

基于 TCP/IP 和 Winsock 文件上传的应用研究

徐向东, 周国祥, 石 雷

(合肥工业大学 计算机与信息学院, 安徽 合肥 230009)

摘 要: 文件上传已经成为当今网络管理信息系统的一个重要组成部分, 文章在对 TCP/IP 协议和 Winsock 技术进行分析研究的基础上, 以保险中介业务管理信息系统的设计为例, 阐述了基于 TCP/IP 协议和 Winsock 的文件上传方法在 C/S 结构下的实现过程; 实践结果表明, 应用该方法可以快速准确地实现文件的上传。

关键词: TCP/IP; 套接字; Winsock; 文件上传

中图分类号: TP311 **文献标识码:** A **文章编号:** 1003-5060(2008)11-1790-04

Research and application of file uploading based on TCP/IP and Winsock

XU Xiang-dong, ZHOU Guo-xiang, SHI Lei

(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: File uploading has already become an important component in the net MIS today. In this paper, the TCP/IP protocol and the Winsock technology are analyzed, and an insurance medium business management information system is taken as an example to expound the realization of the method of file uploading based on TCP/IP protocol and Winsock in C/S structure. The result shows that the files can uploaded exactly and quickly by applying the presented method.

Key words: TCP/IP; socket; Winsock; file uploading

目前网络管理信息系统主要都采用客户机/服务器(C/S)和浏览器/服务器(B/S)2种结构模型,在这2种模型中都涉及到客户端与服务器端的信息交互问题,如从服务器到客户端的文件下载、从客户端到服务器的文件上传等。比较B/S和C/S这2种结构,B/S结构下的文件下载和文件上传都比较容易实现。而在C/S结构下,可以通过FTP编程来实现^[1],但是这种方法要求服务器端必须配置FTP服务,需要用户名和密码,这期间还要有权限的配置,而且调试很不方便,实现起来较为繁琐。此外,还有服务器端建立共享目录的方法^[2],但这种方法需要服务器共享影响安全性,并且同样需要用户名和密码。因此在C/S结构下实现向服务器的指定文件目录中上传文件,是一项比较困难的工作。本文针对这样的问题进行研究,提出了相关的解决方法。

1 TCP/IP 和 Winsock 相关技术

传输控制协议/互联网协议(Transmission Control Protocol/Internet Protocol,简称TCP/IP)是Internet最基本的协议,也是当今世界上使用最广泛而不基于任何特定硬件平台的网络通信协议^[3]。TCP/IP协议在网络通信中主要负责完成路由选择与数据传输等工作。

Windows Sockets(简称Winsock)是微软的窗口系统结构(WOSA)的一部分,是Windows下得到广泛应用、开放、支持多种协议的网络编程接口。它定义并记录了如何使用API与Internet协议族(IPS,通常指的是TCP/IP协议)的连接。

1.1 Sockets(套接字)

Winsock网络编程,其API是建立在Sockets(套接字)的基础上的。Sockets最早是由

收稿日期:2007-12-15;修改日期:2008-04-07

作者简介:徐向东(1983-),男,安徽安庆人,合肥工业大学硕士生;

周国祥(1956-),男,安徽合肥人,合肥工业大学教授,硕士生导师。

Berkeley 大学提出的在 UNIX 操作系统下实现 TCP/ IP 通信协议的开发接口^[4,5]。随着 Internet 在全球范围内的广泛使用, Sockets 已成为网络编程的通用接口。

Sockets 的实质是通信端点的一种抽象, 提供一种发送和接收数据的机制。Windows 套接字主要有数据报套接字(Datagram Socket) 和流式套接字(Stream Socket) ^[6] 2 种形式。

数据报套接字采用 UDP 协议, 它建立在 IP 协议上, 提供无连接的数据报传输, 支持双向的数据流, 但并不保证是可靠、有序和无重复的。

流式(也称面向连接方式) 套接字采用的是 TCP 协议, 它提供了双向、有序、无重复并且无记录边界的数据流服务。在这种方式下, 2 个通讯的应用程序之间先要建立一种虚拟的连接。流式套接字工作流程, 如图 1 所示^[7]。

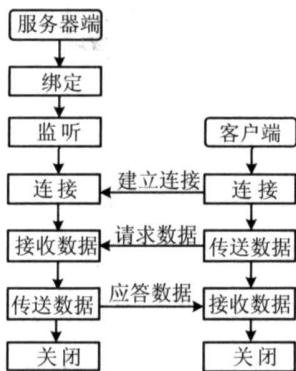


图 1 流式套接字工作流程

首先启动服务器端, 通过调用 Socket 函数建立一个 Socket 对象, 然后调用 bind () 函数将 Socket 对象和本地网络地址绑定到一起, 再调用 listen () 函数使该 Socket 对象处于监听状态, 等到客户端有连接请求时服务器才会再次激活, 收到客户端请求以后, 服务器调用 accept () 函数来接受该连接。客户端在建立 Socket 对象后就调用 connect () 函数和服务器建立连接, 连接一旦建立, 客户机和服务器之间就可以通过调用基本的网络 I/O 函数来发送和接收数据, 传送结束后, 双方关闭初始创建的 Socket 对象。

1. 2 Winsock 通信机制

应用程序的网络通信是利用相同的通信协议来完成信息传输, 应用程序和 Winsock 都工作在 Windows 的用户模式下, 操作系统仅仅通过 Winsock 是不能完成网络间的通信, 还需要底层的支持。而套接字仿真器(套接字核心模式驱动

程序) 和传输驱动程序接口(Transport Driver Interface, TDI) 是负责操作系统核心环境下的网络通信, 起到了 Winsock 和传输协议之间的通信桥梁作用, 其通讯机制示意图如图 2 所示。

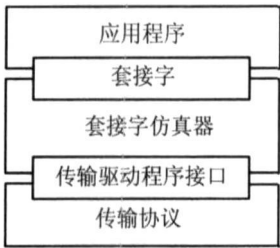


图 2 套接字通信机制示意图

当应用程序利用 Winsock 发送和接受数据时, 并不是由 Winsock 从网络上发送和接收数据的, 而是由核心模式驱动程序 AFDSYS 负责管理发送和接收缓冲区来发送和接收数据。

2 C/S 结构下文件上传的实现

保险中介业务管理信息系统采用的是 C/S 结构, 其中涉及到图片、文档等文件的上传, 所以程序采用了这种基于 TCP/ IP 协议和 Winsock 的文件上传方法, 选择同步流式套接字(Stream Socket) 来完成 C/S 模式的通信, 保证数据能够准确无误的多线程传输^[8]。

在保险中介业务管理信息系统的文件上传过程中, 客户端与服务器端的通信主要分为 2 个部分: 一部分是用于传送控制信息, 例如发送文件的请求, 文件的名称、大小等。本文定义了一个文件信息结构体 tp- MessageInfo 用于存储文件的相关信息(如文件的序号、长度、起始位置、文件名等), 服务器端可以通过这个结构体来存储和获得文件的相关信息。而另一部分就是文件的传输部分, 在服务器端的文件接收程序和客户端的文件上传程序中分别写了相关的线程, 每个线程在具体实现时分别建立有自己的套接字^[9]。

2 1 服务器文件接收系统

在服务器的接收端, 要实现的功能有监听客户的连接请求、与客户建立连接和接收上传的文件。这主要涉及监听线程 BF- ListenThread 和与客户端建立连接并接收文件的线程 BF- ProcessClientConnectThread。

监听线程 BF- ListenThread 中首先创建一个监听套接字进行监听, 并循环接受来自客户端的连接请求, 对于每个连接 socket, 都为其启动一

个单独的线程来进行连接并接收上传的数据。主要代码如下:

```
DWORD WINAPI BF_ListenThread(LPVOID lpParam)
{
    CReceiveFileDialog * pDlg = (CReceiveFileDialog *) lpParam;
    SOCKET socketListen = pDlg->m_socketListen;
    //创建监听套接字
    int nCount; //设置一个变量用于统计连接数
    int nRet = listen(socketListen, 60); //用传递进来的参数所指向的 socket 监听, 允许最大连接数是 60
    if(nRet == SOCKET_ERROR) return 0; //有错误发生, 返回 0
    //循环接收客户端的连接请求, 进行连接计数, 并对于每一个连接 socket, 都为其启动一个单独线程;
    DWORD dwThread; tp_ThreadParameter param;
    //对 param 进行初始化;
    :: CreateThread(NULL, 0, BF_ProcessClientConnectThread, (LPVOID)(&param), 0, &dwThread);
    Sleep(300); //延时的目的是为了 避免首次读取的“协议”被新线程覆盖
    return 1;
}
```

与客户端建立连接并接收文件的线程 BF_ProcessClientConnectThread 中, 首先创建一个连接套接字, 与客户端进行连接, 再设置一个接收缓冲区, 调用 BF_ReadN() 函数从客户端读取消息数据并调用 BF_ReceiveFile() 函数接收上传的文件, 接收完毕后关闭连接用的 socket。主要代码如下:

```
DWORD WINAPI BF_ProcessClientConnectThread(LPVOID lpParam)
{
    tp_ThreadParameter * pParameter = (tp_ThreadParameter *) lpParam;
    SOCKET socketConnect = pParameter->socketConnect;
    tp_MessageInfo * pMsg; //文件消息
    char pBuf[MESSAGE_INFO_SIZE]; //接收缓存, 其中 MESSAGE_INFO_SIZE 为 sizeof(tp_MessageInfo)
    CString sInfo, sFileFullName; int nCount = atoi(sInfo);
    if(nRet == BF_ReadN(socketConnect, pBuf, MESSAGE_INFO_SIZE); //从客户端读取消息数据
    if(nRet < 0) goto ERR_EXIT; //读取数据发生错误
    nRet = BF_ReceiveFile(socketConnect, pMsg->
```

```
nFileSeek, pMsg->nFileLen, sFileFullName, -1); //接收文件
    if(nRet < 0) goto ERR_EXIT; //接收文件失败
    //关闭连接 socket, 连接数进行减 1 操作, 返回 0;
    ERR_EXIT: //关闭连接 socket, 连接数进行减 1 操作, 返回 nRet;
}
```

服务器端的文件接收程序的主要流程如下: 首先启动服务器, 通过调用 socket() 创建一个同步流式套接字, 然后调用 bind() 将该套接字与本地网络地址联系在一起, 接着启动监听线程来监听是否有客户端发送上传文件的请求, 设置允许的最大连接数是 60。监听线程循环接收来自客户端的连接请求, 对于每一个连接 socket, 都为其启动一个单独的线程来建立连接并接收数据, 文件接收完毕后关闭线程。

2.2 客户端文件上传程序

客户端的文件上传程序主要实现多线程上传文件的功能。本文定义一个结构体 tp_UploadFileThreadParameter 来存储待上传文件的文件信息参数, 其中包括拟传送文件在传送列表中的序号、文件的长度、起始位置, 文件在客户端的全文件名(包含绝对路径)和文件在服务器端存储的文件名(含相对路径)等。

客户端的上传程序只涉及到一个文件上传线程 BF_UploadFileThread。在该线程中, 首先接受传入的文件信息参数, 获取服务器的 IP 地址和端口, 然后创建一个连接套接字并连接服务器端, 调用 BF_SendN() 函数发送请求上传文件的消息, 服务器响应后即与服务器端建立连接并调用 BF_SendFile() 函数发送文件数据。文件上传完毕后即关闭连接用的 socket。主要代码如下:

```
DWORD WINAPI BF_UploadFileThread(LPVOID lpParam)
{
    //将传入的参数信息分别赋值给每一个变量;
    sockaddr_in local;
    //对 local 进行初始化;
    SOCKET socketConn = socket(AF_INET, SOCK_STREAM, 0); //创建连接套接字
    int nRet = connect(socketConn, (LPSOCKADDR)&local, sizeof(local)); //连接服务器端
    tp_MessageInfo msgInfo;
    //对结构体 msgInfo 中的每一个成员进行赋值;
    nRet = BF_SendN(socketConn, (char *)&msgInfo, sizeof(tp_MessageInfo)); //发消息给服务器
    if(nRet < 0)
```

```
(返回错误信息; nRet = 101; goto ERR_EXIT;)  
nRet = BF_SendFile (socketConn, nSeek, nLen,  
sFileFullName, nFileIndex); //发文件给服务器  
if(nRet < 0)  
(返回错误信息; nRet = 102; goto ERR_EXIT;)  
(关闭连接 socket, 返回 0; )  
ERR_EXIT: (关闭连接 socket, 返回 nRet; )  
}
```

客户端的文件上传程序流程比较简单, 首先在文件列表中加入待上传的文件, 然后逐一上传, 为每个文件均启动一个文件上传线程, 上传完毕后关闭线程。保险中介业务管理信息系统的客户端文件上传程序的具体实现结果如图 3 所示。

对于每个上传的文件, 通过相关程序在数据库中存储了其在服务器中全路径和文件名, 用户可以通过浏览器查看或下载这些文件。上传成功的文件存放在服务器端及相关程序中预先指定的文件夹中, 如图 4 所示。



图 3 文件上传成功



图 4 已成功上传到服务器端的文件

3 结束语

基于 TCP/IP 协议和 Winsock 的文件上传方法现已成功地应用在保险中介业务管理信息系统中, 解决了图片、文档等文件的上传问题, 并通过相关程序在数据库中存储了上传到服务器的文件的全路径, 用户可以通过浏览器查看或下载这些文件。使用此种方法上传文件, 不受文件大小限制, 不需要用户名和口令的权限验证, 只需运行通信程序即可实现安全可靠的文件上传。

实践表明, 基于 TCP/ IP 协议和 Winsock 的文件上传方法的引入, 解决了在 C/S 结构下实现文件上传的诸多困难, 用户可以从客户端向服务器的指定文件目录中快速准确地上传文件。

[参 考 文 献]

[1] 马永琴, 李正文, 汪刘艳. 基于 WinSock 的文件互传[J]. 重庆大学学报(自然科学版), 2004, 27(4): 49- 52.
[2] 姚占东. 一种基于局域网的文件传输技术[J]. 计算机应用, 2004, 24(6): 45- 46.
[3] 尹浩琼, 李 剑. TCP/IP 详解[M]. 第 2 版. 北京: 电子工业出版社, 2003: 10- 100.
[4] 黄国盛, 梁平原. 通过 WinSock 实现 TCP/IP 网络通信[J]. 吉林大学学报(自然科学版), 2002, 23(2): 66- 69.
[5] 瞿有甜, 盛贤良. 基于 TCP 套接字的应用研究[J]. 计算机应用研究, 2006, 23(7): 47- 49.
[6] 束长宝, 于 照, 张继勇. 基于 TCP/IP 的网络通信及其应用[J]. 微计算机信息, 2006, 22(36): 157- 159.
[7] 葛广英, 彭洲红, 陈莉莉, 等. 基于 WinSock 的多媒体信息网络传输的研究[J]. 微计算机应用, 2004, 25(4): 444- 448.
[8] 王丰锦, 邵新宇, 喻道远. 基于 SOCKET 和多线程的应用程序间通信技术的研究[J]. 计算机应用, 2000, 20(6): 65- 67.
[9] 陈自强, 郭嘉琳, 刘忠信. 具有断点续传功能的文件传输系统的设计与关键技术[J]. 计算机工程, 2002, 28(12): 14- 16.

(责任编辑 吕 杰)