Client-Server UDP Reliable Data Transfer Communication Protocol

Introduction:

In this program, Server is accepting packets from client through UDP connection. Client is sending data packets to server and server is accepting the data packet and uploading it on its side i.e. in Server directory. Client is sending the packets which can be 512 bytes long at the max. Here the we use UDP with reliable data transfer.

Instructions for Compilation:
1) Makefile has been created for the project so to compile it type command "make" when inside the project folder.
2) It will generate two executable named "urft-client" & "urft-server" corresponding to the Client and Server code respectively.

Instructions for Execution:
1) The server can be executed using following command ./urft-server 10000
    Where the 1st parameter represent the port number.
2) The client can be executed using following command ./urft-client 127.0.0.1 10000 filename
    Where the 1st parameter represents the IP address of server
    Where the 2nd parameter represents the port number of server
    Where the 3rd parameter represents the name of the file to be uploaded.

Protocol :
The header sent inside each packet is described below.

1] <u>Connection Establishment phase</u>
Here the client is first of all sending a packet to server for initiation of the connection, while server accepts the packet and establishes connection with the client.
Here, the Client sends a packet which can be represented as follows :

#	Filename	#	File Size	#	Number of parts	#
1b	variable	1b	30b	1b	10b	1b

here b = bytes

When server receives this packet it would retrieves all the parameters it needs for further reliable transportation of the packets and sends an ACK to the client as :

#	1	#
1b	1b	1b

here b = bytes

Now the server and client goes into file transfer phase

NOTE:
1) If the connection request send by client is lost server will not send any ACK in which case the client would time-out after 1 sec and retransmit the connection request.
2) If the ACK send by the server is lost the client would time-out after the 1sec and would resent

the connection request, this time the server will retransmit the ACK for establishing connection.

2] <u>Packet transport phase :</u>

Client Side :
Here the Client sends the packet by dividing the file into the packet of size 512 each in this 512 bytes the 12 bytes is the header which has the format as follows :

Sequence Number	#	Size	#	0	#	Data
5b	1b	3b	1b	1b	1b	500b

here b = bytes

Here, Sequence number holds the current packet sequence number. Size holds the size of data packet and in Data we have the actual payload. The 5th parameter in the above header is “0” for n-1 packets and is set to “1” for the final packet.
If a packets get lost or client doesn't receive any acknowledgment from the server so the client times out after 1 sec and would retransmit the previously send packet.

Server side :
Here the server is waiting for the packet to arrive. As soon as packet arrives it retrieves the head and checks the sequence number if its correct as per expectations and writes the data into the file and sends an acknowledgment as follows :

1	#	Sequence Number	#
1b	1b	5b	1b

here b = bytes

Here 1 represents the ACK and the middle field sends the sequence number of the packet that servers has just received. If the server receives an out of order packet it will sent an ACK response with the sequence number set to the last in order packet that was received.

3] <u>Finish Phase</u>

After completion of packet sending phase both server and client go into Finish Phase

Client Side :
After completion of sending all the packets and getting all acknowledgments for all the packets, Client sends FIN packet to the server. The format for finish packet is as below.

#	1	#
1b	1b	1b

here b = bytes

After sending a FIN packet client wait for its acknowledgment as another FIN packet from the server. If it doesn't receive a FIN packet from server it will wait for 1 sec and resent the FIN packet. It will do this 5 times and then would eventually terminate.

Server Side :
After receiving all the packets server would wait for a FIN packet from client. Client would send a FIN

packet to server and server would send back a FIN packet and terminate.

Conclusion :

Thus using this protocol the Client and Server can communicate reliably over UDP protocol.