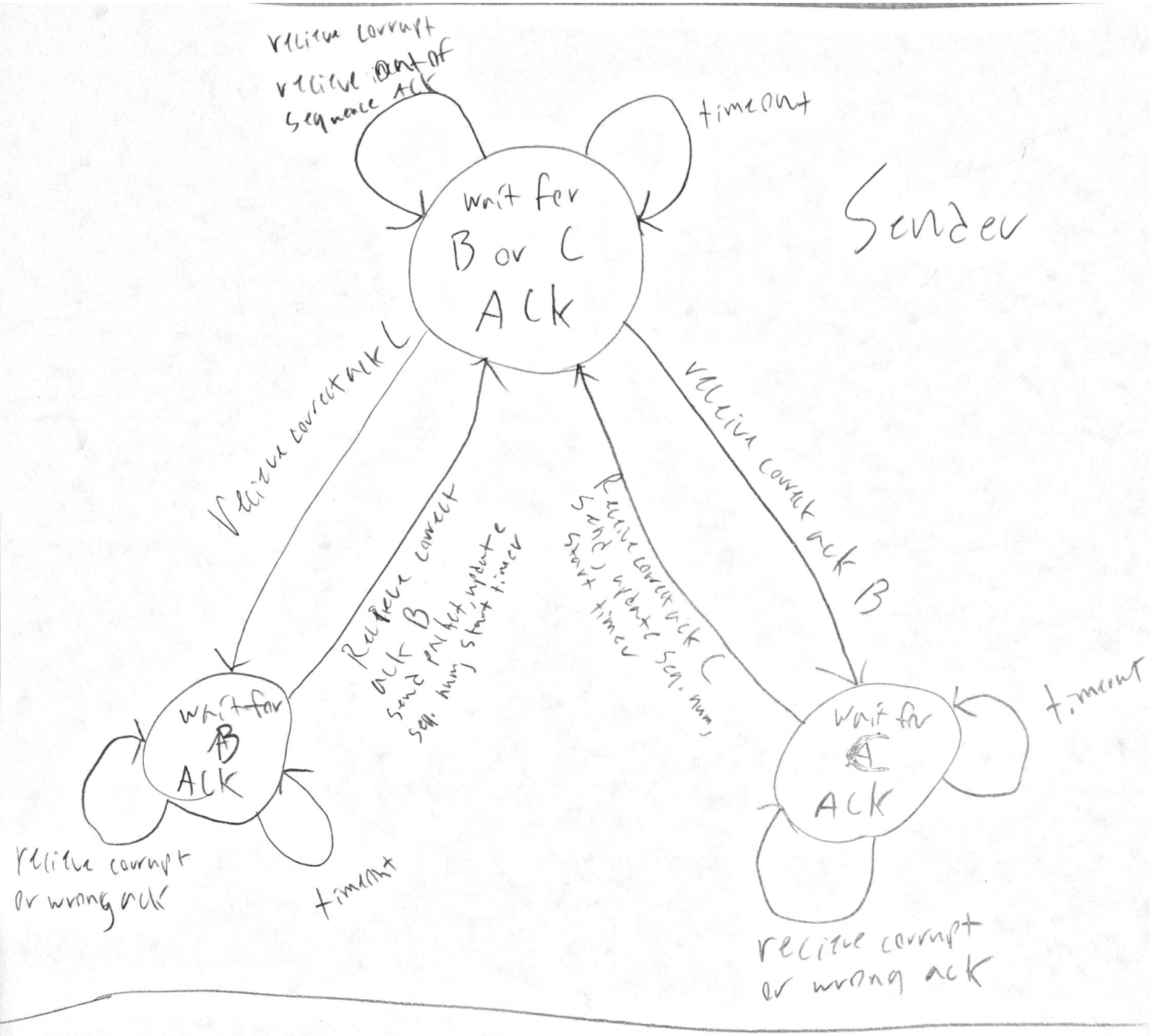
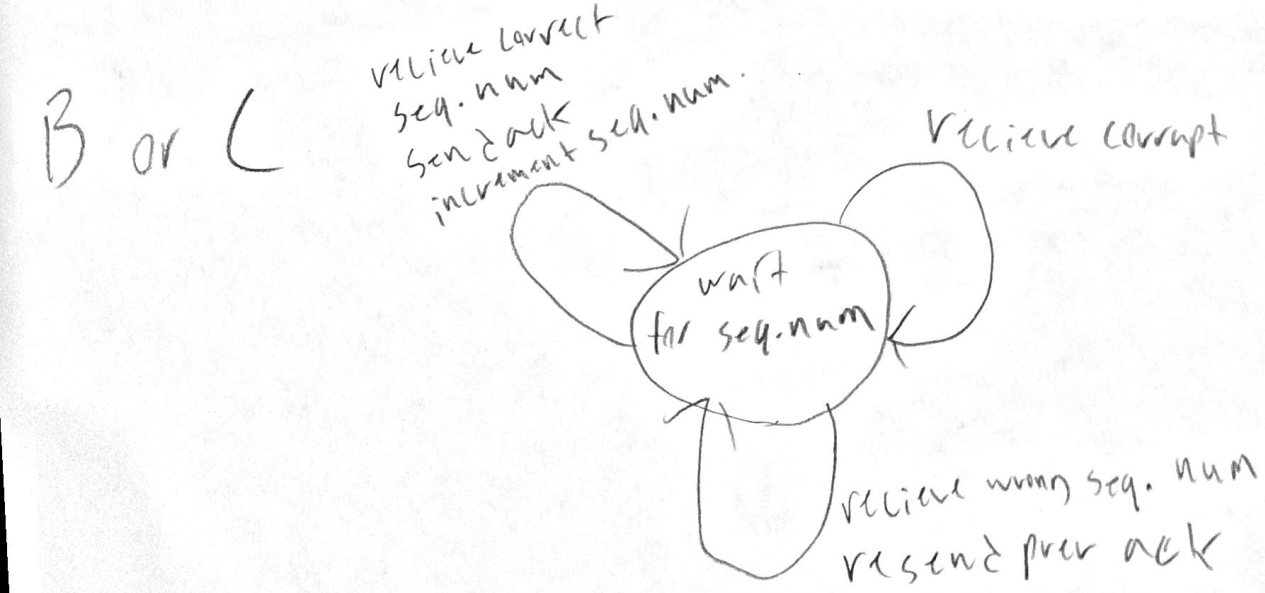
James Barbour
Homework 6

1. See next page
2. (1Gb/s)*RTT/(mss bytes) = window size
3. No, if one connection dominates and which connection will drop connections is non-deterministic, both connection will remain roughly at the same relative rates
4. a) when packet loss occurs, w becomes w/2, continually add 1 until reaching w, sum of i = 0 to w/2 of (w/2 + 1), the window growth is the summation, and the bottom half of the formula is the convergence of the summation
   b) For a large w, 3/8w^2 approaches 3/4w^2, so L is approx. 3/8w^2, or w=sqrt(8/(3L)), inserted into the formula R=(.75*w*mss)/RTT gives us the formula that's given
5. 2RTT+OT/R+(k-1)(ST/R+RTT)-(2^P-1)ST/R
6. Yes, assuming the window size is not sufficiently small to negate the problem, if a file over 4GB is sent over a very fast but very high-latency connection, say pushing a software update to a future Mars rover. Assume the sequence number starts at 1, eventually wraps around and hits 1 again. If a packet is lost for a sequence number that has been hit more than once, we won't know which actual packet it was because that sequence number has occurred more than once.
7. From RFC 2001: "It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire."
   Receiving one or two duplicate packets is not a good indication of packet loss as it could be a simple reordering, but receiving three or more duplicates is a good indication of a lost segment.
8. a) K = log_3(O/S)
   b) Q = log_3(S/R+RTT)
   c) Latency = log_3(O/R+RTT+P*(O/R+RTT))
9. 0: 2^6 00000000-00111111
   1: 2^5 01000000-01011111
   2: 2^5 01100000-01111111
   3: 2^6 10000000-10111111
   4: 2^6 11000000-11111111
10. 1: 223.1.17.64/26   (01)
    2: 223.1.17.128/25 (1)
    3: 223.1.17.0/26   (00)


Sender
wait for B or C ACK
timeout
wait for B ACK
wait for C ACK
timeout
recieve corrupt or wrong ack
timeout
recieve corrupt or wrong ack


B or C
recieve correct seq. num
send ack
increment seq. num
recieve corrupt
wait for seq. num
recieve wrong seq. num
resend prev ack