**ASSIGNMENT REPORT**

**Additional Classes and Files Used**

* AckData: Its object is used to send the details about the acknowldgement.
* InitiateTransfer: Used to send the initial SYN data which includes packet size, window size, Number of packets to the receiver.
* SegmentData: Used to send the data, here we are sending a character as data. We are randomly generating the character data.
* The InitialConfig.txt is the input file which contains the protocol type, Packet size, window size. So to test for Go Back N type GBN on top of InitialConfig.txt file and for Selective repeat type SR

**How To Run The Program**

* Go to the Codebase folder
* Start the server program by typing the following command
  + Java MyReceiver 8080
  + 8080 – port number
* Start the client program by typing the following command
  + Java MySender InitialConfig.txt 8080 20
  + 8080 – port number
  + 20 – number of packets to be sent
* The InitialConfig.txt is the input file which contains the protocol type, Packet size, window size. So to test for Go Back N type GBN on top of InitialConfig.txt file and for Selective repeat type SR

**Implementation of Go Back N**

Here we are setting the window size as 4, the total number of packets send is 20.

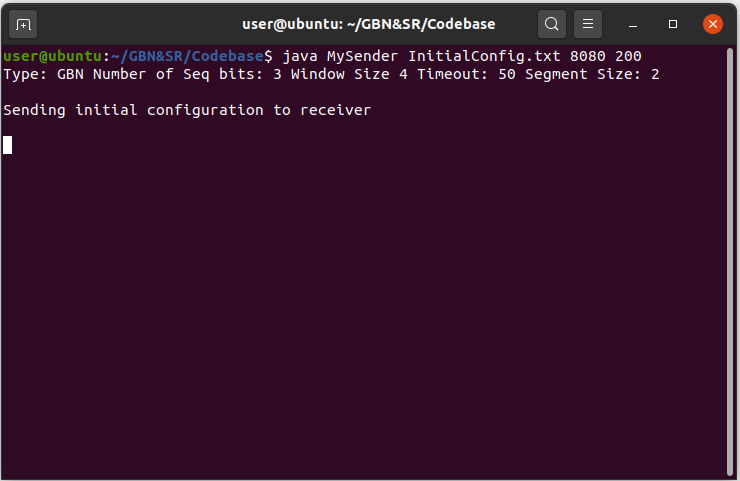
1. Initially the sender sends the synchronization packet containing initial information like packet size, window size, Number of packets, Time out to the receiver.

Figure 1 : Initial configuration

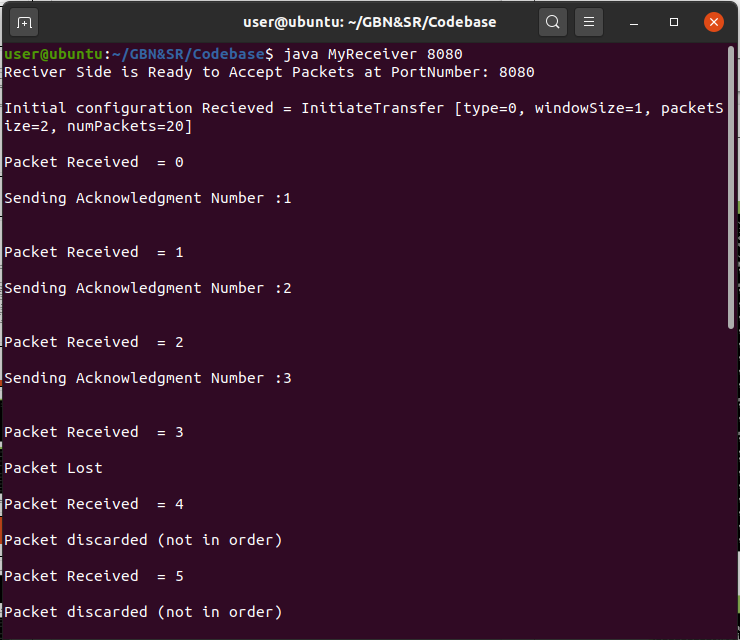
1. The following are the screen shot for the Sender and Receiver, Initially we are sending 4 packets numbered 0 to 3.But on the receiver side the packet 3 haven’t reached so the receiver prints the “Packet Lost in transmission” message also as per the GBN protocol, and the following packets(Packet 3 to Packet 5) are discarded. For this the receiver prints the message “Packet Discarded” in the command line.
2. Once the receiver correctly receives a packet the receiver sends the ACK for that packet, this is displayed in the sender window as a message “Received ACK”.
3. Once the sender receives the ACK packets send by the receiver the window is moved to accommodate the next set of packets. This is shown below in the screenshots, the sender receives ACK for packet 0, packet 1, and packet 2 after that the window is moved 3 slots and packets 4, 5, & 6 are sent.

Figure 2: Packet lost receiver

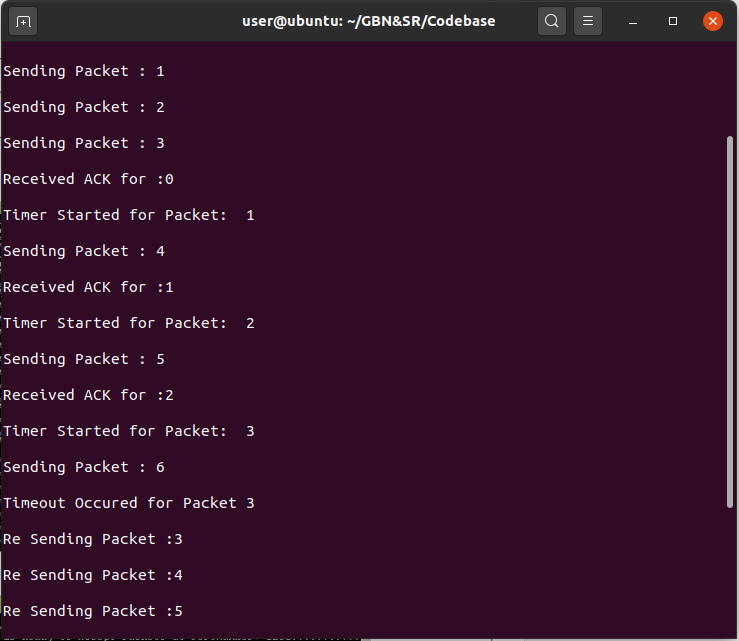


Figure 3: Packet lost sender

1. Here, the Packet 1 received by the receiver have checksum error so that packet is discarded, and the remaining packets are discarded. The timer for packet 1 is started in sender as in fig (4) and when the timeout occurs the packet 11 is resend again and the receiver receives it and delivers the remaining packets in buffer to the application.

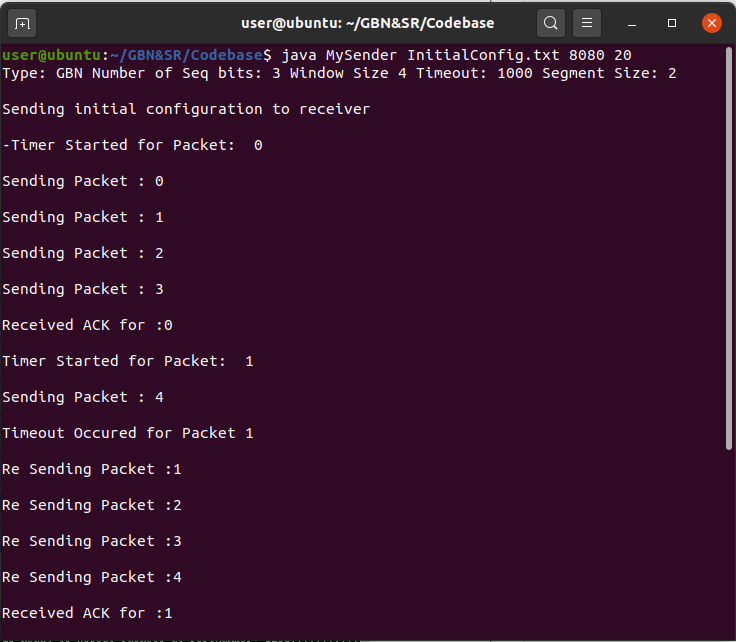


Figure 4: Checksum error sender

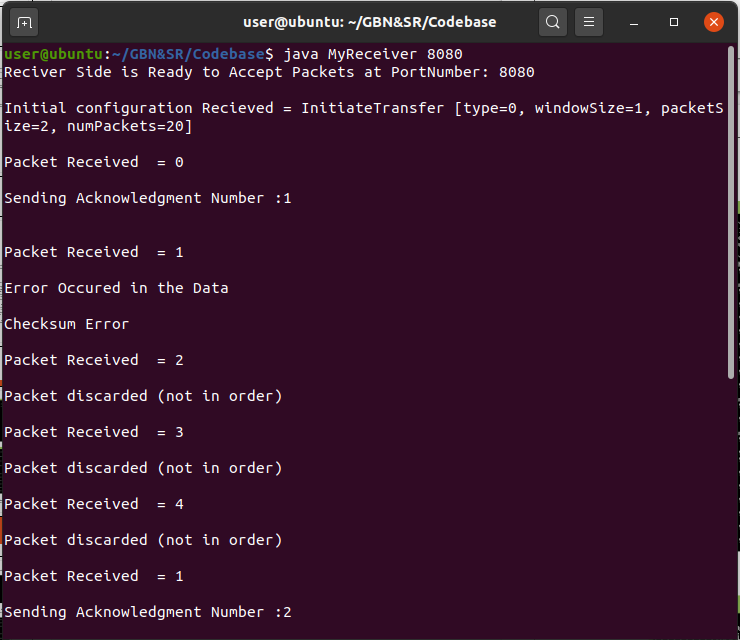


Figure 5: Checksum error receiver

**Implementation of Selective Repeat**

1. Initially the sender sends the synchronization packet containing initial information like packet size, window size, Number of packets to the receiver.
2. We are setting the window size as 4 and the number of packets as 20. The sender sends the packet 0 – 3 in the first window, and receiver sends back the acknowledgement.
3. In Fig (7), the packet 1 is lost in transition, so the packets coming after that ie, packet 2 – 4 is stored in the application buffer, and the timer in the sender for packet 1 starts, since the ACK is not received within the time period(3000 ms), timeout occurs, so sender is sending the packet again. Once the receiver receives the packet 1 it sends the ACK and packet 2-4 stored in buffer is delivered to the application as shown in fig (7).

Figure 6: Packet lost receiver

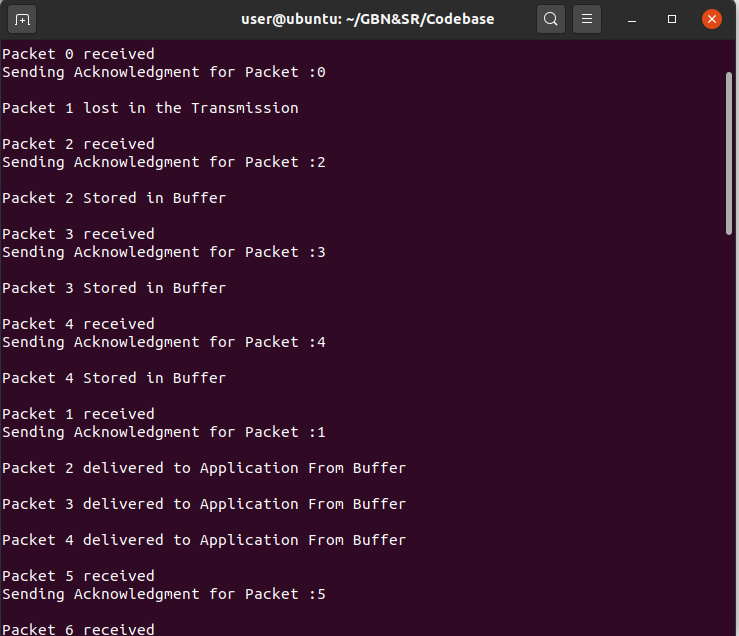
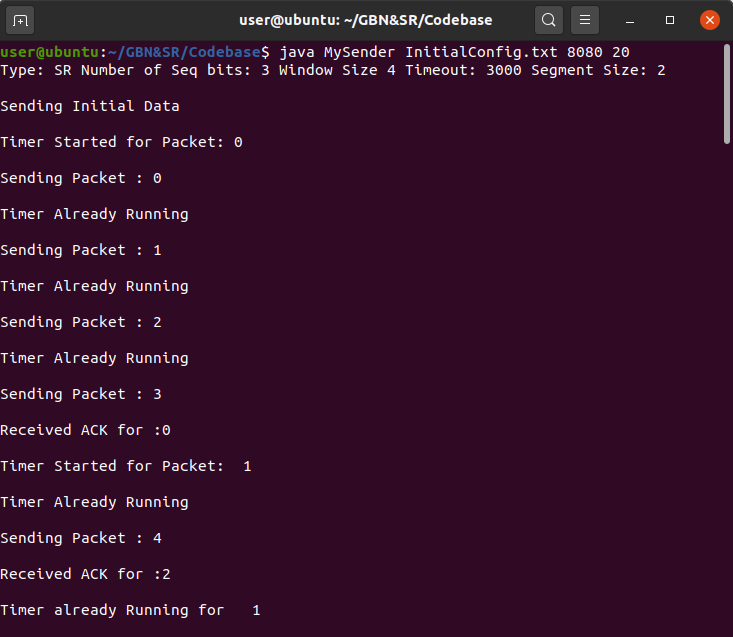


Figure 7: Packet lost sender

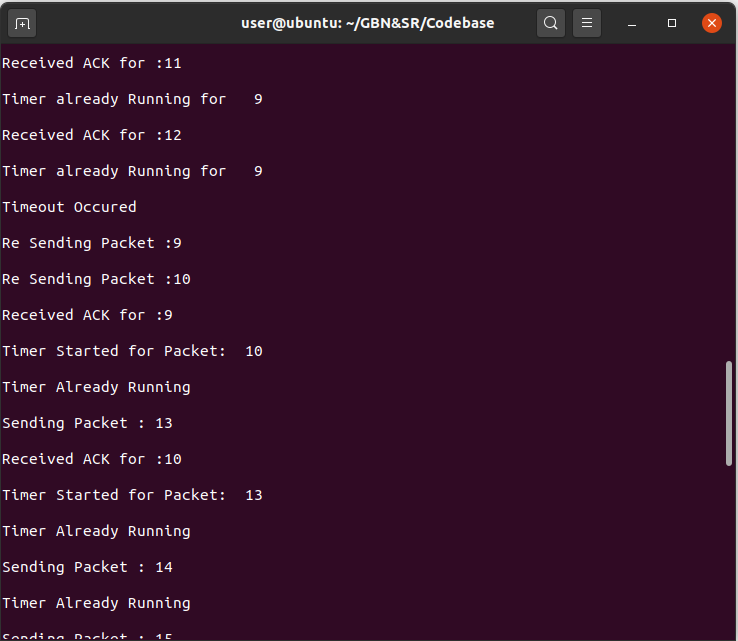
1. Here, the Packet 9 received by the receiver have checksum error so that packet is discarded and the remaining packets in the window is stored in the application buffer. The timer for packet 9 is started in sender as in fig (8) and when the timeout occurs the packet 9 is resend again and the receiver receives it and delivers the remaining packets in buffer to the application.

Figure 8: Checksum error sender

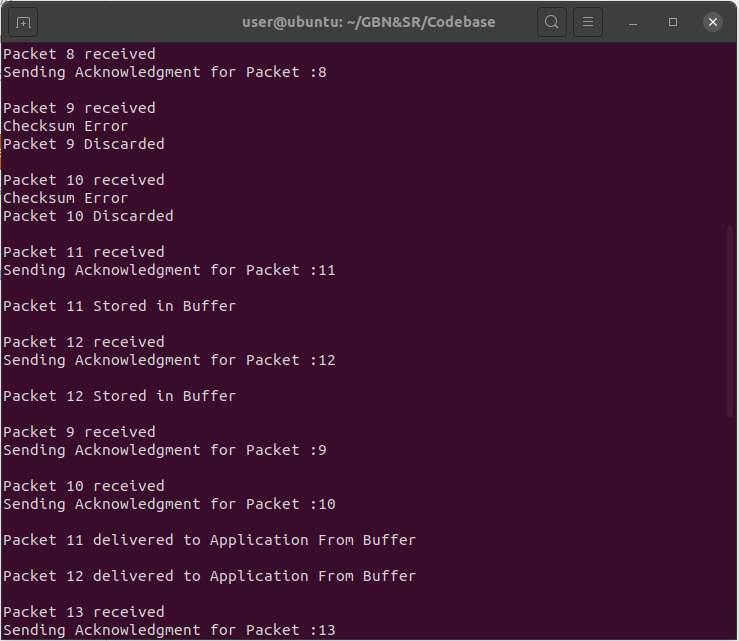


Figure 9: Checksum error receiver

**Performance comparison**

We have taken the following measurements for comparing the performance of selective repeat and go back n.

* Time taken to complete the transmission
* Package miss ratio

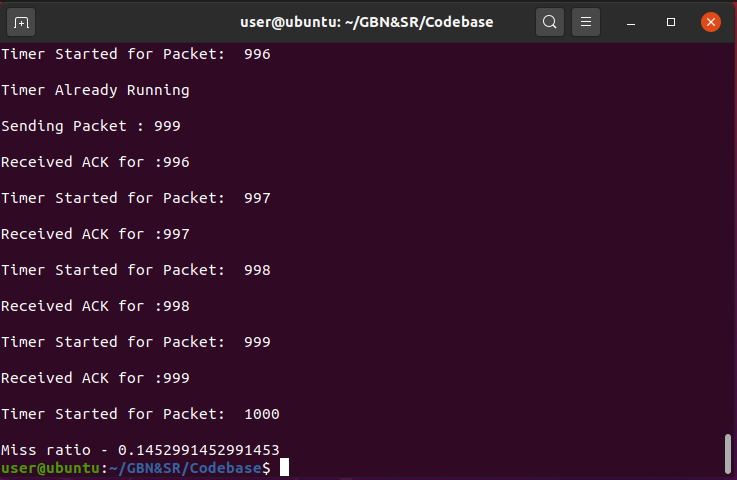


Figure 10: Package miss ratio

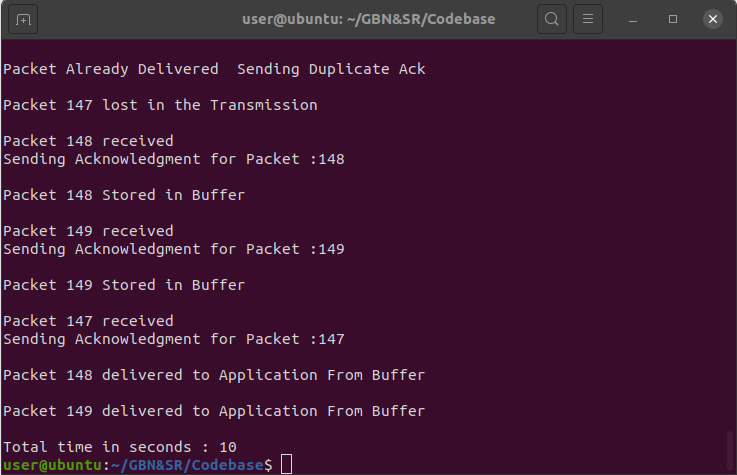
 **Time Taken to Complete the Transmission**

Figure 11: Total time taken

Here we are measuring the time taken to complete the transmission with varying number of packets

**Go Back N**

|  |  |
| --- | --- |
| Number of Packets | Time Taken |
| 150 | 16 |
| 200 | 24 |
| 400 | 47 |
| 500 | 59 |
| 1000 | 114 |

**Selective Repeat**

|  |  |
| --- | --- |
| Number of Packets | Time Taken |
| 150 | 10 |
| 200 | 15 |
| 400 | 29 |
| 500 | 36 |
| 1000 | 73 |

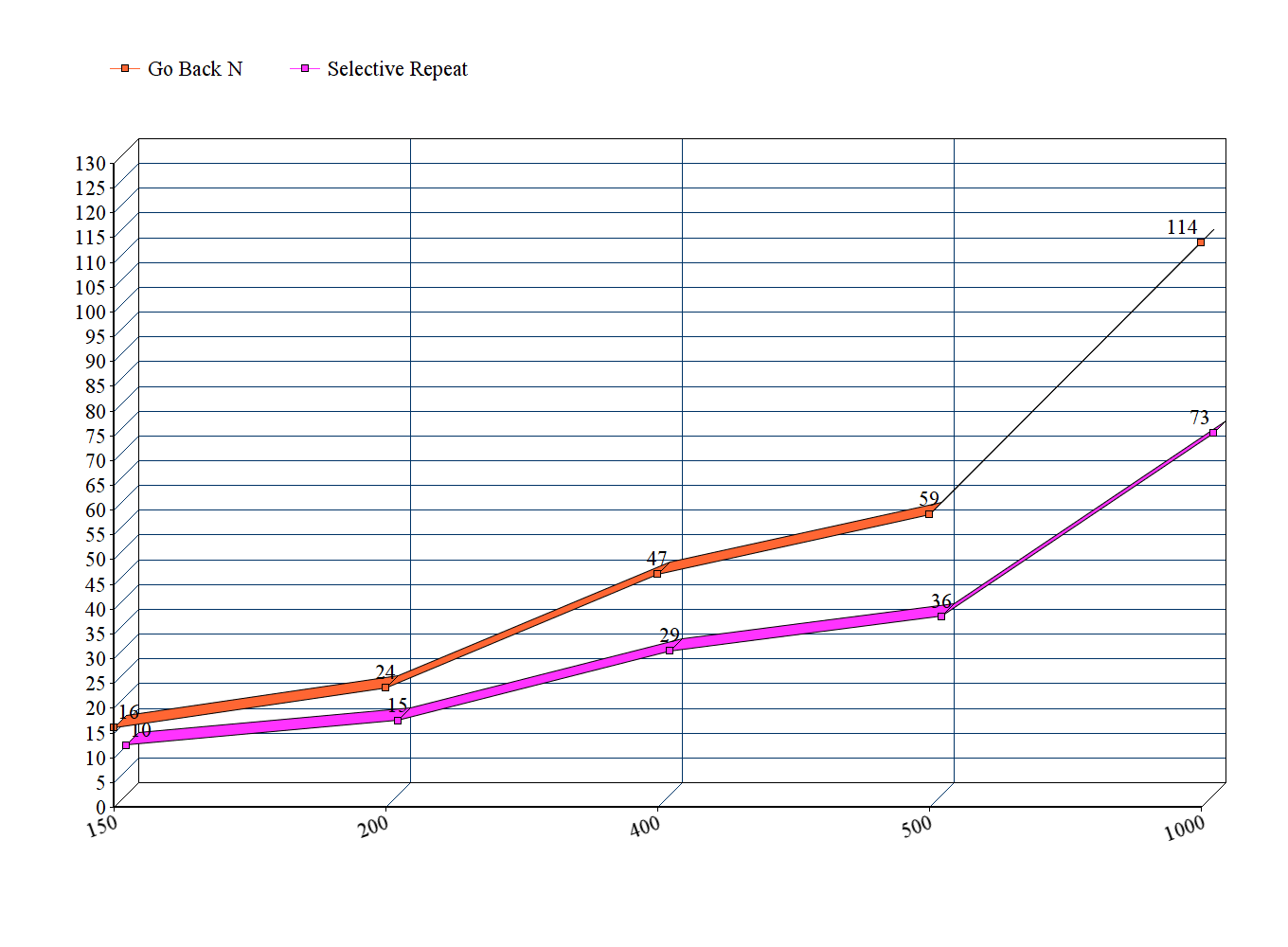


Figure 12: Time taken to deliver packets

**Package Miss Ratio**

**Go Back N**

|  |  |
| --- | --- |
| Number of Packets | Package Miss Ratio |
| 150 | 0.52 |
| 200 | 0.5 |
| 400 | 0.49 |
| 500 | 0.48 |
| 1000 | 0.48 |

**Selective Repeat**

|  |  |
| --- | --- |
| Number of Packets | Package Miss Ratio |
| 150 | 0.17 |
| 200 | 0.18 |
| 400 | 0.21 |
| 500 | 0.15 |
| 1000 | 0.14 |

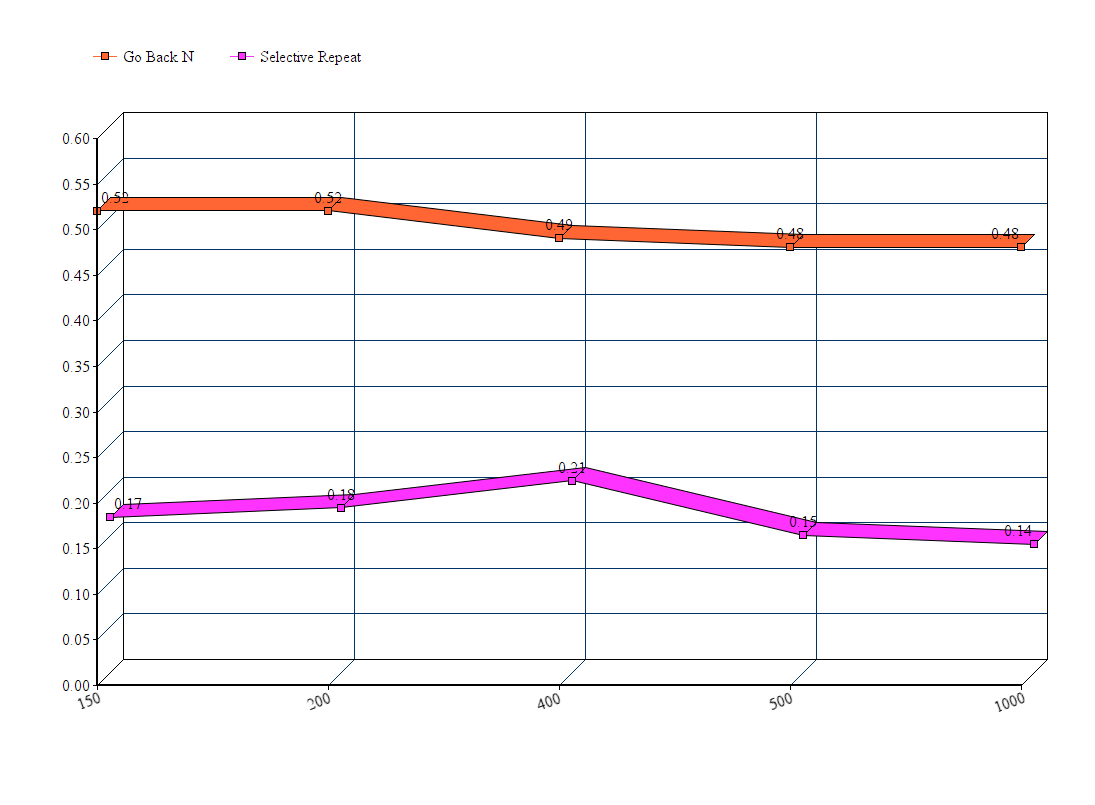


Figure 13: Package miss ratio

**Conclusion**

* Selective Repeat implementation of the UDP protocol takes less time to deliver the packets when compared with Go Back N implementation of UDP
* Selective Repeat approach also shows a low package miss ratio when compared to Go Back N approach