Computer Networks Project 2

Reliable Data Transfer Protocol over UDP

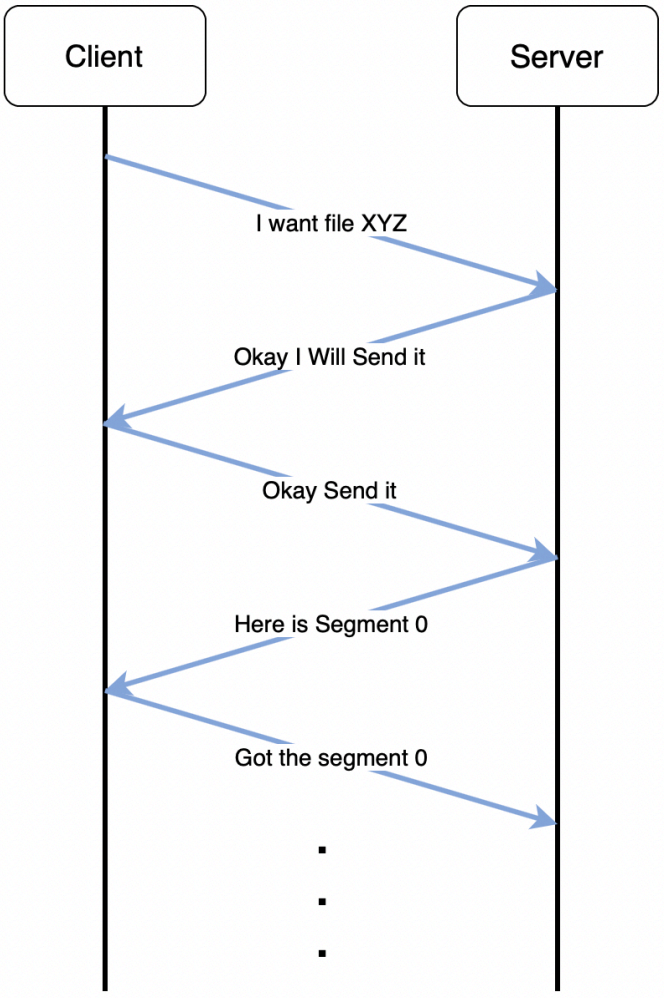
In this project, you are required to **implement a server and a client** that provides a reliable transport protocol (similar to TCP) **using UDP on top of an unreliable channel**. The server uses **Selective Repeat** to send packets to the client we provided. You will develop and run your **server and client codes locally**. You should implement both your client and server programs, considering the following tasks:

- Assume the channel between server and client is unreliable as stated above. Thus, all packets are prone to be lost during transmission. This case includes data packets as well as control packets such as Handshake packets, ACK packets. [1]
- Packet loss is controlled by changing parameters (**errRate**) of **unreliableSend()** function. Packet loss will take values: {1%, 5%, 10%, 20%}
- N is the window size of the selective repeat protocol and N takes values of {1, 10, 50, 100}. You are required to prepare a report and discuss the performance of you implementations for various **errRate**'s and **N.**
- The communication follows connection-oriented policy, so client and server establish a logical connection before sending any data packets.
- The server will detect timed-out packets as explained in the reliable data transfer protocol (rdt3) introduced in our course slides (chapter 3, pages 39, 48-49) .
- The client and the server will use the **Selective Repeat** scheme to pipeline multiple segments.
- You can not use TCP sockets and **you must use the unreliableSend()** function provided in the project files.

[1] During the transmission, you can assume that there will not be any bit errors / corruptions in delivered packets while packet losses are possible. If a packet is delivered, you may assume that it is without any bit errors.

Part 1 - Implementing Triple Handshake

In this part, the client establishes the end-to-end connection by sending its wanted file's name to the server. In this project, file is a simple text file having a word in each line which will be shared along with the client code. The name of the file is arbitrary and is known by the client from the start. When the server receives the client's initiation message, it checks whether the filename is correct. If it is wrong, then server timeouts. If the filename matches, the server sends an ACK packet with sequence number 0 to the client. Then, the client sends this ACK message to the server. After this step the triple handshake is completed.



Part 2 - Implementing RDT protocol

In this part, the server starts sending segments of the text file. The segments can get lost in the way. Your server should notice any lost segments when the ACK for that segment timeouts. The timeout value is 0.0001 seconds as in the sample client code. Thus the server needs to properly handle packet timeouts. Finally you must use the **Selective Repeat** method to pipeline your reliable transport protocol.

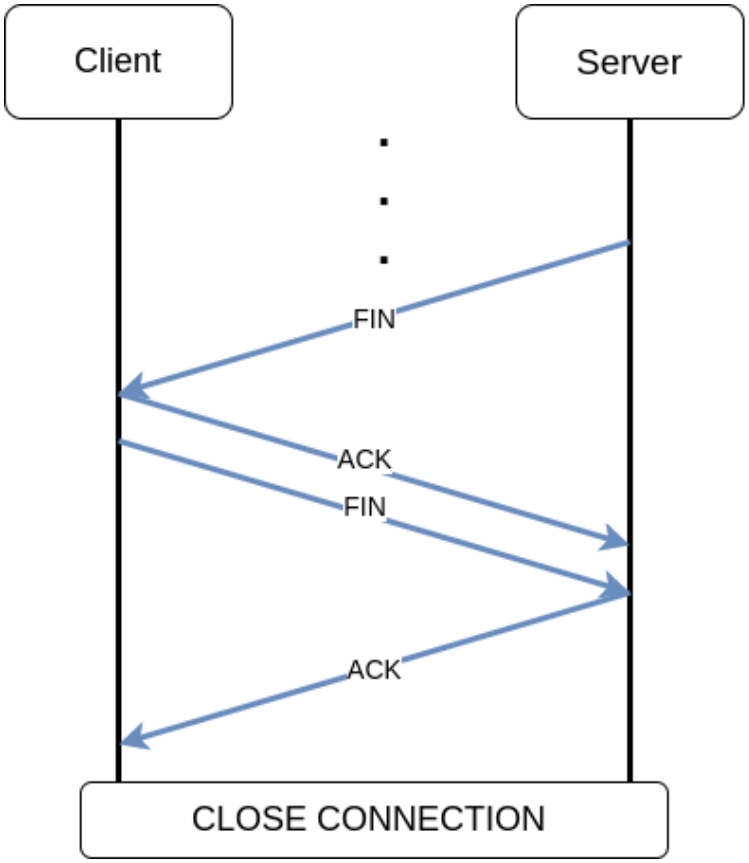


Figure 2: Ending Sequence

When the server receives the ACK for the last segment, it sends the FIN packet to the client. After the client receives the FIN packet it sends ACK(with sequence number = file's sequence number+1) back to the server. Following this the client increments the sequence number and sends its own FIN packet and partially gets closed(if server times out connection gets closed). After receiving the FIN packet, the server sends the ACK packet and closes the connection. If client's ACK and FIN packets get lost, both of the server and client close connection when they experience timeout.

Appendix A: Packet types

First byte of the packet determines the type of the packet. All of the packets are encoded using UTF-8. Payload lengths are given in terms of bytes and max payload length is 255 bytes. Similar to this, sequence numbers are counted with 1 byte.

Packet type	Meaning
0	Handshake
1	ACK
2	DATA
3	FIN

Handshake Packet

This packet is used by the client when requesting a file.

Bytes	Content
0	Packet type = 0
1	Payload Length (Filename)
2-...	Filename

ACK Packet

Bytes	Content
0	Packet type = 1
1	Sequence Number

Data Packet

Bytes	Content
0	Packet type = 2
1	Payload Length (Data)
2	Sequence Number
3-...	Data

FIN Packet

This packet is sent by the server to indicate data transfer is complete.

Bytes	Content
0	Packet type = 3
1	Sequence Number

Appendix B: unreliableSend()

This function takes the packet you created and sends it to the IP of the user(userIP) according to the random criteria. You must use this function for all interactions between the client and server.

```
def unreliableSend(packet, sock, userIP, errRate):
    if errRate < rd.randint(0,100):
        sock.sendto(packet, userIP)
```

Submission Guidelines

- You must upload your server and client code and report to Ninova.
- Your code must work on Python 3.7 or higher.