厦門大學

信息学院软件工程系

《计算机网络》实验报告

题　　目 实验六 利用 Socket API 实现网上点对点通信

班　　级 软件工程 2018 级 1 班

姓　　名 王薪蕾

学　　号 24320182203285

实验时间 2020 年 4 月 22 日

2020 年　4 月　22 日

1 实验目的

在 Windows 或 Linux 操作系统（也可以将客户端部署在 Android、iOS 或 WinPhone 手机）下，分别基于 TCP 和 UDP 协议，利用 Socket API 实现网上点对点通信。
程序一“基于 TCP 的可靠文件传输”，功能包括：
在客户端，用户选择本地的某个文件，并发送到服务器端。
在服务器端，接收客户端传输的数据流，并按 IP 地址保存在服务器端（文件名重复的，可以覆盖）。
如果传输过程中服务器端发现客户端断开，服务器端应删除文件，并在屏幕上提示，如“IP：1.2.3.4 发来 abcd.txt 文件过程中失去连接。”。如果客户端发现服务器端不工作，客户端应有提示“服务器 1.2.3.5:62345 失去连接”。
程序二“基于 UDP 的不可靠文件传输”，功能同上，但不能使用 TCP 协议进行传输。考虑如果传输过程中服务器端、客户端如何发现断开。

2 实验环境

操作系统：windows，编程语言：C++

3 实验结果

程序一：

发送时有两种数据包，文件信息包（文件名和文件大小）和文件数据包（包 id，数据）。前面放一个字符来区分包信息’n’,’d’。

接收时也按包信息分开处理。

下图为用来创建包和分析包的类：Frame

```
class Frame
{
public:
    Frame() { ... }
    /* ... */
    void createFrame(int& packageId, char d[]) { ... }
    /* ... */
    void createFrame( char name[], int & size) { ... }
    //文件信息
    void getFile(char name[], int & size)
    {
        int end=find_(msg,1);
        int s = strlen(msg);
        size = strToInt(msg+1);
        for (int i = 0; i <s-end ; i++)
            name[i] = msg[end + i];
        name[s - end] = '\0';
    }
    //数据帧
    void getData(int& pId)
    {
        int s = strlen(msg);
        int end = find_(msg, 1);
        pId = strToInt(msg + 1);
        for (int i = 0; i < s - end+2; i++)
            data_[i] = msg[end + i];
    }
};
```

传输的内容为 msg 中；文件的读入与写入数据则放在 data_数组中，msg 比 data_要大一些。

```
#define MAX_SIZE 65535
#define DATA_SIZE 65000

WSADATA wsaData;
SOCKET ser_sockfd;/*创建连接的SOCKET */
SOCKET cli_sockfd;/*创建连接的SOCKET */
sockaddr_in ser_addr; /* 服务端地址信息 */
sockaddr_in cli_addr; /* 客户端地址信息 */
int ser_port=1024, cli_port=1025,iSize;
char msg[MAX_SIZE];/* 缓冲区*/
char data_[DATA_SIZE];
```

选择服务内容并输入好信息：

```
printf("%s\n", "请输入选项，1.发送文件、2.接收文件。");
scanf_s("%d", &iOption);

// 发送文件
if (iOption == 1)
{

fileName:
    printf("%s\n", "请输入需要发送的路径文件名。\nWindows 路径文件名格式：\t\tC:\\inst
    cin >> acDirAndFileName;
    fopen_s(&pFile, acDirAndFileName, "rb");
    if (pFile == NULL)
    {
        printf("%s\n", "读取文件失败，请重新输入文件名。");
        goto fileName;
    }
    // 关闭文件
    fclose(pFile);


    printf("%s\n", "请输入客户端口号:");
    cin >> cli_port;
    printf("%s\n", "请输入服务端口号:");
    cin >> ser_port;

    printf("%s\n", "请输入接收文件方的 IP 地址，不能有空格。\n例如：\n192.168.1.104");
    cin >> acIpAddr;

    sendFile(acDirAndFileName, acIpAddr);
}
// 接收文件
else if (iOption == 2){ ... }
```

打开服务端：

```
void openSer(char *cdr)
{
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
        printf("WSAStartup() error!");

    ser_sockfd = socket(PF_INET, SOCK_STREAM, 0);
    if (ser_sockfd < 0){ ... }

    /* 初始化服务器地址*/
    ser_addr.sin_family = AF_INET;
    ser_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    ser_addr.sin_port = htons(ser_port);
    if (bind(ser_sockfd, (SOCKADDR*)&ser_addr, sizeof(sockaddr_in)) < 0)
    { /*绑定失败 */
        printf("绑定失败\n");
        exit(1);
    }
    /*侦听客户端请求*/
    if (listen(ser_sockfd, BACKLOG) < 0)
    {
        printf("没有客户端请求\n");
        closesocket(ser_sockfd);
        exit(1);
    }
}
```

打开客户端：

```
void openCli(char* sdr)
{
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
        printf("WSAStartup() error!");

    cli_sockfd = socket(PF_INET, SOCK_STREAM, 0);
    if (cli_sockfd < 0)
    {/*创建失败 */
        printf("创建失败\n");
        getchar();
        exit(1);
    }

    /* 初始化客户端地址*/
    int addrlen = sizeof(struct sockaddr_in);
    cli_addr.sin_family = AF_INET;
    cli_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    cli_addr.sin_port = cli_port;
    if (bind(cli_sockfd, (struct sockaddr*)&cli_addr, addrlen) < 0)
    {
        /*绑定失败 */
        printf("Bind Error:\n");
        getchar();
        exit(1);
    }
    /* 初始化服务器地址*/
    addrlen = sizeof(struct sockaddr_in);
    ser_addr.sin_family = AF_INET;
    inet_pton(AF_INET,sdr,&ser_addr.sin_addr.s_addr);
    ser_addr.sin_port = htons(ser_port);
    printf("连接服务端请求\n");
    while (connect(cli_sockfd, (struct sockaddr*)&ser_addr, addrlen) != 0)
接*/
    {
        /*连接失败 */
        printf("Connect Errory\n");
        Sleep(10);
```

客户端和服务端的信息处理

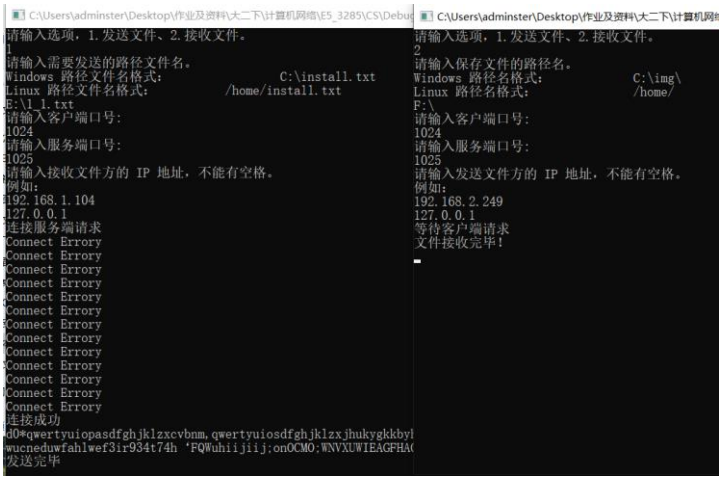
```
//服务端
void recvFile(char* acDirName, char *cdr) { ... }
//客户端
void sendFile(char* acDirName,char *sdr) { ... }
```

客户端：Frame 类封装文件信息包和数据包，发送给服务端。

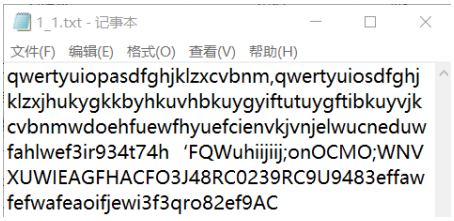
服务端：收到信息包，Frame 类拆分文件信息包和数据包，写入文件。

目前测试 .txt .xml .bin 可以正常传输，但 .jpg .docx 失败。

实验部分截图



发送的文件



接收的文件

.jpg .docx 可以传输但文件打不开，应该是由于读取写入的方式不正确等原因导致？读出来时显示的.jpg 显示“？？” .docx 显示的是一些方框。

程序二：

客户端发送服务端时有两种数据包，文件信息包（文件名和文件大小）和文件数据包（包 id，数据）。前面放一个字符来区分包信息’n’,’d’。

服务端发给客户端两种信号’n’,’d+包 id’，获取文件信息包或者数据包

创建包和分析包的类：Frame

打开 UDP

```
void openUDP(char* acIpAddr)
{
#ifdef _MSC_VER
    // Winsows 启用 socket
    WSADATA wsadata;
    if (WSAStartup(MAKEWORD(1, 1), &wsadata) == SOCKET_ERROR)
    {
        printf("启用 socket 失败\n");
        exit(0);
    }
#endif

    // 新建 socket
    if ((iUDP = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
        printf("新建 socket 失败\n");
        exit(0);
    }

    // 清零
    memset(&serverAddr, 0, sizeof(serverAddr));
    memset(&clientAddr, 0, sizeof(clientAddr));

    // 设置协议 IP 地址及 Port
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(iServerPort);
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    clientAddr.sin_family = AF_INET;
    clientAddr.sin_port = htons(iClientPort);
    inet_pton(AF_INET, acIpAddr, &clientAddr.sin_addr.s_addr);
    // 绑定端口，监听端口
    if (bind(iUDP, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == -1)
    {
        getchar();
```

发送时：

服务端请求文件信息‘n’→客户端

客户端文件信息包→服务端

服务端请求文件数据‘d+包 id‘→服务端

服务端请求文件数据‘d+包 id‘→服务端

。 。 。 。 。

直到文件数据读取完毕。

其中接受和发送文件信息写成一个线程，当服务端和客户端都收到对方信息时才进行文件数据的获取和接收。

```
// 接收文件名
DWORD WINAPI recvName(LPVOID p)
{
    int iAddrSize = sizeof(remoteAddr);
    acReq[0] = 'n'; acReq[1] = '\0';
    acRecvStr[0] = '\0';
    printf("%s\n", "正在发送请求信息！");
    // 发送请求信息
    sendto(iUDP, acReq, strlen(acReq), 0, (struct sockaddr*)&clientAddr, sizeof(clientAddr));
    // 每次发送请求信息后等待一段时间
    printf("send:%s\n", acReq);
    sleepUDP(10);
    // 接收文件名
    iSize = recvfrom(iUDP, acRecvStr, SIZEB, 0, (struct sockaddr*)&remoteAddr, &iAddrSize);
    printf("recv:%s\n", acRecvStr);
    return 0;
}

// 发送文件名
DWORD WINAPI  sendName(LPVOID p)
{
    int iAddrSize = sizeof(remoteAddr);
    acRecvStr[0] = '\0';
    // 接收请求
    printf("%s\n", "正在接收请求信息！");
    recvfrom(iUDP, acRecvStr, 5, 0, (struct sockaddr*)&remoteAddr, &iAddrSize);
    printf("recv:%s\n", acRecvStr);
    // 每次接收请求信息后等待一段时间
    sleepUDP(10);
    return 0;
}
```

4 实验总结

学会编写 TCP，UDP 传输的客户端和服务端，实现部分文件的传输。

有些疑问，写程序时出现了自己发送自己接收的问题，即客户端发送的信息被客户端收到，换了一种写法之后就没有这样的问题，但之前的问题并没有解决。