CSE 589 PA2

Rohin Kumar Reddy Gubbala
UB#50208182

I have read and understood the course academic integrity policy.

Alternating Bit protocol

The protocol uses a static time out is chosen based on the best throughput observed.
As a quick overview of the implementation, the program starts with a sequence number and alternates between 0 & 1. Whenever, a message arrives at A_output(), we build the packet (struct) with all the details of the packet (seqnum, acknum, chksum, payload) and send it to the receiver (via L5). Also, a timer is started here.
If there is a packet already in transit, then, the message is buffered.

At B_input(), we then check for in-order delivery as well as packet correctness(checksum).Then an acknowledgement is built for the correct reception of a packet. This ack is sent to the transmiter and the expected seq-num is updated.

At A_input(), the received ack is checked for inorder delivery as well as correctness similar to the receiver. Upon, correct reception of the ack, the next seq-num is updated and the transmitter waits for another A_out() or transmits the buffered messages by building new packets based on next-seqnum.
The timer is stopped whenever a proper ack is received. Also, when a buffered message is sent to Layer 5 a timer is started along with it.

Through out this process, whenever a timeout occurs the packet previously sent is resent and a timer is started accordingly.

Go Back-N protocol

A static timeout is used here, chosen based on the best throughput observed.
The implementation includes building packets with increasing seq-num's. when a message arrives at A_output(), it is sent to Layer 5 or buffered based on whether it lies within the sender window or not. When a packet is sent to Layer 5 then it is also buffered (win_buffer), so that it maybe resent in the case of a timeout. Here, the timer is started only for the base packet and the next seq-num is updated. If the received message lies out of the window then a packet is built and buffered.

At B_input(), similar to ABT we check for in-order delivery as well as packet correctness. An acknowledgement will be sent based on whether it is the expected packet or not (corrupted packets included).

At A_input(), the received ack is checked whether it is the expected ack or not. In case of an unexpected ack, the previous packet will be resent. Upon correct reception of ack, the timer is stopped or restarted, based on whether it is the last available packet in the window or not. Also, we update the window buffer accordingly.
Here buffered packets, if any, can be sent.

On timeout, the packets that lie within the window buffer are resent.

Multiple Timers

A limitation to the implementation of selective repeat protocol is that it needs multiple timers for the correct functioning of the protocol, as each sent packet uses individual timers rather than a single timer as is the case with Go back-N protocol. However, by implementing virtual timers, multiple timers can be virtually created using a single available timer.

The logic used here to implement virtual timers is that, whenever a timer is running and a new packet is to be transmitted, we store the starting time (simulator time) along with it's possible timeout value. Now, whenever an event such as A_output, A_input, B_input or timer interrupt happens we update the current timeout of the buffered timers (virtual timers). When a timer is stopped normally (via ack) or abnormally (interrupt) we start a new timer for buffered timer (if any) with the least timeout. Thereby, alternating between a real timer and a buffered/virtual timer, we can implement multiple timers using a single real timer.

Selective Repeat protocol

The A_output() builds packets and sends them if they lie within the sender's window else the packets are buffered. We use a window buffer (for resending) similar to GBN where, all the sent packets, currently without an acknowledgement are stored.

At B_input() receives packets within the receiver window and sends them to Layer 5 if they are in order, else it buffers them in recv buffer only if they are not duplicates. Correctly received packets send acknowledgements to the sender. If a base packet is received any contiguous buffered packets are sent to layer 5 along with subsequent generation of their respective ack's. Any packets received within the prev window (Base-N to Base-1) are only acknowledged. The recv window moves forward as per the packets received.

At A_input(), the received acks if correct and lie within the window, are used to update the window buffer, timer buffer, next seqnum, base etc accordingly. If any buffered packets exist they are sent here.

Upon timeout, the packet for which the timeout occurred is only resent.

Also, as stated in the multiple timers section above, the timers alternate between actual timers and buffered timers based on events and individual timeout value. Priority is given to a timeout with the least value. At any time the timeout can be calculated as the difference between maximum timeout and the current system time along with the starting time of the packet.