Client:

```
yuxunnn@LAPTOP-YUXUN:lab2$ ./client
give me an IP to send: 127.0.0.1
server's's port? 9999
Waiting for a commands...
download video.mp4
client: sent 1036 bytes to 127.0.0.1
client: receive 1036 bytes from 127.0.0.1
FILE_EXISTS
Receiving...
        Receive a packet seq_num = 0
        Oops! Packet loss!
        Receive a packet seq_num = 1
        Receive a packet seq_num = 2
        Receive a packet seq_num = 3
        Oops! Packet loss!
        Oops! Packet loss!
        Receive a packet seq_num = 4
```

```
        Oops! Packet loss!
        Receive a packet seq_num = 118
        Oops! Packet loss!
        Oops! Packet loss!
        Oops! Packet loss!
        Oops! Packet loss!
        Receive a packet seq_num = 119
        Oops! Packet loss!
        Receive a packet seq_num = 120
Total cost 11 secs
Waiting for a commands...
```

1. 用 while 迴圈來接收封包，如果遺失了就直接 continue 等待 server 再傳封包過來，沒有遺失的話檢查封包的 seq_num 是不是我們需要的，是的話就把封包裡的資料寫進 buffer 裡，注意要使用 memcpy 而不是 strcat，因為封包有些資料是-1，會影響到 strcat 的功能。

```c
while(1) {
    if (recvfrom(sockfd, &rcv_pkt, sizeof(rcv_pkt), 0, (struct sockaddr *)&info, (socklen_t *)&len) != -1){
        //=======================
        // Simulation packet loss
        //=======================
        if(isLoss(0.5)){
            printf("\tOops! Packet loss!\n");
            continue;
        }
        //============================================
        // Actually receive packet and write into buffer
        //============================================
        printf("\tReceive a packet seq_num = %d\n", rcv_pkt.header.seq_num);
        if (rcv_pkt.header.seq_num == receive_packet){
            snd_pkt.header.ack_num = receive_packet;
            snd_pkt.header.is_last = rcv_pkt.header.is_last;
            receive_packet ++;
            int recvSize = (123431-index < 1024) ? 123431 - index : 1024;
            memcpy(&buffer[index], rcv_pkt.data, recvSize);
            index += recvSize;
            // fwrite(rcv_pkt.data, 1, 1024, fd);
        }
    }
```

2. 如果我們收到的封包是最後一個封包了，就將 buffer 的所有資料寫入 download_video.mp4 這個檔案裡。

```c
//============================================
// Write buffer into file if is_last flag is set
//============================================

if (rcv_pkt.header.is_last){
    fwrite(buffer, 1, sizeof(buffer), fd);
}
```

3. 回傳對應的 ACK 封包給 server，如果收到最後一個封包了就跳出迴圈，完成檔案的傳輸。

```
//====================
// Reply ack to server
//====================
sendto(sockfd, &snd_pkt, sizeof(snd_pkt), 0,(struct sockaddr *)&info, len);
if (rcv_pkt.header.is_last){
    break;
}
```

Server:

```
yuxunnn@LAPTOP-YUXUN:lab2$ ./server 9999
====Parameter====
Server's IP is 127.0.0.1
Server is listening on port 9999
==============
server waiting....
process command....
filename is video.mp4
FILE_EXISTS
server: sent 1036 bytes to 127.0.0.1
trasmitting...
Send a pack seq_num = 0
Receive a packet ack_num = 0
Send a pack seq_num = 1
Timout! Resend packet!
Send a pack seq_num = 1
```

```
Send a pack seq_num = 119
Timout! Resend packet!
Send a pack seq_num = 119
Receive a packet ack_num = 119
Send a pack seq_num = 120
Timout! Resend packet!
Send a pack seq_num = 120
Receive a packet ack_num = 120
send file successfully
server waiting....
```

1. 把 FILE*移動到檔案開頭

```
int seq_number = 0;
fseek(fd, 0, SEEK_SET);
snd_pkt.header.isLast = 0;
```

2. 一開始先建立 receive_thread，之後用 while 迴圈，一次讀取 1024 個 bytes 放入封包的 data，然後傳送封包給 client，同時記錄現在送出的時間，在 receive_thread，我們會不斷接收來自 client 的封包。

```
while(1){
    // At the first time, we need to create thread.
    if(!first_time_create_thread){
        first_time_create_thread = 1;
        pthread_create(&th1, NULL, receive_thread, NULL);
    }
    //==========================
    // Write data into send packet
    //==========================

    fread(snd_pkt.data, 1, 1024, fd);

    //==========================
    // Send video data to client
    //==========================

    sendto(sockfd, &snd_pkt, sizeof(snd_pkt), 0,(struct sockaddr *)&client_info, len);
    printf("Send a pack seq_num = %d\n", snd_pkt.header.seq_num);
    sentTime = (clock()*1000)/CLOCKS_PER_SEC;
```

```
while (recvfrom(sockfd, &rcv_pkt, sizeof(rcv_pkt), 0, (struct sockaddr *)&info, (socklen_t *)&len) != -1){
    pthread_mutex_lock(&mutex);
    printf("Receive a packet ack_num = %d\n", rcv_pkt.header.ack_num);
    pthread_mutex_unlock(&mutex);
}
```

3. 建立 timout_thread，讓 timeout_thread 和 receive_thread 同時進行，並且等到 timeout_thread 執行完畢，在 timeout_thread 裡我們會不斷的檢查是否 TIMEOUT，超過時間就重新傳送封包，並且更新記錄送出的時間，直到 server 收到了正確的 ACK 封包為止，在使用 pthread 的時候，遇到存取同一份資料時，要記得利用 pthread_mutex 上鎖和解鎖，以免發生錯誤。

```
//====================================
// Checking timeout & Receive client ack
//====================================
pthread_create(&th2, NULL, timeout_thread, NULL);
pthread_join(th2, NULL);
```

```
while(1){
    pthread_mutex_lock(&mutex);
    if (rcv_pkt.header.ack_num == snd_pkt.header.seq_num){
        pthread_mutex_unlock(&mutex);
        pthread_exit(NULL);
    }
    pthread_mutex_unlock(&mutex);
    if ((clock()*1000)/CLOCKS_PER_SEC - sentTime >= TIMEOUT){
        sendto(sockfd, &snd_pkt, sizeof(snd_pkt), 0,(struct sockaddr *)&client_info, len);
        printf("Timout! Resend packet!\n");
        printf("Send a pack seq_num = %d\n", snd_pkt.header.seq_num);
        sentTime = (clock()*1000)/CLOCKS_PER_SEC;
    }
}
```

4. 如果收到的是最後一個 ACK 封包，就可以跳出迴圈了，還有剩餘的資料的話就更新 seq_num 跟剩下的 filesize。

```
//====================================
// Checking Receive the last ack
//====================================

if (rcv_pkt.header.isLast) break;

//============================================
// Set is_last flag for the last part of packet
//============================================

snd_pkt.header.seq_num = seq_number = seq_number + 1;
filesize -= 1024;
if (filesize <= 1024){
    snd_pkt.header.isLast = 1;
}
```

5. 結束檔案的傳輸之後記得關閉 receive_thread。

```
first_time_create_thread = 0;
pthread_cancel(th1);
```