Computer Networks, Fall 2020
Instructor: Jitendra Bhatia

Socket Programming

Task 3: Reliable File Transport over UDP

In this assignment, you will be familiar with building a reliable file transport protocol over UDP. As UDP offers no reliability, you will have to implement it within your application using the Go-BackN protocol. Unlike the previous assignment, your server need to handle multiple concurrent clients. You need to submit the server code, even though for your own testing, you will write a client as well.

The protocol is simple and has the following key steps:

1) Bootstrapping: Server will take the port on which it will listen as a command line argument (similar to previous assignments).

2) Request: Client sends a request packet (RRQ) to the server. The request will contain the

file name that it wants to fetch, and the window size (for the GoBackN protocol) that the server should use.

3) In the typical case, the server starts sending data packets. Each DATA packet will have

a sequence number (starting from zero) and will be fixed size (512 bytes). Only the last

packet will have a size less than 512 (it could be zero if the file size is a multiple of 512

bytes). This last packet will signal the end of file.

4) For each data packet, the client sends an ACK which carries the sequence number of

the corresponding data packet that was received (e.g., data packet 0 will generate ACK

0 and so on)

5) The server uses a fixed timeout value of 3 ms. If it doesn't receive an ACK, it retransmits

the entire window (remember this is GoBackN). The client will discard any outoforder

packet that it receives and will also not send an ack for it. So retransmissions from

server will only happen due to timeouts. After 5 consecutive timeouts (for the same

sequence number), the server should stop the communication.

6) In case the server cannot transfer the file (e.g., file doesn't exist), the server will send an

ERROR message (in response to the RRQ message from the client).

Using UDP Sockets

TCP and UDP have many similarities but some key differences too. Some of the things you need to keep in mind are:

● You will no longer use SOCK_STREAM socket type, use SOCK_DGRAM instead.

● You will use recvfrom() and sendto() instead of recv()/read() and send()/write().

● Unlike TCP which may break your message into smaller pieces (or combine smaller pieces into large ones), UDP will preserve message boundaries. So two writes (sendto()) calls on the sender will require two reads (recvfrom()) calls at the receiver. This will actually simplify your message processing (compared to TCP).

Types of Messages

1. Read request (RRQ): type; window size; file name
2. Data (DATA): type, sequence number, data
3. Acknowledgment (ACK): type; sequence number
4. Error (ERROR): type

Format

● Type: 1 byte. (RRQ = 1; DATA = 2; ACK = 3; ERROR = 4)

(e.g., char type = 1; //RRQ)

● Window Size: 1 byte (valid values are 19; don't need to worry about any other value)

(e.g., char win_size = 1)

[note: File transfer with window size = 1 (your protocol will behave similar to stopandwait)]

● Sequence number: 1 byte (valid values are 0 to 50, thus max file size will be limited to this.

e.g., char seq_no = 40;

● Filename: null terminated string (at most 20 bytes including null).

e.g., char fileName[20]

● Data: 0 to 512 bytes depending on the packet size.

e.g., char data[512]