Technical Report
Assignment-4
CS3205,Introduction to Computer Networking

CS18B047,Viswanath Tadi

Aim:

The aim of the assignment is to understand and implement selective-repeat and go-back-n protocols.

Introduction:

Go-back-n(GBN)
In this protocol, window size is used. Using the sliding window protocol described above. But when error occurs, such as ACK is lost, or transmitted frame is lost, or excessive delay occurs after the timeout, all frames with sequence numbers within the sliding window are retransmitted. In GBN, ACK for frames implies all earlier frames are also received correctly. GBN requires full duplex link since during the forward transmission, backward ACKs are also in progress.The choice of window size :

- Window size should be large enough to contain all the sequence numbers of frames on their way to the receiver.
- The sender needs to have a buffer of window size to store all unacknowledged frames;
- Having large window size improves efficiency when the channel is very reliable. But it will also reduce efficiency when there are errors since the whole window's worth of frames have to be retransmitted. Having small window size improves efficiency when the channel is very unreliable.
- There is no need in the receiver to have buffers to store out of sequence frames. Therefore, the receiver has a window size 1.

Therefore, window size is a trade-off factor in determining the overall efficiency, depending on the reliability of the link itself.

Selective-repeat(SRP)

This protocol is mostly identical to GBN protocol, except that buffers are used and the receiver, and the sender, each maintain a window of size .

- Suppose the sequence number of frames is generated modulo , they are 0, 1, 2, ..
- The sender and receiver each maintain a buffer of its own window size
- When there is an error, the receiver freezes the lower edge to the last sequence number before the lost frame sequence number;

- As long as the window size is less than , the receiver continues to receive and acknowledge incoming frames;
- The sender maintains a timeout clock for each of the unacknowledged frame numbers and retransmits that frame after timeout.
- Acknowledgement will be piggybacked to the sender. But when there is no traffic in the reverse direction, piggyback is impossible, a special timer will time out for the ACK, so that the ACK is sent back as an independent packet. If the receiver suspects that the transmission has error, it immediately fires back a negative acknowledgement (NAK) to the sender.

SRP works better when the link is very unreliable. Because in this case, retransmission tends to happen more frequently, selectively retransmitting frames is more efficient than retransmitting all of them. SRP also requires a full duplex link. backward acknowledgements are also in progress.

Simulation setup

We used a linux machine running ubuntu 20.04 LTS.We ran sender and receiver on the same machine.We further assumed that the receiver's ack always reaches the sender and in order.

Threads and their functionality

Inside the Sender files,

generatePacket_thread is used to generate new packets with packgenrate speed.

sendFirstPacket_thread sends the packet for the first time if it is the current window.

recvPacket_thread receives the acknowledgement packets.

sendPacket_thread resends the already sent packets if they timeout.

Problems due to assumptions

There are some problems I faced during the simulations.

We assumed that the acks sent by receiver are always received by the sender.This does not hold good in practical applications.Suppose if the receiver sent a ack and the sender did not receive it,the sender will keep on resending that packet but the receiver ignores that packet as it has already acked.

Another problem I faced is when resending packets.If we resend a packet due to timeout and assume that acks always come,we get duplicate acks.If the windows size is large enough,we may not face this problem.But for small window sizes,this is a issue.

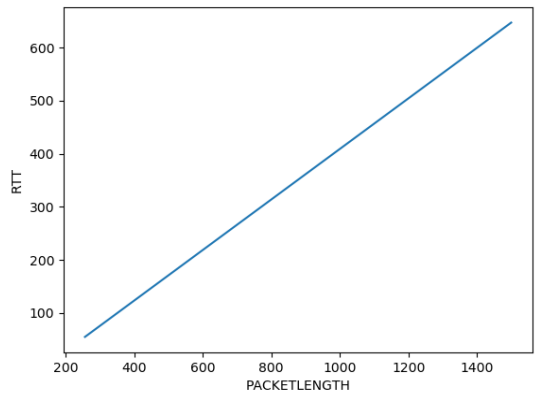
Even with the same conditions,the experiment may not yield similar results due to inherent non-determinism.

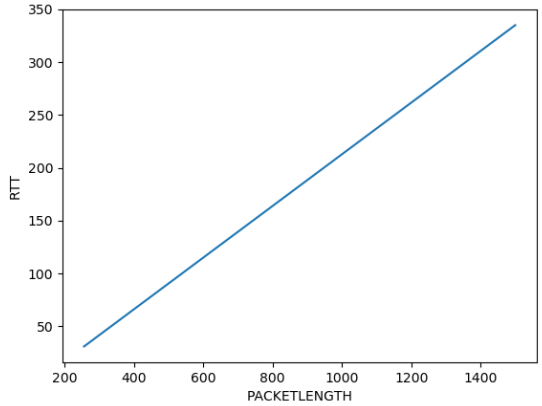
Observations

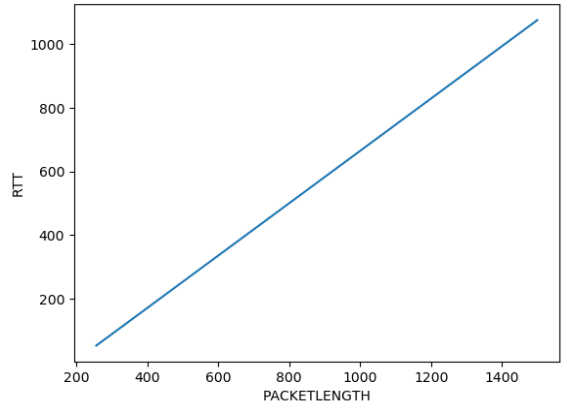
GBN

Packet transmission rate = 20

The following graphs are RTT(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}

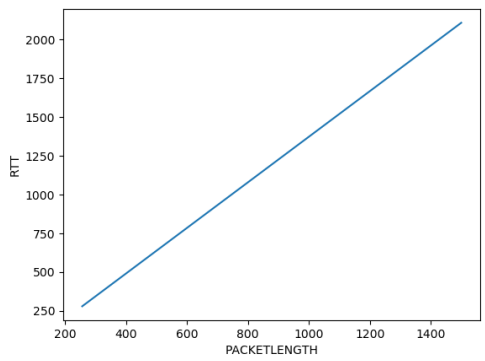


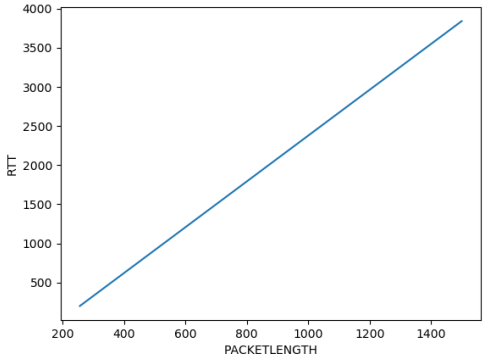


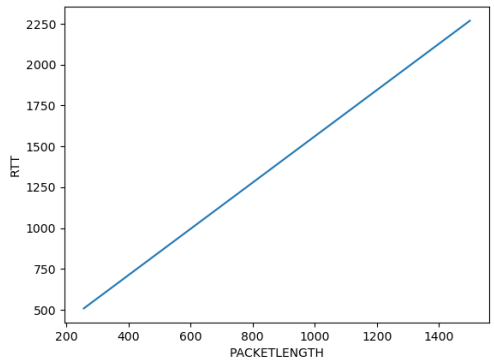


Packet transmission rate = 300

The following graphs are RTT(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}



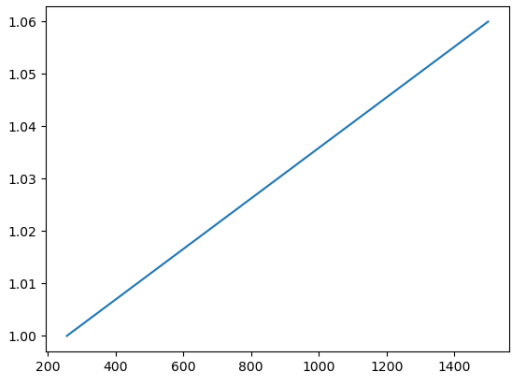


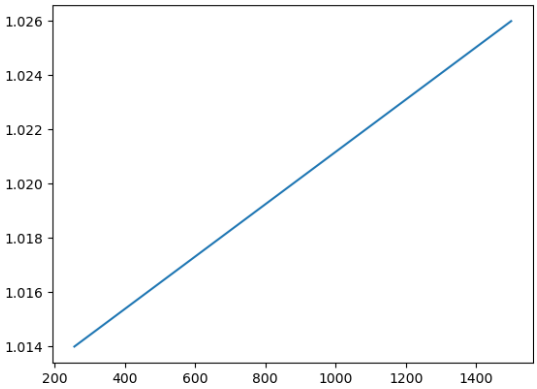


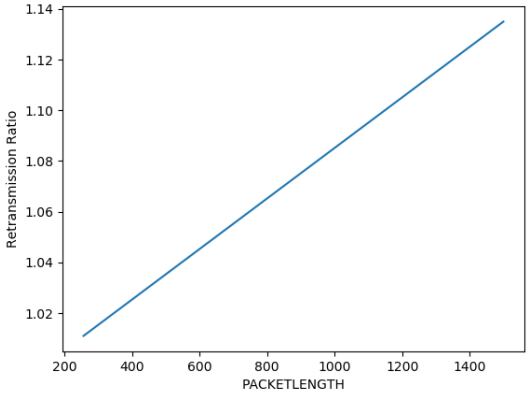
For both the cases,average RTT is increasing with packet length.

Packet transmission rate = 20

The following graphs are Retransmission Ratio(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}

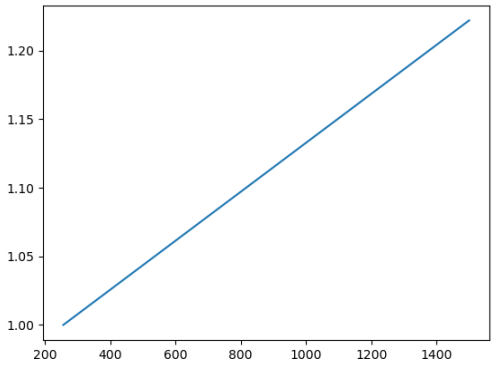


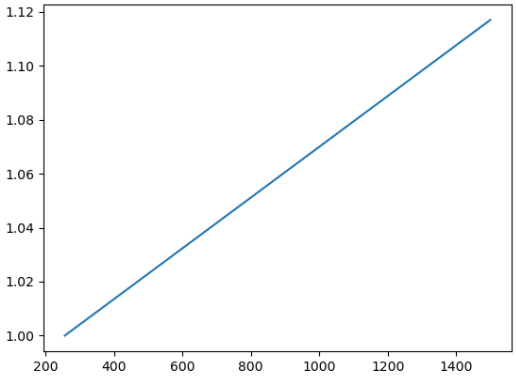


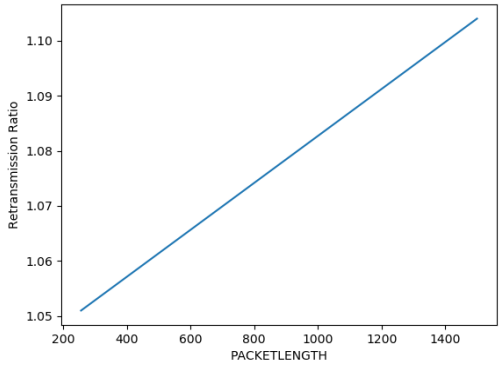


Packet transmission rate = 300

The following graphs are Retransmission Ratio(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}





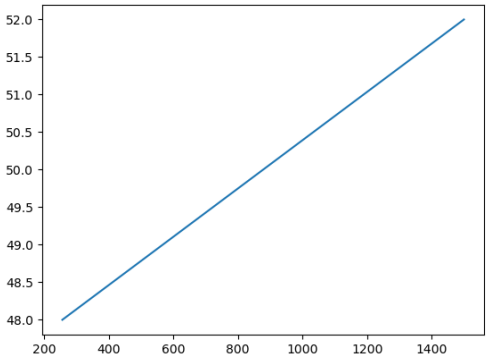


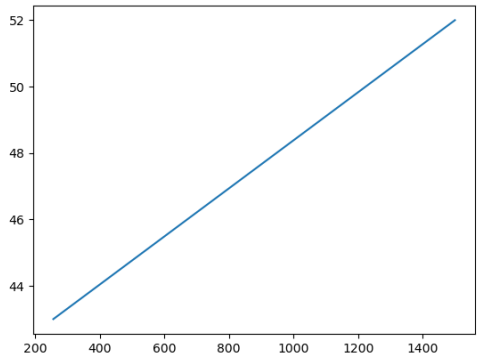
Although the retransmission ratios are small,they increase with packet transmission rate.

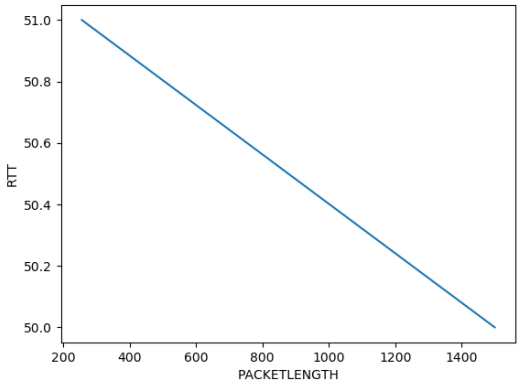
SRP

Packet transmission rate = 20

The following graphs are RTT(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}



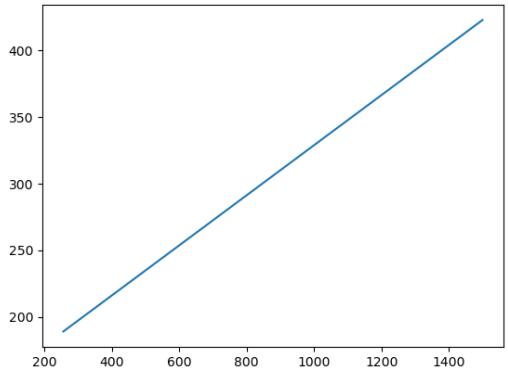


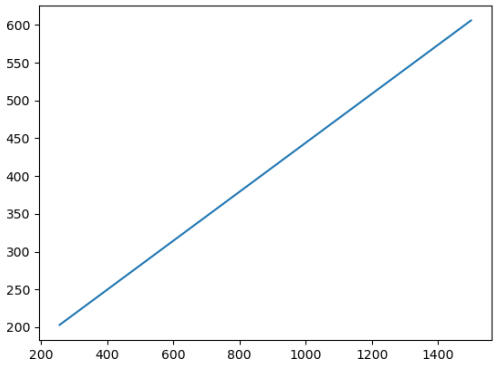


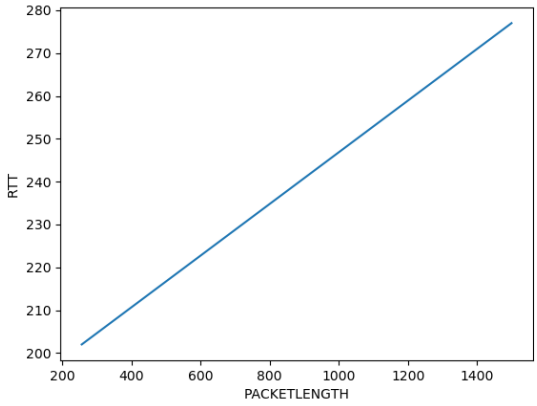
The Retransmission ratio vs packet length graph has all 1s because in SR protocol,we only send what is needed.The error rates are so low that retransmissions could not happen.

Packet transmission rate = 300

The following graphs are RTT(Y) vs Packet Length(X) for error rates 10^{-3},10^{-5},10^{-7}







Similarly here also the Retransmission ratio vs packet length graph has all 1s because in SR protocol,we only send what is needed.The error rates are so low that retransmissions could not happen.

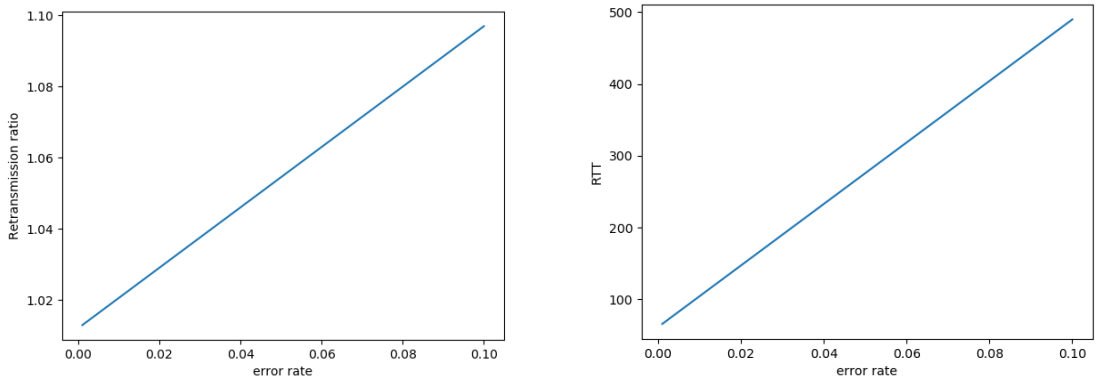
Experiment with high error rate.

Retransmission rate = 20

Packet length = 256

Protocol = SRP

Error rate = 0.001 vs 0.1



From the above plots, it can be observed that retransmission ratio and RTT are increasing with error rate.

Learnings

We understood the difficulties in implementing the SRP and GBN protocols.Synchronization is very difficult to achieve.Our implementation has to be robust enough to face the challenges posed by the unreliable medium.We also gained some insightful observations as to see how the network reacts while we change parameters involved in these protocols.

Additional thoughts

The assumption of the problem statement is controversial.Although the goal of the assumption is to ease the assignment,it makes it tougher.The udp packets need not be in-order and sometimes need not be delivered also.This makes debugging even harder as to see if the mistake is committed by us or udp.Honestly, I don't see any other way of handling this.

Conclusion

From the high error experiment, we found out that RTT and retransmission ratios are increasing with error rate.It is natural because when the packets are lost,multiple sends are needed and more time will be consumed.
Similarly,we observed from GBN and SRP experiments that RTT and retransmission rates increase with packet length and transmission rate.
If the packet length is high,the RTT will increase as sometimes it may be carried in chunks.
If the transmission rate is high, you keep on sending the packets if they are within window size. So if some packet misses, you have to send many in GBN protocol.

References

Read more about GBN here

Read more about SRP here

Udp server-client programming borrowed from here