COMPUTER NETWORKS ASSIGNMENT 2 REPORT

(IMT2024082)

This is a report for the computer networks assignment 2 . We were asked to design programs for 4 problems which include making a simulation for RDT 2.2 , RDT 3.0 , TCP congestion control using Tahoe and Reno . I will explain my work on this and show the output accordingly .

1. Implement RDT 2.2 (NAK-Free Reliable Data Transfer) Protocol

To implement a simulation of RDT 2.2 I made a class called Packets that gives a packet basic values like sequence number , data and checksum and has some utility functions to check corruptions and calculate checksum . I made additional functions to add errors in in packets and ACKs . For implementation of the FSM I made functions for each state of FSM for both the sender and receiver side and the control flow is commanded the conditions of FSM.

The parameter choices I have given in this problem is the error percentage . The user can choose the chances of the packet getting error . I have given it a range between 0 to 10% to get only meaningful results. Though the input messages (packets) to be transmitted are hardcoded in the program

The output for this is program is :

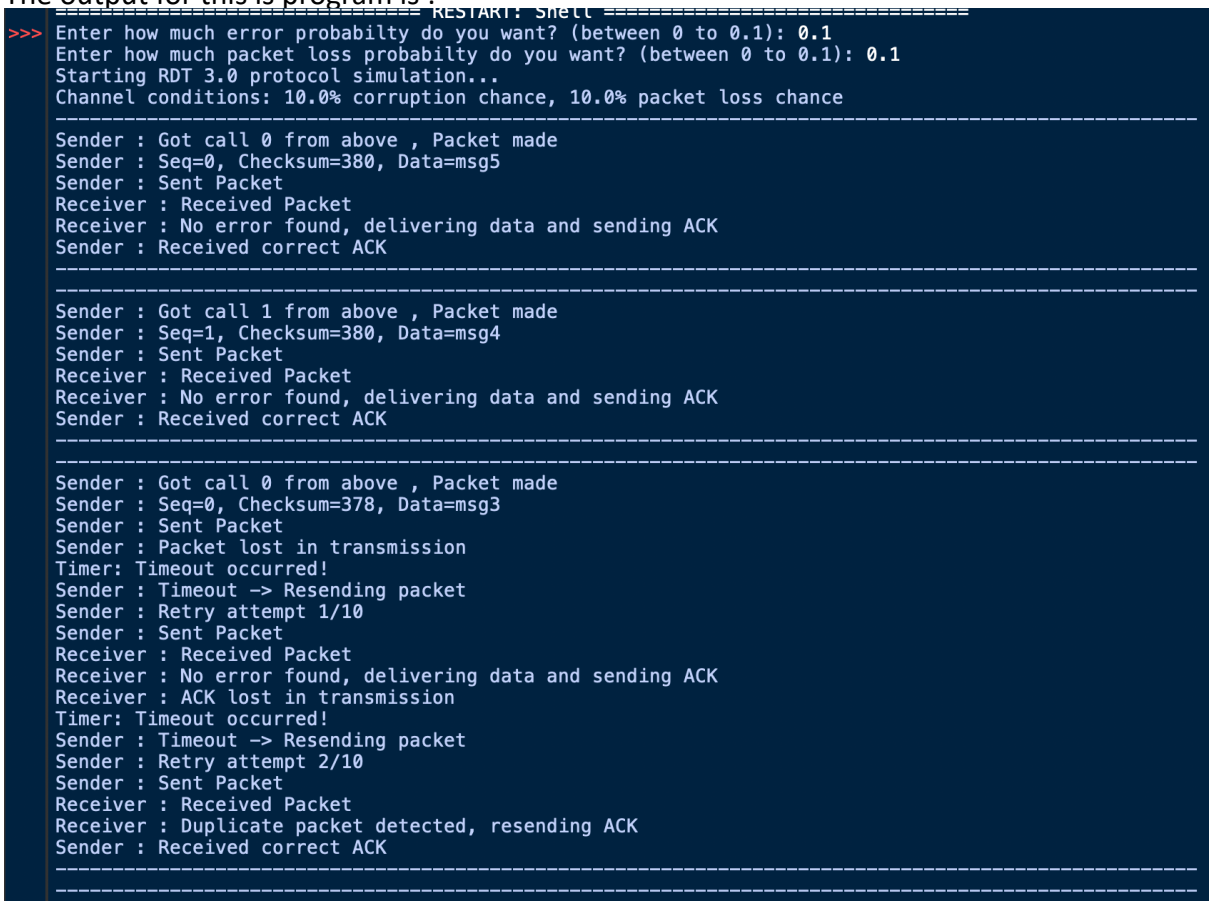
```
========== RESTART: /Users/ujjwalsingh/Desktop/college/cn ass/p1/p1.py =========
Enter how much error probabilty do you want? (between 0 to 0.1): 0.07
------------------------------------------------------------------------------------------------------------
Sender : Got call 0 from above , Packet made
Sender : Seq=0, Checksum=380, Data=msg5
Sender : Sent Packet
Reciver : Recived Packet
Reciver : No error found , delivering data and sending ACK
Sender : Wrong/corrupted ACK (Seq=0, Checksum=207) -> Resending
Reciver : Recived Packet
Reciver : Duplicate packet detected, resending ACK
Sender : Recived correct ACK
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
Sender : Got call 1 from above , Packet made
Sender : Seq=1, Checksum=380, Data=msg4
Sender : Sent Packet
Reciver : Recived Packet
Reciver : No error found , delivering data and sending ACK
Sender : Recived correct ACK
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
Sender : Got call 0 from above , Packet made
Sender : Seq=0, Checksum=378, Data=msg3
Sender : Sent Packet
Reciver : Recived Packet
Reciver : No error found , delivering data and sending ACK
Sender : Recived correct ACK
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
Sender : Got call 1 from above , Packet made
Sender : Seq=1, Checksum=378, Data=msg2
Sender : Sent Packet
Reciver : Recived Packet
Reciver : No error found , delivering data and sending ACK
Sender : Recived correct ACK
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
Sender : Got call 0 from above , Packet made
Sender : Seq=0, Checksum=376, Data=msg1
Sender : Sent Packet
Reciver : Recived Packet
Reciver : No error found , delivering data and sending ACK
Sender : Recived correct ACK
------------------------------------------------------------------------------------------------------------
All data sent. Transmission complete.

Final received messages at receiver:
```

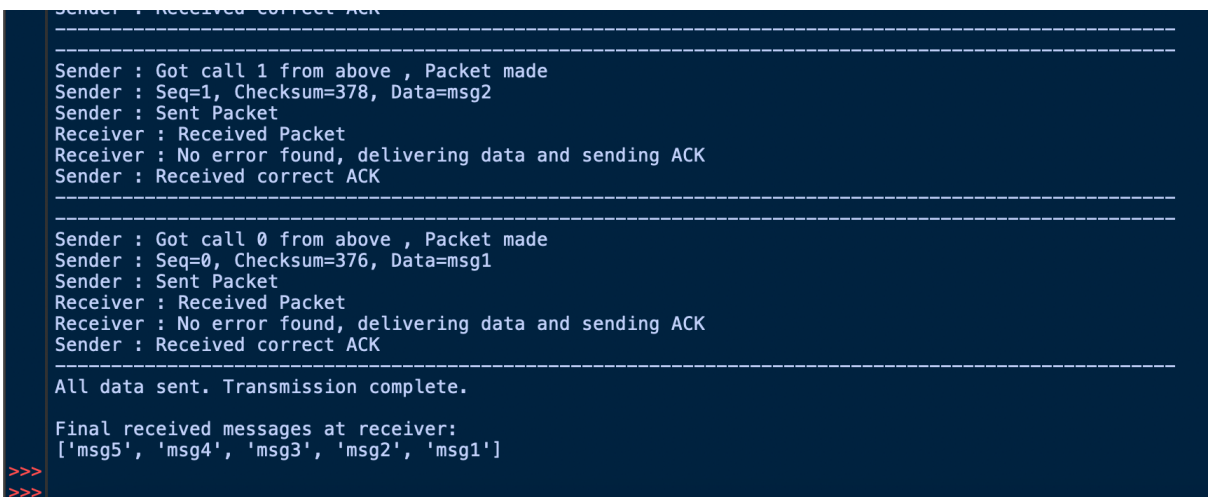
2. Implement RDT 3.0 (Reliable Data Transfer Over a Lossy Channel) Protocol

To implement this program I have used the similar structure to RDT2.2 but added some extra functions to simulate all features . Firstly I have added functions to show packet loss . I have also simulated a timer which considers a timeout if the packet is not received in 1 sec.

The parameter choices I have given in this problem is the error percentage and packet loss percentage . The user can choose the chances of the packet getting error or packet getting lost . I have given it a range between 0 to 10% to get only meaningful results. Though the input messages (packets) to be transmitted are hardcoded in the program .

The output for this is program is :



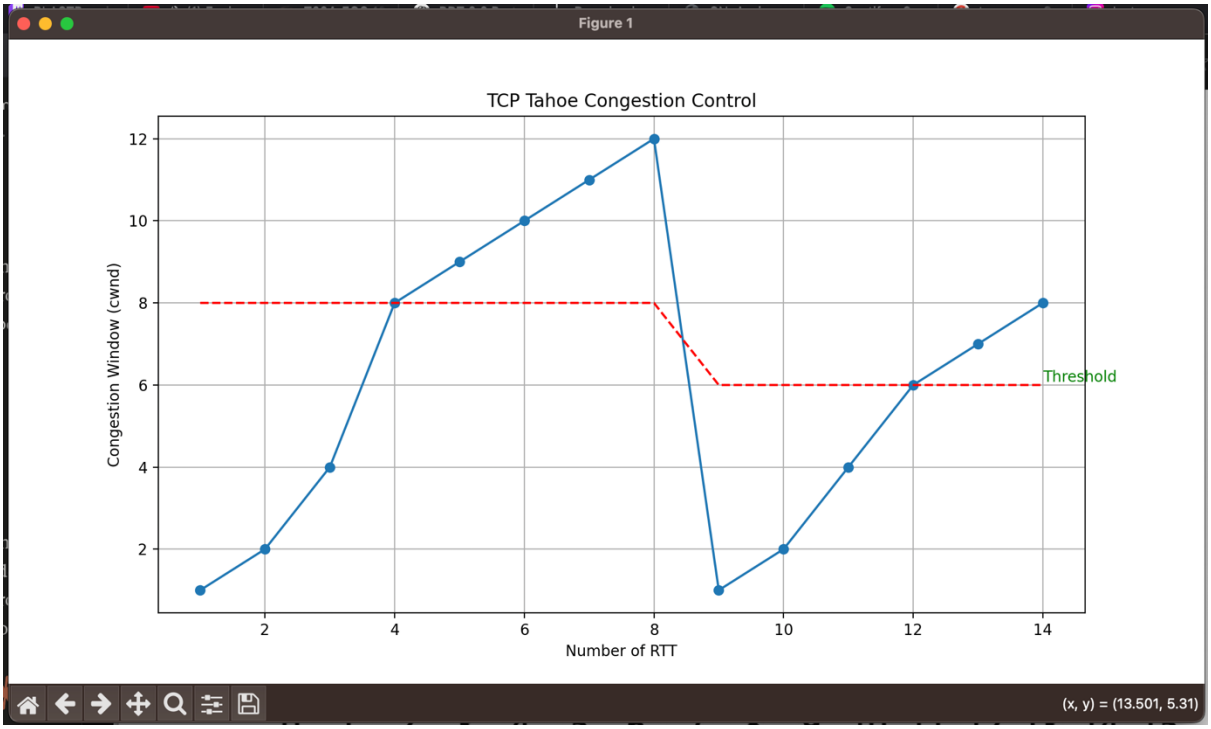


3.Implement TCP Tahoe Congestion Control Mechanism

To implement this program I have defined function for slow state and congestion control . These functions according to the condition and value of CWND either increase it linearly or exponentially or drop it . I store the values of CWND , RTTS and Threshold in a list so that I can plot it. I am using matplotlib from python to plot the graph . I used online resources to learn and apply matplotlib in this program.

The parameter choice I have given to user are MSS , RTTs , initial ssthresh and loss input . These are to define what's the maximum segment size user wants , what is the number of RTT users wants in the program , what's the initial threshold and at what time interval user wants to a packet loss

The output of this program is :



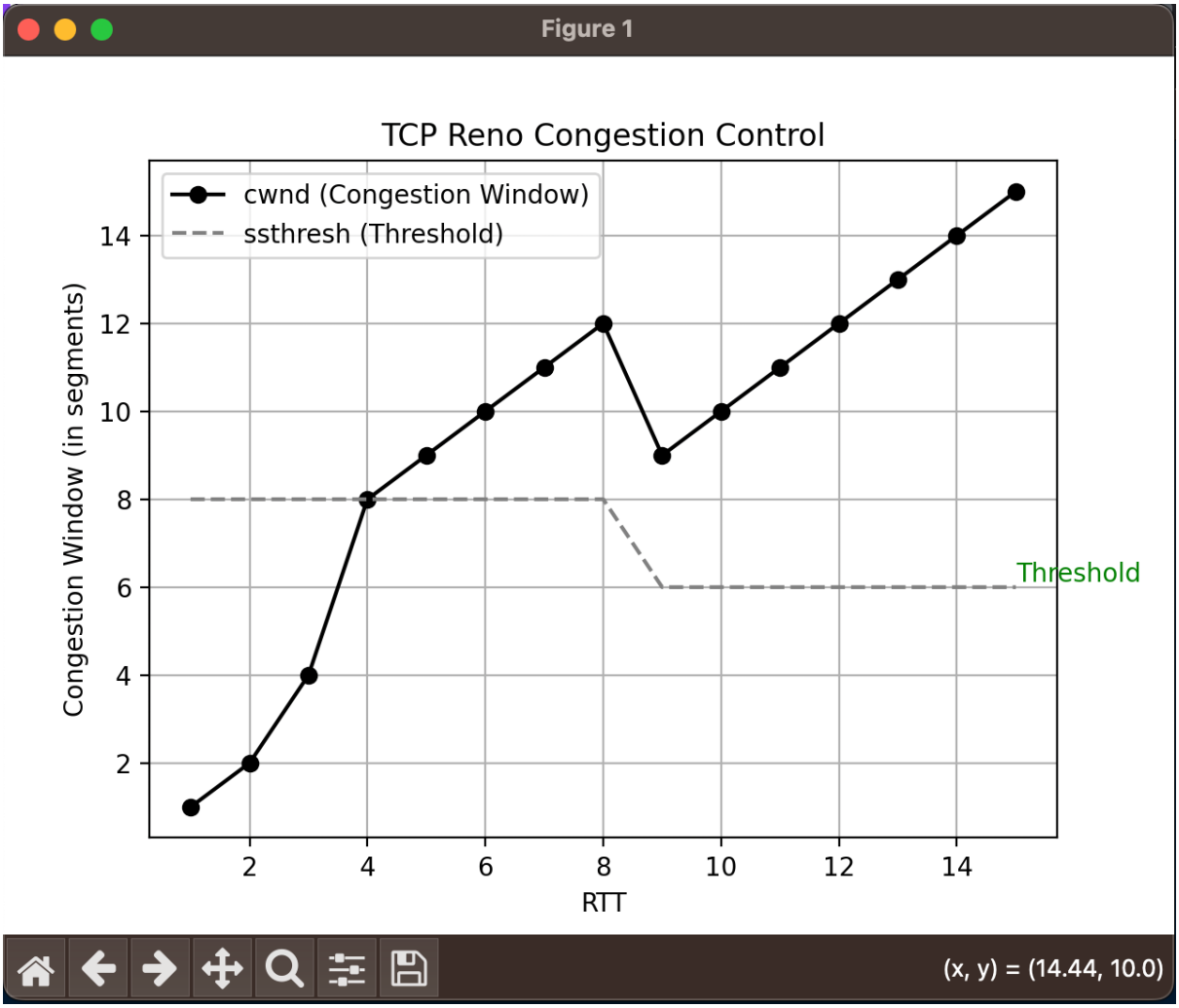

```
========== RESTART: /Users/ujjwalsingh/Desktop/college/cn ass/p1/p3.py =========
Enter MSS: 1
Enter initial ssthresh: 8
Enter total RTTs: 15
Enter loss event RTTs (space-separated): 9
[RTT 2] Slow Start: cwnd = 2
[RTT 3] Slow Start: cwnd = 4
[RTT 4] Slow Start: cwnd = 8
[RTT 5] Switching to congestion avoidance.
[RTT 5] Congestion Avoidance: cwnd = 9
[RTT 6] Congestion Avoidance: cwnd = 10
[RTT 7] Congestion Avoidance: cwnd = 11
[RTT 8] Congestion Avoidance: cwnd = 12
[RTT 10] Slow Start: cwnd = 2
[RTT 11] Slow Start: cwnd = 4
[RTT 12] Slow Start: cwnd = 6
[RTT 13] Switching to congestion avoidance.
[RTT 13] Congestion Avoidance: cwnd = 7
[RTT 14] Congestion Avoidance: cwnd = 8
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[1, 2, 4, 8, 9, 10, 11, 12, 1, 2, 4, 6, 7, 8]
[8, 8, 8, 8, 8, 8, 8, 8, 6, 6, 6, 6, 6, 6]
```

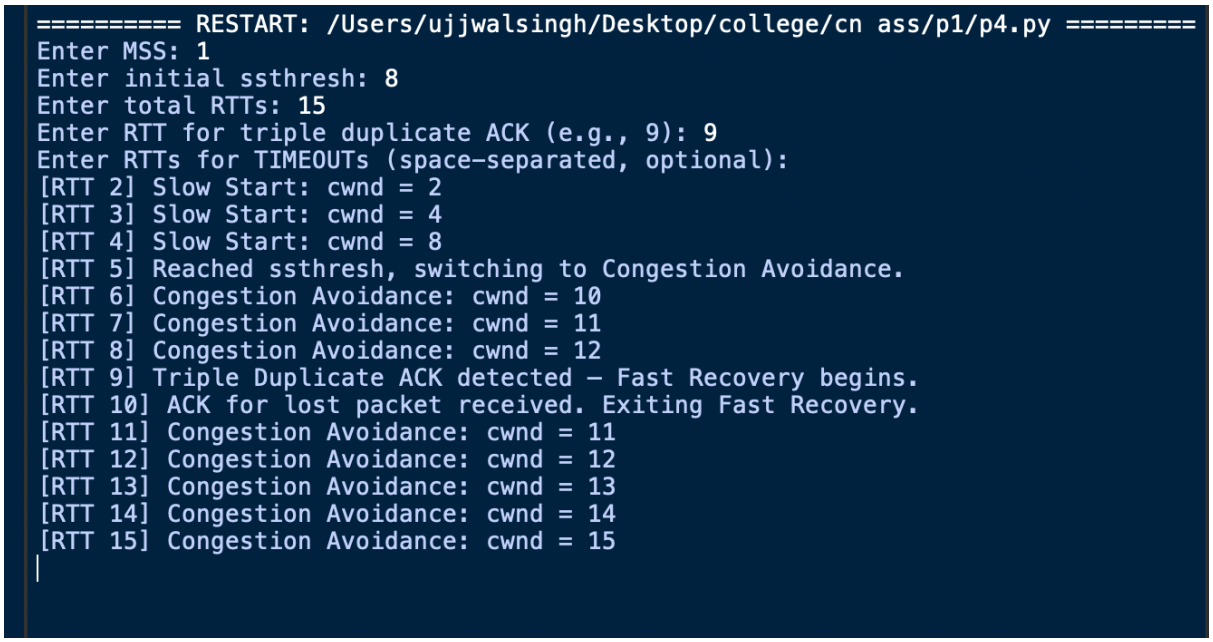
4.Implement TCP Reno Congestion Control Mechanism

To implement this program I have used the structure similar to the last program but instead of function , I have defined each state separately inside one while loop only which is the main control flow . I have added an extra state of fast recovery . I have also added option for duplicate packets along with timeout . Again I have stored CWND , threshold and RTT values in lists to plot the using matplotlib .

The parameter choice I have given to user are MSS , RTTs , initial ssthresh , loss input and duplicate ack. These are to define what's the maximum segment size user wants , what is the number of RTT users wants in the program , what's the initial threshold , at what time interval user wants to a packet loss and at what time interval user wants a duplicate packet

The output of the program is :





BY
SAARTHAK SINGH
IMT2024082

(Sorry the file is extended to 5 pages due to bigger size of the image 🙏)