基于 TCP/IP 的断点续传系统研究

Breaking point supervertion transfer system research based on TCP/IP

周 昕　熊前兴　赵卫利　（武汉理工大学计算机科学与技术学院 武汉 430063）

摘要：针对现代企业的发展需要，提出了一种基于 TCP/IP 的文件断点续传系统的解决方案，该方案采用 C/S 模式，提供了稳定可靠的传输技术，适用于各种网络。文章首先介绍了系统的工作原理，接着分析了系统设计中的一些关键技术，包括 Winsock 编程的特点、通信指令的定义、数据封装等，最后探讨了系统的实现。

关键词：C/S 模式 套接字 TCP/IP 阻塞 数据封装

1 引言

随着现代计算机网络通信技术的飞速发展，在一个现代企业中，办公自动化已经成为一个很重要的组成部分，传统企业中各种文件、逐级上报的方式已经不能满足现有企业的发展需要，因此需要一种快速、稳定、高效的文件传输方式。

本文基于这一现状，通过 Delphi 开发出一种基于 TCP/IP 的大型文件断点续传系统。该系统不仅可以有效可靠的进行各种文件的传输，而且实现了断点续传的功能，因此适用于各种网络（局域网，电话拨号，internet），从而节省网络资源，提高传输效率。文章将对系统的工作原理及关键技术进行探讨。

2 Winsock 编程技术

Winsock 是 Microsoft 以 Sockects（套接字）为基础开发的在 Windows 环境下网络编程接口。

2.1 Winsock 的两种工作方式

第一种是面向连接的流方式，该过程就像打电话。这种方式下，通信的应用程序之间要建立一种连接链路，然后数据才能被正确的接收和发送。这种方式对用的是 TCP 协议。

第二种是无连接的数据报文方式，对应的是 UDP 协议。这种方式不需要事先建立连接。

流方式的特点是通信可靠，对数据有校验和重发的机制，通常用来做数据文件的传输如：FTP、TELNET 等；数据报文方式由于取消了重发校验机制，能够达到较高的通信速率，可以用于对数据可靠性要求不高的通信，如实时的语音，广播消息等。为了数据传输的可靠性，本文采用流方式。

2.2 Winsock 编程特点

网络通信中，由于网络拥挤或者一次发送数据量过大等原因，经常发生数据不能在短时间内传送完，收发数据的函数因此不能返回的现象，这种现象叫阻塞。

Winsock 采用两种方式处理阻塞：阻塞和非阻塞方式。在阻塞方式下，收发数据的函数在被调用后一直要到传送完毕或者出错才能返回；而对于非阻塞方式，函数被调用后会立即返回。本文中服务器采用阻塞方式，客户端采用非阻塞方式。

3 系统结构及工作原理

3.1 系统结构介绍

整个系统由服务器和客户端组成。

客户端主要包括上传文件、下载文件、断点上传、断点下载 4 个功能。系统运行时，客户端是系统的发起者，当需要传送数据时，先向服务器发送各种通信指令，以确定是上传还是下载，是断点上传还是断点下载。

服务器是整个传输系统的核心，它接收客户端发送过来的各种通信指令，通过分析这些指令，然后作出相应的回应。服务器采用多线程，支持多用户与服务器同时进行文件传输，极大的提高的服务器的工作效率。

3.2 系统工作原理

对于上传文件，其过程如图 1。

(1) 客户端向服务器发送上传指令

(2) 服务器接受到该指令后，通过读取上传文件信息记录(该信息可以通过 XML，ACCESS，INI 等文件保存)，以判断是否为断点上传。

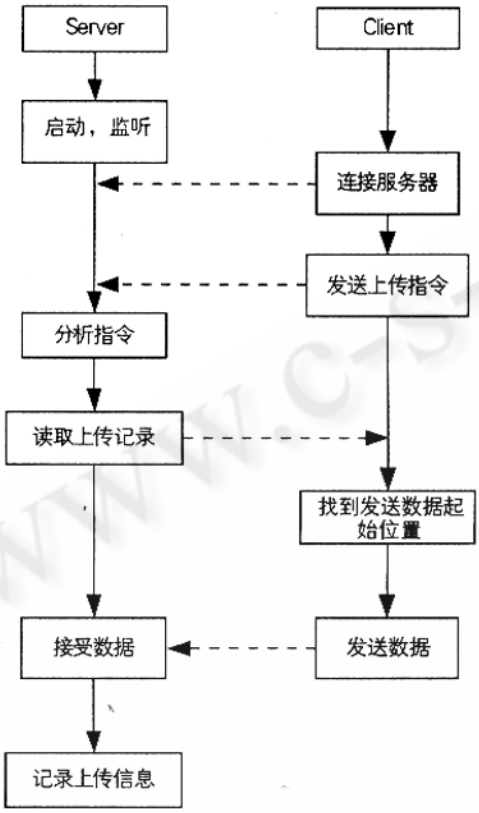

图 1　上传文件工作流程图

(3) 如果不是断点上传，则发送准备接受数据指令，然后客户端从文件的第一个字节开始进行数据的传输。

(4) 如果判断为断点上传，则读取以前上传文件的信息(包括服务器已经接受了多少字节，完成了百分之几等)，再把这些信息发送给客户端，客户端通过这些信息来确定从文件的哪个地方开始上传，然后进行数据的传输。

对于下载文件，其过程如图 2。

(1) 客户端发送下载或者断点下载指令

(2) 如果是下载指令，服务器直接从文件的第一个字节开始发送数据。

(3) 如果是断点下载，客户端首先要读取下载文件信息记录，获取已经下载的字节数，然后把这些信息发送给服务器，服务器根据该信息，找到开始发送的起始位置，然后进行数据的传输。

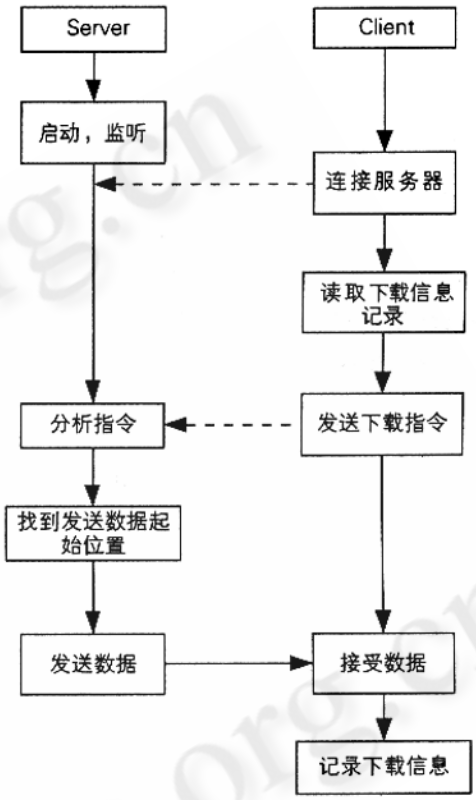

图 2　下载文件工作流程图

4 系统关键技术分析

4.1 通信指令的定义

这里的通信指令是系统中客户端跟服务器进行对话的语言，服务器只有通过这些指令才能了解客户端进行的操作，从而作出相应的回复。系统中使用的通信指令如下：

```
//上传指令
UP_USERINFO = 'UP_CMD00';
UP_QUERY = 'UP_CMD01';
UP_NEXTWILLBEDATA = 'UP_CMD04';
UP_DATA = 'UP_CMD05';
UP_END = 'UP_CMD06';
UP_RESUME = 'UP_CMD07';
```

```
//下载指令
DN_QUERY = 'DN_CMD01';
DN_FILEINFO = 'DN_CMD02';
DN_NEXTWILLBEDATA = 'DN_CMD04';
DN_DATA = 'DN_CMD05';
DN_END = 'DN_CMD06';
DN_RESUME = 'DN_CMD07';
DN_FILENOTEXIST = 'DN_CMD08';
//数据传输完毕标志
MP_SENDDATAEND = 'GF_CMD10';
//结束数据传输标志
MP_SYSTEM_STOP = 'GF_CMD00';
```

4.2 数据封装技术

4.2.1 OSI 参考模型中的数据封装过程

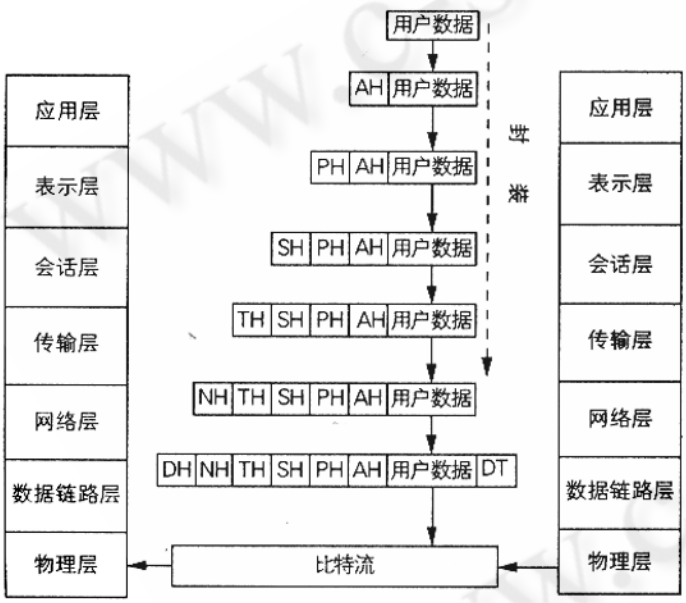

图 3 OSI 参考模型中的数据封装过程

如图 3 所示，在 OSI 参考模型中，当一台主机需要传送用户的数据（DATA）时，数据首先通过应用层的接口进入应用层。在应用层，用户的数据被加上应用层的报头（Application Header，AH），形成应用层协议数据单元（Protocol Data Unit，PDU），然后被递交到下一层－表示层。

表示层并不"关心"上层－应用层的数据格式而是把整个应用层递交的数据包看成是一个整体进行封装，即加上表示层的报头（Presentation Header，PH）。然后，递交到下层－会话层。

同样，会话层、传输层、网络层、数据链路层也都要分别给上层递交下来的数据加上自己的报头。它们是：会话层报头（Session Header，SH）、传输层报头（Transport Header，TH）、网络层报头（Network Header，NH）和数据链路层报头（Data link Header，DH）。其中，数据链路层还要给网络层递交的数据加上数据链路层报尾（Data link Termination，DT）形成最终的一帧数据。

当一帧数据通过物理层传送到目标主机的物理层时，该主机的物理层把它递交到上层－数据链路层。数据链路层负责去掉数据帧的帧头部 DH 和尾部 DT（同时还进行数据校验）。如果数据没有出错，则递交到上层－网络层。

同样，网络层、传输层、会话层、表示层、应用层也要做类似的工作。最终，原始数据被递交到目标主机的具体应用程序中。

4.2.2 系统中数据封装

系统中的数据封装主要解决的问题就是：如何产生"用户数据"，使得服务器跟客户端都能通过分析"用户数据"得出通信指令类型，然后作出相应的操作。下面对系统中数据封装进行说明：

如上传时客户端发送的数据为：

//上传用户的信息

UP_USERINFO + 用户信息 + MP_SENDDATAEND

//上传文件的信息

UP_QUERY + 上传文件信息 + MP_SENDDATAEND

下载时发送的数据

UP_USERINFO + 下载用户的信息 + MP_SENDDATAEND

DN_FILEINFO + 下载的文件信息 + MP_SENDDATAEND）；

数据传输完毕时发送的数据

UP_END + MP_SENDDATAEND

4.3 Delphi 中网络控件使用

4.3.1 服务器端控件 ServerSocket

设置 ServerType 为 stThreadBlocking 即采用阻塞方式；

设置 Port 为 4012（自定义）；

添加 OnGetThread 事件，当有客户端连接服务器时，该事件将被触发。服务器如果采用多线程，可以在该事件中创建新的线程，该线程将与连接的客户端单独进行数据的通信，而具体的数据通信功能可以在线程的 Execute 中完成。如果不采用多线程，则可以直接在该事件中实现数据传输。

线程的创建：

```
TmyServerClientThread = class ( TServerClient-
Thread) ;
SocketThread = TmyServerClientThread. Create ( Cli-
entSocket) ;
SocketThread. ThreadClientSocket: = ClientSocket;
SocketThread. Resume;
```

线程的执行（完成数据传输功能）：

（1）读取客户端传送的数据

```
ReceiveText: array[0..1024] of char;
ThisSocketStream. Read ( ReceiveText, SizeOf ( Re-
ceiveText) ) ;
```

（2）分析数据，提取通信指令类型

原理：因为数据都是已经通过封装了的，所以只需要获取指令 MP_SENDDATAEND 前面的字符串就可以得到通信指令类型。

```
strtmp: = copy ( ReceiveText, 1, length ( MP_SEND-
DATAEND) ) ;
```

（3）根据指令完成相应功能。

如客户端下载数据：

```
if strtmp = DN_DATA then
begin
  //获取客户端已经下载的字节数;
  //寻找发送文件数据的位置;
  fs: = TFileStream. Create( FileName, fmShareDe-
nyWrite) ;
  fs. Seek( 传送字节数,soFromBeginning) ;
 nRetr: = fs. Read( buffer^,1024) ;
  //发送数据;
  ThisSocketStream. Write( buffer^,nRetr) ;
end;
```

4.3.2　客户端控件 ClientSocket

设置 ClientType 为 ctNonBlocking 即非阻塞方式。

设置 Port 为 4012 ，与服务器设置一样。

添加 OnRead 事件，该事件会在 ClientSocket 接到数据时自动调用。客户端的数据传输功能主要在该事件中完成。主要 功能如下：

（1）连接服务器

```
ClientSocket. Address: = 服务器 IP
ClientSocket. Active: = true;
```

（2）读取服务器传送的数据

```
cmdtempstring: = trim( Socket. ReceiveText) ;
```

（3）分析数据，提取通信指令类型

（4）根据指令类型完成相应功能

5　结束语

文章针对现有企业发展要求，提出了一种基于 TCP/IP 的大型文件断点续传系统的解决方案，并在湖北水路交通规费征稽系统中以得到广泛使用。该方案不仅适用于传输条件好的专线网络，而且适用于传输速率低的电话拨号（PSTN）。系统解决了大型文件的传输，多用户并发，断点续传等难题，满足了稳定、可靠、高效的传输要求。

参考文献

1　蒋东兴，Windows Socket 程序设计指南 北京：清华大学出版社，1995.

2　Douglas E. Comer . 用 TCP/IP 进行网际互联（第 3 卷）－客户机 服务器编程和应用【M】。北京：电子工业出版社，2000.

3　吕伟臣，Delphi 6 网络编程，科学出版社，2002.

4　丁展、刘海英等，Visual C + + 网络通信编程实用案例精选，人民邮电出版社.

5　李腊元、李春林，计算机网络技术（第二版），国防工业出版社，2004.