Socket Programming in the Banking Collection Service Counter System

Gang WANG
Hubei Radio and TV University, Wuhan, China
Email: *wg970701@sina.com*

Abstract: This paper describes the use of TCP / IP programming development of Client / Server model of network communication procedures within the principle, and the specific methods of using it to achieve banking collection service of counter system.

Keywords: TCP/IP; sockets; Client/Server collection

Socket 编程在银行代收业务柜面系统中的应用*

汪　刚
湖北广播电视大学，武汉市，中国，430074
Email: wg970701@sina.com

【摘要】 本文讲述使用 TCP/IP 的编程开发 C/S 模式的网络通讯程序的原理,以及使用它实现银行移动代收柜面系统的具体方法。

【关键词】 TCP/IP; 套接字; C/S 代收

1 交易系统介绍

交换平台是专门为银行而开发的一套金融业务系统，随着平台的不断应用和发展，随后又开发了中间业务。它包括许多的银行业务，几乎可以说随着银行业务的发展，交换平台, 中间业务也在发展，但它们的相互关系基本没有大的变化。具体关系如图 1：

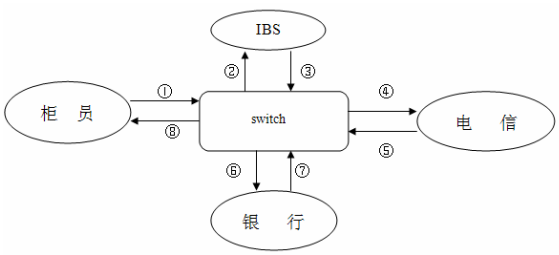

Figure 1. Interchange business switch

图 1. 交换平台

IBS: 交换平台
SWITCH: 中间业务
电信: 电信方主机
银行: 银行主机
柜员: 柜面系统（即本系统）

简单地讲，一个交易首先由柜员方发起，将必要的信息（如交易码等）进行组包后发向交换平台①，交换平台按交易码进行处理，如进行解包，打包格式转换后，按照路由选择发送给中间业务②，中间业务经过相关处理后，将消息反馈给交换平台③，然后，交换平台将数据组包后发送到电信方④，电信方按交易码取得相关数据，并将与此交易相关的数据返回给交换平台⑤，交换平台将信息发送到银行主机⑥并获得其返回后⑦，将数据返回到柜员，由柜员进行最后的处理⑧。

对于整个系统，交换平台起核心作用，它由几个不同的模块组成分别完成格式转换和路由选择等功能。中间业务则利用交换平台的功能对银行业务进行开发。

2 Socket 的设计原理及进行通信连接的过程框图

UNIX 操作系统在其系统内核中实现了 TCP/IP 协

*基金说明：NFD2010 会议费用 1500 元人民币已汇入湖北经济学院银行帐户，论文编号 20101063 ，作者汪刚。

议，并且提供了一个灵活的独立子系统来支持分布式环境下的进程间通讯，即“sockets interface”(套接字接口)。TCP/IP 套接字技术是 TCP/IP 网络编程的标准，是进程间通讯的基础。具体来说，socket 接口是 TCP/IP 网络的 API，它为程序员提供函数或例程来开发 TCP/IP 网络的应用程序。最初 socket 接口的设计者将接口放在 UNIX 操作系统中，但在非 UNIX 的其它环境中，如 Windows, JAVA 等，程序员可以通过调用 socket 库例程来完成 TCP/IP 网络通讯。

图 2 给出了面向连接数据通信的典型时序——首先启动 server，然后在某一时刻启动与 server 连接的 client。

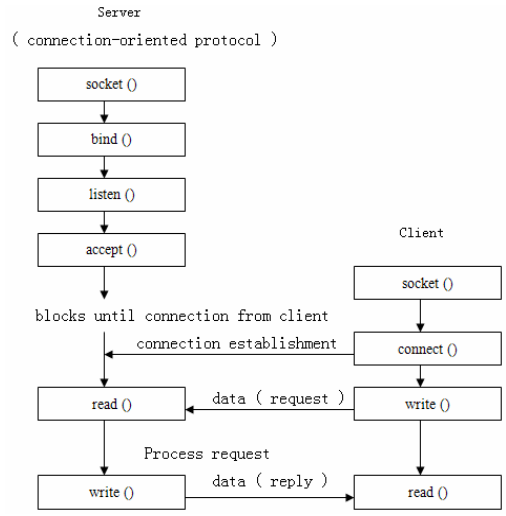

Figure 2. Typical timing of connection－oriented communication

图 2. 面向连接数据通信的典型时序

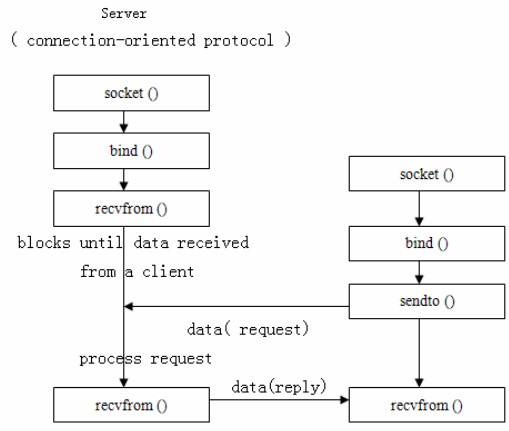

Figure 3. Typical timing of connectionless communication

图 3. 非连接协议的典型时序

对于使用非连接协议的客户服务器，系统调用是不同的。图 3 给出这些系统调用。

面向连接的协议提供了流量控制，而非面向连接的协议则不提供流量控制。这样，当非面向连接的数据报客户由于发送数据太快而使得缓冲区不够时，发送者必须重复发送数据。仅因为这一原因，我们建议使用面向连接的协议。

3 基本的套接字系统调用

3.1 socket()系统调用

socket 是不同主机间的进程进行双向通讯的端点，应用进程要在网络间通讯，必须在网络的两端分别建立起一个 socket，以指明将要使用的通信协议类型（Internet TCP、UDP、XNS、SPP 等）。系统调用 socket（）即用于建立一个 socket，其调用格式为：

socket_id=socket(address_family,type, protocol) ;

3.2 connect()系统调用

使用 Client/Server 模式的网络通讯包括两个端点，其中启动网络服务请求的端点为 client 方,对客户请求作出应答的端点为 server 方。

在 client 方，使用 connect()函数在 socket 结构中保存本地和远地端口的信息，启动和远地主机的直连。其调用格式为：

int connect(int socket_id, struct sockaddr *dest_addr, int addrlen);

3.3 bind()系统调用

在 server 方，要使 socket 对特定的协议端口进行侦听，必须使用 bind()函数。bind()有三种用途：

1) server 在系统中公布其地址;

2) 客户可以为自己登记一个专用地址;

3) 非连接协议的 client 需保证系统给它分配了某一唯一地址，使得其它端点(server)能通过该合法的返回地址给它发送应答信息。

其调用格式为：

bind(int socket_id, struct sockaddr *local_addr, int addrlen);

3.4 listen 系统调用

server 方使用 listen()函数使 socket 处于被动的侦听模式，并将来自 client 的所有服务请求在一个请求

978-1-935068-12-9 © 2010 SciRes.

对列上排队。其调用格式为：

listen(socket_id,quelen);

其中：

socket_id 是本地 socket 号。

quelen 是请求队列的长度，最大长度为 5。

3.5 accept 系统调用

accept()函数使进程在准备好接收来自 client 的服务请求后，进入睡眠状态，直到某一请求到达或进程被某一信号中断，accept()调用才返回。当服务请求到达 accept 函数监视的 socket 时，accept（）生成一个新的 socket，并返回 client 方的 sockaddr_in 结构变量，从而在 server 方应用程序中使用这个被赋予 client 方地址的 socket 同 client 方的 socket 进行连接。其调用格式如下：

new_sock=accept(socket_id,client_addr,addrlen);

new_sock 为一个 socket 描述符。

accept（）自动建立一个新的套接字描述符，并把当前服务器作为并发服务器来处理。如果是这种情况，则典型状况是：

```
int sockfd , newsockfd ;
if ( (sockfd = socket( … )) < 0 )
err_sys( “socket error “ );
if (bind( sockfd, ...) < 0)
err_sys( “bind error “);
if (listen( sockfd , 5 ) < 0)
err_sys(“listen error “);
for ( ; ; )
{
newsockfd = accept( sockfd , ...);     /* blocks */
if ( nesockfd < 0 ) err_sys( “accept error “);
if ( fork() == 0 )
{ close( sockfd );      /* child */
doit( newsockfd ); /* process the request */
exit(0);
  }
close( newsockfd );  /* parent */
}
```

当收一个连接请求并接受它时,进程复制其自身生成一子进程，然后让子进程来为该特定的连接服务，而父进程则转而等待另一个连接请求。

如果 accept 的调用者想要一个重复服务器，那么情况将会如下面的描述：

```
int sockfd, newsockfd ;
if ( ( sockfd = socket( … ) ) < 0 )
 err_sys( “socket error “ );
if ( bind( sockfd, ...) < 0 )
err_sys( “bind error “);
if (listen( sockfd , 5 ) < 0 )
err_sys( “listen error “);
for ( ; ; )
{newsockfd = accept( sockfd , ...);    /* blocks */
if ( nesockfd < 0 )     err_sys( “accept error “);
doit( newsockfd );    /* process the request */
close( newsockfd );
}
```

这里服务器在使用套接字描述符 newsockfd 处理请求后用 close 终止连接，然后再用原始描述符 sockfd 等待另一连接。

4 数据传输

当连接建立后，通常采用 write()和 read()函数进行数据传输。write()的调用格式为：

write(socket_id,buffer,bufferlen)

其中 buffer 为指向发送缓冲区的指针。read()的调用格式与 write()相同。

释放 socket

close(socket_id)函数将释放一个 socket 所占用的空间。

以上所有系统调用均包括在 libsocket.a 中，如果程序员在 PC 上的 UNIX 系统中进行 socket 编程，则在编译时需加上“-llibsocket”参数，而在 AIX 中则无需这样做。

以下是本系统的一些通讯程序。

```
/***********client.c***********/
/* client 端 to server 端 发送与接收
    sendbuf:     发送缓冲区
    receivebuf:接收缓冲区
 return:       0: 单包 Ok
              -1:通讯故障
              -2:交易失败
               1:多包 ok        */
int clienttoserver( char *sendbuf, char *receivebuf,
```

```
int *pid)
    {    FILE   *fptcp;
    char   tcpaddress[32],tcpport[32];
    char   Dir[128];
    int    socket_id,ret;
    struct sockaddr_in sin;
    unsigned char STAT[3];
    struct sjtype *rcv_buf;
    char swap[4],buf[LEN];
    char pbuf[224];
    char *rbuf,fname[31];
    int   i,j=1, n;
    short k=0,h=0;
    FILE *fpp;
    FILE *fp;

sprintf(Dir,"%s/lib/tcpip.cfg",getenv("HOME"));
        if ( ( fptcp=fopen(Dir,"r") ) == NULL)
    { cwtst1("不能打开通讯配置文件!");    return  -1;
    }
        while ( fgets(buf,sizeof(buf),fptcp) != NULL)
    if (strstr(buf,"HostAddress") != NULL)
    {sscanf(buf,"%*s%s%s", tcpaddress,tcpport);
     break;
    }
    fclose(fptcp);
    bzero((char *)&sin,sizeof(sin));
    sin.sin_family    = AF_INET;
    sin.sin_port      = htons( atoi(tcpport) );
    sin.sin_addr.s_addr = inet_addr(tcpaddress);
    /*  建立  socket */
    socket_id=opentcpip (&sin);
    switch(socket_id)
    {
    case -1:ErrorMsg("Error create socket");
            return -1;
    case -2:ErrorMsg("Error create link");
            return -1;
    default:break;
    }
    /*  发送数据包  */
    if ( sendpacket(socket_id, sendbuf ) < 0 )
    { ErrorMsg("通讯失败   Error send data ");
    closetcpip(socket_id);
    return -1;
    }
    {FILE *fp;
    fp = fopen("123","w+");
    fprintf(fp,"sndbuf = [%s]\n", sendbuf);
    fclose(fp);
    }
    /*  接收数据包  */
    if(receivepacket(socket_id,receivebuf,        RE-
CEIVETIMESPAN ) < 0 )
    { ErrorMsg("Error receive data ");
    closetcpip(socket_id);
    return -1;
    }
    closetcpip(socket_id);
    return 0;              /*返回正确的单包*/
    }
```

除此以外，还有许多相关程序，在这里我们就不一一详述，具体包括：

```
int Clienttoserver(sendbuf, receivebuf, pid);
int SetTiaHeader(sndbuf);
Sendtohost(sndbuf,length,temid);
Receivefromhost(buf,temid);
UnEncryptHostStr(src, srclen, dest, destoffs);
int StatJug(unsigned char *STAT, char *buf);
```

5 结束语

Sockets 的应用范围已不再局限于 Unix 操作系统和 TCP/IP 网络,本文主要介绍是基于 Unix 上的 Socket 编程,所有示例程序都在 Sco 上编译通过,并且运行正常。

References (参考文献)

[1] Luo Yafe.The Multi-thread Communication of Socket Based on TCP, Computer Knowledge and Technology, 2009 No. 03.
罗亚非.基于 TCP 的 Socket 多线程通信,电脑知识与技术 2009 年 第 03 期.

[2] Warren W. Gay forward, ZHAN Jun Hu, Yu Wei translation, Actual Linux Socket Programming [M]. Xi'an: Xidian University Press, 2002.
Warren W.Gay 著, 詹俊鹃,于卫译, 实战 Linux Socket 编程[M]. 西安: 西安电子科技大学出版社, 2002.

[3] Motorola, Inc. MPC852TTS / D. PDF [M / CD] REV.1.3, 2003, 4.

[4] Motorola, Inc. MPC852TEC. PDF [M / CD] REV. 2.0, 2003, 12.
[5] Zou Yi. Embedded Linux Design and Application [M]. Beijing: Tsinghua University Press, second edition in April 2002.
邹思轶. 嵌入式 Linux 设计与应用[M].北京: 清华大学出版社 2002 年 4 月第二版.
[6] Hu Hongchao, bayberry more. MPC8260-based embedded system testing, Micro-computer Information, 2006 Vol. 22, No. 2.
扈红超, 杨梅越. 基于 MPC8260 嵌入式系统的测试, 微计算机信息, 2006 Vol. 22, No. 2.
[7] ZhuJinrong.MCP environment Socket programming and application, Financial Computer of Huanan, 2004 12 (8).
朱金荣. MCP 环境下的 Socket 编程与应用, 华南金融电脑, 2004 12(8).