Santosh Kumar Dubey
Person # 5009 7059

Reliable Transport Protocols

Table of Contents

Santosh Kumar Dubey
Person # 5009 7059

Reliable Transport Protocols

1. Implementation :

This section explains the implementation of the three protocols.
The main call flow have described in the assignment clearly. So here the explanation is of how the key functions are implemented for each protocol.

1.1 ABT Protocol :

1. Function A_output() :
a) Check the last packet already finished or not. If not finish yet, just drop current package.
b) Change sender state to not finish.
c) Construct the packet and send to layer3.
d) Start the timer, in case of packet lost or corrupted.

2. Function B_input() :
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum is not expected, just send back an ACK.
c) If the seqnum is expected, send data to layer5, then send back the ACK.

3. Function A_input() :
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum of ACK is not expected, just do nothing.
c) Stop the timer, set new next seqnum, and set flag to accept new packet.

4. Function A_timerinterrupt() :
a) If we do not receive ACK of last packet, retransmit it and start timer again.

1.2 GBN protocol

1. Function A_output() :
a) Add the message to an extra buffer at first.
b) If current window already full, do nothing, just return the function.
c) Retrieve a message from the extra buffer.
d) Get a free packet from the ring buffer.
e) Construct the packet and send to layer3.
f) Start the timer for the oldest not ACKed packet.

2. Function B_input():
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum is not expected, send back ACK with seqnum equal to the seqnum of latest received packet.
c) If the seqnum is expected, send data to layer5, then send back the ACK.

3. Function A_input() :
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum of ACK is not expected, just do nothing.
c) Adjust the window and reset the timer for next unACKed packet.
d) Check the extra buffer, if it not empty, call A_output to transmit the message again.

4. Function A_timerinterrupt() :
a) Retransmit all packets that do not ACKed yet.
b) Start the timer again.

1.3 SR protocol

1. Function A_init() :
a) Init the flag of all packets in buffer to free.
b) Start the timer.

2. Function A_output() :
a) Add the message to extra buffer.
b) If current window already full, just do nothing.
c) Get a message from extra buffer.
d) Get a free packet from the ring buffer.
e) Construct the packet and send to layer3.
f) Set timeout of current packet to g_cur_time plus TIMEOUT.

3. Function B_input() :
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum is not within the window, just send back ACK.
c) Copy data to ring buffer and set packet flag to valid.
d) If the seqnum is the oldest one in window, scan all the buffers, commit all sequence packets to layer5.
e) Adjust the window if needed.

4. Function A_input() :
a) Receive packet from layer3, if the packet have corrupted, just drop it.
b) If the seqnum of ACK is not expected, just do nothing.
c) Adjust the window if needed.
d) Check the extra buffer, if it is not empty, call A_output to send the message.

5. Function A_timerinterrupt() :

a) Add INTERVAL to current time.

b) Scan all packets in ring buffer, if any packet already time out, retransmit the packet, and adjust the timeout again.

2. Key Technologies :

2.1 Ring Buffer

In GBN and SR protocols we transmit multiple packets at the same time, so we need a buffer to store all the packets not finished yet.

Of course, there are many methods to meet this requirement, such as using a simple array or linked list and so on. But they will need frequent memory allocation or memory copy, either is a suitable choice for our requirement.

How ring buffer works?

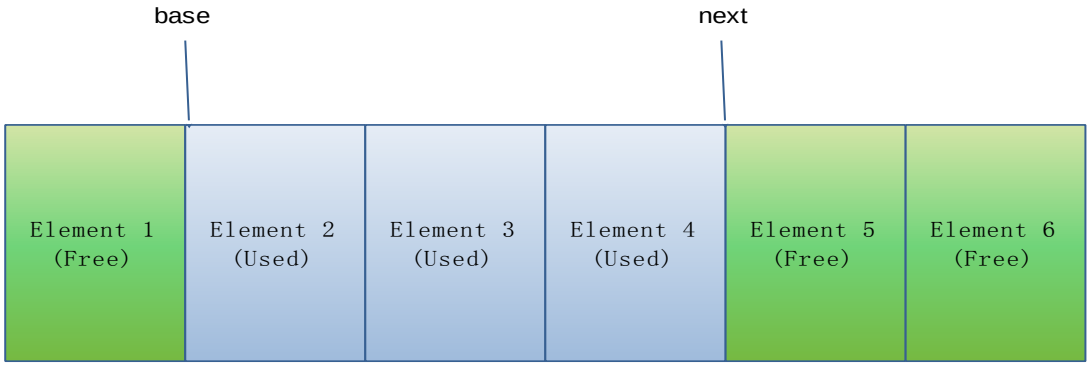
Step 1. When the program start, we define a start base and an end next variables for ring buffer, Also allocation N(size of window) elements at the very beginning.

Step 2. If new packets received or ACKed, we need to shrink the window, then we just need to adjust base to base+1.

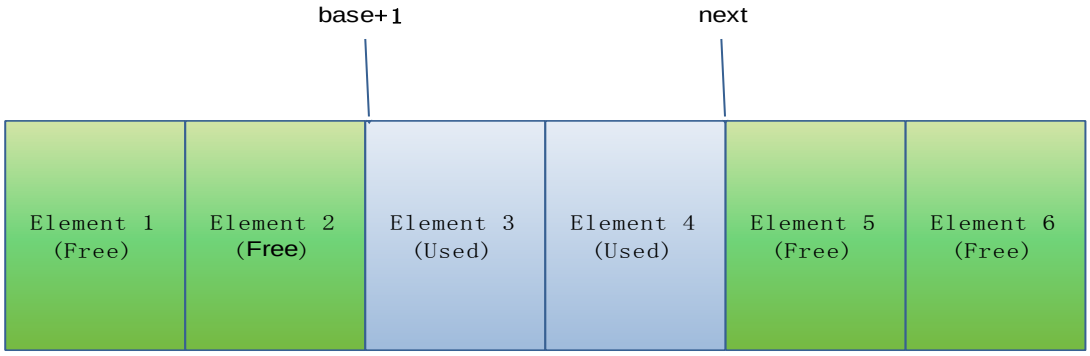
Let me take an example if current window for A is (10, 17), then if we received the ACK for seqnum 10, our window should move to (11, 17) now. So we just need to set base++. It is very efficient, because we do not need any extra memory allocation and free, or any memory movement.

Step 3. When we have more packets to send, we just get the free packet pointed by next. Then add next. If next go to the end of the buffer, we round it to zero again.

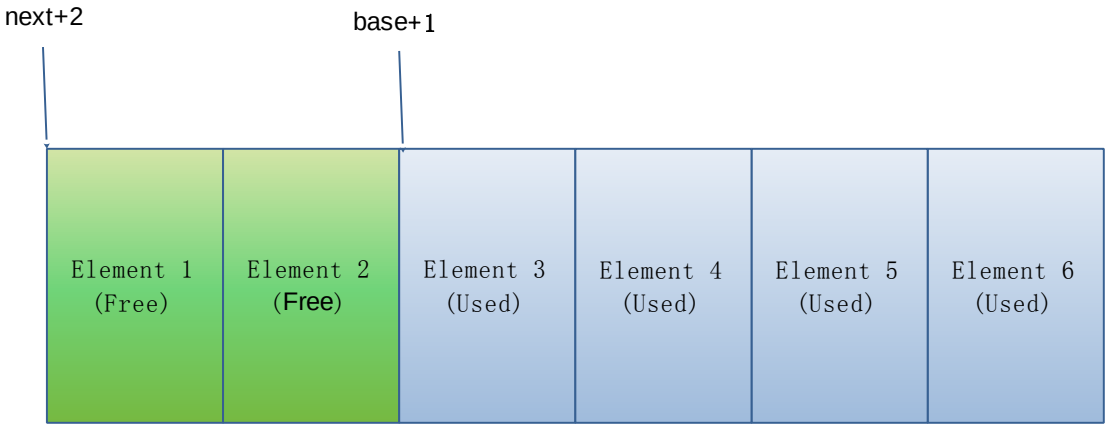
A) Current status of ring buffer:



B) Shrink the window by 1:



C) Send out two packets, now next will round back to zero point.



So ring buffer is very suitable for slipping window. It is very efficient, only need to adjust the start and end point while window changed.

2.2 Logic timer :

While implementing SR protocol, we need multiple timer for each packet that are not finished yet. But we have only one hardware timer. So we need to design a proper timer system to meet our requirement.

How Timers implemented?

Step 1. Define a global time start from zero, and added by TICK, after every time out occur.

Step 2. Define a very little time interval, we could consider it as one TICK. Then we let the hardware timer time out every TICK.

Step 3. While we send out a packet, we set the packet timeout to current time plus N*TICK.

Step 4. When the time out occur, we check all the packets is time out or not, by comparing current time to the time which have already been set in the packet.

Step 5. If there are any packets time out, we retransmit and reset the time out for the packet.

All in one, we let the timer to time out every short time, and remember every packet's time, then compare current time to the packet's time to determine the timeout.

Reliable Transport Protocols

2.3 Extra Buffer :

When a new message comes from layer5, we may still waiting for the last message to finish. At this time, we need to store this message to a buffer. Then when the window size decrease, we can send the message again.

How Extra Buffer implemented ?

We will use a single linked list to implement the extra buffer.

A) The structure for each element is:

```
struct node {
struct msg message; //store the message
struct node *next; //point to the next element
};
```

B) Variables define for linked list. We have a header pointed by list_head, and a tail pointed by list_end.

C) The process of add and delete element from linked list. Every time a new message comes in, we append the message at the tail of the linked list. While we need to send out the message, we pop the message from the head of the linked list.

D) For example if we pass 1000 message from the application and 200 of them has been dropped than in that case they will wait until the window becomes free if the situation is not happening then those packet will be dropped in extra buffer which will be clearly shown in the output message.

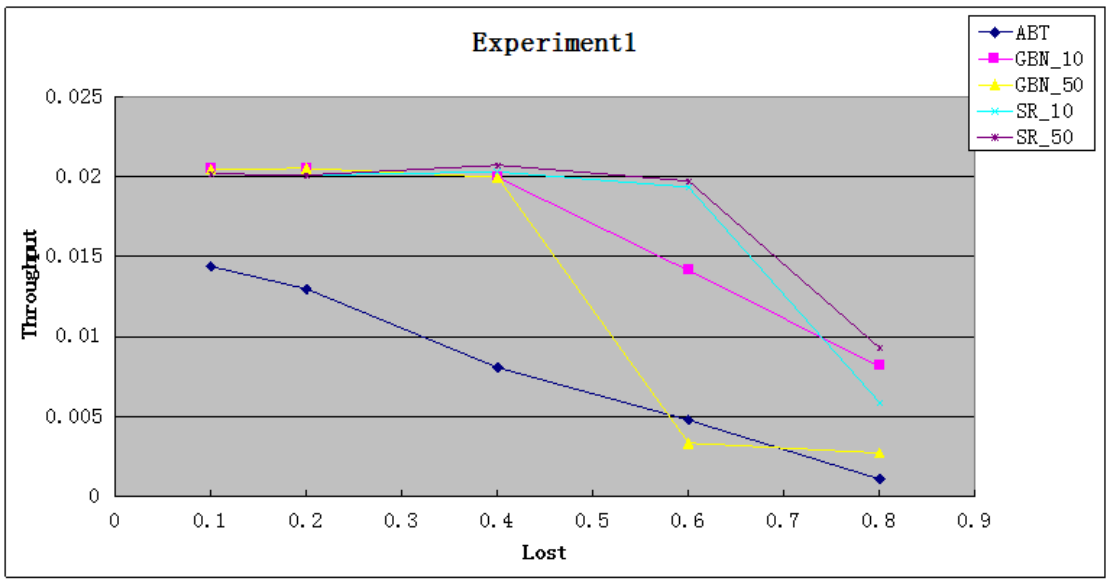
In this example 800 messages will be passed and 200 messages will be in extra buffer.

Reliable Transport Protocols

3. Observation and Analysis:

3.1 Experiment 1 : (Loss Probabilities : 0.1, 0.2, 0.4, 0.6, 0.8)

Observation:



A) GBN and SR protocols are almost faster than ABT protocol at any time;
B) Throughput get less and less while lost rate increase for the three protocols.

Analysis :

i) ABT protocol can only send another packet after the previous packet already ACKed.
Let me consider the average time for each packet transmit is 10ms. Then ABT protocol needs 10ms(send data) + 10ms(receive ACK)=20ms to complete one packet transmit.
ii) Another aspect, GBN and SR protocol, allows N(window size) packets be transmitting. If we need to transmit N packets, we can send them at one time, then wait for the ACKs for the N packets. Then the time cost for the N packets should be 10ms(send data) + 10ms(receive ACK) + N*R(last bit send time - first bit send time).
Generally R is very small, so within 20ms we could send N packets, but ABT can only send one packet.
iii) So GBN and SR protocol are faster than ABT protocol.
iv) As the lost rate increases, we have to resend a lot of packets, the throughput will be less.
v) Let me take ABT protocol for example, we need transmit two packets successfully to finish the transmit of one message. In any packets corruption or lost, we need wait for TIMEOUT(20ms in our program) time to retransmit the packet.
vi) When there is no packet corruption or packet lost, we will need 20ms to finish transmit of one message.

Reliable Transport Protocols

vii) Let me consider corruption rate N and packet lost M, then the possibility of success packet will be (1-N)*(1-M). Then the possibility of transmit one message successfully will be ((1-N)*(1-M))^2. In other words, we can only transmit ((1-N)(1-M))^2 packets every 20ms.

viii) As M(packet lost) increases, ((1-N)(1-M))^2 will be less, so our throughput will be less.

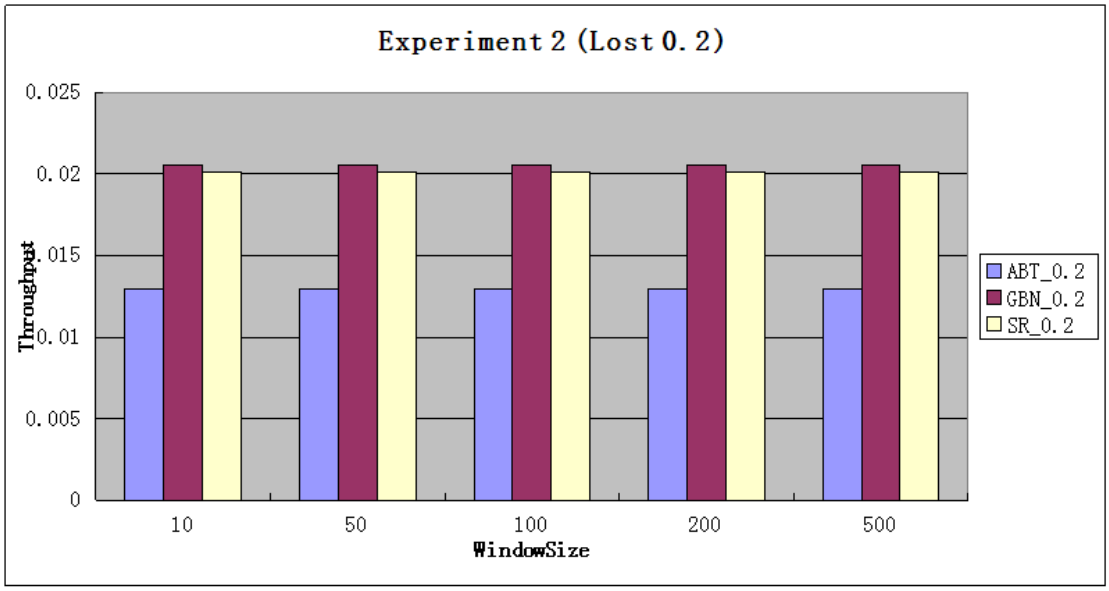
ix) The other two protocol have the same problem as ABT protocol, as a result throughput of them also will be less as packet lost increases.

Reliable Transport Protocols

3.2 Experiment 2 : (Be careful, the window size of ABT protocol are always 1)

Test:1 Experiment 2 (Lost 0.2)

Observation:



i) When network is good, GBN and SR protocol can get maximum throughput with a small window size.

ii) The throughput will not increase as window size be larger.

Analysis:

i) First of all, let me calculate the throughput while lost rate and corruption rate all are zero.

ii) As we know the simulator will generate one message every lambda(50 ms in our experiment) time in average. Also we already know each message transmit successful needs 10ms+10ms=20ms. So the maximum throughput will be 1/50ms=0.02 packets per time unit(ms).

iii) Secondly, From the chart, we can see when window size is 10, the throughput already equals the maximum throughput.

iv) Thirdly, let's prove window size 10 is enough. From experiment 1 we know every 20 ms, we can transmit ((1-N)(1-M))^2 packets successfully. That is ((1-0.2)(1-0.2))^2=0.64^2 =~ 0.4.

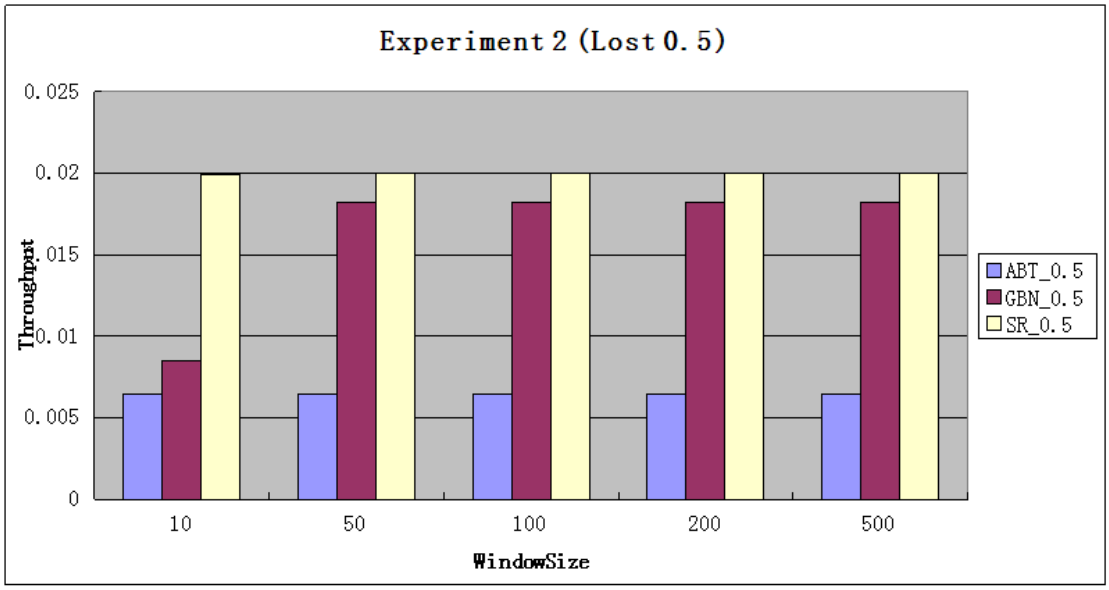
v) So within (10*20ms)=500ms, we can transmit (500ms/20ms)*0.4 = 10 packets. In other words, window size 10 is enough when lost rate is 0.2.

vi) As we already have enough window size, the throughput is limited by the message generate speed, so the throughput will not increase as window size be larger.

Reliable Transport Protocols

Test:2 Experiment 2 (Lost 0.5) :

Observation:



i) Throughput increases while window size increases.
ii) Total time required in all process differs slightly.

Analysis :

i) Using the same analysis of case lost rate 0.2, we can find when lost rate increase to 0.5, the window size will not enough **anymore**.

ii) From the implementation part, we know that, if the window is full, we will drop the message when a new message come in from layer5.

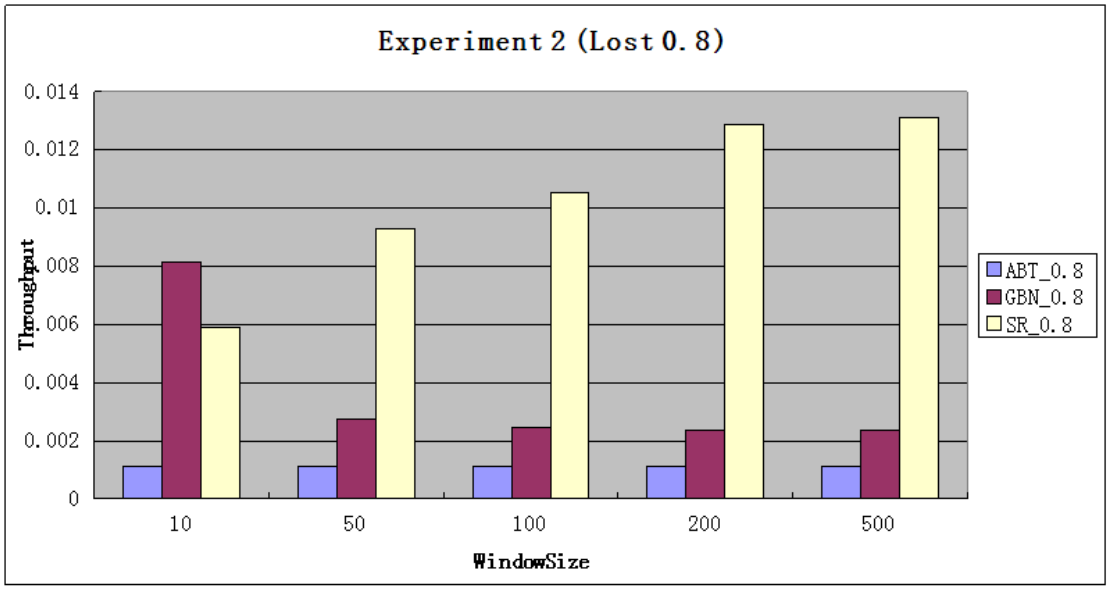
iii) So the throughput will be less than the maximum throughput. Then if we increase the window size to a proper size, the throughput will increase until the maximum throughput.

iv) We can conclude: We need to increase the window size to get maximum throughput, while network become bad and bad.

Reliable Transport Protocols

Test 3 : Experiment 2 (Lost 0.8) :

Observation:



A) When network situation is very bad, throughput of GBN protocol is bad, and it will decreases as window size increases.
B) When network situation is very bad, SR protocol is better than the other two protocols and it will increase as window size increases.
C) As the loss probability increases and window size also than GBN performance decreases comparatively and some packet drops also occurs.

Analysis :

i) First of all, let me calculate the possibility of one message send and receive successfully at the first time. That is ((1-0.8)*(1-0.2))^2=0.0256. It said we need retry 1/0.0256 =~ 40 times in average for one message.

ii) From the analysis of experiment 1, the time cost for sending N packets at the same time should be 10ms(send data) + 10ms(receive ACK) + N*R(last bit send time - first bit send time).

iii) When the window size N is small, we know N*R is a very short time comparing to 20ms. But as N becomes larger and larger, we cannot just ignore it any more.

iv) Especially as we know GBN protocol will retransmit all packets that do not ACKed yet. So we need to transmit 40*N times for one package, in normal case, we only need to transmit one time. That is a terrible thing, and extremely bad affect to the throughput. Especially when N increases, it will be worse.

v) So we can conclude: GBN protocol is not suitable when network is bad and the window size for GBN protocol should never be large with the large loss probability.

vi) As we know SR protocol will only retransmit the single timeout packet. It only need to retransmit 40 times in average for each packet. It is far more less than 40*N, so the throughput is larger than that

Reliable Transport Protocols

of GBN protocol. The throughput will be larger than that of ABT protocol, because of sending multiple packets at the same time.

vii) As the window size increase, more packets are buffered, there will be more opportunity to transmit more packets. So the throughput will increase as window size increases.

viii) Of course there are also some disadvantages for SR protocol, such as the algorithm will be more complex than the other two and we need N(window size) number of timers to check whether the packet is timeout or not.

ix) Large number of timers will affect the performance of system.

x) So we can conclude: SR protocol is suitable for different lost rate.

References :

1) http://unixhelp.ed.ac.uk/CGI/man-cgi?inet_ntoa
2) http://pubs.opengroup.org/onlinepubs/009695399/functions/inet_addr.html
3) http://webmuseum.mi.fh-offenburg.de/index.php?view=exh&src=73
4) http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/
5) http://www.networkinginfoblog.com/post/106/go-back-n-%28gbn%29/