1)

T	Sender state	Receiver state	Packet type sent	Seq # or ACK # sent
0	Wait ACK0	Wait0 from below	data	0
1	Wait ACK0	Wait1 from below	ACK	0
2	Wait ACK1	Wait1 from below	data	1
3	Wait ACK1	Wait1 from below	ACK	0
4	Wait ACK1	Wait1 from below	data	1
5	Wait ACK1	Wait1 from below	ACK	0
6	Wait ACK1	Wait1 from below	data	1

It was passed up one time @ T= 1.

2)

T	Sender state	Receiver state	Packet type sent	Seq # or ACK # sent
0	Wait ACK0	Wait0 from below	data	0
1	Wait ACK0	Wait0 from below	ACK	1
2	Wait ACK0	Wait0 from below	data	0
3	Wait ACK0	Wait0 from below	ACK	1
4	Wait ACK0	Wait0 from below	data	0
5	Wait ACK0	Wait0 from below	ACK	1
6	Wait ACK0	Wait0 from below	data	0

None were passed up.

3)

```
00100100 01111011
10011100 10011111
```

Adds To: 11000001 00011010
Ones Comp: **00111110 11100101 (Checksum)**

4)

```
10100110 00101010
11101011 00101101
```

Adds To: 1 00010001 01010111
Carry: 00010001 01011000
Ones Comp: **11101110 10100111 (Checksum)**

5)

TCP is in slow start from 1-4, 8-11, and 28-32
Congestion Avoidance 4-7, 11-27, 32-35, 37-40
Fast Recovery 36

The first packet is lost at 8, and 28 due to timeout, and 36 due to triple duplicate ACKs.

SSTHRESH changes at 7 to around 5, 27 to around 12.5, and once again at 35 to 15.

6)

TCP is in slow start from 1-4, 9-11, 28-31, and 38-40
Congestion Avoidance 4-8, 12-27, 31-37
Fast Recovery 12

The first packet is lost at 9, 12, 28, and 38 due to timeout, 11, and 27 due to triple duplicate ACKs.

SSTHRESH changes at 8 to around 6.5, 11 to around 2.5, 28 to around 7.5, and again at 38 to approximately 7.

7)

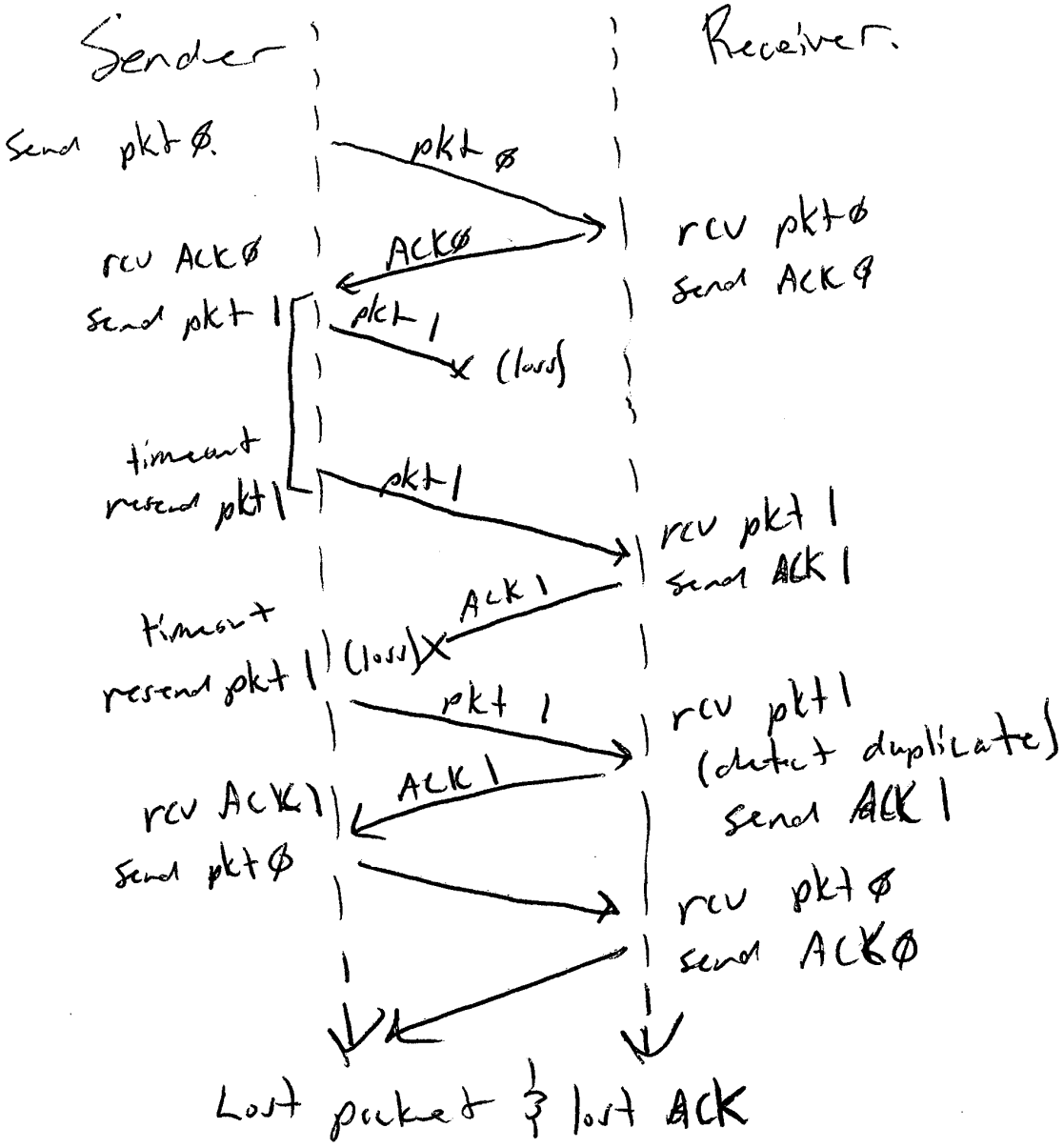
a.
```
  01010011
  01100110
+ ________
  10111001
  01110100
+ ________
  100101101
  00101101      (Overflow discarded)
```
11010010 (Complement of number above)

b. Ones complement is easier to detect errors in, which is why UDP uses it.

c. The receiver detects error by adding the binary checksums together and taking the complement of the number. If it is all 1’s then there is no error.

d. No error bits will ever go undetected because an error free transfer will *always* have a checksum of all 1’s.

8)



9)

A timer would need to be added whose value is greater than the known round-trip prop delay. A timeout would also need to be added to wait for ACK. If the timeout would occur we would just resend the last sent packet. This can be done because during a timeout the receiver never gets the packet, so it will be resent, when it arrives it appears as though the original packet made it to the receiver without a problem. Also if an ACK is lost, the receiver will send the packet on a timeout, which looks like the same thing that would happen if an ACK arrived garbled.. This is okay because the receiver already has a method of dealing with a garbled ACK.

10)

a. Average Packets Dropped (Unmodified program) Averages:

Input	195	205	186	209	189	196.8
Output	117	100	149	77	102	109

b. Average Packets Dropped (Modified maxQ to 50) Averages:

Input	0	0	2	11	0	2.6
Output	0	0	9	0	0	1.8

Until maxQ was set to 49 I was losing more than an average of 5 packets. At 49 the numbers decreased below this.

c. Average Packets Dropped (Modified prob to 0.65) Averages:

Input	1	8	8	3	0	4
Output	4	5	0	1	8	3.6

Until prob was decreased to .65 from 1 I was getting much higher values than 5 for the average number of dropped packets, once it reached this point the average hovered around 4, much lower and the average dropped packets decreased to 0.

d. If there is exactly one packet arriving at each input port at every time unit there will not be any dropped packets. This is because the queues will fill equally instead of one overflowing while others sit empty. The output queues on the other hand may still overflow. If each output queue was also getting exactly one packet per time unit there would be no overflow, but seeing that this is not the case and multiple packets can arrive at once, there is still a risk of flooding the queue.