通过红外线设备进行 TCP/IP 互连

通过红外线设备实现代理上网

彭韬
电子科技大学 信息与通信工程学院 计算机通信专业

摘要：在现代经济社会中，人们经常使用便携式设备来快速获取信息。通过便携机上可以安装的标准红外线设备互连，人们可以感受到获取信息的种种快捷与方便。相对于笨重的有线互连设备，红外连接使人们体验网上冲浪，进行户外工作都变得更加轻松愉快。

在本文中，将对关于如何通过红外线设备在红外协议（IrDA）上实现对互联网高层协议的代理应用进行探讨。

关键词：IrDA，套接字，代理，超文本，文件传输。

引　言

全球信息的膨胀，信息高速公路的建设加速了信息的流通。人们获取信息的手段也在不断丰富，不断变化，从互传信息到互连共享信息，从有线连接到无线连接。现在，人们可以随时随地地通过各种渠道进行通信特别是通过互联网络进行信息交流！

红外通信在人们日常生活中扮演着重要的角色：从电视机、VCD 遥控器，到电梯、门禁系统，乃至便携式电脑，都可以见到红外通信的身影。由于其价格低廉，使用方便，解决了有线连接的许多不便，因而受到了家电设备厂商、电脑外围设备商、以及通信设备厂商的高度重视。

社会在飞速发展，工作节奏不断加快。由于各种需要，人们常常出门在外，于是便携式设备就成了获取信息的主要工具。通过便携机上可以安装的标准红外线设备互连，人们可以更直接地感受到获取信息的种种快捷与方便。

对于近距离互连通信来说，有有线和无线连接之分。对于有线，可采用各种电缆的机械式连接，这样设备简单，造价小，响应速度快，传输速度快，并且对现有应用兼容性也好。然而机械连接有其固有的缺点，特别是对于移动用户，需要附加设备，携带不便，设备安装空间也有限制，对设备安装放置还有要求（如服务设备的空间、结构设计）。对于无线来说，相对于有线设备其机械设备少，无接触，机械故障率小，而且移动用户可采用标准设备，无需另购附加设备。由于采用标准红外协议，接口升级容易，在应用上操作自动化程度较高。当然，由于红外设备的自身特点，接口响应速度较慢，连接传输速度不会很高（不过新的接口规范定义了更高的连接传输速率，所以速度并不是问题）。设备在采用红外线的情况下，容易受外界环境影响，在不同光照、不同温度、不同的客户设备放置方式下，传输效果都不相同。但是总的来说，从对客户端的操作上看，红外线设备具有硬件轻便，配置标准化，安装较为简单，而且操作的自动化程度较高等硬件优点，加上软件的国际支持化程度高，使之有较大发展空间。串口等有线通讯对移动用户的负担比无线通讯要大。

在本文中，对关于如何通过红外线设备在 IrDA 协议上实现对互联网高层协议的代理应用进行了探讨。

一． 关于互联网应用层一些主要服务协议以及进行代理的讨论

OSI 参考模型及应用层的位置

ISO 组织将网络按其功能划分为 7 个功能层，每层都完成一特定功能。图 1 所示为 OSI 参考模型。

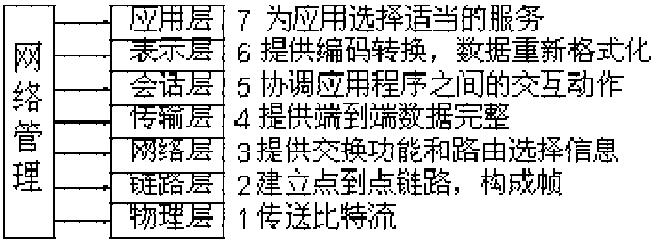


图 1

应用层（Application Layer）

应用层最贴近用户。它与其他层不同，并非面向 OSI 层，而是向 OSI 模型之外的应用提供服务。这些 OSI 之外的应用可能是电子表格、字处理、银行终端之类。

应用层考察通讯对方的能力，使双方的应用程序同步，协调纠错进程和数据完整性的控制。同时，应用层也判断通讯所需的资源是否存在。

1. Http 服务的代理讨论

Internet 的基本协议是 TCP/IP 协议，目前广泛采用的 FTP、Archie Gopher 等是建立在 TCP/IP 协议之上的应用层协议，不同的协议对应着不同的应用。

WWW 服务器使用的主要协议是 HTTP 协议。由于 HTTP 协议支持的服务不限于 WWW，还可以是其它服务，因而 HTTP 协议允许用户在统一的界面下，采用不同的协议访问不同的服务，如 FTP、Archie、SMTP、NNTP 等。另外，HTTP 协议还可用于名字服务器和分布式对象管理。

基于 HTTP 协议的客户/服务器模式的信息交换分为四个过程：建立连接、发送请求信息、发送响应信息、关闭连接。

在 WWW 中，“客户”与“服务器”是一个相对的概念，只存在于一个特定的连接期间，即在某个连接中的客户在另一个连接中可能作为服务器。WWW 服务器运行时，一直在 TCP80 端口(WWW 的缺省端口)监听，等待连接的出现。

这四个过程的协议内部情况请见附件：**HTTP 服务分析**。

对于代理应用来说，采用多线程进行程序界面处理和代理端口监听处理。一旦接收到客户连结，就新开一个代理线程建立起这个与客户端的连接。当代理线程接收到客户端发送来的请求时，需要分析出响应请求的服务器端主机地址。然后根据这个地址信息（从代理机器上）连接到服务器。接着就需要将客户端的请求信息转发到服务器端。在成功发送了请求信息后，代理线程就只需要等待服务器发送回来的响应了，响应一到达，就不做任何修改地转发到客户端。在服务器发送响应结束时，代理线程关闭该连结。

下面就这四个过程的代理分别作一讨论（以后在此基础上加入红外线通道）。

1．1 建立客户连结与服务器连接：

客户端通过指定的代理端口进行 HTTP 操作，代理程序一启动便开始监听该代理端口。当客户程序请求 Http 服务时，将本该发送到 Http 服务器的请求信息先发送到该代理端口。代理应用中监听端口（Listen）的线程接收到这个来自客户端的连结，于是在保存客户端主机信息后，就启动一新的代理线程来处理客户端发来的 Http 请求信息。这样与客户端的连接就建立起来了。

这里有个客户端 HTTP 请求的例子：

GET http://www.tyfo.com/ HTTP/1.0
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: www.tyfo.com
Proxy-Connection: Keep-Alive
Pragma: no-cache

分析请求的过程中要确定连接服务器的目标端口，这个可在 Host: 字段下分析出来，该请求为默认端口 80。如果不是默认端口 80，该字段就在域名后面跟上连接的目标端口。

现在代理应用取得了足够的服务器信息，接着就是对服务器的连结了。在通过请求中的服务器域名获取了服务器 IP 地址后，代理线程就开始连结服务器，建立起与指定的服务器端口的连结。

1． 2 发送客户端请求信息到服务器

在两边的连结都建立起来以后，代理应用处理了客户端的请求信息。当然，必要时还要修改原始的请求信息以便服务器端处理。（比如去掉命令中的主机域名）

所有的参数都在实体后面给出，通过请求信息一次发送到服务器，然后就等待服务器发回超文本信息响应了。到目前为止，所有连结都还保持着。

1． 3 发送服务器的响应信息到客户器

在 Http 中，这个过程不是很复杂。代理将这些响应可以不加处理地全部发送给客户端，使客户端感觉不到代理的存在。

代理线程将通过服务器端连结接收到的响应信息缓存后再通过客户端连结发送到客户端，直到服务器端结束发送响应（收到字节数为零）。这个问题在一个函数内就可以解决。

1． 4 关闭两边的连结

当服务器结束了响应信息的发送后，也就是服务器端连结接收到的响应信息字节数为零，代理线程就可以正常关闭两边的连结了。

2． FTP 服务的代理讨论

FTP 的目标是提高文件的共享性，提供非直接使用远程计算机，使存储介质对用户透明和可靠高效地传送数据。虽然我们也可以手工使用它，但是它的主要作用是供程序使用的。

对于 FTP 及 Telnet 等其他服务应用，服务器和客户器是采用问答协商方式进行信息交换的。比如 FTP，它需要一个控制通道，还需要一个数据通道，在问答协商下决定数据连结端口。所以这就使代理程序的主要工作成了对命令及响应的分析，以便连结到动态的端口及作出自身对客户端的响应。

FTP 协议实现情况请见附件:**FTP 服务分析。**

在建立客户端连结上，FTP服务代理和Http服务代理没有多大区别。只是由于通过问答方式通信，所以服务器信息是在客户端的命令中分析出来的。这就对客户端代理的设置有一定的要求，比如要设置成 USER username@host:port 的命令格式。

代理程序仍然是开启单独的代理线程来处理 FTP 服务的。在控制端口上，代理线程需要分析 FTP 的几个主要命令如 USER 用户登陆,PASS 密码,PSAV 被动模式,PORT 协商数据端口 等，以及 FTP 的几个主要命令响应：如 150 数据传输准备好，227 被动模式，421 服务器拒绝服务，500 命令错误 等等

一旦建立了客户端连结，代理线程就开始对客户端发送欢迎信息，等待客户端的命令：

首先应该是 User 命令，代理线程能够从中分析出服务器域名地址，服务器的 FTP 控制端口（默认为 21）。

接着代理线程根据服务器信息与 FTP 服务器在控制端口上建立控制连结，当服务器表明为不忙时，对其发送经过修改的 User 命令（去掉了主机信息）。

和 Http 一样，在服务器发回响应信息后，代理应该将其转发到客户端，但是在 FTP 的情况下，必须对标准的 FTP 响应代码进行分析。一般服务器在 User 命令通过后，紧接着就要求客户端输入密码。

成功登陆后，作为一般情况，客户是要求下载一个文件。在一系列查询文件信息的命令及响应过后，客户端发送一 Port 命令，开始与服务器协商数据端口。在非 PSAV（被动模式）下，代理线程在对 Port 命令分析得到数据端口后，就要开启一个数据传输线程。

在数据传输线程中，代理应用与 Http 一样同时担任了客户端和服务器端的角色。数据传输代理线程也是在不做任何分析修改的情况下转发来自服务器端的信息流到客户端。在服务器信息流发送结束后，控制端口会给出结束传输的响应，这时候代理线程就可以正常关闭所有连结了。

二．关于（IrDA）红外通讯应用开发的讨论

1．红外通信有关协议

为了实现各类产品的互连，国际上成立了一个红外数据通信协会（IrDA: Infrared Data Association）来协调各方面的工作，并制定了一系列的标准。这一系列标准的出台不仅规范了产品的设计，同时还进一步扩展了红外通信的功能和应用领域（如：数据传输速率即可高达到 4Mb/s，而且进一步的建议中还允许高达 16Mb/s 的速率）。这些标准包括：物理层协议、数据链路层协议、数据链路控制协议以及其它高层的如：流控制协议、串行通信协议、类 HTTP 的实体交换协议、图象交换协议、局域网接入协议等等。

1．1 红外通信主要协议

基础协议：

- **PHY** (Physical Signaling Layer) 物理层协议
- **IrLAP** (Link Access Protocol) 描述了红外通信过程中数据链路层上的有关功能、特性与协议。
- **IrLMP** (Link Management Protocol and Information Access Service (IAS)) 数据链路管理层协议，描述链路的发现、复用与管理。

高层协议：

- **IrCOMM** 描述了基于 IrLMP 层和 IrLAP 层上的串、并口仿真协议。该协议使得此前使用串、并口通信的应用程序可以直接移植到利用红外设备进行通信的应用场合。
- **Tiny tp** 红外通信中的传输协议
- **IrLAN** 红外通信用于局域网互连时的有关协议。
- **IrMC** 红外通信中与应用层相关的协议。

2．2 在 Windows 操作系统下的红外通信应用

IrDA 与 Windows 的 Sockets 的充分结合为应用程序的设计者提供了简单又非常强大的 Win32 API，这些 API 提供了多重、全纠错的流式应用。串行和并行端口操作只是另外一种具备普通用户 API 的点到点通信技术。IrDA 定义了非常丰富的功能，这些功能很多是串行和并行电缆所不具备的。TCP/IP 协议

族与 WinSock API 描述了一种客户端/服务器连接和编程的模式，而 IrDA 正是借用了这种非常成功的模式。

开放的协议对非 Windows 设备有很高的支持。一个实现 TinyTP 的非 Windows 设备将能够非常容易的与 Windows 应用程序交换信息。IrDA 与 WinSock 支持这种易用不需要配置并且总是共享工作数据的应用——例如点到点的通信网络。

2．3 IrDA 与 WinSock API：

在代理应用上，我考虑了三种处理方式：

1．用红外线接口高级协议（如 IrCOMM 等）转换串口通讯应用为红外通讯应用。

关于计算机互连，文件传输，代理上网的应用功能可由串口通讯应用完成，之后通过仿真到红外端口。由于这涉及到串口通信编程，鉴于 IrDA 与 WinSock 的结合所以并未采纳。

2．IrDA 与 WinSock 等 API 结合完成代理事务。

在使用套接口的 IrDA 用户函数，针对各种协议，转发数据包。从而完成计算机互连，文件传输，代理上网等等。

3．在 IP 层上实现 IP 包的转发。这样对高层服务就完全透明了，可以不再关心对高层服务的代理。并且 IP 层的代理还可以使客户程序不必进行什么配置，对客户和服务器可以做到完全透明。然而在实现上不仅仅要对 IP 的底层协议进行解释，还要切入系统，偷梁换柱，转发 IP 包，基本上是把系统的 IP 层软件替换掉了。这在客户端可能有相当大的麻烦，所以我并没有采用这个方案。

由于 IrDA 的标准协议，以及和 WinSock 的结合，使得可以直接从 IrDA 的 WinSock 调用完成通信。通过串口的互连应用需要完成串口底层通信协议，比较复杂，所以决定采用完全的 IrDA 与 WinSock 的透明接口实现系统通信，底层功能由设备驱动完成（如 标准串行红外线设备）。并且在采用了 IrDA 接口后，设备在不同操作系统下使用时，由于 IrDA 协议的标准化，使得后期修改非常容易，这会使修改维护工作变得简单。

三．代理应用采用 Windows Socket 套接口的网络编程

该项代理上网应用主要是通过 Windows Socket 套借口客户机/服务器模式的网络编程来实现的。Windows Sockets 规范定义并记录了如何使用 API 与 Internet 协议族（IPS，通常我们指的是 TCP/IP）连接，尤其要指出的是所有的 Windows Sockets 实现都支持流套接口和数据报套接口.

应用程序调用 Windows Sockets 的 API 实现相互之间的通讯。Windows Sockets 又利用下层的网络通讯协议功能和操作系统调用实现实际的通讯工作。它们之间的关系如图 1.1：

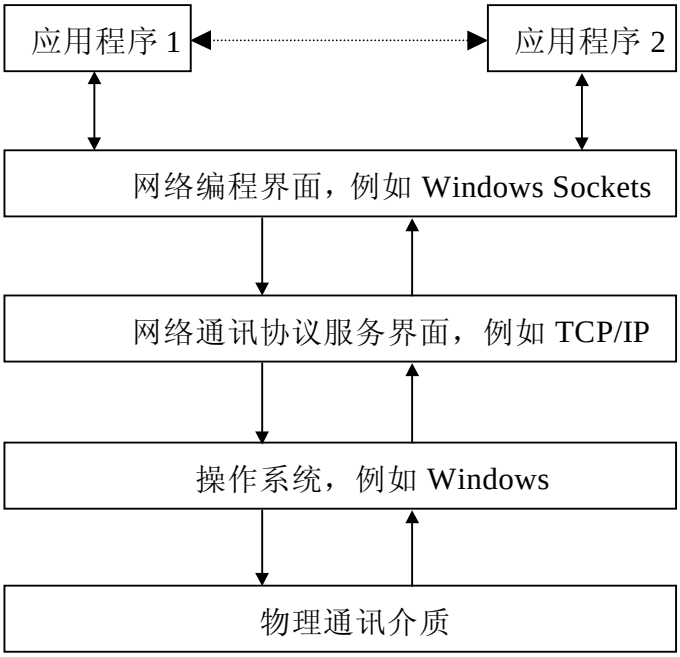


图 1-1 应用程序与 Windows Sockets 关系图

1.1 关于套接口

通讯的基石是套接口，一个套接口是通讯的一端。在这一端上你可以找到与其对应的一个名字。一个正在被使用的套接口都有它的类型和与其相关的进程。套接口存在于通讯域中。通讯域是为了处理一般的线程通过套接口通讯而引进的一种抽象概念。套接口通常和同一个域中的套接口交换数据（数据交换也可能穿越域的界限，但这时一定要执行某种解释程序）。Windows Sockets 规范支持单一的通讯域，即 Internet 域。各种进程使用这个域互相之间用 Internet 协议族来进行通讯（Windows Sockets 1.1 以上的版本支持其他的域，例如 Windows Sockets 2）。

用户目前可以使用两种套接口，即流套接口和数据报套接口。流套接口提供了双向的，有序的，无重复并且无记录边界的数据流服务。数据报套接口支持双向的数据流，但并不保证是可靠，有序，无重复的。也就是说，一个从数据报套接口接收信息的进程有可能发现信息重复了，或者和发出时的顺序不同。数据报套接口的一个重要特点是它保留了记录边界。对于这一特点，数据报套接口采用了与现在许多包交换网络（例如以太网）非常类似的模型。本代理应用是使用的流套接口。

1.2 客户机/服务器模型

在本代理应用中主要是使用客户机/服务器模型。在这种方案中客户应用程序向服务器程序请求服务。这种方式隐含了在建立客户机/服务器间通讯时的非对称性。客户机/服务器模型工作时要求有一套为客户机和服务器所共识的惯例来保证服务能够被提供（或被接受）。这一套惯例包含了一套协议。它必须在通讯的两头都被实现。根据不同的实际情况，协议可能是对称的或是非对称的。在对称的协议中，每一方都有可能扮演主从角色；在非对称协议中，一方被不可改变地认为是主机，而另一方则是从机。一个对称协议的例子是 Internet 中用于终端仿真的 TELNET。而非对称协议的例子是 Internet 中的 FTP。无论具体的协议是对称的或是非对称的，当服务被提供时必然存在“客户进程”和“服务进程”。

一个服务程序通常在一个众所周知的地址监听对服务的请求，也就是说，服务进程一直处于休眠状态，直到一个客户对这个服务的地址提出了连接请求。在这个时刻，服务程序被“惊醒”并且为客户提供服务－对客户的请求作出适当的反应。这一请求/相应的过程可以简单的用图 2-1 表示。虽然基于连接的服务是设计客户机/服务器应用程序时的标准，但有些服务也是可以通过数据报套接口提供的。

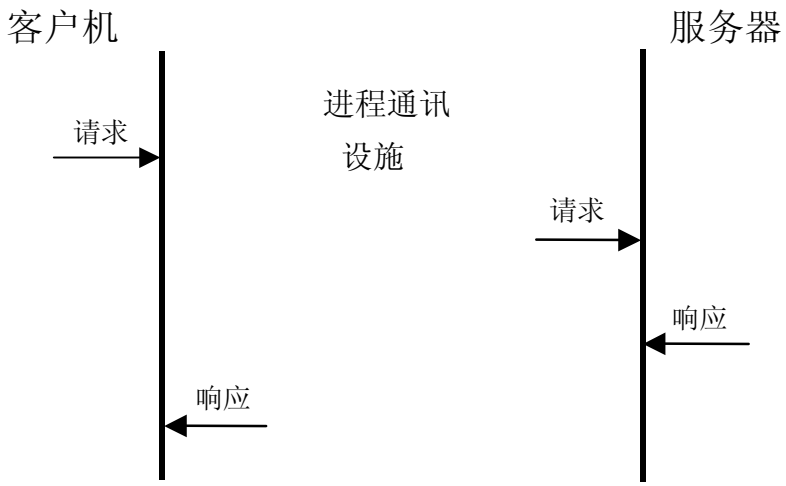


图 2-1 客户机/服务器模型

1．3 套接口网络编程原理

套接口有三种类型:流式套接口,数据报套接口及原始套接口.

流式套接口定义了一种可靠的面向连接的服务,实现了无差错无重复的顺序数据传输.数据报套接口定义了一种无连接的服务,数据通过相互独立的报文进行传输,是无序的,并且不保证可靠,无差错.原始套接口允许对低层协议如IP或ICMP直接访问,主要用于新的网络协议实现的测试等.

无连接服务器一般都是面向事务处理的，一个请求一个应答就完成了客户程序与服务程序之间的相互作用。若使用无连接的套接口编程，程序的流程可以用图3-1表示。

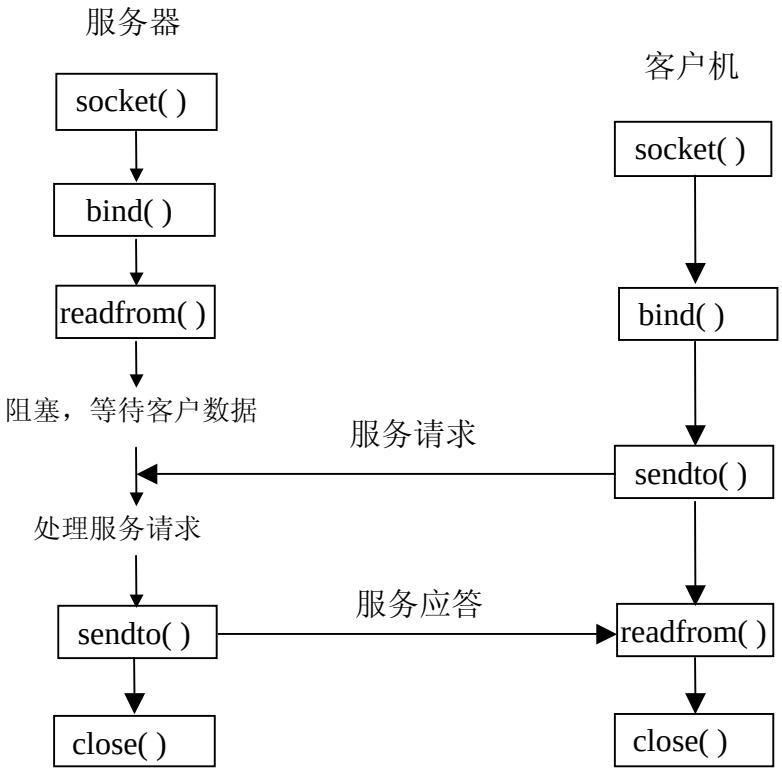


图 3-1 无连接套接口应用程序时序图

由于代理服务的性质，以及所代理的服务对连结的要求，我采用了面向连结的流套接口。

面向连接服务器处理的请求往往比较复杂，不是一来一去的请求应答所能解决的，而且往往是并发服务器。使用面向连接的套接口编程,可以通过图3-1来表示:其时序。

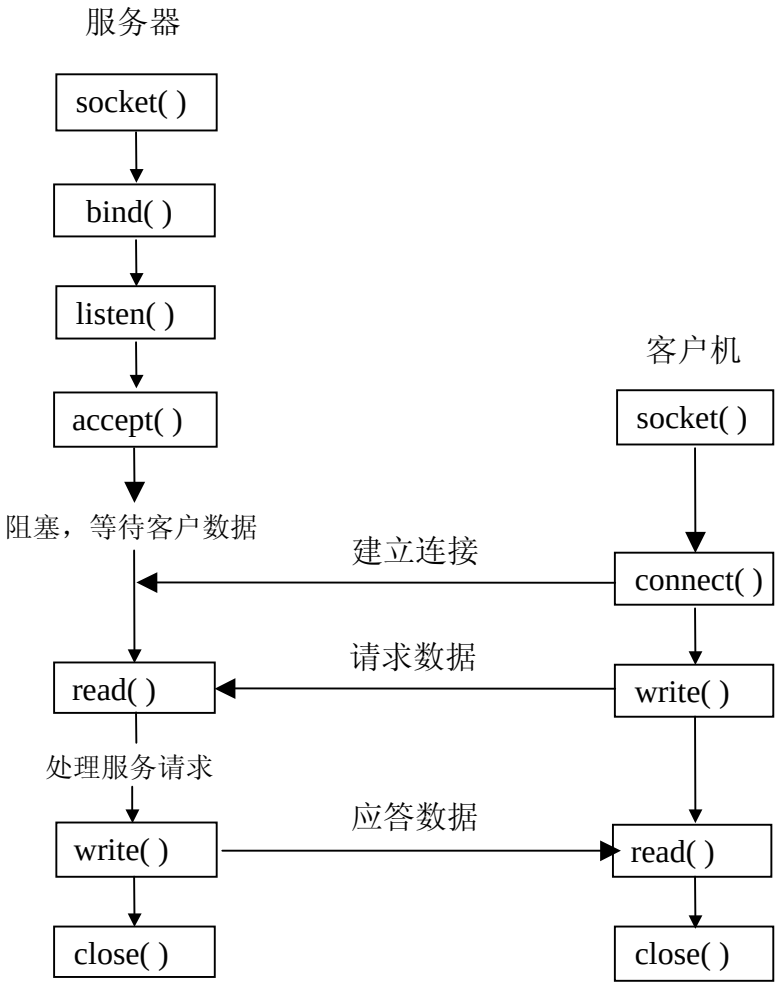


图 3-2 面向连接套接口应用程序时序图

套接口工作过程如下:服务器首先启动,通过调用socket()建立一个套接口,然后调用bind()将该套接口和本地网络地址联系在一起,再调用listen()使套接口做好侦听的准备,并规定它的请求队列的长度,之后就调用accept()来接收连接.客户在建立套接口后就可调用connect()和服务器建立连接.连接一旦建立,客户机和服务器之间就可以通过调用read()和write()来发送和接收数据.最后,待数据传送结束后,双方调用close()关闭套接口.

1．4 代理应用采用的一些 Windows Sockets 编程原理

本代理应用并没有利用Windows的基于消息的特点，没有采用异步机制，而是采用在新线程中处理整个通信事务。因为代理服务可能面临着多个客户的同时使用。

应用所采用的Sockets错误处理：

WINSOCK提供了两个WSAGetLastError()和WSASetLastError()来获取和设置最近错误号.

启动和终止：

由于Windows Sockets的服务是以动态连接库WINSOCK.DLL形式实现的,所以必须要先调用WSAStartup()函数对Windows Sockets DLL进行初始化,协商WINSOCK的版本支持,并分配必要的资源.

在应用程序关闭套接口后,还应调用WSACleanup()终止对Windows Sockets DLL的使用,并释放资源,以备下一次使用。

四．通过红外线的代理上网程序设计实现的主要细节

（以 Http 为例，VC 环境）

有关 IrDA 的所有定义都包含在 af_irda.h 头文件中，当应用工程包含进这个头文件后，只要以有关于 IrDA 的宏定义或常量调用 WinSock 函数，就可以很好地使用标准红外线设备进行通信。

整个应用系统分为代理客户端和代理服务器端。代理客户端仅仅负责监听代理端口，在客户应用发出请求后通过红外通道完整的转发到代理服务器端，对信息不做任何处理。代理服务器端则需要分析客户请求的内容，以便连结到正确的服务器及其端口，并且能够对服务器的响应做出必要的解释。

就实现而言，客户机（便携机）需要下载代理程序的客户端，与服务器上的代理程序服务端同时运行。代理客户端与代理服务端对端口的监听、接受连结、发送接收数据包等事务均可由 Windows Socket 的客户端/服务器端模式完成。对于处理转发事务的红外端口而言，在发送数据前采用一些底层的 API 找到红外设备及系统提供的套接字，然后就可以正常发送数据了；在接收数据前，以 IrDA 作 Socket 调用，然后正常接受连接进行数据收发。

主要使用了以下 API 函数及数据结构：

socket()

该函数建立指定地址格式，数据类型和协议下的套接口，地址格式为AF_INET（在红外线下为AF_IRDA），数据类型SOCK_STREAM表示建立流式套接口，参数三为0，即协议缺省。

```
SOCKET sListenSock;//本地监听套接字
//make socket
sListenSock=socket(AF_INET,SOCK_STREAM,0);
```

SOCKADDR_IRDA Structure

IrDA Socket 地址结构，是关于连结的主机信息

```
SOCKADDR_IRDA addr={ AF_IRDA, 0, 0, 0, 0, "IRClient" };
```

bind()

该函数将建立服务器本地的半相关。其中，server是sockaddr_in结构，其成员描述了本地端口号和本地主机地址，经过bind()将服务器进程在网上标识出来。

```
SOCKADDR_IRDA    ServSockAddr = { AF_IRDA, 0, 0, 0, 0, "SampleIrDAService" };
int    SizeOfSockAddr;
if (bind(ServSock, (const struct sockaddr *) &ServSockAddr,sizeof(SOCKADDR_IRDA)) ==
SOCKET_ERROR)
{
   // WSAGetLastError()
}
```

listen()

然后，建立连接。先是调用listen()函数表示开始侦听，再通过accept()调用等待接收连接。

listen(s,1)表示连接请求队列长度为1，即只允许有一个请求，若有多个请求，则出现错误,给出错误代码WSAECONNREFUSED。

```
if (listen(ServSock, 2) == SOCKET_ERROR)
{
   // WSAGetLastError()
}
```

accept()

accept() 阻塞（缺省）等待请求队列中的请求。一旦有连接请求来，该函数就建立一个和s有相同属性的新的套接口。client也是一个sockaddr_in结构，连接建立时填入请求连接的套接口的半相关信息。

接下来，就可以接收和发送数据了。

```
SOCKET NewSock;
while(1)
{
   sizeofSockAddr = sizeof(SOCKADDR_IRDA);
   if ((NewSock = accept(ServSock, (struct sockaddr *) &PeerSockAddr,
               &sizeofSockAddr)) == INVALID_SOCKET)
   {
      // WSAGetLastError()
      // exit
   }
   // NewSock is a connected socket.
   // Create a new thread and pass it NewSock, return to accept() on main
   // or use NewSock here until done, then close it.
}
```

send() and recv()

上面两个函数分别负责接收和发送数据，recv从ns（建立连接的套接口）接收数据放入buf中，send则将buf中数据发送给ns。至于第四个参数，表示该函数调用方式，可选择MSG_DONTROUTE和MSG_OOB，0表示缺省。

```
int   BytesRead, BytesSent;
char   Buffer[4096];
// recv() example
if ((BytesRead = recv(Sock, Buffer, sizeof(Buffer), 0)) == SOCKET_ERROR)
{
   // WSAGetLastError()
}
if (BytesRead == 0)
{
   // Peer has closed the connection and I have all the data.
   // Close the socket now.
}
// send() example
```

```
if ((BytesSent = send(Sock, Buffer, sizeof(Buffer), 0)) == SOCKET_ERROR)
{
   // WSAGetLastError()
}
// Check that BytesSent == sizeof(Buffer).
```

closesocket()

最后，关闭套接口。

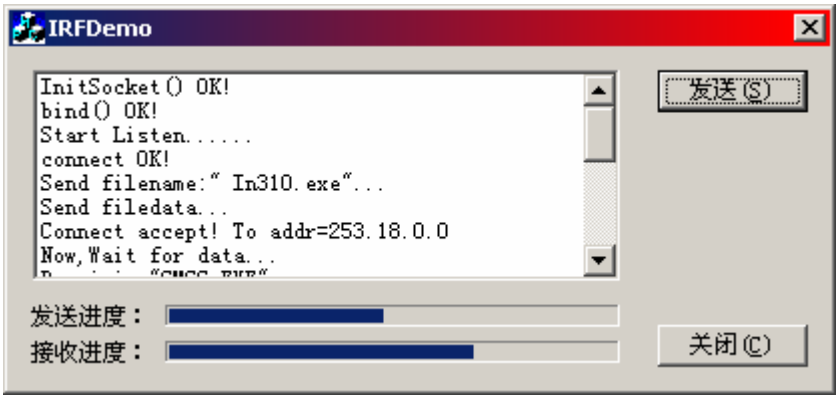
```
if (closesocket(Sock) == SOCKET_ERROR)
{
   // WSAGetLastError()
}
```

结论：

经过系统程序设计和试验，在客户端给出请求后，代里程序只要将服务器的所有响应完整的转发给客户端，就可以完成 Http 的红外线代理上网。

测试程序及成果：

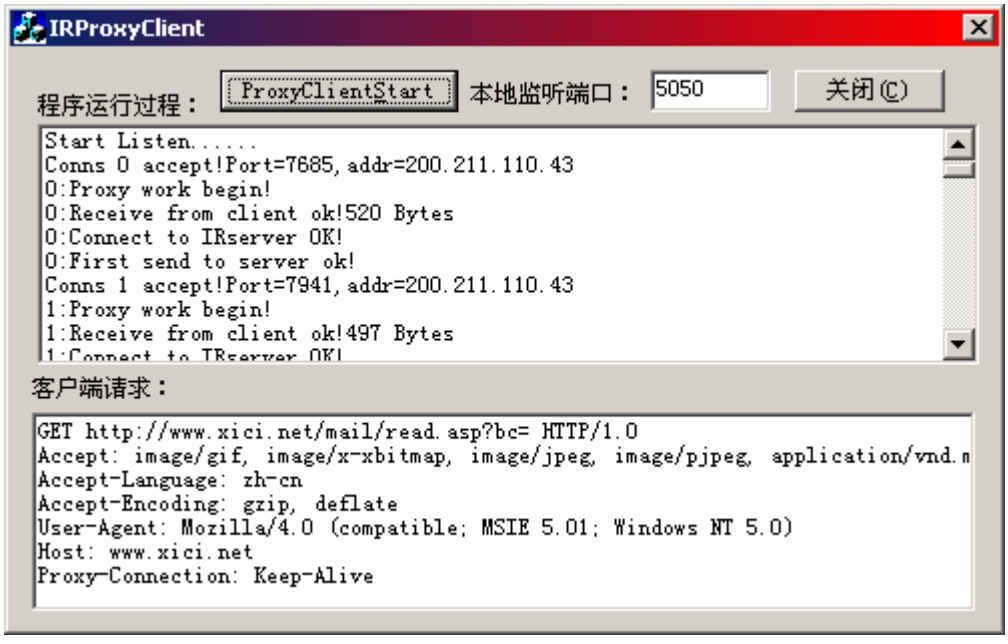
1. 标准红外线互传文件应用：



IRFDemo.exe 客户和服务器两边都可以同时通过红外线互传文件。

有信息到达，软件自动通知用户文件到来，由用户选择是否接收。

2. 标准红外线设备的 Http 代理应用：（需要标准红外线设备）



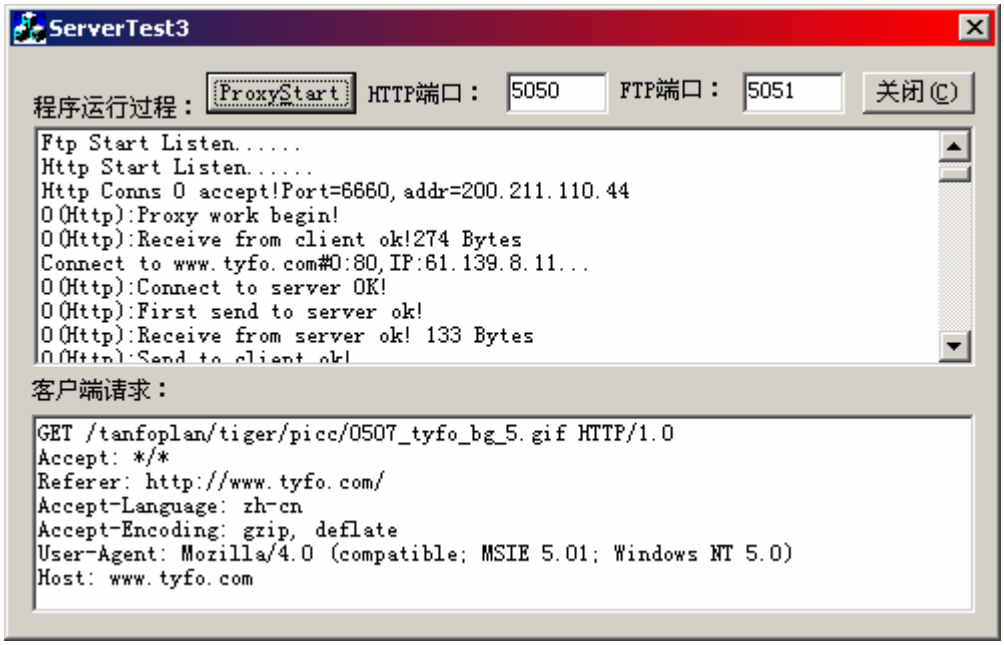
IRProxyServer.exe（代理服务端）与客户端请求的服务器相连接

IRProxyClient.exe（代理客户端）与客户端应用程序相连接。

代理端口由客户端设定。

软件界面是对每一个请求的解析显示。

3. Http 与 Ftp 代理应用（无红外线支持）：



ProxyServer.exe

该程序在代理服务器上运行，通过分别设置 HTTP 端口和 FTP 端口，系统可同时进行超文本和文件传输协议的代理。

软件界面是对每一个请求的解析显示

讨论：

在 Http 下一般客户端对于每一次下载信息，只给出一个请求，所有的参数、客户信息都在一个请求中上传到服务器。代理在对这个请求做出解析后，只要联接上了服务器，成功将请求转发给服务器，就只作服务器端响应的转发了。

参考资料：

0. 用 TCP/IP 进行网际互连（第一卷） DOUGLAS E.COMER 著 电子工业出版社
0. Visual C++ 5 开发人员指南 机械工业出版社
0. Windows Sockets 规范及应用 施炜 李铮 秦颍 编著
1. VC知识库 www.vckbase.com
2. MSDN www.microsoft.com/china/msdn IrDA: Background and Overview
3. 红外协议：http://www.pday.com.cn/technology/irda.htm
7. IrDA 官方站点：http://www.irda.org/

Connect to Internet through an agent with infrared devices

ABSTRACT

Now in the economic society,man can get information quickly by means of luggable devices.Through the standard infrared device on a luggable computer,you can take the shortcut and convenience in the process of capturing information you need.Relative to the lumpish line,infrared connecting make the Internet work and game much more easily and joyfully.

In this paper,the main work is the realization of an application level agent with Irda programming .

Key Word: IrDA,WinSocket,Agent,Http,FTP