有效，但是在实际工作时相当困难。基于端系统的应用层级别上的 QoS 控制策略是一个简单可行的方法，它不需要对当前的网络和路由器进行任何改造，由端系统应用程序来实现 QoS 控制，将整个网络看成一个“黑箱”，通过对网络状态的估测来获取网络状态信息，然后按某种策略调节发送端的输出业务流量与可用带宽匹配来防止拥塞的发生、减少丢包和降低时延。

6.1 端系统 QoS 控制

端系统的 QoS 控制策略主要有拥塞控制策略和差错控制。

6.1.1 拥塞控制策略

由于交换结点缓存空间不足、网络链路带宽不够、交换设备 CPU 处理能力弱和广播(组播)风暴等原因，网络中传送的数据量会在流量高峰或过载时超过网络处理能力，造成网络服务性能下降，这种现象就是拥塞。网络产生拥塞的根本原因在于用户提供给网络的负载大于网络资源容量和处理能力。

网络拥塞是影响 IP 网络视频通信技术的一个重要因素。网络拥塞除了减少信道的有效吞吐量外，还会引起分组丢失、延迟增长和延迟扰动等现象。所以，必须通过采用限制网络载荷的方法进行拥塞控制。拥塞控制主要是通过控制数据流来避免网络过载。

当网络发生拥塞时，往返时间(Round Trip Time, RTT)和丢包率会增大；当网络负载较轻时，RTT 和丢包率相对较小。因此，网络中可用带宽的估计可由端系

统通过 RTT 或者丢包率来间接得到。Dabbous 在文献[19]中证明使用基于 RTT 的反馈拥塞算法的链路吞吐率低于使用丢包率的反馈拥塞算法的链路吞吐率。因此，可以利用某段时间内的丢包率作为判断网络是否拥塞的标准。

对于视频流来说，拥塞控制实际上就是速率控制。通过速率控制可以使发送端发送出的码流速率与网络可利用的带宽相匹配，从而缓解网络的拥塞，使得网络进入一个良性循环。速率控制主要有三种：基于发送端的速率控制、基于接收端的速率控制和混合的速率控制。

1. 基于发送端的速率控制

如果传输码率和网络带宽相匹配则包丢失率会大大下降。基于发送端的速率控制通过调整发送端的编码速率来适应网络的可用带宽。实现时通常需要有一个反向信道，从接收端监测网络的状态，把网络的状态信息返传给发送端。发送端根据网络状态信息进行编码速率调整，可以采用试探的方法和模型法。

1) 基于试探的方法：首先设定一个包丢失率的阈值 pth，在接收端检测包丢失率 ρ，并将 ρ 通过反向信道传回编码端，编码时可以采用以下的策略进行码率调整：

如果($\rho \le pth$)，则 $r = min((r + AIR), MaxR)$；否则，$r = max((a \times r), MinR)$

其中 r 是传输速率，ρ 是包丢失率，$MaxR$ 和 $MinR$ 分别是最大和最小传输速率。

2) 基于模型的方法：采用一个模型来估计网络的带宽，主要是计算网络的 TCP 吞吐量 $\lambda = \frac{1.22 \times MTU}{RTT \times \sqrt{p}}$。其中 λ 是 TCP 连接的吞吐量，MTU 是连接的最大包长。RTT 是包的往返时间，ρ 是包丢失率。计算速率控制后的视频流能够像 TCP 连接一样共享带宽。

2. 基于接收端的速率控制

接收端根据网络状态增加或减少信道，主要用于分级视频编码。在这种控制方法中，编码部分本身并不作调整，只是在发送层进行调整。这种控制策略在组播系统中工作得很好，而基于发送端的速率控制主要用于单播的系统中。基于接收端的速率控制也采用试探的方法和基于模型的方法。这两种方法与上述基于编码端的解决方法相同，只不过每一次发送的码率是以分级码流的级为单位进行调整。

3. 混合的速率控制

编码器的发送端根据反向信道的信息调整速率的同时，接收端增加或减少信道。与基于发送端的方法不一样的是：混合方法采用了多信道，每一个信道的速率不是固定的而且可以根据网络的拥塞状况调整。

6.1.2 差错控制

现有 IP 网络不提供 QoS 服务，拥塞控制只能减少丢包率，所以丢包是不可避免的。通过差错控制机制可以提高在网络丢包情况下视频流的 QoS。差错控制机制的主要目的就是解决传输过程中的丢包问题，提供端到端的出错率保证。视频的差错控制机制可以分为三类：前向纠错(FEC)、重传(ARQ)和错误隐藏[19,39]。

1. FEC

FEC 的原理就是在压缩图像流中增加冗余比特，当包丢失时利用冗余的比特来恢复数据。根据实现的方法 FEC 可以分为三类：信道 FEC、信源 FEC 和信源信道联合编码。

1) 信道 FEC

信道 FEC 就是将数据分成 k 个比特的组，再按编码规则增加冗余比特，构成 n 个比特的数据组。数据进行传送，只要接收端收到了 $K(K>k)$个比特的数据时，就能将数据中的误比特全部准确恢复。终端收到了其中任何 K 个比特都能准确恢复。如果网络出现拥塞时，就可以随意丢掉几个包，只要收到的包超过 k，就可以正确解码。因为每一个接收端的误码不一致，接收端根据各自的误码情况进行独立解码。但是 FEC 也有弊端：增加了传输数据的码率。每一个包增加了 $n-k$ 比特；要等待收完 k 个比特才能进行编码，要等解完 k 个比特才能播放；如果出现了误码还要进行纠错处理，这些都会造成时延；另外，它缺乏自适应性。因为网络状态不是固定的，如果误码超过了 $n-k$ 就不能正确纠错；而误码低于 $n-k$ 时，信道编码的利用率就比较低。

2) 信源 FEC

信源 FEC 和信道 FEC 相似，都是通过增加冗余信息来实现的。信道编码是对输出图像的码流进行编码，信源编码多以图像压缩的形式。因此当出现丢包时，信道编码能够精确恢复，而信源编码只能恢复一个降级的图像。信源 FEC 的优势在于低延时。信源 FEC 也存在传输效率降低、延时大和缺乏对网络自适应性的缺点。

3) 基于信源信道的控制

信源 FEC 和信道 FEC 同时考虑。由于信道的传输能力即传输比特数是一定的，所以需要将比特在信源和信道之间合理分配。实现时，通常在接收端进行 QoS 监测，将网络的状态经过反向信道回传给编码器端。编码器根据信道的带宽和误码特性确定信源和信道的比特分配：信源编码选择一个合适的速率控制方法，使其输出码率符合目标码率；信道编码也是选择一个合适的编码方法能够满足分配的比特，适合信道的误码特性。

2. 重传

数据包在传送过程中出现误码或丢失后，重传是一种最简单有效的解决途径。如果判断重传后数据包能够低于时延的阈值，就可重传。判断可以在接收端或发送端进行。

3. 错误隐藏

人眼能够承受一定的数据变形。所以，当传输包发生丢失时，可以采用错误掩盖技术来消除丢包引起的图像降级，得到更好的图像质量。错误隐藏总体上可以分为两类：时间掩盖技术和空间掩盖技术。前者利用前一帧相同位置或运动矢量指向的位置来取代，后者采用空间上的相邻的宏块来取代出错的宏块。

6.2 本文采用的端系统 QoS 控制策略

考虑到视频会议中视频解码回放对 CPU 时间的占用远远大于音频，且丢弃音频帧会使用户难以忍受，因此网络拥塞主要针对视频流。本文在应用层采用端系统 QoS 控制策略来提高视频流实时传输质量，该策略也包括拥塞控制和差错控制两方面。

6.2.1 拥塞控制策略

在端系统上进行应用层的拥塞控制，主要工作是从接收端和发送端两方面调整视频速率，实现高效率和低失真地传输视频数据，而且它不需要路由器和网络的支持。

6.2.1.1 接收端的拥塞控制

本文在接收端采用改进的随机提前检测(Random Early Detection：RED)的拥塞控制方法。

RED 的思想是：系统每接收到一定数量的 RTP 包时，对缓冲队列进行一次检测，计算出当前缓冲时间 *BufTime*。预先设定出缓冲队列时间的两个门限值 *MinBufTime* 和 *MaxBufTime*。若 *BufTime* 小于 *MinBufTime*，说明缓冲队列工作正常，无须进行拥塞控制；若 *BufTime* 介于 *MinBufTime* 和 *MaxBufTime* 之间，则按照一定的概率 α 丢弃数据包；若 *BufTime* 大于 *MaxBufTime*，说明系统缓冲队列过长，此时将到达的数据包全部丢弃。

先来分析 MPEG-4 视频编码流的特点。一个 MPEG-4 视频流内包含一个或多个视频对象，每个视频对象序列包含多层视频流信息，一层视频流就是一个 VOP (Video Object Plane)序列，VOP 是视频数据的基本单元。MPEG-4 视频流存在三种 VOP： I-VOP、P-VOP 和 B-VOP。I 帧利用图像自身的相关性压缩，提供压缩数据流中的随机存取点，编码不需要其他帧的图像作参考；P 帧是对过去帧内图像或者过去预测得到的图像用动态补偿预测技术进行的编码的图像；B 帧用作双向预测图像，压缩率最高，且不作为预测的参考图像；P 帧和 B 帧以 I 帧

为参考，所以解码必须以一个 I 帧开始。如果 I 帧丢失，即使 I 帧以后的 P 帧和 B 帧存在也不可能用于正常解码，在回放时必定造成视频花屏。

根据 MPEG-4 视频流的特点，本系统为视频帧设置丢弃优先级。丢弃优先级从高到低依次为：B 帧、P 帧和 I 帧。RTP 包在缓冲队列中时无法判断 RTP 包所属的帧类型，需要组帧后才能判断视频帧的类型，本文丢帧的判断依据是视频帧的类型及其丢弃优先级，需要由解码回放线程丢帧。在 *MinBufTime* 与 *MaxBufTime* 之间增加 *BLoseBufTime* 和 *PBLoseBufTime* 两个时间点。接收端根据以下规则决定是否丢帧和丢帧类型：

1. *BufTime*＜*MinBufTime*：说明此时系统工作正常，视频解码回访线程读出已组帧的视频帧后解码回放视频帧；
2. *MinBufTime*≤*BufTime*＜*BLoseBufTime*：视频解码回放线程读出已组帧的 B 帧后将其按概率 α 丢弃，I 帧和 P 帧正常解码回放；
3. *BLoseBufTime*≤*BufTime*＜*PBLoseBufTime*：视频解码回访线程读出已组帧的 P 帧和 B 帧后将其按概率 α 丢弃，I 帧正常解码回放；
4. *PBLoseBufTime*≤*BufTime*＜*MaxBufTime*：将所有帧按照概率 α 丢弃；
5. BufTime≥MaxBufTime：说明此时系统过度繁忙，接收线程将接收到的 RTP 包直接丢弃，不放入数据缓冲区中。

流程如图 6-1 所示：

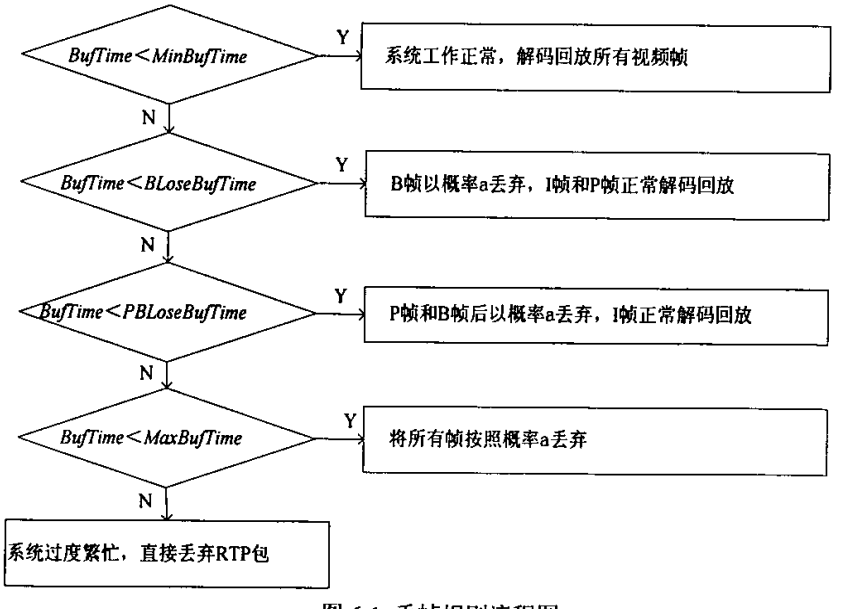

图 6-1 丢帧规则流程图

本系统在具体实现中，以上 *MinBufTime*、*BLoseBufTime*、*PBLoseBufTime* 和 *MaxBufTime* 四个变量的初始值分别设为：5 秒、7 秒、9 秒和 10 秒。

另外，在视频接收端的过滤器设计中，我们规定从视频数据的关键帧即 I 帧开始接收，如果首先到达的视频帧为 P 帧或 B 帧，将其丢弃，直到 I 帧到来为止。

这样，可以在一定程度避免接收端视频画面出现花屏或者马赛克现象，以提高视频效果。

6.2.1.2 发送端的拥塞控制

在视频会议进行期间，各与会者之间周期性地传送 RTCP 包。随着与会成员的增加，RTCP 报文的发送间隔要逐渐增大，才能保证系统中传输的 RTCP 报文最多占用整个视频带宽的 5%。每个 RTCP 包的时间间隔基值应大于 5 秒，会话开始时发送的 SDES 包文的时间间隔基值应大于 2.5 秒。发送端可以利用 RTCP 包中的 RR 报文的信息动态地改变发送速率，通过 SR 报文可以得到当前网络的丢包率。丢包率的计算步骤如下：

1. 获取上次发送 SR 报文后的 RTP 序列号。
2. 获取本次发送 SR 报文时最大的 RTP 序列号。
3. 丢包率 = 发送 SR 间隔的 RTP 序列号之差/发送间隔已收到的 RTP 包的总个数。

在进行 MPEG-4 视频传输时常采用 AIMD(additive increase and multiplicative decrease)拥塞控制算法。首先设定一个丢包率的阀值 *Pth*，把网络负载状态划分为两个域：拥塞和未拥塞；然后由发送端根据接收端反馈的丢包率 ρ 进行网络状态判断，并据此调整发送速率。这里对 AIMD 拥塞控制算法做适当的改进。

在视频发送过滤器中，实时接收由接收端反馈的 RTCP 包，并根据计算出的丢包率。发送端的速率调整策略如下：

1. 如果连续 N 个时间单位内丢包率大于 2%，则按图 6-2 来调整视频数据的发送方式(设正常发送速率为 R)：

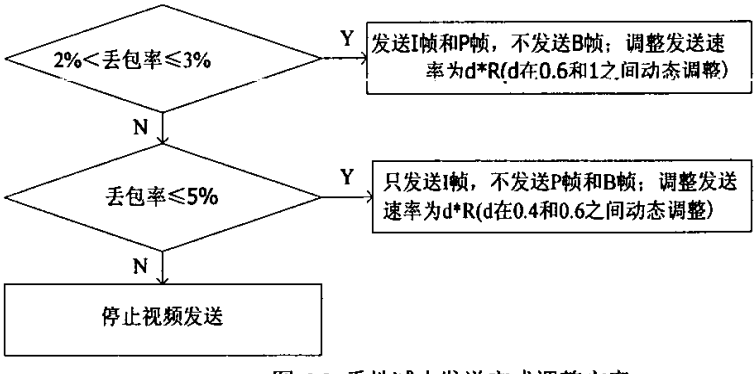

图 6-2 乘性减少发送方式调整方案

其中，d 为衰减因子。

2. 在当前发送速率小于 R 的情况下，如果连续 M 个时间单位内丢包率小于 2%，则调整发送速率为当前速率加上 ΔR (ΔR 为速率增量，$\Delta R \ll R$)，并根据网络状况适当发送 P 帧或 P 帧与 B 帧。

这里，以连续 N 和 M 个单位时间作为判断依据，目的就是避免由少数丢包引起的发送速率的锯齿振荡。

6.2.2 差错控制

在组播网络中最容易发生包丢失的地方是在从源结点到骨干网(称之为源链路，Source Link)和从骨干网到接受结点(称之为尾链路，Tail Link)的数据链路上，局域网和骨干网内部的包丢失率可以忽略不计。本文的差错控制策略从接收端和发送端两个方面考虑。在发送端的视频编码选择上，选用具有很强容错能力的MPEG-4 编码；在重传策略上，采用简单可行的重传策略：延迟受限重传与基于优先级重传相结合的策略，即优先重传那些能在最后期限内到达的基本层数据，再重传那些能在时间阈值内到达的增强层数据。

6.2.2.1 MPEG-4 的容错机制

本文的视频编码采用 MPEG-4 标准，它本身的容错特性包含相互联系的三方面技术：重新同步、数据恢复和错误隐藏。

1) 重新同步

采用重新同步技术的目标是在解码器和码流之间实现数据同步。当码流中出现了错误或者已经检测到错误时，两者能够重新同步，通常出现在错误的同步点和重新建立的同步点之间的数据都被丢弃。重新同步的技术能够有效的定位解码器所丢弃的数据，并用来恢复数据，是一种通过增强隐藏错误能力来实现差错控制的技术。

2) 错误隐藏

编解码器中的错误隐藏策略的有效性主要依赖于重新同步方案的性能，如果重新同步方法能够有效地定位错误，则错误隐藏问题就变得容易处理。对于低码率低延迟的应用，重新同步方案使用一个简单隐藏策略就可以取得较好的效果。

3) 数据恢复

在重新建立同步之后，数据恢复技术可以用来恢复那些丢失的数据。它通过在编码时加入能够容错的冗余信息，来实现一定的纠错功能。也就是说通过在一定程度上降低编码效率来提高容错性能。

采用 MPEG-4 编解码技术的视频流从本身机制上就能够提供很强的容错能力，再加上网络传输的质量保证，则可以进一步提高回放系统的稳定性和适用能力。

6.2.2.2 重传策略

由于网络延迟的影响和网络带宽的限制，IP 网上重传所有出错的包对于实时视频应用来说是不现实的。考虑 MPEG-4 标准采用的压缩方法，视频流中的 I 帧比 P 和 B 帧重要，本文采用有条件地选择重传重要数据。通过优先重传恢复 I 帧的数据包，能提高视频图像的峰值信噪比 PSNR，减少丢弃帧的数量，且能减少网络上的数据传输量，满足实时应用的需要。

1. 在发送端根据 RTP 协议，将 MPEG-4 数据流打包，然后通过 IP 网络发往接收端；

2. 在接收端：

1) 从网络接收 RTP 包，把包解开；

2) 检查包的内容，判断在 I 帧中是否有丢包发生：若 I 帧没有丢包，则不发重传请求；若 I 帧丢包，则根据其它帧的信息求出该帧的回放时间，即下次重传该包必须到达客户端的时间期限，再把包的序号及期限信息用 RTCP 发回发送端；

3. 发送端接收到重传信息，判断时间期限是否超过时间阈值。若没超过，则优先重传接收端所需的 I 帧数据包；如果网络带宽和时间允许，可以考虑重传 I 与 P 帧；只有网络带宽足够宽而且用户要求高清晰度时，才重传 I、P 与 B 帧的数据包。

为了实现以上的重传功能，本文对 RTP/RTCP 协议进行了两方面的扩展：在 RTP 协议头部设置两位为数据包设置优先级：I 帧为 10；P 帧为 01；B 帧为 00。在 RTCP 协议头部设置两个字节表示要求重传的包及该包的时间期限。

6.3 实验测试

6.3.1 QoS 实验测试

为了测试本文提出的视频组播方案和端系统的 QoS 控制策略的实际效果，这里给出一个简单的模拟实验。实验的软件环境：在 Win2000 下选用 NS-2 作为模拟实验的平台，采用 Cygwin 在 Windows 环境中为 NS 提供模拟 UNIX 环境，在 Cygwin 环境中安装使用 ns allinone 2.27。

在本次实验中，网络由三个结点组成，其拓扑结构如图 6-3 所示。其中结点 1 是发送端，结点 2 和结点 3 是接收端。本实验的目的是测试结点 3 在网络带宽动态变换的情况下端系统 QoS 控制策略能否降低丢帧率。

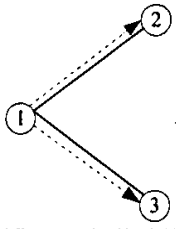

图 6-3　拓扑结构图

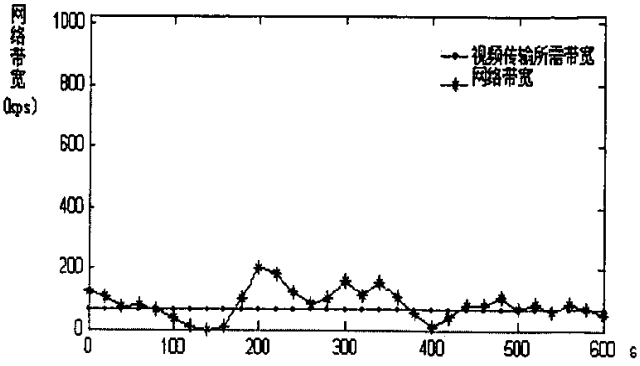

图 6-4　网络可用带宽动态波动图

结点 1 至结点 3 之间的链路的带宽为 1Mbps，其中，可用带宽的波动是随着时间动态变化的。本文为了简单起见，使用 NS-2 中的流量发生器模拟了一个具有固定发送速率的组播源(一般来说，视频组播应用实际所占用的带宽是随着时间的改变而动态变化的)。假设结点 3 接收的组播视频数据所用的带宽是 64kbps。视频组播数据所占用的带宽和可用带宽的动态关系如图 6-4 所示。采用端系统 QoS 控制和未使用端系统 QoS 控制的丢包率的比较如图 6-5 所示，可以看出，端系统 QoS 控制机制可以明显降低视频帧的丢失率，也在一定程度上减轻了网络拥塞程度。

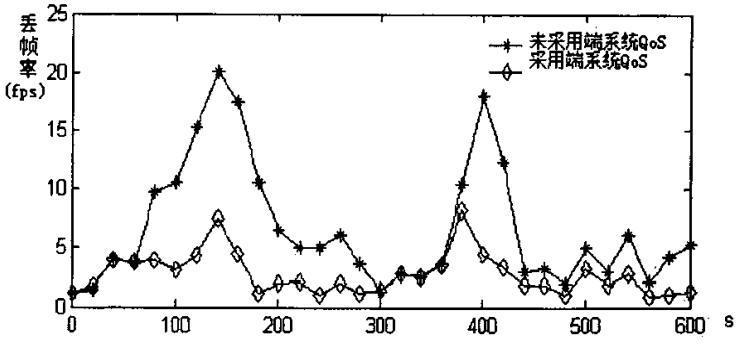

图 6-5 丢包率的比较图

6.3.2 视频效果测试

为测试整个视频传输系统的直观效果，由三人进行了视频通信实验。所用的三台 PC 机硬件配置分别为：

1. CPU：Intel Celeron 1.7G，内存：512MB，摄像头：福晶 FJ-310B；
2. CPU：AMD Sempron 2200+，内存：512MB，摄像头：福晶智慧鸟 PCC-130A；
3. CPU：Intel Celeron 2.0G，内存：256MB，天兴阳光 TX-9838。

在整个会议过程中，视频效果良好。图 6-6 为视频过程中随机抓捕的一个画面：

图 6-6 视频过程中的一个画面

6.4 本章小结

本章主要讨论视频流的 QoS 保障问题，从拥塞控制和差错控制两方面给出了应用层上的端系统 QoS 解决方案，实验结果表明该方案能在一定程度上减轻网络拥塞程度，也能明显降低视频帧的丢失率。视频传输系统的运行效果良好。

7 总结

7.1 总结

在基于 IP 的软件视频会议系统项目中，本人完成了视频实时传输系统的研究和实现：根据视频会议和 IP 网络的特点[44,45]，选取了 UDP 上的 RTP/RTCP 作为传输协议；提出了基于分层排列图的混合式组播并作为视频数据的传输方式；采用 MPEG-4 视频压缩编码、DirectShow 技术和 WinSock 编程技术完成视频数据在 IP 网上的采集和实时传输；从拥塞控制和差错控制两方面为视频传输提供了一定的 QoS 保证。系统具有很强的扩展性，在此基础上进行简单的扩充就可完成视频音频的同时采集与回放功能，也可扩展为其它的专用系统。

7.2 今后的研究方向

本文对 IP 视频会议系统的视频实时传输部分做了一些研究，取得了一定的成果，但仍然存在着许多问题有待于进一步研究。今后主要的研究方向和内容包括：

1. 安全保障是 IP 视频会议系统能否市场化的关键因素，视频数据流的加解密以及不同保密权限等问题有待做进一步的研究；
2. 穿越防火墙方案的实现。IP 视频会议需要提供穿越防火墙功能，具备面向不同网络的支持功能，文中提出的方案有待于进一步实现；
3. 实现基于 B/S 结构的视音频传输，使用户只要具备上网条件和简单的视音频设备就能参加视频会议。

参考文献

[1] CHEN J,HANG H. Source model for transform video coder and its application part 1.11[J].IEEE Trans,1997,CSVT-7(2):287-311

[2] R. Rej aie, M. and D. Quality adaptation for congestion video playback over the Internet. Proc. SIGCOMM, Aug. 1999,99:189-200.

[3] Noguchi T,Yamamoto M,Ikeda H. Reliable multicast protocol applied local FEC[J].IEEE International Conference .2001,8(8):2348-2353

[4] KATTO J, OHTA M. Mathematical analysis of MPEG compression capability and its application to rate control [A].Proceedings International Conference on Image and processing[C].USA:IEEE,1995,11:555-559

[5] Jack Brassil, Henning Schulzrinne.Structuring internet media streams with cueing protocols.IEEE/ACM Transactions on Networking,2002,10(4):466-476

[6] Markku Korpi, Vineet Kumar.Supplementary Services in the H.323 IP Telephony Network.IEEE Communications Magazine,1999,37(7):118-125

[7] B. Lupini and V. Cuperman,"Vector Quantization of Harmonic Magnitudes for Low Rate Speech Coders,".Proc. IEEE Globecom Conf. 1994,vol 2:858-862

[8] Wu Feng, Li S-P, Zhang Y-Q.A framework for efficient progressive fine granularity scalable video coding. IEEE Trams. Circuit Syst. Video Technol. 2001,11:332-344

[9] LEE K W, PURI R KIMTE. An integrated source coding and congestion control framework for video streaming in the Internet[C].Proc IEEE INFOCOM 2000:747-756

[10] Touradj Ebrahimi,etal.MPEG-4 nature video coding overview.Signal Processing: Image Communication.2000,15:365-385

[11] A. Das and A. Gersho, Variable dimensional Spectral coding of Speech at 2.4kb/s and below with Phonetic Classification. in Proc. IEEE Int. Conf. ASSP. 1995: 492-495

[12] Joerg Widmer, Robert Dendaand Martin Mauve.A Survey on TCP-Friendly Congestion Control IEEE Networks. May/June2001:28

[13] Shihua Wang, Andrew Sekey, Allen Gersho.An Objective Measure for Predicting Subjective Quality of Speech Coders. IEEE Journal on Select Area in Communication. vol. 10, no. 5 :819-829

[14] Amitava Das, Ajit V. Rao, Allen Gersho.Variable-Dimension Vector Quantization. IEEE Signal Processing Letters. 1996.vol. 3, no. 7:200-202

[15] Stuhlmuller K,Farber N. Analysis of video transmission over lossy channels. [J]IEEE Journal on Selected Areas in Communications.2000,18(6):1012-1032.

[16] PENDAKARIS D, SHI S. ALMI: an application level multicast infrastructure[A]. Anderson T. The 3rd USENIX Symposium on Internet Technologies and Systems[C]. San Francisco,CA, USA: USENIX Association, 2001: 49-60

[17] Peyravian M.Roginsky,A.Zunic N. Hash-based encryption system. Computer& Security,1999,18(4):345-350.

[18] C.K. Yeo, B.S. Lee, M.H. Er. A survey of application level multicast techniques. Computer Communications (2004) 27:1547–1568

[19] Y Wang, Q.F.Zhu. Error control and concealment for video communication. A review.

Proc.IEEE, May 1998, Vol 86: 974-997.

[20] Y. H. Chu, S. Rao, and H. Zhang, A Case for End System Multicast .ACM SIGMETRIC, June 2000: 1-12.

[21] 田瑞雄,向哲,李星. 基于分层排列图结构的流媒体应用层组播系统. 清华大学学报(自然科学版),2004,44(4):493-497.

[22] 苏育挺,张春田.极低码率视频编码中运动估值技术的研究.通信学报.1999,20(6):8-14

[23] 金光,赵一鸣.应用流式媒体服务实现基于桌面的实时监控和视频会议. 计算机应用研究,2003,20(l):119-121

[24] 郑阿奇. 数字视频音频实时同步传输探索与实现. 南京师范大学学报(工程技术版), 2002,2(3):16-20

[25] 徐昌彪.IP组播及其核心技术探讨.计算机应用研究,2001,18(8):49-53

[26] Schulzrinne H, Rao A. RTP: A Transport Protocol for Real-Time Applications. RFC 1889,1996:1-44

[27] 李琚晟, 余镇危, 潘耘等. 应用层组播综述. 计算机应用研究, 2004, (11):14-17.

[28] 周金星,吉逸,金胜昔.Internet 多媒体数据流实时传输协议的研究及其应用[J].计算机工程与应用,2002,(21).164-167

[29] 温小明,吴志刚,.基于 DIRECTSHOW 的 MPEG-4 视频传输系统的研究与实现[J].计算机与信息技术,2005,(8).23-26

[30] 周亚文,饶若楠,张保稳.视频信号网络传输服务质量和实时传输协议[J].计算机工程,2004,(S1).247-249

[31] 郑力明,张会汀,刘伟平,肖志明,黄伟英.基于 IP 组播技术的分布式视频会议系统的设计与实现[J].计算机工程与应用,2003,(2).153-155

[32] 王安静,金赞,杜春华,王静.Windows 下视频传输纠错技术的实现[J].计算机应用研究,2004,(5).237-260

[33] 阳书平,林行刚.一种适用于网上视频监控的 MPEG-4 码流技术.清华大学学报(自然科学版),2003,(09)1272-1275

[34] 管国辰,邢卫,鲁东明,.安全实时流媒体系统的设计与实现[J].电视技术,2006,(2).71-73

[35] 陶蒙华,黄孝建,沈树群.MPEG 视频源的两种基于 RTP 的建模及其比较[J].北京邮电大学学报,2002,(1).53-57

[36] 许延,常义林,刘增基.多媒体同步系统的缓冲区补偿技术[J].计算机学报,2003,(4).484-490.

[37] 谢亚光,章琦,刘济林.基于 Microsoft DirectShow 的多媒体应用程序开发.计算机应用研究,2003,72-75

[38] 卢选民,史浩山.基于延时估计和漏桶算法的自适应多媒体同步控制[J].计算机工程与应用,2003,(17).188-191

[39] 甘亚莉,钟文丽.视频通信中的错误隐藏技术[J].计算机应用研究,2003,(2). 8-10

[40] Eastlake, D. IP over MIME, Work in Progress.

[41] 席庆等.VisualC++6.0实用编程技术. 中国水利水电出版社,1999

[42] 陈坚，陈伟.Visua1C++网络高级编程. 人民邮电出版社,2001

[43] 张明德,王永东. 视频会议系统原理与应用.第1版. 北京:北京希望电子出版社,1999

[44] 蒋林涛.多媒体通信网.第1版.北京:人民邮电出版社,1999

[45] Andrew S.Tanenbaum著,熊桂喜,王小虎译.计算机网络.第3版.北京:清华大学出版社,2002

[46] 余松煌,张文军,孙军.现代图像信息压缩技术.第1版.北京：科学出版社,1998

[47] 陆其明.DirectShow 实务精选.第1版.北京: 科学出版社,2004

[48] 陆其明.DirectShow 开发指南.第1版.北京: 科学出版社,2003

[49] Carpenter B. , Internet Transparency, RFC 2775, February 2000

[50] ISO/IEC JTC1/SC29/WG11, *Coding of moving pictures and audio*, N2424, October 1998

[51] http://ftp.intron.ac/pub/algorithm/MPEG4/opendivx/encore5Osrc.zip

[52] Postel, J., Transmission Control Protocol, STD 7, RFC 793,September 1981

[53] http://www.chiariglione.org/mpeg

[54] http://www.networksorcery.com/enp/protocol/rtcp.htm

[55] http://www.freesoft.org/CIE/RFC/1889/13.htm

[56] http://en.wikipedia.org/wiki/H.264

攻读硕士学位期间公开发表论文

宋柱芹,熊建设,徐洪梅,杨小辉.一种适用于IP视频会议的混合式组播技术研究.电子技术应用, 2006年第2期

严国清,熊建设,石雷,宋柱芹.在不支持Remap的系统中构建JFFS2.微型机与应用,2005年11期

杨小辉,熊建设,徐洪梅,宋柱芹.Turbo码及交织技术在WCDMA的信道编码方案中的应用.微计算机信息, 2006年第12期

致谢

在此论文的收笔之际，谨代表我个人向所有支持我，帮助我，关爱我的人们献上我诚挚的谢意：

首先衷心感谢我的导师熊建设老师。整个论文在熊老师的悉心指导下完成，在这近三年的时间里，熊老师在学习、生活和科研中给予我诸多的关怀和指导，使我在新的研究领域有所进步；熊老师严谨的科研态度、兢兢业业的工作作风和锐意创新的进取精神为我树立了榜样，是我终生学习的楷模；他渊博的理论知识和丰富的实践经验使我受益匪浅，对此表示由衷的感谢！

特别地，我要深深感谢张海燕老师。张老师的和蔼可亲及对我接纳、帮助和宽容，我将铭记在心！

我要感谢王爱群老师及信息学院徐洪梅老师、程军娜老师、赵犁丰老师、王怀阳老师等等各位老师给我的帮助！

我还要感谢 321 实验室所有成员给我的关爱及建议；感谢舍友展旭卿、杨小辉和窦亚晶对我的关心和鼓励。大家一起愉快相处的日子，让我终生难忘。

在此，对所有关心和帮助过我的各位老师和同学致以崇高敬意和衷心感谢！

特别地，我要深深感谢我的父母，谢谢他们这么多年来默默无闻的支持、关心和鼓励，谢谢他们给了我人生路上前进的动力！

再次感谢给予我帮助和支持的所有人们!祝他们身体健康、工作顺利、心想事成！

最后，向百忙之中抽出宝贵时间来评阅这篇论文的各位专家教授致以衷心的感谢!

2006 年 4 月

宋柱芹