**REPORT ON GBN and SELECTIVE REPEAT**

**GO Back N:**

**Introduction:**

1. In order to increase the efficiency from stop and wait approach we use sliding protocols like Go Back N
2. We will transmit data from sender to receiver not one at a time, but multiple packets are sent at once.
3. Total number of packets which needs to be transmitted is based on WINDOW (which is the size of window).
4. If we have a WINDOW= 4, then it means, we can transmit 4 packets at once.
5. When receiver receives the packets from sender, first we need to check whether it is a cumulative packet or not.
6. Cumulative packet is nothing but, two consecutive packets i.e., packet 0 and packet 1, packet 4 and packet 5 etc.,
7. If we miss the sequencing, then discard the packet.
8. Missing the sequence means, if we get packet 6 before packet 5, then we will discard it and ask to resend the packet 5 by sending the acknowledgment of packet 4 again.
9. Once we get the right packet in order, then we can send the acknowledgment for that sequence number from Receiver to Sender.
10. Once we get the acknowledgment for the packet from Receiver, we move on to the next window and so forth till we reach the End of the file.

**TIMEOUT**:

1. We have a timer set for each window, in between this timer only the sender should get the acknowledgment from the receiver.
2. If we don’t get the acknowledgment from the receiver, and timer time-out, then we will resend the packet from sender from the starting of WINDOW.
3. If we get the acknowledgment from the old timer, initially the WINDOW moves next, and later when new acknowledgment again comes (the one which is sent from sender and acknowledged by receiver) which is nothing but a duplicate acknowledgment, we detect as a duplicate ack and can ignore it.

**Scenarios:** We can have different scenarios on different input conditions

* Window size
* Propagation delay and transmission delay
* Time out Settings.
* Loss Rate

All the scenarios are been tested with a simulation implemented in JAVA.

**WINDOW SIZE:**

Let us assume the WINDOW SIZE = 2, which means we can transmit only 2 packets at once from sender to receiver.

1. Let us consider delay between two ends as 9500 ms and Timer be 4500 ms, so that we can determine how window size affects.
2. When Window Size is less, then it takes more time for transmission.
3. Since, we have less window size, since we are doing cumulative data packets, it takes more number of duplicate acks from receiver to sender.

Let us assume the WINDOW SIZE = 4, which means we can transmit 4 packets at once from sender to receiver.

1. Just like before let us take both ends have a delay of 9500 ms and Time is 4500 ms.
2. Since, we have a bigger window size, we can transmit more number of packets at once.
3. As, it is obvious that when we can send multiple data at once, if transmission happens successfully, i.e., without any loss in packet then obviously the bigger window size will be faster and efficient, but when there is a packet loss, then it might take more time because, it should form a sequence again in the given timeout settings. If we can’t get it done, then retransmission of the packet again starts from first, resulting in lot of duplicate acks when compared to lower Window Size.

So depending upon packet loss rate the window size is affected.

* If window size is less, and packet loss is also more then it might take fewer number of duplicate acks.
* If window size is more and packet loss is also more then it will take more number of duplicate acks than the previous one.

**Delays:**

* Lets us consider two types of delays.
* Transmission Delay and Propagation Delay.

Transmission Delay: Time it takes to push the bits of data onto the link, and is given by L / R, where L is the amount of data and R is the rate.

Propagation Delay: Time it takes for a bit, once on the link, to reach the destination.

* Let us assume the total delay in our simulation as Transmission Delay and Propagation Delay.
* Let us have a fixed Window size to compare our relation.

1. If there are more delay than the timeout settings, then it obviously means that always there would be a duplicate acked packet for each window.
2. If there is a huge difference between delay and timeout setting, such that delay is more then it is obvious that there would be more number of duplicate acks getting generated from receiver towards sender.
3. If the delay is less than the timeout settings, then there is a fair amount of chance that there will be only few duplicate acks.

**Timeout:**

* Timeout settings is nothing but a timer, in between the the packets from the window should reach from sender to receiver and needs to get acknowledged.
* If the timer timeouts then, we have to retransmit again from the first id of window.

1. So if there is more Timeout, then it results in less number of duplicate acked packets.
2. If there is small Timeout, which keeps on expiring, then we might come across a lot of duplicate acks.

**Loss Rate:**

* A loss might occur due to many reasons such as pre-matured time out. In our simulation program, we implicitly make the program sometime to be loss the packet.
* Loss Rate is one of the most critical parameter which makes the entire program less efficient when packet loss occurs.
* When a packet loss occurs, from sender to receiver, Receiver doesn’t take the packet, because we are losing the consecutive bit ordering. When the same packet loss occurs, from receiver to sender during sending acknowledgments, then also sender will re transmit the data again from the starting of window, resulting in more duplicate acks.

1. When there is minimum or low loss rate then even though packet loss occurs and duplicate packets and duplicate acks are transmitted, it is still able to manage the transmission successfully at a long time.
2. When there is a huge amount of loss rate, then duplicate packets and duplicate acks are in a huge number that it can make the socket interface to crash, because it cannot handle data of duplicate acks and duplicate packets in exponential form.

**Observations Made:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Window Size | Transmission delay | Propagation delay | Timer | Loss Rate | Total No of Packets | Total Time Taken |
| 2 | 100 | 200 | 700 | 10 | 9 | 6288 |
| 2 | 100 | 200 | 700 | 20 | 9 | 10211 |
| 4 | 100 | 200 | 700 | 10 | 9 | 7712 |
| 4 | 100 | 200 | 1000 | 20 | 9 | 8294 |
| 2 | 1000 | 2000 | 3000 | 10 | 9 | 51134 |
| 4 | 1000 | 2000 | 3000 | 20 | 9 | 36104 |
| 2 | 100 | 200 | 1000 | 10 | 9 | 9414 |
| 2 | 100 | 200 | 2000 | 10 | 9 | 1310 |

Test-Cases:

1. For different input values of window size, with same values of Transmission delay, propagation delay, Time out Settings, Packet Loss rate and total packets sent, we can observe how window size affects the data transmission rate.
2. Similarly perform the same for other input parameters, and make others constant and note down the values.

**SELECTIVE REPEAT**

Introduction:

1. Just like Go Back N algorithm, selective repeat algorithm also works in the same way by sending multiple packets at once.
2. Instead of cumulative acknowledgment of the packages, we send individual data packets based on window, and receive acknowledgment from receiver.
3. If the first receive the data packets at Receiver, we start sending the acknowledgment bits. However, the packet slides, as soon as the Receiver receives the data at Receiver side.
4. If we get the acknowledgment from Receiver at sender side, we accept the acknowledgment and start moving the window to next packets.
5. However, the ordering of data packets is not required. So, we store the data if ordered is missed in a temporary buffer, and wait for the missing sequence packet to reach, so that ordering can be performed.
6. If the acknowledgment is not received from the receiver, sender retransmits the same packets until it gets acknowledgment.

**TIMEOUT Settings:**

1. Instead of having a timer setting for the whole window, we have individual timer setting for each packet.
2. When timeout occurs, retransmission of packet happens at Sender side.
3. If there is a loss occurring in between the timeout, then the Sender window doesn’t slide forward until we get the acknowledgment from the Receiver.
4. Sender keeps on resending the data until the right acknowledgment is received from the Receiver side.

**Scenarios:** Just like how we have different scenarios of Go back N, let us consider the same scenarios

* Window size
* Propagation delay and transmission delay
* Time out Settings.
* Loss Rate

**Window Size:**

1. With the Increase in window size, it might look like more number of packets are flowed from sender to receiver and vice versa.
2. But one of the issues of Selective repeat is when the window size is too large, and there was acknowledgment lost from Receiver, Sender tries to retransmit the data from 1st Window, but, since receiver has already slider for the first time, when second time packets reach, then it mistakes 1st windows retransmitted bits to the new packets from next slide window transmitted bits.
3. The above case usually happens, when there is loss in acknowledgment and when window size is more.
4. Let us consider delay between two ends as 9500 ms and Timer be 4500 ms, with window size = 2, so that we can determine how window size affects.
5. When Window Size is less, then it takes more time for transmission.
6. Since, we have less window size, It takes more time for sending the packets.
7. When we have same delay and window size = 4, then transmission of data is little bit quicker because, we transmit more amount of data.

**Delays:**

1. When the delay between two ends is less, then the packets flow quicker between both the ends, and there won’t be need of retransmitting duplicate packets from sender to receiver.
2. If we have more amount of delay, then each time sender sends duplicate packets to receiver and receiver sends duplicate acknowledgments to Sender.

It usually works just like how Go Back N works in case of delays.

**Timeout:**

1. This is the major difference where Go Back N and Selective repeat differs.
2. Instead of retransmitting entire window like it was done in Go Back N, in Selective repeat, only packets which require the receiver to individually acknowledge are being transmitted again from sender to receiver.
3. Since, we have individual timer to each data packet in a window, we can retransmit individual packets from a window.
4. Having a bigger timer values means, less number of duplicate acks.
5. If time is less, then there is a probability of having more number of duplicate acks and retransmissions.

**Loss Rate:**

1. Even loss rate plays the most important role in selective repeat.
2. When the packet loss happens, then we need to retransmit the data which has been lost. We need not send the entire window, instead we can only send the data which is been lost.
3. Similarly, when acknowledgment is lost also, sender resends the data which didn’t get acknowledgement.
4. So if the loss rate is high, selective repeat is very inefficient because, we have a chance of acknowledging false window packets like discussed in WINDOW scenario.

**Observations Made:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Window Size | Transmission delay | Propagation delay | Timer | Loss Rate | Total No of Packets | Total Time Taken |
| 2 | 100 | 200 | 700 | 10 | 9 | 7307 |
| 2 | 100 | 200 | 700 | 20 | 9 | 7627 |
| 4 | 100 | 200 | 700 | 10 | 9 | 8922 |
| 4 | 100 | 200 | 1000 | 20 | 9 | 6129 |
| 2 | 1000 | 2000 | 3000 | 10 | 9 | 60410 |
| 4 | 1000 | 2000 | 3000 | 20 | 9 | 36114 |
| 2 | 100 | 200 | 1000 | 10 | 9 | 9743 |

**Test-Cases:**

1. For different input values of window size, with same values of Transmission delay, propagation delay, Time out Settings, Packet Loss rate and total packets sent, we can observe how window size affects the data transmission rate.
2. Similarly perform the same for other input parameters, and make others constant and note down the values.

**Comparison and Conclusion:**

* Go back N doesn’t function well when the window size is too large, because the number of packets grow exponentially when one of packets in widow is wrongly transmitted. So when packet loss is more, then it is not suggestible to use go back n
* When we have noisy link Go Back N is inefficient and selective repeat performs better in noisy links because, we need not send entire window.
* Go back N is less complicated to implement than SR.
* When packet size is small, Go Back N functions better.

We cannot conclude which is better among the both because, each has its own advantages and disadvantages