Akshita Mittel
CS3041
10 November 2015

Reliable Data transfer.

Computer Networks: Assignment 5

1. SELECTIVE REPEAT

1.1. The selective repeat protocol was implemented using UDP sockets which were initialised using the A_init() function. The number of packets to be sent, the time out and trace were taken by the user through command line arguments.

1.2.Each Packet was sent independently on at a time into the main function A_out. Here each packet was made using struct pack. The struct consists of:

1.2.1.Sequence number

1.2.2.Acknowledgement number

1.2.3.Checksum

1.2.4.Data Length

1.3. The message was attached to the struct and sent to the network using the function toLayer3.

1.4.Alternating ack numbers 0 and 1 were used to verify if the packet received by the receiver was in the correct order. The A_sendrecv() function ensured that the packet was sent in the correct order by two mechanisms:

1.4.1.The packet will be sent again in the event of a packet loss.

1.4.2.The packet will be sent again if the correct ack number is not received.

1.5. Once the correct acknowledgment is received, we go on to send the next packet.

1.6. On the receiver side, the receiver keeps receiving packets endlessly.

1.7. The input taken contains the loss probability and corruption probability. These values are converted and the get_damage function gives indices of the packets which are to be lost. The indices are between 0 - 9 and hence the indices list has to be generated after every 10 packets received.

1.8.Once the packet is received and neither lost or corrupted, if the ack received is the correct one, the message is sent to layer5 and the expected ack number is reversed. Else the packet is ignored and the previous ack number is sent back.

2.GO-BACK-N

2.1. The Go-back-n protocol was implemented using UDP sockets which were initialised using the A_init() function. The number of packets to be sent, the time out and trace were taken by the user through command line arguments.

2.2. Each Packet was sent independently on at a time in the main function A_out till either the window was full or al the packets had been transmitted. Here each packet was made using struct pack. The struct consists of:

2.2.1.Sequence number

2.2.2.Acknowledgement number

2.2.3.Checksum

2.2.4.Data Length

2.3.The message was attached to the struct and sent to the network using the function toLayer3.

2.4.Incrementing ack numbers were used from 0 to n to verify if the packet received by the receiver was in the correct order. The A_input() function ensured that the packet was sent in the correct order by two mechanisms:

2.4.1.All the packets will be sent again from the last unpacked packet in the event of a packet loss.

2.4.2.The packets will be sent again if the correct ack number is not received.

2.5. Once the correct acknowledgment is received, we increment the window_left (which tells us how many slots in the window are left) and the sequence number.

2.6.On the receiver side, the receiver keeps receiving packets endlessly.

2.7.The input taken contains the loss probability and corruption probability. These values are converted and the get_damage function gives indices of the packets which are to be lost. The indices are between 0 - 9 and hence the indices list has to be generated after every 10 packets received.

2.8.Once the packet is received and neither lost or corrupted, if the ack received is the correct one, the message is sent to layer5 and the expected ack number is reversed. Else the packet is ignored and the previous ack number is sent back.

3. TRACE:

Either a value 1 or 0 is given in the trace. Giving a value 0 will only give the high-level overview of the simulation. A value of 1 will enable additional debugging information.