Introduction to Computer Networks - Lab 2

Name: Pei Chi, Huang

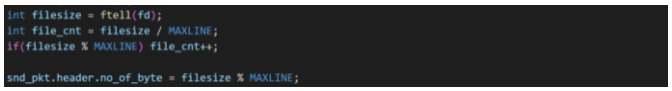
Student ID: 108020017

Department: Interdisciplinary Program of Science

I. Basic Implementation

■ Server side

The server side connects to an UDP socket of port 9999 and wait for a request from a client. It will respond to the client when requested “download fileName”. If the file exists, the server sends packets to the client using stop-and-wait mechanism, and receives ACK from the client. The sender will also handle timeout events by resending the packets if not receives the ACK in 100 milliseconds.

sendFile()	
	Calculate the total number of packets needed, and set the remaining number of bytes at the last packet.
	Read 1024 bytes from the given file name and send to the socket. The sequence number is updated if a new packet is created. Also, set the flags of the last packet to be 1.
	Using a nonblocking recv() with timeout interval. If there is ACK received within the time interval, the packet is successfully transmitted and the server keep sending the next packet. If not, resend the packet.
	If all the packets are sent, close the file and return. Otherwise, repeat the above steps.

* The recv() only returns after a large amount of time (longer than timeout) here.

■ Client Side

The client side connects to the UDP socket of the server and make commands “download fileName” to the server. If the file exists, the client starts to receive the packets and replies an ACK back to the server. During the process, the client may loss packets at random, and will ignore the packet received.

recvFile()	

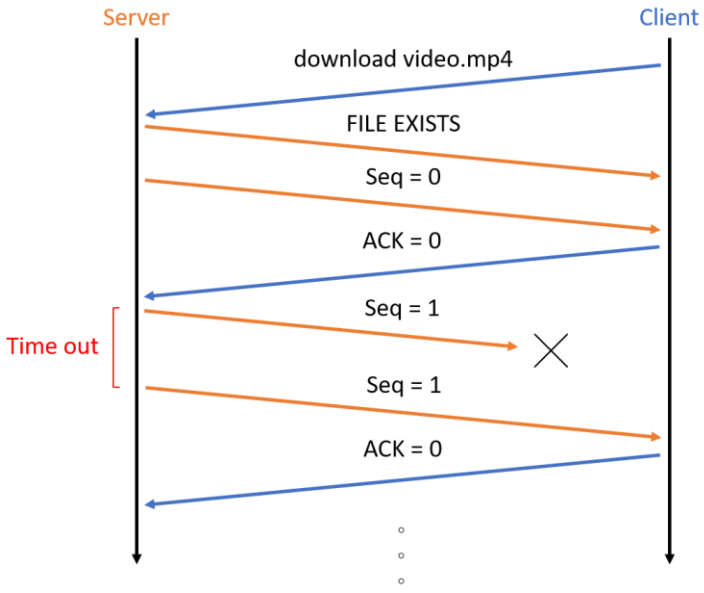
`sprintf(fileName, "download_");`
`strcat(fileName, str);`
 	
`// Open file`
`fd = fopen(fileName, "wb");`
	Open the file requested.

`recvfrom(sockfd, &rcv_pkt, sizeof(rcv_pkt), 0, (struct sockaddr *) &info, (socklen_t *) &len);`	Receiving the packet sent from the sender.
`if(isLoss(0.5)) {` `  printf("\tOops! Packet loss!\n");`	The packet losses with the possibility of 0.5. If a loss event occurs, ignore the packet.

`printf("\tReceive a packet seq_num = %d\n", rcv_pkt.header.seq_num);`
 	
`// ==================`
`//  Reply ack to server`
`// ==================`
`snd_pkt.header.ack_num = rcv_pkt.header.seq_num;`
`snd_pkt.header.seq_num = 0;`
`snd_pkt.header.is_last = 0;`
`sendto(sockfd, &snd_pkt, sizeof(snd_pkt), 0, (struct sockaddr *) &info, len);`
`printf("\tSend ACK of packet %d\n", rcv_pkt.header.seq_num);`
	If the packet is successfully transmitted, the client replies to the server with an ACK of the sequence number of the received packet.

`if(rcv_pkt.header.is_last == 1) {`
`  fwrite(rcv_pkt.data, 1, rcv_pkt.header.last_no_byte, fd);`
`  break;`
`} else {`
`  fwrite(rcv_pkt.data, 1, sizeof(rcv_pkt.data), fd);`
`}`
	Write the data to the download file. If the packet is denoted as the last one, the whole file is transmitted, then return.

■ Diagram



II. Result

Both server.c and client.c are compiled by gcc. The result was shown below:

Server side	Client side