

# MODERN NETWORKING CONCEPTS


CSE 589 spring 2014.

The document is report of the validation of the correctness of the Alternating bit, Go-Back-N and Selective repeat protocol. The protocols are tested with the given experiment input asked to check. The protocol trace is checked to verify the correctness.

## PA2

Validation of the protocols

Vijay Manoharan  
[manohara@buffalo.edu](mailto:manohara@buffalo.edu)  
#50097716

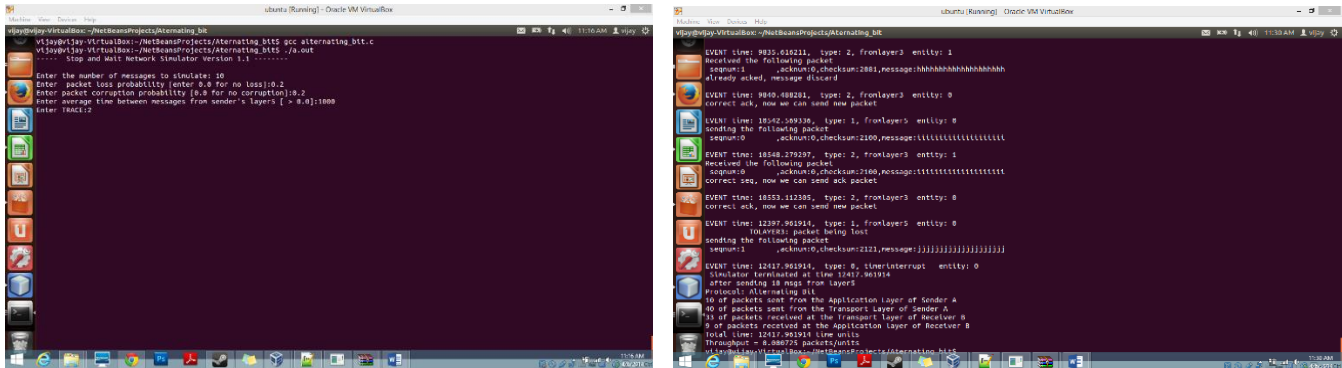


Validation of the protocol as per the given condition.

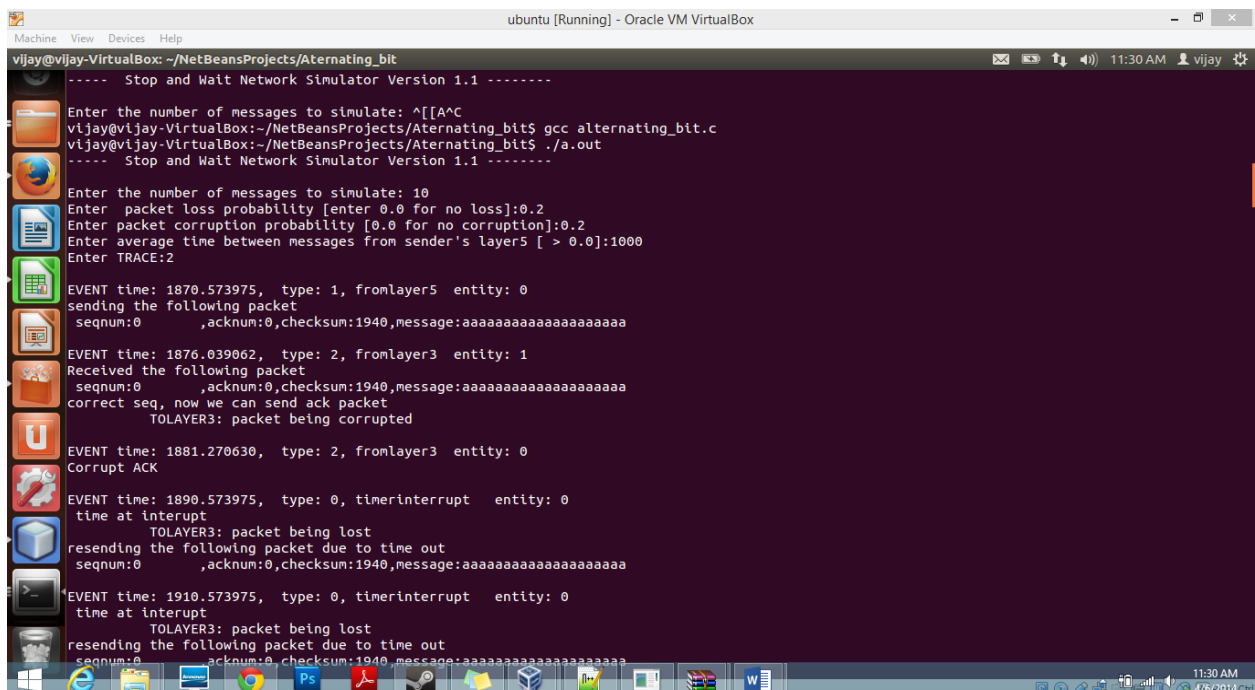
The following are the results of validation of the protocol.

## 1. Alternating bit protocol.

- Here is the screenshot for the input for validation and its output is beside.

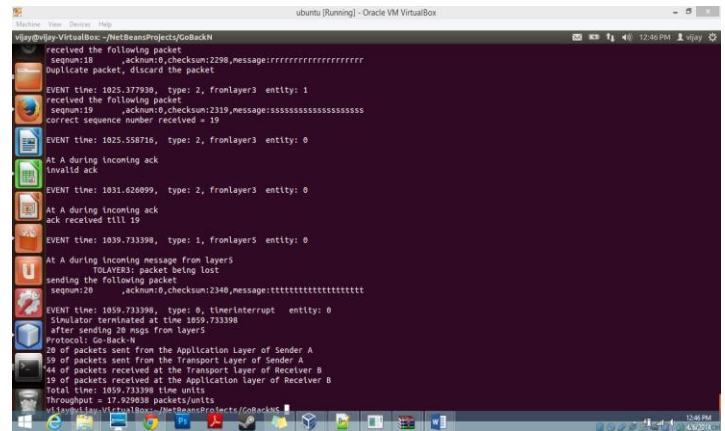
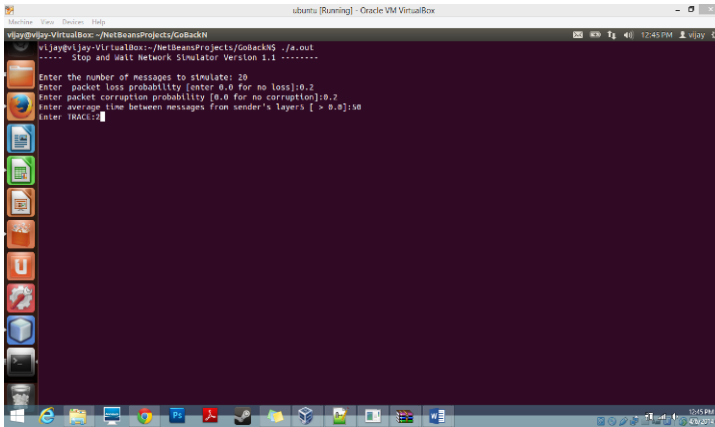


- For working of the protocol's correctness for the given input is show below. Here the first event is sent to a message "aaaaaaaaaaaaaaaaaaaaa" is sent to host B from host A with the sequence number 0 and also creates a timeout event. The timeout even is the time period within which Host A waits for the ack for the packet sent. If Host A does hear any ack from Host B within that period of timeout Host A sends the packet again. The host at B waits for the sequence 0. Since it has got the expected sequence it send ack to with ack # 0. While sending the ack, there is a packet loss. Since now Host A doesn't get the ack within the timeout period it resends the packet to Host B. In the detailed we can show that eventually Host B receives the packet from Host A and Host A gets the ack for the sent packet to Host B, which is irrespective of packet loss or corruption.

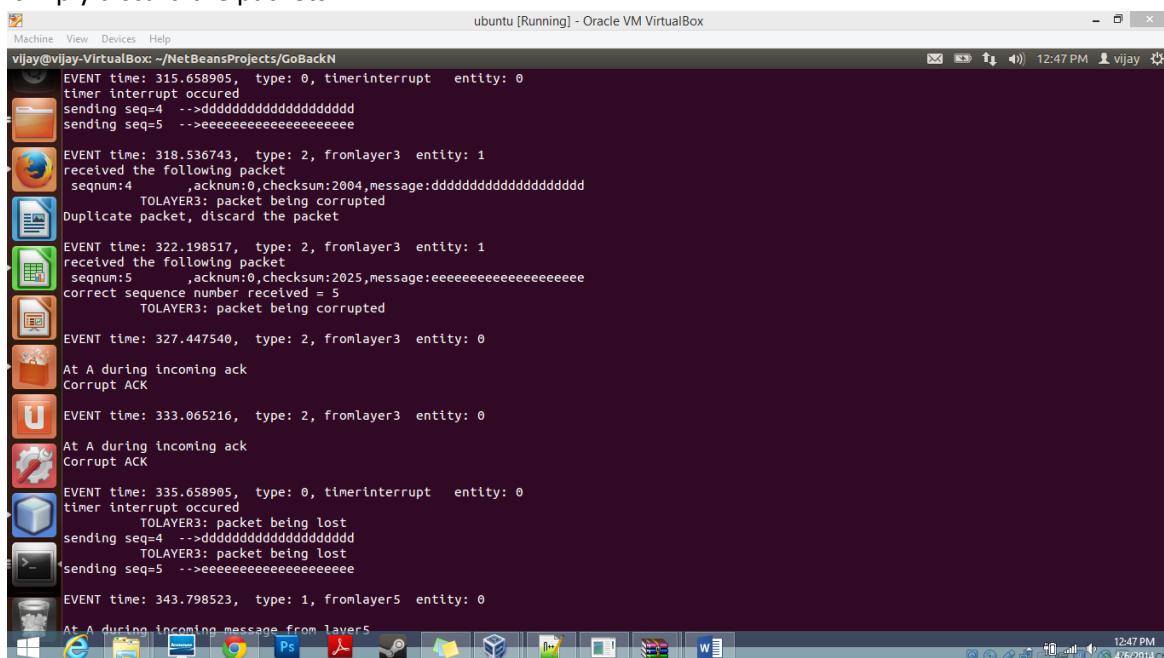


## 2. Go – BACK – N

- Here is the screenshot for the input for validation and its output is beside.

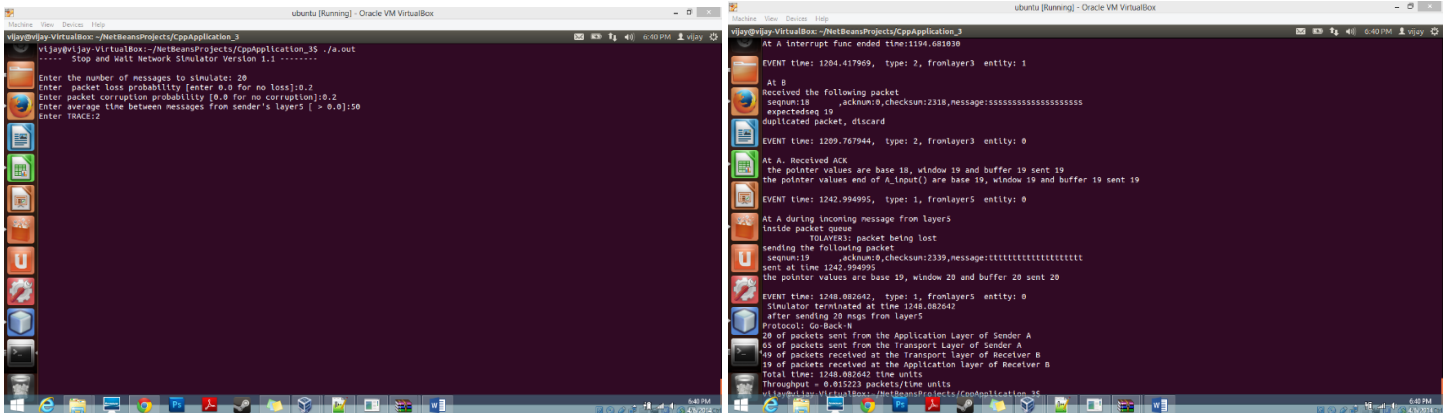


- For the correctness of the protocol we can see the trace snippet in the screenshot provided below. Here in the trace we can see that if there is packet loss or corruption of the packet, even then the packet is received at Host B and also the ack is received at Host A. Let us take a look at the trace we see that the packet sent are "dddddddddddddddddd" and "eeeeeeeeeeeeeeee", which is due to the timeout event that happened at Host A at (time =315~), the packet 'd' reaches Host B at 318~. The packet is corrupt and it is discarded. Now Host A does not know that the packet was received by Host B or not. So Host A waits for a timeout time (=20 time units), during which if it gets the ack the Host A sends next set of packets by updating its window. If not then Host A sends the packets for which the ack is not received. After the timeout is done, Host A sends the packets to Host B again. This happens until Host B receives the packet and host A gets the ack. Eventually all the packets are received at Host B and Host A gets those Ack. If the packets are out of order then Host B simply discards the packets.

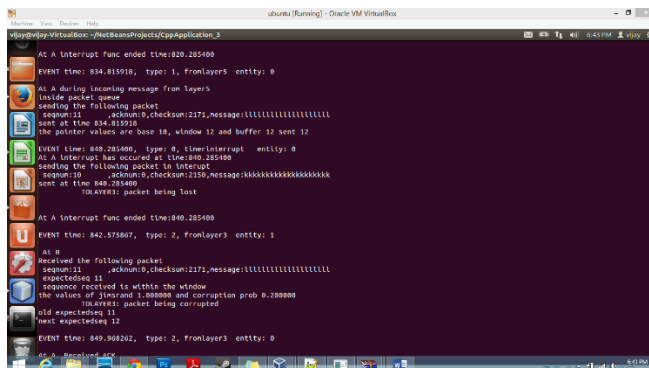


### 3. Selective Repeat protocol.

- Here is the screenshot for the input for validation and its output is beside.



- For the correctness of the implementation, we will use two screenshots to show how the correctness of the protocol can be explained. At the first case, the selective repeat keeps two window one at the sender and another one at the receiver. The Host A collects the packets and keeps in the window and sends them one by one. Host B receives the packet and checks if it is in the sequence, within the window frame or not. If the packet is in sequence it sends the ack to Host A, sends the packet to upper layer and moves the window to next expected sequence. At Host A the packet's ack is received in order then it also moves the buffer or else, it stores the packet as sent and keeps it. Next Host A sends only those packets within the window and not acknowledged. When Host B receives out of order packet, it also stores it and waits for correct sequence from Host A. Once the expected sequenced packet is received then the packets along with the stored but in order are pushed to the upper layer and window is moved. Every time the Host A sends, it waits for the packet to get the Ack from Host B until the timeout event happens. On timeout the Host A sends the packets which caused the timeout. Here we can see that initially Host A send the Message "IIIIIIIIIIIIIIIIIIII" and sequence # 11 at time 834~, waits till 854~. As there was no ack for # 11, it resends the message to Host B at 854 shown below. Once the host sends and it creates a new timestamp for the message with seq# 11. And waits till 874. But at 860~ Host A receives the Ack for Seq#11. So therefore the protocol makes sure that eventually every message reaches host B and it gets ACK for it.



Phase1 : message with Seq# 11 is received application layer of Host A. It creates a packet and sends it to Host B.

