CHAPTER 30

PRACTICE SET

Questions

Q30-1. We cannot define the absolute order, but in normal situation we will rank them as follows:

1. SNMP 2. HYYP 3. SMTP 4. VoIP

Q30-3. We cannot define the absolute order, but in normal situation we will rank them as follows:

1. VoIP 2. SNMP 3. HTTP 4. SMTP

Q30-5. Our choice is VoIP. The goal is to make VoIP similar to the telephone conversation we have over the phone using circuit switching.

Q30-7. None of these applications actually fit in this flow class. It is mostly designed for interactive multimedia applications.

Q30-9. FIFO queueing and priority queueing are techniques used for scheduling.

Q30-11. IntServ uses two flow specifications: Rspec and Tspec.

a. Rspec (Resource specification) defines the resources that the flow needs to reserve before starting the communication.

b. Tspec (Traffic specification) defines the traffic specification.

Q30-13. IntServ is called *destination-base* service because the destination host needs to define the type of service required to start the communication.

Q30-15. IntServ is designed for multicast communication. However, unicast is a special case of multicast in which there is only one member in the group.

Q30-17. In IntServ, the resources needs to be reserved before the communication starts. We need to find the resources needed at each router. We need to send Path and collect Resv packets to find the resources the routers need for communication flow.

Q30-19. The traffic conditioner has four components:

a. Meter

b. Marker

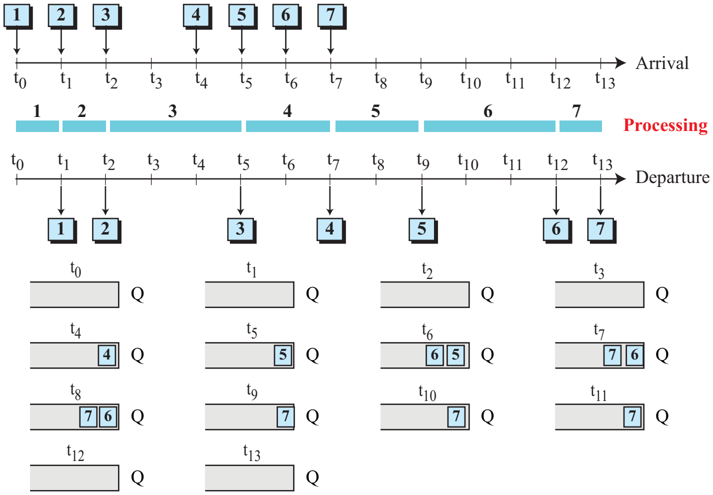
c. Shaper

d. Dropper

Problems

P30-1. We answer each question separately:

a. The following shows the time lines and the contents of the queue:



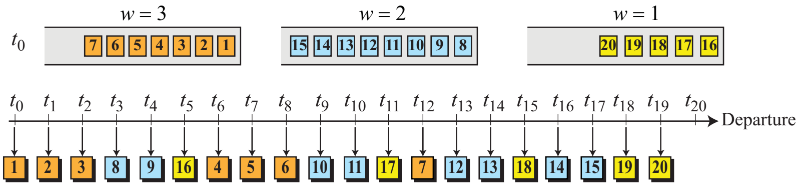
b. The following shows the calculation of the time spent in the router and the relative delay for each packet.

Packets:	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Arrival time	t_0	t_1	t_2	t_4	t_5	t_6	t_7
Departure time	t_1	t_2	t_5	t_7	t_9	t_{12}	t_{13}
Time spent in the router	1	1	3	3	4	6	6
Relative departure delay	–	1	3	2	2	3	1

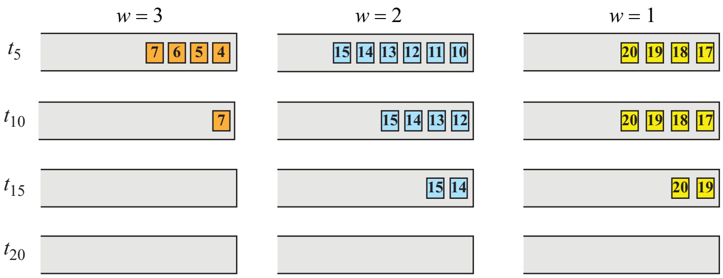
c. Since the relative departure delay between each packet and the previous one is not the same, there is jitter even if all packets have the same propagation delay.

P30-3. We answer each question separately:

a. The following shows the departure time line.


$w = 3$
$w = 2$
$w = 1$
t_0
7 6 5 4 3 2 1
15 14 13 12 11 10 9 8
20 19 18 17 16
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17} t_{18} t_{19} t_{20}
Departure
1 2 3 8 9 16 4 5 6 10 11 17 7 12 13 18 14 15 19 20

b. The following shows the contents of queues at t_0, t_5, t_{10}, t_{15}, and t_{20}.



c. We can calculate the relative delay between packets for the class $w = 3$ as shown below. Since departure delays are different, the packets introduce jitter even if they have the same propagation time.

Packets	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Departure time	t_0	t_1	t_2	t_6	t_7	t_8	t_{12}
Delay with respect to previous packet	–	1	1	4	1	1	4

d. We can calculate the relative delay between packets for the class $w = 2$ as shown below. Since departure delays are different, the packets introduce jitter even if they have the same propagation time.

Packets	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Departure time	t_3	t_4	t_9	t_{10}	t_{13}	t_{14}	t_{16}	t_{17}
Delay with respect to previous packet	–	1	5	1	3	1	2	1

e. We can calculate the relative delay between packets for the class $w = 1$ as shown below. Since departure delays are different, the packets introduce jitter even if they have the same propagation time.

Packets	**16**	**17**	**18**	**19**	**20**
Departure time	t_5	t_{11}	t_{15}	t_{18}	t_{19}
Delay with respect to previous packet	–	6	4	3	1

P30-5.

a. The input during the first minute is (100 gallons/minute) × (12/60 minute) or 20 gallons.

b. The output during the first minutes is (5 gallons/minute) × (1 minute) or 5 gallons.

c. This means that after the first minute, 15 gallons of liquid is left in the bucket. The problem does not mention the input rate after the first minute. If there is no input flow, the bucket would be empty after four minutes.

P30-7. We set the value of n = 1000 bits and the time tick to one second. In each tick of time, only 2 packets will be transmitted, as shown below.

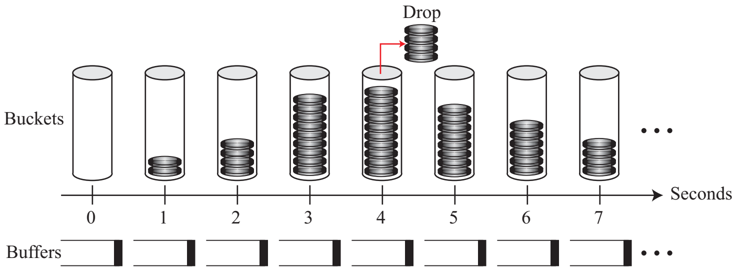
a. The value of the counter is set to 1000; the size of the first packet (400 bits) is less than the value of the counter. The first packet is transmitted. The value of the counter is now decremented to 600.

b. The value of the counter is now 600; the size of the second packet (400 bits) is less than the value of the counter, so it is transmitted. The value of the counter is now decremented to 200.

c. The value of the counter is now 200; the size of the third packet (400 bits) is now larger than the value of the counter, so it cannot be transmitted. It needs to wait for the next tick of time (next second).

d. The process repeats for every second. In each tick of time only two packets can be sent. This means 800 bits per second or the rate of 0.8 Kbps. We have reduced the rate from 4 Kbps to 0.8 Kbps.

P30-9. The following shows that the sender collects credits (tokens) that can be used during the last three seconds. However, the sender loses some of the tokens when the bucket becomes full because it uses very little during the first four seconds and starts using the credits a little bit late.

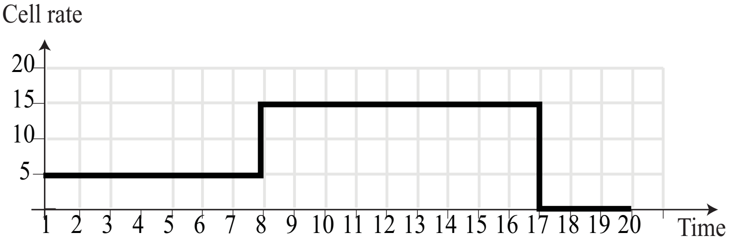


P30-11. To better understand the behavior of the leaky bucket in this problem, we first create a table in each case to show the movement of cells in the system. The flow diagram can follow the table. Although adding, removing, and dropping cells are done for each individual cell, we assume that the cells are first added to the queue, the ones that can be transmitted are removed from the queue, and at the end of each second, the extra cells are dropped (discarded) if the queue is full. Now we create the table and the flow diagrams for each case separately.

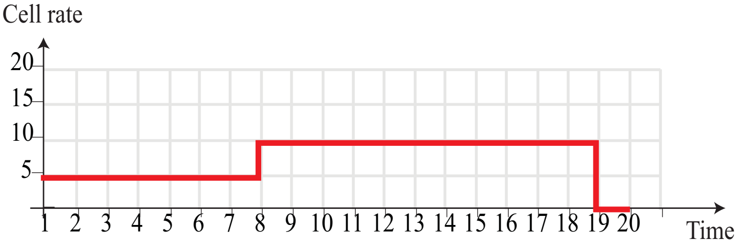
a. The following shows the table for the first case. The table shows that there are occasions when the queue is full and some cells are dropped (t_{12} to t_{16}).

(t_i)	**Previous number of cells in queue**	Number of cells arrived	**Total number of cells in queue**	**Number of cells sent**	**Number of cells left in queue**	**Number of cells dropped**
01	0	5	5	5	0	0
02	0	5	5	5	0	0
03	0	5	5	5	0	0
04	0	5	5	5	0	0
05	0	5	5	5	0	0
06	0	5	5	5	0	0
07	0	5	5	5	0	0
08	0	15	15	10	5	0
09	5	15	20	10	10	0
10	10	15	25	10	15	0
11	15	15	30	10	20	0
12	20	15	35	10	**20**	5
13	20	15	35	10	**20**	5
14	20	15	35	10	**20**	5
15	20	15	35	10	**20**	5
16	20	15	35	10	**20**	5
17	20	0	20	10	10	0
18	10	0	10	10	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0

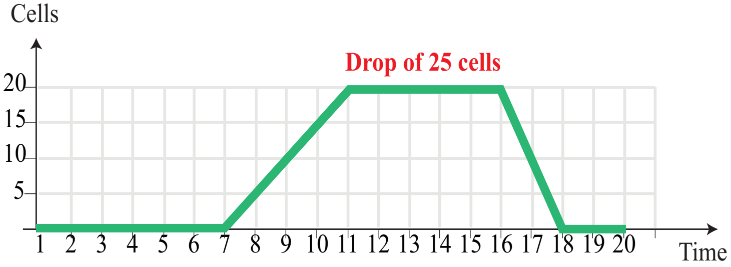
The following shows the flow of cells graphically. The first seven seconds, the ISP rate follows the rate of the customer, but the ISP rate is reduced (to the maximum 10 cells per second. The customer sends 170 cells, but the ISP sends only 145 cells, which means that 25 cells are lost. The third figure shows the number of cells left in the queue in each second. Since some cells are queues and some not, this means that there is uneven delay between the cells (jitter).



Cells arrived at ISP (170)



Cells cent by ISP (145)

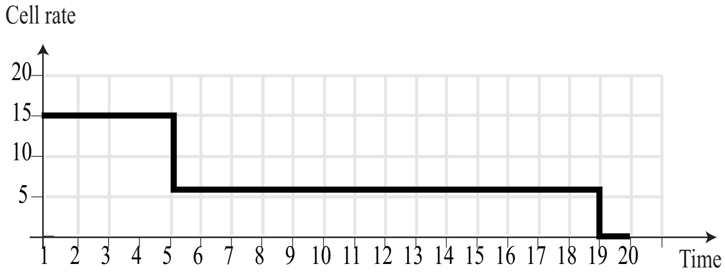


Cells in the queue

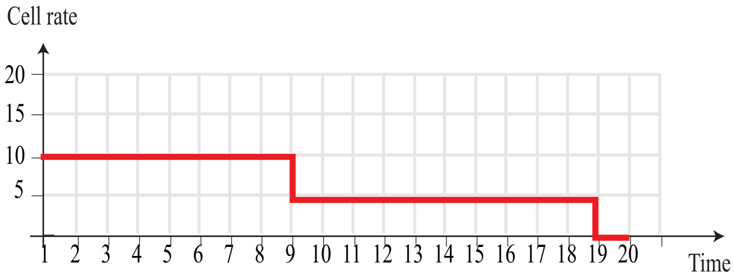
b. The following shows the table for the second case. The table shows that there is no cell lost. Although the customer sends more number of cells in each second than its maximum rate, this only occurs only for four seconds and does not overflow the queue.

(t_i)	**Previous number of cells in queue**	Number of cells arrived	**Total number of cells in queue**	**Number of cells sent**	**Number of cells left in queue**	**Number of cells dropped**
01	0	15	15	10	5	0
02	5	15	20	10	10	0
03	10	15	25	10	15	0
04	15	15	30	10	20	0
05	20	5	25	10	15	0
06	15	5	20	10	10	0
07	10	5	15	10	5	0
08	5	5	10	10	0	0
09	0	5	5	5	0	0
10	0	5	5	5	0	0
11	0	5	5	5	0	0
12	0	5	5	5	0	0
13	0	5	5	5	0	0
14	0	5	5	5	0	0
15	0	5	5	5	0	0
16	0	5	5	5	0	0
17	0	5	5	5	0	0
18	0	5	5	5	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0

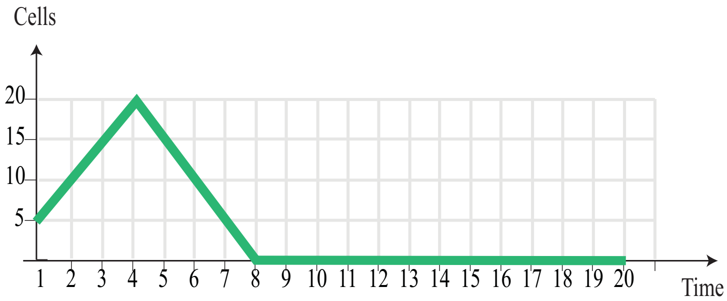
The following shows the flow of cells graphically. The customer sends 130 cells, the ISP also sends 130 cells. There is no cell loss, but there is a delay because some of the cells need to be queued during the first four seconds. The queueing delays all cells after the t_5 (jitter). The queue first becomes full, but it gradually becomes empty.



Cells arrived at ISP (130)



Cells cent by ISP (130)

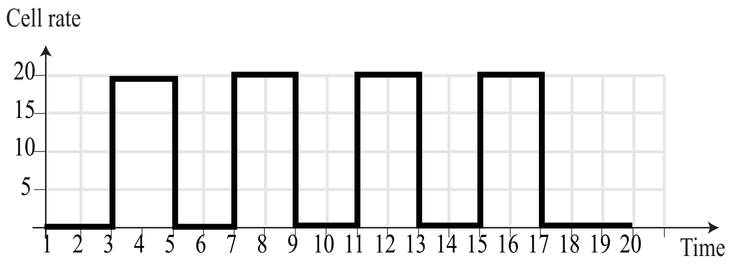


Cells in the queue

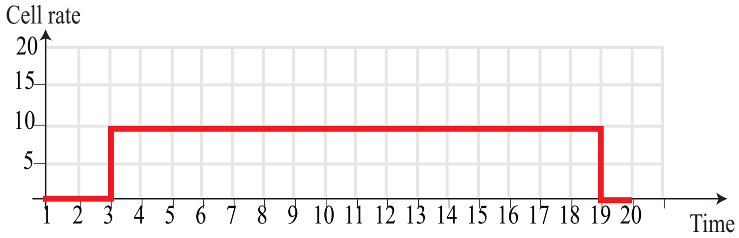
c. The table for the third customer is shown below. The following shows the table for the third case. This is another case in which there is no cell loss.

(t_i)	**Previous number of cells in queue**	Number of cells arrived	**Total number of cells in queue**	**Number of cells sent**	**Number of cells left in queue**	**Number of cells dropped**
01	0	0	0	0	0	0
02	0	0	0	0	0	0
03	0	20	20	10	10	0
04	10	20	30	10	20	0
05	20	0	20	10	10	0
06	10	0	10	10	0	0
07	0	20	20	10	10	0
08	10	20	30	10	20	0
09	20	0	20	10	10	0
10	10	0	10	10	0	0
11	0	20	20	10	10	0
12	10	20	30	10	20	0
13	20	0	20	10	10	0
14	10	0	10	10	0	0
15	0	20	20	10	10	0
16	10	20	30	10	20	0
17	20	0	20	10	10	0
18	10	0	10	10	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0

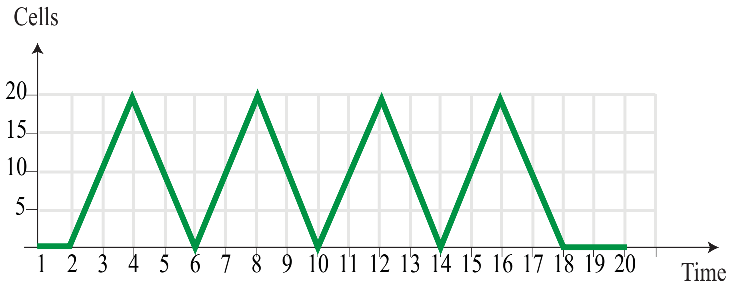
The following shows the flow of cells graphically. This is a very interesting case. Although, some cells are queues, but the cells are departed from the ISP with even delays. If we think that the cells are created at the ISP, we can say that, from t_3 to t_{18}, ISP sends 10 cells per second.



Cells arrived at ISP (160)



Cells cent by ISP (160)



Cells in the queue