
廈門大學



软件学院

《计算机网络》实验报告

题目 利用 Socket API 实现网上点对点通信

姓名 黄勳

学号 22920212204392

班级 计算机网络 2021 级 2 班

实验时间 2023. 5. 19

2023 年 5 月 19 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。

2 实验环境

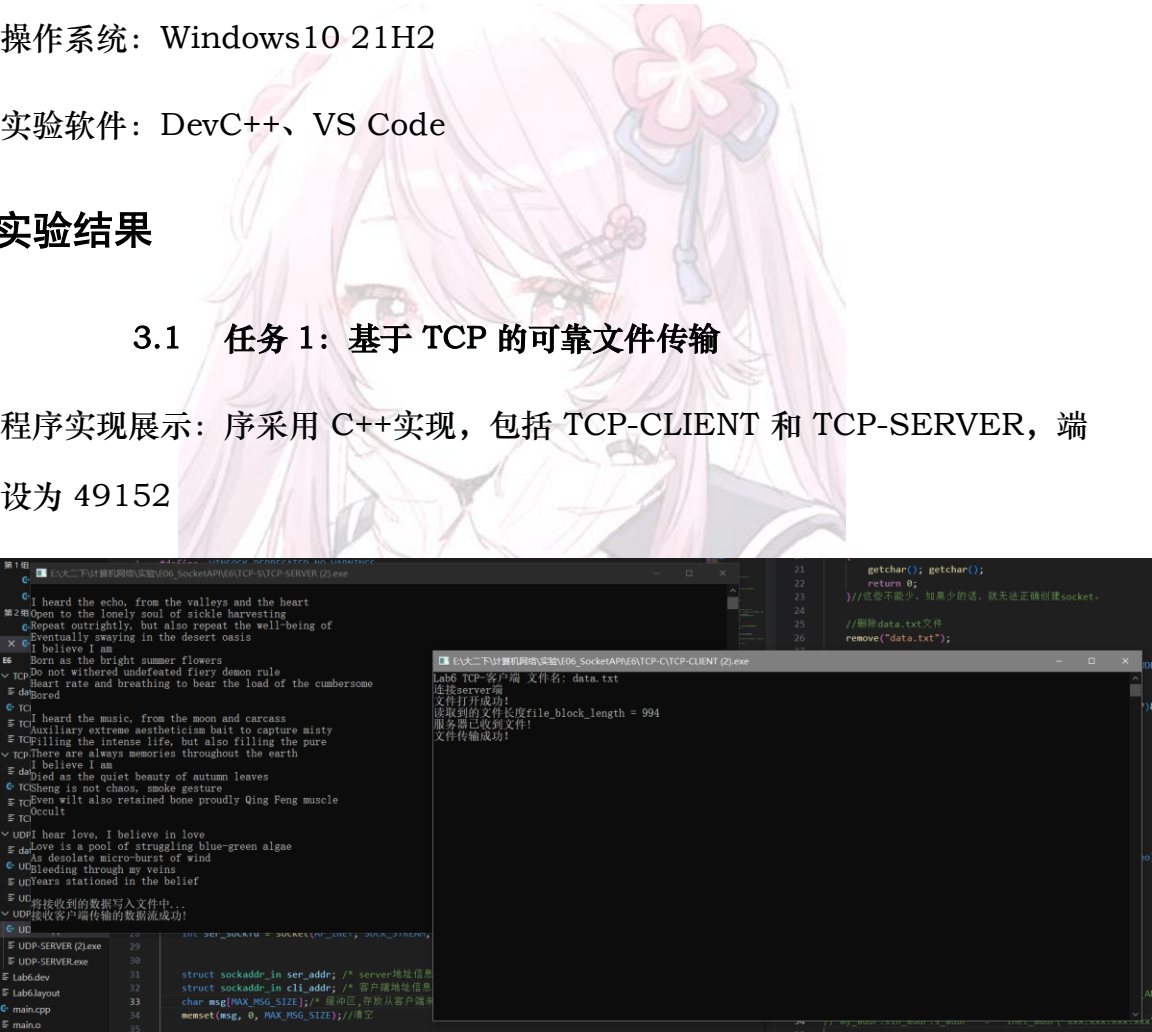
操作系统：Windows10 21H2

实验软件：DevC++、VS Code

3 实验结果

3.1 任务 1：基于 TCP 的可靠文件传输




程序实现展示：序采用 C++实现，包括 TCP-CLIENT 和 TCP-SERVER，端口号设为 49152



server 端接收到的 data.txt

实验 > E06_SocketAPI > E6 > TCP-S

在 TCP-S 中搜索

名称	修改日期	类型	大小
 data.txt	2023/5/19 0:12	文本文档	2 KB
 TCP-S.cpp	2023/5/19 0:07	C++ Source File	4 KB
 TCP-SERVER.exe	2023/5/18 23:09	应用程序	2,887 KB

用任务管理器 kill 掉 client 端后，server 端监测到断开

```
E:\大二下\计算机网络\实验\E06_SocketAPI\E6\TCP-S\TCP-SERVER.exe
I heard the echo, from the valleys and the heart
Open to the lonely soul of sickle harvesting
Repeat outrightly, but also repeat the well-being of
Eventually swaying in the desert oasis
I believe I am
Born as the bright summer flowers
Do not withered undefeated fiery demon rule
Heart rate and breathing to bear the load of the cumbersome
Bored

I heard the music, from the moon and carcass
Auxiliary extreme aestheticism bait to capture misty
Filling the intense life, but also filling the pure
There are always memories throughout the earth
I believe I am
Died as the quiet beauty of autumn leaves
Sheng is not chaos, smoke gesture
Even wilt also retained bone proudly Qing Feng muscle
Occult

I hear love, I believe in love
Love is a pool of struggling blue-green algae
As desolate micro-burst of wind
Bleeding through my veins
Years stationed in the belief

将接收到的数据写入文件中...
接收客户端传输的数据流成功!
客户端 IP: 127.0.0.1 过程中失去连接。
```

一、编写 TCP 客户端相关代码

定义最大字符长度 1024，端口号 49152

```
TCP-C > TCP-C.cpp > main(int, char * [])
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include<winsock.h>
4  #include <string.h>
5  #include<time.h>
6  #include<iostream>
7  #define MAX_MSG_SIZE 1024
8  #define SERVER_PORT 49152
9  #define BACKLOG 5
10 char recData[MAX_MSG_SIZE];
11 char sendData[MAX_MSG_SIZE];
12 #pragma comment(lib, "ws2_32.lib")
13 using namespace std;
14 int main(int argc, char* argv[])
```

调用 socket 建立连接（主动打开）



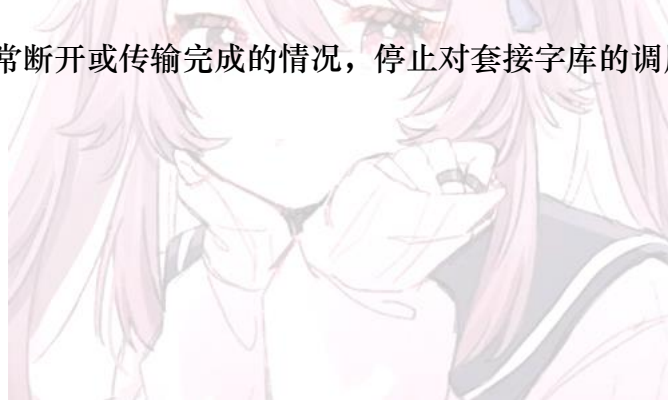
```
printf("Lab6 TCP-客户端 文件名: data.txt\n");
WORD sockVersion = MAKEWORD(2, 2);
//创建套接字, 返回一个描述符, 这个描述符是一个整数, 是套接字在系统中的唯一标识符, 后
WSADATA data;
if (WSAStartup(sockVersion, &data) != 0)
{
    getchar(); getchar();
    return 0;
} //上面这几句必不可少, 否则无法创建socket插口, 即socket()会报错。

SOCKET sclient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
//客户端的插口
if (sclient == INVALID_SOCKET)
{
    printf("invalid socket!");
    getchar(); getchar();
    return 0;
}
sockaddr_in serAddr;
serAddr.sin_family = AF_INET;
serAddr.sin_port = htons(SERVER_PORT);
serAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1"); //ip保留字段, 本机地址
//这就是要准备好server端的信息, 便于插口去连接。
printf("连接server端\n");
if (connect(sclient, (sockaddr*)&serAddr, sizeof(serAddr)) == SOCKET_ERROR)
{
    printf("connect error !");
    closesocket(sclient);
    getchar(); getchar();
    return 0;
}
```

打开文件并读入 sendData 中

```
FILE* fp;
if ((fp = fopen("data.txt", "r"))== NULL)
{
    printf("文件不存在! \n");
}
else
{
    printf("文件打开成功! \n");
    //bzero(buffer, BUFFER_SIZE);
    memset(sendData, 0, MAX_MSG_SIZE);
    int file_block_length = 0;
    //循环将文件file_name(fp)中的内容读取到sendData中
    int i = 0;
    while ((file_block_length = fread(sendData, sizeof(char), MAX_MSG_SIZE, fp)) > 0)
    {
        //测试
        i++;
        if (i == 2) {
            closesocket(sclient);
            WSACleanup(); //终止对套接字库的使用。
            getchar(); getchar();
            return 0;
        }
        printf("读取到的文件长度file_block_length = %d\n", file_block_length);
    }
}
```

处理连接异常断开或传输完成的情况，停止对套接字库的调用



```

        else if (ret <= 0) {
            printf("服务器 127.0.0.1:49152 失去连接!\n");
            getchar(); getchar();
            exit(-1);
        }
    }
    fclose(fp);          //关闭文件描述符fp
    printf("文件传输成功!\n");
}

int ret = recv(sclient, recData, MAX_MSG_SIZE, 0); //返回的是收到的实际字节数
if (ret > 0)
{
    printf(recData);
    printf("\n");
}
else if (ret <= 0) {
    printf("服务器 127.0.0.1:49152 失去连接!\n");
    getchar(); getchar();
    exit(-1);
}

closesocket(sclient);
WSACleanup(); //终止对套接字库的使用。
getchar(); getchar();
return 0;

```

二、TCP-SERVER 相关代码

定义最大字符长度 1024，端口号 49152，与 client 相同

```

TCP-S > TCP-S.cpp > main()
1  #define _WINSOCK_DEPRECATED_NO_WARNINGS
2  #define _CRT_SECURE_NO_WARNINGS
3  #include <stdio.h>
4  #include <winsock2.h>
5  #include <iostream>
6  #define MAX_MSG_SIZE 1024
7  #define SERVER_PORT 49152
8  #define BACKLOG 5
9  #pragma comment(lib, "Ws2_32.lib")
10 using namespace std;
11 int main()

```

调用 socket 建立连接（被动打开）

```
printf("Lab6 TCP-服务器\n");
WORD version(0);
WSADATA wsadata;
int socket_return(0);
version = MAKEWORD(2, 0);
socket_return = WSStartup(version, &wsadata); //开始对套接字库的使用。
if (socket_return != 0)
{
    getchar(); getchar();
    return 0;
} //这些不能少，如果少的话，就无法正确创建socket。

//删除data.txt文件
remove("data.txt");

int ser_sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); /*创建连接的SOCKET */

struct sockaddr_in ser_addr; /* server地址信息 */
struct sockaddr_in cli_addr; /* 客户端地址信息 */
char msg[MAX_MSG_SIZE]; /* 缓冲区,存放从客户端来的请求*/
memset(msg, 0, MAX_MSG_SIZE); //清空

if (ser_sockfd < 0)
{ /*创建失败 */
    fprintf(stderr, "socket Error:%s\n", strerror(errno));
    getchar(); getchar();
    exit(1);
}
/* 初始化服务器地址*/
int addrlen = sizeof(struct sockaddr_in);
memset(&ser_addr, 0, addrlen);
ser_addr.sin_family = AF_INET;
ser_addr.sin_addr.s_addr = htonl(INADDR_ANY); //INADDR_ANY 就是本机的所有ip
```

绑定端口


```

ser_addr.sin_family = AF_INET;
ser_addr.sin_addr.s_addr = htonl(INADDR_ANY); //INADDR_ANY 就是本机的所有ip
如果要手动指定ip可以这么使用
my_addr.sin_addr.s_addr = inet_addr("xxx.xxx.xxx.xxx"); inet_addr用于把32位的ip地址转

ser_addr.sin_port = htons(SERVER_PORT); //为这个server进程指定端口号
if (bind(ser_sockfd, (struct sockaddr*)&ser_addr, sizeof(struct sockaddr_in))
    < 0)
{ /*绑定失败 */
    fprintf(stderr, "Bind Error:%s\n", strerror(errno));
    getchar(); getchar();
    exit(1);
}
printf("服务端打开成功，等待连接客户端...\n");

```

“死循环” 等待客户端建立连接，直到传输完毕

```

while (1)
{ /* 等待接收客户连接请求 */

    //int cli_sockfd = accept(ser_sockfd, (struct sockaddr*)&cli_addr,&addrlen); //传
    // printf("%d",cli_sockfd);
    if (cli_sockfd <= 0)
    {
        fprintf(stderr, "Accept Error:%s\n", strerror(errno));
    }
    else
    { /*开始服务*/

        /* 接受数据 */
        memset(msg, 0, MAX_MSG_SIZE);
        int i=recv(cli_sockfd, msg, MAX_MSG_SIZE, 0);
        if (i <=0) {
            cerr << "客户端 IP: 127.0.0.1 过程中失去连接。" << endl;
            remove("data.txt");
            getchar(); getchar();
            exit(-1);
        }
        printf("接收到客户端连接请求 from %s\n", inet_ntoa(cli_addr.sin_addr));
        printf("收到客户端发送的数据: \n");
        printf("%s\n", msg); /*在屏幕上打印出来 */

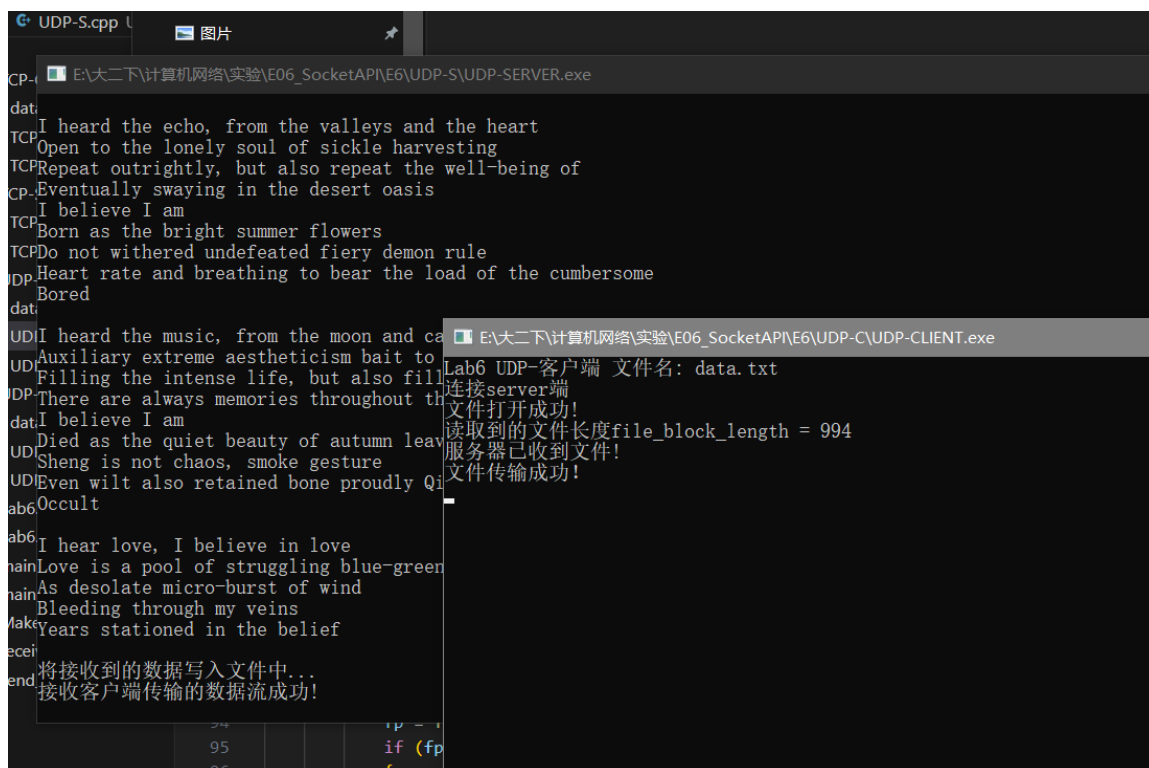
        FILE* fp;
        fp = fopen("data.txt", "a+");
        if (fp == NULL)
        {
            printf("File Can Not Open To Write!\n");
            _exit(-1);
        }
        printf("将接收到的数据写入文件中...\n");
    }
}

```

3.2 任务 2：基于 UDP 的不可靠文件传输

程序采用 C++实现，包括 UDP-CLIENT 和 UDP-SERVER，端口号设为 49152

因为 UDP 是无连接/不可靠传输（事先没有三次握手建立连接，也无法获知报文是否到达目的站）。所以在 server 端设置丢包最大值和一个时延（作为 recvfrom 的等待时间），如超出则自动断开。






```
CP- E:\大二下\计算机网络\实验\E06_SocketAPI\E6\UDP-S\UDP-SERVER.exe
dat
I heard the echo, from the valleys and the heart
TCP Open to the lonely soul of sickle harvesting
TCP Repeat outrightly, but also repeat the well-being of
CP Eventually swaying in the desert oasis
CP I believe I am
TCP Born as the bright summer flowers
TCP Do not withered undefeated fiery demon rule
UDP Heart rate and breathing to bear the load of the cumbersome
UDP Bored
dat
UDP I heard the music, from the moon and ca
UDP Auxiliary extreme aestheticism bait to
UDP Filling the intense life, but also fill
UDP There are always memories throughout th
dat I believe I am
UDP Died as the quiet beauty of autumn leav
UDP Sheng is not chaos, smoke gesture
UDP Even wilt also retained bone proudly Qi
ab6 Occult
ab6 I hear love, I believe in love
main Love is a pool of struggling blue-green
main As desolate micro-burst of wind
main Bleeding through my veins
mak Years stationed in the belief
recei
end
将接收到的数据写入文件中...
接收客户端传输的数据流成功!

E:\大二下\计算机网络\实验\E06_SocketAPI\E6\UDP-C\UDP-CLIENT.exe
Lab6 UDP-客户端 文件名: data.txt
连接server端
文件打开成功!
读取到的文件长度file_block_length = 994
服务器已收到文件!
文件传输成功!
```

实验 > E06_SocketAPI > E6 > UDP-S

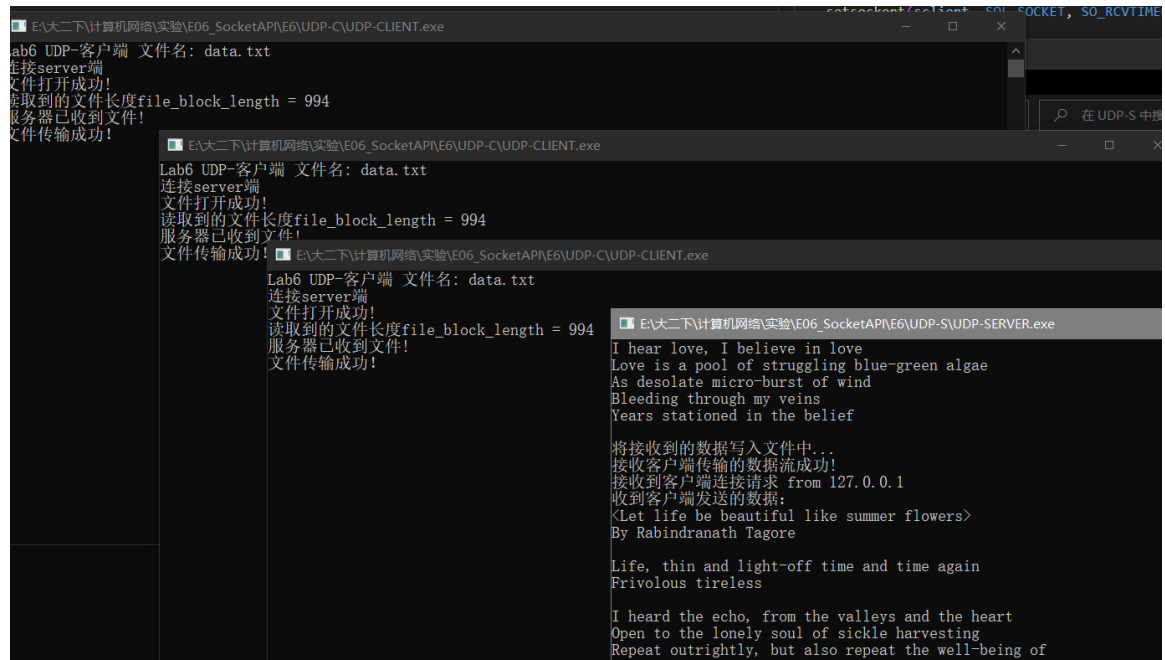
在 UDP-S 中搜索

名称	修改日期	类型	大小
 data.txt	2023/5/19 0:31	文本文档	2 KB
 UDP-S.cpp	2023/5/19 0:07	C++ Source File	4 KB
 UDP-SERVER.exe	2023/5/18 23:20	应用程序	2,886 KB

超过 10s，server 显示断开（注意此时程序认为“传输未完成”，data.txt 会被删掉）。

```
将接收到的数据写入文件中...
接收客户端传输的数据流成功!
客户端 IP: 127.0.0.1 过程中失去连接。
```

可以在 UDP-SERVER 打开时多次运行 UDP-CLIENT (间隔不要大于 10s)



```
E:\大二下\计算机网络\实验\E06_SocketAPI\E6_UDP-C\UDP-CLIENT.exe
Lab6 UDP-客户端 文件名: data.txt
连接server端
文件打开成功!
读取到的文件长度file_block_length = 994
服务器已收到文件!
文件传输成功!

E:\大二下\计算机网络\实验\E06_SocketAPI\E6_UDP-C\UDP-CLIENT.exe
Lab6 UDP-客户端 文件名: data.txt
连接server端
文件打开成功!
读取到的文件长度file_block_length = 994
服务器已收到文件!
文件传输成功!

E:\大二下\计算机网络\实验\E06_SocketAPI\E6_UDP-C\UDP-CLIENT.exe
Lab6 UDP-客户端 文件名: data.txt
连接server端
文件打开成功!
读取到的文件长度file_block_length = 994
服务器已收到文件!
文件传输成功!

E:\大二下\计算机网络\实验\E06_SocketAPI\E6_UDP-S\UDP-SERVER.exe
I hear love, I believe in love
Love is a pool of struggling blue-green algae
As desolate micro-burst of wind
Bleeding through my veins
Years stationed in the belief

将接收到的数据写入文件中...
接收客户端传输的数据流成功!
接收客户端连接请求 from 127.0.0.1
收到客户端发送的数据:
<Let life be beautiful like summer flowers>
By Rabindranath Tagore

Life, thin and light-off time and time again
Frivolous tireless

I heard the echo, from the valleys and the heart
Open to the lonely soul of sickle harvesting
Repeat outrightly, but also repeat the well-being of
But also repeat the well-being of
```

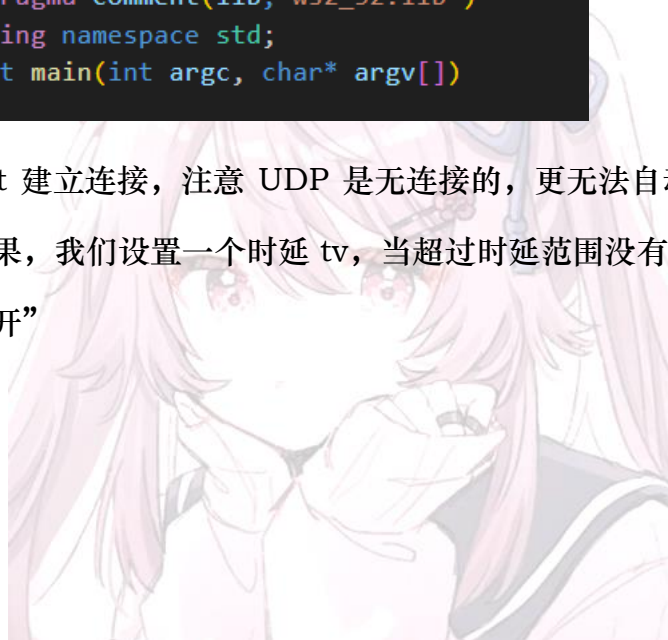
可以看到 SERVER 收到了多份数据内容

三、UDP-CLIENT 相关代码

定义最大字符长度 1024, 端口号 49152

```
UDP-C > UDP-C.cpp X  UDP-S.cpp
UDP-C > UDP-C.cpp > main(int, char * [])
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include<winsock.h>
4  #include <string.h>
5  #include<time.h>
6  #include<iostream>
7  #define MAX_MSG_SIZE 1024
8  #define SERVER_PORT 49152
9  #define BACKLOG 5
10 char recData[MAX_MSG_SIZE];
11 char sendData[MAX_MSG_SIZE];
12 #pragma comment(lib, "ws2_32.lib")
13 using namespace std;
14 int main(int argc, char* argv[])
15 {
```

调用 socket 建立连接，注意 UDP 是无连接的，更无法自动获知传输是否成功。为了实验效果，我们设置一个时延 tv，当超过时延范围没有收到新的 UDP 包就视为“意外断开”



```

17     WORD sockVersion = MAKEWORD(2, 2);
18     WSADATA data;
19     if (WSAStartup(sockVersion, &data) != 0)
20     {
21         getchar(); getchar();
22         return 0;
23     } //上面这几句必不可少，否则无法创建socket插口，即socket（）会报错。
24
25
26
27     SOCKET sclient = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
28     //客户端的插口
29     if (sclient == INVALID_SOCKET)
30     {
31         printf("invalid socket !");
32         getchar(); getchar();
33         return 0;
34     }
35
36     timeval tv = { 100, 0 }; //设置recvfrom时延，timeval(x,y)代表时间x+y*10^-6
37     setsockopt(sclient, SOL_SOCKET, SO_RCVTIMEO, (char*)&tv, sizeof(timeval));
38
39     struct sockaddr_in rec_addr; /* 接收地址信息 */
40     /* memset(rec_addr, 0,); */
41     struct sockaddr_in sen_addr; /* 发送地址信息 */
42     int rec_addrlen = sizeof(struct sockaddr_in);
43     int sen_addrlen = sizeof(struct sockaddr_in);
44
45     sockaddr_in serAddr;
46     serAddr.sin_family = AF_INET;
47     serAddr.sin_port = htons(SERVER_PORT);
48     serAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1"); //ip保留字段，本机地址

```

打开文件并读入 senddata 中

```

52     FILE* fp;
53     if ((fp = fopen("data.txt", "r")) == NULL)
54     {
55         printf("文件不存在! \n");
56     }
57     else
58     {
59         printf("文件打开成功! \n");
60         //bzero(buffer, BUFFER_SIZE);
61         memset(sendData, 0, MAX_MSG_SIZE);
62         int file_block_length = 0;
63         //循环将文件file_name(fp)中的内容读取到sendData中
64         int i = 0;
65         while ((file_block_length = fread(sendData, sizeof(char), MAX_MSG_SIZE, fp)) > 0)
66         {
67
68             printf("读取到的文件长度file_block_length = %d\n", file_block_length);
69
70             // 发送sendData中的字符串到new_server_socket,实际上就是发送给服务器端
71             if (sendto(sclient, sendData, MAX_MSG_SIZE, 0, (struct sockaddr*)&serAddr, sizeof(serAd
72             {
73                 printf("文件发送失败! \n");
74                 break;
75             }
76             //清空sendData缓存区
77             memset(sendData, 0, MAX_MSG_SIZE); //bzero(buffer, sizeof(buffer));
78             memset(recData, 0, MAX_MSG_SIZE);
79             int ret = recvfrom(sclient, recData, MAX_MSG_SIZE, 0, (struct sockaddr*)&rec_addr, &rec_
80             if (ret > 0)

```

处理连接异常断开或传输完成的情况，停止对套接字库的调用

```

71         if (sendto(sclient, sendData, MAX_MSG_SIZE, 0, (struct sockaddr*)&serAddr, siz
72         {
73             printf("文件发送失败! \n");
74             break;
75         }
76         //清空sendData缓存区
77         memset(sendData, 0, MAX_MSG_SIZE); //bzero(buffer, sizeof(buffer));
78         memset(recData, 0, MAX_MSG_SIZE);
79         int ret = recvfrom(sclient, recData, MAX_MSG_SIZE, 0, (struct sockaddr*)&rec_a
80         if (ret > 0)
81         {
82
83             printf(recData);
84             printf("\n");
85         }
86         else if (ret <= 0) {
87             printf("服务器 127.0.0.1:49152 失去连接! \n");
88             getchar(); getchar();
89             exit(-1);
90         }
91     }

```

四、UDP-SERVER 相关代码

定义最大字符长度 1024，端口号 49152，与 client 相同

```
UDP-C.cpp  UDP-S.cpp X
UDP-S > UDP-S.cpp > main()
1  #define _WINSOCK_DEPRECATED_NO_WARNINGS
2  #define _CRT_SECURE_NO_WARNINGS
3  #include<stdio.h>
4  #include <winsock2.h>
5  #include<iostream>
6  #define MAX_MSG_SIZE 1024
7  #define SERVER_PORT 49152
8  #define BACKLOG 5
9  #pragma comment(lib, "Ws2_32.lib")
10 using namespace std;
11 int main()
12
```

调用 socket 建立连接（被动打开）

```
14  WORD version(0);
15  WSADATA wsadata;
16  int socket_return(0);
17  version = MAKEWORD(2, 0);
18  socket_return = WSASStartup(version, &wsadata); //开始对套接字库的使用。
19  if (socket_return != 0)
20  {
21      getchar(); getchar();
22      return 0;
23  } //这些不能少，如果少的话，就无法正确创建socket。
24
25  //删除data.txt文件
26  remove("data.txt");
27
28  int ser_sockfd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP); /*创建连接的SOCKET */
29
30  timeval tv = { 100, 0 };
31  setsockopt(ser_sockfd, SOL_SOCKET, SO_RCVTIMEO, (char*)&tv, sizeof(timeval));
32
33  struct sockaddr_in ser_addr; /* server地址信息 */
34  struct sockaddr_in cli_addr; /* 客户端地址信息 */
35  struct sockaddr_in sen_addr; /* 客户端地址信息 */
36
37  char msg[MAX_MSG_SIZE]; /* 缓冲区,存放从客户端来的请求 */
38  memset(msg, 0, MAX_MSG_SIZE); //清空
39
40  if (ser_sockfd < 0)
41  { /*创建失败 */
42      fprintf(stderr, "socket Error:%s\n", strerror(errno));
43      getchar(); getchar();
44      exit(1);
45  }
46  /* 初始化服务器地址 */
```

绑定端口

```
56     ser_addr.sin_port = htons(SERVER_PORT); // 为这个server进程指定端口号
57     if (bind(ser_sockfd, (struct sockaddr*)&ser_addr, sizeof(struct sockaddr_in))
58         < 0)
59     { /* 绑定失败 */
60         fprintf(stderr, "Bind Error:%s\n", strerror(errno));
61         exit(1);
62     }
63     printf("服务端打开成功，等待连接客户端...\n");
64     /* int cli_sockfd = accept(ser_sockfd, (struct sockaddr*)&cli_addr, &addrlen); */
```

“死循环”等待客户端建立连接，直到传输完毕或超时

```
66     while (1)
67     { /* 等待接收客户端连接请求 */
68
69         memset(msg, 0, MAX_MSG_SIZE);
70
71         /* memset(msg, 0, MAX_MSG_SIZE); */
72         int len = recvfrom(ser_sockfd, msg, MAX_MSG_SIZE, 0, (struct sockaddr*)&cli_addr, &cli_a
73         if (len <= 0) {
74
75             num++;
76             if (num > 100) {
77                 cerr << "客户端 IP: 127.0.0.1 过程中失去连接。" << endl;
78                 remove("data.txt");
79                 getchar(); getchar();
80                 exit(-1);
81             }
82             else {
83                 continue;
84             }
85         }
86         num = 0;
87         printf("接收到客户端连接请求 from %s\n", inet_ntoa(cli_addr.sin_addr));
88         printf("收到客户端发送的数据: \n");
89         printf("%s\n", msg); /* 在屏幕上打印出来 */
90
91         FILE* fp;
92         fp = fopen("data.txt", "a+");
93         if (fp == NULL)
94         {
95             printf("File Can Not Open To Write!\n");
96             _exit(-1);
97         }
98         printf("将接收到的数据写入文件中...\n");
99         // 调用fwrite函数将recv_buf缓存中的数据写入文件中
100        int write_length = fwrite(msg, sizeof(char), len, fp);
101        if (write_length < len)
102        {
103            printf("文件写入失败!\n");
104            break;
105        }
106        printf("接收客户端传输的数据流成功!\n");
107        fclose(fp);
108    }
```

3.3 思考题

1、 一个应用可以对应几个 IP? 几个端口?

一个应用可以对应多个 IP，多个端口，最简单的例子就是 UDP 协议的广播功能，可以向多个 ip 同时发送数据。

PS：一个 IP 只能绑定一个端口，否则会造成 ip 冲突

一个端口可以绑定多个 ip，在一个逻辑端口上配置第二个 ip 时要后缀 sub

2、 一个端口可以给几个应用使用?

一个端口同一时间状态只能供一个应用使用，不过一个应用可以同时调用多个端口

3、 能不能随意使用知名端口? 为什么? 举例。

不能，大多数系统中，只有系统处理程序或特权用户运行进程（process）才可使用知名端口号。在 ICANN 之前，知名端口号一直是由国际因特网地址分配委员会（IANA）管理。如果我们的程序使用知名端口号，如 HTTP 协议的 80、HTTPS 的 443、FTP 的 20/21、SSH 的 22，可能会产生端口冲突，导致实验程序或者对应的服务无法正常工作。

4 实验总结

通过这次实验，我利用 Socket 实现了 TCP 和 UDP 通信，

在客户端，用户选择本地的 data 文件，并发送到服务器端。服务器端，接收客户端传输的数据流，并按保存在服务器端。另：由于条件所限，只有一台电脑，按 IP 作为文件名保存似乎不太有意义。同时实现了传输过程中服务器端、客户端发现断开及其处理。

应当注意到的是 TCP 和 UDP 协议的对比，UDP 协议面向报文和无连接，不能确保传输可靠性。在考量通过 UDP 通信时如何判断服务器端、客户端发现断开时，我采取的策略是从应用层出发（相当于应用程序之间自行约定）给 `recvfrom` 设定一个时延，同时当丢包数量达到一定值时，判断服务器和客户端的连接已经断开。已故应当留意在复现实验的时候，打开 `UDP-SERVER.EXE` 后应尽快（10s 内）打开 `UDP-CLIENT.EXE`，才能实现传输，否则判为连接已断开。

	UDP	TCP
是否连接	无连接	面向连接
是否可靠	不可靠传输，不使用流量控制和拥塞控制	可靠传输，使用流量控制和拥塞控制
连接对象个数	支持一对一，一对多，多对一和多对多交互通信	只能是一对一通信
传输方式	面向报文	面向字节流
首部开销	首部开销小，仅8字节	首部最小20字节，最大60字节
适用场景	适用于实时应用（IP电话、视频会议、直播等）	适用于要求可靠传输的应用，例如文件传输

提交的核心文件如下图所示，每个文件夹中包含源码/测试文件和生成的 release 版可执行文件。

