SYSC 4502 Assignment 1

Jessica Morris 100882290

February 5th, 2017

1. Assuming that collisions are chained to the end of the list, the resulting table is:

   0 → ⏚
   1 → 4371 → ⏚
   2 → ⏚
   3 → 1323 → 6173 → ⏚
   4 → 4344 → ⏚
   5 → ⏚
   6 → ⏚
   7 → ⏚
   8 → ⏚
   9 → 4199 → 9679 → 1989 → ⏚

2. Consider a binary tree of height k. At depth 0, there will be $2^0 = 1$ nodes. At depth 2, there will be $2^2 = 4$ nodes. At depth k, there will be 2^k nodes. If all levels of the tree are full, the number of nodes in the tree is given by:

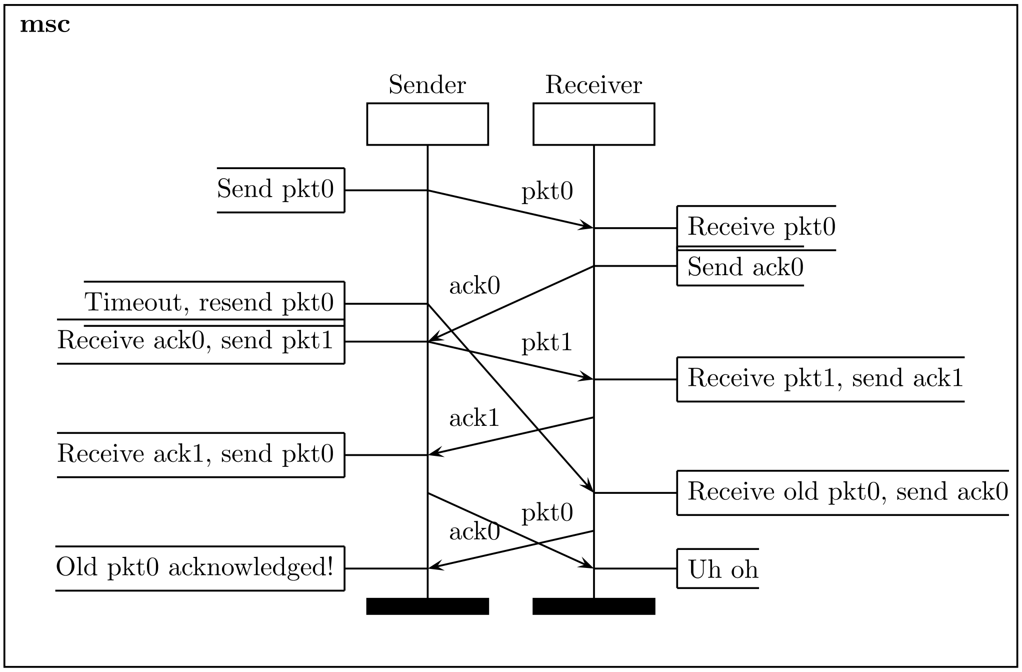
$$1 + 2 + 4 + ... + 2^{k-1} + 2^k = \sum_{i=1}^{k+1} 2^{i-1} = 2^{k+1} - 1$$

   Thus, the maximum number of nodes in a binary tree of height h is $2^{h+1} - 1$.

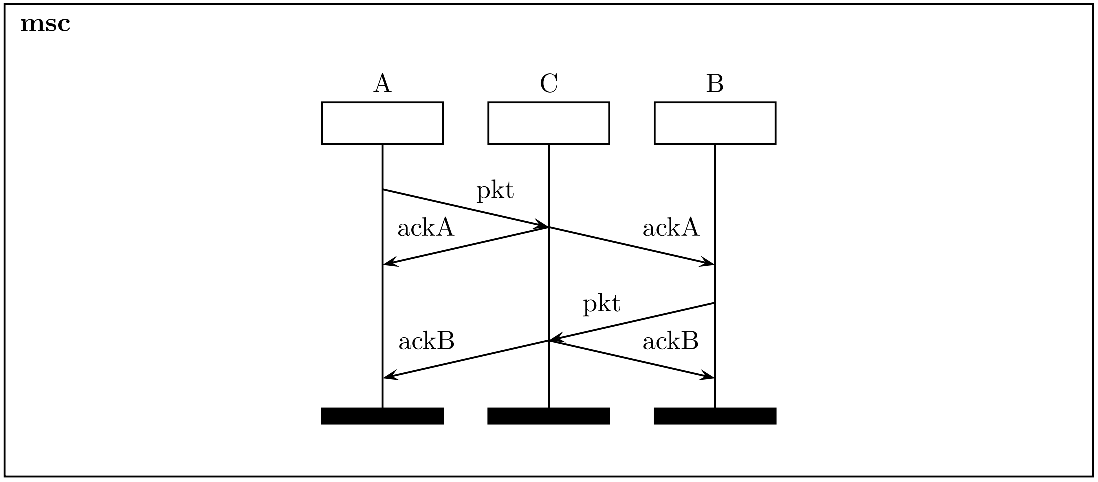
3. To enter a deadlock case, consider the case where the sender is in state "Wait for call 1 from above", while the receiver is in state "Wait for 1 from below". The sender sends a packet with sequence 1, and transitions to "Wait for ACK or NAK 1", while the receiver receives the packet and transitions to "Wait for 0 from below", sending ACK 1. Now, consider if ACK 1 is corrupted on its way to the sender. Upon receiving the corrupted ACK, the sender will re-send packet 1, which the receiver will NAK because it is waiting for 0. So, the sender will re-send packet 1, and the receiver will NAK. The sender will be trapped in state "Wait for ACK or NAK 1", while the receiver will be trapped in state "Wait for 0 from below", and progress cannot be achieved.

4. The NAK-based protocol is less effective than a ACK-based protocol if the sender is sending data infrequently. For the NAK-based protocol, an error for packet x will only be detected when packet $x + 1$ is received by the receiver; which, if data is sent infrequently, may take a while, and therefore result in a long error recovery time. For the case of a data stream over a reliable channel, an ACK-based protocol would not be as desirable as a NAK-based protocol, as using ACKs would introduce significantly more overhead than only NAKs.

5. Alternating bits failing:

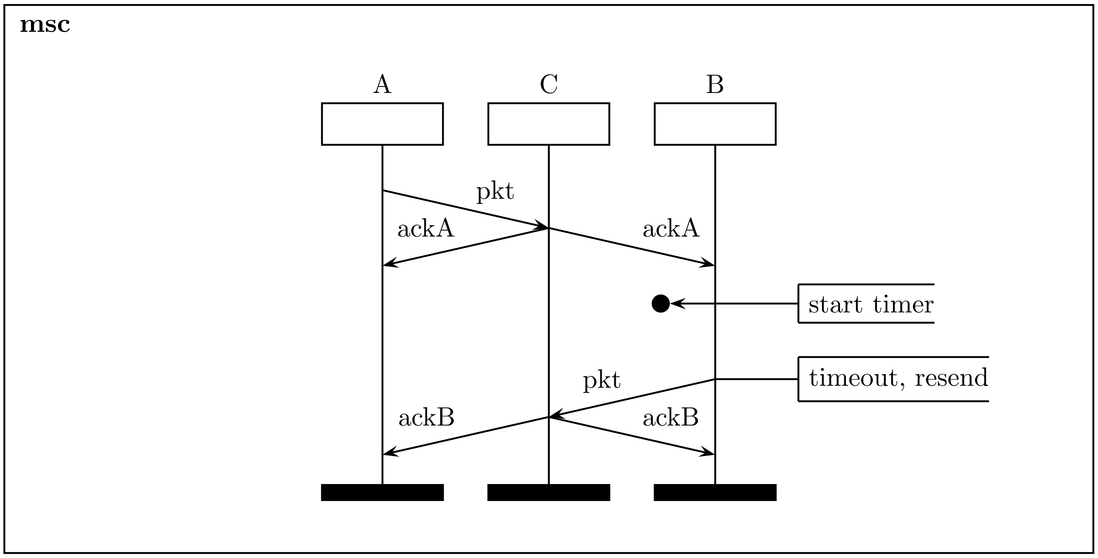


6. (a) Happy path MSC:



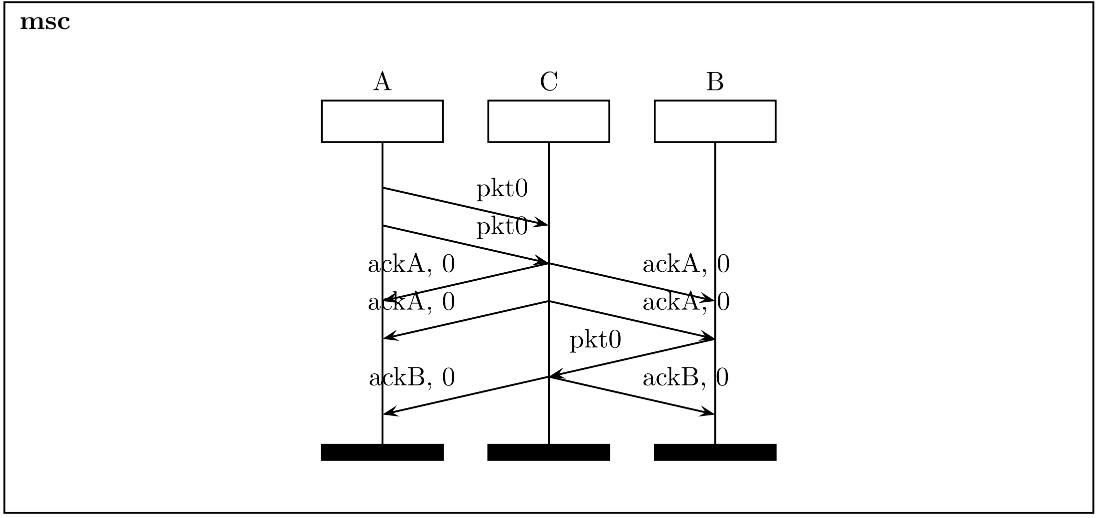
The receiver (C) should notify both senders when it has received a packet from one sender. e.g. Receive a packet from B, then send ACK B to A and B, so that B knows it can yield until an ACK A because it is A's turn to send data, and A knows that it is its turn to send data.

Sad path 1, a data packet is lost or corrupted:



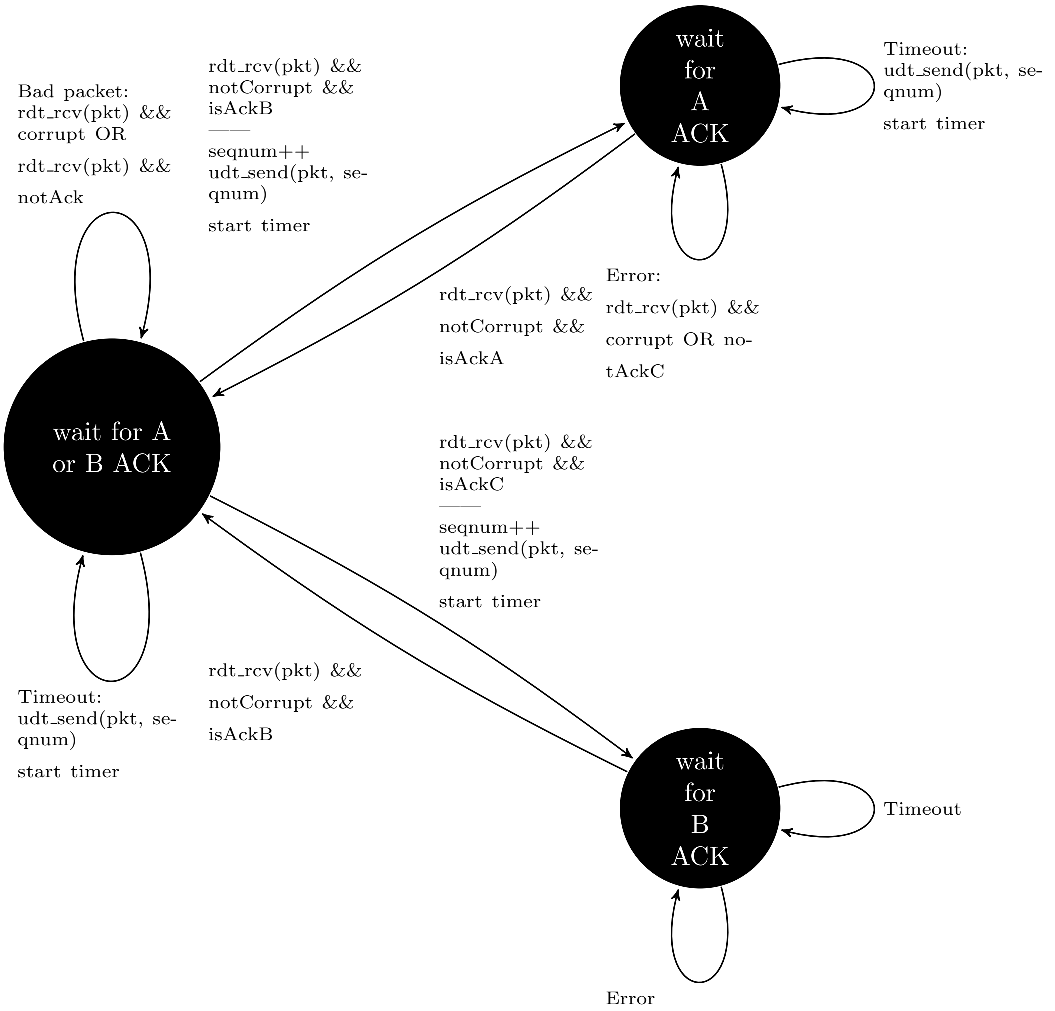
Senders should start a timer after sending data to C; if the data is not acknowledged before timeout, resend the data.

Sad path 2, data gets duplicated by sender or in transmission:

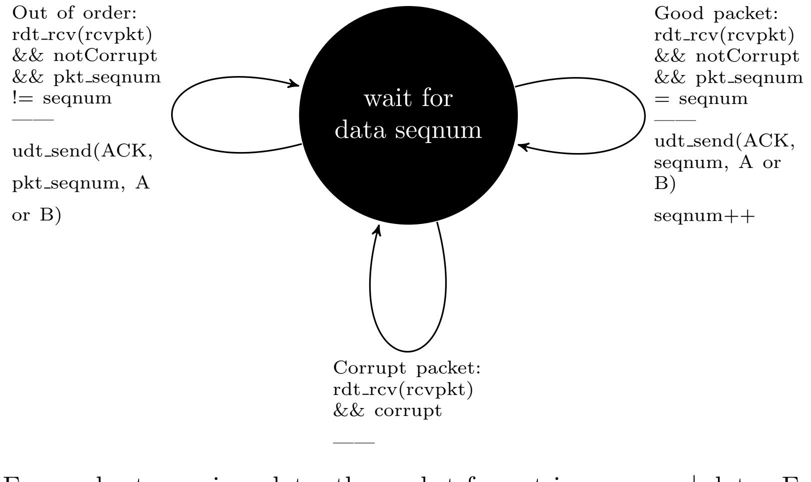


The receiver (C) needs to notify the sender of which sequence number they are acknowledging. Duplicate data should be acknowledged, so that the sender does not send it again.

(b) FSM for sender (A or B):



FSM for receiver (C):


Out of order:
rdt_rcv(rcvpkt) && notCorrupt && pkt_seqnum != seqnum
——
udt_send(ACK, pkt_seqnum, A or B)
wait for data seqnum
Good packet:
rdt_rcv(rcvpkt) && notCorrupt && pkt_seqnum = seqnum
——
udt_send(ACK, seqnum, A or B)
seqnum++
Corrupt packet:
rdt_rcv(rcvpkt) && corrupt
——

(c) For sender-to-receiver data, the packet format is: seqnum | data. For receiver-to-sender control, the packet format is: acknum | sender (A or B).

(d) Interfaces on the sender: rdt_rcv(rcvpkt), rdt_send(data) to upper layer, udt_send(src, dst, pkt, seqnum) call to lower layer. Interfaces on the receiver: rdt_rcv(rcvpkt), udt_send(src, dst, ACK, seq) to lower layer.

7. (a) Consider a server which distributes the data rate of the channel evenly to each peer as $\frac{u_s}{N}$ bits

per second, resulting in the total bandwidth used being $\frac{u_s}{N} \times N = u_s$. This data rate will not exceed the rate at which clients can receive bits, as it is given that $\frac{u_s}{N} \leq d_{min}$. If each client is downloading a file of size F bits, then it will take $\frac{F}{u_s/N} = \frac{NF}{u_s}$ seconds for each client to download their file.

(b) Consider a server which distributes the data rate of the channel to each peer as d_{min} bits per second, giving a total bandwidth of Nd_{min}. This will not exceed the bandwidth that the server can trasmit, as it is given that $u_s \geq Nd_{min}$. If each client is downloading a file of size F bits, then it will take $\frac{F}{d_{min}}$ seconds for each client to download their file.