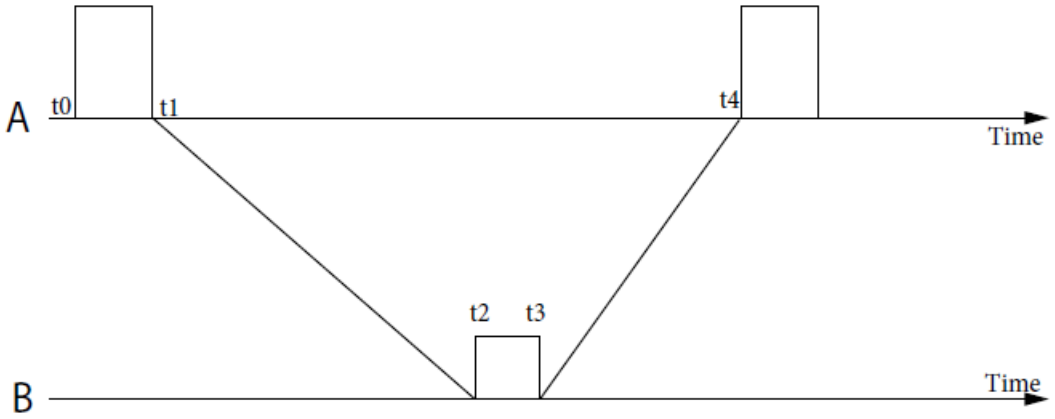
Principles of Computer Networks

Tutorial 4

Problem 1



a) $t_1 = t_0 + N_p/R$

b) $t_2 = t_1 + d_{AB}/C = t_0 + N_p/R + d_{AB}/C$

c) $t_3 = t_2 + N_a/R = t_0 + N_p/R + d_{AB}/C + N_a/R$

d) $t_4 = t_3 + d_{AB}/C = t_0 + N_p/R + d_{AB}/C + N_a/R + d_{AB}/C = t_0 + (N_p + N_a)/R + 2d_{AB}/C$

e) $T_{ct} = t_4 - t_0 = (N_p + N_a)/R + 2d_{AB}/C$

f) Average rate is the number of transmitted bytes in packet per communication time:

$$R_A = \frac{N_p}{T_{ct}} = \frac{N_p}{(Np\ +\ Na)/R\ +\ 2dAB/C}$$

g) Link utilization is link active time (packet transmission time) over the communication time

$$U_L = \frac{N_P/R}{T_{ct}} = \frac{N_p/R}{(Np\ +\ Na)/R\ +\ 2dAB/C}$$

h) All of the *n* packets need the same propagation delay, and the total transmission delay of the *n* packets will be *n* multiples of that of one packet. Further, one ACK will be sent. Consequently, the total time to transmit *n* packets and receive one acknowledgment is $(nN_p + N_a)/R + 2d_{AB}/C$. During this duration, *n* packets are transmitted with total number of bytes of nN_p. Then, the average rate will be

$$R_n = \frac{nN_p}{(nN_p + N_a)/R + 2\frac{d_{AB}}{C}}$$

i) Link utilization is the transmission time of the *n* packets (not acknowledgments) over the total time:

$$U_n = \frac{nN_p/R}{(nN_p + N_a)/R + 2\frac{d_{AB}}{C}}$$

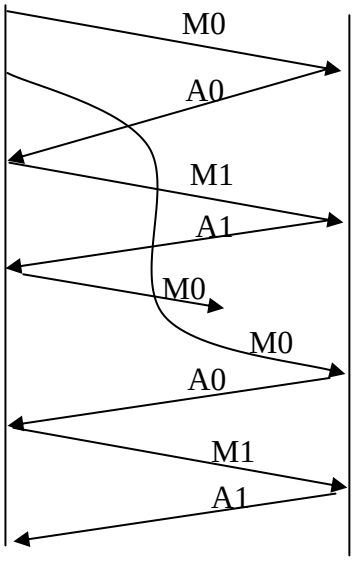
j) Congestion occurs if the total rate is above the channel capacity. So, we need to have $MR_n \leq R$ in order to avoid congestion.

$$\frac{MnN_p}{(nN_p + N_a)/R + 2\frac{d_{AB}}{C}} \leq R$$

$$n \leq \frac{2R\frac{d_{AB}}{C} + N_a}{(M-1)N_p}$$

k) The solution here is similar to that of the single-ACK case in (h), (i), and (j). It is the last sent ACK that will matter, why?

Problem 2



old version of M0 accepted!

Problem 3

Because the A-to-B channel can lose request messages, A will need to timeout and retransmit its request messages (to be able to recover from loss). Because the channel delays are variable and unknown, it is possible that A will send duplicate requests (i.e., resend a request message that has already been received by B). To be able to detect duplicate request messages, the protocol will use sequence numbers. A 1-bit sequence number will suffice for a stop-and-wait type of request/response protocol.
A (the requestor) has 4 states:

- **"Wait for Request 0 from above."** Here the requestor is waiting for a call from above to request a unit of data. When it receives a request from above, it sends a request message, R0, to B, starts a timer and makes a transition to the "Wait for D0" state. When in the "Wait for Request 0 from above" state, A ignores anything it receives from B.

- **"Wait for D0"**. Here the requestor is waiting for a D0 data message from B. A timer is always running in this state. If the timer expires, A sends another R0 message, restarts the timer and remains in this state. If a D0 message is received from B, A stops the time and transits to the "Wait for Request 1 from above" state. If A receives a D1 data message while in this state, it is ignored.

- **"Wait for Request 1 from above."** Here the requestor is again waiting for a call from above to request a unit of data. When it receives a request from above, it sends a request message, R1, to B, starts a timer and makes a transition to the "Wait for D1" state. When in the "Wait for Request 1 from above" state, A ignores anything it receives from B.

- **"Wait for D1".** Here the requestor is waiting for a D1 data message from B. A timer is always running in this state. If the timer expires, A sends another R1 message, restarts the timer and remains in this state. If a D1 message is received from B, A stops the timer and transits to the "Wait for Request 0 from above" state. If A receives a D0 data message while in this state, it is ignored.

The data supplier (B) has only two states:

- **"Send D0."** In this state, B continues to respond to received R0 messages by sending D0, and then remaining in this state. If B receives a R1 message, then it knows its D0 message has been received correctly. It thus discards this D0 data (since it has been received at the other side) and then transits to the "Send D1" state, where it will use D1 to send the next requested piece of data.

"Send D1." In this state, B continues to respond to received R1 messages by sending D1, and then remaining in this state. If B receives a R1 message, then it knows its D1 message has been received correctly and thus transits to the "Send D1" state.