TCP/ IP 在实时通信中的应用*

曹 伟**

摘要 分析了在 Windows95 环境下,利用 TCP/ IP 协议实现网络实时通信的方法。介绍了 TCP/ IP 与 UDP 协议的区别以及两者在网络通信中的编程方法。

关键词 以太网络 SOCKET TCP/ IP UDP

引 言

计算机网络最基本的形式是两台计算机相互连接进行通信。随着计算机网络技术的发展,计算机网络已不局限为几台或几十台计算机互连,而是目前所面临的全球性网络连接即 Internet。随着 Internet 的普及,Windows 环境下的网络编程显得尤其重要。人们在 Windows 环境下通过 Winsocket 网络接口和 TCP/ IP 协议编制网络通信程序,建立网络连接,从而将自己的计算机连接到 Internet 网上来实现网络资源共享。本文首先介绍 TCP/ IP 协议组,然后就 Winsocket 网络接口和 TCP/ IP 协议这两方面进行分析。

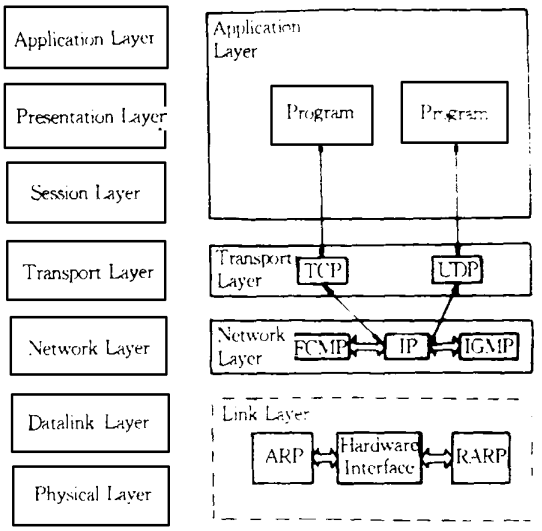
1 TCP/ IP 协议组

1.1 TCP/ IP 协议组与 ISO/ OSI 对应关系

目前我们普遍使用的 TCP/ IP 协议如表 1。TCP/ IP 协议组与 ISO/ OSI 模型所对应的关系如图 1。

表 1

协议	目的
IP	互联网协议是在主机间传递数据的网络层协议
TCP	传输控制协议是在应用程序之间传递数据的传输层协议
UDP	用户数据报协议是另一个传输层协议。UDP 也在应用程序间传递数据,但是 UDP 没有 TCP 那样复杂,也没有它那么可靠
ICMP	互联网控制报文协议携带网络错误信息,并报告网络软件需要注意的其他情况



ARP:地址解析协议 RARP:反向地址解析协议 IGMP:互联网管理协议

图 1 ISO/ OSI 模型与协议组

* 本文于 1998 年 9 月 23 日收到。
** 南京船舶雷达研究所。

1.2　以太网络上的 TCP 数据封装

为了通过网络传输数据,就必须将数据从应用程序传到协议栈的一个协议中。当此协议对数据处理完后,将数据传给栈中的下一个协议。当数据通过协议栈中的每层时,协议模块(网络软件)为栈中的下一层进行数据封装。因此封装是一个按栈中较低层协议要求的格式保存数据的过程。当数据通过协议栈传递时,每层都建立在它前面层的封装上。图 2 显示了在以太网上使用传输控制协议(TCP)时网络软件如何对数据进行封装。

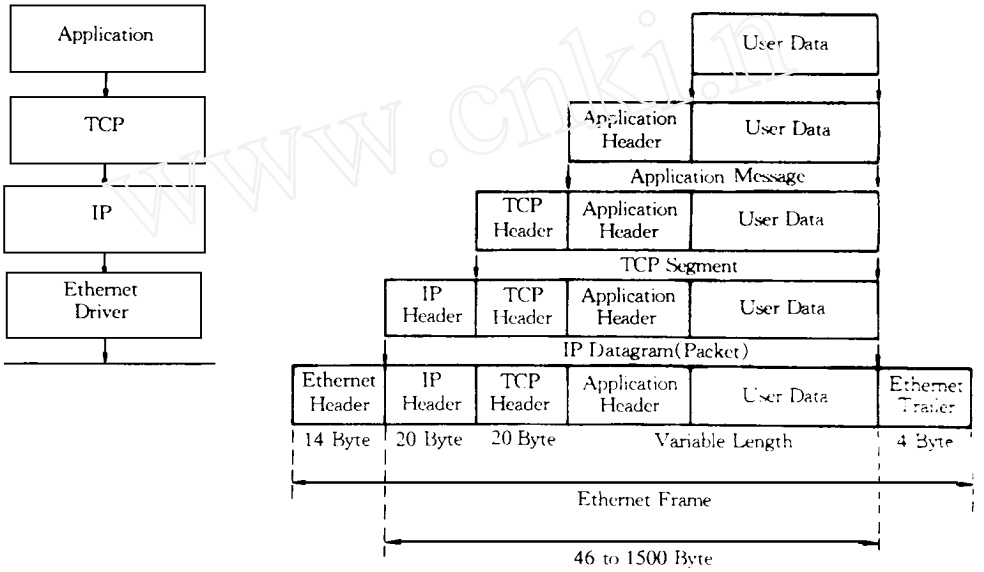


图 2　在以太网上数据封装

从图 2 中可以看到,应用程序模块在应用报文中对用户数据进行封装,程序设计决定程序是否使用应用报文。在许多情况下,程序使用像 TCP 这样的网络协议格式化用户数据,TCP 模块把应用数据格式化为 TCP 段,TCP 段包括应用数据和协议要求的 TCP 头信息,应用数据包括应用头和用户数据。

1.3　TCP 传输控制协议和 UDP 用户数据报协议

一般来说,为了和 Internet 通信,应用程序必须和 TCP/ IP 传输层交换数据,传输层包括两个传输协议:传输控制协议(TCP)和用户数据报协议(UDP)。在编写 Internet 应用程序时,必须按照这些协议中的某一个进行编程。

传输控制协议 TCP 使用可靠的字节流发送和接收数据,是一种面向连接的协议,并为网络通信提供虚电路。用户数据报协议 UDP 发送和接收数据,是一个不可靠、无连接的协议。从表面看,TCP 提供可靠的、面向连接的数据传输服务,所以也就决定了 TCP 比 UDP 要复杂得多。TCP 确保传送正确,并保证目的应用程序按正确顺序接收数据,因此对报文的发送均存一个回答报文。如果报文出错,则请求重发此报文,这在实时系统中均会造成一定的时间开销,从而影响系统的实时性。而 UDP 协议不保证数据正确传送,它通过端口向目的地址发送报文而不需要回应。目前对高可靠、低误码率的网络来说,在传输过程中即使

偶然会产生错误报文,也很快会被下一帧正确报文所替代,因此采用 UDP 协议在实时系统中无疑是一个最佳选择。从图 2 可以知道 TCP 的数据封装形式,但 UDP 数据结构以及通过 UDP 的数据是如何流动的呢? UDP 使用数据报进行传输,不可靠、无连接,能将数据寻址到一台主机的多个目的应用程序。正常情况下,网络给每个目的应用程序分配一个协议端口,并利用 UDP 使用数据报进行数据传送。UDP 数据报包含 UDP 头。UDP 头的结构如图 3 所示,包括四个域源端口、目的端口、报文长度和校验和。

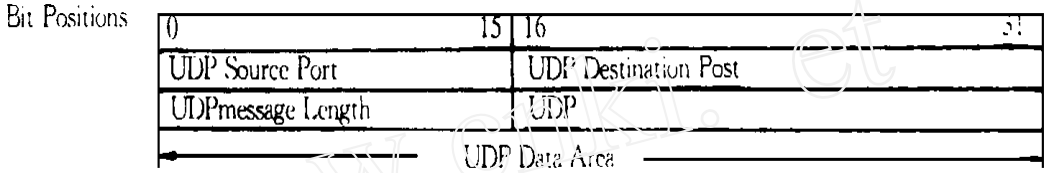


图 3 UDP 数据板的结构

UDP 模块将接收到的数据按照端口号进行分配。图 4 显示了数据怎样从网络层通过 UDP 模块到达应用程序。

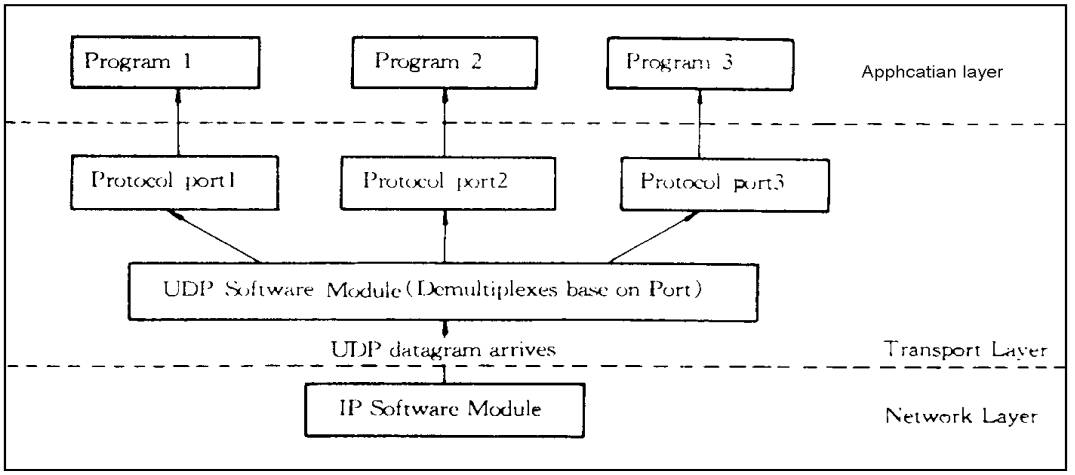


图 4 UDP 模块的数据流动

2 应用 Socket 接口实现 TCP/ IP 协议组的网络编程

Socket 是网络通信的基本操作单元,它提供不同主机间进程双向通信的端点。Socket 编程接口是应用层协议的实现基础。开发 Socket 的目的是隐蔽网络底层复杂的结构与协议,使编程人员能够简单、抽象地对网络进行操作。利用它可以构造任意的跨操作系统和跨网络协议的分布式处理系统。随着 Internet 在全球范围内的广泛使用,Socket 已渐渐成为网络编程的通用接口。

2.1 Windows Socket

Microsoft 公司开发的 Windows Socket API 建立了 Windows 环境下网络间的编程接口,

它以 California 大学 BSD Socket 的编程模块为基础,其中包括具有 BSD 风格的 30 个左右的函数和专门为 Windows 系统扩展的近 20 个函数。

Windows Socket API 有 16 位和 32 位两种版本,都是以动态链接库的形式提供的。其中 Winsock. dll 为 16 位版本,Wsock. dll 为 32 位版本,二者均含有上述两类函数,并且都是通过 Windows 系统中的 message 消息来驱动的。16 位版本是单进程的,32 位版本可在多线程中使用,并提供了线程安全性功能。

根据传输的数据类型不同,又可将 Windows Socket 分为 TCP 和 UDP 两类。二者对 Socket 的使用分别见图 5 和图 6。

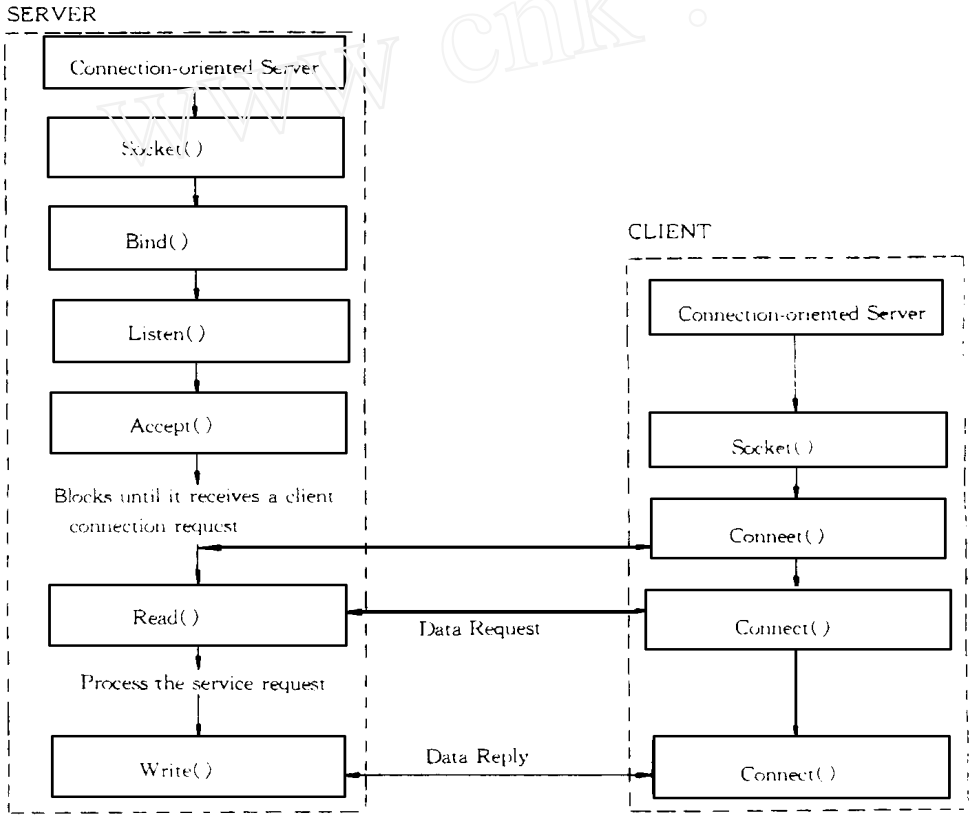


图 5　面向连接协议时 SOCKET 使用

这两种类型的 Windows Socket 都可实现双向通信,主要用于以下三种场合: ①Client/ Server 模型、②对等网络和 ③远端过程调用。

2. 2　Socket 编程

VC + + 的 MFC 提供了两种类(CAsynSockets 和 Csocket)来描述 Windows Sockets,其中 CAsynSockets 类封装了 Windows Sockets API,并将与 Socket 有关的 Windows 消息转换为回调函数。CAsynSockets 类则更加面向底层操作,因而使用起来也就更加灵活些,但要求编程人员应该熟知网络,水平也应该更高些。

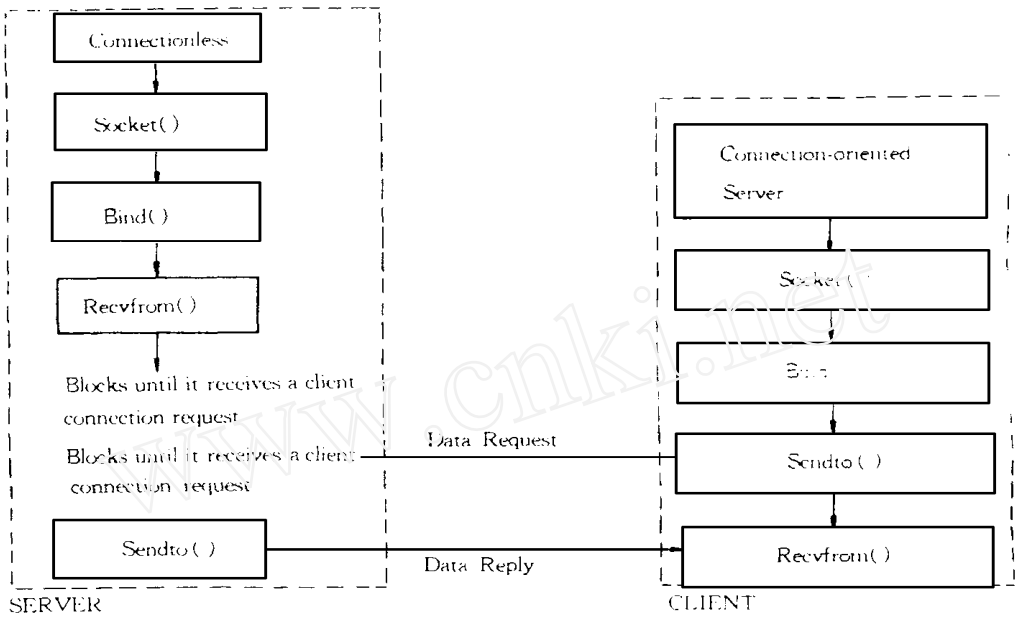


图 6 无连接协议时 SOCKET 使用

Csocket 类是 CAsynSockets 类的导出类,通过 MFC 类的 Carchive 类的对象提供了更高层次的抽象。它封装了 Socket 实现中的许多细节,并将 Sockets 与文档(Archive)相结合,使用它与使用 MFC 中的文档串行化协议相类似。对于 CSocket 编程,目前在有关计算机技术杂志上有详细的叙述,本文只大概讨论一下 Csocket 编程的主要步骤,而更详细论述了 CAsynSockets 类的编程过程。

2.2.1 TCP 下 Csocket 编程的主要步骤

(1) 构造 Csocket 对象。

(2) 使用该对象构造基本的 Socket 句柄。对于 Csocket 客户端对象来说,应该使用缺省参数 Create;对于 Csocket 服务器对象来说,需在 Create 过程中指明一个端口。

(3) 建立客户端 Csocket,调用 Connect 建立与服务器端 Socket 的连接。服务器端调用 Listen 监听,并在收到客户端请求后调用 Accept。

(4) 构造 CSocketFile 对象,并使 Csocket 对象与之关联。

(5) 构造 CArchive 对象,用于接收或发送数据。

(6) 使用 CArchive 对象来进行 Client 端与 Server 端的 Socket 通信。

(7) 删除 Archive、SocketFile、Socket 对象。

2.2.2 UDP 下 Socket 编程

由于 UDP 协议缺少了像 TCP 协议使用 CAsynSocket 类导出的 Csocket 类,因此缺少了 Csocket 实现中许多细节,故对 UDP 编程就应该用 CAsynSocket 类,从面向底层操作开始实现。其主要步骤为:

(1) 构造 CAsynSocket 对象。

(2) 对 Socket 进行初始化:使用 MFC 中的创建 AppWizard(exe),选中 Windows Sockets

复选框后,AppWizard 会自动创建 AfxSocketInit() 初始化模块来对 Socket 进行初始化。

(3) 用 CAsynSocket::Create 创建一个 Socket。如果应用程序需要,可创建多个 Socket,在 UDP 协议中分别对应多个捆绑的端口,其模块原型如下:

Bool Create (UINT nSocketPort,int nSocketType,long lEvent,LPCTSTR lpszSocketAddress)

其中,nSocketPort 为 UDP 所绑定的端口号;NSocketType 为协议类型,UDP 协议为 SOCK_DGRAM;lEvent 为应用程序为网络事件所设置的屏蔽位;LpszSocketAddress 为本节点的 IP 地址。

(4) 以创建的 Socket 调用 CAsynSocket:SendTo 模块,将所要发送的报文上网发送,SendTo 原型为

int SendTo(Const Void *lpBuf,int nBuflen,UINT nHostPort,LPCTSTR lpszHostaddress,int nFlags)

其中,lpBuf 为所要发送报文包的缓冲区;nBuflen 为缓冲区的长度,注意此处以字节为单位;nHostPort 为所送置的端口号;lpszHostAddress 为所送目的节点的 IP 地址;nFlags 为调用的产生方法,缺省为 0。

(5) 通过消息映射函数 CAsynSocket::OnReceive 将端口产生的网上送至报文产生一个 OnReceive 消息并通过 CAsynSocket::ReceiveFrom 对网上报文进行接收。ReceiveFrom 的原型为

int ReceiveFrom (Void *lpBuf,int nBufLen,CString rSocketAddress,UINT rSocketPort,int nFlags)

其中,lpBuf 为接收报文缓冲区;nBufLen 为接收报文缓冲区的长度;rSocketAddress 为发方的 IP 地址;rSocketPort 为发方的端口号;nFlags 指明调用的产生方法,如参数使用符号常数 MSG_OOB 作为标志值,表示可以从协议端口请求带外数据。带外数据是程序必须立即处理的紧急数据。如存在带外数据,函数立即将紧急数据返回给程序。如果没有带外数据,函数返回常数错误值 EINVAL。若使用符号常数 MSGPEEK 作为标志值,表示可以对传输层输入队列中的数据进行分析,如果不需使用这两个标志,可设为缺省值 0。

(6) 删除 Socket 对象。

通过上述过程,就可设计基于 Windows Socket 的应用程序通过 Internet 来传输数据。由于 Inteinet 的传输速率高达几十兆 bps 到几百兆 bps,所以不仅可以用上述方法传输一般的文字、图形、声音等数据,而且还可以用来传输实时的图像数据。