Report - CS3205 Assignment 4

Aniswar Srivatsa Krishnan
Computer Science and Engineering
Indian Institute of Technology Madras
CS18B050

***Abstract*—The objective of this assignment is to implement the Selective Repeat and Go-back-N reliable transmission protocol and measure the round-trip delays..**

***Index Terms*—Selective Repeat, Go-back-N, link layer, UDP**

I. Introduction

A. Selective Repeat ARQ

Selective Repeat ARQ is also known as the Selective Repeat Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. The Go-back-N ARQ protocol works well if it has fewer errors. But if there is a lot of error in the frame, lots of bandwidth loss in sending the frames again. So, we use the Selective Repeat ARQ protocol. In this protocol, the size of the sender window is always equal to the size of the receiver window. The size of the sliding window is always greater than 1.

If the receiver receives a corrupt frame, it does not directly discard it. It sends a negative acknowledgment to the sender. The sender sends that frame again as soon as on the receiving negative acknowledgment. There is no waiting for any time-out to send that frame. In this assignment we do not make use of the negative acknowledgement.

B. Go-Back-N ARQ

Go-Back-N ARQ protocol is also known as Go-Back-N Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.

The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again.

II. Experimental Details

The project requires two separate programs, running at the same time, on two different hosts: a sender program that generates and transmits packets; and a receiver, that accepts the packets, and transmits the acknowledgments to the sender. Note that the receiver does not send any data packet; it only sends acknowledgments. Communication between the sender and receiver is through UDP sockets. The receiver is set up as a UDP server, and the sender is set up as a UDP client.

A. *Selective Repeat:*

1) Sender: The main loop of the sender has these main steps:

- Generate a packet of length, where the packet length follows a uniform distribution: Uniform(40, MAX_PACKET_LENGTH) bytes, where MAX_PACKET_LENGTH is command-line parameter). The first byte(s) of the packet contains the sequence number (depends on the number of bits in the sequence number field).Packets are generated at periodic time intervals specified by the PACKET_GEN_RATE parameter (packets/unit time). The transmit buffer has a capacity specified by the BUFFER_SIZE parameter (number of packets,not bytes). A newly generated packet will be dropped if the Buffer is full. A sequence number is assigned ONLY if the packet is added to the buffer.
- Transmit the packet based on the Window conditions. Start the timeout timer for this packet's sequence number. The timeout is set to 300 ms for the first 10 packets and then $2 * RTT_{ave}$ (in milliseconds) for all other packets.
- Process the next packet (when available) and transmit it if the sender window is not exhausted, i.e. the total number of unacknowledged packets is at most WINDOW_SIZE. Given an n-bit sequence number, the maximum window size will be $2^n - 1$ for Selective Repeat.
- If an ACK packet arrives, process it, update local state variables and cancel timers corresponding to acknowledged packets. Remove the packet from the Transmit Buffer. Note that selective ACKs are used. For each sequence number acknowledged, calculate the Round-trip-Time (RTT) for the packet and update the average RTT (RTT_{ave}) for the packets acknowledged so far.
- If a timer expires, re-transmit only the unacknowledged packet.

The sender terminates after MAX_PACKETS (a command-line parameter) have been successfully ACKNOWLEDGED (OR) if the maximum retransmission attempts for any sequence number exceeds 10.

2) Summary of Command Line Options:: The command line options provided to the sender are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -s string – Receiver Name or IP address.
- -p integer – Receiver's Port Number
- -n integer – Sequence Number Field Length (in bits)
- -L integer – MAX_PACKET_LENGTH, in bytes

- -R integer – PACKET_GEN_RATE, in packets per second
- -N integer – MAX_PACKETS
- -W integer – WINDOW_SIZE (Assume that SWS = RWS)
- -B integer – BUFFER_SIZE

3) Output: The sender will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of -d will turn on DEBUG mode. For both modes, on termination, the sender will print the following information to the screen:

- PACKET_GEN_RATE
- PACKET_LENGTH
- ReTransmission Ratio: Ratio of Total Number of Transmissions (including Retransmissions) to Number of Packets Acknowledged.
- Average RTT Value for ALL Acknowledged Packets

In DEBUG mode, the Sender will also print the following information for EACH packet when its ACK is received:

```
Seq #: Time Generated: xx:yy
RTT: zz Number of Attempts: aa
```

where time is in milliseconds:microseconds format.

4) Receiver: The receiver is always waiting to read a packet from the UDP socket it is listening to. Whenever a packet is delivered to the receiver:

- The receiver randomly decides that packet is corrupted and decides to drop the packet; note that you can use rand, rand48, etc. The probability of packet drop is specified as a command-line parameter, denoted by PACKET_ERROR_RATE. This step is used to simulate random network errors.
- If the packet is NOT corrupted (per step 1 above), the receiver reads the packet and extracts the sequence number. If the receiver buffer is FULL, then the received packets are discarded even if they were correctly received. Otherwise, it follows the Selective Repeat protocol for generating ACKs, and buffering out-of-order packets. The ACK packets are NOT dropped and are always assumed to be delivered to the sender.

The receiver terminates after acknowledging MAX_PACKETS (a command-line parameter).

5) Summary of Command Line Options:: The command line options provided to the receiver are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -p integer – Receiver's Port Number
- -N integer – MAX_PACKETS
- -n integer – Sequence Number Field Length (in bits)
- -W integer – WINDOW_SIZE (SWS = RWS)
- -B integer – BUFFER_SIZE
- -e double – PACKET_ERROR_RATE

6) Output: The receiver will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of -d will turn on DEBUG mode. In DEBUG mode, the Receiver will also print the following information for EACH packet when it is successfully received. Note that the receiver will print this information ONLY in Sequence Number order. For example, if Seq. No. 3 is received before Seq. No. 2, then the receiver will NOT print information for Seq. No. 3 until Seq. No. 2 is received.

```
Seq #: Time Received: xx:yy
```

where time is in milliseconds:microseconds format.

B. Go Back N protocol:

1) Sender:: The main loop of the sender has these main steps:

- Generate a packet of length PACKET_LENGTH (command-line parameter) bytes. packet contains the sequence number.The first byte(s) of the Packet generation rate is given by the command-line parameter PACKET_GEN_RATE, in packets per second. You might need to use a thread that generates packets periodically (based on the above rate) and stores them in a buffer used by the sender's protocol. The maximum size of this sender transmission buffer is given by the command-line parameter, MAX_BUFFER_SIZE (number of packets, not bytes). A newly generated packet will be dropped if the Buffer is full. A sequence number is assigned ONLY if the packet is added to the buffer.
- Transmit the packet based on the Window Size conditions. Start the timeout timer for this packet's sequence number. The timeout is set to 100 ms for the first 10 packets and then $2RTT_{ave}$ (in milliseconds) for all other packets.
- Process the next packet (when available) and transmit it if the sender window is not exhausted, i.e. the total number of unacknowledged packets is at most WINDOW_SIZE.
- If an ACK packet arrives, process it, update local state variables and cancel timers corresponding to acknowledged packets. Note that cumulative ACKs are assumed. For each packet received, calculate the Round-trip-Time (RTT) for the packet and update the average RTT (RTT_{ave}) for the packets acknowledged so far.
- If a timer expires, retransmit all packets from the first unacknowledged packet.

The sender terminates after MAX_PACKETS (a command-line parameter) have been successfully ACKNOWLEDGED (OR) if the maximum retransmission attempts for any sequence number exceeds 5.

2) Summary of Command Line Options:: The command line options provided to the sender are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -s string – Receiver Name or IP address.
- -p integer – Receiver's Port Number
- -l integer – PACKET_LENGTH, in bytes
- -r integer – PACKET_GEN_RATE, in packets per second
- -n integer – MAX_PACKETS
- -w integer – WINDOW_SIZE
- -b integer – MAX_BUFFER_SIZE

3) Output: The sender will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of -d will turn on DEBUG mode. For both modes, on termination, the sender will print the following information to the screen:

- PACKET_GEN_RATE
- PACKET_LENGTH
- Retransmission Ratio: Ratio of Total Number of Transmissions (including Retransmissions) to Number of Packets Acknowledged.
- Average RTT Value for ALL Acknowledged Packets

In DEBUG mode, the Sender will also print the following information for EACH packet when its ACK is received:

```
Seq #: Time Generated: xx:yy
RTT: zz Number of Attempts: aa
```

where time is in milliseconds:microseconds format.

4) Receiver: The receiver is always waiting to read a packet from the UDP socket it is listening to. Whenever a packet is delivered to the receiver:

- The receiver randomly decides that packet is corrupted and decides to drop the packet; note that you can use rand, rand48, etc. The probability of packet drop is specified as a command-line parameter, denoted RANDOM_DROP_PROB. This step is used to simulate random network errors.
- If the packet is NOT corrupted (per step 1 above), the receiver reads the packet and extracts the sequence number. If sequence number matches the NEXT EXPECTED sequence number, it transmits an ACK to the sender, and updates local state variables. Note that Cumulative ACKs are used by the receiver. The ACK packets are NOT dropped and are always assumed to be delivered to the sender. Thus, RANDOM_DROP_PROB value is not used by the sender.

5) Summary of Command Line Options:: The command line options provided to the receiver are listed below:

- -d – Turn ON Debug Mode (OFF if -d flag not present)
- -p integer – Receiver's Port Number
- -n integer – MAX_PACKETS
- -e float – Packet Error Rate (RANDOM_DROP_PROB)

6) Output: The receiver will operate in TWO modes: DEBUG and NODEBUG. The default operation is NODEBUG mode. A command-line flag of -d will turn on DEBUG mode. In DEBUG mode, the Receiver will also print the following information for EACH packet when it is successfully received:

```
Seq #: Time Received: xx:yy
Packet dropped: false
```

where time is in milliseconds:microseconds format.

C. Implementation Details - SenderSR

1) ***argument_handler()****:*

- This function takes care of processing the command line arguments and initializing the corresponding variables in the program.

2) ***get_sock_id()****:*

- This function takes care of creating the socket and binding it to the required port.
- This socket is used by the other routines for sending and receiving messages.

3) ***get_time_now()****:*

- This function returns the current time in microseconds. This is used by the other functions for calculating timers and other details.

4) ***main_sender()****:*

- This function is executed in a separate thread and takes care of performing the tasks of the main loop of the sender.
- These tasks are generating the packets, adding them to buffer, transmit the packets,check timeouts and retransmit packets.
- Various data structures like queue are used by this function.

5) ***receiver()****:*

- This function is executed in a separate thread and takes care of receiving acknowledgment packets from the receiver and updating the RTT and other variables based on the packet received.

6) ***main()****:*

- This function calls `argument_handler()`, `get_sock_id()` and creates 2 threads for executing, `main_sender()`, `receiver()`.

D. Implementation Details - ReceiverSR

1) ***argument_handler()****:*

- This function takes care of processing the command line arguments and initializing the corresponding variables in the program.

2) ***get_sock_id()****:*

- This function takes care of creating the socket and binding it to the required port.
- This socket is used by the other routines for sending and receiving messages.

3) ***get_time_now()****:*

- This function returns the current time in microseconds. This is used by the other functions for calculating timers and other details.

4) ***receiver()****:*

- This function receives packets from the sender and takes care of sending the acknowledgment packets and also dropping packets according to the error rate. It also buffers out of order packets.

5) ***main()****:*

- This function calls `argument_handler()`, `get_sock_id()` and calls `receiver()`.

The functions used in the senders and receivers of GBN are similar except they implement the GBN protocol instead of the SR protocol.

III. Results and Observation

The outputs are obtained for the following parameter combinations.

$$PACKET\_GEN\_RATE \in \{20, 300\}$$
$$PACKET\_LENGTH \in \{256, 1500\}$$
$$RANDOM\_DROP\_PROB \in \{10^{-3}, 10^{-5}, 10^{-7}\}$$

- We observe that the average Round-trip-Time (RTT) and the Retransmission Ratio is calculated by the programs and displayed at termination.
- We observe that the retransmission ratio is positively correlated with packet drop prob. This is because if packets are dropped more often then the need to retransmit arises more often and hence the ratio increases. From the plot, we can also observe that the slope of the graph is higher for Go-Back-N compared to Selective Repeat
- For given values of the parameters we observe that Go-Back-N has a higher transmission ratio and average RTT. This is because in Go-Back-N, when any particular packet leads to timeout the entire window is retransmitted. The difference in average RTT is even more pronounced because of the method of calculation of RTT of resent packets. The RTT for a packet which is sent multiple times is calculated as the time between acknowledgment and the time at which it was first sent. This difference will tend to be high due to various processing performed in between.
- The max packets increases the runtime of the program but does not affect the average RTT or the retransmission ratio by a significant amount. At very low packet drop prob like 10^{-5}, we can say that increasing max packets leads to some packets being dropped.
- Increasing window size increases performance of selective repeat, as increasing generation rate demands increased window sizes.

A. *Tables*

Average RTT values and Retransmission Ratio for SR

GEN RATE	PACKET LENGTH	DROP PROB	Average RTT (ms: μs)	Retransmission Ratio
20	256	10^{-3}	0:728	1.01
20	256	10^{-5}	0:185	1
20	256	10^{-7}	0:89	1
20	1500	10^{-3}	0:101	1.001
20	1500	10^{-5}	0:76	1
20	1500	10^{-7}	0:188	1
300	256	10^{-3}	0:68	1.001
300	256	10^{-5}	0:68	1
300	256	10^{-7}	0:65	1
300	1500	10^{-3}	0:61	1.001
300	1500	10^{-5}	0:79	1
300	1500	10^{-7}	0:58	1

Average RTT values and Retransmission Ratio for GBN

GEN RATE	PACKET LENGTH	DROP PROB	Average RTT (ms: μs)	Retransmission Ratio
20	256	10^{-3}	0:92	1.002
20	256	10^{-5}	0:74	1.002
20	256	10^{-7}	0:76	1.001
20	1500	10^{-3}	0:91	1.002
20	1500	10^{-5}	0:73	1.002
20	1500	10^{-7}	0:84	1.002
300	256	10^{-3}	0:67	1.004
300	256	10^{-5}	0:188	1
300	256	10^{-7}	0:70	1
300	1500	10^{-3}	0:155	1.002
300	1500	10^{-5}	0:140	1
300	1500	10^{-7}	0:184	1

B. *Plot*

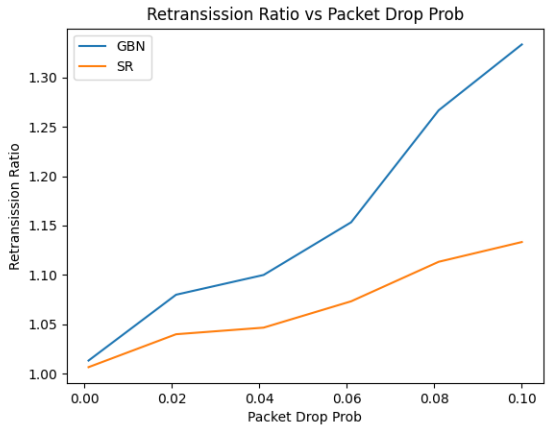

Fig. 1: Retransmission Ratio vs Packet Drop Prob for a particular configuration of Selective Repeat and Go-Back-N protocols

IV. Learnings

This assignment helped us understand the Selective Repeat and the Go-Back-N routing protocol in great detail. We understood the process by which reliable transmission is done by the various layers with the help of timers and retransmission. The same protocols are used at the transport layer as well by the TCP protocol and the similar principles will apply there as well. The assignment also familiarized me with multithreaded programs in C++. Writing multithreaded programs brings with it many challenges like synchronization, which we need to handle with the help of structures like mutexes.

V. Additional Thoughts

- We can try to include negative acknowledgments in the Selective Repeat Protocol, this leads to detecting loss of packets before timeout occurs.
- We can try to include cumulative acknowledgments in the Go-Back-N protocol which will lead to reduction in overall network traffic and hence help in congestion control

VI. Conclusion

Thus the simulation of the Selective Repeat and the Go-Back-N protocol have been completed successfully. It's performance under various network parameters have been observed. We observe that the results of the experiment correlate well with the desired goal of the protocol.

References

[1] Thread - C++ standard library reference documentation

[2] Kurose, J. F., Ross, K. W. (2016). Computer Networking: A Top-Down Approach. Boston, MA: Pearson. ISBN: 978-0-13-359414-0