I have read and understood the course academic integrity

Chapter 3

1) **TCP congestion control** – Consider sending a large file from one host to another over a TCP connection that has no loss.
   a) Suppose TCP uses AIMD for its congestion control without slow start. Assuming cwnd increases by 1 MSS every time a batch of ACKs is received, and assuming constant round-trip times, how long does it take for cwnd to increase from 1 MSS to 6 MSS? **(4 points)**
   b) What is the average throughput (in terms of MSS and RTT) for this connection up through time = 5RTT? **(5 points)**

a) Because in Additive Increase state, sending rate increase by one on every RTT.
   So Ans_time = 6 -1 = 5 RTT

b) Avg.throughput = (1+2+3+4+5) mss / 5RTT = 3/4 * 6MSS / RTT = 3 MSS/RTT

2) **TCP throughput** – Consider that a single TCP (Reno) connection uses one 15 Mbps link which does not buffer any data. Suppose that this link is the only congested link between the sending and the receiving hosts. Assume that the TCP sender has a huge file to send to the receiver, and the receiver's receive buffer is much larger than the congestion window. We also make the following assumptions: each TCP segment size is 1,500 bytes; the two-way propagation delay of this connection is 150 msec; and this TCP connection is always in congestion avoidance phase, that is, ignore slow start.
   a) What is the maximum window size (in segments) that this TCP connection can achieve? **(4 points)**
   b) What is the average window size (in segments) and average throughput (in bps) of this TCP connection? **(4 points)**
   c) How long would it take for this TCP connection to reach its maximum window again after recovering from a packet loss? **(4 points)**
   d) Repeat (a), (b), (c) but replacing the 15 Mbps link with a 15 Gbps link. Do you see a problem for the TCP connection in this scenario? Suggest a simple solution to this problem. **(15 points)**

a) Wmax = LinkSpeed * RTT / SegmentSize
   = 15*10^6 b * 0.15/ （8*1500） bits
   = 187.5 ≈ 187

b) Avg.W = 3/4 W = 3/4 * 187 = 140.25 seg
   Avg.throughPut = 3/4 W*MSS/ RTT = 3/4 * 30*1500bytes/ 0.15*2 = 11220000 bps

c) In Reno the window size will increase by 1 per RTT after cutted in half.
   Time = (W- W/2)*RTT = (187 – 187/2) * 0.15 = 14.025s

d) Wmax = LinkSpeed * RTT / SegmentSize = 187500 seg
   Avg.W = 3/4 W = 3/4*187500 = 140625
   Avg.throughput = 3/4 * 187500*1500*8/0.1 = 1.125*10^10bps
   Time = w/2 * RTT = 14062.5s

   The recover time will be 14062.5s, it is too long. Using TCP CUBIC would be a possible solution, it will increase the recover speed. Or implement ECN to window size also can reduce the recover time.

3) **Longest prefix matching** – Consider a datagram network using 32-bit host addresses.

a) Suppose that a router has three interfaces, numbered 0, 1, 2, and that packets are to be forwarded to these link interfaces as follows. Any address not within the ranges in the table below should not be forwarded to an outgoing link interface. Create a forwarding table using longest prefix matching. **(6 points)**

3.

a)

0000000*	0
01011101	1
0111011*	2

b).

00000000	} 0
000000010	
000000011 0000000 00000000 00000000	
01010101	1
0101011X	2

4) **Subnets** – An organization has been assigned the prefix 222.3.4.0/24 and wants to form subnets for 4 departments A, B, C, and D, with hosts as follows:

A	95 hosts
B	47 hosts
C	28 hosts
D	17 hosts

There are 187 hosts in all. Give a possible arrangement of subnet IP addresses to make this possible. Use the notation a.b.c.d/x. **(16 points)**

4.

A: $95 < 2^7$ $32-7=25$

222.3.4.0/25

B: $47 < 2^6$ $32-6=26$

222.3.4.0/26

C: $28 < 2^5$ $32-5=27$

222.3.4.0/27

D: $17 < 2^5$ $32-5=27$

222.3.4.0/27

5) **OpenFlow** – Consider the SDN OpenFlow network shown in Figure 4.30 (slide 84). Suppose that the desired forwarding behavior for datagrams arriving at s2 is as follows:

- Any datagrams arriving on input port 1 from hosts h5 or h6 that are destined to hosts h1 or h2 should be forwarded over output port 2; (**4 points**)
- Any datagrams arriving on input port 2 from hosts h1 or h2 that are destined to hosts h5 or h6 should be forwarded over output port 1; (**4 points**)
- Any arriving datagrams on input ports 1or 2 and destined to hosts h3 or h4 should be delivered to the host specified; (**4 points**)
- Hosts h3 and h4 should be able to send datagrams to each other. (**4 points**)

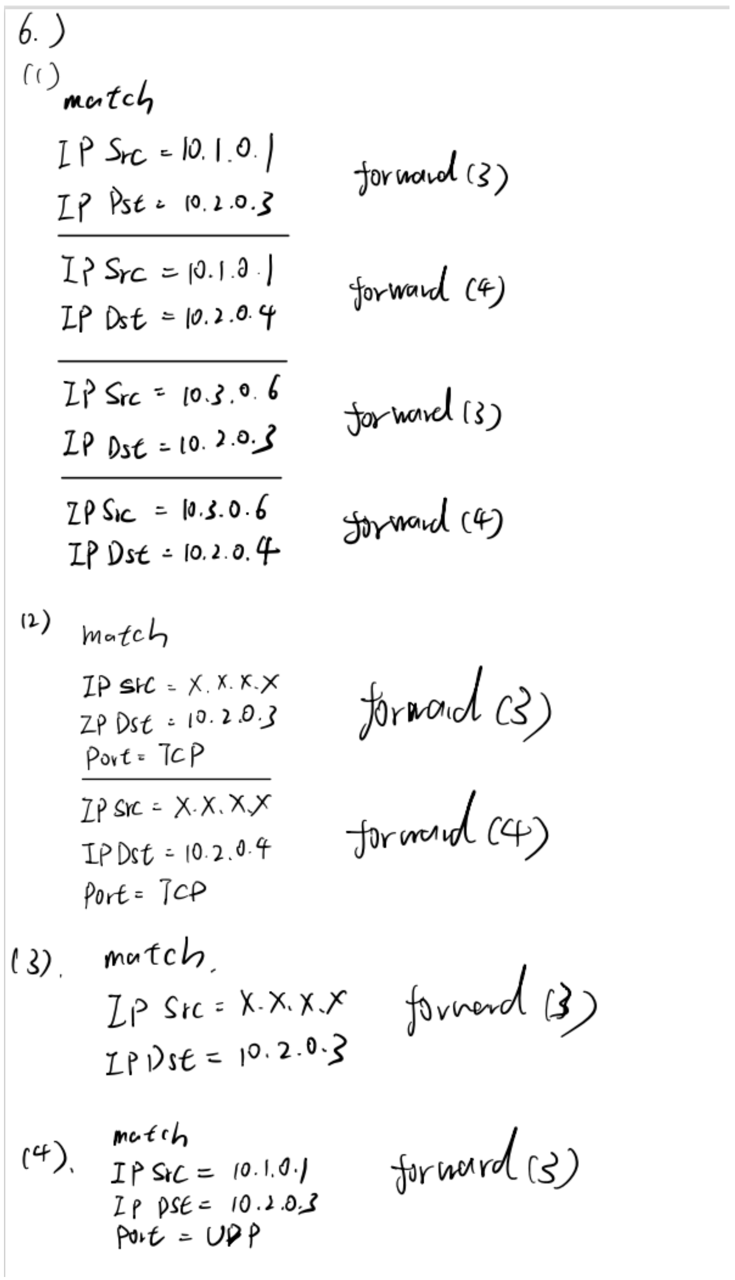
Specify the flow table entries in s2 that implement this forwarding behavior.

5).

match	action
ingress port = 1 IP Src = 10.3.X.X IP Dst : 10.1.X.X	forward (2)
match ingress port = 2 IP Src = 10.1.X.X IP Dst = 10.3.XX	forward (1)
match IP Dst = 10.2.0.3	forward (3)
match IP Dst = 10.2.0.4	forward (4)

6) **OpenFlow** – Consider again the SDN OpenFlow network shown in Figure 4.30 (slide 84). Suppose we want switch s2 to function as a firewall. Specify the flow table in s2 that implements the following firewall behaviors (specify a different flow table for each of the four firewalling behaviors below) for delivery of datagrams destined to h3 and h4. You do not need to specify the forwarding behavior in s2 that forwards traffic to other routers.

- Only traffic arriving from hosts h1 and h6 should be delivered to hosts h3 or h4 (i.e., that arriving traffic from hosts h2 and h5 is blocked). (**4 points**)
- Only TCP traffic is allowed to be delivered to hosts h3 or h4 (i.e., that UDP traffic is blocked). (**4 points**)
- Only traffic destined to h3 is to be delivered (i.e., all traffic to h4 is blocked). (**4 points**)
- Only UDP traffic from h1 and destined to h3 is to be delivered. All other traffic is blocked. (**4 points**)



6.)

(1) match

IP Src = 10.1.0.1 IP Dst = 10.2.0.3	forward (3)
IP Src = 10.1.0.1 IP Dst = 10.2.0.4	forward (4)
IP Src = 10.3.0.6 IP Dst = 10.2.0.3	forward (3)
IP Src = 10.3.0.6 IP Dst = 10.2.0.4	forward (4)

(2) match

IP Src = X.X.X.X IP Dst = 10.2.0.3 Port = TCP	forward (3)
IP Src = X.X.X.X IP Dst = 10.2.0.4 Port = TCP	forward (4)

(3). match

IP Src = X.X.X.X IP Dst = 10.2.0.3	forward (3)

(4). match

IP Src = 10.1.0.1 IP Dst = 10.2.0.3 Port = UDP	forward (3)