CSE 123: Computer Networks

Homework 4

Out: 11/25, Due: 12/02

Instructions:

1. Turn in a physical copy at the beginning of the class on 11/18.
2. Ensure the HW cover page has the following information clearly written:
    a. Name
    b. UCSD email
    c. PID
3. Please contact the TAs or post on Piazza to seek any clarification.
4. The homework is to be done individually.

1. Fair Queuing

Consider a router that is managing three flows, on which packets of constant size arrive at the following wall clock times:
Flow A: 1, 2, 4, 6, 7, 9, 10
Flow B: 2, 6, 8, 11, 12, 15
Flow C: 1, 2, 3, 5, 6, 7, 8

All three flows share the same outbound link, on which the router can transmit one packet per time unit. Assume that there is an infinite buffer space.

a) Suppose the router implements fair queuing. For each packet, give the wall clock time when it is transmitted by the router. Arrival times ties are to be resolved in the order of A, B , C. Note that the wall clock time T = 2 is FQ-clock time A_1 = 1.5.

Wall Clock	A_{i}	Arrivals (F_i)	Sent	A's Queue	B's Queue	C's Queue
1	1	A1(2), C1(2)	A1	A1		C1
2	1.5	A2(3), B2(2.5), C2(3)	C1	A2	B2	C1, C2
3	1.833	C3(4)	B2	A2	B2	C2, C3
4	2.166	A4(4)	A2	A2, A4		C2, C3
5	2.666	C5(5)	C2	A4		C2, C3, C5
6	3.166	A6(5), B6(4.166), C6(6)	A4	A4, A6	B6	C3, C5, C6
7	3.5	A7(6), C7(7)	C3	A6, A7	B6	C3, C5, C6, C7
8	3.833	B8(5.166), C8(8)	B6	A6, A7	B6, B8	C5, C6, C7, C8
9	4.166	A9(7)	A6	A6, A7, A9	B8	C5, C6, C7, C8
10	4.5	A10(8)	C5	A7, A9, A10	B8	C5, C6, C7, C8
11	4.833	B11(6.166)	B8	A7, A9, A10	B8, B11	C6, C7, C8
12	5.166	B12(7.166)	A7	A7, A9, A10	B11, B12	C6, C7, C8
13	5.5	-	C6	A9, A10	B11, B12	C6, C7, C8

14	5.833	-	B11	A9, A10	B11, B12	C7, C8
15	6.166	B15(8.166)	A9	A9, A10	B12, B15	C7, C8
16	6.5	-	C7	A10	B12, B15	C7, C8
17	6.833	-	B12	A10	B12, B15	C8
18	7.166	-	A10	A10	B15	C8
19	7.5	-	C8		B15	C8
20	8.0	-	B15		B15	

b) Suppose the router implements weighted fair queuing, where flows A and B are given equal share of the capacity, and flow C is given twice the capacity of flow A. For each packet, give the wall clock time when it is transmitted.

Wall Clock	A_{i}	Arrivals (F_i)	Sent	A's Queue	B's Queue	C's Queue
1	1	A1(2), C1(1.5)	C1	A1		C1
2	1.5	A2(3), B2(2.5), C2(2)	A1	A1, A2	B2	C2
3	1.833	C3(2.5)	C2	A2	B2	C2, C3
4	2.166	A4(4)	B2	A2, A4	B2	C3
5	2.5	C5(3)	C3	A2, A4		C3, C5
6	3	A6(5), B6(4.0), C6(3.5)	A2	A2, A4, A6	B6	C5, C6
7	3.333	A7(6), C7(4.0)	C5	A4, A6, A7	B6	C5, C6, C7
8	3.666	B8(5.0), C8(4.5)	C6	A4,A6, A7	B6, B8	C6, C7, C8
9	4.0	A9(7)	A4	A4, A6, A7, A9	B6, B8	C7, C8
10	4.333	A10(8)	B6	A6, A7, A9, A10	B6, B8	C7, C8
11	4.666	B11(6.0)	C7	A6, A7, A9, A10	B8, B11	C7, C8
12	5.0	B12(7.0)	C8	A6, A7, A9, A10	B8, B11, B12	C8
13	5.333	-	A6	A6, A7, A9, A10	B8, B11, B12	
14	5.833	-	B8	A7, A9, A10	B8, B11, B12	
15	6.333	B15(8.0)	A7	A7, A9, A10	B11, B12, B15	
16	6.833	-	B11	A9, A10	B11, B12, B15	
17	7.333	-	A9	A9, A10	B12, B15	
18	7.833	-	B12	A10	B12, B15	
19	8.333	-	A10	A10	B15	
20	8.833	-	B15		B15	

2. RED

Consider a RED gateway with MaxP = *p* and with an average queue length halfway between the two thresholds.

a) Calculate the probability that none of the first *n* packets is dropped.

tempP = p/2 (since avg length is half-way)

Probability of drop of 1st packet = P_1 = $\frac{p/2}{1-p/2}$

Similarly, P_2 = $\frac{p/2}{1-p}$

and hence P_i = $\frac{p/2}{1-i*p/2}$

Probability that none of the first n packets are dropped =

$(1-P_1)*(1-P_2)*....*(1-P_n)$

Solving the expression, we get $\frac{1-(n+1)p/2}{1-p/2}$

b) Find *p* such that the probability that none of the first *n* packets is dropped is α .

$$\frac{1-(n+1)p/2}{1-p/2}=\alpha$$

Solving this, we get $p=2(1-\alpha)/(n+1-\alpha)$

3. Token bucket

The transmission schedule (Table 1) for a given flow lists for each second the number of packets sent between that time and the following second. The flow must stay within the bounds of a token bucket filter. What bucket depth does the flow need for the following token rates? Assume the bucket is initially full.

a) 2 packets per second

In this problem, we assume that if packets arrive at t = t, then they need to be sent out at least by t=t+1 so, we get the 2 tokens replenished before we send the packets for that second out.

Time(seconds)	Packets sent	#tokens present	Final tokens
0	6	B	B-6+2 = B-4
1	4	B-4	B-4-4+2 = B-6
2	2	B-6	B-6-2+2 = B-6
3	0	B-6	B-6-0+2 = B-4
4	7	B-4	B-4-7+2 = B-9
5	1	B-9	B-9-1+2 = B-8

To keep the number of tokens in the bucket >= 0, we need B to be atleast 9.

b) 4 packets per second

Time(seconds)	Packets sent	#tokens present	Final tokens
0	6	B	B-6+4 = B-2

1	4	B-2	B-2-4+4 = B-2
2	2	B-2	B-2-2+4 = B
3	0	B	B = B (can't add more)
4	7	B	B-7+4 = B-3
5	1	B-3	B-3-1+4 = B

This time B should be at least 3.

Time(seconds)	Packets sent
0	6
1	4
2	2
3	0
4	7
5	1

Table .1

4. AIMD

Suppose a connection starts with *cwnd*=1 and increments *cwnd* by 1 each RTT with no loss, and sets *cwnd* to *cwnd/2*, rounding down, on each RTT with at least one loss. Assume that the propagation delays dominate so each windowful is sent more or less together. Packets 5, 13, 14, 23 and 30 are lost. Lost packets are retransmitted and the retransmitted packets are not lost again.

a) What is the window size each RTT, up until the first 40 packets are sent?
b) What packets are sent each RTT?

This is assuming there is no caching at the receiver. Selective retransmission could be done as well but the assumption needs to be stated.

Cwnd	**Packets sent**	**RTT**
1	1	1
2	2, 3	2
3	4, 5, 6	3
1	5	4
2	6, 7	5
3	8, 9, 10	6
4	11, 12, 13, 14	7
2	13, 14	8
3	15, 16, 17	9
4	18, 19, 20, 21	10
5	22, 23, 24, 25, 26	11
2	23, 24	12

3	25, 26, 27	13
4	28, 29, 30, 31	14
2	30, 31	15
3	32, 33, 34	16
4	35, 36, 37, 38	17
5	39, 40	18

Hint: in the first RTT, Data[1] is sent. There is no loss, so in the second RTT cwnd = 2 and Data[2] and Data[3] are sent.

5.TCP Reno

Assumptions:

1. each segment is of size 1 byte and Data[N] represents a segment (or a byte).
2. dupACK[M]/N represents duplicate ACK sent for byte (or segment) M triggered byte (or segment) N.

Suppose the window size is 100, and Data[1001] is lost. There will be 99 dupACK[1000]'s sent, which we may denote as dupACK[1000]/1002 through dupACK[1000]/1100. TCP Reno is used.

a) At which dupACK[1000]/N does the sender start sending new data?

```
The sender will send new data that is Data[1101] after it has retransmitted
Data[1001]. The retransmission will happen after the receipt of the 3rd
duplicate ack for data[1000]. Clearly the third dupACK is sent in response to
Data[1004]. Before retransmission the window size cwnd is halved as well. So
current cwnd = 50 and data in flight = 100-3 = 97 (-3 for the three dupACKs).
To be able to send anymore data (basically Data[1101]), we need
```

- `the inflight data < cwnd.`
- `and for each dupACK received, the inflight data reduces by 1 and cwnd increases by 1 (since TCP Reno increases sender's window by 1 for each dupACK received).`

```
Let n = # of inflight data = 97 as seen above.
cwnd = initial sender window = 50 (after halving it)
                               (53 according to rfc 2001. Either is okay)

n – x < cwnd + x

(To find the x dupACKs that we need to receive so that inflight packets are less
than the then sender window size allowing us to send the new Data i.e., Data[1101])
=> x > (n-cwnd)/2
=> x > (97-50)/2
=> x = 24.

So 24 more dupACKs are needed which will come as ACKs for Data[1005],
Data[1006] ...... till Data[1029].
```

The moment we get the **dupACK[1000]/1029**, the sender window would have an empty slot letting us send Data[1101].

b) When the retransmitted data[1001] arrives at the receiver, what ACK is sent in response?

```
When Data[1001] arrives at the receiver, all the packets from 1001 till 1100
would have been received and hence, receiver would send ACK for Data[1100] in
case of selective retransmission.
If no caching at the receiver, then it will ACK only Data[1001].
```

c) When the acknowledgment in (b) arrives back at the sender, what data packet is sent?

```
The moment retransmission happens, the sender window is halved and hence, the
```

window becomes 50. After the retransmission, there are 100-3 packets in flight and 96 more dupACKs are on their way to the sender dupACK[1000]/1005, dupACK[1000]/1006.......dupACK[1000]/1100. For each dupACK received, the window size is increased by 1 so when dupACK[1000]/1100 is received, the window size will be 50 + 96 = 146.

Assuming selective retransmission, after the reception of the last dupACK, the window looks something like this......

1001 (retransmitted)	1101	1102	1103	.	.	.	.	.	1101+146-1 = 1246

Now once Data[1001] gets acknowledged, Data[1247] will be sent.

If you do not assume selective retransmission, all the 99 bytes Data[1002] till Data[1100] will be retransmitted. In that case, the window after the reception of the last dupACK will looks like...

1001 (retransmitted)	1002	1003	1004	.	.	.	.	.	1001+146 = 1147

So, once Data[1001] gets acknowledged, Data [1148] will be sent.