Vikram Singh
Person Number : 50247207

Programming Assignment 2

3rd November 2017

I have read and understood the course academic integrity policy.

ABT Timer Working and Time Out

The Abt timer working is pretty straight forward. The steps implemented are as follows:

1. Whenever the application layer sends a message, ABT first checks if it any message is already in the process of being sent. If another message is being sent then the new message is put in a queue, else, the message is sent and the timer for the message is started.
2. The receiver receives the message and if it receives the expected sequence number, it will send the corresponding acknowledgement for the message.
3. If the sender receives the expected acknowledgement it stops the timer and sets the canSend flag as zero which means that the first message in the queue can now get send.
4. If the sender does not receive the correct acknowledgement on time then its timer times out and the message is sent again.

Time Out

The Time Out value chosen by me is 9 units which gives me the best throughput.

If I choose a higher value of time out say 15 or 20, the throughput decreases, since in ABT the packets are resent on time out and hence when the loss or corruption or both is more, say around 0.4 or higher and the packets are not correctly acknowledged, ABT waits till the time out period before sending the packets again, which decreases the throughput.

On choosing a lower value of time out, say around 4 units, the throughput again decreases as the ABT doesn't get sufficient time to send the packet or receive the acknowledgement and hence on time out, it keeps sending the message to the receiver and the corresponding acknowledement may arrive late, ABT times out before the arrival and sends.

GBN Time Out

The time out value chosen by me for my GBN implementation is 40.

Since GBN sends a cumulative ack, my implementation of GBN performed much better at a higher time out value, At higher time out value if the window size is more, gbn is able to achieve higher through put due to the receiver sending an cumulative ack. If however the loss or corruption or both are higher the GBN does not perform well as it still resends all the packets every time. In this scenario ABT works better then GBN.

For lower time out values, if the corruption is more the GBN implementation starts working very slow. Mostly because in the rare case that a few packets are being successfully transferred, the low time out value stops this rare scenario for a better throughput.

In case of a higher time out value then the one chosen say around 80, the performance again starts declining, for both the cases when the corruption or loss or both are high or low. Because the GBN now starts waiting for a longer period of time before sending the messages. This is though window size dependent. For lower window sizes, this implementation cause a lot of delay but for higher window size it improves the performance. Again if you select an even higher value of the time out it starts decreasing the performance for the same higher window size. There should be an ideal balance. Both having their own tradeoffs.

SR TIME OUT

The time out value chosen by me for SR is 25.

In SR if we select a high value of time out then it does not retransmit the packets that effectively and the throughput decreases.

If the value is too less then we keep on getting timeouts, which decreases the throughtput.

SR TIMER

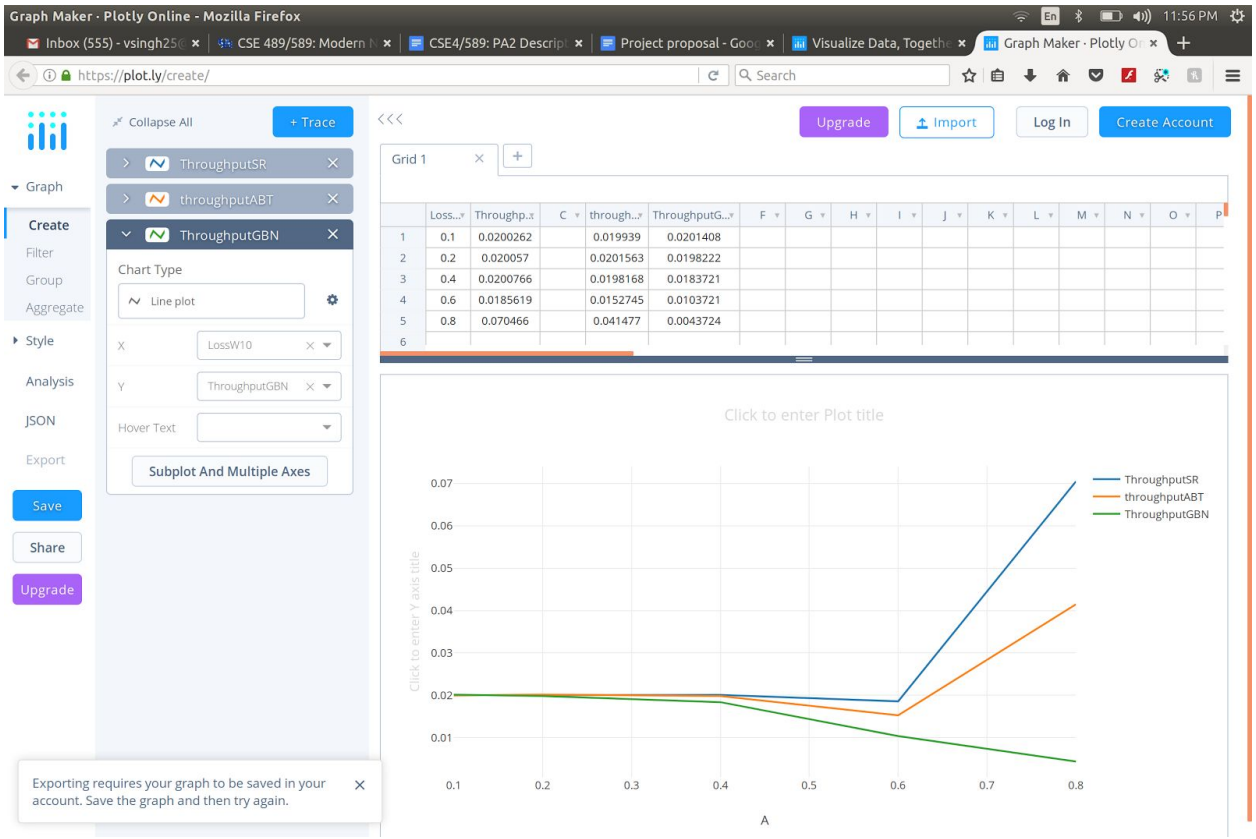
In SR Timer, the **currTimerSeq** field in my program maintains the sequence number of the packet for which the timer has been started.

I have used an array of struct called **packetsW** which stores the time at which the packet was sent(**startTime**) and the time at which the packet time outs(**endTime**).

Now if the timer expires. I resend the packet(**currTimerSeq**) again and update its **startTime** and **endTime.** Then I calculate the packet with the closest end time in my struct array **packetsW** and I start the timer for the difference between the current time and its time out's expiry time(**endTime**).

Hence, the single timer can be thought of as replicating the behaviour of a multiple timer.

If we get the ack only for the packet whose timer is currently running, then we stop the timer and calculate the packet with the closest end time and start the timer for the difference between the now time and the packet's time out time.



With 10 experiments