

CSE 123 Computer Networks

Homework 4 Solutions

(Total Points: 60)

Instructions:

Take a printout of this template, write answers in the spaces provided below each question, and submit the scanned copies of your homework on Gradescope. You can even write your answers electronically.

Q1 Buffering and Scheduling (3 pt)

0.5 pt for each correct answer.

Answer True or False for the below questions:

(a) FIFO is both a buffer management and scheduling scheme. **False**

FIFO is just a scheduling scheme (i.e., decides which packet should go next). When FIFO is backed with the tail drop policy (i.e., decides which packet to drop next) then we can say that it is both a buffer management and scheduling scheme.

(b) Head of Line Blocking problem cannot occur when input queuing with virtual output queuing is implemented. **True**

At the input interface, we are maintaining a virtual output queue for each output port. Thus, a packet destined to a given output port does not need to keep waiting behind a packet destined to some other output port.

(c) RED drops packets with a probability of 0.85 when the router's average queue length is greater than the maximum threshold value. **False**

When the router's average queue depth is greater than the max threshold value then it will always drop packets. Thus, it will drop with probability 1.

(d) RED uses instantaneous queue length as a metric to detect congestion. **False**

RED uses average queue length as a metric to detect congestion.

(e) RED implements the tail drop policy to drop packets. **Full credits are allotted because the question has insufficient details.**

If the question asked that "ONLY tail drop policy is used" then the answer is False.

Because RED implements a tail drop policy when the queue size is more than the max threshold. But, when queue length is less than the max threshold and more than the min threshold, RED drops packets proportional to the queue length.

(f) RED may drop packets when the average queue length is less than the maximum threshold value.

True

When queue length is less than the max threshold and more than the min threshold, RED drops packets proportional to the queue length.

Q2 Traffic Shaping with Token Bucket (14 pt)

Consider a token bucket shaper with the following configuration:

1. The token bucket depth 'b' is 32KB and is full at time $t=0s$.
2. The rate at which tokens fill the bucket 'r' is 8 KB/s.
3. The outgoing link from the token bucket shaper has a capacity 'R' of 10KB/s.

Assume that a continuous flow (zero time gap between two consecutive packets) of 12 KB packets start arriving at the token bucket shaper at time $t = 0s$. **Please show your work for each of the below questions to get full credits.**

Q2.1 How many tokens remain in the bucket at $t = 6.0s$? (6 pt)

Sol:

Serialization delay for one packet: $12KB/10KBps = 1.2s$

(time, token_bucket_size)

(0s, 32 KB)

(1.2s, 29.6 KB) $\Rightarrow 32 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 29.6 \text{ KB} \Rightarrow$ pkt1 just got transmitted fully

(2.4s, 27.2 KB) $\Rightarrow 29.6 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 27.2 \text{ KB} \Rightarrow$ pkt2 just got transmitted fully

(3.6s, 24.8 KB) $\Rightarrow 27.2 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 24.8 \text{ KB} \Rightarrow$ pkt3 just got transmitted fully

(4.8s, 22.4 KB) $\Rightarrow 24.8 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 22.4 \text{ KB} \Rightarrow$ pkt4 just got transmitted fully

(6.0s, 20 KB) $\Rightarrow 22.4 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 20 \text{ KB} \Rightarrow$ pkt5 just got transmitted fully

Token bucket size at $t = 6.0s$ is **20 KB (20K tokens)**.

Q2.2 If we are indexing the packets starting from 1, which packet would be the first to get queued due to the scarcity of tokens? (4 pt)

Sol:

(pkt_id, time, token_bucket_size_when_pkt_arrives)

(1, 0s, 32 KB) $\Rightarrow$ pkt1 arrives

(2, 1.2s, 29.6 KB) $\Rightarrow 32 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 29.6 \text{ KB} \Rightarrow$ pkt2 arrives; pkt1 transmitted fully

(3, 2.4s, 27.2 KB) $\Rightarrow 29.6 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 27.2 \text{ KB} \Rightarrow$ pkt3 arrives; pkt2 transmitted fully

(4, 3.6s, 24.8 KB) $\Rightarrow 27.2 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 24.8 \text{ KB} \Rightarrow$ pkt4 arrives; pkt3 transmitted fully

(5, 4.8s, 22.4 KB) $\Rightarrow 24.8 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 22.4 \text{ KB} \Rightarrow$ pkt5 arrives; pkt4 transmitted fully

(6, 6.0s, 20 KB) $\Rightarrow 22.4 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 20 \text{ KB} \Rightarrow$ pkt6 arrives; pkt5 transmitted fully

(7, 7.2s, 17.6 KB) $\Rightarrow 20 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 17.6 \text{ KB} \Rightarrow$ pkt7 arrives; pkt6 transmitted fully

(8, 8.4s, 15.2 KB) $\Rightarrow 17.6 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 15.2 \text{ KB} \Rightarrow$ pkt8 arrives; pkt7 transmitted fully

(9, 9.6s, 12.8 KB) $\Rightarrow 15.2 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 12.8 \text{ KB} \Rightarrow$ pkt9 arrives; pkt8 transmitted fully

(10, 10.8s, 10.4 KB) $\Rightarrow 12.8 \text{ KB} - 12 \text{ KB} + 9.6 \text{ KB} = 10.4 \text{ KB} \Rightarrow$ **pkt10 arrives and gets enqueued because there are not enough tokens.**

Q2.3 At what time will the packet that was enqueued in Q2.2 finish transmitting (i.e., all the packet bits are successfully passed to the outgoing link)? (4 pt)

Sol:

(*pkt_id*, *time*, *token_bucket_size_when_pkt_arrives*)

(10, 10.8s, 10.4 KB) $\Rightarrow$ *pkt10* got enqueued.

12 KB - 10.4 KB = 1.6 KB worth of tokens required to transmit *pkt10*.

1.6 KB / 8 KBPS = 0.2 seconds.

So, at $t = 10.8 + 0.2 = 11$ seconds, there are enough tokens to let *pkt10* go.

Transmission time for *pkt10* takes 12 KB/10 = 1.2 seconds.

So, at $t = 11 + 1.2 = 12.2$ seconds *pkt10* is fully transmitted.

Q3 Fair Queuing and Weighted Fair Queuing (12 pt)

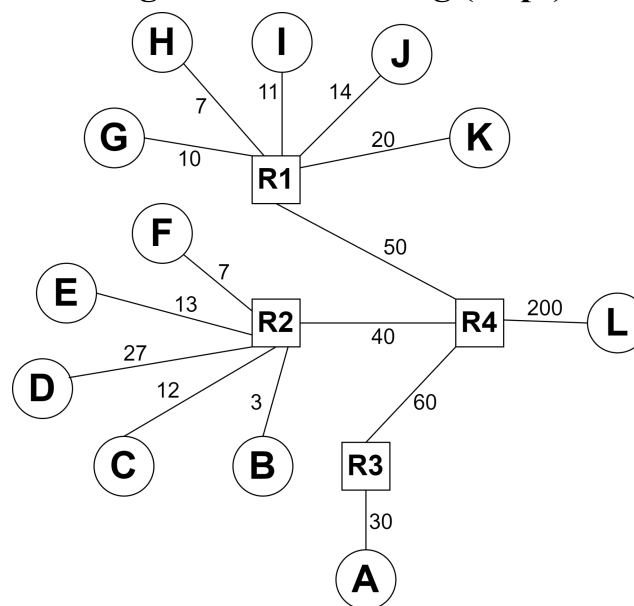


Figure 1: Sample Network Topology

Consider the network topology shown in Figure 1 where circles represent hosts, squares represent routers, and values on edges represent the bandwidth (capacity) of the link in Mbps. Host L is not sending any data on the R4—L link. Every other host is sending a single flow toward node L at a rate equal to the bandwidth of their respective links. Considering these conditions answer the below questions.

(a) What is the maximum throughput in Mbps that can be achieved on R4—L link? (2 pt)

Sol: 50 Mbps + 40 Mbps + 30 Mbps (host A can send max 30 Mbps data) = 120 Mbps

(b) If router R1 is implementing **fair queuing**, then what rate do nodes K and J get on the R1—R4 link? In fair queuing, the rate that a node gets is given by the formula $\min(r_i, f)$ where ' f ' is the fair share value. r_i is the original rate at which host ' i ' is sending its flow. (5 pt)

Sol: 2.5 pt for each correct answer

K = 11 Mbps; J = 11 Mbps

- (c) If router R2 is implementing **weighted fair queuing**, then what rate do nodes E and D get on the R2—R4 link? In weighted fair queuing, the rate that a node gets is given by the formula $\min(r_i, w_i * f)$ where ' f ' is the fair share value. r_i is the original rate at which host ' i ' is sending its flow and w_i is the weight associated with the link where host ' i ' is sending its flow. The link to weight mapping in the topology is {F—R2: 15, E—R2: 3, D—R2: 2, C—R2: 10, B—R2: 20} **(5 pt)**

Sol: 2.5 pt for each correct answer

E= 10.8 Mbps; D = 7.2 Mbps

Q4 TCP Congestion Control (22 pt)

TCP congestion control algorithm discussed in the textbook without Fast Retransmit/Recovery functions as follows:

- (i) At any given time, the sender sends segments worth $\text{minimum}(cwnd, rwnd)$ bytes where $rwnd$ is the receiver's advertised window and $cwnd$ is the sender's congestion window.
- (ii) Sender initializes its $cwnd$ equal to 1 maximum segment size (MSS) and begins the **Slow Start phase**.
- (iii) In the **Slow Start phase**, the sender increments $cwnd$ by 1 MSS for each new ACK that it receives from the sender. Thus, in one RTT new $cwnd$ value would be twice the old $cwnd$ value. In case of a timeout (i.e., ACK for a segment is not received for the timeout duration):
 - (a) The congestion threshold ($ssthresh$) is set equal to half the current $cwnd$ value, and then the $cwnd$ value is reset to 1 MSS.
 - (b) The slow start phase begins again after the above reset operation. The slow start phase continues until a timeout is experienced (in which case it behaves as above) or the $cwnd$ value reaches $ssthresh$.
 - (c) Once the $cwnd$ value is greater than or equal to $ssthresh$, the sender transitions to the **Congestion Avoidance phase**.
- (iv) In the **congestion avoidance phase**, an additive increase multiplicative decrease strategy is followed. Sender increments its $cwnd$ by 1 MSS once ACKs for all segments sent in the previous window are received. Thus, in one RTT $cwnd$ increases by 1 MSS. In case of a timeout, steps (iii) (a) (b) (c) are followed.

The algorithm keeps executing the above steps.

Q4.1 Consider a scenario where a sender and receiver are implementing the aforementioned TCP congestion control algorithm. Assume that at a time ' t ': **(10 pt)**

- (i) Timeout took place at the sender and the $cwnd$ value when the timeout occurred was 32 KB.
- (ii) The maximum segment size at sender and receiver is 2KB.
- (iii) The receiver's advertised window size is 64 KB.
- (iv) The sender always has some data to be sent to the receiver and ACKs are never dropped in the network.

Given the above conditions, how many round trips (RTTs) will it take for the sender's congestion window to return to 32 KB in size? **Please show your work to get full credits. In your work, we expect you to (1) show the $cwnd$ and $ssthresh$ value at the start of each RTT right from the 0 RTT state when the timeout has occurred to the ' x ' RTT state when $cwnd$ reaches 32 KB. (2) mark the start of the slow start or congestion avoidance phase.**

Sol:

1.5 pt for correctly setting $cwnd$ and $ssthresh$ at the slow start phase

2 pt for correctly incrementing $cwnd$ after each RTT in the slow start phase

2 pt for correctly incrementing cwnd after each RTT in congestion avoidance phase

1.5 pt for correctly showing the start of congestion avoidance

3 pt for the final correct answer

(RTT_passed_since_timeout, cwnd, ssthresh)

(0 RTT, 2KB, 16KB) $\Rightarrow$ *ssthresh = cwnd/2; cwnd = 1 MSS; slow start phase begins*

(1 RTT, 4KB, 16KB)

(2 RTT, 8KB, 16KB)

(3 RTT, 16KB, 16KB) $\Rightarrow$ *cwnd $\geq$ ssthresh; congestion avoidance phase begins*

(4 RTT, 18KB, 16KB)

(5 RTT, 20KB, 16KB)

(6 RTT, 22KB, 16KB)

(7 RTT, 24KB, 16KB)

(8 RTT, 26KB, 16KB)

(9 RTT, 28KB, 16KB)

(10 RTT, 30KB, 16KB)

(11 RTT, 32KB, 16KB)

After 11 round trips (RTTs), the sender's congestion window be equal to 32 KB size.

Q4.2 Consider a scenario where a sender and receiver are implementing the aforementioned standard TCP congestion control algorithm. The sender wants to send a 402 KB file to the receiver via an already established TCP connection. At the current time 't' the following is the configuration of the hosts and the TCP connection:

(i) The *ssthresh* value at the sender is 64KB.

(ii) *cwnd* value at the sender is 2 KB and the sender is in the slow start phase.

(ii) The maximum segment size of sender and receiver is 2KB.

(iii) The receiver's window size is 70 KB.

(iv) The sender always has some data to be sent to the receiver and ACKs are never dropped in the network.

Given the above conditions, after how many round trips (RTTs), will the sender successfully receive ACK for the last byte of the 402 KB file that it has sent to the receiver? **Please show your work to get full credits. In your work, we expect you to (1) show the cwnd, ssthresh, and data transferred so far value at the start of each RTT (2) mark the start of the slow start or congestion avoidance phase. (12 pt)**

Sol:

3 pt for correctly incrementing cwnd after each RTT in the slow start phase

3 pt for correctly incrementing cwnd after each RTT in congestion avoidance phase

2 pt for correctly showing the start of congestion avoidance

4 pt for the final correct answer. (2 pts allotted if you made mistake in handling the rwnd bottleneck scenario).

(RTT_passed, cwnd, ssthresh, data_successfully_transferred_so_far)

(0 RTT, 2KB, 64KB, 0KB) $\Rightarrow$ *slow start phase; on each RTT, cwnd_new = 2*cwnd_old*

(1 RTT, 4KB, 64KB, 2KB)

(2 RTT, 8KB, 64KB, 6KB)

(3 RTT, 16KB, 64KB, 14KB)

(4 RTT, 32KB, 64KB, 30 KB)

(5 RTT, 64KB, 64KB, 62 KB) $\Rightarrow$ *cwnd $\geq$ ssthresh; congestion avoidance phase starts, cwnd_new = cwnd_old + 1 MSS*

(6 RTT, 66KB, 64KB, 126 KB)

(7 RTT, 68KB, 64KB, 192 KB)

(8 RTT, 70KB, 64KB, 260 KB)

(9 RTT, 72KB, 64KB, 330 KB) $\Rightarrow cwnd > rwnd$; *cwnd evolves but the sender can only send data equal to rwnd size*

(10 RTT, 74KB, 64KB, 400 KB) $\Rightarrow$ 2 KB worth of file data is still to be sent. It will be sent on 11th RTT.

(11 RTT, 76KB, 64KB, 402 KB)

After 11 RTTs, the sender receives the last ACK for the 402 KB file sent to the receiver.

Q5 Signaling Constraints (9 pt)

- (a) Suppose a channel has a bandwidth of 30 MHz. The channel has a 40 dB signal-to-noise ratio. What is the maximum rate in Mbps (rounded to 1 decimal digit) at which data can be transmitted on this link? **Please show your work. (3 pt)**

Sol:

Applying Shannon's Law,

$$SNR = 10^{(40/10)} = 10000$$

$$C = (30 \times 10^6) \times \log_2(1 + 10000) = (30 \times 10^6) \times 13.288 = 398.6 \text{ Mbps}$$

- (b) Suppose a channel has a bandwidth of 12 MHz. What is the minimum required signal-to-noise ratio (in dB, rounded to one decimal digit) for this channel that can allow delivering data at least at a 108 Mbps rate? **Please show your work. (3 pt)**

Sol:

Applying Shannon's Law,

$$SNR = 2^{(C/B)} - 1 = 2^{(108/12)} - 1 = 511$$

$$SNR(\text{dB}) = 10\log_{10}(SNR) = 27.1 \text{ dB}$$

- (c) Suppose a channel has a bandwidth of 15 MHz and does not introduce any noise. Is it possible to achieve 145 Mbps capacity on this channel using a digital encoding scheme with only 32 levels? **Provide a reason for your claim. (3 pt)**

Sol:

Applying Hartley's Law,

$$C < 2B\log_2(M)$$

$$C < 2 \times (15 \times 10^6) \times \log_2(32)$$

$$C < 2 \times (15 \times 10^6) \times 5$$

$$C < 150 \text{ Mbps}$$

145 Mbps is less than 150 Mbps.

Hence, it is possible to achieve a 145 Mbps channel capacity.