摘　要

随着多媒体技术、计算机技术和网络技术的发展，使得数字化多媒体数据在网络中实时传输成为可能。网络多媒体实时监控平台就是在这种情况下出现的。它综合了音视频数据压缩技术、实时传输技术和远程控制技术，在此基础之上可扩展为远程教学，视频会议等各种具体应用，应用前景非常广泛。

本文首先介绍了课题的相关技术，包括.NET 框架技术、传输媒介和传输协议的选择、.NET Remoting 技术及多媒体数据处理技术等。接下来讨论平台实现方案和实现过程，论述了平台结构和平台实现细节。最后是在平台基础之上实现的一个远程监控系统实例，简述了这个系统的结构及功能。

通过对多媒体网络通信和远程监控技术的研究，提出一种基于微软.NET 框架技术的网络多媒体实时监控平台分层结构。最底层是数据传输层，担负着多种类型数据的传输任务，根据数据传输的可靠性要求采取不同的传输策略；第二层是功能实现层，负责具体实现平台的各种功能；最上层是功能接口层，作为提供给各种应用的访问接口，又作为功能实现层的具体实现的标准。对平台模型的每一层，阐述了关键的实现技术，然后给出了主要实现方法。

为了保证多媒体数据传输的实时性，采用国际上流行的多媒体会话控制协议 SIP 来控制会话的建立、控制及终止，SIP 被设计作为 IETF 的多媒体数据和控制体系的一部分，比传统的 H.323 体系更适合 Internet，它与底层采用何种协议无关。音视频的传输过程由多媒体实时传输协议 RTP/RTCP 负责，多媒体数据由 RTP 传输，RTCP 用于保证一定的 QoS。在对多媒体数据进行处理时，考虑到多媒体通信的特点和实时监控的要求，应用了国际上通用的压缩标准（如 G.729a、H.263）对音视频数据进行压缩；对于可靠性要求高的数据传输，比如文本、文件传输，引入.NET 中的 Remoting 面向对象中间件技术，它提供了一套本地对象与远程对象间透明通信的解决方案，以及对对象激活方式、对象生命周期等要素的管理，提高了数据传输的性能，并极大地简化

了实现过程。

详细介绍了.NET 框架的原理及特点，并分析了它的优势所在。平台应用.NET 中的 DirectShow 技术进行音频和视频的捕获及回放；利用 PInvoke 技术调用非受控的 WIN32 DLL 代码；.NET 组件通过 COM Interop 技术与 COM 组件进行通信；在.NET 的通用语言规范基础上实现多种编程语言集成。在实现过程中讨论了.NET Remoting 和 RTP/RTCP 的原理以及在数据传输中的应用，重点叙述了 SIP 协议栈的设计及实现。

在平台基础之上，实现了一个多层结构的网络多媒体实时监控系统实例，主要功能包括文件传输、屏幕监视、屏幕广播、音频广播、远程硬件控制和远程软件启动。

关键词： 多媒体网络通信，远程监控，.NET 框架，.NET Remoting

ABSTRACT

With the development of multimedia technologies, computer technologies and network technologies, it has been possible to transport digital audio and video data on network in real time. The multimedia real-time monitor-control platform of network integrates audio and video compress technology, real time transport technology and remote control technology. It can be used widely and can extend itself to many applications such as remote teaching, video conference, etc.

First, this paper introduces related technologies including .NET Framework technology, choices of transport media and transport protocol, .NET Remoting technology and multimedia data processing technology, etc. Second, the methods and process of realization are discussed. At last, we show a remote control system instance based on the platform and give a brief introduction of its structure and function.

Through studying the multimedia communication on network and remote control technologies, this paper proposes a hierarchical model of the multimedia real-time monitor-control platform of network based on Microsoft .NET Framework. The lowest level is the data transport layer which is responsible for transporting different types of data. Function realization layer, the second level, realizes all kinds of functions of the platform. The top function interface layer provides the interface for kinds of applications; meantime, it is the criterion of the second layer. For every level of the platform model, it describes key realizing technologies and then the main realizing methods are given.

To assure the real-time property of multimedia data transport, the international prevailing multimedia session

control protocol SIP is adopted and used to initial, control, and terminate a session. Multimedia real-time transport protocols RTP/RTCP are responsible for audio/video transport. When processing the multimedia data, considering the property of multimedia data transport and the requirement of real-time monitor and control, the audio and video data are compressed with the international prevailing standards such as G.729a、H.263. For the data transport that needs high reliability, it imports the middleware .NET Remoting into the project, which improves the transport performance and simplifies the realization process.

The theory and characteristic of the .NET Framework in detail are introduced and its advantage is analyzed. Capturing and playing audio/video data are realized by the DirectShow in .NET. PInvoke is used to invoke unmanaged WIN32 DLL. .NET assembly communicates with COM components using COM Interop technologies in .NET platform. Many programming languages are integrated by .NET Common Language Specification. The emphases are put on the theories and applications in data transfer of .NET Remoting and RTP/RTCP. The design and realization of SIP stack are stressed.

This paper provides a multilayer real-time monitoring system instance on the basis of the platform. The functions of the platform includes transferring files, monitoring screen, broadcasting screen, broadcasting audio , remote control of hardware and software.

Keywords: Multimedia network communication; Remote monitor-control of network; .NET Framework; .NET Remoting

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名：侯海文　　日　期：2004.3.12

关于学位论文使用授权的声明

本人完全了解山东大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权山东大学可以将本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、缩印或其他复制手段保存论文和汇编本学位论文。

(保密论文在解密后应遵守此规定)

论文作者签名：侯海文　导师签名：　　日　期：2004.3.12

1 引言

1.1 课题背景

随着网络技术的发展，互联网的性能得到不断提高，越来越多的企业、团体和个人加入了 Internet。基于互联网的各种应用的飞速发展，文本和图片等静态媒体已经不能满足人们的需求，人们希望互联网能提供更多的服务，譬如 IP 电话、视频点播（VOD）以及视频会议。这些让互联网更“生活化”的应用离不开网络多媒体技术。

所谓网络多媒体技术就是将各种不同的信息如文本、图形、音频、视频等进行综合的表示、存储和传输。多媒体通信具有以下技术特点[1]：

(1) 传输的实时性：音频、视频数据要求实时传输，音频数据传输延迟必须≤200ms，视频数据传输速率应该达到 30 帧/秒，否则将会出现失真或不连续。

(2) 允许一定范围的数据丢失：多媒体数据量大，无法保证 100%传输正确，可以允许一定范围的数据丢失，如音频数据丢失量只要小于 2%，人的听觉基本感觉不出来。

(3) 提供广播服务：视频会议、计算机协同工作（CSCW）经常需要广播服务，多媒体通信可以将多媒体信息有选择地发送到某一特定群体，即组播。

(4) 弹性带宽：各种媒体传输数据量不同，所需带宽不同，使用适当的协议，如 RTP、RTCP 等可以控制各种媒体传输所需带宽，根据网络状况实现带宽弹性分配。

大量涌现的网络多媒体应用在丰富了 IP 网络应用的同时，也对传统 IP 网络路由、传输协议等提出新的挑战和要求。这些网络多媒体应用的共同特点是要求网络能满足传输多种多样的不同的媒体数据的不同传输要求，如实时性、传输带宽、突发性、时延抖动、出错率、媒体同步、协作控制以及分布传输等方面的要求。这些要求引发了对新的网络协议和网络传输机制的研究。为适应不同多媒体应用的不同需求，出现了很多新的传输手段，如 ATM、卫星通信和无线网络等，同时，一些新的技术被采用，这些技术跟传统“尽力而为”的 IP 网络相比，提供更完整的服务质量（QoS[2]），这些技术中包

含一些全新协议，特别是以 IPv6、RSVP 等为核心的新一代协议族，这些新一代协议针对实时传输、资源预留[3]、QoS 控制以及组播等方面面临的问题提出较完善的解决方案，是 IP 网络传输技术的发展方向。

考虑到已有投资，一步到位采用新协议（如 IPv6[4]）的传输手段是不现实的，传统的以 IPv4 为核心的协议族和众多的传输手段将存留很长时间，这就决定了基于传统 IP 网络进行有效的多媒体数据的传输仍将是研究重点。这些研究主要集中在如何有效地在传统 IP 网络上传输实时数据（差错控制、流控制以及拥塞控制）、组播路由算法、可靠和不可靠组播传输协议研究、压缩编码及纠错编码等。

在多媒体传输的应用方面，近年来一些商业产品相继推出，比如视频会议产品 NetMeeting[5]、CU-SeeMe、PictureTel[6]、Proshare、等，流媒体服务产品 NetShow[7]、RealPlayer 等，以及包括共享白板、远程教学以及远程医疗等在内的其它类型的众多多媒体应用产品。作为组播多媒体应用研究为主要目的的 MBone（Multicast Backbone）是 Internet 工程任务组 IETF 在 Internet 上建立的支持组播的长期试验性虚拟网络[8]，他由许多支持组播的“组播岛”构成，每一个岛都是由一些组播路由器连接起来的一些本地子网，岛与岛之间由连接组播路由器的隧道连接，岛中的组播数据包由组播路由器封装到普通 IP 包中，通过隧道传输到其它岛中，再由这些岛中的组播路由器将原始组播数据包释放到岛中，这样就在由许多不支持广播的主干路由器构成的 Internet 上实现了虚拟组播网络。MBone 上经常进行多媒体实时传输应用，开发了一系列多媒体工具如 SD（Session Directory）、NV（Network Video）、Vic（Videoconference）、VAT（Visual Audio Tool）、IVS（INRIA Videoconference System）、Wb（Whiteboad）等。在 MBone 开展的研究大大促进了多媒体传输研究，现在大部分标准化的多媒体和组播协议来源于 IETF 推动或主持的研究。

网络多媒体应用中有许多功能可以通用化或抽象化，为具体应用提供支撑平台，在此基础之上进行二次开发，降低了应用的开发难度，从而大大提高开发效率，增强系统的健壮性、灵活性和可维护性。基于.NET 的网络多媒体实时监控平台结合了网络多媒体技术和分布式对象技术，底层采用微软的.NET 框架，基于组件思想设计。平台为具体应用提供服务，具体应用调

用平台中的通用功能。该平台封装了多媒体通信所需要的协议栈，提供多媒体数据源采集功能，解决了部分 QoS 问题，并向应用层提供监视功能和控制功能接口。监视功能包括程监视远程工作站使用情况，通过摄像头远程监视外部环境。控制功能包括控制远程工作站各种软件、硬件，与远程工作站之间进行视频、音频和文字交流。

1.2 课题研究的主要内容

本课题是山东省科技厅科技发展计划项目“多媒体网络实时监控集成平台研究”的一部分。多媒体网络实时监控集成平台是一个通用的网络监视及控制平台，它能进行文字、图像、音频和视频等多种媒体的实时通信。在此平台之上可以实现各种具体的网络多媒体系统，如工业现场的实时监控、多媒体网络教室和多媒体会议等。

课题着重基于新的网络多媒体技术和软件技术来设计平台模型结构，并在此结构基础上实现平台的主要功能，最后应用此平台实现一个多层结构监控系统的实例。课题研究了多媒体通信的特点，分析了基于.NET 平台开发系统的优势，并结合组件技术，提出网络多媒体监控平台的分层设计思想和实现方法。针对每一层的相关技术和具体实现方法，进行详细论述。在实现网络环境下多媒体数据的处理以及远程交互与控制功能时，为了保证系统的实时性、通用性和兼容性，采用了目前国际上流行的协议（如 RTP、SIP）和算法（如 G.729a、H.263）。平台通过功能接口层为具体的应用提供编程接口，用户可以扩展自己的功能来实现特定的系统。

1.3 课题特点

课题研究的特色和创新之处主要有：

(1) 基于.NET 的分层结构。课题深入分析了网络多媒体应用的特点和.NET 框架的体系结构，提出了基于.NET 的多媒体网络监控平台的分层体系结构模型，为应用层提供可扩展的编程接口。

(2) 基于 RTP 的音视频传输。课题实现了基于组播的实时传输协议(RTP)，任何一个工作站的数据都能同时传输到多播组的所有成员，解决了利用点到点的通信模拟多点广播带来的问题，如浪费带宽资源、效率低等。

(3) 基于 SIP 的会话控制。课题实现了 SIP 协议栈及 UA，SIP 协议是下一代网络中的核心协议，它具有实现简单、与 Internet 结合紧密、功能强大的特点。课题中利用它来实现会话控制功能。

(4) 基于中间件来简化数据传输过程。课题利用.NET 中的面向对象通信中间件 Remoting 来简化网络通信的复杂度，.NET Remoting 中可以设置多种通信协议，多种对象模型来构建由客户端与服务器组成的系统框架。

(5) 基于多种编程语言结合的组件实现。.NET 框架提供了多种语言之间相互兼容的基础设施，使在同一个系统中使用多种语言编写代码成为可能，而且可以将历史组件方便地集成进现有系统。课题中主要采用 C#等新型语言编写功能，并将前期工作中 C++编写的模块融入系统[9]。

综上所述，多媒体网络实时监控平台是一个通用的、综合的、基于 TCP/IP 协议的和纯软件的监控平台。本课题在平台上实现了文本、图像、音频及视频等多种媒体处理方法的有机集成；平台中的多媒体处理功能封装成模块，并提供了丰富的编程接口，用户可以方便地扩展所需的功能，具有良好的可扩展性。对多媒体数据的处理还采用了国际标准化组织制定的相关标准（如音视频数据压缩采用 ITU-T 的 G.729a[10]、H.263[11]等，传输协议采用 RTP/RTCP，控制协议采用 SIP），使平台具有良好的兼容性，并保证了多媒体数据传输的实时性和准确性。

2 相关技术

2.1 .NET 框架（.NET Framework）介绍

Microsoft .NET 框架的目的是使开发者更容易建立网络应用程序和网络服务。图 2-1 显示了.NET 框架的体系结构。建立在操作系统上的第一层服务，是管理运行时代码需求的公共语言运行库(Common Language Runtime，CLR)，运行时代码理论上可以用任何现代编程语言编写[12]。公共语言运行库提供了许多服务，这些服务有助于简化代码和应用程序的开发，同时也可以提高应用程序的可靠性。.NET 框架还包括一套可被开发者用于任何编程语言的基类库。在此之上是许多应用程序模板，这些模板为开发网络应用和网络服务提供高级组件和服务。

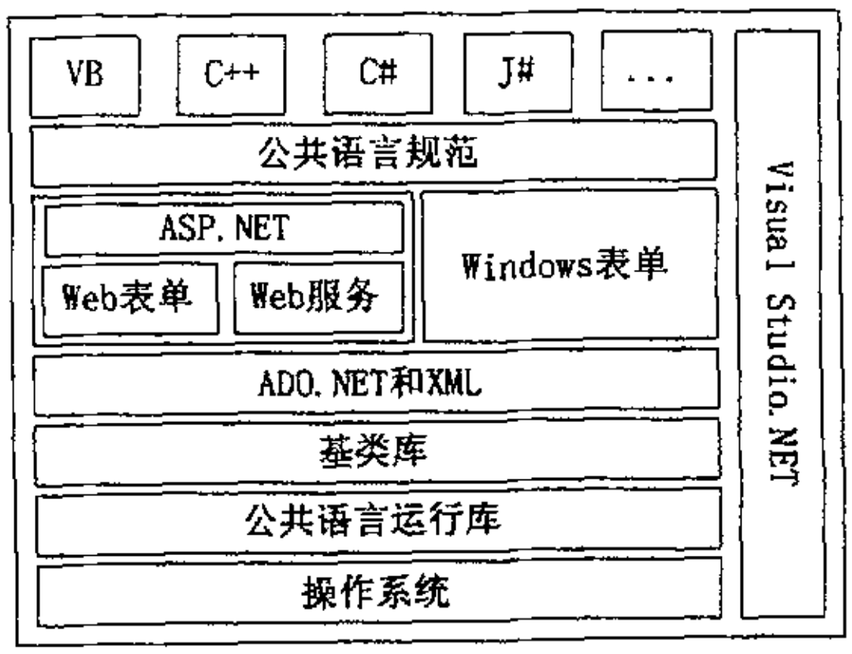

图2-1 .NET Framework体系结构

2.1.1 公共语言运行库（Common Language Runtime）

.NET 提出了一种全新的开发软件的方法。而且. NET 并没有局限在 Windows 操作系统上。. NET 框架允许开发者利用公共语言运行库的特性，并提供许多高层次的服务，以免开发者重复编写相同服务的代码。更为重要的是位于运行库下面的.NET 公共语言运行库引擎提供了快速开发软件的技术。公共语言运行库的主要组成部分如图 2-2。

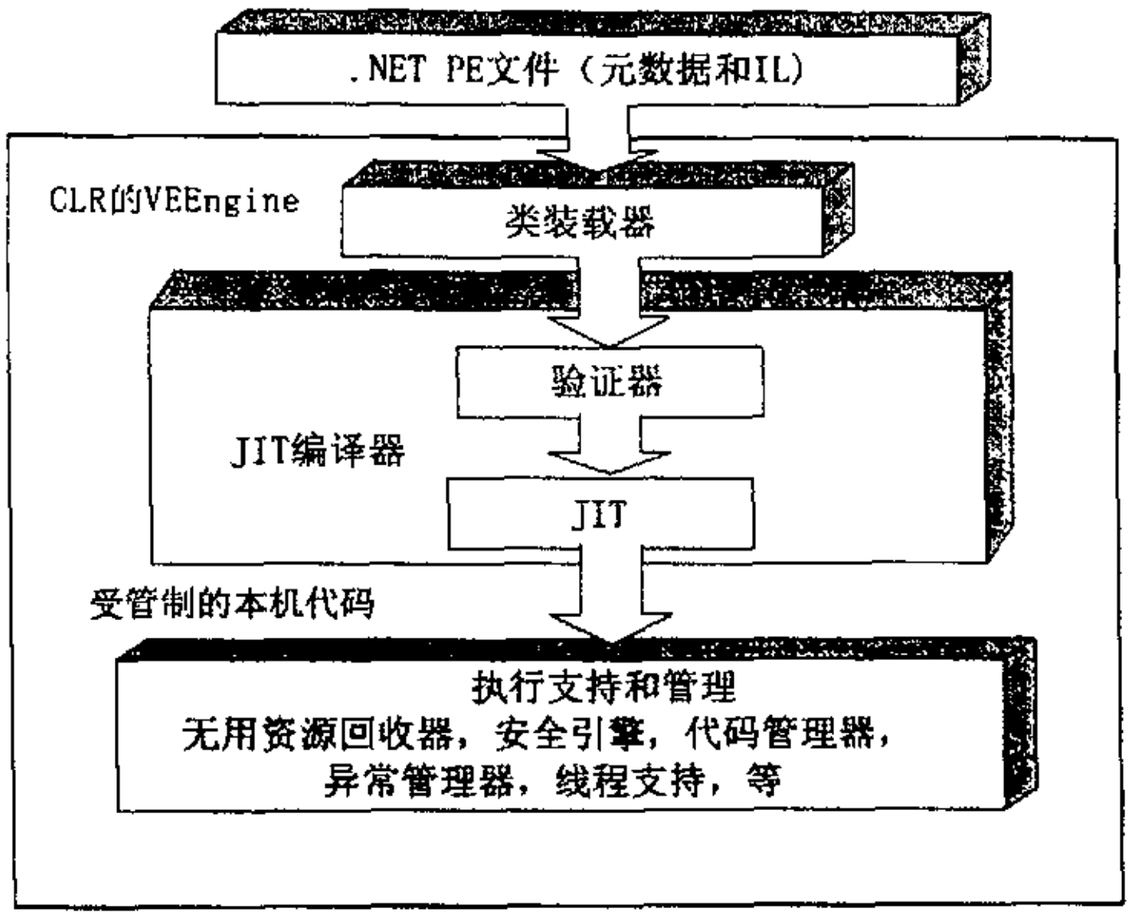

图2-2　主要CLR的组成部分

.NET 公共语言运行库提供了如下特性：

(1)一致的编程模型：目前 Wndows 操作系统的某些功能是可以通过 DLL 函数提供，也可以通过访问 COM 对象来获得。在. NET 中，所有的应用程序服务都是通过一个面向对象的通用编程模型提供的，这就解决了目前 Windows 系统中编程模型不一致的问题。

(2)简化的编程模型：Windows 编程中最复杂的是，程序员必须了解 Win32 和 COM 中一些及其复杂和神秘的结构，还需要了解注册表、GUID、IUnknown、AddRef、Release 和 HRESULT 等概念[13]。在新的.NET 平台中，这些概念根本不存在。

(3)一次运行，永远运行：Wndows 程序员特别是 COM 程序员对 DLL Hell 都很熟悉。DLL Hell 是指：在 Windows 中，安装一个应用程序时，其组件可能会覆盖原有程序的组件，导致原有应用程序运行出错甚至不能运行。.NET 体系结构把不同的应用程序组件分离开来，使应用程序总是只加载其所需的组件，如果应用程序安装后可以正确执行，那么这个应用程序就可以一直正常运行，而不受到以后安装的应用程序的影响，这就很好地解决了 DLL Hell 的问题。

(4)支持多平台：一旦编写并创建了一个受控（managed）.NET 应用程序，就可以运行在任何支持.NET 公共语言运行库的平台上。Microsoft 将在以后逐渐发布非 Wndows 版本的公共语言运行库，可以让开发者在诸如

Linux 等系统上开发和部署.NET 应用程序。

(5)代码重用：开发者可以在.NET 平台中创建给第三方应用程序提供服务的类，代码重用十分简单。

(6)自动资源管理：编写程序应当注意的问题就是如何管理好程序使用的资源，如对文件、内存、显示空间、网络连接和数据库连接等资源的管理不善，就会产生错误，导致该程序或其它程序非正常运行。.NET 公共语言运行库自动跟踪资源使用情况，如果应用程序不再需要某种资源，就自动回收它，保证应用程序不会发生资源泄漏，这种机制在.NET 中称为“垃圾收集”(Garbage Collection)。

(7)类型安全：.NET 公共语言运行库可以检验所编写的代码是否类型安全。类型安全保证了分配的对象总是以兼容的方式访问。比如，一个函数的输入参数定义为 4 字节的值，如果试图以 8 字节的方式访问该参数，公共语言运行库将检测并捕获这种情况。类型安全同时也意味着没有任何方法可以构造一个内存位置的任意引用，也不能让代码在任意引用位置开始执行。这种机制消灭了经常出现的一些程序错误，以及诸如溢出的系统攻击。

(8)丰富的调试支持：因为.NET 公共语言运行库可以支持多语言，所以程序的某些部分可以使用最合适的语言来实现。.NET 公共语言运行库完全支持跨语言边界调试应用程序。公共语言运行库也提供了内置栈访问功能，可以很容易地定位程序故障和错误。

(9)一致的错误处理：以前 Wndows 编程的错误处理方法很不一致，有些函数返 Win32 错误代码，有些函数返回 HRESULT，有些函数引发异常。.NET 中，所有错误都是通过异常(Exception)给出的。异常允许开发者把完成程序逻辑的代码和错误处理分隔开来，这极大地简化了代码编写，方便了阅读和维护代码。同时，异常也是跨模块和语言边界的。

(10)简单的部署：目前，基于 Windows 的应用程序安装和部署起来很困难，通常需要创建几个文件，在注册表中进行设置。而且，要完全卸载一个 Wndows 应用程序很难。在 Windows 2000 中， Microsoft 引入了一个新的安装引擎来帮助解决这些问题，但是发布 Microsoft 安装包的软件公司仍然可能没有按照规定做。.NET 组件不需要在注册表中注册，大多数的基于. NET 的应用程序的安装只需要拷贝文件到程序工作目录下就可以了，而卸载应用

程序只需要简单地删除该目录下的文件。

(11)安全性：传统的操作系统的安全控制提供了独立的用户账号并根据账号进行访问控制，这已经证明了是一个有效的模式，但是其核心是假定所有的代码都是值得信赖的，当所有代码通过物理介质（如 CD－ROM）和公司内部服务器来安装时，这种模式是可行的。但是，随着应用程序对移动代码如 Web scripts、Internet 应用程序下载、电子邮件附件的依赖程度的提高，就需要对程序行为进行更细粒度的控制，为此.NET 引入了 Code Access Security（代码访问安全）的概念。

同时 Microsoft 在.NET 中也加强了基于角色的控制。.NET 支持在应用程序中定义角色，然后基于这些角色对应用程序实施存取控制，这种机制可以扩展到 Internet 和异构环境中。

公共语言运行库可以加载并运行用任何基于.NET 的编程语言编写的代码。基于.NET 的代码称为受控代码(managed code)，受控代码意味着在内部可执行代码与公共语言运行库之间存在已定义好的合作规范。而生成对象、调用方法等任务，都委托给了公共语言运行库，这使得公共语言运行库能为可执行代码增加额外的服务。

.NET 提供了如此多的优良特性，为此 Microsoft 引入了一些新的概念。.NET 的跨语言集成的特性是因为所有基于.NET 的语言都遵循 Common Language Specication（通用语言规范）；.NET 的组件是自描述组件，归因于 Microsoft 在组件中引入了 Metadata（元数据）这一数据结构；NET 所具有的简单配置和版本化特性是因为 Microsoft 引入了 Assembly（组件）的概念；.NET 的集成安全服务是因为.NET 采用了多种安全策略。

2.1.2 通用语言规范（Common Language Specification）

Microsoft 当前的组件技术 COM（COM+[14]）允许用不同语言创建的对象互相通信，而.NET 公共语言运行库向前又迈进了一步，它几乎整合了所有的现代编程语言，同等地对待不同语言创建的对象。 公共语言运行库可以做到这一点是因为它定义了标准类型集合。

编程语言相互之间的差异很大。例如，某些语言的符号是大小写敏感的而另外一些则是大小写不敏感；某些语言支持无符号整型而另外一些则不支

持；某些语言提供操作符重载而另外一些则不提供；某些语言有联合类型而另外一些没有；某些语言支持可变数目参数的函数而另外一些则不支持。.NET 整合这些相互之间差异很大的现代编程语言在于通用语言规范。要创建可以被其它编程语言使用的.NET 类型，就应当保证该语言使用的特性其它语言也有。为此， Microsoft 定义了通用语言规范，规定了基于.NET 的编译器必须遵循的最小集合的特性。

公共语言运行库使用一种新的能表达大部分现代编程语言语义的 CTS（Common Type System，通用类型系统），通用类型系统定义了一套标准类型及生成新类型的规则。公共语言运行库知道怎样生成和执行这些类型。编译器和解释器使用公共语言运行库服务定义类型、管理对象及进行方法调用，而不是使用特定于某个语言的方法。

类型系统的一个主要设计目标是多种语言能够高度集成。用一种语言编写的类能继承用另一种语言编写的类，用一种语言编写的代码产生的异常能被用另一种语言编写的代码捕获，而且调试和 prifiling 之类的操作可以无缝地工作，不用考虑编写代码所用的语言。这就意味着编写可重用类库的开发者，不再需要为每一种编程语言或编译器生成一个版本，而且类库的使用者可以使用其它编程语言开发的类库。

2.1.3 元数据（Metadata）

如果要让客户端使用一个 COM 组件，COM 组件必须提供 IDL 文件或者类型库，这样客户端才能调用组建接口。在.NET 中，IDL 文件和类型库成不再需要，这是因为在.NET 组件中包含了类型信息，.NET 组件是自描述的。

自描述组件简化了开发和配置过程，并提高了系统的可靠性。许多由公共语言运行库提供的服务是由元数据驱动的，它是可执行代码的补充信息。因为所有的信息（可执行代码和类型信息）都存储在一起，所以这样的组件称为自描述组件。

自描述组件的一个主要优点是，使用它们不需要其他文件的支持。类的定义不需要单独的头文件，可以通过检查元数据得到。跨语言或过程边界访问组件不需要各自的 IDL 文件、类型文件或 Proxy/Stubs，所必需的信息已经存在于元数据之中。为了识别开发者请求的服务属性，并不需要部署各自

的配置信息。最主要的是，由于元数据是在编译过程中由源代码生成，并与可执行代码存储在一起，它将永远与可执行代码同步。

.NET 的编译器要做的主要工作就是处理源代码并产生相应的 MSIL 代码，并且负责把元数据嵌入到.NET 兼容的 EXE 和 DLL 中。简单地说，.NET 元数据就是以二进制形式保存在PE文件[15]中的包含文件或.NET组件的类型定义和方法实现的信息集合。元数据来源于 COM 技术中的类型库和 IDL 文件等技术。元数据比它的前身更完善，并且总是和代码放在同一个文件中，它和 EXE 和 DLL 文件是不能分开的。

所有面向.NET 的编译器都需要在已经编译的代码模块中产生每个类型的完整的元数据信息。元数据包含每个类型的定义，以及类型的所有成员（方法、域、属性和事件）的名称和类型。对于每个方法，元数据包含加载程序如何定位方法体的信息。

最后，.NET 框架中进行远程方法调用时也需要元数据。为了进行远程方法调用，公共语言运行库必须使用进程调用序列化（Process Called Serialization）分配一块内存，在内存中放入方法的参数数据。公共语言运行库使用元数据中的信息来决定所要分配的内存块的大小以及如何列集(Marshal)参数数据到内存块中。然后数据块就通过流发送到远程机器上，反向序列化，这时需要元数据再次提供内存块的模板，然后远程机器调用方法，返回值送回到发起远程调用的机器上。

2.1.4 .NET 组件（Assembly）

在.NET 出现之前，基于 Windows 的应用程序的打包和部署都十分复杂。应用程序需要多种类型的文件（如 EXE、DLL 和数据文件）才能运行，因为共享文件之间的不兼容，安装新的应用程序导致程序崩溃的显现时有发生。许多应用程序需要多个注册表设置才能使系统定位到这些文件。应用程序的部署和管理让不能令人满意。

于是在.NET 中引入 Assembly，它是为了完成一项工作所需操作的集合，包含在一个或多个文件中。通常来说部署一个应用程序只需要拷贝一个或多个 Assembly 到指定的应用程序目录下。

除了改善对单个组件的配置，.NET 框架定义了一个应用程序配置模板，

以解决定制应用程序安装和 DLL 版本控制（“DLL Hell”）这些复杂问题，Runtime 提供了支持这个模板的服务。

一个 Assembly 是一组资源和类型的集合，Assembly 包括了描述这些资源和类型的元数据，它是一个配置的基本单元。元数据包括 Assembly 的名单(Manifest)，此外它还包含像类型和资源表等能被 Assembly 外部可见的信息，Manifest 也包含从属关系等信息，例如 Assembly 建立时的版本号。开发人员可以制定版本策略，以指示公共语言运行库是否装入系统上已安装 Assembly 的最新版本，或者加载一个指定版本，或者使用编译时使用的版本。

在同样的操作系统上可以存在某组件的多个拷贝，但是只有一个拷贝能被操作系统注册、调入内存和执行。对系统来说，定位和调入内存的策略是全局的。.NET 公共语言运行库增加了所必需的体系架构以支持管理组件定位和调入的每个应用程序的策略，这在.NET 中称为并行部署(Side by Side Deployment)。

Assembly 可以是一个应用程序私有的，或被多个应用程序共享。一个 Assembly 的多个版本可以同时配置在同一台机器上。应用程序配置信息定义了 Assembly 的位置星系，这样公共语言运行库就能为同时运行的两个不同的应用程序装入同一 Assembly 的不同版本。这就消除了由组件版本的不兼容性引起的问题，提高了系统的整体稳定性。管理员可以在部署时为 Assembly 增加配置信息，让它使用不同的版本，但是编译时提供的原始信息永远不会丢失。

因为 Assembly 是自描述的，所以并不需要在系统上注册。应用程序的部署简单到只需将文件拷贝到目标目录中即可（如果应用包含非受控（Unmanaged）组件，情况会复杂些）。配置信息保存在可以被任何文本编辑器编辑的 XML[16] 文件中。

.NET 不能保证简单名称是唯一的，对于私有 Assembly 来说简单名称已经够用了，但是对公开发布的 Assembly 来说，需要一个更强的标识符来保证它的唯一性。

公共语言运行库通过使用标准的公钥加密技术[17]来解决名称唯一性和代码完整性的问题，它使用公钥作为简单名称的前缀。为了保证只有公私钥

对的拥有者可以发布该共享名称，用私钥来为文件产生一个数字签名，以后可以用公钥值来验证该签名，签名后对可执行文件所做的任何修改都将导致签名认证失败。以这种方法进行数字签名的 Assembly 就拥有了适于发布的共享名称。

共享名称满足下列要求：

(1)依赖唯一的密钥对，能保证名称的唯一性。

(2)防止其它 Assembly 进入该 Assembly 的名字空间。因为开发者是自己私钥的唯一拥有者，没有人能产生和某个开发者一样的标识。这对发布多个 Assembly 版本尤其重要。

(3)提供了标识的安全概念。如果公共语言运行库通过了认证，就可以保证该 Assembly 来自可以信任的人。

共享名称是通过 Assembly 开发者的公钥私钥对产生的。私钥的保护很严密，而公钥保存在 Manifest 中。.NET 提供了一套 API 给编译器和工具软件来产生共享名称以及管理和名称相关联的密钥对。

Assembly 的 Manifest 建立以后，就包含了组成该 Assembly 的所有文件的引用。Manifest 中列出的每个文件都放在一个哈希表中，每个哈希值对应一个文件名。一旦建立了包含 Manifest 的文件，整个文件的内容都进行哈希运算，产生的哈希值采用发布者的私钥签名，该 RSA 数字签名存储在文件的预留部分（该部分不进行哈希运算），这时文件就可以发布了。

当文件被安装在用户的机器上时，系统通过公钥来验证共享名称的数字签名[18]。如果签名验证失败，就认为组件的内容被篡改过而不能信任。值得注意的是，系统在安装时只能检测到 Assemlby 中包含 Manifest 的那个文件的改变，另外那些文件的修改只有在运行时被访问时才能检测到。

程序运行时，当要解析 Assembly 的引用时，就定位包含 Manifest 的文件，这时被引用 Assembly 的公钥就和 Assembly 中记录的公钥相比较，如果不匹配，就产生 TypeLoadExcopion 例外。这也保证了需要引用的 Assembly 可以由私钥的拥有者认证。

最后，当加载 Assembly 的其它文件时，系统计算该文件的哈希值并将它与记录在文件中的值作比较(保存在被引用的 Assembly 的 Manifest 中)。如果这两个值不匹配，文件的内容就被认为是被篡改了而不能被信任。这种

方法的一个缺点是 Assembly 文件是在运行时计算哈希值，导致运行时的性能下降。

类型的每个引用都是由 Assembly 引用来规定范围的，这些信息保存在元数据中，每个文件都包含组件和它们的版本信息。一个 Assembly 引用包含简单名称、兼容版本号和被引用 Assembly 的场所/文化信息。如果被引用 Assembly 有一个共享名，该引用也包含被引用的 Assembly 的公钥。

因为类型的标识包含 Assembly 的标识，所以可以同时加载每个类型的多个版本。实际上，公共语言运行库发布了一系列的技术来提供应用程序的隔离功能。这意味着应用程序所使用的类型的某个版本并不影响另一程序使用的类型的版本。如果程序员在访问资源时很小心，组件可以在. NET 中并行存在。

程序的部署很简单，应用程序目录包含了运行时需要的所有文件。用户可依靠已经安装在机器上的 Assembly，或者提供一个配置文件，告诉公共语言运行库如何进行定位——本地、内部网或者 Internet 上。

2.1.5 验证和安全（Verification and Security）

公共语言运行库也提供完整的、深入的安全服务，以确保未经授权的用户不能访问机器上的资源，并且代码不会执行未经允许的操作。这就提高了系统整体的安全性和可靠性。由于系统使用公共语言运行库装入代码、主成对象、执行方法调用，所以当受控代码装入内存执行时，公共语言运行库能进行安全检查，执行安全策略。

通过代码访问安全机制，开发人员能为应用程序指定完成工作所需的权限。例如，指定代码具有写文件或访问环境变量的权限。这类信息和有关代码标志的信息一起存储在 Assembly 级别上。当代码装入内存及执行方法调用时，公共语言运行库验证是否能给予代码所要求的权限。如果不能，将记录一条安全冲突信息。给予权限的策略，称为信任策略，是由系统管理员建立的，并且是建立在关于代码的信息基础之上的，这些信息包括：代码是谁发布的，是从什么地方获得的，以及 Assembly 的标识和它所请求的权限。开发人员可以指定他们显然不需要的权限，以防止其它人恶意使用他们的代码。如果直到运行时才会知道代码所需的权限，那么安全检查可以通过编程

实现。

除了代码访问安全，公共语言运行库还支持基于角色的安全。基于角色的安全和代码访问安全建立在相同的权限模板上，只是这些权限是建立在用户的身份之上，而不是建立在代码标志之上。角色表明了用户所属的类，可以在开发或配置阶段定义。程序运行时，确认用户的身份，用户将使用这个身份运行代码。公共语言运行库判断用户属于哪个角色的成员，然后基于该角色给予相应的权限。安全问题是.NET 设计的一部分，它在开始加载类的时候就以验证的形式存在了，然后通过代码访问安全机制来控制代码对资源的访问。它提供了确定角色和标识信息的机制，通过上下文、进程和机器边界可以获得这些信息，确保数据在远程访问时不被破坏。安全机制也深入到公共语言运行库体系结构中确保应用程序分隔边界，这些机制被现代操作系统普遍使用。

综上所述，.NET Framework 是一项影响深远、前景广阔的技术，通过项目实践，总结.NET Framework 的特点及优点如下：

(1).NET Framework 从多种角度简化了软件开发过程，它支持自动垃圾回收，简单统一的组件模型，对诸如字符串和日期这样的普通类的支持，并且与 WSDL、XML、SOAP[19]和 HTTP 这样的标准技术无缝集成。

(2)应用程序更加健壮。由共享 DLL 引起的 DLL Hell 旧问题几乎不存在了。在安装新程序或卸载旧程序时再也不会中断其它的.NET 应用程序了。

(3)为了兼容传统项目，.NET Framework 还支持 COM、DCOM 和 ActiveX 组件。

(4)COM 组件将注册表中各种各样的信息都保存下来，导致庞大的注册表文件，与之相比，.NET Framework 不再大量地有求于 Windows 注册表或 win\system32 系统目录了，大多数.NET 组件对注册表没有影响。.NET Framework 将 Windows 注册表又还原成它的最初角色：作为存储像用户设置和优先权这样的应用程序数据的小型存储库的中心。机器上的 win\system32 系统目录也不会作为包含一切 DLL 的文件夹了，只作为那些被操作系统及其扩展所使用的 DLL 的存储库。

(5)跨语言的兼容性。现在可以利用不同的语言开发组件，根据开发团队的技术混合匹配不同的语言。由于可以将一种开发出来的组件和.NET 中

的其它任何语言一起使用，代码的复用率会大大提高。

(6)由于.NET 组件被封装成自描述的形式，不需要系统注册，安装程序也变得简单了，通常使用一个简单的 XCOPY 命令就可以安装完整的应用程序。

2.2 传输媒介

多媒体实时音频视频信息具有庞大数据量，视频图像和声音在时间上是连续的，要想做到音频视频的实时传输，可以通过增加通信信道的带宽来提高通信网络的传输速度。主要方法有：使用宽带的 LAN 或 WAN，或采用 DDN、ISDN、XDSL 等网络接入方式。

LAN 的带宽在 10Mbps 以上，对于音频视频传输基本够用；WAN 的带宽经常不能得到保证，例如在 Internet 上，主干网络的带宽充足，而在非主干网络上的带宽可能较窄。

普通的用 Modem 接入到 PSTN 的传输速率较慢，目前在 56Kbps 以下，而 56K Modem 的上行最高速率为 33.6Kbps，所以 Modem 间的互相传输最高只能连接在 33.6Kbps。在这种情况下，QCIF 图像格式传输的连续性还比较好，CIF 图像格式传输的连续性就下降了。在 PSTN 上采用 ISDN[20]、XDSL[21]等接入方式可以显著提高传输速率。

普通的 Modem 通过将数字脉冲进行频率调制，在 PSTN 上传输模拟信号，而 ISDN(Integrated Services Digital Network)提供端到端的数字传输，它在标准的用户网络接口中提供了从低到高的多种通路：（1）B 通路：速率 64Kbit/s，供传递用户信息使用；（2） D 通路：速率 16Kbit/S 或 64Kbit/s，供传输信令和分组数据使用；（3）H 通路：速率 384Kbit/s，供传递用户信息用；（4）H11 通路：速率 1536Kbit/s，供传递用户信息用；（5） H12 通路：速率 1920Kbit/S，供传递用户信息用。H 系列通路的费用太高，使用最普遍的是 B 通路，其接口类型称为基本接口，它是把现有电话网的普通用户线作为 ISDN 用户线而规定的接口，它由两个 B 通路和一个 D 通路 （2B+D）组成，其最高传输速率是 64*2+16=144Kbit/s（N-ISDN），在这样的传输速率下，图像分辨率和连续性都可以很好，是一种非常好的接入方案。

另一种是采用数字用户线的接入技术：所谓数字用户线（xDSL，Digital

Subscriber Line）技术是采用现有的普通双绞电话线向用户传输高带宽数据（如多媒体和视频数据）的调制解调器技术。xDSL 一词覆盖了许多类似的、目前仍处于竞争的 DSL 形式，包括 ADSL、SDSL、HDSL、RADSL 和 VDSL。

高速数字用户线（HDSL）最高数字传输率达到 1.544Mbps～2.048Mbps；而非对称数字用户线（ADSL）最高数字传输率可达到 6Mbps～7Mbps，足以传输符合广播级质量的实时活动图像的要求。美国采用离散多音频（DMT，Discrete Muti-Tone）调制技术的 DMT ADSL，主要有三种类型：（1） ADSL-1 提供 MPEG－1 编码的视频信息，码率为 1.5Mbps，最大距离 6km；（2） ADSL-2 码率为 3 Mbps，最大距离 4km；（3）ADSL-3 码率为 6 Mbps，最大距离 2.5km。本方案的最大缺点是传输距离有限，虽然可以通过增加中继器的办法来扩大距离，但显然这又增加了成本和技术复杂性。

2.3 传输协议的选择

现有局域网在传输实时音、视频图像时遇到的主要问题是：实时音、视频在时间上是连续的，因此要求不间断地进行实时传输。而现有局域网是基于各站点共享网络带宽思想设计的，它假设各站点间传输的数据在时间上是相互独立的，从而可以把数据打包，分别传送。并要求所有数据包都能够按照发送次序全部达到目的地，不能够有数据包的丢失。因此从这个观点来看，现有局域网技术是不符合多媒体通信要求的。因此必须研究新的高速局域网络协议来满足多媒体通信应用的要求。例如，基于最广泛的 TCP 协议的 HTTP 和 FTP 等各种静态文件的协议都不再适合远距离实时视频、音频的传输，而必须采用基于 UDP 协议上的实时传输协议(RTP)。

2.3.1 TCP/IP

广泛地讲，TCP/IP 软件分成五个概念层次[22]，图 2-3 给出了这些概念性的层次结构以及在这些层次之间传输的数据形式。

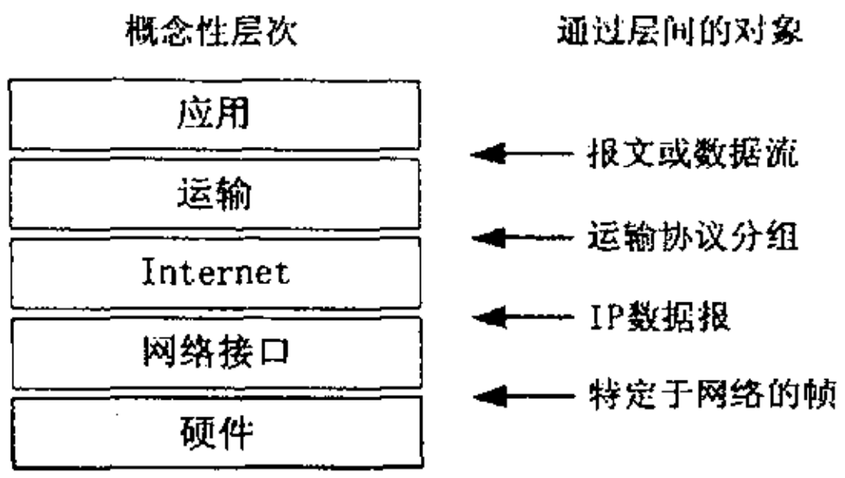

图2-3 TCP/IP的五层参考模型

应用层在最高层，用户调用应用程序通过 TCP/IP 互联网来访问可用的服务。与各个运输层协议交互的应用程序负责接收和发送数据。每个应用程序选择适当的传输服务类型（包括报文传输和字节流传数两种类型）。应用程序把数据按照运输层的格式要求组织好之后向下层传输。

运输层的基本任务是提供应用程序之间的通信服务。这种通信又叫端到端的通信。运输层要系统地管理信息的流动，还要提供可靠的传输服务，以确保数据到达无差错、无乱序。为了达到这个目的，运输层协议软件要进行协商，让接收方会送确认信息及让发送方重发丢失的分组。运输层协议软件把要传输的数据流划分为小块（有时把这些小块称为分组），把每个分组连同目的地址交给下一层发送。

Internet 层用来处理机器之间的通信问题。它接收运输层请求，传输某个具有目的地址信息的分组。该层把分组封装到 IP 数据报中，填入数据报的首部，使用选路算法来确定是直接交付数据报，还是把它传递给路由器，然后把数据报交给适当的网络接口进行传输。该层还要处理传入的数据报，检验其有效性，使用选路算法来决定该对数据报进行本地处理还是应该转发。如果数据报的目的机处于本机所在的网络，该层软件就去除数据报首部，再选择适当的运输层协议来处理这个分组。最后，Internet 层还要根据需要发出和接收 ICMP[23]（Internet 控制报文协议）差错和控制报文。

网络接口层负责接收 IP 数据报并把数据报通过选定的网络发送出去。网络接口层包含一个设备驱动程序，也可能是一个复杂的子系统，使用自己的数据链路协议。

2.3.2 组播

组播是一种允许一个或多个发送者（组播源）发送单一的数据包到多个接收者（一次的、同时的）的网络技术[24]。组播源将数据包发送到特定组播组，而只有属于该组播组的地址才能接收到数据包。组播可以大大地节省网络带宽，因为无论有多少个目标地址，在整个网络的任何一条链路上只传送单一的数据包。它提高了数据传送效率，减少了主干网出现拥塞的可能性。组播组中的主机可以是在同一个物理网络，也可以来自不同的物理网络（如果有组播路由器的支持）。在 TCP/IP 协议中多播通信具有两个层面的重要特征：控制层面和数据层面。其中控制层面定义了组成员的组织方式；而数据层面决定了在不同的成员之间数据如何传送。这两方面的特征既可以是“有根的”，也可以是“无根的”。在一个“有根的”控制层面内，存在一个特殊的多播组成员，即根节点，而剩下的每个组成员都称为叶节点。大多数情况下，根节点负责多播组的建立，其间涉及到建立与任意数量的叶节点连接。而在某些特殊情况下，叶节点可以在以后的某个时间申请加入一个特定的多播组。对任何一个具体的组来说，都只能存在一个根节点。对“无根的”控制层面来说，它允许任何人加入一个组，其间不存在任何例外。在这种情况下，所有组成员均为叶节点，每个成员都有权加入一个多播组。IP 多播便是无根控制层面的一个典型的例子。数据层面也存在有根和无根两种形式。在有根的方式下，根节点发送的信息可以被所有的叶节点收到，但叶节点发送的信息只有根节点可以收到；在无根方式下，所有的组成员都能将数据发给组内的其他所有成员。从一个组成员发出的数据块会被投递给其他所有成员，同时所有接收者都能回送数据。至于谁能接收或发送数据，则不存在任何限制。IP 多播采用的是数据层面上的无根通信方式。

由以上的讨论可知，多播实现了一个发送者对一个或者多个接收者的数据传送，并且只有当有接收者加入其组时，多播数据才会传送给它。比广播方式传送更优越，多播方式的路由器（或交换机）只对于加入其组的主机的路由（或分支）播发多播数据，这样减少了网络的传输压力，同时满足了实时多媒体数据的传输要求。因此，多播方式特别适合于传输实时多媒体数据。

多播方式只适用于 UDP，进行一个对多个主机的 UDP 数据传送。IP 多播

通信使用D类IP地址，即多播地址(从224.0.0.0到239.255.255.255之间，其中有一部分为网络管理保留)。IP多播中使用这个组地址中的某一个地址对一个指定的组进行命名，所有参加这一多播组的进程都使用同一个多播地址。一个主机组可以跨越多个网络，多播组地址与主机所在的网络地址及主机IP地址无关。多播组成员是动态的，主机可以随意加入或退出某个多播组。IP多播组成员在多播工作中可以提供多播数据的发送和接收服务，或者其中之一。实际上某一多任务主机上运行的进程中，可能同时存在运行TCP面向连接服务的进程和运行UDP多播协议的进程，也可能同时有多个属于不同多播组的进程[25]。不同的进程和相应的通信服务之间是通过端口号来连接的，多播的实施同样也离不开网络硬件的支持。首先，使用的网卡必须支持多播通信；另外，当多播组跨越多个网络时，路由器也要支持多播通信方式。

TCP/IP网络中使用IGMP[26]（INTERNET组管理协议）来管理多播。IGMP是IP多播的基础，它负责管理多播客户机以及它们在组内的关系。多播主机使用IGMP通知路由器，路由器通过报文知道所在的子网的一台计算机想加入一个特定的多播组。IGMP报文是一种固定长度的IP数据报。组播路由器使用IGMP报文来跟踪与之连接的网络上组播组成员。要想使组播正常工作，两个组播节点之间的所有路由器都必须提供对IGMP的支持。若一个应用进程要加入组播组，执行一条IGMP “加入”命令便发送了IGMP报文，该报文用于通知所有的路由器：有一个客户机对一个特定的组播地址产生了兴趣。若一个路由器拥有一个或多个组播组，它便会向所有客户机定时发送一条“组查询”消息，查询当初通过一条加入命令通知它的每个组播地址。如果网络上的客户机仍在使用该组播地址，便发送一条IGMP消息作为响应，以便继续转发该地址对应的数据，否则路由器会停止转发该组的数据。在IGMP第二版中，允许客户机向路由器发送一条“离开”消息，明确告诉它停止转发该客户机所在路由的指定多播地址的数据。另外，IGMP报文的TTL(生存时间域)也可控制组播数据的转发路由器数。现在绝大多数路由器都提供了对组播的支持。有些交换式路由器不支持组播，但仍然可通过“隧道”技术来实现[27]。

2.3.3 RTP/RTCP

RTP 提供了端到端的传输实时数据的网络交付服务[28]。尽管 RTP 通常是在 UDP 之上使用的，但它是独立的网络和传输协议。

RTP 能够使用在单点传送和多点传送两种网络服务上。在单点传送网络服务上，数据的多个独立拷贝从源被发送到每一个目的地；而在多点传送网络服务上，数据仅仅从源一次发送，网络承担了将数据传输到多个位置。多点传送对许多媒体应更有效，例如视频会议。标准的 IP 支持多点传送。

RTP 使你能识别正在传送数据的类型、决定数据的包应该按什么样的次序表现和同步来自不同源的媒体流。RTP 数据包不保证包按照它们被发送时的次序到达目的地——事实上，它们不保证所有的包均到达。一直到接收端重构发送端的序列并且利用包头提供的信息检测丢失的包。尽管 RTP 不提供任何机构以确保及时交付或者提供其它的服务保证质量，但它利用能够监视数据分配质量的控制协议 RTCP[29]来控制。RTCP 也提供对 RTP 传输的控制和识别机制。

RTP 通道是一系列利用 RTP 通讯的应用程序的联结点。一个通道由一个网络地址和一对端口号来标识。其中一个端口号被用于媒体数据，而另一个端口号被用于控制(RTCP)数据。

通道的参加者可以是一台的计算机、主机或用户。参加者可以是被动接收数据的接收端（receiver），也可以是主动发送数据的发送端（sender），还可以同时充当两个角色。

每种类型的媒体可以在一个不同的通道里传送。例如，如果一个会议系统同时采用了音频和视频，那么一个通道可以用来传送音频数据，而另一个独立的通道则可以用来传送视频数据。这使得参加者可以自由选择所需要接收的媒体数据的类型，对于一个只拥有低带宽的参加者就可以只接收音频数据。

一个通道的媒体数据是被分为一系列数据包来传送的，我们称这些数据包为 RTP 流。在 RTP 流里的每一个 RTP 数据包包含两个部分：一个是结构化的头，一个是实际的数据。

RTP 应用程序通常分为从网络接收数据端（RTP 客户端）和跨越网络传

输数据端（RTP 服务器端）。某些应用是两者都做——例如，会议应用捕获和传输数据，而在同一时间它们都从网络接收数据。

对某些类型的应用，能够接收 RTP 流是必须的，例如：会议应用需要能够从一个 RTP 通道接收一个媒体流，并且在控制台上给以补偿；电话应答机应用需要能够从一个 RTP 会议通道接收媒体流并且将它们存储到一个文件。

RTP 服务应用跨越网络传输捕获的或存储的媒体流。例如，在一个会议应用中，媒体流可能从某个视频摄像头捕获并被发送到一个或多个 RTP 通道。这个媒体流可能以多种媒体格式编码并被发送到几个 RTP 的具有异种接收机的通道。

2.4 中间件的使用 —— .NET Remoting

Microsoft .NET Remoting 提供了一种允许对象通过应用程序域与另一对象进行交互的框架[30]，其体系结构如图 2-4。

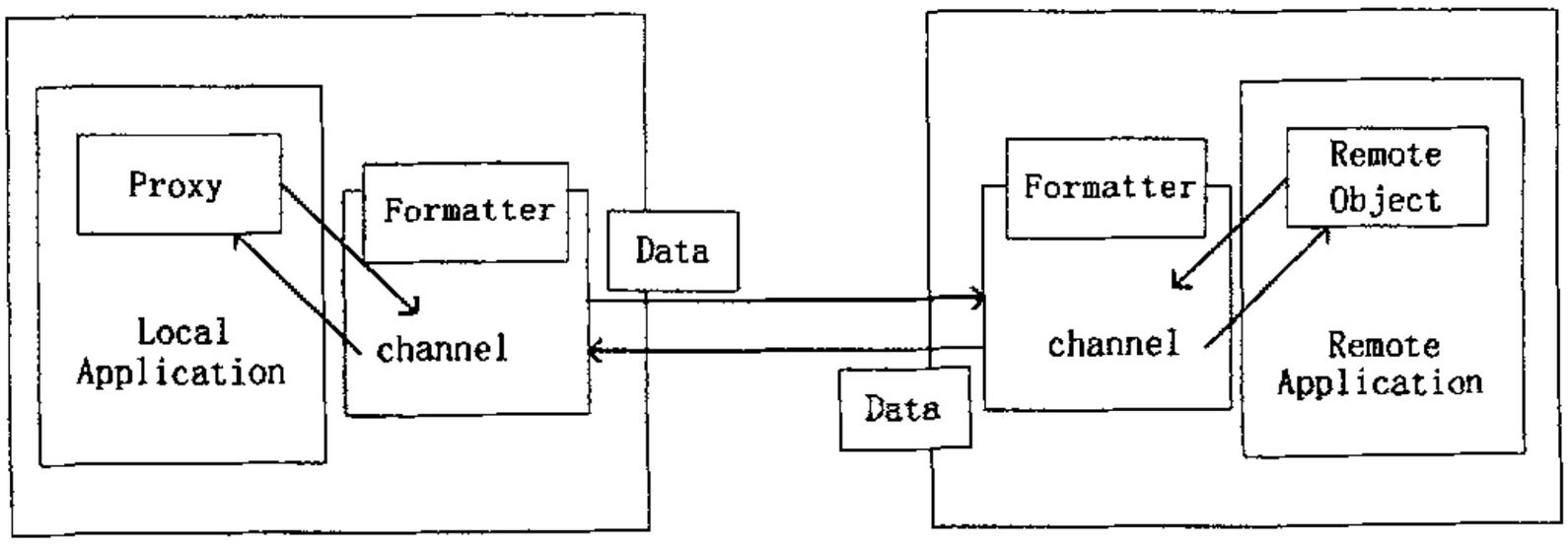

图2-4 .NET Remoting 框架结构

这种框架提供了多种服务，包括激活和生存期支持，以及负责与远程应用程序进行消息传输的通讯通道。格式化程序用于在消息通过通道传输之前，对其进行编码和解码。应用程序可以在注重性能的场合使用二进制编码，在需要与其他远程处理框架进行交互的场合使用 XML 编码。在从一个应用程序域向另一个应用程序域传输消息时，所有的 XML 编码都使用 SOAP 协议。出于安全性方面的考虑，远程处理提供了大量挂钩，使得在消息流通过通道进行传输之前，安全接收器能够访问消息和序列化流。

通常，如果没有底层框架的支持，管理远程对象的生存期会非常麻烦。.NET Remoting 提供了许多可供选择的生存期模型，这些模型分为两个

类别：

(1)客户端激活对象

(2)服务器激活对象

客户端激活对象受生存期管理器的控制，这种管理器确保了租用期满时对象可被回收。而对于服务器激活对象，开发人员则可以选择单一调用(SingleCall)模式或单一元素(Singleton)模式。

2.4.1 远程对象（Remote Object）

任何远程处理框架的主要目的之一就是要提供必要的基础结构，以便隐藏远程对象调用方法和返回结果的复杂性。任何位于调用方应用程序域之外的对象，即使在同一台计算机上执行，也会被认为是远程对象。在应用程序域内部，原始数据类型按数值传递，而所有的对象按引用传递。因为本地对象引用仅在创建对象的应用程序域内有效，所以它们不能以这种方式传递到远程方法调用或从远程方法调用返回。所有必须跨越应用程序域的本地对象都必须按数值来传递，并且应该用 [serializable] 自定义属性作标记，否则它们必须实现 ISerializable 接口。对象作为参数传递时，框架将该对象序列化并传输到目标应用程序域，对象将在该目标应用程序域中被重新构造。无法序列化的本地对象将不能传递到其他应用程序域中，因而也不能远程处理。

通过从 MarshalByRefObject 导出对象，可以使任意对象变为远程对象。当某个客户端激活一个远程对象时，它将接收到该远程对象的代理。对该代理的所有操作都被适当地重新定向，使远程处理基础结构能够正确截取和转发调用。尽管这种重新定向对性能有一些影响，但 JIT 编译器和执行引擎 (EE) 已经优化，可以在代理和远程对象驻留在同一个应用程序域中时，防止不必要的性能损失。如果代理和远程对象不在同一个应用程序域中，则堆栈中的所有方法调用参数会被转换为消息并被传输到远程应用程序域，这些消息将在该远程应用程序域中被转换为原来的堆栈帧，同时该方法调用也会被调用。从方法调用中返回结果时也使用同一过程。

2.4.2 代理（Proxy）

代理对象是在客户端激活远程对象时创建的。作为远程对象的代表，代理对象确保对代理进行的所有调用都能够转发到正确的远程对象实例。当某个客户端激活一个远程对象时，框架将创建 TransparentProxy 类的一个本地实例（该类中包含所有类的列表与远程对象的接口方法）。因为 TransparentProxy 类在创建时用 CLR 注册，所以代理上的所有方法调用都被运行库截取。这时系统将检查调用，以确定其是否为远程对象的有效调用，以及远程对象的实例是否与代理位于同一应用程序域中。如果对象在同一个应用程序域中，则简单方法调用将被路由到实际对象；如果对象位于不同的应用程序域中，将通过调用堆栈中的调用参数的 Invoke 方法将其打包到 IMessage 对象并转发到 RealProxy 类中。此类（或其内部实现）负责向远程对象转发消息。TransparentProxy 类和 RealProxy 类都是在远程对象被激活后在后台创建的，但只有 TransparentProxy 返回到客户端。ObjRef 与这两个代理类的关联方式如下（根据对象是客户端激活对象还是服务器激活对象，以及它们是单一元素对象还是单一调用对象，该进程会有所不同）：

(1)远程对象注册在远程计算机的应用程序域中。远程对象被封送以生成 ObjRef。ObjRef 包含了从网络上的任意位置定位和访问远程对象所需的所有信息，包括：类的增强名称、类的层次结构（其父类）、类实现的所有接口的名称、对象 URI 和所有已注册的可用通道的详细信息。在接收到对某个远程对象的请求时，远程处理框架使用对象 URI 来检索为该对象创建的 ObjRef 实例。

(2)客户端通过调用 new 或某个 Activator 函数（例如 CreateInstance）来激活远程对象。对于服务器激活对象，远程对象的 TransparentProxy 将在客户端应用程序域中生成并返回到客户端，这时不执行任何远程调用。只有在客户端调用远程对象的某个方法时，该远程对象才会被激活。此方案明显不适合客户端激活对象，因为客户端希望框架只在得到请求时才激活对象。当客户端调用某个激活方法时，客户端上会创建一个激活代理，并且将使用URL 和对象 URI 作为终结点在服务器的远程激活器上初始化一个远程调用。远程激活器激活该对象，然后 ObjRef 流向客户端，并被取消封装以

生成一个返回给客户端的 TransparentProxy。

(3)取消封装的过程中会分析 ObjRef 以提取远程对象的方法信息，同时还会创建 TransparentProxy 和 RealProxy 对象。在用 CLR 注册 TransparentProxy 之前，分析后的 ObjRef 内容会被添加到 TransparentProxy 的内部表中。

TransparentProxy 是一种无法替代和扩展的内部类，而 RealProxy 和 ObjRef 类则属于公共类，可以在必要时进行扩展和自定义。因为 RealProxy 类能够处理远程对象的所有函数调用，所以它是执行负载平衡等操作的理想方法。调用 Invoke 时，从 RealProxy 导出的类可以获得网络中服务器的负载信息，并将该调用路由到适当的服务器。简单地为所需的 ObjectURI 从通道请求一个 MessageSink，并调用 SyncProcessMessage 或 AsyncProcessMessage 以将该调用转发至所需的远程对象。当调用返回时，通过调用 RemotingServices 类的 PropagateMessageToProxy 将返回参数推回到堆栈中。

2.4.3 通道（Channel）

通道用于在远程对象之间传输消息。当客户端调用某个远程对象上的方法时，与该调用相关的参数以及其他详细信息会通过通道传输到远程对象。调用的任何结果都会以同样的方式返回给客户端。客户端可以选择“服务器”中注册的任一通道，以实现与远程对象之间的通讯，因此开发人员可以自由选择最适合需要的通道。当然，也可以自定义任何现有的通道或创建使用其他通讯协议的新通道。通道选择遵循以下规则：

(1)在能够调用远程对象之前，远程处理框架必须至少注册一个通道。通道注册必须在对象注册之前进行。

(2)通道按应用程序域注册。一个进程中可以有多个应用程序域。当进程结束时，该进程注册的所有通道将被自动清除。

(3)多次注册侦听同一端口的通道是非法的。即使通道按应用程序域注册，同一计算机上的不同应用程序域也不能注册侦听同一端口的通道。

(4)客户端可以使用任何已注册的通道与远程对象通讯。当客户端试图连接至某个远程对象时，远程处理框架会确保该对象连接至正确的通道。客

户端负责在尝试与远程对象通讯之前调用 ChannelService 类的 RegisterChannel。

所有的通道都由 IChannel 导出，并根据通道的用途实现 IChannelReceiver 或 IchannelSender。大多数通道既实现了接收器接口，又实现了发送器接口，使它们可以在两个方向上通讯。当客户端调用代理上的某个方法时，远程处理框架会截取该调用并将其转为要发送到 RealProxy 类（或一个实现 RealProxy 类的实例）的消息。RealProxy 将消息转发到消息接收器以进行处理。消息接收器负责与远程对象注册的通道之间建立连接，并通过通道（在不同的应用程序域）将消息从调度位置传输到远程对象本身。激活了一个远程对象后，客户端会通过调用选定通道上的 CreateMessageSink 来选择通道，并从其上检索能够与远程对象通讯的消息接收器。

部分答案在于这样一个事实：远程对象并不拥有自己的通道，而是共享通道。作为远程对象宿主的服务器应用程序必须注册要通过远程处理框架公开的对象以及所需的通道。注册后的通道会自动开始在指定的端口侦听客户请求。注册远程对象后，会为该对象创建一个 ObjRef 并将其存储在表中。当通道上传来一个请求时，远程处理框架会检查该消息以确定目标对象，同时检查对象引用表以定位表中的引用。如果找到了对象引用，将从表中检索框架目标对象或在必要时将其激活，然后框架将调用转发至该对象。对于同步调用，在消息调用期间会一直维持来自客户端的连接。因为每个客户端连接都在自己的线程上处理，所以一个通道可以同时服务于多个客户端。

生成商务应用时，安全性是一个重要问题。要满足商务要求，开发人员必须能给远程方法调用添加诸如授权或加密等安全特性。为了实现这一目标，开发人员可以自定义通道，使其能够对与远程对象之间的实际消息传输机制进行控制。在传输到远程应用程序之前，所有的消息都必须流过 SecuritySink、TransportSink 和 FormatterSink，且这些消息传递到远程应用程序后会以相反次序流过同样的接收器。

2.4.3.1 HTTP 通道

HTTP 通道使用 SOAP 协议与远程对象传输消息。所有的消息流过 SOAP

格式化程序时都被转换为 XML 格式且被序列化，所需的 SOAP 头也会被添加到该流中。也可以指定能够生成二进制数据流的二进制格式化程序。然后，数据流会使用 HTTP 协议传输到目标 URI。

2.4.3.2 TCP 通道

TCP 通道使用二进制格式化程序将所有的消息序列化为二进制流，并使用 TCP 协议将其传输到目标 URI。

2.4.4 对象激活方式

远程处理框架支持远程对象的服务器激活和客户端激活。不需要远程对象在方法调用之间维护任何状态时，一般使用服务器激活。服务器激活也适用于多个客户端调用方法位于同一对象实例上、且对象在函数调用之间维持状态的情况。另一方面，客户端激活对象从客户端实例化，并且客户端通过使用基于租用的专用系统来管理远程对象的生存期。

在可以接受客户端的访问之前，所有的远程对象都必须用远程处理框架注册。对象注册一般由宿主应用程序来完成。宿主应用程序启动后，使用 ChannelServices 注册一个或多个通道，使用 RemotingServices 注册一个或多个远程对象。已注册的通道和对象只有在注册它们的进程活动时才可以使用。如果退出了该进程，则会自动从远程处理服务中删除它注册的所有通道和对象。在框架中注册远程对象时，需要以下四项信息：

(1)包含类的程序集名称；

(2)远程对象的类型名称；

(3)客户端定位对象时将使用的对象 URI；

(4)服务器激活所需的对象模式。该模式可以是 SingleCall，也可以是 Singleton。

远程对象可以通过下列两种方式注册：调用 RegisterWellKnownType，将上述信息作为参数传递；或将上述信息存储在配置文件中，然后调用 ConfigureRemoting 并将该配置文件的名称作为参数传递。后一种方法更方便些，因为无需重新编译宿主应用程序即可改变配置文件的内容。

注册了远程对象后，框架将为该对象创建一个对象引用，然后从程序集中提取与该对象相关的必要元数据。随后，这一信息将与 URI 和程序集名

称一起存储在对象引用中（该对象引用将被写入一个用于跟踪已注册远程对象的远程处理框架表中）。除了在客户端试图调用对象上的某个方法或从客户端激活对象时以外，注册进程不会实例化远程对象自身。

任何知道该对象 URI 的客户端都可以使用 ChannelServices 注册通道，并调用 new、GetObject 或 CreateInstance 激活对象，从而获得该对象的一个代理。

GetObject 或 new 可用于服务器激活对象。使用这两个调用时不会实例化对象，实际上不会生成任何网络调用。框架从元数据获得了创建代理所需的足够信息，但并未连接到远程对象上。只有在客户端调用代理上的某个方法时才会建立网络连接。当调用抵达服务器时，框架将从消息中提取 URI，检查远程处理框架表以便定位与 URI 匹配的对象引用，然后在必要时将对象实例化，并将方法调用转发至对象。如果将对象注册为 SingleCall，则完成方法调用后该对象会取消。每次调用一个方法时，都会创建一个新的实例。GetObject 和 new 之间的唯一差别在于，前者允许指定 URL 作为参数，而后者从配置中获得 URL。

CreateInstance 或 new 可用于客户端激活对象。两者都允许使用带参数的构造函数来实例化对象。客户端激活对象的生存期由远程处理框架提供的租用服务控制。对象租用的内容在下一节中说明。

2.4.5 对象的租用生存期

每个应用程序域都包含一个用于管理其租用情况的租用管理器。所有的租用都会被定期检查，以确定租用是否已过期。如果租用过期，则会调用该租用的一个或多个发起者，使它们有机会更新租用。如果所有的发起者都不准备更新租用，则租用管理器会删除该租用并将该对象作为垃圾回收。租用管理器按照剩余租用时间的顺序维护租用列表。剩余时间最短的租用排在列表的顶端。

通过租用来管理远程对象的生存期可以作为引用计数的一种替代方法，因为当网络连接的性能不可靠时，引用计数会显得复杂和低效。尽管有人会坚持认为远程对象的生存期比所需的时间要长，但与引用计数和连接客户相比，租用降低了网络的繁忙程度，将会成为一种非常受欢迎的解决方案。

2.5 多媒体数据的处理技术

网络多媒体计算机远程监控系统是一个集多媒体音频、视频压缩、网络传输、通信和实时监控等技术为一体的综合系统。实现这一系统的最主要的困难在于数字化后的视频、音频数据量太大，为解决这个难题，可以从两方面着手：

(1)从网络着手：硬件方面，通过增加通信信道的带宽来提高通信网络的传输速度（在 2.2 节进行了讨论）；软件方面，灵活地控制传输协议和用户对视频传输应答的灵活设置，以提高视频、音频的传输效率（在 2.3 节进行了讨论）。

(2) 从多媒体本身着手：采用各种音频、视频压缩编码技术对数字化后的音频、视频进厅高倍数压缩，极大地减小数据量，从而在满足一定图像和声音质量的前提下极大地降低数据传输率，以便在现有通信网络上实现多媒体信息的传输。

简单说来就是从视频、音频压缩和传输两个方面来着手解决，本节主要讨论音视频压缩技术。

2.5.1 音频处理技术

多媒体技术的特点是计算机交互式综合处理声文图信息。声音是携带信息的重要媒体。娓娓动听的音乐和解说，使静态图像变得更加丰富多彩。音频和视频的同步，使视频图像更具真实性。随着多媒体信息处理技术的发展，计算机数据处理能力的增强，音频处理技术受到重视，并得到了广泛的应用。如：视频图像的配音、配乐；静态图像的解说、背景音乐；可视电话、电视会议中的话音；游戏中的音响效果；虚拟现实中的声音模拟：用声音控制 Web，电子读物的有声输出。

2.5.1.1 音频压缩的必要性

在传统音响技术中，声音信号的拾取、放大、传输、记录和重放均采用随时间连续变化的模拟信号。传声器、放大器、扬声器等电声设备，也都是模拟信号设备。而在多媒体计算机中，各种媒体信息均采用数字信号进行处理，因而在多媒体技术中对音频信号的处理，首要问题就是将模拟音频信号

进行数字化处理的问题。它主要包括音频信号的模/数、数/模转换及数字音频信号的压缩编码、解码等问题。

数字化声音信号的数据量可以用下面的公式表示[31]：

数字化声音信号数据量 = 采样频率 * 量化位数 * 声道数（bit/s）

如果声音信号的采样频率为 44100Hz，每个取样的量化位数为 16bit，双声道立体声，则声音信号数据量为 44100 * 16 * 2 = 1411200（bit/s），那么一分钟这样的声音信号数据量大约是 10M 字节。如此大的信息量，所以音频压缩就有着很大的意义。

2.5.1.2 数字音频信号的压缩编码

数字音频信息的编码方法很多。如果直接将采样、量化后的数字声音波形进行二进制编码，称为 PCM 编码（脉冲编码调制）。这种方法简单实用，无压缩失真，保真度高。但数据量大，传输速率也要求较高。

对于采样频率较高、量化位数较多的多通道高质量音频信号，若采用 PCM 编码，则对系统的存储容量和传输速率提出了很高的要求。如普通双声道立体声信号，采用中等质量的 16 位量化比特数和 44.1kHz 的采样频率，每分钟信息量将达 10.584MB，即容量为 600MB 的光盘也只能存放 1 个小时的音频数据，速率则要求在 1.4 Mb/s 以上。由于人耳的听觉特性，声音信号中存在着数据冗余。同时在允许一定限度失真的前提下，适当进行数据的有损压缩，也仍可使数据近似恢复，对实际应用效果影响不大，因而进行数据压缩，理论上也是可能的。

实际数据压缩方法总体上分为无损压缩和有损压缩两大类。无损压缩法包括不引入任何数据失真的各种熵编码。信息理论认为：若信息编码的熵大于信源的实际熵，该信源中一定存在冗余度。压缩时去掉冗余量不会减少信息量。典型的无损压缩法有 Huffman 编码、Fano—Shannon 编码、Lempel—Zev 编码、算术编码和游程编码等。无损压缩不产生失真，但压缩比不高，一般在 2:1～5:1 之间，因而大多用在文本和语音识别系统中。

有损压缩法又称为熵压缩法。由于它允许一定的失真，压缩比可以很大，故应用较广，对于高保真音乐、图像和活动视频信号均比较适用。

有损压缩通常分为波形编码和模型参数编码两大类。也有人把同时利用

这两种技术的混合编码划分成第三类。

波形编码[32]方法是目前采用较多的数字音频压缩编码方法。它的编码对象是声音的波形，算法简单、易于实现，而且保真度较高。缺点是压缩比不是很高，且易受量化噪声的干扰。

除 PCM 编码外，目前实际采用较多的波形编码方法有 DPCM（差分脉冲编码调制）和 ADPCM （自适应差分脉仲编码调制）两种。DPCM 只对声音信号的预测值和样本值的差值进行编码，它可以大大降低信息数据的编码率。ADPCM 则是 DPCM 的进一步改进。根据人耳的听觉特性，它对不同频段设置不同的量化比特数，通过对量化阶距的调整进一步压缩数据。

模型参数编码是将声音信号用模型参数来表示，其压缩比很大。它首先建立一个声音信号的产生模型，再对声音对应的模型参数进行编码。这种数字音频解码后，与原来声音的采样值不存在固定的对应关系，靠合成各种声音元码产生声音，因而缺乏原始声音的自然风格和音色特征，只能应用于要求不高的语音和单调音符的发声场合。

2.5.1.3 数字音频压缩标准

1992 年 CCITT 制定了基于短延时码本激励线性预测编码（LD—CELP）的 G.728 标准，其质量与 32 kb/s 的 G.721 标准基本相当，但速率只有它的一半。

GSM 标准是 1983 年欧洲数字移动特别工作组采用长延时线性预测规则码激励（RPE—LTP）制定的标准。CTIA 标准则是 1989 年美国公布的数字移动通讯标准，它们主要采用矢量和激励线性预测技术（VSELP）。这两种标准具有较大的压缩比和较高的语音质量，速率仅为 13kb/s 和 8kb/s，计算量也不很大，因而具有较好的应用前景。

两个 NSA 标准是美国国家安全局（NSA）分别于 1982 年和 1989 年制定的，主要用于保密电话通讯，其速率仅为 4.8kb/s 和 2.4kb/s。

调幅广播质量的音频信号的最高频率是 7kHz，故又称“7kHz 音频信号”。当使用 16 kHz 的采样频率和 14 bit 的量化位数时，信号速率为 224 kb/s。1988 年 CCITT 制定了 G.722 标准。它采用子带编码方法，可将信号速率压缩成 64 kb/s。同时它具有数据插入功能，可在窄带 ISDN 的一个 B 信道上

传输“7kHz”信号。由于这种压缩方法能够在每秒 8kB 的存储量下保证一定的音质，故也适合在需要存储大量音频信号的多媒体系统中使用。

对于频宽达 20～20000Hz 的高保真音频信号，目前基本上均采用“Motion Picture Expert Group”音频压缩标准，简称“MPEG”标准。它是几个国际标准化组织 ISO/IEC/JTC/SC2/WG11 的一个联合小组，最初于 1990 年制定的。后来逐步扩展，至今尚未全部完成。

MPEG 标准[33]是针对 CD—ROM 和有线电视信号制定的，它分成音频、视频和音视频同步三个部分。其中音频部分根据编码计算复杂程度及编码效率分为三个层次。层次 1 与层次 2 具有大致相同的算法。输入音频信号的抽样频率为 48 kHz，44.1kHz 或 32 kHz，经过滤波器组分成 32 个子带。利用人耳听觉的掩蔽效应计算各个频率分量的屏蔽门限，用以控制每一个子带的量化参数，达到数据压缩的目的。层次 1 的典型速率为每通道 192 kb/s，主要用于 VCD 和 DCC 数字音频；层次 2 的典型速率为每通道 128 kb/s，主要用于 DVD 和 DAB。层次 3 进一步引入辅助子带、非均匀量化和熵编码等技术，可进一步提高压缩比，而传输比特率只有 64kb/s，甚至更低，非常适合 ISDN 传输的数字通讯和多媒体数字音响系统。

G.721 标准是 CCITT 于 1984 年公布的，其速率为 32kb/s。它采用 ADPCM 编码，是一种对中等质量音频信号进行压缩编码的有效方法。不仅适用于电话通信，在调幅广播的音频信号和交互式激光唱盘的音频压缩编码中也有广泛的应用。

G.729 编码器是为低时延应用设计的，它的帧长只有 10ms，处理时延也是 10ms，再加上 5ms 的前视，这就使得 G.729 产生的点到点的时延为 25ms，比特率为 8 kbps。这些时延性能在互联网中很重要，因为我们知道任何能减少时延的因素都是非常重要的。

G.729 有两个版本：G.729 和 G.729a。这两个版本互相兼容但它们的性能有些不同，复杂性低的版本（G.729a）性能差一些。两种编码器都提供了对帧丢失和分组丢失的隐藏处理机制，因此在因特网上传输语音时，这两种编码器都是很好的选择。

G.729a 具有如下优点：

（1）压缩率高。G.729a 算法将语音压缩到 8.0Kbps，相当于具有 8:1

的高压缩率。

（2）语音质量好。经 G. 729a 算法压缩后的语音质量 MOS 值能达到 4.0 分，就是说：对于一般没有经过专门训练的耳朵，已经无法区分语音是否经过压缩，这样的语音质量完全满足了通话质量。

2.5.2 视频处理技术

在许多场合通信信道的带宽是有限制的。例如，以 Modem 接入到 PSTN 时，带宽受 Modem 速率限制。即使是宽带的网络，当通信信道由多用户共享时每个用户可得到的带宽也可能严重下降。远程监控系统，已决定了通信信道带宽是有限的。所以在视频传输前，应采用视频压缩编码技术对视频图像进行压缩，从而极大地降低数据量，以便在现有通信网络上实现活动视频的传输。目前比较流行的活动视频压缩算法主要有：MPEG－1，MPEG－2，MPEG－4，H.261，H263 算法等。

2.5.2.1 视频压缩的必要性

数字化革命给人们的生活带来了巨大的变革，数字化视频的低失真、高清晰度和易于计算机加工等优势是模拟视频望尘莫及的。但是，数字视频技术的发展与应用也面临着巨大的挑战，那就是：数字视频对存储容量和传输信道带宽的要求特别大。视频压缩编码的理论基础是信息论。压缩就是从时域和空域两方面去除冗余信息。

数字视频的传输速率（或它的数据率）可表示为：

数字视频的空间分辨率 * 每个象素的平均编码位数（或象素深度） * 数字视频的时间分辨率（或帧率）（单位：Kbit/s 或 Mbit/s）

以 VHS（Video Home System）质量的 NTSC 电视信号为例，帧率为 30 幅/秒，若分辨率为 352 * 240，每个彩色分量的象素量化位数（象素深度）为 24bit，那么，它的未经压缩的视频数据量为 352 * 240 * 24 * 30 ≈ 60.8Mb/s ≈ 7.6MB/s。那么一分钟这样的视频信号为 7.6 * 60 = 456MB/m，即每分钟的视频数据量大约是 456M 字节。

如此大量的未经压缩的音频和视频信号对存储媒体的容量和存储媒体与传输媒体的带宽要求都太高，尤其是动态视频图像信号，以目前的技术来看，未经压缩的多媒体数据的传输和存储根本不可能。为了减少数据存储容

量、降低传输数据率和缩短传输时间，必须采用高效率的压缩算法对数据进行压缩。

2.5.2.2 几种视频压缩算法的比较

MPEG-1 是 1.5Mb/s 固定码率的数字运动图像及其伴音编码压缩的国际标准[34]。视频分辨率为 352*240(NTSC)和 352*288(PAL)，视频帧率为 30 帧/秒，压缩后的视频数据量为 1.2Mb/s；音频按 44.1KHz 采样，16bit 量化精度，双声道，压缩后的音频数据量为 0.192Mb/s；控制视频及声音复用的系统流数据量为 0.1Mb/s。它主要用于中等图像质量的视频存储和传输的编码表示和解码规定。优点：图像质量较好，同时还有伴音。缺点：数据量较大。

MPEG-2 是 4Mb/s～15Mb/s 运动图像及其伴音的压缩编码国际标准。视频分辨率为 720*480(NTSC)和 720*576(PAL)。它主要用于数字视频广播(DVB)、高清晰度电视(HDTV)和数字视盘(DVD)等高质量的视频存储和传输的编码表示和解码规定。优点：图像质量很好，有伴音。缺点：数据量非常大。

MPEG-4 是 4800bps～10Mbps 下的可变码率的音频和视频压缩编码标准。视频分辨率为 352*288(CIF)和 176*144(QCIF)等。它主要用于可视电话、视频电子邮件等的压缩标准，它也支持不同传输信道的不同码率，如：4800bps～64kbps 的低速通道；64kbps～384kbps 的中速通道；384kbpS～2Mbps 的高速通道。优点：图像质量可以调节，有伴音，数据量从小到大可变，具有基于内容检索功能。缺点：由于标准刚刚制定，目前成熟的软硬件产品不多。

H.263 是小于 64Kb/s 下的可变码率的甚低码率视频压缩编码标准。视频分辨率为 352*288(CIF)和 176*144(QCIF)等。它主要用于电视会议和可视电话等压缩标准。优点：图像质量可以调节，压缩率高，数据量从小到大可变，软硬件产品成熟。缺点：仅有视频压缩不包括声音压缩。

3 系统设计与实现

3.1 平台功能需求

平台需要实现以下功能：

(1) 文件传输：监控站与工作站之间可以传送任意大小的文件；

(2) 屏幕监视：监控站可以监控任意工作站的屏幕内容；

(3) 远程进程控制：监控站可以启动、显示、关闭工作站上的用户进程；

(4) 远程硬件控制：监控站可以屏蔽工作站的键盘、鼠标信息，从而锁定屏幕，控制工作站重起、关机；

(5) 音频广播：监控站可以通过麦克风向所有工作站广播；

(6) 视频监控：监控站可以通过摄像头监视任意工作站的工作情况；

(7) 程序共享：所有工作站可以共享监控站上指定的应用程序；

(8) 电子白板：所有工作站可以在共享白板上绘图，交流设计草案；

(9) 文本交谈：工作站与监控站使用文本进行交流。

3.2 系统开发的环境

(1) 系统开发的硬件环境为：

PC 机配置：采用 Intel Pentium 4 1.6G 中央处理器；256M RAM；由于需要视频采集，需要采集设备，这里采用鹰眼 CU99A USB 摄像头；输入输出音频采用 SoundMAX 声卡，以及带有话筒的耳机。

网络环境：以太网是目前最便宜、最简单并且使用最广泛的局域网技术，几乎支持所有的高层网络协议。处理连续媒体通信（如音频和视频）时会产生大量的实时数据，因此低层网络必须能提供高数据传输率来支持连续媒体的实时传输。根据现有条件，本平台选择了 100Mbps 快速以太网[35]。

(2) 系统开发的软件环境为：

操作系统：Windows 2000。

开发平台：Microsoft .NET 平台[36]，Visual Studio.NET[37]。

3.3 平台分层体系结构模型

为了使系统中各模块结构清晰、功能明确，平台采用分层的体系结构，下层为上层提供服务，上层调用下层的功能。由下到上具体分为数据传输层、功能实现层和功能接口层。实时多媒体监控平台的体系结构如图 3-1 所示。

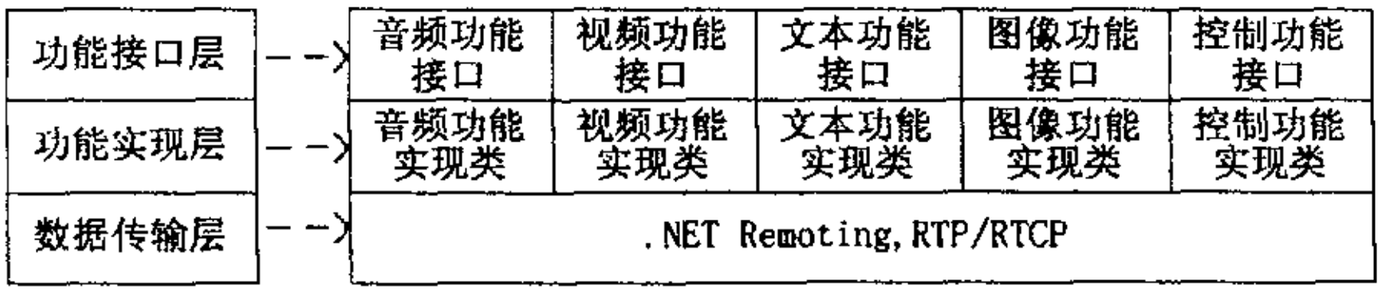

图3-1　实时多媒体监控平台的体系结构

数据传输层是平台的最底层，担负着多种类型数据的传输任务。由于此层位于 TCP/IP 参考模型的应用层，所以既可以利用其他应用层协议，也可以直接利用 TCP/IP 参考模型的传输层协议。根据不同业务需要，数据可采用不同的传输协议。对于文件数据等要求可靠传输的数据，采用 TCP 协议或 HTTP 协议传输；对于实时播放的音视频数据，不需要可靠传输，应采用 UDP 协议或 RTP/RTCP 协议。

功能实现层负责具体实现平台的各种功能。在实现功能时充分利用数据传输层提供的服务，这样就可将主要精力放在客户端或服务器端系统的本地功能实现上。考虑到系统的灵活性，此层设计为可以替换的。当需要对业务逻辑进行更新，而不希望对功能接口层进行改动，则可以用新版本的组件替换旧版本的组件。

功能接口层作为提供给具体应用的访问接口，又作为功能实现层的具体实现的标准。

由于平台基于微软的.NET 开发平台，采用面向组件设计思想，各层之间、各功能之间实现了松散耦合[38]。

3.4 数据传输层的设计与实现

数据传输层处理网络数据传输的所有细节，根据数据传输的不同特点，将功能分为两大部分，一部分为可靠数据传输，基于对象中间件.NET Remoting 技术；另一部分为不可靠数据传输，基于 RTP/RTCP 协议。

数据传输层的传输机制如图 3-2 所示。

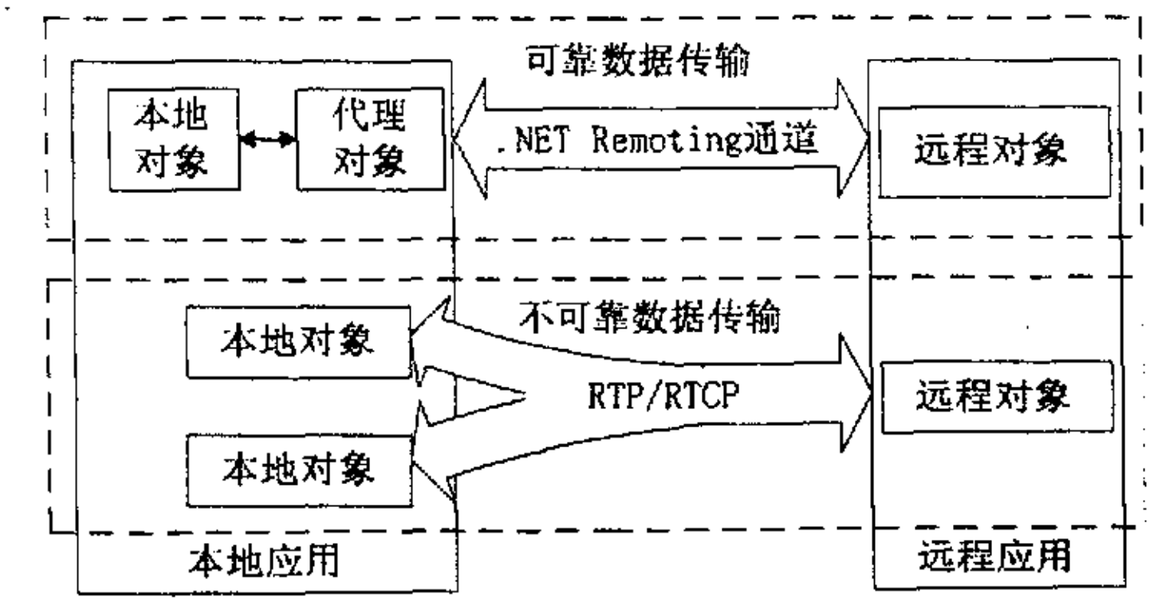

图3-2　数据传输层中的传输机制

可靠数据的传输利用.NET 框架中的.NET Remoting 技术[39]。当客户端调用某个远程对象的方法时，首先与代理对象交互，代理对象包含远程对象的描述，然后代理对象将调用参数以及其他详细信息经过通道传输到远程对象，远程对象从通道取出调用信息并执行，最后调用的结果返回给客户端。

.NET Remoting 提供了完整的消息传递机制，保证消息可靠地在客户端和服务器之间传输。但有些数据不需要可靠传输，不宜采用.NET Remoting，本平台中直接采用.NET 框架类库中的网络功能来实现此类传输。

.NET 框架的类库中的 UdpClient 类封装了利用 UDP 协议请求和接收数据的各种功能，抽象了建立 Socket 实例进行通信的具体细节。UdpClient 类还提供了组播功能，JoinMulticastGroup 方法用于加入组播组，DropMulticastGroup 方法用于退出组播组。

音视频媒体数据流实时性要求远大于可靠性要求，传输协议应该采用基于 UDP 协议的 RTP/RTCP[40]。RTP 提供了序列号和时间戳等项目来保证端到端的数据传输。与 RTP 配合的 RTCP 提供 QoS 控制、媒体同步和身份确认等功能。基于 UDP 的 RTP/RTCP 可以采用组播方式通信，尤其在客户端很多的情况下可大大节约网络带宽资源。

平台通过 MulticastManager 类管理组播数据的发送接收，RTPSessionManager 类管理 RTP 会话，RTCPSessionManager 类管理 RTCP 会话，RTCPTimer 类负责周期性通知 RTCPSessionManager 发送 RTCP 报文。

RTP 发送数据的过程是：①数据压缩编码；②加 RTP 报头；③由 MulticastManager 以组播发送到网络。

RTP 接收数据的过程是[41]：①分析 RTP 报头，判断其版本、长度、负载类型等信息的有效性，检查同步源冲突；②为 RTCP 保存统计信息；③按照时间戳和序列号进行源同步、RTP 分组排序，进入分组队列；④数据解码。

RTCP 报文发送和接收过程如图 3-3 所示。

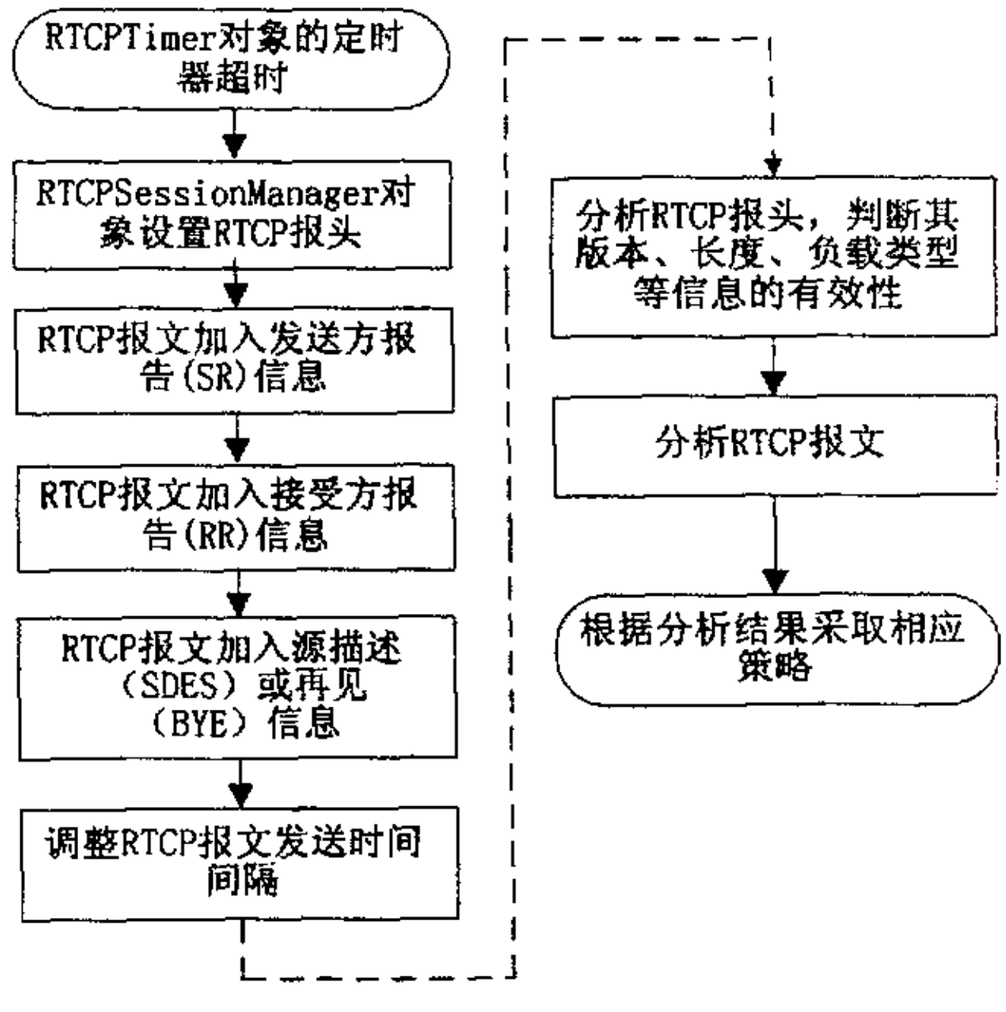

图3-3 RTCP报文发送和接收过程

3.5 功能实现层的设计与实现

功能实现层负责具体实现功能接口层定义的一组接口。接口不包括方法的实现，要通过类或结构来实现接口。功能实现层采用面向对象的方法，结合.NET 框架提供的各种服务，实现了功能接口层定义的各种功能。由于有数据传输层的支持，远程对象之间的消息通信细节被屏蔽掉了，可以像使用本地对象一样使用远程对象；数据发送方式可以有多种选择，可以是单播，也可以是组播；可以是可靠传输，也可以是不可靠传输。

3.5.1 控制功能实现

控制功能是指管理客户端与服务器多媒体会话的信令功能，包括控制多媒体会话的初始、参数改变以及结束等。基于纯 IP 的 SIP 协议是用于创建、修改、终止 IP 网上的多媒体会议或呼叫的应用层控制协议，它借鉴了 HTTP

协议和 SMTP 协议，结构简单并具有可扩充性和可扩展性。此外，SIP 还提供良好的 QoS 支持，这对于在 IP 网络上实现 VoIP 和多媒体通信来讲，SIP 具有独特的优势，将成为下一代网络 VoIP 的重要解决方案。

3.5.1.1 SIP

传统电话技术基于一种媒体。现在不同了，电话可以采用不同质量和不同编码连接到一个电视，照相机，其它电话。SIP 独立于任何应用程序的媒体[42]。SIP 能够在会话中协商媒体。任何多媒体应用都能 SIP 建立会话。

会话初始协议 SIP 是一个应用层控制协议，它可以建立，修改和终止多媒体会话或呼叫。这些多媒体会话包括多媒体会议，远程学习，因特网电话会议以及类似的一些应用。SIP 邀请的对象既可以是活生生的人也可以是“机器人”，比如说是一台用来将会话过程记录下来的存储设备。SIP 既可以邀请用户加入到单播会话也可以邀请他们加入组播会话。而会话的发起者可以不必是它邀请别人加入的那个会话的一个成员。加入到一个已存在的会话中的可以是一个人也可以是一个媒体设备。

SIP 可以用来初始化一个会话；也可以邀请成员加入通过其它方式建立起来的会话。这些通过其它方式建立起来的会话可以用组播协议、Email、LDAP 或网页等方式来公布出来。

SIP 支持名字匹配和重定位服务，这些能力使得对移动用户的支持成为可能。对于用户移动性，在通信智能网服务的术语中是这样定义的：“用户移动性是指端用户在任何地点在任一台终端上发送和接收呼叫并且获取要求的通信服务以及网络能在用户移动时识别他们的能力。用户移动性是基于用户标志唯一性的”。用户移动性是对终端移动性的增补。终端移动性是指将一个端系统从一个子网移动到另一个子网时维持通信的能力。SIP 支持建立和终止多媒体通信的 5 个方面的功能：

(1)用户端定位（user location）：对在通信中使用的端系统的位置的确定。

(2)用户能力（user capabilities）：决定在通信中要使用的媒体和媒体参数。

(3)用户可行性（user availability）：确定被呼叫方是否愿意加入通

信。

(4)呼叫建立（call setup)：呼叫方和被呼叫方双方建立呼叫参数。

(5)呼叫处理（call handling)：包括转移和结束呼叫。

SIP 也可以使用多点控制单元（MCU）或由单播组成的全联结的网来代替组播以进行多方呼叫的初始化。联结多个公共交换电话网（PSTN）的因特网电话网关也可以使用 SIP 在它们之间建立呼叫。

SIP 是被设计作为 IETF 的多媒体数据和控制体系的一部分，其它还有如预约网络资源的 RSVF，传输实时数据和提供 QOS 反馈的 RTP，控制流媒体的发送的 RTSP，用组播来宣布多媒体会话的 SAP；描述多媒体会话的 SDP[43]。然而，SIP 的功能和操作是不依赖于这些协议的。

SIP 也可以和其它的呼叫建立和信令协议一起使用。在这种模式下，一个端系统使用 SIP 的交换信息来从一个给定的独立于协议的地址中确定适当的端系统地址和协议。比如说，可以用 SIP 来确定一个用户是否可以通过 H.323 来到达[44]，是否可以获得 H.245 网关，然后用 H.225.0 来建立呼叫。SIP 不提供会议控制服务并且没有规定会议该怎么进行管理，但是 SIP 可用于引进会议控制协议，SIP 不分配组播地址。SIP 可以邀请用户加入会话而不管是否进行过资源预约。SIP 不预约资源，但是可以向被邀请的系统传达必要的信息以进行资源预约。

SIP 透明支持名字映射和重定向服务，允许 ISDN 和智能网电话用户服务[45]。这些设施也支持个人移动性。

本课题在局域网环境中实现了 SIP 协议栈及 UA[46]功能，省略其它功能（如 Proxy Server[47]）。

3.5.1.2 SIP 协议栈的实现

SIP 协议栈依照 RFC2543 来设计，根据课题需要，目前实现了部分主要功能。协议栈的组成如图 3-4。

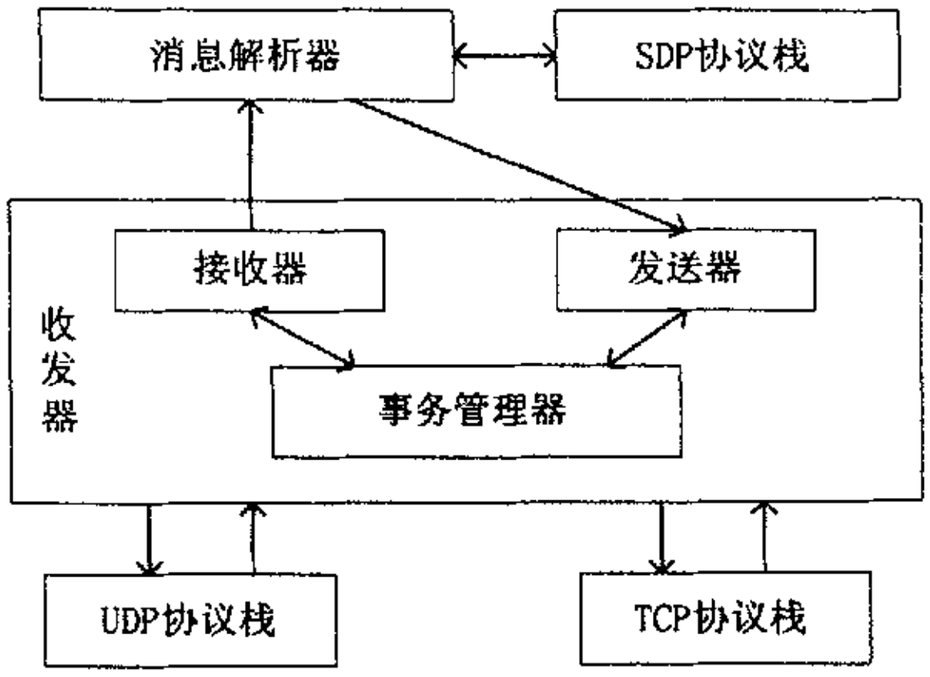

图3-4 SIP协议栈的组成

SIP 的各种请求或应答消息由消息解析器操作，它对消息进行分析，修改和编解码等操作。SDP 协议栈主要用来协助消息解析器操作 SIP 消息体中与媒体流等会话信息有关的内容。由于系统中可能存在多个 SIP 事务，事务管理器负责保存事务及相关状态信息。当利用 UDP 协议栈传输消息时，还需要重传和过滤功能，它们结合事务管理器来判断消息是否需要重传操作。

当收到一个 SIP 消息，消息解析器将它分解为键-值对，键标识 SIP 头，值是未解析的 SIP 头的值。SipHeader 对象包括一个含有键-值对的二维数组，键是一个枚举型的头类型，值是 SipHeaderValue 对象。静态接口 SipMessage::CreateMsg()用于解析一个 SIP 消息并建立一个消息对象。SipMessage 对象提供访问单个头对象的接口。当应用程序请求头时，代表一个头的头对象被建立，它负责头的解析。例如，如果应用程序请求 FROM 头，一个 SipFrom 对象被建立，FROM 值的解析由 SipFrom 对象来完成。任何对头的成员的访问都会导致头值被解析。

事务管理器通过两个事务数据库维护事务状态：SipSentReqDB 和 SipSentRespDB。SipSentReqDB 管理 UAC 的事务，包括 SIP 请求发送，SIP 响应接收。SipSentRespDB 管理 UAS 的事务，包括 SIP 响应发送，SIP 请求接收。

事务数据库根据呼叫的特点，采用层次型组织。逻辑上，一次呼叫是一组呼叫对（Call Leg）的集合，所以第一层为呼叫对节点。SIP 中的呼叫对是指两个 UA 之间端到端的连接，在一个呼叫对之中的每个方向上都存在多

个事务，每个呼叫对由 To、From 和 CallId 标识。第二层节点表示在呼叫对中每个修正或分支，由 CSeq 序号和顶端 via branch 标签标识。第三层用于区别 INVITE 和 CANCEL 的响应，采用 Cseq 方法标签标识。

SIP 协议栈通过几个线程来执行各种功能，例如发送或接收消息，会话定时器等。各线程的交互模型如图 3-5。

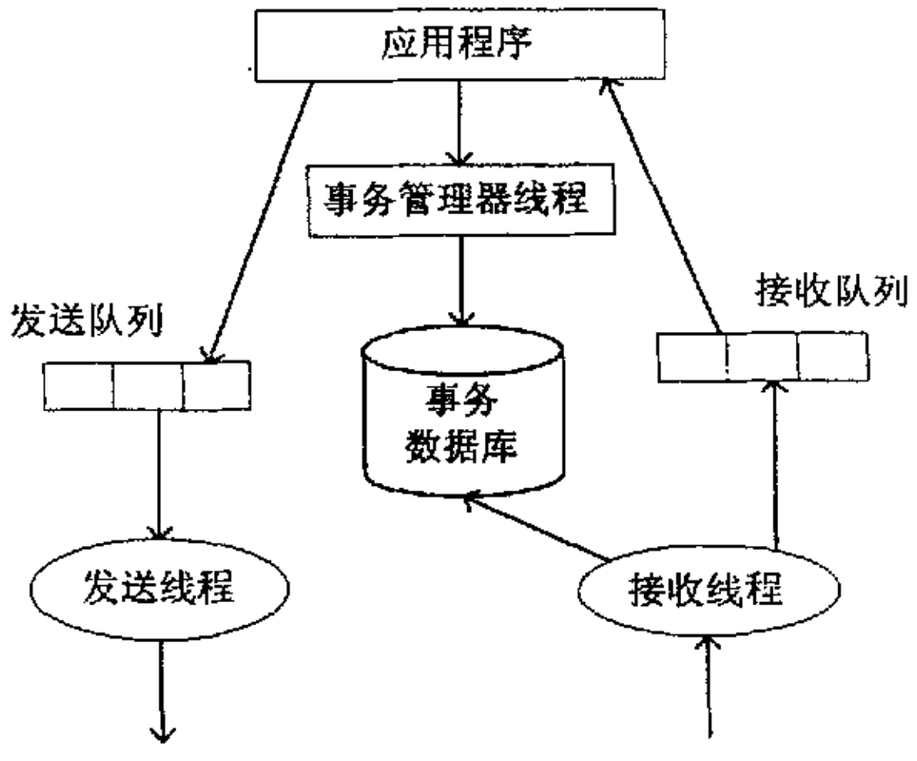

图3-5　SIP协议栈线程交互模型

接收线程接收消息，经过解析后创建消息对象，然后参照事务数据库检查消息是否重复。如果发现消息重复，则丢弃消息；如果没有发现重复消息，则消息进入接收队列。相应的应用程序线程从接收队列取出消息并进行处理。处理完成产生响应消息，并置入事务数据库和发送队列，发送线程从发送队列取出消息并发送到网络。

3.5.1.3 SIP UA（User Agent）的实现

SIP UA 用户代理是呼叫的终端系统元素，用户代理本身具有一个客户机元素 UAC（User Agent Client，用户代理客户机）和一个服务器元素 UAS（User Agent Server，用户代理服务器）。客户机元素初始呼叫，服务器元素应答呼叫。这就使点到点的呼叫通过客户/服务器方式来完成。SIP UA 代理用户生成各种请求消息，以便建立会话，改变会话属性，取消正在进行得 SIP 事务或终止已经存在的会话等。给上层应用程序提供一套控制会话的 API。UAC 和 UAS 一同为用户提供全双工点到点呼叫的基本功能。UA 端的呼叫和应答状态 UML 活动图[48]如图 3-6。

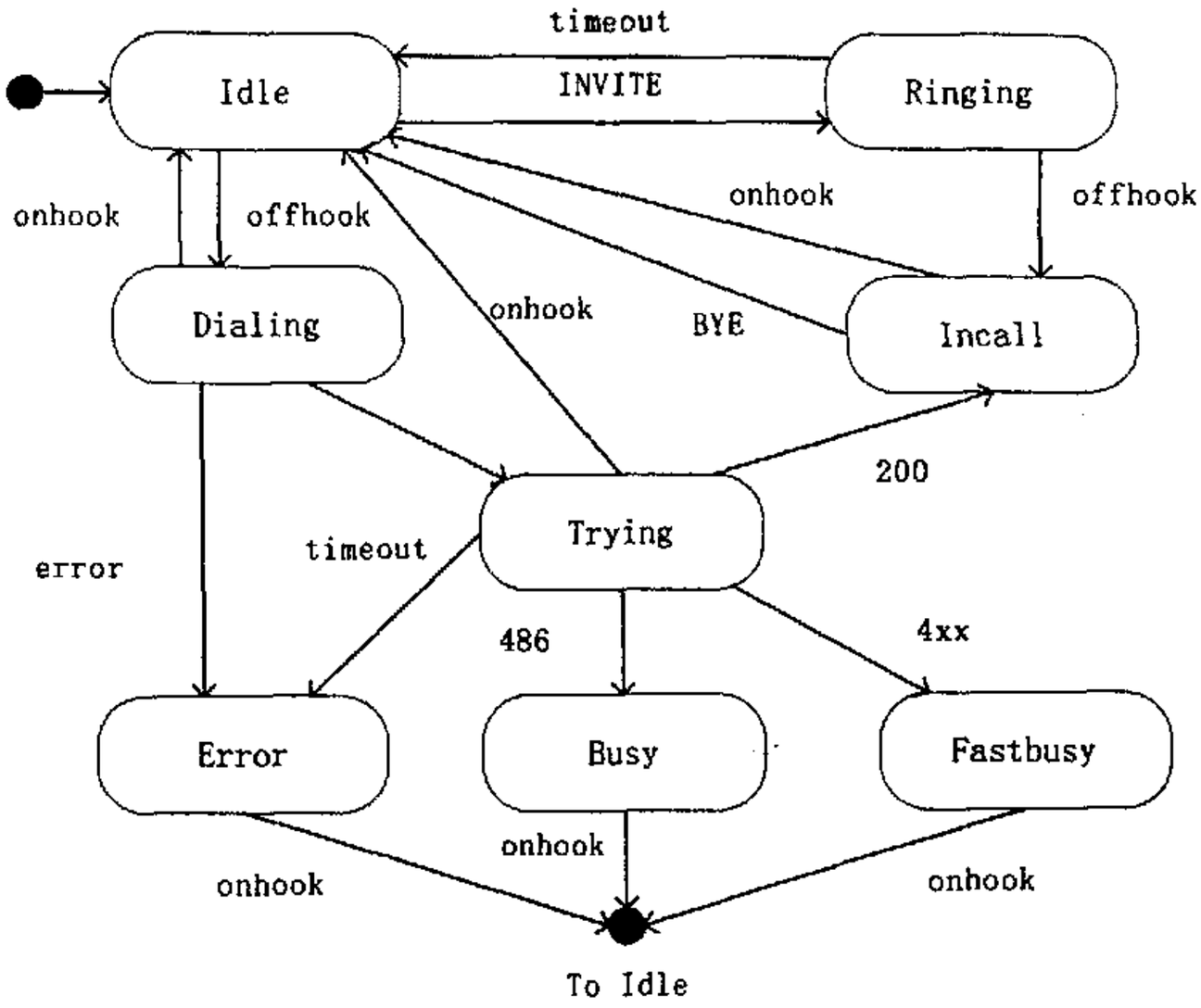

图3-6 UA端的呼叫和应答状态UML活动图

系统在初始化以后会进入 Idle 状态，这个时候系统在等待本地发出的呼叫和远端进入的呼叫，如果用户使触发摘机(offhook)事件，那么这个事件会发送到本地事件处理队列，这样 HandleDial 方法会检测到这个事件并且系统进入 Dialing 状态；这时用户输入被叫地址，在 Dialing 状态中所有的事件被处理的过程并不会让当前的状态从 Dialing 离开，而是维持到输入地址完毕。

当用户输入完毕以后，这个时候会产生一个“呼叫完毕”事件，这个事件是由HandleInvite方法产生,这个操作负责把 INVITE 消息发送到被叫端，并且让系统陷入 Trying 状态。

一旦用户进入 Trying 状态，主叫方将会等待被叫摘起话筒（也就是点击确认按钮）的过程，如果被叫摘机，那么将会发送一个 200 的消息给主叫，主叫端的HandleResp方法将会处理这个消息,并且让系统进入Incall状态，两端的UA之间将打开 RTP/RTCP 通道,开始音视频通讯,当用户挂机(onhook)以后，进入 HandleTerminal 状态，并且互相发送 Bye 消息，结束会话，系统最终返回 Idle 状态[49]。

3.5.2 音视频采集及音视频回放实现

音视频采集及音视频回放采用 DirectX 技术中的 DirectShow[50]技术实现。DirectShow 提供了播放本地文件和 Internet 服务器上多媒体数据，以及从音视频采集卡捕获多媒体流的功能。最新版本的 DirectX9.0 提供了 Managed DirectX 开发包，用于支持在.NET 平台上开发多媒体应用。但 DirectX9.0 还没有提供 Managed DirectShow 技术，需要通过 CLR 的 COM Interop 服务来使用以 COM 组件形式提供的 DirectShow 技术[51]。有多种方法可以将 COM 技术结合到.NET 中，平台采用在编程语言中重写 COM 接口技术，这种技术虽然工作量大一些，但提供了最大的灵活性。首先查看接口描述语言（IDL）描述的 COM 组件的接口内容（DirectX SDK 以.idl 和.odl 文件形式提供），然后将 IDL(ODL)的表达转换为相应编程语言的表达，转换中需要添加属性(Attribute)信息，最后编译为.dll 文件。这样在.NET 平台中使用 DirectShow 功能时，就可以像使用.NET 组件一样使用 DirectShow COM 组件[52]。

平台中基于 DirectShow 的音视频采集和回放流程如图 3-7 所示。

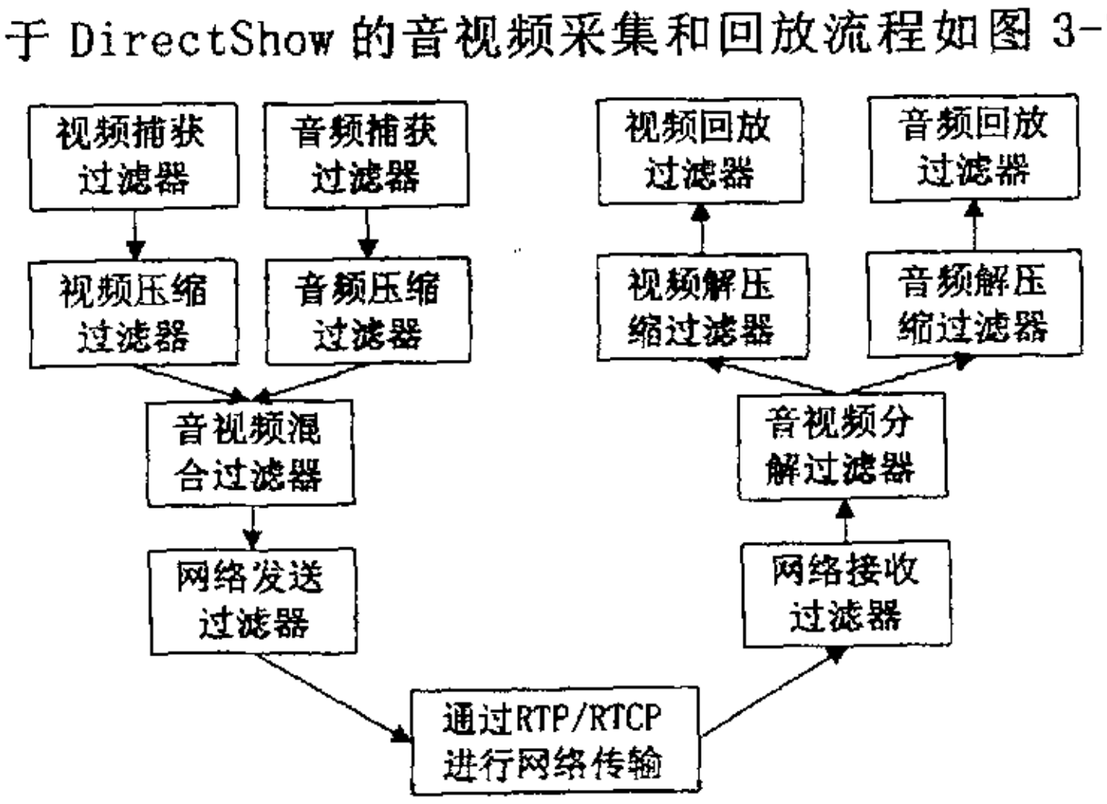

图3-7 基于DirectShow的音视频采集和回放

.NET 类库包含了绝大部分 Wine32 API 的功能，但有些底层功能没有在.NET 类库中实现。控制软硬件资源除了需要用到.NET 框架类库外还需要 WIN32 API 的支持，.NET 平台调用服务(PInvoke)允许受控代码调用非受控的 WIN32 DLL 代码，通过 DllImport 属性来声明要调用的函数信息，然后就可以在受控代码中调用这些函数。比如获取远程桌面信息会用到

GetDesktopWindow 函数，需要如下声明：

```
// 说明包含外部方法实现的动态连接库的位置：
[DllImport("user32.dll")]
//用到的外部方法：
private static extern IntPtr GetDesktopWindow();
```

然后就可以调用它获取桌面图像：

```
//取桌面内容到 Graphic 对象：
Graphics Graphic = Graphics.FromHwnd(GetDesktopWindow());
```

3.5.3 文件传输的实现

下面所示的传输多媒体文件功能 GetMultimediaFile，文件需要可靠的端到端传输，平台采用数据传输层的.NET Remoting 的基础设施，将文件路径 MultimediaFilePath 以消息调用的方式传递，这个过程中看不到底层传输的动作，不需要关心套接字的具体实现细节，客户端通过返回结果得到需要的文件内容。

```
//服务器端 File 类的 GetMultimediaFile 方法，采用.NET Remoting 的 TCP 通
//道传输多媒体文件；
// 传输 MultimediaFilePath 文件；
public byte[] GetMultimediaFile(string MultimediaFilePath)
{
    // 打开文件；
    IO.Stream str= IO.File.Open(MultimediaFilePath, IO.FileMode.Open);
    byte[]  MultimediaData= new byte[str.Length];
    // 读入文件的内容到 MultimediaData；
    str.Read(MultimediaData, 0, (int) str.Length);
    // 返回给客户端的文件内容；
    return MultimediaData;
}
```

由此可见，功能实现层所实现的主要是传输功能以外的部分，当需要传输数据时，交付给下层执行。

3.5.4 远程控制实现

为了实时控制远程计算机，首先要在服务器端配置.NET 服务器端组件，并启动服务。客户端需要有服务器端服务对象的代理对象，激活远程服务对象后，客户端就可以通过中间件.NET Remoting 调用服务对象的方法。客户端取得远程计算机的桌面图像，在客户端操作此桌面，然后利用程序模拟本地鼠标和键盘动作，并将动作指令通过.NET Remoting 发送到远程计算机，远程计算机收到指令后，由服务器端对象解释执行[53]。

```
// 以下是服务器端对象 RemoteObject 的部分定义及实现代码;
public class RemoteObject : System.MarshalByRefObject
{   //取得桌面图像大小;
    public Size GetDesktopBitmapSize()
    { return new Size(GetSystemMetrics(SM_CXSCREEN),
                                        GetSystemMetrics(SM_CYSCREEN));
    }
    //模拟客户端鼠标操作;
    public void PressOrReleaseMouseButton(bool Press, bool Left, int X,
int Y)
        {   //定义输入动作;
            INPUT input = new INPUT();
            input.type              = INPUT_MOUSE;
            input.mi.dx             = (uint) X;
            input.mi.dy             = (uint) Y;
            input.mi.mouseData      = 0;
            input.mi.dwFlags        = 0;
            input.mi.time           = 0;
            input.mi.dwExtraInfo = 0;
            if (Left)
            {  //如果是左键作操;
                input.mi.dwFlags = Press ? MOUSEEVENTF_LEFTDOWN :
                                                MOUSEEVENTF_LEFTUP;
            }
            else
            {//如果是右键操作;
                input.mi.dwFlags = Press ? MOUSEEVENTF_RIGHTDOWN :
```

```
                                                MOUSEEVENTF_RIGHTUP;
        }
         // SendInput 函数将INPUT结构中的事件串行地插入到鼠标输入流中;
            SendInput(1, ref input, Marshal.SizeOf(input));
    }
  //移动光标位置
    public void MoveMouse(int x, int y)
    { //调用系统API函数SetCursorPos完成实际操作;
        SetCursorPos(x, y);
    }
    //模拟客户端键盘操作;
    publi c void SendKeystroke(byte VirtualKeyCode, byte ScanCode, bool
                                KeyDown, bool ExtendedKey)
    {   ……
    }
    //取得远程桌面位图;
    private Bitmap GetDesktopBitmap()
    {   //取得桌面位图的大小;
        Size DesktopBitmapSize = GetDesktopBitmapSize();
        //Graphic与桌面窗口联系起来;
       Graphics Graphic = Graphics.FromHwnd(GetDesktopWindow());

        //按照Graphic的分辨率创建空位图;
        Bitmap MemImage = new Bitmap(DesktopBitmapSize.Width,
                                     DesktopBitmapSize.Height, Graphic);
        //MemGraphic与MemImage联系起来;
       Graphics MemGraphic = Graphics.FromImage(MemImage);
       IntPtr dc1 = Graphic.GetHdc();
       IntPtr dc2 = MemGraphic.GetHdc();
       //从dc1拷贝到dc2，也就是Graphic位图内容拷贝到MemGraphic;
            BitBlt(dc2, 0, 0, DesktopBitmapSize.Width,
                              DesktopBitmapSize.Height, dc1, 0, 0, SRCCOPY);
            ……
        return MemImage  //返回桌面位图;
    }
    //比较两幅位图是否相同;
```

```
private static bool BitmapsAreEqual(ref byte[] a, ref byte[] b)
{   ……
}
 //读取位图数据；
public byte []GetDesktopBitmapBytes()
{   Bitmap CurrentBitmap = GetDesktopBitmap();//取到桌面位图；
    MemoryStream MS = new MemoryStream();
CurrentBitmap.Save(MS, ImageFormat.Png); //桌面存储到内存流；
CurrentBitmap.Dispose();
MS.Seek(0, SeekOrigin.Begin);//定位到开始位置；
byte []CurrentBitmapBytes = new byte[MS.Length];
int NumBytesToRead = (int) MS.Length;//要读取的字节数；
int NumBytesRead = 0;//已读字节数；
while (NumBytesToRead > 0)
{//循环读取位图数据
    int n = MS.Read(CurrentBitmapBytes,NumBytesRead,
                                        NumBytesToRead);
    if (n==0)  // 到达文件末尾；
        break;
        NumBytesRead   += n;
        NumBytesToRead -= n;
    }
    MS.Close();
    byte []Result = new byte[0];
    Result = CurrentBitmapBytes;
    return Result;
}
```

3.5.5 文本交流实现

文本交流组件允许任何客户端连接到服务器，然后与其他客户端进行文本交流。利用.NET Remoting 灵活，强大的网络通信支持功能，可以选用多种通信协议进行客户与服务器之间的文本传输。各组件的组成结构如图 3-8。

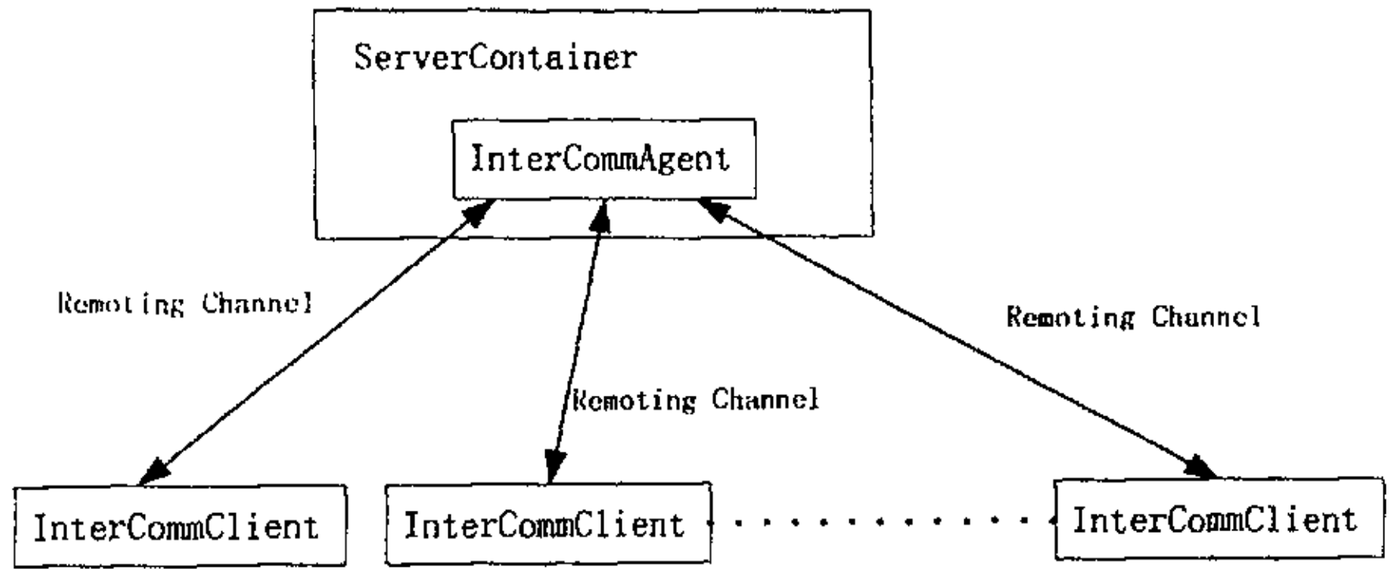

图3-8 文本交流模块中客户端与服务器交互模型

ServerContainer 是远程服务器端对象的容器。容器中的对象 InterCommAgent 是整个模块的枢纽部分。它负责协调各个客户端对象 InterCommClient 之间的交互过程，通过委托(Delegate)和事件(Event)模型来管理客户端与服务器之间的协作[54]。

InterCommAgent 在远程服务器注册为 Singleton 对象。Singleton 对象对多个客户提供服务，并通过存储的客户请求状态共享对象的数据。Singleton 对象适合在客户之间需要共享数据，以及产生、维护对象的费用较少的场合。InterCommAgent 类的定义如下：

```
Namespace   InterCommManger
{ //定义委托类型;
   public delegate void ReceivedMsg(string username,string text);
   public delegate void Userjoined(string username);
   public delegate void Userleft (string username);
   public class InterCommAgent: MarshalByRefObject
   {    //定义文本交流的各个事件;
        //有客户端加入事件;
       public event Userjoined  evtUserjoined;
        //收到文本消息事件;
   public event ReceivedMsg evtReceivedMsg;
   //客户端离开事件;
   public event Userleft    evtUserleft;
   //定义Users数组，用于维护已经加入交流的各个客户端用户;
   public System.Collections.ArrayList Users;
   ……
```

```
public InterCommAgent ()
{   //初始化用户数组;
    Users = new System.Collections.ArrayList();
}
public void SendText(string username,string text)
{   //触发各个客户端evtReceivedMsg事件;
    evtReceivedMsg(username,text);
}
public void AddUser(string username)
{  //触发各个客户端evtUserjoined事件;
    Users.Add(username);
    evtUserjoined(username);
}
public void RemoveUser(string username)
{ //触发各个客户端evtUserleft事件;
    Users.Remove(username);
    this.evtUserleft(username);
}
private bool CheckUserName(string username)
    { //检查是否已经存在此用户，没有则返回false;
        ...
    }
    public string GetServerName()
    { //取服务器名;
        ...
    }
  }
}
```

每个客户端对象 InterCommClient 通过.NET Remoting 连接到服务器端对象后，触发在服务器端定义好的各个事件(evtUserjoined，ReceivedMsg，Userleft)。服务器端事件的处理代码又是在各个客户端对象中定义的，这种实现事件处理的方式可以理解为一种回调。

在客户端对象初始化时将事件与相应处理函数进行映射，例如以下代码将 InterCommAgent 实例 LInterCommAgent 的事件 evtReceivedMsg 与客户端

的处理函数 OnReceiveMsg 联系起来：

```
LInterCommAgent.evtReceivedMsg +=
                    new InterCommManger.ReceivedMsg (this.OnReceiveMsg);
```

客户端的图形用户界面中有一个文本消息发送按钮 SendMsgButton，它的点击事件处理代码为：

```
private void SendMsgButton_Click(object sender, System.EventArgs e)
{    //通过调用LInterCommAgent的SendText来发送文本：
    this.LInterCommAgent.SendText(UserName, TextMsgCtrl.Text);
    ……
}
```

可以看出，系统的流程可归结为：客户端注册事件处理函数，客户端触发事件，服务器端调用所有客户端已经注册的事件处理函数来相应事件。

3.6 功能接口层的设计与实现

3.6.1 接口（Interface）

接口描述了组件对外提供的服务[55]。在组件和组件之间、组件和客户之间都用过接口进行交互。因此组件一旦发布，它只能通过预先定义的接口来提供合理的、一致的服务。这种接口定义之间的稳定性使客户应用开发者能够构造出坚固的应用。一个组件可以实现多个组件接口，而一个特定的组件接口也可以被多个组件来实现。

组件接口必须是能够自我描述的。这意味着组件接口应该不依赖于具体的实现，将实现和接口分离彻底消除了接口的使用者和接口的实现者之间的耦合关系，增强了信息的封装程度。同时这也要求组件接口必须使用一种与组件无关的语言。目前组件接口描述的标准是 IDL 语言。

由于接口是组件之间的协议，因此组件的接口一旦发布，组件的生产者就应该尽可能地保持接口不变，任何对接口语法或语义上的改变，都可能造成现有组件与客户之间的联系遭到破坏。

每个组件都是自主的，有其独特的功能，只能通过接口与外界通信。当一个组件需要提供新的服务时，可以通过增加新的接口来实现，不会影响原接口已存在的客户。而新的客户可以重新选择新的接口来获得服务。

从技术上讲，接口是一组包含了函数型方法的数据结构。通过这组数据

结构，客户代码可以调用组件对象的功能。接口具有不变性，但这并不意味着接口不再发展。类似于类的继承性，接口也可以继承和发展。接口继承和类继承不同。首先，类继承不仅是说明继承，而且也是实现继承；而接口继承只是说明继承。也就是说，派生类可以继承基类的方法实现，而派生的接口只继承了父接口的成员方法说明，而没有继承父接口的实现。其次，有些语言（如 C#）的类继承只允许单继承，但是接口允许多继承，一个子接口可以有多个父接口。

接口包含的成员有方法、属性、索引指示器和事件。由于接口允许多继承，在可能发生二义性的地方可以采用全权名来避免。

可以使用类来实现接口。在类中定位接口成员的实现称之为接口映射。类必须为接口的所有成员提供具体的实现，包括接口中显式声明的成员，以及接口从父类接口中继承而来的成员。同样，在对接口的实现过程中可以采用显式接口成员执行体来避免产生二义性。派生类可以对基类已经实现的接口进行重实现。抽象类也可以实现接口，但接口成员必须映射到抽象类的抽象成员。抽象类的派生类如果是非抽象类，则必须通过方法重载来实现接口成员。

3.6.2 功能接口层的实现

功能接口层的功能是定义提供给应用层的接口，按照数据类型及其在具体应用的特点划分，主要包括音频功能接口，视频功能接口，文本功能接口，图像功能接口和控制功能接口。其中音视频、文本、图像功能接口都含有实时数据采集，实时数据加工，实时数据重定向功能；控制功能接口包含实时数据加工，实时数据重定向功能。

①实时数据采集：采集的数据源包括各种外设，如键盘、显示器、摄像头，声卡等。②实时数据加工：加工过程可以是压缩/解压缩数据，以减少网络带宽占用；也可以对数据加密/解密，以防备网络上的攻击。③实时数据重定向：重定向是指数据（音频、视频、图像、文本、文件、控制信息等）可以有多种流向。控制信息指实时控制软件、硬件资源的相关参数，通过发送控制信息可以控制本地以及其它计算机的软件、硬件资源，比如锁定屏幕，远程关机等。控制信息以外的数据可以交给本地外设或远程外设处理，比如音视频数据的回放，图像或文本信息的显示以及文件的异地保存。

3.7 平台应用实例 —— 三层结构的远程多媒体监控系统

实时多媒体监控系统的系统结构如图 3-9 所示。整个系统分为客户端、Web 服务器和第三层服务器（包括监控服务器和数据库服务器）三层[56]，其中客户端与监控服务器都可以是一个或多个。客户端、Web 服务器和监控服务器都运行在.NET 平台之上，由于微软将在 Windows 操作系统中免费部署.NET 平台，所以.NET 平台可以划归操作系统的一部分。

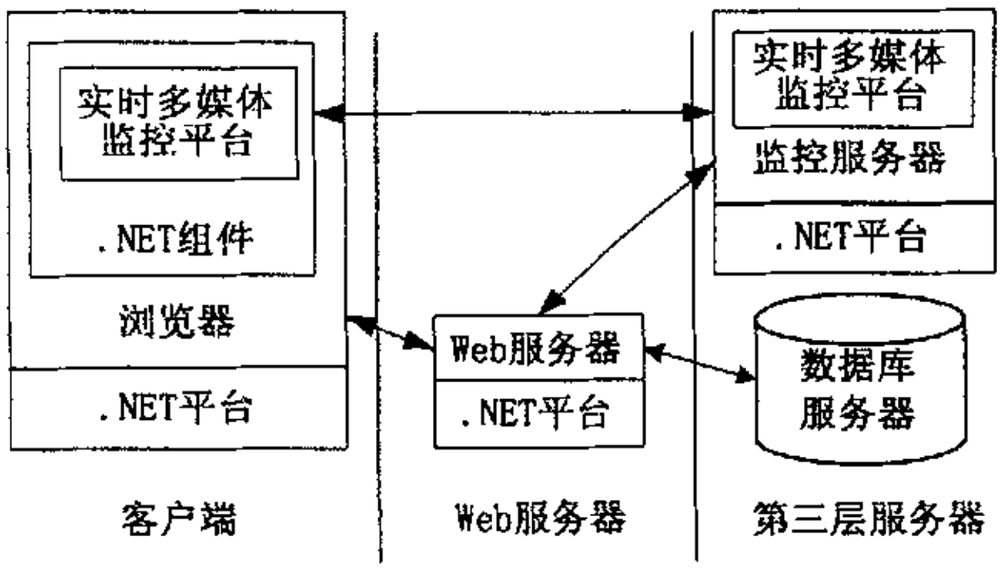

图3-9 三层结构的实时多媒体监控系统 结构图

客户端只需要有浏览器，这就减轻了客户端的负担，减少了客户端程序更新与维护的费用，而将业务逻辑尽可能地转移到后台服务器上。但并不是所有的功能都可以依靠后台服务器，在网络多媒体系统中，必须要在客户端运行一定的功能，与后台服务器相配合完成音视频等多媒体数据的端到端传输。例如客户端需要管理音视频数据的缓冲区，对多媒体数据进行编解码，当客户端与监控服务器进行音视频交流时，客户端要响应服务器呼叫请求，这时客户端也在充当服务器的角色。所以系统采用客户端浏览器嵌入.NET 组件的方式，将必要的功能动态下载到客户端运行。这里的.NET 组件以图形化界面显示，运行于实时多媒体监控平台之上。实时多媒体监控平台负责通用的底层功能，如网络通信、音视频采集和对控制信令进行解释执行等。

位于中间层的 Web 服务器起到纽带作用，为客户端提供 Web 页面及.NET 组件下载，并以事件形式响应客户端的请求。Web 服务器通过 ADO.NET 数据库引擎与数据库服务器交互，只将结果返回给客户端。并且能够搜集各个监控服务器的属性信息和运行状态并提供给管理者。

监控服务器分为软件形式的实时多媒体监控平台和硬件形式的摄像头、

话筒等外设组成，能够采集音视频并进行压缩，以组播形式发布给客户端。数据库用于存储用户身份信息和监控服务器的运行日志。

4 结论

本文论述了基于.NET 的分层结构模型和平台中多媒体数据处理的关键技术及具体实现方法。为了保证多媒体数据传输的实时性，采用国际上流行的多媒体会话控制协议 SIP 和多媒体实时传输协议 RTP；在对多媒体数据进行处理时，考虑到多媒体通信的特点和实时监控的要求，采用了国际上通用的压缩标准，如 G.729a、H.263 对音视频数据进行压缩；对于可靠性要求高的数据传输，引入.NET 中的 Remoting 中间件技术，提高了传输性能，并极大地简化了实现过程。

微软已经提出下一代 Windows 操作系统（Longhorn），.NET 的功能将得到增强，可以肯定.NET 技术将是未来开发平台的趋势。基于.NET 的网络多媒体实时监控平台无论从技术上还是应用上都具有广泛的拓展空间，它不仅能用于多网络媒体实时监控，还可以应用于视频会议、多媒体网络教室和信息服务等众多领域。

本课题的实现环境是局域网，但从设计和实现角度讲，完全可以移植到广域网中。无论是传输协议，媒体压缩标准，还是实现平台都适合广域网环境。

课题的不足之处主要是 RTP/RTCP 及 SIP 协议的实现不够完善，如多线程处理资源占用过多，状态机的设计需要进一步改进，没有考虑传输过程中的安全问题等等。另外，.NET 自身处于不断更新之中，目前还存有许多 Bug，如.NET Remoting 中间件还不够稳定，性能还有待提高。资源预留协议(RSVP)将更加普及，所以协议中将增加 RSVP 功能，以便提供更好的 QoS 保证。未来的主要工作除了对以上进行改进外，还要使之更适合广域网的环境，以及对平台进行系统化的测试。

参考文献

[1] 王兴伟等，多媒体通信：需求与对策，小型微型计算机系统，1997，18(3)： 1-7。

[2] 杜润秋等，一种基于 QoS 的远程监控系统模型，计算机科学，2003，30(2)：89-91。

[3] 康贝等，实时语音数据在 IP 网络中传输的 QoS 保障，航空计算技术，2003，33（1）：106-109。

[4] 周强等，下一代的互联网协议——IPv6，中国数据通信，2003，5（9）：22-25。

[5] 贾凤军等，NetMeeting 在校园办公网的应用，计算机与网络，2003，(19)：40-40。

[6] PictureTel 公司， PictureTel 音频技术：将音频标准提高到新的水平， 世界网络与多媒体， 2000，8（7）：72-73。

[7] 王国平等，NetShow 视频点播系统的成功实现，现代图书情报技术，2001（4）：57-58。

[8] Macrus Goncalves 等，IP 多路广播技术与应用，电子工业出版社，2001.1。

[9] Peter Thorsteinson 等，.NET 构架技术与 Visual C++编程，清华大学出版社，2003.8。

[10] 陈卫珊等，基于 G.729A 的 IP 电话系统设计，计算机工程，2001，27(9)：118-119。

[11] 周送军等，窄带多媒体通信的关键技术，太原理工大学学报，2003，34（5）：591-593。

[12] Don Box， Chris Sells. Essential .NET : the common language runtime[M].Addison Wesley，2002。

[13] 潘爱民，COM 原理与应用，清华大学出版社，1999。

[14] 邹永刚等，基于 Windows DNA 体系结构的 COM+应用编程，计算机应用与软件，2003，20（10)：80-81。

[15] 童学红等，对 Windows2000 可执行文件格式的分析及其应用，计算机应用研究，2003，20（3)：15-19。

[16] 邹筱梅等，XML 技术与应用综述，教育信息化，2003（5)：69-71。

[17] 南湘浩等，网络安全技术概论，计算机安全，2003（30)：76-76。

[18] 施荣华等，基于数字签名的安全认证存取控制方案，软件学报，2002，13（5)：1003-1008。

[19] 李俭兵等，SOAP 技术及其应用，计算机科学，2003，30（4)：111-112。

[20] 徐景平等，ISDN 网络体系及发展潜力，辽宁工学院学报，2002，22(3) ：39-41。

[21] 落红卫等，xDSL 接入技术现状与发展，电信网技术，2003（8）：70-73。

[22] 林瑶等译，用 TCP/IP 进行网际互联第一卷：原理、协议与结构，电子工业出版社，2001。

[23] 马林等，ICMP 协议在网络层的地位及应用，计算机应用研究，2000，17（7）：53-55。

[24] 李军等，多媒体组播协议分析及其实现，计算机工程与设计，2003，24(3)：40-42。

[25] 李海芳等，Intranet 上实时多媒体混播技术研究，计算机工程与应用，2003，39（20）：133-136。

[26] 朱晓刚等，基于 WinSock2 的 Multicast 网络应用的实现，计算机应用研究，1999，16（1）：30-32。

[27] 韩永魁等，组播和隧道相结合的网络频道系统，计算机工程，2002，28（Z1）：198-201。

[28] H.Schulzrinne. RTP： A Transport Protocol for Real-Time Applications，RFC1889，1996.1。

[29] 葛澍等，IP 网络电话的动态 QoS 管理方法，计算机工程与应用，2003，39（14）：185-186。

[30] 李念强等，分布式应用集成模型——.NET 远程框架，计算机工程，2002，28(3)：267-269。

[31] 马华东等，多媒体计算机技术原理，清华大学出版社，1999。

[32] 钱志远等，三类音频压缩技术运用，实用影音技术，2000（1）：48-49。

[33] 罗宏等，MPEG 标准中的音频技术，广东通信技术，2002，22（5）：6-8。

[34] 路林吉等，用于数字监控的图像压缩技术，电子技术，2001，28（8）：56-59。

[35] 张宇杰等，快速以太校园网结构设计研究，测试技术学报，2002，16(2)：152-154。

[36] Jeff Prosise. Programming Microsoft .NET[M].Microsoft Press，2002。

[37] Richard Grimes. Developing Applications with Visual Studio.NET[M].Addison Wesley，2002。

[38] Ted Fasion，Visual C# 基于组件的开发，清华大学出版社，2003.4。

[39] 陈琨等，基于.NET Remoting 利用软件方法实现网络教学的探索，计算机应用，2003，23（8）：130-132。

[40] 任延珍等，基于 RTP/RTCP 协议的实时数据传输与同步控制策略，计算机工程与应用，2003，39（10）：144-147。

[41] 黄文涛等，网络实时视频传输研究及实现，计算机应用，2003，23(2)：100-101。

[42] M. Handley. SIP： Session Initiation Protocol，RFC2543，1999.3。

[43] M. Handley. SDP： Session Description Protocol，RFC2327，1998.4。

[44] 李长河等，多媒体通信协议H.324，H.323及SIP的分析研究，微机发展，2003，13（a02）：106-109。

[45] 代建华等，基于SIP的Internet电话，微电子学与计算机，2000，17(4)：23-27。

[46] 王红燚等，SIP协议栈的实现与应用，北京邮电大学学报，2000，23(4)：74-78。

[47] 柴乔林，SIP协议在开放的VOIP模型中的实现，计算机工程，2002，28（S1）：

222-226。

[48] 鱼滨等，用 UML 构建基于组件的分布式应用系统，计算机工程与应用，2003，39（17）：115-116。

[49] Alan Johnston. SIP telephony call flow examples. IETF Internet-Draft, 2000。

[50] 王群生等，WDM/DirectShow 视频捕捉架构解析，电视技术，2002（10）：6-9。

[51] 蔡波等，基于 COM 规范的实时音频特征提取技术及实现，计算机工程 ，2003，29（11）：116-118。

[52] 蔡龙华等，基于 DirectShow 技术的视频捕获，计算机与现代化，2003(8)：81-84。

[53] Andrew Krowczyk，.NET 网络高级编程， 清华大学出版社，2003.3。

[54] 周存杰等，Visual C#.NET 网络核心编成，清华大学出版社，2002.11。

[55] 沉舟等，C#教程，北京希望电子出版社，2001。

[56] 查卫翔等，ActiveX 控件在基于 B/S 结构的远程监控中的应用，北方交通大学学报，2002，26（1）：58-62。

致　谢

在整个课题的设计、开发及论文的撰写过程中，张华忠教授给予我悉心的指导和不倦教诲，使我顺利地完成了课题研究。在三年的学习期间，张老师尽力创造各种机会锻炼我的动手能力、表达能力和独立解决问题的能力。另外，张老师在思想上和工作态度上对我也有很大的影响，不仅使我开阔了视野，而且养成了认真细致的科学态度，在此向张老师表示深深的谢意。

衷心感谢柴乔林教授，在科研工作和论文撰写过程中给我的真诚指导和无私帮助，我的论文能顺利完成，离不开您的热情支持和无私奉献，在此，我向您致以深深的敬意。同时要感谢刘菲、慈建伟、张志、刘成志和张维勇同学给予我的指导和帮助。

最后，还要感谢我的父母和家人多年来多我的关心、理解、教导和支持。他们教育我要踏踏实实地做人，勤奋努力地学习工作，这种朴实的思想一直促使我在人生的道路上不断追求进步，奋发向上。他们是我能信心百倍地去克服前进道路上的重重困难，不断迎接新的挑战的力量源泉。

攻读硕士学位期间发表的论文

1. 侯海文、张华忠、刘菲，基于.NET 的实时多媒体监控平台的研究，计算机应用，2004.3
2. 侯海文、刘菲、张华忠，基于移动代理的分布式并行计算中间件设计与实现，计算机应用，2004.5
3. 刘菲，张华忠，侯海文，基于移动 agent 的分布式视频点播系统的研究，计算机工程与应用，2004.12

学位论文评阅及答辩情况表

论文评阅人	姓　名	专业技术职务	所在单位	对论文总体评价※
	滨磊	副教授	山东大学	良
	刘乃震	教授	山东师范大学	优

答辩委员会成员		姓　名	专业技术职务	所在单位	备　注
	主席	刘体玉	教授	山东师范大学	
	委员	徐秋亮	教授	山东大学	
		李新井	教授	山东大学	
		史清华	副教授	山东大学	
		肖子水	副教授	山东大学	

答辩委员会对论文的总体评价※	B	答辩秘书	林幸波	答辩日期	2004.5.24
备注					

※ 优秀为“A”；良好为“B”；合格为“C”；不合格为“D”。