

ITCS 6166: Computer Communication Networks – Implementation and Simulation of Go-Back-N and Selective Repeat Protocols

Project description:

Requirement to implement sliding window protocols – Go-Back-N (GBN) and Selective Repeat (SR) using an unreliable channel, UDP. Checksum of the segment to be sent need to be calculated and placed in the packet which is sent from sender to the receiver. The receiver needs to parse the checksum and check for bit errors.

For simulation purposes, the sender and receiver program must include some faulty network behaviors to test the implementation and observe the robustness of the protocol. The different cases are as follows:

- 1) Bit/Checksum error: It means alter the sending message just before sending the packet so that a checksum error will be detected at the receiver end. The response from the receiver need to be recorded here to observe how the receiver reacts to a faulty packet. Probability of occurrence of checksum error can be fixed at 0.1 (10 segments out of 100 sent can have this issue)
- 2) Lost Packet: In this scenario, receiver will treat a packet as lost, even though it received it perfectly. The response from the receiver need to be recorder here to observe how the receiver reacts to a lost packet. Probability of loss of packet can be fixed at 0.1 (10 segments out of 100 gets lost)
- 3) Lost Acknowledgements: Like lost packet, but this occurs at the sender. An acknowledgement to a packet received at the receiver's end is sent perfectly by the receiver, but it gets lost at the sender's end. Response from the sender for the lost packet need to be recorded here to observe how the sender reacts when it does not receive an acknowledgement for a packet under the stipulated time. Probability of the acknowledgment to get lost can be fixed at 0.05 (5 ACK's a=out of 100 gets lost)

Implementation Details:

- 1) Implemented using **Python 2.7**
- 2) For the receiver program, need to pass the port number, protocol, and window size (window size is needed only if the protocol is SR and is not required for GBN) as arguments. To run the receiver program, use the following command format:
python <source filename> <port number> <protocol> [<window size>]
For example:
Protocol = GBN --- *python server.py 7780 GBN*
Protocol = SR --- *python server.py 7780 SR 5*
- 3) For the sender program, need to pass a file which hold the configuration information like protocol to be used, window size, timeout value, maximum segment size (Sample file for

both SR and GBN is a part of the submission), port number, and the number of packets (The message to be sent is hardcoded in the code. Can increase the size of this message by using this parameter. Say, number of packets is 10 and the message to be sent is AB. The final message will be prepared before the sending starts, which will be "ABABABABABABABABABAB". This is done for illustration purposes.) To run the sender program, use the following command format:

python <source filename> <configuration filename> <port number> <number of packets>

For example:

Protocol GBN --- [python client.py GBN.txt 7780 10](#)

Protocol SR --- [python client.py SR.txt 7780 10](#)

4) Selective Repeat

a. Receiver side:

- i. Packet received from the sender's side. Unpack the packet to get sequence number, checksum, header and data
- ii. If the packet is header consists of all 1's, then it's the last packet sent by the sender. Exit the program
- iii. Calculate the probability for packet loss. If the probability is less than the threshold, then the packet is considered lost and proper message is printed for simulation.
- iv. If the packet received is already received once, then print appropriate message and send acknowledgment for the packet
- v. Calculate the checksum for the packet. If the checksum does not match, then the packet is corrupted. Print appropriate message.
- vi. If the checksum is correct, then mark the packet as received and send the acknowledgement. Print appropriate message.

b. Sender side:

- i. Sender prepares the message to be sent as per the input given by the user
- ii. Checksum is calculated for the message to be sent. A packet is created using sequence number, checksum, header, and the message.
- iii. Packet is appended into the send buffer and the timeout value is set for the packet
- iv. Bit error probability is calculated, and if the probability appears to be less than the threshold set, then an error is induced into the message just before it is sent.
- v. Meanwhile, a thread has already been started to look for acknowledgements from the receiver. Upon receiving an acknowledgment, it is unpacked and acknowledgement number is obtained.
- vi. Acknowledgement lost probability is calculated. If it is less than the specified threshold, then the acknowledgement is lost and appropriate message is printed.

- vii. If the acknowledgement is received, then appropriate message is printed and the send buffer and timeout value for the corresponding packet is set to reflect the receipt of the acknowledgement.
- viii. If all the packets are sent and all the acknowledgements are received, then a flag is set to indicate the same. The sender will send a final packet to indicate to the receiver that there are no more packets to send.

5) Go-back-N:

a. Receiver side:

- i. Packet received from the sender's side. Unpack the packet to get sequence number, checksum, header and data
- ii. If the packet header consists of all 1's, then it's the last packet sent by the sender. Exit the program
- iii. Calculate the probability for packet loss. If the probability is less than the threshold, then the packet is considered lost and proper message is printed for simulation.
- iv. Calculate the checksum for the packet. If the checksum does not match, then the packet is corrupted. Print appropriate message.
- v. If the expected packet is received, then send the acknowledgement for that packet send the last received flag to the sequence number of the current packet received.
- vi. If the packet received is not expected, then it is considered out of order and is discarded.

b. Sender side:

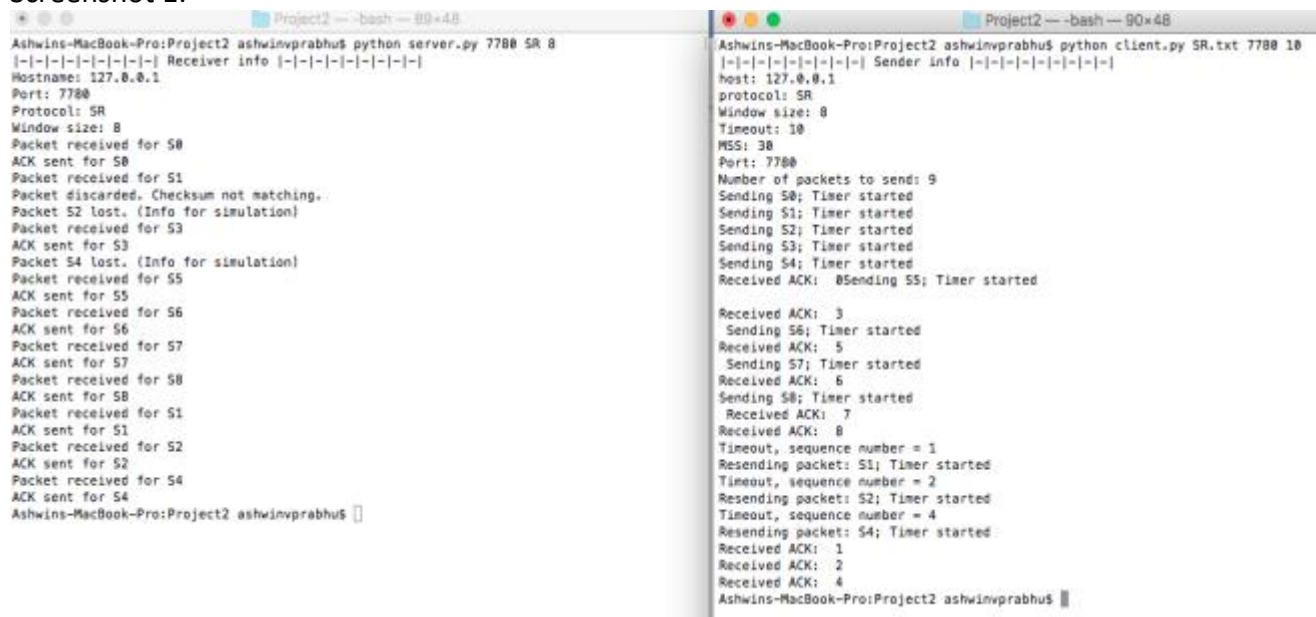
- i. Sender prepares the message to be sent as per the input given by the user
- ii. Checksum is calculated for the message to be sent. A packet is created using sequence number, checksum, header, and the message.
- iii. Packet is appended into the send buffer and the timeout value is set for the packet
- iv. Bit error probability is calculated, and if the probability appears to be less than the threshold set, then an error is induced into the message just before it is sent.
- v. Meanwhile, a thread has already been started to look for acknowledgements from the receiver. Upon receiving an acknowledgment, it is unpacked and acknowledgement number is obtained.
- vi. Acknowledgement lost probability is calculated. If it is less than the specified threshold, then the acknowledgement is lost and appropriate message is printed.
- vii. If the acknowledgement is received for the latest packet sent, then appropriate message is printed and the send buffer and timeout value for the corresponding packet and all the packet preceding it is set to reflect the receipt of the acknowledgement.

- viii. If the acknowledgement is received for some other packet, then appropriate message is printed and the send buffer and timeout value for the corresponding packet is set to reflect the receipt of the acknowledgement.
- ix. If all the packets are sent and all the acknowledgements are received, then a flag is set to indicate the same. The sender will send a final packet to indicate to the receiver that there are no more packets to send.

Output:

1) Selective repeat:

Screenshot 1:



```
Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python server.py 7788 SR 8
[...][...][...][...][...] Receiver info [...][...][...][...][...]
Hostname: 127.0.0.1
Port: 7788
Protocol: SR
Window size: 8
Packet received for 50
ACK sent for 50
Packet received for 51
Packet discarded. Checksum not matching.
Packet 52 lost. (Info for simulation)
Packet received for 53
ACK sent for 53
Packet 54 lost. (Info for simulation)
Packet received for 55
ACK sent for 55
Packet received for 56
ACK sent for 56
Packet received for 57
ACK sent for 57
Packet received for 58
ACK sent for 58
Packet received for 51
ACK sent for 51
Packet received for 52
ACK sent for 52
Packet received for 54
ACK sent for 54
Ashwins-MacBook-Pro:Project2 ashwinprabhu$

Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python client.py SR.txt 7788 18
[...][...][...][...][...] Sender info [...][...][...][...][...]
host: 127.0.0.1
protocol: SR
Window size: 8
Timeout: 10
MSS: 30
Port: 7788
Number of packets to send: 9
Sending 50; Timer started
Sending 51; Timer started
Sending 52; Timer started
Sending 53; Timer started
Sending 54; Timer started
Received ACK: 0 Sending 55; Timer started
Received ACK: 3
Sending 56; Timer started
Received ACK: 5
Sending 57; Timer started
Received ACK: 6
Sending 58; Timer started
Received ACK: 7
Received ACK: 8
Timeout, sequence number = 1
Resending packet: 51; Timer started
Timeout, sequence number = 2
Resending packet: 52; Timer started
Timeout, sequence number = 4
Resending packet: 54; Timer started
Received ACK: 1
Received ACK: 2
Received ACK: 4
Ashwins-MacBook-Pro:Project2 ashwinprabhu$
```

As seen in the above screenshot, the left window is the receiver and the right window is the sender. We can see that the window size is set to be 8, the maximum segment size is set to be 30, and the number of packets to be sent is calculated using the formula = length of the message to send / maximum segment size, which comes out to be 9 packets in this case.

Execution explained:

- Sender starts sending packets 0 to 8 in sequence
- At the receiver's end, packets, 0, 3, 5, 6, 7, and 8 are received correctly, and the acknowledgements are sent for these packets accordingly.
- A checksum error is detected for the packet 1, and hence packet 1 is discarded. No acknowledgement is sent in this case
- Packet 2 and 4 are lost. No acknowledgment is sent in this case.
- Sender receives acknowledgement for packets 0, 3, 5, 6, 7, 8.
- Timeout occurs for packets 1, 2 and 4, and these packets are resent.

Ashwin Venkatesh Prabhu
UNCC ID: 800960400
Email: avenka11@uncc.edu

- Receiver receives packets 1, 2, and 4, and sends the acknowledgements for these packets accordingly.
- Sender receives the ACK's for packets 1, 2, 4. Sender will send the final packet and the communication is stopped.

Screenshot 2:

```
Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python server.py 7788 SR 8
|-|-|-|-|-|-|-|-|-|-| Receiver info |-|-|-|-|-|-|-|-|-|-|
Hostname: 127.0.0.1
Port: 7788
Protocol: SR
Window size: 8
Packet received for S0
ACK sent for S0
Packet S1 lost. (Info for simulation)
Packet received for S2
ACK sent for S2
Packet received for S3
ACK sent for S3
Packet received for S4
ACK sent for S4
Packet received for S5
ACK sent for S5
Packet S6 lost. (Info for simulation)
Packet received for S7
ACK sent for S7
Packet received for S8
ACK sent for S8
Packet received for S1
ACK sent for S1
Packet received for S6
ACK sent for S6
Packet S7 lost. (Info for simulation)
Packet received for S7
ACK sent for S7
Ashwins-MacBook-Pro:Project2 ashwinprabhu$

Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python client.py SR.txt 7788 10
|-|-|-|-|-|-|-|-|-|-| Sender info |-|-|-|-|-|-|-|-|-|-|
Host: 127.0.0.1
Protocol: SR
Window size: 8
Timeout: 10
MSS: 30
Port: 7788
Number of packets to send: 9
Sending S0; Timer started
Sending S1; Timer started
Sending S2; Timer started
Sending S3; Timer started
Sending S4; Timer started
Sending S5; Timer started
Sending S6; Timer started
Sending S7; Timer started
Sending S8; Timer started
Received ACK: 0
Received ACK: 2
Received ACK: 3
Received ACK: 4
Received ACK: 5
Ack 7 lost (Info for simulation).
Received ACK: 8
Timeout, sequence number = 1
Resending packet: S1; Timer started
Timeout, sequence number = 6
Resending packet: S6; Timer started
Timeout, sequence number = 7
Resending packet: S7; Timer started
Received ACK: 1
Received ACK: 6
Timeout, sequence number = 7
Resending packet: S7; Timer started
Ack 7 lost (Info for simulation).
Timeout, sequence number = 7
Resending packet: S7; Timer started
Received ACK: 7
Ashwins-MacBook-Pro:Project2 ashwinprabhu$
```

- Sender sends packets 0 to 8 in sequence.
- Receiver receives packets 0, 2, 3, 4, 5, 7 and 8 correctly. Receiver sends acknowledgements for these packets. Even though the packets 2, 3, 4, 5, 7 and 8 are out of order, these packets are not discarded by the receiver, and is stored in a buffer, till the in order packets are received.
- Packets 1 and 6 are lost at the receiver's end.
- Sender receives the ACKs for packets 0, 2, 3, 4, 5, 8. ACK for packet 7 is lost.
- Timeout occurs for packets 1, 6 and 7, and these packets are resent by the sender to receiver.
- Receiver receives packets 1 and 6, and sends ACKs back for these packets.
- Packet 7 is lost at the receiver's end.
- Send receiver's ACKs for packets 1 and 6. Timeout occurs for packet 7 again, and packet 7 is resent.
- Receiver receives packet 7, and acknowledges it with an ACK.
- The ACK for packet 7 is lost again at the sender's end. The timeout occurs for packet 7 again, and the packet is resent.
- Receiver again receives packet 7 and acknowledges it.
- ACK for packet 7 is received by the sender. Sender will send the final packet and the communication is stopped.

Ashwin Venkatesh Prabhu
UNCC ID: 800960400
Email: avenka11@uncc.edu

2) Go-Back-N:
Screenshot 1:

```
Ashwini-MacBook-Pro:Project2 ashwinvprabhu$ python server.py 7780 GBN
|-|-|-|-|-|-|-|-| Receiver Info |-|-|-|-|-|-|-|-|
Hostname: 127.0.0.1
Port: 7780
Protocol: GBN
Packet received for S8
ACK sent for S8
Packet received for S1
Packet discarded. Checksum not matching.
Packet received for S2
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S3
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S4
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S5
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S6
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S7
(Packet out of order, discarded): last received packet in sequence: packet 0
Packet received for S8
Packet discarded. Checksum not matching.
Packet received for S1
ACK sent for S1
Packet received for S2
ACK sent for S2
Packet received for S3
ACK sent for S3
Packet received for S4
ACK sent for S4
Packet received for S5
ACK sent for S5
Packet received for S6
ACK sent for S6
Packet received for S7
ACK sent for S7
Packet received for S8
ACK sent for S8
Packet received for S9
ACK sent for S9
Packet received for S10
ACK sent for S10
Ashwini-MacBook-Pro:Project2 ashwinvprabhu$
```

As seen in the above screenshot, the left window is the receiver and the right window is the sender. We can see that the window size is set to be 8, the maximum segment size is set to be 30, and the number of packets to be sent is calculated using the formula = length of the message to send / maximum segment size, which comes out to be 9 packets in this case.

Execution explained:

- Packets 0 to 8 are sent in sequence by the sender.
- Receiver receives packet 0 and sends an ACK for the same.
- A checksum error is detected for packet 1 and hence discarded. No ACK is sent for packet 1.
- Packet 2 to 7 are received out of order, and hence discarded. Packet 8 is out of order with a faulty checksum and is discarded nevertheless.
- On the sender's side, timeout for packet 1 has occurred and the packets from 1 through 8 are resent.
- Receiver receives packets from 1 through 8, and all are received correctly. Receiver sends ACKs for packets from 1 through 8.
- Sender receives ACK for packets from 1 through 5. ACK for packet 6 is lost at the sender's side.

Ashwin Venkatesh Prabhu
UNCC ID: 800960400
Email: avenka11@uncc.edu

- Timeout occurs for packet 6 at the sender's side. Packets 6 through 8 are sent again.
- Receiver receives packets 6 through 8, and acknowledges them.
- Sender receives the ACKs for packet 6 through 8. Sender will send the final packet and the communication is stopped.

Screenshot 2: *Number of packets to be sent is reset to 5.*

```
Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python server.py 7788 GBN
|-|-|-|-|-| Receiver info |-|-|-|-|-|
Hostname: 127.0.0.1
Port: 7788
Protocol: GBN
Packet received for 50
ACK sent for 50
Packet received for 51
ACK sent for 51
Packet received for 52
ACK sent for 52
Packet received for 53
ACK sent for 53
Packet received for 54
ACK sent for 54
Packet received for 53
ACK sent for 53
Packet received for 54
ACK sent for 54
Ashwins-MacBook-Pro:Project2 ashwinprabhu$
```

```
Ashwins-MacBook-Pro:Project2 ashwinprabhu$ python client.py GBN.txt 7788 5
|-|-|-|-|-| Sender info |-|-|-|-|-|
host: 127.0.0.1
protocol: GBN
Window size: 8
Timeout: 10
MSS: 30
Port: 7788
Number of packets to send: 5
Sending 50; Timer started
Sending 51; Timer started
Sending 52; Timer started
Sending 53; Timer started
Sending 54; Timer started
Received ACK: 0
Received ACK: 1
Received ACK: 2
Ack 3 lost (Info for simulation).
Timeout, sequence number = 3
Resending packet: 53; Timer started
Resending packet: 54; Timer started
Received ACK: 3
Received ACK: 4
Ashwins-MacBook-Pro:Project2 ashwinprabhu$
```

Execution explained:

- Packets 0 to 4 is sent by the sender's side.
- Receiver receives all packets in order and ACKs is sent for all the packets.
- Sender receives ACK for packets 0 to 2. ACK for packet 3 is lost.
- Timeout occurs for packet 3. Packets 3 and 4 are resent.
- Receiver receives packets 3 and 4, responds with ACK.
- Sender receives ACKs for packets 3 and 4. Sender will send the final packet and the communication is stopped.