《计算机网络协议开发》实验报告

第 16 次实验
综合实验的实现

姓名：元玉慧
学号：101220151
10 级 计算机 系 4 班
邮箱：njucsyyh@gmail.com
时间：2013/06/22

一、 实验目的

通过本实验实现一个能够在自己搭建的 STCP 协议栈上能够正常运行的双工网络环境，并可以运行 IM，实现单聊/群聊/登录退出等功能。

二、实验设计

我们已经是实现了 STCP 协议栈，需要在原来已经建立好的 SIP 实验中的 stcp_server、stcp_client 中实现双工通信。然后实现客户端和服务器端的双向通信即可。

双工实现的核心代码部分：

Stcp_client.c 部分：

-1- 添加了 stcp_client_recv()函数

```
int stcp_client_recv(int sockfd, void* buf, unsigned int length){
  while(1){
    pthread_mutex_lock(tcblist[sockfd].tcb->bufMutex);
    unsigned int recv_count = tcblist[sockfd].tcb->usedBufLen;
        if(recv_count >= length){
       printf("Bingo ---  at sockfd %d --- recv data  length is %d...\n", sockfd
      memcpy((char*)buf, tcblist[sockfd].tcb->recvBuf, length);
      if(recv_count > length){
        char temp[recv_count-length];
        memcpy(temp, tcblist[sockfd].tcb->recvBuf+length, recv_count-length);
        memset(tcblist[sockfd].tcb->recvBuf, 0, RECEIVE_BUF_SIZE);
        memcpy(tcblist[sockfd].tcb->recvBuf, temp, recv_count-length);
      }
      else
        memset(tcblist[sockfd].tcb->recvBuf, 0, RECEIVE_BUF_SIZE);
      tcblist[sockfd].tcb->usedBufLen -= length;
      pthread_mutex_unlock(tcblist[sockfd].tcb->bufMutex);
      break;
    }
    pthread_mutex_unlock(tcblist[sockfd].tcb->bufMutex);
    sleep(RECVBUF_POLLING_INTERVAL);
        printf("sleep 1 sec...\n");
  }
    return 1;
}
```

-2- 在 seg_handler 中添加了处理接收到的报文类型是 DATA 的情况

```
case CONNECTED:{
  if(server_type==DATA){
    /*
     * debug ...
     */
    printf(" \n\n\n*******   client recv data seq is %d-----expectnum is %d **********\n", seq, tcblist[index].tcb->expect_seqNum);
    char c;
    fgets(&c, 1, stdin);

    pthread_mutex_lock(tcblist[index].tcb->bufMutex);
    if(tcblist[index].tcb->expect_seqNum == seq){
      if(tcblist[index].tcb->usedBufLen+data_len <= RECEIVE_BUF_SIZE){
        printf("Server: receive the data ok ...\n");
        memcpy(tcblist[index].tcb->recvBuf+tcblist[index].tcb->usedBufLen, ((seg_t*)buffer)->data, data_len);
        tcblist[index].tcb->usedBufLen += data_len;
        tcblist[index].tcb->expect_seqNum += data_len;
            seg_t* temp = (seg_t *)malloc(sizeof(seg_t));
            temp->header.type = DATAACK;
            temp->header.ack_num = tcblist[index].tcb->expect_seqNum;
            temp->header.src_port = client_port;
            temp->header.dest_port = server_port;
            temp->header.length = 0;
            sip_sendseg(sip_conn, tcblist[index].tcb->server_nodeID, temp);

            printf("\n**********************\n");
            printf("type:  DATAACK\n");
            printf("The expect seq is equal to the recv seq ...\n");
            printf("srcport: %d  \n", server_port);
            printf("destport: %d \n", client_port);
            printf("receive the data seq: %d \n", seq);
            printf("data seq length : %d \n", data_len);
            printf("expect the data seq:  %d\n", tcblist[index].tcb->expect_seqNum);
            printf("**********************\n");
      }
    }
```

```
        else{
          seg_t* temp = (seg_t *)malloc(sizeof(seg_t));
            temp->header.type = DATAACK;
            temp->header.ack_num = tcblist[index].tcb->expect_seqNum;
            temp->header.src_port = client_port;
            temp->header.dest_port = server_port;
            temp->header.length = 0;

            sip_sendseg(sip_conn, tcblist[index].tcb->server_nodeID, temp);
            printf("\n**********************\n");
            printf("type:  DATAACK\n");
            printf("The expect seq No equal to the recv seq ...\n");
            printf("srcport: %d  \n", server_port);
              printf("destport: %d \n", client_port);
              printf("expect the data seq:  %d\n", tcblist[index].tcb->expect_seqNum);
            printf("**********************\n");
        }
        pthread_mutex_unlock(tcblist[index].tcb->bufMutex);
      }
```

Stcp_server.c 部分：

-1-　实现发送数据函数

```c
int stcp_server_send(int sockfd, void* data, unsigned int length){
  int nodeID = tcblist[sockfd].tcb->client_nodeID;
  int count = length/MAX_SEG_LEN;
  int remain = length%MAX_SEG_LEN;
  if(data==NULL){
      printf("Warning: send NULL data ...\n");
      return -1;
  }
  char *char_data = (char*)data;
  pthread_mutex_lock(tcblist[sockfd].tcb->bufMutex);
  if(tcblist[sockfd].tcb->sendBufHead == NULL){
       pthread_t tid;
       pthread_create(&tid, NULL, sendBuf_timer, (void *)tcblist[sockfd].tcb);
  }
  //store the data to the seg_buf
  while(count > 0){
    if(tcblist[sockfd].tcb->sendBufHead == NULL){
    }
    else{
    }
    count--;
  }

  if(remain > 0){
  }

  if(tcblist[sockfd].tcb->unAck_segNum < GBN_WINDOW){
  }
  pthread_mutex_unlock(tcblist[sockfd].tcb->bufMutex);
  return 1;
}
```

-2- 实现 seghandler 处理 DATAACK 报文

```c
    if(client_type==DATAACK){
        pthread_mutex_lock(tcblist[index].tcb->bufMutex);
        printf("\n**********************\n");
        printf("srcport: %d  \n", server_port);
        printf("destport: %d \n", client_port);
        printf("type: DATAACK\n");
        printf("ack_num: %d \n", ack);
        printf("**********************\n");

        segBuf_t *temp = tcblist[index].tcb->sendBufHead;

        if(temp==NULL) printf("the head is null\n");
        while(temp != NULL){
          printf(" *******   the Head seq num is %d\n", temp->seg.header.seq_num);
          if(temp->seg.header.seq_num < ack){
            printf("free the acked data --- the seq is %d...\n", temp->seg.header.seq_num);
            tcblist[index].tcb->sendBufHead = temp->next;
            tcblist[index].tcb->unAck_segNum--;
            free(temp);
            temp = tcblist[index].tcb->sendBufHead;
          }
          else{
            break;
          }
        }

        // all is acked
        if(tcblist[index].tcb->sendBufHead == NULL){
            tcblist[index].tcb->sendBufTail = NULL;
            tcblist[index].tcb->sendBufunSent = NULL;
            tcblist[index].tcb->unAck_segNum = 0;
        }
```

```
437
438             //< GBN so we can send the new seg...
439             if(tcblist[index].tcb->sendBufunSent != NULL){
440               segBuf_t *tt = tcblist[index].tcb->sendBufunSent;
441               int nodeID = tcblist[index].tcb->client_nodeID;
442               while(tcblist[index].tcb->unAck_segNum < GBN_WINDOW && tt != NULL){
443                         gettimeofday(&(tt->sentTime), NULL);
444                         sip_sendseg(sip_conn, nodeID, &tt->seg);
445                         tcblist[index].tcb->unAck_segNum++;
446                         tt = tt->next;
447               }
448               tcblist[index].tcb->sendBufunSent = tt;
449             }
450             pthread_mutex_unlock(tcblist[index].tcb->bufMutex);
451           }
452
453           break;
454         }
```

2 边全双工通信实现的统一的结构体

```
20  typedef struct tcb {
21    unsigned int server_nodeID;
22    unsigned int server_portNum;
23    unsigned int client_nodeID;
24    unsigned int client_portNum;
25    unsigned int state;
26    //send index
27    unsigned int next_seqNum;
28      //send buffer manage ...
29    segBuf_t* sendBufHead;
30    segBuf_t* sendBufunSent;
31    segBuf_t* sendBufTail;
32    unsigned int unAck_segNum;
33    //recv index
34    unsigned int expect_seqNum;
35    //recv buffer manage
36    char* recvBuf;
37    unsigned int  usedBufLen;
38    //mutex to protect the buffer ...
39    pthread_mutex_t* bufMutex;
40  } tcb_t;
41
```

在双工测试 OK 的基础上搭建 IM 测试平台：

im.server.c

具体设计实现原理同前面的 IM 即时通讯实验设计的实现

采用的数据结构以及操作如下：

```
typedef struct Online_list{
    int port_id;
    char username[10];
    char passwd[10];
    bool tag;
    bool logon;
}online_list;
```

```
28  pthread_mutex_t  mutexsum;
29  static int Online_num;
30  static online_list List[100];
31  //im level function ...
32  void init_list();
33  void add_list(int id);
34  void del_list(int id);
35  void logon(int id, char *name, char *wd);
```

服务器的对于客户端的配置

```
#define WAITTIME 15
#define SERVERPORT1 101
#define SERVERPORT2 102
#define SERVERPORT3 103

//serv_type
#define LOGON 0x01
#define LOGOFF 0x02
#define TO 0x03
#define ALL 0x04
#define SHOW 0x05

#define MAXLINE 200 //==max_seg_t data len
#define LISTENQ 20
#define MAXONLINE 100
```

具体功能调用函数略去，main 函数具体实现设计如下：

```
int main() {
  printf("yy 0...\n");
  srand(time(NULL));
  pthread_t tid;
  printf("yy 1...\n");
  int sip_conn = connectToSIP();
  if(sip_conn<0) {
    printf("can not connect to the local SIP process\n");
  }
  stcp_server_init(sip_conn);

  printf("yy 2...\n");
  init_list();
  int i;
  for(i=1; i<4; i++){
    int sockfd;
    if(i==1)
      sockfd = stcp_server_sock(SERVERPORT1);
    if(i==2)
      sockfd=stcp_server_sock(SERVERPORT2);
    if(i==3)
      sockfd=stcp_server_sock(SERVERPORT3);

    if(sockfd < 0) {
       printf("can't create stcp server\n");
       exit(1);
    }
    stcp_server_accept(sockfd);
    printf("New client join in ...\n");
    add_list(sockfd);
    pthread_create(&tid, NULL, &doit, (void *)sockfd);
    printf("sockfd is %d ......\n", sockfd);
  }
  while(1){
    sleep(20);
```

客户端 im_client.c 关键代码

```
/*
   * fix a bug...
 */
signal(SIGPIPE, SIG_IGN);

if(stcp_client_connect(sockfd, server_nodeID, SERVERPORT)<0) {
  printf("fail to connect to stcp server\n");
  exit(1);
}
printf("client connected to server, client port:%d, server port %d\n",CLIENTPORT,SERVERPORT);

system("clear");
printf("\nWelcome to the YYHCHATING HOME!\n");
printf("***********************************************\n");
printf("*               INSTRUCTOR INFO               *\n");
printf("***********************************************\n");
printf("LOGON:username:passwd\n");
printf("LOGOFF\n");
printf("SHOW\n");
printf("ALL:******\n");
printf("TO:username:*******\n");
printf("***********************************************\n");
printf("***********************************************\n");

pthread_create(&tid, NULL, Recv, (void *)sockfd);
while(fgets(sendline, MAXLINE, stdin) != NULL){
  stcp_client_send(sockfd, sendline, MAXLINE);
  memset(sendline, 0, MAXLINE);
}
```

实验测试设计：

由于时间原因，我设计的代码中是每次 3 个客户端想要正确连接服务器的话，需要设置 3 个服务器的端口号，就是手动设置传给 3 个服务器的端口号，感觉很坑爹，对于助教测试来说。。。然后我调试程序的时候也是非常的坑爹！

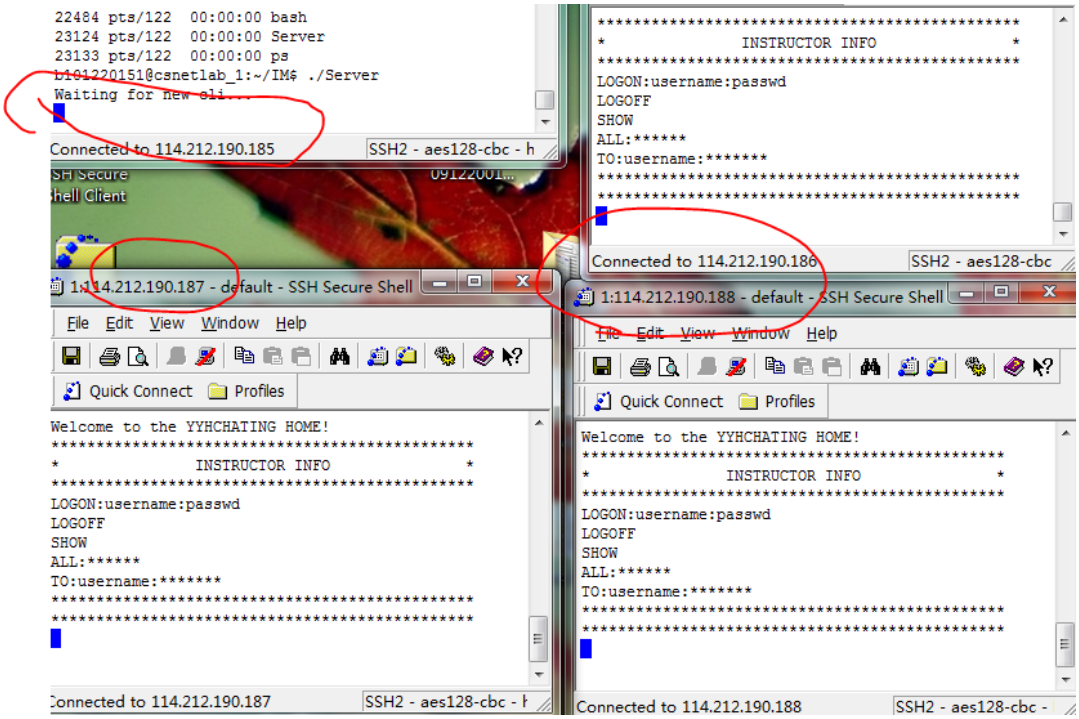
三、实验结果

基于 sip，son, stcp 协议栈我们实现了在自己的协议栈上收发文件，如下分别对 simple 和 stress 的情况进行了测试：

实验测试结果说明：

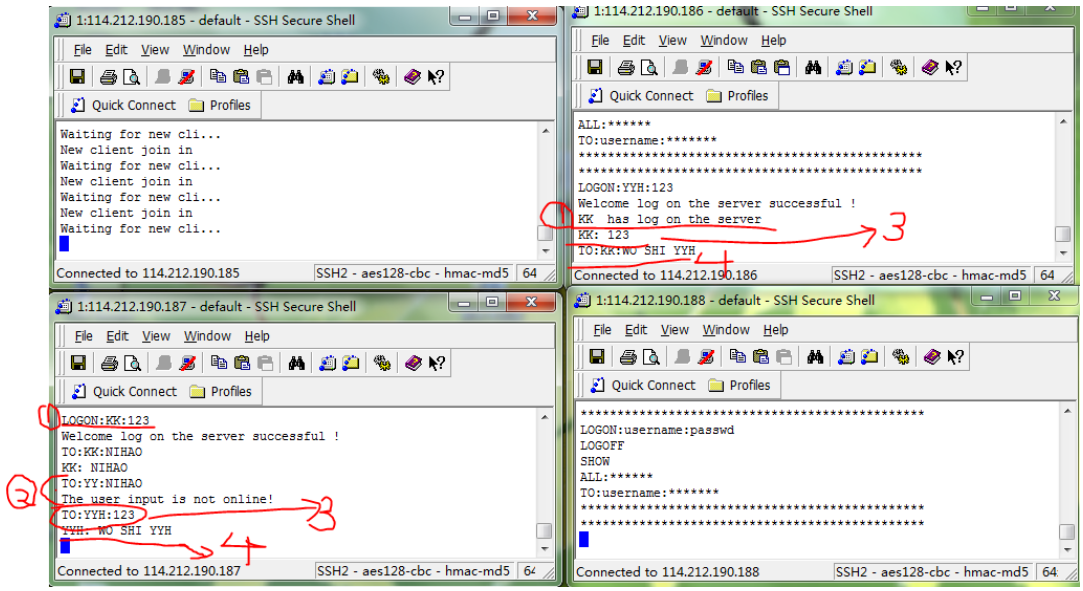
由于本人设计的实验的测试方法比较复杂，然后测试调试都十分困难，然后调好了程序，进行了简单的双工测试和 IM 即时通讯测试：

所以测试的时候程序中可能仍然存在 BUG,不过，在验收的时候，我调试了一会，可以测试通过，但是不保证助教在测试的时候会通过，由于本人设计的测试操作比较繁琐。



后来又修正重新设计了测试的通信界面，如下是部分测试的截图：

正常的端到端的收发消息截图如下：

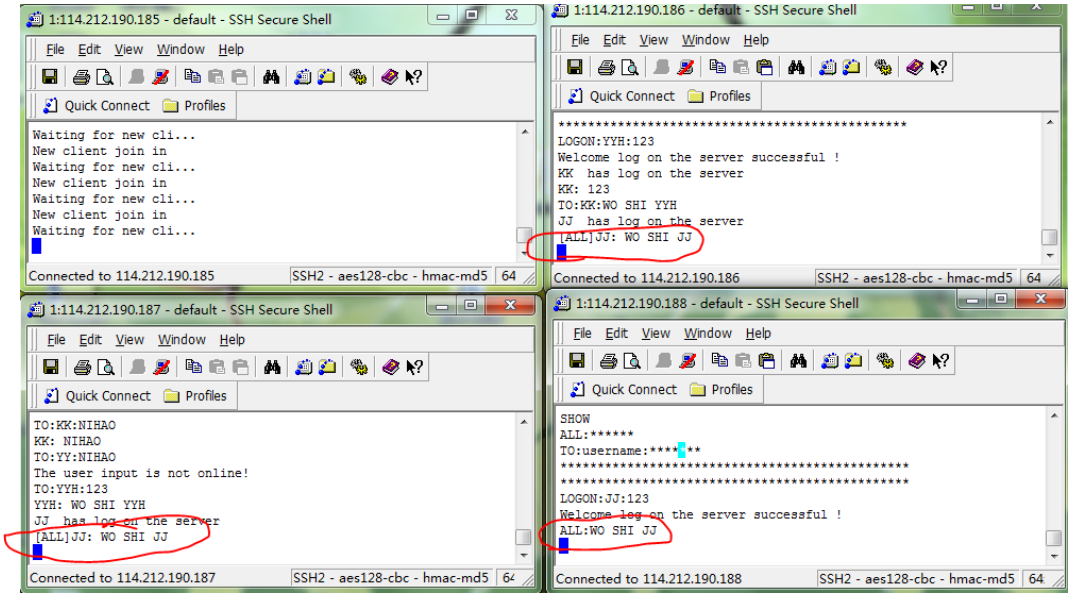


-1-是新的用户登录的时候，其他客户会收到好友登录通知

-2-是用户发送消息时输入的用户名不在线，提示用户

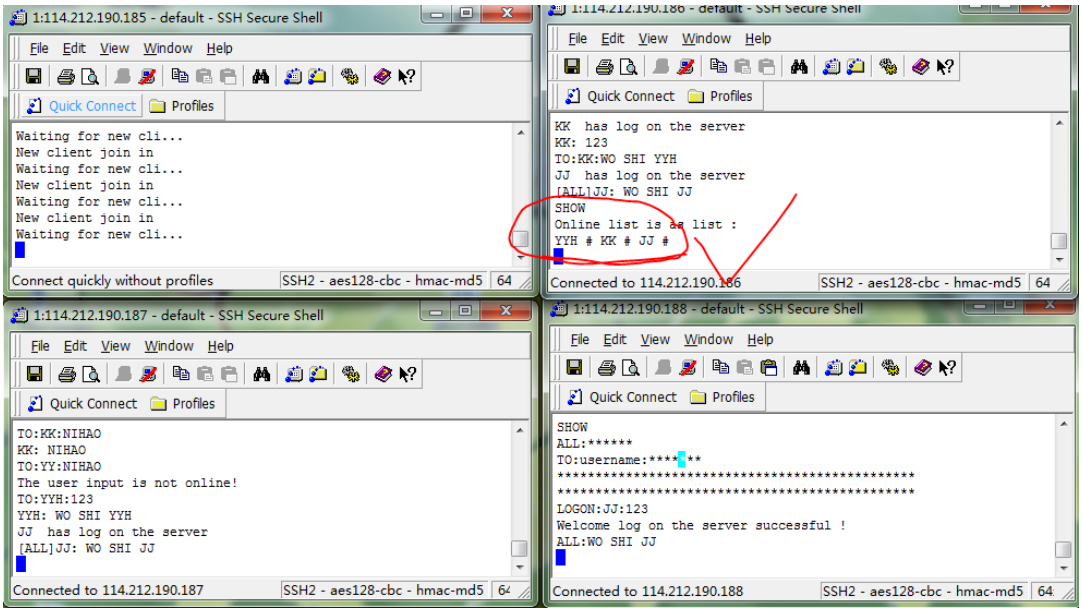
-3，4- 是用户之间互发消息正常

群发消息：



获取当前在线好友列表的命令：

SHOW 命令可以获取当前在线的好友列表：

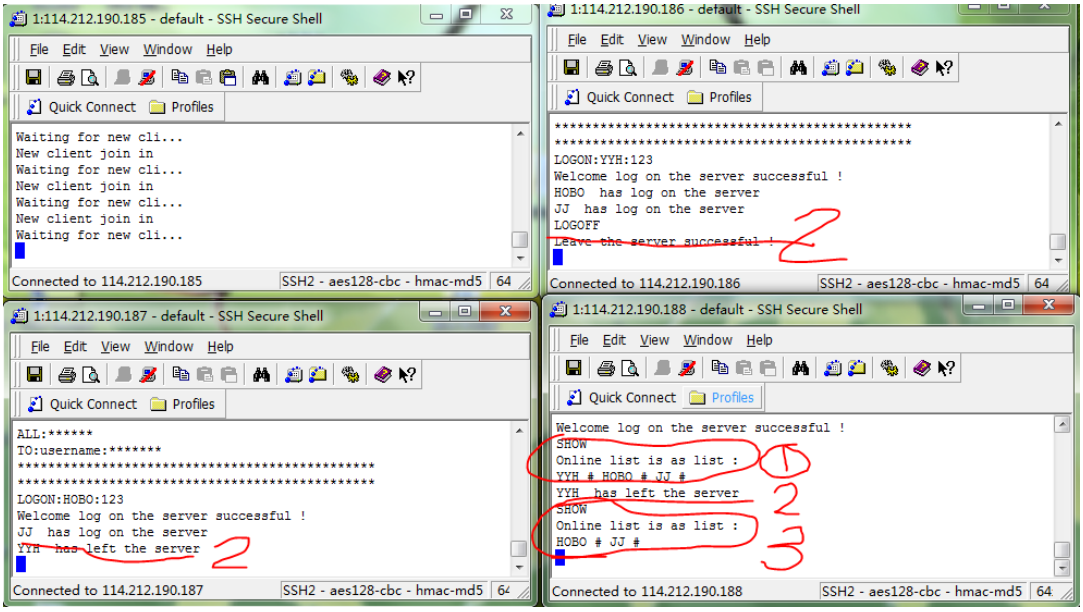


请求离线的命令：

-1- 用户都在线的时候查看在线用户有 3 个

-2-YYH 离线 其他客户端收到 YYH 离线的通知

-3-在其他客户端查询当前在线的客户列表，YYH 不在列表中，在线好友列表正确



五、 实验中遇到的问题

实验中遇到的问题：

(1) **在实现双工的时候遇到了一些问题：**

此次试验的第一步就是在 SIP 的基础上实现双工通信，然后我的思路还是比较清晰的，就是同一两边的收发的 TCB 控制块中的内容，然后就是同一两边的收发函数，还有就是修改 seghandler 线程函数内部处理 DATA 和 DATAACK 的情况，但是由于疏忽，在调试双工通信的时候还是调试了很久，主要是由于自己在实现的时候一些赋值操作出错，然后就是那个端口之间的问题，以及同步通信的时候序列号的问题，都是一些很细节的东西，还是调试了很久，才搞定了双工通信。

(2) **在实现 IM 即时通讯中遇到的问题，及解决情况：**

在实现即时通讯的时候，我是基本上是在以前实现的 IM 上改的，实现的比较简单，就是把所有的 send/recv 系统调用函数全部替换为 stcp_send 和 stcp_recv 但是，由于其中涉及到端口，就是每个客户端对应于服务器都需要分配一个独特的端口，然后我设计的方法是默认其中一个是服务器，然后客户端需要按照 185---→ 186 ---→187 的顺序登录，然后就是 188 必须是服务器，分配给客户端的 3 个端口分别是 101，102，103，然后就是最后都连接上之后，然后客户端登陆，发送 SYN 序列，然后服务器会发送 SYNACK 确认连接 OK,然后就是收发数据，调试了很久，收发数据部分是有部分 BUG，没有调好，就是有时候，客户端之间发送消息发送不出去，服务器可以收到，应该是转发处理的时候存在问题，然后验收的时候调试的差不多。

(3) **在完善 IM 的时候悲剧了。。。**

在验收试验后，我开始修正最后的 IM 实现，然后没有备份，就开始修改代码了，改了代码的实现风格，可是悲剧的事情然后就发生了，由于手贱。。。不知道修改了哪里，客户端之间发送消息的时候经常出问题。。。由于马上期末考试了，最近在期末复习，然后就没有来得及调试好这个手贱的 BUG。。。希望老师手下留情啊~~~~

六、 实验的启示/意见和建议

实验已经做到了最后一个，感受最大就是，写代码其实最重要的不是实现代码的结果，而是在实现的过程中能够学会提高自己的编程素质，但是很可惜，这学期虽然写了很多代码，还是感觉自己的代码素质还是不够高，主要对于自己的期望也比较高，希望以后写代码的时候是带着智商写代码，写出有智商的代码，而不是一味的去做可以直接依赖于以往经验的工作，其实编码中也发现自己有时候过于依赖以往所拥有的经验，
然后通过所有的 STCP 实验，收获很大，发现自己对于很多知识的理解要比动手做实验之前的理解更加深刻。
一学期过来，是磨砺更是成长，是选择更是放弃，由于在网络实验上花的时间比较多，编译原理实验做的不是很好，也是很遗憾的。。。编译原理有 4 个学分！！！ 感觉有点亏啊~~~不过，过去的事情无法改变，回首这学期，真心很充实，很累，很多时候都想放弃，很多时候都很累，很多时候都要熬夜调试程序，多了一个个奋斗的日夜，当然更多的也是成长，相信在以后的路上会更加乐观，无论面对如何的苦难和挫折。都要坚持不懈的奋斗下去，为了 HKUST，继续加油，坚持最初的决定！！！
I am the best !!!