Programming Assignment 2
Implementing a Reliable Data Transport Protocol

Group:

- **Ebtehal Elaraby (2).**
- **Remon Hanna (22).**
- **Shaimaa Mousa (26).**

1 Contents

- Abstraction.
- Introduction.
- System Model.
- Assumptions.
- Evaluation.
- Conclusion.
- Reference.

2 Abstraction

- This program is a simple socket layer and a reliable transfer service on top of the UDP/IP protocol which is a service that guarantees the arrival of datagrams in the correct order on top of the UDP/IP protocol, along with congestion control.this program uses techniques to guarantee the arrival of data correctly like selective repeat , stop and wait and go back n and the results shown that selective repeat technique is the best of them.
- This assignment discusses Reliable UDP (RUDP).
  RUDP is a simple packet based transport protocol. RUDP is based on RFCs 1151 and 908 - Reliable Data Protocol. RUDP is layered on the UDP/IP Protocols and provides reliable in-order delivery (up to a maximum number of retransmissions) for virtual connections. RUDP has a very flexible design that would make it suitable for a variety of transport uses. One such use would be to transport telecommunication signaling protocols.

3 Introduction

3.1 Purpose

This assignment seeks to implement a reliable transfer service on top of the UDP/IP protocol and implement a service that guarantees the arrival of datagrams in the correct order on top of the UDP/IP protocol, along with congestion control by uing several techniques that guarantee the arrival of data and compare between these techniques.

3.2 Background

A reliable transport protocol is needed to transport of telephony signaling across IP networks. This reliable transport must be able to provide an architecture for a variety of applications (i.e. signaling protocols) requiring transport over IP. RUDP is designed to allow characteristics of each connection to be individually configured so that many protocols with different transport requirements can be implemented simultaneously on the same platform.

3.3 Method of Investigation

RUDP is designed to allow characteristics of each connection to be individually configured so that many protocols with different transport requirements can be implemented simultaneously on the same platform. These techniques are Selective repeat , stop and wait and Go Back N.

- Selective repeat:[2]
  With selective repeat, the sender sends a number of frames specified by a window size even without the need to wait for individual ACK from the receiver as in Go-Back-N . The receiver may selectively reject a single frame, which may be retransmitted alone; this contrasts with other forms of ARQ, which must send every frame from that point again. The receiver accepts out-of-order frames and buffers them. The sender individually retransmits frames that have timed out.
- Stop and wait:[2]
  A stop-and-wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with both transmit and receive window sizes equal to 1 and more than one respectively . After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal. After receiving a good frame, the receiver sends an ACK. If the ACK does not reach the sender before a certain time, known as the timeout, the sender sends the same frame again. Timer is set after each frame transmission. The above behavior is the simplest Stop-and-Wait implementation.
- Go Back N:[2]
  Go-Back-N is a protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1. It can transmit N frames to the peer before requiring an ACK.

4 System Model

4.1 External Documentation:

4.1.1 For server:

Just one class main.cpp that has the following methods:

1. serve method:
   First, it initializes the server sockaddr_in struct and create a socket then use it to bind. Then it loop forever to listen to requests, accept coming connections and then create a process for each a client from different port and create another socket to serve this client. Then it checks the protocol type and lead it to the corresponding method if protocol is selective repeat call method sendBySelectiveRepeat, if protocol is stopand wait call method sendByStopAndWait else call method sendByGBN.
2. sendBySelectiveRepeat:
   This method sends the file packet by packet if the packet loss or time out happened, check method receiveAckSelectiveRepeat to check if the acknowledgement arrived or timeout happened,it resends the loss packet only.
3. sendByStopAndWait:
   This method sends the file packet by packet if and only if the packet acknowledgement arrived by checking receiveAckStopAndWait if arrived send the next packet else resend the same packet.
4. sendByGBN:
   This method sends the file packet by packet if the packet loss or time out happened ,check method receiveAckGBN to check if the acknowledgement arrived or timeout happened,it resends the all packets from the loss one to the last one had beent sent.
5. StopAndWaitTimer:
   Check if the time of the sent packet is exceeds or not if exceeds signal that time out happened.
6. SelectiveRepeatTimer:
   Check if the any one of sent and not acknowledged packet time is exceeds or not if exceeds signal that time out happened.
7. GBNTimer:
   Check if the oldest sent and not acknowledged packet time is exceeds or not if exceeds signal that time out happened.

4.1.2 Data Structures for server:

Two structs :

1. struct ack_packet
   {
   uint16_t cksum; /* Optional bonus part */
   uint16_t len;
   uint32_t ackno;
   };
   //holds the acknowledgement packet
2. struct packet
   {
   /* Header */
   uint16_t cksum; /* Optional bonus part */
   uint16_t len;
   uint32_t seqno;
   /* Data */
   char data[500]; /* Not always 500 bytes, can be less */
   };
   // holds the file data packet by packet
3. vector¡char*¿ *frames = new vector¡char*¿; // use to store the window packets in the server side.
4. vector¡bool¿ *ackedFrames = new vector¡bool¿;
   // use to store acknowledgment packets in the server window.
5. vector¡int¿ *times = new vector¡int¿;
   // use to store times of packets in the server window.
6. vector¡int¿ *lens = new vector¡int¿;
   // use to store lengths of packets in the server window.
7. use pthread_mutex_t to handle clients requests.

4.1.3 For client:

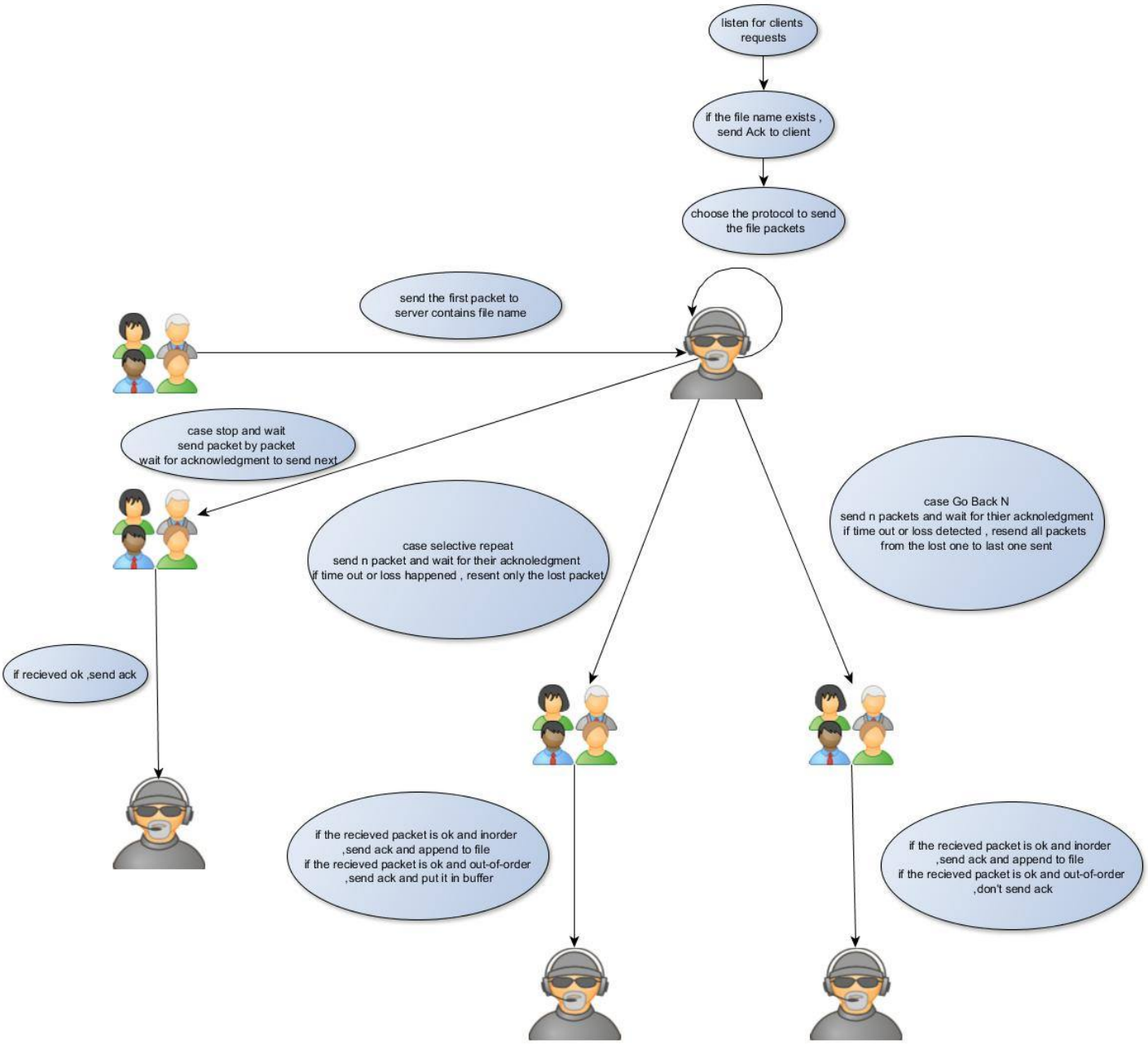
Just one class main.cpp that has the following methods:

1. ConnectToTheServer(char* host, int port):
   This method is used to create socket to connect to the server and send the requested file name to it If receive acknowledgment from server send another ack to it , then according to the protocol type call the corresponding method of receiving.
2. recieveStopAndwait():
   this method is used to receive packets from server if it is ok and not corrupted send ack that its delivered and extract packet and appent it to the file else not send ack.
3. recieveSelectiveRepeat ():
   this method is used to receive packets from server if it is ok and not corrupted and in order send ack that its delivered and append it to the file.if it is out of order and ok ,send ack and store in buffer else not send ack.
4. recieveGBN ():
   This method is used to receive packets from server if it is ok and not corrupted and in order send ack that its delivered and append it to the file .if it is out of order and ok ,not send ack.
5. sendAck():
   send acknowledgment to server if packet is not corrupted

4.1.4 Data Structures for client:

1. vector¡packet*¿ buffer; //to store out of order packets in case of selective repeat.
2. char* fileName; // is the requested file name from client to server.

4.2 Run model:



---

5 Bonus Part

5.1 Checksum:

Check sum is used to detect error that can occur while transmitting a packet it splits any packet into pairs of 16 bit then sum these pairs and adding 1 to if there is a carry then it takes one's complement of the result to represent the sent checksum at the client side we do the same for the arriving packet to check whether it's corrupted or not.

6 Evaluation

6.1 The average of 5 consecutive runs:

1. For PLP = 1

	Time	Throuput
Selective repeat	5.3626	1477.8279
Stop and wait	4.7494	1668.63
Go Back N	7.825	1012.7795

2. For PLP = 5

	Time	Throuput
Selective repeat	8.9752	882.988
Stop and wait	11	720.4545
Go Back N	8.7166	909.1847

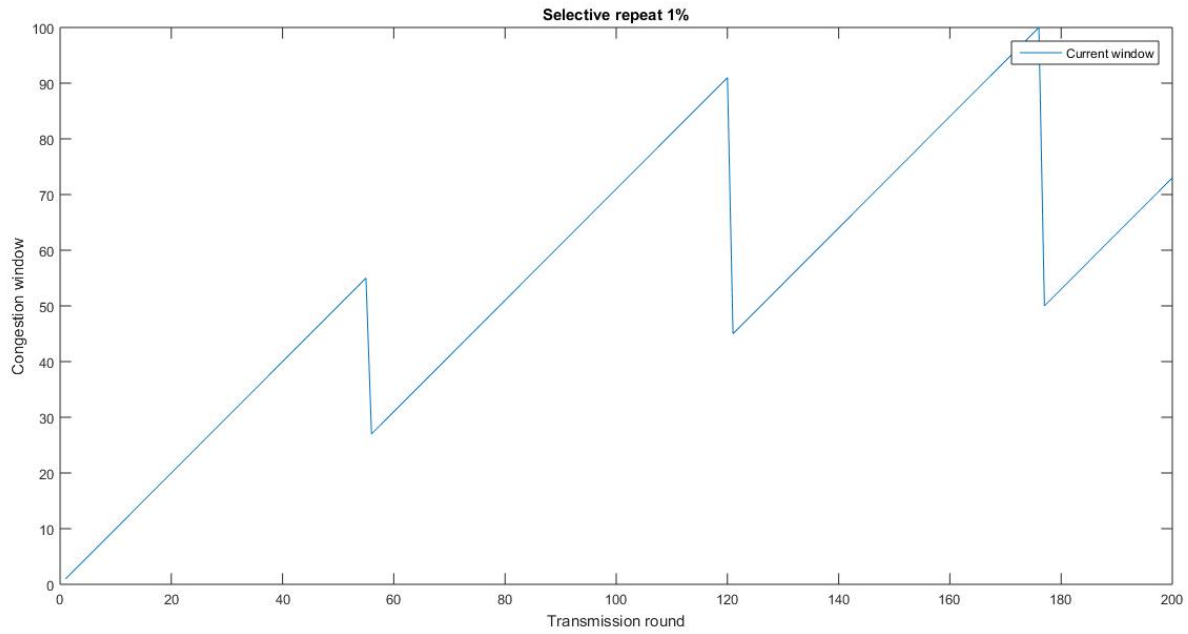
3. For PLP = 10

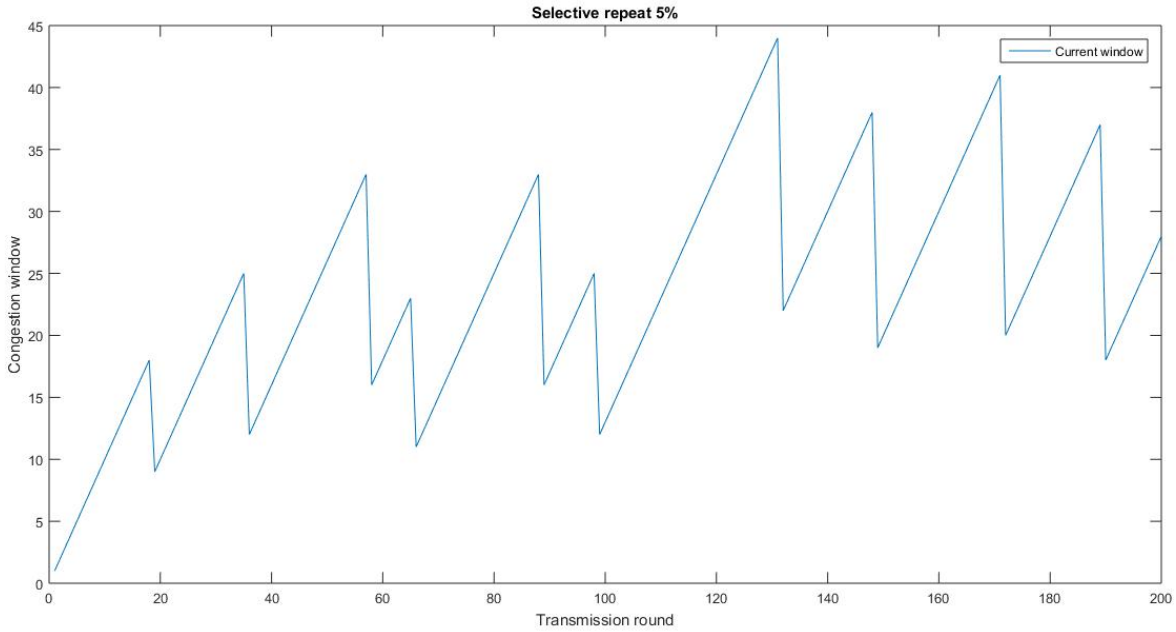
	Time	Throuput
Selective repeat	9.4566	838.039
Stop and wait	14.624	541.917
Go Back N	11.7424	674.9

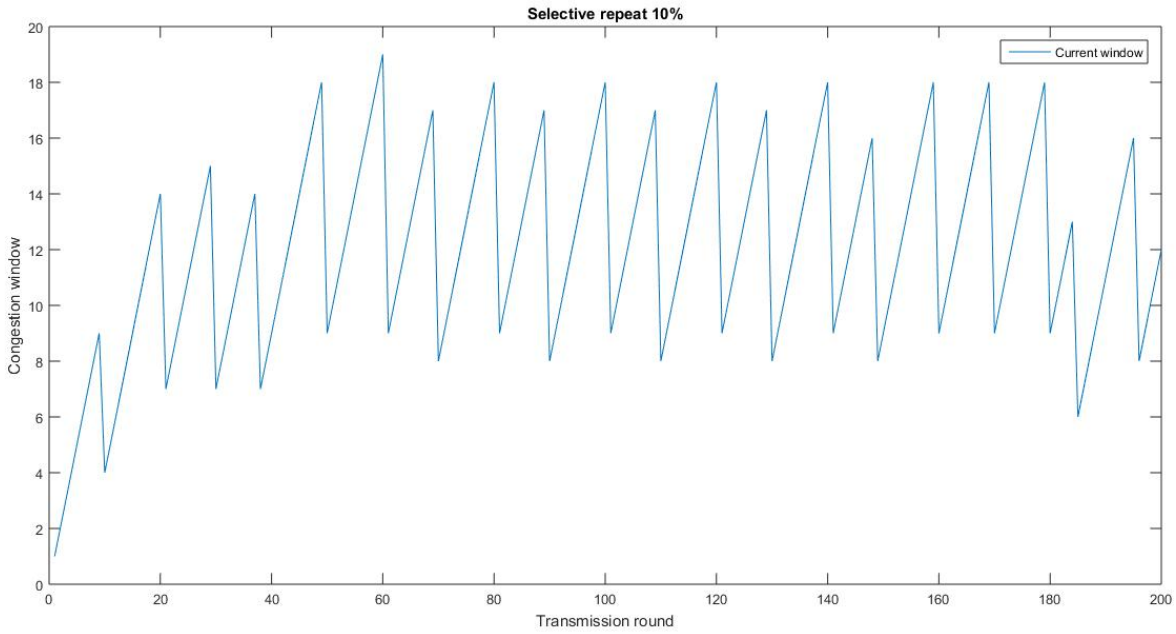
4. For PLP = 30

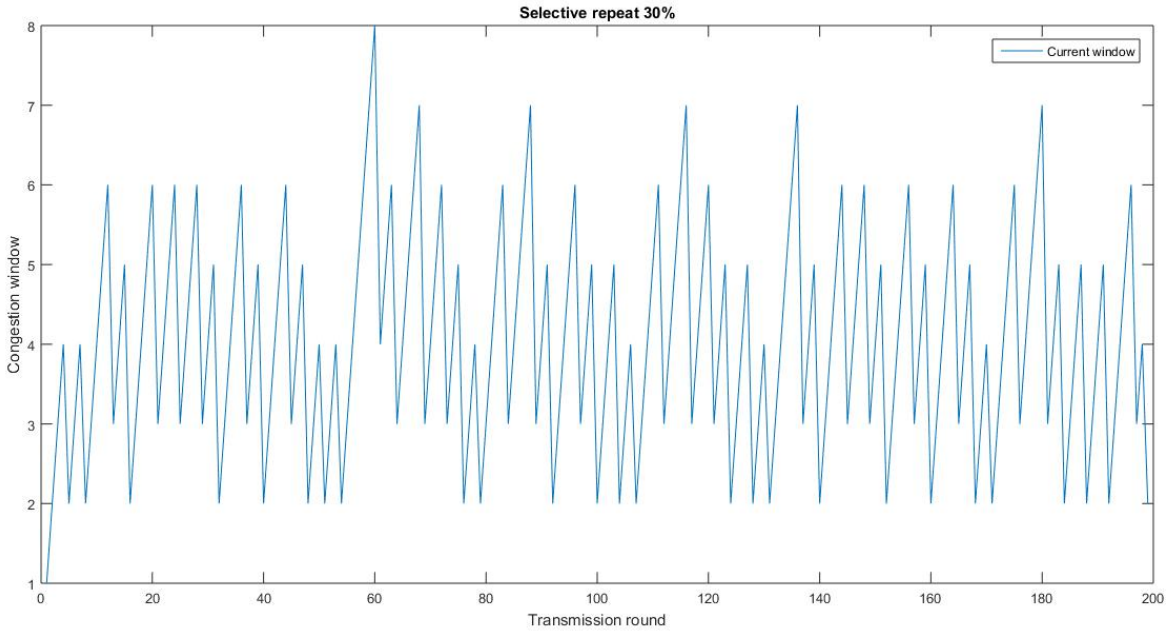
	Time	Throuput
Selective repeat	18.0334	439.462
Stop and wait	39.6424	199.9
Go Back N	20.9564	378.166

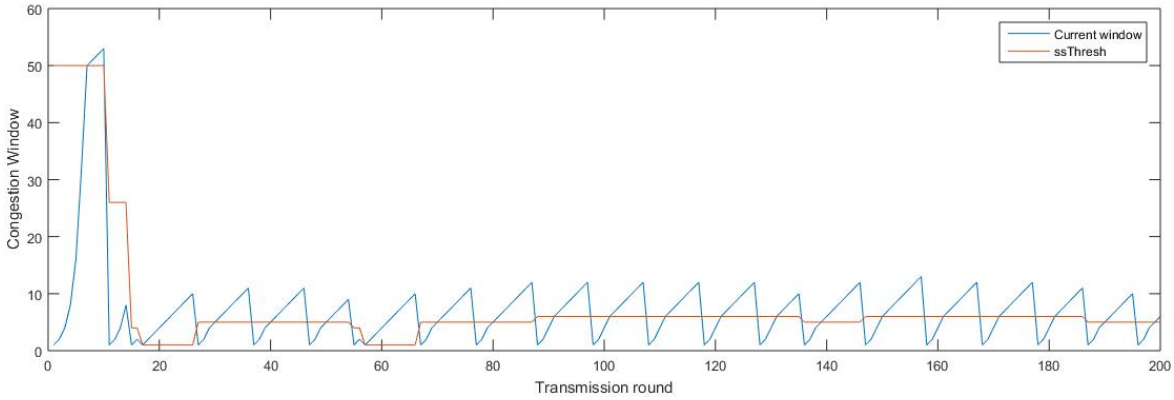
6.2 For the congestion control:

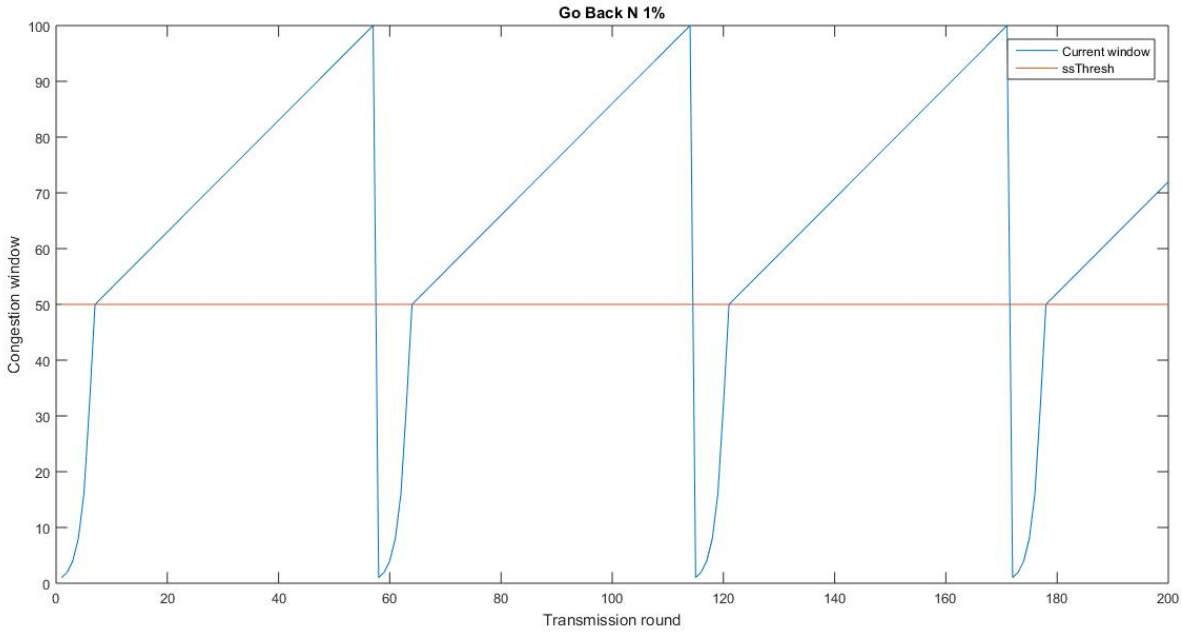


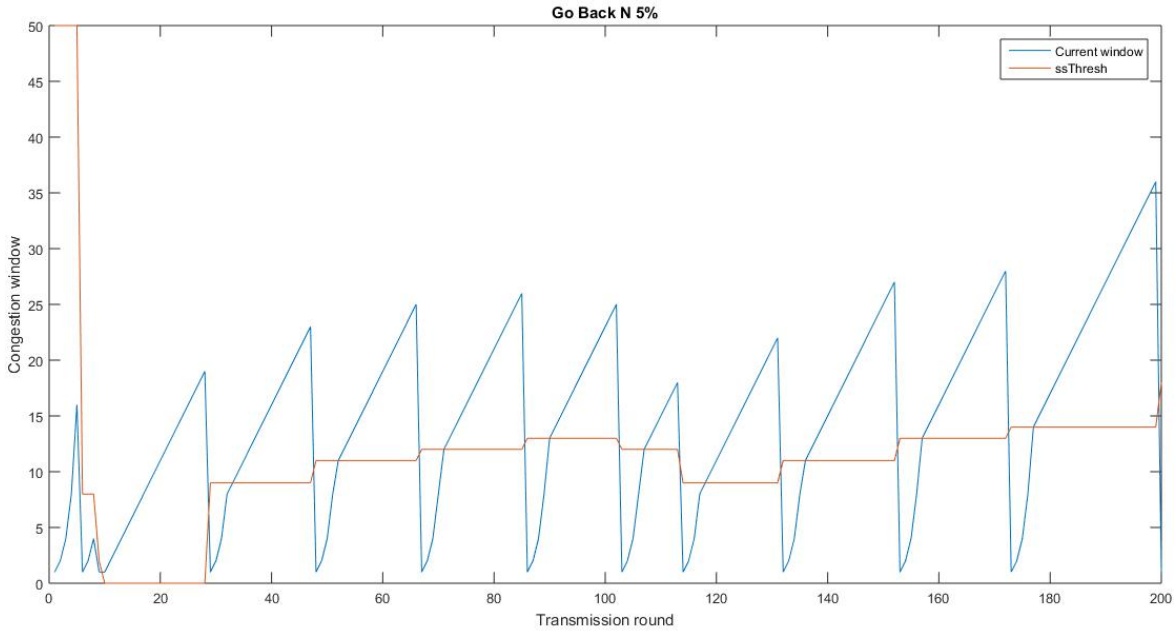

Selective repeat 5%
Current window
Congestion window
Transmission round
0
5
10
15
20
25
30
35
40
45
0
20
40
60
80
100
120
140
160
180
200

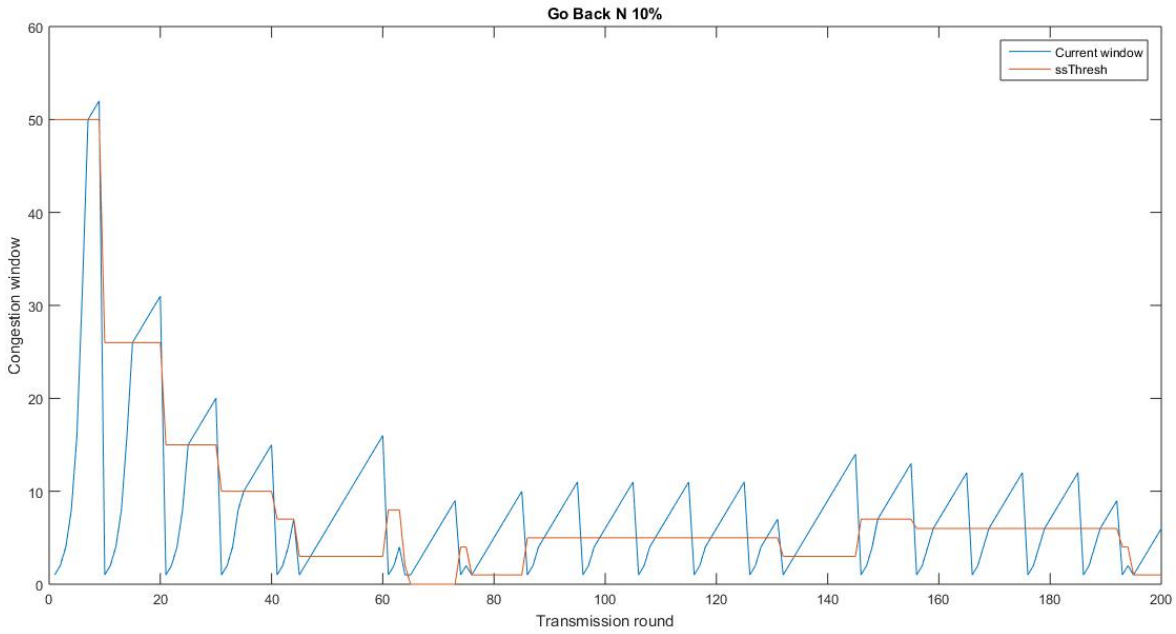

Selective repeat 10%
Current window
Congestion window
Transmission round
0
2
4
6
8
10
12
14
16
18
20
0
20
40
60
80
100
120
140
160
180
200

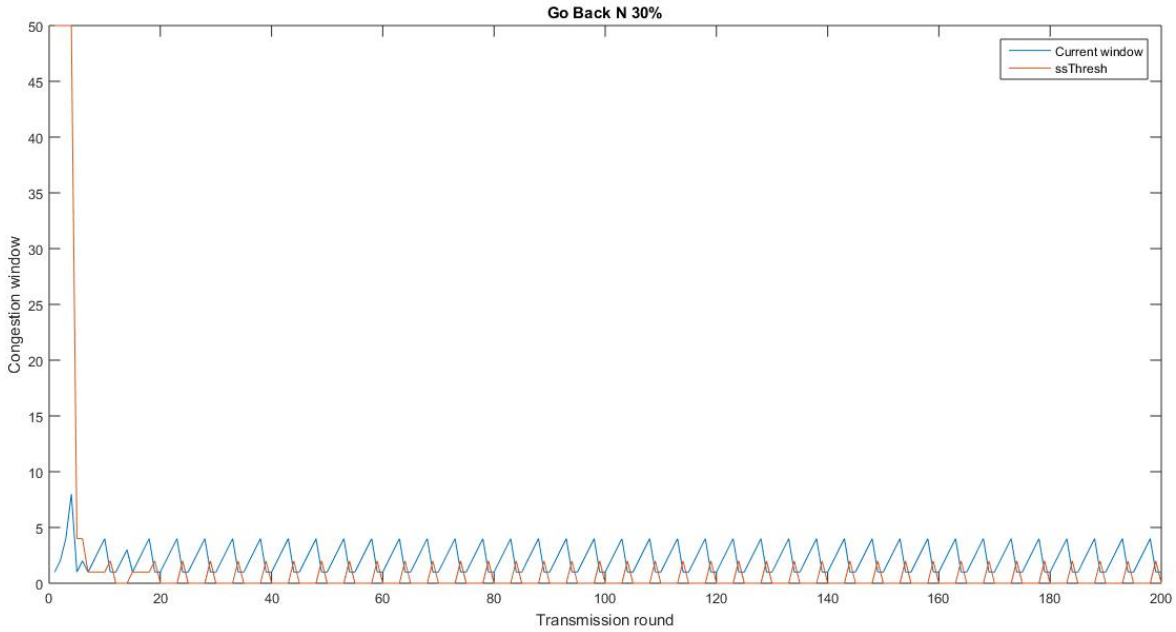

Selective repeat 30%
Current window
Congestion window
Transmission round


Current window
ssThresh
Congestion Window
Transmission round


Go Back N 1%
Current window
ssThresh
Congestion window
Transmission round


Go Back N 5%
Current window
ssThresh
Congestion window
Transmission round
0
5
10
15
20
25
30
35
40
45
50
0
20
40
60
80
100
120
140
160
180
200


Go Back N 10%
Current window
ssThresh
Congestion window
Transmission round
0
10
20
30
40
50
60
0
20
40
60
80
100
120
140
160
180
200



7 Conclusion

We can implement reliable data transfer over UDP using more than one technique . Here we use Selective repeat , stop and wait and Go Back N .The best technique of them is Selective repeat which resend just the loss packet then Go Back N which resend N packets in case of loss base packet and finally stop and wait which the worest performance between the three techniques. Using of congestion control ,improve the throughput and performance over selective repeat and Go-Back-n.

8 Reference

1. Beej's Guide to Network Programming.
2. Computer Networking - A Top-Down Approach, 6th Edition.