Question 1 (6 points)

Answer:

i.	Transport	Provides process-to-process delivery of the entire message.
ii.	Network	Determines how packets are routed from source to destination.
iii.	Link	Provides control access to the shared communication channel.
iv.	Network	Provides a mechanism to identify end-systems across a network of networks.
v.	Application	Makes it possible for computers with different data representations to communicate.
vi.	Link	Provides node-to-node communications with reliable service.

Question 2 (12 points)

Answer:

(i) The network can support at most 10 users, so each user can get a share of 20 Mb/s ÷ 10 = 2 Mb/s.

Total time = $0.22 + {}^{5.6 \times 10^5}/_{2 \times 10^6} + 0.1 = 0.6\ s$

(ii) No. of packets = ${}^{5.6 \times 10^5}/_{1400 \times 8} = 50$ packets

Time to transmit the last packet from P to the network = $50d_{tx}$

Time spent by the last packet across the network = $3d_{tx} + d_{pg}$

Total time = $53 \times d_{tx} + d_{pg} = 53 \times {}^{(1420 \times 8)}/_{20 \times 10^6} + 0.1 = 0.1301\ s$

Question 3 (10 points)

Answer:

i. False. Link-layer switches only have layers 1 through 2 being implemented.
ii. False. UDP server can use one socket to handle N clients at the same time.
iii. True. As utilization U = d_{tx} / (d_{tx} + RTT), increases in d_{tx} or decreases in d_{pg} (hence RTT) would increase U.
iv. False. GBN uses cumulative acknowledgement instead.
v. False. Each side declares its MSS value, which limits the size of the segments going to be sent by the other side.

Question 4 (10 points)

Answer:

i. Upon receiving ACK[3], the sender shifted its window to [4,5,6,7,8]. Because ACK[4] and ACK[5] are lost, the sender's window will keep at [4,5,6,7,8] even when it received ACK[6] and ACK[7].
After a while the retransmission timers of pkt #4 and pkt #5 will go off and the sender will retransmit packets 4 and 5. These two packets will eventually reach the receiver, which will trigger the receiver to send ACK[4] and ACK[5] again. When these two ACKs are received, the sender's window will finally advance to [8,9,10,11,12].

ii. Yes, it is possible because of pre-mature timeout.
Suppose the sender's window is at [3,4,5,6] and the sender sends packet #3. The receiver receives the packet and returns ACK[3]. However, before ACK[3] arrived, the sender experiences timeout of packet 3 and retransmits packet 3. After that, the sender receives ACK[3] and advances the sender's window to [4,5,6,7]. When the retransmitted packet 3 reached the receiver, the receiver regenerates ACK[3]. Now ACK[3] is outside of current sender's window, which is at [4,5,6,7].

Question 5 (7 points)

Answer:

Segment	Sequence Number	Acknowledgement Number
①	19821	462381
②	462381	21021
③	21021	462441
④	21121	462441
⑤	462441	22321
⑥	462441	22321
⑦	22321	462781

Question 6 (8 points)

Answer:
Given:

- R = 200 x 10^6 b/s
- RTT = 0.16 second
- MSL = 15 seconds

To achieve the best performance, once should have a sender's window that is large enough to sustain the full pipeline. To have the full pipeline, the sender's window should be at least larger than R x RTT / 8 bytes = 4 x 10^6 bytes. The number of bits to represent a number larger than this is log_2 (4 x 10^6) = 21.93 ≈ 22 bits.
As MSL is 15 seconds, the maximum number of bytes transmitted by the system within 15 seconds is R x MSL / 8 bytes = 375 x 10^6 bytes. The number of bits to represent a space larger than this is log_2 (375 x 10^6) = 28.48 ≈ 29 bits.

Question 7 (7 points)

Answer:

i. Without pipelining, each HTTP response must follow the request message before issuing another request. This results in one RTT time plus dtx time of the response message per request. With pipelining, all request messages are sent back-to-back and the response messages are also returned back-to-back. This results in just one RTT time for the requests and the dtx times of all the response messages.
ii. The server identifies the end of each HTTP request message by examining the message structure. For GET requests, it detects the end of the message by the last Blank line. For POST requests, it detects the end of the message by the number of bytes after the Blank line which is indicated by the Content-length header line.